Stage de Master Recherche

Rapport bibliographique

Analyse comportementale de logiciels malveillants

Domaine: CyberSécurité

Auteur:
Benjamin Bouguet

Superviseur:
Jean-Louis Lanet
INRIA - TAMIS

Abstract: Un ransomware est un logiciel malveillant qui rend inutilisable les fichiers d'un utilisateur puis, demande une rançon afin de restaurer leurs usages. Cette menace est de plus en plus populaire chez les cybercriminels. Les solutions antivirales classiques ne sont pas efficaces pour le moment pour détecter efficacement et mitiger de telles attaques. Ce stage m'amènera à travailler dans l'équipe TAMIS de l'INRIA, dont les domaines sont l'analyse de logiciels malveillants, de vulnérabilités ainsi que la sécurité des systèmes de paiement. Je serais amené à travailler sur la détection des ransomware, ainsi que sur l'analyse post-mortem de codes binaires partiellement dépaquetés. Dans cet état de l'art, on abordera les solutions existantes en rétro-ingénierie et en aide à la visualisation afin d'assister l'analyse de logiciels malveillants.

Table des matières

1 Introduction **1**

2 Rétro-ingénierie **2**
2.1 Méthodologie 3
2.2 Empaquetage de logiciels 4
2.3 Bac à sable 5
2.4 Recherche du matériels cryptographiques 5

3 Visualisation de fichier binaire **6**
3.1 Localisation du code et des données 6
3.1.1 Visualisation des sections d'un fichier binaire 6
3.1.2 Visualisation du graphe de flot de contrôle 8
3.2 Comparaison 9
3.2.1 Comparaison du comportement 9
3.2.2 Comparaison des similitudes du fichier binaire 10

4 Conclusion **11**

1 Introduction

Un ransomware est un logiciel malveillant capable de bloquer l'accès à un ordinateur et de modifier le contenu des fichiers utilisateur en les rendant inutilisables. L'objectif étant de forcer le paiment d'une rançon en échange de la restauration des fichiers et le cas échéant du débloquage de la machine. Il existe deux types de ransomware. Les "lockers" qui bloquent simplement l'accès à l'ordinateur et les "cryptors" qui chiffrent les données personnelles de l'utilisateur. Dans cet état de l'art, nous nous focaliserons sur le deuxième type : les "cryptors". Une fois sur la machine infectée, ce type de maliciel chiffre tous les fichiers personnels de l'utilisateur et demande un paiement en échange de la clé de déchiffrement. Les ransomware se propagent via les mails (i.e., pièces jointes, liens) pour plus de la moitié des infections, mais aussi par des failles de sécurité dans les navigateurs et des applications malicieuses. Les particuliers ne sont pas les seuls visés, de nombreuses entreprises et institutions sont ciblées (hôpitaux, commissariats, métros, etc) mais aussi des agences nationales (Ministère du travail, Sénat US). Au premier trimestre 2016, 209 millions de dollars ont été payés aux attaquants.

Le but d'un ransomware est de faire gagner de l'argent à l'auteur de l'attaque. Les méthodes de paiement sont variées. Les premiers ransomware utilisaient des transferts bancaires et des cartes prépayées. Des numéros de téléphone et des SMS surtaxés ont aussi été utilisés plus rarement. Ces méthodes ne sont plus utilisées à cause de la facilité pour les forces de l'ordre de retrouver l'initiateur de l'attaque. Les nouveaux ransomware utilisent la crypto-monnaie Bitcoin pour le paiement de la rançon. La confidentialité et l'intégrité de la transaction est assurée par le système Bitcoin, ce qui en fait un moyen prisé par les attaquants. Cette monnaie cryptographique est une des raisons du gain de popularité des ransomware chez les cybercriminels.

Problématique La détection de logiciels malveillants repose principalement sur de la détection par signatures à l'heure actuelle. Cette méthode utilisée par les anti-virus notamment, est limitée et il est facile de la contourner avec des techniques comparables à une chaîne d'assemblage d'usine (i.e., hash factory). L'utilisation de l'apprentissage automatique où de méthodes statistiques donnent de meilleurs résultats mais nécessitent un jeu important de données afin de "calibrer" l'algorithme durant la phase d'apprentissage. Avec de telles méthodes, le nombre de faux positifs peut-être potentiellement élevé. De plus, il est toujours délicat d'évaluer l'efficacité de l'algorithme (nouvelles familles, variantes, etc).

Le but des chercheurs est de trouver des moyens d'empêcher la contamination ainsi que de trouver des façons de déchiffrer les données sans avoir à payer la rançon. Pour cela plusieurs solutions ont été proposées. Les premières contre-mesures proposées sont de faire des sauvegardes régulières des données, de protéger son ordinateur (ou son réseau) avec des logiciels (ou matériels) adaptés tels qu'un pare-feu, un antivirus ou des logiciels de détection d'intrusion. Dans le cas où le logiciel malveillant utiliserait les API de cryptographie de l'OS, une contre-mesure proposée par Adam Young et Moti Yung [15] serait d'avoir une clé physique (sous forme de carte à puce par exemple) qui permettrait d'accéder à l'API. Cela serait efficace uniquement si le ransomware utilise l'API du système d'exploitation.

L'analyse d'un logiciel malveillant se fait par diverses techniques et méthodes. La rétro-ingénierie est utilisée par les analystes et les chercheurs pour comprendre comment fonctionne un logiciel et ainsi chercher des moyens de le détecter. Les logiciels malveillants ont plusieurs techniques pour rendre le travail plus difficile pour l'analyste. Et des outils sont créés par des analystes et des chercheurs pour contrecarrer à leur tour les logiciels malveillants. Parmi les outils de rétro-ingénierie, la visualisation est une des techniques qui se développe de plus en plus. Elle permet d'avoir une représentation graphique (i.e., CFG, ville) d'un logiciel, ce qui apporte une aide précieuse quant à la localisation et compréhension de la structure d'un binaire. L'aide à la visualisation peut se faire directement sur le code source, le fichier binaire ou sur l'image mémoire du logiciel. Dans le cas des ransomware, on se concentrera sur la visualisation de fichier binaire et de l'image mémoire. La visualisation peut aussi servir à modéliser le comportement d'un logiciel sur la mémoire, le système, le réseau, etc.

Pour l'analyse du comportement d'un ransomware, plusieurs solutions préexistent. On peut notamment citer UNVEIL [7] qui crée un environnement utilisateur et analyse les entrées/sorties d'un processus sur le disque (i.e., écriture, lecture) pour trouver des motifs qui identifieraient un ransomware. Celui-ci est dédiée à l'analyse post-mortem dans une sandbox. De façon assez similaire, CryptoDrop [12] analyse les changements opérés sur les fichiers privés de l'utilisateur au cours de l'exécution. Celui-ci compare l'entropie pour chaque fichier avant et après écriture et vérifie si celle-ci a augmenté de manière suspecte. Cette solution rend le système inutilisable, en raison de son coût.

Cet état de l'art est réalisé en vue de mon stage dans l'équipe TAMIS (Threat Analysis and Mitigation for Information Security) de l'INRIA à Rennes. Les thèmes de recherche de cette équipe sont l'analyse de logiciels malveillants, de vulnérabilités et les systèmes de paiement sécurisé. On parlera des techniques de rétro-ingénierie pour chercher des clés de déchiffrement ainsi que des outils pour dépaqueter un fichier binaire. Dans la seconde partie nous parlerons plus spécifiquement des techniques de visualisation afin d'assister l'analyste lors de la rétro-ingénierie. Nous aborderons ensuite l'analyse comportementale du logiciel malveillant sur le système.

2 Rétro-ingénierie

La rétro-ingénierie consiste en l'étude d'un logiciel ou d'un fichier pour en découvrir le fonctionnement. Dans le cadre de ce stage, la rétro-ingénierie est un concept central pour découvrir comment fonctionne un ransomware afin de trouver des solutions efficaces pour le détecter. La rétro-ingénierie est un ensemble de techniques très vaste. Dans cette section, on se concentrera sur les techniques pour analyser un logiciel empaqueté et pour rechercher le matériel cryptographique. L'empaquetage d'un logiciel est une opération qui consiste à obfusquer, compresser et/ou à chiffrer des sections d'un fichier binaire afin de rendre l'analyse statique du logiciel plus difficile. L'empaquetage est utilisé à la fois par des logiciels bénins, pour cacher des secrets de conception et des algorithmes, et par des logiciels malveillants, pour dissimuler leur présence. D'autres opérations peuvent être ajoutées afin de compliquer l'analyse de celui-ci, comme par exemple l'ajout de code anti-debogage ou de code qui détecte une exécution sur une machine virtuelle. Dans cette section, nous allons aborder en premier la méthodologie de la rétro-ingénierie, puis l'empaquetage de logiciel, ensuite les hyperviseurs de machines virtuelles et pour finir, retrouver le matériel cryptographique d'un binaire.

2.1 Méthodologie

Il n'existe aucune méthodologie standard pour la rétro-ingénierie. Les étapes décrites dans cette section servent à introduire des concepts de rétro-ingénierie qui seront utiles lors de mon stage. Cette méthodologie est tirée de l'article de Daniel A. Quist et al. [10].

Machine virtuelle L'utilisation d'une machine virtuelle est utile, spécialement dans le cas de l'analyse d'un logiciel malveillant. Il faut installer sur cette machine virtuelle un système d'exploitation qui correspond à la cible du logiciel malveillant et qui ressemble le plus possible à un environnement d'un utilisateur lambda. Les hyperviseurs de type 2 tels de VMWare ou VirtualBox permettent de créer des instantanés du système installé. Cette fonctionnalité est utile pour remettre le système dans un état sain après l'exécution du logiciel malveillant. L'utilisation d'une machine virtuelle a pour but de ne pas infecter sa propre machine avec un logiciel malveillant et de pouvoir contrôler son exécution. De plus, l'utilisation de machine virtuelle permet d'analyser ou de cloisonner le trafic du réseau plus facilement. En utilisant, par exemple, des logiciels comme Wireshark sur la machine hôte ou un miroir de port sur un commutateur.
Néanmoins, des solutions existent pour faire de l'analyse *baremetal*, c'est-à-dire sur une machine non virtuelle. On peut citer par exemple la structure logicielle de Dhilung Kirat et al. [8] qui permet de faire de l'analyse dynamique sur une machine *baremetal*.

L'obfuscation L'obfuscation est une technique qui consiste à modifier le code d'un programme (source ou binaire) afin de le rendre plus difficilement analysable par un humain mais qui garde toujours la même sémantique pour l'ordinateur. L'obfuscation est utilisé par les logiciels bénins comme par les logiciels malveillants, respectivement pour protéger des secrets industriels et pour empêcher la détection. Beaucoup de méthodes pour protéger le code et les données d'un logiciel ont été développées depuis les années 80. On peut citer la modification des structures de données pour les rendre moins compréhensibles, l'ajout de code "mort" (code qui n'est jamais exécuté mais qui induit l'analyste en erreur), le chevauchement de code assembleur, l'empaquetage (qui sera développé dans une partie suivante) et bien d'autres ... Ces modifications doivent être enlevées si possible automatiquement, avant le désassemblage.

Désassemblage Le désassemblage est le fait de reconstruire le code source d'un fichier binaire, souvent en langage machine (assembleur). IDA Pro est l'un des outils les plus utilisés et les plus puissants pour réaliser cette opération. De plus, de nombreux modules d'extension sont disponibles pour étendre ses fonctionnalités. Le désassemblage permet d'avoir une vision de plus haut niveau. La difficulté de cette opération réside dans la compréhension du code binaire et de distinguer les instructions à exécuter et les données. Certains logiciels malveillants possèdent des techniques pour biaiser l'analyse du désassembleur et ainsi lui faire afficher un code assembleur faux. Une fois toutes les protections du fichier binaire enlevées, l'analyste peut plus facilement comprendre les actions du logiciel et chercher comment le détecter ou désinfecter des machines.

Signature d'un logiciel La signature d'un logiciel est une sorte d'empreinte qui reconnaît le logiciel parmi d'autres. Il s'agit de fragments de codes qui appartiennent au logiciel malveillant et qui permettent au système de détection classique de repérer un logiciel malveillant lorsqu'il trouve ces codes dans un fichier binaire. Ces signatures permettent aussi d'extraire le code malveillant

d'un fichier binaire infecté. Les antivirus ou les systèmes de détection d'intrusion telle que Snort[1] utilisent des signatures. Ils maintiennent une base de données contenant toutes les signatures, qu'ils mettent à jour lorsque les analystes découvrent de nouvelles menaces. D'autres outils, comme Yara[2], utilisent les signatures pour identifier et classer des logiciels malveillants.

2.2 Empaquetage de logiciels

L'empaquetage d'un logiciel compresse et chiffre des sections d'un fichier binaire. Une routine de décompression et/ou de déchiffrement est ajoutée au binaire et est appelée au début de l'exécution. Cela pose problème aux analystes qui souhaitent étudier le logiciel. En effet, récupérer la charge utile du logiciel (la partie déchiffrée) est difficile pour un analyste avec des outils traditionnels tels que IDA Pro. De plus, les logiciels malveillants protègent leur code avec de nombreuses couches d'empaquetages et possèdent des mécanismes d'auto-modification, rendant la détection encore plus difficile. Plusieurs méthodes existent pour récupérer la charge utile d'un binaire empaqueté.

Guillaume Bonfante et al.[1] ont développé CoDisasm, un désassembleur qui permet de récupérer la charge utile d'un binaire empaqueté. Pour cela, il utilise de l'analyse dynamique et statique. Les algorithmes de dépaquetage fonctionnent de la façon suivante : ils lisent des données en mémoire, les déchiffrent puis les exécutent en mémoire. Les algorithmes de dépaquetage peuvent réaliser cette manipulation plusieurs fois de suite si le binaire a été empaqueté plusieurs fois. La première étape de leur architecture de désassemblage analyse dynamiquement le logiciel. Ils collectent les traces de l'exécution du logiciel. Ils récupèrent notamment, l'ID du processus, les valeurs des registres et les accès en lecture et écriture de la mémoire. Grâce à ces accès mémoire, ils identifient les zones de mémoires qui sont écrites puis exécutées. Les auteurs définissent ces actions par le concept de *wave*. Un algorithme de dépaquetage peut exécuter successivement plusieurs *waves*, qui signifient plusieurs étapes de décompression/déchiffrement. Pour chaque *wave* reconnu par le désassembleur, l'outil réalise un instantané de la mémoire. Une fois toutes ces données récupérées, l'outil réalise une analyse statique de reconstruction de code : le désassemblage. Pour chaque *wave*, l'outil désassemble l'instantané à l'aide de la trace d'exécution. CoDisasm produit un exécutable au format Portable Executable (PE) pour chaque *wave*. CoDisasm n'utilise qu'une trace d'exécution pour faire son analyse ce qui le rend vulnérable à des attaques par évasion des logiciels malveillants qui pourraient changer leur exécution s'ils détectent la présence de l'outil.

OmniUnpack est un outil développé par Lorenzo Martignoni et al.[9] et qui possède sensiblement le même fonctionnement que Codiasm. Cet outil surveille également la mémoire d'un processus afin de détecter les accès mémoire en écriture puis en exécution. Un instantané de la mémoire est ici aussi réalisé. La différence réside dans le fait qu'OmniUnpack envoie cet instantané à un détecteur de logiciels malveillants quelconque et arrête l'exécution du logiciel si c'est un maliciel. Il a donc pour but la détection du maliciel plutôt que son analyse. OmniUnpack s'installe sur un système d'exploitation Windows comme un driver. Il est donc insensible aux techniques d'évasion des logiciels malveillants tels que la détection de machines virtuelles, de débogage.

[1] https://www.snort.org/
[2] http://virustotal.github.io/yara/

2.3 Bac à sable

Comme évoqué dans la partie sur la méthodologie de la rétro-ingénierie, les analyses utilisent des machines virtuelles pour étudier les logiciels malveillants. Afin de contrer cette méthode, les auteurs de logiciels malveillants empêchent ou change l'exécution du logiciel malveillant s'il se trouve dans une machine virtuelle. L'une de ces techniques est par exemple, d'utiliser une instruction en assembleur qui n'est pas documentée, peu utilisée par les logiciels en général et qui n'est pas implémenté dans la machine virtuelle. Sur un processeur normal, l'instruction sera reconnue et exécutée normalement, alors que sur la machine virtuelle, un autre comportement peut arriver, allant jusqu'à faire planter la machine virtuelle. Le logiciel malveillant peut détecter ce changement de comportement et changer son exécution en conséquence.

C'est dans ce contexte que Ether[4] a été créé, un hyperviseur basé sur Xen[3]. Ether est un hyperviseur transparent, c'est-à-dire, qu'un logiciel ne peut pas distinguer s'il s'exécute sur une machine virtuelle ou sur un processeur normal. Il s'accompagne d'outil pour l'analyse de logiciels malveillants. Pour être transparent, l'hyperviseur et les outils d'analyse doivent avoir des propriétés particulières. Les outils doivent s'exécuter dans un environnement avec des privilèges sur la machine plus élevés que le logiciel à analyser. Les instructions doivent s'exécuter de la même manière et la mesure du temps doit être la même que sur un processeur classique. Ether peut se résumer à une série de composant pour l'hyperviseur Xen et à des outils d'analyse. Xen est un hyperviseur de type 1, c'est-à-dire qu'il s'exécute directement sur le matériel, à la manière d'un système d'exploitation, contrairement aux hyperviseurs de type 2 (comme VirtualBox) qui s'exécutent sur un système d'exploitation "hôte". Ether modifie Xen et lui permet de détecter des évènements comme les appels système, de suivre l'exécution des instructions et de contrôler les écritures en mémoire. Ether assure les deux premières propriétés citées plus haut, mais pas la contrainte temporelle.

L'une des techniques d'évasion des logiciels malveillants est l'attaque temporelle. L'exécution d'un logiciel dans un environnement virtuel est plus lente que dans un environnement natif. Les logiciels malveillants peuvent se servir de cet indicateur pour modifier leurs fils d'exécution et cacher leurs comportements néfastes aux analystes. L'article de Christopher Thompson et al. [13] montre qu'il est facile de trouver des moyens de détecter ce manque de performance et ainsi déduire que le logiciel s'exécute dans une machine virtuelle. Ils montrent ces différences en utilisant un programme qui compte le temps d'exécution de plusieurs threads. Les résultats obtenus montrent les différences entre plusieurs hyperviseurs et un système *baremetal* (natif).

L'utilisation du déboguer Qira.me[4] permet de contrecarrer ce type d'attaque. Ce déboguer enregistre l'état de la machine à chaque instruction et fait du débogage "dans le passé". Qira.me est basé sur l'émulateur QEMU.

2.4 Recherche du matériels cryptographiques

La fonctionnalité centrale d'un ransomware est sa faculté à chiffrer des fichiers. Pour l'analyste, il est important d'identifier le chiffrement utilisé et de répérer ses fonctions dans l'exécutable ainsi que de chercher les clés de chiffrement, dans le but de trouver un moyen de déchiffrer les données

[3] https://www.xenproject.org/
[4] http://qira.me

sans avoir à payer la rançon. Pour les chiffrements symétriques, la clé de chiffrement est la méme que celle pour déchiffrer. Il doit être possible de retrouver cette clé dans le fichier binaire ou dans la mémoire du processus. Des modules d'extension pour IDA Pro tels que FindCrypt2 ou IDAScope fournissent déjà des algorithmes pour détecter du code de chiffrement en faisant une analyse statique par signatures.

Joan Calvet et al. [2] ont développé l'outil Aligot, qui reconnaît les algorithmes de chiffrement. Cet outil analyse le logiciel de façon dynamique. Il trace l'exécution du logiciel, puis analyse les résultats pour reconnaitre des boucles. Ils font le constat que de nombreuses implémentations d'algorithmes de chiffrement utilisent des boucles. L'outil reconnait et construit des modèles de boucles servant aux chiffrements, il peut ensuite arriver à extraire les paramètres. Les paramètres de ces boucles sont la clé de chiffrement et le texte en clair. Ils font ensuite une comparaison de ce modèle avec des modèles d'algorithmes connus qu'ils auront généré auparavent. Les expérimentations menées par les auteurs montrent que leur outil est efficace contre des algorithmes tels que AES, MD5 ou RC4. Néanmoins, Aligot ne reconnait pas des algorithmes inconnus (dont le modèle de boucles n'est pas connu). De plus, un auteur de logiciel malveillant peut réécrire un algorithme connu de façon à ce que le modèle de boucle ne soit plus reconnu par Aligot.

AES Finder[5] est un programme utilitaire qui récupère les clés de chiffrement et de déchiffrement de l'algorithme AES dans un processus en cours d'exécution. Il est compatible avec Linux, Windows et MacOs. Il prend en entrée le PID d'un processus et lit sa mémoire pour y trouver des clés. Ce programme peut retrouver des clés de 128, 192 et 256 bits. Les ransomware doivent stocker les clés de déchiffrement dans leur mémoire, des outils comme AES Finder peuvent nous aider à les retrouver, dans l'hypothèse où celles-ci ne sont pas stockées dans des conteneurs chiffrés.

3 Visualisation de fichier binaire

L'apparition, toujours plus importante, de nouvelles variantes de logiciels malveillants nécessite de nouvelles techniques pour les analystes. C'est dans ce contexte que la visualisation s'est développée. Cette technique a pour but de créer des images ou des graphiques qui représentent les logiciels, permettant d'avoir un aperçu rapide du maliciel ou de ses propriétés. La visualisation sert à l'analyse, à la détection, à la classification et à la comparaison de logiciels malveillants.

3.1 Localisation du code et des données

La localisation du code d'un fichier binaire, dont on ne connaît a priori aucune information, est important pour commencer la rétro-ingénierie. La visualisation crée des images qui représentent graphiquement un fichier binaire ou génére des graphes de flot de contrôle avec des techniques d'analyse statique ou dynamique.

3.1.1 Visualisation des sections d'un fichier binaire

InSeon Yoo [14] est l'un des premiers à utiliser la visualisation pour détecter un logiciel malveillant. Dans son article, il se penche plus particulièremet sur la détection de virus dans un exécutable

[5] https://github.com/mmozeiko/aes-finder

pour Windows. Il créé une image à l'aide d'un Self-Organizing Map (SOM), aussi appelé carte de Kohonen. Un SOM est une catégorie de réseau de neurones basé sur de l'apprentissage non supervisé. Il arrive a distinguer le code du virus de celui du programme initial car celui-ci ne ressemble pas à du code compilé mais à du code qu'on aurait "forcé" à ajouter. De plus, la position de ce code est aussi un facteur de la présence de virus, le code des virus étant généralement ajouté à la fin du fichier binaire. L'entrainement du SOM est fait avec plusieurs fichiers binaires avant et après leur infection par plusieurs virus et leurs variantes. Les caractéristiques d'un fichier infecté sont apprises par le SOM et possède la capacité de les reconnaitre dans un nouveau fichier binaire. Les résultats du SOM sont sous forme de matrice et des fonctionnalités de MatLab sont utilisées pour retranscrire cette matrice sous forme d'image. Les images obtenues permettent de voir la présence d'un virus dans un exécutable, on peut observer un exemple dans la figure 1. Il s'agit d'une première étape dans la localisation visuelle de code dans un fichier binaire. Je n'ai pas trouvé de cas de ransomware qui infecte des exécutables, mais cet article montre qu'il est possible de localiser des fragments de code qui peuvent nous intéresser dans un fichier binaire.

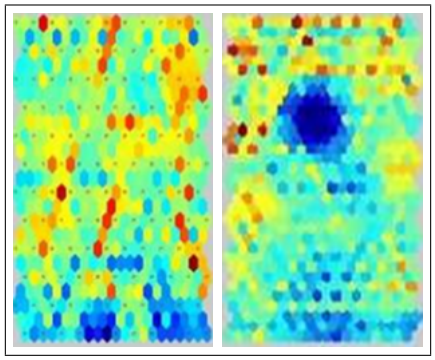

Figure 1: Exemple d'image générée par le SOM. À gauche, l'image d'un fichier sain, à droite celle d'un fichier infecté. La tache bleue représente le code d'un virus. *Images reprises de l'article [14]*

Les techniques plus récentes de visualisation mettent en évidence des structures de données dans un fichier dont on ne connait pas la structure mais aussi dans une image mémoire d'un logiciel. Gregory Conti et al. [3] ont développé un outil qui permet de visualiser un fichier binaire avec différentes représentations. La première associe chaque bit à une couleur, ce qui permet d'avoir une vue d'ensemble du binaire. Trois autres types de visualisations sont disponibles et apportent chacune des informations complémentaires. Les exemples donnés dans l'article sont sur des fichiers binaires, mais il est montré aussi que cela est possible sur une image mémoire d'un logiciel. Cela est intéressant dans le cas d'un ransomware. En effet, afin de contrer les techniques d'obfuscation des maliciels, l'image mémoire du processus est utilisée. Cette méthode serait utile pour visualiser les données du ransomware, afin de réussir à trouver les clés de déchiffrement, ou bien des données qui indiqueraient la méthode de parcours des fichiers du ransomware. On peut imaginer que le ransomware stocke en mémoire une liste des fichiers qu'il a déjà chiffré. Cet outil peut être un bon complément de l'éditeur hexadécimal habituel d'un analyste.

BinVis[6] est un exemple d'outil permettant de visualiser n'importe quel fichier. Cet outil affiche des structures de données similaires avec la même couleur, améliorant ainsi la rétro-ingénierie. Il a aussi la possibilité de calculer et d'afficher l'entropie d'un fichier. Cela permet de voir les zones d'un logiciel qui ont été empaquetées ou de visualiser plus rapidement les endroits du code qui contiennent le matériel cryptographique par exemple. Ce qui peut être utile dans la rétro-ingénierie des ransomware, car retrouver les clés de chiffrements/déchiffrements ainsi que les fonctions associées fait partie de l'analyse de ce type de logiciels malveillants. La génération de l'image est simple, elle affiche en blanc les 0x00, affiche en noir les 0xff, affiche en bleu les caractères ASCII et en rouge les autres éléments. L'arrangement des pixels ne suit pas l'ordre du binaire. En effet, la particularité de cet outil est d'utiliser la courbe de Hilbert pour afficher ses bits. Ce qui permet d'avoir une meilleure localisation des pixels et une meilleure représentation pour l'oeil humain. Un exemple d'image générée est disponible à la figure 2. Cet outil est disponible directement en ligne pour réalier facilement des tests. Le code source est disponible sur GitHub et il est donc possible de l'utiliser en local et d'y apporter des modifications au besoin.

D'autres outils utilisent la visualisation en 3D pour représenter un fichier binaire ou des structures de données. On peut citer par exemple Veles[7] et Binglide[8], dont les codes sources sont ouverts. Ces logiciels possèdent des fonctionnalités proches de celle des outils déjà présentés, mais fournissent aussi des représentations d'histogrammes de bigrammes ou de trigrammes, respectivement en 2D et en 3D.

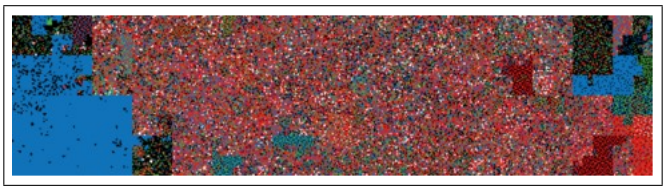

Figure 2: Visualisation d'un fichier ELF de Linux (ARM). *Image prise sur le site de BinVis*

3.1.2 Visualisation du graphe de flot de contrôle

La visualisation peut être utilisée aussi pour générer des graphes et dans notre cas, pour générer des graphes de flot de contrôle qui permettent d'enrichir encore la rétro-ingénierie.

Daniel A. Quist et al. [10] ont développé VERA, une architecture qui repose sur Ether [4]. VERA enregistre l'exécution du logiciel sur la machine et construit un graphe des blocs exécutés par le programme. Ce graphe est ensuite affiché à l'utilisateur et autorise des interactions tel que le zoom. La visualisation de ce graphe met en évidence des techniques d'obfuscation utilisées par les logiciels malveillants pour compliquer le travail des analystes. L'exemple donné dans l'article montre que le trojan Mebroot exécute une longue boucle de calcul inutile pendant un certain temps avant d'effectuer son vrai travail. Cela à pour but de faire perdre du temps à l'analyste qui souhaite étudier son fonctionnement. Le graphe généré par l'outil est disponible à la figure 3.

[6] http://binvis.io/
[7] https://github.com/wapiflapi/veles
[8] https://github.com/wapiflapi/binglide

Figure 3: Graphe de flot de contrôle de Mebroot. On peut voir à gauche la boucle de calcul inutile et à droite le travail effectif du maliciel. *Image reprise de l'article [10]*

3.2 Comparaison

Il existe de nombreuses familles de logiciels malveillants. Une famille regroupe plusieurs logiciels qui ont des propriétés identiques ou sont issus du même maliciel qui sert de base. Les auteurs génèrent des modifications pour rajouter ou modifier les fonctionnalités, ou tout simplement pour ne pas se faire repérer par les systèmes de détection classique. En effet, les anti-virus utilisent un système par signatures qui identifient un logiciel malveillant ou une famille. Si cette signature est trop spécifique, une petite modification du logiciel permet à celui-ci de ne plus se faire repérer. De plus, pour certains logiciels malveillants, comme les ransomware, il est possible de créer des générateurs, qui créent des variantes qui ont toutes une sémantique équivalente mais des codes différents. Il en existe aussi avec des codes polymorphiques, c'est-à-dire, que le code change à chaque exécution. Toutes ces techniques permettent aux logiciels malveillants de ne pas se faire repérer par les systèmes de détection. La visualisation compare et classe les logiciels suivant plusieurs méthodes. L'une des méthodes est d'utiliser le comportement des logiciels sur le système infecté afin de les regrouper. Une autre méthode est de créer une signature du binaire sous forme d'image et d'étudier les similarités de ces images pour identifier les familles.

3.2.1 Comparaison du comportement

Dans cette partie, le comportement d'un logiciel fait référence aux diverses traces qu'il peut laisser sur le système. Il s'agit en général des appels aux fonctionnalités du système. Les appels systèmes sont par exemple, l'accès au système de fichier en lecture ou écriture, l'accès au réseau...

Saxe et al. [11] ont développé un outil qui compare des logiciels malveillants en fonction des appels systèmes. Cet outil construit des graphiques en fonction des séquences d'appels systèmes récupérés depuis des logs. Leur algorithme construit ensuite un vecteur booléen en fonction de l'occurrence des appels systèmes et calcule une matrice de similarité. Une interface graphique permet ensuite d'explorer les résultats de l'analyse et de comparer les logiciels malveillants entre eux. Ceux qui possèdant des appels systèmes proches ou équivalents sont susceptibles d'être de la même famille.

Dans le cas d'un ransomware, les appels systèmes sont des lectures et écritures sur le gestionnaire de fichier. Pouvoir classer les ransomware en fonction du motif d'appel système qu'il effectue est une solution pour les détecter.

D'autres solutions existent, comme SEEM de Robert Gove et al. [5] Ce logiciel compare un grand nombre de logiciels malveillants en même temps sur plusieurs critères. Les expérimentations données par les auteurs montrent que leur logiciel améliore l'efficacité d'un analyste, lui permettant de comprendre plus rapidement le fonctionnement d'un maliciel en le comparant à d'autres connus.

Les techniques de détection de comportement des logiciels malveillants qui utilisent l'analyse des appels systèmes, les appels à des API et le trafic du réseau ne sont pas efficaces dans le cas d'un ransomware car ces derniers utilisent des appels systèmes similaires à des logiciels légitimes de compression ou de chiffrement.

3.2.2 Comparaison des similitudes du fichier binaire

Une autre méthode pour comparer des logiciels malveillants est de les classer en utilisant la visualisation. Il est possible de générer une image qui représente le logiciel et d'utiliser un algorithme de similitude de matrice (une image est une matrice de pixel) pour en comparer plusieurs et ainsi les regrouper.

KyoungSoo Han et al. [6] ont décrit une méthode pour créer ces images et les comparer. Leur solution se compose de trois phases distinctes. La première est de désassembler le logiciel malveillant avec un outil tel que IDA Pro ou tout autre désassembleur. Ils utilisent les opcodes obtenus pour générer les images. Deux fonctions de hachage prennent en entrée les opcodes et fournissent en sortie une coordonnée sur l'image et une couleur RGB. La fonction de hachage utilisée pour obtenir les coordonnées possède une propriété particulière qui assure que pour des entrées similaires, les sorties seront aussi similaires. L'image de départ étant complètement noire, le résultat obtenu est une image noire avec de nombreux points plus clairs. L'image finale est un carré, sa taille est définie par l'utilisateur et doit être assez grande pour éviter les collisions. Des exemples d'images générées par l'outil se trouvent dans la figure 4. Pour la comparaison, l'algorithme divise l'image en plusieurs parties égales et compare un certains nombre de pixels aléatoirement. Le nombre de division et le nombre de comparaison de pixels est laissé à l'appréciation de l'utilisateur. Le résultat de cet algorithme est un nombre entre 0 et 1, 1 étant deux images identiques et 0 deux images complètement différentes. Les résultats des expérimentations menées par les auteurs montrent que des logiciels malveillants de même famille possèdent des images proches (à l'oeil nu) et des scores proches de 1 pour la comparaison avec l'algorithme. Cet outil fonctionne uniquement pour des fichiers binaires qui n'ont pas été empaquetés. Un binaire empaqueté possède un effet une ou des sections chiffrées, celles-ci ressemblent donc à de l'aléa. Deux images de logiciel de la même famille auront des images complètement différentes. De même, un logiciel empaqueté avec deux logiciels différents aura des images différentes. La classification des logiciels malveillants est très important, notamment pour fournir une solution de détection qui prenne en compte l'ensemble des variantes possibles. Le nombre important de variantes d'un logiciel et la facilité pour les créateurs de les générer sont des contraintes qui font qu'il est impossible pour les analystes de fournir une solution de détection unique pour chaque logiciel malveillant.

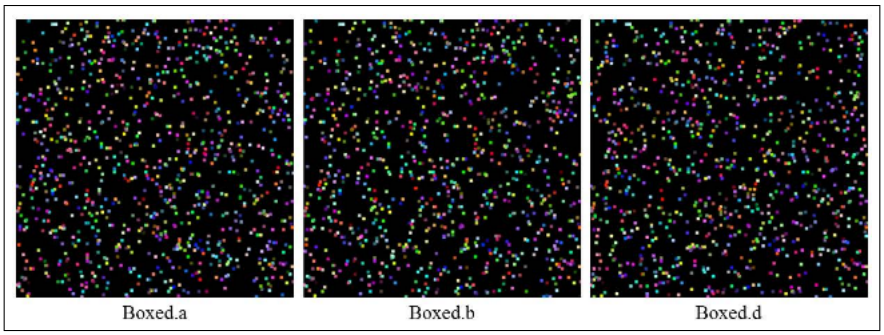

Figure 4: Exemples d'images générés par l'outil. Il s'agit de 3 variantes du trojan Trojan-DDoS.Win32.Boxed. *Image reprise de l'article [6]*

4 Conclusion

Dans cet état de l'art, nous avons abordé les différentes techniques pour la rétro-ingénierie de ransomware. Nous nous sommes concentrés sur les méthodes pour contrecarrer les protections des logiciels malveillants contre l'analyse, ainsi que sur la recherche du matériels cryptographiques. Le chiffrement étant la fonctionnalité centrale d'un ransomware, la recherche d'algorithmes et des clés de chiffrement est l'une priorité d'un analyste. Trouver une signature pour détecter ce logiciel malveillant en est une autre.
Nous avons présenté les outils de base utilisés par les analystes et les chercheurs, pour ensuite parler plus spécifiquement des hyperviseurs et du problème de l'empaquetage des logiciels. Nous avons évoqué les techniques utilisées par les auteurs de logiciels malveillants pour durcir l'analyse de leurs logiciels et les outils que les chercheurs ont développés pour les contourner. La récupération de la charge utile d'un logiciel malveillant permet de désassembler ce logiciel afin de l'analyser et en extraire une signature. Enfin nous avons présenté des outils comme AES Finder pour retrouver les clés de chiffrement et Aligot pour identifier les algorithmes. Dans le contexte d'un ransomware, ces informations pourraient être utiles pour récupérer les données chiffrées d'un utilisateur sans avoir à payer la rançon.
Nous avons présenté aussi l'état de l'art de la visualisation de logiciel, orienté sur la rétro-ingénierie de fichier binaire, ainsi que sur des représentations graphiques qui permettent de comparer et de classer les logiciels malveillants. Enfin, nous avons vu la détection d'un maliciel par son comportement sur le système avec et sans visualisation. Les appels système d'un ransomare sont principalement effectués sur le système de fichiers, pour lire et réécrire le contenu des fichiers présents. L'équipe TAMIS essaye de trouver une signature de ce comportement sur le système, en caractérisant les interactions avec le système de fichiers.
Tous ces outils et ces notions me seront utiles lors de mon stage qui consistera à apporter mon aide aux recherches de l'équipe TAMIS sur la détection des ransomware.

Références

[1] Guillaume Bonfante, Jose Fernandez, Jean-Yves Marion, Benjamin Rouxel, Fabrice Sabatier, and Aurélien Thierry. Codisasm: medium scale concatic disassembly of self-modifying bina-

ries with overlapping instructions. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 745–756. ACM, 2015.

[2] Joan Calvet, José M Fernandez, and Jean-Yves Marion. Aligot: cryptographic function identification in obfuscated binary programs. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 169–182. ACM, 2012.

[3] Gregory Conti, Erik Dean, Matthew Sinda, and Benjamin Sangster. Visual reverse engineering of binary and data files. In *Visualization for Computer Security*, pages 1–17. Springer, 2008.

[4] Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. Ether: malware analysis via hardware virtualization extensions. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 51–62. ACM, 2008.

[5] Robert Gove, Joshua Saxe, Sigfried Gold, Alex Long, and Giacomo Bergamo. Seem: a scalable visualization for comparing multiple large sets of attributes for malware analysis. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, pages 72–79. ACM, 2014.

[6] KyoungSoo Han, Jae Hyun Lim, and Eul Gyu Im. Malware analysis method using visualization of binary files. In *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, pages 317–321. ACM, 2013.

[7] Amin Kharraz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. Unveil: A large-scale, automated approach to detecting ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 757–772. USENIX Association, 2016.

[8] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. Barebox: efficient malware analysis on bare-metal. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 403–412. ACM, 2011.

[9] Lorenzo Martignoni, Mihai Christodorescu, and Somesh Jha. Omniunpack: Fast, generic, and safe unpacking of malware. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 431–441. IEEE, 2007.

[10] Daniel A Quist and Lorie M Liebrock. Visualizing compiled executables for malware analysis. In *Visualization for Cyber Security, 2009. VizSec 2009. 6th International Workshop on*, pages 27–32. IEEE, 2009.

[11] Josh Saxe, David Mentis, and Chris Greamo. Visualization of shared system call sequence relationships in large malware corpora. In *Proceedings of the ninth international symposium on visualization for cyber security*, pages 33–40. ACM, 2012.

[12] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin RB Butler. Cryptolock (and drop it): stopping ransomware attacks on user data. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 303–312. IEEE, 2016.

[13] Christopher Thompson, Maria Huntley, and Chad Link. Virtualization detection: New strategies and their effectiveness. *University of Minnesota.(unpublished)*.

[14] InSeon Yoo. Visualizing windows executable viruses using self-organizing maps. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 82–89. ACM, 2004.

[15] Adam L Young and Moti M Yung. An implementation of cryptoviral extortion using microsoft's crypto api. 2005.