The Research on Security Reinforcement of Android Applications

Feng Xiaorong[1, a], Lin Jun[2,b] and Jia Shizhun[3,c]

[1] Software Quality Testing Engineering Research Center, China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou, Guangdong

[2] Software Quality Testing Engineering Research Center, China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou, Guangdong

[3] Software Quality Testing Engineering Research Center, China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou, Guangdong

[a]fengxr@ceprei.com [b]linjun@ceprei.com, [c]jiasz@ceprei.com

Keywords: Software Protection; security reinforcement; Android platform; packing technology.

Abstract. Along with the rapid progress and widely used of mobile application, security protection to software application has become an important issue. The lack of binary protections makes it important to implement packing services. By analyzing traditional features of software reinforcement technology, the paper describes the basic functions for security reinforcement, illustrates the principle of protection method and put forward corresponding preventive measures and technical solutions. New strategies including elliptic curve encryption and dynamic loading mechanism, agile reinforcement process as well as two-step self-modified obfuscation are proposed to satisfy the effectiveness and efficiency requirements for application reinforcement.

I. Introduction

With rapid development of Android platform, it has gradually become the most popular operating system for intelligent terminals. The number of Android applications is increasing greatly and software developers pay special attention to its copyright protection. It's necessary to ensure about the interests of developers and guarantee the security of application software, so as to prevent the application from decompiling and dynamic debugging by malicious attackers. However, the attack injected with malicious code and reverse engineering would result in security risks and influence reliability of the application. The application security issues haven't got effective solutions, which become an obstacle to terminal users and software developers. Thus application security reinforcement system based on Android platform is essential to ensure the security of the application for the healthy and stable development of the Android system. The user of the system is mainly for Android application developers and users of applications downloaded from the Android platform.

The current security threats on Android platform are getting worse and illegal phenomenon such as illegal copying, reverse engineering and tampering become popular. How to use security reinforcement to protect applications in high efficiency is an important problem for all software developers to solve. Generally speaking, any software could be cracked and modified by means of special methods. After a long period of development, software protection technology has become a relatively independent research field. At present, the technology of the software protection can be generally divided into two categories, including hardware encryption and software encryption technology.

Hardware encryption technology adopts special hardware with encryption function to realize software protection. The hardware devices are added to computer host through various interfaces or socket. Compared with software encryption, the hardware encryption technology can protect software more effective. However, since all sorts of extra hardware cost is high, and the hardware installation could not satisfy the flexible requirement for system maintenance, the encryption method still has many shortcoming and vulnerability problems which restrict its widely used.

Software encryption technology is implemented through some special program added in the software, which can effectively increase the difficulty of source crack. The protection method is relatively flexible with limited increase costs and becomes a general effective measurement.

II. Encryption Shell Protection

In software encryption technology, encryption shell protection is a kind of good method, which is also named software packing. The executable file of the target software is encrypted or compressed with some special code. The control of the operating system will give priority to these special codes when the procedure is executed. Then, the special code would control the execution of the source program, so as to realize the function completely with no difference.

In order to make a better solution for software protection, researchers and code analysts started to develop the skills to unpack target samples for analysis. Software decoding is actually the reverse analysis of program execution. Software crackers usually analysis the concrete frame structure and control flow of the program through binary executable file. The process of reverse analysis is mainly divided into two aspects: static analysis and dynamic analysis. Static reverse analysis refers to analyzing a binary executable file without the software running. While, the dynamic inverse analysis is implemented under the premise of running the software so as to get effective information through analysis of binary code in the program.

Software encryption shell protection is derived from the software decoding technology development. At present, the software packing technique can be classified into three categories: compression, encryption and virtual machine. The three methods mentioned above can be implemented by existing tools.

Software packing with compression and encryption method is implemented through the compression and encryption for original program. Special protection program is set up in the process of decompression and decryption so as to ensure that the packed software is equally authentic with the original program. However, the crackers good at static decompilation and dynamic debugging techniques, may find the physical address in memory through the software packers shelling, and then get the address data. Finally source code is reached without any protection. This kind of crack technology does not need to know specific encryption algorithm. It seems that the protection through source program of the compressed and packer technology completely defective and inefficient, which leads to the protection not safe enough. In order to solve the problem, developers attempt to use a variety of debugging techniques. But these debug technology also has some limitations, for example, the proposed method is only effective for inverse dynamic crack, while is less effective for anti-static crack software. Apart from this, the efficiency of the process is declined significantly using the debugging technique.

The virtual machine technology is developed based on the following technology mentioned above. It is effective to increase the complexity of the software after the packer process, which improves the difficulty of reverse analysis, so as to achieve the goal of protection. In the process of program execution, the virtual machine software packer protection would generate bytecode in computer memory, while the original machine code is hidden. So, it is difficult for software crackers to get logical structure of the machine code and it has become the mainstream technology of software packers. In the virtual machine technology, through packer application software, constructs the corresponding virtual machine this is constructed out of the virtual machine instruction set is different from general structure of the computer instruction set.

The virtual machine is constructed through application software packer method, and the structure of the corresponding instruction set is different from general ones. In virtual machine environment, software is compiled into bytecode with the virtual machine instruction set and will be implemented under the control of the interpreter. In this way, the traditional methods and tools with architecture of computer instruction set would be invalid for the new random virtual machine. Therefore, the encryption shelling technology with the application of virtual machine can greatly increase the complexity of the program. However, this method still has some shortcomings such as the decrease of

procedure efficiency. Therefore, software packer using virtual machine technology is suitable for the program with lower executing efficiency.

III.The Principle and Structure of the Software Reinforcement

On the basis of research for offensive and defensive technology, the software reinforcement scheme on the Android platform needs to have the following functions: software piracy prevention, sensitive information protection, software tampering monitoring and dynamic debugging/decompiled/disassembly prevention, etc.

Through the analysis, software security reinforcement can be divided into two parts, one part is the code confusion for binary code generated on the Java side, while the other part is the dynamic loading for bin file generated after cross compilation on the side of the C code.

A.The principle of the software packing

The principle of the software packing method is as follows:

(1) Compress and encrypt each section block;

(2) Add decompression and decryption programs to the new section block, and add head to the session block table;

(3) Modify the entry address of the original program.

B.The main methods for software packing

The packing method is mainly including the following ways:

(1) Multiple packing technique, which is packing the program files for many times, so as to achieve enhanced protection strength of the software. However, research shows that the protection strength and packing times is not in scale.

(2) Masking packing technique: by changing its characteristic attributes to disguise the program, so as to achieve the effect of interference with crackers' analysis. If the crackers could not distinguish software shell type correctly, the software cracking will be in great difficulty. Meanwhile, it will cost huge resource and time, which is unreasonable.

(3) Hybrid code packing technique: the intermediate code of JAVA and .net Obfuscator program take this way of protection currently.

(4) Transforming code packing technique: using appropriate algorithm to convert the program code. With static disassembling, the transformed code is messy. The realization process of the restoration is finished by shelling program where the transformed code would resume the normal executable program with the same function of original program.

IV. Implement Scheme for Software Reinforcement

A. Agile reinforcement technique based on asymmetric encryption

Based on studying and absorbing the traditional packing technique, agile reinforcement method is innovative attempt in the field of software packing. The principle of agile algorithm used in App reinforcement is realized through the encryption storage for both dex and so file in App, where the program is injected into a shell code. The shell code is executed firstly with the APK application entry point modification, and then, the original program is loaded in the shell. With all the implementation of these steps, the application begins to run.

The contrast of APK package with reinforcement is shown in Figure1:

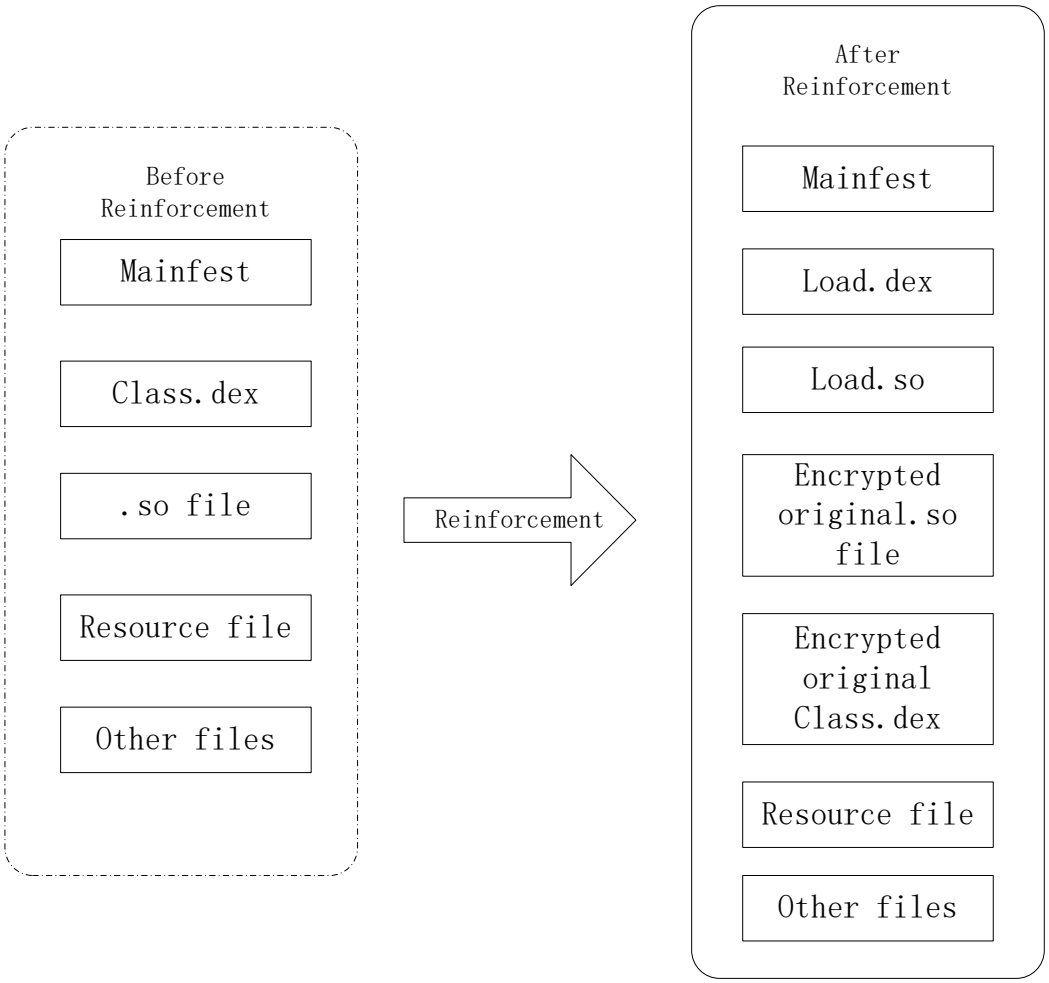

Fig1. The contrast of APK package before and after reinforcement with packing services

The Class.dex and Class.so file is the key protection objection before APK reinforcement. After reinforcement, the procedure generates corresponding encryption Class.dex and Class.so file respectively. Considering the particularity of the intelligent mobile terminal, where its computing power, memory, and network bandwidth are limited in comparison with traditional mobile Internet network, an encryption scheme fit for embedded platform for intelligent mobile terminal is necessary. We proposed elliptic curve encryption public-key encryption algorithm which is simplified for ECC to implement dex file encryption. The ECC algorithm is especially suitable for resource-constrained platform for the intelligent mobile terminal with its high security features, low computational load, small code size and less bandwidth, which is an effective method to improve the security of dex and so files.

By modifying the APK Manifest file, the entry point to the shell code file is Load. dex, thus the Load.dex is loaded by the system and starts running when the application is executed, and then, the original Class.dex and .so file is decrypted in the Load.dex and start running after loaded into memory.

The procedure of proactive class loading and initialization is shown as follows:

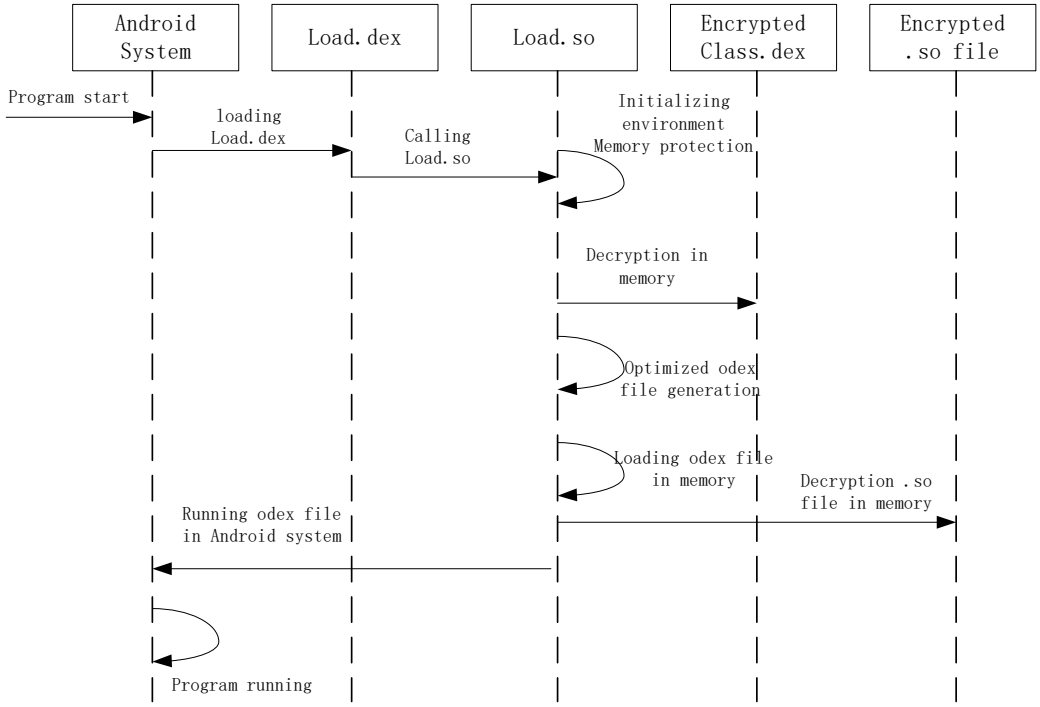

Fig2. The proactive class loading and initialization graphs in packing services

First of all, click on the program after the reinforcement, the Android system based on the user's request begins to load the Load.dex. The dex file calls the Load.so with special functions which includes the memory protection, debugging prevention, injection, etc. Then, decrypt protected Class.dex and .so file in memory. Start the optimization of Android dex dexopt program so as to optimize the original Class. dex file and generate the optimized odex file. Finally, load the corresponding odex file in memory and make its start running.

In the sequence mentioned above, the operations related with original program can only be seen in the memory and it would not generate any content in local files which is a good way to avoid hacking through a simple file copy and getting decrypted file content.

B. Two-step self-modified code obfuscation

Obfuscation aims at preventing analysts from understanding the code. Advanced methods to obfuscate Android Apps have been proposed, such as modifying the names of classes, reordering control flow graphs, encrypting constant strings, etc. Security architecture, code confused complexity and hardware performance should take into consideration in obfuscation strategy. In the packing system, we design an approach of self-modifying code obfuscation based on sensitive program flow. For one thing, it could prevent decompile attack, and is able to run normally. For another, it could reduce the amount of confused performance for mobile phones as much as possible.

The proposed scheme adopt self-modified code model with two-step comparison. It inserts random numbers to destruct the structure of binary code and modifies the confusion code in dex file, in which does not affect the function of the program, so as to resist the disassembly and improve the quality of confusion. In order to reduce the overhead of code confusion and avoid the uncertainty of random selection for confused instructions, we parse the control flow and data flow of the assembly program and find related important instructions in the program for code confusion, so as to improve the robustness of confusion.

The self-modified code model with two-step is shown in Fig 3.

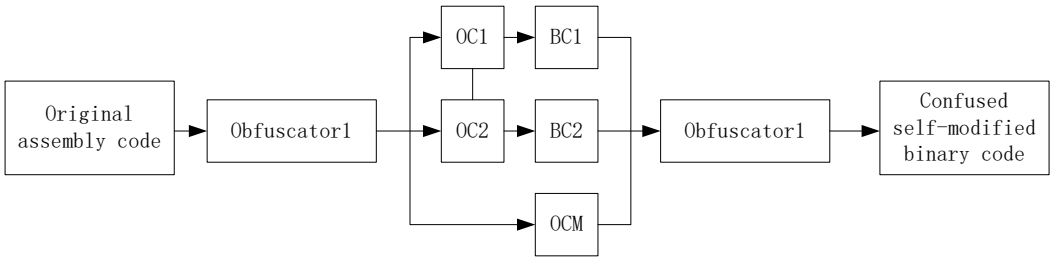

Fig3. The system chart for two-step self-modified obfuscation

The model included two child confusions. Obfuscator1 adopts the program flow sensitivity analysis method which would get confused instructions, two confused code files and a confused map file. Through the control flow and data flow analysis, the system would get corresponding source instruction (SI), repair instructions (RI) and the location of the hidden instructions (HI), and then it will select corresponding RI, HI, temporary confusion (TOI) in according to the source command and finally insert the RI and HI to their position., where there will generate three files: obfuscated code 1 (OC1), obfuscated code 2 (OC2) and obfuscated code mapping (OCM). Since the OC1 and OC2 do not include obfuscated code, RI, HI and TOI code, the OC1 and OC2 files will be distinguished according to TOI1 and TOI2 files. The OCM file is expressed as a triad < TOI1 TOI2, OI > and would be transformed into SI and OI in according to TOI1 and TOI2 converts.

OC1 and OC2 would generate binary assembly code BC1 and BC2, and obfuscator2 will accurate positioning the confused position in binary code by comparing the two confused code file BC1 and BC2. Then it realized dex file confusion through the obfuscated code mapping file so as to further improve the quality of code confusion. The dex file to be reinforced is hidden in the header file, and will be loaded when the program starts running. In the process of program initialization, the obfuscator will find the address and offset for header files and start the program.

V. Conclusion

In this paper, we research the present security risks on the Android platform, analyze the attack methods used in the security threatens, investigate the traditional software protection technology and conduct the systemic investigation on major Android packing techniques. In combination with agile reinforcement based on asymmetric encryption and two-step self-modified code obfuscation technique, a software reinforcement system for Android application is designed.

The different functional modules of security packing system is described in this paper. The ECC encryption and agile algorithm proposed in the system is a good way to ensure about the effectiveness for reinforcement where the program is injected into a shell code through the encryption storage for both dex and so file in App. The two-step self-modified confusion method is designed which the application execution efficiency is not much affected. Based on the improved obfuscators, the security packing method is suitable for the limited efficiency of CPU processor. The research reveals important issues in existing Android application packing services and sheds light on the further study of App security protection.

References

[1] Matignoni, L., Christodorescu., M., Jha, S.: Omniunpack: Fast, generic, and safe unpacking of malware. In: Proc. ACSAC (2007)

[2] Perdisci, R., Lanzi, A., Lee, W.: Classification of packed executables for accurate computer virus detection. Pattern Recognition Letters 29(14) (2008)

[3] Qian, C., Luo, X.., Shao, Y., Chan, A.: On tracking information flows through jni in android applications. In: Proc. IEEE/IFIP DSN(2014)

[4] Qian, C., Luo, X., Yu, L., Gu, G.: Vulhunter: Towards discovering vulnerabilities in android application. IEEE Micro 35(1) (2015)

[5] Rastogi, V., Chen, Y., Jiang, X.:Droidchameleon: evaluating android anti-malware against transformation attacks. In:Proc. ACM ASIACCS(2013)

[6] Roundy, K., Miller, B.:Binary-code obfuscations in prevalent packet tools. ACM Comput. Surv.46(1) (2013)

[7] Shao, Y., Luo, X., Qian, C., Zhu, P., Zhang, L.: Towards a scalable resource-driven approach for detecting repackaged android applications. In: Proc. ACSAC(2014)

[8] Zhang, F., Huang, H., Zhu, S., Wu, D., Liu, P.: Viewdroid: Toward obfuscation resilient mobile application repackaging detection. In: Proc. ACM WiSec(2014)

[9] Zheng, M., Lee, P.P., Lui, J.C.: Adam: an automatic and exetensible platform to stress test android anti-virus systems. In: Proc. DIMVA(2013)

[10] Zhou, W., Zhou, Y., Jiang, X., Ning, P.:Detecting repackaged smatphone applications in third-party android marketplaces. In:Proc. ACM CODASPY(2012)