Code Obfuscation for Effectively Securing Data in the Web-Based Industry

Priyanka Singh, Vriti Sidana, Kanu Priya Aggarwal, A. B. Patki, and R. C. Meharde

***Abstract*—Today reverse engineering poses a serious threat to the protection of intellectual property rights (IPR) of software developers. The incidents of software tampering and piracy have become commonplace. It is possible for software pirates to extract a piece of code and incorporate it into their own programs with considerable ease. Such cases are even more rampant in the web-based industry since it is teeming with programs in easily decompilable formats. In this paper, we target java applications which are available as Java class files. Java bytecode is platform independent and makes Java executables highly susceptible to being reverse engineered. Obfuscation is a technique to protect against this threat. This paper presents a level-based organization of code obfuscation for employing effective software protection. The authors propose incorporation of obfuscation as a means to improve MIQ of software applications. Finally, the paper demonstrates the working of a code obfuscator that operates on java files and produces obfuscated versions in two stages.**

***Index Terms*—Intellectual property rights, machine intelligence quotient (MIQ), obfuscation, reverse engineering, tamper-proofing, watermarking.**

I. Introduction

Today Web-based services have completely changed the industrial scenario. More and more companies are expanding their service offerings via the global portal i.e. the World Wide Web (www) to increase their customer base. The software industry with its evolving service capability also aims at bringing a number of services online. In fact, e-Governance is seen as one of the most viable solutions for improving the economic condition of the country.

As online computing and querying becomes pervasive, concerns about data protection and security have taken on new urgency [1]. Unauthorized users and clients should not be able to tamper with the program codes. What makes securing data difficult is that it is not static, but manipulated in a networked environment. Using reverse engineering, malicious parties can steal the intellectual property associated with such code with relative ease. To address this issue, a number of techniques have been proposed. Prominent among them are watermarking, tamper-roofing and obfuscation [2].

Obfuscation has become a trend in developing software modules for IPR protection. In this paper, we describe the need for employing effective software protection and propose a level-based organization of code obfuscation.

The remainder of this paper is structured as follows. Section II describes the backgrounds for code obfuscation. Section III discusses MIQ (Machine Intelligence Quotient) as a new software evaluation measure and introduces obfuscation as a MIQ parameter. Section IV proposes a three-level code obfuscation approach for achieving high degree of software protection. The implementation of our code obfuscator is illustrated in Section V. The paper concludes in Section VI.

Manuscript received August 29, 2012; revised September 29, 2012.

Priyanka Singh is with Accenture Services Pvt. Ltd. (e-mail: priyanka.singh059@gmail.com).

II. Background

A. Software Engineering

The advent of powerful decompilers has facilitated remarkably faster and easy source code extraction, thus exposing intellectual property to potential security threats. Consequently, software protection has become a prime area of interest. While it is believed that complete protection of software is an unattainable goal, recent researches have shown that protection can be achieved to a large extent. Software protection techniques are aimed at defending the intellectual property of the software against various types of attacks. The intellectual property can be the software design, data contained in the software or the algorithm. Basically three types of attacks on the intellectual property contained in software and their corresponding defenses are identified. A defense against reverse engineering is obfuscation, a process that renders software unintelligible but still functional. A defense against software piracy is watermarking, a process that makes it possible to determine the origin of software. A defense against tampering is tamper proofing, so that unauthorized modifications to the software will result in non functional code [2].

B. Reverse Engineering

In the software world reverse engineering means taking an application or program for which source code and documentation is not available and attempting to recover details regarding its design and implementation [3]. Reverse engineering is related to several aspects of software security. It is used to understand how a program performs some action and to bypass protection. Hence it is necessary to protect the software against such attacks and *code obfuscation* is a promising defense mechanism.

C. Code Obfuscation

The code obfuscation [2], [4] problem is formally stated as follows:

Given a set of obfuscating transformations $T=\{T_1, T_2,\ldots T_n\}$ and a program P consisting of source code objects(classes, methods, statements, etc) $\{S_1, S_{2,}\ldots S_k\}$,find a new program $P'=\{\ldots,S'_j= T_j(S_j),\ldots\}$ such that:

- *P*' has the same observable behavior as *P*, i.e. the transformations are semantic preserving.
- The obscurity of *P*' is maximized, i.e. understanding and reverse engineering *P*' will be strictly more time consuming than understanding and reverse engineering P.
- The resilience of each transformation $T_i(S_j)$ is maximized , i.e. it will be either difficult to construct an automatic tool to undo the transformations or executing such a tool will be extremely time consuming.
- The cost (the execution time/space penalty incurred by the transformation) of *P*' is etc.

III. Improving MIQ of Software Packages

Machine Intelligence Quotient (MIQ) has long been used as an evaluation measure for Human-Machine Cooperative systems [5]. However, until recently, no attempt had been made to measure software intelligence. To assess software products on the basis of their intelligence, authors [6] proposed an MIQ-based evaluation methodology for COTS (Commercial Off-the shelf) software components.

A. *Machine Intelligence Quotient (MIQ)*

The machine intelligence quotient (MIQ) is a measure to assess the intelligence of an autonomous system. Any index, numerical or linguistic framework indicating the degree of autonomy of an intelligent agent can be regarded as MIQ quantification.

MIQ can be considered as a union of machine control intelligence (M_c) and machine interface intelligence (M_F) [6] as

$$M = M_c + M_F \tag{1}$$

Control intelligence is composed of three control abilities to manage three different events that occur from the environment - known events, uncertain events and exceptional events. Interface intelligence indicates intelligence degree needed from human when human and machine exchange information of plant states or control commands.

B. *Obfuscation as MIQ Parameter*

In context of software applications, known events comprise legalized software use without failure. The uncertain and exceptional events might include:

- erratic software behavior
- attacks by malicious agents
- attacks by malicious hosts
- attempts to reverse engineer it back into the source code

We propose incorporation of obfuscation as a means to improve MIQ of software applications. One of the greatest challenges remains the lack of analysis techniques, and metrics for evaluating and comparing the strength of various software protection techniques [7]. MIQ evaluation measure can be applied to measure the effectiveness of software protection tools.

IV. Level Based Organization of Code Obfuscation

We aim to incorporate an enhanced code obfuscating security feature in software at three levels (as illustrated in Fig. 1)

Level 1: Incorporating tamper detection capability in the program. The program should be able to show its status (whether an attempt has been made to change it or not) whenever required.

Level 2: The program should display tamper resistance (armoring) and attempt to shield itself in the face of tampering.

Level 3: The program obfuscates itself dynamically whenever an attempt is made to decompile or change it. This dynamic obfuscation should be random (changing function names, classes, interfaces etc) so that the opponent cannot draw any kind of link between the components.

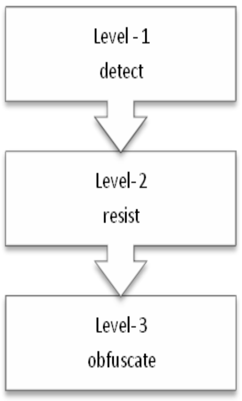

Fig. 1. Implementing software security at three levels.

A software program developed so as to include this security feature will not only obscure code but also employ self-defence against unauthorized access.

V. Software Implementation

We have designed a code obfuscator using Java [8] for Windows platform that works in two stages.

STAGE 1: The obfuscator accepts the candidate java file (to be obfuscated) as input and outputs an obfuscated version after removing comments and indentations from the program. This obfuscated version is written as a new java file in the target folder while the original file remains unchanged. Fig. 2. gives a block diagram representation of Stage 1 obfuscator.

STAGE 2: The obfuscator accepts the candidate java file (to be obfuscated) and a class in it as inputs and obfuscates the following fields of the input class –

1) Declared methods
2) Declared fields
3) Interfaces implemented by the class
4) Nested classes

The above fields are changed to random integers and the obfuscated version is printed as output. The obfuscated version is written as a new java file in the target folder while the original file remains unchanged. Fig. 3. illustrates this.

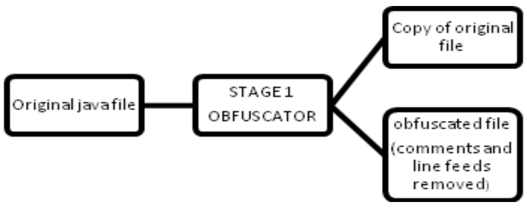

Fig. 2. Figure depicting inputs and outputs of Stage 1 obfuscator.

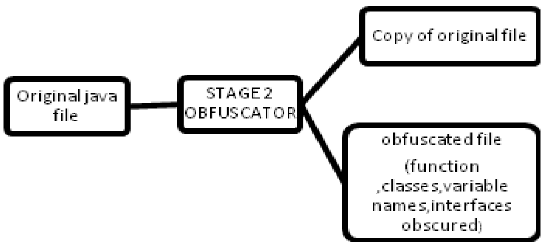

Fig. 3. Figure depicting inputs and outputs of Stage 2 obfuscator.

A. Working Illustration

Example 1 (Stage 1): When the sample java program in Figure 4 is input to Stage 1 obfuscator, an obfuscated version is output as in Figure 5.

```
class circle
{           //start of class circle
    double a,b,rad; //declaration of variables
    circle(double x,double y,double radius)
    {       //start of constructor
a=x;
b=y;
rad=radius;
               }
    circle(circle c)//single arguement constructor
               {
a=c.a;
        b=c.b;
rad=c.rad;
               }
void show()//function to display the value of variables
{
System.out.println("The value of the parameters are :");
        System.out.println("a = "+a+"  b = "+b+"  radius =
"+rad);
}

}// end of class circle
```

Fig. 4. Input java program (Stage 1 Obfuscator)

```
class circle{ double a,b,rad; circle(double x,double
y,double radius){ a=x; b=y;  rad=radius;   }circle(circle
c){ a=c.a; b=c.b; rad=c.rad; }    void show() {
System.out.println("The value of the parameters are :");
System.out.println("a = "+a+"  b = "+b+"  radius =
"+rad); }}
```

Fig. 5. Output java program (Stage 1 Obfuscator)

Example 2 (Stage 2): When the sample java program and its class in Figure 6 are fed as inputs to Stage 2 obfuscator, an obfuscated version is output as in Figure 7.

```
Person()
. . .
private String firstname;
private String lastname;
public String getFirstname() {
return firstname;
}
public void setFirstname(String firstname) {
this.firstname = firstname;
}
public String getLastname() {
return lastname;
}
public void setLastname(String lastname) {
this.lastname = lastname;
public static void main(String a[])
{
p.meth1();
p.meth2();
p.fun1();
. . .
```

Fig. 6. Input java program (Stage 2 Obfuscator)

```
Person()
{
. . .
      private String __19289;
      private String __71567;
      public String __470() {
        return __19289;
      }
       public void __40826(String __19289) {
                  this.__19289 = __19289;
      }
      public String __74317() {
          return __71567;
                                    }
      public void __41381(String __71567) {
          this.__71567 = __71567;
public static void main(String a[])
{
p.__14058();
p.__71200();
p.__71540();
. . .
```

Fig. 7. Output java program (Stage 2 Obfuscator)

VI. Conclusions and Future Work

Obfuscation is proposed as a MIQ parameter for software products. The authors propose a level-based approach to effectively secure software modules against potential security threats. The paper also demonstrates the working of a code obfuscator that operates on java files and produces obfuscated versions in two stages.

The existing obfuscation schemes are applied to programs in a static manner. There is no technique available to obfuscate a piece of code dynamically in the face of an attack. We strive to realize our proposed level-based approach by imparting dynamic decision-making ability to the software. Darwinian evolution accounts for the behavior of naturally occurring intelligent entities and can be used to guide the creation of artificial entities that are capable of intelligent behavior [9]. Our efforts are directed towards stimulating an evolutionary process in software programs so that subsequent obfuscated versions have increased MIQ.

REFERENCES

[1] G. Naumovich and N. Memon, "Preventing piracy, reverse engineering, and tampering," *IEEE Computer*, vol. 36, no. 7, pp. 64-71, July 2003.

[2] C. Collberg and C. Thomborson, “Watermarking, tamper-proofing, and obfuscation - tools for software protection,” *IEEE Trans on Software Engineering*, vol. 28, no. 8, pp. 735–746, Aug. 2002.

[3] E. Eilam, *Reversing:Secrets of Reverse Engineering*, 1st ed. USA: Wiley, 2005.

[4] K. Fukushima, S. Kiyomoto, and T. Tanaka, “ An Obfuscation Scheme Using Affine Transformation and Its Implementation,” in *IPSJ Journal*, vol. 47, no. 8, Aug. 2006, pp. 2556–2569.

[5] H. J. Park, B. K. Kim, and K. Y. Lim ,”Measuring the machine intelligence quotient (MIQ) of human machine cooperative systems,” *IEEE trans on Systems, machines and cybernetics-Part A, Systems and Humans*, vol. 31 , no. 2, March 2001.

[6] P. Singh, V. Sidana, K. P. Aggarwal, N. Verma, S. Verma, and A. B. Patki , “Introducing Machine Intelligence quotient as a new COTS Evaluation measure,” in *proc. 4th National conference*, INDIAcom 2010.

[7] P. C. Van Oorschot, "Revisiting Software Protection," in *proc. 6th International Conf. Information Security (ISC 03), LNCS 2851*, Springer-Verlag, 2003, pp. 1–13.

[8] H. Schildt, *The Complete Reference Java 2*, 5th ed. New Delhi, India: Tata McGraw-Hill, 2002. ch. 17, pp. 546-577.

[9] D. B. Fogel and L. J. Fogel, “Evolution and Computational Intelligence,” in *Proc. IEEE , Neural Networks*, vol. 4, pp. 1938-1941, Dec. 2005.

Ms. Priyanka Singh obtained her B.Tech from Maharaja Agrasen Institute of Technology, GGSIPU, New Delhi in 2011 in Computer Science. During her engineering, she did her summer internship at the Department of Information Technology, Government of India, New Delhi and co-authored a research paper “Introducing Machine Intelligence Quotient as a new COTS Evaluation measure” with her team. The paper was published in the proceedings of the 4th National conference, INDIAcom 2010.

During 2009-2011, she did exploratory work in software obfuscation with emphasis on effectively securing web-based Java files. She has undergone Green Field Training at Accenture Services Pvt. Ltd., India during Dec 2011–Feb 2012 where she was trained in Functional Testing, Test Data Management, Testing Automation and Testing tools primarily Quality Centre (QC) and Quick Test Professional (QTP). She is currently working as an Associate Software Engineer at Accenture Services Pvt. Ltd., Chennai.

Ms. Vriti Sidana completed her graduation from Maharaja Agrasen Insitute of Technology, GGS Indraprastha University, New Delhi in 2011 in Computer Science and Engineering. During her graduation she worked as a trainee at the Department of Information Technology, Ministry of Communications and IT, New Delhi during the period June-July’09 and June-July’10. The internship at the Ministry was her foray into the research field. She was an integral part of the team which worked on introducing Machine Intelligence Quotient(MIQ) as a measure of software intelligence. The team got their research paper published on the same topic in BVICAM.

She worked on a project –‘Code obfuscation for effectively securing data in the web based industry’ during 2009-2010. She is currently working as a Mobile application developer who specialises in iPhone/iPad applications at MapXL Inc., New Delhi. Her application, World Atlas for iPad, has got the best rating at the apple application store in the reference category. She has worked on various other applications like city guides, quizzes and India Atlas.

Ms. Kanu Priya Aggarwal completed her graduation from Maharaja Agrasen Institute of technology, GGSIPU , New Delhi in 2011 in Computer Science and Engineering. During her graduation she did her training at the Central Government Office , New Delhi and was an integral part of the team which was successful in publishing a research paper on ‘Machine Intelligence Quotient’ in BVICAM National Conference.

She worked on the project - ‘Code Obfuscation for effectively securing data in the web based industry’ during 2009-2011. She has undergone extensive training program at Infosys Limited, Mysore, India in Computer Sciences during Aug – Dec 2011 where she was trained in languages like java , .Net, Databases and Software development lifecycle. She is currently working as a Software Engineer at Infosys Limited, Chandigarh . Her current work areas include Content Management and Portals , FAST and Filenet.

Mr. A. B. Patki obtained his M Tech from Indian Institute of Technology (IIT), Kharagpur in 1975 with specialization in computers. He had worked as faculty in Government College of Engineering, Amravati, Maharashtra, during 1972-73. He also worked as Project Officer at IIT, Kharagpur during 1975-77 on hardware/software R&D projects. Since March 1977, he was employed with Department of Information Technology (erstwhile Department of Electronics), Government of India and superannuated in March 2010 as Senior Director/Scientist-G & HoD. He has worked in various capacities on several projects in the areas of Artificial Intelligence, Software Technology Parks, Reliability Engineering, VLSI Design, Soft Computing and E-Commerce. He was also instrumental in spearheading post-legislation activities of Information Technology (IT) Act, 2000. He has been member of Scientists selection committees in DRDO.

His current research areas include Soft Computing for Information Mining, Evidence Based Software Engineering, Professional Outsourcing, ICT enabled Productivity Enhancement, Cyber Laws, Cyber Ethics including cyber forensics & Chaos Theory applications information security.

Mr. Patki has been trained in VLSI Design at Lund University, Sweden and Mentor Graphics, USA. He has been a referee for IEEE Transactions on Reliability for over twenty years. He holds a copyright for FUZOS©- Fuzzy Logic Based Operating Software. He has over fifty International publications. He has delivered invited lectures at leading academic institutions. He has developed courseware for VLSI design using VHDL that has been used for training the teaching faculties at leading engineering colleges and microelectronic industry. He has been supervising B.Tech/ ME thesis and also imparting training for Engineering Interns in computer Science and Information Technology.

Mr. R. C. Meharde obtained his Bachelor of Engineering degree in Electronics with specilisation in Control & Instrumentation, from SGS Institute of Technology and Science, Indore in 1978. He had worked with Hindustan Copper Ltd., a PSU from April, 1979 to January, 1981. In January, 1981 he joined the Directorate General of Light Houses and Lightships, Ministry of Shipping and Transport, Government of India. Since February, 1987, he is employed with Department of Information Technology, (erstwhile Department of Electronics), Government of India, which has been recently renamed as Department of Electroncs and Information Technology by the Government of India.

He has worked extensively in field of Control & Instrumentation, Coastal Navigation System, Futuristic Navigation Systems. His current areas of interest include promotion of Electronics and IT systems and application in areas of Industrial Electronics, Automation System, Intelligent Transportation System and Power Electronics. He has contributed for evolving several R& D projects in these areas which have been successfully implemented and resulted into technology transfer amongst Indian Industry. Presently, he is Senior Director/Scientist G-HoD Electronics Systems Development & Applications (ESDA).