Study on Reinforcement Technology of Application in Android Terminal

Chao Zhou, Haitao Jiang, Jing Guo, Wei Huang & Jinming Chen
State Grid Jiangsu Electric Power Research Institute, Nanjing 210000, China

ABSTRACT: Android system is the most widely used smart phone operating system at present. As the Android system is a Linux based open source operating system, so anyone can operate it on the smart terminal, which brings serious security problems. A security reinforcement scheme for mobile applications based on the Android platform is proposed. Demonstration and implementation of this program is conducted. Reinforcement algorithm proposed mainly uses the encryption, dynamic loading, code confusion, JNI programming, integrity verification and database encryption technology, which from various angles, protects the interests of application developers and users.
KEYWORD: android; mobile terminal security; application reinforcement

1 INTRODUCTION

Currently, Android system is facing a very serious threat of malicious software (Hu Y B et al, 2014). This part of the malicious application is the attacker with a bad purpose and specialized production, trick users to be taken after the malicious operations. And part is normally used by attackers secondary packaging, namely in the application of the normal insertion of malicious code, then re-compile and package, upload to the app store, by the original normal application directory to the user to cheat to entice users to the fooled.

In addition, except for the two package, the application may also be anti-compiler. The attacker can steal the core code of the application by anti compiler, and obtain the core technology of the software author (Li Y P et al, 2015). Then it makes the author's intellectual property rights be infringed upon by selling the core technology or the copy of the application.

Besides, for some paid applications, the attacker, after the anti compiler application, can also break the framework of the payment. This enables the users to bypass the paid links and direct access to what it wants. As a result, the author suffer economic losses.

Moreover, some applications will generate users' privacy data, such as chatting applications, e-mail applications and so on. The main purpose is to steal the users' privacy data (Feng X et al, 2015). Although Android security mechanism can be very good to prevent malicious programs to hack into other applications, once the malicious application cheat root privileges, then the data is completely exposed in front of malicious applications (Liu W et al, 2014). So we should find out ways to enhance the security of the software to ensure the rights of the software author and the security of the users.

2 STRENGTHENING SCHEME DESIGN

In order to deal with these threats, this paper puts forward the application of reinforcement. The flow chart is shown in Figure 1.

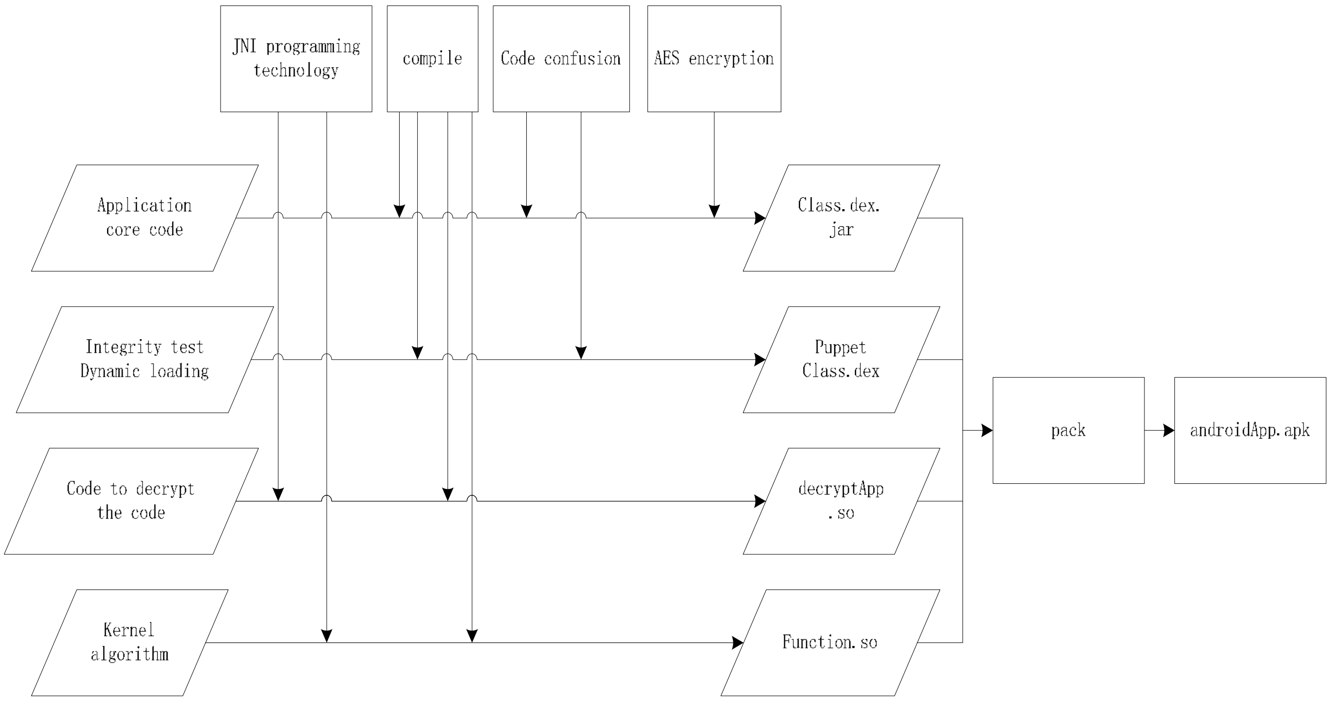

Fig. 1 The overall flow chart of strengthening scheme

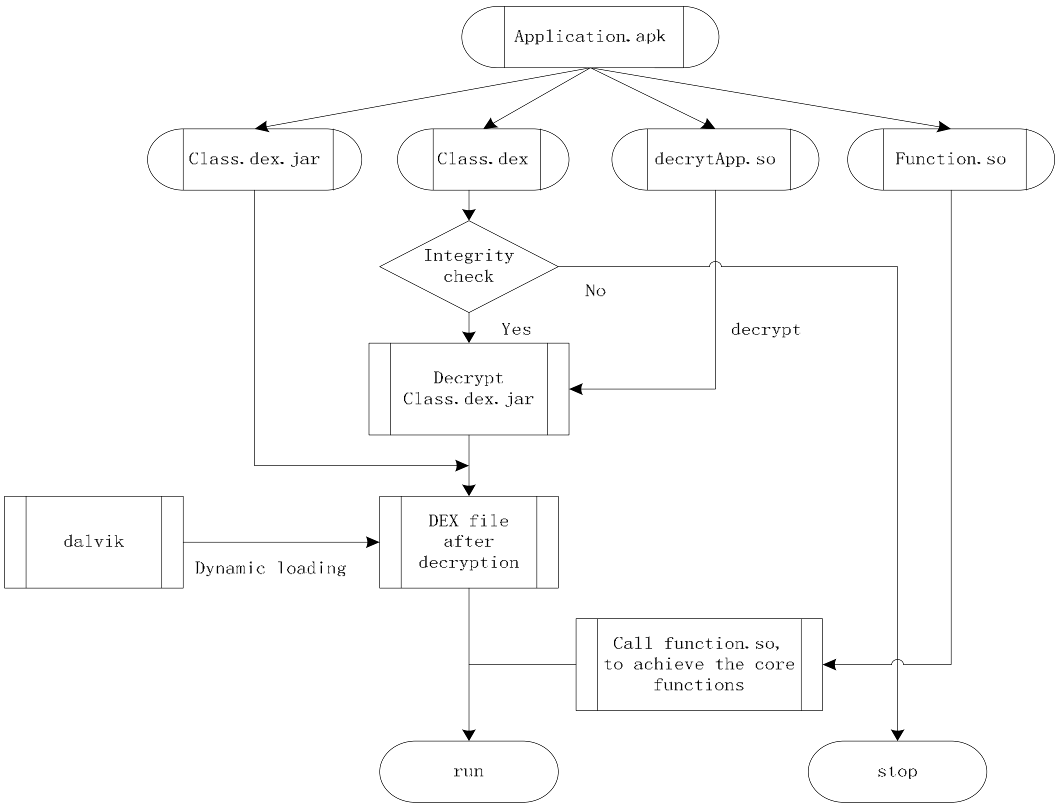

Fig. 2 Running load flow

The reinforcement scheme proposed in this paper is mainly divided into the following aspects:

a. The procedure is divided into four modules, namely: class.dex, class.dex.jar, function.so and decrytApp.so. Class.dex is puppet DEX file, and it is mainly responsible for the integrity of the program verification, class.dex.jar decryption and dynamic loading. Class.dex.jar is a real DEX file, which is used to achieve the main functions of the program (Nie J H et al, 2011). Function.so is a dynamic link library, which provides the underlying core logic functions for the program. DecrytApp.so is used for class.dex.jar decryption.

b. Code confusion for java files, resistance to reverse engineering.

c. Encryption and dynamic load class.dex.jar file. When the program is running, load the puppet class.dex file, decrypt the class.dex.jar file, and then use the dynamic load method to load the DEX file.

d. Use JNI to prepare the decryption library decrytApp.so and the underlying core logic function.so.

e. Join the integrity check-in after partial class.dex file function. When the program is running, first run the integrity check code to check whether the program has been tampered with or be packaged two times. If you find the wrong check results, it will be directly out of running.

Figure 2 describes the process of loading the Android application through the consolidation process.

From Figure 2 we can see, load class.dex at first, run the integrity of the verification code. If the program is found to be tampered, then the program running stops. If the correct call decrytApp.so to decrypt the class.dex.jar, the DEX file is decryption. Using dynamic loading mechanism to load a DEX file in the Dalvik virtual machine, to achieve the main function of the program, and call function.so function library to realize the program's underlying core functions, thus completing the program operation.

3 FEASIBILITY ANALYSIS OF STRENGTHENING SCHEME

Application of reinforcement scheme requires developers to obey certain rules of application development. It will cause some changes to the original development of the developers, of which the main influence is JNI programming technology and dynamic loading technology (Song J et al, 2016). In the application of the reinforcement scheme, the application of the core algorithm and complex logic using C/C++ language, and then use the NDK Android development tools compiled into a dynamic link library, and finally the upper level code calls through the JNI technology. Because the C/C++ program is compiled to generate a binary file and it is very difficult to reverse, so in order to get a higher security we can make the core algorithm written in C/C++ language (Enck W et al, 2011). Theoretically speaking, for Google company's official launch of the JNI+NDK development technology, the feasibility of the reinforcement scheme in this regard can be guaranteed.

The dynamic loading technology is used in the encryption technology. Because there are some limitations of JNI+NDK Technology, the vast majority of an application is developed by using the Java language. Only the core algorithm and time consuming logic will be used in JNI+NDK technology development (Yuan F et al, 2013). In order to prevent this part of the core code from being reverse, core strengthening program is coded into class.dex.jar encrypted file, and the runtime is first decrypted. And then use dynamic loading technique in the loading code. Dynamic loading process is used in Java's reflection, and it is not confused with the core code.

In order to verify the feasibility of strengthening scheme, the paper tests two small Android application. After running several tests, it is found that Android applications use the normal execution after reinforcement program.

4 EFFECT OF REINFORCEMENT SCHEME ON APPLICATION PERFORMANCE

As the application of the reinforcement scheme uses a number of additional safety techniques, it is necessary to test the performance of the reinforcement scheme in the application of operational efficiency. The test is mainly carried out from three aspects, namely: the influence of the strengthening scheme on the application starting time, the influence of the strengthening scheme on the application execution time, and the effect of the reinforcement scheme on the application file size.

4.1 *The influence of the strengthening scheme on the application starting time*

The starting time of the application before and after reinforcement is shown in Figure 3.

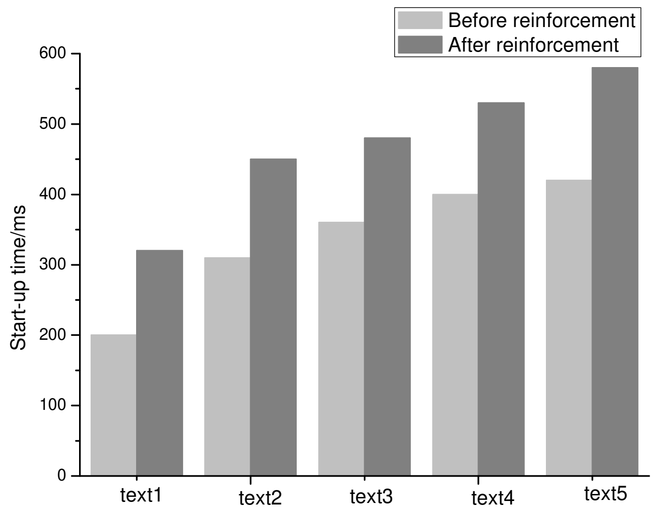

Fig. 3 Comparison of starting time before and after reinforcement

The application of the reinforcement scheme will be carried out at the start of the integrity check and decrypt the class.dex.jar file, so the application will reduce the efficiency of the start.

4.2 *The influence of the strengthening scheme on the application execution time*

The application execution time before and after reinforcement is shown in Figure 4.

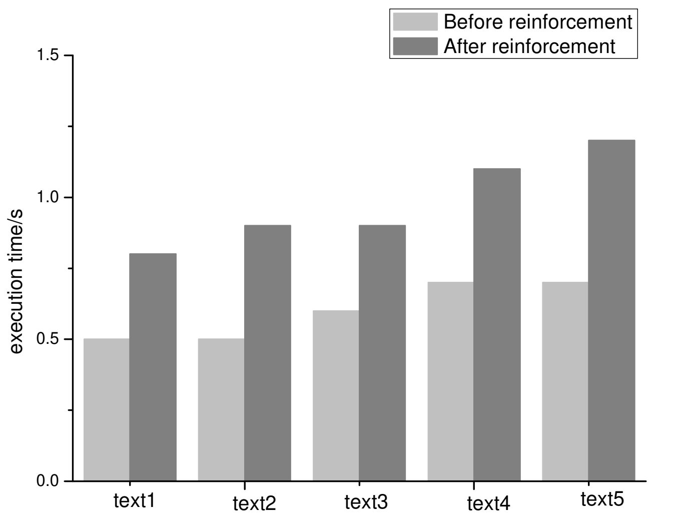

Fig. 4 Comparison of application execution time before and after reinforcement

The implementation of specific features of the reinforcement program requires the JNI to call the underlying dynamic link library, and therefore will increase the cost of this part of the time. Figure 4 shows that the execution time of the application will increase significantly after reinforcement.

4.3 *The effect of the reinforcement scheme on the application file size*

The file size of the application before and after reinforcement is shown in Figure 5.

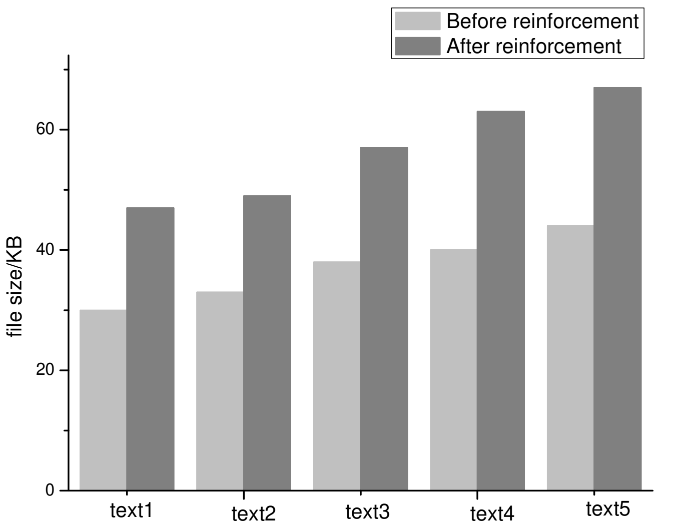

Fig. 5 Comparison of file size before and after reinforcement

The reinforcement scheme will generate the dynamic link library file and the encrypted class.dex.jar file, so the volume will become large compared with the original application. Figure 5 shows that, after reinforcement, volume is increased by an average of 50% or so, but it is just a simple test application, a relatively small volume. So in fact, the normal application volume increased by is not very obvious.

5 CONCLUSION

From the test results, the use of the reinforcement scheme and the application of the volume, starting time and the execution time are increased, which shows that the reinforcement scheme will have a certain impact on the efficiency of the application. However, in the test, a simple Android program is taken as an example, there is no too much function, and therefore it cannot be a complete assessment of the impact on the operational efficiency of the reinforcement scheme applied in the actual environment. In addition, taking the actual environment into account, application volume itself will be relatively large, and the function is more complex. Its starting time and the execution time will be longer than that in the test data. Thus, it is necessary to strengthen the proposal to create the effect of actual data. Therefore, from the comprehensive point of view, the effect of the reinforcement scheme is still in the acceptable range.

REFERENCES

Enck W, Octeau D, Mcdaniel P, et al. A study of android application security[C]// Usenix Conference on Security. USENIX Association, 2011:1175-1175.

Feng X, Lin J, Jia S. The Research on Security Reinforcement of Android Applications[C]// 2015 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering. Atlantis Press, 2015.

Hu Y B, Wang C X, Yuan Jie. Design and Realization of a Mobile Application System for Electric Distribution Network Rush Repair[J]. Jiangsu Electrical Engineering, 2014, 33(3):49-51.

Liu W, Wang S, Zhou Y, et al. An android intelligent mobile terminal application: Field data survey system for forest fires[J]. Natural Hazards, 2014, 73(3):1483-1497.

Li Y P, Ji C Y Fan G X. Designing of Mobile Marketing System Based on the Internet of Things Technique[J]. Jiangsu Electrical Engineering, 2015, 34(5):80-84.

Nie J H, Zhou X H, Yan-Wei Y U, et al. Android Security Reinforcement Technology[J]. Computer Systems & Applications, 2011.

Song J, Zhang M, Han C, et al. Towards fast repackaging and dynamic authority management on Android[J]. Wuhan University Journal of Natural Sciences, 2016, 21(1):1-9.

Yuan F, Xiao L, Ltd D. Modeling of Android Application Security Testing[J]. Netinfo Security, 2013.