Summer Internship Report
2022

Vemuganti Sesha Satvik, Ravikant Gautam

August 16, 2022

Guided By: Dr. Mayank Swarnkar

1 Abstract

The number of smartphone users worldwide has significantly expanded, and so have the number of attacks on these devices. There have been numerous protection strategies for Android malware detection, but the majority of them do not provide early malware detection. As a result, there is a pressing need to develop a system to detect dangerous software before using the data. Another significant issue is achieving high accuracy in detecting Android malware traffic. This study systematically presents the issue beginning with a brief understanding of the system architecture of Android followed by various classifications of Android Malware. The study began with an exploration of tools such as Wireshark and Scapy and consisted of reconstructions of existing Android Malware detection and classification techniques using Deep Learning.

2 Introduction

Since its debut in 2008, Android has overtaken all other mobile operating systems in terms of popularity. About 85% of smartphones sold in 2021 were Android-based. The official marketplace for Android apps, Google Play, had more than 2.8 million apps available at the end of April 2020. Numerous security attack surfaces exist, which substantially jeopardise the integrity of Android programmes because of a variety of reasons, including the open ecology mode of Android applications, its coarse-grained permission control, and its ability to invoke third-party code.

According to prior research, there are three forms of Android malware detection technology: static detection, dynamic detection, and hybrid detection. Static detection relies on questionable code analysis without the Android application being executed. Although it can achieve great code coverage, there are various barriers to it, including dynamic code loading and code obfuscation. On

the other hand, dynamic detection entails running the code and analysing the Android application. This can reveal dangers that are difficult to find through static analysis, but dynamic detection requires a lot of processing power and takes a lot of time. In order to balance the effectiveness and efficiency of detection, the hybrid detection approach combines static and dynamic detection.

In the identification of Android malware, machine learning theory is frequently used, whether it is based on static, dynamic, or hybrid analysis methodologies. Machine learning-based malware detection has the capacity to identify previously unknown types of malware and can perform better in terms of detection efficacy and efficiency compared to traditional methods like signature-based malware detection, which is based on identifying specific patterns of known malware. Machine learning-based methods for detecting Android malware have been covered in some prior publications.

Achieving high accuracy in detecting android traffic has been a critical issue. Gohari et. al. [1], 2021, suggests a deep learning approach for detecting Android malware using network traffic data. Data preprocessing is frequently required for machine learning methods, however this preprocessing takes time. Deep learning algorithms work well for malware detection issues and eliminate the requirement for data preprocessing. This paper presents a technique to extract local features from network flows using the one-dimensional CNN, and then use LSTM to identify the sequential link between the important features. In order to detect android malware, the real-world dataset CICAndMal2017 with network traffic features was used to train the model. The model achieves 99.79% accuracy in binary classification, 98.90% in category classification, and 97.29% in family classifications.

3 Literature Survey

The following articles were surveyed during the internship.

Li et. al [2] introduces a Significant Permission IDentification (SigPID) in this work, a malware detection method based on permission usage analysis. The paper proposes three stages of trimming by mining the permission data to find the most important permissions that can be useful in differentiating between benign and malicious apps, as opposed to extracting and analysing all Android permissions. The research finds that SigPID is more efficient than other cutting-edge methods, detecting 93.62% of the dataset's malware and 91.4% of the samples that are unknown or new.

Lopes et al. [3] (Overview of machine learning methods for Android malware identification) provides an overview of the current machine learning methodologies for mobile malware detection based on static, dynamic, and hybrid analysis. It also presents the benefits and drawbacks of each strategy and makes compar-

isons between them.

The comparison of various static and dynamic analysis methodologies for Android malware is highlighted in Choudhary and Kishore [4] 2018, HAAMD: Hybrid Analysis for Android Malware Detection. The study discusses the comparative data between these two types of analysis and also accommodates several sub-approaches of these analysis techniques with their functionality employed for malware detection. In this study, a new technique known as hybrid analysis—a combination of static and dynamic analysis—has been developed. The effectiveness of this new technique is then compared to that of already-in-use techniques.

Amamra et al. 2013 [5], Smartphone malware detection: From a survey towards taxonomy, reviews the state-of-the-art methods for detecting smartphone malware. These methods have been organised into a taxonomy using three rules. These guidelines were deduced and assembled via a literature review of its own([6], [7], [8], [9], [10], [11]). Reference behaviour, analytic methodology, and malware behaviour depiction are the rules. Reference behaviour rule categorises two primary groups of Smartphone malware detection techniques: signature-based and anomaly-based. Then, implications are determined for these classes in accordance with the analysis method rule and the malware behaviour representation rule.

Alqahtani et al. 2019, A Survey on Android Malware Detection Techniques Using Machine Learning Algorithms [12], focuses on machine learning-based classifiers and analyses the state of the art in Android malware detection techniques.

The permission-based detection methodology and the signature-based detection strategy are the two main methods used to detect static Android malware in Samra et al., A survey of Static Android Malware Detection Techniques [13]. It is a comparative study, thus anybody researching this subject should find it useful. The study uses precise parameters to highlight the similarities and differences as well as the accuracy of significant published studies.

Mobile attackers have increased as a result of the quick rise in smartphone use. The majority of the time, dishonest applications have malicious code inside of them that can destroy both the hardware and the software. These dangerous applications or malware are frequently made to interrupt the device or collect data from it. In an effort to stop these issues, several techniques are suggested. Gyamfi et al. 2019 [14], compares the most widely used and most modern strategies and recommend the most effective.

4 Work Done / Proposed Method

Initial work involved understanding Wireshark. In a capture, Wireshark can automatically resolve IP addresses to domain names, although this feature isn't enabled by default. Once this option is enabled, domain names shall be visible instead of IP addresses. However this pollutes the captured traffic with additional DNS requests since Wireshark looks up each domain name. Wireshark also supports automatic capturing of network traffic. A revolutionary feature enables capture of traffic from remote computers. Remote capture enables Wireshark to capture traffic from a router, server, or another computer in a different location on the network. Although irrelevant to further work done in the internship, Wireshark has a Firewall Access Control List (ACL) tool that enables the user to create firewall rules for a firewall to the network.

Another tool that came in handy was the python library of Scapy. It can send packets over the wire, collect them, match requests and responses, forge or decode packets of many different protocols, and do a lot more. The majority of traditional activities, such as scanning, tracerouting, probing, unit testing, assaults, or network discovery, can be handled with ease using Scapy. Hping, arpspoof, arp-sk, arping, p0f, and even some features of Nmap, tcpdump, and tshark can be replaced by it. Useful in writing quick scripts for the analysis of capture (.pcap) files.

Next came the need for a library to visualise dynamic graphs. Essentially, dynamic graphs are a discrete sequence of static graphs. They can be used to model occerences ranging from Protein-Protein interactions to sports tournaments' standings' evolution. Xu et al. 2016 [15] present a brand-new graph-based representation for each feature vector representation in their work titled "Dynamic Android Malware Classification Using Graph-Based Representations". A variety of methods can be used to classify malware for the Android ecosystem. Dynamic analysis based on system call invocations captured during the execution of Android applications is a significant technique that has gained traction recently. System calls are typically converted into flat feature vectors for dynamic analysis, which are then fed into machine learning algorithms for categorization.

5 Experiments and Results

Malware detection tools for Android are widely available. Static analysis, dynamic analysis, and hybrid analysis are the most widely used mechanisms. However, a lot of analytic techniques concentrate on static, dynamic, or hybrid detection and infrequently take network traffic into account. Today, all attackers use mobile networks to connect with users' malicious apps or obtain crucial information.

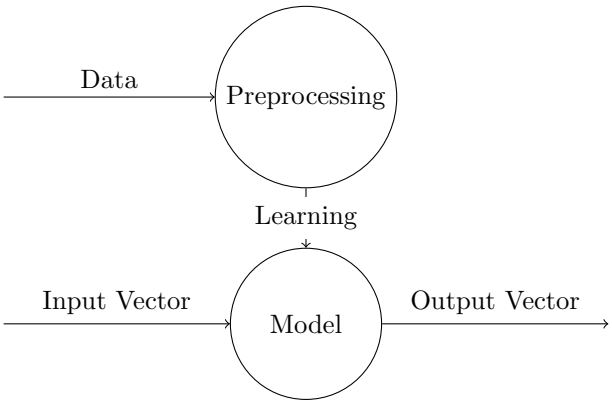

Figure 1: Common ML algorithms with time consuming preprocessing phase

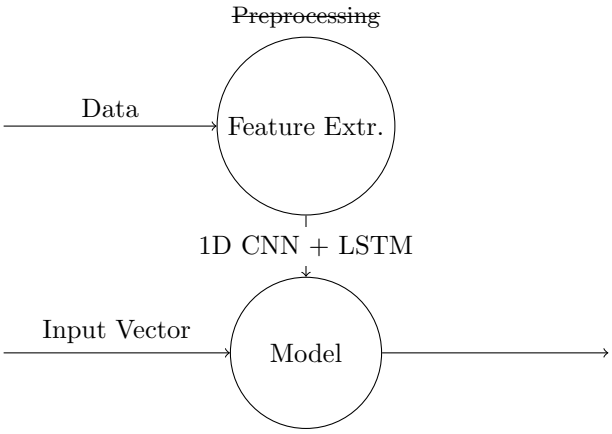

Figure 2: Common ML algorithms with time consuming preprocessing phase

Thus, network traffic can be used to examine Android applications. Network traffic was employed in certain research on Android malware detection, and a number of recently proposed methods used static analysis, dynamic analysis, or both. (See Figure(1))

However Gohari et al. [1] looks at works that combine deep learning techniques to examine static and network traffic aspects. It focuses on network traffic-based detection using extracted flows from PCAP files and deep learning for malware detection. (See Figure(2))

6 Conclusion and Future-Work

In this internship, a deep learning model to identify Android malware was reconstructed. We used the CICAndMal2017 real-world dataset. We used the CICFlowMeter programme and network traffic features to extract features to identify malware Android. Additionally, we used CNN and CNN-LSTM deep learning models to analyse the dataset. We then contrasted our findings with those of other algorithms in the literature [1]. According to the experimental findings (as presented in [1] and reconstructed), CNN-LSTM provides the highest level of binary classification precision for malware. The original paper contrasted these findings with those of other studies, some of which used static attributes, and found that the results using CNN-LSTM were superior. We used all of the network flow features as well as detection based on network traffic. Both network flow features as well as network-traffic based detection were used in the model proposed in [1].

Towards the end, we began a survey of dynamic graph-based methods for android malware detection. In this study [15] the classification abilities of innovative graph-based representations and conventional feature-vector-based representations for system call invocations are compared with each. In order to analyse system call consumption in Android malware, their conventional histogram, n-gram, and Markov chain representations are first computed. Three graph-based representations are suggested where each process is considered as a vertex and labelled with a feature vector in order to increase the classification accuracy of the conventional feature-vector-based representations. Then, graph kernels are used to generate graph similarities between the graph-based representations, which are then categorised using the SVM algorithm. The graphs are then compressed and parallelize the computation is done using a multi-core CPU to accelerate the graph kernel computation. **The methods and results provided in this paper were not reconstructed by us. The paragraph is a brief but interesting recount of the techniques employed by the author.**

As future work, I am looking to explore more about the implementation of dynamic graph techniques (or algorithms) for detection and classification of android malwares. Parallel to this summer internship, I have been writing code for **Software Heritage** - An OSS Organization ([16], [17]) trying to create a universal archive of open source software code repositories, as part of Google Summer of Code 2022. This work has included a study of graph algorithms, their implementation and optimization for implementation on large scale systems. While looking into Open Source graph tools, I came across **Webgraph** [18] which presents a very efficient graph compression algorithm. To begin with dynamic graph algorithms, I shall try to use the dynamic graph techniques and models presented in [15] on the CICAndMal2017 dataset.

References

[1] Mahshid Gohari, Sattar Hashemi, and Lida Abdi. Android malware detection and classification based on network traffic using deep learning. In 2021 7th International Conference on Web Research (ICWR), pages 71–77, 2021.

[2] Jin Li, Lichao Sun, Qiben Yan, Zhiqiang Li, Witawas Srisa-an, and Heng Ye. Significant permission identification for machine-learning-based android malware detection. IEEE Transactions on Industrial Informatics, 14(7):3216–3225, 2018.

[3] João Lopes, Carlos Serrão, Luís Nunes, Ana Almeida, and João Oliveira. Overview of machine learning methods for android malware identification. In 2019 7th International Symposium on Digital Forensics and Security (ISDFS), pages 1–6, 2019.

[4] Mahima Choudhary and Brij Kishore. Haamd: Hybrid analysis for android malware detection. In 2018 International Conference on Computer Communication and Informatics (ICCCI), pages 1–4, 2018.

[5] Abdelfattah Amamra, Chamseddine Talhi, and Jean-Marc Robert. Smartphone malware detection: From a survey towards taxonomy. In 2012 7th International Conference on Malicious and Unwanted Software, pages 79–86, 2012.

[6] Tibra Alsmadi and Nour Alqudah. A survey on malware detection techniques. In 2021 International Conference on Information Technology (ICIT), pages 371–376. IEEE, 2021.

[7] P Vinod, R Jaipur, V Laxmi, and M Gaur. Survey on malware detection methods. In Proceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09), pages 74–79, 2009.

[8] Peyman Kabiri and Ali A Ghorbani. Research on intrusion detection and response: A survey. Int. J. Netw. Secur., 1(2):84–102, 2005.

[9] Grégoire Jacob, Hervé Debar, and Eric Filiol. Behavioral detection of malware: from a survey towards an established taxonomy. Journal in computer Virology, 4(3):251–266, 2008.

[10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3):1–58, 2009.

[11] Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra. A survey on security for mobile devices. IEEE communications surveys & tutorials, 15(1):446–471, 2012.

[12] Ebtesam J. Alqahtani, Rachid Zagrouba, and Abdullah Almuhaideb. A survey on android malware detection techniques using machine learning algorithms. In 2019 Sixth International Conference on Software Defined Systems (SDS), pages 110–117, 2019.

[13] Aiman Ahmad Abu Samra, Hasan N. Qunoo, Fatma Al-Rubaie, and Haneen El-Talli. A survey of static android malware detection techniques. In 2019 IEEE 7th Palestinian International Conference on Electrical and Computer Engineering (PICECE), pages 1–6, 2019.

[14] Nana Kwarne Gyamfi and Ebenezer Owusu. Survey of mobile malware analysis, detection techniques and tool. In 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pages 1101–1107, 2018.

[15] Lifan Xu, Dongping Zhang, Marco A. Alvarez, Jose Andre Morales, Xudong Ma, and John Cavazos. Dynamic android malware classification using graph-based representations. In 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), pages 220–231, 2016.

[16] Roberto Di Cosmo and Stefano Zacchiroli. Software heritage: Why and how to preserve software source code. In iPRES 2017: 14th International Conference on Digital Preservation, Kyoto, Japan, 2017.

[17] Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. The software heritage graph dataset: Large-scale analysis of public software development history. In MSR 2020: The 17th International Conference on Mining Software Repositories, pages 1–5. IEEE, 2020.

[18] Paolo Boldi and Sebastiano Vigna. The webgraph framework i: compression techniques. In Proceedings of the 13th international conference on World Wide Web, pages 595–602, 2004.