Je suis pleinement conscient(e) que le plagiat de documents ou d'une partie de document constitue une fraude caractérisée.
Nom, date et signature :

Robustesse des outils de sécurité

Monnet-Paquet Aurélien
·
Supervised by : Lauradoux Cédric

Juin 2016

Abstract

Keywords Bombe de compression · IDS · Antivirus · Framework

1 Introduction

2 Outils de sécurité

2.1 Antivirus

Les antivirus (AV) sont les programmes de sécurité les plus utilisés. Ce sont des programmes installer sur les machines des utilisateurs. Les antivirus sont lancés par l'OS avant l'initialisation du système de fichiers et du réseau. Cet outil permet de détecter les malwares et les empêche de s'exécuter sur le système. Il existe différentes manières de détecter un malware :

- Scan par signature : L'AV va calculer la signature d'un fichier ou d'un morceau de code et va le comparer à une base de données. Méthode inefficace contre les malwares polymorphes ou capable de changer leur signature.
- Analyse heuristique : Méthode la plus puissante car elle permet de simuler l'exécution du code d'un programme dans une zone contrôlée. Ainsi, l'AV peut observer le comportement du code qui s'exécute et définir si il s'agit d'un malware ou non. Cette méthode peut provoquer des fausses alertes.
- Controle d'integrité : Méthode qui permet de vérifier qu'un fichier n'a pas été modifié au cours du temps. Les informations comme la taille, la date et l'heure de dernière modifications, la somme de contrôle éventuelle du fichier

Monnet-Paquet Aurelien
Université Grenoble Alpes, 38400 Saint Martin d'Hères
E-mail: monnetpa@e.ujf-grenoble.fr

sont analysées lors de la demande d'ouverture du fichier par l'utilisateur (si analyse en temps réel) ou lors d'un scan de l'AV.
Le but de notre expérience est de tester si il est possible de faire un deni de service lorsque l'AV analyse une bombe de compression. En complément à l'article sur le contournement d'antivirus [6].
Nous avons tester deux AV pour Linux : ClamAV (`www.clamav.net`) et Comodo (`www.comodo.com`)

2.2 Agrégateurs d'antivirus

Les agrégateurs d'AV sont des outils qui permettent d'analyser des fichiers avec plusieurs AV en même temps. Le résultat obtenu en analysant un fichier avec ce type d'outils est plus fiable qu'avec un seul AV. Cependant, cet outil ne doit pas remplacer un AV installer sur une machine. Il existe deux types d'agrégateurs.

2.2.1 Sites web

Ce sont des sites web spécialisés dans la détection de malwares grâce à des dizaines antivirus différents. Un partenariat relie les éditeurs d'AV et un site de ce type. Ce sont ses éditeurs qui mettent à disposition des sites leur programme. Ce qui signifie que les éditeurs sont responsable de leur configuration. Il peut arriver qu'un AV sur ce genre de site ne soit pas exactement configuré comme la version commerciale. En effet, il s'agit d'un bon moyen pour les éditeurs de tester de nouvelles fonctionnalités, configurations. Lorsqu'un fichier est soumis par un utilisateur sur cette platforme, il est analysé par les moteurs des différents AV. Dans le cas où au moins un AV détecte un fichier comme malveillant, le fichier est envoyé aux éditeurs des AV qui ne l'on pas détecter pour l'améliorer. Nous avons choisis 3 sites pour réaliser nos tests :

- VirusTotal : `www.virustotal.com`, filiale de Google basée en Espagne, hébergé dans le cloud, comprenant 57 AV et est le plus populaire avec en moyenne 1.5M de soumissions par jours (dont 16.6% provenant de France).
- Jotti : `virusscan.jotti.org`, organisation basée aux Pays-Bas, hébergé dans le cloud, comprenant 19 AV.
- Virscan : `www.virscan.org`, organisation basée en chine, comprenant 39 AV.

Pour utiliser ce service, un utilisateur doit envoyer le fichier suspect vers le site pour que les moteurs des différents AV puisse l'analyser. Après analyse, ce fichier n'est pas détruit, car il est envoyé aux éditeurs d'AV pour améliorer leurs programmes. Il n'est donc pas concevable pour une personne (physique ou morale) d'envoyer des fichiers avec des données sensibles vers des serveurs dont cette personne n'a pas de contrôle. C'est pourquoi il existe aussi les frameworks de detection de malwares.

2.2.2 Frameworks

Ce genre d'outils fonctionnent donc en local, idéalement sur une machine virtuelle pour éviter de contaminer la machine hôte. Ce sont des programmes qui permettent à un utilisateur d'analyser un fichier suspect de différentes façons via des plugins. Pour notre étude, nous avons utiliser deux frameworks :

- Mastiff, `www.korelogic.com`, société basée au USA, composée de professionnels en sécurité informatique.
- Viper, `www.viper.li`, créer par Claudio Guarnieri.

Ses deux outils sont pré-installer dans la distribution REMnux (`remnux.org`) qu'il est facile d'importer dans Virtualbox `https://www.virtualbox.org`.
Mastiff fonctionne de la manière suivante :
cf image.
Mastiff et Viper permettent entre autre:

- D'identifier et classer le type des fichiers analyser grâce aux règles de YARA, `plusvic.github.io/yara`.
- Reconnaître deux fichiers dont le contenu est très similaire grâce au fuzzy hashing (ssdeep). Prenons par exemple les commandes suivantes : "echo message1 > fichier1.txt" et "echo message2 > fichier2.txt". Les deux fichiers générer auront des hashs très différents avec aucune ressemblance possible. Cependant, grâce à ssdeep il est possible de faire ressortir le fait que ses deux fichiers sont presque identiques. Technique pratique pour repérer les malwares polymorphes.
- Extraire et analyser récursivement des fichiers zip. C'est notamment cette dernière qui nous intéresse et que nous allons tester.

2.3 IDS

Les IDS ou sondes de détection d'intrusion sont des programmes permettant d'analyser le trafic réseau de manière transparente. Idéalement l'IDS est situé au point d'entrer du réseau d'une entreprise pour pouvoir analyser tout le trafic réseau. L'IDS fonctionne sur un système de règles et de pattern matching. Un administrateur définit un certain nombre de règles que l'IDS doit vérifier lors de l'analyse du trafic réseau. Suivant les règles définit, l'IDS permet :

- de laisser passer un paquet sans finir l'analyse.
- rejeter le paquet, en générant des paquets ICMP erreur ainsi qu'une alerte
- générer une alerte dans le fichier de logs.

Par exemple, l'administrateur peut décider de créer une règle [3] qui générera une alerte lors qu'une connexion sera établie sur les serveurs de Facebook. Cette alerte permet d'identifier l'adresse IP source de la requête HTTP.
Le pattern matching fonctionne aussi avec le contenu des paquets qui circulent sur le réseau. De cette manière une règle peut être créer pour générer une alerte

lorsqu'un paquet contient les caractères "Euros 2016" par exemple.
L'IDS que nous avons choisis de tester est Suricata [2]. Suricata se démarque de la concurrence par une analyse multi-threadée. Avec toujours de plus en plus de données, l'analyse rapide et efficace devient de plus en plus important. L'article de David J. Day et Benjamin M. Burns [1] analyse les performances de Suricata. Suricata supporte nativment l'IPv6 et est open source. Suricata extrait les fichiers compressés pour analyser leur contenu [4]. Le but de l'étude est de vérifier qu'il n'y ai pas de déni de service possible lorsque Suricata analyse une bombe de compression.

3 Tests

3.1 Protocole de test

Dans un premier temps, nous avons générer un exécutable Windows (payload via metasploit) connu pour être analyser comme un malware par la plupart des outils disponible sur le marché. Ensuite, pour chacun des outils, nous avons effectuer une analyse témoin. Cette analyse se compose du fichier générer précédemment, et d'un fichier contenant uniquement des zéros et qui est donc complètement inoffensif.
Dans un second temps, nous avons analyser les résultats des deux premiers fichiers compressé dans différents formats : .gz, .lzma, .tar.bz2, .tar.gz, .tar.xz, .lz, .lzo, .zip
Et enfin, nous avons analyser l'impact des bombes de compression sur ses outils pour vérifier leur robustesse sur un éventuel déni de service.
Pour rappel : lorsqu'un fichier compressé de petite taille (42 Ko) se décompresse en un autre fichier d'une taille très largement supérieur (4.5 Po), c'est ce qu'on appelle une bombe de compression (42.zip).
Dans notre expérience, les fichiers décompressés sont de taille : 250 Mo, 500 Mo, 1 Go, 4.5 Po.
Nous avons réaliser les tests (et installer les outils nécessaires) sur une machine ayant un OS Linux (Ubuntu 12.04 LTS).
Après avoir installer ClamAV (via les dépôts Ubuntu/Debian) nous avons effectuer des tests dans sa configuration initiale. Pour lancer un scan dans le répertoire courant, il suffit de la commande "clamscan" (avec la configuration initiale). Cependant, deux paramètres sont à prendre en compte pour menée à bien cette expérience :

- "max-recursion= n" : La récursion lors de l'analyse des fichiers compressés, 16 par défaut.
- "max-filesize= n" : Extrait et analyse n octets de chaque archives. 25 Mo par défaut avec une limite de 4 Go.

Nous avons également installer un second antivirus : Comodo (via package du site officiel). Il dispose d'une interface graphique, mais ne nous permet pas de

modifier des paramètres comme la récursion, ou la taille des fichiers analysés. Pour lancer un scan, il faut suivre les indications sur l'interface graphique.

VirusTotal met à notre disposition une API permettant d'envoyer des fichiers pour analyse via un script. Nous avons alors repris puis modifier un script existant pour effectuer nos tests. Ce script (écrit en Perl) effectue deux requêtes vers VirusTotal. Une première (HTTP POST) pour envoyer le fichier suspect. Et une seconde, pour récupérer le résultat sous forme d'un objet JSON. Ensuite le script génère un rapport mis en forme. Ce rapport contient le résultat de tous les antivirus associés au fichier envoyé.

En ce qui concerne Jotti et Virscan, nous avons utilisé l'interface web pour soumettre nos fichiers et récupérer les résultats.

Mastiff est un programme dont le processus peut être exécuter dans un docker. Dans ce cas, un dossier est partagée entre la VM et le docker. Dans ce dossier, nous avons au préalable placé les fichiers à analyser par Mastiff. Dans le cas présent, nous n'avons pas de fichier confidentiel, c'est pourquoi nous avons ajouter l'option d'envoyer les fichiers à analyser vers VirusTotal automatiquement. Lors de son analyse, Mastiff génère un fichier de résultat par moteur d'analyse.

Viper est aussi un programme dont le processus peut s'exécuter dans un docker. A la différence que Viper met à notre disposition une interface web, à partir de laquelle nous pouvons soumettre nos fichiers. Une fois un fichier soumis, une multitude de commandes peuvent être lancées pour tester le fichier suspect. Les résultats de ses commandes se retrouvent sur cette interface web.

Après avoir installer Suricata, nous avons rédiger des règles ([4]) pour que les fichiers soit décompressés puis inspectés. Nous avons pris soin de modifier la configuration initiale de Suricata pour que l'extraction se passe de manière optimale :

- stream.reassembly.depth = 4gb, après avoir ré-assembler le flux TCP, le fichier ne doit pas dépasser une taille de 4Go.
- request_body_limit = 0 (infini), valeur que le corps de la requête HTTP ne peut pas dépasser.
- response_body_limit = 0 (infini), valeur que le corps de la réponse HTTP ne peut pas dépasser.

De plus, sur une autre machine nous avons installer un serveur web disposant des fichiers à tester par Suricata. Ainsi, avec une simple requête sur le serveur, Suricata reconstitue le fichier puis l'inspecte.

3.2 Résultats et Observations

Résultat de l'analyse de ClamAV :

Nom du fichier	**Taille compressé**	**Résultat**	**Temps d'analyse**	**CPU (%)**	**RAM (%)**
payload.exe	n.a				
fichier_inoffensif	n.a				
250Mo.gz	242.7 Ko				
1Go.gz	970.5 Ko				
250Mo.lz	35.4 Ko				
1Go.lz	141.2 Ko				
250Mo.lzma	35.4 Ko				
1Go.lzma	141.2 Ko				
250Mo.lzo	1.1 Mo				
1Go.lzo	??? Mo				
250Mo.tar.bz2	320 o				
1Go.tar.bz2	828 o				
250Mo.tar.gz	242.7 Ko				
1Go.tar.gz	970.6 Ko				
250Mo.tar.xz	36.6 Ko				
1Go.tar.xz	145.7 Ko				
250Mo.zip	??? Ko				
1Go.zip	??? Ko				
r.zip	??? Ko				

Résultats pour les sites web agrégateurs d'AV :

Nom du fichier	Ratio VirusTotal	Ratio Jotti	Ratio Virscan
payload.exe			
fichier_inoffensif			
250Mo.gz			
1Go.gz			
250Mo.lz			
1Go.lz			
250Mo.lzma			
1Go.lzma			
250Mo.lzo			
1Go.lzo			
250Mo.tar.bz2			
1Go.tar.bz2			
250Mo.tar.gz			
1Go.tar.gz			
250Mo.tar.xz			
1Go.tar.xz			
250Mo.zip			
1Go.zip			
r.zip			

Remarque sur l'antivirus VBA32 :

Résultats sur Mastiff et Viper :

Résultats pour Suricata :

4 Conclusion

References

1. David J. Day et Benjamin M. Burns, "A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines"
2. `oisf.net`, The Open Information Security Foundation, organisation à but non lucrative qui développe et met à jour Suricata.
3. `redmine.openinfosecfoundation.org/projects/suricata/wiki`, la documentation pour utilisateurs et développeurs de Suricata.
4. `blog.inliniac.net/2011/11/29/file-extraction-in-suricata/`
5. `perlgems.blogspot.fr/2012/05/using-virustotal-api-v20.html`
6. Célia Rouis et Mathieu Roudaut, "Contournement d'antivirus par génération d'attaques caméléon", Grenoble INP - Ensimag