International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Malware Detection System using ID3 Algorithm for Android

Trupti D. Deshmukh[1], Vrunda K. Bhusari[2]

[1]P.G. Student, Savitribai Phule Pune University, Department of Computer Engineering, BSIOTR, Wagholi, Pune, Maharashtra, India

[2]Assistant Professor, Savitribai Phule Pune University, Department of Computer Engineering, BSIOTR, Wagholi, Pune, Maharashtra, India

***Abstract:** The popularity of Android OS for mobile is inviting the threats such as malwares. The term 'malware' is defined as variety of form of intrusive software. Malware is any program or data which affects the working of a device. Thus malware detection is the invigorating issue in the computer security. To avoid the malware attacks different anti-malwares are also have been developed. But there is a need to evaluate these anti-malwares which can be done by using Droid Chameleon. Droid Chameleon does the transformation of malwares automatically and helps to check the efficiency of anti-malware. Here we propose a system that identifies the malicious apps affected due to malwares. The permissions given by android apps are used as the dataset. The ID3 algorithm is used to apply mining on these datasets i.e. training is provided to generate the trained dataset. The Admin will take care of new entries of malwares as well as apps in the database. The results are shown as whether the given app is malicious or not.*

Keywords: Malware, Anti-malware,Android, mobile

1. Introduction

The adoption rate of mobile devices continues to mount upward, with Android leading the way. Google provides this open-source operating system that is leading in market. More than half smartphones are found which includes Android OS. The research firm Strategy Analytics found that 81.3 percent, or 204.4 million, of smartphones launched in the third quarter of 2013 were powered by Android. Android is an operating system which is used for smartphones and tablets. It is based in Linux kernel with the user-friendly feature. Android applications are developed in Java native interface. All the classes of Android are packed together in single .dex file which is called Dalvik bytecode instead of running on Java bytecode.

The android smart phones are largely targeted by the malware attackers, among the mobile phone users and attackers. The reason behind it is, the open platform is provided by android applications market to all the application. When you download any app into your android phone malware gets entry in the system. Also, it can also become serious threat to businesses. A third person can use a malware infected smart phone and use it as a proxy or a gateway to enter into a restricted business network. Some of the dangerous malware attacks are:

1) Fake Banking Apps: This attracts the customers into entering their online account login details.
2) Android.Geinimi: Genimi is a malware which corrupted many legitimate Android games on Chinese download sites.
3) DroidDream: It infectes devices, breaks the android security sandbox and steals data.
4) AndroidOS fake player: It shows that it is working like a media player and then silently sends SMS to premium SMS numbers.

Polymorphism is technique to avoid detection tools by performing transformation on malwares but with same code. These attacks are being a serious problem for both traditional desktop and server systems. The existing anti-malware softwares are evaluated by DroidChameleon, a systematic framework with several common transformation techniques [1]. DroidChameleon does the transformation of Android application automatically. The term transformation here refers to semantics preserving changes of the program. Here we propose a system which will detect the malicious apps based on the permissions given by Andorid OS. The capabilities of any Android apps are strictly constrained by the permissions users grant to them [2]. Therefore, it will be fascinating to check top permissions requested by malicious apps in the dataset. We propose a system which will use these permissions as an input to the ID3 algorithm, based on which the malicious functionality of the app is recognized.

2. Literature Review

Several studies have been contributed to reduce the malware attacks and to increase the performance of the mobile devices.

2.1 ADAM

ADAM is an extensible platform which is automatic, generic and able to evaluate the Android malware detection systems. ADAM is able to automatically transform an original malware sample to different variants using repackaging and obfuscation techniques in order to evaluate the strength of different anti-virus systems against malware mutation [3]. ADAM is built by connecting different building blocks such as transformation, scanning and analysis of malwares. These blocks help to test different anti-malwares against malware samples. But ADAM is not always able to avoid anti-malware tool. So, it will not always provide the better detection mechanism.

Licensed Under Creative Commons Attribution CC BY

2.2 Automatic Code Obfuscation

It is done to protect the messages which help to preserve privacy policies between sender and receiver [4]. As shown in Figure 1. the obfuscation technique provides the protection of messages between Alice and Bob. By using source message object code is created which is then obfuscated and passed to the server. The server sends it to Bob i.e. client. The reverse operation is done by Bob to get the original source code.

Although the system can easily trace the software pirates but it remains secret until the powerful deobfuscator to be built. So, obfuscated software version release must be within short period.

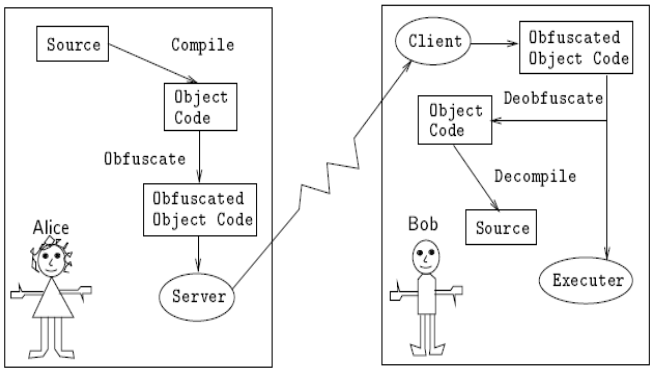


Figure 1: Protection through obfuscation [4].

2.3 Malware Detection by Semantics-preserving

As per the name semantics-preserving malware detectors use pattern-matching technique to search the obfuscations made by hackers [5]. The hackers use obfuscation ; so the detector is used to find out malicious behavior of a program. The detector is easy to be understood by detectors as it is based on syntax analysis but needs large databases to save the patterns of malicious instructions.

2.4 Automatic Security Analysis of Smartphone Applications

The AppPlayground tool is used to do the automation of security analysis. AppPlayground does the integration of multiple components comprising different detection and automatic exploration techniques for this purpose [6]. It does the analysis of security with large number of application, but also it is less effective for automatically detecting privacy leaks and malicious functionalities in application.

2.5 Crowdroid

Burguera et al., [7] proposed behavior –based malware detection system for Android. They used detector which is embedded in an overall framework for a collection of traces collected from unlimited real users based on crowdsourcing. The system analyzed collected data in central server using two types of data sets: artificially created malwares and real malwares. It is an effective method of isolating the malware as well as alerting the users about the downloaded malwares. When it is actually going to apply on mobile, it might result an extra overhead in the processor, causes a faster battery drain.

3. System Implementation

3.1 System Architecture

The Figure 2 shows the overall system architecture. As per shown in figure the process flow goes according to the system architecture. The system architecture includes following components:

1. Application Permissions and Data Import

The different android application permissions are fetched from android applications. These permissions are used as dataset for process.

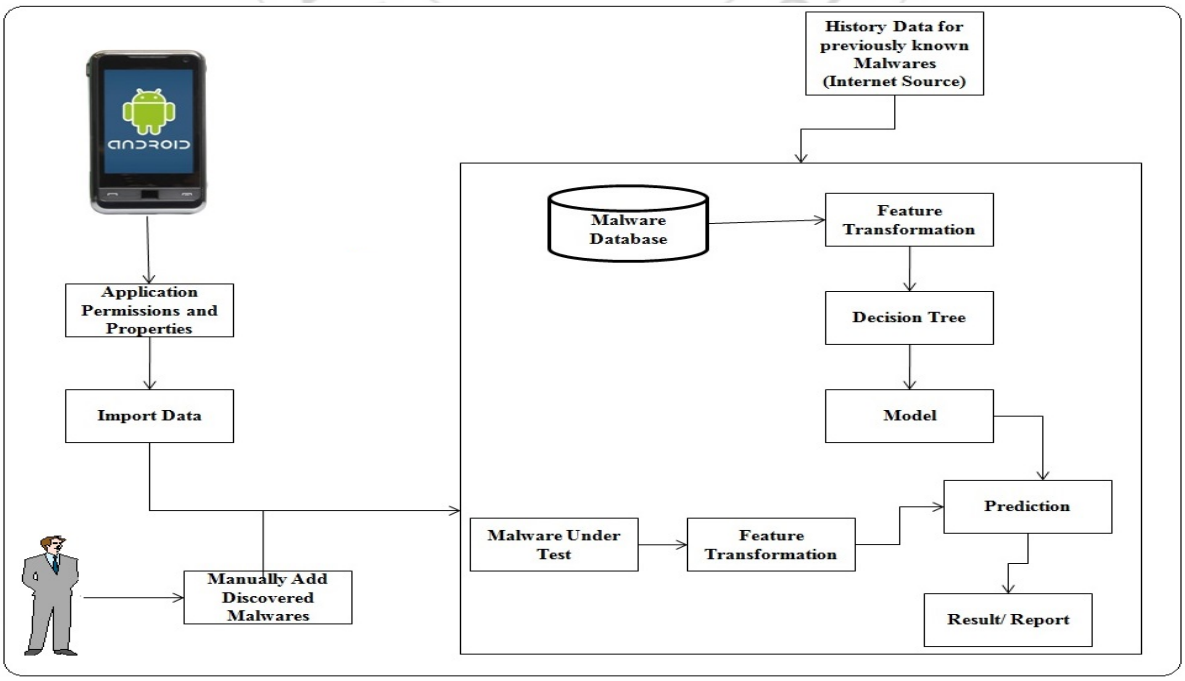


Figure 2: System Architecture.

Licensed Under Creative Commons Attribution CC BY

2. Malware Database

The malware samples will be stored at database.

3. Feature Transformation and Decision Tree

The Decision tree is generated using ID3 algorithm [8]. The permissions of application are given as an input for feature transformation. The permissions are training datasets for the algorithm. These permissions are the initial inputs for feature transformation.

4. Prediction

The datasets of application will be analyzed here and the related prediction of malware samples is done with the help of Decision Tree.

5. Report

The result of whether the given Android App is malicious or not will be shown.

3.2 Algorithm

The classifier algorithm ID3 works as follows:
Assumptions:

I: is the set of input attributes i.e. permissions (Here we will consider the permissions given by android to app to find out classification attribute.)
O: is the output attribute
T: is a set of training data
Function ID3: returns a decision tree

```
Function ID3 (I, 0, T) {
if (T is empty) {
return a single node with the value "Failure";
}
if (all records in T have the same value for O) {
return a single node with that value;
}
if (I is empty) {
return a single node with the value of the most frequent value of
O in T;
}

Compute the information gain for each attribute in I relative to T using permissions;
let X be the attribute with largest Gain(X, T) of the attributes in I;
let {x_j| j=1,2, .., m} be the values of X;
let {T_j| j=1,2, .., m} be the subsets of T when T is partitioned
        according the value of X;
        return a tree with the root node labeled X and
        arcs labeled x_1, x_2, .., x_m, where the arcs go to the
        trees ID3(I-{X}, O, T_1), ID3(I-{X}, O, T_2), .., ID3(I-{X}, O, T_m);
}
```

The Entropy and Information gain are calculated as:

1. Entropy: Entropy gives the measure in information theory, which characterizes the unwanted attributes of an arbitrary collection. If target attribute takes on r different permissions (values), then entropy X relative to this r-wise classification is defined as:

$$\text{Entropy}(X) = \sum_{i=1}^{r} - p_i \log_2 p_i \quad (1)$$

2. Information Gain: The information gain, Gain(X, T) of an attribute T (here the permission), relative to the collection of examples X, is defined as:

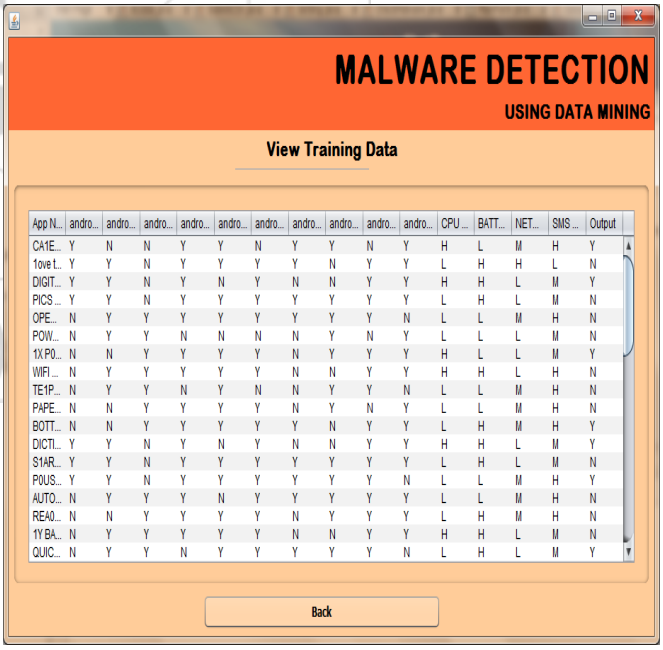
$$Gain(X,T) = \text{Entropy}(X) - \sum_{v \in Values(X)} \frac{|X_v|}{|X|} Entropy(X) \quad (2)$$

Where, Values (T) is the set of all possible values for attribute T and X_v is the subset of X for which the attribute T has value v.

4. Results

The system is developed by using JAVA (Version JDK). The development tool used is NetBeans for desktop application. Eclipse is used for development of Android Application for smartphone. The database used for storing the malwares is Apache Tomcat. The experiments are performed on Core2Duo Intel processor 2 GB RAM. The results are shown as the report whether the app is malicious or not.

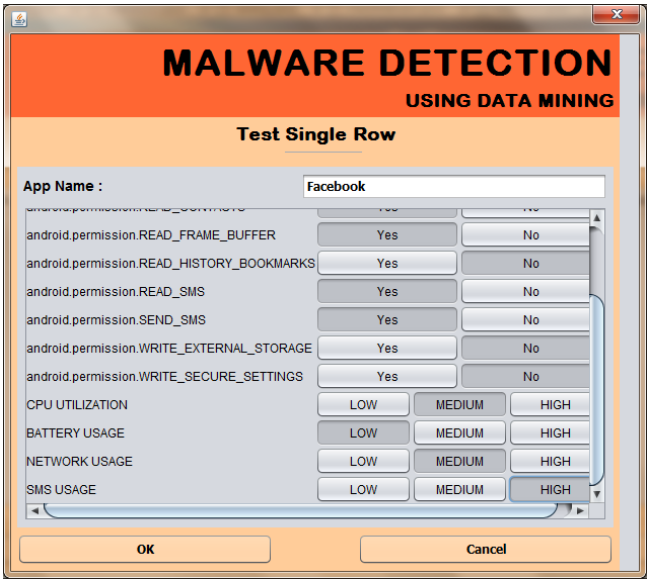
4.1 Dataset of Android permissions for Training



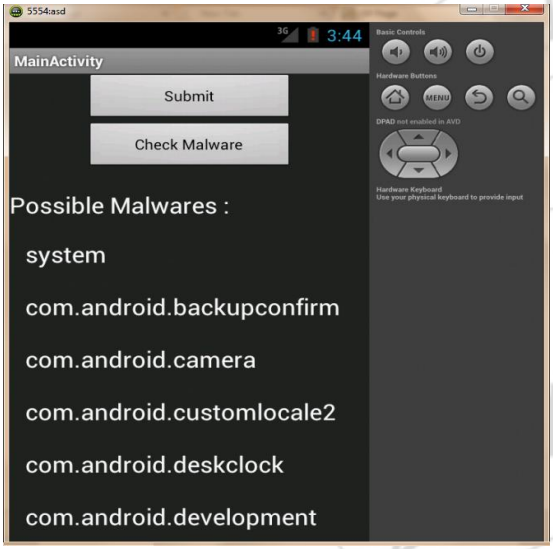
App N...	andro...	andro...	andro...	andro...	andro...	andro...	andro...	andro...	andro...	andro...	CPU ...	BATT...	NET...	SMS ...	Output
CA1E...	Y	N	N	Y	Y	N	Y	Y	N	Y	H	L	M	H	Y
1ove t...	Y	Y	N	Y	Y	Y	Y	N	Y	Y	L	H	H	L	N
DIGIT...	Y	Y	N	Y	N	Y	N	N	Y	Y	H	H	L	M	Y
PICS ...	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	L	H	L	M	N
OPE...	N	Y	Y	Y	Y	Y	Y	Y	Y	N	L	L	M	H	N
POW...	N	Y	Y	N	N	N	N	Y	N	Y	L	L	L	M	N
1X P0...	N	N	Y	Y	Y	Y	N	Y	Y	Y	H	L	L	M	Y
WIFI ...	N	Y	Y	Y	Y	Y	N	N	Y	Y	H	H	L	H	N
TE1P...	N	Y	Y	N	Y	N	N	Y	Y	N	L	L	M	H	N
PAPE...	N	N	Y	Y	Y	Y	N	Y	N	Y	L	L	M	H	N
BOTT...	N	N	Y	Y	Y	Y	Y	N	Y	Y	L	H	M	H	Y
DICTI...	Y	Y	N	Y	N	Y	N	N	Y	Y	H	H	L	M	Y
S1AR...	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	L	H	L	M	N
P0US...	Y	Y	N	Y	Y	Y	Y	Y	Y	N	L	L	M	H	Y
AUTO...	N	Y	Y	Y	N	Y	Y	Y	Y	Y	L	L	M	H	N
REA0...	N	N	Y	Y	Y	Y	N	Y	Y	Y	L	H	M	H	N
1Y BA...	N	Y	Y	Y	Y	Y	N	N	Y	Y	H	H	L	M	N
QUIC...	N	Y	Y	N	Y	Y	Y	Y	Y	N	L	H	L	M	Y

4.2 Single Entry of App for Testing the single App

Licensed Under Creative Commons Attribution CC BY



4.3 The result of malware on Android App



4.4 Graph of Retrieved Objects

Table 1: Retrieved Objects

Dataset Name	*Actual Objects*	*Retrieved Objects*	*Correct Retrieved Objects*
Malware Apps	20	18	17
Nonmalware-Apps	25	24	23

Table 2: Total Accuracy

Dataset Name	*Precision*	*Recall*
Malware Apps	0.94444444	0.85
Nonmalware-Apps	0.95833333	0.92
Total	0.951388889	0.885
Accuracy ercentage	0.885	-

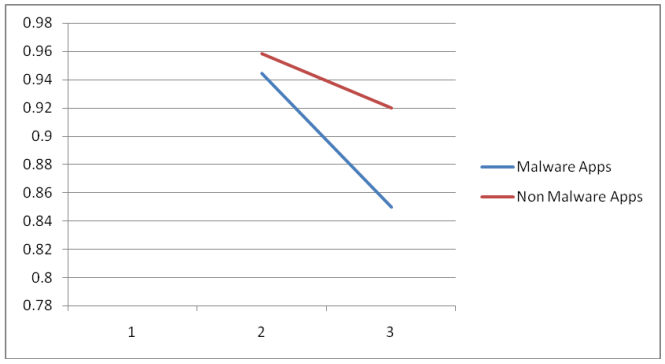


Figure 1: Accuracy Plotted

The Table 1 shows the correct retrieved objects i.e. which apps are found to be malicious and which are not correctly. The actual retrieved objects are showing how much malicious apps are found correctly and how much malicious apps are not found correctly.

Table 2 shows the accuracy plotted in terms of Precision and recall.

5. Conclusion

Mobile malwares are attacking the android systems which cause the vulnerability to the whole application. To avoid this we have proposed classifier based anti-malware which will detect the malwares with different functionalities. The permissions of apps with malicious characteristics are used for finding the malicious behavior. Using the ID3 algorithm the most permission requests used by an app help to classify the smartphone application into malicious and non-malicious application.

As a future work a more comprehensive anti-malware tool is possible to implement using artificial- intelligence. There is a scope to detect large number of malwares.

6. Acknowledgement

I would like to express my sincere gratitude to my guide Prof. Vrunda K. Bhusari for her continuous support, patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and writing of this paper.

References

[1] Vaibhav Rastogi, Yan Chen, and Xuxian Jiang, "Catch Me If You Can: Evaluating Android Anti-Malware Against Transformation attacks", IEEE transactions on information forensics and security, VOL. 9, NO. 1, Jan 2014.

[2] Y. Zhou and X. Jiang, “Dissecting android malware: Characterization and evolution,” in Proc. IEEE Symp. Security Privacy, May 2012, pp. 95–109.

[3] M. Zheng, P. Lee, and J. Lui, “ADAM: An automatic and extensible platform to stress test Android anti-virus systems,” in Proc. DIMVA, Jul. 2012, pp. 1–20.

Licensed Under Creative Commons Attribution CC BY

[4] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," Dept. Comput. Sci., Univ. Auckland, Auckland, New Zealand, Tech. Rep. 148, 1997.
[5] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant, "Semantics-aware malware detection," in Proc. IEEE Symp. Security Privacy, May 2005, pp. 32-46.
[6] V. Rastogi, Y. Chen, and W. Enck, "AppsPlayground: Automatic security analysis of smartphone applications," in Proc. ACM CODASPY, Feb. 2013, pp. 209–220.
[7] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behaviorbased malware detection system for android," in *Proc. 1st ACM Workshop Security Privacy Smartphones Mobile Devices*, 2011,pp. 15–26.
[8] David McG. Squire,, "CSE5230 Tutorial: The ID3 Decision Tree Algorithm," Monash University, Faculty of Information Technology, CSE5230 Data Mining Semester 2, August 26, 2004.

References

Ms. Trupti D. Deshmukh received the Bachelors degree (B.E.) Computer Engineering in 2013 from VPCOE, Baramati. She is now pursuing Masters degree (Computer Engineering), from BSIOTR, Wagholi, Pune, Maharashtra.

Prof. Vrunda K. Bhusari received her M.Tech(Computer Engineering) from Bharati Vidyapeeth, Pune and now she is working as Assistant professor, Department Of Computer Engineering, Bhivarabai Sawant Institute of Technology & Research, Wagholi, Pune, Maharashtra.. Her research areas include Network Security.

Licensed Under Creative Commons Attribution CC BY