**Software Security Testing Tools Research**

**OWASP ZAProxy Tool**
**Zed Attack Proxy**

## Introduction

OWASP ZAP (Zed Attack Proxy) is a free security tool created and maintained by the OWASP foundation. It is available for both Windows, Linux and macOS. ZAP offers a variety of functionality: It has several scanners that can crawl your webpage and detect possible vulnerabilities, it has fuzzers that help you generate and send illegal requests, and a whole lot more. But the most beloved functionality is probably the man-in-the-middle proxy. A proxy is the most useful tool in any security tester's toolbox, as a proxy allows you to look at all traffic between the web server and your browser. You can even pause, resend or change requests! Let's have a closer look.

## Overview

This document is intended to serve as a basic introduction for using OWASP's Zed Attack Proxy (ZAP) tool to perform security testing, even if you don't have a background in security testing. To that end, some security testing concepts, and terminology is included but this document is not intended to be a comprehensive guide to either ZAP or security testing.

## Security Testing Basics

Software security testing is the process of assessing and testing a system to discover security risks and vulnerabilities of the system and its data. There is no universal terminology but for our purposes, we define assessments as the analysis and discovery of vulnerabilities without attempting to actually exploit those vulnerabilities. We define testing as the discovery and attempted exploitation of vulnerabilities. Security testing is often broken out, somewhat arbitrarily, according to either the type of vulnerability being tested, or the type of testing being done. A common breakout is:

- Vulnerability Assessment – The system is scanned and analyzed for security issues.
- Penetration Testing – The system undergoes analysis and attack from simulated malicious attackers.
- Runtime Testing – The system undergoes analysis and security testing from an enduser.
- Code Review – The system code undergoes a detailed review and analysis looking specifically for security vulnerabilities.

Note that risk assessment, which is commonly listed as part of security testing, is not included in this list. That is because a risk assessment is not actually a test but rather the analysis of the perceived severity of different risks (software security, personnel security, hardware security, etc.) and any mitigation steps for those risks.

## More About Penetration Testing

Penetration Testing (pentesting) is carried out as if the tester was a malicious external attacker with a goal of breaking into the system and either stealing data or carrying out some sort of denial-of-service attack.

Pentesting has the advantage of being more accurate because it has fewer false positives (results that report a vulnerability that isn't actually present), but can be time-consuming to run.

Pentesting is also used to test defence mechanisms, verify response plans, and confirm security policy adherence.

Automated pentesting is an important part of continuous integration validation. It helps to uncover new vulnerabilities as well as regressions for previous vulnerabilities in an environment which quickly changes, and for which the development may be highly collaborative and distributed.

## The Pentesting Process

Both manual and automated pentesting are used, often in conjunction, to test everything from servers, to networks, to devices, to endpoints. This document focuses on web application or web site pentesting.

Pentesting usually follows these stages:

- Explore – The tester attempts to learn about the system being tested. This includes trying to determine what software is in use, what endpoints exist, what patches are installed, etc. It also includes searching the site for hidden content, known vulnerabilities, and other indications of weakness.
- Attack – The tester attempts to exploit the known or suspected vulnerabilities to prove they exist.
- Report – The tester reports back the results of their testing, including the vulnerabilities, how they exploited them and how difficult the exploits were, and the severity of the exploitation.

## Pentesting Goals

The ultimate goal of pentesting is to search for vulnerabilities so that these vulnerabilities can be addressed. It can also verify that a system is not vulnerable to a known class or specific defect; or, in the case of vulnerabilities that have been reported as fixed, verify that the system is no longer vulnerable to that defect.

## Introducing ZAP

Zed Attack Proxy (ZAP) is a free, open-source penetration testing tool being maintained under the umbrella of the Open Web Application Security Project (OWASP). ZAP is designed specifically for testing web applications and is both flexible and extensible.

At its core, ZAP is what is known as a “man-in-the-middle proxy.” It stands between the tester’s browser and the web application so that it can intercept and inspect messages sent between browser and web application, modify the contents if needed, and then forward those packets on to the destination. It can be used as a stand-alone application, and as a daemon process.


Figure 1: ZAP act as proxy.

If there is another network proxy already in use, as in many corporate environments, ZAP can be configured to connect to that proxy.

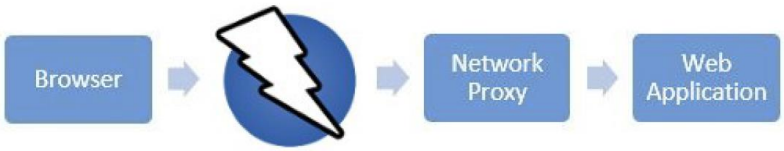


Figure 2: ZAP can be connected to other Network proxies.

ZAP provides functionality for a range of skill levels – from developers to testers new to security testing, to security testing specialists. ZAP has versions for each major OS and Docker, so you are not tied to a single OS. Additional functionality is freely available from a variety of add-ons in the ZAP Marketplace, accessible from within the ZAP client.
Because ZAP is open source, the source code can be examined to see exactly how the functionality is implemented. Anyone can volunteer to work on ZAP, fix bugs, add features, create pull requests to pull fixes into the project, and author add-ons to support specialized situations.

**ZAP advantages:**

- Web App Testing Tool.
- Zap provides cross-platform i.e. it works across all OS (Linux, Mac, Windows).
- Can test apps actively or passively.
- Zap is reusable.
- Can generate reports.
- Ideal for beginners
- Open source and Free tool.
- It can be used in CI/CD for automated testing.
- Support collaboration and works very well with other tools e.g. Burp Suite.

**About OWASP**

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications and APIs that can be trusted.

**How Does ZAP Work?**

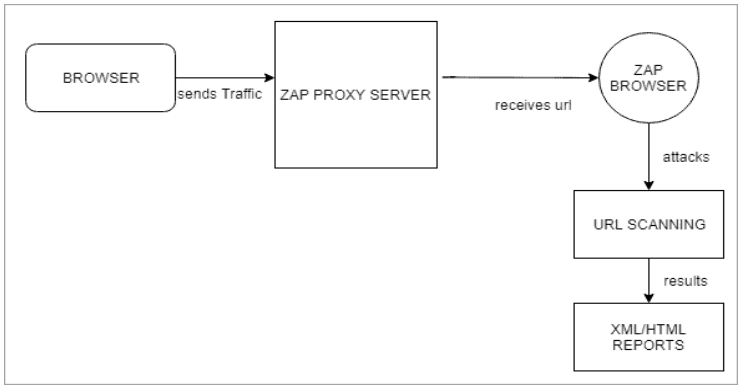


Figure 3: Building blocks for how ZAP testing works.

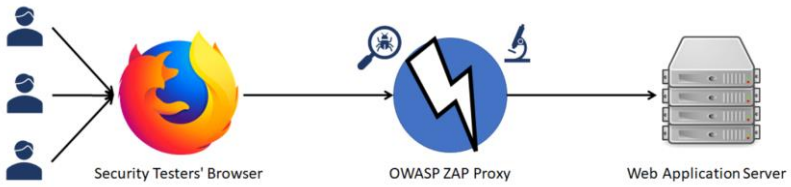


Figure 4: ZAP in Security testers browsers.

ZAP creates a proxy server and makes the website traffic pass through the server. The use of auto scanners in ZAP helps to intercept the vulnerabilities on the website.

- Intercepting proxy.
- Active and Passive Scanners.
- Traditional and Ajax Spiders.
- Brute Force Scanner.
- Port Scanner.
- Web Sockets.

**Main Features Overview**

**8 Key Concepts and Features of the ZAP Scanner**

1. **Active Scan**

   Active scanning uses known attacks to identify potential vulnerabilities, which means it can only find specific vulnerabilities. Active or automatic vulnerability scans cannot find errors in application logic. It is possible only while conducting a manual audit.

2. **Passive Scan**

   By default, ZAP scans HTTP requests and all responses sent and received from your application. Passive scanning does not affect their content. You can add tags or alerts to inform you about any potential errors. While this option is enabled by default, it is also possible to configure it.

3. **OWASP ZAP Fuzzer**

   OWASP ZAP lets you use a fuzzer that sends many unexpected or incorrect data to a tested application. You can create your own payloads, use any of the built-in payloads, or download the payloads add-ons provided by the ZAP community.

4. **OWASP ZAP API**

   OWASP ZAP provides an API that accepts JSON, XML, and HTML. The API's functionality is explained on a web page, specifying that the default allows only the machine running ZAP to connect to the API. However, you can use the configuration options to allow other machines to contact the API.

5. **WebSocket Testing**

   WebSockets are highly pervasive across applications. ZAP lets you easily intercept, analyze, or tamper with the WebSocket traffic between client and server. You can access the WebSocket Message editor by going to the ZAP Menu, choosing Tools, and then selecting WebSocket Message Editor. Use the editor to tamper with the Direction, Opcodes, or the WebSocket message before it is sent back to the server.

6. **JAX Spidering**

   You perform AJAX spidering during a penetration test to identify requests on an AJAX-rich web application. You cannot identify these requests with a standard spidering tool. You can access the AJAX spidering window by going to the menu bar, choosing Tools, and then selecting AJAX Spider (on). This tool includes configuration parameters like maximum depth to crawl, maximum duration, maximum crawl states, and other options to help you prevent infinite crawling.

7. **Scan Policy Management**
    - ZAP lets you compose a scan policy according to specific requirements for each application. You can find the Scan Policy Manager under the menu bat. Go to Analyze, and then choose Scan Policy Manager.
    - Penetration testers need to optimize this scanner according to the target application's abilities to keep up with the app's operational performance and prevent DoS. While creating scan policies, you must consider the scanner machine's bandwidth and processing capabilities.
    - You can alter a scan policy to exclude or include the test to perform. ZAP also lets you configure the definition of parameters such as:
        - Threshold—the likelihood of flagging a potential vulnerability.
        - Strength—the number of requests for each of the tests
    - Penetration testers usually need to test the scenarios they might frequently encounter across the target application. ZAP lets you design a custom scan policy that you can export as a template. You can import the scan policy back into ZAP and reuse it.
8. **ZAP Marketplace**

    ZAP Marketplace includes free and open source add-ons written by the ZAP team as well as the community. These add-ons enable you to extend the functionality of a ZAP implementation. You can browse the marketplace and download add-ons by going to the toolbar, clicking on the Manage Add-ons button, and then choosing the Marketplace tab.

**Getting started with ZAP**

First, you must download and install ZAP. When you've got ZAP up and running, it is time to configure the proxy. One of the good things about ZAP is the user guide, and you can find thorough instructions for how to configure your favorite browser to use ZAP as a proxy. If you use macOS I recommend that you do not use Chrome for this, because Chrome on macOS can not override the system proxy settings. If you change the system proxy settings to use ZAP as a proxy it will work with Chrome, but then ZAP will record all traffic to and from your machine, not just from one particular browser.

**HTTPS**

If the website you want to explore uses HTTPS (as it should), you will have to add a certificate to ZAP, and to trust this certificate in your web browser. By doing this the web server will have a secure connection over HTTPS that is terminated by the ZAP proxy, and then ZAP will set up a new HTTPS connection between its proxy and your browser. This way you will be able to see the HTTPS traffic in plain text in ZAP.

**Using the proxy**

If you have configured your browser to use ZAP as a proxy and set up the ZAP certificate correctly, you should start seeing requests and responses in ZAP when you access some webpage. At the bottom you will see a chronological view of all requests including response

codes, HTTP method, response time and size, and possible vulnerabilities (like cookies without HttpOnly flag or missing headers). ZAP will also create a map of the website under "Site". You can gather a lot of information just by monitoring the traffic. If you right-click on a request, you can choose to resend it, and you can edit it first if you want to.

### Fuzzing

Let's say you are looking at a login page and you have a list of userids, and you want to find out if any of them are valid. It is time consuming to try them all, and besides, you don't have the password. What you could try is to use a *fuzzer*. If you capture the login message with your proxy, you can right-click on it, choose "Attack" and "Fuzz". Now you can choose the username as the parameter that should be changed, add your list of potential userids, and start fuzzing! Pay attention to the response times. Does any of the messages have significantly longer response times than the rest? Chances are that this is a valid user id. This can happen if the server rejects the login request immediately if the user does not exist, and if the user id is valid, it may take some extra time checking if the password is correct. This depends on what kind of password storage is being used, but with for instance bcrypt this can be a problem.

### Install and Configure ZAP

ZAP has installers for Windows, Linux, and Mac OS/X. There are also Docker images available on the download site listed below.

### Where to get ZAP

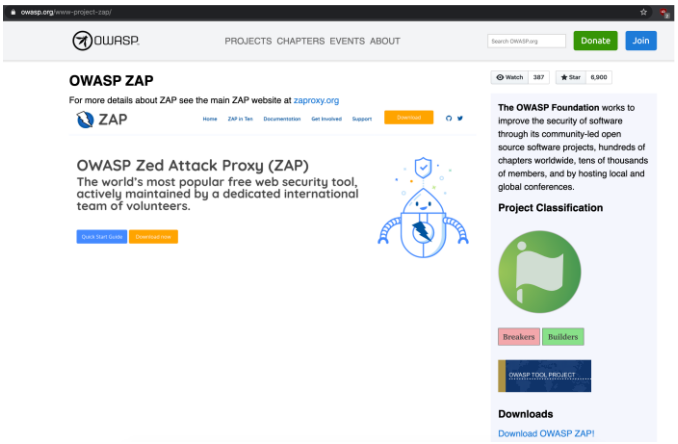


Figure 5: zaproxy.org to download ZAP

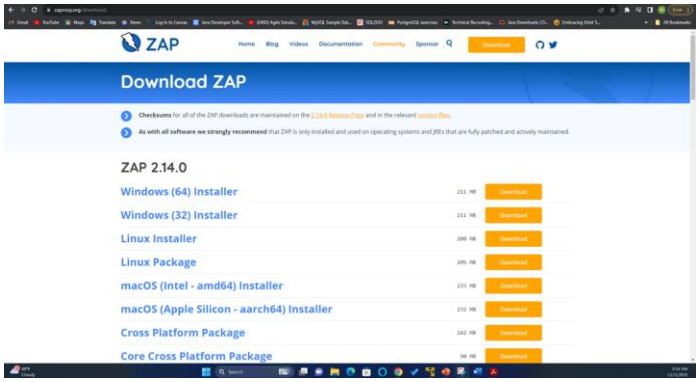


Figure 6: ZAP available for multiple operating systems.

https://www.zaproxy.org/

https://github.com/zaproxy/zaproxy

**Install ZAP**

The first thing to do is install ZAP on the system you intend to perform pentesting on. Download the appropriate installer from ZAP's download location at https://www.zaproxy.org/download/ and execute the installer.

Note that ZAP requires Java 8+ in order to run. The Mac OS/X installer includes an appropriate version of Java but you must install Java 8+ separately for Windows, Linux, and Cross-Platform versions. The Docker versions do not require you to install Java.

Once the installation is complete, launch ZAP and read the license terms. Click Agree if you accept the terms, and ZAP will finish installing, then ZAP will automatically start.

**Persisting a Session**

When you first start ZAP, you will be asked if you want to persist the ZAP session. By default, ZAP sessions are always recorded to disk in a HSQLDB database with a default name and location. If you do not persist the session, those files are deleted when you exit ZAP.

If you choose to persist a session, the session information will be saved in the local database so you can access it later, and you will be able to provide custom names and locations for saving the files.

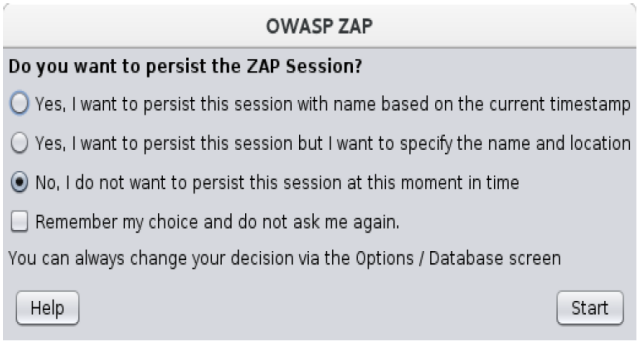


Figure 7: Persisting a Session.

For now, select **No, I do not want to persist this session at this moment in time**, then click **Start**. The ZAP sessions will not be persisted for now.

**ZAP Desktop UI**

The ZAP Desktop UI is composed of the following elements:

1. **Menu Bar** – Provides access to many of the automated and manual tools.
2. **Toolbar** – Includes buttons which provide easy access to most commonly used features.
3. **Tree Window** – Displays the Sites tree and the Scripts tree.
4. **Workspace Window** – Displays requests, responses, and scripts and allows you to edit them.
5. **Information Window** – Displays details of the automated and manual tools.
6. **Footer** – Displays a summary of the alerts found and the status of the main automated tools.

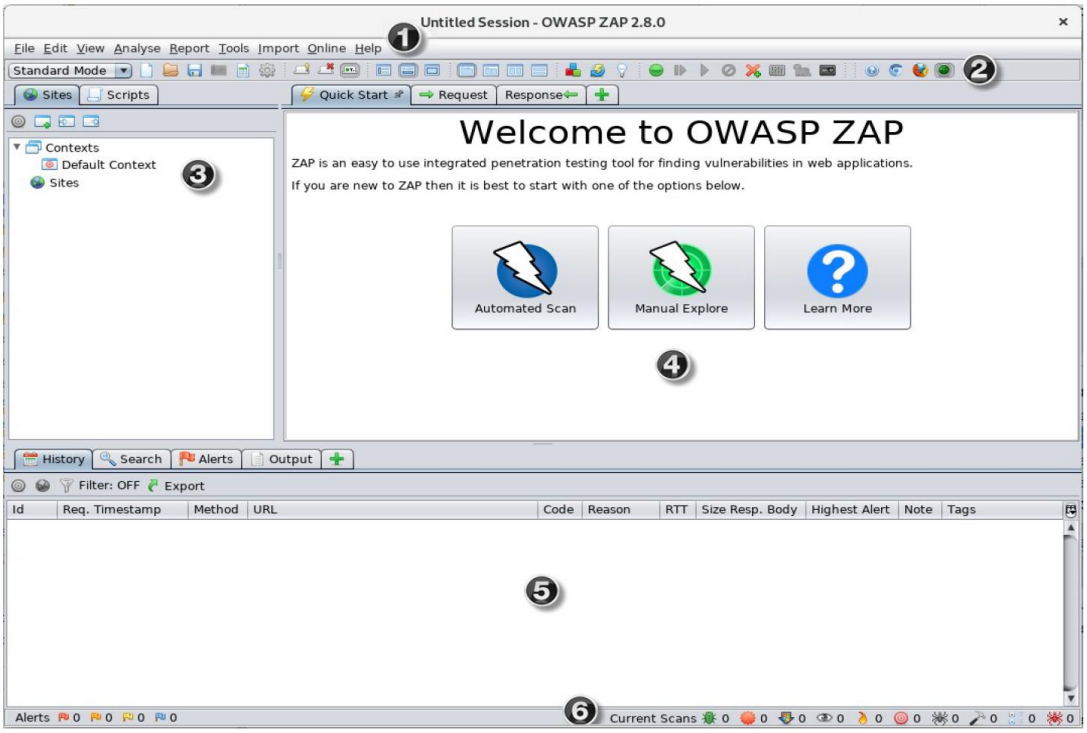


Figure 8: ZAP UI

While using ZAP, you can click **Help** on the Menu Bar or press F1 to access context-sensitive help from the ZAP Desktop User Guide. It is also available online.

For more information about the UI, see ZAP UI Overview in the ZAP online documentation.

ZAP also supports a powerful API and command line functionality, both of which are beyond the scope of this guide.

**IMPORTANT**: You should only use ZAP to attack an application you have permission to test with an active attack. Because this is a simulation that acts like a real attack, actual damage can be done to a site's functionality, data, etc. If you are worried about using ZAP, you can prevent it

from causing harm (though ZAP's functionality will be significantly reduced) by switching to safe mode.
To switch ZAP to safe mode, click the arrow on the mode dropdown on the main toolbar to expand the dropdown list and select **Safe Mode**.

**Running an Automated Scan**
The easiest way to start using ZAP is via the Quick Start tab. Quick Start is a ZAP add-on that is included automatically when you installed ZAP.
To run a Quick Start Automated Scan:

1. Start ZAP and click the **Quick Start** tab of the Workspace Window.
2. Click the large Automated Scan button.
3. In the **URL to attack** text box, enter the full URL of the web application you want to attack.
4. Click the **Attack** button.

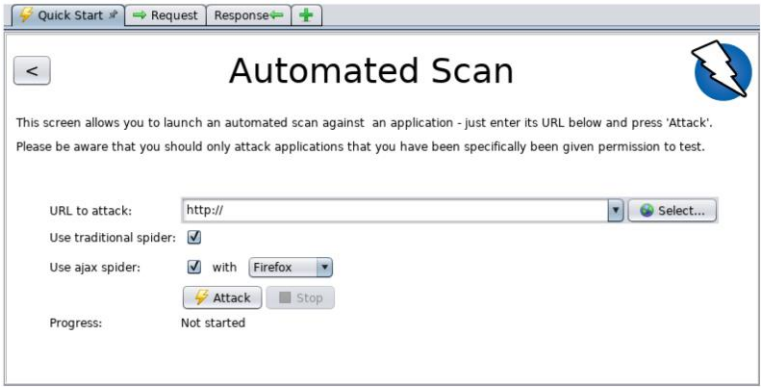


Figure 9: Automated Scan window.

ZAP will proceed to crawl 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.
ZAP provides 2 spiders for crawling web applications; you can use either or both of them from this screen.
The traditional ZAP spider which 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 that generates links using JavaScript.
For AJAX applications, ZAP's AJAX spider is likely to be more effective. 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 and requires additional configuration for use in a "headless" environment.
ZAP will passively scan all of the requests and responses proxied through it. So far ZAP has only carried out passive scans of your web application. Passive scanning does not change responses in any way and is considered safe. Scanning is also performed in a background thread to not slow down exploration. Passive scanning is good at finding some vulnerabilities and as a way to get a

feel for the basic security state of a web application and locate where more investigation may be warranted.
Active scanning, however, attempts to find other vulnerabilities by using known attacks against the selected targets. Active scanning is a real attack on those targets and can put the targets at risk, so do not use active scanning against targets you do not have permission to test.

**Interpret Your Test Results**

As ZAP spiders your web application, it constructs a map of your 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.
To examine a tree view of the explored pages, click the **Sites** tab in the Tree Window. You can expand the nodes to see the individual URLs accessed.
The left-hand side of the Footer contains a count of the Alerts found during your test, broken out into risk categories. These risk categories are:

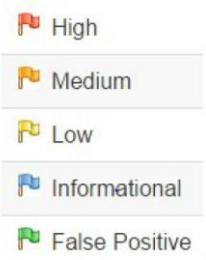


Figure 10: Alerts severity levels

To view the alerts created during your test:

1. Click the **Alerts** tab in the Information Window.
2. Click each alert displayed in that window to display the URL and the vulnerability detected in the right side of the Information Window.
3. In the Workspace Windows, click the **Response** tab to see the contents of the header and body of the response. The part of the response that generated the alert will be highlighted.

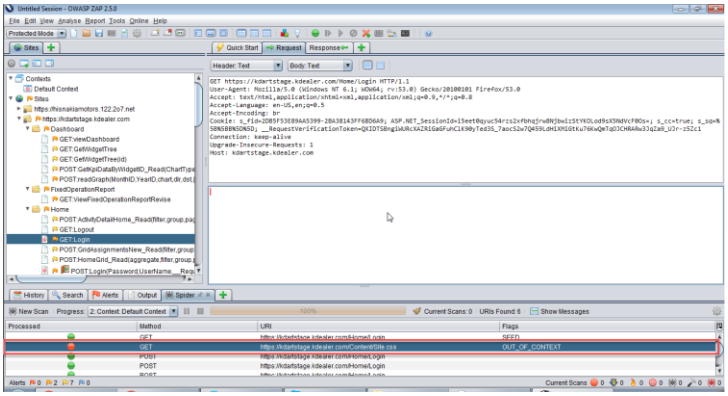

Figure 11: explore resulted alerts.

**Exploring an Application Manually**

The passive scanning and automated attack functionality is a great way to begin a vulnerability assessment of your web application but it has some limitations. Among these are:

- Any pages protected by a login page are not discoverable during a passive scan because, unless you've configured ZAP's authentication functionality, ZAP will not handle the required authentication.
- You don't have a lot of control over the sequence of exploration in a passive scan or the types of attacks carried out in an automated attack. ZAP does provide many additional options for exploration and attacks outside of passive scanning.

Spiders are a great way to explore your basic site, but they should be combined with manual exploration to be more effective. Spiders, for example, will only enter basic default data into forms in your web application but a user can enter more relevant information which can, in turn, expose more of the web application to ZAP. This is especially true with things like registration forms where a valid email address is required. The spider may enter a random string, which will cause an error. A user will be able to react to that error and supply a correctly formatted string, which may cause more of the application to be exposed when the form is submitted and accepted.

You should explore all of your web application with a browser proxying through ZAP. As you do this, 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. It is important to have ZAP explore each page of your web application, whether linked to another page or not, for vulnerabilities. Obscurity is not security, and hidden pages sometimes go live without warning or notice. So be as thorough as you can when exploring your site.

You can quickly and easily 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.

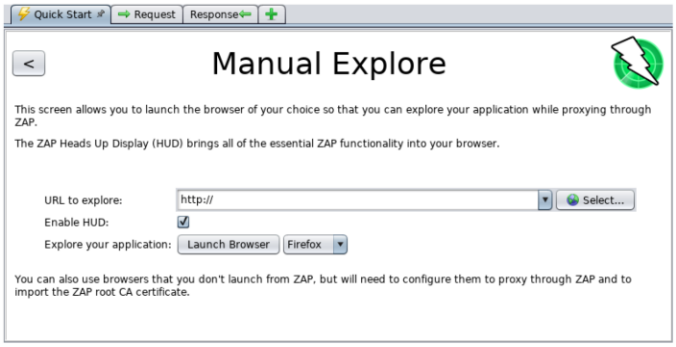


Figure 12: ZAP Manual explore.

To Manually Explore your application:

1. Start ZAP and click the **Quick Start** tab of the Workspace Window.
2. Click the large Manual Explore button.
3. In the **URL to explore** text box, enter the full URL of the web application you want to explore.

4. Select the browser you would like to use.
5. Click the **Launch Browser** button.

This option will launch any of the most common browsers that you have installed with new profiles.
If you would like to use any of your browsers with an existing profile, for example with other browser add-ons installed, then you will need to manually configure your browser to proxy via ZAP and import and trust the ZAP Root CA Certificate. See the ZAP Desktop User Guide for more details.
By default, the ZAP Heads Up Display (HUD) will be enabled. Unchecking the relevant option on this screen before launching a browser will disable the HUD.

**How ZAP attacks work**

Once you click the **Attack** button, ZAP starts crawling the web application with its spider, passively scanning each page it finds. Next, ZAP uses the active scanner to attack all discovered pages, parameters, and functionality.

ZAP includes two spiders that can crawl web applications. You can use one or both of the spiders. Here are key differences between the spiders:

- **A traditional ZAP spider**—this spider discovers links by examining the HTML in responses from a web app. It is fast but not always effective when you want to explore AJAX web applications that generate links using JavaScript.
- **ZAP's AJAX spider**—this spider is for AJAX applications. It explores a web application by invoking browsers, which then follow all the links that were generated. It is slower than the traditional spider and requires extra configuration when using it in a "headless" environment.

**Passive vs. active scanning**

ZAP passively scans all responses and requests that are proxied through it. This type of scanning does not change responses, which is why it is generally considered safe. The scanning is performed in a background thread so as not to slow down exploration.

**Automated Scans**

1. Start ZAP and click the Quick Start tab of the Workspace Window.
2. Click the large Automated Scan button.
3. In the URL to attack text box, enter the full URL of the web application you want to attack.
4. Click the Attack

**Spiders are powerful and Flexible:**

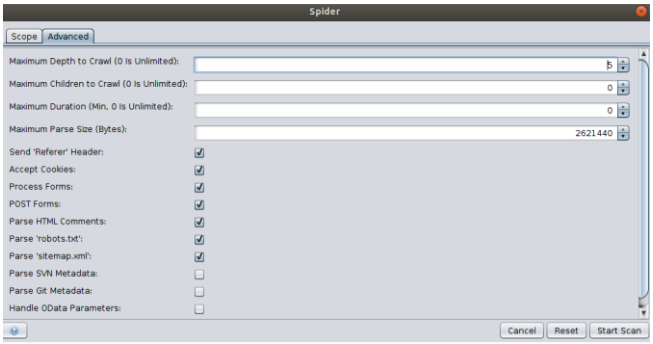


Figure 13: Setting regular spider options

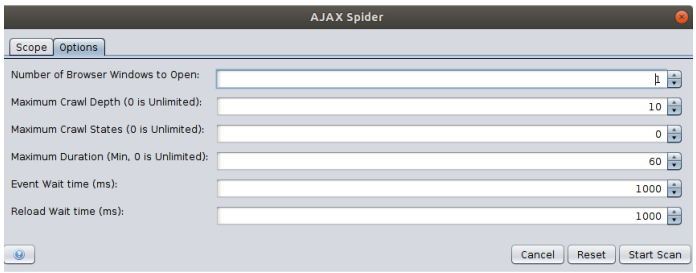


Figure 14: ZAP Ajax Spider

**Generating Reports In ZAP**

- Once the Active scan is done, we can generate reports. For that click OWASP ZAP >> Report >> generate HTML reports >> file path provided >> scan report exported. We need to examine the reports to identify all possible threats and get them fixed.

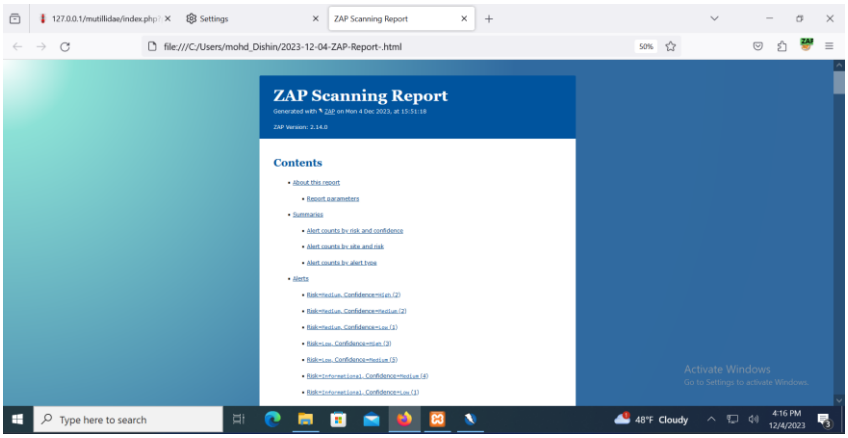


Figure 15: ZAP report.

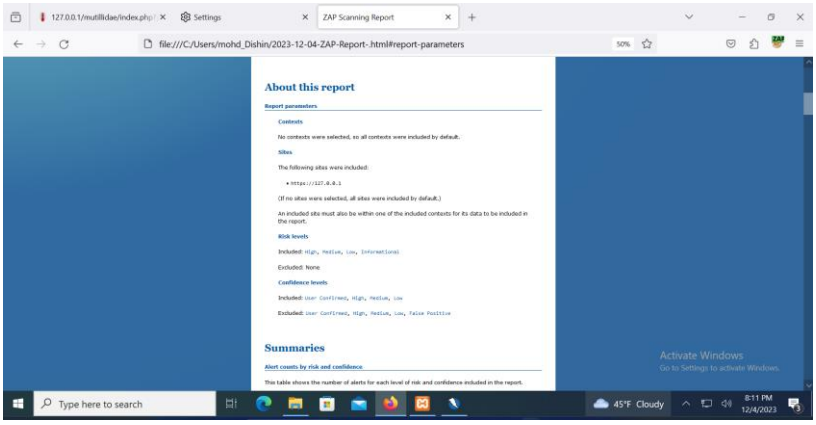


Figure 16: ZAP report details 1.

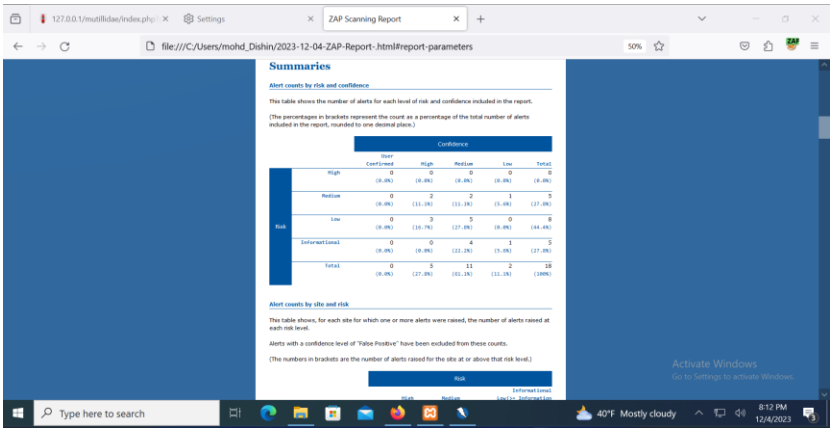


Figure 17: ZAP reports details 2.

**How ZAP attacks work**

- Once you click the Attack button, ZAP starts crawling the web application with its spider, passively scanning each page it finds. Next, ZAP uses the active scanner to attack all discovered pages, parameters, and functionality.
- ZAP includes two spiders that can crawl web applications. You can use one or both of the spiders. Here are key differences between the spiders:
- A traditional ZAP spider—this spider discovers links by examining the HTML in responses from a web app. It is fast but not always effective when you want to explore AJAX web applications that generate links using JavaScript.
- ZAP's AJAX spider—this spider is for AJAX applications. It explores a web application by invoking browsers, which then follow all the links that were generated. It is slower than the traditional spider and requires extra configuration when using it in a "headless" environment.

**Passive vs. active scanning**

ZAP passively scans all responses and requests that are proxied through it. This type of scanning does not change responses, which is why it is generally considered safe. The scanning is performed in a background thread so as not to slow down exploration.

**Active Scan**

- We can perform an Active scan using Zap in many ways. The first option is the Quick Start, which is present on the welcome page of the ZAP tool. Please refer the below screenshot
- The above screenshot shows the quickest way to get started with ZAP. Enter the URL under the Quick Start tab, press the Attack button, and then progress starts.
- Quick Start runs the spider on the specified URL and then runs the active scanner. A spider crawls on all the pages starting from the specified URL. To be more precise, the Quickstart page is like “point and shoot”.

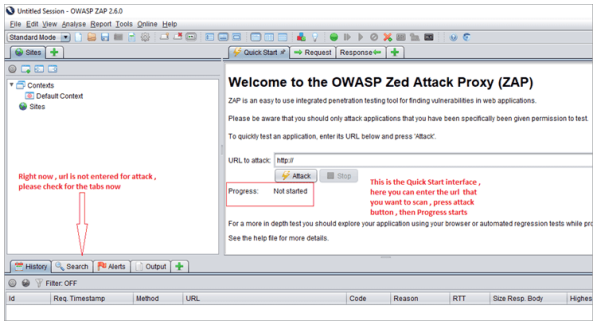


Figure 18: Start attack through quick start interface.

- Here, upon setting the target URL, the attack starts. You can see the Progress status as spidering the URL to discover content. We can manually stop the attack if it is taking too much time.
- Another option for the Active scan is that we can access the URL in the ZAP proxy browser as Zap will automatically detect it. Upon right-click on the URL -> Active scan will launch. Once the crawl is complete, the active scan will start.
- Attack progress will be displayed in the Active scan Tab. and the Spider tab will show the list URL with attack scenarios. Once the Active scan is complete, results will be displayed in the Alerts tab.

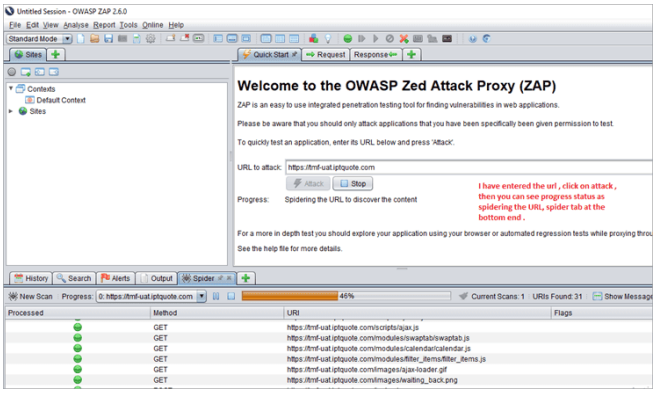


Figure 19: Automated Scan progress.

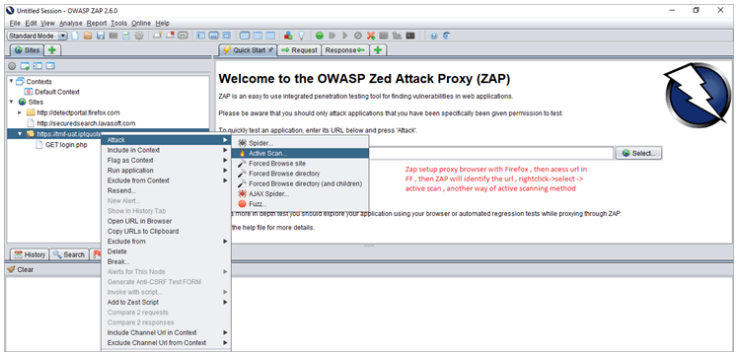


Figure 20: Start active scan by right click targeted page.

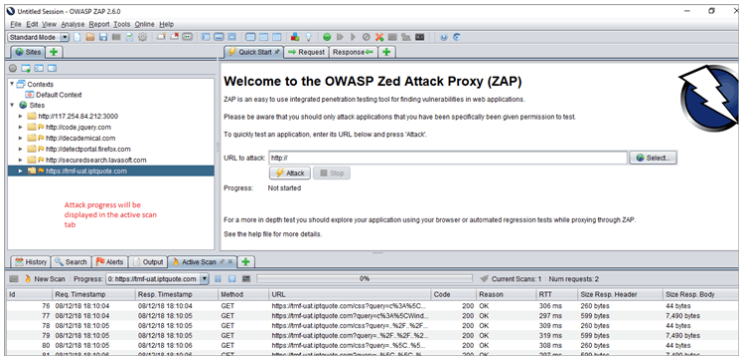


Figure 21: attack progress showed in active scan tab.

**API**

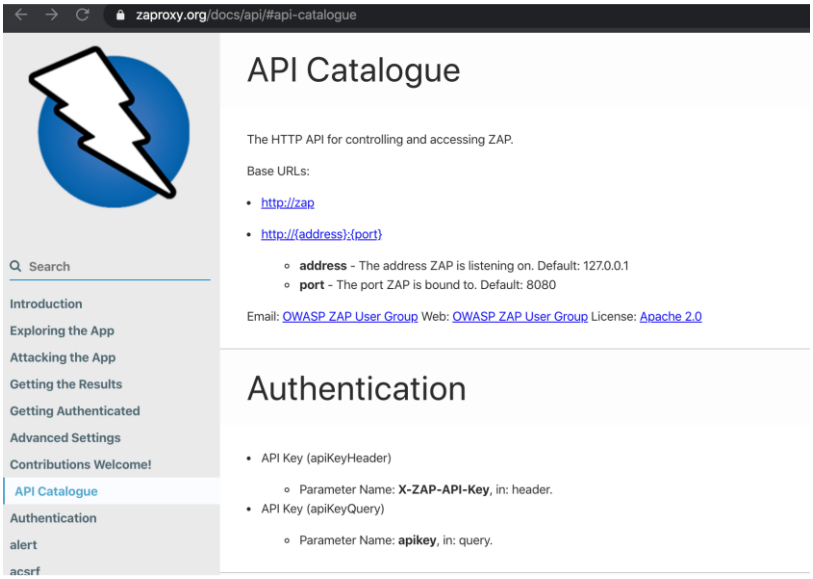


Figure 22: ZAP API https://www.zaproxy.org/docs/api/#api-catalogue

**The Heads-Up Display**

The Heads-Up Display (HUD) is a new innovative interface that provides access to ZAP functionality directly in the browser. It is ideal for people new to web security and also allows experienced penetration testers to focus on an applications functionality while providing key security information and functionality.

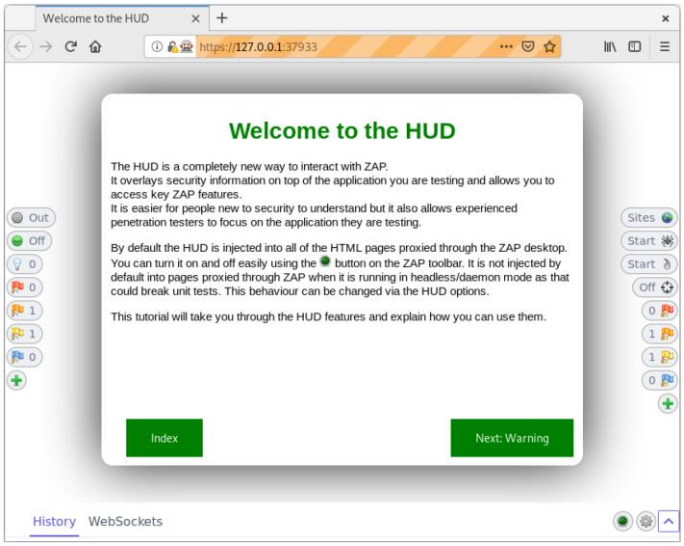


Figure 23: the Heads-up display HUD.

The HUD is overlayed on top of the target application in your browser when enabled via the 'Manual Explore' screen or toolbar option. Only modern browsers such as Firefox and Chrome are supported.
By default, a splash screen is shown for the HUD which includes a link to a tutorial which will take you through the HUD features and explain how you can use them.

**ZAP Advanced Features**
**Advanced Desktop Features**
The desktop has a large number of features that are not immediately apparent so that new users are not overwhelmed.
There are many tabs that are not shown by default. They can be accessed via the right hand tabs with green '+' icons. You can pin any tabs you would like to always appear by right clicking on them. Many of the tabs hidden by default will appear when relevant. For example, the Websockets tab will appear if an application you are proxying through ZAP starts to use Websockets.
The desktop also makes heavy use of context sensitive right click options, so right click everywhere while you are getting used to the user interface.

**The ZAP Marketplace**
The ZAP desktop has a plugin architecture which means that new functionality can be added dynamically.
An online marketplace provides a wide range of ZAP add-ons which add many additional features to ZAP.
The marketplace can be accessed via the 'Manage Add-ons' button on the toolbar:

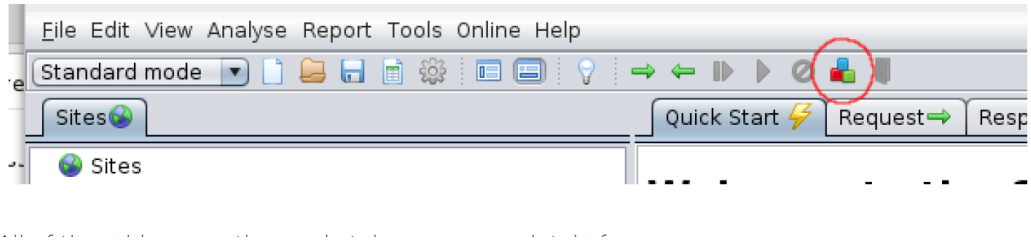


Figure 24: ZAP market place icon.

All of the add-ons on the marketplace are completely free.

**Automation**
ZAP is an ideal tool to use in automation and supports a range of options:

- Docker Packaged Scans
- GitHub Actions
- Automation Framework
- API and Daemon mode

**Learn More About ZAP**
You can learn more about ZAP's capabilities and how to use them from ZAP's Desktop U ser Guide. The User Guide provides step-by-step instructions, references for the API and command-line programming, instructional videos, and tips and tricks for using ZAP.
Additional links are also available via the 'Learn More' button on the Quick Start top screen:

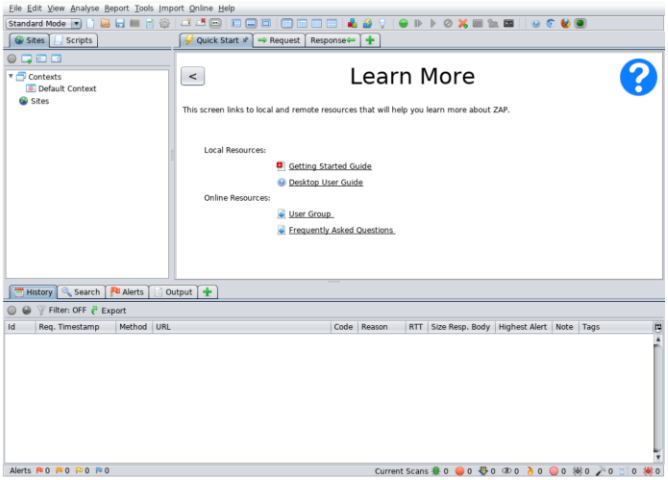

Figure 25: Learn more window.

The page links to both local resources (if available) and online content. Additional links are included below.

### ZAP Authentication, Session and User Management

Let us move on to another Zap feature, handling authentication, session and user management. Please let me know any query that comes into your mind related to this as comments.

### Basic Concepts

- **Context**: It represents a web application or set of URLs together. For a given Context, new tabs are added to customize and configure the authentication and session management process. The options are available in the session properties dialog .i.e Session properties dialog -> Context -> you can either use the default option or add a new context name.
- **Session Management Method:** There are 2 types of session management methods. Mostly, cookie-based session management is used, associated with the Context.
- **Authentication Method:** There are mainly 3 types of Auth method used by ZAP:
    - Form-based Authentication method
    - Manual Authentication
    - HTTP Authentication
- **User management:** Once the authentication scheme has been configured, a set of users can be defined for each Context. These users are used for various actions (**For Example,** Spider URL/Context as User Y, send all requests as User X). Soon, more actions will be provided that make use of the users.

A “Forced-User” extension is implemented to replace the old authentication extension that was performing re-authentication. A ‘Forced-User’ mode is now available via the toolbar (the same icon as the old authentication extension).

After setting a user as the ‘Forced-User’ for a given context or when it is enabled, every request sent through ZAP is automatically modified so that it is sent for this user. This mode also performs re-authentication automatically (especially in conjunction with the Form-Based Authentication) if there is a lack of authentication, ‘logged out’ is detected.

**Let us see a demo testing using ZAP:**

I ran multiple tests to figure out some vulnerabilities on local hosted web application provided by OWASP for testing named Mutillidae web application.

**Step 1:**

First, launch ZAP and access the URL in the proxy browser. Here, I have taken a local hosted web application provided by OWASP for testing with local hosted URL as https://127.0.0.1/mutillidae/index.php. Click on Advanced -> add Exception -> confirm security exception as in page 6 and 7. Then the landing page gets displayed. At the same time ZAP automatically loads the Webpage under Sites as a new session. Refer to the below image.

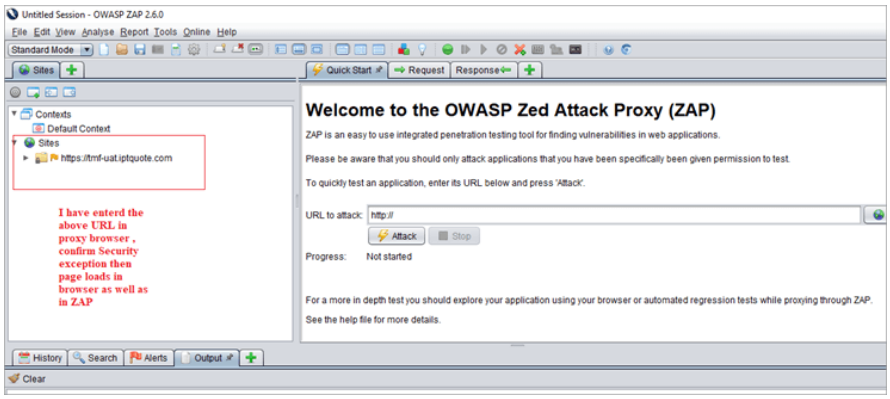


Figure 26: Session hijacking testing.

**Step 2:**

Include it in a context. This can be done either by including it in a default context or adding it as a new context. Refer to the below image.

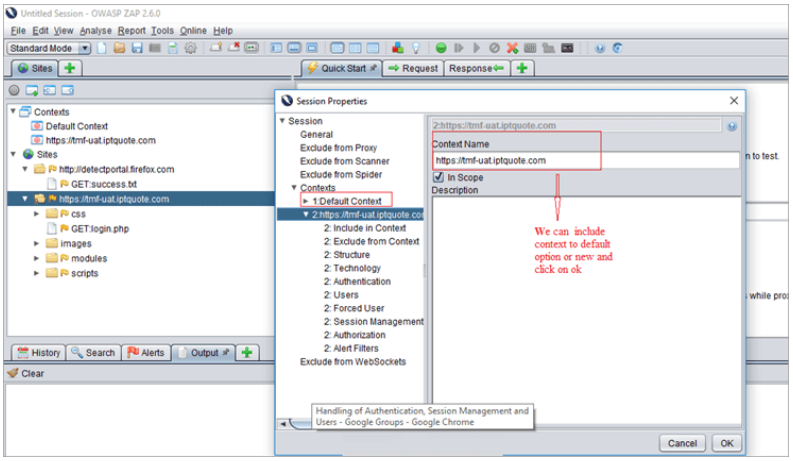


Figure 27: include some context for vulnerability testing.

**Step 3:**

Now, next is the Authentication method. You can see Authentication in that session properties dialog itself. Here we are using the Form-based Auth method.

It should be like authMethodParams as ***"login Url=https://127.0.0.1/tmf-uat.iptquote.com/login.php&loginRequestData=username=superadmin&password=primo868&proceed=login"***

In our example, we need to set the authentication method as Form-based. For this, select the target URL, login request post data field gets pre-filled, after that, change parameter as username and password -> click ok.

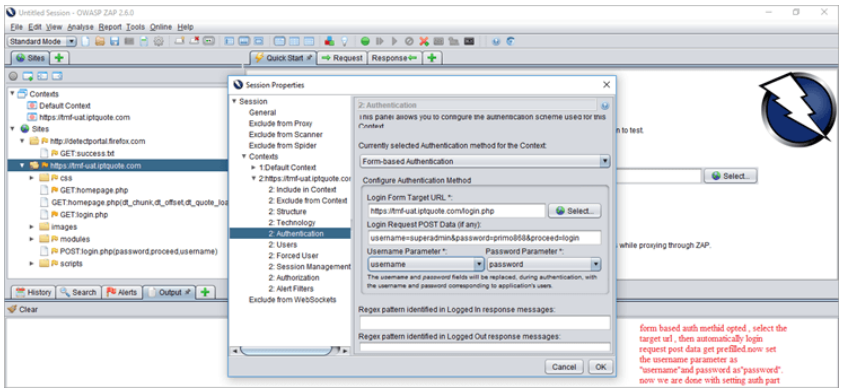

Figure 28: Authentication vulnerabilities testing.

**Step 4:**

Now, set indicators that will tell ZAP when it is authenticated.

**Logged in and logged out indicators:**

- Only one is necessary.

- We can set Regex patterns matched in the response message, need to set either logged in or log out indicator.
- Identify when a response is authenticated or when not.
- **Example for Logged in indicator:** \Qhttp://example/logout\E or Welcome User.*
- **Example of the Logged-out indicator:** login.jsp or something like that.

Here, in our demo application, I have accessed the URL in a proxy browser. Logged in to the application using a valid credential, Username as superadmin & Password as primo868. Navigate through inner pages and click on logout.

You can see in Step 3 screenshot; Zap takes the login request data as one used for the Mutillidae application login [Demo application login].

Flag logged in Regex pattern from the Response of ZAP as Response -> logged out response -> flag it as logged in the indicator.

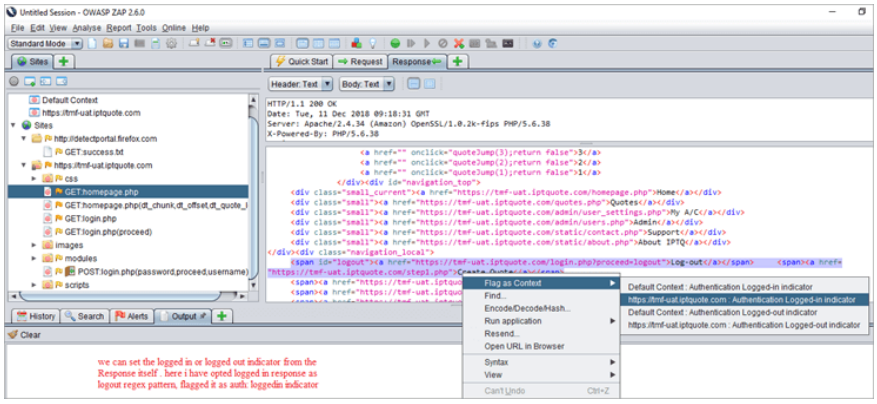


Figure 29: Access control testing.

**Step 5:**

We can save the indicator and verify whether session properties dialog gets added with the logged-in indicator or not. Refer to the screenshot below:

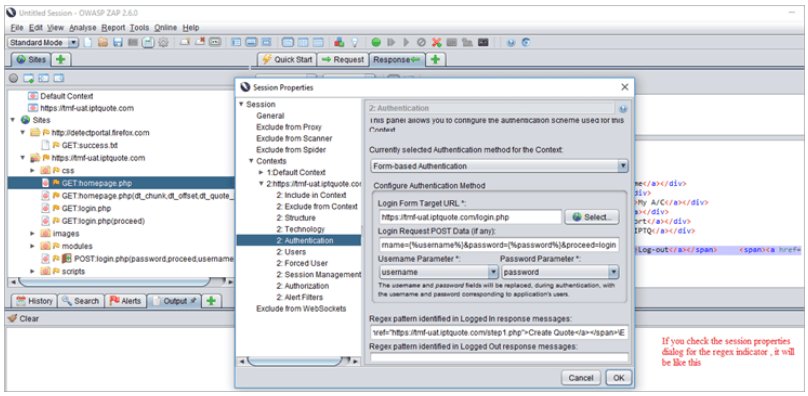


Figure 30: Session cookie testing.

**Step 6:**

We need to add users, valid and invalid users. Apply spider attacks to both and analyze the results.

**Valid User:**

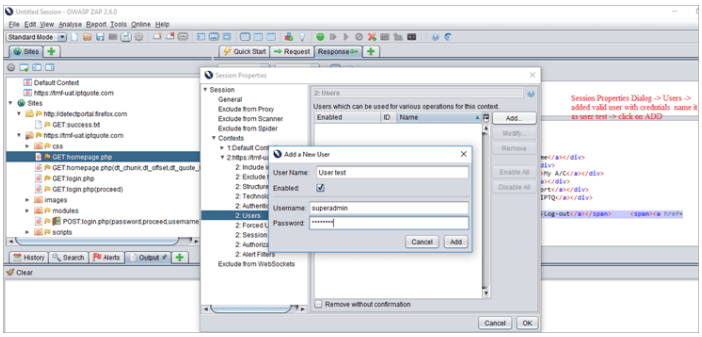

Figure 31: adding valid user.

**Invalid User:**

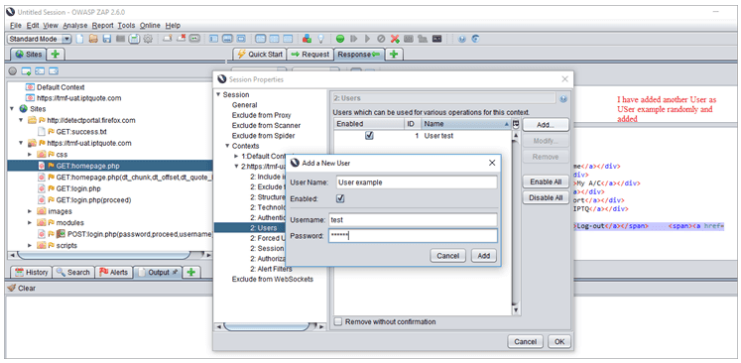

Figure 32: adding invalid user.

**Step 7:**

By default, set the session management as a cookie-based method.

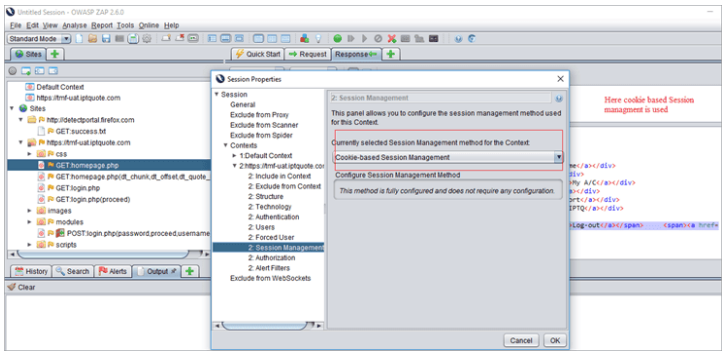

Figure 33: setting the session management as a cookie-based method.

**Step 8:**

Spider URL attack is applied to invalid and valid users and review results/generate reports.

**Invalid user spider attack view 1:**

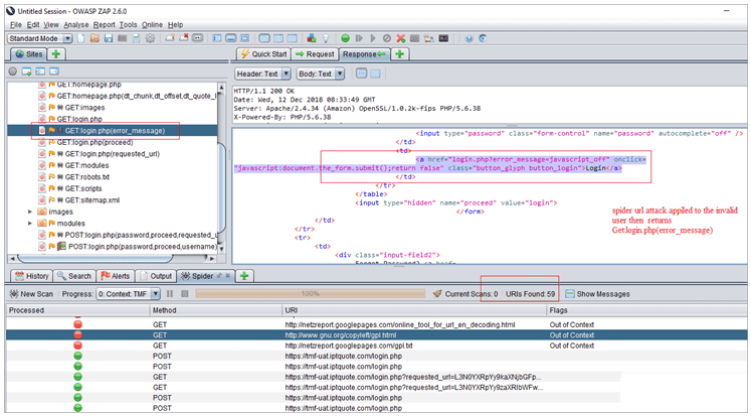


Figure 34: Spider attack applied to invalid user.

Here, a spider URL attack is applied to the invalid user. In the ZAP interface, we can see Get: login.php (error _message), which means authentication has failed. Also, it doesn't pass the URLs through inner Mutillidae pages.

**Step 9:**

To apply spider URL attack for the valid user, go to sites list -> attack -> spider URL -> existing valid user -> here it is enabled by default -> start scan.

Analyze results: As it is a valid authenticated user, it will navigate through all inner pages and display authentication status as successful. Refer below screenshot.

**Valid user:**

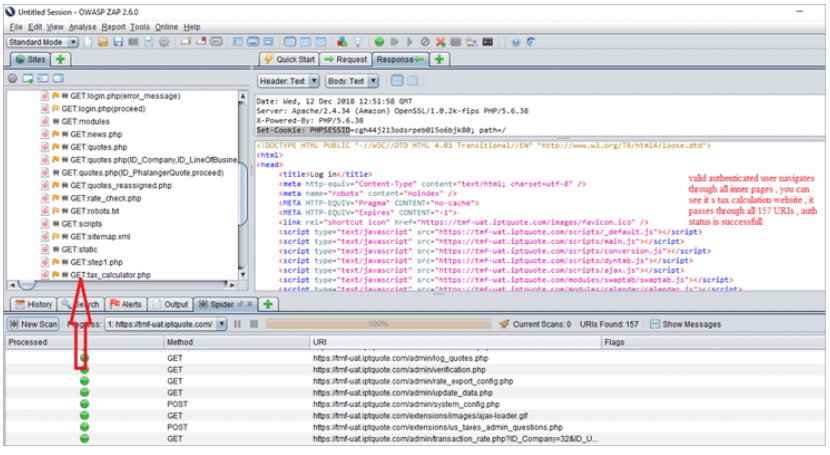


Figure 35: results for spider attack on valid user.

**ZAP Html Report for the conducted attack:**

Once an active scan is completed, we can generate an HTML report for the same. For this, select Report -> Generate Html Report. I have attached a sample content of HTML reports. Here, high, medium and low alerts reports will be generated.

**Alerts**

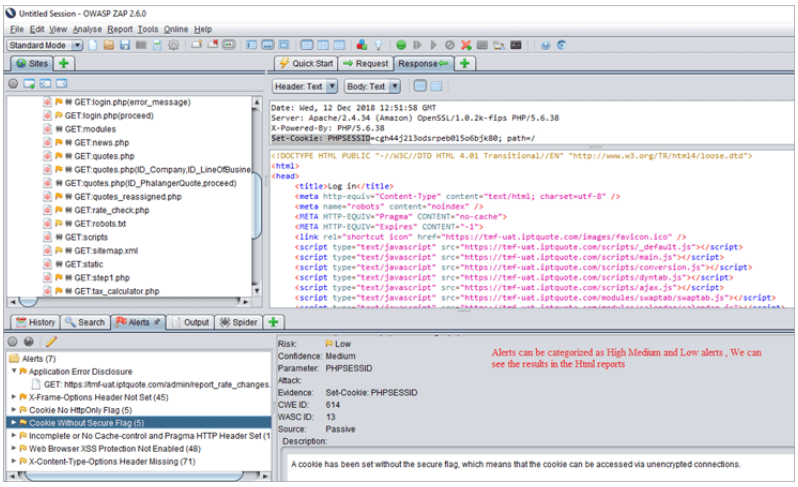


Figure 36: analyzing generated alerts.

**Conclusion**

In this Project, we have seen what ZAP is, how ZAP works, installation, and ZAP proxy setup. Different types of Active scan processes, a demo of ZAP authentication, session and user management, and basic terminologies.

**References and Useful Links:**

- OWASP
- ZED ATTACK PROXY
- zaproxy.org – ZAP's main website
- OWASP ZAP Wiki – The ZAP Wiki
- OWASP ZAP Desktop User Guide - The ZAP Desktop User Guide
- ZAP Users Group – Google group for ZAP users
- ZAP Developers Group – Google group for developers and contributors to ZAP