AYI NEDJIMI Consultants

Sigma Rules : Guide Complet d'Écriture et Déploiement de

Catégorie : SOC et Detection Lecture : 8 min Publié le : 08/03/2026 Auteur : Ayi NEDJIMI

Guide complet Sigma Rules : syntaxe YAML, logsources, modifiers, conditions, corrélation temporelle, pySigma backends, conversion multi-SIEM.

2.1 Structure YAML complète

Une règle Sigma est un document YAML structuré en sections clairement définies. Chaque section a un rôle précis dans le processus de détection et de compilation. Voici la structure complète avec toutes les sections disponibles : Guide complet Sigma Rules : syntaxe YAML, logsources, modifiers, conditions, corrélation temporelle, pySigma backends, conversion multi-SIEM. La détection des menaces repose sur la capacité à identifier les comportements malveillants parmi le bruit. Sigma Rules : Guide Complet d'Écriture et Déploiement de fournit des méthodologies éprouvées pour les analystes SOC. Nous abordons notamment : 8. intégration dans un pipeline ci/cd, questions frequentes et 9. conclusion et ressources. Les professionnels y trouveront des recommandations actionnables, des commandes prêtes à l'emploi et des stratégies de mise en œuvre adaptées aux environnements d'entreprise.

```yaml
# Règle Sigma complète - Structure de référence
title: Mimikatz Credential Dumping via LSASS Access
id: 0d894093-71bc-43c3-a4b0-2a0e3e8f0425    # UUID unique
related:
  - id: 27a72a60-7e5e-47b1-9d17-909c9abafdcd
    type: derived                            # derived | obsoletes | merged | renamed |
similar
status: stable                               # test | stable | experimental | deprecated
| unsupported
description: |
  Détecte l'accès mémoire au processus LSASS avec des droits
  caractéristiques de Mimikatz (sekurlsa::logonpasswords).
  Couverture : T1003.001 - OS Credential Dumping: LSASS Memory.
references:
  - https://attack.mitre.org/techniques/T1003/001/
  - https://github.com/gentilkiwi/mimikatz
  - https://posts.specterops.io/operational-guidance-for-offensive-user-dpapi-
abuse-1fb7fac8b107
author: Ayi NEDJIMI, Florian Roth (original)
date: 2026-02-15
modified: 2026-02-28
tags:
  - attack.credential_access
  - attack.t1003.001
  - attack.s0002                             # Mimikatz software
  - detection.dfir
level: high                                  # informational | low | medium | high |
critical
falsepositives:
  - Legitimate security tools scanning LSASS
  - Windows Defender real-time protection
  - Crash dump utilities (procdump for debugging)

logsource:
  category: process_access
  product: windows

detection:
  selection_target:
    TargetImage|endswith: '\lsass.exe'
  selection_access:
    GrantedAccess|contains:
      - '0x1010'
      - '0x1410'
      - '0x1438'
      - '0x143a'
      - '0x1fffff'
  filter_main_legitimate:
    SourceImage|endswith:
      - '\wmiprvse.exe'
      - '\taskmgr.exe'
      - '\procexp64.exe'
      - '\procexp.exe'
  filter_defender:
    SourceImage|contains: '\Microsoft\Windows Defender\'
    SourceImage|endswith: '\MsMpEng.exe'
  filter_crowdstrike:
    SourceImage|startswith: 'C:\Program Files\CrowdStrike\'
  condition: selection_target and selection_access and not 1 of filter_*

fields:
  - SourceImage
```

```
    - TargetImage
    - GrantedAccess
    - SourceProcessGUID
    - CallTrace
```

Analysons chaque section en détail pour comprendre les bonnes pratiques de rédaction.

2.2 Les métadonnées : identité et contexte

Les métadonnées constituent la carte d'identité de la règle. Le champ `id` est un UUID v4 unique qui identifie la règle de manière universelle -- il ne doit jamais être réutilisé, même si la règle est complètement réécrite. Le champ `related` permet de tracer les filiations entre règles (dérivée d'une autre, fusion de plusieurs règles, remplacement d'une règle obsolète).

Le champ `status` contrôle le cycle de vie. Les règles `experimental` sont en phase de test et peuvent générer des faux positifs. Les règles `stable` ont été validées en production. Les règles `test` sont en cours de développement. Le champ `level` indique la sévérité de l'alerte : `critical` pour les détections à très haute confiance nécessitant une réponse immédiate, `high` pour les détections fiables nécessitant une investigation, `medium` et `low` pour les indicateurs de compromission plus bruités.

Les `tags` suivent la convention ATT&CK : `attack.tactic_name` pour les tactiques, `attack.tXXXX` pour les techniques, et `attack.sXXXX` pour les logiciels. Ces tags sont exploités par les outils de couverture MITRE ATT&CK pour générer les matrices de détection, un élément central de l'approche Detection-as-Code.

2.3 La logsource : abstraction des sources de données

La section `logsource` est le mécanisme d'abstraction qui rend Sigma portable. Au lieu de spécifier un index Splunk ou un data stream Elastic, la règle déclare une **catégorie logique** de données. Le backend de compilation traduit ensuite cette catégorie en identifiant concret pour chaque SIEM.

Les trois champs de la logsource sont :

- **category** : le type d'événement -- `process_creation`, `process_access`, `file_event`, `network_connection`, `registry_event`, `dns_query`, `image_load`, etc. Les catégories sont standardisées dans la spécification Sigma.
- **product** : le système d'exploitation ou le produit source -- `windows`, `linux`, `macos`, `azure`, `aws`, `gcp`.
- **service** : le journal d'événements spécifique -- `sysmon`, `security`, `system`, `powershell`, `dns-server`, `firewall`.

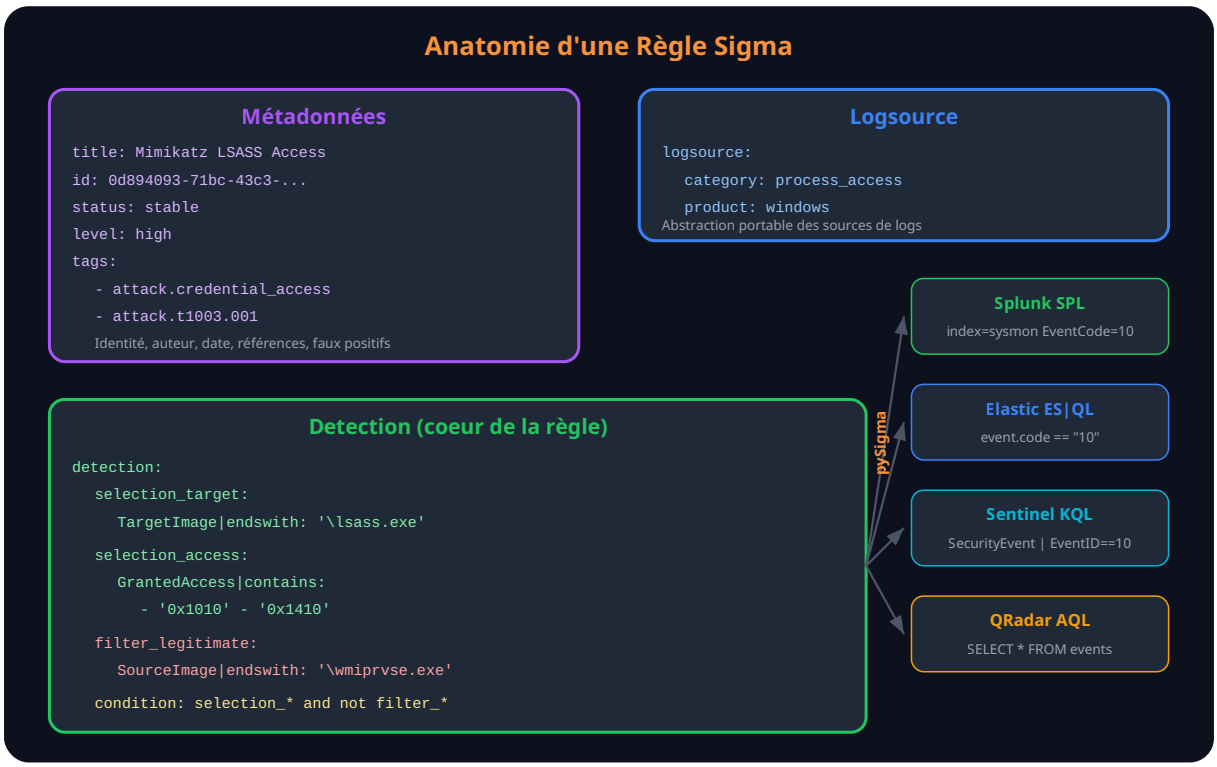

Figure 1 -- Anatomie d'une règle Sigma : métadonnées, logsource, détection et compilation multi-SIEM

La taxonomie des logsources est cruciale pour la portabilité. Les catégories les plus couramment utilisées sont :

Catégorie	Description	Source typique	Event IDs
process_creation	Création de processus	Sysmon EID 1, Security 4688	1, 4688
process_access	Accès mémoire inter-processus	Sysmon EID 10	10
file_event	Création/modification de fichiers	Sysmon EID 11	11
registry_event	Modification du registre	Sysmon EID 12/13/14	12, 13, 14
network_connection	Connexions réseau sortantes	Sysmon EID 3	3
dns_query	Requêtes DNS	Sysmon EID 22	22
image_load	Chargement de DLL	Sysmon EID 7	7
pipe_created	Création de named pipes	Sysmon EID 17/18	17, 18

Notre avis d'expert

Le threat hunting proactif est ce qui distingue un SOC réactif d'un SOC mature. Attendre les alertes ne suffit plus — il faut activement chercher les indicateurs de compromission dans les données historiques et les comportements anormaux.

La condition combine les detection items nommés avec des opérateurs logiques :

```
# Exemples de conditions
condition: selection                          # Simplement le selection
condition: selection and not filter           # Selection moins le filtre
condition: selection1 or selection2           # L'un ou l'autre
condition: selection and not 1 of filter_*    # Wildcard sur les filtres
condition: all of selection_*                 # Tous les selections doivent matcher
condition: 1 of selection_*                   # Au moins un selection matche
condition: selection | count() > 5            # Aggregation : plus de 5 occurrences
condition: selection | count(SourceIP) > 10   # Plus de 10 IPs source distinctes
```

Piège : l'ordre des opérateurs logiques
La condition Sigma utilise la priorité logique standard : `not` > `and` > `or`. Mais pour les cas complexes, utilisez des parenthèses explicites pour éviter toute ambiguïté. La condition `selection1 or selection2 and not filter` est interprétée comme `selection1 or (selection2 and not filter)`, ce qui n'est probablement pas l'intention. Écrivez plutôt `(selection1 or selection2) and not filter`.

3.3 Le pattern de naming : selections et filters

La convention de nommage SigmaHQ recommande un pattern clair pour distinguer les types de detection items :

- **selection_*** : les critères positifs qui identifient le comportement suspect. Exemples : `selection_target`, `selection_access`, `selection_cmdline`.
- **filter_main_*** : les exclusions principales pour les faux positifs bien connus. Exemples : `filter_main_defender`, `filter_main_svchost`.
- **filter_optional_*** : les exclusions secondaires que l'utilisateur peut choisir d'activer ou non selon son environnement. Exemples : `filter_optional_admin_tools`.

Ce pattern permet d'utiliser la condition `selection and not 1 of filter_main_* and not 1 of filter_optional_*` pour combiner élégamment tous les filtres. Il facilite aussi la maintenance : ajouter un nouveau faux positif revient à ajouter un nouveau `filter_main_xxx` sans toucher à la condition.

Cas concret
L'opération de threat hunting menée par CrowdStrike lors de l'investigation de l'intrusion au Comité National Démocrate (DNC) en 2016 a illustré la valeur du hunting proactif. L'identification des groupes APT28 et APT29 n'a été possible que par une analyse manuelle approfondie dépassant les capacités des outils automatisés.

La corrélation `near` n'est pas supportée par tous les backends pySigma. Splunk la traduit en `transaction` ou en `join`, Elastic peut utiliser les EQL sequences, et Sentinel les fonctions KQL `arg_min/arg_max` avec des `join`. Si votre backend ne supporte pas `near`, vous devrez écrire la corrélation manuellement dans le langage natif du SIEM, ou utiliser deux règles distinctes combinées par un système d'enrichissement au niveau du SOAR.

4.3 Règles Sigma v2 : le format de corrélation étendu

La spécification Sigma v2 introduit un format de corrélation plus riche qui permet de référencer plusieurs règles Sigma distinctes et de définir des conditions de corrélation complexes entre elles. Ce format utilise le type `correlation` au lieu de `rule` :

```
# Corrélation Sigma v2 : chaîne d'attaque complète
title: Credential Dumping Attack Chain
type: correlation
id: 4d6e7af4-bg8e-6f4g-c2d9-8f0e1g7b3d4e
status: experimental
description: >
  Corrèle la reconnaissance (enum utilisateurs), l'accès LSASS
  et l'utilisation de credentials volés dans un mouvement latéral.

rules:
  - user_enum:
      id: "ref-user-enumeration-rule-uuid"
  - lsass_access:
      id: "ref-lsass-access-rule-uuid"
  - lateral_move:
      id: "ref-lateral-movement-rule-uuid"

group-by:
  - ComputerName

timespan: 30m
ordered: true                                  # L'ordre des événements compte

condition:
  gte: 2                                       # Au moins 2 des 3 événements
```

Ce format de corrélation est encore en phase d'adoption par les backends, mais il représente l'avenir de la détection comportementale multi-étapes dans Sigma. Il permet de modéliser des kill chains complètes et de générer des alertes de haute confiance basées sur la combinaison d'indicateurs faibles.

La persistance permet à un attaquant de maintenir son accès après un redémarrage. Les clés de registre Run/RunOnce sont le mécanisme de persistance le plus basique mais toujours efficace. Cette règle détecte les modifications suspectes, un vecteur souvent exploité après une compromission par phishing :

```
# Détection de persistance via Registry Run Keys
title: Suspicious Registry Run Key Modification
id: 9h238437-b5fg-87g7-e8f4-6e4i7i2j4869
status: stable
level: medium
tags:
  - attack.persistence
  - attack.t1547.001

logsource:
  category: registry_set
  product: windows

detection:
  selection_target:
    TargetObject|contains:
      - '\CurrentVersion\Run'
      - '\CurrentVersion\RunOnce'
      - '\CurrentVersion\RunServices'
      - '\CurrentVersion\Explorer\Shell Folders'
      - '\CurrentVersion\Explorer\User Shell Folders'
  selection_suspicious_value:
    Details|contains:
      - 'powershell'
      - 'cmd.exe /c'
      - 'mshta'
      - 'wscript'
      - 'cscript'
      - 'regsvr32'
      - 'rundll32'
      - 'C:\Users\Public'
      - 'C:\ProgramData'
      - '\AppData\Local\Temp'
      - 'http://'
      - 'https://'
  filter_installers:
    Image|endswith:
      - '\msiexec.exe'
      - '\setup.exe'
    Image|startswith: 'C:\Windows\Installer\'
  filter_updates:
    Image|contains:
      - '\Microsoft\EdgeUpdate\'
      - '\Google\Update\'
  condition: selection_target and selection_suspicious_value and not 1 of filter_*

falsepositives:
  - Legitimate software installation
  - Windows Update components
```

6.4 Détection d'exfiltration DNS (Data Exfiltration)

L'exfiltration de données via DNS est une technique furtive qui encode les données volées dans les requêtes DNS. Cette règle détecte les anomalies DNS caractéristiques de ce type d'exfiltration :

```
# Détection d'exfiltration via DNS tunneling
title: DNS Tunneling - Suspicious DNS Query Length
id: 0a349438-c6gh-98h8-f9g5-7f5j8j3k5970
status: experimental
level: high
tags:
  - attack.exfiltration
  - attack.t1048.003
  - attack.command_and_control
  - attack.t1071.004

logsource:
  category: dns_query
  product: windows

detection:
  selection_long_query:
    QueryName|re: '^[a-zA-Z0-9]{30,}\.'
  selection_high_entropy:
    QueryName|contains:
      - '.dnscat.'
      - '.tunnel.'
      - '.dns2tcp.'
  selection_subdomain_depth:
    QueryName|re: '([^.]+\.){5,}'
  filter_cdn:
    QueryName|endswith:
      - '.akamaiedge.net'
      - '.cloudfront.net'
      - '.googleapis.com'
      - '.windows.net'
      - '.azure.com'
  filter_internal:
    QueryName|endswith:
      - '.internal.corp'
      - '.ad.company.com'
  condition: (selection_long_query or selection_high_entropy or selection_subdomain_depth)
and not 1 of filter_*

falsepositives:
  - CDN services with long subdomains
  - Anti-virus cloud lookups
  - Certificate validation queries
```

6.5 Détection de Kerberoasting (Credential Access)

Le Kerberoasting est une technique qui exploite les tickets de service Kerberos pour extraire des hashes de mots de passe craquables hors ligne. Cette règle détecte les demandes de tickets TGS anormales :

```
# Détection de Kerberoasting
title: Kerberoasting - Suspicious TGS Request
id: 1b450549-d7hi-09i9-g0h6-8g6k9k4l6081
status: stable
level: high
tags:
  - attack.credential_access
  - attack.t1558.003

logsource:
  product: windows
  service: security

detection:
  selection:
    EventID: 4769
    TicketEncryptionType:
      - '0x17'     # RC4-HMAC (vulnerable, preferred by attackers)
      - '0x18'     # RC4-HMAC-EXP
    TicketOptions: '0x40810000'
  filter_machine_accounts:
    ServiceName|endswith: '$'
  filter_service_accounts:
    ServiceName|startswith: 'krbtgt'
  filter_normal_encryption:
    TicketEncryptionType:
      - '0x11'     # AES128
      - '0x12'     # AES256
  condition: selection and not 1 of filter_*

falsepositives:
  - Legacy applications using RC4 encryption
  - Service accounts with older SPNs
```

Conseil : combinez ces règles avec de l'enrichissement contextuel
Les règles Sigma fournissent la logique de détection, mais leur efficacité augmente considérablement lorsqu'elles sont enrichies par du contexte. Ajoutez des lookups pour identifier les comptes à privilèges, des listes de watchlist pour les assets critiques, et des baselines comportementales pour réduire les faux positifs. Pour une approche plus globale de l'écriture de règles de détection, consultez notre guide sur le Detection Engineering.

Le choix du `level` et du `status` impacte directement le traitement opérationnel de l'alerte :

Level	Usage	Action SOC
`informational`	Événements de contexte, pas d'action immédiate	Enrichissement, corrélation uniquement
`low`	Activité suspecte nécessitant validation	Revue en batch quotidien
`medium`	Activité probablement malveillante	Investigation dans les 4 heures
`high`	Forte probabilité d'incident de sécurité	Investigation immédiate, escalade L2
`critical`	Incident confirmé ou impact majeur	Réponse immédiate, activation du plan IR

Pour les statuts, suivez la progression naturelle : `experimental` (nouvelle règle en test) → `test` (validée en lab, déployée en shadow mode) → `stable` (production, ratio FP acceptable) → `deprecated` (obsolète, remplacée ou plus pertinente).

7.4 Erreurs fréquentes à éviter

- **Règle trop large** : une condition `selection` sans filtres génère un volume d'alertes ingérable. Commencez restrictif et élargissez progressivement
- **Dépendance aux chemins absolus** : `Image: 'C:\Users\admin\mimikatz.exe'` ne détecte qu'un seul scénario. Utilisez `Image|endswith` ou `OriginalFileName`
- **Oubli du case-insensitive** : Sigma est case-insensitive par défaut pour les valeurs, mais attention aux regex personnalisées
- **Filtre qui crée un blind spot** : exclure `powershell.exe` pour réduire le bruit supprime aussi la détection d'attaques PowerShell légitimes
- **Absence de tests** : une règle non testée est une règle non fiable. Validez avec des données TP et TN avant le déploiement
- **ID dupliqué** : chaque règle doit avoir un UUID unique. Les collisions causent des problèmes de déploiement

8. Intégration dans un pipeline CI/CD

8.1 Validation automatique des règles

L'intégration de Sigma dans un pipeline CI/CD garantit la qualité et la cohérence des règles. Voici un workflow GitHub Actions complet pour valider, compiler et déployer des règles Sigma :

```yaml
# .github/workflows/sigma-pipeline.yml
name: Sigma Rules Validation & Deployment

on:
  pull_request:
    paths: ['rules/sigma/**']
  push:
    branches: [main]
    paths: ['rules/sigma/**']

jobs:
  validate:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Setup Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.12'

      - name: Install dependencies
        run: |
          pip install pySigma pySigma-backend-splunk \
                      pySigma-backend-elasticsearch \
                      pySigma-pipeline-sysmon \
                      pySigma-pipeline-windows \
                      yamllint pytest

      # Étape 1 : Lint YAML
      - name: YAML Lint
        run: yamllint -c .yamllint.yml rules/sigma/

      # Étape 2 : Validation Sigma (syntaxe + champs)
      - name: Validate Sigma rules
        run: sigma check rules/sigma/

      # Étape 3 : Vérifier les ID uniques
      - name: Check unique IDs
        run: |
          python -c "
          import yaml, glob, sys
          ids = {}
          for f in glob.glob('rules/sigma/**/*.yml', recursive=True):
              with open(f) as fh:
                  rule = yaml.safe_load(fh)
                  rid = rule.get('id', 'MISSING')
                  if rid in ids:
                      print(f'DUPLICATE ID {rid}: {ids[rid]} and {f}')
                      sys.exit(1)
                  ids[rid] = f
          print(f'All {len(ids)} rule IDs are unique.')
          "

      # Étape 4 : Compilation multi-SIEM
      - name: Compile to Splunk
        run: |
          sigma convert -t splunk -p sysmon \
            rules/sigma/ -o compiled/splunk/

      - name: Compile to Elastic
        run: |
```

```
          sigma convert -t elasticsearch -p ecs_windows \
            rules/sigma/ -o compiled/elastic/

      # Étape 5 : Tests
      - name: Run tests
        run: pytest tests/ -v

  deploy:
    needs: validate
    if: github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Deploy compiled rules
        run: python scripts/deploy.py --target all
        env:
          SPLUNK_TOKEN: ${{ secrets.SPLUNK_HEC_TOKEN }}
          ELASTIC_API_KEY: ${{ secrets.ELASTIC_API_KEY }}
```

8.2 Sigma CLI : commandes essentielles

Le CLI `sigma` (fourni par pySigma) offre plusieurs commandes utiles pour le travail quotidien avec les règles :

```
# Valider une règle
sigma check rules/sigma/credential_access/mimikatz_execution.yml

# Convertir une règle pour Splunk avec le pipeline Sysmon
sigma convert -t splunk -p sysmon rules/sigma/credential_access/mimikatz_execution.yml

# Convertir pour Elastic avec le pipeline ECS Windows
sigma convert -t elasticsearch -p ecs_windows rules/sigma/credential_access/
mimikatz_execution.yml

# Convertir pour Microsoft Sentinel (KQL)
sigma convert -t microsoft365defender rules/sigma/credential_access/mimikatz_execution.yml

# Convertir toutes les règles d'un répertoire
sigma convert -t splunk -p sysmon rules/sigma/ -o compiled/splunk/

# Lister les backends disponibles
sigma list backends

# Lister les pipelines disponibles
sigma list pipelines
```

Pour une approche complète de l'automatisation des règles de détection, consultez notre article sur le pipeline CI/CD Detection-as-Code.

Pour approfondir ce sujet, consultez notre outil open-source soar-playbooks qui facilite l'automatisation des playbooks de réponse.

Questions frequentes

Comment mettre en place Sigma Rules dans un environnement de production ?

La mise en place de Sigma Rules en production necessite une planification rigoureuse, incluant l'evaluation des prerequis techniques, la definition d'une architecture cible, des tests de validation approfondis et un plan de deploiement progressif avec des points de controle a chaque etape.

Pourquoi Sigma Rules est-il essentiel pour la securite des systemes d'information ?

Sigma Rules constitue un element fondamental de la securite des systemes d'information car il permet de reduire significativement la surface d'attaque, d'ameliorer la detection des menaces et de renforcer la posture globale de securite de l'organisation face aux cybermenaces actuelles.

Quelles sont les bonnes pratiques pour Sigma Rules en 2026 ?

Les bonnes pratiques pour Sigma Rules en 2026 incluent l'adoption d'une approche Zero Trust, l'automatisation des controles de securite, la mise en place d'une veille continue sur les vulnerabilites et l'integration des recommandations des organismes de reference comme l'ANSSI et le NIST.

Sources et références : MITRE ATT&CK · MITRE CAR

> Points clés à retenir
>
> - 8. Intégration dans un pipeline CI/CD
> - Questions frequentes
> - 9. Conclusion et ressources

9. Conclusion et ressources

Sigma s'est imposé comme le **standard de facto** pour l'écriture de règles de détection portables. Sa syntaxe YAML déclarative, ses modifiers puissants et l'écosystème pySigma permettent d'écrire une règle une seule fois et de la déployer sur n'importe quel SIEM. Les points clés à retenir :

- Maîtrisez la **logsource** : c'est l'abstraction qui rend vos règles portables
- Utilisez les **modifiers** (`|contains` , `|endswith` , `|re`) pour des détections précises
- Structurez vos détections avec le pattern **selection + filter + condition**
- Explorez les **corrélations** et **agrégations** pour détecter des comportements complexes
- Intégrez **pySigma** dans votre pipeline CI/CD pour automatiser la compilation et le déploiement

- Contribuez à **SigmaHQ** pour enrichir la base de règles communautaire

Pour aller plus loin

L'écriture de règles Sigma est une compétence fondamentale du Detection Engineering. Pour une approche complète, combinez Sigma avec le threat hunting pour transformer vos hypothèses de chasse en règles de détection automatisées. Le déploiement open source avec Wazuh offre une intégration native de Sigma via le mapping des règles.

Articles connexes

SOC & Détection
Detection Engineering : Construire des Règles Efficaces
Pyramide de la douleur, lifecycle, métriques, CI/CD
DevSecOps
Detection-as-Code : Pipeline CI/CD pour SIEM
Automatisation, tests, déploiement continu des règles
Threat Hunting
Threat Hunting : Méthodologie, Outils et Pratique
Hunting hypothesis, data analysis, tools et workflows
SIEM Open Source
Wazuh SIEM/XDR : Déploiement et Configuration
Installation, règles custom, intégration Sigma
Framework
MITRE ATT&CK : Guide Pratique Red & Blue Team
Tactiques, techniques, couverture de détection
Attaques Kerberos
Kerberoasting & AS-REP Roasting : Attaques Kerberos
Techniques d'extraction de credentials et détection

Références et ressources externes

- SigmaHQ -- Sigma Rules Repository -- Dépôt officiel avec plus de 3000 règles communautaires
- pySigma -- Sigma Rule Processing Framework -- Framework de compilation multi-SIEM (Splunk, Elastic, Sentinel)
- Sigma Documentation officielle -- Guide de démarrage et spécification du format
- MITRE ATT&CK Framework -- Base de connaissances pour le tagging des règles
- Atomic Red Team -- Red Canary -- Tests de validation par technique ATT&CK

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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