Proceeding of International Conference on Systems Computation Automation and Networking 2019

A study to Understand Malware Behavior through Malware Analysis

Om Prakash Samantray
Department of Computer Science
Berhampur University
Berhampur, India
om.prakash02420@gmail.com

Satya Narayan Tripathy
Department of Computer Science
Berhampur University
Berhampur, India
snt.cs@buodisha.edu.in

Susanta Kumar Das
Department of Computer Science
Berhampur University
Berhampur, India
skd.cs@buodisha.edu.in

***Abstract*—Most of the malware detection techniques use malware signatures for detection. It is easy to detect known malicious program in a system but the problem arises when the malware is unknown. Because, unknown malware cannot be detected by using available known malware signatures. Signature based detection techniques fails to detect unknown and zero-day attacks. A novel approach is required to represent malware features effectively to detect obfuscated, unknown, and mutated malware. This paper emphasizes malware behavior, characteristics and properties extracted by different analytic techniques and to decide whether to include them to create behavioral based malware signature. We have made an attempt to understand the malware behavior using a few openly available tools for malware analysis.**

***Index Terms*—Malware Packer, Malware Detection, Malware Signature, Malware Behavior.**

I. Introduction

Malware or malicious software has become a biggest threat to the information industry from past few years. According to AV-Test, an independent IT-security institute, every year the number of malware is increasing in an unprecedented rate despite using malware detection techniques[1]. Despite different anti-malware techniques, malware are left undetectable because of the obfuscation techniques used by malware writers to beat signature-based malware detection systems. One of the major demerits of the signature based detection system is, to frequently update the signature database because malware are generated very rapidly with different signatures [2]. According to Cisco 2017 annual cybersecurity report, 95% of their analyzed malware were of age below 24 hours [3]. This indicates the pace of malware evolution which stands as a challenge for malware research.The problem of signature-based detection system motivates the researchers to think about techniques to deal with new and unknown malware. Malware may perform many malicious operations which are natural for a malicious program but sometimes they may appear in legitimate software. Therefore, we need to consider some other properties or activities to create a consistent behavior-based malware signature. In this paper, we have conducted tests on malware packers and discussed the result. We have also discussed about the asymmetric arrangement of ASCII characters in the malware samples. The test is done on a collection of benign files using different tools, software and online malware detection services. The aim is to convey that; though the malware behavior plays a major role in detecting malware samples, still it cannot be relied upon completely. In this work, we have tested different samples to identify malware behaviour not only in malicious sample but also in benign samples. We have shown that the operations and actions usually found in malware cannot be considered as an important element for malware detection because sometimes these are also performed by benign files.

II. Related Work

Our research is motivated by a work done by Wu et. al.[4] who have categorized malware operations into 4 types based on the type of actions they perform. Though they did not explain particulars of their test but, these four groups of malware activities are found suitable[4]. A similar approach was attempted by researchers in [5] which was meant for malware detection and categorization. They also referred other similar works[6-8] and stated that, the actions of GUI can also be another possible group besides the four groups identified earlier. Legitimate files are usually used as training dataset in data-mining detection techniques which helps us to evaluate the detection engines[9]. These are also used in malware analysis techniques based on comparison of opcode frequency in both legitimate and malware files[10]. A disadvantage of this method is that, the malware creators frequently add benign dead code fragments into the malicious programs which helps them to prevent opcode-frequency detection. Josse et. al. have explained the packing and unpacking issues in their research work[11]. A solution to handle the difficulties of malware packing is, the packer vendors can use some type of taggant in packed software and distribute the taggant among all anti-malware vendors[12]. The taggant must hold details regarding packer dealer, packer variety, customer identification and so on. The idea was defined by Industry Connection Security Group (ICSG)[13]. The popularity and use of packer in malware creation is described by Wang and Wu in their research. They have proposed a universal framework for packing detection[14]. C. Linn et. al. in their research, have collected many packed and unpacked files for experiment. They have conducted the experiment with a total of 3784 packed files and 90% of the files were malware. C. Linn et. al. stated that,

@IEEE 978-1-7281-1524-5

sometimes legitimate programs are also packed to protect from software theft [15].

III. Experiment Requirements

The goal of this study is to make a few contributions to modern malware identification approaches. We believe in a statement that, behavior, properties and operations detected in a malware may also present in legitimate or genuine program. This creates an ambiguous situation for malware detection techniques which are based on typical patterns of behavior. We define the analysis as follows. We have defined the analysis as a function, let the name of the function be Sample Analysis (SA) which is applied on a set of samples S_{all} containing two disjoint sets S_{M} and S_{B} denoting sets of malicious and benign samples respectively.

Here,

$$\begin{aligned} S_{\text{all}} &= S_{\text{M}} U S_{\text{B}} \\ SA : S_{\text{all}} \times O &\rightarrow R \end{aligned} \tag{1}$$

Where, O denotes frequent behaviors of malicious software and are considered suitable.The result R can have any one of the two Boolean values True or False. Consider the following equation:

$$SA(S_{\text{all}}, o) = \text{true if o is identified in} S_{\text{all}}, Else false \tag{2}$$

Assumption 1: When the sample analysis function (SA) is applied on a benign sample b belongs to S_{B}, positively detects some malicious property or operation o belongs to O, in the sample b. In other words, there exist some malicious operation o which belong to the entire set of operations O, which are identified positively for some sample b which belong to the set of benign samples SB. This can be represented as follows.

$$\exists o \in O : SA(b, o) = True, b \in S_{\text{B}}, o \in O \tag{3}$$

In order to validate assumption 1 we will try to show that equation 3 holds good for a few relevant samples. In this paper we emphasize on file packers and properties like ASCII character distribution patterns and frequencies in files.

A. Testing Workbench, Tools and Sample Collection

We have used a Linux-based tool-pack called REMnux, which is popularly used by malware researchers for reverse engineering purpose. It provides a collection of free tools which helps the researchers in examining malware in a convenient way[16]. We have used some tools from REMnux for static analysis. The tools we have used for static analysis take parameters which manipulate settings and deal with input and result of analysis. In this paper, we are focusing on file packing. Hence, only the appropriate tools required for our test and investigation are described here. We have used popular packing-tool known as UPX, the ultimate packer for executable files [17] to check if a sample is packed and to unpack packed samples. Michael Sikorski et. al. stated in their book that UPX is popularly used by many virus writers for packing purpose [18]. Bytehist is a tool which generates histograms for all kinds of files with more emphasis on windows based portable executable[19]. Histograms generated from analysis specify frequencies of ASCII characters in hexadecimal values in the range 00 to FF. Initially, the primary histogram is created for the file and then for PE files. Similarly, secondary histograms are created for different sections of the file[20]. From these histograms, we can observe irregular and abnormal byte distribution in different sections of a file. If the byte codes are evenly distributed, then the files are possibly encrypted, packed or compressed. There is another similar tool known as Charcount which produces the output in a text format for the entire file [21]. Besides these tools, we have also used the services provided by VirusTotal.It analyses a file or an Uniform Resource Locator and gives a database of formerly analyzed files. The analysis report given by VirusTotal contains different types of information which are essential for malware research. It is important to mention that, the scan results from many anti-virus solutions have helped us a lot to verify our results. To carry out experiment, we have collected programs intended for administering a system e.g. to remove non-permanent file, removing damaged link, searching redundant file, task management file, optimizing disk, file encryption programs and so on. Keeping in mind our assumption stated earlier, the aim was to inspect benign samples to observe actions, properties which very often performed by malicious samples. We have selected 100 samples to carry out our experiment.

IV. Results of Experiment

Initially we have analyzed installed programs as the part of our experiment. It is obvious for malicious software to organize a way into the system at the time of installation, which is later utilized to perform malevolent action. Hence, behaviour of samples during installation is interesting for analysis.

A. Ultimate Packer for Executables(UPX) and similar packers

The collected samples are analyzed using UPX , which has detected four samples are packed using the tool. We have confirmed this result by performing a test in VirusTotal. VirusTotal uses different packers including UPX to detect if the sample is packed. The result from virusTotal is shown in figure 1(a). In a few of the samples the packers are appended to the file, a few are entrenched into the file and others are appended as well as embedded. Multi-level packing is used for some of the samples using numerous packers all together. For example, it has detected one sample as packed with RAR and after unpacking the sample, it is detected as packed using ASPack. Most frequently used packers include the following: INNO, CAB, Unicode and UTF-82.

INNO is another packer tool used to create installation programs [22]. When a program is packed using this packer, it can only be unpacked after running it. This helps the malware to hide itself during static analysis. The result from VirusTotal

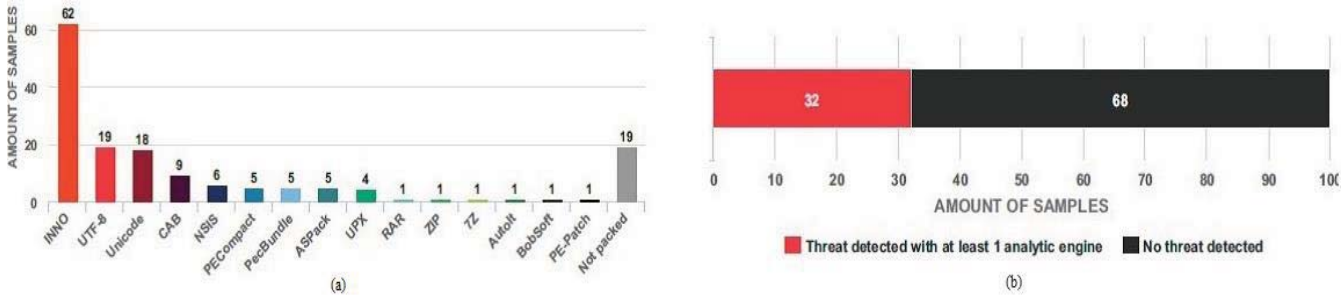

Fig. 1. (a) VirusTotal result on samples packed by different packers. (b) Result of malicious identification in the analyzed samples by virusTotal.

shows that most of the samples are packed with INNO and a few are packed with other packers and multiple packers whereas others are identified as not packed. Around one third of the total verified samples were identified as malicious by minimum 1 malware-detection module used by VirusTotal. Though, there is a possibility that a few of these detections are false positives (FP). The result of VirusTotal which depicts the samples are classified into two categories such as threat and no threat is shown in figure 1 (b).

B. Character Distribution Using Charcount Tool

Charcount tool allows us to have a clear view of distribution of ASCII characters among the samples. Usually the ASCII characters are unevenly distributed in unpacked and uncompressed applications. Charcount tool shows several fluctuations of these samples as depicted in figure 2(a). The figure shows more fluctuations for a few samples which are normal and unpacked files whereas there are few other samples which are considered to be packed and compressed because the graph shows less fluctuation for those samples. On the other hand, marginal or minimal fluctuations are considered as abnormalities and which is considered as being modified or packed with a packer. We have categorized the Charcount results into different categories. We have calculated gap between largest and smallest frequency of characters in a sample and this gap is compared with mean frequency of all characters, except for the hexadecimals 0x00 and 0xFF, because these characters possess irregular frequency in almost all files and hence these are ignored by us thinking that it could lead to inaccuracy in our results. The categories are defined as follows:

cat I : M - m $< (0.1 \times A)$,
$catII : M - m \geq (0.1 \times A)$,
$catIII : M - m \geq (0.2 \times A)$,
$catIV : M - m \geq (0.3 \times A)$,
$catV : M - m \geq (0.4 \times A)$.

where, 'M' denotes the Maximum frequency among characters in a file except those two characters. 'm' is the minimum number of incidences in a file except those two characters. and 'A' is the Average of all incidents of all characters except those two characters. Category-wise samples is given in Figure 2(b). As far as packing is concerned, samples belong to Cat-I and Cat-II look suspicious.

Character distribution in each category is depicted in figure 2(c). A sample from Cat-I has comparatively less fluctuations in occurrence of characters, which are usual for samples that have been packed or compressed. A Cat-II sample is analogous to the preceding one and though it displays numerous extremes, it is possibly also altered. Files belong to Cat-III and IV have observable variations. Cat-V is usually for unchanged programs, which are denoted by several and important variations in occurrence of characters.

C. Distribution of characters in different sections of PE file

In order to have a deep view into a portable executable, we can consider its sections to reveal its internal features. Bytehist can be used to find the compressed sections of a portable executable. This tool generates image files as the result of analysis. The resulted image files contain histograms and usage of ASCII characters. The first histogram is generated for the entire file. In the event of portable executable files, the histograms are corresponding to its different sections. Usually Portable Executable file contains different sections such as; .text,.rdata .bss, .idata, .data and .reloc. If any other content is detected by bytehist beyond the last section, then it is named as .rest. Results of analysis using Bytehist revealed that only twelve out of all the tested files has no content beyond the last normal section (i.e. there is no .rest). The other tested files have .rest section with different sizes. Percentage of the original programs size contained in the section is indicated by histograms of the sections. There are a few tested files which contain the .rest section with size less than half of the original program-size. Percentage of files in the rest section is shown in Figure 3 (a).

In half of the samples, the .rest section contains about 90% of the overall programs size. This infrequent scattering of bytes among sections, or beyond normal sections, can be considered abnormal and hence suspicious. Moreover, if .rest section also exhibits an even scattering, it indicates presence of unwanted concealed property in the sample. We have found merely three samples contain data in .rest section with irregular distribution. Figure 3(b) displays the histograms of .rest segment from two dissimilar test files. The left side of the histogram indicates that 99 percent of the program was found in regular parts and the right side comprises only 33 percent of the total size of

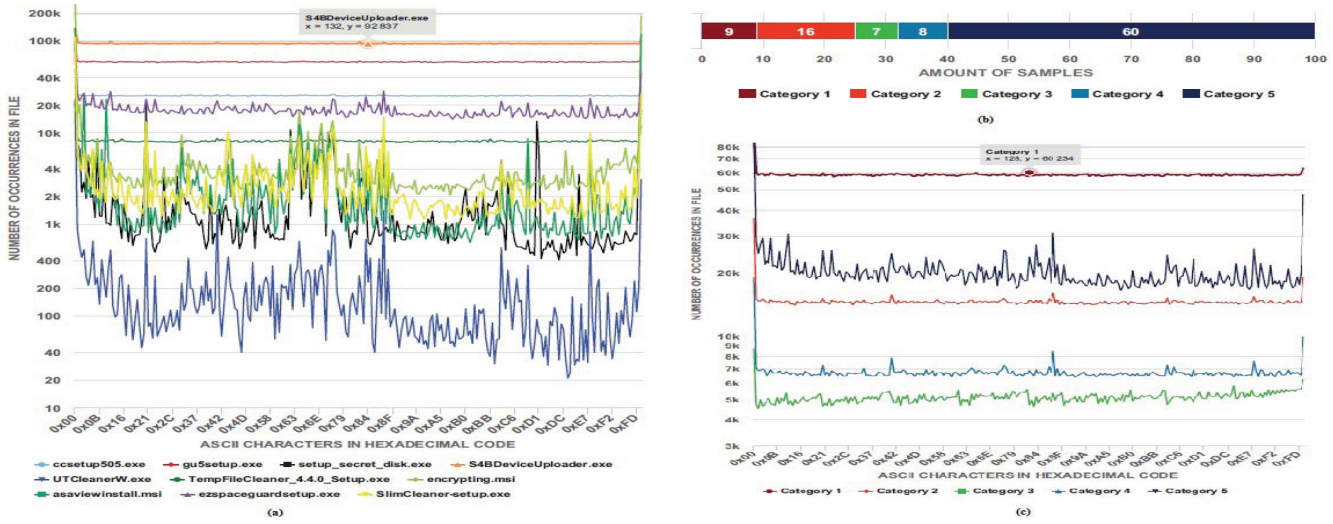

Fig. 2. (a) Charcount result on Samples: Showing packed as well as unpacked files with less and more fluctuations. (b) Every analyzed sample belongs to any category from I to V based on specific relations. (c) Category-wise character distribution.

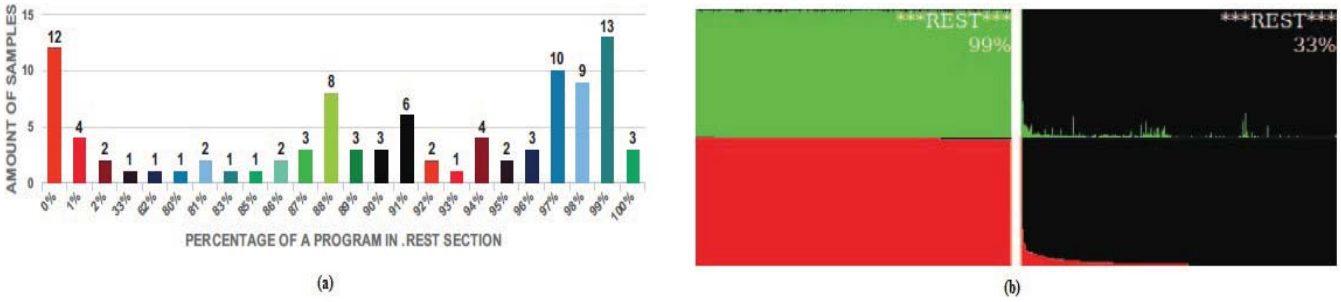

Fig. 3. (a) Bytehist tool analysis on Percentages of a program in .rest section and number of samples with .rest section. (b) Histograms of .rest section from two different samples

the program and shows irregular scattering of characters as compared to the files present in the left.

We predict that, the major part of the programs in the .rest section may be caused by tools like INNO which was used to build installation programs. But more research is needed to verify validity of our prediction. Results of our research explained in this paper show that packed files are not always malware because benign files are also sometimes packed for security. VirusTotal reports are enough to reveal that; though examined files are not detected as malware, they are not completely benign according to the malware detection engines used in VirusTotal. Based on the above results, we understood that; defining whether a sample is malicious or harmful is a challenging task.

V. Conclusion

We have observed that, sometimes benign files may also contain similar operations as that of malicious files which allows the anti-malware system to erroneously detect benign file as malware. When a file performs the same task for which it was created, then no suspicion arise. But, malware writers can misuse users trust and disguise malicious activities behind anticipated functionality. Same kind of problem arises when executable files are packed or encrypted or compressed. Though malware creators extensively use packing technique, they are also useful for protecting integrity of legitimate file which makes it difficult to identify bad and good objectives behind packers usage. As per the results of our experiment the malicious operations are positively detected in multiple samples under consideration. Analysis of 100 samples has supported our assumption positively. As an extension to this work, we will identify and consolidate the changes occurred to the system after the analyzed programs are installed on different testing environments. In our future work, we will emphasize on dynamic analysis because, to discover the hidden behaviour of a packed sample it is essential to execute it. Packing information cannot be overlooked while describing malware or behaviour signature of malware but they should have little significance and should not affect detection in a negative manner. In the next work, we will use different machine learning and data mining algorithms as an attempt to classify the samples either as malware or benign with more accuracy.

REFERENCES

[1] https://www.av-test.org/en/statistics/malware/

[2] P.V.Shijo, A.Salim, Integrated Static and Dynamic Analysis for Malware Detection, the third ICoICT Conference,(2015), pp:804-811.

[3] http://www.cisco.com/c/dam/m/digital/en_us/Cisco_Annual_Cybersecurity_Report _2017.pdf.

[4] Liu, Wu Ren, Ping Liu, Ke Duan, Haixin., "Behavior-Based Malware Analysis and Detection", Proceedings of Int. Workshop on Complexity data mining, 10.1109/IWCDM.2011.17, (2011), pp:39-42.

[5] Michael Bailey,Jon Oberheide,Jon AndersenZ, Morley Mao,Farnam Jahanian and Jose Nazario, Automated classification analysis of internet malware, In: Kruegel C., Lippmann R., Clark A. (eds) Recent Advances in Intrusion Detection. RAID 2007, Lecture Notes in Computer Science, vol 4637. Springer, Berlin, Heidelberg (2007), pp:178-197.

[6] Konrad Rieck, Thorsten Holz,Carsten Willems,Patrick Dusse and Pavel Laskov, Learning and classification of malware behavior, in DIMVA-2008, Springer Berlin Heidelberg, vol. 5137, (2008), pp:108-125.

[7] G. Wagener, R. State, and A. Dulaunoy, Malware behaviour analysis, Journal in Computer Virology, vol. 4, no. 4 (2008), pp.279-287.

[8] U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel, A view on current malware behaviors, in Proceedings of the 2Nd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More, ser. LEET09. Berkeley, CA, USA: USENIX Association, (2009), p.8.

[9] K. Asmitha and P. Vinod, A machine learning approach for linux malware detection, in International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), (2014), pp:825-830.

[10] M. Zolotukhin and T. Hamalainen, Detection of zero-day malware based on the analysis of opcode sequences, in IEEE 11th Consumer Communications and Networking Conference (2014), pp:386-391.

[11] S. Josse, Secure and advanced unpacking using computer emulation, Journal in Computer Virology, vol. 3, no. 3 (2007), pp:221-236.

[12] A.Singh and A. Lakhotia, Game-theoretic design of an information exchange model for detecting packed malware, in 6th International Conference on Malicious and Unwanted Software (MALWARE),(2011) PP:1-7.

[13] S. Association., IEEE-SA - Industry connections security group (ICGS) (2015) [Online]. Available: http://standards.ieee.org /develop/indconn/icsg/index.html.

[14] T.-Y. Wang and C.-H. Wu, Detection of packed executables using support vector machines, in International Conference on Machine Learning and Cybernetics (ICMLC), 2011, vol. 2 (2011), pp:717-722.

[15] C. Linn and S. Debray, Obfuscation of executable code to improve resistance to static disassembly, in Proceedings of the 10th ACM Conference on Computer and Communications Security, ser. CCS 03. New York, NY, USA: ACM (2003), pp:290-299.

[16] https://remnux.org/

[17] Victor Marak, "Windows Malware Analysis Essentials", PACKT Publishing. p. 188. ISBN:978-1-78528-151-8.

[18] Michael Sikorski, Andrew Honig,"Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software", No Starch Press,p.474, Feb 2012,ISBN: 978-1-59327-290-6.

[19] Cameron H. Malin, Eoghan Casey, James M.,"Malware Forensics Field Guide for Windows Systems: Digital Forensics Field Guides",Elsevier,p.246, May-2012, ISBN:978-1-59749-472-4.

[20] https:C.Wojner. Bytehist - cert.at (2015) [Online]. Available: https://www.cert.at/downloads /software/bytehist_en.html

[21] L. Auriemma. Mytoolz - charcount. [Online]. Available: http://aluigi.altervista.org /mytoolz/charcount.zip

[22] J. Russell., Inno setup. (2012) [Online]. Available: http://innounp.sourceforge.net/