PROPCESS HOLLOWING

[TECHNOLOGY REPORT]

Apr, 2018

By WOODONGGYU

목 차

1. 개요

1-1. Process Hollowing 이란

Process Hollowing(프로세스 할로잉)은 간단히 말하면 프로세스에 프로세스 자체를 인젝션하는 기법을 뜻한다. 주로 악성코드에서 사용하며, 공격자는 정상 프로세스에 악성PE 데이터를 인젝션하여 정상 프로세스인 것처럼 위장한다.

이러한 Process Injection 기법에는 Process Doppelganging, Double Process Hollowing 등이 있다.

해당 문서에서는 Process Hollowing 기법과 분석 · 구현에 대해 다루도록 한다.

1-2. 동작 과정

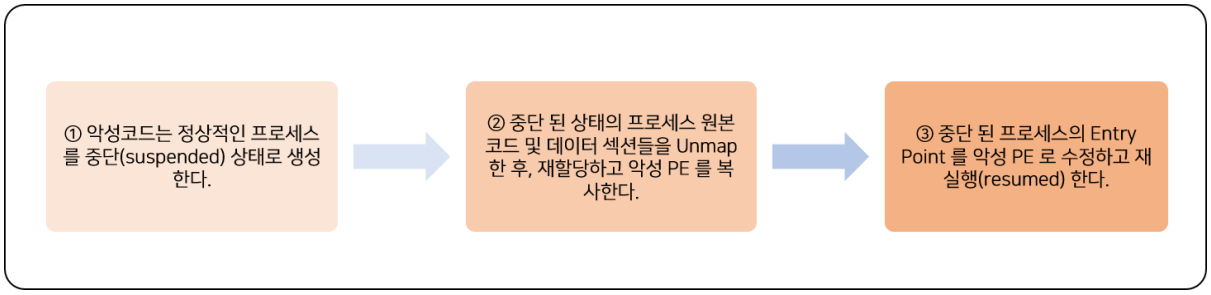


Figure 1. 동작 과정

2. 분석 및 구현

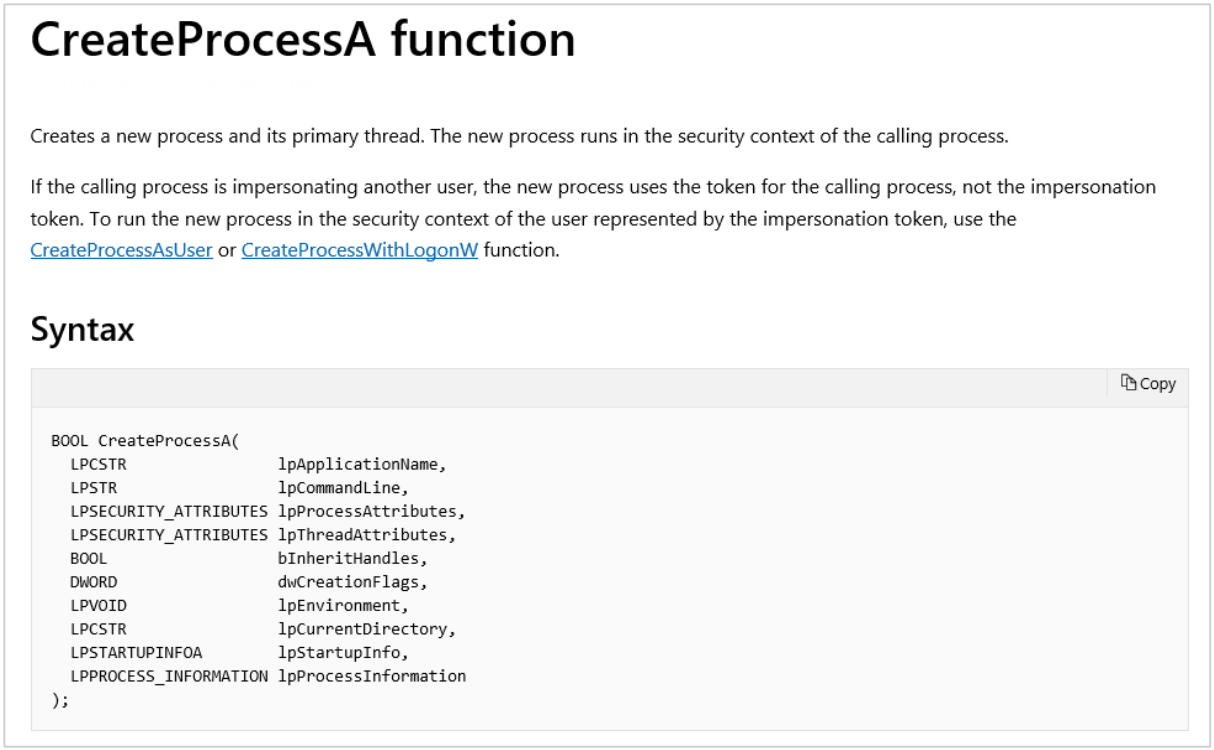
Process Hollowing 의 동작 순서(위 -> 아래)에 따른 사용되는 함수들에 대해 설명한다.

· *CreateProcess – 특정 프로세스 생성한다.*

Return Value : 성공적으로 호출되었다면 0 이 아닌 값.

- lpApplicationName : 0
- lpCommandLine : target process
- lpProcessAttributes : 0
- lpThreadAttributes : 0
- bInheritHandles : 0

- dwCreationFlags : CREATE_SUSPENDED
- lpEnvironment : 0
- lpCurrentDirectory : 0
- lpStartupInfo : &STARTUPINFO
- lpProcessInformation : &PROCESS_INFORMATION



CreateProcessA function

Creates a new process and its primary thread. The new process runs in the security context of the calling process.

If the calling process is impersonating another user, the new process uses the token for the calling process, not the impersonation token. To run the new process in the security context of the user represented by the impersonation token, use the CreateProcessAsUser or CreateProcessWithLogonW function.

Syntax

Copy

```
BOOL CreateProcessA(
  LPCSTR                lpApplicationName,
  LPSTR                 lpCommandLine,
  LPSECURITY_ATTRIBUTES lpProcessAttributes,
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  BOOL                  bInheritHandles,
  DWORD                 dwCreationFlags,
  LPVOID                lpEnvironment,
  LPCSTR                lpCurrentDirectory,
  LPSTARTUPINFOA        lpStartupInfo,
  LPPROCESS_INFORMATION lpProcessInformation
);
```

Figure 2. CreateProcessA Function

· GetThreadContext – 특정 Thread 의 상태를 가져온다.

Return Value : 성공적으로 호출되었다면 0 이 아닌 값.

- hThread : &lpProcessInformation.hThread(가져 올 Thread 핸들)
- lpContext : &CONTEXT(지정된 Thread 의 Context 를 수신하는 CONTEXT 구조체 포인터)

GetThreadContext function

Retrieves the context of the specified thread.

A 64-bit application can retrieve the context of a WOW64 thread using the Wow64GetThreadContext function.

Syntax

Copy

```
BOOL GetThreadContext(
  HANDLE    hThread,
  LPCONTEXT lpContext
);
```

Figure 3. GetThreadContext Function

· *ZwUnmapViewOfSection – 대상 프로세스의 PE Image 할당 해제한다.*

Return Value : 성공적으로 호출되었다면 STATUS_SUCCESS 반환.

- ProcessHandle : 대상 프로세스 핸들
- BaseAddress : Image Base 주소

ZwUnmapViewOfSection function

The **ZwUnmapViewOfSection** routine unmaps a view of a section from the virtual address space of a subject process.

Syntax

Copy

```
NTSYSAPI NTSTATUS ZwUnmapViewOfSection(
  HANDLE ProcessHandle,
  PVOID  BaseAddress
);
```

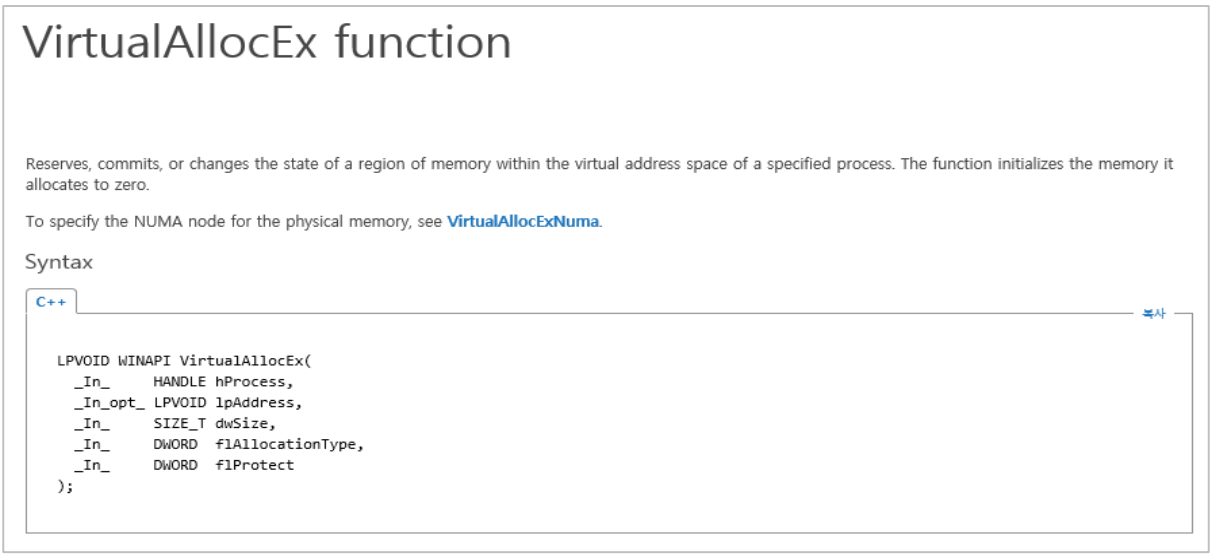
Figure 4. ZwUnmapViewOfSection Function

· *VirtualAllocEx – 지정된 프로세스의 메모리 할당한다.*

Return Value : 메모리에 할당 된 공간의 주소.

- hProcess : 지정된 프로세스 HANDLE 값
- lpAddress :, Image Base
- dwSize : Size of Image
- 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++

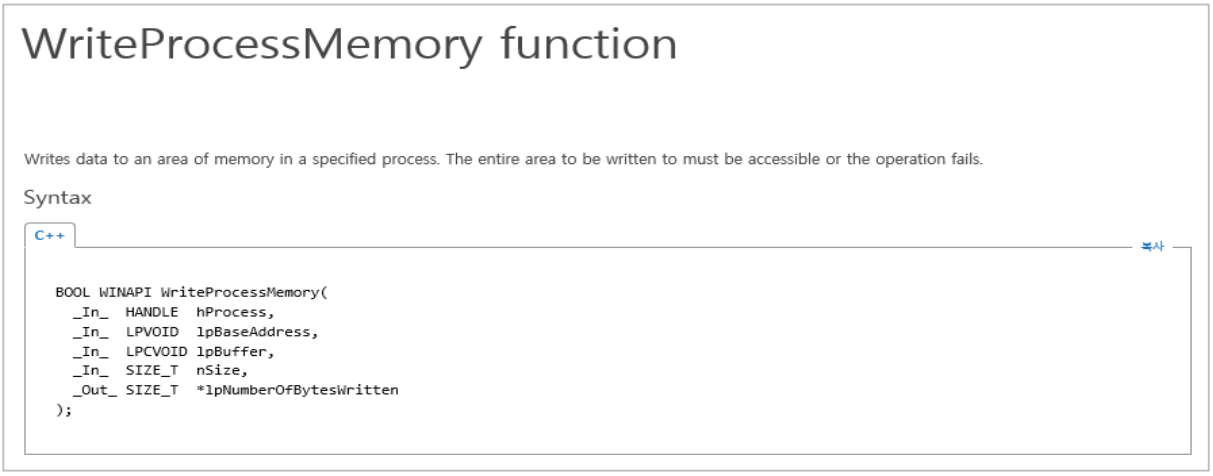
```
LPVOID WINAPI VirtualAllocEx(
  _In_     HANDLE hProcess,
  _In_opt_ LPVOID lpAddress,
  _In_     SIZE_T dwSize,
  _In_     DWORD  flAllocationType,
  _In_     DWORD  flProtect
);
```

Figure 5. 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 6. WriteProcessMemory Function

· *SetThreadContext – 특정 Thread 의 상태를 설정한다.*

Return Value : 성공적으로 설정되었다면 0 이 아닌 값.

- hThread : &lpProcessInformation.hThread(가져 올 Thread 핸들)
- *lpContext : &CONTEXT(지정된 Thread 의 Context 를 설정할 CONTEXT 구조체 포인터)

SetThreadContext function

Sets the context for the specified thread.

A 64-bit application can set the context of a WOW64 thread using the Wow64SetThreadContext function.

Syntax

Copy

```
BOOL SetThreadContext(
  HANDLE        hThread,
  const CONTEXT *lpContext
);
```

Figure 7. SetThreadContext Function

· *ResumeThread – SUSPEND 상태의 Thread 를 재 실행시킨다.*

Return Value : 성공적으로 호출되었다면 Thread 의 이전 중단 횟수를 반환한다.

- hThread : 재 실행시킬 Thread 핸들 값.

ResumeThread function

Decrements a thread's suspend count. When the suspend count is decremented to zero, the execution of the thread is resumed.

Syntax

Copy

```
DWORD ResumeThread(
  HANDLE hThread
);
```

Figure 8. ResumeThread Function

2-2. 구현 및 결과

소스코드는 아래의 링크를 통해 다운로드 및 테스트할 수 있다.

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

2-3. 탐지 방안

Volatility Framework 를 이용하여 탐지할 수 있다. Volatility 는 Hollow Process 를 탐지하기 위한 방법으로 RWX 보호 플래그 값을 가진 메모리 영역을 스캔한다. 일반적으로 메모리 영역은 RW 플래그가 설정되어 있기에 이를 이용한 탐지 방법이다. 만약, 공격자가 VirtualProtectEx 함수를 이용하여 메모리 보호 플래그를 RW 플래그로 수정한다면 탐지가 불가능하다.

그 외에도 부모-자식 프로세스 관계에 의한 탐지, PEB(Process Environment Block) 와 VAD(Virtual Address Descriptor) 구조 비교를 통한 탐지 방법 등이 있다.

3. 기타

3-1. 레퍼런스

- *https://github.com/m0n0ph1/Process-Hollowing/blob/master/sourcecode/ProcessHollowing/ProcessHollowing.cpp*
- *https://cysinfo.com/detecting-deceptive-hollowing-techniques/*
- *https://www.endgame.com/blog/technical-blog/ten-process-injection-techniques-technical-survey-common-and-trending-process*