Object Allocation Pattern as an Indicator for Maliciousness - An Exploratory Analysis

Adamu Hussaini
ahussa7@students.towson.edu
Towson University
Towson, Maryland, USA

Bassam Zahran
bzahran@towson.edu
Towson University
Towson, Maryland, USA

Aisha Ali-Gombe
aaligombe@towson.edu
Towson University
Towson, Maryland, USA

Abstract

Traditionally, Android malware is analyzed using static or dynamic analysis. Although static techniques are often fast; however, they cannot be applied to classify obfuscated samples or malware with a dynamic payload. In comparison, the dynamic approach can examine obfuscated variants but often incurs significant runtime overhead when collecting every important malware behavioral data. This paper conducts an exploratory analysis of memory forensics as an alternative technique for extracting feature vectors for an Android malware classifier. We utilized the reconstructed per-process object allocation network to identify distinguishable patterns in malware and benign application. Our evaluation results indicate the network structural features in the malware category are unique compared to the benign dataset, and thus features extracted from the remnant of in-memory allocated objects can be utilized for robust Android malware classification algorithm.

ACM Reference Format:
Adamu Hussaini, Bassam Zahran, and Aisha Ali-Gombe. 2021. Object Allocation Pattern as an Indicator for Maliciousness - An Exploratory Analysis. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy (CODASPY '21), April 26–28, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3422337.3450322

Keywords: Memory; Forensics; Android; Exploratory analysis

1 Introduction

Android-based devices such as mobile phones, smartwatches, cars, glasses, and TVs have continued to dominate the market in the last decade. Simultaneously, the number of Android applications in the official Google play store alone is estimated to be more than 2.8 million[1] as of April 2020. On the contrary, the amount of malware generated is more than twice the proportional number of Android apps produced. According to the G DATA report[2], there are 4.18 million malicious apps released in 2019 alone. Traditionally, Android apps are examined for malicious functionality using static or dynamic approaches. In static analysis, extracted features from the known malware code/package are compared against new samples using signatures, heuristics, and other machine learning techniques. Although static analysis is often fast, it is limited to examining known variants and unobfuscated samples. On the other hand, dynamic analysis involves executing target malware in a contained environment and monitoring its behavior during execution. Most Android-based dynamic analysis systems are designed by placing hooks in a target application and/or the execution environment to collect runtime data. Gathering every important runtime behavioral data often incurs significant overhead; thus, tools designed for this type of analysis make a tradeoff between robustness and efficiency. This paper examines the feasibility of applying memory forensics as an alternative approach to malware analysis. We perform an exploratory data analysis that utilizes the reconstructed Android process memory object allocation graph to discover distinguishable statistical patterns in malware and benign applications. Our evaluations of 14 memory images grouped into malware and benign samples using univariate, multivariate, and network analysis indicates that the memory allocation pattern in Android malware has distinctive statistical features relative to the benign samples.

2 Forensics Analysis

Memory forensics is a post-execution analysis technique that explores and recovers digital content allocated by process or the Kernel during runtime. The emergence of memory analysis frameworks and tools such as Volatility Framework[5], ReKall[6] and research work such as OAGen[4] has widened the scope of digital forensics beyond digital crime investigations to the analysis of sophisticated malware. This research explores a per-process memory analysis as an alternative technique to fingerprint Android applications for malware analysis. When an application executes at runtime, it creates a process. This process allocates multiple data structures and code in regions of memory. Collectively the allocated contents determine the functionality of the running process. The memory image for a process is a snapshot of a running process's behavior at a point in time. Each snapshot has distinguishable characteristics that differentiate it from other snapshots of different processes. Thus in this paper, we explore whether a reconstructed memory allocation graph of a process can be a useful and distinguishable pattern in malware classification and detection. We employ Droidscraper[3] and OAGen[4] as memory recovery tools to reconstruct 14 memory images and then utilize various exploratory data analysis techniques as a conceptual methodology to understand various aspects of the graphs' structures.

3 Study Design

We run 14 applications comprising 7 benign apps and 7 malware in a Samsung S8 AVD created in a Genymotion emulation environment. The AVD was configured with one Gmail account, a couple of SMS and contacts, and an activated location coordinates. The test apps were manually executed for about 15 minutes each, with all the

ACM ISBN 978-1-4503-8143-7/21/04.
https://doi.org/10.1145/3422337.3450322

requested permissions approved. Each target app's process image is then extracted using memfetch[7]. As shown in the workflow in Figure 1, we employed DroidScraper - a tool for recovering valuable runtime data structures for Android applications by enumerating and reconstructing allocated objects from a process memory image. DroidScraper is used to retrieve each process's heap allocation from the memory image. Furthermore, the OAGen tool is utilized to reconstruct the entire allocation into an object allocation graph. OAGen is a post-execution and semantic analysis algorithm that utilizes points-to analysis to generate an object allocation graph from a process memory image. OAGen reconstructs each object allocation graph as an undirected graph with nodes representing allocated objects and edges representing the relationship between the objects in memory. The overarching objective of this study

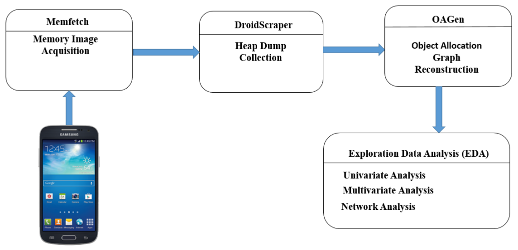
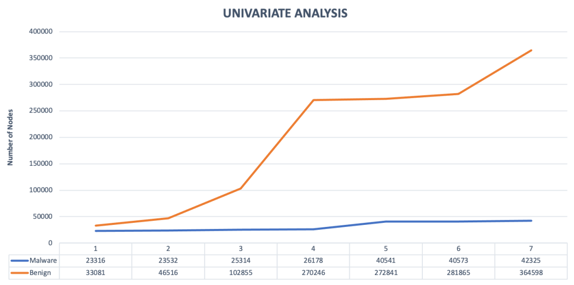


Figure 1: The Study Workflow

is to investigate whether the memory allocations graph's statistical features can indicate maliciousness in Android apps. We apply univariate, multivariate and network analysis to explore unique patterns in the memory images obtained from malware and benign samples. For each recovered network, the number of nodes and edges in the graph were carefully recorded. The nodes represent allocated objects, and edges represent the relationship between the objects in memory as shown in Table1. Other distinctive statistical features for the graph, such as community detection, network diameter etc were obtained by analysing the structure of the object allocation network.

Table 1: The Object Allocation Graph (OAG) generated from the sample apps evaluated in this study. The number of nodes represents the total allocated objects, and the number of edges showed the relations between the objects in the graph.

Applications	Number of Nodes	Number of Edges
Chromium	33081	59425
Evolve_sms	46516	86621
Messaging	102855	137042
Signal	270246	199087
Whatsapp	272841	341715
Keeper	281865	331865
Facebook	364598	285355
Tencent	23316	41393
Yandex	23532	39607
Monkey	25314	44756
Yxxinglin	26178	50362
Easyhin	40541	56087
Peaksel	40573	68112
Husor	42325	63944



	1	2	3	4	5	6	7
Malware	23316	23532	25314	26178	40541	40573	42325
Benign	33081	46516	102855	270246	272841	281865	364598

Figure 2: Univariate Analysis illustrating the difference in the number of nodes (objects allocated) in malware and benign applications

3.1 Univariate Analysis

Univariate analysis is one of the simplest forms of statistical analysis that involves only one variable. It can be either Inferential or descriptive. It helps us explore patterns in a dataset by looking at the range of values and the values' central tendency. In this evaluation, we plot a line chart to show the difference between the number of nodes in malware and benign applications. As shown in Figure 2, benign applications have more nodes, which corresponds to larger objects allocations than the malware samples. Thus, to solve a classification problem using this variable, a specific number of nodes can be set as a threshold to indicate whether an Android app is likely to be malware or benign.

3.2 Multivariate Analysis

One of the significant limitations of Univariate analysis is that it cannot be applied to solve correlation problems, and hence may not be comprehensive and appropriate for malware classification. Thus, in this evaluation, we explore multivariate analysis to find the relationship between nodes and edges in the object allocation graph. This type of analysis involves observing multiple variables at a particular time. Multivariate analysis is applied in scenarios where more than one quantity is required for analysis, and the relationship between the observed quantities and their overall structural features is crucial. As mentioned above, the nodes in the object allocation graph represent the number of objects, while the edges represent the nodes' relations. This evaluation plots a scattered diagram for the number of nodes and edges in malware versus benign applications. We computed the R-squared value to examined the goodness of fit for the relationship. Figure 3, showed a linear relationship between a number of nodes and edges for both malware and benign applications. In addition, the R-squared value for the malware is estimated at around 0.85 and 0.79 for the benign apps, thus indicating that the relationship between the two variables is strong.

3.3 Network Analysis

In addition to finding the relationship between nodes and edges, it is also important to further examine additional structural features and patterns exhibited by the object allocation network. Network analysis is a statistical approach used to explore the organization, structural features, and distinctive relationship between nodes in a network. The techniques used in network analysis range from

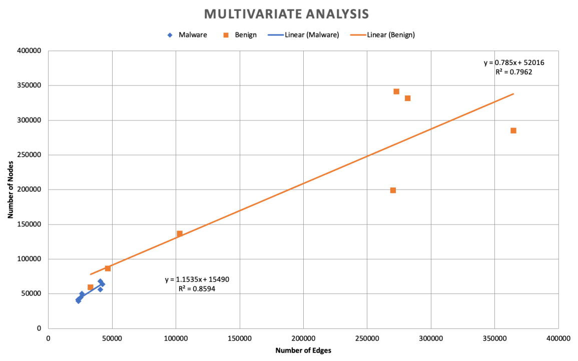


Figure 3: Multivariate Analysis illustrating the relationship between the two variable - number of nodes and number of edges in malware and benign applications

community structure detection, determining the longest and shortest network paths, to exploring different centrality and clustering metrics. In this study, we apply different network analysis metrics to our evaluation dataset. The objective is to evaluate if the network structure in malware has some distinctive characteristics in comparison with the networks in the benign category. As shown in Table 2, the results indicate that benign applications have larger, loosely connected communities, while the malware has much smaller, tightly neat communities. Furthermore, the network diameter, which represents the longest path in the network between two nodes, is slightly higher in benign applications than in the malware category. Other metrics did not show any significant difference. This result showed that the size and structure of a community in an object allocation network could be an important feature in classifying and determining maliciousness in Android apps.

Table 2: Network Analysis Metrics for the Object Allocation Graph in the Malware and Benign Samples

Metrics	Benign	Malware
Communities	**71837**	**4471**
Average Degree	2.35	3.24
Network Diameter	**41**	**33.86**
Weakly Connected Components	**71511**	**4446**
Avg. Clustering Coefficient	0.07	0.08
Eigenvector Centrality	1.15	0.32
Avg. Path Length	**45.07**	**7.99**

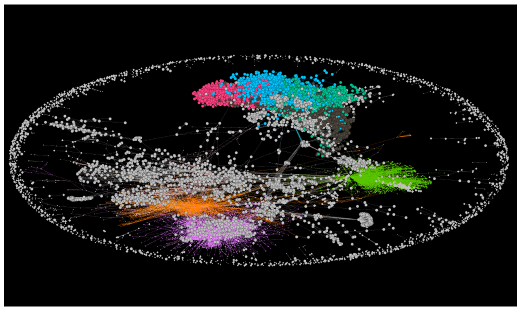


Figure 4: Community Detection in Object Allocation Graphs

4 Community Detection

A community structure within a network is a group of nodes and edges densely connected. This information provides insight into a network and, more profoundly, can be used to uncover interesting properties shared by the members of any group within the community. Community detection metric has become an increasingly crucial technique for finding patterns and behaviors in many research fields such as Mathematics, Economics, Biology, Neuroscience Computer Science, etc. Community detection is especially important in understanding how complex networks are organized and function. Given the complexity and size of a process memory allocation graph, community detection can be utilized to group nodes into substructures with dense connections internally. As shown in the object allocation graph in Figure4, each community represents an object dependency substructure with different colors depicting a separate community. The substructure represents objects that are closely related and, in most cases, depend on other objects in the community or are part of the fields of the objects in that community. This object dependency substructure thus forms a distinctive, cohesive functionality of the running process and, thereby, can be utilized to identify distinguishable patterns in a network. As part of the future work, we aim to apply a Graphical Neural Network (GNN) algorithm for community detection as a technique for robust malware classification.

5 Conclusion and Future Work

In this paper, we present an exploratory data analysis to determine if the structural features in a process object allocation network can be used to determine maliciousness in android apps. We conduct a post-morterm reconstruction of memory images from 14 applications into object allocation networks and then employ univariate, multivariate, and network analysis on each graph. The evaluation results indicate that object allocation networks in malware and benign applications have distinct network characteristics and thus provide alternative features for malware classification. In our proposed future work, we intend to apply GNN for community detection on a much larger dataset for malware classification.

6 Acknowledgement

This work is supported by the National Science Foundation (NSF) under Grant Number 1850054.

References

[1] Smartphone Market Share. Accessed: Apr. 30, 2020. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os

[2] G. Data Software, “G DATA Mobile Malware Report 2019: New high for malicious Android apps”, [Online]. Available: https://www.gdata-software.com/news/g-data-mobile-malware-report-2019-new-high-for-malicious-android-apps

[3] A. Ali-Gombe, S. Sudhakaran, A. Case, and G. G. Richard, “DroidScraper: a tool for Android in-memory object recovery and reconstruction,” in Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses, pp. 547–559, Beijing, China, October 2019.

[4] A. Ali-Gombe, A. Tambaoan, A. Gurfolino, and G. G. Richard, “App-Agnostic Post-Execution Semantic Analysis of Android In-Memory Forensics Artifacts, “In Annual Computer Security Applications Conference (ACSAC), Austin, USA, December 2020.

[5] Volatility Foundation. 2017. Volatility Command Reference. https://github.com/volatilityfoundation/volatility/wiki/Command-Reference. Available:accessed 21-March 2018].

[6] Google, 2016. Rekall. https://github.com/google/rekall.

[7] Michal Zalewski. 2003. Memfetch. https://github.com/citypw/lcamtuf-memfetch [Online; accessed 17-March 2018].