3 - Introduction à la virologie

Jérémy Rubert

rubert.jeremy@gmail.com

29 octobre 2025

Introduction

- **Malware**, contradiction de **Malicious** et de **Software** = code malveillant
- **Malveillant**
  - Notion non formalisable : suite d'appels « légitime » de l'OS
  - Suite d'actions « illégitime » pour l'utilisateur et les outils de détection

  → Particularité au niveau du code
    - Protection du code
    - Furtivité
    - Persistance

Plan

Plan

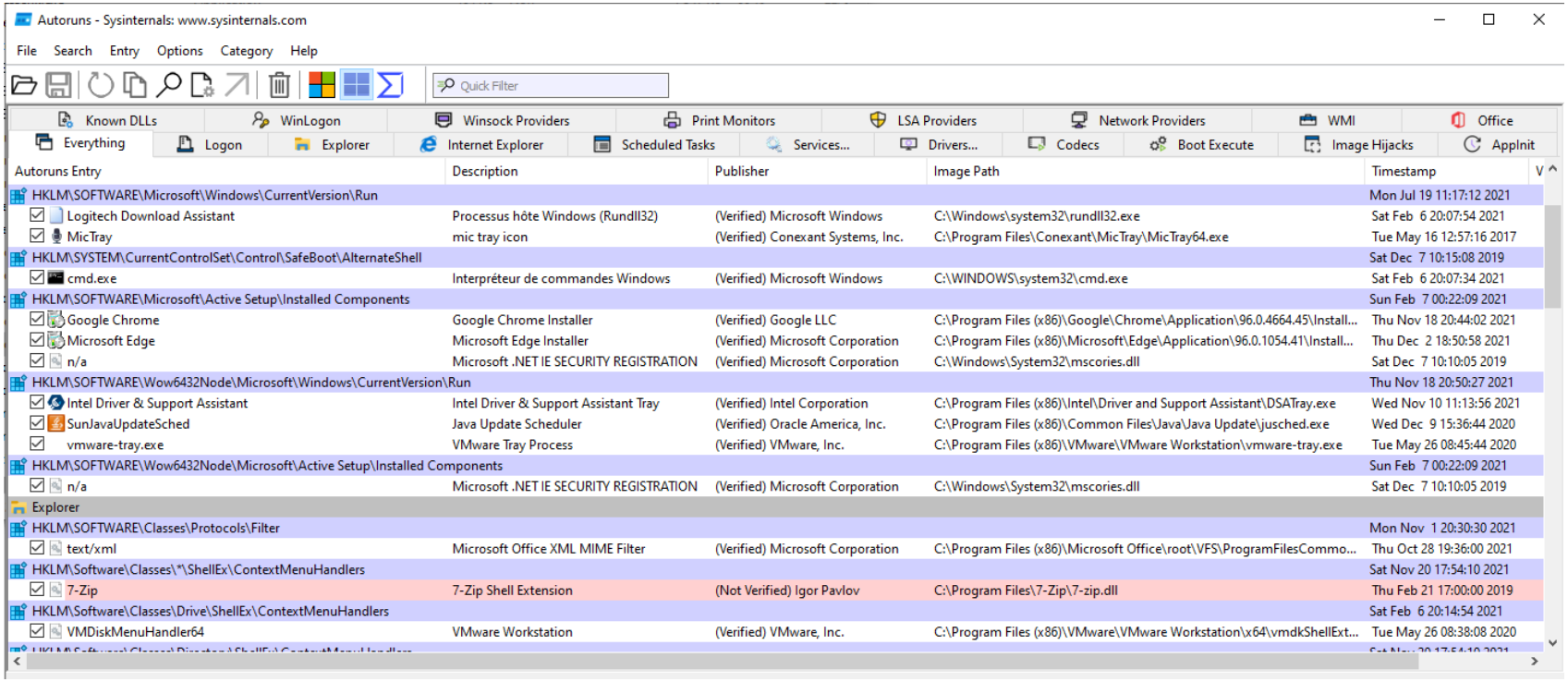
Primo-infection

- **Social engineering**
  - **Phishing** (hameçonnage) : campagne de mail massive pour inciter les utilisateurs à ouvrir une pièce jointe malveillante ou à fournir des données personnelles
  - **Spear phising** (harponnage) : phishing ciblé
- **Water holing** (attaque du point d'eau) : compromission d'un site utilisé par la/les victime(s)
- **Vulnérabilités** : exploitation d'un service vulnérable afin de réaliser une exécution de code distante et ou privilégiée
  - Campagne automatique pour ratisser large
  - Utilisation ciblé d'une vulnérabilité
  - Vulnérabilités 0-day ou 1-day

Persistance

- Consiste pour le malware à assurer sa survie sur le système en cas de redémarrage
- Multiples techniques de base
  - Registre
  - Tâches planifiées
  - Services
  - Shell,
  - ...
- Point de départ pour l'analyse : **autoruns.exe** de la suite SysInternals

Persistance



Persistance : Techniques plus avancées

- **Hijacking** : remplacer un élément légitime par un autre illégitime
  - DLL/EXE
  - COM
  - raccourcis
- **Association de fichiers** : via plusieurs clés de registre
- **Squatting** : ajouter un élément malveillant recherché par défaut par un logiciel légitime voir le système lui-même
- **DLL search order** :
  1. Le répertoire où l'application est installée
  2. Le répertoire système (%SystemRoot%\System32)
  3. Le répertoire de windows (%SystemRoot%)
  4. Le répertoire courant
  5. Les répertoires définis dans la variable d'environnement PATH

Furtivité

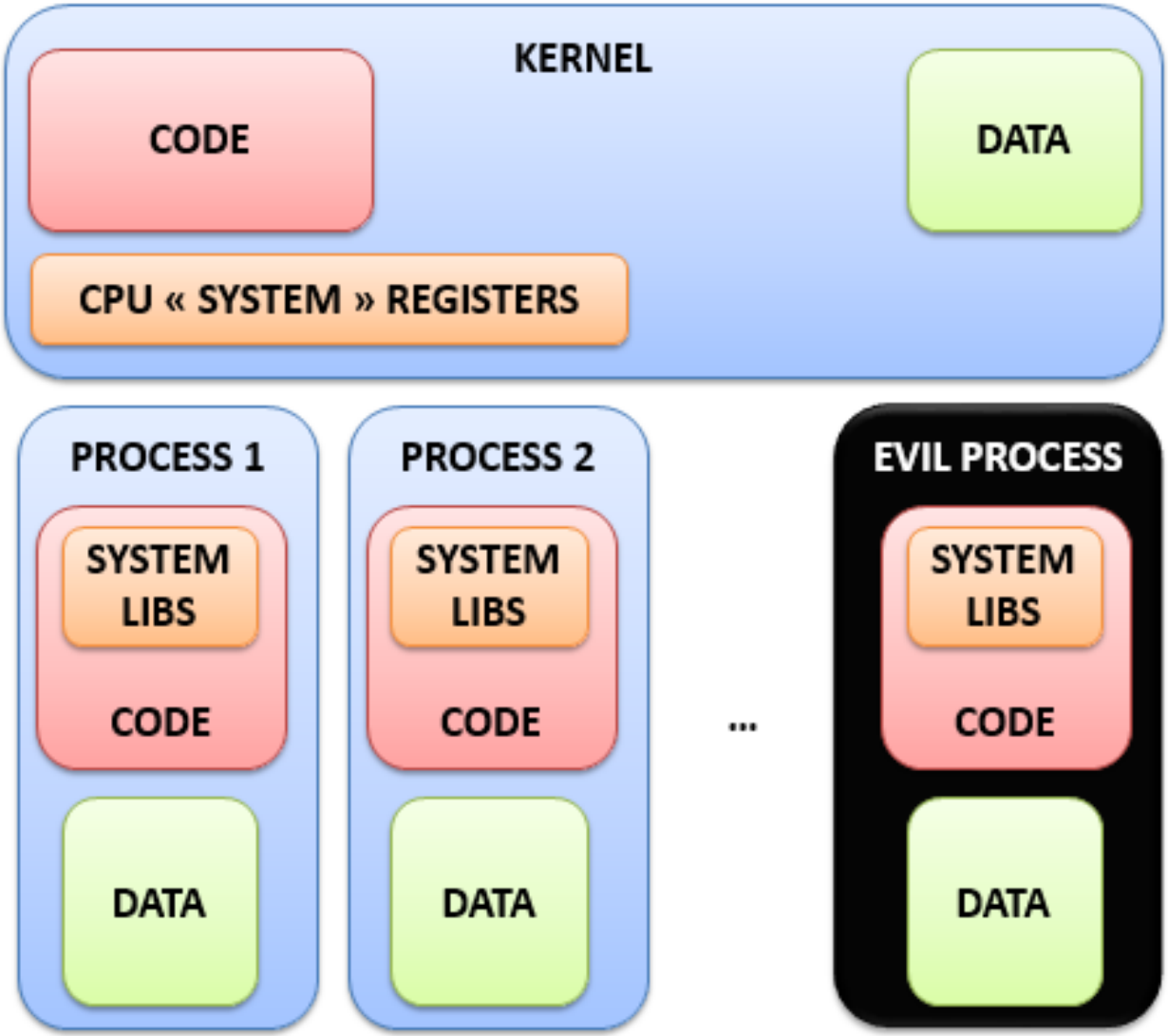
Definition

Consiste à passer inaperçu pour l'utilisateur et les outils de détection : non détection

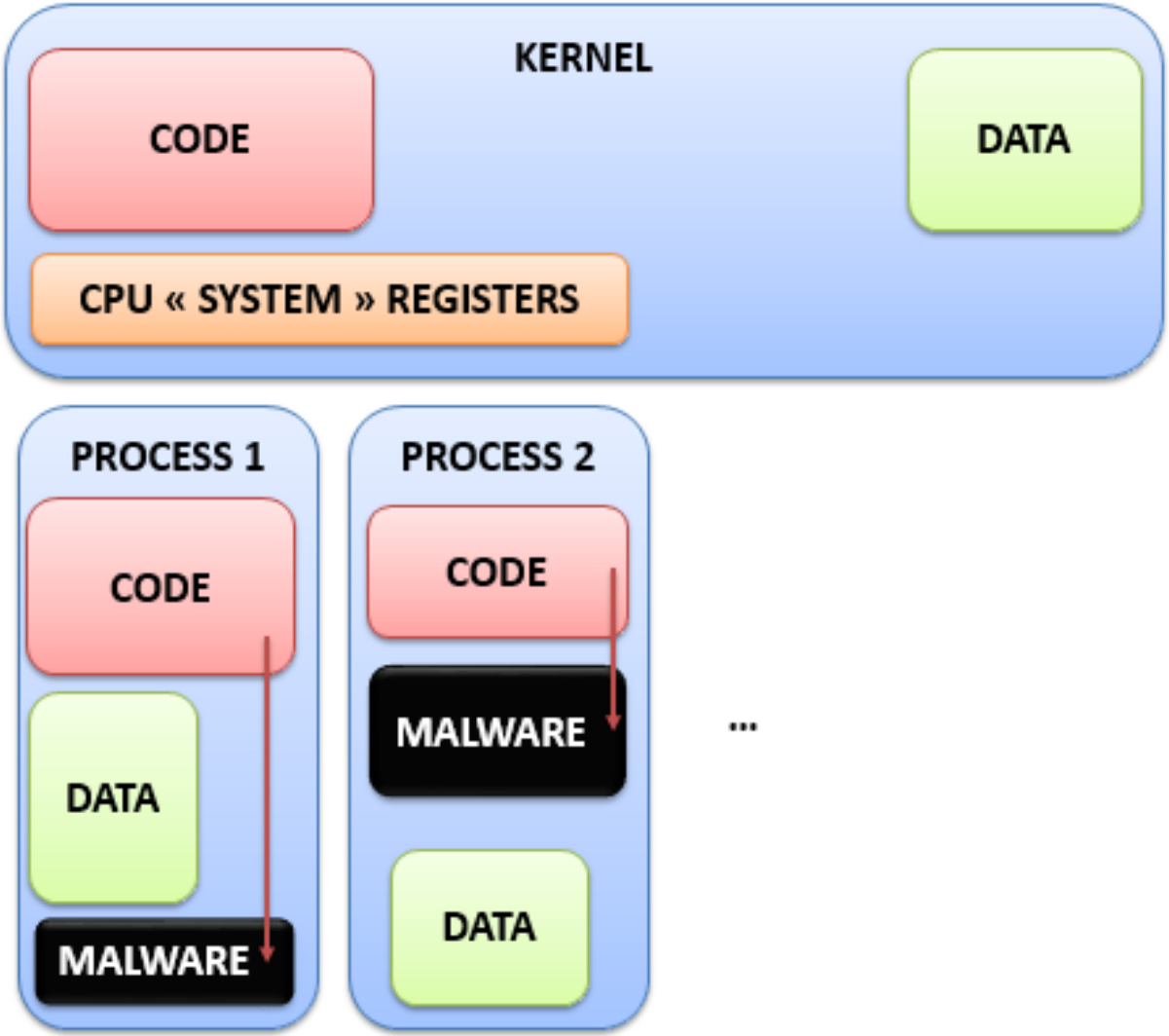
Techniques :

- **Mimétisme** : trojan (cheval de Troie)
- **Infection** : virus
- **Modifications** de l'OS : rootkits
- **Minimisation** des **interactions** avec l'OS

Furtivité : mimétisme

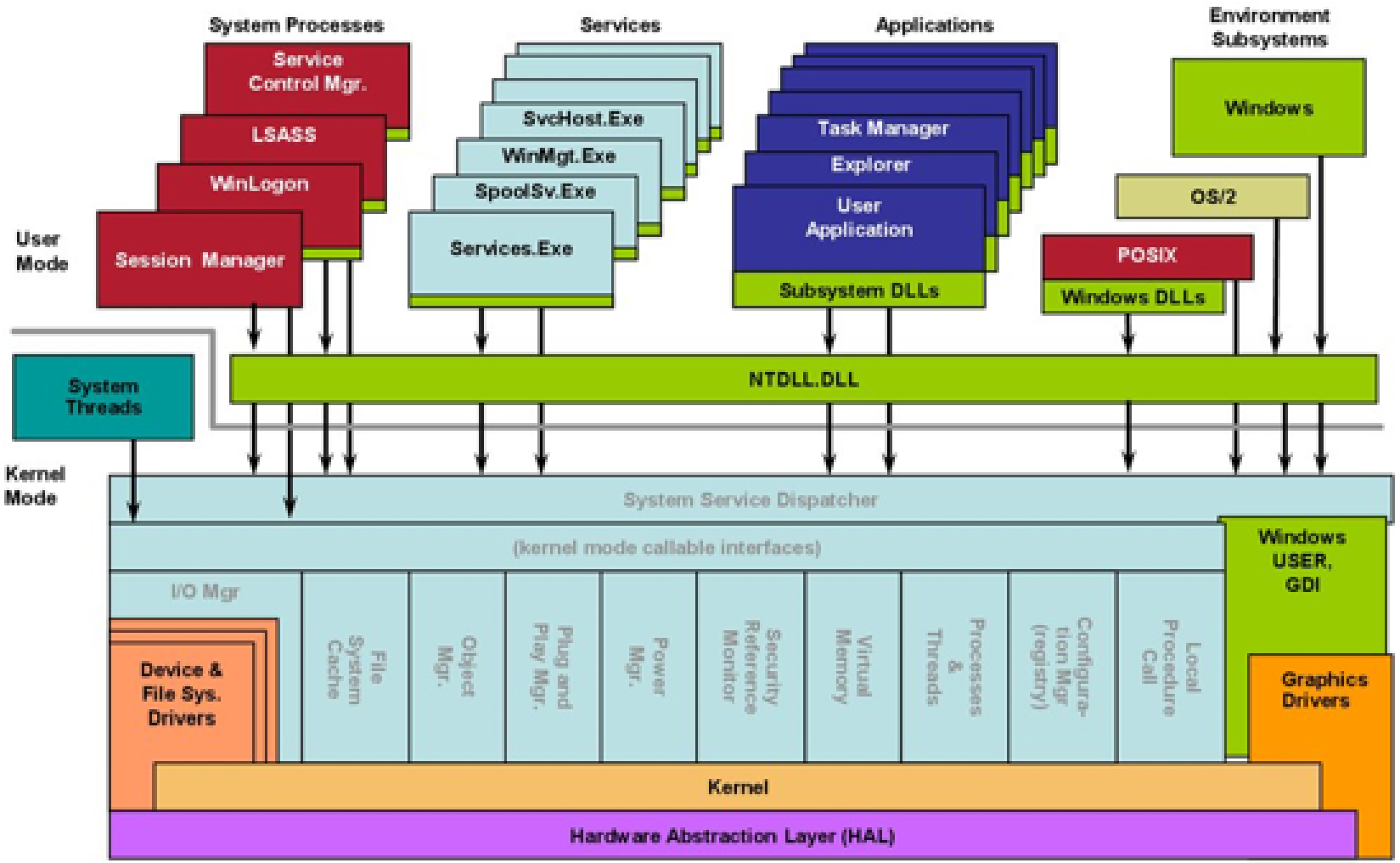


Furtivité : infection



Furtivité : interactions

Minimisation des interactions avec l'OS

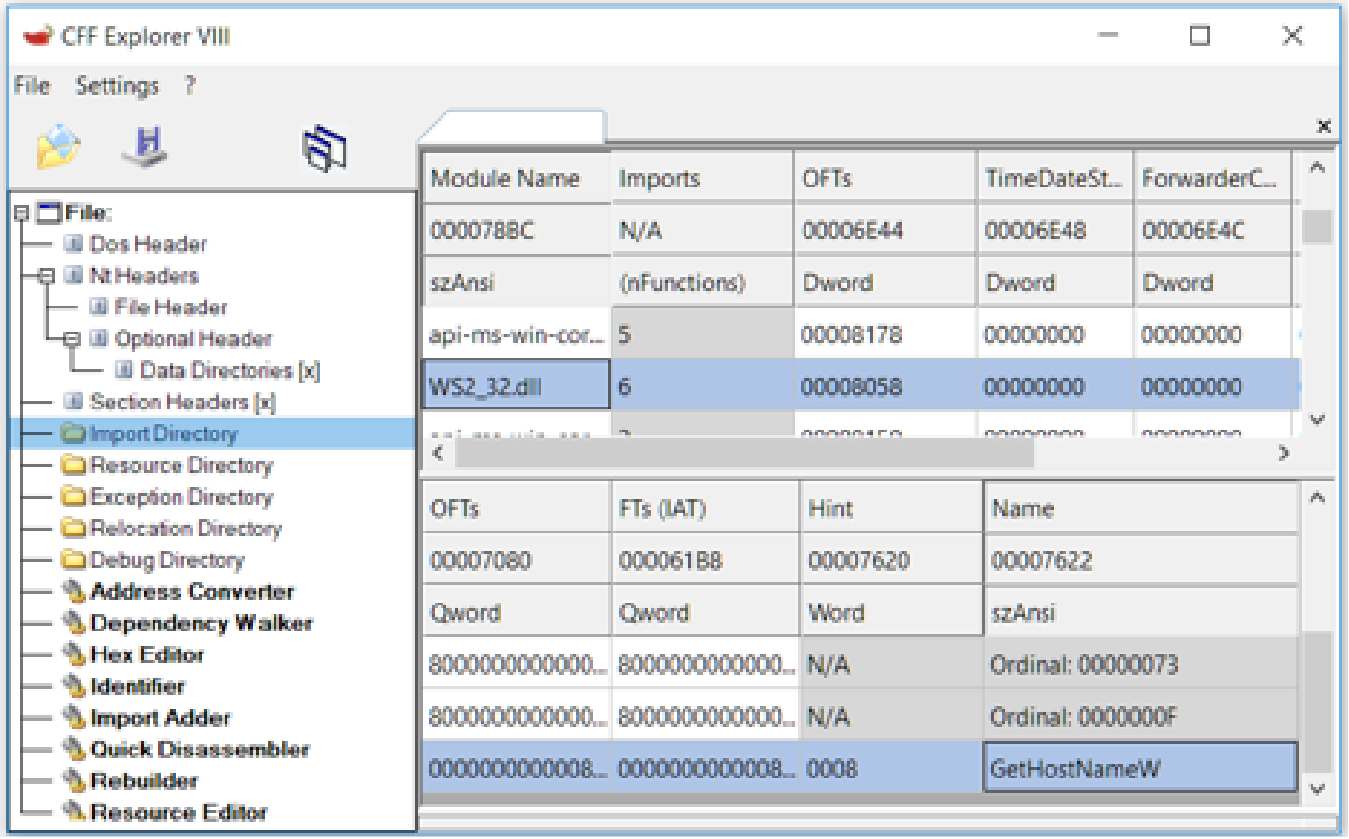


Furtivité : interactions

Exemple

GetHostNameW dans **ws2_32.dll**

1. Appel Direct



Furtivité : interactions

2. Résolution dynamique via **GetModuleHandleA** et **GetProcAddress**

```
//...
HMODULE hWinsock = GetModuleHandleA("ws2_32.dll");
FARPROC GetHostNameFn = GetProcAddress(hWinsock, "GetHostNameW");
//...
```

Furtivité : interactions

2. **GetProcAddress** recodé (on peut faire de même pour **GetModuleHandleA**)

```
FARPROC WINAPI SelfGetProcAddressA(HMODULE hModule, ULONG ProcHashValue){
  // declarations omitted...
  pImageDosHeader = (IMAGE_DOS_HEADER*)hModule;
  pImageNtHeaders = (IMAGE_NT_HEADERS*)(pModuleBase+pImageDosHeader->e_lfanew);
  pImageExportDirectory =
  (pBase+pImageNtHeaders->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_EXPORT].VirtualAddress);

  AddressOfFunctionsRvaTbl= (DWORD*) (pBase+pImageExportDirectory->AddressOfFunctions);
  AddressOfNamesRvaTbl    = (DWORD*) (pBase+pImageExportDirectory->AddressOfNames);
  AddressOfNameOrdinalsTbl= (WORD*)  (pBase+pImageExportDirectory->AddressOfNameOrdinals);

  for(ExportIndex=0; ExportIndex < pImageExportDirectory->NumberOfNames; ExportIndex++){
    FunctionName = pBase+AddressOfNamesRvaTbl[ExportIndex];
    if (crc32((const char*)FunctionName) == ProcHashValue){
      OrdinalIndex = AddressOfNameOrdinalsTbl[ExportIndex];
      ProcAddress  = (FARPROC)(pBase + AddressOfFunctionsRvaTbl[OrdinalIndex]);
      goto End;
    }
  }
  End:
  return (ProcAddress);
}
```

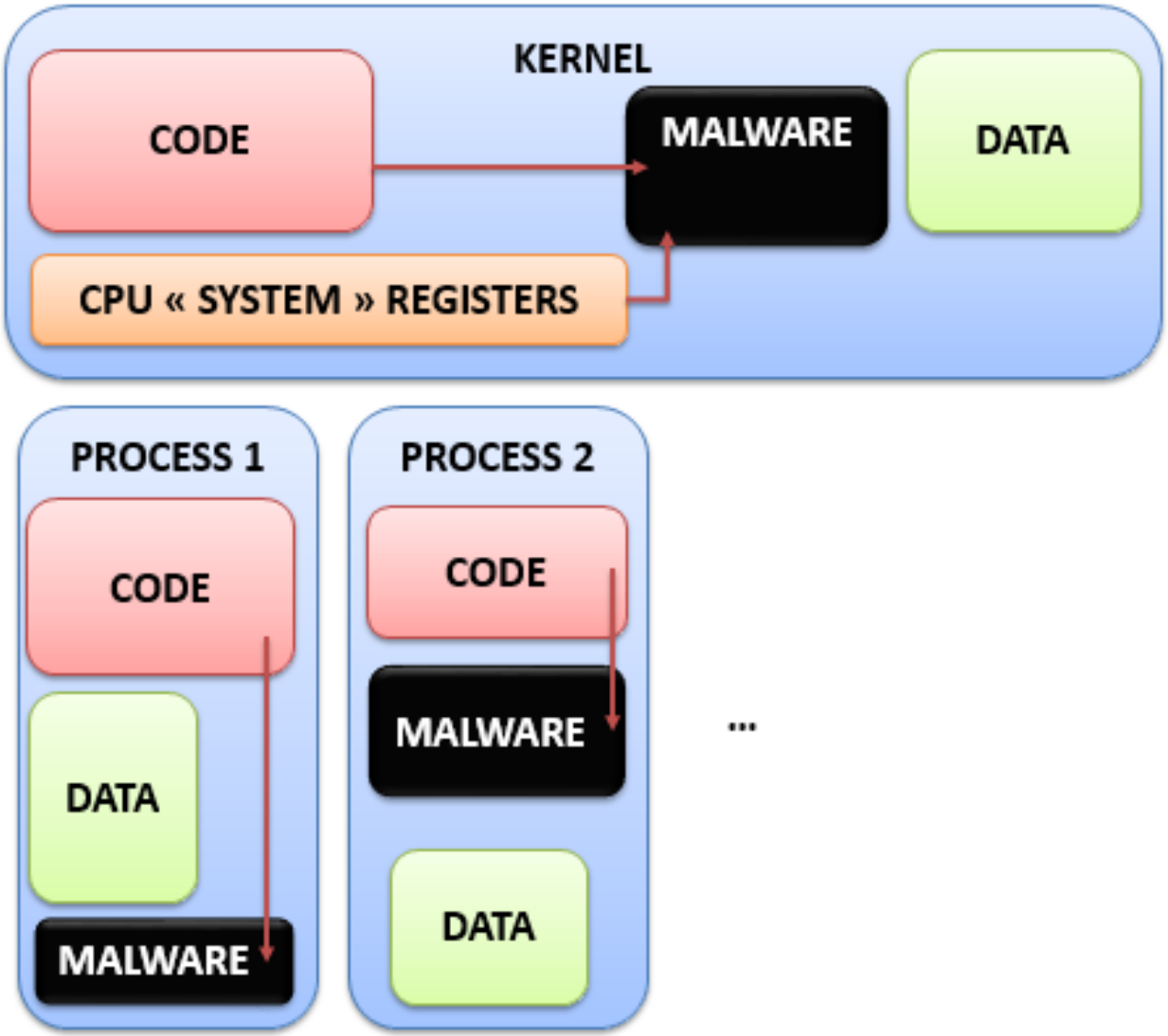
Furtivité : modification de l'OS

Différents niveaux de furtivité (Joanna Rutkowska)

- Type 1 : modification des constantes du système
- Type 2 : modification des variables du système
- Type 3 : virtualisation du système

Furtivité : modification de l'OS

Type 1 : modification des constantes du système



Furtivité : modification de l'OS

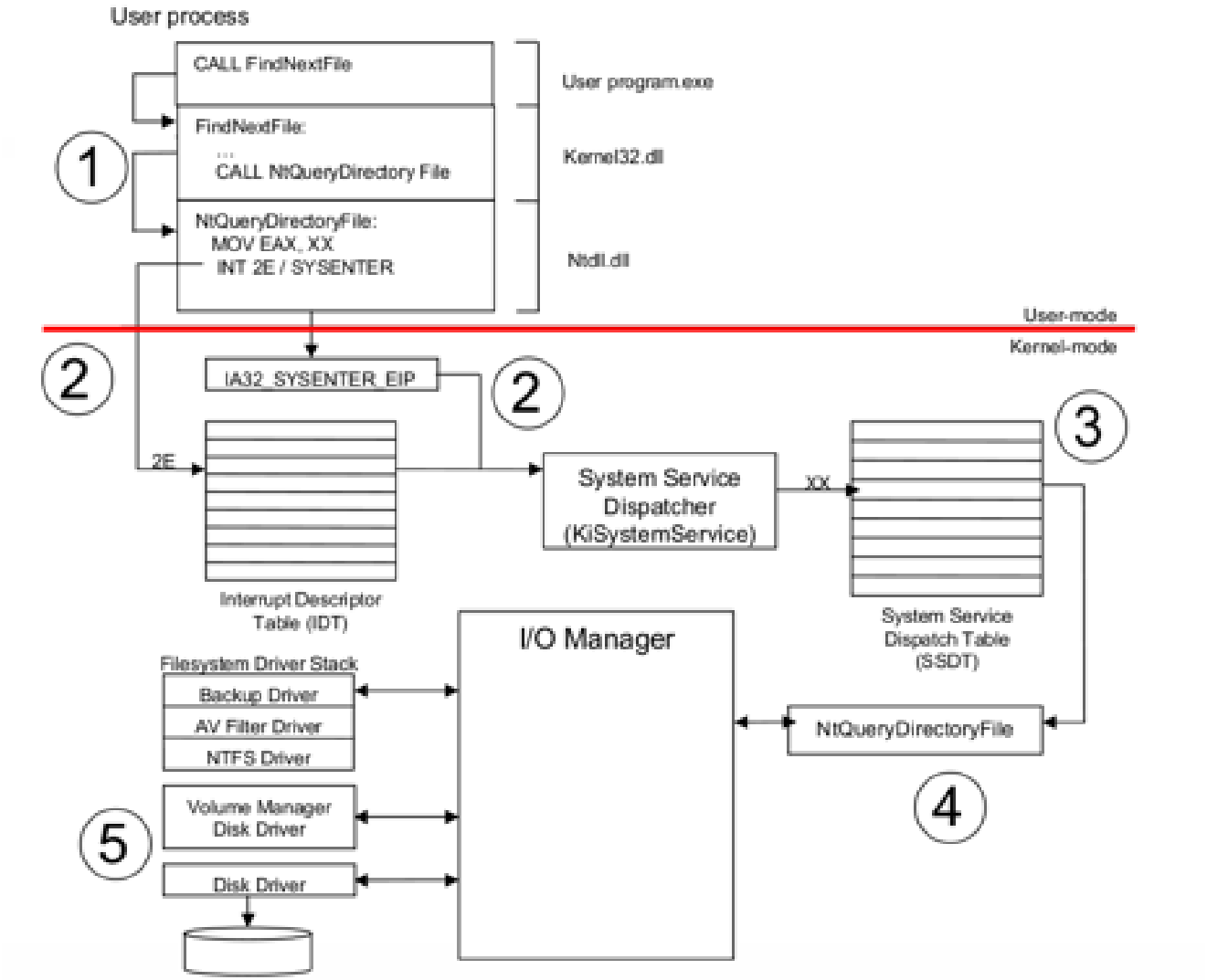
Type 1 : modification des constantes du système

Fonctionnement :

- Kernel Land : hook des principales tables du noyaux
  - SSDT (System Service Dispatch Table)
    - Modification lors d'appels systèmes
  - IDT (Interrupt Descriptor Table)
    - Modification lors d'une interruption
  - Drivers
- User Land :
  - Hook de l'IAT (Import Adress Table)
    - Modification lors de l'appel d'une fonction importée
  - Injection de DLL

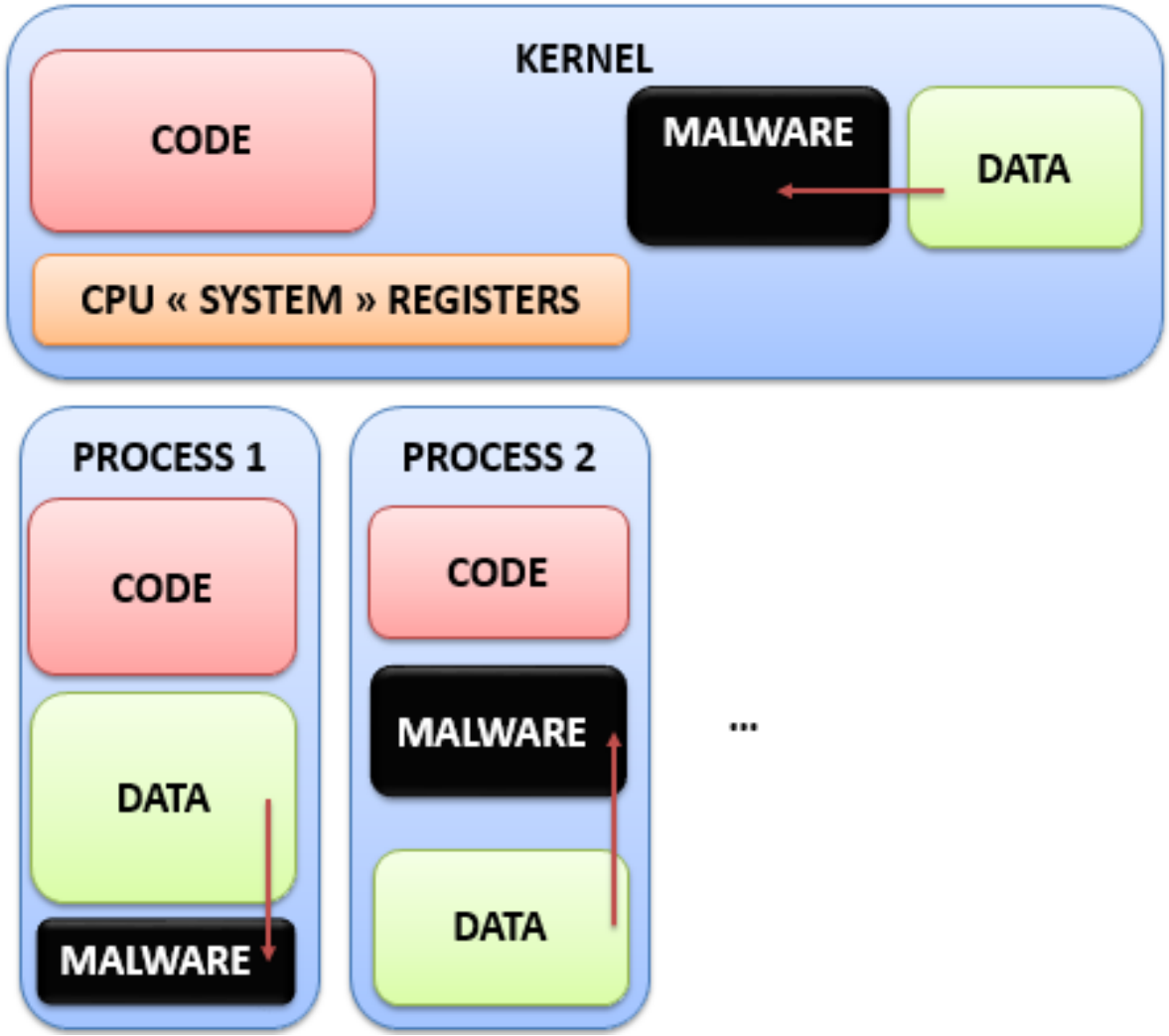
Furtivité : modification de l'OS

Type 1 : modification des constantes du système



Furtivité : modification de l'OS

Type 2 : modification des données



Furtivité : modification de l'OS

Type 2 : modification des données

Fonctionnement :

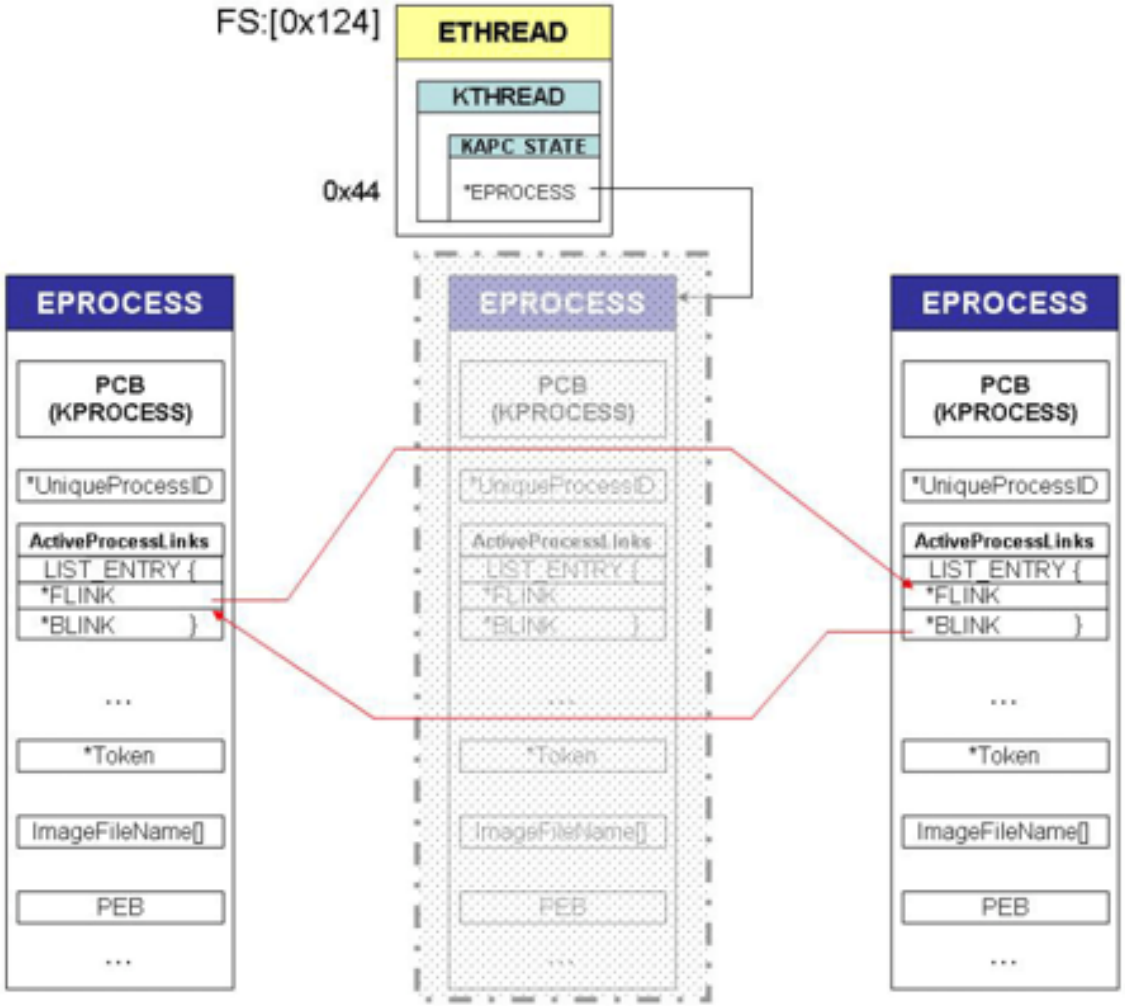
- Exclusivement Kernel Land
- DKOM (Direct Kernel Object Modification)

Exemple

Modification de la liste chaînée des processus (EPROCESS)

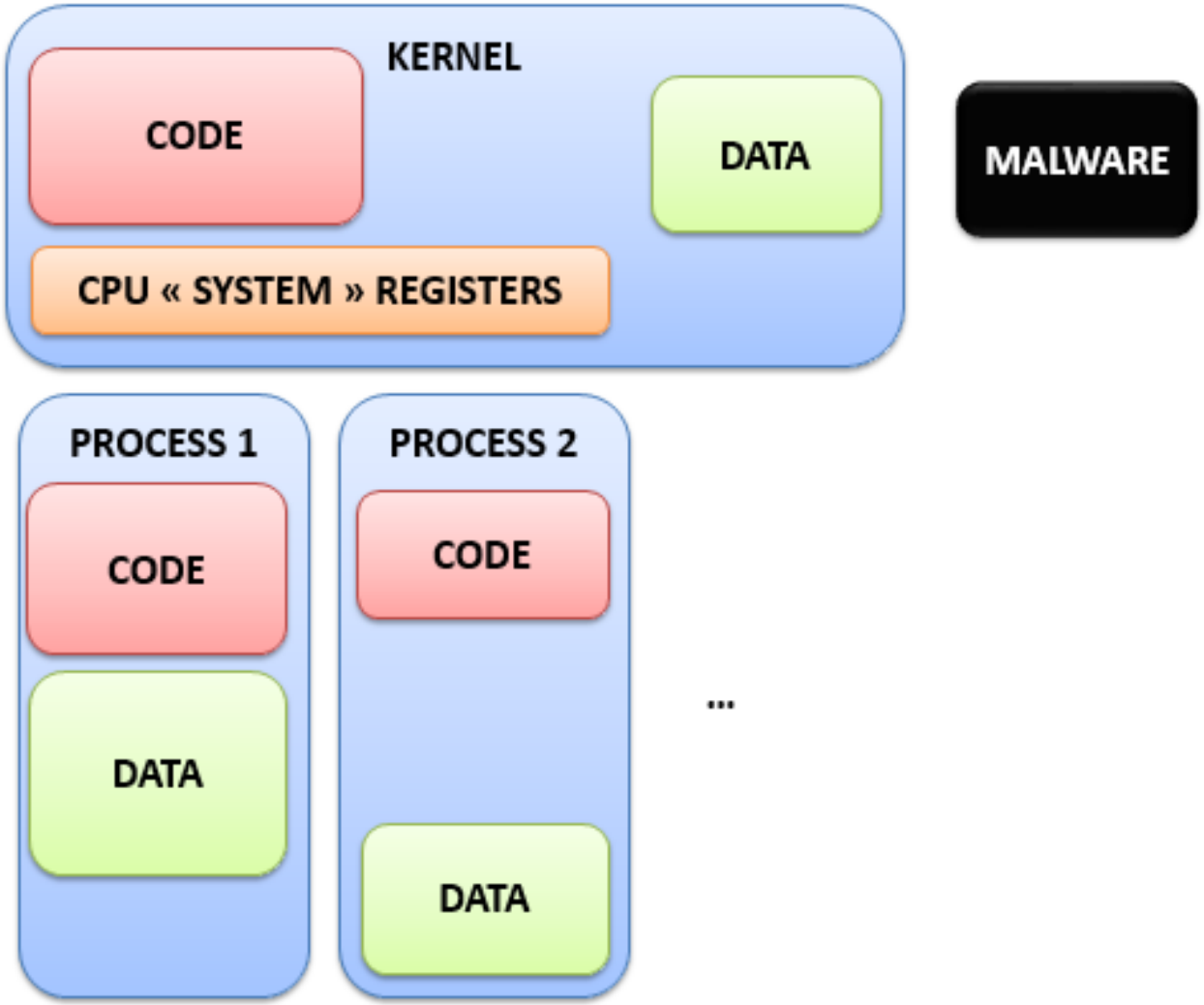
Furtivité : modification de l'OS

Type 2 : modification des données



Furtivité : modification de l'OS

Type 3 : virtualisation du système



Furtivité : modification de l'OS

Type 3 : virtualisation du système

Fonctionnement :

- Utilisation des instructions de virtualisation pour virtualiser l'OS (AMD-V et Intel VT-x)

Attention

Le rootkit ne peut pas utiliser de service de l'OS ⇒ il doit interagir directement avec le matériel

Exemple

BluePill (AMD-V) et Vitriol (Intel VT-x)

Protection contre l'analyse

Consiste à se prémunir autant que possible d'une analyse (statique et dynamique)

- Protection du code :
  - Chiffrement
  - Polymorphisme
  - Métamorphisme
  - Packers
- Détection des débogueurs
- Détection d'instrumentation de code (DBI)
- Détection de VM

Protection contre l'analyse
Chiffrement

- Technique ancienne (1990) : Cascade.1701
- Idée : déchiffrer le corps du virus, l'exécuter et le chiffrer avec une nouvelle clé.
- Chiffrement faible mais suffisant pour contourner une détection par signature

```
            lea si, Start
            mov sp, 0682
Decrypt:    xor [si],si
            xor [si],sp
            inc si
            dec sp
            jnz Decrypt
Start:
            ; Encrypted/Decrypted Virus Body
```

- Mais la routine de déchiffrement est un motif potentiel de détection

Protection contre l'analyse

Polymorphisme

- Technique apparue en 1990 : virus 1260
- Idée : en plus du chiffrement, changer la forme de la routine de déchiffrement

1. **Insertion de code mort**
2. Substitution de registres
3. Permutation d'instructions
4. Substitution d'instructions

Exemple	Signification
add reg, 0	$reg \leftarrow reg + 0$
mov reg, reg	$reg \leftarrow reg$
or reg, 0	$reg \leftarrow reg \vert 0$
and reg, -1	$reg \leftarrow reg \ \& -1$

Protection contre l'analyse
Polymorphisme

- Technique apparue en 1990 : virus 1260
- Idée : en plus du chiffrement, changer la forme de la routine de déchiffrement

1. Insertion de code mort
2. **Substitution de registres**
3. Permutation d'instructions
4. Substitution d'instructions

Programme 1	Programme 2
pop edx	pop eax
mov edi, 04h	mov ebx, 04h
mov esi, ebp	mov edx, ebp
mov eax, 0Ch	mov edi, 0Ch
add edx, 088h	add eax, 088h

Protection contre l'analyse

Polymorphisme

- Technique apparue en 1990 : virus 1260
- Idée : en plus du chiffrement, changer la forme de la routine de déchiffrement

1. Insertion de code mort
2. Substitution de registres
3. **Permutation d'instructions**
4. Substitution d'instructions

Programme 1	Programme 2

mov ecx, 104h
mov esi, [ebp+0xh]
mov edi, [rbp+08h]
repnz movsb
	mov edi, [rbp+08h]
	mov ecx, 104h
	mov esi, [ebp+0xh]
	repnz movsb

Protection contre l'analyse
Polymorphisme

- Technique apparue en 1990 : virus 1260
- Idée : en plus du chiffrement, changer la forme de la routine de déchiffrement

1. Insertion de code mort
2. Substitution de registres
3. Permutation d'instructions
4. **Substitution d'instructions**

Instruction simple	Instructions multiples
xor reg, reg	mov reg, 0
mov reg, Imm	push Imm pop reg

op reg1, reg2	mov mem, reg2
	op mem, mem
	mov reg1, mem

Protection contre l'analyse
Métamorphisme

Idée : muter l'intégralité du code

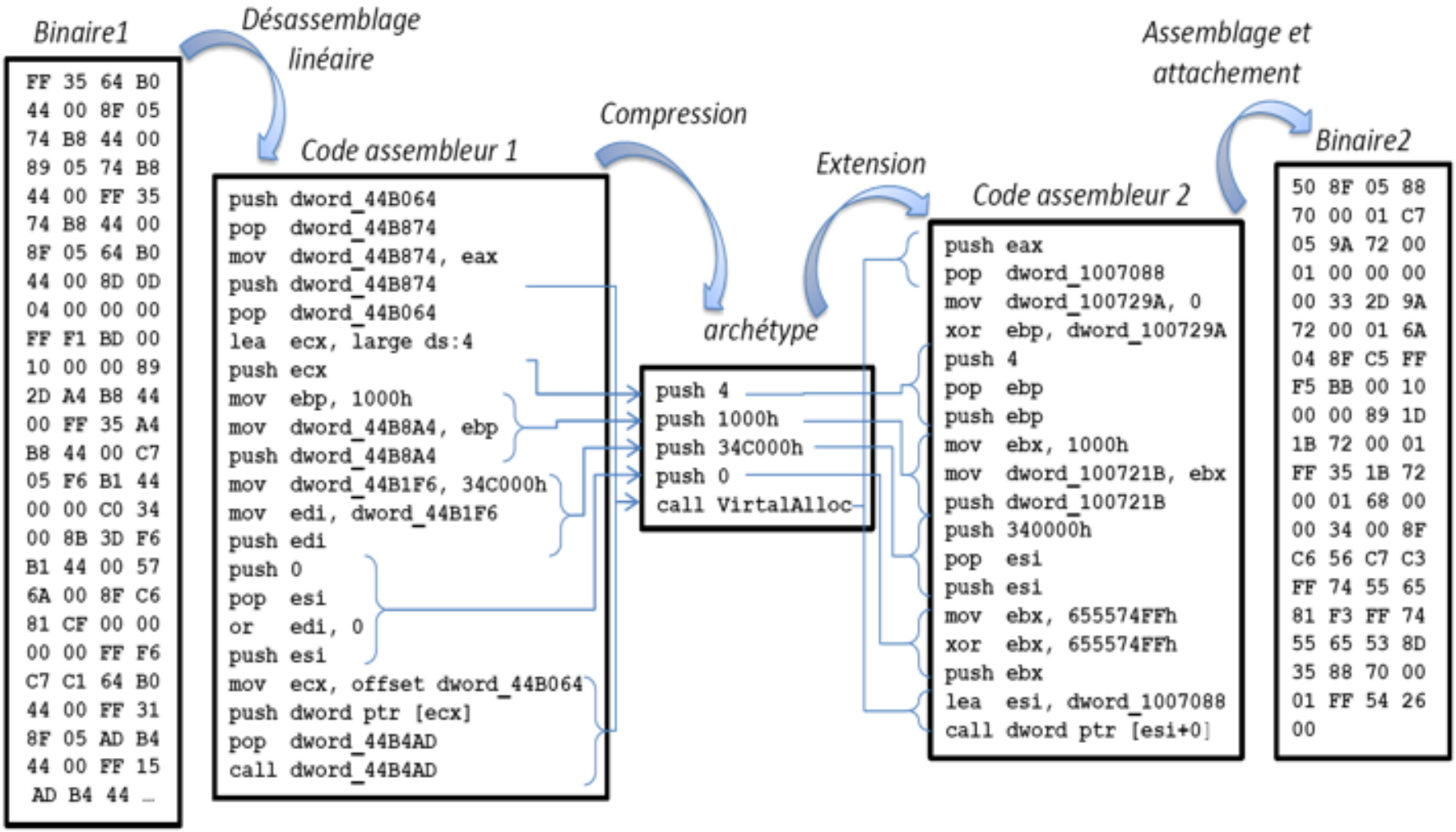
Exemple

Win32.MetaPHOR, réplication en 5 étapes

1. Désassemblage et dépermutation : représentation de ses instructions en pseudo-code
2. Compression : utilisation de règles de réécritures (multiples instr. → simple instr.)
3. Permutation du code au moyen de saut inconditionnels
4. Extension : utilisation de règles de réécritures (simple instr. → multiples instr.)
5. Assemblage : production du nouveau binaire à partir du pseudo-code.

Protection contre l'analyse

Métamorphisme



Protection contre l'analyse
Packers

- À l'origine conçus pour diminuer la taille d'un programme
- Aujourd'hui utilisés pour :
  - Protéger la propriété intellectuelle (ralentir la rétro-conception)
  - Créer des variantes syntaxiques de malwares

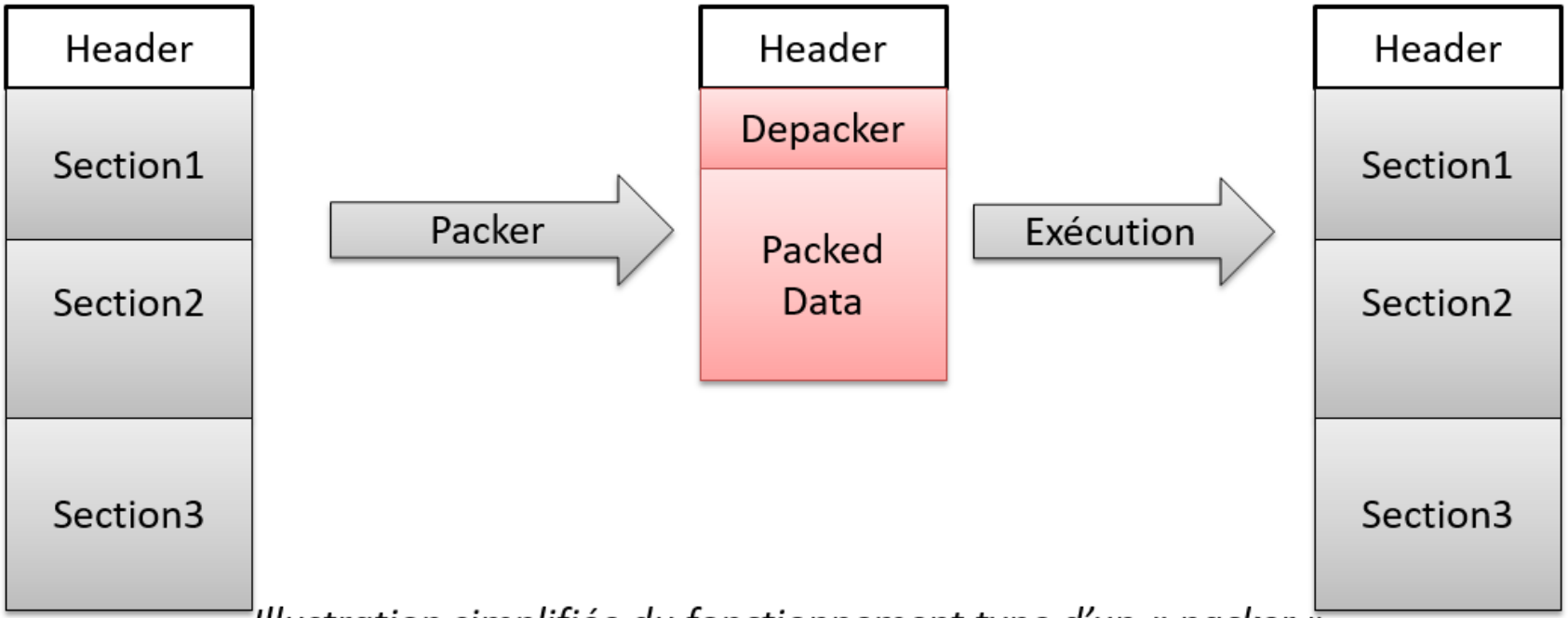


Illustration simplifiée du fonctionnement type d'un « packer »

Protection contre l'analyse

Packers

Exemple

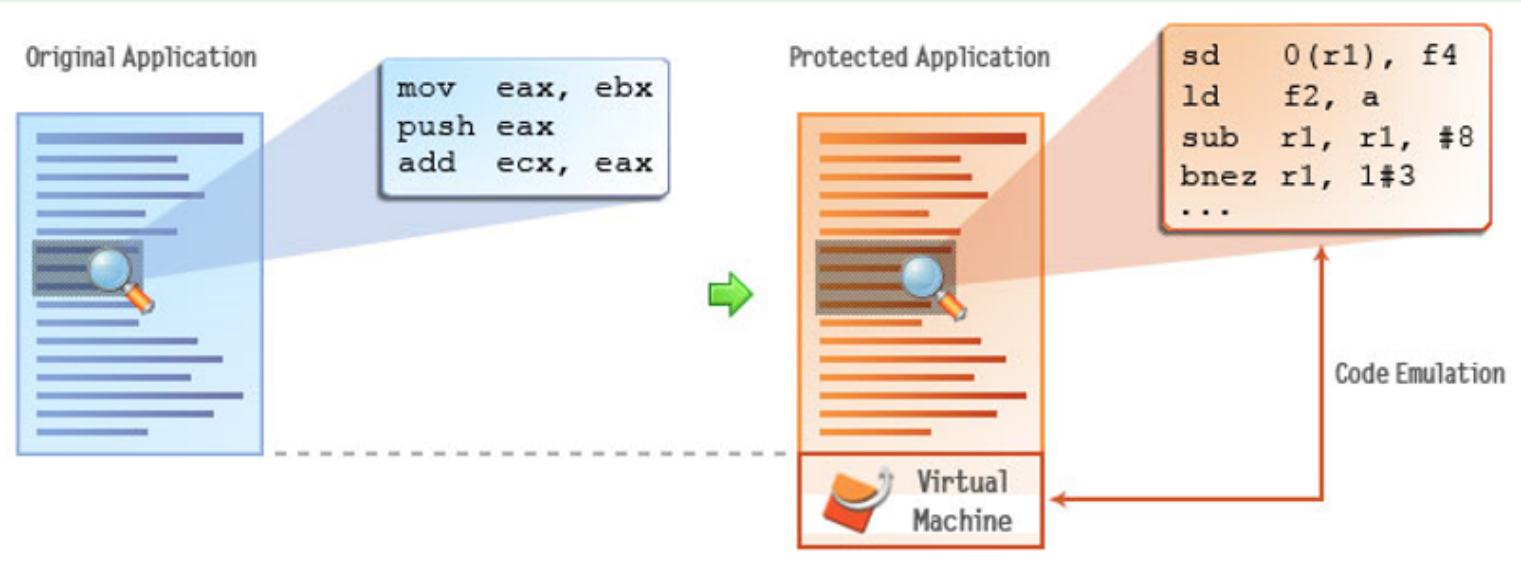
UPX

- Suppression des imports d'origines
  - Kernel32.dll
    - LoadLibraryA
    - GetProcAddress,
    - VirtualAlloc/Free/Protect
    - ExitProcess
- Suppression des sections d'origine
  - UPX0
  - UPX1
  - UPX2
  - .idata pour les imports

Protection contre l'analyse

Packers

Exemple : VMProtect



Fonctionne au moyen de marqueurs

```
#include "VMProtectSDK.h"

VMProtectBegin(MARKER_NAME);
...
VMProtectEnd();
```

Protection contre l'analyse
Détection des débogueurs

- Détection de points d'arrêt
- API de détection
- Flags de debug
- Timing
- Auto-debug

Protection contre l'analyse
Détection des débogueurs

- Détection de points d'arrêt
  - **Logiciels :** 0xCC pour x86
    - Vérification de la présence de 0xCC à la place du code d'origine
    - Plus globalement, vérification de l'intégrité du code (si non auto-modifiant)
  - **Matériels :** registres DR
    - Récupération et modification des registres de debug via GetThreadContext et SetThreadContext
    - Même chose via la gestion d'exception (SEH) qui permet d'accéder au contexte du thread

Protection contre l'analyse
Détection des débogueurs

- API de détection
  - **IsDebuggerPresent :** détermine si le processus appelant est en cours de debug
  - **CheckRemoteDebuggerPresent :** détermine si le processus spécifié est en cours de debug
  - **NtCreateDebugObject/NtQueryObject :** création d'un objet de debug pour un processus donné, ensuite vérification du nombre de handle sur cet objet : 1 pas de debug, >1 debug
  - **CloseHandle/NtClose :** génère une exception STATUS_INVALID_HANDLE sur un handle invalide en cas de debug

Protection contre l'analyse
Détection des débogueurs

- Flags de debug
  - **TrapFlag :** TF du registre EFLAGS, lorsque ce flag est à 1, le CPU génère une interruption 1 après l'exécution d'une instruction (single step)
  - **BeingDebugged :** flag dans le PEB
  - **KdDebuggerEnabled :** flag de la structure KUSER_SHARED_DATA, permet de savoir si le debug kernel est actif

Protection contre l'analyse
Détection des débogueurs

- Timing
  - **GetTickCount :** nombre de millisecondes depuis de dernier démarrage du CPU
  - **QueryPerformanceCounter :** compteur « haute performance » (<1us)
  - **rdtsc** (**R**ead **T**ime **S**tamp **C**ounter) : instruction permettant de connaître le nombre de ticks du CPU depuis le dernier démarrage

Protection contre l'analyse
Détection des débogueurs

- Auto-debug
  - Processus fils débogué par un processus père

  ⇒ **DebugActiveProcess()** échoue sur le fils

Protection contre l'analyse
Détection d'instrumentation de code (DBI)

- **Dynamic Binary Instrumentation (DBI) :** framework d'injection de code pour analyser un programme
  - **PIN** : framework propriétaire développé par Intel pour IA-32 et x86-64 
  - **DynamoRIO** : framework open-source pour x86/AMD64/ARM/AArch64 

Protection contre l'analyse
Détection d'instrumentation de code (DBI)

- **eXait** de coresecurity permet de détecter PIN de plus de 15 façons différentes
  - Chaînes de caractères
  - Présence du module pinvm.dll
  - Noms d'évènements et de sections (PIN_IPC...)
  - Détection du compilateur JIT
    - ntdll.dll hooks
    - Page permission RWX

Protection contre l'analyse
Détection de VM

Par défaut les machines virtuelles ne cherchent pas à être furtives à travers :

- Les instructions émulées
- Les processus et fichiers dédiés
- Leurs configurations
- Leur matériel

Protection contre l'analyse
Détection de VM

- Les instructions émulées
  - **CPUID**, EAX=1 : propriétés du processeur dans ECX avec bit 31 à 0 pour une machine physique et 1 pour une VM
  - **CPUID**, EAX = 0x40000000 : "hypervisor brand" (Microsof HV, VMwareVMware, etc)
  - **IN** avec EAX='VMXh' et EDX='VX', fonctionne uniquement avec VMWare, exception sinon

Protection contre l'analyse
Détection de VM

- Les processus et fichiers dédiés
  - Processus :
    - Vmwareuser.exe
    - Vboxservice.exe
    - ...
  - Drivers :
    - Vmmouse.sys
    - VBoxMouse.sys
    - VBoxGuest.sys
    - ...

Protection contre l'analyse
Détection de VM

- Leurs configurations
  - Registres :
    - HKEY_LOCAL_MACHINE\ HARDWARE\ACPI\DSDT\VBOX_
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\VirtualDeviceDrivers
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\SCSI\
    - ...

Protection contre l'analyse
Détection de VM

- Leur matériel préfixe des adresses MAC
  - 00 :05 :69 (Vmware)
  - 00 :0C :29 (Vmware)
  - 00 :1C :14 (Vmware)
  - 00 :50 :56 (Vmware)
  - 08 :00 :27 (VirtualBox)

Plan

Détection des malwares

Definition

- P ensemble des programmes et $M \subset P$ ensemble des malwares
- Fonction de détection $D : P \to \{0, 1\}$ définie $\forall p \in P$,
  $\begin{cases} D(p) = 1 & \text{si } p \in M \quad \text{(un positif)} \\ D(p) = 0 & \textit{sinon} \quad \text{(un négatif)} \end{cases}$

Détecteur parfait

Détecteur réel	Positif	Négatif
Positif	Vrai Positif (VP)	Faux Positif (FP)
Négatif	Faux Négatif (FN)	Vrai Négatif (VN)
	P	N

Fiabilité = $\frac{VP}{P}$
(tout ce qui doit être détecté l'est)

Pertinence = $\frac{VN}{N}$
(seul ce qui doit être détecté l'est)

Détection des malwares

Techniques de détection

Deux grandes approches de détection

	Statique	Dynamique
+	- Pas de risque liés à l'exécution - Efficace sur des programmes simples - Rapide - Facile à déployer	- Résiste aux mutations de code - Peut détecter de nouvelles menaces
−	- Peu robuste face à de nouvelles menaces - Nécessite un base de connaissances (signatures) importantes - Peu robuste face aux techniques de mutation de code	- Potentiellement risqué (exécution du code malveillant) - Exécution liée à l'environnement - Analyse un unique chemin - Difficile à mettre en place

Détection statique

Signatures

Definition

Motif constant (expression régulière) permettant d'identifier un code binaire

- Si le motif est présent dans un autre binaire ⇒ faux positif
- Si le motif n'est pas trouvé dans une variante ⇒ faux négatif

Attention

Inadaptées pour les codes mutants (auto-modifiants ou polymorphes)

Détection statique
Heuristiques

- Attributs de section suspect (W+X)
- Taille virtuel incorrect
- Redirection du point d'entrée (JMP)
- Nom de section (exécution en .reloc ou .debug, etc.)
- Imports par ordinaux de la Kernel32.dll
- Inconsistance dans les headers (taille du code / données)
- etc.

Détection statique
Identification de graphe

Definition (Graphe de flot de contrôle (GFC))

Graphe orienté (N, E) avec N ensemble de sommets du graphe et E ensemble des arêtes. Chaque sommet est un bloc de base (« Basic Block »), i.e. une suite d'instructions consécutives se terminant par :

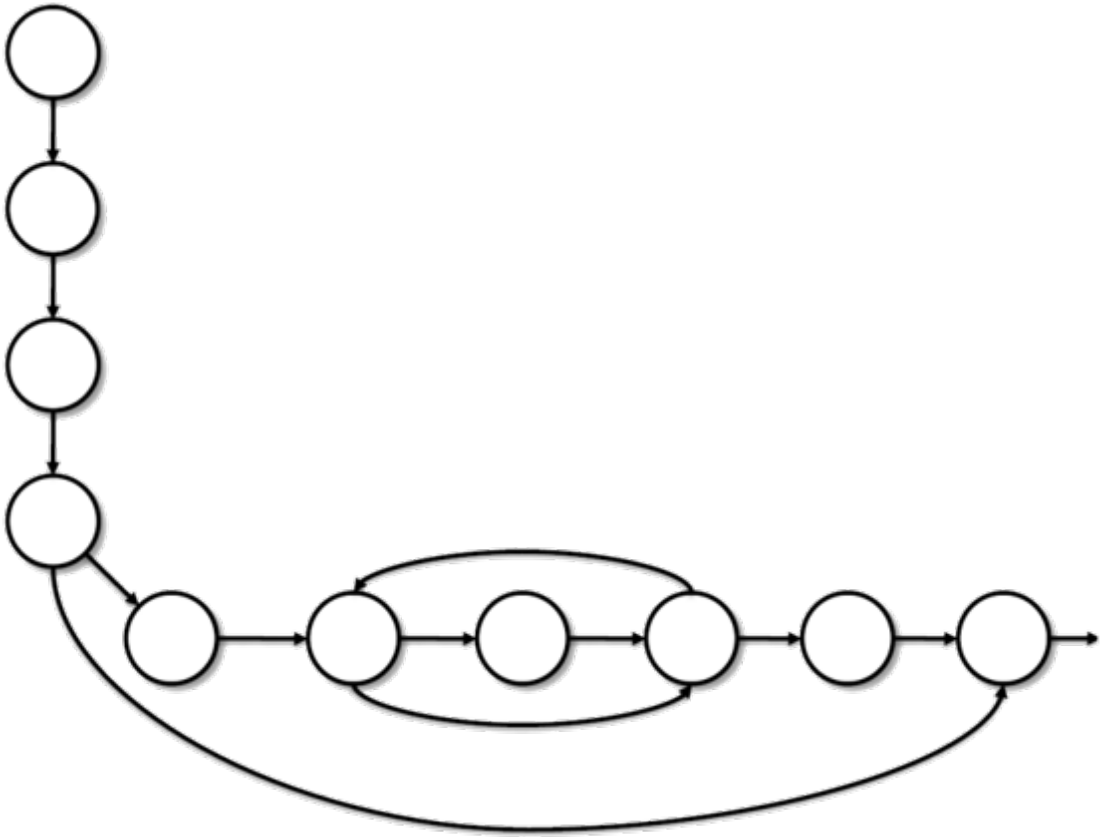
- Soit une instruction de transfert ;
- Soit une instruction séquentielle suivie d'une instruction appartenant à un autre bloc de base.

Chaque arrête e issue d'un bloc de base est une sortie conditionnelle ou inconditionnelle de ce bloc.

Détection statique

Identification du graphe de flot de contrôle

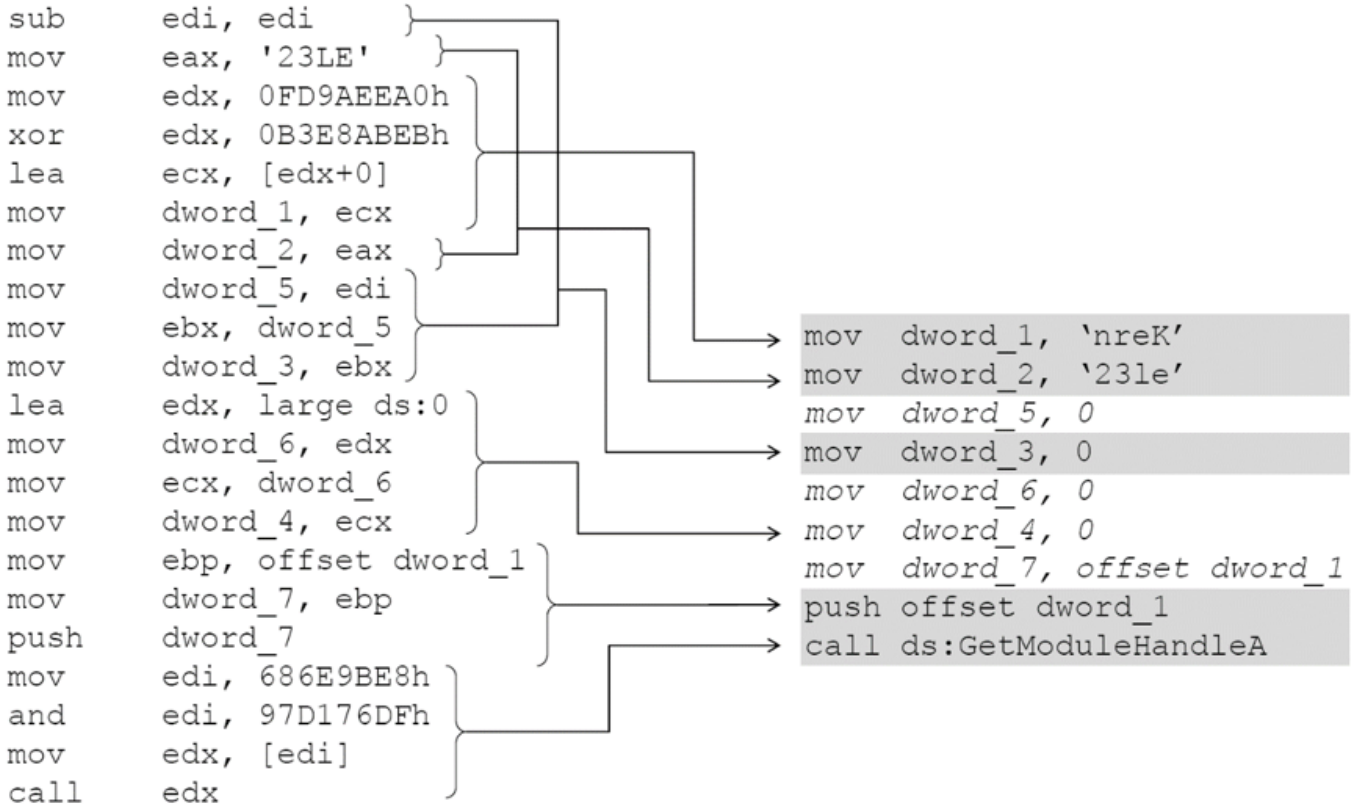
1. **Génération du GFC**
2. Normalisation des blocs
3. Optimisation inter-bloc



Détection statique

Identification du graphe de flot de contrôle

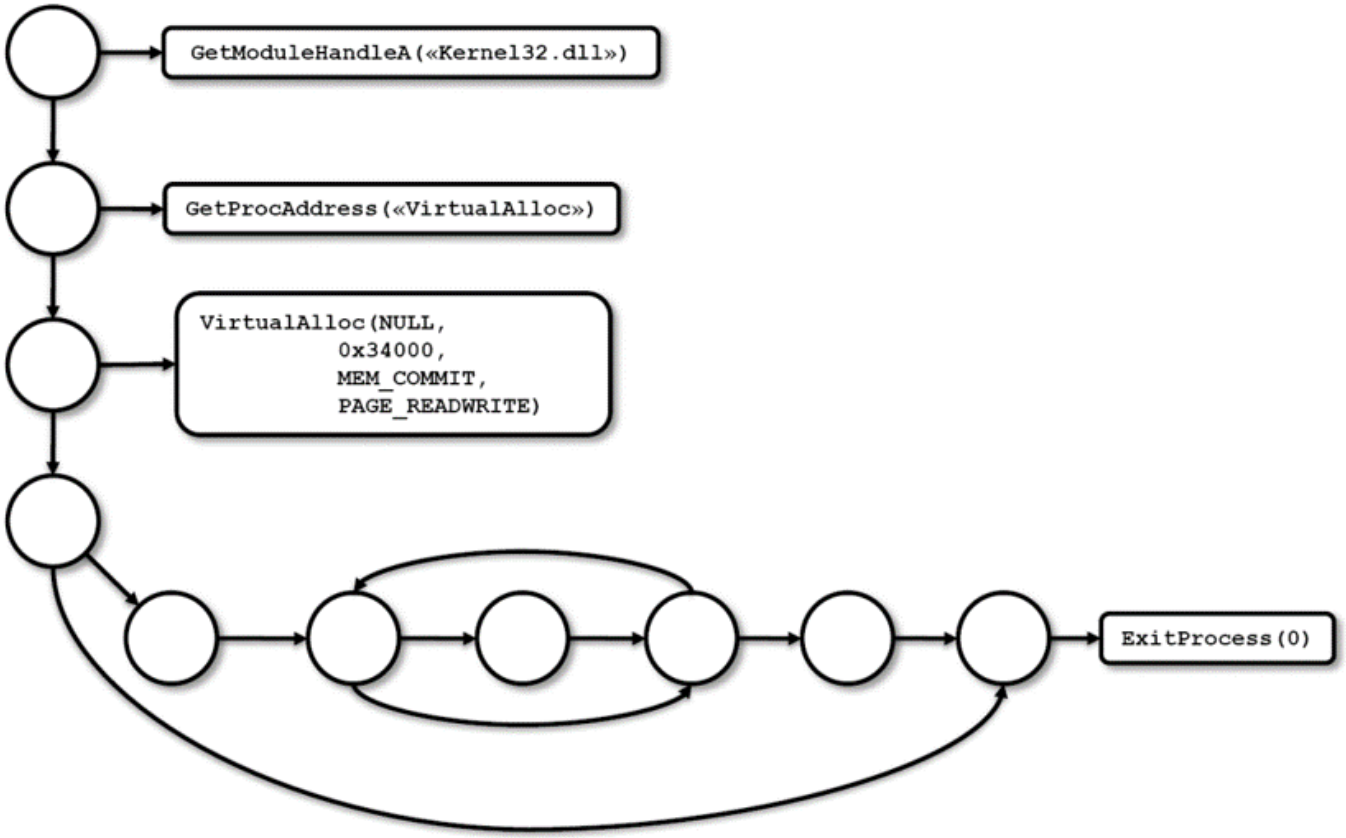
1. Génération du GFC
2. **Normalisation des blocs**
3. Optimisation inter-bloc



Détection statique

Identification du graphe de flot de contrôle

1. Génération du GFC
2. Normalisation des blocs
3. **Optimisation inter-bloc**



Détection dynamique

- Nécessite un environnement d'exécution
  - **Sûr** pour empêcher l'action malveillante sur l'hôte
  - **Transparent** du point de vue du malware pour ne pas altérer son comportement
- Une approche de détection
  - Traitement des traces d'exécution
  - Suivi des API invoquées
  - Dépendances entres API

Détection dynamique

Instrumentation dynamique de binaires (DBI)

Permet d'instrumenter un programme comme on le souhaite pour observer son comportement au niveau des instructions ASM.

Exemple

Pin, DynamoRIO, Valgrind, etc.

Attention

Les DBI sont détectables.

Exemple

L'outil eXait permet la détection de PIN de 15 façons différentes

Détection dynamique
Les machines virtuelles

- Emulation :
  - Simule un CPU (ex : Bochs)
  - Permet d'émuler un CPU sur une autre architecture (ex : ARM sur x86)
- Virtualisation :
  - Exécute directement les instructions non privilégiés par translation de binaire, ex : VMWare, QEMU, VirtualPC, VirtualBox

Attention

Les machines virtuelles sont détectables !

Détection dynamique
La virtualisation matérielle

- S'appuie sur un jeu d'instruction particulier apparu avec AMD-V et Intel-VT
- Permet de filtrer les instructions privilégiées, les entrées/sorties et les accès à la mémoire
  - Xen : hyperviseur open source pour linux
  - Ether : outils basé sur Xen dédié à l'analyse de malware
    - EtherUnpack : analyse pas à pas
    - EtherTrace : monitoring d'appels système

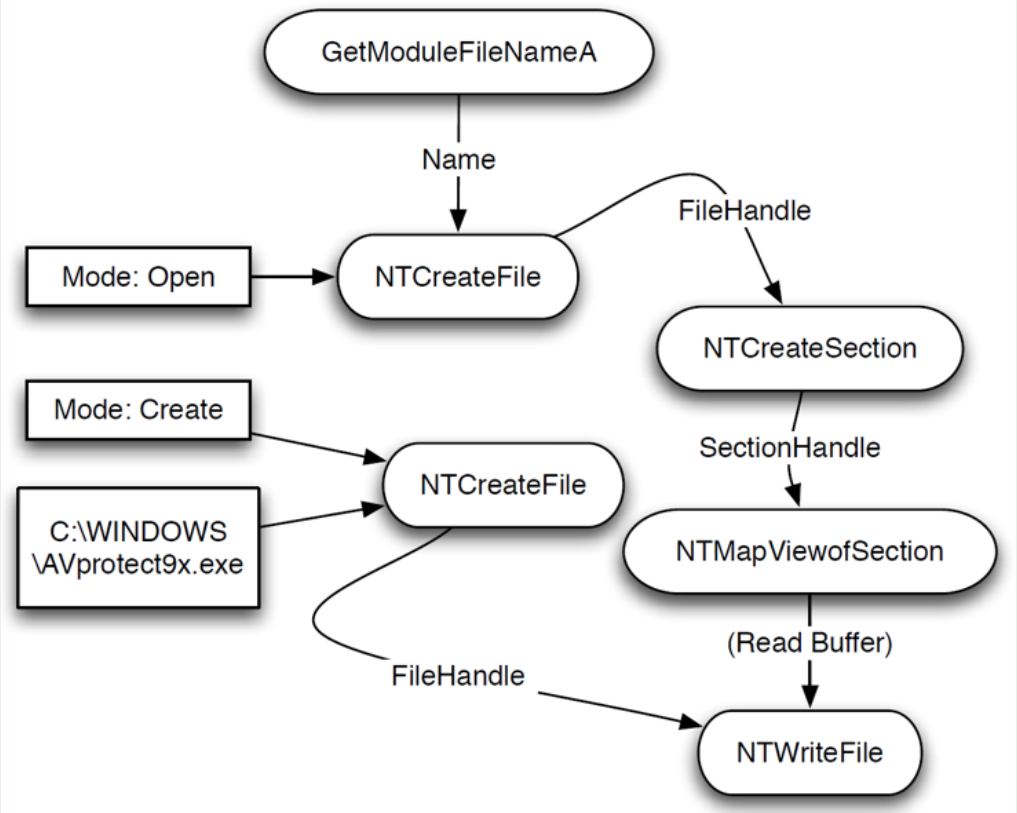
Détection dynamique

Détection comportementale

- Signature « comportementale »

Exemple

réplication, propagation, résidence, etc.



Détection dynamique
Difficultés lié à l'environnement

- Test de connectivité internet
- Droit utilisateur (Admin/User)
- Services ciblés (exploitation d'un programme spécifique)
- Données environnementales (présence d'un fichier particulier)
- Bombes logiques (action malveillante à une date donnée)
- Etc.

Plan

Analyse des malwares
Objectifs

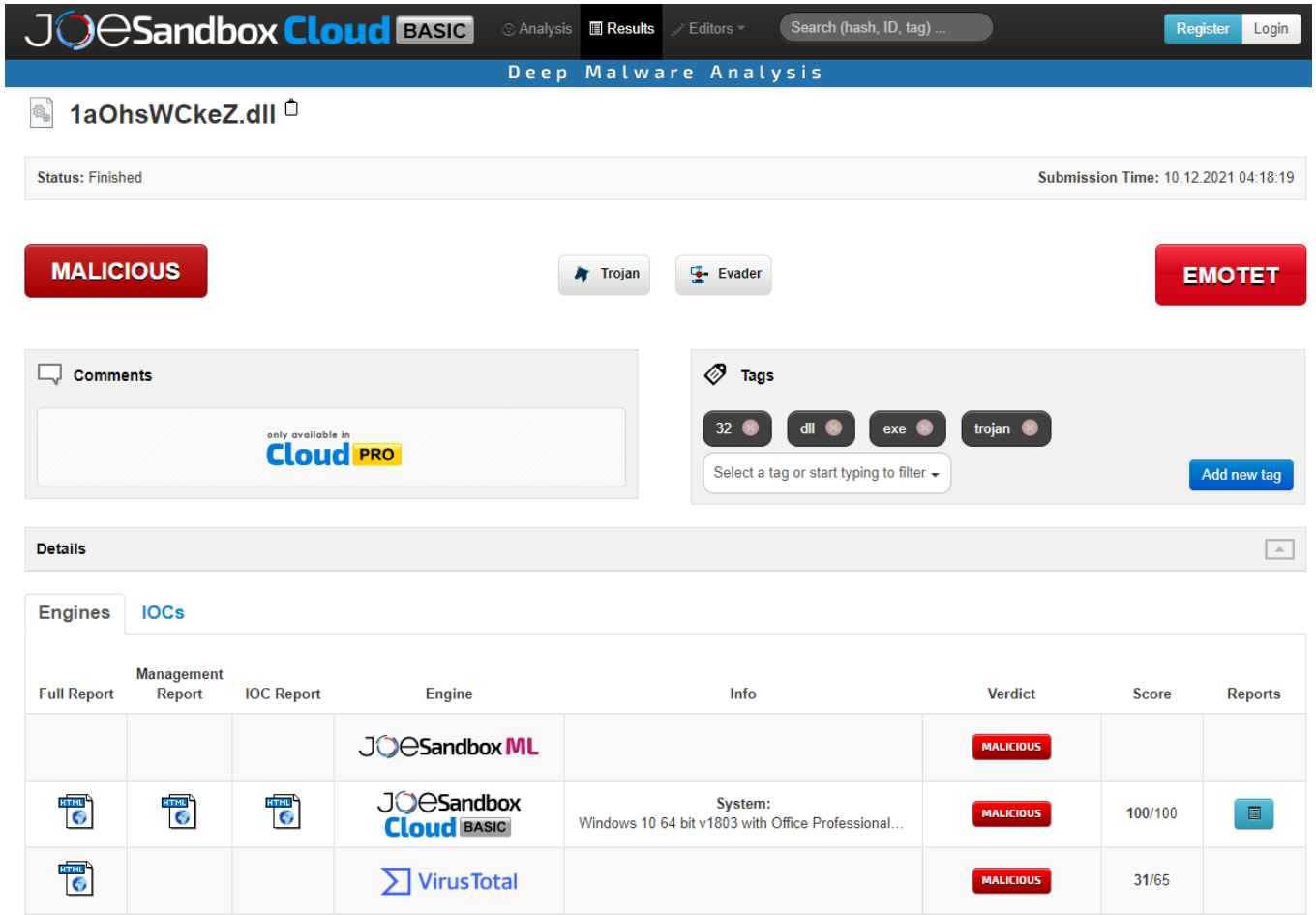
- Connaissance de la menace (Threat Intelligence)
- Déterminer des indicateurs de compromission (IOCs)
  - Hash de binaires
  - Adresses IP, noms de domaines, URLs
  - Artefacts système (noms d'objets)
- Créer des liens entre binaires
- **Protéger son système**

Analyse des malwares
Sandbox

- Cuckoo sandbox, opensource
  - Framework extensible (écrit en python)
  - VBox, KVM, VMWare, Xen
- JoeSecurity, propriétaire avec version basique (gratuite) et pro
  - Machines virtuelles et physiques
- Virus Total
  - Détection du binaire sur plein d'anti-virus
  - Une petite partie de Sandbox
  - Possibilité de télécharger les fichiers, mettre des règles de détections, etc. (payant)

Analyse des malwares

JoeSecurity - Emotet



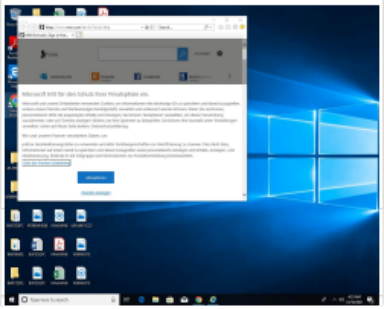
Analyse des malwares

JoeSecurity - Emotet

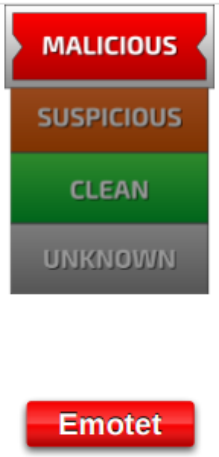
General Information

Sample Name:	1aOhsWCkeZ (renamed file extension from none to dll)
Analysis ID:	537561
MD5:	f5c531623cc0a40..
SHA1:	e82bfc7be0973dd.
SHA256:	ccd871ad115218..
Tags:	32 dll exe trojan
Infos:	

Most interesting Screenshot:



Detection



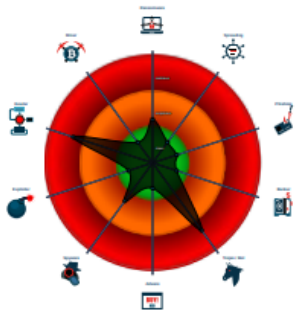
Emotet

Score:	100
Range:	0 - 100
Whitelisted:	false
Confidence:	100%

Signatures

- Found malware configuration
- Snort IDS alert for network traffic (e....
- Multi AV Scanner detection for subm...
- Yara detected Emotet
- System process connects to networ...
- Multi AV Scanner detection for doma...
- Machine Learning detection for samp...
- Sigma detected: Suspicious Call by ...
- C2 URLs / IPs found in malware con...
- Hides that the sample has been dow...
- Uses 32bit PE files
- Queries the volume information (nam...

Classification



Analyse des malwares

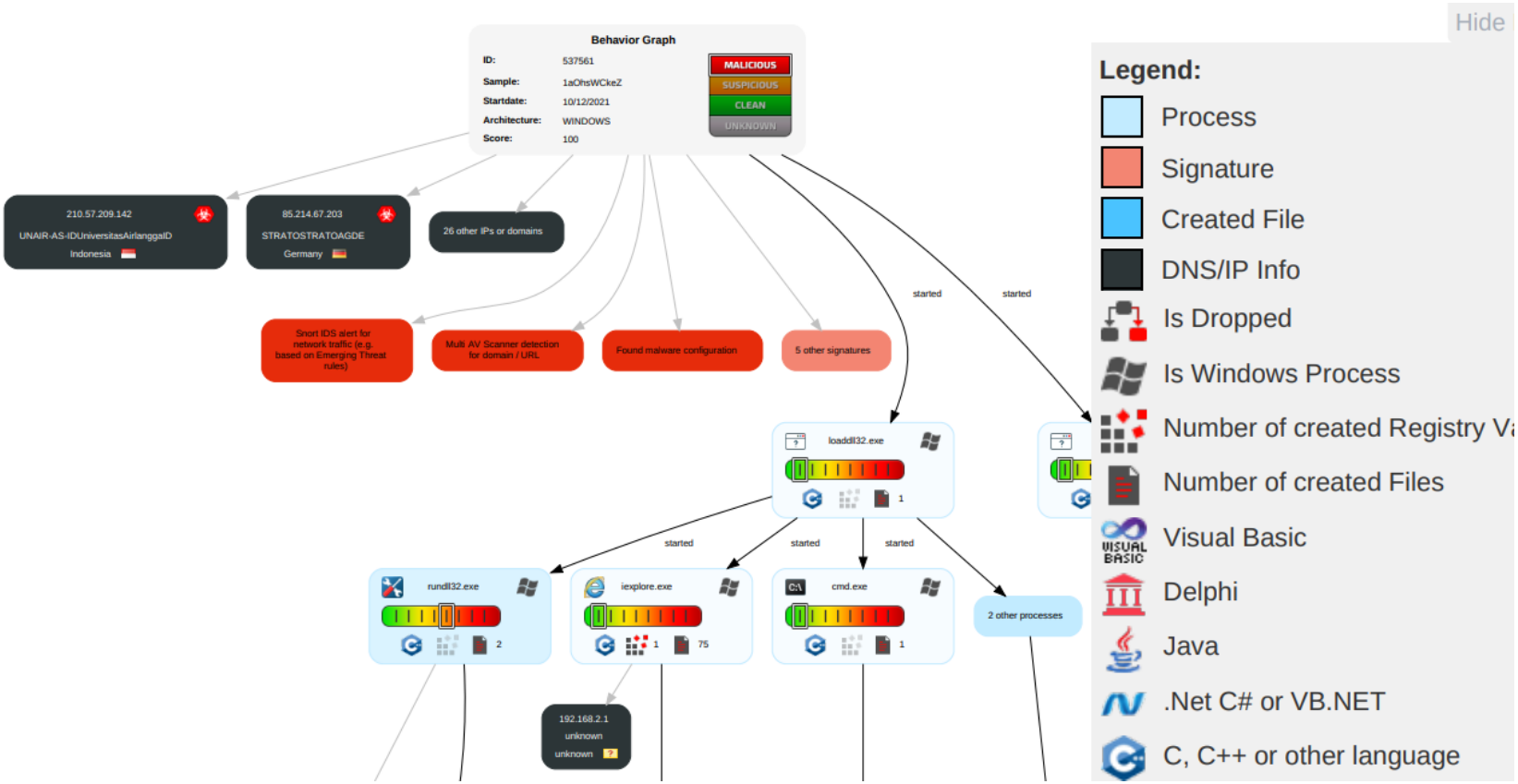
JoeSecurity - Emotet

Process Tree

- **System is w10x64**
- loaddll32.exe (PID: 4800 cmdline: loaddll32.exe "C:\Users\user\Desktop\1aOhsWCkeZ.dll" MD5: 72FCD8FB0ADC38ED9050569AD673650E)
  - cmd.exe (PID: 4788 cmdline: cmd.exe /C rundll32.exe "C:\Users\user\Desktop\1aOhsWCkeZ.dll",#1 MD5: F3BDBE3BB6F734E357235F4D5898582D)
    - rundll32.exe (PID: 408 cmdline: rundll32.exe "C:\Users\user\Desktop\1aOhsWCkeZ.dll",#1 MD5: D7CA562B0DB4F4DD0F03A89A1FDAD63D)
      - rundll32.exe (PID: 4116 cmdline: C:\Windows\SysWOW64\rundll32.exe "C:\Users\user\Desktop\1aOhsWCkeZ.dll",DllRegisterServer MD5: D7CA562B0DB4F4DD0F03A89A1FDAD63D)
  - regsvr32.exe (PID: 5292 cmdline: regsvr32.exe /s C:\Users\user\Desktop\1aOhsWCkeZ.dll MD5: 426E7499F6A7346F0410DEAD0805586B)
    - rundll32.exe (PID: 584 cmdline: C:\Windows\SysWOW64\rundll32.exe "C:\Users\user\Desktop\1aOhsWCkeZ.dll",DllRegisterServer MD5: D7CA562B0DB4F4DD0F03A89A1FDAD63D)
  - iexplore.exe (PID: 808 cmdline: C:\Program Files\Internet Explorer\iexplore.exe MD5: 6465CB92B25A7BC1DF8E01D8AC5E7596)
    - iexplore.exe (PID: 3160 cmdline: "C:\Program Files (x86)\Internet Explorer\IEXPLORE.EXE" SCODEF:808 CREDAT:17410 /prefetch:2 MD5: 071277CC2E3DF41EEEA8013E2AB58D5A)
  - rundll32.exe (PID: 6092 cmdline: rundll32.exe C:\Users\user\Desktop\1aOhsWCkeZ.dll,DllRegisterServer MD5: D7CA562B0DB4F4DD0F03A89A1FDAD63D)
    - rundll32.exe (PID: 2800 cmdline: C:\Windows\SysWOW64\rundll32.exe "C:\Windows\SysWOW64\Aqsuvlkorfglzxov\jizf.eis",ZCoB MD5: D7CA562B0DB4F4DD0F03A89A1FDAD63D)
      - rundll32.exe (PID: 1808 cmdline: C:\Windows\SysWOW64\rundll32.exe "C:\Windows\System32\Aqsuvlkorfglzxov\jizf.eis",DllRegisterServer MD5: D7CA562B0DB4F4DD0F03A89A1FDAD63D)
  - WerFault.exe (PID: 5220 cmdline: C:\Windows\SysWOW64\WerFault.exe -u -p 4800 -s 280 MD5: 9E2B8ACAD48ECCA55C0230D63623661B)
- svchost.exe (PID: 5272 cmdline: C:\Windows\System32\svchost.exe -k WerSvcGroup MD5: 32569E403279B3FD2EDB7EBD036273FA)
  - WerFault.exe (PID: 5876 cmdline: C:\Windows\SysWOW64\WerFault.exe -pss -s 476 -p 4800 -ip 4800 MD5: 9E2B8ACAD48ECCA55C0230D63623661B)
- **cleanup**

Analyse des malwares

JoeSecurity - Emotet

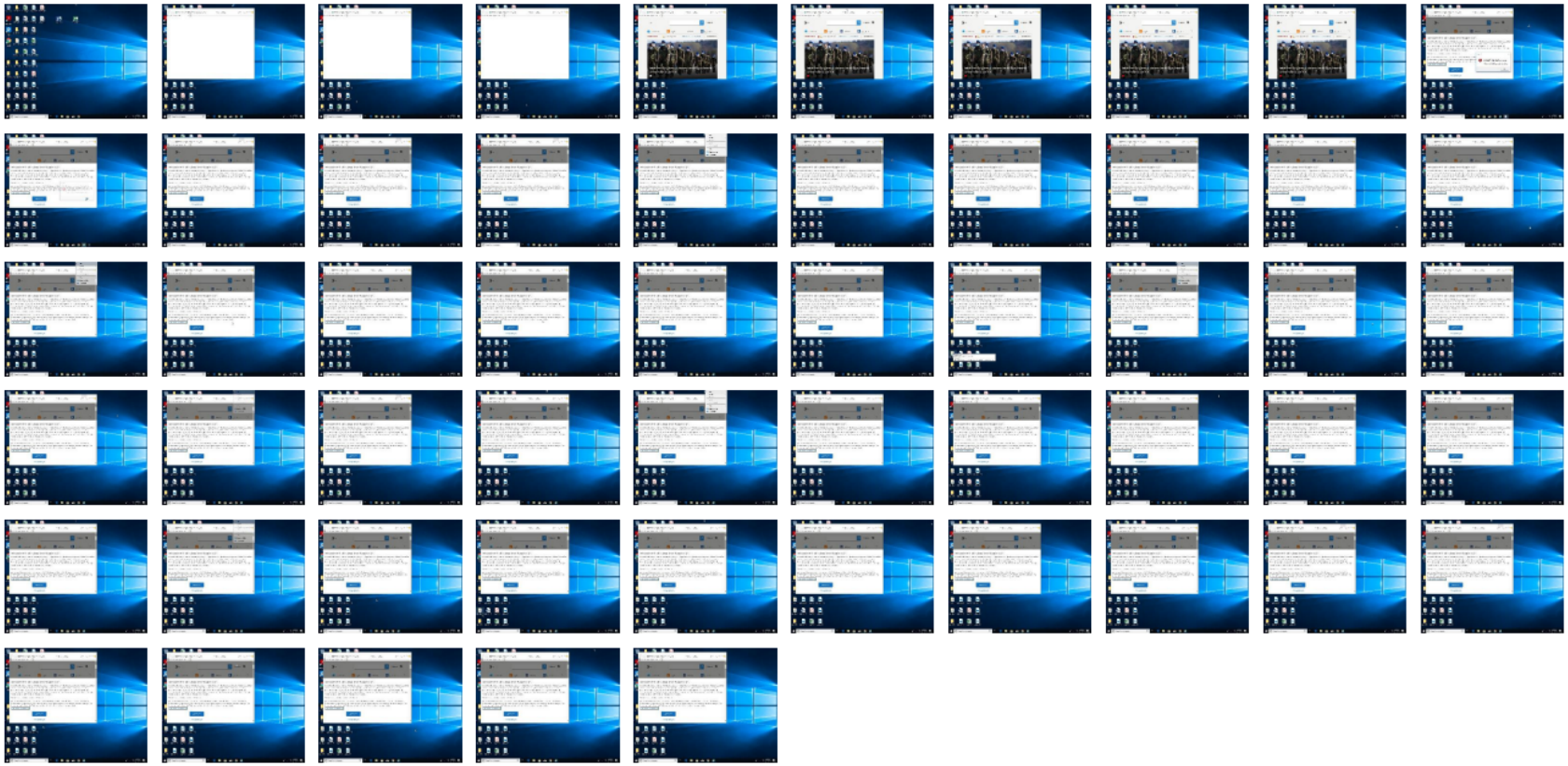


Analyse des malwares

JoeSecurity - Emotet

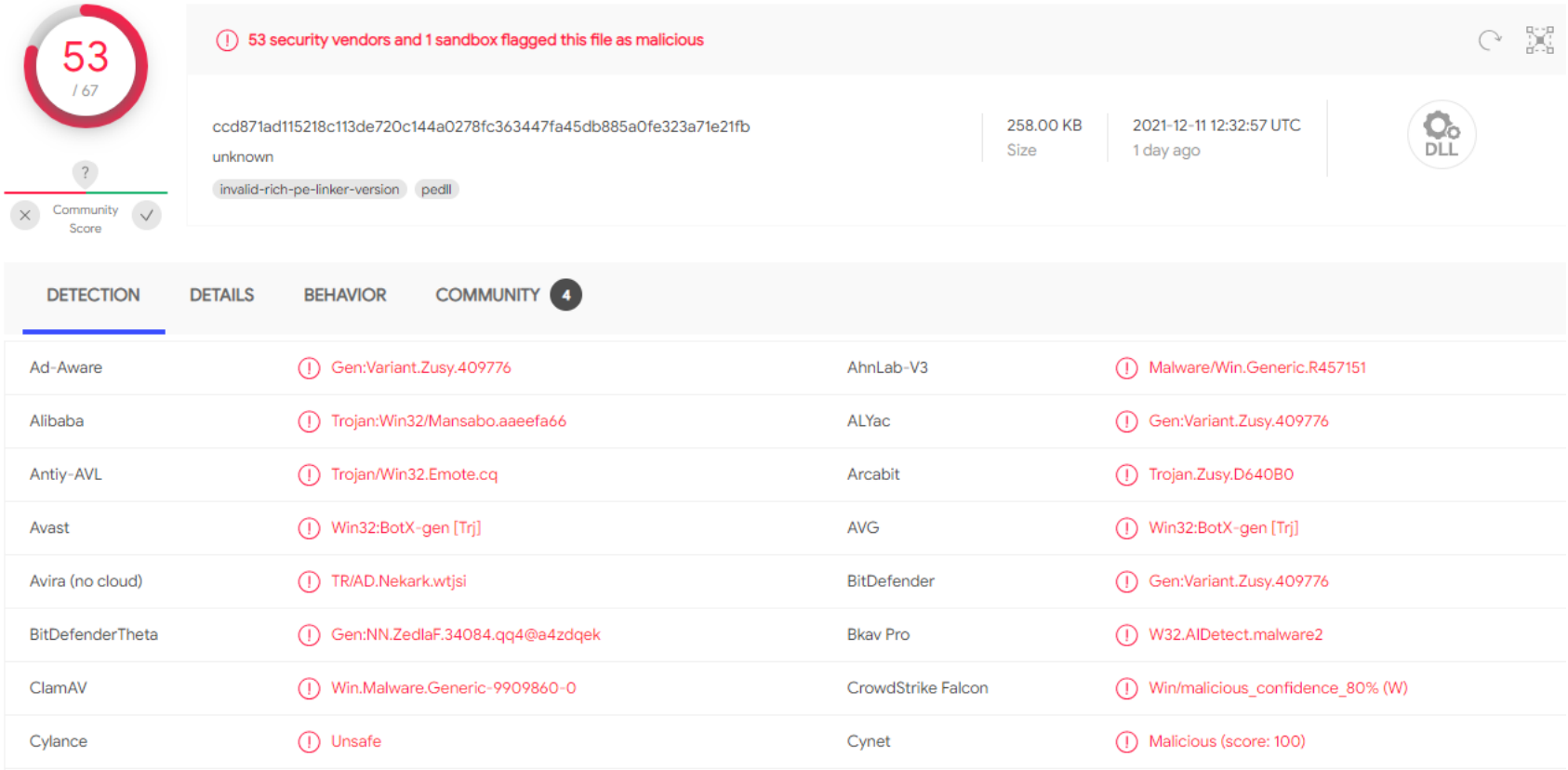
Thumbnails

This section contains all screenshots as thumbnails, including those not shown in the slideshow.



Analyse des malwares

Virus Total - Emotet



Analyse des malwares

Virus Total - Emotet

DETECTION DETAILS BEHAVIOR COMMUNITY 4

C2AE 2

Registry Actions

Registry Keys Set

– HKU\S-1-5-21-575823232-3065301323-1442773979-1000\Software\Microsoft\SystemCertificates\Root\Certificates\0174E68C97DDF1E0EEEA415EA336A163D2B61AFD\Blob

5C 00 00 00 01 00 00 00 04 00 00 00 00 10 00 00 04 00 00 00 01 00 00 00 10 00 00 00 0D BE 92 DE FF 7D 36 BB 48 C4 A6 B1 15 24 95 38 0F 00 00 00 01 00 00 00 20 00 00 00 53 FE B9 19 2E D4 80 F2 09 12 4A 2C 57 D7 E8 97 7A 2E 9F 39 46 1D BF 21 4D F1 12 CB 16 02 4F A2 14 00 00 00 01 00 00 00 14 00 00 00 78 B8 30 FD 63 AC 7B 89 4A 07 3B ED F6 8A 83 9C C3 52 02 65 19 00 00 00 01 00 00 00 10 00 00 00 B5 74 AF 30 C5 C1 BA 3A 69 A7 10 02 00 82 4D D0 03 00 00 00 01 00 00 00 14 00 00 00 01 74 E6 8C 97 DD F1 E0 EE EA 41 5E A3 36 A1 63 D2 B6 1A FD 20 00 00 00 01 00 00 00 F8 05 00 00 30 82 05 F4 30 82 03 DC A0 03 02 01 02 02 09 00 E0 EA 61 4C 28 56 32 64 30 0D 06 09 2A 86 48 86 F7 0D 01 01 0B 05 00 30 81 8E 31 0B 30 09 06 03 55 04 06 13 02 49 4C 31 0F 30 0D 06 03 55 04 08 0C 06 43 65 6E 74 65 72 31 0C 30 0A 06 03 55 04 07 0C 03 4C 6F 64 31 10 30 0E 06 03 55 04 0A 0C 07 47 6F 50 72 6F 78 79 31 10 30 0E 06 03 55 04 0B 0C 07 47 6F 50 72 6F 78 79 31 1A 30 18 06 03 55 04 03 0C 11 67 6F 70 72 6F 78 79 2E 67 69 74 68 75 62 2E 69 6

\+ HKU\S-1-5-21-575823232-3065301323-1442773979-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyServer

\+ HKU\S-1-5-21-575823232-3065301323-1442773979-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyEnable

\+ HKU\S-1-5-21-575823232-3065301323-1442773979-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections\SavedLegacySettings

Process And Service Actions

Shell Commands

%SANDBOX_DLL_LOADER_X86% %SAMPLEPATH% %WORKDIR% 483

%windir%\SysWOW64\rundll32.exe "%SAMPLEPATH%",DllRegisterServer

%windir%\SysWOW64\rundll32.exe "%windir%\SysWOW64\Vdfdnxkwjk\vqyzisgrzupqzg.fdp",UODcSlpGDqmUJ

Analyse des malwares
Virus Total - Emotet

Processes Terminated

%windir%\System32\svchost.exe -k WerSvcGroup

wmiadap.exe /F /T /R

%SANDBOX_DLL_LOADER_X86% %SAMPLEPATH% %WORKDIR% 483

%windir%\SysWOW64\rundll32.exe "%SAMPLEPATH%",DllRegisterServer

%windir%\SysWOW64\rundll32.exe "%windir%\SysWOW64\Vdfdnxkwjk\vqyzisgrzupqzg.fdp",UODcSIpGDqmUJ

Processes Tree

↳ 2232 - %windir%\System32\svchost.exe -k WerSvcGroup

↳ 3004 - wmiadap.exe /F /T /R

↳ 2644 - %SANDBOX_DLL_LOADER_X86% %SAMPLEPATH% %WORKDIR% 483

↳ 2724 - %windir%\SysWOW64\rundll32.exe "%SAMPLEPATH%",DllRegisterServer

↳ 2764 - %windir%\SysWOW64\rundll32.exe "%windir%\SysWOW64\Vdfdnxkwjk\vqyzisgrzupqzg.fdp",UODcSIpGDqmUJ

↳ 2784 - %windir%\SysWOW64\rundll32.exe "%windir%\SysWOW64\Vdfdnxkwjk\vqyzisgrzupqzg.fdp",DllRegisterServer

↳ 3052 - %windir%\system32\wbem\wmiprvse.exe

Analyse des malwares
Correlation de binaires

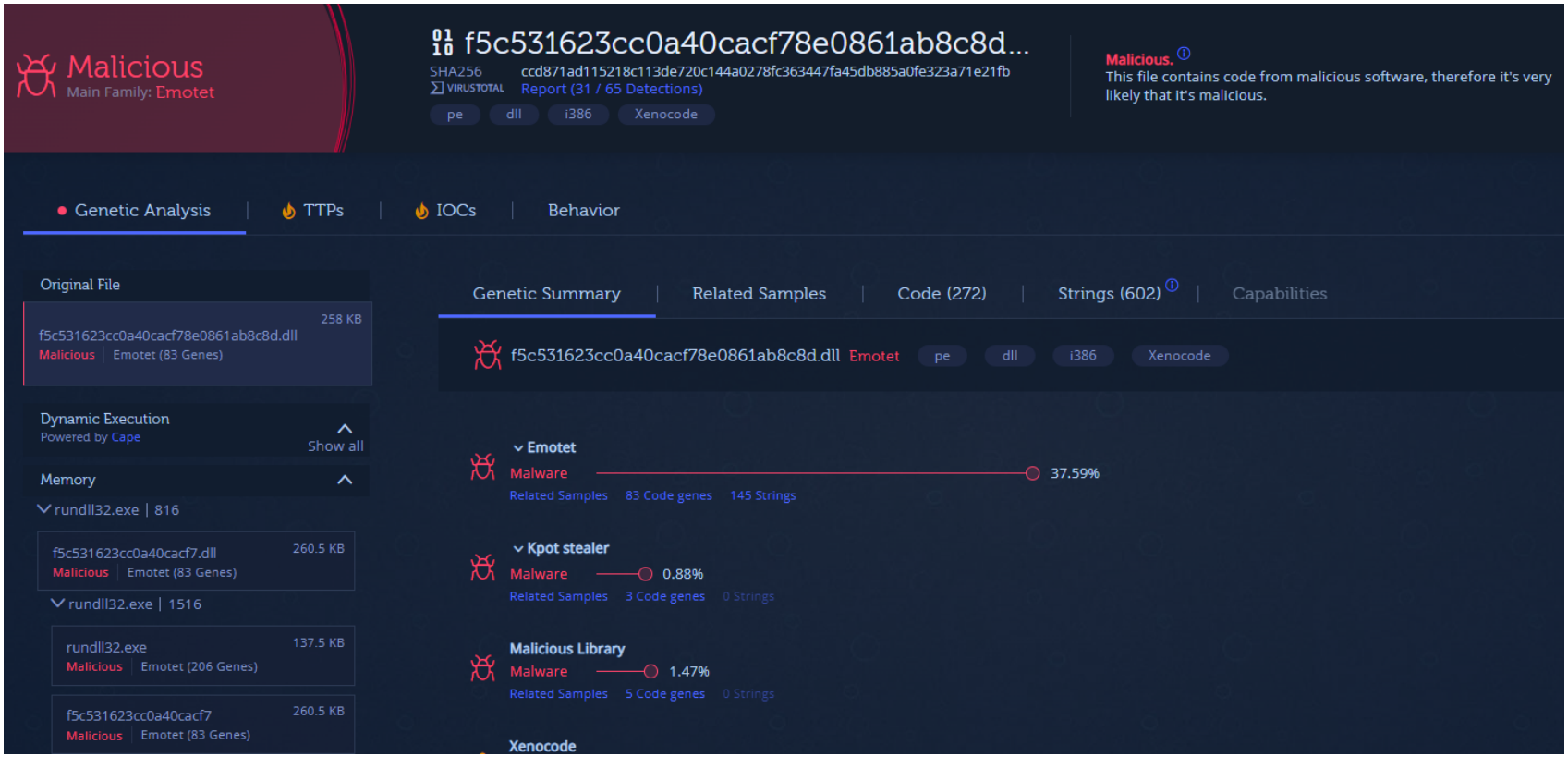
- Glimps
  - Société française (Rennaises)
  - Correlation de binaire basé sur l'IA
  - Startup qui fonctionne très bien
- Intezer
  - Société Israelienne de correlation de binaire
  - Correlation basé sur des "gênes" (abstration de basique bloc)
  - Version gratuite : 50 fichiers par mois
  - Plugin IDA

Attention

Pas de fiabilité à 100%

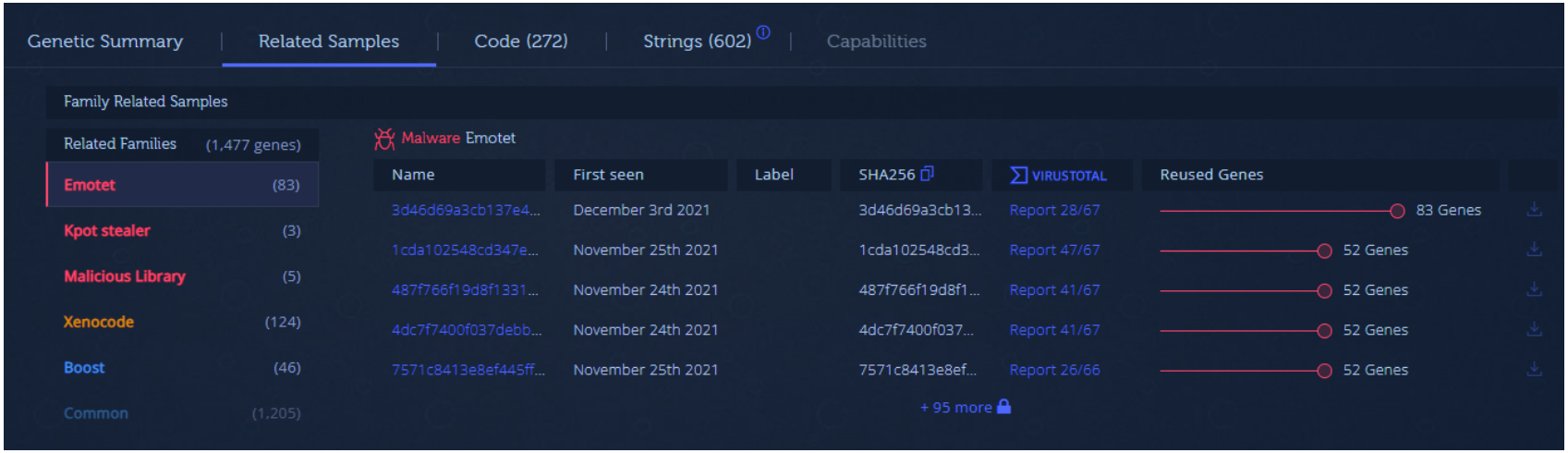
Analyse des malwares

Intezer - Emotet



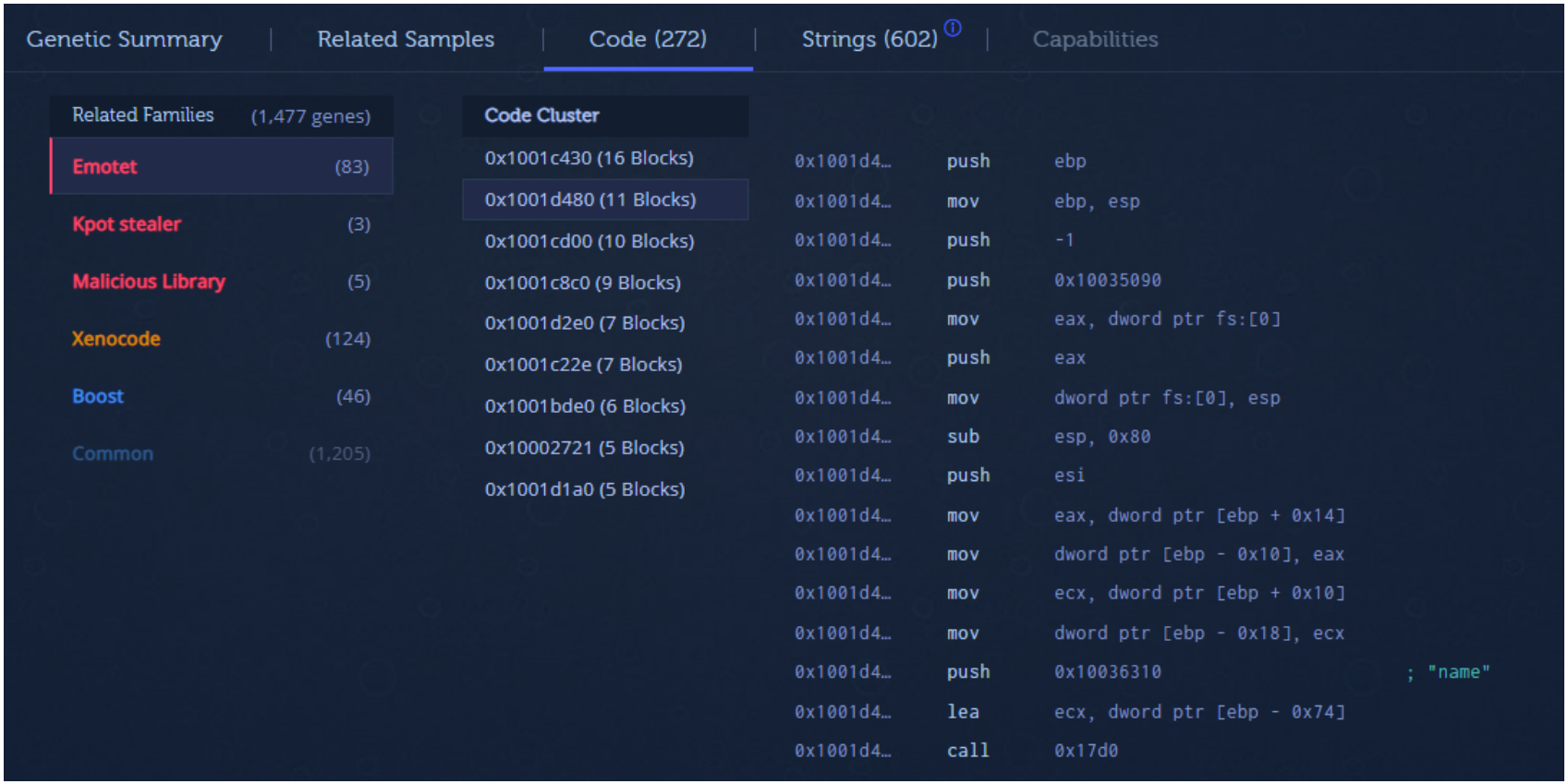
Analyse des malwares

Intezer - Emotet



Analyse des malwares

Intezer - Emotet



Plan

Chevaux de Troie

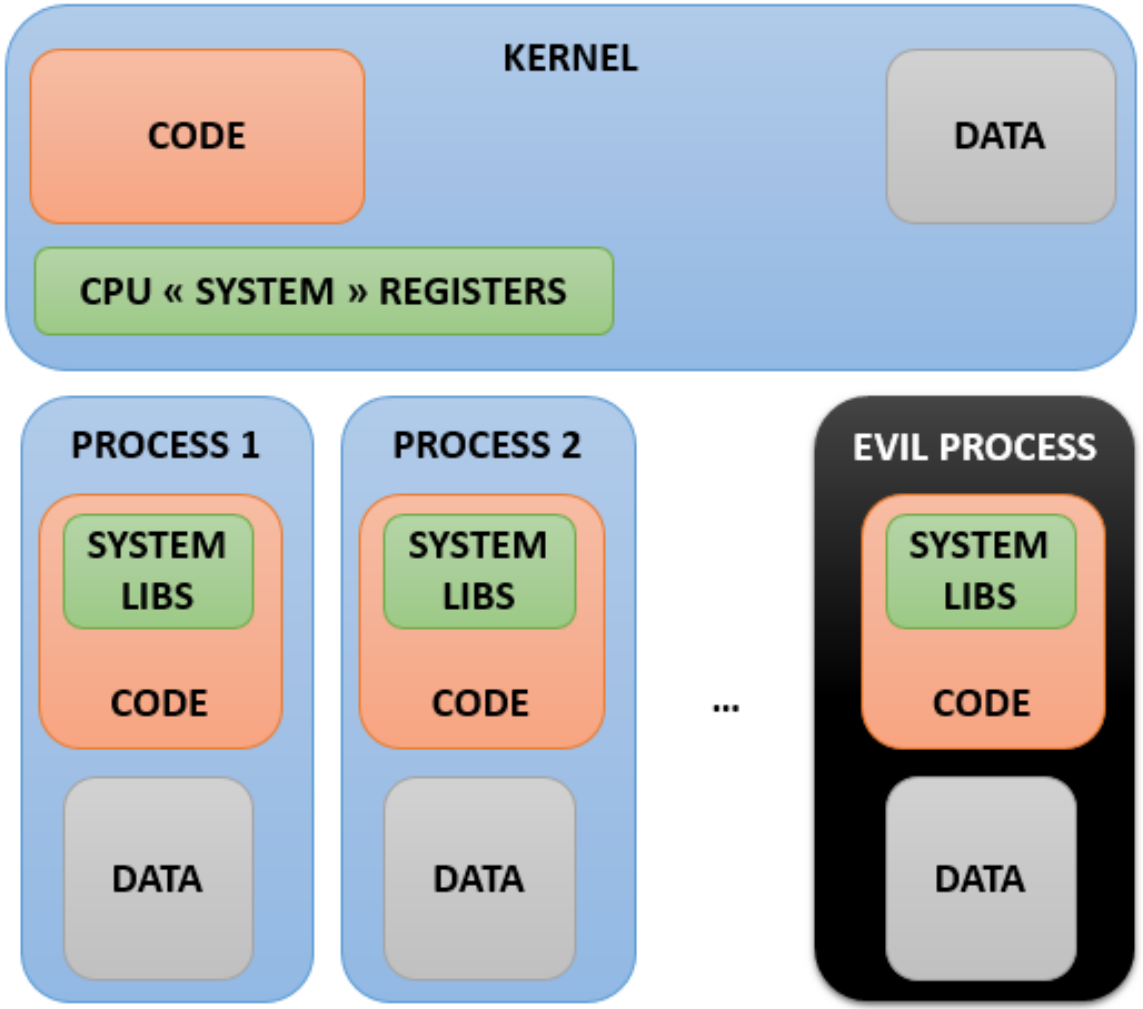
Definition

Malware d'apparence légitime, conçu dans le but d'exécuter des actions malveillantes à l'insu de l'utilisateur

Fonctionnement

Il s'agit d'un programme simple sans technique spécifique

Chevaux de Troie



Vers

Definition

Malware auto-reproducteur se propageant (au moyen du réseau)

Fonctionnement

Différentes techniques assurant la reproduction :

- Pièce jointe d'un mail (« mailer »)
- Exploitation d'un vulnérabilité sur une cible
- Etc.

Prévention

Firewall

Limiter les services ciblés (historiquement SMB, RPC, etc.)

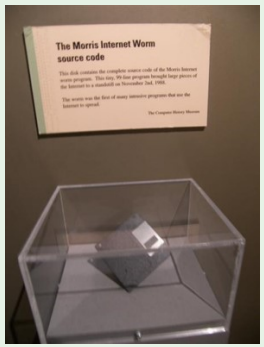
Si le vers ne mute pas, détection du code envoyé qui est le même que l'original dans le cas d'un « mailer ».

Vers
Exemple célèbre

Exemple

Morris (1988)

- 1er ver connu conçu pour se propager uniquement
- Exploitation de :
  - Mode DEBUG de sendmail
  - Buffer overflow dans finger
  - Mots de passe utilisateur faibles
- Création du premier CERT (Computer Emergency Response Team)
- Condamnation de Robert Morris à 400h de travaux d'intérêts publics et plus de 10 000$ d'amende



Vers
Exemple célèbre

Exemple

I love You (2000)

- Pièce jointe d'un mail Love-Letter-for-you.txt.vbs
- Actions :
  - Modifie la page de démarrage d'IE pour télécharger un cheval de Troie (WIN-BUGSFIX.exe)
  - Modifie le registre pour s'exécuter automatiquement
  - Remplace certains fichier par une copie de lui-même
  - Propagation via les contacts outlook et le client IRC mIRC
- 3,1 millions de machines infectés
- Auteur présumé : Onel Guzman (philippin 24ans), relâché sur le champs car aucune loi contre le hacking à l'époque

Vers

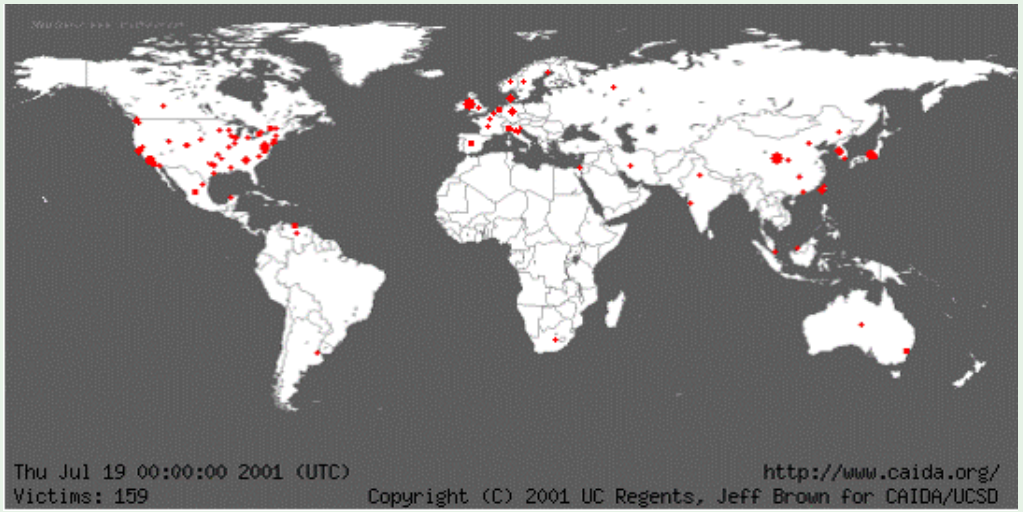
Exemple célèbre - I love You

Vers
Exemple célèbre

Exemple

Code Red (2001)

- Exploite une vulnérabilité dans les servers IIS
- Lancement d'une attaque de type DoS
- 359 000 machines infectées en 24h



Vers
Exemple célèbre

Exemple

Conficker (novembre 2008)

- Exploitation de la faille MS08-067 (RPC) permettant d'exécuter de code à distance en temps que SYSTEM
- Plusieurs millions de machines infectées
- Utilisation d'un AutoRun pour propagation via clé USB
- Actions :
  - Désactivations de Windows Update, centre de sécurité et Windows Defender
  - Connexion à des serveurs de commandes et de contrôle (C&C) :
    - Collecte d'information
    - Téléchargement de charges supplémentaires
    - Etc.

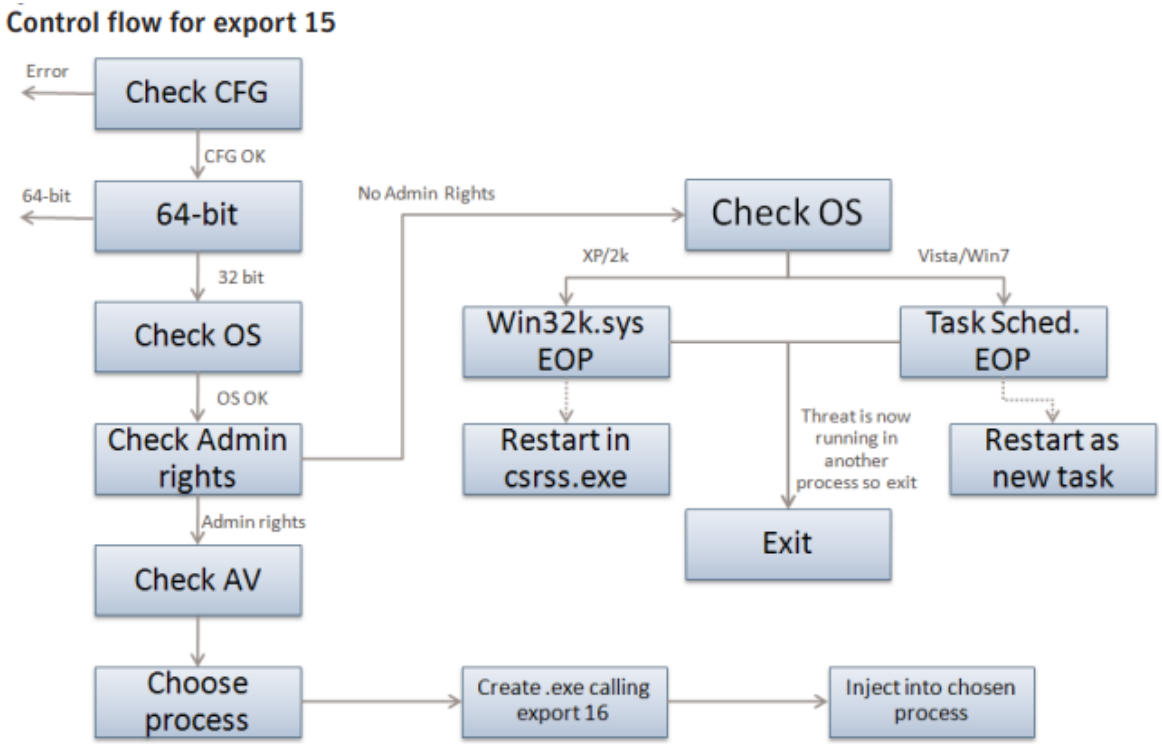
Vers
Exemple célèbre

Exemple

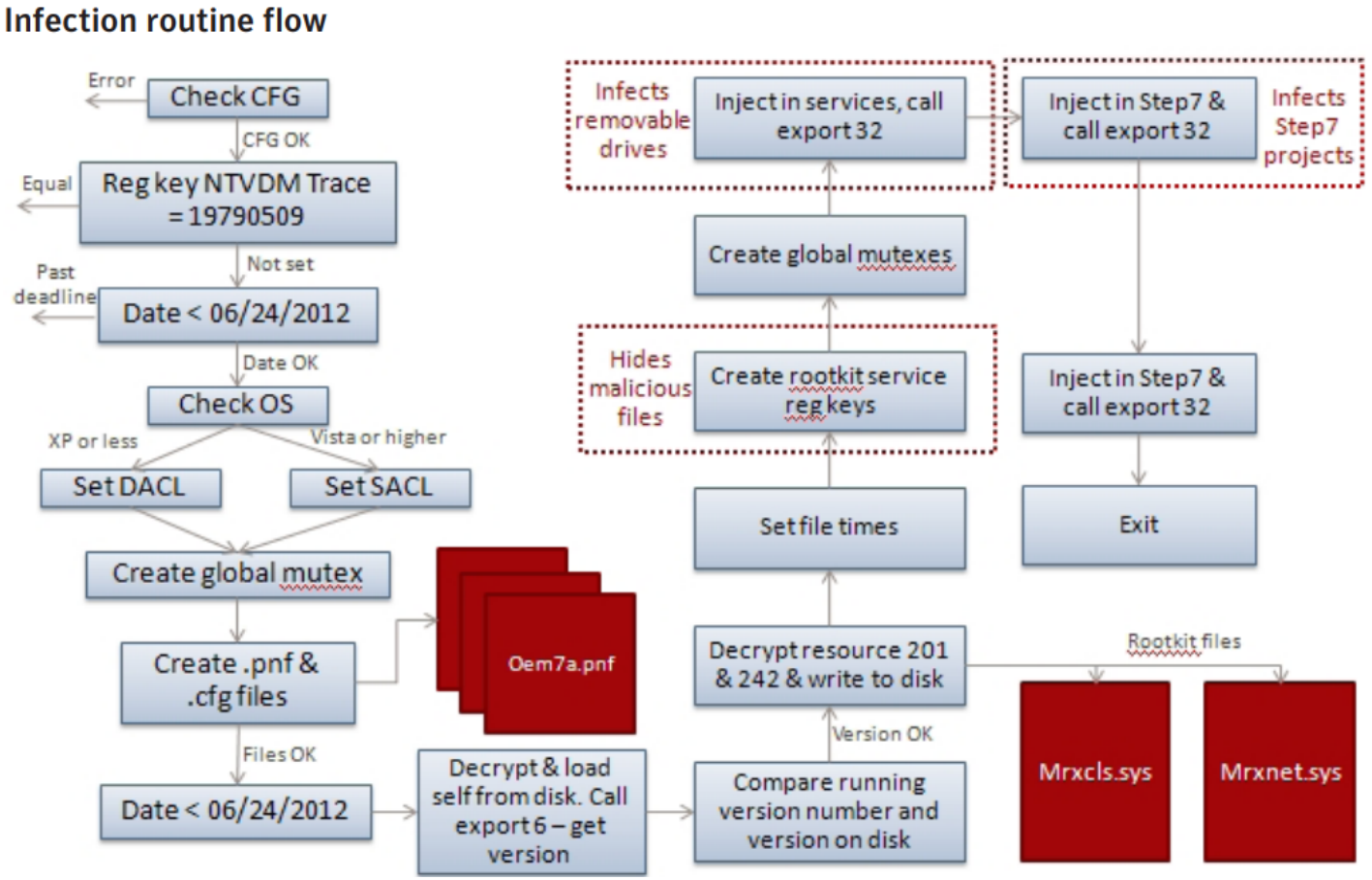
Stuxnet (juin 2010) - NSA & Israël

- 4 0-days utilisés
  - MS08-067 - exécution de code à distance en temps que SYSTEM
  - MS10-061 - exécution de code à distance dans le spouleur d'impression
  - MS10-046 - vulnérabilité LNK pour se propager (avant : AutoRun)
  - MS10-073 - Win32k.sys élévation de priviléges
- Cible spécifiquement les systèmes SCADA
- 45 000 machines infectées
- Utilisation de certificats volés
- Présence d'une date « de destruction » fixée au 24 juin 2012
- Actions :
  - Modification de la vitesse de rotation des centrifugeuses pour les détruires
  - Remonter de fausses informations sur la surpervision
  - Evasion d'anti-virus (en fonction de 9 AV et des versions)
  - Copie sur les disques amovibles
  - Mécanisme P2P de mise à jour (réseau cloisonné)
  - Cache ses actions via un cheval de troie

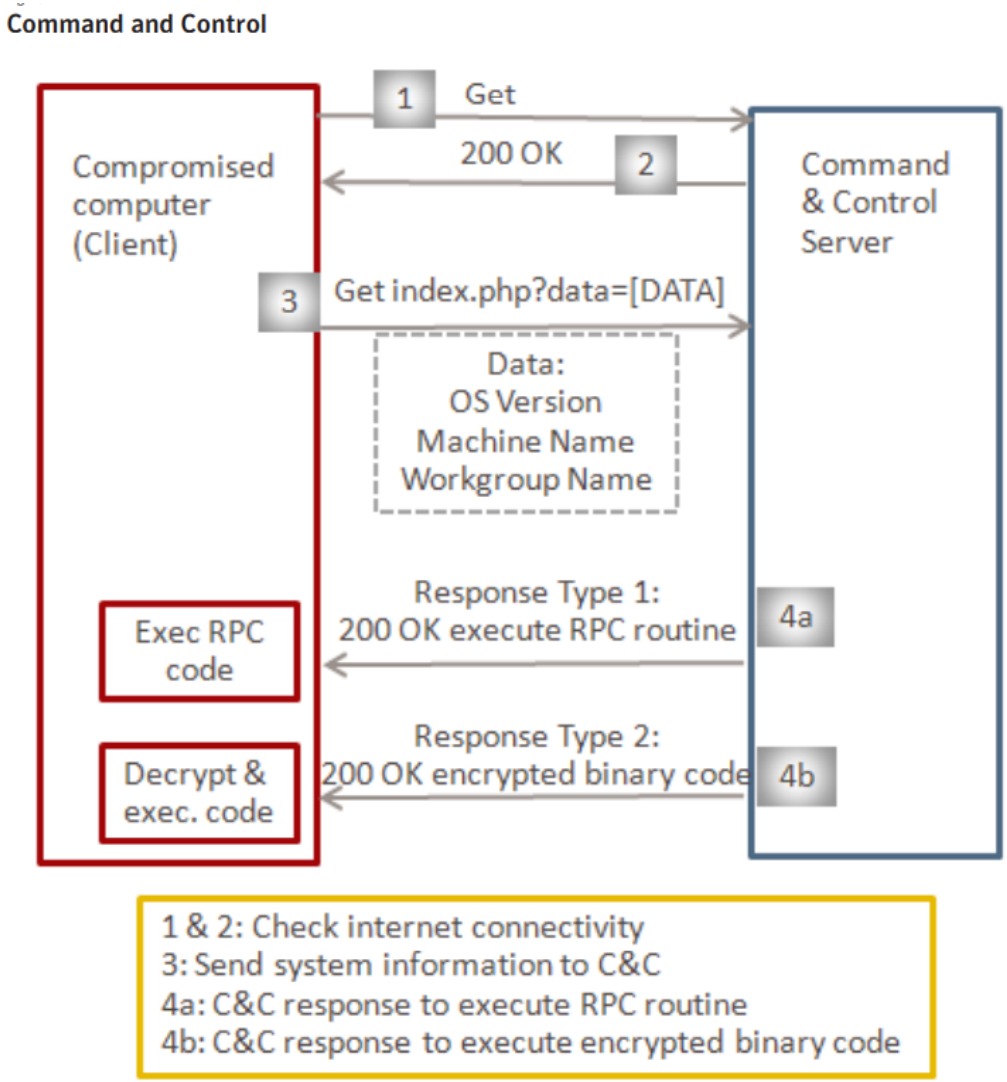
Vers
Exemple célèbre - Stuxnet



Vers
Exemple célèbre - Stuxnet



Vers
Exemple célèbre - Stuxnet



Virus

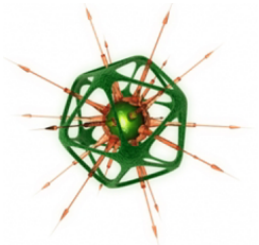
Definition

Malware auto-reproducteur se propageant en infectant un programme hôte

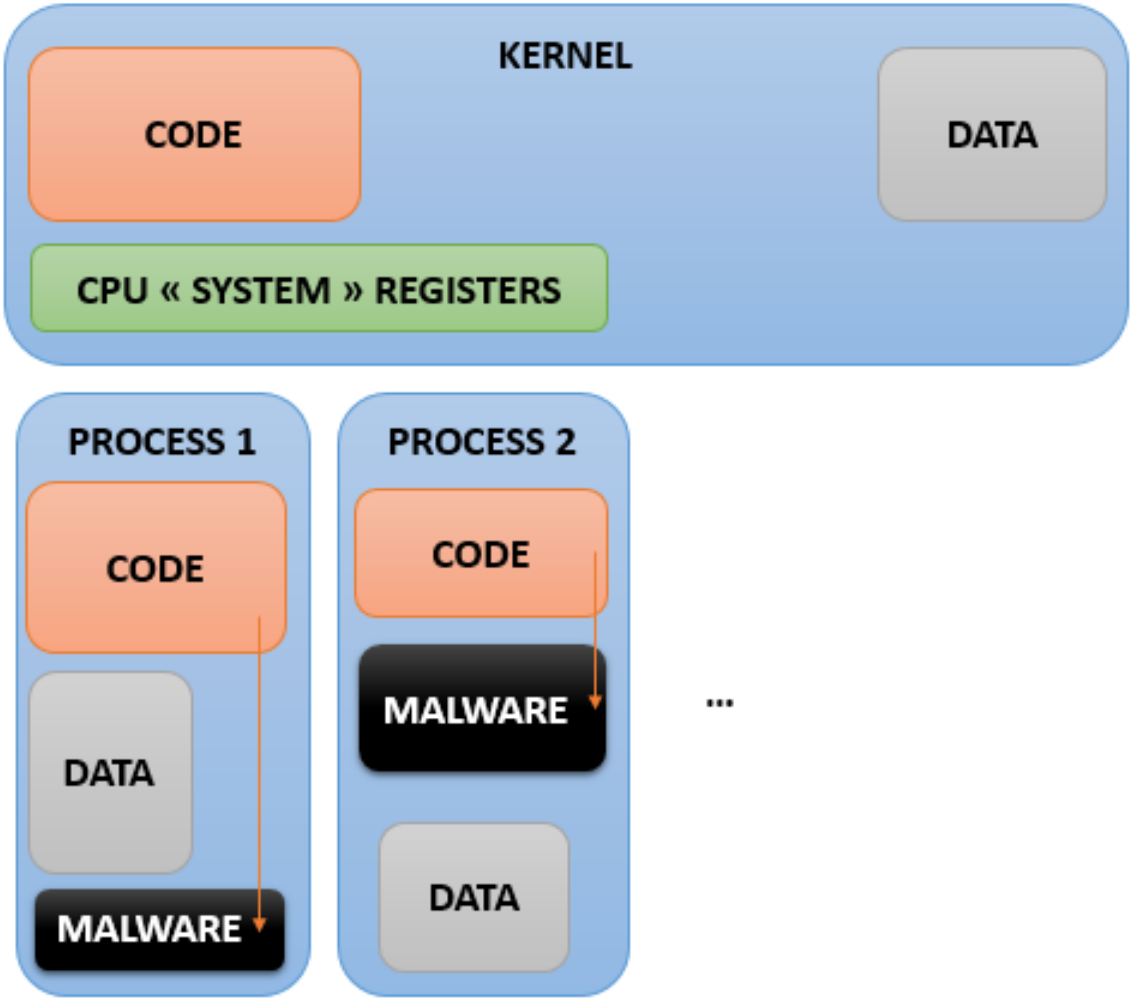
Fonctionnement

Modification du flow de control de l'hôte pour s'exécuter :

- Point d'entrée du programme (Entry Point Obfuscation)
- Fin du programme
- À un endroit précis dont le virus est sûr de l'exécution (ex : modification de l'IAT)



Virus


KERNEL
CODE
DATA
CPU « SYSTEM » REGISTERS
PROCESS 1
CODE
DATA
MALWARE
PROCESS 2
CODE
MALWARE
DATA
...

Virus
Exemple célèbre

Exemple

Tchernobyl (ou CIH 1997)

- Virus particulièrement destructeur
  - Écrasement du 1^{er} Mo de chaque disque dur (et donc du MBR)
  - Tentative de flashage du BIOS
- Auteur du virus arrêté le 29 avril 1999 et relâché sur le champ car aucune plainte déposée !
- Cible Windows 95, 98 et ME

Virus
Exemple célèbre

Exemple

Win32.MetaPHOR (2002)

- Virus métamorphe
- 14000 lignes d'assembleur x86
- 90% du code correspond au moteur de métamorphisme
- Exécution lorsque l'hôte se termine
- Particulièrement difficile à détecter

Virus
Techniques de prévention

- Détection du test de surinfection
- Vérification d'intégrité
- Empêcher la modification de binaires
- Politique de signatures d'exécutables

Rootkit

Definition

Malware permettant de modifier le comportement d'un système d'exploitation afin de cacher du code

Fonctionnement

Différents types de rootkits (Joanna Rutkowska)

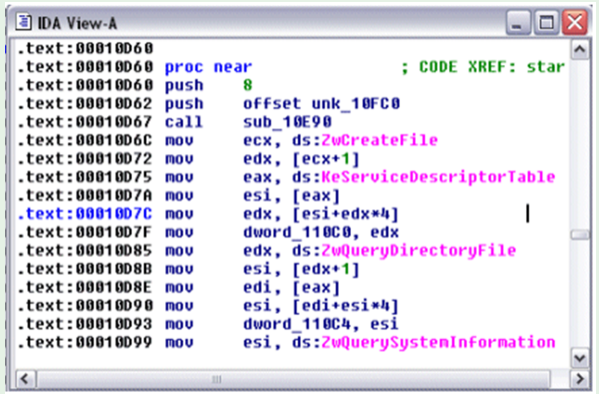
- Type 0 : interactions « documentées » avec le système
- Type 1 : modification des constantes du système
- Type 2 : modification des variables du système
- Type 3 : virtualisation du système

Rootkit
Exemple célèbre

Exemple

Rootkit SONY (2005)

- Rootkit dans des logiciels de gestion de DRM de SONY pour limiter la copie
- Hooks de la SSDT
  - ZwCreateFile
  - ZwQueryDirectoryFile
  - ZwQuerySystemInformation
  - Etc.
- Masque les fichiers, répertoires, clés de registre, processus commençant par « sys »



Rootkit
Protections

- Niveau kernel : patch guard (windows 64 bits)
  - IDT, GDT, SSDT, images systèmes (ntoskrnl.exe, ndis.sys, hal, etc.), MSRs, etc.
  - Fonctionne par contrôle d'intégrité
- Rootkit de niveau 3 très difficiles à détecter mais aucun connu actuellement
- Différents outils pour le userland :
  - Contrôle d'intégrité de fichiers
  - Corrélation TID/PID en cas de DKOM
  - RAW accès au système de fichier
  - Etc.

Bootkit

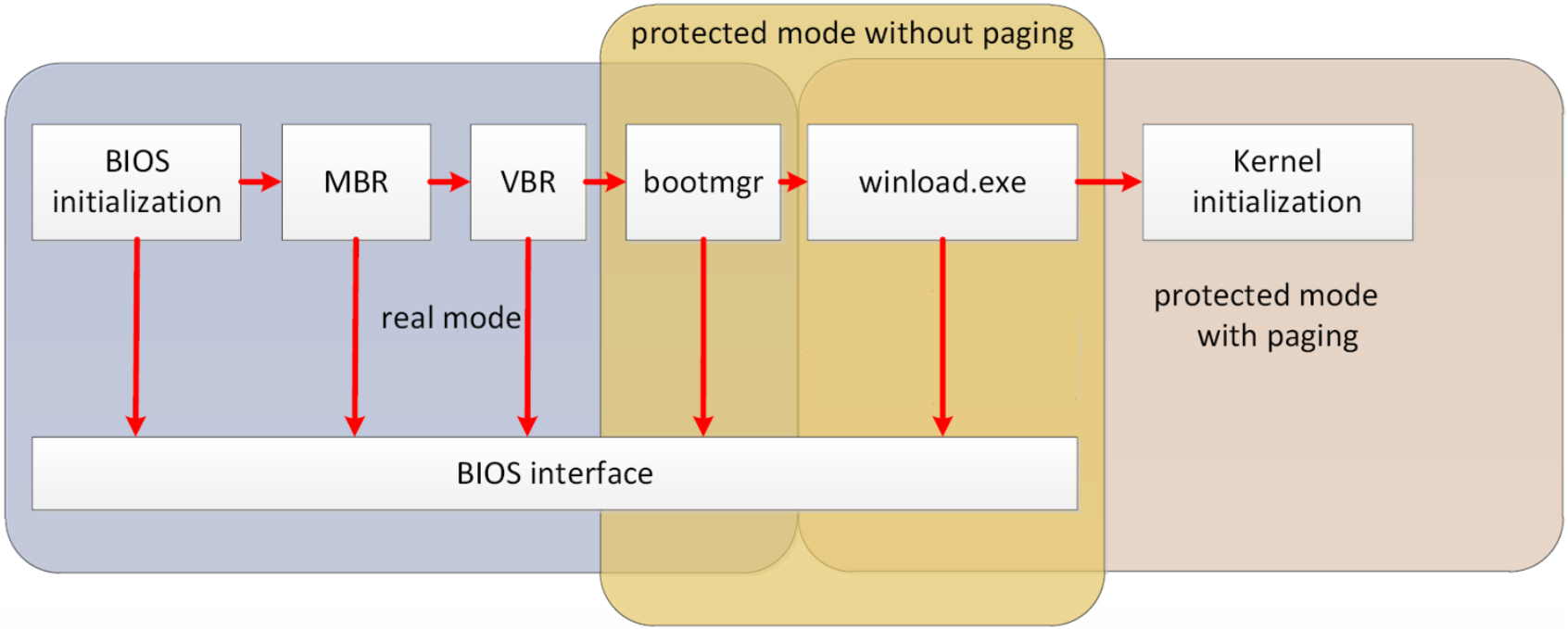
Definition

Malware affectant le processus de boot (amorce) d'une machine afin de rester actif au sein du système d'exploitation

Fonctionnement

Hook d'un service du firmware (BIOS/UEFI) pour prendre le contrôle dans le flow d'exécution du boot (MBR, VBR, etc.)

Bootkit
Fonctionnement



Bootkit
Exemples

Exemple

- Académique
  - Stoned `http://www.stoned-vienna.com`
  - Vbootkit `http://www.nvlabs.in`
  - DreamBoot `https://github.com/quarkslab/dreamboot`
- Réels
  - Win64/Olmarik (TDL4)
  - Win64/Olmasco (MaxSS)
  - Win64/Rovnix

Bootkits
Exemple célèbre

Exemple

Lojax (2008)

- Premier bootkit UEFI affectant un firmware
- Utilisation d'un driver signé (RwDrv.sys) de l'utilitaire RWEverything
- Réécriture de la mémoire flash SPI ⇒ ajout d'un driver UEFI
  - Modification du système de fichier NTFS pour y écrire deux payloads (autoche.exe et rpcnet.exe)

Bootkits
Protection

Secure Boot

Chaîne de confiance (vérification cryptographique) du matériel (composant TPM) jusqu'au système d'exploitation

Nécessite une coopération entre :

- Le matériel (TPM)
- Le firmware (BIOS/UEFI)
- L'OS

Utilisés sur les OS récents, smartphones compris.

Botnets

Definition

Réseau de machines compromises (bot) permettant de mener des actions distribuées (spam, DOS, brute force, etc.)

Fonctionnement

Communication vers un serveur de Commandes et de Contrôle (C&C)

- Canaux IRC (historique)
- P2P pour ne pas être centralisé
- HTTP
- Etc.

Botnets
Exemple célèbre

Exemple

Rustock (2006)

- 150 000 à 2 400 000 machines infectées
- 30 milliards de Spams par jour
- 250 000$ promis par Microsoft pour l'identification des auteurs

Botnets
Exemple célèbre

Exemple

Waledac (2006)

- 1,5 milliards de Spams par jour (1% du spam mondial)
- 277 noms de domaines pour l'envoi de spam
- Communication via P2P
- 70 000 à 90 000 machines infectées

Ransomware

Definition

Malware bloquant l'ordinateur de la victime tant qu'une rançon n'a pas été payée

Fonctionnement

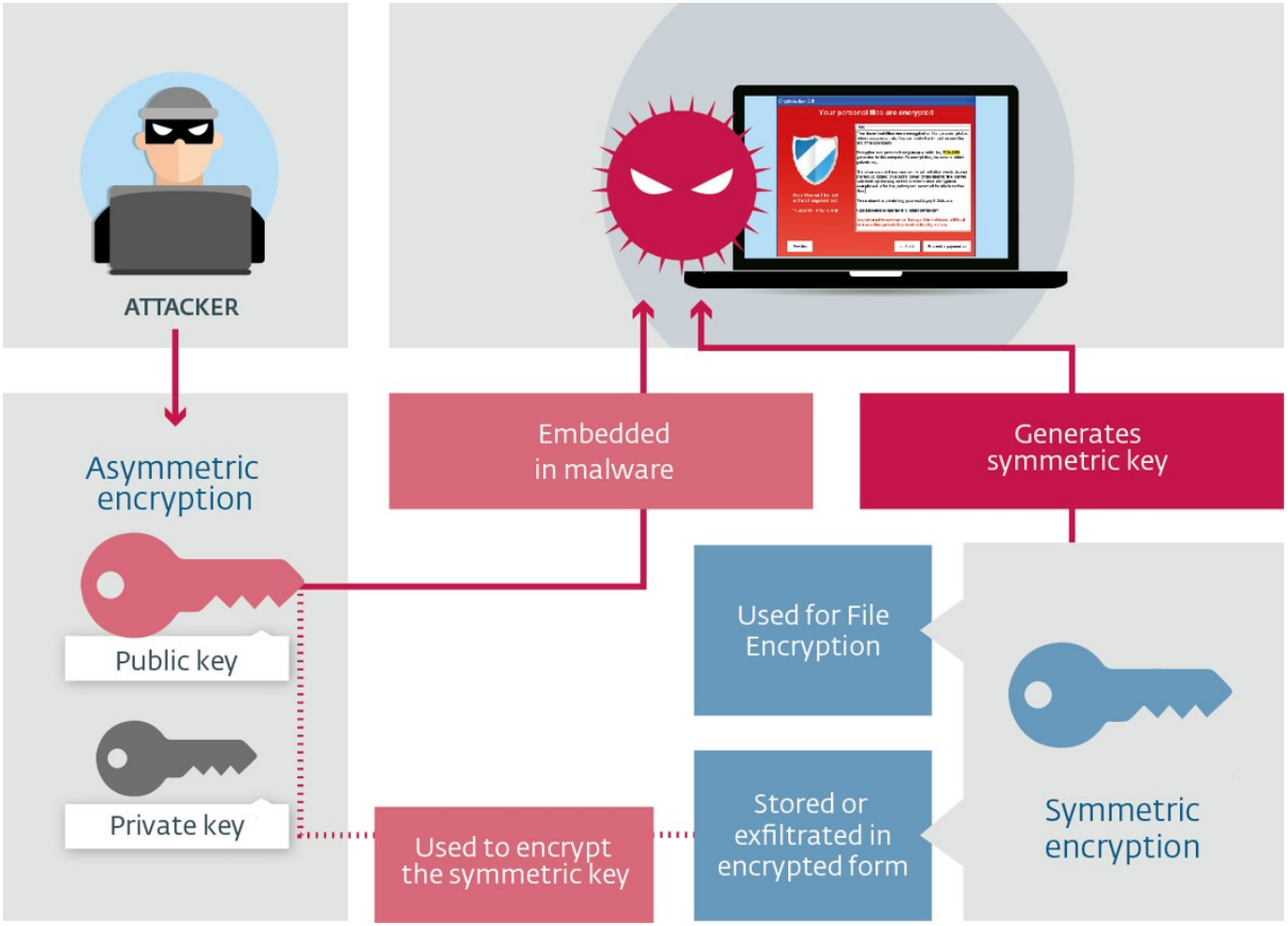
Les bloqueurs

Les crypto-ransomware

Ransomware
Fonctionnement

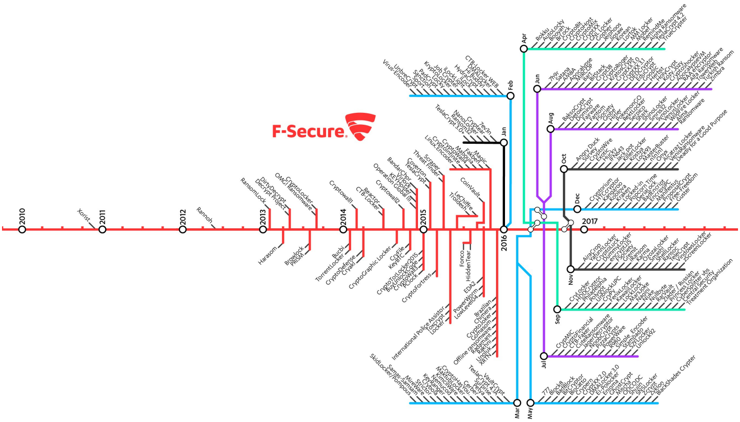
- Le ransomware embarque une clé publique asymétrique Kpub (ex : RSA)
- Cette clé Kpub permet de chiffrer un clé de session symétrique Ks (ex : AES) envoyée au serveur
- La clé Ks est utilisée pour chiffrer les fichiers de l'utilisateur
- Le seul moyen de récupérer les fichiers d'origine consiste à récupérer Ks

Ransomware
Illustration



Ransomware

Evolution des ransomwares de 2010 à 2017



Ransomware

Cause de l'évolution des ransomwares

- Popularisation du Bitcoin qui permet aux hackeurs d'être très difficile à tracer
- Des algorithmes de chiffrement de plus en plus complexes, impossible à déchiffrer sans connaitre la clé
- Professionnalisation des ransomwares et du monde criminel en général
- Des techniques de marketing et social engineering pour inciter à payer
- Pour les entreprises, payer la rançon est souvent l'option la moins couteuse

Ransomware
Quelques conseils

- Ne payer pas la somme demandée
- Ne pas cliquer sur les pièces jointes contenues dans les messages électroniques
- Maintenir vos logiciels à jour
- Utiliser un logiciel de sécurité
- Effectuer des sauvegardes

Ransomware
Exemple célèbre

Exemple

Wannacry (2017)

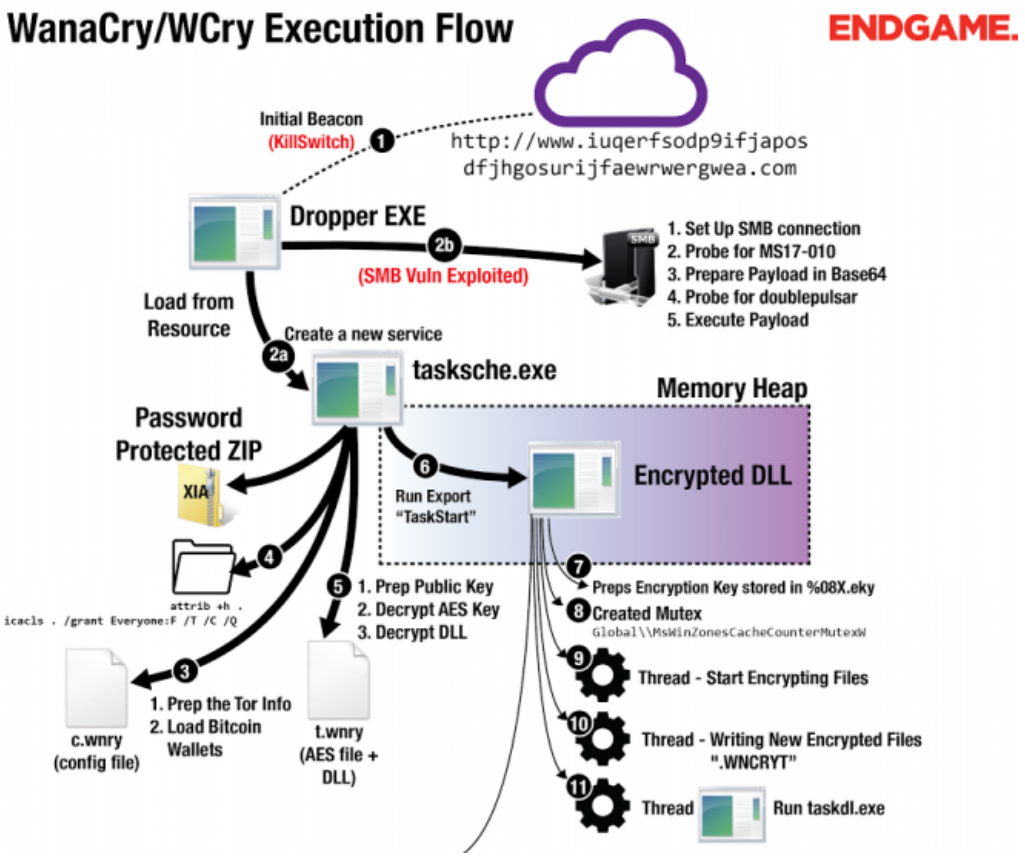
- Utilise l'exploit EternalBlue de la NSA publié par The Shadow Brokers visant le protocole SMB
- Plus de 200000 systèmes infectés à travers 150 pays
- Support 28 langues et chiffres 179 types de fichiers

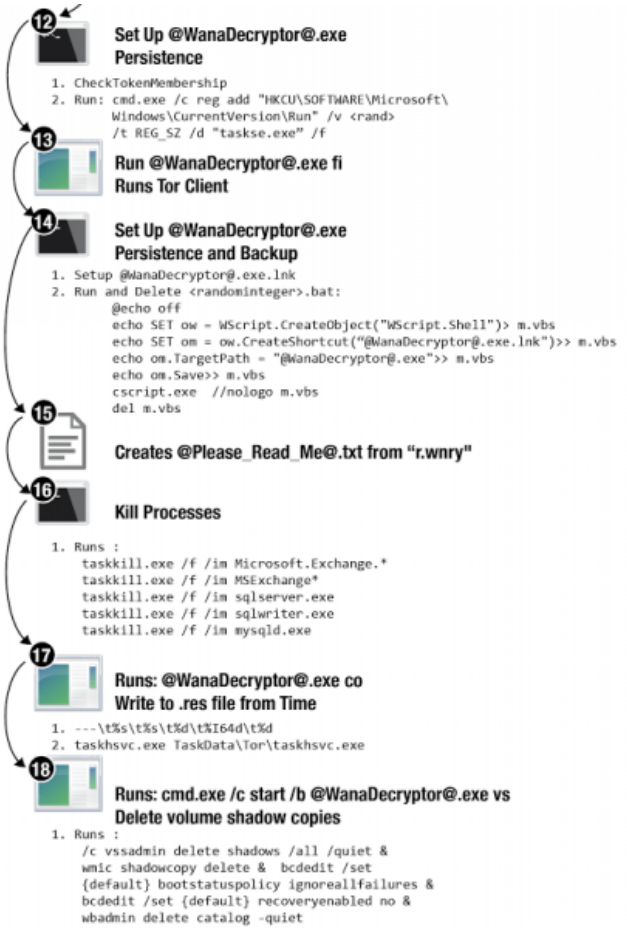
Ransomware

Exemple célèbre - Wannacry

Ransomware

Exemple célèbre - Wannacry

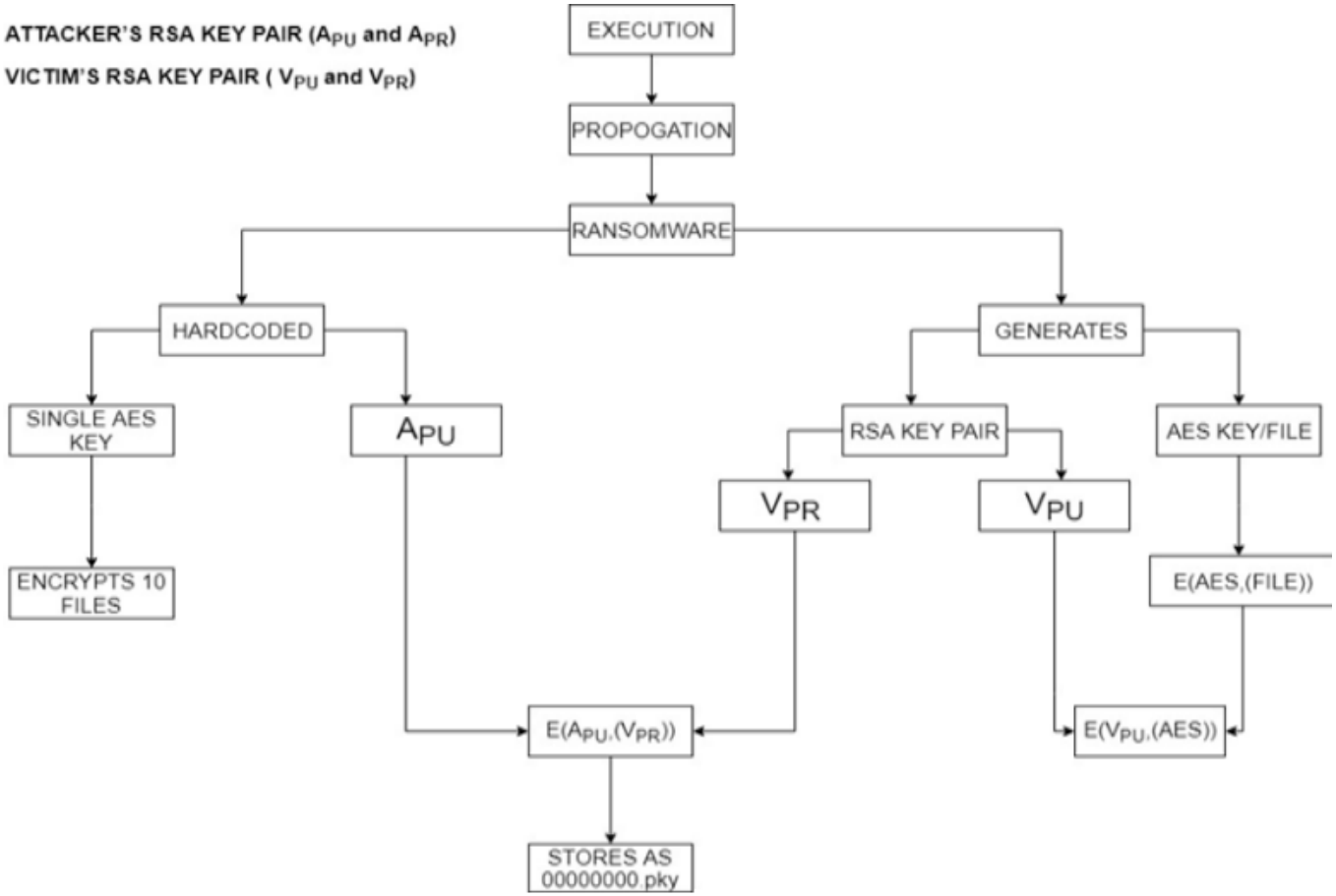




Ransomware

Exemple célèbre - Wannacry

Modèle cryptographique



Ransomware

Exemple célèbre - Wannacry

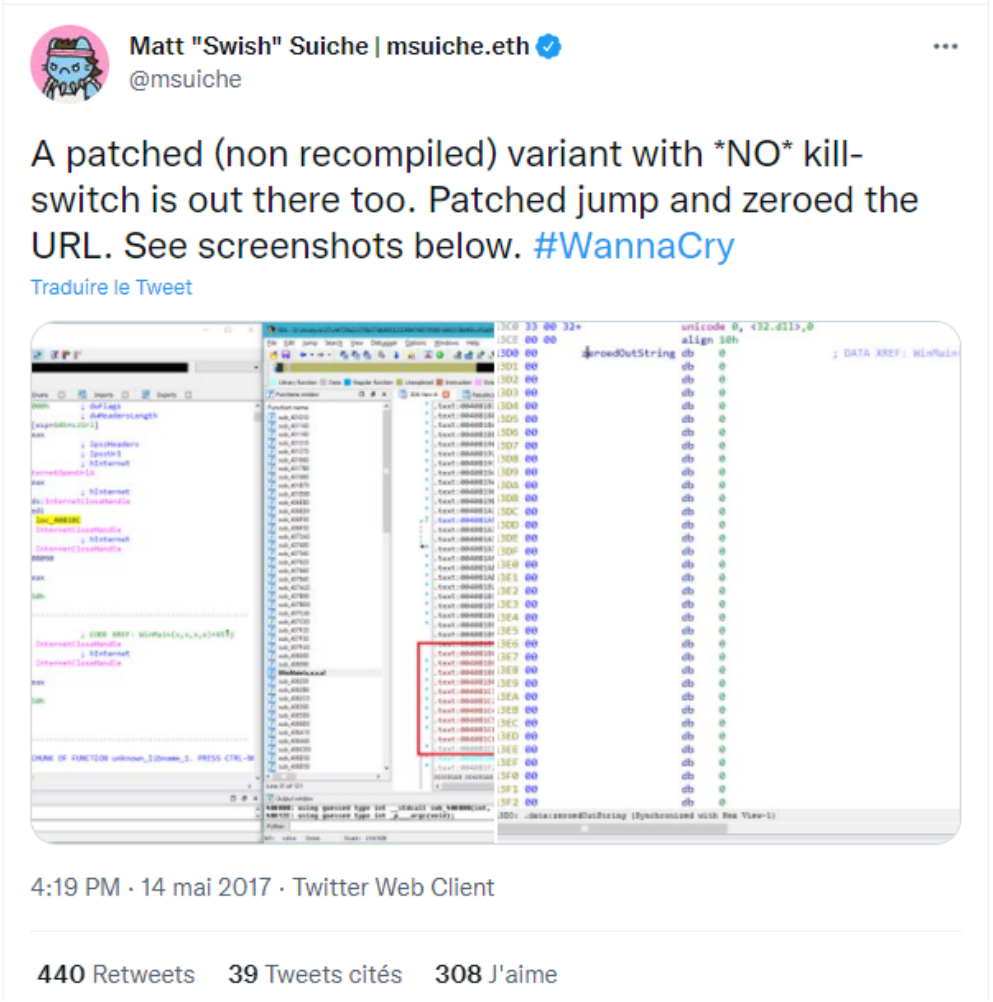
Présence d'un kill switch dans le code

```
qmemcpy(&szUrl, sinkholeddomain, 0x39u);      // previously unregistered domain, now sinkholed
v8 = 0;
v9 = 0;
v10 = 0;
v11 = 0;
v12 = 0;
v13 = 0;
v14 = 0;
v4 = InternetOpenA(0, 1u, 0, 0, 0);
v5 = InternetOpenUrlA(v4, &szUrl, 0, 0, 0x84000000, 0);// do HTTP request to previously unregistered domain
if ( v5 )                                     // if request successful quit
{
  InternetCloseHandle(v4);
  InternetCloseHandle(v5);
  result = 0;
}
else                                          // if request fails, execute payload
{
  InternetCloseHandle(v4);
  InternetCloseHandle(0);
  detonate();
  result = 0;
}
return result;
}
```

Ransomware

Exemple célèbre - Wannacry

Evolution du malware pour supprimer le kill switch



Ransomware

Exemple célèbre - Wannacry

Evolution du malware pour supprimer le kill switch

```
.text:00408183 68 00 00+            push    84000000h       ; dwFlags
.text:00408188 6A 00                push    0               ; dwHeadersLength
.text:0040818A 8D 4C 24+            lea     ecx, [esp+64h+szUrl]
.text:0040818E 8B F0                mov     esi, eax
.text:00408190 6A 00                push    0               ; lpszHeaders
.text:00408192 51                   push    ecx             ; lpszUrl
.text:00408193 56                   push    esi             ; hInternet
.text:00408194 FF 15 38+            call    ds:InternetOpenUrlA
.text:0040819A 8B F8                mov     edi, eax
.text:0040819C 56                   push    esi             ; hInternet
.text:0040819D 8B 35 3C+            mov     esi, ds:InternetCloseHandle
.text:004081A3 85 FF                test    edi, edi
.text:004081A5 75 15                jnz     short loc_4081BC
.text:004081A7 FF D6                call    esi ; InternetCloseHandle
.text:004081A9 6A 00                push    0               ; hInternet
.text:004081AB FF D6                call    esi ; InternetCloseHandle
.text:004081AD E8 DE FE+            call    sub_408090
.text:004081B2 5F                   pop     edi
.text:004081B3 33 C0                xor     eax, eax
.text:004081B5 5E                   pop     esi
.text:004081B6 83 C4 50             add     esp, 50h
.text:004081B9 C2 10 00             retn    10h
.text:004081BC          ; ---------------------------------------------------
.text:004081BC
.text:004081BC          loc_4081BC:                         ; CODE XREF: WinMa
.text:004081BC FF D6                call    esi ; InternetCloseHandle
.text:004081BE 57                   push    edi             ; hInternet
.text:004081BF FF D6                call    esi ; InternetCloseHandle
.text:004081C1 5F                   pop     edi
.text:004081C2 33 C0                xor     eax, eax
.text:004081C4 5E                   pop     esi
.text:004081C5 83 C4 50             add     esp, 50h
.text:004081C8 C2 10 00             retn    10h
.text:004081C8          _WinMain@16 endp
```

```
.text:00408183 68 00 00+            push    84000000h       ; dwFlags
.text:00408188 6A 00                push    0               ; dwHeadersLength
.text:0040818A 8D 4C 24+            lea     ecx, [esp+64h+szUrl]
.text:0040818E 8B F0                mov     esi, eax
.text:00408190 6A 00                push    0               ; lpszHeaders
.text:00408192 51                   push    ecx             ; lpszUrl
.text:00408193 56                   push    esi             ; hInternet
.text:00408194 FF 15 38+            call    ds:InternetOpenUrlA
.text:0040819A 8B F8                mov     edi, eax
.text:0040819C 56                   push    esi             ; hInternet
.text:0040819D 8B 35 3C+            mov     esi, ds:InternetCloseHandle
.text:004081A3 85 FF                test    edi, edi
.text:004081A5 75 00                jnz     short $+2
.text:004081A7
.text:004081A7          loc_4081A7:                         ; CODE XREF: WinMain(x,x,x,x)+65
.text:004081A7 FF D6                call    esi ; InternetCloseHandle
.text:004081A9 6A 00                push    0               ; hInternet
.text:004081AB FF D6                call    esi ; InternetCloseHandle
.text:004081AD E8 DE FE+            call    sub_408090
.text:004081B2 5F                   pop     edi
.text:004081B3 33 C0                xor     eax, eax
.text:004081B5 5E                   pop     esi
.text:004081B6 83 C4 50             add     esp, 50h
.text:004081B9 C2 10 00             retn    10h
.text:004081B9          _WinMain@16 endp
.text:004081BC          ; ---------------------------------------------------
.text:004081BC FF D6                call    esi
.text:004081BE 57                   push    edi
.text:004081BF FF D6                call    esi
.text:004081C1 5F                   pop     edi
.text:004081C2 33 C0                xor     eax, eax
.text:004081C4 5E                   pop     esi
.text:004081C5 83 C4 50             add     esp, 50h
.text:004081C8 C2 10 00             retn    10h
.text:004081C8          ; ---------------------------------------------------
.text:004081CB 90 90 90+            align 10h
```

Similar to nop-nop!!

Unreachable Code

Plan

Etat de la menace

Rapports publiques

De nombreuses entreprises publient des rapports sur l'état de la menace :

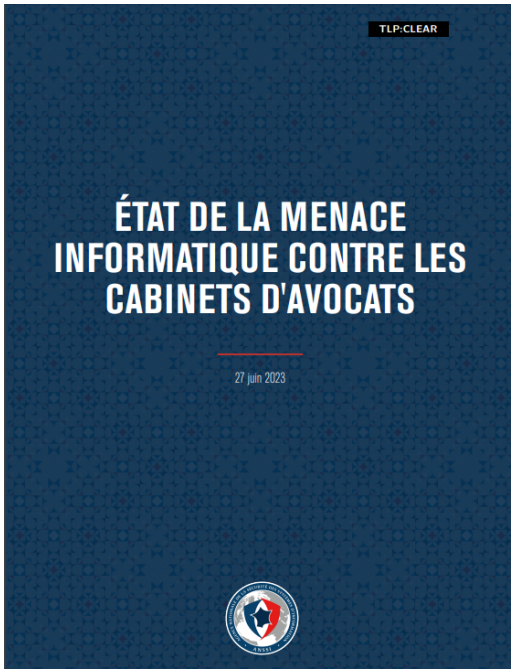
- Editeurs antivirus
  - Norton
  - F-Secure
- Sociétées spécialisées dans la réponse sur incident
  - CrowdStrike
  - Mandiant
- Editeurs de produit de sécurité / de protection / Cyber Threat Intelligence
  - Thales
  - Sekoia
- CERT (Computer emergency response team)

Etat de la menace 2021

Rapports publiques

Etat de la menace 2023

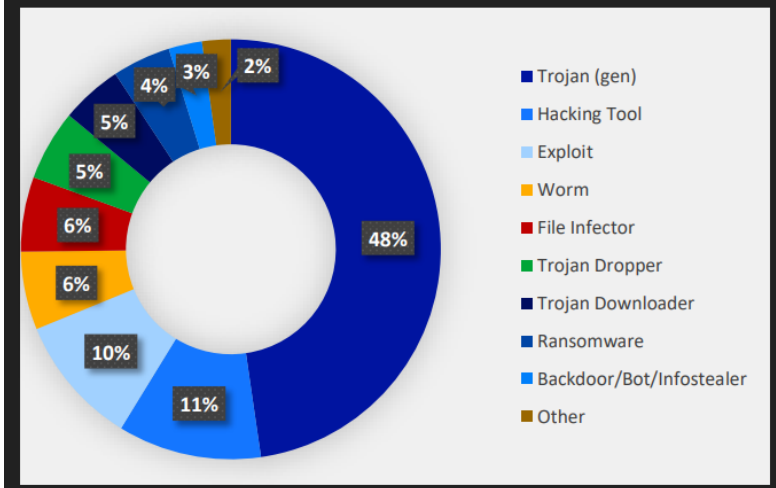
Rapports publiques



Etat de la menace

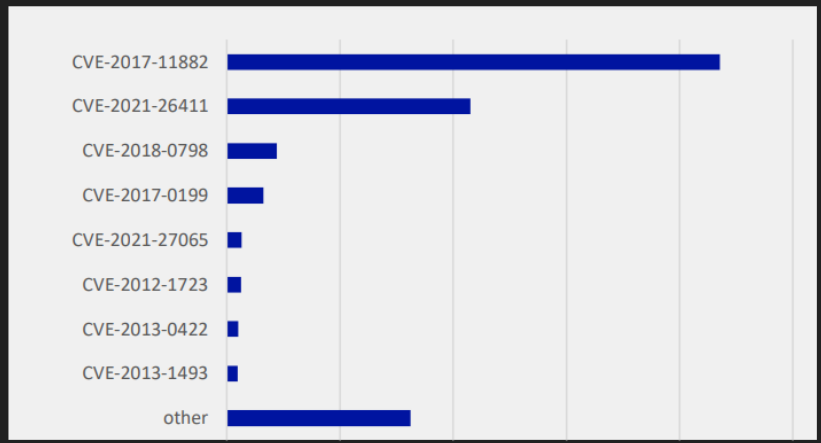
THREAT TYPES

In November, various hacking tools and exploits have been the most prevalent identified threat. Detected hacking tools include software such as Mimikatz. Equation group tools contribute to biggest share of detected hacking tools.



EXPLOITS

CVE-2017-11882 exploits vulnerability in MS office products and provides remote code execution for the attacker. Most common attack vector is via email campaigns. It remains the most popular vulnerability against endpoints. This attack can be mitigated by updating the MS office with latest security updates.
CVE-2021-26411 internet explorer memory corruption vulnerability follows at the second place.



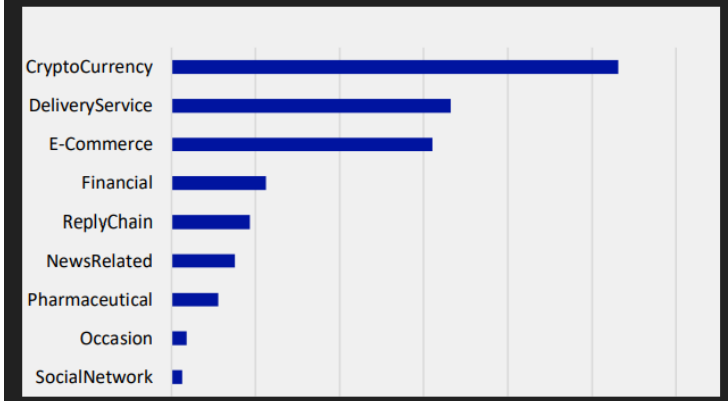
. Source : f-secure-threat-highlights-report-2021-11.pdf
https://www.f-secure.com/en/business/resources/threat-highlights-report

Etat de la menace

SPAM EMAIL THEMES

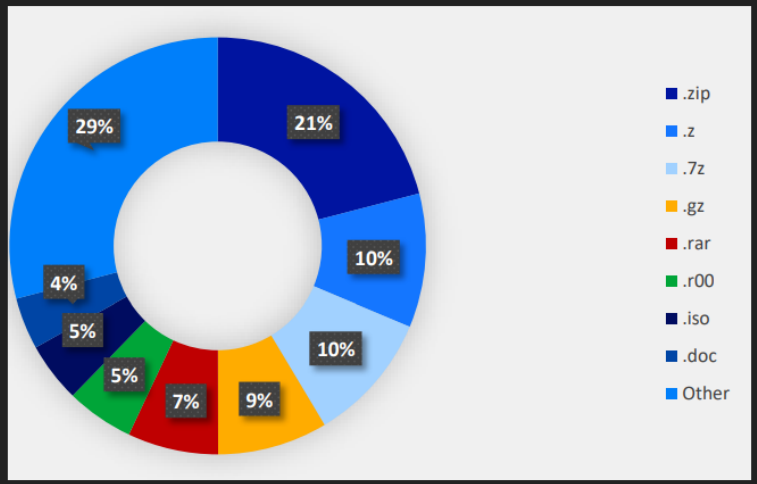
Cryptocurrency themes dominate the spam landscape followed by delivery service and E-Commerce.

FBI has released a public service announcement regarding increasing number of cryptocurrency scams leveraging crypto ATMs and QR codes for payment.
https://www.ic3.gov/Media/Y2021/PSA211104



MALICIOUS EMAIL ATTACHMENTS

In November, .zip attachments in emails were particularly prevalent followed by other archive formats.
Majority of the .zip volume is attributable to Agent Tesla campaigns where the malware is delivered within archives such as .zip & .gz.



. Source : f-secure-threat-highlights-report-2021-11.pdf
https://www.f-secure.com/en/business/resources/threat-highlights-report

Etat de la menace

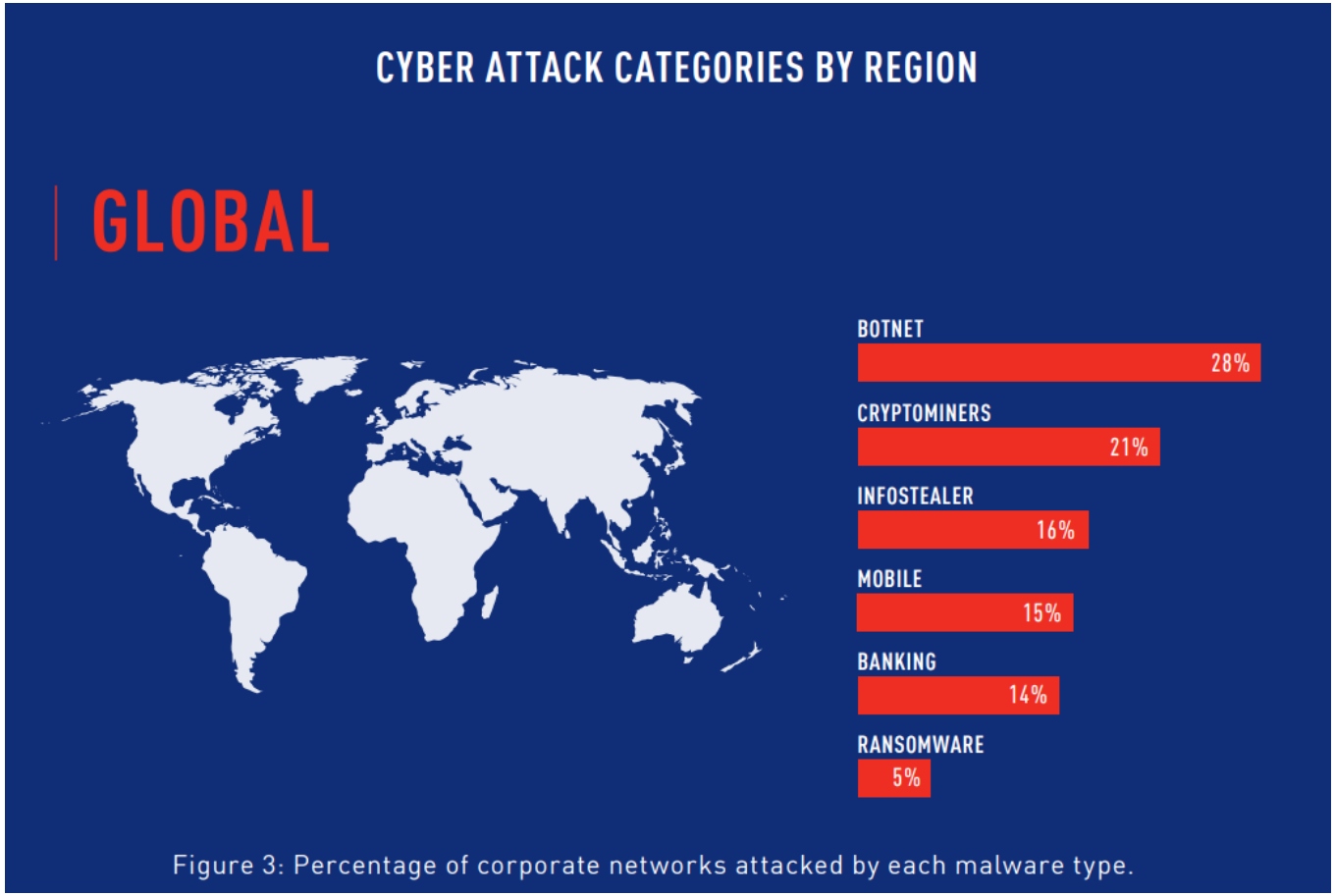


Figure 3: Percentage of corporate networks attacked by each malware type.

. Source : Check Point cyber-security-report-2021.pdf

Etat de la menace

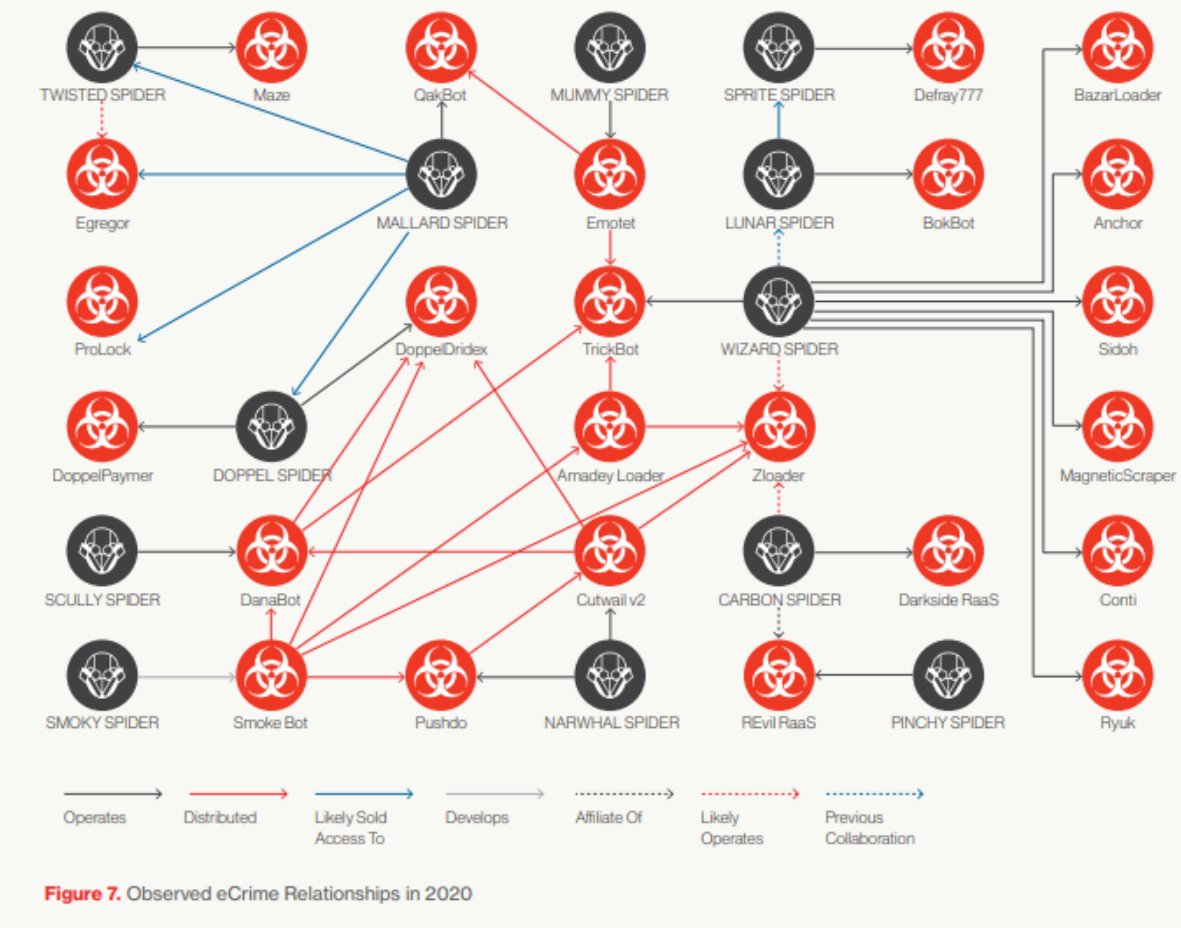
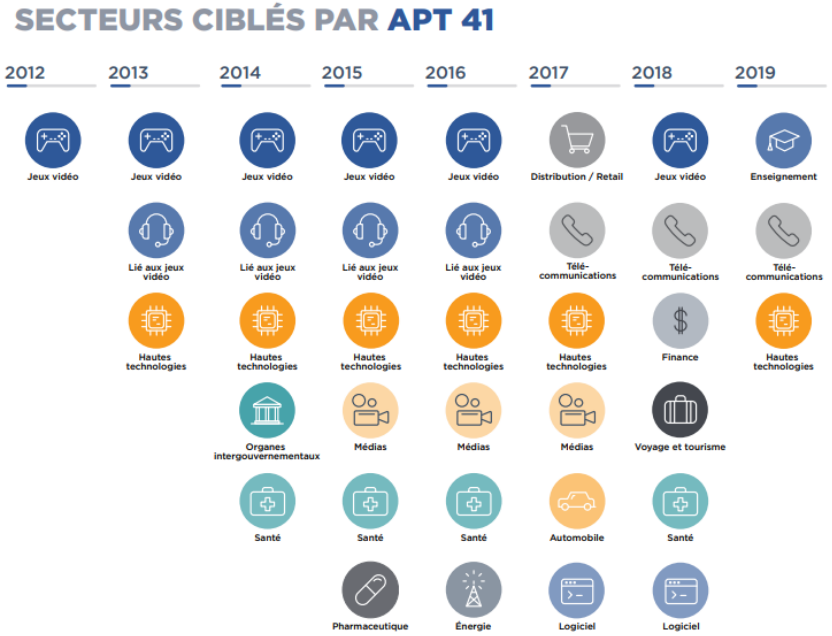


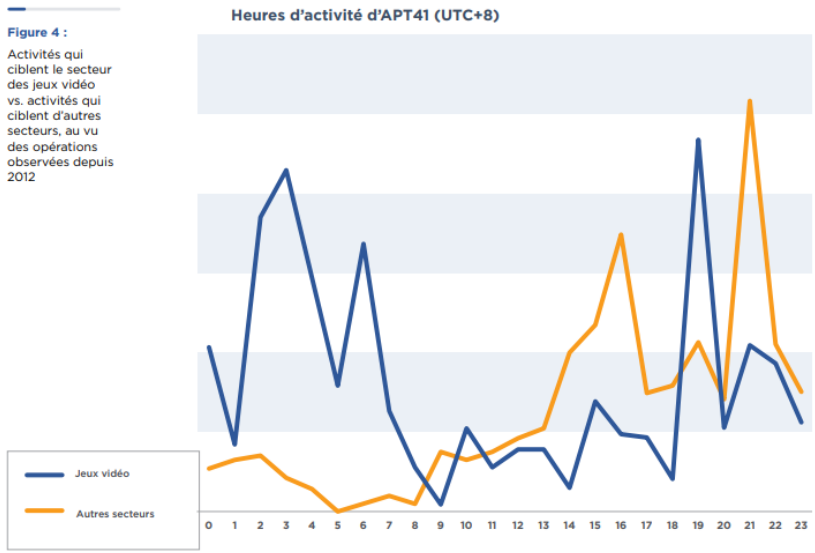
Figure 7. Observed eCrime Relationships in 2020

. Source : Crowdstrike - 2021 Global Threat Report

Etat de la menace

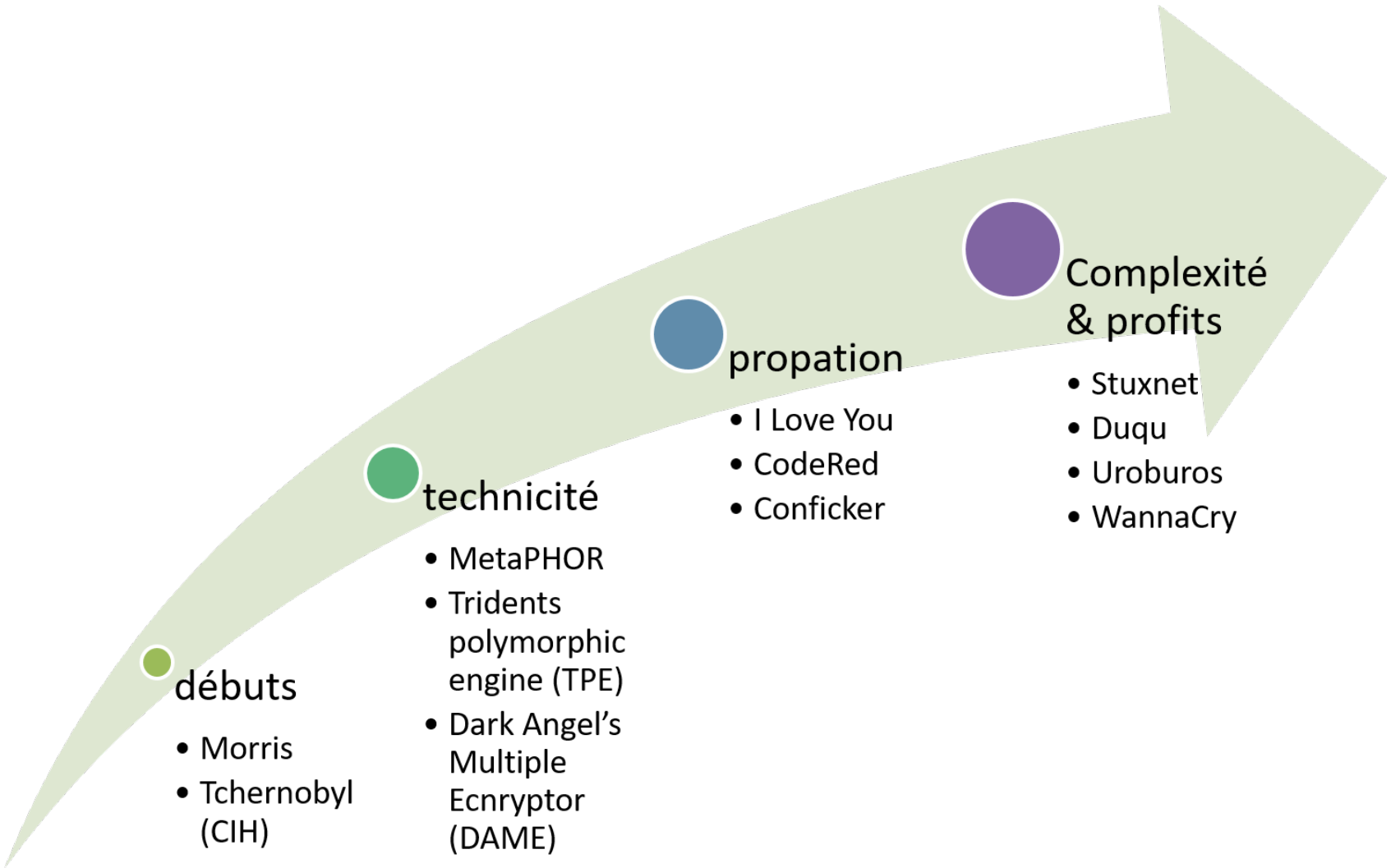
Analyse d'un groupe d'attaquant - APT41





. Source : Fireeye - rpt-apt41.pdf
https://www.fireeye.fr/current-threats/apt-groups.html

Evolution


débuts
• Morris
• Tchernobyl (CIH)
technicité
• MetaPHOR
• Tridents polymorphic engine (TPE)
• Dark Angel's Multiple Ecnryptor (DAME)
propation
• I Love You
• CodeRed
• Conficker
Complexité & profits
• Stuxnet
• Duqu
• Uroburos
• WannaCry

Conclusion

- Difficile de modéliser les malware
  - Menaces complexe et très variés
- Évolution constante en fonction
  - Du matériel ex : virtualisation
  - Des OS : patchguard
  - Des firmware : bootkit
  - Des technologies : cryptomonaies
- Constitue un bonne veille technologique :
  - Exploits
  - Enjeux stratégiques
  - Groupe d'attaquants
  - Etc.

Références

Filiol E.

Les virus informatiques : théorie, pratique et applications.
Springer, 2009.

Hoglund G. and Butler J.

Rootkits : subverting the Windows kernel.
Addison-Wesley Professional, 2006.

Sikorski M. and Honig.

A. Practical Malware Analysis : The Hands-On Guide to Dissecting Malicious Software.
No Starch Press, 2012.

Szor P.

The art of computer virus research and defense.
Addison-Wesley Professional, 2005.