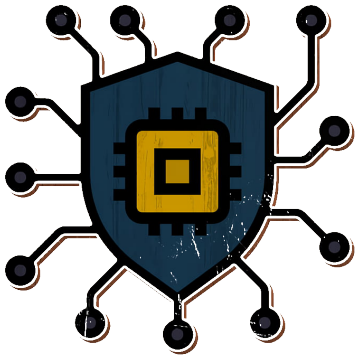


W13D1

Exploit File Upload su DVWA con BurpSuite

[Facoltativo] Ripetizione con Shell Sofisticata

Pratica Extra

★ INDICE

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.

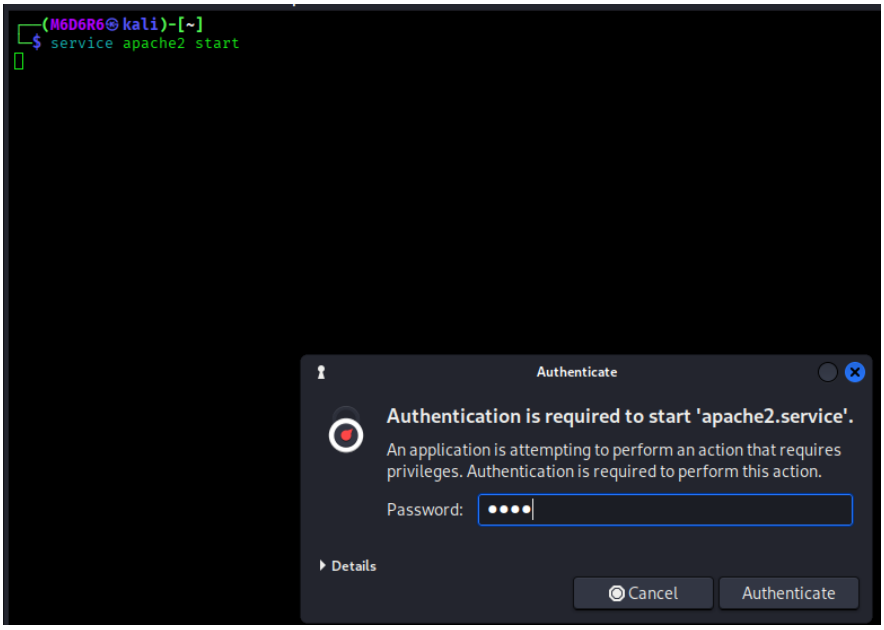
Spiegazione Apache2 serve DVWA

Spiegazione MariaDB gestisce il database DVWA

Comandi:

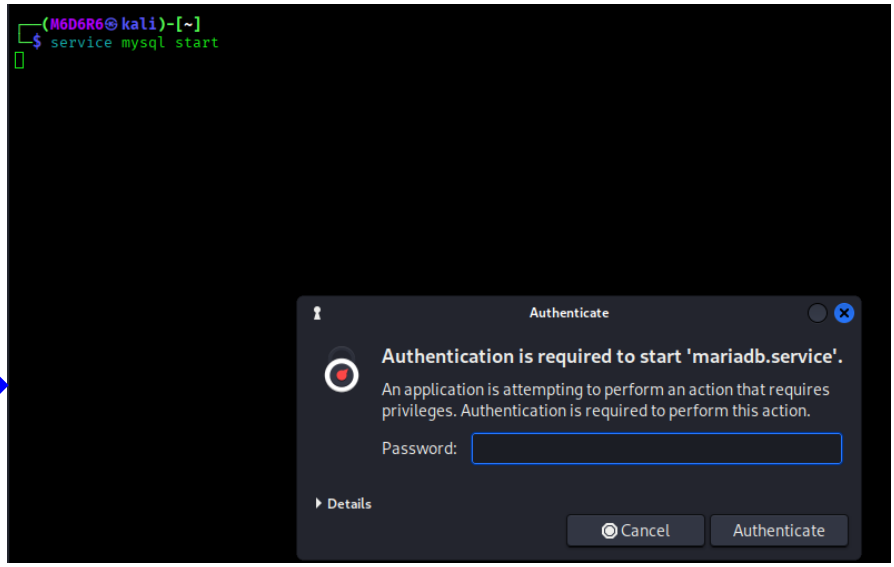
- Per Apache2:

service apache2 start



- Per MariaDB:

service mysql start



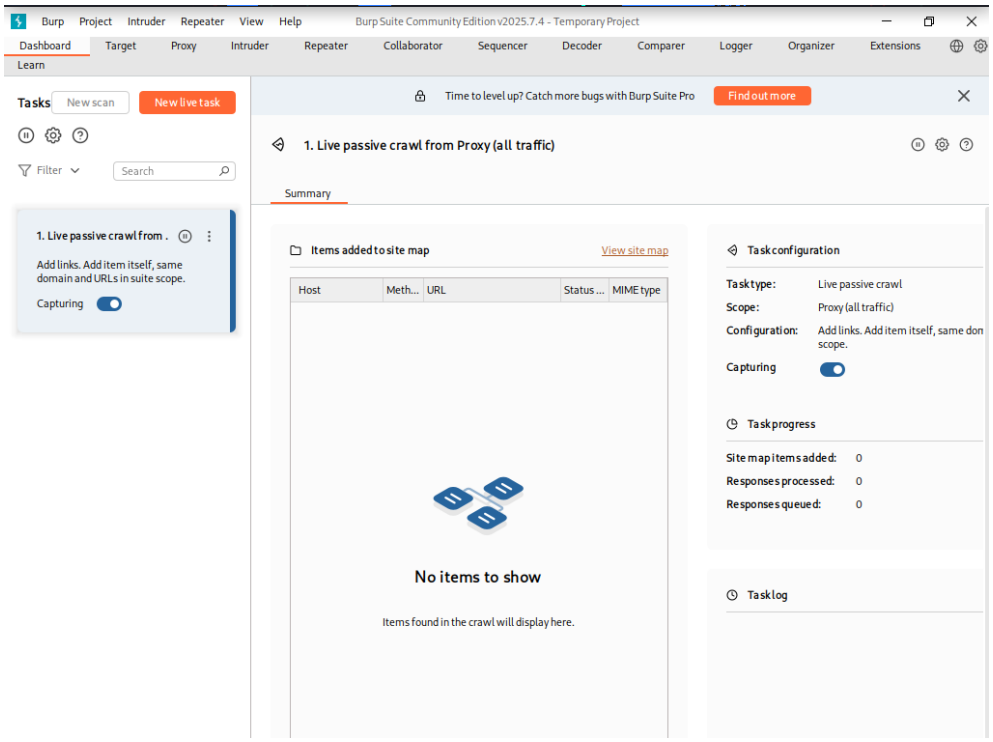
❍ Se si presume che siano gia attivi si puo controllorare il loro srato:

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

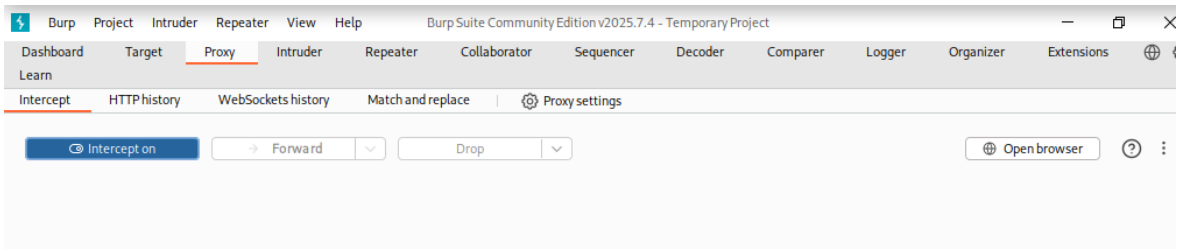
4 Avvio e Configurazione di BurpSuite

- **Descrizione** Avvio BurpSuite e configuro il browser integrato per intercettare il traffico.

- **Comando da termianel** burpsuite

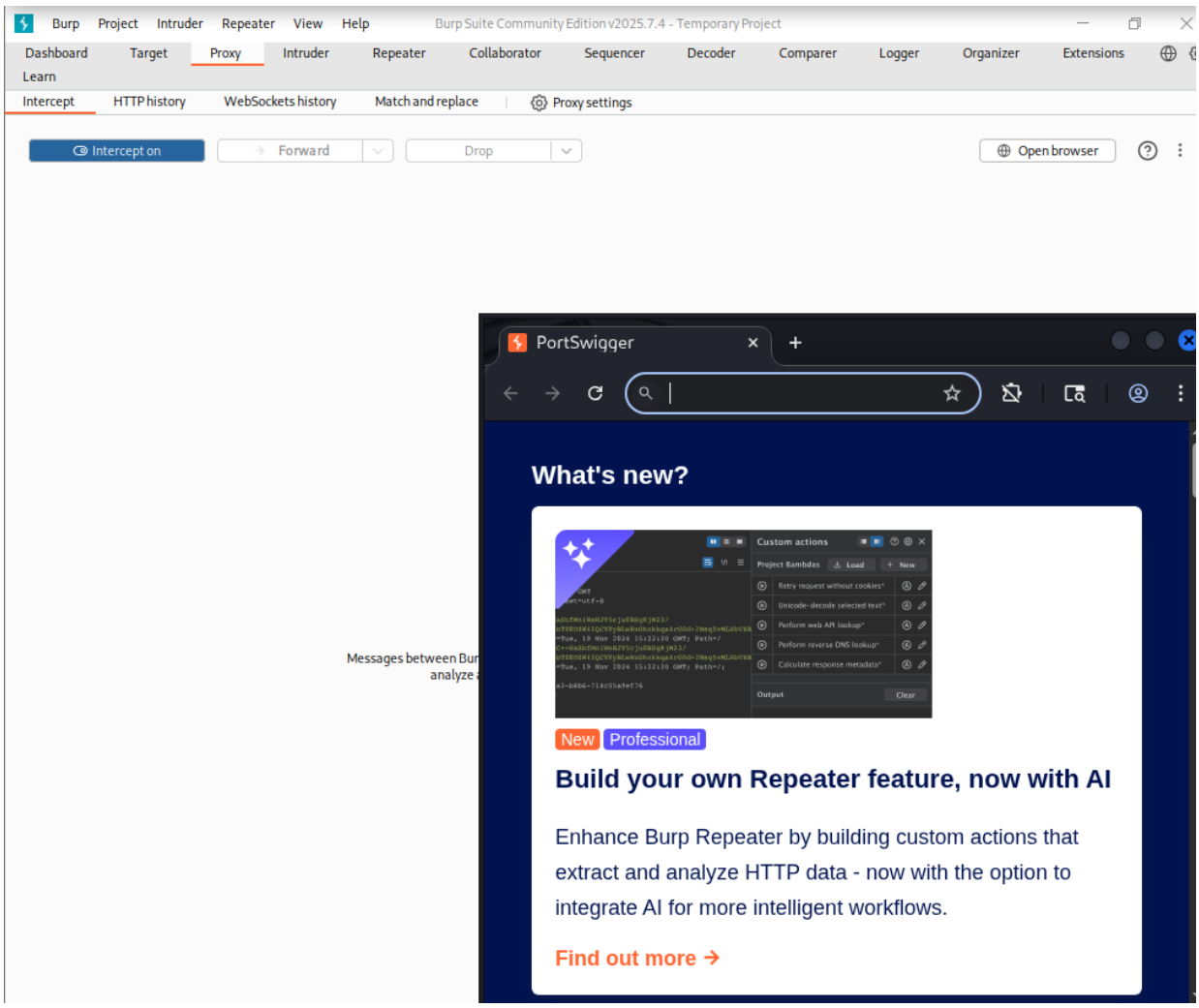


- **Azione** Apro BurpSuite, vado su Proxy > Intercept > Intercept is on.



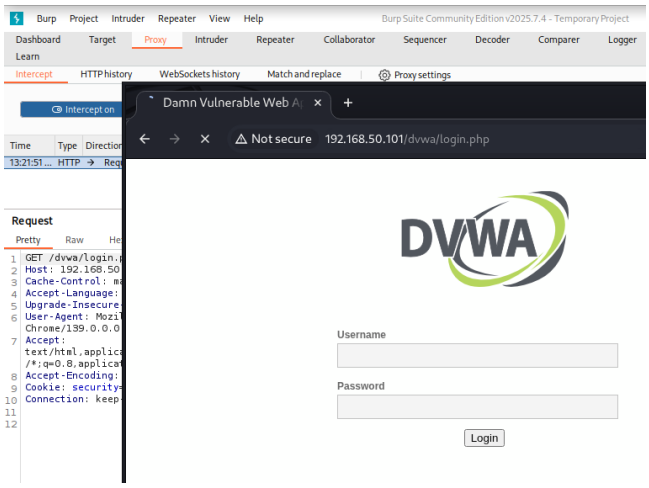
- **Spiegazione** Il browser integrato non richiede configurazione proxy manuale.

- **Azione** Clicco Proxy > Open Browser

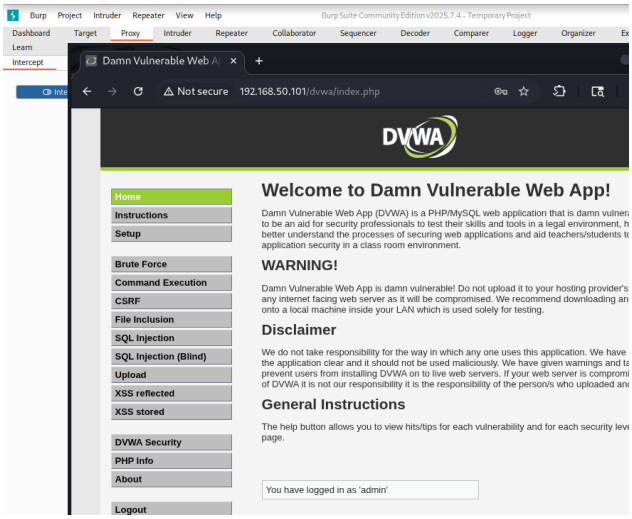


5 Accesso a DVWA e Impostazione Livello Low

- **Descrizione** Accedo a DVWA e imposto il livello di sicurezza su Low.
- **Azione** Nel browser di BurpSuite, vado a

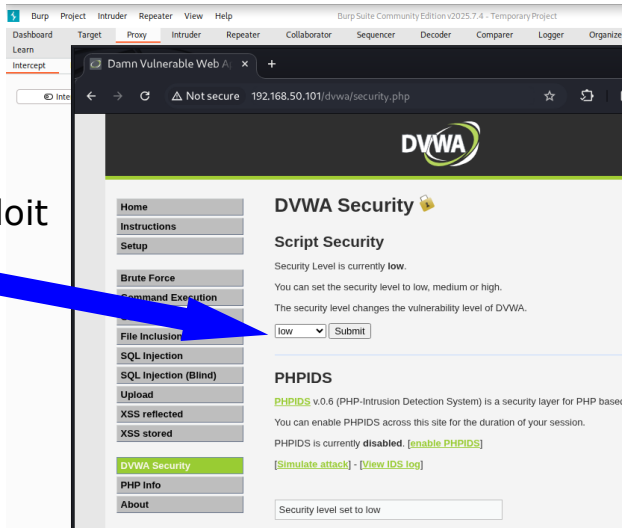


- **Spiegazione** Accede alla pagina di login
  - http://192.168.50.101/dvwa/



Spiegazione Accede alla dashboard

- Login con username admin e password password

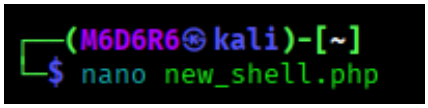


Spiegazione Imposto livello per l'exploit in Low

- Vado a DVWA Security > Seleziona Low > Clicca Submit

7 Creazione e Caricamento della Shell PHP al Livello Low

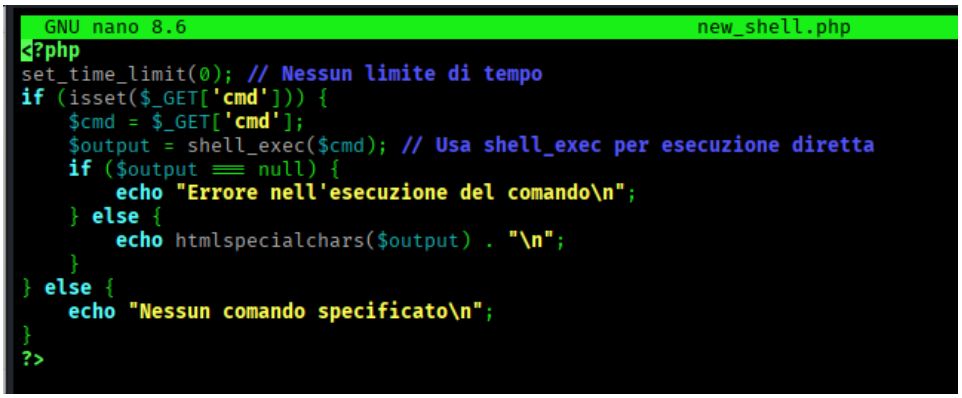
Descrizione Creo e carico shell PHP a livello Low su Metasploitable2.



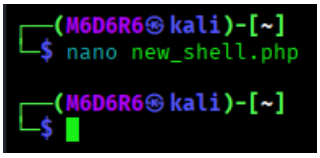
Comando nano new_shell.php

Spiegazione Creo una shell che esegue comandi via GET

Dettaglio editor:

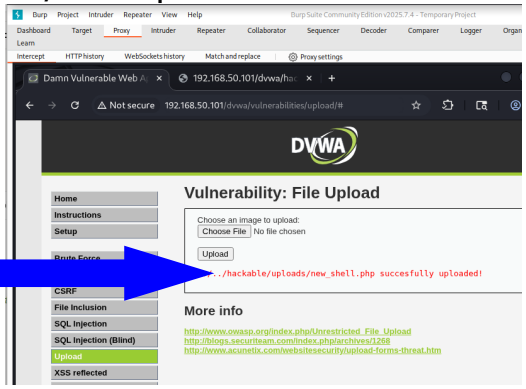


Salvo con Ctrl+O, Enter, Ctrl+X



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

Carico Shell con successo

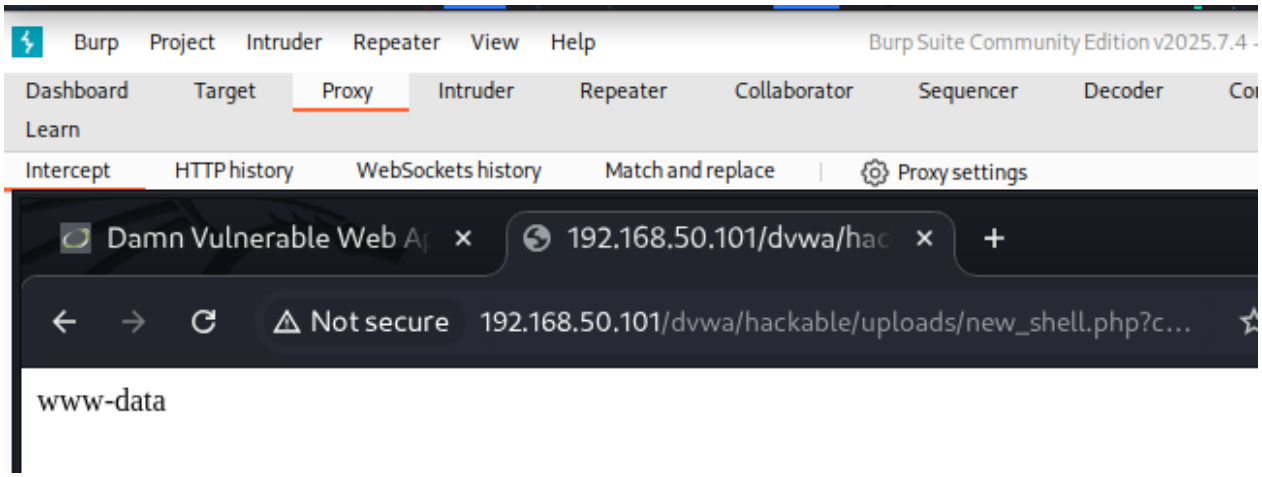


8 Test e Esecuzione della Shell al Livello Low

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



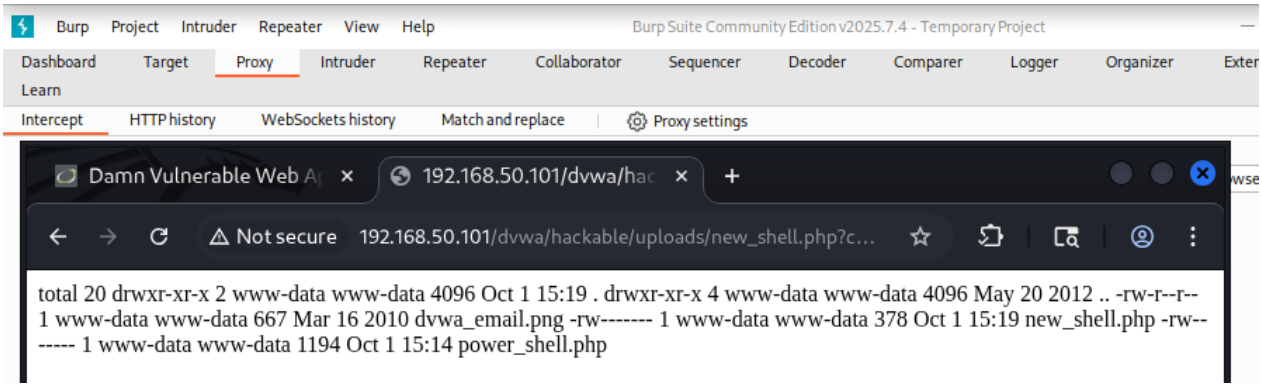
- **Spiegazione** L'output atteso è "www-data" (utente del server web)

- **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)

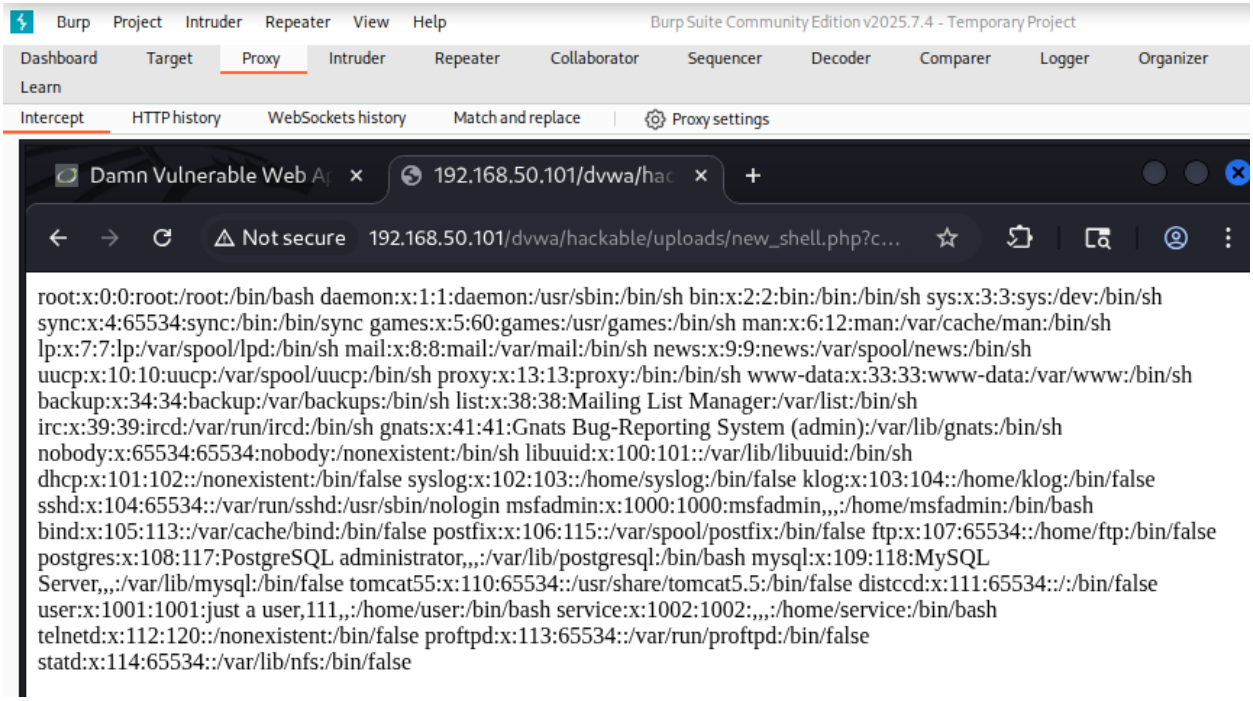


- **Risultato** Test della shell per confermare il controllo remoto andato a buon fine

- **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)



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

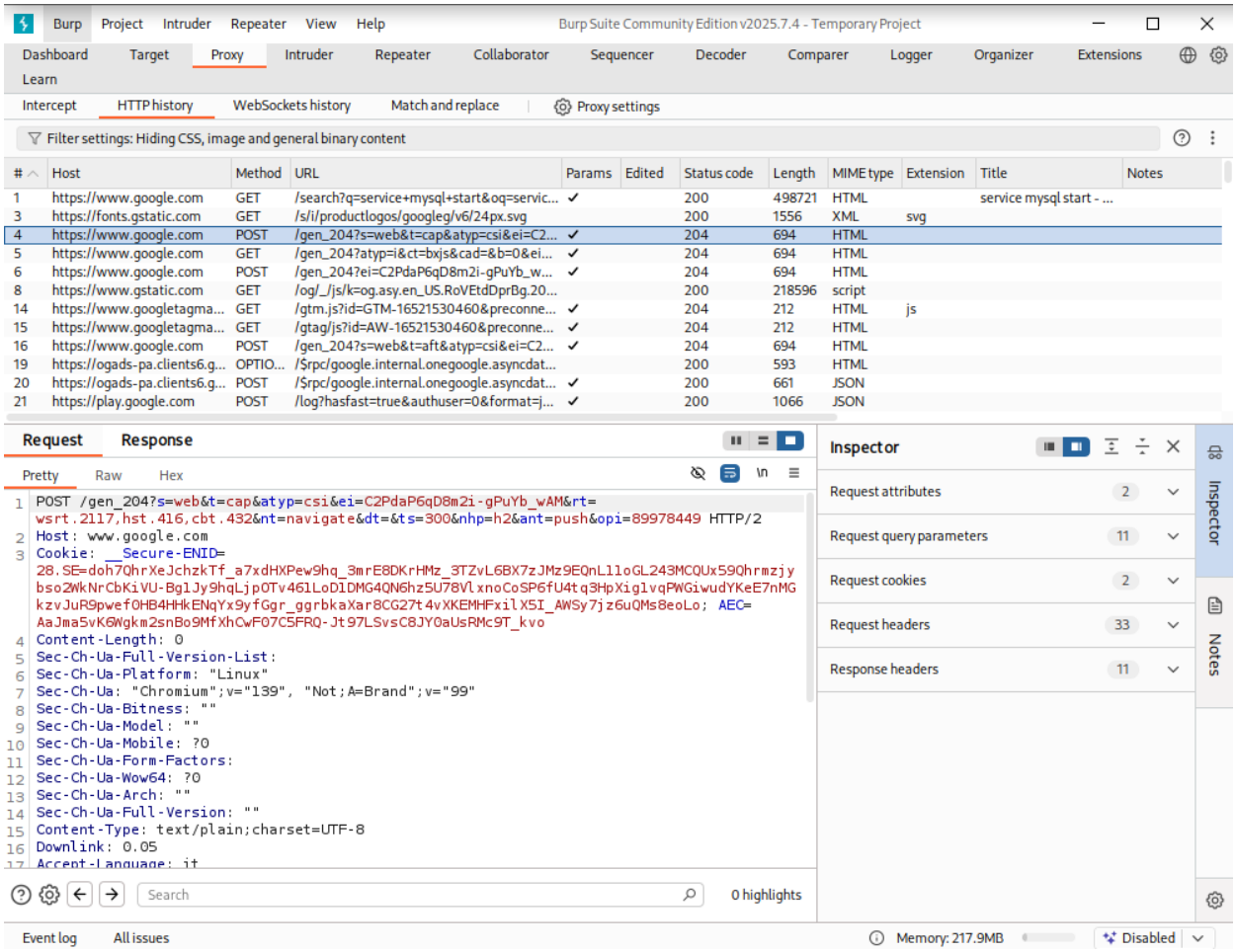
9 Analisi delle Intercettazioni con BurpSuite al Livello Low

- **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 historey con POST e GET evidenziati.

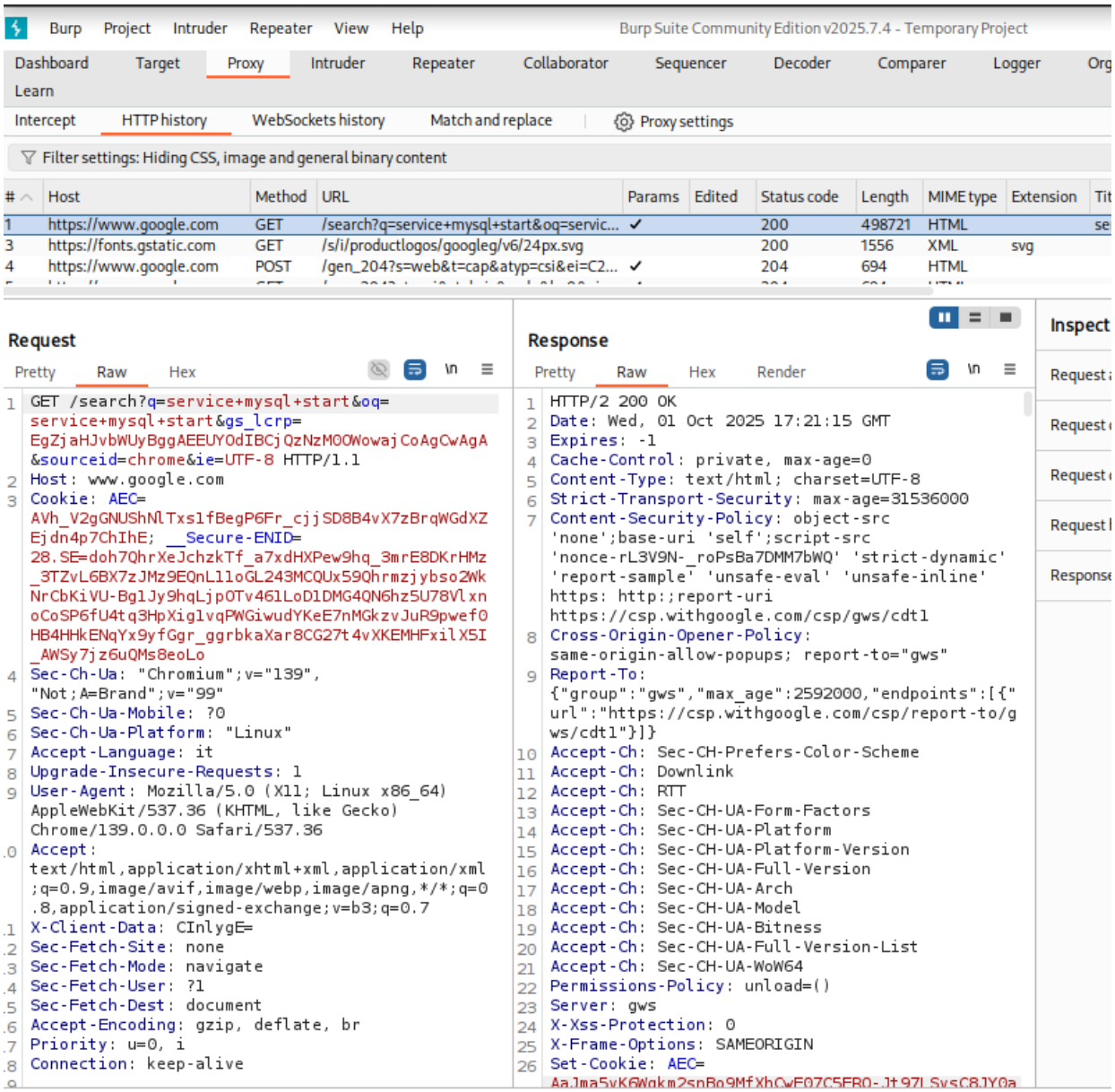


- **Get** Esamino le richieste GET inviate alla shell per eseguire comandi remoti su Metasploitable.

http://192.168.50.101/dvwa/hacka-ble/uploads/new_shell.php?cmd=whoami

- Clicco sulla scheda "HTTP history" e filtro per richieste GET (es. http://192.168.50.101/dvwa/hacka-ble/uploads/new_shell.php?cmd=whoami, ?cmd=ls -la, ?cmd=cat /etc/passwd).

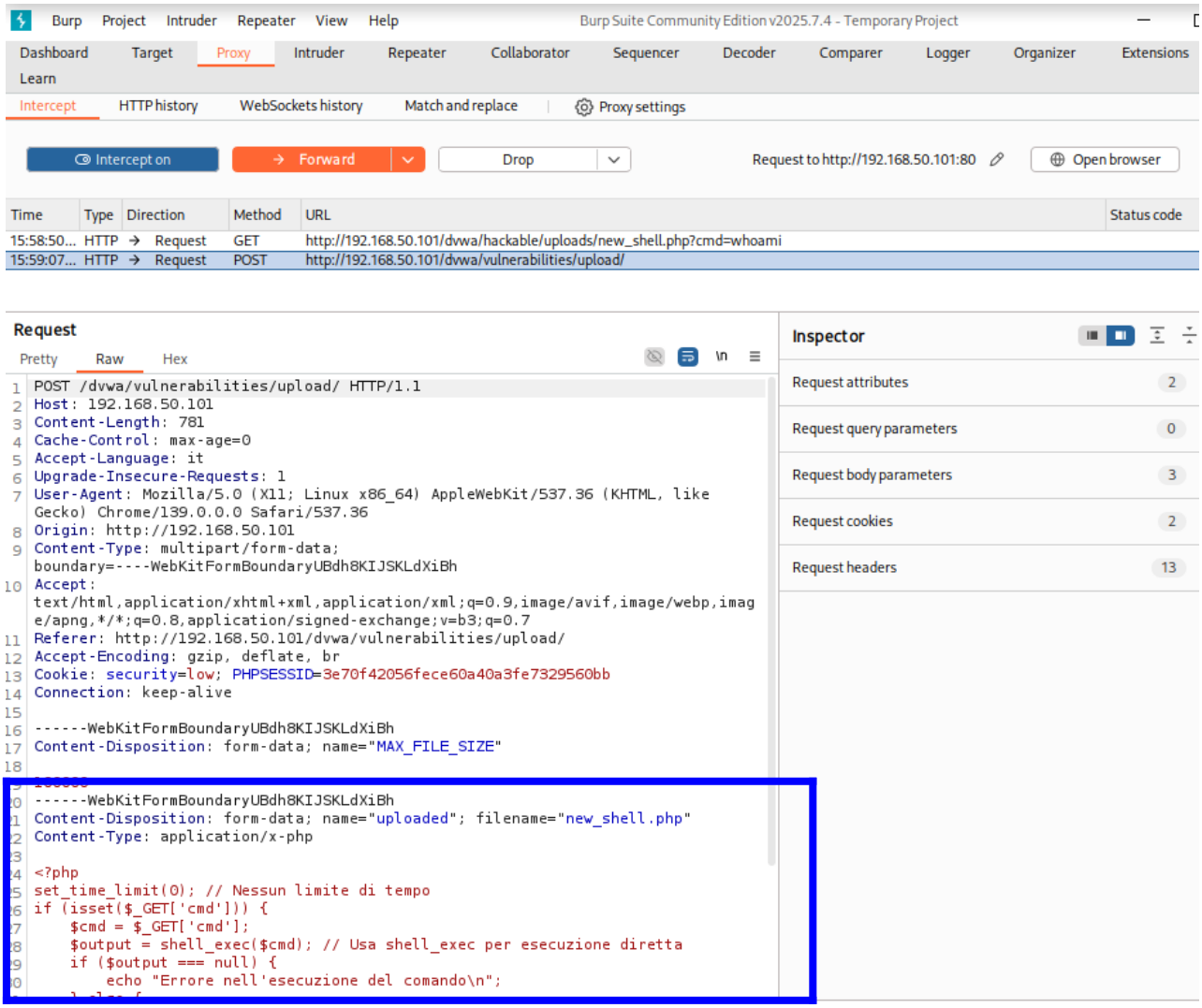
- **Spiegazione** Identifico le richieste GET che passano il parametro cmd alla shell per eseguire comandi.



- **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.



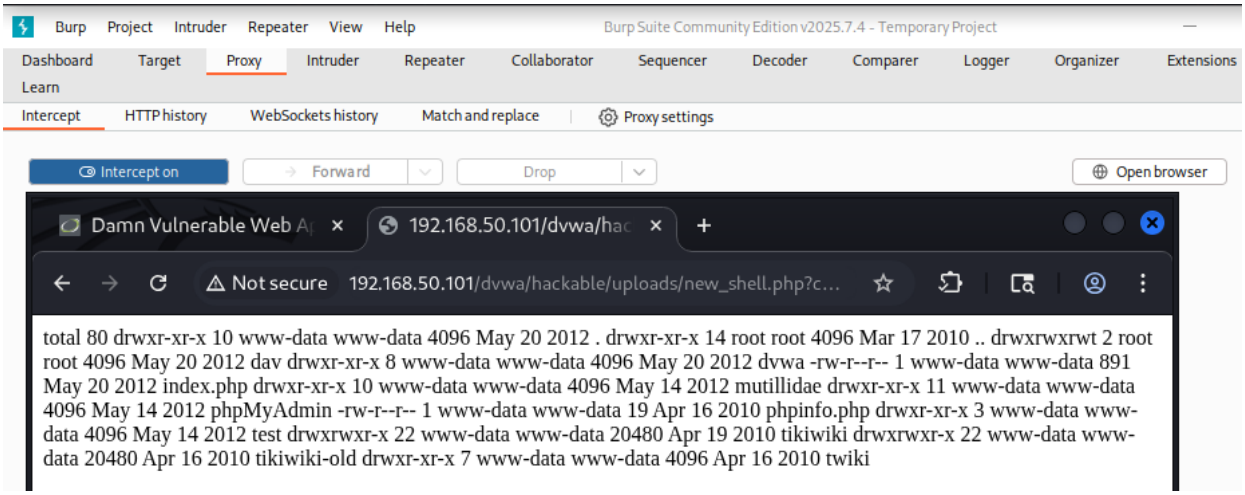
10 Esplorazione della Macchina Target al Livello Low

- **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

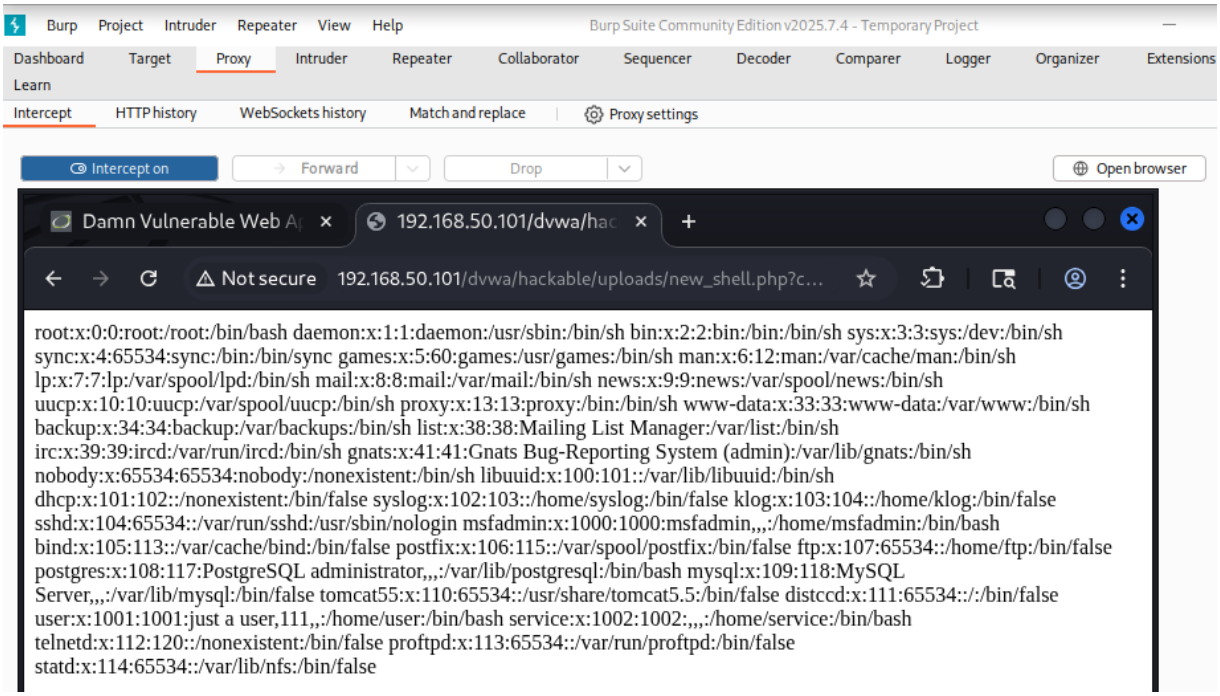
- **Spiegazione** Elenca i contenuti della directory web /var/www con dettagli sui permessi.



- **Azioni 2** Eseguo

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

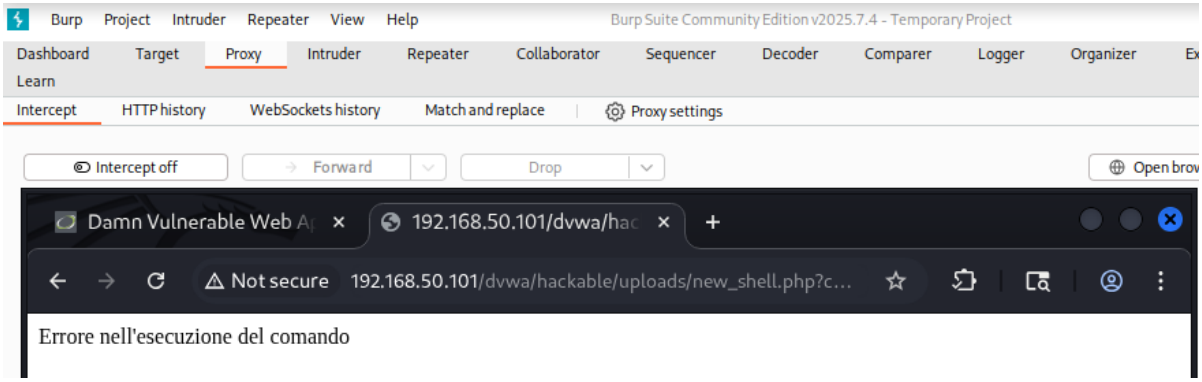
- **Spiegazione** Recupera l'elenco degli utenti registrati su Metasploitable, inclusi root, msfadmin, e www-data.



- **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.

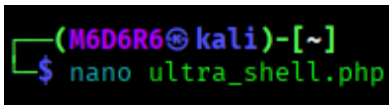


- **Risultato** "L'esplorazione mostra 'ls -la /var/www' con 'html' e permessi, 'cat /etc/passwd' elenca utenti (es. msfadmin), 'cat /etc/shadow' restituisce 'Errore nell'esecuzione del comando' a causa di permessi insufficienti, confermando accesso limitato ma significativo su Metasploitable."

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.

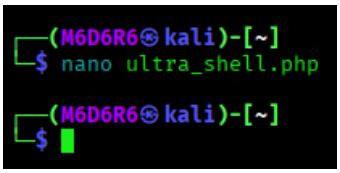
Comando nano ultra_shell.php



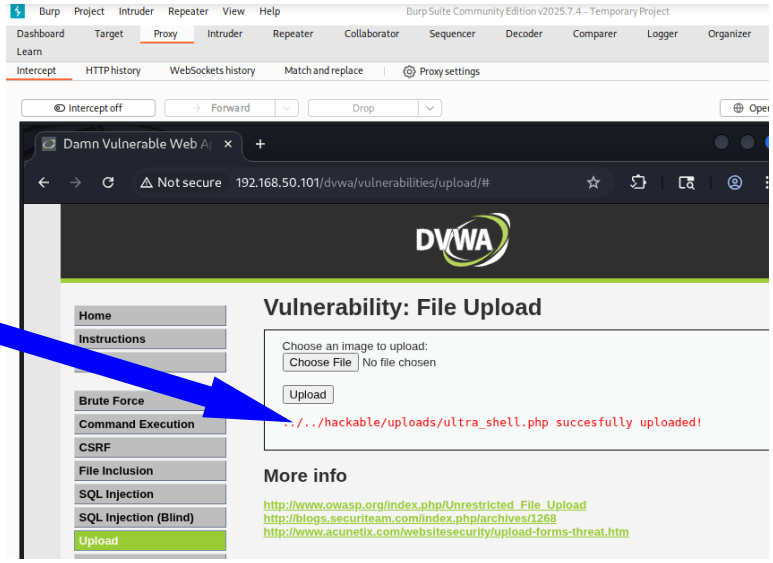
Dettaglio editor:

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



Carico Shell con successo

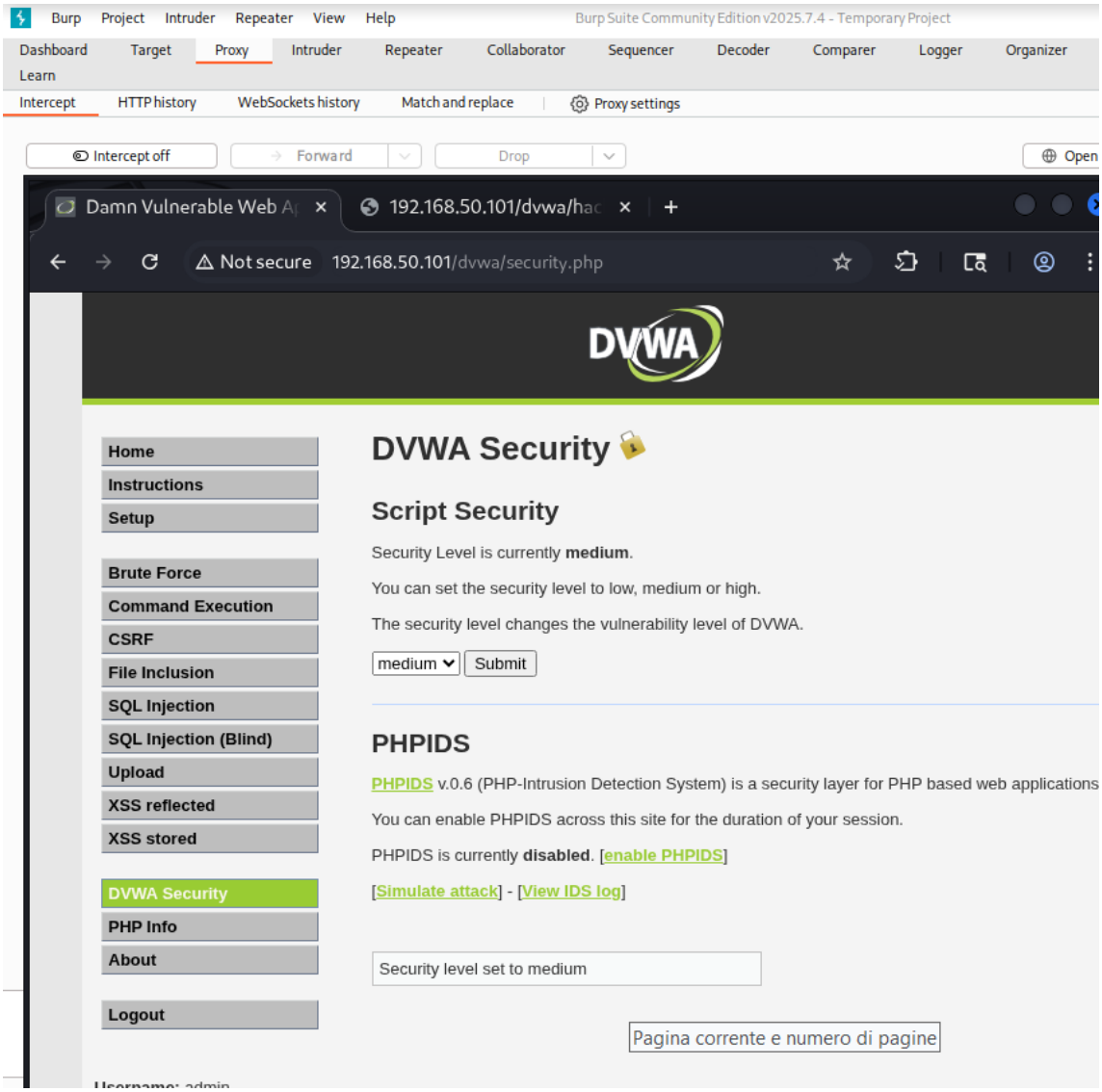


11.1 Test del File Upload al Livello Medium

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"

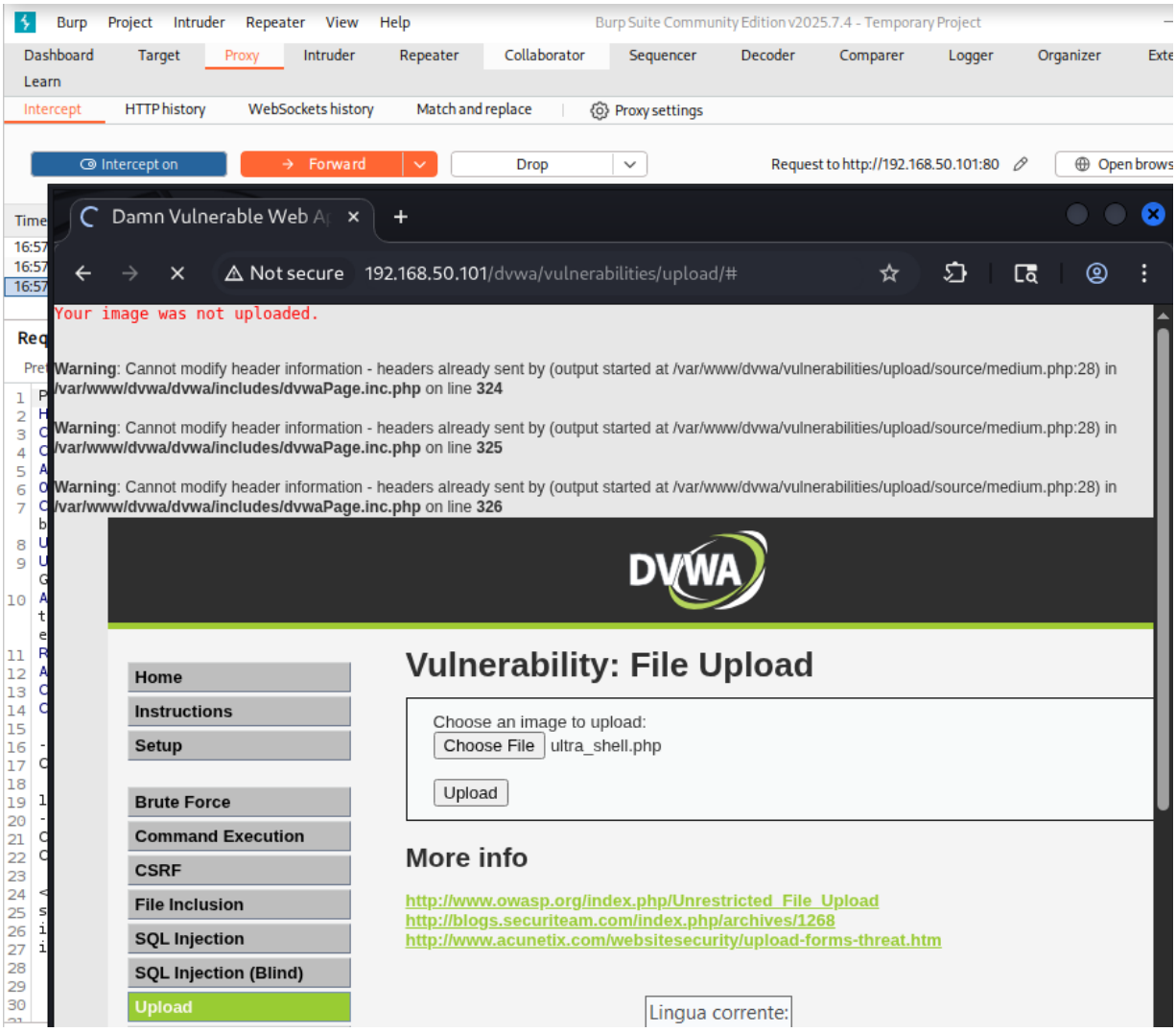
- **Dettagli**: Accedo alla pagina DVWA Security nel browser integrato di BurpSuite, cambio il livello a "Medium" e confermo.
- **Spiegazione**: Imposto un livello di sicurezza intermedio che attivo filtri base, come la verifica delle estensioni dei file (es. .php), per ostacolare l'upload di shell.



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.

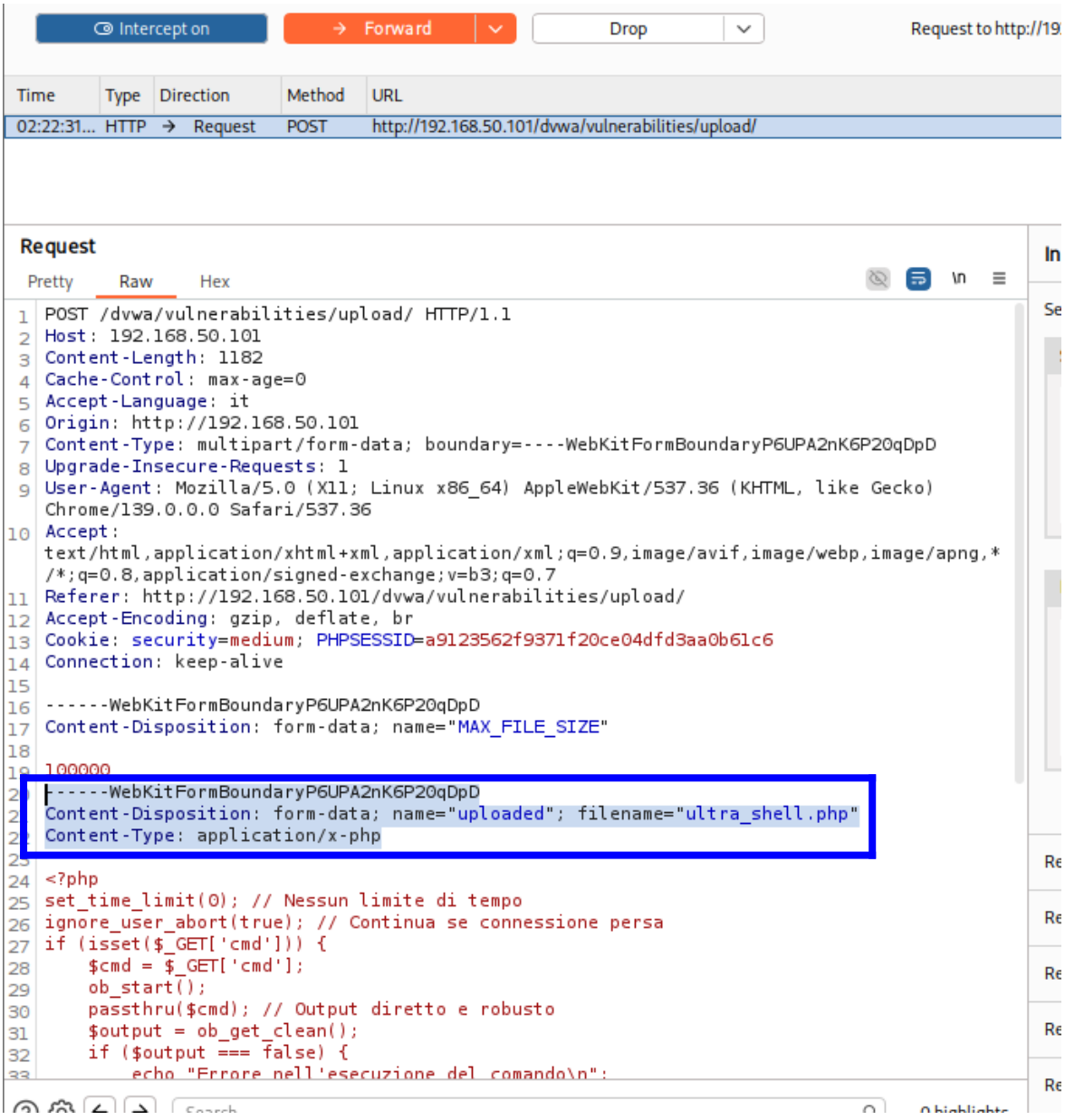


Azione 3 Intercetta la Richiesta

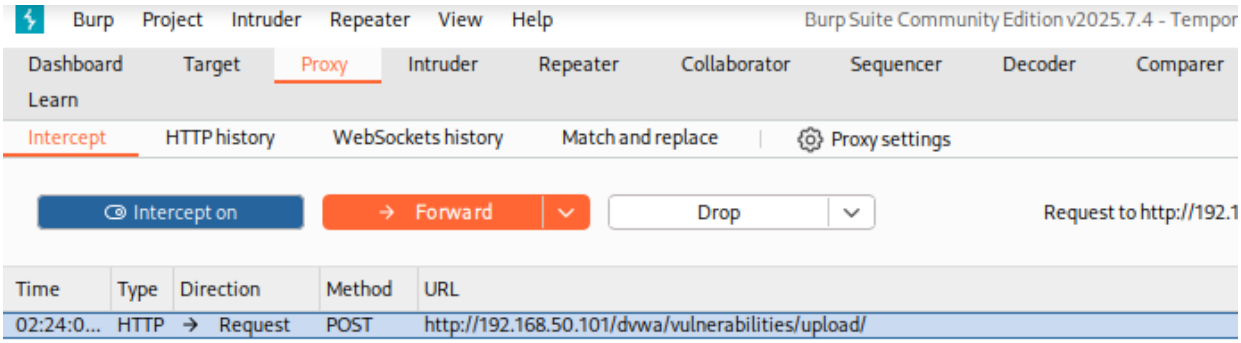
- Nel browser integrato di BurpSuite, vado a "Vulnerabilities > File Upload", seleziono ultra_shell.php, clicca "Upload".
- In BurpSuite (Proxy > Intercept), attendo che la richiesta POST venga intercettata.

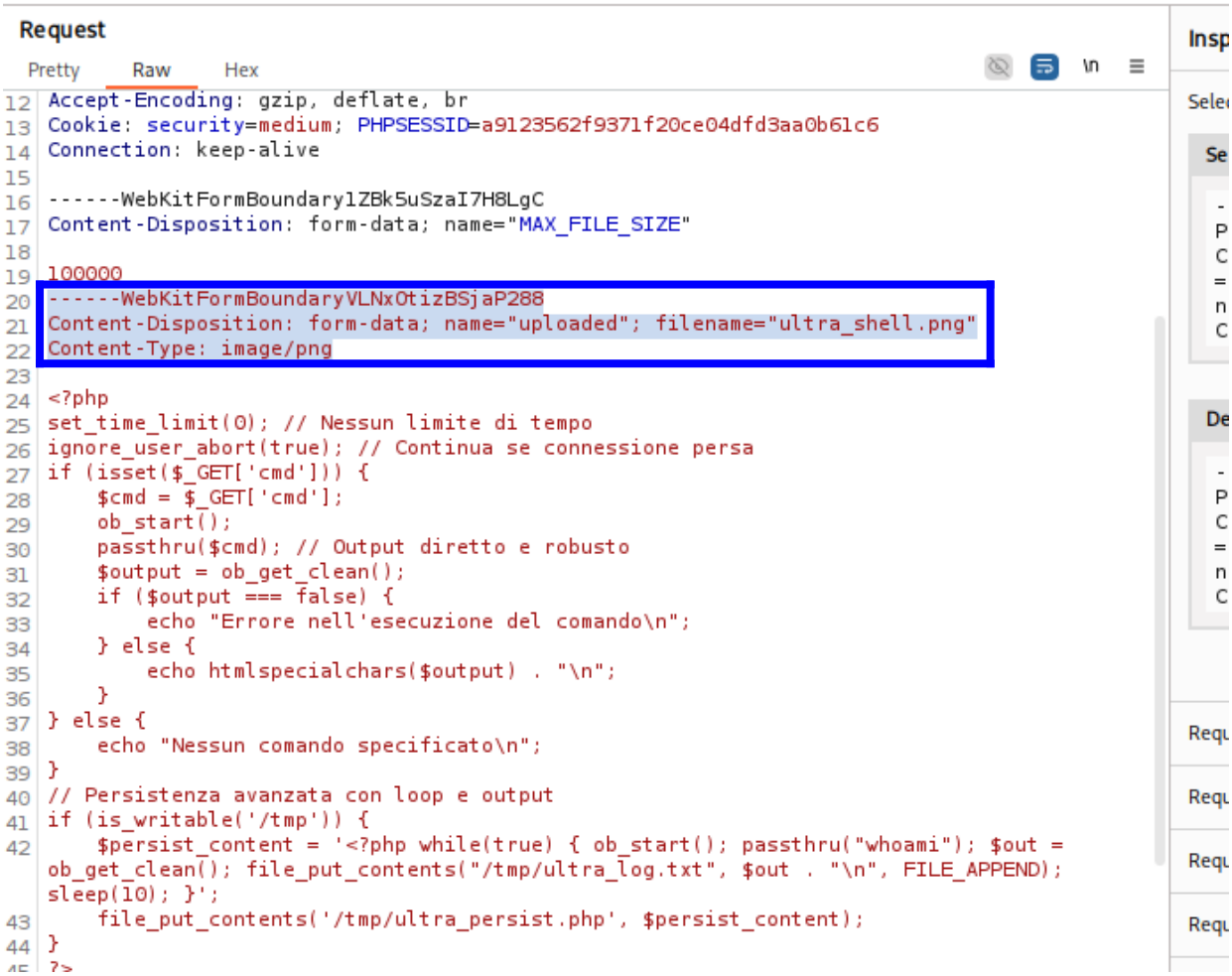
Modifico la Richiesta

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



❍ Sostituisci con





❍ Invia la Richiesta

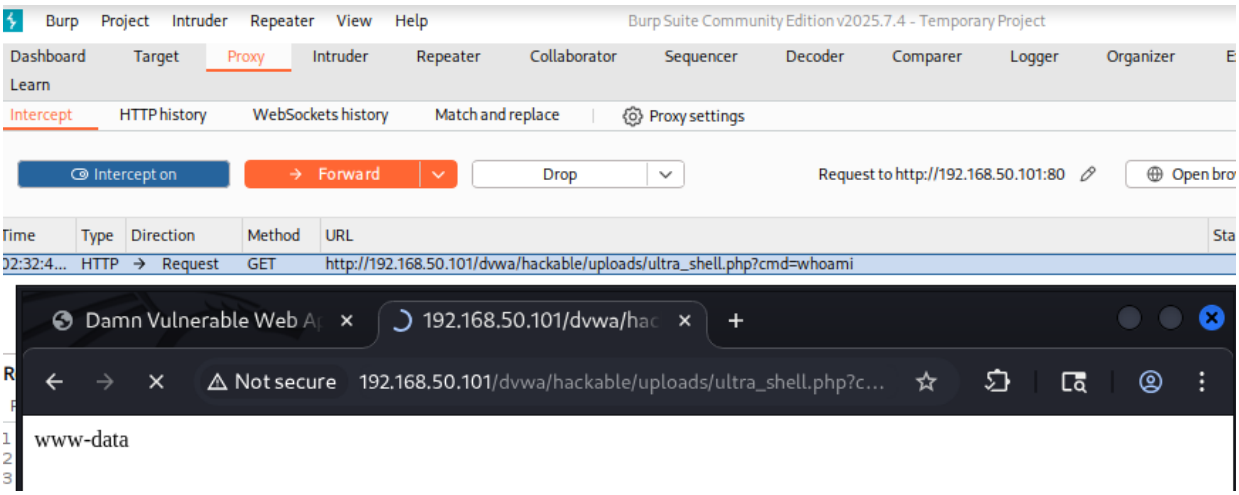
- Clicco "Forward" in BurpSuite per inviare la richiesta modificata al server.

❍ **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".



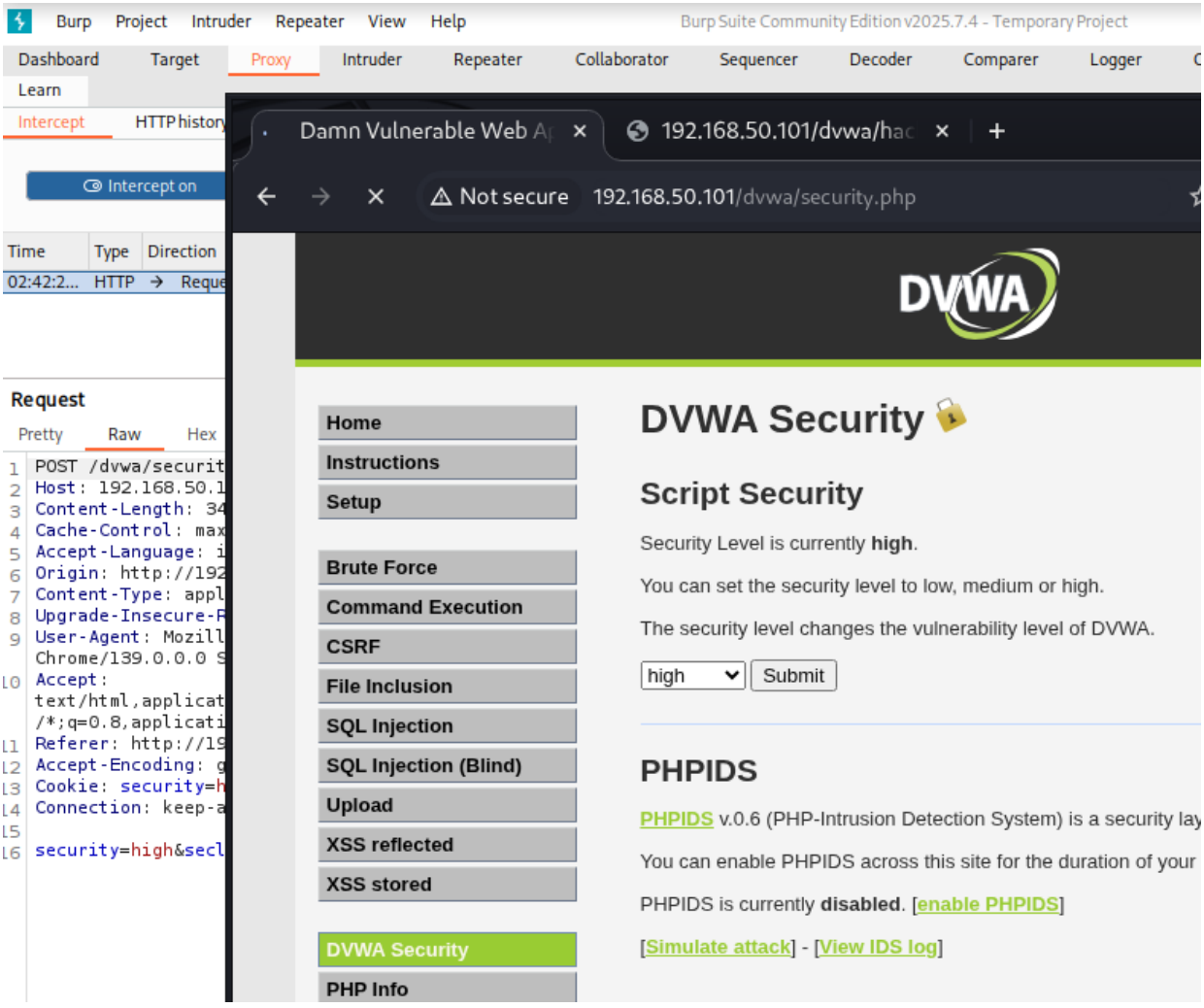
❍ **Risultato** A livello Medium, l'upload iniziale fallisce per il filtro MIME con Your image was not uploaded, correggendo il Bad Request con intestazioni valide e modificando il nome file a .png e Content-Type a image/png in BurpSuite, **il bypass riesce**, e la shell ultra_shell.php (o ultra_shell.png)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

11.2 Test del File Upload al Livello High

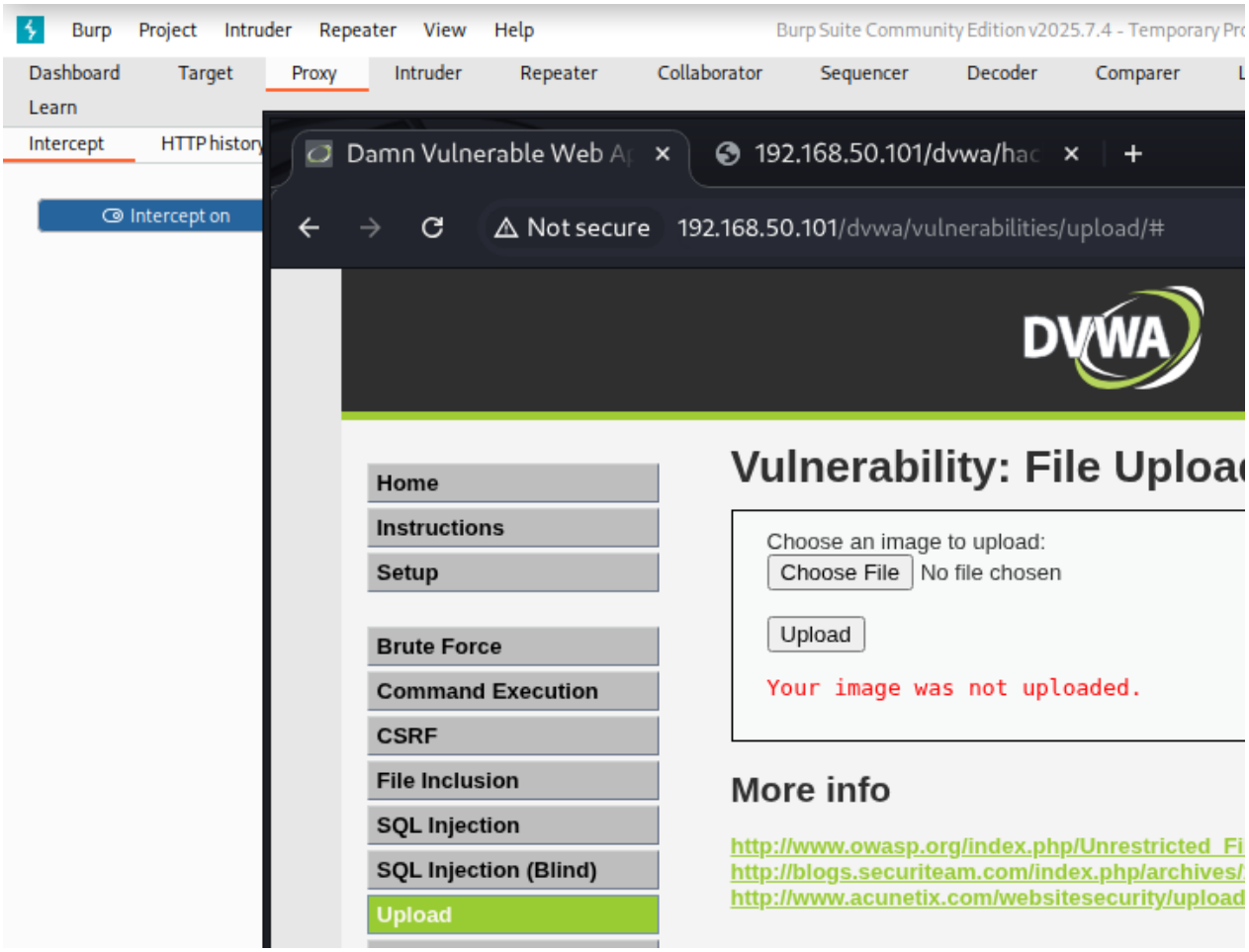
Descrizione Imposto il livello di sicurezza High su DVWA su Metasploitable2 e tento di caricare la shell ultra_shell.php, analizzando i filtri avanzati (come getimagesize() e token CSRF) e bypassandoli con BurpSuite per risolvere l'errore "Your image was not uploaded"

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.

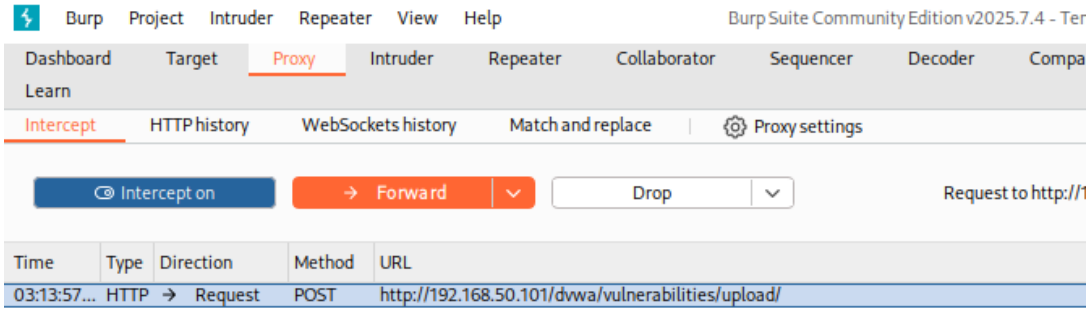


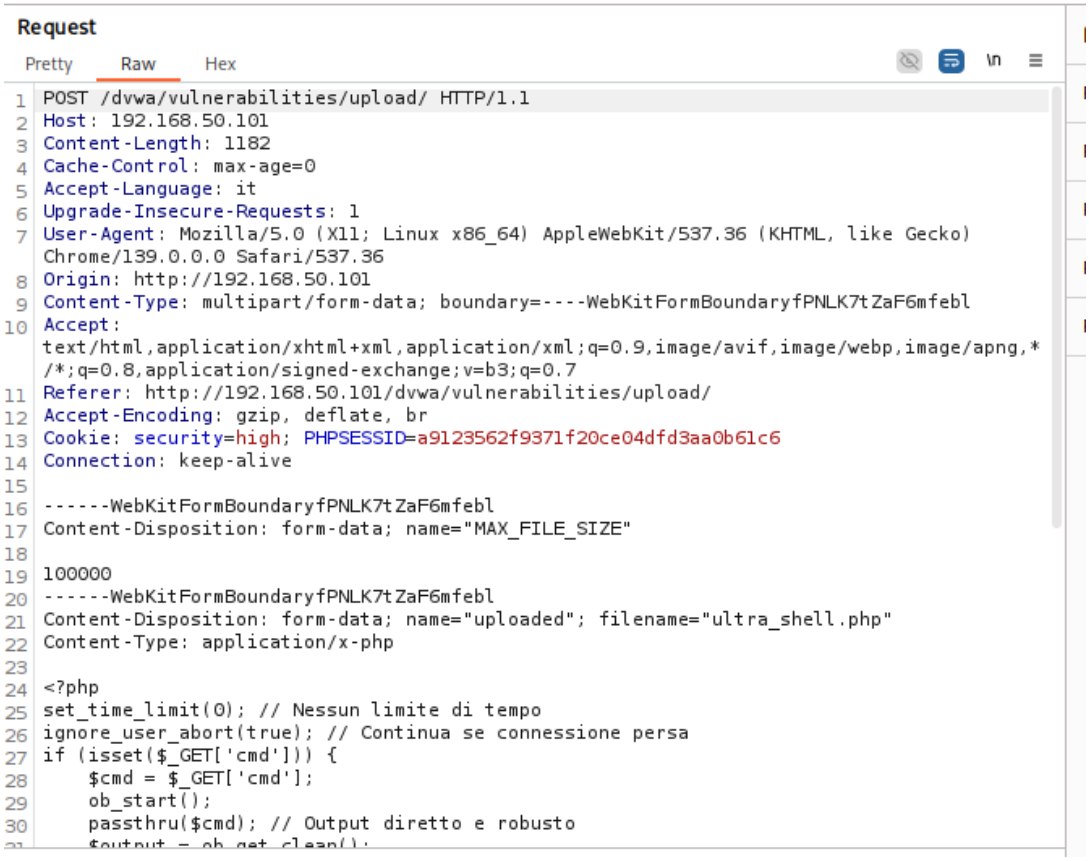
- **Descrizione** Intercetto la richiesta POST in BurpSuite, modifico a GIF89a + PHP, clicco Forward

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

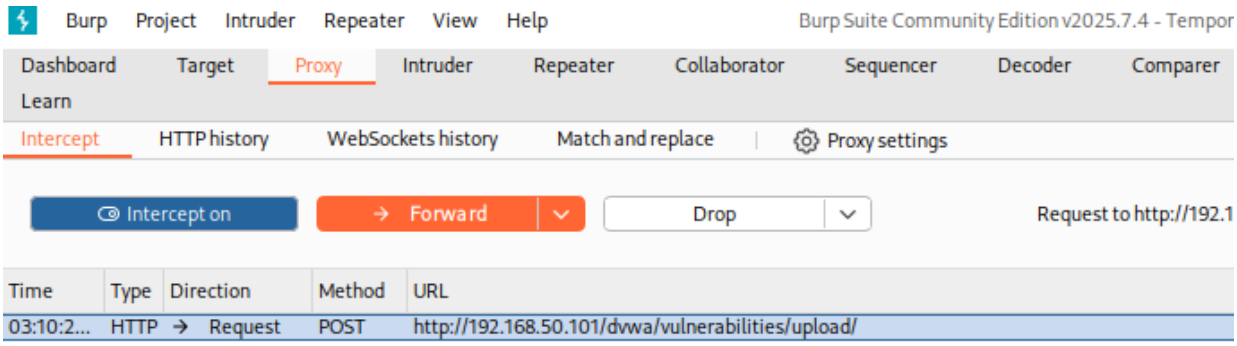
- Aggiungo GIF89a all'inizio del corpo del file (prima del codice PHP) nella sezione multipart/form-data.
- Mantengo il token CSRF invariato
- Clicco "Forward" per inviare la richiesta modificata al server

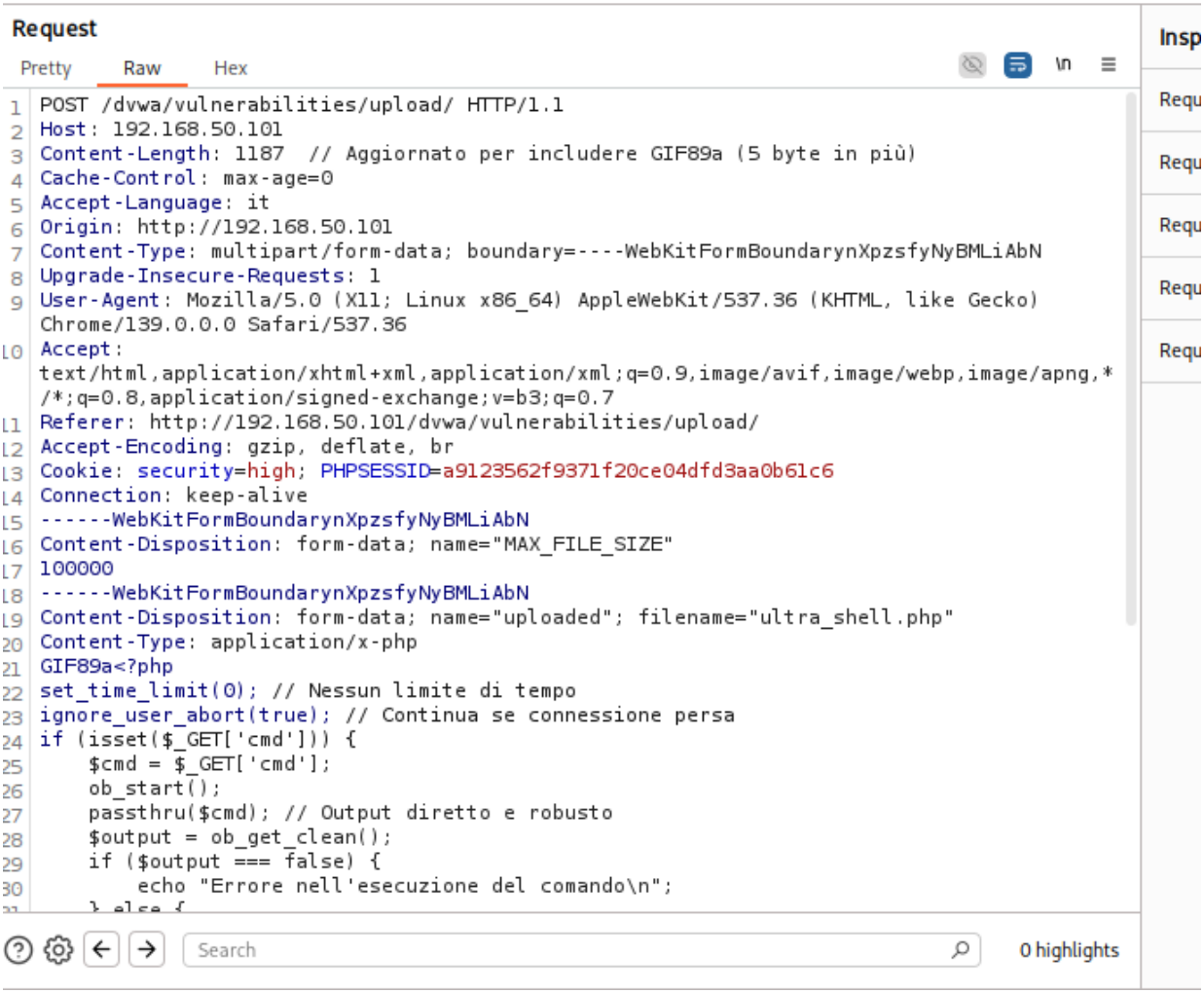
Intercettazione POST Originale





Intercettazione POST Modificata





Invio la Richiesta

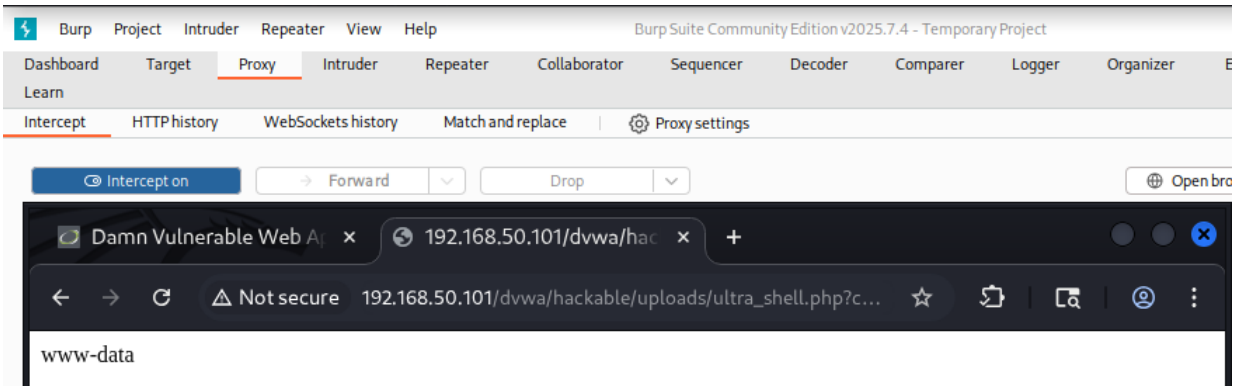
- Clicco "Forward" in BurpSuite per inviare la richiesta modificata al server.

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".

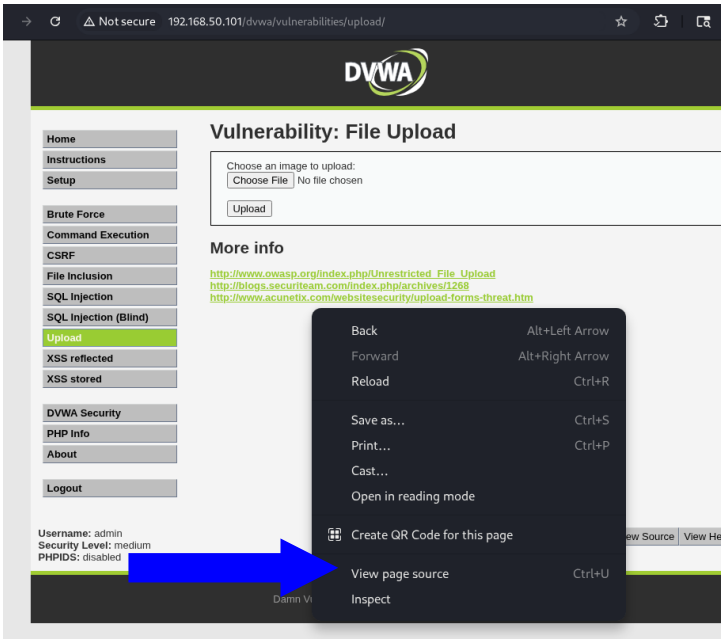
Risultato A livello High, l'upload fallisce per getimagesize() e token CSRF, con GIF89a e token intatto, il bypass riesce, e la shell ultra_shell.php **funziona**, restituendo 'www-data'.



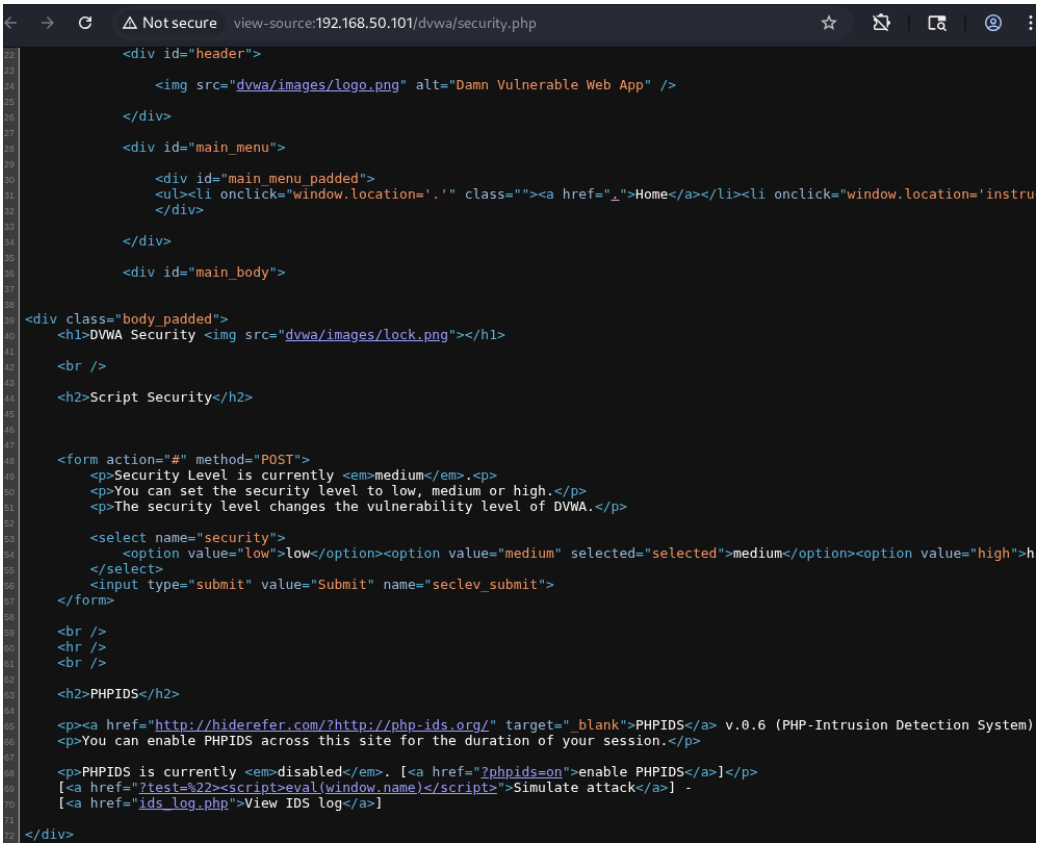
12 [Extra]Analisi del Codice PHP tramite View Source (Medium e High)

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.

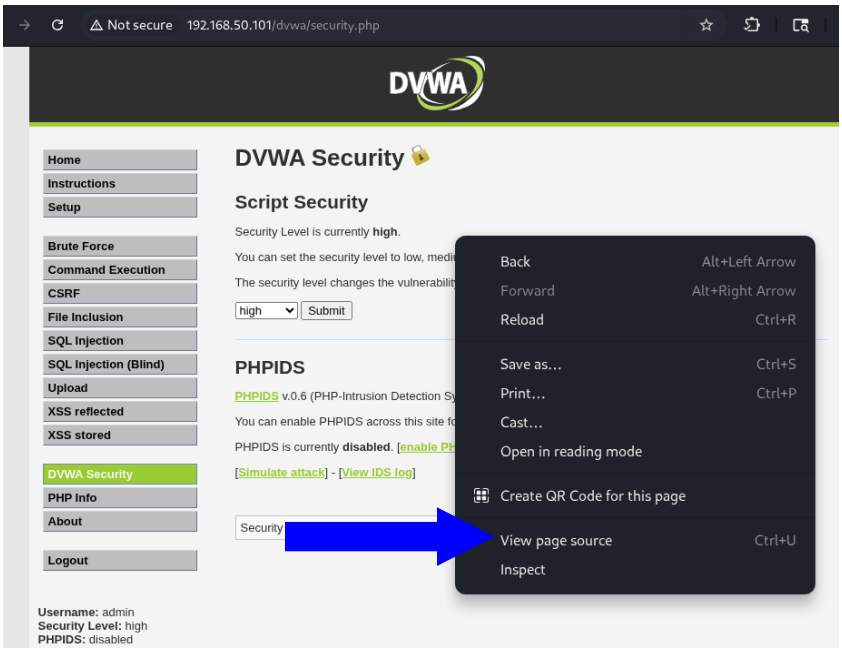
Azione 1 Per **Medium**, apro View Source in File Upload e clicco View page source



- Esamino il codice HTML per dedurre i controlli PHP



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">

                <img src="dvwa/images/logo.png" alt="Damn Vulnerable Web App" />

            </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 <img src="dvwa/images/lock.png"></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.