ESD
Cybersecurity Academy

Analyse de logiciels malveillants

Concepts fondamentaux

ESD
Cybersecurity Academy

6. Retro Ingénierie

- 6.1. Paradigmes : Compilation, semi-compilation, interprétation
- 6.2. Le format PE
- 6.3. Windows :
  - 6.3.1. Concepts
  - 6.3.2. Au coeur des processus
  - 6.3.3. Injection de code
- 6.4. Langages semi-compilés
  - 6.4.1. cas Jigsaw
- 6.5. Langages interpetés
  - 6.5.1. cas IceID / Dridex

7. Rootkit / Bootkit

8. 5 études de cas

9. TP Final

1. Concepts fondamentaux

Définitions

Un logiciel malveillant ou maliciel est un programme informatique développé dans le but de nuire à un système informatique et / ou à son utilisateur.

Cette grande famille de logiciels est composée de plusieurs catégories.

Sources : Norse / www.extremetech.com

1. Concepts fondamentaux

Définitions

Chevaux de troie / Trojans

D'apparence sain, il s'agit d'un programme auquel une charge utile ou **payload** a été ajoutée. Il peut être le vecteur d'autres catégories de maliciel.

A l'insu de l'utilisateur, il dérobe des données (messages, contact, IMEI...).

Zitmo est une famille de maliciel spécialisée dans le vol de données bancaires mTANs (Mobile Transaction Authentication Number).

Sources : Norse / www.extremetech.com

1. Concepts fondamentaux

Définitions

Les portes dérobées / Backdoor

Leur fonction est de **fournir une porte d'accès** à un acteur à l'insu de l'utilisateur final.
En novembre 2016, les chercheurs en sécurité de kryptowire ont découvert Adups, une backdoor permettant d'accéder aux journaux d'appels, sms et historique de plus de 700 millions d'appareils.

Il permet également l'**exécution de commandes** en mode super utilisateur à savoir "root". La société en charge du développement des firmware AdUpstechnology a déployé ce maliciel sur plusieurs terminaux de marques chinoises telles que ZTE et Huawei.

1. Concepts fondamentaux

Définitions

Les vers / Worm

Ils ont la capacité de se **reproduire** et se **propager sans hôte**.

Les premières générations de malware étaient fortement typés (Net-Worm.Win32.Koobface n'avait comme fonction que celles d'un vers) alors que nous faisons face désormais à des malwares multifonctions (Wana Decrypt0r est le premier ransomware avec des capacités de vers informatiques, exploitation de la vulnérabilité MS17-010 - exploit sur le protocol SMB).

Android.Obad.OS est un exemple de vers qui utilise le bluetooth comme moyen de propagation

1. Concepts fondamentaux

Définitions

Les espiongiciels / Spyware

A l'instar des Trojans, ils peuvent s'**apparenter à des applications** saines mais ont pour fonction de **récolter les informations** de l'utilisateur.

1. Concepts fondamentaux

Définitions

Les réseaux de robots / Botnet

Transforme le périphérique en un élément d'un réseau contrôlé par l'attaquant.
L'attaquant possède ainsi une flotte de plusieurs milliers de terminaux appelé botnet qu'il peut utiliser à sa guise.

Mirai est un exemple d'utilisation de ce type de réseau. A permis le contrôle de milliers d'IOT(~300k), principalement des webcams, afin de réaliser des attaques de type DDOS à l'encontre de serveur DNS.

Description du malware **Emotet** : https://www.cert.ssi.gouv.fr/alerte/CERTFR-2020-ALE-019/

1. Concepts fondamentaux

Définitions

Les rancongiciels / Ransomware

De plus en plus répandus, ils procèdent au chiffrement des données de l'utilisateur et demandent une rançon pour permettre le déchiffrement.

Ruyk, samsam ou bien **Wannacry** montrent à quel point ce type de menace est à prendre au sérieux.

Les ransomwares font partie des menaces les plus rentables.
L'année 2021 à vu de nombreuses sociétés touchées et des rançons records

Liens :
https://www.01net.com/actualites/acer-le-geant-du-pc-est-victime-d-un-ransomware-qui-reclame-50-millions-de-dollars-2038979.html
https://www.globalsecuritymag.fr/Les-attaques-de-ransomware-sur-des,20210324,109655.html

1. Concepts fondamentaux

Définitions

Les logiciels à risques / Riskware

Terme générique utilisé par Kaspersky Lab afin de décrire un programme écrit sans réel but de nuire mais possédant des fonctions dites "critiques" pouvant être exploitées.

Le logiciel SmitFraudFix utilisé en décontamination d'ordinateur est reconnu par une dizaine d'antivirus comme potentiellement dangereux (RiskTool.Win32.Reboot.f) car il possède les droits de tuer des processus.

1. Concepts fondamentaux

Classification

Classification ?

De nombreuses méthodes de classification s'appuient sur plusieurs critères:

- Système cible (Windows, Linux ..)
- Méthode de propagation (Vers, virus...)
- Méthode de protection (Packed, Rootkit)
- Fonctions "malignes" (Backdoor, ...)

1. Concepts fondamentaux

Classification

Classification : F-secure

Figure C.i : Classification des menaces (F-secure)

F-secure propose une classification sur 3 thèmes (catégorie, type et plateforme).

Les classifications sont intéressantes car elle elles sont le reflet de l'état de l'art en matière de classification et nature des menaces.
Elles évoluent avec le temps pour s'adapter aux nouvelles menaces.

https://www.kaspersky.fr/resource-center/threats/malware-classifications

1. Concepts fondamentaux

Les différents types d'analyse

- **Analyse statique de base** : Examiner l'exécutable sans que celui-ci ne soit exécuté. Il s'agit d'obtenir des éléments permettant de catégoriser le binaire tel que le format, version, packé ou non. Ces premiers éléments contribueront à catégoriser le binaire en malware ou pas. Rapide mais inefficace contre des techniques d'évasions avancés (obfuscation, chiffrement...)
- **Analyse dynamique de base** : Exécution et observation du comportement du malware dans le but de créer des signatures et de comprendre son comportement. Exécution dans un environnement virtualisé
- **Analyse statique avancée** : Analyse du comportement et fonctionnement du malware par rétro-ingénierie
- **Analyse dynamique avancée** : Analyse approfondie du binaire durant son exécution . Cela peut se faire au travers d'un sandbox ou debugger

Ces catégories d'analyses sont les plus répandues dans la littérature. L'analyse hybride comprenant les analyses statique et dynamique au travers de framework devient incontournable (hybrid-analysis, virustotal).

Ce ne sont que des grandes catégories, par exemple les analyses symbolique, concolique ou teintée sont intéressantes et ne doivent pas être omises.
Des solutions comme Triton ou bien Angr trouveront leur place dans votre boîte à outils.

https://triton.quarkslab.com/
https://github.com/angr/angr

1. Concepts fondamentaux

Les différents types d'analyse

- **Analyse hybride** : Combinaison de l'analyse statique et dynamique
- **Analyse concolique** : Il s'agit d'un mélange entre l'analyse concrète et symbolique.
  L'idée étant de pouvoir extraire le comportement du binaire (dynamique) mais en prenant en compte le plus de chemins d'exécution possibles et non plus qu'un seul.

1. Concepts fondamentaux

Mécanismes d'évasion

Chiffrement

Procédé de cryptographie grâce auquel on souhaite rendre la compréhension d'un message/document impossible à toute personne qui n'a pas la clé de (dé)chiffrement (AES, DES, RSA, etc.).

Offuscation

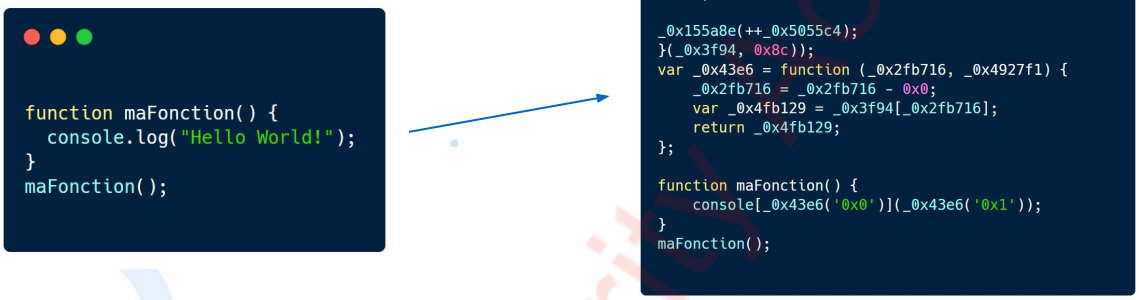
Obscurcir le sens qui peut être tiré d'un message (Polymorphisme, Métamorphisme, Mimimorphisme).

L'offuscation est un mécanisme quasiment auto inclus lors du développement d'un programme de cette nature. Il s'agit de rendre plus difficile l'analyse statique.

https://www.researchgate.net/publication/220269513_Mimimorphism_A_New_Approach_to_Binary_Code

1. Concepts fondamentaux

Exemple d'offuscation de code



```
function maFonction() {
  console.log("Hello World!");
}
maFonction();
```

```
var _0x3f94 = [
    'log',
    'Hello\x20World!'
];
(function (_0x5bb848, _0x5055c4) {
    var _0x155a8e = function (_0x10a8ba) {
        while (--_0x10a8ba) {
            _0x5bb848['push'](_0x5bb848['shift']());
        }
    };

_0x155a8e(++_0x5055c4);
}(_0x3f94, 0x8c));
var _0x43e6 = function (_0x2fb716, _0x4927f1) {
    _0x2fb716 = _0x2fb716 - 0x0;
    var _0x4fb129 = _0x3f94[_0x2fb716];
    return _0x4fb129;
};

function maFonction() {
    console[_0x43e6('0x0')](_0x43e6('0x1'));
}
maFonction();
```

Vous pouvez obtenir le résultat ci-dessous via le site suivant :

code :

```
function maFonction() {
  console.log("Hello World!");
}
maFonction();
```

Lien : https://obfuscator.io/

1. Concepts fondamentaux

Exemple d'offuscation de code

Ces mécanismes rendent plus compliqué l'analyse statique des binaires.

Des outils peuvent aider dans cette tâche :

javascript deobfuscator and unpacker : https://lelinhtinh.github.io/de4js/

Floss : https://github.com/fireeye/flare-floss.
Permet de rendre compréhensible les chaînes de caractères qui pourraient contenir des éléments tels que les noms de domaines, adresses ip relatives à une infection.

Le binaire de FLoss est présent ici (https://github.com/fireeye/flare-floss/releases/tag/v1.7.0)
Cette liste est bien entendu non exhaustive et d'autres programme nous serons présenté par la suite.

1. Concepts fondamentaux

Mise en place d'un laboratoire d'analyse

La mise en place d'un environnement virtualisé présente de nombreux intérêts :

- Exécuter plusieurs machines virtuelles en même temps et donc simuler une infrastructure.
- Automatiser les analyses
- Accélérer le processus d'analyse (snapshot, ...)
- Les VM devront être préparées (patch, mise à jour système, outils, adaptées pour optimiser le bon déroulement du programme malveillant)
- L'hôte devra également être préparé :
    - mise à jour
    - Gnu/linux Hardened
    - Windows 10 préparé (isolation réseau, test des outils de détection de présence d'un environnement virtualisé)

Un laboratoire répond à de nombreuses problématiques :

Cas 1 :

Un laboratoire simple peut être composé de clients utilisant le principe de snapshots.
Il s'agit d'enregistrer un état de la machine virtuelle et de revenir sur celui ci après exécution du malware.
Une fois la charge exécutée et analysée, vous remettez donc dans un état antérieur votre machine virtuelle.

VirtualBox et VmWare (Workstation) propose nativement ces fonctionnalités.

L'inconvénient est le manque d'automatisation.

Cette solution n'exclut pas de prendre les bonnes précautions (connexion réseau, partage avec l'hôte ...)

Cas 2:

Une solution telle que cuckoo (python 2) ou bien CAPE (python 3) qui

va automatiser ces actions d'analyses.

Lien :
https://cuckoosandbox.org/
https://github.com/ctxis/CAPE

1. Concepts fondamentaux

Mise en place d'un laboratoire d'analyse

De nombreuses manières de procéder sont possibles :

- Utiliser Windows 10 en hôte avec des distributions Gnu/Linux dédiées + WSL2. Cela n'est pas optimal, notamment sur l'aspect automatisation et rejeux.
- Penser à desactiver les protections locales (Firewall, EPP, EDR)
- Il peut être intéressant d'ajouter un client de type Sysmon pour l'aspect détection des malwares.
- Penser à activer les snapshots.

1. Concepts fondamentaux

Mise en place d'un laboratoire d'analyse

Des outils permettent d'automatiser le déploiement de VM(s) et d'infrastructure.

- Vagrant et Terraform

HashiCorp Terraform

Ces deux solutions vont automatiser la montée d'une infrastructure.
Il s'agit aussi bien d'un poste que d'une dizaine.
Le travail de personnalisation des VM via packer par exemple et l'automatisation (**chocolatey**, **ansible** ...) est un travail non négligeable.

1. Concepts fondamentaux

Mise en place d'un laboratoire d'analyse

Trois projets présentent une vision intéressante :

https://github.com/GoSecure/malboxes

https://github.com/cuckoosandbox/cuckoo

https://github.com/fireeye/capa

DRAKVUF

https://drakvuf.com/
https://github.com/CERT-Polska/drakvuf-sandbox

Capa est un outil d'analyses de PE et non une solution de virtualisation comme les deux autres logiciels.
Il est conseillé ici car de part ses fonctions, il présente rapidement des axes de recherches.

Malboxes est un projet basé sur Vagrant.

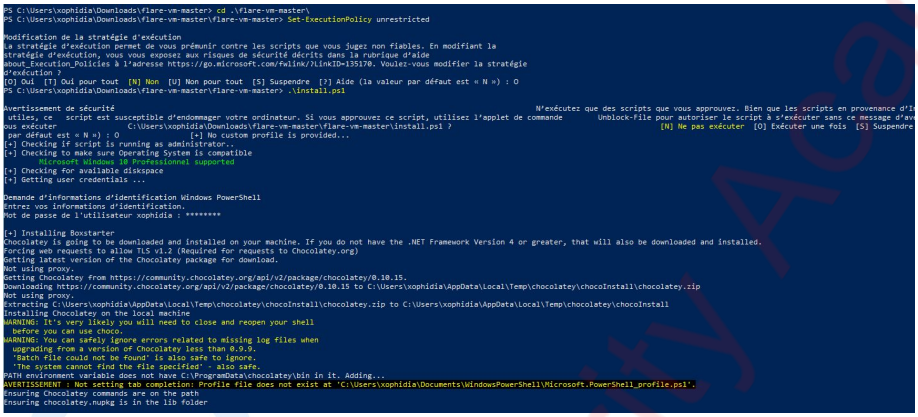
Drakvuf est encore jeune mais possède des avantages certains (facilité de mise en service)

1. Concepts fondamentaux

Mise en place d'un laboratoire d'analyse

Flare VM

https://www.fireeye.com/blog/threat-research/2017/07/flare-vm-the-windows-malware.html



FlareVM est une image basée sur Windows à destination des équipes d'analyse malware, retro-ingénierie, pentest et réponse à incident.
A partir d'une installation >= Win7, exécutez le script suivant : http://boxstarter.org/package/url?https://raw.githubusercontent.com/fireeye/flare-vm/master/flarevm_malware.ps1

Update : https://www.fireeye.com/blog/threat-research/2018/11/flare-vm-update.html
Note : L'installation sous Windows 7 SP1 pose des problèmes. Nous conseillons donc une installation sur un Windows 10

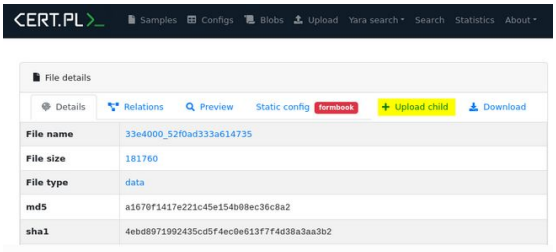
1. Concepts fondamentaux

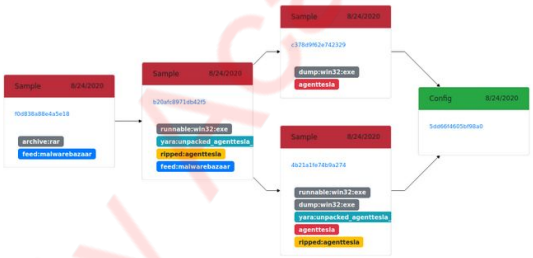
Mise en place d'un laboratoire d'analyse

Malwarecage

Outil permettant la gestion des malwares.

Installation : https://mwdb.readthedocs.io/en/latest/setup-and-configuration.html#id1





Malware repository component for automated malware collection/analysis systems.
Formerly known as Malwarecage.

Lien : https://github.com/CERT-Polska/mwdb-core

1. Concepts fondamentaux

Mise en place d'un laboratoire d'analyse

Malwarecage

Nous conseillons également l'ajout de Karton et son intégration dans malwarecage.
Distributed malware processing framework based on Python, Redis and MinIO.

Server version	2.3.0
User registration	enabled
Rate limits	disabled
Karton integration	enabled
Maintenance mode	disabled
Plugins	enabled

Karon propose de nombreux plugins permettant notamment de tester des règles yara.

Lien : https://github.com/CERT-Polska/karton

L'aspect communautaire est une grosse plus-value à cet outil. Vous pouvez en effet connecter votre installation à celle d'autres acteurs (interne / externe).

1. Concepts fondamentaux

Mécanismes de détection

La machine virtuelle est l'outil idéal pour réaliser des **analyses statiques et dynamiques**. Les hyperviseurs actuels possèdent des marqueurs **caractéristiques** qui peuvent être utilisés par le malware pour s'adapter.

1. Concepts fondamentaux

Mécanismes de détection

Le malware possède de nombreux mécanismes permettant de **détecter la présence d'un environnement virtualisé**.
Cela permet d'éviter, pour le malware, de tomber dans un "bac à sables" et/ou d'être ensuite analysé par les analystes.
Il peut également posséder des **mécanismes d'évasion de la machine virtuelle** afin d'atteindre potentiellement un environnement d'administration/production.

1. Concepts fondamentaux

Mécanismes de détection

La présence d'un hyperviseur de niveau 2 est révélée par la présence de nombreux artefacts.

L'outil **scoopyNG** (http://www.trapkit.de/tools/scoopyng/) propose un rapport indiquant si l'exécution est réalisée sur un OS natif ou non.

L'outil **VMDetector** (https://github.com/robsonfelix/VMDetector) est également intéressant pour les éléments qu'il détecte.

Les hyperviseurs sont classés actuellement en deux catégories :

Type 1 : natif
Un hyperviseur de Type 1, natif, voire « bare metal » est un logiciel qui s'exécute directement sur une plateforme matérielle ;

Type 2 : hosted
Un hyperviseur de Type 2 est un logiciel qui s'exécute à l'intérieur d'un autre système d'exploitation/

1. Concepts fondamentaux

Mécanismes de détection

Détection basée sur la présence de devices

```
public override bool ContainsDevice(IEnumerable<PnPEntity> devices)
{
    return devices.Any(d => d.Name.Equals("vmware pointing device"))
        || devices.Any(d => d.Name.Contains("vmware sata"))
        || devices.Any(d => d.Name.Equals("vmware usb pointing device"))
        || devices.Any(d => d.Name.Equals("vmware vmci bus device"))
        || devices.Any(d => d.Name.Equals("vmware virtual s scsi disk device"))
        || devices.Any(d => d.Name.StartsWith("vmware svga"))
        ;
}
```

Détection basée sur la présence de services

```
public override bool ContainsService(IEnumerable<WindowsService> services)
{
    return services.Any(s => s.CommandLine.Contains("vmware") && s.Name.Equals("vmtools"))
        || services.Any(s => s.CommandLine.Contains("vmware") && s.Name.Equals("tpvcgateway"))
        || services.Any(s => s.CommandLine.Contains("vmware") && s.Name.Equals("tpautoconnsvc"))
        ;
}
```

Exemple des éléments recherchés par l'outil VMDetector

1. Concepts fondamentaux

Mécanismes de détection

```
PS C:\Windows\system32> get-service -Displayname "*vmware*"

Status   Name               DisplayName
------   ----               -----------
Running  VMAuthdService     VMware Authorization Service
Running  VMnetDHCP          VMware DHCP Service
Running  VMUSBArbService    VMware USB Arbitration Service
Running  VMware NAT Service VMware NAT Service


PS C:\Windows\system32> get-service -Displayname "*virtualbox*"
PS C:\Windows\system32> get-service -Displayname "*vboxservice*"
```

Recherche de la présence de l'hyperviseur VMware et VirtualBox en recherchant les services associés.

```
static VirtualMachineDetector()
{
    _detectors = new IVirtualEnvironment[]
    {
            new VmWarePlayer(),
            new HyperVMachine(),
            new QEmuMachine(),
            new VirtualBoxMachine(),
    };

    _computer = Create<ComputerSystem>("Win32_ComputerSystem");
    _bios = Create<BIOS>("Win32_BIOS");
    _motherboard = Create<MotherboardDevice>("Win32_MotherboardDevice");
    _devices = CreateList<PnPEntity>("Win32_PnPEntity");
    _disks = CreateList<DiskDrive>("Win32_DiskDrive");
    _services = GetWindowsServices();
```

Identification de l'hyperviseur

Recherche d'artefacts

```
PS C:\Windows\system32> Get-WmiObject -class win32_DiskDrive


Partitions : 4
DeviceID   : \\.\PHYSICALDRIVE0
Model      : SK hynix SC300 SATA 512GB
Size       : 512105932800
Caption    : SK hynix SC300 SATA 512GB

Partitions : 1
DeviceID   : \\.\PHYSICALDRIVE1
Model      : HGST HTS721010A9E630
Size       : 1000202273280
Caption    : HGST HTS721010A9E630
```

1. Concepts fondamentaux

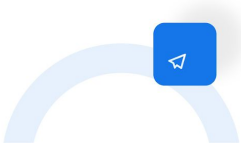
Mécanismes de détection

De nombreux environnements sont disponibles sur internet :

- https://malwr.com
- https://www.hybrid-analysis.com
- https://virustotal.com
- https://any.run
- https://www.joesandbox.com/#windows

Ils présentent un réel intérêt pour comprendre rapidement la nature d'un binaire mais il est souvent intéressant d'avoir sa propre installation pour aller plus loin.

Note : Il est intéressant de tester dans un premier temps avec le hash du binaire à analyser. Il est probable qu'un test a déjà été effectué.



1. Concepts fondamentaux

Mécanismes anti-debug

```
PS C:\Windows\system32> wmic bios get serialnumber
SerialNumber
7305RF2

PS C:\Windows\system32>
```

Exemple de différence entre un environnement émulé et physique. Le numéro de série du BIOS est caractéristique dans le cas d'un hyperviseur.

```
PS C:\Users\xophidia> wmic bios get serialnumber
SerialNumber
VMware-56 4d a1 ef cf b3 fd ef-4e 77 71 a9 78 5f c9 c3
```

1. Concepts fondamentaux

Mécanismes anti-debug

- Le malware peut posséder des mécanismes le protégeant contre les tentatives de débogage.
- L'idée étant de détecter une tentative et d'adapter son comportement en conséquence.

1. Concepts fondamentaux

Mécanismes anti-debug

```
// 5.1
// Reference
// http://www.offensivecomputing.net/ Written by Danny Quist, Offensive Computing
void smsw() {
    unsigned int reax = 0;

    __asm
    {
        mov eax, 0xCCCCCCCC;
        smsw eax;
        mov DWORD PTR [reax], eax;
    }

        if ( (( (reax >> 24) & 0xFF ) == 0xcc) && (( (reax >> 16) & 0xFF ) == 0xcc))
        printf("VM detected\n");
        else
                printf("VM not detected\n");
}
```

La présence d'**opcodes** caractéristiques est un élément discriminant (sgtd, sldt, smsw...)

Il est possible de bypass cela via un patch – modification de la valeur d'un registre.

Un instruction en langage machine est composée de deux éléments :

- l'opcode ou code opération qui contient l'action à réaliser ;
- l'opérande qui précise sur quoi ces opérations vont être réalisées.

1. Concepts fondamentaux

Mécanismes anti-debug

L'API Win32 contient une méthode **IsDebuggerPresent()** qui détecte la présence d'un débugger attaché. Utilisable en mode utilisateur et noyau.

Il est possible de patcher manuellement ou bien d'utiliser des plugins disponibles (Olly Phantom) dans OllyDbg ou autres.

Liens :

- https://orangecyberdefense.com/fr/insights/blog/reponse_a_incident/bypass-des-anti-debugs-de-neurevt/
- https://en.wikibooks.org/wiki/X86_Disassembly/Debugger_Detectors

2. Analyse statique simple

Il s'agit d'utiliser des outils pour comprendre le fonctionnement du programme. Le principe de ce type d'analyse est de **ne pas exécuter le programme**. L'avantage étant d'avoir une vision complète du fonctionnement de ce programme mais cela reste compliqué dans le cas de codes chiffrés ou offusqués. De plus de nombreux indicateurs ne seront pas accessibles (taux d'utilisation CPU, batterie ...).

L'infrastructure nécessaire est cependant plus **simple** à mettre en œuvre.

2. Analyse statique simple

Mécanisme de persistance

Ces mécanismes permettent au malware d'assurer sa survie notamment lors d'un **redémarrage système**.

Les clés suivantes répondent à ce besoin :

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce

Si le malware possède des droits administrateur :

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

Les mécanismes de persistance peuvent être etudié lors de l'analyse statique ou bien dynamique.
Il s'agit en fait d'un point de vue assez subjectif.
Si le malware a déjà contaminé le poste client, il s'agit donc d'analyser les artefacts inhérents à son exécution.
Il n'est plus utile d'exécuter ce dernier et nous rentrons dans l'analyse statique (via artefacts).
Si le malware n'a pas été exécuté, nous tomberons dans l'analyse dynamique.

Lien :
https://support.microsoft.com/pl-pl/help/179365/info-run,-runonce,-runservices,-runservicesonce-and-startup

2. Analyse statique simple

Mécanisme de persistance

Ces mécanismes peuvent également s'appuyer sur les **services**.

Bien que nécessitant des droits administrateurs, ils ne sont pas à négliger.

```
sc create <service_name> binPath= <service_path>  start= auto
sc.exe create NewService binpath= c:\windows\system32\NewServ.exe start= auto
```

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\Services\Once
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\Services

De nombreuses autres techniques permettent d'assurer une persistance au malware :

- https://www.andreafortuna.org/2017/07/06/malware-persistence-techniques/

2. Analyse statique simple

Mécanisme de persistance

Il est également intéressant d'utiliser les **tâches programmées** (Task Scheduled) afin de s'assurer de l'exécution de la charge.

Une autre approche consiste à se placer à l'intérieur de répertoires lancés au démarrage et de créer des raccourcis pointant sur ces éléments. Ils seront donc lancés automatiquement à la prochaine connexion / redémarrage.

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders

2. Analyse statique simple

Mécanisme de persistance

Une approche intéressante est de profiter de la **séquence de recherche des bibliothèques** attachées à un programme lors de l'exécution de celui-ci.

Cette technique connue sous le nom de **DLL Search Order Hijacking** utilise le fait que Windows, lors du chargement des bibliothèques, procède à la recherche de ces dernières à des endroits précis.

Par exemple, ajouter une bibliothèque à un emplacement qui sera parcouru avant l'emplacement légitime ou bien renommer une bibliothèque par un nom légitime toujours dans à un emplacement prioritaire (Binary planting attacks).

Source : https://attack.mitre.org/techniques/T1574/001/

2. Analyse statique simple

Mécanisme de persistance

Détourner l'utilisation de Shimcache :

- Il s'agit d'un artefact très intéressant. Il assure la compatibilité des applications et indique notamment les exécutables présents mais n'ayant pas encore été exécutés.
- Il est donc possible de réaliser des actions sur certains programmes tels que l'ajout de DLL manquantes et profiter de cette fonctionnalité pour intégrer du code malveillant.
- L'utilisation de **sdbinst.exe** est donc un élément à surveiller.

Ces mécanismes sont nombreux et astucieux.

La liste présentée n'est pas exhaustive aussi il est conseillé d'aller plus loin afin de gagner du temps lors des futures analyses.

Lien :
https://any.run/report/d841d9092239fc029b10da01c19868749b0f6bd757926ff04674658468495808/542befa5-d439-489c-bde8-cab96b90cb7f

2. Analyse statique simple

Analyse statique simple

Empreinte numérique(hash) : permet d'identifier un malware via une base de données commune (Virustotal)

- md5sum malware.exe (Linux/bash)
- Get-FileHash malware.exe | Format-List (Windows/PS)
-

Strings (Linux): permet d'afficher les chaînes ASCII et unicode.

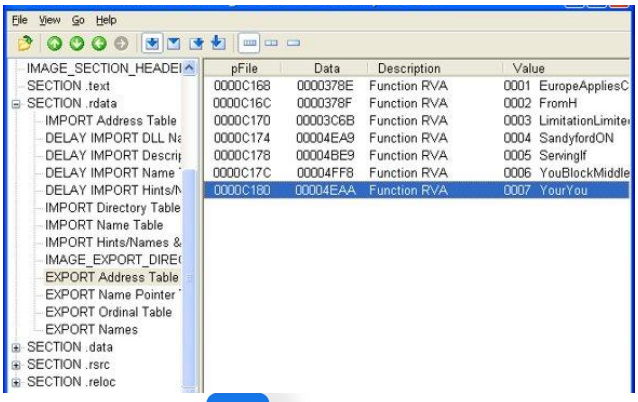
Apporte de précieux renseignements (url, nom de domaine, adresse ip, technologie(Framework, dll,...))

File (Linux): donne des informations sur le type (nature) de fichier

2. Analyse statique simple

Analyse statique simple

Analyse statique simple : PEView



Affiche les nombreuses sections propres au format PE (Portable Executable)

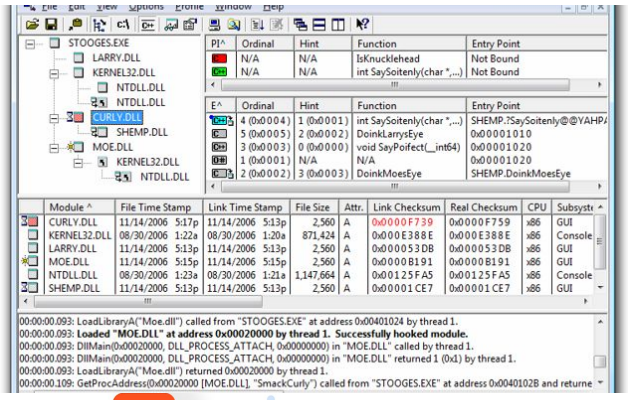
Affiche la structure des binaires au format PE.

Outil : http://wjradburn.com/software/

2. Analyse statique simple

Analyse statique simple

Analyse statique simple : Dependency Walker



Indique de nombreux éléments sur les bibliothèques chargées.

Affiche les dépendances et le fait que celles-ci aient été chargées ou pas.

La grande majorité des binaires qui nous intéresse utiliser la winapi32 et chargent des fichiers de type DLL.
Il est donc intéressant d'avoir une vision des chargements/erreurs lors de l'exécution du programme.

Outil: https://www.dependencywalker.com/

2. Analyse statique simple

Analyse statique simple

En fonction des cas, de nombreux autres outils apportent leurs informations utiles.

- ProcessHacker
- PEid
- Upx
- ...

Cette liste n'est pas exhaustive et **devra être complétée** en fonction des préférences de chacun.

Upx est brièvement cité.
Il s'agit d'un logiciel permet de packer et depacker une binaire.
Cette notion bien que fondamentale sera présente dans la version avancée de ce cours.

Outil : (https://processhacker.sourceforge.io/)

2. Analyse statique simple

Analyse statique simple

Les empreintes numériques sont un vrai atout.

imphash : il s'agit de l'empreinte de la table des imports. Cette notion sera complétée dans ce support.

ssdeep : Il s'agit d'une empreinte qui variera peu s'il y a peu de modification entre les deux éléments analysés (LSH / CTPH)

imphash :
http://9b113d1a.blogspot.com/2019/01/concepte-de-limphash.html
ssdeep :
https://medium.com/@glaslos/locality-sensitive-fuzzy-hashing-66127178ebdc

2. Analyse statique simple

Analyse statique simple

Calcul de la **similarité**.

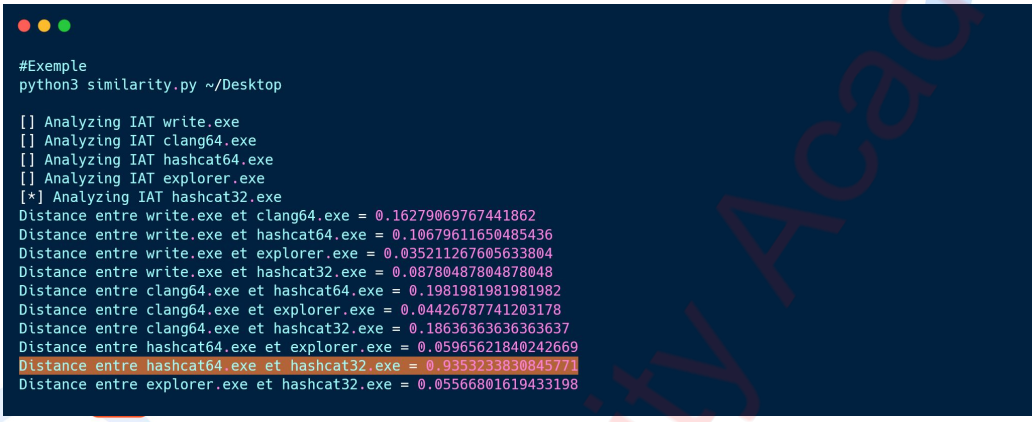
Indice et distance de Jaccard permettent de calculer la similarité et la diversité entre deux éléments.

```
def jaccard(list1, list2):
    intersection = len(list(set(list1).intersection(list2)))
    union = (len(list1) + len(list2)) - intersection
    return float(intersection) / union
```

2. Analyse statique simple

Analyse statique simple

La distance peut se faire sur les strings extraites des binaires, sur les API et sur de nombreux autres éléments



L'exemple ci-dessus présente le calcul de similarité entre deux binaires basé sur la table des imports.
En fonction de la base de données des malwares, c'est un outil intéressant.

2. Analyse statique simple

Analyse statique simple : Limites

Légère et peu coûteuse en terme de ressource, elle présente néanmoins de nombreux facteurs rendant compliqué ce type d'analyse :

- **Offuscation** : Rend plus difficile l'analyse au travers de nombreux mécanismes (polymorphisme, métamorphisme ...);
- **Techniques anti-disassembly** : Regroupe différents type d'offuscation (API, Opcode) mais aussi des techniques telles que (CFG flattening)
- **Packing** : Processus par lequel l'auteur compresse et chiffre le malware.
    - Ce sujet est aussi vaste qu'important. Un chapitre lui sera dédié dans la version avancée de ce cours.

Control Flow Graph Flattening est une technique qui consiste à regrouper tous les basic bloc dans boucle infinie avec un switch pour contrôler le tout.

Un basic bloc est la suite des instructions composants le programme et découpé selon les structures conditionnelles.

Complément : https://fr.wikipedia.org/wiki/Bloc_de_base
https://blog.jscrambler.com/jscrambler-101-control-flow-flattening/