Modify Thread Local Storage(TLS) Directory For Anti-Debugging

Author: TaeKwan Yang (ytk2128@gmail.com)

2020-04-17

Draft: 2017-03-05

Abstract

TLS(Thread Local Storage) Callback 함수는 Main Thread 가 생성되기 전에 실행되므로 EntryPoint 의 코드보다 먼저 실행되는 특징이 있어 Anti-Debugging 기법으로 많이 사용된다.

본 문서는 TLS Callback 함수를 정의하여 Anti-Debugging 을 구현하는 게 아닌 PE32 파일의 TLS Callback Table 을 변조하여 Anti-Debugging 을 구현하는 내용을 포함하고 있다.

Table of Index

1. Introduction

본 문서에서는 PE32 의 기초적인 개념에 대한 설명과 TLS 에 대한 자세한 설명은 생략하기로 했다.

또한 Windows 10 x64 환경에서 해당 변조된 파일을 실행했을 때 PE Loader 의 Exception Handling 방식에 대한 연구도 생략하기로 한다.

2. Implementation

TLS Callback Table 의 Offset 을 찾으려면 PE32 의 IMAGE_OPTIONAL_HEADER 의 DATA DIRECTORIES 중 TLS Table 의 주소를 참조해야 한다. 주소는 RVA(Relative virtual address) 형태이기 때문에 Hex Editor 에서 확인할 때 RVA to Raw 변환이 필요하다.

TLS Table 은 IMAGE_TLS_DIRECTORY 구조이며 AddressOfCallBacks 멤버에 TLS Callback Table 의 주소가 VA(Virtual Address) 형태로 저장되어 있다.
마찬가지로 VA to Raw 변환을 통해 Hex Editor 에서 확인할 수 있다.

```c
#include <stdio.h>
#include <Windows.h>

#pragma comment(linker, "/INCLUDE:__tls_used")

void NTAPI Tls_Callback(PVOID THandle, DWORD Reason, PVOID Reserved) {
    printf("This is TLS Callback.\n");
}

#pragma data_seg(".CRT$XLX")
PIMAGE_TLS_CALLBACK TLS_CALLBACK[] = { Tls_Callback, 0 };
#pragma data_seg()

int main() {
    printf("Anti-Debugging Test ...\n");

    while (1)
        Sleep(200);

    return 0;
}
```

[그림 1] TLS Callback 예제 프로그램

[그림 1]은 테스트를 위한 예제 프로그램이다. Line 11 에서 배열 형태로 TLS Callback 함수들의 주소를 정의한다.

TLS.exe

Member	Offset	Size	Value
StartAddressOfRa...	00006F78	Dword	0041D000
EndAddressOfRaw...	00006F7C	Dword	0041D101
AddressOfIndex	00006F80	Dword	0041A144
AddressOfCallBacks	00006F84	Dword	00417720
SizeOfZeroFill	00006F88	Dword	00000000
Characteristics	00006F8C	Dword	00100000

[그림 2] 컴파일된 예제 프로그램의 TLS Directory

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00005F20	34	13	41	00	00	00	00	00	00	00	00	00	00	00	00	00	4.A..............
00005F30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00005F40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00005F50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00005F60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00005F70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

[그림 3] AddressOfCallBacks 의 File offset 위치에 저장되어 있는 TLS Callback Table

File offset 위치인 0x00005F20 에 코드에서 정의했던 Tls_Callback 함수의 주소가 VA 형태로 저장되어 있다.

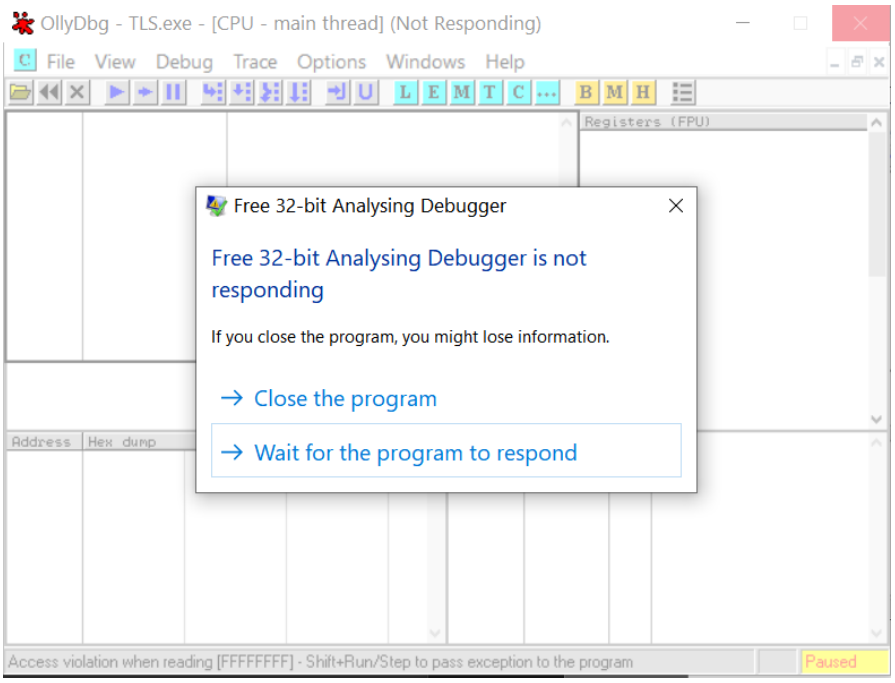
PE Loader 는 AddressOfCallBacks 에 저장되어 있는 Callback Table 의 함수 주소들을 순서대로 호출하고 0 을 만나면 호출을 종료한다. 여기서 다음 4 바이트 주소에 임의의 값을 넣어 보면 아래와 같은 결과를 확인할 수 있다.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00005F20	34	13	41	00	FF	FF	FF	FF	00	00	00	00	00	00	00	00	4.A._...........
00005F30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00005F40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00005F50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00005F60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00005F70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00005F80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

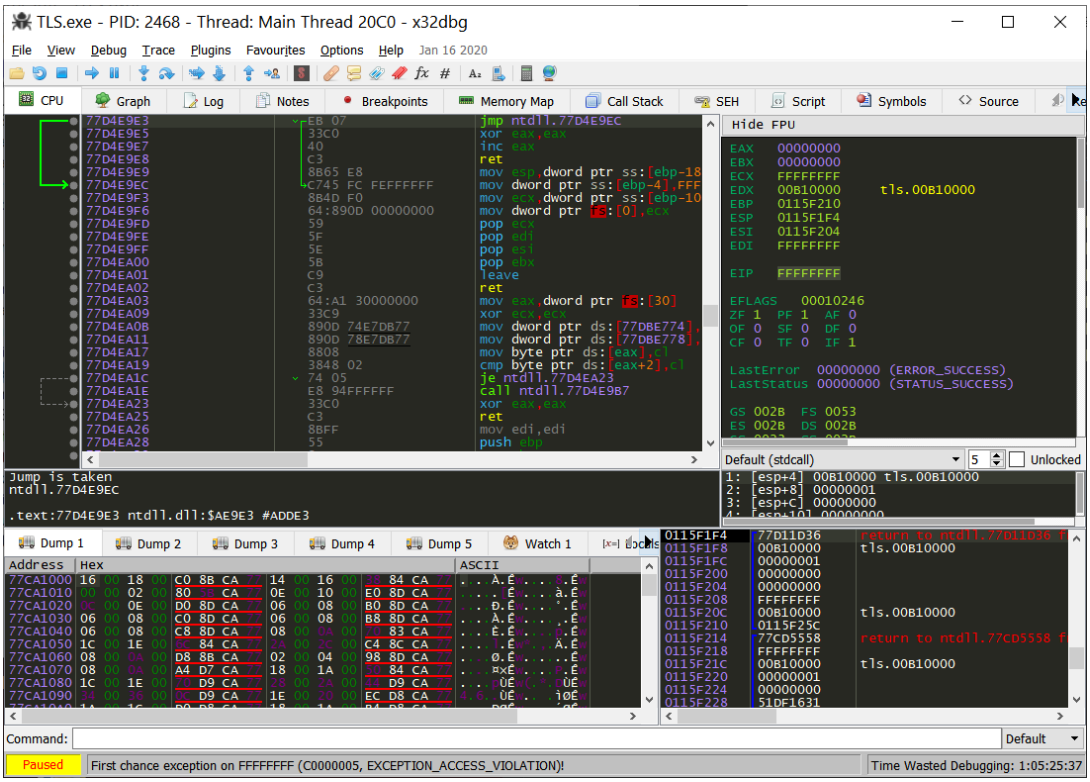
[그림 4] AddressOfCallBacks Table 변조

[그림 4]와 같이 Callback Table 을 변조하면, PE Loader 는 Callback Table 을 참조하여 0x00411334 함수를 호출하고 이어서 0xFFFFFFFF 를 호출하게 된다. 이때 Anti-Debugging 이 가능해진다.

Debugger 에서도 마찬가지로 Main Thread 가 실행되기 전 TLS Callback 함수들이 실행되기 때문에 EIP 가 0xFFFFFFFF 가 되어 오류가 생기게 된다.



[그림 5] ollydbg 220 으로 TLS.exe 를 열었을 때 EIP 가 0xFFFFFFFF 로 이동되어 Crash 발생



[그림 6] x32dbg 로 열었을 때 EIP 가 0xFFFFFFFF 가 되어 실행할 수 없음

각종 Debugger 들로 테스트해본 결과 TLS Callback Table 이 변조된 파일을 Debugger 로 열었을 때 메모리 경곗값을 체크하지 않아 Crash 가 발생되거나 Exception 이 발생되어 F9 으로 Main Thread 를 실행할 수 없는 상태가 되었다.

Anti-Debugging 을 무력화해 주는 대표적인 Plugin 인 Olly Advanced 나 ScyllaHide 의 옵션에서는 TLS 와 관련된 옵션은 Break on TLS 와 같은 옵션뿐이었다. 고의적으로 Error Exception 을 발생시키는 Anti-Debugging 방식에 대한 옵션은 찾아볼 수 없었다.

3. References

1) Microsoft Docs

2) Google