Available online at www.sciencedirect.com

ScienceDirect

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

Computers & Security

Probabilistic analysis of dynamic malware traces

Jan Stiborek [a,b,*], Tomáš Pevný [a,b], Martin Rehák [a,b]

[a] Cisco Systems, Inc., 170 West Tasman Dr., San Jose, CA 95134, USA
[b] Department of Computer Science, FEE, CTU in Prague, Karlovo náměstí 13, 121 35, Praha, Czech Republic

ARTICLE INFO

Article history:
Received 10 May 2017
Received in revised form 8 December 2017
Accepted 15 January 2018
Available online

Keywords:
Malware
Clustering
Dynamic analysis
Probabilistic modeling
Multiple instance learning

ABSTRACT

We propose a method to automatically group unknown binaries executed in sandbox according to their interaction with system resources (files on the filesystem, mutexes, registry keys, network communication with remote servers and error messages generated by operating system) such that each group corresponds to a malware family. The method utilizes probabilistic generative model (Bernoulli mixture model), which allows human-friendly prioritization of identified clusters and extraction of readable behavioral indicators to maximize interpretability. We compare it to relevant prior art on a large set of malware binaries where a quality of cluster prioritization and automatic extraction of indicators of compromise is demonstrated. The proposed approach therefore implements complete pipeline which has the potential to significantly speed-up analysis of unknown samples.

1. Motivation

With the number of new malware samples reaching 120 million in 2016 (AV-Test GmbH, 2017), malware poses serious threat to computer security. Despite the existence of automatic methods for classification of unknown samples, a large portion still requires human analysis. Our work simplifies the work of malware analysts and incident responses by grouping malware into coherent groups according to its behavior.

Traditionally, malware analysis relied heavily on *static analysis*. Its main advantage is the low computational complexity since analyzed binaries do not need to be executed or instrumented. However, due to evasion techniques like polymorphism, obfuscation, encryption, etc., it is increasingly more difficult to keep good recognition rate.

An alternative approach is *dynamic analysis*, which executes the analyzed binary in a controlled environment (sandbox) (Oktavianto and Muhardianto, 2013) and monitors its behavior. The presumed advantage is that the behavior should be more difficult to conceal and therefore it should constitute a more robust signal. A majority of approaches to dynamic analysis relies on analysis of system calls (Ahmadi et al., 2016; Naval et al., 2015; Wüchner et al., 2014), as they are the only means how the binary can interact with operating system and its resources (files, network connections, mutexes, etc.). However, the popularity of this strategy has already triggered the development of evasion techniques such as shadow attacks (Ma et al., 2012), system-call injection attacks (Kc et al., 2003), or sandbox detection (Garcia). Finding new or improving existing approaches for malware analysis is therefore important and makes the evasion more difficult.

This work postulates that the monitoring of system calls is not the only means to gather information about malware's behavior. To provide revenue to its owner, malware has to perform actions such as showing advertisements, encrypting

* *Corresponding author.*
E-mail address: jastibor@cisco.com (J. Stiborek).

https://doi.org/10.1016/j.cose.2018.01.012
0167-4048/© 2018 Elsevier Ltd. All rights reserved.

hard drive, stealing data, etc., which can be observed directly through monitoring of interactions with system resources such as files, kernel and system structures (e.g. mutexes, semaphores or registry keys), communication with network resources and error messages reported by operating system. Although each individual action may not be informative enough, their combination is surprisingly strong as has been already demonstrated in Mohaisen et al. (2015) and Rieck et al. (2011). Moreover, such monitoring is possible without direct integration into controlled environment which increases the robustness of the analysis.

This work extends the support for analysts by automatic recognition of similar binaries and putting them into one group (clustering), automatically extracting humanly understandable description of each group, and recommending group of samples for manual analysts. These goals are approached by using a generative model of interactions of the executed binaries with files, operating system structures (mutexes), network resources, registry keys and of error messages of the operating system. The key differences to the prior art and the contributions of this paper are (i) a wider spectrum of modeled resources, (ii) novel prioritization of identified groups of malware samples, and (iii) automatic extraction of *behavior indicators* (BI) in form of short examples of characteristic interactions with system resources.

The proposed model is evaluated on a corpus of 130 000 malware binaries and compared to a state-of-the-art approach for behavior clustering.

2. Related work

Since the analysis of malicious binaries and recommending them for further analysis has important practical applications, there exists a rich prior art. Although it is frequently divided into two categories, static and dynamic analysis, the boundaries between them are blurred since techniques such as analysis of the execution graph are used in both categories.

2.1. *Static malware analysis*

Static malware analysis treats a malware binary file as a data file from which it extracts features without executing it. The earliest approaches (Lo et al., 1995) looked for a manually specified set of specific instructions *(tell-tale)* used by malware to perform malicious actions but not used by legitimate binaries. Later works, inspired by text analysis, used *n*-gram models of binaries and instructions within (Li et al., 2005). Malware authors reacted quickly and began to obfuscate, encrypt, and randomize their binaries, which rendered such basic models (Sharif et al., 2008) useless. Since reversing obfuscation and polymorphic techniques are in theory NP-hard (Moser et al., 2007), most of the recent state of the art (Ahmadi et al., 2016; Christodorescu and Jha, 2006; Sharif et al., 2008) moved to a higher-level modeling of sequences of instructions / system calls and estimating their action or effect on the operating system. The rationale behind is that higher-level actions are more difficult to hide.

2.2. *Dynamic malware analysis*

An alternative solution to analyzing obfuscation and encryption is the execution of a binary in a controlled environment and analyzing its interactions with the operating system and system resources.

A large portion of the work related to dynamic malware analysis utilizes system calls, since in modern operating systems, system calls are the only way for applications to interact with the hardware and as such the calls can reveal malware actions. The simplest methods view a sequence of system calls as a sequence of strings and use histograms of occurrences to create feature vectors for the classifier of choice (Hansen et al., 2016). The biggest drawback of these naive techniques is low robustness to system call randomization. Similarly to static analysis, this problem can be tackled by assigning actions to groups (clusters) of system calls (syscalls) and using them to characterize the binary (Naval et al., 2015) (Wüchner et al., 2014). One such example that uses analysis of the system calls for malware clustering was proposed by Bayer et al. (2009). Authors taint certain portions of memory, such as output arguments and output values of system calls, and track all operations with the tainted memory to generate traces of system calls. This allows to uncover dependencies between individual system calls even when they are interleaved with unrelated ones and provides information necessary for creating behavioral profile of the analyzed binary. These profiles are then clustered with an algorithm based on locality sensitive hashing.

A wide class of methods identifying malware samples from sequences of syscalls rely on *n*-grams (Lanzi et al., 2010; O'Kane et al., 2013). Malheur (Rieck et al., 2011) uses normalized histograms of *n*-grams as feature vectors, which effectively embeds syscall sequences into Euclidean space endowed with L_2 norm. In this space the algorithm extracts prototypes $Z = \{z_1, \ldots, z_n\}$ using hierarchical clustering. Each prototype captures behavior of the cluster, which should match corresponding malware family. An interesting feature of Malheur is that if a cluster has less than a certain number of samples, the prototype is not created.

AMAL (Mohaisen et al., 2015) uses custom sandbox to intercept and log interactions of the malware binary with files and registry features and its communication over the network. From these interactions AMAL extracts high-level numeric features such as counts or sizes of created, modified or deleted files, counts of created, modified or deleted registry keys, counts of unique IP addresses, etc., and uses single-linkage clustering to identify similar binaries. Unlike AMAL, this work uses resource names instead of their numerical properties to construct its features. Moreover, the generative model allows to prioritize founded clusters and extract typical characteristics of each cluster.

Similarly to the presented model, Rieck et al. (2008) create a representation of analyzed samples without manually defined conversion of the input data, which consists of the names of system calls and its parameters. The calls are treated as words, specifically each system call name together with all its parameters corresponds to one word. To allow generalization, Rieck et al. creates $n + 1$ additional words from a syscall with n parameters by iteratively removing its last parameter. This causes