Journal of Software Engineering 8 (1): 50-57, 2014
ISSN 1819-4311 / DOI: 10.3923/jse.2014.50.57
© 2014 Academic Journals Inc.

An Improved Attack Tree Algorithm Based on Android

Xia Jiang and Weiwei Qi
Laiwu Vocational and Technical College, Laiwu, 271100, China

Corresponding Author: Xia Jiang, Laiwa Vocational and Technical College, Laiwu, 271100, China

ABSTRACT
Android operating system is the most popular mobile system in the world. This study makes a detailed analysis of its security mechanism and existing problems. By analyzing the security architecture of Android operating system, exploited loopholes are pointed out. It also points out hidden security dangers which can be used by hackers. This study introduces the malicious behavior detection method and emphatically introduces the detection method based on behaviors. Due to the disadvantages of the detection method based on behaviors, the detection method is concluded which reduces the rate of false positives. By putting forward an improved attack tree algorithm, a more rigorous theory system has been formed.

Key words: Android, security mechanism, malicious softwares, detection models

INTRODUCTION
The Android operating system (Dempsey, 2011) is currently the most popular mobile operating system (particularly in the intelligent mobile phone area) and open source and free of charge are the biggest characteristics. With the rapid development of Internet, intelligent mobile phone can realize network data browsing (Kucuktunc and Ferhatosmanoglu, 2013), online payment (Cesare *et al.*, 2013) and other functions. Android platform is very convenient to intelligent mobile phone users but the open source makes intelligent mobile phones become hacker attacks. Without the approval of users, malwares (Song *et al.*, 2013) can install in the background and automatically run on handheld terminals. Data flow resources can make high fees or consumption of users. Softwares with those features are known as malwares. The Android platform is open with a lot of users and a large number of network APPs (Serrano *et al.*, 2013). Not all APPs are legitimate and some illegal softwares in the mobile phone terminals can steal user privacies to obtain illegal benefits. Malicious behaviors include spam messages (Xia *et al.*, 2013), the third suction garbage software etc.

Aiming at solving the security problems of Android platform, many security companies have developed all kinds of security softwares. But the security softwares can only maintain some security problems of mobile phone on the surface, such as block spam messages, refusing telephone harassment etc. Security softwares have some limitations, firstly, their various functions are dependent on mobile phone's hardwares. If you want to make the operation smoothly, better hardwares are needed. The operations of security softwares can reduce the speed of mobile phone and even lead to the system crash. Therefore the current security softwares have caused certain influences on the performances of mobile devices. Secondly, the fragmentation of Android platform is very serious and the security softwares in general are not universal. Thirdly, users do not have too much attention on malicious softwares actions, resulting in the rapid spread of malware.

The operation principles and vulnerabilities of the Android system are presented and based on this, a kind of improved attack tree algorithm is put forward in this study.

SECURITY MECHANISM OF THE ANDROID PLATFORM

Android is an open source operating system based on Linux system and Google has announced opening its source codes and it is mainly used in mobile terminal equipments. The Android system consists of Linux operating system and drivers (Linux kernel), the framework and the Java runtime environment android framework and Android applications which are shown in Fig. 1.

In Fig. 1, the Linux operating system and the drivers are realized by the C language and the Android core system services not only rely on the Linux2.6 kernel but also increase the driver kernel. Local framework and Java runtime environment are implemented by C/C++. They use different components in the Android system and the Android application framework are effective for developers. In the Android system, developers can access the core application of the API frameworks which is showed in Fig. 1. Android applications are written by using the Java programming language. Users developing Android applications and the core of Android application are at the same level and they are constructed by the Android system APIs.

The Android system is developed by the Linux kernel. It keeps and inherits the security mechanism of the Linux operating system. All levels of the system architecture have unique characteristic performances in security and each level is as follows.

Kernel level security mechanism of the linux: The Android system of the Linux kernel includes mandatory and discretionary access control mechanism. Mandatory access control mechanism is realized by the Linux security module. Discretionary access control mechanism is realized by the file access control. The control of the Linux file system is composed of users, group, other, Read (R), Write (W) and executive (x). Each file has three basic sets of permissions and the combination can limit, allow, deny users, groups of users and other users access. In general, only

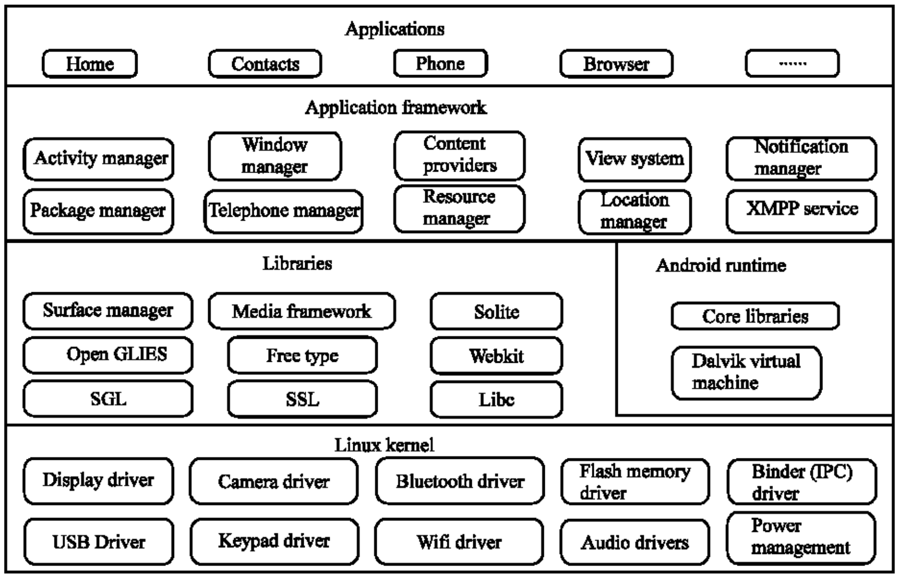

Fig. 1: Architecture of android

users whose uid are "system" or "root" have the Android file access permissions. the application can be achieved on the corresponding file access with the application of Android permissions. Therefore, the Android system uses discretionary access control mechanism of the Linux and the operation of the Dalvik virtual machine can achieve the "sandbox" mechanism (Karlsson and Agerfalk, 2012; Frydenberg, 2013).

Android "sandbox" mechanism: "Sandbox" is used to achieve the mutual isolation between different applications and processes. Each APP and system processes are assigned the unique and fixed User Id and this Id corresponds to the kernel uid. Each APP runs independently in the Dalvik virtual machine with the independent address space and resources. The processes running on the Dalvik virtual machine must rely on the core layer Linux process. Android use the Dalvik virtual machine and the Linux file access control mechanism for realizing the sandbox. Any application accessing system resources or other application resources must be permitted in its own manifest file or sharing the uid.

Access check mechanism of android: Android is a system with "separation of powers" and each application using the limited Android resources (network, telephone, SMS, Bluetooth, mail list, SdCard etc.) must be in the form of XML files in advance to apply for Android system. It can use the corresponding resource after Android system approves. Authority and APIs of Java are many to many mapping.

Digital signature mechanism of android: All applications inmounting to the Android system must have a digital certificate and the certificate is used to build the trust relationship between developers and application identifiers. If the protection level of a permissions is marked as signature or system, the Android system will grant this permission to have the same signature applications or Android packages. Applications of the Android system will not be installed without the digital certificate. If the protection level of a permissions is marked as signature, only the digital signature of an application and its statement have the same digital signature, then they can be authorized by the Android system.

The so-called Android platform malwares refer to those without users' consent to the installation or are secretly installed on the user terminal software by some secret ways. Malicious software will consume users' data traffic and steal users' privacy information. They can reap benefits by these means. Malicious deductions, customized package, download and steal users' privacy are the common Malicious attacks.

This section analyzes the Android system architecture in each layer and a detailed analysis of the security mechanism, the definition of malicious software and some common malicious attacks are presented. The necessities of testing malicious softwares are basis of the following chapter.

AN IMPROVED ATTACK TREE ALGORITHM

The security mechanism of the Android system is very strict but malicious softwares can also find the vulnerabilities and attack the system. Users' weak security consciousness provide the infested soil to malwares. Root behaviors and wanton brush behaviors will damage the security mechanism of Android system, so the detection of malicious behavior is very necessary.

This section analyzes many kinds of detection technology about malicious softwares based on Android platform. By analyzing the comparison of various detection technologies, this section puts forward an improved attack tree algorithm.

Analysis and detection technology of malwares: There are two kinds of detection methods generally adopted in mobile phone terminal about current operation for secure software malwares. One can detect the signature and another can detect the software behavior. By the comparative analysis, this study adopts the latter one.

Detection technologies based on signatures: This is a relatively traditional detection technology. The security mechanism of the Android system and the current popular security software detection mechanism can detect malicious softwares by the signature technologies. Check malware signatures in mobile phone system and determine natures of malicious softwares by comparing the signature. In order to ensure the accuracy of malicious softwares judgment, the signatures in the database have to update for ensuring the effectiveness. Updating the virus database can ensure the accuracy of detection in a certain extent but the increasing malicious make the virus database too large. The detection becomes too difficulty and the energy consumption of the system will increase.

This detection technology can not give the judgement before malwares attack the system. Too many malicious softwares make the operating system threatened by the growing crises. Security software can be updated after malicious softwares attack the system. Measures should be taken to nip malicious acts in the bud. Once the system is infected with malwares, there is a period of time to have a chance to detect malicious software. In a limited period of time, even if the system has been infected with malicious softwares, users can not detect the malicious softwares. Signature detection technology can not prevent unknown malwares and new variants, so this study choses the behavior based detection technology.

Detection technologies based on behaviors: The core technology is to detect malicious softwares by their behavior features. It can identify whether a software is malicious by analyzing its behaviors. This technology has many advantages. First of all, it can detect unknown and known malicious programs; secondly, the detection efficiency is high and it do not have to access the large malicious code database; thirdly, it is composed by the static and dynamic methods and there are advantages and disadvantages.

Static analysis can decompile unknown softwares and analyze files and executable codes. Judge whether the software behaviors exists in the malicious database by artificial. The disadvantage of this method is that program cannot be modified. Dynamic analysis refer to observe the execution of malicious software and it can quickly create files, registry entries, contact the site etc. Dynamic detection does not have advantages of the static detection. Compared with the static detection, the use of time is longer and the code operation is more complicated.

Dynamic detection has two methods. One method is the comparative method but its accuracy is too low. This method only cares behaviors' execution and can not track the effect on the progress. So, the accuracy of this method is low. Another method is the tracking method. This method can obtain the dynamic characteristics of the program and analyze operations. According to the different implementation techniques, tracking method is not same.

Due to harsh conditions, it requires to give the forecast before malicious programs occur. False positives and negatives are its disadvantages. Although the technology improves a lot, it still could not avoid false positives and negatives. To avoid these problems, malware behaviors must be accurately analyzed. Detection technologies based on behaviors are not fully mature. Many aspects still need further improvement and this detection technology has broad prospects for development.

An improved attack tree detection algorithm: The algorithm in this study can make a mobile code execute on the host. The implementation process can generate some temporary files whose relations are regard as a tree, so this algorithm is called the attack tree algorithm.

Concept of the attack tree: Schneier first proposed the attack tree in 1999, aiming to describe the different types of attack systems. It is a formal msethod to provide the system security. Malicious attack behaviors on mobile phones are described as a tree. Its leaves are described as modes of attacks and paths and the root nodes are described as targets of malicious attacks. It can vividly describe the logical relationship between nodes and root nodes. Compared with the attack graphs, attack trees have better scalabilities. However, the attack tree model has disadvantages. Relying on attack tree models can not clearly reflect the influence of each attack on root nodes, the description is not accurate. The weight is introduced into the attack tree model in this study and can be more accurately expressed the influence on attack targets.

Improvements of attack trees: This study redefines the attack tree model and increases the weight description of the attack tree.

Definition 1: The improved attack tree model is just like the attack tree and they are trees with one or more nodes. The model is marked as L = (W, V, Indicator-W, Indicator-V, U):

- W(L) Represents a non-empty set and is an execution body in a real model. The root node is the executive body with replication, initial execution ability
- V is a subset. Supposed there is a directed edge from W_1 to W_2. W1 is called the parent node of W2 and W_2 is the child node of W_1. That is to say that the executive body W_1 creates or calls the executable W_2
- Indicator-W is an execution body who provides the service in the attack tree model. The function IsSys is regarded as the basis judgment of the execution body. When the node W represents a non system executive body, mark IsSys (W) = False, if not, IsSys (W) = True
- Indicator-V is a set composed by the results returned by Boolean functions and associated with the attack tree L. (Wm, Wn) can represent behaviors. For malicious behaviors, mark IsNormal (Wm, Wn) = False, if not, mark IsNormal (Wm, Wn) = True
- U is a set of weights and used to determine effects of the attacks on the root nodes in the model

By analyzing a large number of malicious code samples, an instance of the attack tree model is shown in Fig. 2.

In Fig. 2, the node A represents the executive behavior of a malicious sample. B represents a executive body created during the runtime. When the program runs, B will traverse all executive bodies of the entire system and generate E, F and G nodes. C is generated as a temporary file and D is another executive body generated by the A. By analyzing Fig. 2, A did not produce malicious behaviors in the execution and it only creates a temporary file and two executive bodies. B has malicious behaviors. In order to judge malicious behaviors accurately, a comprehensive analysis of child nodes should be analyzed as the following algorithm:

- **Step1:** Calculate weight of each node
- **Step2:** Analyze the improved model and all nodes and judge whether A is malicious

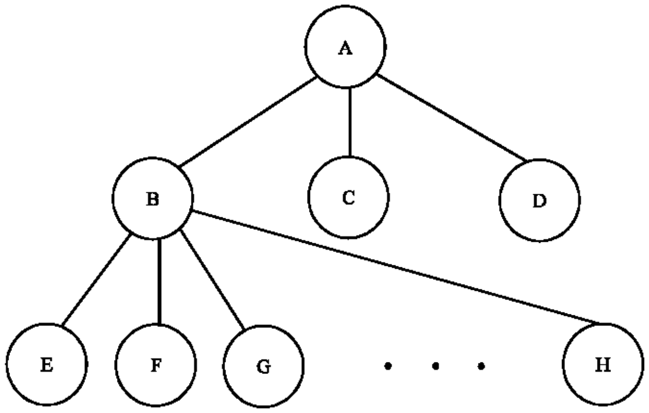

Fig. 2: Attack tree model

Malicious codes start from an executive body and can distort files. The implementation process and the attack tree model are very similar. The attack tree model can be used in the analysis of malicious behaviors. It should be adjusted as follows:

- The Root node represents the attacked target file and it also represents the initial malicious codes of executive bodies. Leaf nodes represent temporary executive bodies
- Directed edges represent relations of executive bodies
- Node weights represent malicious weights

An malicious code detection algorithm based on improved attack trees: In the malicious code detection algorithm based on improved attack trees, the relationship between each node is the basis. Under this model, O is regarded as an operation behavior of W. W is the subject and Z is the object.

This study divides malicious behaviors into three modes and make a definition as follows: $K = (k_a, k_b, k_c)$ represents effect made by malwares on the system and $k_a(u)$, $k_b(u)$ and $k_c(u)$ denote automatically activated, self propagation and breaking capacities of the main body W, respectively. If K is zero, it indicates no malicious behavior; if K becomes bigger, malicious behaviors become more serious. α, β and γ represent the weight of activation ability, communication capacities and failure abilities, respectively and weights can be adjusted.

Definition 2: Let W be a subject and there are n objects, then the influence index can be calculated by:

$$K(z) = \alpha k_a(z)+\beta k_b(z)+\gamma k_c(z) \tag{1}$$

Malicious weights of the subject are:

$$K(w) = \alpha k_a(u)+\beta k_b(u)+\gamma k_c(u)+\sum_{i=1}^{n} K(zi) \tag{2}$$

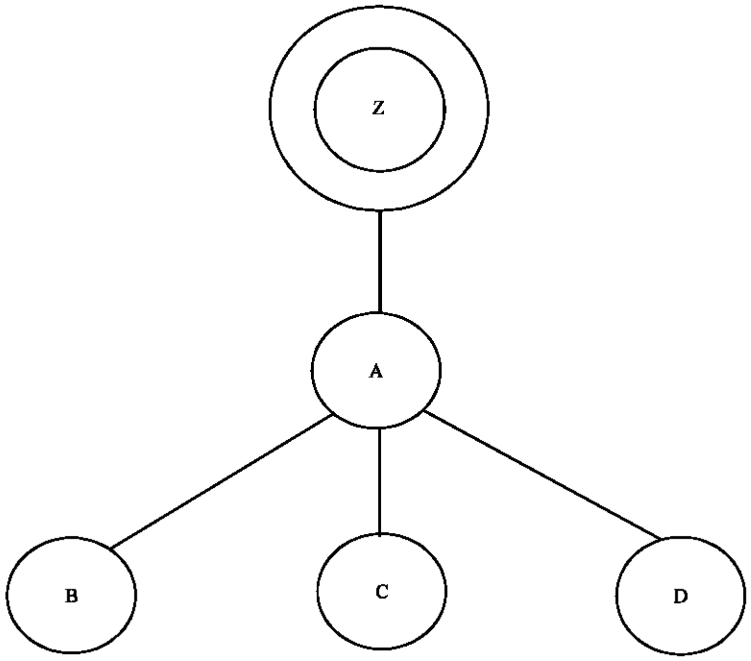

Fig. 3: Attack tree structure containing executive bodies

Malicious comprehensive weight of W is $U^1(W)$ and can be calculated by:

$$U^1(w) = K(w)+\sum_{i=1}^{n} K(q[i]) \tag{3}$$

During the spread of malicious codes, there exits a certain relationship shown as below.

From Fig. 3, the root node represents the executive body. A can execute the transmission by copying, downloading. The malicious weights by downloading does not accumulate to Z. While calculating weights, some safe softwares and malicious software have similar behavior characteristics. It is possible that they have the characteristics set which makes too many errors. According to the different influence, weights should be changed as follow:

- V represents a node with n sub nodes (where the array q_i $(1 = I = n)$ represents n sub nodes). If q_i can satisfy $(w, q_i) \in E(L)$ and IsNormal (v, q_i) = True, have $w(v) = -wt\ (v)$
- V represents a node and satisfies IsSys (w) = True, have $w\ (v) = 0$
- Else, have $w\ (v) = wt\ (v)$

This section introduces the detection technology based on behaviors and analyzes its advantages and disadvantages. An improved attack tree detection algorithm is proposed. Compared with the traditional algorithm, the improved tree algorithm involves edges, nodes and relationships between nodes and weights. It improves the accuracy of malware detection and the false rate is greatly reduced. It also improves the detection technology based on behaviors.

CONCLUSION

With the rapid development of mobile communication networks, more and more people tend to use intelligent mobile terminals. Android system has the high market rate in operating systems. It has a lot of users and we should increase the attention to research its security mechanism. This study analyzes its security mechanism and points out its possible vulnerabilities. The original malware detection algorithm has been improved. The improved attack tree algorithm are more accurate on behavior detection. By calculating weights, it can be able to quickly determine the nature of softwares. There are also many disadvantages in the security mechanism of Android system. A more reasonable safety strategy, reducing false alarm rate and enhancing the ability to identify the malicious software will be the next step.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant 61202016.

REFERENCES

Cesare, S., Y. Xiang and W. Zhou, 2013. Malwise: An effective and efficient classification system for packed and polymorphic malware. IEEE Trans. Comput., 62: 1193-1206.

Dempsey, P., 2011. Carry on regardless [android operating system]. Eng. Technol., 6: 56-57.

Frydenberg, M., 2013. Creating a collaborative learning community in the CIS sandbox. Interact. Technol. Smart Educ., 10: 49-62.

Karlsson, F. and P.J. Agerfalk, 2012. MC Sandbox: Devising a tool for method-user-centered method configuration. Inform. Software Technol., 54: 501-516.

Kucuktunc, O. and H. Ferhatosmanoglu, 2013. λ-Diverse nearest neighbors browsing for multidimensional data. IEEE Trans. Knowl. Data Eng., 25: 481-493.

Serrano, N., J. Hernantes and G. Gallardo, 2013. Mobile web apps. IEEE Software, 30: 22-27.

Song, P., D. Huang, Q. Yang and Y. Zhang, 2013. Research on financial coordinated supervision platform and supervision strategy for online payment under paperless trade. Int. J. Serv. Technol. Manage., 19: 219-239.

Xia, H., Y. Fu, J. Zhou and Q. Xia, 2013. Intelligent spam filtering for massive short message stream. COMPEL: Int. J. Comput. Math. Electr. Electr. Eng., 32: 586-596.