

CyberDef 4 – Groupe 6

Maximilien CHAUX  
Madigan LEBRETON  
Bertrand MARTIN  
Alan PICARD

# **Développement d'une attaque informatique à but de sensibilisation basée sur un rootkit**

Rapport technique

Version 0

# Sommaire

|   |    |
|---|----|
| 1. Introduction   | 3  |
| 1. Définition d'un rootkit                                | 3  |
| 2. Présentation de l'attaque                              | 4  |
| 1. Scénario de l'attaque                                  | 4  |
| 2. Cible de notre attaque                                 | 4  |
| 3. Architecture de la solution                            | 5  |
| 3. Les Composants de l'attaque                            | 5  |
| 1. Package Livraison                                      | 5  |
| 2. Package Exploitation – Installation du package rootkit | 6  |
| Executable Userland – SpoolFool                           | 6  |
| Librairie Userland – Dropper                              | 7  |
| 3. Package Rootkit  | 7  |
| Cacher des processus                                      | 7  |
| Autres techniques de dissimulation des processus          | 10 |
| Cacher des fichiers                                       | 11 |
| Application Userland                                      | 14 |
| 4. Package Command & Control                              | 18 |
| 4. Se défendre face à ces attaques                        | 19 |
| 1. Prévention   | 19 |
| Ce que Microsoft propose                                  | 19 |
| Ce que vous pouvez faire                                  | 23 |
| 5. Détection  | 23 |
| Patchguard  | 23 |
| Minifilter  | 24 |
| Cacher des processus                                      | 24 |
| Connaissance du parc informatique                         | 25 |
| Utilisation d'antivirus et/ou d'EDR                       | 26 |
| Sécurité réseau   | 26 |
| 6. Réaction   | 26 |
| Isolation des machines infectées                          | 26 |
| Résoudre les chemins d'attaques utilisés                  | 26 |
| Mise en place d'équipes spécialisés                       | 27 |
| 5. Conclusion   | 27 |

# 1.Introduction

Le présent document est le rapport technique de l'attaque informatique basée sur un rootkit réalisée sur l'entreprise fictive HealthSIBS et son laboratoire fictif dont elle a récemment fait l'acquisition : Vannalyz'.

Dans ce rapport, l'attaque réalisée sera dans un premier temps présentée. Les différentes techniques utilisées pour l'attaque seront détaillées et expliquées, de sorte à ce que des équipes techniques type DSI puissent en comprendre le fonctionnement.

Ensuite, des recommandations à suivre seront données pour mettre en place des capacités défensives à la hauteur de ce type d'attaque. Ces capacités défensives traiteront de la prévention, de la détection ainsi que de la réponse et de la réaction aux attaques de ce type. Cette partie intéressera notamment les responsables de la sécurité des systèmes d'information (RSSI) car la mise en place de ces recommandations leur permettra de renforcer le niveau de sécurité global de leur système d'information.

Enfin, nous aborderons les limites des recommandations présentées. Les RSSI seront aussi intéressés par ces limites, car une bonne maîtrise de la sécurité d'un système d'information passe notamment par la connaissance de ce système d'information, des outils en place et de leurs faiblesses.

## 1. Définition d'un rootkit

Un rootkit est un type de malware qui a pour but de rester indétecté. Cela permet à un attaquant d'accéder à une machine ou un réseau, afin d'y voler des données, réaliser de l'espionnage etc... Le caractère dangereux est celui de la furtivité. Certaines attaques par rootkit ne furent découvertes seulement que des années après l'intrusion initiale.

A l'origine, le terme rootkit désignait des kits sous Linux permettant, suite à une élévation de privilèges, d'avoir un accès "root" à distance sur la machine, d'où le nom de rootkit. Aujourd'hui, le terme a évolué pour désigner les malwares furtifs.

Les conséquences d'une attaque avec rootkit peuvent être dévastatrices. Comparé au ransomware, les dégâts possibles sont moins évidents, et le caractère furtif a tendance à mettre au second plan ce type d'attaque lors de l'élaboration d'un plan de sécurité. Néanmoins, une entreprise ou une personne infectée par ce type de malware pourrait se voir espionner durant des années. L'avantage concurrentiel d'une entreprise par exemple se verrait complètement réduit à néant si un concurrent venait à utiliser un rootkit. Tous types d'informations confidentielles pourraient être volées sans même que le propriétaire ne s'en aperçoive. C'est pourquoi il est important de considérer ces attaques comme dévastatrices et à considérer comme des attaques de premier plan.

## 2.Présentation de l'attaque

### 1. Scénario de l'attaque

Nous présentons notre scénario d'attaque en nous appuyant sur le Framework **Cyber Kill Chain**. Cela nous permet de présenter les informations ainsi que le déroulé de notre attaque de manière claire et globale.

Ce framework sera utilisé pour caractériser les différents composants de la solution que nous proposons. Ainsi, il sera plus simple de comprendre la chronologie de l'utilisation des composants.

- **Reconnaissance**
  - Etat de l'art sur les rootkits Windows et les vulnérabilités touchant Windows
  - OSINT / sources de risque et SPOF
- **Armement**
  - Développement du rootkit
- **Livraison**
  - Spearphishing : recherche sur les techniques de mail spoofing
  - Rubber Ducky
- **Exploitation**
  - Exécution du programme seringue
- **Installation**
  - Installation et exécution du rootkit
- **Commande & Contrôle**
  - Command & Control HTTPS

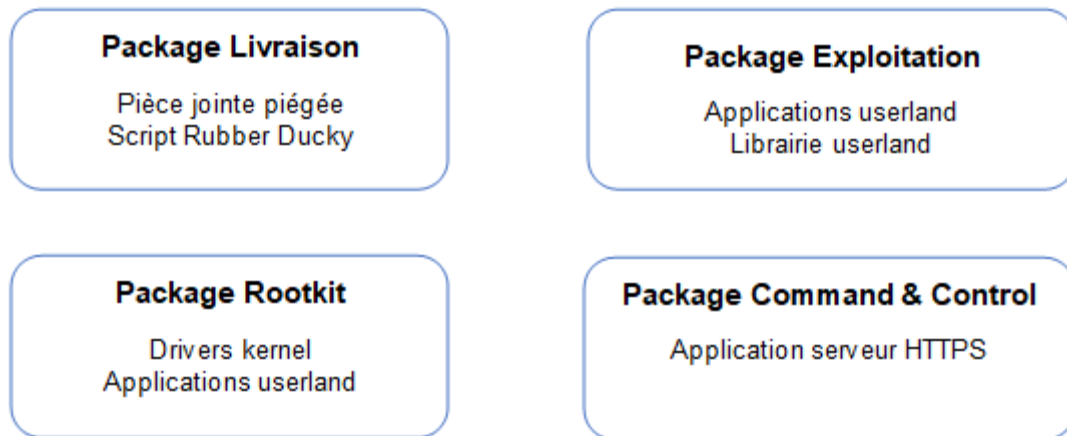
### 2. Cible de notre attaque

Pour développer notre attaque, nous avons dû sélectionner une cible à attaquer. Le système d'exploitation sélectionné est Windows 10, dans sa version 1909. Cette version de Windows est sortie environ deux ans et six mois avant l'écriture de ce document. Elle est donc relativement récente.

Cette version de Windows a été choisie car elle nous permet de charger des drivers kernel (noyau). Windows 10 requiert que les drivers noyau soient signés. Nous avons pu récupérer une paire clé/certificat ayant fuité, nous permettant de signer nos drivers et charger ces derniers. Or, le certificat ayant été révoqué par Microsoft, il est détecté sur les versions plus récentes que Windows 1909 et nous ne pouvons donc pas charger nos drivers sur ces versions.

### 3. Architecture de la solution

La solution que nous proposons se divise en quatre principaux packages, composés de multiples composants. La figure suivante présente les différents packages de notre solution.



Chaque package proposé s'inscrit dans une des étapes de la **Cyber Kill Chain**. Ainsi, nous pouvons présenter l'association suivante entre package et étape.

| Etape de la Cyber Kill Chain | Package proposé           |
|------------------------------|---------------------------|
| Livraison                    | Package Livraison         |
| Exploitation                 | Package Exploitation      |
| Installation                 | Package Rootkit           |
| Commande & Contrôle          | Package Command & Control |

### 3. Les Composants de l'attaque

Les composants des différents packages de la solution sont présentées dans la suite de ce document.

#### 1. Package Livraison

Pour la livraison, nous avons choisi d'employer deux méthodes : le spearphishing ainsi qu'un rubber ducky.

Pour le spearphishing, nous avons choisi d'envoyer un fichier Excel de la suite Microsoft Office avec une macro malveillante. Cette macro se compose de deux méthodes, DownloadFile qui nous permet de télécharger un fichier à un chemin précis, et Payload qui utilise la fonction DownloadFile pour télécharger trois fichiers : SpoolFool.exe, dropper.exe et AddUser.dll qui seront détaillés plus loin. La fonction exécute ensuite la commande « SpoolFool.exe -dll AddUser.dll » qui exécutera ensuite notre dropper (voir package exploitation). Le téléchargement d'un fichier avec une extension .exe est considéré comme une action malveillante. Plusieurs solutions peuvent être employées pour ne pas être détecté. Nous

avons choisi de compresser les fichiers dans un .zip, de le télécharger sur la machine victime puis le décompresser, ce qui suffit à éviter la détection.

Évidemment, ces attaques sont de moins en moins vraisemblables grâce à la sensibilisation sur les pièces jointes malicieuses. Cependant, cela reste un moyen relativement efficace pour entrer dans un système d'information pour le moment. Le niveau technique requis pour écrire une macro malveillante passant à travers AMSI n'est pas très élevé et un script kiddie pourrait réussir.

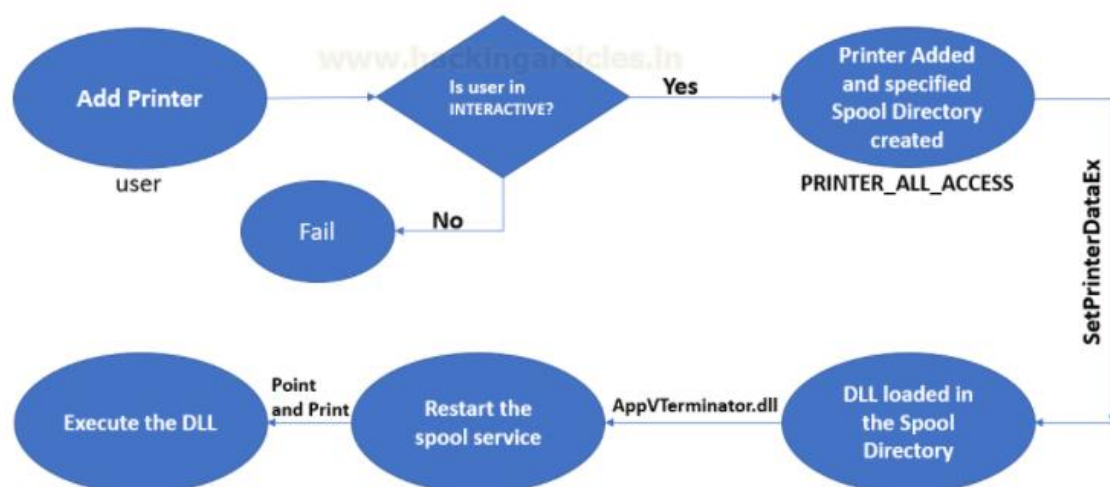
Pour la méthode utilisant le rubber ducky, nous avons choisi de récupérer les fichiers nécessaires sur la clé USB directement. Pour cela, nous avons choisi de flasher le firmware « Twin Duck » disponible sur le Github du projet. Ce firmware nous permet d'utiliser le rubber ducky en tant que clavier dissimulé mais aussi en tant que périphérique de stockage. Pour cela, nous avons renommé nos fichiers et les avons définis comme fichiers cachés pour tenter de les dissimuler du mieux possible. Une fois la clé branchée, il lance un PowerShell grâce à la commande WIN + R. Ce PowerShell récupère le chemin de la clé et exécute le script PowerShell de la clé qui va récupérer notre dropper et l'exécutable SpoolFool sur le stockage. La documentation disponible sur leur site est telle que le temps de programmation du rubber ducky est moindre et tout se fait rapidement avec des outils open source. De même que la méthode par macro, la sensibilisation grandissante fait que les gens sont moins enclins à entrer une clé USB dans leur machine si celle-ci est trouvée.

## 2. Package Exploitation – Installation du package rootkit

Le package exploitation se décompose en 2 fichiers : l'un permettant de réaliser une élévation de privilège, le second, exécuté par le premier, s'attèle à télécharger tous les composants du package rootkit afin de les installer correctement.

### Executable Userland – SpoolFool

C'est le premier exécutable qui est exécuté. Il utilise la vulnérabilité SpoolFool afin d'élever ses privilèges. Cette vulnérabilité exploite le service spouleur d'impression sur les machines Windows. Elle est très récente (février 2022) et se situe dans la lignée de PrintNightmare, vulnérabilité que nous avons envisagée dans le dossier de définition. Voici un schéma résumant le fonctionnement de la vulnérabilité :



Comme nous pouvons le voir, n'importe quel utilisateur appartenant au groupe INTERACTIVE (fréquent) peut ajouter une imprimante et obtenir les droits PRINTER\_ALL\_ACCESS. Il est alors possible de modifier le répertoire de spouleur en spécifiant un nouveau répertoire, ce qui aura pour effet de le créer avec les droits d'écriture pour tout le monde. On peut alors charger la DLL malicieuse dans ce répertoire et l'exécuter en tant que NT Autorite/Systems

## Librairie Userland – Dropper

Cette DLL est une simple librairie possédant une fonction permettant de lancer l'exécutable dropper en tant que NT Autorite/Systems (Administrateur). Le dropper lui va permettre de télécharger tous les composants du package Rootkit depuis le serveur C&C, les installer correctement, et supprimer tous fichiers inutiles par la suite. Afin que nos driver fonctionnent, il nous faut amener sur la machine un certificat que le dropper se chargera d'enregistrer dans le store Windows. C'est avec ce certificat que nous avons signé nos driver. Une fois cela fait, nous pouvons installer correctement les drivers.

Nous avons décidé d'installer tous nos fichiers dans le dossier C:\Windows\System32\ServiceUtilities. Ainsi dans ce dossier apparaîtra le fichier de service du driver kernel permettant de cacher le processus, l'exécutable permettant de communiquer avec le C2 et réalisé les tâches nécessaires sur la machine, ainsi que le fichier de service du minifilter driver permettant de cacher l'entièreté de ce dossier.

Nous utilisons le Service Control (sc) afin de configurer les services nécessaires pour nos driver.

Les 3 fichiers utilisés sont donc les suivants : Spoolfool.exe (exécutable permettant d'exploiter la faille), AddUser.dll (qui permet simplement de lancer le dropper) et dropper.exe (qui télécharge le package rootkit et l'installe)

## 3. Package Rootkit

Le package rootkit est la charge utile de notre exemple d'attaque utilisée pour la sensibilisation. Ce package se résume à 3 composants :

- Un driver kernel permettant de cacher des processus (abandonné)
- Un driver kernel de type minifilter permettant de cacher des fichiers
- Une application userland effectuant les communications avec le serveur Command&Control

### Cacher des processus

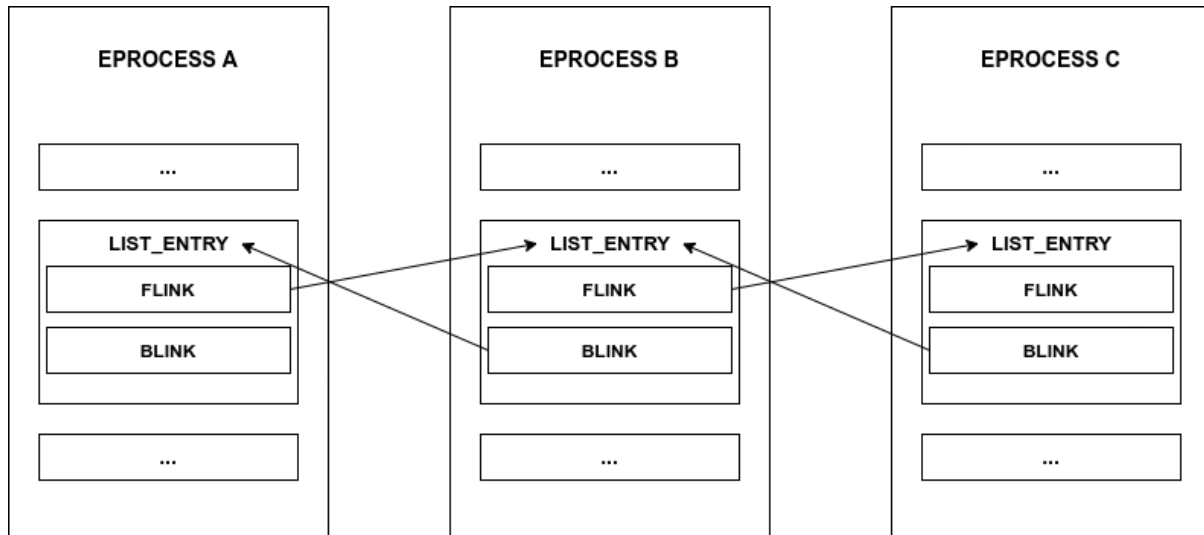
Nous présentons ici les travaux que nous avons réalisé sur la technique DKOM pour Direct Kernel Object Modification. Cependant, nous avons fait le choix de ne pas l'intégrer dans la solution finale. Nous reviendrons plus tard sur les motivations de ce choix.

Un driver kernel nous permet de cacher des processus. Ainsi, les processus cachés n'apparaîtront plus dans le gestionnaire des tâches. Pour ce faire, notre driver utilise aujourd'hui la méthode DKOM (Direct Kernel Object Manipulation). Cela pourra évoluer dans les semaines à venir. Une vidéo démontrant le fonctionnement du driver est disponible dans le rendu.

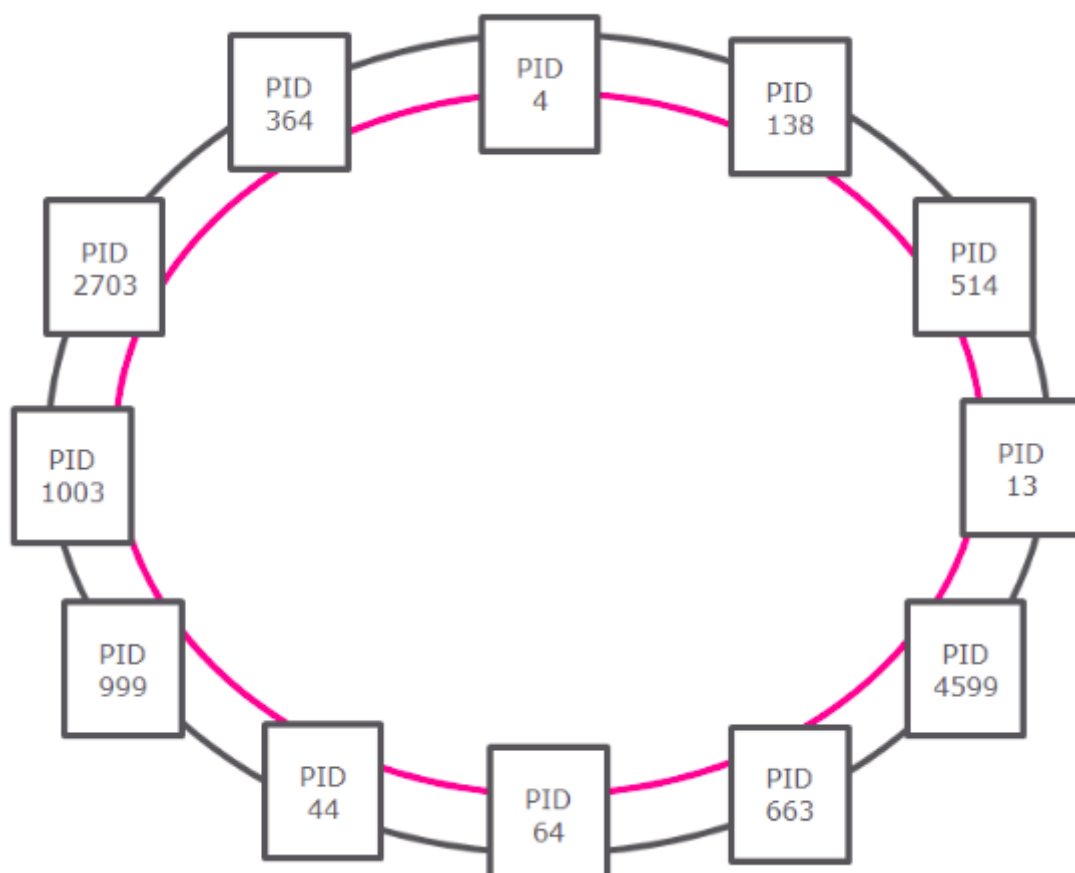
Avant de pouvoir cacher un processus, il est important de connaître la représentation d'un processus au niveau du noyau Windows. Chaque processus est représenté par une structure de type **EPROCESS**. Chaque version de Windows (1809, 1903, 20H2, ...) dispose de sa propre structure **EPROCESS**, les données stockées dans cette structure dépendent de cette

version de Windows. Toutes les structures **EPROCESS** correspondant aux processus lancés, sont liées entre elles en utilisant une liste doublement chaînée. Chaque **EPROCESS** dispose d'une structure **LIST\_ENTRY**. Cette structure **LIST\_ENTRY** indique deux valeurs :

- La structure **LIST\_ENTRY** de la structure **EPROCESS** suivante (variable **FLINK**)
- La structure **LIST\_ENTRY** de la structure **EPROCESS** précédente (variable **BLINK**)

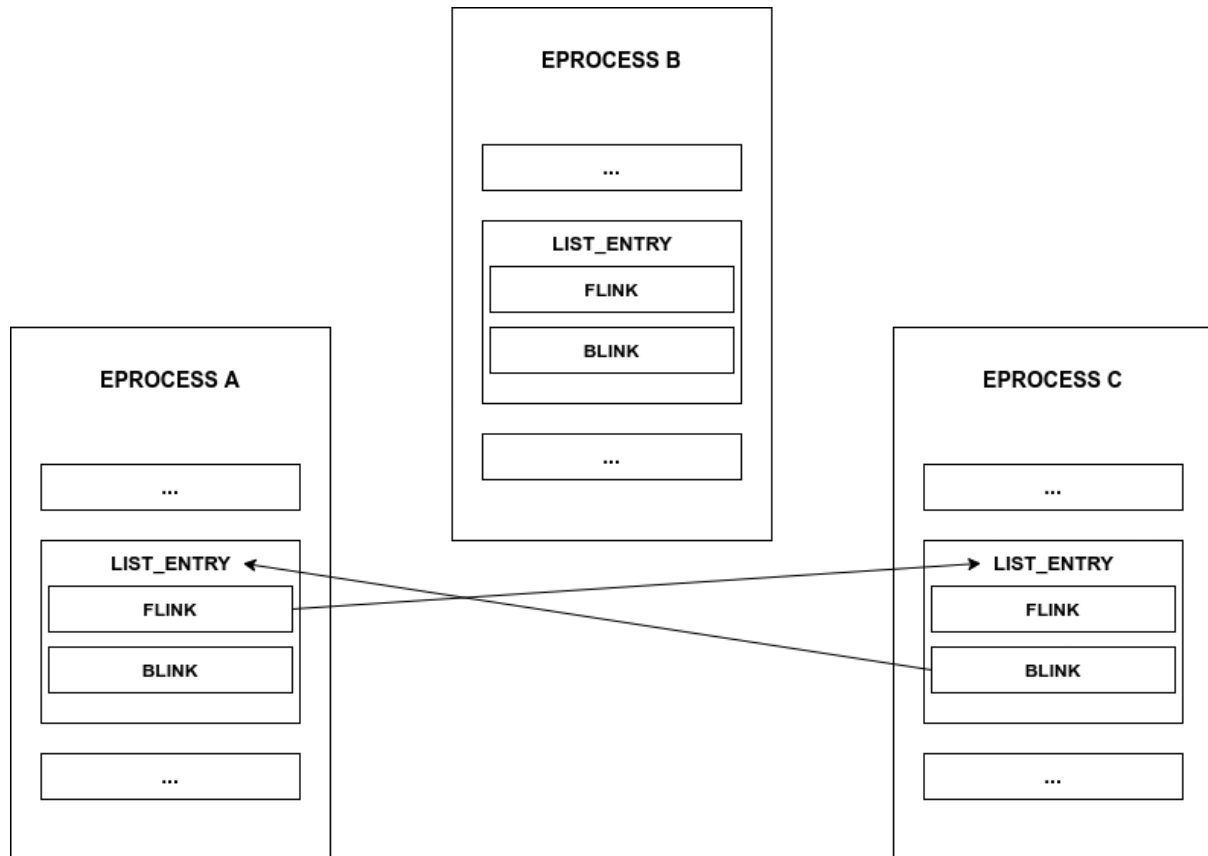


L'ensemble des structures **EPROCESS** peut donc être représenté de la manière suivante :

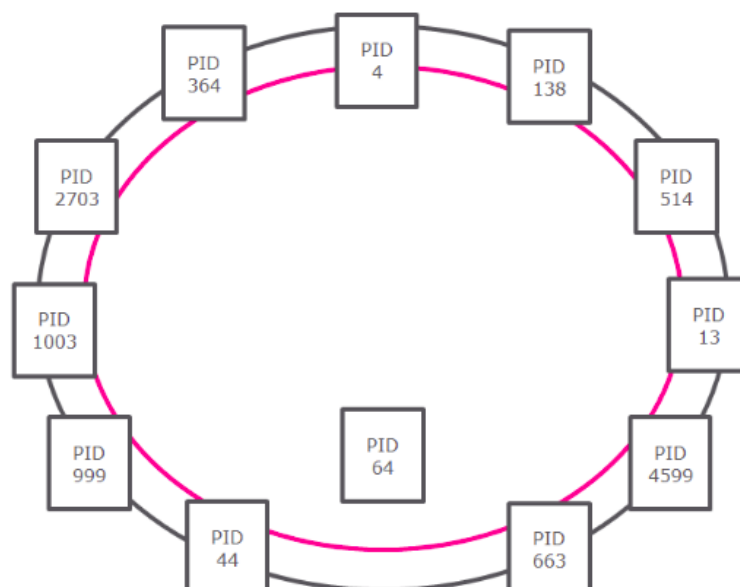




Lister les processus lancés sur le système revient à parcourir l'ensemble des structures **EPROCESS**. Pour cacher un processus, il suffira donc de sortir de la "boucle" la structure du processus correspondant. Ainsi, nous modifierons les **LIST\_ENTRY** des deux processus liés à notre processus à cacher pour ne plus les faire pointer vers celui à cacher, mais l'un vers l'autre. Cela peut être présenté de la manière suivante :



Ainsi, la structure **EPROCESS** correspondant à notre processus à cacher ne sera plus parcouru lors de la recherche de processus. Cela est représenté comme suit, la structure **EPROCESS B** correspond au processus avec le PID 64 :



Aujourd'hui, le driver permet de cacher les processus appelant le driver. Le fait de terminer un processus caché fait crasher le système. Nous avons donc ajouté une option permettant de réinsérer notre processus dans la boucle pour que celui-ci puisse se terminer. Or, lorsque celui-ci se termine en étant visible et en ayant été caché, le système plante. D'après nos recherches, il semble que cela soit dû à PatchGuard, un mécanisme de protection du noyau Windows. C'est pourquoi nous avons décidé de retirer ce driver de la solution. Nous reviendrons sur PatchGuard dans le volet détection de ce document.

## Autres techniques de dissimulation des processus

Aujourd'hui, d'autres techniques existent afin de dissimuler un thread d'exécution, plus simple à mettre en place et requérant moins de privilèges que la technique DKOM.

### **Injection de Processus**

Certains malwares utilisent l'injection de processus afin de dissimuler leur thread d'exécution dans la mémoire d'un autre processus. L'API de Windows permet de réaliser cela très simplement.

### **Injection de DLL**

D'autres malwares utilisent simplement l'injection de DLL en forçant les processus à utiliser leur DLL à la place d'une DLL légitime. Cela leur permet d'exécuter du code dans le thread d'exécution d'autres processus.

Il existe également beaucoup d'autres techniques. Ce papier scientifique en présente beaucoup : [https://www.jstage.jst.go.jp/article/ipsijip/25/0/25\\_866/pdf](https://www.jstage.jst.go.jp/article/ipsijip/25/0/25_866/pdf)

### **Import Address Table-Hooking**

L'import address table permet à un processus de récupérer les adresses des fonctions qu'il importe. L'idée derrière le hooking de l'IAT est de modifier une des adresses afin de faire exécuter par le processus la fonction malveillante que l'attaquant veut faire exécuter.

### **Inline Function Patching**

Cette technique est une amélioration de la technique précédente, où l'adresse malicieuse qui remplace l'adresse légitime est dans un tout autre espace mémoire que la DLL. C'est donc très voyant si on observe l'adresse qui a été modifiée. Ici, le principe est d'éditer directement le code de la DLL afin de remplacer les premiers octets d'une fonction par une instruction jump qui saute directement vers le code malicieux. En procédant de cette façon, l'adresse n'est donc pas modifiée.

Ces solutions présentent plusieurs avantages par rapport à la nôtre :

- Pas besoin de privilèges particuliers
- Pas d'ajout de processus (pour la DLL injection)

Nous aurions pu implémenter l'une de ces solutions. Malheureusement, par manque de moyen, aucune solution n'a pu être mise en place. Néanmoins, ces techniques restent très utilisées, surtout pour bypass les antivirus. C'est pourquoi nous les présentons ici.

La matrice MITRE ATT&CK définit 12 sous-techniques sous l'injection de processus, que nous ne présenterons pas ici. Cela montre à quel point cette technique est développée et que

plusieurs variantes existent. Nous venons juste de présenter les plus populaires. Ces techniques présentent des différences subtiles afin d'arriver au même but : l'exécution de code dans d'autres processus afin de dissimuler la charge malveillante.

Également, d'autres techniques en mode kernel existent, en voici deux :

### **SysENTER-Hooking**

Voici une première technique en mode noyau. Lorsqu'un appel vers l'API windows est réalisé et que du code noyau est exécuté, 3 registres permettent de traiter l'exécution :

- IA32\_SYSENTER\_CS (segment de code)
- IA32\_SYSENTER\_EIP (instruction suivante)
- IA32\_SYSENTER\_ESP (la pile)

Le principe ici est de modifier le registre IA32\_SYSENTER\_EIP afin de faire pointer l'instruction suivante vers le code malicieux de l'attaquant. Il faut bien évidemment remplacer l'instruction par la suite pour que la modification soit invisible.

### **SSDT-Hooking**

La System Service Descriptor Table (SSDT) est comparable à l'IAT mais pour le mode noyau. Comme l'IAT il est possible de hook cette table afin de faire exécuter son propre code. Il faut préalablement les droits ring 0.

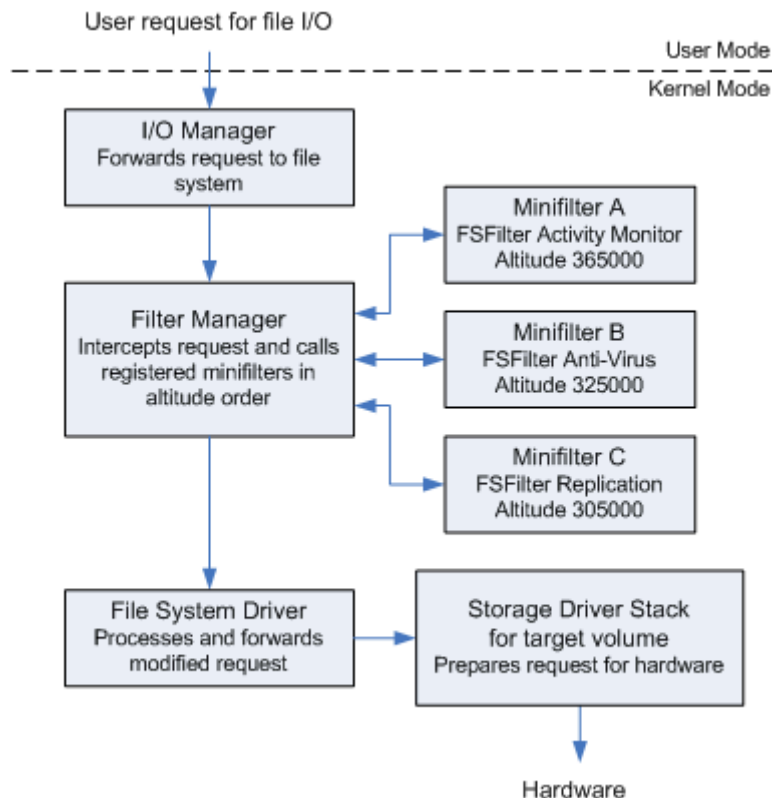
### Cacher des fichiers

Windows propose un nouveau type de "file system filter driver", les "minifilters". Ceux-ci viennent remplacer l'ancien type "legacy file system filter model", apportant une simplicité de développement ainsi qu'une meilleure qualité et robustesse par rapport aux *file system filter driver*.

Fltmgr, le "filter manager" s'occupe de gérer les minifilters qui s'enregistrent sur le système de fichiers. Les minifilters s'enregistrent indirectement, et enregistrent des fonctions de callback sur des requêtes d'entrée sortie envoyées par le système d'exploitation. Ils peuvent enregistrer deux types de fonction :

- Preoperation callback function
- PostOperation callback function
- 

Voici un schéma pour illustrer le système de requêtes d'entrée sortie :



Le schéma démarre par une requête d'entrée sortie. La requête arrive ensuite au filter manager qui va s'occuper d'appeler les fonctions de callback preoperation des différents minifilters qui en auront enregistré sur le type de requête demandé. L'ordre dans lequel ces fonctions seront appelées dépend de l'altitude du minifilter. Lors de son enregistrement, le minifilter déclare une altitude qui déterminera l'ordre de son appel par rapport aux autres minifilters. Deux minifilters ne peuvent avoir la même altitude.

Une fois toutes les fonctions de preoperations callback appelées, le filter manager envoie la requête au "file system driver" qui s'occupera de réaliser la requête sur le hardware directement.

Lors de la remontée, le filter manager appellera, dans l'ordre décroissant cette fois-ci, les fonctions de postoperation callback des différents minifilters.

Les requêtes sont divisées en deux "function code" : le "major function code" qui définit le type de requête demandé et le "minor function code" qui définit un sous type de cette même requête. Pour notre minifilter, le type de requête qui nous intéresse est "IRP\_MJ\_DIRECTORY\_CONTROL" avec le "minor function code" "IRP\_MN\_QUERY\_DIRECTORY". Cette requête est envoyée pour demander la liste des fichiers et sous dossiers d'un répertoire dans Windows, par exemple lors de la navigation dans le système de fichiers via l'explorateur de fichiers ou encore un appel à la fonction dir via cmd sur un dossier.

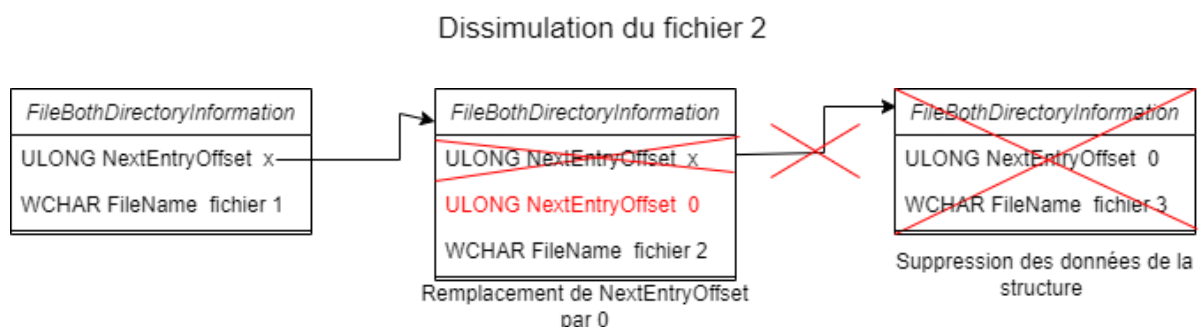
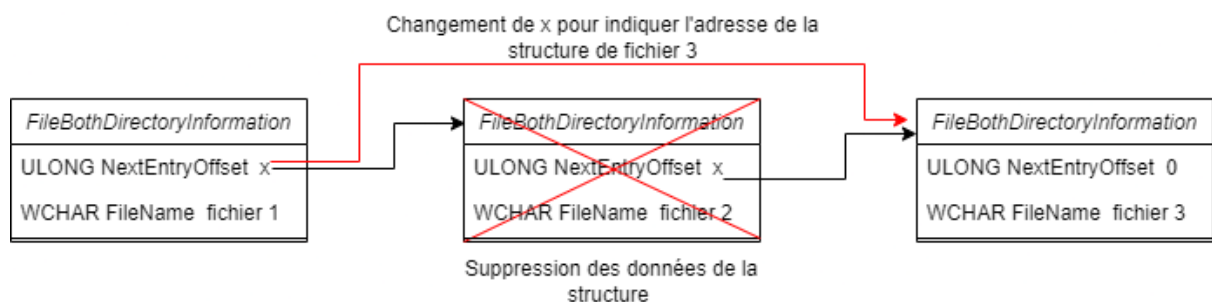
Cette requête retournera une liste de structure. Il y a en tout 9 types de structures retournés, une structure représentant un fichier à l'intérieur du dossier :

- FileBothDirectoryInformation
- FileDirectoryInformation
- FileFullDirectoryInformation
- FileIdBothDirectoryInformation
- FileIdFullDirectoryInformation
- FileNamesInformation
- FileObjectIdInformation
- FileQuotaInformation
- FileReparsePointInformation

Certaines sont plus complètes que d'autres, permettant de renvoyer plus d'informations sur les fichiers en question. Nous nous intéresserons seulement aux 6 premières car elles représentent les requêtes d'information sur les fichiers à l'intérieur d'un dossier, les 3 dernières étant différents et répondant à un autre besoin.

Lors de la remontée de la requête, un buffer est renvoyé contenant la liste des structures représentant les différents fichiers contenus dans le dossier. Le parcours de la liste, contrairement à la liste des structures EPROCESS expliquée plus haut, se fait au moyen d'un offset indiquant le début de la structure suivante.

Ainsi pour notre minifilter, nous avons enregistré une fonction de postoperation callback sur le major function code "IRP\_MJ\_DIRECTORY\_CONTROL" et le minor function code "IRP\_MN\_QUERY\_DIRECTORY". Nous vérifions le type de structure demandé pour ne traiter que les 6 premières présentées plus haut. Dans notre fonction de postoperation callback, le buffer contenant la liste des structures nous est envoyé. Il nous suffit de parcourir cette liste à la recherche des fichiers que l'on souhaite cacher et de supprimer les données à cette adresse dans le buffer. Deux cas de figure spécifiques se présentent à nous : lorsque le fichier se trouve au milieu de la structure et lorsqu'il se trouve en fin de structure. Des schémas seront plus parlants. Nous indiquerons seulement les champs des structures qui nous intéressent :



Dissimulation du fichier 3

Il est à noter que le fichier ne peut jamais se retrouver au début de la liste, car il y aura

toujours les dossiers symboliques “.” et “..” qui occuperont les deux premières structures de la liste. C’est pour la même raison que le buffer renvoyé contenant la liste de structure ne sera jamais vide, même pour un dossier vide.

Ce système fonctionne, il permet de bien cacher les fichiers à tous les processus du système. Seulement, cela pourrait nous poser problème, notamment lorsque notre application Userland ou tout autre processus système aurait besoin d’accéder aux fichiers dissimulés pour fonctionner. Cela pourrait créer des crashes du noyau système. Nous avons donc pris la décision de dissimuler seulement les fichiers lorsque le processus réalisant la requête est soit l’explorateur de fichiers, soit le cmd. Cela est rendu possible à l’aide de la fonction `PsGetCurrentProcess()` qui nous renvoie la structure `EPROCESS` du processus courant, donc celui qui a initié la requête. Il nous suffit alors de récupérer le nom du processus dans un des champs de cette structure afin de vérifier quel est le processus ayant initié la requête. Cela nous a permis d’éviter de nombreux problèmes et de rendre notre minifilter robuste et stable, au prix d’une dissimulation amoindrie. Néanmoins, nous avons jugé que cette dernière serait suffisante et efficace.

Dans le rendu technique, le driver sera nommé Winflt. Comme c’est un minifilter, nous utilisons le fichier Winflt.inf pour l’installer. Le fichier Winflt.sys sera le fichier utilisé pour le service, qui sera installé ici : `C:\Windows\System32\drivers\Winflt.sys`.

L’écriture d’un driver kernel prend du temps et nécessite des connaissances assez poussées ce qui demande un niveau technique assez élevé. Les informations de la documentation Windows sont cependant bien fournies mais difficiles à naviguer lorsque l’on travaille à un aussi bas niveau.

## Application Userland

### Introduction :

Pour faire le lien entre la machine infectée et notre serveur command and control, nous avons développé une application userland. Cette application nommée « winpoolmsc.exe » est une application .Net permettant d’interagir avec le driver « wintbk.sys » ainsi que le server C2 et la machine cible. Cette application devait à la base s’interfacer avec le driver “wintbk.sys”, cependant, dû à des problèmes provenant de PatchGuard, nous avons dû abandonner cette possibilité. Nous expliquerons quand même l’interfaçage.

### Présentation du programme :

Le programme est écrit en .Net Core 6, la dernière version du langage. Nous avons choisi de développer ce dernier en C# pour plusieurs raisons, la première étant la facilité à écrire le code car étant familier avec ce dernier. De plus, le langage étant développé par Microsoft, nous pensions qu’il serait facile d’interagir avec l’OS et de faire des appels systèmes. Enfin, le langage nous permet de n’avoir qu’un seul exécutable portable une fois compilé.

### Dépendances :

Le code utilise et embarque différentes bibliothèques Nuget :

- Newtonsoft.Json
- Flurl.Http

### Interface avec le driver "wintbk" :

Le programme embarque toujours les fonctions lui permettant de s'interfacer avec le driver "wintbk" permettant de cacher des processus. Le code commence par importer les dll permettant d'interagir avec les drivers Windows. On y importe les fonctions CreateFile et WriteFile de la dll Kernel32.dll.

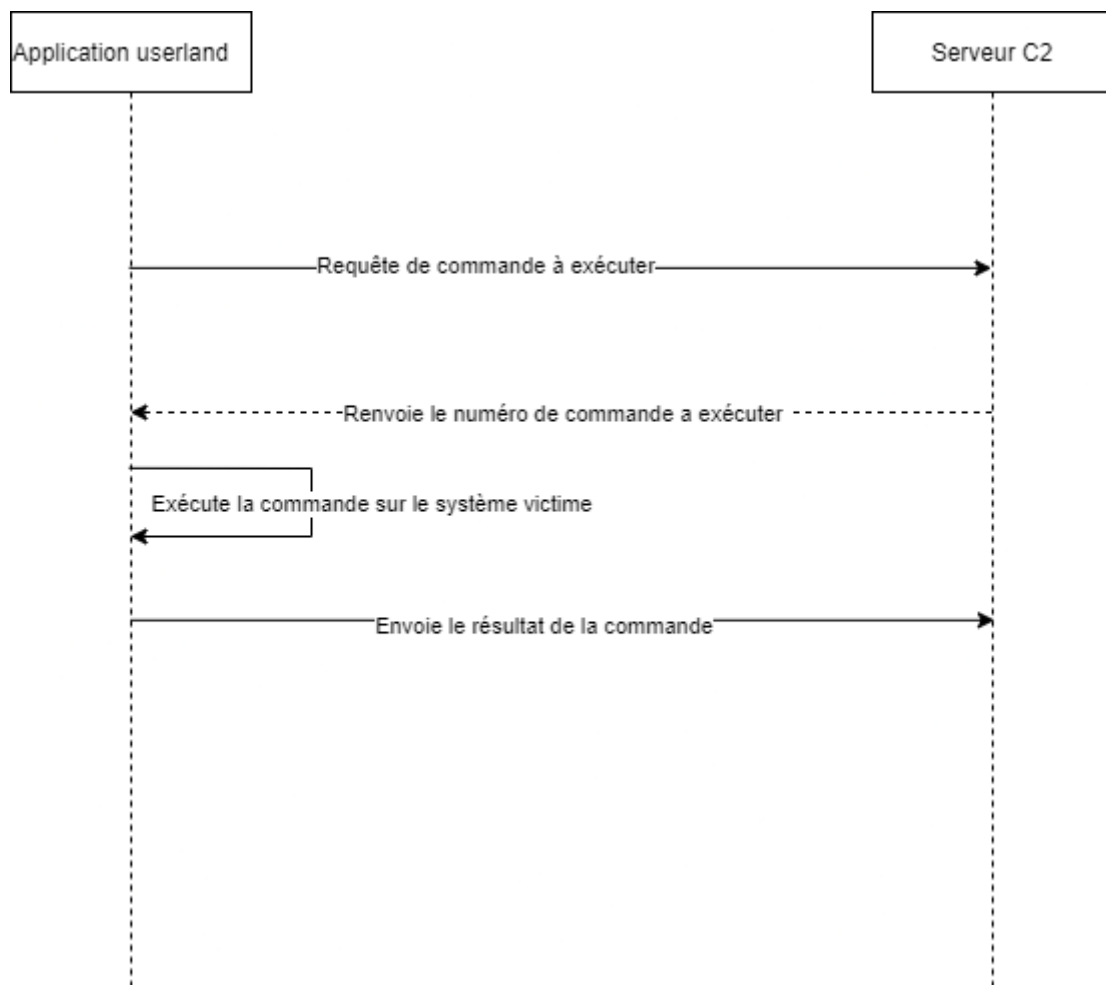
```
IntPtr hFile = CreateFile("\\\\.\\DKOM_Driver", 0x40000000, 0x00000001 | 0x00000002, IntPtr.Zero, 3,
uint bwritten;
NativeOverlapped lp = new NativeOverlapped();
SetConsoleCtrlHandler(new HandlerRoutine(ShutdownCheck), true);
WriteFile(hFile, new System.Text.StringBuilder(""), 4, out bwritten, ref lp);
```

En envoyant un 0 au driver, celui-ci récupère le pid du processus et le cache. Nous avons aussi essayé de récupérer l'évènement d'extinction de la machine, pour réafficher le processus avant de quitter, pour que le système ferme proprement l'application. Le problème étant que l'outil de Windows PatchGuard repérait la manipulation et déclenchait un BSOD (Blue Screen Of Death). Nous avons donc décidé d'abandonner cette protection supplémentaire.

### Fonctionnement :

Fonctionnement global :

Le fonctionnement de cette application est assez simple malgré quelques problèmes que nous avons rencontrés et qui seront expliqués dans la suite de ce document. Il se base sur une boucle infinie qui ponctuellement envoie une requête C2 pour savoir s'il doit exécuter une commande sur la machine victime (voir les commandes dans le package command & control). Le temps d'attente entre différentes commandes est généré aléatoirement entre 10 et 60 secondes.

















#### Persistence :

Pour se rendre persistant, le programme commence par vérifier si une clé de registre à son nom existe au chemin suivant : « Ordinateur\HKEY\_CURRENT\_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run ». Si la clé de registre existe, l'application la force à la valeur "true", sinon, elle l'a créée et la met à "true". Cette clé de registre permet de lancer le programme au lancement de la machine, permettant de se relancer à chaque démarrage du système.

Le problème avec cette méthode, est que le programme apparaît dans l'onglet "Démarrage" du gestionnaire de tâches. Nous avons donc pensé à utiliser la méthode par PowerShell.



| Gestionnaire des tâches  |                          |           |                 |  |
|--|--------------------------|-----------|-----------------|--|
| Fichier Options Affichage  |                          |           |                 |  |
| Processus Performance Historique des applications Démarrage Utilisateurs Détails Services                          |                          |           |                 |  |
| Dernier  |                          |           |                 |  |
| Nom  | Éditeur                  | Statut    | Impact du dé... |  |
|  Microsoft OneDrive               | Microsoft Corporation    | Désactivé | Aucun           |  |
|  Microsoft Teams                  | Microsoft Corporation    | Désactivé | Aucun           |  |
|  Program                          |                          | Activé    | Non mesuré      |  |
|  Razer Synapse 3                  | Razer Inc.               | Activé    | Haut            |  |
|  Skype                            | Skype                    | Désactivé | Aucun           |  |
|  Spotify                          | Spotify Ltd              | Activé    | Haut            |  |
|  Steam                            | Valve Corporation        | Activé    | Bas             |  |
|  TabletDriver                     | Graphic Tablet Compan... | Activé    | Bas             |  |
|  Update                           | GitHub                   | Désactivé | Aucun           |  |
|  Vanguard tray notification.     | Riot Games, Inc.         | Activé    | Non mesuré      |  |
|  wallpaper32.exe                |                          | Activé    | Haut            |  |
|  Windows Security notificati... | Microsoft Corporation    | Activé    | Moyen           |  |
|  Windows Terminal               | Microsoft Corporation    | Désactivé | Aucun           |  |
|  winpoolmsc                     | winpoolmsc               | Activé    | Non mesuré      |  |

Méthode de persistance secondaire :

Durant nos recherches, nous avons trouvé une autre méthode de persistance basée sur PowerShell. Ayant découvert cette méthode vers la fin du projet, nous n'avons pas pu l'implémenter à temps mais nous souhaitons quand même l'expliquer. Cette méthode est utilisée par le dropper "Colibri Loader". Pour cela, l'application est téléchargée via un document malicieux dans le chemin suivant :

"%APPDATA%\Local\Microsoft\WindowsApps" et renomme le binaire "Get-Variable.exe". Une fois le binaire installé, une tâche planifiée va ensuite appeler une fenêtre cachée PowerShell. PowerShell va donc charger différents programmes dont un appelé "Get-Variable.exe". Windows commence par regarder dans le dossier WindowsApps et va donc trouver le programme malicieux portant le nom voulu et va le charger au lieu du binaire légitime.

On peut retrouver une description plus précise sur le site de Malwarebytes : <https://blog.malwarebytes.com/threat-intelligence/2022/04/colibri-loader-combines-task-scheduler-and-powershell-in-clever-persistence-technique/>

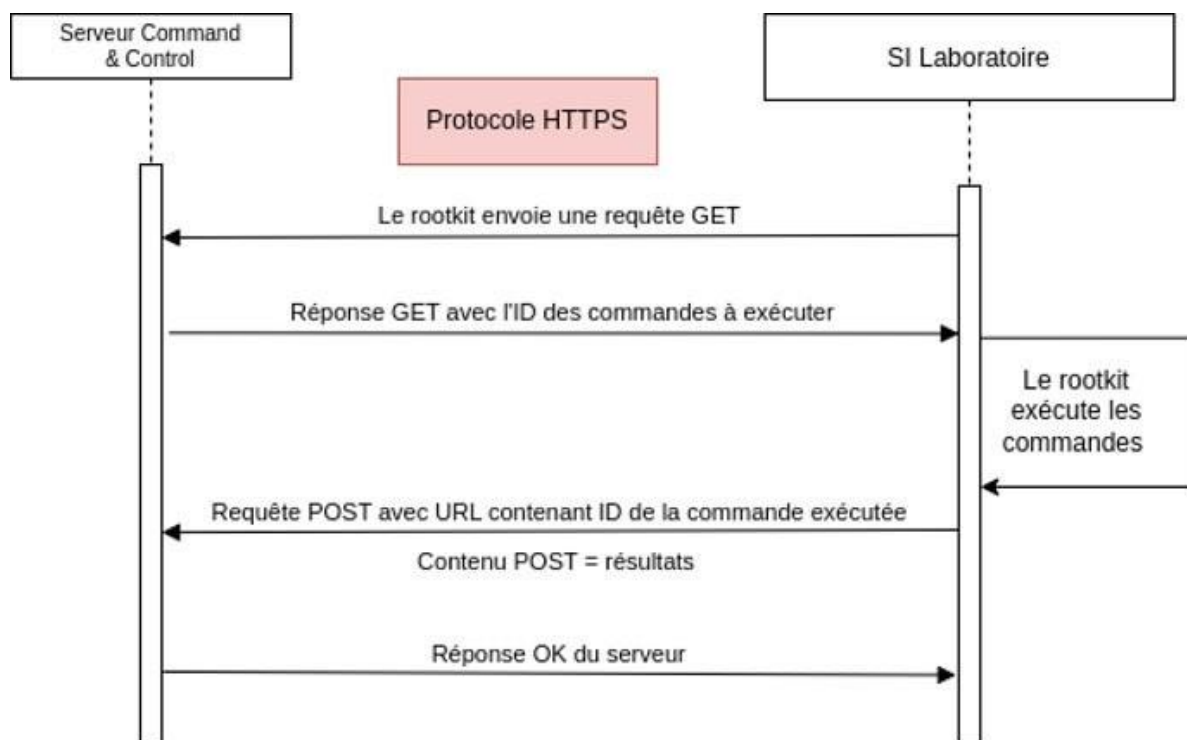
D'autres techniques de persistance peuvent être trouvées sur le web. Par exemple, le site suivant répertorie plusieurs techniques de persistance Windows :

## 4. Package Command & Control

Le package Command & Control correspond à l'application serveur HTTPS avec laquelle l'application userland du package rootkit communiquera.

Notre choix s'est porté sur la plateforme NodeJS utilisant le langage de programmation Javascript. NodeJS nous permet de créer une application rapidement et facilement.

La communication entre le client et le serveur C2 fonctionnera de la manière suivante :



Notre application serveur HTTPS devra donc fournir les informations de commande au rootkit, puis récupérer et stocker les résultats de ces commandes reçues depuis le client (rootkit).

Aujourd'hui, nous avons implémenté 3 commandes côté serveur :

- Commande 1 : Informations génériques (Nom de la machine, Adresse MAC, Version de Windows, Nom de domaine, Adresse IP, état du driver, taille occupée du disque)
- Commande 2 : Chercher tous les fichiers contenant un certain mot-clé passé en paramètre
- Commande 3 : Transférer le fichier spécifié en paramètre du client vers le serveur

Ces commandes sont passées en réponse à la requête GET. Du code correspondant à ces commandes est exécuté par le rootkit. Le résultat produit par le code précédemment cité sera ensuite envoyé au serveur dans une requête POST. Les résultats sont envoyés au format texte pour les commandes 1 et 2. En revanche, pour les résultats de la commande 3, le contenu est encodé en Base64. En effet, les fichiers peuvent contenir des données binaires, il est préférable de les encoder pour ne gérer que le transfert de données de type texte. Les résultats de cette commande seront donc décodés par notre serveur C2.

Enfin, les différents résultats reçus par notre application serveur seront stockés dans des fichiers situés dans des dossiers correspondant à la commande. Le nom des fichiers indiquera la date et l'heure de création de ces fichiers. Cela nous permet de garder un historique de toutes les communications entre notre serveur C2 et le rootkit. Toutes ces données récoltées pourront faire l'objet d'une analyse future.

## 4. Se défendre face à ces attaques

Cette partie du document sera découpée en 3 catégories :

- Prévention
- Détection
- Réaction

Nous aborderons donc ces 3 phases de défenses face aux rootkits en présentant toutes les recherches que nous avons réalisées. Nous présenterons donc les solutions de défense face à ces attaques, ainsi que les limites de ces solutions de défense.

### 1. Prévention

Ce que Microsoft propose

#### [La politique de signature de Microsoft](#)

On remarque que l'évolution de la politique de signature de code du mode kernel est caractérisée par un durcissement continu de la sécurité.

[Le contexte : Kernel mode vs User mode](#)

Pour empêcher les applications utilisateurs d'accéder ou de modifier des données critiques du système d'exploitation, Windows utilise seulement 2 niveaux de privilèges : le niveau 0 (ou ring 0) pour le *kernel mode* et le niveau 3 (ou ring 3) pour le *user mode*.

Le code des applications utilisateurs est exécuté en *user mode*, tandis que le code du système d'exploitation, comme les services systèmes et les *device drivers*, est exécuté en *kernel mode*. Une fois en kernel mode, le code du système d'exploitation et des drivers ont un accès sans restriction au matériel et à la mémoire du système, ils peuvent alors contourner la sécurité de Windows pour accéder aux objets.

Il est donc crucial que les composants, comme les drivers, qui s'exécutent en mode kernel soient minutieusement codés et testés pour s'assurer qu'ils ne mettent pas en danger la stabilité du système. Ce manque de « protection » renforce le besoin de faire preuve de vigilance lorsqu'on installe un driver, surtout s'il n'est pas signé, car une fois en mode kernel, ce driver a un accès sans restriction à toutes les données du système d'exploitation.

[Windows 2000](#)

Le mécanisme de signature des drivers est introduit avec Windows 2000 qui prévient l'utilisateur si une tentative est faite d'installer un driver plug-and-play non signé. Un nouveau mécanisme appelé « Driver Verifier » aide aussi les développeurs de drivers à trouver des bugs.

## Windows 8.1

Sur les versions 64 bit et les versions ARM de Windows 8.1, une politique de signature en mode kernel (*kernel mode code-signing policy – KMCS*) impose que tous les drivers (pas seulement les drivers plug and play) doivent être signés avec une clé assignée par une des autorités majeures de certification. Un utilisateur ne peut forcer l'installation d'un driver non signé, à moins que le système soit placé en mode test.

## Windows 10 et le “ July Anniversary update”

Windows 10 sort le 29 juillet 2015. Avec la version *Anniversary Update* qui sort en juillet 2016 - version 1607, Microsoft durcit encore sa politique.

A partir de cette version 1607, tous les nouveaux drivers Windows 10 doivent être signés par une autorité de certification avec un certificat SHA-2 Extended Validation (*EV Certificate*). Une fois signé avec un certificat EV, le driver doit être soumis à travers le portail SysDev pour recevoir une « signature d'attestation » (*attestation signing*), qui permettra au driver de recevoir une signature de Microsoft.

Ainsi, le kernel ne signera que ces drivers Windows 10 signés par Microsoft, avec quelques exceptions notables :

- Le Secure Boot du BIOS est off (c'est le mode Test)
- Le PC a été mis jour à partir d'une version antérieure de Windows vers Windows 10, version 1607
- Le driver a été signé avant le 29 Juillet 2015 avec un “trusted cross-certificate” 1 (*Drivers was signed with an end-entity certificate issued prior to July 29th 2015 that chains to a supported cross-signed CA*)

## Windows Server 2016

La politique se durcit encore, la seule « signature d'attestation » ne suffit plus. En plus de la nécessité pour les développeurs d'être identifiés avec des *EV certificate*, les nouveaux drivers kernel doivent passer les tests de compatibilité du Hardware Lab Kit pour obtenir une certification totale (*a full cert pass*) Cela ne s'applique pas aux drivers qui sont exécutés dans le user space.

### 2021 : l'expiration des certificats croisés et le nouveau « Vulnerable and Malicious Driver Reporting Center »

Depuis 2021, [seul Microsoft](#) peut fournir les signatures des drivers kernel. La majorité des certificats croisés (certificat émis par des autorités de certification autre que Microsoft, ancien fonctionnement) ont expiré en [juillet 2021](#). Il n'est donc plus possible d'utiliser des certificats croisés expirés pour créer de nouvelles signatures numériques en mode kernel.

De plus en décembre 2021, Microsoft [annonce](#) la création d'un « vulnerable driver and malicious reporting center ». Il sera possible de soumettre le code des drivers suspects à travers ce [nouveau portail](#) pour faire analyser le code par Microsoft qui fera le suivi avec les développeurs. Les drivers non-patchés qui auront toujours des vulnérabilités [seront automatiquement bloqués](#) dans l'écosystème utilisant Microsoft Defender for Endpoint attack surface reduction et Microsoft Windows Defender Application Control (WDAC).

Cette politique de blacklist sera mise à jour automatiquement pour les systèmes qui ont activé Hypervisor-Protected Code Integrity (HPCI). Il est aussi possible de créer ou d'appliquer des listes de drivers vulnérables localement à travers des WDAC policies.

### Une politique imparfaite qui permet des attaques basées sur des drivers vulnérables

Malgré ce durcissement, de nombreux articles font référence à l'utilisation par les attaquants de drivers kernel légitimes mais vulnérables qui leur permettent de contourner le **driver signature enforcement (DES)** et d'exécuter du code malicieux sur le système de la victime.

Cette technique d'attaque est appelée « *Bring Your Own Vulnerable Driver* ». Cette attaque est répertoriée dans [le framework Mitre](#). Le driver légitime et signé - venant de fabricants de confiance - a une vulnérabilité que les attaquants exploitent pour avoir accès au « ring 0 », ce qui leur permet de désactiver les mécanismes de sécurité de Windows, ou bien d'exécuter du code en mode kernel en installant leur propre driver kernel malicieux et non signé.

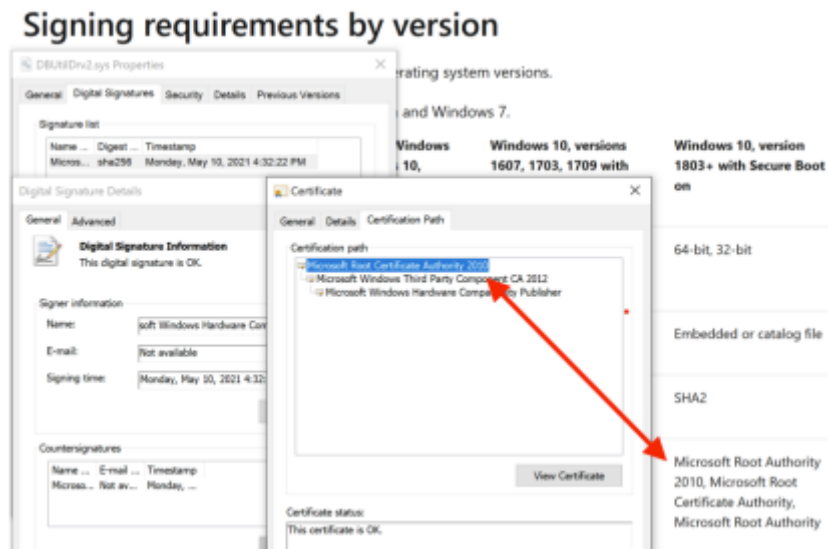
Nous avons identifié trois attaques récentes et significatives utilisant des drivers légitimes :

[Dell drivers + CVE-2021-21551](#) : utiliser des vulnérabilités liées aux drivers Dell pour contourner la protection LSA

Certains drivers de Dell vulnérables à la CVE-2021-21551 peuvent être utilisés pour exécuter du code en kernel mode. Des chercheurs de Rapid7 [en décembre 2021](#) ont montré comment il est possible d'utiliser ces drivers pour contourner la protection LSA. Avec l'accès en ring 0

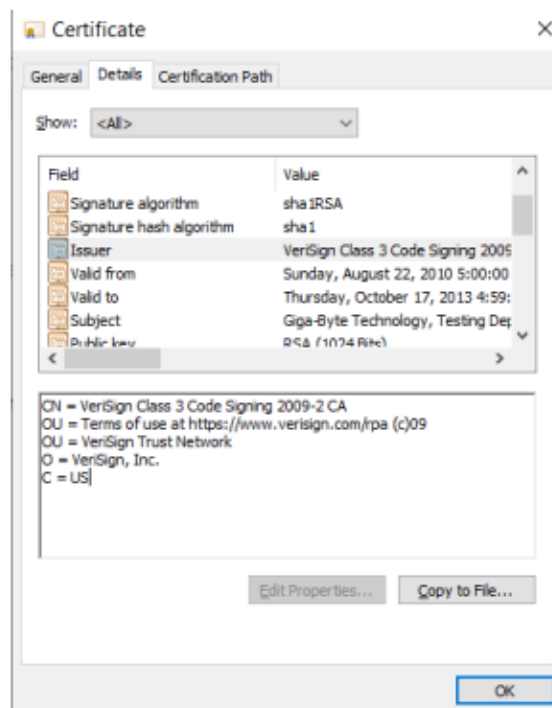
rendu possible grâce à la vulnérabilité de 3 drivers Dell, un attaquant peut atteindre le lsass.exe EPROCESS struct et dump la mémoire de lsass.exe

Ces drivers Dell sont particulièrement vulnérables car ils sont justement compatibles avec la politique de signature de Microsoft et les chances pour que les drivers d'un fabricant comme Dell soient bloqués sont surement faibles. Ces drivers ne sont en tous cas pas dans la driver block list de Microsoft.



Le ransomware Robinhood

[Des chercheurs de Sophos](#) ont montré que des attaques par ransomwares ont été menées en utilisant un driver signé et légitime – Gigabyte driver - pour désactiver le *driver signature enforcement* de Windows et installer leur propre driver non signé et malicieux afin de chiffrer tous les fichiers de la victime.



Le nouveau Vulnerable and Malicious Driver Reporting Center aurait pu empêcher cette attaque si celui-ci avait été actif au moment de l'attaque et que le driver Gigabyte avait été catégorisé comme vulnérable.

Netfilter driver : un driver malicieux signé par Microsoft

En juin 2021, des chercheurs de [la compagnie G Data](#) ont découvert qu'un driver signé par Microsoft – Netfilter - était en fait un rootkit qui permettait de rediriger les communications des victimes vers un serveur ayant une adresse IP basée en Chine. Dans ce cas, la faille viendrait plutôt du processus de certification de Microsoft qui est conçu justement pour éviter ce genre d'attaques.

## Ce que vous pouvez faire

### La sensibilisation des utilisateurs

Pour prévenir ce type d'attaque, il est important de former les employés de l'entreprise. Dans la plupart des attaques, une faille humaine est exploitée (via du spearphishing par exemple). La sensibilisation des employés est une composante essentielle pour éviter des attaques de type cyber.

### La préparation

La prévention passe aussi par une bonne préparation ainsi qu'une connaissance poussée du système d'information. Un suivi du parc informatique doit être mis en place, notamment au niveau des vulnérabilités. Un plan de maintien en condition de sécurité (MCS) doit être appliqué de manière efficace. Des plans de continuité d'activité et de reprise d'activité peuvent être définis pour anticiper une future réponse à une attaque.

## **5. Détection**

### Patchguard

Patchguard est un utilitaire du système d'exploitation Windows. Il est également connu sous le nom de Kernel Patch Protection. Celui-ci monitore plusieurs structures de données dans le kernel afin de détecter une utilisation malveillante de ces structures. Si tel est le cas, Patchguard produira un blue screen of death (BSOD) sur la machine, empêchant le code malveillant de s'exécuter sans impunité.

Patchguard est la raison pour laquelle nous avons décidé de renoncer à la technique DKOM afin de manipuler les structures noyaux dans le but de cacher un processus. En effet celui-ci nous détectait à chaque fois en faisant planter la machine.

Patchguard n'est pas exempt de défauts. Plusieurs solutions de contournement ont été trouvées au fil des années, dont une très récente datant de début 2021 disponible à cette adresse : <https://github.com/kkent030315/NoPatchGuardCallback>. Mais il constitue tout de même une barrière solide face aux rootkits essayant de s'implémenter dans le kernel.



## Minifilter

Il existe plusieurs moyens de détecter un MiniFilter sur votre machine. Sachez d'abord que les MiniFilter drivers sont un type de driver proposé par Microsoft. N'importe qui peut en écrire un, et ils peuvent avoir des utilisations tout à fait légitimes. D'ailleurs, sur une installation neuve de Windows, certains seront déjà installés sur la machine.

Il existe une commande permettant de lister les Minifilter drivers installés sur votre machine : fltmc. Voici la sortie de cette commande :

| Nom de filtre | Nom. d'instn. | Altitude | Cadre |
|---------------|---------------|----------|-------|
| bindflt       | 1             | 409800   | 0     |
| WdFilter      | 9             | 328010   | 0     |
| storqosflt    | 0             | 244000   | 0     |
| wcifs         | 3             | 189900   | 0     |
| gameflt       | 5             | 189850   | 0     |
| CldFlt        | 3             | 180451   | 0     |
| FileCrypt     | 0             | 141100   | 0     |
| luafv         | 1             | 135000   | 0     |
| npsvcrtig     | 1             | 46000    | 0     |
| Wof           | 7             | 40700    | 0     |
| FileInfo      | 9             | 40500    | 0     |

On voit ici que plusieurs Minifilter sont installés sur la machine. Cette commande peut être un bon point de départ pour détecter un Minifilter malveillant. Par exemple, tous les MiniFilter affichés sont présents à cette adresse : <https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/allocated-altitudes>

Cette page regroupe les attributions d'altitude pour les Minifilters connus. L'altitude conditionne l'ordre d'appel des Minifilters. Evidemment un Minifilter malveillant tentera de se faire passer pour l'un d'entre eux, mais c'est toujours intéressant de pouvoir lister les MiniFilters installés sur la machine.

Bon à savoir, sur Windows 11, un nouveau type d'input/output request est disponible. Celle-ci se prénomme BypassIO. Elle permet de contourner toute la pile de MiniFilters afin d'envoyer la requête directement au système de fichier. Cette input/output request permet d'améliorer les performances de la requête en évitant de passer par toute la pile de MiniFilters. Mais elle pourrait également se trouver utile afin de scruter le système de fichier à la recherche de fichier potentiellement dissimuler par l'un des MiniFilters malveillants.

## Cacher des processus

### [Injection de processus / DLL Injection](#)

Plusieurs techniques peuvent être utilisé afin de détecter l'injection de processus. Monitorer l'API Windows pour remarquer les appels à certaines fonctions telles que CreateRemoteThread, VirtualAllocEx, SetThreadContext.



Egalement, regarder le processus parent d'un processus permet de détecter un comportement inhabituel. Certains processus sont toujours lancés par un processus spécifique. Si ce processus est lancé via un autre processus, alors le processus à peut-être été injecté.

Comparer les différences entre la structure PEB (Process Environment bloc) et la structure VAD (Virtual address descriptor). En effet lorsqu'un processus est injecté, sa VAD indiquera une adresse de début et de fin différentes de son PEB. Si tel est le cas, cela peut vouloir dire que le processus a été injecté.

Egalement, volatility permet avec la commande malfind, de détecter l'injection de processus.

### IAT Hooking

Pour détecter l'IAT hooking, il faut regarder les adresses contenues dans la table. Si l'une d'entre elles est assez éloigné des autres, alors c'est sûrement une adresse qui a été remplacée afin de pointer vers une fonction malicieuse.

### Inline function patching

Cette technique permet de bypass la détection de l'IAT hooking. Il faut alors regarder les premières instructions des fonctions de la DLL. Si celle-ci contiennent un jump assez loin dans le code, alors la technique a été utilisée. Parfois, l'instruction jump se trouve plus loin dans le code. La détection est alors plus compliquée, mais reste néanmoins possible en examinant tout le code de la fonction.

### SysENTER Hooking

Pour détecter ce type d'attaque, monitorer l'adresse contenue dans le registre IA32\_SYSENTER\_EIP est une solution. Si celle-ci n'est pas celle de la fonction qui est responsable des appels systèmes sur la version de l'OS, la conclusion n'est autre que cette valeur a été modifié.

### SSDT Hooking

Il est possible de détecter cette attaque de la même manière que pour l'IAT hooking ou Inline function patching. La seule différence avec ces deux techniques est que cette table est contenue dans le noyau de l'OS.

## Connaissance du parc informatique

Au sein d'une DSI, il est important d'assurer un suivi du parc informatique. Ce suivi doit être effectué sur tous les applicatifs déployés sur les machines. Une bonne connaissance du système d'information permet de détecter plus facilement des attaques, en identifiant les éléments inconnus et potentiellement malveillants. Nous pouvons alors parler d'analyse différentielle.

## Utilisation d'antivirus et/ou d'EDR

Pour une protection efficace des machines des utilisateurs, utiliser des antivirus et EDR est essentiel. Cela permet, via les antivirus, de bloquer une bonne partie des logiciels malveillants connus. L'EDR permet d'identifier des comportements potentiellement malveillants sur ces machines.

Ces outils sont tout de même limités et ne suffisent pas à défendre les machines des utilisateurs, notamment lorsque ceux-ci n'ont pas été sensibilisés aux bonnes pratiques cyber. Par exemple, un EDR permet de détecter des comportements potentiellement malicieux via une analyse comportementale, mais ne bloque pas ces comportements dans la plupart des cas.

## Sécurité réseau

Toutes les communications dirigées vers Internet doivent être tracées et, si possible, analysées. Cela peut être effectué via l'utilisation d'un serveur proxy dédié. Tous les serveurs contactés sont tracés et des listes de confiance peuvent être utilisées pour bloquer les communications avec certains serveurs réputés potentiellement malicieux.

De plus, l'utilisation d'un serveur d'un serveur proxy permet d'ajouter des technologies IDS et IPS. Ces outils permettent de détecter et prévenir d'éventuelles tentatives d'intrusion en analysant le contenu des paquets réseaux.

Des outils de sandbox peuvent aussi être mis en place au niveau du serveur proxy. Ainsi, lors du téléchargement de fichiers, ceux-ci pourront être analysés au niveau du serveur proxy avant d'être réellement téléchargés par l'utilisateur final.

Comme toutes les protections, ces outils sont limités. Ils peuvent l'être par la quantité d'informations à traiter car les fonctionnalités sont généralement prioritaires par rapport à la sécurité. De plus, de nouvelles techniques non connues ou difficilement détectables pourraient ne pas être détectés.

## **6. Réaction**

### Isolation des machines infectées

Toute machine infectée doit être isolée du reste du réseau, ceci afin de limiter la portée d'une compromission. La machine, une fois isolée, pourra être analysée pour comprendre le mode opératoire utilisée par l'attaquant.

### Résoudre les chemins d'attaques utilisés

Il est important, suite à une attaque, d'identifier quel a été le chemin utilisé par l'attaquant pour arriver à son objectif. Une fois celui-ci identifié, il est essentiel de mettre en place des mesures pour éviter qu'une autre attaque utilise le même chemin.

## Mise en place d'équipes spécialisés

Des équipes spécialisées peuvent être mises en place, ou engagées via un prestataire pour laisser ces tâches à des experts en réponse à incident. On parle alors de CERT (Computer Emergency Response Team).

## **5. Conclusion**

Les attaques basées sur des rootkits, même si elles sont loin d'être aussi répandues que les attaques par rançongiciels, représentent une sérieuse menace pour les organisations. Dans le scénario de notre attaque, la possibilité de mener des extractions de données médicales et personnelles constituent un réel danger pour les individus, et dans d'autres domaines stratégiques comme la défense ou l'industrie, cela pourrait constituer des atteintes à la souveraineté nationale ou nuire gravement à la compétitivité d'une entreprise. Cette note technique vise à fournir des informations pratiques afin de mieux comprendre la chaîne d'attaque utilisée par les attaquants pour l'accès initial, l'élévation de privilèges et l'extraction de données. Cette note technique ne peut en aucun cas être considéré comme un guide défensif complet, mais vise à servir de ressource pour les responsables de la sécurité pour certaines organisations pour mieux comprendre les rootkits et se protéger contre les attaques par rootkits. Il s'appuie sur nos travaux de recherche et notre projet pour permettre aux organisations de mieux se préparer, de mieux détecter et mieux réagir face à ce type d'attaque.