Poster: Clustering Malware from Creation Tools Using Code Clones

Christopher I. G. Lanclos
Department of Computer Science and Engineering
Mississippi State University
Starkville, USA
cl691@msstate.edu

Dae Glendowne, Ph.D.
Department of Computer Science
Stephen F. Austin State University
Nacogdoches, USA
glendowndj@sfasu.edu

Christopher Archibald, Ph.D.
Department of Computer Science and Engineering
Mississippi State University
Starkville, USA
archibald@cse.msstate.edu

John A. Hamilton, Ph.D.
Department of Computer Science and Engineering
Mississippi State University
Starkville, USA
jah514@msstate.edu

***Abstract*— Malware development and creation has been on the rise for many years and is expected to continue in this trend. One of the reasons for this trend is the ease of creating variants of malware. This research is clustering malware using code clones that are found in the assembly of the different variants of malware to create phylogenetic graphs. This research looks at malware developed from malware creation tools and evaluating the clusters using external and internal criteria.**

Keywords—Malware Analysis, Machine Learning, Clustering, Code Clones, Phylogenetic

I. Introduction

In 2016, around 357 million new pieces of malware were first detected, which is in addition to the 355 million pieces of malware that were detected in 2015 and 274 in 2014 [1]. One of the reasons for such an increase is new malware are based on other malware variants [2]. The development of malware variants is as simple as making slight changes to the source code or even changing compiler settings, which would render signature-based detection tools unreliable. The reuse of code by malicious agents makes the workload of analyzing malware more difficult, due to the amount of malware that needs to be analyzed. The process of analyzing malware through either dynamic or statistical means is costly, either by time, computationally or both.

One of the most informative ways to analyze malware is reverse engineering. Reverse engineering tries to understand a system by identifying key components, artifacts and the relationship between them. In other words, malware analysts can learn what is done by the different variants. Although reverse engineering is one of the most thorough ways to analyze malware, it comes with a high cost of time. In malware analysis, reverse engineering can be beneficial because access to the source code of the malware is rarely available.

Due to the trend of creating malware from previous pieces of malware, research has emerged to find ways to discover malware that share code bases. One of the ways that have been researched is code clones. Code clones are complete copies of a section of source codes or a piece of source code that has been slightly changed but has the same function [3]. Another technique that has been used to find relationships between malware is the machine learning concept called clustering. Clustering has been applied to both dynamic and static malware analysis techniques. Hierarchical clusters have also been considered as a technique to analyze malware. Hierarchical clustering follows a similar process of clustering malware but recursively considers the need to split a cluster [4].

Hierarchical clustering and clustering in general are used many times to create machine learning classification tools, which would help with a labeling issue that is present in the malware analysis community. Currently, the practice is to use the labels that are provided by the antivirus software. The purpose of this research is to deduce the viability of code clones as features in hierarchical clustering of malware to determine phylogenetic relationships. In addition, it would lead to possibly creating a better labeling system then the inherent labels provided by antivirus software.

II. Background

A. Code Clones

Code clones are grouped into two categories: textual and semantic. Textual code clones are based on the similarity of the actual code, while semantic code clones look at the similarity in the functionality even if they are implemented differently. Within those two categories of code clones there are four types of code clones:

- Type 1 (Textual): Identical code fragments except for white space, layouts, and comment variations.
- Type 2 (Textual): Structurally and syntactically identical fragments except for variations in identifiers, literals, types, layouts, and comments.
- Type 3 (Textual): Copied fragments with further modifications. Statements can be changed, added, or removed, in addition to variations in identifiers, literals, types, layouts, and comments.
- Type 4 (Semantic): Are based on code fragments, which perform the same computation, but are implemented using different syntactic variants.

Usually, code clones are based on source code [3], but there is a growing interest in code clones based on the binary of software. The first reason is that there has been an increase in software piracy [3]. The second reason is the increase of new malware variants that are based on other malware samples [2]. As mentioned before the yearly increase in malware is staggering, without doubt is connected to the reuse of existing malware or free malware creation tools [5].

Detection of malware is predominately done with signature-based tools that leave systems vulnerable to new variants of malware. A change in the source code of a malware sample changes the signature of that malware, which makes signature-based tools ineffective against new variants. Currently, malware analysts would have to reverse engineer each of the new variants until they realized the shared components of already existing reversed engineered malware, which could never happen due to the number of pieces of malware that need to be reversed. Code clones have already started being used to assist in this area.

B. *Clustering*

In machine learning, clustering techniques seek to divide the whole data set into homogeneous clusters where similarity inside of a given cluster is maximized, while similarity of a piece of malware outside of given cluster is minimized [4]. This may seem redundant to code clones, but code clones only determine if the same code clones exist in two or more different pieces of malware. Code clone detection does not provide information about the similarity of the whole malware outside of sharing a code clone. It also does not reveal anything about the relational differences malware have with each other.

Clustering has been applied to both dynamic and static malware analysis techniques. Dynamic malware analysis is the process of observing and analyzing malware while it is running, while static analysis is examining the executable without running the malware. Examples of dynamic features used in clustering include OS objects and operations [6]. Control flow graphs, binary sequences, opcodes and mnemonic sequences are all static feature types that have been used in clustering malware [7]. Hierarchical clustering and phylogenetic modeling are closely related because they both organize malware into families and provide information about the relationships between malware [8].

Phylogeny is the set of derivate relationships between a group of species [9]. As a result, malware phylogeny models are estimations of the derivation between the relationships of a malware sample set [9]. In short, malware phylogeny models seek to represent the relational similarities or differences in a set of malware samples. The use of phylogeny in malware analysis is not a well-researched area, but it is believed that deriving such information will be beneficial to the malware analysis community [9]. The benefits of phylogenetic malware clustering can highlight essential features of malware, reveal unknown relationships and the strength of those relationships and reveal outliers in particular datasets.

III. METHODOLOGY

A. *Malware Sample Set*

This research examined malware created by malware creation tools, which would consist of 300 samples. The malware is limit to samples that are Windows 7 compatible. Windows 7 was a focus for two reasons, the first being that it provides a big enough data sample. The second reason is for its wide use. All samples were tested to identify if they were packed and if so were discarded. After gathering the sample sets, traditional static features were extracted for external criteria validation. Afterward, the code clones were identified using Kam1n0 [10]. Kam1n0 examines each of the samples and detect the code clones in that sample set. The code clones found by Kam1n0 are hashed and used as features. Other features are used that are derived from code clones such as the ratio of code clones in samples and ratio of exact code clones to inexact code clones.

B. *Hierarchical Clustering*

The hierarchical clustering of the malware samples is done twice. The first phylogenetic graph was created based on the traditional static features. The second was based on code clone features extracted from Kam1n0. The first phylogenetic graph serves as ground truth to the second. The validity of the phylogenetic graph is based on the ground truth that is known about the malware from the different creation tools. Also, the goodness of the clusters from the second phylogenetic graph, which is the cohesiveness and separation of the clusters.

IV. CONCLUSIONS

The cost to develop malware versus the cost to analyze malware is drastically different. This research is focused on reducing that time by showing the relationship between malware through the development of phylogenetic graphs using code clones. The clustering of code clones in the manner this research is proposing has not been done.

REFERENCES

[1] "Internet Security Threat Report 2016 | Symantec." [Online]. Available: https://www.symantec.com/en/ca/security-center/threat-report. [Accessed: 29-Mar-2017].

[2] B. Anderson, C. Storlie, M. Yates, and A. McPhall, "Automating Reverse Engineering with Machine Learning Techniques," in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, New York, NY, USA, 2014, pp. 103–112.

[3] M. Dong *et al.*, "A New Method of Software Clone Detection Based on Binary Instruction Structure Analysis," in *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, 2012, pp. 1–4.

[4] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*, 3 edition. Elsevier: Morgan Kaufmann, 2011.

[5] R. A. Nolan and P. P. Chen, "MCARTA: A Malicious Code Automated Run-Time Analysis framework," in *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, 2012, pp. 13–17.

[6] H. T. Wang, C. H. Mao, T. E. Wei, and H. M. Lee, "Clustering of Similar Malware Behavior via Structural Host-Sequence Comparison," in *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, 2013, pp. 349–358.

[7] X. Hu, K. G. Shin, and S. Bhatkar, *MutantX-S: Scalable Malware Clustering Based on Static Features*.

[8] L. Kellogg, B. Ruttenberg, A. O'Connor, M. Howard, and A. Pfeffer, "Hierarchical management of large-scale malware data," in *2014 IEEE International Conference on Big Data (Big Data)*, 2014, pp. 666–674.

[9] M. Hayes, A. Walenstein, and A. Lakhotia, "Evaluation of malware phylogeny modelling systems using automated variant generation," *J. Comput. Virol.*, vol. 5, no. 4, p. 335, Nov. 2009.

[10] S. H. H. Ding, B. C. M. Fung, and P. Charland, "Kam1n0: MapReduce-based Assembly Clone Search for Reverse Engineering."