Journal of Computer Science & Computational Mathematics,, Volume 10, Issue 4, December 2020
DOI: 10.20967/jcscm.2020.04.001

Hybrid Obfuscation Technique for Reverse Engineering Problems

Asma'a Mahfoud*, Abu Bakar Md. Sultan, Abdul Azim Abd, Norhayati Mohd Ali, Novia Admodisastro

Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia
*Corresponding author email: selfemoon@email.com

***Abstract*:** Obfuscation is a practice to make certain code difficult and complicated. The main objective of programming code is obfuscated to protect the Intellectual Property (IP) and prevent the attacker from Reverse Engineer (RE) a copyrighted software. Obfuscation contains many methods to encrypt the code such as transforming out potentially revealing data, renaming useful identifiers names to meaningless names, or adding unused or meaningless code and symbols. Obfuscation techniques were not performing effectively recently as the reversing tools are able to break the obfuscated code. We propose in this paper a hybrid obfuscation technique that contains three approaches of renaming. An experimentation was conducted to test the effectiveness of the proposed technique. The experimentation has presented a promising result, where the reversing tools were not able to read the code.

***Keywords*:** obfuscation, anti-reverse engineering, reverse engineering, software security, intellectual property, digital copyright.

1. Introduction

Anti-reverse engineering is a collection of algorithms, techniques and mechanisms that help the developer to harden the code in the source file against prohibited reverse engineering. With anti-reverse engineering, it is possible to use different techniques and algorithms that complicate and harden the code, most of these techniques are used as countermeasure against prohibited reverse engineering [1].

There are different strategies to defend against attacks, and to protect the working of software, such as both legal and technical countermeasures. Copyright and patent are two main approaches to protect software against unlawful copying and stealing of algorithms. Even though copyright protection defends against illegal copying, it does not help in protecting the idea or the implemented algorithms. Software patents help to protect the computer programs inventions including the idea and the algorithm; however, they do not provide solid protection as they are not always enforceable. The major drawback of such patents is the cost [2].

There are usually very expensive to enforce, and therefore unaffordable for small companies. According to the US Digital Millennium Copyright Act (DMCA) and EU Computer Programs Directive legislations, reverse engineering is allowed for the purpose of interoperability between computer programs, if the programs are obtained lawfully. Hence, it is very difficult under these regulations to prevent reverse engineering for the understanding of the inner working of software. Because of these shortcomings, legal protection mechanisms, in most cases, have a small impact on foiling malicious attacks.

For more than a decade code obfuscation is highlighted as the most common anti-reverse engineering and most effective in the software sector. Obfuscation takes many forms; it all depends on the developer to decide which part of the code to protect and what obfuscation method to use. There are several categories of code obfuscation: Layout obfuscation, Data obfuscation, Control obfuscation, and other categories. Figure 1 illustrates the types of obfuscation.

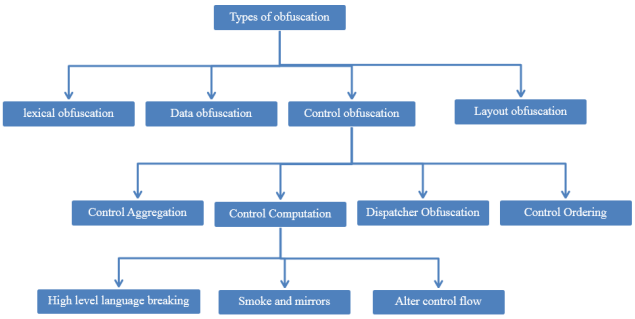

Figure 1. Types of obfuscation techniques

Layout obfuscation is the process to hide the source code meaning when this code is disclosed to a third party. This technique is meant to rename the program identifiers and remove the comments from the source code. Source code obfuscation contains renaming technique. This technique takes several approaches, such as identifiers renaming obfuscation, and string encryption.

Renaming obfuscation technique is known to be effective technique against prohibited reverse engineering. This technique is known to mislead the reverser, it changes the meaning of the code, therefore the reverser will not be able to perform analysis on the code. Identifiers renaming obfuscation is the process to replace the identifiers with meaningless names to unknown language or symbols as new identifiers. Usually, the identifiers have meaning for a better recognition of the source code structures like classes, methods, variables and so forth. Once an identifier is renamed, it is mandatory to provide consistency across the entire application through

replacements of the old names by the new identifiers [3].

Data Obfuscation is a form of data masking where data is purposely scrambled to prevent unauthorized access to sensitive materials. This form of encryption results in confusing data. Data obfuscation techniques are used to prevent the intrusion of private and sensitive online data, such as Electronic Health Records (EHR). However, issues have stemmed from an inability to prevent attacks. Additionally, there is not a set of standards for (DO) technique. Because of these challenges, researchers have proposed a more robust (DO) technique that is known as Nearest Neighbour Data Substitution (NeNDS), which is favoured because of its privacy protection features and ability to sustain data clusters. The same researchers continue to prove that reverse engineering is easily accomplished with geometric transformations related to cluster preservations.

2. Problem Statement

Intellectual property theft is one of the most challenging problems of todays technological era. According to Business software alliance global software piracy rate went noticeably high which lead to loss of $53billion in 2008. Due to the lack of security, software vendors have implemented security algorithms, techniques and tools, but with the help of various reverse engineering tools, software reversers are able to reveal the security algorithms to reveal the original code from the source file.

IT industry loses tens of billions of dollars annually due to security attacks such as reverse engineering. Code obfuscation techniques counter such attacks by transforming code into patterns that resist the attacks. The use of popular languages such as java increases an attacker's ability to steal intellectual property (IP), as the source program is translated to an intermediate format retaining most of the information such as meaningful variables names present in source code. An attacker can easily reconstruct source code from such intermediate formats to extract sensitive information such as proprietary algorithms present in the software.

Obfuscation is known to be the most common and effective technique to prevent prohibited Reverse Engineering. However, none of the current obfuscation technique meets and satisfies all the obfuscation effectiveness criteria to resistance the Reverse Engineering. None of the current code obfuscation techniques satisfy all the obfuscation effectiveness criteria such as resistance to reverse engineering attacks and state space increase. A determined attacker, after spending enough time to inspect obfuscated code, might locate the functionality to alter and succeed in the malicious purpose. The renaming obfuscation, layout obfuscation, harden the code obfuscation, and source code obfuscation can be attacked by the reversing tools that are able to perform analysis to create a new name for the identifiers that are used in the source file [4].

Most of the developers have practiced using only one obfuscation technique to protect the code and used the technique for certain part only of the code. Having one technique to protect the code is proven not to be effective enough to prevent prohibited reverse engineering. Reversing tools are very advanced currently as they can create new code from the obfuscated code that performs the same output even though the original code is obfuscated.

3. Related Studies

Obfuscation technique has many forms and methods to protect the code in the source file. Below discussion presents the forms of obfuscation technique.

3.1 Renaming Obfuscation Technique

It is the process to replace de identifiers with meaningless strings as new identifiers. Usually, the identifiers have meaning for a better recognition of the source code structures like classes, methods, variables and so forth. Once an identifier is renamed, it is mandatory to provide consistency across the entire application through replacements of the old names by the new identifiers. For programming languages that allow the method overloading, method name obfuscation is made by the same identifier string for the methods having different signatures [5].

3.2 Source Code Obfuscation

It is the process to hide the source code meaning when this code is disclosed to a third party to test or maintain it. This technique supposes renaming of the program identifiers and removing the comments. The obfuscation of mobile device software programs leads to a smaller code and faster running. The operations for software obfuscation are implemented by specialized software products called obfuscators [6].

3.3 Packing

Packing measured as one of the significant anti-Reverse Engineering methods in the Microsoft Windows environment. However, several of reversing attacks are usually conducted in the Linux-based embedded system. Packing does not have secure binary tools for LINUX. However, it has for Microsoft applications and systems. In packing methodology, there are two secure packing methods, which are AES encryption and the UPX packer to protect the intelligent property (IP) of software from Reverse Engineering attacks. UPX system is designed for code compression; UPX contains of two anti-debugging methods in the unpacking module of the secure to detect Reverse Engineering attacks. Besides, these two methods are analysed based on their performance from prospective.

- Code size,
- Execution time
- Power consumption.

The analysis results show that the Secure UPX performs better than AES-encryption packing in terms of the code size, execution time, and power consumption. The concept of code packing is converting or transforming the software

into packed program, by compressing or encrypting the original code and data into packed data and combining it with a restoration routine [7].

Restoration routine is a section of code to recover the genuine code and data and setting an execution context to the original code when the packed program is executed. Moreover, packing is one of the effective Anti-Reverse Engineering because it is combinations of multiple techniques, which are anti-debugging, obfuscation, call redirections.

3.4 Key Hiding Obfuscation

Key hiding is a method to protect intellectual property, it is key hiding based. This technique is merged with another encrypting technique. The merging technique can vary according to the way of protection; in this case, symmetric mechanism is used to merge with the key hiding. Key hiding focuses on software executables. The software protection key hiding then is encrypted by threshold key scheme to make it hard for the reverser to find it to break the code. This technique focuses on executable software, leaving the source code and class file as they are [8].

3.5 Design Level Obfuscation

Design level protection of the software is stopping the reverser before reaching the code level which in the java case is class file. The developers assumption is by hiding the design from the reverse, there is no way to know the code meaning and purpose.

To protect a design from design reverse engineering attacks after final product release, depending on the required level of security, a novel mechanism was proposed to select and replace custom CMOS gates in circuit netlist with reconfigurable STT-based LUTs during design implementation.

While an untrusted foundry may have access to the reconfigured design after its release to the market, the selection of custom CMOS gates for replacement is such that the untrusted foundry cannot determine the functionality of reconfigurable LUTs in any reasonable time. The selection mechanism will ensure that original design parametric constraints such as design performance will impact only minimally [9].

4. Limitation of Curent Obfuscation Techniques

The obfuscation techniques provided by different developers contain several gaps. The obfuscation techniques can protect the code to some level; however, the code contains some debugging information. There is not any obfuscator tool found which can be fully declared as the best obfuscation technique. If the secret of any obfuscator identified, reverse engineers can easily perform their tasks by constructing de-obfuscators. These tools of de-obfuscator not released yet but, in the future, there can be possibility of developing de-obfuscators [10].

The byte code contains unknown characters and symbols from the source code. Reverse engineers have broken the secrets of byte code by using Reverse Engineering tools, therefore, it is possible to copy the original code after reversing, improve upon it and re sell it in the market to gain benefit over the original author who has originally developed the code. Some software development companies allocate a hacker or a reverser who must break their code and find out the weakness and vulnerabilities of the software, so the company can fix it before it is hacked for real.

All software programs contain a security key or registration file that holds the protection of the software, reverse engineers convert that file into source code when they remove the registration file from software and use the revealed code for their own illegal development purpose [**?**].

Table 1 presents a summary of several obfuscation techniques limitation.

5. Proposed Obfuscation Technique

The proposed technique contains three approaches of renaming technique. The three approaches are presented below.

5.1 UNICODE Renaming Approach

first approach of renaming is to use UNICODE conversion. The purpose of this approach is to increase the complexity of the code in the source code. Table 2 presents the code before and after conversion.

5.2 String Encryption

Second approach of the proposed technique is string encryption technique. A mathematical equation is used to encrypt the strings in the source file and create chaos stream. The purpose of this approach is to confuse the de-compiler and reverser while reading the reversed code. The mathematical equation contains of four variables to convert the character in the source file into unknown symbol. The following equation presents the mechanism of conversion.

$$\text{Char} = \frac{V}{2+Y+Z}$$

Figure 2 presents the string after converting with the mathematical equation.

```
for ( int ѫ0110: "¤¼ÀÌ¾Ë è ¬□ÀÈµË".toCharArray())
{ System.out.print((char)(ѫ0110 / 2 + Y - Z)); }
```

Figure 2. Text after string encryption obfuscation

5.3 Identifiers Renaming to Junk Code

Junk conversion creates an opportunity to create variety of languages while developing the software for the sake of protection. The class file contains the junk code after compiling the source file. After using the junk conversion, the converted code will be converted again to junk in the class file which increases the level of protection. Below algorithm is to transform identifiers to junk. Figure 3 illustrates the code after conversion to Junk code.

Table 1. Limitation of current obfuscation techniques

List	Limitation
• Logistic map • Cipher block chaining • Symmetric cipher	These techniques focus of chaos stream. They use mathematical equations to replace the text in the string with chaos stream. The technique uses a secret key for the encryption and uses one mathematical equation. The key can be randomly generated the time of encryption or purchased from developer. If the reverser were able to guess the key, then there a possibility to use the key to decrypt the entire code [**?**].
• Renaming • Hiding	These techniques Focus on hiding the codes function and changing layout. This is harder to understand but not impossible on Reverse Engineering process. These tools might hide the code somehow. Still Reverse Engineering process is possible [11].
Key hiding obfuscation	This technique focuses on executable software, leaving the source code and class file as they are. Reversing tools currently can find the key and break it to get to the source file and perform analysis on the code.
Encryption	This technique focuses on cryptographic techniques to encrypt and pack executable code inside binary file. The main limitation of cryptographic implementations is the lawful regulations imposed on programming vendors. Either they limit key or sizes, or they leave just stubs for restricted classes. Encryption uses longer keys to provide better security. While the longer key length to slower encryption speed.
Packing	This technique packs the entire code into one package. The reversing tool currently can unpack the packed code and create new code that is useful and produces same output as original.
Classes combination obfuscation	This technique focuses on hiding the classes by combining them together. The reversing tools can create new classes and open the combined classes. The reversing tools contain a great analysis feature that can find the classes trees and the connection.
Junk code obfuscation	This mechanism focuses on changing the variables and classes names. The renaming process intends to confuse the reverse rather than preventing the reverse engineering process. The reversing tools can create new names for the variables and classes by using characters. Then the reverser can use the refactor feature to create meaningful names.

Table 2. System Keywords Conversion to Unicode

Java system keywords	Unicode
package backencrypt	\u0070\u0061\u0063\u006B\u0061 \u0067\u0065\u0062\u0061\u0063 \u006B\u0065\u006E\u0063\u0072 \u0079\u0070\u0074

捻 \u003D \u006E\u0065\u0077 捻\u0028\u0029\u003B ѣ011邏\u003A"□ÌÈËÀ¼¬Æ ¾¤¼¬ "\u002E \u0074\u0074ѣ011邏 \u003A"□ÌÈËÀ¼¬Æ ¾ ¤¼ ¬"

Figure 3. Identifiers after converting to junk code

5.4 Apply Hybrid Obfuscation Technique in One Source File

Java development is based on object orientation whereas the compiler runs the application based on components unlike structured programs that are developed by C programming language. Therefore, obfuscating the code will not create problem while compiling into machine language or byte code. To use this hybrid obfuscation technique, certain steps must be followed; first step is to use the object junk renaming obfuscation. This conversion must be done firstly to prevent confusion and errors when obfuscation process is running. Second step is to use string encryption obfuscation; this technique must be done secondly, for the developer to encrypt all the strings at once. Final step is Unicode renaming obfuscation technique. Carrying out this proposed technique increases the security level of the code and complicates the reversing process. The string encryption makes the obfuscation technique more effective because it contains so many symbols that help to confuse the de-compiler while parsing and analysis.

6. Conclusion

Hybrid obfuscation technique is complicated to use. It is however effective as it confuses the compiler while reversing and it confuses the reversing tool while reversing and analysis. The complicated and renamed code misleads the reverser and the reversing tool. Reversing tool produces errors and illogical code after reversing the obfuscated code. The proposed technique does not prevent the reverse engineering; it however confuses the reverser and the reversing tool. The implementation of the proposed technique was implemented in NetBeans environment and was tested to evaluate the effectiveness and correctness of the code with four reversing tools.

References

[1] J.-W. Jang, A. De, D. Vontela, I. Nirmala, S. Ghosh, and A. Iyengar, "Threshold-defined logic and interconnect for protection against reverse engineering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[2] N. Gomes, "Software piracy: An empirical analysis," Ph.D. dissertation, 2015.

[3] Z. Tang, K. Kuang, L. Wang, C. Xue, X. Gong, X. Chen, D. Fang, J. Liu, and Z. Wang, "Seead: A semantic-based approach for automatic binary code de-obfuscation," in *2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE, 2017, pp. 261–268.

[4] A. M. Alhakimy and A. B. M. Sultan, "Hybrid algorithm to protect java's code from reverse engineering," *Lecture Notes on Software Engineering*, vol. 3, no. 1, p. 11, 2015.

[5] C. K. Behera and D. L. Bhaskari, "Different obfuscation techniques for code protection," *Procedia Computer Science*, vol. 70, pp. 757–763, 2015.

[6] K. Kuang, Z. Tang, X. Gong, D. Fang, X. Chen, T. Xing, G. Ye, J. Zhang, and Z. Wang, "Exploiting dynamic scheduling for vm-based code obfuscation," in *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2016, pp. 489–496.

[7] J.-M. Borello and L. Mé, "Code obfuscation techniques for metamorphic viruses," *Journal in Computer Virology*, vol. 4, no. 3, pp. 211–220, 2008.

[8] A. Capiluppi, P. Falcarin, and C. Boldyreff, "Code defactoring: Evaluating the effectiveness of java obfuscations," in *2012 19th Working Conference on Reverse Engineering*. IEEE, 2012, pp. 71–80.

[9] S. Budhkar and A. Gopal, "Reverse engineering java code to class diagram: An experience report," *International Journal of Computer Applications*, vol. 29, no. 6, pp. 36–43, 2011.

[10] A. Li, Y. Zhang, J. Zhang, and G. Zhu, "A token strengthened encryption packer to prevent reverse engineering pe files," in *2015 International Conference on Estimation, Detection and Information Fusion (ICEDIF)*. IEEE, 2015, pp. 307–312.

[11] G. C. Deshmukh and S. Patil, "Study for best data obfuscation techniques using multi-criteria decision-making technique," *International Journal of Computer Applications*, vol. 180, no. 43, pp. 50–57, 2018.