Universidade Federal do Rio de Janeiro (UFRJ)
Processo Seletivo do Grupo de Resposta a Incidentes de Segurança (GRIS)
Tag de Web-Hacking
Avaliador: Breno Castilho
Candidato: Felipe de Jesus

1) O que é o protocolo HTTP e como ele funciona?

O HTTP (Hypertext Transfer Protocol), em português HTTP (Protocolo de transferência de Hipertexto), é basicamente uma forma de conversar entre duas máquinas (cliente-servidor). É um protocolo que permite transferir Hipertexto entre computadores por meio da internet. Funciona através do modelo requisição-resposta entre cliente-servidor. O cliente no navegador, estabelece uma conexão com servidor e envia-lhe uma mensagem de requisição HTTP com uma URL e recebe o site em HTML e outros recursos do servidor como resposta.

2) O que é um Response Code? Cite um exemplo de um programa que você pode fazer com ele?

Response Code é um código de resposta da requisição HTTP, serve para verificar os status dessa resposta e verificar se essa requisição foi concluída corretamente. Response codes são agrupados em cinco classes de respostas: informativas(100-199), sucesso(200-299), redirecionamentos(300-399), erros do cliente(400-499) e erros de servidor(500-599). Por fim, uma aplicação/API que utiliza autenticação e retorna o código 401 quando a requisição não é autorizada é um exemplo de programa que podemos manipular Response Code.

3) O que é um HEADER? Cite um uso INSEGURO desse cabeçalho.

HEADERs são cabeçalhos HTTP, permitem que o cliente e o servidor passem informações adicionais com a solicitação ou resposta HTTP. O campo de cabeçalho fica localizado da segunda linha para baixo na requisição ou resposta HTTP. Podem ser classificados de acordo com seus contextos, em cabeçalhos genéricos (utilizados em request/response), de solicitação (contém mais informação sobre o recurso à ser pedido ou do próprio cliente), de resposta (informação adicional sobre a solicitação, como a localização) e de entidade (informações sobre o conteúdo da entidade). Também podem ser classificados de acordo como são manipulados por proxies em End-to-end e Hop-by-hop. Alguns exemplos de headers são Host, User-Agent, Server, Connection, Server, Content-Type e Date.

4) O que é um método HTTP? Explique o funcionamento do método POST, o funcionamento do método GET. Explique qual é considerado mais seguro e por que.

Um método HTTP é basicamente uma forma de indicar a ação a ser executada para um determinado recurso. Ele determina o que o servidor deve fazer com a URL fornecida no momento da requisição de algum recurso. Os métodos HTTP por causa de sua semântica também podem ser chamados de verbos HTTP. Existem oito métodos HTTP, sendo eles GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE e PATCH, dentre estes os mais comuns em requisições HTTP são os métodos GET e POST. Para entender o funcionamento desses métodos, precisamos entender o funcionamento de uma requisição HTTP, na primeira linha da requisição temos três partes separadas por espaço: o método, o caminho do recurso a ser buscado no servidor e a versão do protocolo HTTP, e logo abaixo nas próximas linhas temos os headers (informações adicionais para o servidor) opcionais. O servidor ao receber a solicitação, envia uma resposta que é semelhante ao código da requisição, na primeira linha temos a versão do protocolo HTTP, um código de status e uma palavra que descreve o código de status, além disso esta mensagem possui linhas de cabeçalho e um corpo da mensagem (opcional). É basicamente dessa forma que funcionam o request-response no protocolo HTTP.

O método GET serve para solicitar a visualização de algum recurso do servidor que pode ser, por exemplo, um arquivo html representado por uma página no navegador. Abaixo temos um exemplo da utilização do método GET:

```
GET /index.html HTTP/1.1
Host: www.exemplo.com
```

E logo em seguida o servidor envia uma resposta (seguida com o texto da página solicitada):

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.27 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

O método POST por sua vez, serve para enviar dados para serem processados por um(a) página/recurso específico no servidor, por exemplo, enviar dados para um formulário HTTP. Abaixo, temos um exemplo da utilização deste método:

```
POST /index.html HTTP/1.0
Accept: text/html
If-modified-since: Sat, 29 Oct 1999 19:43:31 GMT
Content-Type: application/x-www-form-urlencoded
Content-Length: 41

Nome=NomePessoa&Idade=99&Curso=Computacao
```

Em relação à segurança destes dois métodos, temos que o GET envia os dados no cabeçalho da requisição, por isso eles podem ser vistos na URL. Como exemplo, temos que ao tentar logar em uma aplicação web em desenvolvimento, obtemos o seguinte link:

```
127.0.1.1:5000/logar?email=armando%40gmail.com&senha=123456
```

Nesse link, podemos perceber o e-mail (armando@gmail.com) e a senha(123456) do cliente ao tentar logar em algum recurso do servidor. Já no método POST, os dados são enviados no corpo da requisição HTTP (como visto anteriormente no exemplo de POST), não aparecendo, portanto, no URL:

```
127.0.1.1:5000/logar
```

Analisando esses dois métodos separadamente percebemos que o mais seguro é o POST, pois possibilita o envio dos dados de forma encapsulada na requisição HTTP, o que esconde as informações importantes na URL de pessoas comuns ou mal intencionadas.

5) O que é cache e como ele funciona? Cite os principais HEADERS de Request e Response responsáveis pelo controle de Cache.

Cache é basicamente o armazenamento temporário no disco de dados de navegação para uso futuro, como por exemplo cópias de páginas acessadas na web. Sendo assim, podemos acessar a mesma página na Web com mais rapidez, pois suas informações já foram armazenadas em cache desde o último acesso. Os principais headers de Request e Response para controle de Cache são: Expires, Cache-Control, Etag, Last-Modified, Content-Length e Vary.

6) O que é Cookie? Qual é o principal ataque relacionado a ele?

Cookies são pequenos fragmentos ou pacotes de dados criados por sites visitados e que são salvos no computador do usuário. Eles são re-enviados ao site junto à requisição em um novo acesso futuro. São usados para manter informações sobre o usuário no site, como carrinho de compras, lista de produtos e preferências de navegação. Eles podem servir principalmente para gerenciamento de sessão,

personalização (preferências, temas e outras configurações) e rastreamento (análise de comportamento e análise do usuário). Como exemplo da utilização de cookies temos que ao logar com nosso e-mail e senha no Facebook, sair e retornar para a página de login, ao clicar no campo e-mail, vemos que nosso e-mail aparece novamente e, portanto, foi salvo pelo navegador. O principal ataque relacionado aos cookies é o Session hijacking (Sequestro de Sessão).

7) O que é OWASP-Top-Ten?

O top 10 da OWASP (Open Web Application Security Project) representam os 10 riscos principais para o desenvolvimento de aplicações Web. É um documento padrão de conscientização sobre segurança dos aplicativos Web para os desenvolvedores. Os 10 riscos principais de acordo com essa fundação são: “Injection”, “Broken authentication”, “Sensitive Data exposure”, “XML External Entities (XXE)”, “Broken Access Control”, “Security Misconfiguration”, “Cross-Site Scripting XSS”, “Insecure Deserialization”, “Using components with Known Vulnerabilities” e “Insufficient logging and Monitoring”.

8) O que é Recon e por que ela é importante?

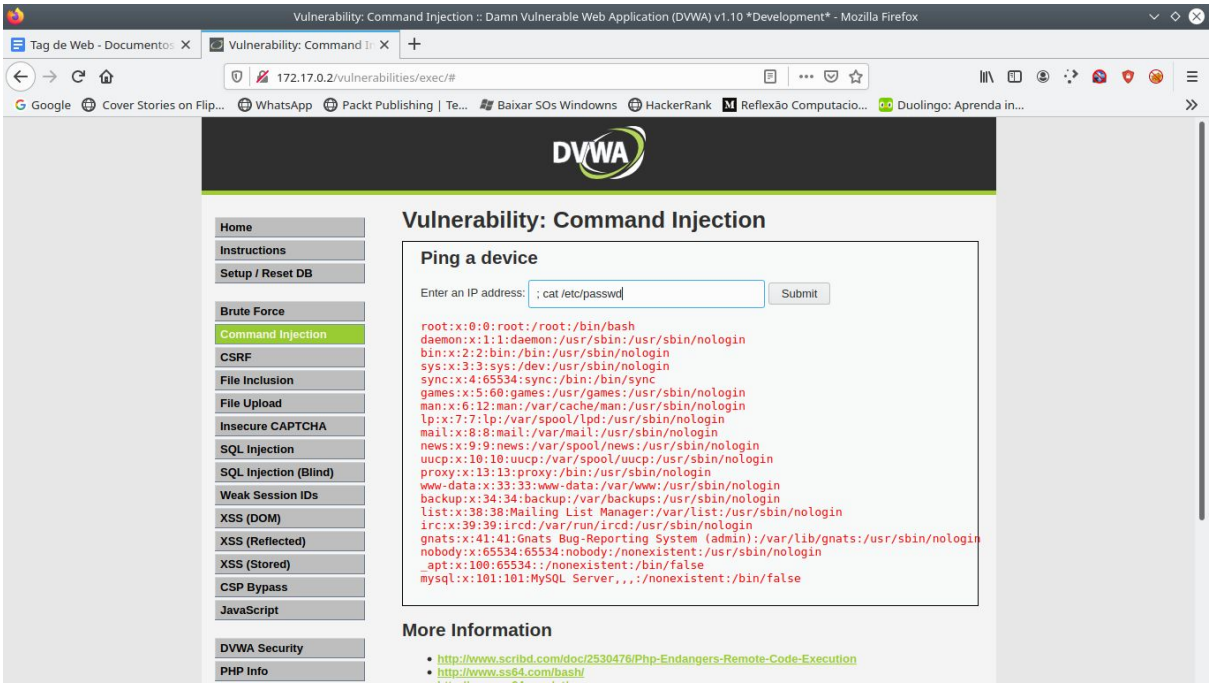
Recon é a fase de reconhecimento da aplicação Web. Esta fase fornecerá informações detalhadas sobre os recursos (páginas, arquivos, pastas, links, imagens, etc) que compõem o programa. O Recon é muito importante para a fase de exploração de falhas (exploitation), pois é com esses dados que iremos interagir na busca por vulnerabilidades da aplicação. Por isso, é de extrema importância descobrir todo e qualquer recurso que a aplicação interage. Além das informações citadas anteriormente, podemos coletar o endereço IP, a topologia da rede, os dispositivos na rede, as tecnologias em uso, versões dos pacotes, etc.

9) Command Injection (SO-Injection)

a) O que é Command Injection?

Significa Injeção de Comando e é uma técnica utilizada para executar comandos em um dispositivo com sistema operacional por meio de uma aplicação Web vulnerável.

b) Mostre um exemplo de Command Injection (PoC da exploração)



10) SQL INJECTION

a) O que é SQL injection?

É a injeção de código SQL em formulários de sites que interagem com procedimentos SQL. Estes comandos inseridos alteram o SQL existente que deveria executar uma instrução pré-definida, de forma que execute um novo comando. Essa alteração compromete a segurança do sistema, pois pode expor dados escondidos, sobrescrever dados valiosos e existentes e rodar comandos de sistema perigosos no servidor.

b) O que é Union Based Attack?

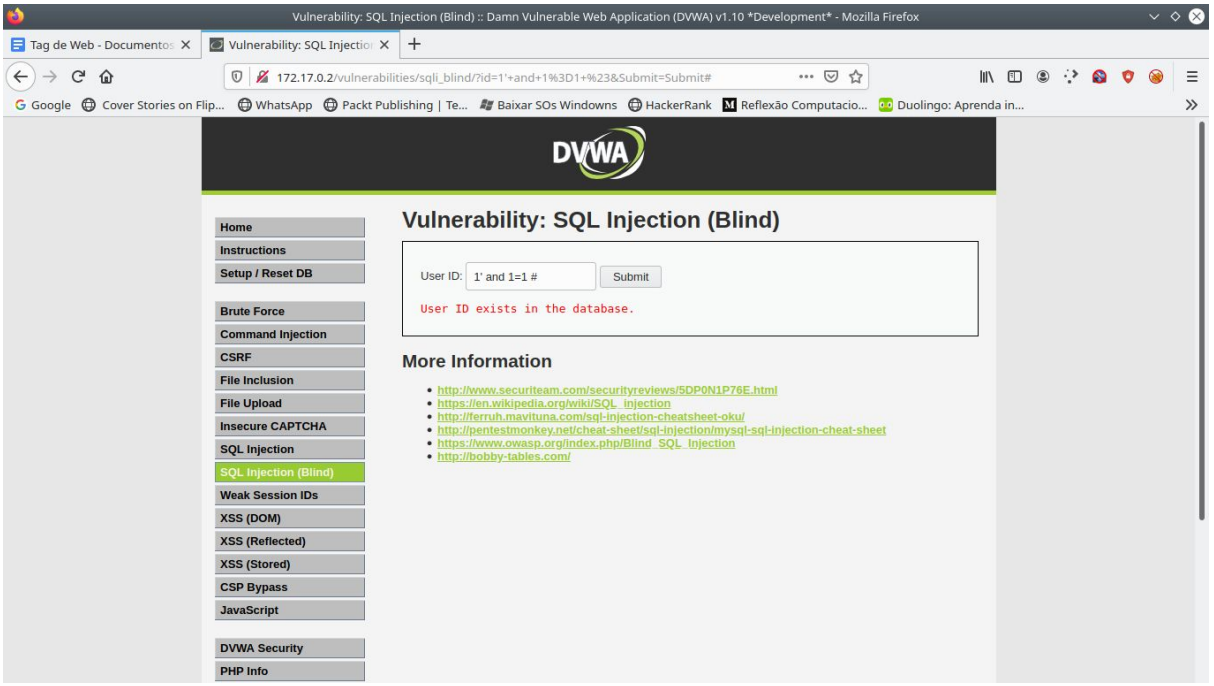
É um ataque que permite ao invasor extrair informações de outras tabelas do banco de dados através do operador UNION. Este operador só pode ser usado se consultas originais/novas forem da mesma estrutura em relação ao número e tipo de dados das colunas.

c) O que é Blind-SQL-I?

São utilizados quando os ataques por meio de SQL Injection normais retornam mensagens de erros. O Blind SQL Injection não retorna nada, com isso

precisamos interpretar valores de lógica booleana para que este tipo de ataque funcione.

d) Mostre um exemplo de Blind SQL-Injection (PoC da exploração)



11) XSS

a) O que é XSS?

Cross-site scripting (XSS) é uma vulnerabilidade, normalmente encontrada em aplicações Web, em que os atacantes injetam códigos/scripts com intenções maliciosas em sites com falhas de programação. Os códigos/scripts geralmente são feitos em JavaScript.

b) Quais são os tipos de XSS? Explique-os.

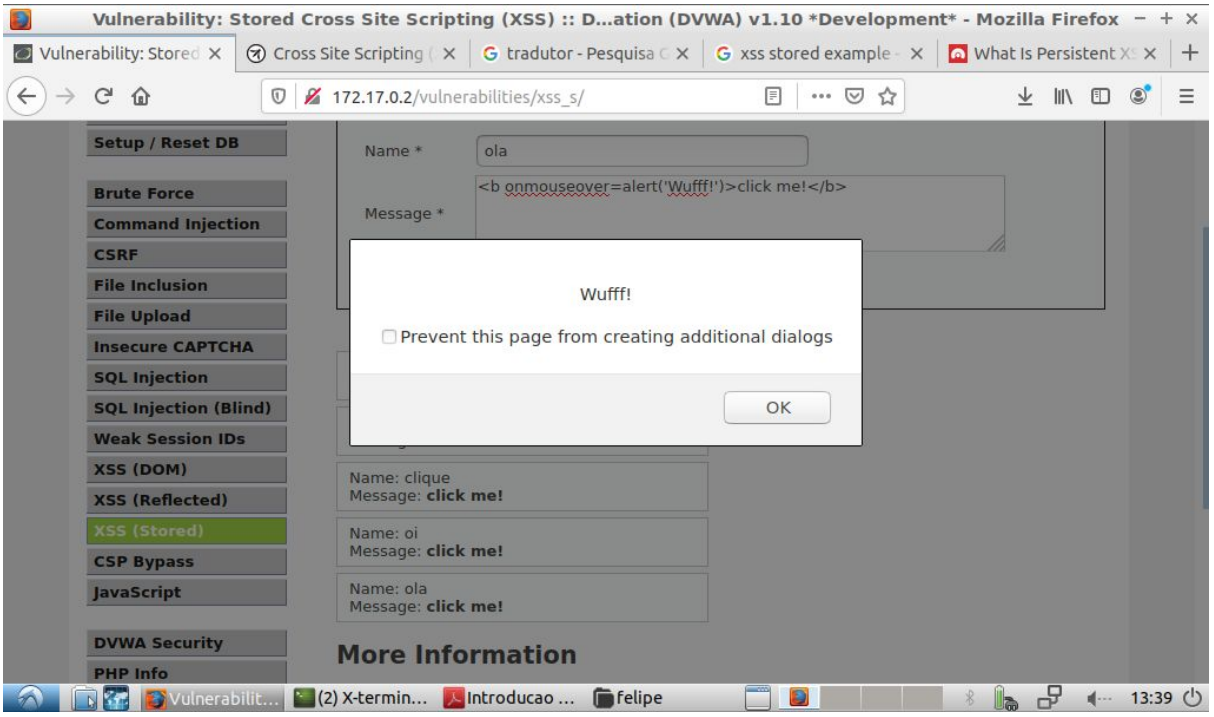
Podem ser divididos em três tipos: stored, reflected e dom-xss.

Ataques xss stored (permanente) são aqueles em que o script é injetado e fica permanentemente armazenado no servidor de destino, como por exemplo, em um banco de dados fórum de mensagens, campo de comentários, etc. Eles são exibidos em páginas “normais” e por estarem escondidos na página, acabam roubando informações dos usuários.

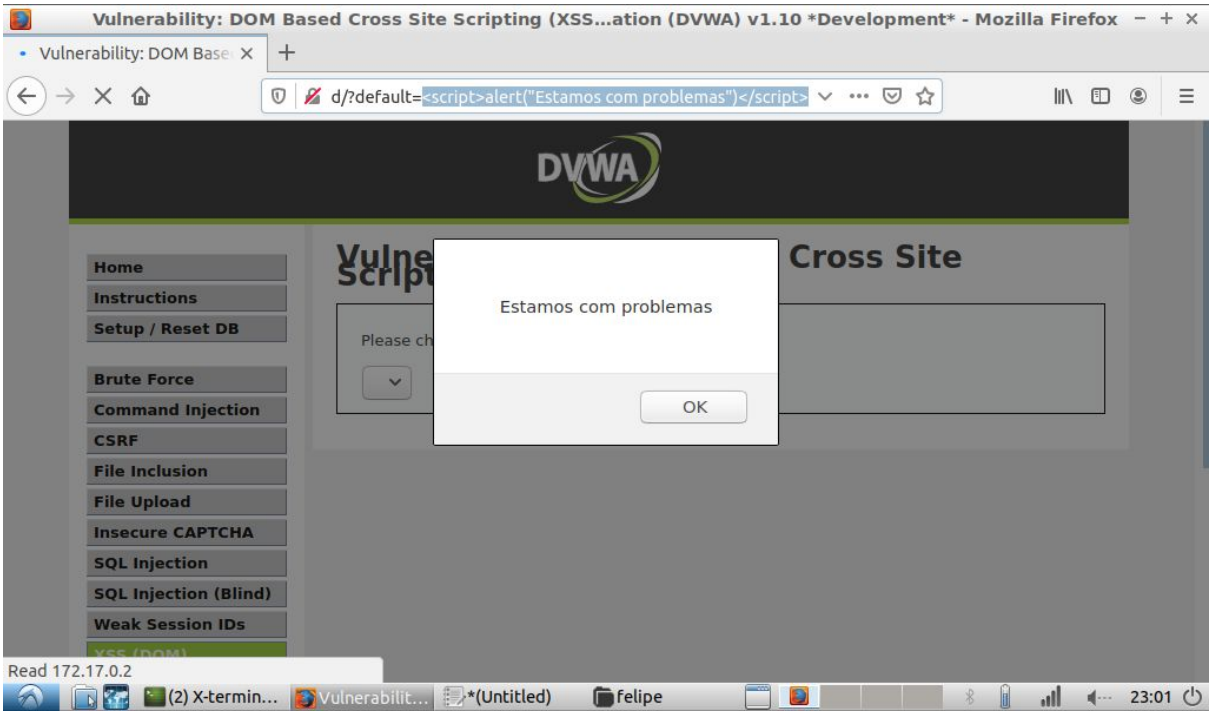
Ataques xss reflected (não permanentes) são aqueles em que o script injetado é refletido no servidor da web, como em uma mensagem de erro. Ataques desse tipo são entregues às vítimas por meio de engenharia social, por exemplo em uma mensagem de erro ou em um resultado de busca. Uma forma comum é um link distribuído por esquemas de phishing. O usuário ao clicar no link ativa o servidor e o script é refletido e executado no navegador da vítima.

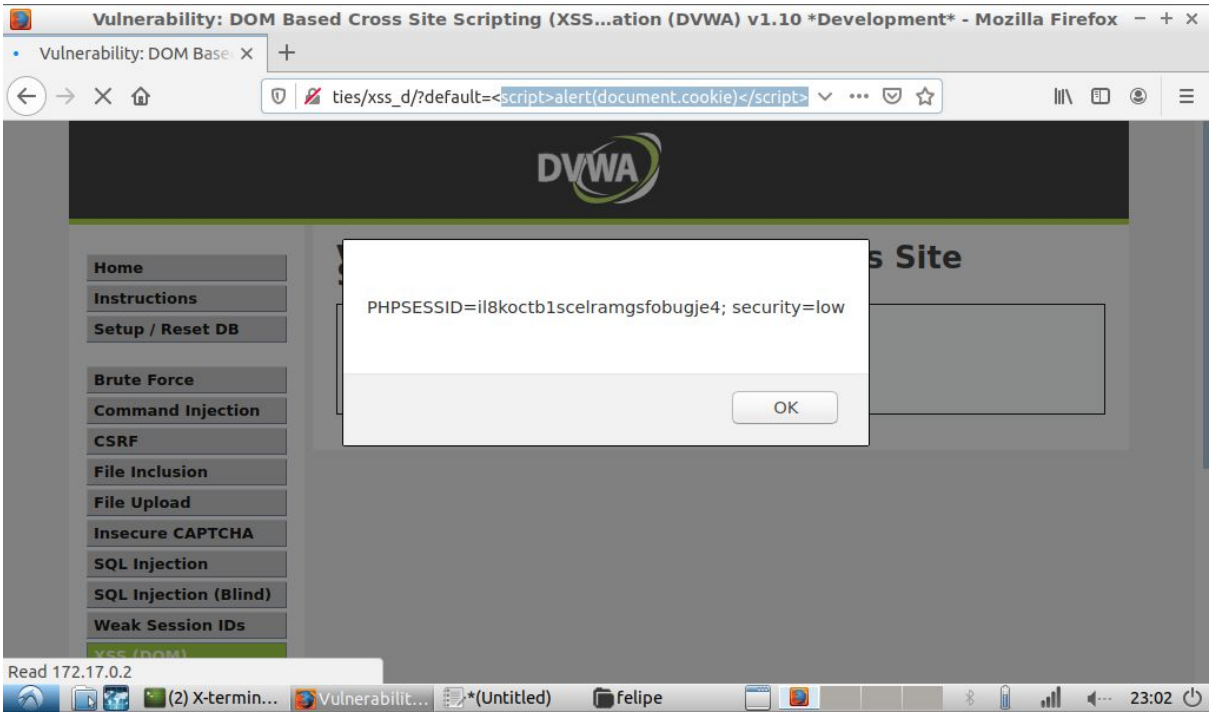
Ataques DOM-xss exploram o Document Object Model (DOM), interface para a leitura de HTML e XML no navegador. O script altera as propriedades destes tipos de extensões, o que pode implicar em roubo de informações confidenciais em um cookie, ataques de phishing e diversas outras ações danosas à segurança do usuário.

c) Mostre um exemplo de um XSS Stored (PoC da exploração)



d) Mostre um exemplo de um DOM-XSS (PoC da exploração)





12) LFI, RFI e Path Traversal

a) O que é LFI?

Local File Inclusion(LFI) é o processo de inclusão de arquivos locais no servidor em aplicações Web, através do processo de exploração de meios de inclusão vulneráveis. Essa falha ocorre, quando por meio do método GET na URL, inserimos um caminho para um arquivo no servidor, porém esse caminho não é

avaliado corretamente pela aplicação Web. Dessa forma, LFI é perigoso, pois arquivos do sistema podem ser acessados e abre brechas para a execução de códigos maliciosos no servidor e ataques do tipo XSS, por exemplo.

b) O que é RFI?

Remote File Inclusion(RFI) é bem semelhante ao LFI, porém arquivos remotos podem ser acessados pela GET na URL, sendo assim, estes arquivos podem ser injetados no servidor. O que deixa o servidor vulnerável à ataques do tipo XSS e por execução de códigos remotos mal intencionados.

c) O que é Path Traversal?

É Utilizado para obter acesso não autorizado a arquivos e diretórios no servidor. Esse ataque é resultado da falta de validações de entrada de usuário na aplicação web. Path Traversal podem ser ataques que exploram falhas no servidor web e ataques que exploram vulnerabilidades no código da aplicação. A URL mostra um arquivo específico do servidor na aplicação web, com isso e mexendo na requisição GET da mesma forma que no LFI, podemos encontrar algum arquivo específico no servidor, como por exemplo o arquivo “passwd” localizado na pasta “/etc”. Para encontrar esta pasta podemos usar “../” diversas vezes, então como exemplo podemos ter um site, www.exemplo.com que mostra o arquivo “exemplo1.php” na tela inicial e supondo que este arquivo pode acessar o sistema inteiro, temos a seguinte requisição HTTP para acessar o arquivo “passwd”:

GET http://www.exemplo.com/exemplo1.php?file=../../../../../etc/passwd HTTP/1.1

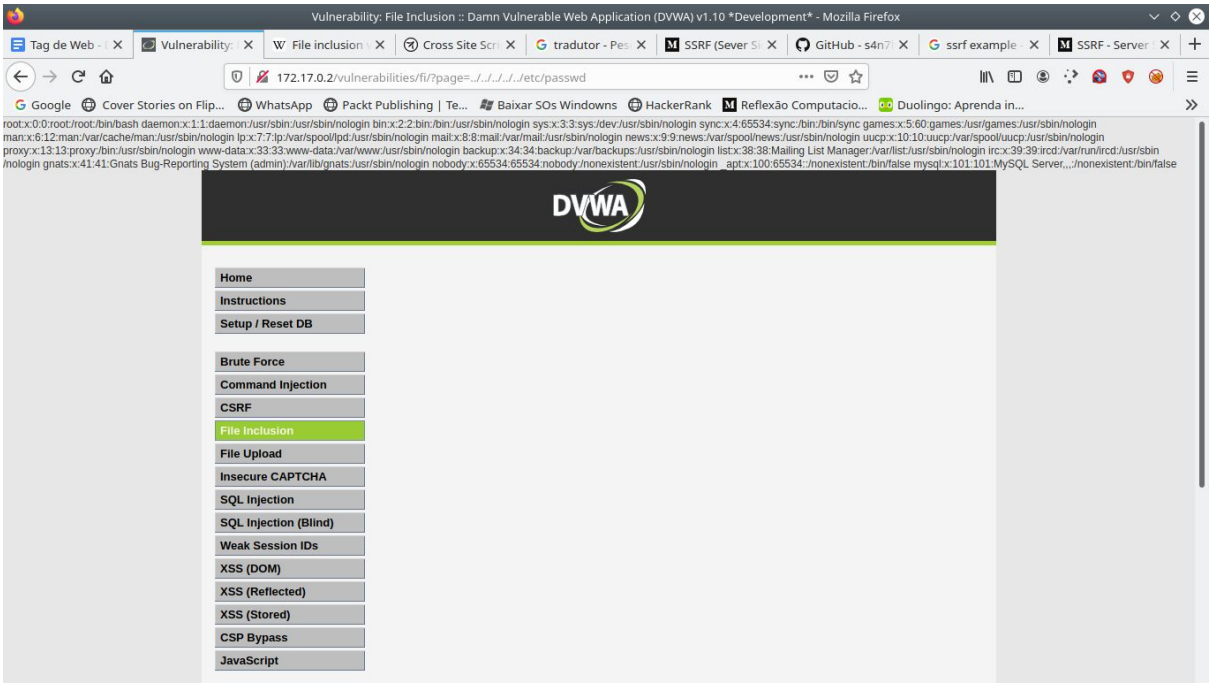
Sendo assim, conseguimos acessar o arquivo “passwd” com esse ataque. Dessa forma, exploramos uma falha e conseguimos exibir o conteúdo de um arquivo não autorizado do servidor web. É basicamente dessa forma que funciona esse ataque.

d) Como aliar Path Traversal e LFI

LFI serve para acessar arquivos no servidor através da URL, pois não possui uma boa implementação para avaliar a URL. Path Traversal funciona basicamente da mesma forma que LFI, porém para encontrar o arquivo que queremos acessar temos que usar ../ até chegar no diretório do arquivo necessário. Com ../ e acesso

de arquivos no servidor web através de uma vulnerabilidade conhecida, temos a união de LFI e Path Traversal.

e) Mostre um exemplo de LFI utilizando a contaminação de LOGS (PoC da exploração)



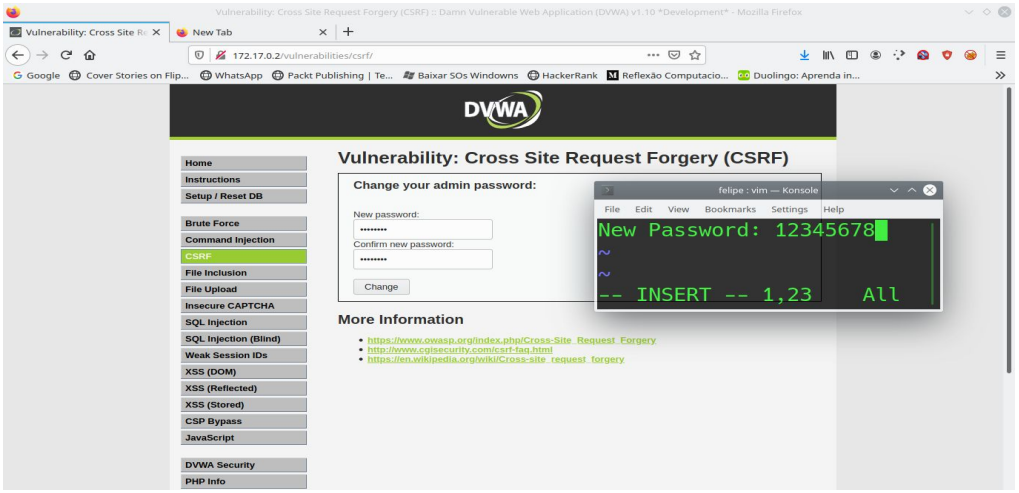
13) CSRF e SSRF

a) O que e CSRF?

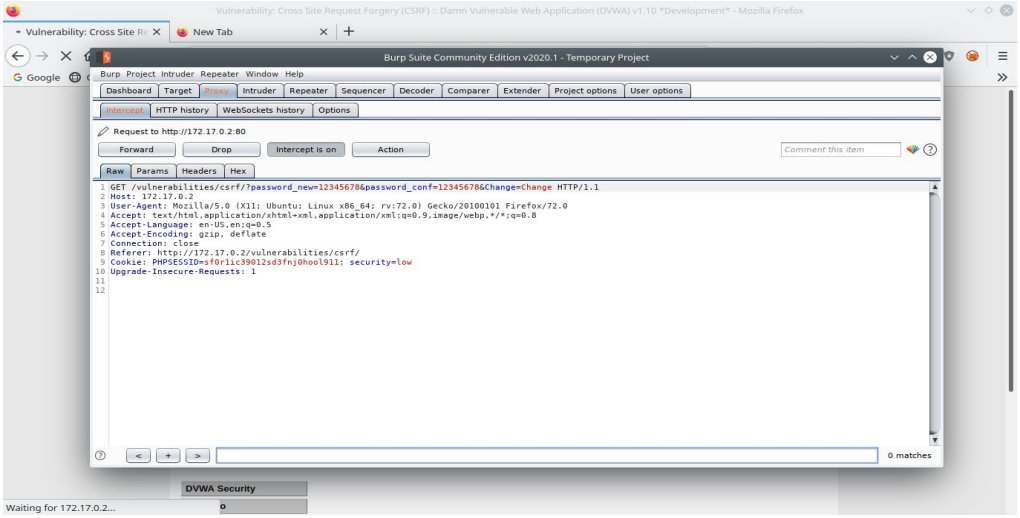
O Cross-Site Request Forgery (CSRF), em português fica Falsificação de Solicitações entre sites, é um tipo de exploit malicioso de um website, em que o atacante explora a confiança do navegador no usuário executando comandos não permitidos na aplicação sem consentimento do usuário. Esse ataque funciona através da inclusão de scripts ou links em páginas web.

b) Mostre um exemplo de CSRF (PoC da exploração)

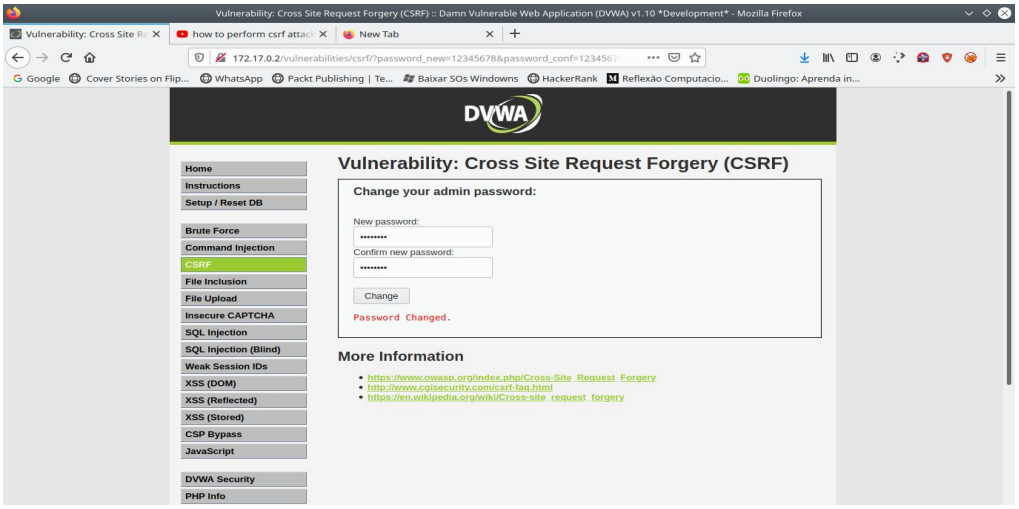
Neste exemplo, inicialmente vamos mudar o password para 12345678 na parte de CSRF em DVWA.



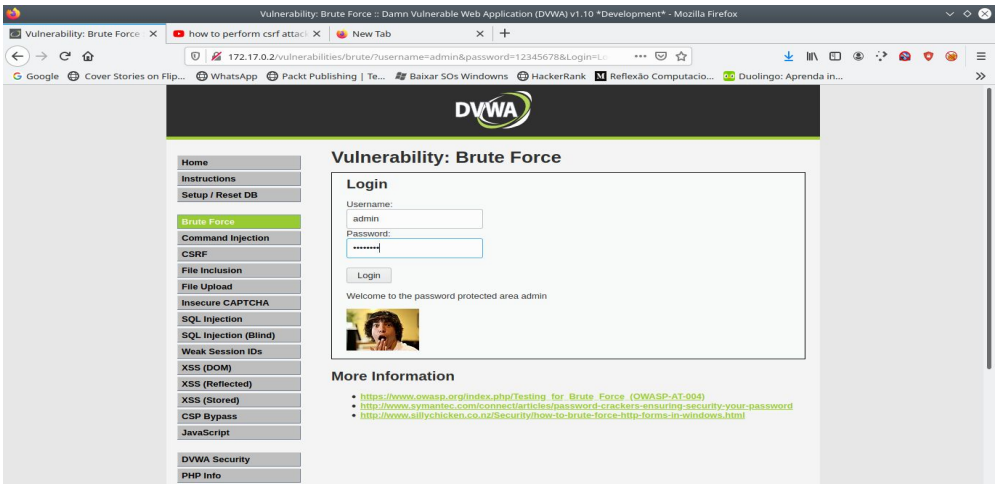
Verificando o código da requisição dessa alteração de senha, temos:



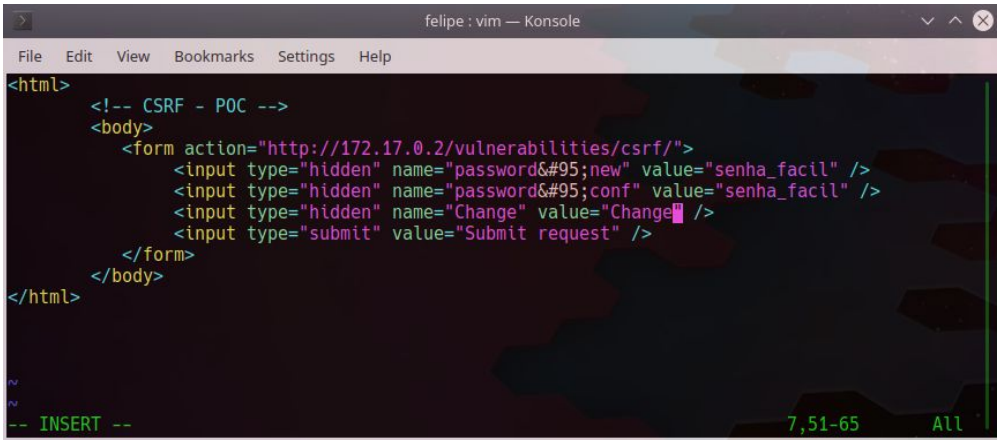
Terminando de alterar o password ...



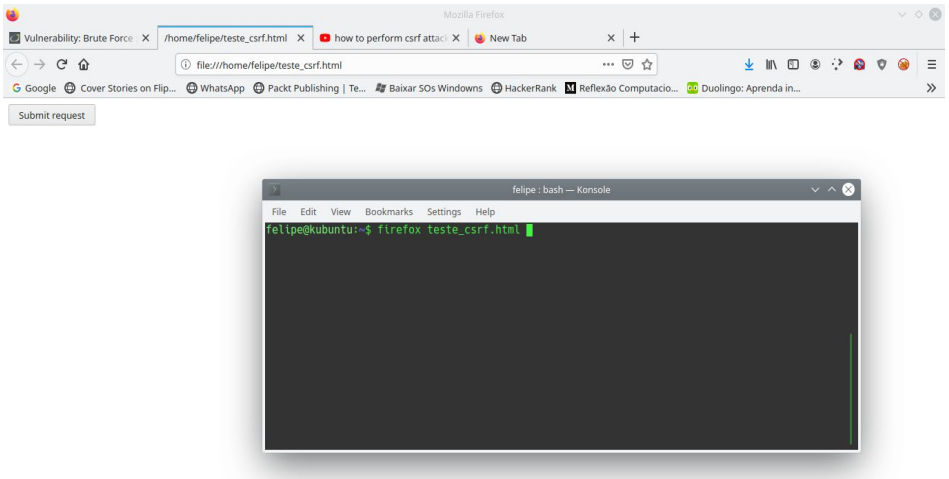
Logando com sucesso na parte de Brute force com o admin e a senha alterada, temos:



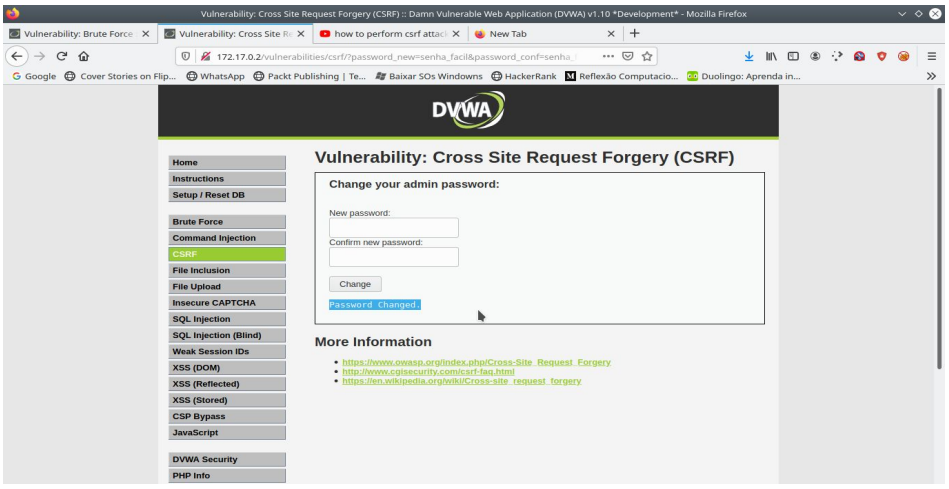
Código malicioso para alterar a senha do sistema em CSRF:



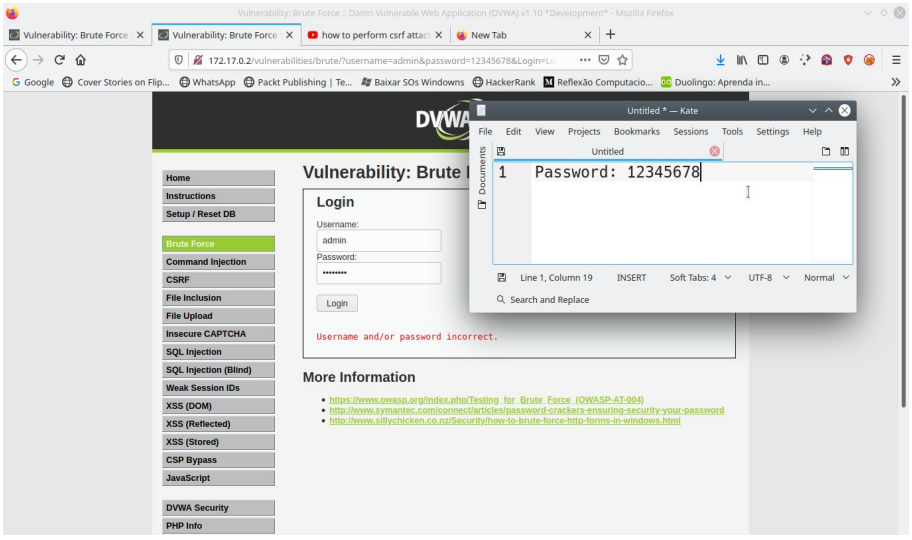
Entrando na página desse código malicioso e clicando no botão “Submit Request” ...



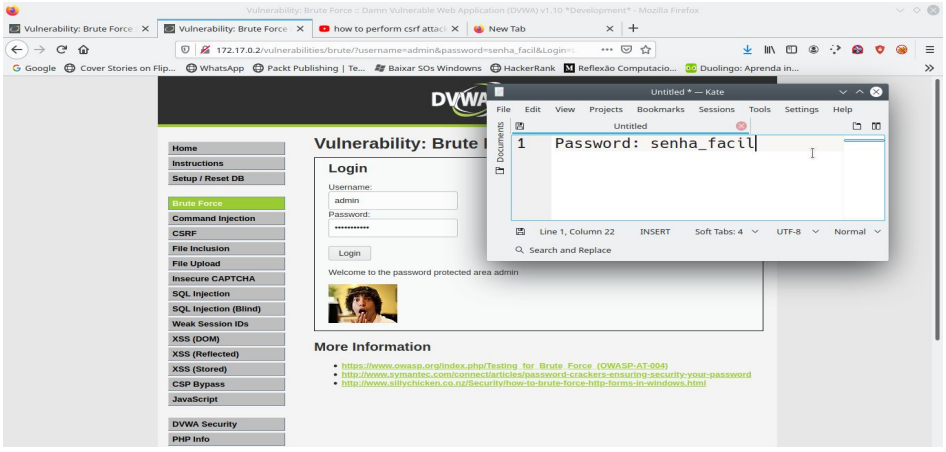
Através desse código conseguimos fazer uma Falsificação de solicitação entre sites e assim alteramos a senha do sistema:



Testando a senha antiga, obtemos um erro:



E portanto a nova senha funciona, como descrita na imagem abaixo:



c) O que e SSRF?

A Falsificação de Solicitação do lado do Servidor (SSRF) é uma vulnerabilidade na qual um atacante força um servidor a executar solicitações em seu nome.

d) Mostre um exemplo de SSRF (PoC da exploração)

e) Como evitar ataques de CSRF?

Algumas dicas para prevenção de ataques CSRF são: Não usar versões modificadas de navegadores sem conhecimento da fonte e objetivo; fazer logout de contas e aplicações; evitar uso da opção “lembrar meus dados” e evitar clicar em links não confiáveis. E para as aplicações, temos que é bom limitar o tempo de vida de cookies de sessão; verificar cabeçalho HTTP Referer de requisições e exigir que o cliente forneça dados de autenticação na solicitação HTTP mesmo se utilizado para realizar qualquer operação com implicações de segurança.