**RELAZIONE HOMEWORK – TRACCIA 2 (SQLi)**

Membri gruppo:

* Davide Cioeta, 2063098
* Gianmarco Cestari,
* Francesco Serva,

**Introduzione**

L'SQL Injection (SQLi) è una delle vulnerabilità più comuni e pericolose nelle applicazioni web che utilizzano database. Consiste nell'inserimento di codice SQL malevolo in un input, che viene poi eseguito direttamente dal server, permettendo all'attaccante di manipolare le query SQL eseguite dall'applicazione.

**Alcuni esempi:**

1. **Autenticazione, Bypass:**

‘ OR 1=1 --

Questo è uno degli esempi più semplici e diffusi. Se viene inserito in un campo di login (es. nome utente o password), viene restituito il primo record della tabella di accesso del database, permettendo l’accesso.

1. **Enumerazione del numero di colonne di una tabella:**

' ORDER BY N --

Serve per scoprire il numero di colonne della query SQL. Si prova con numeri crescenti (N = 1, 2, 3, ...) finché non si ottiene un errore. Quando viene restituito l’errore “Unkwown …” abbiamo trovato il numero di colonne, ovvero N-1.

1. **Verifica delle colonne visibili:**

' UNION SELECT 1, 2, 3, ..., N-1 –

In base alla risposta sarà possibile capire quali colonne della tabella di accesso sono "visibili a schermo", e quindi da quali di queste è possibile iniettare valori personalizzati ed estrarre dati; altrimenti si otterrà il seguente errore: "Errore nella query: The used SELECT statements have a different number of columns".

1. **Scoperta delle tabelle:**

' UNION SELECT 1, table name, 3, 4 FROM information\_schema.tables WHERE table schema = database() LIMIT 1 OFFSET 0 –

Questa query permette di elencare le tabelle presenti nel database. Per controllare la presenza di altre tabelle sarà necessario solo aumentare il numero relativo all'OFFSET; quando negherà l'accesso, vorrà dire che il numero di tabelle sarà uguale a OFFSET-1.

1. **Modificare il contenuto delle tabelle:**

'; UPDATE nomeTabella SET ruolo='amministratore' WHERE username=username; --

Una volta ottenuto il nome della tabella e i suoi campi visibili attraverso le query precedenti, è possibile eseguire una query che modifica il contenuto della tabella scelta. In questo caso si va a modificare il ruolo di un utente nell’applicazione per avere dei benefici migliori.

1. **Eliminare una tabella:**

‘; DROP TABLE nomeTabella; --

Se l’obiettivo dell’attacco fosse quello di eliminare il database, allora questo è possibile eseguendo questa query, inserendo il nome della tabella che si vuole eliminare, ottenuto con le query precedenti.

**Spiegazione degli attacchi e dei metodi:**

Ogni attacco inizia con l’uso dell’apice singolo, così da chiudere il campo “username” della query:

*"SELECT \* FROM utenti WHERE username='$username' AND password='$password';"*

Poi, la query può essere seguita da un’altra condizione, oppure viene chiusa con il punto e virgola “;” e seguita da una nuova istruzione SQL.

Il “--“, invece, commenta tutto il resto della query, annullando le condizioni successive e quindi, permette di bypassare l'autenticazione legata al campo della password; viene detto **commento di fine riga**.

Nel primo esempio, dopo aver chiuso il campo username, viene aggiunta l’istruzione “OR 1=1”, che rende la query sempre vera. Questo tipo di attacco prende il nome di **Tautologia**.

Infine, la pratica di aggiungere ulteriori query SQL rispetto a quella prevista, veicolando l’attacco a dorso di una richiesta legittima, è detto **query Piggybacked**.