

## Les virus aujourd'hui

*Les virus font peur et les fabricant de système anti-virus luttent activement contre cette menace. Faisons le point où nous en sommes au jour d'aujourd'hui.*

### **I. Introduction**

Nous nous souvenons tous des vagues virales ayant paralysé une partie d'internet, de nos collègues de bureau nous envoyant des mails qu'ils n'ont jamais tapés ou encore avoir un compte à rebours avant l'extinction de votre ordinateur. De nos jours ces virus ont totalement disparus et la majorité des utilisateurs possèdent un anti-virus. Nous pouvons alors nous sentir en sécurité derrières nos équipements mais se maintient le point d'interrogation sur « Cela peut il de nouveau arriver ? ». Tout au long de cet article nous tenterons d'apporter des réponses à ces questions et de lever le voile sur notre sécurité personnelle.

Dans un premier temps nous aborderons les caractéristiques des premiers virus / vers, puis nous constaterons le changement de profil des attaquant et finirons avec un tour d'horizon sur les nouvelles menaces.

### **II. Les anciens virus**

Dans les débuts les virus étaient programmés par des amateurs, ceux-ci les développaient pour le plaisir du savoir et de tester les systèmes.

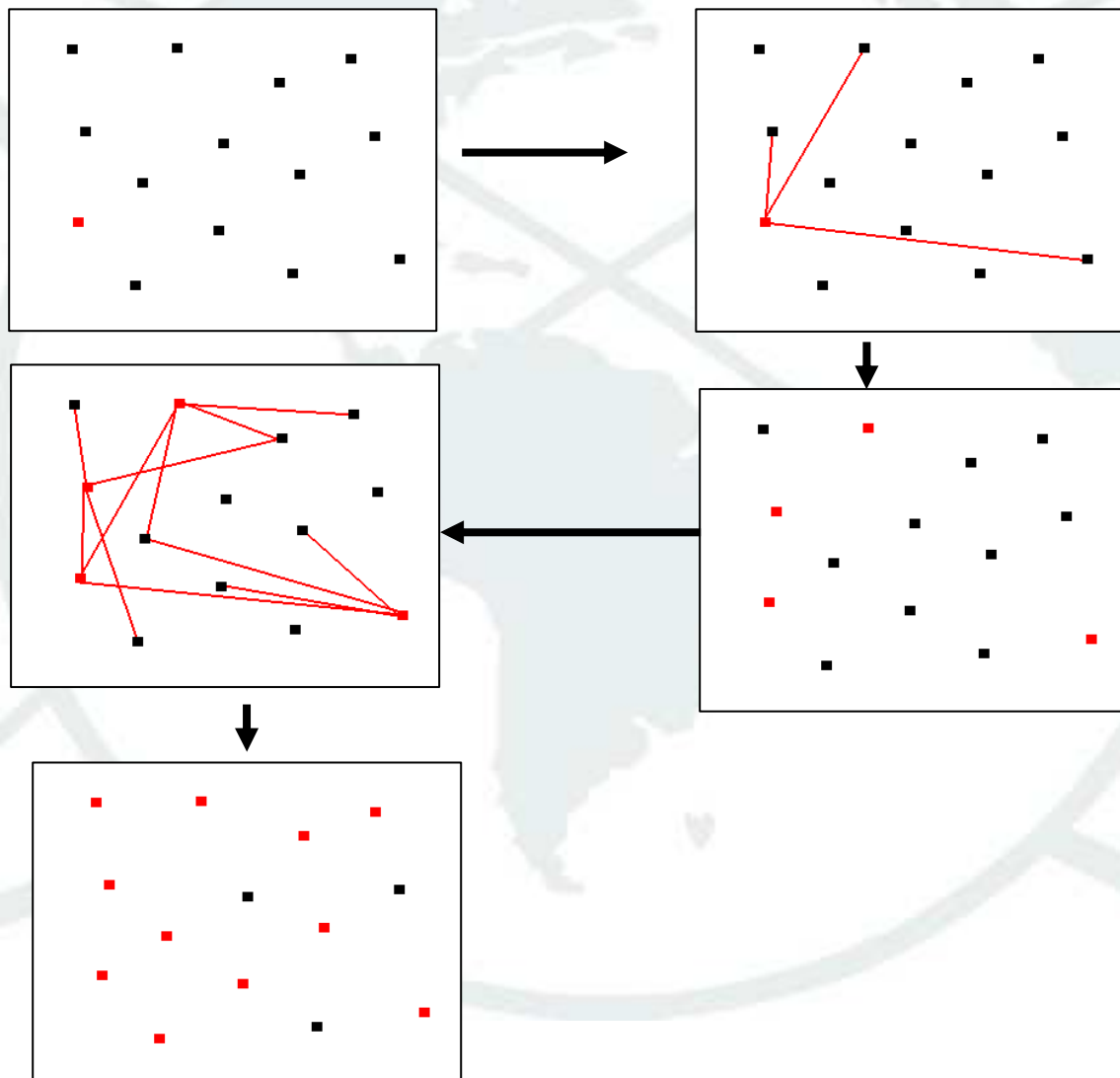
#### ***II.1. Leurs reproductions***

Les virus ont des modes de reproduction très variés, ils peuvent se propager sur des réseaux en exploitant des failles de sécurité (comme pour les Vers), se transmettre par mail à vos

listes de contact ou encore en infectant vos programmes et fichiers personnels.

Dans la majeure partie des cas nous pouvons observer de manière assez simple mais pas moins spectaculaire l'évolution et la multiplication de ces hôtes dans un réseau informatique.

Chaque point noir représente une entité informatique et point rouge une entité infectée. Nous considérons que la reproduction s'effectue par mail.



Comme nous le constatons en seulement deux phases de reproduction le virus a infecté la quasi-totalité de nos équipements.

Le système de réplication est assez simple et basique, les binaires (ou scripts) sont copiés tel quel sur les nouveaux hôtes et sont ainsi sujet à des bases de signatures virales.

## ***II.2. Leurs systèmes d'infections***

Une fois le virus arrivé sur la machine son objectif est d'y rester. Pour se faire il se copie dans toutes les zones pouvant lui permettre de démarrer. Les clés registre « run » sont des cibles privilégiées.

Une fois le processus du virus lancé il déploie les attaques à mener. Ce type de comportement est typique des premiers virus, il est simple et ne donnait pas trop de travail aux anti-virus pour désinfecter une machine (si anti-virus il y avait).

## **III. Les évolutions**

Aujourd'hui les virus sont majoritairement développés par des professionnels, il va sans dire que ces petits programmes ont subi de grands changements et qu'ils n'ont donc plus rien à voir avec leurs prédécesseurs.

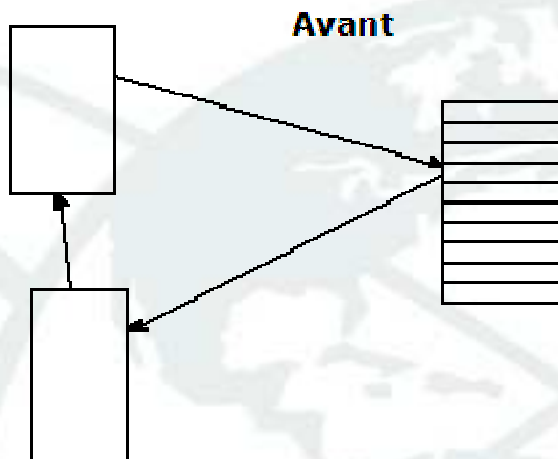
### ***III.1. Infection système***

#### **III.1.a) Type 1**

Avant les virus se cantonnaient souvent au mode utilisateur de l'ordinateur, maintenant ils s'infiltrèrent très fréquemment dans le noyau. Une fois dans celui-ci le code possède tous les droits (encore plus qu'un utilisateur Administrateur), il modifie alors toutes les bases du système d'exploitation pour se rendre le plus furtif possible.

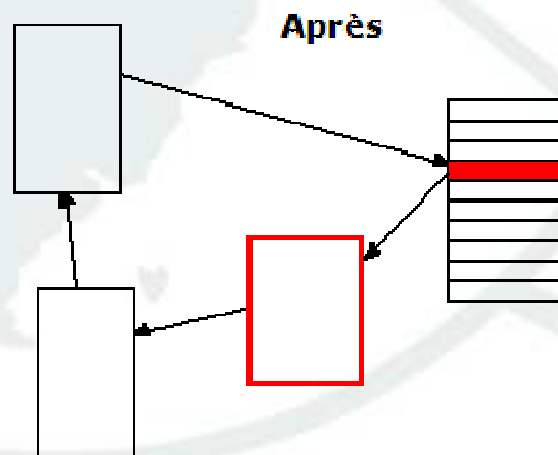
De nos jours le but d'un virus n'est plus principalement d'infecter le plus vite possible mais de perdurer. Ainsi les virus évoluent parallèlement aux rootkits qui eux ont comme unique objectif de rester indétecté sur un système.

Les méthodes utilisées pour rester invisible sont nombreuses et nous n'aborderons qu'un seul aspect de ses différentes faces, le hook. Un hook permet de détourner le flux d'exécution d'une application pour le rediriger vers un code illégitime. Dans un cadre de furtivité le code exécuté servira de filtre pour cacher certaines informations (c'est un exemple parmi tant d'autres).



Entre ces deux schémas nous voyons très clairement que nous avons intercalé un module dans la procédure de traitement. Ce module a été inséré en modifiant une référence en mémoire, cette référence a été copiée dans notre procédure de filtrage pour que notre module sache à

qui rendre la main. Avec une telle architecture nous intercepterons toutes les données devant être échangées et ce sans modifier la structure d'échange des informations. Le code malicieux peut maintenant effectuer tout type de filtrage en modifiant ou non les données sans être détecté ou faire planter le système.



Etudions maintenant un système couramment utilisé pour permettre aux virus de perdurer, la reproduction constante du code en local.

Le virus va infecter d'autres processus actifs pour leurs faire effectuer des actions de contrôle d'infection. Si le virus

commence à être désinfecté il régénérera les sections de code ayant été nettoyées.

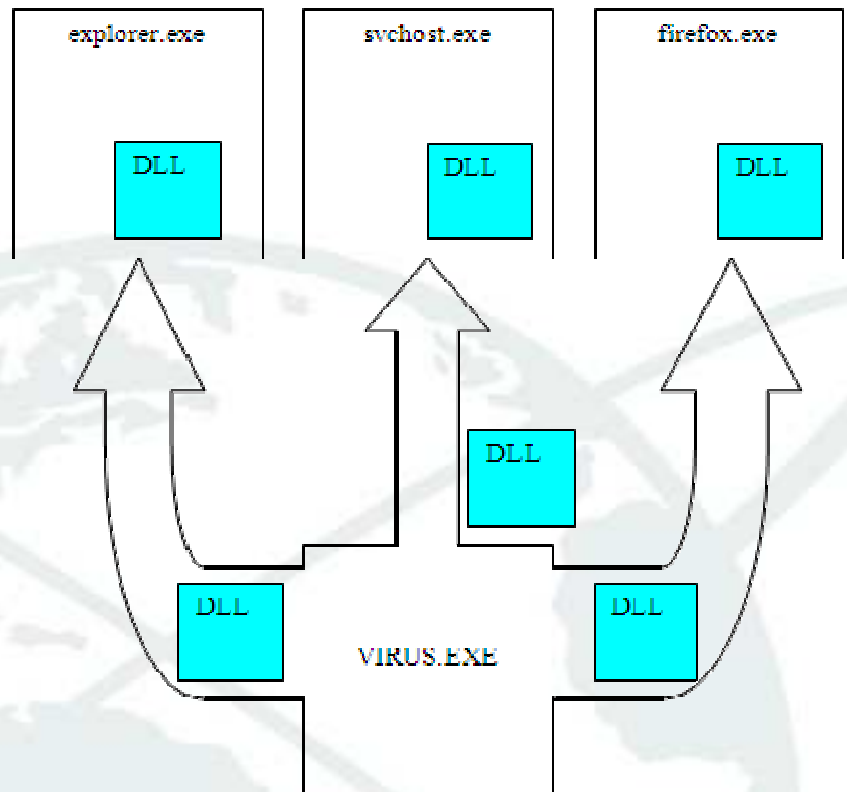
Les fichiers dll sont souvent employés pour injecter des processus, leur code relogeable donne une grande flexibilité de programmation.

De plus comme ces dll seront en cour d'utilisation par tous les processus elles ne pourront pas être supprimées. Les désinfections de ce type peuvent rapidement se transformer en casse tête si nous nous trouvons sur un serveur en production (qui par conséquent ne peut pas être redémarré).

Ces virus de type 1 ont encore beaucoup d'avenir car ils peuvent évoluer en utilisant les droits de l'utilisateur (des droits réduits).

### III.1.b) Type 2

Les virus de type 2 sont bien plus dangereux mais aussi plus dur à programmer que ceux de type 1. Dans le type 1 le malware cachait sa présence grâce aux modifications en noyau et dans son environnement utilisateur, mais en type 2 le virus évolue complètement en noyau ! Le danger est donc de taille car tous les outils utilisateur deviennent impuissant, de plus un anti-virus en noyau possèdera exactement des mêmes droits que notre malware, ils combattront donc à armes égales.



Afin de bien saisir la portée d'une infection totalement noyau rien de mieux qu'un bon schéma. Nous avons un ordinateur protégé par un firewall et un anti-virus :

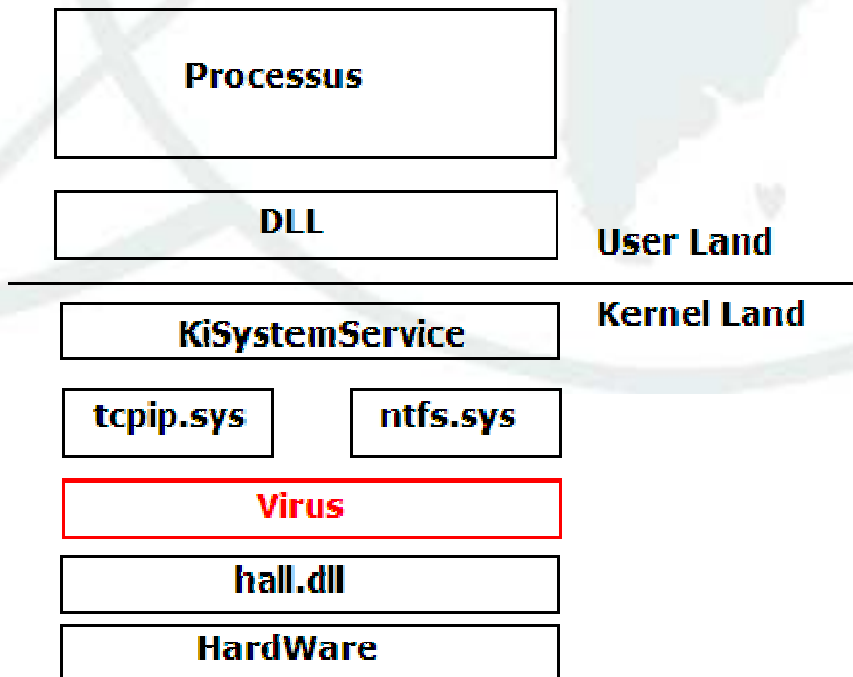
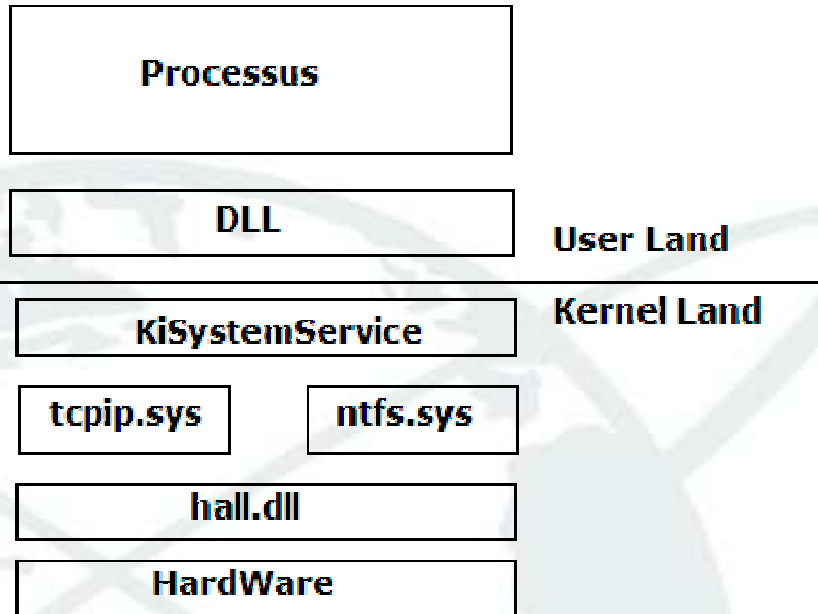
La représentation est très grossière et il manque beaucoup d'informations pour avoir une structure cohérente

(seulement la page ne suffirait pas). Comme nous nous en doutons les interactions se font du processus vers le Hardware pour revenir vers le processus.

Nous avons mis deux drivers, tcpip.sys et ntfs.sys qui servent respectivement à gérer les communications IP et TCP du système et (pour ntfs.sys) à décoder le formatage ntfs. Plus un virus se place proche du hardware et plus il sera difficile à détecter. Nous pouvons aussi noter que plus le nous sommes

proche du hardware et plus les possibilités de contrôles augmentent.

Le firewall vient se places juste au dessous (ou parfois juste au dessus) de tcpip.sys. Il peu alors contrôler toutes les interactions possibles à partir du protocole IP. Le traitement des



données de l'anti-virus vient fréquemment se placer juste au dessus de ntfs.sys. Il pourra donc avoir accès aux fichiers et les tester (par jeu de signature par exemple) afin de détecter d'éventuels virus.

Les virus de type 2 viennent eux se placer juste au dessus de hall.dll, de cette façon ils peuvent traiter les données avant le firewall ou l'anti-virus. Ainsi le malware peut émuler une deuxième pile IP (si besoin est) et interagir avec son propriétaire sans la contrainte du firewall. Si celui-ci intègre un système NTFS il peut aisément empêcher l'apparition d'un fichier sur le disque, ce fichier serai lancé avec le système puis caché par le rootkit.

Les deux types de virus que nous venons de présenter existent et sont en pleine expansion. Nous trouvons même sur internet des codes sources qui pourraient nous amener assez rapidement à la conception de tel programme.

### **III.2. Signatures**

Le point critique de la détection et l'éradication des virus se situe dans la capacité à savoir si le binaire est malveillant ou non avant son exécution. Les anti-virus ont donc créé des bases de signatures.

#### **III.2.a) Détection par signature**

Quand un nouveau virus est découvert une analyse de son code brute est lancée. A l'issue de cette analyse nous déterminons une séquence binaire placée à un endroit précis. Cette séquence nous servira à déterminer si le fichier que nous allons lancer (ou que nous analysons) a déjà été détecté come virus. Si oui nous lançons la procédure de suppression du programme, si non nous rendons la main au système.

Ce procédé est assez simpliste mais encore utilisé aujourd'hui. Je ne vous cache pas que cette méthode a un peu évolué depuis ses premières détections, mais elle reste très similaire.

### III.2.b) Polymorphisme

Les virus ont donc dut s'adapter pour ne plus être repérés. La méthode de contournement est le polymorphisme. Le

```
\xEB\x0F\x5E\x31\xC0\x38  
\x06\x74\x0D\x83\x36\x05  
\x83\xC6\x01\xEB\xF4\xE8  
\xEC\xFF\xFF\xFF\x50\x8C  
\xE0\x86\xE9\x45\xED\x03  
\x05\x05\x05\x55\x44\x50  
\x56\x40\x05\xED\x2E\x05  
\x05\x05\x46\x6A\x69\x6A  
\x77\x25\x53\x64\x69\x70
```

polymorphisme est l'action de donner plusieurs formes à un même code. Regardons notre premier exemple, ce code est capable de changer 99% de son code source en 126 formes différents. Si nous poussons un peu l'analyse nous pouvons distinguer deux blocs, le

premier en vert et un deuxième en rouge. Le premier est ce qui s'appelle un loader, il va servir à décrypter le second. Le code en rouge est lui encodé et ne peut être exécuté avec sa forme initiale. Il va donc subir les modifications du code en vert pour être ensuite lancé. Ce système réduit considérablement la marge de signatures possibles et peut s'adapter à une grande majorité des programmes existants.

```
\xEB\x0F\x5E\x31\xC0\x38  
\x06\x74\x0D\x83\x36\x05  
\x83\xC6\x01\xEB\xF4\xE8  
\xEC\xFF\xFF\xFF\x50\x8C  
\xE0\x86\xE9\x45\xED\x03  
\x05\x05\x05\x55\x44\x50  
\x56\x40\x05\xED\x2E\x05  
\x05\x05\x46\x6A\x69\x6A  
\x77\x25\x53\x64\x69\x70
```

Le gros point faible de cette méthode est qu'il reste toujours un bout du programme quasi statique sur lequel un anti-virus pourra s'appuyer. Il a donc fallu que les malwares trouvent une autre méthode pour échapper aux détections virales.

### III.2.c) Métamorphisme

Le métamorphisme est bien plus complexe que le polymorphisme. Son but n'est pas d'encoder un binaire pour le décoder mais de remplacer certains de ses comportements par des actions similaires mais légèrement différentes et qui donneront le même résultat.

Cette méthode demande une analyse approfondie du programme ainsi que la connaissance du processeur employé par la victime. Une fois ces deux données connus nous pouvons commencer l'analyse.

```
INC ECX
PUSH EBP
PUSH EBX
INC EBP
```

Nous avons ici une séquence de quatre instructions et notre objectif sera de les reproduire mais en les changeant. INC ECX revient à ajouter 1 à la valeur d'ECX, nous allons donc changer cette instruction par un ADD ECX, 1. Nous ne rentrerons pas dans le langage assembleur dans cet article, nous allons donc nous arrêter à ce simple constat, il est possible de réécrire des instructions sous plusieurs formes et qu'elles produisent un même résultat final. Dans notre exemple le code de sortie pourrait avoir des dizaines de signature différentes, nous vous avons mis l'une d'elles mais il y en a bien d'autres.

```
ADD ECX, 1h
SUB ESP, 4h
MOV DWORD PTR SS:[ESP], EBP
ADD ESP, -4h
MOV DWORD PTR SS:[ESP], EBX
SUB EBP, -1h
```

Des programmes entiers peuvent être repris en utilisant cette méthode et cette fois aucune signature ne sera possible.

## IV. Conclusion

Après ce tour d'horizon sur les virus et leurs évolutions nous ne pouvons que constater qu'ils donneront encore beaucoup de fil à retordre aux sociétés anti-virus. Nous sommes loin d'avoir des technologies à la hauteur de l'inventivité des développeurs de rootkits, de plus comme ceux-ci sont des professionnels ils garderont cette longueur d'avance. Il n'est pas nécessaire de prévoir un scénario apocalyptique non plus car les constructeurs anti-virus sont très réactifs aux nouvelles menaces et ont une forte capacité à évoluer.

Stéfan LE BERRE