리버싱 이 정도는 알아야지

Chapter 01. 무작정 분석해보기

http://securityfactory.tistory.com

목 차

Chapter 01

무작정 분석해보기

SECURITY FACTORY

1. 실행파일 동작 확인

이게 뭐냐? “HelloWorld”와 같습니다. 아직은 잘 모르지만 ‘분석이 이런 식으로 이루어지는구나.’를 한번 느껴보는 겁니다. SecurityFactory 폴더 안에는 Sample 01.exe 파일이 있습니다. (Sample 01.exe 위치는 어디든 상관없습니다.)

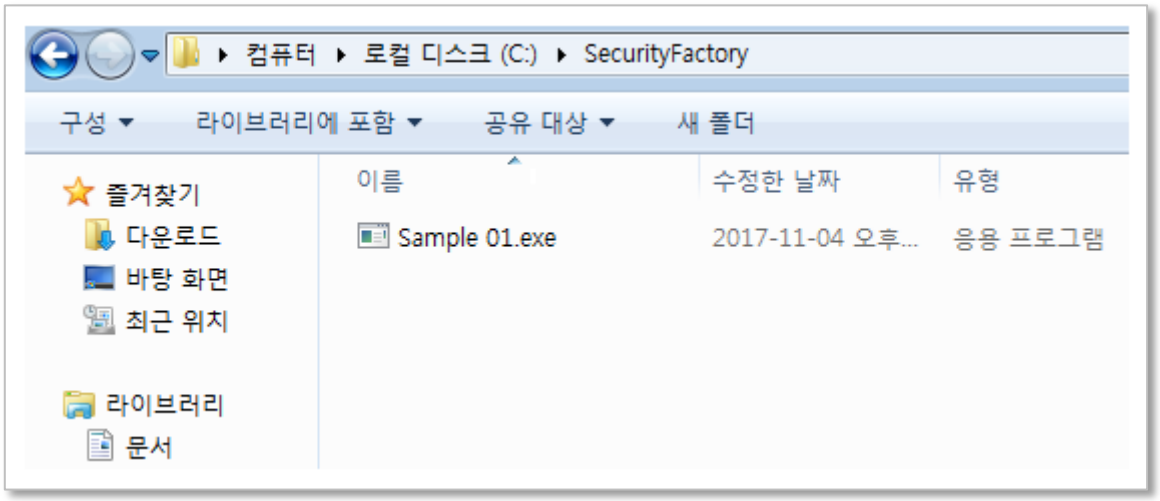


그림 1-1 Sample 01.exe

분석을 시작하기 전에 파일을 실행시켜보는 것이 좋습니다. 분석의 목적과 방향을 설정할 수 있고, 그 외 분석에 필요한 다양한 정보들을 얻을 수 있죠. 따로 분석이 필요 없을 정도로 많은 정보를 얻게 되는 경우도 많습니다. (우리의 목표는 분석을 따로 하지 않아도 동작만 보고 내부 흐름을 그릴 수 있는 실력을 쌓는 것입니다.)

한번 실행시켜보겠습니다. Sample 01.exe 는 비프음 소리와 함께 메시지 박스를 출력합니다.

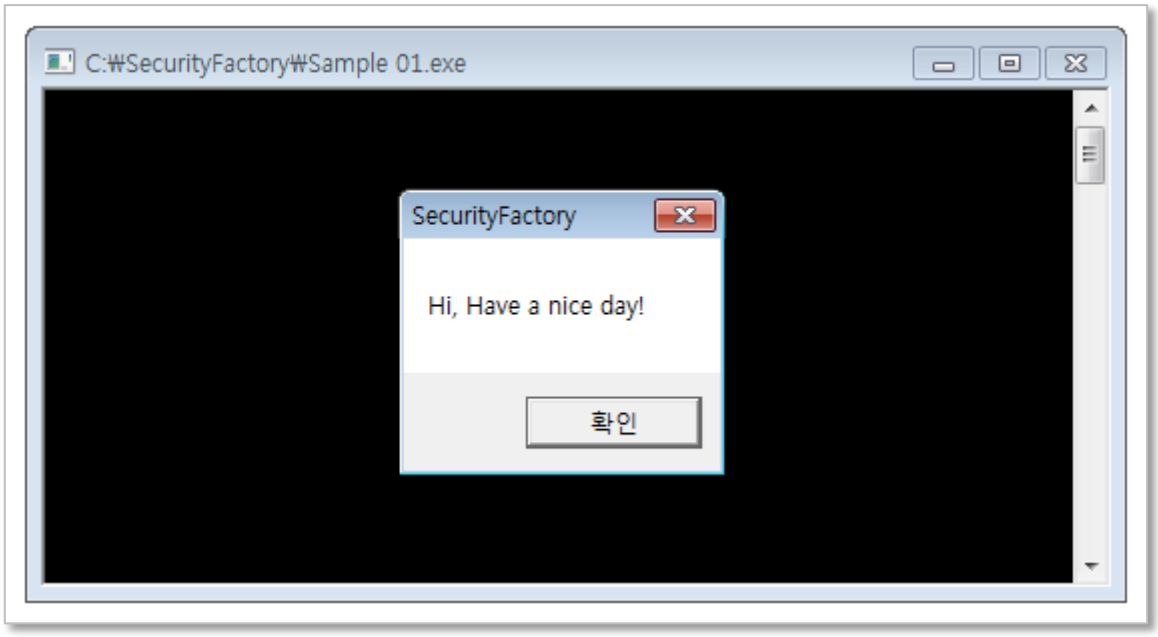


그림 1-2 Sample 01.exe 실행

여기서 우리는 'Sample 01.exe가 잘 실행된다.'는 중요한 정보를 얻었습니다. 이 말은 Sample 01.exe가 32 비트에서 동작하는 실행 파일이고, AntiVM 기법이 적용되어 있지 않다는 말과 같습니다. 그리고 '비프음이 어떻게 발생하고, 메시지 박스는 어떻게 출력되는지 확인해야 한다.'는 분석 목표도 생겼습니다. 이제 우리는 확인만 하면 됩니다.

2. 코드분석_Level.1 | 흐름 파악하기

'나무를 보지 말고 숲을 보라.'라는 말이 있습니다. 이는 보안 분석에 있어서도 아주 중요합니다. 어셈블리 코드를 하나하나 분석하고, 각각의 기능을 파악하는 것만큼 전체적인 흐름도 잘 파악해야 하죠. 그런데 막상 코드를 분석하다 보면 전체적인 흐름을 그리기가 쉽지 않습니다. 한번에 이해하기 힘들만큼 복잡한 코드를 접하기도 하고, 단일 모듈에 집중하다 보면 전체적인 흐름을 잃기도 합니다. 시간이 흘러서 기억하지 못하는 경우도 종종 있습니다. 그래서 지금 우리에겐 연습과 훈련이 필요합니다. 답을 도출하는데 급급하기보다 다양한 가능성을 유추하고 시도하면서 정리하고 기록하는 습관을 키우시기 바랍니다.

본 문서에서는 먼저 전체적인 흐름을 파악하고, 세부적인 분석에 들어가겠습니다. 분석 작업은 어렵고 복

잡하기 때문에 무작정 달려들기보다 도움이 되지 않을까 싶습니다.

[그림 2-1]은 Sample 01.exe 의 전체적인 흐름도입니다.

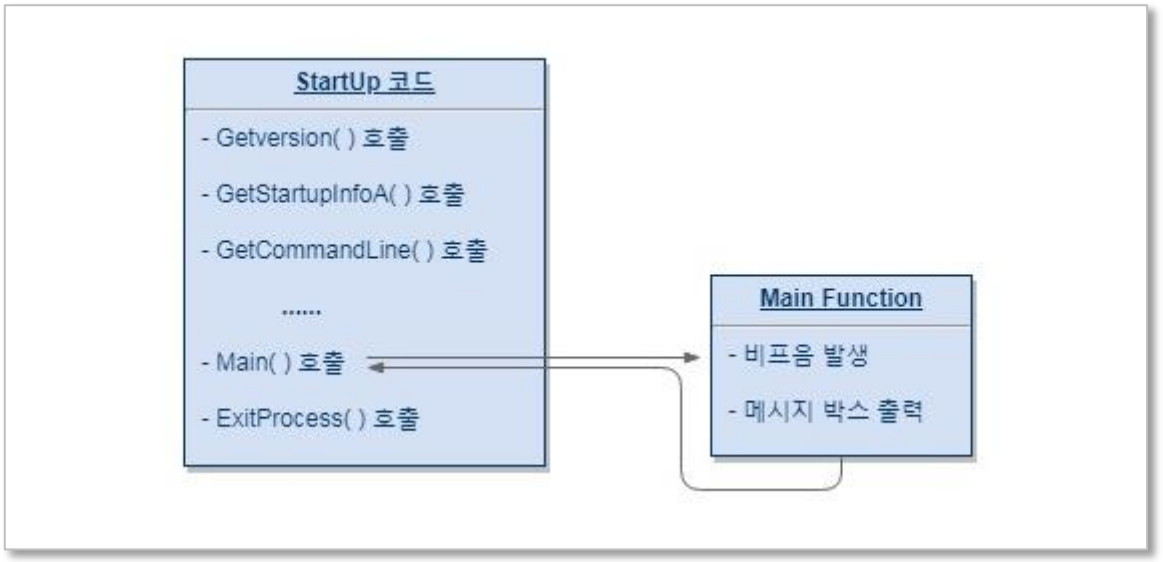


그림 2-1 Sample 01.exe 동작 흐름도

Sample 01.exe 를 OllyDBG.exe 에 올려봅시다. 뭔가 코드가 빠르게 실행되는 것처럼 보이다가 0x00401030 주소에서 멈추죠? 그리고 그 아래 코드들이 보이기 시작할 겁니다. 여기가 StartUp 코드입니다. StartUp 코드는 컴파일러가 프로그램을 만들 때 집어넣는 코드로, main( ) 함수 앞 단에 위치합니다. 그렇기 때문에 StartUp 코드는 빨리 건너뛰고 main( )을 찾아갈 줄 알아야 합니다. 자세한 내용은 '3.4. StartUp 코드 이해하기' 에서 살펴보기로 하겠습니다.

```
00401030  PUSH    EBP
00401031  MOV     EBP, ESP
00401033  PUSH    -1
00401035  PUSH    004050A8
0040103A  PUSH    00401C8C                        SE handler installa
0040103F  MOV     EAX, DWORD PTR FS:[0]
00401045  PUSH    EAX                             kernel32.BaseThread
00401046  MOV     DWORD PTR FS:[0], ESP
0040104D  SUB     ESP, 10
00401050  PUSH    EBX
00401051  PUSH    ESI
00401052  PUSH    EDI
00401053  MOV     [LOCAL.6], ESP
00401056  CALL    DWORD PTR DS:[<&KERNEL32.GetVer  kernel32.GetVersion
```

그림 2-2 StartUp 코드 시작지점

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

3. 코드분석_Level.2 | API 호출 분석

Windows 운영체제는 비프음을 내고, 메시지 박스를 출력하는 것과 같은 이벤트를 Win32 API로 전달 받아서 처리합니다. 이 말은 비프음을 내고 싶으면, API를 사용해서 운영체제에게 요청해야 한다는 겁니다. '2. 코드분석_Level.1 | 흐름 파악하기'에서 Beep( )와 MesssageBoxA( ) 호출이 비프음을 발생하고, 메시지 박스를 출력시켜달라고 요청하는 행위입니다.

사용자 프로세스에서 발생하는 이벤트는 API 호출로 이루어지기 때문에 잘 알아야 합니다. 실제 분석에서도 큰 틀은 API를 중심으로 하다가, 세세한 분석이 필요할 때 어셈블리 코드를 살펴봅니다. 그런데 우리가 생각하는 이상으로 API는 수가 많고, 개개의 API가 가지는 정보들이 다양합니다. 이걸 모두 암기하는 건 미련할 뿐 아니라 불가능하다고 봐야 하죠. 그래서 Microsoft에서는 MSDN이라는 Win32 API 사전을 만들어 놓았습니다.

※ 참고
MSDN은 Microsoft에서 응용 프로그램 개발자들을 위해 Windows에 관한 기술 문서 등을 제공하는 서비스입니다. 그렇기 때문에 Win32 API 외에도 다양한 정보들이 포함되어 있습니다. 그러나 우리는 API에 대한 정보를 찾아보는 용도로 활용할 것입니다.

※ Note

다음은 MSDN에 정의되어 있는 Beep( ) API에 대한 정보입니다.

Beep function

Generates simple tones on the speaker. The function is synchronous; it performs an alertable wait and does not return control to its caller until the sound finishes.

Syntax

```
BOOL WINAPI Beep(
       _In_   DWORD  dwFreq,
       _In_   DWORD  dwDuration
);
```

Parameters

dwFreq [in]

The frequency of the sound, in hertz. This parameter must be in the range 37 through 32,767 (0x25 through 0x7FFF).

dwDuration [in]

The duration of the sound, in milliseconds.

Return value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call GetLastError.

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

MessageBoxA function

Displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information. The message box returns an integer value that indicates which button the user clicked.

Syntax

```
int MessageBoxA(
       _In_    HWND    hWnd,
       _In_    LPCSTR lpText,
       _In_    LPCSTR lpCaption,
       _In_    UINT   uType
);
```

Parameters

hWnd [in]

A handle to the owner window of the message box to be created. If this parameter is NULL, the message box has no owner window.

lpText [in]

The message to be displayed. If the string consists of more than one line, you can separate the lines using a carriage return and/or linefeed character between each line.

lpCaption [in]

The dialog box title. If this parameter is NULL, the default title is Error.

uType [in]

The contents and behavior of the dialog box. This parameter can be a combination of flags from the following groups of flags.

Return value

If a message box has a Cancel button, the function returns the IDCANCEL value if either the ESC key is pressed or the Cancel button is selected. If the message box has no Cancel button, pressing ESC has no effect.

If the function fails, the return value is zero. To get extended error information, call GetLastError.

If the function succeeds, the return value is one of the following menu-item values.

Value	Meaning
MB_OK 0x00000000L	The message box contains one push button: OK. This is the default.
MB_APPLMODAL 0x00000000L	The user must respond to the message box before continuing work in the window identified by the hWnd parameter. However, the user can move to the windows of other threads and work in those windows. Depending on the hierarchy of windows in the application, the user may be able to move to other windows within the thread. All child windows of the parent of the message box are automatically disabled, but pop-up windows are not. MB_APPLMODAL is the default if neither MB_SYSTEMMODAL nor MB_TASKMODAL is specified.

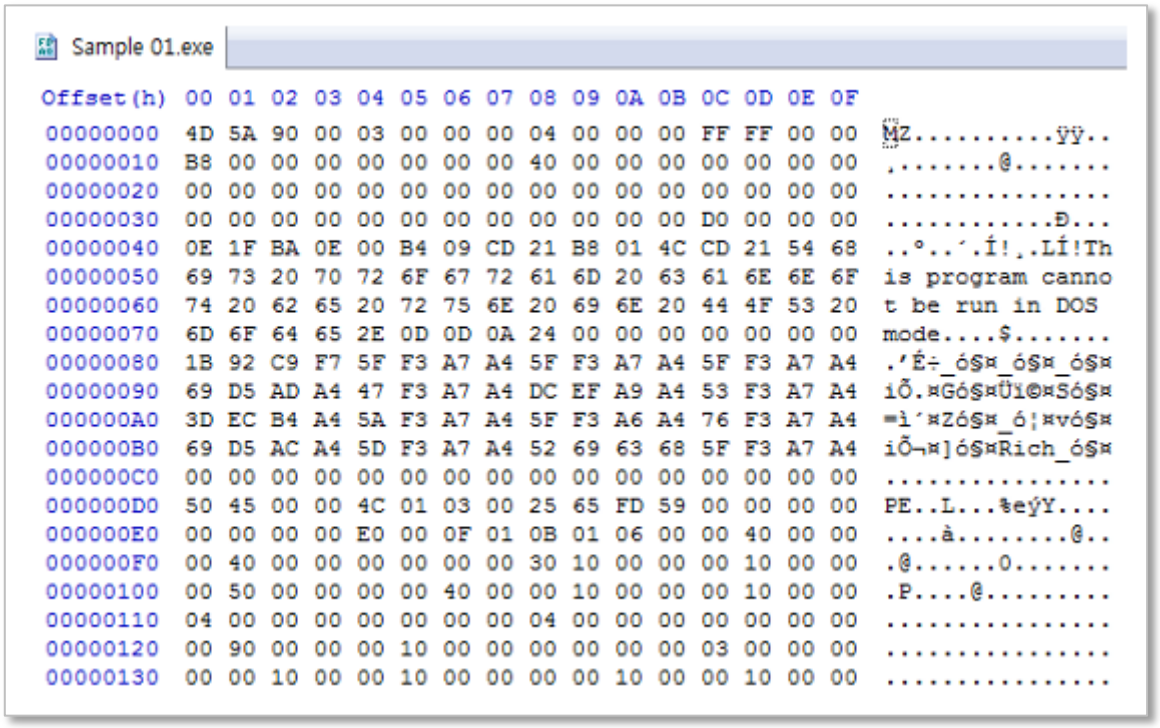
표 3-1 MessageBoxA( ) API의 네 번째 인자로 들어가는 버튼 타입 정보

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

4. 코드분석_Level.3 | 파고들기

실제 분석에서 잘 활용되진 않지만 기본적으로 알아야 할 부분에 대해 확인해보는 시간을 갖도록 하겠습니다. Hxd.exe 를 사용해서 Sample 01.exe 를 열어보면 파일 데이터를 확인할 수 있습니다. (Hxd.exe 는 파일 데이터를 16 진수 값으로 출력해주는 도구입니다.)



Sample 01.exe

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....ÿÿ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	¸.......@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	D0	00	00	00	.....Ð...
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..º..´.Í!¸.LÍ!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....$.....
00000080	1B	92	C9	F7	5F	F3	A7	A4	5F	F3	A7	A4	5F	F3	A7	A4	.'É÷_ó§¤_ó§¤_ó§¤
00000090	69	D5	AD	A4	47	F3	A7	A4	DC	EF	A9	A4	53	F3	A7	A4	iÕ.¤Gó§¤Üï©¤Só§¤
000000A0	3D	EC	B4	A4	5A	F3	A7	A4	5F	F3	A6	A4	76	F3	A7	A4	=ì´¤Zó§¤_ó¦¤vó§¤
000000B0	69	D5	AC	A4	5D	F3	A7	A4	52	69	63	68	5F	F3	A7	A4	iÕ¬¤]ó§¤Rich_ó§¤
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000D0	50	45	00	00	4C	01	03	00	25	65	FD	59	00	00	00	00	PE..L...%eýY....
000000E0	00	00	00	00	E0	00	0F	01	0B	01	06	00	00	40	00	00	....à.....@..
000000F0	00	40	00	00	00	00	00	00	30	10	00	00	00	10	00	00	.@......0.....
00000100	00	50	00	00	00	00	40	00	00	10	00	00	00	10	00	00	.P....@.....
00000110	04	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00	.....
00000120	00	90	00	00	00	10	00	00	00	00	00	00	03	00	00	00	.....
00000130	00	00	10	00	00	10	00	00	00	00	10	00	00	10	00	00	.....

그림 4-1 Sample 01.exe 파일 데이터

실행 파일의 데이터는 Microsoft 에서 정한 규칙에 맞게 기록이 되어 있습니다. 이것을 PE File Format 이라고 합니다. PE File Format 은 'PE 파일이 어떻게 구성되어야 하는지에 대한 규칙'입니다. 크게 PE 헤더, text 섹션, data 섹션으로 나눌 수 있습니다.

그림 4-2 PE File Format에 대한 개략적인 설명

PE 헤더는 파일을 실행시키기 위해 필요한 정보들이 기록되어 있는 영역입니다. 실행 파일이 맞는지, 실행될 때 파일 데이터가 메모리의 어느 위치에 올라가야 하는 지와 같은 정보가 기록되어 있습니다. text 섹션은 파일이 동작하는데 필요한 코드가 기록되어 있는 영역입니다. Sample 01.exe에서 Beep( )와 MessageBoxA( ) 호출 코드가 text 섹션에 기록되어 있습니다. data 섹션은 코드가 실행될 때 필요한 부가적인 정보가 기록되어 있는 영역입니다. MessageBoxA( ) API를 호출할 때 인자로 들어가는 "SecurityFactory", "Hi, Have a nice day!" 문자열 정보가 data 섹션에 기록되어 있습니다.

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

API 호출에 대해서도 간략하게 설명하겠습니다. 메모리 맵을 보면 Sample 01.exe 외에도 수 많은 파일 데이터들이 로드되어 있습니다.

Address	Size	Owner	Section	Contains	Type	Access	Initial
003D0000	00003000				Priv	RW	RW
00400000	00001000	Sample_0		PE header	Imag	R	RWE
00401000	00004000	Sample_0	.text	code	Imag	R	RWE
00405000	00001000	Sample_0	.rdata	imports	Imag	R	RWE
00406000	00003000	Sample_0	.data	data	Imag	R	RWE
00410000	00101000				Map	R	R
00520000	00054000				Map	R	R
01160000	00002000				Priv	RW	RW
75C10000	00001000	KERNELBA		PE header	Imag	R	RWE
75C11000	00043000	KERNELBA	.text	code,imports	Imag	R	RWE
75C54000	00002000	KERNELBA	.data	data	Imag	R	RWE
75C56000	00001000	KERNELBA	.rsrc	resources	Imag	R	RWE
75C57000	00003000	KERNELBA	.reloc	relocations	Imag	R	RWE
76AD0000	00001000	USP10		PE header	Imag	R	RWE
76AD1000	0005B000	USP10	.text	code,imports	Imag	R	RWE
76B2C000	00002000	USP10	.data	data	Imag	R	RWE
76B2E000	0002A000	USP10	Shared		Imag	R	RWE
76B58000	00012000	USP10	.rsrc	resources	Imag	R	RWE
76B6A000	00003000	USP10	.reloc	relocations	Imag	R	RWE
76C70000	00001000	kernel32		PE header	Imag	R	RWE
76C71000	000C5000	kernel32	.text	code,imports	Imag	R	RWE
76D36000	00001000	kernel32	.data	data	Imag	R	RWE
76D37000	00001000	kernel32	.rsrc	resources	Imag	R	RWE
76D38000	0000C000	kernel32	.reloc	relocations	Imag	R	RWE

그림 4-12 Sample 01.exe 메모리 맵

이를 동적 연결 라이브러리 또는 DLL 파일이라고 합니다. “DLL 파일은 또 뭐야?”라는 생각이 들 수 있습니다. 쉽게 생각해서 EXE 파일의 비서라고 보면 됩니다. Windows 운영체제에서는 EXE 파일의 편의를 위해서 DLL 이라는 비서를 제공하고 있습니다. EXE 파일이 모든 동작을 수행할 필요 없이 DLL에게 시키면 특정 행위가 발생하게 되는 것입니다. 여기서 비서에게 요청하는 동작이 바로 API 호출입니다.

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

5. 코드구현_Level.4

분석의 마지막은 코드 구현입니다. 내가 분석한 정보를 가지고 그대로 구현해보는 겁니다. 이만큼 확실한 마무리도 없을 겁니다.

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

리버싱 이 정도는 알아야지

발행일 | 2018 년 09 월
발행자 | SecurityFactory
페북 주소 | http://bitly.kr/OrHQ
이메일 | itseeyou@naver.com


SECURITY FACTORY
http://securityfactory.tistory.com