République Algérienne Démocratique et Populaire

Université Abou Bakr Belkaid–Tlemcen

Faculté des Sciences

Département d'Informatique

Rapport de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option Réseaux et Systèmes Distribués

Thème

Etude Et Implémentation des Mécanismes de détection de malware

Réalisé par :

- **ZERGOUG Khaled**
- **TABET ZATLA Youcef**

Présenté le 25 juin 2020 devant le jury composé de :

- Mr Ilyas BAMBRIK (Encadreur)
- Mme Asma AMRAOUI (Présidente)
- Mme Hanane ABDELDJELIL (Examinatrice)

Année universitaire : 2019-2020

Dédicaces

. 1 – ZERGOUG KHALED

. 2 – TABET ZATLA YOUCEF

Nous dédions ce travail à nos chers parents qui nous ont beaucoup aidés durant notre cursus, ainsi que nos frères et sœurs

À tous nos proches, grands et petits.

À tous ceux qui nous aimons.

À tous nos collègues de la promotion 2019/2020.

À tous nos enseignants pour le savoir et les connaissances qu'ils nous ont offerts.

Remerciements

Tout d'abord, nous remercions dieu qui nous a donné la force et la volonté pour accomplir ce travail.

Nous adressons nos vifs remerciements à notre encadreur

MR. BAMBRIK ILYAS

Pour son aide et ses conseils précieux

Sans son aide, notre travail n'aurait pas été accompli.

Nous remercions chaleureusement les membres du jury, Dr. Amraoui Asma et Dr Abdeldjelil Hanane, qui nous ont fait l'honneur d'accepter d'examiner notre travail.

Et aussi, nous remercions nos enseignants pour la qualité de l'enseignement au cours de ces 5 années passées à l'université Abou-Bekr Belkaid.

Enfin, nous remercions tous ceux qui ont participé de près ou loin pour terminer ce travail.

Merci à tous !

Résumé

Les logiciels malveillants sont des programmes malicieux créés pour endommager les appareils informatiques et l'impact de ces dernier augmente jour après jour. Notamment avec l'augmentation rapide de l'utilisation d'internet. Par conséquent, de nombreuses entreprises ont développé des détecteurs pour les divulguer. Cependant, les techniques utilisées par ces détecteurs ne sont pas efficace à 100%. Dans ce mémoire, on présente en détaille les types des logiciels malveillants ainsi que les techniques d'analyse et de détection des logiciels malveillants. En outre, ce mémoire présente l'implémentation de l'algorithme de Distance Mahalanobis pour la détection d'anomalies dans la structure des fichiers.

Mot clé : logiciel malveillant, technique d'analyse des malwares, distance mahalanobis.

ملخص

البرامج الضارة، وهي عبارة عن برامج خبيثة يتم انشاؤها وبرمجتها لإلحاق الضرر بمختلف اجهزة الاعلام الالي ويزداد تأثيرها يوما بعد يوم. وخصوصا بالزيادة السريعة في استخدام الانترنت. و بالتالي تم انشاء الكثير من الشركات المختصة في الكشف عنها، ومع ذلك التقنيات المعتمدة ليس لها فعالية 100%.في هذه الاطروحة، يتم تقديم مراجعة تفصيلية لأنواع البرامج الضارة، ودراسة تحليل البرامج الضارة وتقنيات الكشف عنها، ضف الى ذلك في هذه الاطروحة يتم تنفيذ نموذج مسافة ماهالانوبيس للكشف عن البرامج الضارة التي على شكل ملفات.

الكلمات المفتاحية: البرامج الضارة، تحليل البرامج الضارة، مسافة ماهالانوبيس.

Abstract

Malwares are malicious softwares that are created to damage computers devices and their impact is increasing day by day. With the rapid increase in Internet use, it has had great impacts that cannot be overlooked. Therefore, many companies have created detectors to disclose them, but the techniques used by the detectors are not 100% effective. In this report, we provide a detailed study of the types of malware, malware analysis and detection techniques. In addition, this study presents the implementation of the mahalanobis distance algorithm for detecting file structure anomalies.

Keywords : malware, malware analysis techniques, mahalanobis distance.

TABLE DES MATIÈRES

Glossaire

AV: **A**nti –**V**irus

PE: **E**xécutable **P**ortable

DM: **D**istance **M**ahalanobis

ED: **D**istance **E**uclidienne

API: **A**pplication **P**rogramming **I**nterface

SVM: **S**upport **V**ector **M**achines

DLL: **D**ynamic **L**ink **L**ibrary

MC: **M**alware **C**ode

CPU: **C**entral **P**rocessing **U**nit

DOS: **D**isk **O**perating **S**ystem

IA: **I**ntelligent **A**rtificiel

NB : **N**aïve **B**ayes

IMDS: **I**ntelligent **M**alware **D**etection **S**ystem

OOA: **O**bject **O**riented **A**ssociation

HAC: **H**ierarchical **A**ssociation **C**lassified

WEKA: **A**cronym **K**nowledge **E**nvironment **W**aikato

BFA: **F**requency **A**nalysis algorithm in **B**yte.

DT: **D**ecision **T**ree.

KNN: **K-N**earest **N**eighbor.

SAE: **S**tacked **A**uto **E**ncoders.

DL: **D**eep **L**earning.

CNN: **C**onvolutional **N**eural **N**etwork

Liste des figures

Liste des tableaux

Introduction générale

Le Malware, « logiciel malveillant », est une séquence d'instructions qui exécutent une activité malveillante sur un ordinateur. L'histoire des programmes malveillants a commencé avec « Computer Virus », un terme introduit pour la première fois par Cohen en 1983 qui désigne un morceau de code qui se réplique en s'attachant aux autres exécutables du système.

Aujourd'hui, divers types de programme malveillants existent en informatique comme les virus, les vers, les chevaux de Troie (Trojan Horse), kits racine (Root kit), les portes dérobées (Backdoor), les robots (Botnet), les logiciels espions (Spyware), les logiciels de scareware et tout autre programme présentant un comportement malveillant. Ces logiciels constituent une menace à croissance rapide pour le monde informatique moderne. La production de logiciels malveillants est devenue une industrie de plusieurs milliards de dollars qui trouve toujours de nouvelles façons pour se diffuser. La croissance du réseau Internet, les réseaux sociaux et la multiplication rapide des réseaux de zombies (Botnets) ont provoqué une augmentation exponentielle de la quantité de logiciels malveillants. En 2010, la propagation par le bais de courriers indésirables (spam) envoyés par des machines faisant partie des réseaux de zombie, été la méthode la plus fréquente pour propagation de malware.

McAfee Labs a rapporté qu'il y avait 6 millions nouvelles infections au cours de chaque mois. La même société a également signalé une détection moyenne de 60 000 nouveaux logiciels malveillants par jour [1]. En janvier 2011, les laboratoires de messagerie de Symantec ont rapporté que, chaque jour une moyenne de 2 751 sites web hébergeait des malwares [1].

La principale et la plus importante défense contre les logiciels malveillants sont les programmes antivirus, par exemple : Norton, McAfee, Sophos, Kaspersky et Clam antivirus. Les fournisseurs des programmes antivirus appliquent fréquemment de nouvelles technologies à leurs produits pour lutter contre les malwares.

Après la phase d'analyse et extraction de code malveillant, une base de données de signatures est utilisée comme outil principal afin de détecter des occurrences du même code malveillant. Bien que la détection basée sur les signatures soit très efficace contre les logiciels malveillants découverts précédemment, elle est inefficace contre les logiciels malveillants nouveaux et inconnus.

Subséquemment, les sociétés de sécurité informatique luttent afin de détecter les variantes de logiciels malveillants connus, ainsi que ceux inconnus pour développer des solutions robustes. Ces techniques d'analyse incluent des méthodes heuristiques, la vérification de l'intégrité et le sandboxing. La plupart des sociétés antivirus utilisent des méthodes manuelles pour le prétraitement de détection des signatures des logiciels malveillants. Toutefois, avec la quantité de nouveaux malwares qui augmente chaque jour, il serait difficile d'utiliser des méthodes manuelles [1]. Par conséquent, des mécanismes alternatifs pour détecter les malwares nouveaux et inconnus sont nécessaires.

Dans ce mémoire, nous proposant une implémentation de la distance de Mahalanobis permettant d'identifier des anomalies dans la structure de fichier. L'avantage de cette méthode de mesure de distance par rapport à une distribution repose sur la souplesse du modèle et sa rapidité de classification. Un exemple pertinent de l'applicabilité de notre application est l'analyse de pièces jointes reçues par mail.

Organisation du mémoire

Ce mémoire est constitué d'une introduction générale, trois chapitres et une conclusion.

- Dans le chapitre 1, on présente en premier temps les types de malwares ainsi les méthodes utilisées par ses créateurs afin de masquer les processus suspects. Ensuite dans la seconde partie de ce chapitre nous parlerons des techniques d'analyse et détection des malwares. Subséquemment, nous présenterons les différents travaux de recherche ainsi qu'une comparaison entre ceux avant et finir par une conclusion.

- Dans le chapitre 2, nous allons présenter un algorithme de détection ainsi que l'implémentation de celui-ci.

- Dans le chapitre 3, nous présenterons l'évaluation de performance de l'algorithme proposé.

Chapitre I

Catégories de malware et stratégies de détection

1.1 Introduction

Entre infiltration/contrôle des machines à distance et attaques fatales des services, les malwares connues jusqu'aujourd'hui sont répartie sur plusieurs catégories. De même beaucoup de code de malwares existants ont été réutilisés/ refaçonnés afin de créer d'autres.
Ce chapitre exploite les catégories de malwares ainsi que des techniques obscurcissements. Par la suite, on discute les travaux de recherche et une comparaison entre ces travaux est présentée.

1.2 Système informatique

Avant d'aborder les malwares, il est nécessaire de définir la cible de ces derniers : les systèmes informatiques. Un système informatique est un ensemble de composants logiciels et matériels permettant de traiter et d'enregistrer les données automatiquement [2]. La nature très complexe de ce système le rend vulnérable à plusieurs niveaux surtout au niveau logiciel. Fréquemment, les développeurs logiciel et matériel publient des mises à jour de sécurité pour leurs produits. De même, les pirates informatiques décompilent les produits logiciels afin de trouver des vulnérabilités exploitables.

1.3 Définition de malware

Les logiciels malveillants sont désignés par de nombreux noms, incluant les logiciels malveillants, les codes malveillants (MC) et les malcodes. De plus, nombreuses définitions ont été proposées pour décrire les logiciels malveillants. Christodorescu et Jha [3] décrivent une instance de malware comme un programme dont l'objectif est malveillant. Similairement, McGraw et Morrisett [4] définissent le code malveillant comme « tout code ajouté, modifié, ou supprimé d'un système informatique afin de causer intentionnellement des dégât ou disfonctionnement prévue du système ». Pour les objectifs de cette étude, nous adoptons la description donnée par Vasudevan et Yerraballi [5], qui décrit le malware comme « un terme générique qui englobe les virus, les chevaux de Troie, les logiciels espions et autres codes intrusifs ».

Cette désignation mérite une discussion un peu plus détaillée sur ces logiciels malveillants particuliers :

- Virus :

Un virus est un code qui se répète en s'insérant dans d'autres programmes. Celui-ci a besoin d'un programme hôte existant pour causer des dégâts. Par exemple, pour s'infiltrer dans un système informatique, un virus peut s'attacher à un utilitaire logiciel comme une application de traitement de texte. Le lancement de l'application de traitement de texte activerait le virus et celui-ci se dupliquerait et désactiver les détecteurs de logiciels malveillants installés sur le système informatique.

- Vers (Worms) :

Un vers informatique se réplique en exécutant son propre code indépendamment de tout autre programme. Par conséquence, la distinction entre cette catégorie de malware se distingue des virus par leur modèle de propagation. En général, les virus tentent de se propager à travers des programmes / fichiers et sur un seul système informatique. Contrairement, les vers se propagent via des connexions réseau dans le but d'infecter d'autres systèmes voisins.

- Chevaux de Troie :

Un cheval de Troie est un malware intégré par son concepteur dans une application ou un système [6]. L'application ou le système semble exécuter une fonction utile (par exemple, fournit la météo locale), mais effectue une action non autorisé (par exemple, capturer les frappes de l'utilisateur et envoyer ces informations à un hôte malveillant). Cette catégorie de malware est généralement associée à l'accès et à l'envoi d'informations non autorisées à partir des hôtes infectés. Ainsi, ils peuvent également être classés comme logiciels espions.

1.4 Les utilisateurs et les créateurs de malware

Les auteurs / utilisateurs de logiciels malveillants portent différents noms, certain des noms les plus populaires sont les chapeaux noirs (black hats), les hackers et les crackers. Les personnes ou organisations réelles qui prennent les noms susmentionnés peuvent être une menace externe / interne, un gouvernement étranger ou un espion industriel [7]. Lors de la

création de nouveaux logiciels malveillants, les chapeaux noirs emploient généralement des techniques comme l'obscurcissement et l'ajout / modification du comportement, afin de contourner les détecteurs de logiciels malveillants.

L'obscurcissement tente de cacher les véritables intentions des codes malveillants sans réprimer les comportements présentés par le malware. La modification crée effectivement de nouveaux logiciels malveillants, bien que l'essence du logiciel malveillant reste identique [8].

1.5 Les actions effectuées par les malwares

Les actions effectuées par les codes malveillants peuvent être répartie dans quatre groupes :

- Elévation de privilège : L'attaquant obtient les droits administrateur afin de manipuler les comptes utilisateurs.
- Contrôle à distance : Exécution de commande MS DoS/Shell à distance dans la machine de la victime.
- Charge financière : L'attaquant obtient les informations financières de la victime et effectue des transactions frauduleuses.
- Vol de données : L'attaquant accède et copie les données d'une entreprise pour les revendre aux parties intéressées.

1.6 Les familles de malware

Des millions de malware existent actuellement, ces malwares ont des fonctionnalités différentes, et peuvent être classés par familles selon leurs impacts et objectifs [9].

1.6.1 Backdoor

Backdoor est un programme malveillant installé par un attaquant sur un système cible pour y accéder à distance grâce à un port ouvert au niveau de la couche transport. Les cybercriminels exploitent diverses vulnérabilités sur la machine victime pour installer des portes dérobées. Parfois, l'utilisateur peut être amené à installer lui-même un backdoor, par

le bais d'un programme considéré légitime. Après l'installation, ces derniers modifient des fichiers de démarrage et/ou les clés de registre exécuté pour la création de taches planifiés. Certain backdoors permettent aux attaquants de modifier les privilèges en administrateur (root) et ainsi l'exécution à distance des commandes avec ce privilège. D'autre permettent aux attaquants de surveiller tous les activités sur un système cible.

1.6.2 Spyware

Un logiciel espion est un programme qui collecte des informations confidentielles sur l'ordinateur de l'utilisateur comme la capture de l'activité de navigation web. Par la suite les données récoltées sont envoyées à un des parties tiers pour des avantages monétaires [10]. Les navigateurs web sont les principales cibles des logiciels espions et les mécanismes plus fréquemment utilisés pour propager les logiciels espions incluent les contrôles ActiveX, les plug-ins et les programmes exécutables. En fait, l'utilisation des contrôles ActiveX est le moyen le plus simple et le plus efficace pour les attaquants afin de distribuer des logiciels espions.

1.6.3 Rootkit

Les Rootkits présentent un comportement de cheval de Troie et procèdent au remplacement de la version d'origine d'un fichier système par une copie infecté. Ensuite, une porte dérobée autorise aux attaquants d'accéder à distance au système. En fonction de l'environnement d'exploitation, les Rootkits sont réparties sur deux types :

- Rootkits en mode utilisateur
- Rootkits en mode noyau

Les Rootkits en mode utilisateur remplacent les applications au-dessus du noyau par du code malveillant pour atteindre leur objectif, cela aide les attaquants à cacher leur présence. Par contre, les Rootkits en mode noyau sont les mêmes que ceux des Rootkits en mode utilisateur, à l'exception de la modification de l'environnement d'exploitation. Dans ce cas, ils modifient complètement le noyau, ce qui nuit ordinateur cible.

1.7 L'évolution de camouflage dans les malwares

Pour développer les techniques d'analyse et de détection des malwares, il est nécessaire d'étudier le camouflage des malwares. Le camouflage des logiciels malveillants fait

référence à la dissimulation des logiciels malveillants pour les cacher le plus longtemps possible des détecteurs de logiciels malveillants. Il existe un certain nombre de techniques utilisées par les logiciels malveillants comme le chiffrement et la métamorphose. [11]

1.7.1 Chiffrement

Les développeurs des malwares veulent toujours que leur programme ne soit pas détecté par l'antivirus et la technique la plus simple qu'ils utilisent pour se camoufler est la cryptographie. Celle-ci constitue la première technique utilisée pour dissimuler les malwares [12]. Cette méthode nécessite des modules de chiffrement et de déchiffrement supplémentaires.

Afin de changer la signature du malware, il suffit que le chiffrement soit effectué avec une clé différente. En 1987, le premier malware crypté CASCADE est apparu [13]. La structure du virus crypté est illustrée dans Figure 1.

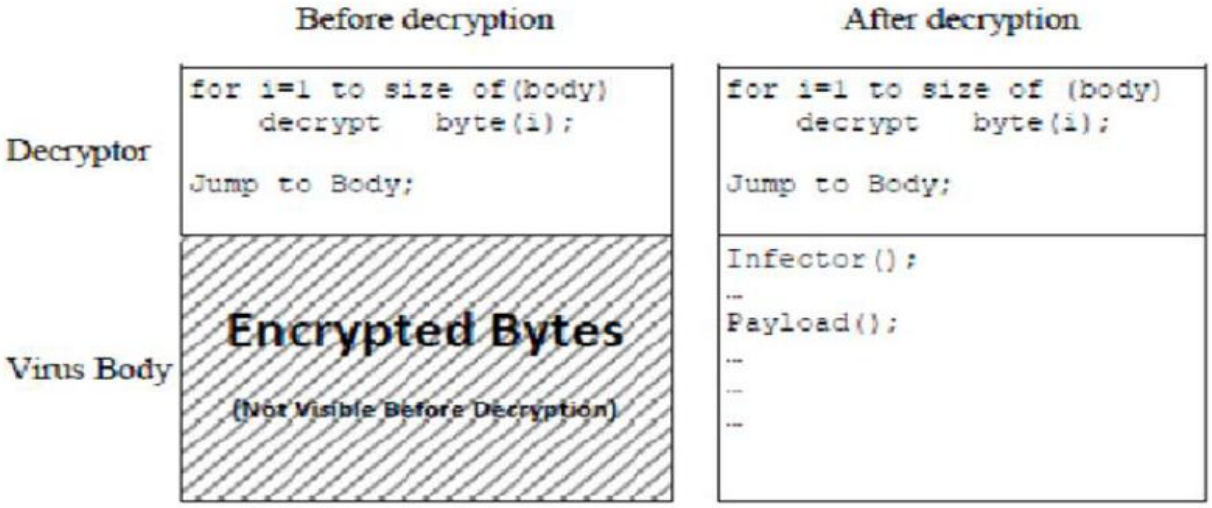

Figure 1. Structure du virus crypté [11]

L'objectif principal de cette technique est d'éviter la détection par l'analyse statique du code. Cette méthode retarde également le processus d'enquête.

1.7.2 Oligiomorphism

Le premier virus oligiomorphe, connu aussi sous le nom semi-polymorphe, est apparu en 1990, sous la forme d'un virus DOS [14]. Ceci été considéré comme une avancée dans le camouflage des malwares. Comme dans le camouflage par chiffrement, le décrypteur reste cohérent pour chaque infection. Bien que l'oligiomorphisme fournit un décrypteur différent d'une liste de décryptage pour chaque nouvelle attaque. Il y a toujours une chance d'être

détecté par un antivirus en vérifiant tout le décrypteur.

1.7.3 Polymorphism

En 1990, le premier virus polymorphe 1260 a été développé par Mark Washburn. Ces virus polymorphes sont une combinaison de cryptage et d'oligiomorphisme mais sont plus complexes que les autres virus. Il est donc très difficile de les détecter car ils changent d'apparence à chaque copie. Cette procédure de changement de code se fait par moteur de mutation.

1.7.4 Metamorphism

Au lieu d'utiliser la cryptographie, dans cette méthode de camouflage le contenu du logiciel malveillant est modifié. Ainsi, cette méthode n'inclue pas de module de chiffrement /déchiffrement. Par contre, un moteur de mutation est incorporé afin de changer tout le corps plutôt que de modifier uniquement le module de chiffrement /déchiffrement. L'idée de base est de changer la syntaxe sur chaque nouvelle copie sans affecter la sémantique afin de conserver le même fonctionnement. Le premier virus métamorphique ACG a été développé en 1998 pour DOS.

1.8 Les techniques d'obscurcissement des malwares

La technique que les programmes malveillants utilisent pour rendre celui-ci difficile à lire et à comprendre est connue sous le nom d'obscurcissement. De même, cette technique permet de masquer le comportement malveillant des malwares. Ces techniques d'obscurcissement sont classées de différentes manières par différents chercheurs [12], et ceux les plus utilisées sont réparties sur les catégories suivantes :

1.8.1 Insertion de code mort

C'est le moyen le plus simple de changer le code sans affecter sa signification. Dans cette technique des instructions garbage sont insérés dans le code en utilisant des opérations *nop* et *push* suivi de *pop*. Ces instructions sont utilisées dans des séquences qui n'affectent pas la sémantique du code.

1.8.2 Remplacement d'instructions

Dans cette technique, une instruction est remplacée par une autre instruction qui génère le même traitement, tout comme les synonymes dans le langage naturelle. Par conséquence, ceci rend la détection de ces malwares plus difficiles. Par exemple, toutes les instructions suivantes ont le même effet sur le registre *eax*. Chaque instruction met la valeur registre à zéro :

*mov eax,*0

xor eax,eax

*and eax,*0

sub eax,eax

1.8.3 Enregistrer la réaffectation

Cette technique réaffecte le registre dans chaque copie sans changer la sémantique du virus. C'est une technique plus simple mais lorsqu'elle est combinée avec une autre technique, elle peut être très difficile à détecter.

1.8.4 Réorganisation des sous-programmes

Un ensemble d'instruction en morceau de code est permuté dans cette technique de telle sorte que le code change dans son apparence mais le comportement reste le même. Un exemple de sous-programme suivi par sa réorganisation est décrit comme suit :

//Sous-programme

mov eax, 0A

push ecx

add esi, ebx

//réorganisation

add esi, ebx

mov eax, 0A

push ecx

1.8.5 Transposition de code

Dans cette technique, le flux des instructions dans le morceau de code d'origine est réorganisé de telle sorte que la sémantique ne change pas [16]. Il existe deux approches qui peuvent être utilisées pour effectuer la transposition de code. L'une consiste à réorganiser les instructions au hasard et pour récupérer le code d'origine. Les instructions et les sauts inconditionnels sont utilisés tandis que dans l'autre méthode, les instructions qui sont indépendantes et n'ont aucun impact sur les autres instructions sont permutées.

1.8.6 Intégration de code

Comme son nom indique, dans cette technique d'obscurcissement de malware, le code malveillant s'intègre dans le programme qui doit être exécuté. Dans ce contexte, le programme d'origine est décomposé et du code malveillant y est ajouté de sorte qu'il ne peut pas être facilement détecté [17].

1.9 Les techniques d'analyse des malwares

L'analyse des logiciels malveillants consiste à l'extraction des comportements ou structure de ces derniers. Ces techniques sont divisées en trois catégories en fonction de la méthode appliquée :

1.9.1 Analyse statique

Lorsqu'un logiciel ou un morceau de code est analysé sans qu'il soit exécution, ce type d'analyse est appelé analyse statique ou analyse de code. Les informations statiques sont extraites du code pour déterminer si le logiciel contient du code malveillant ou non. Dans cette technique, le logiciel est rétro-conçu en utilisant différents outils et/ ou la structure du code malveillant est analysée afin de comprendre comment celui-ci fonctionne. Différents outils peuvent être utilisés pour effectuer une analyse statique comme le débogueur, le désassembleur, le décompilateur et les analyseurs de code source. Les méthodes utilisées pour effectuer une analyse statique comprennent l'inspection du format de fichier, l'extraction de chaines (signatures) et l'empreinte digitale.

1.9.2 Analyse dynamique

Lorsque la fonctionnalité d'un logiciel est analysée et observée en l'exécutant, cette

analyse est connue sous le nom d'analyse dynamique [18]. Cela peut être fait en traçant les appels de fonctions, les flux de contrôle et également en analysant les instructions et les paramètres des fonctions.
Pour ce type d'analyse, les codes malveillants sont exécutés dans un environnement virtuel pour observer leurs comportements et concevoir les actions contre ces comportements suspects. Les outils utilisés pour l'analyse dynamique sont les sandboxs, simulateurs, émulateurs RegShot ainsi que Process Explorer.

L'efficacité de l'analyse dynamique par rapport à l'analyse statique réside dans le fait que le logiciel infecté est exécuté sur une machine virtuelle afin de le surveiller. Ainsi, les techniques d'obscurcissement ne sont pas efficaces contre cette technique d'analyse contrairement à l'analyse statique. Par contre, ce type d'analyse prend plus de temps car ceci nécessite l'exécution du logiciel malveillant dans un environnement virtuel pour une durée suffisante avant de le classifier. De plus, cette stratégie d'analyse renvoie un pourcentage de faux positive élevé.

1.9.3 Hybride

La méthode hybride combine à la fois des techniques d'analyse statiques et dynamiques et peut donc profiter des avantages des deux approches. Tout d'abord, un logiciel est prétraité par analyse de code en vérifiant la signature du malware. Puis, celui-ci est exécuté dans un environnement virtuel pour observer son comportement réel. Une comparaison entre les techniques d'analyse est présentée dans le Tableau 1.

Analyse statique	**Analyse dynamique**	**Analyse hybride**
Rapide et sur.	Long et vulnérable	Détection directe avec l'analyse statique et détection long avec l'analyse dynamique.
Analyse le code et permet la détection d'anomalies sans l'exécution.	Analyser et observer le code pendant l'exécution virtuelle.	Observe l'exécution dans un environnement virtuelle pour détecter les anomalies.
Pourcentage faible de faux positifs	Pourcentage élevé de faux positifs	Pourcentage erreur faible pour l'analyse statique et élevé pour l'analyse

		dynamique.
Inefficace contre les techniques d'obscurcissement.	Efficace contre les techniques d'obscurcissement.	Efficace contre l'obscurcissement.

Tableau 1. Comparaison de l'analyse statique et dynamique

1.10 Les techniques de détection des malwares

Les techniques de détection des logiciels malveillants peuvent être divisées en trois catégories principales : a) Détection par signature, b) Détection par heuristique, c) Détection par spécification. Ces techniques identifient et détectent les malwares et par la suite prennent des contre-mesures pour sécuriser le système hôte.

1.10.1 Détection par signature

Lorsqu'un malware est écrit, une séquence d'octets généralement connue sous le nom de signature, est intégrée dans son code qui peut ensuite être utilisée afin de détecter celui-ci.
Cette technique de détection basée sur les signatures est adoptée par la plus part des programmes antivirus. Le programme antivirus désassemble l'exécutable du fichier infecté et recherche chaque signature déjà connue afin de déterminer s'il existe un code malveillant [19].
Par la suite, chaque fois que de nouvelles signatures malwares sont extraites, celle-ci sont ajoutées à la base de données, puis utilisées à des fins de comparaison ultérieurs. Ce type de technique de détection est également connu sous le nom balayage, correspondance de chaines ou de motifs. Cette méthode de détection, peut être appliquée dans l'analyse statique (recherche dans la structure du fichier) ou dynamique (recherche dans la séquence d'appels).

1.10.2 Détection par heuristique

Les méthodes basées sur l'heuristique détectent le comportement normal et anormal d'un système[20]. Ainsi, les attaques de logiciels malveillants connus et inconnus peuvent être identifiées et résolues. Le processus de détection basé sur l'heuristique comprend deux étapes. Dans une première phase, le comportement du système est observé en l'absence d'attaque et un enregistrement est conservé des informations importantes qui peuvent être

vérifiées en cas d'attaque. La différence dans le comportement est surveillée pendant la deuxième étape pour détecter les logiciels malveillants d'une famille particulière. Le détecteur de comportement utilisé dans une technique basée sur l'heuristique comprend les trois composants de bas suivants :

* **Data collection** : comme son nom l'indique, ce composant collecte des données statiques ou dynamiques.
* **Interprétation** : ce composant interprète les données collectées et les transforme en une forme intermédiaire.
* **Matching algorithm** : Ce composant est utilisé pour créer la signature de comportement avec les informations converties par le composant d'interprétation.

Bien que la détection basée sur l'heuristique soit une méthode efficace, ils existent des limites associées celle-ci car elle nécessite plus de ressources. Au fait, le niveau de faux positifs est également élevé. Cette méthode est aussi connue sous le nom de technique proactive ainsi que technique de détection de comportement ou d'anomalie.

1.10.3 Détection par spécification

Dans la technique de détection basée sur les spécifications, les applications sont surveillées en fonction de leurs spécifications et ainsi la vérification est effectuée pour classifier le comportement normal et anormal. Cette technique est dérivée d'une technique basée sur l'heuristique mais la principale différence réside dans l'utilisation de l'apprentissage automatique ainsi que les méthodes d'IA pour détecter l'activité valide et invalide d'un programme. Par contre, la technique de détection basée sur les spécifications s'appuie sur l'analyse de comportement qui est décrite dans la spécification du système [21].
Cette méthode est en quelque sorte une comparaison des activités normales d'un système avec ses activités actuelles. Par conséquence, celle-ci surmonte la limitation des techniques basées sur l'heuristique en abaissant le niveau de faux positifs.
Les avantages et les inconvénients des trois techniques de détection sont présentés dans le Tableau 2. Dans Figure 2, la classification et les liens entre les techniques de détection des logiciels malveillants sont présentés.
Chacune des techniques de détection des logiciels malveillants peut être statique, dynamique ou hybride. De plus, la technique de détection par spécification de comportement est dérivée d'une technique de détection basée sur l'heuristique.

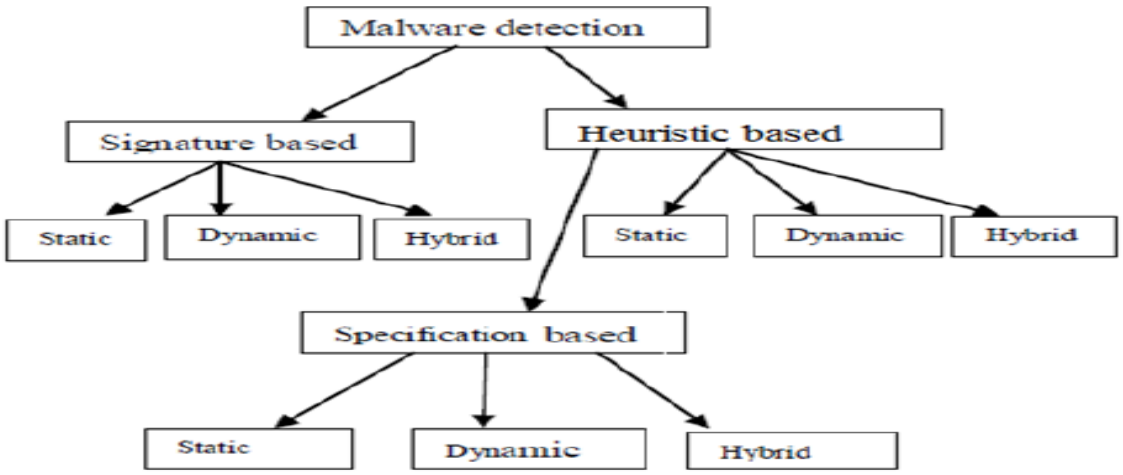

Figure 2. Catégories de techniques de détection de malware [22]

	Les avantages	**Les inconvénients**
Détection par signature	-les malwares connus peuvent être détectés facilement. -Utilise moins des ressources par rapport à d'autres techniques.	-Les malwares inconnus ne peuvent pas être détectés.
Détection par heuristique	-Les malwares connus ainsi que ceux inconnus peuvent être détectés.	-Nécessite plus de ressources par rapport à d'autres techniques. -le pourcentage de faux positifs est élevé.
Détection par spécification	-Les malwares connus et inconnus peuvent être détectés -le niveau de faux positifs diminue par rapport à détection basé sur l'heuristique.	-le développement des spécifications prend beaucoup de temps.

Tableau 2. Avantages et inconvénients des techniques de détection de malwares.

1.11 Les Travaux de recherche sur la détection de malware

Dans cette partie, plusieurs travaux discutons principalement de la détection des malwares

sont comparés.

- **Schultz et al.** a appliqué des méthodes d'exploration de données pour différents types de détection de logiciels malveillants basé sur les caractéristiques suivants: a) les informations d'appel de la bibliothèque de liens dynamiques (DLL), b) les chaines et les séquences d'octets [23].

  Un algorithme d'induction de règles Ripper [24] a été appliqué pour découvrir les modèles basés sur l'ensemble de données d'appel DLL. Par la suite, Naïve Bayes (NB) a été utilisé pour construire les classificateurs reposant sur les chaines et les séquences de 2 octets. Les concepteurs de cette méthode se sont appuyés sur leur collecte de données à partir de 4266 fichiers (3265 malveillants et 1001 bénins). L'algorithme NB a atteint des performances de classification élevées avec une précision de 97.11%.
- **Ye et al 2007.** ont proposés un Système de détection Intelligent Malware (IMDS) pour la détection de logiciels malveillants à l'aide de l'exploration OOA (Object Oriented Association) qui utilise les appels d'API Windows extraites [25]. Afin d'améliorer encore l'efficacité de la détection des logiciels malveillants, les auteurs ont proposé une méthode de post-traitement, nommée CIDCFP [26], pour la construction du modèle de classification. Pour résoudre le problème de répartition des classes de déséquilibre, ils ont également développé le classificateur associatif hiérarchique (HAC) [27] pour la détection des logiciels malveillants.

- **Tian et al.**[28] a extrait des séquences d'appels d'API à partir d'exécutables. Plusieurs classificateurs disponibles dans WEKA [29] ont été utilisés pour la détection de logiciels malveillants, notamment SVM, Random Forest, DT et Instance-based Classifier. Une base d'entrainement composée de 1368 logiciels malveillants et 456 fichiers bénins a été utilisée dans les expériences. Leur travail qui utilise des techniques d'analyse dynamiques pour l'extraction d'entités pour améliorer l'efficacité de la détection des logiciels malveillants a conclu avec une précision de 97%.

- **Wang et al.** [30] a développé un système de détection de spywares qui utilise des fonctionnalités extraites statiques et dynamiques. Les fonctionnalités statiques étaient les DLL et les API extraites de l'ensemble d'échantillons de fichiers, tandis que les fonctionnalités dynamiques étaient les modifications apportées aux fichiers système, aux

registres et aux activités réseau.

Fondé sur la collecte de données comprenant 407 logiciels espions et 740 programmes bénins, en utilisant la méthode du gain d'informations. Ils ont sélectionné 81 caractéristiques parmi les 790, dont 67 statiques et 14 dynamiques. Ensuite, les auteurs de cette approche ont construit un classificateur SVM pour la détection. La précision globale de leur système a atteint 96,43%. Ils ont également affirmé que les performances du système étaient meilleures que certaines applications antivirus bien connues dans la détection des logiciels espions.

- **Santos et al.** [31] ont développé un détecteur de malware hybride en utilisant à la fois des fonctionnalités statiques et dynamiques. L'analyse statique extrait les caractéristiques en modélisant les exécutables sous forme de séquences de codes opérationnels. L'analyse dynamique extrait les fonctionnalités en surveillant les opérations, les appels système et les exceptions levées. Les résultats expérimentaux sur la collection d'échantillons de 1 000 logiciels malveillants et 1 000 fichiers bénins ont démontré que cette approche hybride améliorait les performances de classification.

- **Ye et al.2011** [32], ont étudié comment les relations de fichiers pouvaient aider à améliorer la détection des logiciels malveillants. Pour cet objectif, ils ont développé « Valkyrie », un système de verdict de fichiers basé sur un modèle de classification semi-paramétrique en intégrant le contenu et les relations entre les fichiers pour la détection des logiciels malveillants. Leurs expérimentations ont été menées sur un grand ensemble de données de la société Comodo. La base d'expérimentation comprenait 30950 échantillons de logiciels malveillants (soit 225830 fichiers bénins et 434870 fichiers inconnus).

1.12 Comparaison des travaux de recherche

Travaux	**Méthodes de classification**	**Description**
Schultz et al	Ripper, NB	Basé sur l'ensemble de caractéristiques de chaine, l'algorithme NB produit une précision de 97,11%.
Ye et al 2007	NB, DTs, SVM,	Basé sur les appels d'API Windows, le

	Classification associative (CIDCFP, HAC)	classificateur associatif a surpassé les autres classificateurs.
Tian et al.	SVM, DT, Random Forest, Classificateur base sur une instance.	Ils ont utilisé les séquences d'appels API extraits dynamiquement de 1 368 logiciels malveillants et 456 fichiers bénins. Cette proposition a atteint une précision de plus de 97%.
Wang et al	SVM	Utilisation d'extraits statiques et dynamiques comme caractéristiques. Atteint une précision globale de 96,43%.
Santos et al	DT, KNN, NB, et SVM	Démontre que l'approche hybride améliore les performances. Basé sur l'analyse statique et dynamique.
Ye et al 2011	Classificateur avec modèle semi-paramétrique.	Les résultats expérimentaux ont démontré que la précision et l'efficacité de leur système a surpassé les outils logiciels anti-spyware populaires.

Tableau 3. Comparaison des méthodes utilisées dans la détection des logiciels malveillants

1.13 Conclusion

Nous avons présenté dans ce chapitre les malwares, leurs familles, ainsi que les actions effectuées par les malwares. Ensuite, nous avons montré l'évolution de camouflage dans le malware et les techniques d'obscurcissement. Nous avons présenté l'analyse dynamique, statique et hybride ainsi que, les techniques de détection qui sont basés sur : la signature, l'heuristique et la spécification. Enfin, nous avons résumé les travaux de recherche sur la détection de malware et nous avons fait une comparaison entre ceux. Dans le deuxième chapitre nous parlerons en détail de l'algorithme Distance de Mahalanobis que nous souhaitons implémenter.

Chapitre II

Algorithme de Mahalanobis

2.1 Introduction

Avant de commencer l'implémentation, nous nous intéressons à l'algorithme de détection basé sur la Distance Mahalanobis. On discute l'origine de la Distance Mahalanobis et la différence entre celle-ci et distance euclidienne dans ce chapitre. On termine cette partie par les travaux qui sont basés sur la Distance Mahalanobis.

2.2 Définition de Distance de Mahalanobis

La Distance de Mahalanobis (DM) est une distance généralisée qui peut être considérée comme une mesure unique du degré de divergence dans les valeurs moyennes des différentes caractéristiques d'une population en considérant les corrélations entre les variables. Cette distance est un moyen très utile pour déterminer la similitude d'un ensemble de valeurs d'un échantillon inconnu avec un ensemble de valeurs mesurées à partir d'une collection d'échantillons connus [33]. Cette méthode a été appliquée avec succès pour la discrimination spectrale dans un certain nombre de cas.
L'une des principales raisons d'utiliser DM est qu'elle est sensible aux changements inter-variables dans les données de référence. La supériorité de cette méthode par rapport aux autres distances multidimensionnelles, comme la Distance Euclidienne, est dû à la prise en compte de la distribution des points (corrélations). Traditionnellement, cette distance est utilisée pour classer les observations en différents groupes.

2.3 L'origine de Distance de Mahalanobis

La Distance de Mahalanobis a été proposée pour la première fois par le statisticien indien Mahalanobis en 1936 [34]. Les années 1930 été une période importante pour le développement de concepts multi-variés. Principalement en biologie, économie et psychologie, avec de nombreux contributeurs célèbres dans ce domaine tel que R.A. Fisher et H. Hotelling. Pour les statistiques uni-variées traditionnelles, il est habituel de calculer l'écarts-types d'une observation par rapport au centre d'un ensemble de données et d'utiliser cette valeur pour déterminer diverses statistiques à ce sujet. L'extension de ce concept à l'espace multidimensionnel produit la Distance de Mahalanobis.

2.4 La différence entre Distance Mahalanobis et Euclidienne

Par rapport aux méthodes classiques, la Distance Mahalanobis est utilisée pour déterminer si un nouveau point inconnu est loin/proche par rapport au centre d'un groupe de points connus. Par ailleurs, cette mesure peut être répétée avec la même observation et différent groupe. A la fin, cette nouvelle observation sera classée dans le groupe avec le centre le plus proche. Il existe d'autres techniques de mesure multi-variées, comme la Distance Euclidienne (ED). La Distance Euclidienne donne également la distance entre le point « inconnu » et le centre du groupe. Par contre cette technique présente un majeur désavantage car elle ne prend pas en compte la distribution des points dans le groupe et la covariance/variance des composantes.

2.5 Les travaux utilisant la Distance de Mahalanobis

Dans cette partie nous présentons plusieurs travaux qui utilisent la distance Mahalanobis pour la détection détection de malware ainsi que dans d'autres domaines scientifiques. Malgré la simplicité de celle-ci, les résultats produits dans le contexte de mesure de similarité par rapport à une distribution sont satisfaisants.

McDaniel et Heydari [35] Introduisent les algorithmes pour identifier les types de fichiers en analysant le contenu. Dans l'algorithme d'analyse de fréquence d'octets (BFA), les auteurs calculent la distribution de fréquences d'octet de différents fichiers et génèrent des « empreint digitales » de chaque type de fichier en faisant la moyenne de la distribution de fréquence d'octet de leur fichiers respectifs. Ensuite, les auteurs prennent en compte la différence des fréquences du même octet dans différents fichiers. Si la différence diminue, la force de corrélation augmente vers 1 et vice versa. Dans l'algorithme de corrélation croisée octet-fréquence, la corrélation entre toutes les paires d'octets est calculée. Subséquemment, la fréquence moyenne de toutes les paires d'octets et la force de corrélation similaire à l'algorithme BFA sont calculées. Même si ce travail n'utilise pas la Distance Mahalanobis, il est à l'origine de la plus part des travaux utilisant celle-ci dans l'analyse de fichiers.

Wei-hen Li et al [36] identifient les types de fichiers en utilisant l'analyse n-grams. Les auteurs proposent le calcule distribution de fréquence de 1-grams des fichiers et construisent 3 modèles différents de chaque type de fichier : a) un centroide unique (un modèle de chaque type de fichier), b) un multi-centroide (plusieurs modèles de chaque type de fichier), c) des fichiers exemplaires (ensemble de fichiers de chaque type de fichier) comme centroide. Cette collection de données est appelée « file print ». Dans les modèles mono et multi-centroides, les

auteurs calculent la moyenne et l'écart-type de la distribution de fréquence de 1-grams des fichiers, et utilisent la Distance de Mahalanobis pour comparer ces modèles avec la distribution de 1-grams du fichier pour trouver le modèle le plus proche.
Dans le modèle de fichier exemplaire, cette méthode compare la distribution de 1-grams du fichier exemplaire avec celle du fichier en question sans calculé la variance, et la Distance Manhattan est utilisée à la place de la Distance Mahalanobis. Leur solution ne permet pas d'identifier les fichiers ayant des distributions de fréquences d'octets similaires tels que les formats de fichiers MS Office (tels que Word et Excel). Par contre, ces derniers sont traités comme un groupe ou un type de fichier abstrait [37].

En sciences de l'environnement et de la santé, **Liu et Weng (2012) [38]** ont utilisé la Distance de Mahalanobis dans des études de santé publiques pour améliorer la résolution de l'imagerie satellite. Ces derniers ont effectué une analyse spatio-temporelle de l'épidémie du virus du Nil occidental à Los Angeles en 2007 en utilisant des variables de détection et des enregistrements de surveillance des moustiques infectieux.
La Distance de Mahalanobis a été utilisée pour identifier et cartographier les zones à risque où l'habitant convenait aux moustiques infectieux. Les auteurs ont calculé la distance entre un vecteur de variables environnementales et le vecteur moyen de facteurs environnementaux aux endroits les plus proches des infections à moustiques. Les emplacements avec des valeurs plus faibles des Distances de Mahalanobis ont indiqué un habitat plus favorable pour les moustiques et donc une zone à risque plus élevée.

En chimie analytique, **Shah et Gemperline (1990) [39]** se sont intéressés à l'analyse des spectres de réflectance dans le proche infrarouge des matières premières. Ils ont utilisé la Distance de Mahalanobis comme technique de classification pour la reconnaissance de formes afin de classer de nouveaux échantillons en les comparants à des mesures de classes prédéterminées. Chaque échantillon a été classé selon une valeur associée à sa Distance Mahalanobis par rapport aux centroides des classes.

Dans l'analyse d'images de télédétection, **Foody (2006) [40]** s'est intéressé à mesurer la proximité d'un pixel d'image avec un centroide de classe unique. Pour cela, il a utilisé la Distance de Mahalanobis après avoir converti la Distance de Mahalanobis calculée, d'un pixel d'image particulier à partir d'un centroide de classe spécifié, en une valeur associée à partir de la distribution de la Distance Mahalanobis.

2.6 La formule de Distance Mahalanobis

Pour calculer la distance entre les distributions d'octets des pièces jointes inconnu (Pdf, Word,…) avec des modèles préconstruit, la formule de la (D M) utilisée est la suivante :
$D^2 (x, y) = (x-y)^T C^{-1} (x-y)$.

Ou x et y sont les deux vecteurs de caractéristiques et chaque élément du vecteur représente la fréquence d'un n-grams. x est le vecteur caractéristique de nouvelle observation, et y est le vecteur d'entité moyen, et C^{-1} est la matrice de covariance inverse. L'avantage de Mahalanobis Distance est qu'elle prend en compte la variance et la covariance des variables mesurées ainsi que la valeur moyenne. Par la suite, au lieu de calculer simplement la distance à partir des valeurs moyennes, chaque variable est pondérée par son écart-type et sa covariance.

2.7 Implémentation l'algorithme de Distance Mahalanobis

Dans cette dernière partie de ce chapitre, nous traitons la phase d'implémentation de l'algorithme Distance de Mahalanobis pour la détection Malware. Cette phase consiste à transformer la formule de Distance Mahalanobis établi précédemment en des composants logiciels. Nous allons commencer par l'environnement de travail avant d'élaborer les composantes de notre approche.

2.7.1 Environnement de travail

2.7.1.1 Environnement matériel

Le matériel utilisé dans le développement est caractérisé par :
1. Système exploitation : Windows 10 Intégrale 64-bit.
2. CPU : Intel (R) Core (TM) i5-3230M CPU @2.60GHz 2.60GHz.
3. Mémoire : 8 GB RAM.

2.7.1.2 Environnement logiciel

Le langage utilisé pour le développement est le langage Java et ce choix est justifié par ses nombreux avantages. Pour complémenter le choix de langage, nous avons utilisé Netbeans IDE 8.2 pour la création de notre application. Parmi les avantages présentés par Java :
1. La portabilité (exécutable sur n'importe quel système, à condition avoir installé un

JVM).

2. La robustesse (la bonne gestion de mémoire, garbage collection efficace et une bonne gestion d'exception).

3. Richesse en bibliothèques.

2.7.2 Implémentation

Nous allons présenter les étapes pour la réalisation avec l'illustration des captures écrans de notre application. La première interface, est une interface de démarrage qui permet de lister les fonctionnalités et les informations sur notre application (Figure 3).

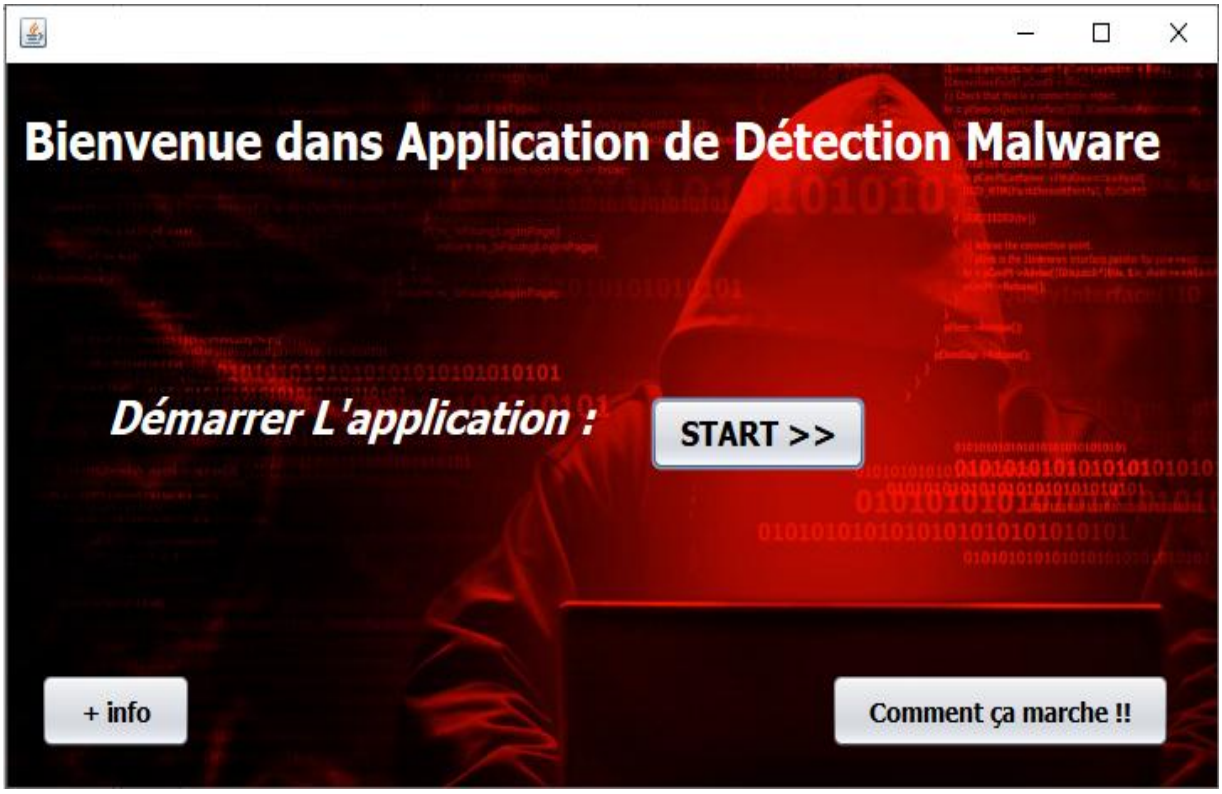

Figure 3. Interface de démarrage d'application de détection Malware

Après le clique sur le bouton **START**, la deuxième interface est lancée. Dans cette dernière on doit choisir l'extension du fichier. Pour le moment, l'analyse pour deux types d'extensions (.docx, .pdf) est disponible. Premièrement, on doit sélectionner le chemin du document que nous voulons tester.

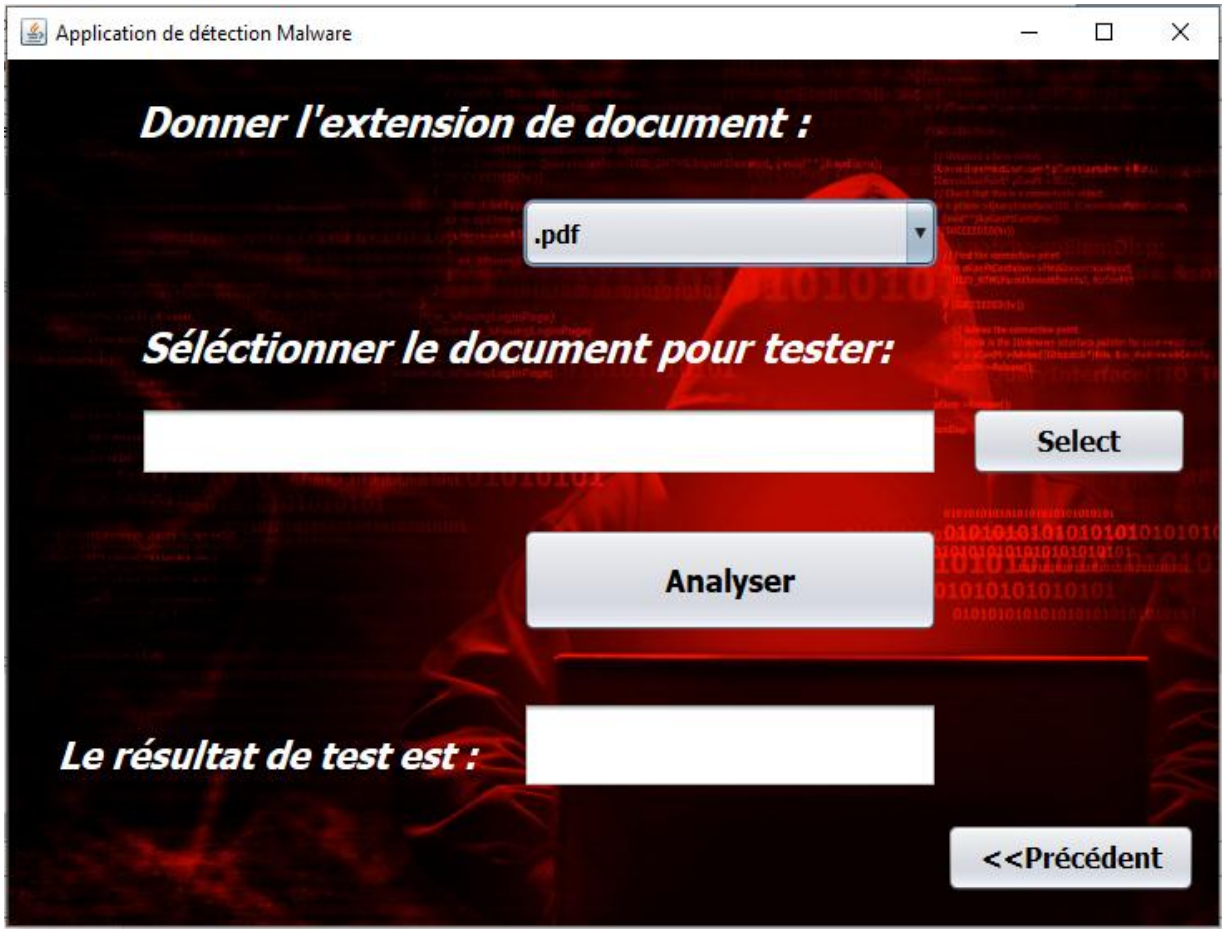

Figure 4. Interface d'application de détection Malware

Ensuite le clique sur le bouton ***Analyser***, permet de calculer la Distance de Mahalanobis entre le document que nous voulons tester et la base de documents. Cette base de documents est utilisée une seule fois pour construire le vecteur moyen et la matrice covariance. Par la suite le résultat est sauvegardé dans un fichier texte. Chaque fois qu'on souhaite analyser un document, on récupère la matrice covariance et le vecteur moyen à partir de fichier texte sauvegardé.

Le calcul de la distance comporte les étapes suivantes :

La lecture de document :

D'abord, on doit lire le document que nous voulons tester, depuis le chemin. Lorsque la phase de lecture est terminée, on obtient une chaine de caractère.

Extraire les 1-grams :

L'étape suivante, consiste à diviser la chaine obtenue après la lecture en 1-grams. Le résultat final est l'obtention une liste de 1-grams.

Construction de la liste de fréquences :

Ensuite, nous prenons la liste de 1-grams correspondante au document que nous voulons tester, et on comptabilise la liste de fréquences.

Construction de la matrice générale et le vecteur moyen :

Subséquemment, une matrice générale est construite qui contient dans les lignes la liste des fréquences et dans les colonnes l'ensemble des listes de 1-grams correspondant à la base de documents. Ainsi, on calcule la somme des fréquences des 1-grams pour chaque document afin de normaliser la matrice générale. Ensuite, on calcule la moyen de chaque 1-grams de liste de fréquences par rapport à l'ensemble des documents.

L'exemple suivant, représente une illustration sur comment la matrice générale et le vecteur moyen sont construits.

Exemple :

Nous supposons que le document que nous voulons tester après la lecture est le suivant :

DocT = « abbcd ».

Et la base de documents après la lecture est la suivantes :

Doc1 = « abcdd », Doc2= « abcdabcd», Doc3= « abcdabcdabdd».

Dans la deuxième étape, l'extraction des 1-grams donne les listes suivantes :

DocT= « a,b,b,c,d ».

Doc1= « a,b,c,d,d », Doc2= « a,b,c,d,a,b,c,d », Doc3=« a,b,c,d,a,b,c,d,a,b,d,d ».

La liste de fréquence de DocT est construite par la figure suivante :

$$\text{ListeFrq} = \begin{matrix} a \\ b \\ c \\ d \end{matrix} \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

Figure 5. La liste de fréquences

Comme nous avons mentionné précédemment, la matrice générale contient la liste de fréquences de chaque 1 gram pour les documents de la base d'entrainement (Doc1 & Doc2 & Doc3), et on calcule la somme des 1-grams dans chaque document pour normaliser la matrice générale :

	Doc1	Doc2	Doc3
a	1	2	3
b	1	2	3
c	1	2	2
d	2	2	4
Somme1	5	8	12

Figure 6. La matrice générale

Après la construction de la matrice générale, le vecteur Somme1 est obtenu. Pour la normalisation de la matrice générale nous divisons chaque valeur de la matrice par la valeur correspondant dans le vecteur Somme1 :

	Doc1	Doc2	Doc3	Somme	Moyen
a	1/5	1/4	1/4	7/10 = 0.7	7/30 = 0.23
b	1/5	1/4	1/4	7/10 = 0.7	7/30 = 0.23
c	1/5	1/4	1/6	37/60 = 0.61	37/180 = 0,2
d	2/5	1/4	1/3	59/60 = 0.98	59/180 = 0.327

Figure 7. La matrice générale normalisée et le vecteur Moyen

Ainsi, le vecteur Moyen est exprimé par la somme des fréquences de chaque 1-grams dans chaque document divisée sur le nombre de document. Finalement, on a construit la matrice générale normalisé et le vecteur Moyen.

Construction de la matrice de covariance :

La matrice de covariance est une matrice carrée, qui contient les variances et les covariances associées à plusieurs variables. Les éléments dans la diagonale de la matrice contiennent les variances des variables, tandis que les éléments hors diagonale contiennent les covariances entre toutes les paires de variables. La figure suivant illustre la matrice de covariance.
(Var : Variance, Cov : Covariance).

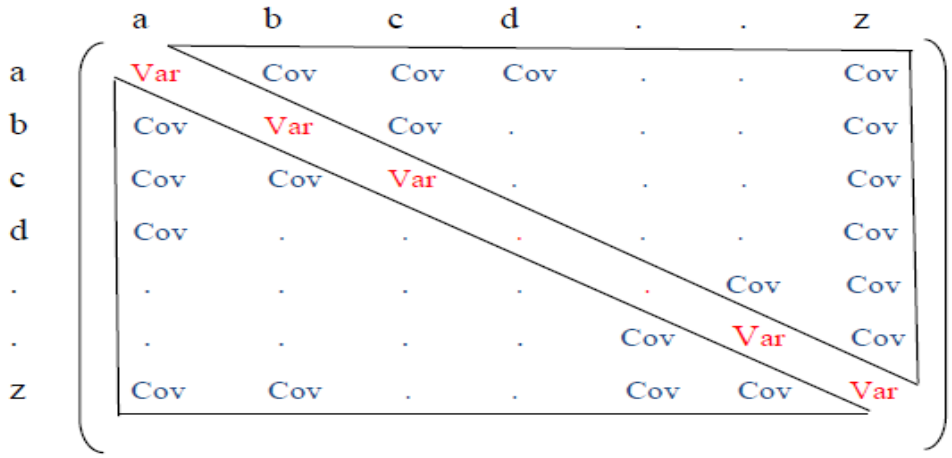

Figure 8. Schéma de la matrice de covariance

La formule pour calculer la variance et la covariance est suivante :

$$Var(x) = \frac{1}{N-1}\sum [x(i) - Moyen(x)]^2.$$

$$Cov(x, y) = \frac{1}{N-1} \sum [x(i) - Moy(x)]*[y(i)- Moy(y)].$$

Moy(x) et Moy(y) : sont les vecteurs des moyennes des variables x et y, construits à partir de la matrice générale précédente.

N : le nombre des documents dans la matrice générale.

La matrice de covariance est symétrique ce qui signifie que :
Cov(X, Y) = Cov(Y, X). Et Cov(X, X) = Var(X).

Exemple :
En suivant l'exemple précédent, la matrice de covariance suivante est produite :

<u>La variance :</u>

$Var(a) = \frac{1}{2}[(1/5 -7/30)^2 + (1/4 -7/30)^2 + (1/4 -7/30)^2]$.

$Var(c) = \frac{1}{2}[(1/5 -37/180)^2 + (1/4 -37/180)^2 + (1/6 -37/180)^2]$.

La covariance :

Cov(a,b)= $\frac{1}{2}$ [(1/5 -7/30) *(1/5 -7/30) +(1/4 -7/30)*(1/4 -7/30) +(1/4 – 7/30)*(1/4- 7/30)] .

Cov(c,d)= $\frac{1}{2}$ [(1/5 -37/180) *(2/5 -59/180) + (1/4 -37/180)*(1/4 -59/180) + (1/6 – 37/180)*(1/3-59/180)] .

Et après la propriété : Cov(a, b) = Cov(b, a) et Cov(c,d) = Cov(d,c).

La figure suivante illustre le résultat obtenu dans la matrice de covariance :

	a	b	c	d
a	$8.33*10^{-4}$	$8.33*10^{-4}$	$1.38*10^{-4}$	-0.0018
b	$8.33*10^{-4}$	$8.33*10^{-4}$	$1.38*10^{-4}$	-0.0018
c	$1.38*10^{-4}$	$1.38*10^{-4}$	0.00175	-0.0020
d	-0.00180	-0.00180	-0.0020	0.00564

Figure 9. La matrice de covariance

La matrice de covariance inversée :

Avant de calculer la Distance Mahalanobis, on doit calculer l'inverse de la matrice de covariance. Pour cela on a téléchargé et importé la bibliothèque d'algèbre linéaire [41]. Cette bibliothèque propose toutes les opérations d'algèbre linéaire (inversion d'une matrice, multiplication entre deux matrices, multiplication entre matrice et vecteur,….). Parmi ses avantages, les résultats sont précis et rapide. Avec l'aide de cette bibliothèque on a obtenu la matrice inversée de covariance.

Le seuil de la Distance Mahalanobis :

Le seuil de la Distance Mahalanobis est un intervalle de la distribution normale, dans laquelle 95% des individus sont inclus [-146,6239 ; 146,6239]. On a obtenu cet intervalle à partir du tableau *chi square* avec une probabilité de faux positif α inférieur ou égale à 5%. Si la distance que nous avons déterminé est inclut dans cette intervalle, cela signifie que le document est correct (n'est pas malveillant). Sinon le document est un malware. Les figures suivantes montrent une expérience sur notre application, pour tester si le document est un malware ou non :

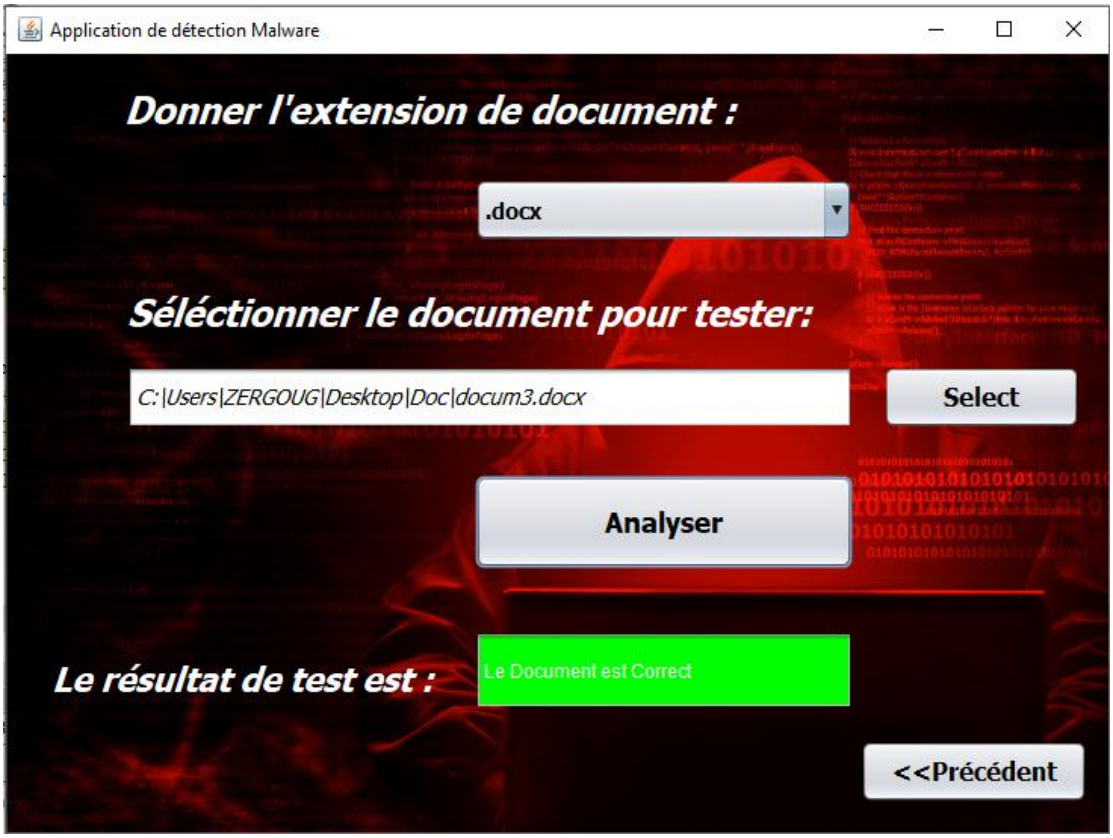

Figure 10. Teste de l'application

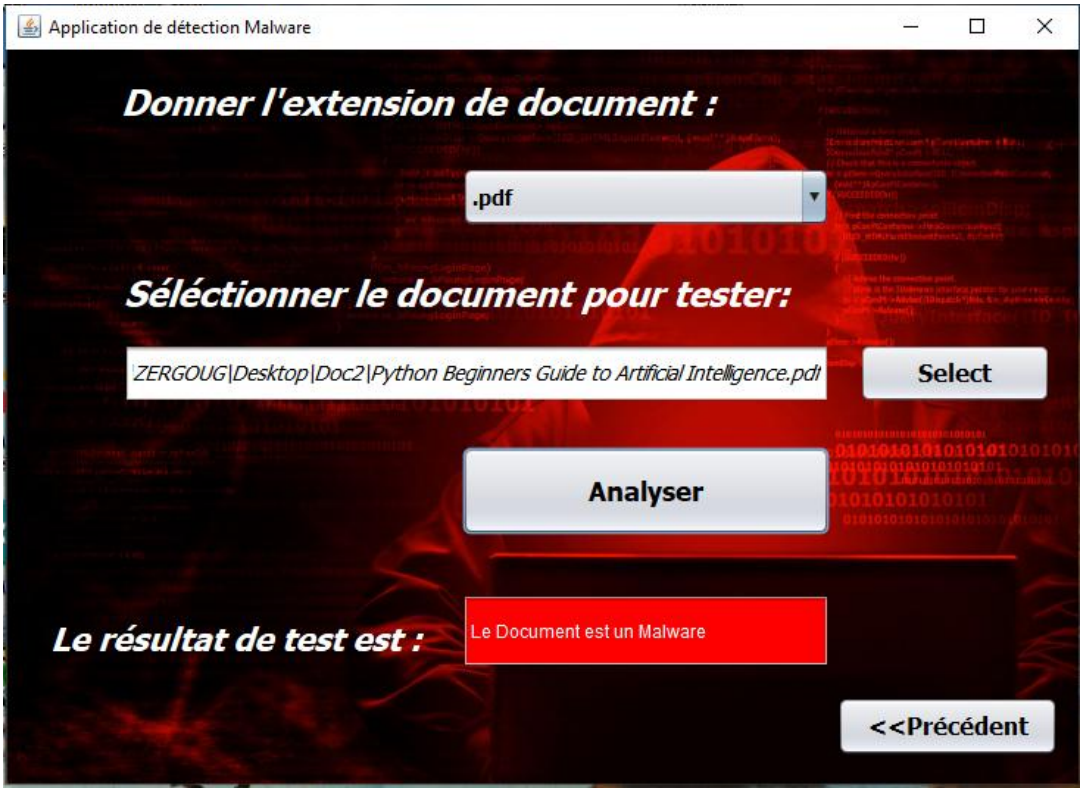

Figure 11. Affichage de détection de malware positif

2.8 Conclusion

Nous avons présenté dans ce chapitre l'algorithme de Distance Mahalanobis, sont origine et la différence entre cette distance et la Distance Euclidienne. Ensuite nous avons résumé les travaux qui sont basés sur la Distance de Mahalanobis. Finalement, nous avons illustré le déroulement de l'algorithme Distance Mahalanobis, et nous avons discuté sur l'environnement de travail. Dans le troisième chapitre, nous détaillerons l'évaluation de performance de cet algorithme.

Chapitre III

Implémentation et évaluation de performance

3.1 Introduction

Dans ce chapitre, nous nous intéressons à la complexité des opérations requises pour le déroulement de notre implémentation de la Distance Mahalanobis. Pour cela nous allons présenter les pseudos algorithmes des différentes opérations principales que nous avons implémenté avec leurs limites asymptomatiques en termes de complexité. Pour terminer nous analyserons le temps d'exécution de celles-ci. L'implémentation présentée dans ce chapitre propose une solution pour l'identification d'anomalie dans des fichiers de structure connus au préalable (par exemple PDF, Word, Excel, JPG). En outre, l'implémentation que nous proposons se base sur les mêmes principes que le travail présenté par Wei-hen Li et al [36].

3.2 Analyse de complexité algorithmique

3.2.1 Définition

La complexité d'un algorithme est une mesure du temps et/ ou de l'espace requis par un algorithme pour une entrée d'une taille donnée (N). Cette complexité est divisée sur deux parties. Premièrement, la complexité temporelle qui décrit le temps nécessaire pour l'exécution d'un algorithme, généralement estimée en comptant le nombre d'opérations élémentaires effectuées par l'algorithme, en supposant que chaque opération élémentaire prend un temps fixe.

Ainsi, la durée et le nombre d'opérations élémentaires effectuées par l'algorithme sont supposés différés d'au plus un facteur constant. En outre, la complexité spatiale est une mesure de l'espace utilisé par un algorithme, est exprimée comme fonction de la taille de l'entrée. Celle-ci est estimée par le nombre maximum de cases mémoires utilisées simultanément pendant un calcul.

3.2.2 La complexité

Afin d'estimer, la complexité d'algorithme Distance Mahalanobis, il faudra calculer la complexité de chaque procédure intermédiaire :

La lecture des documents :

D'abord, cette méthode doit lire le document spécifié par le chemin et obtient une chaine d'octet. La complexité de cette méthode est O(N), avec N la taille de la chaine obtenue.

Extraction des N-grams :

Un N-gram est une séquence de symboles consécutifs. Dans notre cas, un symbole est un octet (0-255). Ainsi, afin de construire l'ensemble des N-gram dans un fichier, une fenêtre glissante, comme celle utilisée dans TCP ou un filtre CNN, est utilisée avec un pas d'avancement d'un symbole. Dans un document de taille *D*, l'ensemble des n-grams sera de taille *D-n+1*.

Notre algorithme utilise des N-grams avec n=1. Ce choix est justifié par la complexité spatiale pour les valeurs de n>1. Les valeurs possibles pour les N-grams selon la taille n sont les suivantes : 0-255 pour 1-grams, 0-65535 pour 2-grams, 0-16777215 pour 3-grams et 0-4294967295 pour 4-grams. La complexité de la création de la liste des n-grams est O(N), avec N est la taille de document en octet.

L'algorithme pour extraire les 1-grams est :

*Entrées : tableau **Contenu_Fichier** Byte.*

Initialisation :
***N_grams**, liste d'octets*.

Début
*Pour i allant de 0 à taille de (**Contenu_Fichier**) Faire*
*Ajouté (**Contenu_Fichier [i..i+1]**) dans la liste **N_grams**.*
Fin pour.

*Retourner (**N_grams**).*

Fin.

Calcul de la matrice générale :

La matrice générale contient dans ses lignes la liste des fréquences pour une valeur d'un n-grams unique. Dans les colonnes de celle-ci, les listes des fréquences des 1-grams correspondant aux documents inclus dans la base de documentaire. L'algorithme suivant effectue ce calcule :

*Entrées : tableau **Liste_Fichiers** file //(Répertoire des documents)*

Initialisation :

***Matrix_generale**, matrice d'éléments réels, taille 256 lignes et nombre de colonnes égale à taille de **Liste_Fichiers**.*

***Somme**, tableau d'éléments entiers, taille=taille de **Liste_Fichiers**.*

***Fichier,** pointeur vers fichier à lire.*

***Document,** tableau d'éléments octets, taille= taille du **Fichier** lu*

Liste_Ngr,** liste de **Ngrams** , taille =taille de **Document

*Entier **Valeur**.*

Début

*Pour i allant de 0 à taille de répertoire **Liste_Fichiers** Faire*

***Fichier** <- **Liste_Fichiers** [i] // (récupérer le **Fichier**)*

***Document** <- Chaine(Chemin de **Fichier**) // (Méthode de lecture de **Fichier**)*

***Liste_Ngr** <- N_grams(**Document**) // (Méthode de liste 1-grams)*

*Pour j allant de 0 à taille de liste **Ngr** Faire*

***Valeur**<- **Liste_Ngr**.get(x) AND_LOGIQUE ((1<<8)-1) //(prend seulement les 8 premier bit)*

***Matrix_generale**[**Valeur**,j] <- **Matrix_generale**[**Valeur**,j] + 1*

***Somme**[j] <- **Somme**[j] + 1*

Fin pour.

Fin pour.

Pour i allant de 0 à 255 Faire

*Pour j allant de 0 à taille de répertoire **Liste_Fichiers** Faire*

***Matrix_generale** [i, j] <- **Matrix_generale** [i, j] / **Somme**[j].*

Fin pour.

Fin pour.

*Retourner (**Matrix_generale**)*

Fin

La complexité de la création de la matrice générale est O (N *M) avec N le nombre de fichiers dans la base de documents ***Liste_Fichiers*** et M la taille de document.

Calcul du vecteur de moyen :

Cette procédure calcule le vecteur moyen à partir de la matrice générale. La complexité de cet algorithme est O(1). L'algorithme de tableau des moyens est le suivant :

Entrées : matrice ***Matrix_generale*** *réel, entier* ***NbreDocument****.*

Initialisation :
Somme, *réel.*
Moyen*, tableau à taille de 256.*

Début

Pour i allant de 0 à 255 Faire
Somme *<- 0.*
Pour j allant de 0 à ***NbreDocument*** *Faire*
Somme *<-* ***Somme*** *+* ***Matrix_generale*** *[i, j].*
Fin pour.
Moyen*[i] <-* ***Somme / NbreDocument****.*
Fin pour.

*Retourner (****Moyen****).*

Fin

Calcul de la matrice de covariance :

L'algorithme de calcul la matrice covariance est :

Entrées : matrice ***Matrix_generale*** *réel, tableau* ***Moyen*** *réel, entier* ***NbreDocument****.*

Initialisation :
***Matrix_covariance**, matrice de nombre réels, taille égale 256 colonnes et 256 lignes*

Début
Pour i allant de 0 à 255 Faire
Pour j allant de i a 255 Faire
***Matrix_covariance**[i,j] <- 0.*

*Pour x allant de 0 à **NbreDocument** Faire*
***Matrix_covariance**[i,j] <- **Matrix_covariance** [i, j] + ((**Matrix_generale**[i,x]-**Moyen**[i])*(**Matrix_generale**[j,x]-**Moyen**[j])).*
Fin pour.
***Matrix_covariance**[i,j]<- (1/(**NbreDocument**-1))*(**Matrix_covariance**[i,j]).*
***Matrix_covariance**[j,i] <- **Matrix_covariance**[i,j].*

Fin pour.
Fin pour.

*Retourner(**Matrix_covariance**).*

Fin

La complexité de cet algorithme est O(N) avec N qui représente le nombre de documents dans la base de documents.

Calcul de Distance Mahalanobis :

Cette opération calcule la Distance Mahalanobis entre le vecteur moyen et le nouveau document analysé. La complexité de cet algorithme est O(1) car la taille du vecteur moyen et du vecteur des fréquences représentatif du document sont constants :

*Entrées : matrice **Ma_Inversé** qui est la matrice inversée de **Matrix_covariance**,*
*tableau **Moyen** réel, matrice **Liste_Frequence** réel.*

Initialisation :
***Resultat**, tableau réel avec une taille de 256.*
***Vecteur**, matrice réel de type doubleMatrix.*
***Distance**, matrice réel de type doubleMatrix.*
Début

Pour i allant de 0 à 256 Faire
***Resultat**[i] <- **Liste_Frequence**[i,0]- **Moyen**[i].*
Fin pour.
***Vecteur**<- **Resultat**.*
***Distance**<- (**Vecteur**)^T * **Ma_inversé** * (**Vecteur**).*

*Retourner (**Distance**).*

Fin

Le résultat de l'application :

La procédure suivante teste si la Distance de Mahalanobis est inclus dans l'intervalle ou pas. La complexité de ce traitement est O(1).

*Entrées : valeur **Distance** réel.*
Début :
*Si (**Distance** > où égale -146,6239 et **Distance** < où égale 146,6239)*
*Retourner(**1**).*
Fin si.
Sinon
*Retourner(**0**).*
Fin sinon.

Fin

3.3 Le temps d'exécution totale d'application

Pour mesurer la performance de l'algorithme en termes de temps d'exécution, il suffit de calculer la différence entre l'instant du début du processus et sa fin. Pour ceci, on a utilisé les procédures suivantes de la bibliothèque Java standard :

long tempsDebut = System.currentTimeMillis(); //cet appel est invoqué au début du processus.

long tempsFin = System.currentTimeMillis(); // calcule le temps à la fin de l'exécution.

Par le calcul de la différence entre le temps du début et de fin d'exécution, nous avons établi un test de temps d'exécution pour notre application. Figure 12 illustre les temps d'exécution nécessaires pour des documents de tailles différentes :

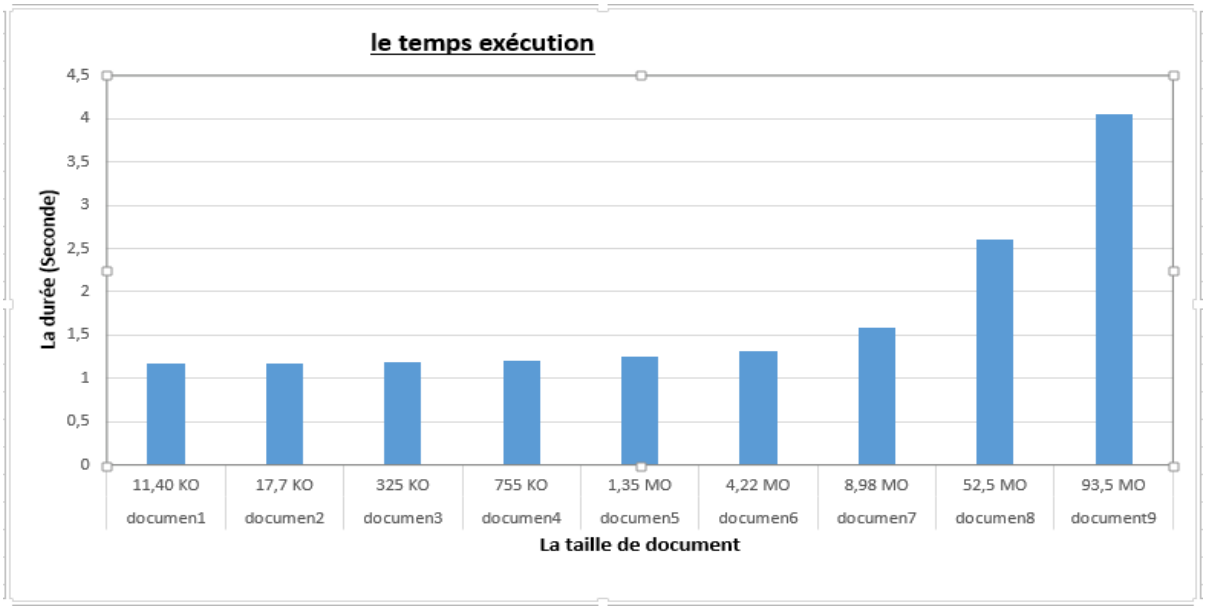

Figure 12. Le temps d'exécution pour des documents de différentes tailles

3.4 Conclusion

Dans ce chapitre, nous avons présenté la complexité des procédures utilisées dans l'algorithme de Distance Mahalanobis ainsi que les algorithmes détaillés de chaque procédure. Par la suite, nous avons calculé le temps d'exécution de notre application en fonction des tailles des documents.

Conclusion générale

Le malware est un terme générique largement utilisé pour désigner tous les différents types de programmes logiciels indésirables, et utilisé par des cybers criminels pour atteindre leurs objectifs. Afin de lutter contre ces logiciels malveillants, on a besoin d'utiliser des applications de détection d'anomalies.

Dans cette perspective, ce travail présente un algorithme pour la détection de Malware, Nous avons présenté dans le premier chapitre les techniques utilisées par les développeurs des malwares. Ainsi que, les techniques de camouflage où d'obscurcissement. En parallèle, nous avons mis en évidence les techniques utilisées par les logiciels anti-programmes malveillants pour l'analyse et la détection de ces derniers. On a finalisé le chapitre par les travaux de recherches focalisés sur la détection de malware.

Nous avons approfondi la méthode principale de notre travail, qui est la Distance de Mahalanobis, dans le deuxième chapitre en présentant la définition de cet algorithme, son origine ainsi que sa formule. Par la suite, nous avons présenté les travaux basés sur cette distance. L'implémentation que nous présentons dans ce mémoire possède l'avantage d'être souple en termes d'exigences mémoire et temps d'analyse. Cette souplesse permet à la Distance de Mahalanobis d'être applicable pour des appareils de faible capacité. L'application la plus adéquate à cet algorithme est l'analyse de pièces jointes reçues par mail. Même si le taux de faux positifs (document non malware détecté comme malware) peut être élevé, cette méthode peut être utilisée comme un prétraitement à l'analyse statique, dynamique ou hybride. Dans un éventuel futur travail, nous souhaitant comparer ce model avec d'autres comme le réseau de neurones et l'arbre de décision.

Références Bibliographiques:

[1] Ravula, R. R., Chien-Chung C., Chand K. M., (2011), Classification of Malware Using and Data Mining Techniques. A thesis presented to The Graduate Faculty of the University of Akron.

[2] Bonaventure O., Detal G., Paasch C., (2013), Systèmes informatiques. https://sites.uclouvain.be/SystInfo/notes/Theorie/html/intro.html, consulté le 28/06/2020.

[3] Christodorescu, M., Jha, S., Seshia, S. A., Song, D., & Bryant, R. E. (2005, May). Semantics-aware malware detection. In 2005 IEEE Symposium on Security and Privacy (S&P'05) (pp. 32-46). IEEE.

[4] McGraw, G., & Morrisett, G. (2000). Attacking malicious code: A report to the infosec research council. IEEE software, 17(5), 33-41.

[5] Vasudevan, A., & Yerraballi, R. (2006, January). Spike: engineering malware analysis tools using unobtrusive binary-instrumentation. In Proceedings of the 29th Australasian Computer Science Conference-Volume 48 (pp. 311-320).

[6] Lawton, G. (2008). Is it finally time to worry about mobile malware?. Computer, 41(5), 12-14.

[7] Killcrece, G., Kossakowski, K. P., Ruefle, R., & Zajicek, M. (2003). State of the practice of computer security incident response teams (CSIRTs) (No. CMU/SEI-2003-TR-001). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

[8] Idika, N., & Mathur, A. P. (2007). A survey of malware detection techniques. Purdue University, 48, 2007-2.

[9] Paul Rascagneres (2013). Malwares Identification, analyse et éradication.

[10] Adware, Spyware and other unwanted; http://www.cexx.org/adware.htm ; Consulté le 14/03/2020.

[11] Rad, B. B., Masrom, M., & Ibrahim, S. (2012). Camouflage in malware: from encryption to metamorphism. International Journal of Computer Science and Network Security, 12(8), 74-83.

[12] You, I., & Yim, K. (2010, November). Malware obfuscation techniques: A brief survey. In 2010 International conference on broadband, wireless computing, communication and applications (pp. 297-300). IEEE.

[13] Beaucamps, P. (2007). Advanced polymorphic techniques. International Journal of Computer Science, 2(3), 194-205.

[14] Szor, P. (2005). The Art of Computer Virus Research and Defense. Pearson

Education.

[15] Or-Meir, O., Nissim, N., Elovici, Y., & Rokach, L. (2019). Dynamic malware analysis in the modern era—A state of the art survey. ACM Computing Surveys (CSUR), 52(5), 1-48.

[16] Christodorescu, M., & Jha, S. (2006). Static analysis of executables to detect malicious patterns. WISCONSIN UNIV-MADISON DEPT OF COMPUTER SCIENCES.

[17] Konstantinou, E., & Wolthusen, S. (2008). Metamorphic virus: Analysis and detection. Royal Holloway University of London, 15, 15.

[18] Khalid, Y., Vashisht, S. O., & Otvagin, A. (2020). U.S. Patent No. 10,581,874. Washington, DC: U.S. Patent and Trademark Office.

[19] Landage, J., & Wankhade, M. P. (2013). Malware and malware detection techniques: A survey. International Journal of Engineering Research and Technology (IJERT), 2(12), 2278-0181.

[20] Jacob, G., Debar, H., & Filiol, E. (2008). Behavioral detection of malware: from a survey towards an established taxonomy. Journal in computer Virology, 4(3), 251-266.

[21] Robiah, Y., Rahayu, S. S., Zaki, M. M., Shahrin, S., Faizal, M. A., & Marliza, R. (2009). A new generic taxonomy on hybrid malware detection technique. arXiv preprint arXiv:0909.4860.

[22] Ye, Y., Li, T., Adjeroh, D., & Iyengar, S. S. (2017). A survey on malware detection using data mining techniques. ACM Computing Surveys (CSUR), 50(3), 1-40.

[23] Schultz, M. G., Eskin, E., Zadok, F., & Stolfo, S. J. (2000, May). Data mining methods for detection of new malicious executables. In Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001 (pp. 38-49). IEEE.

[24] Cohen, W. W. (1995). Fast effective rule induction. In Machine learning proceedings 1995 (pp. 115-123). Morgan Kaufmann.

[25] Ye, Y., Wang, D., Li, T., & Ye, D. (2007, August). IMDS: Intelligent malware detection system. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1043-1047).

[26] Ye, Y., Li, T., Jiang, Q., & Wang, Y. (2010). CIMDS: adapting postprocessing techniques of associative classification for malware detection. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 40(3), 298-307.

[27] Ye, Y., Li, T., Huang, K., Jiang, Q., & Chen, Y. (2010). Hierarchical associative classifier (HAC) for malware detection from the large and imbalanced gray list. Journal of Intelligent Information Systems, 35(1), 1-20.

[28] Tian, R., Islam, R., Batten, L., & Versteeg, S. (2010, October). Differentiating

malware from cleanware using behavioural analysis. In 2010 5th international conference on malicious and unwanted software (pp. 23-30). Ieee.

[29] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1), 10-18.

[30] Wang, T. Y., Horng, S. J., Su, M. Y., Wu, C. H., Wang, P. C., & Su, W. Z. (2006, July). A surveillance spyware detection system based on data mining methods. In 2006 IEEE International Conference on Evolutionary Computation (pp. 3236-3241). IEEE.

[31] Santos, I., Devesa, J., Brezo, F., Nieves, J., & Bringas, P. G. (2013). Opem: A static-dynamic approach for machine-learning-based malware detection. In International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions (pp. 271-280). Springer, Berlin, Heidelberg.

[32] Ye, Y., Li, T., Zhu, S., Zhuang, W., Tas, E., Gupta, U., & Abdulhayoglu, M. (2011, August). Combining file content and file relations for cloud based malware detection. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 222-230).

[33] Taguchi, G., & Jugulum, R. (2002). The Mahalanobis-Taguchi strategy: A pattern technology system. John Wiley & Sons.

[34] Mahalanobis, P. C. (1936). On the generalized distance in statistics. National Institute of Science of India.

[35] McDaniel, M., & Heydari, M. H. (2003, January). Content based file type detection algorithms. In 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the (pp. 10-pp). IEEE.

[36] Li, W. J., Wang, K., Stolfo, S. J., & Herzog, B. (2005, June). Fileprints: Identifying file types by n-gram analysis. In Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop (pp. 64-71). IEEE.

[37] Meghanathan, N., Boumerdassi, S., Chaki, N., & Nagamalai, D. (Eds.). (2010). Recent Trends in Network Security and Applications: Third International Conference, CNSA 2010, Chennai, India, July 23-25, 2010 Proceedings (Vol. 89). Springer.

[38] Liu, H., & Weng, Q. (2012). Enhancing temporal resolution of satellite imagery for public health studies: A case study of West Nile Virus outbreak in Los Angeles in 2007. Remote Sensing of Environment, 117, 57-71.

[39] Shah, N. K., & Gemperline, P. J. (1990). Combination of the Mahalanobis distance and residual variance pattern recognition techniques for classification of near-infrared reflectance spectra. Analytical Chemistry, 62(5), 465-470.

[40] Foody, G. M. (2004). Sub-pixel methods in remote sensing. In Remote sensing image analysis: Including the spatial domain (pp. 37-49). Springer, Dordrecht.

[41] JBlas https://mikiobraun.github.io/jblas/jars/jblas-1.2.4.jar, consulté le 22/04/2020.