

Analisi di un Attacco SQL Injection: Report Completo

Questo documento presenta un'analisi dettagliata di un attacco SQL Injection registrato nel file PCAP SQL_Lab.pcap. L'obiettivo è esaminare il traffico HTTP con Wireshark, identificare i punti chiave dell'iniezione, la risposta del database e il recupero delle credenziali tramite hash cracking. Il report include screenshot delle attività svolte, conclusioni sull'importanza della sicurezza dei database e metodi per prevenire attacchi SQL Injection.

L by Luca Tavani

Identificazione degli IP Coinvolti e Apertura del File PCAP

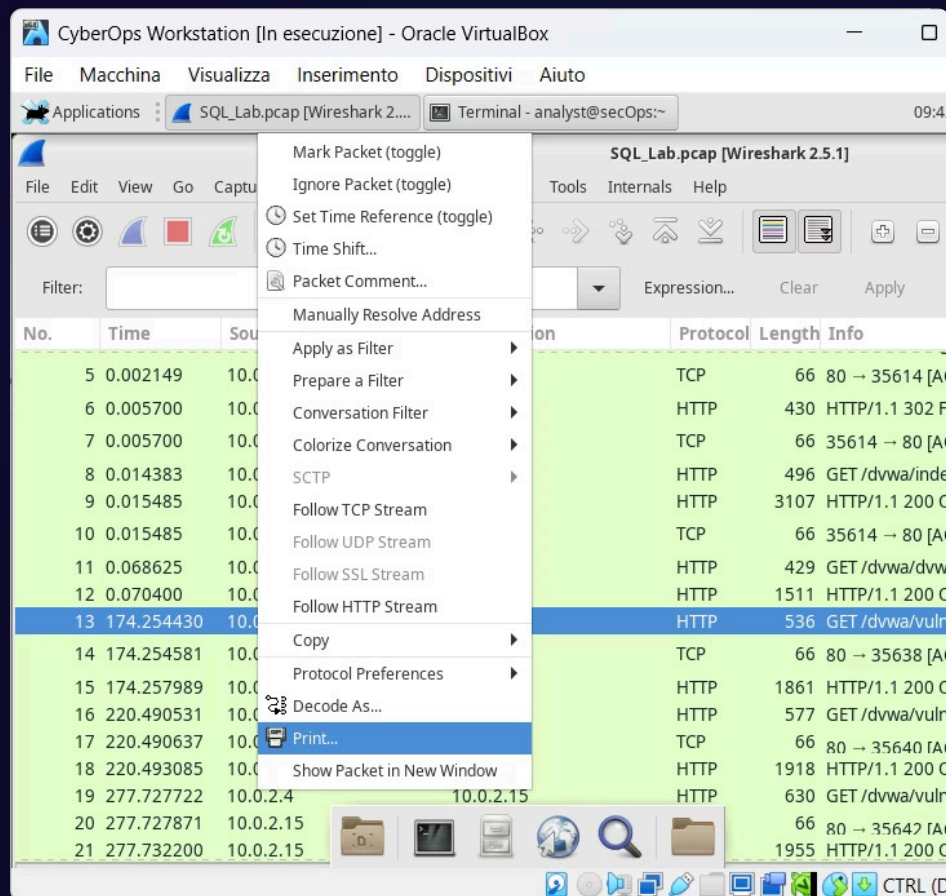
Il primo passo è stato aprire il file **SQL_Lab.pcap** con Wireshark, situato nella directory **/home/analyst/lab.support.files/**. Successivamente, sono stati identificati gli indirizzi IP coinvolti nell'attacco:

- Attaccante: **10.0.2.4**
- Bersaglio: **10.0.2.15**

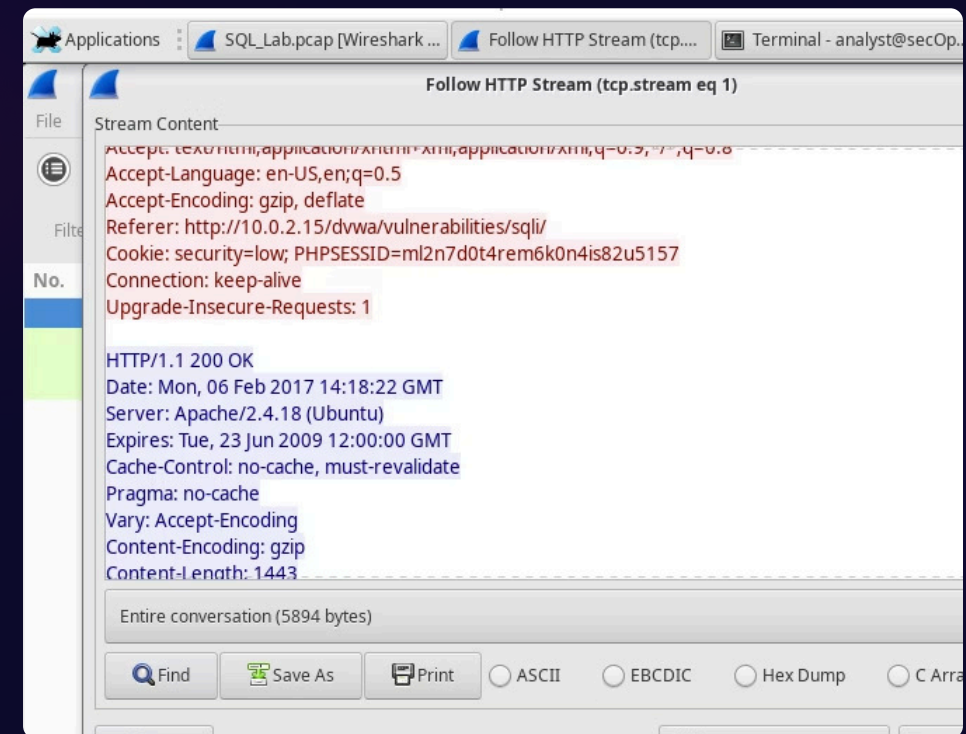
Questi indirizzi saranno fondamentali per filtrare e analizzare il traffico rilevante durante l'attacco SQL Injection. L'identificazione precoce degli IP aiuta a circoscrivere l'analisi e a focalizzarsi sulle comunicazioni sospette.

Test di Vulnerabilità e Inizio dell'Attacco

È stato seguito l'**HTTP Stream** a partire dalla riga 13 per intercettare la richiesta. La richiesta iniziale è stata: **GET /dvwa/vulnerabilities/sqli/** con il parametro **1=1**.



La risposta del server con dati validi ha confermato che il sito è vulnerabile a SQL Injection. Questo test preliminare è cruciale per valutare la sicurezza di un'applicazione web. Un risultato positivo indica la necessità di ulteriori analisi e interventi correttivi immediati.

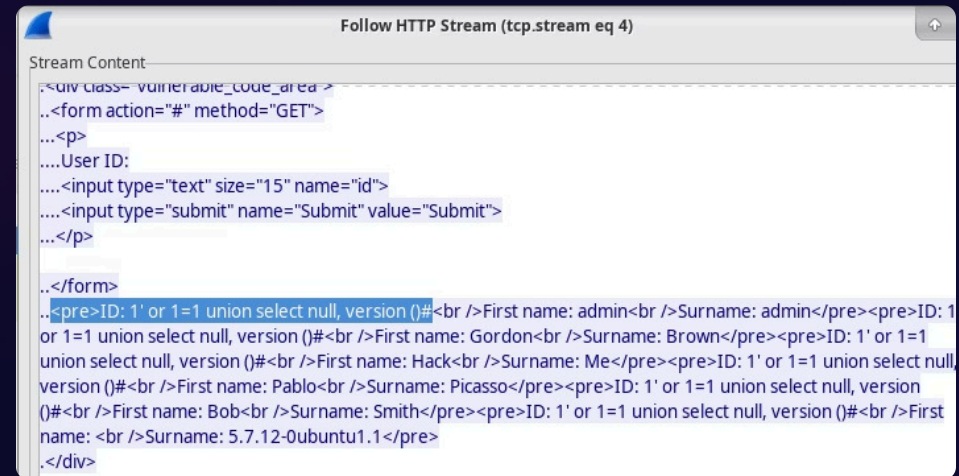


SQL Injection: Estrazione di Informazioni Critiche (database() e user())

Dalla riga 19, è emersa la query SQL Injection: **1' or 1=1 union select database(), user()#**.

Il risultato ottenuto è stato:

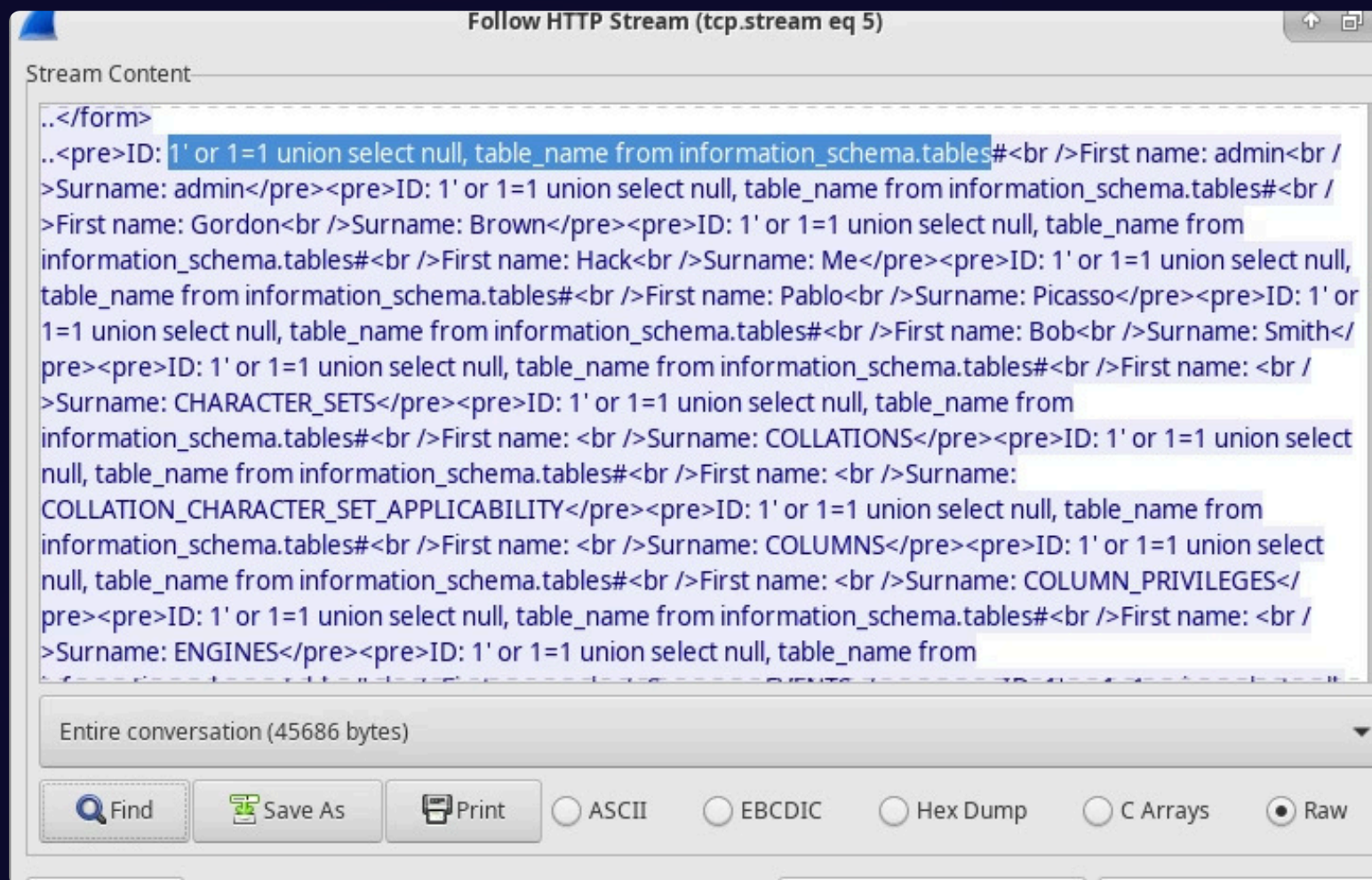
- Database: **dvwa**
- Utente SQL: **root@localhost**



```
Follow HTTP Stream (tcp.stream eq 4)
Stream Content
...<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>
```

Questa tecnica permette all'attaccante di ottenere informazioni cruciali sul database e sull'utente SQL, aprendo la strada a ulteriori attacchi e all'accesso a dati sensibili.

SQL Injection per la Versione del Database e Enumerazione delle Tabelle



Versione del Database

L'attaccante ha utilizzato la query: **1' or 1=1 union select null, version()#**.

Risultato della versione MySQL: **5.7.12-0ubuntu1.1**.

Enumerazione Tabelle

Con la query: **1' or 1=1 union select null, table_name from information_schema.tables#** l'attaccante ottiene la lista delle tabelle disponibili nel database (es. users, collations, columns, engines, ecc.).

La conoscenza della versione del database e della struttura delle tabelle consente all'attaccante di personalizzare ulteriormente gli attacchi, sfruttando vulnerabilità specifiche e accedendo a dati sensibili in modo mirato.

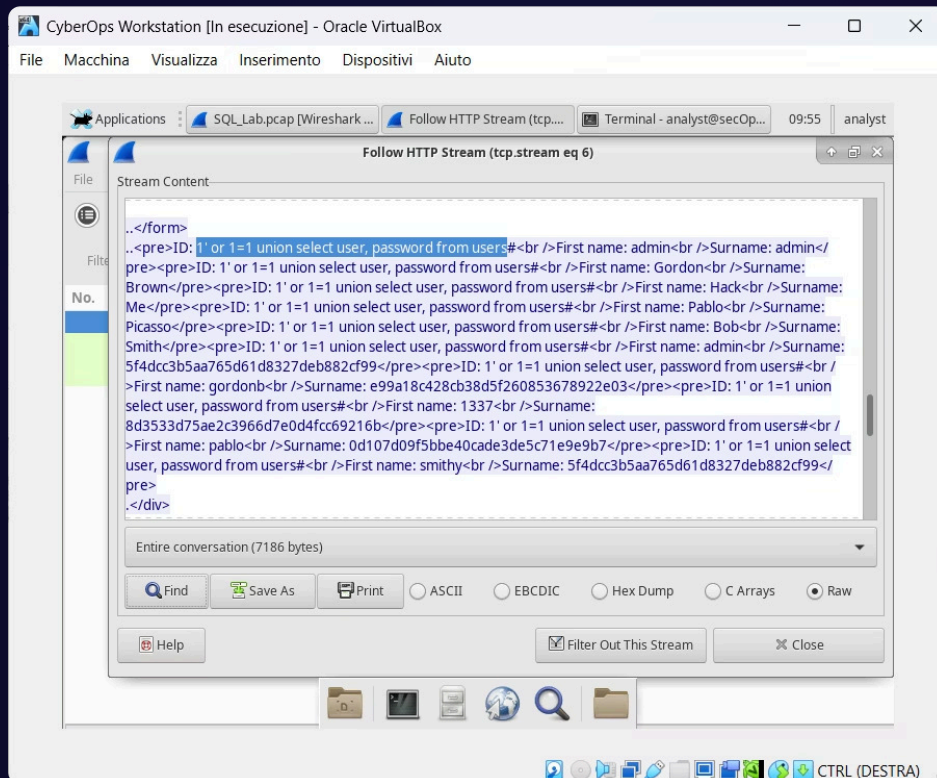
Estrazione di Utenti e Hash delle Password

La query usata per estrarre gli utenti e le password è stata: **1' or 1=1 union select user, password from users#**.

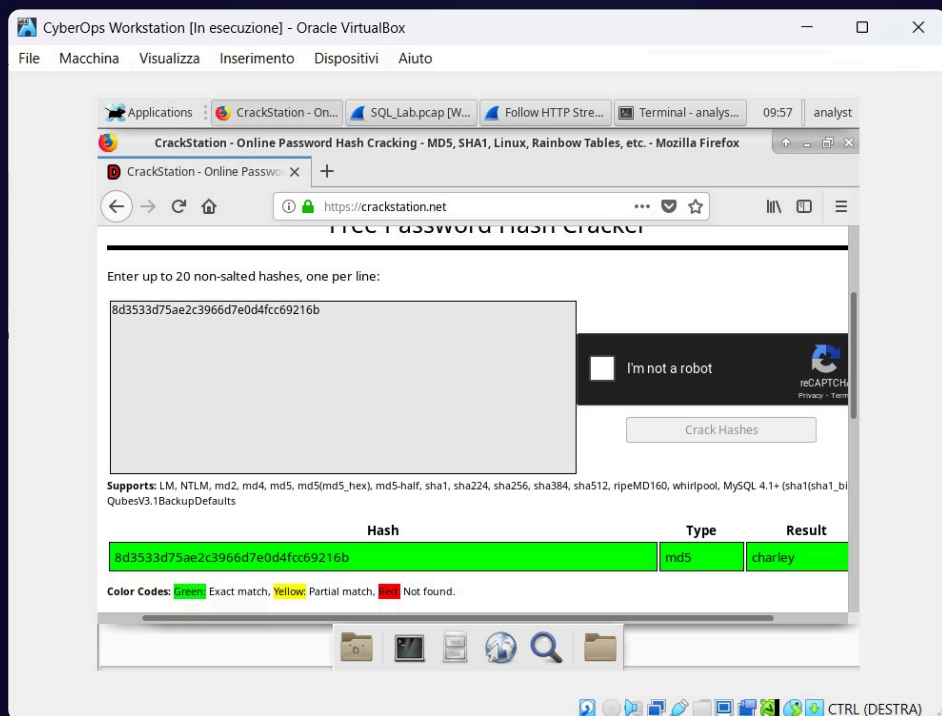
Il database ha risposto con una lista di utenti e hash delle loro password. Esempio:

- User: **1337**
- Hash: **8d3533d75ae2c3966d7e0d4fcc69216b**

Questo è un passaggio critico in un attacco SQL Injection, poiché l'accesso agli hash delle password permette all'attaccante di tentare il cracking delle credenziali e ottenere l'accesso non autorizzato al sistema.



Cracking dell'Hash e Recupero della Password in Chiaro



L'hash **8d3533d75ae2c3966d7e0d4fcc69216b** è stato decifrato con successo tramite **CrackStation**. La password in chiaro recuperata è: **charley**.

Questo dimostra come un semplice hash MD5 possa essere facilmente compromesso con strumenti gratuiti disponibili online. La facilità con cui è stato possibile recuperare la password sottolinea l'importanza di utilizzare algoritmi di hashing più robusti e di implementare misure di sicurezza aggiuntive, come il **salting**, per proteggere le credenziali degli utenti.

Conclusioni e Metodi di Prevenzione degli Attacchi SQL Injection

Questo laboratorio ha evidenziato la pericolosità degli attacchi SQL Injection e la facilità con cui un sito vulnerabile può rivelare dati sensibili, la struttura del database e le credenziali degli utenti. È fondamentale sanificare l'input e utilizzare query parametrizzate per prevenire tali attacchi.

Due metodi efficaci per prevenire attacchi SQL Injection sono:

1. Utilizzo di **prepared statements** (query parametriche): In questo modo, i dati forniti dall'utente vengono trattati come parametri e non come parte del codice SQL, prevenendo l'iniezione di codice malevolo.
2. Implementazione di un **WAF (Web Application Firewall)**: Un WAF funge da filtro tra l'applicazione web e l'utente, analizzando il traffico HTTP e bloccando input potenzialmente pericolosi.