Testing Android Anti-Malware against Malware Obfuscations

Gunjan Kapse
Computer Science Department
Jayawantrao Sawant College of Engineering
Pune, India

Aruna Gupta
Computer Science Department
Jayawantrao Sawant College of Engineering
Pune, India

ABSTRACT
There is an increasing threat of malware on mobile. Since Android is the most popular and maximum sold mobile phone, the malware attack on Android mobile is increasing day by day. The commercial antimalware products available in the market can detect common and old malwares easily. Different types of transformations can be applied to a malware which make it difficult for antimalware to detect. The different transformations can be majorly classified into 1.Trivial transformations, 2.DSA (Detectable by static analysis) transformation, and 3.NSA (Not detectable by static analysis). Researchers have evaluated the strength of different commercial antimalware tools by passing the transformed malware samples to them and found that all the antimalware tools can be evaded by applying either a single transformation or combination of transformations. We propose to add more malware samples in the framework namely KMIN, PJAPPS, ROOTEXPLOIT, and YZHC. These are Android malware samples. We shall apply Trivial, DSA and combination of DSA transformations to them. After transformation, we pass them to Android mobile antimalware products Aegis Lab, Bkav Security, CM Security, Rinix, and Hornet and systematically evaluate them regarding their resistance against various transformations.

Keywords
Malware, DSA, NSA, Mobile, Android.

1. INTRODUCTION
Android is an open source, Linux based lightweight operating system. First version of Google Android was launched in 2008. Earlier, the mobile phones were not connected to internet, so there was no malware spread. But since the discovery of smart phones which offer high computing power and better connectivity, the propagation of threat has increased. In September 2008, security research group from Chinese university announced first attack for Android. Since 2011, the Android malware has spread over very fast [6]. Now mobile virus is becoming a big concern. Cabir virus appeared when Bluetooth was used to communicate between devices. SMS Trojans on Android are DroidDream, DroidKungFu, and Plankton. They try to acquire root access of the device and can remotely install applications on device without any user intervention. DroidKungFu and Plankton also sends sensitive information of device to the attacker. Android malware Geinimi was discovered in 2010. If the Geinimi malware is installed on a phone, it can receive commands from a remote server which allow the owner of that server to control the phone. It can collect specific information including location coordinates and unique identifiers for the device (IMEI) and SIM card (IMSI). Spyware is also a kind of malicious code which collects the user's private information.

2. OBFUSCATION TECHNIQUES
2.1 Trivial obfuscations
Trivial obfuscations [1] are the kind of transformations in which the application code is not modified. Modifications like changing the signature of APK, changing the package name in AndroidManifest.xml, Disassemble and reassemble the Android APK are performed in trivial transformation. Apktool is used to obtain the AndroidManifest.xml and Smali code from the APK [8, 9]. Package name in Android Manifest.xml can be changed in trivial obfuscation. Signature of APK can also be changed after opening it in Android SDK and exporting signed application package.

2.2 DSA Obfuscations
After running the Apktool, dex2jar tool can be used to get the jar file from classes.dex. The java code can be obtained from this jar. DSA obfuscation involves changing the code thus obtained. DSA stand for Detectable by static analysis. Changing method name, class name or identifier name are such obfuscations. Encoding data, junk code insertion, reordering code can also be performed under this category.

2.3 NSA obfuscations
It is the category of obfuscation which cannot be detected by static analysis. Following are the obfuscations in this category.

2.3.1 Reflection
It involves converting a method call into call through reflection. Modern software languages like VB.NET and Java have reflection API. With such an API we can query and access types and members at runtime identified by their name. Reflection is the ability of a program to examine and modify the structure and behaviour (specifically the values, meta-data, properties and functions) of the program at runtime.

2.3.2 Byte code encryption:
It involves storing the application code in encrypted form which is decrypted at runtime with decryption routine. The antimalware has the decryption routine. User defined class loader loads the dex file and decrypts it. Since it is directly decrypted at runtime, static analysis of code is not possible.

3. SYSTEM ARCHITECTURE
To assess the strength of different antimalware tools available in market, we obfuscate the malware in different forms and test them with the antimalware tools. Figure 1 shows the process of applying different obfuscation methods to malware samples. The sequence of execution of applying transformation and testing them is:

i Apply trivial obfuscation techniques to malware samples one by one and test the obfuscated malware with antimalware tool. If the transformed malware is

detected, then stop. Else, apply next trivial transformation.

ii Apply the different DSA obfuscation techniques to malware sample one by one. If the antimalware is able to detect the transformation then stop. Else apply next DSA transformation.

iii Apply combination of DSA transformations. Test the transformed malware with Antimalware. If malware is detected, then stop. Else apply next combination of DSA obfuscation.

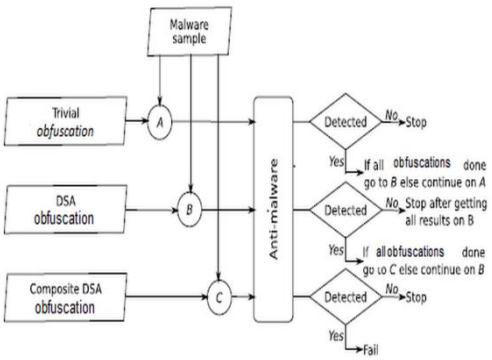

Fig 1. Assessing anti malware

Results of testing different antimalware tools against various obfuscations of DroidDream malware are shown in figure 2. Cross indicate failure of detection. Other malwares tested with transformations are Geinimi, Fakeplayer, Bgserv, Basebridge, and Plankton. Results show that all of the antimalware products can be evaded after applying some transformation or combination of transformation.

We plan to do comprehensive study and test one more set of antimalware products against obfuscation attacks with another set of malware samples. The antimalware tools which we shall use are Aegis Lab, Bkav Security, CM Security, Rinix, and Hornet which are android mobile antimalware products. The malware families to be used are KMIN, PJAPPS, ROOTEXPLOIT, and YZHC.

	AVG	Symantec	Lookout	ESET	Dr. Web	Kaspersky	Trend M.	ESTSoft	Zoner	Webroot
Repack										
Disassemble and reassemble									x	
Rename package	x							x	x	
Encrypt payload & native exploit			x						x	
Rename identifier		x							x	x
Encode string & array data									x	
Reorder code									x	
Call indirection									x	
Insert junk code									x	
Rename id+Encrypt p&ne		x	x	x					x	x
Encrypt p&ne + encode str&arr			x			x			x	
Encrypt p&ne + Rename file			x				x		x	
Encrypt p&ne + Call indirection			x		x				x	

Fig 2. Droid Dream obfuscations and Result of Testing with Anti-malware

4. MALWARE DETECTION TECHNIQUES

4.1 Signature based detection

It matches the application's signature which is actually a sequence of bytes with the antimalware signature. It matches, and then the application is detected as malware. This technique is used to detect variants of same malware family or we can say existing malware or transformations of that malware. Advantage of Signature based detection is that it has less false positives means very less wrong detection. It can detect the malware fast. Polymorphic malware transforms/mutates itself, keeping the original code intact[7]. The polymorphic engine of virus generates mutants of virus every time it is executed. For each variant of malware, the signature is different. So it is hard to detect polymorphic virus. Disadvantage is that, it cannot detect new and unfamiliar malware.

4.2 Behaviour based detection

It detects patterns in a dataset that do not conform to a fixed normal behaviour. It tries to estimate abnormal behaviour by checking the deviation of behavior from normal behaviour and comparing it with a threshold. If it exceeds the given threshold, then it is categorized as malware. Advantage of Behaviour based detection is that it can detect previously unseen or new malwares. Disadvantage being that there can be many false positives that is, the tendency to classify a correct application as malware is high. Also it need big set of training data to construct normal profile.

5. RELATED WORK

Mihai Christodorescu [5] introduced technique Semantic aware malware detection. The method first disassembles the binary program. It then generates CFG (Control Flow graph) of instruction sequence. The malicious behaviour is represented with templates with instruction sequences in which variables and symbolic constants are used. For each template node a matching node in program is searched and

mapped. The node matching is done based on expressions used. Also the template variables and program expressions should have similar update pattern. Thus bindings are generated by unification of template and program node. False positive evaluation was done on 2000 binaries yielding no false positives. Obfuscation resilience was evaluation was also performed with Garbage insertion codes like nop insertion, stack op insertion and math op insertion. Different variants of obfuscations were generated and tested. The tool could catch all the obfuscations except math op replacement transformation.

Advantages

i Obfuscations like register renaming can be detected by this method.

ii Low value of false positives rate.

iii Junk code insertion can be found and detected. Since it checks the flow of data and update patterns, such type of transformations can be detected.

Disadvantages

i IT mandates the IR instructions of template and that of the program to be same. No equivalent substitute of that operation is accepted. Example, if template has x=x+constant, then it is matched with program if same + operation is found and not any other equivalent operation.

ii It enforces memory updates should be of same order for both program and the template.

Zarni Aung, Win Zaw[4] introduced new model Permission-Based Android Malware Detection. It is based on 3 phases. 1. Feature selection 2. K means clustering of the selected features 3. Classification of data set as good ware or malware. This was followed by performance evaluation with accuracy, precision and recall. The different permissions requested by an application are analysed like

android.permission.INTERNET,

android.permission.CHANGE_CONFIGURATION,

android.permission.WRITE_SMS,

android.permission.SEND_SMS,

android.permission.CALL_PHONE.

The features which are extracted are stored in dataset with ARFF format. Best features are selected with largest value of Information gain. Then machine learning is used for malware classification and detection. K means clustering and Decision tree classification is used.

Advantage of this method is, it is able to detect new malware. It can also detect unfamiliar malware.

Disadvantage being, it can have higher number of false Positives.

Yu Feng, Saswat Anand [3] presented Apposcopy: It is a again a semantic way of identifying an application as goodware or malware. It aims at detecting a specific class of malware which steals private information of user. The algorithm combines static taint analysis and creation of Inter component call graph to detect Android malware application.

There are 2 steps of doing this analysis:

Step1: Construction of ICCG of given application: ICCG indicate the control flow of the signature. Nodes of ICCG represent different components. The edges between components indicate the communication between the components. For example component X starts component Y by calling start Activity method of Android.

Step2: Static taint analysis: It indicates the truth values of data flow in the signature. Example of Data-flow predicates, is flow (i, Sr, j, Sn), express that the application contains flow from source Sr in component i to sink Sn in component j. It may indicate that component sends the device and subscriber id of the phone over the Internet.

Advantages

i Good accuracy of detection.

ii Can detect transformations such as: Change of component, method names.

iii Redirection of method calls through proxy method.

iv Encryption of component names, data type values of intents.

Disadvantages

i Unknown malware family cannot be detected by this approach.

ii Not possible to detect below obfuscations:

a. Dynamic loading of code at runtime.

b. Change of method or class name in combination with Reflection cannot be detected.

iii Detection of malware instantly on smart phones is not possible.

Daniel Arp, Michael Spreitzenbarth [2] introduced method DREBIN. Since monitoring of application at runtime impede lots of resources, DREBIN does broad static analysis of application for malware detection. It extracts different features from application's AndroidManifest.xml and code. It then maps the features into vector space where each dimension is either 0 or 1. Number of dimension equals to total number of number of features. Applications with similar features thus reside close to each other. It uses SVM (Support Vector Machines) as the learning based detection method. The method also provides explanations about the detection results. The features extracted from xml and code is given different weights based on their severity. The detection result explanation is based on this weight. The runtime performance of the method is evaluated by applying on Google play store applications. Drebin analyses an application in 10 seconds. On older models, it takes 20 seconds to evaluate. Renaming of components between learning phase and detection phase may decrease the result strength.

Advantages

i This can be directly implemented on Smart phones and can be run when applications are downloaded.

ii It can also give explanations about the detection. For example The application sends SMS message. Communication with host gval.amict.net etc.

iii Number of false alarms is less. The method has high accuracy. It can detect new and unknown malware well.

Disadvantages

i Not able to detect obfuscations which are not detectable by static analysis like Reflection and byte code encryption.

ii Latest malware sample collection is required.

iii If benign/good features or fake variants are included into malicious applications, the detection score of the method is lowered.

6. CONCLUSION

We studied the different malware obfuscation methods which are trivial, DSA and NSA obfuscations. We also introduced basic malware detection techniques which are signature based and behaviour based detection. We evaluated the strength of commercial antimalware products against different malware obfuscation methods. The result of evaluation shows that all the commercial antimalware products can be evaded by applying some transformation or combination of transformations. We shall collect more malware samples namely KMIN, PJAPPS, ROOTEXPLOIT, and YZHC and antimalware tools for android namely Aegis Lab, Bkav Security, CM Security, Rinix, and Hornet and assess the strength of antimalware products against transformed malware.

7. REFERENCES

[1] V. Rastogi, Y. Chen, and X. Jiang, "DroidChameleon: Evaluating Android anti-malware against transformation attacks", *Proc. ACM ASIACCS*, May 2013, pp. 329–334.

[2] D. Arp, M. Spreitzenbarth, M. H□ubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket", NDSS, Internet Society, February 2014, USA

[3] Yu Feng, Saswat Anand, Isil Dillig, Alex Aiken, "Apposcopy: Semantics-Based Detection of Android Malware through Static Analysis," in *ACM SIGSOFT Int. Symp*, November, 2014

[4] Zarni Aung, Win Zaw, "Permission-Based Android Malware Detection," in IJSTR Vol 2, Mar 2013

[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] (2013, Feb.).*CNET* [Online]. Available: http://news.cnet.com/8301-1035_3-57569402-94/android-ios-combine-for-91-percent-of-market/

[7] Symantec, Mountain View, CA, USA. (2013, Dec. 3). *Server-Side Polymorphic Android Applications* [Online]. Available: http://www.symantec.com/connect/blogs/server-side-polymorphicandroid-applications

[8] (2013, Dec. 3). *Smali: An Assembler/Disassembler for Android's Dex Format* [Online]. Available: http://code.google.com/p/smali/

[9] (2013, Dec. 3). *Android-Apktool: A Tool for Reengineering Android APK Files* [Online]. Available: http://code.google.com/p/android-apktool/