Dynamic Malware Analysis for Dummies

Philippe Lagadec

NATO/NC3A
philippe.lagadec(à)nc3a.nato.int

Résumé Cet article décrit différentes méthodes pour effectuer l'analyse de logiciels malveillants (malware), plus particulièrement sur les plate-formes Windows. Il se penche en particulier sur l'analyse dynamique, qui consiste a exécuter le code malveillant sur un système spécialement configuré pour enregistrer toutes ses actions. Cela permet d'obtenir rapidement des résultats pour déterminer la nature du logiciel malveillant et décider comment réagir. L'article montre également comment construire un laboratoire d'analyse dynamique simple et à moindre coût, détaille la méthode employée et propose quelques pistes pour aller plus loin.

1 Introduction

L'analyse de codes malveillants (malware analysis) est aujourd'hui devenue une activité très importante dans le cadre de la gestion des incidents de sécurité informatique. Les organisations qui prennent sérieusement en compte la sécurité de leurs réseaux sont souvent confrontées a des fichiers suspects capturés grâce à leurs antivirus, IDS et systèmes de supervision sécurité, ou encore lors d'analyses forensics. Il est alors nécessaire d'analyser rapidement ces fichiers pour déterminer s'il s'agit de code malveillant connu, d'une attaque ciblée ou bien d'une fausse alerte. Connaître le comportement d'un code malveillant est capital pour déterminer si d'autres machines sur le réseau peuvent être compromises ou non, et pour décider comment réagir.

L'analyse statique de code exécutable par désassemblage a beaucoup de succès et permet d'obtenir d'excellents résultats, cependant la complexité de ce processus demande un long apprentissage, beaucoup de temps et d'énergie. De plus il n'est pas rare de rencontrer des codes malveillants protégés contre le désassemblage et le débogage, ce qui peut ralentir considérablement l'analyse.

L'analyse dynamique de code malveillant est une autre méthode qui consiste à faire exécuter un échantillon de code sur une plate-forme conçue pour observer toutes ses actions. Dans la plupart des cas, c'est une approche très efficace et rapide pour déterminer la nature et le comportement du code malveillant, au moins dans un premier temps.

Cet article décrit comment il est possible d'installer facilement et à moindre coût un modeste laboratoire d'analyse dynamique de code malveillant, même sans compétences en désassemblage.

2 Malware, code malveillant, maliciel

Il existe plusieurs sortes de codes malveillants, dont les définitions sont relativement bien connues mais peuvent varier : Virus, vers, chevaux de Troie, portes dérobées (backdoors), bombes logiques, keyloggers, espiogiciels (spyware), rootkits, botnets, ... (cf. [1] par exemple)

Au-delà de ces définitions, certaines caractéristiques sont particulièrement importantes pour l'analyse de code malveillant, notamment :

- capacité à se répliquer
- techniques de camouflage
- connexion réseau externe pour contrôle à distance
- protection contre le désassemblage et/ou le débogage
- détection de machine virtuelle
- nature du code malveillant : fichier exécutable, document malformé contenant un exploit, script dans une page web, ...

3 Méthodes d'analyse de code malveillant

3.1 Objectifs

Voici quelques-uns des objectifs habituels d'une analyse de code malveillant :
- Vérifier si un fichier suspect est effectivement malveillant ou s'il s'agit d'un faux positif.
- Déterminer s'il s'agit d'un code malveillant générique (connu ou non) ou bien d'une attaque ciblée.
- Obtenir des informations sur la source de l'attaque.
- Déterminer toutes les actions du code malveillant sur le système local : fichiers créés/modifiés/supprimés, processus lancés/stoppés, clés de registre, services installés, ...
- Déterminer toutes ses actions sur le réseau : connexions sortantes, réplication, serveur en écoute, ...
- Déterminer ses "fonctionnalités" malveillantes : virus, ver, cheval de Troie, porte dérobée, bombe logique, rootkit, écoute clavier ou réseau, téléchargement, relais de spam, déni de service, botnet, ...
- Déterminer quelles sont les machines ciblées et lesquelles pourraient être vulnérables ou déjà compromises sur le réseau.
- Fournir des informations pour mieux cibler une éventuelle analyse forensics d'autres machines (noms de processus, empreintes MD5 de fichiers, clés de registre, mots clés, ...)
- Décider quelles actions entreprendre pour gérer l'incident.
- Comprendre comment la compromission s'est déroulée afin d'améliorer les protections à l'avenir.

Pour tout cela il existe plusieurs méthodes complémentaires d'analyse de code malveillant :

3.2 Analyse statique de fichiers

L'analyse statique de fichiers consiste à explorer le contenu des fichiers suspects l'aide de divers outils, dans le but d'extraire le maximum d'informations sans exécuter le code malveillant qu'ils pourraient contenir. A ce stade il s'agit uniquement d'observer le contenu "visible" sans tenter de désassembler le code binaire. Le premier outil à employer dans ce cas est toujours un simple afficheur texte/hexadécimal.

Une analyse directe de ce type permet souvent de révéler beaucoup d'informations utiles grâce aux chaînes de caractères qui apparaissent en clair, aux entêtes de fichiers, aux méta-données, etc. Il est même parfois possible d'extraire des données camouflées par des méthodes simples (i.e. "XOR"), sans recourir à des méthodes d'analyse sophistiquées. Dans le cas de document malformés (par exemple des exploits pour MS Office), des outils de "file carving" [2] peuvent souvent extraire les fichiers exécutables malveillants stockés à l'intérieur.

Voici quelques exemples d'outils typiques employés dans ce type d'analyse :

- Afficheur ou éditeur texte/hexadécimal
- Antivirus
- Scanner d'exploits connus
- Outils pour rechercher des motifs ou des mots-clés particuliers
- Outils d'analyse d'exécutables
- Explorateur de documents OLE (MS Office)
- Outils de File carving

Voici le type d'informations utiles qu'il est souvent possible d'extraire :

- Types de fichiers
- Présence de code malveillant connu ou non
- Adresses IP, URLs, noms de domaines ou de machines (cibles du code malveillant, serveur de téléchargement ou bien connexion retour vers un serveur de contrôle)
- Date de création des fichiers, auteur, organisation et autres informations (métadonnées dans les documents)
- Encodages utilisés (qui peuvent trahir une provenance exotique)
- Présence possible de shellcodes ("NOP sleds" : typiquement une longue suite d'octets 0x90)
- Mots clés particuliers qui peuvent indiquer une attaque ciblée : nom de l'entreprise, noms de serveurs, d'applications ou d'utilisateurs spécifiques, mots de passe internes, etc.

3.3 Désassemblage

Le désassemblage statique consiste à ouvrir un fichier exécutable dans un outil comme IDA Pro [19] afin de retrouver le code assembleur correspondant au contenu binaire du fichier. Certains produits proposent même des fonctions de décompilation pour reconstruire (dans une certaine mesure) le code source de haut niveau employé au départ (en géneral du C).

A la lecture du code assembleur il est alors possible de deviner le comportement de chaque partie du programme, pour en déduire toutes les fonctionnalités.

L'avantage principal de cette méthode est que (en théorie) tous les chemins d'exécution du programme sont visibles "à plat". Il est donc possible d'analyser l'intégralité du code y compris certaines parties qui ne s'exécutent que sous des conditions particulières.

Cette méthode permet également de trouver d'éventuelles routines de camouflage souvent employées pour masquer les données "sensibles" du code malveillant : mots de passe, adresses IP ou URLs de serveurs de contrôle par exemple.

Cependant le désassemblage a quelques inconvénients :

- Il requiert un opérateur très compétent, formé pour lire et analyser du code assembleur, qui connaît bien la correspondance entre ce code, les langages de haut niveau et les appels système.
- Une analyse complète d'un fichier peut demander beaucoup de temps (de plusieurs jours à plusieurs semaines) et d'énergie, en fonction de la taille et de la complexité du code.
- De nombreux codes malveillants sont protégés contre le désassemblage à l'aide de techniques de compression ou de chiffrement (packers, encrypters). Ces protections ne sont jamais parfaites, mais certaines peuvent demander beaucoup d'efforts et ralentissent l'analyse.

3.4 Débogage

Le débogage consiste à exécuter pas-à-pas le code malveillant en employant un débogueur (comme Ollydbg, SoftIce, Syser Debugger ou encore Immunity Debugger), afin d'analyser son comportement en suivant le code assembleur, les registres du processeur et les données en mémoire.

C'est une méthode souvent employée en complément du désassemblage, utile pour mieux comprendre certaines parties du code ou pour contourner certaines protections.

Comme pour le désassemblage, cette méthode nécessite un opérateur très compétent, et les auteurs de code malveillant emploient également des techniques pour détecter ou perturber le débogage.

3.5 Analyse dynamique, analyse comportementale

A l'inverse des méthodes statiques, l'analyse dynamique consiste à exécuter réellement le code malveillant dans un environnement adéquat afin d'observer toutes ses actions et d'en déduire son comportement global. Ce type de méthode est parfois appelée analyse comportementale, ou encore "dynamic malware analysis", "runtime analysis", "live analysis" ou "behavioral analysis" en anglais.

Cette méthode apporte plusieurs avantages par rapport au désassemblage statique :

- Elle peut révéler le comportement d'un code malveillant dans un temps très court. Souvent quelques minutes sont suffisantes pour obtenir les principaux résultats.
- Elle n'est pas sensible aux techniques de protection anti-désassemblage et anti-débogage.
- Elle ne requiert pas autant de connaissances en assembleur que le désassemblage ou le débogage.

Cependant, l'analyse dynamique n'est pas efficace dans tous les cas :

- Si certaines fonctionnalités du code malveillant nécessitent des conditions ou des événements particuliers pour s'exécuter, elles peuvent ne pas être détectées lors de l'analyse.
- Certains codes malveillants peuvent avoir été conçus pour détecter les outils d'observation et ne pas s'exécuter dans ce cas.

Au final ce type de méthode est très efficace pour analyser rapidement des codes malveillants relativement simples, dont le comportement est automatique et reproductible. C'est par exemple le cas typique d'un cheval de Troie qui s'installe sur une machine et qui cherche ensuite à se connecter à l'extérieur pour recevoir des commandes, ou celui d'un ver qui cherche à infecter d'autres machines vulnérables sur le réseau.

Dans le cas de codes malveillants plus complexes, l'analyse dynamique constitue essentiellement une première étape rapide, qui est souvent suivie d'un désassemblage/débogage lorsque cela est jugé nécessaire.

3.6 Légalité de l'analyse de codes malveillants

Dans la loi française, les articles concernant la rétro-conception ne couvrent pas explicitement le cas de l'analyse de codes malveillants. En théorie cette activité n'est donc pas autorisée, même si elle est a priori indispensable, notamment chez les éditeurs d'antivirus et autres IDS/IPS. Il ne semble cependant pas exister de jurisprudence, et on voit mal un auteur de code malveillant poursuivre quelqu'un en justice pour avoir analysé son oeuvre.

En revanche, la rétro-conception est autorisée dans le but d'assurer l'intéropérabilité entre logiciels. On pourrait donc considérer que cela couvre l'analyse de codes malveillants, puisqu'il s'agit justement d'assurer leur *NON*-interopérabilité avec les systèmes protégés ! ;-)

4 Un laboratoire d'analyse dynamique de codes malveillants

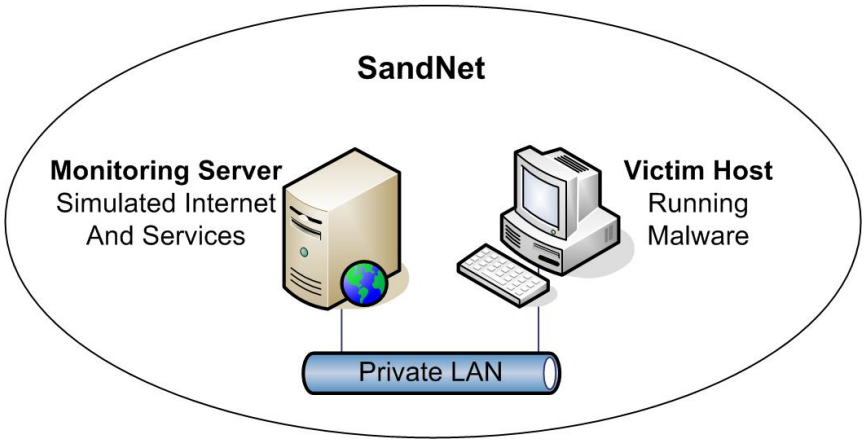
Ce chapitre décrit comment construire simplement et à coût réduit un laboratoire d'analyse de codes malveillants pour effectuer des analyses statiques et dynamiques, en utilisant essentiellement des outils gratuits ou libres.

L'objectif principal est d'analyser soit des fichiers exécutables soit des documents malformés, sur une plate-forme Microsoft Windows. Il serait bien sûr possible cette approche à d'autres systèmes.

Le laboratoire nécessite uniquement deux ordinateurs reliés par un réseau indépendant (i.e. un câble croisé ou un petit concentrateur), appelés "poste victime" (victim host) et "poste d'observation" (monitoring server). Ces deux postes peuvent aussi être constitués de deux machines virtuelles sur une seule machine physique.

Le poste victime contient le système ou est exécuté le code malveillant, tandis que le poste d'observation permet de simuler une connexion Internet et d'enregistrer toute activité réseau.

Ce type d'architecture a déjà été évoqué dans plusieurs articles comme [3,4,5], on peut le désigner par le terme "sandnet" par extension du terme classique "sandbox" (bac à sable).



4.1 Poste d'observation

Le poste d'observation a deux principales fonctions :

- Il enregistre toute activité réseau provenant du poste victime, pour détecter les éventuelles tentatives de connexion du code malveillant.
- Il permet de simuler une connexion à Internet afin de placer le code malveillant dans un environnement où il peut tenter de se connecter à des serveurs externes. Pour cela il peut être nécessaire de simuler divers services comme DNS, HTTP, SMTP, FTP ou encore IRC. Il faut également pouvoir simuler n'importe quelle adresse IP sur demande du code malveillant.

Bien sûr ce serveur ne doit pas être vulnérable vis-à-vis du code malveillant : il doit être suffisamment durci, sécurisé et maintenu à jour. Il est recommandé d'employer un système d'exploitation différent du poste victime pour limiter les risques si jamais le code malveillant emploie un "zero-day" pour se répliquer.

Il est possible d'employer des outils de détection d'intrusion comme Snort [20] ou Bro [21] pour détecter les tentatives d'attaque réseau connues. Dans ce cas il est utile d'employer les signatures Bleeding Threats [22] pour améliorer la détection de malware connus, comme l'indique [23].

4.2 Poste victime

Le poste victime est l'ordinateur où le code malveillant est lancé. Il doit si possible contenir le même système d'exploitation et les mêmes applications que le système à protéger ou que celui où le

spécimen a été capturé. Par exemple si le fichier suspect est un classeur Excel détecté sur un poste Windows XP SP2 avec MS Office XP, alors le poste victime devrait avoir la même configuration.

Au contraire des postes habituels, le but ici est de pouvoir exécuter le code malveillant sans le bloquer. Il est donc nécessaire (sauf cas particulier) de désactiver ou désinstaller toute protection qui pourrait perturber l'analyse :

- antivirus (à l'accès)
- pare-feu personnel
- logiciel de scan automatique
- HIPS (système de prévention d'intrusion local, maintenant intégré à de nombreux antivirus et pare-feux personnels)
- paramètres de sécurité trop restrictifs

La configuration réseau du poste victime doit aussi être adaptée pour diriger toute activité réseau ou Internet vers le poste d'observation :

- Passerelle par défaut
- Serveur DNS
- Fichier "hosts"
- Proxy web

S'il est nécessaire d'observer les capacités d'auto-réplication du code malveillant, il est possible d'étendre l'architecture réseau en ajoutant un ou plusieurs postes victimes supplémentaires connectés au même concentrateur.

4.3 Virtualisation

L'architecture de test peut être installée soit sur deux machines physiques, soit sur deux machines virtuelles. Aucune des deux solutions n'est idéale pour couvrir tous les cas de figure, il est donc utile de prévoir les deux possibilités.

Avantages de la virtualisation :

- Coût et flexibilité : une seule machine physique permet de simuler deux machines ou plus, et de construire différentes architectures réseau suivant les besoins.
- Sécurité : les machines virtuelles peuvent être connectées par un réseau virtuel totalement indépendant de tout réseau opérationnel, sans risque d'infecter d'autres machines.
- Rapidité et efficacité : une machine virtuelle peut être stoppée, restaurée et redémarrée en quelques secondes.
- Facilité d'emploi : n'importe quel état de fonctionnement peut être sauvegardé dans un "snapshot"en quelques secondes. Il est ensuite très simple de restaurer tout snapshot précédent, et de comparer plusieurs états entre eux.
- Les machines virtuelles peuvent être facilement copiées, dupliquées et modifiées pour constituer une bibliothèque de toutes les versions d'un système d'exploitation ou d'une application.

Inconvénients de la virtualisation :

- Il existe de nombreuses techniques connues utilisables par un code malveillant pour détecter s'il s'exécute dans une machine virtuelle, comme par exemple [6]. Dans ce cas il peut éviter de s'exécuter ou perturber l'analyse.

– Certains environnements de virtualisation peuvent être vulnérables et permettre à un code malveillant spécifiquement conçu de "s'échapper" d'une machine virtuelle pour compromettre le système hôte. Il est donc utile de prendre quelques précautions lors de toute exécution de code malveillant.

Avantages des machines physiques :
– Au contraire des machines virtuelles, tout code malveillant devrait normalement s'exécuter sans perturbation sur une machine physique, tant qu'il n'est pas conçu pour détecter les outils d'observation employés.
– Il est très simple de s'assurer que le réseau physique constitué des quelques machines de test est bien déconnecté de tout autre réseau.

Inconvénients des machines physiques :
– Il est nécessaire d'employer des outils de sauvegarde/restauration d'images disque à chaque analyse, et à chaque fois qu'un état particulier doit être sauvegardé. Cette tâche prend beaucoup de temps et peut vite se révéler pénible à la longue, même s'il est possible de l'automatiser en partie.
– Il n'est pas possible de "geler" le système pour analyse et reprendre l'exécution sans que le code malveillant puisse s'en apercevoir.

Configuration sécurisée de VMware : VMware Workstation est un des outils de virtualisation les plus utilisés aujourd'hui pour l'analyse de code malveillant. Il a fait l'objet de nombreuses recherches dans ce domaine et il existe de nombreux moyens connus pour détecter son utilisation depuis une machine virtuelle, par exemple [6]. Il est cependant possible de se protéger contre certaines méthodes de détection en modifiant simplement quelques paramètres des machines virtuelles :
– Désactiver les options d'accélération
– Désactiver ou désinstaller les VMware tools
– Désactiver les fonctions de presse-papier, drag&drop et dossiers partagés
– Désactiver toute connexion réseau depuis une VM vers une machine physique (Host-only, Bridged ou NAT)
– Désactiver la connexion automatique de périphériques USB
– Les lecteurs CD/DVD ne doivent pas être connectés au démarrage, sauf s'ils correspondent à une image ISO

4.4 Similitude avec les pots de miel (honeypots)

L'analyse dynamique de codes malveillants a beaucoup de points communs avec la technique des pots de miel (honeypots, cf. [25]), puisque dans les deux cas il s'agit de faire exécuter du code malveillant dans un environnement contrôlé pour observer son comportement. Il devrait donc être possible de réutiliser certains outils de type honeypot pour rendre l'analyse de code malveillant plus efficace.

On peut tout de même noter quelques différences importantes :
– Un honeypot est généralement conçu pour être connecté à Internet afin d'attirer d'éventuels attaquants. Un système d'analyse de malware est plus souvent déconnecté de tout réseau, il doit donc simuler l'équivalent d'une connection Internet.

– Un honeypot peut générer beaucoup d'activité et capturer beaucoup de fichiers dont la plupart n'est pas utile. Une fonction primordiale est donc de pouvoir détecter automatiquement (ou pas) ce qui est réellement intéressant et ce qui ne l'est pas. A l'inverse, un fichier exécuté sur un système d'analyse est déjà suspect, il n'y a donc pas besoin d'effectuer ce tri parmi de nombreux fichiers.

Il est donc utile de réutiliser et adapter certaines fonctionnalités des honeypots actuels, notamment les services système et réseau qui permettent de simuler un système vulnérable réaliste et enregistrer toutes les actions du code malveillant (cf. [26,28]).

5 Outils

Voici une liste (non exhaustive) d'outils permettant de construire un laboratoire d'analyse de codes malveillants. Cette liste contient essentiellement des logiciels gratuits ou libres.

5.1 Sur le poste d'observation

– Un outil de capture réseau : Wireshark, tcpdump ou équivalent.
– En option un outil de détection d'instrusion réseau : Snort, Bro.
– Un outil de scan de ports : nmap ou équivalent.
– Un émulateur de serveur DNS pour rediriger toutes les requêtes vers le poste d'observation : pyminifakedns [30], Malcode Analysis Pack [29] ou encore fauxdns.pl [35].
– Divers émulateurs de services réseau (HTTP, SMTP, FTP, POP3, IRC, ...) permettant d'enregistrer toutes les actions du code malveillant : fauxservers [35], scripts Honeyd [26] ou par exemple des scripts adaptés basés sur les serveurs fournis en standard avec Python (HTTPServer, smtpd) ou Perl.

5.2 Sur le poste victime

– Afficheur ou éditeur texte/hexadécimal
– Antivirus : plusieurs si possible, avec détection "à l'accès" désactivée.
– Scanner d'exploits connus : OfficeCat [7], Fess [8]
– Outils pour rechercher des motifs ou des mots-clés particuliers : grep, reScan [18], scripts maison
– Outils d'analyse d'exécutables : PEiD [9], pefile [10], Dependancy Walker [11], Scanbin [12]
– Explorateur de documents OLE (MS Office) : SSView [16], OleFileIO [17]
– File carving : Hachoir-subfile [13], Foremost [14], Scalpel [15]
– Process Monitor, Process Explorer, Autoruns, Strings [31]
– SysAnalyzer [29]
– interpréteurs Python, Perl ou tout autre langage de script pour automatiser certaines actions répétitives.

6 Méthodologie d'analyse

6.1 Préparation

Tout d'abord il est nécessaire de mettre à jour tous les outils contenant des signatures avant chaque analyse, en particulier les antivirus, IDS, ainsi qu'OfficeCat et Fess.

6.2 Analyse statique de fichiers

Cette phase de l'analyse consiste à extraire toutes les informations possibles des fichiers suspects sans avoir à les exécuter ou à les ouvrir dans l'application associée.

Comme indiqué précédemment, une bonne pratique est toujours de commencer par ouvrir un fichier dans un éditeur hexadécimal, afin de se faire une première idée rapide sur sa nature et son contenu réels.

Analyser un fichier exécutable :

1. Tester le fichier avec tous les antivirus disponibles, pour vérifier s'il s'agit d'un code malveillant connu ou d'une variante. Si oui, consulter les détails fournis par les éditeurs antivirus. La plupart des antivirus fournissent une interface ligne de commande permettant de scripter l'analyse.
2. Détecter si le fichier est compressé ou chiffré : PEiD pour détecter un packer connu, pefile pour calculer l'entropie des différentes sections sinon. Une entropie élevée (entre 7 et 8) est généralement signe d'une section compressée ou chiffrée.
3. Observer les noms de section et les valeurs de l'entête PE (pefile, scanbin) : Des valeurs non usuelles peuvent également trahir un packer.
4. Extraire toutes les chaînes en clair (Strings).
5. Rechercher des motifs ou mots-clés intéressants (grep, reScan) : adresses IP, adresses e-mail, noms de domaines, URLs, noms de fichiers exécutables ou particuliers, noms de fonctions Win32/JavaScript/VBScript particulières, etc.
6. Rechercher des mots-clés liés à l'entreprise, qui pourraient indiquer une attaque ciblée : nom de l'entreprise, noms de projets, de divisions internes, de personnels, de serveurs ou de machines internes, etc.
7. Tester en série divers algorithmes de camouflage simples du style XOR ou décalage, en faisant varier la clé et en recherchant les mêmes motifs que précédemment. Dans la pratique beaucoup d'auteurs de code malveillant ne sont pas des experts en cryptographie, heureusement pour nous. ;-)

Analyser un document malformé : Ces dernières années il est devenu de plus en plus courant de détecter des codes malveillants camouflés dans des documents malformés. Ces fichiers peuvent être de divers types : MS Office, JPEG, GIF, archives, etc. Ils contiennent en général un exploit correspondant à une vulnérabilité dans un logiciel courant, qui déclenche un shellcode de taille plus ou moins réduite. Ce shellcode peut se suffire à lui-même, mais il s'agit souvent d'un code intermédiaire pour activer un fichier exécutable plus évolué. Ce fichier exécutable se trouve généralement stocké dans le même document, ou bien il doit être téléchargé depuis un serveur sur Internet. Il est ensuite écrit sur le disque pour assurer la persistance du code malveillant, sous forme camouflée ou non.

Voici un exemple de méthodologie pour analyser un document suspect :

1. Tester le fichier avec tous les antivirus disponibles, pour vérifier s'il contient un code malveillant connu ou une variante. Certains antivirus possèdent des signatures génériques pour détecter la plupart des exploits connus, mais dans la pratique la couverture n'est pas toujours aussi efficace qu'on pourrait l'espérer... (cf. [32])

2. Pour un document MS Office : explorer la structure OLE et les méta-données (SSView, OleFileIO), détecter les éventuelles erreurs dans la structure OLE (OleFileIO). Ces outils permettent également d'extraire les flux OLE internes s'il s'agit de fichiers camouflés à l'intérieur du document. Les méta-données révèlent parfois des informations utiles (date de création, auteur, société), car les auteurs de malware peuvent aussi oublier de nettoyer leurs documents...
3. Détecter certains exploits connus (OfficeCat, Fess).
4. Rechercher des motifs ou mots-clés intéressants (grep, reScan) : comme pour les fichiers exécutables, en y ajoutant la détection de séries de NOPs (0x90 par exemple) qui accompagnent souvent un exploit.
5. Utiliser des outils de file carving (hachoir-subfile, foremost, scalpel) pour extraire tous les fichiers exécutables qui pourraient être stockés en clair dans le document. Comme pour les exécutables il peut être utile de refaire ce test après divers décodages simples (XOR par exemple).

6.3 Analyse dynamique

Cette seconde phase consiste à exécuter ou ouvrir le document afin de déclencher le code malveillant et observer/enregistrer toutes ses actions.

Préparation :

1. S'assurer que le poste victime et le poste d'observation sont dans un état stable et sain. Si besoin, restaurer une image complète du système.
2. Sauvegarder une image la plus complète possible du système initial du poste victime, qui servira de référence : soit une image du disque, soit une liste détaillée des fichiers du disque (par exemple "dir C :\ /s /a /o :n ¿list.txt") et de la base de registre. On peut également sauvegarder une liste des empreintes de fichiers grâce à un outil de type "md5sum". Cette image peut être soit obtenue depuis le système
3. Sauvegarder la liste des processus actifs, des clés de registre intéressantes (Autoruns), des utilisateurs, des services Windows, et de toute autre information susceptible d'être modifiée par un code malveillant.
4. Démarrer les services réseau émulés sur le poste d'observation.
5. Démarrer et configurer tous les outils d'observation sur les deux postes : Wireshark, Snort, Process Monitor, etc.

Exécution :

1. Lancer ou ouvrir le fichier suspect. Il est généralement préférable de le faire depuis un compte administrateur pour éviter de brider le code malveillant.
2. Observer les actions sur les deux postes, pour voir les nouveaux processus et les tentatives de connexion réseau.
3. Observer toutes les modifications sur le système de fichiers et la base de registre (Process Monitor). Faire une copie de tout fichier créé par le code malveillant, au cas où il serait effacé par la suite.

4. Si le code malveillant essaye de se connecter à une adresse IP précise au lieu de faire une requête DNS, il peut être nécessaire de modifier la configuration réseau du poste d'observation pour qu'il soit accessible à cette adresse.
5. Il peut aussi être nécessaire de simuler un nouveau service s'il essaye de se connecter à un port non ouvert.
6. Utiliser Sysanalyzer pour observer tous les appels système intéressants. Cet outil est aussi parfois capable de sauvegarder une image mémoire du processus malveillant, ce qui peut être très utile si le fichier exécutable est chiffré par un packer sur le disque.
7. Si les actions observées ne semblent pas complètes (pas de nouveau processus, pas de fichier créé), cela peut venir de plusieurs raisons :
   (a) Soit le code malveillant n'a pas pu s'exécuter car il n'est pas dans le bon environnement (version du système, application non vulnérable).
   (b) Soit il a détecté qu'il était dans un environnement de test (virtualisation, outils employés, etc).
   (c) Soit il s'agit d'un rootkit capable de camoufler ses actions. Dans ce cas il peut être utile d'employer des techniques de "forensics mémoire" comme décrit dans [24], et de comparer les images disque complètes avant et après compromission.
8. Observer l'évolution pendant quelques minutes : le code malveillant peut effectuer des actions programmées dans le temps.

Après exécution :

1. Sauvegarder une image complète du système du poste victime en suivant la même méthode que précédemment lors de la préparation.
2. Il peut être utile de redémarrer la machine en employant un CD de démarrage (par exemple Linux ou BartPE) afin de sauver cette image depuis un système sain. Cette précaution permet de s'affranchir d'un éventuel rootkit pour détecter les fichiers camouflés.
3. Comparer les deux images sauvegardées avant et après exécution, pour détecter toutes les modifications : fichiers, clés de registre, processus, services, etc.
4. Il pourrait également être utile de placer le système en hibernation, ou d'utiliser un snapshot d'une machine virtuelle, afin d'inspecter la mémoire si le code malveillant n'est pas visible sur le disque (cf. [24]).

6.4 Phases suivantes : Désassemblage et Débogage

Les deux premières phases (analyse statique et dynamique) permettent d'obtenir rapidement une vue partielle du code malveillant et de son comportement.

Si le temps le permet et si des fichiers intéressants ont été obtenus, il est alors possible d'analyser le code malveillant plus en profondeur, à l'aide de techniques de désassemblage et de débogage. Ces techniques, déjà très bien documentées, ne seront pas décrites dans cet article.

6.5 Document malformé : Déterminer quelle vulnérabilité est exploitée

Quand un document malformé exploite une vulnérabilité pour exécuter du code malveillant, un des objectifs est d'identifier quelle vulnérabilité est concernée, afin de mieux identifier le risque et les autres machines vulnérables.

Si les outils à disposition (antivirus, OfficeCat, Fess) ne fournissent pas suffisamment d'information, une méthode simple permet d'en savoir plus :

1. Commencer avec une version ancienne de l'application, non mise à jour.
2. Installer pas-à-pas chacun des correctifs de sécurité publiés.
3. Après chaque correctif, sauvegarder une nouvelle image saine du système. Il est utile de les conserver pour de futures analyses, si l'espace disque le permet.
4. A chaque fois, ouvrir ensuite le fichier suspect, et vérifier si le code malveillant s'exécute ou non. S'il ne s'exécute pas, la vulnérabilité exploitée correspond donc à ce correctif.
5. Cependant, un correctif corrige souvent plusieurs vulnérabilités. Il est alors nécessaire de corréler cette information avec les résultats des autres outils (OfficeCat, Fess, antivirus) et les informations disponibles en ligne (CVE, BugTraq, vendeur) pour deviner quelle(s) vulnérabilité est(sont) exploitée(s).

6.6 Obtenir des informations à propos de la source de l'attaque

Il est parfois possible d'obtenir plus d'informations sur la source apparente d'une attaque en exploitant les différentes données collectées lors de l'analyse (adresses IP, noms de domaines et de serveurs sur Internet). Pour cela de nombreux outils existent pour faire des requêtes auprès de bases de données publiques : DNS, Whois [33], rWhois [34], etc.

Il faut cependant se méfier des résultats obtenus car les codes malveillants emploient souvent des sites temporaires. Un nom de domaine peut changer de propriétaire très rapidement, donc ce type de requête doit être fait et sauvegardé le plus tôt possible lors de l'analyse.

7 Améliorations possibles

L'architecture de test et la méthodologie décrites dans cet article sont avant tout conçue pour être simples et peu coûteuses. Il est bien sûr possible d'améliorer les résultats et l'efficacité du processus en ajoutant d'autres outils ou des logiciels commerciaux.

7.1 Analyse connectée ou semi-connectée

Dans certains cas il peut être nécessaire de fournir une connexion Internet réelle au code malveillant pour pouvoir analyser son comportement de façon plus complète. Bien sûr cela est plus risqué car le code malveillant pourrait se répliquer, et son auteur peut voir qui exécute le code si celui-ci se connecte à un serveur de contrôle.

Dans cette configuration, le poste victime peut être considéré comme un honeypot à haute interaction [25]. Certains outils permettent de restreindre la connection du code malveillant à Internet pour éviter sa réplication.

7.2 Automatisation

La méthodologie décrite emploie un grand nombre de petits outils indépendants pour effectuer des tâches simples. C'est donc un processus manuel très lourd et sujet à erreurs, en particulier lorsqu'il faut analyser un grand nombre de fichiers suspects.

Il serait possible d'automatiser une grande partie de ces tâches en tirant partie des langages de script et des APIs disponibles pour contrôler certains logiciels (par exemple VMware) :

- l'analyse de nombreux fichiers par des outils de détection (antivirus, OfficeCat, Fess, ...)
- la sauvegarde des images système avant et après exécution
- la comparaison de ces images
- le lancement des outils d'observation
- le lancement des fichiers suspects, et la détection des actions (processus, fichiers, registre, etc)
- l'arrêt de l'analyse après un temps fixé ou sous certaines conditions

Cependant, quelques tâches nécessitent réellement une action manuelle (notamment l'analyse statique et l'exploitation des résultats), et certains outils graphiques peuvent difficilement être contrôlés par script.

Des outils comme [35] proposent une base intéressante pour bâtir un système d'analyse dynamique totalement automatisé.

7.3 Outils et services d'analyse dynamique avancés

Depuis quelques années sont apparues diverses solutions commerciales pour effectuer des analyses dynamiques de code malveillant de façon plus efficace et robuste. Elles sont basées sur différentes technologies (virtualisation modifiée, simulation d'API Windows, API hooking, ...), mais toutes ces méthodes visent à tracer le flot d'exécution d'un code malveillant pour en déduire son comportement et ses caractéristiques.

Certaines solutions sont accessibles en ligne, via une interface de dépôt de fichier, comme [37,36,38,39,40]. Ces interfaces sont généralement gratuites, mais la confidentialité des résultats n'est pas assurée, et l'environnement de test est rigide puisqu'il est impossible de choisir une plateforme spécifique. De plus, il n'est généralement proposé que l'analyse de fichiers exécutables Win32, les documents malformés ne sont donc pas couverts.

Certaines de ces solutions comme [36,38] sont aussi vendues comme des produits afin de pouvoir les installer localement dans un environnement qui peut être configuré suivant les besoins.

Avantages :

- Automatisation : permet l'analyse automatisée de très nombreux fichiers.
- Résultats plus complets et reproductibles
- Fiabilité améliorée

Inconvénients :

- Coût d'acquisition et de maintenance élevé, de l'ordre de plusieurs dizaines de milliers d'euros par an. Ces produits sont donc plutôt destinés aux organisations et entreprises ayant besoin d'analyser régulièrement de très nombreux fichiers suspects.
- Manque de confidentialité et de flexibilité pour les solutions gratuites en ligne.

7.4 Nouvelles fonctionnalités de VMware

Les futures versions de VMware Workstation et ESX devraient intégrer de nouvelles fonctionnalités extrêmement utiles pour l'analyse dynamique de code malveillant, notamment :

- VProbes : interface limitée de supervision pour VMware Workstation 6.5, permettant d'observer les registres du processeur, les processus en cours ainsi que quelques autres informations.
- Interface compatible gdb : pour déboguer le noyau de la machine virtuelle.
- VMSafe : interface fournie par l'hyperviseur de VMware ESX permettant de contrôler en temps réel toute la mémoire, les fichiers et les connexions réseau de la machine virtuelle. Toute lecture/écriture/exécution du code malveillant en mémoire pourrait alors être détectée et tracée, et les outils d'observation resteraient indétectables car situés à l'extérieur de la machine virtuelle.

8 Conclusion

Il est possible de bâtir un laboratoire simple à moindre coût pour effectuer des analyses dynamiques de code malveillant. La méthode décrite dans cette article est loin d'être parfaite, mais elle a l'avantage d'être accessible à la plupart des organisations ou entreprises ayant besoin d'analyser des fichiers suspects et qui ne disposent pas toujours d'outils avancés et de personnels formés au désassemblage.

Plusieurs pistes d'amélioration ont été également proposées pour bâtir un système plus complet et professionnel pour l'analyse de codes malveillant.

D'éventuelles mises à jour de ce document seront disponibles en version électronique sur les sites `http://actes.sstic.org` ou `\http://www.decalage.info` après juin 2008.

Références

1. Logiciel malveillant, Wikipedia, `http://fr.wikipedia.org/wiki/Logiciel_malveillant`
2. File carving, `http://www.forensicswiki.org/wiki/Carving`
3. An Environment for Controlled Worm Replication and Analysis or : Internet-inna-Box, Ian Whalley, Bill Arnold, David Chess, John Morar, Alla Segal, Morton Swimmer, IBM TJ Watson Research Center, `http://www.research.ibm.com/antivirus/SciPapers/VB2000INW.htm`
4. Malware Analysis for Administrators, S. G. Masood, SecurityFocus, 2004-05-20, `http://www.securityfocus.com/infocus/1780`
5. Behavioral Malware Analysis Using Sandnets, Joe Stewart, SecureWorks, `http://www.sv-issa.org/sandnets.pdf`
6. Trapkit - VMM fingerprinting, `\\url{http://www.trapkit.de/research/vmm/index.html}`
7. OfficeCat, SourceFire, `http://www.snort.org/vrt/tools/officecat.html`
8. Fess, SecureWorks, `http://www.secureworks.com/research/tools/fess.html`
9. PEiD, `http://peid.has.it/`
10. pefile, `http://dkbza.org/pefile.html`
11. Dependancy Walker, `http://www.dependencywalker.com/`

12. Scanbin, http://jc.bellamy.free.fr/fr/scanbin.html
13. Hachoir, http://hachoir.org/
14. Foremost, http://foremost.sourceforge.net/
15. Scalpel, http://www.digitalforensicssolutions.com/Scalpel/
16. Structured Storage Viewer, MiTec, http://www.mitec.cz/ssv.html
17. OleFileIO, http://www.decalage.info/python/olefileio
18. reScan, http://www.decalage.info/python/rescan
19. IDA Pro, http://www.hex-rays.com/idapro/
20. Snort, http://www.snort.org/
21. Bro Intrusion Detection System, http://www.bro-ids.org/
22. Bleeding Threats, http://www.bleedingthreats.net/
23. Network Forensic : cherchez les traces sur le réseau, Guillaume Arcas, MISC magazine 35, janvier-février 2008
24. Analyse post mortem tout en mémoire sous Windows, Nicolas Ruff, Samuel Dralet, MISC magazine 35, janvier-février 2008
25. Honeypots, Wikipedia, http://en.wikipedia.org/wiki/Honeypot_%28computing%29
26. Honeyd, http://www.honeyd.org/
27. The Honeynet Project, http://www.honeynet.org/
28. mwcollect : Nepenthes, HoneyTrap, HoneyBow, http://www.mwcollect.org/
29. Malcode Analysis Software Tools, iDefense labs, http://labs.idefense.com/software/malcode.php
30. Mini Fake DNS Server, http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/491264
31. Windows Sysinternals, http://technet.microsoft.com/en-us/sysinternals
32. Investigating and responding to suspicious Office files, Maarten Van Horenbeeck, SANS Internet Storm Center, http://isc.sans.org/diary.html?storyid=2967
33. Whois, RIPE, http://www.db.ripe.net/whois
34. RWhois, http://www.rwhois.net/rwhois/prwhois.html
35. Truman - The Reusable Unknown Malware Analysis Net, Joe Stewart, http://www.secureworks.com/research/tools/truman
36. CWSandbox, http://www.cwsandbox.org/, \http://www.sunbelt-software.com/Developer/Sunbelt-CWSandbox/
37. Anubis : Analyzing Unknown Binaries, http://analysis.seclab.tuwien.ac.at/index.php
38. Norman Sandbox, http://www.norman.com/Product/Sandbox-products/en-us
39. PCTools ThreatExpert, http://www.pctools.com/threat-expert/
40. Joebox, http://www.joebox.org/articles.php