

Installation

Mac

[URL](#) - does not seem to require setting up the DB manually

- Install Java as a prerequisite

Sonar & Sonar Scanner

```
brew install sonar  
brew install sonar-scanner
```

ENV Vars

In .zshrc, or .bashrc

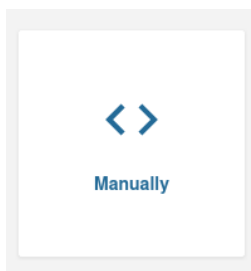
```
export SONAR_HOME=/usr/local/Cellar/sonar-scanner/{version}/libexec  
export SONAR=$SONAR_HOME/bin export PATH=$SONAR:$PATH
```

Create Project in SonarQube

Provide info and generate a token

Project Key & Display Name

1. <http://localhost:9000>
2. Click **Create a new project** button [in v9.6.1 it is a tile that looks like this]



Then enter:

- a. Project Key
 - b. Display Name
- [can be identical, but display name can have spaces for better viewing]
3. Click **Setup** button

Generate Token

Under Provide a token

1. Select Generate a token
2. Give your token a name
3. Click the Generate button, and
4. Click Continue.

token_test1: **sqp_021419b98f11c1e462354a63b221e33211c5579c**

Get / Generate Sonar Commands for Running Analysis

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

SalesProject master

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token ✓ SalesProduct:feb2c84dcaaf080d46ac52214edd1b32f8784b7a

2 Run analysis on your project

What is your build technology?

Maven Gradle .NET Other (for J5, TS, Go, Python, PHP, ...)

What is your OS?

Linux Windows macOS

Download and unzip the Scanner for macOS

And add the bin directory to the PATH environment variable

Download

Execute the Scanner from your computer

Running a SonarQube analysis is straightforward. You just need to execute the following commands in your project's folder.

```
sonar-scanner \  
-Dsonar.projectKey=SalesProject \  
-Dsonar.sources=. \  
-Dsonar.host.url=http://172.20.10.2:9000 \  
-Dsonar.login=feb2c84dcaaf080d46ac52214edd1b32f8784b7a
```

Copy

Please visit the official documentation of the Scanner for more details.

1. Select your project's main language under Run analysis on your project, and Operating System.
2. Download Sonar Scanner commands to execute a Scanner on your code.

Here the example is for Mac OS.

cd <project_fodler>

```
sonar-scanner \  
-Dsonar.projectKey=<project_key> \  
-Dsonar.sources=. \  
-Dsonar.host.url=http://localhost:9000 \  
-Dsonar.login=<token>
```

```
mvn clean verify sonar:sonar \
  -Dsonar.projectKey=test1_key \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.login=sqp_021419b98f11c1e462354a63b221e33211c5579c
```

Add a **sonar-project.properties** file in your project directory

```
sonar.python.coverage.reportPaths=../coverage.xml
# unique project identifier (required)
sonar.projectKey=SalesProject
# project metadata (used to be required, optional since SonarQube 6.1)
sonar.projectName=SalesProject
sonar.projectVersion=1.0
# path to source directories (required)
sonar.sources=../src
sonar.sources=src
# path to test source directories (optional)
#sonar.tests=tests
```

Ubuntu Linux

Install PostgreSQL

[URL](#) - on 20.04 but works on 22.04 as well

[URL](#) - How to List Databases and Tables in PostgreSQL Using psql

Install SonarQube

Download SonarQube - [URL](#), for manual download.

Installed community edition: sonarqube-9.6.1.59531

Install SonarQube on Ubuntu 20.04 LTS

[URL](#) - problem is that after version 8 they removed sonar.sh and expect you to start sonar with systemd.

Solution [found here](#). Of course, you have to adjust the paths here to match your Java installation and the location of the sonar-application-<OUR SONARQUBE VERSION> file:

ExecStart=/bin/nohup /opt/java/bin/java -Xms32m -Xmx32m

-Djava.net.preferIPv4Stack=true -jar

/opt/sonarqube/lib/sonar-application-<YOUR SONARQUBE VERSION>.jar

StandardOutput=**journal**

Don't use "syslog", it is deprecated.

This is for the Developer edition - switch home directory of user sonar

sudo usermod **-d** /opt/sonarqubedev **sonar**

How to Install SonarQube on Ubuntu 22.04 LTS

[URL](#) - similar issue; it does work with the version they are installing - SonarQube 8.9

I managed to get the latest SonarQube version, 9.6.1, to run with this sonar.service file (note that ExecStart is all on one line):

```
[Unit]
Description=SonarQube service
After=syslog.target network.target

[Service]
Type=simple
User=sonar
Group=sonar
PermissionsStartOnly=true
ExecStart=/usr/bin/nohup /usr/lib/jvm/java-11-openjdk-amd64/bin/java
-Xms32m -Xmx4000m -Djava.net.preferIPv4Stack=true -jar
/opt/sonarqube/lib/sonar-application-9.6.1.59531.jar
StandardOutput=journal
LimitNOFILE=131072
LimitNPROC=8192
TimeoutStartSec=5
Restart=always
SuccessExitStatus=143

LimitNOFILE=65536
LimitNPROC=4096

[Install]
WantedBy=multi-user.target
```

Operating the Server

The [real documentation is here](#).

- How to run it without SystemD
- How to run it **with** SystemD

Another section in the sidebar is the ENV Variables. That is also pretty extensive.

Token on Mac: sqp_8205cf9e6ee0525db3c7fe1b5f83f6802a17fc07

Token name: test_security_with_sonar_1

Bugs

Elasticsearch Cannot Start and Keeps Crashing

You constantly see this message in sonar.log: Waiting for Elasticsearch to become ready. Then you see how SonarQube stops and starts multiple times with the fans of the CPUs blowing at max.

Solution

[URL](#)

Summarized version

#1. Increase Heap Size

Linux: [/opt/sonarqube/conf/](#)**sonar.properties**

```
sonar.web.javaAdditionalOpts=-Xmx2G
sonar.ce.javaAdditionalOpts=-Xmx6G -XX:+HeapDumpOnOutOfMemoryError
sonar.search.javaAdditionalOpts=-Xmx6G -Xms6G
-XX:+HeapDumpOnOutOfMemoryError -Dnode.store.allow_mmap=false
```

#2. Delete Data Directory

Also deleted elastic search data directory: [/opt/sonarqube/data/](#)**es7**

Sonar has rebuilt the index and everything works.

[There is a lock in that directory. You may try deleting it first and restarting SonarQube. If that does not work, delete the entire directory as show above. If you succeed on the first try, then SonarQube won't have to rebuild the Elasticsearch index and SonarQube will start faster.]

Downloads

Developer Edition

[URL](#)

Run SonarQube

[URL](#)

Installation Guide

[URL](#)

Paid Plugin - 450 Euro - Free Trial Available

[URL](#) - really nice; adds two menus under the More top level menu => you can do CWE or OWASP type of reporting after a scan and export to PDF - using a button on the top right.

SONARSOURCE SONARQUBE UP TO 7.7 PROJECT LINK CROSS SITE SCRIPTING

[URL](#)

... The attack technique deployed by this issue is [T1059.007](#) according to MITRE ATT&CK. [[note this T-number; for the meta info, look at the right sidebar](#)]

SOFTWARE SUPPLY CHAIN EXPLOITATION PART 1

[URL](#)

OWASP - SonarQube Website

[URL](#)

Relevant but NOT Current

Attack and Defense Strategies with MITRE ATT&CK Framework

[URL](#) - 2021-04-15

- Quick and Dirty ATT&CK and Shield explanation

MITRE Shield Active Defense - How to Use Shield Techniques to Stop Targeted Ransomware Attacks

[URL](#) - 2020;

Selected MITRE Enterprise Techniques

- [T1190](#) - [Exploit Public-Facing Application](#)

<https://attack.mitre.org/techniques/T1190/>

Here search for "**SQL Injection**"; **G0007** - **APT28** (first listed procedure example)) has it and many more procedures below, but this one fits the bill really well. This is definitely detected by SonarQube - see your resources for a code snippet that has user input that is not sanitized ...

- [T1189](#) - [Drive-by Compromise](#) - aka **Cross-Site Scripting** (one of the varieties)

<https://attack.mitre.org/techniques/T1189/>

Here search for "**cross-site scripting**"; under Drive-By Compromise (first bullet);

- [T1078](#) - [Valid Accounts](#)

<https://attack.mitre.org/techniques/T1078/004/>

Here search for "identity"; Only the Cloud Accounts section has this word present. The rest of them mostly deal with credentials (may have to update the word "identity" in the outline above. Example: SonarQube detects credentials in configuration files - either in clear text or just hashed; or in source code - hardcoded.

About CAPEC

[URL](#) - compared ATT&CK. CAPEC; it focuses on app security

Security Related Rules

These are the [rules SonarQube 9.5 covers](#). The top sections of the page are very important.

What to expect from security-related rules

The following should be more than **sufficient for the purposes of the course**.

Security Injection Rules - [URL](#) - SonarQube Docs

- [CWE-89](#): SQL Injection; [Improper Neutralization of Special Elements used in an SQL Command \('SQL Injection'\)](#)
- [CWE-79](#): Cross-site Scripting
- [CWE-94](#): Code Injection - maps to [T1055](#) (parent) - [Process Injection](#) in MITRE ATT&CK

Security Configuration Rules

- [CWE-1004](#): Sensitive Cookie Without 'HttpOnly' Flag
- [CWE-297](#): Improper Validation of Certificate with Host Mismatch
- [CWE-327](#): Use of a Broken or Risky Cryptographic Algorithm

Which security-standards are covered?

Our security rules are classified according to well-established security-standards such as:

- [CWE](#): SonarQube is a CWE compatible product [since 2015](#).
- [OWASP Top 10](#))
- [SANS Top 25](#) - outdated

The standards to which a rule relates will be listed in the **See** section at the bottom of the rule description. More generally, you can search for a rule on rules.sonarsource.com:

- [Java-vulnerability-issue-type](#): all vulnerability rules for Java language.
- [Java-hotspots-issue-type](#): all security-hotspot rules for Java language.
- [Java-tag-injection](#): all security-injection rules for Java language.

Explanations

- CWE is MITRE's [Common Weakness Enumeration](#). The references in the format CWE-XXXX point to exploitation techniques. The **trick** here is to map these references to ATT&CK techniques and very likely lower level explanations. At this point I believe CWE-XXXX are too granular => dig into the ATT&ACK references at lower levels to find the corresponding top-level techniques.
- The **OOTB rules** in the SonarQube security docs described above are **covering Java. Stick with those for the purposes of the course** and just mention there are plug-ins for other languages.
- [OWASP](#) maps fairly well to CWE
 - Cross Site Scripting **A7** = [CWE-79](#): Cross-site Scripting = [T1189](#): Drive-by Compromise ([docs](#))

| | | Java | C# | Python | PHP | JS | TS | |
|---------------------------------------|------------|------|----|--------|-----|----|----|--|
| Injection | A1 | | | | | | | |
| Broken Authentication | A2 | | | | | | | |
| Sensitive Data Exposure | A3 | | | | | | | |
| XML External Entities (XXE) | A4 | | | | | | | |
| Broken Access Control | A5 | | | | | | | |
| Security Misconfiguration | A6 | | | | | | | |
| Cross-Site Scripting XSS | A7 | | | | | | | |
| Insecure Deserialization | A8 | | | | | | | |
| Components with Known Vulnerabilities | A9 | | | | | | | |
| Insufficient Logging & Monitoring | A10 | | | | | | | |

HotSpot
 Vulnerability

Plugin Version Matrix

[URL](#)

Malware Detection—Discovering Cross-Site Scripting Attacks

POSTED BY BRIAN LAING ON NOV 9, 2017

Nearly everyone has at some point had their web browser pop up a message that says something like “Your PC is Infected . . . Click Here to remove the virus”. That message is likely the result of a cross-site scripting (XSS) attack, and clicking on the link will connect to a site that installs malware, or encourages the victim to pay for fraudulent virus removal services.

The above example is just one form of XSS attack. In reality, there are numerous types of XSS attacks and cybercriminals frequently use them to commit their crimes. They pose a major challenge for malware detection systems.

Fortunately, new technologies can identify XSS attacks and the infections they cause.

What is a Cross Site Scripting Attack?

Cross-Site Scripting (XSS) attacks are a type of injection attack where cybercriminals deliver malicious script or code to a client browser, often via a vulnerable web application. In this type of attack, cybercriminals trick users' browsers into executing malicious code. A classic example is causing a browser to display a popup with a link to a website that installs malware. In other cases, an XSS attack will cause a victim's browser to send confidential data or cookies containing login credentials to the attacker.

An XSS attack can happen when a web application allows users to input information but fails to validate that input. This vulnerability allows cybercriminals to enter malicious code such as JavaScript into a form or search box, and the victim's browser will execute that code.

For example, imagine a scenario where a web application allows visitors to enter a comment. Unless the application filters it out, an attacker can enter a comment that includes malicious JavaScript. Any browser that subsequently loads that page will read the comment and execute the embedded JavaScript. Because browsers don't normally display JavaScript, it will be invisible to users and administrators that are viewing the page. The code will execute without their knowledge.

....

XSS attacks create the ideal environment for attackers to escalate an initial foothold to a more extensive and damaging intrusion. With social engineering and the browser capabilities listed above, cybercriminals can use an XSS attack to execute sophisticated operations including cookie theft, keylogging, user impersonation, session hijacking, data theft, and many other malicious activities.

My thoughts on using the MITRE ATT&CK framework for SIEM detection's

[URL](#)

- Explains MITRE ATT&CK framework
- Advises on how it should be used as part of your SIEM detection strategy

The majority of SIEM platforms come with good out of the box use cases, such as –

- [Azure Sentinel Detections](#)
 - [QRadar Use Case Manager](#)
 - [Elastic Detection Rules](#)
-

SonarQube Tutorial

Run a scan on Win

<https://www.loginradius.com/blog/engineering/sonarqube/>

Taking the angst out of SAST analysis

[URL](#)

Today most SAST (Static Application Security Testing) tools are still owned and run outside the development team. Results are delivered intermittently, and if the past is a guide, no one really owns code security. **[use this in the What Is? clip]**

- The security team can't own it because they can't impact it; they can't change the code.
- And developers can't own it because they can't take ownership of it. They don't own the rules, the timelines, or the reports, so they can't own the result.

Fortunately, you have the tools in hand today to break that pattern and shift code security - like code quality and reliability - to developers. With the SonarSource model:

- **SAST is integrated into the development workflow** by making it part of the tools developers already use: SonarQube and SonarCloud analysis.
- That means **quick security feedback**, and - because the code is still fresh in mind - **efficient, effective fixes**.
- Giving developers control of **SAST tooling lets them take ownership of code security**, so pushback turns into pride of workmanship.

About SonarQube

[URL](#) - used this to create the slide with the description in their own words;

Scan Source Code

[URL](#) - use this for architecture and to describe what SonaQube does when it scans source code
Also shows the properties file and how to configure it

10 Most Common Java Vulnerabilities You Need to Prevent

[URL](#)

Sonarqube Download, Install, Configure and Scan Codes for Vulnerabilities | Hacknikal

[URL](#)

- Sample setup
- DB setup - postgres
- Sample scan - does not show security issues but there are some detected
- With **PHP**, not Java

XSS Tutorial

[URL](#) - full explanation of XSS with examples that progress and become more malicious

Vulnerable Code Snippets

XSS In Java

[URL](#) - nice example, but do the same with Spring MVC instead of using Servlets
=> see if you can scan the code with SonarQube

XSS in Java - GiHub

[URL](#) - very simple REST controller that exemplifies XSS
NOT DETECTED by SonarQybe

Authentication Bypass - probably = Broken Authentication (A2)

[URL](#) - same repo as above; the example is partial; shows some sort of Spring Security Filter Code

TODO: Try to find a better example of Authentication Bypass

SQL Injection - Java

[URL](#) - the example is also partial - just instructions;

TODO: Wrap in method within a class, e.g., a Spring MVC Controller

<https://www.sonarqube.org/features/multi-languages/java/index/Java-security-vulnerability-RSP-EC-2278.png>

Go to Security / Vulnerability tab:

<https://www.sonarqube.org/features/multi-languages/java/#vulnerability> - the png above is from there.

Also, check this [URL](#) - Database queries should not be vulnerable to injection attacks

Broken Authentication

[URL](#) - great Java examples; Spring based, with explanations.

XSS Cross Site Scripting Java Demo

[URL](#)

Blog with Source code; search for "expand" on the page to find where the code is.

[URL](#) - **NOT detected** by SonarQube

[URL](#) - RSPEC-5131, XSS

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks

OWASP

[URL](#) - example queries; categorization of vulnerability

Source Samples

Insecure Algorithm 1

https://raw.githubusercontent.com/JetBrains/jdk8u_jdk/master/src/share/classes/com/sun/security/ntlm/NTLM.java

Repo [URL](#)

A small portion of this JDK * code is on the SonarQube website in screenshot that shows Security Analysis in action.

- <https://www.sonarqube.org/features/security/> - main page
- <https://www.sonarqube.org/features/security/index/lightbox/Java-security-vulnerability-RSPEC-2278.png> - image on page

https://github.com/JetBrains/jdk8u_jdk/blob/master/src/share/classes/com/sun/security/ntlm/NTLM.java

Glossary

CWE - Common Weaknesses Enumeration (MITRE)

SAST - Static Application Security Testing

OWASP - Open Web Application Security Project

CVE - Common Vulnerabilities and Exposures