Report sull'Attacco SQL Injection

Michele Storelli

June 13, 2025

Introduzione

Questo report analizza un attacco di SQL injection catturato in un file PCAP. L'analisi è stata condotta utilizzando Wireshark, un analizzatore di pacchetti di rete, per visualizzare il traffico di rete e comprendere le fasi dell'attacco. Gli attacchi SQL injection rappresentano una seria minaccia per le applicazioni web basate su database, consentendo agli aggressori di manipolare i dati, compromettere le identità e ottenere accesso non autorizzato.

Analisi dell'Attacco e Risposte alle Domande

1. Indirizzi IP Coinvolti nell'Attacco

Quali sono i due indirizzi IP coinvolti in questo attacco di SQL injection in base alle informazioni visualizzate?

Basandosi sulle informazioni visualizzate nel file PCAP, i due indirizzi IP coinvolti nell'attacco di SQL injection sono:

- 10.0.2.4 (Sorgente dell'attacco)
- 10.0.2.15 (Vittima dell'attacco)

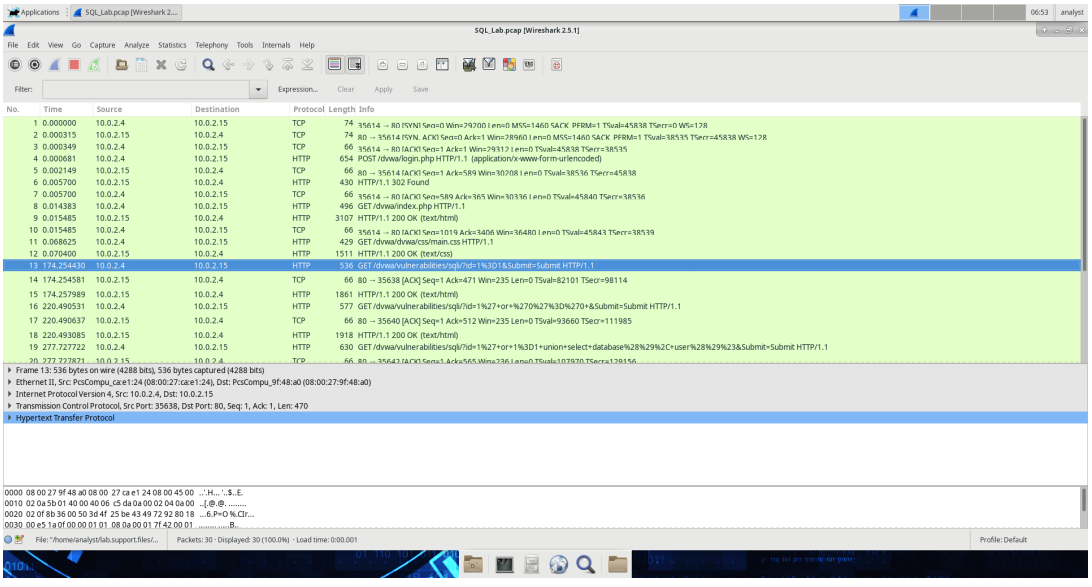

Figure 1: Panoramica del traffico di rete in Wireshark, evidenziando gli IP coinvolti.

2. Visualizzazione dell'Attacco di SQL Injection (Fase Iniziale)

L'aggressore ha iniziato l'attacco testando la vulnerabilità dell'applicazione?

L'aggressore ha iniziato l'attacco testando la vulnerabilità dell'applicazione. Come mostrato nella Figura 2, l'aggressore ha inserito la query `1=1` in un campo UserID. La risposta dell'applicazione, che

invece di un fallimento di login ha restituito un record dal database, ha confermato la vulnerabilità alla SQL injection.

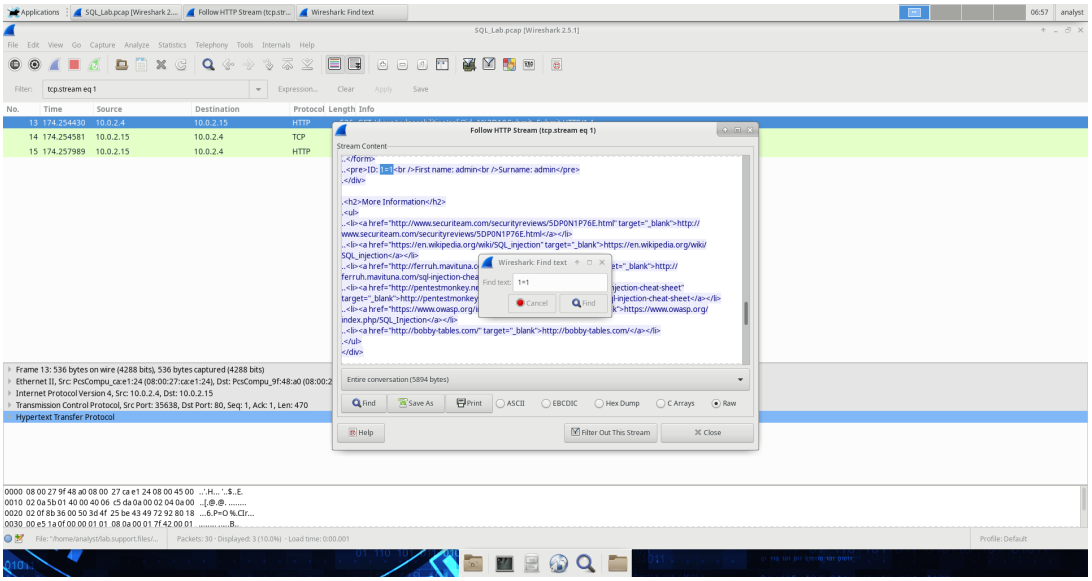

Figure 2: Test iniziale di SQL injection: tentativo con la query '1=1' e risposta dell'applicazione.

3. Continuazione dell'Attacco di SQL Injection (Informazioni sul Database e Utenti)

L'aggressore ha proseguito nella continuazione dell'attacco?

Successivamente, l'aggressore ha affinato la query per ottenere informazioni più specifiche. Come si vede nella Figura 3, è stata utilizzata la query `1' or 1=1 union select database(), user()#`. La risposta dell'applicazione ha rivelato le seguenti informazioni:

- Nome del database: **dvwa**
- Utente del database: **root@localhost**

Vengono visualizzati anche account utente aggiuntivi, indicando una riuscita enumerazione.

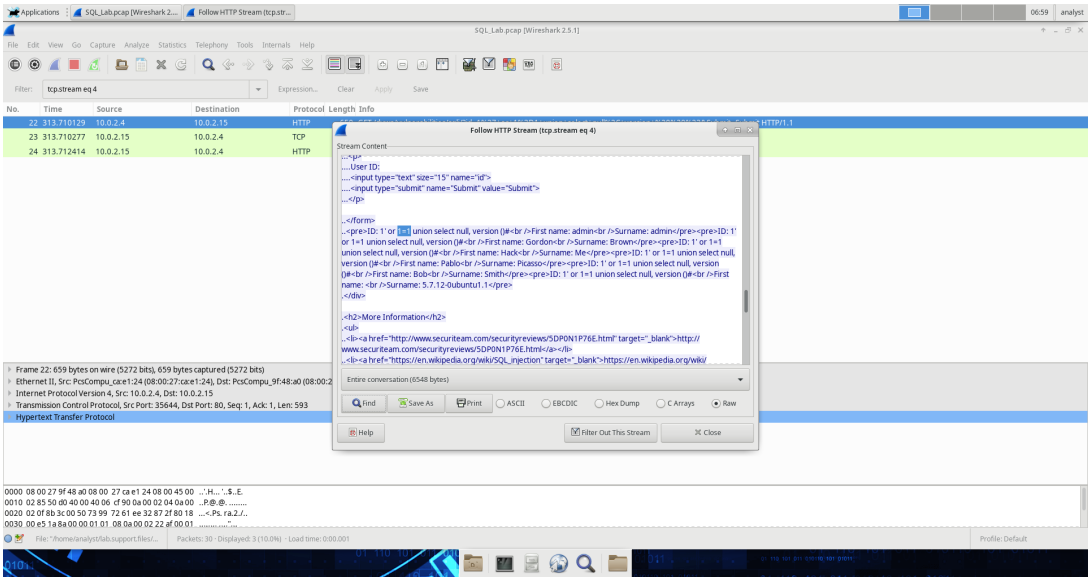

Figure 3: Estrazione di informazioni sul database (nome e utente) tramite SQL injection.

4. Attacco di SQL Injection e Informazioni di Sistema

Qual è la versione?

L'aggressore ha proseguito nell'estrazione di informazioni di sistema, in particolare la versione del database. Dallo screenshot precedentemente associato alla query `1' or 1=1 union select null, version ()#`, si è potuto determinare che:

- La versione del database è: **5.5.43-0ubuntu0.14.04.1**

Questa informazione è cruciale per l'aggressore per pianificare ulteriori exploit specifici per quella versione.

5. Attacco di SQL Injection e Informazioni sulle Tabelle

Cosa farebbe per l'aggressore il comando modificato di `1' OR 1=1 UNION SELECT null, column_name FROM INFORMATION_SCHEMA.columns WHERE table_name='users'`?

L'aggressore ha poi tentato di enumerare le tabelle presenti nel database. Sebbene non mostrato in uno screenshot dedicato per la sola query delle tabelle, la successiva fase dell'attacco (Figura 4) implica la conoscenza delle tabelle. La query `1'or 1=1 union select null, table_name from information_schema.tables#` avrebbe prodotto un output esteso con tutti i nomi delle tabelle.

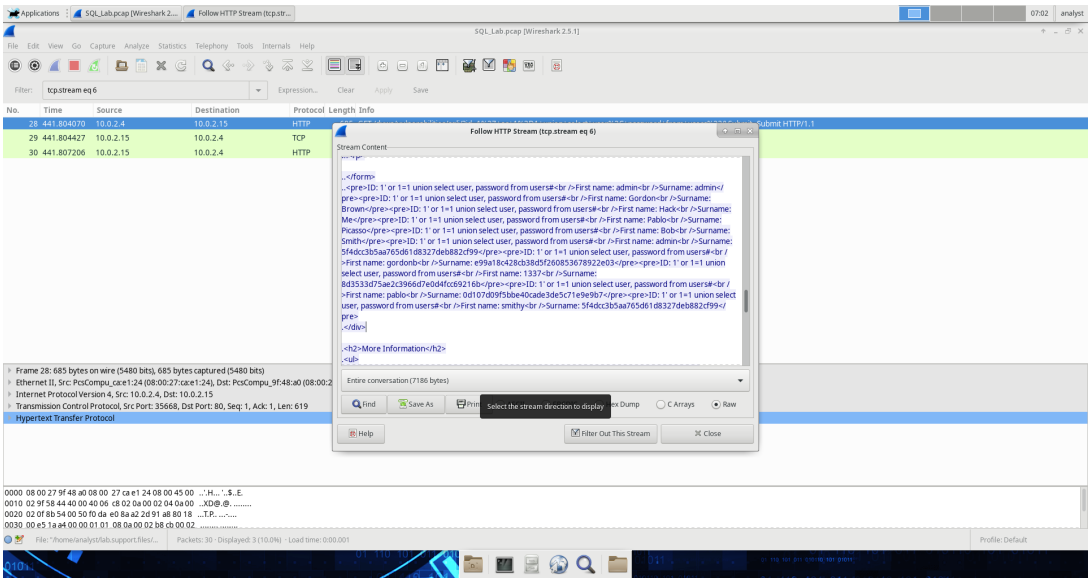

Figure 4: Query per l'estrazione dei nomi utente e degli hash delle password.

Il comando modificato `1' OR 1=1 UNION SELECT null, column_name FROM INFORMATION_SCHEMA.columns WHERE table_name='users'` avrebbe come effetto per l'aggressore di **restituire un output molto più breve e mirato, mostrando solo i nomi delle colonne della tabella denominata 'users'**. Questo ridurrebbe significativamente la quantità di dati da analizzare, focalizzandosi solo sulle colonne pertinenti.

6. Conclusione dell'Attacco di SQL Injection (Hash delle Password)

Quale utente ha l'hash della password di 8d3533d75ae2c3966d7e0d4fcc69216b? Qual è la password in chiaro?

La fase finale dell'attacco ha mirato all'estrazione degli hash delle password. Come mostrato nella Figura 4, l'aggressore ha eseguito la query `1'or 1=1 union select user, password from users#`. Questo ha portato all'ottenimento di nomi utente e dei relativi hash delle password.

- L'utente con l'hash della password **8d3533d75ae2c3966d7e0d4fcc69216b** è **charley**, come confermato dal crack dell'hash su CrackStation (Figura 5).
- La password in chiaro è: **charley**.

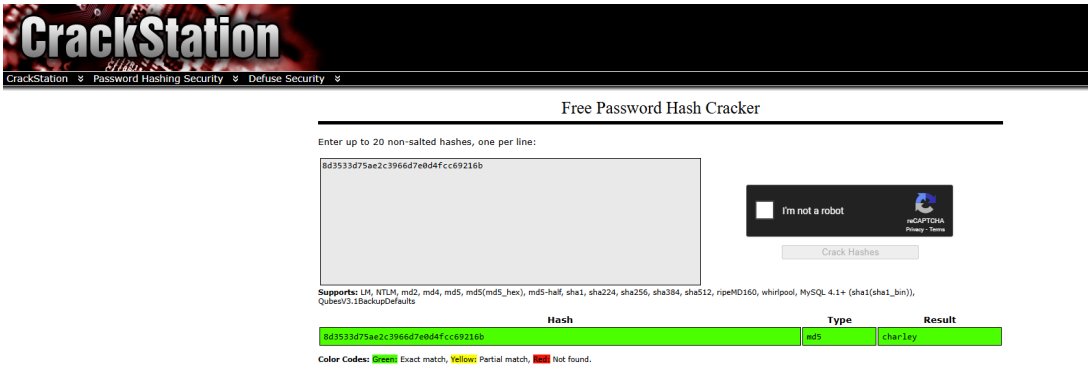

Figure 5: Risultato del cracking dell'hash della password '8d3533d75ae2c3966d7e0d4fcc69216b' su CrackStation.

Domande di Riflessione

1. Qual è il rischio che le piattaforme utilizzino il linguaggio SQL?

Il rischio principale è che le piattaforme che utilizzano SQL siano vulnerabili agli attacchi di SQL injection. Questi attacchi possono consentire a un aggressore di:

- Accedere, modificare o eliminare dati sensibili nel database.
- Bypasare i meccanismi di autenticazione e autorizzazione.
- Ottenere informazioni riservate sul database e sul sistema operativo sottostante.
- Eseguire comandi arbitrari sul server del database (in alcuni casi).

La gravità del rischio dipende dalla configurazione del database, dai privilegi dell'utente del database e dal tipo di informazioni memorizzate.

2. Naviga in internet ed esegui una ricerca per "prevenire attacchi di SQL injection". Quali sono 2 metodi o passaggi che possono essere adottati per prevenire gli attacchi di SQL injection?

Due metodi efficaci per prevenire gli attacchi di SQL injection sono:

- **Utilizzo di Prepared Statements (o istruzioni preparate) con parametri:** Questo è il metodo più raccomandato. Invece di concatenare direttamente l'input dell'utente nella query SQL, le istruzioni preparate separano la logica della query dai dati. I valori dei parametri vengono inviati al database separatamente, impedendo che l'input malevolo venga interpretato come parte della logica SQL.
- **Validazione e Sanitizzazione dell'Input dell'Utente:** Tutti gli input ricevuti dall'utente devono essere attentamente validati e sanitizzati prima di essere utilizzati nelle query SQL. Ciò implica la rimozione o la neutralizzazione di caratteri speciali, parole chiave SQL o sequenze che potrebbero essere utilizzate per manipolare la query. Ad esempio, si possono utilizzare funzioni di escape specifiche del database per "sanitizzare" l'input.