Evaluation of Tamper-Resistant Software Deviating from Structured Programming Rules

Hideaki Goto, Masahiro Mambo, Hiroki Shizuya, and Yasuyoshi Watanabe
Graduate School of Information Sciences, Tohoku University
Kawauchi Aoba Sendai, 980-8576 Japan

Abstract. Recently the demand to make software resistant to manipulation is increasing. Similarly the demand to hide operation of software or to hide secret used in software is increasing. Software possessing such properties is called tamper-resistant software. One of methods to realize tamper-resistant software is obfuscation of software, and evaluating such software objectively and quantitatively has been an important research subject. One of the known objective and quantitative methods is the method using a parse tree of a compiler proposed in [GMMS00]. This method takes into account the complexity in one module of software but not the complexity originated from relationships among modules. We propose at first several obfuscation methods to create a complicated module structure which violates the structured programming rules. Then, we propose a new evaluation method which can measure the difficulty caused by complicated structure among modules. Its effectiveness is proven through experiments. One of experiments shows the grades obtained by the proposed evaluation well reflects the actual reading time required by analysts.

1 Introduction

Tamper-resistance is a property such that secret object hidden inside is hardly observed or modified from the outside. Software/hardware with such attribute is called tamper-resistant software/hardware. Tamper-resistant hardware intrinsically requires a special physical device so that there are problems of cost and handling. In contrast, tamper-resistant software [Auc96, MTT97, MMO98] is expected to require less production cost. Also, due to no physical limitation, it can be delivered through electronic network. If we can create promising tamper-resistant software, we can replace a certain type of tamper-resistant hardware with its software version.

There is high demand for tamper-resistant software in the electronic commerce systems and agent systems. For example, a bank wants to prevent customers from modifying its software for handling electronic money. Customers succeeding in the modification may be able to cheat merchants as well as the bank. Similarly, mobile agents should not be modified in a remote place. If tamper-resistant software has enough strength against analysis and manipulation, its users have no choice but to obey the process designated by the software.
Obfuscation is one of approaches to generating tamper-resistant software. In this approach the description of software is converted into another one which analysts cannot easily read. Analysts who cannot understand the algorithm of software fail to properly modify the software. We can consider obfuscation in different levels of language, e.g. assembly language and high-level language like C. Software is often distributed in a binary form, but it is sometimes distributed in source code. One can imagine free application software for UNIX and codes written in script languages like Perl and Java Script for such distribution. Meanwhile, even software distributed in a binary form may be transformed into source code by reverse engineering. Therefore, obfuscation of source code has its own importance.

There are several known methods for making software hard to read. For example, several basic operations such as dummy code insertion, code replacement and code shuffling are proposed for the assembly language in [MMO98]. Modification of class files into a complicated form is proposed for Java in [CTL97, KM00]. Modification of the structure of loop into a complicated form and separating source code into modules are proposed for the C language in [MTT97] and in [TOM97], respectively.

In order to produce reliable tamper-resistant software, it is necessary to evaluate the difficulty of reading tamper-resistant software. So far the following evaluation methods are known. In [MTT97] a subject is requested to read tamper-resistant source code of C language and its reading time is counted. Without doubt this method is affected by the skill and subjectivity of each analyst. Thus an alternative objective and quantitative evaluation method should be established. There are several evaluation methods which are regarded to be objective and quantitative. In [MMO98] the distribution of opcodes is observed for evaluating the assembly language. In [GMMS00] the depth and weights of a parse tree created by a compiler is counted for evaluating the high-level language. There is another approach of [AM00] which tries to evaluate the complexity of finding out a secret hidden inside tamper-resistant software. In this method data of a block cipher appearing in memory is observed and time required for identifying a secret key out of the data is counted.

In this paper we seek to objectively and quantitatively evaluate the difficulty of reading tamper-resistant software written by a high-level language. As explained above, there is a proposal of [GMMS00] for such evaluation. However, the method proposed in [GMMS00] solely evaluates the complexity of the internal structure of a module, and does not take into account the complexity originated from relationship among modules. Therefore, we examine i) how to create a complicated structure among modules and ii) how to evaluate the complexity originated from relationship among modules. Regarding the second subject we give experimental results on the validity of our measure in comparison with the actual reading time required by analysts. Such a comparison was not examined in [GMMS00].

This paper is organized as follows. After the introduction, we explain in Sect. 2 notations, definitions and the evaluation method used in [GMMS00], which is