Intégration de SonarQube dans la pipeline Wayhost

Ce document décrit le plan d'intégration de SonarQube dans la chaîne de sécurité existante de Wayhost (auth-api, panel, website), en complément des outils déjà en place — gitleaks, Semgrep, bun audit, Trivy, Dockle, Kubesec.

Pourquoi SonarQube sur Wayhost

La pipeline actuelle couvre déjà la détection de secrets (gitleaks), le SAST (Semgrep), le SCA (bun audit), le scan de conteneurs (Trivy, Dockle) et la configuration Kubernetes (Trivy config, Kubesec). SonarQube n'a pas vocation à dupliquer ces détections — sa valeur ajoutée se situe sur deux axes que la pipeline actuelle ne couvre pas :

Dette technique mesurée — Semgrep détecte des patterns de vulnérabilités précis, mais ne calcule pas de score global de maintenabilité (complexité cyclomatique, duplication de code, taille des fonctions). SonarQube fournit cette vue agrégée.

Quality Gate configurable — un seuil unique qui peut bloquer la pipeline si la qualité du code descend sous un niveau défini, distinct des seuils de sévérité HIGH/CRITICAL déjà utilisés par les scanners de sécurité.

La vue Security Hotspots de SonarQube reste complémentaire à Semgrep plutôt que redondante : elle couvre des zones de risque au niveau de l'infrastructure-as-code (Dockerfile, manifests) que Semgrep, orienté code applicatif, ne couvre pas nativement.

Prérequis avant intégration en CI/CD

Trois conditions doivent être réunies avant de basculer d'un test local à une intégration continue dans `.gitlab-ci.yml` :

1. Hébergement continu du serveur SonarQube

Le test en local via `docker run -d --name sonarqube -p 9000:9000 sonarqube:lts-community` n'est pas viable pour une pipeline CI/CD — le serveur doit être accessible en permanence depuis les runners GitLab. Options à évaluer :

- VPS dédié avec SonarQube en Docker Compose, suivant le même modèle que le déploiement DefectDojo existant (`redteam.wayhost.cloud:8443`)
- Service managé SonarQube Cloud (anciennement SonarCloud) si l'hébergement auto-géré n'est pas souhaité

2. Token stocké en variable CI/CD sécurisée

Le token d'analyse ne doit jamais apparaître en clair dans `.gitlab-ci.yml`. Il doit être créé dans **Settings > CI/CD > Variables** avec les options **Masked** et **Protected** activées, suivant exactement le même modèle que `DISCORD_WEBHOOK` et `DEFECTDOJO_URL` déjà en place.

```
variables:
  SONAR_HOST_URL: "https://sonarqube.wayhost.cloud"
  SONAR_TOKEN: $SONAR_TOKEN  # variable CI/CD masquée
```

3. Quality Gate adapté au projet existant

Les seuils par défaut de SonarQube (souvent calibrés pour un projet neuf) sont généralement trop stricts pour un code déjà en production. Un Quality Gate trop sévère bloquerait chaque pipeline sans réelle valeur ajoutée au démarrage. La démarche recommandée :

- Lancer une première analyse en mode informatif (sans blocage de pipeline) pour établir une baseline
- Définir un Quality Gate progressif, qui se concentre d'abord sur le **nouveau code** (New Code) plutôt que sur l'historique complet — c'est le mode de fonctionnement par défaut recommandé par SonarQube pour les projets brownfield

Structure du job proposé

À insérer dans le stage `security`, après `semgrep-scan` et avant `k8s-config-scan`, en suivant les conventions déjà établies dans la pipeline (cache, `interruptible`, notification Discord en cas d'échec).

```
        sonarqube-scan:
          stage: security
          image: sonarsource/sonar-scanner-cli:4.8
          interruptible: true
          variables:
            GIT_DEPTH: 0
            SONAR_HOST_URL: "https://sonarqube.wayhost.cloud"
          cache:
            key: sonar-cache-${CI_PROJECT_NAME}
            paths:
              - .sonar/cache
          needs:
            - job: semgrep-scan
          script:
            - sonar-scanner
                -Dsonar.projectKey=${CI_PROJECT_NAME}
                -Dsonar.sources=.
                -Dsonar.host.url=${SONAR_HOST_URL}
                -Dsonar.login=${SONAR_TOKEN}
                -Dsonar.qualitygate.wait=true
            - |
              if [ $? -ne 0 ]; then
                [ "$CI_COMMIT_REF_NAME" = "main" ] && export ENVIRONMENT="PRODUCTION" || export
ENVIRONMENT="DEVELOPMENT"
                curl -s -X POST "$DISCORD_WEBHOOK" \
                  -H "Content-Type: application/json" \
                  -d "{\"content\":\"🚨 **Echec SONARQUBE - ${ENVIRONMENT}** | Quality Gate non
respecté | Projet: ${CI_PROJECT_NAME} | Branch: ${CI_COMMIT_REF_NAME} | Pipeline:
${CI_PIPELINE_URL}\"}" || true
              fi
          <<: *security_rules_critical
          tags:
            - wayhost-deploy
```

Note sur `GIT_DEPTH: 0` — comme pour gitleaks, SonarQube bénéficie de l'historique Git complet pour son analyse SCM (attribution des lignes de code, blâme), bien que ce ne soit pas strictement

obligatoire contrairement à gitleaks.

Note sur `-Dsonar.qualitygate.wait=true` — cette option force le scanner à attendre le traitement du rapport côté serveur et à retourner un exit code non nul si le Quality Gate échoue, rendant le blocage de pipeline possible.

Versionnement du scanner

Suivant le principe déjà appliqué sur gitleaks et Trivy dans la pipeline existante, la version du scanner SonarQube doit être fixée explicitement plutôt que de suivre une étiquette flottante :

```
        image: sonarsource/sonar-scanner-cli:4.8.1
```

À ajuster selon la version Java disponible côté serveur SonarQube hébergé — vérifier la compatibilité avant tout changement de version du serveur ou du scanner.

Phasage recommandé

Phase	Action	Blocage pipeline
1	Déploiement du serveur SonarQube sur VPS dédié	—
2	Premier scan sur chaque microservice (auth-api, panel, website) en mode informatif	Non
3	Revue manuelle des Security Hotspots et établissement de la baseline	Non
4	Activation du Quality Gate sur le nouveau code uniquement	Oui, sur New Code
5	Extension progressive du Quality Gate au code existant si pertinent	À évaluer selon la dette technique réelle observée

Ce phasage évite de bloquer immédiatement les pipelines existantes sur des centaines de findings hérités, tout en garantissant que tout nouveau code respecte un niveau de qualité défini dès son introduction.