INTEGRATED SOFTWARE PLATFORM FOR MALWARE ANALYSIS OF MOBILE TERMINALS

LtCol. Eng. Associate Professor Dragoș-Iulian BĂRBIERU, PhD*
Col. Ștefan-Antonio Dan ȘUTEU, PhD**
Associate Professor Elena ȘUȘNEA, PhD***

Beyond the marketing of IT companies, in the context of escalating cyber-attacks that affect organizations around the world, cyber security solutions have become the primary element in protecting IT infrastructures and devices. The proliferation of Intelligent Mobile Devices and Cloud Technologies, the Internet of Things requires new technological solutions, implemented both at hardware and software levels, to combat threats. This paper summarizes the Integrated Software Platform for Malware Analysis of Mobile Terminals which aims to integrate various software technologies to protect mobile devices.

Keywords: malware analysis; cyber security; mobile terminals.

Introduction

The malware applications analysis for mobile terminals is a difficult process due to the diversity of mobile platforms and existing security mechanisms, the frequency of occurrence of operating system versions and the use of malware code protection techniques. In the context of the national and international situation shaped by security trends, there has been a need to develop a software platform integrating, in a unitary manner, various open-source and commercial malware analysis solutions for mobile telephony. Most cyber actors are adapting to an existing environment, but information and technology supremacy is achieved through innovation, as Vice Admiral Arthur K. Cebrowski says: "I realized that military competition wasn't about how fast one could align with reality, but how fast one could leap over it and create a new reality"[1].

Cyber security and security in general are closely linked, as security comes from most cyber-attack and defense methods and techniques. The 7 stages of the Cyber Kill Chain[2] presented by the Lockheed-Martin Corporation are identical to the stages of an attack against a person or group of people. The static analysis of malware can be seen as an investigation to establish a person's psychological profile. Although it is more cost-effective than dynamic analysis, a program can hide a malicious code by encryption or different methods, just as a person can cheat at a personality questionnaire. Dynamic analysis involves executing a program and tracking all parameters to identify suspicious activities in a controlled environment.

The honeypot concept, the techniques to check if the environment where you are acting is secure are similar to real life when a person is under the magnifying glass of a detector in a safe environment or designed for sure by the one who wants to track down certain events. The attacks such as distributed denial of service can be related to the intoxication of an opponent with false information that consumes time and resources until they are exhausted. The rapid development of information and communications technology and "easy Internet access have not only yielded indisputable benefits but also it brings some vulnerabilities to security environment"[3]. Hybrid warfare through various third parties is mirrored today in the world of the Internet, using various proxy technologies and specialized hacking groups. Regardless of present or future technologies, patterns that are limited in number can be identified, but the manifestation is

*_Security and Defence Faculty_
e-mail: *oldboy@yahoo.com*
**_Command and Staff Faculty_
e-mail: *dan-suteu.antonio@unap.ro*
***_Security and Defence Faculty_
e-mail: *susnea.elena@unap.ro*

inexhaustible. These patterns are not characteristic of present days, but have their roots in our species' history and are forms of attack and defense, many of which are borrowed from biology. Camouflage and mimicry are weapons from the animals' arsenal and can secure victory against a possible opponent[4]. We believe that cyber space, much more diverse through intrusion and protection manifestations, operates a limited set of existing patterns in biology as well, resulting from the long evolutionary process.

Malware uses different camouflage techniques. It can be installed in the distribution chain, so a user cannot see any changes to the device activity that often occurs after a program has been installed. Compromising the signal processor inevitably leads to interception of telephone calls and messages, but using existing correlations between DSP and CPU, attackers can get extensive capabilities on applications running on the mobile terminal. By offering free apps or apps from unofficial stores, people can insert malicious codes. The control flow obfuscation procedure prevents the dynamic analysis of malware. Using encryption algorithms will lead to the inability to disassemble and decompile the code of an application.

Detecting the malicious behavior of mobile terminals involves 3 types of analysis: static analysis involves disassembling and decompiling an application to identify malicious code, dynamic analysis tracks different parameters and events in a sandbox-controlled environment to identify suspicious behaviors, the hybrid analysis combines the two types of analysis briefly presented above. Malicious code detection typically uses a signature list, and if this process fails, artificial intelligence algorithms or manual analysis can be used. The approach to malware behavior from a machine learning perspective involves a series of steps: "selecting the initial set of data (training set), as a rule, an equal number of safe and malicious applications from which some features are extracted"[5]. Based on some feature selection methods, it selects the most relevant to build a model using a classification algorithm. In the test phase, the model is checked to evaluate the accuracy using different metrics. Choosing features is not a random process. The range of classification algorithms is diverse, with approaches based on statistics, neural networks and kernel-based methods. The most common classification algorithms are Naive Bayes, K-Neares Neighbors and Vector Machine Support.

Vulnerabilities can be of two types, preinstalled or generated by the complexity of the internet. Preinstalled vulnerabilities can be installed by a producer or in the distribution chain. It is almost impossible to check and test each piece of code.

Software development platforms, such as GitHub, could provide tools to check for various errors in the code in the future, in order not to be exploited by attackers.

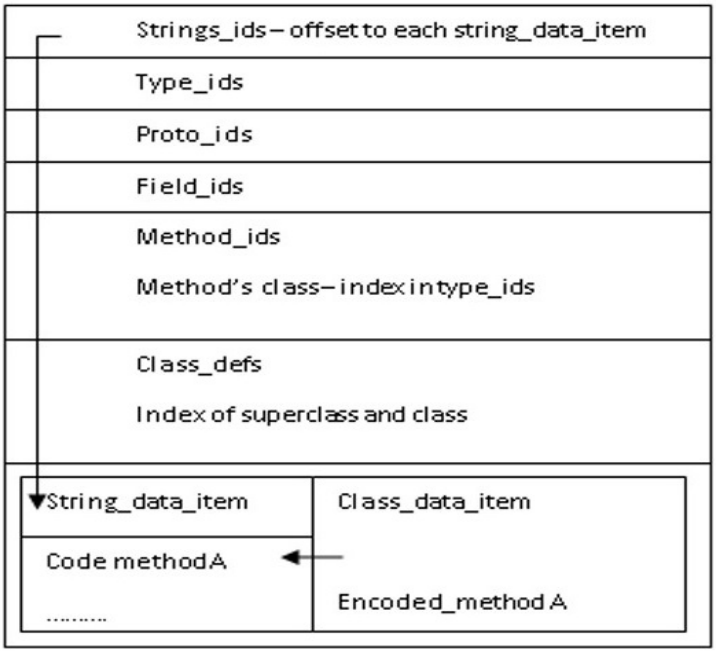

Figure 1. *Structure of a dex file*

The methods of hiding malware code are diverse and depend on the operating system features of the mobile terminal[6]. For example, the structure of a file (Figure 1) with the dex extension has areas assigned to Header, String_ids, Type_ids, Proto_ids, Fields, Methods, Classes and Data.

A class is found in the *class_defs matrix* in the form of an index that points to another index in the string *strings_ids*, the latter being connected to *string_data_item* that can return the class name. Each class defined in the code area is described by a class_data_item structure that contains its variables and methods. The methods are declared as a structure with the name encoded_method.

This structure consists of: *flag access* – such as method (public, private, protected, etc), *offset code* - the address where the method code is located from the beginning of the file *.dex* and *method_idx_diff* - an increment index for each method in the structure method_ids. To hide a method, the first step is to manipulate the *encoded_method* structure to reference another method, recalculate the SHA1 checksum, modify the .dex file header and package the application. Handling the *encode_method* structure involves a value for *method_idx_diff*, which can be 0 and the change of address that accesses the method code.

The camouflage technique used in cyberspace is identified by methods such as encryption, oligomorphism, polymorphism and metamorphism[7]. Semi-polymorphic or oligoorphic malware uses different encryption algorithms at each infection. The major difference between oligomorphism and polymorphism is that the latter can use an unlimited number of encryption algorithms. Metamorphism completely changes the malware code and does not use encryption algorithms. The mimicry type techniques can be identified in methods of obfuscation of the code. The most common methods of obfuscation are the use of junk code, variable permutation and replacement substitution, code transposition, and big code loop. When malware resides in hardware, the analysis process is much more difficult.

Integrated software platform for malware analysis of mobile terminals

In the first stage of development of malware software for mobile terminals, a series of necessary steps were identified. The first step involves identifying common and less common ways to infect mobile terminals with different security reports, and defining taxonomy by certain features such as attack vectors, source, targets, vulnerability exploited, threat type, etc.

In the second step, open-source or commercial applications were tested to choose solutions that meet security requirements. Various research projects and scientific papers were studied that relate to the detection of suspicious behavior and a customized firmware concept was proposed to improve the operating system. Among the tested solutions we could mention the Cellebrite UFED Pro Series, Cellebrite UFED Field, Cellebrite UFED Analytics, Oxygen Forensics, BlackBag Technologies, Forensic Toolkit, EnCase Forensic Software, Belkasoft Evidence Center, Autopsy, Computer Aided Investigation Environment, Mobile Security Testing Live Environment, MOBILedit, etc. A series of malware detection frameworks were installed and/or tested, such as: MODELZ[8], Andromaly[9], MADAM[10], ComDroid[11] and ProfileDroid[12]. The analyzed frameworks use different features of mobile terminals. MODELZ analyzes the power consumed by the battery when running different applications and based on these features identifies a signature. In my opinion, the main drawback of this analysis is the need to implement an external device to acquire the history of energy consumption in a precise way. The authors test on an external oscilloscope, Agilent Infinium 54851-A, and suggest building an inexpensive external circuit based on an Atmel AVR microcontroller. The Andromaly Framework uses an application installed on the mobile device that exploits various parameters such as CPU usage, number of Wi-Fi packs, number of processes running, battery level to deduce the normal operation of the device. The number of tracking features is 88.

The use of artificial intelligence algorithms for grading on a large number of features extracted from the mobile terminal, some of them redundant or irrelevant, suffers from several issues, such as: inefficiency of the learning algorithm, overuse, reduced generality, increasing model complexity and time execution. In our opinion, the application that implements classification algorithms should not run on mobile devices because they are often restricted by data storage and processing capabilities,

as well as battery power. The process of detecting malicious behavior becomes cumbersome when some malicious activities are of short duration and do not provide sufficient data to detect or engage the model, there is no possibility of accessing an unlimited database of malicious applications to increase the accuracy of algorithms, malicious behavior of an application is generated by multiple attack vectors, so a classification is difficult and the small number of malicious applications used as inputs generates an imbalance problem. The major disadvantage of the MADAM framework, though it uses 13 features and has been tested on real mobile devices, is that the mobile terminal has to be rooted. This framework monitors system calls, running processes, memory and CPU usage, called phone numbers, Bluetooth and Wi-Fi functionality, incoming or outgoing SMS, idle and activity times, key presses. The Droid Detective Framework[13] proposes an analysis based on the grouping of permissions for malware detection. After permissions are extracted, their occurrence frequency will be calculated when grouped (permission grouping starts from a permission to a group of 6 permissions) on safe and malicious applications. The group of permissions that indicates malicious behavior is identified for applications that use the features: ACCESS_NETWORK, READ_PHONE_STATE, INTERNET, READ_SMS, and WRITE_SMS. A number of authors[14] propose using several classification algorithms to improve the accuracy of malware detection. Multiple sets of features in the learning phase are used: API functions, permissions and commands of the SO. The algorithms used are: Decision Tree, Simple Logistic, Naive Bayes, Partial Decision Tree and Ripple Down Rule learner. The total number of selected features is 179, of which 125 permissions and 54 API functions and OS commands. As an input data set, the McAfee database with 2925 malicious applications and 3938 secure applications was used. To evaluate the performance of the classification algorithms, the 10-fold cross validation method is used (involving partitioning the 10-part initial set, 9 training on one and testing one, repeating the procedure, and checking the accuracy). RIDOR and PART have the best detection rate. Authors have a complete approach because they use sets of different features simultaneously with varied classification algorithms. The way to select the relevant features is not specified.

An interesting approach[15] is to analyze the permissions required by the application during execution and those in the manifest file. A permission that is not required in the initial phase may be required later. The idea is that there may be a difference between the required permissions and those used by the application. The bottom line is that malware requests more permissions than secure applications.

It is possible to build a classifier based on the set of lower level instructions using the N-gram model[16]. As a working procedure, the application is disassembled to generate smali type files. Each file contains a class of related methods in the Dalvik bytecode format. Disassembling an application is performed with the apktool utility (Figure 2). The instructions of each method are extracted from the resulting files into a string and their occurrence frequencies are calculated. Each Dalvik bytecode format has 1-byte size. The instruction number is 256 at 130, of which 218 instructions are used. There are 218 possibilities to arrange these instructions. The unique n-opcode number is calculated by the formula: N = X - (N - 1), where X is the number of instructions in the application and N represents the number of instructions in a pair. Thus, a 10-instruction method has 10 pairs of 1 instruction, 9 pairs of 2 instructions, 8 pairs of 3 instructions, etc.

Classification of malware can be done after a small set of instructions[17], respectively 6 instructions. These are: move, jump, packed-switch, sparse-switch, invoke, if. The initial premise is based on two questions: are the features chosen to distinguish between malicious and secure applications?

Furthermore, does the combination of the chosen features bring added value to the case when they are individually approached in malware analysis? The scientific contribution can be summarized as follows: the uniqueness of the chosen characteristics with good results using few resources for malware analysis. The significant difference that identifies malware is given by the move and jump instructions. The if and invoke instructions do not bring significant differences. The basic idea is that malware does not implement an application logic as complex as secure applications.

Due to the fact that iOS is closed, the security challenges are fewer. It allows revocation/ providing the identification of mobile terminals technical features, to include installed applications,

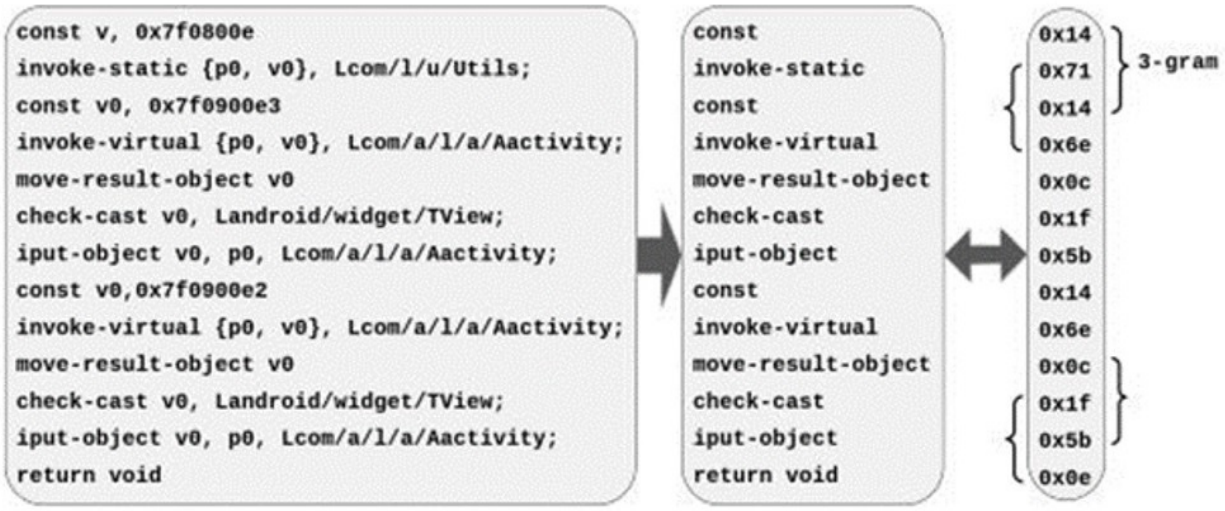

Figure 2. *Extract from a smali file the instructions and generate the 3-gram vector*

acceptance of dynamic permissions, executes ARM binary code that is difficult to disassemble, packing content is a tedious process with dex files. One of the attack vectors is the use of private API calls in applications.

The platform architecture was modularly designed so that it can integrate forensic software tools without compatibility issues (Figure 3). Each module performs specialized tasks as follows:

Web Central Platform

1. User Interface Module – This module performs management activities for the investigation cases, producing both dynamic and static security reports and, assigning risk scores for mobile terminals based on specific analysis and evaluation.

2. Authentication /Authorization Module. This module manages the authentication privileges for defined users as well as the access to the Web Central Platform.

3. Parameterization Module. This module manages the nomenclatures and provides the means to configure the parameters of the Web Central Platform.

4. Data Collection Module. The module gathers and disseminates the data, the results of analyses as well as contamination indices and specific mobile terminal applications.

5. Forensic Management Module. The module manages the forensic work tools and procedures, collecting mobile terminal artefacts and supporting the forensic analysis process for web services.

6. Monitoring Module. This module performs as a push Agent, analysing and evaluating all the applications installed on the mobile terminals, producing lists of suspicious applications and subsequently loading those suspicious programs as well as loading alerts, status and key performance indicators. This module is designed to enlarge the spectrum of malware threats identification by monitoring the behaviour of the mobile terminal installed applications and transmitting the results obtained to the central application responsible for Data Collection and Analysis.

7. Reverse Engineering Module. This module features reverse engineering capabilities, performing uploads and downloads of specific programs to be analysed as well as the analysis products.

8. Online Behaviour Integration Module. This module is directly connected to the Online Behaviour Analysis Module, to which it transmits the updated AI/ML algorithms, and from which it receives the results of online behavioural analysis for further processing.

9. Online Behaviour Analysis Module. The module features a web administrative interface, providing various capabilities such as Proxy, SSL Termination, VPN, Wireless Access Point, USB and Ethernet. The module records the traffic data produced when the mobile terminal is connected to

the Web Central Platform through Wi-Fi. Through network analysis, the module provides intrusion prevention services, runs AI/ML algorithms, detects applications anomalous behaviour caused by malware and transmits those traffic anomalies to the Online Behaviour Integration Module for further processing. Lastly the module profiles the mobile terminal in correlation with the default configurations and recorded traffic, acquiring lists with web sites classified as hazardous, accessing and integrating online threat intelligence sources.

mobile applications, manages alerts, supports the detection of malware based on signature lists, profiles the mobile terminal, correlating the default configuration with the installed applications.

12. Reverse Engineering Applications Module. Through a Sandbox type system, this module is designed to provide static and dynamic analysis of mobile terminals. It submits JSON/HTML reports to the web central platform, assesses the results of analysis, and identifies automatically the behaviour of specific malware.

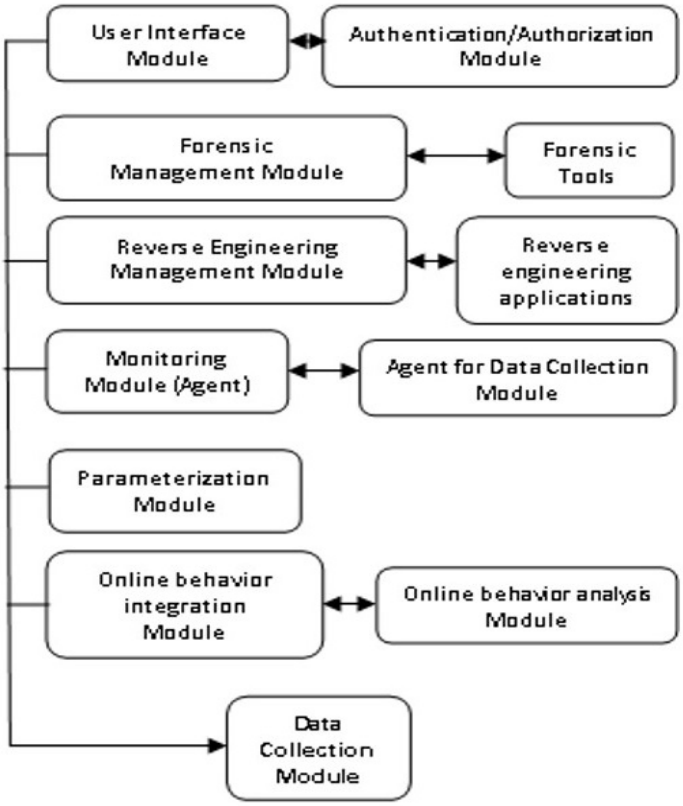

Figure 3. *Integrated software platform architecture*

10. Forensic Tools Module. The module displays a user-friendly interface. It enables parallel extraction and analysis from multiple mobile terminals and performs various tasks such as the physical and logical extraction of data, subsequent analysis of data, report configuration for data extraction, advanced analysis of mobile applications, as well as password and file recovery functionalities.

11. Agent for Data Collection Module. The agent gathers data and mobile applications, monitoring several security features. Thus, it collects key performance indicators, comparing hashes received from apps stores with hashes of installed

The Proxmox KVM server virtualization server was used for the necessary functionalities on which the virtual machines that support various services and applications are created. Virtual machines use Docker containers that are orchestrated by Kubernetes. LXD, which is a new generation system, has been used to handle containers and provides API REST services.

Using containers allows the creation of micro-services – applications are thus decoupled and can be installed and managed dynamically. The virtual machine VM3 is responsible for the PostgreSQL database for running the torsim-database service. The Apache Kafka and Elasticsearch tools are

Table 1. *Fluxes between applications and services*

No	Applications and services		
	Service/application	*Virtual machine*	*Role*
1	nginx	VM 1	The Front end receives requests from customers
2	torsim-proxy	VM 1	The application's graphical interface and secure services receive requests from the front end (nginx)
3	torsim-bro-logtail	VM 2	It takes the traffic recorded by the car and saves it in a queue from Kafka
4	torsim-message-processor	VM 1	It takes requests from torsim-proxy
5	MobSF, CuckooDroid	VM 1	It takes requests from torsim-proxy
6	torsim-adb	VM 1	The torsim-adb API takes requests from torsim-proxy
7	Torsim-database	VM 3	The torsim-database API receives requests from torsim-adb and torsim-proxy
8	PostgreSQL	VM 3	It takes requests from the torsim-database container
9	Elasticsearch	VM 3	It takes requests from the torsim-message-processor
10	Kafka	VM 3	It takes requests from the torsim-message-processor

also installed in this virtual machine. For the forensic functionality, the ADB utilities – Android Debug Bridge and MOBILedit – were used. The torsim-adb service encompasses the ADB client, communicating with the ADB server installed on the laptop, and communicating with the ADB deamon on the mobile terminal. At the moment, integration with MOBILedit is done at the procedural level, MOBILedit manually runs and the desired ratio is obtained, which then loads into the central platform. In order to intercept the traffic generated by the mobile devices, the Bro utility sends the intercepted packets into a Kafka message queue, which is then taken over by the torsim-message processor service and sent to Elasticsearch. For Trafficking Analysis and Malicious Behavior, Mail trail Maltese Traffic Detection System is used. The Mail trail application uses public lists of trusted and malicious sites, information from reports of various antivirus products, custom lists where signatures can be domain names, IPs, HTTP User-Agent header value, and heuristic mechanisms that can help detection of malware unknown yet. On the virtual machine VM 1 CuckooDroid is installed, an extension of Cuckoo Sandbox, an open source software used to analyze suspicious files with capabilities in static and dynamic analysis of Android apps. The MobSF framework also enables static and dynamic analysis of mobile applications. A Docker container was used for the Static Analysis MobSF application, and the integration with the central platform was done through the MobSF API, so applications can be submitted for analysis and both the PDF report and the JSON format are obtained. The latter is used to store scan data in the database and to be displayed in the web interface. An important step was testing the platform by verifying all the parameters introduced and obtaining the right reports in malware analysis. Agent mode is still in the development phase and will be supported by Android and IOS and a number of artificial intelligence algorithms on different features. In our opinion, an algorithm that tracks malicious behavior only on the basis of

permissions has a low efficiency. The agent must run on no rooting phones and track the required permissions for applications installed before and during running applications. The following artifacts can be tracked: accessing network and sensitive data such as contact list and location, receiving and transmitting SMS, clipboard data, access to different hardware components, number of clicks during the user's intense activity period correlated with the period of inactivity. The agent can be integrated with the public API and made available by VirusTotal to check the authenticity of the apk package by comparing the application's hash with the site's database. Check the application configuration files to identify the version of the application, the hardware resources it will require, the permissions to be assigned, the components and the list of dangerous permissions. The existence of suspicious character strings in the application may be an indicator of the presence of a malware infection. Entropy detects if there are encrypted areas.

Conclusions

New advances in Artificial Intelligence – Machine Learning have allowed the emerge of a new stage in evolution of cyber security. Continuously improving the possibilities of identifying and combating future threats is a viable solution to the fight against malware. Literature studied during the project period included only algorithms from supervised learning. Various mobile terminal security software solutions have been tested and hardware and software infrastructure built. The research project is not completed, following the testing phase of malware applications selected by project experts.

Acknowledgment

This work was possible with the financial support of the Executive Agency for Higher Education, Research, Development and Innovation Funding – UEFISCDI / Romanian Ministry of National Education, under the project number PN-III-P2-2.1-SOL-2016-05-0070 with the title "Integrated Software Platform for Malware Analysis on Mobile Terminals".

NOTES:

1 James R. Blake, "Transforming military", Praeger Security International, May 2007.

2 https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

3 Elena Şuşnea, Adrian Iftene, "The Significance of Online Monitoring Activities for the Social Media Intelligence (SOCMINT)", Conference on Mathematical Foundations of Informatics MFOI'2018, Institute of Mathematics and Computer, Chisinau, Moldova, pp. 230-240, 2018.

4 Reza Hedayat, Lorenzo Cavallaro, *The Devil's Right Hand: An Investigation on Malware-oriented Obfuscation Techniques*, Computer Weekly, August 2016.

5 Dragoş Bărbieru, Alexandru Stoica, "Malware Analysis on Mobile Phone", The International Scientific Conference eLearning and Software for Education, Vol. 4, 11-15, "Carol I" National Defence University, Bucharest, pp. 11-15, 2018.

6 https://fortiguard.com/events/755/2013-10-25-playing-hide-and-seek-with-dalvik-executables

7 Babak Bashari Rad†, Maslin Masrom ††, Suhaimi Ibrahim, *Camouflage in Malware: from Encryption to Metamorphism,* IJCSNS International Journal of Computer Science and Network Security, Vol.12, No.8, August 2012.

8 Hannsang Kim, Member IEEE, Kang G. Shin, Padmanabhan Pillai, *MODELZ: Monitoring, Detection and Analysis of Energy-Greedy Anomalies in Mobile Handsets,* IEEE Transactions on mobile computing, Vol. 10, July 2011.

9 Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, Yael Weiss, *"Andromaly": a behavioral malware detection framework for android devices*.

10 Gianluca Dini, Fabio Martinelli, Andrea Saracino, Daniele Sgandurra, *MADAM: a Multi-Level Anomaly Detector for Android Malware*, Computer Network Security: 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2012, St. Petersburg, Russia, October 17-19, 2012.

11 Chin E., Felt A. P., Greenwood K., and Wagner D.: "Analyzing inter-application communication in Android". Proc. 9th Int. Conf. On Mobile Systems, Applications, and Services (MobiSys '11). ACM, Washington, DC, USA, June 2011, pp. 239-252.

12 Wei X., Gomez L., Neamtiu I., and Faloutsos M.: "ProfileDroid: multi-layer profiling of android applications" Proc. 18th Int. Conf. On Mobile Computing and Networking (Mobicom '12), ACM, Istanbul, Turkey, August, 2012, pp. 137-148.

13 Shuang Liang; Xiaojiang Du, *Permission-combination-based scheme for Android mobile malware detection,* IEEE International Conference on Communications (ICC), June, 2014.

14 Suleiman Y. Yerima, Sakir Sezer, Igor Muttik, *Android Malware Detection Using Parallel Machine Learning Classifiers*, Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, December, 2014.

15 Xing Liu, Jiqiang Liu, *A Two-Layered Permission-Based Android Malware Detection Scheme*, 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, April, 2014.

16 BooJoong Kang, Suleiman Y. Yerima, Sakir Sezer, Kieran McLaughlin, International Journal on Cyber Situational Awareness, Vol. 1, No. 1, 2016, pp. 231-255.

17 Gerardo Canfora, Francesco Mercaldo, Corrado Aaron Visaggio, *Mobile malware detection using op-code frequency histograms*, 12th International Joint Conference on e-Business and Telecommunications (ICETE), July, 2016.

BIBLIOGRAFIE

Babak Bashari Rad, Maslin Masrom, Suhaimi Ibrahim, "Camouflage in Malware: from Encryption to Metamorphism", *IJCSNS International Journal of Computer Science and Network Security*, vol. 12, No. 8, August, 2012.

Bărbieru Dragoş, Stoica Alexandru, "Malware Analysis on Mobile Phone", *The International Scientific Conference eLearning and Software for Education*, Vol. 4, "Carol I" National Defence University, Bucharest, 2018.

Blake R. James, *Transforming military*, Praeger Security International, May, 2007.

Canfora Gerardo, Mercaldo Francesco, Visaggio Corrado Aaron, "Mobile malware detection using op-code frequency histograms", *12th International Joint Conference on e-Business and Telecommunications (ICETE)*, July 2016.

Chin E., Felt A.P., Greenwood K., Wagner D., "Analyzing inter-application communication în Android", *Proc. 9th Int. Conf. On Mobile Systems, Applications, and Services (MobiSys '11). ACM*, Washington, DC, USA, June, 2011.

Dini Gianluca, Martinelli Fabio, Saracino Andrea, Sgandurra Daniele, "MADAM: a Multi-Level Anomaly Detector for Android Malware", *Computer Network Security: 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2012*, St. Petersburg, Russia, October, 17-19, 2012.

Hannsang Kim, Member IEEE, Kang G. Shin, Padmanabhan Pillai, "MODELZ: Monitoring, Detection and Analysis of Energy-Greedy Anomalies în Mobile Handsets", *IEEE Transactions on mobile computing*, Vol. 10, July, 2011.

Hedayat Reza, Cavallaro Lorenzo, "The Devil's Right Hand: An Investigation on Malware-oriented Obfuscation Techniques", *Computer Weekly*, August, 2016.

Kang BooJoong, Yerima Y. Suleiman, Sezer Sakir, McLaughlin Kieran, *International Journal on Cyber Situational Awareness*, Vol. 1, No. 1, 2016.

Shabtai Asaf, Kanonov Uri, Elovici Yuval, Glezer Chanan, Weiss Yael, *Andromaly: a behavioral malware detection framework for android devices*.

Shuang Liang, Xiaojiang Du, "Permission-combination-based scheme for Android mobile malware detection", *IEEE International Conference on Communications (ICC)*, June, 2014.

Şuşnea Elena, Iftene Adrian, "The Significance of Online Monitoring Activities for the Social Media Intelligence (SOCMINT)", *Conference on Mathematical Foundations of Informatics MFOI'2018*, Institute of Mathematics and Computer, Chisinau, Moldova, 2018.

Wei X., Gomez L., Neamtiu I., Faloutsos M., "ProfileDroid: multi-layer profiling of android applications", *Proc. 18th Int. Conf. On Mobile Computing and Networking (Mobicom '12). ACM*, Istanbul, Turkey, August, 2012.

Xing Liu, Jiqiang Liu, "A Two-Layered Permission-Based Android Malware Detection Scheme", 2nd *IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, April, 2014.

Yerima Y. Suleiman, Sezer Sakir, Muttik Igor, "Android Malware Detection Using Parallel Machine Learning Classifiers", *Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, UK, Oxford, December, 2014.

https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

https://fortiguard.com/events/755/2013-10-25-playing-hide-and-seek-with-dalvik-executables