Travail de maturité

Création d'un antivirus

Yoann Klingele

Professeur accompagnant : Samuel Vannay

INFOR_20_1_VA_03

14 septembre 2020

Lycée Collège Creusets

Abstract

Les dangers sur Internet sont bien souvent où on ne s'y attend pas : l'ouverture d'une pièce jointe, la consultation de certains sites Web. Il est primordial dans cette ère du numérique de se renseigner à ce sujet afin de ne pas être victime de ces menaces. C'est pour cela que dans notre travail, nous nous sommes intéressés à une partie de ses dangers, connus sous le nom de malware pour ensuite comprendre le fonctionnement d'un antivirus en vue de la réalisation d'un programme informatique capable de détecter des virus. Pour cela, il fallut de plus apprendre le langage de programmation *Python 3* à l'aide d'un cours en ligne afin de se lancer dans l'écriture du programme. Après plusieurs impasses et problèmes, le code était enfin opérationnel. Les résultats obtenus nous décrivent bien la complexité du monde des malwares et confirment donc cette vigilance importante à avoir lors de l'utilisation d'Internet.

Table des matières

Chapitre 1

Introduction

Plus de 4,6 milliards de personnes ont accès à Internet en 2020 [1]. Cela représente plus de la moitié de la population mondiale. Cet univers si vaste regorge toutefois de multitudes de dangers et continue malgré la prévention ainsi que les logiciels de sécurité informatique à causer des dégâts estimés à plusieurs milliers de milliards par année dans le monde [2]. Pour tenter de remédier à ces problèmes, les antivirus actuels développent des techniques de détection de virus de plus en plus sophistiqués afin d'étendre leur zone d'action.

Ce projet s'intéresse aux malwares ainsi qu'aux antivirus dans le but de créer un antivirus sur le logiciel "Python 3". Ce dernier ne sera malheureusement pas autant sophistiqué et efficace que les antivirus actuels, par faute de difficulté de confection ainsi que de réalisation sur un langage de programmation pour un débutant en la matière. Le code imitera le fonctionnement d'antivirus plus anciens, qui calculait la signature d'un fichier afin de la comparer à une base de données spécifique aux virus.

Dans ce travail, nous détaillerons tout d'abord les malwares pour ensuite relever les évènements historiques qui ont marqué l'évolution des antivirus. Une seconde partie sera consacrée aux virus dans le but de comprendre leur fonctionnement, élément capital en vue de leur destruction. Ensuite nous évoquerons les trois différentes méthodes de détection utilisées par les antivirus et mentionnerons les défauts de chaque approche. En définitive, la réalisation pratique de l'antivirus sera détaillée avec notamment plusieurs choix et difficultés rencontrés lors de la réalisation du code et finalement un regard critique sera porté sur ce logiciel, particulièrement sur ses performances.

Chapitre 2

Les malwares

Un malware est un terme générique qui englobe tous les programmes malveillants pouvant être dangereux pour un système. Il a comme fonction d'envahir ainsi qu'à mettre hors-service les ordinateurs et d'autres appareils informatiques tels qu'un téléphone ou encore une tablette. Le but principal d'un malware est de soutirer illégalement de l'argent à l'utilisateur, en volant ou supprimant les données sensibles présentes comme des mots de passe ou encore des données bancaires. La présence d'un malware peut entraîner des comportements anormaux tels que ceux-ci [3] :

- L'ordinateur est plus lent en utilisant Internet ou autres programmes.
- L'espace de stockage se rétrécit sans avoir téléchargé des fichiers volumineux.
- Le système bloque ou un écran bleu apparaît, ce qui arrive le plus souvent sur le système d'exploitation Windows.
- L'antivirus ne peut plus télécharger les nouvelles mises à jour et ne fonctionne plus correctement.
- L'ordinateur utilise une quantité anormale de ressources et le ventilateur tourne à pleine puissance.

Cependant, un malware peut malgré le bon fonctionnement apparent de l'ordinateur. Les malwares efficaces sont très bien cachés et opèrent au tréfonds de la machine. Le terme malware regroupe tous les logiciels malveillants dont voici les formes les plus fréquentes :

2.1 Les adwares

Un adware est un logiciel élaboré pour faire apparaître de la publicité sur l'écran, fréquemment sans l'utilisation d'un navigateur Internet. Ce dernier a souvent l'apparence dans programme normal pour maximiser les chances de téléchargement. Or, il ne faut pas confondre un adware avec de la publicité présente sur certains sites internet ou applications qui sont-elles contrôlées et associées avec ceux-ci. L'intérêt du développeur d'adware est financier : il peut se faire payer par des entreprises pour diffuser leurs publicités mais, il peut également vendre les informations de

fréquentation des sites internet à des tiers en vue de mieux cibler la publicité sur Internet.

2.2 Les spywares

Un spyware collecte essentiellement à propos de l'utilisateur, incluant les sites qu'il visite, ses téléchargements, les emails envoyés et reçus ou encore les mots de passe et informations bancaires. Il fonctionne en toute discrétion et peut surveiller les activités de l'ordinateur pour en soutirer des données sensibles, évoquées précédemment. Une fois découvert, il reste cependant difficile à désinstaller de par sa capacité à agir sur la machine.

2.3 Les chevaux de Troie

Tiré du célèbre poème de Virgil, "L'Énéide", le cheval de Troie utilise comme son homologue grec la tromperie et l'apparence en exécutant des programmes à première vue inoffensifs mais qui cachent en réalité un malware redoutable. Ce dernier est autonome et sert notamment à diffuser différentes menaces, comme des spywares ou autre malwares dans l'ordinateur infecté. Des comportements malavisés comme le téléchargement d'applications crackées ou de programmes gratuits inconnus peuvent être la cause d'une présence d'un cheval de Troie [4].

2.4 Les ransomwares

Le ransomware, aussi appelé malware de rançonnage, bloque l'accès aux fichiers personnels ou au système d'exploitation à l'utilisateur et exige une rançon si celui-ci veut récupérer l'accès complet de sa machine. Ce malware se propage principalement avec les spams malveillants, composés d'e-mails indésirables ainsi qu'une pièce jointe contenant le ransomware. Il existe néanmoins plusieurs types de ransomwares, variants selon leur sévérité [5] :

2.4.1 Les scarewares

Cette technique inoffensive consiste à faire ouvrir une fenêtre web, par exemple, qui informe l'utilisateur que son ordinateur est infecté et que l'unique moyen de le retirer et de payer une somme donnée. Si l'utilisateur ne paye rien, son seul risque est que de nouvelles fenêtres de ce type réapparaissent, car aucune société commercialisant des antivirus n'aborderait jamais des clients de cette façon et de plus tant qu'un antivirus n'est pas installé, il ne peut pas détecter des virus.

2.4.2 Les verrouilleurs d'écran

Ce type de ransomware bloque totalement l'accès à l'ordinateur et affiche uniquement une fenêtre qui contient une reproduction d'une institution gouvernementale,

comme la CIA ou autres instances de justice. Il nous prévient qu'une activité illégitime a été décelée et qu'une amande doit être payée sur-le-champ. Il est néanmoins possible de payer la rançon, mais il n'y a aucune garantie que le rançonneur libère l'accès. Un expert en cybersécurité peut essayer de s'en débarrasser mais là encore, aucune garantie qu'il y parvienne selon l'efficacité du verrouilleur.

2.4.3 Les chiffreurs

Les chiffreurs peuvent s'avérer très dangereux car ils volent les fichiers sensibles et les chiffrent pour les rendre alors inaccessibles pour l'utilisateur. L'auteur du malware demande une rançon contre la restauration des fichiers. Malheureusement, personne ne pourra récupérer les fichiers et l'unique moyen d'espérer revoir ces derniers est de payer la rançon sans aucune garantie qu'ils seront de nouveau accessibles.

Le ransomware "WannaCry"

Cette attaque du ransomware chiffreur est apparu en le 12 mai 2017 et fit rapidement le tour du globe grâce à une faille dans le système d'exploitation Windows appelé "EternalBlue". Celle-ci a été développée par la NSA mais a fuitée en avril 2017, juste un mois après la mise à jour des vulnérabilités de Windows. Il laissait 3 jours à l'utilisateur pour payer un montant allant de 300 $ jusqu'à 600 $. Puis, au-delà des 3 jours, le montant était doublé et au bout du 7^e jour les fichiers chiffrés étaient détruits. Ce malware a seulement duré 4 jours, du à la réaction rapide de Microsoft qui stoppa sa propagation en corrigeant le problème lié à "Eternal Blue", mais toucha environ 200'000 ordinateurs à travers 150 pays. Parmi ceux-ci, il infecta notamment un ordinateur de la gare de Francfort-sur-le-Main, chargé d'afficher les différents horaires de train. Les bénéficiaires de ce malware auraient, jusqu'au 3 août 2017, reçu 338 paiements d'une valeur totale de 142'361.51$ selon un programme mis en place par Keith Collins, et qui recensait toutes les transactions du compte [6], reçues en bitcoin, qui est une cryptomonnaie. En décembre 2017, le Royaume-Uni et les États-Unis ont déclaré la Corée du Nord responsable de ce malware [7].

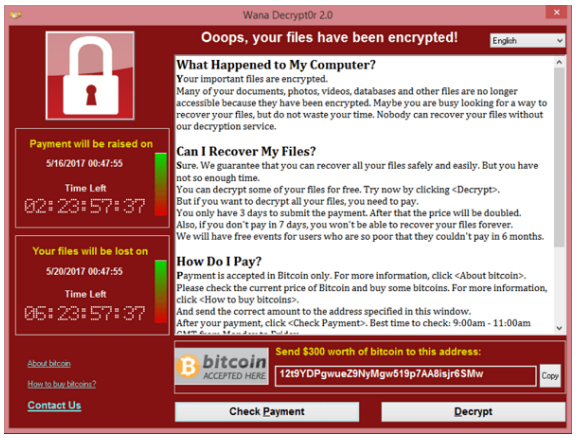

FIG. 1 : Photo du ransomware sur un écran de la gare de Francfort-sur-le-Main

2.5 Les virus

Par définition, un virus informatique est un programme informatique réplicatif qui à l'origine était inoffensif mais de nos jours principalement malveillants et avec comme but de se répandre par tous les moyens technologiques : disques durs, mails,

téléchargements, clés USB...

Le code du virus est créé pour changer le comportement de la machine hôte : il s'insère ou s'attache à un fichier ou à un programme fiable. Il reste dormant jusqu'à ce que l'utilisateur lance le programme ou le fichier qui contient le virus, ce qui exécutera le code du virus au lieu du programme voulu. Une fois lancé, ce dernier peut infecter d'autres appareils connectés au même réseau, voler des datas ou des mots de passe, corrompre les fichiers ou encore prendre le contrôle total de la machine sans que l'utilisateur ne puisse rien faire. Ils possèdent également une signature qui est propre à chaque virus et permet ainsi son identification : celle-ci est un algorithme qui est représenté par une série de chiffre généralement composée d'hexadécimales.

Il est primordial d'y prêter une forte attention car dans notre monde constamment connecté, on peut faire face à des virus dans nos mails, les fichiers joints, des liens Internet ou encore sur des applications mobiles. Voici un exemple d'un célèbre virus pour réaliser les dégâts non négligeables de ce dernier.

2.5.1 Le virus "ILOVEYOU"

"ILOVEYOU" est un virus apparu le 4 mai 2000, à Manille, aux Philippines. En seulement quelques jours, le virus parcourut le monde entier et toucha notamment le Pentagone, la CIA et de grandes entreprises de cosmétiques et d'agroalimentaires.

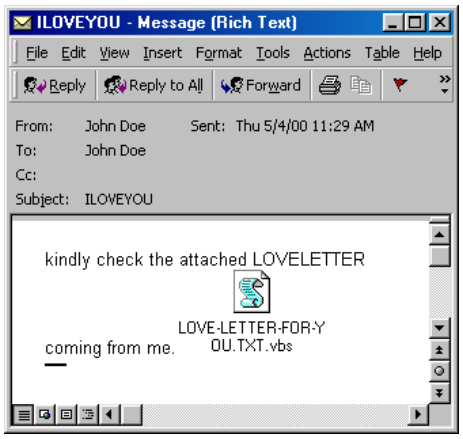

FIG. 2 : Mail original reçu dans une boîte mail à l'époque.

Le principe était simple : sur le système d'exploitation "Windows 9x" un mail pouvait apparaître avec comme message "consulte cette lettre d'amour ci-jointe" avec un fichier nommé "LOVE-LETTER-FOR-YOU.TXT.vbs"

La plupart des personnes, intrigués par cette lettre, ont ouverts ce fichier. Or, ce n'était point un fichier texte mais un *.vbs*, tirant son abréviation de *VBScript*, qui est un langage de programmation. Le logiciel détruisait la mémoire de l'ordinateur en supprimant tout type de fichiers : photos, vidéos, musiques etc... De plus, le virus recherchait la base de données des adresses mails Outlook pour ensuite propager le virus en envoyant le même mail reçu à ces dernières. Le nombre de périphériques touchés est chiffré à plus de 10 millions, ce qui représentait environ 10% des ordinateurs connectés sur le globe à l'époque. Les dommages économiques, quant à eux sont estimés entre 5,5 et 8,7 milliards de dollars.

Le plus étonnant est que l'auteur de ce virus, Onel de Guzman, identifié quelques jours après le lancement, s'en sortit indemne car la loi des Philippines n'interdisait pas encore la programmation de malwares. Cet incident entraina en juillet 2000 l'introduction de la loi "d'E-Commerce" englobant ainsi un grand nombre d'activités dangereuses en ligne [8].

2.6 Les vers

Les vers sont un type de malwares similaires aux virus mais ils ne requièrent aucun un autre programme pour se propager.

2.6.1 Le ver “Stuxnet”

Ce ver informatique fut découvert en juin 2010 et aurait été développé par la NSA et avec l'aide de l'unité de renseignement israélienne, avec comme cible les centrifugeuses iraniennes d'enrichissement d'uranium. Le ver aurait modifié la vitesse de rotation des turbines de la centrale, entrainant leur dégradation et leurs explosions [9]. Ce serait selon des chercheurs de Symantec, le premier ver conçu non seulement pour espionner des industries, mais aussi pour les contrôler à distance [10].

Chapitre 3

Les antivirus

Par définition, un antivirus est un logiciel informatique servant à anticiper, détecter et détruire les virus.

3.1 Survol historique

L'histoire de l'antivirus est marquée par plusieurs dates ainsi que périodes clés à son développement au fil des années.

Pour commencer, le fonctionnement même d'un virus avait été imaginé en 1949, par John von Neumann, qui avait comme objectif de créer une machine capable de s'auto-répliquer et ceci à chaque fois de manière plus complexe. Il publia par la suite le fruit de ses recherches avec "Theory of Self-Reproducing Automata". En 1970, Gregory Benford publia "The Scarred Man" une courte histoire de science-fiction dans laquelle il évoque un programme nommé "VIRUS" décrit comme un code malicieux se cachant sur des ordinateurs et pouvant se répandre rapidement et où les créateurs du programme, en avaient fait un business en supprimant le virus de l'ordinateur du client en échange d'argent. Le livre ne connut pas fort succès, mais le virus décrit reste remarquablement semblable à ceux présents de nos jours [11].

Le premier vrai virus informatique connu se dénommait "Creeper". Créé par Bob Thomas en 1971, ce virus n'agissait pas de manière dangereuse pour la machine infectée : son unique pouvoir était d'afficher un message sur l'écran de l'utilisateur qui stipulait :"I'm the Creeper : catch me if you can" qui, une fois traduite donne : "Je suis le Creeper : attrape-moi si tu peux". Il peut s'avérer étonnant que ce logiciel ne puisse que faire apparaître un texte, mais à l'époque les ordinateurs ainsi que les logiciels de programmation en sont eux aussi à leurs débuts. Malgré le fait que ce logiciel est inoffensif, il a été majoritairement accepté comme le premier virus du fait de sa capacité à s'auto-répliquer, fonction propre aux virus. La même année, en réponse à ce premier virus quel que peu provocateur, Ray Tomlinson, un programmeur qui a notamment créé le premier système capable d'envoyer un mail entre deux utilisateurs, créa un prototype nommé "Reaper" qui avait comme unique capacité de supprimer le logiciel "Creeper". Il est donc considéré comme le premier antivirus de par sa fonction mais il est en fait lui-même aussi un virus, de par son code. Au cours de la même décennie le nombre de virus ne cesse d'augmenter.

De même dans les années 80, où de plus en plus de programmeurs se spécialisent dans le codage de virus, en les améliorant et en leur ajoutant des fonctionnalités malveillantes : avec la commercialisation des ordinateurs privés, les hackers voulaient contrôler cette nouvelle technologie qui n'avait pas vraiment de défense solide. C'est seulement en 1983 que Fred Cohen définit un virus, terme qui été auparavant substitué par un programme [12]. Puis commença la lutte pour commercialiser le premier antivirus du marché, qui allait connaître un succès énorme au vu de la démultiplication des virus : en 1987, Bernd Fix publia le 1er logiciel documenté capable d'anéantir un virus, mais non commercialisé. Néanmoins, la même année, Andreas Lüning et Kai Figge, fondateurs de la société "G Data Software", dévoilèrent leur produit destiné à éliminer les virus sur les appareils "Atari ST". Pour concurrencer ces nouvelles venues sur le marché, toujours en 1987, "L'Ultimate Virus Killer" est lui aussi commercialisé et sera par la suite l'antivirus standard pour la gamme d'ordinateurs "Atari" et sa neuvième et dernière version fut publié en 2004 [13]. McAfee, 3^e entreprise sur le marché de l'antivirus en octobre 2019 [14], a elle aussi été créée en 1987 qui est incontestablement une année-charnière dans l'histoire de l'antivirus.

Dans les années 90, les virus connaissaient une croissance spectaculaire : AV-TEST, entreprise spécialisée dans la sécurité de logiciel, dénombrait en 1994 28'613 signatures de virus dans leur base de données puis en 1999, le chiffre s'éleva à 98'428. Mais cette évolution impressionnante ne s'arrêta pas là : en 2005, AV-TEST répertoriait 333'425 signatures. En 2007, cette même entreprise enregistrait 5'490'960 nouveaux échantillons de virus. Ce nombre impressionnant de virus implique une réaction des logiciels de sécurité informatique, qui ont dû s'adapter et inventer de nouvelles méthodes afin d'être le plus efficace possible.

De nos jours, les répercussions financières annuelles causées par les virus se factures à plusieurs milliards de dollars et représentent donc une filière intéressante pour les cybercriminels. Il est donc primordial de s'en protéger et de s'informer.

3.2 Détection de virus

Intéressons-nous désormais aux moyens mis en œuvre par les antivirus pour repérer un virus.
Il existe trois types principaux de détection : le Signature-based, le Behaviour-based et l'analyse heuristique.

3.2.1 Signature-based

Cette technique est pour commencer, celle qui a été le plus répandue lors des débuts de l'antivirus. En effet, le système est plutôt simple : le logiciel calcule la signature d'un fichier et la compare avec d'autres signatures appartenant à des virus répertoriés dans une base de données propre à chaque antivirus. Les chercheurs doivent donc établir les signatures issues de virus pour maintenir l'antivirus à jour. L'inconvénient principal de cette méthode est qu'elle ne peut pas reconnaître des virus non identifiés. C'est notamment de ce fait que les pirates informatiques créèrent des virus métamorphiques : lors de leur utilisation, un logiciel intrinsèque modifie le

virus ainsi que son propre logiciel métamorphique mais tout en gardant les mêmes fonctionnalités malveillantes. Cela rend donc la signature du virus obsolète après chaque usage et le rendant donc introuvable par les antivirus utilisant uniquement cette technique.

3.2.2 Behaviour-based

Cette méthode de recherche consiste à contrôler le comportement d'un fichier pour évaluer si celui-ci est potentiellement dangereux ou non.

Une Sandbox est un dispositif destiné à effectuer des tests sur un fichier dans un environnement sécurisé, simulant exactement la machine hôte : mémoire, processeur, etc... le but étant d'exécuter le fichier suspect et d'analyser si son comportement sur le système est néfaste ou non. La faille de ce processus réside dans le fait que certains virus détectent lorsqu'ils sont dans une sandbox, et par conséquent agissent de manière totalement passive. Les pirates informatiques utilisent aussi des formats de fichiers très peu utilisés pour que l'environnement artificiel ne puisse pas lancer le programme ou alors, ils font en sorte que le fichier soit trop volumineux pour qu'il ne puisse pas être lu par la sandbox [15].

3.2.3 L'analyse heuristique

Cette approche a été créée pour contrer les virus métamorphiques, évoqués précédemment. Elle utilise un système expert, capable de simuler les décisions humaines : le programme va comparer le code du fichier suspect à des codes de virus connus, répertoriés dans une base de données. Si une similitude est repérée, l'utilisateur est immédiatement prévenu que le fichier est potentiellement dangereux. Le désavantage de ce procédé provient des fausses alertes qu'il peut provoquer. Il est donc préférable de coupler cette méthode avec celles citées précédemment pour une meilleure efficacité.

Chapitre 4

Création d'un antivirus :

Abordons désormais une partie plus pratique avec la réalisation d'un antivirus sur l'éditeur de code *Python 3*.

4.1 Réalisation technique

4.1.1 Apprendre la programmation

Étant un débutant en codage, il était primordial d'apprendre les bases du langage de programmation *Python 3* assez rapidement. Pour cela, un cours en ligne proposé par OpenClassrooms, composé de 40 chapitres, a permis un survol global mais toutefois assez détaillé de ce langage. Il était composé de travaux à réaliser dont voici un exemple détaillé :

Le "ZCasino"

L'exercice a comme consigne de simuler une partie de casino avec ces paramètres : le joueur doit initialement disposer de 1000 $. Il choisit un nombre compris entre 0 et 49 et mise une partie de son argent. Si le nombre choisi correspond à celui généré aléatoirement par le code, le joueur triple sa mise, autrement, il la perd. Le but de ce travail personnel est d'utiliser et comprendre les variables tel que `argent = 1000`, les exceptions avec `try:` et `except:ValueError` ainsi que les boucles `while:`. La difficulté rencontrée lors de ce premier code était la syntaxe même de *Python 3* : il fallait faire attention à l'indentation qui malgré sa grande importance peut être vite oubliée, tout comme les deux-points qui suivent les `if, else, while` etc...

La création d'un pendu faisait également partie du cours et une dernière activité pratique consistait à créer un dictionnaire ordonné qui lui était principalement centré sur l'utilisation des listes. Les cours d'OC Informatique ainsi que d'Applications des Mathématiques ont tous les deux abordé *Python 3* ce qui nous a permis de consolider les bases acquises et de notamment découvrir de nouveaux horizons tels que le module *turtle* ou encore le module *Jupyter* sur un *Raspberry Pi*.

4.1.2 Le choix de la méthode de détection

Après avoir appris les bases de la programmation, il a fallu choisir une méthode à imiter pour notre antivirus. Trois possibilités subsistaient donc : le Signature-based, le Behaviour-based ou l'analyse heuristique. Le choix s'est logiquement porté sur la méthode de Signature-based : elle était réalisable avec beaucoup d'éléments de *Python 3* déjà connus tandis que les deux autres méthodes étaient beaucoup trop complexes à réaliser avec notamment la surveillance des activités des fichiers.

4.1.3 Les particularités du code

Le code, ci-joint en annexe, possède quelques particularités qui vont être détaillées :

Le hasher

```
hasher = hashlib.sha256()
with open(str(cwd), 'rb') as afile:
    buffer = afile.read(BLOCKSIZE)
    while len(buffer) > 0:
        hasher.update(buffer)
        buffer = afile.read(BLOCKSIZE)
    mon_hash = open("hash.txt","w")
    mon_hash.write(hasher.hexdigest())
    mon_hash.close()
```

Le hasher est la partie du code qui va s'occuper d'obtenir la signature de chaque fichier pour ensuite la stocker dans la variable `mon_hash`. Il a été choisi en sha256, un algorithme de hashing spécifique, pour qu'il coïncide avec les signatures répertoriées dans la base de données, qui sont elles aussi en sha256. Cette partie utilise une autre variable `BLOCKSIZE`, définie au début du code. Elle sert à éviter que la fonction `afile.read(BLOCKSIZE)` charge des fichiers trop volumineux dans la mémoire de l'ordinateur, ce qui peut s'avérer dangereux pour ce dernier. La limite fixée par la variable est de 65536 car cela correspond à la valeur standard dans le stockage informatique [16].

Le format du fichier ou dossier demandé

Lorsqu'il est demandé à l'utilisateur d'indiquer le fichier qu'il souhaite vérifier, avec `fichier = input(" Entrez le chemin du fichier/dossier à analyser.")`,il lui est spécifié de rentrer le chemin d'accès du fichier. Celui-ci est le lieu précis dans lequel il est stocké. Il a été choisi de demander le chemin d'accès du fichier ou dossier pour deux raisons :

1. Il se peut que deux fichiers aient le même nom mais qu'ils ne soient pas stockés dans le même dossier. Si uniquement le nom était alors demandé, le programme pourrait choisir le mauvais. Il est donc préférable de préciser le chemin d'accès du fichier pour éviter ce problème car dans un même dossier, deux fichiers ne peuvent avoir le même nom.

2. Récupérer directement le chemin d'accès ne demandait pas par la suite d'étapes intermédiaires pour convertir le nom d'un fichier en son chemin d'accès, qui est obligatoire avec l'utilisation de la fonction `if os.path.isdir(fichier):`.

4.1.4 La base de données

Essentielle au bon fonctionnement de l'antivirus, cette dernière est composée de 174 signatures, tirées de l'algorithme de hashing sha256. Ces dernières correspondent toutes à de vrais virus plus ou moins anciens. Ces signatures sont tirées d'un project GitHub.[17]

4.1.5 Les difficultés rencontrées

Il est arrivé lors de la réalisation de ce programme informatique que certaines parties deviennent problématiques voire énigmatiques, voici donc les plus marquantes :

Lister tous les fichiers d'un dossier

N'ayant aucune idée de comment procéder pour cette partie, une recherche Internet "file in folder python" s'imposa. Les codes trouvés n'étaient pas satisfaisants : ils répertoriaient bien tous les fichiers d'un dossier, mais si celui-ci contenait des sous-dossiers, tous les fichiers présents dans ces derniers n'étaient pas pris en compte. Une idée était d'emboiter deux de ces systèmes donnant ce code :

```
fichier = input("Quel est le fichier/dossier à analyser?")
if os.path.isdir(fichier):
    for file in glob.glob("*"):
            if os.path.isdir(file):
                for file1 in glob.glob("*"):
                    print(file1)
```

Malheureusement, ce code ne fonctionnait que pour les premiers sous-dossiers, or le logiciel doit pouvoir être capable d'aller chercher un fichier contenu dans mille sous-dossiers. Après bien des jours à tenter de différentes techniques en vain, un adverbe résolut à lui seul le problème : **récursivement**. Lorsque ce mot fut ajouté à la recherche Internet, un code satisfaisant pû alors être utilisé :

```
for root, directories, filenames in os.walk(fichier):
    for filename in filenames:
        print(filename)
```

Analyser tous les fichiers

Juste après avoir réussi à lister tous les fichiers, un autre problème apparut : lorsque le programme appliquait la fonction hashing aux fichiers, il n'arrivait pas à le faire à ceux présents dans les sous-dossiers de celui indiqué par l'utilisateur. L'erreur renvoyée a la fonction `except FileNotFoundError`, ce qui était étrange car le code

précédant listait bien correctement tous les fichiers. Une vérification du chemin d'accès des fichiers était donc nécessaire et ceci à l'aide de cette commande : `print(os.path.abspath(filname))`. Pour un fichier "abc.txt" qui avait comme chemin d'accès intial : C :\Users\yoann\Desktop\test\tset\abc.txt, le programme ressortait lui ce chemin d'accès : C :\Users\yoann\Desktop\test\abc.txt. Le sous-dossier "tset" à été supprimé du chemin d'accès, car sans aucun ajustement, le programme n'ajoute pas les sous-dossiers aux chemins d'accès des fichiers. Il fallut alors trouver une fonction capable de le faire. Après avoir recherché dans la documentation de la fonction `os.path`, deux fonctions combinées ont réglé ce problème :

```
path = os.path.dirname(filename)
cwd = os.path.join(root, filename)
```

L'emplacement de la base de données

L'objectif était de copier le fichier *.txt* de la base de données dans le dossier ou sous-dossier à analyser car lorsque le programme tentait d'analyser un fichier provenant d'un sous-dossier, il ne reconnaissait plus la base de données présente dans le même dossier que celui analysé. Malgré plusieurs tentatives avec la fonction `shutil.copyfile(original, target)`, la base de données n'a pas pu être copiée. Il fallut donc changer d'approche. En faisant des recherches à ce sujet, une solution toute simple mais auparavant inconnue apparue : Il n'est pas nécessaire de copier la base de données, il suffit qu'elle se trouve dans le même dossier que le programme Python et le code la trouvera peu importe quel fichier il traite. C'est pourquoi il est primordial de regrouper la base de données et le programme *.py* dans un même dossier.

4.2 Discussion

Dans cette dernière partie, nous allons porter une vue d'ensemble sur les capacités de notre programme.

Du fait de sa méthode de détection, notre antivirus ne pourra pas détecter les virus non répertoriés. Pour cela il faudrait lui implémenter d'autres modes de reconnaissance tels que l'analyse heuristique et le behaviour-based. La base de données devrait contenir plus d'éléments pour améliorer l'efficacité de l'antivirus,car elle ne contient que 174 signatures, ce qui représente une infime partie des virus connus. Pour rappel une base de données similaire à la nôtre était composée en 1994 de 28'613 éléments alors qu'en 2020, c'est environ 1.1 milliard de malwares qui sont répertoriés. Cette base de données enregistre en moyenne 350'000 nouveaux malwares par jour ce qui rend difficile de tenir à jour une base de données comme la nôtre. Pour le processus d'analyse du programme est assez lent : Il lui a fallut vingt secondes pour analyser 3'009 fichiers, ce qui revient à environ 150 fichiers par seconde avec un ordinateur relativement puissant. Cela vient du fait que la machine doit comparer une signature à 174 autres et cela pour chaque fichier. Cependant, le comparateur peut être amélioré avec l'emploi d'une base de données externe au format "SQLite", dont le code ferait appel pour lui soumettre la signature. Le logiciel en charge de la base de

données irait ensuite procéder à la comparaison du fichier à ceux répertoriés, mais avec des techniques bien plus optimisées et ciblées que de comparer chaque élément à la signature suspecte. Le code a néanmoins l'avantage de pouvoir traiter tout type de fichier là où d'autres méthodes n'en sont pas capables.
Il faut néanmoins préciser que l'efficacité de l'antivirus n'était pas un facteur important, mais le but de ce projet était de combiner les connaissances acquises sur *Python 3* ainsi que celles sur la cybersécurité, dans ce cas-là la méthode signature-based, attribuant à ce projet une valeur de "proof of concept". Le programme avait néanmoins plus de sens avec base de données réelles que d'utiliser la signature de fichiers test qui aurait pu être de simples documents Word ainsi que des fichiers texte. Cependant, les parties mal optimisées de ce programme montrent bien la complexité à réaliser un antivirus efficace de nos jours.

Chapitre 5

Conclusion

En définitive, nous avons au cours des neuf derniers mois développés et approfondis nos connaissances dans le domaine de la cybersécurité, les malwares ainsi que la programmation pour ensuite les mettre à profit dans la création d'un antivirus *Python 3*. Les principes de ce langage de programmation, inconnus au début du projet, ont dû rapidement être intégrés grâce à plusieurs exercices de programmation. Puis, nous nous sommes penchés sur une partie plus théorique en abordant les dangers présents sur Internet en se focalisant sur les types de malwares les plus fréquents sur le Web. Nous avons ensuite étudié le fonctionnement d'un antivirus, avec en particulier ses méthodes de détection de virus, afin d'en appliquer une dans notre programme informatique. Dès lors, la réalisation de ce dernier pû être entamé. Parsemé d'embuches, la réalisation requis de la persévérance mais fut au final satisfaisante car elle mena à un programme certes peu optimisé, mais qui remplit l'objectif de reproduire le fonctionnement d'un antivirus.

Avec tous les dangers que comporte Internet dont seulement une partie a été évoquée dans ce travail ainsi que l'informatisation de plus en plus récurrente dans notre société, plusieurs réflexions peuvent être alors mentionnées : ne devrions-nous pas accorder une place plus grande à l'informatique et à la prévention de ses dangers dans nos écoles ? Les langages informatiques ne sont-ils pas de nos jours, autant importants que les langages plus "classiques" comme l'anglais ou l'allemand ?

Yoann Klingele
Sion, le 14 septembre 2020

Chapitre 6

Bibliographie

[1] *Internet Live Stats - Internet Usage & Social Media Statistics.* en. https ://www.internetlivestats.com/. (consulté le 03/04/20)

[2] *Cybercrime Damages $6 Trillion by 2021.* en-US. Fév. 2018. (consulté le (05/08/20)

[3] *Qu'est-ce qu'un malware ?* fr. https ://fr.malwarebytes.com/malware/. (consulté le 06/09/20)

[4] *Cheval de Troie : de quoi s'agit-il ? Virus ou malware ?* fr. https ://fr.malwarebytes.com/trojan/. (consulté le 06/09/20)

[5] *Ransomwares : de quoi s'agit-il et comment s'en débarrasser.* https ://-fr.malwarebytes.com/ransomware/. (consulté le 06/09/20)

[6] *actual ransom sur Twitter.* fr. https ://twitter.com/actual_ransom/status/892928051360784384. (consulté le 07/09/20)

[7] "WannaCry Ransomware Attack". en. In : *Wikipedia* (consulté le 05/09/20).

[8] "ILOVEYOU". en. In : *Wikipedia* (consulté le 11/08/20).

[9] "Stuxnet". fr. In : *Wikipédia* (consulté le 05/09/20).

[10] Glenn Kessler. "Computer Worm May Have Targeted Iran's Nuclear Program". en-US. In : (nov. 2010). issn : 0190-8286. (consulté le 07/09/20)

[11] *Advent of Computing* http ://adventofcomputing.com/.(consulté le 23/08/20)

[12] Yves Gr et montagne. *Sécurité : le virus informatique a 30 ans.* fr-FR. https ://www.silicon.fr/securite-virus-informatique-30-ans-90886.html. Nov. 2013. (consulté le 19/08/20)

[13] "Antivirus Software". en. In : *Wikipedia* (consulté le 15/03/20).

[14] *Symantec, ESET et McAfee dominent le marché des logiciels anti-malware*

Windows. fr. https ://www.zdnet.fr/actualites/symantec-eset-et-mcafee-dominentle-marche-des-logiciels-anti-malware-windows-39894049.htm.(consulté le 20/08/20)

[15] *What is Sandbox Security ?* en. https ://www.forcepoint.com/cyber-edu/sandboxsecurity. (consulté le 09/04/20).

[16] *Hashing Files with Python* en. https ://www.pythoncentral.io/hashing-files-with-python/ (consulté le 12/03/20)

[17] *ytisf/theZoo* en. https ://github.com/ytisf/theZoo/tree/master/malwares (consulté le 13/06/20)

Chapitre 7

Annexes

1. Le code complet de l'antivirus :

```
import hashlib
import os
repcor = True
clean = True
BLOCKSIZE = 65536
def hashing(cwd):
    global clean, repcor
    hasher = hashlib.sha256()
    with open(str(cwd), 'rb') as afile:
        buf = afile.read(BLOCKSIZE)
        while len(buf) > 0:
            hasher.update(buf)
            buf = afile.read(BLOCKSIZE)
        mon_hash = open("hash.txt","w")
        mon_hash.write(hasher.hexdigest())
        mon_hash.close()

        fichier1=open("hash.txt","r")
        fichier2=open("bdd.txt","r")

        for line1 in fichier1:
            for line2 in fichier2:
                if line1==line2:
                    repcor = True
                    clean = False
                    while repcor :
                        suppr = input("Le fichier " + os.path.basename(cwd) +
                        "contient un virus, voulez vous supprimez ce fichier?
                        "(oui/non) \n")
                        if suppr == "oui":
                            print("Le fichier " + os.path.basename(cwd) +
                                  "a été supprimé.")
```

```
                                afile.close()
                                os.remove(cwd)
                                repcor = False
                                break
                            if suppr == "non":
                                print("Le fichier "+ os.path.basename(cwd)+
                                      "a été conservé.")
                                repcor = False
                            else:
                                print("Veuillez écrire oui ou non !")
                f1.close()
                f2.close()

    fichier = input("Entrez le chemin du fichier/dossier à analyser.\n")
    try:
        if os.path.isdir(fichier):
            for root, directories, filenames in os.walk(fichier):
                for filename in filenames:
                    path = os.path.dirname(filename)
                    cwd = os.path.join(root, filename)
                    hashing(cwd)
            if clean == True :
                print("Aucun virus détecté")
        else:
            hashing(fichier)
            if clean == True :
                print("Aucun virus détecté")
        print("Tous les fichiers ont étés traités")
        os.remove("hash.txt")
    except FileNotFoundError :
        print("Analyse impossible : Ce chemin de fichier/dossier n'existe pas")
```

Photo de la première page : https ://www.pcworld.com/article/3434097/why-you-can-stop-paying-for-antivirus-software.html

Figure 1 : https ://en.wikipedia.org/wiki/WannaCry_ransomware_attack

Figure 2 : https ://en.wikipedia.org/wiki/ILOVEYOU