Large scale malware collection: lessons learned

Julio Canto
Hispasec Sistemas
Spain
jcanto@hispasec.com

Marc Dacier
Symantec Research Labs
Sophia Antipolis, France
marc_dacier@symantec.com

Engin Kirda, Corrado Leita
EURECOM
Sophia Antipolis, France
{leita,kirda}@eurecom.fr

Abstract

In order to assure accuracy and realism of resilience assessment methods and tools, it is essential to have access to field data that are unbiased and representative. Several initiatives are taking place that offer access to malware samples for research purposes. Papers are published where techniques have been assessed thanks to these samples. Definition of benchmarking datasets is the next step ahead. In this paper, we report on the lessons learned while collecting and analysing malware samples in a large scale collaborative effort. Three different environments are described and their integration used to highlight the open issues that remain with such data collection. Three main lessons are offered to the reader. First, creation of representative malware samples datasets is probably an impossible task. Second, false negative alerts are not what we think they are. Third, false positive alerts exist where we were not used to see them. These three lessons have to be taken into account by those who want to assess the resilience of techniques with respect to malicious faults.

1 Introduction

This work reports on lessons learned with the largest freely available malware collection initiative, namely VirusTotal [2], as well as with two other threats related collection engines: SGNET [7] and ANUBIS [1]. VirusTotal is an online service where users can freely upload a file and check, thanks to a large number of anti virus scanners, if that file contains some sort of malware or not. SGNET is a distributed system of honeypots relying on the Scriptgen technology [6] that aims at capturing malware observed in the wild. ANUBIS is a sandbox that is available as an online service where users can freely submit malware to see what behaviors they exhibit when executed and monitored during a limited amount of time.

These three systems are in use and have collected a significant amount of malware samples. They offer us three different viewpoints when looking at these samples. Some ongoing work is taking place in the context of the European funded WOMBAT project [3] to use these tools for threats intelligence discovery. More information about these activities can be found on the WOMBAT project web site [3]. In the following pages, we aim at sharing with a larger community the lessons we have learned. Namely, we want to make the following points clear to all those who are trying to assess the resilience of systems with respect to malicious faults:

Lesson 1 The amount of active malware is extremely important and new ones are created on a daily basis at a very high pace. Therefore, every attempt to build a fix, static, representative set of malware samples for evaluation purposes is deemed to generate artificial results.

Lesson 2 It is an error to consider that a given antivirus fails when it does not generate an alert for a malware recognised as such by others. The reasons therefore are described in this paper and they highlight the difficulties linked to the assessment of antivirus tools, or other similar intrusion detection engines. More precisely, they force us to revisit the underlying concept of false negative alerts.

Lesson 3 It is an error to consider that a given antivirus does not fail when it does generate an alert for a malware recognised as such by others. Hereto, details are given in the paper and they lead to the same conclusions regarding the assessments of AV tools by forcing us to revisit the underlying concept of false positive alerts.

The rest of the document is structured as follows. Section 2 describes the VirusTotal setup and how the results obtained in using it led to the first lesson learned. Section 3 introduces the SGNET and ANUBIS environments that are used to provide a refined analysis of the results obtained from VirusTotal for some malware samples. Section 4 discusses the second and third lesson mentioned here above. Section 5 concludes the paper.

2 VirusTotal

2.1 Description of the data collection and redistribution mechanisms

VirusTotal is a free online service that enables Internet users to scan dubious files thanks to 36 different antivirus tools. Initially, this service had been designed as an internal tool to be used within Hispasec Sistemas, the company that created it and still manages it. The goal was to have a flexible framework to check the behavior of antivirus products confronted with malware threats happening at that time. In June 2004, the company decided to make it a public, free, online service for the benefit of the whole Internet community.

The main functionality provided is very simple: the user sends a file to the system, thanks to the email or the web interface. He will get a report back when all AV tools will have finished examining the submitted file. That report includes the output of each engine, URLs with extra information about the potential threat (if any), file metadata size, various hashes of the file, etc. It can also contain, when appropriate, packer identification or the Portable Executable (PE) structure information of the malware.

VirusTotal, with its 36 AV engines, offers a valuable service not only to the end users but also to the community of the AV vendors. Indeed, VirusTotal can provide them with samples of malware that match certain criteria of interest to them.

In the general case, VirusTotal sends a malware sample to AV vendor X if

- at least one other AV engine has detected the sample as being malicious whereas the AV engine of X has not.
- or if the AV engine from X has detected that sample as being malicious thanks to a generic pattern or a heuristic.

Most AV vendors do follow these two rules but some of them impose other criteria. For instance, some have decided to get samples that are detected by, at least, N out of K AV engines and that their own has missed. Others do restrict even further the conditions by imposing that all engines from a well defined subset of engines must have detected the sample and that their own has missed it.

Clearly, the amount of samples to be sent to the AV vendors is a function of the filtering rules they have chosen. It is worth noting though that, in the general case, some vendors do get as many as 10000 samples per day!

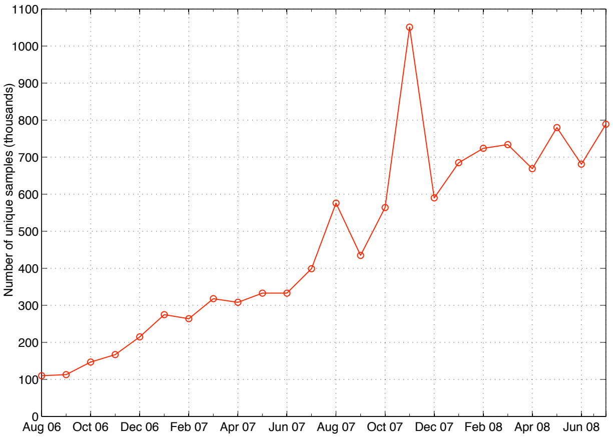

Figure 1. Amount of samples received every month

2.2 Lesson 1: Evolution of the threats

Thanks to its popularity, VirusTotal does get a very large amount of files and is, therefore, in a privileged situation to observe the evolution of the threats over time. During the first month of public exposure the site received 8.400 files, in September 2006 more than 120.000 files, today VirusTotal scans almost 1 million files per month! Around 70% of these files are detected as being malicious by at least one antivirus engine. For the sake of conciseness, we will refer to these samples as the malicious samples in the following. It is worth noting that most of these samples are actually unique instances of malware. Indeed, in a given month, around 70% of the malicious samples do have a unique MD5 hash value!. For example, in June 2008, 681.561 were detected as malicious and 493.776 of them had a unique MD5 has value. This comes down to a new sample every six seconds!

This could be a concrete manifestation of a phenomenon that is believed to have started around 3 years ago when the motivation of malware writers switched from ego boosting to financial earning. It has been reported that they adopted a strategy of mass creation of modified versions of different families by means of semi automated repacking techniques and other different obfuscations methods, hoping that such long tail distribution of appearances would improve their infection rate as well as their survivability in the infected systems. Years ago, most of the impact due to malware infections was due to a limited number of families of malware whereas today the ratio of infection per family seems to be more evenly distributed over a large number of families. [9] provides other evidences of this massive increase of malware samples during the last 12 months. It is part of the WOMBAT objectives to validate, or not, such conjectures by means of rigorous analysis.

This observation leads to the conclusion that, assuming one could build a representative dataset of malware samples observed at time T, one cannot ensure that the same dataset will still be representative of the threats observed at time T'. As a consequence, extreme caution must be taken when trying to build datasets for the sake of testing the efficiency of AV or intrusion detection mechanisms. This is all the more true when it comes to try to define datasets for benchmarking purposes. Some convincing arguments must be produced in order to validate their representativeness under the light of the previous observation.

2.3 False Positive and False Negatives

Before discussing the issues surrounding the notions of false positives and false negatives, there are a couple of important remarks to be made in order to understand the specificities of the VirusTotal setup.

First of all, most AV engines do offer a layered approach against malware. The AV engine used to scan a binary file against a set of signatures does not necessarily represent the complete detection capability of the corresponding product. This is true, for instance, for behavioral signatures that can detect abnormal actions while the program is running. These features are not used within the VirusTotal framework. VirusTotal, by design, uses the command line interfaces provided by the vendors to invoke their AV engine. As said, in many cases, such invocation can bypass other detection capabilities that are present in the desktop counterpart of that product. This can lead to an apparent degradation of the performance of a given product. This is especially true for samples that are packed or compressed in some way. If the unpacking procedure is implemented outside the AV engine invoked, it will not take place in the VirusTotal environment and no signature will be found.

On the other hand, some scanners do rely on heuristics that can be tuned thanks to some parameters. Experience shows that the parameters used by some vendors in the VirusTotal environment are set to different values than the ones in place in the desktop version of their products. As a result, this improves their detection capability but, at the same time, greatly increases their false positive rate as well.

These two important information have to be kept in mind when looking at statistics coming from VirusTotal reporting the false positive or negative rates of AV engines. However, there are other, more subtle reasons, that undermine the mere notions of false positive and false negatives. They are described in the coming Sections but, before that, we present two other threat related data collection environments we use to derive them.

3 SGNET and ANUBIS

3.1 SGNET

SGNET [5] is a distributed honeypot deployment aiming at collecting information on the Internet malicious activity. SGNET is the most recent evolution of the research work performed within the Leurré.com project [7]. SGNET honeypots are deployed on low-end hosts provided by volunteering partners interested in exploiting the data collected by the project.

SGNET integrates different tools, namely ScriptGen [6], Argos [8] and Nepenthes [4] and exploits their characteristics to emulate code injection attacks and collect malware. SGNET benefits from a set of properties that enable it to gather a very peculiar view on Internet attacks and malware.

Firstly, SGNET is protocol agnostic. Following the idea initially proposed in ScriptGen, no assumption is made on the structure of network protocols and on their interaction. Through the usage of bioinformatics techniques, SGNET is able to *learn* the behavior of network protocols and thus handle exploits without an a-priori assumption on their behavior. This potentially allows SGNET to handle new or rare exploits that may not be supported by other malware collection solutions such as Nepenthes.

Secondly, SGNET retrieves in depth information on the structure of the observed attacks. This information is collected in a central database and presented at different aggregation levels. Such information is then enriched through a number of analysis tools organized in an easily extensible framework.

The information enrichment properties of SGNET allow to correlate the observations with a large variety of tools that are automatically run on the collected data: geolocation information on the origin of the attackers, DNS information, and much more. In this context, the ability of SGNET to emulate code injection attacks up to the point of the download of malware samples is extremely valuable. Every malware sample collected by the SGNET framework is automatically submitted to VirusTotal and Anubis, and the corresponding information provided by these services is stored in the SGNET database as an enrichment of the honeypot observations.

While in the Anubis case each collected sample is submitted for analysis a single time, a single sample is submitted multiple times to VirusTotal on a daily basis in order to gather statistics on the evolution of the ability of different AV vendors to recognize it. Each collected malware is thus re-submitted to VirusTotal at least 30 times, and the result of each submission is stored in the SGNET database. After this period, if no evident changes have happened in the output of the AV products in the last 7 days, the submission for the sample is stopped.

3.2 ANUBIS

Anubis is a tool that automates the process of analyzing malware to allow a human analyst to quickly get a basic understanding of the actions of an unknown executable. Running a binary under Anubis results in the generation of a report that contains information to give the human analyst a very good impression about the purpose and the functionality of the analyzed sample. This report includes detailed data about modifications made to the Windows registry and to the file system, information about interactions with the Windows Service Manager and other processes, as well as a complete log of all generated network traffic.

Anubis uses emulation to run the unknown binary together with a complete operating system in software. Thus, the malware is never executed directly on the processor. The analysis is comprehensive because Anubis monitors calls to native kernel functions as well as calls to Windows API functions. It also provides support for the analysis of complex function call arguments that contain pointers to other objects. The complete control offered by an emulator potentially allows the analysis that is performed to be even more fine-grain. Similar to the functionality typically provided by a debugger, the code under analysis can be stopped at any point during its execution and the process state (i.e., registers and virtual address space) can be examined. Unlike a debugger, however, Anubis does not have to resort to breakpoints, which are known to cause problems when used for malicious code analysis. The reason is that software breakpoints directly modify the executable and thus, can be detected by code integrity checks.

4 Lessons learned

4.1 Lesson 2: To be or not to be a false negative

The SGNET database comprises at the 1st of August 2008 a total of 4119 different malware samples, where each sample is identified through its MD5 hash.

The integration of the behavioral information generated by Anubis in the SGNET database highlighted an extremely interesting phenomenon: a good portion of the samples downloaded by SGNET cannot be executed on a Windows host! This is mainly due to problems in the download of the samples, such as premature drop of the TCP connection in case of TCP protocols. Out of 4119 malware samples currently in the SGNET database, we have identified 1045 corrupted malware samples. The technology used by SGNET to download malware is derived from Nepenthes, an open-source honeypot to collect malware samples used by several malware collection projects as their main source of samples. The problem that we are underlining here is thus a generic problem likely to be shared by other malware repositories.

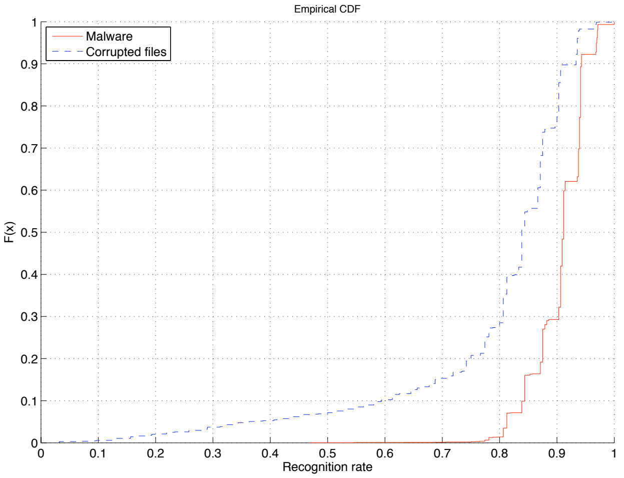

Figure 2. Impact of corrupted samples on AV performance

In order to evaluate the impact of corrupted malware samples when evaluating the performance of the different vendors, we compared the performance of all the AV vendors in handling regular and corrupted samples. As a measure of performance, we took into consideration the recognition rate, that is the ratio between the amount of AV vendors that correctly recognized a sample and the total amount of AV vendors provided by VirusTotal. Among all the reports generated by the resubmission policy for each malware sample, we selected the most recent one, that corresponds to the best recognition rate. Figure 2 shows the CDF for the recognition rates achieved by the AV vendors for the two classes. While 80% of the vendors always recognize the regular malware samples, the CDF for the corrupted ones is much less steep and underlines a significant difference in performance of the various AV solutions.

The output of the AV engine when analyzing such corrupted samples is not easy to define. On the one hand, the specific implementation of an engine or of the corresponding signature may or may not be affected by missing parts of the original binary. On the other hand, different vendors may have different policies with respect to these corrupted files. Figure 3 validates this intuition. Each bar on the X axis corresponds to one of the 36 vendors supported by VirusTotal, and represents the percentage of corrupted samples recognized by that vendor. A minority of the vendors raises almost no alerts for the corrupted files, considering them harmless. But the majority of AV products ignores the inconsistent structure of the executable file (easily detectable looking at its headers) and considers the sample as

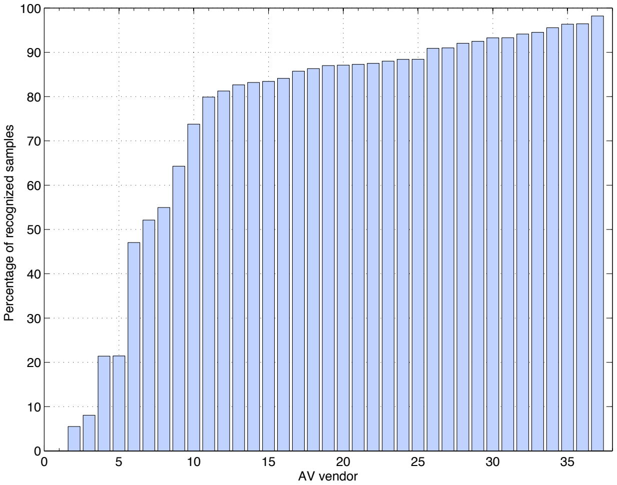

Figure 3. Performance of different vendors when analyzing corrupted samples

Vendor1		Vendor2
Allaple.gen6	3	Net-Worm.Win32.Allaple.e
	106	Net-Worm.Win32.Allaple.b
W32/Virut.P	1	Net-Worm.Win32.Allaple.d
	16	Virus.Win32.Virut.q
	1	Backdoor.Win32.Rbot.bni
	7	Net-Worm.Win32.Allaple.e
	1	Net-Worm.Win32.Allaple.b
W32/Virut.BF	2	Backdoor.Win32.VanBot.ps
	159	Virus.Win32.Virut.n
Allaple.gen10	10	Net-Worm.Win32.Allaple.e
	1	Backdoor.Win32.Rbot.bni
Allaple.gen1	37	Net-Worm.Win32.Allaple.d
	85	Net-Worm.Win32.Allaple.e
	17	Net-Worm.Win32.Allaple.b
W32/Virut.T	2	Net-Worm.Win32.Allaple.b
	2	Virus.Win32.Virut.q
	1	Backdoor.Win32.Rbot.bni

Table 1. Labelling inconsistencies among vendors

malicious.

It is difficult to argue that the AV engines that have recognized precise elements of a malware have, in fact, produced a false positive alert simply because the malware was harmless. Even if corrupted, the “raison d’être” of this file is to be a malware. But if these engines have not produced a false positive, can we mechanically derive that those who have not generated an alert are responsible of a false negative? Should they have warned the user of the presence of a threat? Some would say yes because, for instance in the case of a Trojan horse, this would prevent the user from trying to download the file a second time and get compromised this time. Some may say no since there is no threat present and it is useless to cry for wolves.

Clearly, the response lies in a very precise definition of the terms false positive, negative, threats, alerts, etc. Such a discussion lies outside the scope of this paper but we hope that, under the light of these observations, researchers presenting, e.g., ROC curves highlighting the performance of a given tool will first take the time to very precisely define what a false positive and a false negative is in their own experimental environment.

4.2 Lesson 3: To be or not to be a false positive

The ambiguity in the performance of different engines when facing corrupted files is not the sole problem. Looking at the reports generated for non-corrupted samples, we found a considerable number of inconsistencies in the labels assigned by vendors to the different samples. Two different types of inconsistencies have been identified.

Looking at successive reports for the same sample, one would expect the label given by a vendor to remain constant. We detected instead a considerable amount of variations in the label assigned by a vendor to the same malware sample. In total, we have been able to observe 10314 modifications to the name given to a sample by a given vendor. We identified 1081 different modifications often applied to groups of samples. Many of these modifications consist in a better specification of the name. For instance, 625 MD5s initially classified by a vendor as “Suspicious file” have been later classified as “Win32.Allaple.b”. Other modifications instead involve names associated to completely different behavior, and thus underline a labelling error made by the vendor when generating the signature. For instance, 25 MD5s have been classified by the same vendor as “suspicious”, as “Allaple.gen3”, as “Virus.Win32.Virut.n” and as “W32/Virut.BF” in different days. An exhaustive analysis of all these cases is left for future work, but this brief overview clearly shows the labelling problems inherent in AV signatures.

Labels are assigned by each vendor in a independent way and thus it is not easy to compare labels assigned by different vendors. But even if the label string differs, one would expect the grouping of samples performed by two vendors to be consistent. Grouping the samples in sets according to their label, we have compared the groups generated by two different vendors on a set of 848 samples analyzed on the same day by VirusTotal. We have detected six inconsistencies in the resulting grouping as represented in Table 1. The table shows how the different elements of a set defined by the first vendor are mapped on the sets defined by the second one.

For instance, we found 16 malwares all labeled by the first vendor as “W32/Virut.P” but called with five distinct names by the second vendor. One sample was named “Net-Worm.Win32.Allaple.d”, sixteen other were called “Virus.Win32.Virut.q”, another one was labeled “Backdoor.Win32.Rbot.bni”, seven other samples were identified as “Net-Worm.Win32.Allaple.e” and the last sample was referred to as “Net-Worm.Win32.Allaple.b”.

It must be stressed that we show these six inconsistencies for illustration purposes but that we could have shown many other ones involving other pairs of AV vendors. The point here is not all to fingerpoint at any vendor in particular but, instead, to highlight a problem which, to our knowledge, has not been debated in the literature, namely: ”If a detector raises an alert with an erroneous labeling, does that constitute a false positive?”

It is clear from the previous examples that the labeling problem needs to be taken into consideration when benchmarking AV solutions. A successful detection often gives erroneous information to the user on the nature of the threat that he is dealing with. We are able to identify in our datasets samples that are named in completely different ways by different releases of the signatures for a given AV engine. Also, different engines classify the samples in different and often inconsistent ways that suggest the label information to be extremely unreliable.

Here to, we hope that these observations, motivated by a large scale collection and analysis of malware samples, will motivate some further research not only into the precise definition of notions such as false positive alerts but, more importantly, into concrete solutions to help deciding when an alert is, or not, a false positive in a more precise way.

5 Conclusions

Thanks to the collection and analysis of a large number of malware samples in a collaborative effort between the VirusTotal, SGNET and ANUBIS environments, we have presented in this document three lessons learned from our journey through this large amount of data. First of all, the amount and the dynamics of today’s malware make the building of a representative dataset of malware samples an almost impossible task to achieve. Second, we found cases where two detectors could take opposite decisions with respect to a given malware without any of them being faulty. That observation forces to, at least, redefine in much more precise terms that is usually done, what a false negative alert means. Conversely, we found cases where detectors were generating erroneous alerts that could not be, per se, be qualified as false positive.

Our hope is that these observations will be of use to those who are involved in the assessment of resilience techniques and tools in the presence of malicious faults. We do firmly believe that the lack of precise, formal, definitions of the elements we are trying to measure as well as the lack of representative datasets to carry out testing campaigns are research issues that should be tackled in the near future.

Acknowledgment

This work has been partially supported by the European Commissions through project FP7-ICT-216026-WOMBAT funded by the 7th framework program. The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the European Commission.

References

[1] Anubis, analyzing unknown binaries. Website, 2008. http://anubis.iseclab.org//.

[2] Virustotal, free online virus and malware scan. Website, 2008. http://www.virustotal.com/.

[3] Wombat: a worldwide observatory of malicious behaviors and attack threats. Website, 2008. http://www.wombat-project.org//.

[4] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, September 2006.

[5] C. Leita and M. Dacier. SGNET: a worldwide deployable framework to support the analysis of malware threat models. In *Proceedings of the 7th European Dependable Computing Conference (EDCC 2008)*, May 2008.

[6] C. Leita, M. Dacier, and F. Massicotte. Automatic handling of protocol dependencies and reaction to 0-day attacks with ScriptGen based honeypots. In *RAID 2006, 9th International Symposium on Recent Advances in Intrusion Detection, September 20-22, 2006, Hamburg, Germany*, Sep 2006.

[7] C. Leita, V. H. Pham, O. Thonnard, E. Ramirez-Silva, F. Pouget, E. Kirda, and M. Dacier. The Leurre.com Project: Collecting Internet Threats Information using a Worldwide Distributed Honeynet. In *Proceedings of the 1st Wombat Workshop*, 2008.

[8] G. Portokalidis, A. Slowinska, and H. Bos. Argos: an emulator for fingerprinting zero-day attacks. *Proc. ACM SIGOPS EUROSYS*, 2006.

[9] D. Turner, M. Fossi, E. Johnson, T. Mack, J. Blackbird, S. Entwisle, M. K. Low, D. McKinney, and C. Wueest. Symantec global internet security threat report, trends for july to december 07. Website, April 2008. http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiii_04-2008.en-us.pdf.