Preventing Illegal Usage of Mobile Phone Software

Yun-Sam Kim  Eun-Sun Cho
School of Electrical & Computer Engineering, Chungbuk National University
San 12 Gaeshin-dong Heungduk-gu Cheongju Chungbuk, Korea, 361-763
{bijak,eschough}@cbnu.ac.kr

Abstract

Mobile software industry is suffering from the illegal usage of the software. PKI will not be a solution for small software distribution. We introduce a lightweight but safe method to prevent illegal usage of mobile phone software, by instrumentation on binary for device-identity check. The method is now being implemented on a presumable code on demand (COD) server for WIPI (Wireless Internet Platform for Interoperability).

1 Motivation

These days, the market share of mobile software dramatically increases as the mobile phone gets more and more functionalities including multimedia processing, health-check, and authentication. Such tendency will be accelerated more rapidly, as ubiquitous dream comes true and the wireless devices play important roles.

However, there exist some barriers to progress of mobile software industry. One of them is the diversity of hardware platforms which makes software distribution difficult. As a solution, in South Korea, the three major operators, TTA(Telecommunications Technology Association) and ETRI developed a standard for mobile Internet called WIPI (Wireless Internet Platform for Interoperability), which is supposed to be commercially provided in April 2005.

Another problem of mobile software industry is the illegal usage of the software[4]. It is necessary to ensure that software is distributed only to right users, since it is possible for a malicious user to buy software to resell it illegally to other people on lower price. Actually, there already occurs such kind of crime in South Korea. In addition, WIPI will encourage more illegal usage because a code once made on WIPI can run any mobile hardware supporting WIPI.

The previous solutions including PKI protocols can hardly be applied without modification, since the mobile devices have so different properties from traditional computing systems, like low power and lack of computing resource.

In this paper, we introduce a lightweight but safe method to prevent the illegal usage of mobile phone software. To do this, we propose a modified code on demand (COD) server for WIPI which supports binary instrumentation for user-identity-checking at runtime.

2 Overview of The Proposed System

The COD server in WIPI transforms source programs into a platform neutral binary codes. Among the possible implementations of a COD server, we choose the architecture depicted in Fig.1. A COD server takes the original source program (C/C++ or Java byte code programs) as its input, and normalizes it as C/C++ code, which is then compiled into hardware neutral WIPI binary. The result code is packaged and distributed to users together with related multimedia resources.

The right-bottom module in Fig.1 is responsible for rewriting target binary code to protect it against illegal usage. First, the module inserts some routines to check to see if the user is legal, at random positions in the input binary.
Instrumenting binary codes instead of the C source codes jumbles the patterns that the usual compilers remain in the target codes, to complicate reverse-engineering. We could modify the target compiler, but a pluggable module has advantage of interoperability for any COD servers. In addition the proposed rewriting module has an obfuscation component to make the code temper-proof enough against reverse engineering.

3 Instrumentation Scheme

The code inserted to check if the usage is legal may be positioned in the middle of the code or attached at the end of the code. We choose the latter approach due to the structural simplicity. It is similar to a kind of computer virii which jump to the end to run the attached malicious code and then jump back to the original program[5].

In the attached routine, the phone number and the serial number of the device are examined. Currently, we depend on "MC_knlGetSystemProperty()" function to detect the identity of the device. For example, it will check if the phone number is the same as the one of which owner ordered the software. If so, the program will continue running, otherwise the execution will stop immediately. When an attacker runs the software illegally on a host server rather than on a mobile device to redistribute it to other people, the phone number, starting with a special number for non-mobile devices, will cause immediate stop without further checks.

4 Code Protection

In fact, rewriting method is vulnerable in that any illegal user can analyze all the jumping instructions into the end of the code[1]. We devise several methods to prevent it.

First, branch instructions to the attached code will be inserted at random positions in the code body. The jump target also starts at various points of the binary, different from code to code and from user to user, and a various-sized gap padded with garbage is laid between the added code and the main body of the original code. This will confuse attackers to analyze the location that the jump target starts.

Second, all the padding garbage consist of sequence of meaningful instructions, to confuse the reverse-engineers. And such garbage are added in the gap between the original code body and the attached code, and attached at end of the inserted code to harden reengineering about where to jump and to jump back.

Third, we do some obfuscation with opaque predicate insertion[3]. The opaque predicate is a predicate that always returns true or false in a given context. Conditional jumps on opaque predicates will act as unconditional jumps but they look conditional ones. We make use of the never-executed branch of the jump by padding with tricky instructions which will yield a wrong disassembled code.

5 Discussions and Conclusions

Compared with serious PKI or key-distribution protocol, our method is a light weight way for illegal usage checking. We have done this by modifying only the target code, without additional modules or much computing overhead on mobile devices platforms.

Distributors will have the new COD servers augmented with instrumentation and obfuscation modules. Our module can be easily plugged in any COD servers of different architectures.

We cannot prevent all attackers from getting rid of the attached checking routine to get the original code. The purpose of our approach is, however, to make the disassembling process harder. If the analysis is not cost-effective, the attacker would not make efforts on the illegal works.

As known before, there have suggested various ways of obfuscation[2, 4]. We are planning to combine some of them with our method to confuse the attackers more efficiently. We are now scheduled to test our method with various commercial mobile software being distributed by game software makers including GameNeo co. (http://www.gameneo.com) against commonly used disassemblers including IDAPro (http://www.datarescue.com).

References