Report di Laboratorio Cybersecurity: Exploiting DVWA

---

1. Obiettivo

L'obiettivo di questo laboratorio era configurare un ambiente di penetration testing virtualizzato e sfruttare le vulnerabilità web sulla **Damn Vulnerable Web Application (DVWA)**. Nello specifico, l'assegnazione richiedeva di dimostrare con successo lo sfruttamento di:

- **Cross-Site Scripting (XSS):** Sia nella variante *Reflected* che *Stored*.
- **SQL Injection (SQLi):** Esecuzione di injection manuale e sfruttamento automatizzato utilizzando **SQLMap**.

2. Ambiente di Laboratorio

- **Macchina Attaccante:** Kali Linux (`192.168.50.102`)
- **Macchina Target:** Metasploitable 2 hosting DVWA (`192.168.50.101`)
- **Livello di Sicurezza:** Impostato su **"Low"** (Basso) per simulare un'applicazione priva di sanitizzazione degli input.

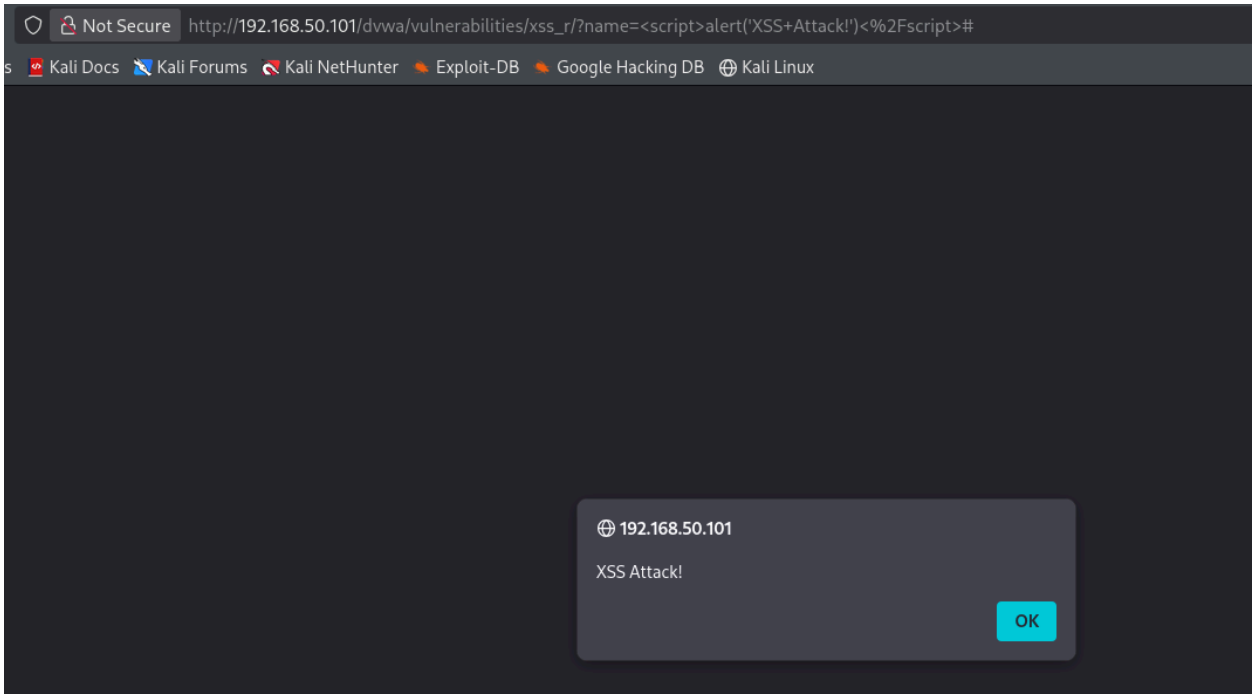
---

3. Esecuzione e Risultati

3.1. Reflected Cross-Site Scripting (XSS)

L'XSS Reflected si verifica quando l'input dell'utente viene restituito dall'applicazione web immediatamente senza essere memorizzato o sanitizzato.

- **Vettore di Attacco:** Il campo di input "What's your name?".
- **Payload Utilizzato:** `<script>alert('XSS Attack!')</script>`
- **Risultato:** Il browser ha eseguito il payload JavaScript iniettato, visualizzando un pop-up di avviso. Questo conferma che l'applicazione è vulnerabile all'esecuzione di codice lato client.

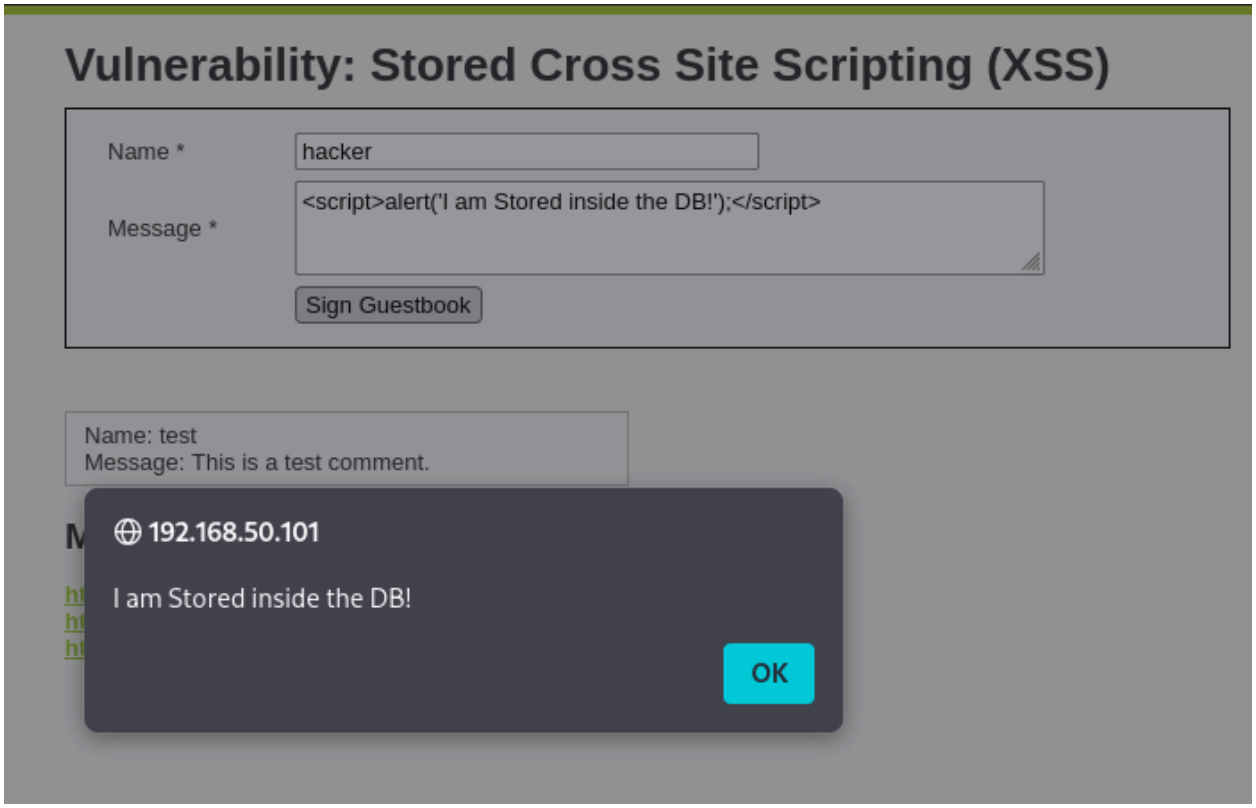


(Figura 1: Esecuzione riuscita dell'XSS Reflected che mostra il pop-up di avviso)

3.2. Stored Cross-Site Scripting (XSS)

L'XSS Stored (o Persistente) si verifica quando l'input dannoso viene salvato nel database del server e successivamente mostrato ad altri utenti che visualizzano la pagina.

- **Vettore di Attacco:** I campi "Name" e "Message" del "Guestbook".
- **Payload Utilizzato:** `<script>alert('I am Stored inside the DB!');</script>`
- **Risultato:** Dopo l'invio della voce, lo script è stato salvato. Ogni volta che la pagina del Guestbook viene caricata, lo script dannoso viene eseguito automaticamente.

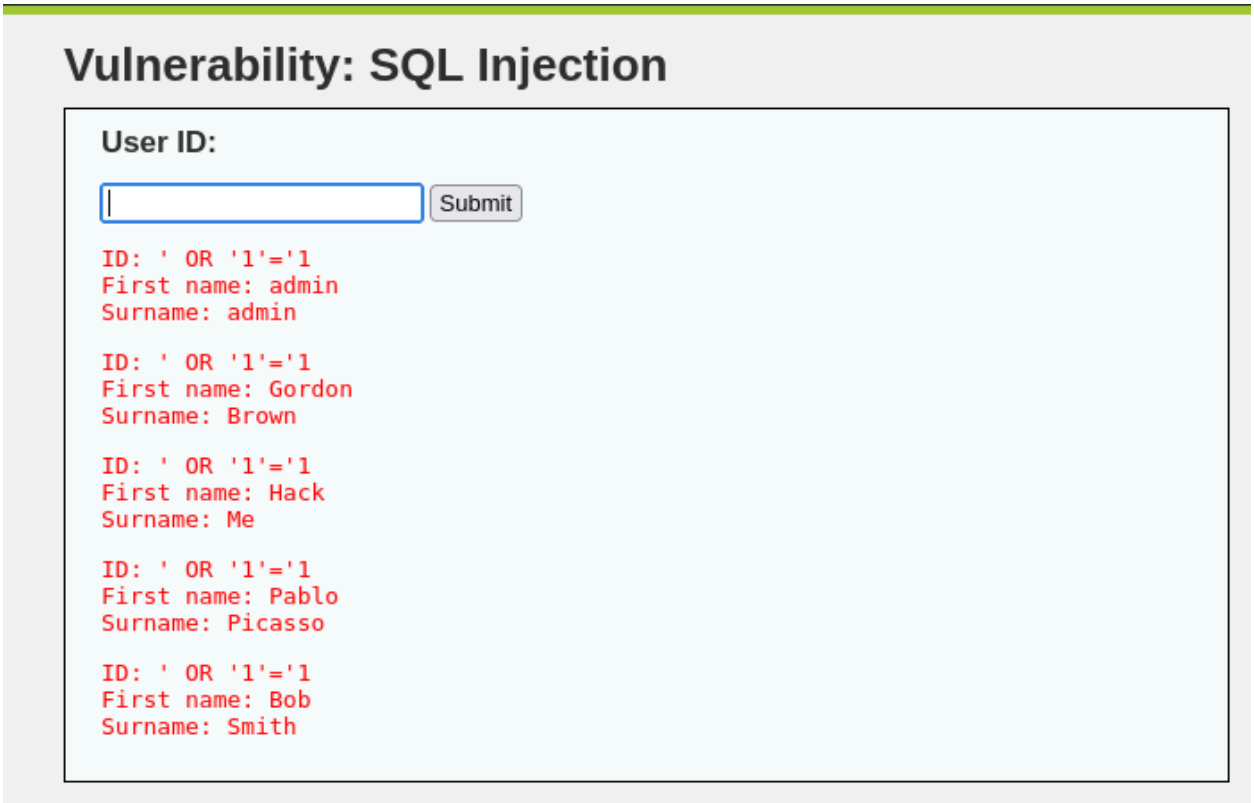


(Figura 2: XSS Stored che attiva un avviso immediatamente al caricamento del Guestbook)

3.3. SQL Injection Manuale

La SQL Injection è una vulnerabilità in cui un attaccante manipola la query del database iniettando comandi SQL dannosi.

- **Vettore di Attacco:** Il campo di input "User ID".
- **Logica:** Iniettando una condizione logica **OR** che è sempre vera (`' 1 ' = ' 1`), forziamo il database a restituire tutti i record invece di uno solo.
- **Payload Utilizzato:** `' OR '1'='1`
- **Risultato:** L'applicazione ha mostrato ogni utente presente nel database (admin, Gordon Brown, ecc.) anziché un singolo ID specifico.



(Figura 3: SQL Injection manuale che estrae tutti i record utente)

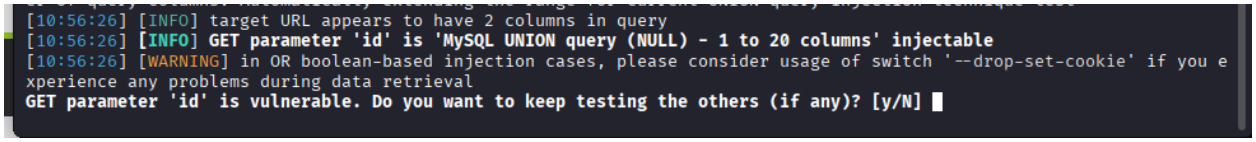
3.4. Sfruttamento Automatizzato con SQLMap

Per dimostrare uno sfruttamento avanzato, è stato utilizzato lo strumento **SQLMap** per automatizzare il rilevamento e l'estrazione dei dati.

Fase 1: Rilevamento ed Enumerazione

Abbiamo lanciato SQLMap contro l'URL vulnerabile, fornendo il cookie di sessione per l'autenticazione.

- **Comando:** `sqlmap -u "http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=...; security=low" --dbs`
- **Risultato:** SQLMap ha identificato il DBMS di backend come **MySQL** ed elencato i database disponibili, incluso il database target **`dvwa`**.



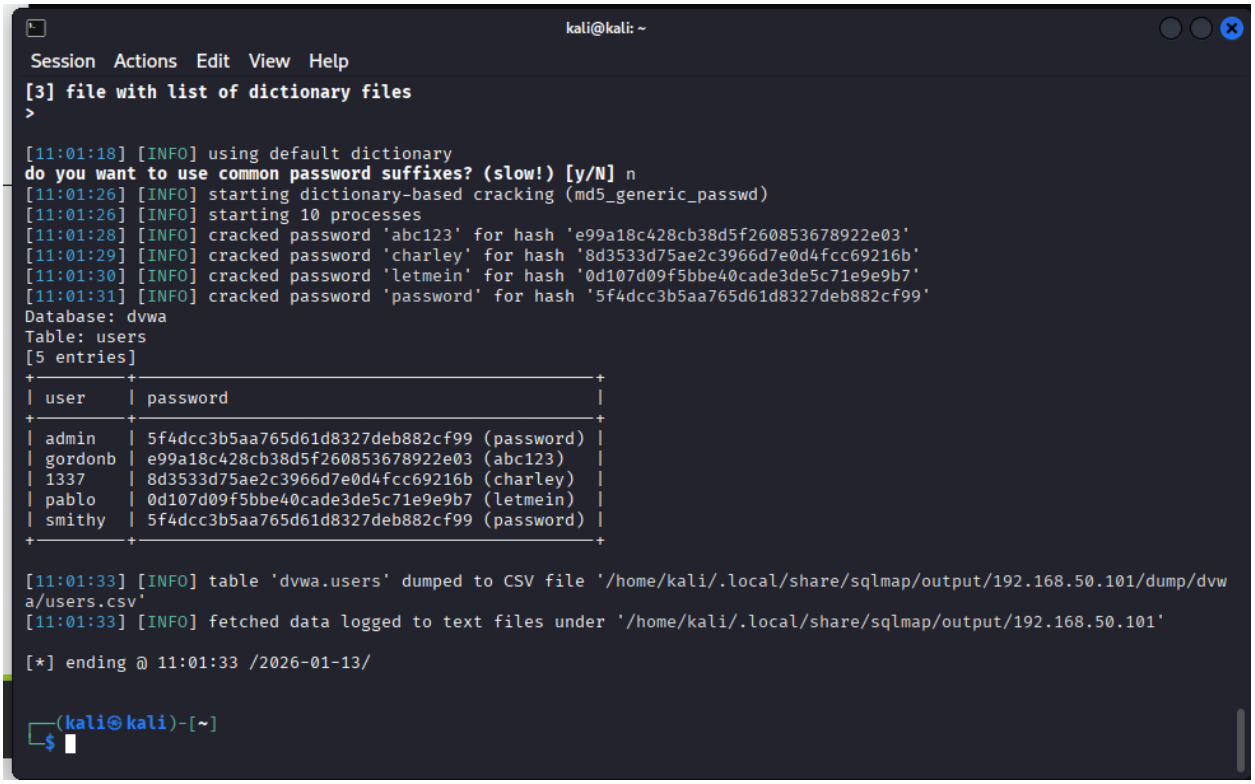
(Figura 4: SQLMap conferma che il parametro 'id' è vulnerabile)

Fase 2: Estrazione Dati e Cracking

Abbiamo mirato alla tabella `users` all'interno del database `dvwa` per estrarre le credenziali.

- **Comando:** `sqlmap ... -D dvwa -T users -C user,password --dump`
- **Risultato:** SQLMap ha scaricato gli hash degli utenti. Successivamente ha eseguito un attacco a dizionario per crackare gli **hash MD5** e convertirli in testo in chiaro.

(Figura 5: SQLMap enumera le colonne della tabella users)



(Figura 6: Risultato Finale - SQLMap ha crackato con successo tutte le password, rivelando le credenziali di amministratore)

---

4. Conclusione

Questo laboratorio ha dimostrato con successo i rischi critici di una gestione impropria degli input.

1. **XSS** ha permesso l'esecuzione arbitraria di codice nel browser della vittima.
2. **SQL Injection** ha permesso la compromissione completa del database, portando all'esposizione di credenziali amministrative sensibili (`admin` / `password`).