Vulnerability Assessment & Penetration Test – SQL Injection

Oggetto: Analisi della sicurezza Web Application (DVWA) – Vulnerabilità SQL Injection

Autori: datashields

Sintesi

Il presente documento documenta l'attività di laboratorio svolta per analizzare la sicurezza dell'applicazione web DVWA ospitata su macchina **Metasploitable 2**. È stato condotto un ***Vulnerability Assessment*** e un successivo ***Penetration Test*** focalizzato sulla vulnerabilità di **SQL Injection (SQLi)**.

L'attività è stata eseguita su due livelli di difficoltà (Low e Medium) e ha avuto esito positivo, permettendo l'enumerazione completa del database, l'esfiltrazione delle credenziali utente (*hash*) e il successivo *cracking* delle password.

Scopo del test e analisi dello scenario

Scenario e Obiettivi

L'attività si svolge in un ambiente controllato costituito da due macchine virtuali attestate sulla stessa sottorete locale.

- **Attacker:** Kali Linux (**192.168.13.100**), utilizzata per l'analisi e l'attacco.
- **Target:** Metasploitable 2 (**192.168.13.150**), ospitante l'applicazione vulnerabile DVWA.
- **Focus della vulnerabilità:** SQL Injection sulla pagina di ricerca User ID.

L'obiettivo principale è sfruttare la mancata (o parziale) validazione dell'input utente per manipolare le query SQL inviate al database *backend*. Questo permette di bypassare i controlli di autenticazione o, come in questo caso, estrarre dati sensibili arbitrari (***Data Exfiltration***).

Strumenti Utilizzati

- **Burp Suite (Community Edition):** Piattaforma integrata per il test di sicurezza delle applicazioni web. Utilizzata in modalità **Proxy/Repeater** per intercettare e manipolare le richieste HTTP (metodo POST) e bypassare i controlli lato client.
- **Firefox Browser:** Utilizzato per l'interazione con l'interfaccia web e l'iniezione diretta (livello Low).
- **John the Ripper:** Strumento di password cracking utilizzato nella fase di post-exploitation per decifrare gli *hash MD5* esfiltrati.

Svolgimento: Livello Low (Basic Exploitation)

Analisi

Nel livello di sicurezza "**Low**", l'applicazione presenta una casella di input testuale per la ricerca di utenti tramite ID. Il codice backend accetta l'input e lo concatena direttamente nella query SQL senza alcuna sanitizzazione.

Exploitation

1. **Verifica Vulnerabilità:** L'inserimento del carattere apice (') genera un errore di sintassi SQL ("*You have an error in your SQL syntax*"), confermando che l'input viene interpretato dal DBMS.

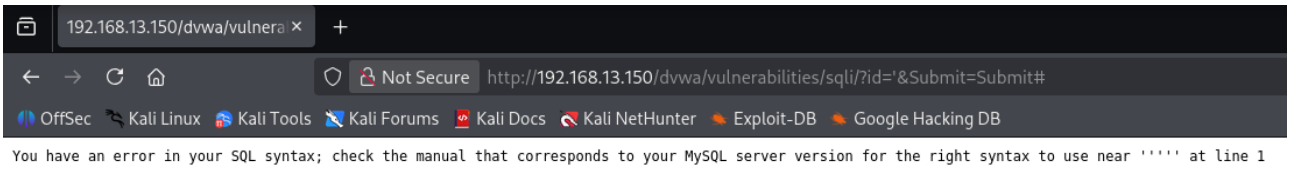

Figura 1 Errore di sintassi che verifica presenza vulnerabilità

2. **Enumerazione:** Tramite iniezione di comandi `UNION SELECT`, è stato possibile determinare il numero di colonne e mappare la struttura del database (user(), database()).

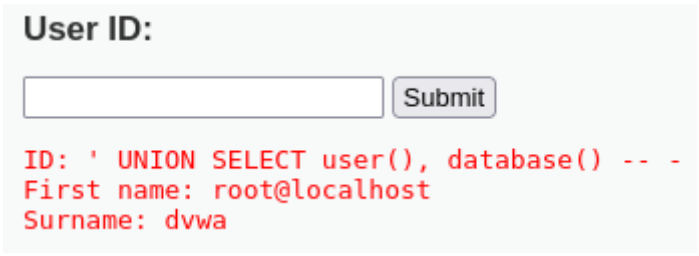

Figura 2 Scoperta nome database

3. **Data Extraction:** Utilizzando la query iniettata `' UNION SELECT user, password FROM users -- -`, è stato eseguito il dump della tabella users, recuperando username e password hashate.

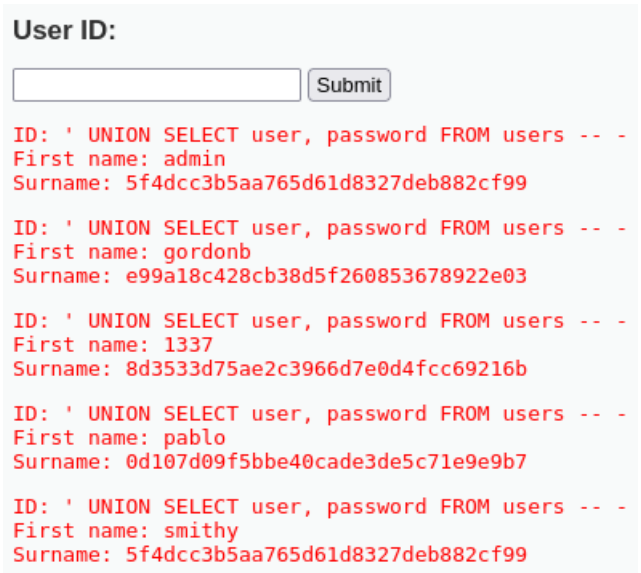

Figura 3 dump del db

Svolgimento: Livello Medium (Bypass & Obfuscation)

Analisi e Hardening Rilevato

Nel livello "**Medium**", l'applicazione introduce due meccanismi di difesa:

1. **Frontend:** La casella di testo pretende l'utilizzo di ID numerici.
2. **Backend:** Viene utilizzata la funzione ***mysql_real_escape_string()*** che esegue l'escape dei caratteri speciali (come gli apici '), rendendo inefficaci le injection basate su stringhe classiche (es. ' OR 1=1).

Bypass e Exploitation

L'analisi ha rivelato che il parametro id è trattato come intero (*integer*), rendendo inutile l'uso degli apici. Tuttavia, il filtro sugli apici impedisce di specificare stringhe nella clausola WHERE (es. WHERE user='Pablo').

La tecnica di attacco adottata ha previsto:

1. **Intercettazione:** Utilizzo di **Burp Suite** per intercettare la richiesta POST HTTP prima dell'invio al server, aggirando il limite del menu a tendina.
2. **Hex Encoding (Obfuscation):** Per bypassare il filtro sugli apici nella clausola WHERE, le stringhe target ("Pablo", "Picasso") sono state convertite in esadecimale (es. 0x5061626c6f).
3. **Payload Finale:** È stata iniettata la seguente query direttamente nel pacchetto HTTP intercettato:
   id=1%20UNION%20SELECT%20user%2C%20password%20FROM%20users%20WHERE%20first_name%20%3D%200x5061626C6F%20AND%20last_name%20%3D%200x5069636173736F
   Questa query è URL encoded e presenta gli esadecimali per ogni clausola presentante =' '

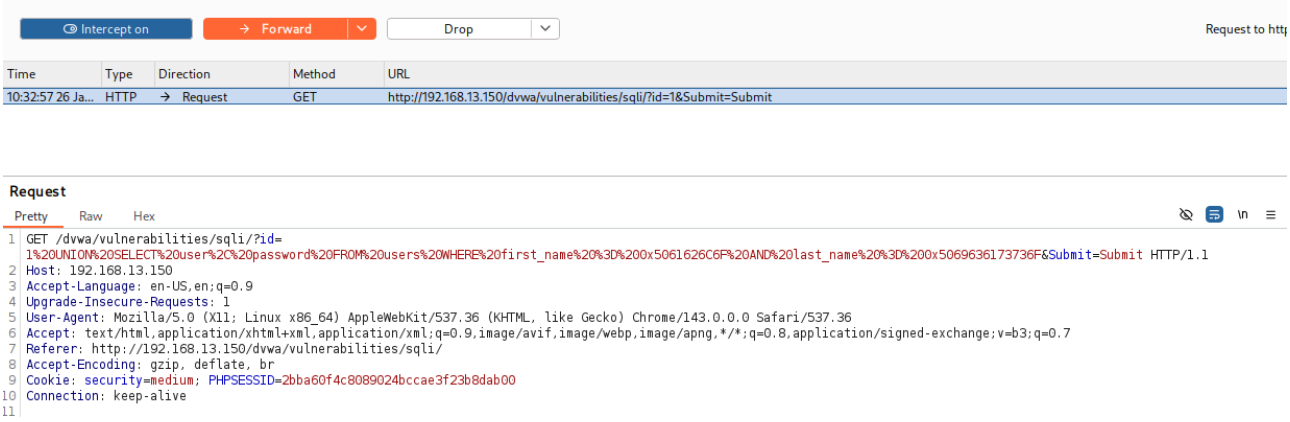

Figura 4 Payload finale

L'attacco ha avuto successo, permettendo l'esfiltrazione dei dati nonostante i meccanismi di protezione attivi.

Advanced Post-Exploitation (Cross-Database Enumeration)

Oltre al database dell'applicazione (dvwa), la vulnerabilità SQL Injection ha permesso di interrogare lo schema globale del DBMS per mappare l'intera infrastruttura dati presente sul server.

Mapping dell'Infrastruttura

Sfruttando l'accesso in lettura al database di sistema information_schema, è stato possibile ottenere la lista completa di tutti i database e le relative tabelle presenti sull'istanza MySQL.

- **Query Iniettata (Medium):**
  `1 UNION SELECT table_schema, table_name FROM information_schema.tables #`
- **Risultato:** L'output ha rivelato la presenza di database critici quali mysql, information_schema e altri database applicativi (owasp10, tikiwiki, tikiwiki195).

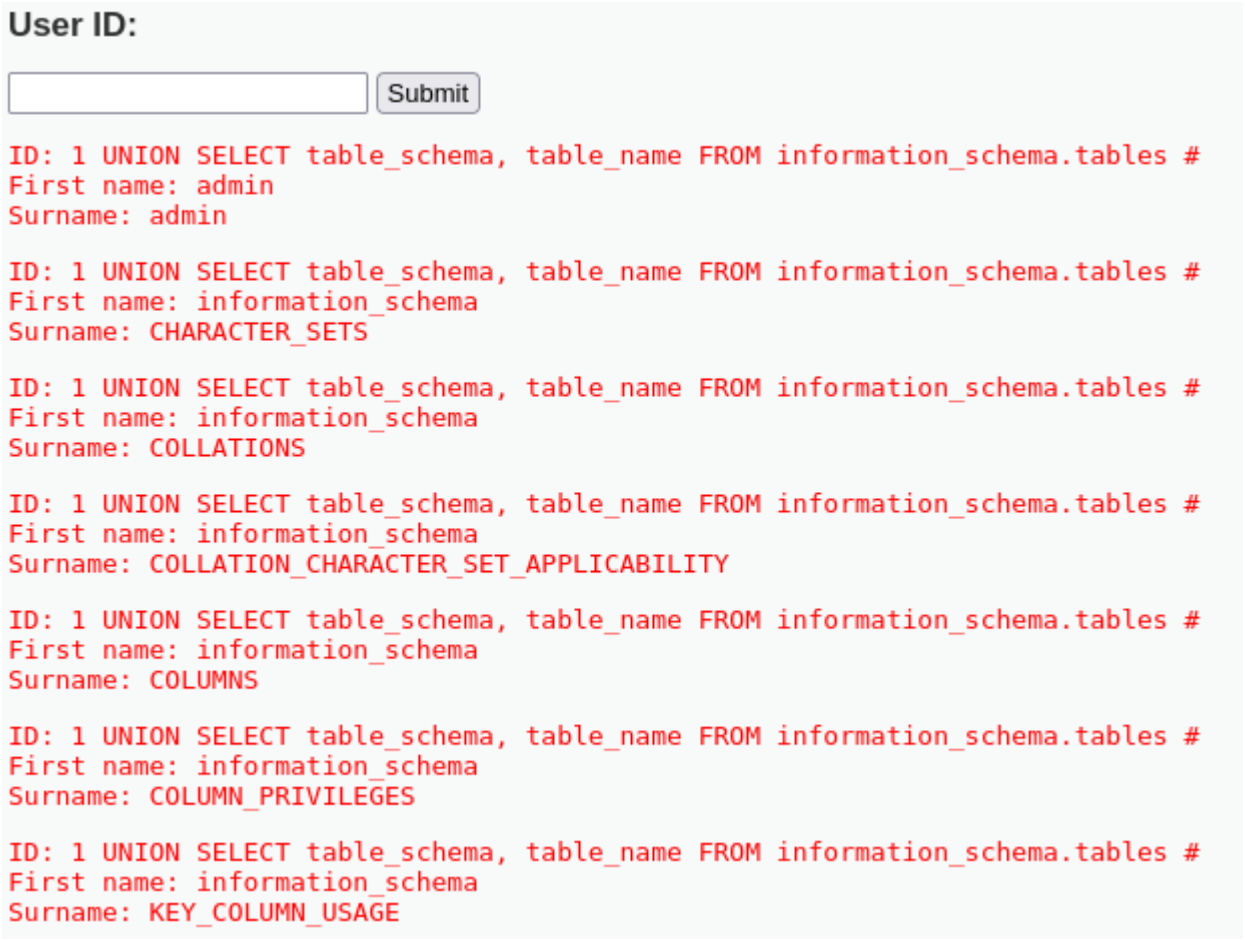

Figura 5 Output query di mappaggio dell'infrastruttura

Accesso a Database di Sistema

Una volta identificato il database di sistema **mysql**, è stata eseguita una query mirata per estrarre le credenziali degli utenti amministrativi del server database (root), utilizzando la sintassi database.tabella.

- **Query Iniettata:**

  `1 UNION SELECT user, password FROM mysql.user #`

- **Impatto:** Questa operazione ha permesso l'esfiltrazione delle password degli amministratori di sistema, in questo caso vuote, elevando il rischio da compromissione della singola applicazione a compromissione dell'intero server database (Privilege Escalation orizzontale/verticale).

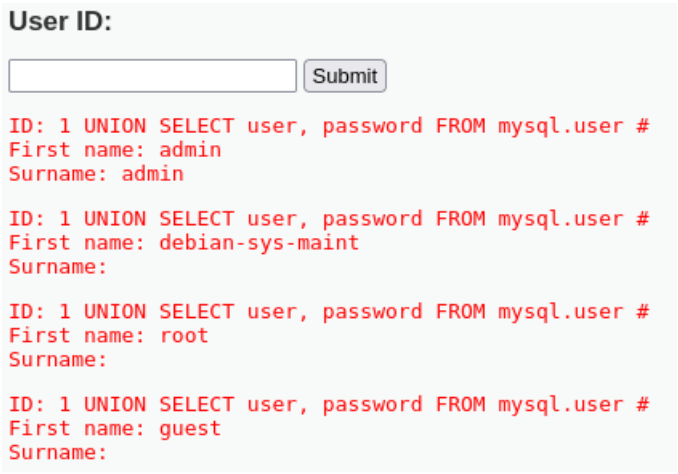

Figura 6 Output query di esfiltrazione password mysql

Post-Exploitation (Password Cracking)

Una volta ottenuti i dati dal database, si è proceduto all'analisi delle credenziali esfiltrate.

- **Target:** Utente pablo
- **Hash Esfiltrato:** 0d107d09f5bbe40cade3de5c71e9e9b7
- **Analisi:** L'hash è stato identificato come formato MD5 (Raw).

Utilizzando **John the Ripper** con wordlist *rockyou.txt*, l'hash è stato crackato con successo:

```
$ john --format=Raw-MD5 pablo.txt
```

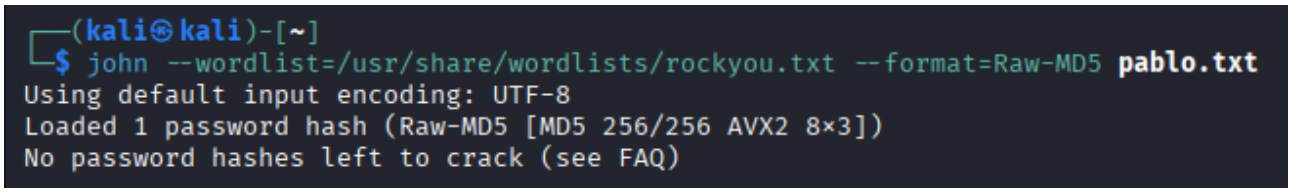

Figura 7 john cracking

Abbiamo poi visualizzato il risultato tramite il comando:

```
$ john --show pablo.txt
```

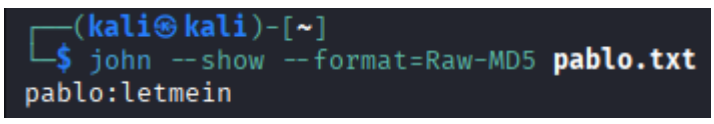

Figura 8 jonh risultato

La password in chiaro recuperata è: letmein.

Conclusioni e Mitigazione

L'attività ha dimostrato che meccanismi di difesa parziali (come limitazioni interfaccia o sanitizzazione dei soli caratteri speciali) non sono sufficienti a prevenire la SQL Injection se non accompagnati da una corretta gestione delle query.

Si raccomandano le seguenti azioni di mitigazione:

1. **Prepared Statements (Query Parametrizzate):** L'unica difesa definitiva. Utilizzare librerie (come PDO in PHP) che separano rigorosamente la struttura della query dai dati, impedendo all'input utente di alterare la logica SQL.
2. **Input Validation:** Validare rigorosamente il tipo di dato atteso lato server (es. assicurarsi che un ID sia strettamente numerico tramite *casting*).
3. **Principio del Minimo Privilegio:** Assicurarsi che l'utente del database utilizzato dall'applicazione web abbia solo i permessi strettamente necessari, limitando l'accesso a tabelle di sistema (information_schema).