Especialização em Engenharia de Software com DevOps

Marcos Eduardo

Universidade Estadual do Ceará - UECE
Centro de Ciências e Tecnologia - CCT

2023

UNIVERSIDADE ESTADUAL DO CEARÁ

Análise de Segurança de Projeto Java Utilizando Sonar em Container Docker

Roteiro

1. **Escolha do Sistema e Ferramenta**
2. **Preparação para Execução da Ferramenta**
3. **Execução da Ferramenta**
4. **Práticas de Mitigação**

Escolha do Sistema e Ferramenta

Escolha do Sistema e Ferramenta

https://github.com/marcoseduardoss/security-demo-projects
/demo-sonar-project (link)

Preparação para Execução da Ferramenta

Preparação para Execução da Ferramenta

Passos

1. Instalação docker
2. Criação Docker Compose para:
    1. Baixar imagem e criar container do SonarQube
    2. Baixar imagem e criar container do PostgreSQL
    3. Criar volumes para postgres e Sonar
    4. Criar uma rede para os dois containers: sonar-network
3. ...

Preparação para Execução da Ferramenta

Passos

1. Instalação docker
2. Criação Docker Compose para:
    1. Baixar imagem e criar container do SonarQube
    2. Baixar imagem e criar container do PostgreSQL
    3. Criar volumes para postgres e Sonar
    4. Criar uma rede para os dois containers: sonar-network
3. ...

docker-compose.yaml | novo 129 | morgan.py | novo 130 | novo 132 | novo 131

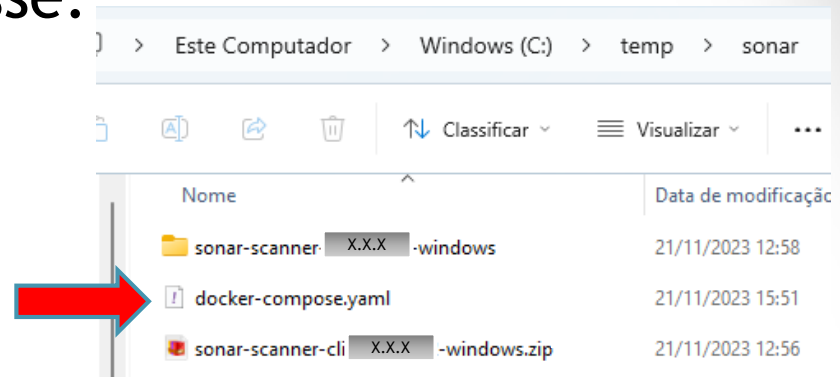
```yaml
version: "3.8"
services:
  sonarqube:
    image: sonarqube
    depends_on:
      - postgres_sonar
    ports:
      - "9000:9000"
    networks:
      - sonar-network
    environment:
      SONAR_JDBC_URL: jdbc:postgresql://postgres_sonar:5432/sonar
      SONAR_JDBC_USERNAME: sonar
      SONAR_JDBC_PASSWORD: sonar
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs
      - sonarqube_temp:/opt/sonarqube/temp
  postgres_sonar:
    image: postgres
    networks:
      - sonar-network
    environment:
      POSTGRES_USER: sonar
      POSTGRES_PASSWORD: sonar
      POSTGRES_DB: sonar
    volumes:
      - pgdata:/var/lib/postgresql/data

networks:
  sonar-network:
    driver: bridge

volumes:
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
  sonarqube_temp:
  pgdata:
```

Preparação para Execução da Ferramenta

3. Execução do Docker Compose

- Comando para execução do docker composse:
  docker-compose up –d

Obs.: A flag -d é para liberar o terminal.




```
C:\temp\sonar>docker compose up -d
[+] Running 8/8
 ✓ sonarqube 7 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled                                  102.2s
   ✓ 43f89b94cd7d Pull complete                                                            17.6s
   ✓ a4452d37e1e4 Pull complete                                                            23.4s
   ✓ cae6cc00f059 Pull complete                                                            37.3s
   ✓ dfaec5da5e63 Pull complete                                                            37.3s
   ✓ eb7dcc43773c Pull complete                                                            37.4s
   ✓ af715a573fc3 Pull complete                                                            95.8s
   ✓ c193b65af6ff Pull complete                                                            95.9s
[+] Running 8/8
 ✓ Network sonar_sonar-network          Created                                            0.3s
 ✓ Volume "sonar_sonarqube_data"        Created                                            0.1s
 ✓ Volume "sonar_sonarqube_extensions"  Created                                            0.0s
 ✓ Volume "sonar_sonarqube_logs"        Created                                            0.0s
 ✓ Volume "sonar_sonarqube_temp"        Created                                            0.0s
 ✓ Volume "sonar_pgdata"                Created                                            0.0s
 ✓ Container sonar-postgres_sonar-1     Started                                            5.2s
 ✓ Container sonar-sonarqube-1          Started                                            6.2s
```

Preparação para Execução da Ferramenta

4. Corrigir memoria da máquina virtual, muito baixa para executar o SonarQube:

```
2021.11.11 18:50:54 INFO  es[][o.e.n.Node] starting ...
2021.11.11 18:50:54 INFO  es[][o.e.t.TransportService] publish_address {127.0.0.1:34393}, bound_addresses {127.0.0.1:34393}
2021.11.11 18:50:55 INFO  es[][o.e.b.BootstrapChecks] explicitly enforcing bootstrap checks
ERROR: [1] bootstrap checks failed. You must address the points described in the following [1] lines before starting Elasticsearc
bootstrap check failure [1] of [1]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]
ERROR: Elasticsearch did not exit normally - check the logs at /opt/sonarqube/logs/sonarqube.log
2021.11.11 18:50:55 INFO  es[][o.e.n.Node] stopping ...
2021.11.11 18:50:55 INFO  es[][o.e.n.Node] stopped
2021.11.11 18:50:55 INFO  es[][o.e.n.Node] closing ...
2021.11.11 18:50:55 INFO  es[][o.e.n.Node] closed
```

Passos para correção:

- Comando para acessar a máquina virtual que executa o Docker
  wsl -d docker-desktop

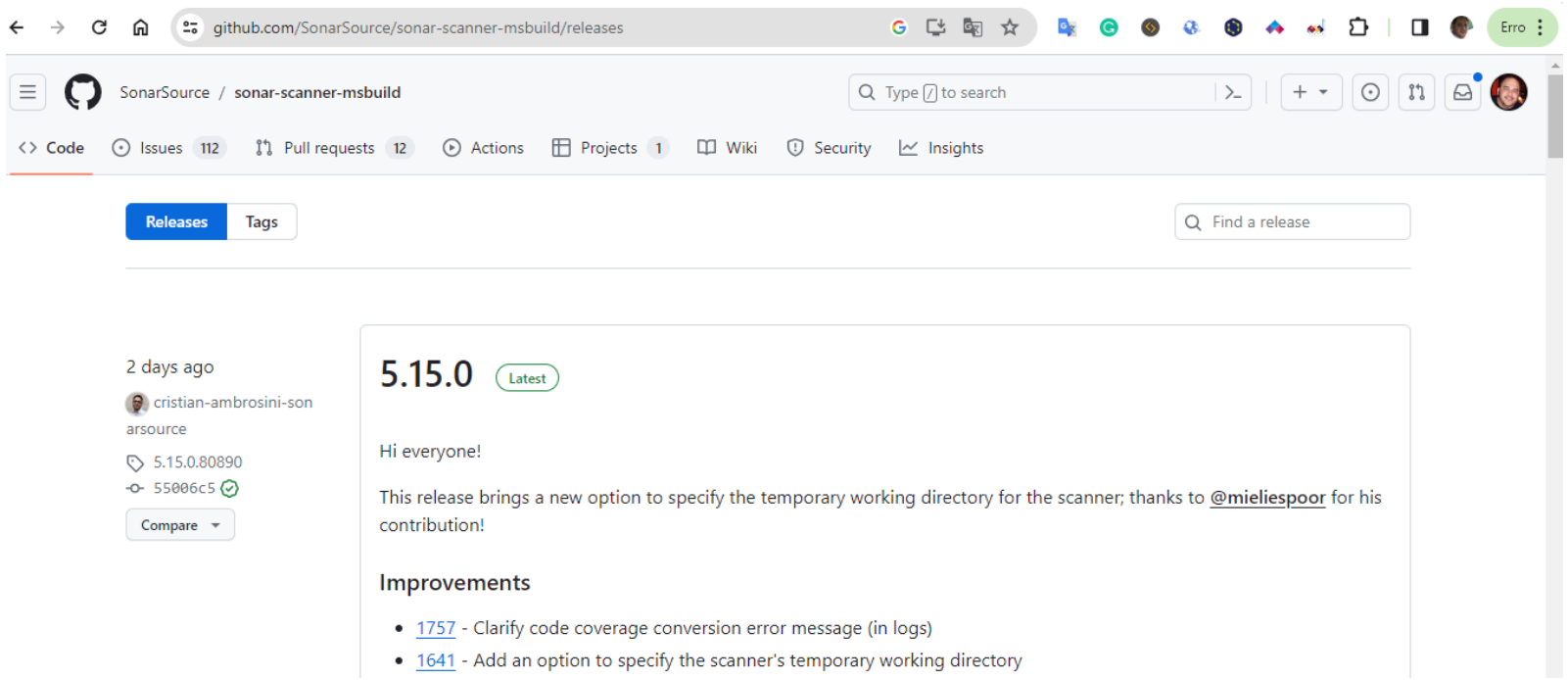
- Comando para aumentar a memória:
  sysctl -w vm.max_map_count=262144

Preparação para Execução da Ferramenta

5. Configurar SonarScanner

5.1. Baixar SonarScanner for Maven (escolha a mesma versão do SonarQube implantado)

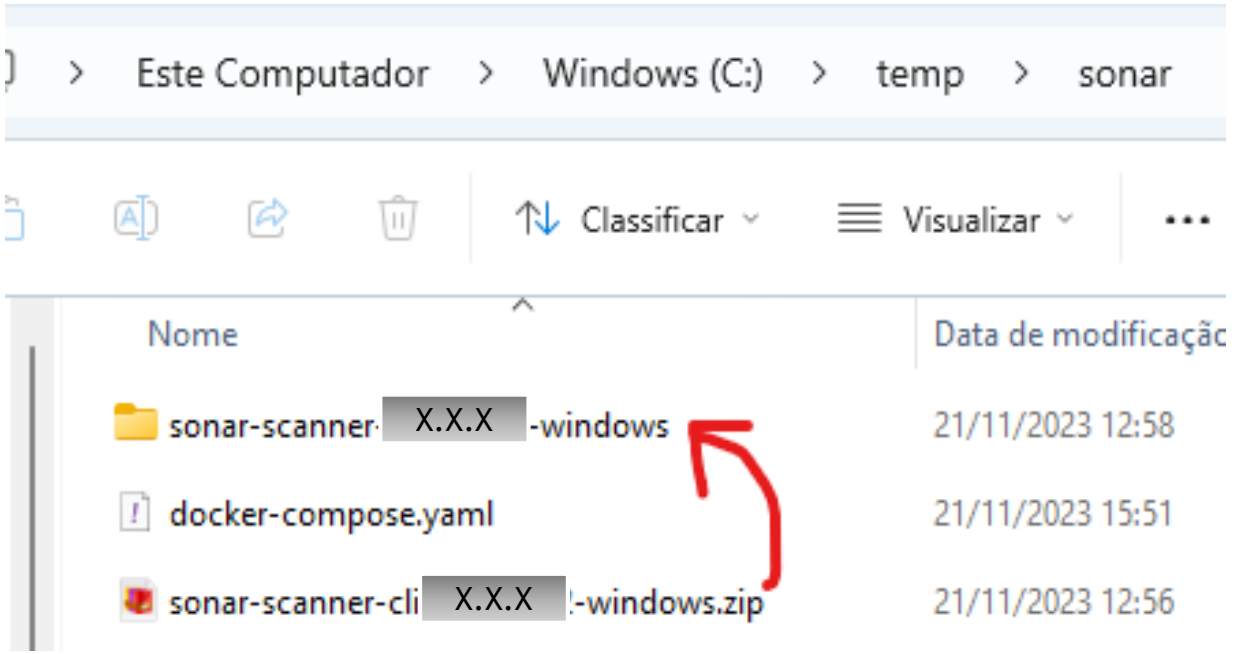
URL para Download:
https://github.com/SonarSource/sonar-scanner-msbuild/releases



Preparação para Execução da Ferramenta

6. Configurar SonarScanner

6.1. Após baixar o Sonar Scanner, faça o unzip do mesmo para uma pasta de sua preferência:

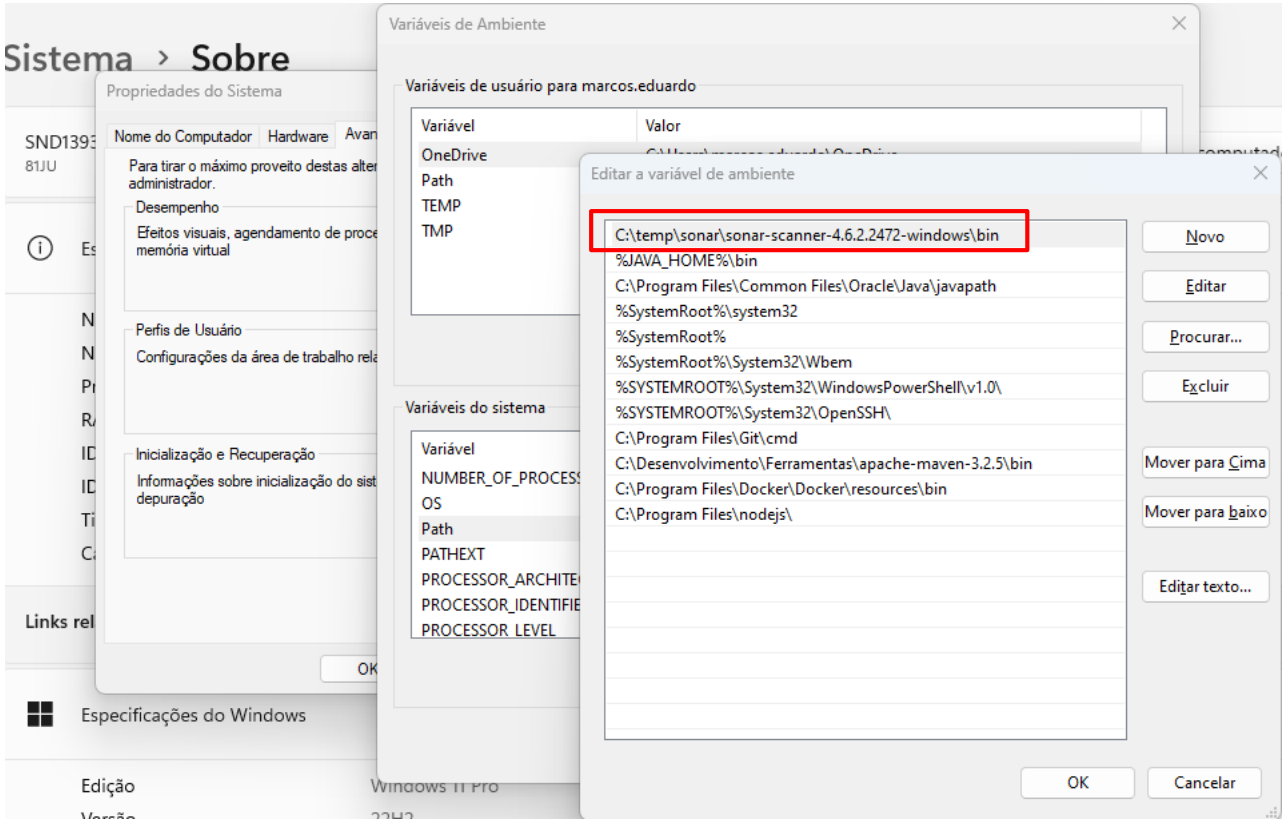


Preparação para Execução da Ferramenta

7. Configurar SonarScanner

7.1. Configuração de Variável de ambiente

Mesmo utilizando Docker é necessário configurar a variável de ambiente de localização do **path** do SonarScanner



Execução da Ferramenta

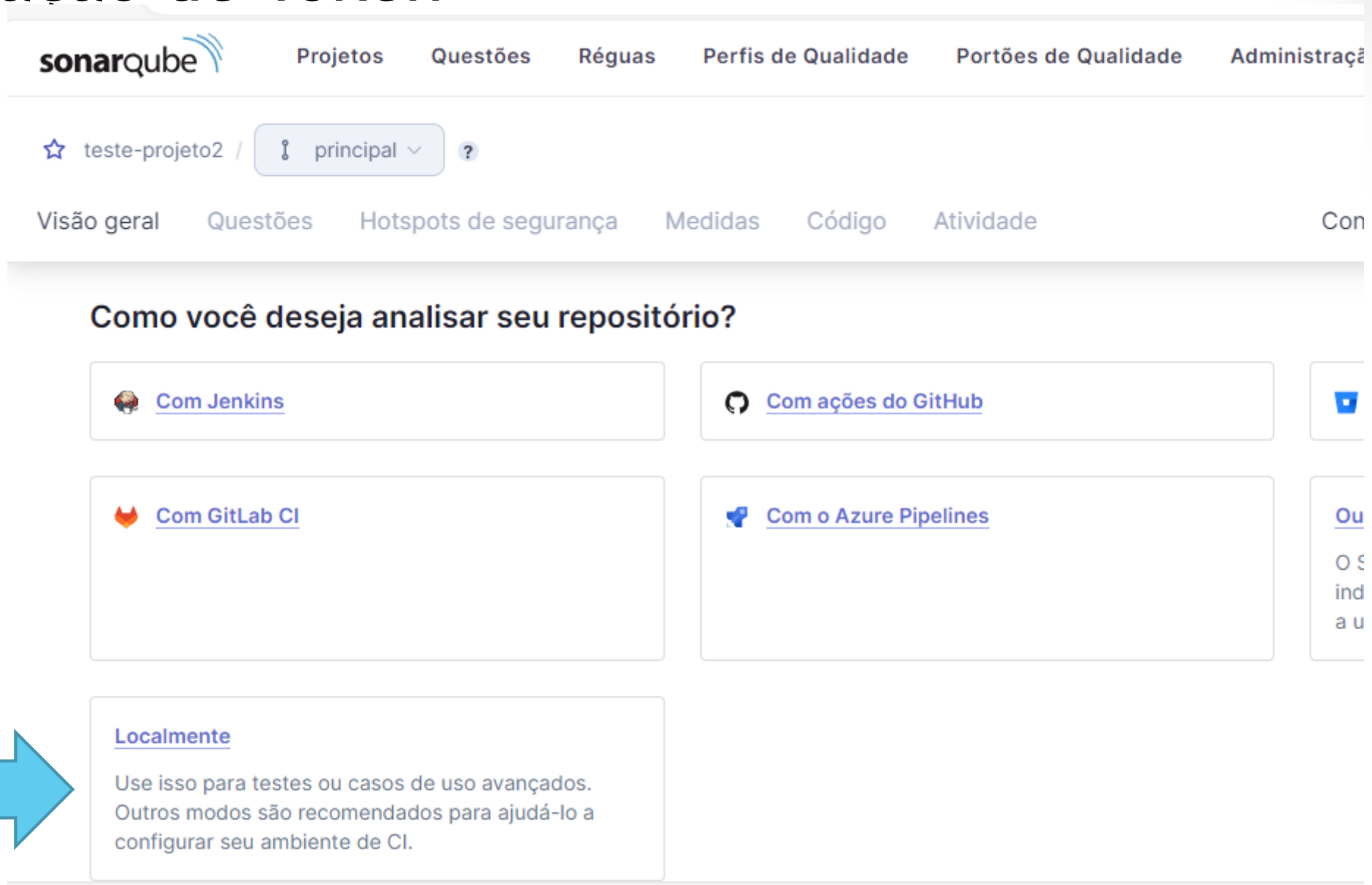
Criação de Projeto

Configuração do Projeto

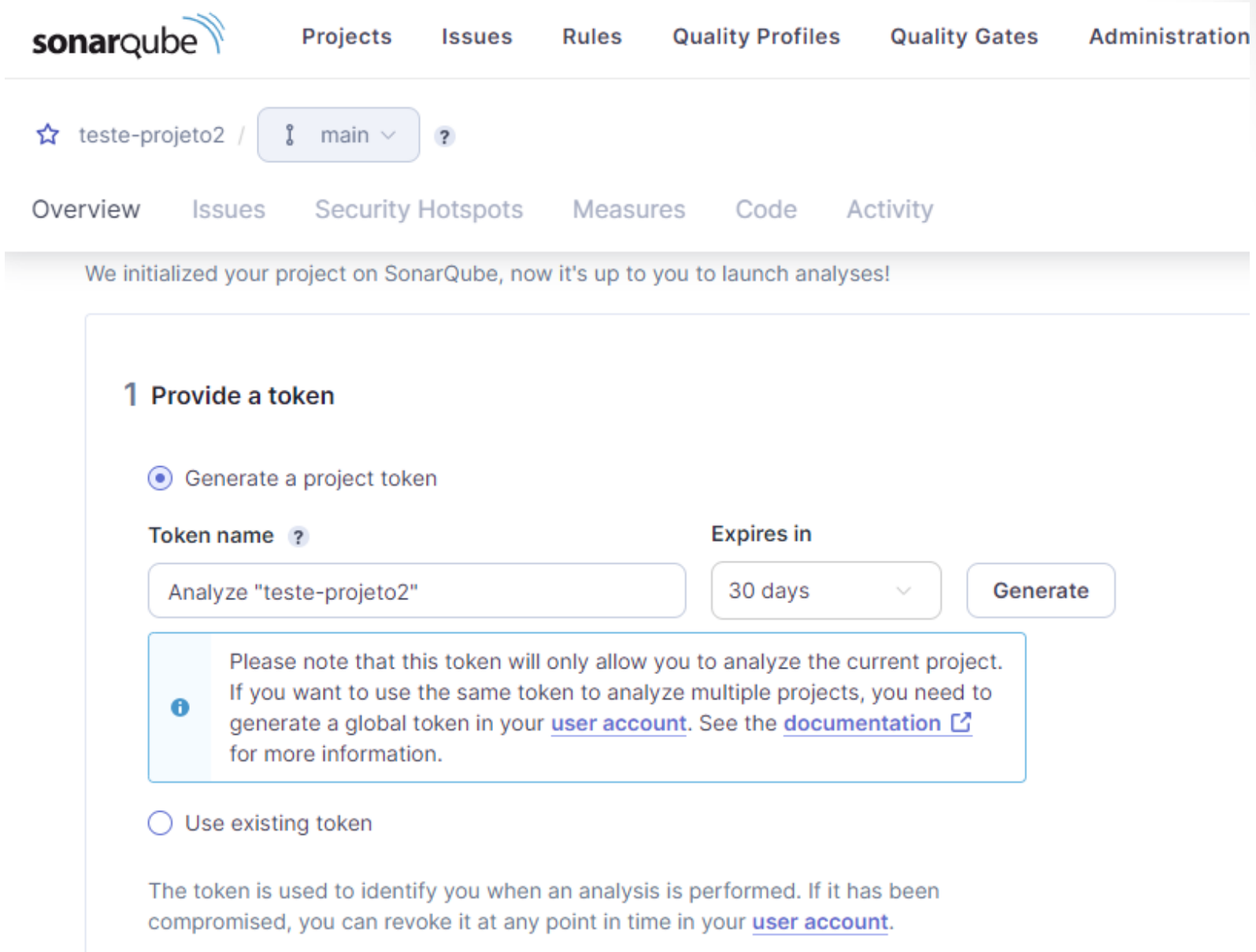
Criação de Projeto

Configuração do Projeto

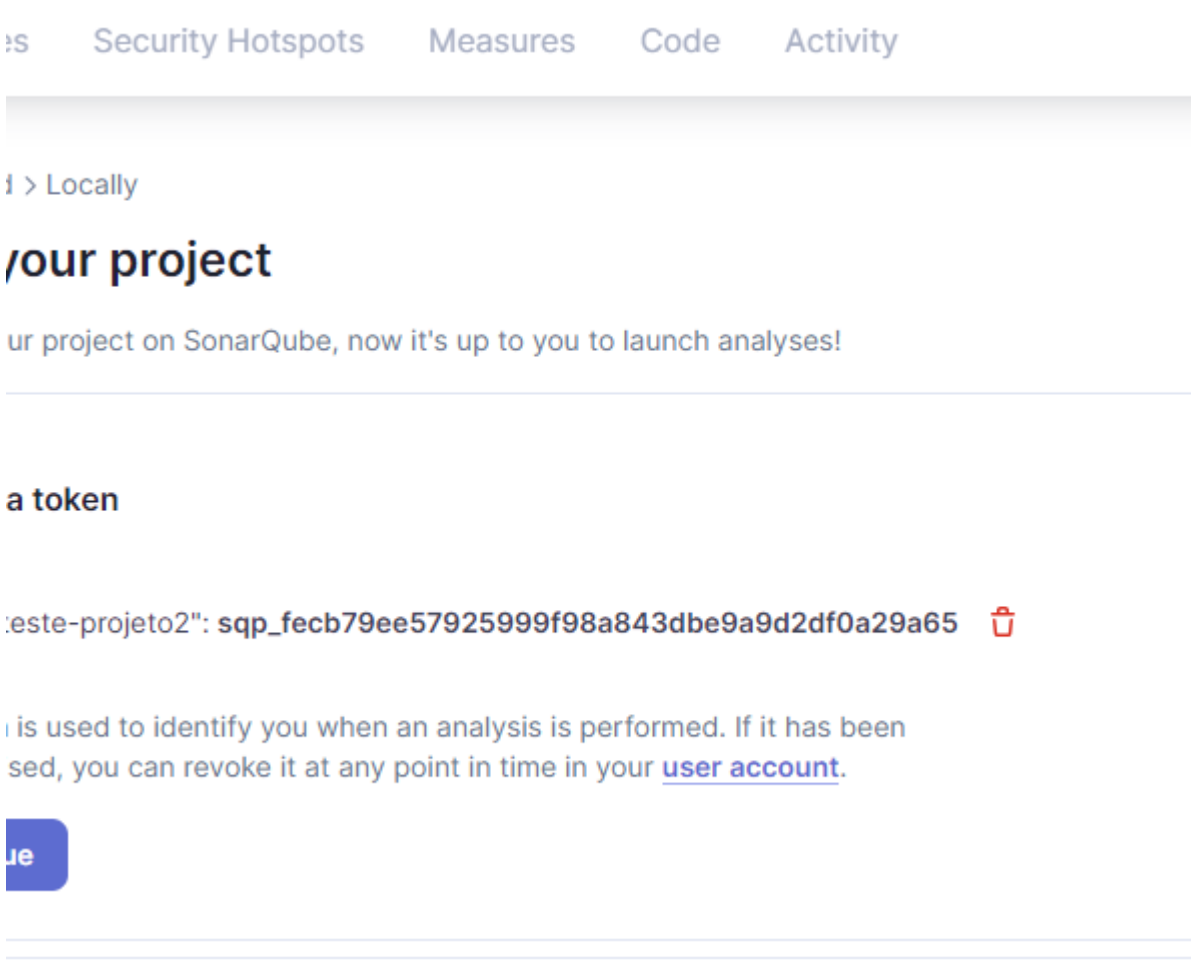
Criação de Token



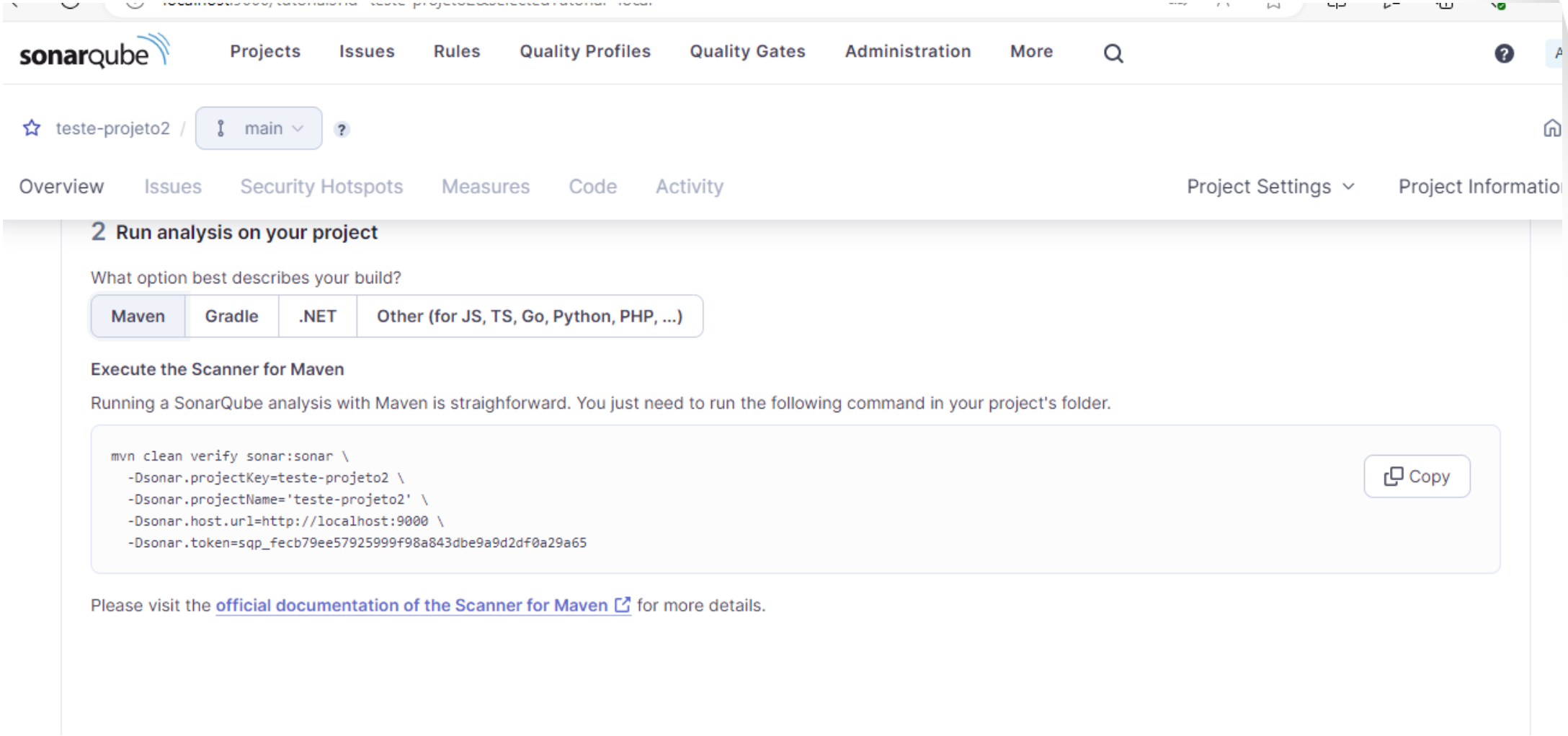
Criação de Chave



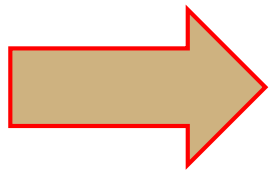
Token Criado

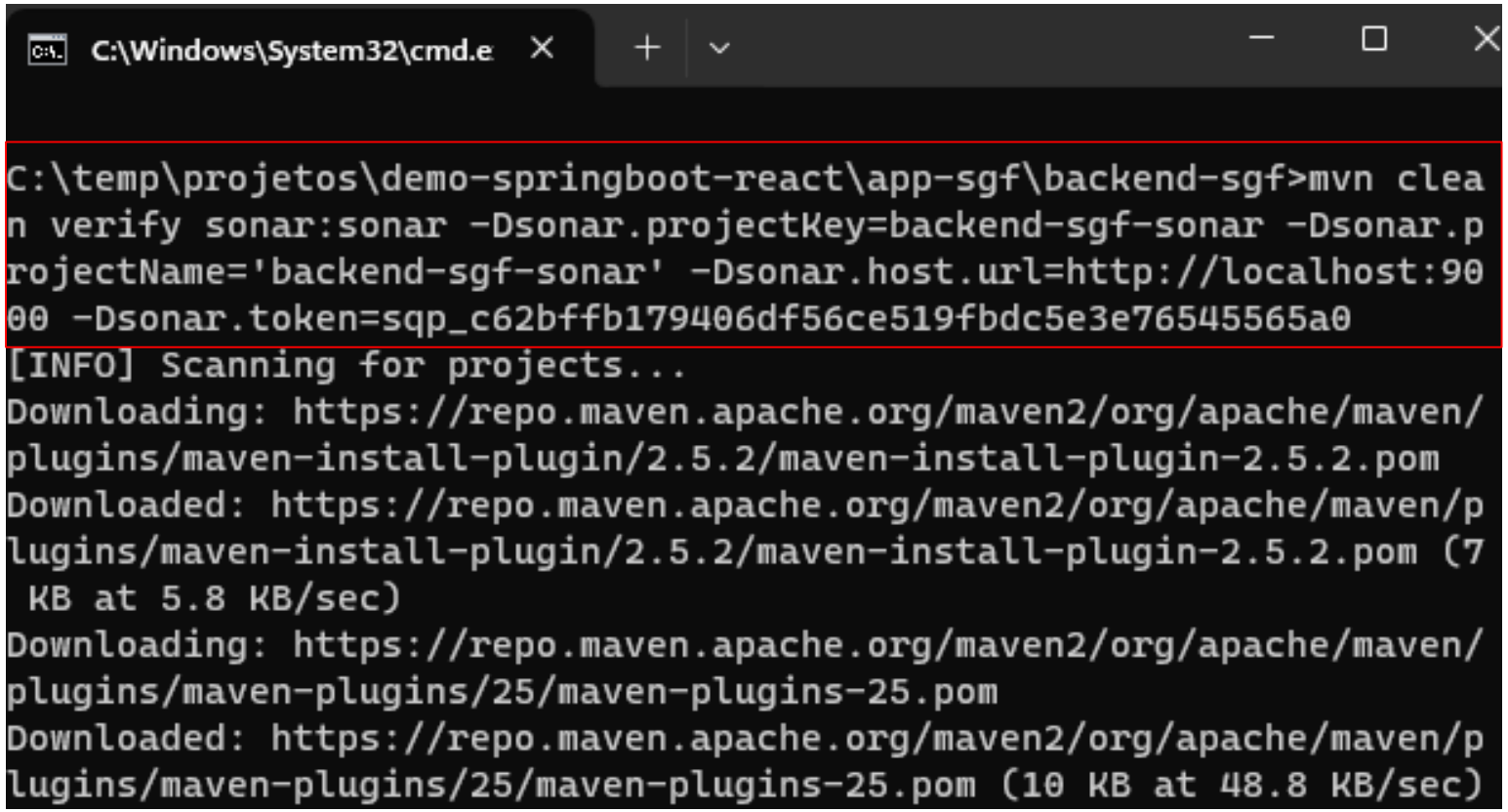


Execução de Comando na Raiz do Projeto

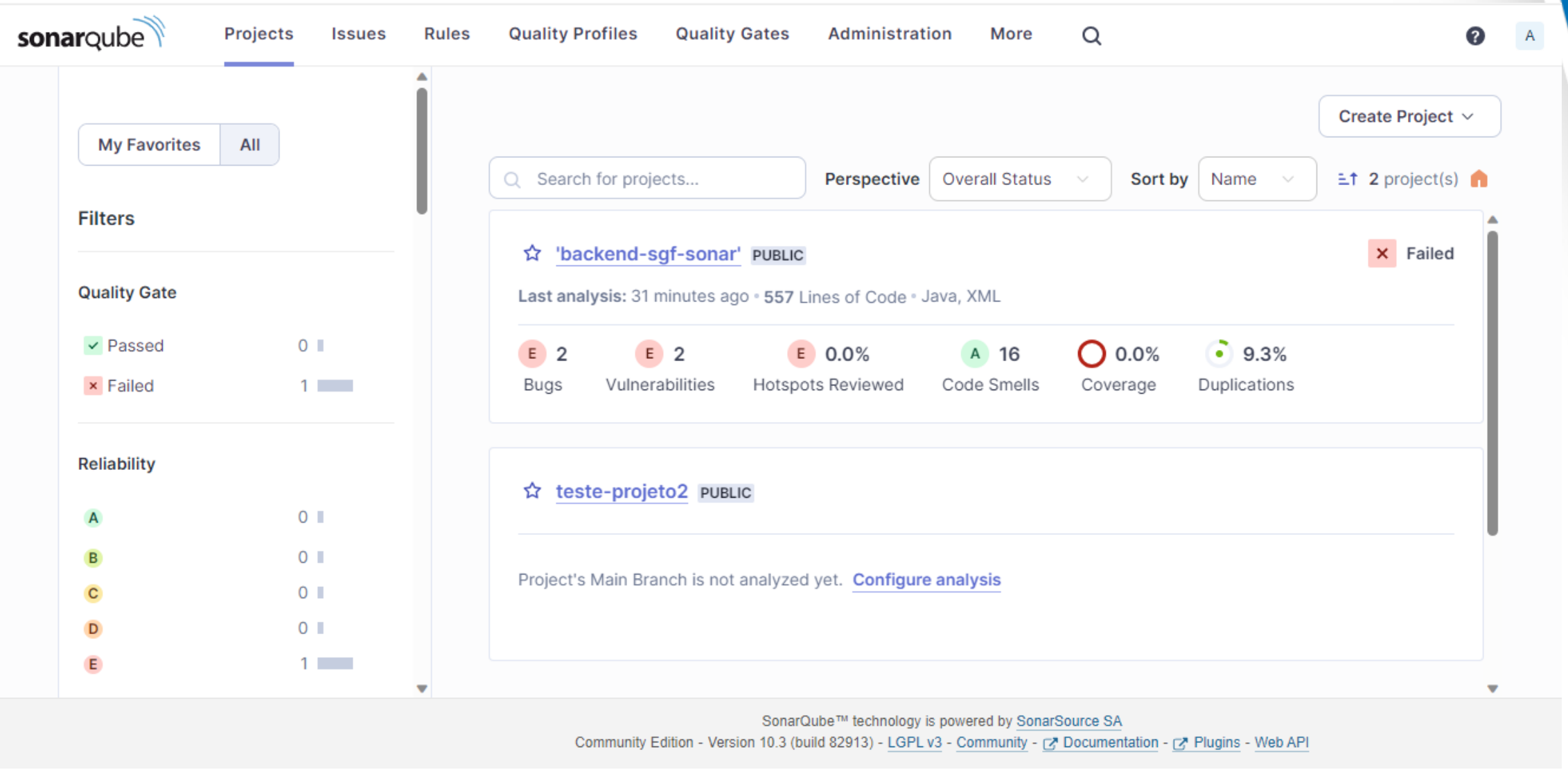


Execução de comando na raiz do projeto

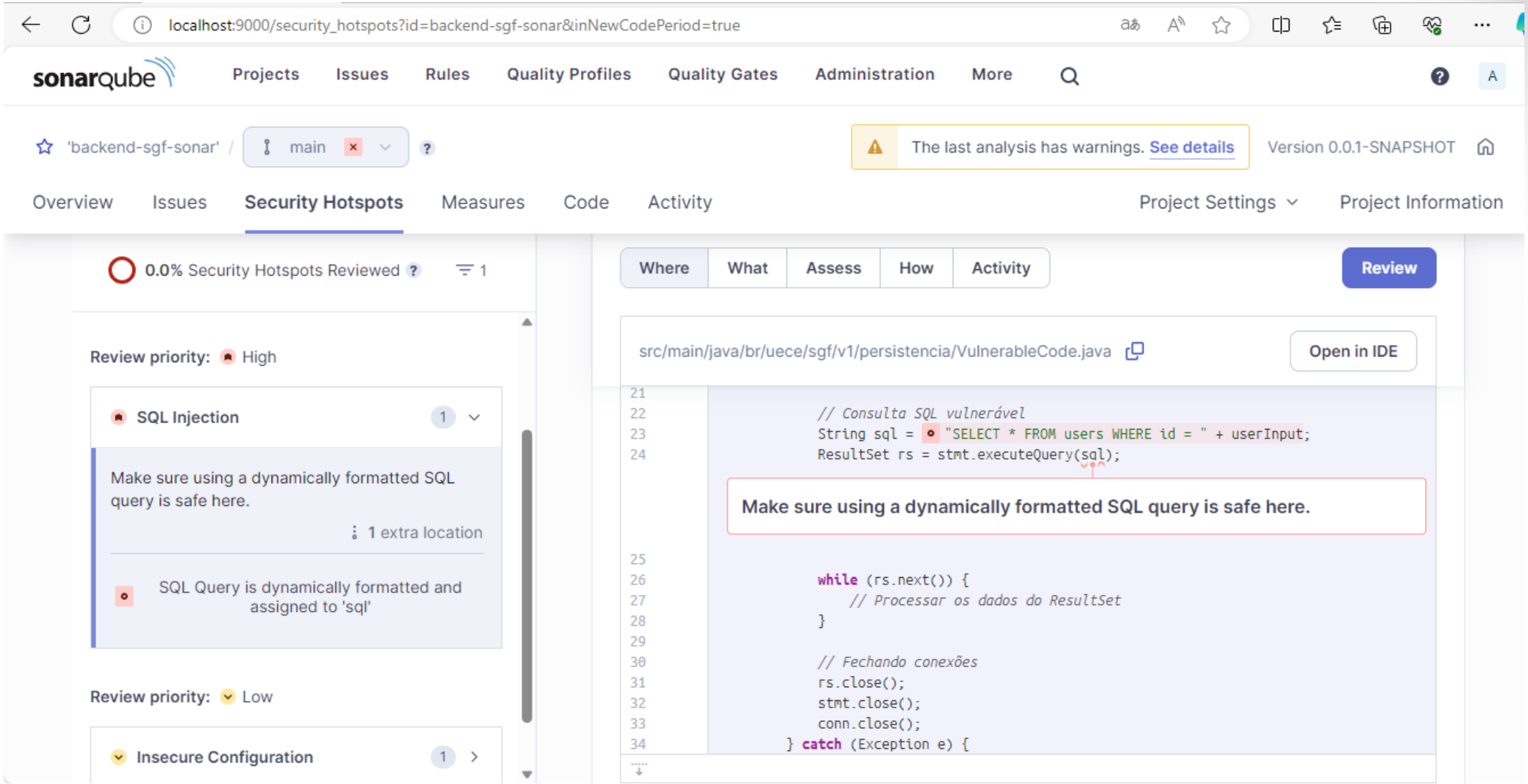




Execução de Comando na Raiz do Projeto



Exemplo de Vulnerabilidade



Práticas de Mitigação

Vulnerabilidade 1: SQL Injection

VulnerableCode.java | DemoApplicationTests.java | RepositorioFuncionario.ja... | RepositorioFuncionarioId...

```java
package br.uece.sgr.vi.persistencia;
import java.sql.Connection;

public class VulnerableCode {
    public static void main(String[] args) {
        // Dados do banco de dados (exemplo)
        String url = "jdbc:mysql://localhost:3306/mydb";
        String user = "username";
        String password = "password";

        try {
            // Conexão com o banco de dados
            Connection conn = DriverManager.getConnection(url, user, password);
            Statement stmt = conn.createStatement();

            // Entrada do usuário
            String userInput = "1 OR 1=1"; // Exemplo de entrada perigosa

            // Consulta SQL vulnerável
            String sql = "SELECT * FROM users WHERE id = " + userInput;
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
                // Processar os dados do ResultSet
            }

            // Fechando conexões
            rs.close();
            stmt.close();
            conn.close();
```

Injeção de SQL é um tipo de ameaça de segurança que se aproveita de falhas em sistemas que trabalham com bases de dados realizando ataques com comandos SQL

Mitigação da Vulnerabilidade 1

*VulnerableCode.java | DemoApplicationTests.java | RepositorioFuncionario.j... | RepositorioFuncionarioJ...

```java
public class SecureCode {
    public static void main(String[] args) {
        // Dados do banco de dados (exemplo)
        String url = "jdbc:mysql://localhost:3306/mydb";
        String user = "username";
        String password = "password";

        try {
            // Conexão com o banco de dados
            Connection conn = DriverManager.getConnection(url, user, password);

            // Entrada do usuário
            String userInput = "1"; // Supondo uma entrada segura

            // Consulta SQL segura usando PreparedStatement
            String sql = "SELECT * FROM users WHERE id = ?";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, userInput); // Configuração segura do parâmetro

            ResultSet rs = pstmt.executeQuery();

            while (rs.next()) {
                // Processar os dados do ResultSet
            }

            // Fechando conexões
            rs.close();
            pstmt.close();
            conn.close();
```

- Uso de prepared statement
- Executa a mesma instrução protegendo contra injeções de SQL

Vulnerabilidade 2: Excesso de Informações no Log

SecureLogging.ja... | *VulnerableCod... | DemoApplication... | RepositorioFunc... | RepositorioFunc...

```java
package br.uece.sgf.sec;
import java.util.logging.Logger;

public class SecureLogging {
    private static final Logger LOGGER = Logger.getLogger(SecureLogging.class.getName());

    public void processUserLogin(String username, String password) {
        // Simula a verificação das credenciais do usuário
        boolean loginSuccess = checkCredentials(username, password);

        if (loginSuccess) {
            LOGGER.info("Usuário " + username + " logado com sucesso.");
            // Processos adicionais após o login bem-sucedido
        } else {
            LOGGER.warning("Tentativa de login falhou para o usuário " + username);
        }
    }

    private boolean checkCredentials(String username, String password) {
        // Simulação de uma verificação de credenciais
        return "admin".equals(username) && "admin123".equals(password);
    }

    public static void main(String[] args) {
        new SecureLogging().processUserLogin("admin", "admin123");
    }
}
```

O exemplo a seguir demonstra um código Java que registra excessivamente informações, incluindo dados sensíveis do usuário, em um log.

Este comportamento pode expor informações confidenciais.

Mitigação da Vulnerabilidade 2

```java
import java.util.logging.Logger;

public class InsecureExceptionHandling {
    private static final Logger LOGGER = Logger.getLogger(
            InsecureExceptionHandling.class.getName()
    );


    public void processData(String data) {
        try {
            // Código que pode lançar uma exceção
            System.out.println("Processando dados: " + data);
            // Mais código...
        } catch (Exception e) {
            // Logando a exceção de maneira segura
            LOGGER.severe("Erro ao processar dados: " + e.toString());

            // Exibindo uma mensagem genérica para o usuário
            System.out.println("Ocorreu um erro interno. Por favor, tente "
                    + "novamente mais tarde.");

        }
    }

    public static void main(String[] args) {
        new InsecureExceptionHandling().processData("dados_importantes");
    }
}
```

A versão corrigida do código evita registrar informações sensíveis, como a senha do usuário.

Em vez disso, apenas informações necessárias e não sensíveis são registradas.

Vulnerabilidade 3: Tratamento Inadequado de Exceções

*SecureLoggi... *VulnerableC... RepositorioF... RepositorioF... InsecureExc...

```java
package br.uece.sgf.sec;
public class InsecureExceptionHandling {

    public void processData(String data) {
        try {
            // Código que pode lançar uma exceção
            System.out.println("Processando dados: " + data);
            // Mais código...
        } catch (Exception e) {
            // Exibindo informações detalhadas da exceção para o usuário
            System.out.println("Ocorreu um erro: " + e.toString());
        }
    }

    public static void main(String[] args) {
        new InsecureExceptionHandling().processData("dados_importantes");
    }
}
```

Este exemplo Java demonstra um tratamento inadequado de exceções, onde informações detalhadas sobre a exceção são exibidas ao usuário.

Isso pode potencialmente expor detalhes internos do sistema ou pontos fracos de segurança.

Mitigação da Vulnerabilidade 3

```java
package br.uece.sgf.sec;
import java.util.logging.Logger;

public class InsecureExceptionHandling {
    private static final Logger LOGGER = Logger.getLogger(
            InsecureExceptionHandling.class.getName()
    );

    public void processData(String data) {
        try {
            // Código que pode lançar uma exceção
            System.out.println("Processando dados: " + data);
            // Mais código...
        } catch (Exception e) {
            // Logando a exceção de maneira segura
            LOGGER.severe("Erro ao processar dados: " + e.toString());

            // Exibindo uma mensagem genérica para o usuário
            System.out.println("Ocorreu um erro interno. Por favor, tente "
                    + "novamente mais tarde.");
        }
    }

    public static void main(String[] args) {
        new InsecureExceptionHandling().processData("dados_importantes");
    }
}
```

A versão corrigida do código evita expor detalhes da exceção ao usuário.

Em vez disso, apenas uma mensagem genérica é mostrada, enquanto os detalhes da exceção são registrados de forma segura para análise interna.

Referências

- https://www.linkedin.com/pulse/qualidade-do-c%C3%B3digo-com-sonarqube-e-docker-tiago-perroni/?originalSubdomain=pt
- https://blog.4linux.com.br/analise-sast-com-sonarqube-devsecops/