REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE BLIDA 01

Faculté des sciences
Département d'informatique

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du diplôme Master en

Informatique Option : Système Informatique et Réseaux

THÈME :

Détection des ransomwares en utilisant les techniques d'intelligence artificielle (Machine/Deep Learning)

Réaliser par : Naoui Oussama

- Mr. Guerroumi Mohamed — Encadreur
- Mr. Bitit Rahmoune — Co-Encadreur

Blida, Septembre 2022

Dédicaces

Je dédie ce mémoire à :

Mon père, qui peut être fier et trouver ici le résultat de longues années de sacrifices et de privations pour m'aider à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutien permanent venu de toi.

Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie, reçois à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude..

Je tiens à adresser mes plus vifs remerciements à Mr Guerroumi Mohamed, Mr Derhab Abdelouhid et Mr Bitit Rahmoune pour m'avoir encadré et pour les recommandations qu'ils m'ont prodiguées et qui m'ont été d'un grand apport.

Je tiens à adresser mes plus vifs remerciements aux membres du jury, pour avoir accepté d'évaluer mon travail.

Mes professeurs de département informatique qui doivent voir dans ce travail la fierté d'un savoir bien acquis.

Remerciements

En tout premier lieu, je remercie Dieu, tout puissant, de m' avoir donné la force pour survivre, ainsi que l'audace pour dépasser toutes les difficultés.

J'adresse mes grands remerciements à mon promoteur Guerroumi Mohamed Mr Derhab Abdelouhid et Mr Bitit Rahmoune d'avoir accepté de diriger ce travail, pour tous leurs conseils, leur patience et leur orientation tout au long de l'accomplissement de ce travail sans lesquels ce travail n'aurait pas vu le jour.

J'adresse aussi mes remerciements au président et aux membres du jury d'avoir accepté le jugement de mon travail.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail trouvent ici mes sincères remerciements.

Résumé

Ransomware ou Rançongiciel en français est un logiciel malveillant qui est conçu pour chiffrer les informations des utilisateurs et/ou verrouiller les systèmes ciblés. L'objectif principal des attaques par Ransomware est de demander une rançon pour le décryptage des données des utilisateurs et/ou le déverrouillage des systèmes affectés. Les montants de paiement de rançon vont de quelques centaines à des centaines de milliers de dollars. Payable en crypto-monnaies comme le Bitcoin. Cependant, payer la rançon ne garantit pas l'obtention de la clé de déchiffrement et/ou le déverrouillage de système infecté.

Les dernières années ont prouvé que les ransomwares constituaient une menace majeure pour les systèmes informatiques, où des individus, des gouvernements, des entreprises, des hôpitaux et diverses industries ont été victimes de cette menace.
Ce type de malware est devenu une menace sérieuse pour le monde informatique. Il existe donc un besoin réel des méthodes capables de lutter contre ce type d'attaque. Plusieurs méthodes ont été proposées dans la littérature, dont le but est de détecter l'attaque, classifier les ransomwares, ou détecter la pré-attaque d'un ransomware. Néanmoins, ces études restent limitées par les données et les familles des ransomwares utilisées.

Dans ce contexte, ce projet est proposé dans le but est d'étudier et d'analyser les méthodes de détection existantes et ensuite proposer et évaluer une approche de détection en utilisant les techniques de Machine/Deep Learning selon le plan suivant :

- Une étude sur les Ransomwares (caractéristiques, familles, …)
- Un état de l'art sur la détection des Ransomwares
- Proposition d'une méthode de détection des Ransomwares
- Implémentation et évaluation de la méthode proposée.

Table des figures

Liste des tableaux

TABLE DES MATIERES

Chapitre II :

L'état de l'art sur la détection des ransomwares

Chapitre III :

Proposition d'une méthode de détection des ransomwares

Chapitre IV :

Implémentation et évaluation

Introduction générale

De nos jours, les attaquants des systèmes informatiques utilisent des techniques intelligentes pour générer de nouveaux types de logiciels malveillants rentables. Parmi ces attaques qui sont fortement répandue récemment sont les Ransomwares. Contrairement aux autres problèmes de sécurité, les ransomwares sont irréversibles et difficiles à arrêter [1].

Le ransomware (rançongiciel en français) est désormais une arme populaire entre les mains d'acteurs malveillants qui tentent quotidiennement de nuire aux gouvernements, aux entreprises et aux particuliers. Dans ce cas, la victime du rançongiciel est susceptible de subir des pertes économiques, soit en payant la rançon demandée, soit en payant les frais de recouvrement de la perte si celle-ci ne respecte pas les exigences de l'attaquant. Au vu du nombre croissant d'incidents, il est évident que l'hypothèse ne repose pas sur le fait de savoir «si» on en sera victime mais plutôt «quand» cela se produira. Cependant, dans la majorité des pays qui luttent contre les rançongiciels, plusieurs problématiques doivent encore être abordées, comme le manque de coordination et de collaboration entre les agences et les autorités, ainsi que l'absence de législation permettant clairement de faire des attaques par rançongiciel un délit.

La stratégie de ce malware est basée sur la restriction d'accès aux fichiers des utilisateurs en les chiffrant et demande une rançon afin d'obtenir la clé de déchiffrement. Selon Symantec Corporation 2016, des centaines de millions de dollars sont imposés pour être payés par les utilisateurs en rançon chaque année. En 2016, Osterman Research and Inc. [2] a mené une enquête auprès d'environ 290 organisations de divers secteurs industriels en Europe et aux États-Unis. L'enquête a révélé que 50% d'entre eux avaient été victimes d'un ransomware pendant un an. Environ 40% de ces victimes ont payé aux agresseurs. D'autre part, un rapport statistique de VirusTotal (https://www.virustotal.com/en/statistics) a décrit qu'en février 2017, environ 1,37 million de nouveaux échantillons de cyber-attaques ont été soumis [3].

La différence essentielle entre les logiciels malveillants et les rançongiciels par le temps pris pour l'attaque et le comportement de l'attaque. Alors que les logiciels malveillants se cachent derrière les applications, puis infectent et endommagent l'ordinateur sans demander de payer la rançon [4]. Selon Chittooparambil et al. [1], aucune des méthodes existantes ne permet de détecter et de stopper ce type d'attaque. Par ailleurs, Wecksten, M., et al. [5] et Kharraz, A., et al. [6], a confirmé la difficulté d'arrêter ce type d'attaque. Par conséquent, il est urgent d'introduire une nouvelle technique pouvant être utilisée pour détecter les ransomwares.

Dans ce cadre, le présent travail a pour but d'étudier et analyser les méthodes de détection des ransomwares existantes et ensuite proposer et évaluer une approche de détection en utilisant les techniques de Machine/Deep Learning.

Pour bien mener ce projet, on a découpé ce document en quatre chapitres :

- Le premier chapitre est consacré à l'étude des ransomware : définition, classes, fonctionnement, types ainsi que auteurs.
- Le deuxième chapitre est consacré à l'état de l'art sur la détection des ransomwares par la présentation des différentes méthodes de détection: dynamique, statique et hybride utilisée par les chercheurs et une comparaison entre ces méthodes.
- Le troisième chapitre est consacré à la présentation de la méthode de détection proposée en utilisant « Le Deep Learning ».
- Le dernier chapitre est consacré à l'implémentation et l'évaluation de la méthode proposée en utilisant les environnements : Anaconda, Jupyter et Cuckoo Sandbox

On clôture ce document par une conclusion générale.

Chapitre 1: Généralités sur les Ransomwares

Introduction :

Les ransomwares sont aujourd'hui l'une des plus grandes menaces de cyber-sécurité auxquelles sont confrontés les entreprises et les particuliers. Dans ce chapitre, nous présentons des généralités sur les ransomwares. En particulier, nous présentons leur définition, leurs classes et leurs types, leurs modes, leurs caractéristiques et en fin leurs étapes d'exécution.

1.1 La définition des ransomwares :

Un ransomware est un type de malware, c'est-à-dire un programme malveillant, qui chiffre des fichiers, voire tout un ordinateur, puis exige une rançon en échange de l'accès aux fichiers ou à l'ordinateur. Les ransomwares utilisent le chiffrement pour bloquer l'accès aux fichiers ou aux ordinateurs infectés rendus inutilisables par la victime. Ils s'attaquent en général à tous les types de fichiers, documents personnels ou professionnels.

Une fois que le ransomware a mené à bien son attaque, les pirates ou les cybercriminels qui en sont à l'origine contactent la victime pour lui communiquer leurs exigences, en promettant de débloquer son ordinateur ou de déchiffrer ses fichiers une fois la rançon payée (figure 1.1), en général en bitcoins ou autre cryptomonnaie.

Bien que la prise de conscience générale du danger croissant que représentent les ransomwares remonte au milieu des années 2000, les attaques de ransomwares ciblent les particuliers, les entreprises et les gouvernements depuis plusieurs décennies. La première attaque de ransomware attestée, connue sous le nom de AIDS Trojan (cheval de Troie SIDA) ou PC Cyborg, a été lancée en 1989 par le Dr Joseph Popp, un biologiste de l'évolution formé à Harvard [7].

Figure 1.1 Capture d'écran: Demande de rançon d'un ransomware

1.2 Les classes des ransomwares

Il est possible de classifier les attaques par ransomware selon trois types :

1.2.1. Les campagnes d'attaques non ciblées : Caractérisées par leur faible coût de mise en œuvre ainsi que par leur faible sophistication. Généralement, la campagne d'infection est massive et les cibles manquent de protection numérique.

1.2.2. Les campagnes massives automatiques : Représentées par l'unique exemple qu'est WannaCry .La particularité de cette campagne d'attaques est qu'elle n'a nécessité aucune interaction avec la victime pour l'infecter et aucune action manuelle de l'attaquant pour se propager dans son réseau. L'attaque Wannacry mettait en œuvre un code d'exploitation de vulnérabilité appelé EternalBlue, supposément développé par la NSA et divulgué en source ouverte deux mois plus tôt par l'avatar Shadow Broker.

1.2.3. Les attaques ciblées dites « Big Game Hunting » : En recrudescence depuis 2018. Elles ne reposent plus sur un grand nombre de compromissions pour générer de l'argent, mais sur la capacité de l'attaquant à se propager au sein du réseau ciblé de manière furtive et à identifier et chiffrer les ressources clés de la cible, ainsi que la capacité financière de la cible à payer des rançons de montant important et la criticité de sa continuité d'activité.

1.3. Fonctionnement des ransomwares

Un point commun à la plupart des ransomwares est le fait que ceux-ci doivent pouvoir être contrôlés à distance (récupération d'informations, actions à exécuter, etc.) par leurs auteurs. Cette définition permet d'introduire le terme de botnet. Un botnet est un ensemble de postes infectés (bot ou zombie) contrôlés sur Internet par un cybercriminel (le botmaster). La figure 1.2 présente un schéma simple d'un botnet.

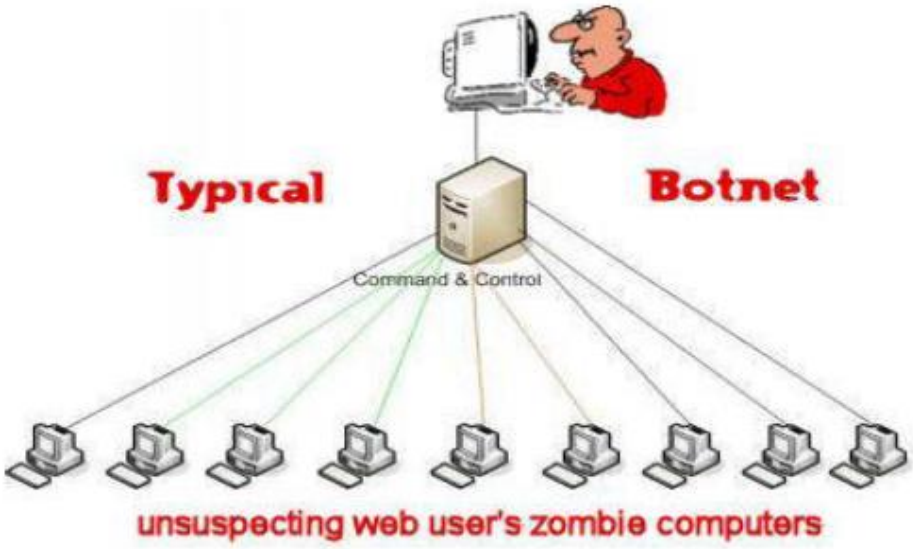


Figure 1.2 Schéma général d'un botnet

Plus un botnet compte de machines infectées, plus ses capacités de nuisance et son profit seront grands. C'est le nombre qui fait la force des botnets . Depuis des années, on découvre ainsi fréquemment des botnets composés de millions d'ordinateurs infectés.

Parmi les botnets les plus connus, nous pourrions citer : Storm, Rustock et Srizb.

1.4 Les modes d'infection :

Un ransomware peut infecter votre ordinateur de plusieurs manières. L'une des méthodes les plus courantes actuellement consiste à utiliser des spams malveillants (malspams), qui sont des e-mails indésirables employés pour livrer des malwares. Ces e-mails peuvent inclure des pièces jointes piégées (PDF, fichiers Word, etc.) ou des liens vers des sites Web malveillants.

Les malspams ont recours à l'ingénierie sociale afin d'inciter les utilisateurs à ouvrir des pièces jointes ou à cliquer sur des liens qui semblent provenir de sources légitimes, comme un ami ou une institution. Les cybercriminels utilisent l'ingénierie sociale dans d'autres types d'attaques de ransomwares, par exemple lorsqu'ils se font passer pour le FBI afin d'intimider les utilisateurs et les persuader de leur payer une rançon pour déverrouiller leurs fichiers.

Une autre méthode d'infection est le malvertising, qui s'est avéré particulièrement populaire en 2016. Le malvertising, ou publicité malveillante, est une méthode qui consiste à utiliser des publicités en ligne pour distribuer des malwares et qui nécessite peu ou pas d'interactions avec les utilisateurs. Alors qu'ils surfent sur le Web, y compris sur des sites légitimes, les utilisateurs peuvent être renvoyés vers des serveurs criminels sans avoir même cliqué sur une seule publicité. Ces serveurs répertorient des informations concernant les ordinateurs des victimes et leur emplacement, puis ils sélectionnent les malwares les plus susceptibles de les infecter. Bien souvent, ce malware est un ransomware.

Le malvertising utilise souvent un iframe (élément invisible d'une page web). Celui-ci renvoie vers une page contenant un exploit et le code malveillant attaque le système depuis cette page via un kit d'exploit. Ce type d'attaque s'appelle un « drive-by download », ou téléchargement intempestif, car l'utilisateur n'a aucune idée de ce qui est en train de se passer.

1.5 Les types de ransomwares :

En fonction de la manière dont ils affectent le fonctionnement de votre ordinateur, la plupart des programmes de ransomware d'aujourd'hui sont classés en les deux types suivants :

1.5.1 Verrouilleurs d'ordinateur (Computer Lockers):

Aussi connus sous le nom de ransomware de verrouillage, les verrouilleurs d'ordinateur bloquent l'accès à l'interface de votre ordinateur, vous empêchant ainsi de l'utiliser. Si votre ordinateur est infecté par un ransomware de verrouillage, un écran comportant un message de son auteur ainsi que les instructions de paiement apparaissent au démarrage du système. L'auteur peut aussi tenter de vous persuader que la rançon est en fait une amende qui vous a été émise par une autorité légale.

Ce type de ransomware empêche généralement uniquement l'accès à l'interface de votre ordinateur et n'affecte ni les fichiers ni le système. Vous pouvez donc pouvoir supprimer le ransomware et conserver vos fichiers intacts.

1.5.2 Bloqueurs de données :

Étant donné qu'ils modifient les fichiers individuels et ne bloquent pas simplement l'accès à l'interface de l'ordinateur, les bloqueurs de données sont potentiellement plus dangereux que les verrouilleurs d'ordinateur. Aussi connu sous le nom de crypto ransomware, ce type de logiciel scanne votre ordinateur à la recherche de fichiers importants et change leur extension. Il la remplace pour une extension que votre ordinateur ne sera pas capable de reconnaître. Pour débloquer vos fichiers, vous devez payer la rançon et obtenir la clé de décryptage.
Les pirates derrière les bloqueurs de données ciblent principalement les individus qui ne sauvegardent pas régulièrement leurs données importantes. Une fois confrontés à la possibilité de perdre tous leurs fichiers, les victimes sont plus susceptibles de payer la rançon. Un bon décrypteur de ransomware peut néanmoins les aider à regagner l'accès à leurs fichiers sans payer.

1.5.3 Autres types de ransomware

De nouveaux types de ransomware ont fait leur apparition ces dernières années, notamment :

1.5.3.1 Locky

Locky est un ransomware qui a été utilisé pour la première fois dans le cadre d'une attaque menée en 2016 par un groupe de pirates informatiques organisés. Locky a chiffré plus de 160 types de fichiers et s'est propagé au moyen de faux emails contenant des pièces jointes infectées. Les utilisateurs se sont laissés prendre au piège de l'émail et ont installé le ransomware sur leurs ordinateurs. Cette méthode de diffusion s'appelle le phishing et constitue une forme de ce que l'on appelle l'ingénierie sociale. Le ransomware Locky cible des types de fichiers qui sont souvent utilisés par les concepteurs, les développeurs, les ingénieurs et les testeurs[8].

1.5.3.2 WannaCry

WannaCry était un ransomware qui s'est propagé dans plus de 150 pays en 2017. Celui-ci a été conçu pour exploiter une faille de sécurité dans Windows, créée par la NSA et divulguée

par le groupe de pirates informatiques Shadow Brokers. WannaCry a touché 230 000 ordinateurs à l'échelle mondiale. L'attaque a touché un tiers de tous les hôpitaux du NHS au Royaume-Uni, causant des dommages estimés à 92 millions de livres sterling. Les ordinateurs des utilisateurs ont été verrouillés, et une rançon payable en Bitcoins leur a été demandée. L'attaque a mis en évidence le problème des systèmes obsolètes, car le pirate informatique a exploité une vulnérabilité du système d'exploitation pour laquelle un correctif existait depuis longtemps au moment de l'attaque. Le préjudice financier mondial causé par WannaCry s'élève à environ 4 milliards de dollars américains[8].

1.5.3.3 Bad Rabbit

Bad Rabbit était un ransomware de 2017 qui s'est propagé via des attaques par téléchargement furtif. Des sites Web non sécurisés ont été utilisés pour mener à bien ces attaques. Dans le cadre d'une attaque par téléchargement furtif par ransomware, un utilisateur visite un site Web réel, sans savoir que celui-ci a été compromis par des pirates informatiques. Pour la plupart des attaques par téléchargement furtif, il suffit qu'un utilisateur appelle une page qui a été compromise de cette manière. Dans ce cas, cependant, l'exécution d'un programme d'installation contenant une application malveillante camouflée a conduit à l'infection. Il s'agit d'une application malveillante furtive. Bad Rabbit demandait à l'utilisateur d'exécuter une fausse installation d'Adobe Flash, infectant ainsi l'ordinateur avec des applications malveillantes[8].

1.5.3.4 Ryuk

Ryuk est un cheval de Troie de chiffrement qui s'est propagé en août 2018 et qui désactive la fonctionnalité de récupération des systèmes d'exploitation Windows. Il était donc impossible de restaurer les données chiffrées sans une sauvegarde externe. Ryuk chiffrait également les disques réseau. Les conséquences ont été énormes, et de nombreuses organisations américaines ciblées ont payé les rançons demandées. Le total des dommages est estimé à plus de 640 000 $[8].

1.5.3.5 Shade/Troldesh

L'attaque par ransomware Shade ou Troldesh a eu lieu en 2015 et s'est propagée via des emails de spam contenant des liens ou des pièces jointes de fichiers infectés. Il est intéressant

de noter que les attaquants de Troldesh communiquaient directement avec leurs victimes par email. Les victimes avec lesquelles ils avaient établi une « bonne relation » recevaient des réductions. Toutefois, ce type de comportement est une exception plutôt que la règle[8].

1.5.3.6 Jigsaw

Jigsaw est une attaque de ransomware qui a commencé en 2016. L'attaque doit son nom à l'image qu'elle a affichée de la célèbre marionnette de la franchise Saw. À chaque heure supplémentaire où la rançon n'était pas payée, le ransomware Jigsaw supprimait davantage de fichiers. L'utilisation de l'image du film d'horreur provoque un stress supplémentaire auprès des utilisateurs[2].

1.5.3.7 CryptoLocker

CryptoLocker est un ransomware qui est apparu pour la première fois en 2007 et qui s'est propagé via des pièces jointes infectées. Le ransomware recherchait des données importantes sur les ordinateurs infectés et les chiffrait. On estime que 500 000 ordinateurs ont été touchés. Les forces de l'ordre et les entreprises de sécurité ont finalement réussi à prendre le contrôle d'un réseau mondial d'ordinateurs domestiques détournés qui ont été utilisés pour diffuser CryptoLocker. Cette mesure a permis aux agences et aux entreprises d'intercepter les données envoyées sur le réseau sans que les criminels s'en aperçoivent. En fin de compte, il était possible d'accéder à un portail en ligne où les victimes pouvaient obtenir une clé leur permettant de déverrouiller leurs données. Leurs données pouvaient ainsi être libérées sans qu'il soit nécessaire de payer une rançon aux criminels[8].

1.5.3.8 Petya

Petya (à ne pas confondre avec ExPert) est un ransomware qui est apparu pour la première fois en 2016 et qui a été ressuscité sous le nom de GoldenEye en 2017. Au lieu de chiffrer certains fichiers, ce ransomware chiffrait l'intégralité du disque dur de la victime. Pour ce faire, le Master File Table (MFT) était chiffré, ce qui rendait impossible l'accès aux fichiers du disque dur. Le ransomware Petya s'est propagé dans les services RH des entreprises par le biais d'une fausse application contenant un lien Dropbox infecté.

Une autre variante de Petya est Petya 2.0, qui diffère sur certains aspects essentiels. Toutefois, en ce qui concerne les conditions d'exécution de l'attaque, les deux variantes sont tout aussi fatales pour l'appareil[9].

1.5.3.9 GoldenEye

La résurrection de Petya sous le nom de GoldenEye a entraîné une infection mondiale par ransomware en 2017. GoldenEye, surnommé le « frère mortel » de WannaCry, a touché plus de 2 000 cibles, dont d'importants producteurs de pétrole en Russie et plusieurs banques. Dans un retournement de situation alarmant, GoldenEye a forcé le personnel de la centrale nucléaire de Tchernobyl à vérifier manuellement le niveau de radiation sur place, après que les ordinateurs Windows du personnel ont été verrouillés[8].

1.5.3.10 GandCrab

GandCrab est un ransomware désagréable qui menaçait de divulguer les habitudes pornographiques de ses victimes. Il prétendait avoir piraté la webcam de la victime et lui demandait une rançon. Si la rançon n'était pas payée, des images embarrassantes de la victime étaient censées être publiées. Après sa première apparition en 2018, le ransomware GandCrab a continué de se développer en plusieurs versions. Dans le cadre de l'initiative « No More Ransom », les fournisseurs de sécurité et les services de police ont conçu un outil de déchiffrement des ransomwares pour permettre aux victimes de récupérer leurs données sensibles après qu'elles ont été infectées par le virus GandCrab[8].

1.5.3.11 B0r0nt0k

B0r0nt0k est un ransomware cryptographique qui vise particulièrement les serveurs Windows et Linux. Ce ransomware nuisible chiffre les fichiers d'un serveur Linux et y joint une extension de fichier « .rontok ». L'application malveillante ne menace pas seulement les fichiers, elle modifie également les paramètres de démarrage, désactive des fonctions et des applications, et ajoute des entrées de registre, des fichiers ainsi que des programmes[8].

1.5.3.12 Le ransomware Dharma Brrr

Brrr, le nouveau ransomware Dharma, est installé manuellement par des pirates informatiques qui s'introduisent dans les services de bureau connectés à Internet. Dès que le ransomware est

activé par les pirates informatiques, il commence à chiffrer les fichiers qu'il trouve. Les données chiffrées portent l'extension de fichier suivante : « .id-[id].[email].brrr »[9].

1.5.3.13 Le ransomware FAIR RANSOMWARE

FAIR RANSOMWARE est un ransomware qui vise à chiffrer les données. Grâce à un algorithme puissant, tous les documents et fichiers privés de la victime sont chiffrés. Les fichiers chiffrés par cette application malveillante portent l'extension « .FAIR RANSOMWARE »[2].

1.5.3.14 Le ransomware MADO

Le ransomware MADO est un autre type de ransomware Crypto. Les données chiffrées par ce ransomware portent l'extension « .mado » et ne peuvent donc plus être ouvertes[9].

1.6 Les auteurs de ransomware

Les auteurs de ransomware peuvent être divers (personne isolée, équipe, crime organisé, état, etc.), néanmoins au fil des années, force est de constater leur professionnalisation ainsi que la complexité des malwares.

Parmi les pays les plus actifs dans le domaine, les russes ainsi que les chinois tiennent une place assez prépondérante dans le classement.

1.7 Conclusion

Les ransomwares sont devenus une réalité pour la majorité des utilisateurs et bien peu d'entre eux y échappent (en particulier sous les systèmes Windows & Android). Créé au départ par jeu ou par défi, ils sont vite devenus au fil des années de véritables armes numériques. Connaître leur mode de fonctionnement ainsi que les méthodes de propagation restent le meilleur moyen de s'en protéger. Dans le chapitre suivant, nous allons présenter un état de l'art sur les ransomwares.

Chapitre 2: L’état de l’art sur la détection des ransomwares

Introduction :

De nos jours, le ransomware est devenu une menace sérieuse défiant le monde informatique qui nécessite une réflexion immédiate pour éviter le chantage financier et moral.

Le processus de détection de ransomware permet de stopper ce type d'attaque. La plupart des méthodes de détection suivaient une technique d'analyse dynamique qui implique un processus compliqué. D’autres méthodes utilisent l'analyse statique pour détecter les ransomwares.

Dans ce chapitre, nous présentons les différentes techniques de détection de ransomwares. Pour avoir une vision claire sur ces techniques, nous les comparons selon plusieurs paramètres.

2.1. Les techniques de détection de ransomwares :

D'une manière générale, les techniques de détection sont classées en trois catégories : la première est l’analyse dynamique, la deuxième est l’analyse statique, et la troisième analyse utilise un système hybride qui combine l'analyse dynamique et l'analyse statique.

2.1.1 L’analyse dynamique :

La plupart des solutions de détection de ransomwares reposent sur la détection comportementale appelée « analyse dynamique » [10–11].

Takeuchi et al. [12] ont proposé une méthode d'analyse dynamique pour détecter des ransomware basé sur le classificateur SVM. Ils ont d'abord extrait une fonctionnalité de rançongiciel spécifique appelée la programmation d'application d'interface (API), puis étudiez l'historique des appels d'API et ses comportements à l'aide de Cuckoo Sandbox.

Les appels API sont représentés par des vecteurs q-gram. Ils ont utilisé 276 ransomwares et 312 fichiers de goodwares. Les résultats ont montré une précision de 97,48% dans la détection rançongiciel utilisant SVM.

Vinayakumar et al. [13] ont proposé une nouvelle méthode en utilisant l'analyse dynamique pour collecter les séquences d'API à partir d'un bac à sable(sandbox).

Dans leur expérience, ils ont téléchargé sept familles de rançongiciels. En utilisant le perceptron multicouche (MLP) pour classer les ransomwares et les goodwares.

Cette méthode a révélé une précision de 98 %.

Kharraz et al. [10] ont utilisé un système d'analyse dynamique appelé UNVEIL pour détecter les ransomwares. Le système crée un environnement d'exécution artificiel et réaliste pour détecter les ransomwares. Ce système présentait une précision d'environ 96,3 %.

Homayoun et al. [11] ont introduit un système de détection de rançongiciel basé sur Sequential Pattern Mining en tant que fonctionnalités candidates à utiliser comme entrée pour les techniques d'apprentissage automatique (MLP, Bagging, Random Forest et J48) à des fins de classification. Les résultats ont montré une précision d'environ 99 % pour la détection des rançongiciels.

Weckst'en et al. [14] ont analysé le comportement pour quatre types de crypto ransomwares dans une machine virtuelle installée dans le système d'exploitation Windows 7. Les auteurs ont utilisé le moniteur de processus logiciel, la manipulation du registre, l'activité du système de fichiers et les regshots pour suivre l'activité du processus. Ils ont affirmé que les attaques de crypto-ransomware dépendent essentiellement du fichier vssadmin.exe. Par conséquent, les utilisateurs doivent éviter d'accéder au logiciel vssadmin.exe afin d'empêcher cette attaque.

Tseng et al. [15] ont utilisé une méthode d'apprentissage en profondeur pour analyser le comportement des ransomwares à partir du fichier d'en-tête des paquets réseau.

Chen et al. [16] ont proposé un réseau contradictoire génératif(GAN). La technique proposée pourrait produire automatiquement des fonctionnalités dynamiques.

On remarque dans les travaux précèdent que la méthode d'analyse dynamique utilisée pour la détection des ransomwares a un taux de précision élevé . Cependant, cette analyse prend un temps relativement long à traiter et à analyser, alors qu'à ce stade, la charge utile malveillante a probablement déjà été livrée [17]. En même temps, si l'environnement est identifié par le ransomware, il n'est pas en mesure d'extraire des séquences d'API significatives [18].

2.1.2 L'analyse statique :

Quelques analystes ont proposé des méthodes basées sur l'analyse statique pour détecter les attaques de ransomware.

Une étude récente menée par Zhang et al. [18] utilise des opcodes basés sur des fonctionnalités pour la détection des rançongiciels. Cette méthode comprenait le transfert de séquences d'opcodes vers des séquences de N-grammes, puis Term Frequency-Fréquence de document inverse (TF-IDF).

Cinq méthodes d'apprentissage automatique ont été utilisées pour faire la distinction entre les ransomwares et les goodwares tels que : Arbre de décision, forêt aléatoire, K-Nearest Neighbor, Naive Bayes et Gradient boosting. La meilleure précision de 91,43 % a été obtenue en utilisant la forêt aléatoire.

Baldwin et Dehghantanha [19] ont utilisé l'analyse statique pour détecter les ransomwares. Ils ont extrait les caractéristiques de l'opcode en tant que fonctionnalités à utiliser comme entrée de la technique d'apprentissage automatique représentée par le classificateur SVM. L'ensemble d'outils d'apprentissage automatique WEKA a été utilisé dans ce travail. La meilleure précision obtenue était d'environ 96,5 % pour cinq familles de crypto-ransomwares.

2.1.3 L'analyse hybride :

Certains chercheurs ont utilisé une technique hybride combinant l'analyse dynamique et l'analyse statique pour détecter les ransomwares.

Subedi et al. [20] ont utilisé l'analyse dynamique et l'analyse statique à trois niveaux différents ; assemblage, appels de fonction et bibliothèque. En outre, ils ont conçu CRSTATIC qui est un outil d'analyse qui crée des signatures pour identifier les familles de

ransomwares en utilisant l'ingénierie inverse. Shaukat et Ribeiro [21] ont introduit une Strong Trap Layer en utilisant un système hybride d'analyse dynamique et statique avec l'utilisation de techniques d'apprentissage automatique. Ils ont utilisé 74 échantillons de 12 familles Cryptographic Ransomware. Les résultats ont montré un taux de détection d'environ 98,25 % en utilisant l'algorithme Gradient Tree Boosting.

Ferrante et al. [22] ont proposé une approche hybride pour la détection des rançongiciels Android. Le système est combiné entre analyse dynamique et statique. La méthode de détection dynamique prend en compte l'utilisation de la mémoire, les statistiques des appels système, l'utilisation du processeur et l'utilisation du réseau, tandis que la méthode de détection statique utilise la fréquence des opcodes.

Honeypot est une autre technique utilisée par de nombreux chercheurs pour détecter les ransomwares.

Moore [23] a utilisé un dossier pot de miel pour surveiller les changements se produisant dans le dossier. Quelques chercheurs ont inventé des outils spéciaux pour détecter les ransomwares.

Kolodenker et al. [24] ont proposé un outil PayBreak qui stocke les clés de chiffrement cryptographiques dans un coffre de clés. Ces clés sont utilisées pour déchiffrer les fichiers affectés après une attaque de ransomware.

Dans un autre travail, Scaife et al. [25] ont suggéré d'utiliser le système CryptoDrop qui alerte l'utilisateur lors d'une activité de fichier suspecte à l'aide d'un ensemble d'indicateurs de comportement.

Continella et al. [26] ont introduit le système ShieldFS qui analyse la mémoire de processus et recherche toute indication d'utilisation de la cryptographie.

D'après les connaissances de l'auteur, aucun travail antérieur n'a tenté de détecter les ransomwares à l'aide d'une analyse statique pour les fonctionnalités au niveau de l'octet. Dans la présente étude, l'analyse statique au niveau de l'octet a été utilisée pour surmonter les lacunes de l'analyse dynamique. Les fonctionnalités sont extraites directement des octets bruts du fichier exécutable, puis l'extraction fréquente de modèles a été utilisée.

Pour le processus de classification, le classificateur d'apprentissage automatique Random Forest a été utilisé pour classer les fichiers de ransomware et de goodware.

2.2 Comparaison de techniques utilisées pour la détection des ransomwares

La technique utilisée (Algorithme)	Type d'analyse	Nom du Data Set	Nom d'auteur	La précision de la technique
SVM	L'analyse dynamique	La programmation d'application d'interface (API) appel	Takeuchi et al	97,48%
Le perceptron multicouche (MLP)	L'analyse dynamique	sept familles de rançongiciels.	Vinayakumar et al	98%
UNVEIL	L'analyse dynamique	Le système travail avec un environnement d'exécution artificiel	Kharraz et al	96,3 %.
MLP, Bagging, Random Forest et J48	L'analyse dynamique	Sequential Pattern Mining	Homayoun et al	99 %
le moniteur de processus logiciel	L'analyse dynamique	Les registres , les fichiers et les regshots	Weckst'en et al	/
une méthode d'apprentissage en profondeur	L'analyse dynamique	fichier d'en-tête des paquets réseau	Tseng et al	/

un réseau contradictoire génératif(GAN)	L'analyse dynamique	les fichiers	Chen et al	/
Cinq méthodes d'apprentissage automatique (Arbre de décision, forêt aléatoire, K-Nearest Neighbor, Naive Bayes et Gradient boosting.)	L'analyse statique	des opcodes basés sur des fonctionnalités	Zhang et al	91,43 %
la technique d'apprentissage automatique représentée par le classificateur SVM	L'analyse statique	WEKA	Baldwin et Dehghantanha	96,5 %
CRSTATIC	L'analyse hybride	les familles de ransomwares	Subedi et al	/
L'algorithme Gradient Tree Boosting.	L'analyse hybride	74 échantillons de 12 familles Cryptographic Ransomware	Shaukat et Ribeiro	98,25 %

Honeypot	L'analyse hybride	Des rançongiciels Android	Ferrante et al	/
Un outil PayBreak	L'analyse hybride	les fichiers affectés après une attaque de ransomware	Kolodenker et al	/
Le système CryptoDrop	L'analyse hybride	fichier suspecte	Scaife et al	/
le système ShieldFS	L'analyse hybride	la mémoire de processus	Continella et al	/

Tableau 2.1 : Comparaison des méthodes de détection des malwares

On remarque dans le Tableau 2.1 que la méthode dynamique a une grande précision que les autres méthodes à cause de l'utilisation des algorithmes Deep learning (SVM, Random Forest …).

2.3. Conclusion

Dans ce chapitre, j'ai présenté l'état de l'art des méthodes de détection des Ransomwares. J'ai conclu qu'il existe plusieurs méthodes de détection et j'ai fait une comparaison entre ces dernières. Dans le chapitre suivant je vais présenter la méthode de détection proposée en utilisant Les algorithmes du « Deep Learning ».

Chapitre 3: Proposition d'une méthode de détection des Ransomwares

3.1 Introduction

Avec l'évolution de la technologie ,les cybercriminels essayent toujours de développer des nouveaux ransomwares (logiciels malveillants) qui représentent une réelle menace pour la sécurité des systèmes informatiques, ces derniers peuplent notre quotidien et interviennent dans toutes nos activités .

Malheureusement ,ils profitent chaque fois des bugs existant dans nos systèmes pour lancer des attaques avec diverses familles des logiciels malicieuses. Bref, la qualité et la diversité des ransomwares rendent les défenses inefficaces.

Par ailleurs, la détection de ces rançongiciel reste classique et incapable de protéger nos machines car elle utilise des anti virus commerciaux qui nécessite par fois des mise a jour en ligne. Dans ce chapitre, nous présentons notre problématique avec quelques travaux connexes. Par la suite nous ajoutons une analyse exploratoire des données, vient ensuite l'étape de préparation des données et enfin nous abordons notre proposition avec une architecture globale de notre approche

3.2 Le Deep Learning

Le Deep learning (DL) est un sous-ensemble des méthodologies et techniques de machine learning (ML) qui utilisent le réseau neuronal artificiel (ANN). C'est l'adaptation des réseaux neuronaux qui imite la structure du cerveau humain. La force de DL réside dans le fait que la machine peut extraire des caractéristiques et apprendre toute seule, indépendamment de l'intervention d'un expert. Il a été appliqué dans de nombreux domaines différents (traitement des images, textes, paroles et vidéos). Le succès de DL appartient à la disponibilité de plus de données d'entraînement. Google, Facebook et Amazon ont déjà commencé à l'utiliser pour faire l'analyse de leurs énormes quantités de données. DL est l'une des raisons qui ont conduit

les récents mouvements et progrès de l'IA, et c'est la principale cause qui fait penser que finalement, il existe une possibilité pour l'IA de devenir plus réaliste. Alors, qu'est-ce que DL ? Selon les fondateurs Yann LeCun, Yoshua Bengio Geoffrey Hinton dans [27] : "L'apprentissage profond permet aux modèles informatiques composés de plusieurs couches de traitement d'apprendre des représentations de données avec plusieurs niveaux d'abstraction." Autre définition par les auteurs dans [28] : "L'apprentissage profond est une classe de techniques d'apprentissage machine, où l'information est traitée en couches hiérarchiques pour comprendre les représentations et les caractéristiques des données dans des niveaux de complexité croissante."

3.2.1 Le concept du Deep Learning

Deep Learning ou apprentissage en profondeur ou DL est une branche du Machine Learning entièrement basée sur des réseaux de neurones artificiels [29]. Le concept d'apprentissage en profondeur existe depuis plusieurs années, mais il a été laissé à l'abandon faute de moyens nécessaires. Dans le milieu des années 2000 le Machine Learning fait rage dans les compétitions de reconnaissance visuelle, en 2012 deep mind une startup dans le domaine de l'IA arrive dans la compétition avec un algorithme de deep learning qui bat largement tous les autres compétiteurs, l'année suivante tous les compétiteurs se sont tournés vers le deep learning au vu des résultats obtenus. L'avancée du DL est dû à l'augmentation en exponentiel qu'ont connu les machines en capacités de calculs, et de stockages, ainsi que la disponibilité de données de masses (big data), ses 3 ingrédients étaient nécessaires pour exploiter le potentiel du DL qui fût chose impossible dans les années 90. Les pionniers qui ont soutenus le DL tel que Geoffrey Hinton, ou alors Yoshua Bengio qui a développé les réseaux GAN(generative adversal networks), Yann leCun qui est au coeur d'une avancée fulgurante dans le domaine de reconnaissance d'images avec les réseaux Convolutionnels CNN voir section (II.5), et son architecture LetNet. Geoffrey Hinton a prouvé que l'apprentissage profond pouvait résoudre des problèmes insolubles par d'autres approches [30].

3.2.2 Machine Learning vs Deep Learning:

La majeure différence qu'on note entre ces 2 concepts provient de la manière dont les données sont présentées au système (modèle).

● Les algorithmes de ML nécessitent presque toujours des données structurées, alors que les réseaux d'apprentissage approfondis reposent sur des couches de réseaux de neurones artificiels (RNA).

● On voit aussi une différence au sein de l'architecture des modèles qui les composent, on note que les modèles type DL sont plus profond que les modèles type ML. 15

● Deep learning n'utilise que les réseaux de neurones, alors que pour le ML les réseaux de neurones sont qu'une approche de conception des modèles parmis tant d'autres.

En considérant le fait que le DL est la prochaine étape de l'évolution du ML inculquant aux machines la manière de prendre leurs décisions de façon précise sans l'intervention de l'expert humain.

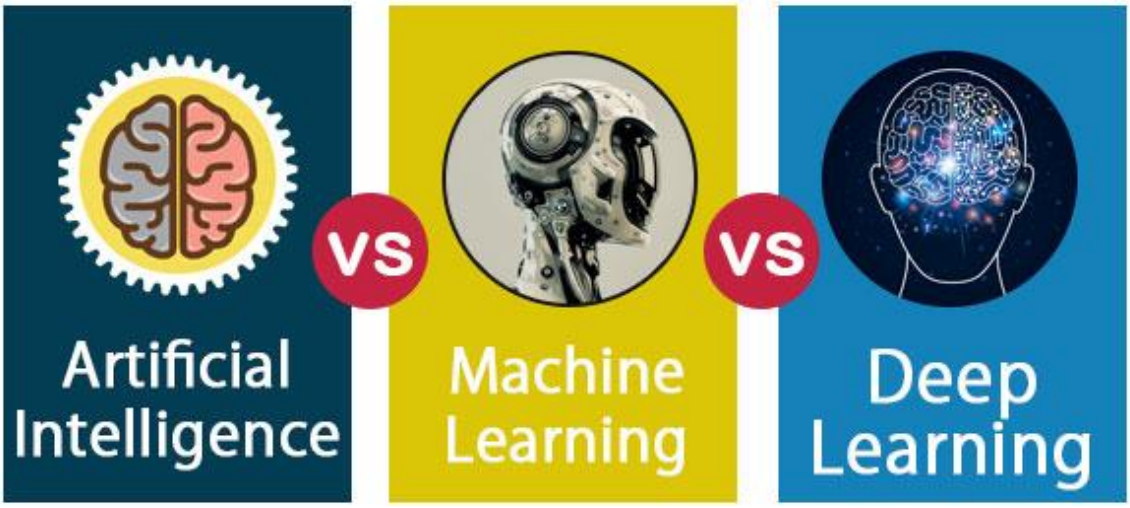


Figure 3.1 : IA vs ML vs DL

3.2.3 L'utilité du Deep learning:

Il s'agit d'une combinaison de facteurs dont :

● L'omniprésence des données: Nous somme dans l'ère de l'informatisation (Internet Of Things), et le propre du Deep-Learning est de tirer parti d'une grande quantité de données pour en estimer une représentation abstraite et en tirer parti.

● La puissance de calcul: La théorie des réseaux de neurones existe depuis quelques décennies, mais c'est grâce à la puissance de calcul accessible aujourd'hui qui se démocratise, notamment depuis que les GPUs sont devenus la plateforme de choix pour le Deep-Learning.

● Des besoins croissants dans le domaine de l'IA : vision par ordinateur, reconnaissance vocale, traitement du langage, …etc.

● Un effet de mode. On a tendance à vouloir appliquer le Deep-Learning partout alors que ça reste un moyen et non une fin. Certains problèmes sont tout à fait solubles par d'autres méthodes d'apprentissage statistique. Cela dit, si beaucoup de gens sont prêts à investir dans le Deep-Learning, il est normal qui deviennent si populaire.

● Capacité de Stockage: qui sont devenu beaucoup plus accessible à prix raisonnable

. ● Apparition de plateformes et communautés forte encouragent l'évolution de ce domaine, ainsi que sa démocratisation.

3.3 Les méthodes proposées pour la classification :

3.3.1 L'algorithme CNN :

Les réseaux de neurones convolutionnels (CNN pour Convolutional Neural Networks) proposés initialement par Le Cun [31]. Ce choix a été motivé principalement par ce qu'il intègre implicitement une phase d'extraction de caractéristiques et il a été utilisé avec succès dans de nombreuses applications [32]. Ils sont réputés pour leur robustesse aux faibles variations d'entrée et le faible taux de prétraitement nécessaire à leur fonctionnement. Le CNN est un réseau de neurone multicouche qui est spécialisé dans des tâches de

reconnaissance de forme [33]. Ces réseaux ont été inspirés par les travaux de Hubel et Wiesel sur le cortex visuel chez les mammifères [34] qui combine trois idées principales :

- Les champs récepteurs locaux,
- Les poids partagés et
- Le sous-échantillonnage.

L'architecture de CNN repose sur plusieurs réseaux de neurones profonds consistant en une succession de couches de convolution et d'agrégation (pooling) est dédié à l'extraction 35 automatique de caractéristiques, tandis que la seconde partie, composée de couches de neurones complètement connectées, est dédiée à la classification [32]. Chaque cellule des couches de convolution est connectée à un ensemble de cellules regroupées dans un voisinage rectangulaire sur la couche précédente. Les champs récepteurs locaux permettent d'extraire des caractéristiques basiques. Les couches sont dites « à convolution » car les poids sont partagés et chaque cellule de la couche réalise la même combinaison linéaire (avant d'appliquer la fonction sigmoïde) qui peut être vue comme une simple convolution. Ces caractéristiques sont alors combinées à la couche suivante afin de détecter des caractéristiques de plus haut niveau

Chaque cellule des couches de convolution est connectée à un ensemble de cellules regroupées dans un voisinage rectangulaire sur la couche précédente. Les champs récepteurs locaux permettent d'extraire des caractéristiques basiques. Les couches sont dites « à convolution » car les poids sont partagés et chaque cellule de la couche réalise la même combinaison linéaire (avant d'appliquer la fonction sigmoïde) qui peut être vue comme une simple convolution. Ces caractéristiques sont alors combinées à la couche suivante afin de détecter des caractéristiques de plus haut niveau.

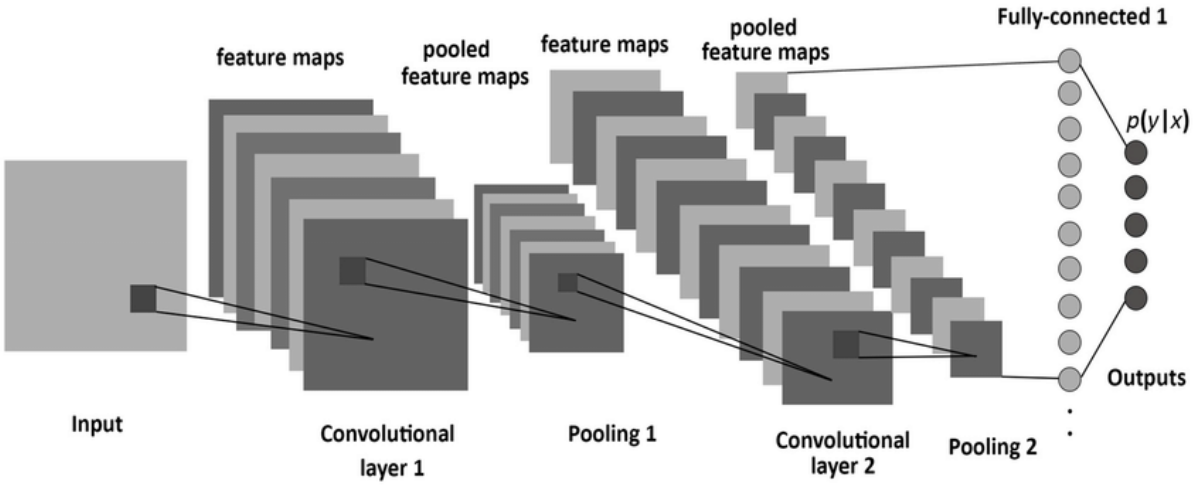


Figure 3.2 L'architecture d'un réseau de neurone conventionnel

Entre deux phases d'extraction de caractéristiques, le réseau réduit la résolution de la carte des caractéristiques par un moyen de sous-échantillonnage. Cette réduction se justifie à deux titres : diminuer la taille de la couche et apporter de la robustesse par rapport aux faibles distorsions. Couche de convolution : La convolution est une opération mathématique comme l'addition et la multiplication, il est très utile de simplifier des équations plus complexes, cette opération est largement utilisée dans le traitement du signal numérique. Lorsque l'on applique la convolution au traitement d'image, on réalise la convolution (combiner) l'image d'entrée avec un sous-régime de cette image (filtre). Le filtre est aussi connu sous le nom du noyau de convolution. La sortie de cette couche est l'image entrée avec des modifications qui est souvent appelée une carte de caractéristique (Feature Map).

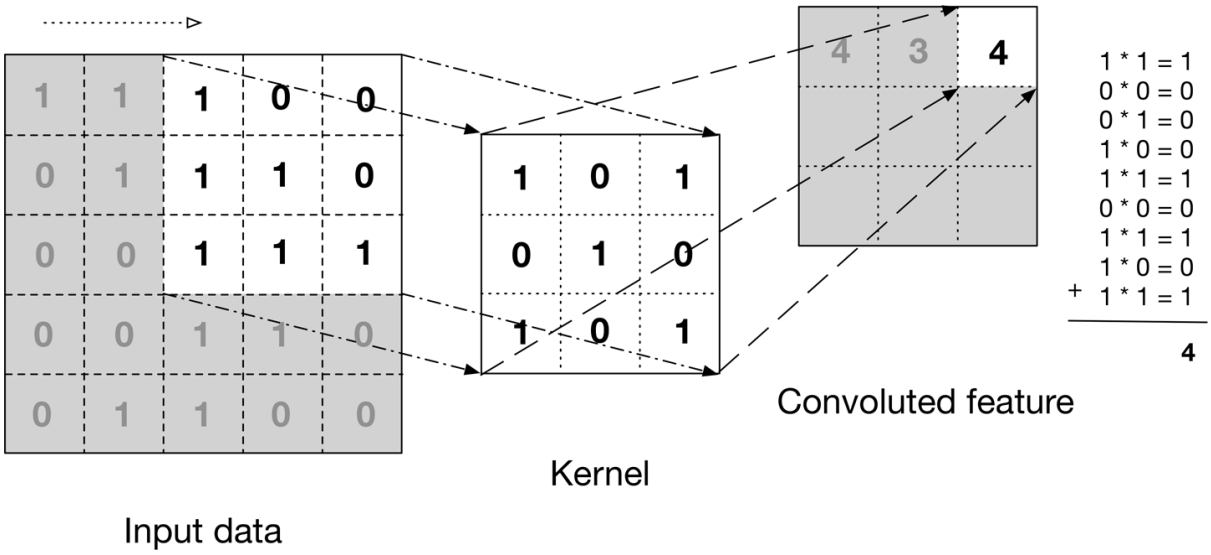


Figure 3.3: L'opération de convolution

En terme mathématique, une couche de convolution Ci (couche i du réseau) est paramétrée par son nombre N de cartes de convolution □□ □ (□ ∈ {□ … . □}), la taille des noyaux de convolution □□ (souvent carrée), et le schéma de connexion à la couche précédente □□ . Chaque carte de convolution est le résultat d'une somme de convolution des cartes de la couche précédente par son noyau de convolution respectif. Un biais est ensuite ajouté et le résultat passe à une fonction de transfert non-linéaire. Dans le cas d'une carte complètement connectée aux cartes de la couche précédente, le résultat est alors calculé par : Couche de sous-échantillonnage (Pooling) : Dans les architectures classiques de réseaux de neurones convolutionnels, les couches de convolution sont suivies par des couches de sous échantillonnage (couche d'agrégation). Cette dernière réduit la taille des cartes de caractéristique pour but de diminuer la taille de paramètre, et renvoie les valeurs maximales des régions rectangulaires de son entrée [33].

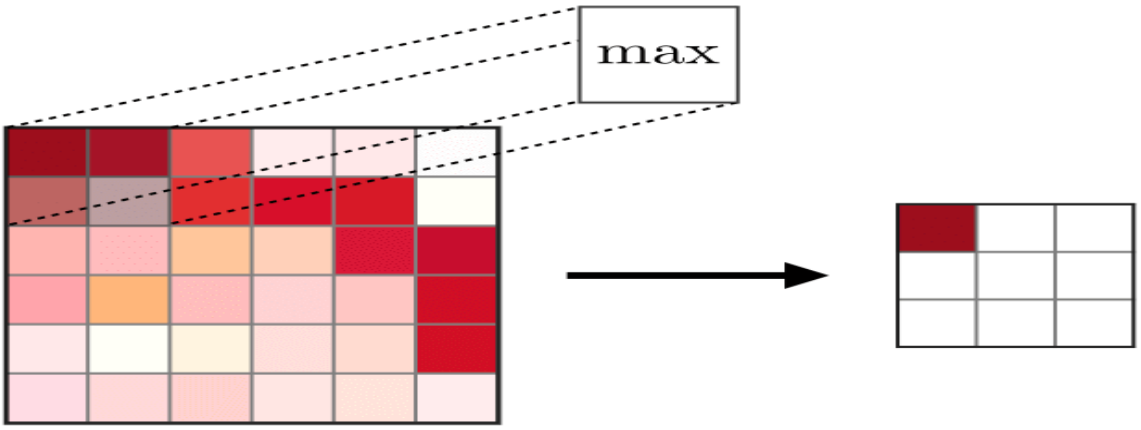


Figure 3. 4 L'opération de sous-échantillonnage (Pooling Layer)

Couche entièrement connectée : Les paramètres des couches de convolution et de max agrégation sont choisis de sorte que les cartes d'activation de la dernière couche soient de taille 1, ce qui résulte en un vecteur 1D d'attributs. Des couches classiques complètement connectées composées de neurones sont alors ajoutées au réseau pour réaliser la classification. La dernière couche, dans le cas d'un apprentissage supervisé, contient autant de neurones que de classes désirées. Cette dernière couche contient N neurones (nombre des classes dans la base), et une fonction d'activation de type sigmoïde est utilisée afin d'obtenir des probabilités d'appartenance à chaque classe.

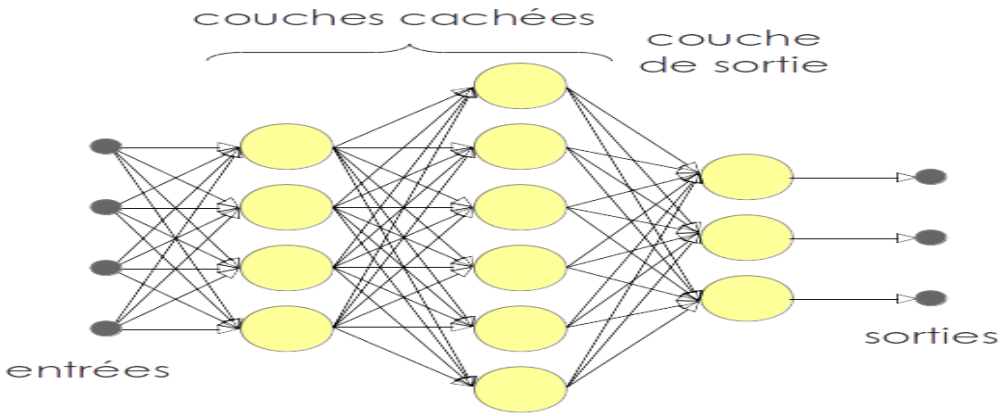


Figure 3. 5 Couche entièrement connectée (Fully connected)

3.3.2 L'algorithme LSTM :

La mémoire longue à court terme (LSTM) est un réseau de neurones artificiels utilisé dans les domaines de l'intelligence artificielle et de l'apprentissage en profondeur. Contrairement aux réseaux de neurones à anticipation standard, LSTM a des connexions de rétroaction. Un tel réseau neuronal récurrent (RNN) peut traiter non seulement des points de données uniques (tels que des images), mais également des séquences entières de données (telles que la parole ou la vidéo). Par exemple, LSTM est applicable à des tâches telles que la reconnaissance d'écriture manuscrite non segmentée et connectée, la reconnaissance vocale, la traduction automatique, le contrôle de robots, les jeux vidéo,et les soins de santé. LSTM est devenu le réseau neuronal le plus cité du 20ème siècle.

Le nom de LSTM fait référence à l'analogie selon laquelle un RNN standard possède à la fois une "mémoire à long terme" et une "mémoire à court terme". Les poids et les biais de connexion dans le réseau changent une fois par épisode d'entraînement, de manière analogue à la façon dont les changements physiologiques des forces synaptiques stockent les souvenirs à long terme ; les schémas d'activation dans le réseau changent une fois par pas de temps, de manière analogue à la façon dont le changement d'instant en instant des schémas de déclenchement électrique dans le cerveau stocke les souvenirs à court terme. L'architecture LSTM vise à fournir une mémoire à court terme pour RNN qui peut durer des milliers de pas de temps, donc une "mémoire longue à court terme".

Une unité LSTM commune est composée d'une cellule, d'une porte d'entrée, d'une porte de sortie et d'une porte d'oubli. La cellule se souvient des valeurs sur des intervalles de temps arbitraires et les trois portes régulent le flux d'informations entrant et sortant de la cellule.

Les réseaux LSTM sont bien adaptés à la classification, au traitement et à la réalisation de prédictions basées sur des données de séries chronologiques, car il peut y avoir des décalages de durée inconnue entre des événements importants dans une série chronologique. Les LSTM ont été développés pour traiter le problème du gradient de fuite qui peut être rencontré lors de la formation des RNN traditionnels. L'insensibilité relative à la longueur de l'écart est un avantage de LSTM par rapport aux RNN, aux modèles de Markov cachés et à d'autres méthodes d'apprentissage de séquences dans de nombreuses applications

Un réseau de mémoire à long court terme se compose de quatre portes différentes à des fins différentes, comme décrit ci-dessous : –

1. Forget Gate(f) : Il détermine dans quelle mesure oublier les données précédentes.

2. Porte d'entrée (i) : elle détermine l'étendue des informations à écrire sur l'état interne de la cellule.

3. Porte de modulation d'entrée (g) : elle est souvent considérée comme une sous-partie de la porte d'entrée et une grande partie de la littérature sur les LSTM ne la mentionne même pas et suppose qu'elle se trouve à l'intérieur de la porte d'entrée. Il est utilisé pour moduler les informations que la porte d'entrée écrira sur la cellule d'état interne en ajoutant de la non-linéarité aux informations et en rendant les informations zéro-moyennes . Ceci est fait pour réduire le temps d'apprentissage car l'entrée à moyenne nulle a une convergence plus rapide. Bien que les actions de cette porte soient moins importantes que les autres et soient souvent traitées comme un concept de finesse, il est recommandé d'inclure cette porte dans la structure de l'unité LSTM.

4. Porte de sortie (o) : elle détermine quelle sortie (prochain état caché) générer à partir de l'état de cellule interne actuelle.

Le flux de travail de base d'un réseau de mémoire à long court terme est similaire au flux de travail d'un réseau de neurones récurrent, la seule différence étant que l'état interne de la cellule est également transmis avec l'état caché.

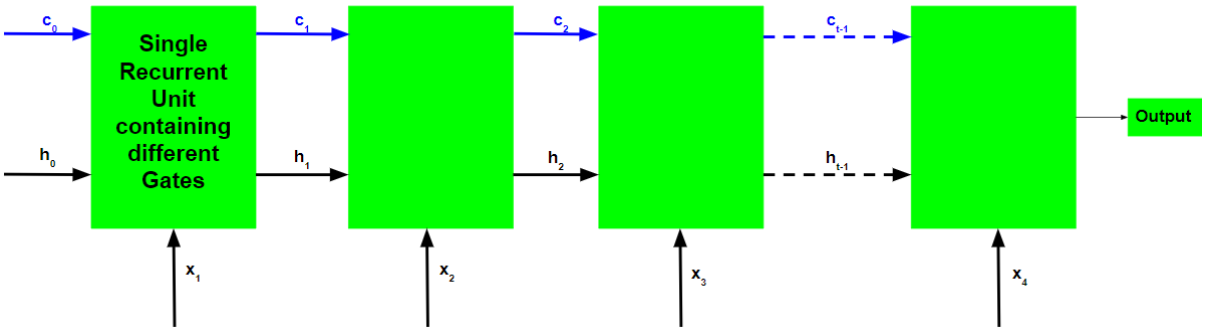


Figure 3. 6 Les Portes LSTM

3.3.2.1 Fonctionnement d'une unité récurrente LSTM :

Prenez en entrée l'entrée actuelle, l'état masqué précédent et l'état de cellule interne précédent.

Calculez les valeurs des quatre portes différentes en suivant les étapes ci-dessous :

Pour chaque porte, calculez les vecteurs paramétrés pour l'entrée actuelle et l'état caché précédent par multiplication élément par élément avec le vecteur concerné avec les poids respectifs pour chaque porte.

Appliquer la fonction d'activation respective pour chaque élément de porte sur les vecteurs paramétrés. Vous trouverez ci-dessous la liste des portes avec la fonction d'activation à appliquer pour la porte.

Calculez l'état actuel de la cellule interne en calculant d'abord le vecteur de multiplication par élément de la porte d'entrée et de la porte de modulation d'entrée, puis calculez le vecteur de multiplication par élément de la porte oubliée et de l'état de cellule interne précédent, puis en ajoutant les deux vecteurs.

$$c_t = i \odot g + f \odot c_{t-1}$$

Calculez l'état caché actuel en prenant d'abord la tangente hyperbolique par élément du vecteur d'état de cellule interne actuel, puis en effectuant une multiplication par élément avec la porte de sortie.

Le fonctionnement indiqué ci-dessus est illustré ci-dessous:

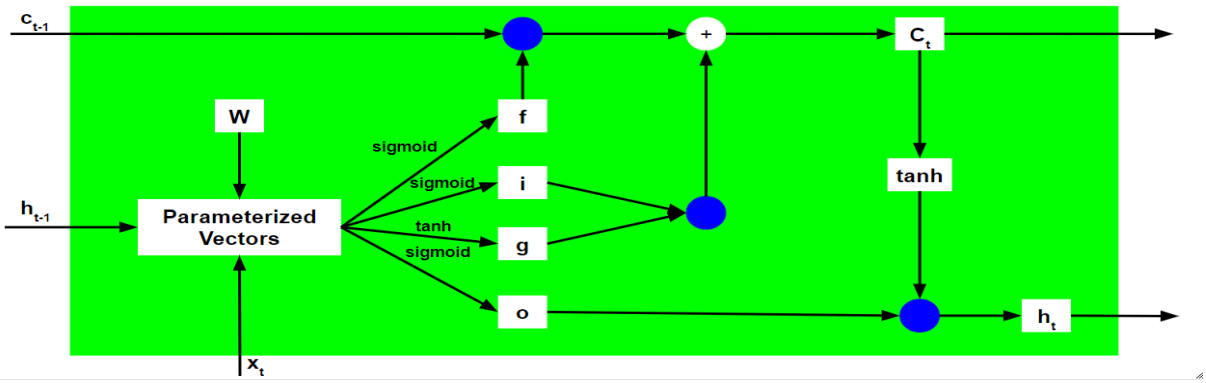


Figure 3. 7 le fonctionnement du LSTM

Notez que les cercles bleus indiquent une multiplication élément par élément. La matrice de poids W contient différents poids pour le vecteur d'entrée actuel et l'état caché précédent pour chaque porte.

Tout comme les réseaux de neurones récurrents, un réseau LSTM génère également une sortie à chaque pas de temps et cette sortie est utilisée pour entraîner le réseau à l'aide de la descente de gradient.

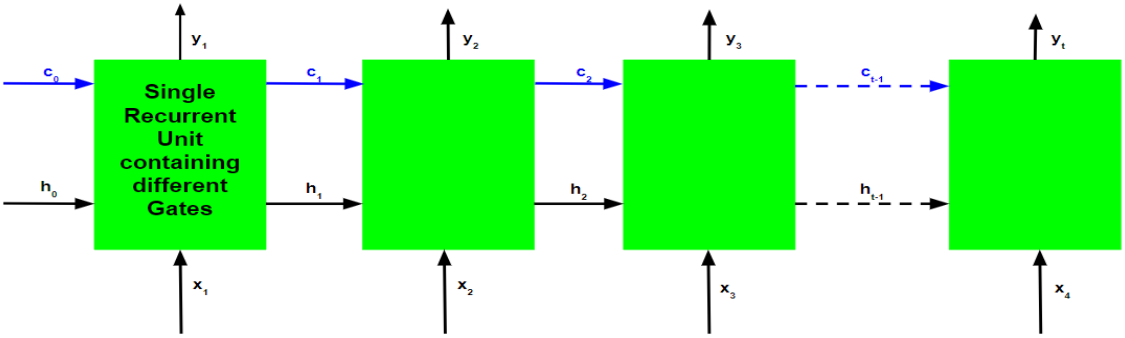


Figure 3. 8 fonctionnement LSTM "2"

La seule différence principale entre les algorithmes de rétro-propagation des réseaux de neurones récurrents et des réseaux de mémoire à court terme est liée aux mathématiques de l'algorithme.

Soit $\overline{y}_t$ la sortie prédite à chaque pas de temps et y_t soit la sortie réelle à chaque pas de temps. Alors l'erreur à chaque pas de temps est donnée par :-

$$E_t = -y_t log(\overline{y}_t)$$

L'erreur totale est donc donnée par la somme des erreurs à tous les pas de temps.

$$E = \sum_t E_t$$

$$\Rightarrow E = \sum_t -y_t log(\overline{y}_t)$$

De même, la valeur $\frac{\partial E}{\partial W}$ peut être calculée comme la somme des gradients à chaque pas de temps.

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

En utilisant la règle de la string et en utilisant le fait qui $\overline{y}_t$ est une fonction de h_t et qui est en effet une fonction de c_t, l'expression suivante apparaît : –

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \overline{y}_t} \frac{\partial \overline{y}_t}{\partial h_t} \frac{\partial h_t}{\partial c_t} \frac{\partial c_t}{\partial c_{t-1}} \frac{\partial c_{t-1}}{\partial c_{t-2}} \cdots\cdots \frac{\partial c_0}{\partial W}$$

Notez que l'équation de gradient implique une string de ∂c_t pour une rétropropagation LSTM tandis que l'équation de gradient implique une string de ∂h_t pour un réseau neuronal récurrent de base.

Comment LSTM résout-il le problème des gradients qui disparaissent et explosent ?

l'expression de c_t.

$$c_t = i \odot g + f \odot c_{t-1}$$

La valeur des gradients est contrôlée par la string de dérivées à partir de $\frac{\partial c_t}{\partial c_{t-1}}$. Développer cette valeur en utilisant l'expression pour c_t:-

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial c_t}{\partial f} \frac{\partial f}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial i} \frac{\partial i}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial g} \frac{\partial g}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial c_{t-1}} + \frac{\partial c_t}{\partial c_{t-1}}$$

Pour un RNN de base, le terme $\frac{\partial c_t}{\partial c_{t-1}}$ après un certain temps commence à prendre des valeurs soit supérieures à 1 soit inférieures à 1 mais toujours dans la même plage. C'est la cause première du problème des gradients qui disparaissent et explosent. Dans un LSTM, le terme $\frac{\partial c_t}{\partial c_{t-1}}$ n'a pas de modèle fixe et peut prendre n'importe quelle valeur positive à n'importe quel pas de temps. Ainsi, il n'est garanti que pour un nombre infini de pas de temps,

le terme convergera vers 0 ou divergera complètement. Si le gradient commence à converger vers zéro, alors les poids des portes peuvent être ajustés en conséquence pour le rapprocher de 1. Puisque lors de la phase d'apprentissage, le réseau n'ajuste que ces poids, il apprend donc quand laisser le gradient converger vers zéro et quand le conserver.

3.3.3 L'algorithme DNN :

Les réseaux de neurones profonds (DNN) sont des versions améliorées de l'ANN conventionnel avec plusieurs couches. Les modèles DNN sont récemment devenus très populaires en raison de leurs excellentes performances pour apprendre non seulement le mappage entrée-sortie non linéaire, mais également la structure sous-jacente des vecteurs de données d'entrée.

Lors de la mise en œuvre de la formation DNN dans la génération à court terme ou la prévision de charge, un ensemble d'échantillons de formation comprenant un vecteur d'entrée composé de la météo, du calendrier ou d'autres variables et un vecteur cible, qui est la puissance générée (pour la prévision de production) ou la charge puissance (pour la prévision de charge), doivent être collectées et un problème de régression pour l'optimisation sera formulé. Les paramètres DNN sont par conséquent estimés en minimisant la fonction d'erreur de somme des carrés calculée à partir des sorties DNN. À partir d'une étape d'initialisation où les paramètres du modèle sont définis sur un ensemble initial de valeurs, un algorithme de descente de gradient stochastique est exécuté en continu pour réduire la fonction d'erreur jusqu'à ce qu'elle converge vers une valeur la plus basse spécifiée . La formation DNN implique deux passes basées sur l'algorithme de rétropropagation des erreurs, qui sont la passe avant et la passe arrière. Dans le premier, la transformation affine et l'activation non linéaire sont calculées couche par couche de la couche d'entrée à la couche de sortie. Dans la dernière, les dérivées de la fonction d'erreur par rapport aux poids individuels sont calculées dans l'ordre inverse, c'est-à-dire de la couche de sortie à la couche d'entrée.

Les DNNs (Deep Neural Network) effectuent la propagation vers l'avant de l'information (Feed-forward). Ils comprennent une couche d'entrée, plusieurs couches cachées et la couche de sortie (figure 3.9). Les couches cachées et la couche de sortie se composent de nœuds où la sortie d'une couche est une entrée de la couche suivante. Les nœuds effectuent une fonction d'activation linéaire suivie par une fonction d'activation non linéaire sur la valeur d'entrée

(figure 3.10). Dans la figure 3.10, la valeur d'entrée de nœud □ (□) = ((□) . □ (□−1) + □ (□)) où □(□) . □ (□−1) + □ (□) est une fonction d'activation linéaire et ((□) . □ (□−1) + □ (□)) une fonction d'activation non linéaire.

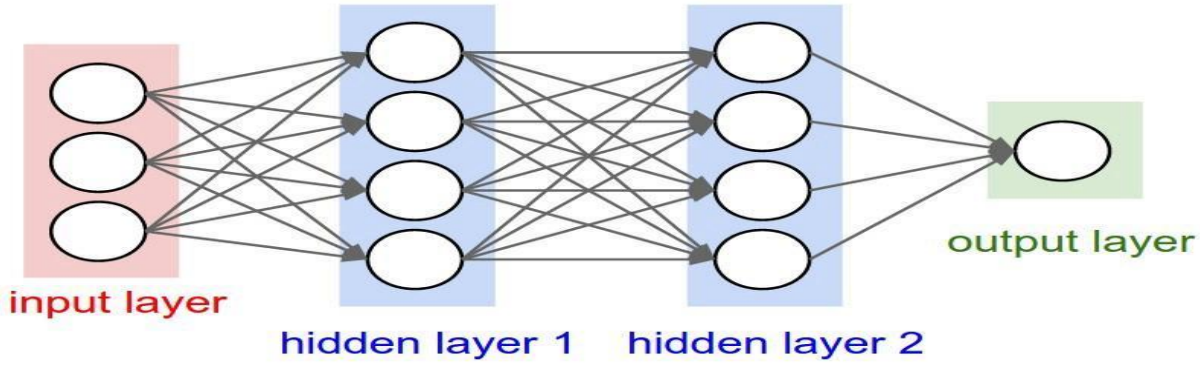


Figure 3.9 : l'architecture des DNNs

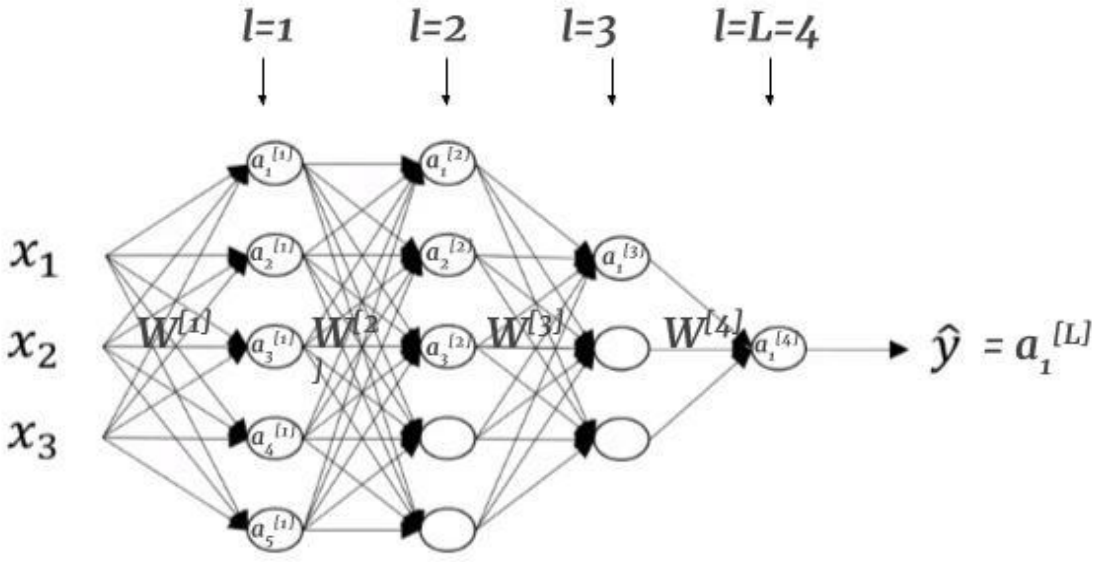


Figure 3.10 : l'activation des DNNs

3.3.4 L'algorithme Convo-LSTM :

Ce projet propose une nouvelle combinaison de CNN avec LSTM de sorte que la capacité d'extraction de caractéristiques de CNN puisse bénéficier de la capacité de cartographie de récurrence de séquence des réseaux de neurones récurrents (RNN). Une couche d'entrée, une couche de convolution unidimensionnelle, une couche de regroupement, une couche cachée LSTM et une couche de connexion complète sont les principaux composants qui construisent la structure principale de Convo-LSTM. Lecun et al. ont proposé le modèle de réseau CNN en 1998. L'opération de convolution extrait les attributs des vecteurs de la couche d'entrée. Dans ce cas, des opérations exclusivement unidimensionnelles - 1 D ne sont effectuées qu'en raison de la structure des données.

Les poolinglayers sont déployés pour réduire les besoins de stockage et éviter les énormes coûts de formation dans le système. La couche de mise en commun subdivise les petits morceaux rectangulaires de la couche convolutive pour générer une sortie unique à partir de chaque bloc. La mise en commun peut se faire de différentes manières, par exemple en calculant la moyenne ou le maximum. La mise en commun moyenne prend la valeur moyenne du bloc qu'elle regroupe, tandis que la mise en commun maximale prend le maximum du bloc qu'elle regroupe. Tout d'abord, la couche CNN extrait les caractéristiques des données, qui sont les relevés des relevés du capteur de courant, de tension, de température et de vibration collectés au cours de l'année précédente. Le LSTM est ensuite utilisé pour prévoir la sortie, Vibration, sur la base des données de caractéristiques récupérées. Selon les résultats de l'expérience, avec la précision de prédiction maximale, le CNN-LSTM qui est Convo-LSTM peut fournir une prévision crédible du paramètre de sortie (Vibration).

Complexité temporelle de Convo-LSTM :

Pour déterminer la complexité temporelle des processus de propagation vers l'avant et de rétropropagation, le nombre total d'opérations au niveau de chaque couche CNN 1D doit d'abord être déterminé, puis le nombre total d'opérations doit être agrégé pour déterminer la complexité temporelle globale. Pendant la propagation vers l'avant P, le nombre de connexions à la couche précédente au niveau d'une couche CNN, l, est Nl−1Nlle nombre de connexions de la couche précédente est Nl−1Nl, une convolution linéaire individuelle, qui est une somme linéaire pondérée, est évaluée.

Soit Sl−1 et Wl−1 représentent les tailles vectorielles de la sortie de la couche précédente, Sl−1k et le noyau (poids), respectivement. Une convolution linéaire est constituée de (Sl−1Wl−1)2multiplications et Sl−1additions à partir d'une même connexion, sans tenir compte des conditions aux limites. Si le biais est ignoré, le nombre total de multiplications et d'additions dans la couche l sera :

$$N(mul)l=Nl-1Nl*Sl-1*(Wl-1)2$$

$$N(addition)l=Nl-1Nl*Sl-1$$

Une faible complexité de calcul est atteinte dans tous les CNN 1D. Ainsi, en propagation aller, le nombre total de multiplications T(mul) et le nombre total d'additions T(add), dans la couche CNN l seront :

$$\textbf{TF P (mul)} =LXl=0Nl-1Nl*Sl-1*(Wl-1)2$$

$$TF\ P\ (addition) =LXl=0Nl-1Nl*Sl-1$$

Maintenant, de même, à l'itération de rétropropagation, le nombre total de multiplications et d'additions dues à la première convolution sera donc :

$$TB\ P\ (mul) =LXl=0Nl+1Nl*Sl+\ 1\ *(Wl+1\ )2$$

$$TB\ P\ (additionner) =LXl=0Nl+1Nl*Sl+\ 1$$

Ainsi à chaque itération BP, le nombre total de multiplications et d'additions sera respectivement de :

TF P (mul) + TF P (ajouter) + TB P (mul)TB P (ajouter)

3.4 .Conclusion

Dans ce chapitre, j'ai présenté la méthode de détection des Ransomwares que j'ai proposée en utilisant le « Deep Learning » . Ce dernier adopte quatre modèles : le modèle CNN, LSTM, DNN et CONVO-LSTM . J'ai utilisé quelques méthodes pour améliorer les résultats. L'implémentation et la validation de l'efficacité de ma proposition font l'objectif du chapitre suivant

Chapitre 4: Implémentation et évaluation

4.1 Introduction

Après avoir décrit notre approche dans le chapitre précédent, dans cette partie nous allons parlé d'une manière générale sur les différents moyens techniques (langages, bibliothèques et environnements) utilisés pour implémenter notre solution. Ensuite nous présentons également une architecture détaillée de notre implémentation. Enfin, nous allons discuter des résultats obtenus.

4.2 Environnement de développement:

4.2.1 Python:

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom Python vient d'un hommage à la série télévisée Monty Python's Flying Circus dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991.

La dernière version de Python est la version 3. Plus précisément, la version 3.7 a été publiée en juin 2018. La version 2 de Python est désormais obsolète et cessera d'être maintenue après le 1er janvier 2020. Dans la mesure du possible évitez de l'utiliser

La Python Software Foundation 1 est l'association qui organise le développement de Python et anime la communauté de développeurs et d'utilisateurs. Ce langage de programmation présente de nombreuses caractéristiques intéressantes.

1. Il est multiplateforme. C'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS X, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.

2. Il est gratuit. Vous pouvez l'installer sur autant d'ordinateurs que vous voulez (même sur votre téléphone !).

3. C’est un langage de haut niveau. Il demande relativement peu de connaissances sur le fonctionnement d’un ordinateur pour être utilisé.

4. C’est un langage interprété. Un script Python n’a pas besoin d’ être compilé pour être exécuté, contrairement à des langages comme le C ou le C++.

5. Il est orienté objet. C'est-`à-dire qu’il est possible de concevoir en Python des entités qui miment celles du monde réel (une cellule, une protéine, un atome, etc.) avec un certain nombre de règles de fonctionnement et d’interactions.

6. Enfin, il est très utilisé en bioinformatique et plus généralement en analyse de données. Toutes ces caractéristiques font que Python est désormais enseigné dans de nombreuses formations, depuis l’enseignement secondaire jusqu'à l’enseignement supérieur [29].

Figure 4.1 : Python

4.2.2 Anaconda:

Anaconda est un outil dont la distribution est libre et open source. Il est destiné à la programmation dans un environnement Python et R. Anaconda est largement utilisé en sciences de données, en intelligence artificielle ou Machine Learning. Cette distribution scientifique de Python renferme de nombreux packages nécessaires à l'analyse de données. Anaconda est également un gestionnaire d'environnement open source.

Cet outil qui recense plus de 20 millions d'utilisateurs dans le monde comprend entre autres :

- Uune installation de l'environnement Python,
- Des IDE (environnement de développement intégré) de dernière génération à l'instar de Jupyter ou de Spyder,

- Des packages de Data Science comme Panda, Numpy, Scikit-Learn…

L'outil Conda pour la gestion des environnements et des répertoires de package.

Anaconda assure grâce à ses différents outils et à l'environnement Python, une collecte et une transformation à grande échelle des données. Jupyter, l'un des IDE présents dans Anaconda, prend en charge désormais plus de 40 langages de programmation [29].

Figure 4.2 : Anaconda

4.2.3 Jupyter Notebook:

Jupyter Notebook est une application client-serveur créée par l'organisation à but non lucratif Project Jupyter. Elle a été publiée en 2015. Elle permet la création et le partage de documents Webauformat JSON constitués d'une liste ordonnée de cellules d'entrées et de sorties et organisés en fonction des versions successives du document. Les cellules peuvent contenir, entre autres, du code, du texte au format Markdown, des formules mathématiques ou des contenus médias (Rich Media). Le traitement se fait avec une application client fonctionnant par Internet, à laquelle on accède par les navigateurs habituels. Il est nécessaire pour cela que soit installé et activé dans le système le serveur Jupyter Notebook. Les documents Jupyter créés peuvent s'exporter aux formats HTML, PDF, Markdown ou Python par exemple, ou bien se partager par email, avec Dropbox, GitHub ou un lecteur Jupyter Notebook.

Les deux éléments principaux de Jupyter Notebook sont un jeu de différents noyaux (interpréteurs) et le tableau de bord (dashboard). Les noyaux sont des petits programmes qui traitent des requêtes dans un langage particulier et qui réagissent avec les réponses correspondantes. Le noyau standard est le Python, un interpréteur de lignes de commande, qui permet de travailler avec Python. On trouve en plus une bonne cinquantaine d'autres noyaux qui permettent l'utilisation d'autres langages, comme C++, R, Julia, Ruby, JavaScript, CoffeeScript, PHP ou Java. Le tableau de bord sert d'une part l'interface de gestion des différents noyaux, et d'autre part de centrale pour la création de nouveaux documents Notebook, ou pour ouvrir des documents existants. Jupyter Notebook est disponible sous licence BSD modifiée et donc librement utilisable par tous [30].

Figure 4.3 : Jupyter Notebook

4.2.4 Cuckoo Sandbox:

Un Cuckoo Sandbox est un outil utilisé pour lancer des logiciels malveillants dans un environnement sécurisé et isolé. L'idée est que le bac à sable trompe le logiciel malveillant en lui faisant croire qu'il a infecté un hôte authentique.

Le bac à sable enregistrera ensuite l'activité du logiciel malveillant, puis générera un rapport sur ce que le logiciel malveillant a tenté de faire dans cet environnement sécurisé.

Ceux-ci sont parfaits pour les équipes de sécurité et les analystes de logiciels malveillants car ils peuvent être utilisés pour rassembler rapidement les IOC qui peuvent être nécessaires pour un incident de sécurité ou un point de départ pour un élément d'information, cela vous donne

des informations rapides et détaillées sur la façon dont le logiciel malveillant est susceptible de se comporter.

La plupart des bacs à sable de logiciels malveillants commerciaux sont chers, comme la version d'entreprise de McAfee appelée Artemis. Cependant, Cuckoo est open source et téléchargeable gratuitement, et d'après mon expérience, le résultat obtenu est presque identique.

Même si Cuckoo est téléchargeable gratuitement, il peut être assez compliqué et long à configurer pour la première fois, cela est dû au fait que Cuckoo nécessite un certain nombre de dépendances, mais une fois en place, c'est un outil incroyablement utile.

Une fois configuré, Cuckoo est capable d'analyser de nombreux fichiers malveillants différents (exécutables, documents bureautiques, fichiers pdf, e-mails, scripts malveillants) ainsi que des sites Web malveillants.

4.3 Le dataset utilisé :

Le dataset utilisé dans notre projet est CICAndMal2017. Ce dadset collecte plus de 10 854 échantillons (4 354 logiciels malveillants et 6 500 bénins) provenant de plusieurs sources et plus de six mille applications bénignes du marché Googleplay publiées en 2015, 2016, 2017.

Il contient 5 000 des échantillons collectés (426 logiciels malveillants et 5 065 bénins) sur des appareils réels. Les données CICAndMal2017 sont classées en quatre catégories :

-Adware

-Ransomware

-Scareware

-SMS Malware

Les échantillons proviennent de 42 familles de logiciels malveillants uniques. Les types de famille de chaque catégorie et le nombre d'échantillons capturés sont les suivants :

4.3.1 Adware

-Famille Dowgin, 10 échantillons capturés

-Famille Ewind, 10 échantillons capturés

-Famille Feiwo, 15 échantillons capturés

-Famille Gooligan, 14 échantillons capturés

-Famille Kemoge, 11 échantillons capturés

-famille koodous, 10 échantillons capturés

-Famille Mobidash, 10 échantillons capturés

-Famille Selfmite, 4 échantillons capturés

-Famille Shuanet, 10 échantillons capturés

-Famille Youmi, 10 échantillons capturés

4.3.2 Ransomwares

-Famille de chargeurs, 10 échantillons capturés

-Famille Jisut, 10 échantillons capturés

-Famille Koler, 10 échantillons capturés

-Famille LockerPin, 10 échantillons capturés

-Famille Simplocker, 10 échantillons capturés

-Famille Pletor, 10 échantillons capturés

-Famille PornDroid, 10 échantillons capturés

-Famille RansomBO, 10 échantillons capturés

-Famille Svpeng, 11 échantillons capturés

-Famille WannaLocker, 10 échantillons capturés

4.3.3 Scareware

-AndroidDefender 17 échantillons capturés

-Famille AndroidSpy.277, 6 échantillons capturés

-AV pour la famille Android, 10 échantillons capturés

-Famille AVpass, 10 échantillons capturés

-Famille FakeApp, 10 échantillons capturés

-Famille FakeApp.AL, 11 échantillons capturés

-Famille FakeAV, 10 échantillons capturés

-Famille FakeJobOffer, 9 échantillons capturés

-Famille FakeTaoBao, 9 échantillons capturés

-Famille Penetho, 10 échantillons capturés

-Famille VirusShield, 10 échantillons capturés

4.3.4 SMS Malware

-Famille BeanBot, 9 échantillons capturés

-Famille Biige, 11 échantillons capturés

-Famille FakeInst, 10 échantillons capturés

-Famille FakeMart, 10 échantillons capturés

-Famille FakeNotify, 10 échantillons capturés

-Famille Jifake, 10 échantillons capturés

-Famille Mazarbot, 9 échantillons capturés

-Famille Nandrobox, 11 échantillons capturés

-Famille de plancton, 10 échantillons capturés

-Famille SMSsniffer, 9 échantillons capturés

-Famille Zsone, 10 échantillons capturés

Afin d'acquérir une vue complète de nos échantillons de logiciels malveillants, nous avons créé un scénario spécifique pour chaque catégorie de logiciels malveillants. Nous avons également défini trois états de capture de données afin de surmonter la furtivité d'un logiciel malveillant avancé :

Installation : Le premier état de capture de données qui se produit immédiatement après l'installation du logiciel malveillant (1-3 min).

Avant le redémarrage : le deuxième état de capture de données qui se produit 15 minutes avant le redémarrage des téléphones.

Après le redémarrage : le dernier état de capture de données qui se produit 15 minutes après le redémarrage des téléphones.

Pour l'extraction et la sélection de fonctionnalités, nous avons capturé les fonctionnalités de trafic réseau (fichiers .pcap) et extrait plus de 80 fonctionnalités à l'aide de CICFlowMeter-V3 pendant les trois états mentionnés (installation, avant redémarrage et après redémarrage).

4.4. Les bibliothèques utilisées:

4.4.1. TensorFlow :

Est un Framework de deep learning de bout en bout, open-source, développé par Google et publié en 2015. Il est connu pour sa documentation et son support de formation, ses options de production et de déploiement évolutives, ses multiples niveaux d'abstraction et sa prise en charge de différentes plateformes, telles qu'Android. TensorFlow est une bibliothèque mathématique symbolique utilisée pour les réseaux de neurones et est la mieux adaptée à la programmation de flux de données pour toute une série de tâches. Elle offre de multiples niveaux d'abstraction pour la construction et l'entraînement de modèles. TensorFlow est une entrée prometteuse et en pleine croissance dans le monde du deep learning. Elle offre un écosystème flexible et complet de ressources communautaires, de bibliothèques et d'outils qui facilitent la création et le déploiement d'applications de deep learning.

4.4.2 Keras :

Est une interface de programmation d'application (API) de réseau de neurones de haut niveau efficace, écrite en Python. Cette bibliothèque de réseau de neurones à code source ouvert est conçue pour permettre une expérimentation rapide des réseaux de neurones profonds et peut fonctionner en utilisant TensorFlow comme système de gestion. Keras est une API modulaire, conviviale et extensible. Il ne traite pas les calculs de bas niveau, mais les transmet à une autre bibliothèque appelée Backend. Keras a été adopté et intégré dans TensorFlow à la mi-2017. Les utilisateurs peuvent y accéder via le module tf.keras. Toutefois, la bibliothèque Keras peut toujours fonctionner séparément et indépendamment.

4.5 Implémentation de l'architecture proposée :

Notre application est constituée de deux phases: prétraitement des données et apprentissage, nous présentons dans cette section chaque module avec les détails de sa réalisation :

4.5.1 Phase de prétraitement:

Comme nous l'avons déjà expliqué dans le chapitre 3, le traitement de la base passe par plusieurs étapes par les algorithmes du Deep Learning. Nous expliquerons ces étapes dans ce qui suit **:**

-La préparation de l'environnement de l'exécution Anaconda.

-L'importation des bibliothèques nécessaires Numpay, Pandas, Matplotlib, Sklearn.

-L'importation et la préparation de la base de données CICAndMal2017. Nous avons supprimé les valeurs nulle dans la base.

-Concaténer tous les fichiers de chaque classe dans un seul fichier csv nommé "Full_Data".

-Remplacement la dernière colonne de chaque fichier csv par (0=fichier normal(bening) ,1=fichier malware, 2=fichier ransomware).

-Fractionner les données en apprentissage et test (0.8-0.2), puis refaire les opérations par la fonction model.fit(X_train, y_train, epochs=NB) pour meilleur apprentissage .

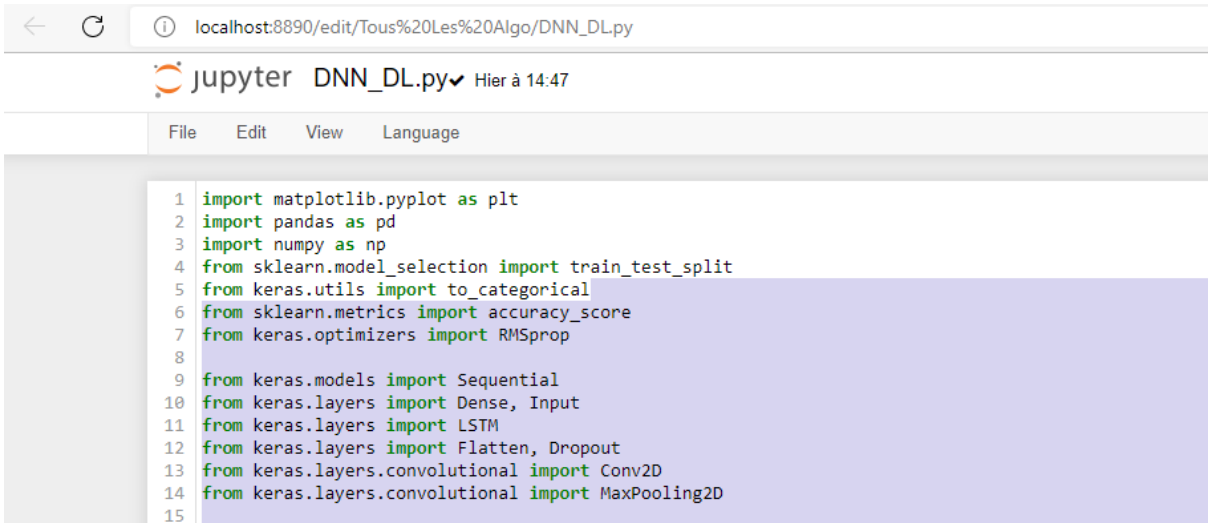
-Appliquer le modèle (CNN,CNN,LSTM,CONVO-LSTM).

4.5.2 Phase d'apprentissage :

maitenent je présente les détails du modèle proposé qui applique les 4 algorithmes du deep learning (CNN,CNN,LSTM,CONVO-LSTM). j'ai suivi les étapes suivantes pour l'apprentissage du modèle :

-La préparation de l'environnement de l'exécution Jupyter .

- L'importation des bibliothèques nécessaires Tensorflow, Keras, Numpy, Matplotlib,Sklearn, et j'ai importé les couches principales du réseau :



-L'importation des bases des données test_data.csv avec la fonction read_csv().

- Construire le modèle comme nous avons expliqué dans le chapitre précédent. -

-L'évaluation de modèle.

- Sauvegarder le modèle obtenu.

-Pour avoir les meilleurs paramètres qui conduisent aux bons résultats nous avons relancé l'apprentissage plusieurs fois. Le paramètre principale a fixé pour l'apprentissage est : epoch

- Epoch : est le nombre total d'itérations d'apprentissage, il est définit comme un critère d'arrêt que ce soit les résultats. Nous l'avons fixé à 100 epochs.

```
# model.add(Dropout(0.25))
model.add(Dense(3, activation='softmax'))
model.summary()

optimizer = RMSprop(learning_rate=0.0001)
optimizer = "adam"
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

model.fit(X_train, y_train, epochs=100)  #, verbose=0
```

4.5.3 Les résultats de chaque modèle :

DNN :

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from sklearn.metrics import accuracy_score
from keras.optimizers import RMSprop

from keras.models import Sequential
from keras.layers import Dense, Input
from keras.layers import LSTM
from keras.layers import Flatten, Dropout
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D

import glob
import pandas as pd

# Get data file names
path = r'test data/'


# read folder of normal data
filenames = glob.glob(path + "normal/*.csv")
dfs = []
for filename in filenames:
    dfs.append(pd.read_csv(filename))

# Concatenate all data into one DataFrame
normal_data = pd.concat(dfs, ignore_index=True)
```

```
normal_data=normal_data.drop(['Flow ID', ' Source IP',' Destination IP',' Timestamp'], axis=1)

normal_data=normal_data.dropna(how='any')
# changer la classe (normal)  with code  0
normal_data[" Label"]=0



# read folder of malwares data
filenames = glob.glob(path + "/malwares/*.csv")
dfs = []
for filename in filenames:
    dfs.append(pd.read_csv(filename))

# Concatenate all data into one DataFrame
malwares_data = pd.concat(dfs, ignore_index=True)

malwares_data=malwares_data.drop(['Flow ID', ' Source IP',' Destination IP',' Timestamp'], axis=1)

malwares_data=malwares_data.dropna(how='any')
# changer la classe (malwares)  with code  1
malwares_data[" Label"]=1


# read folder of ransomwares data
filenames = glob.glob(path + "/ransomwares/*.csv")
dfs = []
for filename in filenames:
    dfs.append(pd.read_csv(filename))

# Concatenate all data into one DataFrame
ransomwares_data = pd.concat(dfs, ignore_index=True)
```

```
ransomwares_data=ransomwares_data.drop(['Flow ID', ' Source IP',' Destination IP',' Timestamp'], axis=1)

ransomwares_data=ransomwares_data.dropna(how='any')
# changer la classe (ransomwares)  with code  2
ransomwares_data[" Label"]=2

#%%

Full_Data= pd.concat([normal_data,malwares_data,ransomwares_data], ignore_index=True)
Full_Data['empty'] = 0

Full_Data.to_csv("Full_Data.csv")

# Creer les données de train et de test
X = Full_Data.drop(" Label", axis=1)
Y = Full_Data[" Label"]
X = np.array(X)
Y = np.array(Y)

Y = to_categorical(Y, dtype ="uint8")

X = np.reshape(X,(len(X),9*9))
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =0.2)


input_shape = (9*9,)
model = Sequential()

model.add(Dense(300, activation='relu',input_shape=input_shape))
model.add(Dense(500, activation='relu'))
model.add(Dense(300, activation='relu'))
model.add(Dense(128, activation='relu'))
```

```
model.add(Dense(300, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
# model.add(Dropout(0.25))
model.add(Dense(3, activation='softmax'))
model.summary()

optimizer = RMSprop(learning_rate=0.0001)
optimizer = "adam"
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

model.fit(X_train, y_train, epochs=100)  #, verbose=0

y_pred = model.predict(X_test)


#%%
y_test2 = np.argmax(y_test,axis=1)
y_pred2 = np.argmax(y_pred,axis=1)



# Afficher la Matrice de confusion
# from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix(decisionTree, X_test, y_test)
# plt.show()


# Afficher le score de precision

print(accuracy_score(y_test2, y_pred2))
```

Résultat: 84.22 %

CNN:

```
X = np.reshape(X,(len(X),9,9,1))
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =0.2)


input_shape = (9, 9, 1)
model = Sequential()

model.add(Conv2D(64, kernel_size=(3, 3), padding='same',  activation='relu', input_shape=input_shape))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(3, activation='softmax'))
model.summary()

optimizer = RMSprop(learning_rate=0.0001)
optimizer = "adam"
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

model.fit(X_train, y_train, epochs=20)  #, verbose=0

y_pred = model.predict(X_test)


#%%
y_test2 = np.argmax(y_test,axis=1)
y_pred2 = np.argmax(y_pred,axis=1)



# Afficher la Matrice de confusion
# from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix(decisionTree, X_test, y_test)
# plt.show()


# Afficher le score de precision

print(accuracy_score(y_test2, y_pred2))
```

Résultat: 85.37 %

LSTM:

```
X = np.reshape(X,(len(X),1,9*9))
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =0.2)


#%%

model = Sequential()

model.add(LSTM(64, input_shape=(None, 81)))
model.add(BatchNormalization())

model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(3, activation='softmax'))
model.summary()
#%%
optimizer = RMSprop(learning_rate=0.0001)
optimizer = "adam"
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

model.fit(X_train, y_train, epochs=20)  #, verbose=0

y_pred = model.predict(X_test)


#%%
y_test2 = np.argmax(y_test,axis=1)
y_pred2 = np.argmax(y_pred,axis=1)

```

```

#%%
y_test2 = np.argmax(y_test,axis=1)
y_pred2 = np.argmax(y_pred,axis=1)



# Afficher la Matrice de confusion
# from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix(decisionTree, X_test, y_test)
# plt.show()


# Afficher le score de precision

print(accuracy_score(y_test2, y_pred2))

```

Résultat: 83.17 %

CONVO-LSTM:

```
X = np.reshape(X,(len(X),81,1))
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =0.2)


#%%
input_shape = (81,1 )

model = Sequential()

model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=input_shape))
model.add(MaxPooling1D(pool_size=2))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(50, activation='relu'))

model.add(BatchNormalization())

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(3, activation='softmax'))
model.summary()
#%%
optimizer = RMSprop(learning_rate=0.0001)
optimizer = "adam"
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

model.fit(X_train, y_train, epochs=20)  #, verbose=0

y_pred = model.predict(X_test)
```

```
#%%
y_test2 = np.argmax(y_test,axis=1)
y_pred2 = np.argmax(y_pred,axis=1)



# Afficher la Matrice de confusion
# from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix(decisionTree, X_test, y_test)
# plt.show()


# Afficher le score de precision

print(accuracy_score(y_test2, y_pred2))
```

Résultat : 87.91 %

4.6 Conclusion :

Ce chapitre a été divisé en deux parties : implémentation, résultats. Dans la première partie, j'ai présenté les environnements de travail, les langages de programmation, les bases d'apprentissage et de test, les détails de l'implémentation de la phase d'apprentissage. Dans la deuxième partie, j'ai présenté les résultats. D'après les résultats, le modèle proposé CNN-LSTM (CONVO-LSTM) donne les meilleurs résultats.

Conclusion générale

Avant de déposer la plume et concrétiser les résultats auxquels a abouti mon travail qui consiste à étudier et analyser les méthodes de détection des Ransomwares existantes et ensuite proposer et évaluer une approche de détection, il est nécessaire de rappeler l'ensemble des étapes que j'ai suivi :

En première étape j'ai étudié les Ransomware (définition, classes, fonctionnement, types et auteurs)

En deuxième étape j'ai présenté les différentes méthodes de détection des Ransomware: dynamique, statique et hybride et comparé entre ces méthodes.

En troisième étape j'ai présenté la méthode de détection proposée en utilisant Les algorithmes du « Deep Learning ».

En dernière étape j'ai implémenté et évalué la méthode proposée en utilisant les environnements : Anaconda, Jupyter et Cuckoo Sandbox pour

A travers ce projet j'espère avoir contribué à la lutte contre ce type de malware (Ransomware) qui est devenu une menace sérieuse pour le monde informatique en proposant une méthode adéquate dont le but est de détecter l'attaque, classifier les ransomwares, ou détecter la pré-attaque d'un ransomware.

Bibliographie

[1] Helen Jose Chittooparambil, Bharanidharan Shanmugam ,Sami Azam , Krishnan Kannoorpatti , Mirjam Jonkman ,and Ganthan Narayana Samy, “A review of ransomware families and detection methods”, in: International Conference of Reliable Information and Communication Technology, 2018, pp. 588–597.

[2] Understanding the depth of the global ransomware problem, http://www.malwarebytes.com/pdf/. page consultée le 16 août 2022.

[3] Pete Burnapa , Richard French, Frederick Turner , KevinJones , “Malware classification using self organizing feature maps and machine activity data”, Computers & Security. Vol 73 (2018) 399–410.

[4] Xin Luo, Qinyu Liao, “Awareness education as the key to ransomware prevention’’ Information Systems Security, Vol 16,195–202, (2007).

[5] https://www.avast.com/fr-fr/c-malwares, page consultée le 02 août 2022.

[6] https://www.kaspersky.de/resource-center/threats/ransomware-wannacry, page consultée le 10 août 2022.

[7] https://www.savoirdanslavie.com/ransomware-attack-steps-to-take/ page consultée le 25 août 2022.

[8] http://lepouvoirclapratique.blogspot.fr/ , page consultée le 02 août 2022.

[9] Agence de l’Union européenne pour la cybersécurité (ENISA). www.enisa.europa.eu , page consultée le 02 September 2022.

[10] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda,

“A large-scale, automated approach to detecting ransomware”, in: 25th {USENIX} Security Symposium ({USENIX} Security 16), 2016, pp. 757–772.

[11] Sajad Homayoun, Ali Dehghantanha, Marzieh Ahmadzadeh, Sattar Hashemi, Raouf Khayam, " Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence", IEEE Transactions on Emerging Topics in Computing (Volume: 8, Issue: 2, 01 April-June 2020).

[12] Yuki Takeuchi, Kazuya Sakai, Satoshi Fukumoto, "Detecting ransomware using support vector machines",in: ICPP '18: Proceedings of the 47th International Conference on Parallel Processing Companion, August 2018, Article No:1-Pages 1–6

[13] Vinayakumar R, Soman KP and K.K.Senthil Velan, Shaunak Ganorkar ,"Evaluating shallow and deep networks for ransomware detection and classification", in: 2017 International Conference on Advances in Computing, Communications and Informatics,(ICACCI), pp. 259–265.

[14] Mattias Wecksten, Jan Frick, Andreas Sjostrom, Eric Jarpe, "A novel method for recovery from crypto ransomware infections", in: 2016 2nd IEEE International Conference on Computer and Communications (ICCC), pp. 1354–1358.

[15] Aragorn Tseng, YunChun Chen, YiHsiang Kao, Tsungnan Lin ,"Deep learning for ransomware detection", IEICE Tech. Rep., vol. 116, no. 282, IA2016-46, pp. 87-92, Nov. 2016.

[10] Li Chen, Chih-Yuan Yang, Anindya Paul, Ravi Sahita ,"Towards resilient machine learning for ransomware detection", 2019, arXiv preprint arXiv:1812.09400.

[17] Matilda Rhode, Pete Burnap, Kevin Jones , "Early-stage malware prediction using recurrent neural networks", Computers & Security ,77 (2018) 578–594.

[18] Hanqi Zhang, Xi Xiao , Francesco Mercaldo , Shiguang Ni , Fabio Martinelli , ArunKumar Sangaiah, "Classification of ransomware families with machine learning based on N-gram of opcodes". Future Gener Comput Systems90 (2019) 211–221.

[19] James Baldwin, Ali Dehghantanha, "Leveraging support vector machine for opcode density based detection of crypto-ransomware", Cyber Threat Intell. (2018) 107–136.

[20] Kul Subedi, Daya Ram Budhathoki ,Dipankar Dasgupta ,"Forensic analysis of ransomware families using static and dynamic analysis", in: 2018 IEEE Security and Privacy,Workshops (SPW), 2018, pp. 180–185.

[21] Saiyed Kashif Shaukat, Vinay J. Ribeiro, "RansomWare: A layered defense system against cryptographic ransomware attacks using machine learning", in: 2018 10th International Conference on Communication Systems & Networks (COMSNETS), 2018, pp. 356–363.

[22] Alberto Ferrante, Miroslaw Malek, Fabio Martinelli, "Extinguishing ransomware-a hybrid approach to android ransomware detection", in: International Symposium on Foundations and Practice of Security, 2017, pp. 242–258.

[23] Chris Moore, "Detecting ransomware with honeypot techniques", in: 2016 Cybersecurity and Cyberforensics Conference (CCC), 2016, pp. 77–81.

[24] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele, " PayBreak: Defense against cryptographic ransomware", in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 2017, pp. 599–611.

[25] Nolen Scaife, Henry Carter, Patrick Traynor, Kevin R.B. Butler, "Cryptolock (and drop it): stopping ransomware attacks on user data", in: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), 2016, pp. 303–312.

[26] Andrea Continella , Alessandro Guagnelli , Giovanni Zingaro, Giulio De Pasquale,

" ShieldFS : a self-healing, ransomware-aware file system ", in: Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016, pp. 336–347.

[27] Yann LeCun, Yoshua Bengio ,Geoffrey Hinton "Deep Learning", Insight review, Vol 5 2 1 Nature ,pp .4 3 7-447 ,2015.

[28] Ivan Vasilev, Daniel Slater, Gianmario Spacagna, Peter Roelants, Valentino Zocca, "Python Deep Learning", Packt Publishing, Second Edition ,p 379,2019.

[29] https://www.news-medical.net/health/Diabetic-Retinopathy-Risk-Factors-(French).aspx

page consultée le 15 août 2022.

[30]https://www.has-sante.fr/portail/upload/docs/application/pdf/2011-03/fiche_de_synthese_r , page consultée le 20 août 2022.

[31] Yann LeCun,Yoshua Bengio, “Convolutional Network for Speech, Image and Time-Series”, AT&T Bell Laboratory, Holmdel, 1991

[32] Soufiane Necib,’’ Fusion de face 3D couleur,profondeur et profil pour srv3D’’ ,mémoire de master, Université de Mohamed khaider , Biskra, 2013

[33] Pierre Buyssens, Abderrahim Elmoataz, ‘’Réseaux de neurones convolutionnels multi-échelle pour la classification cellulaire’’ , RFIA 2016, Clermont-Ferrand, France, Jun 2016.

[34] Moez Baccouche, ‘’Apprentissage neuronal de caractéristiques spatio-temporelles pour la classification automatique de séquences vidéo’’ Thèse de doctorat, INSA de Lyon, France, 2013.

[35] https://www.jedha.co/formation-python/ananconda-python , page consultée le 10 août 2022.

[36] https://www.ionos.fr/digitalguide/sites-internet/developpement-web/jupyter-notebook/ ,page consultée le 04 septembre 2022.

[37] https://www.varonis.com/blog/cuckoo-sandbox , page consultée le 02 août 2022.