THE PROBLEM OF MALWARE PACKING AND ITS OCCURRENCE IN HARMLESS SOFTWARE

Jana ŠŤASTNÁ, Martin TOMÁŠEK
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic,
E-mail: {jana.stastna, martin.tomasek}@tuke.sk

ABSTRACT

Analysis of software behaviour and its other properties is largely used as a method for uncovering malicious features in software, especially in cases of unknown malware. Traditional malware signatures can be circumvented, e.g. by obfuscation, therefore in our endeavour to formulate malware behavioural signatures we study behaviour and various properties detectable in malware. However, in this article we present different point of view on this issue. In our experiments we analyse a set of freely available software that is harmless and compare data extracted from analysis with malicious programs. In this article we focus on results related to so-called packing and show that this typical malware feature may be present in harmless software as well.

Keywords: *malicious software, non-malicious software, packing, bytes distribution, malware indicators*

1. INTRODUCTION

As malware detection techniques advanced, malware writers improved concealing harmful code with encryption, mutation and packing. Although analysts know about these techniques, they still seem efficient in defeating detection systems based on malware signatures. Since traditional signatures are constructed from syntactical form of malicious programs and, as Moser et al. mention in their work [1], this form is relatively easy to modify while preserving semantics of the program, malware creators have opportunity to create large number of malware variants which will not be immediately detected. Methods of malware obfuscation, e.g. encryption, dead-code insertion, register reassignment, subroutine reordering, instruction substitution, code transposition, which are described in [2], complicate static analysis and detection of malware, and in some cases, according to Moser et al. [1] and Landi [3], make it even impossible.

Malware signatures have still very important role in malware detection, although their effectiveness on malicious samples concealed by techniques that alter syntactic form of a program is questionable. What is more, with rapidly growing number of new malware samples the extraction of signatures requires a lot of precious time. Griffin et al. addressed this problem and presented a system for automated generation of malware signatures [4]. An interesting part of their work describes features which they analysed in malicious programs. Concerning syntactic form of a program authors mention patterns emerging in operational code which may represent precursors of non-standard or suspicious behaviour of the program:

- Constant values like IP addresses, email addresses,
- access to memory with unusual offset,
- local function calls, non-library function calls, context of a function call and used parameters,
- suspicious mathematical operations which may indicate obfuscation.

These patterns are used for refining potential malware signatures through, as they call it, *code interestingness check*. In our research they served as an inspiration for searching malicious features in programs and looking for patterns which could be employed as indicators of malicious intentions.

The specific feature addressed in this article is known among malware researchers as *packing*. Packing employs compression for reduction of programs size, combined with encryption to obstruct programs reverse-engineering, signature-based detection and static analysis [5]. Unfortunately, software packers are popular among malware writers [6].

We analysed a set of harmless programs available on the internet for free and in our observations we aimed at features related to packing. Our goal was to figure out whether typical malware feature like packing, which is considered as an indicator of suspicious or even dangerous files, is a sufficient malware marker on its own or it is not, since we assume that it occurs in harmless software as well. Our first results of the research were published in conference articles [7], [8], and this article extends presented results with another findings concerning the issue.

In summary, we try to answer the following questions:

1. Does harmless software present typically malicious feature, specifically packing, just like malicious software?

2. Can we distinguish malicious usage of packer from the harmless one by analysing several program's features on the syntactic level?

2. METHODS

As a very interesting approach to malicious software analysis we consider a system called REMnux [9], which is a collection of tools for static and dynamic malware analysis, installed on Linux distribution derived from Ubuntu. The large amount and diversity of utilities and analytic programs are a major advantage of REMnux. We used several tools from REMnux distribution in our experiments which allowed static analysis of our prepared set of programs.

Even if dynamic analysis compared with static analysis indicates smaller issues with malware obfuscation, static analysis allows to detect malicious features which seem to occur randomly during program execution or in specific execution environment. What is more, nature of tools purposed for static analysis enables us to automate the analytic process for large amount of experimental samples. This approach is therefore less time consuming in comparison with dynamic analysis, which requires execution of each sample for adequate amount of time.

2.1. Experimental tools

Tools for static analysis of Portable Executable files[1] used in the experiment have a form of a terminal application, which accepts arguments that modify settings and set input and output of analysis. This allowed us to create helper programs for automation of large number of samples' analysis.

From various analytic tools we chose for the experiment the following:

UPX is a file packing tool for executable files [10]. It allows us to check whether a tested file is packed or not and to unpack files that are packed. As a book by Davis et al. states [11], numerous computer viruses use specifically UPX packer, therefore detection of UPX packer in analysed sample arises suspicion.

Bytehist and **Charcount** examine byte usage for all types of files. Charcount tool outputs number of occurrences of program's byte-codes in a range of hexadecimal values 0x00 - 0xFF [12], Bytehist outputs histograms of these occurrences [13]. Histograms of PE files are generated with respect to sections of the program. The main histogram is produced for the whole program and sub-histograms are generated corresponding to program's sections. These histograms reveal anomalies in distribution of bytes for each section - uniform distribution of byte-codes indicates that the file is compressed, encrypted or packed. Uniformity in distribution is normal for ordinary compressed archives like .zip or .rar, however, in case of PE files section it suggests packing for the purpose of code obfuscation.

Densityscout determines density of an analysed file. Description of this tool explains that malware and goodware usually have contrasting densities. While goodware tends to have density higher than 1000 (in CHI mode, for files bigger than 100kB), suspicious packed files gain density value below 100 [14]. No explanation is mentioned for values in the interval <100, 1000>, so we expect that they correspond to ambiguous cases.

Virustotal is an online malware analysis service. To evaluate properties extracted from the tested set of programs we needed to collect frequent characteristics of malicious behaviour. For security reasons and to obtain desired data within a short time we decided to perform the analysis of malware not by ourselves but to use analytic results from a reliable and independent source. Many online services for malware analysis provide access to database of already analysed samples via web interface. We examined several of them, considering amount of analysed samples and quality of data they offered. Huge amount of files from thousands of users - various sources - were tested in this manner. We needed as many types of data as possible to understand characteristic malware behaviour, also for purposes outside of the scope of this experiment. Analysis offered by VirusTotal [15] met our requirements. Reports generated after analysis contain various types of information. Worth mentioning are e.g. scan results form over 50 anti-virus solutions which in our case helped to check whether experimental files are truly harmless. Reports state also detection of packers, information from PE header, PE sections, dll imports and exports and in case of PE file analysis, behavioural information describe operations performed by dynamically analysed program, e.g. opening, reading, editing or deleting a file, launching new processes, creating mutexes, executing a shell command.

2.2. The experimental set of programs

The set of experimental samples consisted of freely available utility software, e.g. for deleting temporary files, broken links removal, duplicate files search, tasks management, memory optimization or personal files encryption. We targeted this specific group of software because of operations that these programs are designated to perform. We assume that software which legitimately accesses registry entries, processes, file system, etc., may be a promising target for malware writers which create malicious imitations of the original harmless software. With this in mind, it would be interesting to compare analysis results of harmless system maintenance tools and their fake malicious counterparts. Our experimental set consisted of 100 harmless programs and 100 samples which were verified as malicious. We did not succeed in obtaining malicious versions of all harmless samples, but regardless of that we preserved domain of selected malicious samples in system maintenance.

3. RESULTS

The experiment was performed in two stages: First we analysed the harmless set of programs and evaluated results. After that we proceeded with examining features of malicious samples.

3.1. Harmless samples

Analysis revealed that among 100 samples only 19 were not packed by any known packing tool. The other 81 samples were packed by 15 different packers, e.g. UPX which is preferred by numerous computer viruses [11], PECompact, PecBundle, INNO, ASPack, PE-Patch. Many samples were modified with several packers simultaneously - multiple layers of packing were applied.

Detecting a packer which is often employed by malware does not necessarily mean that the sample is malicious. A closer look at distribution of bytes in analysed samples uncovered further differences between samples. Distribution

[1] Portable Executable files are often shortly denoted as PE files.

of bytes in basic applications that are not packed, compressed or encrypted, is typically uneven, it exhibits several fluctuations which are well visible. On the contrary, minimal or no fluctuations (uniformly distributed bytes) are considered as anomalies and suggest that the sample was modified with a packer, encryption or compression engine. Fig. 1 contains graphs of bytes distribution from several samples. Specifically samples 1, 2, 4 and 6 represent typical packed programs.

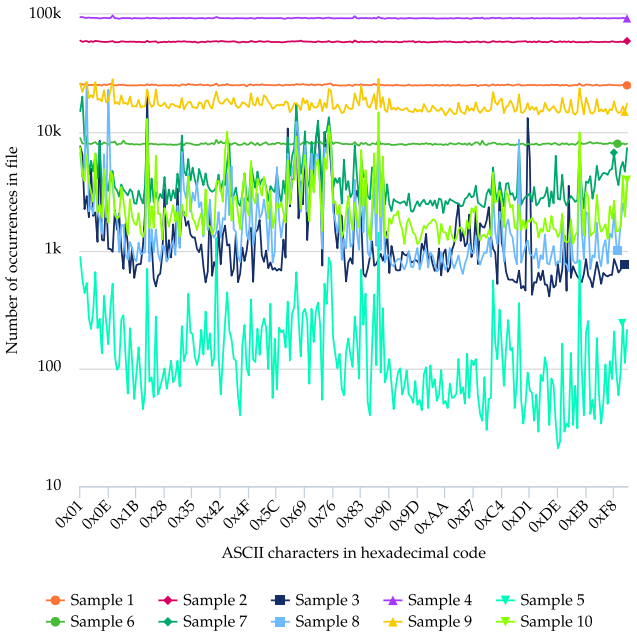

Fig. 1 Distribution of bytes for 10 selected samples from analysed set. They illustrate difference between uniform (in this context anomalous) bytes distribution and distribution with high fluctuations.

Distributions of bytes obtained from samples by analytic tool Charcount were divided into 5 categories[2] based on intensity of fluctuations in distribution. Categories 1 and 2 represent samples with uniform or almost uniform bytes distribution and are related to usage of packers. On the other hand, category 5 represents samples with numerous significant fluctuations typical for normal programs. Although only 19 samples were not packed according to the first analysis above, 60 samples indicate bytes distribution of normal (not packed) programs (Fig. 2).

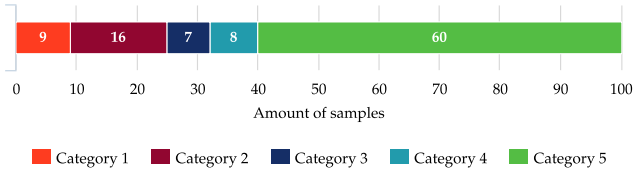

Fig. 2 Number of samples in categories of bytes distribution.

More detailed look into a PE file can uncover which section of the file is packed, if any. Analytic tool Bytehist creates several images with histograms of bytes usages. The first corresponds to the sample as a whole file. In case of a PE file the other histograms correspond to its sections, typically `.text`, `.bss`, `.rdata`, `.data`, `.idata`, `.reloc`. If Bytehist detects some content even after the last section, it is presented as a section `.rest`. Examination of results from Bytehist showed that only 12 of 100 samples contained no data after the last ordinary section. For the rest of samples, the amount of bytes in `.rest` varied. Histograms of sections also indicate percentage of the original program's size contained in the section. Only 19 samples contained `.rest` section under 50% of the program's size and 90% or more of program's bytes was present in more than a half of samples. This unusual distribution of bytes across sections, or rather behind common sections, is suspicious. Furthermore, if section `.rest` exhibits uniform distribution, it suggests some hidden feature in the sample. Only 3 programs indicated data in section `.rest` with uneven distribution.

An interesting observation we made recently is that samples which had majority of program's contents in the `.rest` section also showed abnormally uniform distribution of bytes in this section (Fig. 3), presented as *entropy* by analytic service VirusTotal.

Fig. 3 Entropy of samples' `.rest` section compared with percentage of file in this section. Entropy is converted to percentage range - 100% is equivalent to maximal entropy = 8.

Concerning data obtained from Densityscout tool, 21 harmless samples resulted with density typical for obfuscated files, i.e. below 100. Surprising is, that 6 samples from mentioned 21 were not detected as packed, not even by one packer. Number of detected packers is well visible in Fig. 4 where density is set to transparent color for better readability. On the other hand, from 27 samples having density of typically normal, not obfuscated files, above density 1000, 14 were detected as packed with at least 1 packer (Fig. 5). We deduce that these contradicting results are

[2]Details about categories and criteria of sample's placement into particular category are described in the article [7].

caused by various degrees of protection provided by packers. While low level of protection may only compress size of the file, high level of protection affects file's density.

Calculation of entropy and its usage in malware research is described e.g. by Mueller [16], Desnos and Erra [17]. Entropy reaches values from interval <0, 8>and the higher is the value, the more randomly are bytes distributed in the file, like when its syntactic structure is modified by encryption and compression.

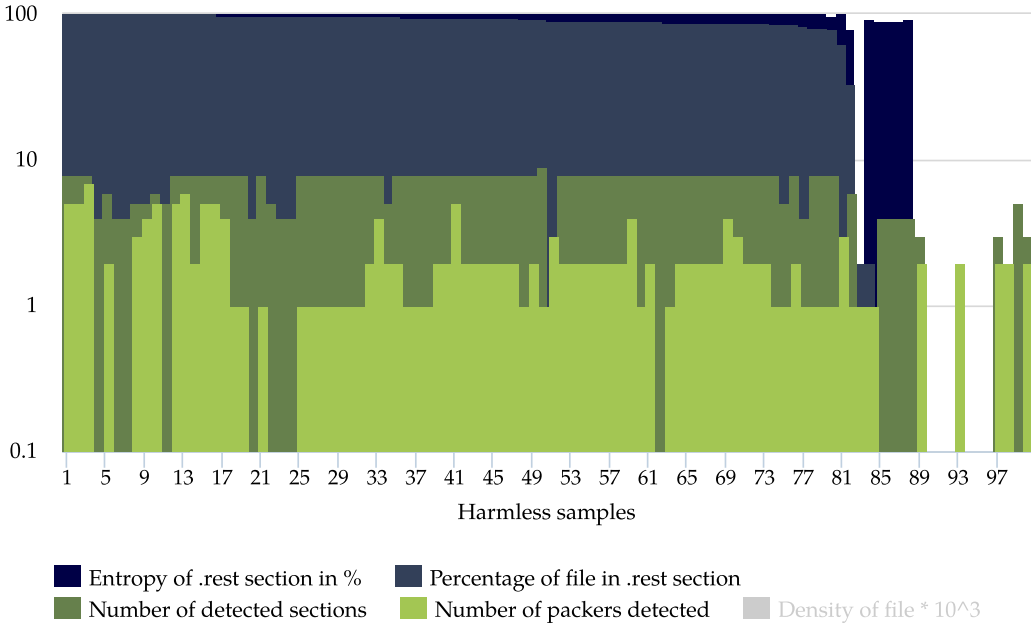

Fig. 4 Summary of results obtained from analysis of 100 harmless programs. Entropy is converted to percentage range - 100% is equivalent to maximal entropy = 8. Density is set to transparent color for better readability.

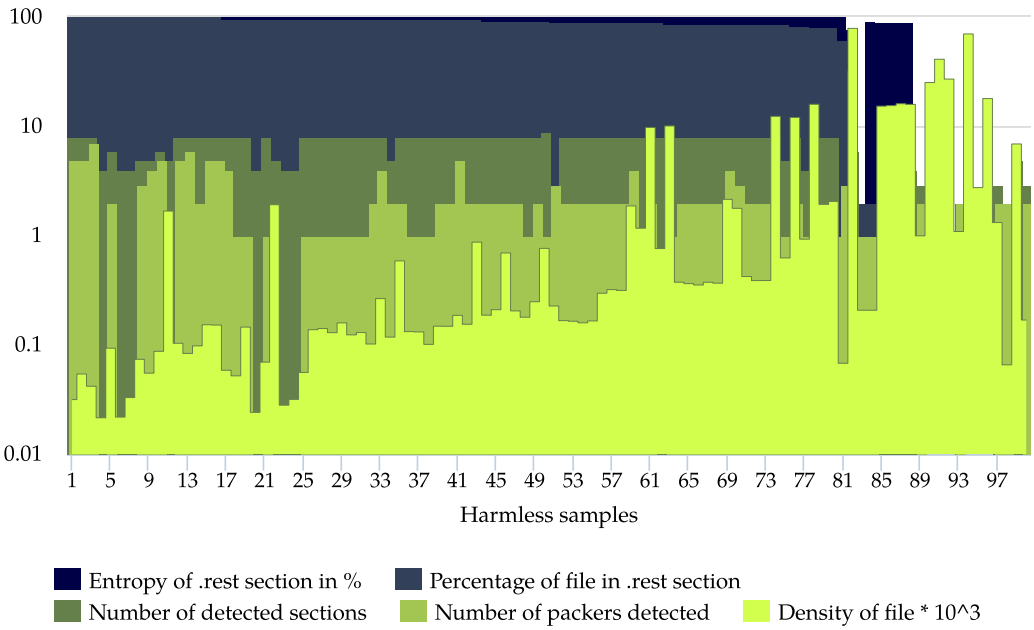

Fig. 5 Complete summary of results obtained from analysis of 100 harmless programs. Original density values were divided by 10^3 because of high variance of values.

3.2. Malicious samples

When comparing results of malicious samples' analysis with data obtained from harmless samples so far [7], we can state that the most used packer among both experimental sets remains the same - INNO. What changed dramatically is the amount of samples detected as packed - only 36 were revealed to be packed by at least one packer (Fig. 6).

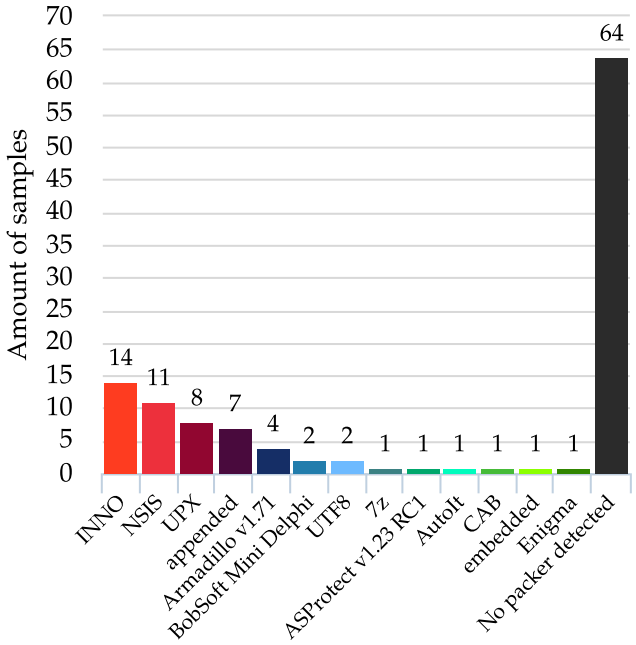

Fig. 6 Types and numbers of packers detected in malicious samples.

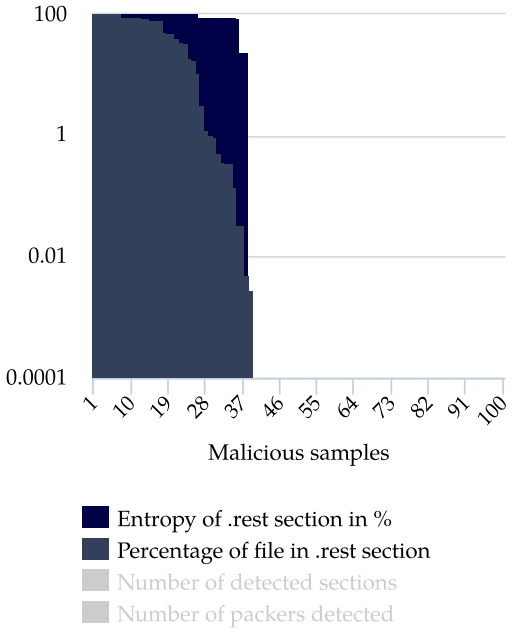

Fig. 7 Entropy of malicious sample's .rest section compared with percentage of file in this section. Entropy is converted to percentage range - 100% is equivalent to maximal entropy = 8.

Concerning data after the last program's section, this feature manifested in much fewer malicious samples than in the set of harmless samples. Still, the relation between percentage of file in the `.rest` section and entropy of this section seems to be retained (Fig. 7). From summarised results (Fig. 8) it seems that malicious usage of packers is harder to detect even for specialised tools and analytic services. Since many packer detectors, like PEiD used by VirusTotal service, employ signatures to recognise known packers, custom malware packers remain unnoticed.

Also `.rest` section, which hides potentially malicious parts of a program, occurs less in samples which are modified by undetectable packer or do not employ this kind of concealing at all. Despite these results several anomalies emerged. One sample was packed repeatedly by the same packer - in 46 layers. Another 2 samples which were reportedly not packed had section names corresponding to UPX packer. Also unnamed or strangely named sections with high entropy appeared in several samples.

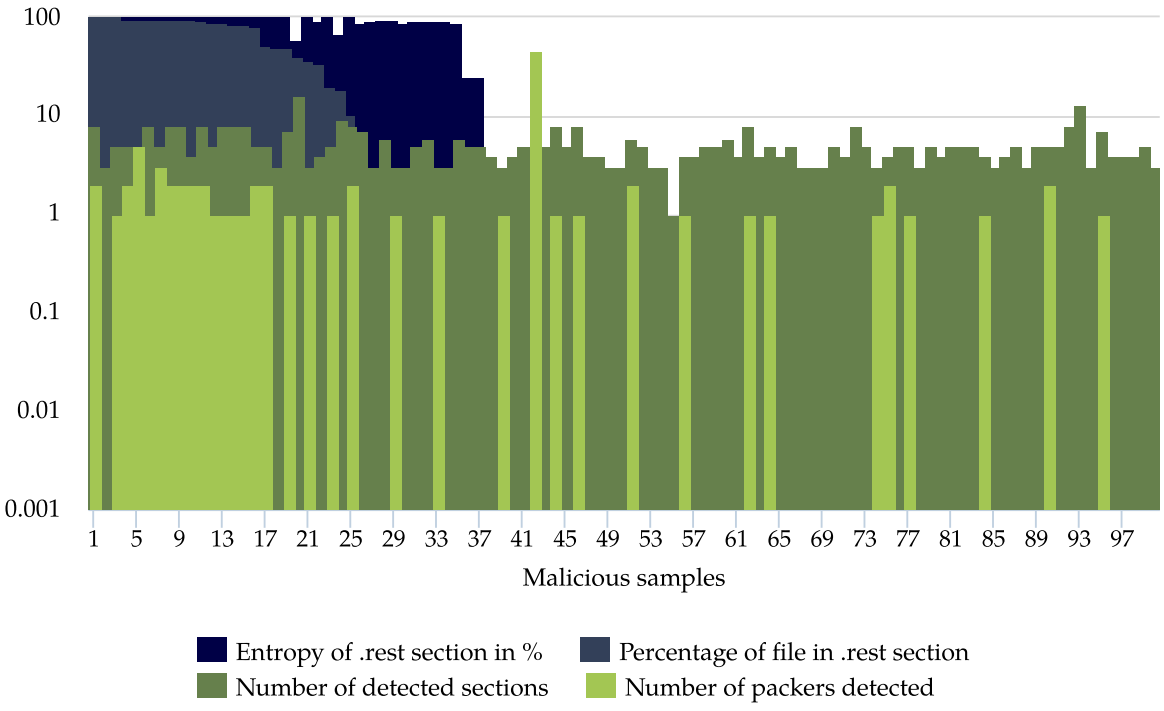

Fig. 8 Complete summary of results obtained from analysis of 100 malicious programs.

4. CHANGING TRENDS IN PACKING

Shadowserver provides useful statistics aimed at, but not limited to, malware packers [18]. According to their packer statistics in April for last 60 days[3] the most prevalent packer among malware was *Allaple* with 3 867 583 occurrences, followed by *UPX All_Versions* with 312 444 samples and *ASPack v2.12* with 161 888 packed samples (Fig. 9).

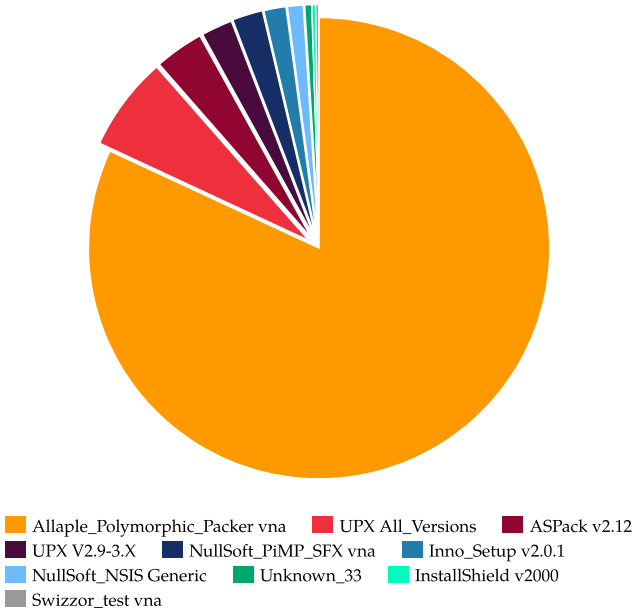

Fig. 9 Shadowserver packers statistics - 10 mostly used packers in April 2016.

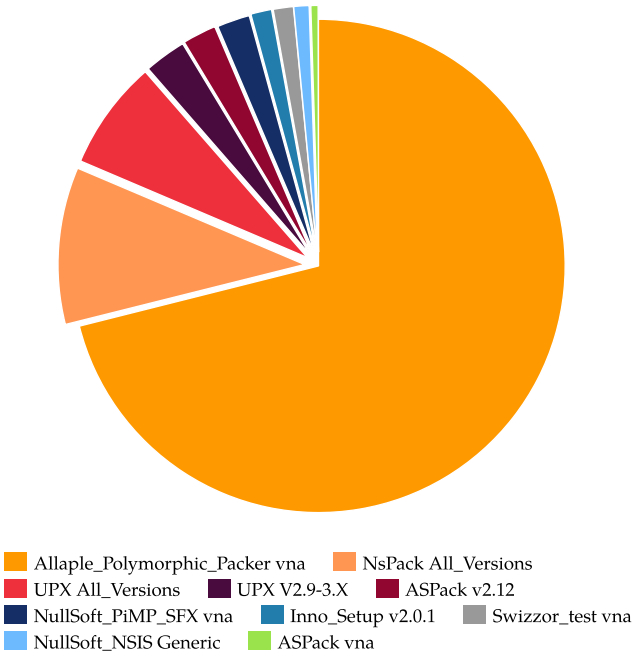

Fig. 10 Shadowserver packers statistics - 10 mostly used packers in June 2016.

[3]Data obtained 30/4/2016.
[4]Data obtained 27/6/2016.

Two months later packers' trends changed and according to Shadowserver statistics in June[4] another packer made it into top 10 chart - *NsPack* as the second most popular packer with 1 075 382 occurrences (Fig. 10). We can see that *ASPack v2.12* and *UPX V2.9-3.X* switched positions in the chart and *Swizzor_test vna* gained popularity.

With these bi-monthly statistics we can monitor current trends in packers' usage. It seems that malware writers reconsidered their selection, as relying to *Allaple* became too obvious as a choice for hiding malicious code. We may expect that malware creators will change their packing strategies from time to time, but since many new malicious samples are only updated versions of existing threads, also their hiding strategies are not completely random and follow certain trends.

5. CONCLUSION

A packer is the first barrier that malware analyst needs to overcome in order to ascertain whether a program performs any malicious actions.

Concerning questions presented in the Introduction, we conclude that harmless and malicious software evidently share suspicious features like packing, program's code in unusual `.rest` section and uniform distribution of bytes. Although harmless software does indeed present packing, the symptoms related to it are less concealed in comparison with malicious software. From results we obtained it seems that malicious usage of a packer is not easily recognisable from syntactic form of a program. This leads us to a conclusion that assumptions about typical malware features are not always reliable. In case of packing, it should not be considered an indicator of maliciousness yet, since it is linked with both malicious and harmless software. Unless the purpose of packing the sample is known, solid conclusions can not be made. If analytic tools which can detect unknown custom packers are not available, we can not state that some sample is (not) packed for sure.

To distinguish malware from harmless software, we need to get underneath the protective packed layer of programs. As several anomalies suggested, which we observed when processing malware analysis data, number of program's sections, their names and entropy may indicate potential use of unknown custom packer or another kind of malicious obfuscation. Since packers are purposely designed to inhibit reverse engineering and techniques of static analysis, we need to execute the analysed program in a controlled environment in order to extract unpacked program's instructions.

According to research of Jacob et al. [19], detection of packed malware solely by static analysis is possible in case when analysed samples are similar, re-packed, re-encrypted variants of already recognised malware. However, to cover samples which do not match the case of Jacob's et al. work, combination of static and dynamic analysis is necessary.

ACKNOWLEDGEMENT

This work has been supported by Grant No. FEI-2015-18: Coalgebraic models of component systems.

REFERENCES

[1] MOSER, A. et al.: *Limits of Static Analysis for Malware Detection*, Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), 421–430, IEEE (2007).

[2] YOU, I. – YIM, K.: Malware Obfuscation Techniques: A Brief Survey. 2010 International Conference on Broadband, Wireless Computing, Communication and Applications, 297–300, IEEE (2010).

[3] LANDI, W.: *Undecidability of static analysis*, ACM Lett. Program. Lang. Syst. **1**, No. 4 (1992) 323–337.

[4] GRIFFIN, K. et al.: *Automatic Generation of String Signatures for Malware Detection*, Recent Advances in Intrusion Detection, ser. LNCS 5758, 101–120, Springer Berlin Heidelberg (2009).

[5] JOSSE, S.: *Secure and advanced unpacking using computer emulation*, Journal in Computer Virology **3**, No. 3 (2007) 221–236.

[6] GUO, F. – FERRIE, P. – CHIUEH, T.:*A study of the packer problem and its solutions*, Recent Advances in Intrusion Detection, ser. LNCS 5230, 98–115, Springer Berlin Heidelberg (2008).

[7] ŠŤASTNÁ, J. – TOMÁŠEK, M.: *Exploring malware behaviour for improvement of malware signatures*, 2015 IEEE 13th International Scientific Conference on Informatics, 275–280, IEEE (2015).

[8] ŠŤASTNÁ, J.: *Searching for Malware Markers - Packing and Harmless Software*, 16th Scientific Conference of Young Researchers, 232–233 (2016).

[9] ZELTSER, L.: REMnux: A free Linux Toolkit for Reverse-Engineering and Analyzing Malware (2016). `https://remnux.org/`

[10] OBERHUMER, M.– MOLNÁR, L. – REISER, J. F.: Ultimate Packer for eXecutables (2013). `http://upx.sourceforge.net/`.

[11] DAVIS, M. – BODMER, S. – LeMASTERS, A.: Hacking exposed malware & rootkits, McGraw Hill, New York (2010).

[12] AURIEMMA, L.: MYTOOLZ - Charcount. `http://aluigi.altervista.org/mytoolz/charcount.zip`

[13] WOJNER, C.: Bytehist - CERT.at (2015). `https://www.cert.at/downloads/software/bytehist_en.html`

[14] WOJNER, C.: DensityScout - CERT.at (2013). `https://www.cert.at/downloads/software/densityscout_en.html`

[15] VIRUSTOTAL: Advanced features & tools (2016). `https://www.virustotal.com/en/documentation/`

[16] MUELLER, L.: File Entropy explained (2013). `http://www.forensickb.com/2013/03/file-entropy-explained.html`

[17] DESNOS, A. – ERRA, R.: *Descriptional Entropy: Application to Security Software Analysis*, Advanced Infocomm Technology: 5th IEEE International Conference, ICAIT 2012, 225–230, Springer Berlin Heidelberg (2013).

[18] Shadowserver Foundation: Packer Statistics (2016). `https://www.shadowserver.org/wiki/pmwiki.php/Stats/PackerStatistics`

[19] GRÉGOIRE, J. et al.:*A Static, Packer-Agnostic Filter to Detect Similar Malware Samples*, Detection of Intrusions and Malware, and Vulnerability Assessment, ser. LNCS 7591, 102–122, Springer Berlin Heidelberg (2013).

Received July 4, 2016, accepted September 13, 2016

BIOGRAPHIES

Jana Šťastná is a PhD student at the department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University of Košice. In 2014 she graduated (MSc) with distinction in the field of Informatics. Her scientific research is centred on malicious software behaviour.

Martin Tomášek received the master degree in computer science in 1998 and PhD degree in software and information systems in 2005 both at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice, Slovakia. Currently he is an associate professor at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice, Slovakia. His research interests include multi-agent systems, distributed systems, component-based systems, and concurrency theory.