HERVÉ SCHAUER CONSULTANTS
Cabinet de Consultants en Sécurité Informatique depuis 1989
Spécialisé sur Unix, Windows, TCP/IP et Internet

Bénéfices de l'accès direct aux disques à l'aide du framework Metasploit

Danil Bazin <Danil.Bazin@hsc.fr>

HSC by Deloitte

7 avril 2016

HSC by Deloitte

Conseil en sécurité des systèmes d'information depuis 1989.

- PME française avec 26 ans d'expérience.
- Exclusivement des interventions d'expertise SSI:
  - Pas de distribution, ni intégration, ni infogérance, ni délégation de personnel ;
  - Prestations : conseil, études, audits, tests d'intrusion, formations ;
  - Garantie d'indépendance.
- Domaines d'expertise
  - Sécurité Windows / Unix et linux / embarqué / informatique industrielle / applications
  - Enquêtes inforensiques / Expertise judiciaire
  - Sécurité des réseaux : TCP/IP, téléphonie, réseaux opérateurs, réseaux industriels...
  - Organisation de la sécurité, droit des systèmes d'information

Modules Metasploit

Développées par HSC

- Modules de post-exploitation de Metasploit:
  - *file_from_raw_ntfs* : permet de contourner les verrous en écriture de Windows sur des fichiers clés comme les fichiers de ruches Windows ou le fichier NTDS.DIT
  - *bitlocker_fvek* : permet de récupérer la clé de chiffrement maitre de Bitlocker
- Point commun de ces deux modules : utilisation d'un accès direct aux partitions / disques.

Metasploit

Framework de tests d'intrusion

- Framework OpenSource sous licence libre (BSD) crée par H.D Moore en 2003
  - OpenSource signifiant que n'importe qui peut accéder au code source
  - La licence libre autorise la modification du programme
- Rachetée par Rapid7 en octobre 2009
- Développé en Ruby
- Hébergé sur Github.io

Metasploit

Framework de tests d'intrusion

- La force de Metasploit lui vient de la séparation entre:
  - Les *exploits* (1524 dans la version 4.11.19) : le code qui permet de déclencher la vulnérabilité et d'exécuter du code arbitraire, par exemple:
    - Un PDF forgé pour une version vulnérable d'Adobe Reader
    - PSexec pour exécuter du code à distance sur un Windows
    - ...
  - les *payloads* (charges utiles) (436 dans la version 4.11.19) : le code arbitraire qui sera exécuté sur le poste de la victime, par exemple:
    - Une porte dérobée en écoute sur le port 443
    - Un reverse-shell allant se connecter sur le port 80 de l'attaquant
    - ...
  - les modules de post-exploitation (260 dans la version 4.11.19) : le code qui utilise la connexion avec la *payload* pour piller la victime ou l'utiliser pour rebondir:
    - *cachedump* : récupère les condensats des mots de passe des comptes du domaines qui se sont connectés sur le poste Windows

Metasploit

Framework de tests d'intrusion

- Intensivement utilisé en test d'intrusions internes:
  - Peut être couplé avec une base de données PostgreSQL
  - Peut être alimentés avec le résultat de scan *nmap*
  - Contient de nombreux modules d'attaque par force brute et dictionnaires de comptes par défaut
  - Ajout automatique des mots de passe trouvés à la base de données
- Gére l'exécution de code malveillant sur de nombreux systèmes d'exploitation:
  - Windows
  - Linux
  - Android
  - AIX
  - ...

Meterpreter

Interpréteur Metasploit

- Le Meterpreter est la *payload* la plus puissante de Metasploit
  - Silencieuse : n'écrit aucun fichier sur le disque
  - Utilise des communications chiffrés
  - Extensible : Des fonctionnalités peuvent être ajoutés dynamiquement pendant l'exécution
  - Multi-canaux : Plusieurs connexions en parallèles peuvent être transmises à travers le même Meterpreter
- La version Windows est celle qui possède le plus de fonctionnalités:
  - Permet de s'injecter dans la mémoire d'un processus et de migrer dynamiquement dans d'autres
  - Un *keylogger* permet de saisir les frappes claviers de l'utilisateur
  - La webcam de la victime peut être utilisé pour prendre des photos

Meterpreter
Développement

- Quelques astuces de développeur Metasploit:
  - Après déploiement du Meterpreter, la commande *irb* lance un terminal ruby permettant de tester le code du module en cours de développement
  - La commande *reload* appliqué sur le module en cours de développement permet de recharger le module depuis le code source
  - La commande *reload* ne s'applique qu'au module courant, il est donc plus rapide pour les tests de grouper toutes les classes en développement dans le fichier du module

Meterpreter
Railgun

- Railgun : fonctionnalité de post-exploitation permettant de dialoguer avec l'API Windows de la victime en utilisant la connexion avec un Meterpreter.
  - Peut être facilement appelé depuis un Meterpreter avec *irb*
  - les appels s'effectuent avec des commandes de type session.railgun.*nomdll*.*nomfonction*(*liste des arguments de la fonction*)
- 5 fonctions de l'API Windows suffisent pour les modules d'accès direct aux disques:
  - kernel32.GetFileAttributesW
  - kernel32.CreateFileW
  - kernel32.CloseHandle
  - kernel32.ReadFile
  - kernel32.SetFilePointer

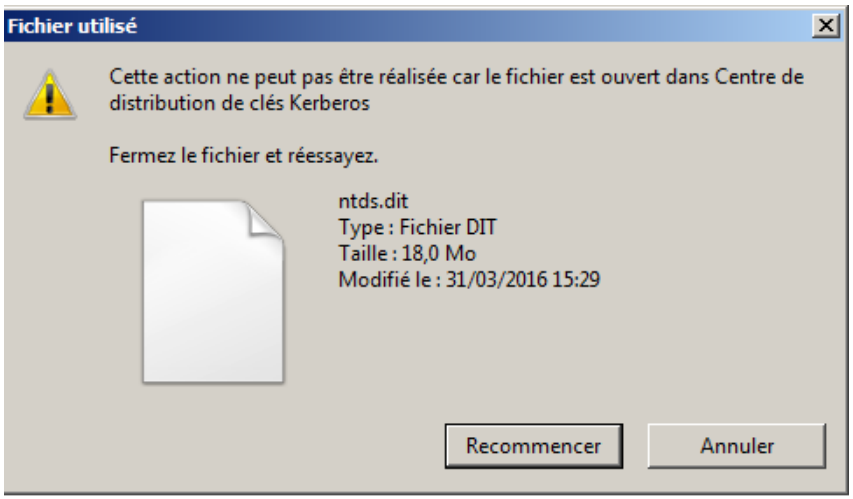
Win32 Device Namespaces

- La fonction kernel32.CreateFileW de l'API Windows permet d'ouvrir les fichiers sous Windows sous la forme d'un descripteur de fichier.
- Mais aussi des disques, volumes etc. , en utilisant le préfixe "\\.\"
  - Par exemple le volume C peut s'ouvrir avec "\\.\C:"
  - Celui-ci permet d'accéder à la partition NTFS, (même si Bitlocker est utilisé)
  - L'accès direct à un disque s'effectue avec "\\.\PhysicalDrive0"
  - Ce descripteur de fichier accède au disque qui commence généralement par la table des partitions
  - puis les partitions sont accessibles, en version chiffré si Bitlocker est utilisé

Contournement du verrou Windows sur les fichiers systèmes

Objectifs

- Lors de la compromission d'un contrôleur de domaine Windows, un attaquant cherche à récupérer le fichier *NTDS.dit*.
  - Ce fichier contient entres autres la liste des utilisateurs du domaine et les condensats de leurs mots de passe
  - Mais ce fichier est verrouillé par le système pour éviter des actions concurrentes :



Contournement du verrou Windows sur les fichiers systèmes

Solutions existantes

- Afin de contourner cette restriction, la technique la plus couramment employée est d'utiliser les *Volume Shadow Copy*:
  - (Get-WmiObject -list win32_shadowcopy).create("C:\", "ClientAccessible")
  - vssadmin.exe List Shadows
  - cmd /c mklink /D C:\vss2 \\ ?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\
- Mais cette technique génère un instantané de la partition entière, ce qui est assez "bruyant" et intrusif

Contournement du verrou Windows sur les fichiers systèmes

NTFS

- Une autre solution pour récupérer ce fichier consiste à utiliser un accès direct au disque pour le récupérer
  - Mais une fois la partition ouverte, il faut parcourir la structure NTFS pour accéder aux fichiers
- Windows utilise son pilote NTFS pour parser les systèmes de fichier NTFS mais ne propose pas d'API
- Les outils de Forensics comme FTKimager ou The Sleuth Kit utilisent l'accès direct aux disques pour extraire des fichiers
  - Mais ils sont difficilement intégrable à Metasploit

Contournement du verrou Windows sur les fichiers systèmes

NTFS - Premiers octets

- Une partition NTFS commence par un secteur de *Boot*:

- Dans cette section, les tailles des unités sont définies en fonction de la taille totale de la partition:
  - Nombre d'octets par secteur
  - Nombre de secteur par cluster
  - Nombre de cluster par enregistrement de la MFT
  - Numéro de cluster de la MFT et de la copie de la MFT (MFTMirr)
  - Numéro de série du volume NTFS

Contournement du verrou Windows sur les fichiers systèmes

NTFS - Premiers octets

- Par exemple avec l'outil de *The Sleuth Kit, fsstat*:

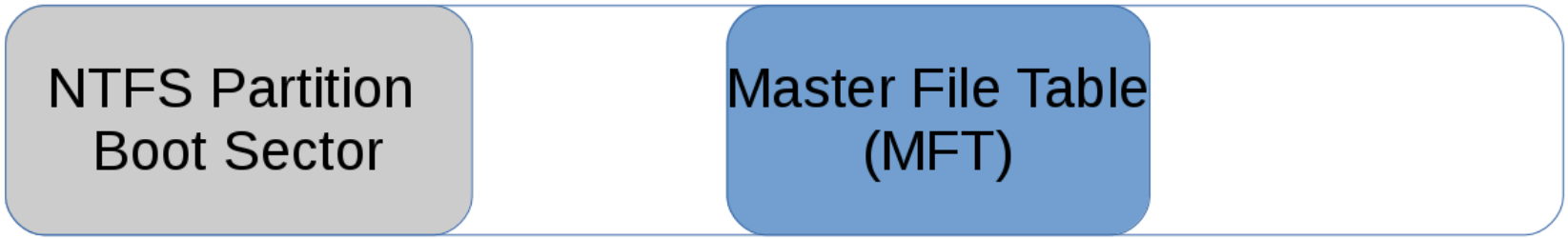
```
C:\sleuthkit-4.1.3-win32\bin>fsstat.exe \\.\C:
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: NTFS
Volume Serial Number: 063C921B3C920639
OEM Name: NTFS
Version: Windows XP

METADATA INFORMATION
--------------------------------------------
First Cluster of MFT: 786432
First Cluster of MFT Mirror: 2
Size of MFT Entries: 1024 bytes
Size of Index Records: 4096 bytes
Range: 0 - 168448
Root Directory: 5

CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 4096
Total Cluster Range: 0 - 7838206
Total Sector Range: 0 - 62705662
```

Contournement du verrou Windows sur les fichiers systèmes

NTFS - MFT - Premiers enregistrements



- La MFT est un tableau d'enregistrement de taille fixe
- Chaque enregistrement représente un fichier/dossier
- Les premiers enregistrements sont fixes:
  - 0 - $MFT Représente la MFT
  - 1 - $MFTMirr réprésente une copie des premiers enregistrements de la MFT
  - ...
  - 5 - $ Racine du système de fichier

Contournement du verrou Windows sur les fichiers systèmes

NTFS - MFT - Attributs

- Chaque enregistrement de la MFT est un tableau d'attribut
  - Un attribut peut être stocké dans la MFT s'il est suffisamment petit, on parle d'attribut résidant
  - ou seule ses adresses peuvent être stockés sous forme de *data-run* (listes d'adresses), on parle d'attribut non-résidant
  - Plusieurs type d'attributs existent:
    - $DATA contient le contenu du fichier
    - $STANDARD_INFORMATION contient le propriétaire et les dates de création, dernière modification, accès et modification des metadonnées
    - $FILENAME : nom, dates de création, dernière modification, accès et modification des métadonnées (valeurs moins précise que celles stockés dans $STANDARD_INFORMATION)
    - $INDEX_ROOT et $INDEX_ALLOCATION sont présents pour les enregistrements de répertoires et stockent la liste des fichiers qu'ils contiennent

Démonstration

- Module *file_from_raw_ntfs*

Indicateurs de compromission (IOC)

- Sans compter ceux générés par l'exécution du Meterpreter : Aucun

Références

- Brian Carrier. (2005). File System Forensic Analysis.

Extraction de la clé maitre Bitlocker

Objectifs

- Un attaquant cherche toujours à pérenniser son accès le plus discrètement possible.
  - Si accès physique ultérieur possible (cas d'un ordinateur portable par exemple) : récupération des clés de chiffrement pour accéder au système de fichiers ultérieurement.

Solutions existantes pour l'extraction de la FVEK

- Deux solutions pour extraire la clé maitre:
  - *Dislocker* utilisé avec la clé de recouvrement (par exemple) si une image hors-ligne est disponible
    - En mode verbeux, la clé maitre est affichée
  - Depuis une image de la RAM (avec *winpmem* par exemple)
    - Puis analyse avec *Volatility* avec le plugin bitlocker de elceef

Bitlocker

Clés de chiffrement intermédiaires

- Plusieurs secrets peuvent être utilisés pour déchiffrer une partition Bitlocker:
  - TPM (Trusted Platform Module)
  - Mot de passe
  - Fichier de clé (Startup key), présente sur une autre partition ou sur une clé USB
  - Clé de recouvrement
  - ...

Bitlocker

Clé de recouvrement

- Les clés de recouvrement sont de la forme:
  - *709434-416845-498971-632753-185251-065296-323609-146927*
- Elles peuvent être générées depuis un volume Bitlocker ouvert à l'aide de la commande
  - *manage-bde -protectors -add -rp C:*
- Extraites à l'aide de la commande:
  - *manage-bde -protectors -get C:*
- Dérivées selon la recette suivante:
  - Chaque partie est divisée par 11 puis concaténée
  - Un sel est ajouté, la *stretch key*
  - Condensée 1048576 fois (0x100000) avec la fonction SHA256

Bitlocker

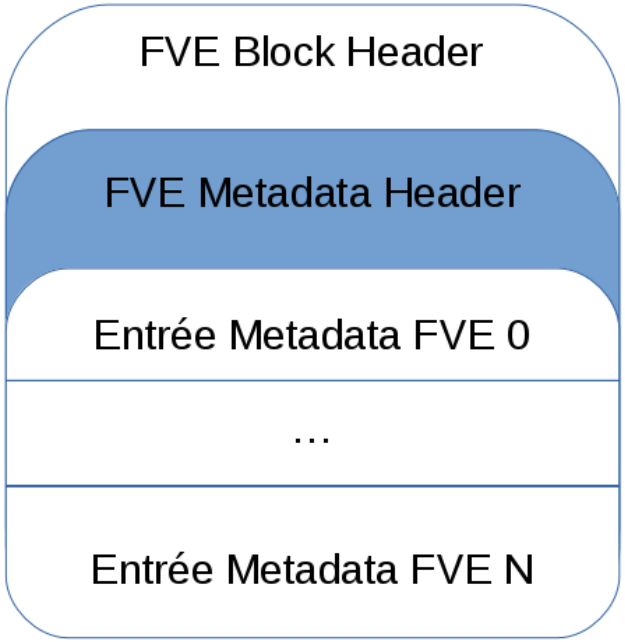
Clés

- Toutes ces clés de chiffrement permettent de déchiffrer une clé de volume (Volume Master Key (VMK))
- Cette clé de volume va déchiffrer la clé AES (Full Volume Encryption Key(FVEK)) utilisée pour chiffrer le volume
  - La FVEK ne peut être changée, en cas de compromission, le disque doit être déchiffré puis rechiffré

Bitlocker

Structures du disque

- Une partition Bitlocker commence par un en-tête au troisième octet : ”-FVE-FS-”
- L’offset 176 contient l’adresse de la structure contenant les clés de chiffrement du volume:



Bitlocker

Openssl-CCM

- Les clés sont chiffrés en utilisant AES-{128,256}-CCM
- La bibliothèque Openssl de ruby ne contient pas ce mode de chiffrement
- Une implémenation openssl-ccm en ruby est disponible sur GitHub mais ne fonctionnait pas :

```ruby
class CCM
  # Searches for supported algorithms within OpenSSL
  #
  # @return [[String]] supported algorithms
  def self.ciphers
    l = OpenSSL::Cipher.ciphers.keep_if { |c| c.end_with?('-128-CBC') }
    l.length.times { |i| l[i] = l[i][0..-9] }
    l
  end
```

- À cause d'une confusion entre taille de clé et taille de block

Démonstration

- Module *bitlocker_fvek*

Indicateurs de compromission (IOC)

- Génération puis suppression d'une clé de recouvrement si celle-ci n'existait pas

Références

- La documentation de libbde est très compréhensible: `https://github.com/libyal/libbde/blob/master/documentation/BitLockerDriveEncryption(BDE)format.asciidoc`

Questions ?