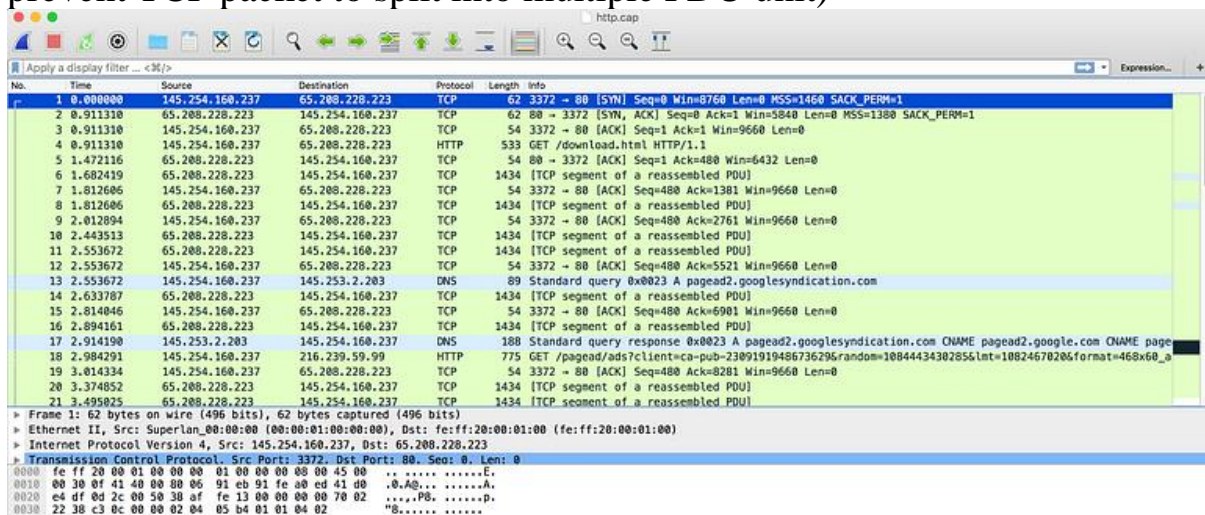


## Installation of Wireshark Software

- Open the web browser.
- Search for 'Download Wireshark.'
- Select the Windows installer according to your system configuration, either 32-bit 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)

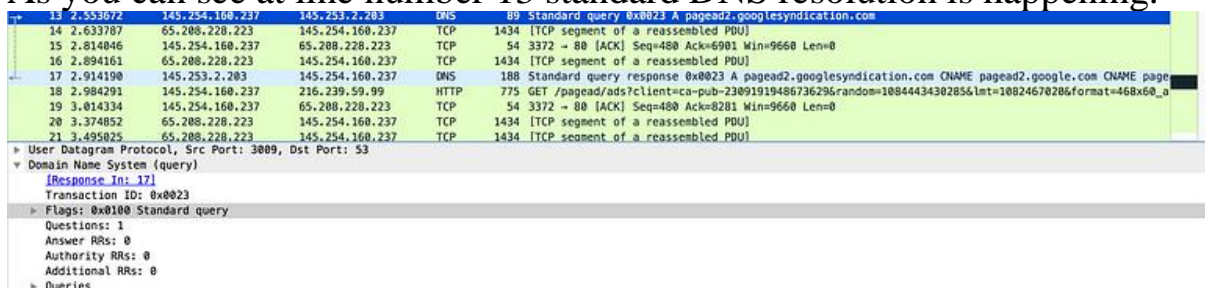


The screenshot shows a Wireshark packet capture of an HTTP transaction. The packet list on the left shows 21 packets. Packet 13 is a DNS standard query for pagead2.google.com. Packet 14 is a TCP segment of a reassembled PDU. Packet 15 is a TCP segment of a reassembled PDU. Packet 16 is a TCP segment of a reassembled PDU. Packet 17 is a DNS standard query response for pagead2.google.com. Packet 18 is a TCP segment of a reassembled PDU. Packet 19 is a TCP segment of a reassembled PDU. Packet 20 is a TCP segment of a reassembled PDU. Packet 21 is a TCP segment of a reassembled PDU. The packet details pane on the right shows the selected packet (13) and its details: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The packet bytes pane at the bottom shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	145.254.160.237	65.208.228.223	TCP	62	3372 → 80 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=1
2	0.911310	65.208.228.223	145.254.160.237	TCP	54	80 → 3372 [ACK] Seq=1 Win=5840 Len=0 MSS=1380 SACK_PERM=1
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=1 Ack=1 Win=9660 Len=0
4	0.911310	145.254.160.237	65.208.228.223	HTTP	533	GET /download.html HTTP/1.1
5	1.472116	65.208.228.223	145.254.160.237	TCP	54	80 → 3372 [ACK] Seq=1 Ack=480 Win=6432 Len=0
6	1.682419	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=1381 Win=9660 Len=0
8	1.812606	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
9	2.012894	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=2761 Win=9660 Len=0
10	2.443513	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
11	2.553672	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
12	2.553672	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=5521 Win=9660 Len=0
13	2.553672	145.254.160.237	145.253.2.203	DNS	89	Standard query 0x0023 A pagead2.google.com
14	2.633787	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
15	2.814046	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=6901 Win=9660 Len=0
16	2.894161	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
17	2.914190	145.253.2.203	145.254.160.237	DNS	188	Standard query response 0x0023 A pagead2.google.com CNAME pagead2.google.com CNAME pagead2.google.com
18	2.984291	145.254.160.237	216.239.59.99	HTTP	775	GET /pagead/ads?client=ca-pub-2309191948673629&random=10844434302856&mt=1082467020&format=468x60_o
19	3.014334	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=8281 Win=9660 Len=0
20	3.374852	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
21	3.495825	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]

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.



The screenshot shows a Wireshark packet capture of a DNS transaction. The packet list on the left shows 21 packets. Packet 13 is a DNS standard query for pagead2.google.com. Packet 14 is a TCP segment of a reassembled PDU. Packet 15 is a TCP segment of a reassembled PDU. Packet 16 is a TCP segment of a reassembled PDU. Packet 17 is a DNS standard query response for pagead2.google.com. Packet 18 is a TCP segment of a reassembled PDU. Packet 19 is a TCP segment of a reassembled PDU. Packet 20 is a TCP segment of a reassembled PDU. Packet 21 is a TCP segment of a reassembled PDU. The packet details pane on the right shows the selected packet (13) and its details: User Datagram Protocol, Domain Name System (query), and [Response In: 17]. The packet bytes pane at the bottom shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
13	2.553672	145.254.160.237	145.253.2.203	DNS	89	Standard query 0x0023 A pagead2.google.com
14	2.633787	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
15	2.814046	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=6901 Win=9660 Len=0
16	2.894161	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
17	2.914190	145.253.2.203	145.254.160.237	DNS	188	Standard query response 0x0023 A pagead2.google.com CNAME pagead2.google.com CNAME pagead2.google.com
18	2.984291	145.254.160.237	216.239.59.99	HTTP	775	GET /pagead/ads?client=ca-pub-2309191948673629&random=10844434302856&mt=1082467020&format=468x60_o
19	3.014334	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=8281 Win=9660 Len=0
20	3.374852	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
21	3.495825	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]

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

17	2.914190	145.253.2.203	145.254.160.237	DNS	188	Standard query response 0x0023 A pagead2.googlesyndication.com CNAME pagead2.google.com CNAME pagead2.google.com
18	2.984291	145.254.160.237	216.239.59.99	HTTP	775	GET /pagead/ads?client=ca-pub-2309191948673629&random=10844434302856&mt=1082467020&format=460x60_u
19	3.014334	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=8281 Win=9660 Len=0
20	3.374852	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]
21	3.495025	65.208.228.223	145.254.160.237	TCP	1434	[TCP segment of a reassembled PDU]

Transaction ID: 0x0023

Flags: 0x0180 Standard query response, No error

Questions: 1

Answer RRs: 4

Authority RRs: 0

Additional RRs: 0

Queries

- pagead2.googlesyndication.com: type A, class IN

Answers

- pagead2.googlesyndication.com: type CNAME, class IN, cname pagead2.google.com
- pagead2.google.com: type CNAME, class IN, cname pagead2.google.akadns.net
- pagead2.google.akadns.net: type A, class IN, addr 216.239.59.104
- pagead2.google.akadns.net: type A, class IN, addr 216.239.59.99

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)

4	0.911310	145.254.160.237	65.208.228.223	HTTP	533	GET /download.html HTTP/1.1
5	1.472116	65.208.228.223	145.254.160.237	TCP	54	80 → 3372 [ACK] Seq=1 Ack=480 Win=6432 Len=0
6	1.682419	65.208.228.223	145.254.160.237	HTTP	1434	HTTP/1.1 200 OK
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=1381 Win=9660 Len=0
8	1.812606	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
9	2.012894	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=2761 Win=9660 Len=0
10	2.443513	65.208.228.223	145.254.160.237	HTTP	1434	Continuation

Frame 4: 533 bytes on wire (4264 bits), 533 bytes captured (4264 bits)

Ethernet II, Src: Superlan\_00:00:00 (00:00:00:00:00:00), Dst: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)

Internet Protocol Version 4, Src: 145.254.160.237, Dst: 65.208.228.223

Transmission Control Protocol, Src Port: 3372, Dst Port: 80, Seq: 1, Ack: 1, Len: 479

Hypertext Transfer Protocol

GET /download.html HTTP/1.1\r\n

Host: www.ethereal.com\r\n

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040113\r\n

Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,\*/\*;q=0.1\r\n

Accept-Language: en-us,en;q=0.5\r\n

Accept-Encoding: gzip,deflate\r\n

Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7\r\n

Keep-Alive: 300\r\n

Connection: keep-alive\r\n

Referer: http://www.ethereal.com/development.html\r\n

\r\n

[Full request URI: http://www.ethereal.com/download.html]

[HTTP request 1/1]

[Response in frame: 6]

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 find 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,

6	1.682419	65.208.228.223	145.254.160.237	HTTP/_	1434	HTTP/1.1 200 OK
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=1381 Win=9660 Len=0
8	1.812606	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
9	2.012894	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=2761 Win=9660 Len=0
10	2.443513	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
▶ Frame 6: 1434 bytes on wire (11472 bits), 1434 bytes captured (11472 bits) on interface 0 ▶ Ethernet II, Src: fe80:20:00:01:00 (fe80:20:00:01:00), Dst: Superlan_00:00:00:00:00:00 (00:00:01:00:00:00) ▶ Internet Protocol Version 4, Src: 65.208.228.223, Dst: 145.254.160.237 ▶ Transmission Control Protocol, Src Port: 80, Dst Port: 3372, Seq: 1, Ack: 480, Len: 1380 ▼ Hypertext Transfer Protocol ▶ HTTP/1.1 200 OK\r\n Date: Thu, 13 May 2004 10:17:12 GMT\r\n Server: Apache\r\n Last-Modified: Tue, 20 Apr 2004 13:17:00 GMT\r\n ETag: "9a01a-4696-7e354b00"\r\n Accept-Ranges: bytes\r\n Content-Length: 18070\r\n Keep-Alive: timeout=15, max=100\r\n Connection: Keep-Alive\r\n Content-Type: text/html; charset=ISO-8859-1\r\n \r\n [HTTP response 1/1] [Time since request: 0.771109000 seconds] [Request in frame: 4] File Data: 1086 bytes						

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

6	1.682419	65.208.228.223	145.254.160.237	HTTP/_	1434	HTTP/1.1 200 OK
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=1381 Win=9660 Len=0
8	1.812606	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
9	2.012894	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=2761 Win=9660 Len=0
10	2.443513	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
11	2.553672	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
12	2.553672	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=5521 Win=9660 Len=0
13	2.553672	145.254.160.237	145.253.2.203	DNS	89	Standard query 0x0023 A pagead2.googlesyndication.com
14	2.633787	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
15	2.814846	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=6901 Win=9660 Len=0
16	2.894161	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
17	2.914190	145.253.2.203	145.254.160.237	DNS	188	Standard query response 0x0023 A pagead2.googlesyndication.com CNAME pagead2.google.com CNAME page
18	2.984291	145.254.160.237	216.239.59.99	HTTP	775	GET /pagead/ads?client=ca-pub-2309191948673629&random=1084443430285&lt=1082467020&format=468x60_a
19	3.014334	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=8281 Win=9660 Len=0
20	3.374852	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
21	3.495825	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
22	3.495825	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=11041 Win=9660 Len=0
23	3.635227	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
24	3.645241	216.239.59.99	145.254.160.237	TCP	54	80 → 3371 [ACK] Seq=1 Ack=722 Win=31460 Len=0
25	3.815486	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=12421 Win=9660 Len=0
26	3.915630	216.239.59.99	145.254.160.237	HTTP	1484	HTTP/1.1 200 OK (text/html)
27	3.955688	216.239.59.99	145.254.160.237	HTTP	214	Continuation
28	3.955688	145.254.160.237	216.239.59.99	TCP	54	3371 → 80 [ACK] Seq=722 Ack=1591 Win=8760 Len=0
29	4.105904	65.208.228.223	145.254.160.237	HTTP	1434	Continuation
30	4.216062	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=13801 Win=9660 Len=0

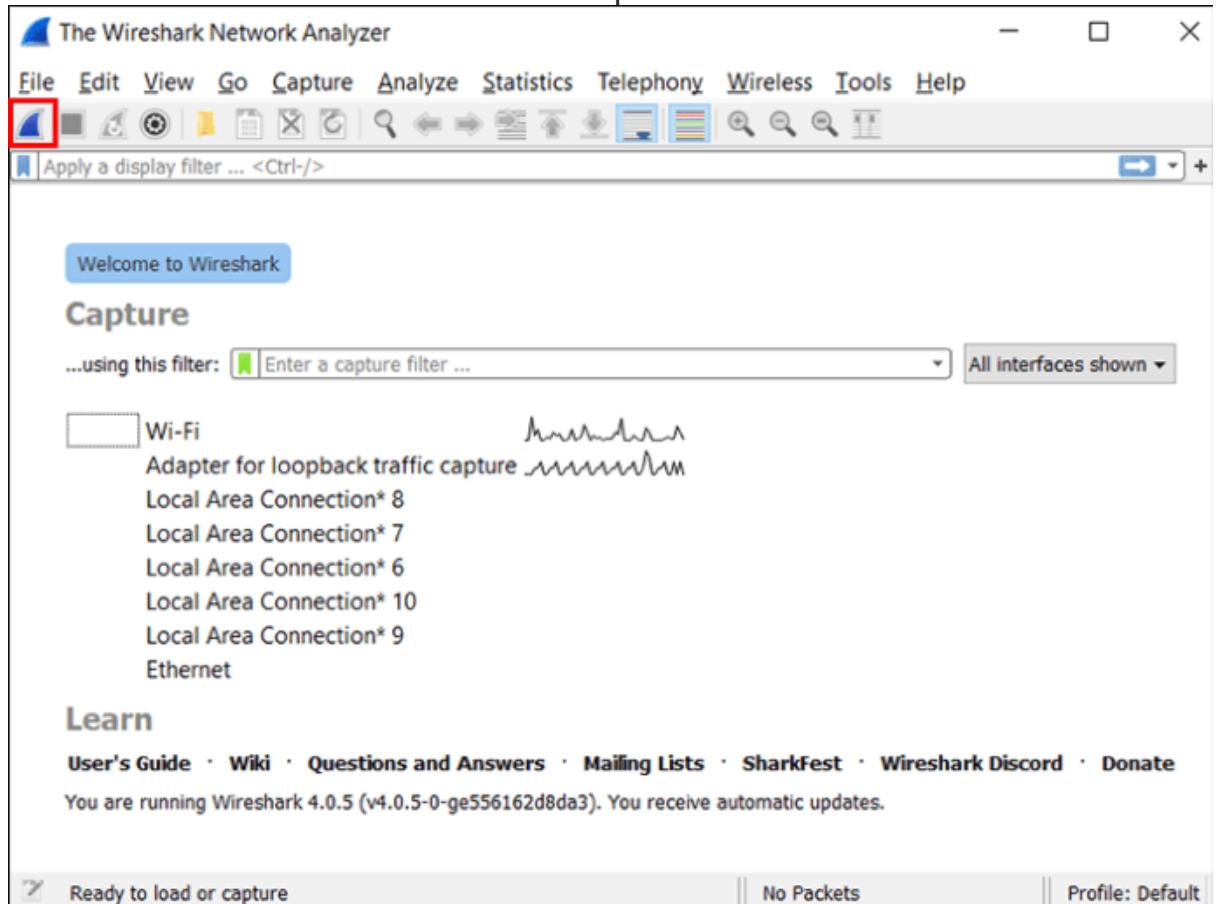
and at that top some usual TCP handshake

1	0.000000	145.254.160.237	65.208.228.223	TCP	62	3372 → 80 [SYN] Seq=0 Win=0 MSS=1460 SACK_PERM=1
2	0.911310	65.208.228.223	145.254.160.237	TCP	62	80 → 3372 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 SACK_PERM=1
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=1 Ack=1 Win=9660 Len=0

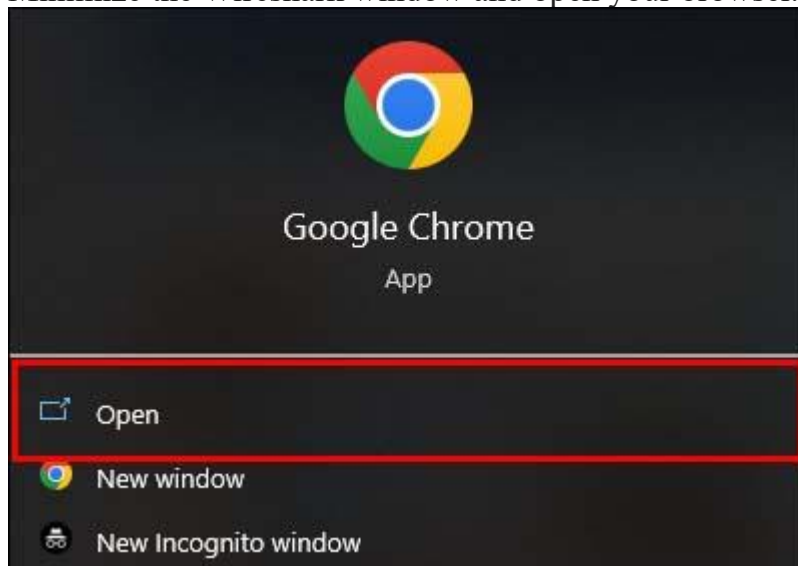
## capture HTTPS in Wireshark

### Capture and Decrypt Session Keys

1. Launch Wireshark and start an unfiltered capture session.

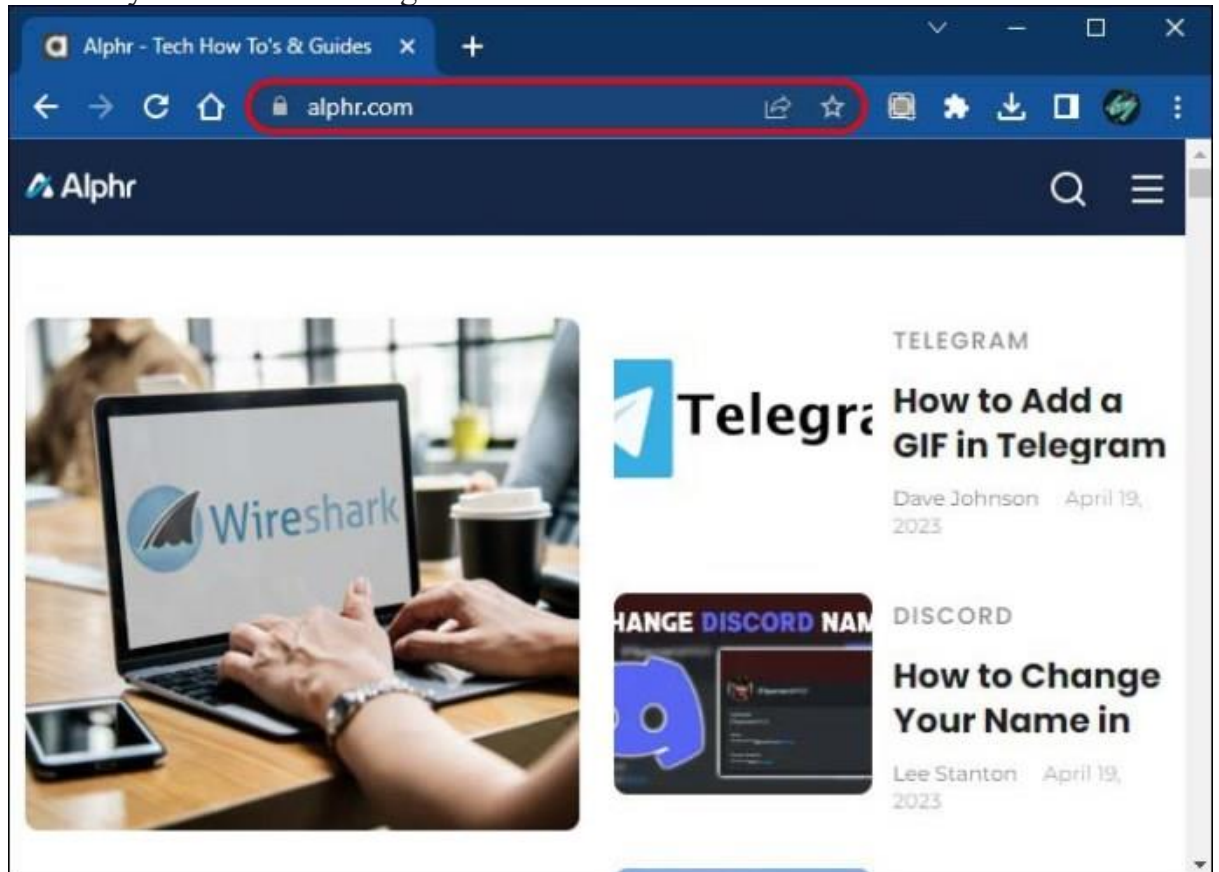


2. Minimize the Wireshark window and open your browser.

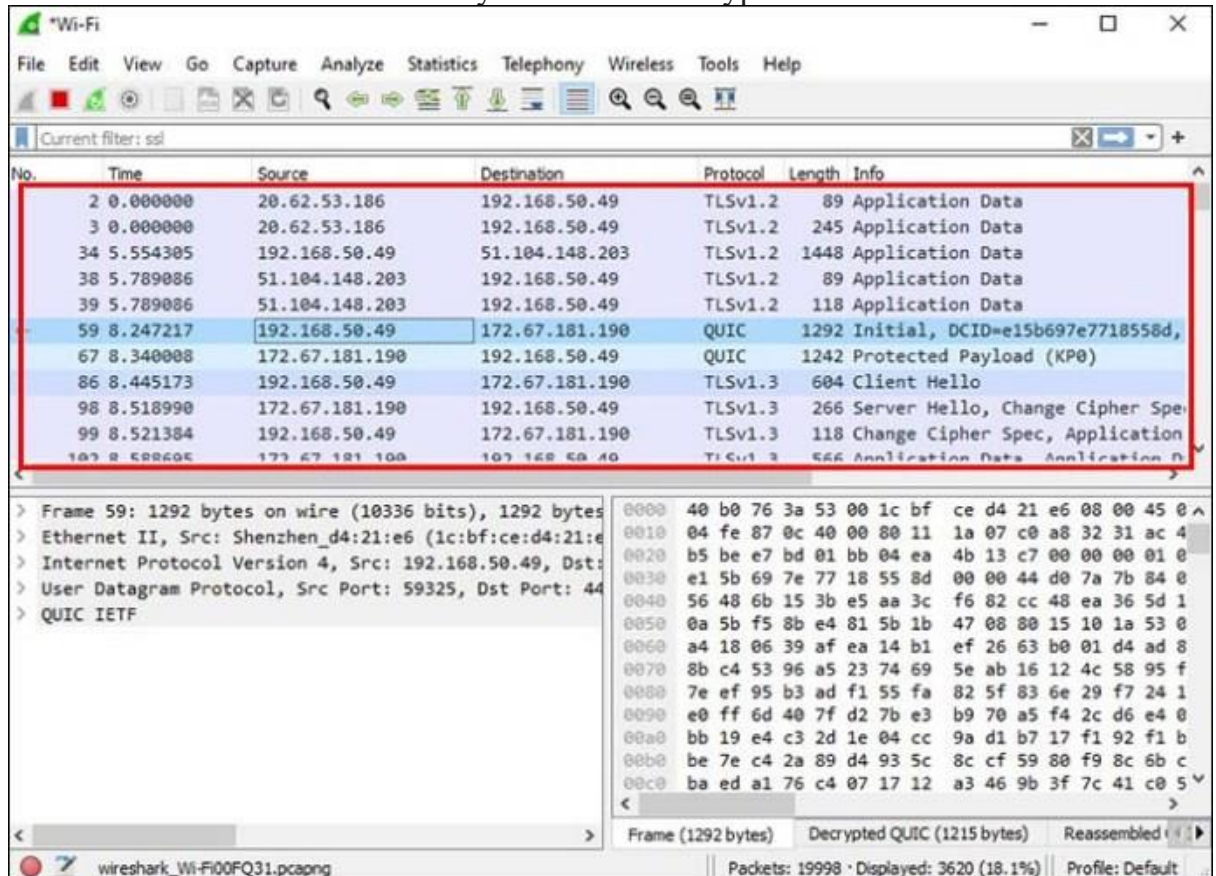




3. Go to any secure website to get data.



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



- 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.



The screenshot shows the OWASP ZAP website's download page. At the top, there's a navigation bar with links for Home, Blog, Videos, Documentation, and Community, along with a search icon and a 'Download' button. Below the navigation bar is a large blue banner with the text 'Download ZAP'. Underneath the banner, there are two informational points: one about checksums being maintained on the 2.11.1 Release Page and version files, and another strongly recommending that ZAP be installed on fully patched and actively maintained operating systems and JREs. The main section is titled 'ZAP 2.11.1' and lists seven download options with their respective sizes and 'Download' buttons: Windows (64) Installer (183 MB), Windows (32) Installer (183 MB), Linux Installer (188 MB), Linux Package (186 MB), MacOS Installer (213 MB), Cross Platform Package (204 MB), and Core Cross Platform Package (55 MB). Below the list, there are several bullet points providing additional information: most files contain a default set of functionality that can be extended via the ZAP Marketplace; the core package is the minimal set needed to get started; Windows and Linux versions require Java 8 or higher; the macOS version includes Java 11, while Linux and Cross Platform versions do not; installers use a multi-platform installer builder with an unattended mode; and a link to release notes for more information.

Download Option	Size	Action
<a href="#">Windows (64) Installer</a>	183 MB	<a href="#">Download</a>
<a href="#">Windows (32) Installer</a>	183 MB	<a href="#">Download</a>
<a href="#">Linux Installer</a>	188 MB	<a href="#">Download</a>
<a href="#">Linux Package</a>	186 MB	<a href="#">Download</a>
<a href="#">MacOS Installer</a>	213 MB	<a href="#">Download</a>
<a href="#">Cross Platform Package</a>	204 MB	<a href="#">Download</a>
<a href="#">Core Cross Platform Package</a>	55 MB	<a href="#">Download</a>

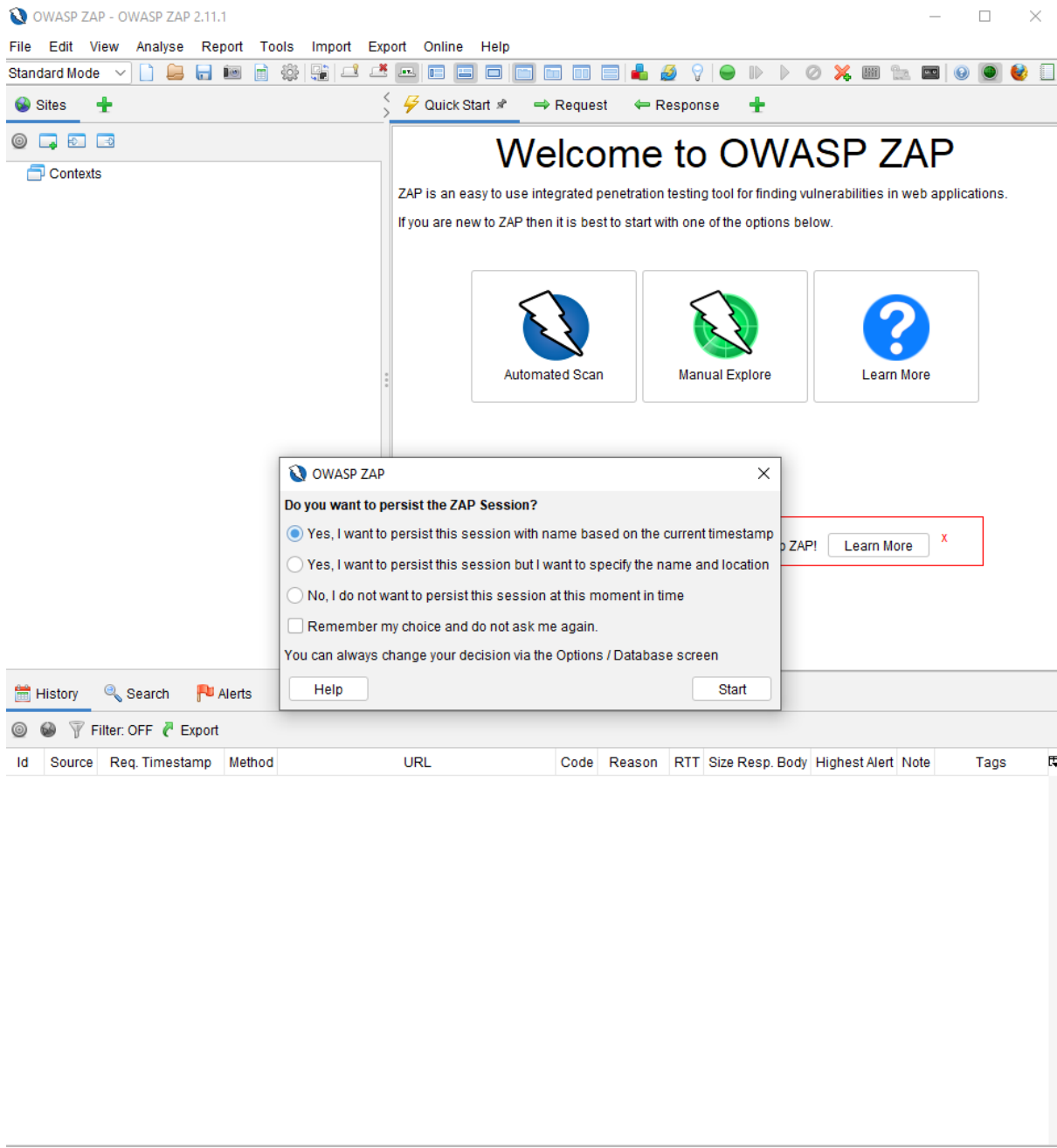
- Most of the files contain the default set of functionality, and you can add more functionality at any time via the [ZAP Marketplace](#).
- The core package contains the minimal set of functionality you need to get you started.
- The Windows and Linux versions require Java 8 or higher to run.
- The macOS version includes Java 11 - you can use the Linux or Cross Platform versions if you do not want to download this.
- The installers are built using a [multi-platform installer builder](#) which provides an [unattended mode](#).
- For more information about this release see the [release notes](#).

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

## 2. Persisting a session

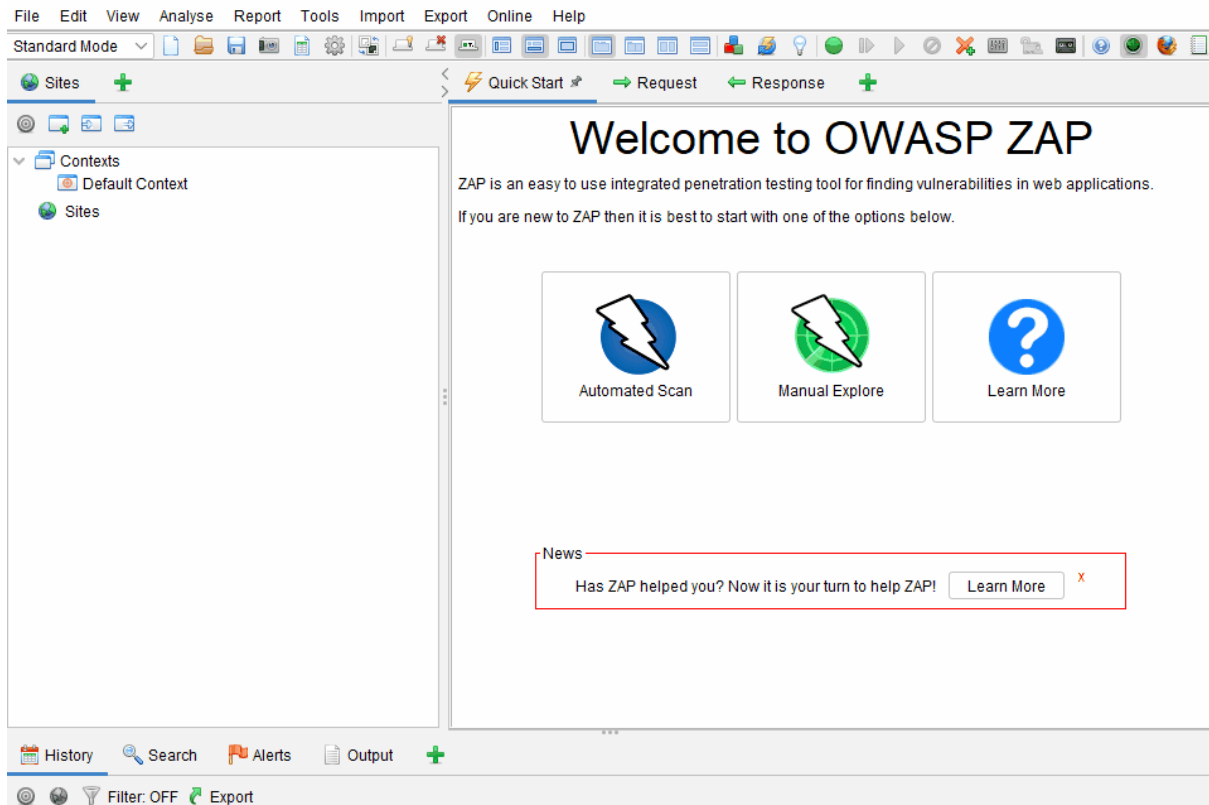
**Persisting a session in OWASP ZAP means that the session will be saved and can be reopened at a later time.** This is useful if you want to continue testing a website or application at a later time.

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.



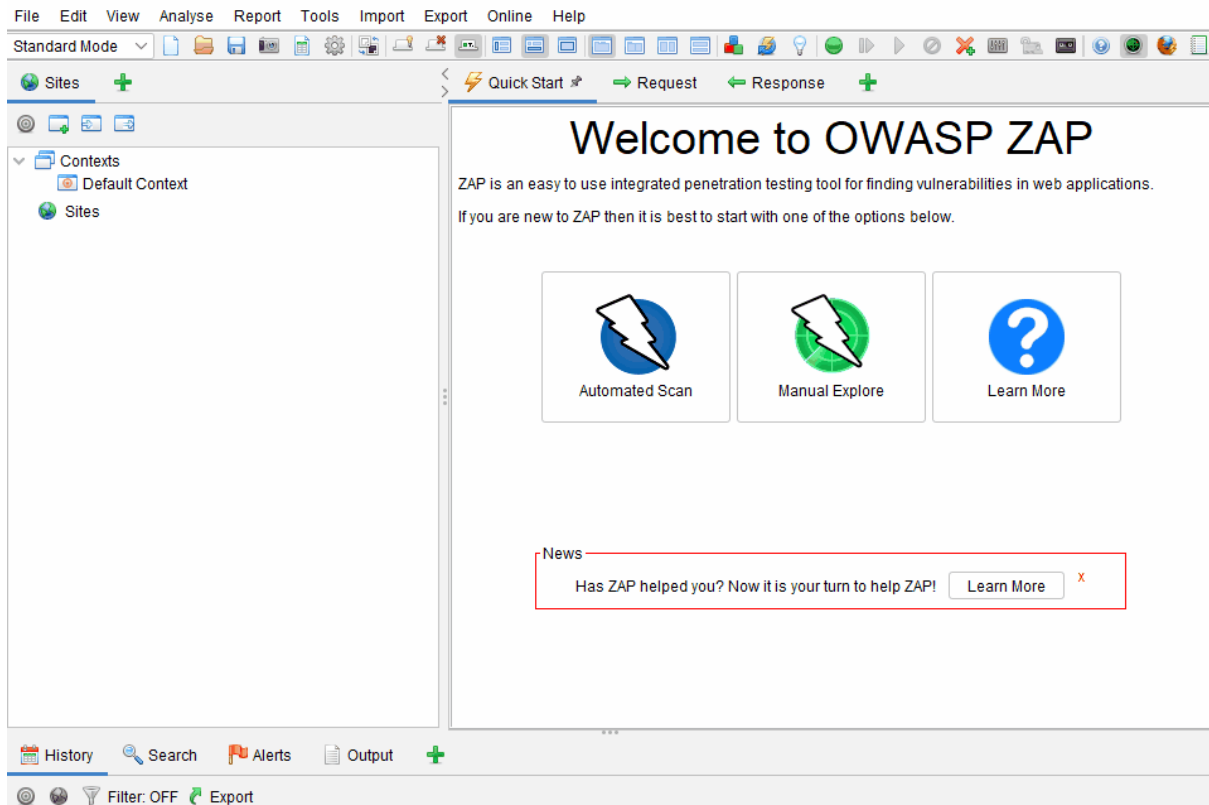
### 3. Running an automated scan

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.

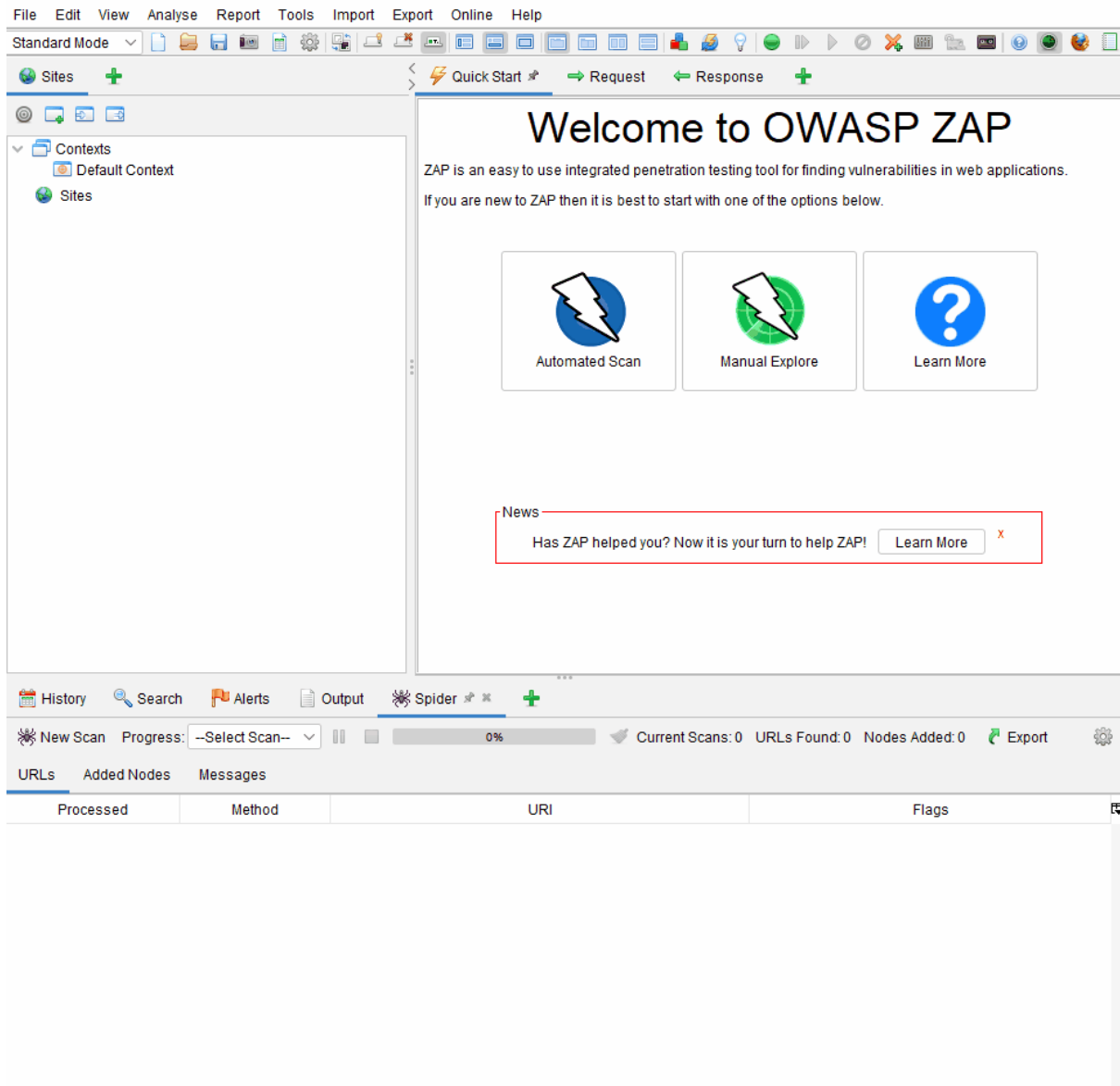
With OWASP ZAP, you can use a ZAP spider or the AJAX spider. So what's the difference?

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.



The screenshot displays the OWASP ZAP web application scanner interface. The main window is titled "Automated Scan" and contains instructions for launching a scan. The "URL to attack" field is populated with "http://google-gruyere.appspot.com/". The "Use traditional spider" checkbox is checked, and the "Use ajax spider" checkbox is also checked, with "Firefox Headless" selected as the engine. The "Attack" button is visible, and the "Progress" status shows "Actively scanning (attacking) the URLs discovered by the spider(s)".

Below the main window, the "Spider" tab is active, showing a progress bar at 100%. The "Current Scans" section indicates 0 scans, 158 URLs found, and 118 nodes added. The "Export" button is visible. The "URLs" tab is selected, displaying a table of discovered URLs.

Processed	Method	URI	Flags
●	GET	http://google-gruyere.appspot.com/5724281281280560729885...	
●	GET	http://google-gruyere.appspot.com/5724281281280560729885...	
●	GET	https://google-gruyere.appspot.com/static/codeindex/html	
●	GET	https://google-gruyere.appspot.com/static/codeindex.html	
●	GET	https://google-gruyere.appspot.com/code/resources/error.gtl	
●	GET	https://google-gruyere.appspot.com/code/sanitize.py	
●	GET	https://google-gruyere.appspot.com/code/resources/home.gtl	
●	GET	https://google-gruyere.appspot.com/code/gtl.py	
●	GET	https://google-gruyere.appspot.com/5724281281280560729885...	
●	GET	https://google-gruyere.appspot.com/resetbutton/57242812812...	
●	GET	https://google-gruyere.appspot.com/code/gruyere.py	
●	GET	https://google-gruyere.appspot.com/code/data.py	
●	GET	https://google-gruyere.appspot.com/code/resources/feed.gtl	

## 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**.

The screenshot shows the OWASP ZAP interface with the 'Automated Scan' window open. The scan has completed, showing a progress of 100%. The URL to attack is 'http://google-gruyere.appspot.com/'. The scan used a traditional spider and an ajax spider with Firefox Headless. The progress bar indicates 'Attack complete - see the Alerts tab for details of any issues found'.

Below the scan window, the 'Sent Messages' tab is active, displaying a list of HTTP requests and responses. The table below contains the data from this tab.

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"})
```

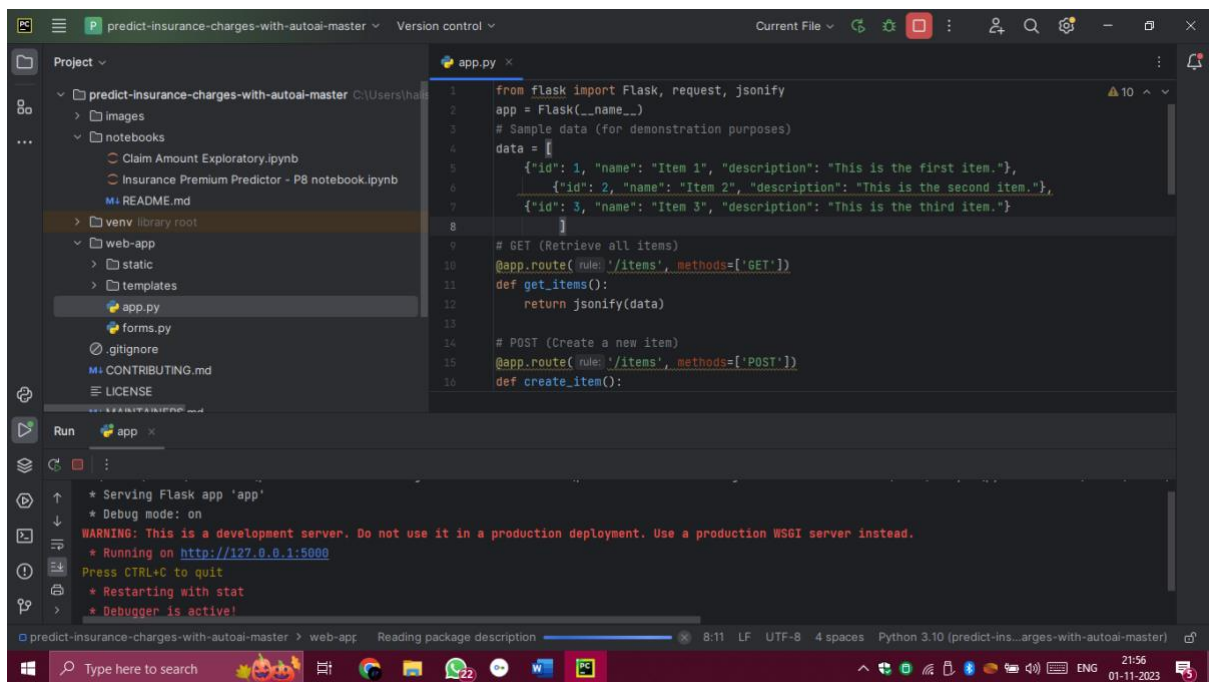
```
    else:
```

```
        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:

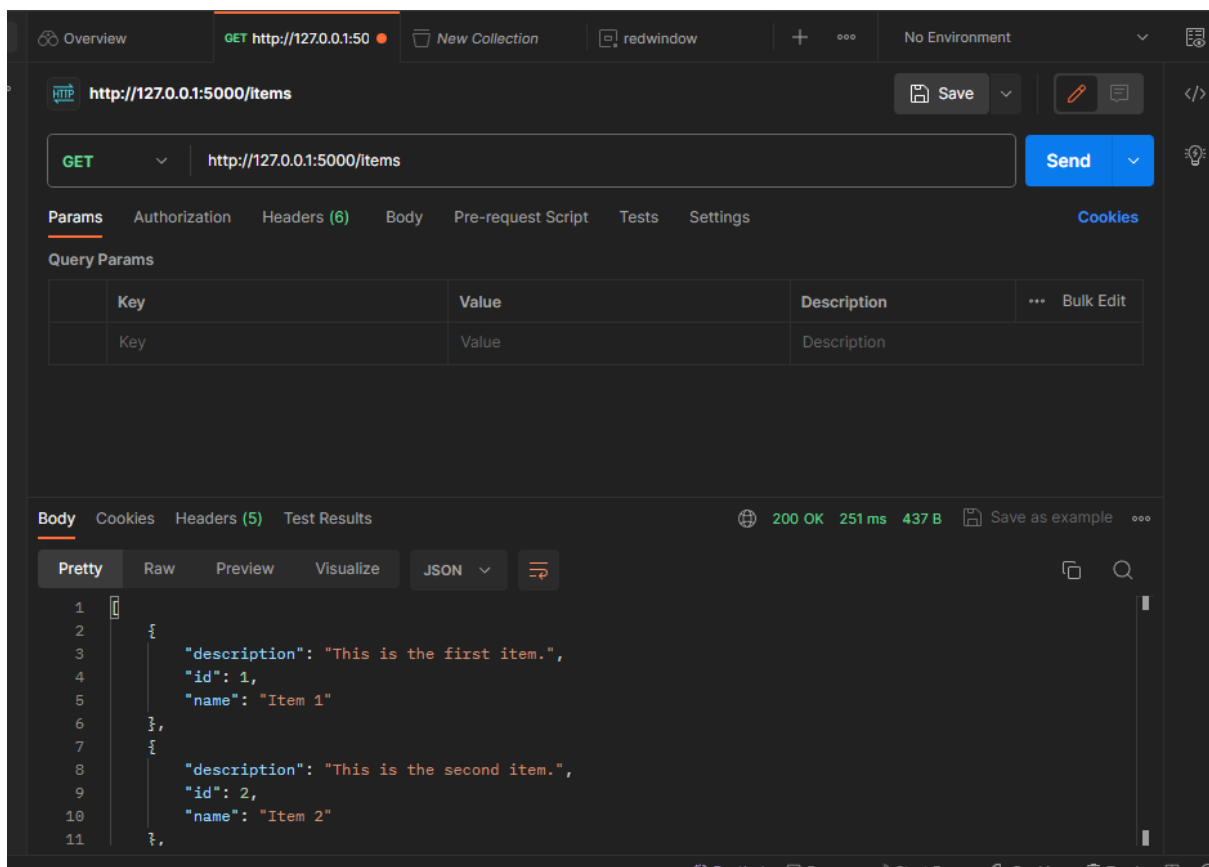


The screenshot shows a VS Code editor with a project named 'predict-insurance-charges-with-autoai-master'. The file explorer on the left shows the project structure, including a 'web-app' directory containing 'app.py'. The main editor window displays the code for 'app.py', which is a Flask application. The code defines a Flask app, sets sample data, and implements two routes: a GET route for '/items' and a POST route for '/items'. The terminal at the bottom shows the output of running the application, indicating it is serving on http://127.0.0.1:5000.

```
1 from flask import Flask, request, jsonify
2 app = Flask(__name__)
3 # Sample data (for demonstration purposes)
4 data = [
5     {"id": 1, "name": "Item 1", "description": "This is the first item."},
6     {"id": 2, "name": "Item 2", "description": "This is the second item."},
7     {"id": 3, "name": "Item 3", "description": "This is the third item."}
8 ]
9
10 # GET (Retrieve all items)
11 @app.route('/items', methods=['GET'])
12 def get_items():
13     return jsonify(data)
14
15 # POST (Create a new item)
16 @app.route('/items', methods=['POST'])
17 def create_item():
```

Run

```
* 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!
```



The screenshot shows the Redocly API client interface. The top bar indicates the current request is a GET to http://127.0.0.1:5000/items. The 'Params' tab is selected, showing a table for query parameters. The 'Body' tab is also visible, showing the JSON response of the request. The response is a list of three items, each with an id, name, and description.

Overview GET http://127.0.0.1:5000/items New Collection redwindow + No Environment

http://127.0.0.1:5000/items Save Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 251 ms 437 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "description": "This is the first item.",
3   "id": 1,
4   "name": "Item 1"
5 },
6 {
7   "description": "This is the second item.",
8   "id": 2,
9   "name": "Item 2"
10 },
11 {
```



HTTP

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

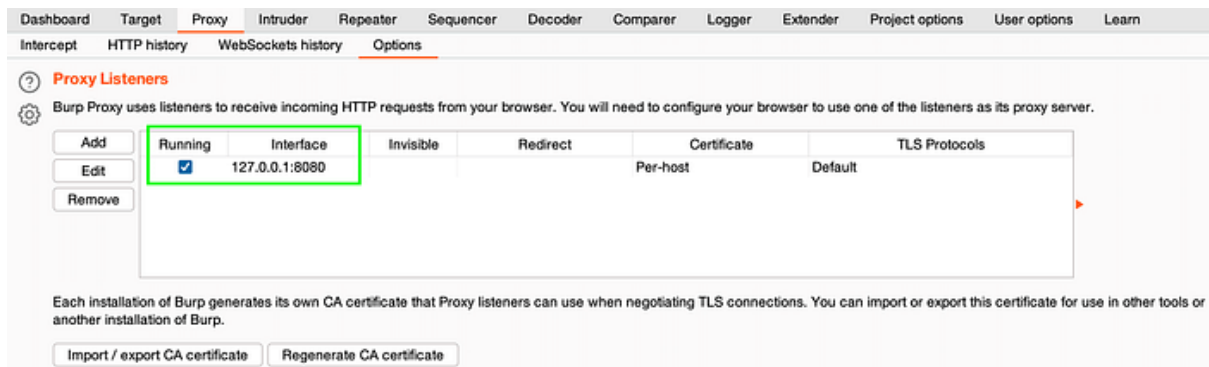
Query Params

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	id	4			
<input checked="" type="checkbox"/>	name	item 4			
<input checked="" type="checkbox"/>	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**



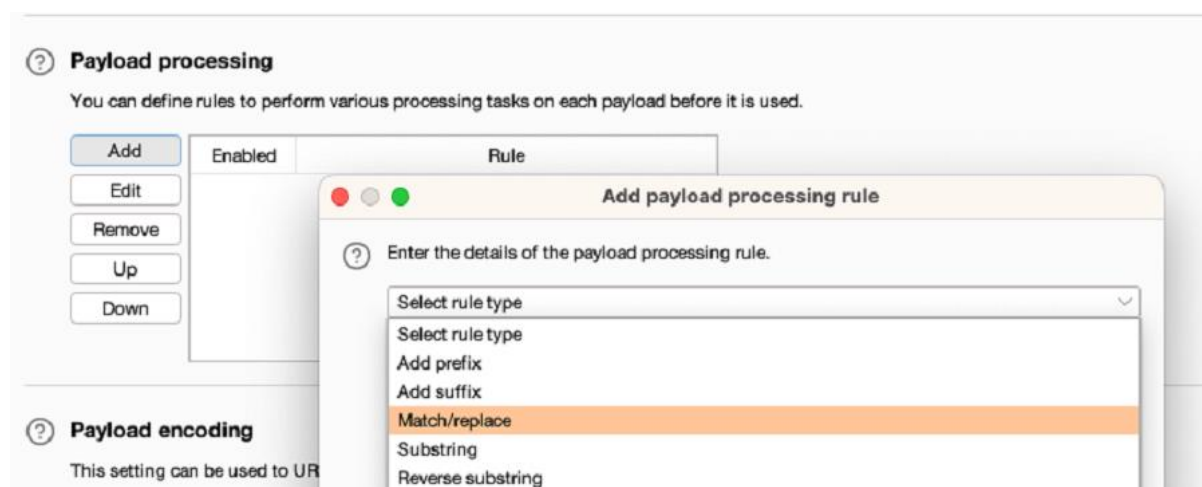
## Procedure

### 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.



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.

**Payload positions**

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target:  ☒ Update Host header to match target

1 GET /?search=**body%20\$\$=1>** HTTP/2  
 2 Host: 0a59000c0331ed4980c4efde006500d3.web-security-academy.net  
 3 Cookie: session=vgyF395fMRiyk2jZkF7tEdZ6cJFbTdp2  
 4 Sec-Ch-Ua: "Chromium";v="113", "Not-A.Brand";v="24"  
 5 Sec-Ch-Ua-Mobile: ?0  
 6 Sec-Ch-Ua-Platform: "macOS"

Buttons: Add \$, Clear \$, Auto \$, Refresh

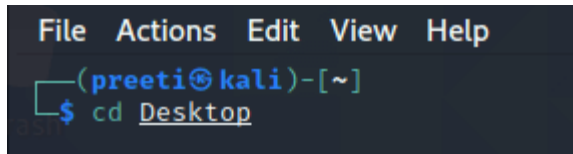
2. Go to **Intruder > Payloads**. Click **Clear** to remove the list of tags that you tested in the previous step.
3. Under **Payload settings [Simple list]** add a list of attributes that you want to test. For example, use the attributes in the [XSS cheat sheet](#).
4. Click **Start attack**. The attack 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`



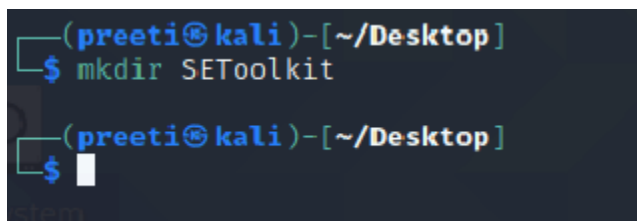
```
File Actions Edit View Help


```
(preeti@kali)-[~]
$ 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`



```
(preeti@kali)-[~/Desktop]
$ mkdir SEToolkit

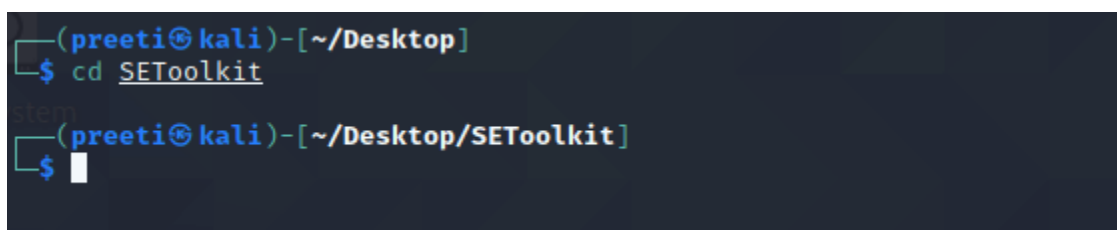


```
(preeti@kali)-[~/Desktop]
$
```


```

**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`



```
(preeti@kali)-[~/Desktop]
$ cd SEToolkit



```
(preeti@kali)-[~/Desktop/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.txt`

```
(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)) (20.0.1)
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=7ddddd3a531c35a8d26af25586fcc0a1f45377dcd1cbeb4a15e576309f682de
  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

```
(preeti@kali)-[~/Desktop/SEToolkit/setoolkit]
$ python setup.py
[*] Installing requirements.txt...
Requirement already satisfied: pexpect in /usr/lib/python3/dist-packages (from -r requirements.txt (line 1)) (4.8.0)
Requirement already satisfied: pycrypto in /home/preeti/.local/lib/python3.9/site-packages (from -r requirements.txt (line 2)) (2.6.1)
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from -r requirements.txt (line 3)) (2.25.1)
Requirement already satisfied: pyopenssl in /usr/lib/python3/dist-packages (from -r requirements.txt (line 4)) (20.0.1)
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: pymysql<3.0 in /usr/lib/python3/dist-packages (from -r requirements.txt (line 11)) (2.1.4)
[*] Installing setoolkit to /usr/share/setoolkit..
/home/preeti/Desktop/SEToolkit/setoolkit
mkdir: cannot create directory '/usr/share/setoolkit/': Permission denied
mkdir: cannot create directory '/etc/setoolkit/': File exists
cp: target '/usr/share/setoolkit/' is not a directory
cp: cannot create regular file '/etc/setoolkit/set.config': Permission denied
[*] Creating launcher for setoolkit...
Traceback (most recent call last):
  File "setup.py", line 15, in <module>
    filewrite = open("/usr/local/bin/setoolkit", "w")
IOError: [Errno 13] Permission denied: '/usr/local/bin/setoolkit'
```

**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

```
File Actions Edit View Help

!!\_____/!!\
!!  Social-Engineer Toolkit  !! \
!!          Free          !!  !
!!          #hugs         !!  !
!!        By: TrustedSec   !!  /
!!\_____/!!\
!/_/_/_/_/_/!/_/_/_/_/_/
!/_/_/_/_/_/!/_/_/_/_/_/
/0000 0000 0000 0000 /!
/ooooooooooooooooooooooooo/
/ooooooooooooooooooooooooo/
/C=_____/_/

[---] The Social-Engineer Toolkit (SET) [---]
[---] Created by: David Kennedy (ReL1K) [---]
[---] Version: 8.0.3 [---]
[---] Codename: 'Maverick' [---]
[---] Follow us on Twitter: @TrustedSec [---]
[---] Follow me on Twitter: @HackingDave [---]
[---] Homepage: https://www.trustedsec.com [---]
Welcome to the Social-Engineer Toolkit (SET).
The one stop shop for all of your SE needs.

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

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

Select from the menu:

1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> 
```

**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.

```
File Actions Edit View Help

--prettierkali: ~/Desktop/SEToolkit/setoolkit
--$ setoolkit.M""bgd `7MM""YMM MMP""MM""YMM
,MI "Y MM `7 P' MM `7
The Social-Engineer Toolkit (SET) by David Kennedy (ReL1K)
`YMMNq. MMmmMM MM
Not running a MMoolMM Y , MM
Mb dM MM ,M MM
Exiting P"Ybmmd"al.JMMmmmmMMool.JMML:ET).

[---] The Social-Engineer Toolkit (SET) [---]
[---] Created by: David Kennedy (ReL1K) [---]
Thank you for using Version: 8.0.3 Social-Engineer Toolkit.
Codename: 'Maverick'
[---] the Git Follow us on Twitter: @TrustedSec with more than 1[---]handshakes.
[---] Follow me on Twitter: @HackingDave [---]
[---] Homepage: https://www.trustedsec.com [---]
--prettierkali Welcome to the Social-Engineer Toolkit (SET).
--$ setoolkit The one stop shop for all of your SE needs.

The Social-Engineer Toolkit is a product of TrustedSec. (K)

Not running Visit: https://www.trustedsec.com

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

Select from the menu: with the Social-Engineer Toolkit.

Ha1) Spear-Phishing Attack Vectors are worth more than handshakes.
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listenerkit/setoolkit
5) Mass Mailer Attack SET
Rea6) Arduino-Based Attack Vector
Bus7) Wireless Access Point Attack Vector
Rea8) QRCode Generator Attack Vector
E: 9) Powershell Attack Vectors
10) Third Party Modules

--prettierkali ~/Desktop/SEToolkit/setoolkit
99) Return back to the main menu.
Reading package lists... Done
set> [ ] g dependency tree... Done
--prettierkali ~/Desktop/SEToolkit/setoolkit
```

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



The **Multi-Attack** method will add a combination of attacks through the web attack menu. For example you can utilize the Java Applet, Metasploit Browser, Credential Harvester/Tabnabbing all at once to see which is successful.

Reading state information... Done

The **HTA Attack** method will allow you to clone a site and perform powershell injection through HTA files which can be used for Windows-based powershell exploitation through the browser.

~/Desktop/SEToolkit/setoolkit

python3 install SEToolkit

- 1) Java Applet Attack Method
- 2) Metasploit Browser Exploit Method
- 3) Credential Harvester Attack Method
- 4) Tabnabbing Attack Method
- 5) Web Jacking Attack Method
- 6) Multi-Attack Web Method
- 7) HTA Attack Method

sh: no such file or directory: https://github.com/trustedsec/ptf

99) Return to Main Menu

~/Desktop/SEToolkit/setoolkit

set:webattack>

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

1. option 1

```
set:webattack>3

The first method will allow SET to import a list of pre-defined web
applications that it can utilize within the attack.
Hack the Gibson... and remember... hugs are worth more than handshakes.
The second method will completely clone a website of your choosing
and allow you to utilize the attack vectors within the completely
same web application you were attempting to clone.

The third method allows you to import your own website, note that you
should only have an index.html when using the import website
functionality.
Not running as root.
  1) Web Templates
  2) Site Cloner -Engineer Toolkit (SET).
  3) Custom Import

  99) Return to Webattack Menu
Thank you for using SET with the Social-Engineer Toolkit.
set:webattack>1
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a rep
ort
set:webattack>
set:webattack> ~/Desktop/SEToolkit/setoolkit

--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---
Building dependency tree... Done
The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:
install SEToolkit
Reading package lists... Done
If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.
set:webattack> ~/Desktop/SEToolkit/setoolkit
set:webattack> IP address for the POST back in Harvester/Tabnabbing [10.0.2.15]:
```

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

```

File  Actions  Edit  View  Help

3) Custom Import ~/Desktop/SEToolkit/setoolkit)

99) Return to Webattack Menu
The Social-Engineer Toolkit (SET) - by David Kennedy (ReL1K)
set:webattack>1
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a report
ing the Social-Engineer Toolkit (SET).

--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---
Thank you for using SET with the Social-Engineer Toolkit.
The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:
set:webattack>1 ~/Desktop/SEToolkit/setoolkit)

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [10.0.2.15]:

on handshakes.

**** Important Information ****

For templates, when a POST is initiated to harvest
credentials, you will need a site for it to redirect.
Reading package lists... Done
You can configure this option under:
Reading state information... Done
# vim /etc/setoolkit/set.config

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

set:webattack>1 ~/Desktop/SEToolkit/setoolkit)
1. Java Required
2. Google file or directory: https://github.com/trustedsec/ptf
3. Twitter
set:webattack>1 ~/Desktop/SEToolkit/setoolkit)
set:webattack> Select a template:

```

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

```

set:webattack> IP address for the POST back in Harvester/Tabnabbing [10.0.2.15]:
root@kali: ~/Desktop/SEToolkit/setoolkit

**** Important Information ****
The Social-Engineer Toolkit (SET) - by David Kennedy (ReL1k)
For templates, when a POST is initiated to harvest
credentials, you will need a site for it to redirect.

You can configure this option under: (SET).

/etc/setoolkit/set.config

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

root@kali: ~/Desktop/SEToolkit/setoolkit

reading package lists... Done
1. Java Required tree... Done
2. Google information... Done
3. Twitter locate package SET

set:webattack> Select a template:2
root@kali: ~/Desktop/SEToolkit/setoolkit
[*] Cloning the website: http://www.google.com
[*] This could take a little bit...
reading state information... Done
The best way to use this attack is if username and password form fields are available. R
egardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
root@kali: ~/Desktop/SEToolkit/setoolkit

```

**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.

