Secure DevSecOps CI/CD Pipeline – From Code to Production

Projet personnel DevSecOps – Sécurité intégrée au cycle CI/CD

Auteur : Katia AIT BACHIR
Rôle ciblé : DevSecOps / Cloud Security Engineer
Date : 2026
Repo GitHub : (à ajouter plus tard)

---

1 INTRODUCTION

1.1 Contexte

Avec l'adoption massive du **CI/CD**, les pipelines sont devenues des **cibles critiques** pour les attaquants (supply chain attacks, secrets leaks, images compromises).

La sécurité doit être **intégrée dès le début** (Shift Left Security).

1.2 Objectifs du projet

- Concevoir une **pipeline CI/CD sécurisée**
- Automatiser les contrôles de sécurité
- Bloquer les déploiements non conformes
- Simuler des scénarios d'attaque réalistes
- Documenter les décisions comme en entreprise

1.3 Périmètre

Inclus :

- CI/CD
- Sécurité du code
- Sécurité des dépendances
- Sécurité des conteneurs
- Sécurité Infrastructure as Code
- Monitoring & détection

Exclus :

- Haute disponibilité production
- Cloud payant réel (AWS/GCP)

2️⃣ ARCHITECTURE GLOBALE

2.1 Vue d'ensemble

```
Developer → GitHub → GitHub Actions
   ├── SAST
   ├── SCA
   ├── Secret Scanning
   ├── Build Docker
   ├── Container Scan
   ├── IaC Scan
   └── Deploy Kubernetes
```

2.2 Choix technologiques (JUSTIFIÉS)

Composant	Outil	Justification
CI/CD	GitHub Actions	Standard industrie, intégration native GitHub
SAST	Semgrep	Rapide, règles security-oriented
SCA	OWASP Dependency-Check	Open-source, CVE fiables
Secrets	Gitleaks	Référence pour détection de secrets
Container Scan	Trivy	Simple, efficace, très utilisé
IaC	Terraform + Checkov	Policy as Code
Orchestration	Kubernetes	Standard cloud

👉 **Très apprécié en entretien** : tu sais expliquer *pourquoi*.

3️⃣ MODÈLE DE MENACES (THREAT MODELING)

Objectif du threat modeling

Identifier **les menaces de sécurité** pesant sur la pipeline CI/CD afin de :

- réduire les risques
- définir des contrôles de sécurité adaptés
- empêcher les attaques de type **supply chain**

---

1 PÉRIMÈTRE DU THREAT MODELING

📌 Système analysé

Pipeline CI/CD DevSecOps basée sur :

- GitHub
- GitHub Actions
- Docker
- Kubernetes
- Terraform

📌 Ce qui est inclus

- Code source
- Pipeline CI/CD
- Images Docker
- Secrets
- Infrastructure as Code
- Déploiement Kubernetes

---

2 IDENTIFICATION DES ACTIFS CRITIQUES

👉 **Un actif = ce qui a de la valeur et doit être protégé**

Actif	Description	Criticité
Code source	Code de l'application	Élevée
Pipeline CI/CD	Automatisation build & deploy	Critique
Secrets CI/CD	Tokens, clés API	Critique
Images Docker	Artefacts déployés	Élevée
Infra (Terraform)	Config cloud/K8s	Élevée
Cluster Kubernetes	Environnement d'exécution	Critique

---

3 IDENTIFICATION DES ACTEURS / ATTAQUANTS

Acteur	Description
Attaquant externe	Personne sans accès GitHub
Développeur malveillant	Accès au repo
Compte GitHub compromis	Vol de token
Dépendance compromise	Librairie vulnérable
Action GitHub vulnérable	Supply chain

---

4 IDENTIFICATION DES MENACES (STRIDE)

On applique la méthode **STRIDE** aux actifs.

🟥 S — Spoofing (Usurpation d'identité)

Menace :
Vol d'un token GitHub Actions

- **Actif ciblé :** Pipeline CI/CD

- **Scénario :** Token exposé → attaquant déclenche des pipelines
- **Impact :** Déploiement malveillant

Contrôles de sécurité :

- Secrets GitHub chiffrés
- Permissions minimales (least privilege)
- Rotation des secrets

🟧 T — Tampering (Altération)

Menace :
Image Docker modifiée avant déploiement

- **Actif ciblé :** Image Docker
- **Scénario :** Injection de code malveillant
- **Impact :** Backdoor en production

Contrôles :

- Trivy (scan image)
- Build depuis Dockerfile contrôlé
- Images immuables

🟨 R — Repudiation (Non-répudiation)

Menace :
Impossible d'identifier l'auteur d'une action

- **Actif ciblé :** Pipeline
- **Scénario :** Action malveillante non tracée

- **Impact :** Enquête impossible

Contrôles :

- Logs GitHub Actions
- Historique Git
- Audit Kubernetes

🟦 I — Information Disclosure (Divulgation d'informations)

Menace :
Secret exposé dans un commit Git

- **Actif ciblé :** Secrets CI/CD
- **Scénario :** Clé API committée
- **Impact :** Accès non autorisé

Contrôles :

- Gitleaks
- Blocage pipeline
- Rotation immédiate des secrets

⭐ **Menace la plus réaliste et la plus appréciée en entretien**

🟪 D — Denial of Service (Déni de service)

Menace :
Pipeline bloquée volontairement

- **Actif ciblé :** CI/CD
- **Scénario :** Jobs lourds ou loops

- **Impact :** Blocage des déploiements

Contrôles :

- Timeouts GitHub Actions
- Quotas
- Jobs parallèles limités

⬛ E — Elevation of Privilege (Élévation de privilèges)

Menace :
Pod Kubernetes exécuté en root

- **Actif ciblé :** Cluster K8s
- **Scénario :** Exploitation d'un conteneur
- **Impact :** Compromission du cluster

Contrôles :

- SecurityContext Kubernetes
- Non-root containers
- RBAC strict

---

5 SYNTHÈSE DES RISQUES (TABLEAU FINAL)

Menace	Impact	Probabilité	Niveau
Secret exposé	Critique	Élevée	🔴
Image compromise	Élevé	Moyen	🟠
Token volé	Critique	Moyen	🔴
Pod en root	Élevé	Moyen	🟠

Pipeline bloquée	Moyen	Faible	🟡

---

6 LIEN AVEC LA PIPELINE CI/CD (TRÈS IMPORTANT)

👉 Chaque **menace** correspond à **un outil sécurité** dans la pipeline :

Menace	Outil
Secret exposé	Gitleaks
Code vulnérable	Semgrep
Dépendance vulnérable	Dependency-Check
Image compromise	Trivy
IaC non sécurisée	Checkov

➡️ **Cohérence totale** entre rapport et implémentation.

🎤 Phrase GOLD en entretien

> *“Le threat modeling a guidé le design de ma pipeline CI/CD et le choix des outils de sécurité.”*

💥 Très professionnel.

4️⃣ PIPELINE CI/CD SÉCURISÉE

4.1 Étapes CI

1. Checkout du code
2. SAST
3. SCA
4. Secret scanning

5. Build Docker
6. Scan image

4.2 Security Gates 🚨

La pipeline **échoue automatiquement** si :

- Vulnérabilité CRITIQUE détectée
- Secret exposé
- Image Docker non conforme
- Infra non sécurisée

👉 **Concept clé DevSecOps**.

5️⃣ SÉCURITÉ DÉTAILLÉE (CŒUR DU RAPPORT 🔥)

5.1 SAST

- Outil : Semgrep
- Vulnérabilités détectées :
    - Injection
    - Mauvaises pratiques
- Exemple réel (sera ajouté après implémentation)

5.2 SCA

- Détection CVE
- Score CVSS

- Décision :
    - Upgrade
    - Acceptation du risque
    - Mitigation

5.3 Secret Scanning

- Détection d’un secret simulé
- Blocage pipeline
- Rotation du secret

5.4 Sécurité des conteneurs

- Scan image Docker
- Suppression des vulnérabilités critiques
- Image finale durcie

5.5 Sécurité IaC

- Détection :
    - Ports ouverts
    - Mauvaise config réseau
- Correction Terraform

6️⃣ SCÉNARIO D’ATTAQUE SIMULÉ ⭐⭐⭐

Exemple : fuite de secret

1. Secret ajouté volontairement dans un commit
2. Détection par Gitleaks
3. Pipeline stoppée
4. Analyse d’impact
5. Correctif
6. Leçons apprises

👉 **Les recruteurs adorent cette partie**.

7️⃣ INCIDENT RESPONSE (MINI)

- Détection
- Containment
- Eradication
- Recovery
- Lessons learned

8️⃣ BONNES PRATIQUES DEVSECOPS

- Shift Left
- Zero Trust
- Least Privilege
- Automation
- Policy as Code

9️⃣ CONCLUSION & PERSPECTIVES

- Compétences acquises
- Limites du projet
- Évolutions possibles (cloud réel, SOC, SIEM)

—----------

info :

Threat modeling = identifier et analyser les menaces de sécurité AVANT qu'elles arrivent.

C'est répondre à 4 questions essentielles :

1. **Qu'est-ce que je protège ?** (actifs)
2. **Contre qui ?** (attaquants)
3. **Comment on peut m'attaquer ?** (menaces)
4. **Que je mets en place pour me protéger ?** (contrôles)

Méthode la plus utilisée : STRIDE

STRIDE est une **méthode simple et très reconnue**.

Lettre	Signification	Exemple CI/CD
S	Spoofing	Vol d’un token GitHub
T	Tampering	Image Docker modifiée
R	Repudiation	Actions non tracées
I	Information Disclosure	Secret exposé
D	Denial of Service	Pipeline bloquée
E	Elevation of Privilege	Pod Kubernetes en root