Advanced or not? A comparative study of the use of anti-debugging and anti-VM techniques in generic and targeted malware

Ping Chen, Christophe Huygens, Lieven Desmet, and Wouter Joosen

iMinds-DistriNet, KU Leuven
3001 Leuven, Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract Malware is becoming more and more advanced. As part of the sophistication, malware typically deploys various anti-debugging and anti-VM techniques to prevent detection. In this paper, we investigate the use of anti-debugging and anti-VM techniques in modern malware, and compare their presence in 16,246 generic and 1,037 targeted malware samples (APTs). As part of this study we found several counter-intuitive trends. In particular, our study concludes that targeted malware does not use more anti-debugging and anti-VM techniques than generic malware, although targeted malware tend to have a lower antivirus detection rate. Moreover, this paper even identifies a decrease over time of the number of anti-VM techniques used in APTs and the *Winwebsec* malware family.

1 Introduction

In recent years, a new category of cyber threats, known as Advanced Persistent Threat (APT), has drawn increasing attention from the industrial security community. APTs have several distinguishing characteristics which make them quite different from traditional threats [7]. For example, APTs target mostly companies in critical sectors and governmental institutions [11]; the threat actors in APT attacks are highly-organized and well-resourced group, and can even be state-sponsored [17], and they use stealthy techniques, stay low and slow to evade detection.

APT attacks are widely assumed to be more advanced than traditional attacks, mainly because the threat actors are highly organized, working in a coordinated way, and are well-resourced, having a full spectrum of attack techniques. However, it is unclear whether the targeted malware (malware used in APT attacks) are also more advanced than generic malware (malware used in traditional attacks) or not. To better understand APT attacks, we investigate the difference between targeted malware and generic malware, in order to answer the research question: *"Is targeted malware more advanced than generic malware?"* In particular, we focus on comparing the usage of anti-debugging and anti-VM techniques in targeted and generic malware.

To defend against malware, defenders have turned to the collection and analysis of malware as mechanisms to understand malware and facilitate detection of malware. In response to this, malware authors developed anti-debugging and anti-VM techniques to avoid being analyzed, hence increasing the difficulty of detection. In this paper, we use the presence of anti-debugging, the presence of anti-VM techniques and the antivirus detection rate as metrics to measure the malware's ability to evade malware analysis and antivirus products. All three measurements can be achieved via static analysis on the malware samples.

By analyzing 1,037 targeted malware samples, as well as 16,246 generic malware samples from 6 different families, we report the usage of anti-debugging and anti-VM techniques in malware. We then compare the presence measurements between targeted and generic malware, and correlate them with their antivirus detection rate, and we examine their evolution over time.

As part of this study we found several counter-intuitive trends. In particular, our study concludes that targeted malware does not use more anti-debugging and anti-VM techniques than generic malware. Moreover, this paper even identifies a decrease over time of the number of anti-VM techniques used in APTs and the *Winwebsec* malware family.

The contributions in this paper are as follows:

- We report on the presence of anti-debugging and anti-VM techniques on 17,283 malware samples, and their associated antivirus detection rate (Section 4)
- We analyse and discuss the presence of anti-debugging and anti-VM techniques over time (Section 5.2)
- We analyse and discuss the correlation between the presence of anti-debugging and anti-VM techniques and the antivirus detection rate (Section 5.3)

2 Overview

2.1 Research Questions

In this paper, we compare the targeted malware and generic malware by investigating the following research questions:

Q1: Does targeted malware use more anti-debugging techniques?
Q2: Does targeted malware use more anti-VM techniques?
Q3: Does targeted malware have lower antivirus detection rate?

Since APT attacks are more advanced and sophisticated, one might expect that the targeted malware (the weapons of APT attacks) may use more anti-debugging and anti-VM techniques to evade defensive analysis, and have lower antivirus detection rate. We describe the details about these three metrics in Section 3, and present the analysis result on these questions in Section 4.

Additionally, we are interested about the evolution of the usage of anti-debugging and anti-VM techniques, and how does the use of anti-debugging and anti-VM techniques impact antivirus detection. More specifically, we test the following hypotheses:

H1a: The use of anti-debugging techniques in malware is increasing over time
H1b: The use of anti-VM techniques in malware is increasing over time
H2a: The use of anti-debugging techniques has negative effect on antivirus detection
H2b: The use of anti-VM techniques has negative effect on antivirus detection

While defenders put more and more effort to fight against malware, we assume malware authors are using more and more anti-debugging and anti-VM techniques to thwart the defense, in other words, the use of anti-debugging and anti-VM techniques in malware might increase over years. And the use of these evasive techniques might help malware to evade some antivirus products. To test the hypotheses, we present correlation analysis in Section 5.

2.2 Dataset

The targeted malware samples used in our study are collected from various publicly available reports on APT attacks [17,24,15,9,14,19]. These reports are published by security companies such as FireEye and Kaspersky, to provide technical analysis over various APT attacks, and they typically include the hashes of the discovered targeted malware. With the malware hashes, we then use VirusTotal Private API [2] to search and download these samples.

In this way, we collected 1,037 targeted malware samples ranging from 2009 to 2014. The date information of a malware is extracted from the timestamp attribute in a PE file. For our comparative study, a dataset of more than 16,000 malware samples that belong to 6 generic malware families was collected from VirusTotal. For each malware family and each year (from 2009 to 2014), we use VirusTotal Private API to search for maximum 600 malware samples.

Compared to targeted malware, these malware families are more popular and well understood in the industry. The number of samples belonging to each malware family and the corresponding brief description are shown in Table 1.

Malware family	Discovered year	# of samples	Brief description
Sality	2003	2926	general and multi-purpose [5]
Zbot	2007	3131	banking trojan
Winwebsec	2009	2741	rogueware, fake antivirus [6]
Ramnit	2010	2950	information stealer [4]
Zeroaccess	2011	1787	botnet, bitcoin mining [22]
Reveton	2012	1711	ransomware [25]
Targeted (APT)	2009	1037	targeted malware

Table 1: Overview of malware dataset

3 Metrics

In this paper, we use the presence of anti-debugging, the presence of anti-VM techniques and the antivirus detection rate as metrics to measure the malware's ability to evade malware analysis and antivirus products.

We only focus on these three metrics that can be detected through static analysis. While there are other metrics that can be used to measure the sophistication of malware, such as stealthy network communication, self-deleting execution, they require executing the malware in a targeted environment. Since it is difficult to determine the targeted environment for executing malware, we leave out the dynamic analysis.

3.1 Anti-debugging techniques

In order to thwart debuggers, malware authors use anti-debugging techniques to detect the presence of debuggers and compromise the debugging process. There are many anti-debugging techniques, we focus on detecting the anti-debugging techniques that are known in literature [10]. Since the complete list of anti-debugging techniques is too verbose, we only show those are detected in our malware dataset, as shown in Table 2.

The anti-debugging techniques that are found in our study can be categorized into three types. The use of Windows APIs is the easiest way to detect a debugger. Additionally, malware can check several flags within the PEB (Process Environment Block) structure and the process' default heap structure to detect debuggers. The third way to use some instructions that trigger characteristic behavior of debuggers (e.g., use `RDTSC` to measure execution time).

Type	Name
Windows APIs	`IsDebuggerPresent, SetUnhandledExceptionFilter, FindWindow, CheckRemoteDebuggerPresent, NtSetInformationThread, NtQueryInformationProcess, GetProcessHeap, GetTickCount, NtQuerySystemInformation, OutputDebugString, BlockInput, QueryPerformanceCounter, VirtualProtect, SuspendThread, WaitForDebugEvent, SwitchDesktop, CreateToolhelp32Snapshot`
Flags	`PEB fields (NtGlobalFlag, BeingDebugged)` `Heap fields (ForceFlags, Flags)`
Instructions	`RDTSC, RDPMC, RDMSR, ICEBP, INT3, INT1`

Table 2: Popular anti-debugging techniques

To detect these anti-debugging techniques in a malware sample, we first look for the Windows APIs in the import address table (IAT) of the PE file. Next, we use IDA [1] to automatically disassemble the sample and generate an assembly listing file, and then search for the specific instructions in the assembly listing

file to detect the use of flags and instructions. If any of these techniques are found in the IAT or the assembly listing file, we consider the malware sample use anti-debugging techniques.

3.2 Anti-VM techniques

There are mainly three types of VM detection techniques [20,21]: (1) Interaction based. Sandboxes emulate physical systems, but without a human user. Malware detects VM by checking common human interactions such as mouse movement and mouse clicks. (2) Artifacts based. Virtual machines may have unique artifacts such as service list, registry keys, etc. And some CPU instructions such as `SIDT` have characteristic results when executed inside virtual machines. Malware can leverage these differences to detect sandboxing environment. (3) Timing based. Due to the large number of file samples to examine, sandboxes typically monitor files for a few minutes. Malware authors can configure the malware to execute only after some sleeps, or after a given date and time, in order to avoid being analyzed. Table 3 shows the anti-VM techniques that are found in our malware samples. Details about these techniques can be found in [20,21].

Type	Name
Windows APIs	`GetCursorPos, Sleep, NtOpenDirectoryObject, NtEnumerateKey GetSystemFirmwareTable, NtQueryVirtualMemory, NtQueryObject`
Instructions	`SIDT, SLDT, SGDT, STR, IN, SMSW, CPUID`
Strings	`'sbiedll.dll', 'dbghelp.dll', 'vmware'`

Table 3: Popular anti-VM techniques

To detect these anti-VM techniques in a malware sample, we follow the same method for detecting anti-debugging techniques. Additionally, we extract strings from a PE file, in order to search for the specific strings. If any of these techniques are found, we consider the malware sample use anti-VM techniques.

3.3 Antivirus detection rate

Since the adoption of AV products, malware authors are consistently trying to evade them. There are various techniques and tools [18] to build malware that can bypass common AV products.

In this paper, we use antivirus detection rate as a metric to compare malware's ability to bypass AV products. We get the detection results from 56 AV engines provided in VirusTotal. Since AV engines frequently update their signatures in order to detect malware samples that are not previously spotted by their engines, the reports in VirusTotal might not reflect the current status of these AV engines. To compare the AV detection rate of different malware families, we rescanned all malware samples within two days in December 2014 by using VirusTotal's API [2] to get the most recent detection results.

4 General Findings

4.1 The usage of anti-debugging and anti-VM techniques

To answer questions **Q1, Q2**, we first calculate the percentage of samples that use anti-debugging and anti-VM techniques for each malware family. As shown in Table 4, the majority of samples use either anti-debugging or anti-VM techniques. 68.6% targeted malware samples use anti-debugging techniques, which is less than most generic malware families, and 84.2% targeted malware samples use anti-VM techniques, which is more than all generic malware families. Thus by simply comparing the percentage of samples in each family, we can see that anti-debugging techniques are less popular in targeted malware, and anti-VM techniques are more commonly found in targeted malware.

Family	% Anti-debug.	% Anti-VM	Family	% Anti-debug.	% Anti-VM
Sality	89.6%	76.2%	Ramnit	85.8%	71.6%
Zbot	72.9%	39.7%	Zeroaccess	41.6%	50.4%
Winwebsec	80.0%	52.9%	Reveton	74.8%	62.8%
Targeted (APT)	68.6%	84.2%			

Table 4: Percentage of samples using anti-debugging/anti-VM techniques in each malware family

We then calculate the average the number of detected anti-debugging/anti-VM techniques in each family, and compare the average numbers of generic malware family to targeted malware using Z-test. Z-test is a statistical function that can be used to compare means of two samples. With a Z value bigger than 2.58 and p-value smaller than 1%, we can say that the means of two samples are significantly different.

As shown in Table 5, the average number of detected anti-debugging techniques in targeted malware is neither smallest nor biggest. Since all the Z values are bigger than 2.58, with p-values smaller than 1%, we can accept all the hypotheses that the average number of detected anti-debugging in targeted malware is significantly different to all generic malware families. In other words, targeted malware do not necessarily use more anti-debugging techniques than generic malware. Thus the answer to question **Q1** is still negative.

As for the use of anti-VM techniques, it is the same case as the use of anti-debugging techniques. Targeted malware do not necessarily use more anti-VM techniques than generic malware. Hence the answer to question **Q2** is also negative. The results can be better illustrated in box plots. As shown in Figure 1, the average number of anti-debugging/anti-VM techniques in targeted malware is less than some generic malware family.

Malware Family	Anti-debugging		Anti-VM	
	# of techniques	Z value, p-value	# of techniques	Z value, p-value
Sality	3.59	$11.4, 4.17 \times 10^{-30}$	1.25	$8.4, 3.60^{-17}$
Zbot	2.05	$6.2, 6.34 \times 10^{-10}$	0.48	$15.9, 5.83^{-57}$
Winwebsec	1.75	$11.4, 2.63 \times 10^{-30}$	0.71	$8.8, 1.06^{-18}$
Ramnit	3.76	$13.3, 2.57 \times 10^{-40}$	1.30	$10.2, 1.57^{-24}$
Zeroaccess	0.96	$20.3, 2.27 \times 10^{-91}$	0.17	40.0, 0
Reveton	1.78	$7.5, 4.89 \times 10^{-14}$	0.48	$15.2, 2.45^{-52}$
Targeted (APT)	2.57	Not Applicable	0.94	Not Applicable

Table 5: Average number of anti-debugging/anti-VM techniques in each family

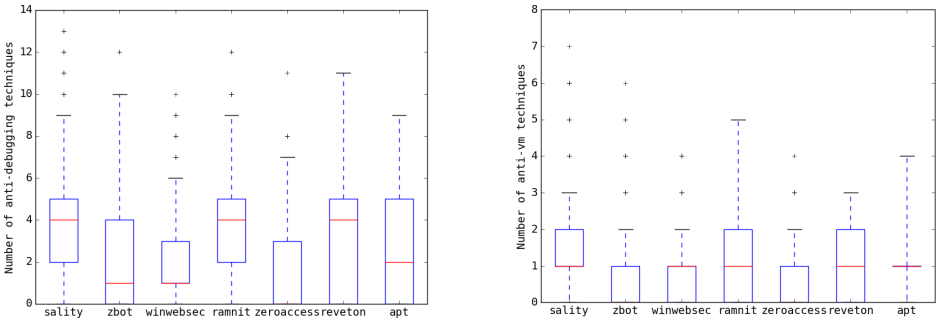

Figure 1. Number of detected anti-debugging/anti-VM techniques in each sample in each family

4.2 Antivirus detection rate

To answer question **Q3**, we calculate the average number of antivirus detections from 56 AV scanners for each malware family, and then compare the average numbers of generic malware family to targeted malware using Z-test. As shown inTable 6, targeted malware has the smallest average number of antivirus detections. And the Z-test results shows that all the Z values are bigger than 2.58, with p-value smaller than 1%. So we accept the hypothesis that targeted malware has significant lower antivirus detections than generic malware, and the the answer to question **Q3** is positive.

Family	# detections	Z value, p-value	Family	# detections	Z value, p-value
Sality	45.7	20.2, 1.35×10^{-90}	Ramnit	49.8	37.8, 0
Zbot	44.6	15.3, 8.49×10^{-50}	Zeroaccess	47.9	36.0, 3.04^{-284}
Winwebsec	46.7	35.5, 4.39×10^{-276}	Reveton	44.5	16.5, 1.71^{-61}
APT	39.5	Not Applicable			

Table 6: Average number of antivirus detections for each malware family

To better illustrate the result, we made a box plot to show the number of AV engines that detected each sample in each family (in Figure 2). We can clearly observe that targeted malware have a lower antivirus detection rate, compared to generic malware. Figure 2 shows that all box plots have long whiskers (with the exception of the Reveton malware), which indicates some malware samples (0.8%) are only detected by a few antivirus engines (less than 10).

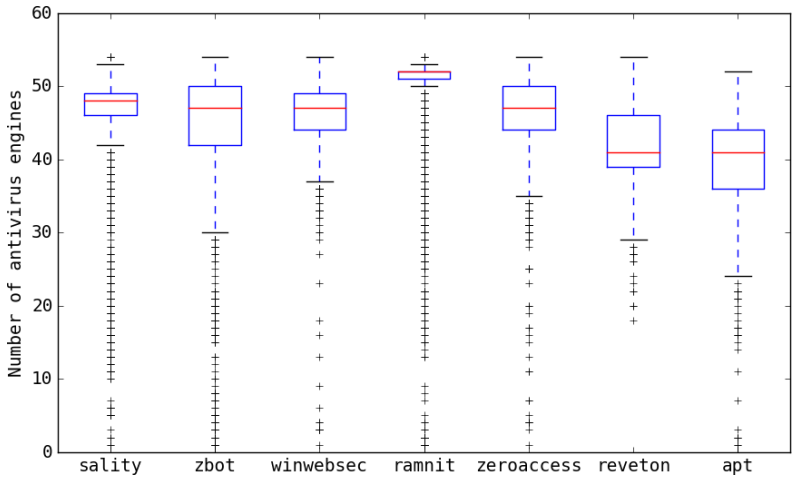

Figure 2. Number of AV engines that detected each sample in each family

As for the evolution of antivirus detection rate (in Figure 3), we can observe that the detection rate tends to decrease over years. This is because malware samples that have been discovered earlier are well populated in antivirus repositories, and being analyzed more often than newly discovered malware samples. Compared to generic malware, targeted malware samples have lower detections for most of the time. We would expect older malware samples to have high detections, but there are still about 13 antivirus engines that cannot detect targeted malware that already discovered in 2009 and 2010.

5 Correlation Analysis

In order to test hypothesis **H1a, H1b, H2a, H2b**, we use Spearman's rank correlation to investigate the evolution of the use of anti-debugging and anti-VM techniques, and the correlation between the use of anti-debugging (or anti-VM) techniques and antivirus detection rate.

5.1 Spearman correlation

Spearman's rank correlation coefficient is a nonparametric measure of the monotonicity of the relationship between two variables. It is defined as the Pearson

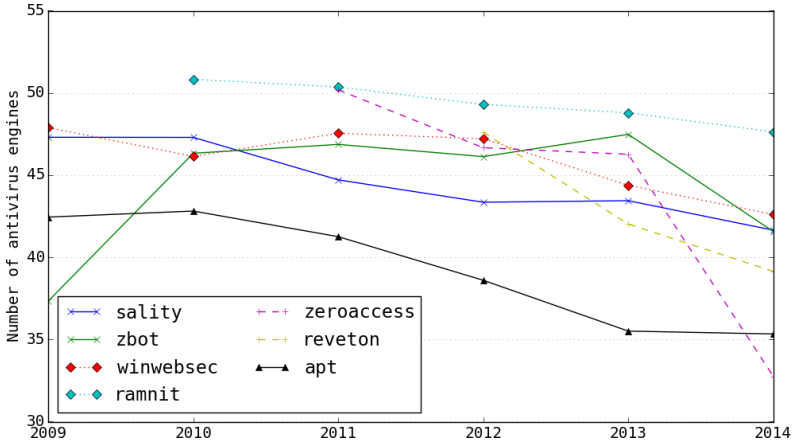

Figure 3. Evolution of antivirus detection rate

correlation coefficient between the ranked variables. However, unlike the Pearson correlation, the Spearman correlation does not assume that both variables are normally distributed. It is a nonparametric statistic, which do not rely on assumptions that the dataset is drawn from a given probability distribution. The result of Spearman correlation varies between -1 and $+1$, and a positive coefficient implies that as one variable increases, the other variable also increases and vice versa. When using Spearman correlation to test statistical dependence, we set the significance level to 5%. The p-value is calculated using Student's t-distribution. We accept the hypothesis only if the p-value is smaller than the significance level.

5.2 Evolution of the use of anti-debugging and anti-VM techniques

To test hypothesis **H1a, H1b**, we use Spearman correlation to measure the correlation between the number of anti-debugging/anti-VM techniques found in malware and the time when the malware is created. The build date of a malware sample is extracted from the timestamp attribute in the PE file. While the timestamp attribute might be incorrect, since malware authors can set arbitrary value for it, there is little incentive for them to do this.

Table 7 shows the Spearman correlation coefficient and p-value for each malware family. We can observe that there is positive correlation for most malware families, which implies that malware authors tend to use more and more anti-debugging techniques over years. The *Winwebsec* and *Zbot* family also have a positive correlation coefficient, but the p-values are bigger than the significance level, thus we reject the hypothesis for *Winwebsec* and *Zbot* family.

While for the use of anti-VM techniques, only four malware families have a positive correlation coefficient, the others do not show a positive correlation between the use of anti-VM techniques and build date. The *Winwebsec* and APT

family have negative correlation coefficients, and the p-values are smaller than the significance level, which implies that the use of anti-VM techniques decreases over years. We think this decrease may be attributed to the great increase in the usage of virtualization. Some malware authors are starting to realize that the presence of a virtual machine does not necessarily mean the malware is being analyzed, since more and more organizations are adopting virtualization technology.

Malware Family	anti-debugging vs. time		anti-VM vs. time	
	coefficient	p-value	coefficient	p-value
Sality	0.23	9.1×10^{-38}	0.31	1.9×10^{-66}
Zbot	0.31	0.08	−0.01	0.39
Winwebsec	0.02	0.36	−0.43	6.7×10^{-129}
Ramnit	0.29	6.1×10^{-62}	0.26	1.7×10^{-48}
Zeroaccess	0.56	4.3×10^{-149}	0.52	8.2×10^{-127}
Reveton	0.45	2.2×10^{-85}	0.54	2.1×10^{-129}
Targeted (APT)	0.29	1.1×10^{-20}	−0.26	4.6×10^{-16}

Table 7: Spearman correlation between the use of anti-debugging (or anti-VM) techniques and build date

To better illustrate the evolution of the use of anti-debugging and anti-VM techniques, we group malware samples by the year in which they are compiled and then calculate the percentage of samples using anti-debugging (or anti-VM) techniques in each group. As shown in Figure 4 and Figure 5, the percentage of samples using anti-debugging techniques in APT malware tend to go up, while the percentage of samples using anti-VM techniques decrease over years. The evolution trends are consistent with the Spearman correlation coefficients in Table 7.

5.3 Correlation between the use of anti-debugging (or anti-VM) techniques and antivirus detection rate

To test hypothesis **H2a, H2b**, we use Spearman correlation to measure the correlation between the number of anti-debugging (or anti-VM) techniques found in malware and the number of positive detections. As shown in Table 8, most malware families (except the *Winwebsec* malware) show negative correlation between the use of anti-debugging techniques and antivirus detection rate, which implies that the use of anti-debugging techniques might help malware to evade antivirus products. While for the use of anti-VM techniques, there are four malware families having a negative correlation coefficient. The *Winwebsec* and APT malware show positive correlation between the use of anti-VM techniques and antivirus detection rate, this might due to the decreasing use of anti-VM techniques in both families, as shown in the previous section.

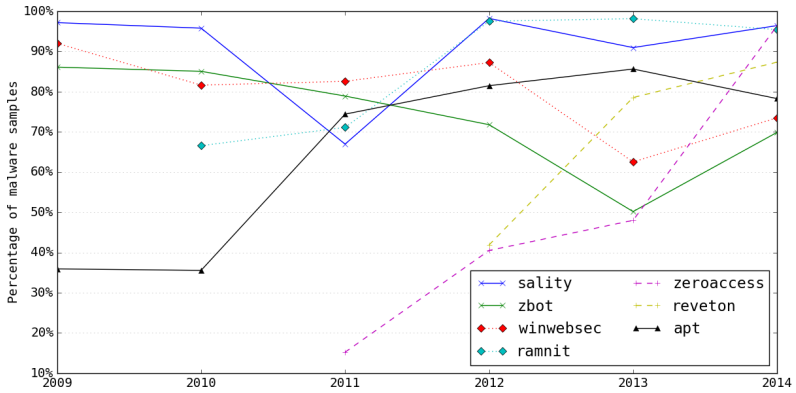

Figure 4. Evolution of the use of anti-debugging techniques

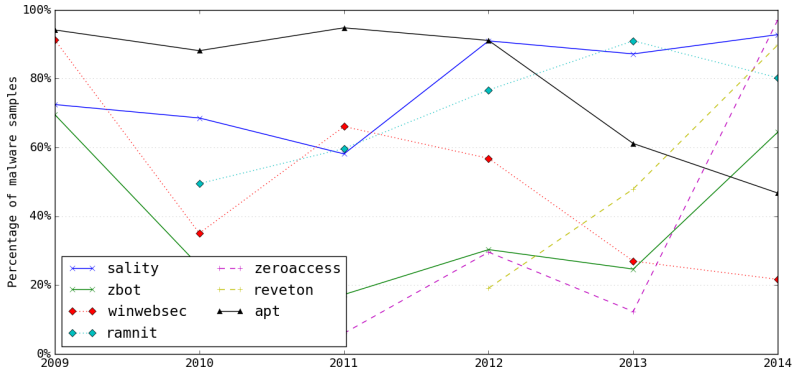

Figure 5. Evolution of the use of anti-VM techniques

Malware Family	detection rate vs. anti-debugging		detection rate vs. anti-VM	
	coefficient	p-value	coefficient	p-value
Sality	-0.1	8.1×10^{-9}	-0.07	5.2×10^{-5}
Zbot	-0.17	3.3×10^{-22}	-0.20	3.8×10^{-30}
Winwebsec	0.05	0.004	0.29	3.7×10^{-57}
Ramnit	-0.13	1.6×10^{-13}	0.004	0.80
Zeroaccess	-0.63	1.1×10^{-198}	-0.61	8.7×10^{-183}
Reveton	-0.22	7.2×10^{-21}	-0.30	3.5×10^{-37}
Targeted (APT)	-0.26	1.2×10^{-16}	0.13	1.6×10^{-5}

Table 8: Spearman correlation between the use of anti-debugging (or anti-VM) techniques and antivirus detection rate

5.4 Summary

We summarize the hypotheses testing results in Table 9. For the use of anti-debugging techniques, hypothesis **H1a** and **H2a** are accepted for targeted malware and most generic malware (except the *Winwebsec* and *Zbot* family), which indicates that both targeted and generic malware are increasing use anti-debugging techniques and the use of anti-debugging techniques might help malware to evade antivirus products.

For the use of anti-VM techniques, we observe two different trends. Some malware families (*Sality, Ramnit, Reveton*) accept hypothesis **H1b** and **H2b**, while targeted malware and *Winwebsec* malware reject hypothesis **H1b** and **H2b**. There are two possible explanation for the decreasing usage of anti-VM techniques in targeted malware and *Winwebsec* malware: (1) Some targeted machines are increasingly using virtualization technology, thus malware authors discard anti-VM techniques in order to target these machines. (2) Malware authors are using new anti-VM techniques which we cannot detect.

Family	**H1a**	**H1b**	**H2a**	**H2b**	Family	**H1a**	**H1b**	**H2a**	**H2b**
Sality	A	A	A	A	Ramnit	A	R	A	NA
Zbot	NA	NA	A	A	Zeroaccess	A	A	A	A
Winwebsec	NA	R	R	R	Reveton	A	A	A	A
APT	A	R	A	R					

A: Accepted, NA: Not Accepted due to a bigger p-value
R: Rejected, the opposite hypothesis is accepted

Table 9: Hypotheses testing results with Spearman correlation

6 Related work

APT attacks. Research on targeted attacks and APTs are mostly from industrial security community. Security service providers (e.g., FireEye, Symantec) periodically publish technical reports that various APT attacks [17,24,15,9,14,19]. Recently, this topic also become hot in academia. In [23], Thonnard et al. conducted an in-depth analysis of 18,580 targeted email attacks, showing that a targeted attack is typically a long-running campaign highly focusing on a limited number of organizations. In [16], Le Blond et al. presented an empirical analysis of targeted attacks against a Non-Governmental Organization (NGO), showing that social engineering is an important component of targeted attacks.

Giura and Wang [12] introduced an attack pyramid model to model targeted attacks, and implemented a large-scale distributed computing framework to detect APT attacks. Hutchins et al. [13] proposed a kill chain model to track targeted attack campaigns and proposed an intelligence-driven strategy to adapt defense based on the gathered intelligence.

Anti-debugging and anti-VM in malware Chen et. al. developed a detailed taxonomy for anti-debugging and anti-VM techniques [8], and they also proposed a novel defensive approach to mislead the attacker, by disguising the production systems as monitoring systems. A recent survey of the use of anti-debugging and anti-VM techniques in malware is presented by Branco et. al. [3], in which they introduced various static detection methods for anti-debugging and anti-VM techniques, and run an analysis over 4 million samples to show the state of evasion techniques in use.

7 Conclusion

In this paper, we have analyzed the presence of anti-debugging and anti-VM techniques in 17,283 malware samples, by using static analysis. As part of this analysis, we have compared the presence measurements between targeted and generic malware, we have correlated them with their antivirus detection rate, and we have examined their evolution over time.

As expected, we have observed that both targeted malware and generic malware often use anti-debugging and anti-VM techniques. The analysis results also confirmed the hypotheses that the number of anti-debugging techniques used tend to increase over years, and that their presence has a negative correlation with the antivirus detection rate.

At the same time, this study revealed two counter-intuitive trends: (1) The study concluded that targeted malware does not use more anti-debugging and anti-VM techniques than generic malware, whereas targeted malware tend to have a lower antivirus detection rate; (2) This paper identified a decrease over time of the number of anti-VM techniques used in APTs and the winwebsec malware family. This conflicts with the original hypothesis that APTs try to evade analysis and detection by using anti-VM techniques, and strongly contrasts with other malware families where the opposite trend holds.

Acknowledgements We would like to thank VirusTotal for providing us a private API, and the anonymous reviewers for their comments. This research is partially funded by the Research Fund KU Leuven, iMinds, IWT, and by the EU FP7 projects WebSand, NESSoS and STREWS. With the financial support from the Prevention of and Fight against Crime Programme of the European Union (B-CCENTRE).

References

1. IDA. `https://www.hex-rays.com/products/ida/`.
2. VirusTotal Private API. `https://www.virustotal.com`.
3. Rodrigo Rubira Branco, Gabriel Negreira Barbosa, and Pedro Drimel Neto. Scientific but Not Academical Overview of Malware Anti-Debugging, Anti-Disassembly and Anti-VM. *Blackhat*, 2012.

4. Microsoft Malware Protection Center. Win32/Ramnit. http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32/Ramnit.
5. Microsoft Malware Protection Center. Win32/Sality. http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32/Sality.
6. Microsoft Malware Protection Center. Win32/Winwebsec. http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32/Winwebsec.
7. Ping Chen, Lieven Desmet, and Christophe Huygens. A Study on Advanced Persistent Threats. In *Proceedings of the 15th IFIP TC6/TC11 Conference on Communications and Multimedia Security*, 2014.
8. Xu Chen et al. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *IEEE International Conference on Dependable Systems and Networks*, pages 177–186, 2008.
9. Cylance. Operation Cleaver. 2014.
10. Peter Ferrie. The Ultimate Anti-Debugging Reference. 2011.
11. FireEye. FireEye Advanced Threat Report: 2013. 2014.
12. Paul Giura and Wei Wang. Using large scale distributed computing to unveil advanced persistent threats. *SCIENCE*, 1(3), 2013.
13. E. M. Hutchins et al. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. In *Proceedings of the 6th International Conference on Information Warfare and Security*, 2013.
14. Kaspersky. The Icefog APT: A Tale of Cloak and Three Daggers. 2013.
15. Kaspersky. Energetic Bear - Crouching Yeti. 2014.
16. Stevens Le Blond, Adina Uritesc, Cédric Gilbert, Zheng Leong Chua, Prateek Saxena, and Engin Kirda. A look at targeted attacks through the lense of an ngo. In *Proceedings of the 23rd USENIX conference on Security Symposium*, pages 543–558. USENIX Association, 2014.
17. Mandiant. APT1: Exposing One of China's Cyber Espionage Unit. 2013.
18. Debasis Mohanty. Anti-Virus Evasion Techniques Virus Evasion Techniques Virus Evasion Techniques and Countermeasures. http://repo.hackerzvoice.net/depot_madchat/vxdevl/papers/vxers/AV_Evasion.pdf.
19. Arbor Networks. Illuminating the Etumbot APT Backdoor. 2014.
20. N. Rin. Virtual Machines Detection Enhanced. http://artemonsecurity.com/vmde.pdf, 2013.
21. Abhishek Singh and Zheng Bu. Hot Knives Through Butter: Evading File-based Sandboxes. 2014.
22. Symantec. Trojan.Zeroaccess. http://www.symantec.com/security_response/writeup.jsp?docid=2011-071314-0410-99.
23. Olivier Thonnard et al. Industrial espionage and targeted attacks: Understanding the characteristics of an escalating threat. In *Proceedings of the 15th Symposium on Research in Attacks, Intrusions, and Defenses*, pages 64–85. Springer, 2012.
24. Nart Villeneuve et al. Operation Ke3chang: Targeted Attacks Against Ministries of Foreign Affairs. 2013.
25. Wikipedia. Ransomware -Reveton. http://en.wikipedia.org/wiki/Ransomware#Reveton.