Relazione Sicurezza

Simone Turco

June 6, 2024

Traccia 3

Realizzare un attacco di SQL injection di tipo inband (ovvero stesso canale utilizzato per l'injection della query malevola e per ricevere i risultati), basato su input dell'utente (ovvero l'attaccante inietta i comandi SQL fornendo un input opportunamente costruito) e che utilizzi una o più delle seguenti modalità: Tautologia, commento di fine riga; query piggybacked. Mediante l'injection di opportuni comandi mostrare che e' sia possibile compromettere almeno due delle proprietà CIA. Suggerimenti:

- *Supponete che il sistema da attaccare sia un server su cui e' installato un DBMS, su cui risiede il database e su cui risiede l'applicazione web vulnerabile (ad esempio una pagina Php che genera query al DB)*
- *Il server può' essere emulato mediante una macchina virtuale o un docker container*
- *Scegliere versioni del software (OS, DB, Php, etc. . .) che siano vulnerabili ad attacchi SQLi*

Introduzione

La traccia scelta richiede quindi di eseguire un attacco di tipo SQL injection su un sito che offre un scarsa protezione sui dati sfruttando la natura delle pagine web che interagiscono in modo diretto sul database.

1 Preparazione dell'ambiente

Per la realizzazione del server è stato utilizzato docker per creare i contenitori. Il file definisce le impostazioni necessarie per avviare i due servizi, mappare le porte appropriate e garantire che i dati del database siano persistenti. Il docker-compose file definisce quindi un ambiente di container docker con due servizi:

- **Servizio Web**. Per il servizio web è stata utilizzata un'immagine per un'applicazione php (8.2) che implementa apache. Sono quindi state installate tutte le dipendenze in modo da poter comunicare correttamente con il database, impostato la directory di lavoro e copiato l'applicazione php nel container.
- **Servizio database**. Per il servizio database è stata utilizzata un'immagine MySQL nell'ultima versione. Sono stati montati i volumi per l'inizializzazione del database, definite le variabili d'ambiente per per la configurazione, specificando il nome, nome utente e password, ed infine esposta la porta '3306' per la comunicazione con il servizio web.

Per l'esecuzione del container, lanciare da terminale il seguente comando:

```
docker-compose up -build
```

2 MySQL

Nel database sono quindi state inserite e popolate le tabelle specificate nel file nel file 'init.sql', ovvero:

- studente (matricola, nome, cognome, pass)
- indirizzo (studente, citta, via)
- corso (nome, cfu)
- esame (studente, corso, valutazione)

Al momento della creazione del database, non è stata definita alcuna politica di accesso, inclusi permessi di accesso e modifica alle tabelle, in modo da azzerare la solidità del sistema, permettendo ad ogni utente di avere accesso alle tabelle senza distinzione. Non è presente inoltre un protezione delle password di alcun tipo, escludendo metodi come la crittografia.
Questa politica porta ad potenziale un effetto catastrofico sui dati, in quanto un utente malintenzionato potrebbe, senza troppa difficoltà compromettere la disponibilità, la confidenzialità e l'integrità dei dati.

3 Php e html

Il web server utilizza php, per comunicare con il database, inviando richieste tramite funzioni implementate nella libreria mysqli, che consente la comunicazione delle parti.
Il sito HTML, a cui è incorporato php, prevede una pagina di login, in cui l'utente (in questo caso lo studente) inserisce matricola e password, per accedere al sistema. All'intenrno della pagina viene quindi in primo luogo stabilita un connessione con il database, tramite le credenziali fornite in precedenza, e una volta che l'utente inserisce le credenziali di accesso, viene controllato che il valore inserito non sia nullo e in seguito inviata una query al DB:

```
SELECT matricola, nome, cognome FROM studente WHERE matricola = "$matricola" and pass
= "$password"
```

Se la query restituisce almeno un riga, allora l'utente è stato trovato e l'accesso è garantito.
Una volta effettuato l'accesso, l'utente può eseguire diverse operazioni come:

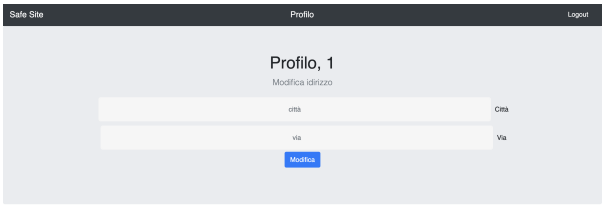
- visualizzare i corsi di studio. Per questa operazione è presente un form che prende in input una stringa, corrispondente al corso che si vuole cercare. Una volta inserita, viene inviata una richiesta a mysql, per ottenere tutti i corsi il cui nome contiene la stringa, e verrà quindi generata in modo dinamico una tabella contenente nome e cfu dei corsi.
- cambiare indirizzo, nella sezione profilo. Anche qui è presente un form dove inserire citta e via.
- effettuare il logout, ritornando alla pagina di index ed eliminando i dati della sessione.

I metodi utilizzati per le query in mySQl da php sono:

- **mysqli_query**, per un eseguire una normale query
- **mysqli_multi_query**, per eseguire più query concatenate

Questo tipo di metodi sono particolarmente vulnerabili e si prestano ad un attacco di tipo SQL injection, in quanto non viene adottata nessuna misura di sicurezza come codifica difensiva, rilevamento o prevenzione.

(a) home



(b) option

4 SQL Injection

L'attacco di injection SQL condotto sfrutta le vulnerabilità di sicurezza descritte in precedenza in modo da compromettere le proprietà CIA. La categoria dell'attacco appartiene a quella degli **inband**, dove l'aggressore sfrutta le vulnerabilità di un'applicazione web per eseguire comandi SQL non autorizzati direttamente sul database dell'applicazione stessa.

Il primo passo e quindi quello di **identificare le vunlerabilità** all'interno dell'applicazione Web, ovvero un punto di ingresso come ad esempio un campo di input che permette di iniettare il codice SQL malevolo. L'identificazione delle vulnerabilità può essere eseguita anche attraverso metodi alternativi, ovvero **attacchi inferenziali**, formulando query illegali, per capire la struttura e il tipo di database, oppure tramite injection cieco, per tentare di dedutrre i dati presenti nel sistema.

La pagina di login si presta in modo ottimale all'injection poiché non viene effettuato un controllo di alcun tipo sull'input inserito. In questo specifico caso è possibile ottenere l'accesso al sistema in due modi:

- **end-of-line**

  username: 1'; #
  password: x

  L'attaccante riesce ad effettuare un acccesso non autorizzato, sfruttando il fatto che la query viene terminata in modo anticipato, commentando il codice della query successivo dove avviene l'effettivo controllo della password.

- **tautologia e end-of-line**

  username: 1
  password: x or 1=1; #

  Tautologia ed end-of-line vengono utilizzate in combinazione facendo in modo che la query venga valutata sempre vera, in quanto viene aggiunta la condizione 'or 1=1'. Poi la query viene terminata in modo anticipato inserendo un commento.

La proprietà della CIA maggiormente lesa in questo caso è quella della **confidenzialità** dei dati, che riguarda la protezione delle informazioni dall'accesso non autorizzato, assicurando che solo le persone autorizzate abbiano accesso ai dati sensibili.
L'attaccante in questo caso ha ottenuto l'accesso al sistema, nel secondo caso anche con l'identità di un utente al sistema, potendo quindi visualizzare informazioni che dovrebbero essere riservate.

Una volta ottenuto l'accesso, autorizzato o non autorizzato, al sistema ci si trova sulla pagina home,

dove è possibile effettuare una ricerca sui corsi, in base ad una parola chiave inserita. Verrà quindi creata in modo dinamico una tabella che restituirà il nome dei corsi, che contengono la parola chiave, insieme al numero di CFU assegnati al corso. Anche in questo caso, il mancato controllo sulla stringa in input, potrebbe portare a risultati indesiderati, permettendo ad un utente malintenzionato, di sfruttare la creazione della tabella per ottenere dati il cui normale accesso è riservato.
Il campo input può essere sfruttato da un attacco di tipo **union**, utilizzato in primo luogo per ottenere il nome di tutte le tabelle del database,

```
x' UNION SELECT table_name, null FROM information_schema.tables #
```

e in seguito per ottenere tutti gli attributi di una singola tabella, tramite il comando inniettato:

```
' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name
= 'studente' #
```

Una volta iniettati questi comandi è possibile venire a conoscenza di tutta la struttura del database, comprese tabelle e attributi. Può quindi essere sfruttata la tabella prodotta in modo dinamico da PHP per visualizzare la matricola e la password degli utenti, tramite il seguente codice:

```
lgo' UNION SELECT matricola, pass FROM studente #
```

-
Anche questo tipo di attacco compromette la **confidenzialità** dei dati, in quanto un utente malintenzionato riesce ad avere un accesso non autorizzato alle credenziali per il sistema e a tutte le informazioni delle tabelle del database.

Tramite queste informazioni, un attaccante potrebbe svoglere qualsiasi operazione nel database, come ad esempio eliminare i dati di una tabella o eliminare una tabella intera.
Per effetture un operazione di questo tipo si può sfruttare un attacco di tipo **piggybacked**, dove l'attaccante aggiunge uteriori query oltre quella intenzionale. Per il suo successo si ha quindi la necessità di eseguire una query multipla.
Dopo una breve analisi, inserendo input mal formattati e visualizzando gli errori restituiti, si nota facilmente che la funzione utilizzata per eseguire le query sql, negli input form visti in precedenza, è **mysqli_query**, che non blocca l'invio di query multiple al DB.
Tuttavia può essere sfruttata la funzione **mysqli_multi_query**, utilizzata nella sezione profilo, per modificare l'indirizzo di un utente, poiché permette l'invio di query multiple. E' possibile quindi inniettare il codice per effettuare il drop di una tabella,

```
via'); drop table esame; #
```

oppure eliminare tutti i dati di una tabella, ad esempio quella degli utenti, negando di conseguenza

l'accesso a questi ultimi, tramite il codice seguente, dove il primo comando disbilita il constrollo delle chiavi esterne, mentre il secondo elimina tutte le righe della tabella studente.

```
via'); SET FOREIGN_KEY_CHECKS=0; delete from studente; #
```

-

Tramite questo tipo di attacco vengono violate due delle proprietà CIA:

- **Integrità**, in particolare l'integrità dei dati, a causa della compromissione e della rimozione non autorizzata di dati e tabelle.
- **Disponibilità**, in quanto avendo eliminato le credenziali degli utenti, qualsiasi accesso leggittimo verrà negato, e ed interrotto finchè non avverrà un eventuale recupero del sistema

5 Prevenzione e recupero

Per evitare la compromissione del sistema, dovuta ad attacchi SQL injection di tipo inband, è necessario prendere contromisure che possono riguardare diversi aspetti come ad esempio la codifica difensiva, che comprende pratiche manuali o la parametrizzazione dell'input, il rilevamento e la prevenzione run-time. La libreria mysql per PHP offre diversi metodi per il controllo dell'input, in modo tale che non siano effettuate richieste indesiderate al database.
Un'altra linea di difesa efficace include il controllo degli accessi al database, in modo da definire i ruoli e i permessi legati a questi ultimi, specificando le particolari azioni che gli utenti del sistema possono eseguire. E' inoltre consigliato di mantenere un backup dei dati del database, protetto in modo sicuro, per permettere il ripristino immediato del sistema una volta che l'attacco è avvenuto.