Licence Pro Alternance: 2007/2008

Les virus informatiques

attaques* et *défenses

Renaud Tabary

Plan

- Introduction
- Première génération
  - Les virus de boot
  - Premiers antivirus
- Deuxième génération
  - Les infecteurs de fichiers
  - Signatures et heuristique
- Troisième génération
  - Encryption, Polymorphisme et Métamorphisme
  - Emulation et autres techniques
- Quatrième génération : internet
  - Les vers
  - Détection dynamique
- Conclusion

Introduction : Définition

- Définition de F.Cohen : "*A virus is a program that is able to infect other programs by modifying them to include a possibly evolved copy of itself.*"
- Virus != malware
- Programmes souvent écrits en assembleur
- Entre 10 lignes et 20000 lignes de code utiles

Introduction : Le marché des antivirus

- Marché éclaté entre plus de 20 marques (3 leaders : McCafee, Symantec et Trend Micro)
- Marché en progression (+ 13,4% en 2006)
- Collaboration entre les différents éditeurs
- Explosion du marché avec les nouveaux téléphones mobiles

Introduction : Pourquoi les étudier ?

- La plupart des techniques des rootkits, vers, trojan et shellcodes sont issues des virus

- Même si ils sont moins répandus aujourd'hui, ils restent une menace importante

- Un peu plus que de simples ''format c:\'' :
  - Théorie des langages
  - Programmation système
  - Reverse engineering et obfuscation de binaires

- Le problème de détection est insoluble

  - Langages context-free

Introduction: menace pour l'entreprise

- Remet en jeu **l'intégrité** et la **disponibilité** du Système Informatique:
  - Suppression de fichiers
  - Backdoors
  - Charge réseau (vers)
- Parfois la **confidentitalité**
  - Vol de mots de passe
- Explosion du marché avec les nouveaux téléphones mobiles

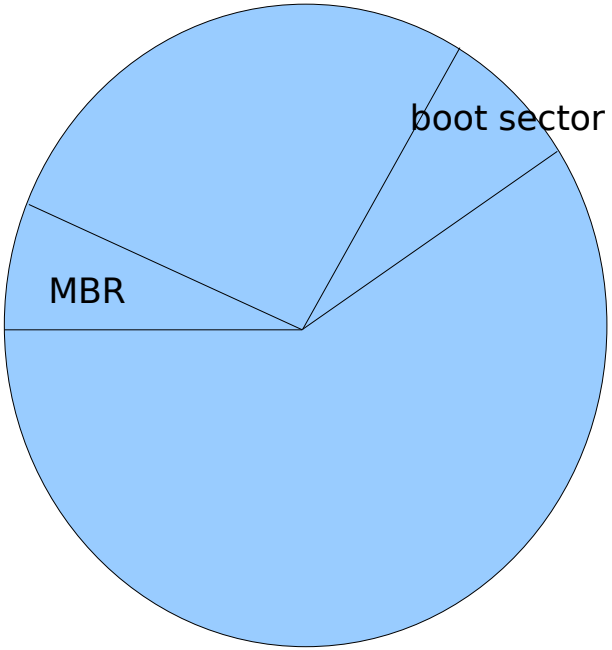
Première génération : les virus de boot

- 1er des virus informatique
- créé en 1986, le virus *Brain*
- avantage : OS indépendant
- démarrage du virus avant l'OS

Première génération : les virus de boot

Principe :

- impossible de charger l'OS depuis la ROM
- lecture du *master boot record* (MBR) et chargement en mémoire
- la MBR contient un *boot strap loader* qui charge la partition active depuis la table des partitions
- la partiition active contient le code de chargement de l'OS



Première génération : les virus de boot

Les types d'infection du MBR :

- Remplacement *boot strap code (Stoned)*
  - déplacement de la table des partitions (fin de la MBR ou sur le disque)
  - insertion du code à la place du *boot strap code*
- Remplacement *de la MBR (Asuza 91)*
  - pas de sauvegarde de la table des partitions
  - chargement de la partition active par le virus
- Modification de la table des partitions *(StarShip)*
  - modification de la table des partitons
  - chargement d'un secteur différent, où est situé le corps du virus

Première génération : les scanners « home-made »

- Pas encore de réelles sociétés d'antivirus
- Souvent un antivirus pour chaque virus
- Apparition de la recherche par signature :

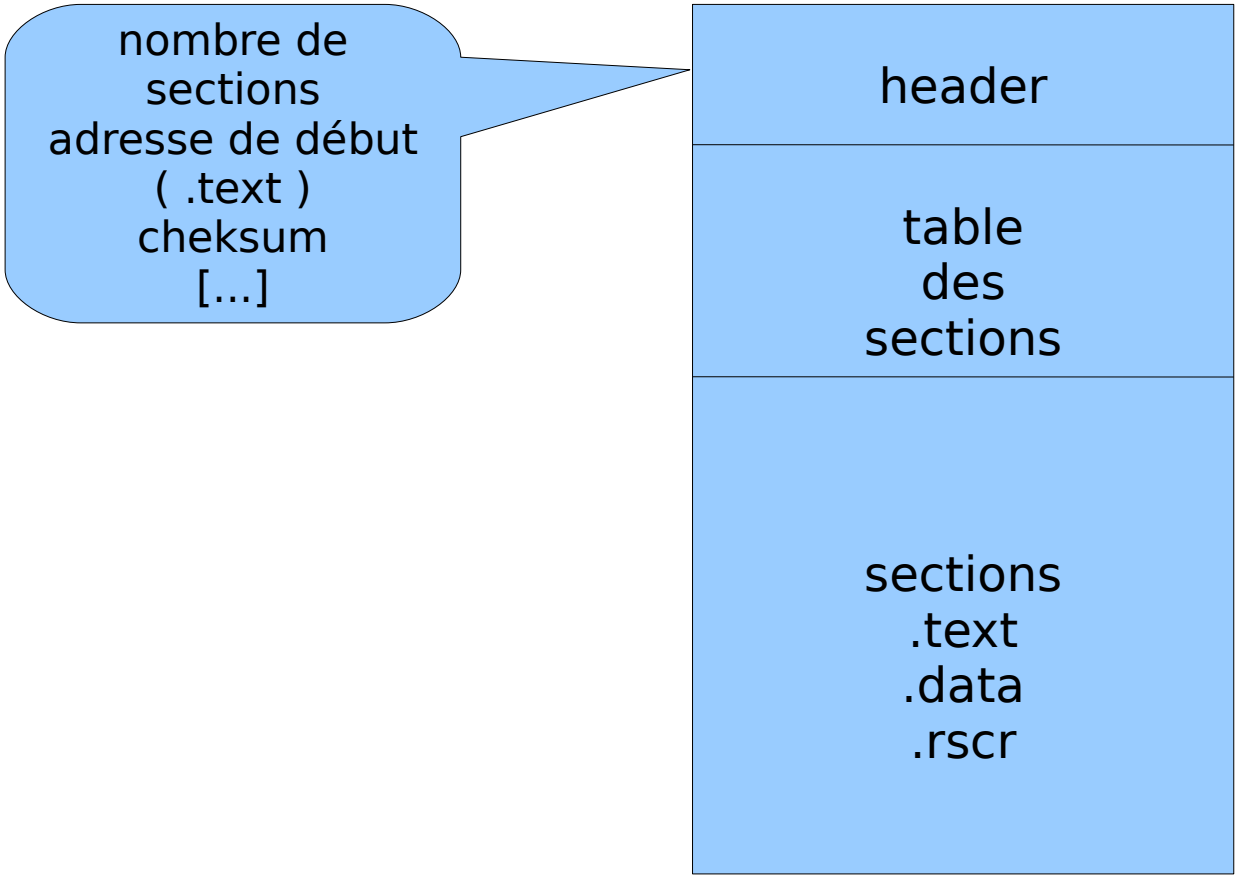
Signature

```
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
```

Le virus Stoned

Deuxième génération : les infecteurs de fichiers

Le format de fichier exectutable classique : Le Portable Executable



Deuxième génération : les infecteurs de fichiers

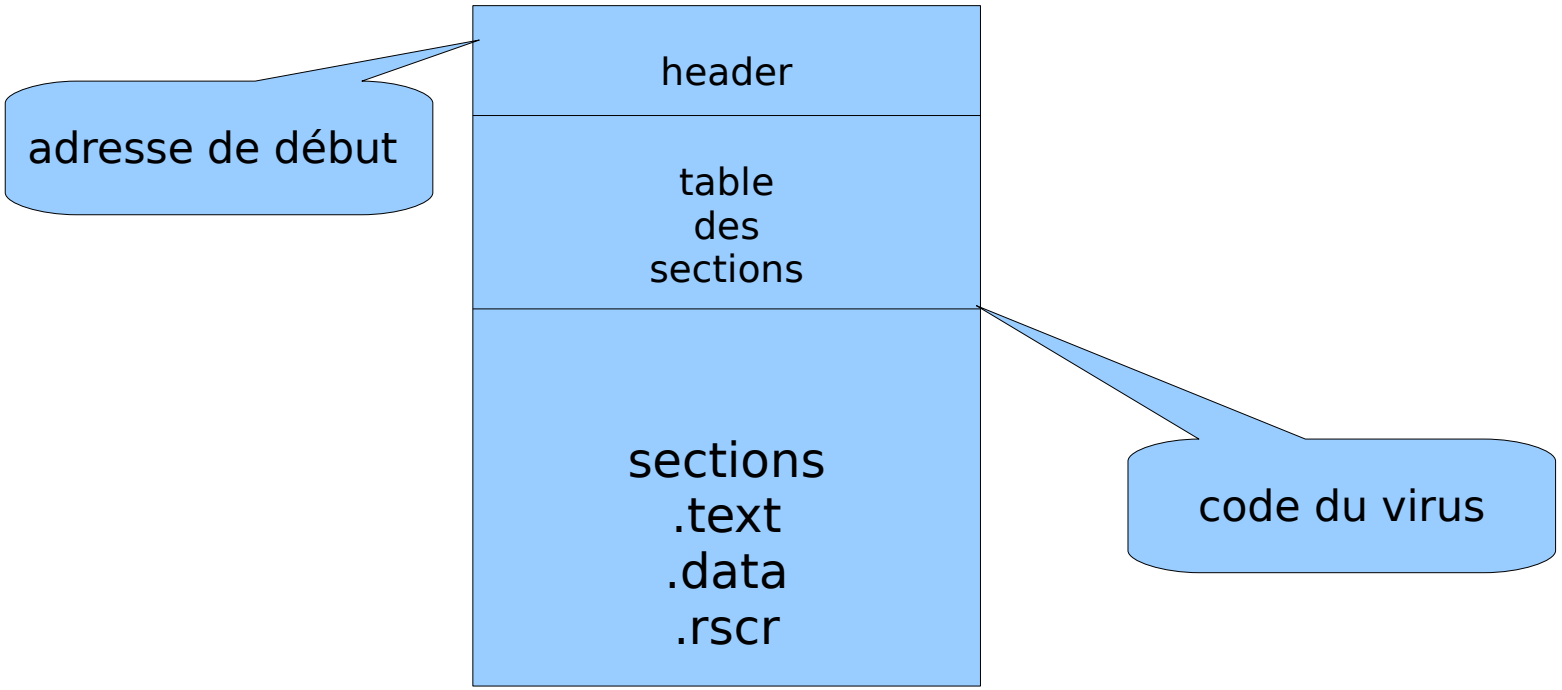
Les compagnons (Spawn)

- création d'un executable en .COM
- Le .COM est executé avant le .EXE
- appel à la routine CreateProcessA() pour lancer le vrai executable

Deuxième génération : les infecteurs de fichiers

Les infecteurs de header (Murkry)

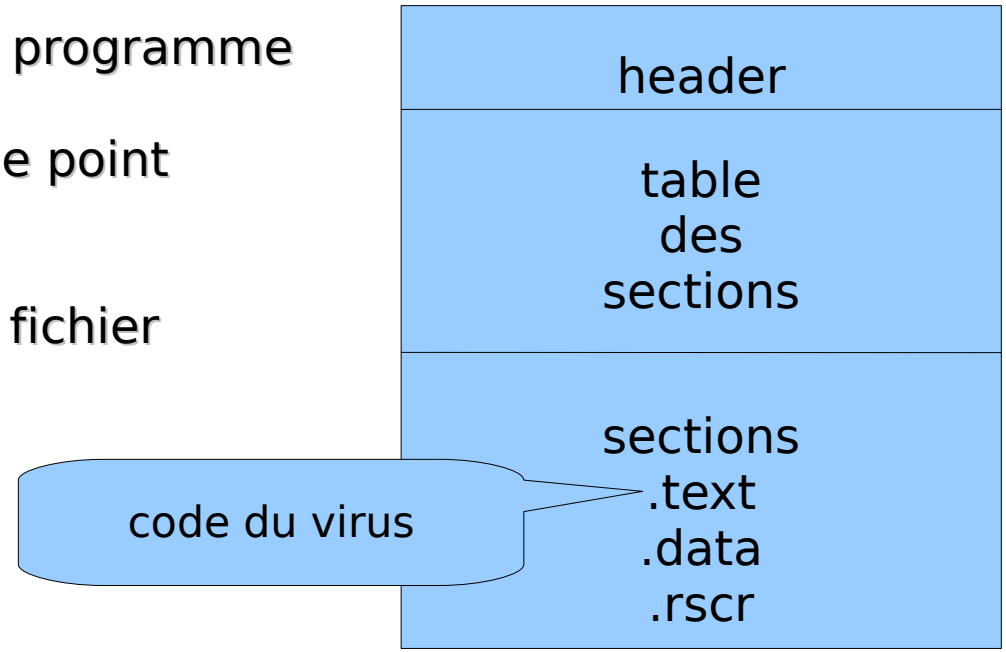
- code du virus très court inséré après la table des sections
- modification de l'adresse de point d'entrée dans l'en-tête



Deuxième génération : les infecteurs de fichiers

Les infecteurs de début de fichier
(Prepending Virus)

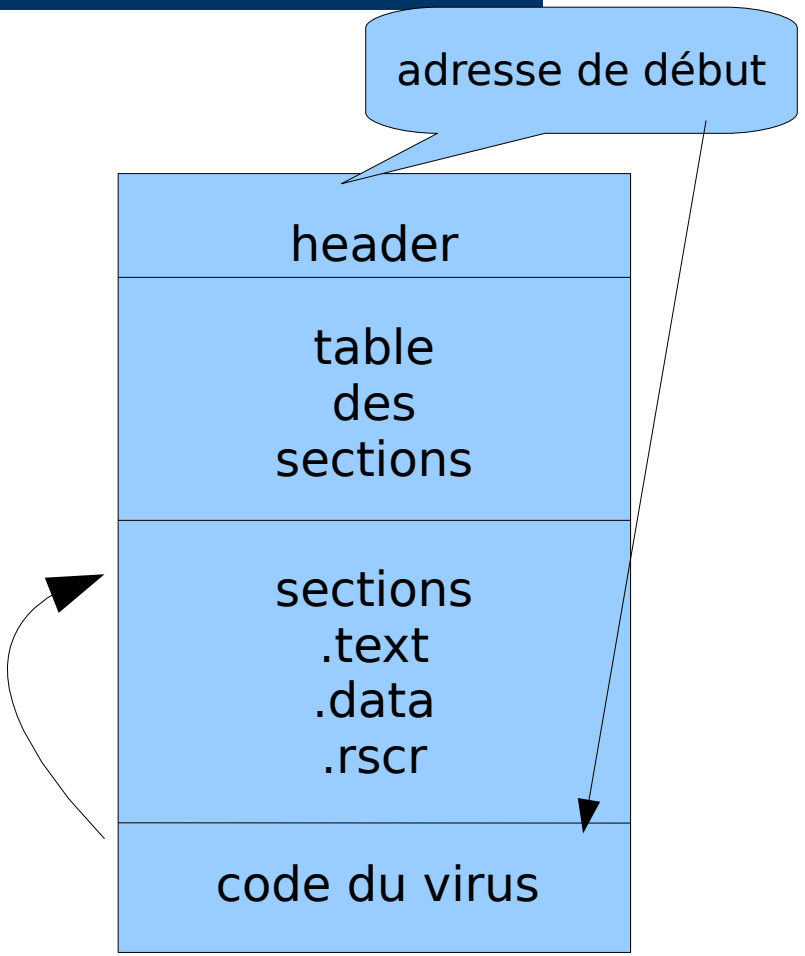
- très simple à mettre en place
- écrasement du code démarrant le programme
- pas de modification de l'adresse de point d'entrée dans l'en-tête
- pas de modification de la taille du fichier



Deuxième génération : les infecteurs de fichiers

Les infecteurs de fin de fichier (Appending Virus) - (Boza)

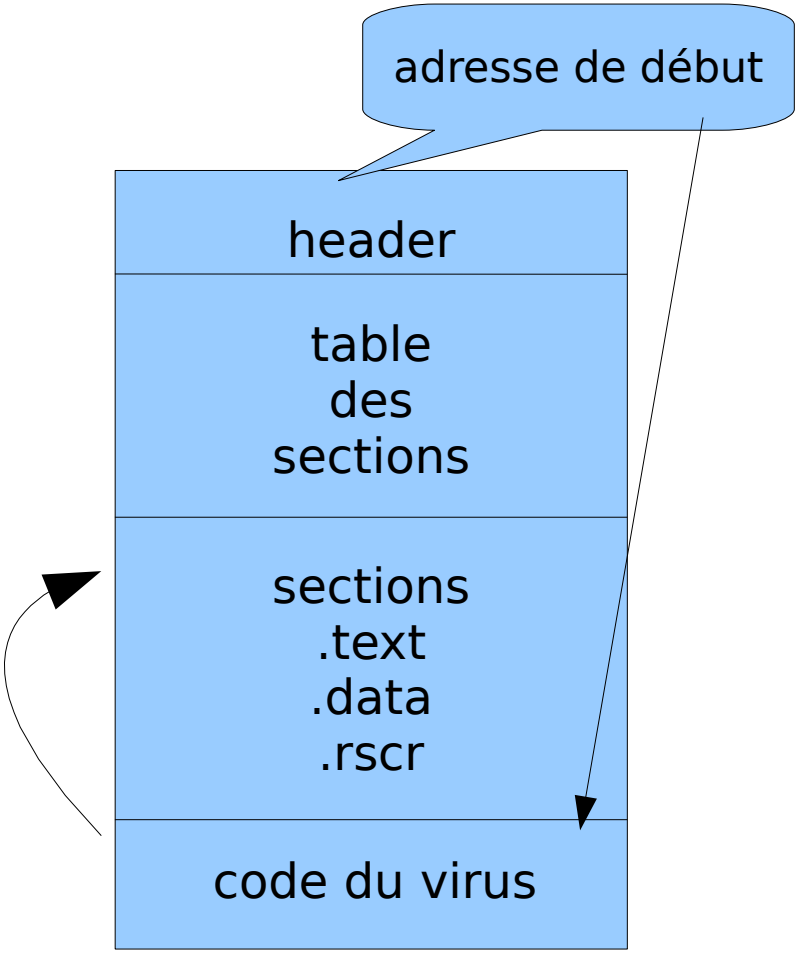
- ajout d'une nouvelle section contenant le code du virus (en-tête, table des sections, sections)
- modification de l'adresse de point d'entrée dans l'en-tête pour pointer sur la nouvelle section
- rend la main au début du code une fois exécuté



Deuxième génération : les infecteurs de fichiers

Les infecteurs de fin de fichier
(Appending Virus) - (Anxiety)

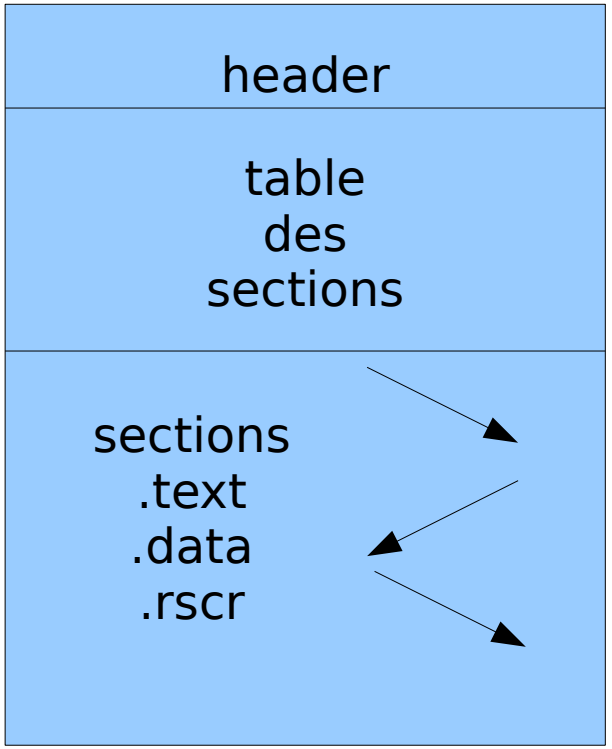
- utilisation de la dernière section
- modification de l'adresse de point d'entrée dans l'en-tête pour pointer sur la nouvelle section
- modification du dernier champ de la table des sections et insertion du code du virus en fin de cette section



Deuxième génération : les infecteurs de fichiers

Les infecteurs de cavités (Cavity Virus) - (CIH)

- utilisation des espaces entre les différentes sections (0xCC)
- deplacement des fragments de code vers une zone mémoire
- le code de chargement peut-être court et former un corps de virus très long



Deuxième génération : La recherche par signature

- Généralisation de la détection par signature
  => Un antivirus actuel contient plus de 100000 signatures

- Exécutables plus gros, virus plus complexes
  => Nécessité d'une recherche « intelligente »

  - Recherche de signature à partir de la fin du fichier exécutable
  - Recherche de signature à partir de l'adresse de début
  - Hashing des premiers octets de la signature
  - « Jokers » pour détecter plusieurs virus (ou variantes) avec une seule signature :

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

Deuxième génération : L'heuristique

- Vérification statique de la structure des exécutables :
  - L'adresse du début est dans la dernière section ?
  - La dernière section est exécutable ?
  - Certains champs du header sont mal calculés ?
  - Une section .reloc ou .data recoit le flot d'exécution ?
  - Transfert du flot d'exécution d'une section à une autre
  - Apis suspectes importées (FindFistFile/FindNextFile/WriteFile ...)
  - Etc.

Deuxième génération : L'heuristique

- Méthode générique pour détecter des fichiers suspects
- Ne permet pas une détection exacte du virus
- Introduits rapidements des faux positifs
- Peut néanmoins d'indiquer rapidement à l'antivirus si il doit passer du temps supplémentaire à l'analyse du fichier
- Heuristique actuelle très performante : détecte 90% des virus inconnus « simples »

Troisième génération : polymorphisme et métamorphisme

Principe

- technique d'auto-protection des virus
- objectif : éviter la détection par les scanners
- moyen : cacher le code du virus
- moyens :
  - cryptage
  - polymorphisme
  - métamorphisme

Troisième génération : polymorphisme et métamorphisme

le cryptage

- technique simple : crypter le code du virus
- le code du virus commence par un décodeur, code constant et en clair

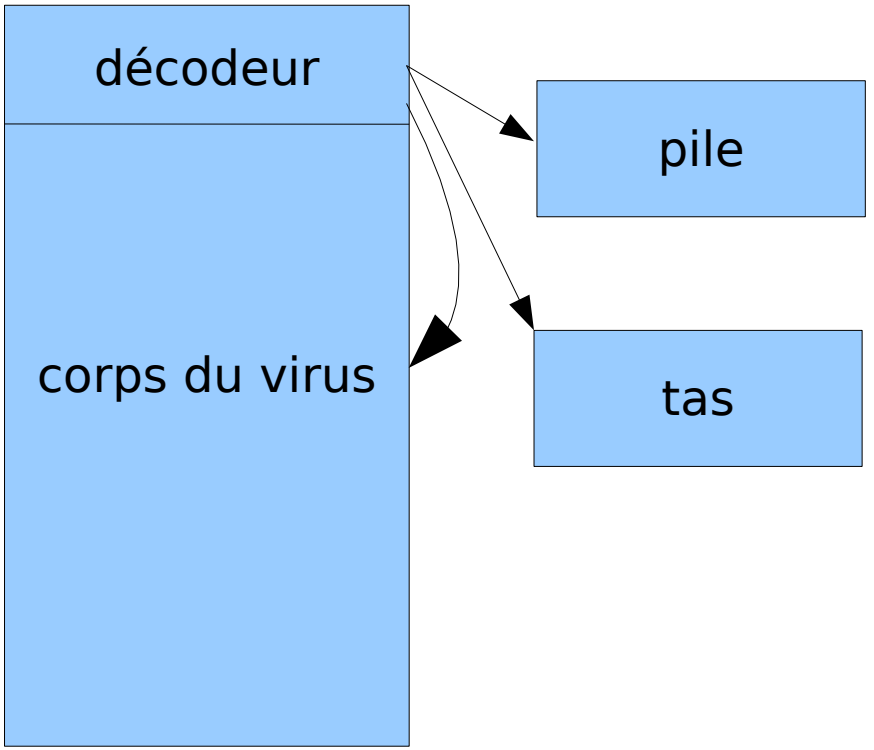
Exemple du virus Mad :

```
        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
```

Troisième génération : polymorphisme et métamorphisme

le cryptage

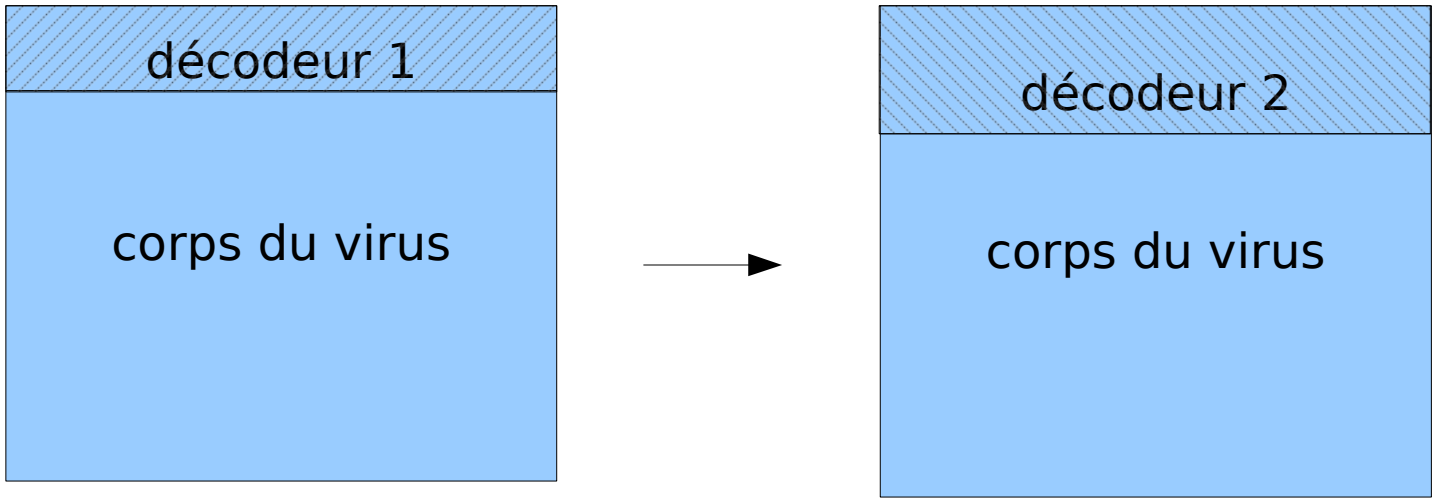
- le décodeur peut changer de position
- les décodeurs peuvent s'enchainer
- la clé n'est pas forcément présente (brute force) (RDA fighter)
- utilisation d'un API particulière de cryptage/décryptage (Crypto)



Troisième génération : polymorphisme et métamorphisme

l'oligomorphisme

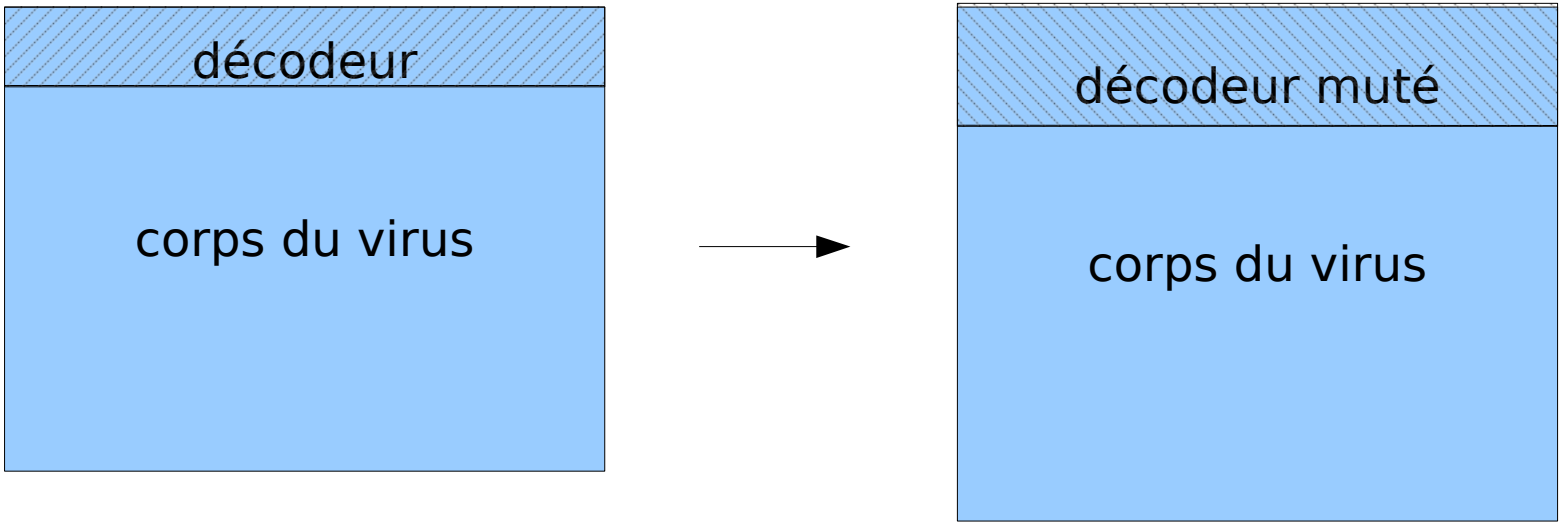
- le décodeur reste en clair
- utilisation de plusieurs décodeurs (Whale utilise 12 décodeurs, Memorial 96)
- choix aléatoire d'un décodeur



Troisième génération : polymorphisme et métamorphisme

le polymorphisme

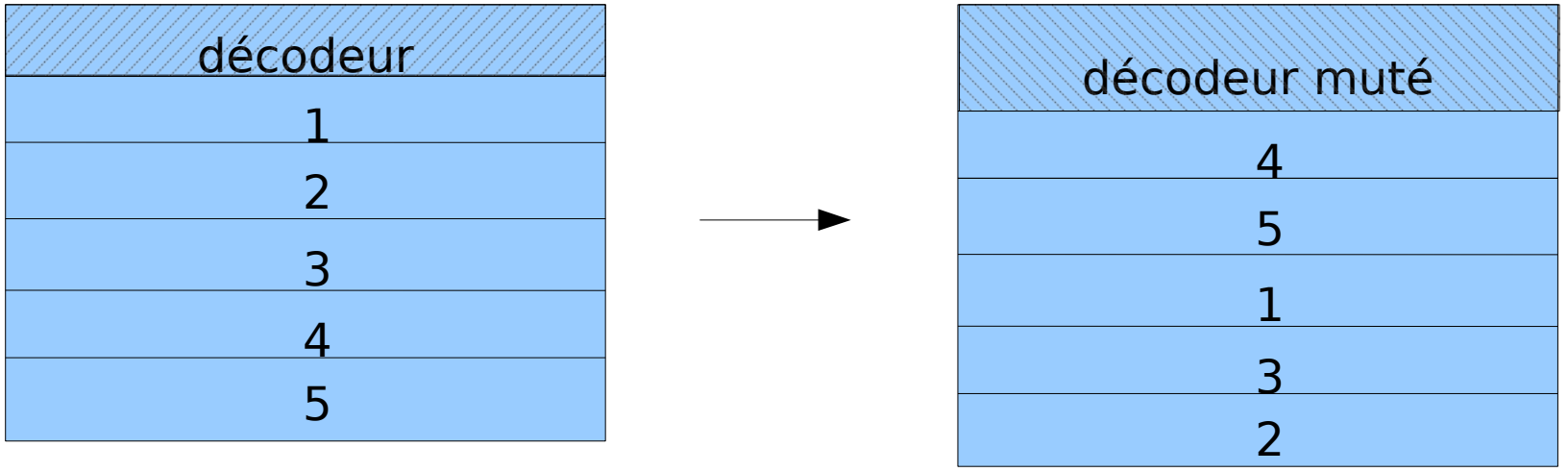
- l'objectif est de « muter » le décodeur d'une génération à l'autre
- le 1er virus polymorphique : 1260 (1990)
- insertion de 'code mort' pour changer l'apparence du décodeur



Troisième génération : polymorphisme et métamorphisme

le métamorphisme

- problème : le corps du virus decrytper reste constant et identique
- le métamorphisme : polymorphisme du corps du virus
- 1er virus : Regswap (1998). Utilisation de registres différents
- Ghost Virus : réordonnancement des sous-routines du code



Troisième génération : polymorphisme et métamorphisme

l'ultime virus : Zmist par Zombie (2000)

- intégration d'un decompilateur
- modification du code de l'exécutable (intégration du virus)
- compilation de ce nouvel exécutable
- intégration de permutation (condition inversée, XOR/SUB, MOV/PUSH, ...)
- détecté à 99,1% au bout de 7 ans

Troisième génération : détection de virus cryptés

- Pour les virus cryptés, des signatures sont faites sur le décrypteur qui reste en clair

- Certains anti-virus essayent de décrypter le corps du virus lorsque l'algorithme est simple (XOR, ADD, ROR ...) : méthode X-Ray

- Utilisation massive de « jokers » dans les signatures pour les décrypteurs polymorphiques les plus simples

- La recherche par signature est impuissante face aux virus polymorphiques complexes

Troisième génération : l'émulation

- Émuler les premières instructions du virus jusqu'à passer le décrypteur : le corps du virus apparaît alors en clair
- Méthode lente et incertaine : comment émuler un accès à un site web ?
- Les virus tirent profits de cette lenteur:
  - insertion de code inutile long à émuler
  - insertion d'instructions non supportées par l'émulateur
- L'émulation peut cependant être optimisée:
  - Suppression des instruction « do-nothing »
  - Exécution des boucles inoffensives en natif

Troisième génération : l'état de l'art

- Combinaison : signatures + heuristique + émulation
- La détection heuristique est lancée depuis l'émulateur :
  - Détection d'accès mémoire séquentiel
  - Statistiques sur les fréquences des instructions
  - Détection de routines classiques (recherches des apis en mémoire, listing des fichiers *.exe, etc.)
- Sandboxing (expérimental) : émulation d'un os en entier
- La majorité des virus et vers sont détectés sans mise à jour de l'antivirus

Quatrième génération : internet

Méthode d'infection bien plus rapide : internet

- Apparition des vers
- Apparition de virus/vers

les différentes infections

- vers
- exploit
- vulnérabilités
- overflow de mémoire

Quatrième génération : internet

les vers :

- infection destiné aux réseaux
- utilisent :
  - un localisateur de cible (email, scan du réseau)
  - un propagateur d'infection (confiance utilisateur, exploit, buffer overflow)
  - une interface de contrôle
  - un manager de vie (suicide, activation)

Quatrième génération : internet

exploitation des failles :

- le buffer overflow :

```
char  buffer[256];

for (i=0;i<512;i++)
   buffer[i] = 'A';
```

Quatrième génération : internet

exploitation des failles :

- le buffer overflow :

```
char  buffer[256];

for (i=0;i<512;i++)
   buffer[i] = 'A';
```

buffer[256]
@ EBP = 0x0012FFF0
@ EIP = 0x00401000

AAAAAAAAAAAAAAAA
@ EBP = 0x0012FFF0
@ EIP = 0x00401000

AAAAAAAAAAAAAAAA
@ EBP = 0x41414141
@ EIP = 0x41414141

Quatrième génération : internet

exploitation des failles :

- le heap overflow :

```
char *buffer = (char *) maloc(16)
char *input = (char *) maloc(16)

strcpy(buffer, ''AAAAAAAAAAAAAAAA'');
strcpy(input,argv[1]);
```

Quatrième génération : internet

exploitation des failles :

- le heap overflow :

```
char *buffer = (char *) maloc(16)
char *input = (char *) maloc(16)

strcpy(buffer, ''AAAAAAAAAAAAAAA'');
strcpy(input,argv[1]);
```

Cas 1 :

Adresse	Variable	Valeur
00300350	Input	BBBBBBBBBBBBBBB
00300360	????	
00300370	Buffer	AAAAAAAAAAAAAAA

Cas 2 :

00300350	Input	BBBBBBBBBBBBBBB
00300360	????	BBBBBBBBBBBBBBB
00300370	Buffer	BBBBAAAAAAAAAAA

Quatrième génération : internet

exploitation des failles :

- format string

```
print(argv[1]);
```

%X -> 401064

123456%n -> écriture de 6 à @401064

AAAA%x%x%n -> écriture @41414141

Quatrième génération : internet

exploitation des failles :

- le vers Morris (1988)
- utilisation du protocole 'finger' sur le port 79
- fonction 'gets' sur BSD vulnérable
- finger utilise un buffer de 512 octets
- le vers envoie une chaîne en assembleur de 536 octets
- modification de l'adresse du retour du 'main' pour exécuter le code qui lance un shell (exec /bin/sh)
- pour les 10 ans de ce vers, des hackers ont créé le vers 'ADM' utilisant une vulnérabilité de BIND

Quatrième génération : Internet

- Combinaison : signatures + heuristique + émulation
- Incorporation d'un firewall dans les antivirus
- Détections dynamiques des vers par api hooking :
  - L'antivirus prend la main avant l'OS
  - Observation des méthodes sensibles comme CreateRemoteThread et WriteprocessMemory
- Détection dynamique (expérimental) : le virus est détecté à travers son comportement
  - Séquence d'apis utilisées
  - Fréquence des apis utilisées

Conclusion

- L'augmentation des fréquences CPU joue en faveur des Antivirus dont la capacité d'émulation augmente
- L'augmentation de la taille des disques durs joue en faveur des virus qui peuvent être plus complexes. Néanmoins :
  - Les virus restent majoritairement détectés dès leur sortie
  - Très peu de virus innovant sont crées actuellement
  - Arrestations massives il y a 3-4 ans : 29A

Conclusion

ATTAQUE	DEFENSE
Virus Boot	Signature
Infecteur de fichier	Signature Heuristique
Polymorphisme	Émulation Signature
Métamorphisme	Émulation Détection comportementale
Vers	Antivirus + Firewall

Conclusion

- L'essor d'Internet profite aux vers :
  - Détectés plus facilement qu'un virus, mais se propagent encore plus vite
  - Quelques lignes de C suffisent pour créer la plupart
  - Tirent profit des différents packeurs/encrypteurs du commerce
- Écrire des malwares pour de l'argent :
  - Réseaux de zombis
  - Relais de spam
  - Keyloggeurs, encrypteurs
- Nouveau créneau : les portables nouvelles génération