# Progetto S11 – L5

## BONUS 2 – Attacco a un database MySQL

---

## Executive Summary

In questo laboratorio ho a**nalizzato un file PCAP contenente un attacco di SQL Injection contro un database MySQL utilizzando Wireshark**. Attraverso l'**analisi del traffico HTTP** ho osservato come l'aggressore abbia sfruttato una **vulnerabilità applicativa per estrarre informazioni sensibili, tra cui nome del database, versione del server e hash delle password**. L'analisi evidenzia l'impatto critico della mancata validazione dell'input utente.

---

## Introduzione

L'obiettivo dell'esercizio è **analizzare un attacco di SQL Injection** esaminando traffico di rete precedentemente catturato. Utilizzando Wireshark, ho studiato le **richieste HTTP e le risposte del server per comprendere come l'attaccante sia riuscito a manipolare le query SQL ed ottenere accesso a informazioni riservate del database**.

---

## Nota metodologica

In questo esercizio BONUS **non sono stati inseriti screenshot a scopo dimostrativo, poiché l'analisi è stata condotta su un file PCAP fornito dal laboratorio e le informazioni rilevanti sono esplicitamente contenute nel traffico HTTP analizzato**.

L'obiettivo dell'attività è comprendere e spiegare la **dinamica dell'attacco SQL Injection attraverso l'interpretazione tecnica del traffico di rete, non documentare un'operazione eseguita in tempo reale**.

Per questo motivo **è stata privilegiata una descrizione tecnica dettagliata e strutturata del comportamento dell'attaccante e delle risposte del database.**

---

# Parte 1 – Apertura del file PCAP

Avvio la macchina virtuale CyberOps Workstation e apro Wireshark.
Carico il file **SQL_Lab.pcap**, che contiene traffico di rete catturato per circa 441 secondi, relativo a un attacco SQL Injection contro un server web con backend MySQL.

### Domanda:

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

### Risposta:

Dall’analisi del traffico emergono due host principali:

- 10.0.2.4 → Attaccante
- 10.0.2.15 → Vittima (server web/database)

---

# Parte 2 – Inizio dell’attacco SQL Injection

Analizzando il traffico HTTP si osserva che l’aggressore inserisce nel campo UserID la stringa:
1=1

Questa condizione è sempre vera e viene utilizzata per verificare se l’applicazione è vulnerabile alla SQL Injection.

Il server risponde restituendo un record del database invece di negare l’accesso.
Questo comportamento conferma che l’applicazione è vulnerabile.

---

# Parte 3 – Estrazione delle informazioni sul database

Nel traffico successivo si osserva l’inserimento della query:
**1' OR 1=1 UNION SELECT database(), user()#**

Il server risponde fornendo:

- Nome del database: dvwa
- Utente del database: root@localhost
- Elenco di più account utente

L’aggressore riesce quindi a ottenere informazioni sensibili sulla struttura del database.

---

# Parte 4 – Identificazione della versione del database

Nel traffico HTTP viene individuata la query:
**1' OR 1=1 UNION SELECT null, version()#**

### Domanda:

Qual è la versione?

### Risposta:

La versione del database MySQL restituita è:

5.7.21-0ubuntu0.16.04.1

Conoscere la versione consente all’aggressore di ricercare eventuali vulnerabilità specifiche del DBMS.

---

# Parte 5 – Enumerazione delle tabelle

L’aggressore esegue la query:
**1' OR 1=1 UNION SELECT null, table_name FROM information_schema.tables#**

Questa istruzione consente di ottenere l’elenco delle tabelle presenti nel database.

### Domanda:

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')?**

### Risposta:

Il comando permetterebbe di visualizzare esclusivamente le colonne della tabella **users**, consentendo di identificare campi come username e password, facilitando ulteriori fasi dell’attacco.

---

# Parte 6 – Estrazione delle credenziali

Nella fase finale dell’attacco viene eseguita la query:
**1' OR 1=1 UNION SELECT user, password FROM users#**

Questa query estrae i nomi utente e gli hash delle password dal database.

### Domanda:

Quale utente ha l'hash della password di 8d3533d75ae2c3966d7e0d4fcc69216b?

### Risposta:

L’utente associato a quell’hash è:

admin

### Domanda:

Qual è la password in chiaro?

### Risposta:

Decifrando l’hash tramite un servizio di cracking, la password risulta essere:

qwerty

---

# Domande di Riflessione

### Domanda 1:

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

### Risposta:

Se l’input dell’utente non viene validato correttamente, un aggressore può:

- Manipolare le query SQL
- Accedere a dati riservati
- Modificare o cancellare informazioni
- Compromettere completamente il database

La gravità dell’attacco dipende dai privilegi associati all’utente del database.

---

### Domanda 2:

Quali sono 2 metodi o passaggi che possono essere adottati per prevenire gli attacchi di SQL injection?

### Risposta:

1. Utilizzo di query parametrizzate (prepared statements)
2. Validazione e sanitizzazione rigorosa dell’input utente

Ulteriori misure di sicurezza includono:

- Implementazione di un Web Application Firewall
- Applicazione del principio del minimo privilegio
- Monitoraggio e logging delle query SQL

---

# Conclusioni

L’analisi del traffico di rete ha dimostrato come una vulnerabilità SQL Injection possa consentire a un aggressore di ottenere informazioni critiche sul database, inclusi nomi di tabelle, versione del sistema e credenziali utente.

Questo laboratorio evidenzia l’importanza della sicurezza applicativa, dell’uso di query parametrizzate e della corretta validazione dell’input per prevenire compromissioni gravi dei sistemi informativi.