Introduction à la virologie informatique: principes, attaques et défenses

Master 2 CSI - Université de Bordeaux 1

Renaud Tabary: `renaud.tabary@labri.fr`

12 novembre 2009

Plan

Introduction

Definition

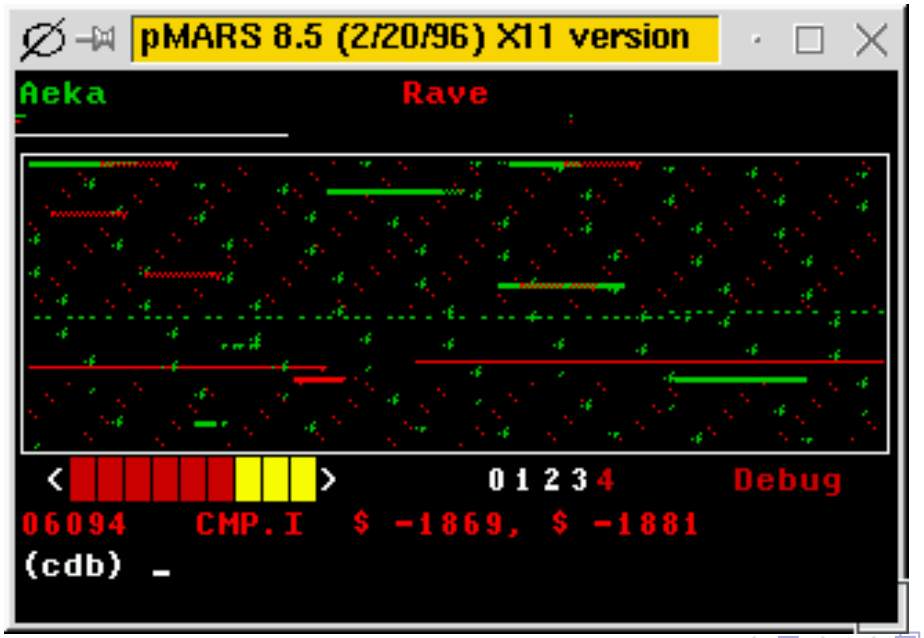
"A virus is a program that is able to infect other programs by modifying them to include a possibly evolved copy of itself."

F.Cohen

- Virus !=malware
- Nous ne **parlerons pas** des vers (trivial)
- Programmes souvent écrits en assembleur (entre 10 lignes et 20000 lignes de code)
- Grande variété de virus :
  - Différentes cibles
  - Différentes charges malicieuses
  - Différents vecteurs de propagations

Corewar

- Un exemple simple
  - Machine virtuelle minimaliste
  - Deux programmes s'affrontent pour *coloniser* la mémoire
  - Le code vit dans la mémoire
- Exemple : corewar



Plan

Pourquoi les étudier ?

- C'est la plus ancienne nuisance informatique
- La **quasi totalité des techniques** des rootkits, vers, trojan et shellcodes **sont issues des virus**
- Un peu plus que de simples `format c :`
  - Théorie des langages
  - Programmation système
  - Reverse engineering et obfuscation
  - et bien d'autres ...
- Le problème de détection d'un virus est **indécidable**
  - Preuve de F.Cohen basée sur l'arrêt d'une machine de Turing

Pourquoi les étudier ? (2)

Parce qu'ils sont dangereux !

- Remet en jeu l'**intégrité** et la disponibilité du Système Informatique :
  - Suppression de fichiers
  - Backdoors
  - Charge réseau (Netsky)
  - Ransomwares (Gpcode)
- Parfois la **confidentialité** :
  - Vols de mots de passe (jeux en ligne, banque en ligne, etc.)
- Augmentation du nombre de postes connectés et de médias amovibles

Pourquoi les étudier ? (3)

Situation actuelle :

- Marché éclaté entre plus de 20 marques (3 leaders : McCafee, Symantec et Trend Micro)
- Marché en progression (+12,5% en 2008)
- Collaboration entre les différents éditeurs
- Plus de 100 nouvelles détections par jour
- ... et toujours **pas d'antivirus parfait** (*cf. Vigard*)

Objectifs de ce cours

- Culture générale du spécialiste en sécurité
- Effacer les idées préconcues
- Tracer un historique rapide de l'évolution de la virologie informatique
  - Un bon aperçu du monde de la sécurité informatique
- Introduction aux technique d'infection et de mutation de code
  - Polymorphisme, métamorphisme, chiffrement de code, anti-émulation
  - Se retrouvent dans tous les malwares actuels
- Appréhender les techniques de détection de virus
  - Signature, émulation, heuristique, code normalization, détection comportementale

Contenu du cours

- Historique de la virologie informatique
  - du point de vue de l'attaquant
  - du point de vue du défenseur
- Quelques notions théoriques
  - Théorie des langages, Chomsky hierarchy
  - Il existe des approches très formelles de la virologie que nous n'aborderons pas (cf. Fillol)
- Revue des principales techniques d'obfuscation
  - chiffrement du code, polymorphisme, métamorphisme, anti-*
- TP sur machine virtuelle Windows
  - écriture d'une routine de chiffrement pour un virus existant
  - écriture de règles de polymorphisme
  - test de détection sur un AV récent

Plan

Le premier virus informatique

- Les virus de boot sont les 1er virus informatique qui se sont propagés
- En 1986 : le virus Brain
- Avantage : indépendant de l'O.S (surtout à l'époque !)
- Démarrage du virus avant l'O.S

Le virus Brain

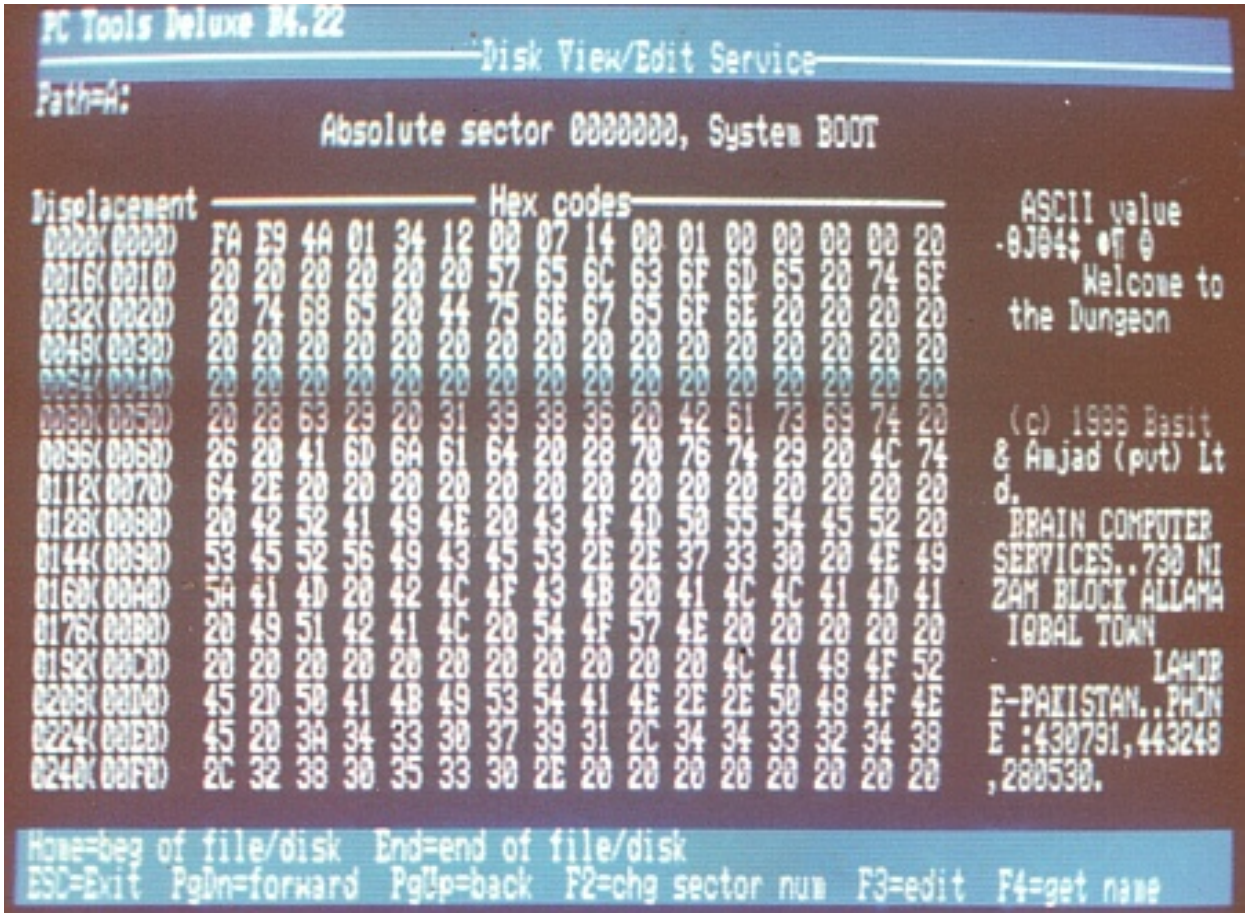
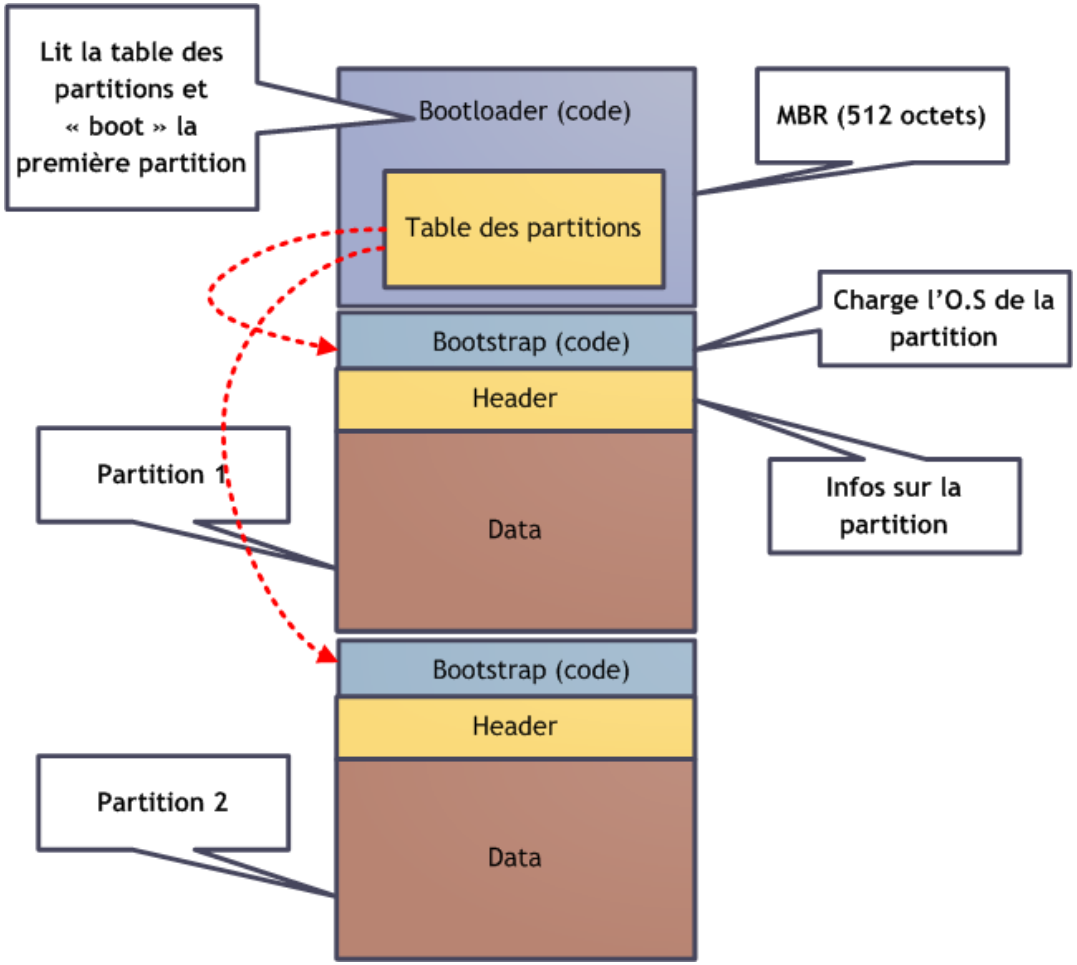


FIGURE: Le virus brain

Fonctionnement

- Il n'est pas possible de charger l'O.S depuis la ROM
- Au démarrage du PC, le premier secteur du disque dur (le MBR) est chargé en mémoire
- Le MBR contient un *bootstrap loader* qui charge la partiton active depuis la table de partitions
- La partition active contient le code de chargement de l'O.S

Le MBR


Lit la table des partitions et « boot » la première partition
Bootloader (code)
Table des partitions
MBR (512 octets)
Charge l'O.S de la partition
Bootstrap (code)
Header
Partition 1
Data
Infos sur la partition
Bootstrap (code)
Header
Partition 2
Data

Les types d'infections du MBR

- Remplacement du bootstrap (*Stoned)*
  - Déplacement de la table de partitions (fin du MBR ou espace libre du disque)
  - Insertion du code à la place du bootstrap
- Remplacement du MBR (*Azura 91)*
  - Pas de sauvegarde de la table de partitions (plus d'espace)
  - Chargement de la partition active par le virus
- Modification de la table de partition (*Starship*)
  - La table de partitions est parsée et modifiée
  - Chargement d'un secteur différent, où est situé le virus

Vecteur de propagation

- A l'époque, les programmes étaient distribués sous forme de disquette bootable
  - Pas vraiment d'OS
- On s'échangeait des programmes en recopiant des disquettes
- Les virus de boot :
  - Se chargeaient en mémoire au démarrage de l'ordinateur
  - Repéraient la routine du BIOS chargée d'écrire sur une disquette (`int 13h`)
  - **Détournaient** cette routine vers du code viral
  - Lorsque cette routine était appelée :
    - Écriture des données demandées
    - ... puis écriture du virus sur le MBR

Propagation des virus MBR

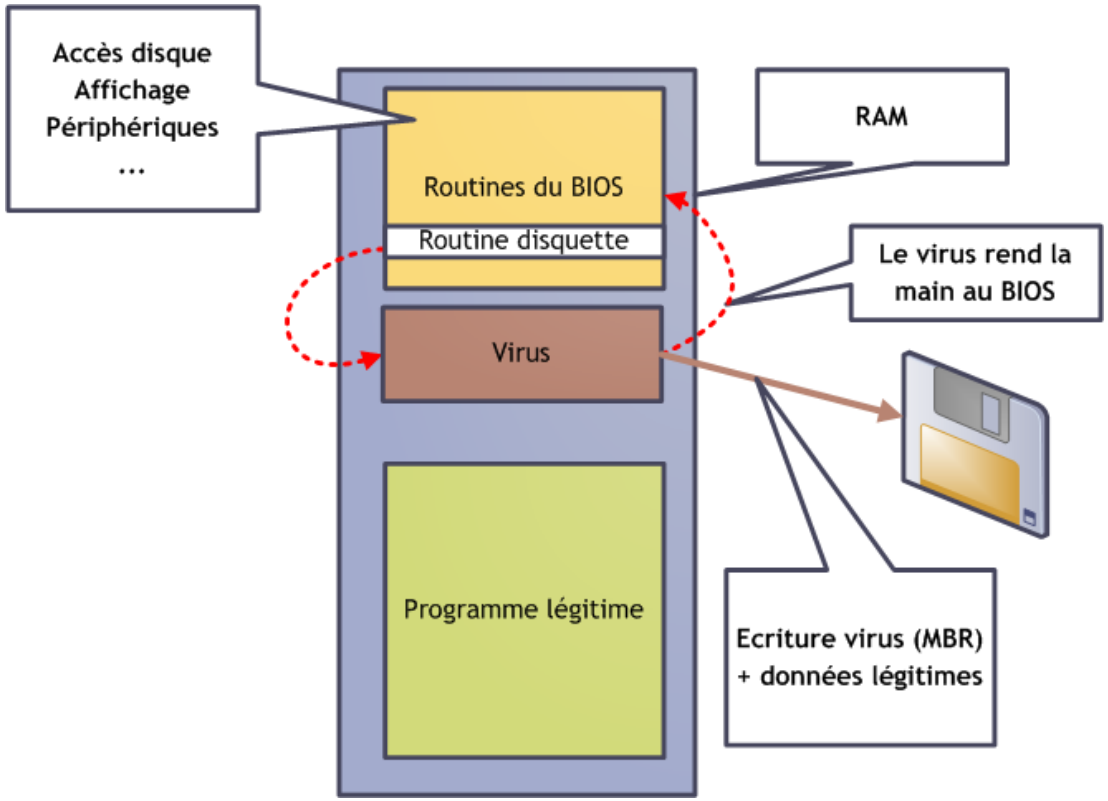


FIGURE: Propagation d'un virus type MBR

Plan

Et maintenant ?

Les virus Autorun sont les descendants actuels des virus de boot

- Sous Windows, lorsqu'un disque amovible est inséré, le fichier `autorun.inf` est examiné

Autorun

```
[autorun]
OPEN=virus.doc.exe
ICON=msword.ico
...
```

- Une fois le virus lancé, il scannera en permanence la présence de disques amovibles
  - Pour y ajouter virus.exe et autorun.inf
  - En fichier caché bien sur
- Encore plus simple qu'avant !
- Et toujours aussi efficace ! (cf. *Seagate*)

Plan

Première génération : les scanners artisanaux

- Pas encore de société d'antivirus
- Souvent, un antivirus différent pour chaque virus
- Apparition de la recherche par signature

```
seg000:7C40 BE 04 00                    mov     si, 4           ; Try it 4 times
seg000:7C40                                                     ;
seg000:7C43
seg000:7C43                             next:                   ; CODE XREF: sub_7C3A+27↓j
seg000:7C43 B8 01 02                    mov     ax, 201h        ; read one sector
seg000:7C46 0E                          push    cs
seg000:7C47 07                          pop     es
seg000:7C48                             assume es:seg000
seg000:7C48 BB 00 02                    mov     bx, 200h        ; to here
seg000:7C4B 33 C9                       xor     cx, cx
seg000:7C4D 8B D1                       mov     dx, cx
seg000:7C4F 41                          inc     cx
seg000:7C50 9C                          pushf
seg000:7C51 2E FF 1E 09 00              call    dword ptr cs:9  ; int 13
seg000:7C56 73 0E                       jnb     short fine
seg000:7C58 33 C0                       xor     ax, ax
seg000:7C5A 9C                          pushf
seg000:7C5B 2E FF 1E 09 00              call    dword ptr cs:9  ; int 13
seg000:7C60 4E                          dec     si
seg000:7C61 75 E0                       jnz     short next
seg000:7C63 EB 35                       jmp     short giveup
```

La recherche par signature

- Architecture de Von Neumann : code $\simeq$ données
- Un programme binaire peut être vu comme un fichier quelconque
- Les premiers antivirus :
  - Lecture du MBR
  - Recherche d'une signature simple (on ne parle même pas de regexp)
  - Si signature trouvée $\longrightarrow$ alerte ou réparation du MBR
- Un antivirus ne ciblait qu'un ou deux virus
- Acceptable, car moins d'une dizaine de nouveaux virus par an

Plan

Les infecteurs de fichiers

Les infecteurs de fichiers exécutables

- Même fonctionnement que les virus biologiques
- Virus informatiques les plus répandus
- De nombreuses cibles :
  - PE-exe, ELF, bat, sh, .py, .vbs
- De nombreuses méthodes d'infection :
  - appenders, prependers, cavity infectors, compagnons

Principe général

Les infecteurs de fichiers :

- Un exécutable est un fichier standard avec un format défini
  - Format PE : `http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx`
  - Format ELF : `http://www.asm-x86.fr/le-format-elf_a8`
- Il est tout à fait légal de créer/modifier un fichier exécutable
  - `gcc`, `patch`, `ld`
- Il est ainsi possible d'**ajouter** et de **modifier** le code d'un programme

Infection d'un exécutable

- Comment infecter un fichier exécutable ?

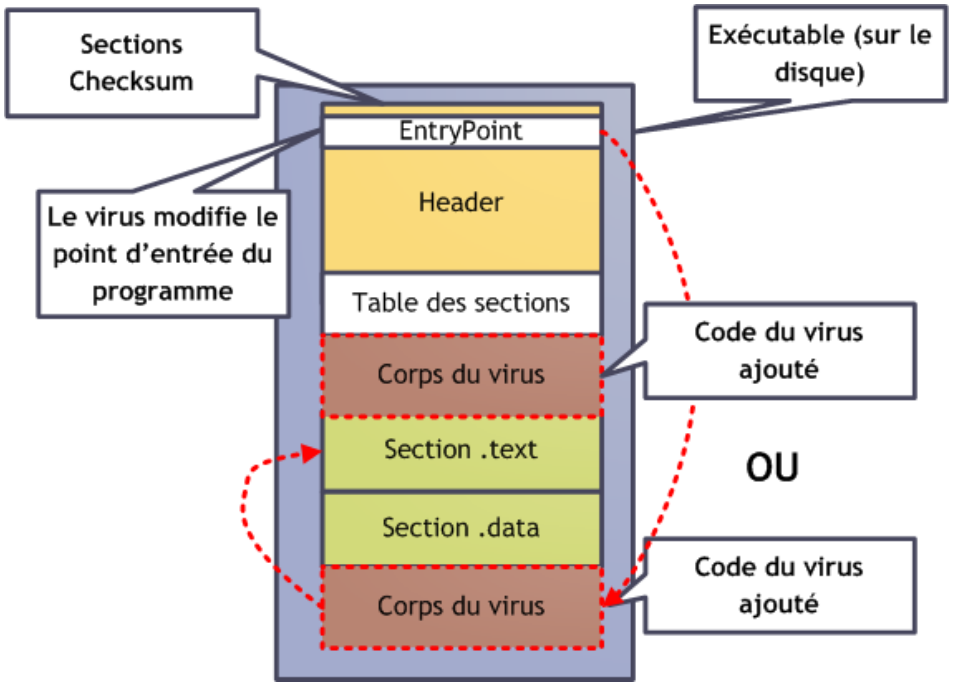


FIGURE: Infection d'un exécutable

Spécificités

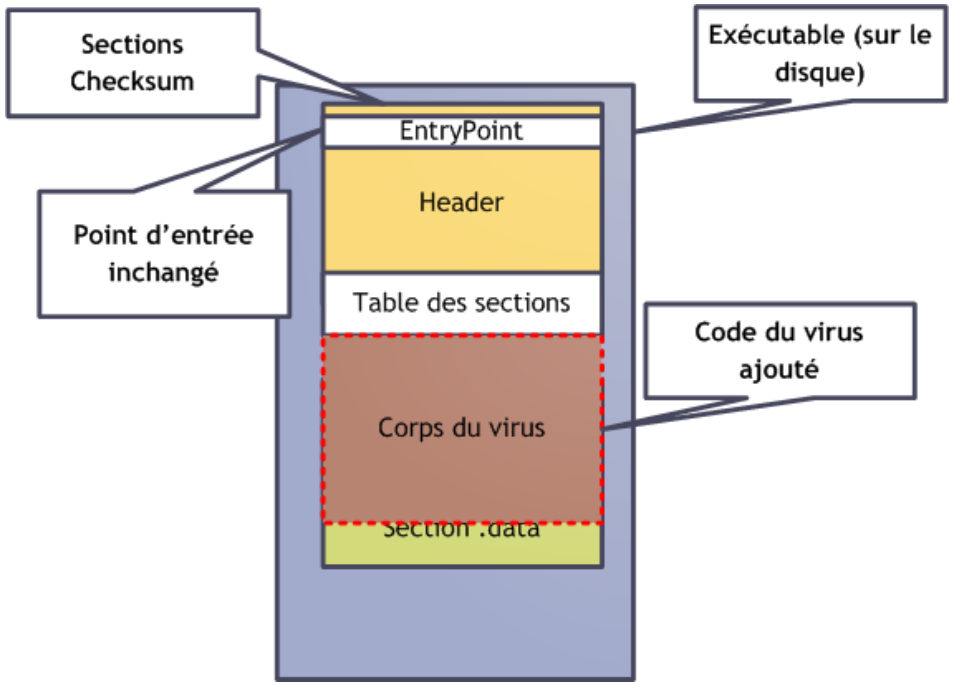
- Le code du virus doit être *position independant*
  - Difficile de savoir où l'exécutable sera chargé (ASLR ...)
- Il faut modifier l'exécutable hôte pour y insérer le virus
  - Conserver la cohérence de la structure du fichier (checksum, en-têtes, ...)
- *Détournement* du flot de contrôle au sein de l'exécutable
  - Modification de l'EP, per process residency, etc.
- Plusieurs générations sur un même système hôte (proche du virus biologique)
  - Possibilité de *mutation* (polymorphisme, métamorphisme, ...)

Plan

Les virus parasites

Overwriting infectors

- Virus déstructif
- Le corps du virus est écrit **sur** le fichier hôte
- Le virus, une fois exécuté, ne rendra pas la main
- Premiers infecteurs d'exes



Les virus parasites (2)

Avantages

- Extrêmement simples
  - Infecte tous types de fichiers
- Pas besoin de PIC
  - C'est une copie d'exécutable

Inconvénients

- Détectable par l'utilisateur
- Besoin de plus ?

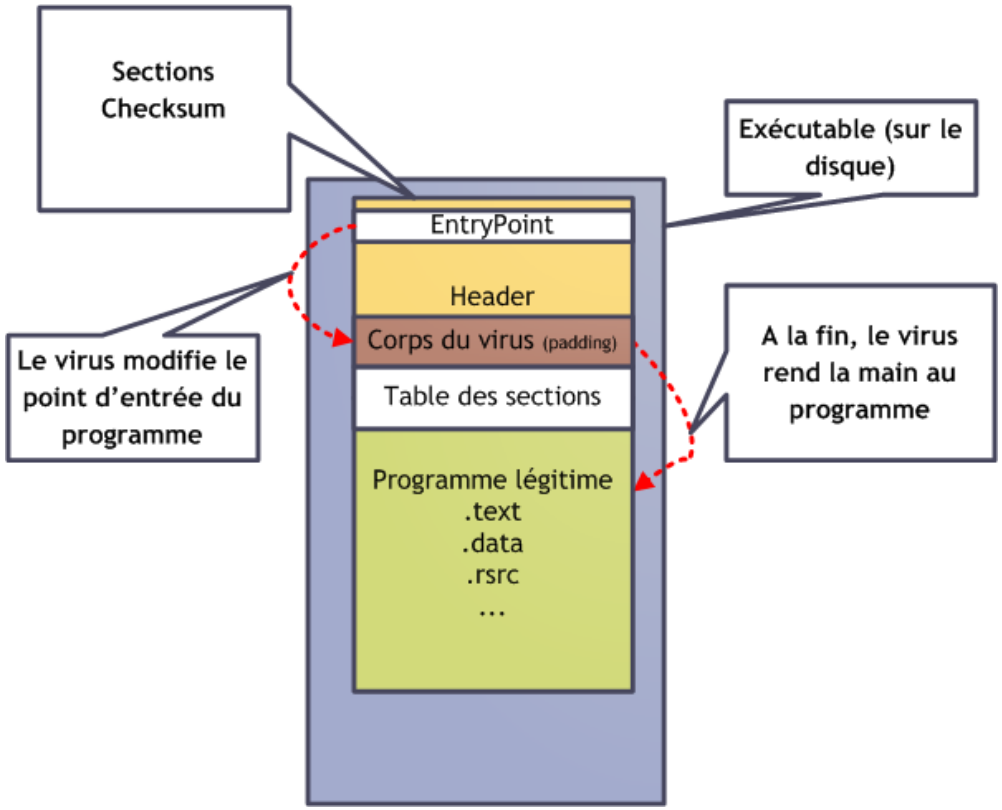
Exemple

Les premiers virus DOS

Les infecteurs de header

Header infectors

- Code du virus très court
- Inséré après les header de l'exécutable (padding)
- Modification du point d'entrée



Les infecteurs de header (2)

Avantages

- Pas de modification de la taille du fichier
  - Contre les whitelists
- Pas de création/modification des sections
  - Contre les heuristiques les plus grossières

Inconvénients

- Corps du virus nécessairement très court
- Pas toujours de padding (XP, Vista)

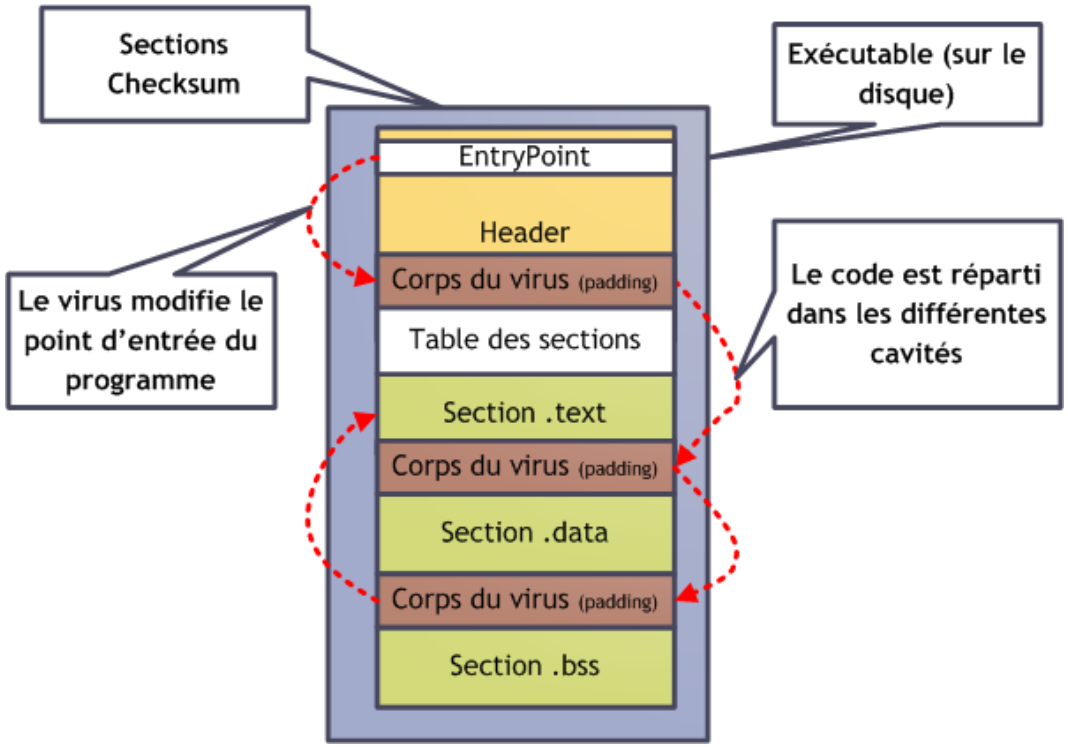
Exemple

Murky

Les infecteurs de cavité

Cavity infectors

- Code du virus assez court
- Fragmenté en N parties, liées entre elles par des `jmp`
- Inséré dans les sections *inutiles* (`.reloc`), dans le *padding*
- Modification du point d'entrée



Les infecteurs de cavité (2)

Avantages

- Pas de modification de la taille du fichier
  - Contre les whitelists
- Pas de création/modification des sections
  - Contre les heuristiques les plus grossières

Inconvénients

- Plus complexe (nécessite de reloger chaque partie)
- Plus de place que pour l'infection de header, mais reste court (- de 4k)

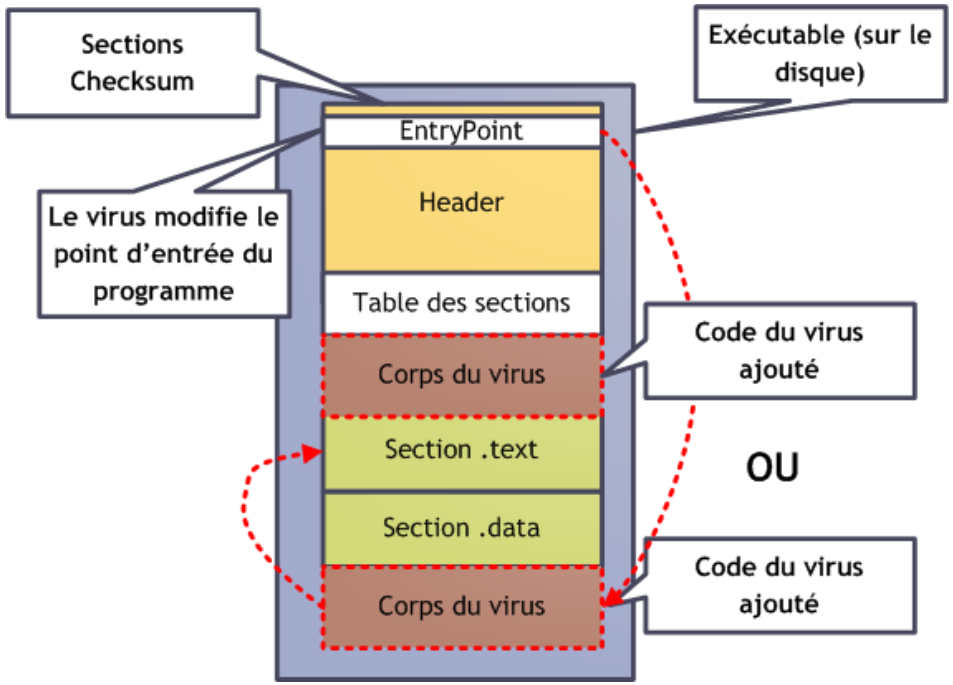
Exemple

CIH, W2K/Installer

Les infecteurs de fin/début de fichier

File infectors

- Code du virus arbitrairement long
- Inséré en début ou fin de fichier
- Modification de la table des sections
- Modification du point d'entrée



Les infecteurs de début de fichier

Infection en début de fichier

- Code du virus inséré **avant** la toute première section
- Décalage de toutes les sections suivantes
  - Nécessite les informations de relocation
  - Sauf si on utilise l'*imagebase trick*
- Plutôt discret (les heuristiques observent plus souvent la fin du fichier)

Les infecteurs de fin de fichier

Infection en fin de fichier

- Type d'infection le plus répandu
- Code du virus inséré **après** la dernière section
- Éventuellement, ajout d'une section supplémentaire
- Moins discret que l'insertion à la fin du fichier
  - ... mais plus simple

Les infecteurs de fin/début de fichier (2)

Avantages

- Portable
  - Quasiment tous les exécutables peuvent être infectés par cette méthode
- Simple à mettre en oeuvre
- Corps du virus arbitrairement long

Inconvénients

- Modification
  - de la taille du fichier
  - de la table des sections
  - de l'entrypoint
- Les antivirus regardent principalement la fin et le début des exécutable
- Virus en un seul bloc
  - signature plus aisée

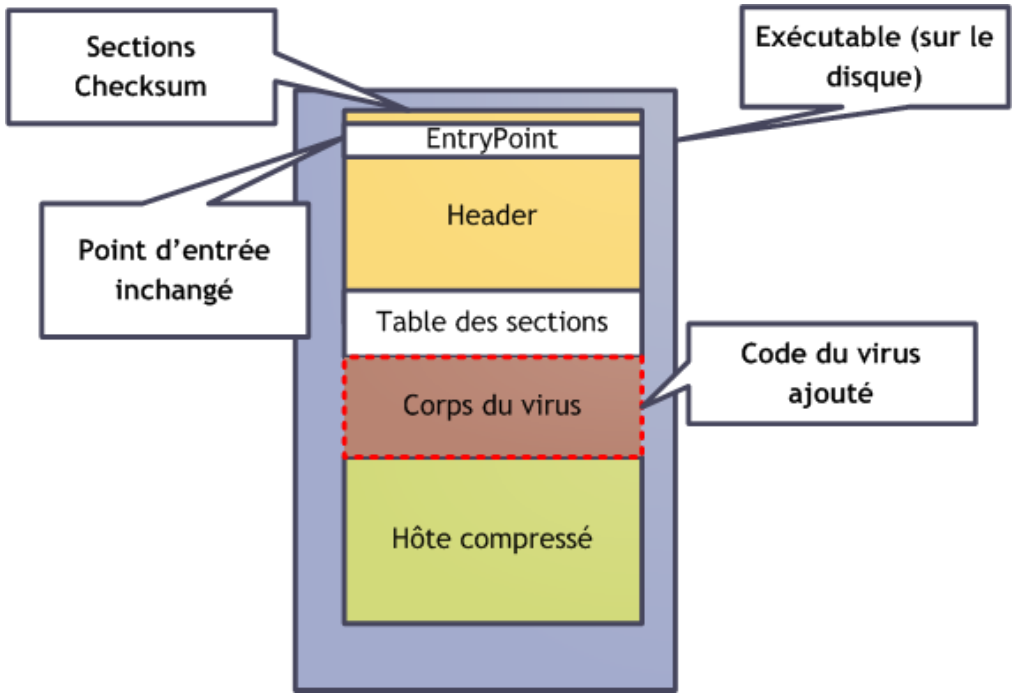
Exemple

Anxiety, 90% des virus

Les virus *compresseurs*

Compressing infectors

- Code du virus relativement court
- Inséré à la place du code de l'hôte
- L'hôte est compressé (ZIP,LFZ ...)
- ... et déplacé à la fin du fichier



Les virus *compresseurs* (2)

Avantages

- Pas de modification de la taille du fichier
  - Contre les whitelists
- Virus non-PIC
- Plus difficile à désinfecter

Inconvénients

- Plus complexe
- Code nécessairement plus petit
- Infection pas toujours possible

Exemple

W32/Redemption, W32/Aldebera

Démo

Démonstration

- Cible : Notepad
- Différences avant/après infection

Plan

La recherche par signature

- Scanner de première génération $\simeq$ FSA
  - Code $\Leftrightarrow$ données
  - Signature $\Leftrightarrow$ expression régulière
- Permet une détection exacte
- Complexité ?
- Un antivirus actuel contient plus de 100000 signatures

La recherche par signature (2)

```
seg000:7C40 BE 04 00                    mov     si, 4           ; Try it 4 times
seg000:7C40                             ;
seg000:7C43
seg000:7C43                 next:                               ; CODE XREF: sub_7C3A+27↓j
seg000:7C43 B8 01 02                    mov     ax, 201h        ; read one sector
seg000:7C46 0E                          push    cs
seg000:7C47 07                          pop     es
seg000:7C48                             assume es:seg000
seg000:7C48 BB 00 02                    mov     bx, 200h        ; to here
seg000:7C4B 33 C9                       xor     cx, cx
seg000:7C4D 8B D1                       mov     dx, cx
seg000:7C4F 41                          inc     cx
seg000:7C50 9C                          pushf
seg000:7C51 2E FF 1E 09 00              call    dword ptr cs:9  ; int 13
seg000:7C56 73 0E                       jnb     short fine
seg000:7C58 33 C0                       xor     ax, ax
seg000:7C5A 9C                          pushf
seg000:7C5B 2E FF 1E 09 00              call    dword ptr cs:9  ; int 13
seg000:7C60 4E                          dec     si
seg000:7C61 75 E0                       jnz     short next
seg000:7C63 EB 35                       jmp     short giveup
```

FIGURE: Stone : entrypoint

Signature : 0400 B801 020E 07BB 0002 33C9 8BD1 419C

La recherche par signature (3)

Format des signatures

- Wildcards (joker) pour les virus **oligomorphiques**

Exemple

```
0400 B801 020E 07BB ? ?02 %3 33C9 8BD1 419C
```

- Mismatches
  - mismatch de 2 $\Leftrightarrow$ autorise une distance ¡=2 entre chaque lexème du pattern
- Bookmarks
  - Zone de code stratégique du virus (OEP, clé de chiffrement)
  - Généralement utilisé pour la désinfection

Optimisation de la recherche par signature

- Filtrage de fichiers
  - Ne considérer que les `*.exe`
- Signature hashing
  - Les X premiers octets de la signature sont hashés
- Recherche localisée (smart scanning)
  - En début, fin de fichier, à l'entrypoint
- Smart scaning
  - Ignorer les opcodes inutiles (nop, std ...)

L'heuristique

Vérification statique de la structure des exécutables :

- L'entrypoint est dans la dernière section ?
- La dernière section est exécutable ?
- Certains champs du header sont mal calculés ?
- Une section .reloc ou .data reçoit le flot d'exécution ?
- Transfert du flot d'exécution d'une section à une autre ?
- Apis suspectes importées (FindFistFile/FindNextFile/WriteFile ...) ?

Heuristique

Exemple de sortie d'un moteur heuristique

```
Analyzing SGWW2202.VXE
-Execution starts in last section
-Suspicious code section characteristics
-Virtual size is incorrect in header
-Suspicious relocation
-Suspicious code section
-Using KERNEL32 address directly and looking for PE00
=> Possibly infected with an unknown Win32 virus (Level:
    9)
```

Plan

Principe

La mutation de code

- Technique de protection des virus
- Objectif : éviter la détection par les **scanners**
- Méthode : camoufler le code du virus
- Moyens :
  - Chiffrement
  - Oligomorphisme
  - Polymorphisme
  - Metamorphisme

Le chiffrement

- Technique simple : chiffrer le corps du virus
- Le code du virus commence par un petit décodeur **en clair**
  - Code constant
  - Clé embarquée, codée en dur
  - la clé **change à chaque génération**
- Une fois déchiffré, saut vers le corps du virus

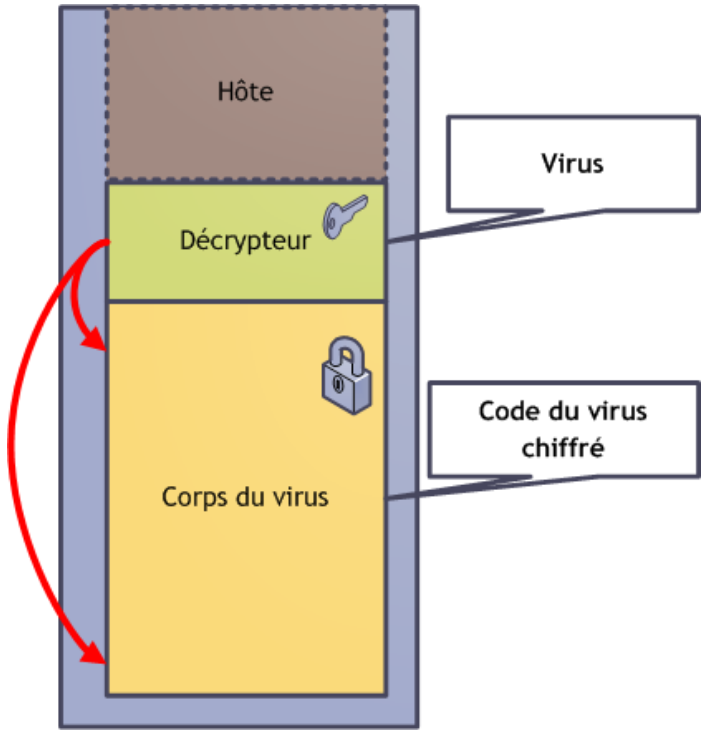


FIGURE: Chiffrement de virus

Le chiffrement (2)

Décrypteur du virus Mad

```
mov edi, Start
add edi, ebp
mov ecx, 0A6Bh
mov a1, [Key]
Decrypt :
xor [edi], a1
inc edi
loop Decrypt
jmp Start
Start :
; code du virus
```

Le chiffrement (3)

Technique plus avancées :

- Le décodeur peut changer de position
- Les décodeurs peuvent s'enchaîner (layers)
- Utilisation d'API crypto
- Le décodeur bruteforce la clé (`RDA fighter`)

A votre avis

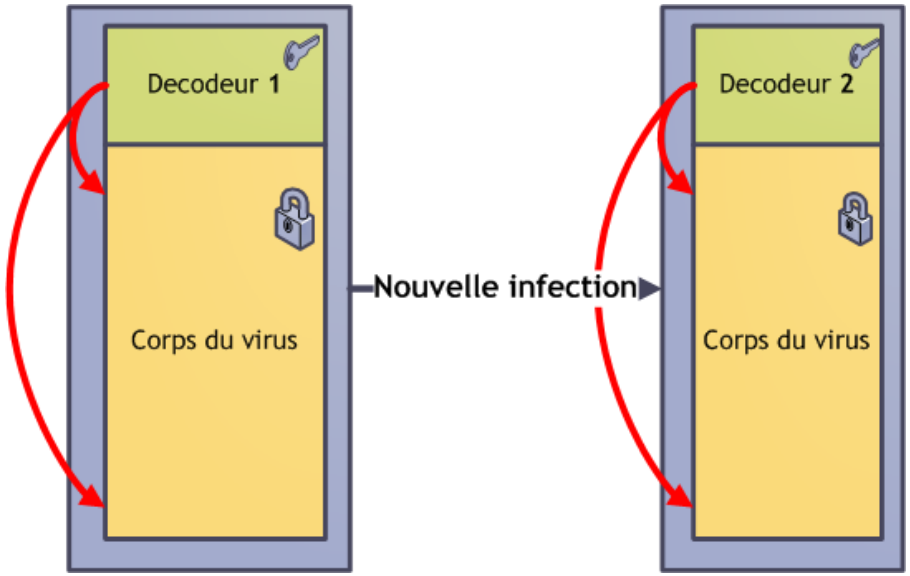
- Quelles contremesures possibles ?

Plan

L'oligomorphisme

Oligomorphisme

- Le décodeur reste en clair
- Utilisation de plusieurs décodeurs différents
  - Whale 12, Memorial 96
- Choix aléatoire d'un décodeur
- Chaque décodeur est écrit par le vxer



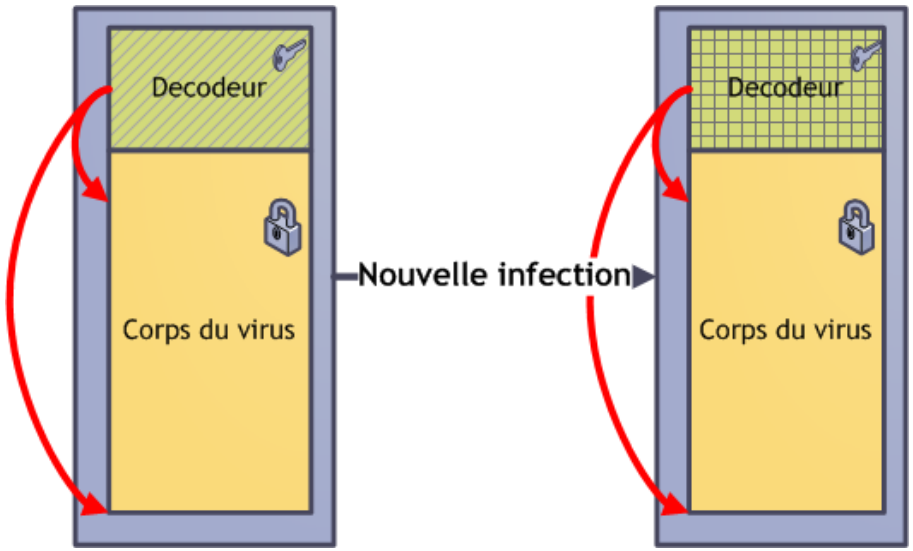
A votre avis

- Quelles contremesures possibles ?

Le polymorphisme

Le polymorphisme

- Muter le code du décodeur à chaque génération
- Code différent mais fonctionnement équivalent
- Premier virus polymorphique : 1260 (1990)



Mutation de code

Quelques mutations possibles :

- Insertion de code mort
- Réorganisation (géographique) du code
- Mutations syntaxiques (cf. TP)
  - Utilisation de registres différents
  - Utilisation d'opcodes différents
- Modification du CFG
- ... et beaucoup d'autres

```
A Taxonomy of Obfuscating Transformations -
C.Collberg
```

Un décrypteur de 1260

```
inc     si          ; optional, variable junk
mov     ax,0E9B     ; set key 1
clc                 ; optional, variable junk
mov     di,012A     ; offset of Start
nop                 ; optional, variable junk
mov     cx,0571     ; this many bytes - key 2
; Group 2 Decryption Instructions
Decrypt:
xor     [di],cx     ; decrypt first word with key 2
sub     bx,dx       ; optional, variable junk
xor     bx,cx       ; optional, variable junk
sub     bx,ax       ; optional, variable junk
sub     bx,cx       ; optional, variable junk
nop                 ; non-optional junk
xor     dx,cx       ; optional, variable junk
xor     [di],ax     ; decrypt first word with key 1
; Group 3 Decryption Instructions
inc     di          ; next byte
nop                 ; non-optional junk
clc                 ; optional, variable junk
inc     ax          ; slide key 1
; loop
loop    Decrypt     ; until all bytes are decrypted  slide key 2
; random padding up to 39 bytes
Start:
;       Encrypted/decrypted virus body
```

Démo

Démonstration

- Code source :

```
push 0xDEADBEEF
push 0xDEADBABE
```

- Mutation du code

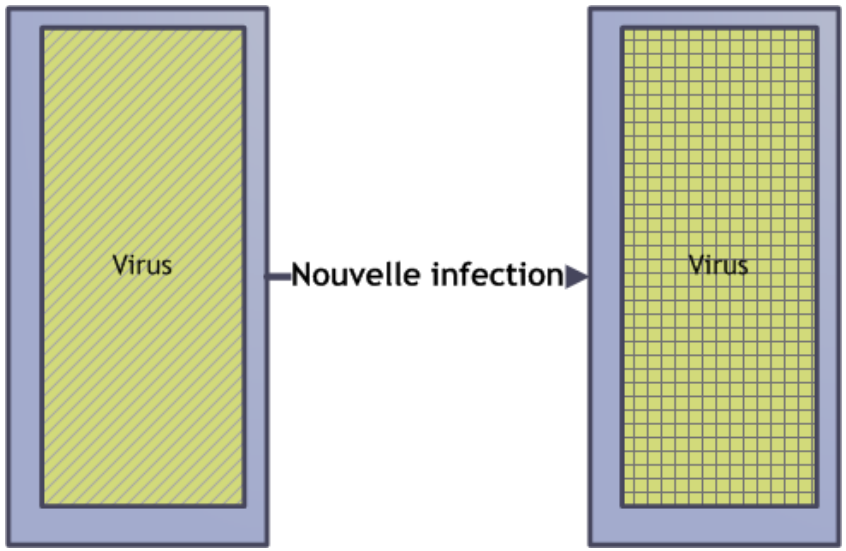
A votre avis

- Quelles contremesures possibles ?

Le metamorphisme

Le metamorphisme

- Plus de décodeur, trop "voyants"
- Polymorphisme de tout le virus
- Parfois metamorphisme de l'hôte
- Premier virus métamorphique : `Regswap` (1998)



Zmist

Zmist par Zombie (2000)

- Virus métamorphique
  - mutations syntaxiques
  - modification du CFG
- Intégration du virus dans l'hôte
  - Désassemblage de l'hôte
  - Relocation
  - Intégration du virus par îlots
- Recompilation du nouvel exécutable
- Détecté à 99,1% au bout de 7 ans

A votre avis

- Quelles contremesures possibles ?

Plan

Le problème de détection

- Virus chiffrés simples ou oligomorphiques $\Rightarrow$ détection par signature sur le décodeur
  - Au plus, N signatures nécessaires
- Technique *X-Ray*
  - Cryptanalyse automatisée pour les chiffrement les plus simples
  - Détection du **corps** du virus
- Heuristique
- Virus polymorphiques **simples** $\Rightarrow$ détection par signature**s**

Le problème de détection (2)

- Virus poly/métamorphiques complexes ?
  - Une infinité de décodeurs peut être générée
  - On parle du **langage** du moteur de poly/métamorphisme
  - Symboles terminaux $\Leftrightarrow$ instructions machines
- **Détection** du virus $\simeq$ **reconnaissance de ce langage**
- Quelle complexité ?
- Un peu de théorie ...

Hiérarchie des langages formels

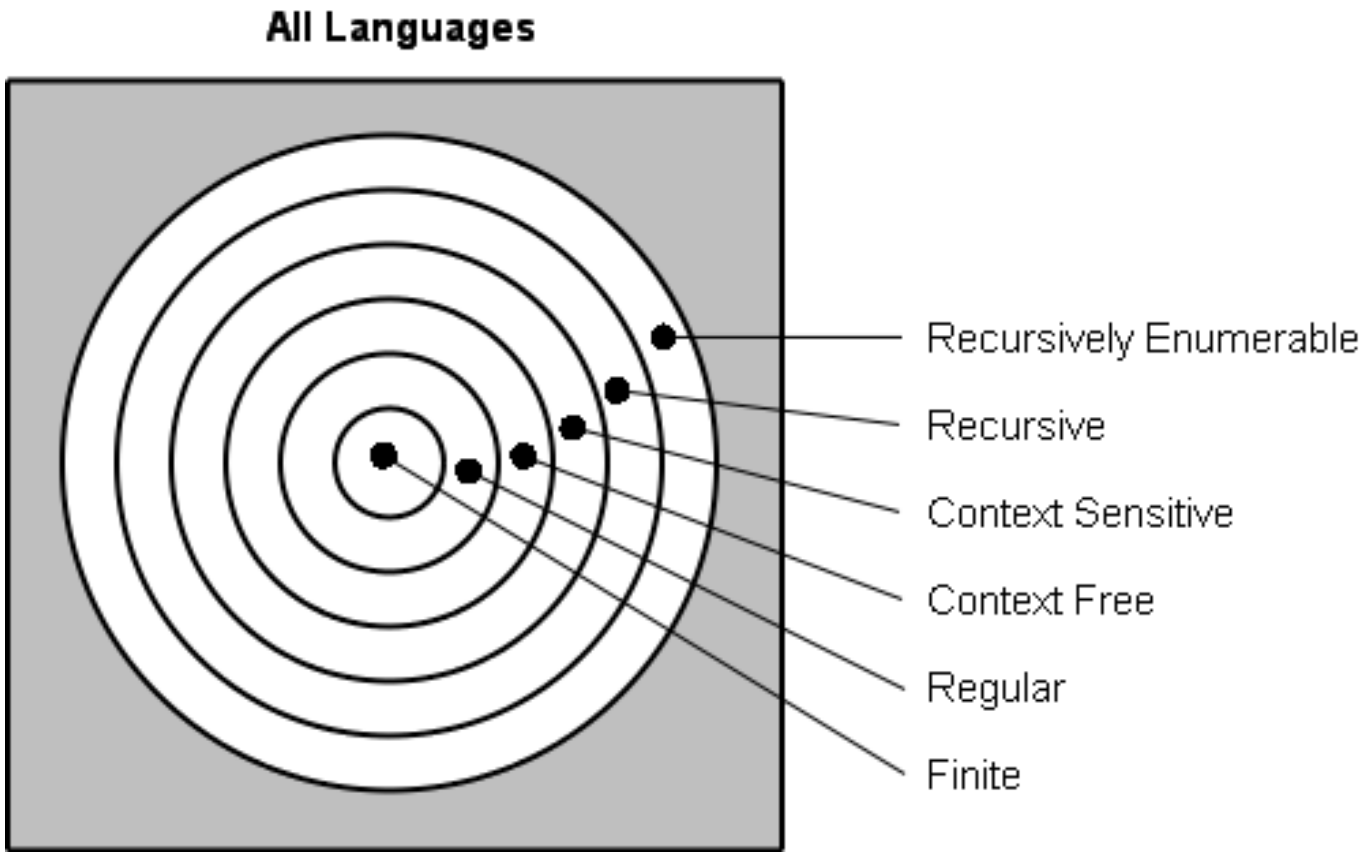


FIGURE: Langages formels

Hierarchie des langages et complexité

Chomsky level	Grammaire	Langage	Autom. min.	Complexité
Type 0	Unrestricted	Recursively enumlerable	Turing machine [1]	-
-	-	Recursive	Decider	?
Type 1	Contexte-sensitive	Contexte sensitive	Linear bounded TM	?
-	Tree adjoining	Middly context-sensitive	Embedded pushdown	P
Type 2	Context-free	Context-free	Nondet. pushdown	?
-	$LL(k)$ [2]	$LL(k)$	Det. lookahead pushdown	?
-	Det. context-free	Det. context-free	Det. pushdown	?
Type 3	Regular	Regular	DFA	?
-	-	Star-free	Aperiodic finite	?

- `mov reg,cst` $\Rightarrow$ `nop ; mov reg,cst` ?
- `mov reg,cst` $\Rightarrow$ `push reg2 ; mov reg2,cst ; mov reg,reg2 ; pop reg2` ?

1. qui peut ne pas s'arrêter
2. et $LR(k)$ ...

Le problème de détection (3)

- Le problème de détection général est **indécidable**, que faire ?
- Heuristique
- Une détection exacte n'est pas obligatoire
  - Détection de séquences d'instructions **suspectes** (signature)
- Analyse statistique des instructions

Scan antivir

```
C:\taf\cours\09-10\lp\secu\cours\virus\demo1\testnotepad.exe
    [RESULTAT]  Contient le code suspect : GEN/Malware
    [REMARQUE]  Le résultat positif a été classé comme suspect.
```

L'émulation

- Pour les virus polymorphiques, le corps du virus **déchiffré** reste constant
- Idée : **Emuler** le décodeur du virus
  - Interprétation du code machine instruction par instruction
  - Arrêt de l'émulateur à la fin du décodage
- Le corps du virus apparaît en clair dans la VM de l'antivirus
- $\Rightarrow$ signature sur le corps du virus

L'émulation(2)

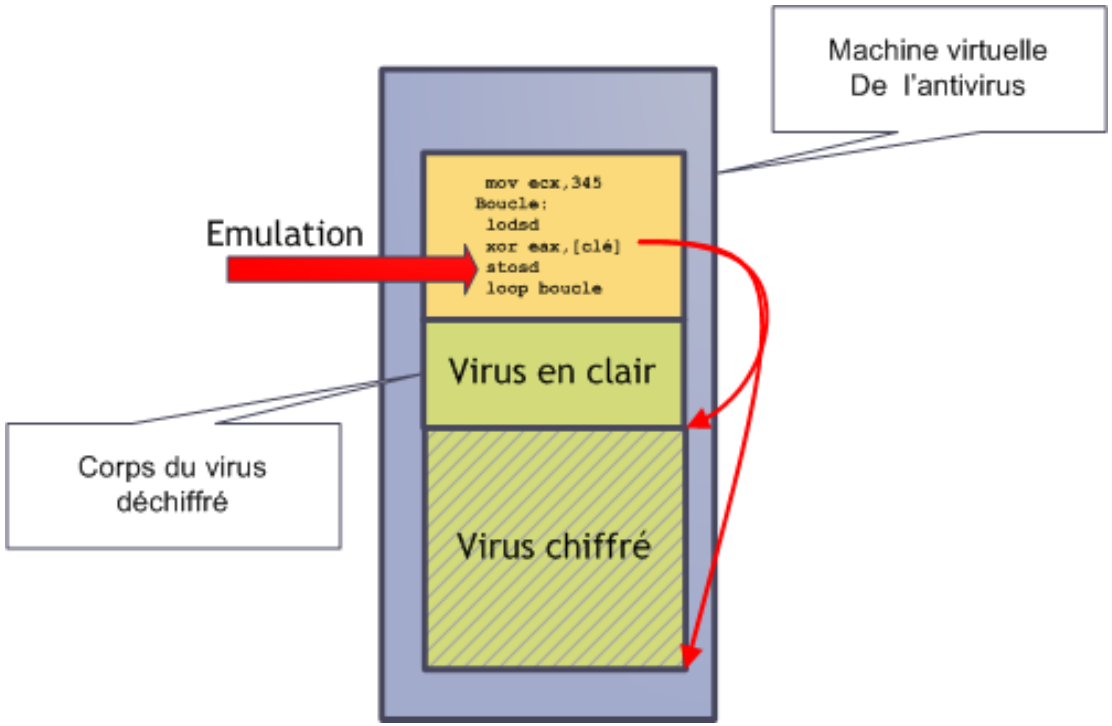


FIGURE: Emulation du décodeur du virus

L'émulation : problèmes et solutions

- Emuler du code est **lent** (et **non décidable**)
- Les virus tirent profit de cette lenteur :
  - Insertion de code inutile long à émuler
  - Insertion d'instructions non supportées par l'émulateur
- Contre-mesures :
  - Suppression du code mort
  - Exécution des boucles **inoffensives** en **natif** (*code normalization*)

La détection comportementale

- Virus métamorphiques complexes : que faire ?
- L'antivirus détourne les principaux appels systèmes
- A l'**exécution** d'un programme, la séquence d'appels système est vérifiée
  - Détection de sous-séquence d'appels systèmes suspects (`FindFirstFile(*.exe)`, `WriteFile ...`)
  - Détection de fréquences d'appels systèmes suspects
- Avantage : très efficace contre les vers/trojans (API réseau)
- Inconvénients :
  - Ralentit le système de l'utilisateur
  - Détection parfois trop tardive

Détection comportementale

Malware behaviour analysis *Wagener, G. and State, R. and Dulaunoy, A.* - 2008

Etat de l'art

Etat de l'art :

- Signature + heuristique + émulation
- Heuristique au sein de l'émulateur
  - Détection d'accès séquentiel au code du programme
  - Détection de routines virales classiques (scan *.exe, etc.)
  - Code auto-modifiant
- Sandboxing (expérimental) : émulation de l'os en entier
- la majorité des virus sont détectés sans mise à jour de l'antivirus

Plan

Actuellement

- L'augmentation de la fréquence des CPU joue en faveur des antivirus
  - Capacité d'émulation augmentée
- L'augmentation de la taille des disques et du débit réseau joue en faveur des virus
  - Plus complexes
- Néanmoins :
  - Les virus restent majoritairement détectés dès leur sortie
  - Très peu de virus innovants actuellement
  - Arrestations massives il y a 4-5 ans (29A)

Plan

Changement sociologiques

- Presque plus de vxers hobbyistes
- La mode est au **profit**
  - Equipe de plusieurs programmeurs
  - Réutilisation des techniques de virologie
  - Ajout de propagation via le réseau (mix virus/vers/botnets)
- Une faille = un vers lancé sous deux semaines
- Objectif : spam $\Rightarrow$ profit

La rentabilité des botnets

Spamalytics : An Empirical Analysis of Spam Marketing Conversion
`http://www.icsi.berkeley.edu/pubs/networking/2008-ccs-spamalytics.pdf`

Récapitulatif

Attaque	**Défense**
Virus boot	Signature
Infecteur de fichier	Signature, heuristique
Polymorphisme	Émulation+Signature, heuristique
Metamorphisme	Détection comportementale, heuristique
Vers, chevaux de troie	Détection comportementale, signature

- La **quasi totalité des techniques** des rootkits, vers, trojans et shellcodes **sont issues des virus**