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

La sécurité des outils de sécurité

Monnet-Paquet Aurélien
.
Supervised by : Lauradoux Cédric

Juin 2016

Abstract Les logiciels de sécurité sont très important pour les entreprises qui collectent des millions de données personnelles sur ses utilisateurs. Ces logiciels fonctionnent de manière préventive, en temps réel, *offline* ou encore *online*. Il est aussi important que ces outils soient le plus robuste possible. C'est ce que nous allons tester avec les bombes de compression : fichiers compressés qui à la décompression engendrent des fichiers de très grande taille.

Keywords Bombe de compression · IDS · Antivirus · *Framework* d'analyse de malwares · DoS

1 Introduction

Selon l'étude de McAfee/CSIS de 2014, les pertes économiques dues à la cybercriminalité représentent 445 milliards de dollars par an, avec des attaques en forte hausse. C'est pourquoi il existe des logiciels de sécurité pour pouvoir protéger nos données.
Les bombes de compression sont créées avec des fichiers rempli d'un seul caractère mais répété de nombreuses fois. A l'image de 42.zip qui a une taille de 42 Ko compressé et une taille de 4.5 Po une fois décompressé. Si les logiciels de sécurité n'ont pas de protection suffisante pour analyser ce type de fichier, il est possible de provoquer un déni de service ou DoS.
La Section 2 de ce rapport décrit les logiciels de sécurité sur lesquels nous avons testé leur robustesse à analyser des bombes de compression. La section 3 présente les résultats et analyses de nos tests.

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

2 Outils de sécurité

2.1 Antivirus

Les antivirus (AV) sont des programmes très importants en sécurité. Ce sont des programmes installés sur les machines des utilisateurs et ils permettent de détecter les malwares et les empêchent de s'exécuter sur le système. Il existe différentes manières de détecter un malware :

- scan par signature : on calcule la signature d'un fichier ou d'un morceau de code et on le compare à une base de données. C'est la méthode la plus populaire chez les concepteurs d'antivirus. Cette méthode est cependant inefficace contre les malwares polymorphes ou capable de changer leur signature.
- Analyse heuristique : la plus puissante des méthodes 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 s'il s'agit d'un malware ou non. Cette méthode peut provoquer des fausses alertes, coûte du temps CPU et ne garantie pas toujours un résultat.
- Contrôle d'intégrité : permet de vérifier qu'un fichier sensible n'a pas été modifié au cours du temps. Les informations comme : les droits (d'exécuter, de modifier ...), les propriétaires, sont comparées à des références lors de la demande d'ouverture du fichier (si analyse en temps réel) ou lors d'un scan de l'AV. Si une anomalie apparaît, l'AV alerte l'utilisateur.

Le but de notre expérience est de tester s'il est possible de faire un déni de service lorsque l'AV analyse une bombe de compression.
Nous avons testé 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. Cet outil ne doit pas remplacer un AV avec une analyse en temps réel installé sur une machine. Il existe deux types d'agrégateurs:

2.2.1 Sites web

Ces sites web spécialisés permettent d'analyser les fichiers grâce à des dizaines d'antivirus différents. Un partenariat relie les éditeurs d'AV et le site. Les éditeurs d'AV mettent à disposition des sites leur programme. Cela signifie que les éditeurs sont responsables de leur configuration. Il peut arriver qu'un AV soit configuré différemment sur un site par rapport à la version commerciale. En effet, il s'agit d'un bon moyen pour les éditeurs de tester de nouvelles fonctionnalités, configurations et d'inciter les utilisateurs à utiliser leur version commerciale. L'utilisation de plusieurs antivirus permet de combiner plusieurs algorithmes de détection et plusieurs bases de données. Lorsqu'un fichier est soumis par un utilisateur sur cette plate-forme, il est analysé par les moteurs des différents AV. Dans le cas où au

moins un AV détecte un fichier comme malveillant, l'utilisateur est averti sur le caractère malicieux du fichier.

Site web	**VirusTotal**	**Jotti**	**Virscan**
Nombre d'antivirus	57	19	39
Localisation	Google, Espagne	Pays-bas	Chine
Type d'hébergement	Cloud	Cloud	Cloud

Table 1 Présentation des 3 sites web agrégateurs d'AV que nous avons choisis pour réaliser nos test.

Remarque : VirusTotal est le plus populaire avec en moyenne 1.5M de soumissions par jours (dont 16.6% provenant de France).

Pour utiliser ce service, un utilisateur doit envoyer le fichier suspect vers le site pour que les moteurs des différents AV puissent l'analyser. 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 [5] existant pour effectuer nos tests. Écrit en Perl, ce script 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 des antivirus partenaires de VirusTotal.

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

Après analyse, le fichier n'est pas détruit s'il est considéré comme dangereux, 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 il n'a pas de contrôle et pour lequel il ne maîtrise pas la juridiction. C'est pourquoi il existe aussi des « Framework de détection de malwares ».

2.2.2 Frameworks

Ce type d'outils fonctionne 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 utilisé deux *frameworks* :

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

Ces deux outils sont pré-installés dans la distribution REMnux (`remnux.org`) qu'il est facile d'importer dans Virtualbox (`www.virtualbox.org`).
Mastiff et Viper permettent entre autre:

- d'identifier et classer le type des fichiers analysés grâce aux règles de YARA : un outil qui identifie et classe les malwares grâce a des règles, (`plusvic.github.io/yara`).
- De reconnaître deux fichiers dont le contenu est très similaire grâce au fuzzy hashing (ssdeep) (`ssdeep.sourceforge.net`). Technique permettant de calculer une signature d'un fichier sur des probabilités et sur le filtre de Bloom [7]. Ainsi, il est plus facile de mettre en relation deux fichiers avec un contenu proche mais pas identique. Le fonctionnement détaillé de cette méthode est décrite en [8]. Technique pratique pour repérer les malwares polymorphes.
- D'extraire et analyser récursivement des fichiers zip. C'est notamment cette dernière qui nous intéresse et que nous allons tester.

Mastiff est un programme dont le processus peut être exécuté dans un docker[1]. Dans ce cas, un dossier est partagé 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 ajouté l'option d'envoyer les fichiers à analyser vers VirusTotal automatiquement. Lors de son analyse, Mastiff génère un fichier de résultats 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 ces commandes se retrouvent sur cette interface web.

2.3 IDS

Les systèmes de détection d'intrusion (IDS) 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 que l'administrateur a définies, 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 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.
L'analyse s'effectue aussi sur le contenu des paquets qui circulent sur le réseau. De cette manière une règle peut être créée pour générer une alerte lorsqu'un paquet contient les caractères "Euro 2016" par exemple.
L'IDS que nous avons choisi de tester est Suricata [2]. Suricata se démarque de la

[1] Un docker permet d'isoler l'exécution d'un processus dans un container virtuel. Plus léger qu'une machine virtuelle (VM) car c'est l'OS de la machine hôte qui est sollicité.

concurrence (snort en particulier) par une analyse multithread. C'est une caractéristique importante. L'article de David J. Day et Benjamin M. Burns [1] analyse les performances de Suricata. Suricata supporte nativement 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 a pas de déni de service possible lorsque Suricata analyse une bombe de compression.
Après avoir installé Suricata, nous avons rédigé des règles pour que les fichiers soient 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 réassemblé 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 installé 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 Expérimentations

3.1 Méthodologie

Dans un premier temps, nous avons généré un exécutable Windows (payload via metasploit) connu pour être analysé comme un malware par la plupart des outils disponibles sur le marché. Ensuite, pour chacun des outils, nous avons effectué une analyse témoin. Cette analyse se compose du fichier généré précédemment, et d'un fichier contenant uniquement des zéros et qui est donc complètement inoffensif.
Dans un second temps, nous avons analysé les résultats des deux premiers fichiers compressés dans différents formats : .gz, .lzma, .tar.bz2, .tar.gz, .tar.xz, .lz, .lzo, .zip pour tenter de leurrer les AV. Il existe d'autres méthodes de contournement comme celles décrites dans [6].
Enfin, nous avons utilisé des bombes de compression dans le but de provoquer un déni de service lors de l'exécution des AV.

Nous avons réalisé les tests (et installer les outils nécessaires) sur une machine ayant un OS Linux 3.13.0-32-generic (Ubuntu 12.04 LTS).
Pour l'utilisation de ClamAV deux paramètres sont à prendre en compte pour mener à bien cette expérience :

- "max-recursion= n" : la profondeur maximale d'exploration lors d'une décompression, 16 par défaut.
- "max-filesize= n" : Extrait et analyse n octets de chaque archives. 25 Mo par défaut avec une limite de 4 Go.

Comodo ne nous permet pas de modifier des paramètres comme la récursion, ou la taille des fichiers analysés.

Logiciel	Catégorie	Version
ClamAV	Antivirus	0.98.7
Comodo	Antivirus	1.1.268025.1
Mastiff	Framework	0.7.1
Viper	Framework	0.12.9
Suricata	IDS	3.0

Table 2 Présentation des logiciels testés.

3.2 Résultats et observations

Nom du fichier	ClamAV		Comodo	
	Résultats	Temps	Résultats	Temps
payload.exe	✓	9.2s	✓	4s
fichier_inoffensif	X	8.9s	X	4s
250Mo.gz	X	9.6s	X	8s
1Go.gz	X	10s	X	20s
250Mo.lz	X	8.7s	X	4s
1Go.lz	X	8.2s	X	4s
250Mo.lzma	X	8.3s	X	4s
1Go.lzma	X	8.7s	X	4s
250Mo.lzo	X	8.3s	X	4s
1Go.lzo	X	8.6s	X	4s
250Mo.tar.bz2	X	11.9s	X	14s
1Go.tar.bz2	X	12.5s	X	54s
250Mo.tar.gz	X	9.7s	X	11s
1Go.tar.gz	X	9.8s	X	52s
250Mo.tar.xz	X	10s	X	4s
1Go.tar.xz	X[1]	10s	X	4s
250Mo.zip	X	12s	X	8s
1Go.zip	X	25.7s	X	21s
r.zip	X[2]	8.4s	X	4s
42.tar.bz2	X	10.9s	X	6s[3]

Table 3 Récapitulatif des analyses de **ClamAV** et **Comodo**

[1] : Warning constaté lors de la décompression pour 1Go.tar.xz, "decompress file size exceeds limits - only scanning 104857600 bytes".
[2] : En forçant environ 400 récursions (au lieu de 16 par défaut), il y a une erreur de segmentation.
[3] : Limite de récursion atteinte mais non connue. Utilisation du CPU très importante lors de l'analyse d'une sous partie de 42.tar.bz2.

Observations : L'analyse témoin se déroule normalement, c'est-à-dire que le payload est détecté par les deux AV et que le fichier inoffensif n'est pas détecté.
On remarque que les deux AV ne détectent aucune des bombes de compression. De plus, certain format de compression ne semblent pas être détectés puisque le temps d'analyse des deux AV est assez bas par rapport à des formats reconnus.

Nom du fichier	**Ratio de détection**		
	VirusTotal	**Jotti**	**Virscan**
payload.exe	42/56	17/19	20/39
fichier_inoffensif	0/56	0/19	0/39
250Mo.gz	0/56	1/19	2/39
1Go.gz	1/56	3/19	3/39
250Mo.lz	0/57	0/19	0/39
1Go.lz	0/56	0/19	0/39
250Mo.lzma	0/57	0/19	0/39
1Go.lzma	0/57	0/19	0/39
250Mo.lzo	0/56	0/19	0/39
1Go.lzo	0/56	0/19	0/39
250Mo.tar.bz2	0/56	4/18	1/39
1Go.tar.bz2	1/57	5/19	2/39
250Mo.tar.gz	0/57	2/19	1/39
1Go.tar.gz	1/57	3/16	2/39
250Mo.tar.xz	0/57	2/20	0/39
1Go.tar.xz	0/57	2/20	0/39
250Mo.zip	1/56	2/20	0/39
1Go.zip	2/55	3/20	0/39
r.zip	0/55	1/18	0/39
42.tar.bz2	1/56	4/15	0/39

Table 4 Résultats pour les sites web agrégateurs d'AV

On remarque que les analyses témoins se déroulent correctement. A savoir, que notre payload est détecté par une grande majorité d'AV et que notre fichier inoffensif n'est pas détecté.
Il existe des formats de compression qui ne sont pas reconnus par les AV puisque aucun d'entre eux ne nous alerte sur le coté malveillant des bombes.
Le format le plus détecté est : tar.bz2.
La variation du nombre total d'AV pour chaque site est dû à l'ajout ou à la suppression d'AV partenaire. Durant nos tests, VirusTotal comprenait entre 56 et 57 AV. Si cette valeur descend en dessous de 56, c'est qu'un ou plusieurs AV n'ont pas répondu dans les temps. Le nombre normal d'AV pour Jotti est compris entre 19 et 20.
Il existe une limite de taille pour la soumission des fichiers sur chaque site. Comme nos bombes de compression dépassent largement cette limite (une fois décompressées), il existe une protection qui consiste à limiter le temps d'analyse alloué

aux AV. Ainsi, pour Virscan si un AV n'a pas trouvé de malware en moins de 60 secondes alors le résultat est : "Rien n'a été trouvé".

Nom fichier	VirusTotal	Jotti	Virscan
250Mo.gz		avast	Fprot, Panda
1Go.gz	VBA32	Arcabit, avast, VBA32	Fprot, Panda, VBA32
250Mo.tar.bz2		AVG, Arcabit, avast, Sophos	Fprot
1Go.tar.bz2	VBA32	AVG, Arcabit, avast, sophos, VBA32	VBA32, Fprot
250Mo.tar.gz		Arcabit, avast	Fprot
1Go.tar.gz	VBA32	Aracabit, avast, VBA32	Fprot, VBA32
250Mo.tar.xz		Arcabit, AVG	
1Go.tar.xz		Arcabit, AVG	
250Mo.zip	VBA32	avast, VBA32	
1Go.zip	Baidu, VBA32	Arcabit, avast, VBA32	
r.zip		Sophos	
42.tar.bz2	Zillya	Arcabit, avast, AVG, Sophos	

Table 5 Antivirus ayant détectés un malware.

Nous pouvons remarquer que les résultats des antivirus sont cohérents sur un seul et même site mais ne semblent pas être configurés de la même manière entre les différentes plates-formes. Cependant, VBA32 semble avoir un comportement cohérent sur tous les sites sur lequel il est testé. En effet, rien de suspect pour VBA32 sur les fichiers de 250Mo mais il y a une détection sur les bombes de 500Mo. Après analyse approfondie : la limite de détection est de 315Mo.
ClamAV et Comodo analysent les fichiers sur Virscan et on remarque qu'il n'y a pas de différence entre la version installée et la version online : rien n'est détecté.

Antivirus sans réponse	
VirusTotal	**Jotti**
TotalDefense	ESET
Zoner	F-Secure
	Trend Micro
	GData
	Sophos

Table 6 Récapitulatif des AV ayant un temps de réponse dépassé.

Il doit être possible de provoquer un deni de service, dans certaines conditions sur ces AV.

Nom fichier	**Ratio de détection**			**Résultats**	
	VirusTotal	Jotti	Virscan	ClamAV	Comodo
analyse témoin	42/56	17/19	20/39	✓	✓
payload.gz	38/56	18/20	17/39	✓	✓
payload.lz	2/56	1/19	1/39	X	X
payload.lzma	7/56	4/19	4/39	X	X
payload.lzo	4/56	3/19	5/39	✓	X
payload.tar.bz2	32/56	17/20	18/39	✓	✓
payload.tar.gz	33/56	17/20	16/39	✓	✓
payload.tar.xz	18/56	13/20	5/39	✓	X
payload.zip	40/56	18/20	12/39	✓	✓

Le fait de compresser le payload dans un format très répandu comme (.zip) leurre qu'une petite minorité d'AV. En revanche, compresser le payload dans un format qui n'est pas très répandu (.lz) permet de leurrer jusqu'à 95% des antivirus.

Résultats pour Mastiff et Viper :
On a pu remarquer que Mastiff analyse les fichiers compressés récursivement. Dans le cas de r.zip : un fichier qui se décompresse en un autre fichier compressé indéfiniment, Mastiff analyse ce nouveau fichier à chaque fois et cela sans condition d'arrêt. Ce comportement peut occasionner l'envoie de requêtes de manière abondante vers VirusTotal lorsqu'on le spécifie dans le fichier de configuration de Mastiff.

Résultats pour Suricata :
Lors du téléchargement des bombes de compression, on observe que les règles créer se déclenchent bien et ainsi remontent des alertes dans le fichiers de logs. Cependant, on ne constate pas de déni de service.

4 Conclusion

Nous avons compressé un fichier malveillant dans différents formats. Et il s'avère que la plupart des antivirus du marché ne reconnaissent pas tout les formats de compression. Ainsi, le simple fait de compresser un malware dans le format .lz permet de leurrer jusqu'à 95% des AV qui avaient détecté ce même malware sans compression.
Ensuite, l'analyse des bombes de compression par les logiciels de sécurité, permet de mettre en évidence le fait qu'il existe des protections contre les DoS. Mais qu'elles ne sont pas tout le temps suffisantes. En effet, certain AV des sites web n'ont pas répondu dans les temps lors de certaines analyses. Une erreur de segmentation est survenue en poussant jusqu'à 400 la profondeur maximale dans ClamAV. Mastiff ne s'arrête jamais lors de l'analyse de r.zip. Enfin, les antivirus comme ClamAV et Comodo ne détectent jamais les bombes de compression.
Qu'en est il pour un éventail plus large d'antivirus installés sur un environnement Windows ?

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
7. `fr.wikipedia.org/wiki/Filtre_de_Bloom`, page wikipédia du Filtre de Bloom.
8. Jonathan Tournier, "Evaluating Forensics Tools", Juin 2015.