International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12– No.32, August 2020 – www.ijais.org

Feature Mining 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
The practice of using Machine Learning Methods in detecting Malware is growing massively. The prerequisite for implementing Machine Learning methods is the input of the dataset to it. A researcher needs to create a dataset of its own for performing Malware Detection using Machine Learning. Our dataset generation process includes Android File Collection, Decompilation, and Feature Mining Phases. We have already collected 15508 Malware Files and 4000 benign files in our Android File Collection phase and decompiled them in the Decompilation phase. Here we are discussing our Feature Mining Phase. So our goal in this paper is to select appropriate features for dataset generation. For the selection of proper features, we have also performed a Static Analysis process using online Malware Scanners. By using our static Analysis process we have selected a total of 215 features. Here we also propose the process of automating the Feature Mining from the APK files. We also have developed and implemented a Feature Mining Script in Python. Using the automated Feature Mining Script we have generated a final dataset of 16300 files. We have also discussed the working flow of feature mining script and in this paper.

Keywords
APK file, Malware, Dataset, Android, Machine Learning, Feature Mining, and Malware Detection.

1. INTRODUCTION
The usage of Machine Learning methods for malware detection compared to conventional methods is increasing immensely [3] [5]. Machine Learning methods are capable of detecting unknown malware also [4]. The Dataset is a prerequisite for using supervised machine learning methods for Malware Detection. For the proper investigation of malware, there is a need for features of different independent flavors. We tried to search for an existing dataset having different features but we found only the Drebin dataset that was available with lesser features. So for different independent flavors of features that we wanted to explore we created our dataset for getting better results. Our dataset generation process mainly involves Android File Collection, Decompilation, and Feature Mining phases. In the Android File Collection phase [2], we have successfully collected 15508 malware files from Android's world-famous Malware Datasets and 4000 benign files. In the decompilation phase[1], we have successfully decompiled all the collected files in Android File Collection. In this paper, we will discuss our Feature Mining phase. The Android File Collection and Decompilation phases are already discussed in our earlier papers.

There was also no proper mechanism available for proper selection of features for malware detection. So there was a need for Static Analysis to observe the behavior of current malware scanners on the malware files and for the selection of the proper features for malware detection. We have performed the Static Analysis process for proper feature selection using online malware scanners [4] in the first phase. In this paper, we present a solution to select appropriate features for malware detection from APK files. Using our Static Analysis process we have selected a total of 215 features. The Static Analysis process was performed using online malware scanners named Andrototal, AVC Undroid, VirScan, Hybrid Analysis, VirusTotal, and NvisoAPKScan [4] [6 - 11]. The Static Analysis results of all these online scanners were analyzed and based on that the features were selected. The main features selected are Permissions, Intents, and API calls.

The Permissions and Intents are extracted from the Manifest.xml file and the API calls are extracted from the java source code files. In the Android File Collection [2] the malware files consist of different varieties of malware families. The Drebin dataset [2] contains malware files from 179 different malware families and PRAGuard contains malware files from 50 different Malware families [2]. The features were selected based on the study of some malware families. Every application uses some permissions, API calls, and Intents. Specific malware exploits some typical permissions, API calls, and Intents. The features selected are both normal and dangerous based on the Malware and Benign APK files collected. Based on the static analysis results all the sensitive features were mainly selected. Table 1 consists of a list of some sensitive features. The sensitive features mainly contain READ and WRITE permissions of the different resources, Sending SMS permissions, Accessing Location permissions, getting access to Accounts, Internet and WIFI permissions, etc.

In this paper, we propose a process to automate the feature mining from the APK files. For automating the Feature Mining process we have created a Feature Mining Script in python which will retrieve the contents of the APK files search for the features, extract them and save into the dataset. Using our Feature Mining Phase we have successfully generated the dataset of around 16300 files. As discussed in the earlier paper our Android Malware Detection process also contains the Machine Learning phase and our final generated dataset will be provided as input to our generalized detection engine. The Machine Learning phase will be discussed in another paper.

This paper is divided into the following sections. Section 2 describes our Static Analysis phase and Feature Selection process. Section 3 describes our Feature Mining Phase and Section 4 describes the conclusion of the paper.

2. STATIC ANALYSIS FOR FEATURE SELECTION
The static analysis process plays an essential role in Feature Selection. The appropriate selection of features helps in

proper investigation of the malware for obtaining better results. Here we discuss our methodology of the Static Analysis Phase and Feature Selection process. The features are divided into 2 protection levels normal and dangerous. The normal level features are also considered as lower risk features that give requesting applications access to only application-level features with the minimum risk to all other applications, systems, or users. For normal level features, the system automatically grants them to a requesting application without taking the user's approval explicitly. The dangerous level features are also considered as higher risk features that give access to private user data or control over the device to the requesting application that can negatively impact the user. These type of features introduces potential risk to the system. An application will contain both the types of features but the segregation of dangerous features is equally important as malware exploits the higher risk features only for getting access to the private user data or control over the system and to harm them. The permission READ_EXTERNAL_STORAGE is considered to be dangerous as it may allow the application to access the user's private data of the external storage. The permission READ_CONTACTS is considered to be dangerous as it allows the application to access the user's private contacts. So malware exploits dangerous features and gains access to the user's private data or device.

2.1 Static Analysis

This section describes the Architectural flow of the Static Analysis Phase. The static analysis process is performed to investigate and select the appropriate features from the APK files which differentiates a Malware and benign file. Using this process one can determine the exact features to be selected and extracted from the APK file. This process is performed using online Android Malware scanners named Andrototal, AVC Undroid, VirScan, Hybrid Analysis, VirusTotal, and NvisoAPKScan [4] [6 - 11]. An APK file is selected from the Android Files Repository and uploaded on the website. The website after getting connected to the server sends the file for processing. The server after receiving the APK file scans the file and performs Malware Analysis. After processing the file the server responds to the website with the Malware Analysis results. The Malware Analysis results are saved to some physical location. All the collected Malware files in the Android File Collection phase are scanned with online Malware scanning tools by using the Static Analysis process for appropriate Feature Selection.

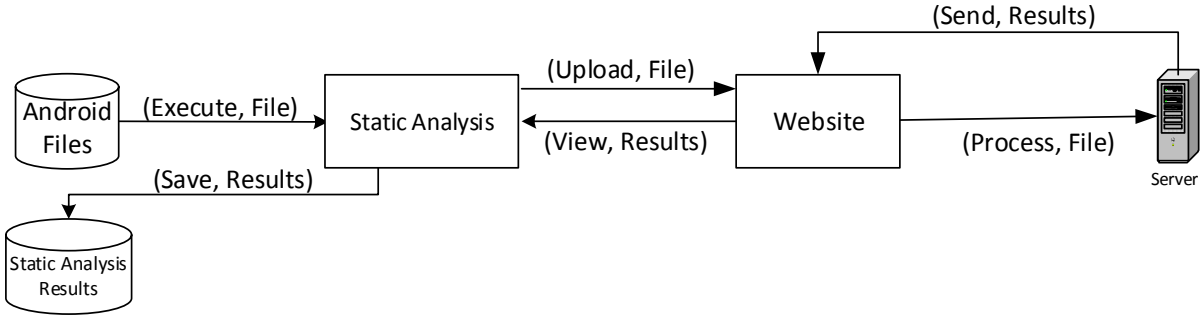

Figure 1: Architectural Flow of Static Analysis Phase

2.2 Feature Selection Process

Feature Selection is an important process for investigating Android Malware. The Static Analysis results stored in the Repository were analyzed and used for appropriate Feature selection. Sandroid Results [12] and Android Malware datasets [12] were also analyzed and studied for the Feature selection process. The classification of the Malware Detection Scanners and Sandroid Reports [6-11] [12] is based on the features like Requested Permissions, API calls responsible for used Permissions, Potentially Dangerous or Sensitive API Calls, Intents, Services, Broadcast Receivers, and Activities. Requested permissions are those which are requested by an application for using the resources or information outside its sandbox. If the application wants to use the internet than it will request the INTERNET permission. Used permissions are the subset of the requested permissions that are used by an application. API calls are the functions that are invoked through the code and require the permissions to use for their execution. The API calls which refer to dangerous permissions are known as Sensitive API calls. Sensitive API calls are the functions that are dangerous and provide access to the user's data or system's complete access to the application invoking that call. The Sensitive API calls will use dangerous permissions from the Manifest.xml file. The Network API call sendDataMassage() requests SEND_SMS permission to send the SMS. So the malware exploits the Network API call and sends the sensitive data outside through SMS. The non-sensitive calls are the functions that are normal and allows only application-level access and will use normal permissions from the Manifest.xml file. Intents are the asynchronous messages that allow you to interact with components of the same application as well as other components of different applications. A service runs in the background to perform long operations. Broadcast receivers allow you to register for system or application events. An activity represents a single screen in an android application and an application consists of multiple activities. Intents activate services, activities, and Broadcast Receivers and register their type using intent-filters in the Android Manifest.xml file. When a Malware File invokes Sensitive API calls than some dangerous permissions, activities, services, and broadcast receivers are often exposed and used.

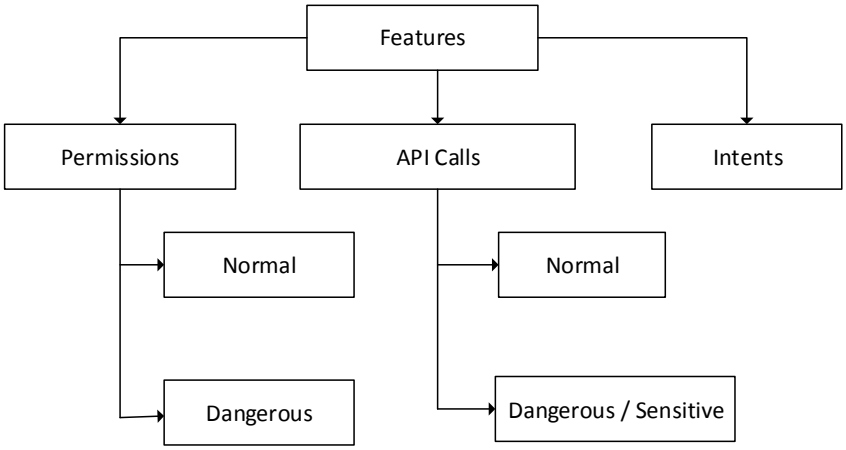

Figure 2: Classification of Features

Based on the classification of the Malware Detection Scanners Results and Sandroid Reports [6-11] [12] we have focused on the selection of features like API calls, Intents, and Permissions. Figure 2 presents the classification of features. The Permission protection level can be further classified into normal and dangerous [12]. According to the Malware Detection Scanners Results [6-11], the permissions are divided into Requested Permissions and Used Permissions. The requested permissions contain all the listed permissions that an application may request for use. It contains both Normal and Dangerous permissions. The used permissions are those which are used by the application and a Malware application contains Dangerous permissions. The API calls can be further classified into normal and dangerous/sensitive [6]. The sensitive API calls invoke the dangerous permissions and exploit them. If a normal SMS Application requests the permission of READ_EXTERNAL_STORAGE and BLUETOOTH than it suspects some malicious behavior. After performing the Feature Selection process we have selected a total of 215 features containing Permissions, API calls, and Intents. The next section contains the Feature Mining phase of all the selected features from the APK files. Table 1 describes some identified potentially dangerous permissions and Sensitive API calls [6 -11] [12].

Table 1: Sensitive Features

Features	Category
ACCESS_COARSE_LOCATION	Dangerous Permissions
CALL_PHONE	
CAMERA	
CHANGE_WIFI_STATE	
INTERNET	
MANAGE_ACCOUNTS	
READ/WRITE_CALENDAR	
READ/WRITE_CONTACTS	
READ/WRITE_EXTERNAL_STORAGE	
BLUETOOTH	
DISABLE_KEYGUARD	
READ_LOGS	
GET_TASKS	
SEND_SMS	
WRITE_MEDIA_STORAGE	
SYSTEM_ALERT_WINDOW	
INSTALL/UNINSTALL_SHORTCUT	
sendMultipartTextMessage	
getSubscriberId	
sendDataMessage	

getPackageInfo	Sensitive API Calls
getNetworkOperator	
getDeviceId	
getLine1Number	

3. FEATURE MINING FROM ANDROID FILES COLLECTION

The Feature Mining phase is crucial for our dataset generation process The Android file collection [2] contains a rich variety of malware files consisting of different malware families. The Drebin dataset [2] contains malware files from 179 different malware families and PRAGuard contains malware files from 50 different Malware families [2]. The Androzoo dataset also contains malware files from many families. We have collected various malware files from the world's most widely used malware datasets and ensured that our malware files collection contains a huge variety of different malware from different malware families. Some of the malware families covered are Contagio, Anserver Bot, BaseBridge, Bean Bot, Coin Pirate, Dogwars, Droid Coupon, Droid Dream, DroidKungFu1, DroidKungFu2, DroidKungFu3, DroidKungFu4, FakeNetFlix, FakePlayer, GamblerSMS, Geinimi, GGTracker, GoldDream, HippoSMS, JiFake, LoveTrap, NickyBot, NickySpy, Plankton, SMSReplicator, SndApps, Zitmo, OpFake, etc.

3.1 Feature Mining Script

For automating the process of Feature Mining we have developed a Feature Mining Script in Python for extracting the features from APK files. The script will examine the Android Manifest.xml and all the Java Source code files for feature extraction. It will extract all the available features from a decompiled APK file. Figure 3 describes the Feature Mining Script Flow. It is divided into different steps:

1. The process will start by looking for a Decompiled APK file in the Repository.
2. If the decompiled file is found then the Script will process the file.
3. The process will end if the Decompiled file is not found.
4. Once the decompiled file starts processing the script will find the Android Manifest.xml file.
5. If the Android Manifest.xml file exists the script will extract Permissions and Intents from it.
6. If the Android Manifest.xml file does not exist then the script will look for a new Decompiled APK file in the Repository.
7. After the processing of the Android Manifest.xml file, the script will find java files.
8. If the Java file exists then the script will extract API calls from it.
9. After extracting API calls from a single java file it will again look for another java file and this process will continue until all the Java files are processed.
10. After processing all the java files if another java file is not found then the script will look for a new Decompiled APK file in the Repository.
11. This script will process all the Decompiled APK files from the Repository and extract features from each file.

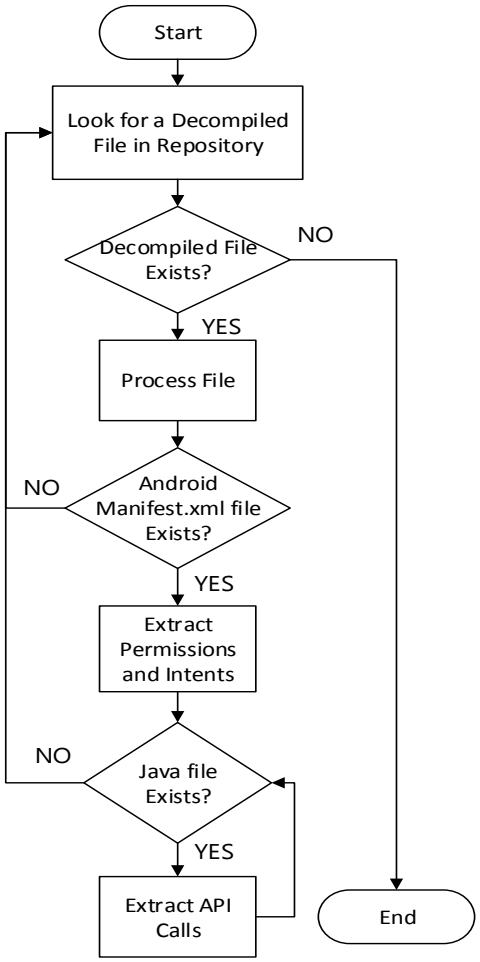

Figure 3: Feature Mining Script Flow

3.2 Feature Mining Process

This section presents the architectural flow of the Feature Mining Phase. Features represent the attributes of the Android files which helps to detect and classify the malicious files and benign files. Here features like Permissions, API calls, and Intents are extracted for Malware Detection. Figure 4 illustrates the overall flow structure for the Feature Mining phase. The selected features from the Static Analysis Results will be given as an input to the Content Retrieval of Features process. A decompiled file will be selected from the Decompiled Files Repository and Feature Mining Script will be executed. It will look for the Permissions, API calls, and Intents in Java files and Android Manifest.xml file in the Content Retrieval process. The Script file will look for all 215 features and extract the features from the files and save them into the dataset. Let V be the vector of all the selected features. For every *ith* Android file in the Android Files

Collection, we generate a binary sequence *Si = {f1, f2… fj}* and

$fj = \{1\ if\ jth\ feature\ exists, 0\ otherwise\}$

If the feature exists 1 is stored for that specific feature otherwise 0. Also, we consider one more variable C for classification of each Android file where C € {Malware, Benign}. The value *s* indicates Malware application and *b* for benign application. The vector V for each Android file is stored in a CSV file and the final features dataset is prepared for all malware and benign applications. The Malware files are collected from the world's largest datasets. As we have a huge number of files downloaded from different malware datasets available and due to bulk files in each dataset containing different malware families some files may be repeated in different datasets. So there is a possibility of having duplicate entries in the dataset. We have also applied the process of removing the Duplicates entries from the dataset for obtaining more accurate results in malware detection. Using this Feature Mining Phase we have generated a final dataset of around a total of 16300 files.

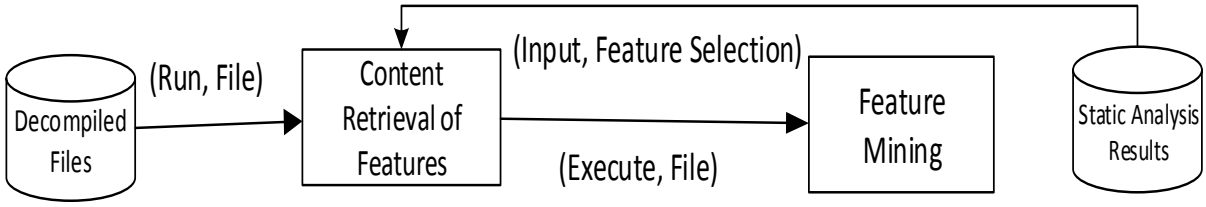

Figure 4: Architectural Flow of Feature Mining Phase

4. CONCLUSION

The practice of using Machine Learning Methods for detecting the malware requires a dataset that will train the model and test it for providing better results. For that, a researcher needs to create a dataset of its own. Our dataset generation process mainly includes Android File Collection, Decompilation, and Feature Mining phases. In the Android File Collection phase, we have collected 15508 Malware files and 4000 benign files and decompiled them in the Decompilation phase. The Android File collection and Decompilation phases are already discussed in previous papers. Here we have discussed our Feature Mining phase and proposed a process for automating the feature mining from APK files. We have also proposed and implemented a Feature Mining Script in Python. For the appropriate feature selection from the APK files, we have also conducted the Static Analysis process using the online malware scanners.

We have presented the solution of the feature selection process by implementing our Static Analysis phase and selected a total of 215 features by studying the results of the Static Analysis[4] and analyzing different malware reports of Sandroid [12]. By implementing our Feature Mining phase we have generated a final dataset of around 16300 files. We have also shown the working flow of the feature mining script The machine learning phase will be discussed in another paper.

5. REFERENCES

[1] Prerna Agrawal, Bhushan Trivedi, "Unstructured Data Collection from APK files for Malware Detection", International Journal of Computer Applications (IJCA), Vol 176, Issue 28, June 2020, pp. 42-45, ISBN 973-93-80901-12-5, ISSN 0975 – 8887, DOI 10.5120/ijca2020920308

[2] 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 2278-685.

[3] 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.

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

[5] 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.

[6] "AVC UnDroid Online Scanner", Online Link: https://undroid.av-comparatives.org

[7] "AndroTotal: Scan Android Application", Online Link: http://andrototal.org.

[8] "VirusTotal: Analyse suspicious files", Online Link: https://www.virustotal.com

[9] "NVISO ApkScan: Scan Android Applications for Malware", Online Link: https://apkscan.nviso.be/

[10] "VirSCAN.org: Submit and scan your file", Online Link: http://www.virscan.org

[11] "Hybrid Analysis Online Scanner", Online Link: https://www.hybrid- analysis.com

[12] "Sandroid: Android Application Analysis System", Online Link: http://sanddroid.xjtu.edu.cn/#overview

[13] "Machine Learning Datasets", Online Link: https://figshare.com/articles/Android_malware_dataset_for_machine_learning_1/5854590/1