REPORT S6/L2

Obiettivo

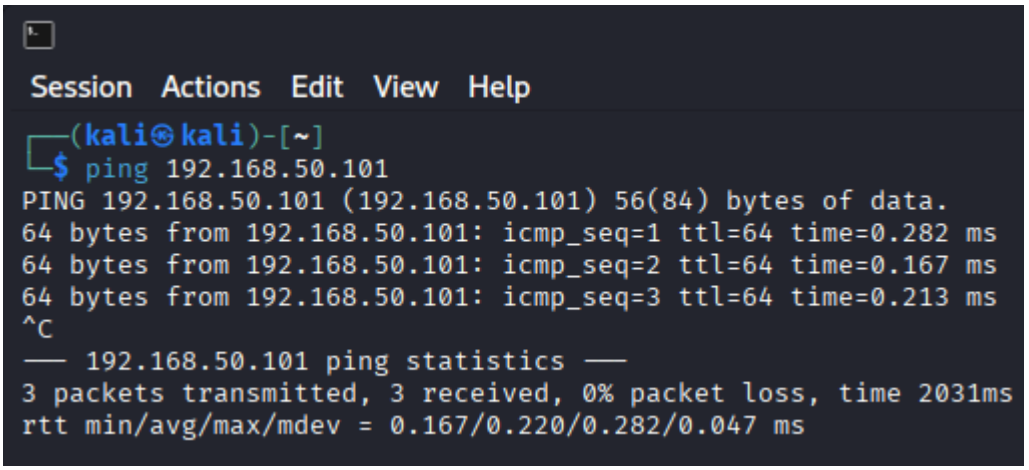
Sfruttare le Vulnerabilità Web (XSS & SQL Injection) su DVWA.

Target: Metasploitable 2 (DVWA - Security Level: LOW)

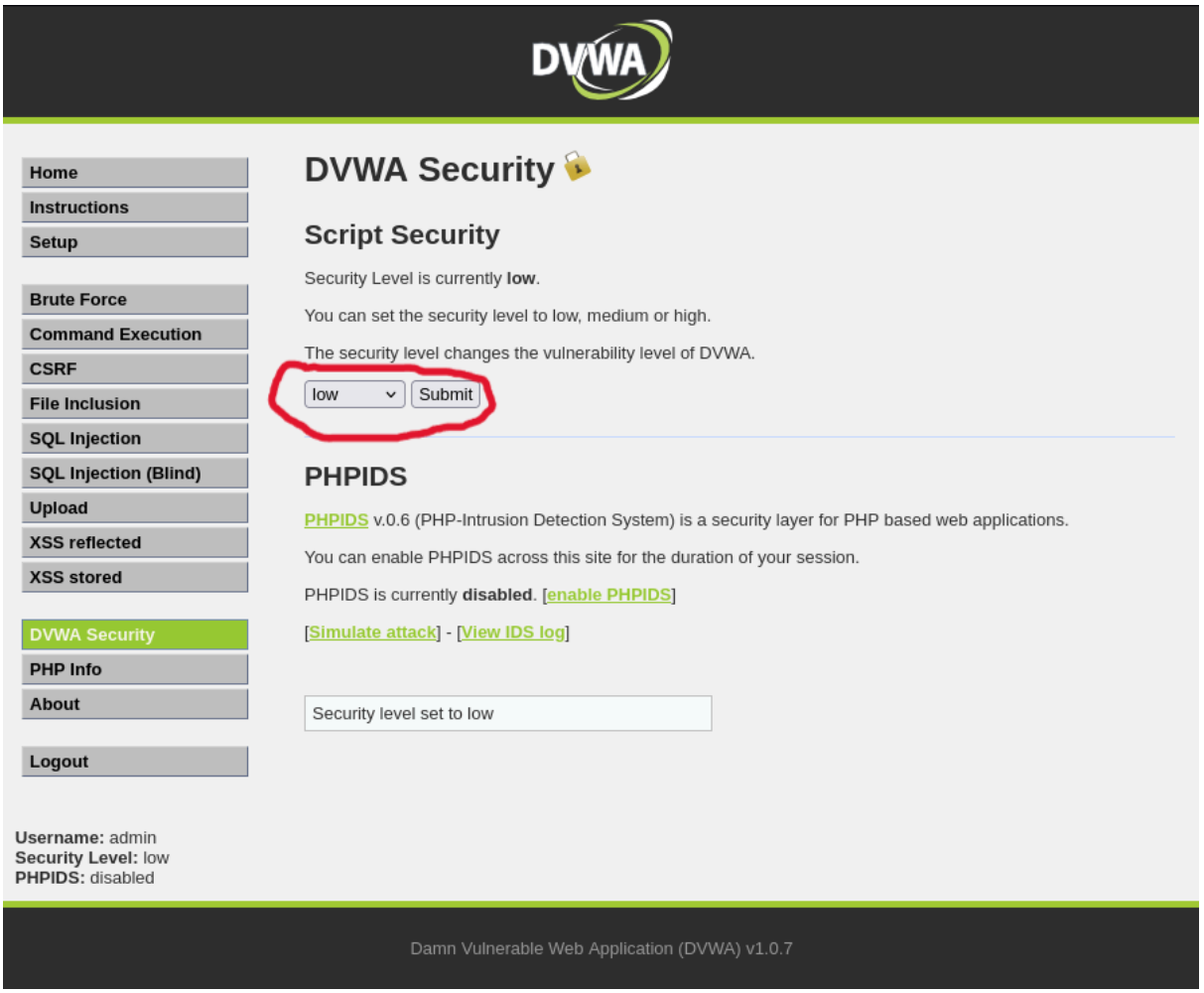
1. Configurazione dell'Ambiente

Obiettivo: Verificare la connettività tra la macchina attaccante (Kali Linux) e la vittima (Metasploitable) e preparare l'ambiente.

Ho verificato la raggiungibilità della macchina target tramite il comando **ping**. La comunicazione è avvenuta con successo, confermando che entrambe le macchine sono sulla stessa rete virtuale.



Successivamente, ho effettuato l'accesso alla web application **DVWA** e ho impostato il livello di sicurezza su **LOW** per simulare un'applicazione priva di controlli di input sanitization.



---

2. Sfruttamento Vulnerabilità XSS Reflected

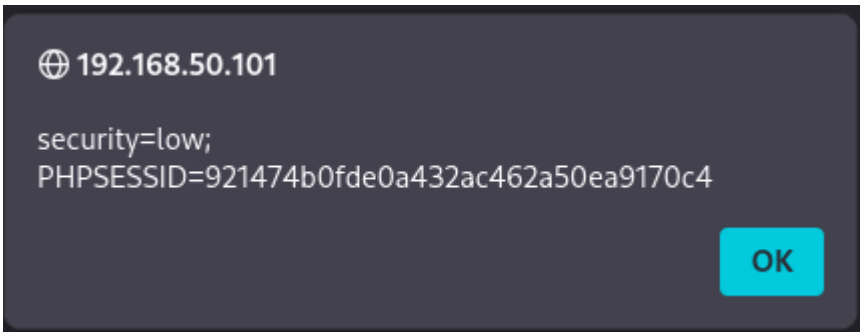
Teoria: L'attacco **Cross-Site Scripting (Reflected)** avviene quando l'applicazione riceve dati in input da una richiesta **HTTP** e li include nella risposta in modo non sicuro.

Esecuzione: Ho navigato nella sezione "**XSS (Reflected)**". L'applicazione chiede un nome in input. Poiché non vi è validazione, inserendo il tag **<script>**, il browser lo interpreta come codice eseguibile.

- **Script utilizzato:** <script>alert(document.cookie);</script>

Risultato: Premendo "Submit", il browser ha eseguito lo script, aprendo un pop-up di avviso con l'impostazione della sicurezza ed il token della sessione (alert).

Se un **attaccante** reale ricevesse quel cookie, potrebbe **rubare** la **sessione dell'amministratore.**



---

3. Sfruttamento Vulnerabilità SQL Injection

Teoria: La **SQL Injection** consiste nell'inserire codice SQL in un campo di input per **alterare** la query che l'applicazione invia al **database**. Nel livello **Low**, l'input non viene "escaped" (sterilizzato).

Esecuzione: Nella sezione "**SQL Injection**", l'applicazione richiede un **User ID**. Ho utilizzato l'operatore **UNION** per combinare i risultati della query originale con una mia query personalizzata per estrarre informazioni sensibili sul database.

- **Payload utilizzato:** 1' UNION SELECT user(), database()#

Analisi del Payload:

- **`1'`** : Chiude la stringa dell'ID originale.
- **`UNION SELECT`** : Unisce un nuovo set di risultati.
- **`user(), database()`** : Funzioni SQL per mostrare l'utente corrente e il nome del DB.
- **`#`** : Commenta il resto della query originale per evitare errori di sintassi.

Risultato: L'applicazione ha restituito, al posto del cognome, l'utente con cui gira il database e il nome del database stesso.

---

4. Conclusioni

L'esercizio ha dimostrato come la mancanza di validazione degli input (input sanitization) permetta a un attaccante di eseguire codice javascript (XSS) o di interrogare il database in modo non autorizzato (SQLi).