



WAS final - Lab Manual for Web application Security

web application security (Anna University)



Scan to open on Studocu

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

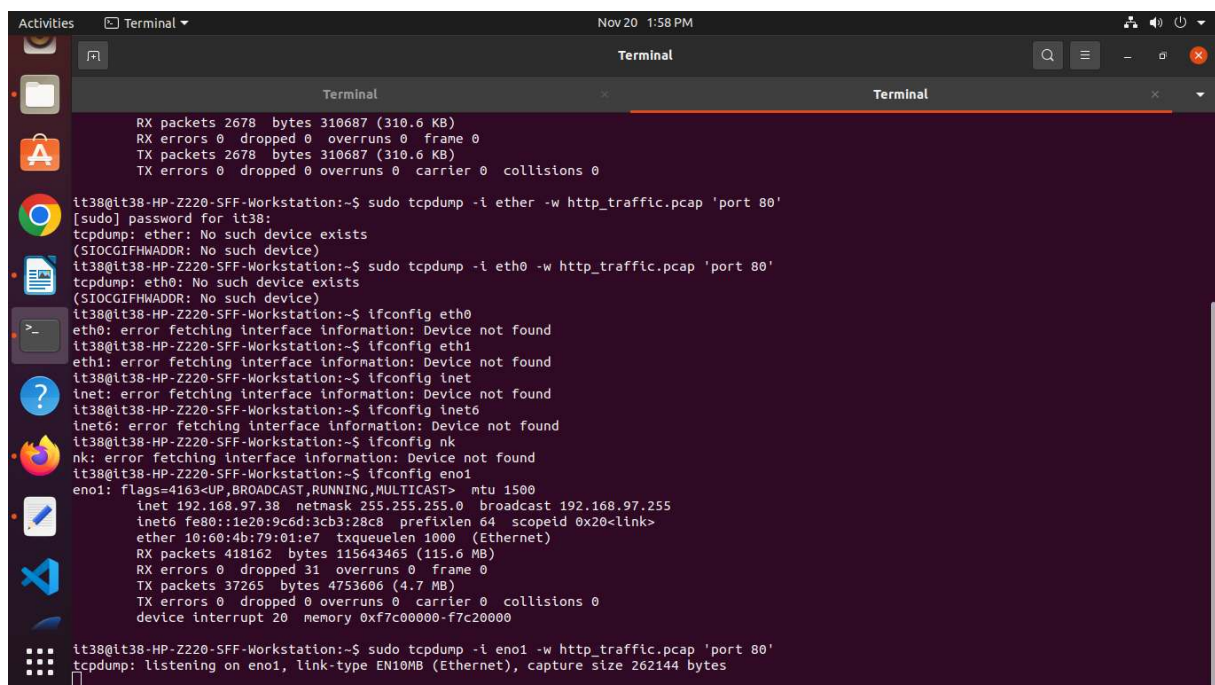
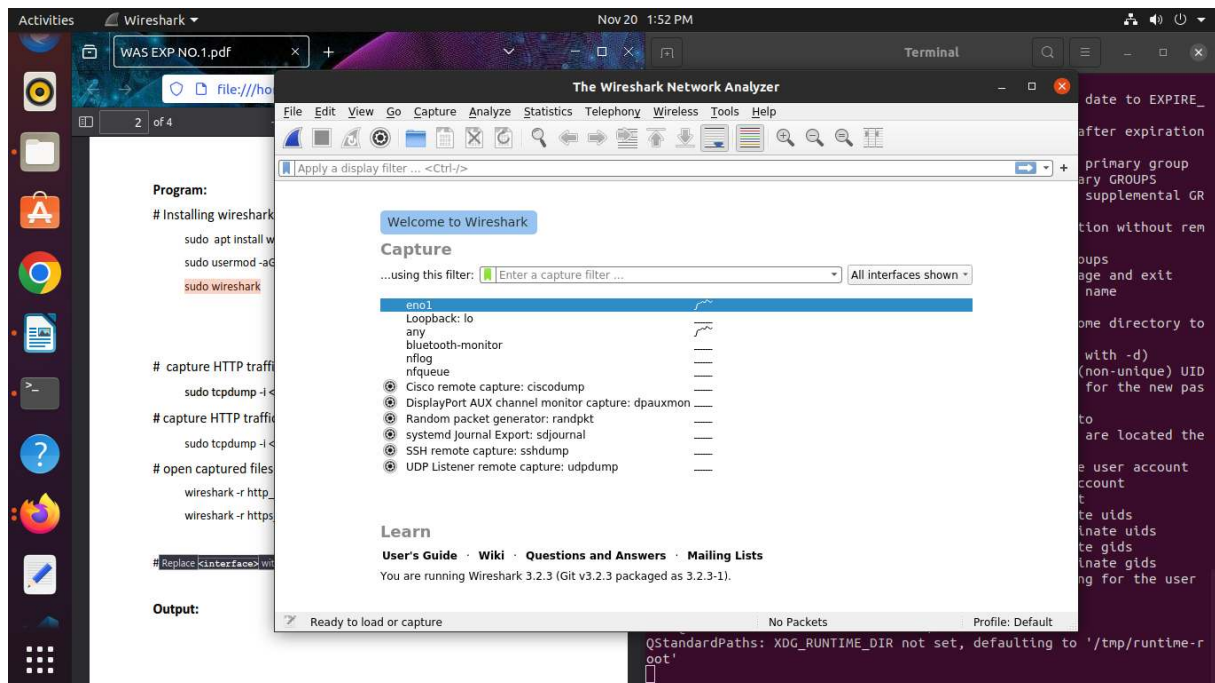
Program:

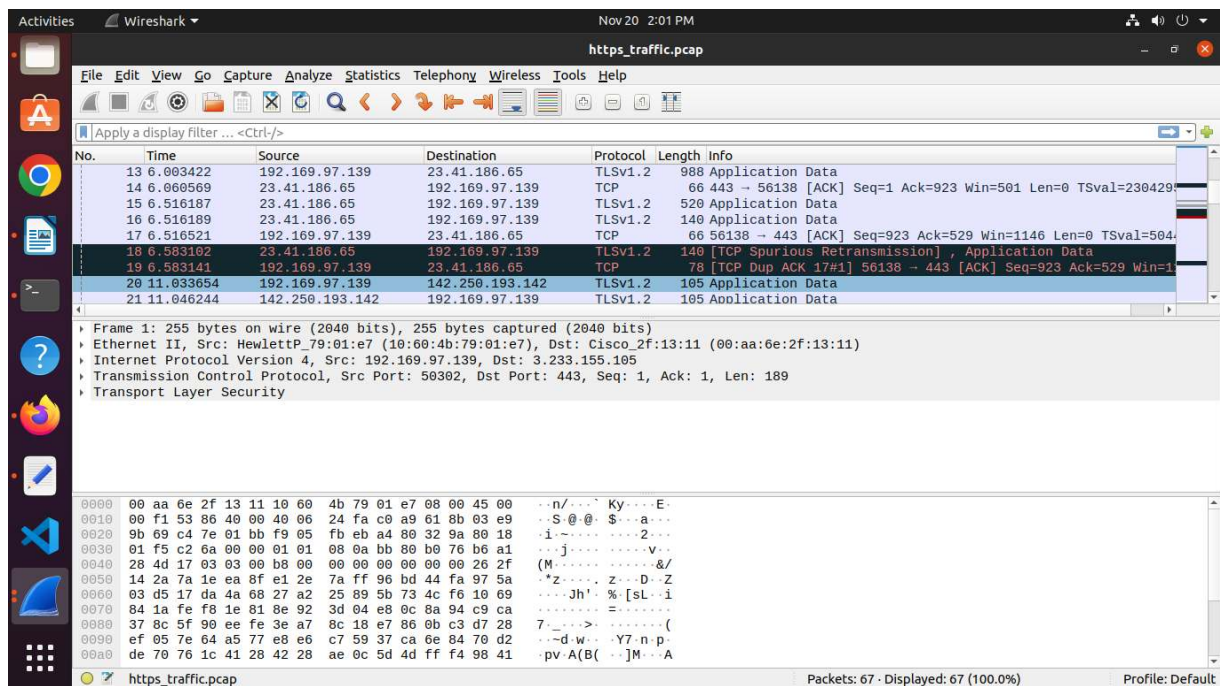
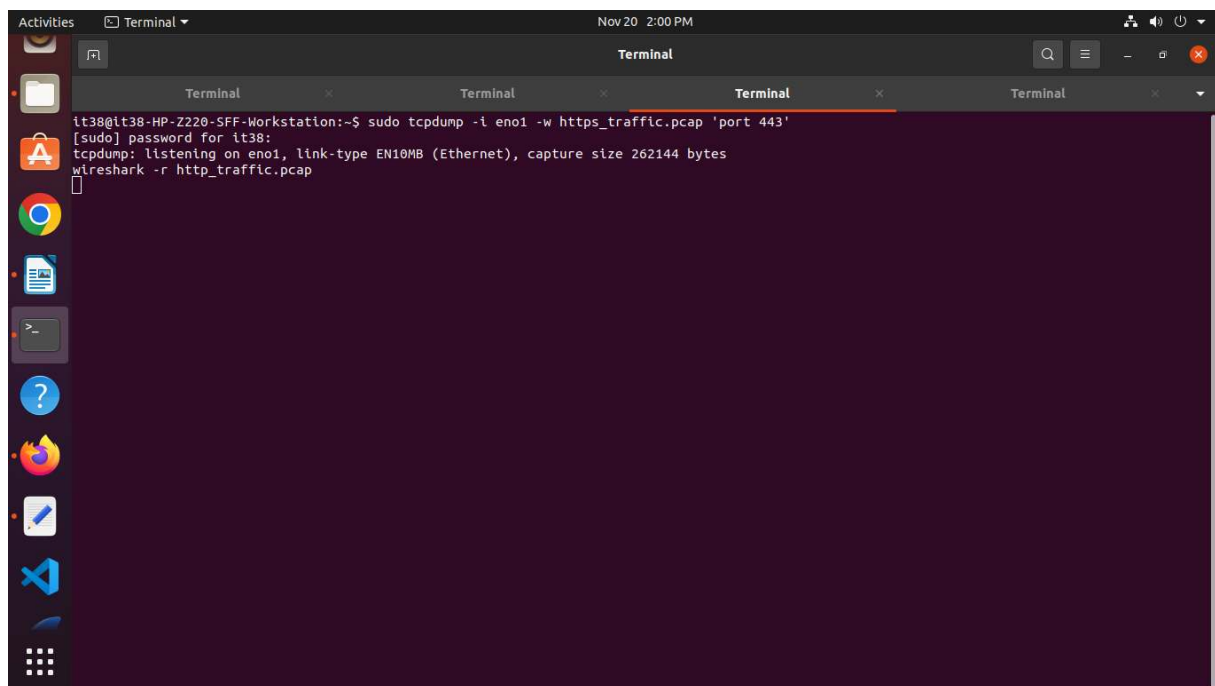
```
# Installing wireshark in Ubuntu:
sudo apt install wireshark
sudo usermod -aG wireshark $USER
sudo wireshark

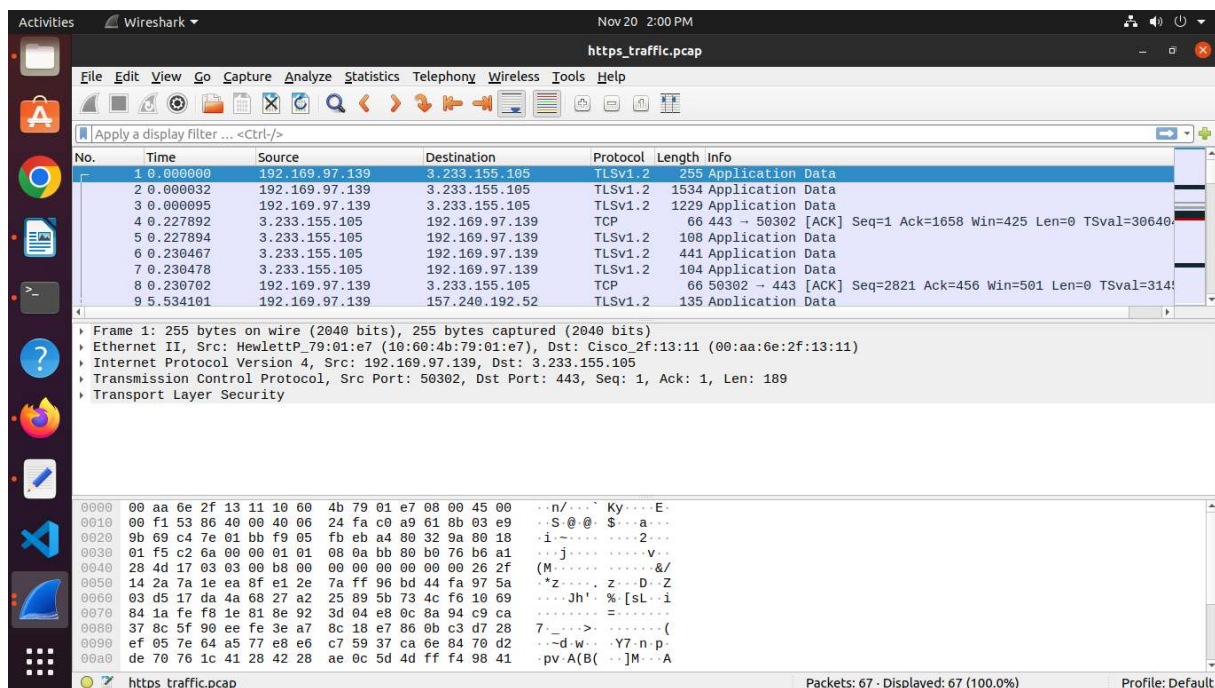
# capture HTTP traffic:
sudo tcpdump -i <interface> -w http_traffic.pcap 'port 80'
# capture HTTPS traffic:
sudo tcpdump -i <interface> -w https_traffic.pcap 'port 443'
# open captured files in wireshark:
wireshark -r http_traffic.pcap
wireshark -r https_traffic.pcap

# Replace <interface> with your network interface.
```

Output:







Result:

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

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:

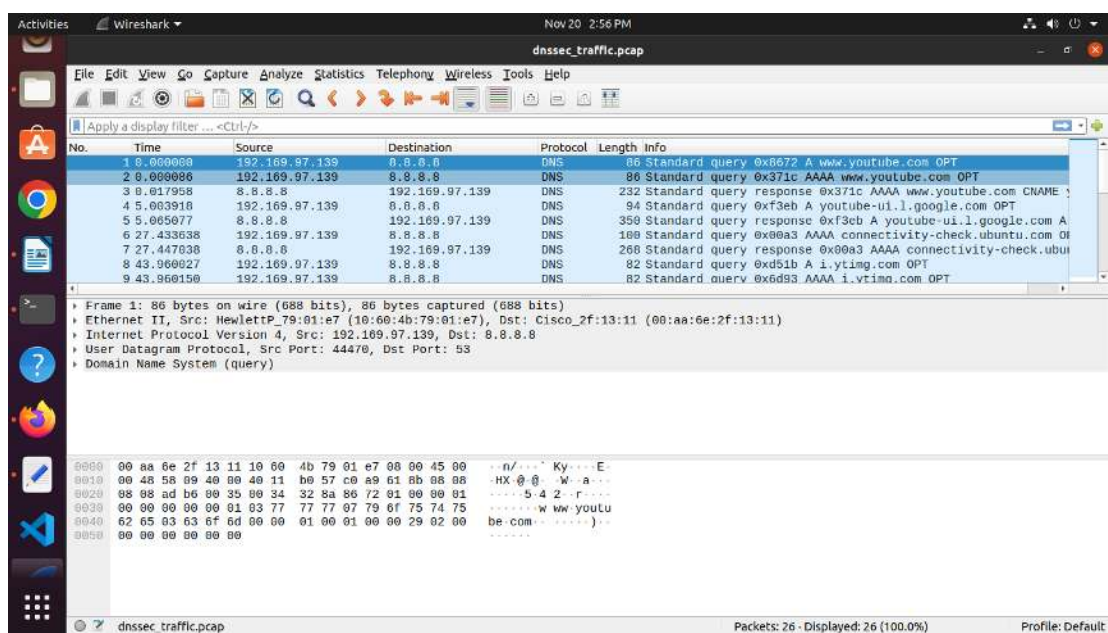
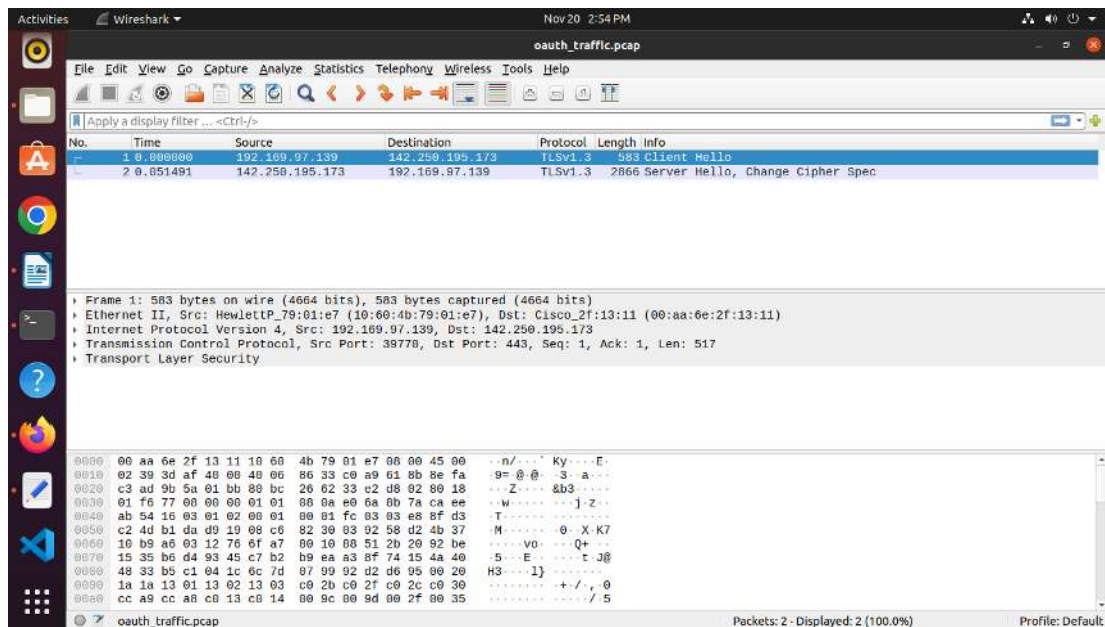
```
sudo tcpdump -i <interface> -w oauth_traffic.pcap 'port 443 and (tcp[((tcp[12] & 0xf0) >> 2):1] = 0x16 or tcp[((tcp[12] & 0xf0) >> 2):1] = 0x80)'
```

#after capturing packets , analyze them using wireshark:

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

Replace **<filename.pcap>** 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.

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:

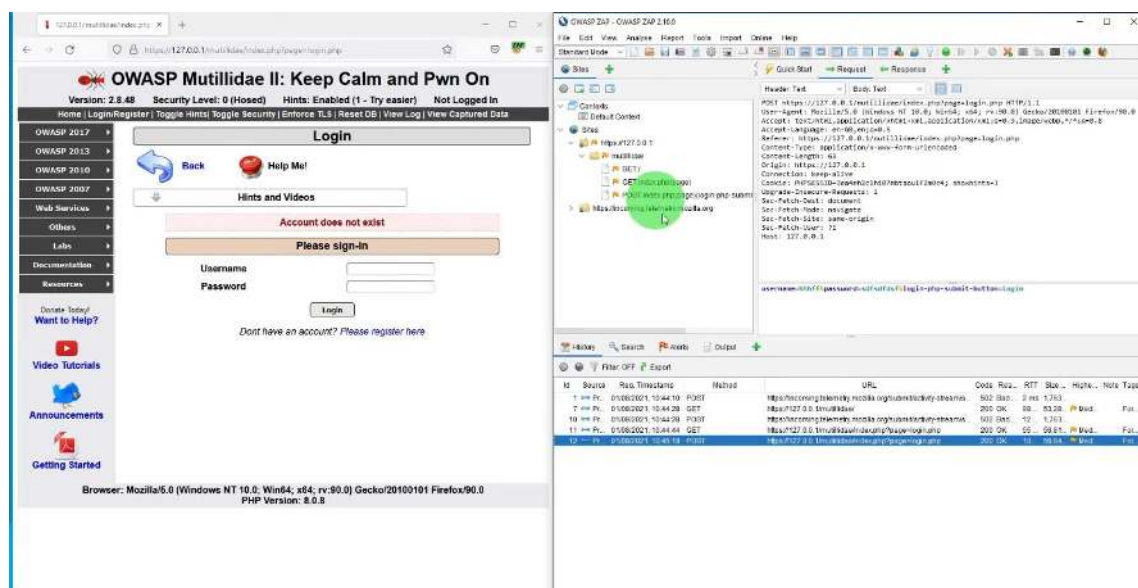
- Go to the "Report" tab.
- Generate a comprehensive report summarizing the identified vulnerabilities and their details.
- Choose the appropriate report format (HTML, PDF, etc.).

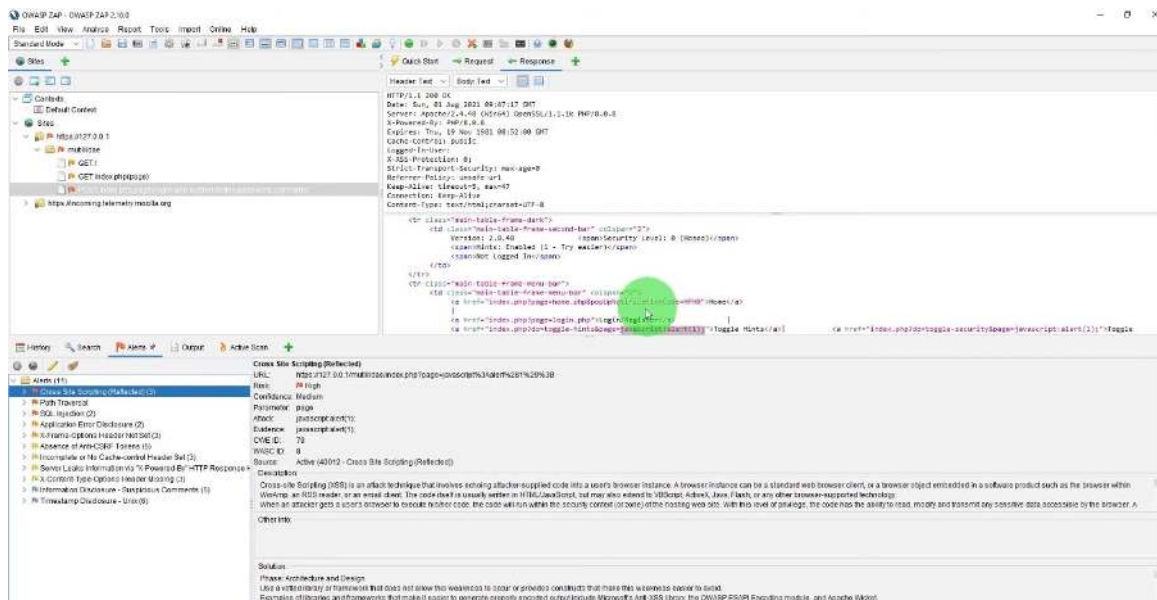
10. Remediation and Re-scan:

- 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.

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

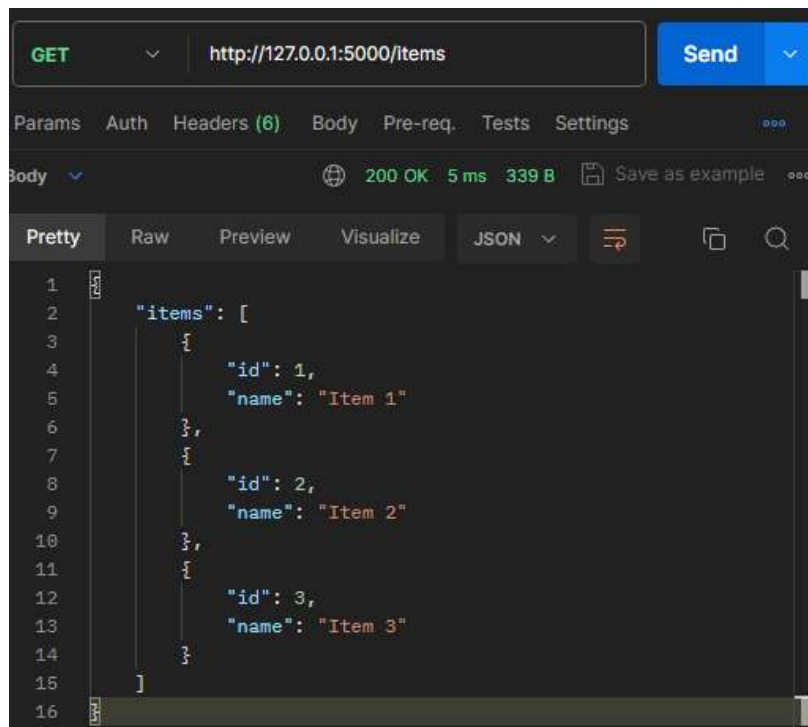
```
>>>python app.py
```

Copy the url produced <http://127.0.0.1:5000>

Step 3: Use Postman to Test Endpoints

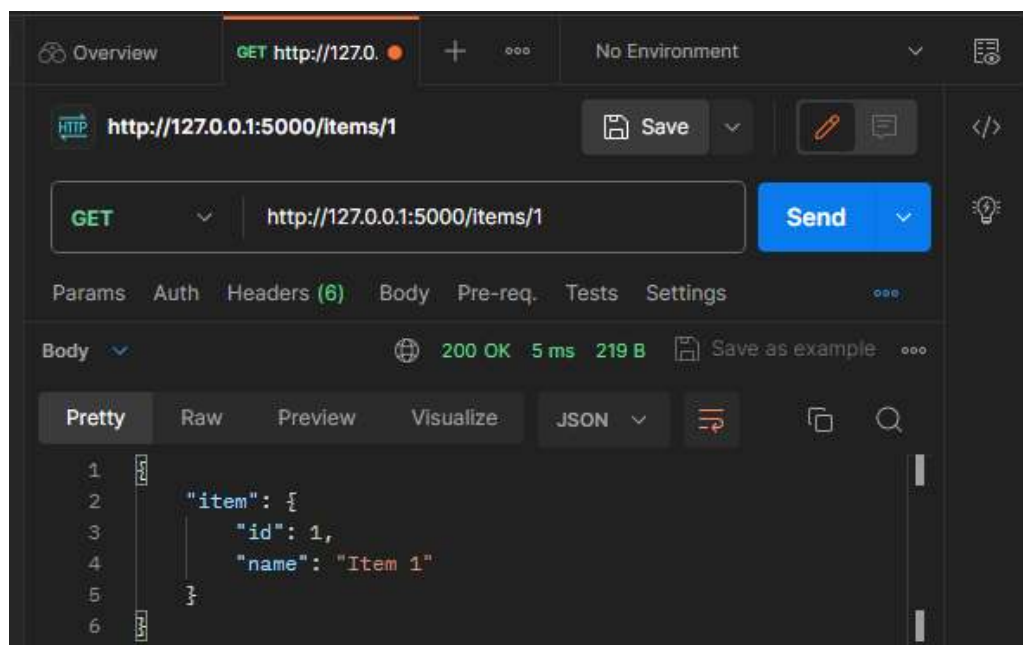
1. GET Request to Retrieve All Items:

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



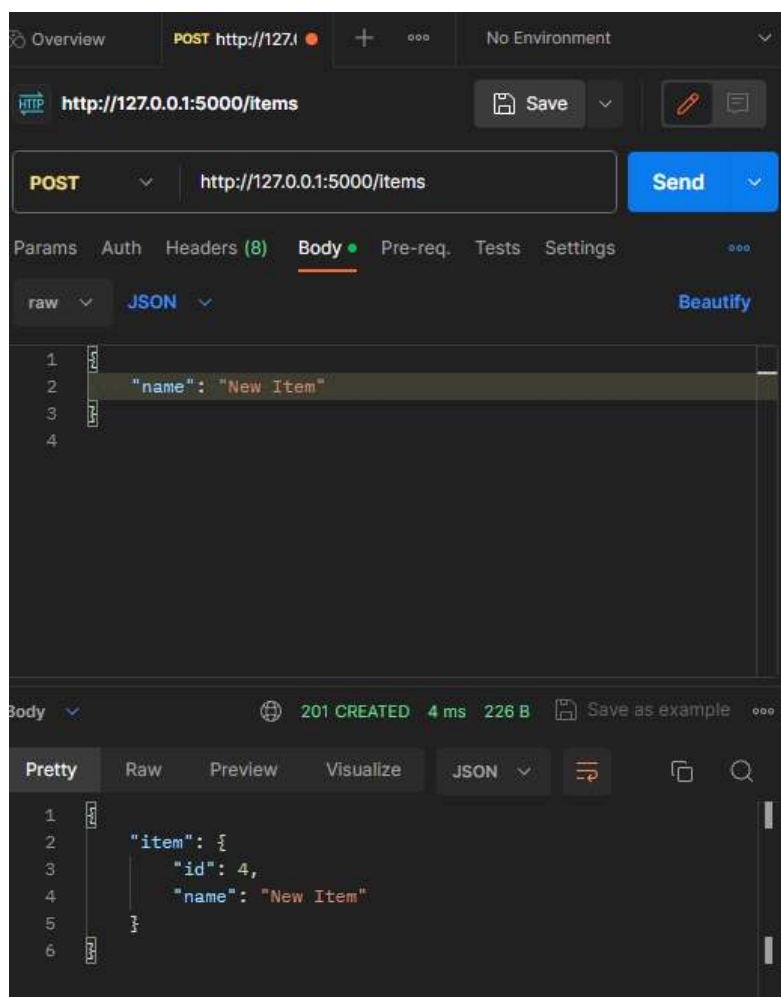
2. GET Request to Retrieve a Specific Item by ID:

- 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:

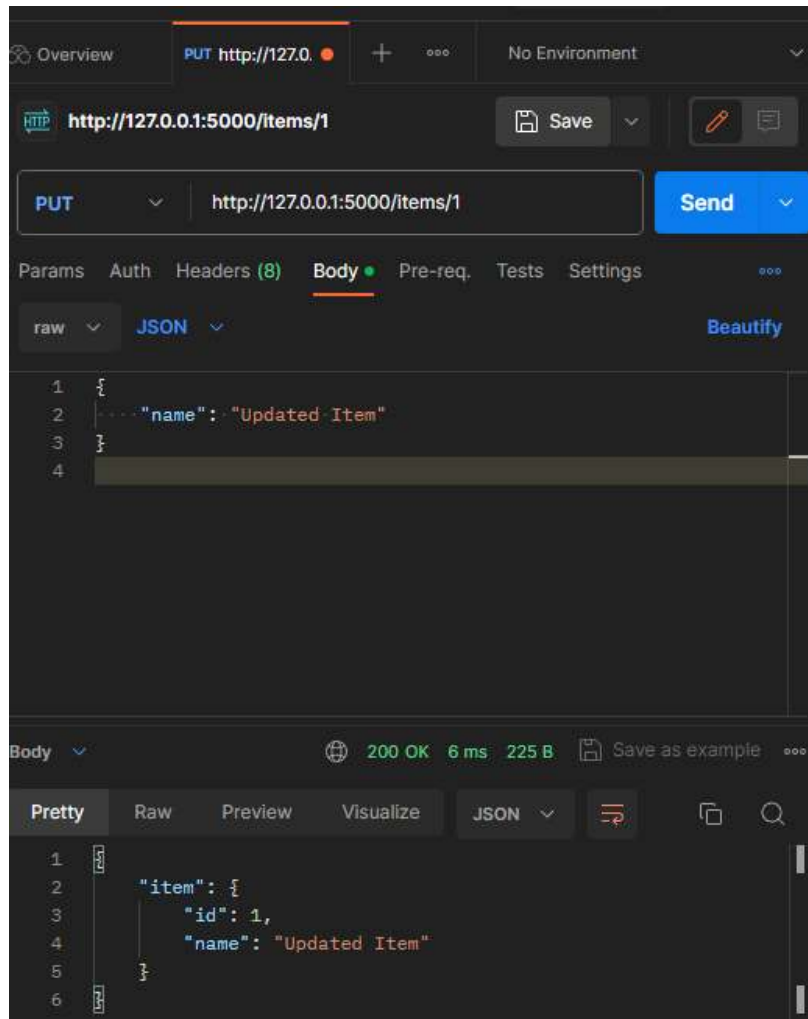
- Set the request type to **POST**.
- Enter the URL: **http://127.0.0.1:5000/items**
- Go to the "Body" tab, select "raw" and choose "JSON (application/json)". Enter the request body
- Click "Send."



4. PUT Request to Update an Existing Item:

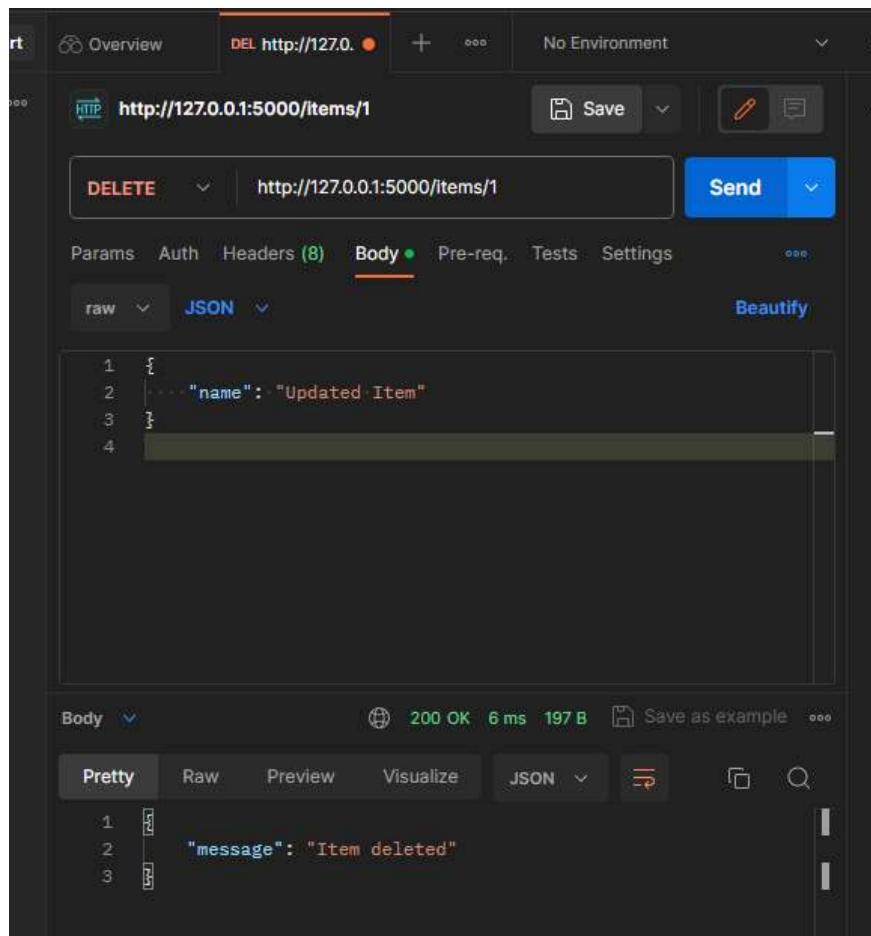
- Set the request type to **PUT**.
- Enter the URL for a specific item ID, for example: **http://127.0.0.1:5000/items/1**

- 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.

Aim:

To Install Burp Suite to do following vulnerabilities:

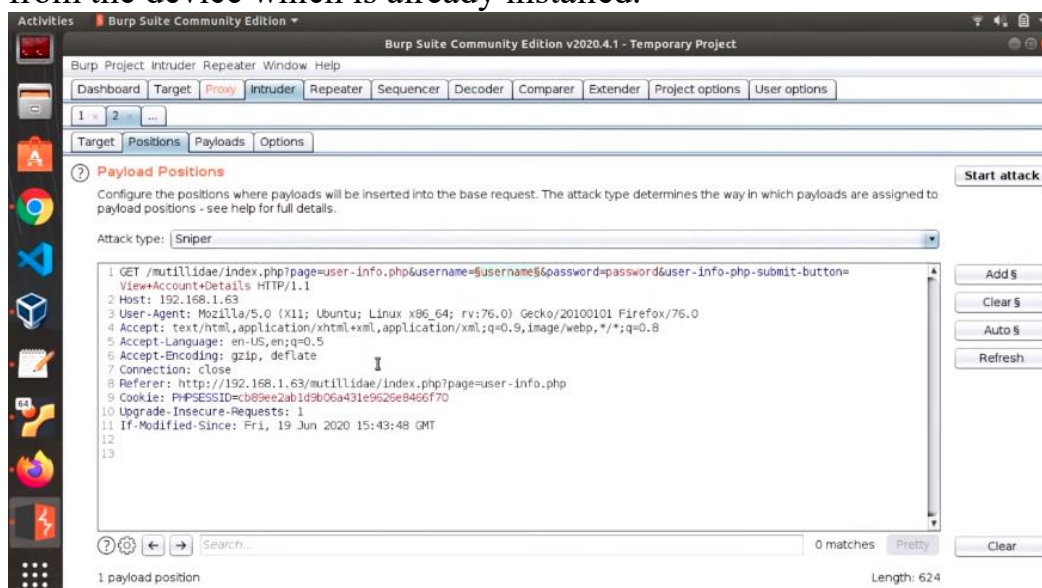
- SQL Injection

Procedure:

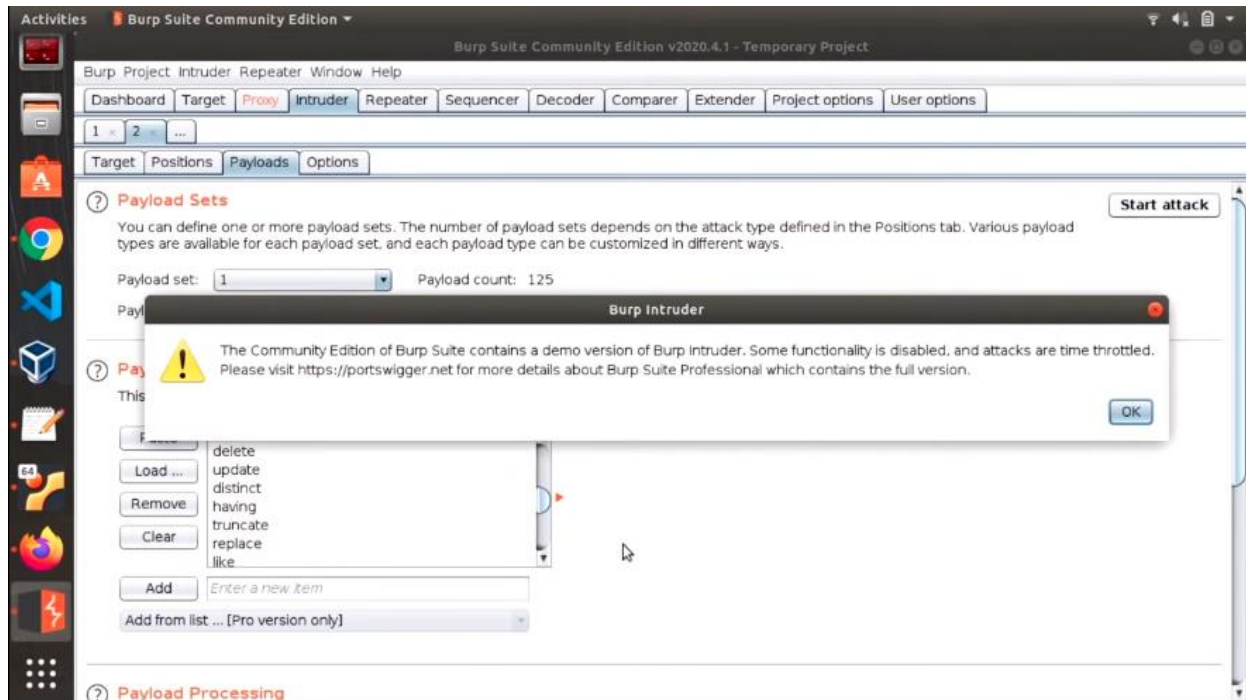
1. Install Burpsuite and connect the burpsuite proxy in browser proxy settings.
2. Turn on the intercept and search for the website which needs to be captured.



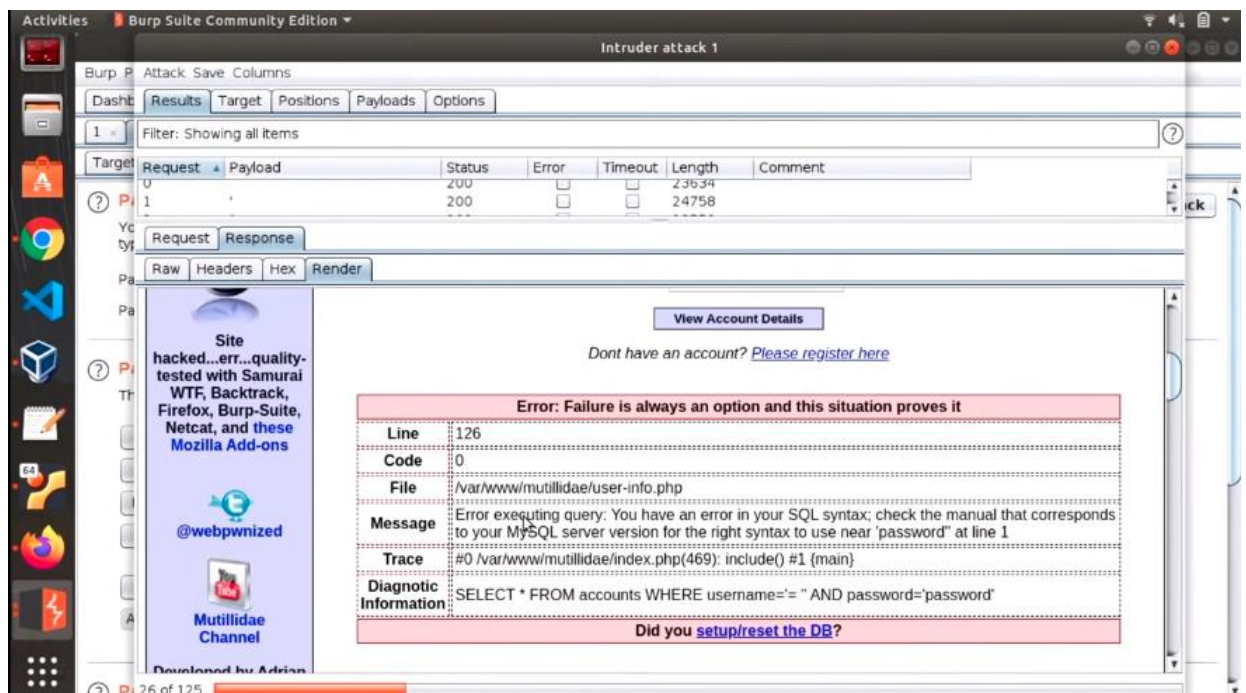
3. Send the intercepted request to the intruder and load the SQL Injection File from the device which is already installed.

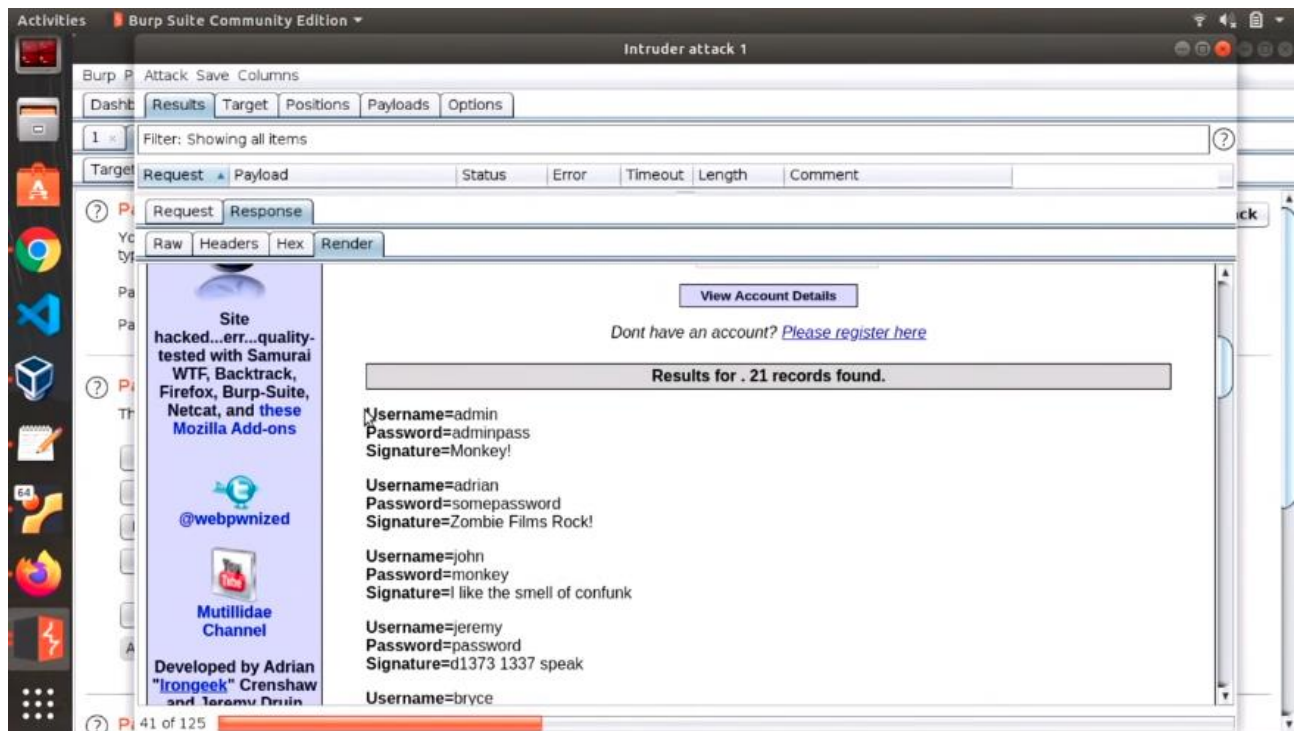


4. Start the attack in the intruder and search for the requests & responses in the render screen for SQL Injection.



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.

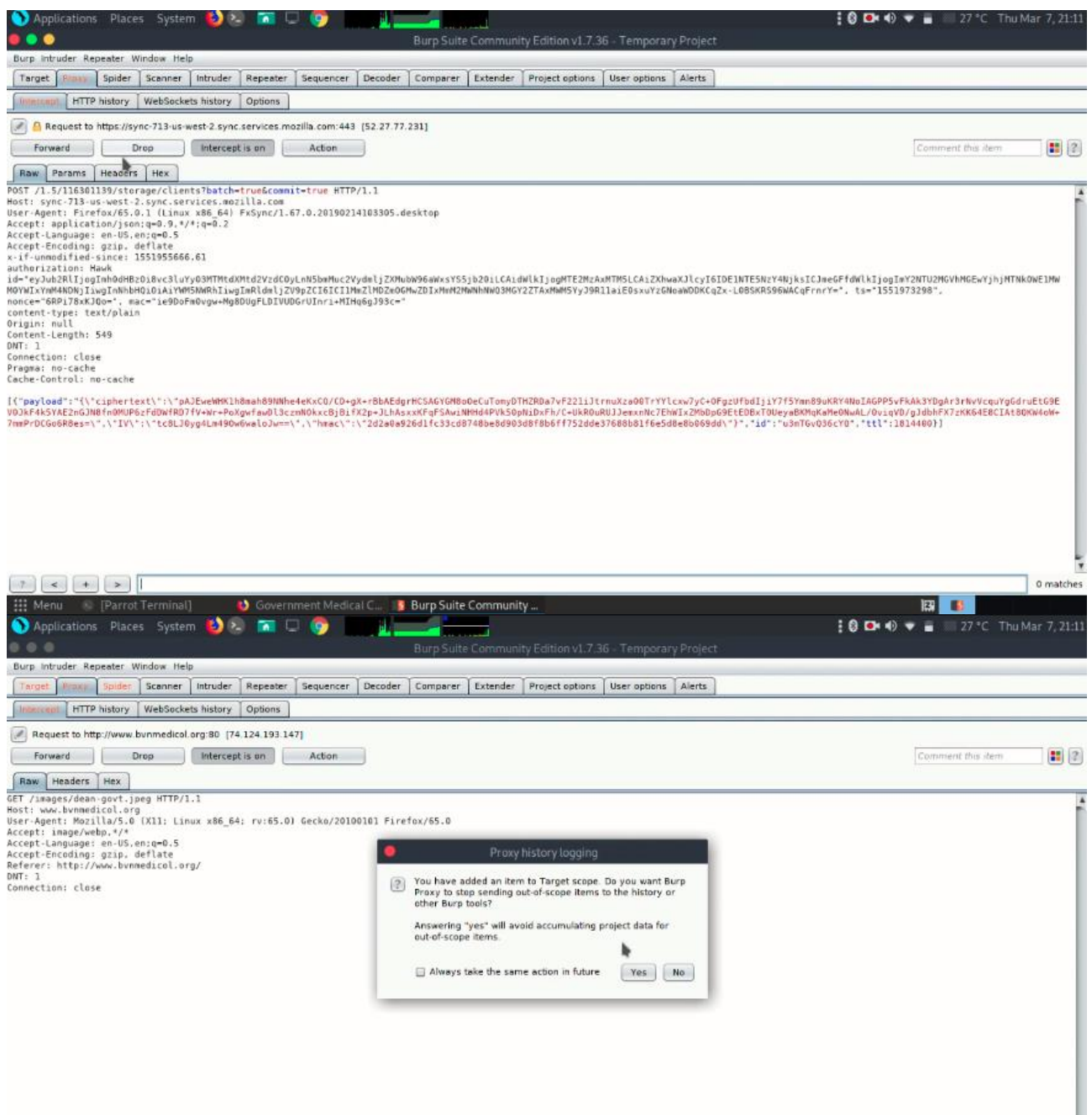
Aim:

To Install Burp Suite to do following vulnerabilities:

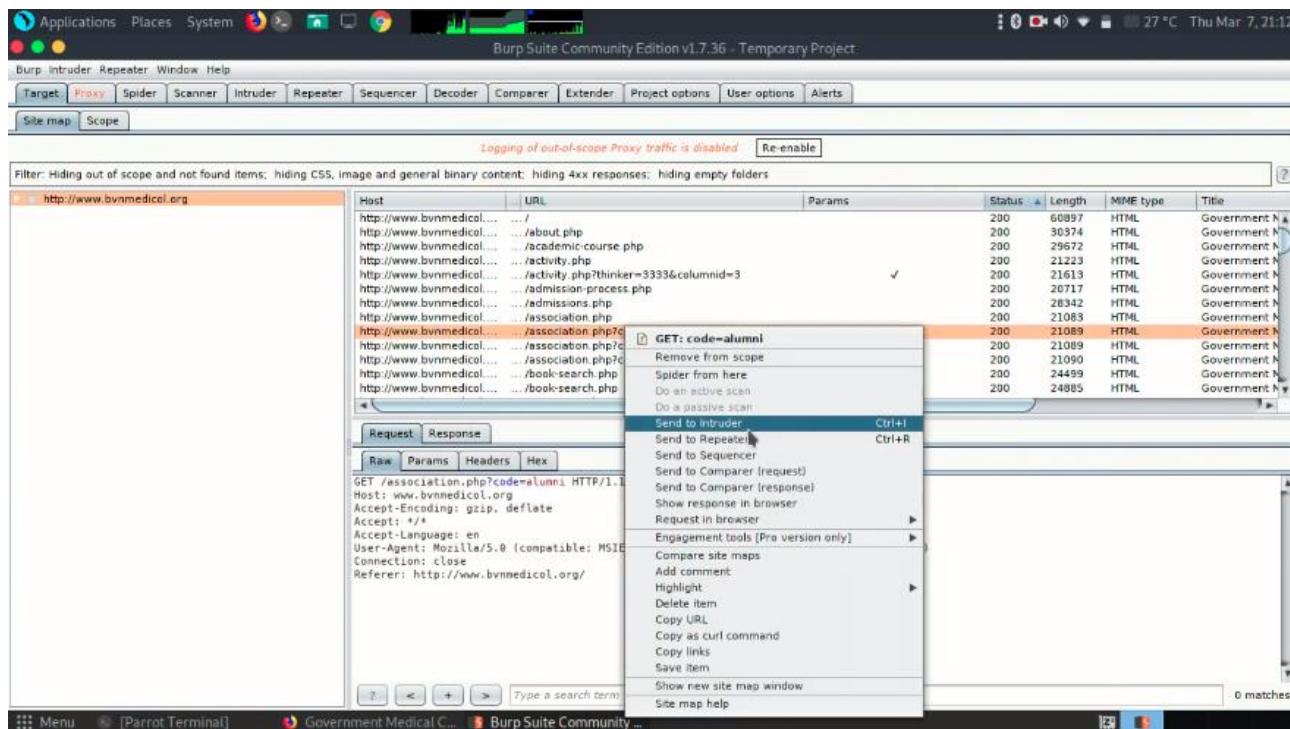
- Cross-Site Scripting (XSS)

Procedure:

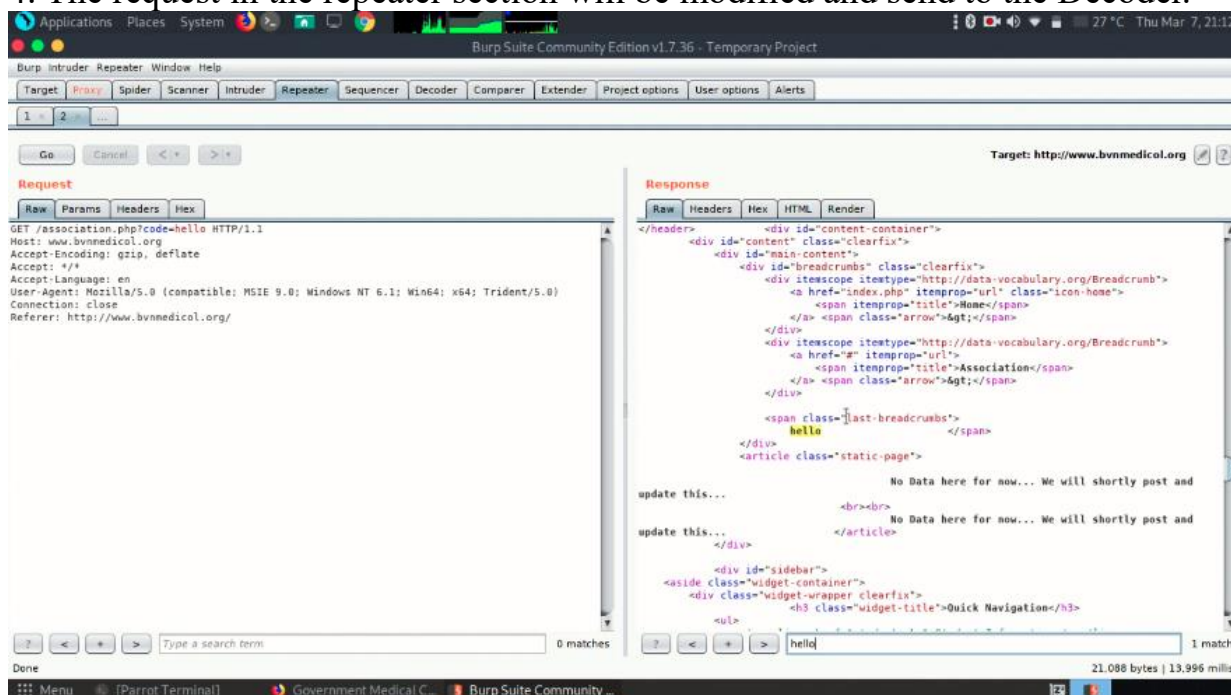
1. Turn on the intercept and search for the website which needs to be captured.
2. Add the captured request to the Target scope.

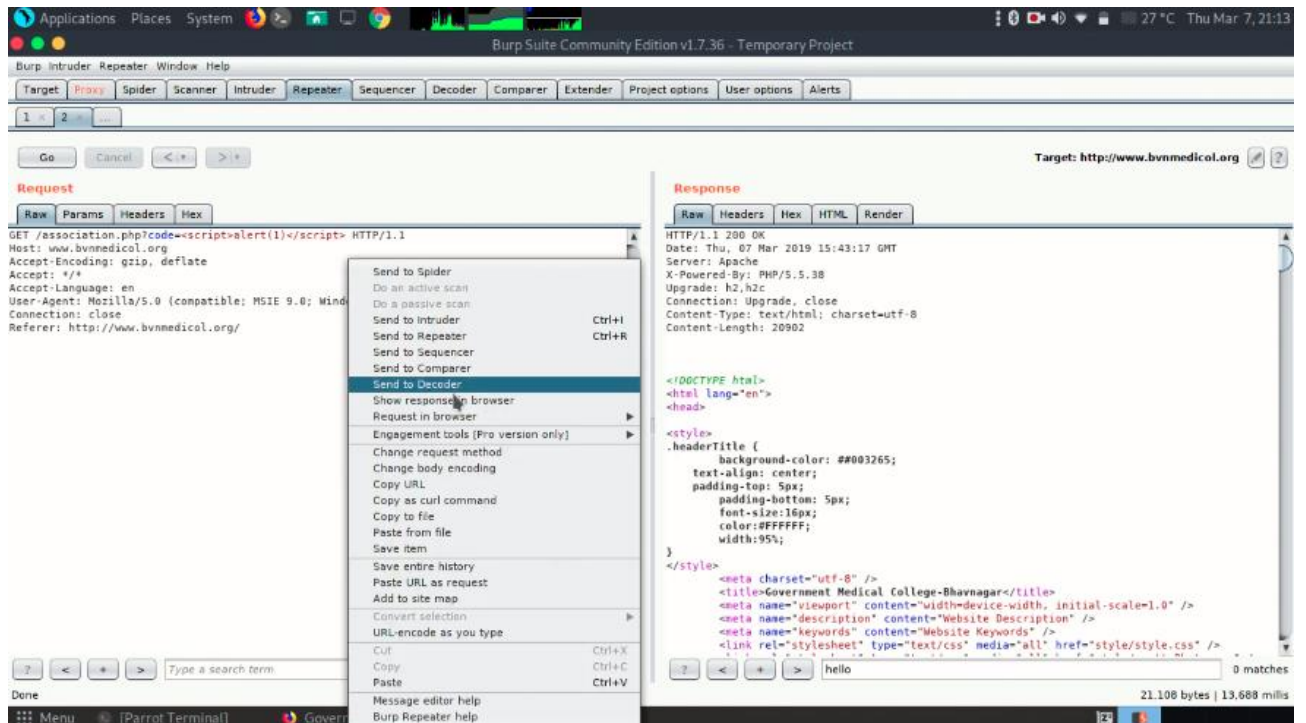


3. Go to Target section and search for the captured request in the item field and send the target item to the repeater.

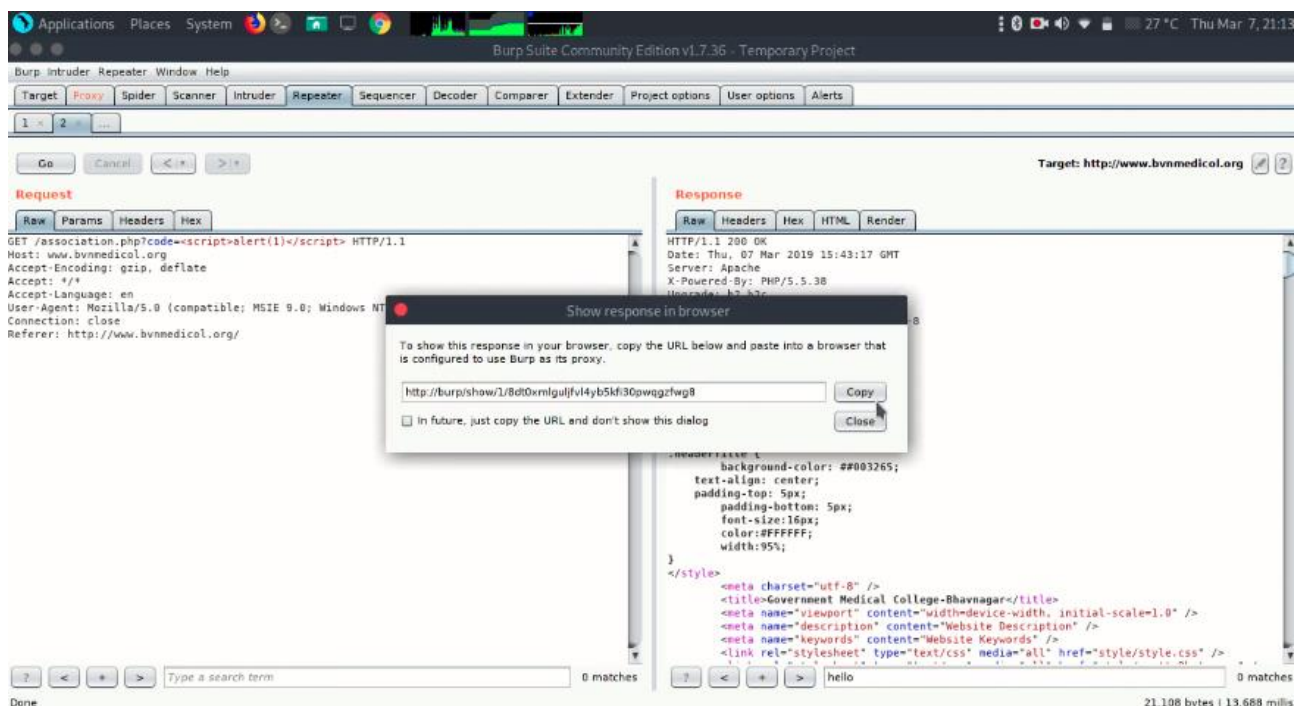


4. The request in the repeater section will be modified and send to the Decoder.

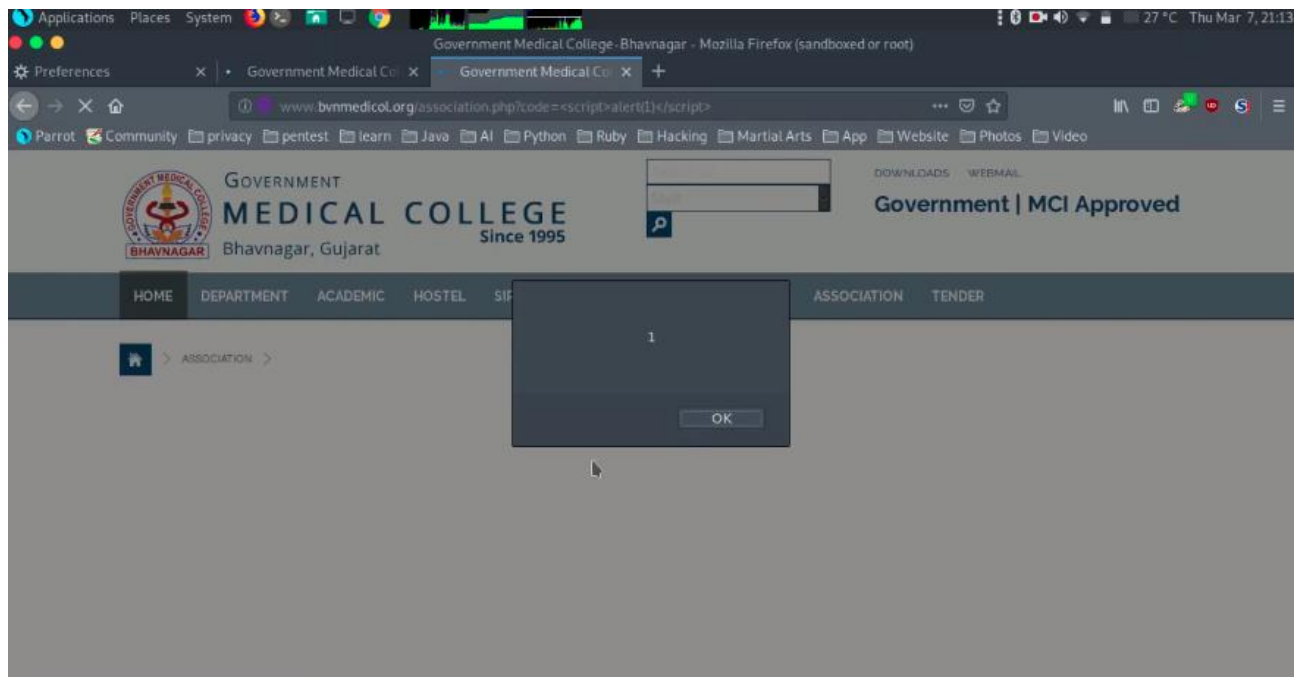




5. Before sending the response to the browser, Copy the URL below and paste into a browser that is configured to use Burp as its proxy.



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.

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 requirements.txt (line 1)) (4.6.0)
Requirement already satisfied: pycrypto in /usr/lib/python3/dist-packages (from -r requirements.txt (line 2)) (2.6.1)
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from -r requirements.txt (line 3)) (2.22.0)
Requirement already satisfied: pyopenssl in /usr/lib/python3/dist-packages (from -r requirements.txt (line 4)) (19.0.0)
Requirement already satisfied: pefile in /usr/lib/python3/dist-packages (from -r requirements.txt (line 5)) (2019.4.18)
Requirement already satisfied: impacket in /usr/lib/python3/dist-packages (from -r requirements.txt (line 6)) (0.9.20)
Requirement already satisfied: qrcode in /usr/lib/python3/dist-packages (from -r requirements.txt (line 8)) (6.1)
Requirement already satisfied: pillow in /usr/lib/python3/dist-packages (from -r requirements.txt (line 9)) (6.2.1)
Requirement already satisfied: pymssql<3.0 in /usr/lib/python3/dist-packages (from -r requirements.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.

```

root@kali: ~/Desktop/SEToolkit/setoolkit
File Actions Edit View Help

[—] The Social-Engineer Toolkit (SET) [—]
[—] Created by: David Kennedy (Re11K) [—]
      Version: 8.0.3
      Codename: 'Maverick'
[—] Follow us on Twitter: @TrustedSec [—]
[—] Follow me on Twitter: @HackingDave [—]
[—] Homepage: https://www.trustedsec.com [—]
Welcome to the Social-Engineer Toolkit (SET).
The one stop shop for all of your SE needs.

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!

Select from the menu:

```


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

```
It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!

Select from the menu:

1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino-Based Attack Vector
7) Wireless Access Point Attack Vector
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) Third Party Modules

99) Return back to the main menu.

set> █
```

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

```
For templates, when a POST is initiated to harvest
credentials, you will need a site for it to redirect.

You can configure this option under:

    /etc/setoolkit/set.config

Edit this file, and change HARVESTER_REDIRECT and
HARVESTER_URL to the sites you want to redirect to
after it is posted. If you do not set these, then
it will not redirect properly. This only goes for
templates.

-----

1. Java Required
2. Google
3. Twitter

set:webattack> Select a template:█
```


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

```
root@kali: ~/Desktop/SEToolkit/settoolkit
File Actions Edit View Help
after it is posted. If you do not set these, then
it will not redirect properly. This only goes for
templates.

-----

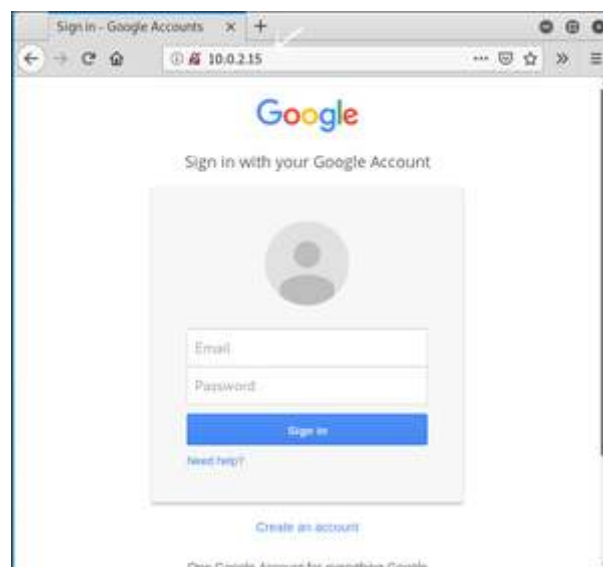
1. Java Required
2. Google
3. Twitter

set:webattack> Select a template:2

[*] Cloning the website: http://www.google.com
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. R
egardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

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.