Empirical Study to Fingerprint Public Malware Analysis Services

Álvaro Botas[1], Ricardo J. Rodríguez[2], Vicente Matellán[1], and Juan F. García[1]

[1] Instituto de Ciencias Aplicadas a la Ciberseguridad, Universidad de León, Spain
[2] Centro Universitario de la Defensa, Academia General Militar, Spain

Abstract. The evolution of malicious software (*malware*) analysis tools provided controlled, isolated, and virtual environments to analyze malware samples. Several services are found on the Internet that provide to users automatic system to analyze malware samples, as VirusTotal, Jotti, or ClamAV, to name a few. Unfortunately, malware is currently incorporating techniques to recognize execution onto a virtual or sandbox environment. When analysis environment is detected, malware behave as a benign application or even show no activity. In this work, we present an empirical study and characterization of automatic public malware analysis services. In particular, we consider 26 different services. We also show a set of features that allow to easily fingerprint these services as analysis environments. Finally, we propose a method to mitigate fingerprinting.

1 Introduction

In the last few years, there has been a steady growth in the number and complexity of applications with non-legitimate intentions (known as malicious software, or *malware*). Similarly, Symantec reports an increase from 252 million of malware variants in 2013 to 317 million in 2014 [14].

To detect the malicious behavior of a binary file, an binary analysis is performed, mainly considering its interaction with the OS and the network. The huge number of malware samples, however, makes manual analysis unfeasible. Thus, several methods were developed to automatize analysis using the principle of isolation, in particular, using virtual environment and sandboxes. An automatic malware analysis service allow any user to upload files to be analyzed using different means, such as web forms, API REST, or others. Once submitted, the file is analyzed and a report is given indicating whether it is malicious.

Cybercriminals enhance their malware by adding new malicious functions to detect when they are being analyzed (and then behave beningly), since the longer the malware stays as *no malicious file*, the more likely the cybercriminals earn more income. Malware with capabilities of analysis detection are named as *analysis-aware* or *split personality malware* [1, 8]. These detection mechanisms are mainly based either on indirect techniques (i.e., time or event triggered) or on direct techniques as binary analysis, sandboxing, or virtual and memory analysis [12].

In this work, we present an empirical study and characterization of 26 different automatic public malware analysis services (PMAS)[3]. We also analyze their configuration individually to observe recurrent patterns that might be used by analysis-aware malware to succeed on analysis recognition. Finally, we propose a method to improve the degree of concealment of a PMAS.

This paper is organized as follows. Section 2 reviews related work. Section 3 introduces our characterization of PMAS. Then, in Section 4 we describe our methodology to obtain data from a PMAS server and discuss our preliminary results. Our proposal to improve the degree of concealment of PMAS is introduced in Section 5. Section 6 concludes the paper and states future work.

2 Related Work

There are several works in the literature regarding the presence (or absence) of an analysis system. Numerous techniques for virtual machines and emulators detection were provided in [6,11]. They qualitatively compared those techniques, although no quantitative assessment was provided. Recently, methods to fingerprint anti-virus emulators were introduced in [2].

In [3], static analysis was used to detect anti-analysis techniques. Similarly, dynamic analysis was used in [7] to detect 78 anti-analysis signatures based on sequences of system calls over 2810 malware samples.

Detection of anti-analysis malware was performed in [4] comparing the binary execution between real and virtual environments. However, no insights about analysis-aware techniques used by the malware were provided. Different methods to detect Cuckoo Sandbox, an open-source tool to create a sandbox environment for statically and dynamically analyzing binaries, were provided (along with countermeasures) in [5].

A taxonomy and non-exhaustive survey on techniques used for detecting analysis environments were introduced in [12]. Besides, a tool was provided to trick an anti-analysis malware sample running on a virtual environment as running on a real environment. A recent study on detection of anti-analysis malware has been recently provided in [15], in which anti-analysis signals are divided into weak signals, strong signals, and composite signals, depending on those signals are normally used by a benign software or by a malware.

Other techniques for detecting execution on hypervisors using time information were provided in [10, 16]. Detection based on unexpected semantics when executing certain CPU instructions was also proposed in [13].

Recently, the FFRI malware dataset was used in [9] to evaluate the most used analysis detection methods. Their approach was based on analyzing Windows APIs calls and results were used to improve Cuckoo Sandbox system.

Regarding PMAS fingerprinting, it is worth mentioning [17]. They evaluated 15 PMAS, being able to fingerprint them using as feature the IP source address of network responses. Our work is similar to theirs, but we perform a more exhaustive analysis of features to fingerprint PMAS.

[3] In this paper, we indistinguishably use PMAS as singular and plural acronym.

3 Characterization of Public Malware Analysis Services

PMAS allow users to send a file, analyze it, and report whether it is classified as malware or as benign software. In this paper, we have finally evaluated 26 PMAS. To characterize each of these services, we consider the following features:

– **How the suspicious files are sent to the service**. Here, we consider methods as external application, web form, email, or others.
– **Where the analysis is done**. The service may perform the analysis on its own infrastructure, or otherwise, use other infrastructures (metaservice). A well-known example of metaservice is VirusTotal, which checks the submitted file with a bunch of anti-virus solutions.
– **How the analysis report is given to the user**. The report can be given to the user through a report file, a web page, or by email. The report may have, besides, different information granularity. For instance, some services just report whether the suspicious file is malware, while others provide a detailed report regarding malware behavior in the system (mainly, interaction of the file with the OS – files, processes, and Windows registry – and with the Internet). Some services only reports about new files created, while others also provide the content of new files.
– **Price of the service**. We distinguish between free, paid, or freemium (basic options for free plus advanced options by subscription) services.
– **Accepted file types**. Each service may accept different file types and from different OSes. Some services also allow to upload URLs for scanning.

Table 1 summarizes the characterization of the different services that we consider in the paper, regarding the features aforementioned ("N/A" means not available, i.e., we do not have enough knowledge to state a conclusion). We initially considered 100 PMAS, although they were reduced to 26 by different reasons[4]. For instance, some of the services are only paid services. Then, we only considered free services or freemium. Furthermore, some well-know PMAS as Anubis have been discontinued and evolved to systems aimed at providing professional services to large companies or institutions.

4 Experiments and Discussion

4.1 Experiment Description

For experimentation, we developed a Windows executable file. Thus, from the initial 100 PMAS we discarded also the ones that focus on different filetypes. We aim at testing these services as future work.

The Windows executable file used as probe was specifically crafted for obtaining information about the PMAS upon execution. During execution, network packages are sent to a web server under our control to collect the data. The worfkflow diagram to collect data is shown in Fig. 1. These steps are as follows:

[4] Microsoft is composed by Microsoft Other, Microsoft Defender 10, Microsoft Defender 8, Microsoft Defender 7, Microsoft Vista, XP, Essentials and Others

Service	**App send form**	**metaservice**	**Report**	**Report Data Specified**	**Price**	**Document**		
						Accepted Filetype	**Url's**	**Operating System Platform**
360TotalSecurity	Online form, Mail	Yes	Email	N/A	Free	EXE N/A	No	N/A
Jotti	Online Form	Yes	WebPage	Hash Information	Free	EXE, DMG JPG, PDF, APK, . . .	No	Desktop OS Win, MacOSX, Linux Mobile OS
UploadMalware	Online Form	Yes	Webpage	Hash Information	Free	JPG, PDF, APK, . . .	No	Win, MacOSX, Linux Mobile OS
Virustotal	Online Form Mail Public API External App	Yes	Webpage	Depends of the file (platform)	Free	Executables images,doc,flash Apk, ipa office	YesYes	Desktop OS Win, MacOSX, Linux
BleepingComputer	Online Form	N/A	No	No	Free	N/A	No	N/A
Roboscan	External app	N/A	Email	Malware or not	Free	Several	No	N/A
Fortinet	Online Form	N/A	Email	N/A	Free	PEx	No	Windows
Microsoft	Online Form	N/A	Webpage	Malware or not	Free	PEs	No	Windows
Bitdefender	Online Form	N/A	Email	N/A	Free	PDF, flash, office PEs, DMG APK, Flash, OFFICE	Yes	Multiplatform
Sophos	Online Form	N/A	Email	N/A	Free	Several	Yes	Multiplatform
F-Secure	Online Form	No	Email	Malware or not	Free	PEs, DMG	Yes	Desktop OS
Avira	Online Form	No	Email Webpage	Malware or not	Free	EXE, DMG	Yes	Desktop OS
PayloadSecurity	Online Form	No	Webpage	Created files paths Keychan, File Details Network details	Freemium	PE, Office, PDF APK files and more (e.g. EML)	Yes	Windows Android
McAfee	External app Email	No	N/A	N/A	Free	PEs	No	Windows
Malwr	Online Form	No	Webpage	Created files paths Registry Network activity	Free	PEs PDF, Office, ASCII Flash, Java	No	Windows
ClamAv	Online Form	No	Email	N/A	Free	Several	No	N/A
Valkyrie	Online Form	No	Webpage	Created files paths Registry PE Imports	Freemium	PEs	Yes	Windows
Zoner	Online Form	No	Email	N/A	Free	N/A	No	Multiplatform
ThreatAnalyzer	Online Form	No	WebPage File	Several	Paid	Several	No	Windows (allow you to choose versions)
SonicWall	Online Form	No	N/A	N/A	Free	N/A	No	N/A
AVG	Online Form	No	N/A	N/A	Free	Several	No	Multiplatform
Symantec	Online Form External app	No	N/A	N/A	Free	Several	Yes	Multiplatform
Nanoav	Online Form	No	Email	Malware or not	Free	N/A	No	N/A

Table 1. Characterization of Public Malware Analysis Systems.

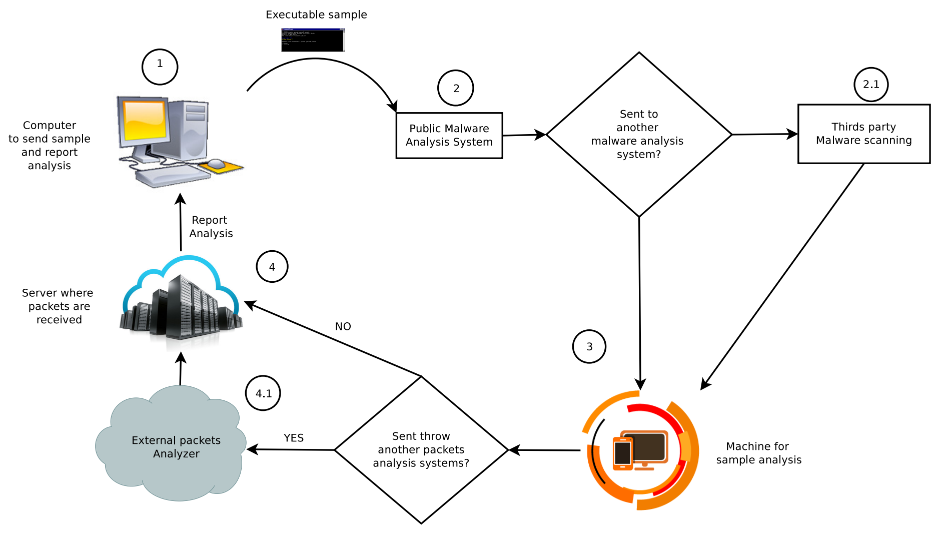

Executable sample
1
Computer to send sample and report analysis
2
Public Malware Analysis System
Sent to another malware analysis system?
2.1
Thirds party Malware scanning
Report Analysis
4
Server where packets are received
NO
3
4.1
External packets Analyzer
YES
Sent throw another packets analysis systems?
Machine for sample analysis

Fig. 1. Workflow diagram to collect PMAS information.

1. Our probe is specifically crafted for a given PMAS. Then, the probe is sent to that PMAS.
2. The probe can be sent by the PMAS to another external service by the analysis service, or otherwise, the PMAS analyze the probe itself.
3. The sample is analyzed in a PMAS (or in an external service). The sample sends network packages to our server with the information obtained from the PMAS that analyzed it.
4. Network packages are sent from the PMAS to our server. Thus, we obtain from what source IP address packages are coming.
5. Requests are are collected in our server. Finally, data is parsed and analyzed.

Although we initially sent our probe to 70 services, we only received responses from 26 services. Response dataset date from April 2016 to January 2017. Surprisingly, we observed that multiple responses were obtained from certain services. This indicates that some services executed the probe multiple times. Fig. 2 shows the response ratio of received over sent probes. Although it is unknown in some cases whether the PMAS is a metaservice, we assumed that it might be a metaservice when the ratio is greater than 5.

PMAS that use external engines to analyze malware (metaservices) are shown in left-hand side of Fig. 2 (red bars). Services which do not indicate clearly whether they sent file to external engines are shown in yellow bars. Finally, services using own engines are shown in right-hand side (blue bars).

4.2 Machine Characterization

We characterized each PMAS considering both physical level (hardware) and logical level (software). Specific features that we collected are depicted in Fig. 3.

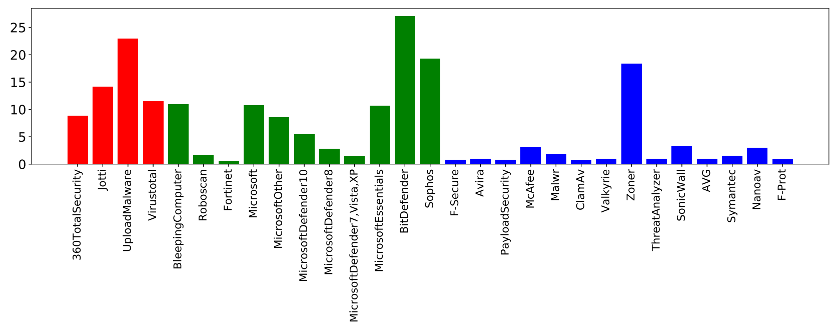


Fig. 2. Ratio of samples received for each probe sent.

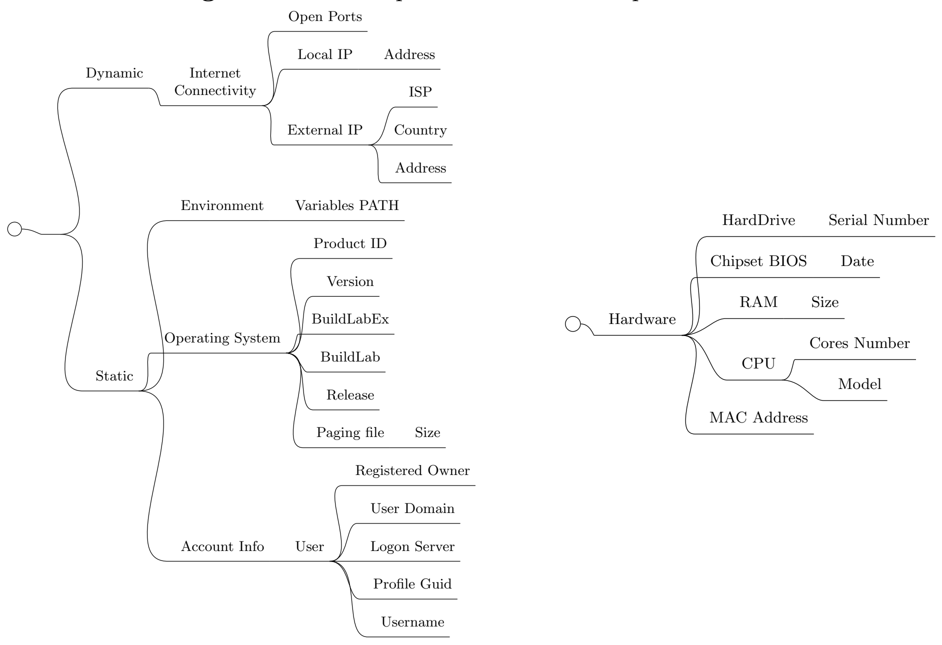
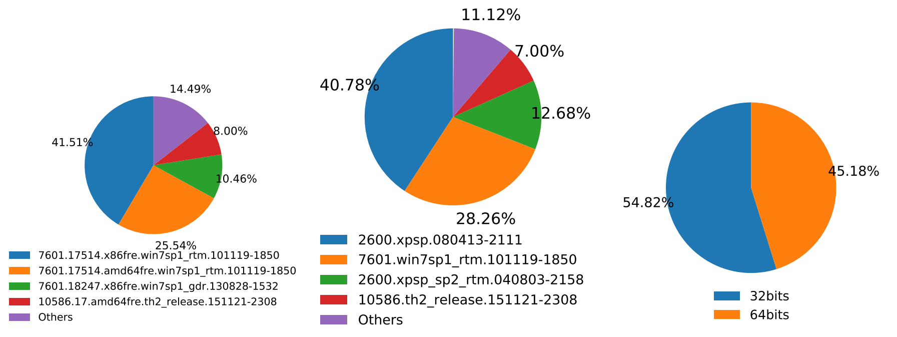


Fig. 3. Software (left-hand) and hardware (right-hand) characterization features.



(a) *BuildLabEx* (b) *Build Lab* (c) OS arquitecture width

Fig. 4. Characterization of (a, b) Windows versions and (c) OS architecture width.

Regarding software, we distinguished between *static data* (i.e., data that are not modified at runtime) and *dynamic data*. As dynamic data, we distinguish elements belonging to *Internet connectivity*, such as*open ports*, *Local IP address*, and *external IP address*. Data of interest as *ISP* or *country* can be extracted from these elements. As static part, we consider different elements: *environment variable paths*, *paging file*, and *OS-related data* (such as *Product ID*, i.e., the Windows product identifier; *Build Lab*, i.e., the Windows version; or *Build Lab Ex*, i.e., extended information of Windows version). Regarding the OS's user account, we collected the *registered owner*, *user domain*, *logon server* (that is, network machine name), *Profile Guid*, and *user name*.

Regarding hardware, we consider *hard drive serial number*, *BIOS date*, *RAM size*, *CPU processors number*, *processor model*, and *MAC address* of the machine.

4.3 Discussion

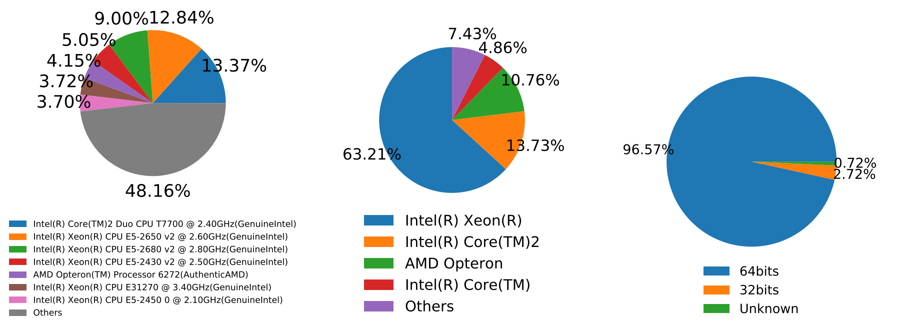
As previously described, we only obtained responses from 26 different PMAS. In particular, the number of responses were, in total, 7680 over 1500 sent. Note that the number of received responses is sensitively greater than those sent. This is mainly motivated because some PMAS analyzed the probes several times, as well as metaservices received a report from each external service used.

Some responses were discarded, since they contain no value regarding features of interest (see Section 4.2). We believe that this is motivated by several reasons: (i) the response was unable to reach our server; (ii) the execution of the file was stopped prior a normal finishing its activity (i.e., a timeout or an exception occurs); or (iii) the PMAS does not allow to access to that values (or are unavailable for that particular OS version of the PMAS). As last step, we normalized responses to have an equal response distribution among PMAS.

For the sake of space, in the following we briefly discuss some of the characterization features under study.

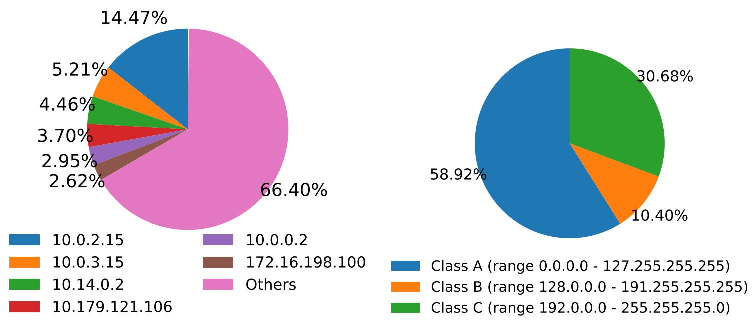
Fig 4 shows results of OS-related data. Although 20.22% of responses were obtained (*BuildLabEx*), we observe a prevalence of Windows 7 as OS. Let us remark here that for certain OSes, as Windows XP, this value cannot be obtained. In this regard, the field of *BuildLab* becomes more useful (46.22% of responses obtained). Surprisingly, Windows XP is found in different flavors (with SP1 or SP2 installed) in little more than half of responses, and Windows 7 SP1 is more than 25%. Data coming from *BuildLabEx* was used to infer the OS architecture width. As shown, 32-bit and 64-bit are almost equally distributed.

Fig.5 plots the characterization of CPUs used by PMAS (we just received a 16.65% of responses), considering CPU model, family, and architecture width. Results show a prevalence of QEMU-based virtualization systems. For instance, *Intel Core i7 9xx (Nehalem Class Core i7) (GenuineIntel)* is one of the CPU models that can be chosen when configuring a QEMU virtual machine. Regarding CPU family, we observed a high ratio of Intel Xeon technology. Regarding CPU architecture width, more that 95% are 64-bit CPUs, although the OS architecture was just a half when considering data coming from *software*.



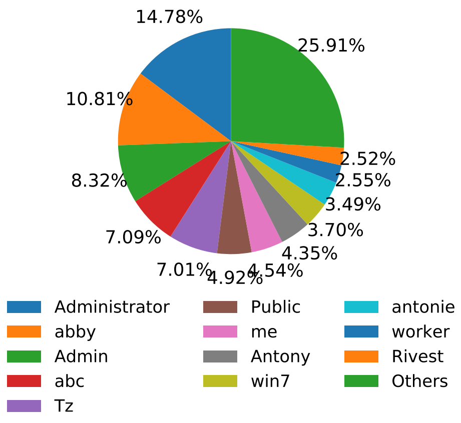
(a) CPU model (b) CPU family (c) CPU arquitecture width

Fig. 5. Characterization of CPUs of PMAS.



(a) Local IP address (b) IP address classes

Fig. 6. Characterization of IP addresses of PMAS.



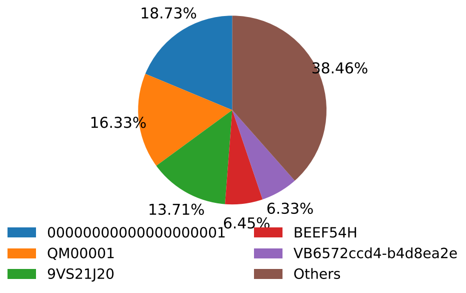


Fig. 8. Distribution of *HardDrive serial number*.

Fig. 7. Distribution of *Username*.

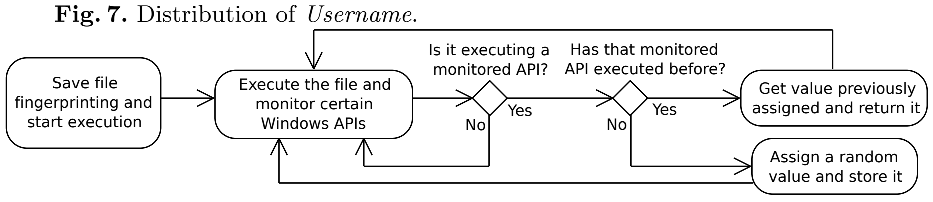

Save file fingerprinting and start execution
Execute the file and monitor certain Windows APIs
Is it executing a monitored API?
Yes
No
Has that monitored API executed before?
Yes
No
Get value previously assigned and return it
Assign a random value and store it

Fig. 9. Theoretical model to evade PMAS fingerprinting.

Regarding *Internet connectivity*, we had 86.49% data responses. Results of characterization of local IP addresses are shown in Fig. 6. Note that we obtained a huge diversity in results, without a clear prevalence of any value although the three most common IP addresses represent more than a 25%. If we consider IP address classes, class A is the most common IP address class. This IP address class is normally used in business and non-domestic environments.

Regarding *username*, we obtained 49.33% valid responses. Fig. 7 shows the distribution of these values. Note that *Administrator*, *abby*, and *Admin* represents more than 50% of collected usernames.

Regarding *HardDrive serial number*, we obtained 57.06% valid responses. Results are plotted in Fig. 8. Note that *00000000000000000001*, *QM00001*, and *9VS21J20* represent almost a 50% of collected values.

5 Evasion of PMAS Fingerprinting

The aforementioned features have frequently repeated values and hence, they allow to easily fingerprint a PMAS. Here, we propose a theoretical model (see Fig. 9) to evade PMAS fingerprinting when malware is dynamically analyzed. To collect PMAS features, malware needs to execute certain Windows APIs. Thus, we propose to monitor those APIs and return random values. To avoid detection by checking result of consecutive monitored API calls, we propose to fingerprint each file (e.g., using its MD5 hash), keep track of results given to a specific file, and then return always the same results. This theoretical model may prevent a malware to fingerprint and recognize a PMAS, regardless the features checked.

6 Conclusions and Future Work

Software with malicious intentions (malware) have increased in number and complexity during last decade. To detect it, automatic public malware analysis services (PMAS) are used. These services allow users to upload files and report them about its malicious behavior. Thus, malware started to introduce techniques to detect these analysis environments and behave as benign software.

In this paper, we presented an empirical study of PMAS. In particular, we have characterized 26 different PMAS regarding different features (how the files are sent to the service, where the analysis is done, how the report is given, price of the service, and accepted file types). Then, we have developed an application to fingerprint hardware and software features of those PMAS. Our results show that few features (such as OS, CPU, or username) are enough to fingerprint some of those PMAS. Finally, we proposed a (theoretical) execution model to evade PMAS fingerprintg.

As future work, we aim at identifying the relationships between PMAS and fingerprinting each PMAS in detail. Furthermore, we aim at implementing and evaluating our evasion model.

Acknowledgments: The research of A. Botas, V. Matellán, and J. F. García was supported by INCIBE according to the rule 19 of the Digital Confidence Plan and by the University of León under contract X43. The research of R. J. Rodríguez was supported in part by the Spanish MINECO project CyCriSec (TIN2014-58457-R), by University of Zaragoza and Centro Universitario de la Defensa project UZCUD2016-TEC-06, and by "Ayudas para estancias de Investigadores visitantes en el CEI Triangular-E3" (hosted by University of León).

References

1. Balzarotti, D., Cova, M., Karlberger, C., Kirda, E., Kruegel, C., Vigna, G.: Efficient Detection of Split Personalities in Malware. In: NDSS 2010 (2010)
2. Blackthorne, J., Bulazel, A., Fasano, A., Biernat, P., Yener, B.: AVLeak: Fingerprinting Antivirus Emulators through Black-Box Testing. In: 10th USENIX Workshop on Offensive Technologies. USENIX Association (2016)
3. Chen, P., Huygens, C., Desmet, L., Joosen, W.: Advanced or Not? A Comparative Study of the Use of Anti-debugging and Anti-VM Techniques in Generic and Targeted Malware. In: IFIP SEC 2016. pp. 323–336 (2016)
4. Chen, X., Andersen, J., Mao, Z.M., Bailey, M., Nazario, J.: Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In: DSN 2008. pp. 177–186 (June 2008)
5. Ferrand, O.: How to detect the Cuckoo Sandbox and to Strengthen it? Journal of Computer Virology and Hacking Techniques 11(1), 51–58 (2015)
6. Garfinkel, T., Adams, K., Warfield, A., Franklin, J.: Compatibility is Not Transparency: VMM Detection Myths and Realities. In: 11th USENIX Workshop on Hot Topics in Operating Systems. pp. 6:1–6:6. USENIX Association (2007)
7. Kirat, D., Vigna, G.: MalGene: Automatic Extraction of Malware Analysis Evasion Signature. In: CCS 2015. pp. 769–780. ACM (2015)
8. Kumar, A.V., Vishnani, K., Kumar, K.V.: Split Personality Malware Detection and Defeating in Popular Virtual Machines. In: SIN '12. pp. 20–26. ACM (2012)
9. Oyama, Y.: Trends of anti-analysis operations of malwares observed in API call logs. Journal of Computer Virology and Hacking Techniques pp. 1–17 (2017)
10. Pék, G., Bencsáth, B., Buttyán, L.: nEther: In-guest Detection of Out-of-the-guest Malware Analyzers. In: EUROSEC'11. pp. 3:1–3:6. ACM (2011)
11. Raffetseder, T., Krügel, C., Kirda, E.: Detecting System Emulators. In: 10th International Conference on Information Security. pp. 1–18 (2007)
12. Rodríguez, R.J., Rodríguez-Gastón, I., Alonso, J.: Towards the Detection of Isolation-Aware Malware. IEEE Latin America Transac 14(2), 1024–1036 (2016)
13. Shi, H., Alwabel, A., Mirkovic, J.: Cardinal Pill Testing of System Virtual Machines. In: 23rd USENIX Security Symposium. pp. 271–285 (2014)
14. Symantec: ISTR - Internet Security Threat Report. Tech. rep. (2016)
15. Tan, J.W.J., Yap, R.H.C.: Detecting Malware Through Anti-analysis Signals - A Preliminary Study. In: CANS 2016. pp. 542–551. Springer (2016)
16. Wang, G., Estrada, Z.J., Pham, C., Kalbarczyk, Z., Iyer, R.K.: Hypervisor Introspection: A Technique for Evading Passive Virtual Machine Monitoring. In: 9th USENIX Workshop on Offensive Technologies. USENIX Association (2015)
17. Yoshioka, K., Hosobuchi, Y., Orii, T., Matsumoto, T.: Your Sandbox is Blinded: Impact of Decoy Injection to Public Malware Analysis Systems. Journal of Information Processing 19, 153–168 (2011)