Implementation of the Function for User Program Execution Control in the Windows Environments

Jang Senng Ju

ABSTRACT

In this paper, we design a program that controls the execution permissions for the running application in the Windows system environment. It does not allow general users to execute the program converting the specific location information of the execution program, to any of the bit values with the formal structure information in window program. The converted bit value can be returned to the original bit value in the case of an authorized user, so that the original program can be normally performed. By doing so, it can be more safely used in the risk of reverse engineering for Windows executable program. We implemented the control program for the program execution authority we proposed in this paper, and the experiment was performed. At the results of experiments, it was confirmed that the control function to permit execution for the user program was working properly in the Windows environment.

Keywords : program execution rights, program execution rights access, windows system, PE modified, PE recovery

1. Introduction

Stability of the software security means that the software is intended to run only as of the developer. Software is started in the source code of abstraction, becomes a machine-recognizable binary form of executable file through the compilation process up. This executable file is run as the number of users in a variety of computing environments.

Safe software running will have difficulties that it should accurately reflect the development intention of the actual executable file depending on the complex situation and the steps. Unmodulated program flow control is an essential condition for safe operation of the program. Control of the program flow is generally in order, but If you call a function or return from the called function, it causes the movement of the control program in a specific location. At the moment of this program control movement, if program control is not protected, it will receive an direct effect on the safety of the software. Attacks such as buffer overflow attacks can be called an attack which takes place to bring the control of the program to the attacker, aiming the moment of the program control movement [1-3].

In this paper, we design a system that controls the running privileges of the user program in the Windows environment. The system is also a way to make source code conversion method difficult, through reverse engineering techniques in the Window program. The proposed contents is to design a system that does not allow authorized users to run Windows running programs.

. It modifies Windows executable program code to allow only authorized users to run Windows while running programs. This program does not run properly if you run like a normal Windows application. In order to run the program, the program should be changed to the normal operation code by driving the program that can restore the modified portion in the program. Restoring program should be allowed only authorized users to use. The proposed design in this paper was implemented and the experiments were performed.

We describe the related researches in Chapter 2. We explain design information of the executing authority control program in Chapter 3. In Chapter 4, we perform experiments and evaluation for checking whether the content is designed for normal operation. We make a conclusion in Chapter 5.

2. Related Research

Fig.1 shows the structure of the window system program for driving the window. 윈도우 운영체제는 링 구조 (Windows Ring Structure)로 되어 있다. Windows operating system is a ring structure (Windows Ring Structure).

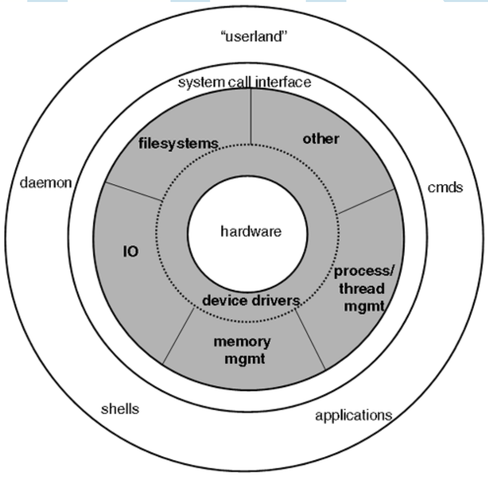


Fig. 1 Hierarchical Structure for Windows System

A hierarchical structure of a window system can be maintained at a high state when the security level of the system is met, and each layer operates independently to each of the security element. The following Figure 2 shows the internal structure of the Windows operating system.

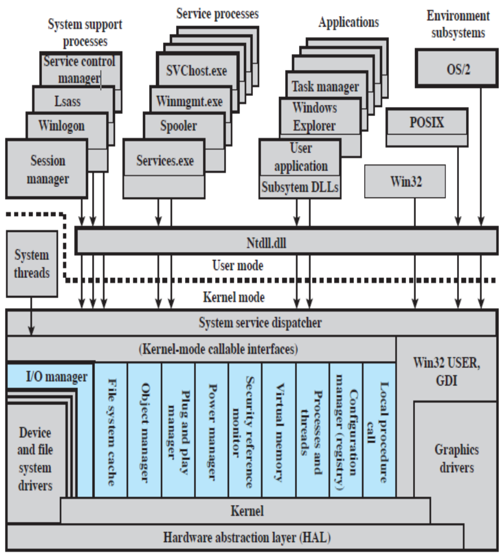


Fig. 2 Windows O.S Internal Structure

HAL (Hardware Abstraction Layer) is designed to enable seamless communication between the new attached hardware and system even when the driver is attached to the system if developers develop the driver in accordance with the standard the HAL system. The role of micro-kernel (Micro Kernel) is a job-sharing to multiple managers (Manager) and it only controls the communication with the hardware. Input and output manager controls the input and output of the system, and passes the messages between the device drivers. And it provides a passage through which applications can communicate directly with the hardware. Object manager provides information about the object, such as files, port, process, thread.

A Security Reference Manager is forced to take on the security settings of the system by permitting or denying the control of each data or system resources. A process Manager creates a thread and performs the necessary tasks. A virtual memory manager allocates the main memory unit according to the request from the application, and undertakes the main memory control.

Win32 subsystem, a window of the basic subsystems, allows 32-bit applications to run. And, it provides a user interface of the basic windows. POSIX (Portable Operating System Interface) is responsible for the relevant POSIX interface as a standard set of operating system interface that is based on the UNIX operating system. Security Subsystem is a service system created to protect data when a user logs in and ensure the operating system to be able to control them.

The execution control-related research in the Windows system environment is not so much. A protection scheme for a Windows executable program uses the encryption to the PE header. The user who cannot decrypt is not allowed to execute, by encrypting the PE header. In addition, as a way to prevent reverse engineering and blocking access to Windows executable program, the packing (packing) technique is quite frequently used for Windows executable program [6-8].

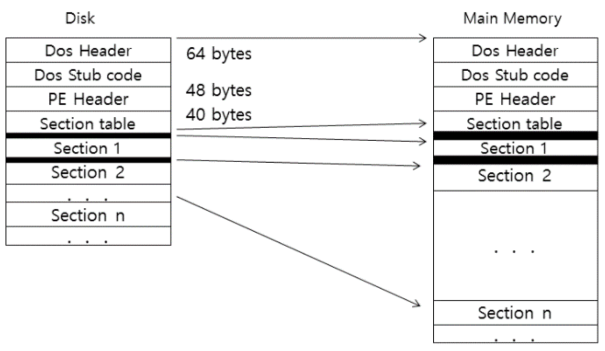
Packing technique is generally used for the purpose of reducing the file size by using the technique of compression for Windows PE header and making reverse engineering analysis difficult.

Furthermore, code obfuscation has an important role in the copyright protection by making others difficult to understand the code or program, without changing the functionality of the computer program for the purpose of making analysis difficult. In addition, it also provides code obfuscation tool that obfuscated source code and binary executables [9-12].

In the case of programs applied to this obfuscation technique, it is difficult to analyze than the existing obfuscation techniques.

3. Design of execute permissions control

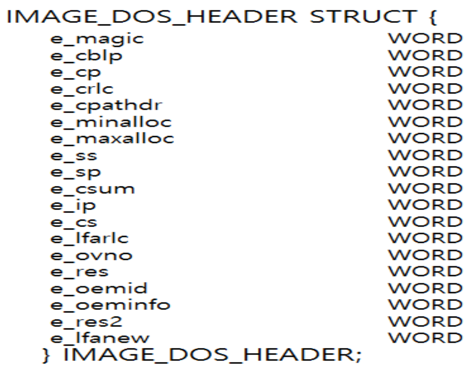
In this paper, we design a function that can prohibit for the unlicensed user to execute the program for executable files that run on Windows. Content designed in this paper is targeted at programs that run on Windows. In order to design such environment, analyzing the structure of the window program should be preceded.



Structure of the executable program that runs on the Windows system is shown in Figure 3.

Fig. 3 Windows Execution Program Structure

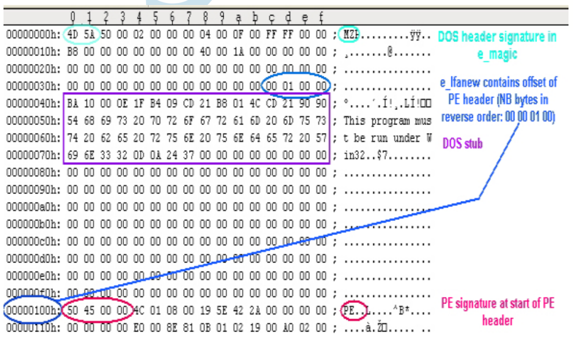
The structure of operating system that runs on the Windows is shown in Figure 2 above. The DOS header structure is shown in the following figure 4.



```
IMAGE_DOS_HEADER STRUCT {
    e_magic                 WORD
    e_cblp                  WORD
    e_cp                    WORD
    e_crlc                  WORD
    e_cpathdr               WORD
    e_minalloc              WORD
    e_maxalloc              WORD
    e_ss                    WORD
    e_sp                    WORD
    e_csum                  WORD
    e_ip                    WORD
    e_cs                    WORD
    e_lfarlc                WORD
    e_ovno                  WORD
    e_res                   WORD
    e_oemid                 WORD
    e_oeminfo               WORD
    e_res2                  WORD
    e_lfanew                WORD
  } IMAGE_DOS_HEADER;
```

Fig. 4. DOS Header Structure

Data structure of DOS header is the same as above in Figure 4. Here, the MZ string is usually involved in e_magic. This indicates the start of the DOS header. And, finally, there is lfanew field. This represents the end of the DOS header. And it has start address of the PE header. And, it has the value 00 01 00 00 at the end. This is reversed as address 00 00 01 00. This value is a start address of the PE header. Figure 5 shows the structure of the PE header.



```
            0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 4D 5A 50 00 02 00 00 00 04 00 0F 00 FF FF 00 00 ; MZP.........ÿÿ..
00000010h: B8 00 00 00 00 00 00 00 40 00 1A 00 00 00 00 00 ; ¸.......@.......
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 ; ................
00000040h: BA 10 00 0E 1F B4 09 CD 21 B8 01 4C CD 21 90 90 ; º....´.Í!¸.LÍ!□□
00000050h: 54 68 69 73 20 70 72 6F 67 72 61 6D 20 6D 75 73 ; This program mus
00000060h: 74 20 62 65 20 72 75 6E 20 75 6E 64 65 72 20 57 ; t be run under W
00000070h: 69 6E 33 32 0D 0A 24 37 00 00 00 00 00 00 00 00 ; in32..$7........
00000080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000a0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000b0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000c0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000d0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000e0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000f0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000100h: 50 45 00 00 4C 01 08 00 19 5E 42 2A 00 00 00 00 ; PE..L....^B*....
00000110h: 00 00 00 00 E0 00 8E 81 0B 01 02 19 00 A0 02 00 ; ....à.Ž□..... ..
```

DOS header signature in e_magic
e_lfanew contains offset of PE header (NB bytes in reverse order: 00 00 01 00)
DOS stub
PE signature at start of PE header

Fig. 5 PE Header Structure

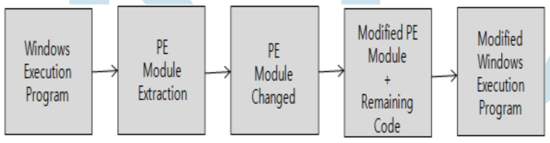
PE header on Figure 5 above is a data structure that stores critical data. DOS Header, DOS Stub code, PE Header, Section Table consists of DOS Header, DOS Stub code, a PE Header, Section Table + more than 1 section.

DOS Header has a fixed size of 64bytes, and is located in the first part of the disc and the memory ImageBase. DOS Stub Code can be found in the place aparting as much as the size of DOS Header, 64bytes from the starting point (ImageBase on the first part of the memory or file). It runs on DOS mode and is not currently in use.

PE Header 구조체는 다음과 같다. PE Header structure is as follows.

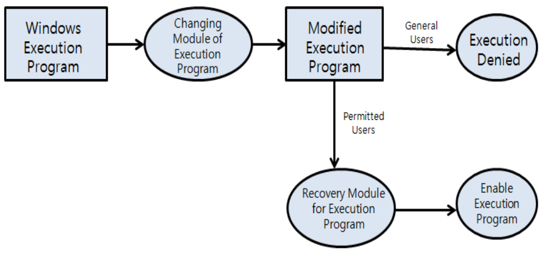
```
typedef struct _IMAGE_NT_HEADERS {
        DWORD Signature;
        IMAGE_FILE_HEADER FileHeader;
        IMAGE_OPTIONAL_HEADER OptionalHeader;
} IMAGE_NT_HEADERS, *PIMAGE_NT_HEADERS;
```

PE Header structure is made up of Signature, FIleHeader, OptionalHeader, each contains a basic attribute information of the program. File Header and OptionalHeader each consists of IMAGE_FILE_HEADER, IMAGE_OPTIONAL_HEADER structure.



Windows Execution Program
PE Module Extraction
PE Module Changed
Modified PE Module + Remaining Code
Modified Windows Execution Program

On the basis of the structure of a window executable program mentioned from the top, the structure of executable permissions control program on a Windows program implemented in this paper are as follows: Figure 6.



Windows Execution Program
Changing Module of Execution Program
Modified Execution Program
General Users
Execution Denied
Permitted Users
Recovery Module for Execution Program
Enable Execution Program

Fig. 6 Execution Program Control Structure for Windows Program

As shown in Figure 6 above, it is the structure for interrupting the execution of the program, unless the user has permissions for programs that run on Windows. In order to ensure that only authorized users run the Windows executable program, it should modify the code so that not to run in a Windows program. This program does not run properly if you run like a normal Windows program. To run the program, it should be changed to the normal operation code by driving a program that can be restored to the original with respect to the modified portion in the program. Restore program allows only authorized users to use. The restoration module to be modified for executing the program can be called a kind of key. You must have this key to access to the executable program.

The following Figure 7 shows the process of modification to allow only certain users to use the actual Windows executable program code.

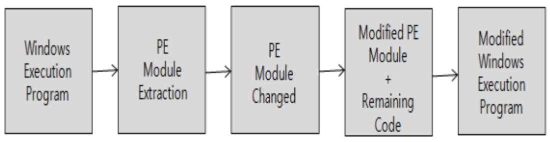

Windows Execution Program
PE Module Extraction
PE Module Changed
Modified PE Module + Remaining Code
Modified Windows Execution Program

Fig. 7 PE Changing Procedure for Windows Program

Figure 7 above shows the procedure to change the Windows executable program. First, it reads the Windows executable program. It extracts a portion of the PE header of loaded Windows executable program. It changes specific or arbitrary data area in the header of PE to the specified string.

By this change, a Windows PE header cannot be recognized by the Windows system environment. The program is no longer able to run on the window system because header has been modified. That is, if the header cannot recognize a Windows PE, the program cannot be run. It will save modified PE header and the rest of the code as overwritten on the original Windows executable program. By doing so, any users will not be able to run this program, except users with Windows PE header information before the change.

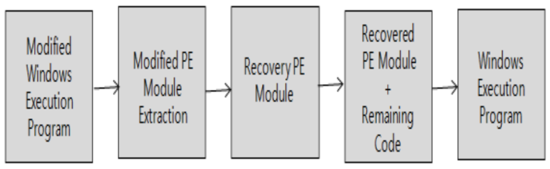

Modified Windows Execution Program
Modified PE Module Extraction
Recovery PE Module
Recovered PE Module + Remaining Code
Windows Execution Program

Fig. 8 PE Recovery Procedure for the Modified Windows Program

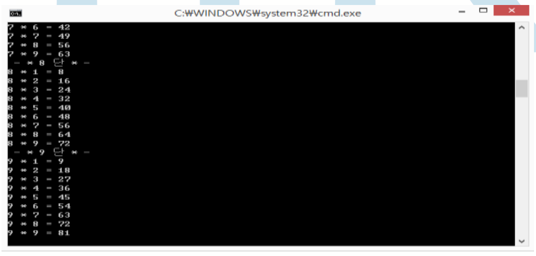
Figure 8 shows a process of recovering a modified PE header of the program window in the Figure 7 process.

Users with permission to restore the PE header for the modified Windows executable programs are able to run the recovery program. First, it reads the PE header of the modified program. The modified data in the modified PE header are thereby reduced to the original PE header data. The PE header recovered as the original data and the remaining part of the program will be overwritten on the executable program file. This stored executable program is to be brought back to the original program.

4. Experiment and evaluation

The proposed programs designed for Windows executable permissions control information is implemented and the experiment was performed as to whether or not the normal operation. Experiments were mainly performed in the Windows program environment that runs on the Windows 7 operating system. To experiment, Microsoft Visual Studio 2010 was used for the window compiling environment.

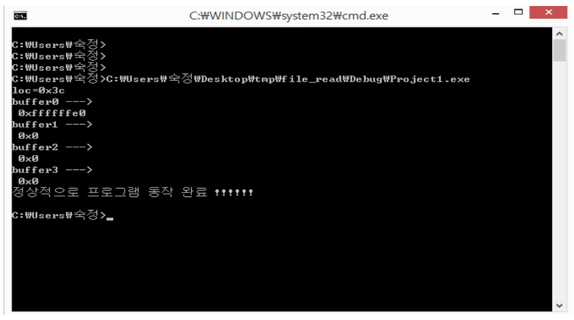
The user program for use in Windows environments for experiments is a program that calculates the multiplication tables. Performing result of the multiplication table calculation program is shown in the following Figure 9.



```
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
 -- * 8 단 * --
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
 -- * 9 단 * --
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

Fig. 9 Multiplication Table Program

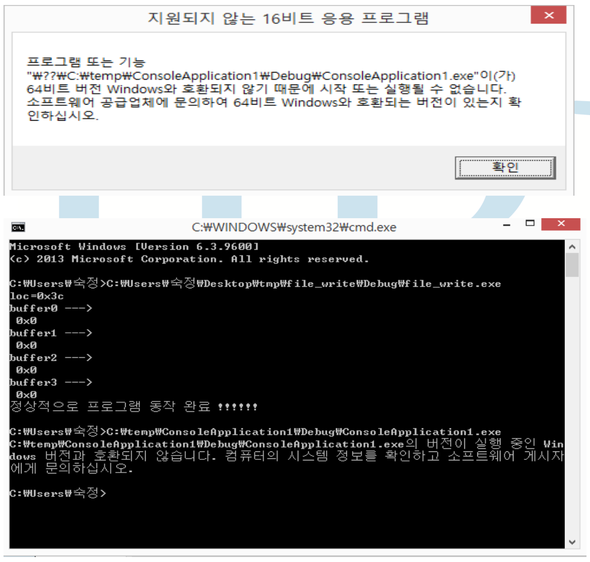
For the above multiplication program, the process that prevent unauthorized users to execute the program is as follows. In order to prevent an unauthorized user from accessing to the above multiplication program on Figure 9, it drives the proposed programs that can modify some code in the executable file of multiplication program. Operation screen when driving a launcher program modifications proposed in this paper is as shown in Figure 10.



```
C:₩WINDOWS₩system32₩cmd.exe

C:₩Users₩숙정>
C:₩Users₩숙정>
C:₩Users₩숙정>
C:₩Users₩숙정>C:₩Users₩숙정₩Desktop₩tmp₩file_read₩Debug₩Project1.exe
loc=0x3c
buffer0 --->
 0xffffffe0
buffer1 --->
 0x0
buffer2 --->
 0x0
buffer3 --->
 0x0
정상적으로 프로그램 동작 완료 !!!!!!

C:₩Users₩숙정>_
```

Fig. 10 Modifying Procedure for Multiplication Table Program

As Figure 10, after transforming the multiplication program, this program will not operate in the normal form when the general users run the program. When modified multiplication program run as a normal user privileges, an error is generated as shown in next Figure 11.

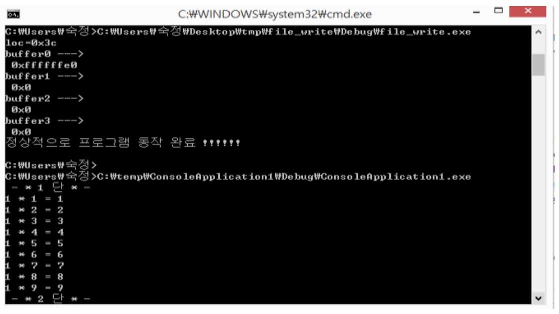


```
지원되지 않는 16비트 응용 프로그램

프로그램 또는 기능
"₩??₩C:₩temp₩ConsoleApplication1₩Debug₩ConsoleApplication1.exe"이(가)
64비트 버전 Windows와 호환되지 않기 때문에 시작 또는 실행될 수 없습니다.
소프트웨어 공급업체에 문의하여 64비트 Windows와 호환되는 버전이 있는지 확
인하십시오.

확인
```

```
C:₩WINDOWS₩system32₩cmd.exe

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:₩Users₩숙정>C:₩Users₩숙정₩Desktop₩tmp₩file_write₩Debug₩file_write.exe
loc=0x3c
buffer0 --->
 0x0
buffer1 --->
 0x0
buffer2 --->
 0x0
buffer3 --->
 0x0
정상적으로 프로그램 동작 완료 !!!!!!

C:₩Users₩숙정>C:₩temp₩ConsoleApplication1₩Debug₩ConsoleApplication1.exe
C:₩temp₩ConsoleApplication1₩Debug₩ConsoleApplication1.exe의 버전이 실행 중인 Win
dows 버전과 호환되지 않습니다. 컴퓨터의 시스템 정보를 확인하고 소프트웨어 게시자
에게 문의하십시오.

C:₩Users₩숙정>
```

Fig. 11. Program Error Screen for Executing Modifying Multiplication Table Program

A user with permission to run the modified multiplication program is possible to restore it to the normal operation programs, using the program which can revert the modified multiplication table program normally. Next "file_write.exe" program is a program for this. Running the program can recover it to a multiplication table program before modification. Figure 12 below shows this process.


C:₩WINDOWS₩system32₩cmd.exe

```
C:₩Users₩숙정>C:₩Users₩숙정₩Desktop₩tmp₩file_write₩Debug₩file_write.exe
loc=0x3c
buffer0 --->
 0xffffffe0
buffer1 --->
 0x0
buffer2 --->
 0x0
buffer3 --->
 0x0
정상적으로 프로그램 동작 완료 !!!!!!

C:₩Users₩숙정>
C:₩Users₩숙정>C:₩temp₩ConsoleApplication1₩Debug₩ConsoleApplication1.exe
 - * 1 단 * -
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
 - * 2 단 * -
```

Fig. 12 Access Permitted User Execution Procedure for the Modifying Multiplication Table Program

Figure 12 above shows the process executed by privileged users that can use the modified multiplication program. A user with permissions to execute the multiplication tables can restore the modified program with "file_write.exe" program which can restore the normal multiplication table. After Multiplication tables is recovered to a normal executable file using a "file_write.exe" program, the original "ConsoleApplication1.exe" file is generated. When running the "ConsoleApplication1.exe" will show the normal multiplication calculation results as shown in Figure 12. It may determine that it cannot execute the program of the multiplication table except for authorized users. Therefore, the structure presented in this paper can be seen that accurate control of the program execution authority.

5. Conclusion

In this paper, we design a control program that allows you to control the execution permissions for the application running on the Windows system environment. Normal users should not allow to run the program through converting specific location information of the executor to an arbitrary bit value, using the formal structure of the window program information. Only when authorized users revert thus converted bit value to the original value of the bit, the program can be properly carried out.

Execution control program designed for Windows executable program proposed in this paper is designed to be used only in an authorized user to run the program. In order to ensure that only authorized users run the Windows executable program, it should modify the code so that not to run in a Windows program. The program does not run properly if you run like a normal Windows application. To run the program, it should be changed to the normal operation code by driving a program that can be restored to the original with respect to the modified portion in the program. Restore program allows only authorized users to use.

It was examined with respect to whether or not the normal operation control over the design features for Windows executable program proposed in this paper. Results of the experiment confirmed that normal users cannot execute the modified executable program. And in the case of an authorized user, it could be confirmed that the normal operation is possible by recovering the original PE header, using a program that can recover the modified PE header information.

REFERENCES

[1] B.M.Cho, J.W. Nho, H.S.Oh, et. al. "Implement and test data obfuscation tool for C ++ language", *KIISE Proceedings*, pp. 292-294. 2006.

[2] J.H. Lee, "Data structures reverse engineering techniques for the executable file fragmentation", *Journal of Information Security and Cryptology*, vol. 22. no. 3, pp. 615-619. 2012.

[3] Korea Copyright Commission, "SW reverse analysis and technical protection", *Korea Copyright Commission*, 2009.

[4] S.H.Oh, et. al., "Study on the compatibility of the virtual machine for the reverse engineering", *Journal of Korea Society of Automotive Engineers*, vol. 22. no. 6, 31-38, 2014.9.

[5] C. H. Lee, et. al., "Anti-reverse engineering techniques for Android applications", *Journal of Security Engineering Research Institute*, vol. 10. no. 1, 2013. 2.

[6] J.H.Kim, et. al., "Robust anti-reverse engineering techniques for using the AES algorithm to protect Android applications", *Journal of Information Science*, vol. 42. no. 12, pp. 1611-1622. 2015.12.

[7] K.H.Kang, et. al., "Static Code Analysis with Open Source based tool chain", *Journal of Information Science*, vol. 21. no. 2, pp. 148-153. 2015.2.

[8] S.J.Cho, et.al., "Software copyright protection technology trends", *Korea Society of Information Technology*, vol. 11. no. 2, pp. 23-32, 2013.12.

[9] Yuxue Piao, et.al., "Pro guard obfuscation tool structure and function analysis", *Journal of Korea Institute of Communication Sciences*, vol. 38. no. 8, pp. 654-662, 2013.8.

[10] C.S.Chun, et.al., "Dynamic analysis of executable obfuscation techniques to virtualization", *Journal of Information Science*, vol. 40. no. 1, pp. 61-71, 2013.1.

[11] S.H.Lee, et.al., "The original structure and meaning extraction method of dynamic virtual machines running file-based obfuscation", *Journal of Information Science*, vol. 41 no. 10, pp. 859-869. 2014.10.

[12] S.K.Jung, et.al., "protection method of applications and licenses through the executable file encryption portion", *Journal of Information Science Proceedings*, vol. 39. no. 2C, pp. 157-159, 2012.11.