Introduction to ZAP with Selenium

Efficiency On the GO

Content

Introduction to ZAP and Selenium Automation

ZAP (Zed Attack Proxy) is a powerful open-source web application security scanner. When combined with Selenium, it allows for automated security testing of web applications. This document outlines the requirements and setup needed to integrate ZAP with Selenium in a Java-based environment using Eclipse.

Requirements

1. **ZAP Application**: Download and install OWASP ZAP from https://www.zaproxy.org/.
2. **Java Development Kit (JDK)**: Ensure JDK is installed and configured.
3. **Eclipse IDE**: Set up Eclipse for Java development.
4. **Selenium WebDriver**: Install Selenium WebDriver dependencies.
5. **ZAP Client API dependency from maven**: Add the zap clientapi dependencies.

```
<!-- https://mvnrepository.com/artifact/org.zaproxy/zap-clientapi -->
<dependency>
    <groupId>org.zaproxy</groupId>
    <artifactId>zap-clientapi</artifactId>
    <version>1.16.0</version>
</dependency>
```

6. **ZAP API Key**: Configure the API key for secure communication.
7. **Set Environment Variables**: Define API key, port, and URL in Eclipse.
8. **ZAP Proxy Configuration:**
    1. ZAP must be running before executing test.

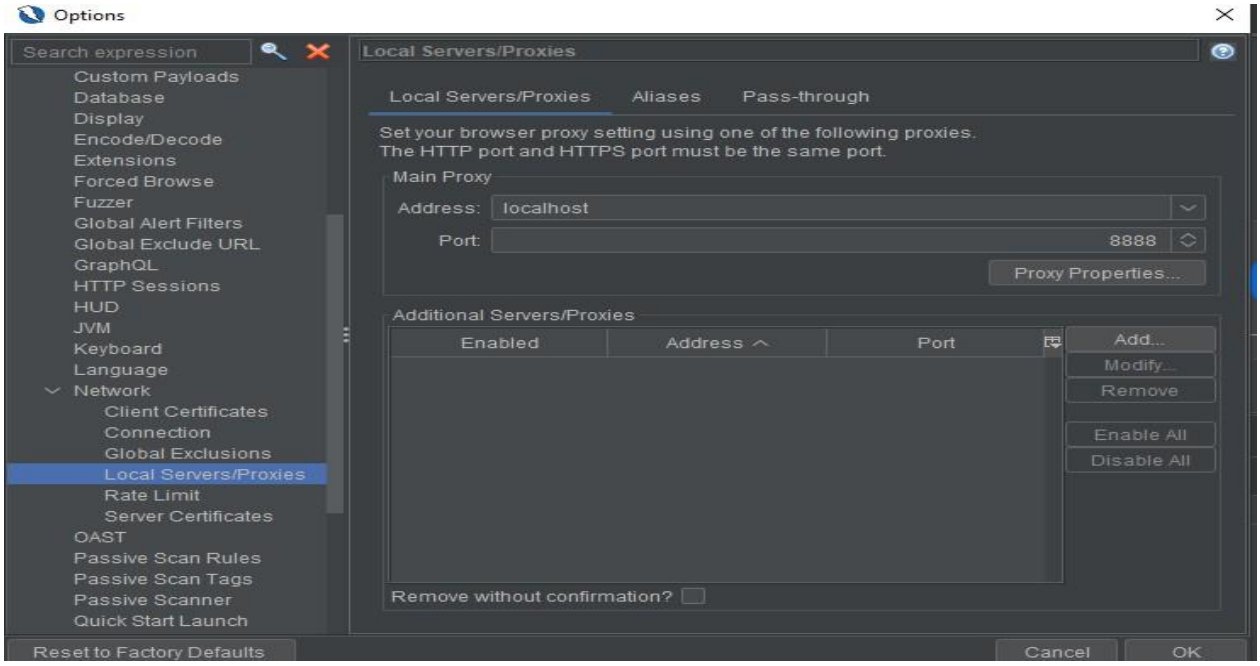
Types of scans in this document:

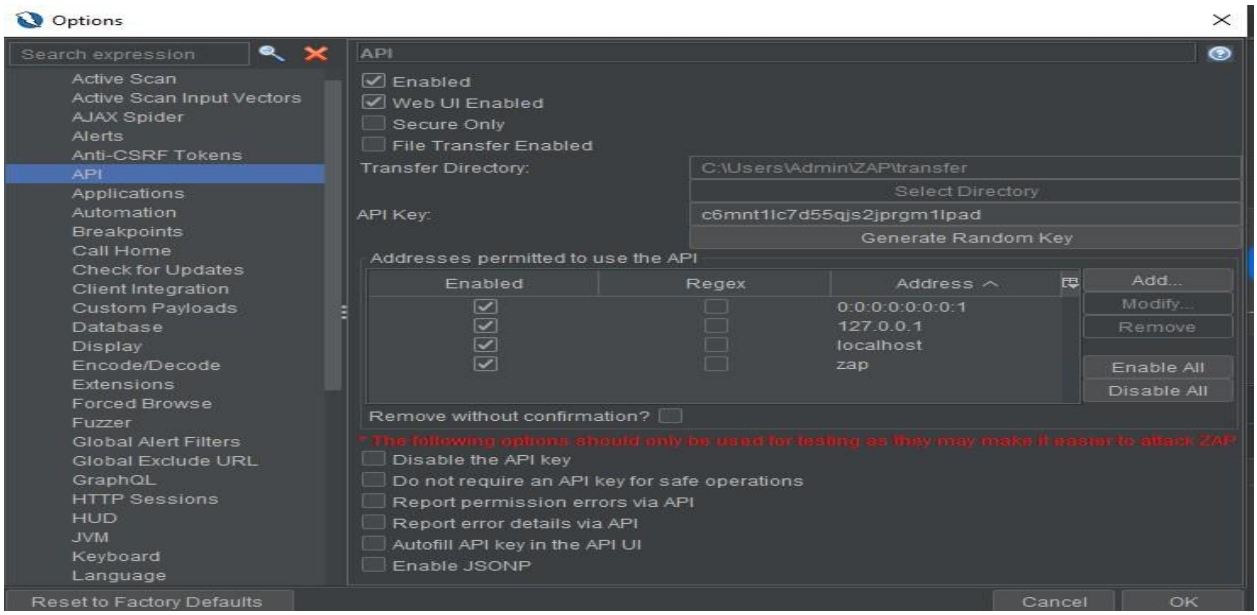
- Passive Scan: Runs without interacting with the application.
- Active Scan: Actively tests vulnerabilities by attacking input fields, etc.
- Spider Scan: which crawl the web application

Setting Up ZAP for Selenium Testing

1. **Install and Launch ZAP**

   - Download and install OWASP ZAP from the official website.
   - Open ZAP and configure the proxy settings to 127.0.0.1:8888.





   - Navigate to Tools > Options > API and enable the API key.

- Copy the API key for use in the Selenium test script.

Passive Scan:

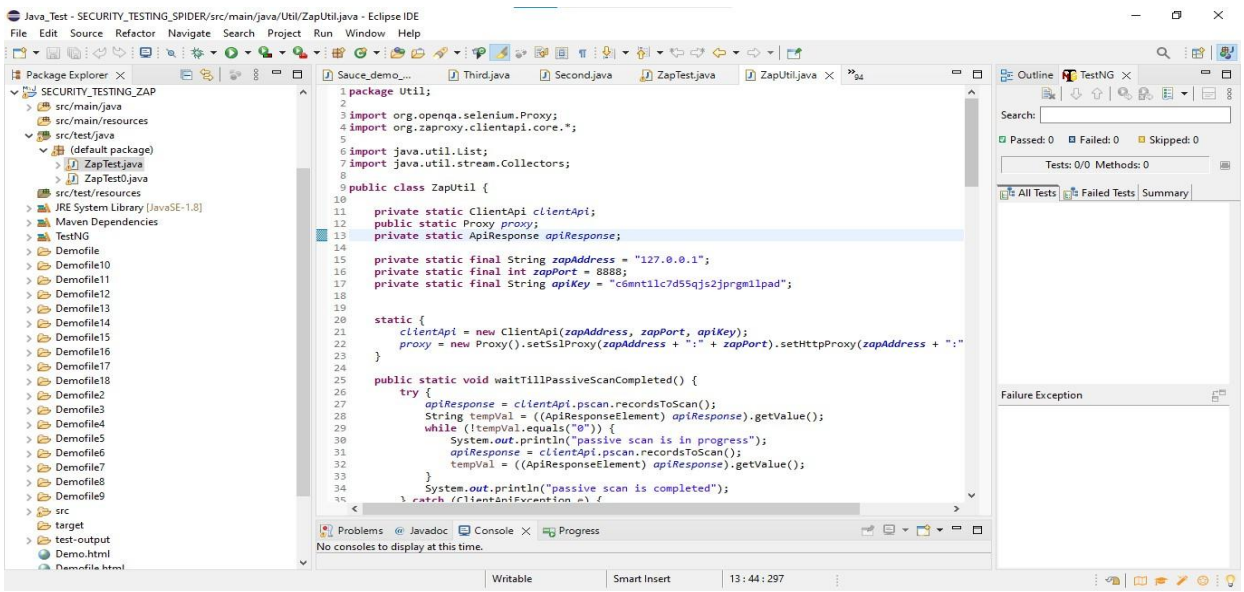
Example of WebGoat application for testing

- First we are setting up zap proxy i.e url, port and api key in util class
  - **private static final String zapAddress = "127.0.0.1";**
  - **private static final int zapPort = 8888;**
  - **private static final String apiKey = "c6mnt1lc7d55qjs2jprgm1lpad";**

- Second Step, all these three variable are stored under the client api object and a proxy is setup using the Proxy class.
  - **clientApi = new ClientApi(zapAddress, zapPort, apiKey);**
  - **proxy = new Proxy().setSslProxy(zapAddress + ":" + zapPort).setHttpProxy(zapAddress + ":" + zapPort);**

- Third, there are two types of scans in zap active and passive scan at first scenario we are doing passive scan on a website. For that a method is written in that a pscan is called from the class of zap client api which will do passive scan.
  - ***apiResponse* = *clientApi*.pscan.recordsToScan();**

- Fourth, we will wait until the passive scan is complete then we will call the generate report method.
  - In this reports.generate is a method which helps to generate report and there are many report template you can choose as per the choice.
    - **clientApi.reports.generate(title, template, theme, description, contexts, sites, sections,includedconfidences, includedrisks, reportfilename, "", reportdir, display);**



- Fifth, we will create test class were we add the webgoat web application url to scan and here using selenium we can use sendKeys method to login and the scan the dashboard and website so that we can explore more flaws in the website.

  - **private final String urlToTest = "http://127.0.0.1:8080/WebGoat/login";**

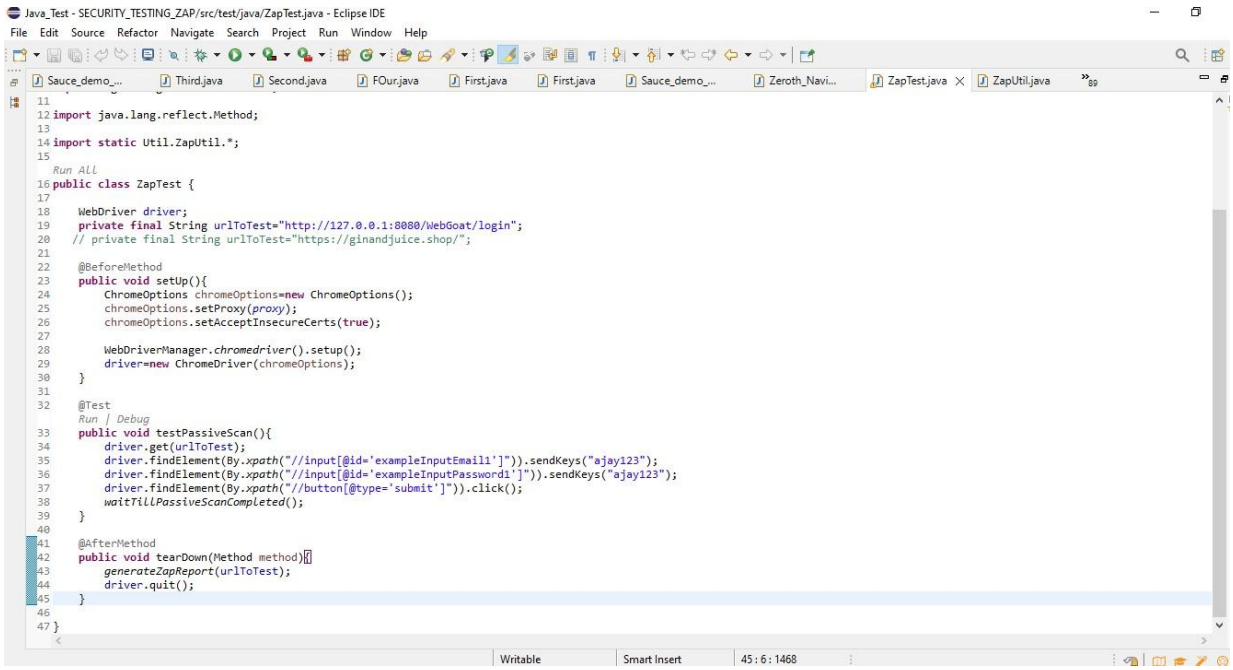
  - **@Test**

public void testPassiveScan() { driver.get(urlToTest);

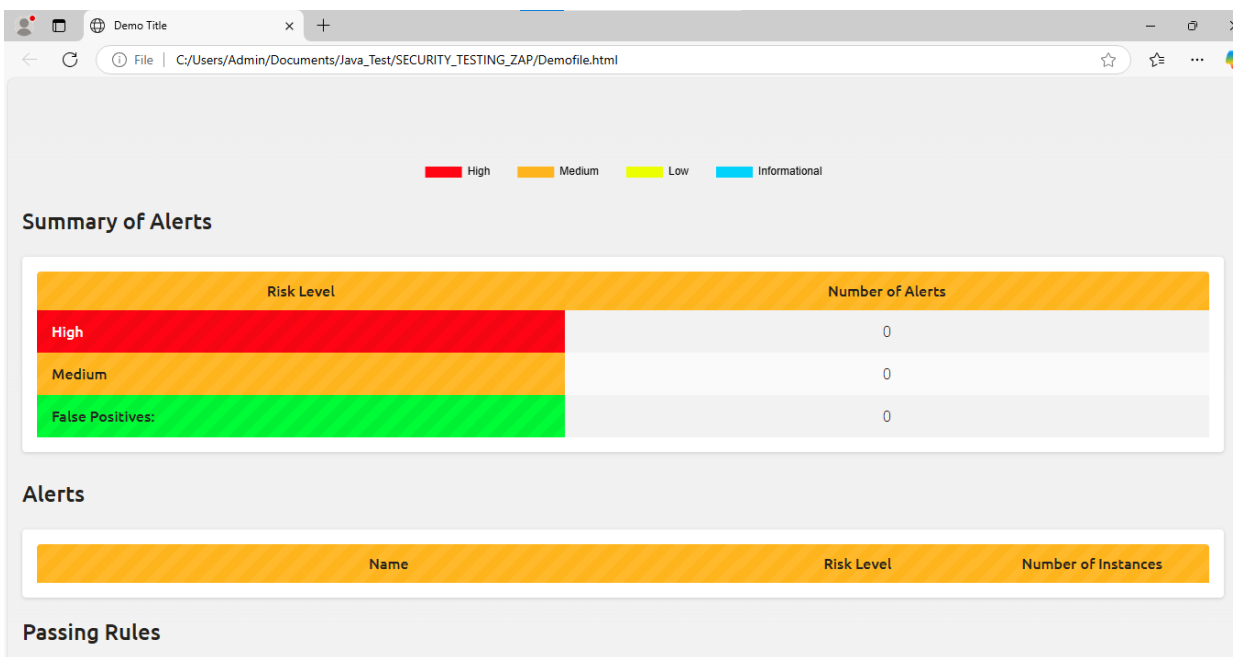
driver.findElement(By.xpath("//input[@id='exampleInputEmail1']")).sendKeys ("ajay123");
driver.findElement(By.xpath("//input[@id='exampleInputPassword1']")).send Keys("ajay123");
driver.findElement(By.xpath("//button[@type='submit']")).click();

waitTillPassiveScanCompleted();

}



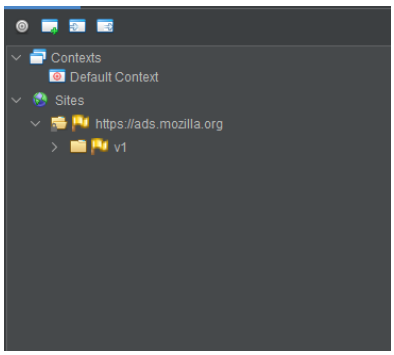
- After Scan is complete a report is generated in the given format zap supports pdf, html and other formats.



Active Scan:

Scenario of ginandjuice.shop application for testing.

- For Active Scan same as for passive scan her we use ascan api for scanning the webpage.
    - First we are setting up zap proxy i.e url, port and api key in util class
        - **private static final String zapAddress = "127.0.0.1";**
        - **private static final int zapPort = 8888;**
        - **private static final String apiKey = "c6mnt1lc7d55qjs2jprgm1lpad";**
    - Second Step, all these three variable are stored under the client api object and a proxy is setup using the Proxy class.
        - **clientApi = new ClientApi(zapAddress, zapPort, apiKey);**
        - **proxy = new Proxy().setSslProxy(zapAddress + ":" + zapPort).setHttpProxy(zapAddress + ":" + zapPort);**
    - Third, before scanning the ascan we need to add the url to scan tree that is in zap .



In eclipse we will create a method to add the scan tree

```
public static void addURLToScanTree(String site_to_test) throws ClientApiException {
    clientApi.core.accessUrl(site_to_test, "false");
    if(getUrlsFromScanTree().contains(site_to_test))
        System.out.println(site_to_test+ " has been added to scan tree");
    else
        throw new RuntimeException(site_to_test +" not added to scan tree, active scan will not be possible");
}
```

clientApi.ascan.scan(url, recurse, inscopeonly, scanpolicyname, method, postdata, contextId);

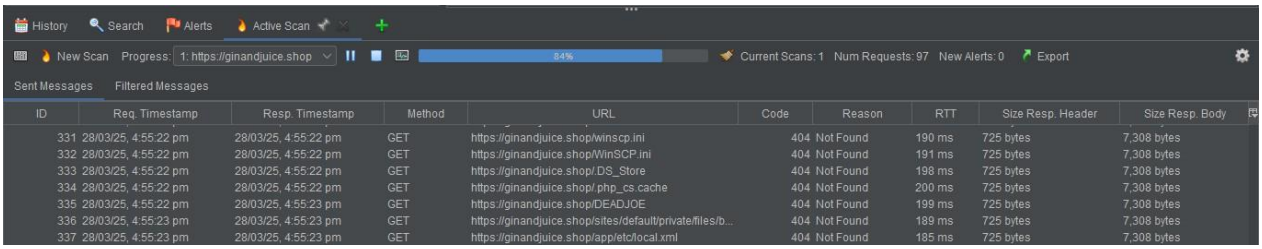
- **Fourth, we will wait until the active scan is complete then we will call the generate report method.**

```
private static void waitTillActiveScanIsCompleted(String scanId) throws ClientApiException {
    apiResponse=clientApi.ascan.status(scanId);
    String status=((ApiResponseElement)apiResponse).getValue();

    while (!status.equals("100")){
        apiResponse=clientApi.ascan.status(scanId);
        status=((ApiResponseElement)apiResponse).getValue();
        System.out.println("Active scan is in progress");
    }

    System.out.println("Active scan has completed");
}
```

- **clientApi.reports.generate(title, template, theme, description, contexts, sites, sections,includedconfidences, includedrisks, reportfilename, "", reportdir, display);**

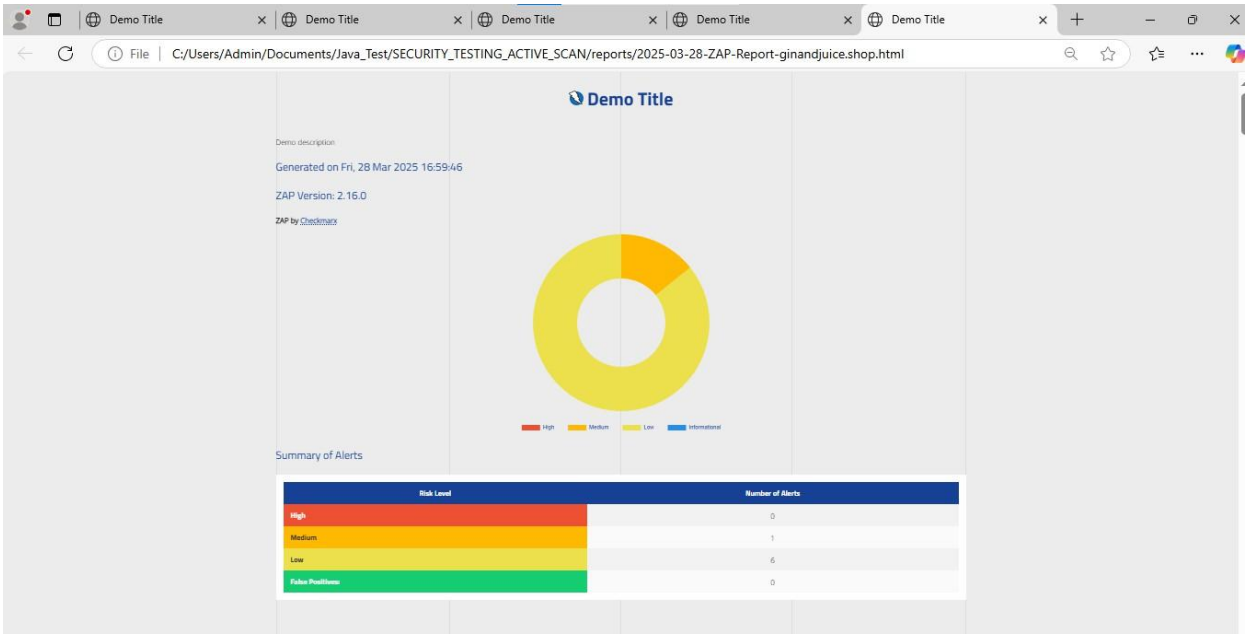
```
Problems  @ Javadoc  Console ×  Progress
ZapTest [TestNG] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (28-Mar-2025, 4:54:47 pm) [pid: 15508]
[RemoteTestNG] detected TestNG version 7.4.0
16:54:49.807 [main] DEBUG io.github.bonigarcia.wdm.WebDriverManager - Using WebDriverManager 5.8.0
16:54:50.667 [main] DEBUG io.github.bonigarcia.wdm.cache.ResolutionCache - Resolution chrome=134 in cache (valid unt
16:54:50.669 [main] DEBUG io.github.bonigarcia.wdm.cache.ResolutionCache - Resolution chrome134=134.0.6998.165 in ca
16:54:50.672 [main] INFO io.github.bonigarcia.wdm.WebDriverManager - Using chromedriver 134.0.6998.165 (resolved dri
16:54:50.782 [main] DEBUG io.github.bonigarcia.wdm.WebDriverManager - Driver chromedriver 134.0.6998.165 found in ca
16:54:50.785 [main] INFO io.github.bonigarcia.wdm.WebDriverManager - Exporting webdriver.chrome.driver as C:\Users\A
Mar 28, 2025 4:54:52 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find CDP implementation matching 134
Mar 28, 2025 4:54:52 PM org.openqa.selenium.chromium.ChromiumDriver lambda$new$5
WARNING: Unable to find version of CDP to use for 134.0.6998.167. You may need to include a dependency on a specific
https://ginandjuice.shop/ has been added to scan tree
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
Active scan is in progress
```

This is in zap



ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
331	28/03/25, 4:55:22 pm	28/03/25, 4:55:22 pm	GET	https://ginandjuice.shop/winscp.ini	404	Not Found	190 ms	725 bytes	7,308 bytes
332	28/03/25, 4:55:22 pm	28/03/25, 4:55:22 pm	GET	https://ginandjuice.shop/WinSCP.ini	404	Not Found	191 ms	725 bytes	7,308 bytes
333	28/03/25, 4:55:22 pm	28/03/25, 4:55:22 pm	GET	https://ginandjuice.shop/.DS_Store	404	Not Found	198 ms	725 bytes	7,308 bytes
334	28/03/25, 4:55:22 pm	28/03/25, 4:55:22 pm	GET	https://ginandjuice.shop/.php_cs.cache	404	Not Found	200 ms	725 bytes	7,308 bytes
335	28/03/25, 4:55:22 pm	28/03/25, 4:55:23 pm	GET	https://ginandjuice.shop/DEADJOE	404	Not Found	199 ms	725 bytes	7,308 bytes
336	28/03/25, 4:55:23 pm	28/03/25, 4:55:23 pm	GET	https://ginandjuice.shop/sites/default/private/files/b...	404	Not Found	189 ms	725 bytes	7,308 bytes
337	28/03/25, 4:55:23 pm	28/03/25, 4:55:23 pm	GET	https://ginandjuice.shop/app/etc/local.xml	404	Not Found	185 ms	725 bytes	7,308 bytes

Report of the Active scan



Risk Level	Number of Alerts
High	0
Medium	1
Low	6
False Positives:	0

Spider Scan:

Example using the WebGoat application.

- First we are setting up zap proxy i.e url, port and api key in util class
  - **private static final String zapAddress = "127.0.0.1";**
  - **private static final int zapPort = 8888;**
  - **private static final String apiKey = "c6mnt1lc7d55qjs2jprgm1lpad";**

- Second Step, all these three variable are stored under the client api object and a proxy is setup using the Proxy class.
  - **clientApi = new ClientApi(zapAddress, zapPort, apiKey);**
  - **proxy = new Proxy().setSslProxy(zapAddress + ":" + zapPort).setHttpProxy(zapAddress + ":" + zapPort);**

- Third Step, performing spider using spider class and scan method

***apiResponse=clientApi*.spider.scan(site_to_test,null,null,null,null);**

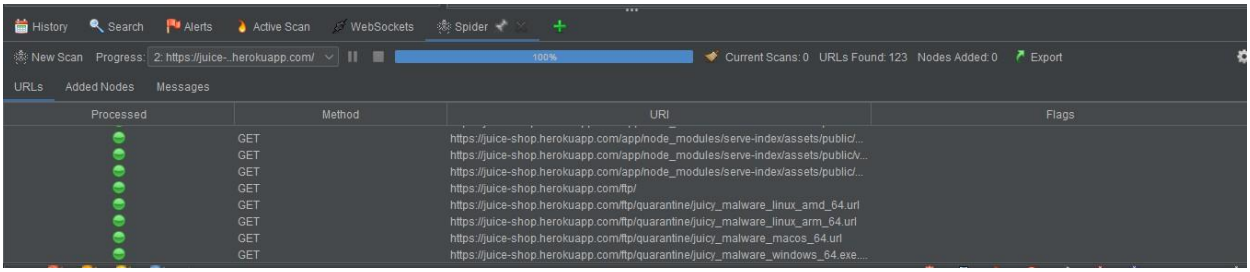
```
public static void performSpidering(String site_to_test, String contextName) throws ClientApiException {
    apiResponse=clientApi.spider.scan(site_to_test,null,null,null,null);
    String spiderScanId=((ApiResponseElement)apiResponse).getValue();

    apiResponse=clientApi.spider.status(spiderScanId);
    String spiderScanStatus=((ApiResponseElement)apiResponse).getValue();

    while (!spiderScanStatus.equals("100")){
        apiResponse=clientApi.spider.status(spiderScanId);
        spiderScanStatus=((ApiResponseElement)apiResponse).getValue();
        System.out.println("Spidering is in progress, current status="+spiderScanStatus);
    }

    waitTillPassiveScanCompleted();

    System.out.println("starting active scan--");
    performActiveScan(site_to_test, contextName);
}
```

In ZapTest we will call the performspidering method by sending url as parameter to spider.

```
@Test
Run | Debug
public void testSpider() throws ClientApiException {
    performSpidering(urlToTest,contextName);
}

@AfterMethod
```

Results:

```
ZapTest [TestNG] [pid: 5788]
[RemoteTestNG] detected TestNG version 7.4.0
17:07:11.394 [main] DEBUG io.github.bonigarcia.wdm.WebDriverManager - Using WebDriverManager 5.8.0
17:07:12.610 [main] DEBUG io.github.bonigarcia.wdm.cache.ResolutionCache - Resolution chrome=134 in cache (valid unt
17:07:12.612 [main] DEBUG io.github.bonigarcia.wdm.cache.ResolutionCache - Resolution chrome134=134.0.6998.165 in ca
17:07:12.613 [main] INFO io.github.bonigarcia.wdm.WebDriverManager - Using chromedriver 134.0.6998.165 (resolved dri
17:07:12.712 [main] DEBUG io.github.bonigarcia.wdm.WebDriverManager - Driver chromedriver 134.0.6998.165 found in ca
17:07:12.715 [main] INFO io.github.bonigarcia.wdm.WebDriverManager - Exporting webdriver.chrome.driver as C:\Users\A
Mar 28, 2025 5:07:14 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find CDP implementation matching 134
Mar 28, 2025 5:07:14 PM org.openqa.selenium.chromium.ChromiumDriver lambda$new$5
WARNING: Unable to find version of CDP to use for 134.0.6998.167. You may need to include a dependency on a specific
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
Spidering is in progress, current status=0
```



Java_Test - SECURITY_TESTING_ZAP/src/test/java/ZapTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- Fifth_Week_Hybrid_FrameWork
- First
- Learn_Java
- RestAssured_Automation_FrameWork
- SECURITY_TESTING_ZAP
  - src/main/java
    - Util
      - ZapUtil.java
  - src/main/resources
  - src/test/java
    - (default package)
      - ZapTest.java
  - src/test/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - TestNG
  - Demofile
  - Demofile2
  - Demofile3
  - Demofile4
  - Demofile5
  - Demofile6
  - src
  - target
  - test-output
  - Demofile.html
  - Demofilename.html
  - pom.xml
- Selenium
- SELENIUM_FIFTH_WEEK

Slider.java | Java_Script_... | Single_File_... | ZapUtil.java | *ZapTest.java

```
27          chromeOptions.setProxy(proxy);
28          chromeOptions.setAcceptInsecureCerts(true);
29
30          WebDriverManager.chromedriver().setup();
31          driver=new ChromeDriver(chromeOptions);
32      }
33
34      @Test
        Run | Debug
35      public void testPassiveScan(){
36          driver.get(urlToTest);
37          driver.findElement(By.xpath("//input[@id='exampleInputEmail1']")).sendKeys("ajay123");
38          driver.findElement(By.xpath("//input[@id='exampleInputPassword1']")).sendKeys("ajay123");
39          driver.findElement(By.xpath("//button[@type='submit']")).click();
40          waitTillPassiveScanCompleted();
41      }
42
43      @Test
        Run | Debug
44      public void testActiveScan() throws ClientApiException {
45          addURLToScanTree(urlToTest);
46          performActiveScan(urlToTest,contextName);
47      }
48      /*@Test
49      public void testSpider() throws ClientApiException {
50          performSpidering(urlToTest,contextName);
51      }*/
52      @AfterMethod
53      public void tearDown(Method method) throws ClientApiException{
54          generateZapReport(urlToTest);
55          cleanTheScanTree();
56          //driver.quit();
57      }
58
59  }
```