Les solutions de détection de malware peuvent-elles vraiment nous protéger face à des attaques ciblées ?

Tuteur: Valérian Comiti (GLIMPS)

Mickael Courtiade, Tangui Didailler, Thomas Lebrun

Remerciements

Tout d'abord, nous tenons à exprimer notre profonde gratitude envers notre tuteur, Valérian Comiti, qui a été un soutien constant tout au long de la réalisation de ce mémoire. Son expertise sur le sujet, ses idées innovantes et son suivi attentif ont été d'une aide précieuse pour notre travail. Sa patience et sa compréhension face aux défis rencontrés nous ont permis de progresser et de surmonter les obstacles.

Nous souhaitons également remercier chaleureusement Samuel Bouvier et Benjamin Costé, les responsables de notre formation, pour leur écoute attentive et leur soutien tout au long de notre mémoire. Leur disponibilité a été essentielle pour nous, même face aux défis et aux imprévus rencontrés. Leur compréhension et leur soutien face à ces situations ont été d'une grande aide et nous ont permis de nous concentrer sur notre mémoire avec sérénité.

Un merci tout particulier à Samuel Kervella pour son esprit critique et sa bienveillance lors de la relecture et de la correction de certaines parties de notre mémoire. Ses conseils avisés nous ont aidés à améliorer la qualité de notre travail.

Pour toutes ces raisons, nous tenons à exprimer notre gratitude sincère et profonde à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire. Votre soutien et vos encouragements ont été essentiels pour nous aider à avancer.

Abstract

Depuis une dizaine d'années, le secteur de la cybersécurité fait face à des attaques de plus en plus croissantes et de plus en plus ciblées. Pour faire face à ce problème, les entreprises du domaine mettent au point de nouvelles techniques afin de détecter et d'analyser les menaces par malware de plus en plus précises et efficace. Ce mémoire a pour but d'éclaircir le sujet en proposant une revue systématique explicative des différentes techniques utilisées afin de détecter ces menaces, des moyens d'obfuscation utilisé par les attaquants afin de les contourner ainsi que d'une description de ces attaques et des moyens utilisés dans la nature pour s'en protéger. Enfin, nous émettrons un avis sur l'efficacité des méthodes de défenses proposé face à ses attaques et proposerons des améliorations.

Table des matières

Remerciements **1**

Abstract **2**

Introduction **5**

Qu'est ce que ce que la detection de malware ? Quelles sont les méthodes ? **10**
La détection de malware . . . 10
Statique . . . 10
Détection par signature . . . 10
Hachage (hash basique, fuzzy hash) . . . 10
Désassemblage / Décompilation . . . 12
Modèles de langage naturels . . . 13
Malware Features Extraction . . . 15
Dynamique . . . 17
Émulation . . . 17
Exécution . . . 18
Hybride . . . 20
Corrélation . . . 20
Heuristique . . . 21
Conclusion . . . 22

Techniques de dissimulation des malwares **23**
Obfuscation de code . . . 24
Obfuscation basée sur les données . . . 24
Obfuscation basée sur les structures de contrôle . . . 27
Packers . . . 31
Types de packers . . . 32
Fonctionnement d'un packer . . . 32
Techniques Anti débogueur . . . 35
Détection de débogueur par le malware . . . 35
Modification du comportement du malware en présence d'un débogueur . . . 37
Techniques Anti VM . . . 39
Les composants physiques . . . 39
Caractéristiques du système . . . 40
Évasion basée sur le temps . . . 41
L'activité de l'utilisateur . . . 42

Détection réseau 42
Conclusion 43

Attaques ciblées **45**
Panorama des attaques 45
MITRE ATT&CK 46
APTs 47
Modélisation par Kill Chains 49
Coût d'une attaque 52
ISO 27001 52
Quels sont les moyens employés pour se protéger d'une attaque ? 52
Macro 52
Micro 55
Conclusion 58

Conclusion **58**

Références **62**

Introduction

Les malwares ont fait leur apparition pour la première fois dans les années 1960. Les attaquants concevaient des virus principalement pour faire des blagues et s'amuser. Leurs créations affichaient généralement des messages inoffensifs(Joseph Regan et Ivan Belcic, 2022).

Dans les années 1960 à 1970, les virus étaient tous maintenus dans un environnement sécurisé dans des laboratoires et n'ont jamais réussi à s'échapper dans la nature. Le premier qui a réussi cet exploit était *Elk Clone* en 1981. Il s'est propagé via des disquettes infectées et affichait simplement un poème sur l'écran de l'utilisateur sans causer de dommage délibéré. Ensuite, le premier virus pour PC Microsoft à vu le jour, appelé Brain, qui était également le premier virus connu qui était capable de dissimuler sa présence sur le disque, évitant ainsi la détection.

En 1988, le premier exemple de virus avec volonté de nuire a été créé, le virus Vienna, qui permettait de chiffrer des données et de détruire des fichiers. Les premiers outils antivirus ont été crée suite à l'apparition de ce virus (Wikipedia, 2022).

Au début des années 1980, le concept de virus s'est répandu, et de nombreux exemples de virus ont vu le jour sur les ordinateurs Apple et IBM. Avec l'arrivée d'Internet dans les années 1990, les virus sont devenus popularisés, au point que Yisreal Radai à inventer le terme *malware* en 1990(Ellie Burns, 2021).

Au cours des décennies suivantes, tant sur leurs complexité qu'en nombre d'échantillons. Selon AV Atlas(AV Atlas, s.d.), la quantité de nouveaux échantillons de malwares générés a explosé au cours des dernières années, comme le montre la figure 1, a tel point qu'envion 5 nouveaux malwares sont actuellement générés par seconde en 2021, ce qui correspond à environ à 410 000 par jours.

Il est important de souligner que la pandémie de Covid-19 a également joué un rôle significatif dans l'augmentation du nombre de malwares en 2021. Avec la montée en flèche du travail à distance, les cybercriminels ont vu une opportunité de s'infiltrer dans les réseaux des entreprises grâce à des vulnérabilités liées à l'utilisation d'outils de travail, tels que les VPN, applications de visioconférence, etc. Les attaques de phishing ont également connu une forte augmentation, avec des emails qui prétendaient avoir des informations liées au Covid-19. Cette période de vulnérabilité globale a permis aux attaques par malwares de se multiplier rapidement.

Cependant, le nombre de malwares détectés ne reflète pas nécessairement la gravité des attaques en cours. Les attaquants peuvent utiliser des techniques sophistiquées pour contourner les systèmes de sécurité existants et accéder aux réseaux de leurs cibles sans être détectés. Ces attaques ciblées, également appelées "Advanced Persistent Threats" (APTs), sont devenues des menaces de plus en plus fréquente pour les entreprises et les gouvernements du monde entier. La guerre en Ukraine, par exemple, a été marquée par des cyberattaques ciblées de la part de groupes soutenus par des États, qui ont utilisé des malwares sophistiqués pour espionner et perturber les infrastructures critiques

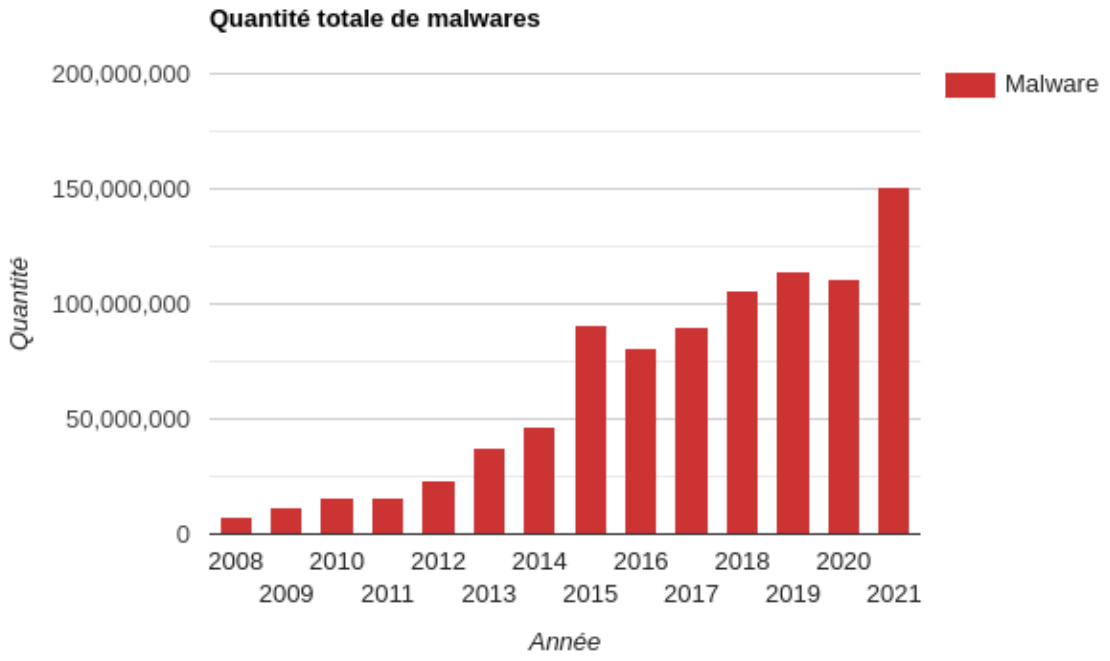

Figure 1: Quantité totale de malwares de 2008 à 2021 (AV Atlas, s.d.)

du pays. De telles attaques sont souvent conçues pour rester discrètes et indétectables pendant de longues périodes. Les solutions de détection de malware peuvent être utiles pour détecter les malwares courants, mais peuvent ne pas être suffisantes pour détecter les attaques ciblées les plus sophistiquées. Il est donc important de se demander si les solutions de détection de malware peuvent réellement nous protéger face à ce genre d'attaque.

Maintenant que nous avons abordé les défis posés par les attaques ciblées, il est important de comprendre comment les solutions de détection de malwares peuvent aider à faire face à ce type d'attaques. Dans la première partie de ce mémoire de recherche, nous étudierons les différentes techniques de détection de malwares utilisées de nos jours.

Cependant, il est important de comprendre que les malwares sont de plus en plus sophistiqués et qu'ils utilisent des techniques avancées de dissimulation pour échapper à la détection. Dans la deuxième partie, nous analyserons les techniques de dissimulation de malwares.

Enfin, dans la troisième partie de ce mémoire, nous étudierons la caractérisation d'une attaque ciblée sur le plan organisationnel, en examinant les différentes phases d'une attaque ciblée et les techniques utilisées par les attaquants pour compromettre les systèmes. Les mesures de sécurité organisationnelles qui peuvent être mises en place pour réduire les risques d'attaques ciblées et protéger les systèmes d'informations seront également abordés.

Ce mémoire de recherche vise à fournir une analyse détaillée de l'efficacité des solutions de détection de malwares dans la protection contre les attaques ciblées, en examinant les techniques de détection de malwares, les techniques de dissimulation de malwares et la caractérisation d'une attaque ciblée sur le plan organisationnel.

Avant d'entrer dans le vif du sujet, il est important de définir ce qu'est un malware. Malware provient de la contraction des mots "Malicious" et "Software", nous pouvons définir un malware comme étant un logiciel qui cherche à répondre aux intentions malveillantes d'un attaquant(Gadhiya et Bhavsar, 2013).

Les criminels utilisent les malwares pour accéder aux données sensibles, faire des demandes de rançons ou simplement causer le plus de dommages possibles aux systèmes qu'ils affectent. De manière générale, ils ont des objectifs variés. Quelques exemples des utilisations les plus courantes (Casey Crane, 2020) :

- Profiter financièrement (soit directement, soit par la vente de leurs produits ou services) . Par exemple, ils peuvent utiliser des logiciels malveillants pour infecter les appareils des cibles afin de voler des informations de compte ou de la crypto-monnaie. Ils peuvent vendre leurs logiciels malveillants à d'autres cybercriminels pour qu'ils les utilisent comme ils l'entendent ou peuvent les vendre en tant qu'offre de service;
- Comme moyen de vengeance. Par exemple, *Brian Krebs* de *Krebs on Security* a été frappé par une importante attaque DDoS en 2016 après avoir parlé de ce type d'attaque sur son blog;
- Pour mener à bien un agenda politique ou social. Des exemples parfaits de cela seraient les acteurs de l'État-nation (comme les groupes de hackers gérés par l'État en Chine et en Corée du Nord) et les groupes de hackers comme Anonymous;
- Comme un moyen de se divertir. Certains cybercriminels sont un peu comme des joyriders - ils choisissent de chasser l'ennui en victimisant les autres.

La prolifération des malwares est en grande partie due à la diversité des méthodes de diffusion utilisées. En général, les utilisateurs sont infectés lorsqu'ils ouvrent un fichier exécutable malveillant sur leur ordinateur, soit par inadvertance, soit par inattention.

- Phishing : L'envoi de messages de phishing (hameçonnage) est la méthode de diffusion de malwares la plus courante. Un fichier peut être joint au message ou transmis par le biais d'un lien Web qui amène l'utilisateur à télécharger le fichier depuis un emplacement de stockage en ligne(terranovasecurity, 2022).
- Clés USB : Il est fréquent que les cybercriminels procèdent en déposant des clés USB contenant des fichiers nommés de telle manière à inciter quiconque les trouvent à brancher la clé et à ouvrir le fichier, entraînant la diffusion du malware sur leur ordinateur(terranovasecurity, 2022).

- Sites web : Que ce soit via les réseaux sociaux ou un site légitime compromis, les sites Web peuvent être utilisés pour diffuser un malware sur les appareils d'internautes peu méfiants(terranovasecurity, 2022).
- Réseau : Type de malware capable de se propager sur d'autres appareils connectés au même réseau, au-delà de son objectif principal. Une fois installé au moyen d'une des autres méthodes citées précédemment, ce malware recherchera d'autres appareils à infecter(terranovasecurity, 2022).
- Exploitation de vulnérabilités : fragements de code conçus pour tirer parti d'une vulnérabilité ou d'une faiblesse de sécurité dans un logiciel ou un matériel pour ensuite infecter d'autres appareils(Joseph Regan et Ivan Belcic, 2022).

Les malwares peuvent prendre différentes formes et avoir des objectifs différents. Les types de malwares les plus courants sont :

- Rançongiciel (Ransomware) : qui détourne l'appareil de la victime, en lui refusant l'accès à ses fichiers jusqu'à ce qu'elle paie une rançon pour obtenir la clé de déchiffrement. Un exemple de ransomware est Lockbit qui est une variante de ransomware de type Ransom-as-a-Service (RaaS) qui attaque et chiffre rapidement les données de grandes organisations avant d'être détectée par les systèmes de sécurité et les équipes informatiques. Lorsque la rançon est payée, le gang LockBit partage les gains avec l'affilié qui dirige l'attaque(Ahmed Bahgat, 2022).
- Ver (Worm) : qui peut se propager rapidement et causer des dommages considérables. Les vers infectent souvent des réseaux entiers en utilisant chaque machine infectée pour en infecter d'autres. Un exemple de ver connu est MyDoom, qui a été conçu pour attaquer les entreprises en 2004 et qui a causé des pertes financières estimées à environ 20 milliards de dollars.(Help Net Security, 2004).
- Cheval de Troie (Trojan) : se fait passer pour des logiciels légitimes ou sont cachés dans des logiciels légitimes pour permettre aux cybercriminels d'accéder au système de l'utilisateur. Parmi les exemples populaires de chevaux de Troie, nous pouvons citer, FinFisher, Beast, Tiny Banker, Zeus, Netbus et Shedun(Ahmed Bahgat, 2022).
- Logiciel espion (Spyware) : conçu pour espionner. Il se cache en arrière-plan et collecte les données, y compris les mots de passe, la localisation GPS et les informations financières(Joseph Regan et Ivan Belcic, 2022).
- Logiciel publicitaire (Adware) : légèrement similaires aux logiciel espion car ils collectent aussi des informations tels que les activités de navigation. Cependant, il ne récupère pas les frappes au clavier et son but principal est de personnaliser les publicités de l'utilisateur. Certains sont moins intrusifs et demandent la permission de collecter des informations(Joseph Regan et Ivan Belcic, 2022).

- Logiciel malveillant sans fichier (fileless malware) : qui peut être difficile à détecter car il n'a pas besoin d'être stocké dans un fichier ou d'être installé directement sur un appareil. Au lieu de cela, il s'infiltre directement dans la mémoire et commence à exécuter des codes malveillants ou à extraire des données(Joseph Regan et Ivan Belcic, 2022). Une méthode courante pour d'utilisation de ce type de malware est par le biais de l'ouverture d'e-mails de spam qui permettent aux attaquants d'accéder directement à la mémoire de l'appareil. Selon une étude récente, le taux d'attaques de malware sans fichier a connu une augmentation de près de 900% au cours du dernier trimestre de 2020(pandasecurity, 2021).
- Rootkit : considéré comme l'un des malwares les plus dangereux. Il permet aux attaquants d'avoir un accès complet et de contrôler l'appareil infecté avec des privilèges élevés.

Ainsi, il est évident que les malwares sont devenus une menace omniprésente et en constante évolution dans le monde numérique actuel. La diversité et la sophistication des attaques posent des défis importants en matière de détection et de protection. Dans ce mémoire, nous explorerons en détails les solutions de détection de malwares, les techniques de dissimulation employées par les attaquants et les stratégies organisationnelles pour lutter contre les attaques ciblées. En comprenant ces éléments clés, nous chercherons à déterminer si les solutions de détection de malwares actuelles sont suffisamment efficaces pour nous protéger contre ces menaces, et quelles améliorations pourraient être apportées pour renforcer la sécurité dans un paysage numérique de plus en plus complexe. Avec cette introduction posant les bases de notre étude, passons maintenant à la première partie, où nous examinerons les différentes techniques de détection de malwares utilisées aujourd'hui.

Qu'est ce que ce que la detection de malware ? Quelles sont les méthodes ?

La détection de malware

Il existe principalement deux façons de faire de la détection de malware, avec des techniques sous-jacentes, l'analyse statique et l'analyse dynamique. La première méthode va examiner le malware sans exécuter le code tandis que la seconde va chercher à examiner le comportement de celui-ci en exécutant le code (Ö. Aslan et Samet, 2020). Une troisième façon, l'analyse hybride mélange les techniques statiques et dynamiques pour obtenir un résultat plus fiable et avoir un taux de vrai positif/negatif plus important.

Statique

Détection par signature

La détection par signature est un type de détection très fréquemment utilisé et de nombreuses solutions de protection en sont dotées. Une signature est une chaîne de caractère/binaire qui permet de savoir si un fichier est un malware, on peut comparer une signature à un bout d'ADN :

> La détection basée sur les signatures s'appuie sur les signatures (séquences/chaînes d'octets spécifiques et uniques extraites d'échantillons de logiciels malveillants) pour détecter la présence de fichiers malveillants dans un système. (Michèle Crepaldi, 2021, p. 38).

La création de signature est assez simple. Pour ce faire, il suffit d'extraire une ou plusieurs chaînes de caractère/binaire qui permettent d'identifier un malware distinctement d'un autre. L'extraction de signature provenant d'un malware est très souvent faite par des experts dans le domaine.

Cette technique montre néanmoins quelques inconvénients :

- Avec un nombre grandissant de malware cette technique devient de plus en plus chronophage.
- Elle n'est pas adaptée pour les malware obfusqués et les packers.
- Elle est incapable de détecter des familles de malware inconnue.
- Inéficace contre la plupart des malwares polymorphiques et les nouvelles versions de malwares connus qui réussissent souvent à échapper aux détections par signature.

Hachage (hash basique, fuzzy hash)

Une fonction de hachage est une fonction mathématique qui prend une chaîne de caractères ou un fichier en entrée et produit un code unique, généralement de longueur fixe appelée "condensat" ou

"hash". Les fonctions de hachage sont très souvent utilisées dans la sécurité informatique, car elle permet par exemple de vérifier l'intégrité des données, de stocker et retrouver rapidement des données en réduisant leur taille.

Pour être considérées comme efficaces, les fonctions de hachage doivent répondre à un ensemble de critères spécifiques. Ces critères comprennent(M., s. d.) :

- **Le déterminisme**, ce qui signifie que les résultats de la fonction doivent rester identiques quels que soient la personne qui l'utilise, le moment où elle est utilisée ou le nombre de fois où elle est utilisée, tant que le message de départ est le même.
- **L'efficacité**, ce qui signifie fournir des résultats immédiats sans ralentir les systèmes informatiques
- **La résistance aux attaques**, c'est-à-dire qu'il ne doit pas être possible de recréer le message d'origine à partir du résultat de la fonction, sauf en essayant tous les résultats possibles
- **L'aisance de distinction**, la fonction doit permettre de distinguer facilement deux messages similaires, même si cela n'est pas idéal pour la détection des logiciels malveillants, car il est alors difficile de détecter les degrés de similitude entre les messages.

Pour la détection de malware, il existe deux types de hash. Les hash dit traditionnels qui respectent les critères définis par Benjamin M., et les hash appelée "Fuzzy Hash" ou "Hachage flou" en français qui eux ne sont pas du tout basés sur les mêmes principes. Pour ces deux techniques, les hash vont être comparés avec une base de données de malwares. Ces bases de données peuvent être publiques comme être privées (propriétaires).

La différence principale entre les hash traditionnels et les fuzzy hash est que les hash traditionnels ne peuvent pas détecter les modifications mineures dans les fichiers en entrée. En revanche, les Fuzzy Hash sont conçus pour cela, ils sont utilisés pour détecter des fichiers similaires avec de légères modifications.

Les hash traditionnels sont plus rapide à calculer et prennent moins de ressources système, mais cette technique n'est pas souple. Il est très simple de modifier une toute petite partie du malware qui changera totalement le hash. L'avantage de cette technique est qu'il n'y a que très peu de faux positifs si ce n'est aucun. Les Fuzzy hash quant à eux sont beaucoup plus long et prennent plus de ressources système à générer, mais sont conçu pour détecter des changements mineurs aux fichiers en entrée, et peuvent donc être utilisés pour détecter des ressemblances entre les malwares et même entre les familles de malwares. Néanmoins, cette technique étant beaucoup plus souple, elle peut être amenée à ressortir des faux positifs.

Désassemblage / Décompilation

Le désassemblage et la décompilation sont deux autres méthodes de détection des malwares très utilisées pour la rétro-ingénierie approfondie d'un programme, nécessitant une analyse par la suite par un humain ou un programme spécialisé. Ces deux méthodes sont généralement utilisées de paire.

Le désassemblage consiste à traduire le code machine (0 et 1) du programme en code assembleur. Celui-ci étant lisible par un humain et traduisant très précisément ce que le programme réalise dans la mémoire ainsi que ses interactions avec le système d'exploitation et le processeur.

La décompilation consiste à traduire le code machine (0 et 1) du programme en code plus au niveau que l'assembleur (C, C++, Java, …). Cette méthode est utilisée pour une analyse plus rapide du code.

Cette méthode est souvent utilisée pour des besoins moins approfondis, car les compilateurs sont généralement moins précis sur la forme et la nature des données qu'ils renvoient (types inconnus, structures à créer, …), rendant parfois le code moins compréhensible que dans sa version désassemblée et nécessitant plus de travail de la part de l'analyste pour avoir un code fidèle à l'original.(program Transformation, s. d.)

Les désassembleurs les plus connus sont (Wikipedia, s. d.) :

- **GDB** (libopcodes et libbfd) : Debugger standard du projet GNU, il vient avec un désassembleur intégré qui est fréquemment utilisé pour l'analyse de malware
- **x64dbg** : debugger pour Windows intégrant son propre désassembleur, utilisé par les analystes sous Windows.
- **Radare2** : Framework libre créé pour et par des rétro-ingénieurs, il intègre son propre désassembleur lui aussi très fréquemment utilisé.
- **Rizin** : Fork de radare2 ayant pour objectif une accessibilité améliorée et d'avantages de fonctionnalités.
- **Ghidra** : Logiciel de rétro-ingéniérie open-source de la NSA. Il intègre son propre désassembleur.
- **IDA** : IDA pour Interactive Disassembler, c'est aussi un logiciel de rétro-ingéniérie cette fois-ci payant et propriétaire. Il intègre son propre désassembleur connu pour être particulièrement performant et supporter un grand nombre de formats d'exécutable et de processeur.

Les décompilateures les plus connus sont (Backerstreet, s. d.) :

- **Ghidra** : Logiciel de rétro-ingéniérie open-source de la NSA qui intègre un puissant décompilateur.
- **Reko** : Décompilateur pour Windows connu pour supporter de nombreux formats de fichier.
- **Hex Rays** : Décompilateur utilisé comme plugin dans IDA Pro. Il utilise les capacités de désassemblage de IDA pour fournir des résultats très concluants.

Les défis du désassemblage et de la décompilation sont l'obfuscation du code (dissimulation du code et du comportement), le chiffrement et les techniques anti-analyse (anti-virtualisation, anti-débogage,

leur but est d'échapper et d'agir différemment lorsque analysé). Ces méthodes sont utilisées en grandes quantités par les acteurs pour complexifier la tâche des analystes et nécessitent parfois l'utilisation de logiciels spécialisés.[Visser (s. d.)](P. Transformation, s. d.)

Il existe de nombreux exemples de décompilation réussie de logiciels malveillants, dans lesquels les analystes ont pu identifier et neutraliser des menaces avant qu'elles ne causent des dommages importants. Un exemple notable est le ver Stuxnet, découvert en 2010 et conçu pour cibler les systèmes de contrôle industriels.

Les analystes ont pu décompiler le ver et déterminer qu'il avait été conçu pour cibler des automates programmables industriels (API) Siemens spécifiques, utilisés, entre autres, dans les centrifugeuses nucléaires en Iran et pour la distribution d'eau potable. En exploitant les vulnérabilités des automates, le ver a pu affecter 45 000 systèmes informatiques, dont 30 000 situés en Iran et causer des dommages importants au programme nucléaire iranien.(Monitor, s. d.)

Modèles de langage naturels

L'analyse de code est un processus crucial pour assurer la qualité du logiciel. Elle permet de détecter les erreurs de programmation, les vulnérabilités de sécurité et les inefficacités du code. L'analyse de code peut être effectuée manuellement ou automatiquement. Cependant, avec l'avènement de l'intelligence artificielle (IA), l'analyse de code peut être effectuée de manière plus précise et efficace. Dans cet état de l'art, nous allons explorer comment l'IA peut être utilisée pour analyser le code.

Les modèles de langage naturel pré entraîné qui utilise la technologie de traitement du langage naturel (NLP) pour générer du texte. Ils sont entraînés sur des données de texte massives et peuvent être finement réglés pour répondre à des tâches spécifiques. Ils sont capables de comprendre le langage naturel et de produire des réponses cohérentes et précises en fonction des entrées de texte. Cette capacité peut être utilisée pour analyser le code.

Principaux modèles de langage naturel actuels utilisable pour de l'analyse de malware :

- **GPT3** (Generative Pre-trained Transformer 3) est un modèle de langage naturel avancé qui est capable de générer du texte en utilisant une grande quantité de données d'entraînement. GPT-3 peut être utilisé pour détecter les caractéristiques communes des codes malveillants et pour identifier les mots-clés et les expressions couramment utilisés dans les attaques.(Brown *et al.*, 2020);
- **GPT2** (Generative Pre-trained Transformer 2) est un modèle de langage naturel pré-entraîné qui est capable de générer du texte en utilisant des techniques de modélisation de langage profond. GPT-2 peut être utilisé pour analyser le code malveillant et détecter les anomalies ou les comportements suspects[(s. d.)](Radford *et al.*, 2019);

- **BERT** (Bidirectional Encoder Representations from Transformers) est un modèle de langage naturel développé par Google qui est capable de comprendre le contexte des mots dans une phrase. BERT peut être utilisé pour détecter les similitudes entre les différents morceaux de code malveillant et pour identifier les mots-clés et les expressions couramment utilisés dans les attaques [Turc *et al.* (2019)](Yesir et Soğukpinar, 2021).

L'analyse de code via un modèle d'intelligence artificielle commence par l'entraînement du modèle sur des exemples de code. Le modèle apprend les modèles et les structures du code et est capable de les reconnaître lorsqu'il est présenté avec de nouveaux exemples de code. L'analyse de code via IA peut être utilisée pour détecter les erreurs de syntaxe, les erreurs de logique, les vulnérabilités de sécurité et les incohérences dans le code.

L'une des principales utilisations de l'IA dans l'analyse de code est la correction automatique de code. Le modèle peut être entraîné à identifier les erreurs de syntaxe courantes et à proposer des corrections appropriées. Cela peut aider les programmeurs à économiser du temps et à améliorer la qualité du code. Elle peut également être utilisée pour détecter les erreurs de logique dans le code. Le modèle peut être entraîné à identifier les erreurs courantes de logique et à proposer des solutions pour les résoudre. Cela peut aider les programmeurs à améliorer l'efficacité et la précision de leur code.

En outre, les modèles d'IA peuvent être utilisés pour détecter les vulnérabilités de sécurité dans le code. Le modèle peut être entraîné à identifier les vulnérabilités courantes de sécurité et à proposer des solutions pour les résoudre. Cela peut aider à prévenir les attaques de sécurité et à améliorer la sécurité du logiciel.

Le code assembleur fourni est une représentation du code binaire qui sera exécuté par un processeur. Il s'agit du code assembleur de la fonction `main` d'un programme C compilé.

Voici une description étape par étape du code :

* `lea ecx,[esp+0x4]` : Charge l'adresse du premier argument passé à la fonction `main` (stocké dans `esp+0x4`) dans le registre `ecx`.
* `and esp,0xfffffff0` : Aligne la pile sur une limite de 16 octets en mettant à zéro les 4 bits les moins significatifs de l'adresse stockée dans le registre `esp`.
* `push DWORD PTR [ecx-0x4]` : Empile le premier argument de la fonction `main` sur la pile.
* `push ebp` : Sauvegarde la valeur du registre `ebp` sur la pile.
* `mov ebp,esp` : Copie l'adresse courante de la pile dans le registre `ebp` pour créer un nouveau cadre de pile (stack frame) pour la fonction `main`.
* `push ecx` : Sauvegarde la valeur du registre `ecx` sur la pile.
* `sub esp,0x14` : Alloue 20 octets d'espace sur la pile pour les variables locales de la fonction `main`.
* `mov DWORD PTR [ebp-0xc],0x0` : Initialise une variable locale de la fonction `main` à 0.
* `jmp 804842c <main+0x31>` : Saute à l'adresse `804842c` pour commencer la boucle principale du programme.
* `sub esp,0x8` : Alloue 8 octets d'espace sur la pile pour les arguments de la fonction `printf`.
* `push DWORD PTR [ebp-0xc]` : Empile la valeur de la variable locale sur la pile pour l'utiliser comme argument de la fonction `printf`.
* `push 0x80484c0` : Empile l'adresse de la chaîne de caractères "Hello, world!" sur la pile pour l'utiliser comme deuxième argument de la fonction `printf`.
* `call 80482d0 <printf@plt>` : Appelle la fonction `printf` avec les arguments empilés sur la pile.
* `add esp,0x10` : Dépile les arguments de la fonction `printf` de la pile.
* `add DWORD PTR [ebp-0xc],0x1` : Incrémente la variable locale de la fonction `main` de 1.
* `cmp DWORD PTR [ebp-0xc],0x13` : Compare la valeur de la variable locale de la fonction `main` à la valeur décimale 19 (0x13 en hexadécimal).
* `jle 8048415 <main+0x1a>` : Saut à l'adresse `8048415` si la comparaison précédente est vraie (c'est-à-dire si la valeur de la variable locale de la fonction `main` est inférieure ou égale à 19).
* `mov eax,0x0` : Met la valeur décimale 0 dans le registre `eax`.
* `mov ecx,DWORD PTR [ebp-0x4]` : Copie la valeur stockée dans le registre `ebp-0x4` dans le registre `ecx`.
* `leave` : Restaure la valeur de `esp`

Figure 2: Exemple d'utilisation de ChatGPT pour l'analyse de code en assembleur

L'utilisation de l'IA dans l'analyse de code peut permettre de considérablement augmenter la vitesse de résolution des vulnérabilités classiques. Elle peut aider à détecter les erreurs de syntaxe, les erreurs de logique, les vulnérabilités et les incohérences dans le code. De plus, le modèle peut être entraîné à proposer des corrections et des solutions pour améliorer la qualité et la sécurité du code. En fin de compte, l'utilisation de l'IA dans l'analyse de code peut aider les analystes en sécurité et les développeurs à économiser du temps et à améliorer l'efficacité de leur travail.2

Malheureusement elles comporte assez régulièrement des erreurs, ces modèles peuvent être un gain de temps certains pour des acteurs compétents dans le domaine visés, mais peuvent très rapidement induire en erreur des acteurs moins compétents.

Malware Features Extraction

N-grams Les n-grams sont une méthode de traitement de texte utilisée pour l'analyse de la langue naturelle, la reconnaissance de la parole et la classification de texte. Cette méthode découpe les textes en séquences de lettres ou de mots, ce qui permet d'analyser les fréquences et les motifs d'utilisation des termes dans un texte. Les n-grams sont couramment utilisés dans l'analyse de malware pour identifier les signatures de code malveillant et pour détecter des comportements malveillants.

Les n-grams sont simplement des séquences de n termes qui sont extraits d'un texte, où n est un entier positif. Par exemple, si n = 3, un texte serait divisé en séquences de trois mots consécutifs, appelés trigrammes. Si le texte est "Le chat noir est rapide", les trigrammes seraient "Le chat noir", "chat noir est", "noir est rapide". Les n-grams peuvent être utilisés pour identifier les schémas de comportement et les caractéristiques d'un texte, tels que la fréquence d'utilisation de certains termes, le nombre de termes différents et les modèles de syntaxe.

Malware dataset Peu importe le domaine, en recherche, il est important de disposer d'ensembles de données (datasets) pour pouvoir tester de nouvelles technologies ou de nouvelles méthodes. En ce qui concerne la recherche en cybersécurité, il existe un grand nombre d'ensembles de données disponibles. Toutefois, la plupart des ensembles de données existantes ne sont pas facilement accessibles à des fins de recherche et ne sont parfois pas dans des formats appropriés pour les utiliser sans traitement préalable.

Plusieurs grands ensemble de données de malware sont (Ö. A. Aslan et Samet, 2020) :

- NSL-KDD: un ensemble de données de réseau pour la détection d'intrusions publié en 2009. Il contient des données générées à partir de l'exploitation de vulnérabilités et des attaques DOS et des scans de port.

- Drebin: un ensemble de données de logiciels malveillants Android publié en 2014. Il contient plus de 120 000 échantillons d'applications Android malveillantes, y compris des chevaux de Troie, des vers et des spywares.
- Microsoft Malware Classification Challenge: un ensemble de données de logiciels malveillants publié en 2015. Il contient des données sur les caractéristiques des fichiers suspects, y compris les types de fichiers, les noms de fichiers, les tailles de fichiers et les horodatages.
- ClaMP (Classification of Malware with PE headers): un ensemble de données de logiciels malveillants publié en 2016. Il contient des données sur les caractéristiques des fichiers suspects, y compris les en-têtes PE, les caractéristiques d'importation et d'exportation et les informations sur les sections.
- AAGM (Android Adware and General Malware Dataset): un ensemble de données de logiciels malveillants Android publié en 2018. Il contient plus de 7 000 échantillons d'applications malveillantes, y compris des adwares, des chevaux de Troie et des spywares.
- EMBER dataset: un ensemble de données de logiciels malveillants publié en 2018. Il contient des données sur les caractéristiques des fichiers suspects, y compris les en-têtes PE, les fonctions d'importation et d'exportation, les informations sur les sections et les fonctions de désassemblage.

On retrouve aussi des données plus déstructurées, mais très souvent utilisées en recherche sur des bases de données communautaire ou indépendante telles que MalwareBazaar (abuse.ch, s. d.) ou VXUnderground (vx-underground, s. d.). Elles sont ici fournies par des chercheurs indépendants ou des entreprises et sont régulièrement mises à jour.

Dynamique

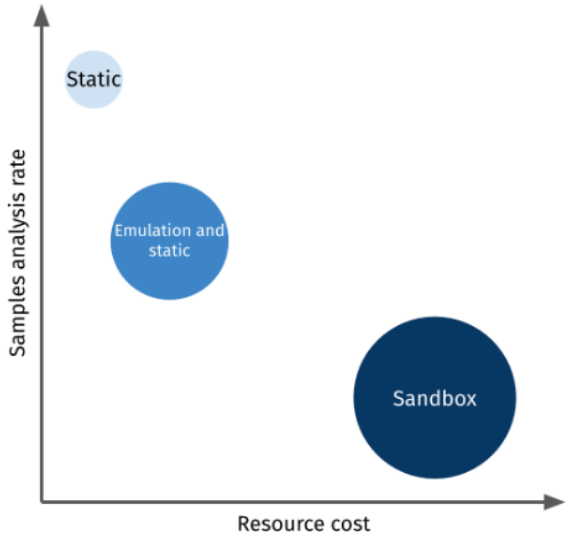

Figure 3: Ratio entre le coût et l'efficacité de l'analyse statique et dynamique.

L'analyse dynamique des programmes est aujourd'hui une composante clé de l'analyse de malware. Elle consiste à analyser le comportement d'un programme lorsque exécuté pour détecter s'il est potentiellement malveillant.3

Émulation

L'émulation est un élément essentiel de l'analyse dynamique des logiciels malveillants. L'émulation consiste à simuler l'exécution d'un code afin d'identifier son comportement dans un environnement contrôlé. En simulant l'exécution du code, les entreprises peuvent identifier les vulnérabilités potentielles et prendre des mesures proactives pour prévenir les infections.

L'émulation peut être utilisée pour analyser différents types de code, notamment les logiciels malveillants, les systèmes d'exploitation et les applications. En simulant l'exécution du code, les entreprises peuvent identifier les vulnérabilités potentielles et prendre des mesures proactives pour prévenir les infections.

Elle a comme principal avantage de ne pas être une réelle exécution et ne nécessite donc pas un environnement parfaitement clos, car toutes les opérations du programme seront simulées et émulées. Le malware n'enverra jamais aucunes données et ne réalisera jamais aucune action réelle sur le système.

Le coût en ressources de l'émulation est plus important que l'analyse statique pour une vitesse de

traitement plus faible. Le taux de vrai positif et sa capacité à détecter des familles de malware jusque là inconnues est quant à lui bien supérieur(Vouvoutsis *et al.*, 2022)

Exemple de données collectées par un émulateur

Pour les fichiers exécutables (Kaspersky, s. d.):

- Appel système
- Changement sur le système de fichier
- Changement dans les registres
- Changement dans les processus
- Requête le réseau effectué
- Dump de mémoire

Pour les scripts(Kaspersky, s. d.) :

- Arguments et retours des opérations sur des chaînes
- Appels de librairies et fonctions du système
- Changement sur le système de fichier
- Événements

Exécution

L'exécution, aussi appelée sandboxing, consiste en l'exécution du programme sur une machine virtuelle totalement isolé et contrôlée pour avoir un comportement le plus proche de la réalité possible. Le but de l'exécution est d'observer le comportement du programme dans le temps sans compromettre le système hôte ni le réseau. Cela permet par la suite d'avoir une analyse détaillée des actions du programme.

Une sandbox est purement et simplement une machine virtuelle classique avec des ressources limités et un accès restreint et durci au système hôte.

Un avantage très important de l'exécution est que l'obfucation devient beaucoup moins possible, toutes les opérations du programme vis à vis de la machine étant contrôlés, une chaîne obfusquée dans le code finira par apparaître déobfusquée au moment de son utilisation par le programme.

Le coût en ressources de l'émulation est plus important que l'analyse statique et l'émulation pour une vitesse de traitement plus faible. Le taux de vrai positif et sa capacité à détecter des familles de malware jusque là inconnues est encore supérieur à celui de l'émulation (Vouvoutsis *et al.*, 2022)

Sandbox les plus connues :

- **Cuckoo** : Sandbox Open-Source créée en 2010 pour le Google Summer of Code. Elle est une, si ce n'est là, sandbox la plus utilisée à ce jour;

- **CAPE** : Sandbox Open-Source dérivée de Cuckoo en 2016, offre des performances importantes et un grand nombre de fonctionnalités tout en restant accessible et automatisable en Python;
- **TRIAGE** : Sandbox propriétaire créée en 2018 par un ancien développeur de Cuckoo Sandbox, elle se veut plus moderne et performante que Cuckoo.

Exemple de données observée par une sandbox

- Modification dans le système de fichier
- Modification des registres
- Appel de librairies externes
- Liste des appels systèmes
- Création de Mutex (accès exclusif à des ressources partagées sur le système)
- Liste des processus
- Dump mémoire (système et de chaque processus)
- Captures d'écran
- Flux réseau (PCAP)

Hybride

Corrélation

> Corrélation : Relation existant entre deux notions dont l'une ne peut être pensée sans l'autre, entre deux faits liés par une dépendance nécessaire.(Larousse, s. d.)

La corrélation consiste à rechercher des liens ou des relations entre différents éléments de données qui peuvent aider à détecter ou à comprendre le fonctionnement d'un malware, elle est autant utilisée en analyse dynamique que en analyse statique(Han *et al.*, 2019). Par exemple en analyse dynamique, la corrélation peut être utilisée pour identifier des comportements anormaux d'un ordinateur, comme une utilisation excessive de la mémoire ou des entrées/sorties de disque anormales, qui peuvent être le signe d'une infection par un malware.

Pour une corrélation efficace, les analystes doivent collecter et analyser un grand nombre de données provenant de différentes sources, comme les journaux d'événements de l'ordinateur, les analyses de réseau et les échantillons de code malveillant. Ils utilisent ensuite des outils d'analyse de données pour rechercher des liens ou des relations entre ces données, afin d'identifier des modèles ou des tendances qui pourraient indiquer la présence d'un malware.

Cas d'usage La corrélation peut être utilisée de différentes manières, selon les objectifs de l'analyse[Alomari *et al.* (2023)](Han *et al.*, 2019)(Eskandari *et al.*, 2013). Par exemple :

- Identifier des comportements anormaux d'un ordinateur qui pourraient être le signe d'une infection par un malware.
- Analyser les méthodes utilisées par un malware pour se propager et établir des liens entre les différentes étapes de son cycle de vie.
- Établir des corrélations entre différents échantillons de code malveillant pour identifier des groupes ou des familles de malwares.
- Analyser les réseaux de commande et de contrôle utilisés par les auteurs de malwares pour communiquer avec leurs logiciels et comprendre comment ils fonctionnent.

Points forts Elle est particulièrement utile et efficace lorsque les analystes(Han *et al.*, 2019) :

- disposent d'un grand nombre de données provenant de différentes sources, comme les journaux d'événements de l'ordinateur, les analyses de réseau et les échantillons de code malveillant. La corrélation peut permettre d'identifier des liens ou des relations entre ces données qui pourraient être difficiles à repérer à l'aide d'autres méthodes d'analyse.

- cherchent à comprendre le fonctionnement d'un malware, en particulier son cycle de vie et ses méthodes de propagation. La corrélation peut aider à établir des liens entre les différentes étapes de ce cycle de vie et à comprendre comment le logiciel se propage.
- cherchent à identifier des groupes ou des familles de malwares.

Heuristique

Les heuristiques permettent la détection de morceaux de code ou de comportements pour pouvoir détecter de nouveaux virus jusqu'alors inconnus ou encore de nouveau variants.

Elles se divisent en deux parties majeures :

- les heuristiques dites statiques qui analysent le code-machine du programme pour trouver des schémas ou des symboles pouvant donner des informations sur ce qu'il fait, et donc pouvoir détecter s'il est suspect ou s'il a des points communs avec une famille de malware connue.
- les heuristiques dites dynamiques qui analysent le comportement du programme (dans une sandbox par exemple) pour détecter de la même manière un comportement suspect (appel de librairie souvent utilisée par les malwares, ouverture de terminaux, exécutions de commandes, création de tâches planifiées, etc.) et potentiellement pouvoir relier ce comportement à une famille de malware déjà connue pour détecter un variant.

Bien que moins communes, on retrouve aussi des heuristiques comportementales[ESET (s. d.)](BitDefender, s. d.), celles-ci sont une liaison de l'analyse heuristique statique et dynamique et de machine learning, elle se sert des résultats de l'analyse comportementale basée avec un modèle de machine learning pour pouvoir appliquer les heuristiques dynamiques à un programme n'ayant pas été exécuté en sandbox ou machine virtuelle ni émulé.

Les heuristiques sont très utilisées par les anti-virus pour leurs capacités à détecter de nouveaux virus et de pouvoir les identifier comme variants d'autres virus. Malheureusement, le principe des heuristiques est basé sur des règles, et donc de ce fait de l'expérience, ce qui fait que malgré leur grande capacité pour la détection de zero-day, des malwares plus complexes peuvent être considérés comme sain lorsqu'ils utilisent les techniques encore inconnues, car aucune règle leur correspondant n'auront été créées jusque-là.

Le taux de faux positif n'est pas non plus à oublier, à cause de la multiplication des règles et des techniques utilisées, de nombreux programmes sains peuvent correspondre à ces règles et être considéré comme potentiellement malveillants.[Malwarebyte (s. d.)](Mishra, 2013)

Des études suggèrent que les heuristiques dynamiques offrent de meilleurs résultats pour la détection de malware complexes et sophistiqués.

Fonctionnement des règles d'heuristiques Les règles d'heuristiques utilisent de nombreuses valeurs pour la détection de comportement suspect, entre autres :

- les appels systèmes ;
- les codes d'opération (Opcode);
- les n-grams;
- les graphes de flot de contrôle (control flow graph ou CFG);

```
rule Malware
{
   meta :
      description = " This is a mal"
   strings:
      $a1 = "LdrGetProcedureAddress <CreateThread> LdrGetProcedureAddress"
      $a2 = "GetSystemMetrics GetSystemMetrics GetSystemMetrics"
      $a3 = "FindResourceExW LoadResource FindResourceExW LoadResource"
      $a4 = "NtOpenFile NtOpenFile NtOpenFile  NtOpenFile NtOpenFile"
   condition:
      any of them }
```

Figure 4: Exemple de règle Yara

Les règles YARA sont un exemple de règles pouvant être utilisées pour l'analyse heuristique.4

Conclusion

La détection de malware est composée de nombreuses méthodes, plus efficace lorsque utilisée les unes avec les autres.

La difficulté de l'intégralité de ces méthodes est de réussir à passer outre les défenses que les acteurs malveillants mettent en place dans les malwares. Pour adapter les techniques et en trouver de nouvelles, il faut d'abord connaître ces techniques de défense et comprendre leur effet sur les méthodes de défenses actuelles.

Techniques de dissimulation des malwares

L'obfuscation de code est une pratique courante visant à transformer un programme pour le rendre difficile à comprendre tout en préservant son fonctionnement. Cette technique permet de rendre le code beaucoup plus difficile à analyser par les différentes solutions de sécurité existantes. L'obfuscation est utilisée dans plusieurs cas de figure, notamment pour protéger une propriété intellectuelle, ralentir les analyses ou rendre les solutions de détection de malwares moins efficaces : (Paul RASCAGNERS and Sébastien LARINIER, 2022)

- En protégeant leur savoir-faire, certaines sociétés cherchent à empêcher leurs concurrents de copier leurs produits ou de les utiliser illégalement sans payer de licence;
- Dans le cas de l'analyse de malware, ralentir l'analyse peut permettre au malware de rester fonctionnel plus longtemps;
- En obfusquant le binaire, les solutions de détection de malwares peuvent être moins efficaces, car le binaire modifié peut paraître différent et ne pas correspondre à une signature connue de comportement malveillant;
- Après obfuscation, le binaire est différent et ne correspond pas forcément à un comportement malveillant connu ou une signature connue.

De nombreuses techniques d'obfuscation existent pour rendre difficile l'analyse des solutions de détection de malwares. Cependant, aujourd'hui, les malwares sont capables de se cacher tout en exécutant des actions malveillantes, et des techniques d'évasion peuvent contourner les solutions de sécurité mises en place, permettant aux malwares de s'exécuter sur les systèmes informatiques sans être détectés. Ainsi, il est important de comprendre les techniques d'obfuscation pour mieux protéger les systèmes informatiques contre les attaques ciblées.

Obfuscation de code

L'obscurcissement est une technique qui rend généralement les programmes plus difficiles à comprendre (Arini Balakrishnan et Chloe Schulze, 2005), tant pour les humains que pour les outils automatiques. Pour ce faire, elle transforme un programme en une nouvelle version qui est structurellement différente et plus difficile à analyser tout en conservant les mêmes fonctionnalités que l'original. La nouvelle version du programme est donc dite équivalente, d'un point de vue informatique, à la version originale (Ilsun You et Kangbin Yim, 2010).

Au départ, cette technologie avait pour but légitime de protéger les droits de propriété intellectuelle des développeurs de logiciels, mais elle est maintenant également utilisée de manière abusive par les créateurs de logiciels malveillants pour éviter d'être détectée(Evgenios Konstantinou, 2008). Plus spécifiquement, les malwares utilisent souvent des techniques d'obfuscation pour évoluer vers de nouvelles versions qui sont encore plus complexes à analyser et à décortiquer, dans le but de tromper les solutions de détections. Les techniques d'obfuscation de code sont généralement divisées en deux sous-catégories :

- Obfuscation basée sur les données;
- Obfuscation basée sur le contrôle.

Cependant, les auteurs de logiciels malveillants combinent généralement ces deux types de techniques d'obfuscation de manière complexe et ingénieuse afin de renforcer la protection résultante (Bruce Dang; Alexandre Gazert; et Elias Bachaalany, 2014).

La qualité de l'obfuscation est évaluée sur 4 critères(Arini Balakrishnan et Chloe Schulze, 2005) :

- La puissance : le degré d'obfuscation qu'elle ajoute au programme;
- La résilience : quelle est la difficulté à casser pour un désobfuscateur automatique;
- Furtivité : comment le code obfusqué se fond dans le reste du programme;
- Coût : combien de charge de calcul il ajoute à l'application obscurcie.

Obfuscation basée sur les données

L'obfuscation basée sur les données est un type d'obfuscation qui vise à rendre plus difficile la compréhension d'un programme en altérant les données qu'il utilise. Nous allons aborder les techniques suivantes :

- Renommage des fonctions, variables, bibliothèques;
- Encodage et chiffrement des données.

Renommage des fonctions, variables, bibliothèques Le renommage des fonctions, variables et bibliothèques est une technique couramment utilisée dans l'obfusction de code. Elle consiste à donner des noms aléatoires aux fonctions, variables et bibliothèques utilisées dans le programme afin de rendre leur compréhension plus difficile. Cette technique est souvent utilisée en parallèle d'autres techniques d'obfuscation pour renforcer la difficulté de l'analyse.

Le renommage des fonctions est généralement effectué en utilisant des noms aléatoires, souvent générés automatiquement par un outil d'obfuscation. Par exemple sur la figure ci-dessous, la fonction `myfunction` peut être renommée en `_i23fdfcnj` et la variables `mystring` en `_dfeakia1f92`. Les commentaires originaux peuvent être remplacés par des commentaires aléatoires, tels que `_dafaljfamfdn`. Cela peut rendre la compréhension du code plus difficile, mais il peut également rendre le débogage plus difficile et peut également augmenter la charge de calcul de l'application.(Wei Xu, Fangfang Zhang et Sencun Zhu, n.d.)

```
function myfunction(txt)
{
    alert(txt);
}
var mystring = "Hello World!";
myfunction(mystring);
```

```
function _i23fdfcnj (_fdji230fdj)//_32akfaj0ufa
{// _dafaljfamfdn
    alert ( _fdji230fdj) ;//_dkfahajkla13
}
var _dfeakia1f92 //_gcvdseaepk
= "Hello World!"; //_gpqkjk3424pkl
_i23fdfcnj(_dfeakia1f92);
```

Le renommage des fonctions, variables, bibliothèques, etc est l'une des technique qui fonctionne le mieux, en particulier lorsqu'elle est combinée avec d'autres techniques d'obfuscation.

Encodage et chiffrement des données L'encodage est une technique qui permet de transformer des données en un format différent, mais qui reste lisible pour la machine. L'objectif est de rendre les données moins compréhensibles pour un humain qui les observerait, mais sans pour autant les rendre inutilisables pour le programme qui les manipule. Le Base64 est par exemple couramment utilisé par les malwares.

Le chiffrement, quant à lui, consiste à transformer les données en utilisant une clé secrète afin de rendre les données illisibles pour quiconque n'a pas la clé de déchiffrement. L'objectif est de protéger les données contre des accès non autorisés. Le ROT-13 est un exemple de chiffrement par substitution

monoalphabétique qui est simple et réversible. Il remplace chaque lettre par celle située 13 rangs plus loin dans l'alphabet français/latin(dCode, n.d.).

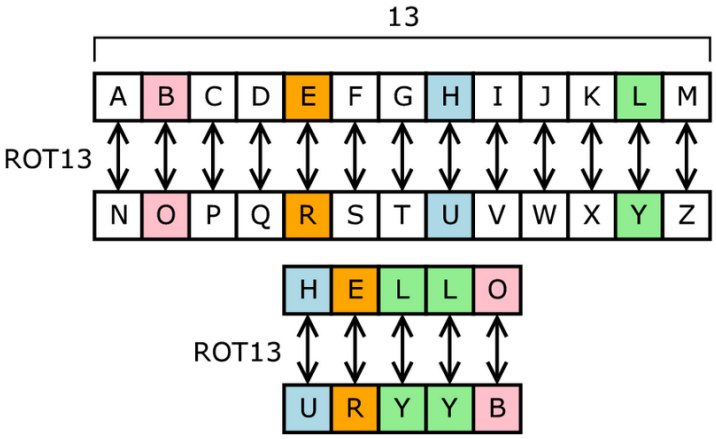

Figure 5: le mot "HELLO" qui devient "URYYB" avec l'algorithme ROT-13

Un exemple de malware utilisant le ROT-13 est le botnet appelé HerpesNet, dont le hash MD5 est `db6779d497cb5e22697106e26eebfaa8`. Les chaînes de caractères affichables du malwares sont encodées en ROT-13 pour éviter d'être facilement détectées par les solutions de détections(Université Paris-Est Marne-la-Vallée, n.d.). Ci-dessous un script *Python* qui passe toutes les chaînes de caractères affichables du malware qui étaient en ROT-13 à un format compréhensible par l'humain(malwarelu, 2012) :

```
$ python decode.py db6779d497cb5e22697106e26eebfaa8.exe
tcerfhygy : gpresultl
3.0 : 3.0
uggc://qq.mrebkpbqr.arg/urecarg/ : http://dd.zeroxcode.net/herpnet/
74978o6rpp6p19836n17n3p2pq0840o0 : 74978b6ecc6c19836a17a3c2cd0840b0
uggc://jjj.mrebkpbqr.arg/urecarg/ : http://www.zeroxcode.net/herpnet/
sgc.mrebkpbqr.arg : ftp.zeroxcode.net
uggc://sex7.zvar.ah/urecarg/ : http://frk7.mine.nu/herpnet/
hcybnq@mrebkpbqr.arg : upload@zeroxcode.net
hccvg : uppit
ujsdsdbbngfgjhhuugfgfujd : hwfqfqooatstwuuhhtstshwq
rffggghooo : essttubbb
Ashfurncsmx : Afusheapfzk
```

D'autres techniques de chiffrement plus avancées, telles que l'utilisation de XOR avec clé statique, XOR

avec clé dynamique,RC4,RSA, AES, etc peuvent être utilisés pour obfusquer le code(Paul RASCAGNERS and Sébastien LARINIER, 2022).

En résumé, l'encodage et le chiffrement des données font partie des techniques les plus utilisées pour masquer le code.

Obfuscation basée sur les structures de contrôle

L'obfuscation basée sur le contrôle est un type d'obfuscation qui vise à rendre plus difficile là l'analyse d'un programme en altérant son comportement. Nous allons aborder les techniques suivantes :

- Ajout de code inutile;
- Réaffectation de registre;
- Control-flow flattening.

Ajout de code inutile L'insertion de code mort est une technique simple qui consiste à ajouter des instructions inefficaces à un programme pour en modifier l'apparence, mais en conserver le comportement (Arini Balakrishnan et Chloe Schulze, 2005) (Wing Wing et Mark Stamp, 2014) (Evgenios Konstantinou, 2008). Un exemple de telles instructions est nop. Les figures 6 et 7a montrent que le code original est facilement obfusquer par l'insertion d'instructions nop. Cependant, les scanners antivirus basés sur les signatures peuvent déjouer cette technique en supprimant simplement les instructions inefficaces avant l'analyse. Par conséquent, afin de rendre la détection plus difficile, les logiciels malveillants ajoutent de la complexité au code en effectuant des opérations inutiles en manipulant le registre EDX, qui est un registre utilisé par les processeurs x86 pour stocker des données temporaires lors de l'exécution d'opérations, comme le montre la figure 7b(Ilsun You et Kangbin Yim, 2010).

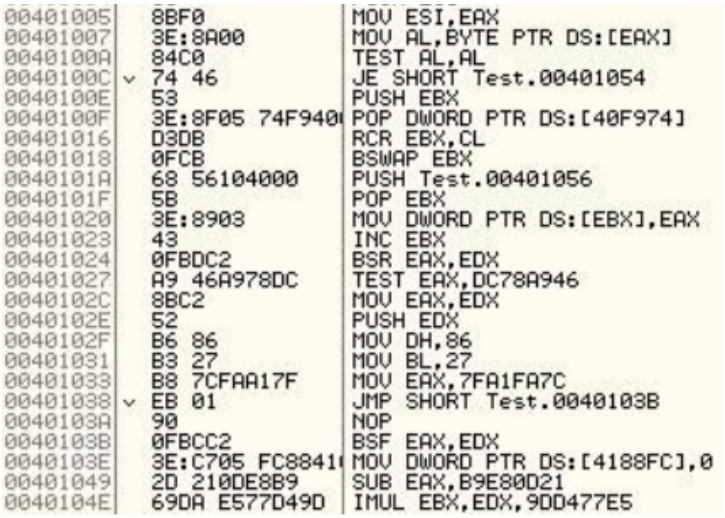

Figure 6: Un exemple de code

(a) Insertion du code mort dans le code d'exemple **(b)** Ajout de code qui ne fait rien dans le code d'exemple

Figure 7: Code inutile

Réaffectation de registre La réaffectation de registre est une technique qui consiste à modifier l'utilisation habituelle des registres dans le code source. Les registres sont des emplacements de stockage temporaire utilisés par le processeur pour stocker des données et effectuer des calculs. En réaffectant les registres, le code source devient plus difficile à comprendre et à suivre pour l'analyse.

Lorsqu'un compilateur génère du code assembleur, il utilise des registres pour stocker des valeurs temporaires et des résultats intermédiaires. Cependant, ces registres sont limités en nombre, ce qui signifie que le compilateur doit régulièrement déplacer des valeurs entre les registres et la mémoire pour éviter les collisions(Ilsun You et Kangbin Yim, 2010).

La réaffectation de registre consiste à modifier le code assembleur pour réutiliser les registres de manière inattendue. Cela rend le code plus difficile à comprendre et à déboguer, car les valeurs peuvent apparaître dans des registres différents à chaque fois qu'elles sont utilisées(Ilsun You et Kangbin Yim, 2010).

Un exemple serait de modifier une boucle pour utiliser des registres différents à chaque itération. Code de base :

```
MOV ECX, 10
MOV EAX, 0
BOUCLE:
ADD EAX, 1
LOOP BOUCLE
```

Cela utilise la boucle BOUCLE pour ajouter 1 à EAX 10 fois. Cependant, si nous réaffectons les registres à chaque itération, cela peut ressembler à cela :

```
MOV ECX, 10
MOV EAX, 0
BOUCLE:
ADD EBX, 1
ADD EAX, EBX
XOR EBX, EAX
LOOP BOUCLe
```

Maintenant, nous avons utilisé EBX pour stocker la valeur intermédiaire de la boucle. À chaque itération, nous ajoutons 1 à EBX, ajoutons EBX à EAX, puis écrasons EBX avec la valeur actuelle de EAX. Cela rend le code plus difficile à comprendre, car les registres sont réutilisés de manière inattendue.

Control-flow flattening Le “control-flow flattening” est une technique qui transforme la structure du code pour la rendre plus difficile à comprendre. Contrairement à la réaffectation de registre, cette technique modifie la structure du code plutôt que son exécution.

L’exemple présenté dans la figure 8 montre la transformation de boucle en *blocs de base* (partie de code qui est exécuté de manière continue, sans qu’il n’y ait de saut de programme entre les instructions), cela ne signifie pas simplement qu’on divise la *tête de la boucle*(conditions de sortie de la boucle) de son corps. Il faut changer la structure de la boucle pour que cela fonctionne correctement. Si on garde les mêmes fonctions (`while`, do ou `for`) dans le code aplati, cela donnera un résultat incorrect car une tête de boucle avec son corps détaché ne peut pas reproduire le comportement d’origine. Pour que cela fonctionne, on doit remplacer la tête de la boucle par une instruction `if`. Le prédicat est conservé à partir de la structure d’origine et les branches(instructions qui permettent de diriger le flux d’exécution du programme dans différentes directions) assurent le flux de contrôle correct en attribuant des valeurs appropriées à la variable de contrôle. Cela signifie que nous devons nous assurer que la boucle fonctionne de la même manière qu’avant la transformation(Timea Laszlo et Akos Kiss, n.d.).

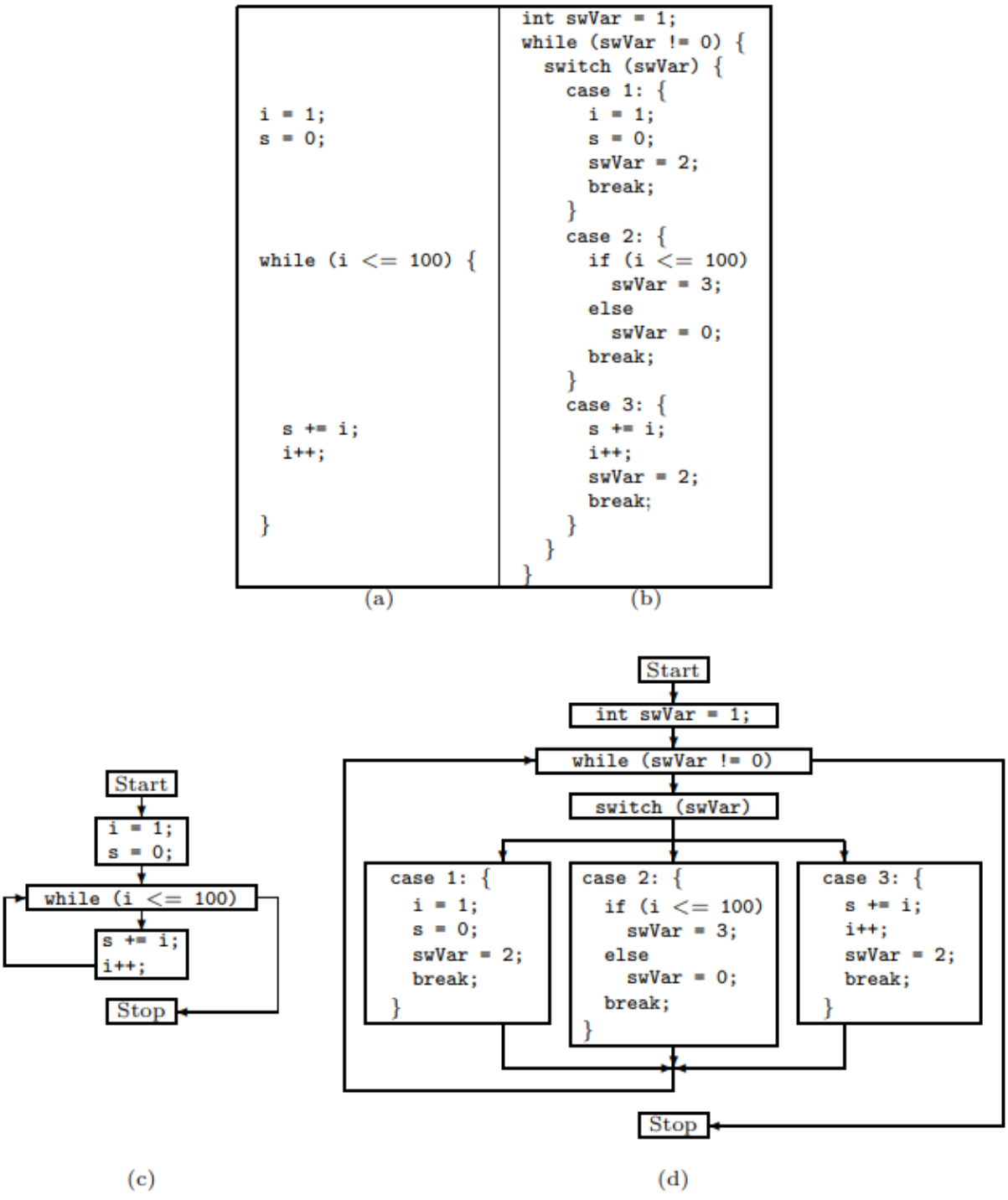

Figure 8: L'effet du "controle-flow flattening" (a: origine, b: aplati) et sur le graphe de flux de contrôle (c: origine, d: aplati)(Timea Laszlo et Akos Kiss, n.d.)

Packers

Les packers sont couramment utilisés dans les logiciels malveillants. Ils sont utilisés pour rendre l'analyse des malwares plus difficiles. Un packer permet de chiffrer, compresser ou encoder un binaire tout en préservant son fonctionnement. Il existe plusieurs types, qui ont différentes utilités, certains sont connus pour rendre difficile la rétro-ingénierie, alors que d'autres ne sont utilisés que pour compresser les binaires pour les rendre plus léger.

L'analyse d'un malware protégé par un packer nécessite la récupération ou la restauration du binaire original qui est dans le binaire packagé, cette opération s'appelle l'unpacking. Quelquefois, les malwares utilisent plusieurs couches de packers comme des poupées russes, il faut donc les déballer couche par couche jusqu'à obtenir le binaire de départ.

Le fonctionnement de chaque packer varie, mais dans la plupart des cas, le binaire original est chargé en mémoire (entièrement ou partiellement), l'objectif de l'analyse étant de le localiser et de l'extraire. Cela s'applique aux packers qui utilisent la zone mémoire pour stocker le binaire original après décompression.(Paul RASCAGNERS and Sébastien LARINIER, 2022)

Selon le papier "Generic Packing Detection using Several Complexity Analysis for Accurate Malware Detection", plus de 80% des malwares semblent utiliser des packers. 50% d'entre eux sont des versions reconditionnées de malware déjà existants(voir la figure ci-dessous). Ce papier explique qu'il y a eu 20 000 échantillons de malwares collectés en avril 2008 venant de 150 familles de malwares différentes. La classification des types de packer basé sur les signatures devient de plus en plus compliqué, car le nombre de nouveaux packer est en constante évolution, environ 15 par mois en 2009 (H., 2009).

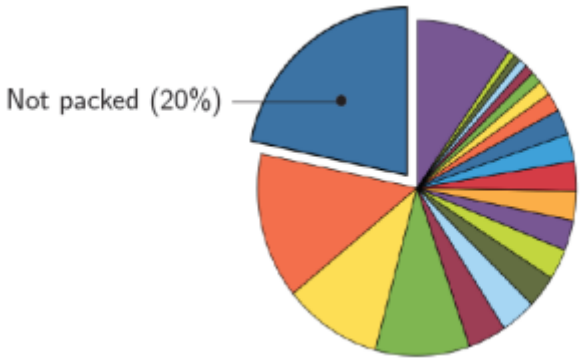

Figure 9: 80% des nouveaux malwares utilisent des packers, 50% d'entre eux sont des versions reconditionnées de malwares déjà existants(Dr. Mafaz Mohsin Khalil Al-Anezi, 2014).

Types de packers

Il existe différents types de packers utilisés par les malwares. Chacun d'entre eux a sa propre méthode pour cacher les codes malveillants. Les principaux types sont les suivants(Alexandre D'Hondt, 2023) :

- **Compression** : se concentre sur la réduction de la taille du binaire pour une distribution plus facile. La méthode de compression utilisée dépend du packer, mais le but est toujours le même : rendre le fichier plus petit(Michele Crepaldi, 2021). Le packer de compression le plus connu est UPX;
- **Chiffrement** : se concentre sur la protection des codes malveillants en les chiffrant. Le but est d'empêcher les solutions de détection de malware de détecter les codes en les chiffrant(Michele Crepaldi, 2021). Les deux packers de chiffrement les plus connus sont FSG et PESpin;
- **Protection** : combine les caractéristiques de compression et de chiffrement. Ils utilisent également des techniques anti débogueur et anti vm que nous verrons par la suite(Michele Crepaldi, 2021). Deux exemples de packers de protection : Aramdillo et Themida;
- **Bundlers** : permet de regrouper plusieurs fichiers en un seul pour une distribution plus facile. Les fichiers peuvent être des binaires exécutables, des scripts, des images, etc.(Michele Crepaldi, 2021). Les bundlers sont souvent utilisés pour distribuer des malwares en les masquant comme des applications légitimes. Deux exemples de packers bundler : Cameyo et ThinApp;
- **Virtualisation** : embarque une machine virtuelle qui permet de virtualiser les instructions de l'exécutable. Cela rend le code malveillant plus difficile à analyser car il est exécuté dans un environnement virtuel(Ashu Sharma et S. K. Sahay, n.d.). Exemples de packers de virtualisation : VMProtect et Enigma Virtual Box;
- **Mutation** : comprends les binaires oligomorphes, polymorphes et métamorphes. Les malwares oligomorphes sont capables de muter leur code en changeant légèrement leur structure et en utilisant différentes clés de chiffrement à chaque fois qu'ils se propagent. Les malwares polymorphes utilisent également des techniques de mutation pour éviter la détection, mais leur code source reste le même. Les malwares métamorphes sont encore plus avancés, car ils sont capables de générer un code malveillant entièrement nouveau à chaque fois qu'ils se propagent, rendant la détection presque impossible(Ashu Sharma et S. K. Sahay, n.d.);

Fonctionnement d'un packer

Le processus de compression des binaires est généralement effectué en deux étapes :

1. Le packer analyse le binaire original et identifie les parties du code qui peuvent être compressées. Les parties de codes inutiles, telles que les espaces vides, sont supprimées. Le packer crée ensuite une section de données supplémentaire dans le fichier, dans laquelle il stocke le code compressé et toutes les informations nécessaires pour le décompresser.
2. Ensuite, il modifie le point d'entrée du binaire pour qu'il pointe vers la section de données contenant le code compressé. Lorsque le binaire est exécuté, le point d'entrée redirige l'exécution vers la section de données contenant le code compressé. Le packer décompresse ensuite le code et le stocke dans une zone de mémoire réservée à cet effet. Une fois le code décompressé, l'exécution est redirigée vers le point d'entrée d'origine, à partir duquel le code malveillant peut s'exécuter normalement.

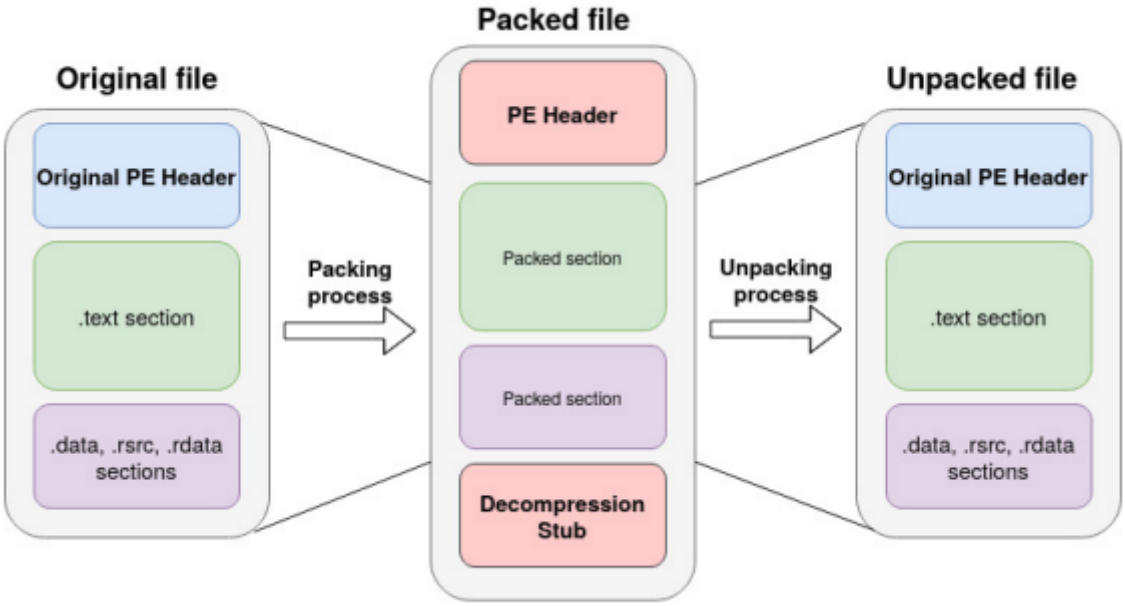

Figure 10: Représentation du cycle de vie d'un binaire packagé(Charles-Henry Bertrand Van Ouytsel, Thomas Given-Wilson, Jeremy Minet, Julian Roussieau et Axel Legay, 2021)

Certains packers peuvent également inclure des techniques d'obfuscation pour rendre le code malveillant encore plus difficile à détecter. Ces techniques incluent l'utilisation de sauts indirects (des instructions qui transfèrent le contrôle d'exécution à une adresse déterminée dynamiquement), de tables de sauts (des structures de données contenant des adresses vers lesquelles le programme peut sauter) et de code auto-modifiant (du code qui modifie sa propre structure ou son comportement pendant l'exécution). Ces méthodes peuvent rendre l'analyse du code malveillant plus complexe pour les outils de détection de malwares.

Nous pouvons d'ailleurs observer ci-dessous la différence entre plusieurs Packers, allant du packer

simple au packer plus avancé.

- Le simple packer (à gauche) va juste ajouter une couche de chiffrement;
- Le deuxieme “malware” va ajouter plusieurs couches de chiffrement qui rendent plus difficile l’analyse;
- Le dernier, quant à lui, va combiner plusieurs techniques comme des techniques de chiffrement avancée, anti débogage…

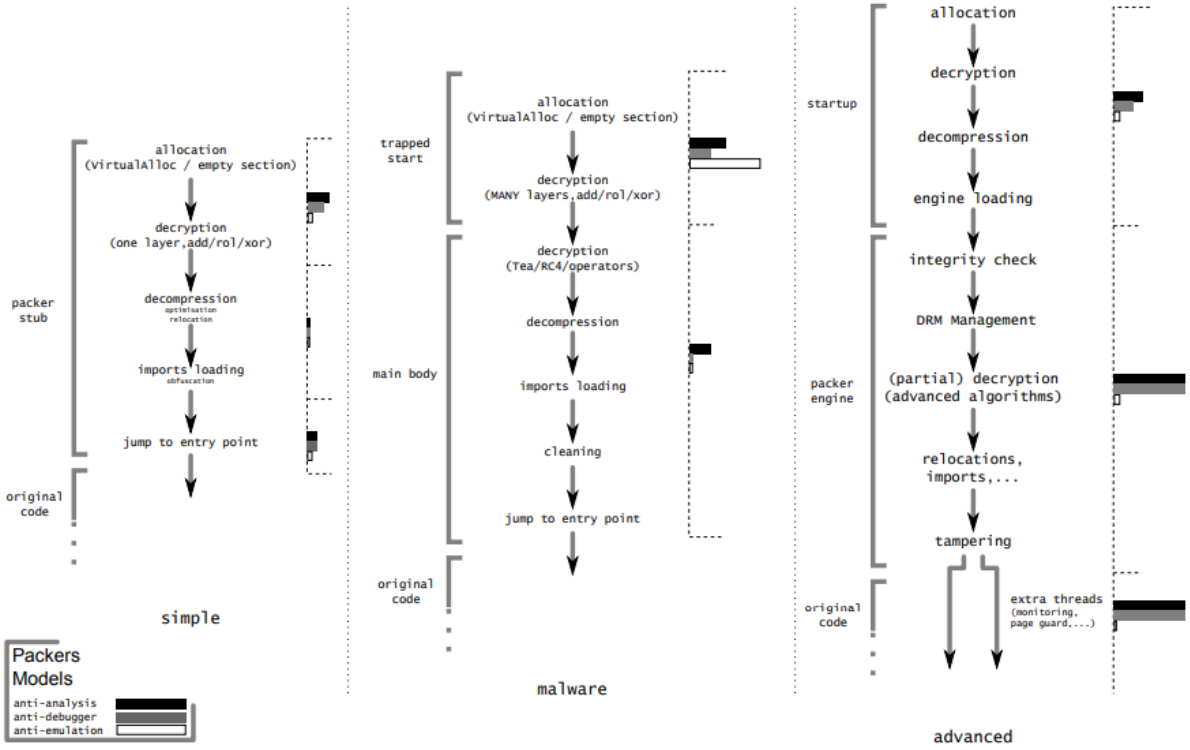

Figure 11: Modèle de packer(Ange Albertini, 2010)

En outre, certains packers peuvent utiliser des techniques de chiffrement pour protéger le code malveillant. Dans ce cas, le code est chiffré avant d’être stocké dans la section de données supplémentaire. Lorsque le binaire est exécuté, le packer déchiffre le code et le stocke en mémoire, à partir duquel il peut être exécuté normalement.

En somme, les packers sont des outils qui permettent de masquer du code malveillant dans des binaires apparemment légitimes, ce qui complique la détection de malwares.

Techniques Anti débogueur

Les débogueurs sont des outils essentiels pour les développeurs pour trouver et corriger les erreurs dans le code ainsi que pour l'analyse de malwares, qui permet de comprendre leur fonctionnement.

Lorsqu'ils sont utilisés sur des malwares, certains programmes malveillants peuvent utiliser des mécanismes d'évasion pour rendre leur détection et leur analyse plus difficiles. Lorsqu'une application est en cours de débogage, elle laisse souvent des traces ou des artefacts dans l'environnement d'exécution. Cette détection d'artefacts est un peu similaire à celle utilisée pour la partie Anti-VM que nous allons voir par la suite. Si un malware détecte la présence d'un débogueur, il peut modifier son comportement pour se désengager ou se dissimuler.

Détection de débogueur par le malware

Utilisation de l'API système L'API système (Application Programming Interface) est un ensemble de fonctions et de procédures fournies par le système d'exploitation pour permettre aux applications d'interagir avec celui-ci. Ils peuvent utiliser l'API système pour détecter la présence d'un débogueur. Les fonctions de l'API système Windows les plus utilisées :

- `IsDebuggerPresent` : permet de déterminer si l'exécution se fait dans un débogueur. Cette fonction retourne `true` si un débogueur est présent et `false` s'il n'y en a pas;
- `CheckRemoteDebuggerPresent` permet de déterminer si l'exécution est attachée à un débogueur distant;
- `NtQueryInformationProcess` permet d'obtenir des informations sur le processus en cours d'exécution, y compris si un débogueur est associé au processus;
- `OutputDebugString` permet d'envoyer des chaînes de caractères à un débogueur.

Voici une fonction simple qui permet de terminer l'exécution du processus si un débogueur est présent (Basatwar, 2022) :

```
#include <windows.h>

int main()
{
    if (IsDebuggerPresent())
    {
        ExitProcess(0);
    }
    return 0;
}
```

PEB Le PEB (Process Environment Block) est une structure de données qui contient des informations sur l'environnement dans lequel le processus est en cours d'exécution(Paul RASCAGNERS and Sébastien LARINIER, 2022). Il contient notamment beaucoup de flag qui permettent la détection d'un débogueur.

Une méthode couramment utilisée consiste à récupérer le flag `BeingDebugged` à partir du PEB. Un exemple de code assembleur qui permet cela :

```
mov eax, fs:[0x30] ; Récupère le pointeur vers le PEB
movzx eax, byte ptr [eax+0x2]; Récupère le flag BeingDebugged
```

```
lkd> dt nt!_PEB
   +0x000 InheritedAddressSpace : UChar
   +0x001 ReadImageFileExecOptions : UChar
   +0x002 BeingDebugged    : UChar
   +0x003 BitField         : UChar
   +0x003 ImageUsesLargePages : Pos 0, 1 Bit
   +0x003 IsProtectedProcess : Pos 1, 1 Bit
   +0x003 IsImageDynamicallyRelocated : Pos 2, 1 Bit
   +0x003 SkipPatchingUser32Forwarders : Pos 3, 1 Bit
   +0x003 IsPackagedProcess : Pos 4, 1 Bit
   +0x003 IsAppContainer   : Pos 5, 1 Bit
```

Figure 12: PEB qui contient le flag BeingDebugged(hasherezade, 2021)

Le flag `BeingDebugged` est un byte qui vaut `0x01` si le processus est en cours de débogage, sinon il sera égale à `0x00`.

Le flag `NtGlobalFlag` (ou `NtGlobalFlag2`, qui est la version plus récente) est également présent dans le PEB, et peut être utilisé pour détecter un débogueur. Ce flag est un entier qui contient plusieurs bits d'information, dont un qui indique si le processus est en cours de débogage. Si un débogueur est présent dans le processus, la valeur de `NtGlobalFlag` sera égale à `0x70`(hasherezade, 2021). Cette valeur peut varier en fonction des options de débogage activées et du système d'exploitation utilisé. Un exemple de code assembleur qui permet cela :

```
mov eax, fs:[0x30] ; Récupère le pointeur vers le PEB
movzx eax, byte ptr [eax+0x68]; Récupère le flag NtGlobalFlag
```

En utilisant ces informations, il est donc possible de détecter la présence d'un débogueur de manière plus furtive que l'utilisation de l'API système.

Vérification temporelle La vérification temporelle est une méthode simple et efficace pour découvrir si une application peut être sous le contrôle d'outils d'analyse. La vérification du temps est souvent

```
+0x04c ReadOnlySharedMemoryBase : 0x7feb0000 Void
+0x050 HotpatchInformation : (null)
+0x054 ReadOnlyStaticServerData : 0x7feb0720  -> (null)
+0x058 AnsiCodePageData : 0x7ffb0000 Void
+0x05c OemCodePageData  : 0x7ffc0228 Void
+0x060 UnicodeCaseTableData : 0x7ffd0650 Void
+0x064 NumberOfProcessors : 8
+0x068 NtGlobalFlag     : 0x70
+0x070 CriticalSectionTimeout : _LARGE_INTEGER 0xffffe86d`079b8000
+0x078 HeapSegmentReserve : 0x100000
+0x07c HeapSegmentCommit : 0x2000
+0x080 HeapDeCommitTotalFreeThreshold : 0x10000
+0x084 HeapDeCommitFreeBlockThreshold : 0x1000
+0x088 NumberOfHeaps     : 1
```

Figure 13: PEB qui contient le flag NtGlobalFlag(hasherezade, 2021)

implémentée à l'aide de l'instruction RDTSC (Read Time-Stamp Counter). Cette instruction stocke la valeur d'un compteur dans les registres EDX et EAX. Ils peuvent utiliser cette instruction pour mesurer le temps qu'il faut pour exécuter une partie de leur code. Le temps doit être mesuré au moins deux fois, et comparé à un seuil. Si le temps mesuré est supérieur au seuil, le malware peut supposer qu'il est en cours de débogage et modifier son comportement en conséquence(Michele Crepaldi, 2021).

Résidus d'anciennes analyses Les résidus de système des analyses précédentes sont également utilisés. Les entrées de registre peuvent être examinées pour déterminer si un débogueur a été utilisé sur le système tel que (Sikorski et Honig, 2012) :

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug`

Modification du comportement du malware en présence d'un débogueur

Les malwares peuvent perturber le fonctionnement du débogueur en injectant du code ou en créant des threads pour perturber les processus en cours d'exécution. Par exemple, un malware peut créer un thread qui effectue continuellement des opérations de calcul intensif, ce qui peut rendre difficile la visualisation des processus par débogueur. De plus un malware peut injecter du code pour masquer ou modifier ses activités lorsqu'il est sous le contrôle du débogueur.

Une autre technique nommé "TLS Callbacks" (Thread Local Storage Callbacks) est aussi utilisée pour masquer leurs activité lorsqu'ils sont sous le contrôle du débogueur. Les TLS Callbacks sont des

fonctions définies par l'utilisateur qui sont appelées lorsqu'un thread est créé. Ils peuvent utiliser ces fonctions pour effectuer des actions telles que le masquage de leur activité ou la modification de leur comportement lorsqu'ils sont sous le contrôle du débogueur. Par exemple, un malware peut utiliser un TLS Callback pour masquer certaines variables ou zone mémoire qui seraient normalement visibles par le débogueur(Sikorski et Honig, 2012).

Les exceptions sont une autre technique fréquemment utilisée, elles permettent de déclencher des exceptions intentionnellement pour perturber le débogueur. Par exemple, un malware peut déclencher une exception pour perturber le débogueur pendant qu'il effectue une opération importante, comme l'exécution du code malveillant ou le transfert de données sensibles(Sikorski et Honig, 2012).

Les interruptions sont des instructions de bas niveau utilisées pour contrôler les instructions et les opérations du processeur. Par exemple, elles sont utilisées pour suspendre temporairement le processus en cours d'exécution, ce qui peut rendre difficile la visualisation du processus par le débogueur(Sikorski et Honig, 2012).

Enfin, les malwares peuvent exploiter des vulnérabilités présentes sur les débogueurs pour échapper à la détection. Cela peut inclure l'exploitation de failles de sécurité dans le débogueur lui-même.

Les techniques de détection de débogueur sont nombreuses et variées, allant de l'utilisation de l'API système aux méthodes plus discrètes avec le PEB. Cependant, il existe également des outils tels que ScyllaHide, TitanHide et SharpOD qui permettent de masquer le débogueur et de rendre la détection plus compliquée, par exemple SharpOD est en capacité de cacher les flags liés au débogueur sur le PEB.

Ces outils permettent aux éditeurs de logiciels légitimes de protéger leur code contre l'ingénierie inverse, mais peut être utilisé aussi par les développeurs de malwares pour cacher leur code.

Techniques Anti VM

Les machines virtuelles (VM) sont énormément utilisées de nos jours pour l'analyse dynamique de malwares. Elles offrent un environnement isolé qui permet d'exécuter des échantillons de malwares de façon contrôlé. Cependant, les malwares modernes sont conçus pour détecter la présence de machine virtuelle, et donc pour éviter d'être détectés, ils utilisent des techniques de dissimulation que l'on appelle techniques Anti VM.

Ils existe un grand nombre de techniques Anti VM et peuvent inclure des méthodes de détection basées sur le comportement humain, les caractéristiques du système, l'environnement de la machine, etc… Avec ces techniques, les malwares ont la capacité de détecter la présence d'une machine virtuelle et de prendre des mesures pour éviter l'analyse, telle que la mise en sommeil.

On va étudier les différents mécanismes utilisés pour détecter que l'on est dans une machine virtuelle ainsi que les techniques de modification de comportement utilisées par les malwares.

Les composants physiques

Une techniques pouvant être utilisée par les malware pour détecter un environnement de virtualisation est d'utiliser les composants physiques. Les techniques les plus couramment utilisées :

- **Nombre de processeur** : les machines virtuelles ont souvent 2 coeurs ou moins, donc un malware peut vérifier que le nombre de coeurs du processeur est bien supérieur à 2 sinon il ne s'exécute pas;
- **Quantité de RAM** : si elle est inférieure à une certaines quantité ou bien même impaire, cela peut être interprété comme étant une machine virtuelle;
- **Taille du disque** : beaucoup de malwares utilisent la taille du disque pour détecter un environnement d'analyse. Si la taille du disque est inférieure à 64 Go par exemple, le malware peut arrêter son exécution. Cette technique est utilisée par les malwares, Azorult, Danabot, HawkEye, Ursnif, Agent Telsa et Lokibot;(Ayelen Torello and Francis Guibernau, n.d.)
- **Résolution de l'écran** : qui est aussi un facteur à prendre en compte, par exemple le malware Shark ne s'exécute pas si la résolution de l'écran est inférieure à 600 pixels, les sandbox n'ayant pas forcément le besoin d'une grande résolution d'écran ce malware arrivait à esquiver l'analyse des sandbox;(Siamesekitten, 2021)
- **Vérification thermique** : l'utilisation des capteurs thermiques pour savoir si le système est réel ou virtuel est également une technique utilisée. Si les informations thermiques ne sont pas disponibles ou bien avec une température anormalement élevée ou basse, il pourrait considérer qu'il est dans une machine virtuelle. Par exemple, le malware GravityRAT utilise cette technique pour éviter la détection;(Paul Rascagneres, 2018)

- **Vérification des périphériques** : des périphériques reliés tels que les claviers, souris ou bien imprimantes peuvent servir à déterminer si l'exécution se fait dans une machine virtuelle.

Caractéristiques du système

Les caractéristiques du système sont des informations qui sont accessibles à partir du système d'exploitation de la machine. Les malwares peuvent utiliser ces informations pour déterminer si le système est virtuel ou non. Il y a plusieurs types d'informations sur lesquelles se basent les malwares.

Les instructions systèmes Les instructions systèmes sont des commandes que le système d'exploitation utilise pour accéder aux informations sur le matériel et le logiciel de la machine.

Une de ces instructions est le `CPUID` qui permet de manière logicielle d'avoir des informations sur le processeur de la machine. Par exemple, "lorsqu'elle est exécutée avec `EAX = 1` en entrée, la valeur retournée renverra les caractéristiques du processeur. Le 31e bit de la valeur ECX sur une machine virtuelle sera égal à 0, tandis que sur une machine physique, il sera égal à 1"(Ayelen Torello and Francis Guibernau, n.d.).

Les requêtes Windows Management Instrumentation (WMI) permettent également d'obtenir des informations sur l'hôte, la liste des disques, fichiers/répertoires, etc.

Processus Il y a aussi les processus en cours d'exécution qui indiquent s'il s'agit d'une machine virtuelle ou non. Les outils de virtualisation comme VMWare, VirtualBox ou Xen ont souvent des processus en cours d'exécution sur les machines virtuelles qui donnent une indication. Par exemple, les processus qui ont ces noms permettent assez facilement de détecter une machine virtuelle(YUCEEL et Labs, 2022) :

- `vmware.exe`;
- `xenservice.exe`;
- `vmsrvc.exe`;
- `vboxservice.exe`;
- `joeboxserver.exe`;
- `prl_cc.exe`.

Clés de registre Les clés de registre sont des fichiers qui stockent des informations sur la configuration du système d'exploitation. Par exemple, ils peuvent vérifier la présence de sandbox à l'aide des clés de registre, comme la valeur qui fait référence à l'outil de virtualisation VirtualBox dans la clé de registre suivante (Shaul Vilkomir-Preisman, 2019) :

```
HKEY_LOCAL_MACHINE\SOFTWARE\Oracle\Virtual Box Guest Additions
```

Réseau Ils peuvent également collecter des informations niveau réseau comme l'adresse MAC (une adresse MAC commençant par `08:00:27` peut être associée à VirtualBox) ou le nom de l'adaptateur réseaux. Certains malwares, comme GoldMax, peuvent également vérifier que les adresses MAC de ses adaptateurs réseau ne correspondent pas avec l'adresse MAC de la sandbox Windows (`c8:27:cc:c2:37:5a`).(Ramin Nafisi and Andrea Lelli, 2021)

Empreinte du système L'environnement du système est aussi un paramètre à prendre en compte. Par exemple, un système d'exploitation fraîchement installé est plus susceptible d'être utilisé dans une machine virtuelle, alors qu'un système d'exploitation qui a été utilisé pendant une période conséquente est plus susceptible d'être utilisé par un vrai utilisateur. Les malware utilisent aussi une technique qui consiste à récupérer le hash du système et à le comparer avec une base de données de sandbox.(Ayelen Torello and Francis Guibernau, n.d.)

Évasion basée sur le temps

L'évasion basée sur le temps est sans doute la méthode la plus utilisée par les attaquants pour échapper à l'analyse de sandbox. Le principe est de retarder l'exécution réelle du malware en attendant par exemple un certain temps ou un événement spécifique. Les sandbox ayant un temps limité pour l'analyse cette méthode fonctionne bien. Voici les techniques les plus répandues :

- **Sommeil prolongé** consiste à retarder l'exécution en attendant une période avant de s'exécuter. L'analyse de la sandbox se termine avant que le malware ne s'exécute réellement(Deloitte Threat Library Team, Jeff Felling, Red Canary, Jorge Orchilles, SCYTHE and Ruben Dodge, 2021);
- **Bombe à retardement** utilise une condition de déclenchement basée sur la date. Il ne fonctionnera pas avant une certaine date, qui permet comme la méthode précédente de contourner les mécanismes d'analyses de sandbox(Deloitte Threat Library Team, Jeff Felling, Red Canary, Jorge Orchilles, SCYTHE and Ruben Dodge, 2021);
- **Événements du système** consiste à programmer le déclenchement du malware lorsqu'un événement particulier arrive sur la machine. Les déclencheurs sont souvent des événements qui ont très peu de chances de s'activer pendant l'analyse de la sandbox. Par exemple, l'événement en question pourrait être un utilisateur qui se déconnecte, l'arrêt ou redémarrage de la machine, etc… (Deloitte Threat Library Team, Jeff Felling, Red Canary, Jorge Orchilles, SCYTHE and Ruben Dodge, 2021)
- **Temps de fonctionnement du système** est basé sur la vérification du temps de fonctionnement du système. `GetTickCount()` , qui permet de déterminer depuis combien de temps un

ordinateur a été allumé. Souvent, lors d'une analyse dans une machine virtuelle, le temps est très court entre le démarrage et l'analyse ce qui permet aux malwares de détecter la machine virtuelle qui a généralement des temps de disponibilité réduit par rapport à une utilisation classique(Sikorski et Honig, 2012).

L'activité de l'utilisateur

Les techniques basées sur les activités de l'utilisateur sont conçues pour détecter un comportement humain normal, les malwares attendent une interaction particulière avant de commencer à s'exécuter.

L'intéraction utilisateur : L'utilisation du clavier et de la souris donne beaucoup d'indices sur l'activité de l'utilisateur. Les malwares sont en capacité d'utiliser les frappes de claviers, la vitesse des mouvements de la souris ainsi que la fréquence des cliques pour détecter si le malware est dans une machine virtuelle. Par exemple, le cheval de Troie Trojan.APT.BanChan(Dilshan Keragala, 2016) et le malware Okrum(Zuzana Hromcova, 2019) attendent quelques clics de souris avant de commencer à s'exécuter.

Il y a aussi l'attente d'interactions avec certaines applications comme des applications d'email, des navigateurs,etc…

La nécessité de devoir scroller ou appuyer sur un bouton pour que le malware s'active. Ces techniques sont souvent utilisées sur les documents Office. Par exemple, le malware FIN7 utilise cette technique, il va attendre que l'utilisateur double-clique sur une image avec des instructions comme sur la figure 14. En ajoutant une interaction de ce genre, ce malware est fait pour contourner la détection via sandbox.

Configuration du poste utilisateur Les malwares peuvent vérifier si le bureau de l'utilisateur est propre, si l'histoire des navigateurs n'est pas vide, le cache ainsi que les fichiers présents dans le répertoire courant de l'utilisateur.

Détection réseau

La détection réseau se base sur la présence ou non de certaines fonctionnalités ou présence d'activités pour déterminer s'ils sont dans un environnement isolé. Ci-dessous 3 techniques (Ayelen Torello and Francis Guibernau, n.d.):

- **Connexion disponible** : si aucune connexion n'est détectée, certains malwares ne s'exécuteront pas, car ils nécessitent une connexion pour télécharger des éléments supplémentaires ou communiquer avec le serveur de commande et de contrôle;

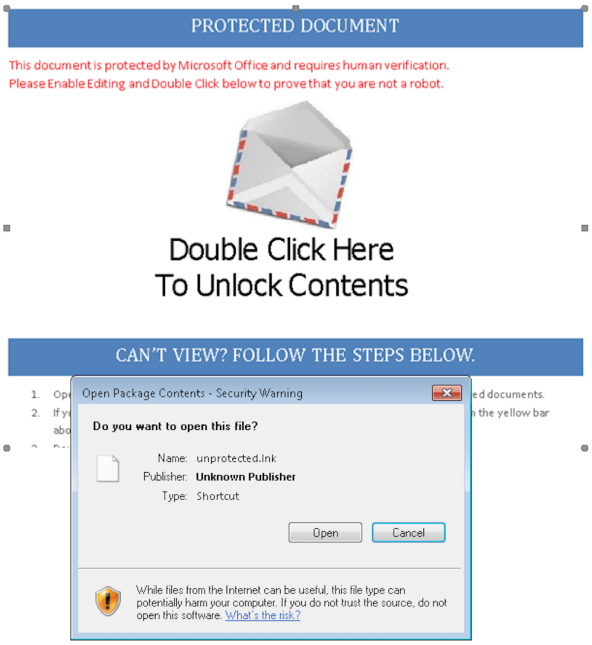

Figure 14: Leurre FIN7 malveillant demandant à la victime de double-cliquer pour déverrouiller le contenu

- **Ports ouverts sur la machine** : la vérification des ports ouverts est aussi une solution pour déterminer s'il est dans une sandbox, souvent certains ports étranges sont ouverts pour pouvoir communiquer avec le serveur principal de la sandbox;
- **Scan réseau** : s'ils ne détecte qu'un seul ordinateur ou qu'un petit nombre, cela peut être suspect et il peut décider de ne pas s'exécuter.

En conclusion, les techniques d'évasion basées sur la détection des environnements de machine virtuelle peuvent être très efficaces pour les malwares cherchant à éviter l'analyse des sandboxes. Toutes les techniques abordées sont utilisées par les malwares de nos jours. Bien que les environnements de machine virtuelle soient des outils indispensables pour l'analyse de malwares, ils continuent à trouver des manières de les contourner en utilisant des techniques de plus en plus sophistiquées.

Une sandbox étant un outil très utilisé pour l'analyse de malware, les techniques d'évasion de machine virtuelle sont des éléments clé de l'arsenal des malwares modernes.

Conclusion

Pour conclure, les solutions de détection de malware ont fait des progrès significatifs dans la détection et la prévention de nombreux types de malwares. Cependant, les malwares sont de plus en plus sophistiqués et utilisent des techniques de dissimulation de plus en plus avancées pour éviter d'être

détectés. Les techniques d'obfuscation, telles que l'obfuscation de code et les packers, sont de plus en plus utilisées pour cacher les malwares et rendre leur analyse plus difficile. De plus, les malwares sont de plus en plus capables de s'exécuter sans être détectés en utilisant des techniques d'évasion sophistiquées. Par conséquent, il est peu probable que les solutions de détection de malware puissent nous protéger complètement contre des attaques ciblées. Cependant, ces solutions restent néanmoins importantes pour protéger les systèmes informatiques contre les malwares courants et pour détecter les attaques avant qu'elles ne causent des dommages importants. En fin de compte, il est important de continuer à développer des techniques de détection et de prévention pour lutter contre les malwares, mais il est également important de prendre des mesures supplémentaires, telles que la mise en place de politiques de sécurité strictes et la formation des utilisateurs, pour réduire les risques d'attaques ciblées.

Attaques ciblées

Panorama des attaques

En 2022 le niveau de la menace s'est maintenu avec 831 intrusions avérées contre 1082 en 2021, mais cela ne doit être interprété comme une baisse de la menace, mais bien un maintien de la menace à un niveau élevé en ciblant des organisations moins bien protégé (*Panorama de la cybermenace 2022 ; CERT-FR — cert.ssi.gouv.fr*, 2023).

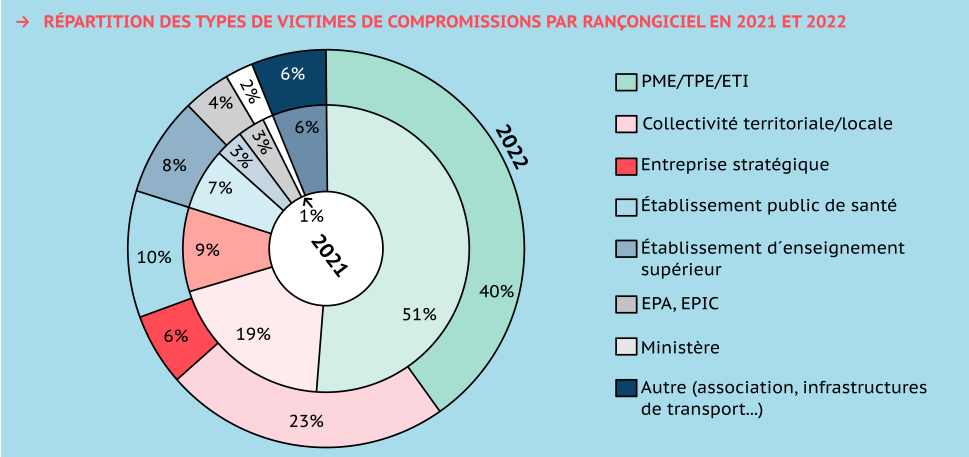

Figure 15: Répartition des types de victimes de compromissions par rançongiciel en 2021 et 2022, tiré de (*Panorama de la cybermenace 2022 ; CERT-FR — cert.ssi.gouv.fr*, 2023)

Comme nous pouvons le voir dans le graphique en figure 15 les rançongiciels ont diminué leurs attaques vers les entreprises pour cibler les collectivités territoriales et locales ainsi que les établissements publics de santé qui sont des cibles connues pour être moins bien protégées.

Historiquement et encore aujourd'hui les motivations des attaquants sont :

- les gains financiers
- le vol d'informations sensibles
- les manœuvres de déstabilisation

(Berady, 2022), (Alshamrani *et al.*, 2019)

Comme vu dans la figure 16, nous pouvons voir que les sources d'attaques peuvent être variées allant de hackers isolées à des pays étrangers, bien sûr pour chaque les motivations seront différentes.

Figure 16: Les différents profils d'attaquants, tiré de (Li et Liu, 2021)

MITRE ATT&CK

MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) est une base de connaissance sur les attaques, les techniques d'attaques et les attaquants dans le domaine de la cybersécurité permettant de décrire les TTPs (tactics, techniques and procedures) (Al-Shaer *et al.*, 2020).

La base de connaissance est divisée en 4 grandes catégories (The MITRE Corporation, s. d.d):

- Les techniques d'attaque : ces techniques répondent au "comment" une tactique est réalisée. Par exemple, il est possible de scanner le disque à la recherche de clés d'accès (SSH par exemple) pour réaliser un mouvement latéral (qui est une tactique dans la matrice MITRE ATT&CK) (The MITRE Corporation, s. d.b).
- Les tactiques d'attaque : ces tactiques répondent au "pourquoi" d'une technique. Elles représentent ce qui sera acquis une fois l'attaque réalisée (reconnaissance, accès, mouvement latéral, …) (The MITRE Corporation, s. d.a).
- Les groupes : ces groupes sont tous les attaquants connus ayant réalisé des attaques, ils sont reliés à leurs logiciels et donc aux techniques et tactiques utilisés par leurs logiciels. Les différents groupes peuvent aussi avoir des relations les uns avec les autres et ces associations sont indiqués dans la base de connaissance de MITRE (The MITRE Corporation, s. d.c).
- Les logiciels : tous les logiciels connus ayant permis de réaliser une attaque. Ils sont constitués de techniques et de tactiques et sont reliés à un ou des groupes (The MITRE Corporation, s. d.e).

Ces catégories permettent de réaliser un profil type des groupes d'attaquants et des attaques pour pouvoir mieux les détecter et les contrer. Elles peuvent aussi aider à la reconnaissance et la rétro-ingénierie de logiciels, malveillant ou non. En listant les fonctionnalités et comportements potentiellement mal-

veillant d'un logiciel (ouverture de fichier, récupération d'informations sur la machine, connexion à un serveur distant, …) la tâche de compréhension de ce que le logiciel fait est facilitée. La reconnaissance peut aussi l'être en détectant des comportements similaires à un autre malware ou famille de malwares.

MITRE ATT&CK va principalement être utilisée par les équipes CTI et par des logiciels d'analyse comportementale pour aider à la compréhension des actions d'un logiciel.

APTs

Contrairement aux attaques dîtes "opportunistes" (Berady, 2022), les attaques modernes sont, comme nous avons pu le remarquer, ciblées et furtives. (Berady, 2022), (Alshamrani *et al.*, 2019)

Dans le panorama de ces menaces d'attaques ciblées ressorts les APTs (Advanced Persistent Threat ou Menace persistante avancée en français), il est important de ne pas confondre ses types d'attaques avec le reste des attaques ciblées.

> Si la réponse à l'un de ces critères relatifs ci-dessous est vraie pour le cas d'attaque en question, alors cette attaque n'est pas une attaque APT, mais une attaque ciblée :
>
> Cette attaque aurait pu être évitée de plus d'une façon.
>
> - Compte tenu du processus d'attaque et de l'environnement cible, si l'attaque n'est pas une surprise et qu'elle est hautement possible, alors elle aurait dû être évitée avec des contre-mesures et des contrôles de sécurité en place.
>
> Cette attaque n'a pas nécessité une grande adaptation de la part des attaquants. par les attaquants
>
> - Si les tentatives des attaquants, pour atteindre leur objectif, ne nécessitent pas une grande adaptation ou de techniques d'évasion intenses aux tentatives du défenseur, alors le système de défense de l'environnement cible doit être remis en question.
>
> Cette attaque ne présentait aucune nouveauté dans ses variantes.
>
> - Une nouveauté dans les méthodes ou techniques d'attaque comme jamais auparavant est ce qui créer souvent le succès d'une attaque APT. S'il n'y a rien de nouveau dans les d'attaque, alors elle est censée être détectée avec les outils et les techniques outils et techniques existants (Alshamrani *et al.*, 2019)

Les différentes étapes d'une attaque ciblée, et plus particulièrement d'une APT, peuvent se retrouver dans le schéma 17.

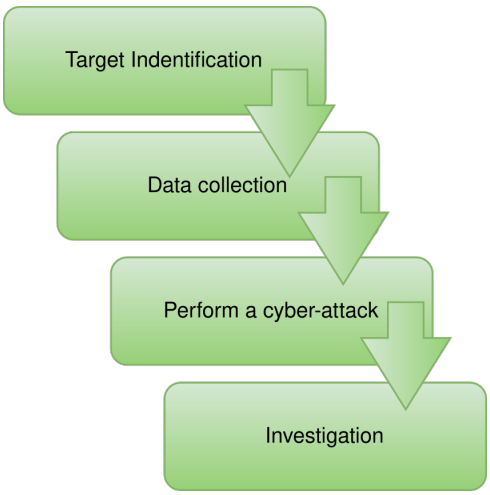

Figure 17: Simplification des étapes d'une attaque ciblées, tiré de (Li et Liu, 2021)

L'étape "Target identification" (en français "identification de la cible"), correspond à l'identification précise de la cible, nous citerons ici quelques critères, mais la liste donnée n'est pas exhaustive :

- Quel est son métier/rôle ? exemple: une PME, une commune, un hôpital
- Quel est sa taille ? exemple: approximation à vue de nez du parc, nombre d'employés
- Correspond-elle à la motivation ? Si motivation par le gain, a-t-elle les moyens / peut-elle débloquer ses fonds
- etc.

Par la suite l'étape de "Data collection" (en français "collecte de données") est sûrement l'étape la plus importante, car là où l'étape précédente permet de déterminer la cible et les étapes suivantes permettent de créer l'attaque ainsi que d'avoir un résultat, cette étape va donner à l'attaquant un maximum d'informations afin de déterminer le meilleur chemin d'attaque et de garantir un résultat. Ici l'attaquant va donc essayer de récupérer un maximum d'informations, et ce, de la manière la plus furtive possible. Cette étape peut être reliée aux techniques de "Reconnaissance" de la matrice MITRE. Il peut donc être amené à utiliser des techniques :

- de renseignement à sources ouvertes autrement appelé OSINT afin de récupérer de manière totalement passive toute information disponible sur sa cible ainsi que les personnes physiques et morales reliées à celle-ci (employés, partenaires, etc.)
- d'ingénierie sociale, par exemple avec du phishing ou vishing (phishing par téléphone), afin de piéger les personnes physiques rattachées à sa cible et ainsi récupérer des informations qu'il n'aurait pas pu avoir autrement
- de scan passif afin de récupérer des informations techniques de façon furtive voir, combiné à des techniques de dissimulation, de scan actif dans le but d'avoir un maximum de connaissances

techniques sur la cible.
- etc.

Arrive alors le moment de “Perform a cyber attack” (en français “effectuer une cyberattaque”). À ce moment l’attaquant à récolter toutes les informations dont il a besoin et a peut-être même pu vérifier certaines techniques qu’il compte utiliser lors de l’attaque, il peut alors effectuer celle-ci.

Vient alors l’étape “Investigation”. Ici l’attaquant va se servir de ses résultats.

Figure 18: Processus de récupération par le support de LockBit, tiré de (Team, 2020)

- Récupération des informations dans le cas d’un “info stealer”, ou d’espionnage
- Discussion avec la victime concernant la rançon, l’illustration 18 présente la page de support du célèbre groupe LockBit.

Modélisation par Kill Chains

Afin de répondre au problème soulevé par ces APTs et puisque les méthodes de réponses à incidents conventionnels n’étaient pas suffisantes, (Hutchins *et al.*, 2010) à proposer un modèle basé sur la notion de Kill Chain.

Cette méthode se décrit par la description d’une attaque sur 7 phases :

1. Reconnaissance : cette phase se traduit par la recherche, l’identification et la sélection des cibles, elle reflète la phase “Target Identification” que nous avons pu voir que nous décrivions les méthodes des APTs.

2. Weaponization : c'est au cours de cette phase que l'attaquant va utiliser une faille dans un utilitaire afin d'y insérer sa charge offensive, par exemple : insérer un reverse shell (connexion de la cible donnant un accès distant) dans une macro Excel
3. Delivery : ici l'attaquant va chercher à envoyer l'arme à ça cible, pour reprendre l'exemple précédent, envoyer le fichier Excel infecter par hameçonnage ciblé.
4. Exploitation : dans cette phase, l'utilisation de l'utilitaire par la cible va lancer la charge malveillante.
5. Installation : l'attaquant va, à présent, installer d'un système de porte dérobée ou d'un accès distant afin de se garantir une persistance dans l'environnement.
6. Command and Control : il est à présent nécessaire à l'attaquant de créer un canal C2 lui permettant une maitrise de l'environnement cible.
7. Action and objectives : c'est enfin, au cours de cette phase, que l'attaquant va chercher à atteindre son objectif final, par exemple : de l'exfiltration/la violation de données

Ce modèle est décrit sous forme de chaîne puisque, comme nous pouvons le remarquer, chaque maillon, autrement appelé phase, est nécessaire pour atteindre le maillon suivant.

Malheureusement, ce modèle présente des défauts :

- il est basé sur son accès initial (Berady, 2022) et, par conséquent, ne décrit pas l'évolution de l'attaque dans le réseau (Pols, 2017) ;
- il se focalise sur l'utilisation d'un malware, ce qui se remarque si l'on regarde les 6 premières phases (Pols, 2017), il ne prend ainsi pas en compte d'autres vecteurs d'initialisation (Pols, 2017), le rendant inadapté pour parler des attaques sur plusieurs mois (Berady, 2022).

Plusieurs modèles ont alors vu le jour afin de pallier ces défauts et (Pols, 2017), suite à une étude de celle-ci, a décrit un modèle les unifiant en les combinant aux TTPs décrits par MITRE ATT&CK, tout en gardant des descriptions simple à chaque phase. Ainsi, il crée la Unified Kill Chain dont la version initiale comportée 17 phases présentées en figure 19.

Ce modèle est représenté sous forme de boucles mettant en valeur la notion de répétitivité (Berady, 2022). Une version plus avancée peut être vue en figure 20.

Par la suite (Berady *et al.*, 2019) propose un nouveau modèle : NUKE, afin de répondre à des besoins de description :

- de la régression de l'attaquant vers un état antérieur ;
- de la répétitivité de l'atteinte des objectifs finaux
- de l'espace de propagation
- des connaissances acquise à propos de l'environnement de la victime

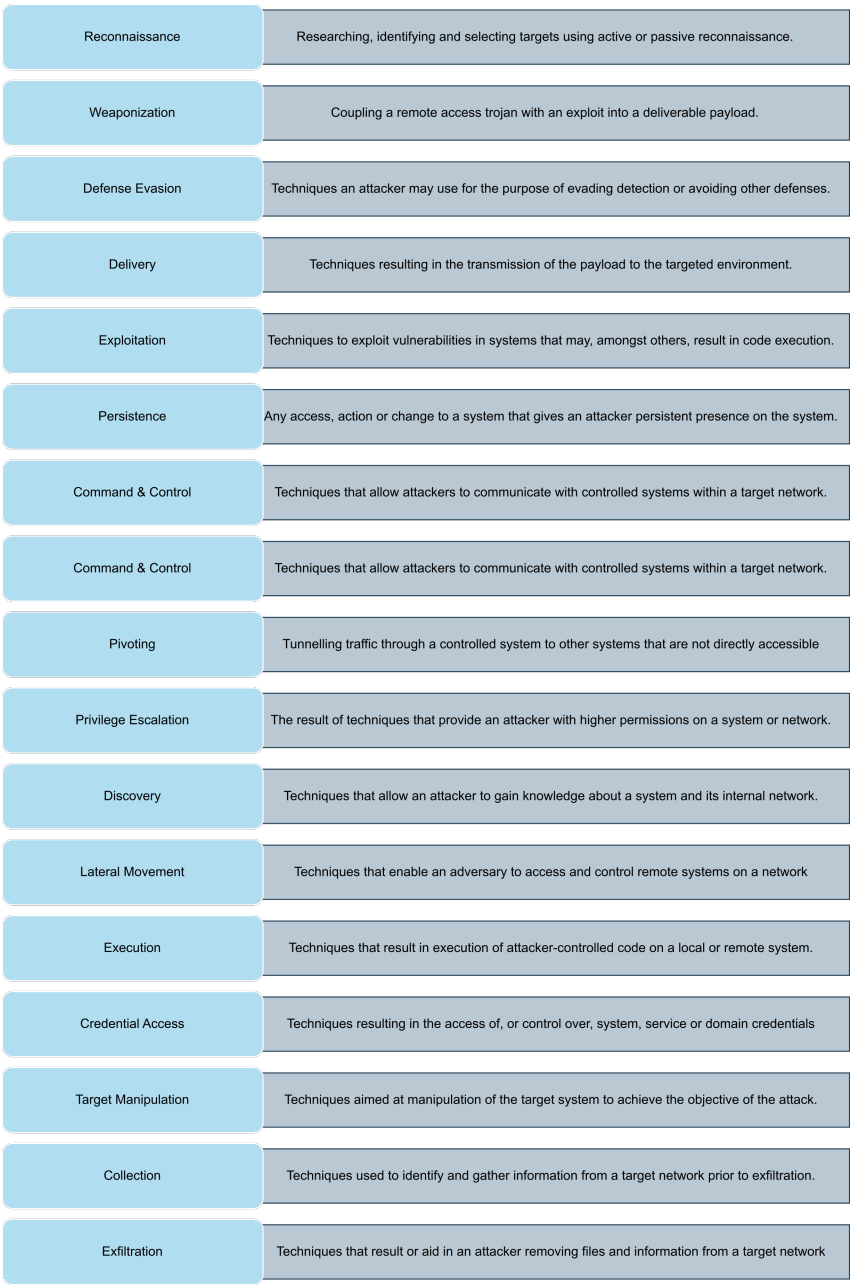

Figure 19: Les 17 phases de l'initiale Unified Kill Chain, tiré de (Pols, 2017)

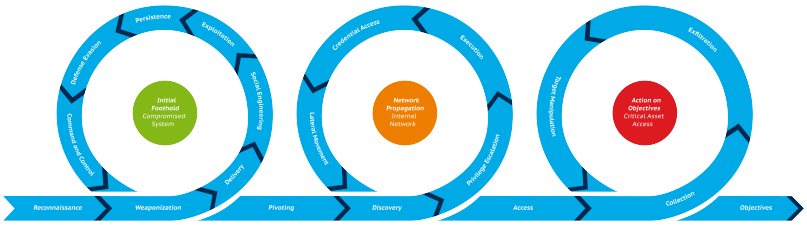

Figure 20: Unified Kill Chain sous forme de boucle tiré de (Berady, 2022)

Coût d'une attaque

Nous avons pu décrire les attaques potentielles en utilisant des cadres tels que MITRE ATT&CK et les schémas de Kill Chains et la description du cadre des APTs, nous avons aussi pu voir les motivations des attaquants. Afin de créer de se protéger, il est important de déterminer le coût d'une attaque sur le système afin que celui-ci reste plus important que les gains potentiels qui créaient la motivation de l'attaquant. Dans cette optique (Derbyshire *et al.*, 2021) mets en lumière l'utilisation du standard ISO27001 et de l'évaluation de risque afin de qualifier ses risques. Ces outils peuvent être couplés aux outils vus précédemment afin de mieux prévoir en détails et de se protéger face à la menace.

ISO 27001

L'ISO 27001 est un standard de sécurité, faisant partie du groupe des ISO 27000 décrivant l'implémentation et l'adoption d'un SMSI (système de gestion de la sécurité de l'information), il est le seul standard de son groupe à délivrer un certificat permettant de prouver que les recommandations de ces standards ont bien été suivies. (Junaid, 2023) Les standards ISO 27000 sont publiés par l'International Organization for Standardization (ISO) en partenariat avec l'International Electrotechnical Commission (IEC) afin de délivrer une méthode pour toute structure. Le standard est en constante évolution afin de rester à jour face à la complexité des infrastructures et des types de menaces de sécurité.

Quels sont les moyens employés pour se protéger d'une attaque ?

Macro

Afin de se protéger d'une attaque, il est important de prendre du recul afin d'avoir une vision d'ensemble et d'étudier sa gestion de risque. Au sein de la gestion de risque se tient un point important :

> l'évaluation des risques, qui est le processus d'identification, d'estimation et de hiérarchisation des risques liés à la sécurité de l'information. (Maček *et al.*, 2020)

Celle-ci se découpe en trois phases :

- Définition de la cible (infrastructure, système, programme, etc.)
- Identification des risques
- Évaluation de ceux-ci (Zambon *et al.*, 2010)

> L'évaluation des risques est considérée comme la phase de planification la plus importante et la plus critique du SMSI dans le processus de gestion des risques lui-même, qui garantit que les risques restent dans des limites acceptables compte tenu du niveau d'appétence pour le risque défini par la direction. (Maček *et al.*, 2020)

Plusieurs techniques d'évaluation des risques ont existé, les plus répandues durant une certaine période furent :

- La Fault Tree Analysis (noté FTA) ou l'analyse de l'arbre de défaillance en français est selon (Safeopedia, 2017), une technique graphique arborescente utilisée pour déterminer les causes profondes d'une défaillance de sécurité, d'un accident ou d'un événement indésirable. Cette technique permet d'évaluer la probabilité qu'un événement indésirable se produise et de prévenir sa survenue en saisissant et en propageant cette probabilité à travers le modèle FTA. Un exemple de FTA peut être vu dans la figure **??**.

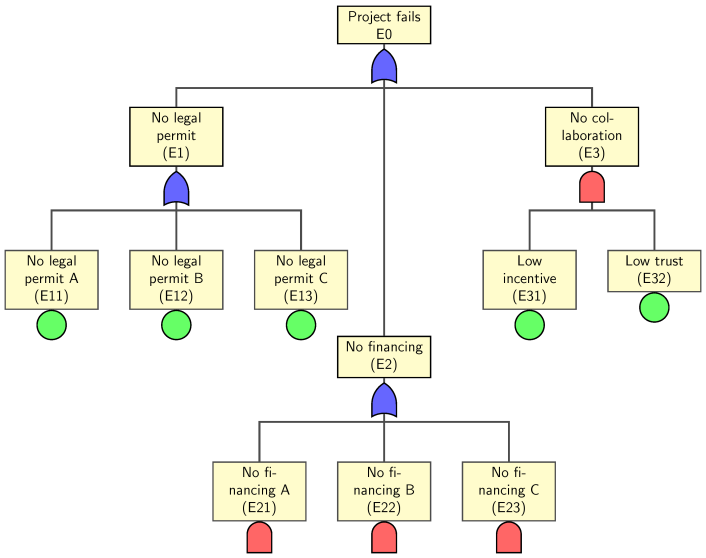

Figure 21: Exemple d'utilisation de FTA

- Un graphe de cyber-attaque est selon (Cyber, 2023), un graphe de cyber-attaque est une représentation de tous les chemins d'attaque possibles contre un réseau de cybersécurité, qui illustre un état où un attaquant a réussi une intrusion. Il y a deux formes populaires de graphes

d'attaque. La première est un graphe direct où les nœuds représentent les états du réseau et les bords représentent les exploits qui transforment un état en un état plus compromis, qui montre finalement une attaque réussie. La deuxième forme est un graphe direct où les nœuds représentent les pré ou postconditions d'un exploit et les arêtes représentent les conséquences d'une précondition qui permet un exploit postcondition.

Malheureusement, ces techniques étaient trop coûteuses en termes de temps et de ressources requis (Zambon *et al.*, 2010).

Ainsi de nouvelles méthodes d'analyse de risques sont apparues :

L'analyse quantitative consiste à attribuer des valeurs numériques et monétaires aux différents éléments du processus d'analyse des risques, qui sont ensuite utilisées pour déterminer le risque total et résiduel. Cependant, en raison de sa complexité, de sa longueur, de son coût et de sa difficulté à être réalisée entièrement, cette méthode est rarement utilisée seule. (Maček *et al.*, 2020)

Elle est souvent combinée avec une approche qualitative, qui repose sur le jugement subjectif de l'équipe d'évaluation des risques pour déterminer le risque global. Dans cette approche, le risque est évalué à l'aide d'une échelle hiérarchique de gravité, telle que Critique, Élevé, Moyen et Faible, sans l'utilisation de valeurs numériques. Les méthodes qualitatives d'évaluation des risques comprennent les jugements, les meilleures pratiques, l'intuition et l'expérience de l'évaluateur, qui sont utilisés pour évaluer les probabilités de risque et les contre-mesures possibles. (Maček *et al.*, 2020)

EBIOS Risk Manager, publiée par l'ANSSI et faisant suite à EBIOS, est une méthode d'analyse qualitative qui a pour principes majeurs :

- Synthèse entre conformité et scénarios
- Prise en compte de l'écosystème
- Points de vue du défenseur et de l'attaquant
- Représentativité plutôt qu'exhaustivité
- Sources de risque intentionnelles
- Intégration et compatibilité avec les démarches et processus de management des risques globaux

Elle est dîtes qualitatives car elle

> se fonde sur un socle de sécurité solide, construit grâce à une approche par conformité, qui se traduit par un ensemble de mesures de sécurité d'état de l'art complétées par des mesures réglementaires ou normative. La démarche sélective par scénarios de risque vient ensuite solliciter et « malmener » ce socle face à des menaces contextualisées, potentiellement ciblées ou sophistiquées, qui prennent en compte l'écosystème métier et technique dans lequel l'organisation évolue. (Maček *et al.*, 2020)

Il est important de noter ici que

> La vraisemblance des scénarios est évaluée sous l'angle du retour sur investissement du point de vue de l'attaquant (Maček *et al.*, 2020)

L'analyse EBIOS se déroule alors en 5 ateliers qui sont les suivants :

- Cadrage et socle de sécurité
- Sources de risque
- Scénarios stratégiques
- Scénarios opérationnels
- Traitement du risque

Ainsi grâce à cette méthode, ou une autre méthode d'évaluation des risques selon le contexte, l'organisation peut alors qualifier, se préparer et traiter les différents risques. À cette méthode et ses outils de gestion d'un point de vue macroscopique doivent alors s'accompagner différents outils pour répondre aux besoins techniques et microscopiques qui en découlent.

Micro

À la suite d'une évaluation des risques, ceux-ci sont identifiés et classifiés, il est alors temps de mettre en place des actions tangibles afin de répondre aux traitements de ces dits risques. Lors de l'évaluation, les risques identifiés peuvent provenir de différentes sources ils peuvent être :

- interne
- externe
- automatisé
- de source humaine
- provenir de mauvaises configurations
- être découvert lors de l'arrivée de nouvelles failles sur les technologies utilisées

Il est alors important de réfléchir à la bonne méthode pour répondre à ses différents risques.

Afin d'atténuer le risque interne provenant de source humaine, il est souvent demandé de former les utilisateurs aux différentes politiques de sécurité permettant d'éviter des mauvaises pratiques. À cela il ne faut pas oublier que

> Souvent, les employés ont recours à des rationalisations, appelées techniques de neutralisation, pour surmonter les effets de la dissuasion lorsqu'ils décident d'enfreindre ou non une politique. (Barlow *et al.*, 2013)

Exemple de techniques de neutralisation :

"J'ai donné mes identifiants de session à mon collègue, car j'avais besoin de rester connecté et il avait urgemment besoin d'imprimer un document depuis sa clé USB"

Ici, en cumulant trois techniques de neutralisation :

- J'ai besoin de garder ma session active
- Mon collègue à urgemment besoin d'imprimer
- Le document de mon collègue se trouve sur sa clé USB

la personne enfreint deux politiques :

- Les identifiants de session sont propres et permettent par essence d'identifier une personne aussi il est interdit de les partager.
- Il est interdit de brancher des périphériques USB sur les postes.

Les organisations ont souvent recours à une formation axée sur la dissuasion pour réduire les violations commises par les employés en utilisant des messages à connotation négative. Cependant, d'après (Barlow *et al.*, 2013), les techniques de neutralisation sont plus efficaces que les sanctions pour déterminer si les employés vont violer la politique. Par conséquent, dans les programmes d'éducation, de formation et de sensibilisation à la sécurité (SETA), la communication sur la sécurité devrait également se concentrer sur les moyens d'atténuer les comportements de neutralisation. (Barlow *et al.*, 2013) suggèrent que la communication de sécurité pourrait aider à réduire la probabilité de neutralisation en soulignant la raison d'être de la politique et la manière dont sa violation peut entraîner des dommages à l'entreprise, aux clients ou aux employés.

Afin que la communication ait un réel impact, il faut aussi bien soigner le fond que la forme en ce point, par exemple les jeux en ligne et les courts métrages d'animation sont les médias les plus répandus pour l'enseignement de la cybersécurité. Cependant, il est important de noter que l'efficacité de ces méthodes d'enseignement manque d'études à long terme les comparants à d'autres méthodes d'enseignement. (Zhang-Kennedy et Chiasson, 2021)

Malgré cela, il est intéressant de s'appuyer sur cette étude afin de choisir les médias de communications qui seront utilisés dans la formation des utilisateurs. Le support de formation doit être visé vers son champ d'application.

Puisque l'erreur reste humaine, à la formation des utilisateurs doit s'accompagner des outils afin de sécuriser l'infrastructure, les réseaux et les produits. De nombreux outils existent à ce jour afin d'accompagner les administrateurs et les utilisateurs dans la sécurité des infrastructures, nous nous concentrerons ici sur quelques-uns puisque la liste de ces outils et de leurs types est longue et ne fait que s'agrandir, aussi aucune liste ne peut être exhaustive.

Dans un premier lieu nous pouvons parler des antivirus notés AVs dont nous avons déjà pu discuter plus tôt dans notre partie sur les techniques de détection. Ceux-ci ont dans un premier temps utilisé

des techniques statiques basées sur des signatures afin de reconnaître une charge malveillante et ont aujourd'hui évolué afin de rassembler d'autres techniques aussi bien statique que dynamique. Ils vont pour la plupart de nos jours fournir une "protection en temps réel", le principe est que l'antivirus va scanner les fichiers qui arrivent sur la machine afin de détecter une charge malveillante, dans le cas échéant, l'antivirus va isoler, ou directement supprimer, le ou les fichiers contenant cette charge.

Par la suite, le pare-feu (en anglais firewall) autrement dit est

> Un dispositif ou une combinaison de matériel et de logiciel utilisé pour renforcer la sécurité des communications en bloquant l'accès non autorisé à un réseau local. (« IEEE Standard for Security Management for Customer Cryptographic Assets on Cryptocurrency Exchanges », 2022)

Aujourd'hui, les pare-feux se retrouvent sur tout type de machine, du pare-feu physique d'entreprise, au pare-feu logiciel installé sur un ordinateur personnel, en passant par le pare-feu logiciel installé dans la box d'un fournisseur d'accès à internet.

À cela s'ajoutent les IPS et IDS, respectivement système de prévention et de détection d'intrusion (en anglais intrusion detection system et intrusion prevention system (« IEEE Standard for an Architectural Framework for the Internet of Things (IoT) », 2020))

> La détection d'intrusion est le processus d'observation des événements qui se produisent dans un système ou un réseau informatique, et d'analyser ces événements pour déterminer les intrusions. Il existe diverses menaces dont les logiciels malveillants, les attaques DoS-DDoS, les accès non autorisés, l'escalade des privilèges ou l'attaque par sonde. Bien que de nombreux événements qui semblent nuisibles au système sont en fait des attaques, il existe quelques exceptions. L'utilisateur peut se tromper d'adresse d'ordinateur ou se connecter sans le savoir au mauvais système. Le système doit correctement séparer les intrusions du trafic réseau normal. En conclusion, un IDS est un logiciel qui simplifie et automatise le processus de détection des attaques.

Les IPS vont avoir cette même recherche et vont se connecter à un pare-feu afin de bloquer les intrusions.

Puisque nous parlons ici d'attaques ciblées, nous nous devons de parler d'honeypots.

> Un honeypot est une installation de sécurité d'une importance vitale créée pour être sondé, attaqué et compromis, dans le but de piéger les adversaires et d'enquêter sur les attaques connues et surtout, les attaques inconnues. (Fan *et al.*, 2019)

Grâce à ce système une entreprise peut mieux comprendre les attaques, et tentatives d'attaques, commises à son encontre, que celles-ci soient connues ou non.

> Afin de minimiser cette menace (les attaques par internet), il est nécessaire de disposer d'un système de sécurité qui a la capacité à détecter les attaques de type "zero-day" et à les bloquer. "Honeypot" est la technologie de défense proactive, dans laquelle les ressources placées dans un réseau dans le but d'observer et de capturer de nouvelles attaques". (Kondra *et al.*, 2016)

Un honeypot connecté à un IDS peut ainsi permettre de générer des règles sur un IDS afin de détecter si les attaques capturées via notre piège sont lancée sur notre infrastructure de production. En ce sens le travail de (Kondra *et al.*, 2016) fait une analyse de performance de ce type de système.

Nous pouvons ainsi penser à agrandir cette idée vers un système d'interconnexion entre différents outils de sécurité afin de détecter et de se protéger des attaques efficacement. En ce sens les OpenXDR peuvent répondre à cette problématique.

> Open XDR est un type de solution ou de plateforme de sécurité de détection et de réponse étendue (XDR) qui prend en charge les intégrations tierces pour collecter des formes spécifiques de télémétrie afin de permettre la détection, la chasse et l'investigation des menaces à travers différentes sources de données et d'exécuter des actions de réponse.

En Europe, nous avons une solution fédérée d'OpenXDR nommé l'OpenXDRPlatform.

> Des pureplayers européens chevronnés et des cyber-experts spécialisés dans les contrôles de sécurité les plus essentiels se sont associés pour proposer une cyberdéfense fédérée sous la forme d'une pile technologique unique (*OPEN XDR PLATFORM | Federate your cyberdefence. — openxdrplatform.io*, s. d.)

Conclusion

En conclusion, la catégorisation des attaques et les moyens de s'en défendre s'améliorent de façon continuelle. Nous avons pu voir que l'information, le tri et le partage de celle-ci porte une place primordiale dans cette volonté de défense, cependant des améliorations reste à faire afin de suivre les plans d'attaques en constant renouveau.

Conclusion

Dans ce mémoire de recherche, nous avons évalué l'efficacité des solutions de détection de malwares face aux attaques ciblées. Pour cela, nous avons exploré les techniques de détection de malwares, les méthodes de dissimulation des malwares et la caractérisation d'une attaque ciblée au niveau organisationnel.

Nous avons étudié les différentes techniques de détection de malwares, notamment l'analyse statique, dynamique et hybride. Bien qu'efficaces contre les malwares courants, ces méthodes peuvent être contournées par des malwares sophistiqués utilisant des techniques d'obfuscation.

Ensuite, nous avons examiné les techniques d'obfuscation de malwares, telles que l'obfuscation de code, les packers, les techniques anti-débogueur et les techniques anti-VM. Ces méthodes sont couramment utilisées pour rendre les malwares plus difficiles à analyser et à détecter par les solutions de sécurité existantes. Comprendre ces techniques est essentiel pour mieux protéger les systèmes informatiques contre les attaques ciblées.

Dans la dernière partie, nous avons étudié la caractérisation d'une attaque ciblée sur le plan organisationnel, en examinant les différentes phases d'une attaque ciblée et les techniques utilisées par les attaquants pour compromettre les systèmes. Nous avons également abordé les mesures de sécurité organisationnelles pouvant être mises en place pour réduire les risques d'attaques ciblées et protéger les systèmes d'information.

Nous avons constaté que, bien que les solutions de détection de malwares soient importantes pour protéger les systèmes informatiques, elles ne peuvent pas offrir une protection complète contre les attaques ciblées. Les malwares sophistiqués utilisent des techniques d'obfuscation pour échapper à la détection, et les attaquants peuvent recourir à des techniques d'évasion pour contourner les solutions de sécurité existantes.

Il est donc essentiel de mettre en place des mesures de sécurité organisationnelles robustes pour protéger les systèmes d'information contre les attaques ciblées, notamment en sensibilisant les employés à la sécurité informatique et en effectuant des mises à jour régulières du système d'exploitation et des logiciels.

Cependant, ce mémoire de recherche présente quelques limites importantes qui doivent être prises en compte lors de l'interprétation des résultats. Les techniques de détection de malwares et de dissimulation évoluent constamment, les attaquants développent sans cesse de nouvelles méthodes pour échapper à la détection et contourner les solutions de sécurité. Par conséquent, les résultats de cette étude peuvent rapidement devenir obsolètes à mesure que de nouvelles techniques et technologies émergent.

De plus, notre mémoire de recherche s'est concentrée sur les techniques de détection de malwares, de dissimulation et les attaques ciblées actuellement connues. Cependant, il est probable que de nombreuses autres techniques et attaques n'aient pas été abordées dans cette étude. La portée limitée de notre recherche peut donc affecter la pertinence des résultats pour l'ensemble des menaces informatiques auxquelles les organisations sont confrontées.

Il convient également de noter que notre étude ne s'est pas penchée sur les performances et capacités spécifiques des différentes solutions de détection de malwares disponibles sur le marché. Par consé-

quent, il est difficile d'évaluer l'efficacité de chaque solution de détection face aux attaques ciblées et aux techniques de dissimulation décrites dans cette étude.

Enfin, la méthodologie de recherche employée dans ce mémoire peut également présenter des limites. Les résultats sont basés sur une analyse de la littérature et des rapports existants sur les techniques de détection de malwares et les attaques ciblées. Il se peut que des études plus approfondies, notamment des tests en conditions réelles, révèlent des résultats différents ou complémentaires.

En dépit de ces limites, ce mémoire de recherche fournit un aperçu important des défis auxquels les organisations sont confrontées pour se protéger contre les attaques ciblées et des domaines dans lesquels les solutions de détection de malwares peuvent être améliorées. La mise en place de mesures de sécurité organisationnelles robustes et la compréhension des techniques de dissimulation utilisées par les malwares sont des éléments clés pour renforcer la protection des systèmes d'information contre les attaques ciblées.

En conclusion, cette étude souligne l'importance de la détection de malwares et des mesures de sécurité organisationnelles pour faire face aux attaques ciblées. Bien que les solutions de détection de malwares jouent un rôle crucial dans la protection des systèmes informatiques, il est essentiel de mettre en place des mesures de sécurité organisationnelles complémentaires pour minimiser les risques d'attaques ciblées. Les chercheurs en sécurité informatique doivent continuer à développer de nouvelles techniques de détection et de dissimulation pour contrer l'évolution constante des malwares et des attaques ciblées. De plus, une collaboration étroite entre les organisations, les chercheurs et les autorités compétentes est nécessaire pour partager les informations et les meilleures pratiques afin de créer un environnement de cybersécurité plus résilient.

Dans les perspectives futures, il serait intéressant d'explorer davantage les enjeux techniques et organisationnels liés à la détection de malwares et la sécurité informatique en général. D'un point de vue technique, l'importance d'une détection rapide des malwares est cruciale pour les organisations, car elle leur permet de réagir rapidement et de limiter les dommages causés par les attaques.

Du point de vue organisationnel, la communication et la sensibilisation jouent un rôle primordial dans la création d'une culture de cybersécurité au sein des entreprises. Il est crucial de transmettre le message adéquat de la manière appropriée et aux personnes concernées, en tenant compte des différences entre les employés IT et non-IT. Des ateliers, des formations et des simulations d'attaques pourraient être organisés pour renforcer la compréhension des enjeux de cybersécurité et impliquer les employés non-IT dans les discussions et la prise de décisions relatives aux règles de sécurité. Cela pourrait favoriser l'adoption de ces règles et les rendre moins perçues comme un fardeau.

L'interface homme-machine (IHM) et l'expérience utilisateur (UX/UI) constituent également des éléments essentiels pour faciliter l'adoption des mesures de sécurité par les employés. Les solutions de sécurité devraient être conçues en prenant en compte l'usage des utilisateurs et les avantages qu'ils pourraient retirer en contournant les dispositifs de sécurité. Les concepteurs pourraient s'attacher à

développer des solutions qui s'intègrent harmonieusement dans les flux de travail des employés, en étudiant par exemple l'utilisation de clés de sécurité à la place des mots de passe, ou de gestionnaires de mots de passe permettant à la fois d'augmenter la sécurité et de simplifier l'utilisation pour les employés. De plus, des études pourraient être menées pour identifier les meilleures pratiques en matière d'IHM et d'UX/UI, afin de rendre les solutions de sécurité plus accessibles et intuitives pour les utilisateurs de tous niveaux de compétence.

Références

(s. d.).

abuse.ch. (s. d.). *MalwareBazaar*. https://bazaar.abuse.ch.

Ahmed Bahgat. (2022). *Types de logiciels malveillants : Apprends à mieux vous protéger*. https://kinsta.com/fr/blog/types-de-logiciels-malveillants/

Alexandre D'Hondt, C. H. B. V. O. et A. L. (2023). *Experiment Toolkit for Manipulating Executable Packing* [mémoire de master]. Université Catholique de Louvain, Belgique.

Alomari, E. S., Nuiaa, R. R., Alyasseri, Z. A. A., Mohammed, H. J., Sani, N. S., Esa, M. I. et Musawi, B. A. (2023). Malware Detection Using Deep Learning and Correlation-Based Feature Selection. *Symmetry*, *15*(1). https://doi.org/10.3390/sym15010123

Al-Shaer, R., Spring, J. M. et Christou, E. (2020). Learning the Associations of MITRE ATT & CK Adversarial Techniques. In *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE. https://doi.org/10.1109/cns48642.2020.9162207

Alshamrani, A., Myneni, S., Chowdhary, A. et Huang, D. (2019). A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities. *IEEE Communications Surveys & Tutorials*, *21*, 1851-1877. https://doi.org/10.1109/comst.2019.2891891

Ange Albertini. (2010). *Packers*.

Arini Balakrishnan et Chloe Schulze. (2005). *Code Obfuscation Literature Survey*.

Ashu Sharma et S. K. Sahay. (n.d.). *Evolution and Detection of Polymorphic and Metamorphic Malwares: A Survey*.

Aslan, Ö. A. et Samet, R. (2020). A Comprehensive Review on Malware Detection Approaches. *IEEE Access*, *8*, 6249-6271. https://doi.org/10.1109/ACCESS.2019.2963724

Aslan, Ö. et Samet, R. (2020). *A Comprehensive Review on Malware Detection Approches*.

AV Atlas. (s.d.). https://portal.av-atlas.org

Ayelen Torello and Francis Guibernau. (n.d.). *Environment Awareness*.

Backerstreet. (s. d.). *Reverse Engineering Resources - Decompilers*. https://www.backerstreet.com/decompiler/decompilers.htm.

Barlow, J. B., Warkentin, M., Ormond, D. et Dennis, A. R. (2013). Don't make excuses! Discouraging neutralization to reduce IT policy violation. *Computers & Security*, *39*, 145-159. https://doi.org/https://doi.org/10.1016/j.cose.2013.05.006

Basatwar, G. (2022). Anti-Debugging – A Quick Guide To Avoid Malwares And Mobile App Hacks. In *Appsealing.com*. https://www.appsealing.com/anti-debugging/

Berady, A. (2022). *Comprendre les menaces sophistiquées : Intentions, moyens, manières et connaissances convergentes des adversaires* [theses, CentraleSupélec]. https://theses.hal.science/tel-03861429

Berady, A., Tong, V. V. T., Guette, G., Bidan, C. et Carat, G. (2019). Modeling the Operational Phases of APT Campaigns. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE. https://doi.org/10.1109/csci49370.2019.00023

BitDefender. (s. d.). *Pourquoi choisir les solutions de protection BitDefender ?* http://download.bitdefender.com/resources/files/Main/file/1_La_technologie_BitDefender_Antivirus.pdf.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). *Language Models are Few-Shot Learners*. https://arxiv.org/abs/2005.14165

Bruce Dang; Alexandre Gazert; et Elias Bachaalany. (2014). *Practical Reverse Engineering: x86, x64, ARM, Windows kernel, reverseing tools, and obfuscation*. WILEY.

Casey Crane. (2020). *What Is Malware? 10 Types of Malware & How They Work*. https://www.thesslstore.com/blog/what-is-malware-types-of-malware-how-they-work/

Charles-Henry Bertrand Van Ouytsel, Thomas Given-Wilson, Jeremy Minet, Julian Roussieau et Axel Legay. (2021). *Analysis of Machine Learning Approaches to Packing Detection*.

Crepaldi, Michèle. (2021). *Automatic Malware Signature Generation* [thèse de doctorat]. Politecnico di Torino.

Crepaldi, Michele. (2021). *Automatic Malware Signature Generation*.

Cyber, X. (2023). *What are Attack Graphs*. XM Cyber.

dCode. (n.d.). *Chiffre ROT-13*. https://www.dcode.fr/chiffre-rot-13

Deloitte Threat Library Team, Jeff Felling, Red Canary, Jorge Orchilles, SCYTHE and Ruben Dodge. (2021). *Virtualization/Sandbox Evasion: Time Based Evasion*. https://www.microsoft.com/en-us/security/blog/2021/03/04/goldmax-goldfinder-sibot-analyzing-nobelium-malware/

Derbyshire, R., Green, B. et Hutchison, D. (2021). “Talking a different Language”: Anticipating adversary attack cost for cyber risk assessment. *Computers & Security*, *103*, 102163. https://doi.org/10.1016/j.cose.2020.102163

Dilshan Keragala. (2016). Detection Malware and Sandbox Evasion Techniques. *SANS Institute*.

Dr. Mafaz Mohsin Khalil Al-Anezi. (2014). *Generic Packing Detection using Several Complexity Analysis for Accurate Malware Detection*.

Ellie Burns. (2021). *What is malware?* https://techmonitor.ai/what-is/what-is-malware-4924141

ESET. (s. d.). *Inspection comportementale avancée pour contrer les attaques*. https://www.eset.com/fr/about/newsroom/press-releases/produits/inspection-contre-attaques/.

Eskandari, M., Khorshidpour, Z. et Hashemi, S. (2013). HDM-Analyser: A hybrid analysis approach based on data mining techniques for malware detection. *Journal of Computer Virology and Hacking Techniques*, *9*. https://doi.org/10.1007/s11416-013-0181-8

Evgenios Konstantinou. (2008). *Metamorphic Virus: Analysis and Detection*.

Fan, W., Du, Z., Smith-Creasey, M. et Fernández, D. (2019). HoneyDOC: An Efficient Honeypot Architecture Enabling All-Round Design. *IEEE Journal on Selected Areas in Communications*, *37*(3), 683-697. https://doi.org/10.1109/JSAC.2019.2894307

Gadhiya, S. et Bhavsar, K. (2013). Techniques for Malware Analysis. *International Journal of Advanced Research in Computer Science and Software Engineering*.

H., N. (2009). *Complexity-based Packed Executable Classification with High Accuracy* [mémoire de master]. School of Engineering, Information; Communications University, Korea.

Han, W., XUE, J., Wang, Y., Huang, L., Kong, Z. et Mao, L. (2019). MalDAE: Detecting and Explaining Malware based on Correlation and Fusion of Static and Dynamic Characteristics. *Computers & Security*, *83*. https://doi.org/10.1016/j.cose.2019.02.007

hasherezade. (2021). *Understanding and countering malware's evasion and self-defence*.

Help Net Security. (2004). *Mydoom.A: Timeline of an Epidemic*. https://www.helpnetsecurity.com/2004/02/03/mydooma-timeline-of-an-epidemic/

Hutchins, E. M., Cloppert, M. J. et Amin, R. M. (2010). *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains*.

IEEE Standard for an Architectural Framework for the Internet of Things (IoT). (2020). *IEEE Std 2413-2019*, 1-269. https://doi.org/10.1109/IEEESTD.2020.9032420

IEEE Standard for Security Management for Customer Cryptographic Assets on Cryptocurrency Exchanges. (2022). *IEEE Std 2140.2-2021*, 1-20. https://doi.org/10.1109/IEEESTD.2022.9676563

Ilsun You et Kangbin Yim. (2010). *Malware Obfuscation Techniques: A Brief Survey*.

Joseph Regan et Ivan Belcic. (2022). *What Is Malware ? The Ultimate Guide to Malware*. https://www.avg.com/en/signal/what-is-malware

Junaid, T.-S. (2023). *ISO 27001: Information Security Management Systems* [thèse de doctorat]. https://doi.org/10.13140/RG.2.2.36267.52005

Kaspersky. (s. d.). *Emulator*. https://www.kaspersky.com/enterprise-security/wiki-section/products/emulator.

Kondra, J. R., Bharti, S. K., Mishra, S. K. et Babu, K. S. (2016). Honeypot-based intrusion detection system: A performance analysis. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (p. 2347-2351).

Larousse. (s. d.). *Corrélation*. https://www.larousse.fr/dictionnaires/francais/corrélation/19435.

Li, Y. et Liu, Q. (2021). A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. *Energy Reports*, *7*, 8176-8186. https://doi.org/10.1016/j.egyr.2021.08.126

M., B. (s. d.). *Que signifient hash et hachage ?* https://cryptoast.fr/hash-hachage-bitcoin-blockchain/.

Maček, D., Magdalenić, I. et Ređep, N. B. (2020). A Systematic Literature Review on the Application of Multicriteria Decision Making Methods for Information Security Risk Assessment. *International Information and Engineering Technology Association*.

Malwarebyte. (s. d.). *Explained: False positives*. https://www.malwarebytes.com/blog/news/2017/09/explained-false-positives.

malwarelu. (2012). *Analysis & pwonage of herpesnet botnet*. https://malware.lu/articles/2012/05/21/analysis-and-pownage-of-herpesnet-botnet.html

Mishra, U. (2013). *Finding and Solving Contradictions of False Positives in Virus Scanning*. arXiv. https://doi.org/10.48550/ARXIV.1306.4652

Monitor, C. (s. d.). *Stuxnet malware is 'weapon' out to destroy ... Iran's Bushehr nuclear plant?* https://www.csmonitor.com/USA/2010/0921/Stuxnet-malware-is-weapon-out-to-destroy-Iran-s-Bushehr-nuclear-plant/(page)/2.

OPEN XDR PLATFORM | Federate your cyberdefence. — openxdrplatform.io. (s. d.). https://www.openxdrplatform.io/.

pandasecurity. (2021). *Fileless Malware Attacks Surge by 900 percentage and Cryptominers Make a Comeback*. https://www.pandasecurity.com/en/mediacenter/news/internet-security-report-q4-watchguard/

Panorama de la cybermenace 2022 ; CERT-FR — cert.ssi.gouv.fr. (2023). https://www.cert.ssi.gouv.fr/cti/CERTFR-2023-CTI-001/.

Paul Rascagneres. (2018). GravityRAT - The Two-Year Evolution Of An APT Targeting India. *Cisco Talos Intelligence Blog*.

Paul RASCAGNERS and Sébastien LARINIER. (2022). *Cybersécurité et Malwares - Détection, analyse et Threat Intelligence*. ENI.

Pols, P. (2017). [thèse de doctorat, Cyber Security Academy (CSA)]. In *The Unified Kill Chain*. https://www.unifiedkillchain.com/assets/The-Unified-Kill-Chain-Thesis.pdf

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. et Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*.

Ramin Nafisi and Andrea Lelli. (2021). *GoldMax, GoldFinder, and Sibot: Analyzing NOBELIUM's layered persistence*. https://www.microsoft.com/en-us/security/blog/2021/03/04/goldmax-goldfinder-sibot-analyzing-nobelium-malware/

Safeopedia. (2017). *Fault Tree Analysis*. Safeopedia.

Shaul Vilkomir-Preisman. (2019). Malware Evasion Technique Part 2: Anti-VM. *Deep Instinct*. https://www.deepinstinct.com/blog/malware-evasion-techniques-part-2-anti-vm-blog

Siamesekitten. (2021). New Iranian Espionage Campaign. *ClearSky*.

Sikorski, M. et Honig, A. (2012). *Practical malware analysis: the hands-on guide to dissecting malicious software*. No Starch Press.

Team, A. O. I. (2020). *Tales From the Trenches; a Lockbit Ransomware Story | McAfee Blog — mcafee.com*. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/tales-from-the-trenches-a-lockbit-ransomware-story/.

terranovasecurity. (2022). *Attaque par malware : les exemples les plus courants*. https://terranovasecurity.com/fr-fr/exemples-attaque-par-malware/

The MITRE Corporation. (s. d.a). *Enterprise tactics*. https://attack.mitre.org/tactics/enterprise/.

The MITRE Corporation. (s. d.b). *Enterprise Techniques*. https://attack.mitre.org/techniques/enterprise/.

The MITRE Corporation. (s. d.c). *Groups*. https://attack.mitre.org/groups/.

The MITRE Corporation. (s. d.d). *Les différentes techniques de détection des virus et malwares*. https://attack.mitre.org/.

The MITRE Corporation. (s. d.e). *Software*. https://attack.mitre.org/software/.

Timea Laszlo et Akos Kiss. (n.d.). *Obfuscation C++ Programs via Control Flow Flattening*.

Transformation, P. (s. d.). *Tamper Resistant Software*. http://www.program-transformation.org/Transform/TamperResistantSoftware.html.

Transformation, program. (s. d.). *Why Decompilation*. http://www.program-transformation.org/Transform/WhyDecompilation.

Turc, I., Chang, M.-W., Lee, K. et Toutanova, K. (2019). Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv preprint arXiv:1908.08962v2*.

Université Paris-Est Marne-la-Vallée. (n.d.). *L'analyse de malware*. https://www-igm.univ-mlv.fr/~dr/XPOSE2013/introduction_analyse_malware/obfuscation.html

Visser, E. (s. d.). *Program Obfuscation*. http://www.program-transformation.org/Transform/ProgramObfuscation.html.

Vouvoutsis, V., Casino, F. et Patsakis, C. (2022). *On the Effectiveness of Binary Emulation in Malware Classification*. arXiv. https://doi.org/10.48550/ARXIV.2204.04084

vx-underground. (s. d.). *VX Underground*. https://www.vx-underground.org/.

Wei Xu, Fangfang Zhang et Sencun Zhu. (n.d.). *The Power of Obfuscation Techniques in Malicious JavaScript Code: A Measurement Study*.

Wikipedia. (s. d.). *Désassembleur*. https://fr.wikipedia.org/wiki/D%C3%A9sassembleur#D%C3%A9sassembleurs_gratuits_et/ou_libres/.

Wikipedia. (2022). *Logiciel antivirus*. https://fr.wikipedia.org/wiki/Logiciel_antivirus

Wing Wing et Mark Stamp. (2014). *Hunting for Metamorphic Engines*.

Yesir, S. et Soğukpinar, İ. (2021). *Malware Detection and Classification Using fastText and BERT*. https://doi.org/10.1109/ISDFS52919.2021.9486377

YUCEEL, H. C. et Labs, P. (2022). Virtualization/Sandbox Evasion - How Attackers Avoid Malware Analysis. In *Picus Security*. https://www.picussecurity.com/resource/virtualization/sandbox-evasion-how-attackers-avoid-malware-analysis

Zambon, E., Etalle, S., Wieringa, R. J. et Hartel, P. (2010). Model-based qualitative risk assessment for availability of IT infrastructures. *Formal Aspects of Computing*.

Zhang-Kennedy, L. et Chiasson, S. (2021). A Systematic Review of Multimedia Tools for Cybersecurity Awareness and Education. *ACM Comput. Surv.*, *54*(1). https://doi.org/10.1145/3427920

Zuzana Hromcova. (2019). Okrum and Ketrican: An overview of recent re3chang group activity. *ESET*.