Proceedings of the Federated Conference on Computer Science and Information Systems pp. 569–572
DOI: 10.15439/2017F354
ISSN 2300-5963 ACSIS, Vol. 11

High-Level Malware Behavioural Patterns: Extractability Evaluation

Jana Šťastná, Martin Tomášek
Department of Computers and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovakia
Email: {jana.stastna, martin.tomasek}@tuke.sk

***Abstract*—Many promising malware research projects focus on malware behaviour analysis, however, in the end they tend to build new detection systems and stick to measuring detection ratios. Our approach focuses on malware behavioural analysis for defining (characterising) malicious software on rather high level of abstraction, in order to break the endless cycle of evolving malware and malware analysts trying to catch up on new threats. As our research outlines, even such high-level behavioural information as numbers of occurrences of some behavioural events, can be successfully extracted from program samples and interpreted for extraction of repeating behavioural patterns. While this may seem simple at the first glance, there are plenty variables entering the process of behavioural data acquisition and pattern extraction.**

I. Introduction

A TRANSITION from syntactic to semantic view on malicious software leads the research in several last years. The reason for this change is quite simple: Traditional detection signatures, built upon fragments of executable code extracted from malicious samples, characterise a specific malware type on a syntactic level. However, syntactic features are relatively easy to obscure or modify, as also pointed out by Moser et al. [1], mainly by techniques of encryption, packing [2], [3], [4], polymorphism, metamorphism, and code obfuscation, implemented on control-flow or data-flow of a program [5].

Malware researchers try to deal with code morphism by behavioural detection. Yet, Borojerdi and Abadi point out in their work [6] that not every semantics-based technique is successful. The level of abstraction on which program's behaviour is captured plays an important role. They mention that patterns of system calls on low level of abstraction can be circumvented but behavioural patterns related to utilisation of specific system resources provide more optimistic results.

We believe that when malware and security researchers don't focus primarily on creating new detection mechanisms but on defining (characterising) malicious software on rather high level of abstraction, they will gain solid foundations for such detection mechanisms, which will not lose applicability after short usage - and that is also our main goal: searching for various forms for characterising malware, on varied levels of abstraction and detail.

The aim of our work, presented in this paper, is summarised as follows:

- We look for proper form of malware behaviour representation on high level of abstraction. Our current formulation of behavioural pattern is presented (Section II).
- We try to find out whether representation of malicious behaviour on high level of abstraction is feasible, in order to improve detection precision or expand scope of detected malicious behaviour in future research. We present behavioural data extraction (Section III) and success rate of behavioural pattern extraction (Section IV).

II. High-Level Malware Behavioural Patterns

Concerning level of abstraction for malware features extraction, in our research we decided for a strategy to start with the most general, abstract features describing behaviour, with a possibility to gradually employ more detailed features and lower the level of abstraction later, when appropriate. At the current state of our research we aim at categories of behaviour, based on the area of influence on the infected operating system, such as behaviour affecting filesystem, actions on processes, network activity, modifications on registry entries (Table I). Analysis of behaviour regarding these categories is quantitative, i.e. we observe how many times each category of behaviour occurred in analysed program sample.

On this level of abstraction, which is quite high, we managed to observe repetitions in amounts of behaviour occurrences among behavioural categories. As it turned out, there were groups of distinct malicious samples, belonging to the same types of infiltrations (malware signatures), which performed actions according to some pattern. In our initial

TABLE I
12 CATEGORIES OF PROGRAM BEHAVIOUR WHICH TAKE PART IN FORMATION OF HIGH-LEVEL BEHAVIOURAL PATTERNS.

FC	file creation
FD	file deletion
MC	mutex creation
PC	process creation
SC	service creation
SS	service starting
RE	registry entry
D	DNS
WD	Winsock DNS
HG	HTTP get
HP	HTTP post
TF	TCP flow

work concerning this matter [7] we used formal notation to define high-level behavioural patterns as 12-tuple p_{label} of elements:

$$p_{label} = (n_{FC}, n_{FD}, n_{MC}, n_{PC}, n_{SC}, n_{SS}, n_{RE}, n_D, n_{WD}, n_{HG}, n_{HP}, n_{TF}), \quad n_{FC}, n_{FD}, \ldots, n_{TF} \in \mathbb{N}^0, \tag{1}$$

where $n_{FC}, \ldots, n_{TF}$ are numbers of occurrences of behaviours, listed in Table I, and $label$ is a name or an identifier of malicious signature with which is the pattern p associated.

The 12-tuple, as given in the definition (Equation 1), describes a case, when all samples of one type of infiltration (malware signature) show the same amounts of behaviour occurrences in behavioural categories, listed in Table I.

As we discovered by analysing behavioural data, such uniformity in behaviour is not that common and even if pattern is clearly recognisable, slight variability is present in some of behavioural categories. Thus, variability of behaviour occurrences was introduced in our work [7] by defining a set V_{label} of n-tuples v_k^l, which capture behaviour with varied occurrence and possibly group behaviours with potential interdependence (Equation 2):

$$V_{label} = \begin{cases} \emptyset, \text{ iff no variability in behaviour is present,} \\ \\ \{v_1^l, v_2^l, \ldots, v_k^l \mid v_k^l = (x_1, \ldots, x_n), \\ x \in \mathbb{N}^0, \;\; k, n \in \{1, 2, \ldots, 12\}, \;\; l \in \mathbb{N}^+\}, \\ \text{otherwise.} \end{cases} \tag{2}$$

Behavioural patterns can be graphically visualised for improved readability, e.g. Fig. 1 describes behavioural pattern with variability in behaviour occurrences.

The definition of pattern with variability of behaviour occurrences (Equation 2) is further explained and practical application is demonstrated in our work [7].

As the previous work showed, there are malware instances definable on high level of abstraction, by patterns comprising numbers of executed actions from 12 behavioural categories.

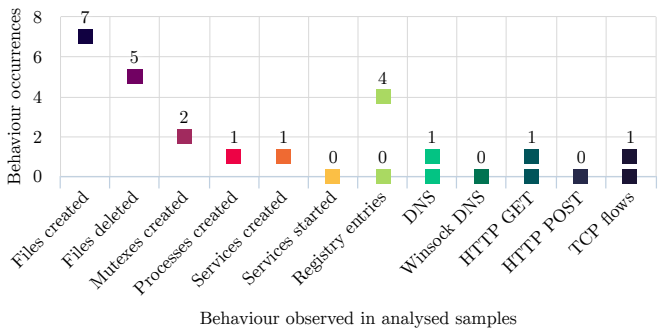

Fig. 1. Behavioural pattern of malicious program samples with signature labelled internally as E. Analysed samples demonstrated minor variability in number of behaviour occurrences regarding categories: *Registry entries*, *DNS*, *HTTP GET* and *TCP flows*. We avoid stating the real signature label on purpose, because disclosing such details may negatively influence employability of presented behavioural patterns in potential detection mechanisms.

III. Malware Behavioural Data Extraction and Sources

In the work aimed at defining malware, the nature of analytic data resources, their quality, and process of extraction, determine the end results of research experiments.

In several last years a form of crowdsourcing gains popularity concerning malware data acquisition. Online analytic services like *Totalhash*[1] or *VirusTotal*[2] provide numerous anti-virus engines or analytic tools to analyse user-provided suspicious files and assemble all analytic results. Not only they provide the analytic service, they serve also as a repository of previous analyses.

Our initial research in high-level malware behavioural patterns employed online analytic service *Totalhash*. First, we investigated what kinds of data are provided by the service and which of them have a potential for determining whether behavioural patterns are present among samples of the same type of malware. We succeeded in our efforts and behavioural patterns on high-level of abstraction have been found [7]. The research continued with advanced inspection of data from malware analyses that we gathered.

Online analytic service Totalhash provides various data in a form of a report, summarising results of analysis. Not all the reports contain the same types of information, it depends on the type of file that was analysed (Windows executable file, text document, image, script, ...) and the process of analysis itself - whether a certain stage of analysis was successful or not.

Analytic reports are quite extensive, so processing all of their data would be complicated and time consuming. That is why we resorted to simplification of behavioural data in a form of abstracting amounts of executed actions per sample, in accordance with the list of considerable program activity (Table I).

Assembling of behavioural data from Totalhash service is carried out by our custom software tool, which serves as a mediator for accessing the vast database in a simple and automatized way. The tool and its usage are described in our other papers [7] [8].

IV. Extractability of Malware Behavioural Patterns

As mentioned in Section II, a malware behavioural pattern was defined as a 12-tuple (Equation 1), stating numbers of occurrences for actions from each of 12 behavioural categories (Table I). At the time of writing of this paper, we have managed to process 34 099 analytic reports obtained from Totalhash service, even though assembling of more data is still in the process - over 200 000 entries of behavioural data are currently in our database, ready for future inspection.

Advanced analyses of data set with 34 099 samples have been made to assess employability of our approach for extracting malware behavioural patterns. Results of anti-virus analyses cannot be taken as a 100% reliable detection authority,

[1] Available at: http://totalhash.com/
[2] Available at: https://virustotal.com/

and to our knowledge, no such authority exists yet. With this in mind, we figured relative maliciousness and harmfulness of samples from the data set as follows:

- amount of samples detected by *each* of (at that time) available anti-virus engines as malicious with some malware signature was *664*,
- amount of samples detected as malicious with some malware signature by 16 anti-virus engines, which were selected as highly reliable based on independent anti-virus comparisons made by AV-Comparatives[3], AV-test[4] and VB100[5], was *1969*,
- amount of samples detected as potentially malicious, i.e. detected with some malware signature by *at least one* anti-virus engine, was *34 016*, so from the set of 34 099 samples, only *83* were "absolutely safe" - detected with *no virus*,
- amount of samples detected as potentially harmless, i.e. all of 16 highly reliable anti-virus engines, which were selected based on independent anti-virus comparisons made by AV-Comparatives, AV-test and VB100, detected no threat in those samples, was *739*,

Each anti-virus engine assigns a specific malware signature to the sample which was positively detected as malicious, thus numerous samples may belong to the same malware signature. Investigation of the data set revealed significant differences in amounts of malware signatures recognised among samples. Fig. 2 summarises these differences for 16 selected anti-virus engines, which were also mentioned above in the list. We do not mention names, just anonymised labels 1-16, of anti-virus engines on purpose, since it is not relevant information for this research.

By looking at Fig. 2, the difference between anti-virus engines is evident. While there is an engine which recognised totally 11 338 different malware signatures among 34 099 samples, on the opposite side of the chart, the other engine recognised only 1 656 malware signatures among the same set of samples. This situation probably only mirrors known issues regarding inconsistency of malware signatures labelling [9] among anti-virus products and also malware researchers.

In addition to amount of malware signatures recognised by various anti-virus engines, we had a look at amount of behavioural patterns found among analysed samples, separately for each of anti-virus engines. The numbers we obtained are separated into two groups:

- all behavioural patterns in total, where a pattern has at least one value of the 12-tuple common for all the samples which belong to malware signature corresponding with the pattern. In other words, these patterns may show variabilities in 11 behavioural categories from the 12-tuple, or less, even no variabilities at all,
- plain behavioural patterns, which have no variabilities in behaviour among samples at all, i.e. they correspond with the notation from the basic behavioural pattern definition (Equation 1).

The amount of patterns, when we looked at behavioural data in accordance with various anti-virus labelling systems, is summarised on Fig. 3. While there have been significant differences between amounts of recognised malware signatures, the relation between amount of recognised signatures and extractable behavioural patterns is quite similar for most of the considered anti-virus engines. The percentage of extractable behavioural patterns from recognised malware signatures is on average 10.72%, although much lower value 2.12% occurred with one anti-virus engine (number 6 on Fig. 3), and value significantly above average, more than 15%, occurred with three anti-virus engines (numbers 13, 15 and 16 on Fig. 3).

[3]Available at: https://www.av-comparatives.org/
[4]Available at: https://www.av-test.org/en/antivirus/
[5]Available at: https://www.virusbulletin.com/testing/

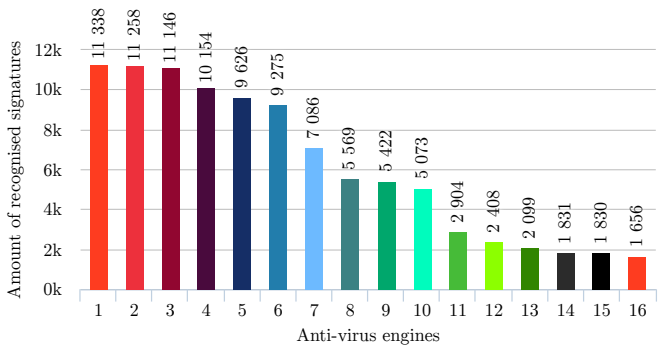

Fig. 2. Amount of malware types recognised under unique malware signatures for 16 anti-virus engines, selected as highly reliable based on independent anti-virus comparisons made by AV-Comparatives, AV-test and VB100.

V. Related Work

To our current knowledge, our approach to defining malware behaviour by patterns, comprising amounts of occurrences of actions from defined behavioural categories, is unique. However, there are research works worth mentioning, with which we share some techniques and research ideas.

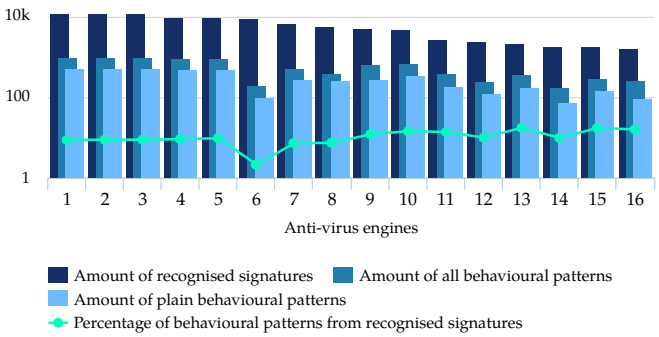

Fig. 3. Amount of malware types recognised under unique malware signatures, amount of all behavioural patterns extracted from samples corresponding with these signatures, amount of plain behavioural patterns (with no variations), and percentage of all behavioural patterns from amount of recognised signatures, for 16 anti-virus engines, selected as highly reliable based on independent anti-virus comparisons made by AV-Comparatives, AV-test and VB100.

Research of malware behaviour on the level of system function calls is discussed in work of Canzanese, Kam and Mancoridis [9]. They observed the amount of calls to system kernel API per second for each kernel function separately or for a sequence - two unique kernel functions. This is quite similar to our approach, however, they observed amounts of function calls for 235 different system functions, and we used higher level of abstraction - instead of system functions themselves as categories, we used types of actions based on the area of their influence in the infected operating system.

Cho and Im use in their work [10] analysis of system API call sequences for extracting patterns of API calls, which should define malware samples belonging to the same malware family. Authors were inspired by techniques of DNA sequencing from bioinformatics. Very interesting from our perspective is that they categorised API functions into 13 categories, according to their influence on host system resources: *registry, file system, process, service, network, socket, synchronization, system, device, threading, hooking, misc., Windows*. They could serve as an inspiration for enhancing our categorisation of behaviour.

Hellal and Romdhane statically extract function calls of system API from analysed programs and divide them into 32 main categories of behaviour, with additional 4 subcategories for 4 types of actions - open, read, write and close, so in total 128 behaviour categories are used [11]. They also observe sequence of function calls, which is statically extracted from a program as an API call graph. In comparison to our work, they use more detailed description of behaviour, but mainly their extensive *fine-grained* categorisation may serve for our inspiration.

Various approaches of behaviour analysis in the area of network security share the idea of pattern extraction, e.g. work of Konorski et al. [12]. Despite similarity of the concept, it is crucial to note that analysing network traffic and events initiated through network is markedly different from analysing actions performed during software execution.

VI. Future Work Proposal

Regarding assembling of behavioural data from online analytic service, our software tool which carries out the task will be adjusted for cooperating with more analytic services which provide data, not only with Totalhash.

Beside 12 behavioural categories which are currently included in our analytic system, also readable strings extracted from executable code are available for each malware sample which analytic report has been obtained from Totalhash service. These readable strings have not yet been analysed.

We also considered to enhance number of analysed behavioural categories, e.g. by taking inspiration form work of Cho and Im [10], mentioned in Related Work (Section V), who use 13 behavioural categories in their experiments.

An interesting inspiration comes also from work of Hellal and Romdhane, which was also mentioned in Related Work (Section V). We could observe behavioural patterns built on several different levels of malware behaviour categorisation, and compare those patterns. Currently we employ 12 behavioural categories, but inspired by Hellal and Romdhane, we could try to build patterns on 32 x 4 = *128* behavioural categories from the same data set, and compare extractability and relevance of those two levels of patterns.

From a long-term perspective, in our research we would like to proceed with more detailed information about malware behaviour, not only to observe amounts of behaviour occurrences in 12 behavioural categories, but to track e.g. which specific system functions implement the behaviour, or whether malware samples, belonging to the same malware signature, use the same types of system functions.

Acknowledgment

This work was supported by the Slovak Research and Development Agency under the contract No. APVV-15-0055 and by project KEGA no. 079TUKE-4/2017.

References

[1] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection," in *Twenty-Third Annual Computer Security Applications Conference, ACSAC 2007*, Dec 2007. doi: 10.1109/ACSAC.2007.21 pp. 421–430.

[2] S. Josse, "Secure and advanced unpacking using computer emulation," *Journal in Computer Virology*, vol. 3, no. 3, pp. 221–236, 2007. doi: 10.1007/s11416-007-0046-0

[3] J. Stastna and M. Tomasek, "Exploring malware behaviour for improvement of malware signatures," in *IEEE 13th International Scientific Conference on Informatics, 2015*, Nov 2015. doi: 10.1109/Informatics.2015.7377846 pp. 275–280.

[4] J. Šťastná and M. Tomášek, "The problem of malware packing and its occurrence in harmless software," *Acta Electrotechnica et Informatica*, vol. 16, no. 3, pp. 41–47, 2016. doi: 0.15546/aeei-2016-0022

[5] J.-M. Borello and L. Mé, "Code obfuscation techniques for metamorphic viruses," *Journal in Computer Virology*, vol. 4, no. 3, pp. 211–220, 2008. doi: 10.1007/s11416-008-0084-2

[6] H. R. Borojerdi and M. Abadi, "Malhunter: Automatic generation of multiple behavioral signatures for polymorphic malware detection," in *3th International eConference on Computer and Knowledge Engineering (ICCKE), 2013*, Oct 2013. doi: 10.1109/ICCKE.2013.6682867 pp. 430–436.

[7] J. Šťastná and M. Tomášek, *Characterising Malicious Software with High-Level Behavioural Patterns*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2017, vol. 10139, pp. 473–484. doi: 10.1007/978-3-319-51963-0_37

[8] P. Hlinka, M. Tomášek, and J. Šťastná, "Collecting significant information from results of malicious software analysis," *Electrical Engineering and Informatics 7*, pp. 103–108, 2016.

[9] R. Canzanese, M. Kam, and S. Mancoridis, "Toward an automatic, online behavioral malware classification system," in *2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, Sept 2013. doi: 10.1109/SASO.2013.8 pp. 111–120.

[10] I. K. Cho and E. G. Im, "Extracting representative api patterns of malware families using multiple sequence alignments," in *Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems*, ser. RACS. New York, NY, USA: ACM, 2015. doi: 10.1145/2811411.2811543 pp. 308–313.

[11] A. Hellal and L. B. Romdhane, "Minimal contrast frequent pattern mining for malware detection," *Computers & Security*, vol. 62, pp. 19 – 32, 2016. doi: https://doi.org/10.1016/j.cose.2016.06.004

[12] J. Konorski, P. Pacyna, G. Kolaczek, Z. Kotulski, K. Cabaj, and P. Szalachowski, "Theory and implementation of a virtualisation level future internet defence in depth architecture," in *Int. J. of Trust Management in Computing and Communications*, vol. 1, no. 3, 2013. doi: 10.1504/IJTMCC.2013.056431 pp. 274–299.