DLL INJECTION
using CreateRemoteThread

[TECHNOLOGY REPORT]

Dec, 2017

By WOODONGGYU

목 차

1. 개요

1-1. DLL Injection 이란

DLL Injection 이 무엇인지 설명하기 앞서 DLL 파일과 사용 목적 등에 대해 먼저 간단히 설명하도록 하겠다.

DLL(Dynamic Linking Library) 는 "동적 라이브러리" 라 불리며, 응용 프로그램의 일부를 동적으로 링크할 수 있는 라이브러리이다. 실행 모듈만 Load 하여 메모리 사용량을 줄이고, 프로그램의 특정 부분만 수정이 가능하기에 디스크 및 메모리를 절약할 수 있고, 쉽게 제작이 가능하다.

DLL Injection 은 다른 프로세스에 특정 DLL 파일을 강제로 로딩시키는 기법이다. 일반적으로 DLL Injetion 기법을 사용하는 목적은 기능 개선 및 버그 수정을 통한 패치이다. 그러나 악성코드는 해당 기법을 활용해 정상 프로세스 위장, 백도어 및 키로그와 같은 추가 악성 행위 등 공격자가 원하는 무엇이든 할 수 있다.

이러한 DLL Injection 기법으로는 레지스트리 조작, SetWindowsHookEx, NtCreateThreadEx, Atom-Bombing 등 다양한 Injection 기법이 존재한다.

이 문서에서는 CreateRemoteThread 함수를 이용한 DLL Injection 에 대해 다룬다.

1-2. 동작 과정

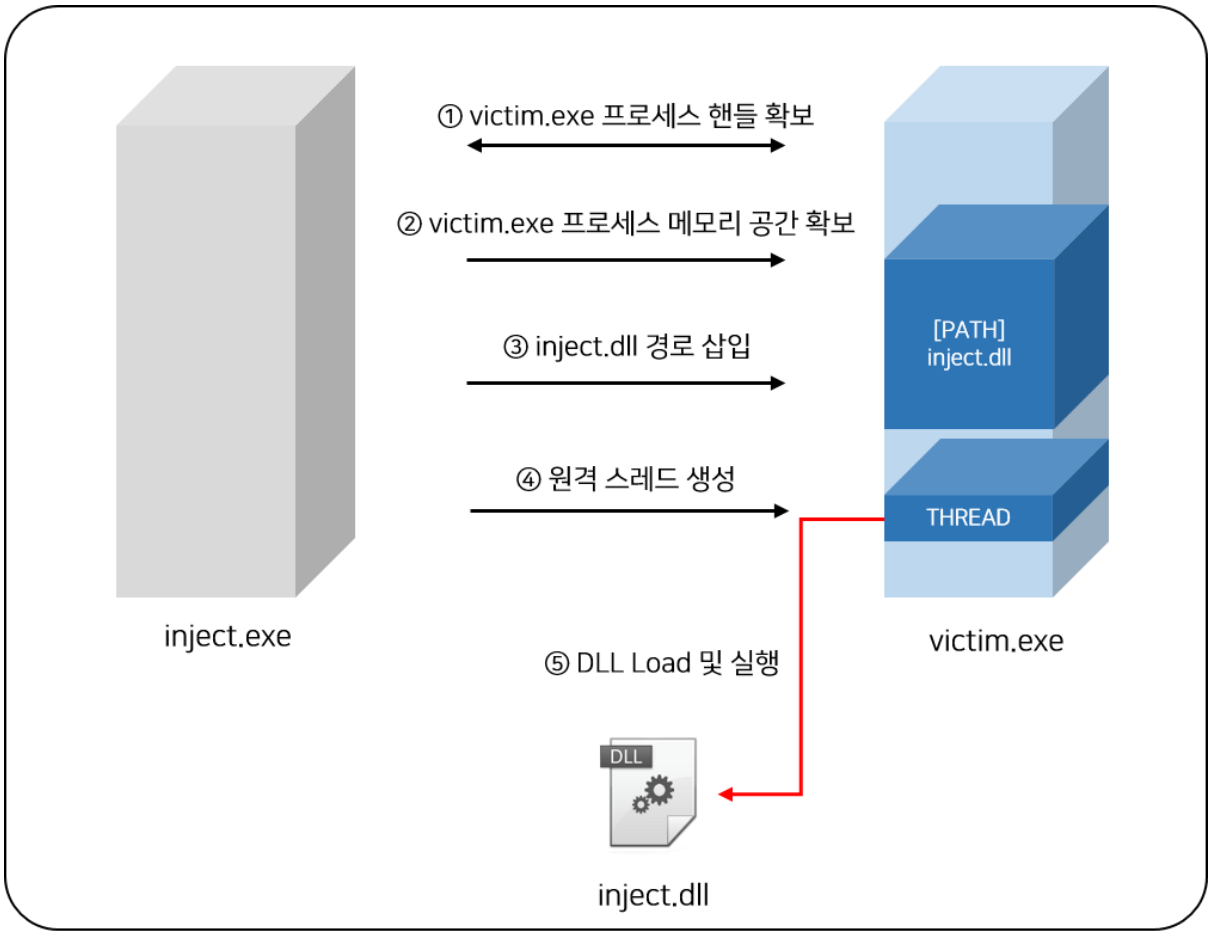


Figure 1. 동작 과정

2. 분석 및 구현

2-1. 동작 분석

DLL Injection 을 위한 동작 순서(위 -> 아래)에 따른 사용되는 함수와 인자들에 대해 설명한다.

· *OpenProcess – 프로세스의 HANDLE 값을 얻어온다.*

Return Value : 지정된 프로세스의 HANDLE 값.

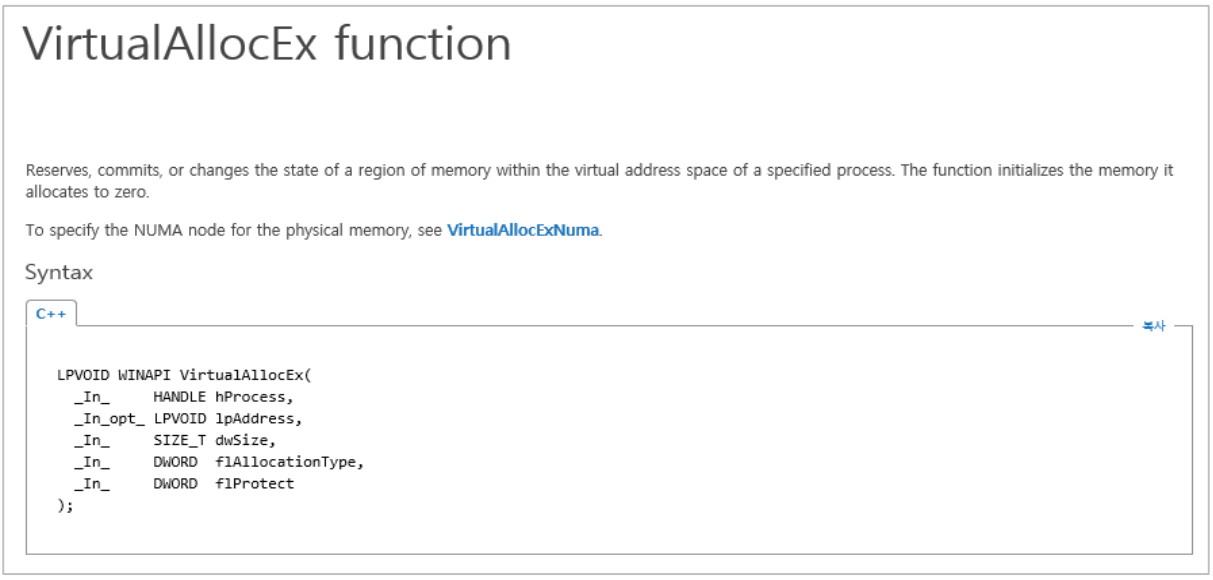
- dwDesiredAccess : PROCESS_ALL_ACCESS(프로세스 객체에 대한 모든 가능한 권한 확보)
- bInheritHandle : FALSE(프로세스가 핸들 상속하지 않음)
- dwProcessId : PID(HANDLE 값을 얻어올 프로세스 식별자)

Figure 2. OpenProcess Function

· *VirtualAllocEx – 지정된 프로세스의 메모리 할당한다.*

Return Value : 메모리에 할당 된 공간의 주소.

- hProcess : 지정된 프로세스 HANDLE 값
- lpAddress :, NULL(비어있는 공간 할당)
- dwSize : NULL(하나의 페이지 크기 지정)
- flAllocationType : MEM_COMMIT(지정 예약된 메모리에 대한 할당)
- flProtect : PAGE_EXECUTE_READWRITE(읽기, 쓰기 및 실행 가능)



VirtualAllocEx function

Reserves, commits, or changes the state of a region of memory within the virtual address space of a specified process. The function initializes the memory it allocates to zero.

To specify the NUMA node for the physical memory, see **VirtualAllocExNuma**.

Syntax

C++

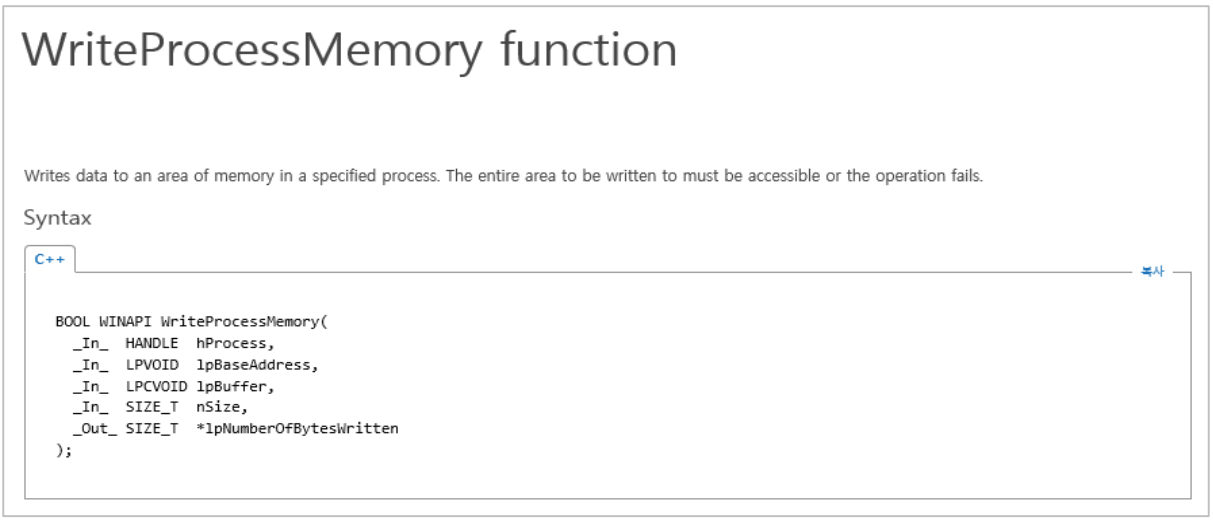
```
LPVOID WINAPI VirtualAllocEx(
  _In_     HANDLE hProcess,
  _In_opt_ LPVOID lpAddress,
  _In_     SIZE_T dwSize,
  _In_     DWORD  flAllocationType,
  _In_     DWORD  flProtect
);
```

Figure 3. VirtualAllocEx Function

· *WriteProcessMemory – 지정된 프로세스의 할당된 메모리 공간에 데이터를 쓴다.*

Return Value : 정상적으로 쓰여졌다면 0 이 아닌 값.

- hProcess : 지정된 프로세스의 HANDLE 값
- lpBaseAddress : VirtualAllocEx Return 값(지정된 프로세스의 데이터가 쓰여지는 메모리 공간의 주소)
- lpBuffer : 지정된 공간에 쓰일 데이터의 주소
- nSize : 쓰여질 데이터의 바이트 수
- *lpNumberOfBytesWritten : NULL(삽입한 데이터의 크기를 저장하지 않음)



WriteProcessMemory function

Writes data to an area of memory in a specified process. The entire area to be written to must be accessible or the operation fails.

Syntax

C++

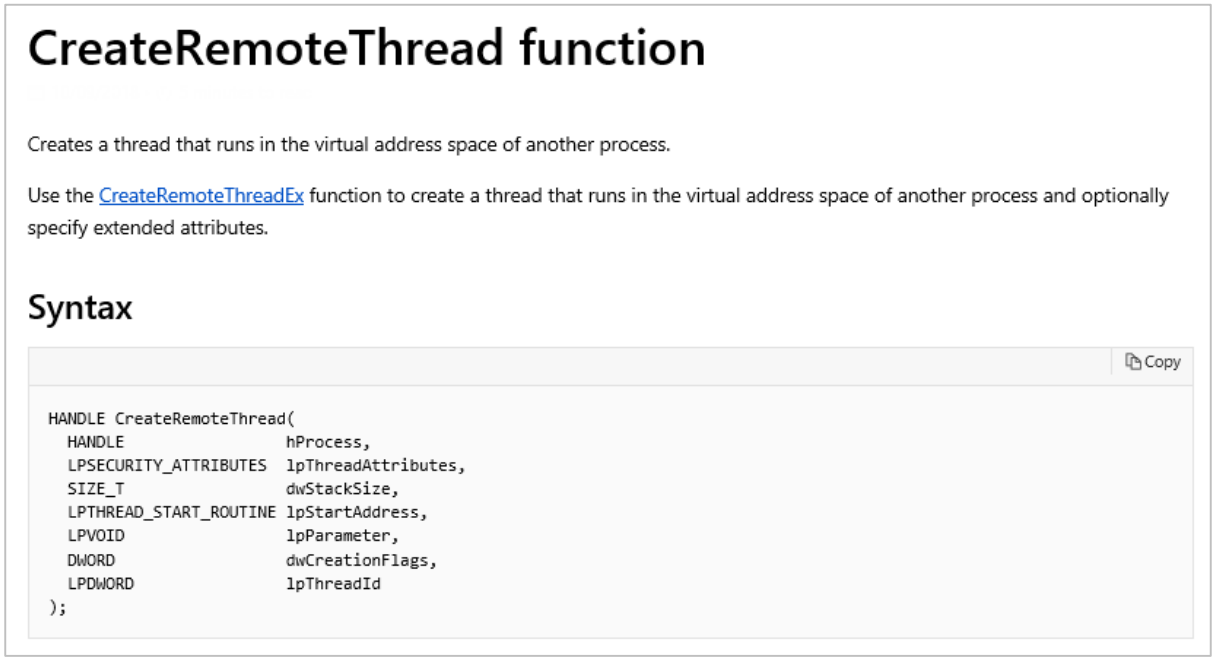
```
BOOL WINAPI WriteProcessMemory(
  _In_  HANDLE  hProcess,
  _In_  LPVOID  lpBaseAddress,
  _In_  LPCVOID lpBuffer,
  _In_  SIZE_T  nSize,
  _Out_ SIZE_T  *lpNumberOfBytesWritten
);
```

Figure 4. WriteProcessMemory Function

· *CreateRemoteThread – 지정된 프로세스에서 실행되는 Thread 를 생성한다.*

Return Value : 성공 시 반환 값은 새로운 스레드의 핸들 값.

- hProcess : 지정된 프로세스의 HANDLE 값
- lpThreadAttributes : NULL(자식 프로세스에 Thread Handle 상속 여부)
- dwStackSize : 0(Thread Stack 크기 0)
- lpStartAddress : Loadibrary Address(Thread 가 실행될 메모리의 주소)
- lpParameter : 할당된 메모리의 주소(LoadLibrary("Load 할 DLL 경로 값"))
- dwCreationFlags : NULL(Thread 속성, 별도로 설정하지 않으므로 NULL)
- lpThreadId : NULL(Thread ID 설정 인자)



CreateRemoteThread function

Creates a thread that runs in the virtual address space of another process.

Use the CreateRemoteThreadEx function to create a thread that runs in the virtual address space of another process and optionally specify extended attributes.

Syntax

Copy

```
HANDLE CreateRemoteThread(
  HANDLE                 hProcess,
  LPSECURITY_ATTRIBUTES  lpThreadAttributes,
  SIZE_T                 dwStackSize,
  LPTHREAD_START_ROUTINE lpStartAddress,
  LPVOID                 lpParameter,
  DWORD                  dwCreationFlags,
  LPDWORD                lpThreadId
);
```

Figure 5. CreateRemoteThread Function

2-2. 구현 및 결과

소스코드는 아래의 링크를 통해 다운로드 및 테스트할 수 있다.

https://github.com/woodonggyu/injection-techniques/blob/master/CreateRemoteThread/CreateRemoteThread.c

테스트 결과는 아래와 같이 정상적으로 동작한 것을 확인할 수 있다.

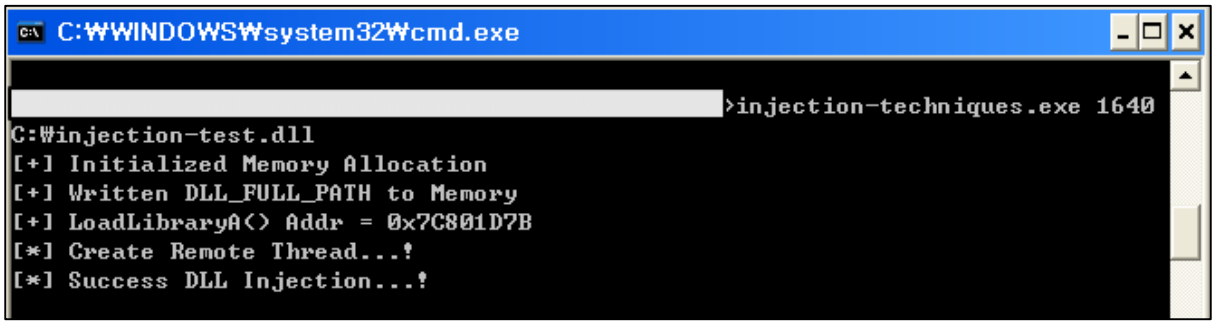


Figure 6. 인젝션 프로그램 실행

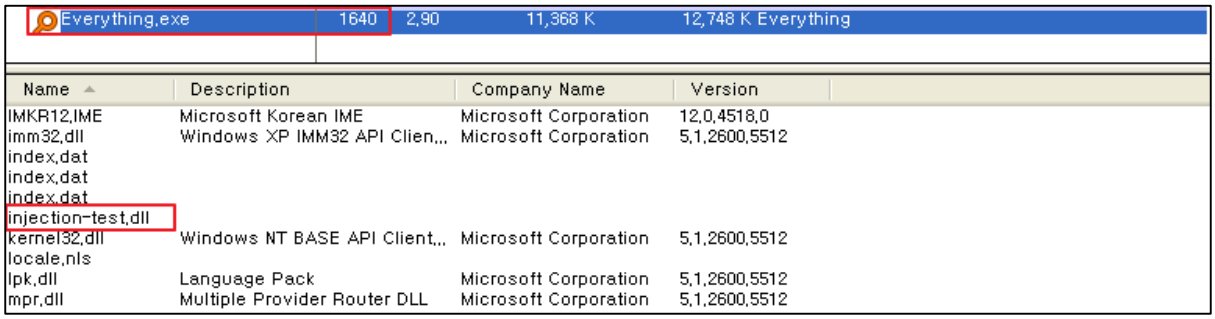


Figure 7. 실행 결과

2-3. 탐지 방안

CreateRemoteThread 함수 호출 시, CreateRemoteThread() -> CreateThreat() -> BaseThreadStartThunk() 순으로 함수를 호출된다. 이 때 BaseThreadStartThunk 함수를 후킹하여 lpStartAddress 인자가 LoadLibrary 또는 GetProcAddress 함수 호출 시 DLL Injection 으로 간주할 수 있다.

3. 기타

3-1. 레지스트리 조작을 통한 DLL Injection

· *HKEY_LOCAL_MACHINE₩Software₩Microsoft₩Windows NT₩CurrentVersion₩Windows₩AppInit_DLLs*

- AppInit_DLLs 값에 Injection 하길 원하는 DLL 경로 입력 후 재부팅 할 시, 이후 Windows 운영체제는 재부팅하면서 실행되는 모든 프로세스에 해당 DLL 을 Injection 한다.

· *HKEY_LOCAL_MACHINE₩Software₩Microsoft₩Windows NT₩CurrentVersion₩Windows₩LoadAppInit_DLLs*

- AppInit_DLLs 활성화 여부 확인 항목(0 : 비활성화, 1: 활성화)

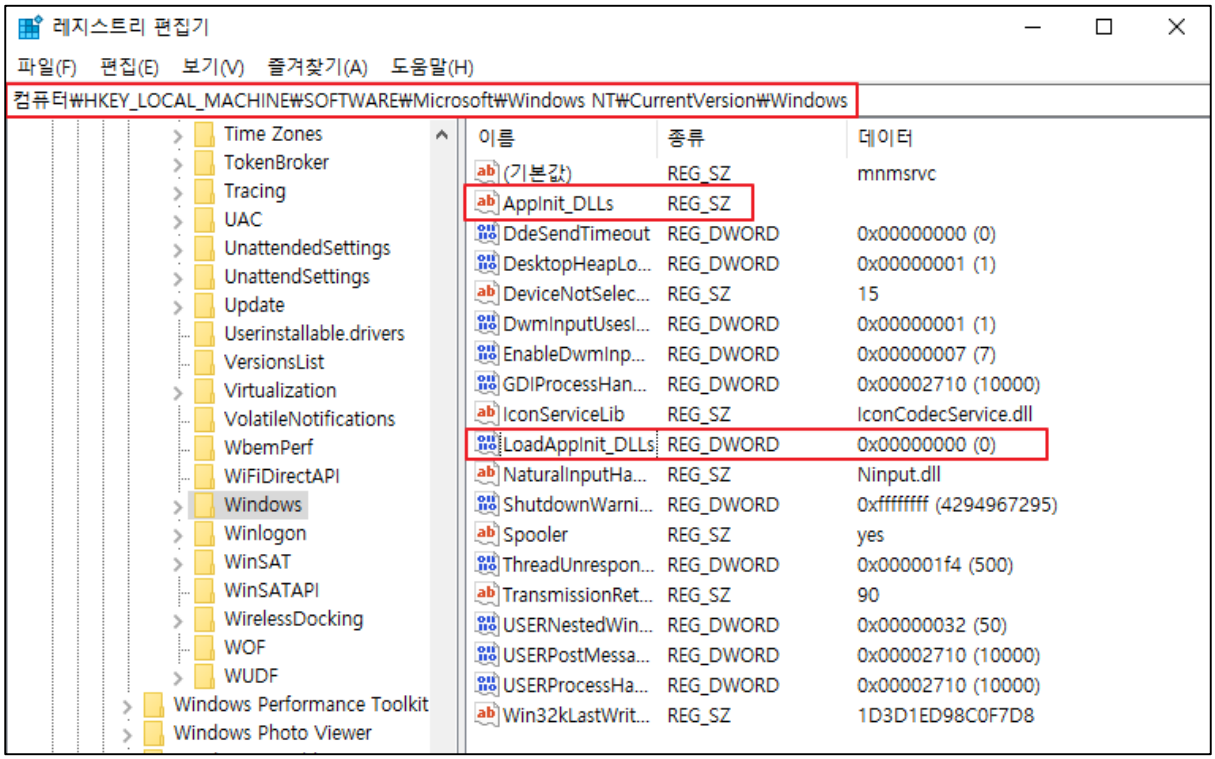


Figure 8. AppInit_DLLs, LoadAppInit_DLLs 레지스트리 경로

3-2. 레퍼런스

· *[Book] 리버싱 핵심 원리*

· *Injection 기법_20140417_공개 버전.pdf*