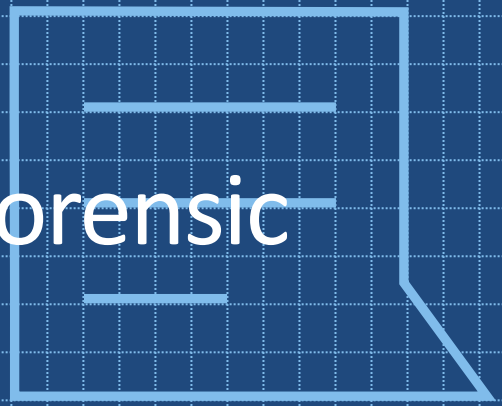
Malwares Analysis & Forensic



EFREI-PARIS

B. AIT SALEM

Plan

Quelques notions sur les codes malveillants (malwares)

Analyse d’un code malveillant

Bibliographie

The Art of Memory Forensics - Detecting Malware and Threats in Windows, Linux, and Mac Memory – Edition Wiley - Michael Hale Ligh, Andrew Case, Jamie Levy, AAron Walters.

The Antivirus Hacker's Handbook - Joxean Koret, Elias Bachaalany – Wiley

Gray Hat Hacking, The Ethical Hacker's Handbook, Fourth Edition - Pedram Amini et al. – Mc Graw Hill Education

Windows Malware Analysis Essentials - Victor Marak – PACKT Publishing

Practical Malware Analysis - Michael Sikorski et Andrew Honig - no strach press

Learning Linux Binary Analysis – Ryan O'Neill – PACKT Publishing Open source

Malware Analyst's Cookbook Tools and Techniques for Fighting Malicious Code - Michael Hale Ligh et al. – Wiley Publishing

Practical Binary Analysis Build Your Own Linux Tools for Binary Instrumentation, Analysis, and Disassemblyby Dennis Andriesse

Définition d’un Malware

Un logiciel malveillant qui endommage ou désactive une ou plusieurs fonctionnalités d’un système informatique et donne un contrôl total ou partiel à l’attaquant à des fins de vol ou de …

Vecteurs d'attaque

- Messagerie instantanée
- IRC (Internet Relay Chat)
- Disques amovibles
- Pièces jointes
- Bugs ou vulnérabilités sur les navigateurs web et logiciels de messagerie
- Partage de fichiers
- Programmes factices
- Sites Webs malveillants et logiciels distribués gratuitement
- Téléchargement de fichiers et de jeux vidéos sur Internet
- Logiciel légitime déguisé et distribué par un employé

Techniques de propagation de malwares sur le Web

- Bien noter les pages contenant des malwares sur les moteurs de recherche.
- Inciter les utilisateurs à cliquer sur des pages web en apparence innocente via les réseaux sociaux par exemple (en faisant du social engineering)
- Usurper une institution légitime (ex. son site Web ou via un courrier électronique) dans le but de voler des informations d'authentification (ex. les login/password)
- Compromettre un site web légitime ou mieux encore un réseau publicitaire qui propagerait le malware à ses visiteurs/destinataires
- Exploiter les failles sur les navigateurs web en installant le malware sur le poste de la victime lorsqu'il visite une page web

Types de logiciels malveillants

- Rootkit
- Worms
- Virus
- Trojan Horse
- Ransomware
- Botnet
- Backdoor
- Spyware
- Adware
- Crypter

Types de logiciels malveillants

-

Virus

Code en langage machine qui se **greffent** sur un programme utilisé sur le système cible, afin d'en modifier le comportement.

Le virus peut être **tout entier** contenu dans ce greffon, ou il peut s'agir d'une **simple amorce**, dont le rôle va être de télécharger un programme plus important qui sera le vrai virus.

Une fois implanté sur son programme-hôte, le greffon possède aussi en général la capacité de se **recopier** sur d'autres programmes

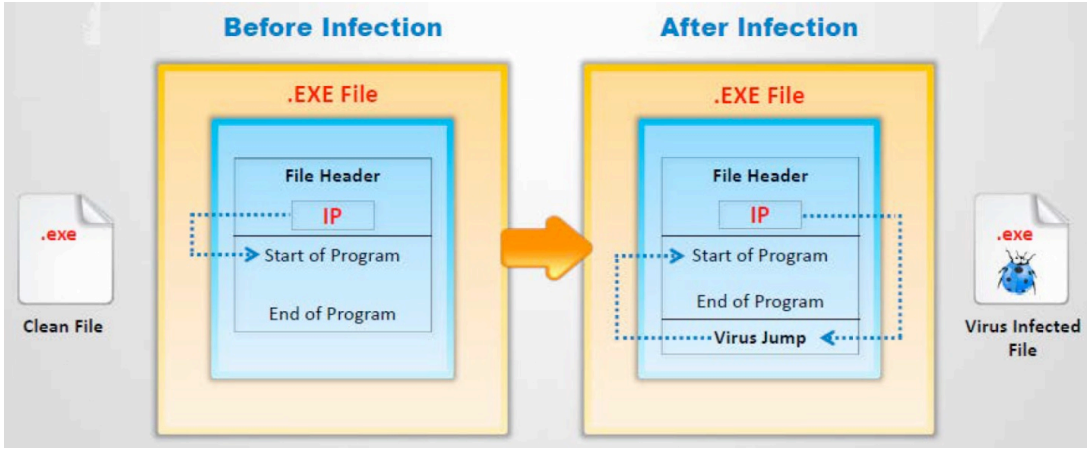
Types de logiciels malveillants

-

Virus

-

Méthode d'infection



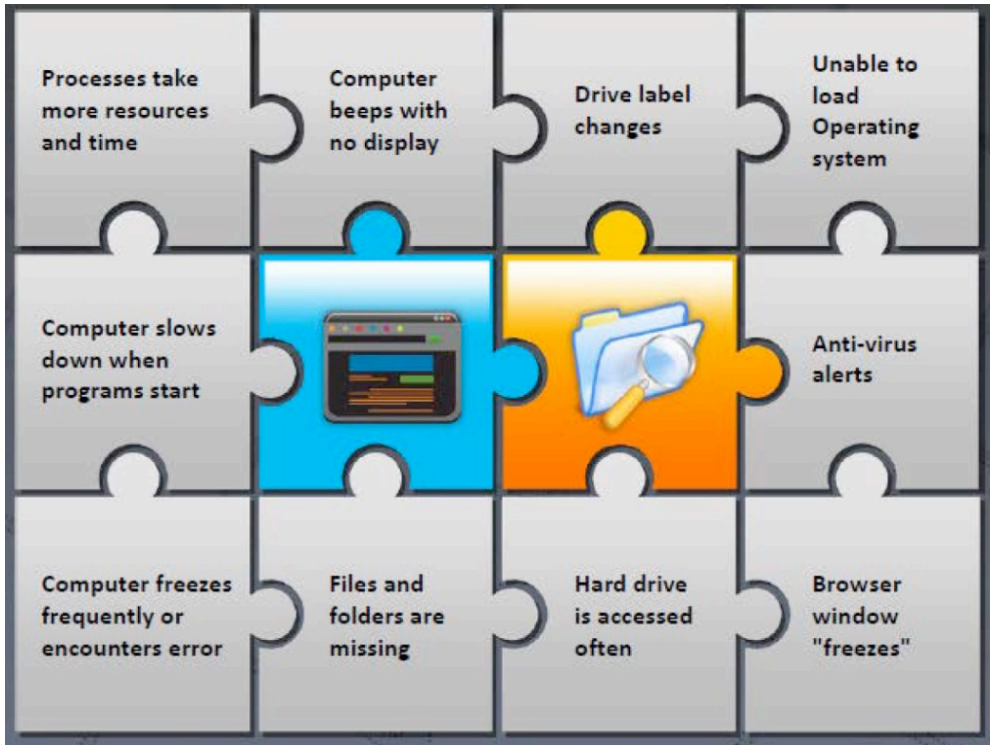
Types de logiciels malveillants

-

Virus

-

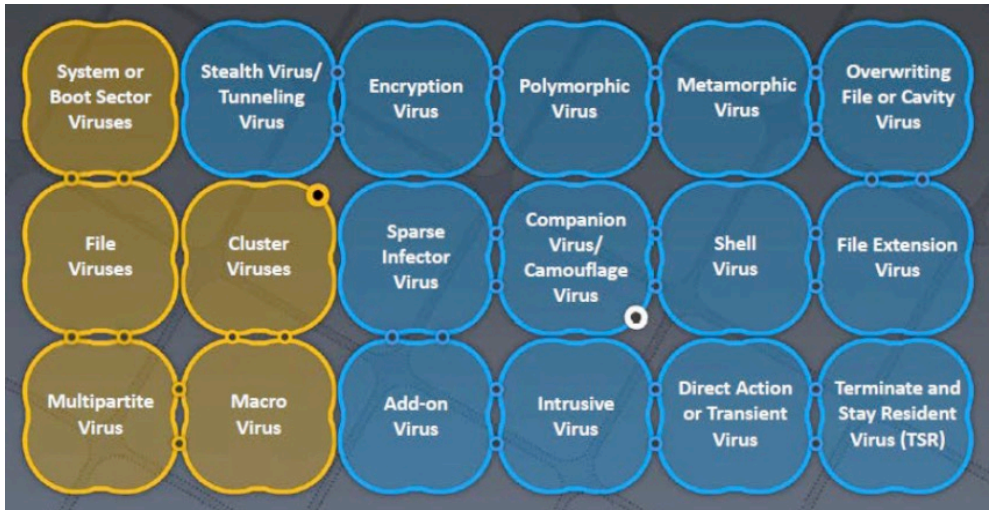
Indicateurs



Types de logiciels malveillants

-

Types de Virus



Types de logiciels malveillants

-

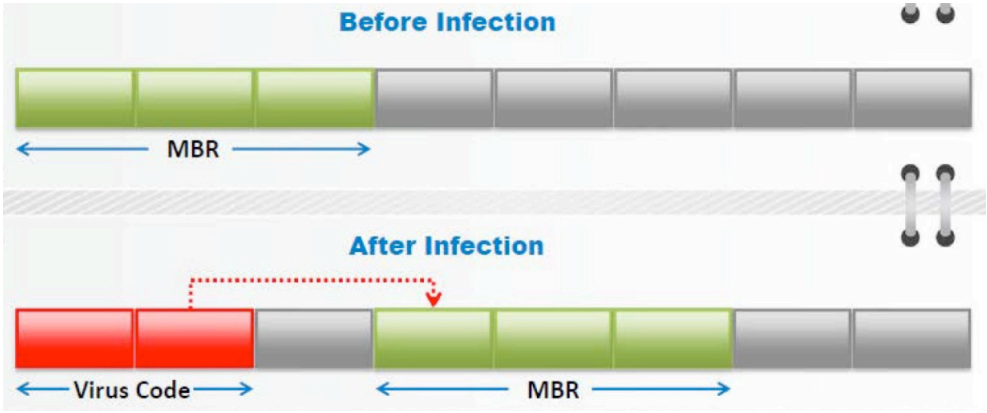
Types de Virus

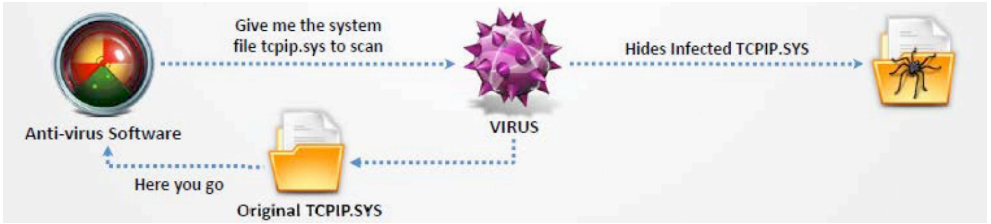
-

Virus de boot

-

Stealth / Tunneling Virus





Types de logiciels malveillants

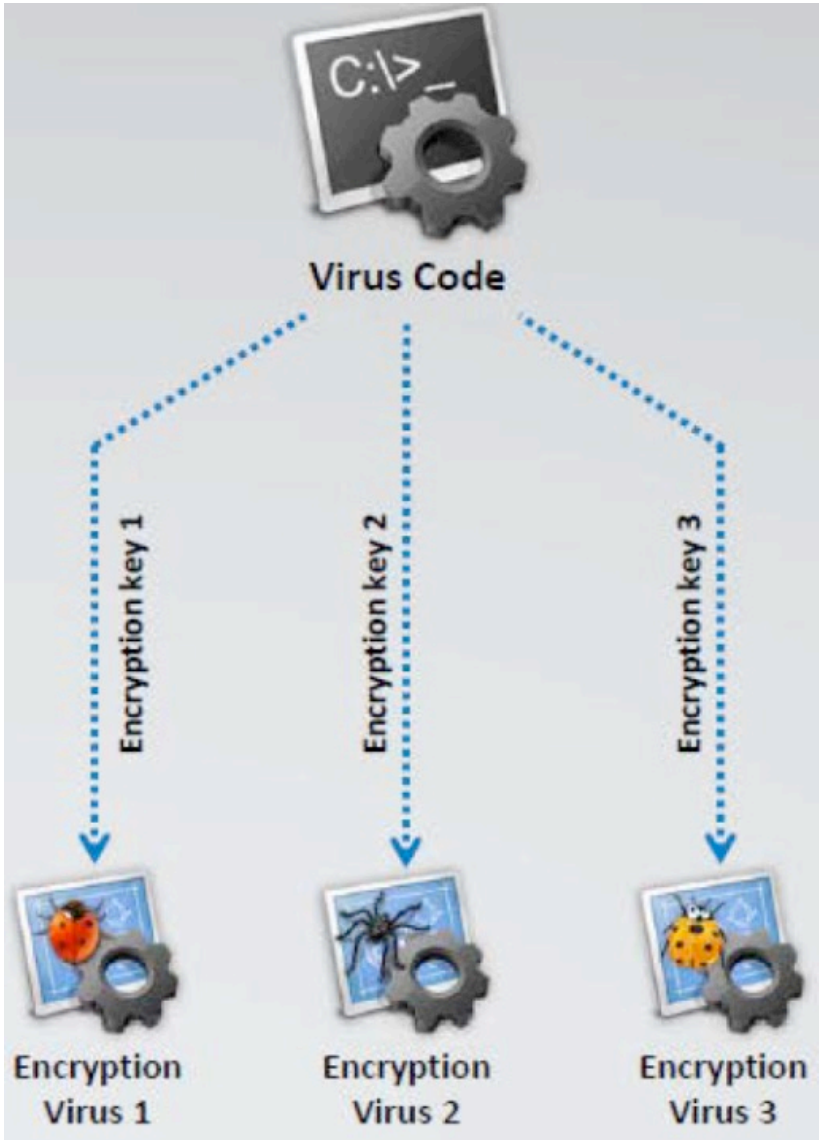
-

Types de Virus

-

Virus chiffré

- Chiffrement du code
- Une nouvelle clé de chiffrement par infection
- Un anti-virus ne pourra pas les détecter en utilisant une approche basée sur les signatures



Types de logiciels malveillants

-

Types de Virus

-

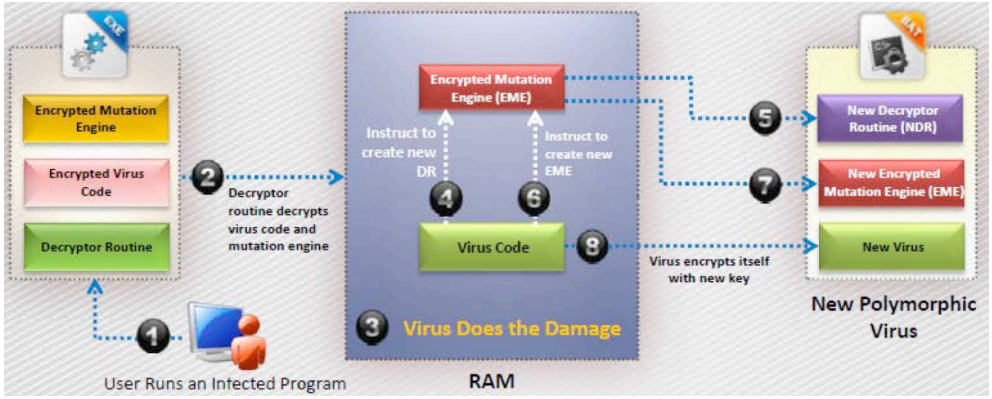
Virus polymorphic

- Le code mute à chaque infection, mais pas l'algorithme originel
- Pour cela, le virus embarque ce qu'on appelle un « polymorphic engine »
- Un bon polymorhic virus est celui où chaque partie du code est complètement différente à chaque infection

-

Virus Metamorphic

- Le virus réécrit complètement sont code à chaque infection



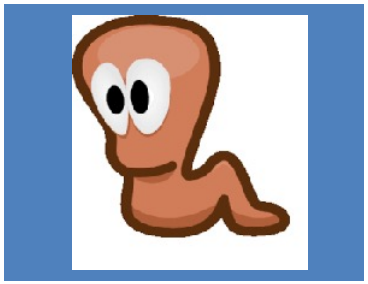
Types de logiciels malveillants

-

Types de Virus

-

Ver



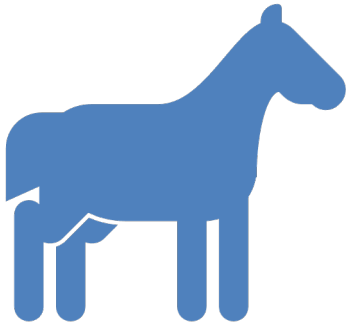
Un ver (**worm**) est une variété de virus qui se **propage par le réseau**. Il peut s'agir d'un bot.

VS

Autrefois les virus n'étaient pas des vers (ils ne se propageaient pas par le réseau) et les vers n'étaient pas des virus, aujourd'hui la confusion entre les deux catégories est presque totale.

Types de logiciels malveillants
-
Cheval de Troie

- Un cheval de Troie *(**Trojan horse**) est un logiciel qui se présente sous un jour* honnête, utile ou agréable, et qui une fois installé sur un ordinateur y effectue des actions cachées et pernicieuses.

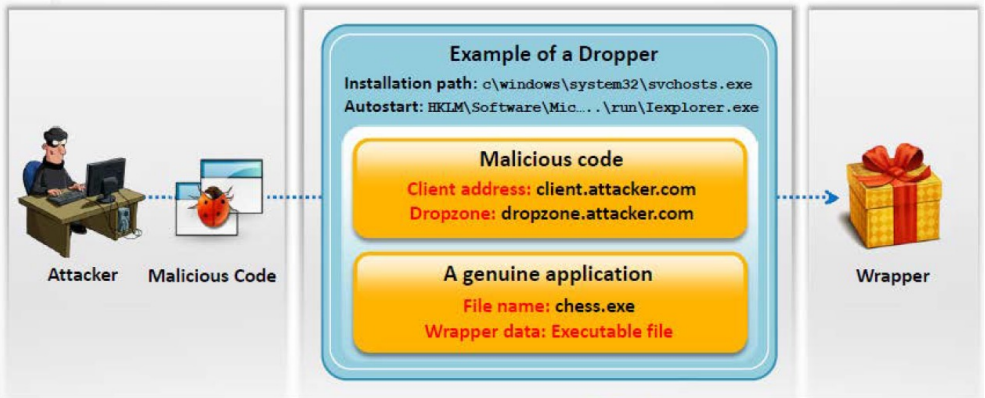


Objectifs des Trojans

- Supprimer ou remplacer des fichiers critiques du système d'exploitation
- Désactiver l'anti-virus et/ou le pare-feux
- Créer du trafic réseau inutile pour mener une attaque DoS
- Créer une porte dérobée (backdoor) pour obtenir un accès distant
- Enregistrer des captures d'écrans, audio et vidéos
- Infecte la machine victime et la transforme en un serveur proxy pour des attaques par relais
- Utilise le pc de la victime pour faire du spamming
- Utilise le pc de la victime comme Botnet (DDoS)
- Télécharge des spywares et adwares et des fichiers malveillants sur le pc de la victime
- Vol d'informations comme les mots de passe, codes de sécurité, et les informations des cartes de crédits en utilisant des keyloggers

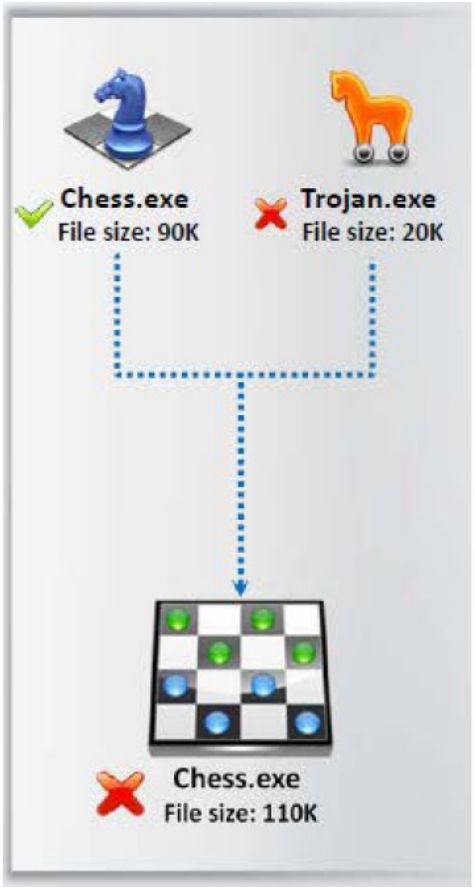
Démarche d'infection des Trojans

1. Créer un nouveau paquet Trojan en utilisant un **kit de construction de chveaux de troie** (**ex.** Metasploit)
2. Créer un "**dropper**" à inclure dans le paquet qui a pour fonction d'installer le **code malveillant** sur le système cible

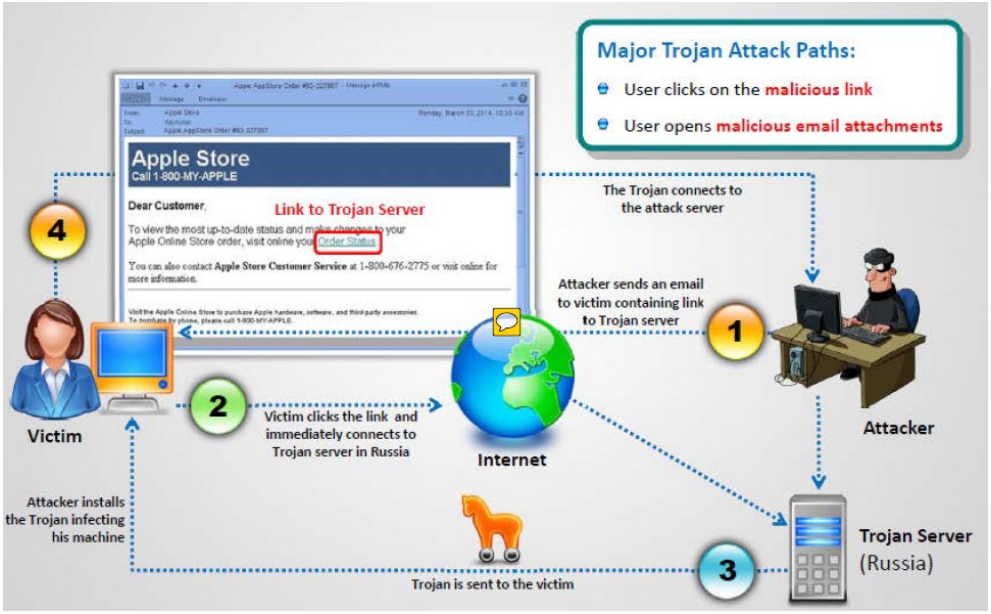


Démarche d'infection des Trojans

- Créer une enveloppe (**wrapper**) afin d'installer le Trojan sur le poste de la victime
- Propager le Trojan
- Exécuter le dropper



Déploiement d'un Trojan



Trojan: Eviter la détection par un anti-virus

Décomposer le Trojan en plusieurs parties et les compresser comme un unique fichier

Toujours écrire son propre Trojan

Modifier la syntaxe du Trojan

- Convertir un EXE en VB script
- Modifier l'extension .exe en .doc.exe, ppt.exe ou pdf.exe (Windows cache les extensions connues par défaut)

Changer le contenu du Trojan et aussi modifier le checksum et chiffrer le fichier

Ne jamais utiliser les Trojans téléchargés sur le Web (les antivirus peuvent facilement les détecter)

Types de Trojan

Command Shell Trojans

Offre un shell distant à la machine victime

C’est le type de Trojan le plus répandu

Le trojan est installé sur le poste de la victime, qui ouvre un port pour que l’attaquant s’y connecte.

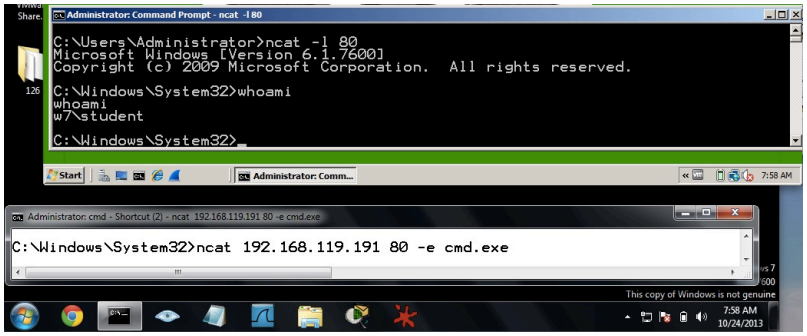
Un client est installé sur le poste de l’attaquant, qui est utilisé pour lancer des commandes sur le poste de la victime

- Command shell Trojan gives **remote control of a command shell** on a victim's machine
- Trojan server is installed on the victim's machine, which **opens a port for attacker** to connect. The client is **installed on the attacker's machine**, which is used to launch a command shell on the victim's machine

```
C:\>nc.exe -h
[v1.10 NT]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [options] [hostname] [port]
options:
        -d              detach from console, stealth mode
        -e prog         inbound program to exec [dangerous!!]
        -g gateway      source-routing hop point[s], up to 8
        -G num          source-routing pointer: 4, 8, 12, ...
        -h              this cruft
        -i secs         delay interval for lines sent, ports scanned
        -l              listen mode, for inbound connects
        -L              listen harder, re-listen on socket close
        -n              numeric-only IP addresses, no DNS
        -o file         hex dump of traffic
```

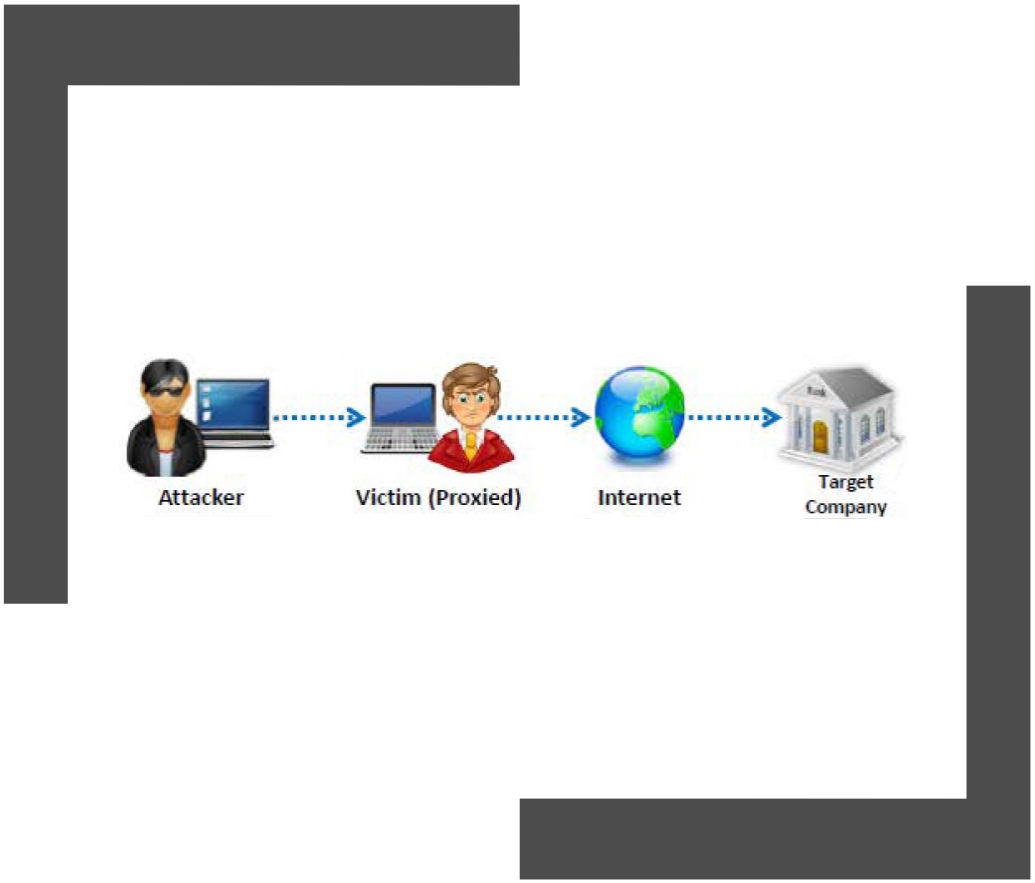
Command Shell Trojans

Reverse command Shell Trojans



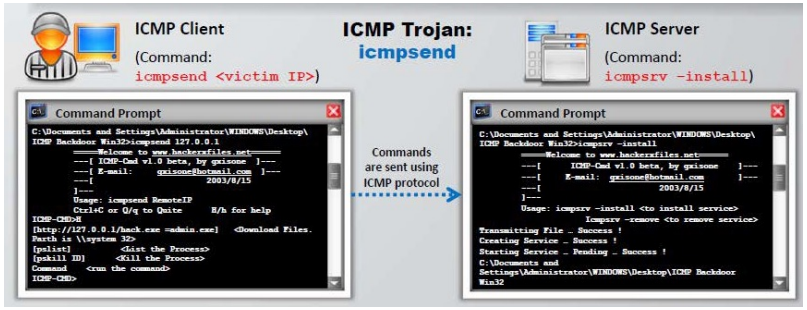
- La machine infectée se connecte à la machine de l'attaquant, demandant les commandes à exécuter.

Proxy Trojan



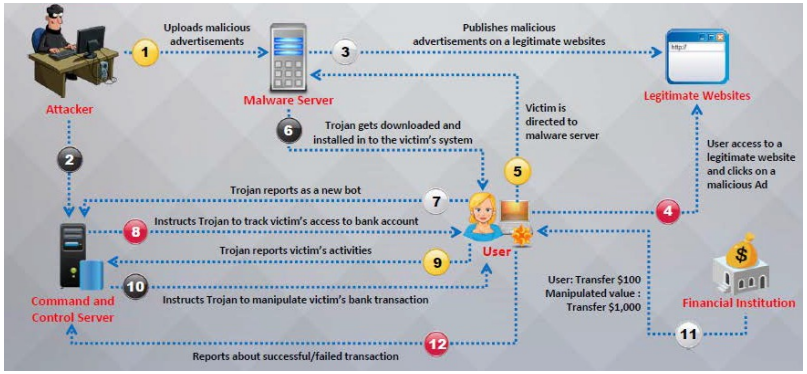
- Utilise la machine infectée comme proxy pour accéder à Internet

Trojan utilisant un Tunnel



- Permet de cacher un protocole dans un autre protocole non détectable comme ICMP

E-Backing Trojan



- Intercepte les données des comptes (numéros de cartes de crédit, détail des relevés de compte, etc.) des clients avant leur chiffrement et leur envoi
- Envoie ces données par e-mail, FTP, SSH, etc. à l'attaquant

E-Backing Trojan Exemples

TAN Grabbler

- Le trojan intercepte un TAN valide (Transaction Authentication Number) du client
- Le remplace par un TAN généré aléatoirement qui sera rejeté par la banque
- Envoi le TAN intercepté

Injection HTML

- L'attaquant crée un faux formulaire (nouvelles entrées) sur les pages du site web de la banque

Form Grabber

- Le trojan analyse les requêtes POST et les réponses du navigateur Web de la victime
- Pour Intercepter les entrés du clavier virtuel

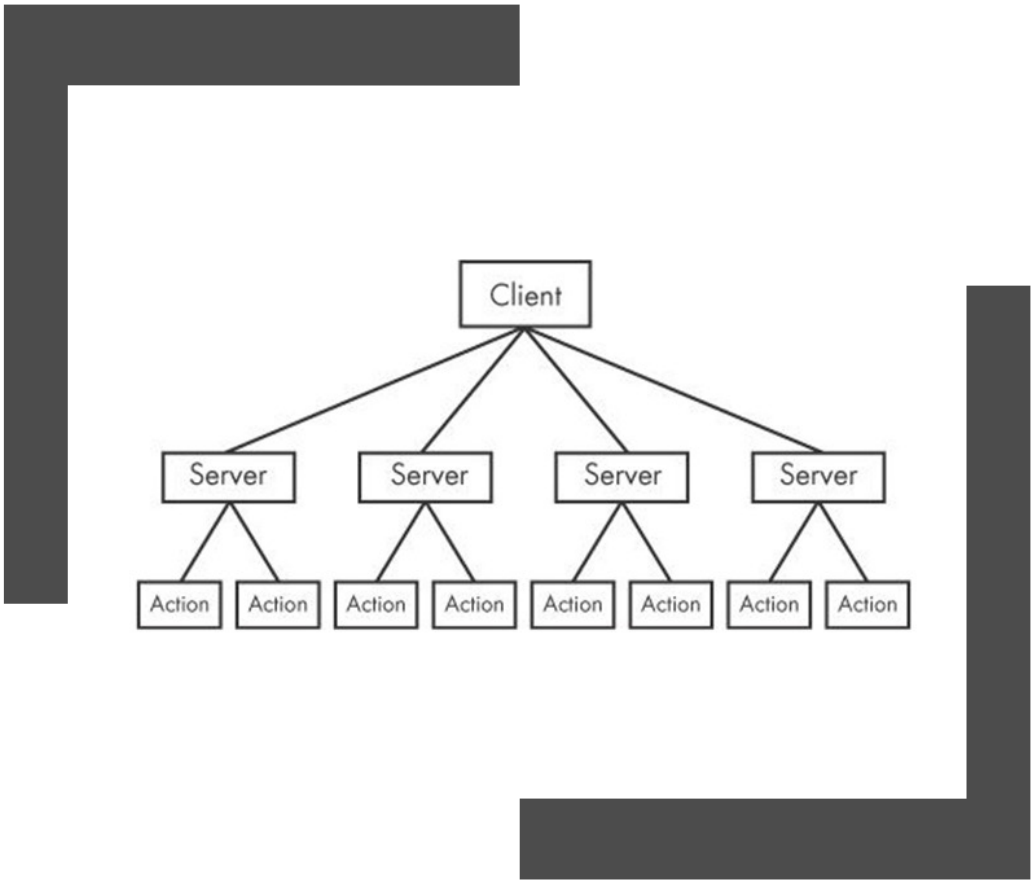
Types de logiciels malveillants

- **Botnet**
  - La cible d'un virus informatique peut être indirecte : il y a des exemples de virus qui se propagent silencieusement sur des millions d'ordinateurs connectés à l'Internet, sans y commettre le moindre dégât. Puis, **à un signal donné**, ou à une **heure fixée**, ces millions de programmes vont se connecter à un même serveur Web par exemple
  - Ceci provoquera son effondrement. C'est ce qu'on appelle un déni de service distribué *(**Distributed Denial of Service, DDoS**).*
  - On appelle un tel virus un ***bot**, et l'ensemble de ces virus déployés un **botnet**. Les ordinateurs infectés par des bots sont nommés **zombis**.*

Types de logiciels malveillants

- **RATs (Remote Administration Tools)**
  - Il est utilisé pour gérer à distance une machine ou plusieurs machines
  - Il est utilisé dans des attaques ciblées avec un objectif bien précis comme le vol d'informations ou se déplacer horizontallement dans le réseau de la victime
  - Utilise dans la plupart des cas les ports 80 et 443

Types de logiciels malveillants



- **RATs (Remote Administration Tools)**
  - Le serveur se connecte au client et attend des commandes de lui
  - Ex: Poison Ivy

Types de logiciels malveillants

- **Botnets Vs RATs**
  - Les Botnet contrôlent beaucoup de machines alors que les RATs n'en contrôlent que quelques unes
  - Les botnets sont contrôlés simultanément
  - RATs sont utilisées dans des attaques ciblées alors que les botnets sont utilisés dans des attaques de masse tels que le déni de service

Types de logiciels malveillants

- **Porte dérobée**
  - Une porte dérobée *(backdoor) est un logiciel de communication caché, installé par* exemple par un virus ou par un cheval de Troie, qui donne à un agresseur extérieur un accès à l'ordinateur victime, par le réseau.

Types de logiciels malveillants

- **Bombe logique**
  - Une bombe logique est une fonction, cachée dans un programme en apparence honnête, utile ou agréable, qui se déclenchera à **retardement**, lorsque sera atteinte une **certaine date**, ou lorsque surviendra un **certain événement**.

Types de logiciels malveillants

- **Logiciel espion**
  - Un logiciel espion, comme son nom l'indique, collecte à l'insu de l'utilisateur légitime des informations au sein du système où il est installé, et les communique à un agent extérieur, par exemple au moyen d'une porte dérobée.

  - Une variété particulièrement toxique de logiciel espion est
    - le ***keylogger*** *(espion* dactylographique), qui enregistre fidèlement tout ce que l'utilisateur tape sur son clavier et le transmet à son honorable correspondant ; il capte ainsi notamment identifiants, mots de passe et codes secrets.
    - Un programme qui dump des informations sauvegardées comme le hash des mots de passe

Types de logiciels malveillants

- ***Ransomware***
  - Un ***ransomware***, **rançongiciel** ou **logiciel de rançon** est un logiciel malveillant qui prend en otage des données personnelles. Pour ce faire, un rançongiciel chiffre des données personnelles puis demande à leur propriétaire d'envoyer de l'argent en échange de la clé qui permettra de les déchiffrer.

Types de logiciels malveillants

- ***Rootkit***
  - C'est un malware qui mofidie une fonctionnalité de l'**OS** afin de **cacher son existence**
  - Cacher un fichier, un processus, une connexion réseau,…
  - Ceci afin de rendre sa détection plus compliquée
  - Rootkit en mode utilisateur Vs Rootkit en mode noyau

Analyse d’un code malveillant

Réponse à une attaque

Cas réel

- L'administrateur système d'une entreprise possédant 10 agences trouve un malware sur un des postes de travail de l'entreprise
- Il fait appelle à un de ses techniciens pour nettoyer le poste et éventuellement le réinstaller

Doit-il considéré l'incident clos ?

Réponse à une attaque

Après la découverte d'un code malveillant, on doit se poser les questions suivantes

- L'attaquant a-t-il installé un rootkit ou un *Trojan* sur votre système ?
- L'attaquant est-il vraiment parti?
- Qu'a-t-il volé ou ajouté?
- Comment l'attaquant est-il entré ?
  - Analyser les origines de l'intrusion

Breach clean-up cost LinkedIn nearly $1 million, another $2-3 million in upgrades

Summary: *LinkedIn executives reveal on quarterly earnings call just what the June theft of 6.5 million passwords cost the company in forensic work and on-going security updates.*

By John Fontana for Identity Matters | August 3, 2012 -- 17:10 GMT (10:10 PDT)

Follow @johnfontana

LinkedIn spent nearly $1 million investigating and unraveling the theft of 6.5 million passwords in June and plans to spend up to $3 million more updating security on its social networking site.

Le But de l'analyse de codes malveillants

Disséquer le malware pour comprendre

- Son fonctionnement
- Comment l'identifier
- Comment l'éliminer

C'est l'étape critique d'une réponse à un incident (attaque)

Le But de l'analyse de codes malveillants

- Comprendre ce qui s'est exactement passé
- Identifier tous les fichiers et toutes les machines infectés
- Trouver les mesures adéquates pour contenir les dommages
- 
  Trouver les signatures de l'intrusion qui vont alimenter les anti-virus (YARA) et les IDS (Intrusion Detection System)

Le But de l'analyse de codes malveillants

-

Signatures

Host-based signature

- Identifie un fichier ou une clef de registre sur la machine de la victime
- Se focalise sur ce que le malware fait sur le système et non sur le malware lui-même
  - Différente de la signature des anti-virus

Network signature

- Détecte la malware en analysant le trafic réseau

Sujets aux faux positifs

Techniques d’analyse de codes malveillants

Statique vs Analyse Dynamique

Analyse statique

- Examiner le malware sans l'exécuter
- Outils: VirusTotal, strings, un désassembleur comme IDA Pro et radare2, readelf, etc.

Analyse dynamique

- Exécuter le malware et surveiller ses effets sur la machine (changement sur la base de registres, création de fichiers,..) et sur le réseau (tentatives de connexion à des serveurs distants,…)
- Utilisation de machines virtuelles et réaliser des snapshots
- Outils: RegShot, Process Monitor, Process Hacker, virtualbox, debugger comme Immunity debugger
- Analyse de la RAM: Mandant Redline et Volatility

Analyse Basique

Analyse statique basique

- Analyser le malware sans regarder son code
- Outils: VirusTotal, strings, readelf
- Facile et rapide mais s'avère inefficace contre des malwares complexes et ne permet pas d'identifier tous les comportements du malware

Analyse dynamique basique

- N'est pas efficace contre tous les types de malwares qui notamment intègrent des mécanismes anti-virtualisation
- Facile mais nécessite un environnement de test sécurisé

Analyse avancée

Analyse statique avancée

- Reverse-engineering avec un désassembleur
- Complexe et nécessite une connaissance du langage assembleur

Analyse dynamique avancée

- Exécuter le malware dans un debugger
- Examiner l’état interne d’un malware en exécution

Règles générales lors de l'analyse d'un malware

Règles générales lors de l'analyse d'un malware

Ne pas essayer de comprendre tous les détails

- Vous n'avez pas besoin de comprendre 100% du code
- Il faut se concentrer sur les fonctionnalités clefs

Essayez plusieurs outils

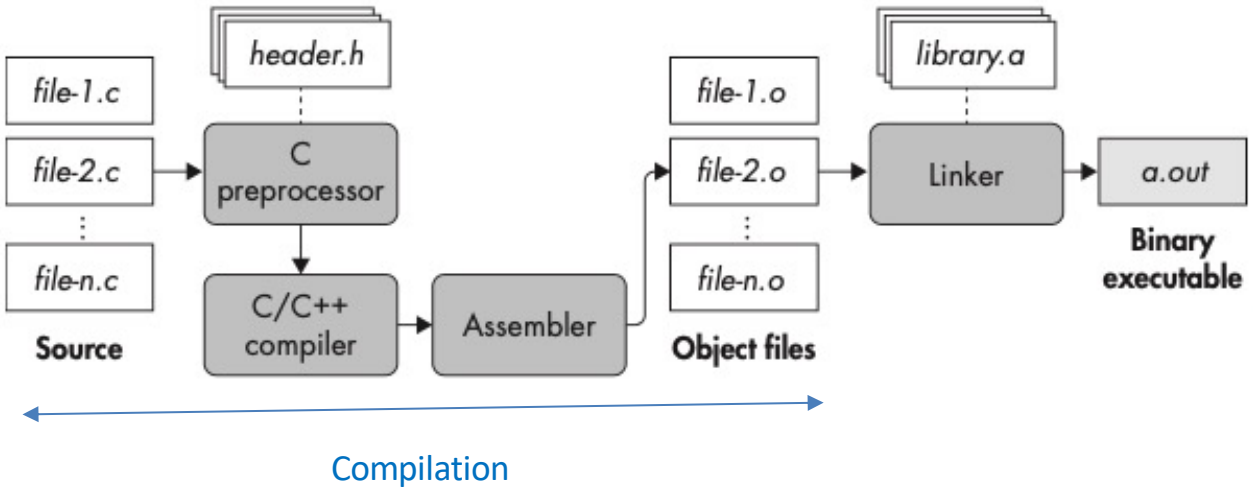
Structure d’un Programme

Chaîne de compilation

```
gcc compilation_example.c
```

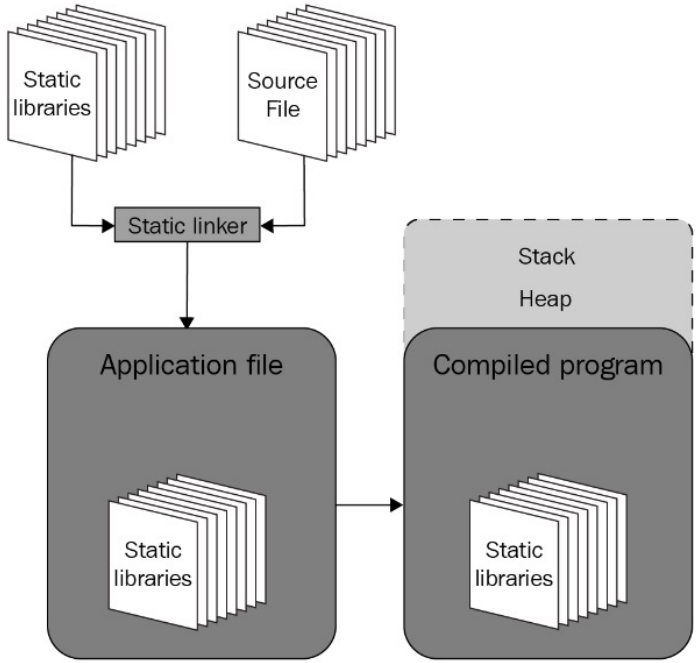
```c
#include <stdio.h>
#define FORMAT_STRING "%s"
#define MESSAGE "Hello, world!\n"

int main(int argc, char *argv[]) {
    printf(FORMAT_STRING, MESSAGE);
    return 0;
}
```



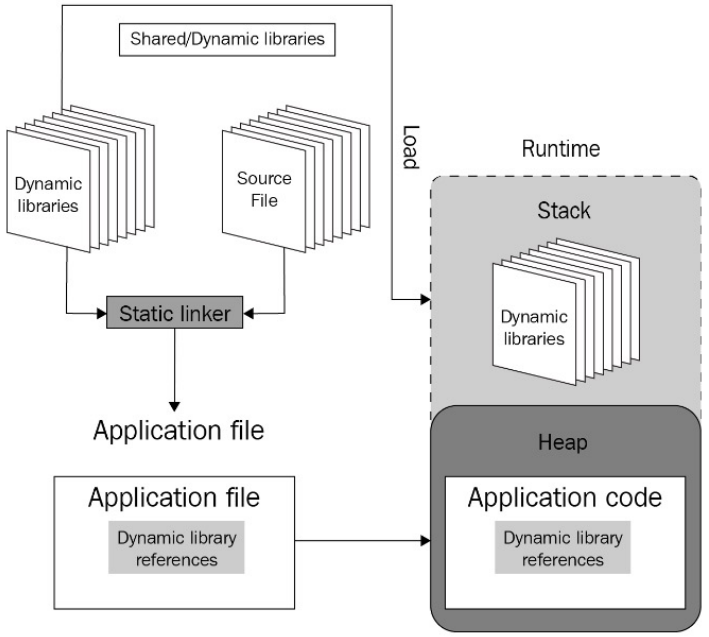
Edition de liens statique

- Rarement utilisée par les exécutables Windows
- Courante dans Unix et Linux
- Tout le code de la librairie est copié dans l'exécutable principal
- Produit des exécutables de taille importante

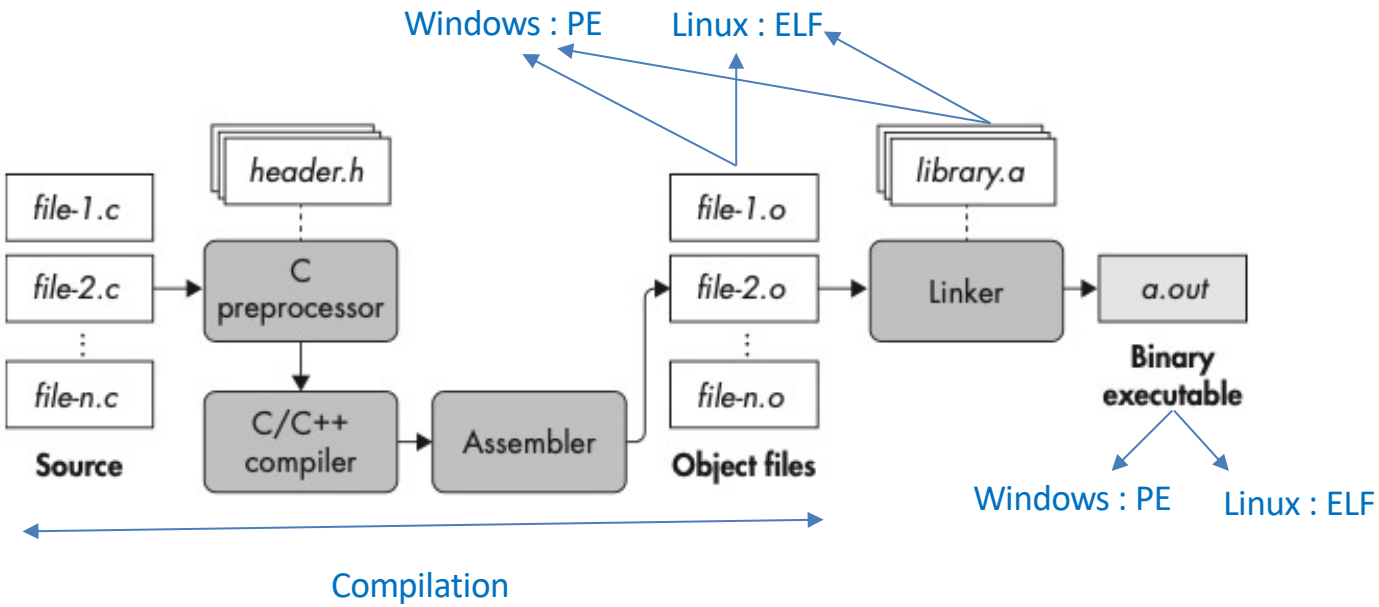


Edition de liens dynamique

- La méthode la plus utilisée
- L'OS recherche les librairies nécessaires au chargement du programme principal



Chaîne de compilation



Format ELF

Un programme ELF est composé :

- D'un entête général
- D'entêtes de sections
- De sections:

.init contient **du code exécutable** réalisant certaines initialisations et s'exécutant avant tout autre code

.fini est analogue à .init mais s'exécute après le main

.text **section de code** où le main et les fonctions du programme résident

.rodata **section de données** qui est en lecture seule et destinée au stockage des constantes du programme

.data section de données en écriture et contient les variables initialisées

.bss section de données en écriture et contient les variables initialisées

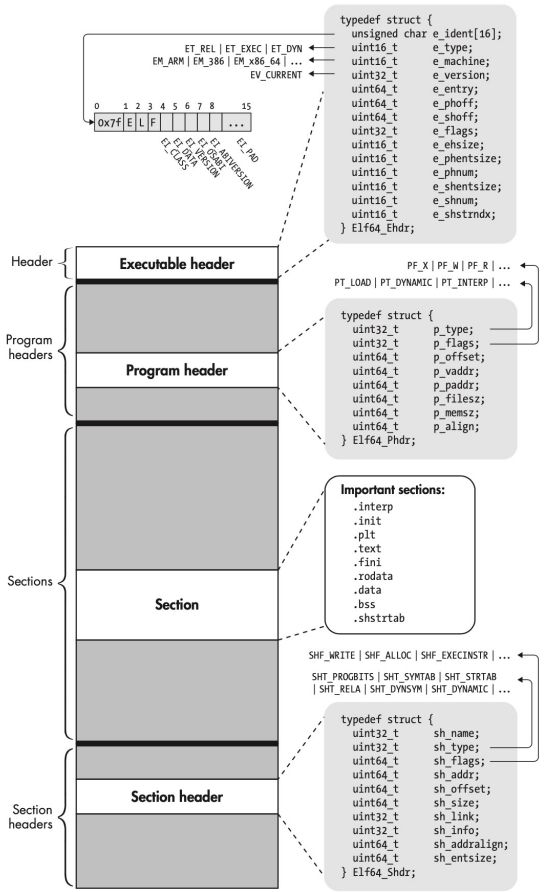
.interp : Contient le chemin vers l'éditeur de lien

rel.* et .rela.* : tables de **relocation** de **symboles**

.got *et* ***.got.plt*** : On reparlera dans la section librairies partagées

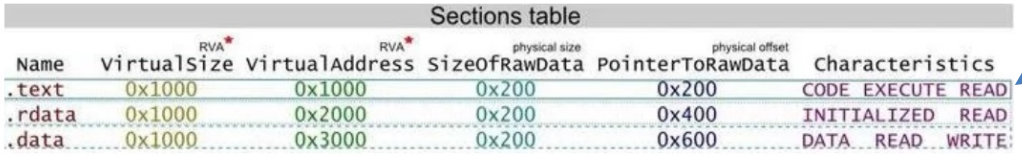
.plt : On reparlera dans la section librairies partagées

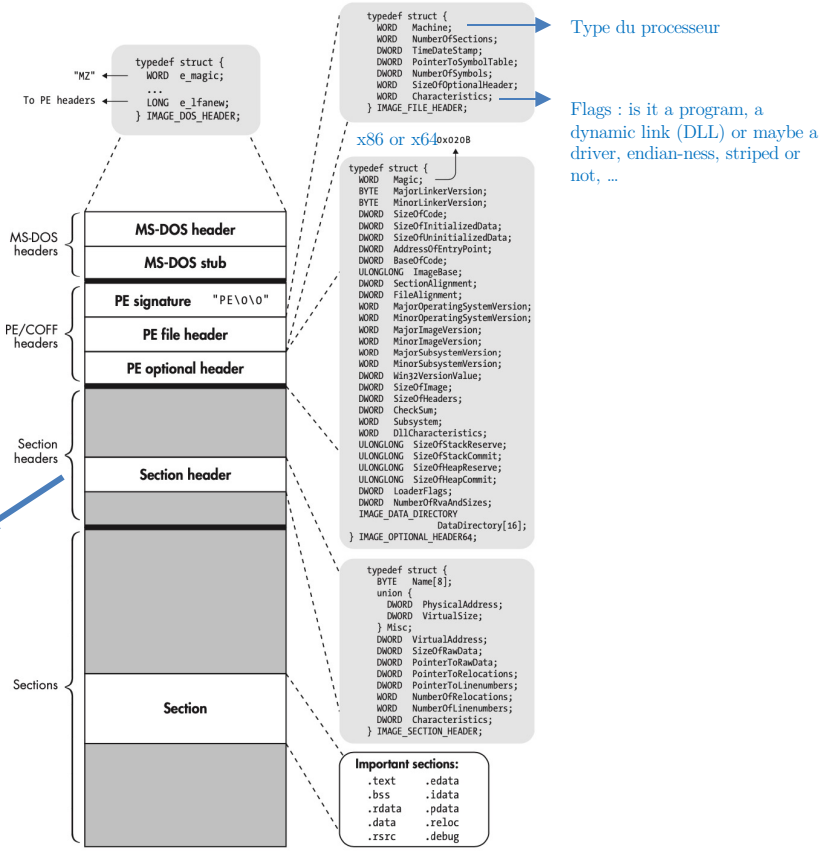
- D'entêtes de programmes



PE (Portable Executable) - Format

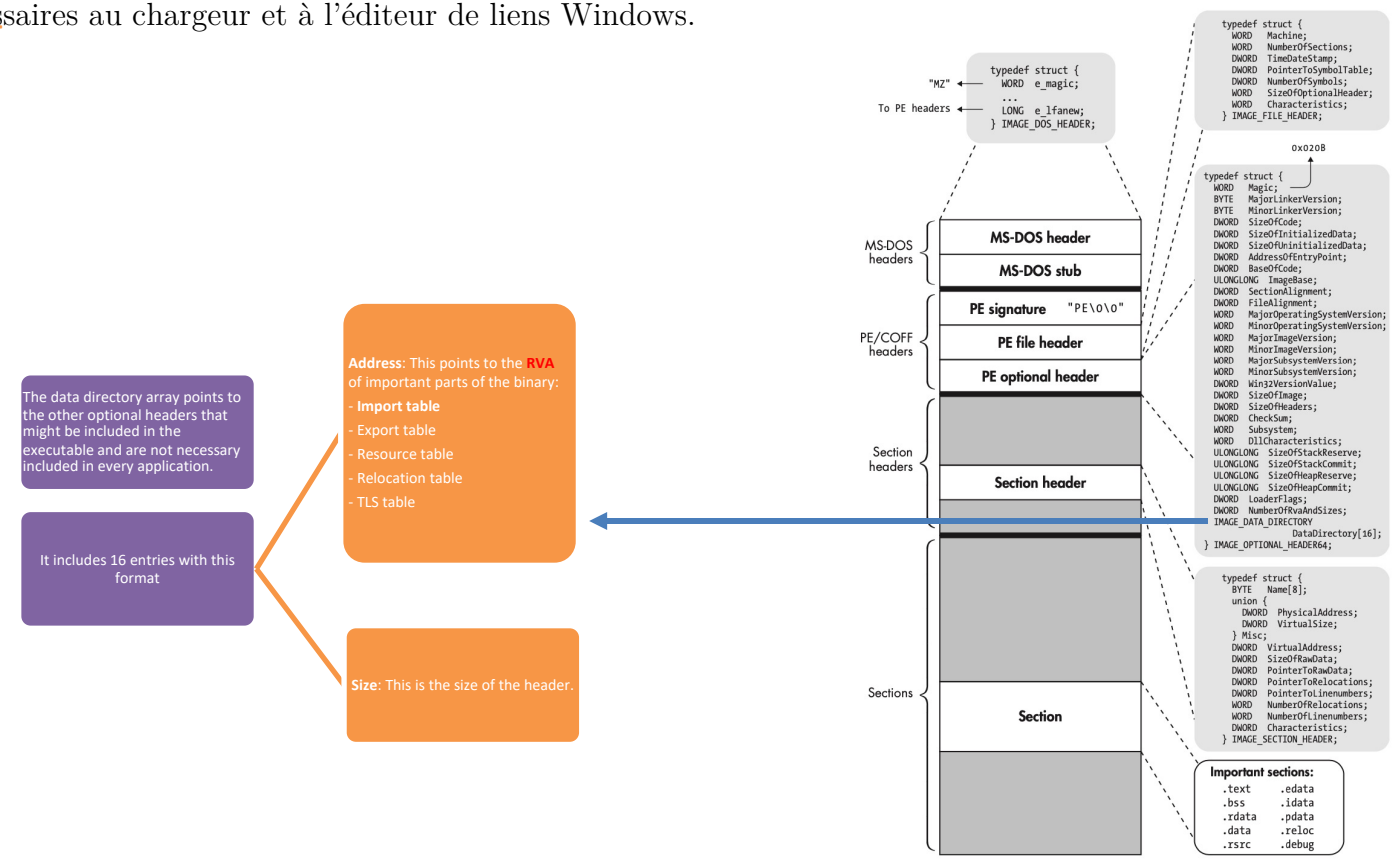
- Format Windows pour les fichiers exécutables, codes objets, DLLs, les dumps du noyau Windows
- C'est une structure de données contenant les informations nécessaires au chargeur et à l'éditeur de liens Windows.





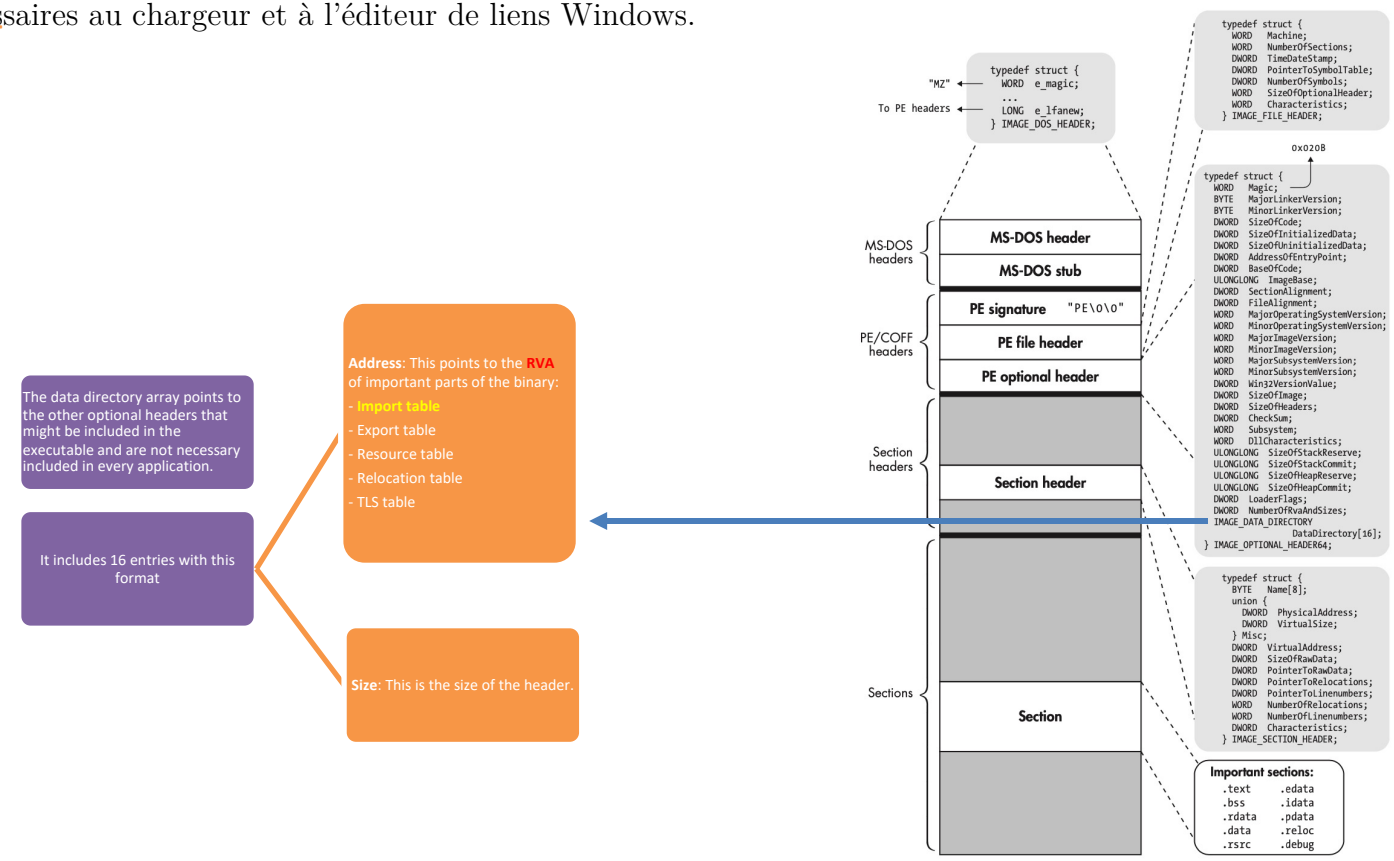
PE (Portable Executable) - Format

- Format Windows pour les fichiers exécutables, codes objets, DLLs, les dumps du noyau Windows
- C'est une structure de données contenant les informations nécessaires au chargeur et à l'éditeur de liens Windows.

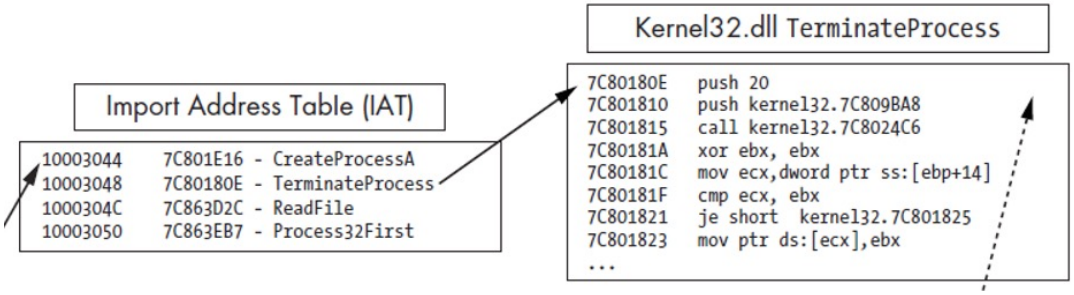


PE (Portable Executable) - Format

- Format Windows pour les fichiers exécutables, codes objets, DLLs, les dumps du noyau Windows
- C'est une structure de données contenant les informations nécessaires au chargeur et à l'éditeur de liens Windows.



PE (Portable Executable) - IAT Import table



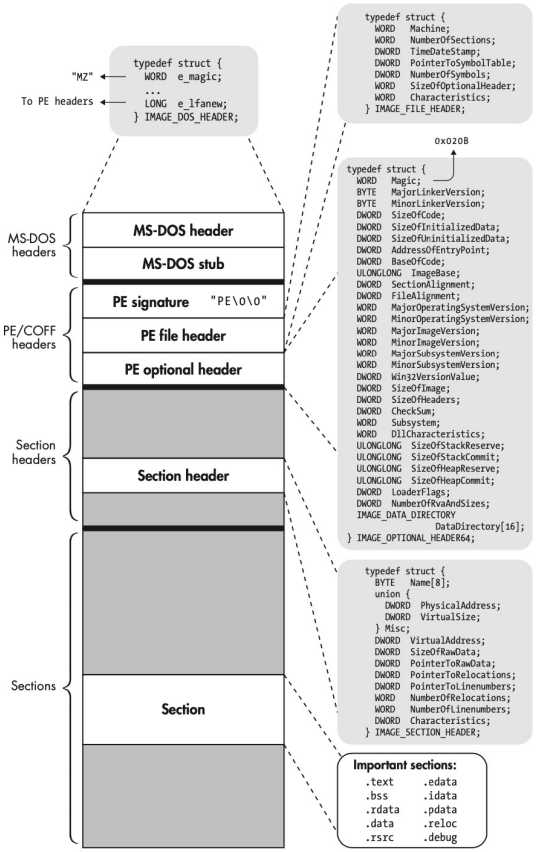
PE (Portable Executable) - Format

- Format Windows pour les fichiers exécutables, codes objets, DLLs, les dumps du noyau Windows
- C'est une structure de données contenant les informations nécessaires au chargeur et à l'éditeur de liens Windows.

La section **.rsrc** : sauvegarde les ressources (Strings, icons, and menus,…) utilisées par l'exécutable

La section **.reloc** contient les adresses des symboles à recalculer

La section **.rdata** -- imports & exports



Sections table

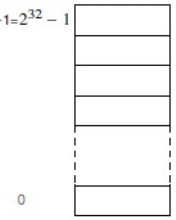
Name	VirtualSize (RVA*)	VirtualAddress (RVA*)	SizeOfRawData (physical size)	PointerToRawData (physical offset)	Characteristics
.text	0x1000	0x1000	0x200	0x200	CODE EXECUTE READ
.rdata	0x1000	0x2000	0x200	0x400	INITIALIZED READ
.data	0x1000	0x3000	0x200	0x600	DATA READ WRITE

PE (Portable Executable) - Section Table

- Following the 16 entries of the data directory array come the section headers.
- This is a list of headers with each header representing a section of the PE file.
- The number of headers in total is the exact number stored in the NumberOfSections field in FileHeader.

Structure d'un programme en mémoire centrale

Un programme sur un adressage sur 32 bits voit la mémoire de l'adresse 0 à $2^{32} - 1$, c-à-d $4Go$:



Les segments :

- chaque section possède des droits d'accès différents : `read/write/execute`
- **Code** : les instructions du programme (appelé parfois «*Text*») ;
- **Data** : les variables globales initialisées ;
- **Bss** : les variables globales non initialisées (elles seront initialisées à 0) ;
- **Heap** : mémoire retournée lors d'allocation dynamique avec les fonctions «`malloc/calloc/new`» : *elle croît vers le haut* ;
- **Stack** : stocke les variables locales et les adresses de retour : *elle croît vers le bas* ;
- **Shared libraries** : les bibliothèques partagées, c-à-d le code des fonctions partagées par plusieurs processus (fonctions d'entrée/sortie, mathématiques, *etc.*).

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int our_init_data = 30;
int our_noinit_data;

void our_prints(void)
{
  int our_local_data = 1;
  char c;
  int *our_dynamic_data = (int *)calloc(1,sizeof(int));

  printf("Pid of the process is = %d\n", getpid());
  printf("Addresses which fall into:\n");
  printf("1) Data   segment = %p\n", &our_init_data);
  printf("2) BSS    segment = %p\n", &our_noinit_data);
  printf("3) Code   segment = %p\n", &our_prints);
  printf("4) Stack segment = %p\n", &our_local_data);
  printf("5) Heap segment = %p\n", our_dynamic_data);
  scanf("%c", &c);
}
```

```c
int main()
{
  our_prints();
  return 0;
}
```

xterm

```
$ ./segments
Pid of the process is = 4697
Addresses which fall into:
 1) Data   segment = 0x804a020
 2) BSS    segment = 0x804a02c
 3) Code   segment = 0x8048494
 4) Stack segment = 0xbfe47454
 5) Heap segment = 0x8d76008
```

xterm

```
$ more /proc/4697/maps
08048000-08049000 r-xp 00000000 08:01 424814   ./segments
08049000-0804a000 r--p 00000000 08:01 424814   ./segments
0804a000-0804b000 rw-p 00001000 08:01 424814   ./segments
08d76000-08d97000 rw-p 00000000 00:00 0        [heap]
bfe28000-bfe49000 rw-p 00000000 00:00 0        [stack]
```

xterm

```
$ nm -f sysv segments
Symbols from segments:
Name                  Value   Class  Type     Size     Section
main                |08048558|   T  |     FUNC|00000012||.text
our_init_data       |0804a020|   D  |   OBJECT|00000004||.data
our_noinit_data     |0804a02c|   B  |   OBJECT|00000004||.bss
our_prints          |08048494|   T  |     FUNC|000000c4||.text
```

On remarque que sur la sortie du fichier «`maps`» :

- *le segment `code` possède les droits d'exécution ;*
- *les segments `bss`, `data` possèdent les droits de lecture/écriture mais* ***pas d'exécution.***

Structure d'un programme en mémoire centrale

Exo
(Trouvez
la section
de chaque
variable)

```
/* memory.c */
int index = 5;
char * str;
int nothing;
void funct1(int c){
    int i=c;
    str = (char*) malloc (10 * sizeof (char));
    strncpy(str, "abcde", 5);
}
void main (){
    funct1(1);
}
```

Analyse Statique Basique

Techniques

Scan d’anti-virus

Calcul de Hashes (empreinte)

Extraire les chaînes de caractères utilisées par le code du malware

Scan d’anti-virus

VirusTotal

- Anti-virus en ligne utilisant la base de signatures de plusieurs anti-virus du marché
- Le Malware peut facilement changer sa signature
- VirusTotal est un bon moyen, mais peut alerter l'attaquant

Calcul du Hash d’un malware

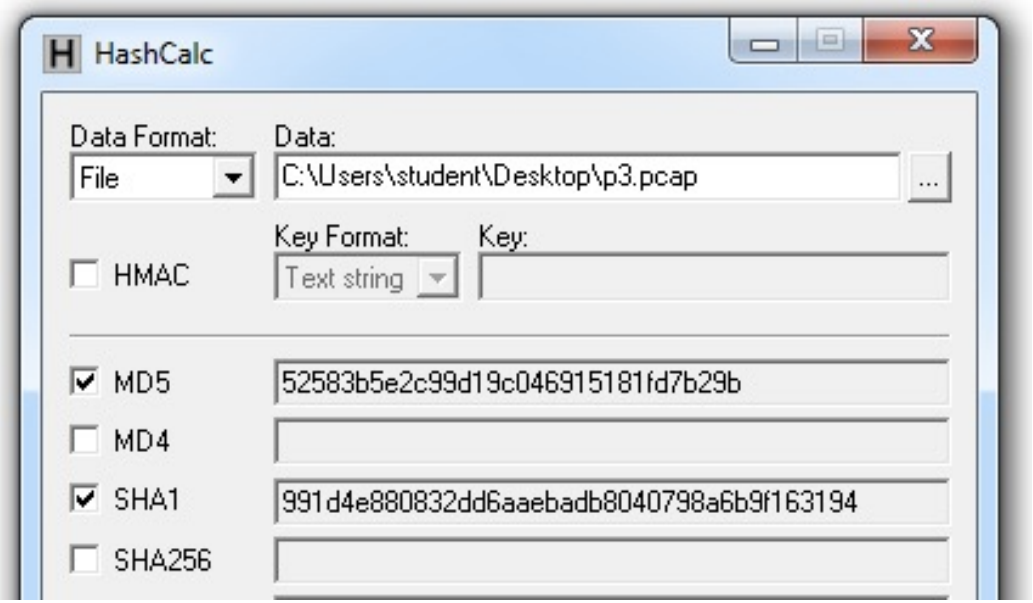
Calcul du Hash d'un malware

Calcul l'empreinte d'un fichier de taille quelconque.
L'empreinte produite est de taille fixe

Identifie de manière unique un fichier, même si on pratique il existe une faible probabilité de collision

MD5 or SHA-1

Calcul du Hash d'un malware - Comment



echo "fichier à hasher" | openssl md5

Calcul du Hash d'un malware - Utilité

Partager l'empreinte avec d'autres analystes pour identifier le malware

Rechercher l'empreinte sur internet pour vérifier si une autre personne a déjà identifié le malware

Recherche de chaînes de caractères

Recherche de chaînes de caractères

-

Pourquoi ?

- Adresses IP
- URLs
- Noms de domaine

- Noms de fonctions (API Windows)
- Noms de librairies
- Noms de fichiers / dossiers
- Noms de clés de registre
- Noms de mutex

Recherche de chaînes de caractères

-

Comment ?

Commande strings ou avec radare2 (Native sur Linux, existe aussi sur Windows)

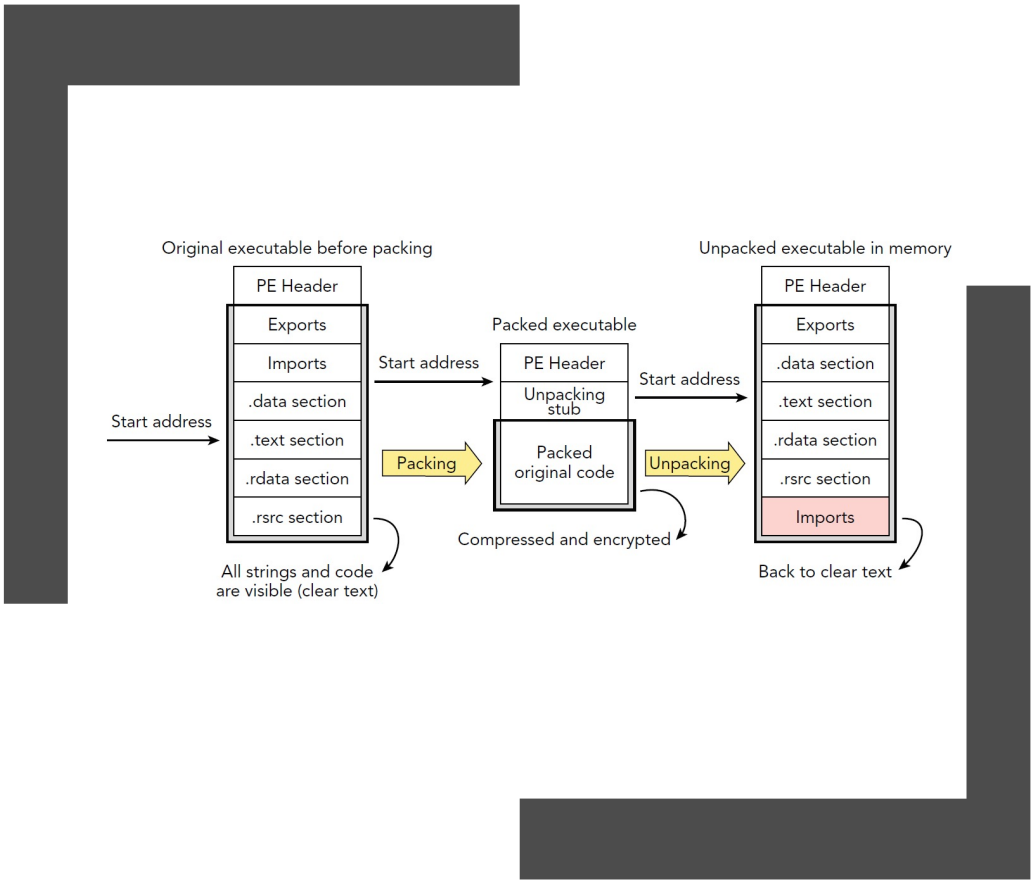
Recherche toutes les chaînes de caractères d’un fichier de plus de 3 caractères de long

Recherche de chaînes de caractères Exemple

```
C:>strings bp6.ex_
VP3
VW3
t$@
D$4
99.124.22.1 4
e-@
GetLayout 1
GDI32.DLL 3
SetLayout 2
M}C
Mail system DLL is invalid.!Send Mail failed to
send message. 5
```

- Les items en gras peuvent être ignorés GetLayout et SetLayout sont des fonctions de l'API Windows
- GDI32.DLL est une librairie partagée (Dynamic Link Library)

Recherche de chaînes de caractères dans des “Packing Files”



- Le code est compressé comme un fichier Zip
- Ceci rend les chaînes de caractères et les instructions non lisible
- Tout ce qu’on peut voir est un bout de code qui décompresse le code malveillant lors de l’exécution

Recherche de chaînes de caractères
Packing Files (Détection)

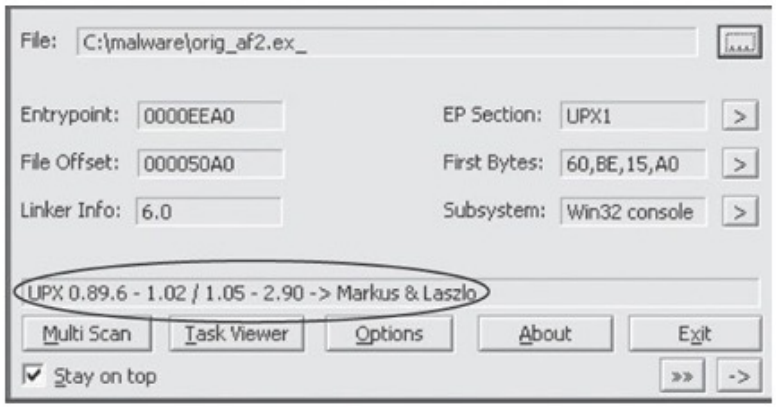


Figure 2-5. The PEiD program

Recherche de chaînes de caractères - Packing Files

```
root@kali:~/126# strings chatty | wc
   1962    4498   33817
root@kali:~/126# strings chatty-packed | wc
   3950    4290   23623
root@kali:~/126#
```

root@kali: ~/126

File Edit View Search Terminal Help

```
root@kali:~/126# cat chatty.c
#include <stdio.h>
main()
{
char name[10];
printf("This program contains readable strings\n");
printf("Enter your name: ");
scanf("%s", name);
printf("Hello %s\n", name);
}

root@kali:~/126# gcc -static chatty.c -o chatty
root@kali:~/126# upx -o chatty-packed chatty
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2011
UPX 3.08        Markus Oberhumer, Laszlo Molnar & John Reiser   Dec 12th 2011

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    592800 ->    272588   45.98%  linux/elf386   chatty-packed

Packed 1 file.
root@kali:~/126# ls -l
total 852
-rwxr-xr-x 1 root root 592800 Aug 16 20:34 chatty
-rw-r--r-- 1 root root    174 Aug 16 20:27 chatty.c
-rwxr-xr-x 1 root root 272588 Aug 16 20:34 chatty-packed
root@kali:~/126#
```

KALI LINUX

Librairies

Imports

Fonctions utilisées par un programme et qui sont stockées dans un autre programme comme une librairie

Connectées à l'EXE principal par l'édition de liens (**Linking)**

Il existe trois types d'édition de liens

- **Statically (statique, à la compilation)**
- **Dynamiquement**
- **At Runtime (à l'éxécution)**

Edition de liens à l'exécution

Pas souvent utilisée dans les programmes classiques

Courante dans les malwares, surtout dans le malware packé

L'édition de liens est réalisée au besoin et pas nécessairement au démarrage du programme

Par utilisation des fonctions **LoadLibrary** et **GetProcAddress**

Les Libraries dans l'analyse de malware

Le fichier PE liste toutes les librairies et les fonctions chargées de manière statique et dynamique

Leur noms nous renseignent sur les fonctionnalités et les objectifs du malware

Ex. URLDownloadToFile nous indique que le programme peut télécharger quelque chose

Quelques DLLs dans l'API Windows

Table 2-1. Common DLLs

DLL	Description
Kernel32.dll	This is a very common DLL that contains core functionality, such as access and manipulation of memory, files, and hardware.
Advapi32.dll	This DLL provides access to advanced core Windows components such as the Service Manager and Registry.
User32.dll	This DLL contains all the user-interface components, such as buttons, scroll bars, and components for controlling and responding to user actions.
Gdi32.dll	This DLL contains functions for displaying and manipulating graphics.

Quelques DLLs dans l'API Windows

Ntdll.dll	This DLL is the interface to the Windows kernel. Executables generally do not import this file directly, although it is always imported indirectly by *Kernel32.dll*. If an executable imports this file, it means that the author intended to use functionality not normally available to Windows programs. Some tasks, such as hiding functionality or manipulating processes, will use this interface.
WSock32.dll and *Ws2_32.dll*	These are networking DLLs. A program that accesses either of these most likely connects to a network or performs network-related tasks.
Wininet.dll	This DLL contains higher-level networking functions that implement protocols such as FTP, HTTP, and NTP.

Import et Exports

DLLs **exportent** des fonctions

EXEs **importent** des fonctions

IAT et EAT du fichier PE

Example: Keylogger

Importe la librairie User32.dll et utilise la fonction **SetWindowsHookEx** qui est une méthode très utilisée par les keyloggers pour recevoir les entrés du clavier

Il exporte la fonction (callback) **LowLevelKeyboardProc** et **LowLevelMouseProc** pour envoyer les données ailleurs

Attention aux programme packé

- Très peu de fonctions

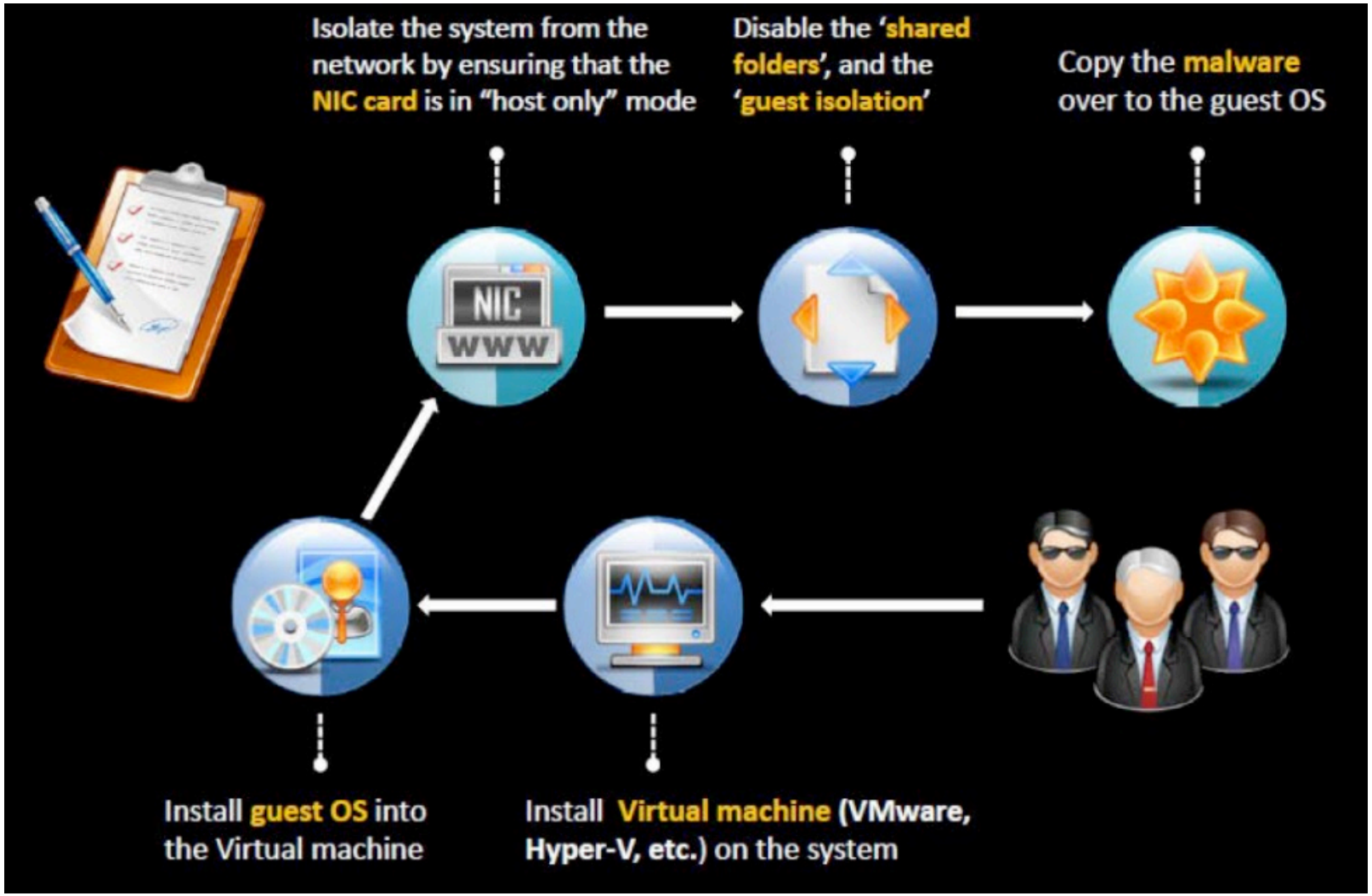
Table 2-3. DLLs and Functions Imported from PackedProgram.exe

Kernel32.dll	User32.dll
GetModuleHandleA	MessageBoxA
LoadLibraryA	
GetProcAddress	
ExitProcess	
VirtualAlloc	
VirtualFree	

Fiche - TP 1

Lien du LAB

Analyse basique dynamique



Fiche - TP 2

Lien du LAB