Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

DFRWS 2018 USA — Proceedings of the Eighteenth Annual DFRWS USA

Multinomial malware classification via low-level features

Sergii Banin [a], [*], Geir Olav Dyrkolbotn [a], [b]

[a] Department of Information Security and Communication Technology, NTNU, Gjøvik, Norway
[b] Norwegian Defence Cyber Academy (NDCA), Jørstadmoen, Norway

Keywords:
Information security
Malware detection
Malware classification
Multinomial classification
Low-level features
Hardware activity

A B S T R A C T

Because malicious software or (”*malware*”) is so frequently used in a cyber crimes, malware detection and relevant research became a serious issue in the information security landscape. However, in order to have an appropriate defense and post-attack response however, malware must not only be detected, but also categorized according to its functionality. It comes as no surprise that more and more malware is now made with the intent to avoid detection and research mechanisms. Despite sophisticated obfuscation, encryption, and anti-debug techniques, it is impossible to avoid execution on hardware, so hardware (“*low-level*”) activity is a promising source of features. In this paper, we study the applicability of low-level features for multinomial malware classification. This research is a logical continuation of a previously published paper (Banin et al., 2016) where it was proved that memory access patterns can be successfully used for malware detection. In this research we use memory access patterns to distinguish between 10 malware families and 10 malware types. In the results, we show that our method works better for classifying malware into families than into types, and analyze our achievements in detail. With satisfying classification accuracy, we show that thorough feature selection can reduce data dimensionality by a magnitude of 3 without significant loss in classification performance.

1. Introduction

Malware detection is an important part of information security. Recently there were several major cyber attacks that influenced power grids, banking and transportation systems, manufacturing facilities and so on Reuters (2017), The Verge (2017) and all of them used malware for achieving their final goals. Despite the use of antivirus solutions, complicated anti-detection techniques allowed adversaries to avoid defense mechanisms. This fact points out a need for improvements in malware detection.

Malware is used for different purposes: to show ads to users, spread spam, track user activity, steal data, create backdoors and so on. Malware is often not created with a single specific purpose, but rather as a part of more advanced threats. APT or Advanced Persistent Threat is a human being or organization (WAMPTY Enterprise) that operates a campaign of intellectual property theft, the undermining of a company's or country's operations through stealthy, targeted, adaptive and data focused (Cole, 2012) attack techniques. *Something* has to exploit a victim's weaknesses, *something* has to aid in the installation of persistence tools, *something* has to communicate with command and control servers, and *something* has to perform actions in the victim system. Even though specific actions might be launched manually from the command and control server, they may rely on remote access trojans and backdoors (Rudd et al., 2017) present in the victim system. As we can see, malware could be used for different purposes and goals.

Because of the variety of malware functionality, it is important not only to detect malice (*malware detection*), but to differentiate between different *kinds* of malware (*multinomial malware classification* or *malware classification*) in order to provide better understanding of malware capabilities, describe vulnerabilities of systems and operations as well as to use appropriate protection and post-attack actions.

Malware *classification* or *categorization* is a common problem that is analyzed in many research articles (Tabish et al., 2009; Sathyanarayan et al., 2008). There are two widely used malware categorization approaches: *malware types* and *malware families*. However, literature studies show that authors rarely provide proper definitions of these terms. This can lead to the various misunderstandings and non-valid comparisons. E.g. in Tabish et al. (2009), authors mention *viruses, backdoors, trojans* etc. while talking about classifying *malware types* and *families*. Another example

* Corresponding author.
E-mail addresses: sergii.banin@ntnu.no (S. Banin), geir.dyrkolbotn@ntnu.no (G.O. Dyrkolbotn).

https://doi.org/10.1016/j.diin.2018.04.019

of inconsistent terminology can be found in Sathyanarayan et al. (2008). In this paper, authors claim that their system is capable of detecting the *malware families* (in their case *trojans, backdoors, worms*). Nevertheless, they compare their results to the results from other papers where research was done on the *malware types*. Authors of Saeed et al., (2013) attempted to elaborate on the definition of the term *malware*; however, later on they use term *malware family* when talking about *viruses, trojans, worms* and other *malware types*. It might happen, that the use of inconsistent terminology is more common among academics and not malware analysis practitioners. Therefore, we must emphasize, in this paper that we use the following definitions created after reviewing descriptions of malware categories provided by well-known vendors (e.g. Microsoft, Symantec etc):

Malware type is assigned according to general functionality.
Malware is grouped into a **malware family** according to its particular functionality.

Where general functionality is about *what* malware does (which goals it pursues), and particular functionality is about *how* malware acts (which methods it uses in order to achieve its goals).

As it appears, it is insufficient to know that some malware is affecting operations: knowledge about its category (*family* or *type*) can aid in restoring a system's state as well as in developing new security mechanisms to prevent similar problems in the future. This necessitates standard definitions of different malware kinds and methods that allow the effective categorization of detected malware.

To avoid detection, malware creators develop additional evasive methods to thwart detection by antimalware software. They utilize various obfuscation techniques such as metamorphism, polymorphism, encryption, dead code insertions, and instruction substitution (Schiffman, 2010). Such methods allow altering the appearance of a file and its static characteristics. The basic example is changing hash sums (such that SHA-1 or md5) used as file signatures by means of changing different strings in the file. Moreover, dead code insertions will change opcode sequences in the executable, making detection more difficult.

There are two main ways to perform malware analysis which are widely used and described in the literature (Distler and Hornat, 2007; Kendall): *static* and *dynamic*. *Static analysis* is performed without execution of a malicious file. The main purpose of this approach is to collect different static properties: bytes, opcodes and API n-grams frequencies, properties of Portable Executable header, strings (e.g. commandline commands, URLs etc) and others (Schiffman, 2010; Uppal et al., 2014). *Dynamic analysis* is done by executing malware in a controlled environment (a virtual machine or emulator) and recording actions it has done in the system. These include patterns of a registry, network and disk usage, monitoring of API-calls, tracing of executed instructions, investigation of memory layout and so on (Egele et al., 2012). Specialized sandboxes like Cuckoo (Cuckoo Sandbox, 2015) or other Virtual Machines can be used. They might be assisted by a debugger or other tracing software. Some authors assume (Egele et al., 2012; Prakash et al., 2015) disk and network activities are essential for malware detection, but few authors explored the capabilities of memory properties analysis (Kawakoya et al., 2013; Khasawneh et al., 2015).

Though malware creators use a variety of sophisticated evasive techniques (Rudd et al., 2017), it is impossible to avoid execution on the system's hardware. Earlier low-level (or hardware) activity has proven to be efficient in malware detection (Banin et al., 2016). In this paper, we use a similar technique for multinomial malware classification. Achieved results and findings will be used in future work, where combinations of high- and low-level activity will be used for malware categorization according to the specific context.

In this paper, we use sequences of memory access operations generated by a set of malicious executables as a source of features for machine learning algorithms. We apply dynamic analysis inside the virtualized environment as it is a safe (we don't let real malware samples spread outside of our environment) and time-efficient solution (experiments on physical machines would take significantly longer). We find the best features for distinguishing between ten predefined malware families and ten types. However, our models should be simple enough so that we can build a connection between low-level and high-level activity in the future work. Therefore, we may choose less accurate but simpler models to make analysis easier. Our initial hypothesis predicts that since malware types and families have a valuable difference in high-level behavior, we might be able to find distinctive low-level behavior patterns among malware categories. In the future work, we will test our models on the dataset of newer malware in order to check their capabilities against previously unknown (as for the models) malware. Our second hypothesis is that since malware families are assigned according to their particular functionality (e.g. exploiting of a certain vulnerability), they might generate more explicit activity that allows distinguishing better between families than between types.

The remainder of the paper is arranged in the following order: Section 2 contains State of the Art, Section 3 describes our methodology, Section 4 describes our results, Section 5 presents analysis of the results achieved, Section 6 presents a series of short remarks, conclusions, and a projection of future work.

2. State of the art

As was written above, in order to perform appropriate counteractions (to prevent) or postactions (to recover), we need additional information about malware category. With knowledge about malware types, we can apply appropriate defense mechanisms: e.g. in order to protect against Ransomware, we should keep an up-to-date backup of the data, while defense against self-replicating (Viruses) malware could be implemented with a thorough managing of a network traffic and removable media. In addition to knowledge about malware type, knowledge about malware family can help to set up appropriate defense mechanisms. Moreover, information about malware family can serve well in incident response actions: proper definition of malware family points to the potentially affected system components.

Many authors have performed research on malware classification. Different techniques and features are used to classify unknown malware into known malware categories or to detect outliers and perform a thorough analysis of such anomalies. For example, the authors of Kong and Yan (2013) combined different types of malware attributes (opcodes, API calls, flags, registers etc) in order to classify malware into 11 families. They used discriminant distance metric learning and pairwise graph matching in ensembled classifier to create an efficient framework that is capable of detecting previously unknown samples. Authors of Tian et al (2008) used a length of functions for classifying Trojans into 7 different families. They created pretty fast ($O(n)$ training and classification time) and relatively accurate (around 80% average accuracy) method for malware classification. They also warn, that their approach might not be as successful on other malware types such as Viruses, where malicious code is difficult to extract. The same authors in their newer paper (Tian et al., 2010) used API calls and their parameters as features for malware detection, and classification of 10 malware families. They managed to achieve up to 97% accuracy in malware detection, and up to 95% accuracy in malware classification.