# Poster: Using Semantic Snippets of Malware Traces for Efficient Behavioral Analysis

Jaime C. Acosta (Project Lead)
U.S. Army Research Laboratory
White Sands Missile Range, NM 88002–5513
http://www.jaimeacosta.info/

Brenda G. Medina (Project Member)
U.S. Army Research Laboratory
White Sands Missile Range, NM 88002–5513

## I. Introduction

Given the myriad of new malware instances released each year, complete manual analysis is infeasible. However, it is important for a human analyst to document the behavior for future identification and for forensic investigations. On the bright side, many of the malware instances are slightly changed and recycled; this leads to substantial duplication. Current automated methods identify commonalities in malware activity logs and cluster similar instances. However, these methods lack semantic behavior identification and attention to multi-purpose malware.

Our work focuses on identifying sequences of semantically-rich log events that are shared among malware instances. These events will be labeled by analysts (see Figure 1). Future encounters of these log events will be replaced with these labels to reduce duplicate work and to facilitate analysis of multi-purpose malware. These labels will also be used to build a system to automatically classify malicious behaviors. In this paper, we first detail issues related to the current analysis process. Next, we provide our methodology and our steps up to now. Lastly, we describe our future plans.

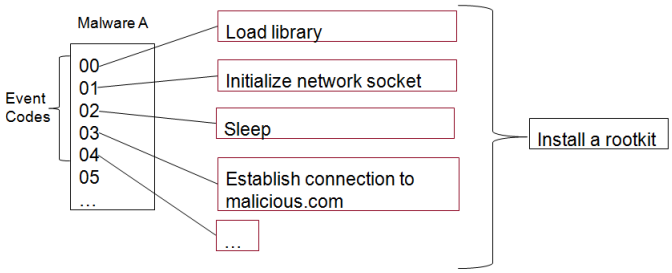


Fig. 1. Eventually, semantically-rich log sequences can be labeled by analysts.

## II. The Analysis Process

The process of malware analysis is long and requires high technical expertise. This process involves capturing and analyzing a binary file. If the binary is malicious, a hash signature is generated and the behavior is documented.

Most malicious executables are captured with honeypots and then hash signatures are generated. This allows traditional anti-virus scanners to identify known malware on systems by matching these hash signatures to executables. However, these scanners are not capable of identifying even slightly modified and recycled malware. Novel methods, therefore, attempt to identify malware based on behavior.

Two prominent methods for behavior analysis are static and dynamic. Static analysis requires reverse engineering binary files. Many times malware is obfuscated, meaning that the malicious code is intentionally hidden. This makes the analysis extremely complex and limited to small binaries. Alternatively, dynamic analysis [1] involves running malware in a sandbox environment and reviewing activity logs (dynamic analysis). Even using dynamic analysis, finding malicious behavior is non-trivial due to the large sized logs. For this reason, automated similarity detection is used.

Bayer et al. [2] use machine learning algorithms to identify similarities in malware instances by comparing their activity logs, which include system calls, their dependencies, and network behavior. Next, the malware instances are clustered based on their dynamic behavior. A limitation of this approach is that the algorithm is trained with a fixed set of malware. It does not allow retraining with additional malware samples during the clustering phase. Malheur [3] extends this by establishing an iterative mechanism that consists of clustering and then classifying new instances into existing clusters. In his work, similarity is determined by the presence of shared fixed-length instruction sequences.

After the instances are clustered, an analyst may have to conduct deeper investigation, such as finding exact differences and similarities in the binaries. It may be the case that malware in different clusters share common behaviors. This results in redundant analysis by a human analyst. Another issue is that instances in a cluster are not exactly the same. There may be malicious behavior that is unique to one instance within a cluster. One way to alleviate these issues is to, instead of determining similarity by using fixed length sequences as in previous work, develop techniques that are not tied to sequence length and automatically detect varied sized semantically-representative sequences. These sequences can then be labeled by an analyst to reduce future workload and eventually these labels can be used to automatically determine malicious sequences.

## III. Completed Milestones

We will build a system to improve malware analysis. Our methodology consists of efficiently identifying common substrings among malware, facilitating the labeling process, and finally using these labels to build a system to automatically identify and label malware behavior substrings.

### A. Common behaviors among malware

Similar malware produce similar activity logs. Using a public dataset [4], we showed that by labeling a set of short malware logs, an analyst's workload can be greatly reduced. In this work, we identified common substrings of a minimum length, $n$, among a set of short malware logs. The short set contained 2071 logs totaling 44MB. We then used these common substrings to determine their presence in a set of longer malware activity logs. The longer set contained 1,060 logs totaling 490MB. Regarding the dataset, we used only event descriptions and omitted event parameters (e.g., for an *open socket* event, we did not use the port number, interface name, etc.) Test results using different lengths $n$ are shown in Figure 2.

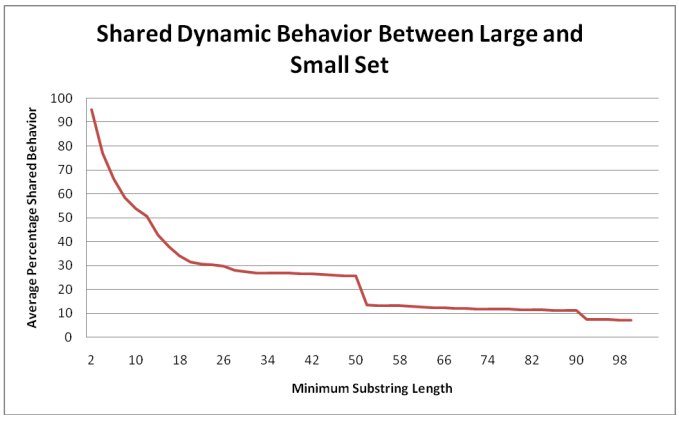


Fig. 2. Average percentage of the large set accounted for by small set's common substrings

The results show that the similarities are not restricted to short sequences. Using $n = 12$, the short set's common substrings account for half of the longer set. A problem during this work was the long time taken to extract common substrings. For this reason, the next step was to determine if parallelization could improve performance.

### B. Efficient Common Substring Identification

We considered four implementations of the common substrings algorithm. Along with the original dynamic programming approach, we implemented two parallel versions (one using threads and a semantically equivalent CUDA counterpart). We tested performance with multiple input strings. Figure 3 shows that using the CUDA, execution time is decreased by roughly 7 times and has seemingly linear growth.

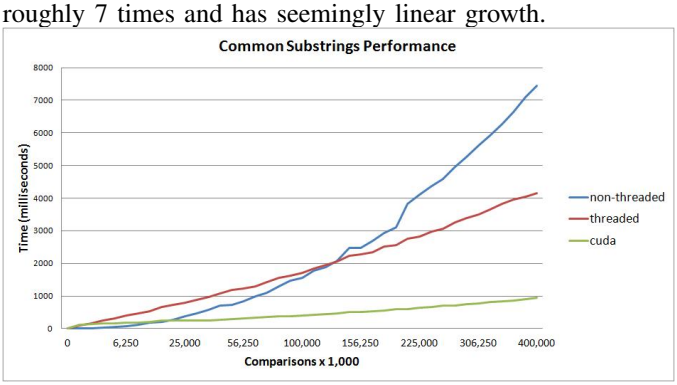


Fig. 3. Performance of common substring implementations with multiple inputs.

## IV. Work in Progress

### A. Graphical User Interface

Our current focus is to facilitate the labeling of common behaviors. Our envisioned use case starts with an analyst given a set of malware activity logs. Using a graphical user interface, the analyst will first run the common substrings algorithm. Next, the analyst will choose a short activity log. The interface will display all common substrings found among the set of malware. An analyst can then label these substrings and the labels will replace other occurrences with these labels. We are currently implementing this such an interface and we will measure its quantitative utility.

### B. Automatic Behavior Labeling

Still underway is research to determine ways to automatically label malicious behaviors in activity logs. Obviously, not all common substrings in malware are malicious. An open research question is how to distinguish these automatically. We plan to use machine learning for this task. At first we plan to use the analysts' labels to classify malicious versus non-malicious substrings. Eventually, we will investigate if more sophisticated behaviors can be labeled automatically.

## References


[1] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A Survey on Automated Dynamic Malware Analysis Techniques and Tools," *ACM Computing Surveys, to appear*, 2011.

[2] U. Bayer, P. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, Behavior-Based Malware Clustering," in *Network and Distributed System Security Symposium (NDSS)*. Citeseer, 2009.

[3] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[4] J. Acosta, "Using the longest common substring on dynamic traces of malware to automatically identify common behaviors," in *Proceedings of the 6th International Conference on Information Warfare and Security*, 2010.