SRM EASWARI ENGINEERING COLLEGE
(Autonomous)
Bharathi Salai, Ramapuram,Chennai ,Tamil Nadu – 600 089.

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

LABORATORY RECORD

Academic Year: 2024-2025 / Even Semester

191CSE062J: Web Application Security Laboratory

Name of the Student :

Register Number :

Year/Semester/Section :

EASWARI ENGINEERING COLLEGE

(Autonomous)

Bharathi Salai, Ramapuram, Chennai 600 089.

Department : .....

Laboratory : .....

Name :

Roll No. :

Semester :

Branch :

Subject :

EASWARI ENGINEERING COLLEGE

(Autonomous)
Bharathi Salai, Ramapuram, Chennai 600 089.

Department : .....

PRACTICAL EXAMINATIONS ..... (Month / Year)

BONAFIDE CERTIFICATE

This is to certify that this practical work titled .....
(code)

.....
(Name of the Laboratory)

is the bonafide work of Mr./Miss.....
(Name of the Student)

with Register Number..... in

Semester..... of..... Year in the Department of

.....during the

academic year 20..... –20 .....

Faculty Incharge **Head of the Department**

Submitted for Practical Examination held on ...../...../..... at Easwari

Engineering College, Ramapuram, Chennai – 89.

Internal Examiner **External Examiner**

Index

Exp No: 1	Analyze the difference between HTTP vs HTTPS
Date:	

Aim:

To Analyze the difference between HTTP vs HTTPS

Algorithm:

Step 1: Start
Step 2: Install wireshark Step
3: Start wireshark
Step 4: Analyze the difference between HTTP vs HTTPS
Step 5: View Server Output
Step 6: Stop

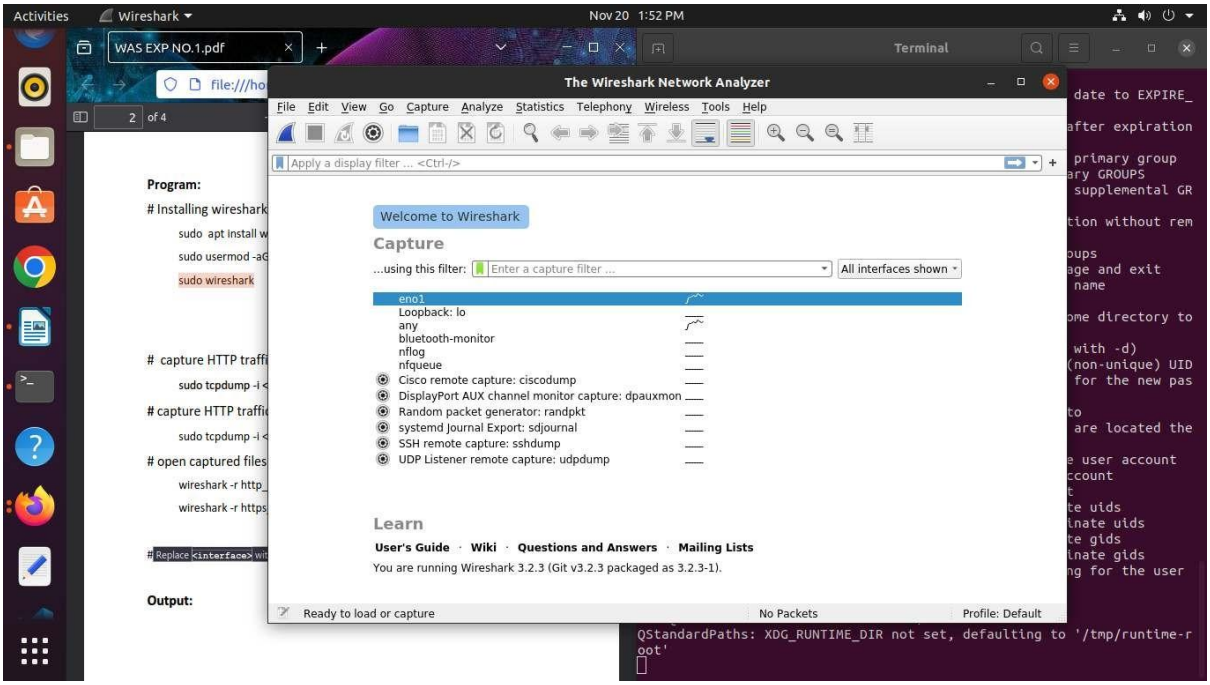
Program:

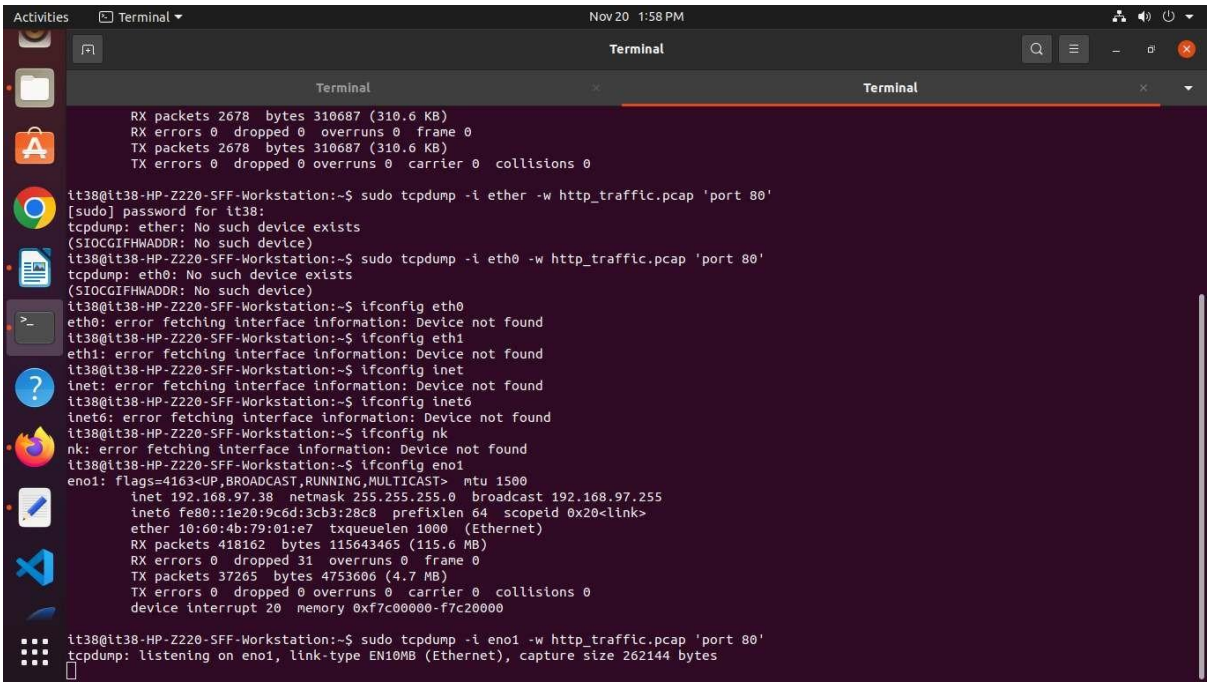
# Installing wireshark in Ubuntu:
sudo apt install wireshark
sudo usermod -aG wireshark $USER
sudo wireshark

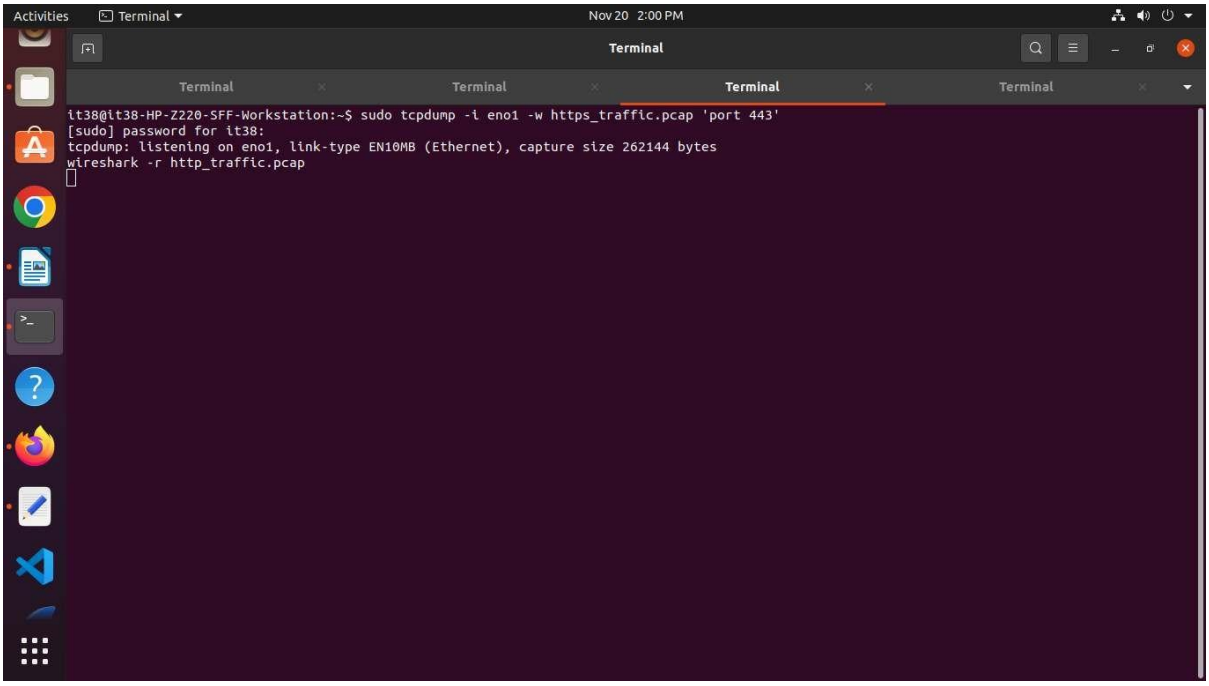
# capture HTTP traffic:
sudo tcpdump -i <interface> -w http_traffic.pcap 'port 80' #
capture HTTP traffic:
sudo tcpdump -i <interface> -w https_traffic.pcap 'port 443' #
open captured files in wireshark:
wireshark -r http_traffic.pcap wireshark
-r https_traffic.pcap

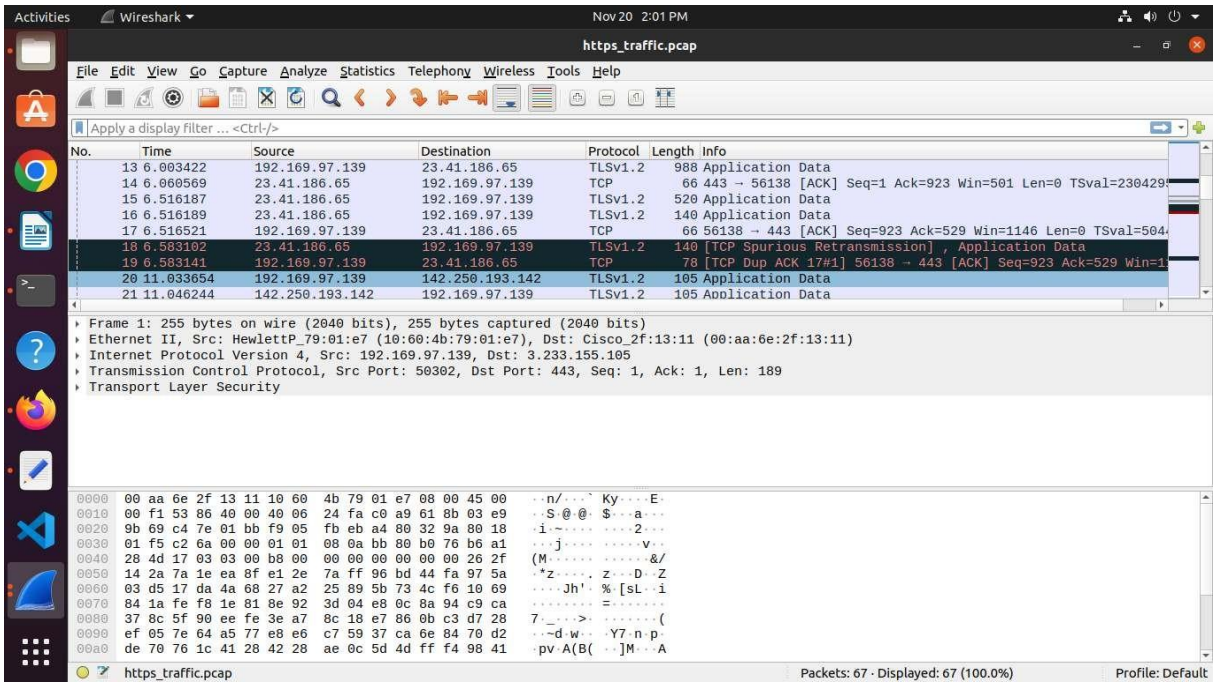
Replace <interfac with your network interface.

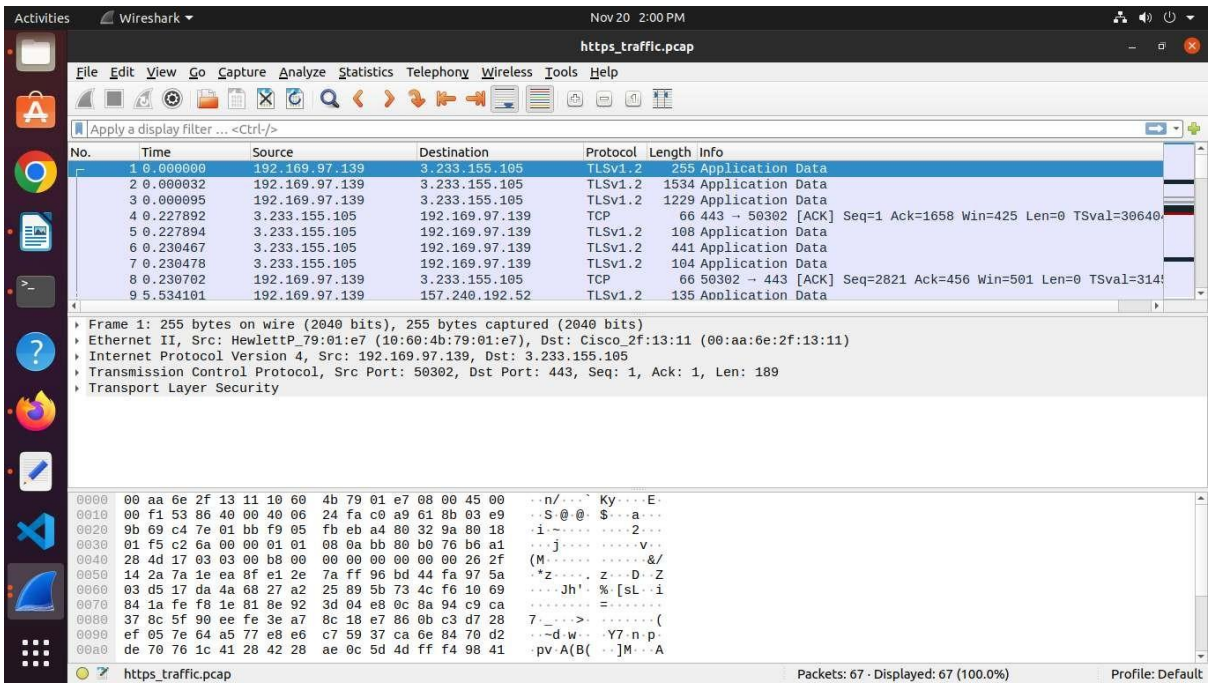
Output:


Activities
Wireshark
Nov 20 1:52 PM
WAS EXP NO.1.pdf
Terminal
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter: Enter a capture filter ...
All interfaces shown
eno1
Loopback: lo
any
bluetooth-monitor
nflog
nfqueue
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).
Ready to load or capture
No Packets
Profile: Default
Program:
Output:
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'


Activities
Terminal
Nov 20 1:58 PM
Terminal
Terminal
Terminal
RX packets 2678 bytes 310687 (310.6 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 2678 bytes 310687 (310.6 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
it38@it38-HP-Z220-SFF-Workstation:~$ sudo tcpdump -i ether -w http_traffic.pcap 'port 80'
[sudo] password for it38:
tcpdump: ether: No such device exists
(SIOCGIFHWADDR: No such device)
it38@it38-HP-Z220-SFF-Workstation:~$ sudo tcpdump -i eth0 -w http_traffic.pcap 'port 80'
tcpdump: eth0: No such device exists
(SIOCGIFHWADDR: No such device)
it38@it38-HP-Z220-SFF-Workstation:~$ ifconfig eth0
eth0: error fetching interface information: Device not found
it38@it38-HP-Z220-SFF-Workstation:~$ ifconfig eth1
eth1: error fetching interface information: Device not found
it38@it38-HP-Z220-SFF-Workstation:~$ ifconfig inet
inet: error fetching interface information: Device not found
it38@it38-HP-Z220-SFF-Workstation:~$ ifconfig inet6
inet6: error fetching interface information: Device not found
it38@it38-HP-Z220-SFF-Workstation:~$ ifconfig nk
nk: error fetching interface information: Device not found
it38@it38-HP-Z220-SFF-Workstation:~$ ifconfig eno1
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.97.38 netmask 255.255.255.0 broadcast 192.168.97.255
inet6 fe80::1e20:9c6d:3cb3:28c8 prefixlen 64 scopeid 0x20<link>
ether 10:60:4b:79:01:e7 txqueuelen 1000 (Ethernet)
RX packets 418162 bytes 115643465 (115.6 MB)
RX errors 0 dropped 31 overruns 0 frame 0
TX packets 37265 bytes 4753606 (4.7 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 20 memory 0xf7c00000-f7c20000
it38@it38-HP-Z220-SFF-Workstation:~$ sudo tcpdump -i eno1 -w http_traffic.pcap 'port 80'
tcpdump: listening on eno1, link-type EN10MB (Ethernet), capture size 262144 bytes


Activities
Terminal
Nov 20 2:00 PM
Terminal
Terminal
Terminal
Terminal
Terminal
it38@it38-HP-Z220-SFF-Workstation:~$ sudo tcpdump -i eno1 -w https_traffic.pcap 'port 443'
[sudo] password for it38:
tcpdump: listening on eno1, link-type EN10MB (Ethernet), capture size 262144 bytes
wireshark -r http_traffic.pcap


Activities
Wireshark
Nov 20 2:01 PM
https_traffic.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
13 6.003422 192.169.97.139 23.41.186.65 TLSv1.2 988 Application Data
14 6.060569 23.41.186.65 192.169.97.139 TCP 66 443 → 56138 [ACK] Seq=1 Ack=923 Win=501 Len=0 TSval=230429
15 6.516187 23.41.186.65 192.169.97.139 TLSv1.2 520 Application Data
16 6.516189 23.41.186.65 192.169.97.139 TLSv1.2 140 Application Data
17 6.516521 192.169.97.139 23.41.186.65 TCP 66 56138 → 443 [ACK] Seq=923 Ack=529 Win=1146 Len=0 TSval=5044
18 6.583102 23.41.186.65 192.169.97.139 TLSv1.2 140 [TCP Spurious Retransmission] , Application Data
19 6.583141 192.169.97.139 23.41.186.65 TCP 78 [TCP Dup ACK 17#1] 56138 → 443 [ACK] Seq=923 Ack=529 Win=1
20 11.033654 192.169.97.139 142.250.193.142 TLSv1.2 105 Application Data
21 11.046244 142.250.193.142 192.169.97.139 TLSv1.2 105 Application Data
Frame 1: 255 bytes on wire (2040 bits), 255 bytes captured (2040 bits)
Ethernet II, Src: HewlettP_79:01:e7 (10:60:4b:79:01:e7), Dst: Cisco_2f:13:11 (00:aa:6e:2f:13:11)
Internet Protocol Version 4, Src: 192.169.97.139, Dst: 3.233.155.105
Transmission Control Protocol, Src Port: 50302, Dst Port: 443, Seq: 1, Ack: 1, Len: 189
Transport Layer Security
0000 00 aa 6e 2f 13 11 10 60 4b 79 01 e7 08 00 45 00
0010 00 f1 53 86 40 00 40 06 24 fa c0 a9 61 8b 03 e9
0020 9b 69 c4 7e 01 bb f9 05 fb eb a4 80 32 9a 80 18
0030 01 f5 c2 6a 00 00 01 01 08 0a bb 80 b0 76 b6 a1
0040 28 4d 17 03 03 00 b8 00 00 00 00 00 00 00 26 2f
0050 14 2a 7a 1e ea 8f e1 2e 7a ff 96 bd 44 fa 97 5a
0060 03 d5 17 da 4a 68 27 a2 25 89 5b 73 4c f6 10 69
0070 84 1a fe f8 1e 81 8e 92 3d 04 e8 0c 8a 94 c9 ca
0080 37 8c 5f 90 ee fe 3e a7 8c 18 e7 86 0b c3 d7 28
0090 ef 05 7e 64 a5 77 e8 e6 c7 59 37 ca 6e 84 70 d2
00a0 de 70 76 1c 41 28 42 28 ae 0c 5d 4d ff f4 98 41
https_traffic.pcap
Packets: 67 · Displayed: 67 (100.0%)
Profile: Default



Result:

Thus, the experiment to analyze the difference between HTTP vs HTTPS is executed and verified successfully.

Exp No: 2	Analyze the various security mechanism embedded with different protocols
Date:	

Aim:

To Analyze the various security mechanism embedded with different protocols

Algorithm:

Step 1: Start
Step 2: Start wireshark
Step 3: Analyze the various security mechanism embedded with different protocol
Step 4: View Server Output
Step 5: Stop

Program:

#capture HTTPS traffic:

```
sudo tcpdump -i <interface> -w https_traffic.pcap 'port 443'
```

#capture IPsec traffic:

```
sudo tcpdump -i <interface> -w ipsec_traffic.pcap 'ip proto 50 or ip proto 51'
```

#capture SSH traffic:

```
sudo tcpdump -i <interface> -w ssh_traffic.pcap 'port 22'
```

#capture WPA/WPA2 traffic:

```
sudo tcpdump -i <wireless_interface> -w wpa_traffic.pcap 'type mgt subtype assoc-req or type mgt subtype assoc-resp'
```

#capture DNSSEC traffic:

```
sudo tcpdump -i <interface> -w dnssec_traffic.pcap 'port 53'
```

#capture OAuth traffic:

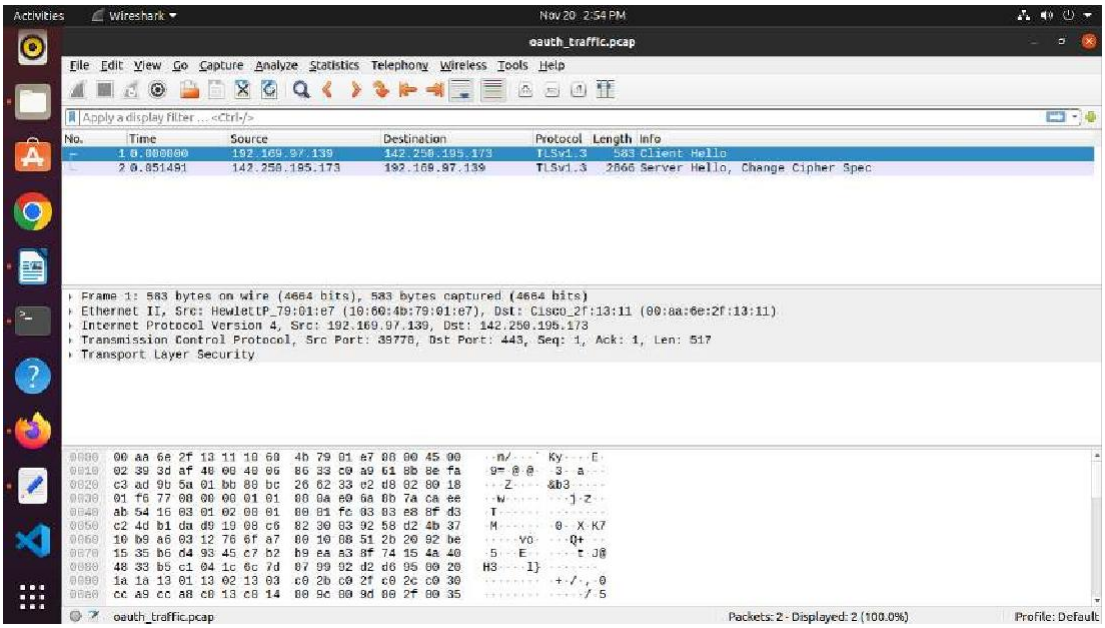
```
sudo tcpdump -i <interface> -w oauth_traffic.pcap 'port 443 and (tcp[((tcp[12] & 0xf0) >> 2):1] = 0x16 or tcp[((tcp[12] & 0xf0) >> 2):1] = 0x80)'
```

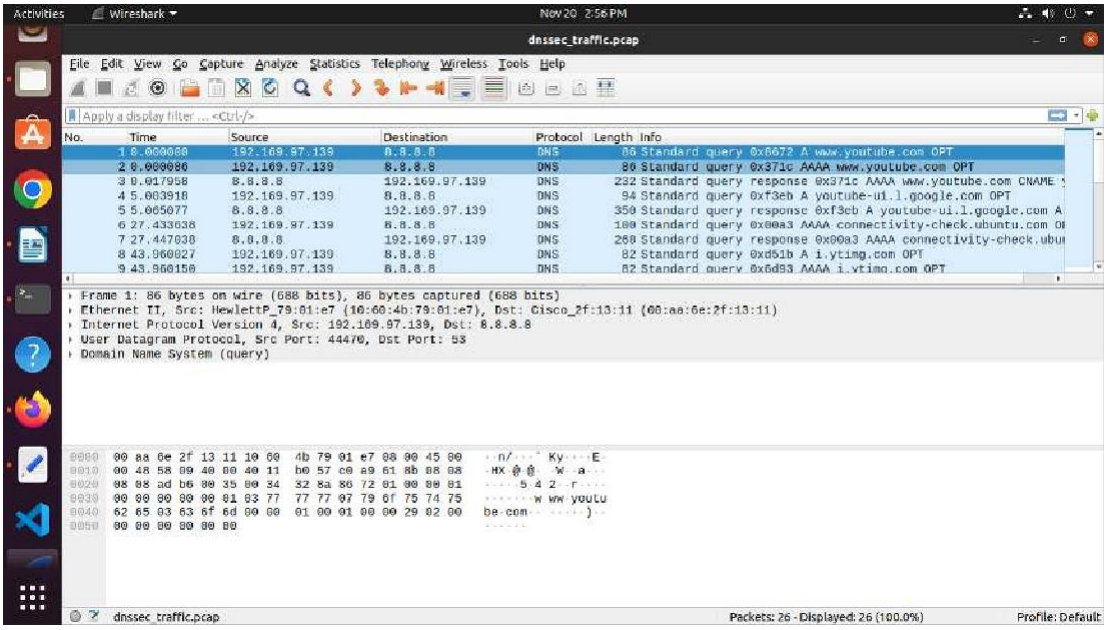
#after capturing packets , analyze them using wireshark:

```
wireshark -r <filename.pcap>
```

Replace **<filename.pca**with the name of the captured file. This opens Wireshark with the specified packet capture file for detailed analysis.

Output:





Result:

Thus, the experiment to analyze the various security mechanism embedded with different protocols is executed and verified successfully.

Exp No: 3	Identify the Vulnerabilities Using Owasp Zap Tool
Date:	

Aim:

To Identify the Vulnerabilities Using Owasp Zap Tool

Procedure:

1. Install OWASP ZAP:
   - Download and install OWASP ZAP from the official website.
2. Configure Browser Proxy
   - Set up your browser to use ZAP as a proxy server (Default: localhost, Port: 8080).

Experiment Steps:

1. Launch OWASP ZAP:
   - Open the OWASP ZAP tool
2. Start ZAP Proxy:
   - In ZAP, click on the 'Quick Start' tab.
   - Start the ZAP Proxy.
3. Set Target Application:
   - Go to the "Sites" tab.
   - Enter the URL of the target application.
   - Right-click on the URL and choose "Include in Context" > "Default Context" to add it for scanning.
4. Spider the Application:
   - Go to the "Spider" tab.
   - Right-click on the target URL and select "Spider" to crawl the application.
   - Let ZAP crawl and map the application structure.
5. Active Scan:
   - Go to the "Attack" tab.
   - Choose "Active Scan."
   - Configure the scan settings (scope, intensity, etc.).
   - Start the active scan on the target application.
6. Review Scan Results:
   - After the scan completes, go to the "Alerts" tab.
   - View the list of vulnerabilities discovered by ZAP.

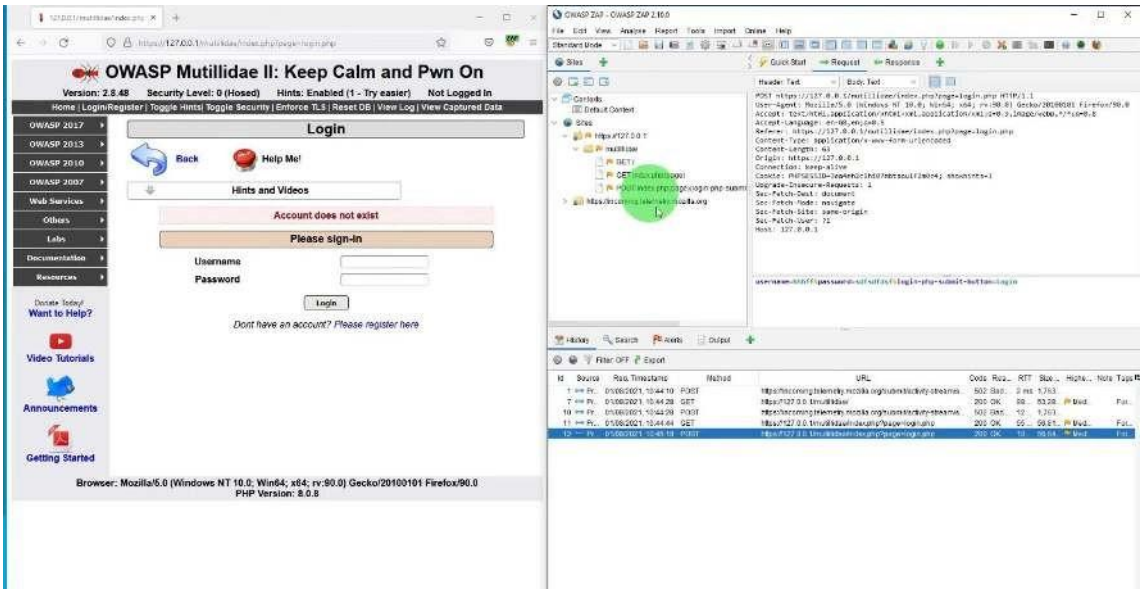
7. Investigate Vulnerabilities:
   - Click on each vulnerability to get detailed information.
   - Verify and understand the nature and potential impact of each issue.

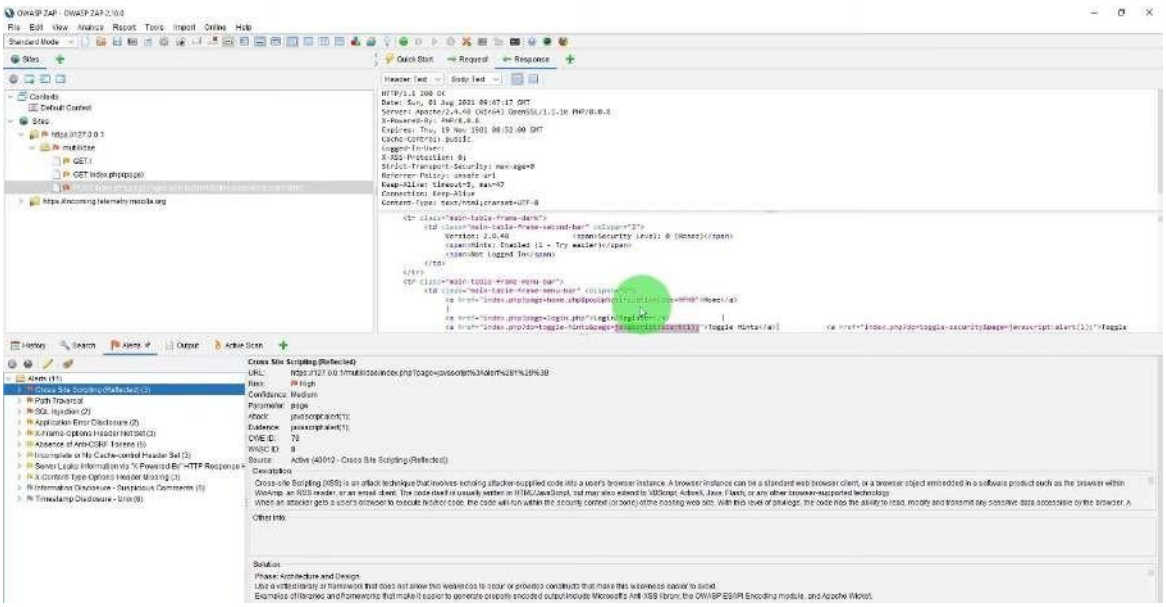
8. Prioritize and Document:
   - Prioritize vulnerabilities based on severity and potential impact.
   - Document the identified vulnerabilities with descriptions, severity levels, affected URLs, and possible remediation steps.

9. Report Generation:
   - Go to the "Report" tab.
   - Generate a comprehensive report summarizing the identified vulnerabilities and their details.
   - Choose the appropriate report format (HTML, PDF, etc.).

10. Remediation and Re-scan:
    - Work on fixing or mitigating the identified vulnerabilities.
    - After making changes, perform another scan using ZAP to verify that the issues have been resolved.

11. Continuous Monitoring:
    - Schedule regular scans using ZAP to continuously monitor the application's security posture.
    - Regularly review and update the security measures based on new findings





Result:

Thus, the experiment to identify vulnerabilities using OWASP Zap tool is executed and verified successfully.

Exp No: 4	Create a simple REST API using python to do the GET, POST, PUT and DELETE operations
Date:	

Aim:

To create a simple REST API using python to do the GET, POST, PUT and DELETE operations

Algorithm:

Step 1: Start
Step 2: Install Flask
Step 3: Start the Flask App
Step 4: Use Postman to Test Endpoints
Step 5: View Server Output
Step 6: Stop

Program:

```
from flask import Flask, jsonify, request
app = Flask(__name__)
# Sample data data
= [
   {'id': 1, 'name': 'Item 1'},
   {'id': 2, 'name': 'Item 2'},
   {'id': 3, 'name': 'Item 3'}
]
# GET request to retrieve all items
@app.route('/items', methods=['GET'])
def get_items():
   return jsonify({'items': data})
# GET request to retrieve a specific item by ID
@app.route('/items/<int:item_id>', methods=['GET'])
def get_item(item_id):
   item = next((item for item in data if item['id'] == item_id), None) if
   item:
      return jsonify({'item': item})
   else:
      return jsonify({'message': 'Item not found'}), 404
# POST request to add a new item @app.route('/items',
methods=['POST'])
def add_item():
```

```
    new_item = {'id': len(data) + 1, 'name': request.json['name']}
    data.append(new_item)
    return jsonify({'item': new_item}), 201
# PUT request to update a specific item by ID
@app.route('/items/<int:item_id>', methods=['PUT'])
def update_item(item_id):
    item = next((item for item in data if item['id'] == item_id), None) if
    item:
        item['name'] = request.json['name'] return
        jsonify({'item': item})
    else:
        return jsonify({'message': 'Item not found'}), 404
# DELETE request to remove a specific item by ID
@app.route('/items/<int:item_id>', methods=['DELETE'])
def delete_item(item_id):
    global data
    data = [item for item in data if item['id'] != item_id]
    return jsonify({'message': 'Item deleted'}), 200
if __name__== '__main__':
    app.run(debug=True)
```

Procedure and Output:

Step 1: Install Flask

>>>pip install flask

Step 2: Start the Flask App

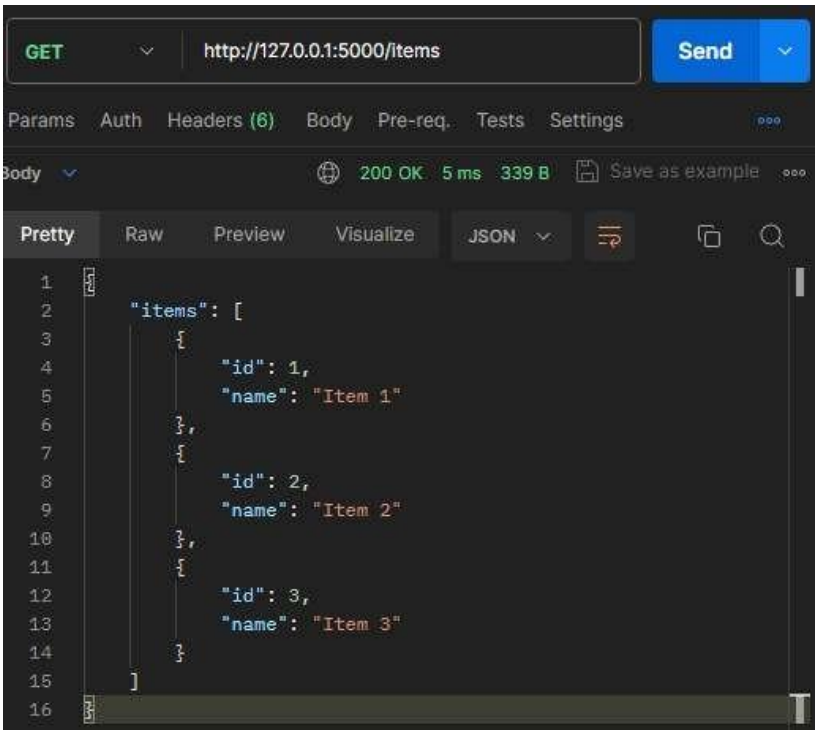
Save the code as app.py and execute

>>>python app.py

Copy the url produced http://127.0.0.1:5000

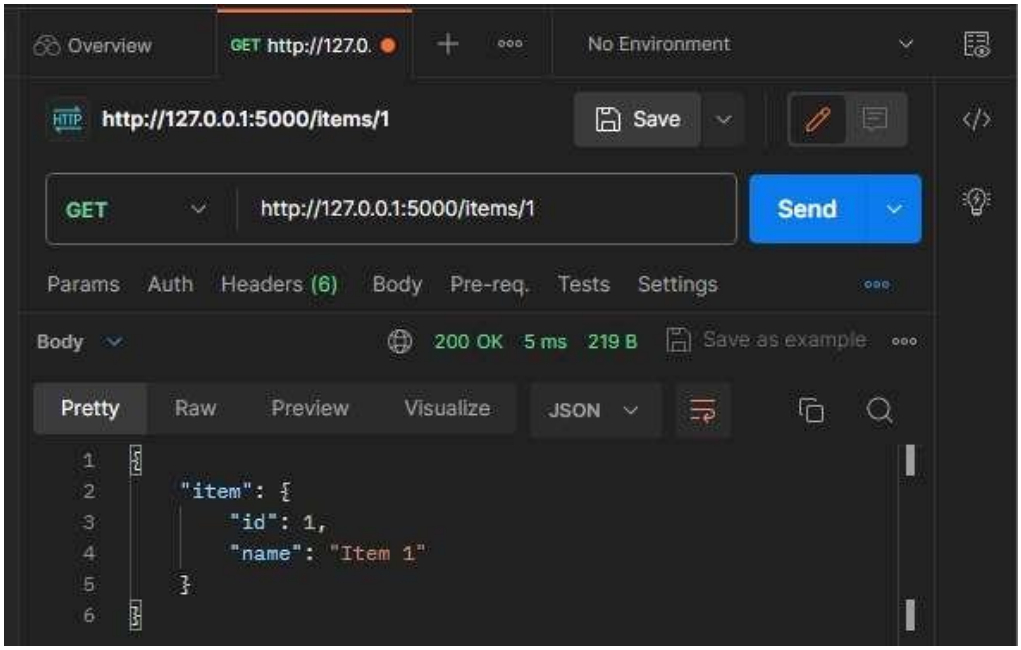
Step 3: Use Postman to Test Endpoints

1. **GET Request to Retrieve All Items:**
    - Set the request type to **GET**.
    - Enter the URL: **http://127.0.0.1:5000/items**
    - Click "Send."

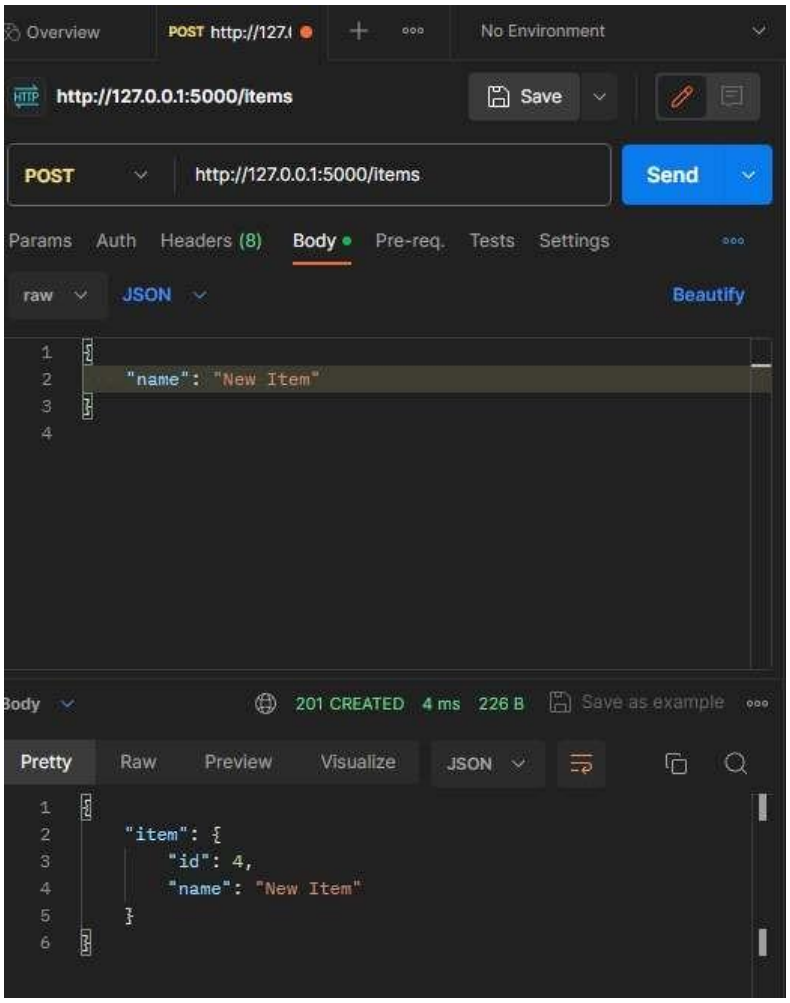


2. GET Request to Retrieve a Specific Item by ID:

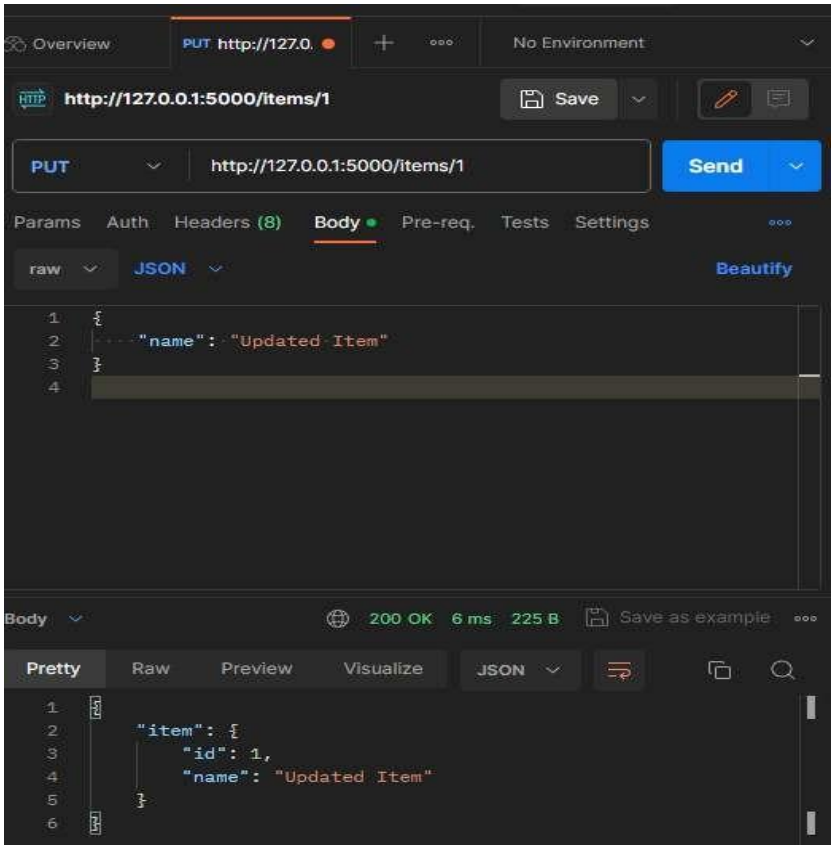
- Set the request type to **GET**.
- Enter the URL for a specific item ID, for example: **http://127.0.0.1:5000/items/1**
- Click "Send."



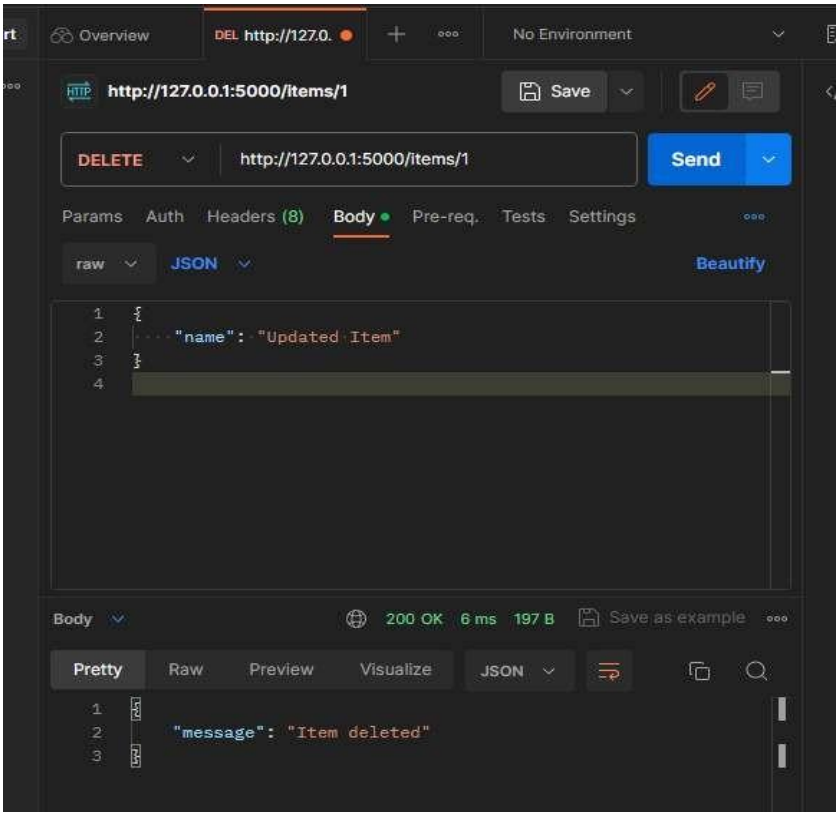
3. **POST Request to Add a New Item:**
    - Set the request type to **POST**.
    - Enter the URL: **http://127.0.0.1:5000/items**
    - Go to the "Body" tab, select "raw" and choose "JSON (application/json)".Enter the request body
    - Click "Send."



4. **PUT Request to Update an Existing Item:**
    - Set the request type to **PUT**.
    - Enter the URL for a specific item ID, for example: **http://127.0.0.1:5000/items/1**
    - Go to the "Body" tab, select "raw" and choose "JSON (application/json)".
    - Enter the updated information
    - Click "Send."



5. **DELETE Request to Remove a Specific Item by ID:**
    - Set the request type to **DELETE**.
    - Enter the URL for a specific item ID, for example: **http://127.0.0.1:5000/items/1**
    - Click "Send."



Step 4: View Server Output

```
C:\Users\NAVEEN\Desktop>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 598-854-429
127.0.0.1 - - [16/Nov/2023 18:40:00] "GET /items HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2023 18:40:08] "GET /items/1 HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2023 18:40:25] "POST /items HTTP/1.1" 201 -
127.0.0.1 - - [16/Nov/2023 18:40:38] "PUT /items/1 HTTP/1.1" 200 -
127.0.0.1 - - [16/Nov/2023 18:40:44] "DELETE /items/1 HTTP/1.1" 200 -
```

Result:

Thus, the experiment to create a simple REST API using python to do the GET, POST, PUT and DELETE operations is executed and verified successfully.

Exp No: 5	Install Burp Suite to do following vulnerabilities-SQL Injection
Date:	

Aim:

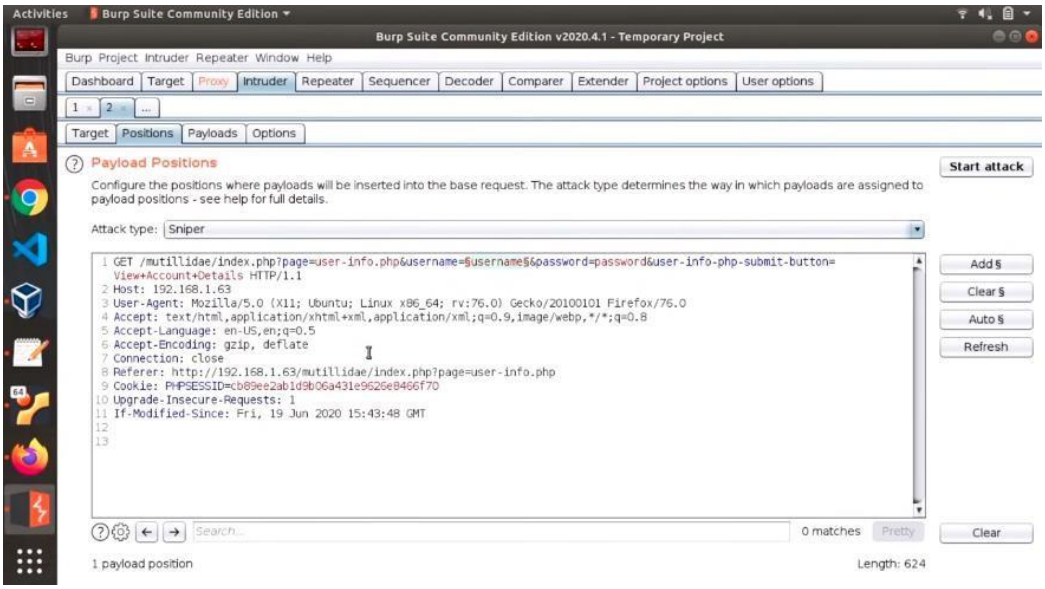
To Install Burp Suite to do following vulnerabilities:

- SQL Injection

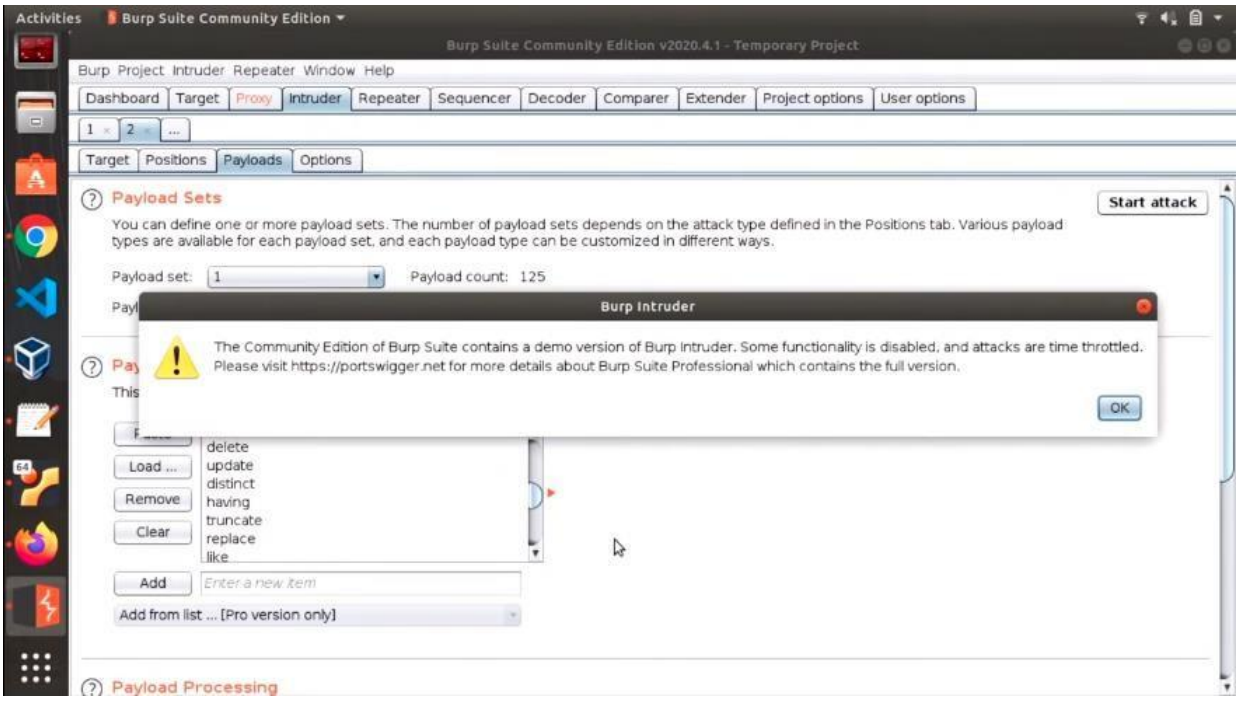
Procedure:

1. Install Burpsuite and connect the burpsuite proxy in browser proxy settings.
2. Turn on the intercept and search for the website which needs to be captured.

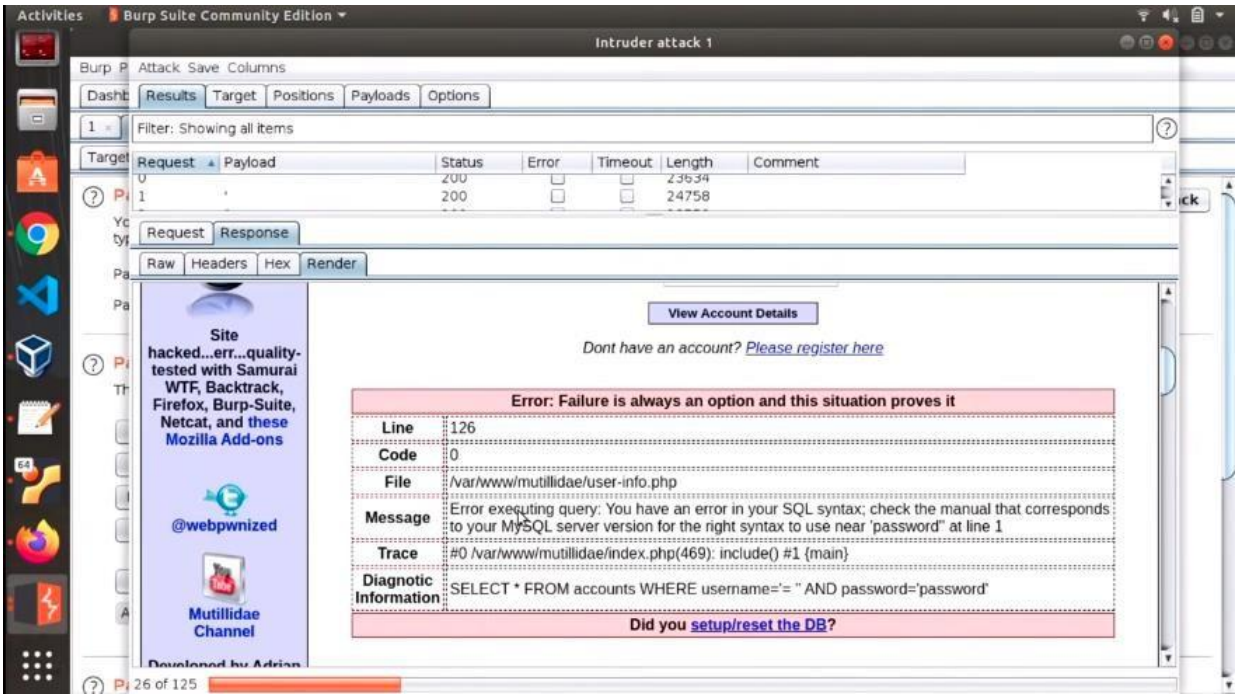
3. Send the intercepted request to the intruder and load the SQL Injection File from the device which is already installed.

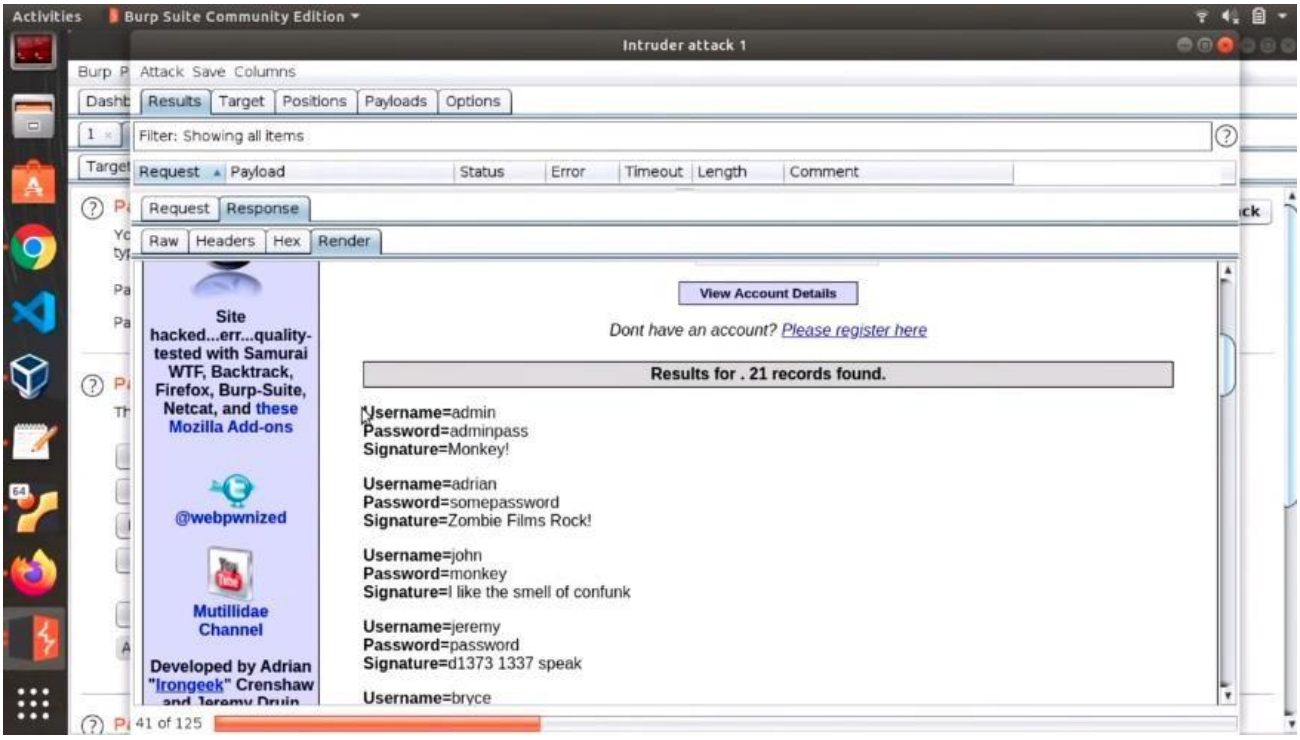


4. Start the attack in the intruder and search for the requests & responses in the render screen for SQL Injection.



5. After the attack, some response render shows the username and password for the webpage.





Result:

Thus the above vulnerability is successfully executed and verified.

Exp No: 6	Install Burp Suite to do following vulnerabilities-Cross-Site Scripting (XSS)
Date:	

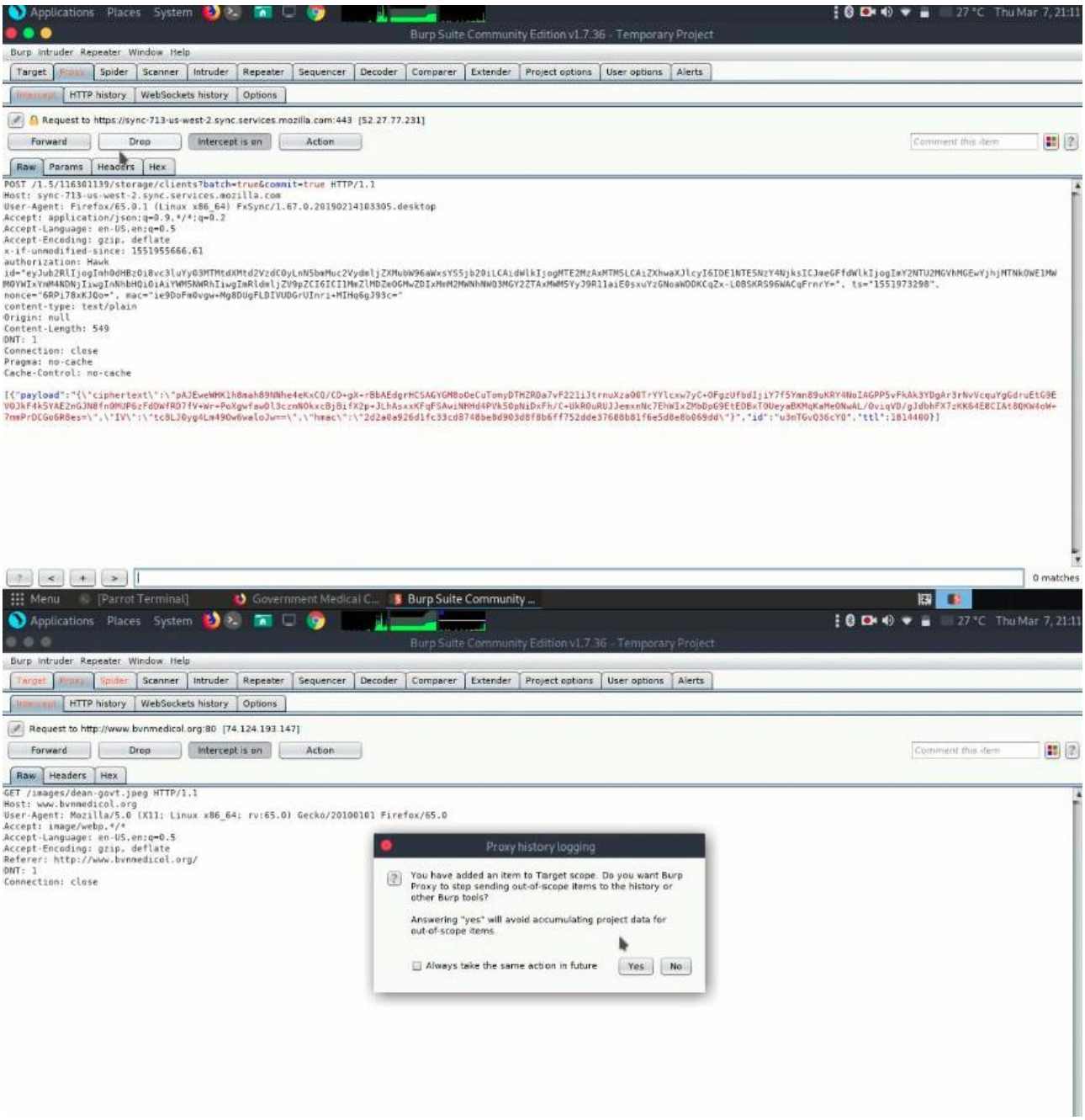
Aim:

To Install Burp Suite to do following vulnerabilities:

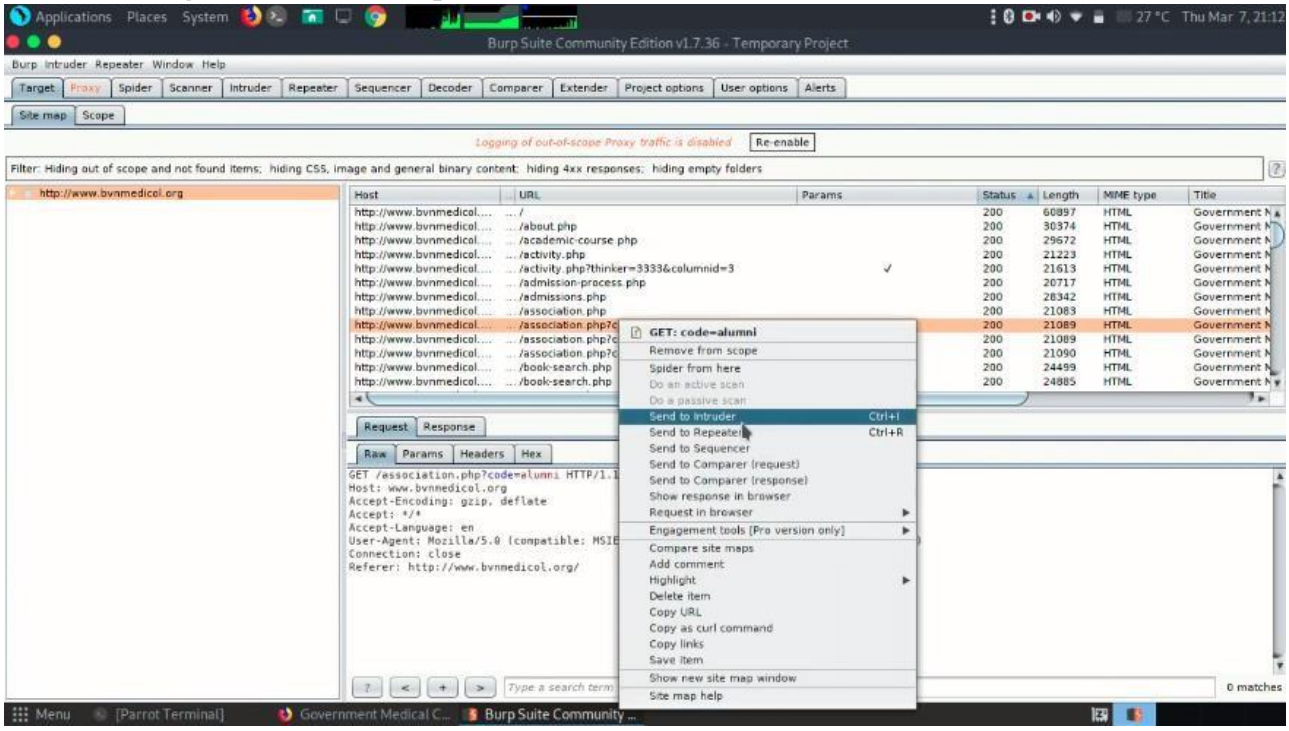
- Cross-Site Scripting (XSS)

Procedure:

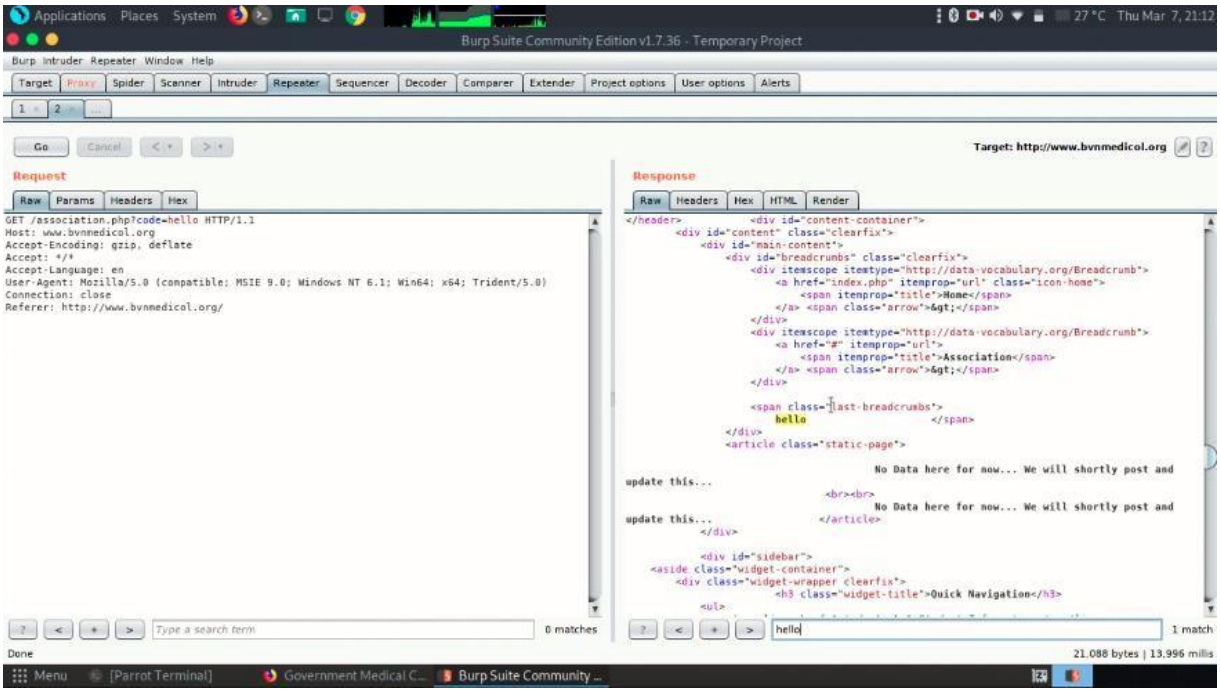
1. Turn on the intercept and search for the website which needs to be captured.
2. Add the captured request to the Target scope

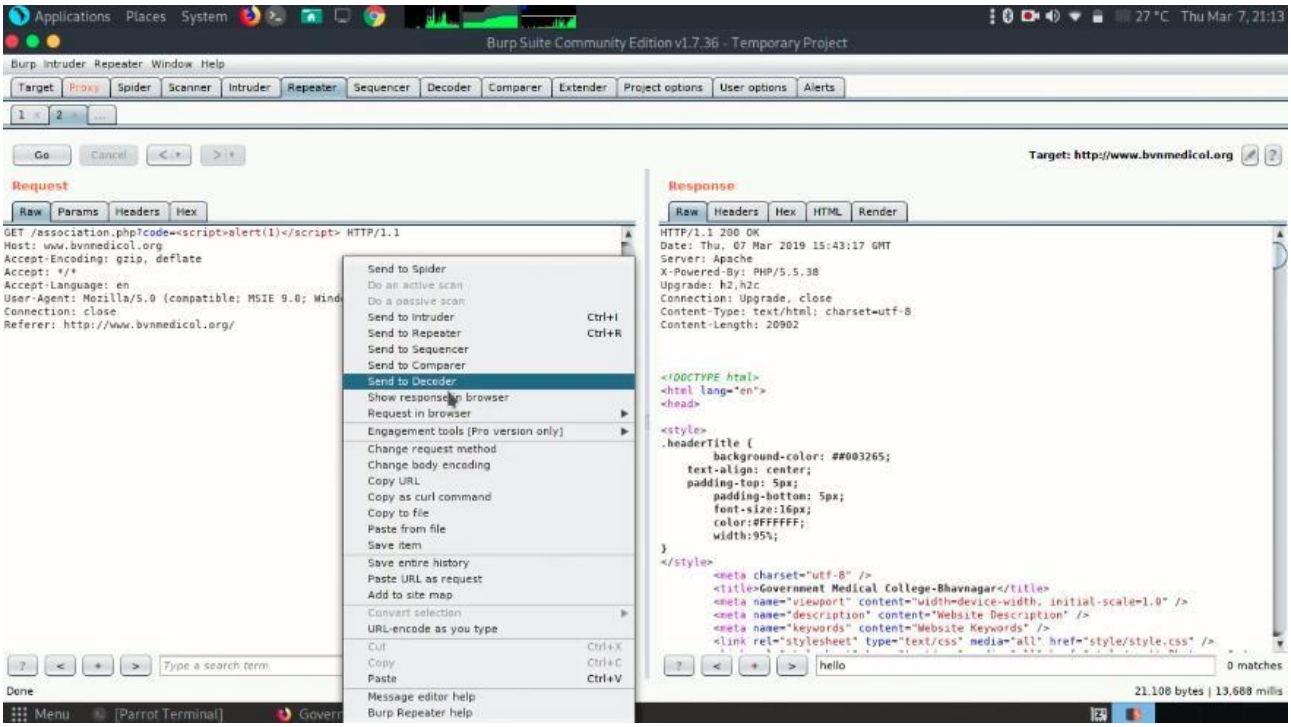


3. Go to Target section and search for the captured request in the item field and send the target item to the repeater.

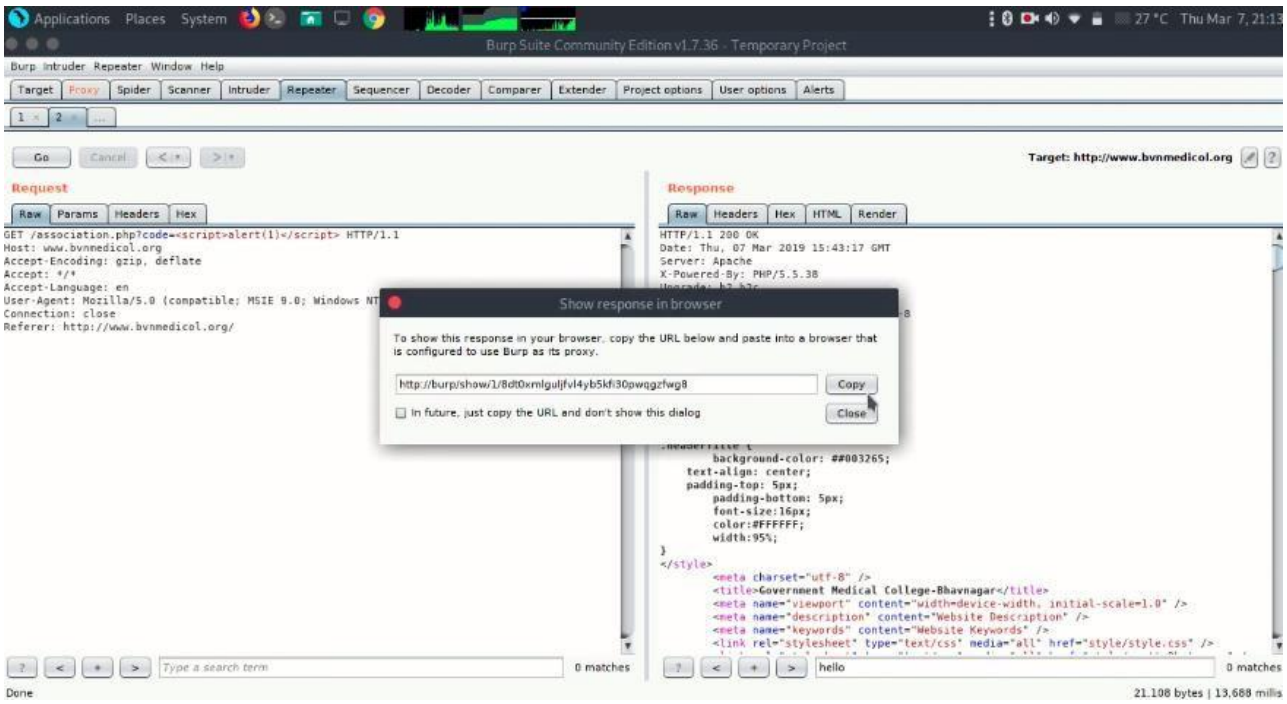


4. The request in the repeater section will be modified and send to the Decoder.

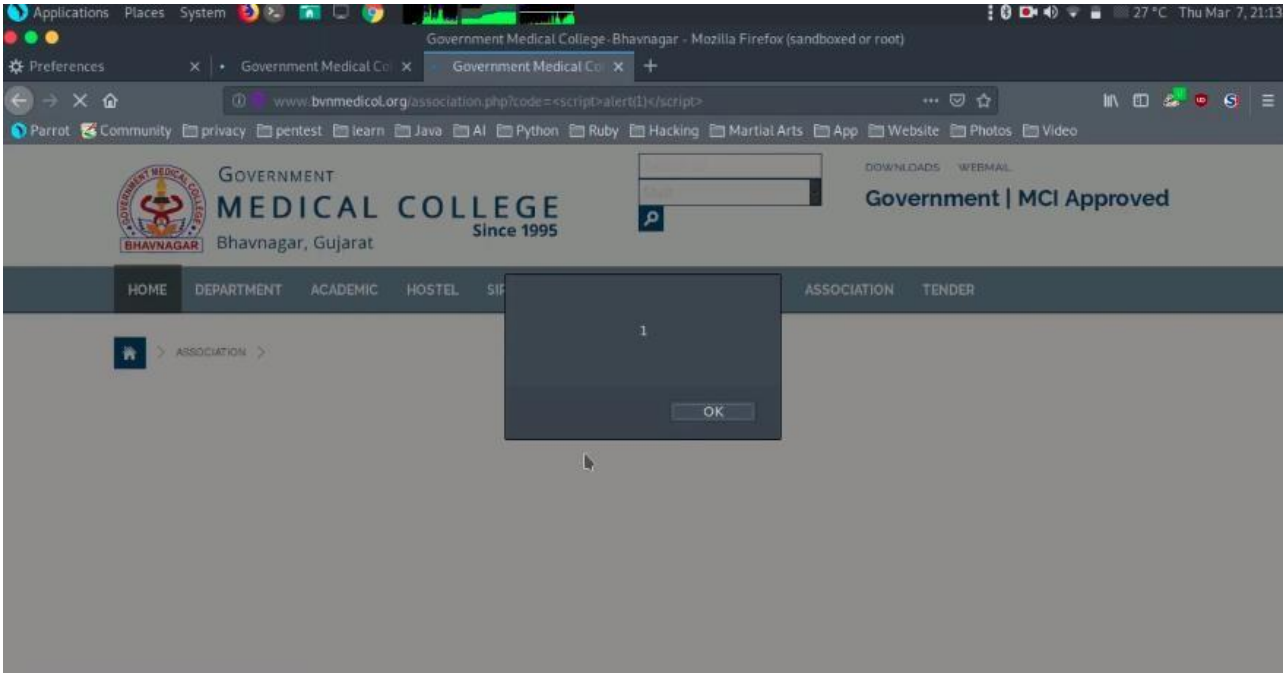




5. Before sending the response to the browser, Copy the URL below and paste into a browser that to configured to use Burp as its proxy.



6. Open the browser to see the modified response. An alert message is popup while opening the website.



Result:

Thus the above vulnerability is successfully executed and verified.

Exp No: 7	**Attach the website using social engineering method**
Date:	

Aim:

To attach the website using social engineering method

Procedure & Output:

Installation of Social engineering toolkit :

Step 1: Open your Kali Linux Terminal and move to Desktop

>>>cd Desktop

Step 2: As of now you are on a desktop so here you have to create a new directory named SEToolkit using the following command.

>>>mkdir SEToolkit

Step 3: Now as you are in the Desktop directory however you have created a SEToolkit directory so move to SEToolkit directory using the following command

>>>cd SEToolkit

Step 4: Now you are in SEToolkit directory here you have to clone SEToolkit from GitHub so you can use it.

>>>git clone https://github.com/trustedsec/social-engineer-toolkit setoolkit/

Step 5: Social Engineering Toolkit has been downloaded in your directory now you have to move to the internal directory of the social engineering toolkit using the following command.

>>>cd setoolkit

Step 6: Congratulations you have finally downloaded the social engineering toolkit in your directory SEToolkit. Now it's time to install requirements using the following command.

`pip3 install -r requirements.txt

```
root@kali:~/Desktop/SEToolkit/setoolkit# pip3 install -r requirements.txt
Requirement already satisfied: pexpect in /usr/lib/python3/dist-packages (from -r requir
ements.txt (line 1)) (4.6.0)
Requirement already satisfied: pycrypto in /usr/lib/python3/dist-packages (from -r requi
rements.txt (line 2)) (2.6.1)
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from -r requi
rements.txt (line 3)) (2.22.0)
Requirement already satisfied: pyopenssl in /usr/lib/python3/dist-packages (from -r requ
irements.txt (line 4)) (19.0.0)
Requirement already satisfied: pefile in /usr/lib/python3/dist-packages (from -r require
ments.txt (line 5)) (2019.4.18)
Requirement already satisfied: impacket in /usr/lib/python3/dist-packages (from -r requi
rements.txt (line 6)) (0.9.20)
Requirement already satisfied: qrcode in /usr/lib/python3/dist-packages (from -r require
ments.txt (line 8)) (6.1)
Requirement already satisfied: pillow in /usr/lib/python3/dist-packages (from -r require
ments.txt (line 9)) (6.2.1)
Requirement already satisfied: pymssql<3.0 in /usr/lib/python3/dist-packages (from -r re
quirements.txt (line 11)) (2.1.4)
Requirement already satisfied: ldapdomaindump≥0.9.0 in /usr/lib/python3/dist-packages (
from impacket->-r requirements.txt (line 6)) (0.9.1)
```

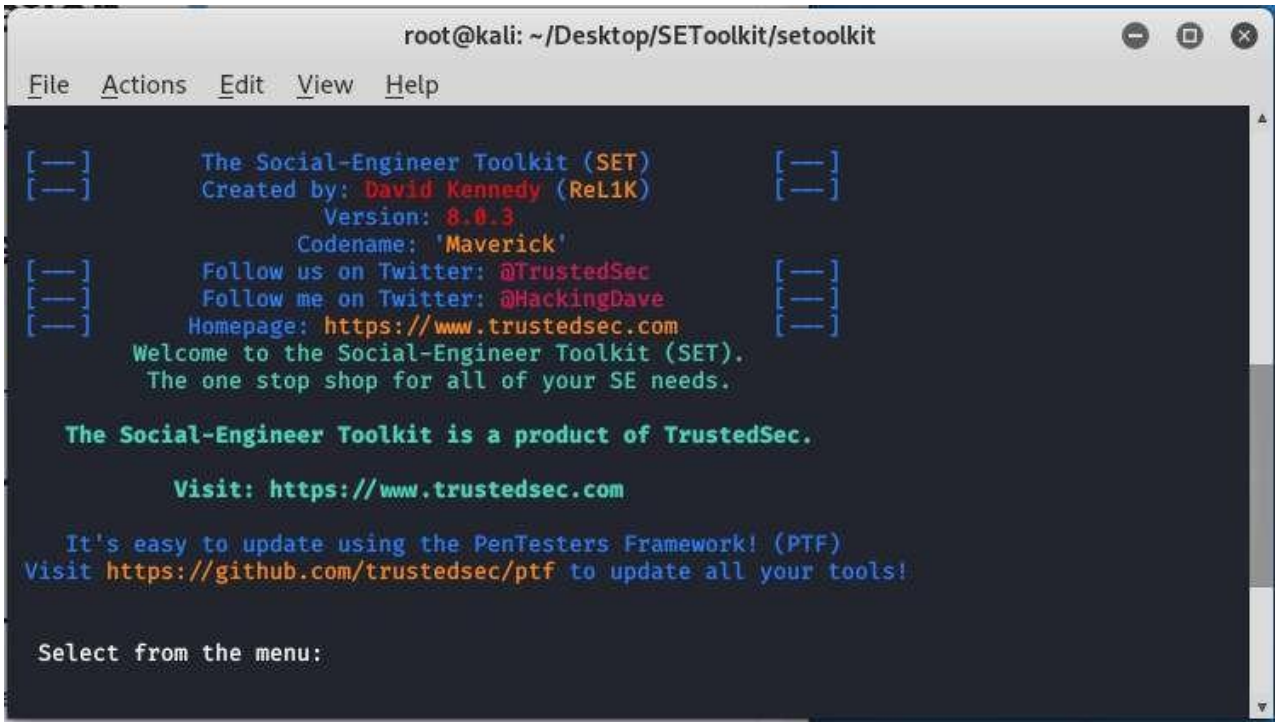
Step 7: All the requirements have been downloaded in your setoolkit. Now it's time to install the requirements that you have downloaded

>>>python setup.py

Step 8: Finally all the processes of installation have been completed now it's time to run the social engineering toolkit .to run the SEToolkit type following command.

>>>Setoolkit

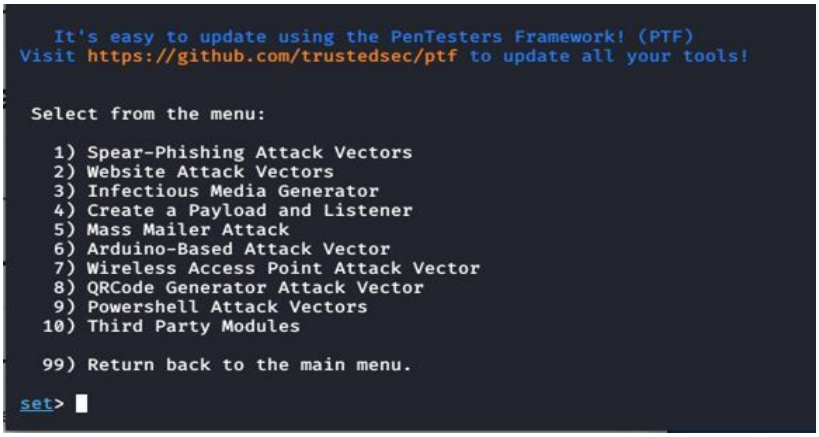
Step 9: At this step, setoolkit will ask you (y) or (n). Type y and your social engineering toolkit will start running.



Step 10: Now your setoolkit has been downloaded into your system now it's time to use it .now you have to choose an option from the following options .here we are choosing option 2

Website Attack Vector

Option: 2



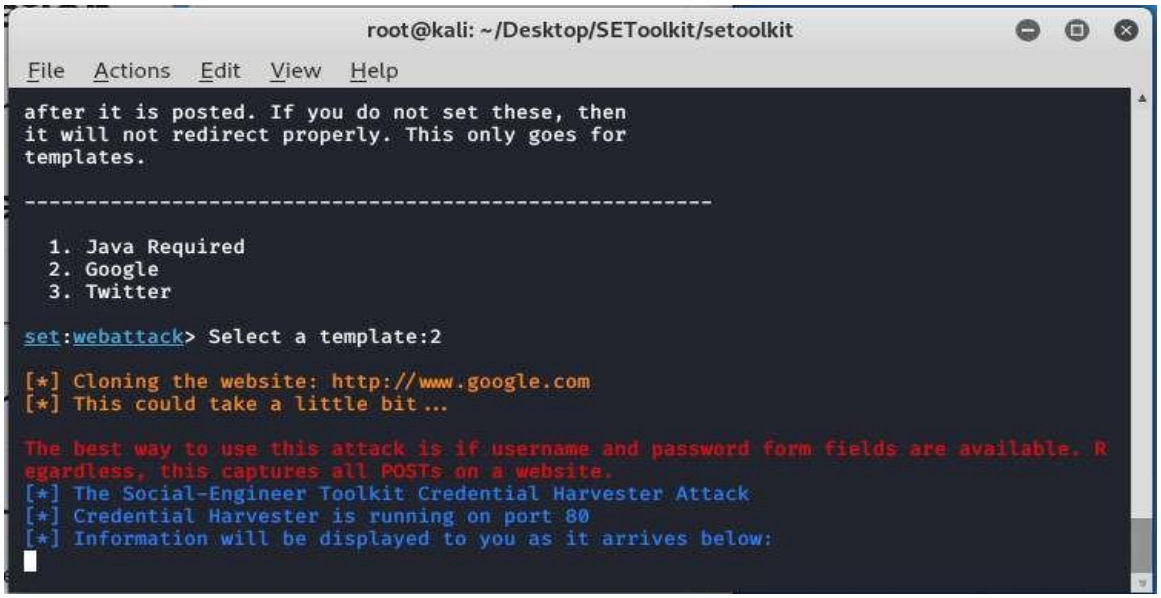
Step 11: Now we are about to set up a phishing page so here we will choose option 3 that is the credential harvester attack method.

Option: 3

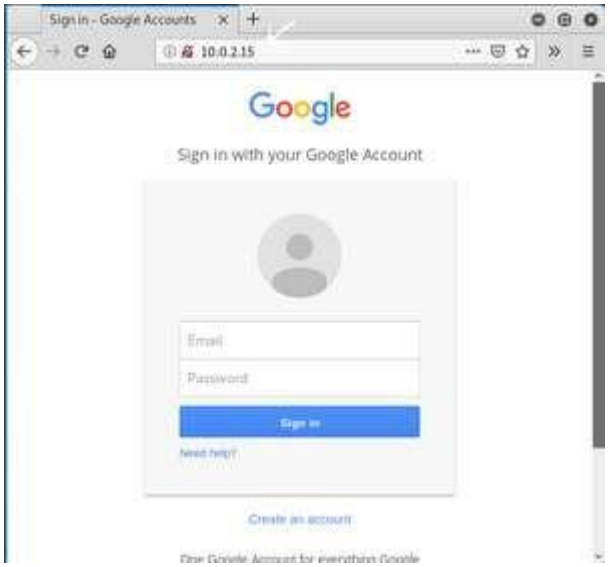
Step 12: Now since we are creating a Phishing page so here we will choose option 1 that is web templates.

Option: 1

Step 13: Create a google phishing page so choose option 2 for that then a phishing page will be generated on your localhost.



Step 14: Social engineering toolkit is creating a phishing page of google.



RESULT:

Thus, the experiment to attach the website using social engineering method is executed and verified successfully.