Code Obfuscation by using Array Transformation Techniques

R. Deepas Babu
Dept. of Comp. Science & Systems Engg.
Andhra University College of Engineering (A)
Visakhapatnam, India

Chandan Kumar Behera
Dept. of Comp. Science & Systems Engg.
Andhra University College of Engineering (A)
Visakhapatnam, India

D. Laitha Bhaskari
Dept. of Comp. Science & Systems Engg.
Andhra University College of Engineering (A)
Visakhapatnam, India

***Abstract*— Source codes generally face a high risk of piracy, code tampering and reverse engineering attacks. In order to handle the piracy related attacks, obfuscation is one of the best techniques, which is used to make the reverse engineering much harder. Program having different components like control statements, loops, storage specifiers, constants, expressions etc. this paper targets on arrays (if available in the program). Arrays can be split, merged, folded and flattened. This research work will help to obfuscate the arrays available in the program by that the program can be obfuscated.**

Keywords- Code Obfuscation; Array Splitting; Array folding; Array flattening;

I. INTRODUCTION

IT industries spend billions of dollars annually for preventing from security attacks such as tampering and malicious reverse engineering. Because of huge application and development on internet technologies and multimedia, the vast necessity for research on security and protection is needed. Every organization is having its own intellectual ideologies and it is a big challenge for them to protect their data, form software piracy or injection of malicious code etc. Also data processing through the application should be confidential, otherwise understanding the data may damage the software purchaser's business directly [1].

Obfuscation is a technique in which code is modified to prevent reverse engineering and produce the code in such a way that no one can understand except the programmer. Obfuscation is also applied to programs to ensure intellectual property protection from reverse engineering. Simply, obfuscation techniques can be used to dynamically convert source code into a program that works in the same way, but is much harder to understand and to crack [2]. The better idea is to use mechanisms or methods within the application software. By that the system software will not be vulnerable. There is another method called obfuscation, a novel area of research in the field of software protection, which has been gaining more importance at present. Usually encryption and firewalls are used on software/application to keep from threat of the attackers. But, these approaches do not help to protect the software, when the attacker is an end-user. Among the various code protection methods from different attacks, code obfuscation is one of the most popular alternative, for preventing from code comprehension, code tampering. Now-a-day's code obfuscation is a largely adopted solution, and many different obfuscation approaches has been proposed. This is also a type of software protection against unauthorized reverse-engineering [3]. However, a determined attacker, after spending enough time to analyse obfuscated code, might find the functionality to alter and succeed in her/his malicious purpose. For this reason, obfuscation techniques are implemented with other approaches, such as code replacement/update, code tampering detection, protections through updating so that the attacker are bound with in a time frame which is hold to attack [4]. Practically, encryption, protection by server-side, hardware-based security solutions, different signed native codes, tamper proofing, watermarking, software aging, packing are some of the most commonly used fields, where obfuscation techniques can be applied. Generally obfuscation methods include code re-order, code replacement with meaningful identifier names in the original code with meaningless random names, junk code insertions, unconditional and conditional jumps, insertion branch, variable reassign, dead code insertion, merge local integers, string encoding, suppression of constants, meshing of control flows and many more [5].

A. Types of obfuscation

a. Data obfuscation: Data obfuscation is a form of data covering where data is purposely jumbled to prevent unauthorized access to sensitive information. There are two types of data obfuscation encryption methodologies:
   1. Cryptographic Data Obfuscation: Input data encoding prior to being transferred to another encryption schema.
   2. Network security Data Obfuscation: Payload attack methods are purposely enlisted to avoid detection by network protection systems.

b. Code obfuscation:

It is a technique used to protect software against malicious reverse engineering attacks or users. Obfuscating a source code means to transform the code to another form which will be harder to understand even with the help of tools. But one can specify exactly what to be hidden and what not to be. Also, the obfuscated code

will be difficult to piracy and tampering, but has the same functionality as the original program [6].

B. Reverse engineering:

Reverse engineering is a technique used to analyse software in order to understand and recreate the program by exploiting its weaknesses or strengths. Software companies reverse engineer there products to find out where and how improvements can be made even if they. Some companies use reverse engineering when they don't have similar products yet, to improve products of their own. Those who intend to build their product based on an existing one, often prefer to reverse engineer for creating from scratch [7, 8].

C. Deobfuscation:

Deobfuscate is a method to convert a program that is difficult to understand into one that is simple, understandable and straightforward. There are tools available also to Deobfuscate. Obfuscation may also be used to conceal malicious content in software, so de-obfuscating tools are used to reverse-engineer such programs [9].

D. Advantages and Disadvantages

Important advantages of obfuscation techniques are:

a) Protection: obfuscation protects against static and dynamic analysis attacks i.e. the attacker requires more time or resources to achieve his goal.
b) Diversity: it is possible to create different instances of the original program.
c) Low cost: obfuscation involves a low maintenance cost due to automation of the transformation process and compatibility with existing systems.
d) Platform independence: obfuscation transformations can be applied on high-level code so that platform independence is preserved.

Important disadvantages:

a) Performance overhead: every transformation introduces an extra cost in terms of memory usage and execution time necessary to execute the obfuscated program.
b) No long-term security: Obfuscation by code transformation does not provide perfect security, i.e. makes analysis infeasible though not impossible.

II. CODE OBFUSCATION

Many researches had been placed in order to achieve the most level of obfuscation and deter reverse engineering form de-compilation of the code. Many free products and commercial tools available obfuscation technique and also highly required. Inspire of many free products and commercial tools still there is a high need of implements of obfuscation techniques, became implementing those techniques. The size of the source code files and obfuscated additional computation time increase, which reduces the efficiency of program [10].

A. Array folding

By using folding method n-dimensional array is folded into n+1–dimensional array, n+ 2 dimensional arrays… so on. For suppose, one-dimensional array can be folded to 2-dimensional array, two-dimensional array folded to 3-diemsional array, one-dimensional array folded to three-dimensional… So on.

If the data is in high-dimensional array, the attacker or reverse engineer can easily understand and crack the code. To make harder to understand the given high-dimensional array logic is implemented using array folding techniques (converted to high-dimensional array) and adding the code obfuscation techniques. The code becomes harder to perform reverse engineering and code piracy, tempering [11].

B. Array flattening

By using flattening method n-dimensional array is flattened (transformed) to n-1 dimensional array, n-2 dimensional array … So on. Suppose 3-dimensional array flattened to 2-dimensional array, 2-dimensional array is flattened to 1-dimensional, and three-dimensional array is flattened to 1-dimensional array.

If the data is in high-dimensional array, the attacker or reverse engineer can easily understand and crack the code. To make harder to understand the given high-dimensional array logic is implemented using array flattening techniques (high-dimensional array converting to low-dimensional array) and adding the code obfuscation techniques. The code becomes harder to perform reverse engineering and code piracy, tempering [12].

C. Array Splitting

Array splitting is nothing but splitting an array of "n" elements into m-sub arrays and each sub-array size is "x". Decide the number of sub-array and each sub-array size is depended on the original array size. If the original array size is increased then automatically sub-arrays size will increase and also increases the number of sub-arrays.

If the data is in array format, the attacker or reverse engineer can easily understand and crack the code. To make harder to understand the given array logic is implemented using array splitting techniques (more than one array) and adding the code obfuscation techniques [13]. By using the above combination the code of the program become much harder to crack, code tampering and code piracy by attacker or reverse engineering.

D. Array merging

The array merging technique is one of the code transformation techniques. This technique generally used for hiding data. So, this technique can be useful to obfuscate the data of the arrays basically, 'n' number of arrays can be combined or merged to make less than 'n' number of arrays with storing the data in different sequences [14].

III. EXITING TECHNIQUES

The existing techniques used in proposed methods for using obfuscating algorithm

i. Remove comments in a given program
ii. Remove whites spaces in a given program
iii. Replace the variable names with meaningless names

Example:

Before code obfuscation source program

```
#include<stdio.h>
void main ( )
{
  int a, b, c; //this is declaration of variables in c
  printf (“Enter any three values”);
     // printing the something on the screen in c
  scanf(“%d %d %d ”,&a, &b, &c);
    // reading inputs from keyboards in c
  if (a>b)
       c=a; /* this condition is true a is big and
            assessing the value of a is c */
  else
       c=b; // this condition is true a is big and
             assessing the value of b is c
 }
```

After code obfuscation source program

```
#include<stdio.h>#define ~au~ if
#define ~au` else #define ~au% a
#define ~au%~ b #define ~au%au c
#define ~au%au~ int #define ~au%u main()
~au%u { ~au%au~ ~au%,~au%~,~au%au ;
printf(“Enter any three values”);
scanf(“%d %d %d”,~au%, ~au%~,
~au%au);~au~(~au%>~au%~)
~au%au=~au%~;  ~au`~au%au=~au%~;}
```

After execution of the above functions like comments removing, whitespace removing and replace the variable names with meaningless variable names.

IV. PROPOSED TECHNIQUES

This section presents proposed techniques based on arrays splitting, folding, flattening and merging are the new methods which are proposed and implemented. These methods are extension to the exiting techniques on arrays which are expiation below in detail.

A. Array Splitting

Array splitting is nothing but splitting an array of “n” elements into m-sub arrays and each sub-array size is “x”. Decide the number of sub-array and each sub-array size is dependent on the original array size. If the original array size is increased then automatically sub-arrays size will increase and also increases the number of sub-arrays [15].

Here array splitting obfuscation technique is discussed. The proposed array splitting technique changes the structure of the array by placing the elements in new arrays. (Number arrays are based on the original array size. If array size increases number of arrays are also increases.) It is showing as it single array is used, but internally the data will be stored in more than one array.

```
int A=new int [n] int [] B1=new int [x];
                  int [] B2=new int [x];
                  int [] B3=new int [x]
                                .....
                  int [] Bm=new int [x];
```

B. Array flattening: (Reduce multi-dimensional to low-dimensional)

If the data is in high-dimensional array, the attacker or reverse engineer can easily understand and crack the code. To make harder to understand the given high-dimensional array logic is implemented using array flattening techniques (high-dimensional array converting to low-dimensional array) and adding the code obfuscation techniques. The code becomes harder to perform reverse engineering and code piracy, tempering [16].

a) Three-Dimensional array to One-dimensional array transformation (array flattening):

To make harder to understand the given one-dimensional array logic is implemented using array flattening (one-dimensional array converting to two-dimensional array) and adding the code obfuscation techniques the program becomes harder to understand reverse engineering. Let us consider a three-dimensional array is transformed into one-dimensional array. The given three-dimensional array is

b) Three-Dimensional array to two-Dimensional array transformation:

To make harder to understand the given three-dimensional array logic is implemented using array flattening (three-dimensional array converting to two-dimensional array) and adding the code obfuscation techniques the program becomes harder to understand reverse engineering. The given three-dimensional array A[z_di][x_di][y_di] is transformed into n two-dimensional arrays. The number of two-dimensional sub arrays are “n” and each two-dimensional sub array size is nc*nr (for a natural numbers nr and nc. nr indicates the number of rows and nc indicates the number of columns in a two-dimensional array).

c) Two-Dimensional array to one-Dimensional array transformation:

To make harder to understand the given two-dimensional array logic is implemented using array flattening (two-dimensional array converting to one-dimensional array) and adding the code obfuscation techniques the program becomes

harder to understand reverse engineering. The following sample source code reads the two-dimensional array values into one- dimensional.

C. Array folding: (Increasing 1-dimensional to multi-dimensional.)

If the data is in high-dimensional array, the attacker or reverse engineer can easily understand and crack the code. To make harder to understand the given high-dimensional array logic is implemented using array folding techniques (converted to high-dimensional array) and adding the code obfuscation techniques. The code becomes harder to perform reverse engineering and code piracy, tempering.

a) One-Dimensional array to two-Dimensional array transformation:

To make harder to understand the given one-dimensional array logic is implemented using array folding (one-dimensional array converting to two-dimensional array) *and* adding the code obfuscation techniques the program becomes harder to understand reverse engineering. Let use consider a one-dimensional array is A[n]. This array is transformed into two-dimensional array. The one-dimensional array size is n. if check the given n is prime number than add the n +1 otherwise cell the array folding algorithm. This algorithm stored the row size is temp[0] and column size is temp[1]. The following sample source code read in two-dimensional array values.

b) One-Dimensional array to three-Dimensinal array transformation:

To make harder to understand the given one-dimensional array logic is implemented using array folding (one-dimensional array converting to three-dimensional array) and adding the code obfuscation techniques the program becomes harder to understand reverse engineering. The one-dimensional array size is n. if check the given n is prime number than add the n +1 otherwise cell the array folding algorithm. is transformed into three-dimensional array. This algorithm stored the x-direction size is temp[0] and y-direction size is temp[1] and x-direction is temp[2]. The following sample source code read in three-dimensional array values.

```
for i=0 to temp[0] do // z-direction
  {
    for j=(0 ; j<temp[1];j++)  // y-direction
     {
      for (k=0 ;k<temp[2] ;j++) // x-direction
        {
         read a[i][j][k];
                    //read the data in three dimensional
      } //end for loop
    } //end for loop
} //end for loop
```

c) two-Dimensional array to three-Dimensinal array transformation:

To make harder to understand the given two-dimensional array logic is implemented using array flattening (two-dimensional array converting to three-dimensional array) and adding the code obfuscation techniques the program becomes harder to understand reverse engineering. The two-dimensional array size is A[row_size][col_size]. Multiplying the row size and column size n= row_size * col_size. If check the given n is prime number than add the n +1 otherwise cell the array flattening algorithm. This algorithm transformed into three-dimensional array. This algorithm stored the x-direction size is temp[0] and y-direction size is temp[1] and x-direction is temp[2]. The given two-dimensional array is A[row_size][col_ize]. The following sample source code reads the two-dimensional array values into one- dimensional.

```
for i=0 ;i<temp[0] ;i++) // z-direction
   {for j=(0 ; j<temp[1];j++) // y-direction
      {for (k=0 ;k<temp[2] ;j++) // x-direction
         {read a[i][j][k];
                    //read the data in three dimensional
          } //end for loop
       } //end for loop
} //end for loop
```

V. FRAMEWORK

A proposed framework is implemented for a generic program obfuscator which deals with obfuscation techniques related to arrays. As shown in the below diagram, any source file will be taken as input and an equivalent obfuscated source file will be generated as output file by applying all the proposed array obfuscation techniques.

Step1: Input the source file.

Step2: Generate a random sequence, for calling the modules in that sequence

Step3: Each module is defined for a particular obfuscation technique. After generating the random sequence, the corresponding modules will be executed in that order. All modules take a file as input file and then applying corresponding techniques. Then the output file will be generated. This process will be continued for all the modules and finally obfuscated file will be generated.

Step4: Obfuscated file can be excited or run then gives the output successfully same as original program.

A. Workig of each module:

1. Input the source file.
2. *Removing comments from input program*: This module takes as input source file then recognizing comments and removing comments then generating output c-source file as without comments in our program.
3. *Removing white spaces from input program*: This module takes as input source file then recognizing and remove the white space in our program then it's

generating output source file as without waste of white space in our program.

4. *Identifying the variables from input program*: This module takes as the input file is any program check the throughout in our program have variables then its replace the new variable names and also keywords names. Replace the name by same confused type variable names (or) useless variable names.
5. *Array folding*: This module takes as the input file is any program it's check the throughout in our program have low-dimensional arrays then its replace the high-dimensional array source code.
6. *Array flattening*: This module takes as the input file is any source program. if any high-dimensions array then its replace the low-dimensional array source code (it's reduce the high dimensional to low dimensional).
7. *Splitting of array*: This module takes as the input file is any source program identifying any one-dimensional arrays then its replace the splitting array source code "n"sub-arrays.

VI. CONCLUSION AND FUTRE WORK

Due to the increasing rate of piracy, a novel attempt is made in this paper based on array obfuscation. Normally after obfuscation, the complexity of the code increases logically as well as structurally, because of the insertions, removal or rearrangement of the data. The techniques proposed in this paper implemented and are found to be effective with minor increase in the time complexity. The proposed array obfuscation techniques like Array folding flattening and splitting merging is implemented in this paper to be simple but are very complex to analyze this feeing the purpose of obfuscation. Also, this paper helps in understanding array transformation concepts which can be futures used in array obfuscation techniques.

REFERENCES

[1] Zhu W, Thomborson C D, Wang F Y. Obfuscate array by homomorphic function.GrC.2006:770-773

[2] Christian Collberg, C. Thomborson, D. Low, A taxonomy of obfuscating transformations, Technical Report 148, Dept. of Computer Science, Univ. of Auckland, 1997.

[3] Marius Popa ,Binary Code Disassembly for Reverse Engineering, Journal of Mobile, Embedded and Distributed Systems, ISSN 2067 – 4074 , vol. IV, no. 4, 2012

[4] Jean-Maries Borello and Ludovic Me, Code Obfuscation Techniques for Metamorphic Viruses, Springer, 2008

[5] P. Mishra, A taxonomy of software uniqueness transformations, master's thesis, San Jose State University, 2003.

[6] M. Madou, B. Anckaert, B. De Bus, K. De Bosschere, J. Cappaert, B. Preneel, On the effectiveness of source code transformations for binary obfuscation. In H. R. Arabnia and H. Reza, editors, Software Engineering Research and Practice, pages 527–533. CSREA Press, 2006

[7] C. Collberg, J. Nagra, Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Program Protection. Addison-Wesley Professional, 2009.

[8] Kumar, Gaurav. "Best Plan for System Security." International Journal of Advance Research in Computer Science & Technology 3, 2015.

[9] R. Milner. A theory of type polymorphism in programming. Journal of Computerand System Sciences, 17:348–375,1978.

[10] C. Collberg and C. Thomborson.Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection. IEEE Transactions on Software Engineering, 28(8):735-746, August 2002.

[11] P.Sivadasan and P.Sojan Lal. "ARRAY BASED JAVA SOURCE CODE OBFUSCATION USING CLASSES WITH RESTRUCTURED ARRAYS", CoRR 2008.

[12] S. Praveen, P. Sojan Lal 2007. Array Data Transformation for Source Code Obfuscation. Proceedings of World Academy of Science, Engineering and Technology (PWASET) Volume 21 MAY 2007, ISSN 1307-6884.

[13] Sivadasan, Praveen, Lal, P Sojan, Sivadasan, Naveen, "JDATATRANS for Array Obfuscation in Java Source Code to Defeat Reverse Engineering from Decompiled Codes", http://cdsweb.cern.ch/record/1128190, Sep, 2008.

[14] S. Praveen, P. Sojan Lal 2007. Array Data Transformation for Source Code Obfuscation. Proceedings of World Academy of Science, Engineering and Technology (PWASET) Volume 21 MAY 2007, ISSN 1307—6884

[15] Y. Zhong, M. Orlovich, X. Shen, and C. Ding. Array regrouping and structure splitting using whole-program reference affinity. In Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation (PLDI '04), 2004

[16] S. Drape. Generalising the array split obfuscation. Information Sciences, 177(1):202– 219, 2007.

.

[17] E. E. Ogheneovo and C.K. Oputeh, "Source Code Obfuscation: A Technique for Checkmating Software Reverse Engineering". International Journal of Engineering Science Invention ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726 www.ijesi.org Volume 3 Issue 5 May 2014 PP.01-10.

[18] Wei Ding, ZhiMin Gu, A Reverse Engineering Approach of Obfuscated Array, 18th International Conference on Advanced Communication Technology (ICACT), Jan. 31 – Feb. 3, 2016, DOI: 10.1109/ICACT.2016.7423318 2016F

AUTHORS PROFILE

Deepas Babu Rebbavarapu is a M.Tech Student in the department of Computer Science and Systems Engineering. He has done his B.Tech. from Prakasam Engineering College, Kandukur, Andhra Pradesh, India. His area of interests for reasearch is in software Obfuscation.

Chandan Kumar Behera is a Research Scholar in the department of Computer Science and Systems Engineering. He has done his M.Tech. from Indian Institute of Technology, Kharagpur, India. He is having more than 8 years of teaching experience, before joing into Ph.D. His area of interests for reasearch is in Code security and optimization.

Prof. D. Lalitha Bhaskari is a Professor in the department of Computer Science and Systems Engineering of Andhra University with a vast experience of more than 17 years. She was awarded as 'Young Engineer' for the year 2008 in the field of Computer Science by Institution of Engineers (INDIA) during the 93rd annual convention held at Jodhpur. One of her papers was judged as Best paper at WMSCI held during June -06 at Florida, USA. Her areas of interest include Data Security, Computer Vision, Cyber Forensics, Big Data, and Pattern Recognition. Apart from her regular academic activities she holds prestigious responsibilities like Associate Member in the Institute of Engineers, Life Member in CSI, Member in CRSI, Associate Member in the Pentagram Research Foundation, Hyderabad, India.