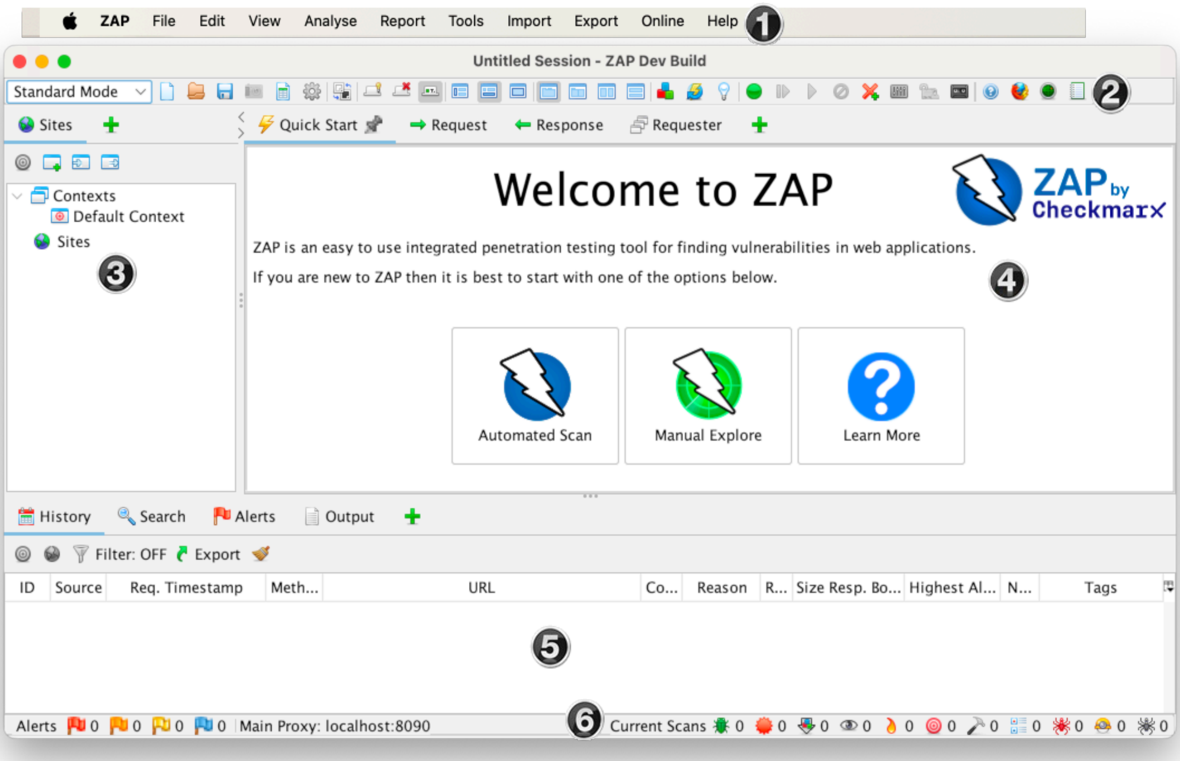
ZAP

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.



ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

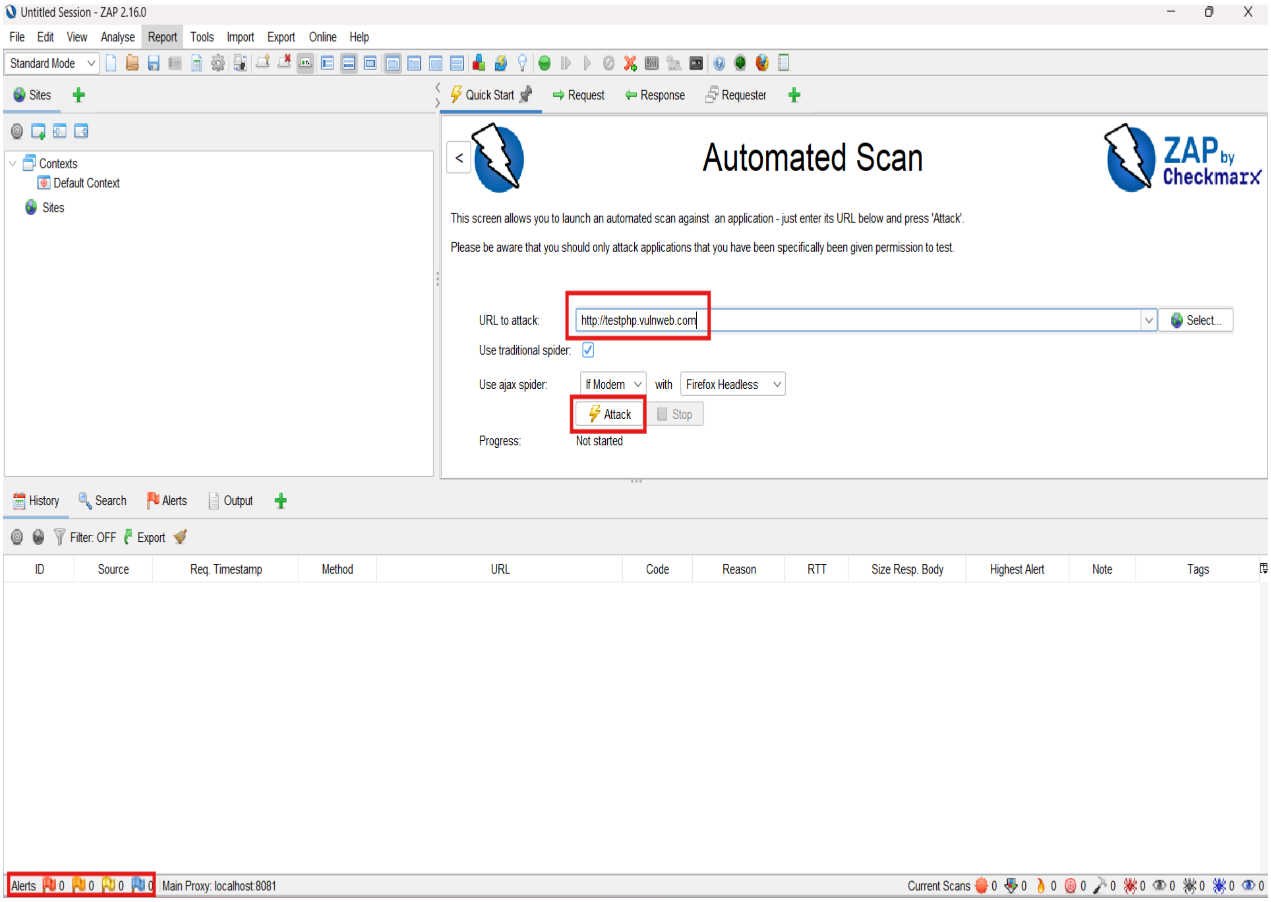
If you are new to ZAP then it is best to start with one of the options below.

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 http://testphp.vulnweb.com
4. Click the Attack



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://testphp.vulnweb.com

Use traditional spider: ☑

Use ajax spider: If Modern with Firefox Headless

Progress: Not started

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, we 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.

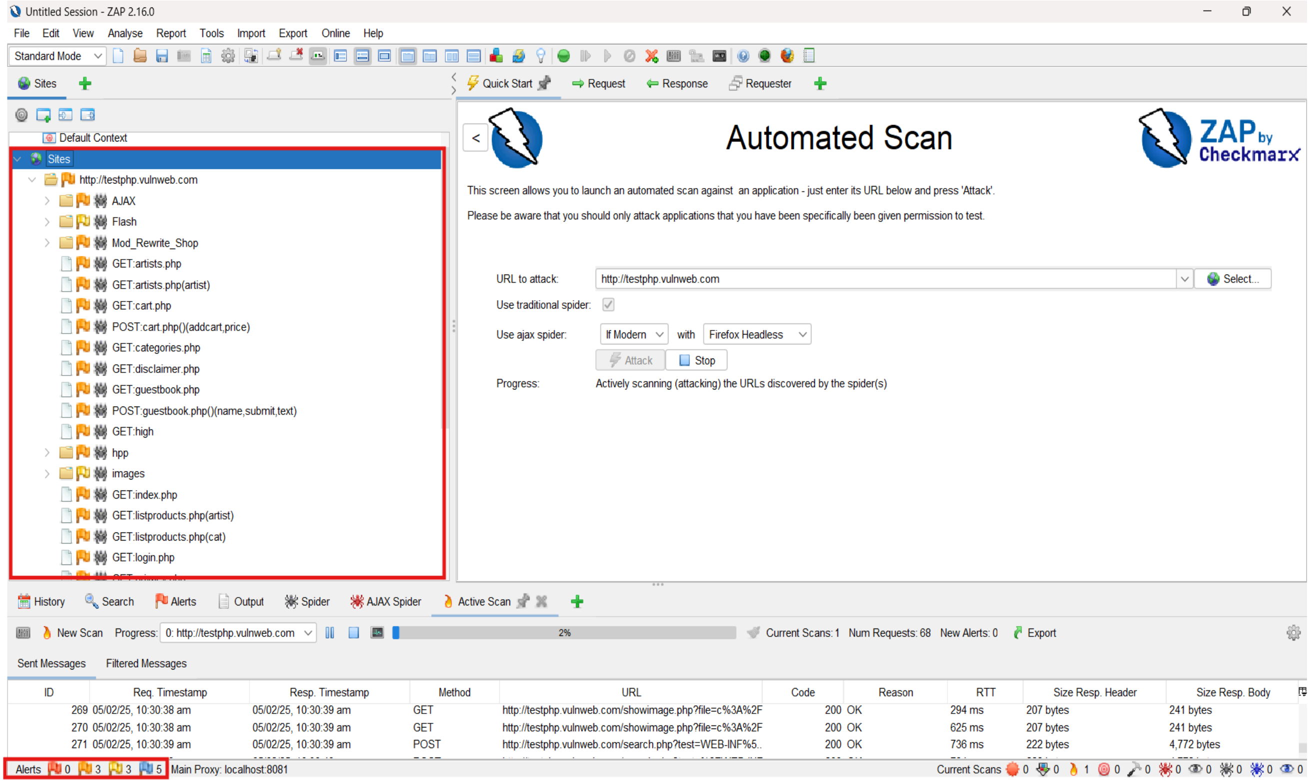
For AJAX applications, ZAP’s AJAX spider is likely to be more effective. This spider explores the web application by invoking browsers which then follow the links that have been generated. The AJAX spider is slower than the traditional spider and requires additional configuration for use in a “headless” environment.

ZAP will passively scan all of the requests and responses proxied through it. So far ZAP has only carried out passive scans of 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.

See Explored Pages

To examine a tree view of the explored pages, click the Sites tab in the Tree Window.



- Sites
  - http://testphp.vulnweb.com
    - AJAX
    - Flash
    - Mod_Rewrite_Shop
    - GET:artists.php
    - GET:artists.php(artist)
    - GET:cart.php
    - POST:cart.php()(addcart,price)
    - GET:categories.php
    - GET:disclaimer.php
    - GET:guestbook.php
    - POST:guestbook.php()(name,submit,text)
    - GET:high
    - hpp
    - images
    - GET:index.php
    - GET:listproducts.php(artist)
    - GET:listproducts.php(cat)
    - GET:login.php

Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://testphp.vulnweb.com

Use traditional spider: ☑

Use ajax spider: If Modern with Firefox Headless

Attack Stop

Progress: Actively scanning (attacking) the URLs discovered by the spider(s)

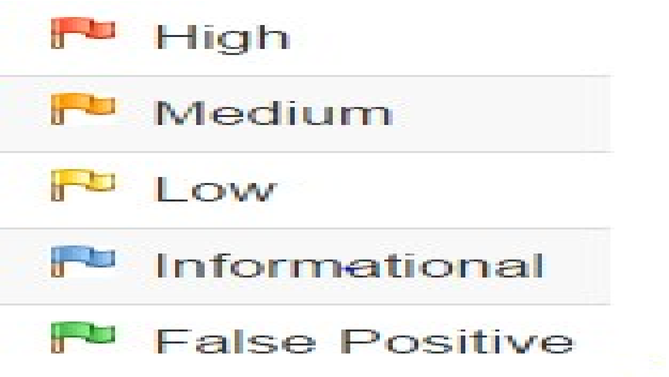
New Scan Progress: 0: http://testphp.vulnweb.com 2% Current Scans: 1 Num Requests: 68 New Alerts: 0 Export

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
269	05/02/25, 10:30:38 am	05/02/25, 10:30:39 am	GET	http://testphp.vulnweb.com/showimage.php?file=c%3A%2F	200	OK	294 ms	207 bytes	241 bytes
270	05/02/25, 10:30:38 am	05/02/25, 10:30:39 am	GET	http://testphp.vulnweb.com/showimage.php?file=c%3A%2F	200	OK	625 ms	207 bytes	241 bytes
271	05/02/25, 10:30:39 am	05/02/25, 10:30:39 am	POST	http://testphp.vulnweb.com/search.php?test=WEB-INF%5..	200	OK	736 ms	222 bytes	4,772 bytes

Alerts 0 3 3 5 Main Proxy: localhost:8081 Current Scans 0 0 1 0 0 0 0 0 0 0

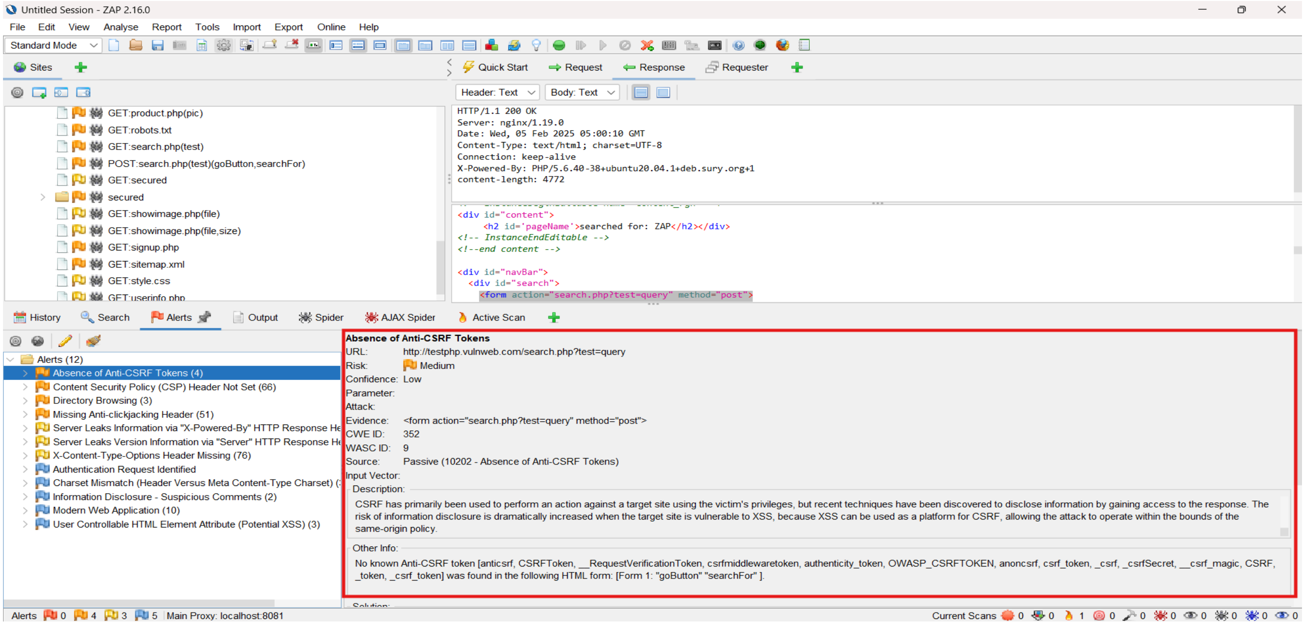
View Alerts and Alert Details

The left-hand side of the Footer contains a count of the Alerts found during test, broken out into risk categories. These risk categories are:



To view the alerts created during 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.



GET:product.php(pic)
GET:robots.txt
GET:search.php(test)
POST:search.php(test)(goButton,searchFor)
GET:secured
secured
GET:showimage.php(file)
GET:showimage.php(file,size)
GET:signup.php
GET:sitemap.xml
GET:style.css
GET:userinfo.php

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Wed, 05 Feb 2025 05:00:10 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
content-length: 4772
```

```
<div id="content">
     <h2 id='pageName'>searched for: ZAP</h2></div>
<!-- InstanceEndEditable -->
<!--end content -->

<div id="navBar">
  <div id="search">
    <form action="search.php?test=query" method="post">
```

Alerts (12)
- Absence of Anti-CSRF Tokens (4)
- Content Security Policy (CSP) Header Not Set (66)
- Directory Browsing (3)
- Missing Anti-clickjacking Header (51)
- Server Leaks Information via "X-Powered-By" HTTP Response He
- Server Leaks Version Information via "Server" HTTP Response He
- X-Content-Type-Options Header Missing (76)
- Authentication Request Identified
- Charset Mismatch (Header Versus Meta Content-Type Charset) (
- Information Disclosure - Suspicious Comments (2)
- Modern Web Application (10)
- User Controllable HTML Element Attribute (Potential XSS) (3)

Absence of Anti-CSRF Tokens

URL:	http://testphp.vulnweb.com/search.php?test=query
Risk:	Medium
Confidence:	Low
Parameter:	
Attack:	
Evidence:	<form action="search.php?test=query" method="post">
CWE ID:	352
WASC ID:	9
Source:	Passive (10202 - Absence of Anti-CSRF Tokens)
Input Vector:	

Description:

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

Other Info:

No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "goButton" "searchFor"].

Solution:

Alerts 0 4 3 5 Main Proxy: localhost:8081

Current Scans 0 0 1 0 0 0 0 0 0 0