REPORT ESERCITAZIONE: Sfruttamento Vulnerabilità File Upload (DVWA)

Studente: Rocco Paolo Caccamo

Data: 12 Gennaio 2026

Target: DVWA (Metasploitable 2)

Strumenti: Kali Linux, Burp Suite, Firefox.

1. Executive Summary

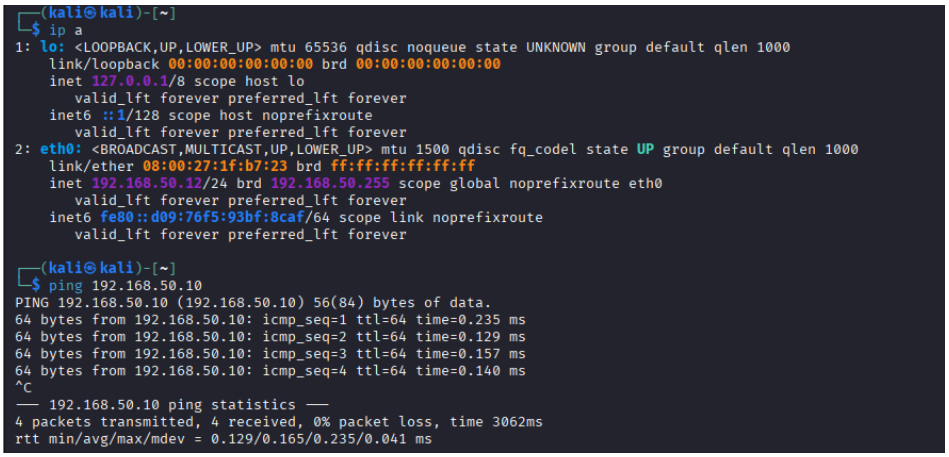
L'attività ha simulato uno scenario di *Red Teaming* volto a sfruttare una vulnerabilità di "Unrestricted File Upload" sulla web application DVWA. L'esercizio ha dimostrato come la mancata sanitizzazione dei file in ingresso permetta a un attaccante di caricare script malevoli (Web Shell), ottenendo l'esecuzione remota di codice (RCE) e compromettendo l'integrità del server.

2. Preparazione dell'Ambiente (Setup)

Prima di procedere con l'attacco, è stata verificata la connettività di rete tra la macchina attaccante (Kali Linux) e la macchina target (Metasploitable 2) all'interno della rete virtuale dedicata.

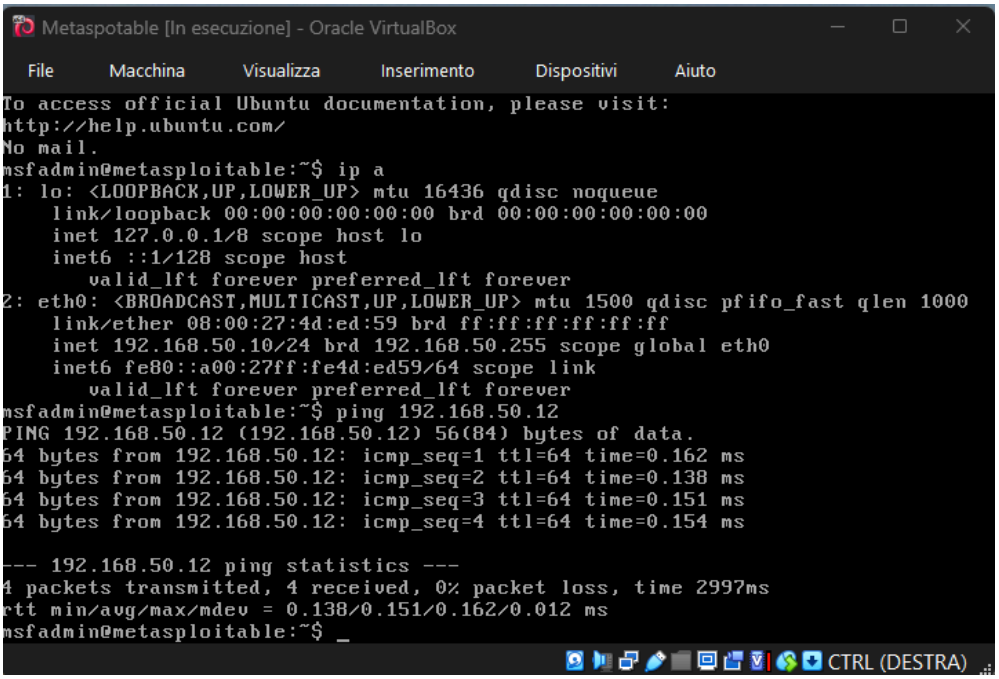
- **IP Attaccante (Kali):** 192.168.50.12
- **IP Target (Metasploitable):** 192.168.50.10

Come evidenziato dallo screenshot sottostante, il comando `ping` ha confermato la raggiungibilità del target.



(Figura 1: Verifica della configurazione di rete su Kali Linux e test di connettività verso il target)

Analogamente, è stata verificata la comunicazione inversa dalla macchina Metasploitable verso Kali per assicurare la stabilità del laboratorio.



(Figura 2: Verifica IP su Metasploitable e ping verso la macchina attaccante)

3. Metodologia di Attacco

3.1 Scansione e Analisi del Traffico

Accedendo alla DVWA, è stata mappata l'applicazione. Utilizzando **Burp Suite** come proxy HTTP, è stato possibile intercettare e storicizzare tutte le richieste effettuate al server. La scheda "HTTP History" mostra la sequenza logica: Login -> Impostazione Livello Sicurezza -> Accesso alla pagina Vulnerabile.

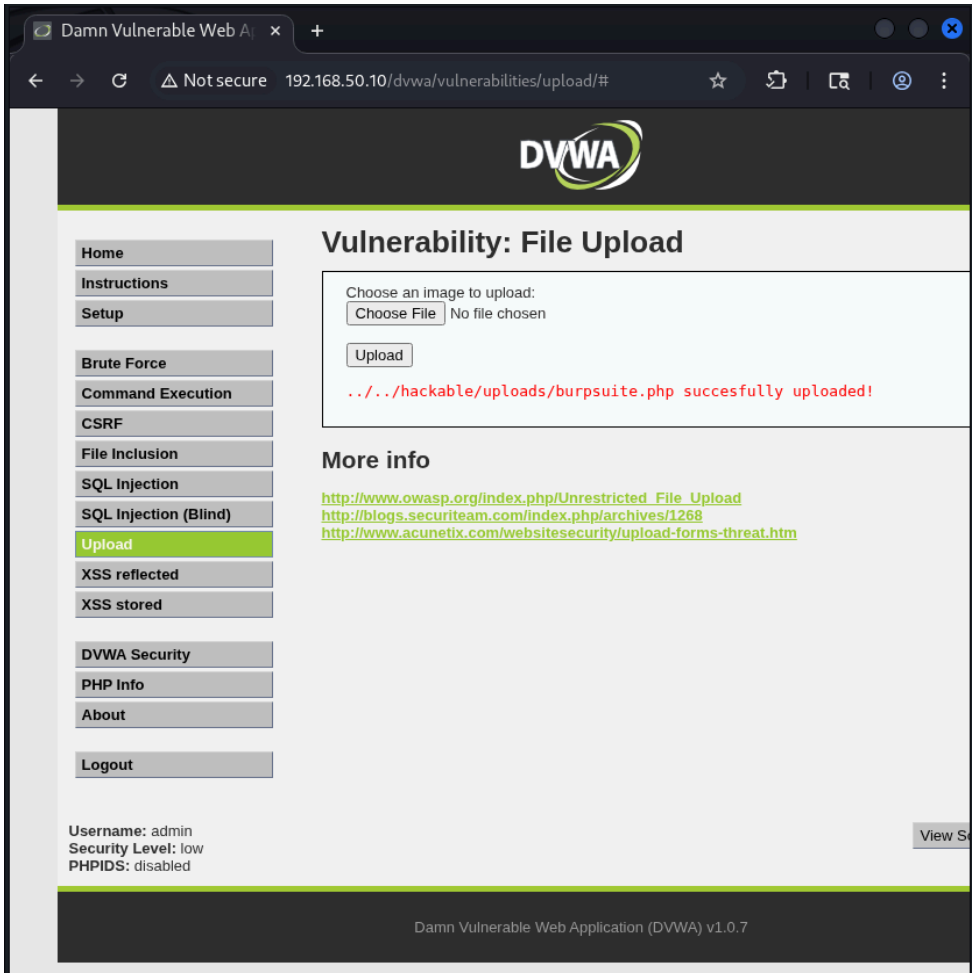
11	http://192.168.50.10	GET	/dvwa/		302	482	HTML		192.168.50
12	http://192.168.50.10	GET	/dvwa/login.php		200	1636	HTML	php	Damn Vulnerable We...	192.168.50
13	http://192.168.50.10	POST	/dvwa/login.php	✓	302	392	HTML	php	192.168.50
14	http://192.168.50.10	GET	/dvwa/index.php		200	4932	HTML	php	Damn Vulnerable We...	192.168.50
15	http://192.168.50.10	GET	/dvwa/security.php		200	4454	HTML	php	Damn Vulnerable We...	192.168.50
16	http://192.168.50.10	GET	/dvwa/security.php		200	4453	HTML	php	Damn Vulnerable We...	192.168.50
17	http://192.168.50.10	POST	/dvwa/security.php	✓	302	427	HTML	php	192.168.50
18	http://192.168.50.10	GET	/dvwa/security.php		200	4534	HTML	php	Damn Vulnerable We...	192.168.50
19	http://192.168.50.10	GET	/dvwa/vulnerabilities/upload/		200	4863	HTML		Damn Vulnerable We...	192.168.50
20	http://192.168.50.10	POST	/dvwa/vulnerabilities/upload/	✓	200	4933	HTML		Damn Vulnerable We...	192.168.50
21	http://192.168.50.10	GET	/dvwa/hackable/uploads/burpsuite....		200	253	HTML	php	192.168.50
22	http://192.168.50.10	GET	/dvwa/hackable/uploads/burpsuite....	✓	200	291	HTML	php	192.168.50

(Figura 3: Storico delle richieste su Burp Suite che mostra il flusso di navigazione fino all'endpoint di upload)

3.2 Creazione e Upload del Payload

È stato generato un payload PHP didattico (`burpsuite.php`) contenente una semplice backdoor: `<?php system($_REQUEST['cmd']); ?>`

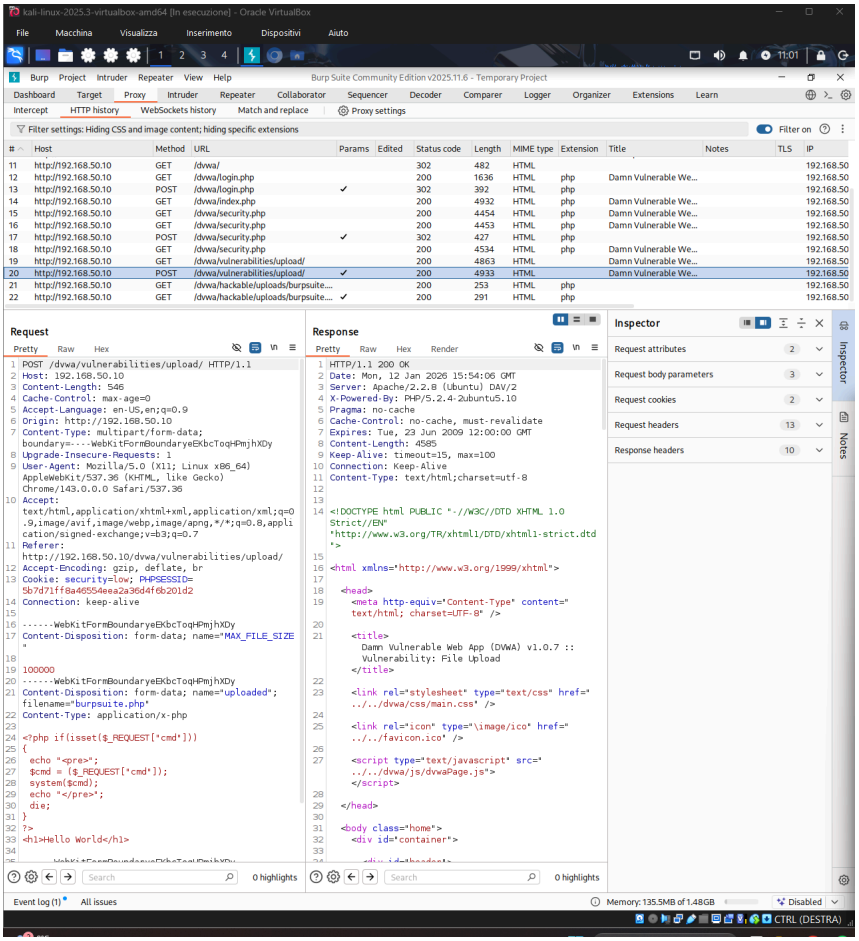
Il file è stato caricato tramite il form di upload della DVWA (Livello Security: Low). Il server ha risposto confermando il percorso di salvataggio: `../../hackable/uploads/burpsuite.php`.



(Figura 4: Conferma visiva del caricamento avvenuto con successo)

3.3 Analisi della Richiesta Malevola

Ispezionando la richiesta `POST` intercettata durante l'upload, si nota che l'applicazione non applica filtri sul `Content-Type` né sull'estensione del file, accettando ed eseguendo codice `application/x-php`.

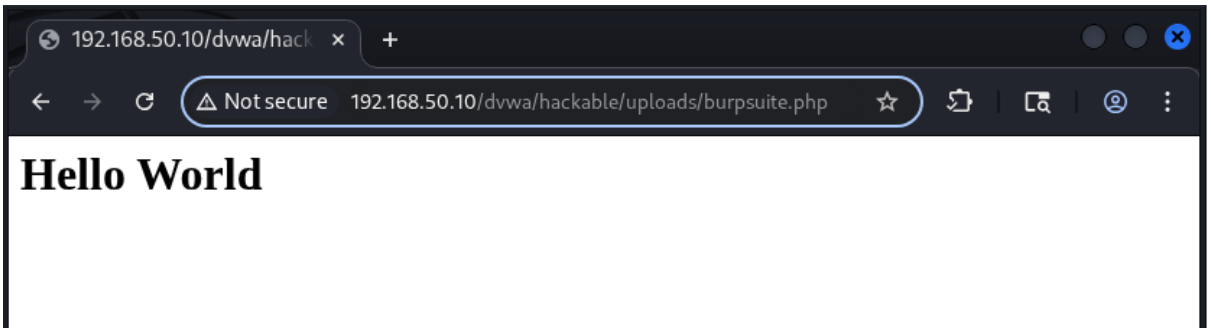


(Figura 5: Dettaglio del pacchetto HTTP in Burp Suite contenente il codice PHP iniettato)

4. Proof of Concept (Exploitation)

4.1 Verifica Esecuzione Codice

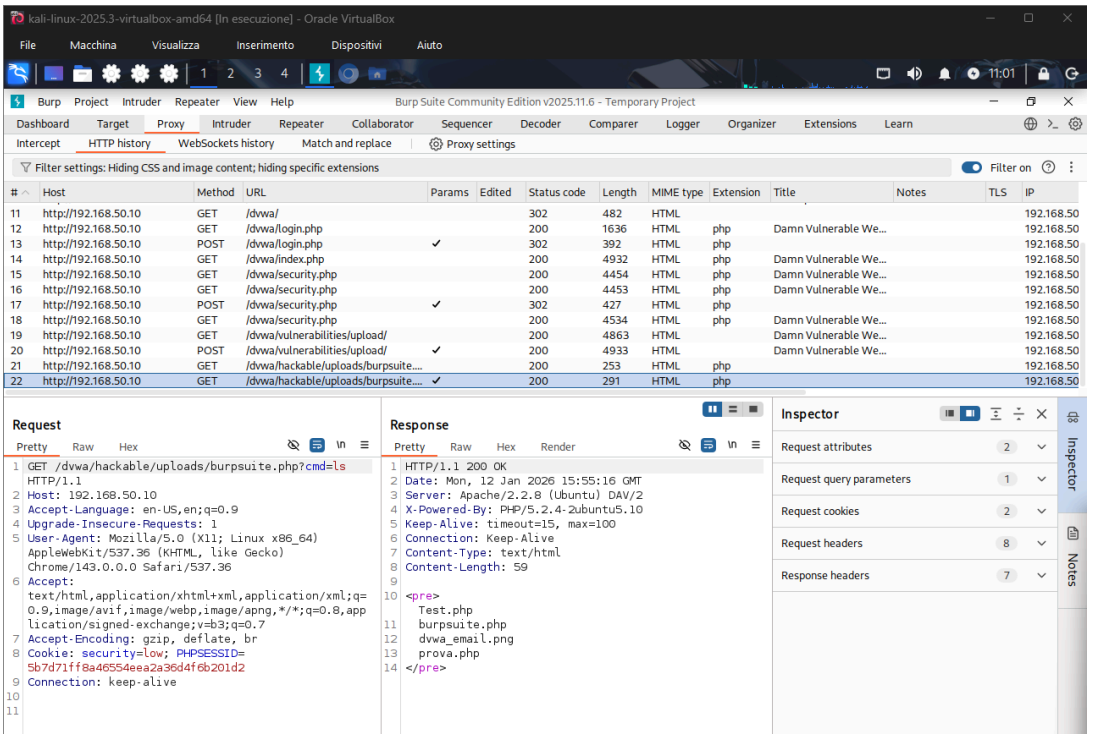
Navigando direttamente all'URL del file caricato, il server ha interpretato il codice PHP, visualizzando l'output HTML previsto ("Hello World"), confermando che il file non viene trattato come semplice testo ma come script eseguibile.



(Figura 6: Rendering del file caricato nel browser)

4.2 Remote Code Execution (RCE)

Per finalizzare l'attacco, è stato passato il comando di sistema `ls` (list directory) tramite il parametro GET `cmd`. La risposta del server, intercettata su Burp, mostra l'elenco dei file presenti nella directory del server web. Questo conferma che l'attaccante ha il controllo sulla macchina ospite.



(Figura 7: Risposta del server che elenca i file di sistema (Test.php, dvwa_email.png, ecc.), prova inconfutabile dell'RCE)

5. Conclusioni e Remediation

L'esercizio ha evidenziato come una configurazione insicura del modulo di upload esponga il server a compromissione totale. **Contromisure suggerite:**

1. Impedire l'esecuzione di script nelle directory di upload.
2. Validare rigorosamente le estensioni dei file (Allow-list).
3. Rinominare i file caricati per oscurarne il percorso originale.