Lab 4

Auditing Security

Review Question

- What is SAST ?

- What is DAST ?

- What is fuzzing ?

- What is spidering ?

Hands-on Lab : Practicing SAST with SonarQube

SonarQube is an open source quality management platform, designed to analyze and measure your code's technical quality. It is used to test code written in the main programming languages such as C/C++, JavaScript, Java, C#, PHP, and Python, and even a combination of several languages simultaneously.

SonarQube not only provides a complete report of code bugs, syntax errors, and vulnerabilities, but also suggestions and examples about how to fix your code. It also measures the technical debt, so you can easily calculate the time you will spend fixing these issues.

Security Reports by SonarQube quickly give you the big picture on your application's security, with breakdowns of just where you stand in regard to each of the OWASP Top 10, and SANS Top 25 categories, and CWE-specific details.
The Security Reports are fed by the analyzers, which rely on the rules activated in your quality profiles to raise security issues. If there are no rules corresponding to a given OWASP category activated in your Quality Profile.

In this lab, you will learn how to run a code analysis using a sample project, use the SonarQube user interface (UI) to obtain all details about the issues and run new analysis when the issues have been solved. To take a step further in the use of SonarQube, this tutorial will walk you through the process of integrating your GitHub repositories and Travis builds with SonarQube, so you can focus on writing more and better code.

1. Installing SonarQube

Installing from a zip file

1. Download the SonarQube Community Edition.
2. Unzip it, let's say in C:\sonarqube or /opt/sonarqube.
3. Start the SonarQube Server:

```
# On Windows, execute:
C:\sonarqube\bin\windows-x86-xx\StartSonar.bat

# On other operating systems, as a non-root user execute:
/opt/sonarqube/bin/[OS]/sonar.sh console
```

   If your instance fails to start, check your logs to find the cause.
4. Log in to http://localhost:9000 with System Administrator credentials (login=admin, password=admin).
5. Click the **Create new project** button to analyze your first project.

Using Docker

A Docker image of the Community Edition is available on Docker Hub. You can find usage and configuration examples there.
Start the server by running:

```
$ docker run -d --name sonarqube -p 9000:9000 sonarqube/lts
```

2. Analyzing Source Code

Once the SonarQube platform has been installed, you're ready to install a scanner and begin creating projects. To do that, you must install and configure the scanner that is most appropriate for your needs. Do you build with:

- Maven - use the SonarScanner for Maven
- Gradle - SonarScanner for Gradle
- MSBuild - SonarScanner for MSBuild
- anything else (CLI) - SonarScanner

Configuring SonarScanner for Maven

The ability to execute the SonarQube analysis via a regular Maven goal makes it available anywhere Maven is available (developer build, CI server, etc.), without the need to manually download, setup, and maintain a SonarQube Runner installation. The Maven build already has much of the information needed for SonarQube to successfully analyze a project. By preconfiguring the analysis based on that information, the need for manual configuration is reduced significantly.

Edit the settings.xml file, located in $MAVEN_HOME/conf or ~/.m2, to set the plugin prefix and optionally the SonarQube server URL.

```
<settings>
    <pluginGroups>
        <pluginGroup>org.sonarsource.scanner.maven</pluginGroup>
    </pluginGroups>
    <profiles>
        <profile>
            <id>sonar</id>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <properties>
         <!-- Optional URL to server. Default value is http://localhost:9000 -->
                <sonar.host.url>
                  http://myserver:9000
                </sonar.host.url>
            </properties>
        </profile>
     </profiles>
</settings>
```

Analyzing

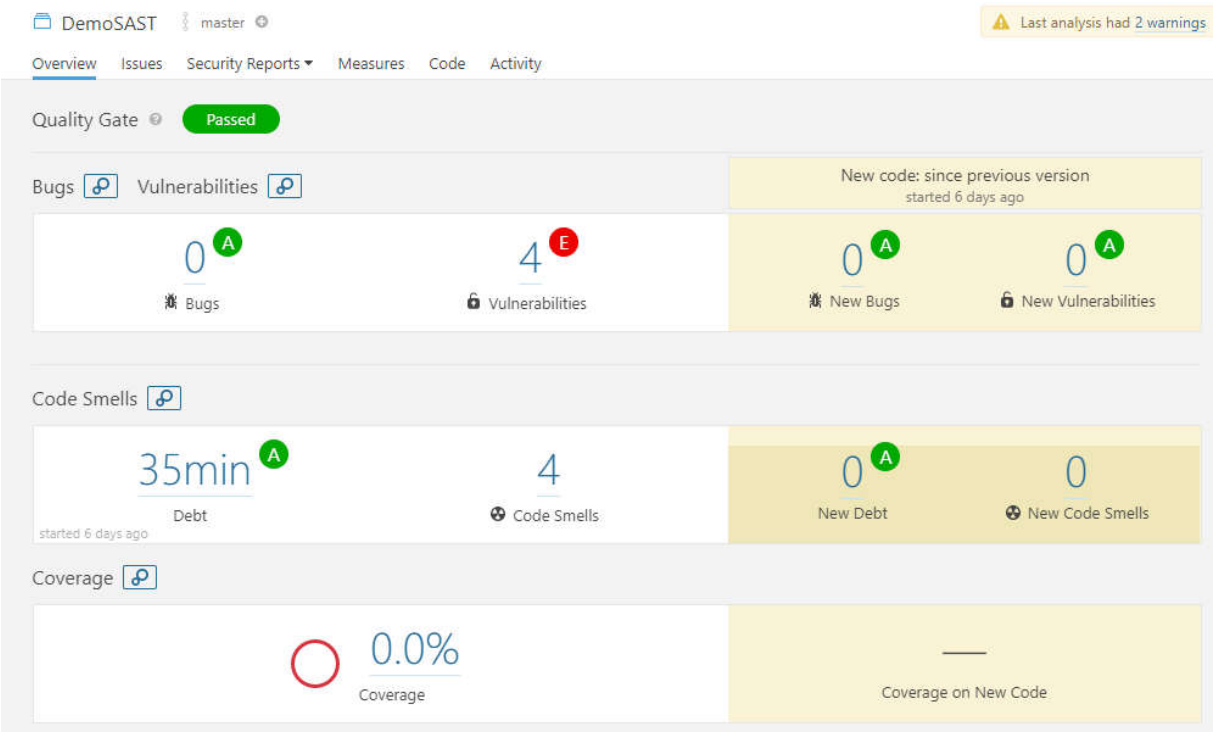
Analyzing a Maven project consists of running a Maven goal: sonar:sonar from the directory that holds the main project pom.xml.

```
mvn clean verify sonar:sonar
```

In some situations you may want to run the sonar:sonar goal as a dedicated step. Be sure to use install as first step for multi-module projects

```
mvn clean install
mvn sonar:sonar
```

The results are displayed on the SonarQube dashboard.



To get the details on security issues, select the “Security Reports” menu. Security Reports quickly give you the big picture on your application's security, with breakdowns of just where you stand in regard to each of the OWASP Top 10, and SANS Top 25 categories, and CWE-specific details.

Categories	Vulnerabilities	Security Hotspots		
		Open	In Review	Won't Fix
A1 - Injection	0 A	2	0	0
A2 - Broken Authentication	0 A	0	0	0
A3 - Sensitive Data Exposure	4 E	0	0	0
A4 - XML External Entities (XXE)	0 A	0	0	0
A5 - Broken Access Control	0 A	0	0	0
A6 - Security Misconfiguration	0 A	0	0	0
A7 - Cross-Site Scripting (XSS)	0 A	0	0	0
A8 - Insecure Deserialization	0 A	0	0	0
A9 - Using Components with Known Vulnerabilities	0 A	0	0	0
A10 - Insufficient Logging & Monitoring	0 A	0	0	0
Not OWASP	0 A	1	0	0

What's the difference between a Security Hotspot and a Vulnerability?

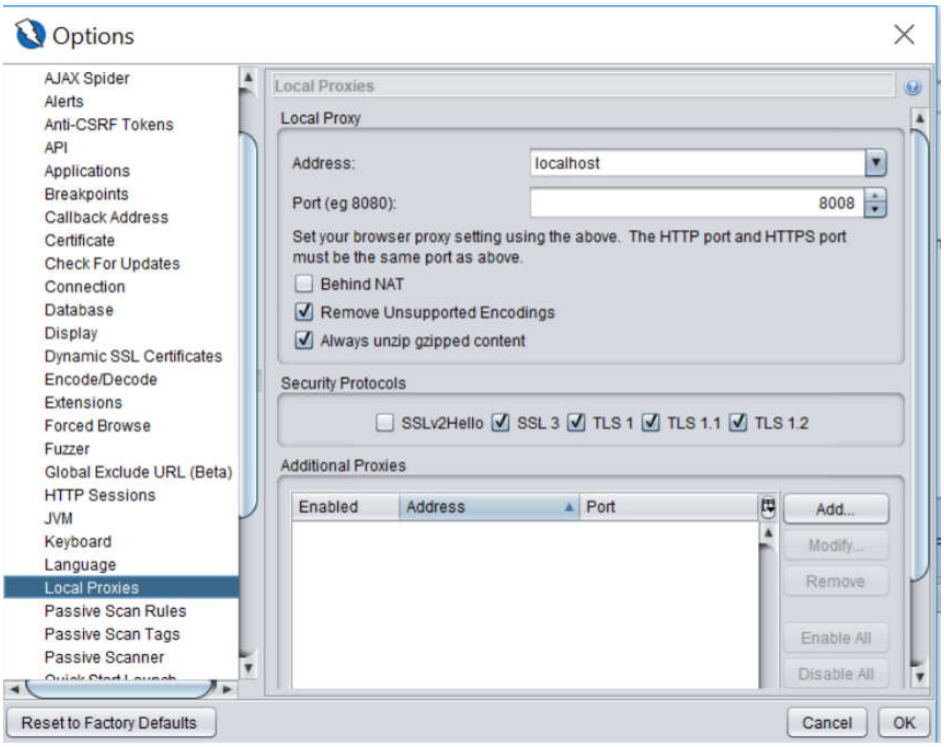
Vulnerabilities are points in the code which are open to attack. Security Hotspots highlight security-sensitive pieces of code that need to be manually reviewed to ensure the sensitive piece of code is being used in the safest manner. Security hotspots also help educate developers on security issues.

Hands-on Lab : Practicing DAST with OWASP Zap Proxy

1. Installing and Configuring Zed Attack Proxy (ZAP)

The OWASP Zed Attack Proxy (ZAP) is a free security tool and is actively maintained by hundreds of international volunteers. It can help you automatically find security vulnerabilities in your web applications while you are developing and testing your applications. It is also a great tool for experienced penetration testers to use for manual security testing.

- Download Zed Attack Proxy (ZAP) at https://github.com/zaproxy/zaproxy/wiki/Downloads. Choose the right ZAP 2.7.0 Standard download file for your OS.
- Download the Getting Started Guide (pdf), https://tinyurl.com/ybcydtun
- Follow the instructions on page 3, section “Install and Configure ZAP” to install ZAP.
- After the installation is completed, follow the Getting Started Guide until the end of page 4 to get acquainted with the user interface.
- From the menu bar, choose Tools - Options - Local Proxies. You will see the interface as shown in Figure below. Set the Address as localhost and port as **8008**.



- Disconnect your computer from the Internet since we will run WebGoat. Running WebGoat would make your computer vulnerable to the threats. (vii) Start WebGoat as you did in last week’s lab and login.

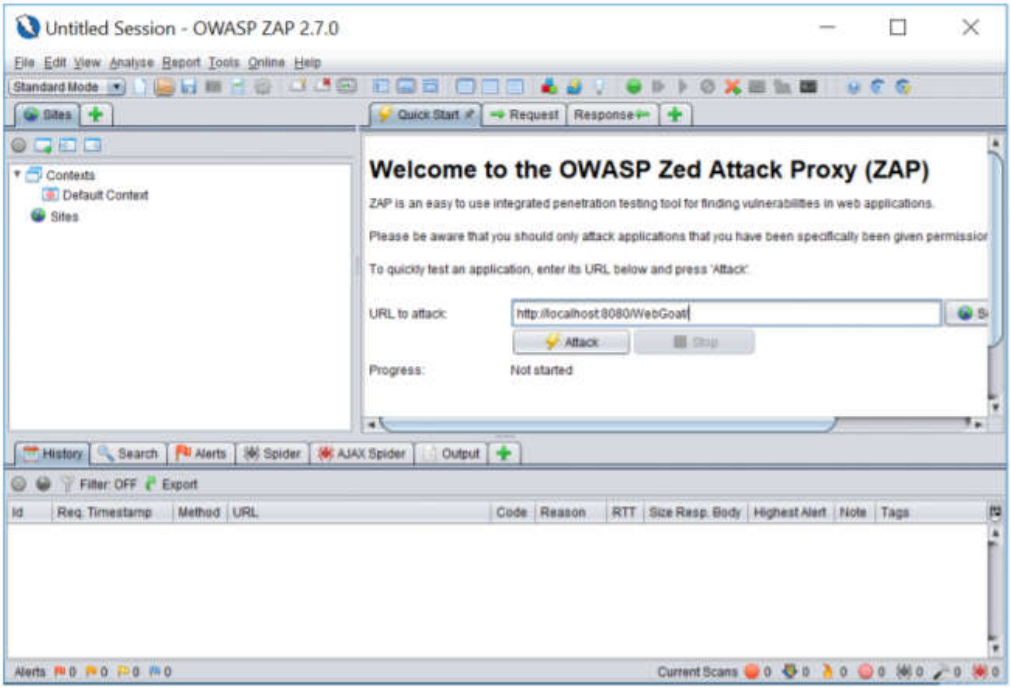
2. Testing WebGoat

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.

2.1 Run a Quick Start Test

- Click the Quick Start tab of the Workspace Window as shown in Figure below and enter the URL of the WebGoat application in the URL to attack field. Click the Attack button.

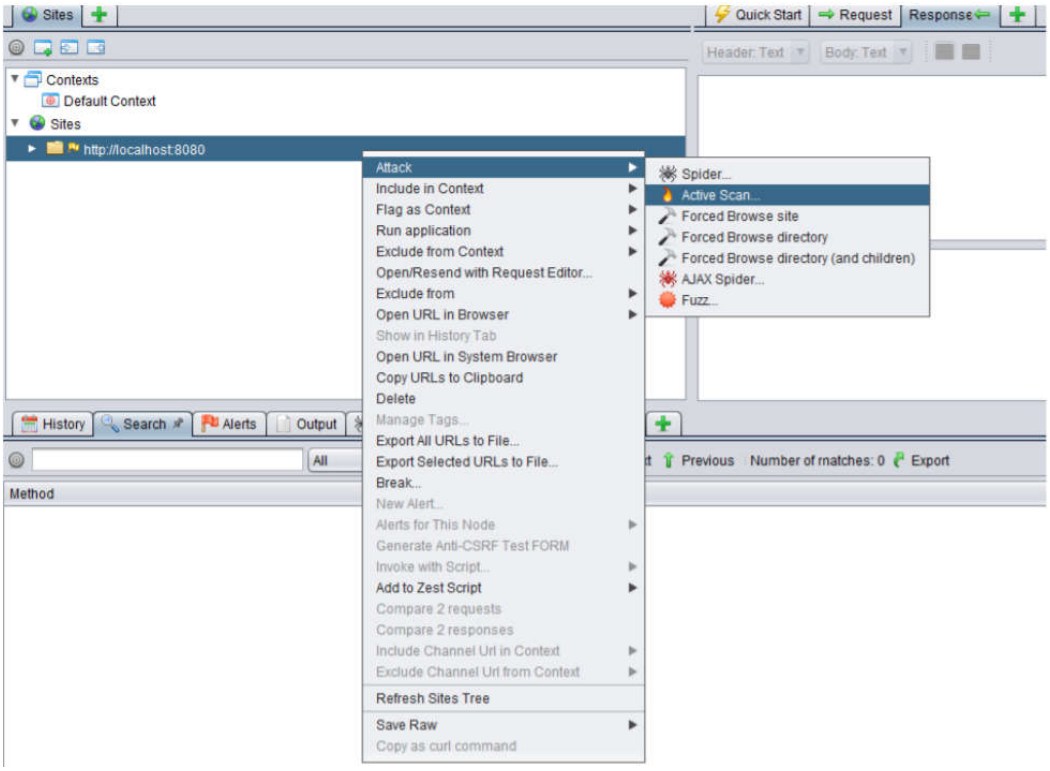
  ZAP will proceed to crawl the web application with its spider, then passively scan eachpage it finds. Then ZAP will use the active scanner to attack all of the discovered pages, functionality, and parameters.



- Interpret your test results by following the instructions on page 6 of the Getting Started Guide.

Run an Active Scan with ZAP

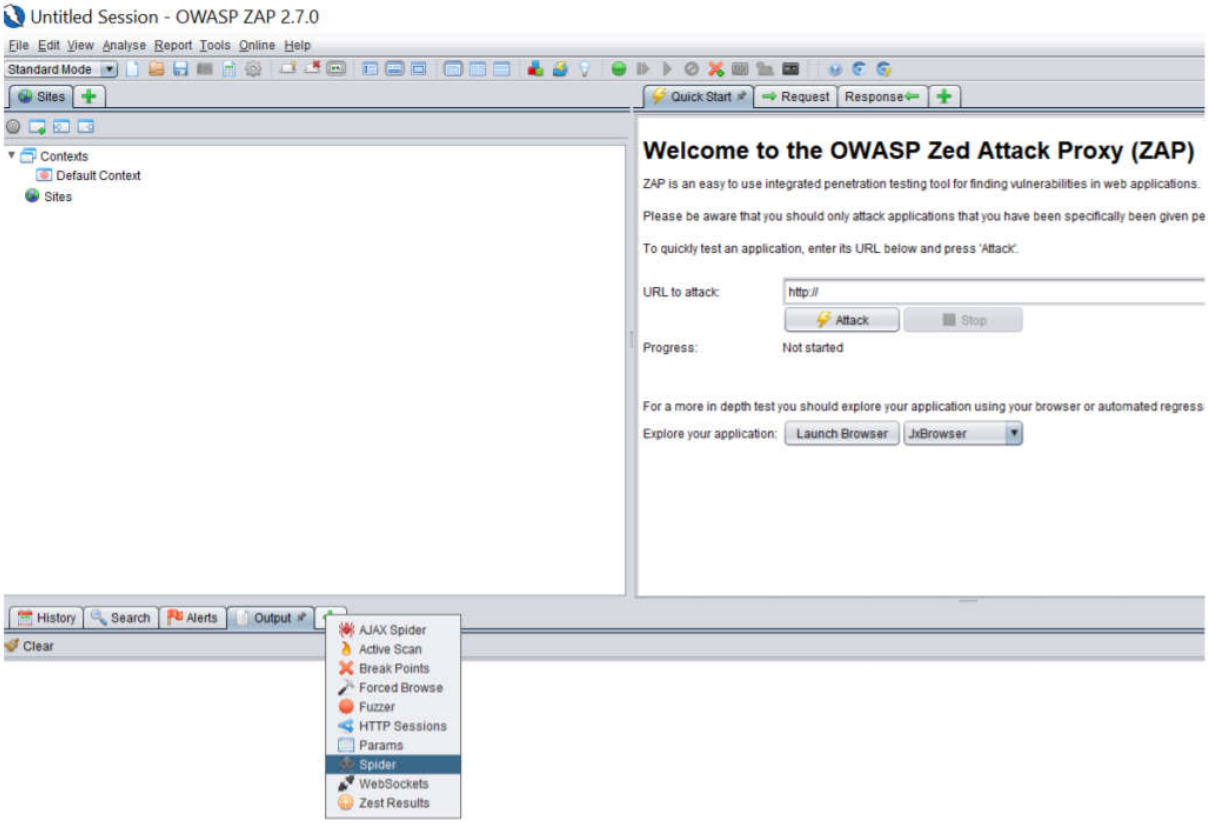
- In the Tree View, go to Sites tab and select the sites you want to perform an active scan on.
- Right-Click the selected site and select Attack - Active Scan as shown in the Figure below.



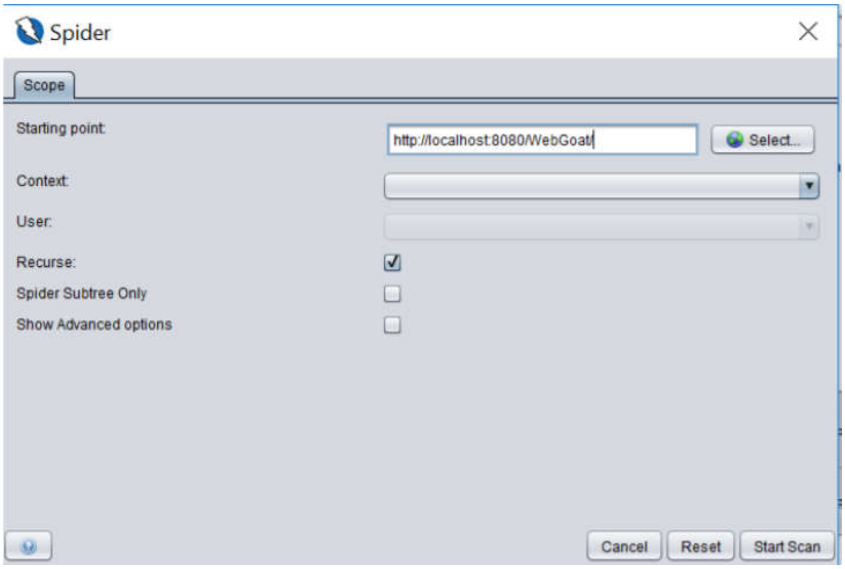
- Interpret your test results by following the instructions on page 6 of the Getting Started Guide.

Configure and Run a Spider with ZAP

- On the Information Window, click the green plus sign (+).
- Click Spider to create a Spider tab as shown in the Figure below



- Click the New Scan Button on the Spider tab. Run the Spider for the WebGoat application as shown in Figure below



- Interpret your test results by following the instructions on page 6 of the Getting Started Guide.

Answer to Review Question

What is SAST ?

SAST, which stands for Static Application Security Testing, is one of the white-box testing methods. SAST tools look at the source code or binaries of an application for coding or design flaws, which are indicative of security vulnerabilities, and even concealed malicious code.

SAST tools analyze the application from the “inside out” to test areas such as control structure, security, input validation, error handling, file update, and function parameter verification. They only work before the system is implemented as they do not require the application to be deployed or running.

What is DAST ?

A dynamic application security testing (DAST) tool is a program which communicates with a web application through the web front-end in order to identify potential security vulnerabilities in the web application and architectural weaknesses.[1] It performs a black-box test. Unlike static application security testing tools, DAST tools do not have access to the source code and therefore detect vulnerabilities by actually performing attacks.

DAST tools allow sophisticated scans, detecting vulnerabilities with minimal user interactions once configured with host name, crawling parameters and authentication credentials. These tools will attempt to detect vulnerabilities in query strings, headers, fragments, verbs (GET/POST/PUT) and DOM injection

What is fuzzing ?

Fuzzing (Fuzz testing) is a quality assurance technique used to discover coding errors and security loopholes in software, operating systems or networks. It involves inputting massive amounts of random data, called fuzz, to the test subject in an attempt to make it crash. If a vulnerability is found, a

software tool called a fuzzer can be used to identify potential causes. Fuzz testing was originally developed by Barton Miller at the University of Wisconsin in 1989.

Fuzzers work best for discovering vulnerabilities that can be exploited by buffer overflow, DOS (denial of service), cross-site scripting and SQL injection.

What is spidering ?

A web crawler (also known as a web spider or web robot) is a program or automated script which browses the World Wide Web in a methodical, automated manner. This process is called spidering or Web crawling.

Web spiders are the most powerful and useful tools developed for both good and bad intentions on the internet. Many legitimate sites, in particular search engines, use spidering as a means of providing up-to-date data.