

WINDOWS MALWARE HUNTER HANDBOOK

Page de service

Référence : N/A

Plan de classement : Cybersécurité, Malwares, Microsoft Sysinternals suite

Niveau de confidentialité : Corporate

Mises à jour :

VERSION	DATE	AUTEUR	DESCRIPTION
1.0	21/01/2023	Thomas PRADEAU	Création du document
2.0	02/08/2023	Thomas PRADEAU	Ajout des notions d'API REST et utilisation de Virustotal.
2.1		Thomas Pradeau	Présentation d'un script de détection Python/API Virustotal.
3.0	01/03/2023	Thomas PRADEAU	Ajout des notions de signature, certificat et chiffrement.
3.1	03/03/2023	Thomas PRADEAU	Amélioration du robot de scan. Vérification de la présence d'une signature.
3.2	05/03/2023	Thomas PRADEAU	Ajout de la table bibliographique

Validations :

VERSION	DATE	NOM	RÔLE
1.0	29/01/2023	Jérôme VALENTI	Enseignant
2.1	19/02/2023	Jérôme VALENTI	Enseignant
3.2	05/03/2023	Jérôme VALENTI	Enseignant

Diffusions :

VERSION	DATE	NOM	CADRE DE LA DIFFUSION
1.0	29/01/2023	Jérôme VALENTI	Remise pour correction
2.1	19/02/2023	Jérôme VALENTI	Remise pour correction
3.2	05/03/2023	Jérôme VALENTI	Remise pour correction

Sommaire

Page de service	1
Sommaire	2
Rappel du contexte	4
Objectifs	4
Les concepts fondamentaux	5
Le processeur et les processus	6
Le processeur	6
Les processus	6
Les threads	6
Priorité et affinité	7
Handle	8
La mémoire vive	9
La mémoire de masse	10
Outils d'analyse	11
Contexte	11
Registres Windows	11
Process Explorer	12
Détail des processus	13
Virus Total	14
Le mode d'exécution	15
Process Monitor	16
Détail des évènements	17
Les filtres	18
Cas 1 – Modification de la page d'accueil du navigateur	20
Contexte	20
But de l'attaque	20
Comment repérer la supercherie ?	20
Intercepter la modification	21
Création du piège	22
Conclusion	24
Cas 2 – DNS Empoisonné	25
Contexte	25
But de l'attaque	25
Comment repérer un DNS empoisonné ?	25
Identifier les paramètres du DNS	26
Capture des résultats sous Process Monitor	28
Conclusion	29
Détection d'un programme malveillant	30
Détection par signature	30

Détection par comportement	30
Détection par contrôle de l'intégrité	30
Détection heuristique	31
Virustotal	31
Scan de fichier	32
Scan de site Web	33
Fonctionnalité de recherche	33
Conclusion	33
Automatisation et API	34
Les scripts	34
Les APIs	35
Robot de scan Virustotal	35
Analyse approfondie d'un fichier	46
Contexte	46
Structure du fichier - PE	46
Structure de fichier - EXE	48
Le code assembleur	49
Un exécutable, programme légitime ou malware ?	50
Les algorithmes de hash	52
La signature de code	54
Les certificats de signature de code	55
Implémentation des certificats dans Windows	56
Les magasins de certificats	56
La MMC (Microsoft Management Console)	57
L'utilitaire sigcheck	58
Signature dans Process Explorer	59
L'utilitaire signtool	60
Vérification de signature par le robot de scan	61
Type de fichier	61
Vérification du certificat	61
Avertir l'utilisateur	61
Bibliographie	62
Processus & Threads	62
Programmation objet avec PowerShell	62
Utilisation de la suite Sysinternals dans la cybersécurité	62
Méthodes de détection virale	62
Utilisation de Python et des APIs Web	62
Table des illustrations	63

Rappel du contexte

NSI (Networking Solution Incorporated) est une entreprise de services du numérique (ESN)¹ qui se charge de la réalisation et de la maintenance des infrastructures matérielles et logicielles de ces clients.

Les ESN sont spécialisées dans les nouvelles technologies et englobent en général plusieurs métiers. Celles-ci ont comme objectif principal d'accompagner leurs clients dans la réalisation de leurs projets.

Objectifs

Le présent document a pour objectifs de former et introduire aux collaborateurs de l'entreprise NSI les différentes menaces opérantes sous le système d'exploitation **Windows 10**², ainsi que différents moyens de les détecter.

Windows 10 est présentement le système d'exploitation le plus utilisé aussi bien chez les particuliers que les professionnels, que ce soit sur les postes ou même sur des serveurs avec la famille à part entière Windows Server³.

Les notions de processus et comment un système d'exploitation organise les exécutions des programmes sur une machine seront également abordés afin de mieux comprendre comment il est possible de distinguer un processus malveillant des autres programmes. La très grande majorité des postes sont protégés derrière un ou plusieurs pare-feux, en plus d'un antivirus, qu'il soit intégré au système ou non. Or il s'avère que plus de 75 % des virus passent au travers de ces protections. Un travail d'analyse plus approfondie est donc nécessaire en plus de l'utilisation de ces outils.

¹ https://fr.wikipedia.org/wiki/Entreprise_de_services_du_num%C3%A9rique

² https://fr.wikipedia.org/wiki/Windows_10

³ https://fr.wikipedia.org/wiki/Windows_Server

Les concepts fondamentaux

Afin d'être en mesure de comprendre comment les programmes malveillants affectent les machines, il nous faut dans un premier temps comprendre les concepts fondamentaux, à savoir :

- Notion de processeur, de processus et thread
- La mémoire vive
- La mémoire de masse (mémoire morte)

Ces notions sont communes pour toutes les machines et systèmes d'exploitation. Bien que ceux-ci fonctionnent différemment en interne, ils gardent tous certains points communs. Tels que la gestion des processus et l'utilisation de mémoire vive pour le traitement et la mémoire de masse pour le stockage à plus ou moins long terme.

Le processeur et les processus

Le processeur

Commençons par quelques définitions, le **processeur** est le composant physique dans un ordinateur qui va effectuer différents traitements à une certaine fréquence, exprimé en hertz. Un processeur est constitué de cœurs physiques, ce sont ces cœurs qui effectuent les calculs et traitements.

La figure ci-dessous détaille l'anatomie d'un processeur, ainsi que ces différentes parties :

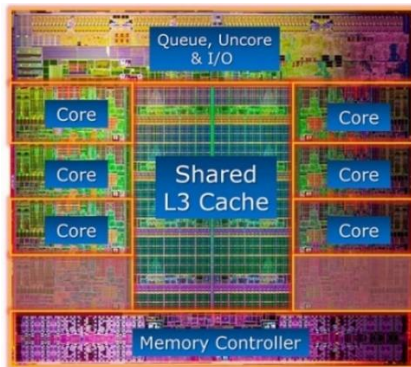


Figure 1 Anatomie du processeur

On peut constater les différents cœurs « **core** ». Mais également la mémoire de traitement interne au processeur, appelée mémoire cache « **Shared L3 Cache** ».

Le contrôleur mémoire « **memory controller** » est le composant qui assure la liaison entre le processeur et la mémoire vive.

Ainsi que le module de gestion des entrées sorties entre le processeur et le chipset, noter « **Queue, Uncore & I/O** ».

Le fonctionnement beaucoup plus détaillé est disponible sur l'article suivant <https://www.freecodecamp.org/news/how-does-a-cpu-work/>. Ce qu'il faut retenir principalement est que le processeur s'occupe de l'exécution des programmes sur la machine.

Les processus

Le **processus** est un programme en cours d'exécution. Un programme peut être composé de plusieurs processus. À ne pas confondre avec le thread, qui lui est une unité de base à laquelle le système d'exploitation alloue du temps processeur. Autrement dit, un processus est composé d'un ou plusieurs threads. Un thread est ensuite traité par le processeur pendant un certain temps, avant de passer à un autre thread.

Les threads

Le thread lui, ou fil d'exécution est un ensemble de code machine (Assembleur), qui ensuite exécuté par le processeur durant la période qui lui est accordée. Le thread est le fruit du code du programme (Java, Python, C++)⁴, qui a été compilé afin d'être compréhensible par le processeur.

En somme, l'exécution de code par le processeur peut être schématisée comme il suit :

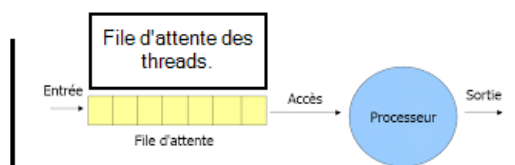


Figure 2 Attente et exécution des threads par le processeur

⁴ https://fr.wikipedia.org/wiki/Langage_de_programmation

Chaque thread est exécuté chacun son tour. L'ordre de passage et le temps d'exécution sont gérés par le système d'exploitation. Ce mécanisme est aussi dénommé « Ordonnanceur »⁵.

Priorité et affinité

L'ordonnanceur organise l'ordre d'exécution des threads en fonction de leurs affinités et de leurs priorités.

- L'affinité indique les cœurs physiques sur lesquels sont exécutés les threads. Par exemple, un thread possédant une affinité de 0-1 sera exécuté sur les cœurs 1 et 2 du processeur. Il est ainsi possible de modifier cette affinité pour par exemple réserver un ou plusieurs cœurs du processeur aux programmes les plus gourmands. Il est cependant recommandé de ne pas modifier cette affinité au risque de rendre le système plus lent ou instable.
- La priorité d'un thread influe sur son ordre de passage dans l'ordonnanceur. En effet, plus la priorité est haute, plus on accordera de temps un processeur à ce thread, et vice versa. Par défaut, les processus système, comme les **drivers**⁶, possèdent une priorité plus élevée que les programmes standards.

Ces paramètres sont facilement modifiables via le **gestionnaire des tâches** intégré à Windows. Dans celui-ci, sélectionnez l'onglet « **détail** », puis la liste détaillée des processus en cours d'exécution s'affiche.

La figure suivante montre la modification de l'affinité et la priorité d'un processus par le biais du gestionnaire des tâches :

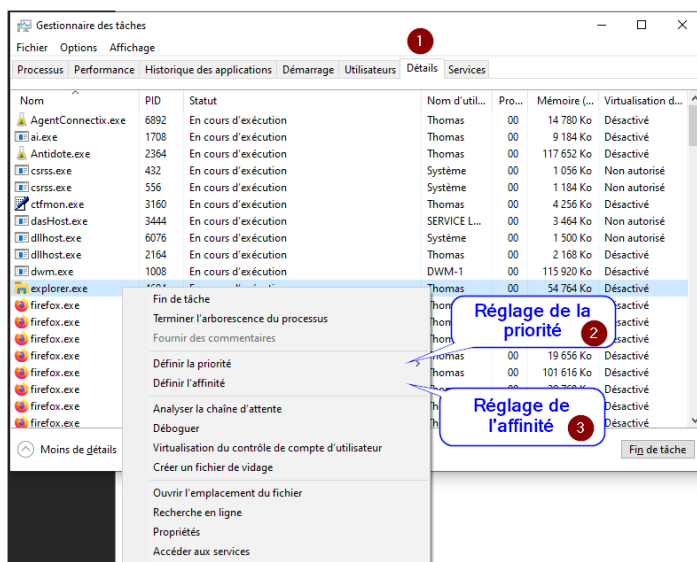


Figure 3 Définition de la priorité et de l'affinité d'un processus

⁵ [https://fr.wikipedia.org/wiki/Ordonnancement dans les syst%C3%A8mes d%27exploitation](https://fr.wikipedia.org/wiki/Ordonnancement_dans_les_syst%C3%A8mes_d%27exploitation)

⁶ <https://www.malekal.com/pilotes-drivers-windows/>

Handle

La dernière notion liée aux processus est les handle⁷. Un handle, qui peut se traduire par « poignée ». Un handle est une référence abstraite à une ressource mémoire ou un fichier.

Il s'agit d'un pointeur qui permet à un thread d'accéder rapidement et facilement à une portion de code contenu dans la mémoire de traitement.

Chaque thread crée un nombre variable de handles, il en crée autant que celui-ci en a besoin pour fonctionner.

Afin d'approfondir les connaissances sur les processus, threads et handles, il est recommandé de lire cette documentation technique fournie par Microsoft :

<https://learn.microsoft.com/fr-fr/windows/win32/procthread/processes-and-threads>

À retenir :

Le processeur exécute les instructions. Un programme en cours d'exécution est représenté par un ou plusieurs processus, qui eux-mêmes possèdent plusieurs threads. Un thread ou fil d'exécution contient le code compilé exécuté par le processeur. Les threads sont organisés et triés en fonction de leurs priorités et de leurs affinités. Cela est effectué par le système d'exploitation, plus précisément l'ordonnanceur.

⁷ <https://stackoverflow.com/questions/902967/what-is-a-windows-handle>

La mémoire vive

La **mémoire vive** ou plus couramment appelée « **RAM** », qui signifie « Random Access Memory », est un type de mémoire utilisé pour l'exécution des programmes. Cette mémoire est directement reliée au processeur et est beaucoup plus rapide en termes de bande passante et latence que la mémoire de stockage.

La RAM stocke les données utilisées par le processeur. À l'exécution d'un programme, les données du programme lui-même, ainsi que ses dépendances toutes deux stockées sur mémoire de masse (Disque dur), sont chargées en mémoire vive. Le code du programme est ainsi compilé puis exécuté par le processeur sous forme de threads comme expliqués précédemment.

Étant donné que la mémoire vive stocke les données utiles à l'exécution des programmes, il s'agit d'un endroit critique et est possiblement une porte d'entrée pour les programmes malveillants. En effet il serait théoriquement possible d'altérer cette mémoire pour produire des effets indésirables. Dans la pratique les programmes sont isolés dans la mémoire et ne peuvent pas modifier la mémoire en dehors de l'espace qui leur est alloué.

On se repère dans la mémoire à l'aide d'adresses, noté en hexadécimal (base 16). Ces adresses ressemblent à ceci :

➤ 0x00034ab1

Ce sont ces adresses qui permettent de communiquer avec la mémoire afin d'y stocker les données voulues.

Afin de faciliter l'accès aux données les plus utilisées par le processeur, on trouve également de la mémoire cache, présentée sur la [figure 1](#), qui est une mémoire encore plus rapide, mais en quantité limitée.

La figure suivante représente les interactions entre le processeur et la mémoire vive et la mémoire cache :

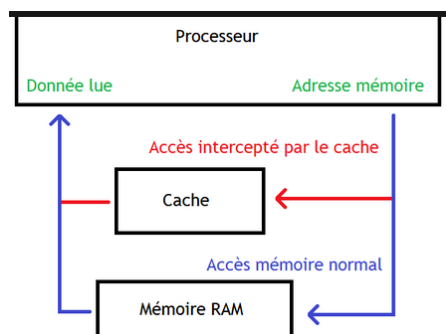


Figure 4 Interactions entre le processeur et les mémoires de traitement

À retenir :

La mémoire cache et la mémoire vive sont des mémoires très rapides et à faibles latences utilisées par le processeur pour l'exécution des programmes. Le processeur accède et interagit avec ces mémoires par le biais d'adresses notées en hexadécimal.

La mémoire de masse

La mémoire de masse, à l'inverse de la mémoire vive, sert à stocker des données de manière persistante. La mémoire de masse existe sous plusieurs formes, telles que le disque dur, le SSD ou le SSHD qui combine les deux premiers.

C'est bien en se penchant sur la mémoire de masse que l'on comprend l'intérêt des mémoires de traitements vues dans la [partie précédente](#). En effet, la mémoire de masse est beaucoup plus lente que la mémoire vive.

À titre de comparaison, le temps de latence moyen d'un disque dur est de 15 millisecondes, celui d'un SSD est de 0,2 milliseconde, et celui de la mémoire vive est de l'ordre de la nanoseconde, soit mille fois plus rapides⁸.

Le temps de latence représente le temps que met la mémoire pour répondre à une demande du système.

Alors la mémoire de masse ne sert qu'à retenir les données lorsque la machine n'est plus sous tension, ce qui n'est pas le cas de la RAM, puisqu'il s'agit de mémoire vive. A contrario de la mémoire de masse, aussi dénommée mémoire morte.

Lors de l'exécution d'un programme, toutes les données nécessaires à l'exécution sont copiées de la mémoire de stockage à la mémoire vive. C'est cette partie que l'on appelle le temps de chargement. L'exécution du programme se poursuit alors avec les concepts présentés dans la partie concernant [le processeur](#) et [la mémoire vive](#).

À retenir :

La mémoire de masse s'occupe de la rétention des données à court et long termes. Contrairement à la mémoire vive, la mémoire de masse est beaucoup plus lente, mais permet de retenir les données hors tension et accepte de plus grandes capacités de stockage.

⁸ [https://fr.wikipedia.org/wiki/Temps_de_r%C3%A9ponse_\(informatique\)](https://fr.wikipedia.org/wiki/Temps_de_r%C3%A9ponse_(informatique))

Outils d'analyse

Contexte

Après avoir fait un tour d'horizon des notions fondamentales nécessaires afin de comprendre la vie d'un programme, de son lancement à son arrêt. Nous allons à présent voir comment utiliser certains outils d'analyse afin de repérer des processus malveillants et ses comportements suspects, tels que l'accès à des clés de registre ou fichiers de configuration par exemple.

Nous allons explorer les différentes possibilités disponibles avec les outils **ProcessMonitor** et **ProcessExplorer**.

Ces outils sont disponibles dans la suite **Sysinternals**, proposée par Microsoft à ce lien : <https://download.sysinternals.com/files/SysinternalsSuite.zip>

Bien qu'ils puissent sembler similaires, ces outils sont complémentaires.

Registres Windows

Les registres Windows permettent de stocker les paramètres de bas niveau du système. Les registres sont utilisés par le système et peuvent l'être par les applications, ceux-ci permettent en effet d'y stocker des valeurs comme la version ou la licence rattachée à un logiciel. Le registre permet d'enregistrer des paramètres sur la mémoire de masse.

Il s'agit donc d'un point critique qui peut être affecté par certains processus malveillants.

Ce registre s'organise en clés de registres, dans lesquelles l'on trouve les valeurs. Chacune de ces valeurs possède un type, tel que DWORD ou REG_SZ. Le type définit la valeur qui est stockée. La valeur quant à elle peut être un entier ou une chaîne de caractères.

Les clés de registres sont-elles organisées sous forme d'arborescence ? La figure ci-dessous montre à quoi ressemble une partie de la structure des registres sous Windows 11 :

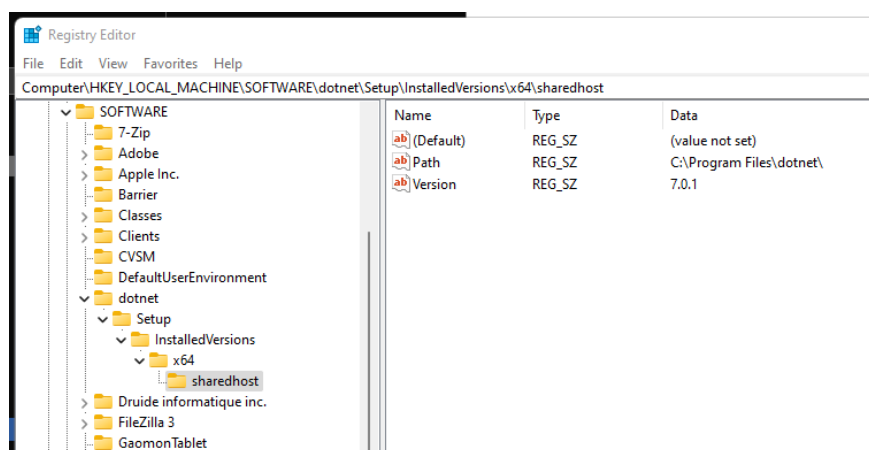


Figure 5 : Structure du registre sous Windows 11

Il est possible d'accéder et modifier ce registre à l'aide d'un outil graphique intégré à Windows, qui se nomme l'éditeur de registres (Regedit.exe).

Process Explorer

Process Explorer est un outil qui permet de lister tous les processus en cours d'exécution à un instant T sur le système.

La figure suivante montre la fenêtre principale de Process Explorer :

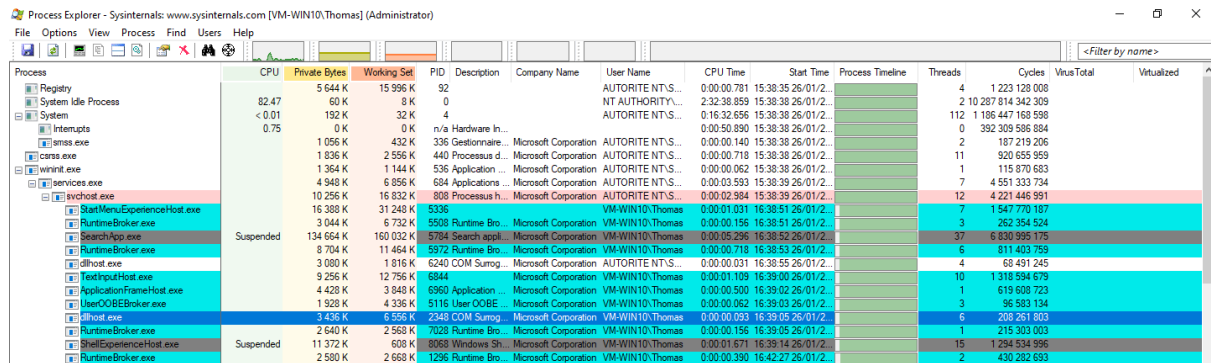


Figure 6 : Fenêtre principale de Process Explorer

Les processus en cours d'exécution sont présentés sous forme d'arborescence. On constate alors que de très nombreux processus sont en réalité des processus d'enfant, qui dépendent de leurs processus parents. On remarque quatre processus parents principaux :

- System
- Wininit.exe⁹
- Winlogon.exe¹⁰
- Explorer.exe

Explorer.exe est le processus qui permet la gestion du bureau et de l'interface utilisateur, il sert également de processus parent à de nombreux programmes, telle le montre la figure suivante :

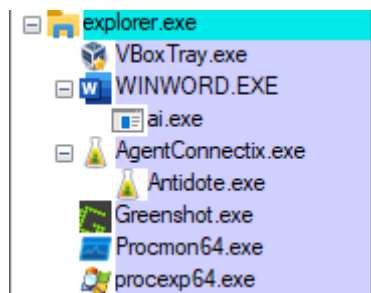
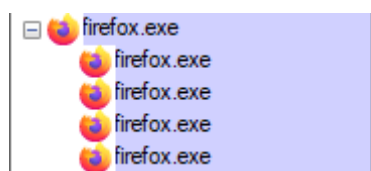


Figure 7 : Processus enfants d'Explorer.exe

Au contraire, d'autres processus comme firefox.exe sont eux-mêmes leurs processus parents, tel le montre la figure suivante :

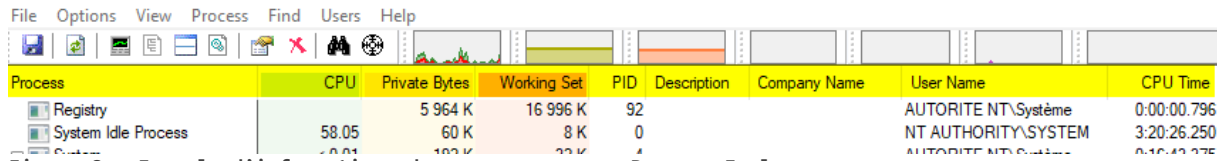


⁹ <https://social.technet.microsoft.com/Forums/ie/en-US/df6f5eeb-cbb9-404f-9414-320ea02b4a60/wininitexe-what-is-is-and-why-is-it-constantly-running?forum=win10itprosecurity>

¹⁰ <https://www.malekal.com/winlogon-exe/>

Détail des processus

Par rapport au gestionnaire des tâches inclus dans Windows, Process Explorer présente beaucoup plus de détails en rapport avec les processus. Ces informations sont organisées sous forme de colonnes, tel le montre la figure suivante :



Process	CPU	Private Bytes	Working Set	PID	Description	Company Name	User Name	CPU Time
Registry		5 964 K	16 996 K	92			AUTORITE NT\Système	0:00:00.796
System Idle Process	58.05	60 K	8 K	0			NT AUTHORITY\SYSTEM	3:20:26.250
System	0.04	100 K	32 K	4			AUTORITE NT\Système	0:00:00.000

Figure 8 : Exemple d'informations des processus sous Process Explorer

Cette figure n'est pas du tout exhaustive qu'en aux informations disponibles sur les processus, voici donc les principales :

Concernant les performances, la colonne « **CPU** » indique en pour cent l'utilisation du processeur, la colonne « **Private Bytes** » indique le nombre d'octets réservés en mémoire RAM avant l'exécution d'une application¹¹. La colonne « **Working Set** » indique en octets la taille de l'ensemble de travail, il s'agit des éléments en mémoire récemment utilisés par les threads du processus en question.

Concernant les informations en rapport avec la provenance du processus, la colonne « **PID** » indique l'identifiant du processus, soit un identifiant unique du processus. La colonne « **Description** » offre la description contenue dans la signature de l'exécutable à l'origine du processus. La colonne « **Company Name** » nous donne le nom de l'entreprise à l'origine de l'exécutable à l'origine du processus. Et enfin la colonne « **User Name** » indique le nom de l'utilisateur ainsi que son domaine¹² qui est à l'origine du processus.

Il est possible d'afficher ou non des colonnes supplémentaires un faisant un clic droit sur celles-ci, on clique ensuite sur « Select columns », la figure suivante montre les colonnes disponibles :

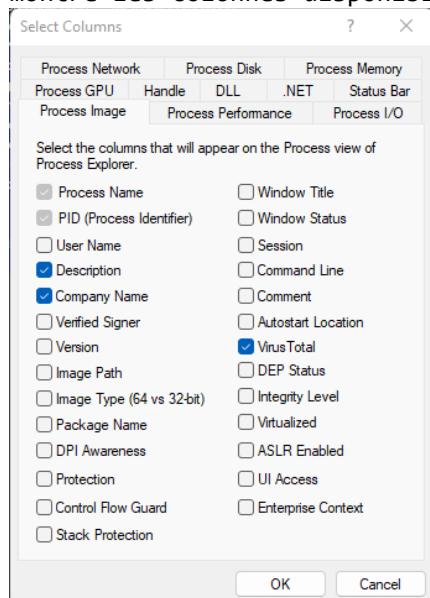


Figure 9 : Étendue des informations disponibles sous Process Explorer

Afin de comprendre l'étendue des possibilités offertes par cet outil, il est recommandé d'aller visionner cette vidéo :

https://www.youtube.com/watch?v=svLwLwB_How

Réalisée par **Mark Russinovich**, celle-ci rentre en profondeur dans les différentes fonctionnalités les plus poussées de Process Explorer.

¹¹ <https://stackoverflow.com/questions/1984186/what-is-private-bytes-virtual-bytes-working-set>

¹² <https://learn.microsoft.com/fr-fr/windows-server/identity/ad-fs/deployment/join-a-computer-to-a-domain>

Virus Total

Process Explorer permet de scanner certains fichiers à l'aide du service Virus total. Celui-ci scanne les fichiers avec plusieurs antivirus différents, afin de maximiser la détection de code malveillant.

Process Explorer possède un raccourci vers cet outil, qui est en temps normal, accessible directement par le biais d'un navigateur Web, à cette adresse :

<https://www.virustotal.com/gui/home/upload>

Sous Process Explorer, il suffit de faire un clic droit sur le processus que l'on souhaite scanner, puis dans le menu contextuel, cliquez sur « Check Virustotal.com », comme le montre la figure suivante :

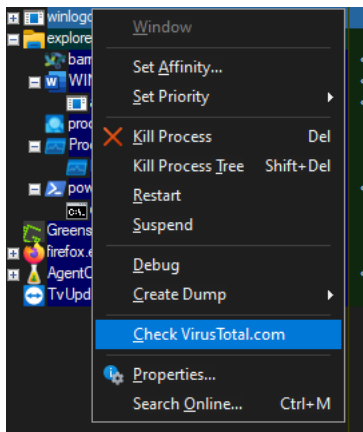


Figure 10 : Menu contextuel Virus total

Une fois scanné, un score sur 74 apparaîtra dans la colonne Virus Total. Ce score représente le nombre d'antivirus qui considèrent cet exécutable comme malveillant. Si la colonne Virus Total n'est pas présente, il suffit de l'afficher en répétant les opérations précédentes afin de modifier les colonnes affichées par défaut.

La figure ci-dessous montre un exemple de résultat Virus total :

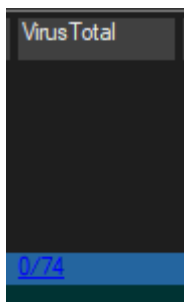


Figure 11 : Exemple de résultat Virus total

Remarque :

Un score non nul ne signifie pas obligatoirement qu'un exécutable est dangereux pour la machine. Cela peut signifier que la somme de contrôle¹³ de l'exécutable n'est pas connue de Virus Total.

¹³ <https://academy.bit2me.com/fr/que-es-hash/>

Le mode d'exécution

Sur les postes exécutant le système d'exploitation Windows, il existe deux modes d'exécution.

- Le mode utilisateur
- Le mode noyau

Ces deux modes sont importants à comprendre puisque ceux-ci définissent les restrictions en rapport avec l'espace d'adressage d'une application. Dit plus simplement, le mode d'exécution vient modifier les autorisations qu'a un processus vis-à-vis de la mémoire de traitement, la RAM.

Tel qu'expliqué dans [la partie](#) sur la mémoire de traitement. La RAM contient toutes les données nécessaires aux processus en cours d'exécution pour fonctionner correctement. En outre, la mémoire vive contient aussi bien les données du système d'exploitation que des programmes ou des drivers.

Il est donc nécessaire d'introduire une isolation entre ces processus. Interviennent alors les modes d'exécution¹⁴. La différence entre ces deux modes est très simple.

- **Le mode utilisateur** concerne tous les programmes exécutés par l'utilisateur, ainsi que certains drivers. En mode utilisateur, l'espace d'adressage est propre à chaque application. Autrement dit, un processus en mode utilisateur ne peut pas aller modifier la mémoire allouée à un autre processus que lui-même. Ainsi, une application mal conçue ou qui serait prône à des plantages n'affecterait pas les autres applications ou le système.
- **Le mode noyau** est l'opposé du mode utilisateur. En mode noyau, les processus s'exécutent dans une plage de mémoire commune. Tous les processus qui s'exécutent en mode noyau sont autorisés à altérer les adresses mémoires qui font partie de la plage d'adresses allouée à ce mode d'exécution. Sans isolation, un processus qui pourrait affecter les données du système d'exploitation lui-même est causé des instabilités.

¹⁴ <https://learn.microsoft.com/fr-fr/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode>

Process Monitor

Là où Process Explorer ne fait que lister de manière détaillée les processus en cours d'exécution sur la machine. Process Monitor analyse le comportement de ces processus.

Si Process Explorer répond à la question « Qui ? », Process Monitor répond à la question « Fait quoi ? ».

Sur un ordinateur, le système d'exploitation fait le lien entre la couche applicative et la couche matérielle. Les applications effectuent donc des appels au système afin de lui demander des ressources, telles qu'un emplacement dans la mémoire vive, des fichiers sur le disque dur, ou l'accès à un périphérique tel qu'une imprimante par exemple.

Process Monitor permet donc de visualiser tous ces appels ! Il permet de voir tout ce que font les processus. Par exemple, quels fichiers ils modifient, quelles valeurs du registre ils lisent ou même les accès réseau des processus.

Il s'agit donc d'un outil très puissant dans le domaine de la cybersécurité. Il donne accès à tout un panel d'informations qui permettent de déterminer si un processus est malveillant ou non.

La figure ci-dessous montre la fenêtre principale de Process Monitor :

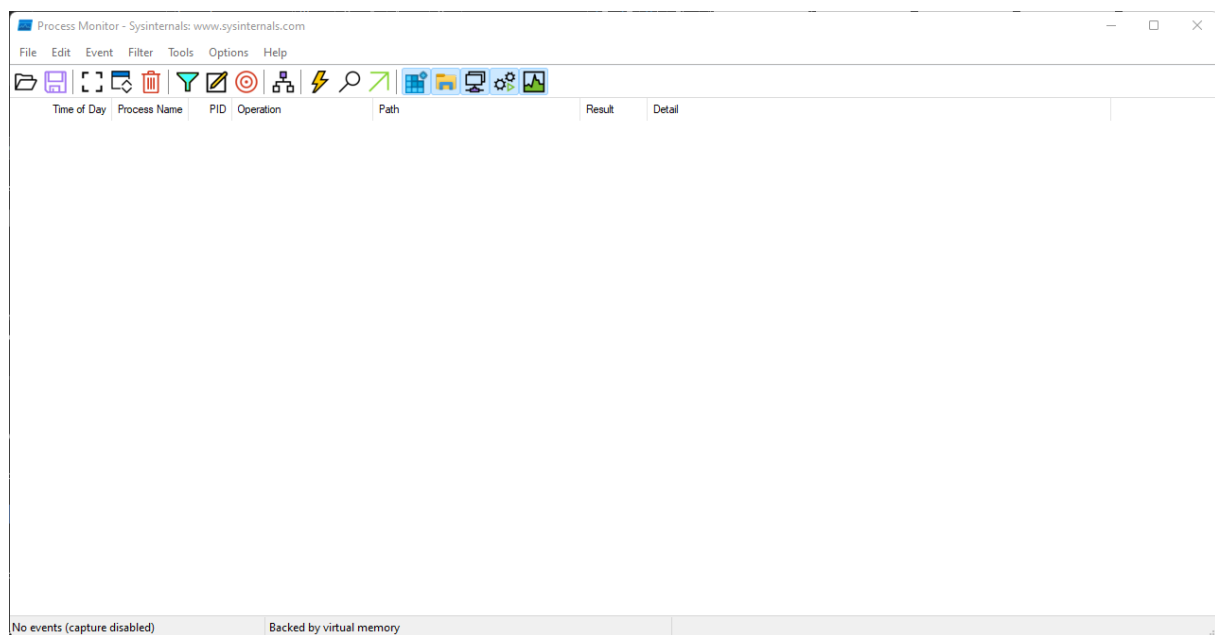


Figure 12 : Fenêtre principale de Process Monitor

Tous les événements capturés par Process Monitor apparaissent sous forme de liste. Où il est détaillé certaines informations en rapport avec ces événements.

Afin de tirer le meilleur parti de cet outil, commençons par analyser ces fonctionnalités principales. La figure ci-dessous montre l'accès aux fonctionnalités les plus couramment utilisées :

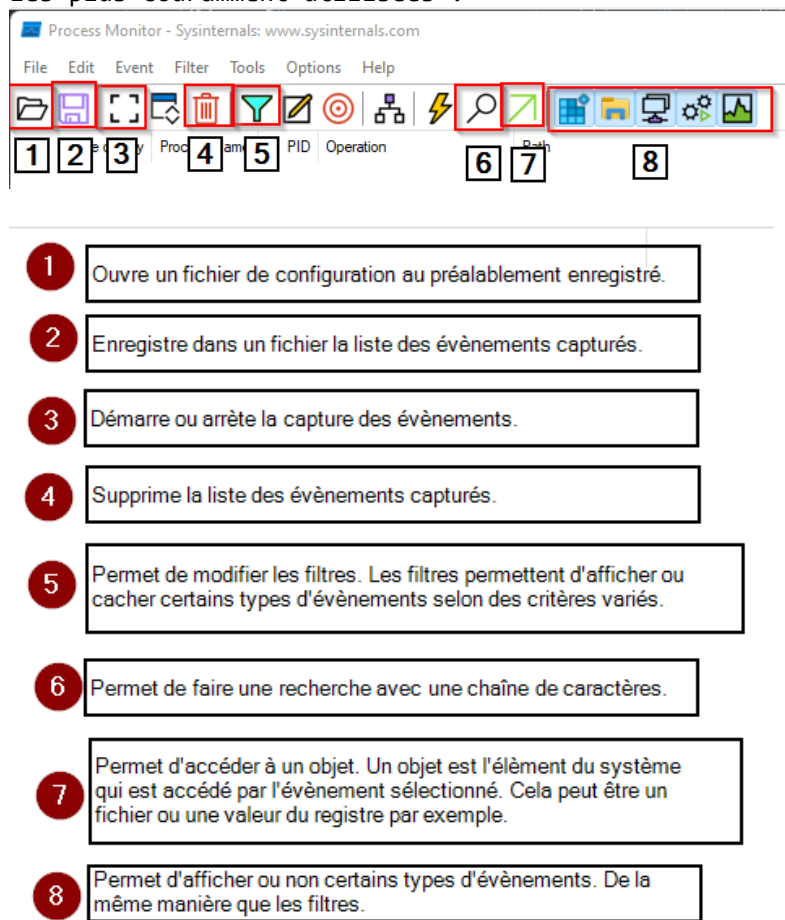


Figure 13 : Fonctionnalités principales de Process Monitor

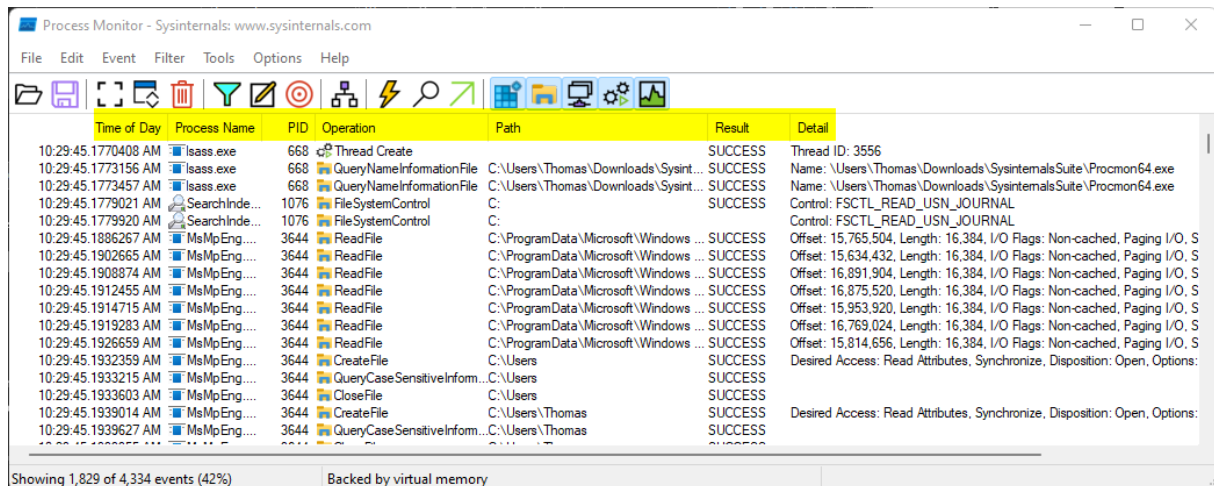
Cette liste n'est pas exhaustive concernant les fonctionnalités de Process Monitor, cependant celle-ci regroupe toutes les fonctionnalités utilisées le plus souvent.

Détail des événements

Commençons par capturer des événements, pour cela, cliquez sur le bouton noté « 3 » sur [cette figure](#) pour commencer la capture. Attendez quelques secondes, puis recliquez sur le même bouton pour stopper la capture.

Process Monitor a pendant ces quelques secondes capturé tous les événements système non filtrés. C'est-à-dire tous puisqu'il n'y a aucun filtre pour l'instant.

La figure ci-dessous montre un exemple de résultat obtenu :



Time of Day	Process Name	PID	Operation	Path	Result	Detail
10:29:45.1770408 AM	lsass.exe	668	Thread Create		SUCCESS	Thread ID: 3556
10:29:45.1773156 AM	lsass.exe	668	QueryNameInformationFile	C:\Users\Thomas\Downloads\Sysint...	SUCCESS	Name: \Users\Thomas\Downloads\SysinternalsSuite\Procmon64.exe
10:29:45.1773457 AM	lsass.exe	668	QueryNameInformationFile	C:\Users\Thomas\Downloads\Sysint...	SUCCESS	Name: \Users\Thomas\Downloads\SysinternalsSuite\Procmon64.exe
10:29:45.1779021 AM	SearchIndexing.exe	1076	FileSystemControl	C:	SUCCESS	Control: FSCTL_READ_USN_JOURNAL
10:29:45.1779920 AM	SearchIndexing.exe	1076	FileSystemControl	C:	SUCCESS	Control: FSCTL_READ_USN_JOURNAL
10:29:45.1886267 AM	MsMpEng.exe	3644	ReadFile	C:\ProgramData\Microsoft\Windows ...	SUCCESS	Offset: 15,765,504, Length: 16,384, I/O Flags: Non-cached, Paging I/O, S
10:29:45.1902665 AM	MsMpEng.exe	3644	ReadFile	C:\ProgramData\Microsoft\Windows ...	SUCCESS	Offset: 15,634,432, Length: 16,384, I/O Flags: Non-cached, Paging I/O, S
10:29:45.1908874 AM	MsMpEng.exe	3644	ReadFile	C:\ProgramData\Microsoft\Windows ...	SUCCESS	Offset: 16,891,904, Length: 16,384, I/O Flags: Non-cached, Paging I/O, S
10:29:45.1912455 AM	MsMpEng.exe	3644	ReadFile	C:\ProgramData\Microsoft\Windows ...	SUCCESS	Offset: 16,875,520, Length: 16,384, I/O Flags: Non-cached, Paging I/O, S
10:29:45.1914715 AM	MsMpEng.exe	3644	ReadFile	C:\ProgramData\Microsoft\Windows ...	SUCCESS	Offset: 15,953,920, Length: 16,384, I/O Flags: Non-cached, Paging I/O, S
10:29:45.1919283 AM	MsMpEng.exe	3644	ReadFile	C:\ProgramData\Microsoft\Windows ...	SUCCESS	Offset: 16,769,024, Length: 16,384, I/O Flags: Non-cached, Paging I/O, S
10:29:45.1926659 AM	MsMpEng.exe	3644	ReadFile	C:\ProgramData\Microsoft\Windows ...	SUCCESS	Offset: 15,814,656, Length: 16,384, I/O Flags: Non-cached, Paging I/O, S
10:29:45.1932359 AM	MsMpEng.exe	3644	CreateFile	C:\Users	SUCCESS	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options:
10:29:45.1933215 AM	MsMpEng.exe	3644	QueryCaseSensitiveInform...	C:\Users	SUCCESS	
10:29:45.1933603 AM	MsMpEng.exe	3644	CloseFile	C:\Users	SUCCESS	
10:29:45.1939014 AM	MsMpEng.exe	3644	CreateFile	C:\Users\Thomas	SUCCESS	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options:
10:29:45.1939627 AM	MsMpEng.exe	3644	QueryCaseSensitiveInform...	C:\Users\Thomas	SUCCESS	

Figure 14 : Exemple de résultat de capture

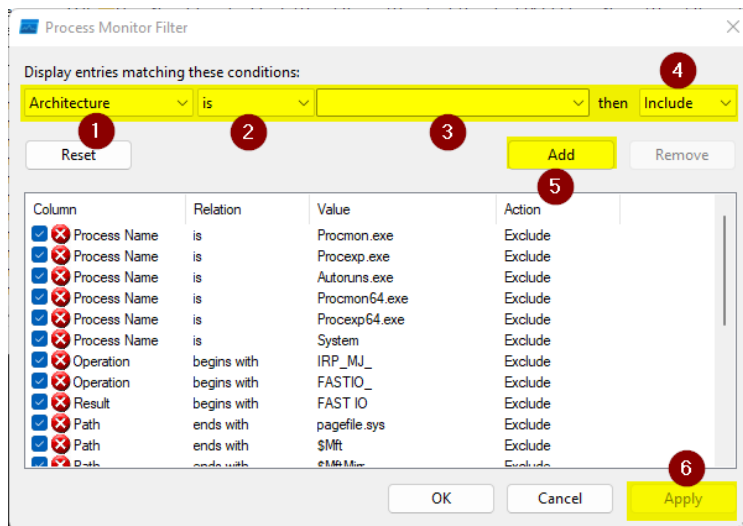
On retrouve parmi les colonnes affichées par défaut les informations suivantes :

- **Time of Day:** l'heure de la journée à laquelle s'est produit l'évènement. Cette information est donnée à la microseconde près.
- **Process Name:** indique le nom du processus à l'origine de l'évènement. Plus précisément, celui de son exécutable.
- **PID :** Process identifier, il s'agit d'un identifiant unique à chaque processus qui permet de facilement les différencier.
- **Operation:** il s'agit de la nature de l'évènement. Plus précisément ce que le processus a essayé de faire. L'opération peut se traduire par un verbe d'action.
- **Path:** le chemin physique sur la mémoire de masse relative à l'évènement.
- **Result:** le résultat de l'opération décrite dans la colonne « Operation ».
- **Detail:** Détails concernant l'évènement.

Les filtres

Les filtres permettent de masquer ou d'afficher certains évènements. Il est possible de n'afficher que les évènements qui possèdent l'opération « ReadFile » par exemple. À l'inverse il est aussi possible de masquer certains évènements qui obstruent la lecture des résultats.

Pour paramétrer les évènements, l'on clique sur le bouton noté « 5 » sur [cette figure](#), la fenêtre de configuration des filtres apparaît alors comme le montre la figure suivante :



La procédure pour ajouter un filtre est la suivante :

1. Sélectionner le type de filtre, le type est identique aux noms des colonnes présentées plus tôt.
2. Sélectionner la condition, telle que « est », ou « contient ».
3. Indiquer la valeur à vérifier. Cette valeur sera comparée selon la condition choisie.
4. Définir si le filtre doit inclure ou exclure les événements qui respectent la condition définie.
5. Cliquer sur ajouter pour ajouter le filtre à la liste des filtres actifs.
6. Appliquez les changements à la liste des événements capturés.

Cas 1 — Modification de la page d'accueil du navigateur

Contexte

Les utilisateurs de l'entreprise Vinci, cliente de NSI, se connectent à l'extranet de l'entreprise avec un identifiant et un mot de passe, via un portail de connexion Web.

Or, bien que la page de connexion n'ait rien d'anormal, l'URL d'accès à cette page n'est pas tout à fait la même, et pourrait tromper certains salariés de l'entreprise qui ne verraient pas la supercherie. Une fois les identifiants entrés, le pirate peut y accéder comme bon lui semble, et l'utilisateur se retrouve redirigé vers le vrai extranet de Vinci.

Cette page de connexion a été définie comme étant la page par défaut du navigateur Web Mozilla Firefox.

But de l'attaque

Le but de cette attaque est de récupérer les identifiants de connexion des utilisateurs de l'extranet de Vinci afin de pouvoir y accéder par le biais de comptes utilisateurs compromis. Les identifiants sont dérobés par le biais d'un faux portail de connexion, hébergé par le pirate. Contrairement au vrai portail de connexion qui lui est hébergé sur le serveur Web de l'entreprise.

La fausse page de connexion est très similaire à la page originale, seule l'URL peut différer est trompé par les salariés.

Comment repérer la supercherie ?

Dans un premier temps, il faut vérifier si la page d'accueil du navigateur a été altérée ou non. Pour cela, allez dans les paramètres de la page d'accueil de Firefox. Ces paramètres sont accessibles par le petit engrenage situé en haut à droite de la page d'accueil du navigateur. Puis cliquez sur « Gérer plus de paramètres ».

Les figures suivantes détaillent la procédure :

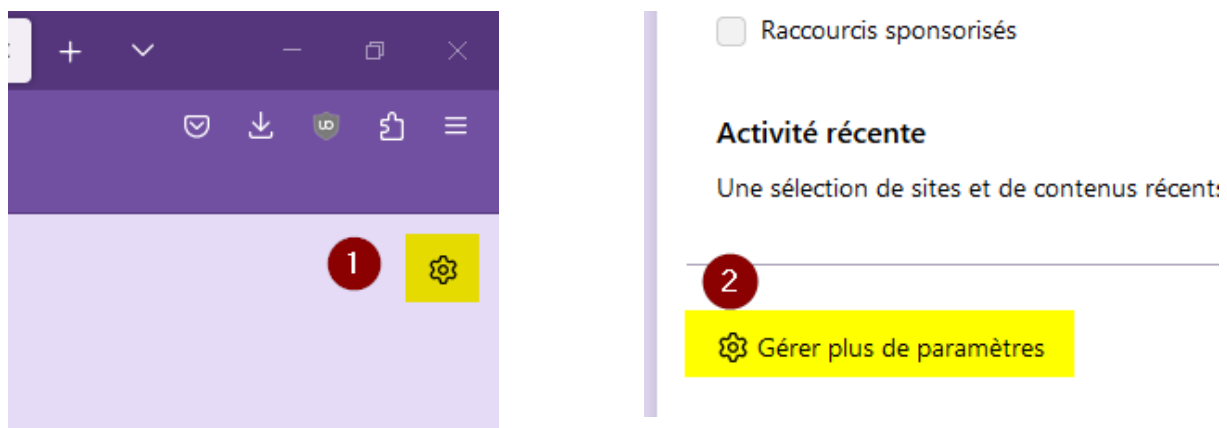


Figure 15 : Procédure d'accès aux paramètres de Firefox.

Les paramètres de gestion de la page d'accueil apparaissent :

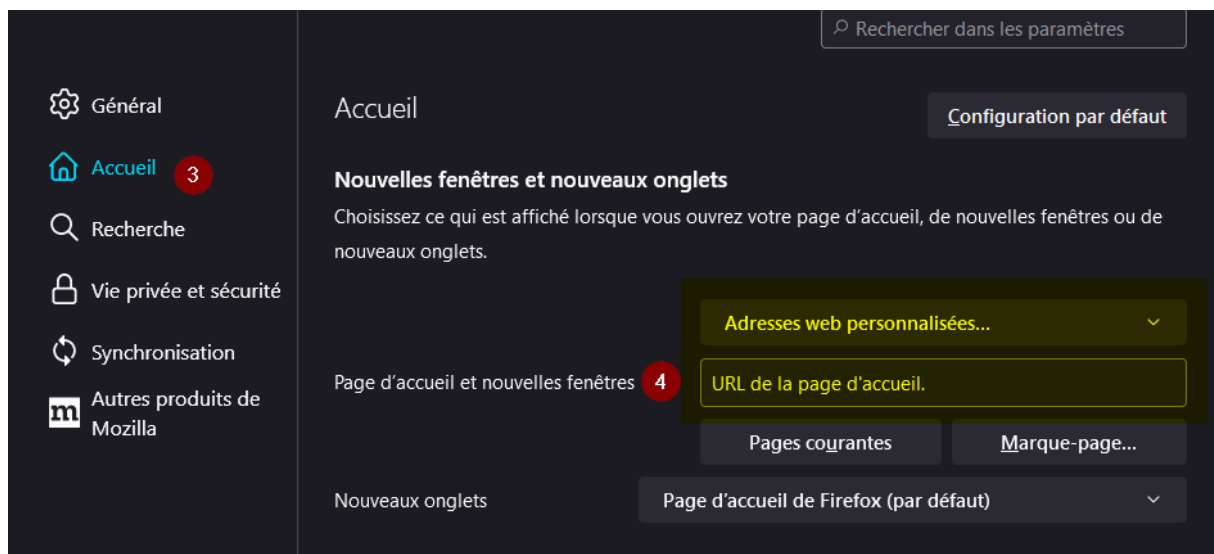


Figure 16 : Paramètres de gestion de la page d'accueil de Firefox

Il suffit ensuite de vérifier si l'URL indiquée dans le champ 4 indiqué ci-dessus correspond à l'URL du portail Web authentique.

Intercepter la modification

La prochaine étape est de capturer le processus responsable de ce changement. Pour commencer, démarrez Process Monitor. Gardez sur une moitié de l'écran le navigateur Web, et sur l'autre moitié Process Monitor.

Si des événements sont déjà capturés, videz le cache de Process Explorer en cliquant que le bouton noté 4 sur [cette figure](#). Supprimez également les filtres en place s'il y en a. Pour cela, cliquez sur le bouton noté 5 sur [cette figure](#) et cliquez sur le bouton « Reset », ce qui réinitialisera les filtres.

Ensuite, démarrez la capture en cliquant sur le bouton noté 3 sur [cette figure](#).

Après avoir démarré la capture, retournez sous Firefox et modifiez manuellement la page d'accueil en suivant les étapes décrites plus tôt pour accéder aux paramètres du navigateur.

Après avoir modifié la page d'accueil, arrêtez la capture.

Le but ici est de voir où sont enregistrées les préférences de l'utilisateur. Il nous faut donc trouver l'événement sous Process Explorer qui témoigne de la modification de cette page d'accueil.

Afin de trouver le bon événement, on utilisera deux filtres :

- Un filtre d'inclusion de catégorie « Write ».
- Un filtre d'inclusion de nom de processus « firefox.exe ».

Pour savoir comment ajouter des filtres, reportez-vous à [cette partie](#) du document concernant les filtres de Process Monitor.

Après avoir été filtrés, on remarque dans les résultats de la capture que Firefox semble réaliser des opérations de création et d'écriture dans un fichier préfixé « prefs », qui correspondrait à « preferences », tel en témoigne la figure suivante :

PM	firefox.exe	6744	CreateFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs.js
PM	firefox.exe	6744	CreateFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs-1.js
PM	firefox.exe	6744	CreateFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs-1.js
PM	firefox.exe	6744	CreateFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs-1.js
PM	firefox.exe	6744	WriteFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs-1.js
PM	firefox.exe	6744	WriteFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs-1.js
PM	firefox.exe	6744	WriteFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs-1.js
PM	firefox.exe	6744	WriteFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs-1.js
PM	firefox.exe	6744	WriteFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs-1.js
PM	firefox.exe	6744	SetRenameInformationFile	C:\Users\Thomas\AppData\Roaming\Mozilla\Firefox\Profiles\{u4c8ypq8.default-release}\prefs-1.js

Figure 17 : Partie des résultats de la capture de Firefox.

Ce fichier « prefs.js » se situe dans le répertoire du profil de l'utilisateur, ici « u4c8ypq8.default-release ». Il s'agit du profil Firefox par défaut.

La nature du fichier concorde donc bien avec nos attentes.

Création du piège

Désormais nous savons où Firefox enregistre les préférences de l'utilisateur. Nous allons maintenant créer un piège pour capturer le processus qui serait susceptible de modifier la page d'accueil, soit modifier ce fichier « prefs.js ».

Dans Process Monitor, nous allons modifier quelques filtres :

- Modifiez le filtre d'inclusion de nom de processus ayant comme valeur « firefox.exe », en passant ce filtre en mode exclusion. Puisque nous cherchons un processus autre que Firefox.
- Ajoutez un filtre d'inclusion de type « Path », ayant comme condition « contains », et comme valeur « prefs.js ». Grâce à ce filtre, nous ne
- verrons uniquement les modifications apportées à ce fichier.

La figure suivante montre les filtres finaux :

Column	Relation	Value	Action
<input checked="" type="checkbox"/> Path	contains	prefs.js	Include
<input checked="" type="checkbox"/> Category	is	Write	Include
<input checked="" type="checkbox"/> Process Name	is	firefox.exe	Exclude

Figure 18 : Filtres du piège de capture Firefox

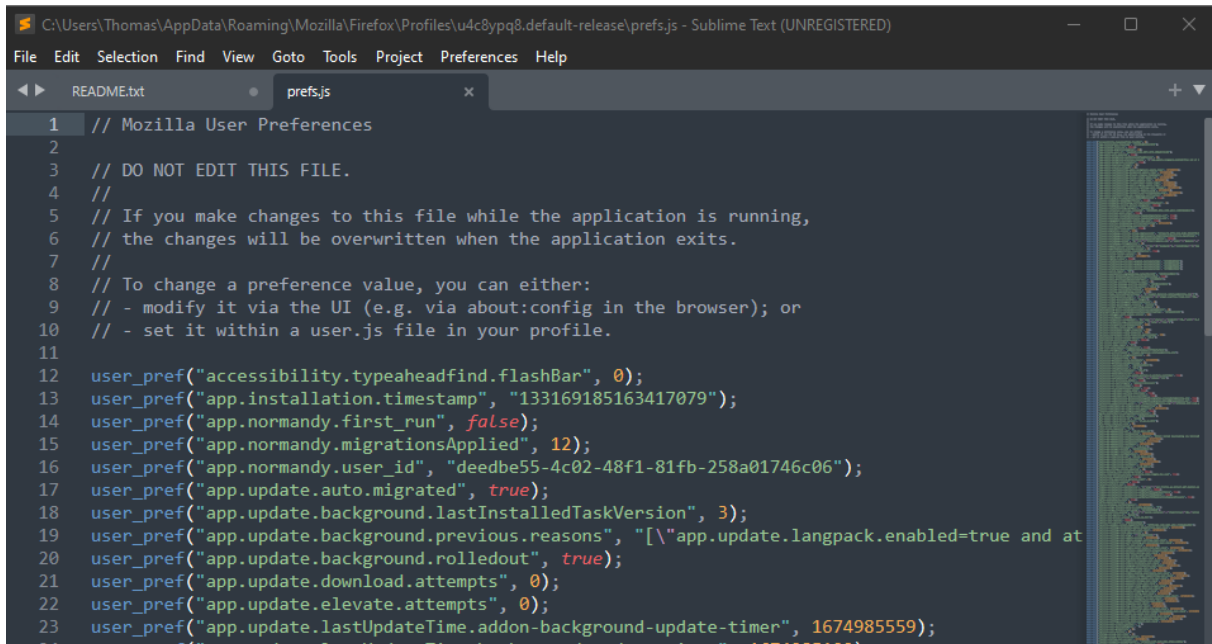
À présent, on peut redémarrer la capture. Tous les événements qui s'afficheront dans la liste seront des processus tiers qui modifient le fichier « prefs.js ».

On peut faire le test en modifiant manuellement ce fichier. Pour cela, rendez-vous dans le répertoire « % appdata %\Mozilla\Firefox\Profiles », puis ouvrez le dossier correspondant à votre profil Firefox¹⁵.

Enfin dans ce répertoire, cherchez le fichier « prefs.js », puis ouvrez-le avec n'importe quel éditeur de texte, essayons avec Sublimetext.

¹⁵ <https://support.mozilla.org/fr/kb/gestionnaire-profils-creer-supprimer-changer-profils-firefox>

Ci-dessous, la figure montre un aperçu du contenu du fichier « prefs.js » ouvert avec Sublimetext :



```

1 // Mozilla User Preferences
2
3 // DO NOT EDIT THIS FILE.
4 //
5 // If you make changes to this file while the application is running,
6 // the changes will be overwritten when the application exits.
7 //
8 // To change a preference value, you can either:
9 // - modify it via the UI (e.g. via about:config in the browser); or
10 // - set it within a user.js file in your profile.
11
12 user_pref("accessibility.typeaheadfind.flashBar", 0);
13 user_pref("app.installation.timestamp", "133169185163417079");
14 user_pref("app.normandy.first_run", false);
15 user_pref("app.normandy.migrationsApplied", 12);
16 user_pref("app.normandy.user_id", "deedbe55-4c02-48f1-81fb-258a01746c06");
17 user_pref("app.update.auto.migrated", true);
18 user_pref("app.update.background.lastInstalledTaskVersion", 3);
19 user_pref("app.update.background.previous.reasons", "[\"app.update.langpack.enabled=true and at
20 user_pref("app.update.background.rolledout", true);
21 user_pref("app.update.download.attempts", 0);
22 user_pref("app.update.elevate.attempts", 0);
23 user_pref("app.update.lastUpdateTime.addon-background-update-timer", 1674985559);
24 user_pref("app.update.lastUpdateTime.background-update-timer", 1674985100);

```

Figure 19 : Aperçu du contenu du fichier « prefs.js ».

Remarque :

Avant de modifier le fichier, vérifier que la capture sous Process Monitor est bien active !

Dès lors que nous modifions et enregistrons le fichier, un évènement s'affiche dans Process Monitor, tel que le montre la figure suivante :

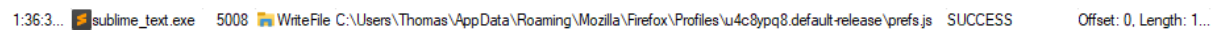


Figure 20 : Évènement de modification du fichier « prefs.js » par « sublime_text.exe ».

Notre piège fonctionne donc parfaitement ! N'importe quel processus qui apportera une modification au fichier « prefs.js » sera capturé par Process Monitor.

Dans ce cadre de test, le processus responsable de la modification est bien évidemment notre éditeur de texte favori. Mais dans un cas réel, il peut s'agir d'un tout autre processus.

Process Monitor nous indique l'exécutable, ainsi que le PID. On a alors utilisé Process Explorer pour enquêter sur ce processus et ainsi procéder au nettoyage et à la désinfection de machines.

Les différents filtres pour la mise en place du piège sont disponibles dans le sous-dossier « src » du build, portant le nom « [firefox homepage trap filters.pmc](#) ». Les résultats de ce piège sont également disponibles sous le fichier « [firefox homepage trap results.PML](#) ».

Conclusion

Ce jeu de filtres nous a effectivement permis de déceler quel processus est à l'origine de la modification de la page d'accueil de Firefox.

Dans le cadre où l'analyse est à porter sur un parc de machines, alors il serait possible d'utiliser un script PowerShell afin d'automatiser le processus. Le programme Process Explorer peut en effet être démarré de cette manière, et on peut y passer en paramètres un fichier de configuration.

Il ne reste plus qu'à redirigé les résultats de sortie dans un fichier texte, puis de laisser le script effectuer son travail. Lors de l'analyse des résultats, si l'on remarque qu'un processus X est à l'origine du changement sur plusieurs machines différentes. Alors il est possible de mener un travail de recherche post-analyse afin de procéder au nettoyage du parc informatique et à la restauration des pages d'accueil de Firefox.

Cas 2 — DNS Empoisonné

Contexte

L’helpdesk de niveau 1 reçoit un ticket d’incident. Un utilisateur de chez Vinci se plaint du changement de la page de connexion à l’extranet de l’entreprise. De ce cas présent, nous sommes en présence d’un empoisonnement de DNS (Domain Name System).

But de l’attaque

Ici le but de cette attaque est de mener l’utilisateur vers une page externe qui reproduit très fidèlement des services internes à l’entreprise, tels que la page de connexion comme décrite précédemment.

La prochaine étape est d’utiliser un serveur DNS pour rediriger l’utilisateur vers la page piégée. Le but du DNS est de lier une adresse IP à un nom de domaine. Il suffit alors de lier l’adresse IP du serveur Web sur lequel se trouve la page piégée à un nom de domaine très proche du nom de domaine utilisé en temps normal, en voici un exemple concret :

Cas	URL	Adresse du serveur Web de destination
DNS Sein	https://vinci.com	102.178.87.19
DNS Empoisonne	https://vinci.com	134.15.7.90

Le DNS empoisonné redirige la machine vers un tout autre serveur Web.

Comment repérer un DNS empoisonné ?

Un DNS empoisonné se définit comme étant un DNS non voulu par l’administrateur du système. Il s’agit donc de toute adresse IP différente de l’adresse IP du DNS souhaité.

À présent il faut repérer où Windows enregistre les paramètres DNS spécifiques à une carte réseau. Puisque ces paramètres doivent être restaurés au démarrage de la machine, ceux-ci sont stockés sur mémoire de masse. Cependant ils peuvent être stockés sous différentes formes, telles qu’une clé de registre ou un fichier de configuration par exemple.

Nous allons donc utiliser l’outil Process Explorer pour repérer où sont stockés ces paramètres et qui les modifient.

Ce procédé se scinde en quatre étapes principales :

1. Modification manuelle des paramètres
2. Identification de l’emplacement des paramètres
3. Création d’un piège de capture
4. Capture du processus à l’origine de la modification

Identifier les paramètres du DNS

Pour commencer, il faut trouver où et comment Windows enregistre les paramètres du DNS. Pour cela, on utilisera les filtres de Process Explorer pour capturer toutes les modifications des paramètres DNS.

Étant donné que les paramètres DNS sont des paramètres de bas niveau, propres à chaque interface, alors l'on peut en déduire qu'il est très probable que les paramètres DNS soient stockés sous forme de valeur, à l'intérieur d'une clé de registre.

On peut alors commencer par poser deux filtres :

- Le premier sera de type « Category » et prendra la valeur « WRITE ». Puisque l'édition du registre implique forcément des écritures sur la mémoire de masse.
- Le second sera de type « Operation » et prendra la valeur « RegSetValue ». Cette opération définit l'édition d'une valeur du registre.

La figure ci-dessous montre les filtres désormais configurés :



Column	Relation	Value	Action
<input checked="" type="checkbox"/>  Operation	is	RegSetValue	Include
<input checked="" type="checkbox"/>  Category	is	WRITE	Include

Figure 21 : Filtres de capture du DNS Empoisonné

Remarque :

Le jeu de filtres utilisé est disponible dans le fichier de configuration nommé « [poisoned_dns_filters.pmc](#) », qui se situe dans le répertoire « data » du build.

Afin de percevoir des éditions dans le registre, il faut modifier manuellement les valeurs du serveur DNS préféré. Pour cela, tapez dans le menu démarrer « Connexions Réseau ». La figure suivante montre la fenêtre des connexions réseau :

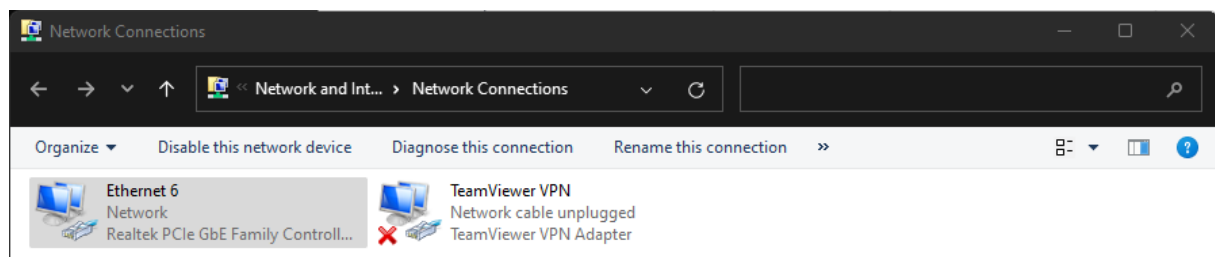


Figure 22 : Fenêtre des connexions réseau

Puis faites un clic droit sur la connexion de votre choix, dans notre cas, le choix de la connexion choisie n'importe pas sur les résultats.

Comme le montre la figure suivante, dans le menu contextuel, cliquez sur propriétés :

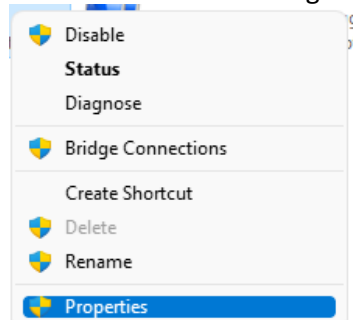
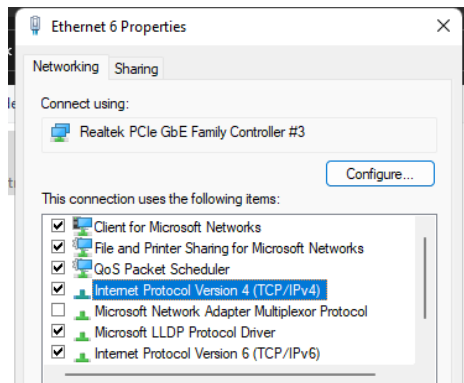


Figure 23 : Menu contextuel de la connexion

Les propriétés de l'interface apparaissent alors, double cliquez sur « Protocole Internet Version 4 » comme le montre la figure suivante :



Les paramètres IPv4 de l'interface apparaissent. Dans la nouvelle fenêtre, sélectionnez « Utiliser les paramètres DNS suivants », puis renseignez l'adresse IP d'un serveur DNS au choix. Prenons par exemple le serveur DNS de Google, tel que « 8.8.8.8 ». La figure suivante indique les étapes de configuration :

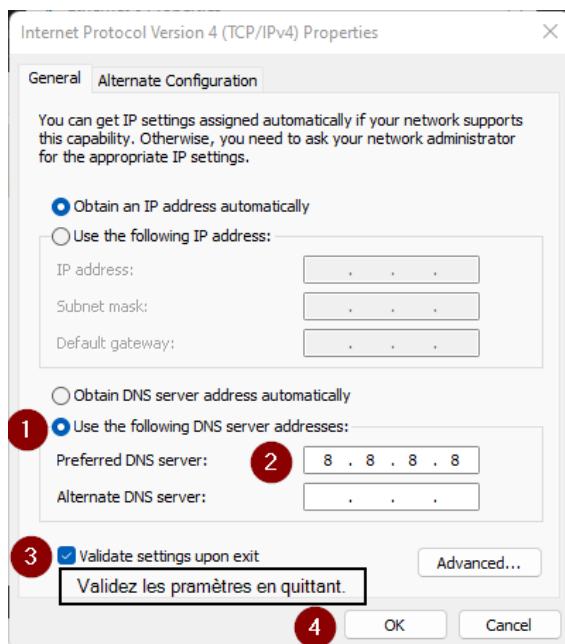


Figure 24 : Étapes de configuration manuelle du serveur DNS préféré.

Attention :

Le paramétrage d'un DNS erroné sur l'interface active résultera en l'incapacité d'accéder à Internet.

Avant de cliquer sur « OK », vérifiez que la capture sur Process Monitor est active !

Capture des résultats sous Process Monitor

Après ce paramétrage, cliquez sur « OK » sur chaque fenêtre pour valider les changements. Une fois validés, retournez sur Process Monitor et arrêtez la capture. Si les filtres ont correctement été définis, des événements devraient être apparus sous Process Monitor, tel que le montre la figure suivante :

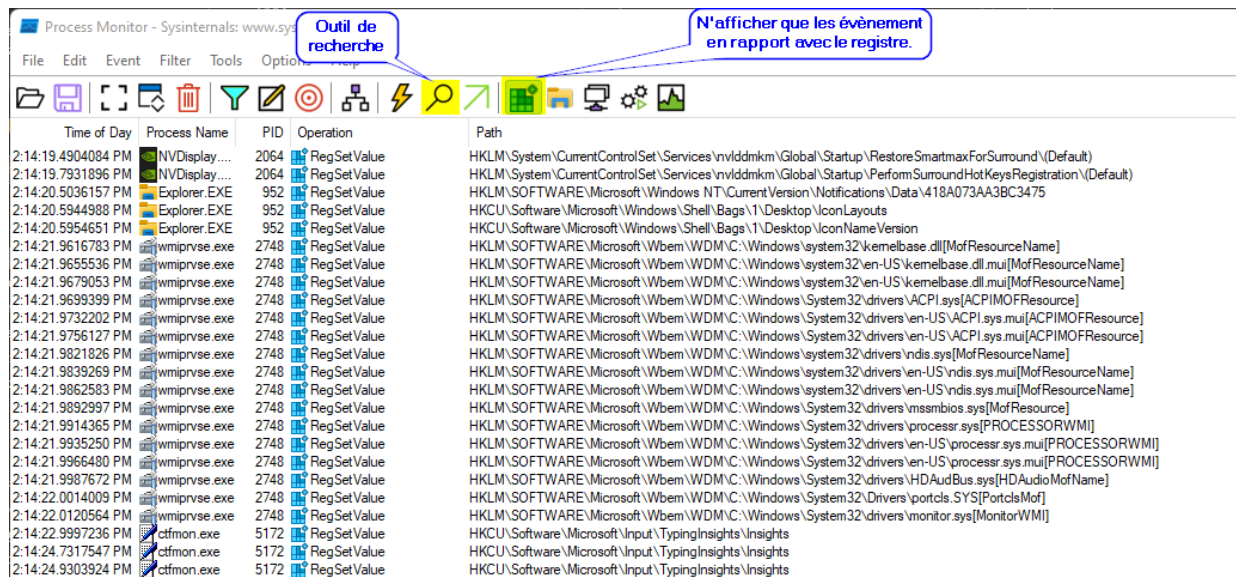


Figure 25 : Exemples d'événements capturés sous Process Monitor

Cependant un très grand nombre de ces événements ne nous sont d'aucune utilité. Nous allons alors utiliser l'outil recherché, symbolisé par l'icône de loupe, mise en valeur sur la figure ci-dessus.

Il est également possible de n'afficher uniquement les événements en lien avec le registre en ne sélectionnant uniquement l'icône bleue mise en valeur sur la figure ci-dessus.

L'outil de recherche permet d'effectuer une recherche par chaîne de caractères, attention, la recherche est sensible à la casse. La figure ci-dessous montre la fenêtre de l'outil de recherche :

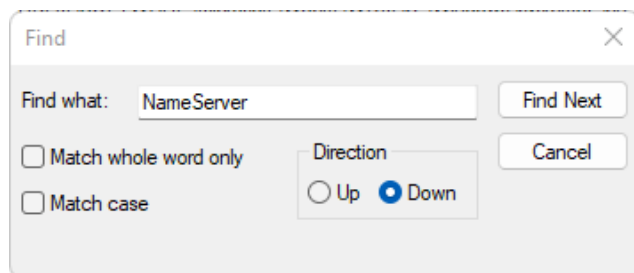


Figure 26 : Outil de recherche de Process Monitor

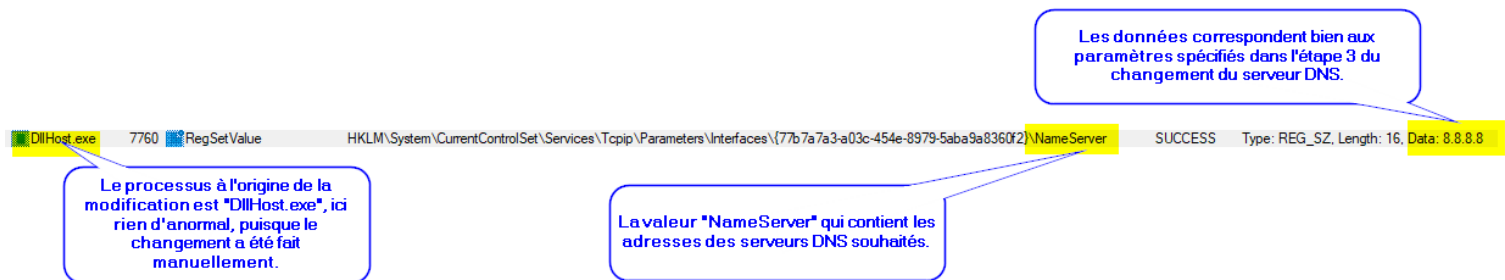
Dans le registre, la valeur « NameServer » contient la ou les adresses des DNS paramétrés pour chaque interface¹⁶. Il nous suffit alors de réaliser une recherche avec cette chaîne de caractères, tel le montre la figure ci-dessus.

Si plusieurs résultats ressortent, effectuer la même recherche affichera le prochain résultat.

¹⁶ <https://serverfault.com/questions/856244/where-does-windows-store-dns-entry-list>

Dans le cas où une valeur du registre est modifiée. Process Monitor affiche les nouvelles données dans la colonne « Detail », au côté de la mention « Data : ». Après avoir validé la recherche, il suffit alors de vérifier la colonne « Detail » et d'y trouver la mention « Data : ». L'adresse du DNS entrée plus tôt devrait y être affichée. Si tel est le cas, alors vous avez trouvé la bonne capture.

La figure ci-dessous montre à quoi devrait ressembler la capture indiquant le changement de la valeur « NameServer » du registre par la valeur du nouveau serveur DNS :



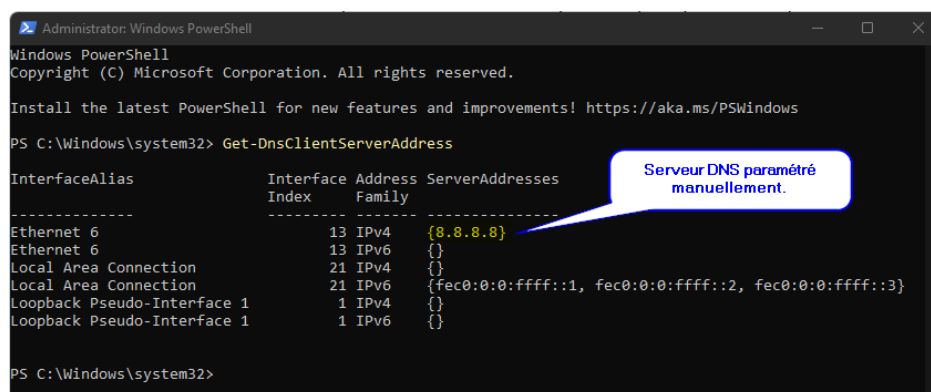
On remarque d'ailleurs que la valeur « NameServer » est contenue dans la clé de registre portant comme nom un UUID¹⁷ (Universal Unique Identifier). Cet UUID représente la connexion réseau sélectionnée dans le panneau de configuration. En outre, les paramètres sont donc propres à chaque interface réseau de l'ordinateur.

Conclusion

Dans le cas où la machine serait saine, les filtres proposés plus tôt permettent comme nous l'avons pu le démontrer de capturer quel processus précis est à l'origine du changement de DNS. À partir de là, il suffirait alors d'analyser l'exécutable à l'origine du processus et mener le nettoyage et la désinfection de la machine.

Cependant, si le DNS est déjà modifié, alors il ne sera pas possible de savoir quel processus le modifie via Process Explorer. Il faudrait alors vérifier manuellement ou via un outil tel que **PowerShell**¹⁸, la valeur actuelle du DNS paramétré.

Avec PowerShell par exemple, la commande « Get-DnsClientServerAddress » permet de lister les serveurs DNS paramétrés pour chacune des interfaces de la machine. La figure ci-dessous montre l'exécution et le résultat de cette commande :



¹⁷ https://fr.wikipedia.org/wiki/Universally_unique_identifier

¹⁸ <https://learn.microsoft.com/fr-fr/powershell/scripting/overview?view=powershell-7.3>

Détection d'un programme malveillant

Afin de détecter et supprimer du code malveillant, l'on utilise le plus souvent un antivirus. Il s'agit le plus couramment d'un programme « tout-en-un », qui scanne les fichiers et les mets dans une zone de quarantaine s'il y a une suspicion de la part de l'antivirus.

Pour procéder au scan, un antivirus utilise un « moteur de détection », il s'agit d'une procédure de détection qui permet de déterminer par diverses vérifications si un fichier est dangereux ou non. Ces moteurs parviennent à la détection via quatre méthodes, telles que :

- Détection par signature
- Détection par comportement
- Détection par contrôle de l'intégrité
- Détection par méthode heuristique

Nous allons décrire ci-dessous chacune de ces méthodes de détection.

Détection par signature

Cette méthode¹⁹ de détection consiste à repérer un programme malveillant par le biais d'une signature caractéristique. Il s'agit plus précisément d'une suite d'octets caractéristique qui permet d'identifier un fichier comme étant malveillant.

Cette méthode de détection est la plus ancienne et est de moins en moins utilisée pour être remplacée par d'autres méthodes de détection plus performantes.

Détection par comportement

Cette méthode²⁰ consiste à analyser le comportement d'un programme afin de déterminer si celui-ci est malveillant ou non.

Par exemple, reprenons le [cas numéro 1](#) portant sur la page d'accueil de Firefox. Si celle-ci venait à être modifiée par un programme autre que Firefox, alors celui-ci serait repéré comme étant un programme malveillant, de par l'analyse de son comportement.

Détection par contrôle de l'intégrité

Cette méthode consiste à détecter la présence d'un programme malveillant par le biais de la modification de fichiers ou programmes présents sur le système. En effet, cette méthode repose que le fait qu'un programme malveillant est susceptible de porter des modifications à d'autres programme.

On repère alors l'atteinte à l'intégrité de ces ressources afin de déterminer la présence d'un programme malveillant.

¹⁹ <https://www.kaspersky.fr/blog/signature-virus-disinfection/6166/>

²⁰ <https://www.malekal.com/virustotal-comportement-activite-malware/>

Détection heuristique

Cette méthode de détection²¹ est la plus complète et efficace, en effet dans le cas où un antivirus repère un fichier suspect, celui-ci le met en zone de quarantaine. Il s'agit d'une zone où le programme potentiellement malveillant est dans l'incapacité de nuire à la machine.

Dans le cas de la méthode heuristique, l'antivirus ne va pas seulement mettre le programme en quarantaine, mais viendra également simuler son exécution. C'est-à-dire qu'il va laisser le potentiel programme malveillant agir dans un endroit sécurisé et analyser son comportement.

À terme de l'analyse, si le programme présente un comportement type d'un programme malveillant, tel que le chiffrement intempestif des autres fichiers ou l'accès à des ressources protégées du système, alors l'antivirus procédera à la destruction du programme malveillant et le cataloguera dans sa base de données.

Ces quatre méthodes combinées permettent de détecter une grande majorité des programmes malveillants, bien que certains puissent encore passer au travers.

Virustotal

Virustotal est une plateforme en ligne offrant la possibilité de scanner n'importe quel fichier à l'aide d'une grande quantité de moteurs de détection afin de déterminer si le fichier est malveillant ou non.

Ces différents moteurs utilisent une partie ou toutes les [méthodes de détection](#) décrites plus haut dans le document. Les grands avantages de Virustotal sont sa facilité d'utilisation et l'importante précision de ces résultats, du fait du grand nombre de moteurs de détection utilisés durant l'analyse.

La plateforme Virustotal est accessible par l'URL donnée ci-dessous :

<https://www.virustotal.com/gui/home/upload>

La figure suivante présente la page d'accueil de Virustotal :

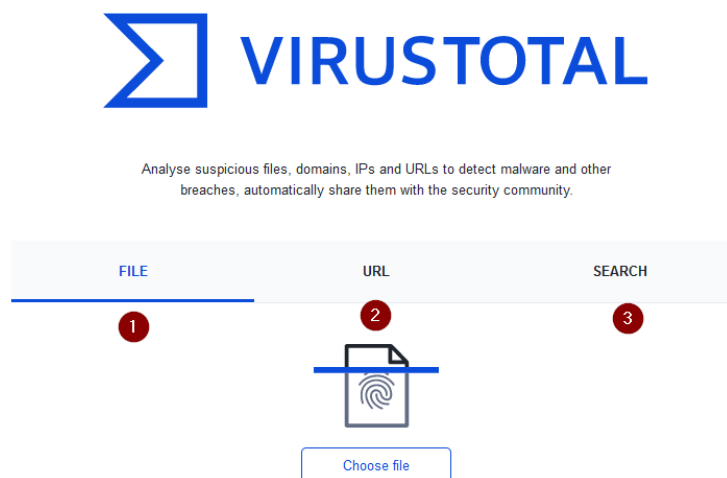


Figure 27 : Page d'accueil de Virustotal

²¹ <https://fr.malwarebytes.com/antivirus/>

Virustotal présente trois manières de scanner une ressource :

Scan de fichier

Il est possible d'envoyer directement le fichier à scanner à Virustotal. Il suffit pour cela de choisir l'option notée 1 sur [cette figure](#), puis de cliquer sur « Choose files ». Il vous sera alors proposé de choisir le fichier à envoyer. Il est également possible de directement glisser et déposer le fichier voulu.

Afin de démontrer l'utilisation de Virustotal, nous allons essayer de scanner un simple fichier texte contenant l'[empreinte](#) du virus EICAR²². Cette empreinte est utilisée à titre de tests de détection pour les antivirus, et ne comporte donc aucun risque.

La figure suivante montre le contenu du fichier texte de test :

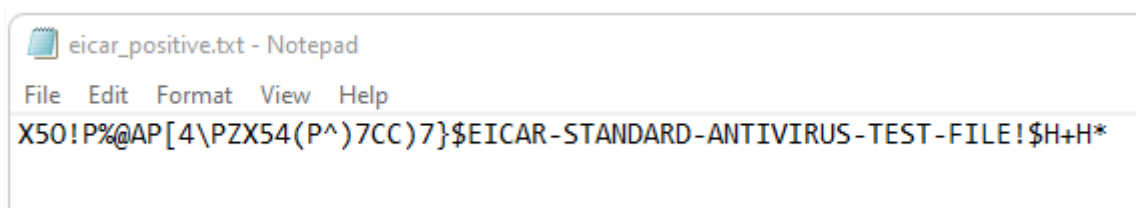


Figure 28 : Contenu du fichier de test comportant l'empreinte EICAR

Pour procéder à l'analyse, l'on suivra les étapes décrites plus haut afin d'envoyer le fichier à Virustotal.

La figure suivante montre les résultats d'analyses du fichier :

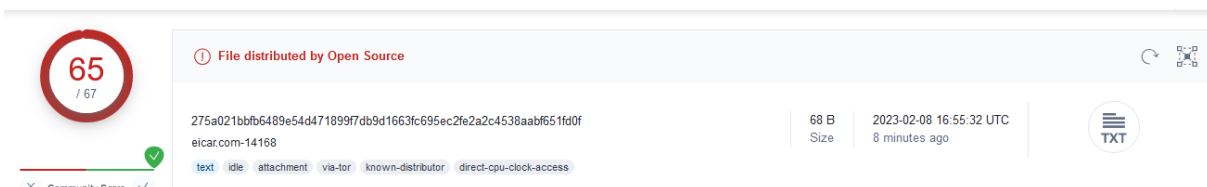


Figure 29 : Résultats d'analyses du fichier de test de détection EICAR.

On constate alors que Virustotal a correctement réagi à notre fichier de test.

²² <https://www.eicar.org/download-anti-malware-testfile/>

Scan de site Web

Virustotal propose également de scanner une URL, cela a pour but de vérifier si un site Web est légitime ou non. Pour cela, il suffit de sélectionner l'option notée 2 sur [cette figure](#), puis de renseigner l'URL du site dans le champ prévu à cet effet.

La figure ci-dessous présente le résultat d'analyse du site Web <https://youtube.com> :

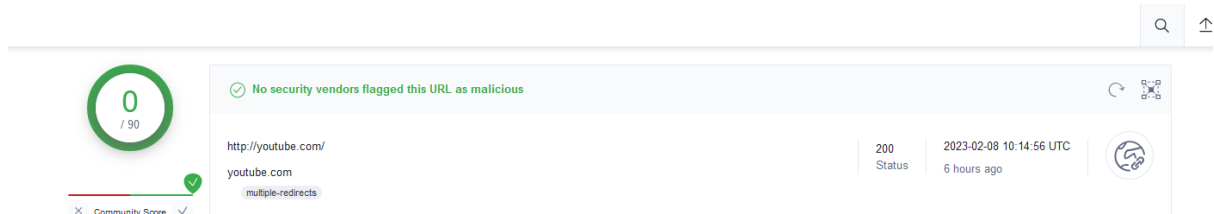


Figure 30 : Résultats d'analyse du site youtube.com sur Virustotal

Ce site Web est bien entendu parfaitement sain.

Fonctionnalité de recherche

Enfin, Virustotal propose un outil de recherche qui permet de vérifier si un programme ou fichier malveillant est déjà répertorié dans la base de données de Virustotal.

Pour effectuer une recherche, cliquez sur l'option notée 3 sur [cette figure](#), puis renseignez les mots clés de la recherche.

Conclusion

Virustotal est donc un outil gratuit et puissant permettant à n'importe qui de procéder à une analyse antivirus approfondie d'un fichier. Les résultats d'analyses tels que présentés dans les deux dernières figures représentent le nombre de moteurs de détection qui ont repéré une menace virale dans le fichier ou la ressource Web.

Un score non nul ne signifie pas obligatoirement qu'il s'agit d'un fichier malveillant. Cependant, plus le score est élevé, plus la probabilité d'une menace réelle est forte.

Automatisation et API

La partie précédente faisait mention de l'intérêt de Virustotal dans la vérification rapide et facile de fichiers. Or, il serait plus intéressant d'automatiser cette tâche afin de scanner de grandes quantités de données.

C'est alors pour répondre à cette question qu'entrent en jeu les scripts et les APIs. Avant de passer à la pratique, voyons un peu plus en profondeur ces deux notions.

Les scripts

Un script est par définition un ensemble d'instructions qui permettent d'automatiser une ou plusieurs tâches précises. Les scripts peuvent être écrits dans divers langages de programmation, comme Python²³, Batch²⁴, Bash²⁵, PowerShell²⁶, etc.

Les scripts peuvent être exécutés manuellement, via un terminal ou une invite de commande. Ou alors via un planificateur de tâches, tel que **cron**²⁷ sous Linux ou **Task Scheduler**²⁸ sous Windows.

Les scripts sont très utiles qu'en il s'agit de répéter une suite d'instruction précise et définie, et permettent donc de gagner un temps précieux.

La figure suivante montre un exemple de script qui permet de monter différents lecteurs réseau (NFS) sous Linux au démarrage de l'ordinateur :

```
#!/bin/sh

nas_ip="192.168.1.70"
dev_srv_ip="192.168.56.10"

echo "Monter le NAS ? [o/n]"
read usr_buffer

if [ "$usr_buffer" = "o" ]; then
    echo "Trying to mount NAS."
    ping -c 2 $nas_ip
    if [ $? -eq 0 ]; then
        echo "NAS Available, mounting shares..."
        mount $nas_ip:/mnt/NAS/NAS /mnt/NAS
        mount $nas_ip:/mnt/NAS/Archives /mnt/Archives
        mount $nas_ip:/mnt/NAS/Plex /mnt/Plex
    else
        echo "NAS offline or not accessible for now, please try again later..."
    fi
fi
```

Figure 31 : Exemple de script sous Linux (Shell)

²³ <https://www.python.org/doc/essays/blurb/>

²⁴ <https://www.techtarget.com/searchwindowsserver/definition/batch-file>

²⁵ <https://opensource.com/resources/what-bash>

²⁶ <https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.3>

²⁷ <https://www.freecodecamp.org/news/cron-jobs-in-linux/>

²⁸ <https://www.computerhope.com/jargon/t/taskscd.htm>

Les APIs

Une API, qui signifie (Application Programming Interface)²⁹, ou interface de programmation de l'application, permet aux développeurs de créer une application qui utilise les fonctionnalités d'une autre application.

À titre de comparaison, là où une interface utilisateur (ou interface graphique) permet à un humain d'utiliser une application. Les APIs elles, permettent à des applications d'utiliser d'autres applications.

Une API permet également aux développeurs de créer des fonctionnalités plus complexes. Il est possible grâce aux API d'ajouter une couche supplémentaire afin d'utiliser, avec une syntaxe simple, des fonctionnalités plus complexes à prendre en main.

Il existe plusieurs types d'API, dans le domaine du Web par exemple, l'on utilise les APIs REST, qui utilisent le protocole HTTP pour transmettre des informations. Les APIs SOAP, qui utilisent le langage XML et offrent une sécurité accrue par rapport à l'API REST, ou encore les APIs JavaScript, qui utilise le langage du même nom.

Si vous souhaitez découvrir les APIs plus en profondeur, je vous conseille la documentation écrite et publiée par Mozilla, qui traite de l'introduction aux API Web :

https://developer.mozilla.org/fr/docs/Learn/JavaScript/Client-side_web/APIs/Introduction

Robot de scan Virustotal

Nous pouvons dès à présent commencer à imaginer comme serait-il possible d'automatiser le processus de vérification des fichiers à l'aide du service Virustotal.

Virustotal étant un service Web, celui-ci dispose d'une API proposée par ses développeurs, d'une API REST pour être plus précis. Ce type d'API, comme mentionné dans la partie précédente abordant ce sujet, utilise le protocole HTTP afin de dialoguer entre nous et l'API.

Le centre de documentation en rapport avec cette API est disponible à l'adresse ci-contre : <https://developers.virustotal.com/reference/overview>

Cependant, nous avons besoin d'un programme client, afin d'établir le contact avec l'API. Virustotal dispose de trois bibliothèques clientes³⁰ pour utiliser cette API.

- Python
- Golang
- Java

Pour des raisons de simplicité, nous choisirons le langage Python pour coder notre « robot » de scan, utilisant donc la bibliothèque fournie par Virustotal pour utiliser l'API.

²⁹ <https://en.wikipedia.org/wiki/API>

³⁰ https://support.virustotal.com/hc/en-us/articles/360006819798-API-Scripts-and-client-libraries#h_10f07166-4521-4730-9910-da4e810ffaf1

Prérequis

Installation de Python

Afin de pouvoir faire fonctionner ce script, quelques prérequis sont nécessaires.

Premièrement, Python doit être installé sur le poste afin d'être en mesure d'exécuter des scripts écrits dans ce langage. Pour vérifier si Python est installé, tapez « **python3** » ou « **py** » dans un terminal ou invité de commande.

Si Python est installé, alors vous devriez vous trouver dans l'interpréteur Python.

La figure suivante montre le résultat attendu de la commande :

```
PS C:\Users\Thomas> py
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 32 : Interpréteur Python sous Windows

Si les deux commandes renvoient une erreur, alors Python n'est très probablement pas installé, je vous recommande d'aller voir la ressource mentionnée ci-dessous afin de procéder à l'installation de Python sur le poste :

<https://docs.python.org/fr/3/using/windows.html>

Installation des bibliothèques

Afin de fonctionner, le script a besoin de trois bibliothèques.

1. La première étant bien sûr la bibliothèque du client API Virustotal, nécessaire afin d'utiliser très simplement l'API.
2. La seconde est une bibliothèque permettant de lire les fichiers YAML³¹. Il s'agit d'un format simple permettant de structurer des données dans un fichier de manière qu'elles soient facilement compréhensibles pour un humain ainsi qu'un programme. Dans notre cas, le format YAML est utilisé pour le fichier de configuration du programme Python, permettant d'activer ou non certaines fonctionnalités.
3. La troisième bibliothèque est nécessaire afin de faire fonctionner la fonctionnalité d'alerte par SMS, détaillée dans [cette partie](#) du document.

Celles-ci n'étant pas incluses dans Python lors de l'installation, il faut les ajouter nous-mêmes pour les utiliser.

Pour procéder à l'installation, Python dispose d'un gestionnaire de bibliothèques, aussi appelées « packages », qui permet de gérer les bibliothèques installées. Ce gestionnaire se nomme « pip³² ».

Pour l'utiliser, il suffit de taper dans un terminal ou invité de commandes, la commande « python -m pip »³³, suivie de l'opération ainsi que du nom du package.

Afin d'installer les trois bibliothèques voulues, nous utiliserons les commandes suivantes :

- python -m pip install py-vt
- python -m pip install pyyaml
- python -m pip install vonage
- python -m pip install pefile

³¹ <https://www.redhat.com/en/topics/automation/what-is-yaml>

³² https://www.w3schools.com/python/python_pip.asp

³³ <https://pip.pypa.io/en/stable/installation/>

Présentation

Ce robot fonctionne de manière très simple. Afin d'automatiser le processus de scan, tous les répertoires ainsi que les fichiers à scanner sont répertoriés dans le fichier de configuration, celui-ci est lu au démarrage et tous les chemins absolus sont regroupés sous forme d'une liste, afin d'être scannés un à un.

Dans le cas d'un dossier, seuls les fichiers sont scannés. Pour des raisons de performances, les sous-dossiers sont ignorés. Car il serait par la suite difficile de contrôler quels fichiers nous souhaitons réellement scanner ?

Remarque : Depuis la version 3.1, le robot dispose d'une interface graphique permettant de simplifier son utilisation et sa configuration.

Les logs

Les logs servent à garder trace des résultats de scan des fichiers. Ce robot dispose de deux types de logs, les logs textuels simples, ainsi que les logs HTML plus complets.

Le premier a comme objectif d'être plus facilement exploitable en ligne de commande et rapide à analyser, celui-ci ne contient que le strict minimum en termes d'informations. On y trouve l'heure, la somme de contrôle³⁴ SHA256³⁵ du fichier scanné, le résultat global, donc si le fichier est sain ou non, et enfin le nom de fichier ainsi que son extension.

La figure suivante montre un exemple de log résultant d'un scan :

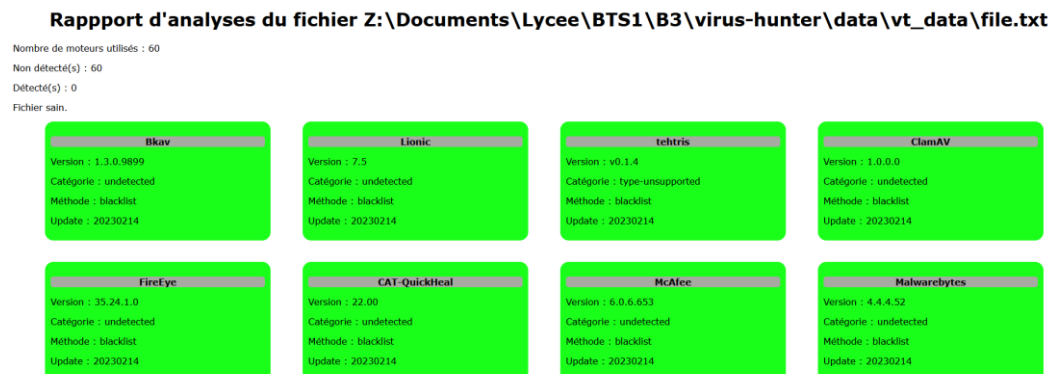
```
1 15:09:20 SHA256 > e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855 : Fichier sain (test.txt)
2 15:09:21 SHA256 > 4b9d687ac625690fd026ed4b236dad1cac90ef69e7ad256cc42766a065b50026 : Fichier sain (desktop.ini)
```

Figure 33 : Exemple de log post-scan

Les logs textuels se classés par jours, les différents fichiers sont nommés selon le jour actuel.

Le second type de logs est beaucoup plus détaillé, en effet celui-ci est au format HTML, et se visualise donc dans un navigateur Web. Ces logs répertorient le nom du fichier scanné, ainsi que toutes les informations les plus utiles en rapport avec l'analyse effectuée par Virustotal.

La figure ci-dessous montre un exemple de rapport d'analyse au format HTML :



³⁴ <https://www.malekal.com/quest-ce-qu-une-somme-de-contrôle-checksum-et-a-quoi-cela-sert/>

³⁵ <https://fr.wikipedia.org/wiki/SHA-2>

Tous les moteurs utilisés l'or du scan sont listés dans le fichier, y compris leurs noms, versions, la catégorie du scan, la méthode ainsi que la dernière mise à jour du moteur.

La catégorie correspond à la réponse globale du moteur, donc s'il détecte une présence virale ou non. La méthode correspond à la manière qu'a utilisé le moteur pour scanner le fichier, par exemple, « blacklist » signifie que le moteur a vérifié si l'empreinte du fichier n'était pas contenue dans une blacklist d'empreintes connues.

Enfin, un code couleur très simple est utilisé afin de facilement différencier les résultats. Un code couleur vert signifiant que le moteur n'a rien détecté d'anormal. Cependant un code couleur rouge indique que le moteur a repéré une présence virale, si tel est le cas, alors le type de la menace sera clairement indiqué dans les résultats d'analyse du moteur en question.

La figure ci-dessous montre un exemple de résultat positif d'un moteur :

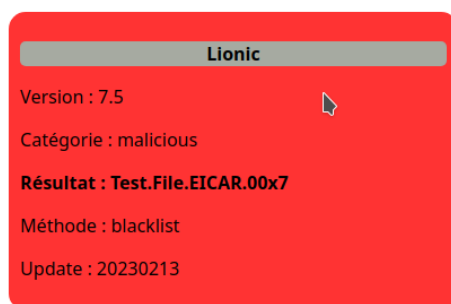


Figure 34 : Exemple de résultat positif, logs format HTML

Ici, le moteur « Lionic » a repéré le fichier de test EICAR.

À noter que les fichiers de log au format HTML sont individuels pour chaque fichier scanné.

La configuration

Le fichier de configuration permet d'affecter le fonctionnement du robot en précisant certaines fonctionnalités à activer ou non, ou encore les fichiers et dossiers à scanner.

Le fichier de configuration se nomme « conf.yaml » et se situe dans le répertoire « /data/vt_data ». Tel qu'il y est fait mention lors de [l'installation des bibliothèques](#), ce fichier est au format YAML, ce qui lui permet d'être à la fois facilement utilisable par le programme appelant, et facilement compréhensible et modifiable par l'utilisateur.

La figure ci-dessous présente le fichier de configuration permettant de configurer toutes les fonctionnalités du robot :

```
data > vt_data > conf.yaml > [ ] folderlist
Cheatsheets - Cheatsheets config file schema (cheatsheets.json)
1 | #Paramètres globaux
2 | enableDirScan: True
3 | logHealthyFiles: True
4 | createHTMLRapport: True
5 |
6 | #Paramètre du limiteur de requêtes
7 | queryThreshold: 4
8 | queryCooldown: 50
9 | enableQueryLimiter: True
10 |
11 | #Clé API Virustotal
12 | VT_API_Key: "Clé API Virustotal"
13 |
14 | #Destination des logs
15 | htmlOutputFolder: "./logs/html/"
16 | textLogsOutputFolder: "./logs/txt/"
17 |
18 | #Fichiers et dossiers à scanner
19 | filelist:
20 |   - "../data/vt_data/positive_eicar.txt"
21 |   - "../data/vt_data/test_file.txt"
22 |
23 |
24 | folderlist:
25 |   - "C:/Windows/"
26 |   - "C:/Windows/System32/"
27 |
28 |
29 | #Vonage API Settings
30 | enableSMSAlert: False
31 | Vonage_API_Key: "Clé public API Vonage"
32 | Vonage_API_Secret: "Mot de passe secret API Vonage"
33 | SMSSender: "Virustotal"
34 | receiver: "Téléphone destinataire"
35 |
```

Figure 35 : Fichier de configuration YAML du robot de scan Virustotal

On constate dès l'or que le format YAML³⁶ fonctionne principalement avec des clés et des valeurs. Une clé étant un paramètre, et sa valeur le réglage. D'autres clés telles que « filelist » ou « folderlist » ont pour valeur une liste, qui permet de stocker plus de données.

Tous les paramètres sont détaillés dans le fichier [README.txt](#), situé dans le même répertoire que le fichier de configuration.

Le limiteur de requêtes

La fonctionnalité limiteur de requêtes permet de faire une pause de x secondes tous les N requêtes. Cela permet de garantir l'obtention des résultats dans le cas de l'utilisation de la version gratuite de l'API Virustotal. En effet, la version gratuite de l'API est limitée à **4 requêtes par minute**, et **500 requêtes par jour**. Il peut alors être intéressant de limiter le flux de requêtes afin de s'assurer de la bonne obtention des résultats par l'API.

³⁶ <https://sweetohm.net/article/introduction-yaml.html>

La clé d'API

Pour utiliser l'API, nous avons besoin d'une **clé API publique**, celle-ci permet de s'authentifier auprès de l'API et ainsi de pouvoir l'utiliser et être limité ou non en fonction de notre profil utilisateur.

Pour obtenir cette clé, il vous faudra créer un compte ou vous connecter sur le site Web de Virustotal, disponible à l'adresse mentionnée ci-dessous :

<https://www.virustotal.com/gui/home/upload>

Une fois connecté, toujours sur la page d'accueil, cliquez sur votre profil, puis sur « API Key », tel qu'illustré sur la figure ci-dessous :

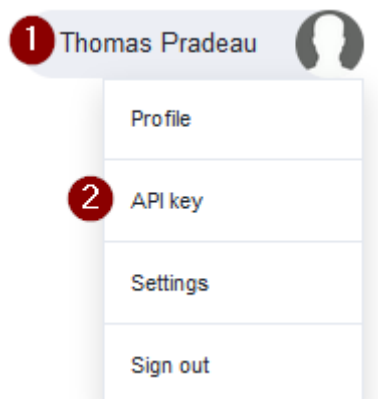


Figure 36 : Procédure d'accès à la clé d'API publique.

Enfin, la clé d'API vous sera donnée dans la rubrique « API Key », vous pourrez alors la copier dans le presse-papier afin de la coller dans la configuration du robot. Tel qu'illustré sur la figure ci-dessous :



Figure 37 : Récupérer la clé d'API publique.

Renseignez bien là votre clé dans le paramètre « VT_API_Key », du fichier de configuration, tel que le montre la figure suivante :

```
VT_API_Key: "Votre_clé_ici_entre_guillemets"
```

Figure 38 : Mise en place de la clé API dans le fichier de configuration.

En fonction de votre profil, veuillez à bien paramétrer le limiteur de requêtes afin de pleinement exploiter le robot de scan. Les limitations que vous avez sont affichées en dessous de la rubrique « API Key ».

La figure suivante montre les limitations du compte gratuit :

API quota allowances for your user

You own a standard free end-user account. It is not tied to any corporate group and so it does not have access to [VirusTotal premium services](#). You are subjected to the following limitations:

Access level	⚠ Limited , standard free public API Upgrade to premium
Usage	Must not be used in business workflows, commercial products or services.
Request rate	4 lookups / min
Daily quota	500 lookups / day
Monthly quota	15.50 K lookups / month

Figure 39 : Limitation de l'API du compte gratuit

La zone de quarantaine

Si un fichier scanné est considéré comme étant malveillant, c'est-à-dire qu'au moins un moteur a détecté une présence virale, alors celui-ci sera déplacé en zone de quarantaine.

Il s'agit d'une zone dans laquelle le fichier malveillant ne peut opérer, il s'agit également d'un endroit où les fichiers résultants d'un scan positif peuvent être conservés si l'utilisateur le souhaite.

La zone de quarantaine est un sous-dossier du répertoire des données de l'application « vt_data », nommé « quarantine ».

Alerte par SMS

Cette fonctionnalité permet d'envoyer un SMS au numéro de téléphone spécifié dans la configuration. Dans le cas où un fichier serait analysé comme malveillant, alors un SMS d'avertissement sera envoyé à l'utilisateur en y spécifiant le nom complet du fichier, ainsi que son emplacement sur le disque dur.

Pour utiliser cette fonctionnalité, la bibliothèque client de l'API Vonage doit au préalable être installée sur le poste, et ce, en suivant les étapes d'installation précisées l'or de [l'énumération des prérequis](#).

Également, la clé d'API ainsi que le mot de passe ne doivent tous deux être renseignés dans le fichier de configuration dans les champs appropriés.

Pour récupérer ces informations, il faut avoir au préalable créé un compte sur la plateforme Vonage, le lien énoncé ci-dessous permet d'y accéder :

<https://www.vonage.fr/>

Une fois le compte créé, sur la page d'accueil, cliquez sur « Se connecter ».

Puis sur la page suivante, sélectionnez « Connexion aux API de communication », et enfin renseignez vos identifiants si besoin.

Vous arriverez sur le panneau de contrôle destiné aux API de communication de Vonage, tel que le montre la figure suivante :

Vonage API Dashboard

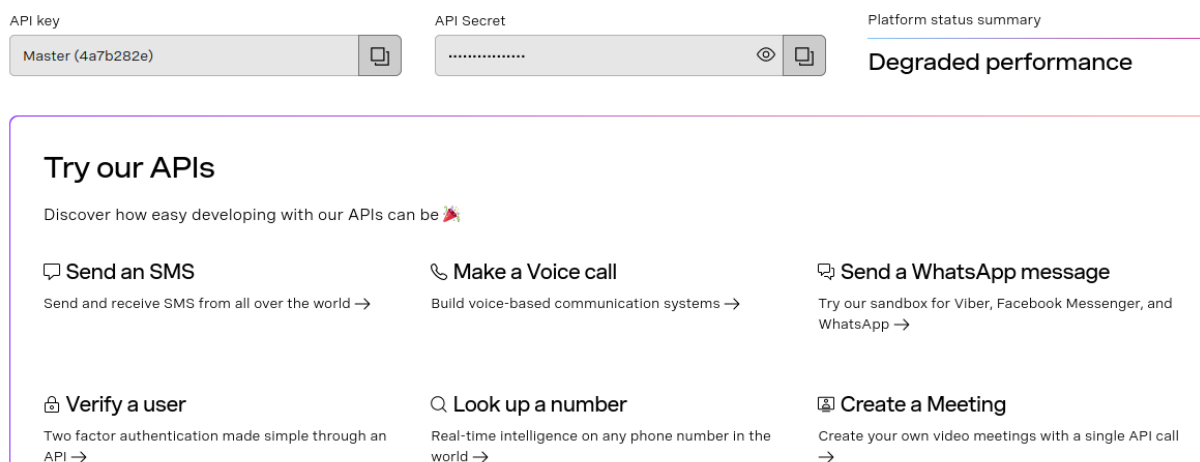


Figure 40 : Panneau de contrôle des API de communication Vonage

Tel que vous pouvez le voir sur la figure ci-dessus, une fois connecté, vous avez accès à votre clé d'API ainsi qu'au mot de passe.

Il ne vous reste plus qu'à renseigner votre numéro de téléphone dans le fichier de configuration afin de pouvoir recevoir les messages sur votre mobile.

Vous pouvez également, si vous le souhaitez, modifier le nom de l'expéditeur du message d'alerte, en modifiant le champ « SMSSender » de la configuration, le nouveau nom ne doit pas être trop long pour ne pas être tronqué.

Test en conditions réelles

Voyons maintenant le robot en fonctionnement. Nous utiliserons le fichier de configuration par défaut, tel qu'il est fourni dans la **release 2.1**, disponible dans le dépôt GitHub, disponible au lien ci-dessous :

<https://github.com/Kiwize/virus-hunter>

Pour reproduire ce test, veuillez à bien renseigner les informations en rapport avec les API Virustotal et Vonage.

Pour commencer, placez-vous dans le répertoire « src/ » avec votre terminal, puis exécutez le script « pythonDefenser.py » avec l'interpréteur Python installé.

Si les trois bibliothèques sont correctement installées, alors le scan du premier fichier constituant le jeu d'essais devrait débuter. Le jeu d'essais est constitué de deux fichiers, un fichier texte simple et un fichier texte contenant l'empreinte de test EICAR.

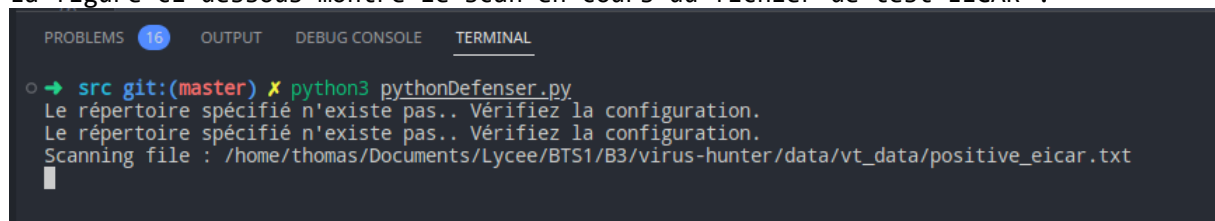
Selon la configuration, le fichier EICAR nommé « positive_eicar.txt » est analysé en premier, suivi de l'autre fichier texte, nommé « test_file.txt ».

Important : L'API Virustotal peut rencontrer des difficultés à scanner un fichier, si tel est le cas, alors le robot enverra le message d'erreur suivant en console :

« Erreur l'or du scan, veuillez réessayer. »

Dans ce cas-là, attendez quelques minutes avant de réitérer le scan.

La figure ci-dessous montre le scan en cours du fichier de test EICAR :



```

PROBLEMS 16 OUTPUT DEBUG CONSOLE TERMINAL
○ → src git:(master) X python3 pythonDefenser.py
Le répertoire spécifié n'existe pas.. Vérifiez la configuration.
Le répertoire spécifié n'existe pas.. Vérifiez la configuration.
Scanning file : /home/thomas/Documents/Lyce/BTS1/B3/virus-hunter/data/vt_data/positive_eicar.txt

```

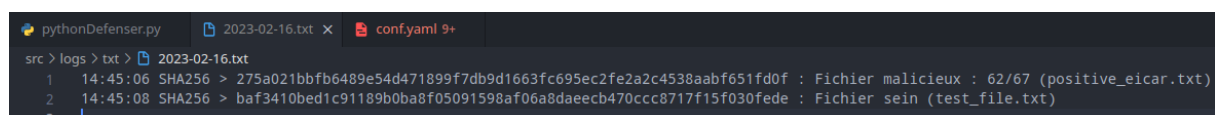
Figure 41 : Scan du fichier de test EICAR par Virustotal par l'API

Après quelque temps, le scan devrait de compléter. Il est possible que l'API mette du temps à répondre, tant qu'aucun message d'erreur n'est apparu en console, laisser le scan continuer.

Analyse des logs textuels

Une fois les scans effectués, nous trouvons les résultats des scans, dans les logs textuels et au format HTML.

Analysons le contenu du fichier de logs textuels, la figure suivante montre le contenu au terme du scan des deux fichiers du jeu d'essais :



```

pythonDefenser.py 2023-02-16.txt x conf.yaml 9+
src > logs > txt > 2023-02-16.txt
1 14:45:06 SHA256 > 275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f : Fichier malicieux : 62/67 (positive_eicar.txt)
2 14:45:08 SHA256 > baf3410bed1c91189b0ba8f05091598af06a8daeeeb470ccc8717f15f030fede : Fichier sein (test_file.txt)
3

```

Figure 42 : Contenu du fichier de logs textuels au terme du scan des fichiers du jeu d'essais.

On constate que le fichier « positive_eicar.txt » a bien été indiqué comme malveillant, et l'autre fichier est un fichier sain. Les résultats sont donc corrects en tenant compte du jeu d'essais utilisé.

Analyse des logs HTML

Regardons maintenant les logs au format HTML. La figure suivante montre les logs HTML générés pour donner suite à l'analyse des fichiers du jeu d'essais :

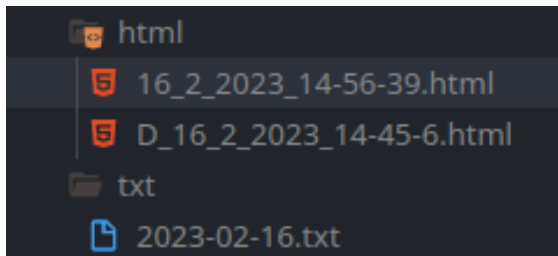


Figure 43 : Logs HTML générés à la suite de l'analyse des fichiers du jeu d'essais.

On constate que deux fichiers ont été générés, l'un commençant par la lettre « D » et non l'autre. Cela est parfaitement normal et considéré comme le contenu du jeu d'essais. Tous les logs HTML commençant par la lettre « D » signifient que le résultat est positif, c'est donc que le fichier scanné est malveillant.

Ici les deux fichiers de logs sont bien cohérents avec le contenu du jeu d'essais. La figure suivante montre le contenu du fichier de logs HTML du fichier « positive_eicar.txt » :

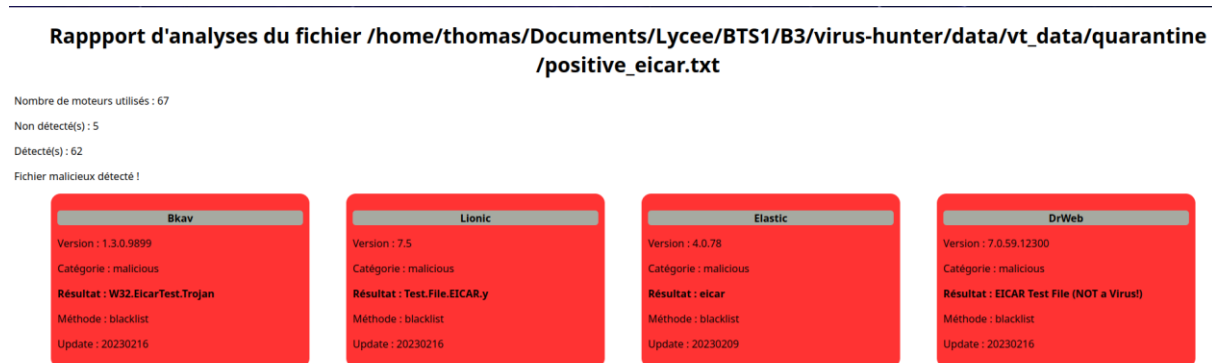


Figure 44 : Résultats du scan du fichier de test EICAR

Les figures ci-dessous montrent les résultats du scan précis :

Nombre de moteurs utilisés : 67

Non détecté(s) : 5

Détecté(s) : 62

Fichier malicieux détecté !

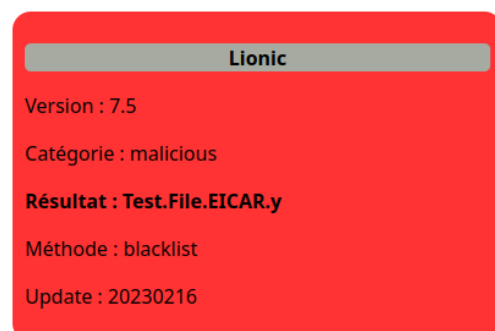


Figure 45 : Résultats précis du scan du fichier de test EICAR

On constate également d'après le nom du fichier indiqué dans l'avant-dernière figure que le fichier malveillant a bien été mis dans l'espace de quarantaine.

La figure suivante montre le contenu du dossier de l'espace de quarantaine, après le scan des fichiers du jeu d'essais :

```
→ virus-hunter git:(master) x ls -l data/vt_data/quarantine
total 4
-rw-rw-r-- 1 thomas thomas 68 févr. 15 09:55 positive_eicar.txt
→ virus-hunter git:(master) x cat data/vt_data/quarantine/positive_eicar.txt
X50!P%AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*%&
```

Figure 46 : Aperçu du fichier malveillant dans la zone de quarantaine

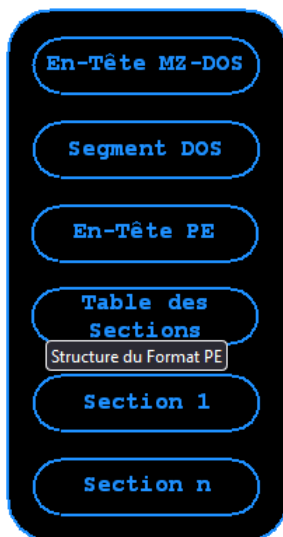
Analyse approfondie d'un fichier

Contexte

Afin de savoir si un fichier est malveillant ou non, il est possible d'analyser le contenu du fichier afin de déterminer si le code qu'il contient est malveillant ou non. Tel que nous l'avons vu dans la partie précédente de ce document, où nous avons utilisé le service Virustotal afin de procéder à cette vérification.

Il est également possible de vérifier la provenance d'un fichier, ainsi que sa nature, et ce afin d'en déduire sa légitimité. On vérifiera donc éventuellement l'auteur du code, et si possible, l'organisation pour laquelle travaille l'auteur du programme.

Structure du fichier — PE



Un fichier « PE » ou « Portable Executable³⁷ » est une structure de fichier utilisé par d'autres formats de fichier, tels que les fichiers « EXE » ou « DLL ». Cette structure permet de normaliser la structure des fichiers exécutables, afin qu'ils soient capables de fonctionner sur n'importe quel poste utilisant le noyau Windows NT ou MS-DOS (Windows 9x)³⁸.

Le développeur du programme n'a ainsi qu'à mettre une condition sur la version minimale de Windows nécessaire pour faire fonctionner le programme. La figure ci-contre montre la manière dont est segmenté un fichier PE.

Figure 47 : Segmentation d'un fichier au format PE (Portable Exécutable)

Voici la description simplifiée des parties les plus importantes constituant le format PE :

- L'entête MS-DOS permet de reconnaître le fichier comme étant un exécutable valide dans le cas où le fichier serait lancé dans un environnement MS-DOS.
- Le segment DOS est exécuté si Windows ne reconnaît pas le fichier comme étant au format PE valide.
- L'entête PE contient toutes les informations importantes en rapport avec le fichier, tel que la « signature » permettant d'identifier le fichier ou le « FileHeader », contenant les informations en rapport avec la structure du fichier.
- La table des sections contient toutes les informations en rapport avec les différents binaires constituant le fichier, devant être chargée en mémoire.

³⁷ https://fr.wikipedia.org/wiki/Portable_Executable

³⁸ <https://www.toutwindows.com/historique-de-windows/>

- Les sections contiennent différents types de données du programme, telles que le code à exécuter, les variables initialisées ou les ressources du fichier.

Tous les fichiers dérivant de la structure PE sont architecturés de la même façon. Cela permet notamment de pouvoir traiter le contenu des fichiers dérivant de cette architecture de manière normalisée pour le développeur.

Structure de fichier — EXE

Le fichier exécutable « EXE » utilisant la structure PE vue précédemment est le type de fichier exécutable le plus couramment utilisé. Ce type de fichier est utilisé pour exécuter un programme, plus précisément, le code assembleur contenu dans le fichier est chargé en mémoire, puis exécuté par le processeur.

Contrairement à un fichier DLL (Dynamic Link Library), un fichier exécutable peut embarquer toutes ces dépendances, et peut ainsi être exécuté de manière autonome. Un exécutable possède ainsi la possibilité de créer un nouveau processus, c'est-à-dire de lancer un programme. Là où une DLL ne peut pas être exécutée individuellement.

Les DLL sont utilisées par les fichiers exécutables. Une DLL contient du code compilé, donc de l'assembleur, qui peut être utilisé par un autre programme. Autrement dit, un fichier exécutable.

Les fichiers exécutables suivent donc la structure des exécutables portables. Il est possible de la constater en analysant le contenu d'un fichier exécutable. Cela est possible avec un logiciel spécialisé, tel que « PEView³⁹ », qui permet d'analyser le contenu de n'importe quel fichier PE.

La figure ci-dessous montre l'aperçu du fichier « notepad.exe », ouvert dans le programme PEView :

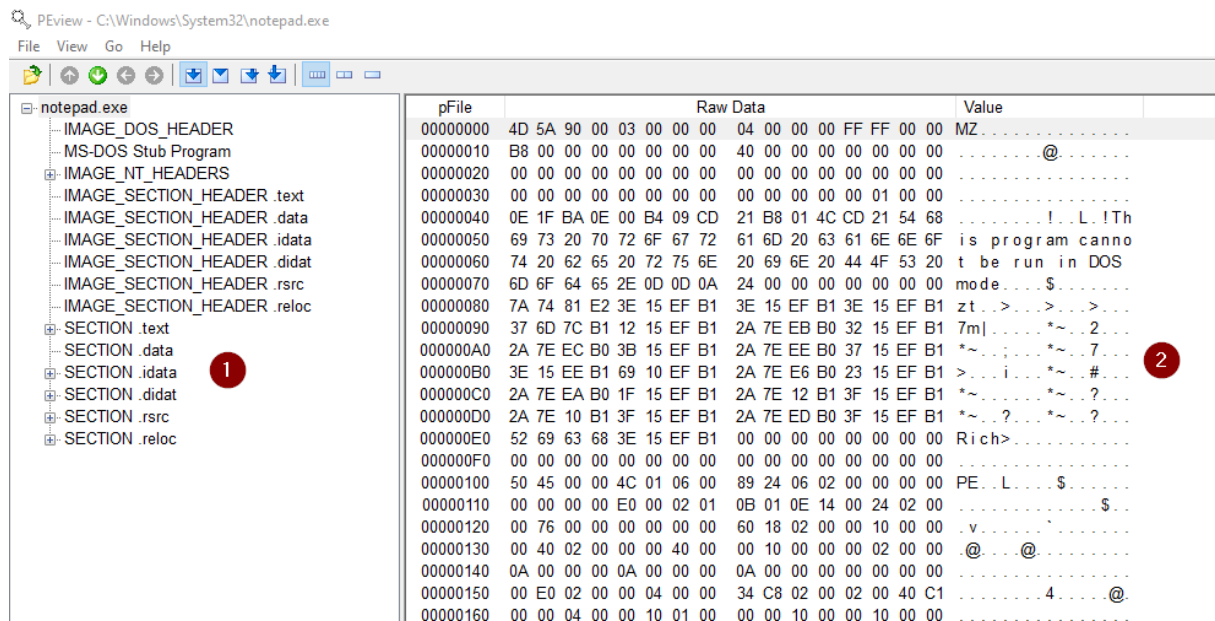


Figure 48 : Aperçu du fichier « notepad.exe » ouvert dans le logiciel « PEView ».

On peut voir sur la droite de la figure, noté 1, la structure du fichier, tel que décrite sur [cette figure](#).

L'autre partie de la fenêtre montre le contenu de la partie de l'exécutable sélectionnée dans la partie 1 de la figure. Le contenu est affiché en hexadécimal.

Un programme plus avancé tel que IDA permet de décompiler le fichier exécutable afin de pouvoir étudier le code assembleur.

³⁹ <http://wjradburn.com/software/>

La figure ci-dessous montre un aperçu du code assembleur contenu dans l'une des sections précédemment illustrées du fichier exécutable « notepad.exe » :

```
.text:00000000140012BA1      mov     edi, 50h ; 'P'
.text:00000000140012BA6      lea     rcx, ?FR@@3UtagFINDREPLACEW@@@A ; void *
.text:00000000140012BAD      mov     r8d, edi ; Size
.text:00000000140012BB0      xor     edx, edx ; Val
.text:00000000140012BB2      call   memset_0
.text:00000000140012BB7      mov     rax, cs:?hwndNP@@@3PEAUHWND_@@EA ; HWND__ * hwndNP
.text:00000000140012BBE      lea     r9, [rbp+1170h+1Param] ; lParam
.text:00000000140012BC5      mov     rcx, cs:?hwndEdit@@@3PEAUHWND_@@EA ; hwnd
.text:00000000140012BCC      lea     r8, [rbp+1170h+1Param+4] ; wParam
.text:00000000140012BD3      lea     edx, [rdi+60h] ; Msg
.text:00000000140012BD6      mov     cs:?FR@@3UtagFINDREPLACEW@@@A.hwndOwner, rax ; tagFINDREPLACEW FR
.text:00000000140012BDD      mov     cs:?FR@@3UtagFINDREPLACEW@@@A.lStructSize, edi ; tagFINDREPLACEW FR
.text:00000000140012BE3      call   cs:__imp_SendMessageW
```

Figure 49 : aperçu de code assembleur contenu dans le fichier « notepad.exe ».

Le code assembleur

Dans la figure ci-dessus, l'on se rend compte que l'assembleur est un langage très bas niveau comparé à d'autres langages tels que le Java, Python ou C++.

L'assembleur peut ainsi être exécuté très facilement par le processeur. Le procédé visant à traduire les langages de programmation de plus haut niveau à l'assembleur se nomme « compilation », c'est une couche de traduction entre la machine et le développeur.

Remarque : Il n'est pas possible de « décompiler » de l'assembleur en code de plus haut niveau. Mais l'on peut cependant le déduire en étudiant le fonctionnement du code assembleur, et ainsi reconstituer le code de plus haut niveau. On réalise alors du « reverse engineering », ou ingénierie inversée.

Il est donc possible de découvrir des failles exploitables d'un programme par le biais de la décompilation, bien qu'il s'agisse d'un procédé très chronophage et incertain, cela reste une possibilité.

Un exécutable, programme légitime ou malware ?

Dans les précédentes parties de ce document, nous avons pu entrevoir la structure d'un fichier exécutable Windows par l'utilisation de programmes permettant de mettre au jour les entrailles de ces fichiers.

Cela n'est cependant pas une tâche aisée à réaliser pour des novices et l'interprétation des données contenues peut porter à confusion. C'est pour cela que l'on préférera une autre méthode afin de vérifier la légitimité d'un programme. Nous allons donc maintenant parler de la signature de code !

Le chiffrement

Le chiffrement est une méthode utilisée pour protéger les informations confidentielles en les transformant de manière qu'elles ne soient pas accessibles à des personnes non autorisées. Cela se fait en utilisant des algorithmes de chiffrement qui transforment les données claires, ou non chiffrées, en données chiffrées, ou cryptées, qui sont difficiles à comprendre sans la clé de déchiffrement appropriée.

Il existe deux types de chiffrement couramment utilisés : le chiffrement symétrique et le chiffrement asymétrique.

Le chiffrement symétrique, également appelé chiffrement à clé secrète, utilise une seule clé pour chiffrer et déchiffrer les données. Cette clé doit être partagée entre les parties qui souhaitent communiquer de manière sécurisée. Le chiffrement symétrique est rapide et efficace pour chiffrer de grandes quantités de données, mais il présente un risque de sécurité si la clé est compromise. Si une personne non autorisée obtient la clé, elle peut accéder aux données chiffrées. Les algorithmes de chiffrement symétrique couramment utilisés incluent AES⁴⁰, DES⁴¹ et Blowfish⁴².

Le chiffrement asymétrique, également appelé chiffrement à clé publique, utilise deux clés différentes, une clé publique et une clé privée. La clé publique est utilisée pour chiffrer les données, tandis que la clé privée est utilisée pour déchiffrer les données. La clé publique peut être distribuée librement, tandis que la clé privée doit être gardée secrète. Le chiffrement asymétrique est plus lent que le chiffrement symétrique, mais il offre une meilleure sécurité, car la clé privée n'est pas partagée et ne peut donc pas être compromise. Les algorithmes de chiffrement asymétrique couramment utilisés incluent RSA⁴³, DSA⁴⁴ et ECC⁴⁵.

⁴⁰ https://en.wikipedia.org/wiki/Advanced_Encryption_Standards

⁴¹ https://fr.wikipedia.org/wiki/Data_Encryption_Standard

⁴² <https://fr.wikipedia.org/wiki/Blowfish>

⁴³ <https://www.techtarget.com/searchsecurity/definition/RSA>

⁴⁴ <https://www.simplilearn.com/tutorials/cryptography-tutorial/digital-signature-algorithm>

⁴⁵ https://en.wikipedia.org/wiki/Elliptic-curve_cryptography

Les figures ci-dessous montrent schématiquement le fonctionnement du chiffrement symétrique et asymétrique :

Chiffrement symétrique

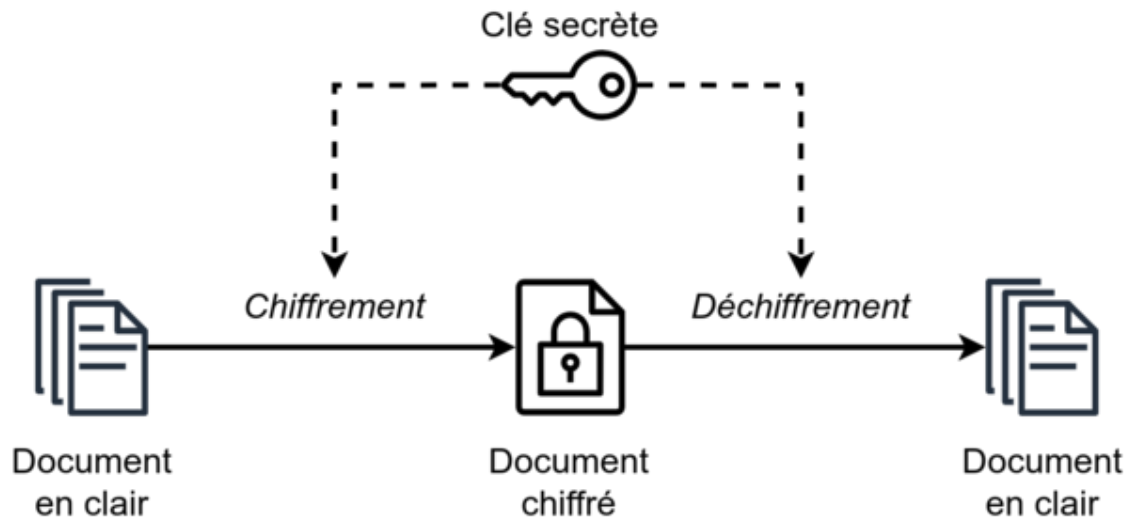


Figure 50 : Fonctionnement du chiffrement symétrique

Chiffrement asymétrique

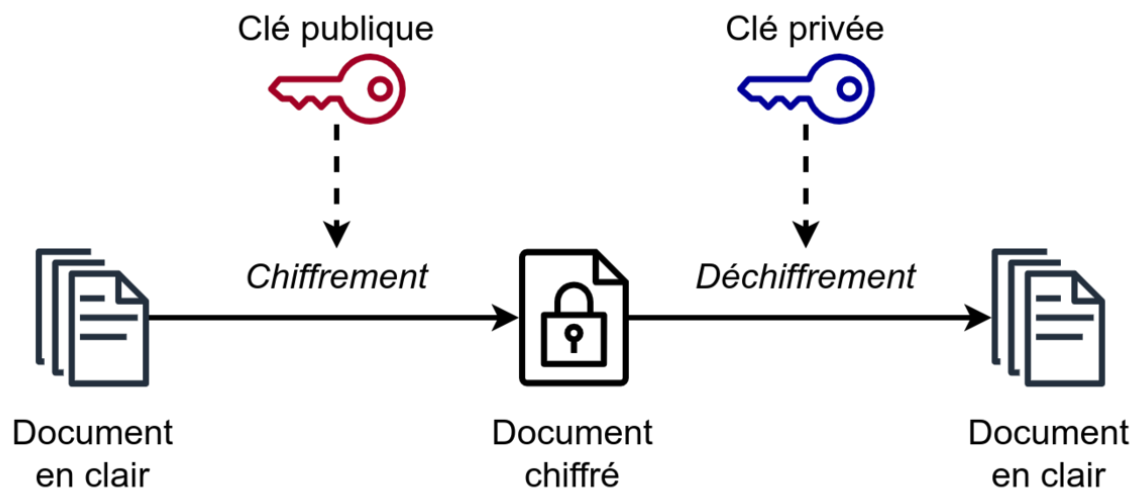


Figure 51 : Fonctionnement du chiffrement asymétrique

En résumé, le chiffrement est une méthode essentielle pour protéger les données confidentielles. Le chiffrement symétrique est rapide et efficace pour chiffrer de grandes quantités de données, mais il présente un risque de sécurité si la clé est compromise. Le chiffrement asymétrique est plus lent, mais il offre une meilleure sécurité, car la clé privée n'est pas partagée.

Les algorithmes de hash

Une autre composante essentielle de la signature de code est le hash. Un algorithme de hachage est une fonction mathématique qui prend en entrée une quantité de données quelconque (un fichier, un message, une image, etc.) et qui produit en sortie une empreinte numérique unique de cette entrée, appelée « hash » ou « hachage ». Cette empreinte numérique est généralement représentée sous forme d'une chaîne de caractères hexadécimaux⁴⁶ ou en base64.

Les algorithmes de hachage ont de nombreuses applications dans le domaine de la sécurité informatique. Voici quelques exemples d'utilisation des algorithmes de hachage :

1) Stockage de mots de passe :

Au lieu de stocker les mots de passe en clair, les systèmes d'authentification stockent souvent l'empreinte numérique des mots de passe en utilisant un algorithme de hachage. Lorsqu'un utilisateur tente de s'authentifier, le système compare l'empreinte numérique du mot de passe entré avec l'empreinte stockée pour vérifier si les deux correspondent.

2) Vérification de l'intégrité des données :

Les algorithmes de hachage sont souvent utilisés pour vérifier l'intégrité des fichiers et des messages. Si le hash d'un fichier ou d'un message change, cela indique que le fichier ou le message a été modifié.

3) Signature numérique :

Les signatures numériques utilisent souvent des algorithmes de hachage pour créer une empreinte numérique du document ou du message à signer. La signature numérique est ensuite créée en chiffrant l'empreinte numérique à l'aide de la clé privée du signataire.

4) Vérification de la provenance :

Les algorithmes de hachage sont également utilisés pour vérifier la provenance des fichiers et des messages. Si le hash d'un fichier ou d'un message correspond à celui fourni par l'expéditeur, cela indique que le fichier ou le message provient de l'expéditeur légitime.

Les algorithmes de hachage les plus couramment utilisés sont MD5⁴⁷, SHA-1, SHA-2 (qui inclut SHA-256 et SHA-512) et SHA-3⁴⁸. Cependant, certaines de ces fonctions ont été compromises ou considérées comme insuffisamment robustes face à des attaques, et il est donc recommandé d'utiliser les versions les plus récentes et les plus sûres des algorithmes.

Il est important de noter que les algorithmes de hachage ne sont pas inviolables et peuvent être sujets à des attaques de type « collision »⁴⁹ où deux entrées différentes produisent la même empreinte numérique, mais ces attaques sont difficiles à mettre en place dans la pratique. Il est également important de protéger la clé privée utilisée pour chiffrer les empreintes numériques pour éviter toute compromission de la sécurité de l'algorithme de hachage.

⁴⁶ https://fr.wikipedia.org/wiki/Syst%C3%A8me_hexad%C3%A9cimal

⁴⁷ <https://en.wikipedia.org/wiki/MD5>

⁴⁸ https://fr.wikipedia.org/wiki/Secure_Hash_Algorithm

⁴⁹ <https://freemanlaw.com/hash-collisions-explained/>

La figure suivante montre le cas concret de l'utilisation des algorithmes de hachage :

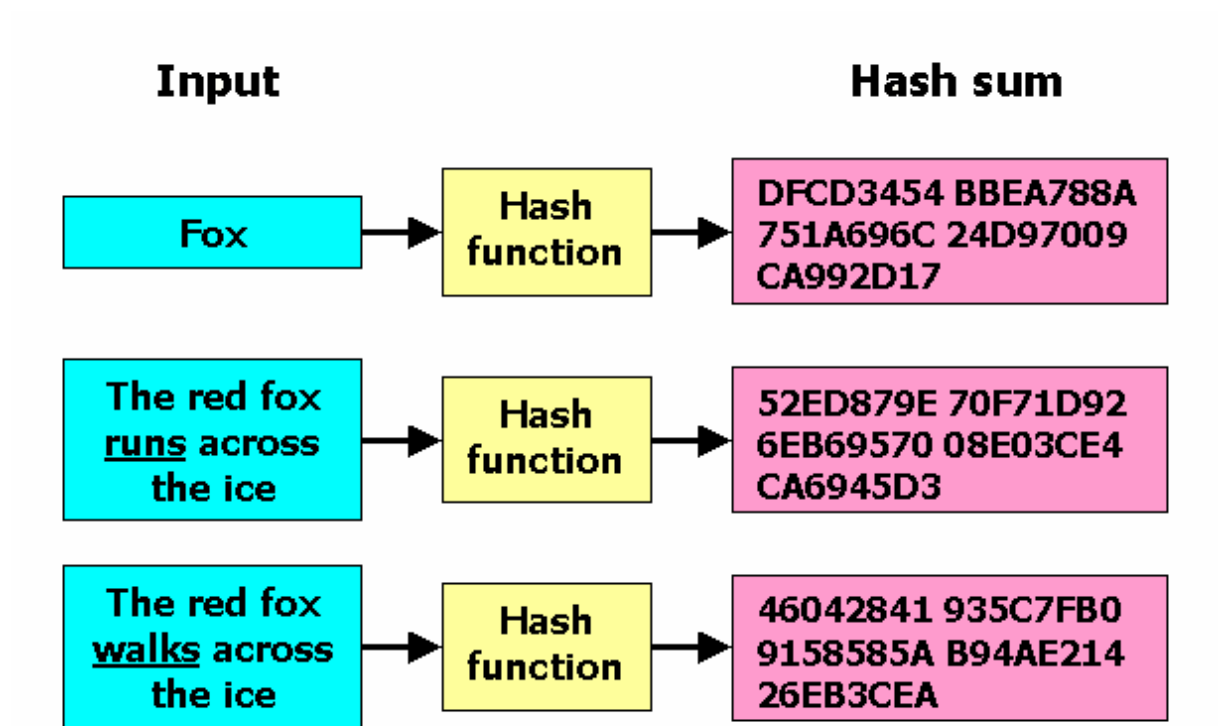


Figure 52 : Cas concret d'utilisation des algorithmes de hachage.

La signature de code

La signature de code, également appelée signature numérique, est une méthode utilisée pour garantir l'authenticité et l'intégrité des logiciels et des fichiers exécutables. Elle permet aux utilisateurs de vérifier que le logiciel qu'ils téléchargent a été créé par un éditeur de confiance et qu'il n'a pas été modifié par des tiers malveillants.

La signature de code utilise un algorithme de hachage pour créer une empreinte numérique unique du logiciel ou du fichier exécutable. Cette empreinte numérique est ensuite chiffrée à l'aide de la clé privée de l'éditeur pour créer une signature numérique. La signature numérique est ensuite incluse dans le logiciel ou le fichier exécutable.

Lorsqu'un utilisateur tente d'exécuter le logiciel ou le fichier exécutable, le système d'exploitation vérifie la signature numérique à l'aide de la clé publique de l'éditeur. Si la signature numérique est valide, le système d'exploitation permet à l'utilisateur d'exécuter le logiciel ou le fichier exécutable en toute confiance. La signature de code offre plusieurs avantages. Tout d'abord, elle garantit que le logiciel ou le fichier exécutable provient d'un éditeur de confiance, ce qui permet aux utilisateurs de télécharger des logiciels en toute sécurité. De plus, elle garantit que le logiciel ou le fichier exécutable n'a pas été modifié depuis sa publication initiale, ce qui prévient les attaques de type « man in the middle » ou les attaques de modification de logiciel malveillant.

Cependant, il est important de noter que la signature de code n'est pas une garantie absolue de sécurité. Bien que la signature de code puisse prévenir de nombreuses attaques, elle ne peut pas protéger contre toutes les formes d'attaques de logiciels malveillants. De plus, la clé privée utilisée pour créer la signature numérique doit être protégée de manière appropriée pour garantir l'authenticité de la signature numérique.

En résumé, la signature de code est une méthode importante pour garantir l'authenticité et l'intégrité des logiciels et des fichiers exécutables. Elle permet aux utilisateurs de vérifier que le logiciel qu'ils téléchargent a été créé par un éditeur de confiance et qu'il n'a pas été modifié par des tiers malveillants. Cependant, il est important de noter que la signature de code n'est pas une garantie absolue de sécurité et doit être utilisée en conjonction avec d'autres méthodes de sécurité pour une protection optimale.

Les certificats de signature de code

Un certificat de signature de code est un fichier numérique qui contient des informations sur l'identité du développeur ou de l'éditeur d'un logiciel, ainsi qu'une signature numérique pour prouver que le logiciel n'a pas été modifié depuis sa création. Les certificats de signature de code sont utilisés pour garantir l'intégrité et l'authenticité des logiciels, ce qui est particulièrement important pour les logiciels téléchargés sur Internet.

Les **autorités de certification**⁵⁰ (CA pour « Certificate Authority » en anglais) sont des organisations qui délivrent des certificats de signature de code. Ces organisations vérifient l'identité du développeur ou de l'éditeur avant de délivrer un certificat de signature de code. Les autorités de certification sont généralement des entreprises privées, mais il existe également des autorités de certification publiques, qui sont généralement liées à des gouvernements.

Lorsqu'un développeur souhaite signer un logiciel avec un certificat de signature de code, il soumet une demande à une autorité de certification. Cette demande doit contenir des informations sur l'identité du développeur, ainsi que des informations sur le logiciel à signer. L'autorité de certification vérifie l'identité du développeur et valide la demande avant de délivrer le certificat de signature de code.

Une fois le certificat de signature de code obtenu, le développeur peut l'utiliser pour signer le logiciel. La signature numérique du certificat est alors ajoutée au logiciel, ce qui garantit que le logiciel n'a pas été modifié depuis la création du certificat. Lorsqu'un utilisateur télécharge le logiciel, son système d'exploitation vérifie la signature numérique pour s'assurer que le logiciel est authentique et n'a pas été modifié.

Les certificats de signature de code et les autorités de certification sont essentiels pour garantir la sécurité des logiciels téléchargés sur Internet. En effet, sans ces certificats, il serait difficile pour les utilisateurs de savoir si un logiciel est authentique et n'a pas été modifié depuis sa création. Les certificats de signature de code et les autorités de certification contribuent donc à renforcer la confiance des utilisateurs dans les logiciels téléchargés sur Internet.

La figure suivante montre l'implémentation d'un certificat de signature de code :

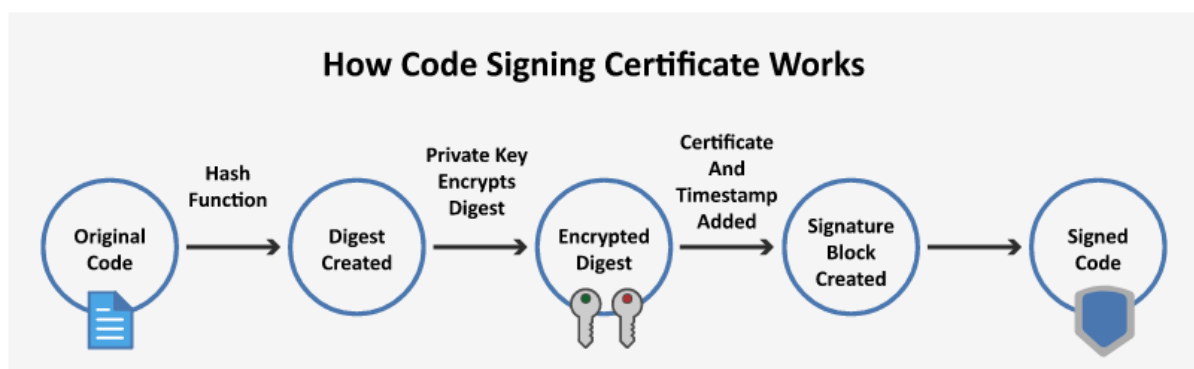


Figure 53 : Schéma de l'implémentation d'un certificat de signature de code.

⁵⁰ <https://www.ssl.com/fr/faq/qu%27est-ce-qu%27une-autorit%C3%A9-de-certification/>

En résumé, les certificats de signatures de codes permettent de vérifier l'identité du développeur et/ou de l'organisation à l'origine d'un logiciel. Cela permet également de s'assurer que le logiciel n'a pas été modifié après l'émission du certificat, et donc d'assurer l'intégrité du logiciel. (Utilisation des algorithmes de hachage).

Les systèmes d'exploitation implémentent eux, la gestion de ces certificats afin de prévenir l'utilisateur dans le cas où celui-ci essayera d'exécuter un programme non signé. Cela comporte en effet un risque majeur de sécurité, puisque ce programme non signé ne donne aucune information tangible quant à son auteur ou son intégrité.

Implémentation des certificats dans Windows

Windows intègre différents outils permettant de manipuler ces certificats que nous avons abordés dans les dernières parties de ce document.

La première de ces implémentations est le magasin de certificats.

Les magasins de certificats

Le magasin de certificats sous Windows est un composant du système d'exploitation qui permet de stocker et de gérer des certificats numériques. Les certificats numériques sont des fichiers numériques qui contiennent des informations d'identification, telles que l'identité d'une personne ou d'une organisation, ainsi que des clés publiques et privées utilisées pour la sécurisation des communications en ligne.

Le magasin de certificats est une base de données qui stocke les certificats numériques des utilisateurs, des ordinateurs et des services. Il existe plusieurs types de magasins de certificats sous Windows, chacun destiné à stocker des certificats pour un usage spécifique.

Il existe trois magasins de certificats sous Windows :

- Le magasin de certificats utilisateur : Il s'agit du magasin de certificats pour les utilisateurs individuels. Les certificats sont stockés dans ce magasin pour une utilisation sur l'ordinateur local.
- Le magasin de certificats de l'ordinateur : Il s'agit du magasin de certificats pour l'ordinateur en lui-même. Les certificats sont stockés dans ce magasin pour une utilisation par tous les utilisateurs de l'ordinateur.
- Le magasin de certificats de service : Il s'agit du magasin de certificats pour les services système tels que IIS (Internet Information Services). Les certificats sont stockés dans ce magasin pour une utilisation par les services système.

Le magasin de certificats sous Windows peut être géré à l'aide de l'outil de gestion des certificats de Windows, qui permet aux utilisateurs et aux administrateurs système de visualiser, d'importer, d'exporter et de gérer les certificats numériques stockés dans le magasin.

La MMC (Microsoft Management Console)

Il est possible d'accéder aux différents magasins de certificats par le biais de la MMC (Microsoft Management Console).

La MMC est un outil de gestion de système inclus dans les systèmes d'exploitation Windows. Il permet aux administrateurs système de gérer différents aspects du système d'exploitation, tels que les services, les paramètres de sécurité, les disques, les périphériques, les utilisateurs et les groupes.

Il s'agit d'une interface graphique qui permet aux administrateurs système de créer des consoles personnalisées en ajoutant différents modules appelés « snap-ins ». Ces snaps-ins permettent d'ajouter des fonctionnalités spécifiques à la console, telles que la gestion des comptes utilisateurs, la gestion des stratégies de groupe, la gestion des services, etc.

Dans notre cas, il est possible de voir les magasins de certificats en ajoutant un module. Pour cela il suffit d'aller dans la MMC, en tapant « mmc » dans la barre de recherche.

Ensuite, cliquez sur « Fichier » dans la barre d'action en haut à gauche de la fenêtre. Puis cliquez sur « Ajouter/Supprimer un module », tel que le montre la figure suivante :

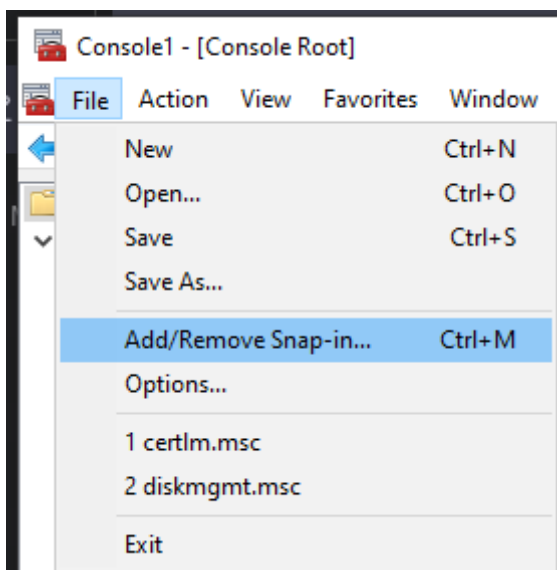


Figure 54 : Ajouter un module dans la MMC.

Dans la fenêtre de gestion des modules, il suffit ensuite de sélectionner le module « Certificats », puis de cliquer sur « Ajouter ». Il vous sera ensuite demandé quel type de magasin de certificat vous souhaitez ajouter parmi les différents types de magasin vu dans la précédente partie.

Il suffit de réitérer cette étape pour ajouter les types souhaités de magasins de certificats.

Une fois terminé, on peut valider notre choix en cliquant sur le bouton « OK ».

La figure suivante montre le magasin de certificat nouvellement installé dans la MMC :

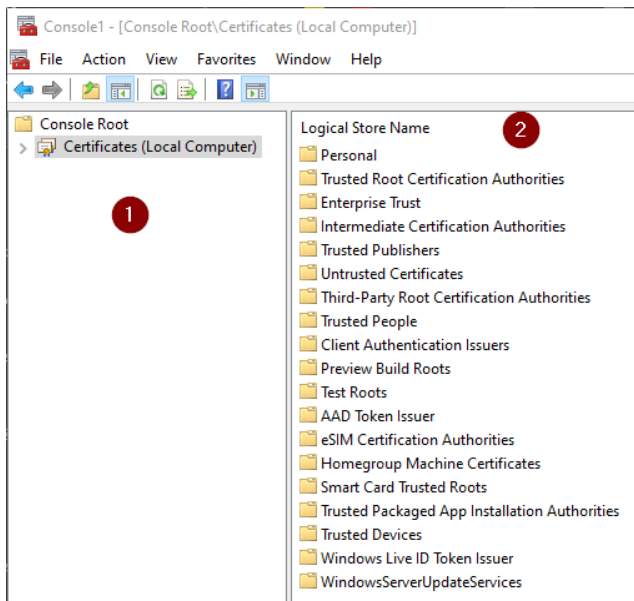


Figure 55 : Magasin de certificats ajouté dans la MMC.

La partie de la fenêtre notée 1 sur la figure montre les modules installés dans la MMC. La seconde partie de la fenêtre notée 2 montre le contenu du module, en l'occurrence, il s'agit du magasin de certificats local de l'ordinateur.

On peut ensuite voir tous les certificats installés sur la machine en cliquant sur les différentes sections présentes dans la partie 2 de la figure.

L'utilitaire sigcheck

L'utilitaire « Sigcheck », ou « Signature check », est un utilitaire en ligne de commandes présent dans la **Sysinternals Suite**, dont nous avons déjà parlé dans [cette partie](#) du document.

Cet utilitaire permet de vérifier la présence d'une signature de code, mais également de connaître certaines informations en rapport avec l'exécutable. Pour utiliser cet outil, il suffit d'ouvrir un terminal à l'emplacement où se trouve l'exécutable « signtool ».

Il suffit ensuite d'appeler ce programme, en spécifiant en argument l'emplacement de l'exécutable que l'on souhaite vérifier. La commande ci-dessous permet de vérifier la signature de l'exécutable « notepad.exe » :

```
./sigcheck C:\Windows\System32\notepad.exe
```

La figure suivante montre le résultat de la commande :

```
c:\windows\system32\notepad.exe:
  Verified:      Signed
  Signing date:  02:36 21/07/2022
  Publisher:     Microsoft Windows
  Company:       Microsoft Corporation
  Description:   Notepad
  Product:       Microsoft« Windows« Operating System
  Prod version:  10.0.19041.1865
  File version:  10.0.19041.1865 (WinBuild.160101.0800)
  MachineType:  64-bit
```

Figure 56 : Résultat de l'appel au programme « sigcheck » sur l'exécutable « notepad.exe »

On peut constater que le programme vérifié est bien signé. On peut également voir certaines informations supplémentaires telles que l'organisation à l'origine de l'exécutable ou la date de signature.

Cet outil est particulièrement pratique pour les administrateurs système puisqu'il s'utilise intégralement en ligne de commandes, et donc permet d'automatiser le processus de vérification.

Signature dans Process Explorer

L'utilitaire Process Explorer, dont nous avons déjà parlé dans cette partie du document, permet aussi de vérifier si un fichier exécutable est signé.

Process Explorer permet de vérifier tous les exécutables à l'origine des processus en cours d'exécution sur la machine. Pour cela commencez par faire apparaître la colonne « Verified Signer », ou signataire vérifié, pour cela, faites un clic droit sur les colonnes présentes, puis sélectionner « select columns ».

Dans la section « Process image », cochez la case « Verified signer », puis validez en cliquant sur « OK ».

Enfin, tel que le montre la figure suivante, cliquez sur « Options » en haut de la fenêtre, puis sélectionnez « Verify image signatures » :

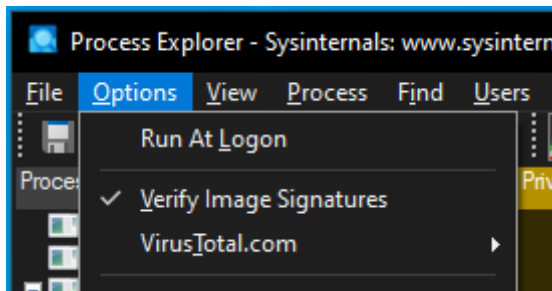


Figure 57 : Activation de la vérification des signatures dans Process Explorer.

Après quelques instants, dans la colonne « Verified signer » devrait apparaître le nom du signataire pour chaque exécutable, si ceux-ci possèdent une signature valide, tel que le montre la figure suivante :

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name	Verified Signer
Registry		8 944 K	82 808 K	100			
System Idle Process	71.24	60 K	8 K	0			
System	0.73	208 K	6 800 K	4			
Interrupts	0.37	0 K	0 K	n/a	Hardware Interrupts and DPCs		
smss.exe		1 068 K	1 000 K	416			
Memory Compression	< 0.01	1 464 K	478 904 K	2276			
csrss.exe	< 0.01	2 196 K	5 628 K	520			
wininit.exe		1 432 K	6 400 K	620			
services.exe	< 0.01	5 732 K	9 700 K	700			
svchost.exe	0.37	10 752 K	28 136 K	904	Host Process for Windows S...	Microsoft Corporation	(Verified) Microsoft Windows Publisher
dllhost.exe		3 164 K	10 784 K	4992	COM Surrogate	Microsoft Corporation	(Verified) Microsoft Windows
StartMenuExperience...	< 0.01	23 384 K	69 800 K	6100			(Verified) Microsoft Windows
RuntimeBroker.exe		5 792 K	23 360 K	4844	Runtime Broker	Microsoft Corporation	(Verified) Microsoft Windows
SearchApp.exe	Susp...	203 168 K	290 096 K	6304	Search application	Microsoft Corporation	(Verified) Microsoft Windows

Figure 58 : Affichage des signataires vérifiés dans Process Explorer.

La mention « (Verified) » indique bel et bien la présence d'une signature valide et vérifiée.

L'utilitaire signtool

Signtool est un utilitaire de ligne de commande inclus dans les outils de développement de Microsoft, qui permet de signer numériquement des fichiers exécutables, des bibliothèques de liens dynamiques (DLL), des contrôles ActiveX, des pilotes de périphériques et d'autres types de fichiers.

Signtool peut être utilisé avec un certificat numérique émis par une autorité de certification tiers ou avec un certificat autosigné. Cependant, il est recommandé d'utiliser un certificat émis par une autorité de certification de confiance pour garantir l'authenticité du fichier signé.

Pour utiliser Signtool, il faut disposer d'un certificat numérique valide et d'une clé privée associée. On peut ensuite utiliser l'utilitaire Signtool pour signer numériquement un fichier, en spécifiant le chemin d'accès au fichier à signer, le certificat numérique à utiliser et d'autres options de signature telles que la date d'expiration du certificat.

Vérification de signature par le robot de scan

Type de fichier

Nous connaissons dès à présent le fonctionnement et l'intérêt des signatures et certificats de code. Il serait donc plutôt intéressant d'améliorer le robot de scan permettant de vérifier la présence d'une signature valide, et si non, d'en informer l'utilisateur.

Cette modification est plutôt simple, dans un premier temps, on viendra vérifier si un fichier en cours de scan est un exécutable valide ou non. Pour cela, on viendra simplement vérifier si le nom du fichier se termine par :

- .exe
- .bat
- .cmd
- .com

Soit les quatre principales extensions d'exécutables et scripts sous Windows.⁵¹

Vérification du certificat

Si le fichier est un exécutable, alors on peut commencer à vérifier la présence d'un certificat valide. Pour ce faire, nous utiliserons la bibliothèque « pefile⁵² », que nous installerons avec la commande suivante :

```
pip install pefile
```

L'opération qui suit vise dans un premier temps à extraire les informations de l'entête PE du fichier ouvert via la bibliothèque « pefile ». Les informations en rapport avec la signature du code se situent dans la table de sécurité, présente dans l'entête du fichier. Soit la partie notée « En-tête PE » sur la figure présente dans [cette partie du document](#).

Pour savoir si un certificat est présent, il suffit ensuite de tenter de charger les données dans un objet « ContentInfo » de la classe « Cryptography⁵³ ». Si la lecture échoue, une exception Python est levée. Il suffit alors de rattraper cette erreur et donc par déduction, avertir l'utilisateur qu'aucun certificat n'est présent.

Avertir l'utilisateur

Afin d'avertir l'utilisateur qu'aucun certificat n'est présent, on viendra spécifier dans les [logs textuels](#) et [HTML](#) les résultats de la vérification. Si un certificat est trouvé, aucun message n'est ajouté.

⁵¹ https://fr.wikipedia.org/wiki/Fichier_ex%C3%A9cutable

⁵² <https://pypi.org/project/pefile/>

⁵³ <https://pypi.org/project/cryptography/>

Bibliographie

Vous pourrez retrouver la bibliographie consultée pour l'élaboration de ce Handbook dans le répertoire « biblio » du build, ou directement en ligne, via les URLs suivantes :

Processus & Threads

<https://sites.ualberta.ca/~smartynk/Resources/CMPUT%20379/beck%20notes/process.pdf>

https://sritsense.weebly.com/uploads/5/7/2/7/57272303/unit_i-chapter_2.pdf

Programmation-objet avec PowerShell

https://denispallez.i3s.unice.fr/lib/exe/fetch.php?media=doc:power_tp1.pdf

Utilisation de la suite Sysinternals dans la cybersécurité

<https://download.sysinternals.com/files/SysinternalsMalwareCleaning.pdf>

Méthodes de détection virale

<https://www.labri.fr/perso/ly/publications/virus.pdf>

Utilisation de Python et des APIs Web

<http://web.univ-ubs.fr/lmba/lardjane/python/c4.pdf>

Table des illustrations

Figure 1 Anatomie du processeur	6
Figure 2 Attente et exécution des threads par le processeur	6
Figure 3 Définition de la priorité et de l'affinité d'un processus	7
Figure 4 Interactions entre le processeur et les mémoires de traitement	9
Figure 5 : Structure du registre sous Windows 11	11
Figure 6 : Fenêtre principale de Process Explorer	12
Figure 7 : Processus enfants d'Explorer.exe	12
Figure 8 : Exemple d'informations des processus sous Process Explorer	13
Figure 9 : Étendue des informations disponibles sous Process Explorer	13
Figure 10 : Menu contextuel Virus total	14
Figure 11 : Exemple de résultat Virus total	14
Figure 12 : Fenêtre principale de Process Monitor	16
Figure 13 : Fonctionnalités principales de Process Monitor	17
Figure 14 : Exemple de résultat de capture	18
Figure 15 : Procédure d'accès aux paramètres de Firefox.	20
Figure 16 : Paramètres de gestion de la page d'accueil de Firefox	21
Figure 17 : Partie des résultats de la capture de Firefox.	22
Figure 18 : Filtres du piège de capture Firefox	22
Figure 19 : Aperçu du contenu du fichier « prefs.js ».	23
Figure 20 : Évènement de modification du fichier « prefs.js » par « sublime_text.exe ».	23
Figure 21 : Filtres de capture du DNS Empoisonné	26
Figure 22 : Fenêtre des connexions réseau	26
Figure 23 : Menu contextuel de la connexion	26
Figure 24 : Étapes de configuration manuelle du serveur DNS préféré.	27
Figure 25 : Exemples d'évènements capturés sous Process Monitor	28
Figure 26 : Outil de recherche de Process Monitor	28
Figure 27 : Page d'accueil de Virustotal	31
Figure 28 : Contenu du fichier de test comportant l'empreinte EICAR	32
Figure 29 : Résultats d'analyses du fichier de test de détection EICAR.	32
Figure 30 : Résultats d'analyse du site youtube.com sur Virustotal	33
Figure 31 : Exemple de script sous Linux (Shell)	34
Figure 32 : Interpréteur Python sous Windows	36
Figure 33 : Exemple de log post-scan	37
Figure 34 : Exemple de résultat positif, logs format HTML	38
Figure 35 : Fichier de configuration YAML du robot de scan Virustotal	39
Figure 36 : Procédure d'accès à la clé d'API publique.	40
Figure 37 : Récupérer la clé d'API publique.	40
Figure 38 : Mise en place de la clé API dans le fichier de configuration.	40
Figure 39 : Limitation de l'API du compte gratuit	41
Figure 40 : Panneau de contrôle des API de communication Vonage	42
Figure 41 : Scan du fichier de test EICAR par Virustotal par l'API	43
Figure 42 : Contenu du fichier de logs textuels au terme du scan des fichiers du jeu d'essais.	43
Figure 43 : Logs HTML générés à la suite de l'analyse des fichiers du jeu d'essais.	44
Figure 44 : Résultats du scan du fichier de test EICAR	44
Figure 45 : Résultats précis du scan du fichier de test EICAR	44
Figure 46 : Aperçu du fichier malveillant dans la zone de quarantaine	45
Figure 47 : Segmentation d'un fichier au format PE (Portable Exécutable)	46
Figure 48 : Aperçu du fichier « notepad.exe » ouvert dans le logiciel « PEView ».	48
Figures 49 : aperçu de code assembleur contenu dans le fichier « notepad.exe ».	49
Figure 50 : Fonctionnement du chiffrement symétrique	51
Figure 51 : Fonctionnement du chiffrement asymétrique	51
Figure 52 : Cas concret d'utilisation des algorithmes de hachage.	53
Figure 53 : Schéma de l'implémentation d'un certificat de signature de code. ...	55

Figure 54 : Ajouter un module dans la MMC.....	57
Figure 55 : Magasin de certificats ajouté dans la MMC.....	58
Figure 56 : Résultat de l'appel au programme « sigcheck » sur l'exécutable « notepad.exe ».....	58
Figure 57 : Activation de la vérification des signatures dans Process Explorer.	59
Figure 58 : Affichage des signataires vérifiés dans Process Explorer.....	59