FAIBLESSES DES MÉCANISMES D'AUTHENTIFICATION DE WINDOWS : QUELLES SOLUTIONS ?

Marc Lebrun - marc.lebrun@xmco.fr ; marc.lebrun.mailbox@gmail.com - @MarcLebrun - Consultant chez XMCO

mots-clés : WINDOWS / ADMINISTRATION / PASS-THE-HASH / NTLM / ACCESS TOKEN / AUTHENTICATION PACKAGES

La récupération de hashes et la réalisation d'attaques Pass-The-Hash font partie des techniques les plus fréquemment mises en œuvre lors de tests d'intrusion en environnement Windows. 15 ans après la première publication sur l'attaque Pass-The-Hash, celle-ci est le B.A.-BA du pentester et une pléthore d'outils est à la disposition des attaquants. La pratique montre d'ailleurs que cette technique n'a rien perdu de son efficacité... Mais alors, comment s'en prémunir ? Quelles recommandations apporter une fois que l'on a démontré au client que son SI est vulnérable ?

1 Introduction

La récupération des *hashes* est en effet un incontournable du test d'intrusion interne. Pourquoi se priver de cette technique qui facilite tellement le travail de l'auditeur, lui permettant d'évoluer sans grande difficulté sur le SI, pour aboutir à l'inévitable prise de contrôle du domaine ?

L'exercice le plus difficile est surtout d'émettre des recommandations à la fois réalistes et efficaces. C'est le point de départ de cette modeste étude, qui s'est principalement focalisée sur l'obtention des hashes de comptes de domaine, notamment en mémoire. Nous avons ensuite souhaité compléter cette expérimentation par un petit *benchmark* des outils les plus efficaces et polyvalents :-)

1.1 « Pass pass le hash »

Afin d'éviter que les mots de passe des utilisateurs ne puissent être dérobés, Windows les stocke sous la forme de « hashes », condensats cryptographiques fournissant une empreinte ne pouvant être utilisée pour retrouver le texte original sans une puissance de calcul phénoménale (attaque dite « en force brute »).

Ces mécanismes cryptographiques développés par Microsoft ont été mis à mal au fil du temps, exposant des failles permettant de les « casser » rapidement afin de retrouver le mot de passe original **[WEAKLM]**. Ainsi, ces hashes LM (*LanManager*) et NTLM (NT *LanManager*) sont devenus une cible privilégiée pour les attaquants. Cependant, un autre usage que la cryptanalyse peut être fait de ces données. En effet, en souhaitant faciliter l'utilisation des différentes fonctionnalités réseau de Windows (partage de fichiers, d'imprimantes, etc.) et fournir une expérience utilisateur plus fluide, les équipes de Microsoft ont donc basé leur *Single Sign-On* sur un mécanisme transparent ne nécessitant qu'un minimum d'interaction avec l'utilisateur. C'est pourquoi les phases d'authentification reposent sur des défis/réponses qu'il est possible de résoudre même si l'on ne dispose que du hash, qui est stocké en mémoire par Windows. En conséquence, tous les hashes LM/NTLM obtenus sur un système Windows peuvent alors être chargés dans la session en cours afin de les « passer » lors des phases d'authentification réseau et ainsi usurper l'identité d'un autre utilisateur dont le hash a été dérobé.

Les hashes des comptes locaux, présents au sein du fichier SAM, ne seront abordés que brièvement dans cet article. Nous avons souhaité en effet nous concentrer sur la récupération de données exploitables sur le réseau. Bien sûr, la réutilisation de mots de passe de comptes locaux sur d'autres machines est un vecteur d'attaque valide, mais les hashes et mots de passe de domaine constituent le précieux « sésame » tant recherché.

1.1.1 « Papa, c'est quoi ce hash ? »

Plusieurs bibliothèques intégrées à Windows, les *Authentication Packages*, implémentent les différents mécanismes permettant de s'authentifier sur un système ou une ressource distante (MSV1_0, Wdigest, Kerberos, TsPkg, etc.).

A priori, seul le *package* MSV1_0, introduit avec Windows NT4, est basé sur un mécanisme défi/réponse impliquant les hashes LM et NTLM. Cependant, d'autres packages, notamment WDigest, stockent ces hashes en mémoire afin de recalculer les réponses aux défis en fonction du domaine sur lequel on souhaite se connecter. Mais si seulement il n'y avait que des hashes en mémoire...

Ces Authentication Packages reçoivent en entrée le mot de passe en clair, libre à eux ensuite d'en faire ce que bon leur semble (calcul de hash, de défis cryptographiques, négociation de tickets ou autre). Ainsi, les packages Wdigest, Kerberos et TsPkg conservent ce mot de passe sous une forme chiffrée mais réversible. Au moins deux outils publics (WCE **[WCE]** et Mimikatz **[MIMI]**) tirent parti de cette faiblesse et sont capables d'extraire ces mots de passe de la mémoire...

Il faut également citer une dernière forme qui peut être exploitée par les attaquants : les hashes MS-Cache et MS-Cache V2. Ce sont en fait des condensats cryptographiques de la forme `MD4(MD4(Unicode(Mot de passe)) + Unicode(Lowercase(Utilisateur)))` permettant le stockage des données de connexion en cache dans le cas où le contrôleur de domaine ne serait pas joignable lors de la procédure d'authentification d'un utilisateur. Ils sont cependant d'un moindre intérêt pour nous car non utilisables en l'état afin de réaliser la fameuse attaque « Pass-The-Hash ». Ils doivent d'abord être cassés grâce à une attaque de *bruteforce* ou via l'utilisation de *rainbow-tables*. Il faut d'ailleurs noter que puisque le nom de l'utilisateur est utilisé comme diversifiant, ces rainbow-tables sont spécifiques au nom d'utilisateur en question (Administrateur par exemple), ce qui fait de ces hashes une cible de dernier recours.

1.1.2 « All your tokens are belong to us »

On peut également noter que certains outils sont capables d'extraire et de réutiliser des *Access Tokens*. Ces objets décrivent le contexte de sécurité associé à un processus et certains peuvent être exploités afin d'effectuer des actions dans un contexte différent de celui de l'utilisateur courant.

Deux types de *tokens* existent **[MSTOK]** :

- Les *Primary Tokens*, qui décrivent le contexte de sécurité du processus et ne sont pas exploitables ;
- Les *Impersonation Tokens*, qui autorisent un *thread* à utiliser un contexte de sécurité différent de celui du processus courant. Ils implémentent quatre niveaux d'*impersonation* :
  - → *Anonymous* : non utilisé ;
  - → *Identify* : permet d'identifier l'utilisateur, mais pas d'effectuer des actions en son nom ;
  - → *Impersonate* : permet d'effectuer des actions locales dans le contexte de l'utilisateur ;
  - → *Delegate* : permet d'effectuer des actions distantes dans le contexte de l'utilisateur.

Bien que l'exploitation de ces *Delegation Tokens* reste marginale, la suite de cet article démontrera leur utilité dans les situations où les hashes ne sont pas disponibles.

1.2 Objectifs de l'étude

De nombreuses données d'authentification sont donc présentes, soit sur le disque (registres, `NTDS.DIT`), soit en mémoire, et peuvent être extraites afin de tenter de s'authentifier sur d'autres machines du SI. La réutilisation de ces hashes, tokens ou mots de passe permet d'élever facilement ses privilèges, à la fois latéralement en évoluant de machine en machine, mais également verticalement en extrayant des identifiants appartenant à un compte disposant de privilèges plus élevés sur le domaine.

Tous les consultants ayant eu l'occasion de réaliser ce genre de prestation savent comment elles se terminent en général : sur un contrôleur de domaine, à récupérer les hashes de tous les utilisateurs contenus dans `NTDS.DIT`...

De ce constat, plusieurs problématiques se dégagent. Tout d'abord, du point de vue de la victime, que faire pour se protéger ? La réponse de Microsoft face à la présence de hashes ou d'identifiants de connexion en mémoire peut se réduire à : « il faut mettre en place des processus de défense en profondeur ». Mais n'est-il pas également possible de limiter les risques en utilisant des protocoles d'administration ne laissant pas ou peu de données sensibles en mémoire ?

De plus, du point de vue de l'attaquant, certains outils ne sont pas compatibles avec toutes les versions de Windows ou ne supportent pas les systèmes 64 bits, quand ils ne sont pas carrément instables (une injection LSASS qui dérape et c'est la plateforme de production qui tombe...).

L'objectif de cette étude est donc triple :

- Identifier des méthodes d'administration qui ne laissent pas de traces exploitables ;
- Effectuer un *benchmark* des principaux outils existant afin d'identifier les plus efficaces ;
- Et enfin, en fonction des résultats obtenus, émettre des recommandations techniques concrètes, qui sortent des grands classiques, du genre « installez des antivirus, les outils de *hack* seront bloqués et tout ira bien »...

2 Méthodologie

Tout d'abord, nous aurions pu nous limiter à l'étude académique de la documentation existante sur les différentes méthodes d'administration à distance, accompagnée d'une analyse statique du code source des outils (quand il est disponible, et complet...). Mais une approche empirique semblait pouvoir fournir des résultats plus complets et plus fiables. Et puis, quand on pratique la « sécurité offensive », on ne se refait pas ;-)

2.1 Définition du périmètre des tests

2.1.1 Les tests

Première étape : inventorier les outils d'administration à distance pris en charge par Windows et susceptibles de permettre l'exécution de tâches de maintenance courantes.

Voici la liste des méthodes retenues :

- authentification locale via la mire de Windows ;
- commande `RunAs` ;
- bureau à distance (*Terminal Services*) ;
- ligne de commandes WMI (WMIC) ;
- Psexec de la suite Sysinternals ;
- Telnet ;
- montage de partage distant via la commande `NET USE` ;
- Microsoft Management Console (*MMC Snap-ins*)
- édition du registre à distance (Regedit) ;
- authentification *Active Directory* sur Microsoft IIS ;
- création de tâches planifiées.

Cette liste couvre la plupart des cas d'usage rencontrés par les administrateurs en environnement Windows / Active Directory. L'inclusion de solutions tierces (comme VNC par exemple) a été considérée dans un premier temps avant d'être écartée, car en général soit elles fournissent leur propre méthode d'authentification ne dépendant pas des identifiants Active Directory, soit elles reposent sur les mécanismes fournis par Windows, recoupant les méthodes sélectionnées (utilisation des RPC, *Interactive logon*, etc.).

Programme	Procédure mise en œuvre lors des tests	Données récupérables
gsecdump	`gsecdump.exe -a`	Hashes LM/NTLM de comptes locaux Hashes LM/NTLM en mémoire
pwdump7	`pwdump7.exe`	Hashes LM/NTLM de comptes locaux
fgdump	`fgdump.exe -s -r -v -v -k -T 3 -O [32|64]`	Hashes LM/NTLM de comptes locaux Hashes MS-CACHE Hashes LM/NTLM extraits du fichier `NTDS.DIT`

Mimikatz	`privilege::debug`
	`sekurlsa::logonPasswords Full`
	`divers::secrets Full`
	(voir note sur les tâches planifiées)
		Hashes LM/NTLM de comptes locaux
		Hashes LM/NTLM en mémoire
		Mots de passe en mémoire
		Certificats et clés en mémoire

Meterpreter	hashdump cachedump	Hashes LM/NTLM de comptes locaux Hashes MS-CACHE
PWDumpX	`pwdumpx.exe -clph 127.0.0.1 + +`	Hashes LM/NTLM de comptes locaux Hashes MS-CACHE
WCE	`wce.exe -w` `wce.exe -l -v`	Hashes LM/NTLM en mémoire Mots de passe en mémoire
QuarksPwDump	`quarkspwdump.exe -dhl` `quarkspwdump.exe -dhdc`	Hashes LM/NTLM de comptes locaux Hashes MS-CACHE Hashes LM/NTLM extraits du fichier NTDS.DIT
cachedump	`cachedump.exe -v`	Hashes MS-CACHE
incognito	`incognito.exe -h 127.0.0.1 list_tokens -u`	Access Tokens

Liste des outils utilisés

RUNAS

Lors des tests, la commande RunAs a été utilisée sans l'argument /savecred, qui permet de sauvegarder un mot de passe pour les commandes ultérieures. L'utilisation de cette option est en effet considérée comme non sûre, puisque les mots de passe ainsi stockés le sont de manière permanente et sont facilement récupérables avec des outils tels que netpass [NETPASS].

2.1.2 La boîte à outils

De la même manière, il convenait d'établir une liste d'outils permettant la collecte des hashes et des autres informations disponibles permettant à un attaquant de s'authentifier sur une autre machine du SI.

La liste des outils retenus est la suivante : (voir tableau page précédente).

MIMIKATZ ET LES TÂCHES PLANIFIÉES

Comme l'explique Benjamin Delpy sur son blog [KIWI], la récupération des mots de passe en clair associés à des tâches planifiées peut se faire selon deux approches.

Dans le contexte de l'utilisateur ayant créé la tâche, ou si l'on a pu élever ses privilèges à ceux de SYSTEM :

```
divers::secrets full
```

Sinon, il est nécessaire de s'injecter dans le processus LSASS pour extraire les données du *CredentialManager* :

```
privilege::debug
inject::service samss sekurlsa.dll
@getCredman full
```

Rien de vraiment surprenant donc, la plupart de ces outils font partie de la panoplie standard du bon *pentester*. D'autres outils existent, mais ont été écartés car ils sont obsolètes (pwdump2, pwdump6, ...), voire instables sur les systèmes récents. Le défi principal de cette sélection a d'ailleurs résidé dans la nécessité de réduire suffisamment cette liste de programmes afin que la réalisation des tests ne soit pas trop chronophage, tout en conservant un panel suffisamment large pour obtenir les résultats les plus exhaustifs possible.

DÉTECTION PAR LES ANTIVIRUS

Le taux de détection de ces outils par les différentes solutions antivirales existantes est également un critère intéressant, mais il sort du cadre de cet article. Ce paramètre pourrait en effet rentrer en ligne de compte lors du choix de l'« outil idéal », mais plusieurs techniques permettent de limiter la détection de ces « outils de hack » par les antivirus (encodage, recompilation, *packing*, ...).

2.2 Environnement de test et prérequis

Pour les besoins des tests, nous avons utilisé un environnement virtualisé basé sur VMware ESX. Celui-ci était composé de plusieurs machines, toutes intégrées au même domaine ainsi qu'un contrôleur de domaine sous Windows 2008 R2.

L'unique GPO mise en œuvre sur le domaine avait pour seul objectif d'assurer la disponibilité des accès RDP et Telnet à tous les utilisateurs. Aucun durcissement de la sécurité n'a été mis en œuvre afin d'effectuer les tests dans un environnement neutre.

Le domaine comprenait des machines virtuelles embarquant les versions suivantes de Windows :

- Windows XP 32 bits ;
- Windows 7 64 bits ;
- Windows Server 2003 SP1 32 bits ;
- Windows Server 2008 R2 64 bits.

Des comptes locaux ont été créés et des comptes sur le domaine ont également été provisionnés. Puis ces machines ont été préparées afin de fournir un environnement de test consistant, avec les prérequis suivants :

- services nécessaires installés et activés (RDP, Telnet, etc.) ;
- *firewall* désactivé, afin qu'il n'influe pas sur les tests ;
- boîte à outils déposée sur la machine, prête à l'emploi ;
- prise d'un *snapshot* propre, avec aucune trace en mémoire (sauf celles de l'utilisateur courant, dédié à la récolte des résultats).

La méthodologie de test peut alors être résumée à :

```
:begin
Restauration du snapshot propre
Authentification via la méthode choisie
Utilisation de l'outil et récupération des résultats
Tool++
goto begin
```

Cette procédure est ensuite répétée pour chaque méthode d'administration et sur chaque machine virtuelle.

3 Les résultats

3.1 « Qui laisse encore traîner ses hashes partout ? »

Le tableau ci-dessous présente les résultats obtenus à l'issue des tests, notamment si au moins un des outils testés a pu récupérer les données recherchées.

On constate que toutes les méthodes d'administration reposant sur un Interactive Logon laissent des traces exploitables en mémoire, contrairement aux méthodes effectuant un Network Logon. Parmi ces dernières, on note pour certaines la présence de Delegation Tokens (Psexec et Telnet). Ces Access Tokens peuvent être utilisés pour effectuer des opérations distantes malgré l'absence de hashes en mémoire, comme la création de compte sur le domaine par exemple :

```
> incognito.exe add_user -h <adresse du contrôleur de domaine>
<identifiant> <mot de passe>
> incognito.exe add_groupuser -h <adresse du contrôleur de
domaine> "Admins du domaine" <identifiant>
```

Cette action nécessite bien sûr que le compte associé à ce token dispose lui-même des droits d'administration sur le domaine.

Les tâches planifiées représentent quant à elles un cas particulier puisque c'est le CredentialManager de Windows qui stocke les mots de passe en mémoire, purement et simplement. L'utilisation d'outils tels que Mimikatz, permet de les récupérer en clair, et de les réutiliser sur le réseau. Les hashes et les Delegation Tokens sont quant à eux présents uniquement si la tâche à été exécutée au moins une fois.

Enfin, il est intéressant de noter que le comportement de Psexec n'est pas le même quand il est utilisé pour effectuer des actions distantes dans le contexte de l'utilisateur courant et quand on spécifie les identifiants à utiliser avec l'option « -u ». En effet, dans le premier cas, Psexec effectue un Network Logon et ne laisse pas de trace sur la machine distante, mais dans le second cas, il s'agit d'un Interactive Logon et les hashes de l'utilisateur spécifiés sont bel et bien en mémoire durant toute la durée d'exécution du processus distant.

On constate donc que pour protéger les comptes d'administration sensibles, il vaudrait mieux privilégier l'utilisation d'outils tels que WMIC, la Microsoft Management Console ou encore Psexec. Les sessions interactives, quant à elles (session locale ou *Remote Desktop* par exemple), devraient être proscrites car elles laissent les hashes, Delegation Tokens et mots de passe en mémoire...

3.2 Les outils : remise des prix

3.2.1 Les disqualifiés

En considérant comme éliminatoire le fait de ne pas être compatible avec les systèmes 64bits, il est d'ores et déjà possible d'écarter PWDumpX et cachedump, car ils ne fonctionnent tout simplement pas sur ces plateformes (mais notons que ce sont tout de même les précurseurs des outils d'extraction de hashes). Ces faiblesses éliminent de fait la possibilité de les utiliser sur un Windows Server 2008 R2 par exemple...

Pwdump7 a également été écarté à cause de ses fonctionnalités limitées. En effet, il n'est capable de récupérer que des hashes de comptes locaux (issus du fichier SAM).

3.2.2 Les challengers

Petit point négatif pour fgdump, puisqu'il embarque cachedump pour récupérer les hashes MS-CACHE, opération qui échoue systématiquement sur les architectures 64bits. De plus, il n'effectue aucune extraction des données présentes en mémoire. Il reste

Méthode / service	Présence de hashes LM/ NTLM en mémoire	Présence d'un Access Token	Présence de hashes MS-CACHE(V2)	Présence du mot de passe en mémoire
Local Logon	Oui	Delegation	Oui	Oui
RunAs	Oui	Delegation	Oui	Oui
Terminal Services	Oui	Delegation	Oui	Oui
Psexec	Non / Oui	Delegation	Non	Non
Telnet	Non	Delegation	Non	Non
WMIC	Non	Impersonation	Non	Non
NET USE	Non	Impersonation	Non	Non
MMC snap-ins	Non	Impersonation	Non	Non
Remote Regedit	Non	Impersonation	Non	Non
IIS	Non	Impersonation	Non	Non
Scheduled Tasks (la tâche a été lancée au moins une fois)	Oui	Delegation	Non	Oui

LOGON TYPES

Les différents types d'ouverture de session Windows, appelés *Logon Types*, sont consultables au sein de l'observateur d'événements.

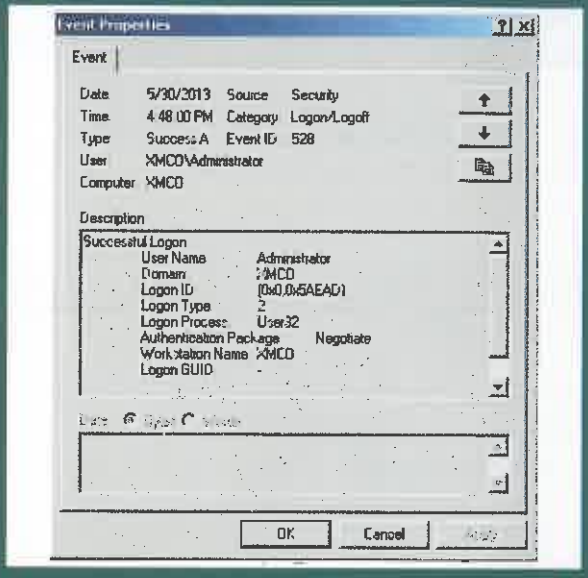

Figure 1

Cette classification permet la journalisation des ouvertures et fermetures de sessions ainsi que les méthodes d'authentification mises en place. Parmi les plus couramment utilisées, on retrouve :

- ***Interactive Logon*** **(type 2) : authentification physique, au clavier ou grâce à une *smartcard* ;**
- ***Network Logon*** **(type 3) : authentification distante via les RPC de Windows ;**
- ***Cached Interactive*** **(type 11) : authentification grâce au cache Active Directory (MS-CACHE).**

Il en existe 8 autres, assurant l'enregistrement d'événements tels que le déverrouillage d'un poste, l'utilisation de l'authentification Basic sur IIS ou encore le lancement d'un service.

cependant incontournable pour la récolte des hashes locaux ainsi que l'extraction des hashes des utilisateurs de domaines du fichier `NTDS.DIT`.

Gsecdump est également un outil fiable, permettant la récolte de hashes de comptes locaux ainsi que ceux présents en mémoire. Il extrait également les « secrets » en mémoire gardés par le processus LSASS.

Enfin, incognito est le seul (enfin presque...) à fournir la possibilité de récupérer et exploiter les fameux Access Tokens, qui peuvent parfois se révéler bien utiles. Après tout, le mauvais token qui traîne (ou le bon, question de point de vue) et il devient possible de créer un compte administrateur de domaine en quelques commandes...

3.2.3 « And the winners are... »

Sur le podium, les trois finalistes à toujours avoir sous la main sont donc Mimikatz, WCE et QuarksPwDump.

Mimikatz est en effet un outil très complet, activement maintenu, compatible avec toutes les versions de Windows actuellement supportées par Microsoft. Sa fonctionnalité « star » est de pouvoir extraire facilement les mots de passe en clair de la mémoire, et ce, sans injection dans le processus LSASS. Il implémente également de nombreuses autres fonctionnalités, comme la récupération de certificats et de clés privées ou le contournement de certaines restrictions imposées par la GPO (accès registre, utilisation de `CMD.EXE`, etc.). Il est même capable de « passer le hash ».

WCE, dont l'utilité première est d'effectuer cette fameuse attaque « Pass-The-Hash », n'est pas en reste. Il récupère sans difficulté les hashes de comptes locaux, hashes du domaine en mémoire, ainsi qu'en cache (MS-CACHE), mais extrait également les mots de passe en clair de la mémoire, comme Mimikatz.

Enfin, QuarksPwDump dispose des fonctionnalités classiques d'extraction de hashes de mots de passe de comptes locaux ou de domaine, en cache ou locaux. Il complète par ailleurs parfaitement les fonctionnalités des deux outils précédents en parcourant le fichier `NTDS.DIT` afin d'extraire les hashes de tous les utilisateurs du domaine. Il est de surcroît très stable, car il n'effectue pas d'opération d'injection dans les processus système.

Une mention spéciale est décernée à Meterpreter, qui intègre sous la forme de modules les fonctionnalités d'incognito et même depuis peu celles de Mimikatz **[MIMTER]**. En bonus, Meterpreter est également capable de lancer un exécutable sur la machine compromise à la manière de Psexec, à la différence notable que le tout se passe en mémoire, sans laisser de traces sur le disque :

```
meterpreter > execute -H -c -i -m -f Mimikatz.exe
```

Hors concours donc (c'est quand même quasiment de la triche). On déplorera par contre que l'implémentation actuelle de cette opération ne fonctionne pas sur les systèmes 64bits...

Figure 2

EXPLORATION DE NTDS.DIT AVEC METASPLOIT

Un script a récemment été inclus dans le framework Metasploit (`ntds_hashextract.rb` [MNTDS]) et permet l'extraction *offline* des hashes d'un fichier `NTDS.DIT` exporté, grâce à la bibliothèque `Libesedb` [LESBD]. Il fonctionne en mode *standalone* et n'est donc pour l'instant pas intégré dans les modules de post-exploitation de meterpreter.

Il peut cependant être utilisé en conjonction avec le module ntdsgrab [NTDSG], qui utilise la fonctionnalité *Volume Shadow Copy* de Windows afin d'extraire les fichiers nécessaires à la reconstruction des hashes utilisateur.

Le top du top reste bien sûr de développer ses propres outils. On maîtrise ainsi totalement les technologies employées et il est possible d'implémenter plusieurs approches pour récupérer ces données (injection, accès registre, SE_DEBUG, Volume Shadow Copy, etc.). De plus, la signature de ces outils ne sera pas présente dans les bases de données des antivirus...

Figure 3

4 Quelles contre-mesures ?

Alors soit, les méthodes d'administration les plus utilisées, comme Terminal Services et plus généralement les sessions interactives, sont également les plus susceptibles de faciliter la tâche des attaquants. Mais que faire face à ce constat ? Faut-il bannir totalement ce type de connexions ? Il est vrai que le risque serait grandement réduit si l'on préférait l'utilisation d'outils d'administration effectuant des Network Logons (Psexec, WMIC, ...). Il est par ailleurs possible de forcer une telle restriction à un groupe d'utilisateurs via une Stratégie de Groupe (voir Figure 4).

Mais cette recommandation passe souvent pour irréaliste, voire farfelue : « Pas de Remote Desktop, mais comment va-t-on faire pour administrer les machines ? ». Sans compter que, bien qu'il soit un outil puissant, l'usage de WMIC en entreprise reste marginal.

4.1 Les classiques

Alors il y a bien sûr les recommandations classiques qui tiennent surtout de la « défense en profondeur », si chère à Microsoft :

- Ne pas autoriser la connexion avec des comptes privilégiés sur des machines considérées comme « à-risque ».
- Restreindre le nombre de comptes à hauts privilèges sur le domaine.
- Ne pas autoriser les comptes locaux à ouvrir des sessions à distance.
- Faire respecter la politique du « Least User Access », en utilisant de préférence des comptes non privilégiés pour les connexions distantes.
- Utiliser des postes et des comptes dédiés à l'administration du domaine.
- Ne pas se connecter avec des comptes sensibles sur un équipement « à risque ».
- Désactiver les comptes d'administration locaux.
- Ne pas réutiliser les mots de passe.
- Créer les tâches planifiées avec des comptes dédiés, disposant de privilèges limités.
- Utiliser NTLMv2 uniquement, ou encore mieux, Kerberos (sic).
- Limiter le nombre de données d'entrées MS-CACHE stockées localement.
- Ne pas intégrer les utilisateurs aux groupes d'administration locaux.

Le défi avec ces recommandations est de savoir les appliquer aux bons groupes d'utilisateurs et aux bons *assets*. La lecture du document « Mitigating Pass-the-Hash Attacks and Other Credential Theft Techniques » publié par Microsoft donne par ailleurs de nombreuses

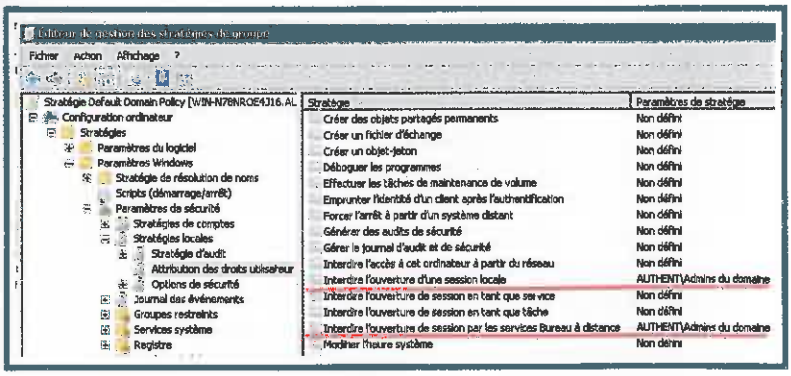

Figure 4

clés ainsi que des explications détaillées à ceux qui souhaiteraient mettre en œuvre ces suggestions **[MICRO]**.

4.2 Les autres

Mais il existe quelques solutions alternatives, à la fois plus pragmatiques et faciles à mettre en œuvre, qui peuvent mettre des bâtons dans les roues des attaquants.

En effet, ces faiblesses sont pour la plupart dues à la volonté de Microsoft d'assurer la rétrocompatibilité entre ses produits. Or certaines de ces fonctionnalités ne sont pas forcément indispensables au bon fonctionnement de chaque infrastructure.

4.2.1 Désactiver les Authentication Packages inutilisés

Désactiver les packages Wdigest, TsPkg et Kerberos permet de faire disparaître les mots de passe stockés en mémoire. Il suffit pour cela d'éditer la clé registre `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages`, qui est en fait une liste des différents packages utilisés. Après redémarrage, les éléments supprimés de la liste ne seront plus chargés. Assurez-vous toutefois au préalable de ne pas avoir besoin de ces packages afin d'éviter de sérieux problèmes, notamment dans le cas de Kerberos !

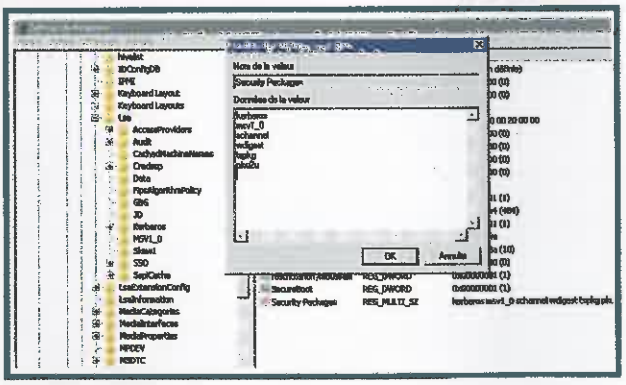

Figure 5

4.2.2 Désactiver les privilèges de débogage

Une solution moins contraignante pour éviter que les outils présentés ne puissent extraire de telles informations de la mémoire du processus LSASS est de refuser le privilège `SE_DEBUG` à tous les comptes utilisateur, comme le recommande d'ailleurs le SANS **[SANS]**. Il est en effet peu probable que l'attribution de ce privilège soit nécessaire dans un environnement de production. Cette recommandation est tout à fait applicable via une Stratégie de Groupe Active Directory :

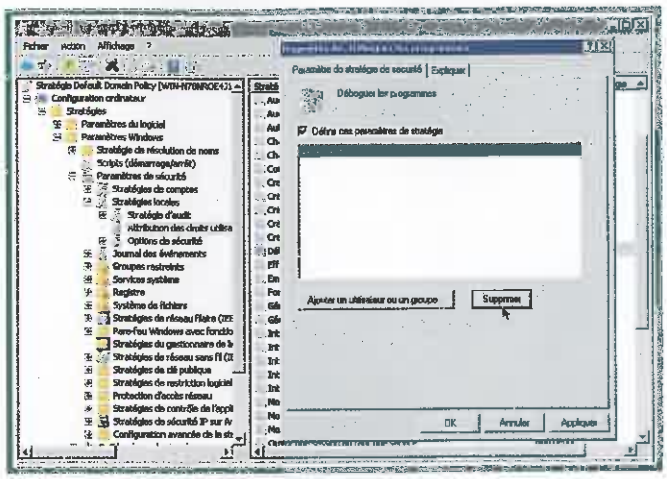

Figure 6

4.2.3 Désactiver la délégation

Le problème des Access Tokens est également assez facile à prendre en charge au niveau du domaine, en désignant les comptes d'administration du domaine comme « sensibles et ne pouvant être délégués » :

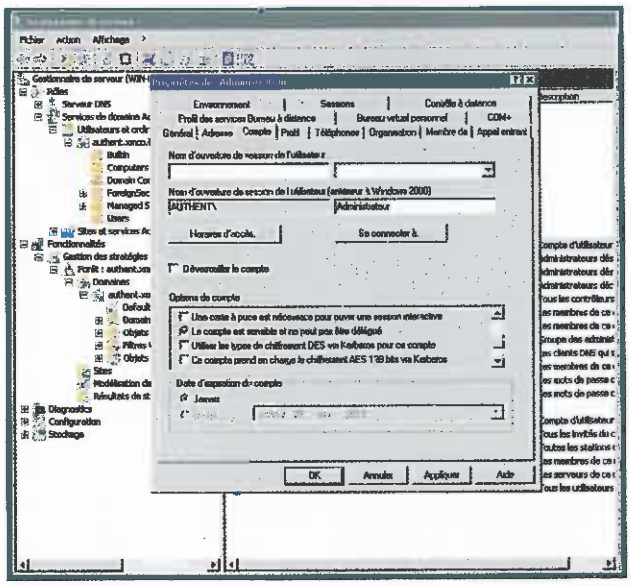

Figure 7

Il faut préciser toutefois qu'un Delegation Token sera tout de même généré dans le cas d'une session interactive, mais le contrôleur de domaine refusera les tentatives d'authentification à distance, en forçant l'exécution des actions distantes sous un Anonymous Logon.

4.2.4 Désactiver le service Volume Shadow Copy

Enfin, quand aucun outil ne semble fonctionner, l'ultime recours du pentester est de faire appel au service « Volume Shadow Copy ». Cette méthode de récupération n'a pas

été incluse dans la liste des outils sélectionnés car il s'agit en réalité d'une fonctionnalité interne à Windows, utilisée pour réaliser les sauvegardes du système. Ce service permet de copier n'importe quel fichier, y compris ceux verrouillés par le système d'exploitation. Il est donc très pratique lorsqu'on souhaite extraire manuellement les ruches SAM, SYSTEM, SECURITY ou la base `NTDS.DIT`. Dans un deuxième temps, d'autres outils (samdump, QuarksPwDump, ntds_dump_hash ou autre) permettent d'extraire ces hashes hors-ligne.

Or si les mécanismes de sauvegardes en place sur le SI ne nécessitent pas l'activation de ce service, pourquoi ne pas intégrer à ses GPO une règle désactivant ce service, ainsi que son utilitaire de contrôle en ligne de commandes (`vssadmin`).

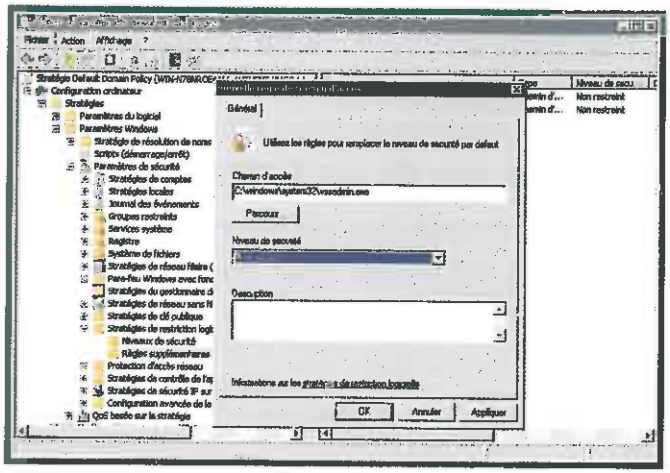

Figure 8

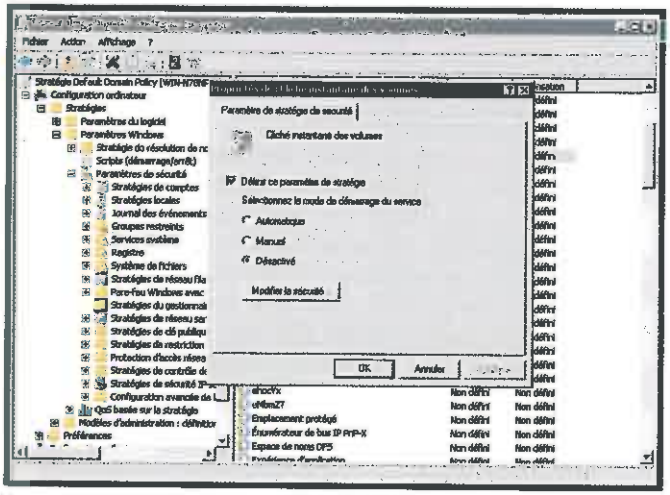

Figure 9

Conclusion

Il existe donc des solutions pour se protéger et les faiblesses inhérentes à l'utilisation des produits Microsoft au sein de son infrastructure ne sont pas des fatalités. Aucune des recommandations formulées ici n'est d'ailleurs particulièrement complexe à mettre en œuvre. Il est donc surprenant de voir que le taux de succès des outils identifiés dans cet article en conditions réelles reste très élevé, la principale difficulté étant bien souvent d'échapper à la détection des antivirus.

Et on notera également que pour peu qu'on sache s'en servir, un bon vieux WMIC ou même un PSexec vaut toujours mieux qu'un MSTSC :-) ■

REMERCIEMENTS

Je remercie tous les membres de l'équipe de XMCO, ainsi que Nicolas Ruff et Benjamin Caillat, pour leurs nombreux conseils avisés et leurs critiques constructives.

RÉFÉRENCES

[WEAKLM] http://en.wikipedia.org/wiki/LM_hash#Security_weaknesses

[WCE] http://www.ampliasecurity.com/research/wcefaq.html#preventcleartextpwddump

[MIMI] voir MISC n°66 « Utilisation avancée de Mimikatz », G. Lopes et M. Mauger

[MSTOK] http://msdn.microsoft.com/en-us/library/Aa374909.aspx

[NETPASS] http://www.nirsoft.net/utils/network_password_recovery.html

[KIWI] http://blog.gentilkiwi.com/securite/mimikatz/sarcleuse-credman-planificateur-taches

[MIMTER] https://github.com/rapid7/meterpreter/pull/9

[LESBD] https://code.google.com/p/libesedb/

[SUDOMAN] https://twitter.com/sud0man

[NTDSG] https://github.com/rapid7/metasploit-framework/pull/1023

[MNTDS] https://github.com/rapid7/metasploit-framework/pull/1024

[MICRO] http://download.microsoft.com/download/7/7/A/77ABC5BD-8320-41AF-863C-6ECFB10CB4B9/Mitigating%20Pass-the-Hash%20(PtH)%20Attacks%20and%20Other%20Credential%20Theft%20Techniques_English.pdf

[SANS] https://files.sans.org/summit/forensics11/PDFs/Protecting%20Privileged%20Domain%20Accounts%20during%20Live%20Response.pdf

Les outils cités dans l'article sont téléchargeables aux adresses suivantes :

gsecdump : http://www.truesec.se/sakerhet/verktyg/saakerhet/gsecdump_v2.0b5

pwdump7 : http://www.tarasco.org/security/pwdump_7/

fgdump : http://www.foofus.net/~fizzgig/fgdump/

Mimikatz : http://blog.gentilkiwi.com/mimikatz

Meterpreter : http://www.metasploit.com/

PWDumpX : http://packetstormsecurity.com/Crackers/PWDumpX14.zip

WCE : http://www.ampliasecurity.com/research.html

QuarksPwDump : https://github.com/quarkslab/quarkspwdump

cachedump : http://packetstormsecurity.com/files/36781/cachedump-1.1.zip.html

incognito : http://sourceforge.net/projects/incognito/