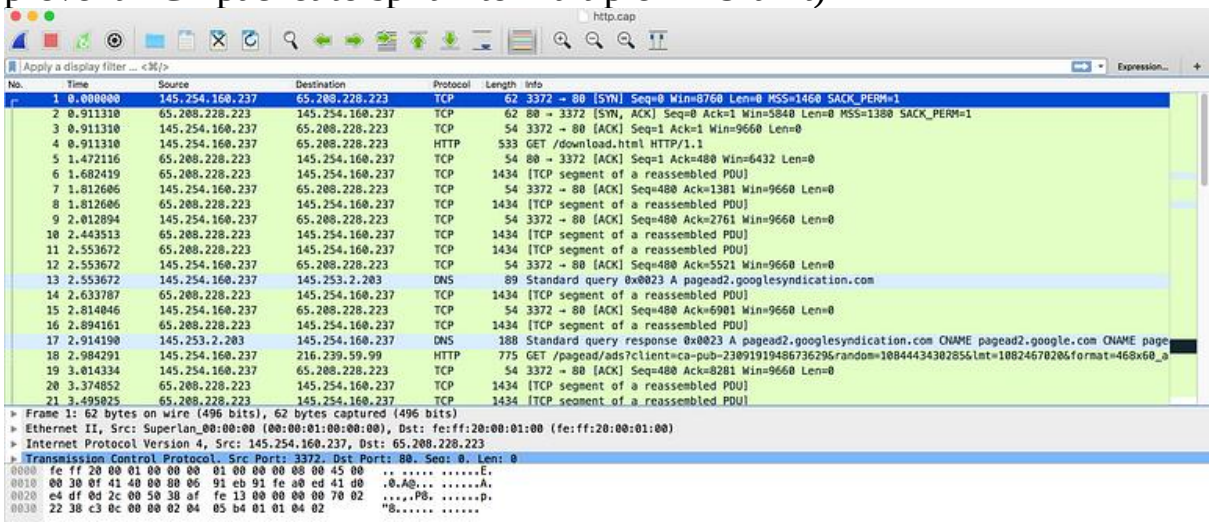
Installation of Wireshark Software

- Open the web browser.
- Search for '**Download Wireshark**.'
- Select the Windows installer according to your system configuration, either 32-bt or 64-bit. Save the program and close the browser.
- Now, open the software, and follow the install instruction by accepting the license.
- The Wireshark is ready for use.

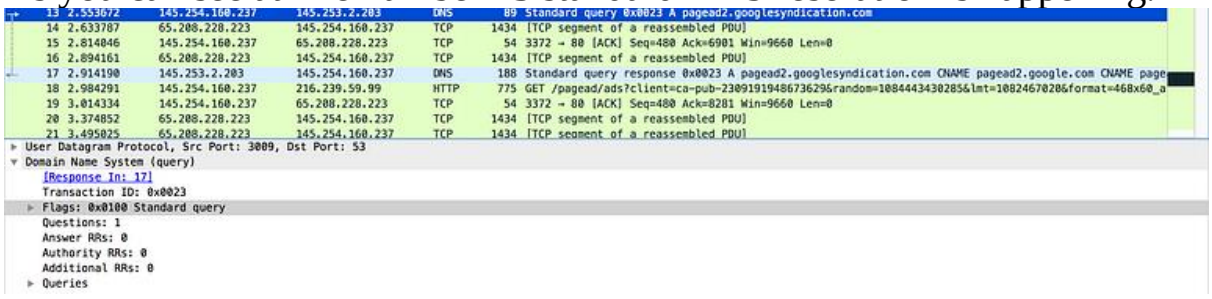
HTTP capture

Before start analyzing any packet, please turn off "**Allow subdissector to reassemble TCP streams**"(Preference → Protocol → TCP)(This will prevent TCP packet to split into multiple PDU unit)

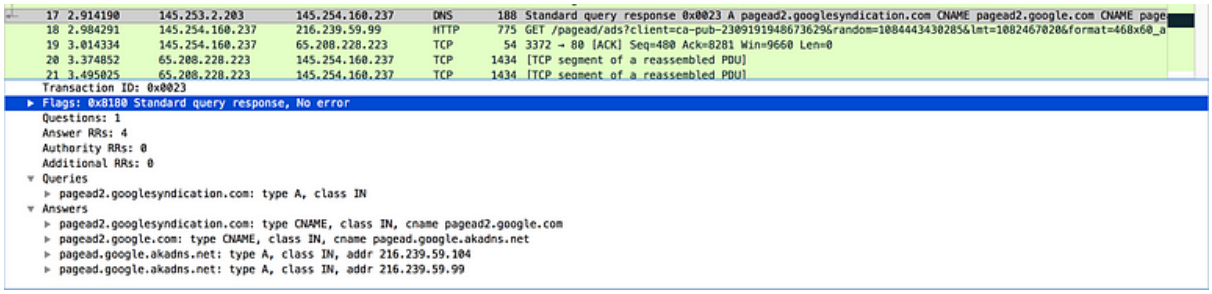


As you can see I am using HTTP so that the encryption will not be hidden behind TLS.

As you can see at line number 13 standard DNS resolution is happening.



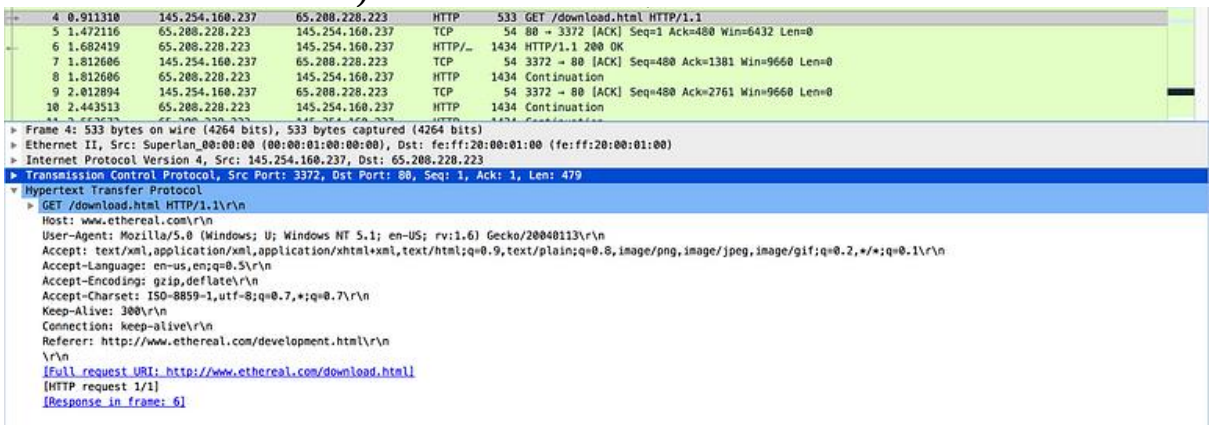
In line number 17 you see the response we are getting back with full DNS resolution



Now if you look at Packet number 4 i.e is get request,HTTP primarily used two command

1: **GET:** To retrieve information

2: **POST:** To send information(For eg: when we submit some form we fill some data i.e is POST)



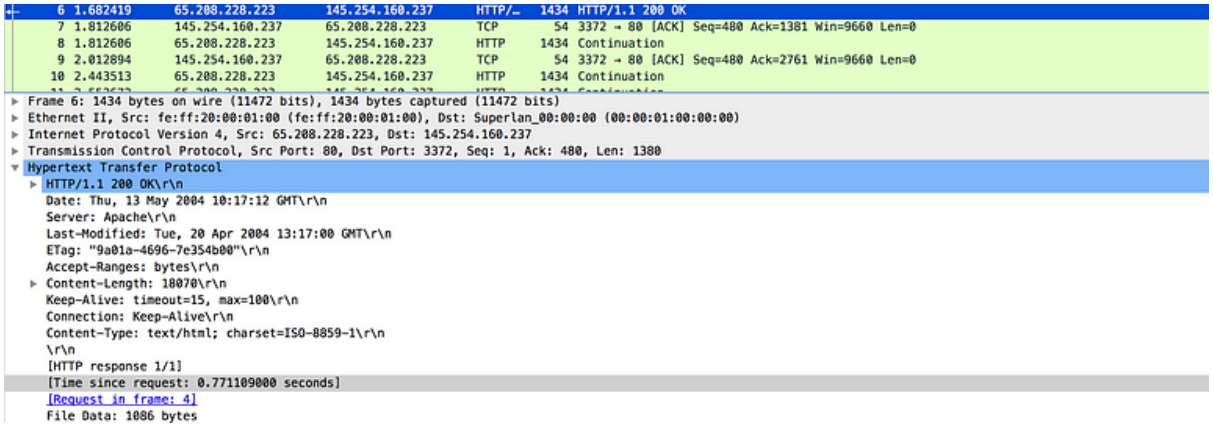
Here I am trying to get download.html via HTTP protocol 1.1(The new version of protocol is now available i.e 2.0)

Then at line number 5 we see the acknowledgment as well as line number 6 server was able to found that page and send HTTP status code 200.

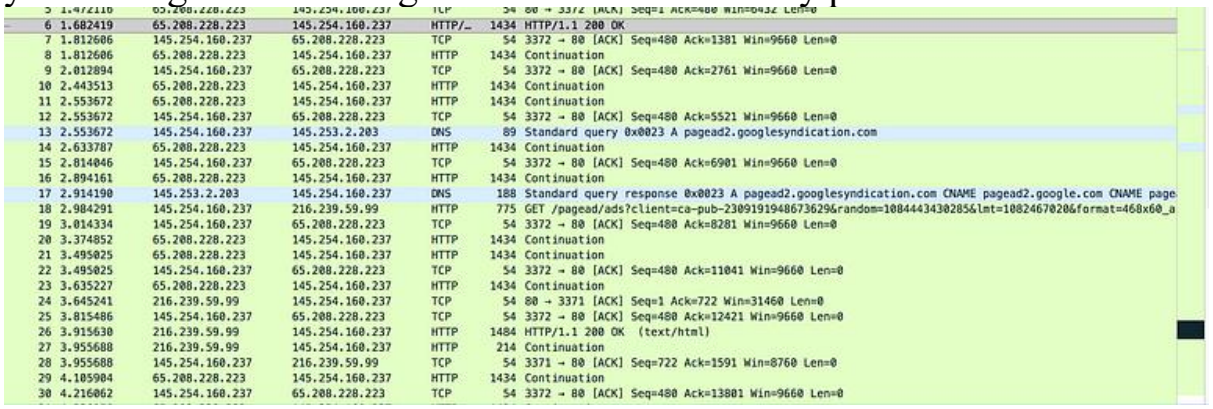
If you want more info about **HTTP status code**

HTTP Status Codes

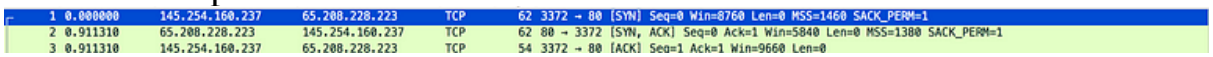
You will see some more info like for packet 6, like Server type is Apache, content type is HTML, how long is the content length is,



Then you will see bunch of continuation that is due to TCP window where you don't get acknowledgement for each and every packet



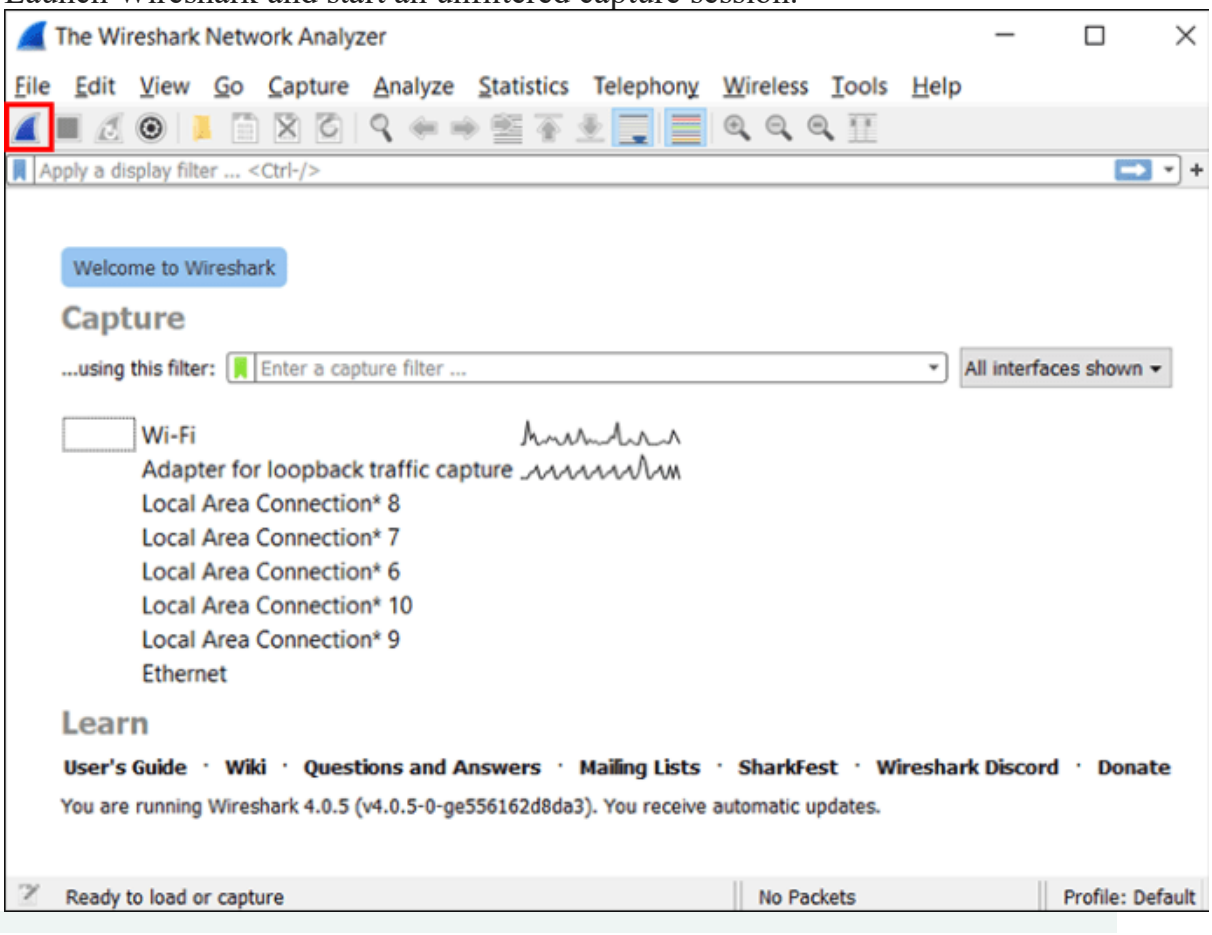
and at that top some usual TCP handshake



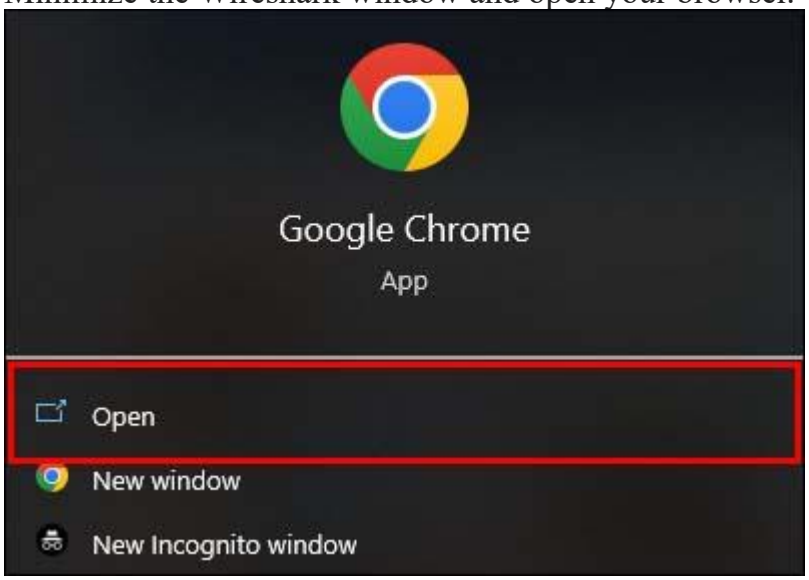
capture HTTPS in Wireshark

Capture and Decrypt Session Keys

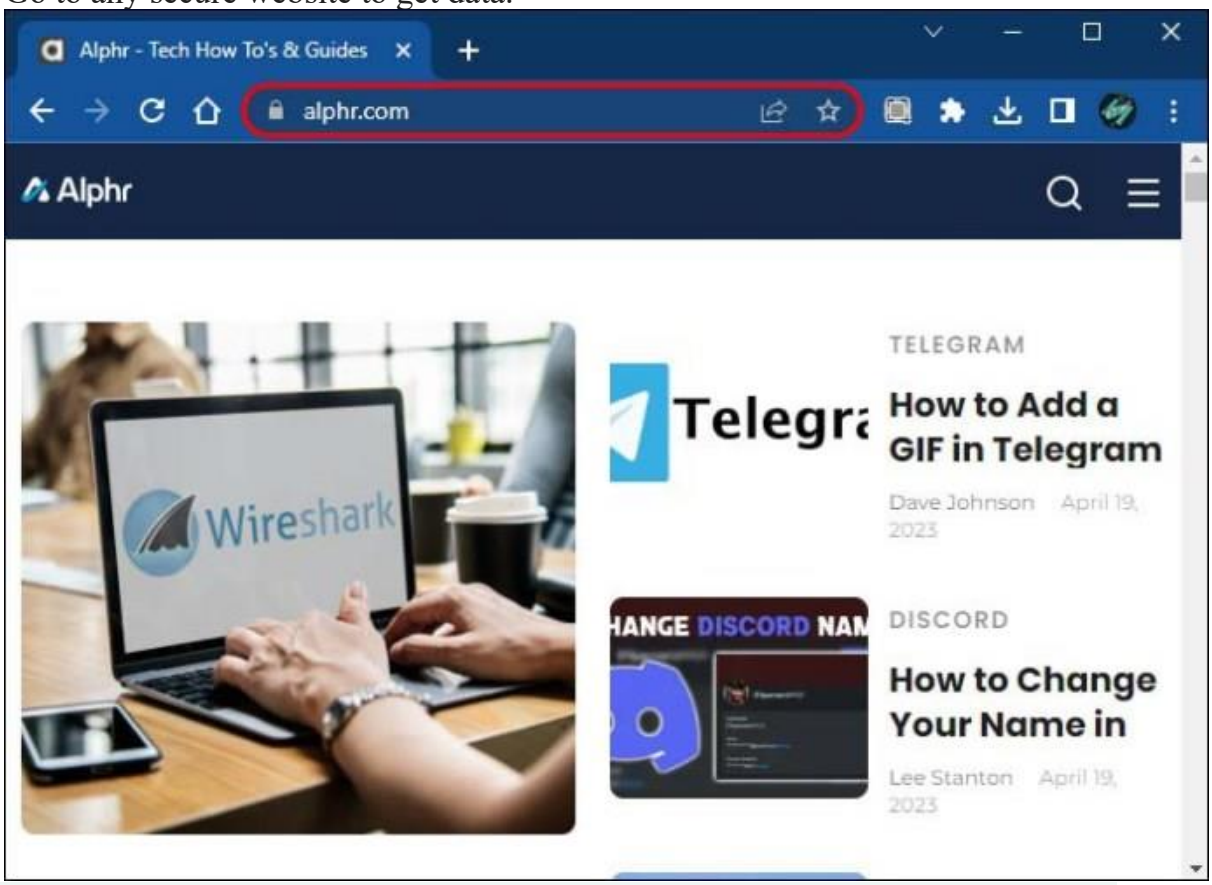
1. Launch Wireshark and start an unfiltered capture session.



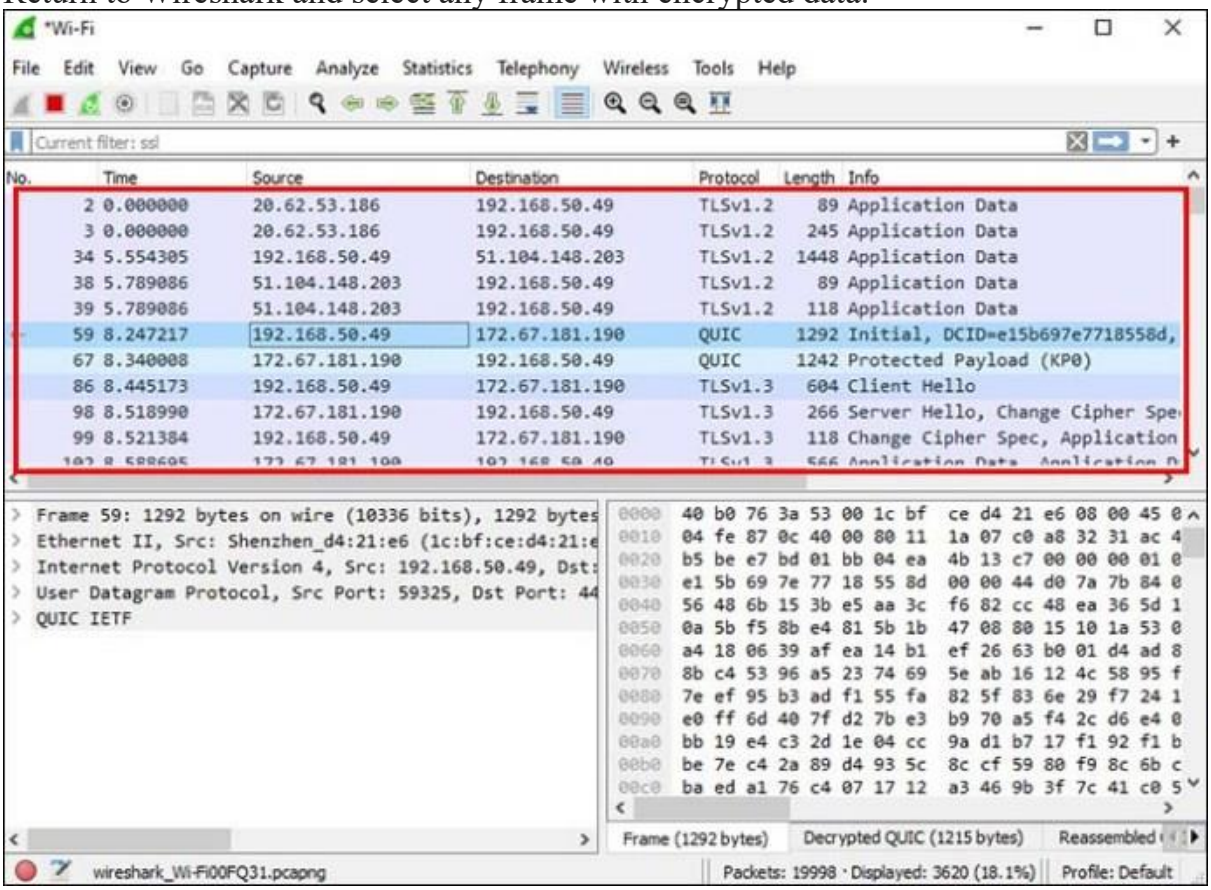
2. Minimize the Wireshark window and open your browser.



3. Go to any secure website to get data.



4. Return to Wireshark and select any frame with encrypted data.



5. Find “Packet byte view” and look at “Decrypted SSL” data. HTML should now be visible.

Conclusion

HTTPS provides better security than HTTP due to the following reasons:

- **Encryption**: The use of SSL/TLS encryption in HTTPS ensures that the data being transmitted between the client and the server is secure and cannot be intercepted by an attacker.
- **Authentication**: HTTPS uses digital certificates to authenticate the identity of the website, making it harder for attackers to impersonate a website and carry out a man-in-the-middle attack.
- **Integrity**: HTTPS provides data integrity which ensures that the data being transmitted between the client and the server has not been modified or tampered with.
- **SEO**: HTTPS is now a ranking signal for search engines, and having an HTTPS site can improve your search engine ranking.

Analyzing the security mechanisms embedded in different protocols using Wireshark involves examining network traffic captures to understand how various security features are implemented and whether they are effective. Below are some common security mechanisms embedded in different protocols that you can analyze using Wireshark:

1. Transport Layer Security (TLS/SSL):

- TLS/SSL is used to encrypt and secure data transmission over the network.

- In Wireshark, you can identify TLS/SSL traffic by looking at the protocol field, which should indicate TLS or SSL.

- You can analyze the TLS handshake process, certificate exchange, and the encryption ciphers being used.

2. Secure Shell (SSH):

- SSH is a secure protocol for remote access and data exchange.

- Wireshark can capture SSH traffic, and you can analyze the key exchange, authentication methods, and encryption algorithms being used.

3. IPsec (Internet Protocol Security):

- IPsec is used to secure IP communication through encryption and authentication.

- Wireshark can capture and analyze IPsec packets to understand the negotiation of security associations (SAs), encryption algorithms, and authentication methods.

4. HTTP Secure (HTTPS):

- HTTPS is a secure version of HTTP that uses TLS/SSL for encryption.

- Wireshark can capture HTTPS traffic and allow you to examine the encrypted content after decryption by examining the TLS layer.

5. Virtual Private Network (VPN) Protocols:

- VPN protocols like OpenVPN, L2TP, or PPTP provide secure tunnels for data transmission.

- Wireshark can capture VPN traffic, and you can analyze the encapsulation, authentication, and encryption mechanisms used within the VPN tunnel.

6. Secure Sockets Layer (SSL) VPN:

- SSL VPNs create secure connections for remote access.

- Wireshark can capture SSL VPN traffic and allow you to analyze the SSL handshake, certificate exchange, and data transmission.

7. Kerberos:

- Kerberos is a network authentication protocol.

- You can capture and analyze Kerberos traffic in Wireshark to observe the ticket granting process, authentication, and encryption.

8. SNMPv3 (Simple Network Management Protocol version 3):

- SNMPv3 includes security features such as authentication and encryption.

- Wireshark can help you understand how SNMPv3 is used to secure network management.

9. DNSSEC (Domain Name System Security Extensions):

- DNSSEC adds security to the DNS by signing DNS records.

- Wireshark can be used to capture DNSSEC-signed DNS queries and responses to validate the digital signatures.

10. OAuth and OAuth2:

- OAuth is used for delegated authorization, often in web and API interactions.

- You can capture OAuth flows and analyze the token exchange and authentication mechanisms.

When analyzing security mechanisms using Wireshark, it's essential to focus on packet captures related to the specific protocol or technology you're interested in. Look for indicators of encryption, authentication, key exchanges, and other security features to assess the effectiveness and security of the implementation. Additionally, pay attention to any anomalies or potential vulnerabilities that may be present in the captured traffic.

1. Installing ZAP

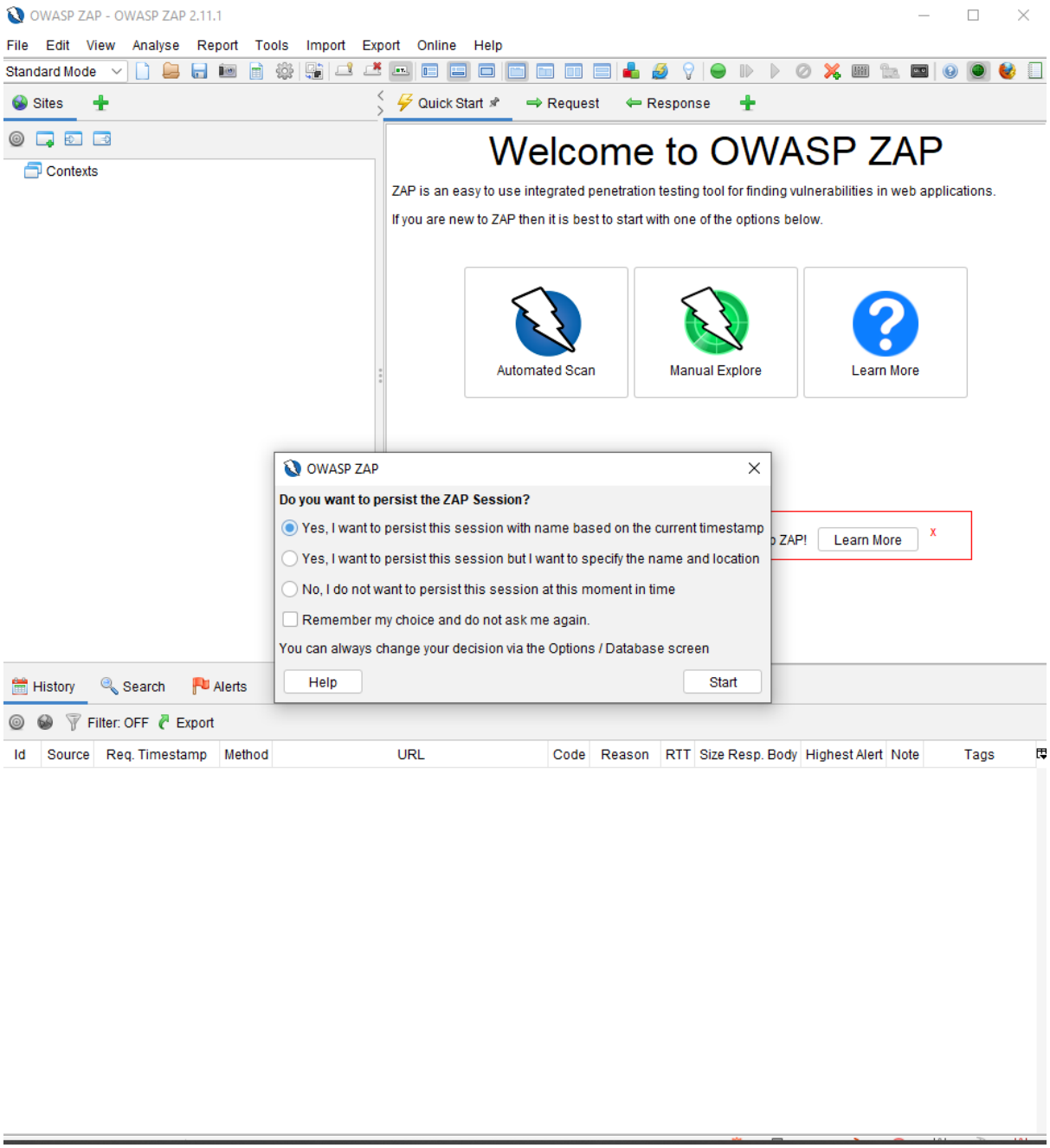
You can download the latest version from the OWASP ZAP website for your operating system to install ZAP or reference the ZAP_docs for a more detailed installation guide.

Once completed, follow the prompts to install OWASP ZAP on your machine.

2. Persisting a session

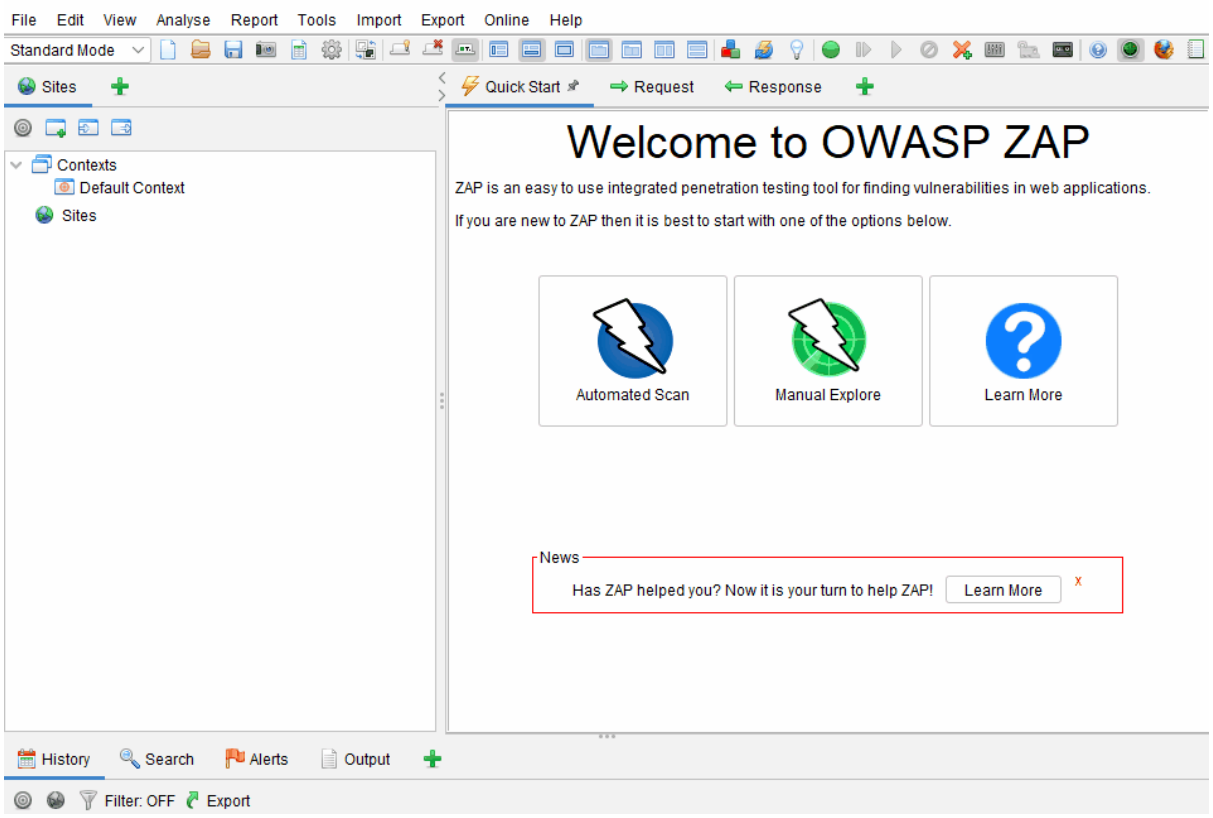
Persisting a session in OWASP ZAP means that the session will be saved and can be reopened at a later time. This is useful if you want to continue testing a website or application at a later time.

Once you've started OWASP ZAP, you will see a screen that looks like this:



The prompt gives two options to persist in the session. **You can use the default to name the session based on the current timestamp or set your name and location.**

Alternatively, you can persist a session by going to ‘File’ and choosing ‘Persist Session…’. Give your session a name and click on the ‘Save’ button.



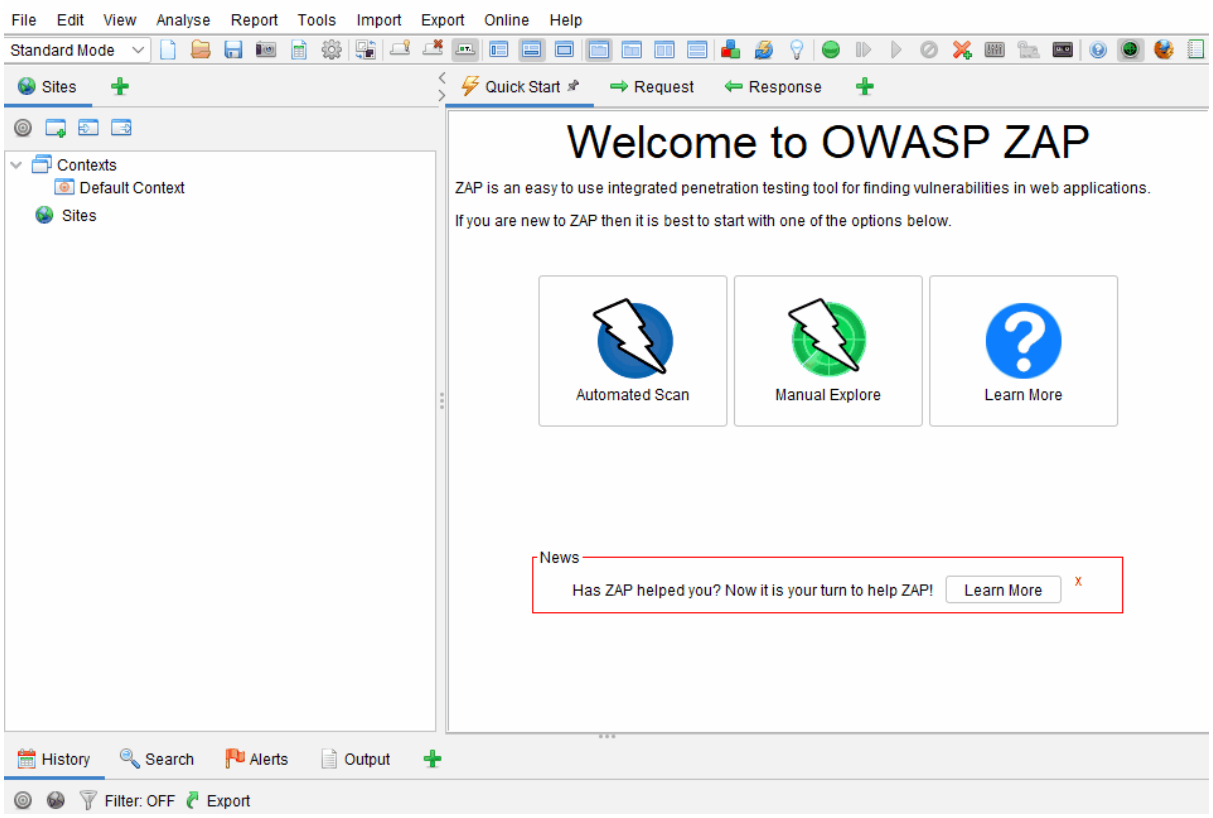
3. Running an automated scan

Running an automated scan in OWASP ZAP is a way to check for common security vulnerabilities in web applications. **This is done by sending requests to the application and analyzing the responses for signs of common vulnerabilities.** It can help to find security issues early in the development process before they are exploited.

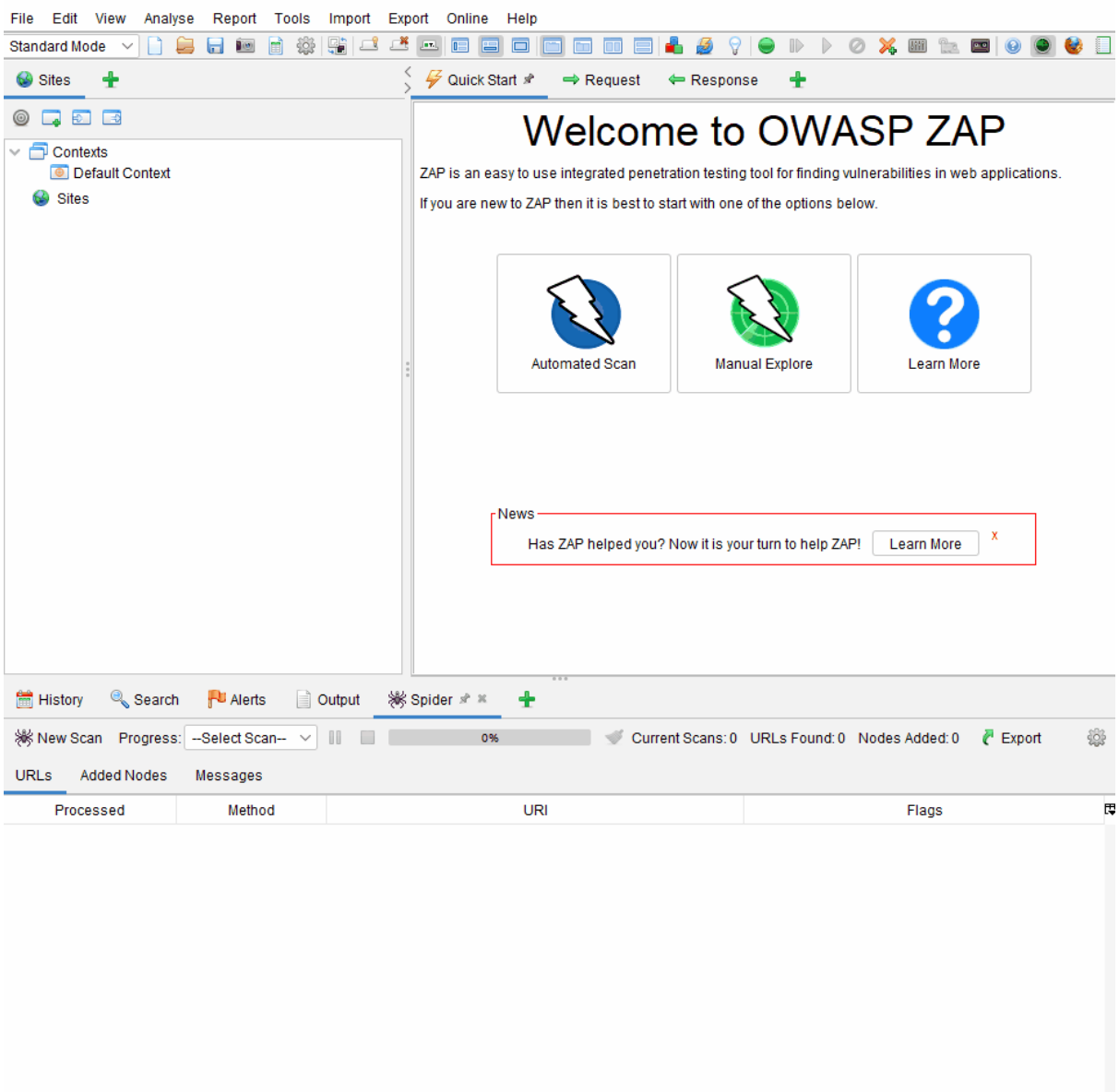
With OWASP ZAP, you can use a ZAP spider or the AJAX spider. So what's the difference?

ZAP spider is a web crawler that can **automatically find security vulnerabilities in web applications**. Meanwhile, the AJAX spider is a web crawler designed to crawl and **attack AJAX-based web applications.**

Clicking on the 'Tools' option will give you a list of available pentesting tools provided by OWASP ZAP.



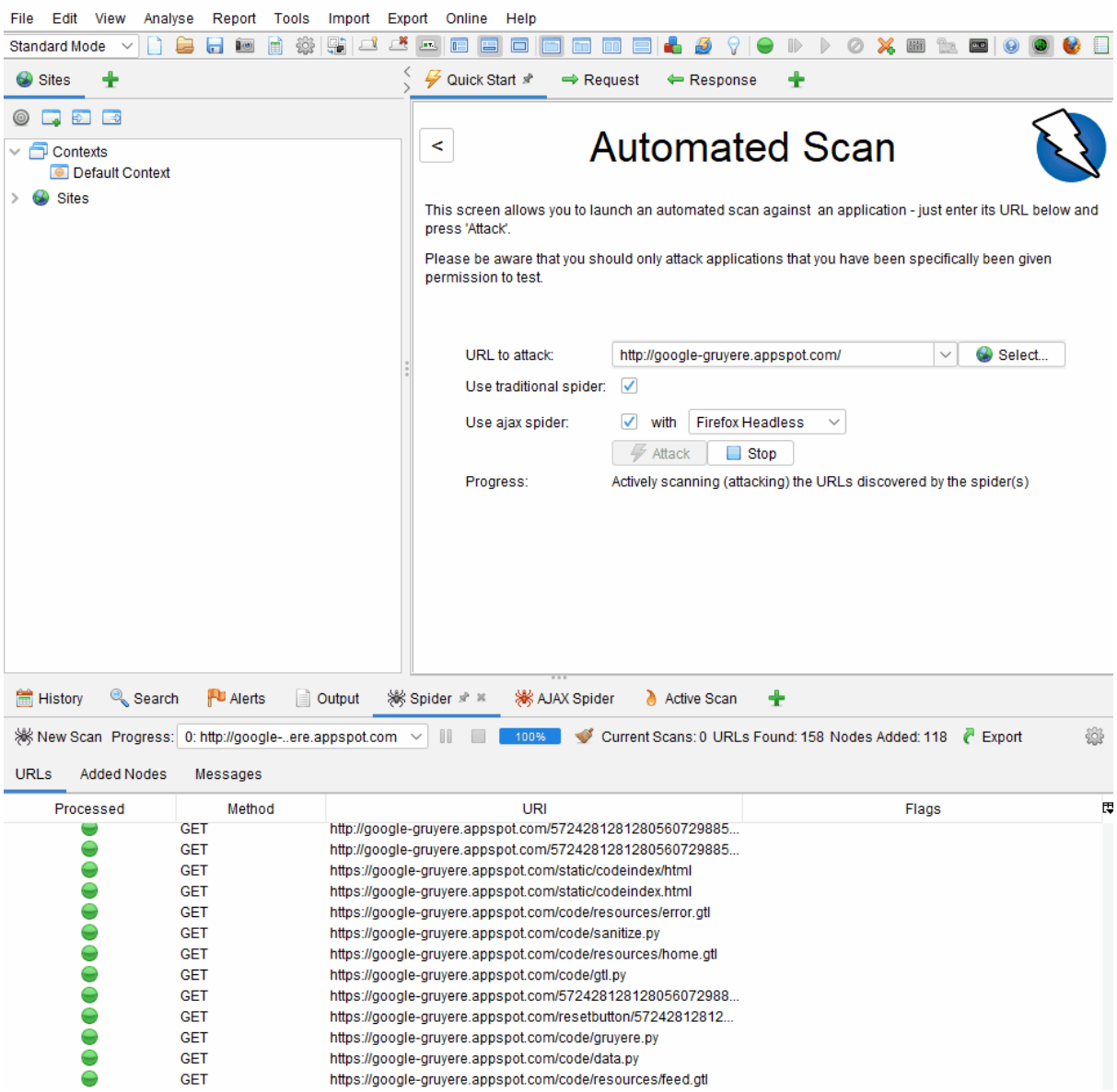
To run an automated scan, you can use the quick start “Automated Scan” option under the “Quick Start” tab. Enter the URL of the site you want to scan in the “URL to attack” field, and then click “Attack!”.



4. Interpreting test results

Interpreting test results in OWASP ZAP is vital to understand the scan findings and determine which issues require further investigation. Additionally, it can help to prioritize remediation efforts.

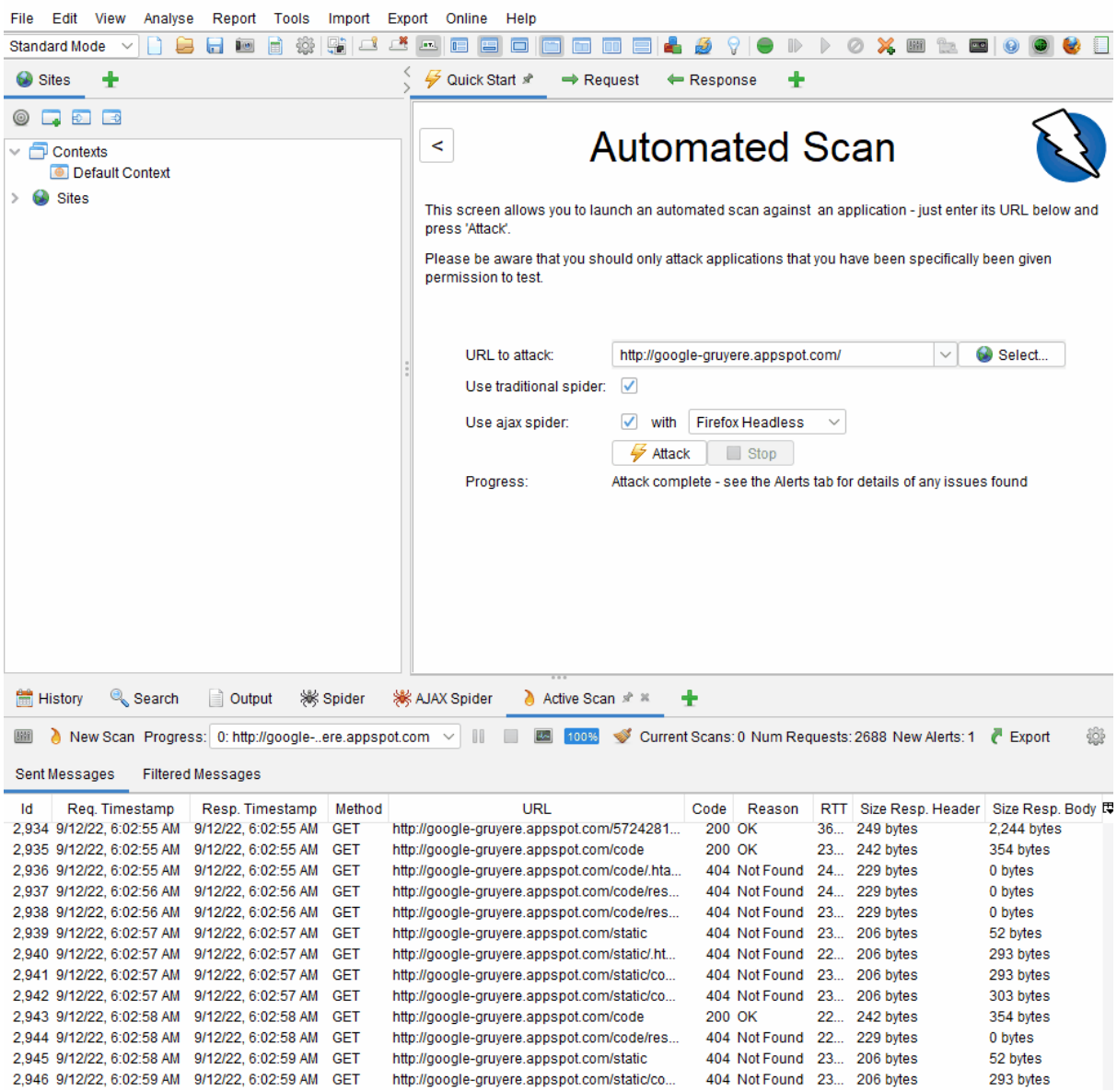
In OWASP ZAP, you can view alerts by clicking on the "Alerts" tab. This tab will show you **a list of all the alerts that have been triggered during your testing.** The alerts are sorted by risk level, with the highest risk alerts at the top of the list. OWASP ZAP will give details of the discovered vulnerabilities and suggestions on how you can fix them.



5. Viewing alerts and alert details

Viewing alerts and alert details in OWASP ZAP is a way to **see what potential security issues have been identified on a website.** It can help security and administrators understand what needs to be fixed to improve the app's security.

If you cannot find your ‘Alerts’ tab, you can access it via the ‘View’ menu, along with other options available in OWASP ZAP. Once you have your ‘Alerts’ tab, you can **navigate the various vulnerabilities discovered and explore the reports generated by OWASP ZAP.**



Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
2,934	9/12/22, 6:02:55 AM	9/12/22, 6:02:55 AM	GET	http://google-gruyere.appspot.com/5724281...	200	OK	36...	249 bytes	2,244 bytes
2,935	9/12/22, 6:02:55 AM	9/12/22, 6:02:55 AM	GET	http://google-gruyere.appspot.com/code	200	OK	23...	242 bytes	354 bytes
2,936	9/12/22, 6:02:55 AM	9/12/22, 6:02:55 AM	GET	http://google-gruyere.appspot.com/code/.hta...	404	Not Found	24...	229 bytes	0 bytes
2,937	9/12/22, 6:02:56 AM	9/12/22, 6:02:56 AM	GET	http://google-gruyere.appspot.com/code/res...	404	Not Found	24...	229 bytes	0 bytes
2,938	9/12/22, 6:02:56 AM	9/12/22, 6:02:56 AM	GET	http://google-gruyere.appspot.com/code/res...	404	Not Found	23...	229 bytes	0 bytes
2,939	9/12/22, 6:02:57 AM	9/12/22, 6:02:57 AM	GET	http://google-gruyere.appspot.com/static	404	Not Found	23...	206 bytes	52 bytes
2,940	9/12/22, 6:02:57 AM	9/12/22, 6:02:57 AM	GET	http://google-gruyere.appspot.com/static/.ht...	404	Not Found	22...	206 bytes	293 bytes
2,941	9/12/22, 6:02:57 AM	9/12/22, 6:02:57 AM	GET	http://google-gruyere.appspot.com/static/co...	404	Not Found	23...	206 bytes	293 bytes
2,942	9/12/22, 6:02:57 AM	9/12/22, 6:02:57 AM	GET	http://google-gruyere.appspot.com/static/co...	404	Not Found	23...	206 bytes	303 bytes
2,943	9/12/22, 6:02:58 AM	9/12/22, 6:02:58 AM	GET	http://google-gruyere.appspot.com/code	200	OK	22...	242 bytes	354 bytes
2,944	9/12/22, 6:02:58 AM	9/12/22, 6:02:58 AM	GET	http://google-gruyere.appspot.com/code/res...	404	Not Found	22...	229 bytes	0 bytes
2,945	9/12/22, 6:02:58 AM	9/12/22, 6:02:59 AM	GET	http://google-gruyere.appspot.com/static	404	Not Found	23...	206 bytes	52 bytes
2,946	9/12/22, 6:02:59 AM	9/12/22, 6:02:59 AM	GET	http://google-gruyere.appspot.com/static/co...	404	Not Found	23...	206 bytes	293 bytes

6. Exploring an application manually

Exploring an application manually in OWASP ZAP is a process of manually testing the application for security vulnerabilities**.** It is done to identify any potential security risks that may be present in the application. Doing this can help ensure that the application is as secure as possible.

The manual scan complements the automated scan by providing a more in-depth analysis of the application and allowing you to navigate the pentest process. The automated scan may miss some vulnerabilities, but the manual scan may pick up missed issues. However, the manual scan can be time-consuming and may not be feasible for large applications.

To explore an application manually, select “Manual Explore.” Select your browser, and OWASP ZAP will launch a proxy in your browser. Here, you will be given pentesting tools such as spiders, and if a vulnerability is discovered, an alert flag will be added to the alerts panel.

Code:

1.Install Flask:

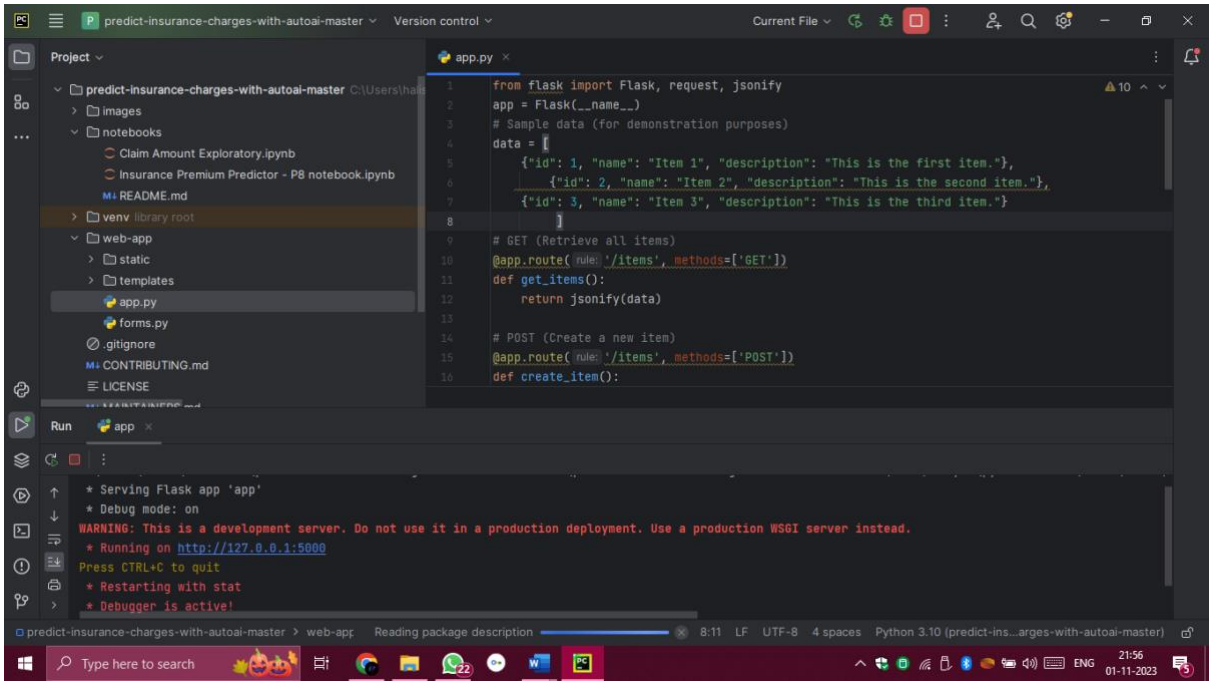
```
pip install flask
```

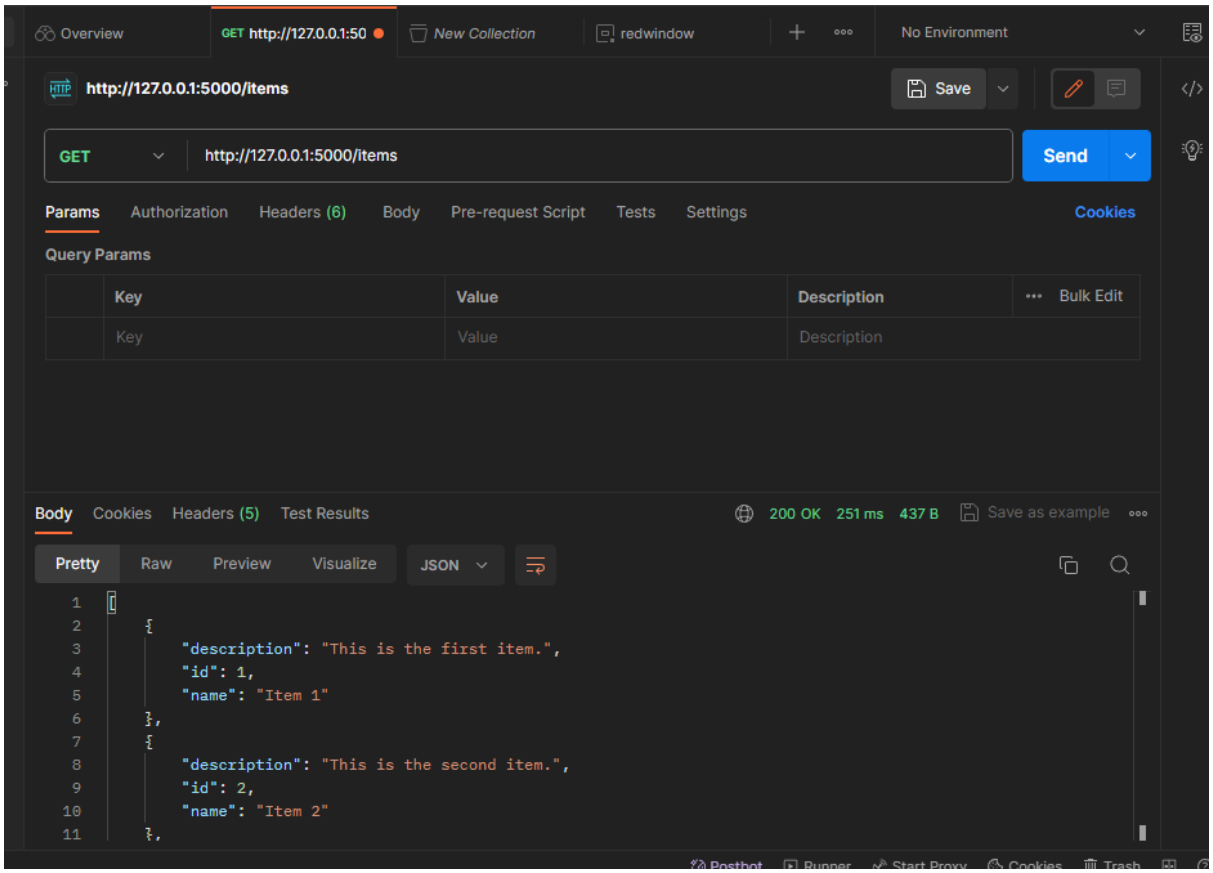
2.Create a new file called **app.py**:

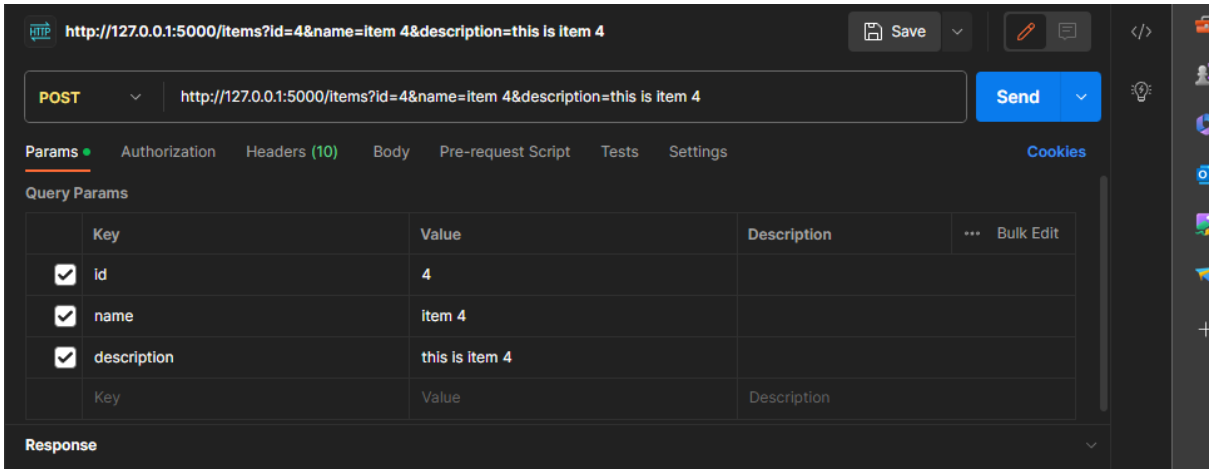
```
from flask import Flask, request, jsonify
app = Flask(__name__)
# Sample data (for demonstration purposes)
data = [
    {"id": 1, "name": "Item 1", "description": "This is the first item."},
    {"id": 2, "name": "Item 2", "description": "This is the second item."},
    {"id": 3, "name": "Item 3", "description": "This is the third item."}
        ]
# GET (Retrieve all items)
@app.route('/items', methods=['GET'])
def get_items():
    return jsonify(data)

# POST (Create a new item)
@app.route('/items', methods=['POST'])
def create_item():
    item = request.get_json()
    data.append(item)
    return jsonify({"message": "Item created successfully"})

# PUT (Update an item)
@app.route('/items/<int:item_id>', methods=['PUT'])
def update_item(item_id):
    if 0 <= item_id < len(data):
        updated_item = request.get_json()
        data[item_id] = updated_item
        return jsonify({"message": "Item updated successfully"})
    else:
```

```
        return jsonify({"message": "Item not found"}, 404)

# DELETE (Delete an item)
@app.route('/items/<int:item_id>', methods=['DELETE'])
def delete_item(item_id):
    if 0 <= item_id < len(data):
        del data[item_id]
        return jsonify({"message": "Item deleted successfully"})
    else:
        return jsonify({"message": "Item not found"}, 404)

if __name__ == '__main__':
    app.run(debug=True)
```

Output:

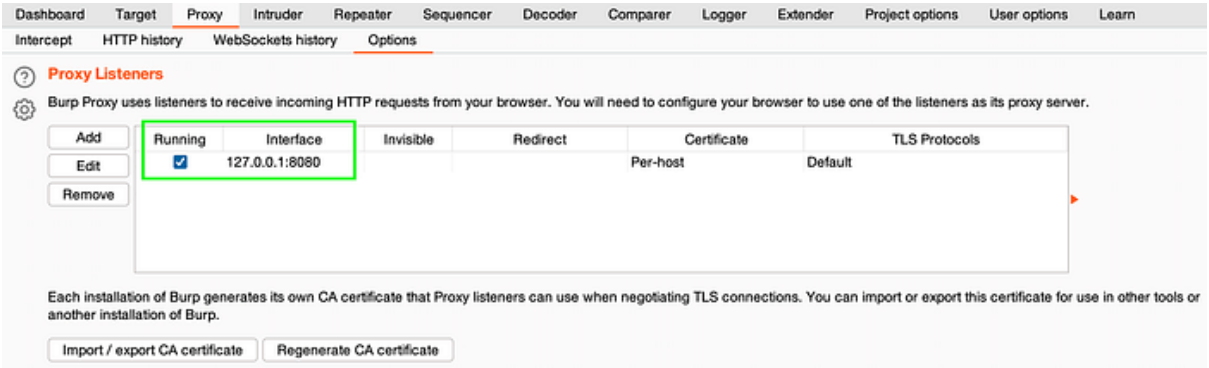

predict-insurance-charges-with-autoai-master
Version control
Current File
Project
app.py
predict-insurance-charges-with-autoai-master
images
notebooks
Claim Amount Exploratory.ipynb
Insurance Premium Predictor - P8 notebook.ipynb
README.md
venv library root
web-app
static
templates
app.py
forms.py
.gitignore
CONTRIBUTING.md
LICENSE
from flask import Flask, request, jsonify
app = Flask(__name__)
# Sample data (for demonstration purposes)
data = [
{"id": 1, "name": "Item 1", "description": "This is the first item."},
{"id": 2, "name": "Item 2", "description": "This is the second item."},
{"id": 3, "name": "Item 3", "description": "This is the third item."}
]
GET (Retrieve all items)
@app.route('/items', methods=['GET'])
def get_items():
return jsonify(data)
POST (Create a new item)
@app.route('/items', methods=['POST'])
def create_item():
Run
app
* 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!
Type here to search


Overview
GET http://127.0.0.1:50
New Collection
redwindow
No Environment
http://127.0.0.1:5000/items
Save
GET
http://127.0.0.1:5000/items
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (5)
Test Results
200 OK
251 ms
437 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
[
{
"description": "This is the first item.",
"id": 1,
"name": "Item 1"
},
{
"description": "This is the second item.",
"id": 2,
"name": "Item 2"
},


http://127.0.0.1:5000/items?id=4&name=item 4&description=this is item 4
Save
POST
http://127.0.0.1:5000/items?id=4&name=item 4&description=this is item 4
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
id
4
name
item 4
description
this is item 4
Key
Value
Description
Response

Prepare the Burp Suite

1. Download and install Burp Suite Community Edition;
2. Run Burp Suite Community Edition and choose on the start screen: Temporary project → [Next] → Use Burp defaults → [Start Burp]
3. Check Burp's proxy settings: Proxy → Options → Proxy Listeners. Burp's proxy should listen **127.0.0.1:8080**

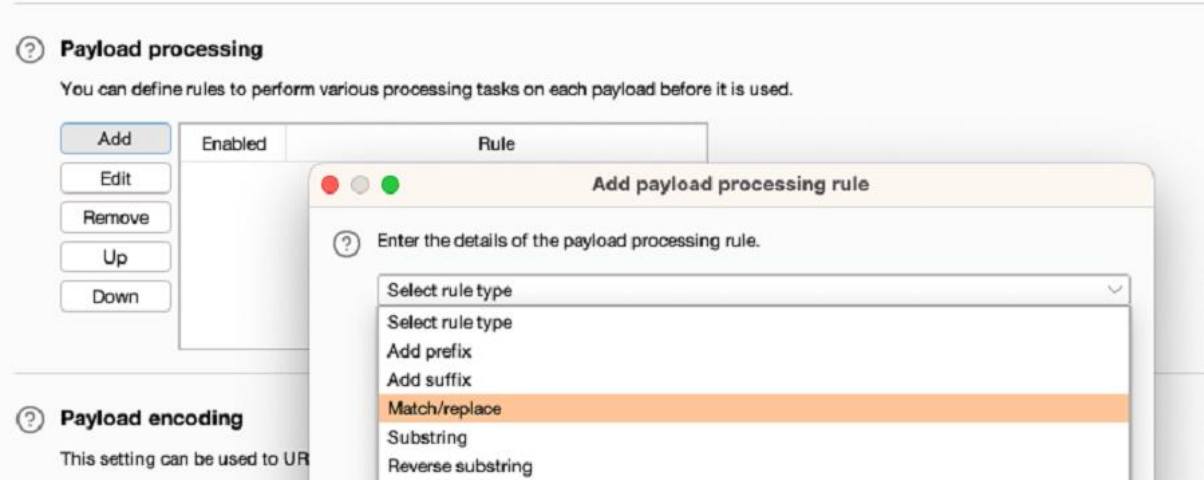


Procedure

SQL injection

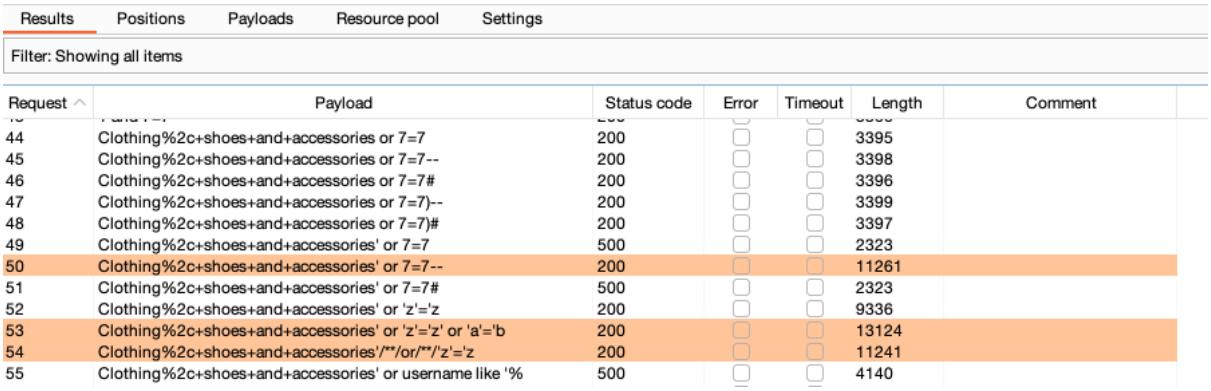
1. Identify a request that you want to investigate.
2. In the request, highlight the parameter that you want to test and select Send to Intruder.
3. Go to the **Intruder > Positions** tab. Notice that the parameter has been automatically marked as a payload position.
4. Go to the **Payloads** tab. Under **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.

    If you're using the lab, look for responses with a longer length. These may include additional products.



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	

cross-site scripting (XSS)

Bypassing XSS filters by enumerating permitted tags and attributes

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.

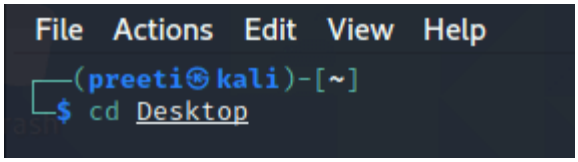
3. Identify whether any attributes are permitted:
    1. In **Intruder > Positions**, update the payload position. Add a tag that you enumerated in the previous step, then add payload markers to test different attributes.

    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 attributes 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.

Procedure

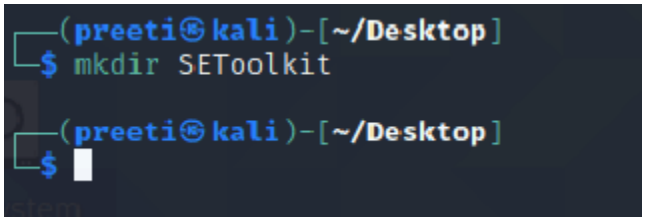
Step 1: First, we have to open the Kali Linux Terminal and move to Desktop.

1. cd Desktop



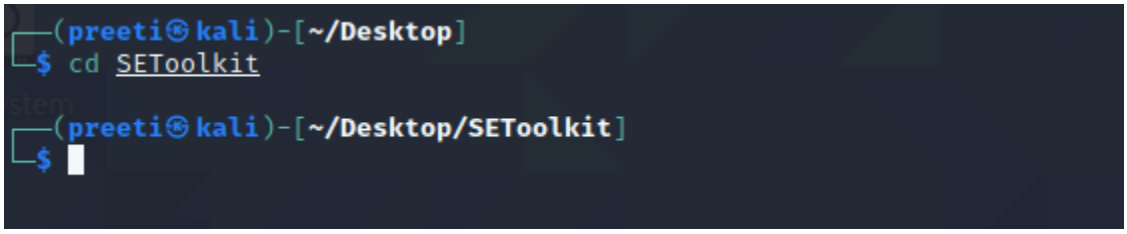
Step 2: Now, we are on a desktop so use the following command in order to create a new directory called SEToolkit.

1. mkdir SEToolkit



Step 3: Now, we are in the Desktop directory though we have created a SEToolkit directory so go to SEToolkit directory using the following command.

1. cd SEToolkit



Step 4: Now we're in the SEToolkit directory, we will need to clone SEToolkit from GitHub in order to utilize it.

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

```
┌──(preeti㉿kali)-[~/Desktop]
└─$ cd SEToolkit

┌──(preeti㉿kali)-[~/Desktop/SEToolkit]
└─$ git clone https://github.com/trustedsec/social-engineer-toolkit setoolkit/
Cloning into 'setoolkit'...
remote: Enumerating objects: 110242, done.
remote: Counting objects: 100% (180/180), done.
remote: Compressing objects: 100% (151/151), done.
remote: Total 110242 (delta 95), reused 63 (delta 28), pack-reused 110062
Receiving objects: 100% (110242/110242), 175.29 MiB | 9.77 MiB/s, done.
Resolving deltas: 100% (68354/68354), done.
```

Step 5: Now, the Social Engineering Toolkit has been downloaded to our directory, we have to use the following command in order to navigate to the social engineering toolkit's internal directory.

1. cd setoolkit

```
┌──(preeti㉿kali)-[~/Desktop/SEToolkit]
└─$ cd setoolkit

┌──(preeti㉿kali)-[~/Desktop/SEToolkit/setoolkit]
└─$ ls
Dockerfile  readme      requirements.txt  seproxy    setup.py  src
modules     README.md   seautomate        setoolkit  seupdate
```

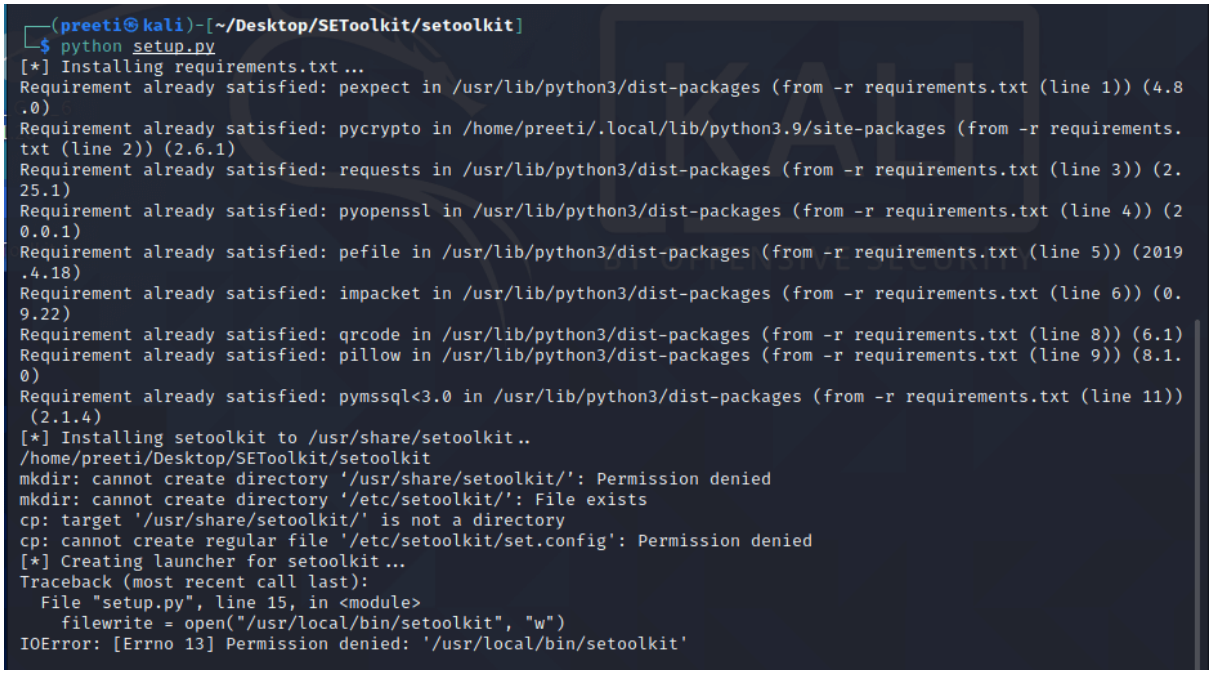
Step 6: Now we have successfully downloaded the social engineering toolkit in our directory SEToolkit. Now we can use the following command to install the requirements.

1. pip3 install -r requirements.text

```
┌──(preeti㉿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)) (4.8
.0)
Collecting pycrypto
  Downloading pycrypto-2.6.1.tar.gz (446 kB)
     |████████████████████████████████| 446 kB 4.1 MB/s
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from -r requirements.txt (line 3)) (2.
25.1)
Requirement already satisfied: pyopenssl in /usr/lib/python3/dist-packages (from -r requirements.txt (line 4)) (2
0.0.1)
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.
9.22)
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)) (8.1.
0)
Requirement already satisfied: pymssql<3.0 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 11))
 (2.1.4)
Building wheels for collected packages: pycrypto
  Building wheel for pycrypto (setup.py) ... done
  Created wheel for pycrypto: filename=pycrypto-2.6.1-cp39-cp39-linux_x86_64.whl size=526405 sha256=7ddddde3a531c
35a8d26af25586fcc0a1f45377dcd1cbeb4a15e576309f682de
  Stored in directory: /home/preeti/.cache/pip/wheels/9d/29/32/8b8f22481bec8b0fbe7087927336ec167faff2ed9db849448f
Successfully built pycrypto
Installing collected packages: pycrypto
Successfully installed pycrypto-2.6.1
```

Step 7: All the requirements have been downloaded to our setoolkit. Now it's time to install the requirements we have downloaded.

1. python setup.py

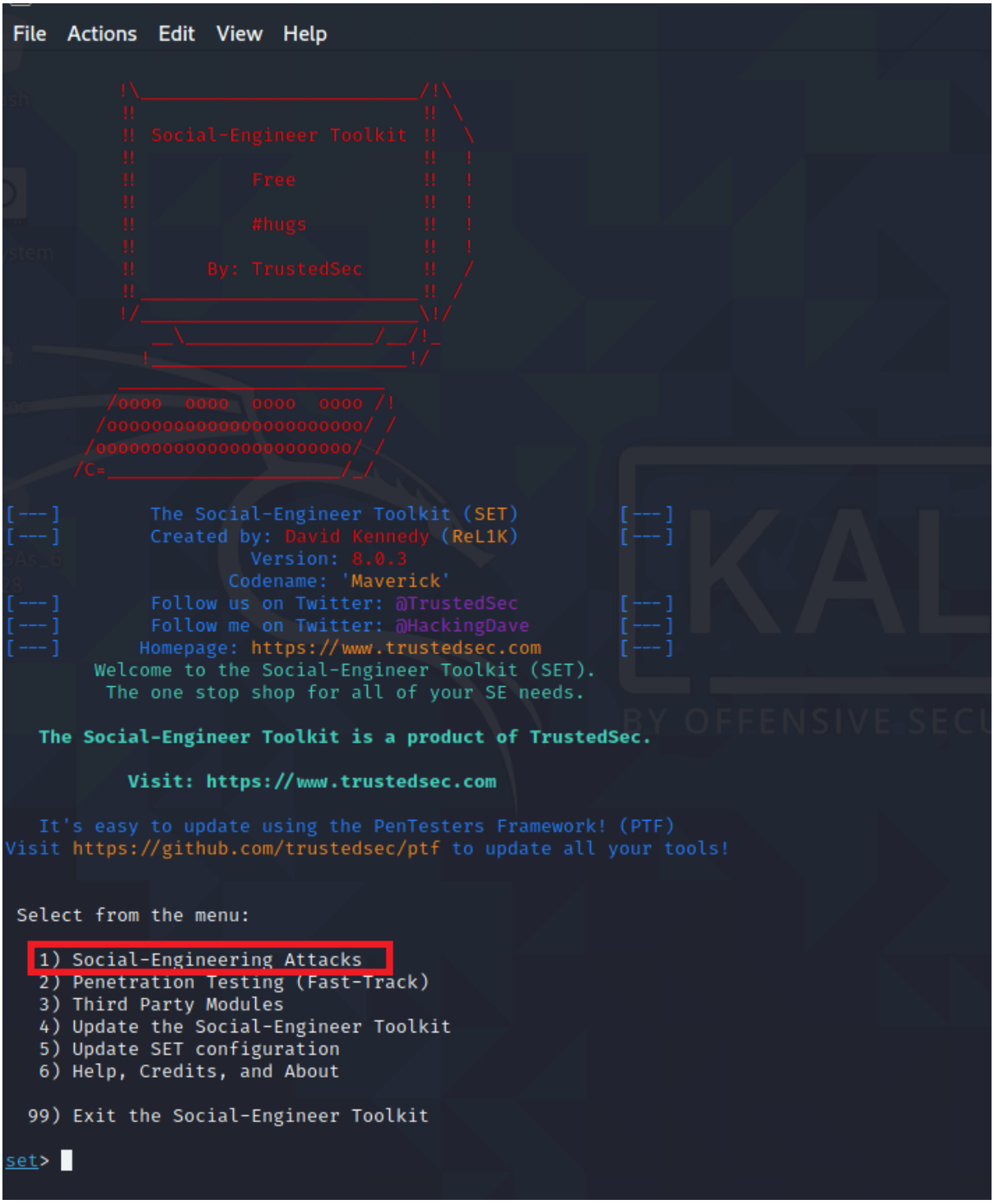


Step 8: Finally all the installation process is complete now it's time to run the Social Engineering Toolkit. We have to type the following command in order to run the SEToolkit.

1. Setoolkit

Step 9: At this point, setoolkit will ask us (y) or (n). When we type y, our social engineering toolkit will start running.

1. Y



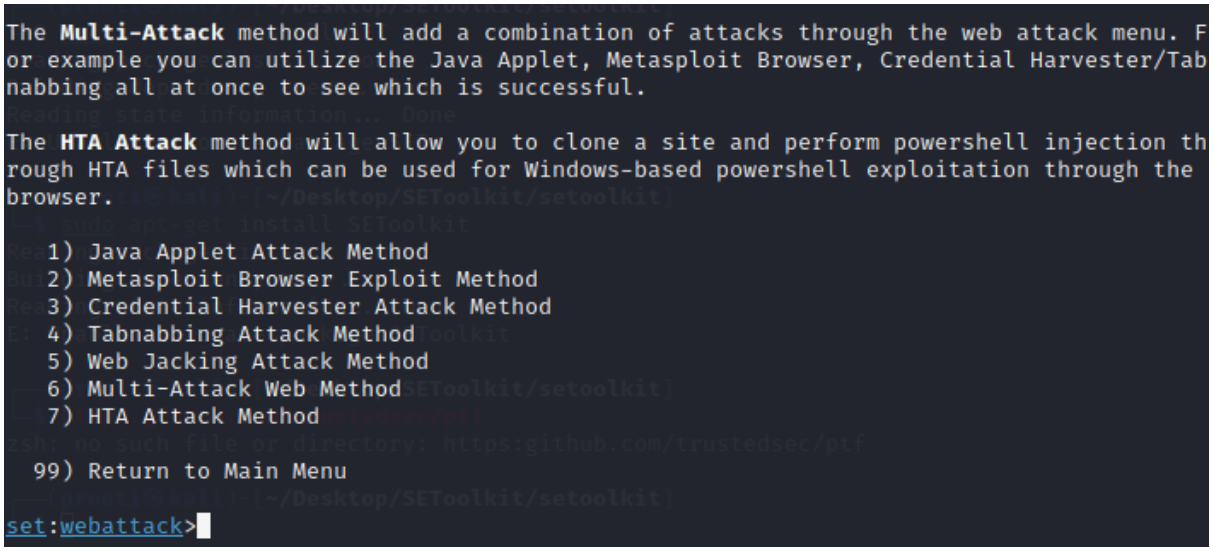
Step 10: Now our SEToolkit has been downloaded on our system, it's, time to use it. Now, we have to select the option from the following options. **Option 2** is the one we've chosen.

Website Attack Vectors:

1. Option 2

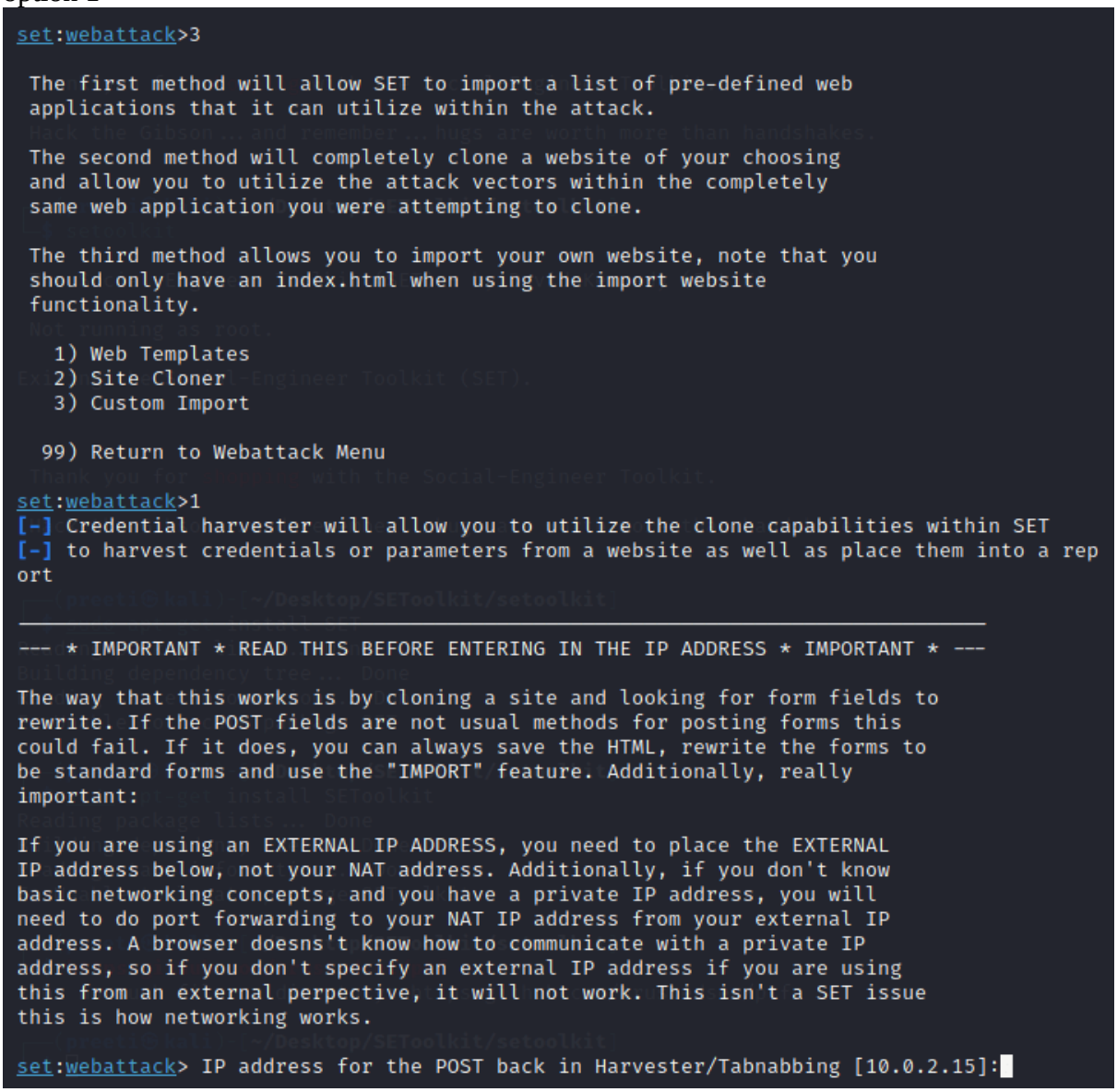
Step 11: Now, we are ready to set up a **phishing page,** we'll go with **option 3,** which is a **credential harvester attack method.**

1. option 3

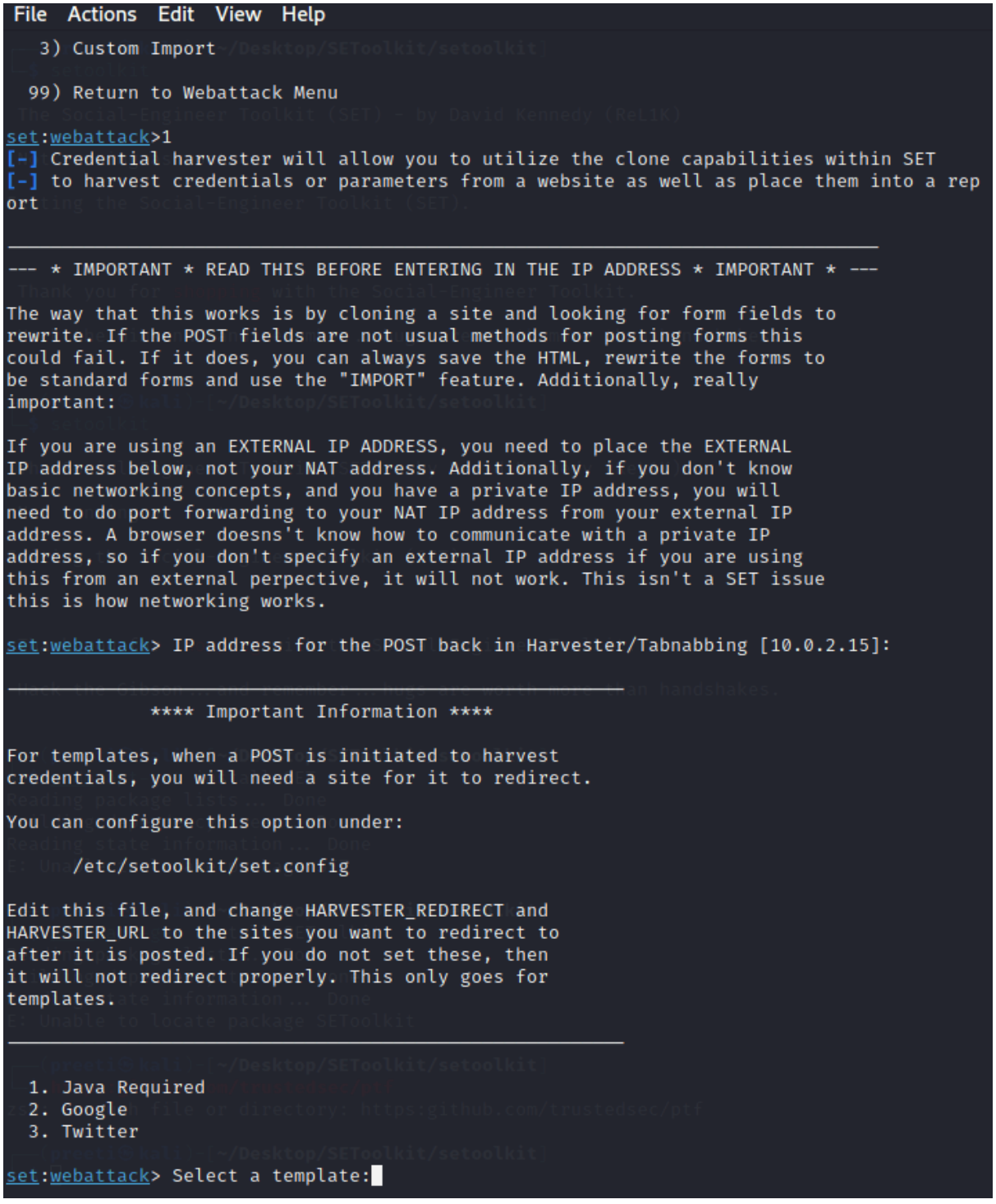


Step 12: Since we are making a **phishing page;** we'll go with **option 1,** which is a **web template.**

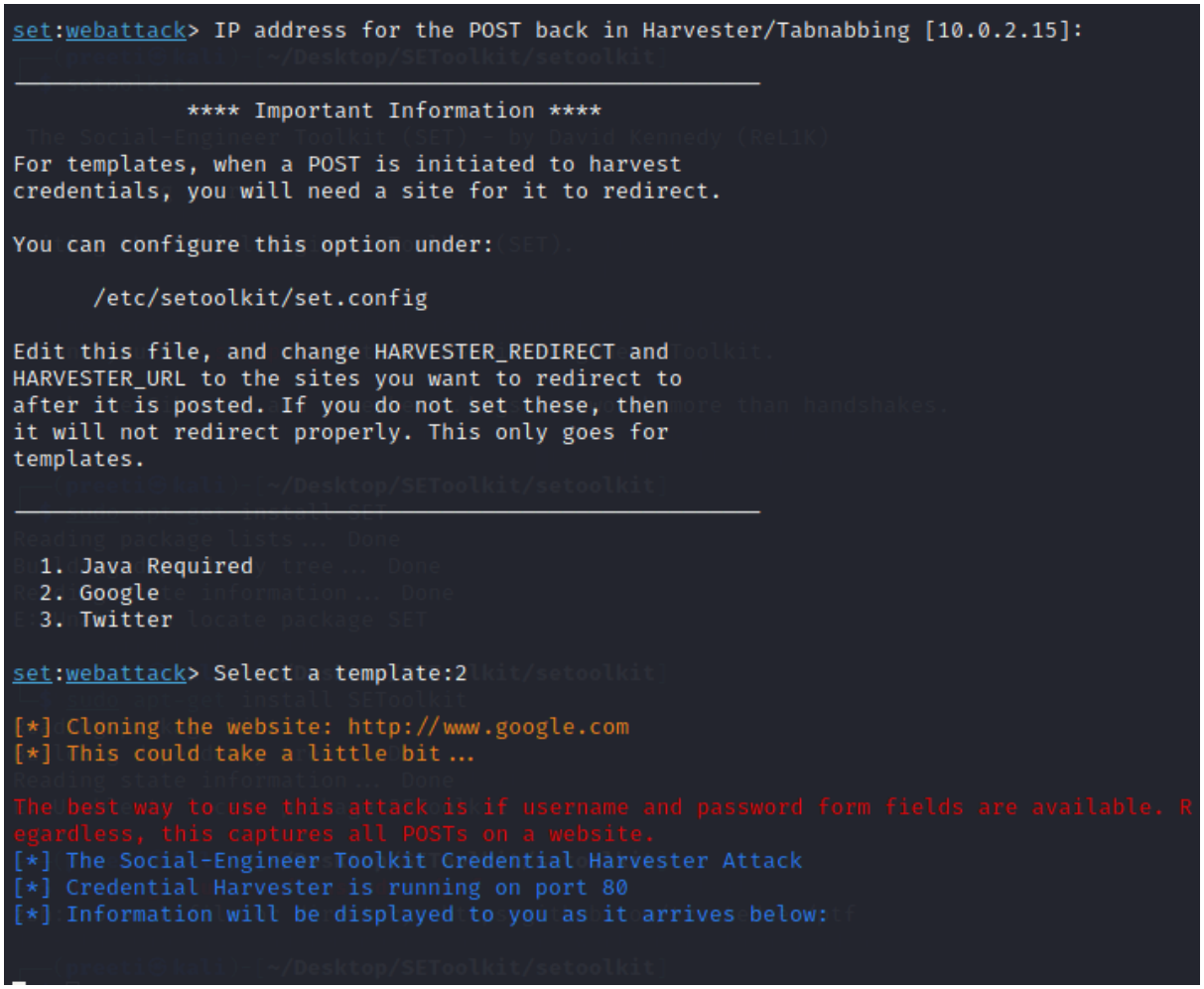
1. option 1



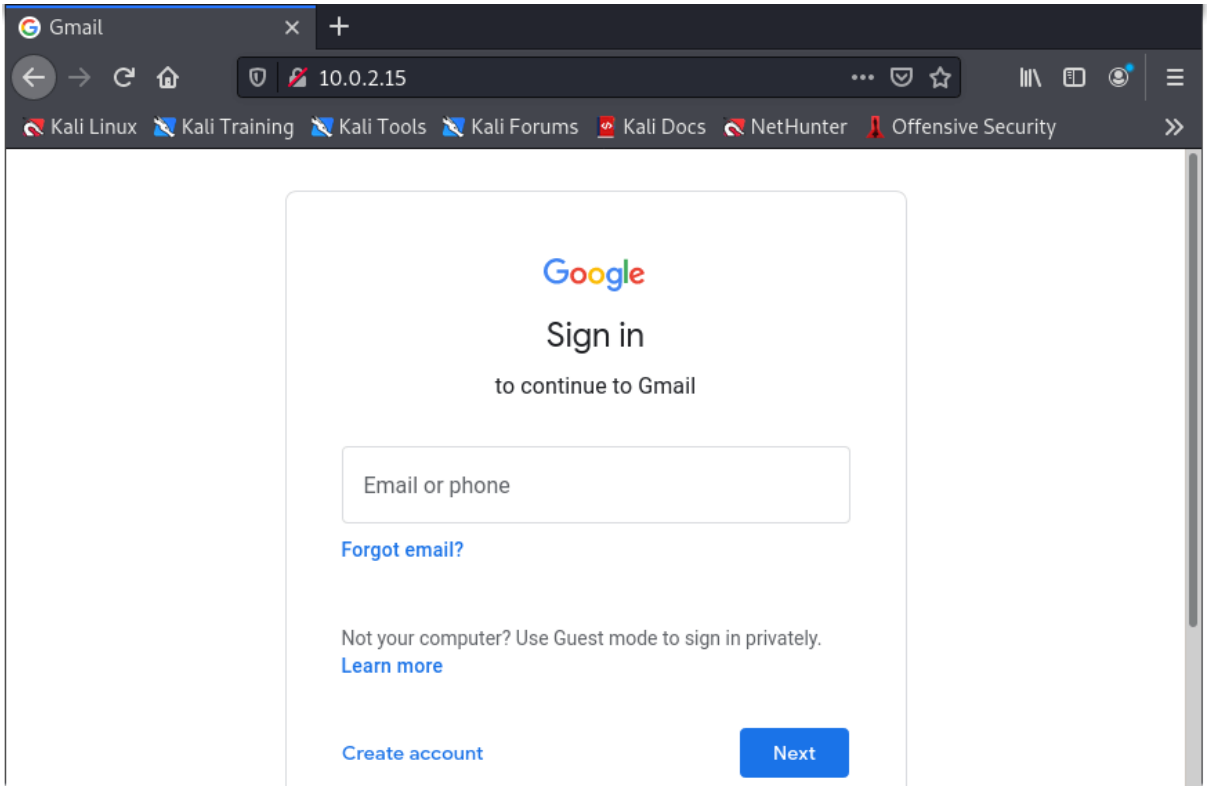
Step 13: The social engineering tool will now create a phishing page on our localhost.



Step 14: Choose **option 2** in order to create a Google phishing page, and a phishing page will be generated on our localhost.



Step 15: A **phishing page** for **Google** is being created using the **social engineering toolkit.** As we can see, **SEToolkit** generate a phishing page of Google on our localhost (i.e., on our IP address). The social engineering toolset works in this manner. The social engineering toolkit will design our phishing page. Once the victim types the **id password** in the fields the **id password** will be shown on our terminal where SET is running.


Gmail
10.0.2.15
Kali Linux
Kali Training
Kali Tools
Kali Forums
Kali Docs
NetHunter
Offensive Security
Google
Sign in
to continue to Gmail
Email or phone
Forgot email?
Not your computer? Use Guest mode to sign in privately.
Learn more
Create account
Next