Cloud Based Malware Detection Through Behavioral Entropy

Kambiz Vahedi [1], Khadijeh Afhamisisi [2]
[1] *Faculty of Computer Science and Engineering,* [2] *Department of Electrical Engineering*
[1] *Shahid Beheshti University, Tehran, Iran,* [2] *Iran University of Science and Technology, Tehran, Iran*
[1] kambizvahedi@gmail.com, [2] afhami@iust.ac.ir

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671981

***Abstract*—Most of malware virus detection platforms rely on static techniques and checking for specific signatures files. In this paper we propose a cloud based malware detection method capable of monitoring behavioral characteristics of file; which subsequently results in some behavioral pattern that is accordingly classified into corresponding malware families. The proposed method significantly improves malware detection rates along with minimal false positives**

***Index Terms*—Cloud malware analysis, Behavioral entropy, Dynamic analysis, Behavioral features, File similarity.**

I. Introduction

There are computing architectures, some of which include cluster computing, grid computing and cloud computing. Cloud computing has a cloud-like structure that allows users to access applications, services, software and hardware resources from anywhere in the world. Cluster computing is currently less favored than the other two, with grid computing coming in the second place. Cloud computing comes in first rank and is by far more in use. Thus, the world of computing is rapidly moving towards the development software that is available to millions of consumers as a service instead of running on individual computers. In this study, we try to improve malware detection by using cloud computing resources. The main advantage of behavior-based malware detection methods is the ability to detect multi-modal malware that signature-based techniques cannot detect. On the other hand, long scan time is one of the main disadvantages of behavior-based malware detection methods [1].

As shown in Figure 1, storing the bank of malware behavioral signatures prepared on the cloud will improve the performance of detecting malware and reduces false positive alerts. We will inevitably need a larger bank to record malware signatures and behaviors, however, using cloud space, which has a larger volume for storage and far more processing power (e.g., elastic search) solves this problem. Sending large files to the cloud server would be time consuming and cause most users to stop using the cloud based antivirus section. Thus, it is more efficient to extract behavioral features and encode the behavioral features on the user side [2] and then send the behavioral signature to the cloud environment for comparison against the existing bank of malware signatures.

Most of dynamic analysis tools are capable of monitoring what system functions have been invoked by the sample file. Also, analyzing the parameters passed to system functions

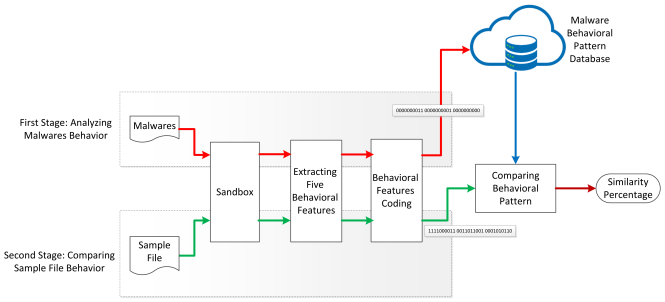

Fig. 1. The overview of cloud based malware detection method based on behavioral entropy.

allows for a semantic summary on multiple functions; which provides a report on the functionality and behavior of the sample file. The reports illuminate the behavioral characteristics of the malware; which results in some specific behavioral pattern and eventually clustering and classification of the malware in malware families. While the malware might avoid signature-based detection through obfuscation and metamorphism methods, yet, the dynamic analysis and extraction of behavioral pattern illustrated association to known malware families.

Malware detection methods based on similarity test can be classified into different categories, including basic test, edit distance, pair wise alignment, Hidden Markov Models, n-gram similarity, opcode graph-based similarity and structural entropy. Most of similarity test methods operate based on static analysis of malware through disassembling malware and measuring level of similarity between the sample file and known malware families. However, by injection of garbage code and/or replacing instructions with equivalent instructions towards changing the overall structure and subsequently lowering malware detection rates.

This study investigates the issue of detecting malware files through behavioral similarity and subsequently integration into a cloud based malware detection platform. The proposed system would employ the concept of behavioral entropy towards analyzing behavior of the file and then sending behavioral features onto the cloud for similarity test against known malware families. This involves two stages: (1) initially, extracting behavioral features through behavioral analysis of the behavior of the file and (2) comparing the extracted behavioral features. The behavioral analysis is performed inside a sandbox envi-

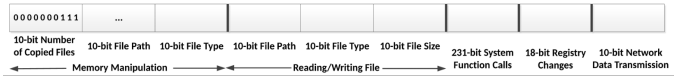

Fig. 2. Numerical string produced by encoding behavioral features.

ronment to help with extraction of behavioral features given the behaviors of the malware. Before proceeding with in-depth discussion of the proposed method, we will provide a brief review of literature to help with understanding similarity test methods and further pave the way for presenting similarity test based on behavioral entropy.

II. Previous Work

The opcode similarity graph is calculated based on weighted directed graph of opcode sequence [3]. Replacing instructions with equivalent instructions changes the opcode sequence and subsequently changes the graph layout; which adversely affects calculation of similarity. Moreover, inserting garbage code into malware changes weight of the edges in the graph, thus deteriorating efficiency of malware detection.

The Hidden Markov Model (HMM) involves two stages of training and detection. In the training stage, a hidden Markov model is produced over a malware family. In the detection stage, some threshold value is used for detecting nature of files and whether the malware is associated with an already known malware family. However, HMM is expensive both in terms of memory and computation-time. Moreover, training a sample malware in HMM requires multiple malware from the same family. In case there is not enough quantity of malware from the same family, then the method cannot be used for detecting future versions of the malware. Further, in case of malware apply replacement of instructions with equivalent instructions, HMM method would be incapable due to changes in the sequence of instructions.

In contrast with the HMM-based and opcode graph-based methods similarity test based on structural entropy is directly computed over the executable file. The method focuses on the order of code and data regions within a file, describing file through byte sequences and length of file. Static analysis of file through structural entropy forms the basis for similarity measurement, which involves the following two stages of file segmentation and comparing sequence of segments. In file segmentation stage entropy measurement and wavelet analysis are used for file segmentation. The second stage involves measuring similarity of files based distance between sequence of segments.

III. Behavioral Entropy Similarity Measurement

The proposed method, involves determining similarity of sample files through measuring behavioral entropy. The method involves two stages. In the initial stage extraction of behavioral features is performed in sandbox environment. The behavioral features in the form of qualitative strings are encoded and converted to quantitative numbers for the sake of comparison. Next, the produced numerical values are registered into the database.

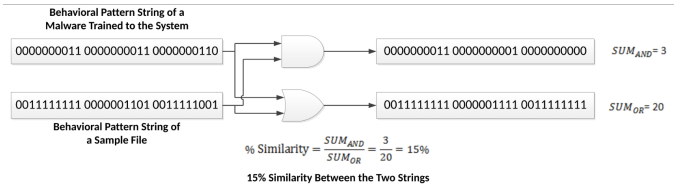

Fig. 3. Calculating similarity percentage of a sample file with a malware.

A. Hard disk manipulation:

- The number of files copied into the hard disk: a 20-bit string is allocated for representing quantity of files copied into the hard disk. The same principle applies to some other behavioral features in the following.
- File path in the hard disk: a 10-bit string is allocated for conventional paths used copying files into the hard disk, including paths of System32, Temp, AppData folders and etc. in the Windows OS. Each path is dedicated a single bit; and for each path that the file is copied into the corresponding bit is set to 1 and other bits remain 0.
- File type: a 10-bit string is allocated to illustrate the types of copied files with different extensions into the hard disk (i.e., exe, dll, ini, files without extension, etc.).

The three strings produced from number of files, file path and file type characteristics along each other constitute a 40-bit string of values.

B. File Read/Write:

- File path: a 10-bit string is used to illustrate the created file paths.
- File type: a 10-bit string for 10 common extensions including log, tmp, inf and etc.
- File size: a 10-bit string is used for representing various sizes; each bit denoting a size range (i.e., 0-2 KB, 2-5 KB, 5-10 KB, 10-50 KB, 50-100 KB, 100-500 KB, 500 KB-1 MB and values larger than 1MB and a single bit is used for representing anomaly in file sizes).

C. Invoking system functions:

a list, containing 231 system functions (APIs) invoked by the malware families divided into 21 categories based on their functionality and activities (e.g. operations transmitting data over the network, operations on processes).

D. Changing registry keys:

An 18-bit string for majority of registry keys changes (i.e., 9 keys for registering the malware into startup of the system, 2 keys for removing the program in Task Manager, 2 keys for elimination of security notifications, 2 keys for removing ”Folder Options” from ”Tools” menu within Windows Explorer and 3 keys for hiding file extensions, not showing hidden files and changing the icon for executable files, respectively).

E. Data transmission over the network:

a 10-bit string is allocated to represent data transmitted over the network in several ways including HTTP(S), FTP, SMTP and etc.

The behavioral feature string is produced for each malware, as shown in Figure 2, consisted of 320 bits per malware. The produced strings would be transferred to the cloud database. Next, as shown in Figure 3, similarity of two files is compared by performing logical OR operation on the behavioral pattern string of the sample file and the behavioral pattern string of malware registered into the cloud database. The quantity of 1s in the output represents the number of behaviors observed in either of the files. Then the logical AND operation is performed on the same two strings; the quantity of 1s in the output represents the number of similar behaviors in both of the files. Next, the ratio of output for AND operation to the output of OR operation yields similarity of the sample file to the compared malware. Figure 3 illustrates calculating the similarity percentage of the sample file with the malware given logical OR/AND operations. The ratio is separately calculated for each behavioral feature; obtaining the similarity percentage per behavioral feature. Finally, the average value of similarity percentages results similarity of the sample file and the malware. Given the similarity threshold, in case of similarity between the sample file and a malware is higher than the threshold value then the sample file is detected as a malware of that malware family.

IV. Evaluation Results

The evaluation results were achieved based on 75 malware and benign files n different conditions. The sample files were produced in different sizes ranging from 4KB to 548KB. The 75 files include:

- 40 malware produced by VCL32.
- 20 benign files. The size of benign files ranged from 1 KB to 32 KB.
- 15 malware were randomly selected from various types of Viruses, Worms, Trojans, etc.

Initially 40 malware files produced by VCL32 malware are run in the cuckoo sandbox environment and their behaviors are observed. Next, behavioral features are extracted similar to the string shown in Figure 2. The resulting binary string is registered into the cloud database. Moreover, the same procedure is repeated for the set of 20 benign files.

The produced binary strings from malware files are compared against each other and their similarity threshold is specified; also similarity between malware and benign files is assessed. By calculating the ratio of AND over OR for the two strings being compared, the similarity of strings can be obtained and eventually the similarity threshold for files of malware family is achieved. Table I shows the maximum and minimum values of similarity calculations for the VCL32 malware family vs. benign files vs. other malware (e.g., viruses, worms and etc.).

In Figure 4, blue symbols represent the level of similarity between pairs of files from VCL32 malware family; whereas red symbols denote similarity between pairs of malware files and other malware files (e.g., Viruses, Worms, Trojans, etc.). In Figure 5, blue symbols represent the level of similarity between pairs of files from VCL32 malware family; whereas red symbols denote similarity between pairs of malware files and benign files.

TABLE I
Similarity Percentage for VCL32 Malware

Comparison	Average %	Highest %	Lowest %
VCL32 vs. VCL32	89.4 %	100 %	67.9 %
VCL32 vs. Other Malware	37.3 %	100 %	8.6 %
VCL32 vs. Benign	24.1 %	27.2 %	17.3 %

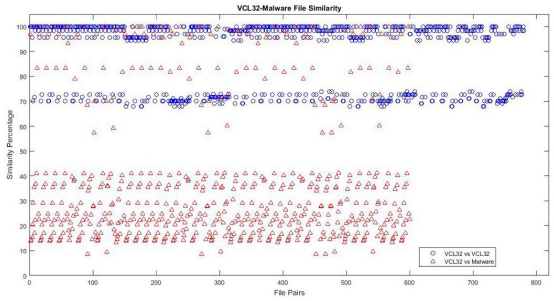

Fig. 4. Calculating similarity percentage of a sample file with malware.

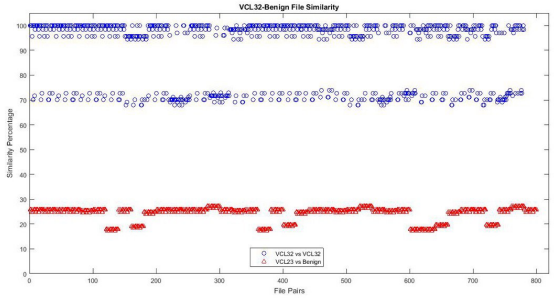

Fig. 5. Calculating similarity percentage of a sample file with benign files.

V. Conclusion

This study proposed a cloud based malware detection method with behavioral entropy at the core, which involved extraction behavioral features into a database for future comparisons and classification into appropriate malware families. The proposed method greatly improves malware detection rates while minimizing false positives.

References

[1] M. Egele, T., Scholte, E., Kirda, & C. Kruegel. A survey on automated dynamic malware-analysis techniques and tools. ACM computing surveys (CSUR), 44(2), (pp. 1-42), 2008.

[2] K. Vahedi, M. Abbaspour, K. Afhamisisi, & M. Rashidnejad. Behavioral Entropy Towards Detection of Metamorphic Malwares. In 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE) (pp. 78-84). IEEE.

[3] N. Runwal, R. M. Low, & M. Stamp. Opcode graph similarity and metamorphic detection. Journal in Computer Virology, 2012, (pp. 37-52).