Malware Behavior Modeling with Colored Petri Nets

Bartosz Jasiul, Marcin Szpyrka, Joanna Śliwa

Malware behavior modeling with Colored Petri nets

Bartosz Jasiul[1], Marcin Szpyrka[2], and Joanna Śliwa[1]

[1] Military Communication Institute
C4I Systems' Department
ul. Warszawska 22a, 05-130 Zegrze, Poland
{b.jasiul, j.sliwa}@wil.waw.pl
[2] AGH University of Science and Technology
Department of Applied Computer Science
al. Mickiewicza 30, 30-059 Kraków, Poland
mszpyrka@agh.edu.pl

Abstract. We propose a solution which provides a system operator with a mechanism that enables tracking and tracing of malware behavior which – in consequence – leads to its detection and neutralization. The detection is performed in two steps. Firstly single malicious activities are identified and filtered out. As they come from the identification module, they are compared with malware models constructed in the form of Colored Petri nets. In this article we present our approach to malware modeling. Proposed method was implemented and practically verified in laboratory environment with emulated malicious activity at the hosts level.

Keywords: malware, cyber attack, Colored Petri net, malware detection, behavioral analysis

1 Introduction

Computer systems are prone to cyber attacks even though they have number of security controls already deployed. Cyber criminals are focused on finding a way to bypass security controls and gain access into a protected network or a single host. For that reason organizations, companies, governments and institutions as well as ordinary citizens all over the world are interested in detection of all attempts of malicious actions targeted on their computer networks and single machines.

In general, the success rate of the applied method for malware detection depends on the reliability of the used malware model. Usually they are based on signatures which are bits of application code typical for malicious activity. Security controls (e.g. antivirus tools) might be maladjusted because signatures of new threats are not identified yet. Hackers often use existing parts of code in order to implement new types of malware. This allows, in return, to quickly develop signatures of new dangerous software. Therefore, the more signatures are deployed the more malicious applications are identified. On the other hand, one of the methods of misleading the signature-based detection systems is code obfuscation, the aim of which is generating – from already existing code – a new application that cannot be assessed yet as risky by security control [29],

[31]. This technique is simple to be used and potentially successful, so that also successful countermeasures are necessary. One of the examples is to follow behaviors of malicious software in order to identify them and eliminate from the protected system and computer machines. This article is focused on modeling of malware behavior with Colored Petri nets (CP-nets) [12], [30]. Our approach was implemented and verified in laboratory environment with success and various types of malware were detected.

2 Evading virus detection technologies

The method of evading antivirus tools is generally called obfuscation. It is a technique aimed at generating new software that realizes the same functions as the original one but does not have its specific code signatures. It can be realized by modification of Java scripts, additional loops in the code that return to the point of execution (zero loops), encryption techniques run at program execution, etc.

The list of obfuscation techniques includes, but is not limited to:

- *Parasitic obfuscation* that is used to append, prepend, or insert code into data sections of files on disk [4].
- *Self-modification* that allows malware to modify its code during every infection. Thus, each infected file contains different variant of the virus [19].
- *Polymorphic coding* that is an obfuscation that consists in infecting files with an encrypted copy of the virus [1]. At each time an encryption key or even encryption method can be modified, therefore virus codes are different from one another in infections causing their signatures to be hard to detect [6]. If some part of code remains the same, an anti-virus tool can decrypt the code using an emulator. However, it is not always a successful technique. It allows to detect some malware and produce new signatures for them.
- *Metamorphic coding* that is a technique of rewriting the functions of software at each infection in a different way [5], [23]. Viruses that utilize this technique are very large and complex. Metamorphism makes viruses almost undetectable by signature-based tools.

Obfuscation techniques are very successful in hiding malicious code against byte-level content analysis [13], [15] and static analysis methods [8], [9] which make cyber attacks undetectable. Significant effort is made by cyber criminals in order to thwart detection by anti-malware tools. Moreover, methods of evading antivirus products will be developed as long as cyber crimes are profitable.

3 Malware detection techniques – an overview

Great effort has been put lately in static analysis of malicious codes because this technique generally has brought good accuracy in malware detection [3], [14], [25]. Even though it is an appropriate technique [16] in case of traditionally compiled machine code, the most difficult problem it faces is difficulty to handle obfuscated binaries [28]. Additionally, obfuscation techniques are perceived as NP-hard for static analysis [20].

On the other hand, dynamic malware analysis is directed at reaching reliable tracks of executed malicious codes. Dynamic malware analysis may be based on setting up behavior clusters from sequences and measuring distances between single events [2], [18]. However, this approach suffers from the lack of external rules for data analysis. According to [7], [24] a successful method of dynamic malware analysis is comparison of specifications of malicious behavior with *hooked* processes at application level. Therefore one of the problems of dynamic malware analysis is necessity of building models of this malicious activity. The approach to malware modeling proposed in this article is based on utilization of Colored Petri nets used during the dynamic malware analysis.

It is worth to mention that Petri nets were already successfully adapted for identification of cyber threats. In work [17] authors observed that mathematical representation of Petri nets allows for modeling of computer misuse. Proposed mechanisms consisted in representation of known attack as a sequences of events. In this case the attack was presented as a Petri net graph. Comparing misuses with the Petri net graph allowed for detection of unwanted actions.

Colored Petri nets were also utilized for detection of DoS attacks in Wide Area Networks [10]. In this case Colored Petri nets were adapted to model router network connections in the area of The United States. It was proved that modifications in the network infrastructure made by DoS attacks can be detected by comparison of the current state to the modeled one. Moreover, this method was proposed as an early warning system against network attacks. Additionally, it can support development of network infrastructure security strategies.

Next major contribution in utilization of Colored Petri nets was identified in work [33] supported by US Air Force Office of Scientific Research. This outstanding research presents a new approach to formal specification of the malicious functionalities based on activity diagrams (Unified Modeling Language – UML diagrams) defined in an abstract domain. It introduces abstract functional objects that, along with system objects, could be used for creating generic specifications covering multiple functionality realizations. Methodology proposed in this work utilizes Colored Petri nets for recognition of functionalities at the system call level.

Our particular usage is also crucial in this article for CP-nets application in cyber defence, especially in malware modeling and detection, what has been shown in the following sections.

4 PRONTOnet – malware tracking

4.1 An architecture of the solution

In our work we proposed and developed behavior-oriented malware hunting tool, so-called PRONTO, that could be used in parallel to existing signature-based tools. The main assumption for the introduced method is that the malware was not recognized yet by the signature mechanisms. The aim therefore is to track its suspicious activities in order to find it while running in the system. PRONTO hunting tool performs its activity in two stages (Fig. 1):

– **Filtering of the system events** registered by the system monitors (sensors) to discover the main features of the hostile activity. These features are related to particular objects and actions triggered on that objects – e.g. registry (add entry, modify entry, delete registry entry, etc.), process (start, stop process, etc.), file (copy, delete, run, open, close file, etc.), domain (connect to, etc.), IP address (connect to, etc.);
– **Tracking suspicious activity** in order to discover malicious exploits running in the system. Filtered events are correlated in order to find similarities with the stored malware activities modeled in the form of Colored Petri nets. The result of malware tracking is the alarm that contains information vector about malicious activity, similarity to the known attacks and list of incidents that affected the system.

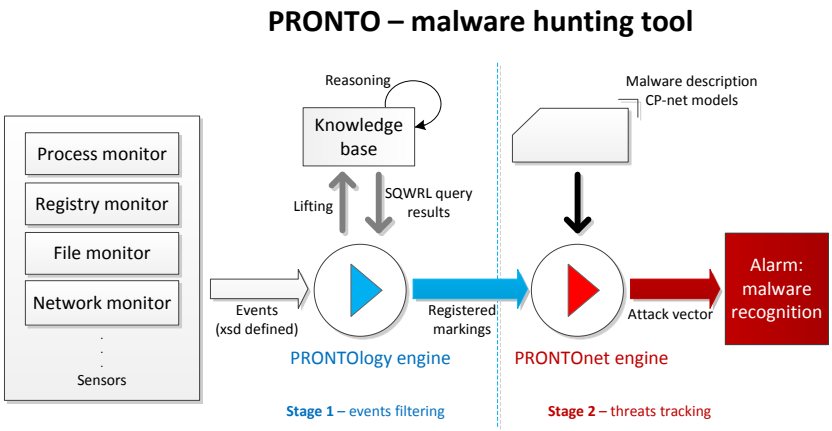

Fig. 1. PRONTO – malware hunting tool

The first stage which is related to capturing events from sensors and analyzing them with an expert system that uses – defined for the purpose of the method – comprehensive ontology, so called PRONTOlogy. Registered events in the form of XML objects are sent to the PRONTOlogy engine and lifted to add entries to the Knowledge Base. PRONTOlogy describes events registered by system monitors and is able, on the basis of rule engine and inference, with the use of specially defined rules [21], [22], to classify an event as potentially suspicious, malicious or regular. As a result, markings of the modeled malware in the form of CP-nets are delivered for further analysis.

This article presents only the second stage PRONTOnet which is related to malware modeling and detection with the use of Colored Petri nets. PRONTOnet provides formal model of malware behavior and allows to track suspicious activities potentially assigning them to the class of known malware types or identifying unknown ones. Known exploits can be invisible to signature-based malware detecting tools after their code has been obfuscated, although their activities can be easily observed. It also happens often that a new malware piece of software is composed of known components from other ones. This results in another behavior pattern that can be tracked as a new exploit, not

identified yet. The result of threats tracking stage is an alert informing about identification of suspicious or malicious events with a certain similarity rate to the known malware types.

4.2 Colored Petri nets

Colored Petri nets (CP-nets) [12] provide graphical notation typical for Petri nets, but net elements are described using high level programming language (e.g. CPN ML). They take the form of bipartite directed graph, with place nodes holding values (called *tokens*) of selected types (called *color sets*), transition nodes consuming and producing tokens, and arcs between these two kinds of nodes specifying dependencies between transitions and places, and the values on them (see Fig. 3, 4, 5, 6). A transition is *enabled* if sufficient tokens are available on its input places. Such an enabled transition can *fire* by consuming tokens along each input arc and producing tokens along each output arc. To cope with tokens of different types the inscription language uses variables and expressions typical for programming languages.

For an effective modeling CP-nets enable to distribute parts of the net across multiple subnets called modules. The result of such an approach is a hierarchical CP-net [12]. *Substitution transitions* and *fusion places* are used to combine modules. The former idea allows the user to refine a transition and its surrounding arcs to a more complex net, which usually gives a more precise and detailed description of the activity represented by the substitution transition. A fusion of places allows users to specify a set of places that should be considered as a single one. It means, that they all represent a single conceptual place, but are drawn as separate individual places (e.g. for clarity reasons).

4.3 An approach to malware tracking

Let us assume we model malicious behavior of malware with the use of vocabulary applied in the theory of the Colored Petri nets. In this case, places are any nodes, files, protocols, processes or any other assets in the monitored system, transitions are any operations made on system assets, arcs are shifts that activate assets in the system, and color sets are sets of values or pointers indicating particular system assets.

Now let us assume the situation (as presented in Fig. 2) when in the monitored computer system a web browser is activated in order to visit some Internet resources (e.g. a web page). This web page contains malicious code which is downloaded to the system and then executed. After being activated it is responsible for logging user keystrokes when https sites are visited (e.g. banks) and sending registered streams to the command and control center (so called C&C). When analyzing this case it is assumed that the code of the malicious software program passed successfully rigorous verification by signature based mechanisms (an anti-virus application), was not recognized so far and is able to activate itself in the system. Firstly, a virus is setting up his presence in the system. This is called the establishing presence phase and includes downloading additional codes and commands from C&C, deactivating security metrics (switching off firewall, antivirus, other intrusion detection mechanisms and security controls). After this step, the exploit is able to execute malicious activities and disseminate itself to other systems.

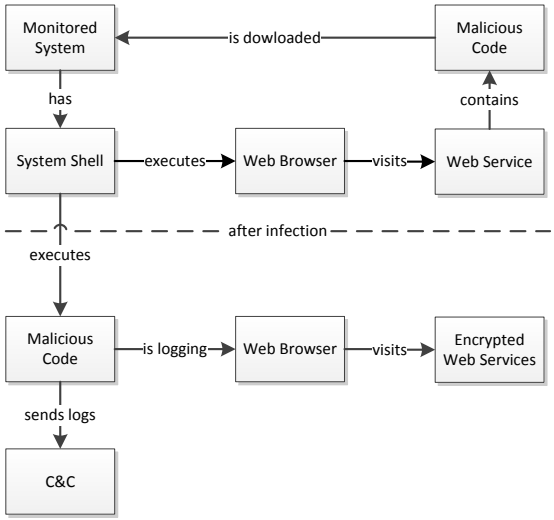

Fig. 2. Example of malicious activity

In order to follow our CP-net modeling approach and get familiar with the basis of CP-nets, one can investigate Fig. 2, which presents relationships among assets that take part in malicious activities that are run in the system. The assets such as System Shell, Web Browser, Malicious Code, Web Service, etc. are in fact places in the CP-net model. Transitions are actions realized on those system assets such as execution, logging, browsing, sending, etc. Arcs are arrows depicted in the picture. Color sets are e.g. names of registry entries, locations of files, their handlers after execution, sent data, IP addresses or domain names of C&Cs.

4.4 Utilization of CP-net models for malware tracking

Malware that is produced almost never is deployed without a obfuscation technology that hides malicious code against signature based anti-virus tools. Obfuscation is a fast and the most popular way for hackers to generate new malicious tools without much investment. These techniques cause signature based mechanisms insufficient for detecting new malicious activities. Regarding that, a new exploit which is composed of known malware executes the same functions and operations on the system. Therefore, even performed among many actually harmless actions, it is possible to detect some diverse actions like:

- operations on particular files,
- operations on registry entries,
- executed processes and applications,
- communication with specific IP addresses,
- communication with domains.

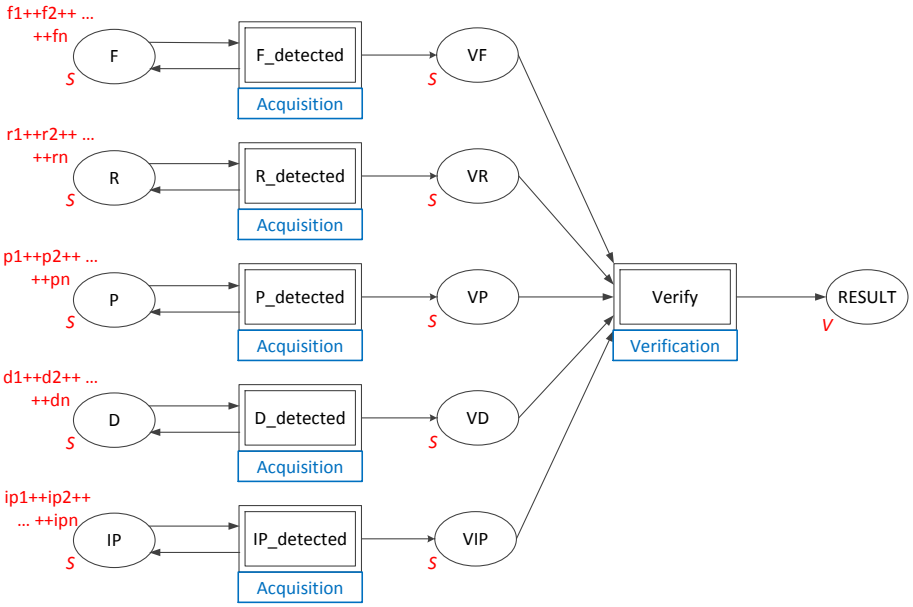

Fig. 3. CP-net model of PRONTOnet – prime module

The malware detection takes as an input a set of suspicious events received from process' hooking engine, so called PRONTOlogy, developed by authors of this article [11]. This engine is based on ontology reasoning [26], [27], [32] used for the purpose of filtering single malicious incidents among hundred thousands regular ones. These single events are passed to PRONTOnet engine for further investigation.

PRONTOnet uses CP-net models of malware and performs malware detection passing through particular places in the model, using CP-net vocabulary and characteristics. The CP-net model defined by the authors is hierarchical and has been described by four modules layers: *prime* (Fig. 3), *acquisition* (Fig. 4), *verification* (Fig. 5), and classifiers layer. A classifier module for the *Virut* malware is shown in Fig. 6.

The *prime module* of CP-net model representing PRONTOnet threat tracking tool is depicted in Fig. 3. On the left hand side of this figure there is a column of places storing tokens that represent particular assets that might be affected by malware:

- F – a place storing tokens indicating files;
- R – registry entries;
- P – processes;
- D – domains;
- IP – IP addresses that malware may communicate with.

The second column in Fig. 3 is composed of *substitute transitions* that are related to the *Acquisition* process depicted in Fig. 4. The next column is made up of places indicating particular assets affected by malware activated in the monitored system. Markings of these places (i.e. tokens stored in them) are processed by the substitute transition

called *Verify* in order to deduce that system is infected by certain malware type. In consequence, place *RESULT* is marked with a vector informing about malware and symptoms that indicate the malware.

Let us also explain color sets and inscriptions in Fig. 3. For this purpose we have defined color types and inscriptions in PRONTOnet as presented in Listing 1.1.

Listing 1.1. Color types and inscriptions in PRONTOnet prime module

```
colset S = String;
colset I = Integer;
colset Symptoms = List S;
colset V = Product I * S * Symptoms;
```

Color set `S` is a string type. This way particular assets are described by variables `f`, `r`, `p`, `d`, `ip` of type `S`, e.g. file `C:\[WINDIR]\System32\svchost.exe` or IP address `66.232.126.195`. Color set `I` is the Integer type and is used as threat identity number, and color set `Symptoms` is a list of strings describing assets suspected to be infected by malware. Color set `V` is a product that indicates threat identity number, name of the threat, and list of symptoms that were used for identification which attack was executed in the system.

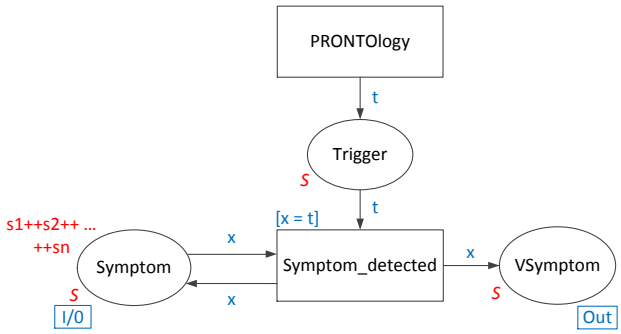

Fig. 4. CP-net – Acquisition module

Symptom acquisition process and co-operation with PRONTOlogy module is presented in Fig. 4. The module is assigned to each substituted transition from the higher level. Place *Symptom* is an input/output port that indicates appropriate places *F, R, P, D,* and *IP* from the higher level module (Fig. 3). In the *Acquisition module* tokens that represent filtered suspicious activities identified by PRONTOlogy are passed to the *VSymptom* place if the same token exists at *Symptom* place. Identified suspicious actions mark *VSymptom* place for further processing by the *Verify* transition. Marking of *Symptom* place contains all tracked elements of the monitored system, e.g. all IP addresses that the system may communicate with and download malicious software. Transition *Symptom_detected* is developed in order to test existence of appropriate token in *Symptom* place in case the *Trigger* place is marked. If compared tokens are different, the transition does not react. The conformity of tokens is required by the guard $[x = t]$. The module

is also prepared to detect more than one exploit that uses the same *tricks* to switch off system security controls or even two or more malware using the same malicious code. Marking of *Symptom* place is not reduced while *Symptom_detected* transition is enabled. This module shows an important role of PRONTOlogy in detection of suspicious events and passing them to PRONTOnet module.

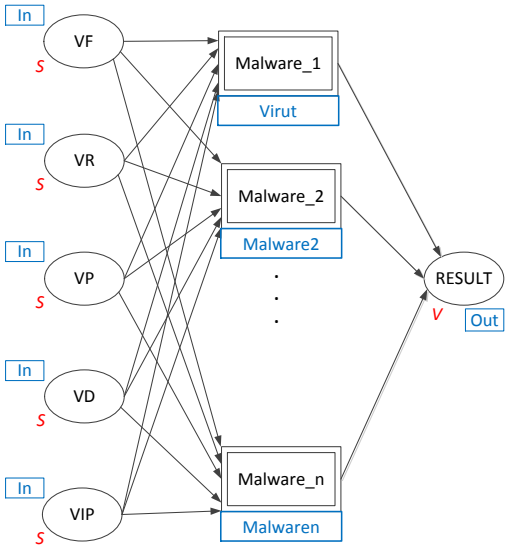

Fig. 5. CP-net – Verification module

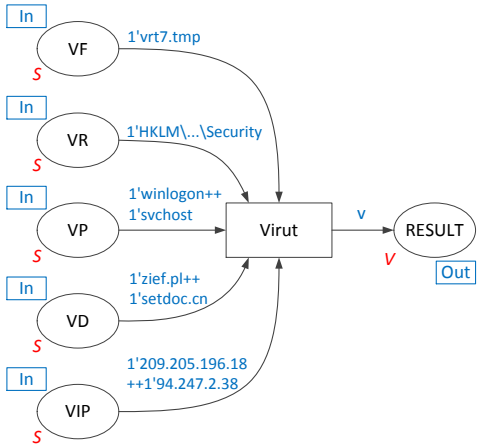

Fig. 6. CP-net – Virut module

Verification module is presented in Fig. 5. It must be noted that substitute transition *Verify* in *primary module* represents multiple transitions designed for identification of various malware types. This indicates that in particular marking of *V* places appropriate transition for particular malware (verification process) is enabled. An exemplary virus detection for chosen marking is shown in Fig. 6, which presents detection of malware called Virut. In the presented example, assuming appearance of the current marking, transition `Virut` is enabled, which in consequence leads to receiving vector `v` informing about detection of the Virut malware. The structure of vector `v` is as follows:

Listing 1.2. Structure of vector informing about detection of the Virut malware

```
1’ 1 | Virut | vrt7.tmp, HKLM\...\Security, winlogon, svchost,
zief.pl,  setdoc.cn, 209.205.196.18, 94.247.2.38.
```

The CP-net models presented in this section allow to easily update symptoms of a new attack by changing initial markings of places. Every time when a new malware model is entered to the PRONTOnet appropriate actualization of places markings must be realized.

It is also worth to emphasize that the detection of malware is not limited only to depicted resources. PRONTOnet may be also used to identification of exploits through analysis of network traffic and system statistics. If some resource is identified as useful for malware detection, the *primary module* needs only to be updated with additional places and transitions.

5 Conclusions and future work

On the basis of presented method an application for malware detection and modeling was developed. Symptoms of particular malware that form the CP-net model are edited with the use of the tool presented in Fig. 7. It offers *drag and drop* functionality which allows to add subsequent places to the model easily. This software is in the development

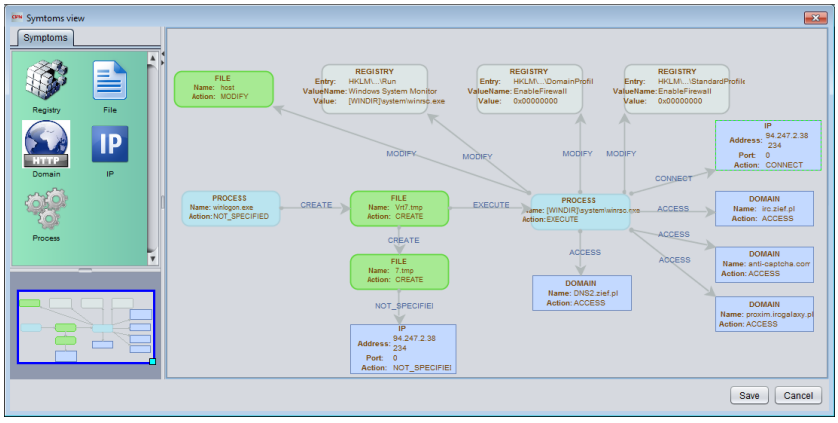

Fig. 7. Editor of malware symptoms

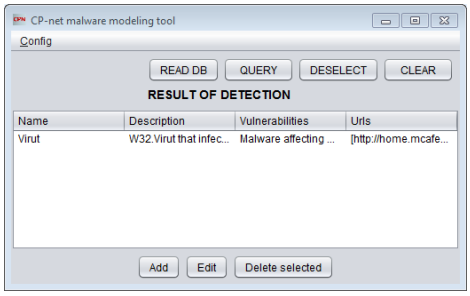

Fig. 8. Result of detection Virut malware

stage, therefore its functions are limited. Transitions are shown, currently, as operations over places. This software will be updated continuously until the end of the ongoing project in this matter at Military Communication Institute. Its results are planned to be demonstrated and tested at the nearest annual edition of NATO CWIX exercises. Nowadays this tool allowed us to detect dozens of malware. One of them is the Virut malware, the detection of which is presented in Fig. 8. A separate article about malware detection and malware modeling tool will be published after advanced tests in the malicious laboratory environment and real exercises with hostile software.

Acknowledgements

This work has been partially supported by the National Centre for Research and Development project no. PBS1/A3/14/2012 "Sensor data correlation module for detection of unauthorized actions and support of decision process", by the European Regional Development Fund the Innovative Economy Operational Programme, under the INSIGMA project no. 01.01.02-00-062/09 and by AGH University of Science and Technology internal project no. 11.11.120.859.

References

1. Aucsmith, D.: Tamper–resistant software: An implementation. Lecture Notes in Computer Science: Information Hiding: First International Workshop **1174**, 317–333 (1996)
2. Bailey, M., Andersen, J., Morleymao, Z., Jahanian, F.: Automated classification and analysis of internet malware. In: Proceedings of Recent Advances in Intrusion Detection (2007)
3. Bereziński, P., Szpyrka, M., Jasiul, B., Mazur, M.: Network anomaly detection using parameterized entropy. In: Computer Information Systems and Industrial Management Proceedings of the 13th IFIP TC8 International Conference CISIM 2014, LNCS. Springer-Verlag (2014)
4. Bonfante, G., Kaczmarek, M., Marion, J.Y.: A classification of viruses through recursion theorems. In: Proceedings of the 3rd conference on Computability in Europe: Computation and Logic in the Real World, CiE '07, pp. 73–82. Springer-Verlag (2007)
5. Borello, J.M., Mé, L.: Code obfuscation techniques for metamorphic viruses. Journal in Computer Virology **4**(3), 211–220 (2008). DOI 10.1007/s11416-008-0084-2

6. Cappaert, J., Preneel, B., Anckaert, B., Madou, M., De Bosschere, K.: Towards tamper resistant code encryption: practice and experience. In: Proceedings of the 4th int. conf. on Information security practice and experience, ISPEC'08, pp. 86–100. Springer-Verlag (2008)
7. Christodorescu, M., Jha, S., Kruegel, C.: Mining specifications of malicious behavior. In: Proc. of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Int. Symposium on Foundations of Software Engineering, pp. 5–14 (2007)
8. Christodorescu, M., Jha, S., Seshia, S., Song, D., Bryant, R.: Semantics-aware malware detection. In: IEEE Symposium on Security and Privacy, pp. 32–46 (2005)
9. Flake, H.: Structural comparison of executable objects. In: Proc. of the IEEE Conference on Detection of Intrusions and Malware & Vulnerability Assessment, pp. 161–173 (2004)
10. Healy, L.: A model to study cyber attack mechanics and denial-of-service exploits over the internet's router infrastructure using Colored Petri Nets. Tech. rep., http://commons.emich.edu/theses/218 (2009). Masters Theses and Doctoral Dissertations.
11. Jasiul, B., Śliwa, J., Gleba, K., Szpyrka, M.: Identification of malware activities with rules. In: Proceedings of the Federated Conference on Computer Science and Information Systems. Warsaw, Poland (2014)
12. Jensen, K., Kristensen, L.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems, 1st edn. Springer-Verlag, Berlin, Heidelberg (2009)
13. Karim, M., Walenstein, A., Lakhotia, A., Parida, L.: Malware phylogeny generation using permutations of code. Journal in Computer Virology **1**, 13–23 (2005)
14. Kirda, E., Kruegel, C., Banks, G., Vigna, G., Kemmerer, R.: Behavior-based spyware detection. In: Usenix Security Symposium (2006)
15. Kolter, J., Maloof, M.: Learning to detect and classify malicious executables in the wild. Journal of Machine Learning Research **7**, 2721–2744 (2006)
16. Kruegel, C., Robertson, W., Vigna, G.: Detecting kernel-level rootkits through binary analysis. In: Proceedings of the Annual Computer Security Applications Conference (2004)
17. Kumar, S., Spafford, E.: A Pattern Matching Model for Misuse Intrusion Detection. Tech. rep., http://docs.lib.purdue.edu/cstech/1170 (1994). Computer Science Technical Reports.
18. Lee, T., Mody, J.: Behavioral classification. In: Proceedings of EICAR Conference (2006)
19. Linn, C., Debray, S.: Obfuscation of executable code to improve resistance to static disassembly. In: Proceedings of the 10th ACM Conf. on Computer and Communications Security, pp. 290–299. ACM (2003)
20. Moser, A., Kruegel, C., Kirda, E.: Limits of static analysis for malware detection. In: Proceedings of the Annual Computer Security Applications Conference (2007)
21. Nalepa, G., Bobek, S.: Rule-based solution for context-aware reasoning on mobile devices. Computer Science and Information Systems **11**(1), 171–193 (2014)
22. Nalepa, G., Ligęza, A.: Designing reliable Web security systems using rule-based systems approach. In: Advances in Web Intelligence. First International Atlantic Web Intelligence Conference AWIC 2003, *LNCS*, vol. 2663, pp. 124–133. Springer (2003)
23. Rad, B., Masrom, M., Ibrahim, S.: Camouflage in malware: from encryption to metamorphism. In: Int. Journal of Computer Science and Network Security, vol. 12, pp. 74–83 (2012)
24. Rieck, K., Holz, T., Willems, C., Düssel, P.: Learning and classification of malware behavior. In: Fifth Conf. on Detection of Intrusions and Malware & Vulnerability Assessment (2008)
25. Sharif, M., Yegneswaran, V., Saidi, H., Porras, P., Lee, W.: Eureka: A framework for enabling static malware analysis. In: Proceedings of the 13th European Symposium on Research in Computer Security, ESORICS '08, pp. 481–500. Springer-Verlag (2008)
26. Śliwa, J., Gleba, K., Chmiel, W., Szwed, P., Głowacz, A.: IOEM – Ontology engineering methodology for large systems. In: Computational Collective Intelligence. Technologies and Applications, *LNCS*, vol. 6922, pp. 602–611. Springer-Verlag (2011)

27. Śliwa, J., Jasiul, B.: Efficiency of dynamic content adaptation based on semantic description of web service call context. In: Proceedings - IEEE Military Communications Conference MILCOM 2012, Orlando, USA, pp. 1–6 (2012). DOI 10.1109/MILCOM.2012.6415810
28. Szor, P.: The Art of Computer Virus Research and Defense. Addison–Wesley Professional. Symantec Press series (2005)
29. Szpyrka, M., Jasiul, B., Wrona, K., Dziedzic, F.: Telecommunications networks risk assessment with Bayesian networks. In: Computer Information Systems and Industrial Management Proceedings of the 12th IFIP TC8 International Conference CISIM 2013, *Lecture Notes in Computer Science*, vol. 8104, pp. 277–288. Springer-Verlag (2013)
30. Szpyrka, M., Szmuc, T.: Decision tables in Petri net models. In: Rough Sets and Intelligent Systems Paradigms, *LNCS*, vol. 4585, pp. 648–657. Springer-Verlag (2007)
31. Szwed, P., Skrzyński, P.: A new lightweight method for security risk assessment based on fuzzy cognitive maps. International Journal of Applied Mathematics and Computer Science **24**(1), 213–225 (2014)
32. Tarapata, Z., Chmielewski, M., Kasprzyk, R.: An algorithmic approach to social knowledge processing and reasoning based on graph representation: A case study. In: Proceedings of the Second International Conference on Intelligent Information and Database Systems, *LNCS*, vol. 5991, pp. 93–104. Springer-Verlag (2010)
33. Tokhtabayev, A., Skormin, V., Dolgikh, A.: Dynamic, resilient detection of complex malicious functionalities in the system call domain. In: MILCOM, Military Communications Conference, pp. 1349–1356 (2010). DOI 10.1109/MILCOM.2010.5680136