

# GUIDE D'IMPLÉMENTATION TECHNIQUE

Ce document détaille les procédures d'installation et de configuration pour chaque outil de sécurité mentionné dans la stratégie de défense. Chaque section explique le rôle de l'outil, sa place dans l'infrastructure et fournit un guide pas-à-pas pour son déploiement.

---

## AXE 1 : TUTORIELS DE PRÉVENTION

### 1. Calico — Micro-segmentation

#### Rôle / place :

Calico s'intègre au cluster Kubernetes comme CNI. Il fonctionne comme pare-feu distribué en appliquant des **NetworkPolicy** entre pods — essentiel pour un modèle Zero Trust.

#### Installation (opérateur Calico) :

- `kubectl create -f`  
<https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/tigera-operator.yaml>
- `kubectl create -f`  
<https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/custom-resources.yaml>

#### Vérification :

- `kubectl get pods -n calico-system`

#### Exemple — isoler la base de données (playbook Ansible)

Fichier **deploy\_db\_policy.yml** :

- - name: Appliquer la politique réseau de la base de données
- hosts: localhost
- tasks:
- - name: Déployer la NetworkPolicy
- kubernetes.core.k8s:
- state: present

- definition:
- apiVersion: networking.k8s.io/v1
- kind: NetworkPolicy
- metadata:
- name: allow-api-to-db
- namespace: production
- spec:
- podSelector:
- matchLabels:
- app: db-poufsouffle
- policyTypes:
- - Ingress
- ingress:
- - from:
- - podSelector:
- matchLabels:
- app: api-serdaigle
- ports:
- - protocol: TCP
- port: 5432

Exécution : `ansible-playbook deploy_db_policy.yml`

---

## 2. HashiCorp Vault — Gestion des secrets

### Rôle / place :

Vault est le coffre-fort central (souvent hors du cluster). Les applications obtiennent un jeton d'identité et récupèrent les secrets dynamiquement (mot de passe DB temporaire, etc.).

### Provisionnement (ex. Terraform) :

- resource "google\_compute\_instance" "vault\_server" {
- name = "vault-server-prod"
- machine\_type = "e2-medium"
- zone = "europe-west1-b"
- # ... boot disk, réseau, etc.
- }

### Installation / configuration (ex. playbook Ansible — simplifié) :

- - name: Installer et configurer Vault

- hosts: vault\_servers
- tasks:
- - name: Installer Vault
- # téléchargement + installation du binaire
- - name: Configurer Vault
- # création du fichier de conf + service systemd
- - name: Initialiser et désealer Vault
- # commande `vault operator init` et distribution des clés

### Exemple — Authentification Kubernetes :

- vault auth enable kubernetes
- 
- vault write auth/kubernetes/config \
- token\_reviewer\_jwt="..." \
- kubernetes\_host="..." \
- kubernetes\_ca\_cert="..."
- 
- vault write auth/kubernetes/role/api-serdaigle \
- bound\_service\_account\_names=api-serdaigle \
- bound\_service\_account\_namespaces=production \
- policies=db-access-policy \
- ttl=24h

Côté application : lecture du token du ServiceAccount → échange avec Vault → obtention d'un token Vault → demande d'un secret dynamique (souvent automatisé via sidecar/injector).

---

## 3. Sécurisation CI/CD — Trivy, Cosign, Harbor

### Rôle / place :

S'intègrent dans le pipeline CI (ex: GitLab CI). Trivy scanne les images, Cosign signe, Harbor stocke les images validées.

### Pipeline (extrait `.gitlab-ci.yml`) :

- stages:
- - build
- - scan
- - sign\_and\_push
-

- build\_image:
- stage: build
- script:
- - docker build -t \$CI\_REGISTRY\_IMAGE:\$CI\_COMMIT\_SHA .
- 
- scan\_image:
- stage: scan
- image: aquasec/trivy:latest
- script:
- - trivy image --exit-code 1 --severity CRITICAL  
\$CI\_REGISTRY\_IMAGE:\$CI\_COMMIT\_SHA
- 
- sign\_and\_push\_image:
- stage: sign\_and\_push
- image: gcr.io/projectsigstore/cosign:v1.3.1
- script:
- - echo \$HARBOR\_PASSWORD | docker login harbor.votre-ecole.fr -u  
\$HARBOR\_USER --password-stdin
- - docker push  
harbor.votre-ecole.fr/production/\$CI\_PROJECT\_NAME:\$CI\_COMMIT\_SHA
- - cosign sign  
harbor.votre-ecole.fr/production/\$CI\_PROJECT\_NAME:\$CI\_COMMIT\_SHA

**Notes :** Installer Harbor (VM ou k8s), stocker les credentials sécurisés dans GitLab CI variables ([HARBOR\\_USER](#), [HARBOR\\_PASSWORD](#)), et activer signatures OIDC pour Cosign si possible.

---

## AXE 2 : TUTORIELS DE DÉTECTION

### 4. Falco — Détection d'intrusion temps réel

#### Rôle / place :

DaemonSet sur chaque nœud ; surveille appels systèmes et déclenche alertes sur comportements suspects.

#### Installation via Helm :

- helm repo add falcosecurity https://falcosecurity.github.io/charts
- helm repo update
- 
- helm install falco falcosecurity/falco \
- --namespace falco --create-namespace \
- --set falcosidekick.enabled=true \

- --set falcosidekick.webui.enabled=true

### Règle personnalisée (ex. écriture dans /etc) :

Fichier `custom-rules.yaml` :

- - rule: Write below etc
- desc: an attempt to write to /etc file
- condition: open\_write and fd.name contains /etc/ and not proc\_name in (user\_known\_write\_etc\_procs)
- output: "File below /etc opened for writing (user=%user.name command=%proc.cmdline file=%fd.name)"
- priority: ERROR

Mettre à jour le déploiement Helm pour charger `custom-rules.yaml`.

---

## 5. Pile EFK — Centralisation des logs (Elasticsearch / Fluentd / Kibana)

### Rôle / place :

Fluentd en DaemonSet collecte logs ; Elasticsearch stocke/indexe ; Kibana offre UI pour recherche/visualisation.

### Installation (Helm) :

- helm repo add elastic https://helm.elastic.co
- helm install elasticsearch elastic/elasticsearch --namespace logging
- 
- # Fluentd (configurer elasticsearch.host)
- helm install fluentd fluent/fluentd --namespace logging --set elasticsearch.host="elasticsearch-master"
- 
- helm install kibana elastic/kibana --namespace logging

### Exemple — tableau de bord erreurs 4xx :

Dans Kibana → Discover : filtrer sur `kubernetes.namespace_name: "production"` et `http_status_code: 4*`. Créer une visualisation et l'ajouter à un tableau de bord.

---

## AXE 3 : TUTORIELS DE DÉFENSE ACTIVE

### 6. Kube-Honeypot — Leurres

#### Rôle / place :

Déployé dans un namespace isolé ; expose de fausses ressources pour attirer et observer les attaquants.

#### Déploiement (extrait) :

- `kubectl create namespace honeypot`
- `kubectl apply -f honeypot-deployment.yaml -n honeypot`

#### Isolation (NetworkPolicy) — `honeypot-deny-all.yaml` :

- `apiVersion: networking.k8s.io/v1`
- `kind: NetworkPolicy`
- `metadata:`
  - `name: deny-all-egress`
  - `namespace: honeypot`
- `spec:`
  - `podSelector:`
    - `matchLabels:`
      - `app: kube-honeypot`
  - `policyTypes:`
    - `- Egress`
  - `egress: []`

```
kubectl apply -f honeypot-deny-all.yaml
```

---

### 7. Canary Tokens — Marqueurs de détection passive


#### Rôle / place :

Fichiers/fausses clés disséminés (conteneurs, partages, repo). Ils déclenchent une alerte lorsqu'ils sont utilisés/accédés.

#### Génération & déploiement :


- Générer tokens sur [Canarytokens.org](https://canarytokens.org) (ou équivalent).

- Stocker tokens dans un répertoire sécurisé géré par Ansible.
- Playbook d'exemple :
  - - name: Disséminer des Canary Tokens
  - hosts: kubernetes\_nodes
  - tasks:
    - - name: Placer un faux fichier de configuration AWS
    - copy:
      - dest: "/etc/secrets/aws\_credentials.conf"
      - content: |
      - [default]
      - aws\_access\_key\_id = AAAAAA... # fausse clé
      - aws\_secret\_access\_key = XXXXXX...
  - - name: Placer le document Word piégé sur un partage
    - copy:
      - src: files/Accès\_Production\_Confidentiel.docx
      - dest: /mnt/partage\_commun/Direction/




## Your Canarytoken was triggered

**MS Word** Canarytoken has been triggered by the Source IP  
**195.15.137.164**



**Reminder**  
workshop 25



**Source IP**  
195.15.137.164

---

## 8. CrowdSec — Neutralisation automatisée

### Rôle / place :

Agent sur nœuds/serveurs exposés ; bouncers appliquent les décisions (ex: Nginx).  
Partage communautaire des IP malveillantes.

### Installation Agent (Debian/Ubuntu) :

- `curl -s https://packagecloud.io/install/repositories/crowdsec/crowdsec/script.deb.sh | sudo bash`
- `sudo apt-get install crowdsec`

### Installer un bouncer (ex: Nginx) :

- `sudo apt-get install crowdsec-nginx-bouncer`

### Commandes utiles :

- `sudo cscli metrics`
- `sudo cscli decisions list`

**Exemple d'action :** détection d'un scan → décision de bannir l'IP → bouncer Nginx applique rejet 403 → décision partagée à la communauté.

---

## 9. Tarptitting — ralentir la reconnaissance

### Rôle / place :

Technique appliquée au périmètre (pare-feu Linux) pour ralentir les scanners.

### Exemple (iptables) :

- `sudo iptables -N TARPIT`
- `sudo iptables -A TARPIT -p tcp -j TCPMSS --set-mss 1`
- `# réduire la MSS et rejeter avec reset (implémentation simple)`



- `sudo iptables -A TARPIT -j REJECT --reject-with tcp-reset`
- 
- `# Autoriser connexions établies`
- `sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT`
- 
- `# Autoriser services légitimes`
- `sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT`
- `sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT`
- `sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT`
- 
- `# Envoyer le reste vers le pot de goudron`
- `sudo iptables -A INPUT -p tcp -j TARPIT`