ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

ZERO WATERMARKING USING SECTIONAL OBFUSCATION SCHEME AND STEALTHY CODE OBFUSCATION TECHNIQUE

Ms.R. Saranya [1] Mrs . R.Arthy[2]

PG Student, ANNA UNIVERSITY, Nodal Centre- Kamaraj College Of Engineering & Technology[1]

Assistant Professor, Department of IT, Kamaraj College Of Engineering & Technology[2]

Abstract: Stealing of Watermarks is a fashionable trend in the scientific field ,the most common medium for exchange of information used is the plain text which suffers from tampering attacks. There are very limited techniques available for plain text watermarking and authentication. The traditionally used methods are obfuscation and watermarking. In order to overcome such limitation the concept of code obfuscation and zero watermarking are combined. The use of opaque predicates as one of the building blocks of obfuscating transformation conceals the control flow of the program in the protection of intellectual property. By this the ownership of the software products can be proved, which increases the security level of the software to a greater extend.

Keywords: obfuscation, software security, zero water marking, authentication.

I. INTRODUCTION

A. General Information

Obfuscation, in general, describes a practice that is used to intentionally make something more difficult to understand. In a programming context, it means to make code harder to understand or read, generally for privacy or security purposes. A tool called an *obfuscator* is sometimes used to convert a straight-forward program into one that works the same way but is much harder to understand. Common reverse engineering techniques rely on function and code clarity when copying program code. Obfuscation creates ambiguous code, which makes reverse engineering difficult.

Obfuscation methods are classified depending on the information they target. Some simple transformations target the lexical structure of the program while others target the data structures or the control flow. Obfuscation methods are further classified based on the kind of operation they perform on the targeted information. Some methods manipulate the aggregation of control or data, while others affect the ordering. Some of the code obfuscation methods are layout obfuscation, data obfuscation (Storage obfuscation, Encode obfuscation,), control obfuscation

B. Watermark Attack Methods

Even though in static analysis the inputs are not known, several global analyses succeed in extracting information. Techniques include: constant propagation, range propagation, etc. Also, techniques such as abstract interpretation have been proven useful.

C. Various attack methods

Even though in static analysis the inputs are not known, several global analyses succeed in extracting information. Techniques include: constant propagation, range propagation, etc. Also, techniques such as abstract interpretation have been proven useful.

D. Execution of attack methods

One of the code obfuscation attack method need ten groups of testing programs which are embedded with watermark with those algorithms mentioned above. Then the ten groups of testing programs will be attacked by code obfuscation and each of these attacked programs will be checked whether the watermark embedded into these programs are damaged. Attack method needs the same testing programs. In order to attack these testing programs, a set of randomly selected instructions is embedded into the testing programs.

II. GENERAL METHODS FOR OBFUSCATION

An easy way to comply with the conference paper General code obfuscation techniques aim to confuse the understanding of the way in which a program functions. These can range from simple layout transformations to complicated changes in control and data flow.

The control flow transformations - used for obfuscation can be described affecting the aggregation, ordering or computations of the flow of control.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

Aggregation transformation - breaks up computations that are logically related and merges computations that are not.

Control ordering transformations - randomize the order in which the computations are carried out.

Computation transformations - insert new code or make algorithmic changes to source application.

III. PE FILE FORMAT

MS-DOS Header
PE File Header
.text Section Header
.rdata Section Header
.debug Section Header
.txt Section
.rdata Section
.debug Section

Fig. 1. PE File Format Structure

The Windows NT version 3.1operating system introduces a new executable file format called the Portable Executable (PE) file format. Fig. 1 shows the PE file format structure.

The term "Portable Executable" was chosen because the intent was to have a common file format for all flavours of Windows, on all supported CPUs.

A module in memory represents all the code, data, and resources from an executable file that is needed by a process. Other parts of a PE file may be read, but not mapped in (for instance, relocations). Some parts may not be mapped in at all, for example, when debug information is placed at the end of the file.

IV. OBFUSCATION QUALITY

Potency: Potency defines to what degree the transformed code is more obscure than the original.

Resilience: Resilience defines how well the transformed code can resist automated deobfuscation attacks. It is a combination of the programmer effort to create a deobfuscator and the time and space required by the deobfuscator.

Stealth: Stealth defines how well the obfuscated code blends with the rest of the program.

Cost: Cost is the execution time and space overhead in the obfuscated code compared to the original code.

V. OBFUSCATION SCHEME

Obfuscation is a transformation of program into program, which can be understood as the special case of data coding not for all. This transformation is done without affecting the control flow of the program. Obfuscation method is used to prevent others to understand the program by changing the structural aspect of the program or else by confusing workflow of the program.

Obfuscation of code is also done at the disassembly phase. There are two methods of disassembly: Static disassembly -,Dynamic disassembly

Jump statements in assembly code have number of instructions that jump between statements and the position where it jumps is known as jump height. The difference between two values of jump heights is called jump distance. The range of jump height is called jump range.

Goals of obfuscation

- Improve software security.
- Hard to reverse engineer code.
- Protects the owner's intellectual property.
- Could also be used to hide malicious software.

VI. SECTIONAL OBFUSCATION PROCESS

Sectional Obfuscation Scheme is a method of obfuscation. Before embedding the watermarking text is converted into ASCII and then to Binary format(32-bit) so called as watermarking code table. The main workflow of sectional obfuscation is shows in Fig. 2 as follows:

First step is to find the code section of PE File and Divide the code section into several shares. Then set j = 1 and n equals the number of instructions of code section.

Accessing the separated share. Then acquired the value of x from the code table. Divide the selected share into several basic blocks.

All these divided basic blocks are placed upside down and mandatory jump statements should be added into the inverted assembly code.

The value of x is taken from the code table. If x equals 0, the process jumps to WMArray where the jump distance is recorded as $\pi 1$.

If x equals 1, the process will jumps to WMArray where the jump distance is recorded as $\pi 2$.If j = j + 1 then the process should jump.If j is greater than n, the process will end.

To extract the embedded watermarking the above obfuscated codes should be reversed in the same sequential manner.

So that the secret information can be revealed. The information of watermark text which is not directly embedded into the program is so called as zero watermarking..

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

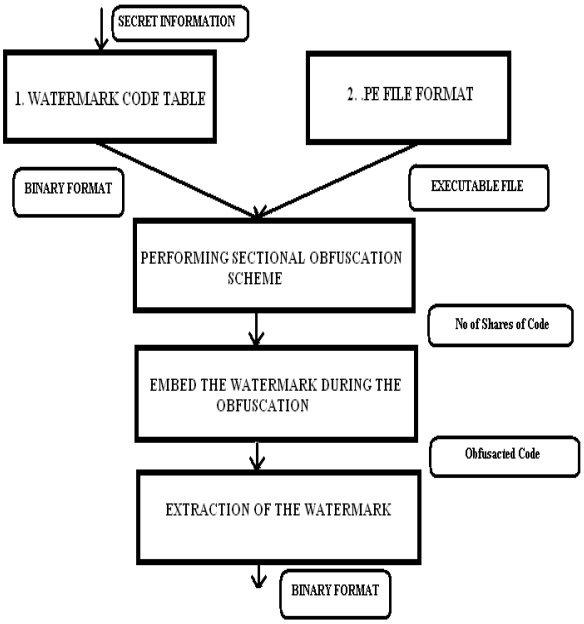


Fig. 2. Sectional Obfuscation Process

VII.PROPOSED SCHEME

The host text document is not altered to embed watermark, rather the characteristics of text are utilized to generate a watermark. As per the author choice the selection of the keyword from the text is done .

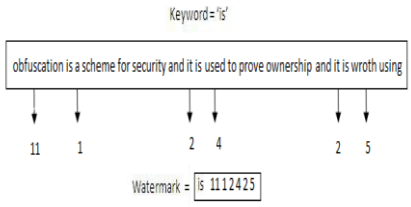
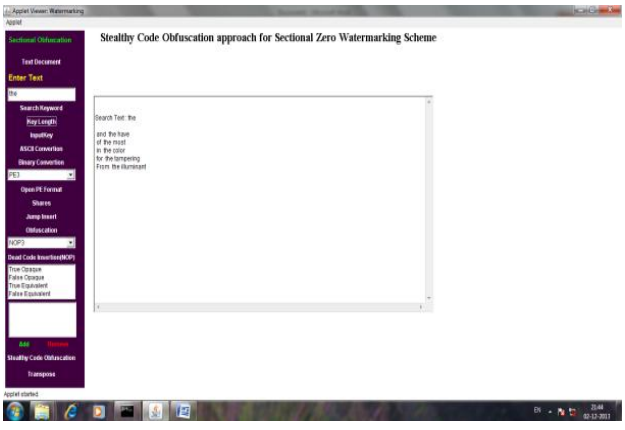


Fig. 3. Generation of Waremark Key

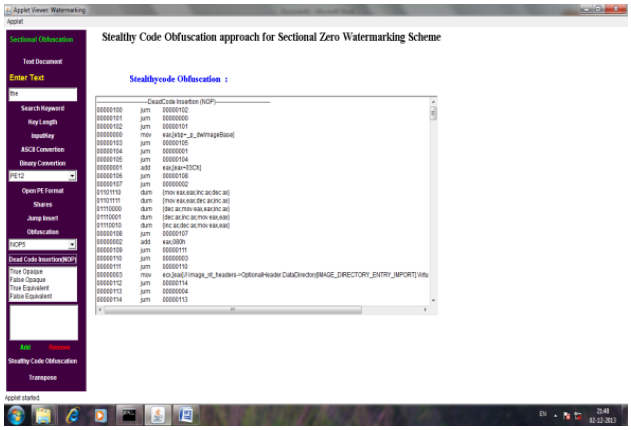
The random selection of the key information to be embedded in the code section depends on the length of preceding and next word length. The watermarking text used in the existing scheme is of 32-bit.In the proposed scheme the use of 40 bit is to make the process of extracting the original text (deobfuscation) to be a more tedious one.



In the figure the selection of preceding and next word along with the length, and the random choosing of watermarking key is done.

Except the extraction process the same sectional obfuscation process is to be followed and further continued with stealthy code process. The obfuscator engine contains a large pool of code transformations which are applied repeatedly to the input file until the required obfuscation potency is achieved or the maximum cost is exceed.

A. Dead code insertion



Inserting dead code or do-nothing instruction does not affect the execution of the original code and creates different looking programs with the same functionality. NOP can be composed of more complex instructions that are never executed.

B. True Opaque predicates

Opaque predicates (i.e. obfuscator that do not belong to the original source code) are the main technique for designing control altering transformations. Being able to create opaque predicates which are difficult for a deobfuscator to crack is a major challenge to the obfuscator. True predicates PT is nothing but the dead code should only be placed in the else block.

C .False Opaque predicates

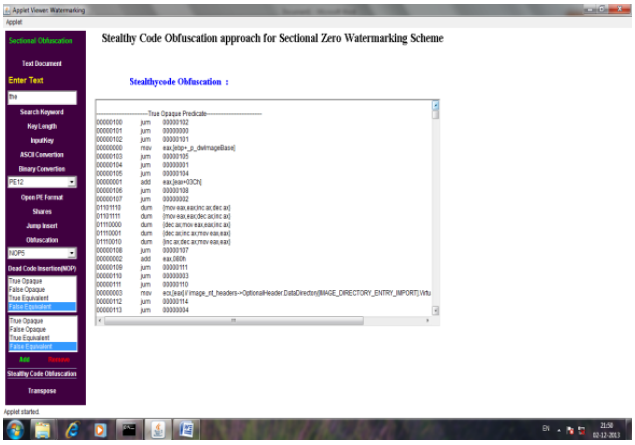
These opaque predicates are injected at randomly selected location in the program. False predicates PF where the execution path runs via false branch.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

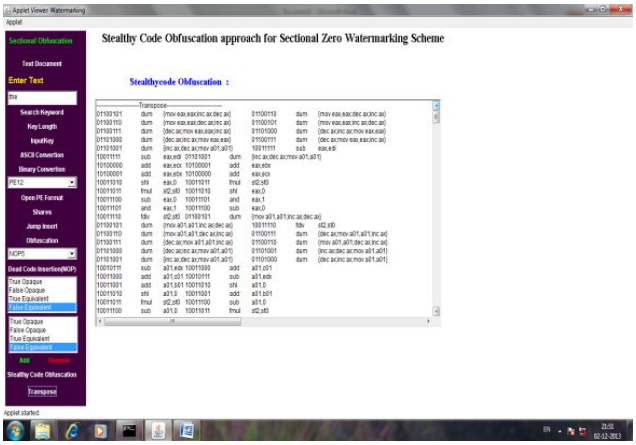
D. Equivalent instruction substitution

Equivalent code substitution is the process in which the given instruction block replaced with another instruction block while keeping the same semantics. True predicates PT is nothing but the dead code should only be placed in the else block by substituting the instructions .False predicates PF where the execution path runs via false branch by substituting the instructions .



E. Transpose Algorithm

Transposition or instruction permutation modifies the order of execution of a program without changing the semantics of the original one. This can be done only if there is no existence of dependency among instructions. The transposes of instructions is done by swapping between two following instructions which is not affect the flow of program. The output of stealthy code obfuscation process is to be stored in a temporary file. Then transposition process is applied to few of the instructions which are randomly chosen from the output file of stealthy code obfuscation process. Then the entire output will be stored in the permanent file.



F. Obfuscation Efficiency

If the disassembler fails to identify correctly the fraction of instructions is defined as efficiency of obfuscation called as *Confusion Factor* (*CF*).Let A be the set of all actual instruction addresses encountered when the program is executed, and let P be the set of all perceived instruction addresses produced by disassembler, then $CF = |A - P| / |A|$. The calculation of CF for basic blocks and functions to determine whether the errors in disassembling instructions are clustered in a small part of the code The outcome result proves that it is too tedious to disassemble programs even only some instructions have been obfuscated.

VIII.CONCLUSION

Watermark is used to prove the copyright of software. But it would lose the function if the watermark is damaged by illegal means. The obfuscation scheme is used to reduce the reuse of the code. The Zero-watermarking can increase the performance of security to greater extents. The sectional obfuscation scheme which is used made the reverse engineering process to be a difficult one.The combination between watermark and obfuscationand the implementation of stealthy code concept is even more secure for protecting the watermark information and make the reverse engineering even more difficult to perform.

Future work will investigates the most advanced opaque predicates techniques in order to find out new ways to enhance the obfuscation process.

REFERENCES

[1] [1] Guangxing Xu, Guangli Xiang, *"AMethod of Software Watermarking,"* 2012 International Conference on Systems and Informatics (ICSAI2012)

[2] [2] Saad M. Darwish, Shawkat K. Guirguis, Mohamed S. Zalat, *"Stealthy Code Obfuscation Technique for Software Security,"* University of Alexandria.

[3] [3] Zunera Jalil, Anwar M. Mirza, Maria Sabir, *"Content based Zero-Watermarking Algorithm for Authentication of Text Documents,"* (IJCSIS) International Journal of Computer Science and Information Security, February 2010.

[4] [4] Gang Chen, "Research on Software Watermark Based on PE File," Hunan University, June 2008.

[5] [5] Hu Chaoju, Wang Xuning, *"Zero Watermark Protocol Based on Time-stamp and Digital Signature,"* 2009 International Forum on Information Technology and Applications.

[6] [6] Jens Palsberg, Sowmya Krishnaswamy, Minseok Kwon, Di Ma, Qiuyun Shao, Yi Zhang, *"Experience with Software Watermarking,"* 2000 Purdue University, West Lafayette.

[7] Mariano Ceccato, Massimiliano Di Penta, Jasvir Nagra, Paolo ,Falcarin,Filippo Ricca,MarcoTorchiano, Paolo Tonella, Fondazione Bruno Kessler, Unita, Politecnico di Torino, *"The Effectiveness of Source Code Obfuscation: an Experimental Assessment,"* ICPC 2009.

[8] [8] Maria Chroni, Stavros D. Nikolopoulos, *"An Embedding Graph-based Model for Software Watermarking,"* University of Ioannina.

[9] [9] Mila Dalla Preda, Roberto Giacobazzi, *"Control Code Obfuscation byAbstract Interpretation,"* Proceedings of the Third IEEE International Conference on Software Engineering and Formal Methods, 2005.

[10] [10] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich,Amit Sahai, Sali Vadhan, Ke Yang, *"On the (im)possibility of obfuscating programs,"* In Proceedings of CRYPTO 2001.

[11] [11]Cong Jin, Xiao-Liang Zhang, Yan Chao, Hong-Feng Xu, *"Behaviour Authentication Technology using Digital Watermark,"* 2008 International Conference on Multimedia and Information Technology.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

BIOGRAPHIES

Saranya.R received her B.E.(CSE) degree from RVS College of Engineering and Technology.Dindugal in 2010 and pursuing M.E.(CSE) degree at ANNA UNIVERSITY, Nodal Center- Kamaraj College Of Engineering & Technology.Her area of interest are Network Security and Software Security.

Arthy.R received her M.E.(CSE) degree from Raja College of Engineering and Technology, Madurai and currently pursuing her Ph.D degree. Her research area is 3D Image Security.