AYI NEDJIMI Consultants

Detection Engineering : Construire des Règles de : Guide

Catégorie : SOC et Detection Lecture : 7 min Publié le : 08/03/2026 Auteur : Ayi NEDJIMI

Guide complet Detection Engineering : pyramide de la douleur, MITRE ATT&CK mapping, lifecycle des règles SIEM, types de détection, testing, métriques.

```
# Exemple de détection par anomalie dans Splunk SPL
# Détection de connexions RDP inhabituelles (baseline sur 30 jours)
index=windows sourcetype=WinEventLog:Security EventCode=4624 Logon_Type=10
| stats count as daily_rdp_count by src_ip, dest, _time
| eventstats avg(daily_rdp_count) as avg_count, stdev(daily_rdp_count) as stdev_count by
dest
| where daily_rdp_count > (avg_count + 3 * stdev_count)
| table _time, src_ip, dest, daily_rdp_count, avg_count
```

5.3 Détection comportementale (Behavioral)

La détection comportementale est le sweet spot du detection engineering. Elle se situe au sommet de la pyramide de la douleur en détectant les **TTPs** plutôt que les indicateurs éphémères. Elle modélise le *comportement* de l'attaquant : "un processus non système accède à la mémoire de LSASS", "un script PowerShell télécharge et exécute du code en mémoire", "un compte de service se connecte depuis un poste utilisateur". Guide complet Detection Engineering : pyramide de la douleur, MITRE ATT&CK mapping, lifecycle des règles SIEM, types de détection, testing, métriques. Ce guide couvre les aspects essentiels de detection engineering regles efficaces : méthodologie structurée, outils recommandés et retours d'expérience opérationnels. Les professionnels y trouveront des recommandations directement applicables.

Avantages : résistante au changement d'outils de l'attaquant, se concentre sur l'intention, longue durée de vie.

Limites : plus complexe à écrire, nécessite une bonne compréhension du contexte normal, peut nécessiter une corrélation de plusieurs événements. La corrélation de logs est un sujet que nous approfondissons dans notre article sur l'audit avancé Microsoft 365.

```
# Exemple de détection comportementale dans Elastic KQL
# Détection : processus non standard accédant à LSASS memory
# MITRE ATT&CK: T1003.001 - OS Credential Dumping: LSASS Memory

process where event.action == "access" and
  process.Ext.target.name == "lsass.exe" and
  not process.executable : (
    "C:\\Windows\\System32\\*.exe",
    "C:\\Windows\\SysWOW64\\*.exe",
    "C:\\Program Files\\*\\*.exe",
    "C:\\Program Files (x86)\\*\\*.exe"
  ) and
  process.Ext.call_trace_summary : "*UNKNOWN*"
```

Critère	Signature	Anomalie	Comportementale
Menaces zero-day	Non	Oui	Oui
Faux positifs	Très bas	Élevé	Moyen
Complexité	Faible	Élevée	Élevée
Durée de vie	Courte	Variable	Longue
Pyramide de la douleur	Base (Hash/IP)	Milieu	Sommet (TTPs)
Recommandation	Complémentaire	UBA/UEBA	Prioritaire

```
# Détection avancée : Lateral Movement via PsExec/Remote Service
# Corrélation entre la création de service (Event 7045) et l'exécution (Event 4688)
# MITRE ATT&CK: T1021.002 (SMB/Windows Admin Shares) + T1569.002 (Service Execution)

index=windows sourcetype=WinEventLog:System EventCode=7045
  Service_File_Name="*PSEXESVC*" OR Service_File_Name="*\\cmd.exe*" OR
Service_File_Name="*\\powershell*"
| rename Computer as dest_host
| join type=inner dest_host
  [search index=windows sourcetype=WinEventLog:Security EventCode=4624 Logon_Type=3
   | rename Computer as dest_host
   | where src_ip != "127.0.0.1" AND src_ip != "::1"
   | stats earliest(_time) as logon_time, values(src_ip) as src_ip by dest_host,
Account_Name]
| eval time_diff = _time - logon_time
| where time_diff > 0 AND time_diff < 300
| table _time, dest_host, Account_Name, src_ip, Service_File_Name, Service_Name, time_diff
| sort -_time
```

6.4 Règle Elastic KQL : détection de Living-off-the-Land

Les attaques **Living-off-the-Land (LOLBins)** utilisent des binaires légitimes du système pour exécuter des actions malveillantes. Ces techniques sont particulièrement difficiles à détecter car les outils utilisés sont légitimes. Voici une détection de l'utilisation suspecte de `certutil.exe` pour télécharger des fichiers :

```
# Elastic Security Rule (KQL)
# Détection : certutil.exe utilisé pour télécharger un fichier
# MITRE ATT&CK: T1105 (Ingress Tool Transfer) + T1140 (Deobfuscate/Decode Files)

process where event.type == "start" and
  process.name : "certutil.exe" and
  process.args : ("-urlcache", "-split", "-decode", "-encode", "-decodehex") and
  not process.parent.executable : (
    "C:\\Windows\\System32\\svchost.exe",
    "C:\\Windows\\System32\\mmc.exe"
  )

# Version ESQL (Elasticsearch Query Language)
FROM logs-endpoint.events.process-*
| WHERE process.name == "certutil.exe"
  AND (process.args LIKE "*-urlcache*" OR process.args LIKE "*-decode*")
| STATS count = COUNT(*) BY host.name, user.name, process.command_line
| WHERE count > 0
| SORT count DESC
```

6.5 Bonnes pratiques d'écriture de règles

L'écriture de règles de détection de qualité repose sur plusieurs principes fondamentaux :

- **Documenter l'intention** : chaque règle doit expliquer *pourquoi* elle existe, pas seulement *ce qu'elle* fait. Le titre et la description doivent permettre à un analyste SOC de comprendre l'alerte en 5 secondes
- **Minimiser les faux positifs dès la conception** : inclure des filtres d'exclusion pour les processus légitimes connus. Documenter les faux positifs attendus
- **Mapper sur ATT&CK** : chaque règle doit référencer au moins une technique ATT&CK. C'est non négociable
- **Fournir un runbook** : chaque règle doit être accompagnée d'un runbook qui guide l'analyste SOC dans l'investigation. Questions à poser, logs complémentaires à consulter, actions de remédiation
- **Tester avant le déploiement** : jamais de règle en production sans test. Utiliser Atomic Red Team ou un dataset de logs pour valider
- **Limiter la complexité** : une règle trop complexe est difficile à maintenir et à débugger. Préférer plusieurs règles simples corrélées par le SOAR

Template de règle standardisé
Adoptez un template standardisé pour toutes vos règles. Chaque règle doit contenir : **ID unique**, **nom descriptif**, **description**, **auteur**, **date de création/modification**, **sévérité**, **mapping ATT&CK**, **sources de données requises**, **logique de détection**, **faux positifs connus**, **runbook SOC**, et **statut** (draft/review/testing/production/deprecated). Ce template garantit la cohérence et facilite l'onboarding de nouveaux detection engineers.

Pour des scénarios de test plus complexes impliquant des chaînes d'attaque multi-étapes, **MITRE Caldera** est un framework de simulation d'adversaire qui exécute automatiquement des séquences de techniques ATT&CK. Il permet de tester la capacité du SOC à détecter non pas une technique isolée, mais un **parcours d'attaque complet**.

D'autres outils de validation sont également précieux :

- **Sigma Test Utility (sigma-test)** : teste les règles Sigma contre des jeux de données JSON, vérifie que la logique de détection est correcte sans nécessiter un SIEM
- **DetectionLab** : un environnement de lab automatisé avec Windows AD, Sysmon, Splunk pré-configuré pour le testing des détections
- **Vectr** : plateforme de Purple Team qui suit les résultats de test et le coverage ATT&CK au fil du temps
- **Log replay** : rejouer des logs d'incidents passés (anonymisés) pour vérifier que les nouvelles règles auraient détecté l'attaque

Testing en production vs environnement de test
Les tests de True Positive (simulation d'attaque) ne doivent **jamais** être exécutés directement en production sans coordination avec les équipes SOC et IT. Utilisez un environnement de test dédié (lab) pour les simulations d'attaque, ou coordonnez un exercice Purple Team avec notification préalable. Un test non coordonné peut déclencher une réponse à incident réelle et gaspiller des heures de travail des analystes. C'est également une source potentielle de perturbation si le test atomique modifie des configurations système.

Figure 3 -- Pipeline Detection Engineering avec métriques et boucle de feedback

8.3 Dashboards de suivi

Le Detection Engineering doit disposer de dashboards dédiés pour piloter la performance du programme. Les tableaux de bord essentiels sont :

- **Rule Health Dashboard** : état de chaque règle (production/testing/deprecated), nombre de TP/FP sur les 30 derniers jours, dernière date de test, dernier tuning
- **ATT&CK Coverage Heatmap** : visualisation de la matrice ATT&CK avec un code couleur par niveau de couverture (rouge = non couvert, jaune = partiel, vert = couvert et testé)
- **SOC Performance** : MTTD, MTTR, volume d'alertes par catégorie, taux de faux positifs global, tendances sur 90 jours
- **Detection Backlog** : nombre de règles en développement, en attente de review, en testing. Velocity de production de nouvelles détections

Astuce : KPI minimal pour démarrer
Si vous ne devez suivre que trois métriques au début, choisissez : (1) le **taux de faux positifs par règle** pour identifier les règles bruyantes, (2) la **couverture ATT&CK** (nombre de techniques couvertes / total pertinent) pour guider les priorités, et (3) le **MTTD** pour mesurer l'impact réel sur la sécurité. Ajoutez les autres métriques progressivement.

Un pipeline CI/CD de détection se compose de plusieurs étapes exécutées automatiquement à chaque pull request :

```yaml
# .github/workflows/detection-pipeline.yml
name: Detection Rules Pipeline

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

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

      # Étape 1 : Validation syntaxique
      - name: Validate YAML syntax
        run: |
          pip install yamllint
          yamllint -c .yamllint.yml rules/

      # Étape 2 : Validation Sigma
      - name: Validate Sigma rules
        run: |
          pip install pySigma
          sigma check rules/sigma/

      # Étape 3 : Compilation multi-SIEM
      - name: Compile to target SIEMs
        run: |
          sigma convert -t splunk -p sysmon rules/sigma/ -o compiled/splunk/
          sigma convert -t elastic -p ecs_windows rules/sigma/ -o compiled/elastic/

      # Étape 4 : Tests unitaires
      - name: Run detection tests
        run: pytest tests/ -v --tb=short

      # Étape 5 : Analyse de couverture
      - name: ATT&CK coverage report
        run: python scripts/coverage_report.py --output coverage.json

  deploy:
    needs: validate
    if: github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    steps:
      # Déploiement automatique vers le SIEM
      - name: Deploy to Splunk
        run: python scripts/deploy_splunk.py --env production
        env:
          SPLUNK_TOKEN: ${{ secrets.SPLUNK_TOKEN }}

      - name: Deploy to Elastic
        run: python scripts/deploy_elastic.py --env production
        env:
          ELASTIC_API_KEY: ${{ secrets.ELASTIC_API_KEY }}
```

9.4 Tests automatisés pour les règles de détection

Chaque règle doit être accompagnée de tests qui valident son comportement attendu. Les tests vérifient deux aspects : les **true positives** (la règle détecte bien ce qu'elle doit détecter) et les **true negatives** (la règle ne déclenche pas sur l'activité légitime) :

```
# tests/test_mimikatz_detection.py
import pytest
from sigma.rule import SigmaRule
from sigma.backends.splunk import SplunkBackend

class TestMimikatzDetection:
    @pytest.fixture
    def rule(self):
        return SigmaRule.from_yaml(open('rules/sigma/credential_access/
mimikatz_execution.yml'))

    def test_rule_compiles_splunk(self, rule):
        """Vérifie que la règle se compile correctement pour Splunk"""
        backend = SplunkBackend()
        result = backend.convert_rule(rule)
        assert len(result) > 0
        assert 'mimikatz' in result[0].lower() or 'sekurlsa' in result[0].lower()

    def test_true_positive_process_name(self, rule):
        """Vérifie la détection par nom de processus"""
        test_event = {
            'Image': 'C:\\Users\\attacker\\mimikatz.exe',
            'CommandLine': 'mimikatz.exe "privilege::debug" "sekurlsa::logonpasswords"',
            'User': 'CORP\\admin'
        }
        assert rule.matches(test_event)

    def test_true_negative_legitimate(self, rule):
        """Vérifie l'absence de faux positif sur activité légitime"""
        test_event = {
            'Image': 'C:\\Windows\\System32\\lsass.exe',
            'CommandLine': 'lsass.exe',
            'User': 'NT AUTHORITY\\SYSTEM'
        }
        assert not rule.matches(test_event)
```

Conseil : intégrez Atomic Red Team dans votre pipeline
Les tests unitaires avec des événements simulés sont un bon début, mais ils ne remplacent pas les tests en conditions réelles. Intégrez **Atomic Red Team** dans un environnement de lab pour exécuter automatiquement les techniques ATT&CK correspondant à vos règles et vérifier que les alertes se déclenchent effectivement dans votre SIEM. Cela valide l'ensemble de la chaîne : collecte de logs, parsing, et détection.

10. Checklist Detection Engineering

Cette checklist synthétise les bonnes pratiques présentées dans cet article. Utilisez-la comme référence pour chaque nouvelle règle de détection et comme guide d'audit pour votre programme existant :

10.1 Avant d'écrire une règle

- Identifier la technique ATT&CK ciblée et son data source
- Vérifier que les logs nécessaires sont effectivement collectés dans le SIEM
- Rechercher les règles existantes (SigmaHQ, Elastic Detection Rules) avant de créer une règle from scratch
- Définir l'hypothèse de détection : quel comportement attaquant spécifique vise-t-on ?
- Évaluer le niveau dans la Pyramide de la Douleur : privilégier les TTP aux IoC
- Estimer le volume d'alertes attendu en requêtant les logs historiques

10.2 Pendant l'écriture

- Utiliser le format Sigma pour garantir la portabilité multi-SIEM. Voir notre guide d'écriture Sigma
- Nommer la règle de manière descriptive (verbe + objet + contexte)
- Documenter les champs `description`, `falsepositives`, `level`, `tags`
- Ajouter des filtres d'exclusion pour les processus légitimes connus
- Préférer la détection comportementale (TTP) à la détection par signature (hash, IP)
- Tester la règle sur un jeu de données contenant des TP et des TN

10.3 Avant le déploiement

- Faire une revue par un pair (pull request avec commentaires)
- Exécuter le pipeline CI/CD : validation syntaxique, compilation, tests
- Déployer en mode **shadow/silent** pendant 1-2 semaines pour mesurer le bruit
- Vérifier que la règle ne génère pas plus de 50 alertes/jour (seuil ajustable selon le contexte)
- Valider avec un test Atomic Red Team que la technique est effectivement détectée
- Documenter le runbook d'investigation pour les analystes SOC

10.4 Après le déploiement

- Surveiller le ratio TP/FP pendant les 30 premiers jours
- Itérer sur les filtres d'exclusion en fonction des retours analystes
- Mettre à jour la couverture ATT&CK dans le dashboard
- Planifier des re-tests trimestriels pour vérifier que la règle fonctionne toujours
- Intégrer les retours d'incidents (retex) pour enrichir la détection
- Déprécier les règles obsolètes avec une justification documentée

Conclusion : le Detection Engineering est un programme, pas un projet

Le Detection Engineering n'est pas une initiative ponctuelle mais un **programme continu** qui s'améliore itérativement. Commencez par les techniques ATT&CK les plus critiques pour votre environnement, mettez en place un pipeline minimal (Git + CI), et progressez vers une couverture exhaustive. La clé du succès réside dans la **boucle de feedback** entre les analystes

SOC et les detection engineers : chaque incident, chaque faux positif et chaque technique manquée est une opportunité d'amélioration. Pour aller plus loin, explorez notre article sur le threat hunting qui complète la détection automatisée par une recherche proactive de menaces.

Articles connexes

SOC & Détection
Sigma Rules : Guide Complet d'Écriture et Déploiement
Format YAML, modifiers, corrélation, pySigma backends
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
Framework
MITRE ATT&CK : Guide Pratique Red & Blue Team
Tactiques, techniques, couverture de détection
SIEM Open Source
Wazuh SIEM/XDR : Déploiement et Configuration
Installation, règles custom, intégration Sigma
Techniques d'Attaque
Mouvement Latéral : PsExec, WMI, WinRM, DCOM
Techniques de propagation et détection associée

Références et ressources externes

- SigmaHQ -- Sigma Rules Repository -- Dépôt officiel des règles Sigma communautaires
- MITRE ATT&CK Framework -- Base de connaissances des tactiques et techniques adverses
- Atomic Red Team -- Red Canary -- Framework de tests de détection par technique ATT&CK
- detect.fyi -- Detection Engineering Resources -- Curation de ressources Detection Engineering
- MITRE Caldera -- Plateforme de simulation d'adversaire automatisée

Pour approfondir ce sujet, consultez notre outil open-source siem-correlation-rules qui facilite l'optimisation des règles de corrélation SIEM.

Questions frequentes

Comment mettre en place Detection Engineering dans un environnement de production ?

La mise en place de Detection Engineering 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 Detection Engineering est-il essentiel pour la securite des systemes d'information ?

Detection Engineering 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.

Combien de règles de détection faut-il pour démarrer avec Detection Engineering : Construire des Règles ?

Commencez par 20 à 30 règles alignées sur les techniques MITRE ATT&CK les plus courantes. Mieux vaut peu de règles bien calibrées que des centaines qui génèrent du bruit.

Sources et références : MITRE ATT&CK · MITRE CAR

> Points clés à retenir
>
> - 10. Checklist Detection Engineering
> - Questions frequentes
> - Conclusion

Conclusion

Cet article a couvert les aspects essentiels de 1. Introduction : du SOC réactif au SOC engineering, 2. La pyramide de la douleur, 3. MITRE ATT&CK : le framework de référence. La mise en oeuvre de ces recommandations permet de renforcer significativement votre posture de securite et de repondre aux exigences des referentiels en vigueur.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

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