Unstructured Data Collection from APK files for Malware Detection

Prerna Agrawal
Faculty of Computer Technology (MCA)
GLS University
Gujarat, India

Bhushan Trivedi
Faculty of Computer Technology (MCA)
GLS University
Gujarat, India

ABSTRACT

For Malware Detection Machine Learning methods are applied extensively in ascertaining if the given APK file is malware or not. Machine learning methods are found to be less time consuming and less resource consuming compared to non-machine learning-based techniques. We have focused on Machine Learning methods for detecting unknown malware. For detecting the malware a researcher needs to create a dataset of its own. Our dataset generation process includes Android File Collection, Decompilation, and Feature Mining phases. We have already discussed the Android File Collection phase in our previous paper [1]. We have collected 15508 Malware files and 4000 Benign Files using Android File Collection. Android Files contains unstructured data in the form of text and XML files which are complex to process and store. Here our goal is to perform the decompilation of these collected Android files such that we get all the resources as well as the source code in a single instance. We aim to handle the big data in terms of Android Files and process them properly performing the Decompilation. In this paper, we have proposed an available automated solution for decompiling the files that also solves the complexity of handling and processing the big data. We have also discussed our Decompilation phase and presented the structure of the reverse-engineered APK file. We have used an online JADX decompiler [5] for performing the reverse engineering of the APK files.

Keywords

Malware, APK files, Decompilation, Reverse Engineering, Machine Learning, Malware Detection

1. INTRODUCTION

Malware Detection using Machine Learning Algorithms is been used extensively nowadays [4]. Conventionally Malware Detection was based on signature-based patterns, permissions, components using static Analysis [3] which was only able to detect known malware types. Using Machine Learning methods it is possible to detect the unknown and recent malware also [2]. So we have focused on Machine Learning-based approach for our generalized Malware Detection Engine. The overall Logical representation of our Android Malware Detection process is already described in the previous paper [1]. It contains Android File Collection, Decompilation, Feature Mining, and Machine Learning phases. In the Android File Collection phase we have collected 15508 Malware files from the world's famous Android Malware Projects and 4000 Benign Files. For mining, the features from the APK file the reverse engineering of an APK file is needed to extract the java source code and XML files and so the Decompilation phase is needed. In this paper, we have covered our Decompilation phase for performing the reverse engineering process of the files and to collect the unstructured data from APK files. Using the Feature Mining phase all the features will be extracted from the files and our final dataset will be generated. For implementing Machine Learning methods in the Machine Learning phase the dataset will train and test the models for investigating and providing better results. The Feature Mining and Machine Learning phases will be covered in other papers.

Big Data deals with a huge amount of data and is unstructured that is heterogeneous and comes in various formats like text files, images, audio, video, etc [14]. The unstructured data does not have any fixed structure of its own rather than it comprises of its internal structure [15]. The Android Files have their internal structure and they consist of text files and XML files. The Android file collection phase consists of a collection of 15508 Malware files and 4000 Benign Files. So for managing and storing the massive amount of Android files which represents a big data problem and performing its Decompilation is itself a challenge. Here we aim to handle the big data in terms of Android Files and process them properly performing the Decompilation.

According to the existing study [6-10] many researchers have used different tools for decompiling the APK files. There were many challenges faced like the tools were incapable to extract the Java code and XML files in a single instance, a lot of manual intervention was needed, the offline tools were very slow and time-consuming. Again the offline tools were inefficient to handle the unstructured data format of Android Files. Here our objective is to decompile the file by using an appropriate tool that overcomes all the mentioned challenges. So here we propose an available automated solution for decompiling the files. For this we have also studied the existing tools used by the researchers in the existing study [6-10]. We have performed Decompilation using an online JADX compiler [5] of 15508 Malware files and 4000 benign files we collected in the Android File Collection.

This paper is divided into the following sections: Section 2 provides the Related Work for Decompilation process of APK files done by the researchers. Section 3 provides a study of different online and offline decompilers. Section 4 provides our Decompilation phase of files collected in the Android Files collection phase. Section 5 provides the conclusion of the paper.

2. RELATED WORK

The first phase of our work that is the Android File Collection described in another paper is already implemented. The second phase is Decompilation which includes Reverse Engineering of the APK files. We have studied other approaches used by the researchers. In paper [6] the first step is the conversion of APK to .dex file. Dex file is a Dalvik Executable file that contains the java code. The second step is the conversion of .dex to jar files. Here jar files are extracted from dex files by using the

Dex2Jar tool. The jar file contains .class files of java code. The third step is the conversion of class jar files to .java jar files by using FernFlower Library. In paper [7] the APK files are Reverse Engineered by using the offline Androguard tool. In paper [8] the APK files are reverse engineered using APK tool. The APK tool extracts dex files. Now the dex files are again extracted to get the java code in both the approaches described in the paper [7] [8]. In paper [9] the APK files are reverse engineered using APK tool. In paper [10] the APK files are reverse engineered using offline Androguard tool.

All the existing works are already using a 2 to 3 step process for Reverse Engineering process. Each step is performed manually with offline tools and libraries it becomes very time consuming and it is more error-prone. There is no method used for the extraction of XML and java files together in a single instance. For the selection of the proper tool for our task a study of different decompilation tools used in the existing work is being discussed in the next section. In this paper, an online JADX decompiler [5] is used for performing the reverse engineering of the files which extracts XML and java code together.

3. TYPES OF DECOMPILATION TOOLS

There are many online and offline 3rd party tools available for decompiling an APK file used by the researchers. We have studied those tools used by the researchers and are open source free to download for selection of the proper one for our work. The online and offline tools studied are Dex2jar, APK tool, JADX, javadecompilers.com, FernFlower, Androguard tool.

The Dex2Jar is an offline tool, freely available for download, and is command-line. This tool is used to decompile the .dex files into .jar files. The APK tool is offline, freely available for download, and is command-line. This tool is used to extract .dex file and all resources files including Manifest.xml file from the APK file. FernFlower is an offline Java Decompiler which is used to convert .jar files into java source code. Androguard is an offline tool with Python Library used to interact with APK files [12] and is freely available for download. In the decompilation of an APK file, it uses decompilers like dad and dex2jad [13]. The limitation of all these tools is that it will not extract direct java source code in a single instance. The JADX tool is also known as Dex to Java Compiler is offline and freely available for download. It provides both command line and GUI version. It will take the APK file as an input and will extract Java source code and resources with the Manifest.xml file in a single shot. The limitation of this tool offline is that it takes a huge amount of time in decompiling a single file. Java decompilers [5] is an online website providing various facilities for the decompilation of files It provides APK decompiler online which will decompile APK file and extract resources and java files with Manifest.xml. For this, it uses the JADX compiler online. This website is fast and flexible for decompiling the APK files. It takes no extra resources and all the load for decompilation of a file is given on its servers. The advantage of this online website is it decompiles java source code and resources in a single instance.

After the study of different available tools we selected the Java Decompilers [5] as it fulfils the solution to all the mentioned challenges for our Decompilation phase. The next section discusses the Decompilation phase with its Architectural flow and also the structure of a decompiled APK file.

4. DECOMPILATION OF ANDROID FILES

This section contains the various challenges to unstructured data mining and also the decompilation phase.

4.1 Unstructured Data Mining Challenges

Big Data deals with a huge amount of datasets that are unstructured and heterogeneous [16 -18]. The unstructured data does not have predefined schema or models and have their internal structure so the regular RDBMS is inefficient to store them [14] [15]. Big data deals with 3 Vs they are 1) Volume 2) Velocity and 3) Variety [16] [18].

- Volume: It contains big size datasets with a complex data structure. The challenge with volume is to handle the complexity of the data structure [16] [18].
- Velocity: It is the need to handle the speed of new data set creation or updating the existing dataset. This factor applies to machine-generated data e.g. sensing of the mobile device. The challenge with velocity is to handle the capacity of the streaming system as they are limited and obtaining useful information from continuous new dataset creation [16] [18].
- Variety: The datasets come from multiple sources and it can be of various formats like text, audio, video, graph, sensors, etc. [16] [[18]. The variety of data provides more information to solve problems. The challenge with Variety is to handle different formats of data by integrating different technologies [16].

A Decompiled file is the directory with a collection of multiple text and XML files. Every decompiled file leads to an unstructured data format which is quite complex and the existing studies [5-10] of tools are quite incapable to handle such big data. Big data Analysis and Mining contains a 3 tier Processing Framework [16]. Figure 1 represents the big data Analysis and Mining Framework.

Tier 1: Data Accessing And Computing
Tier 2: Data Privacy and Domain Knowledge
Tier 3: Big Data Mining Algorithms

Fig 1: Big Data Analysis and Mining Framework

In Tier 1 for the large datasets the data mining procedures require intensive computing units and clusters for data analysis and comparisons [16]. For maintaining the high-performance big data processors rely on cluster computers where data mining task is deployed by running some parallel programming tools such as Map-Reduce [16]. Tier 2 deals with Data Privacy and Domain knowledge. One way to achieve data privacy is by restricting access to the data such that sensitive data access is limited to certain users only. The second way for maintaining data privacy is to anonymize the data fields such that sensitive information is not highlighted to an individual record [16]. Tier 3 deals with big data mining Algorithms. For designing the algorithms there is a need for deep analysis which is quite complex [17]. Mining algorithms are autonomous and run in a decentralized fashion [16]. For Big Data Mining there is a need for Machine Learning and Data Mining Algorithms which need high computation power and resources [16]. So every Tier in Big data Analysis and Mining deals with all the mentioned challenges which are very complex.

To overcome all the challenges of the unstructured data analysis and mining we proposed an available automated online JADX decompiler [5] that decompiles and stores the file on the cloud servers. It solves all the challenges managing volume, velocity, and variety of data automatically on its cloud servers. Again it solves all the challenges of big data Analysis and mining framework as the cloud servers itself manages the complexity of the data analysis, processing clusters, data privacy, and data mining algorithms. The online decompiler [5] easily manages the unstructured data decompiles the files on the cloud servers and returns it.

4.2 Decompilation Phase

Every APK file is a collection of classes, .dex file, and resources.arsc file. Android uses the Dalvik Virtual Machine (DVM) to execute the java source code that gets converted into byte codes and forms the .dex file. The resources.arsc file contains all the resources used in the application. These resources comprise of .xml files and also contains the Manifest.xml file from which the execution of the application starts. Manifest.xml files contain various permissions, intents, services, intent-filters, broadcast receivers, and activities used in the application.

The decompilation of the files is an essential requirement for feature mining. The existing studies [5-10] claimed the usage of some existing tools used for decompilation. After conducting the study of these existing tools some challenges were found with the usage of them. Mostly all the tools available were offline having resource, memory consumption, needed manual intervention, and were slow processing. Again all the existing tools were incapable of extracting java source code and XML files in a single instance. To overcome these challenges there was a need for some other alternative solution. So we proposed an available automated solution that overcomes all these challenges and is very secure and flexible to use.

The second phase of our work is the Decompilation phase and it is also known as Reverse Engineering of an APK File. Reverse Engineering is a process to generate source code from any executable file. The Decompilation phase segregates the Java code and XML files from an APK file. For extracting the features from the APK files like permissions, Intents, API calls the decompilation of the files is necessary. For the feature selection process the Static Analysis is also performed to segregate the malware and benign files. The feature extraction and mining phase will be discussed in another paper.

Figure 2 illustrates the overall flow of architectural flow for the Decompilation phase. Firstly the user selects any Benign or Malware Android File from Android files Repository and runs the Decompilation module. Secondly, that selected file uploads on the website [5], the website connects to the server and sends the file to the server for decompilation. Thirdly, the server processes that APK file segregates the XML and java files both in a single shot and sends the decompiled zip file back to the website. Fourthly, the user downloads the zip file from the website and the zip file is saved at some physical location.

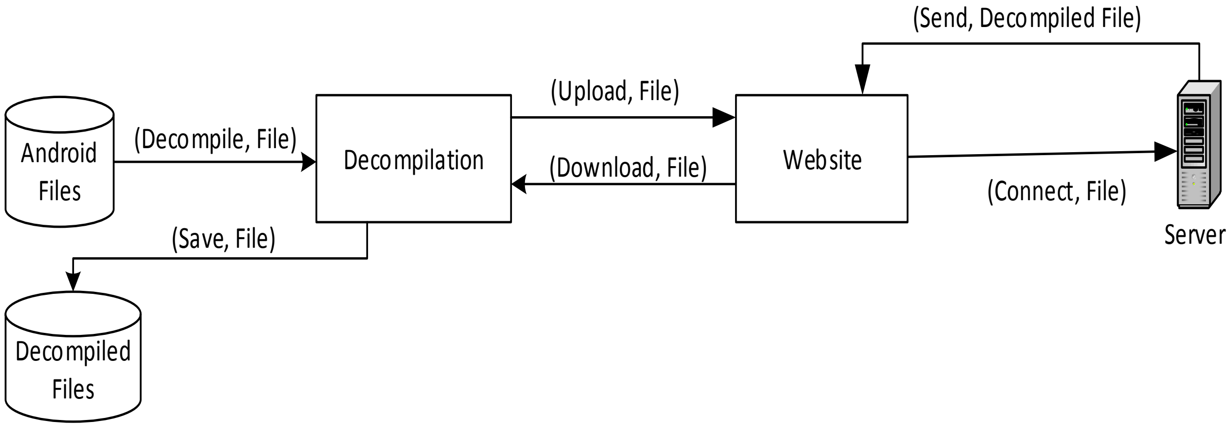

Fig 2: Architectural Flow of Decompilation Phase

After the accomplishment of the Decompilation phase with the APK files, Figure 3 shows the structure of the decompiled APK file. Here the decompiled file contains two folders sources and resources with .dex file and resources.arsc file. The .dex file is further decompiled and sources folder is obtained which contains all the java source code. With the decompilation of resources.arsc file the Manifest.xml file and resources folder are extracted. The resources folder contains all the resources used and .xml files of the application.

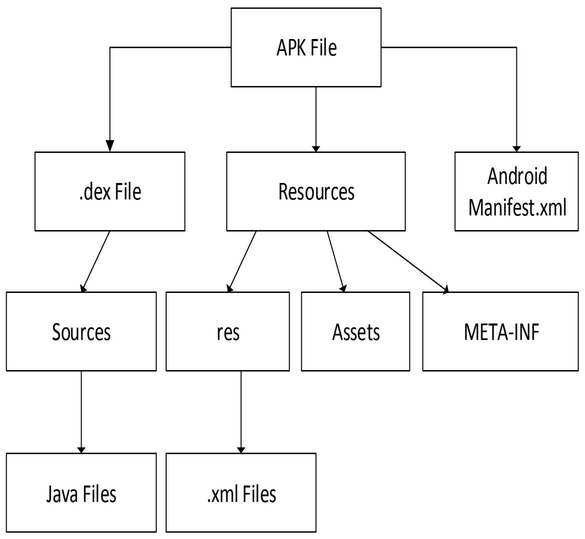

Fig 3: Decompiled APK File

5. CONCLUSION

Implementing Machine Learning models in the investigation of Malware Detection generation of the dataset is a critical part. The decompilation of files is needed for mining the features from the APK files and generating the dataset. The dataset generation process includes Android File collection, Decompilation, and Feature Mining phases. In the Android File collection phase we have already collected 15508 Malware files and 4000 Benign Files. In this paper, we have proposed an available automated solution for decompiling the files and also solves the complexity of handling and processing the big data.

We have accomplished the Decompilation of 15508 Malware files and 4000 Benign files using the online JADX Compiler [5] which is a fast, automated, flexible available solution that can process the unstructured data on cloud servers and store them compared to all other existing decompilation tools. The whole process of our Decompilation phase with the structure of decompiled APK files is discussed here. The Feature Mining and Machine Learning phases will be discussed later in other papers.

6. ACKNOWLEDGEMENTS

We would like to acknowledge our three students Ms. Sabera Kadiwala, Ms. Indushree Shetty, and Mr. Vraj Shah of GLSICT, GLS University who helped us in decompiling the APK files from the website [5] and dedicating their time for this task.

7. REFERENCES

[1] Prerna Agrawal, Bhushan Trivedi, "Automating the process of browsing and downloading APK Files as a prerequisite for the Malware Detection process ", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Vol 9, Issue 2, March - April 2020, pp. 013-017, ISSN 2278-6856

[2] Prerna Agrawal, Bhushan Trivedi, "Machine Learning Classifiers for Android Malware Detection", 4th International Conference on Data Management, Analytics and Innovation (ICDMAI) Springer AISC Series, New Delhi, Jan 2020. (Paper to be Published)

[3] Prerna Agrawal, Bhushan Trivedi, "Analysis of Android Malware Scanning Tools", International Journal of Computer Sciences and Engineering (IJCSE), Vol.7, Issue.3, pp.807-810, Mar 2019.

[4] Prerna Agrawal, Bhushan Trivedi, "A Survey on Android Malware and their Detection Techniques", Third International Conference on Electrical, Computer and Communication Technologies (ICECCT) IEEE, Feb 2019.

[5] Decompilation of APK Files, Online Link: http://www.javadecompilers.com/APK

[6] Meet Kanwal, Sanjeev Thakur, "An App Based on Static Analysis for Android Ransomware", International Conference on Communication and Automation (ICCCA), 2017.

[7] Neeraj Chavan, Fabio Di Troia, Mark Stamp, "A Comparative Analysis of Android Malware", 3rd International Workshop on Formal Methods for Security Engineering (ForSE), 2019.

[8] Suleiman Yerima, Sakir Sezer," Android Malware Detection Using Parallel Machine Learning Classifiers", 8th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST), Sept 2014.

[9] Zi Wang, JurongCai "DroidDeepLearner: Identifying Android Malware Using Deep Learning" Sarnoff Symposium IEEE, Sep 2016.

[10] J.D. Koli, "Randroid: Android Malware Detection using Random Machine Learning Classifiers", International Conference on Technologies for Smart City Energy Security and Power (ICSESP) IEEE, Mar 2018.

[11] JADX Decompiler Download Files and Download Instructions, Online Link: https://github.com/skylot/jadx

[12] Androguard Tool Project Download and Description, Online Link: https://pypi.org/project/androguard/

[13] Androguard tool API Docs, Online Link: https://androguard.readthedocs.io/en/latest/api/androguard.html

[14] K.V.Kanimozhi, Dr.M.Venkatesan, "Unstructured Data Analysis-A Survey", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 3, March 2015

[15] Min Chen, Shiwen Mao, Yunhao Liu, "Big Data: A Survey", Mobile Network Applications Springer, 2014, DOI: 10.1007/s11036-013-0489-0

[16] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, Wei Ding, "Data Mining with Big Data", IEEE Transactions on Knowledge and Data Engineering Vol 26, Issue 1, Jan 2014.

[17] Jinchuan Chen, Yueguo Chen, Xiaoyong Du, Cuiping LI, Jiaheng LU, "Big data challenge: a data management perspective", Frontiers of Computer Science Springer-Verlag Berlin Heidelberg, April 2013.

[18] Fan W, Bifet A, "Mining big data: current status, and forecast to the future", ACM SIGKDD Explor Newsletter, Vol 4, Issue 2, 2013, pp.1–5.