1) HTTP significa **H**yper**T**ext **T**ransfer **P**rotocol, e é um protocolo de transferência para que seja inserido URL de um website, para que seja possível acessá-lo recebendo os dados existentes.É a comunicação base da Internet para sites e afins carregarem hiperlinks. HTTP é um protocolo baseado em texto sem conexão.

   O HTTP funciona com uma nova conexão sendo feita para cada solicitação de acesso. Se a URL pertencer a um domínio próprio, o navegador primeiro se conecta a um servidor e resgatará o endereço IP do servidor; o navegador se conecta ao servidor e envia uma solicitação HTTP para a página da web desejada (que, neste exemplo, é o seu site); o servidor recebe a solicitação e verifica a página desejada. Se a página existir, o servidor a mostrará. Se o servidor não conseguir encontrar a página solicitada, ele enviará uma mensagem de erro o navegador, então, recebe a página de volta e a conexão é fechada; quando o navegador terminar de carregar todos os elementos, a página será carregada na janela do navegador.

2) São códigos que indicam o que ocorreu com a requisição HTTP. São os seguintes: ▪ 1xx -- informativo ▪ 2xx -- sucesso □ principais: 200(success), 201(created), 202(accepted) ▪ 3xx -- redirecionamento □ principais: 304(Not Modified), 302(Moved temporarily), 301(Moved Permanently) ▪ 4xx -- erro cliente □ principais: 400(Bad Request), 401(Unauthorized),403(Forbidden), 404(Not Found), 405(Method Not Allowed),407(Proxy Authentication Required) ▪ 5xx -- outros erros □ principais: 500 (Internal Server), 511(Network Authentication Required). Exemplo: QRCode.
3) HTTP Headers são componentes de uma seção das mensagens do HTTP. Eles definem parâmetros de operação de uma transação deste protocolo.

   Campos de cabeçalho HTTP são componentes da seção de cabeçalho das mensagens de requisição e resposta no Protocolo de Transferência de Hipertexto (HTTP). Eles definem os parâmetros de operação de uma transação HTTP. Os campos de cabeçalho são transmitidos depois da linha de requisição ou resposta, a qual é a primeira linha de uma mensagem. A ausência de um header seguro pode deixar o sistema vulnerável a ataques xss.

4) São métodos de requisição responsáveis por indicar a ação a ser executada para um recurso dado. Elas podem ser do tipo safe, idempotent ou cacheable. **Exemplos de métodos HTTP: GET - HEAD - POST - PUT - DELETE - OPTIONS - TRACE.**

   POST - É um método para realizar ações, em que os parâmetros podem ser enviados tanto pelo url quanto pelo corpo da requisição, o que o torna mais seguro que o método GET, por evitar que a informação da requisição fique salva no histórico. **GET-** É um método para retornar algum recurso pedido, que utiliza a url para enviar parâmetros, tornando-o mais inseguro para informações.

5) Cache é uma área de armazenamento temporário onde os dados do website são armazenados. Por armazenar esses dados, o navegador web pode melhorar a performance/velocidade carregando dados direto do disco, ao invés da internet, caso esses dados sejam necessários. Os principais HEADERS de Request e Response responsáveis pelo controle de cache são: **Cache-Control**, **ETag**, **Pragma** e **Expires**.

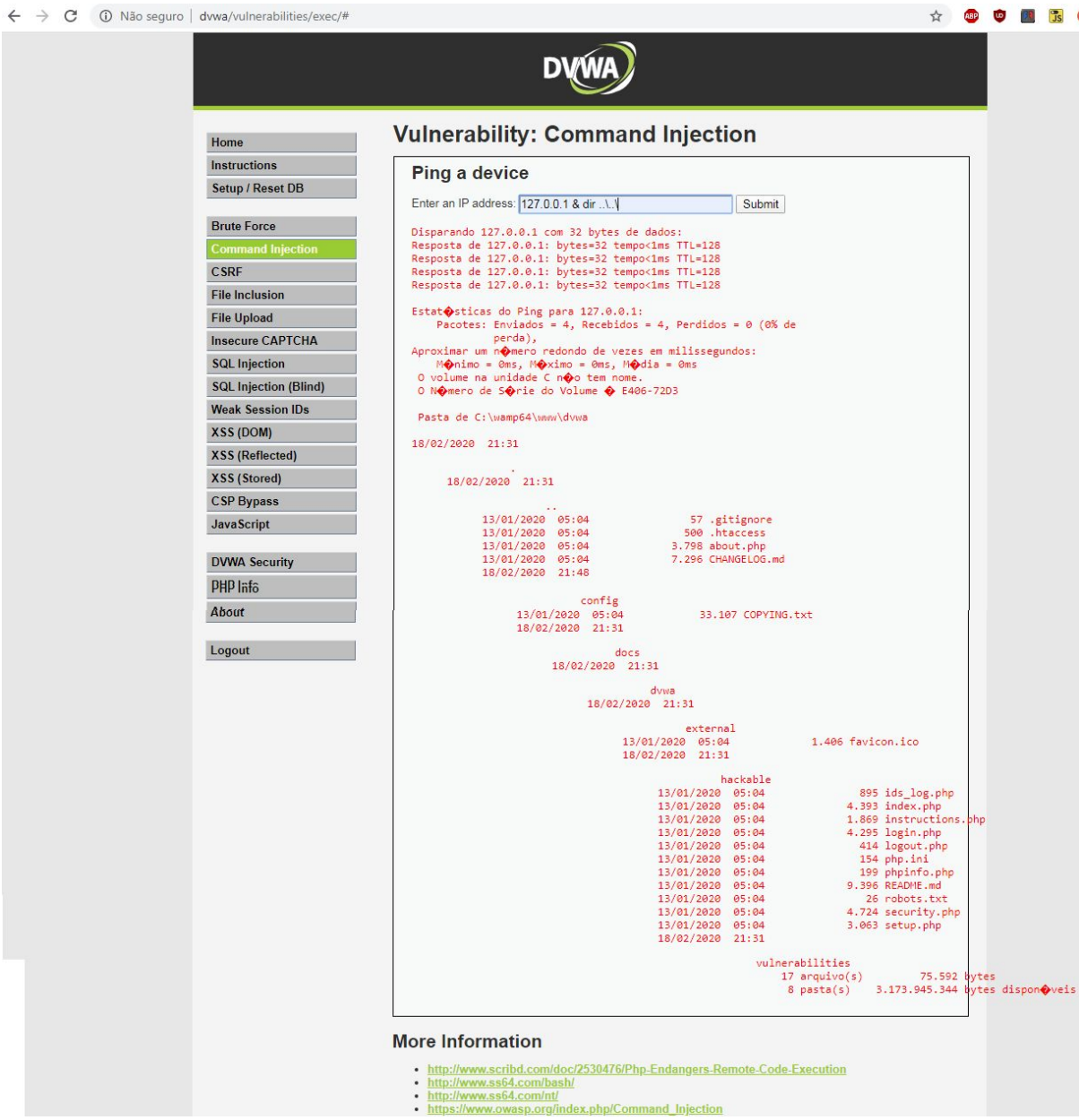
6) São pequenos arquivos em que são armazenados no computador de um usuário. Eles são designados a segurar uma boa quantidade de dados especificos para um cliente particular e site, e pode ser acessado tanto por um web server ou o computador de um cliente. Isso permite o servidor a entregar uma página feita ou alterada para um usuário em específico, e torna possível carregar informação de uma visita ao website para o próximo.
   O principal ataque relacionado a **Cookies** é o *Session Hijacking Attack*.

7) OWASP Top10 é um documento que mostra as 10 vulnerabilidades mais críticas para aplicações web em um consenso geral nos últimos anos.

8) É um tipo de ataque em que o intruso entra no sistema alvo para obter informações sobre vulnerabilidades. O ataque frequentemente usa Port Scanning, para descobrir portas vulneráveis. Isso é importante pois pode ser

usado para detectar vulnerabilidades antes delas serem exploradas por pessoas mal-intencionadas.

9) Command Injection (SO-Injection)
    a) Command Injection é um ataque em que o objetivo é a execução de comandos arbitrários no host operando no sistema por meio de uma aplicação vulnerável. É possível quando uma aplicação passa dados de usuários sem segurança (formulários, cookies, HTTP Headers, etc) para a shell de um sistema. Normalmente, são executados com os privilégios da aplicação vulnerável.
    b)



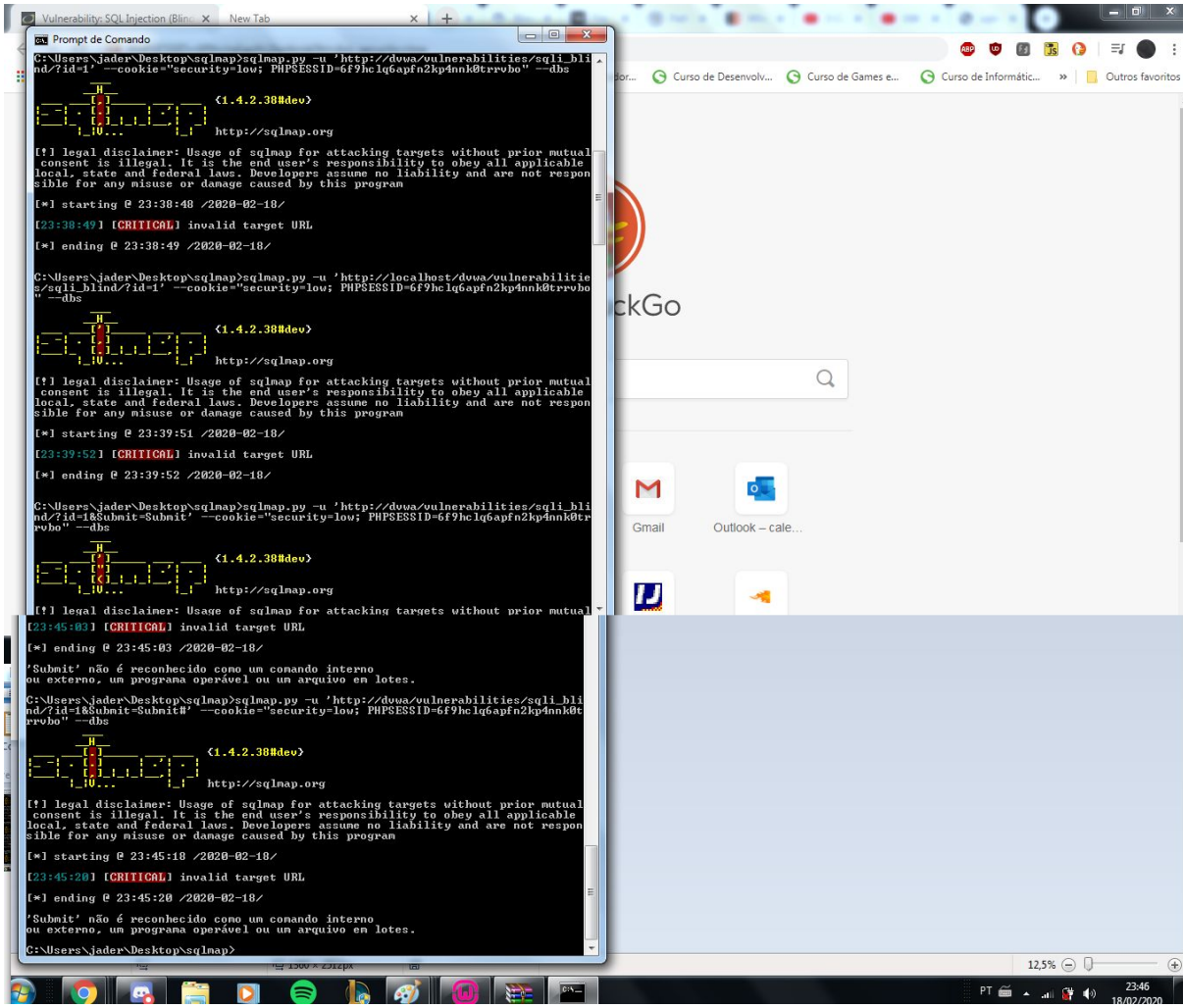
10)SQL INJECTION

a) SQL Injection usualmente ocorre quando você pede entradas como nome de usuário, e o usuário insere código SQL que irão rodar no banco de dados. Isso pode destruir um banco de dados, e ainda é uma das técnicas mais comuns de hacking.

b) Quando uma aplicação é vulnerável a SQL Injection e os resultados do campo são retornados com a resposta da aplicação, a palavra-chave UNION pode ser usada para conseguir dados de outras tabelas do banco de dados.

c) O usuário faz diferentes requisições SQL que perguntam ao banco de dados, questões booleanas, com retorno de TRUE ou FALSE. Se utilizando geralmente de coisas como 1=1 ser TRUE 1=2 ser FALSE. Então ele analisa as diferenças das respostas para conseguir vulnerabilidades.

d) EU FIZ COM TUTORIAL DE VÍDEO, COM AMIGO AJUDANDO E SÓ DÁ ERRO. PORÉM, EU SEI COMO FAZ (EU ACHO), botei as PoF(Proof of Fail) aí



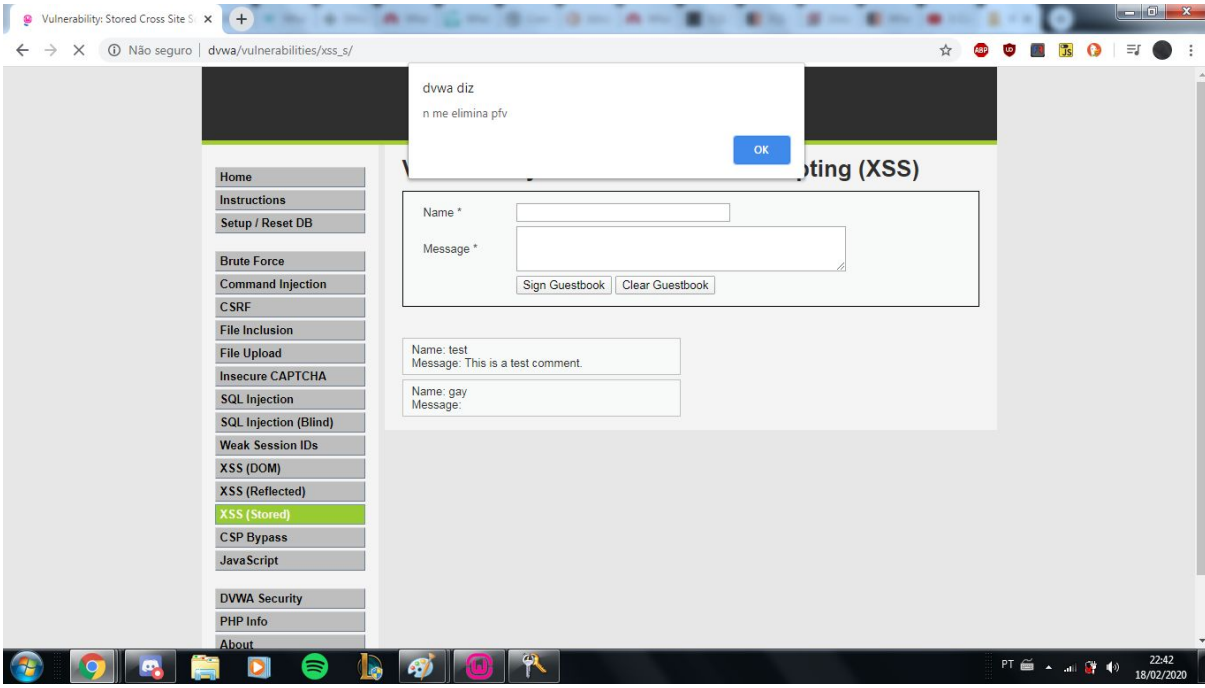
11)XSS

a) XSS é um tipo de vulnerabilidade que é muito espalhada e de fácil detecção e também e uma das mais importantes do OWASP TOP 10. É quando um indivíduo pode injetar “snippets” de JS na aplicação sem validação, então o JavaScript é executada pela vítima que está visitando a área atacada do site.

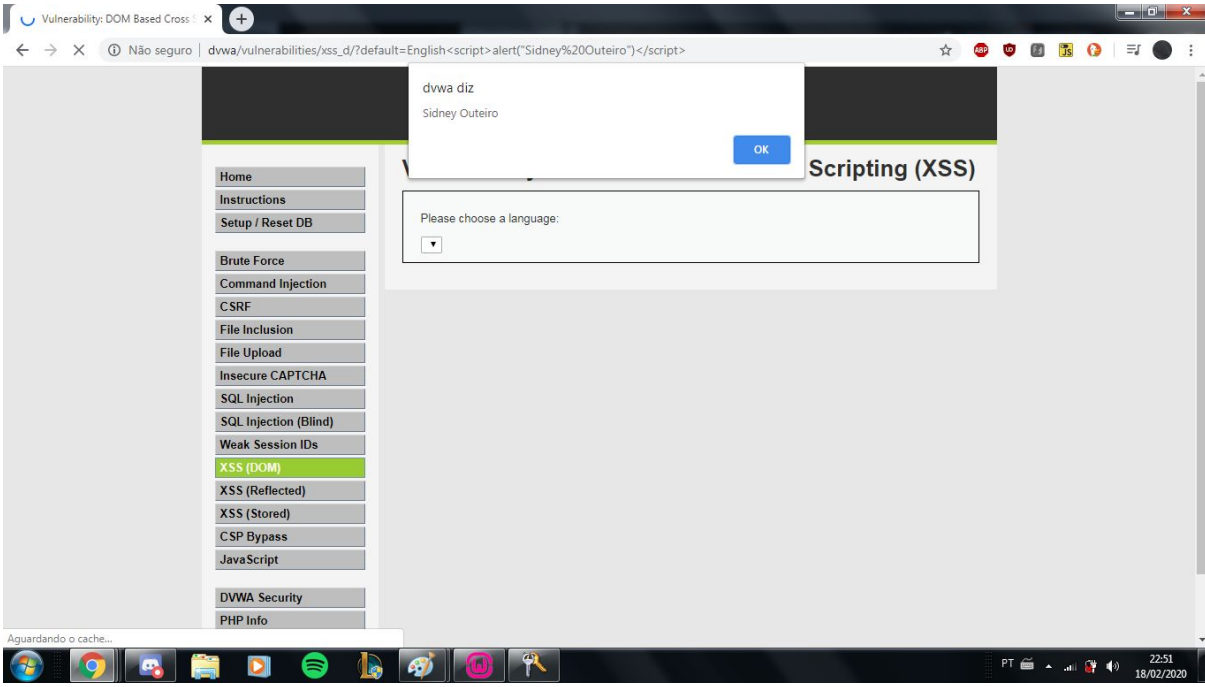
b) É classificado em 3 tipos:

- Reflected XSS - É quando o script malicioso vem da própria requisição http. e necessita, normalmente, de uma engenharia social para a vítima fazer a requisição.
- Stored XSS - Consiste em utilizar uma página acessada para executar o script do ataque ao site.
- DOM-XSS - Quando um javascript do lado de um cliente processa dados de uma fonte sem confiança e sem sanitização.

c) XSS STORED



d) DOM-XSS



12)LFI , RFI e Path Traversal

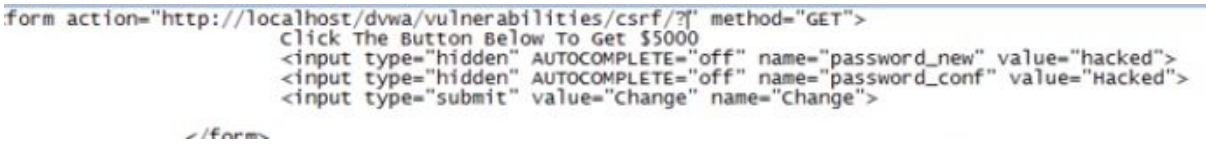
a) LFI é **L**ocal **F**ile **I**nclusion, e consiste, em geral, explorar o parâmetro utilizado no include mal feito. Ela é usada constantemente em conjunto com falha do Path Traversal (como logs).
b) **R**emote **F**ile **I**nclusion permite o usuário a executar arquivos de um servidor remoto, explorando parâmetros no include mal feito.
c) **P**ath **Traversal** é uma vulnerabilidade que permite o usuário a resgatar arquivos de um servidor local, “escalando diretórios” basicamente por meio de parâmetros mal sanitizados.
d) O Path Traversal permite escalar os diretórios, enquanto o LFI permite executar os arquivos e diretórios, por isso são constantemente usados juntos.
e) PoCs de Log Poisoning(retirado da internet pois não consegui realizar no DVWA os exemplos achados): Apache sempre loga usuários autorizados. O header de autorização é parte do protocolo HTTP. Ele cria uma janela requisitando um usuário e uma senha quando você tenta chegar a uma pasta protegida.

Autorização: Basic e Senha:**PD9QSFAgZWNobyBzeXN0ZW0oJF9HRVRbJ3knXSk/Pjo=** . Onde **PD9QSFAgZWNobyBzeXN0ZW0oJF9HRVRbJ3knXSk/Pjo=** em base64 é a string encodada de **<?PHP echo system($_GET[‘y’])?>:** Esse código vai ser descompactado para o Apache diretamente para os logs. E nisso podemos obter o RCE através do acesso de logs.

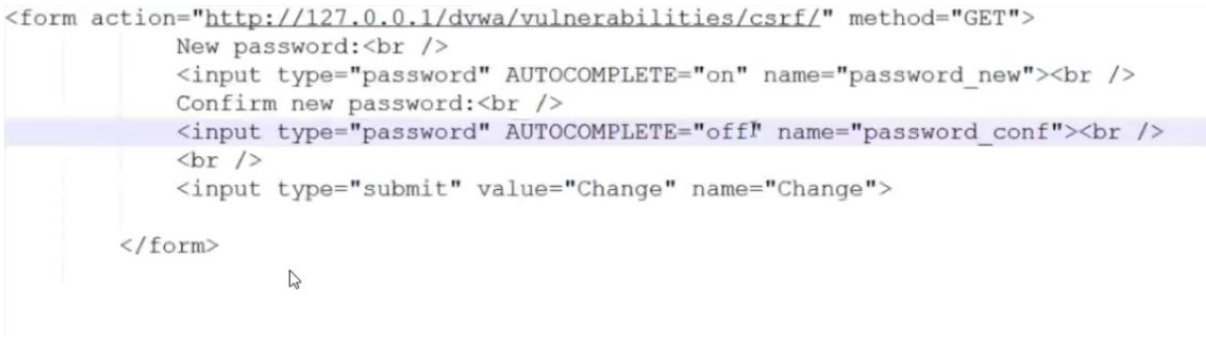
13)CSRF e SSRF

a) Cross-Site Request Forgery (CSRF ou XSRF ou sea-surf) é uma vulnerabilidade em que o usuário mal-intencionado induz outros usuários a realizar ações que eles não tinham intenção. Como incluindo código malicioso em uma página legítima da internet. O ataque ocorre quando a vítima visita o site e executa o código. A página web se torna um veículo para o código malicioso, e explora a confiança que o Site tem no navegador do Usuário.

b) Pegar a seguinte parte do código, e colocar no lugar senha nova



trocando essa parte do código, podemos alterar o resultado



c) Server-Side Request Forgery: O atacante induz a aplicação a fazer uma requisição (Server side) para um domínio, e ela possibilita o atacante a forjar que o servidor está conseguindo acessar a si mesmo, dessa maneira o servidor, muitas vezes, permite acessar serviços com um maior privilégio escalonando

d) Não fiz porque só acho exemplos de como fazer em coisas ilegais, como Facebook.

e) Evitar o uso de “lembrar-me”, usar extensões como RequestPolicy, CsFire e NoScript, as aplicações devem verificar o campo de referer e usar cookies de página para evitar Session Hijacking.