Relatório de Ferramentas utilizadas para scans de segurança

Introdução

Este documento detalha a seleção estratégica de ferramentas para integrar práticas de segurança (Sec) ao ciclo de vida de desenvolvimento e operações (DevOps), estabelecendo uma esteira DevSecOps robusta.

O objetivo é automatizar a detecção de vulnerabilidades o mais cedo possível ("Shift-Left Security"), cobrindo diferentes vetores de ataque: segredos expostos, vulnerabilidades no código-fonte, vulnerabilidades em dependências de terceiros (Supply Chain) e configurações inseguras de contêineres.

As ferramentas selecionadas—Gitleaks, SonarQube (SonarCloud), Trivy e OWASP Dependency Tracker—foram escolhidas por sua complementaridade, capacidade de automação em CI/CD e cobertura abrangente das principais áreas de risco.

Mapeamento das Ferramentas no Ciclo DevSecOps

A estratégia visa cobrir as seguintes etapas do pipeline:

1. **Pré-Commit / Commit (Desenvolvimento):** Detecção de segredos (Gitleaks).
2. **Build / Integração Contínua (CI):**
    - Análise Estática de Segurança (SAST) e Qualidade de Código (SonarQube).
    - Análise de Composição de Software (SCA) e Scan de Imagens (Trivy).
3. **Monitoramento / Pós-Deploy:** Gerenciamento contínuo de vulnerabilidades (Dependency Tracker).

Detalhamento das ferramentas utilizadas

Secret Scanning

Ferramenta: Gitleaks + Trivy

Descrição da ferramenta

O Gitleaks é uma ferramenta de linha de comando que varre repositórios Git (incluindo todo o histórico de commits) em busca de segredos hardcoded, como chaves de API, senhas, tokens de acesso e chaves privadas.

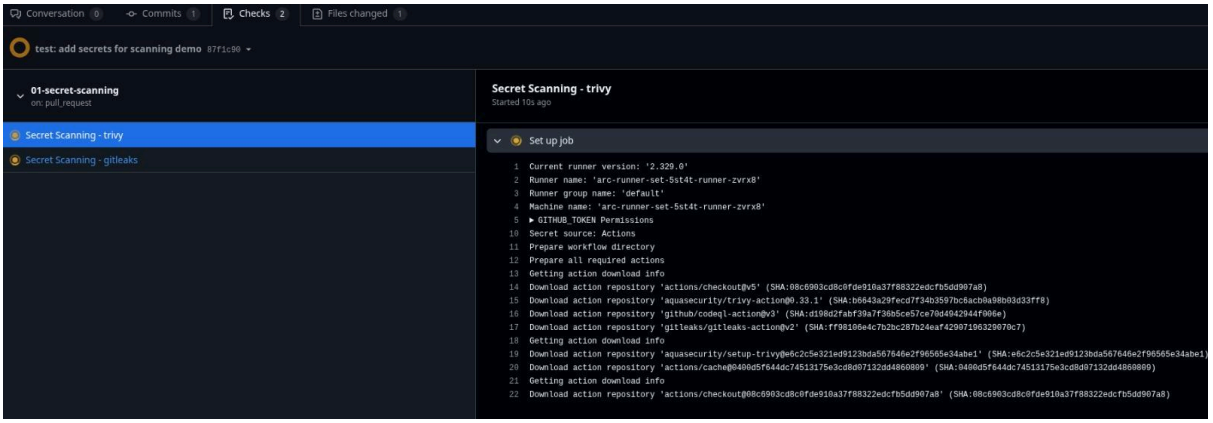
Posicionamento no Pipeline

- **Ideal:** Hooks de Pré-Commit (impedindo o desenvolvedor de comitar o segredo).
- **Obrigatório:** Pipeline de CI (garantindo que nenhum segredo chegue ao repositório central).

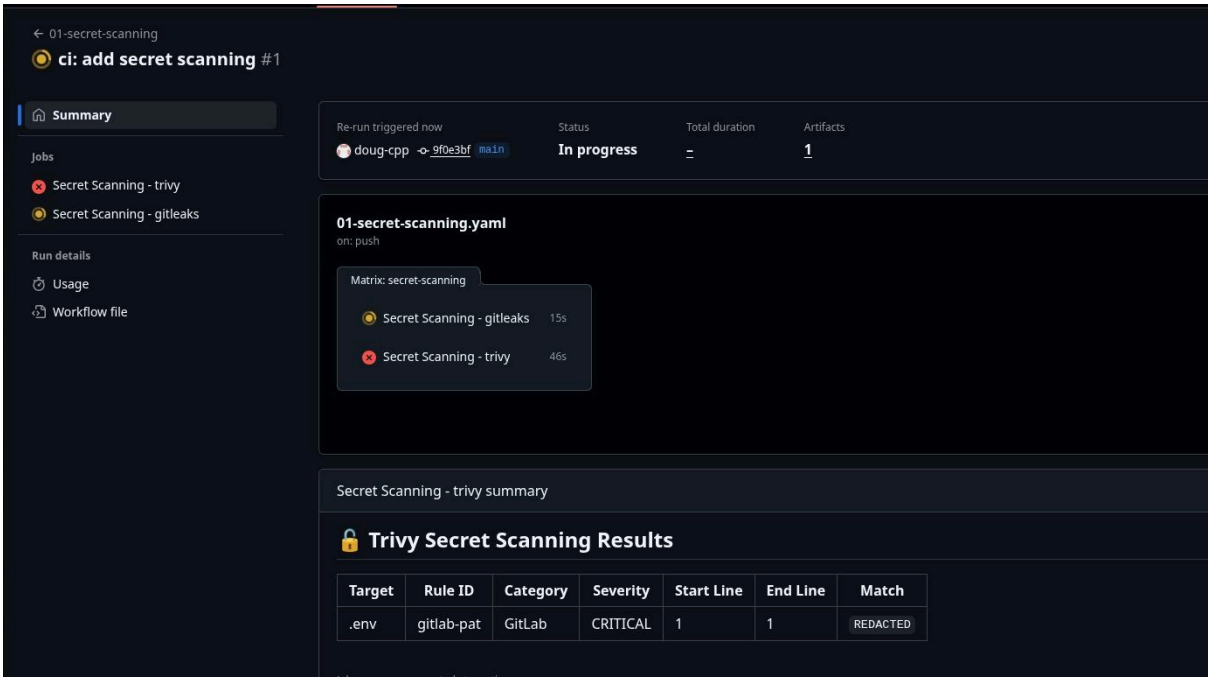
Motivo da Escolha

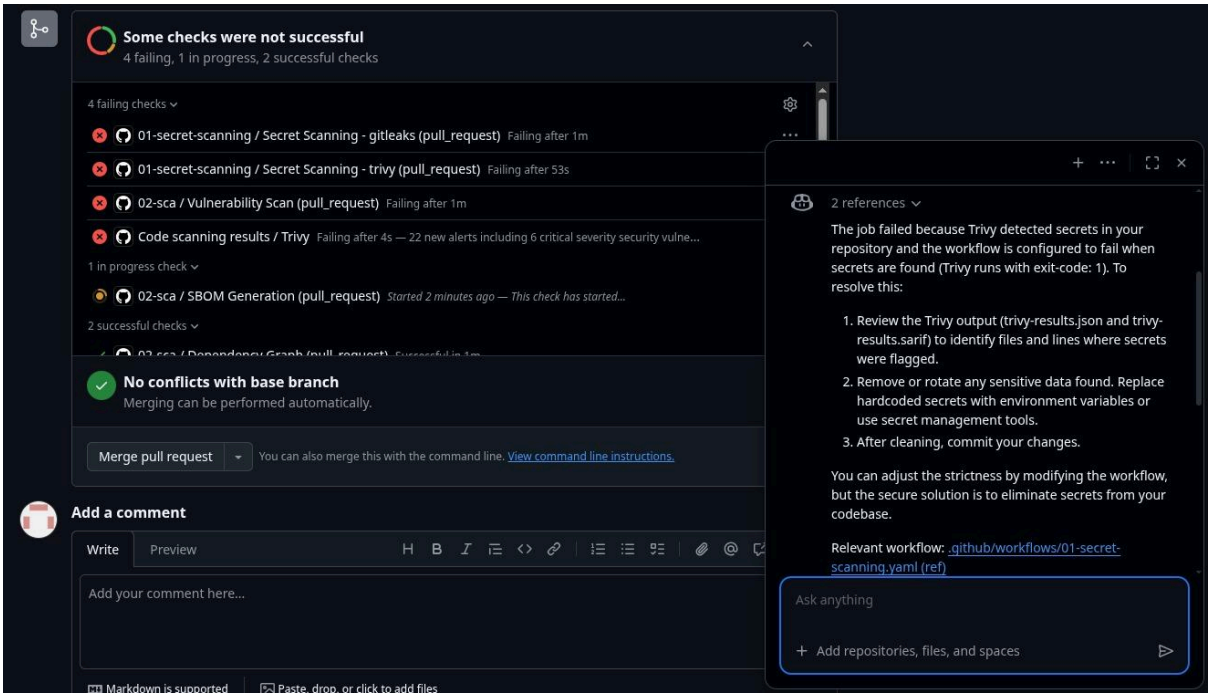
- **Prevenção Primária:** Vazamento de credenciais é uma das falhas de segurança mais comuns e impactantes. O Gitleaks aborda esse risco na origem.
- **Velocidade e Eficiência:** É extremamente rápido e utiliza regras de alta entropia e expressões regulares para identificar segredos com precisão.
- **Abrangência Histórica:** Diferente de outras ferramentas que olham apenas o código atual, o Gitleaks varre o histórico. Um segredo comitado há 6 meses, mesmo que removido hoje, ainda está no histórico do Git e é um risco.
- **Facilidade de Integração:** Sendo um binário simples, é trivial adicionar em scripts de pre-commit e em qualquer sistema de CI (GitHub Actions, GitLab CI, Jenkins).

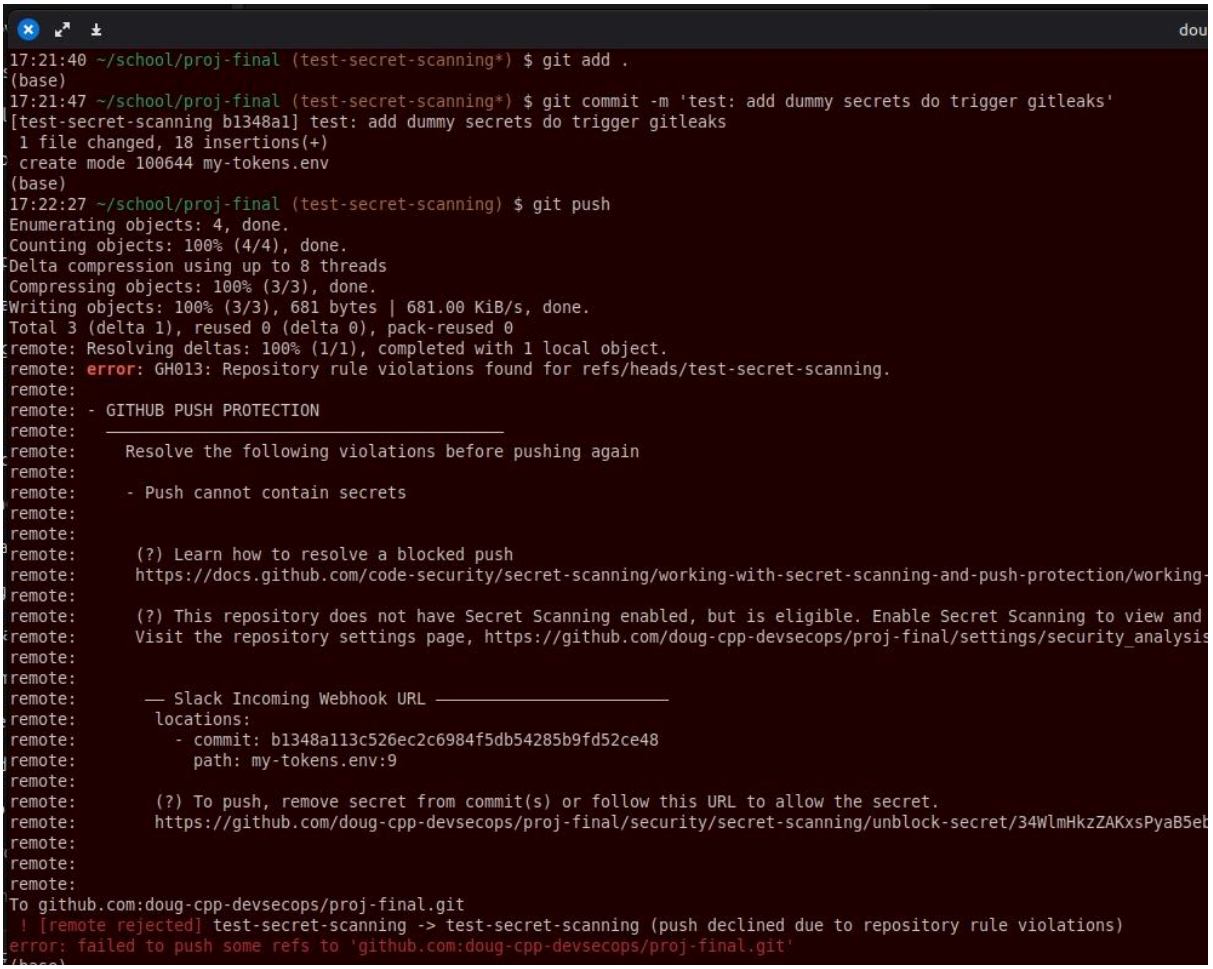
Evidência de execução



Vulnerabilidades encontradas






17:21:40 ~/school/proj-final (test-secret-scanning*) $ git add .
(base)
17:21:47 ~/school/proj-final (test-secret-scanning*) $ git commit -m 'test: add dummy secrets do trigger gitleaks'
[test-secret-scanning b1348a1] test: add dummy secrets do trigger gitleaks
1 file changed, 18 insertions(+)
create mode 100644 my-tokens.env
(base)
17:22:27 ~/school/proj-final (test-secret-scanning) $ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 681 bytes | 681.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote: error: GH013: Repository rule violations found for refs/heads/test-secret-scanning.
remote:
remote: - GITHUB PUSH PROTECTION
remote: ——————————————————————————
remote: Resolve the following violations before pushing again
remote:
remote: - Push cannot contain secrets
remote:
remote:
remote: (?) Learn how to resolve a blocked push
remote: https://docs.github.com/code-security/secret-scanning/working-with-secret-scanning-and-push-protection/working-
remote:
remote: (?) This repository does not have Secret Scanning enabled, but is eligible. Enable Secret Scanning to view and
remote: Visit the repository settings page, https://github.com/doug-cpp-devsecops/proj-final/settings/security_analysis
remote:
remote:
remote: —— Slack Incoming Webhook URL ——————————————————
remote: locations:
remote: - commit: b1348a113c526ec2c6984f5db54285b9fd52ce48
remote: path: my-tokens.env:9
remote:
remote: (?) To push, remove secret from commit(s) or follow this URL to allow the secret.
remote: https://github.com/doug-cpp-devsecops/proj-final/security/secret-scanning/unblock-secret/34WlmHkzZAKxsPyaB5eb
remote:
remote:
remote:
To github.com:doug-cpp-devsecops/proj-final.git
! [remote rejected] test-secret-scanning -> test-secret-scanning (push declined due to repository rule violations)
error: failed to push some refs to 'github.com:doug-cpp-devsecops/proj-final.git'

Push blocked because a secret was detected

Secret scanning found a **Slack Incoming Webhook URL** secret in your attempted push.

Allowing this secret risks exposure. Instead, consider removing the secret from your commit and commit history.

Exposing this secret can allow someone to:

- Verify the identity of this **Slack Incoming Webhook URL** secret
- Know which resources this secret can access
- Act on behalf of the secret's owner

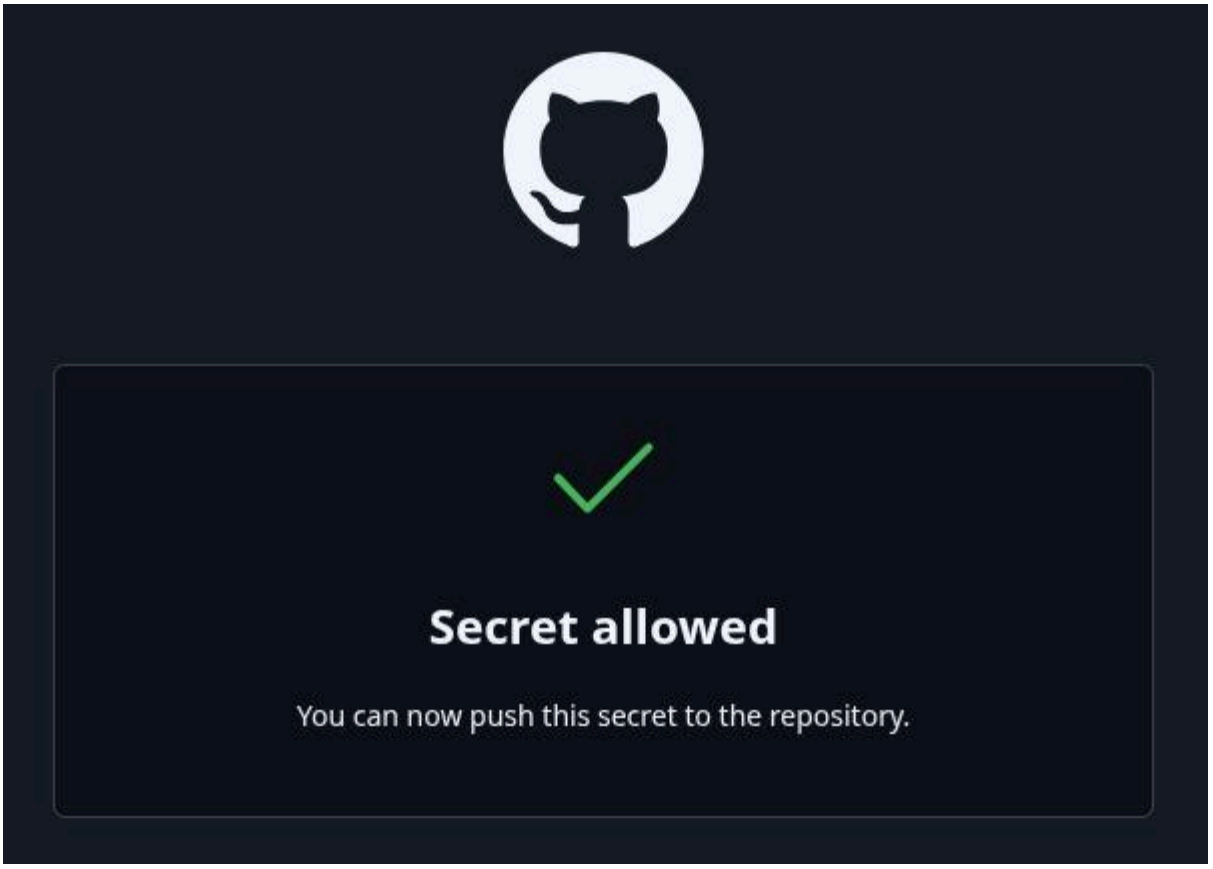
Push protection is enabled for your account. To disable, visit your code security settings.

Allow me to expose this secret

Correção da vulnerabilidade

```
(base)
17:01:14 ~/school/proj-final (main) $ git checkout -b test-secret-scanning
Switched to a new branch 'test-secret-scanning'
(base)
17:03:17 ~/school/proj-final (test-secret-scanning) $ cat > test-secrets.env << 'EOF'
AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
GITHUB_TOKEN="ghp_1234567890abcdefghijklmnopqrstuvwxyz"
DATABASE_PASSWORD="super-secret-password-123"
EOF
(base)
17:04:11 ~/school/proj-final (test-secret-scanning*) $ git status
On branch test-secret-scanning
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test-secrets.env

nothing added to commit but untracked files present (use "git add" to track)
(base)
17:04:15 ~/school/proj-final (test-secret-scanning*) $ git add .
(base)
17:04:27 ~/school/proj-final (test-secret-scanning*) $ git commit -m 'test: add secrets for scanning demo'
[test-secret-scanning 87f1c90] test: add secrets for scanning demo
 1 file changed, 3 insertions(+)
 create mode 100644 test-secrets.env
(base)
17:04:52 ~/school/proj-final (test-secret-scanning) $ git push
```



SCA (Software Composition Analysis)

Ferramenta: Dependency Track

Descrição da ferramenta

O Dependency Tracker (DT) é uma plataforma de análise de composição. Aonde seu foco é *gerenciar* e *monitorar* componentes de software ao longo do tempo.

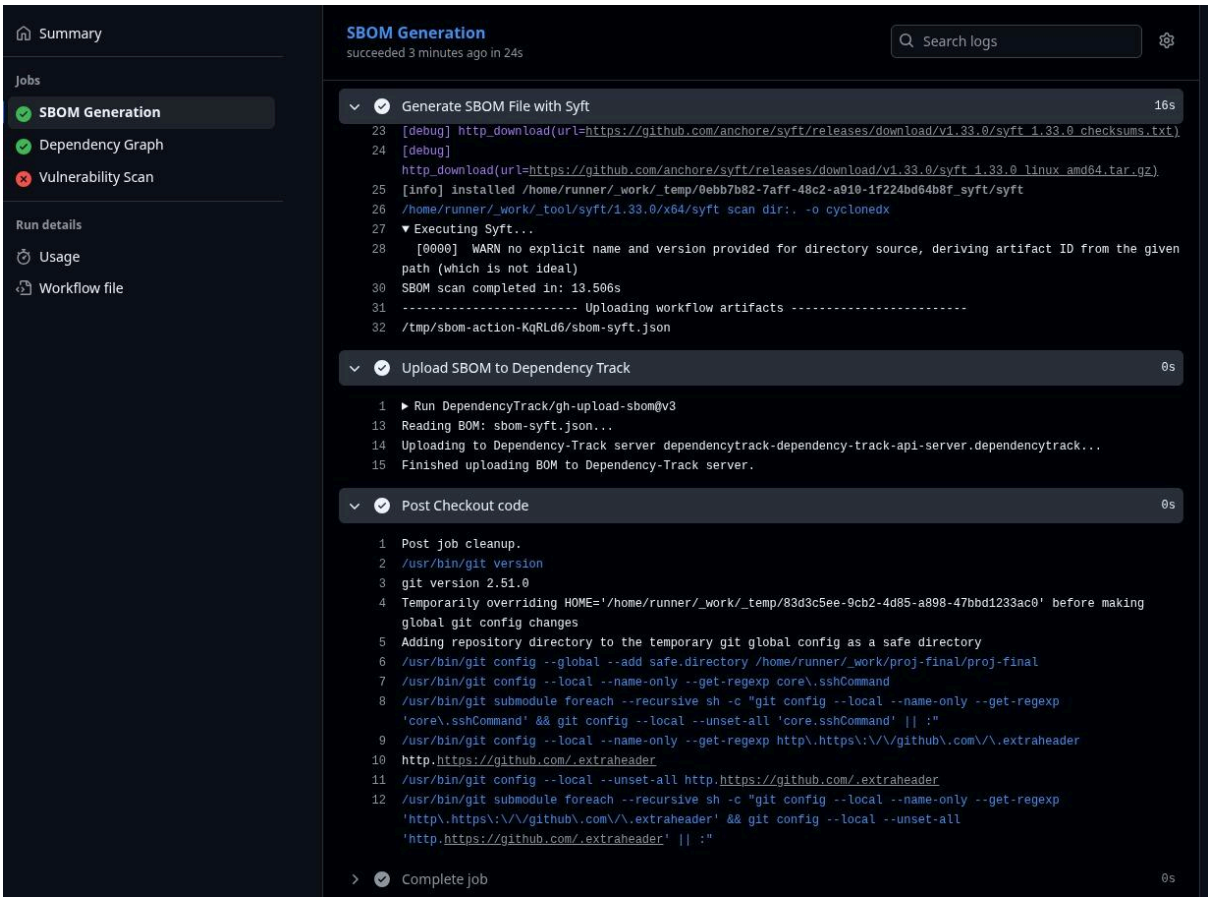
Posicionamento no Pipeline

- Etapa de Pós-Build (CI) ou Pós-Deploy (CD). O pipeline deve gerar um SBOM e enviá-lo para o Dependency Tracker.

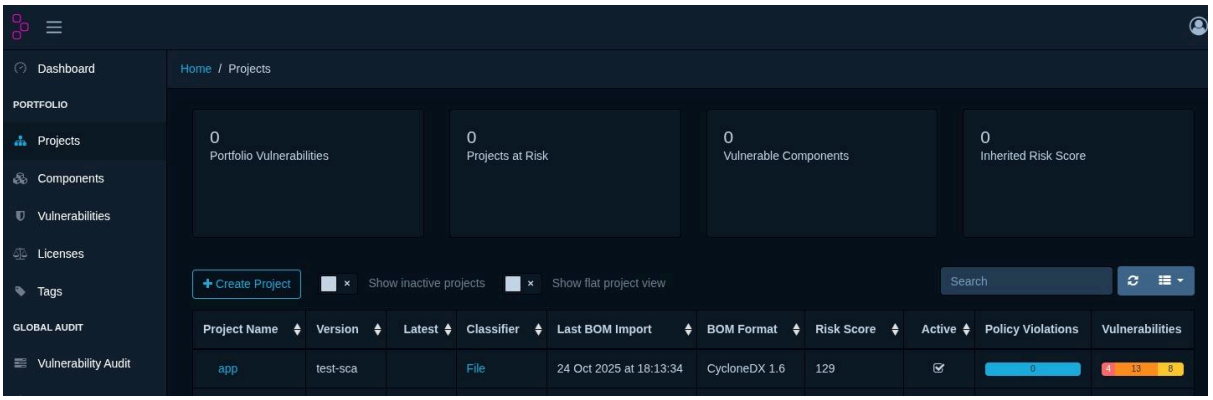
Motivo da Escolha

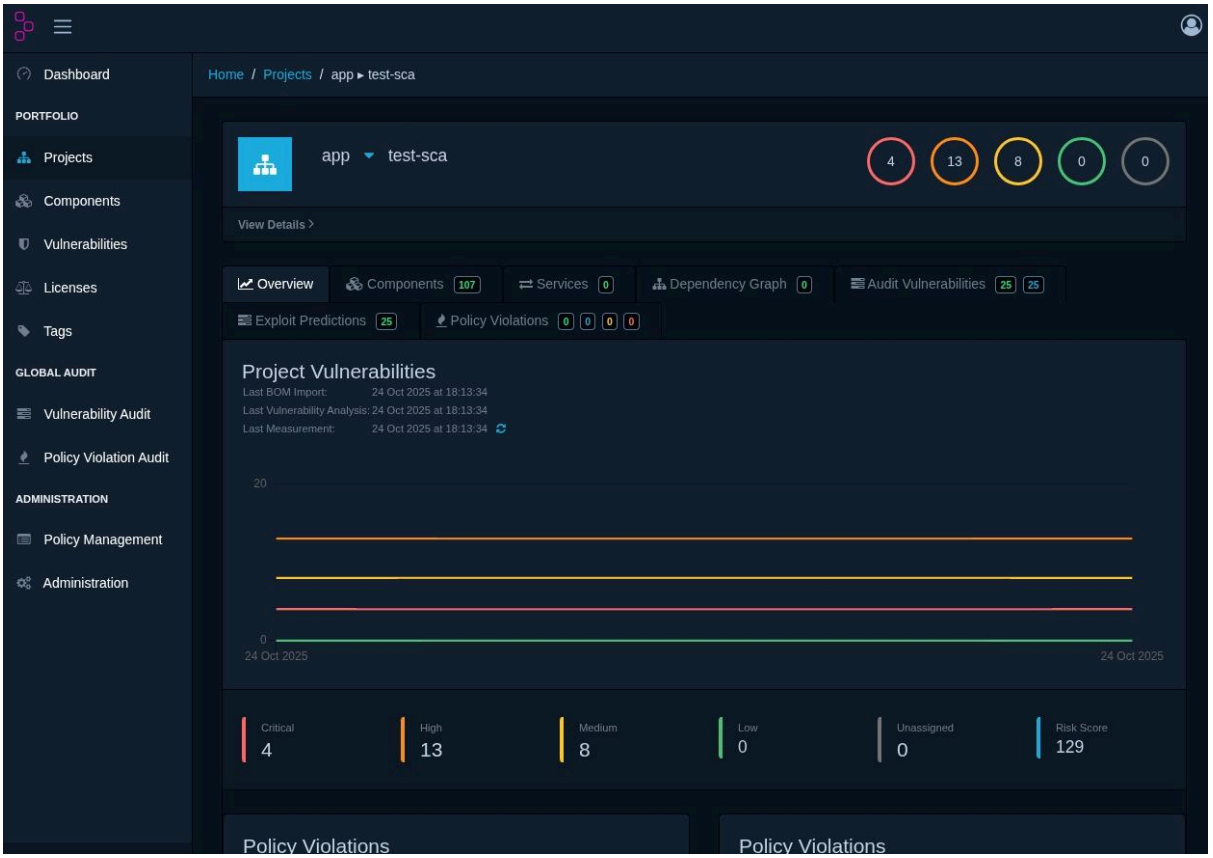
- **Visão Centralizada e Contínua:** O Trivy e o SonarQube nos dizem se o *build atual* é seguro. O Dependency Tracker nos diz se os *projetos em produção* estão seguros. Ele centraliza o inventário de componentes de *todos* os projetos da organização.
- **Monitoramento de "Novas" CVEs:** Esta é sua maior força. Uma biblioteca pode estar segura hoje, mas uma nova CVE pode ser descoberta para ela amanhã. O Dependency Tracker monitora ativamente os componentes e alerta proativamente quando um componente *já em produção* se torna vulnerável.
- **Análise de Risco de Licença:** Além de segurança, ele analisa as licenças de software (ex: GPL, MIT, Apache) das dependências, identificando riscos legais ou de conformidade.
- **Foco em SBOM (Supply Chain Security):** A plataforma é construída em torno do conceito de SBOM (como CycloneDX ou SPDX), que é o padrão emergente da indústria para transparência e segurança da cadeia de suprimentos de software.

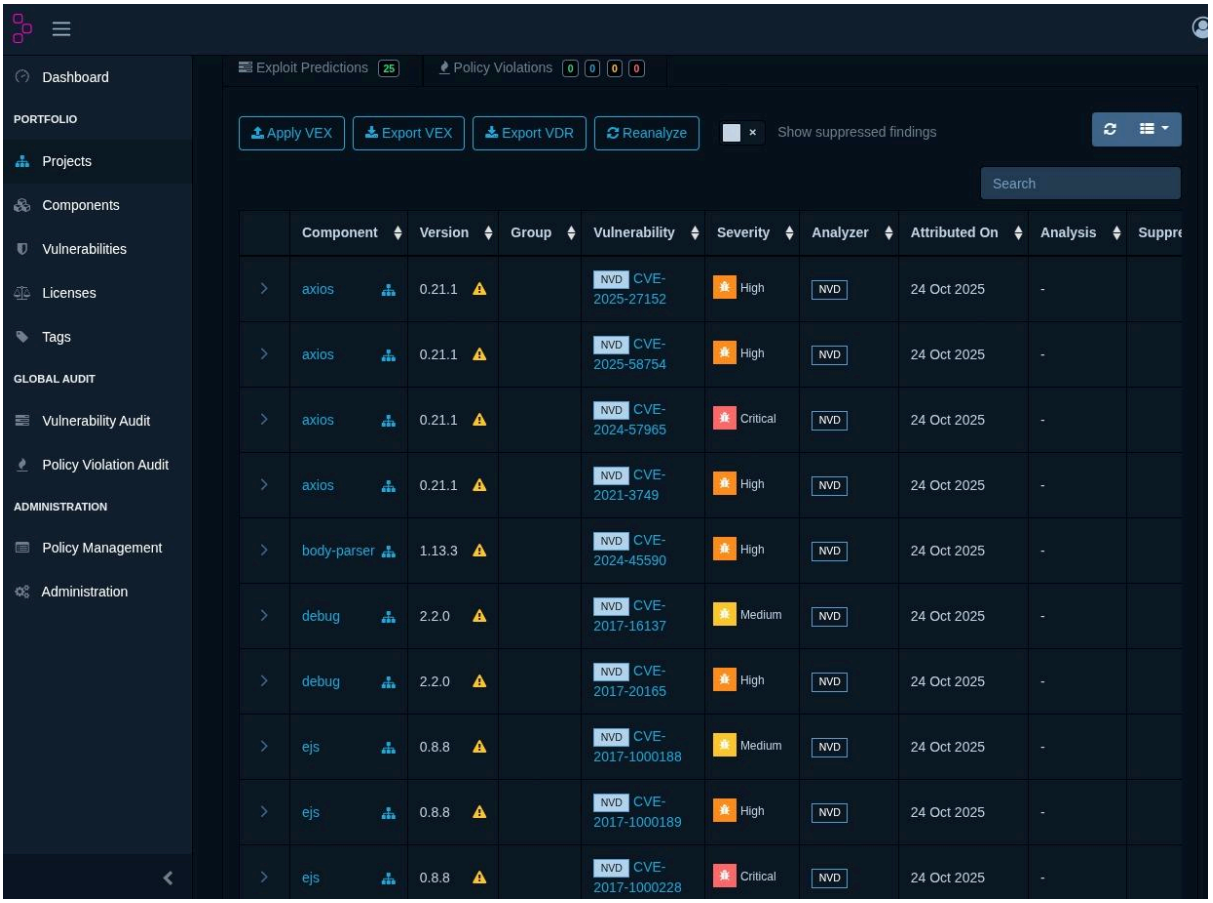
Evidência de execução



Vulnerabilidades encontradas







SAST (Static Analysis)

Ferramenta: SonarQube (SonarCloud)

O SonarQube (auto-hospedado) ou SonarCloud (SaaS) realiza uma análise estática profunda do código-fonte. Ele identifica não apenas "Code Smells" (má qualidade) e "Bugs", mas também "Security Hotspots" e "Vulnerabilities" (falhas de segurança).

Posicionamento no Pipeline

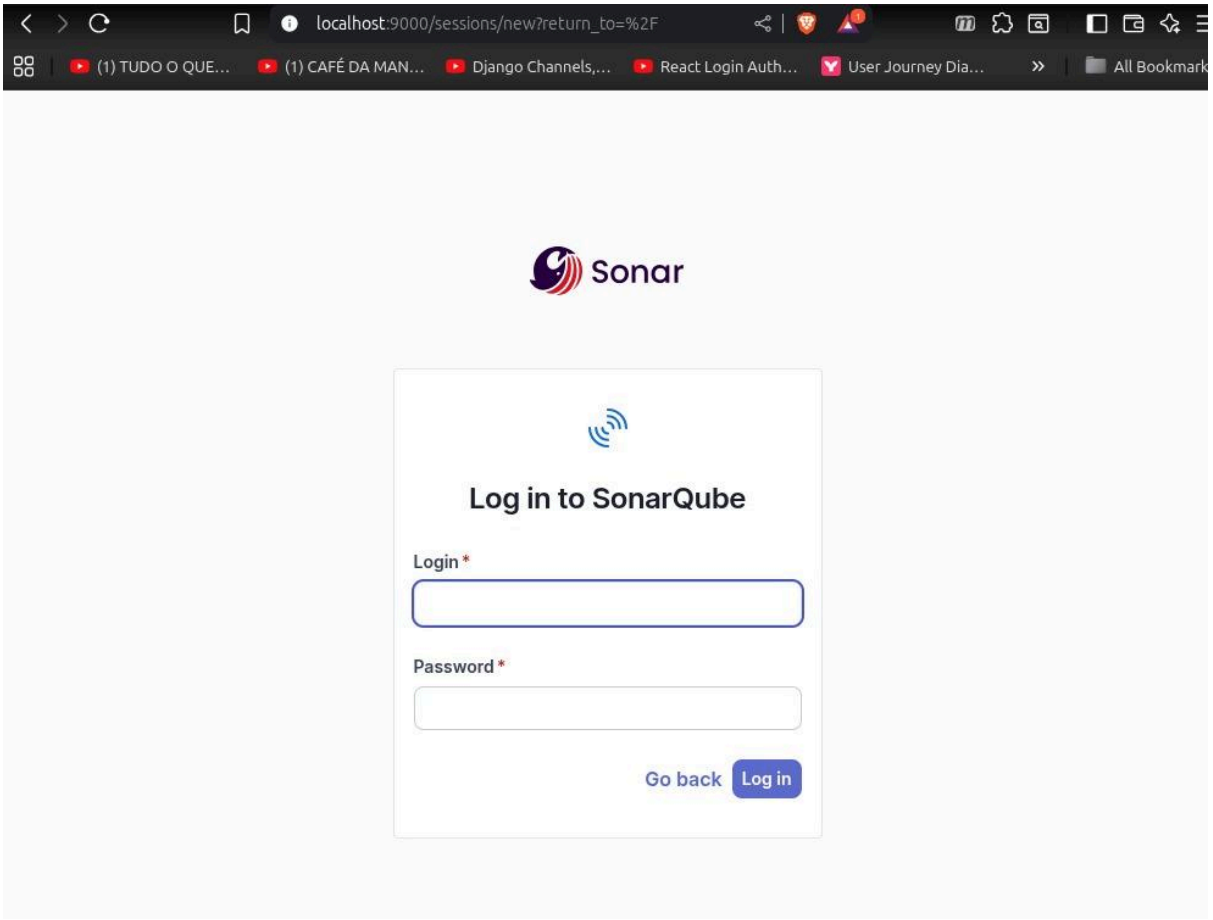
- Pipeline de CI, integrado ao Pull Request (PR) / Merge Request (MR).

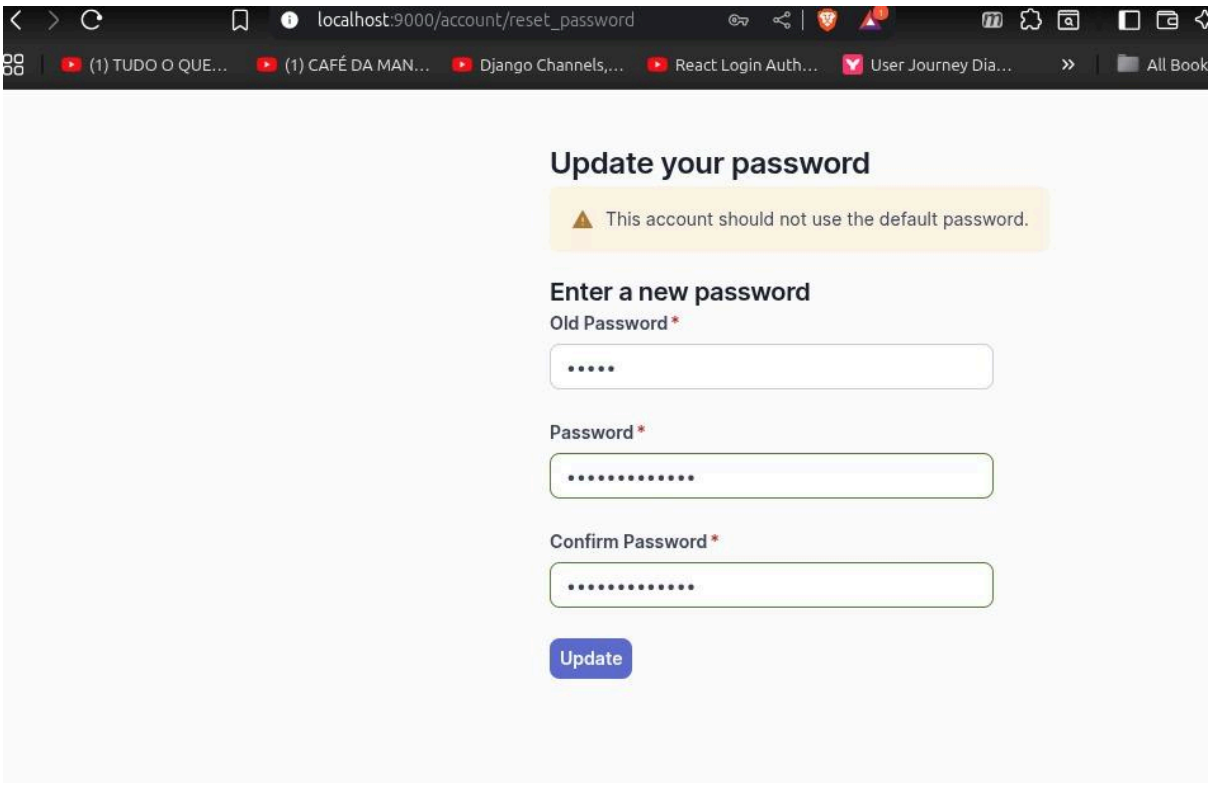
Motivo da Escolha

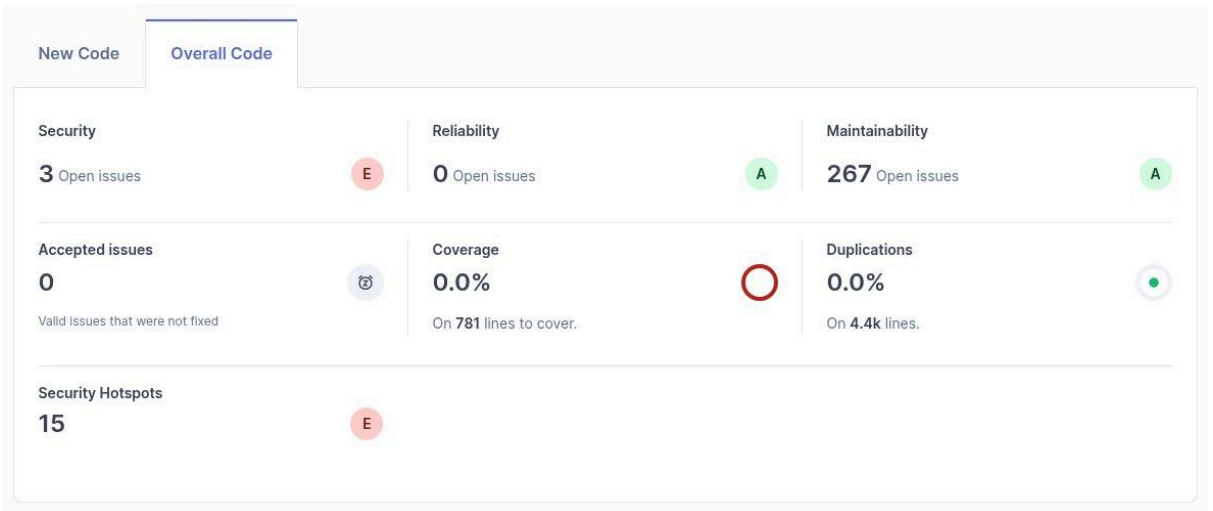
- **Análise de Lógica de Negócio:** Enquanto outras ferramentas focam em dependências, o SonarQube foca no *código que a equipe escreve*. Ele é essencial para encontrar falhas como SQL Injection, Cross-Site Scripting (XSS), e lógica de controle de acesso insegura.

- **Quality Gates:** Permite a definição de "Portões de Qualidade" (Quality Gates) que podem *falhar o build* se novos problemas de segurança ou qualidade forem introduzidos, garantindo um padrão mínimo antes da mesclagem de código.
- **Gerenciamento de Dívida Técnica:** Fornece um dashboard claro sobre a saúde do projeto, ajudando a priorizar a correção de vulnerabilidades e a gerenciar a dívida técnica de segurança ao longo do tempo.
- **Ampla Cobertura de Linguagens:** Suporta uma vasta gama de linguagens de programação, tornando-se um padrão de mercado.

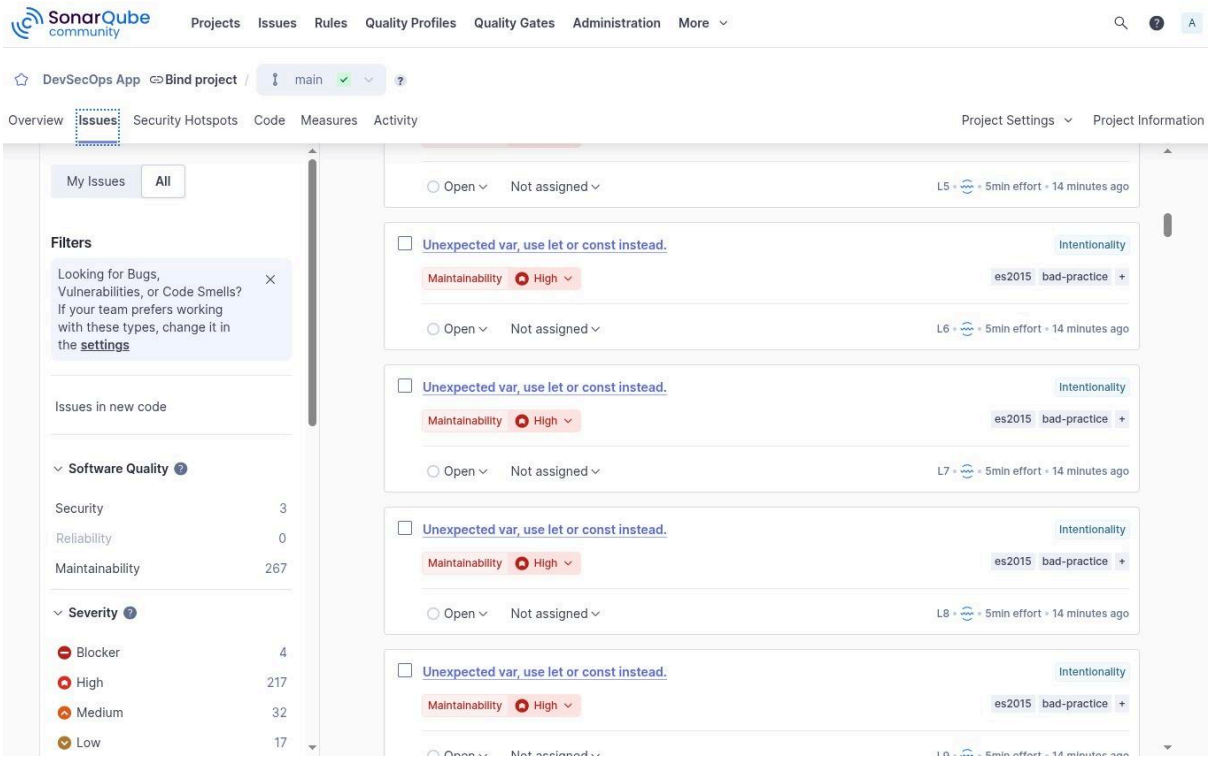
Evidência de execução

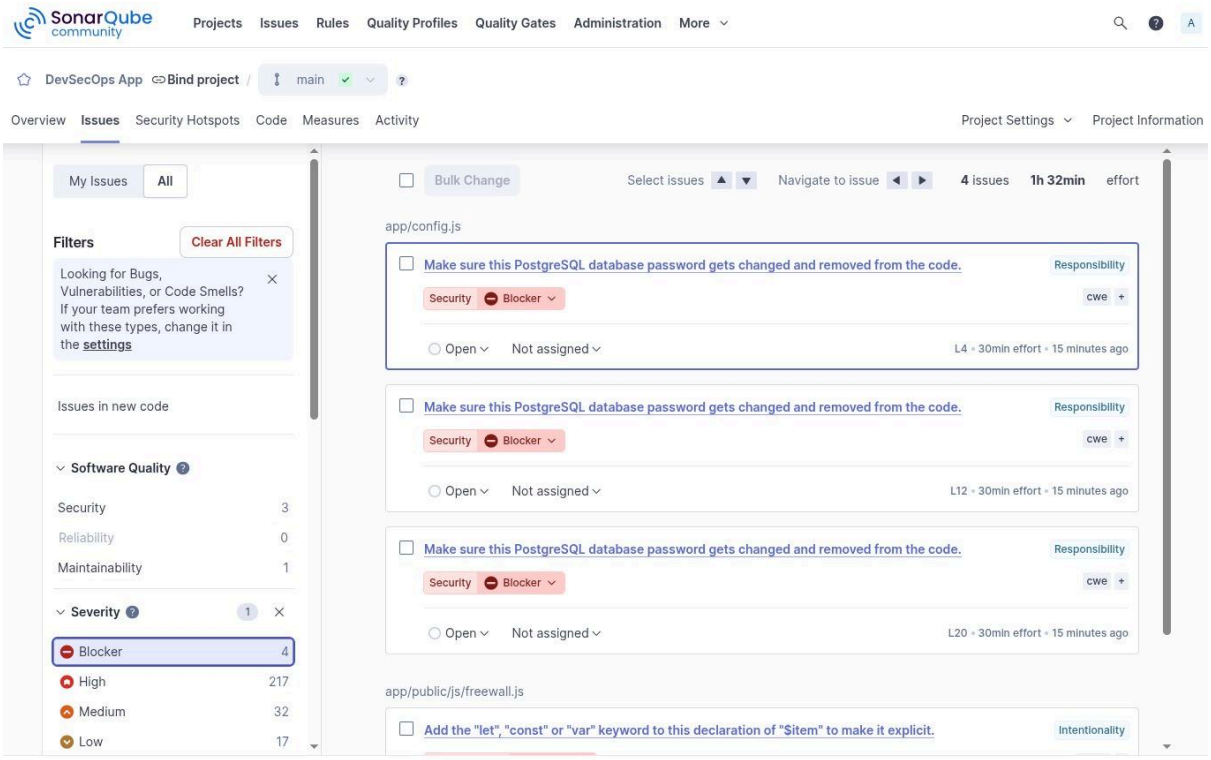


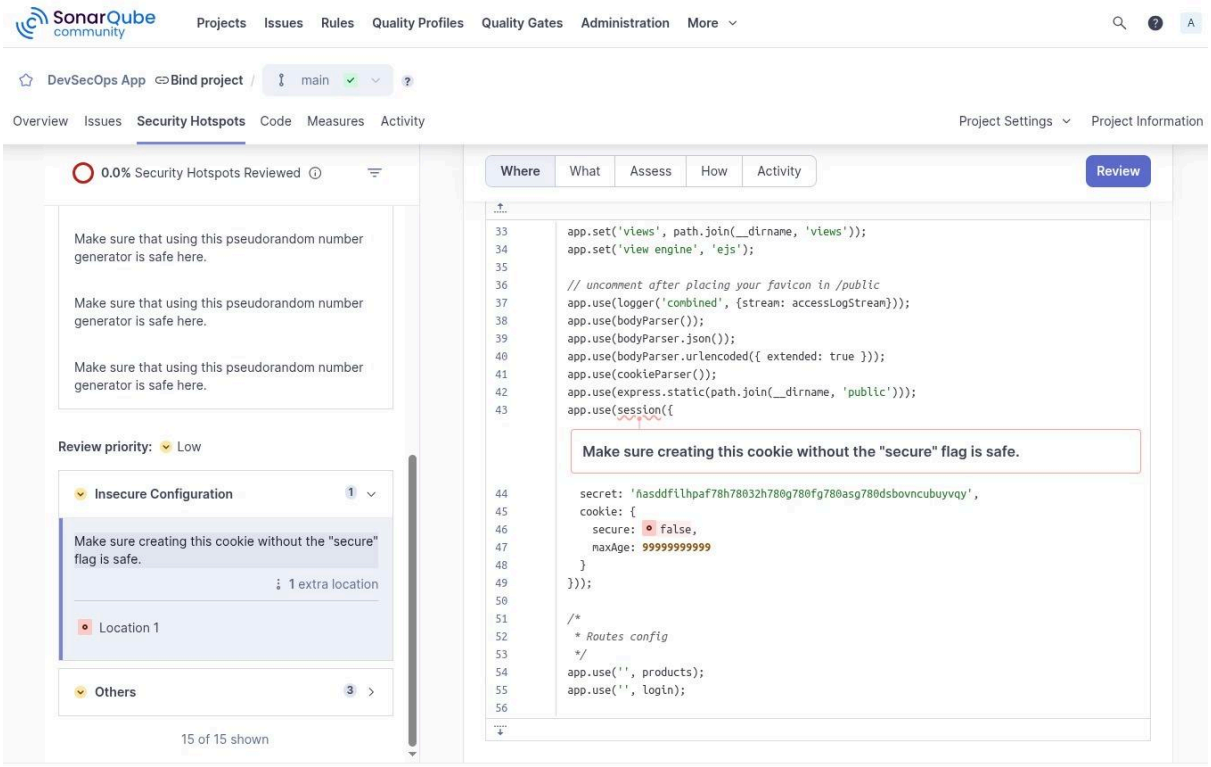


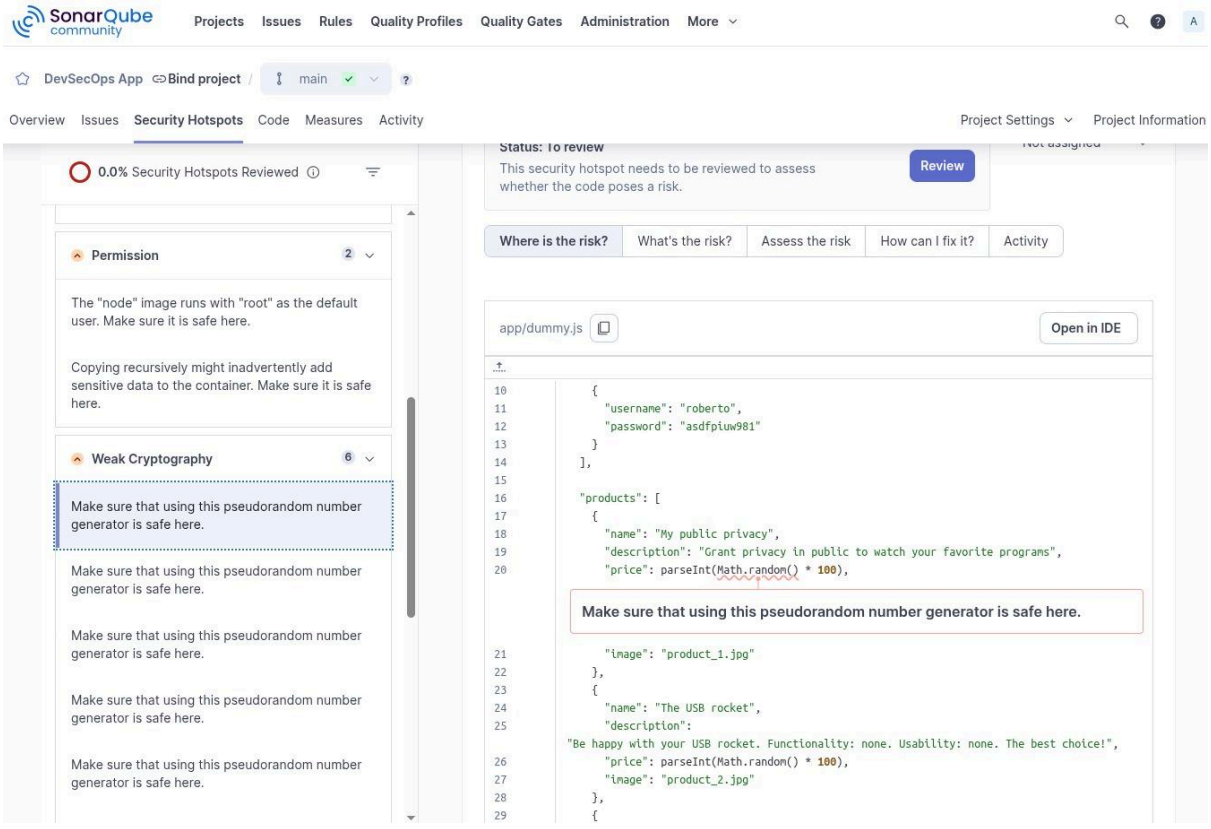


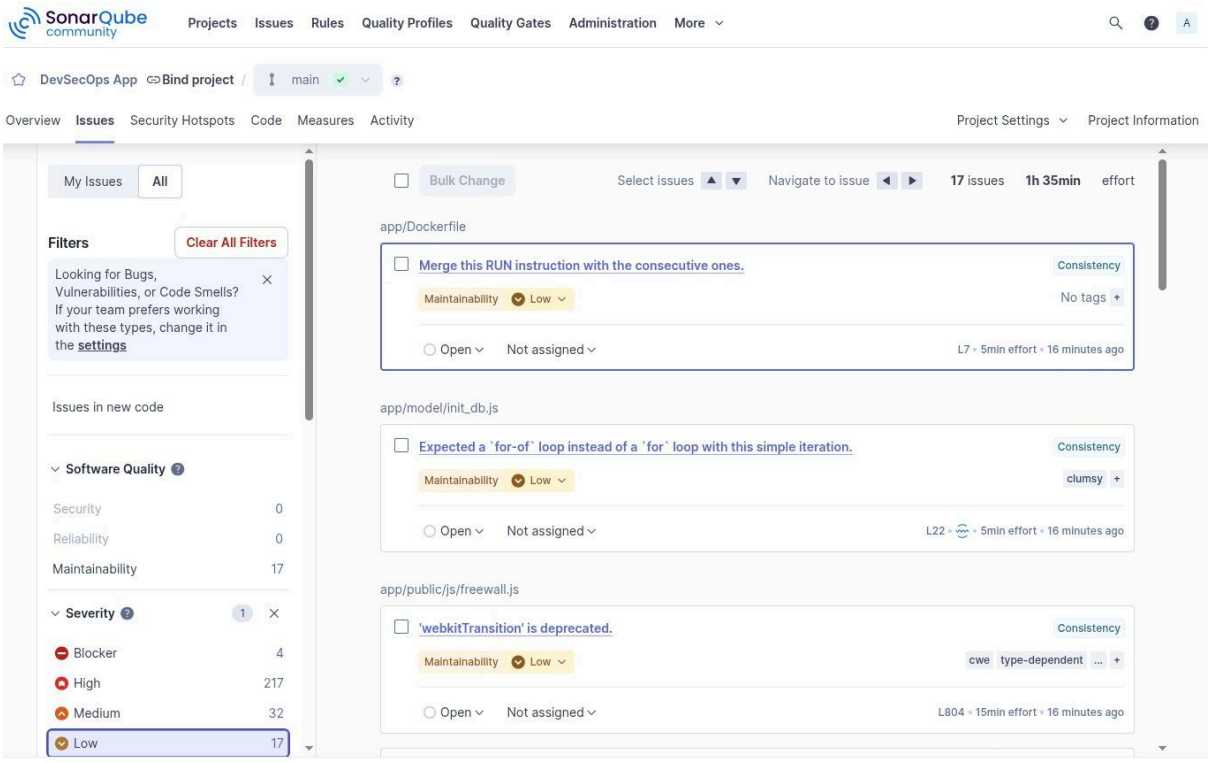
Vulnerabilidades encontradas


SonarQube community
Projects Issues Rules Quality Profiles Quality Gates Administration More
DevSecOps App Bind project / main
Overview Issues Security Hotspots Code Measures Activity
Project Settings Project Information
My Issues All
Filters
Looking for Bugs, Vulnerabilities, or Code Smells? If your team prefers working with these types, change it in the settings
Issues in new code
Software Quality
Security 3
Reliability 0
Maintainability 267
Severity
Blocker 4
High 217
Medium 32
Low 17
Open Not assigned L5 • 5min effort • 14 minutes ago
Unexpected var, use let or const instead.
Intentionality
Maintainability High
es2015 bad-practice +
Open Not assigned L6 • 5min effort • 14 minutes ago
Unexpected var, use let or const instead.
Intentionality
Maintainability High
es2015 bad-practice +
Open Not assigned L7 • 5min effort • 14 minutes ago
Unexpected var, use let or const instead.
Intentionality
Maintainability High
es2015 bad-practice +
Open Not assigned L8 • 5min effort • 14 minutes ago
Unexpected var, use let or const instead.
Intentionality
Maintainability High
es2015 bad-practice +

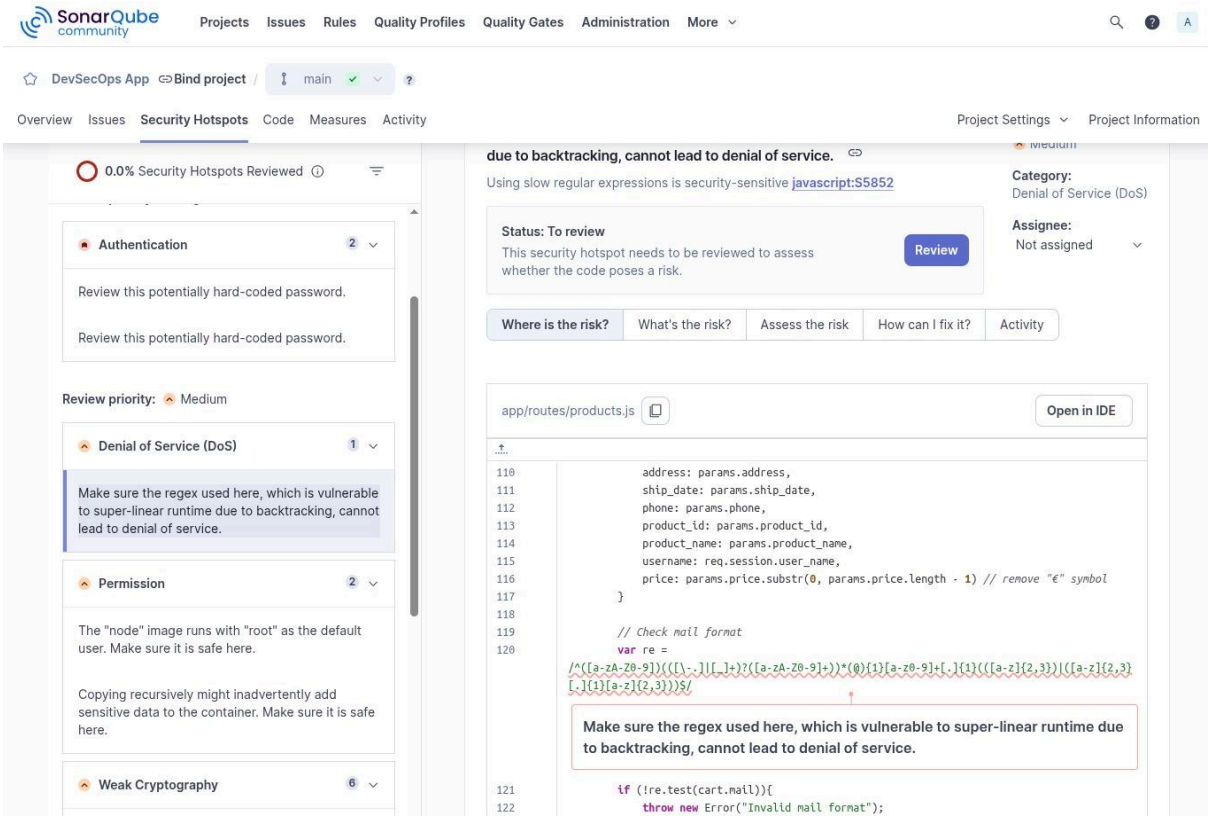

SonarQube community
Projects Issues Rules Quality Profiles Quality Gates Administration More
DevSecOps App Bind project / main
Overview Issues Security Hotspots Code Measures Activity
Project Settings Project Information
My Issues All
Bulk Change
Select issues
Navigate to issue
4 issues 1h 32min effort
Filters
Clear All Filters
Looking for Bugs, Vulnerabilities, or Code Smells? If your team prefers working with these types, change it in the settings
Issues in new code
Software Quality
Security 3
Reliability 0
Maintainability 1
Severity 1
Blocker 4
High 217
Medium 32
Low 17
app/config.js
Make sure this PostgreSQL database password gets changed and removed from the code.
Responsibility
Security Blocker
cwe +
Open Not assigned L4 • 30min effort • 15 minutes ago
Make sure this PostgreSQL database password gets changed and removed from the code.
Responsibility
Security Blocker
cwe +
Open Not assigned L12 • 30min effort • 15 minutes ago
Make sure this PostgreSQL database password gets changed and removed from the code.
Responsibility
Security Blocker
cwe +
Open Not assigned L20 • 30min effort • 15 minutes ago
app/public/js/freewall.js
Add the "let", "const" or "var" keyword to this declaration of "$item" to make it explicit.
Intentionality

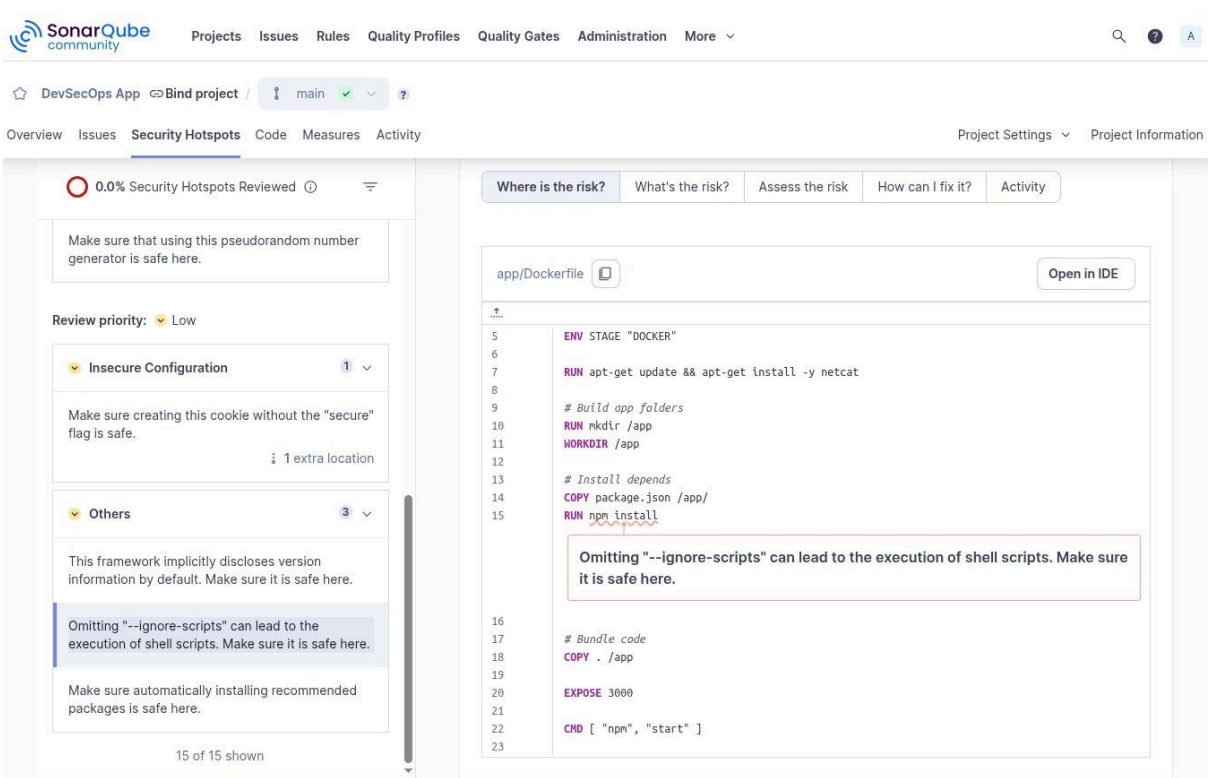

SonarQube community
Projects Issues Rules Quality Profiles Quality Gates Administration More
DevSecOps App Bind project / main
Overview Issues Security Hotspots Code Measures Activity
Project Settings Project Information
0.0% Security Hotspots Reviewed
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Review priority: Low
Insecure Configuration 1
Make sure creating this cookie without the "secure" flag is safe.
1 extra location
Location 1
Others 3
15 of 15 shown
Where What Assess How Activity
Review
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
// uncomment after placing your favicon in /public
app.use(logger('combined', {stream: accessLogStream}));
app.use(bodyParser());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
app.use(session({
Make sure creating this cookie without the "secure" flag is safe.
secret: 'ñasddfilhpaf78h78032h780g780fg780asg780dsbovncubuyvqy',
cookie: {
secure: false,
maxAge: 99999999999
}
}));
/*
* Routes config
*/
app.use('', products);
app.use('', login);

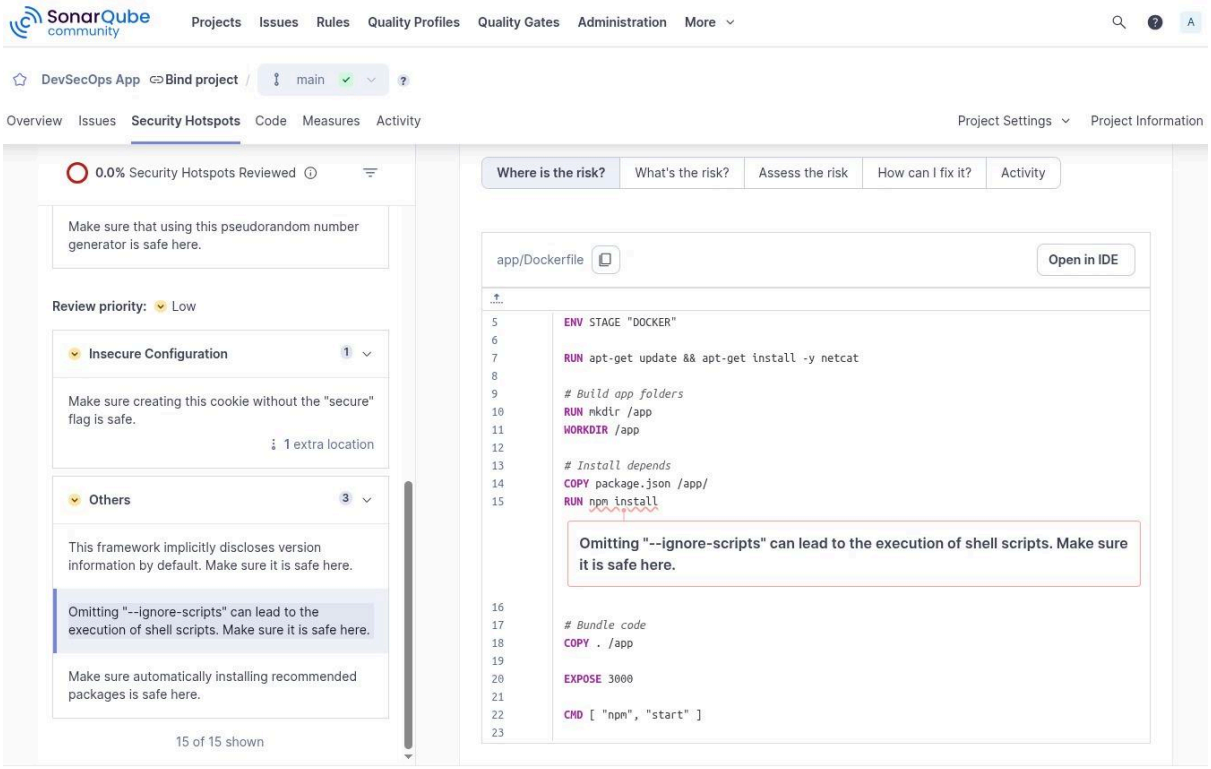

SonarQube community
Projects Issues Rules Quality Profiles Quality Gates Administration More
DevSecOps App Bind project / main
Overview Issues Security Hotspots Code Measures Activity
Project Settings Project Information
0.0% Security Hotspots Reviewed
Permission 2
The "node" image runs with "root" as the default user. Make sure it is safe here.
Copying recursively might inadvertently add sensitive data to the container. Make sure it is safe here.
Weak Cryptography 6
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Status: To review
This security hotspot needs to be reviewed to assess whether the code poses a risk.
Review
Where is the risk? What's the risk? Assess the risk How can I fix it? Activity
app/dummy.js
Open in IDE
{
"username": "roberto",
"password": "asdfpiuw981"
}
],
"products": [
{
"name": "My public privacy",
"description": "Grant privacy in public to watch your favorite programs",
"price": parseInt(Math.random() * 100),
Make sure that using this pseudorandom number generator is safe here.
"image": "product_1.jpg"
},
{
"name": "The USB rocket",
"description":
"Be happy with your USB rocket. Functionality: none. Usability: none. The best choice!",
"price": parseInt(Math.random() * 100),
"image": "product_2.jpg"
},
{

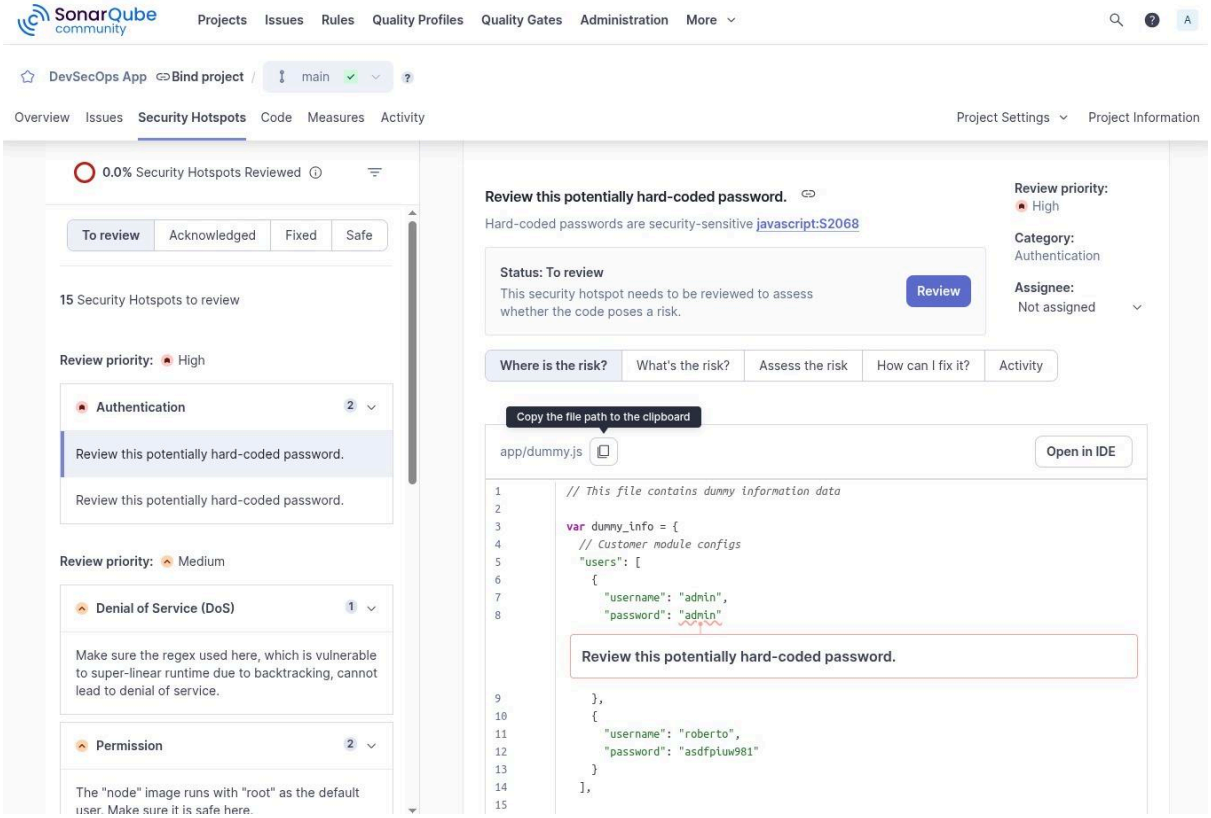


Detalhes das vulnerabilidades


SonarQube community
Projects Issues Rules Quality Profiles Quality Gates Administration More
DevSecOps App Bind project main
Overview Issues Security Hotspots Code Measures Activity
Project Settings Project Information
0.0% Security Hotspots Reviewed
Authentication 2
Review this potentially hard-coded password.
Review this potentially hard-coded password.
Review priority: Medium
Denial of Service (DoS) 1
Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service.
Permission 2
The "node" image runs with "root" as the default user. Make sure it is safe here.
Copying recursively might inadvertently add sensitive data to the container. Make sure it is safe here.
Weak Cryptography 6
due to backtracking, cannot lead to denial of service.
Using slow regular expressions is security-sensitive javascript:S5852
Category: Denial of Service (DoS)
Status: To review
This security hotspot needs to be reviewed to assess whether the code poses a risk.
Review
Assignee: Not assigned
Where is the risk? What's the risk? Assess the risk How can I fix it? Activity
app/routes/products.js
Open in IDE
address: params.address,
ship_date: params.ship_date,
phone: params.phone,
product_id: params.product_id,
product_name: params.product_name,
username: req.session.user_name,
price: params.price.substr(0, params.price.length - 1) // remove "€" symbol
// Check mail format
var re =
/^([a-zA-Z0-9])(([\-.]|[_]+)?([a-zA-Z0-9]+))*(@){1}[a-z0-9]+[.]{1}(([a-z]{2,3})|([a-z]{2,3}[.]{1}[a-z]{2,3}))$/
Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service.
if (!re.test(cart.mail)){
throw new Error("Invalid mail format");


SonarQube community
Projects Issues Rules Quality Profiles Quality Gates Administration More
DevSecOps App Bind project main
Overview Issues Security Hotspots Code Measures Activity
Project Settings Project Information
0.0% Security Hotspots Reviewed
Make sure that using this pseudorandom number generator is safe here.
Review priority: Low
Insecure Configuration 1
Make sure creating this cookie without the "secure" flag is safe.
1 extra location
Others 3
This framework implicitly discloses version information by default. Make sure it is safe here.
Omitting "--ignore-scripts" can lead to the execution of shell scripts. Make sure it is safe here.
Make sure automatically installing recommended packages is safe here.
15 of 15 shown
Where is the risk? What's the risk? Assess the risk How can I fix it? Activity
app/Dockerfile
Open in IDE
ENV STAGE "DOCKER"
RUN apt-get update && apt-get install -y netcat
Build app folders
RUN mkdir /app
WORKDIR /app
Install depends
COPY package.json /app/
RUN npm install
Omitting "--ignore-scripts" can lead to the execution of shell scripts. Make sure it is safe here.
Bundle code
COPY . /app
EXPOSE 3000
CMD ["npm", "start"]


SonarQube community
Projects Issues Rules Quality Profiles Quality Gates Administration More
DevSecOps App Bind project main
Overview Issues Security Hotspots Code Measures Activity
Project Settings Project Information
0.0% Security Hotspots Reviewed
Make sure that using this pseudorandom number generator is safe here.
Review priority: Low
Insecure Configuration 1
Make sure creating this cookie without the "secure" flag is safe.
1 extra location
Others 3
This framework implicitly discloses version information by default. Make sure it is safe here.
Omitting "--ignore-scripts" can lead to the execution of shell scripts. Make sure it is safe here.
Make sure automatically installing recommended packages is safe here.
15 of 15 shown
Where is the risk? What's the risk? Assess the risk How can I fix it? Activity
app/Dockerfile
Open in IDE
ENV STAGE "DOCKER"
RUN apt-get update && apt-get install -y netcat
Build app folders
RUN mkdir /app
WORKDIR /app
Install depends
COPY package.json /app/
RUN npm install
Omitting "--ignore-scripts" can lead to the execution of shell scripts. Make sure it is safe here.
Bundle code
COPY . /app
EXPOSE 3000
CMD ["npm", "start"]


SonarQube community
Projects Issues Rules Quality Profiles Quality Gates Administration More
DevSecOps App Bind project main
Overview Issues Security Hotspots Code Measures Activity
Project Settings Project Information
0.0% Security Hotspots Reviewed
To review Acknowledged Fixed Safe
15 Security Hotspots to review
Review priority: High
Authentication 2
Review this potentially hard-coded password.
Review this potentially hard-coded password.
Review priority: Medium
Denial of Service (DoS) 1
Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service.
Permission 2
The "node" image runs with "root" as the default user. Make sure it is safe here.
Review this potentially hard-coded password.
Hard-coded passwords are security-sensitive javascript:S2068
Status: To review
This security hotspot needs to be reviewed to assess whether the code poses a risk.
Review
Review priority: High
Category: Authentication
Assignee: Not assigned
Where is the risk? What's the risk? Assess the risk How can I fix it? Activity
Copy the file path to the clipboard
app/dummy.js
Open in IDE
// This file contains dummy information data
var dummy_info = {
// Customer module configs
"users": [
{
"username": "admin",
"password": "admin"
Review this potentially hard-coded password.
},
{
"username": "roberto",
"password": "asdfpiuw981"
}
],

Conclusão

Nenhuma das ferramentas listadas é redundante, ou seja, elas são complementarse e cobrem diferentes estágios e tipos de risco:

- **Gitleaks** protege contra o vazamento de *credenciais*.
- **SonarQube** protege contra falhas no *código-fonte escrito pela equipe* (SAST).
- **Dependency Tracker** protege contra vulnerabilidades conhecidas no *supply chain* (SCA) de forma *contínua e centralizada*, mesmo após o deploy.

A adoção conjunta dessas quatro ferramentas fornece uma cobertura de segurança automatizada, multicamada e alinhada com as práticas de "Shift-Left". Elas permitem que a equipe de desenvolvimento identifique e corrija falhas de segurança (sejam elas segredos, bugs de lógica ou dependências vulneráveis) de forma rápida e eficiente dentro do fluxo de trabalho de CI/CD.