RESEARCH ARTICLE OPEN ACCESS

Comparison of Security Level between Control Flow Integrity (CFI) Method and Address Space Layout Randomization (ASLR) on Buffer Overflow Attack Protection on Windows 10 Operating Systems

Avip Kurniawan*, **Nezim****,**Ridwan*****
*Departement of Information Technology, Budi Luhur University, Jakarta
** Departement of Information Technology, Budi Luhur University, Jakarta
***Departement of Information Technology, Budi Luhur University, Jakarta

--************************----------------------------------

Abstract:

The buffer overflow problem has been around for some time and continues to be a problem that is always there. Messages in memory corruption in programs are usually exploited by attackers in the form of stack and heap based attacks. The main purpose of this attack is to change the flow of programs that allow attackers to execute arbitrin code such as opening a shell with the same access rights during the attack process. Some examples of protection that have been adopted that make the benefits for software increasingly sharp. This study compares the ASLR and CFI methods on Windows 10-based applications. From different results it shows that flow control integrity (CFI) is superior in level based on the address of space randomization method (ASLR), but professionally the CFI method has twice execution time. fold compared to ASLR.

Keywords— **Security, address-space layout randomization, Buffer overflow, control flow integrit, memory corruption**

--************************--------------------------------

I. INTRODUCTION

Security is a major concern of computer users today. The goal of computer security, among others, is to protect information against theft and availability of information. A computer security system is an effort made to secure performance and processes on a computer. Buffer overflow is one method used to exploit a computer system that has a weakness in one of the applications used by the system. This exploitation is known as an input validation attack which can result in a system crash (buffer overflow), that is, the variables available in the application cannot accommodate input that is intentionally overused (Foster, etc, 2004). The main purpose of attackers who abuse this type of vulnerability is to manipulate control data that is in the process of virtual memory addresses (Aristizabal, etc, 2013).

Buffer overflow was first documented by James P. Anderson in 1972. Since then attacks like Morris Worm in 1988 and the so-called vulnerabilities have been reported as called after use free in Internet Explorer in 2014. Despite buffer overflow attacks has been around for more than 40 years, attacks triggered by memory corruption continue to be a serious threat until now (Saito, etc, 2016).

Buffer overflow attacks account for most of all the number of attacks. Based on statistics from the Common Vulnerabilities Exposure (CVE) this institution is supported by the United States Department of Homeland Security whose data sources are taken from several sources such as the National Vulnerability Database (NVD), exploit-db and those that have been published in Metasploit, the number of vulnerabilities that reportedly continues to increase, as shown in Figure 1 is the application vulnerability data that has been reported. Based on these data, we can see the trend of the number of application volatility continues to increase even though it had decreased in 2011 and 2015, the 2017 vulnerability report has increased.

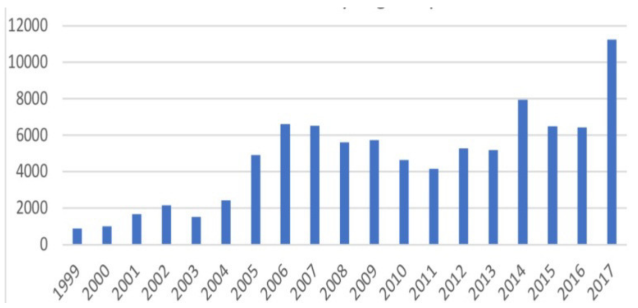


Figure 1: Application Vulnerability reported to CVE

If taken from the data buffer error error according to the number of reports continues to increase until 2017.

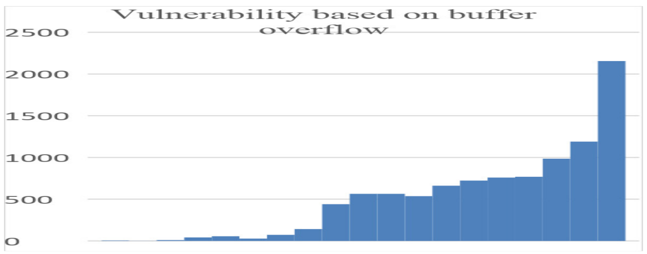


Figure 2: Vulnerability base on buffer error

Several protection mechanisms that have been adopted that make the utilization of vulnerabilities in software increasingly challenging (Joly, 2013), previous protection mechanisms such as Data Execution Prevention (DEP) (Andersen and Abella, 2004), use Address-Space Layout Randomization (ASLR) (PaX Team, 2003) and checks and makes sure the memory address that is directed to the target pointer is correct so as to prevent further exploitation of the application gap, this method is called Control Flow Integrity (CFI) (Abadi, etc, 2005).

Based on the protection characteristics described above can be a basis and consideration for comparing protection methods between ASLR and CFI on Windows 10-based applications.

II. LITERATUR REVIEW

2.1. *Memory corruption*

Memory corruption vulnerability is usually caused by a lack of input validation on the program created and programmers are given the freedom to determine when and how to handle input. This flexibility often results in better application performance. Examples of memory corruption attacks are buffer overflow, heap overflow, and stack overflow (Saito, etc, 2016).

2.2. Buffer Overflow

Buffer overflow is one method used to exploit a computer system that has a weakness in one of the applications used by the system.

Buffer Overflow is a software vulnerability created accidentally by the programmer. An example in a program, when writing data to allocated memory, if data is given to unallocated memory, it will overwrite the allocated memory limit. The most popular languages C and C ++ also do not have a built in boundary check, do not automatically check data trying to access the memory location of an array that is outside the limit (total size) of an array, examples can be seen in Figure 3

```
Void main ()
{
int buffer [10];
buffer [30] = 45;
}
```

Figure 3: Program with flaw

Here the total size of the array is 10, but the data assigned to location 30 is outside the array limit. Boundary checks can eliminate vulnerabilities.

The attacker's main goal is to take advantage of buffer overflow as an advantage and take privileged control of the program by subverting the function of the program.

2.3. Address Space Layout Randomization (ASLR)

Address space location randomization (ASLR) is a protection technique by randomizing the layout of virtual memory addresses introduced by PaX Team in 2003 on the Linux operating system as a mechanism to inhibit buffer overflow attacks. Microsoft first implemented ASLR on Windows Vista. This protection is used to ensure the address range of the memory address in the memory segment that is used is scrambled for each code execution or every boot system. In particular, ASLR scrambles the base address of the memory structure such

CFI was first introduced by Martin Abadi, et al in 2005. Control Flow Integrity (CFI) is a mitigation that prevents the transfer of control flow to unexpected space (Abadi, et al, 2005). CFI applies target address validation before it is used in indirect flow transfers. This approach detects the pointer code (including the sender's address) that is modified before being sent to a different target than the set that has been statically specified.

III. METHODOLOGY AND DESIGN

3.1.Research methods

This study applies a memory protection method from buffer overflow attacks on Windows 10 and compares the two methods that are still used today, namely address space layout randomization (ASLR) and control flow integrity. It is expected that with this research, it can be known that the method is superior in handling buffer overflow attacks and provides a reference comparison of buffer overflow attack protection methods.

This research was conducted using the experimental method. This experimental method is carried out by researchers by manipulating conditions according to the needs of the problems faced in the study. By manipulating this condition, the results of this study will display a comparison group so that the results achieved in this study are more accurate.

At the initial stage the method used in this study is the CFI and ASLR method. Each method is intended to secure the application when compiling the source code. Then each compilation result will be tested against stack overflow and heap overflow. Furthermore, exploitation will be carried out further with control-flow attack attack methods, code reuse attacks and code-injection attacks. And analyze it in memory using the tool immunity debugger.

3.2.Instrumentation

3.2.1 Hardware

The hardware used in this study is a laptop that has the following specifications:

Operating System	: Windows 10 Profesional Edition
Processor	: 2,6 GHz Intel Core i7
Memory	:16 GB 2133 MHz LPDDR3
Hard Disk	: 320 GB

3.2.1. Software

The following software is used in this study

1. Windows Operating System
2. Microsoft visual studio 2015
3. VMWare Fusion 8.5.7
4. Immunity debugger

3.3.Analysis technique

The analytical technique used in this study is quantitative data analysis techniques, carried out by analyzing the process of compiling the source code and then running it on Windows 10. Analyzing the differences in the quality of protection obtained after the compilation of source code.

The test will use an attack vector test platform to provide objective empirical data about the effectiveness of each protection mechanism.

3.4.Selection of Research Objects

The research object was chosen based on learning and the development of buffer overflow attack protection methods with its implementation in digital media, as well as the interest of researchers in the field of application security science.

The research object will use the application source code that will be compiled using both protection methods (ASLR and CFI).

IV. RESULT AND DISCUSSION

4.1. Analysis of Protection Methods

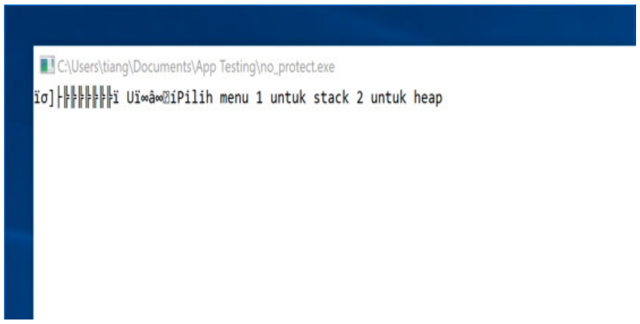
When compiling the source code in visual studio, the user first chooses the method to be used, the ASLR or CFI method.

The source code prepared has used several program functions according to the check vector attack platform checklist, each menu represents a checklist.

Performance testing is done using the features of Microsoft Visual Studio 2015, this will issue a report on the duration of execution of each protection method.

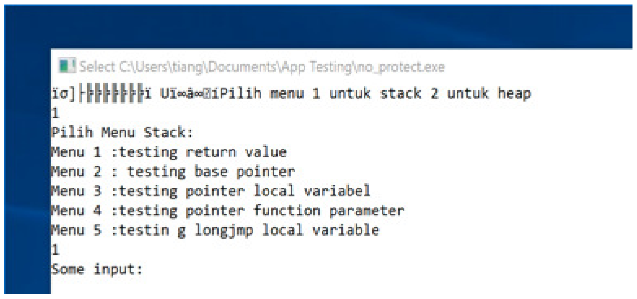
4.2. Implementation Result

At this stage contains the results of the implementation of the research in the form of applications, broadly speaking are as follows:
On this main page there are two choices, 1 for stack and 2 for heap.



```
C:\Users\tiang\Documents\App Testing\no_protect.exe
ïσ]├╫╫╫╫╫╫ï Uï∞ä∞ïPilih menu 1 untuk stack 2 untuk heap
```

Figure 5 : Application



```
Select C:\Users\tiang\Documents\App Testing\no_protect.exe
ïσ]├╫╫╫╫╫╫ï Uï∞ä∞ïPilih menu 1 untuk stack 2 untuk heap
1
Pilih Menu Stack:
Menu 1 :testing return value
Menu 2 : testing base pointer
Menu 3 :testing pointer local variabel
Menu 4 :testing pointer function parameter
Menu 5 :testin g longjmp local variable
1
Some input:
```

Figure 6 : Application

The test results are in the form of graphs and tables which illustrate the comparison of the number of attacks that can be prevented based on the attack vector test platform table.

Tabel 1: The results of the comparison are in the form of tables

Stack overflow to target	**ASLR**	**CFI**	**Without protection**
Return value	-	+	-
base pointer	-	+	-
Pointer local variabel	-	+	-
Pointer function parameter	-	+	-
longjmp buffer	N/A	N/A	N/A
Heap/BSS overflow to target			
Return value	+	+	-
base pointer	+	+	-
Pointer local variabel	-	-	-
Pointer function parameter	-	-	-
longjmp buffer	N/A	N/A	N/A

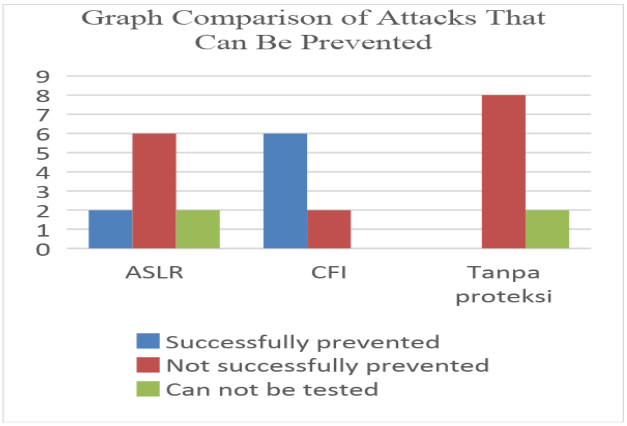


Figure 7 : Comparison of graphs

The test results in the form of graphs that were tested using visual studio illustrate the length of time the execution process was based on the protection method used

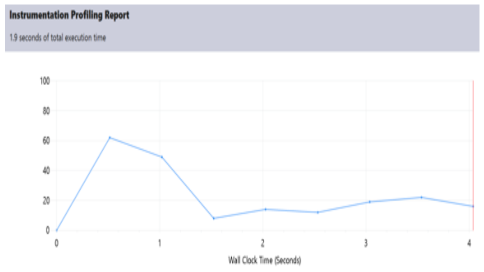


Figure 8 : execution time with ASLR

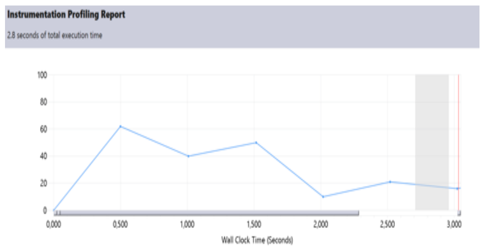


Figure 9: execution time with CFI

V. CONCLUTIONS

Based on the results of the analysis carried out, testing and the results of the research that has been carried out, the conclusions obtained are as follows:

1. The level of security method Control flow integrity (CFI) is superior in handling buffer overflow attacks on Windows 10 operating systems compared to the Address space layout randomization (ASLR) method.
2. The Address space layout randomization (ASLR) method is faster in execution time than the Control Flow Integrity (CFI) method.

VI. RECOMENDATION

In this study found several weaknesses, so that in further development it is necessary to pay attention to the following suggestions:

1. For now, the operating system variant used is only Windows 10. For further research can be made a comparison of these two methods on several operating system variants such as Linux, Windows and Solaris.
2. For further research, protection for linux and solaris based applications can be applied.

VII. REFERENCES

[1]. Andersen, S., Abella, V. 2004. *Part3:Memory Protection Technologies*. [online] Available at: https://technet.microsoft.com/en-us/library/bb457155.aspx[accessed nov 2018

[2]. Bruice, S. 2007, Applied Cryptography, Cryptography Protocols, Algorithm and Source Code in C, Second edition. *Wiley India edition*

[3]. C. Cowan, P. Wagle, C. Pu, S. Beattie, J. Walpole, 2000, *Buffer overflows: Attacks and defenses for the vulnerability of the decade 2000; In Proceedings of the DARPA Information Survivability Conference and Expo (DISCEX),* South Carolina, Hilton Head.

[4]. David Herrera Aristizabal, David Mora Rodriguez, Ricardo Yepes Guevara, 2013, *Measuring ASLR Implemetation on Modern Operating System*, Columbia, Universidad de Antioquia Medellín.

[5]. Erick Leona, Stefan D. Bruda, 2016, *Counter-measures against stack buffer overflow in GNU/Linux Operating system.*

[6]. James C. Foster, VitalyOsipov, Nish Bhalla, NielsHeinen, *"Buffer overflow attacks detect, exploit, prevent", published by Syngress Publishing, Inc.*

[7]. John Wilanderdan Mariam Kamkar, 2003, *A Comparison of Publicly Available Tools for Dynamic Buffer Overflow Prevention, Dept. Of computer and Information Science*, Linkoping Universitet.

[8]. J. Wilander, J. Wilander, M. Kamkar, *2003, A Comparison of Publicly Available Tools for Dynamic Buffer Overflow Prevention, in Proceedings of the 10th Network and*

Distributed System Security Symposium, San Diego, CA.

[9]. Mark Shaneck, 2003, *An Overview of Buffer Overflow Vulnerabilities and Internet Worms, CSCI.*

[10]. Martin Abadi, etc, 2005.*Control-Flow Integrity Principles, Implementations, and Applications,*University of California, Santa Cruz and Microsoft Research, Silicon Valley

[11]. M. Prasad,M. Prasad, Tzi-ckerChiueh, 2003, *A Binary Rewriting Defense against Stack based Buffer Overflow Attacks,* New York, State University of New York.

[12]. Nathan Burow, Scott Carr, Stefan Brunthaler, Mathias Payer, Josep Nash, Per Larsen, Michael Franz, 2016, Purdue University, SBA Research, University of California.

[13].PaxTeam, 2003, address space layout randomization (ASLR) http://pax.grsecurity.net/ docs/aslr.txt.

[14]. Takamichi Saito, Ryohei Watanabe, ShotaSugawa, Masahiro Yokoyama, 2016, *A Survey Prevention/Mitigation againt Memory Corruption Attacks*, Meiji University, Japan.

[15].Yong-Joon, Park, Gyungho Lee, 2004, *Repairing Return Address Stack for Buffer Overflow Protection,* Italy ACM