

**Раздел:** Безопасная разработка ПО appsec и внедрение DevSecOps.  
Модуль 2. Средства статического и динамического анализа ПО и инфраструктуры.

**Выполнил:** Александр Ганицев.

### **Задача**

Просканировать приложение на python с помощью SonarQube и заполнить отчёт о найденных уязвимостях.

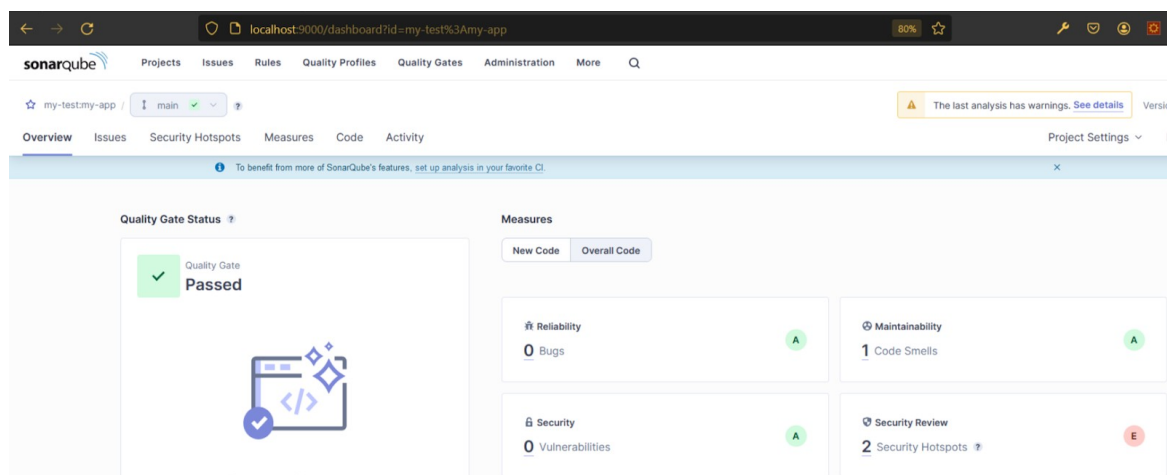
### **Что конкретно нужно сделать:**

1. Запустить локально sonarqube с базой данных postgresql по примеру из первого юнита.
2. Скачать из папки курса исходный код приложения на python – [скачать отсюда](#).
3. Просканировать его в SonarQube.
4. Получить результат и заполнить отчёт по найденным уязвимостям.
- 5.\* Исправить найденные уязвимости по рекомендации SonarQube, а также устранить все недочёты, которые будут обнаружены в колонке «Maintainability». Достаточно добавить в конец отчёта скриншот со страницы SonarQube, чтобы был виден проект, адрес в браузере, исправленные уязвимости.

\*(для получения максимального балла за задание).

### **Что должно быть в отчёте**

1. Нужно приложить скриншот со страницы SonarQube, чтобы был виден проект, адрес в браузере и обнаруженные уязвимости. Пример такого скрина ниже.



Здесь видим найденные Security Hotspots – 2

Какие уязвимости – Cross-Site Request Forgery (CSRF)

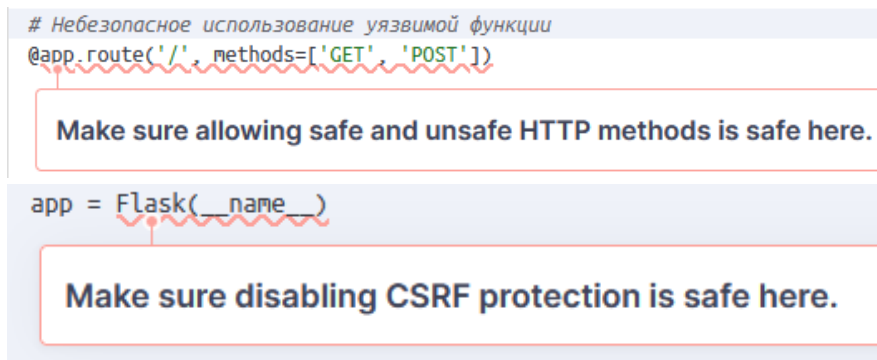
- Make sure allowing safe and unsafe HTTP methods is safe here.
- Make sure disabling CSRF protection is safe here

2. Описание уязвимости из отчёта. Например:

*A cross-site request forgery (CSRF) attack occurs when a trusted user of a web application can be forced, by an attacker, to perform sensitive actions that he didn't intend, such as updating his profile or sending a message, more generally anything that can change the state of the application.*

*The attacker can trick the user/victim to click on a link, corresponding to the privileged action, or to visit a malicious web site that embeds a hidden web request and as web browsers automatically include cookies, the actions can be authenticated and sensitive.*

3. Скриншоты, на которых видно, где была обнаружена уязвимость. Например,



4. Рекомендации SonarQube по устранению уязвимостей. Например,

For a [Flask](#) application,

- the `CSRFProtect` module should be used (and not disabled further with `WTF_CSRF_ENABLED` set to `false`):

```
app = Flask(__name__)
csrf = CSRFProtect()
csrf.init_app(app) # Compliant
```

- and it is recommended to not disable the CSRF protection on specific views or forms:

```
@app.route('/example/', methods=['POST']) # Compliant
def example():
    return 'example '

class unprotectedForm(FlaskForm):
    class Meta:
        csrf = True # Compliant

    name = TextField('name')
    submit = SubmitField('submit')
```

5. Скриншот со страницы SonarQube, на котором виден проект, адрес в браузере, исправленные уязвимости. Например,

← → ↺

localhost:9000/dashboard?id=my-test%3Amy-app

80% ☆ 🔍 ⚙️ 👤 ⚠️

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMoreQ

☆ my-testmy-app /

main ✓ ↕

The last analysis has warnings. [See details](#)

Versi

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project Settings ▾


To benefit from more of SonarQube's features, set up analysis in your favorite CI.

×

Quality Gate Status ?

✓ Quality Gate

Passed



Enjoy your sparkling clean code!

Some Quality Gate conditions on New Code were ignored because of the small number of New Lines

Measures

New CodeOverall Code

New Code: Since January 25, 2024 Started 1 hour ago

Reliability

0 New Bugs

A

Maintainability

0 New Code Smells

A

Security

0 New Vulnerabilities

A

Security Review

0 New Security Hotspots ?

A

Coverage

0.0% Coverage

Coverage on 6 New Lines to cover

Duplications

0.0% Duplications

Duplications on 6 New Lines

## Выполнение.

1. Входим в Docker и создаём образы:

```
alexandrganitev@Alexandrs-MBP sonar % docker login
Authenticating with existing credentials...
Login Succeeded
```

```
alexandrganitev@Alexandrs-MBP sonar % docker pull sonarqube:latest
latest: Pulling from library/sonarqube
31bd5f451a84: Pull complete
26611c45681a: Pull complete
7657bba016af: Pull complete
e5c532b68318: Pull complete
65e11da75820: Pull complete
6e7b8f52fd19: Pull complete
3344630584a8: Pull complete
4f4fb700ef54: Pull complete
```

```
alexandrganitev@Alexandrs-MBP sonar % docker pull postgres:latest
latest: Pulling from library/postgres
c57ee5000d61: Pull complete
81b575116500: Pull complete
e12fff61d996: Pull complete
50a849db7317: Pull complete
432dd17f42df: Pull complete
a1f5bcbba6b6: Pull complete
6e501216828b: Pull complete
ea24c7671c3d: Pull complete
b7a5cd7c9b9a: Pull complete
db7d78d9f46e: Pull complete
8c786fbf8634: Pull complete
2831031f2a0e: Pull complete
75c5b068b243: Pull complete
9590d9e20e85: Pull complete
```

Образы SonarQube и Postgres были добавлены в Docker Hub:

[sonarqube](#)

2defe7e3bff0 

latest

[postgres](#)

b0b90c1d9579 

latest

## 2. Создаём виртуальную сеть:

```
alexandrganitev@Alexandrs-MBP sonar % docker network create host-net  
cd4af4b3441c2b53d5ab28778738589bd7aee0041f390f357d73177b7a6d5813
```

## 3. Создаём директории:

```
drwxr-xr-x  2 alexandrganitev  staff   64  9 Feb 23:05 postgresql  
drwxr-xr-x  2 alexandrganitev  staff   64  9 Feb 23:05 postgresql_data  
drwxr-xr-x  2 alexandrganitev  staff   64  9 Feb 23:04 sonarqube_data  
drwxr-xr-x  2 alexandrganitev  staff   64  9 Feb 23:05 sonarqube_extensions  
drwxr-xr-x  2 alexandrganitev  staff   64  9 Feb 23:05 sonarqube_logs
```

## 4. Запускаем наши образы:

### Postgres:

```
docker run --hostname=postgres --env=POSTGRES_USER=sonar --  
env=POSTGRES_PASSWORD=w31coMe!  
--env=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/  
lib/postgresql/16/bin --env=PGDATA=/var/lib/postgresql/data  
--volume=/Users/alexandrganitev/.docker/sonar/postgresql:/var/lib/postgresql  
--volume=/Users/alexandrganitev/.docker/sonar/postgresql_data:/var/lib/  
postgresql/data --volume=/var/lib/postgresql/data -p 5432:5432 --name  
postgres --network host-net -d postgres:latest
```

		postgres	postgres:latest	Running	0.08%	5432:5432
906f7b171e75						

```
2024-02-09 23:23:16 server started  
2024-02-09 23:23:17 CREATE DATABASE
```

### Sonar:









```
docker run --hostname=sonar --user=sonarqube --  
env=SONAR_JDBC_URL=jdbc:postgresql://postgres:5432/sonar --  
env=SONAR_JDBC_USERNAME=sonar --  
env=SONAR_JDBC_PASSWORD=w31coMe!  
--env=PATH=/opt/java/openjdk/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/  
usr/bin:/sbin:/bin --env=JAVA_HOME=/opt/java/openjdk --  
env=LANG=en_US.UTF-8 --env=LANGUAGE=en_US:en --
```



```

env=LC_ALL=en_US.UTF-8 --env=JAVA_VERSION=jdk-17.0.9+9 --
env=DOCKER_RUNNING=true
--env=SONARQUBE_HOME=/opt/sonarqube --
env=SONAR_VERSION=10.3.0.82913
--env=SQ_DATA_DIR=/opt/sonarqube/data --
env=SQ_EXTENSIONS_DIR=/opt/sonarqube/extensions --
env=SQ_LOGS_DIR=/opt/sonarqube/logs
--env=SQ_TEMP_DIR=/opt/sonarqube/temp
--volume=/Users/alexandrganitev/.docker/sonar/sonarqube_data:/opt/
sonarqube/data
--volume=/Users/alexandrganitev/.docker/sonar/sonarqube_extensions:/opt/
sonarqube/extensions
--volume=/Users/alexandrganitev/.docker/sonar/sonarqube_logs:/opt/
sonarqube/logs --workdir=/opt/sonarqube -p 9000:9000 --runtime=runc --
name sonar --network host-net -d sonarqube:latest

```

		<b>postgres</b> 906f7b171e75 	<a href="#">postgres:latest</a>	Running	0%	<a href="#">5432:5432</a> 
		<b>sonar</b> d5e3059563f6 	<a href="#">sonarqube:latest</a>	Running	3.51%	<a href="#">9000:9000</a> 

Заметка: Контейнеры были созданы, но если контейнер Postgres запускался и работал, то контейнер Sonar запускался и останавливался через 15 секунд, не найдя базу данных для подключения. Пришлось переподключить их к общей сети.

```

alexandrganitev@Alexandrs-MBP sonar % docker network create sonarqube-network
Error response from daemon: network with name sonarqube-network already exists
alexandrganitev@Alexandrs-MBP sonar % docker network connect sonarqube-network postgres
alexandrganitev@Alexandrs-MBP sonar % docker network connect sonarqube-network sonar

```

Результат, на моей системе не возможно запустить Sonar:

2024-02-12 00:27:28 Caused by:

com.zaxxer.hikari.pool.HikariPool\$PoolInitializationException: Failed to initialize pool: FATAL: password authentication failed for user "sonar"

2024-02-12 00:27:28 at

com.zaxxer.hikari.pool.HikariPool.throwPoolInitializationException(HikariPool.java:596)

2024-02-12 00:27:28 at  
com.zaxxer.hikari.pool.HikariPool.checkFailFast(HikariPool.java:582)  
2024-02-12 00:27:28 at  
com.zaxxer.hikari.pool.HikariPool.<init>(HikariPool.java:100)  
2024-02-12 00:27:28 at  
com.zaxxer.hikari.HikariDataSource.<init>(HikariDataSource.java:81)  
2024-02-12 00:27:28 at  
org.sonar.db.DefaultDatabase.createHikariDataSource(DefaultDatabase.java:  
159)  
2024-02-12 00:27:28 at  
org.sonar.db.DefaultDatabase.initDataSource(DefaultDatabase.java:148)  
2024-02-12 00:27:28 at  
org.sonar.db.DefaultDatabase.start(DefaultDatabase.java:126)  
2024-02-12 00:27:28 ... 48 common frames omitted  
2024-02-12 00:27:28 Caused by: org.postgresql.util.PSQLException:  
FATAL: password authentication failed for user "sonar"  
2024-02-12 00:27:28 at  
org.postgresql.core.v3.ConnectionFactoryImpl.doAuthentication(Connection  
FactoryImpl.java:693)  
2024-02-12 00:27:28 at  
org.postgresql.core.v3.ConnectionFactoryImpl.tryConnect(ConnectionFactor  
yImpl.java:203)  
2024-02-12 00:27:28 at  
org.postgresql.core.v3.ConnectionFactoryImpl.openConnectionImpl(Connect  
ionFactoryImpl.java:258)  
2024-02-12 00:27:28 at  
org.postgresql.core.ConnectionFactory.openConnection(ConnectionFactory.j  
ava:54)  
2024-02-12 00:27:28 at  
org.postgresql.jdbc.PgConnection.<init>(PgConnection.java:263)  
2024-02-12 00:27:28 at  
org.postgresql.Driver.makeConnection(Driver.java:443)  
2024-02-12 00:27:28 at org.postgresql.Driver.connect(Driver.java:297)  
2024-02-12 00:27:28 at  
com.zaxxer.hikari.util.DriverDataSource.getConnection(DriverDataSource.ja  
va:121)



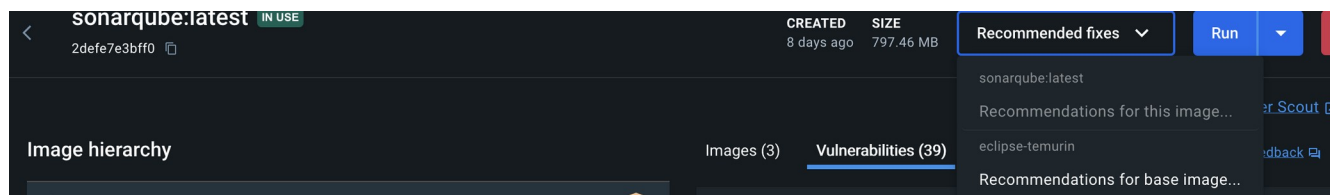
```
2024-02-12 00:27:28    at
com.zaxxer.hikari.pool.PoolBase.newConnection(PoolBase.java:359)
2024-02-12 00:27:28    at
com.zaxxer.hikari.pool.PoolBase.newPoolEntry(PoolBase.java:201)
2024-02-12 00:27:28    at
com.zaxxer.hikari.pool.HikariPool.createPoolEntry(HikariPool.java:470)
2024-02-12 00:27:28    at
com.zaxxer.hikari.pool.HikariPool.checkFailFast(HikariPool.java:561)
2024-02-12 00:27:28    ... 53 common frames omitted
2024-02-12 00:27:28 2024.02.11 21:27:28 INFO  web[]
[o.s.p.ProcessEntryPoint] Hard stopping process
2024-02-12 00:27:28 2024.02.11 21:27:28 INFO  app[][o.s.a.SchedulerImpl]
Process[Web Server] is stopped
2024-02-12 00:27:28 2024.02.11 21:27:28 WARN  app[]
[o.s.a.p.AbstractManagedProcess] Process exited with exit value
[ElasticSearch]: 143
2024-02-12 00:27:28 2024.02.11 21:27:28 INFO  app[][o.s.a.SchedulerImpl]
Process[ElasticSearch] is stopped
2024-02-12 00:27:28 2024.02.11 21:27:28 INFO  app[][o.s.a.SchedulerImpl]
SonarQube is stopped
```

## Выводы:

Пересоздание контейнеров из заново загруженных образов и перезагрузка системы не дали результатов. Читал в сети особенности конфигурации “docker run” команды, сверял с материалом данным преподавателем, провал разные варианты запуска, но тем не менее sonar включается на 15 секунд и останавливался.

Если запустить чистый sonar контейнер (Recommended fixes), то он прекрасно работает, именно настройки нашего описанного в модуле являются проблемными.

На попытки запуска, перенастройки и чтение потратил более 7 часов.



Принцип работы sonar мне понятен. Так же я сверился с коллегой и он мне показал, как данные два контейнера работают у него в Docker Desktop (спасибо Всеволод!).