Data: 06 Dicembre 2024

Alessandro Pietro Salerno

Network Traffic Analysis: Investigazione di un Attacco SQL Injection

Strumenti Utilizzati: Wireshark, Linux Shell, John the Ripper

Introduzione e Obiettivi

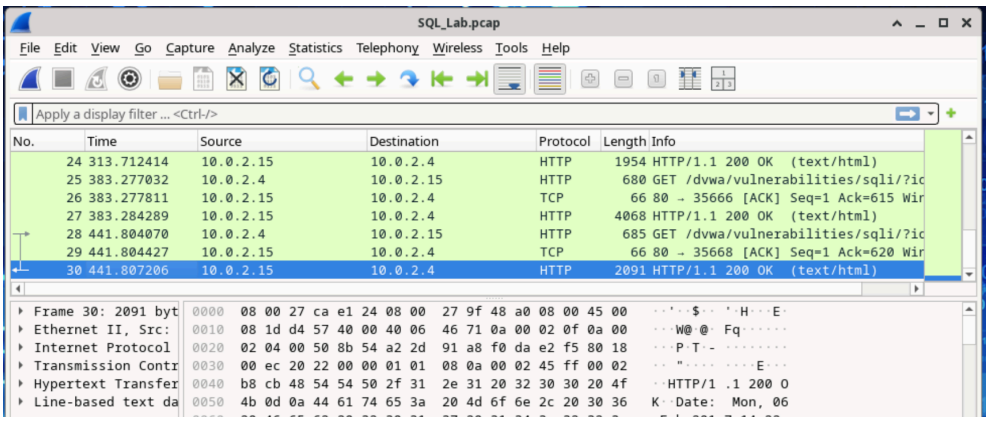
In questo progetto, ho condotto un'analisi forense di rete su un file di cattura pacchetti (PCAP) relativo a un incidente di sicurezza su un'applicazione web. L'obiettivo dell'analisi era ricostruire la "Kill Chain" dell'attacco, identificare gli attori coinvolti, determinare la natura della vulnerabilità sfruttata e quantificare l'esfiltrazione dei dati.

Questa attività simula uno scenario operativo reale all'interno di un **Security Operations Center** (SOC), dimostrando la mia capacità di analizzare il traffico di rete grezzo per rilevare minacce, comprendere le tecniche di attacco web e proporre strategie di mitigazione efficaci.

Analisi dell'Infrastruttura e Identificazione degli Attori

Attraverso l'analisi preliminare dei flussi TCP e HTTP in Wireshark, ho isolato il traffico sospetto e identificato gli host coinvolti nell'incidente:

- **Attacker (Source IP):** 10.0.2.4 - Questo IP è stato identificato come l'origine delle richieste HTTP malevole.
- **Victim (Destination IP):** 10.0.2.15 - Il server web che ospita l'applicazione vulnerabile (DVWA - Damn Vulnerable Web App).
- **Protocollo:** Il traffico analizzato è interamente HTTP sulla porta 80 .
- **Durata**: come si può notare la durata dell’ attacco SQL injection è durata 8min “441 secondi”.



Analisi Tecnica dell'Attacco (Kill Chain)

L'analisi approfondita dei payload HTTP ha rivelato un classico attacco di **SQL Injection (SQLi)** di tipo *UNION-based*. L'attaccante ha manipolato il parametro id nell'URL per interagire direttamente con il database backend.

Fase 1: Verifica della Vulnerabilità

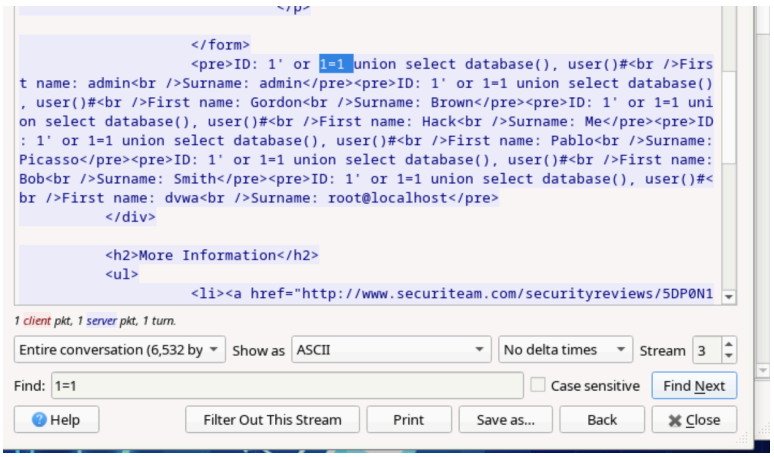
L'attaccante ha inizialmente testato la resilienza dell'applicazione iniettando una condizione logica booleana sempre vera.

- **Payload:** 1' or 1=1 #
- **Analisi:** L'istruzione OR 1=1 forza il database a restituire tutti i record, bypassando eventuali filtri WHERE.
- **Risultato:** Il server ha risposto con un codice 200 OK restituendo una lista completa di utenti, confermando la presenza della vulnerabilità.

Fase 2: Enumerazione del Sistema (Fingerprinting)

Una volta ottenuto l'accesso, l'attaccante ha eseguito una ricognizione per identificare la versione del database, fondamentale per scegliere gli exploit successivi.

- **Payload:** 1' or 1=1 union select null, version() #



- **Evidenza:** Analizzando il corpo della risposta HTTP (packet stream), ho estratto la versione del database esposta nell'output HTML.
- **Versione Rilevata:** MySQL 5.7.12-0ubuntu1.1.

Fase 3: Mappatura del Database

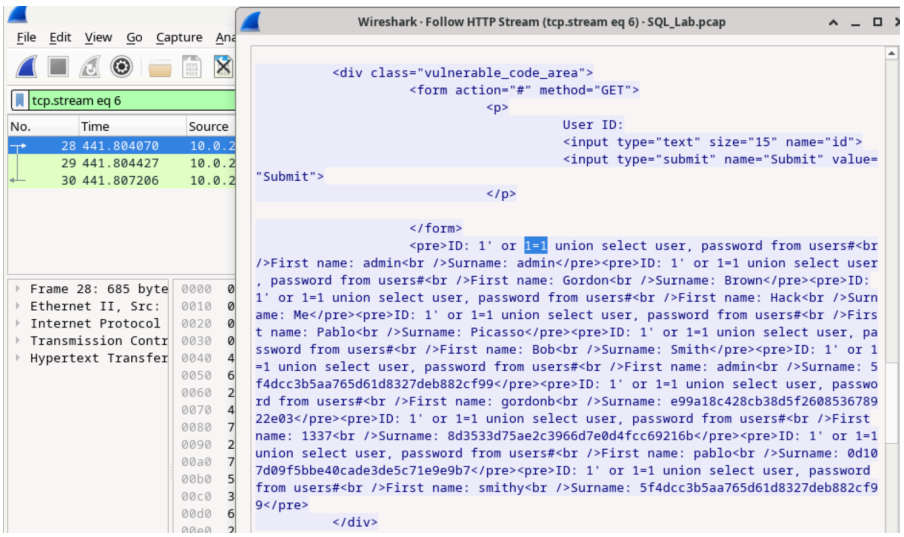
L'attaccante ha interrogato lo schema information_schema per elencare le tabelle presenti.

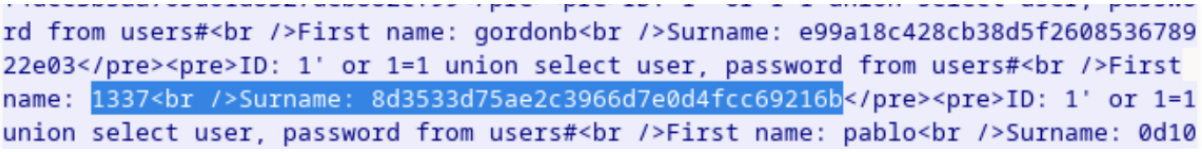
- **Payload:** 1' or 1=1 union select null, table_name from information_schema.tables #.
- **Analisi:** Sebbene il comando abbia generato molto "rumore" (elencando tutte le tabelle di sistema), l'attaccante ha identificato con successo la tabella di interesse: users.

Fase 4: Esfiltrazione dei Dati (Data Breach)

La fase finale dell'attacco ha mirato al furto di credenziali sensibili.

- **Payload:** 1' or 1=1 union select user, password from users #.
- **Dati Esposti:** La risposta del server conteneva una lista di username e hash delle password.





- **Record Compromesso:** Ho isolato un record specifico per l'analisi:
  - **User:** 1337
  - **Password Hash:** 8d3533d75ae2c3966d7e0d4fcc69216b.

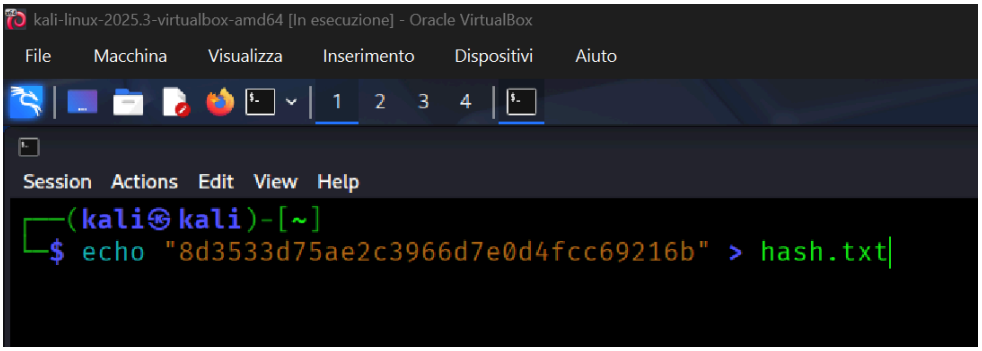
Post-Exploitation: Cracking della Password

Per valutare la gravità della compromissione, ho proceduto al recupero della password in chiaro partendo dall'hash esfiltrato. Sebbene sia possibile utilizzare servizi online, ho optato per un approccio locale utilizzando **John the Ripper** su ambiente Linux per garantire una migliore OPSEC (Operational Security) ed usufruire di ulteriori vantaggi.

Procedura:

1. Identificazione dell'algoritmo di hashing: La stringa di 32 caratteri esadecimali è stata identificata come **MD5**.
2. Esecuzione dell'attacco a dizionario (Wordlist attack) utilizzando la wordlist rockyou.txt.

Dopo avere opportunamente creato il file hash.txt da dare a john the Ripper



Comando Eseguito:

```
(kali㉿kali)-[~]
$ echo "8d3533d75ae2c3966d7e0d4fcc69216b" > hash.txt

(kali㉿kali)-[~]
$ john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
```

Risultato: L'hash è stato decifrato con successo.

```
(kali㉿kali)-[~]
$ john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
charley          (?)
1g 0:00:00:00 DONE (2025-12-06 12:30) 50.00g/s 153600p/s 153600c/s 153600C/s my3kids..dangerous
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

- **Password in chiaro:** charley.

Strategie di Mitigazione e Remediation

Sulla base dell'analisi effettuata, raccomando le seguenti azioni correttive per mettere in sicurezza l'applicazione:

1. **Prepared Statements (Query Parametrizzate):** Questa è la difesa primaria. L'uso di parametri vincolati assicura che il database tratti l'input dell'utente (come 1' or 1=1) esclusivamente come dati e mai come codice eseguibile.
2. **Input Sanitization:** Implementare whitelist rigorose per l'input, rifiutando caratteri non necessari come apici singoli o commenti SQL (--, #).

3. **WAF (Web Application Firewall):** Implementare regole per bloccare pattern SQL comuni (es. UNION SELECT) a livello di rete.
4. **Principio del Minimo Privilegio:** L'applicazione web dovrebbe connettersi al database con un utente che ha permessi limitati, non come root (come invece osservato in questo incidente, dove l'utente del DB era root@localhost).

Conclusioni

Questa analisi ha dimostrato come un input non sanitizzato possa portare alla completa compromissione di un database in meno di 8 minuti. Attraverso l'uso di Wireshark, sono stato in grado di ricostruire ogni passaggio dell'attaccante, dalla ricognizione iniziale all'esfiltrazione dei dati.

L'esercizio evidenzia la mia competenza tecnica nell'analisi dei pacchetti, nella comprensione profonda del protocollo HTTP e del linguaggio SQL, e la mia capacità di utilizzare strumenti di sicurezza offensiva (come i password cracker) per valutare il rischio e proporre difese adeguate. Sono pronto ad applicare queste metodologie di analisi e difesa in contesti aziendali per proteggere le infrastrutture critiche da minacce simili.