Exploit File upload

Obiettivo

L'attività laboratoriale si prefigge i seguenti scopi tecnici:

- Sviluppare una **Web Shell** in linguaggio PHP (payload) e caricarla sul server target sfruttando la vulnerabilità di **Unrestricted File Upload** presente in DVWA (Damn Vulnerable Web App).
- Utilizzare la shell caricata per ottenere una **RCE (Remote Code Execution)** ed eseguire comandi arbitrari sulla macchina target (Metasploitable).
- Intercettare e analizzare il traffico HTTP/HTTPS generato durante le fasi di upload ed esecuzione mediante l'utilizzo di **Burp Suite**.
- Esaminare le richieste e le risposte del server per comprendere il vettore di attacco e identificare le lacune di sicurezza.

Panoramica

In questo report viene illustrata la metodologia di attacco che consente a un attore malevolo di caricare uno script dannoso (Malware/Web Shell) su un server web vulnerabile tramite un modulo di upload non sicuro. L'analisi si avvale del software **Burp Suite** configurato come Proxy Interceptor. Questo strumento permette di porsi come *Man-in-the-Middle* tra il client e il server, intercettando le richieste, analizzando la struttura dell'HTML e dei pacchetti HTTP, e verificando come i dati vengano processati dal backend. Tale analisi è fondamentale per individuare vulnerabilità sfruttabili.

Sviluppo del Payload (Web Shell PHP)

Per l'esecuzione dell'attacco, è stato necessario scrivere un malware base (Web Shell) utilizzando il linguaggio PHP.

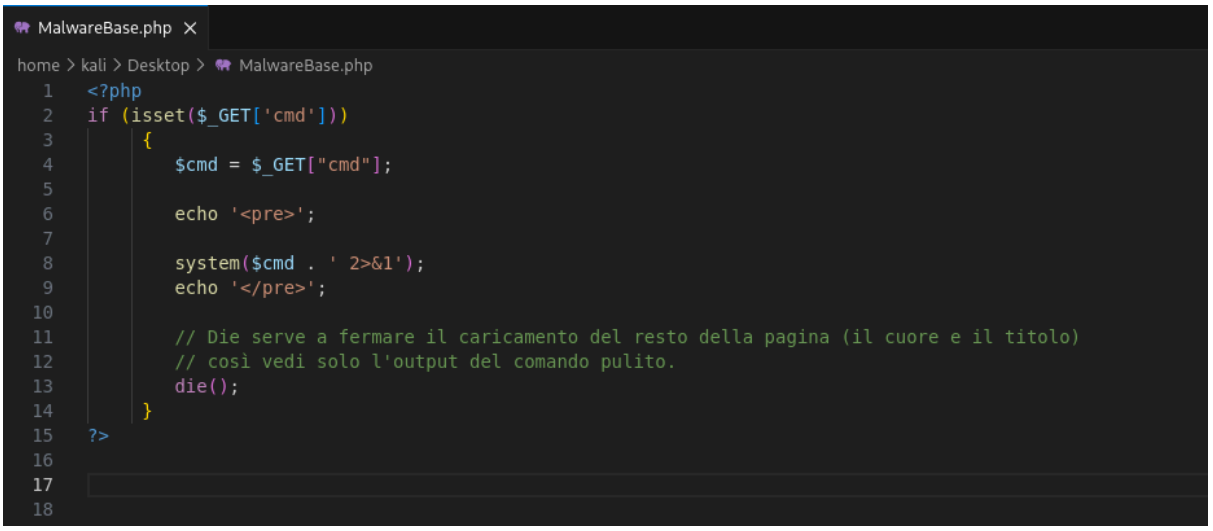

Figura1 : ScreenShot Visual Studio Code del malware PHP

Come evidenziato nello screenshot, lo script PHP utilizza funzioni di sistema per eseguire comandi direttamente sul sistema operativo ospitante. Il codice è strutturato per accettare input dinamici tramite il metodo **GET** . Ciò significa che l'attaccante può passare il comando desiderato direttamente tramite l'URL, e lo script restituirà l'output del comando direttamente nella pagina web.

Intercettazione e Analisi della Richiesta di Upload

Durante la fase di caricamento del file malevolo, la richiesta è stata intercettata tramite Burp Suite per analizzarne la struttura.

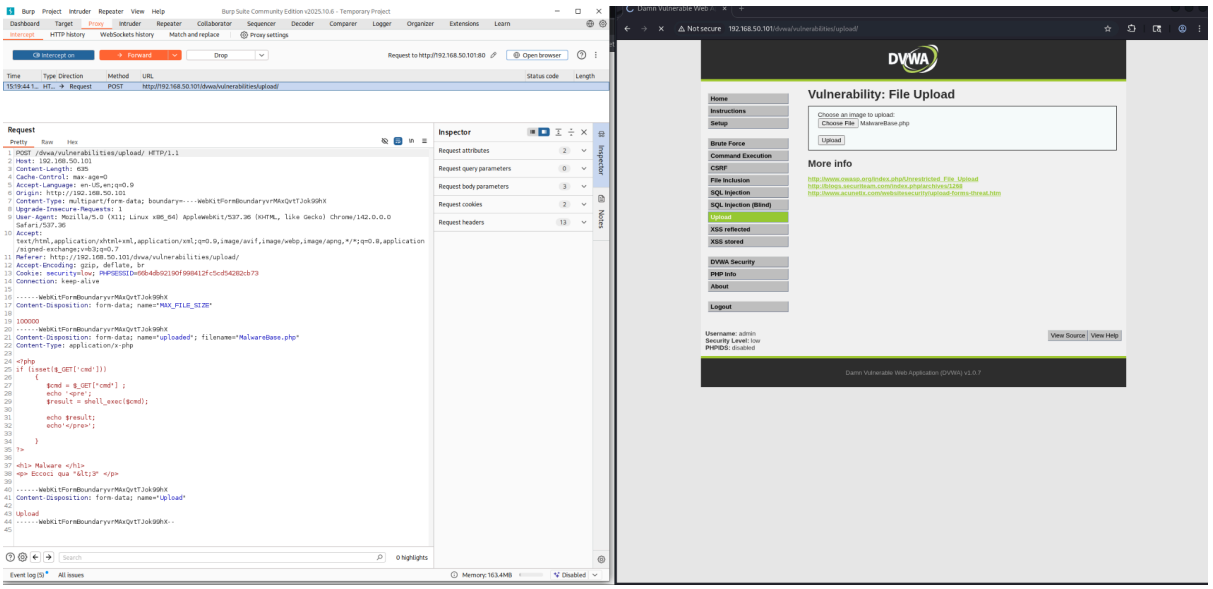

Figura 2: Screenshot Burp Suite durante upload

- La richiesta viene effettuata tramite il metodo **POST**.
- Il contenuto del malware (il codice PHP) è visibile in chiaro nel **Body** della richiesta.
- È possibile osservare gli header HTTP che il server utilizza per validare il tipo di file.

Caricamento del File

Successivamente, si è proceduto all'invio della richiesta al server DVWA tramite la sezione "Upload".

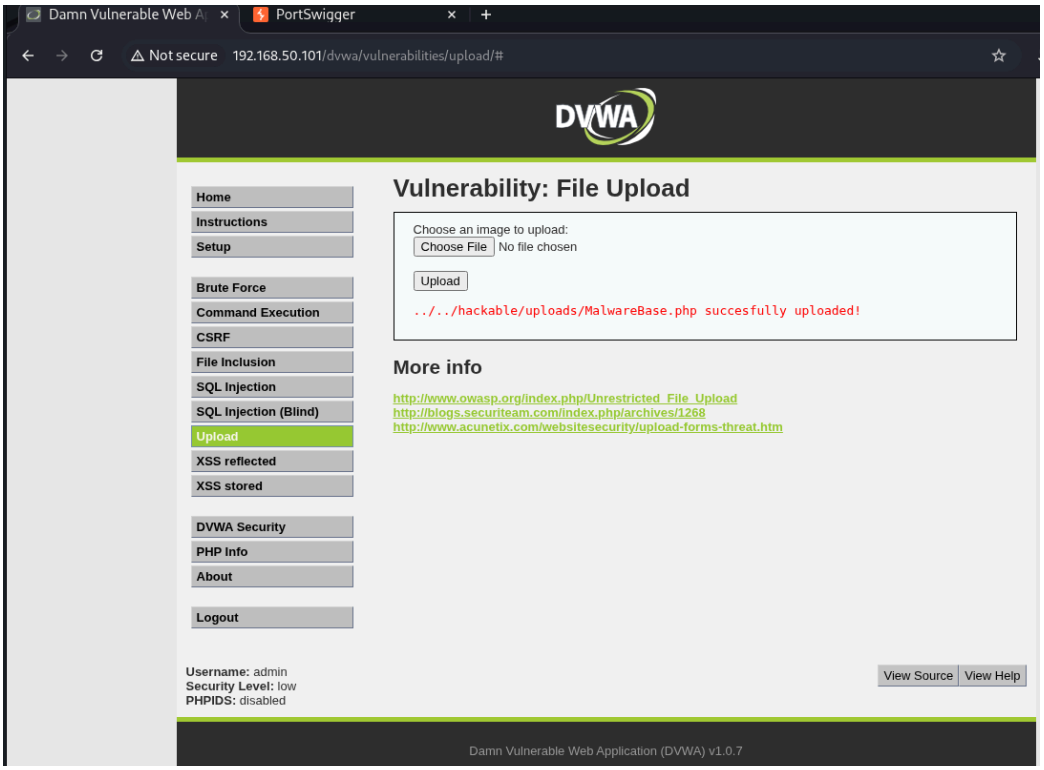

FIgura 3 : Successivamente si pricede a caricare il file PHP creato su DVWA tramite la sezione Upload.

Una volta completato l'upload, l'applicazione restituisce un messaggio di successo. È importante notare una grave vulnerabilità secondaria: la **Path Disclosure**. Il server, infatti, rivela il percorso assoluto o relativo dove il file è stato salvato:
`../../hackable/uploads/MalwareBase.php successfully uploaded!`

Questa informazione è critica per l'attaccante, poiché fornisce l'indirizzo esatto necessario per richiamare ed eseguire il payload.

Esecuzione dei Comandi

Avendo il percorso del file e sapendo come il codice gestisce i parametri GET, è stato possibile interagire con la Web Shell.

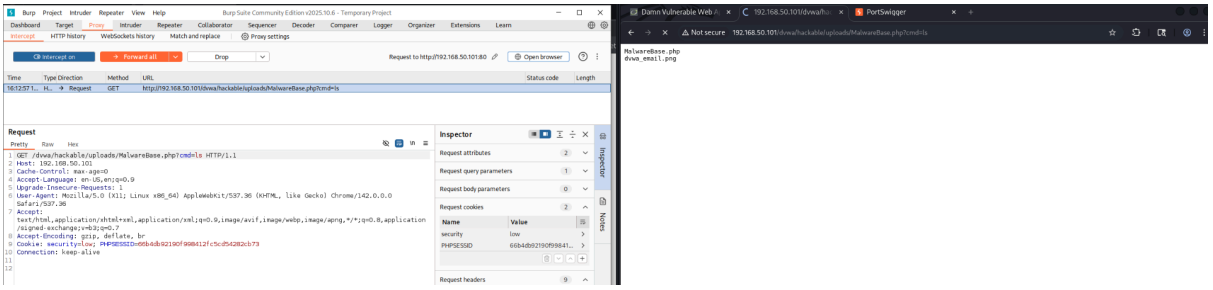

Figura 4: Intercettazione della richiesta HTTP contenente il comando `cmd=ls` tramite parametro GET.

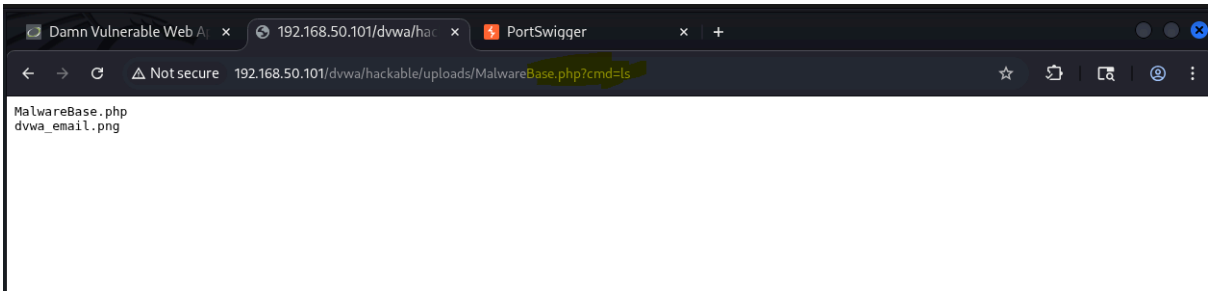

Figura 5: Screenshot con visualizzazione dell'output del comando `ls` direttamente nel browser.

Navigando all'URL del file caricato e aggiungendo la query string *?cmd=*, è stato possibile impartire comandi al sistema operativo sottostante:

- **Reconnaissance:** È stato eseguito il comando *`ls`*, che ha restituito la lista dei file presenti nella directory corrente, confermando l'avvenuta esecuzione remota (RCE).
- **Impact:** In test successivi, è stato verificato che l'utente del web server possiede permessi sufficienti per eseguire comandi distruttivi, come *`rm`*, che permetterebbe la cancellazione di file critici, compromettendo l'integrità e la disponibilità del sistema.

Conclusione

Il report ha dimostrato con successo come l'assenza di controlli adeguati sui file in ingresso (Unrestricted File Upload) permetta a un attaccante di caricare una Web Shell PHP su DVWA. Questo scenario rappresenta un rischio critico: un attaccante potrebbe non solo esfiltrare dati sensibili, ma anche modificare o cancellare file, portando alla compromissione totale del server.