

Kimsuky's Advanced Attack Techniques: JSONPing, Webex Spoofing, and a New HttpSpy Variant

enki.co.kr/en/media-center/blog/kimsuky-s-advanced-attack-techniques-jsonping-webex-spoofing-and-a-new-httpspy-variant

May 27, 2026



Executive Summary

- Through April 2026, we identified multiple cases where Kimsuky deployed malware against South Korean military and corporate targets.
- Kimsuky employed a range of tailored social engineering tactics, such as spoofing security software installation pages and crafting a fake Webex meeting page that leveraged a legitimate meeting schedule.
- We identified a technique ("JSONPing") in which the distribution page uses JSONP to verify in real time whether the victim has executed the malware.
- We identified the final payload as an HttpSpy variant, now operating through a new three-stage execution chain (Installer - Loader - HttpSpy) that replaced the previous single-binary architecture.
- We confirmed several indicators linking to Kimsuky across attack infrastructure, code patterns, and encryption key reuse.

1. Overview

This report details how Kimsuky targeted South Korean military and enterprises through April 2026, combining tailored social engineering with a revamped HttpSpy execution chain.

Our analysis of the Webex-spoofing case revealed the full execution chain of the final payload, an HttpSpy variant. Unlike previous versions of HttpSpy that operated as a single binary, this variant splits the installation process into three stages. In the Zsecurity software-spoofing case, we were only able to recover artifacts up to the downloader stage; however, we attributed both campaigns to the same threat actor based on shared RC4 keys, infrastructure, and code patterns.

Kimsuky added JSONP communication functionality to the fake web pages, allowing them to query a local server set up by the malware on the victim's system. This enabled the page to check whether the malware had been executed and prompt installation if it had not (JSONPing; see Section 4.4). Additionally, we observed evidence that Kimsuky leveraged meeting schedule information from a presumably already-compromised victim to craft a fake meeting page and distribute malware to other participants.

2. Background

Kimsuky is a DPRK-nexus threat actor, [first identified by Kaspersky in 2013](#), that primarily targets South Korea.

Kimsuky has frequently disguising malware as installers from South Korean security software vendors when targeting South Korean entities. A sample [analyzed by ESTSecurity in 2023](#) followed this pattern. Upon execution, it launched a legitimate installer while silently dropping and executing a second-stage payload malware in the background.

In a [case reported by AhnLab in 2024](#), Kimsuky directly compromised a construction-related association website to distribute malware disguised as security software. The malware delivered through the site was signed with a valid certificate from a South Korean company.

This tactic of abusing compromised code-signing certificates also surfaced in the [May 2025 C.J Olivenetworks certificate abuse case](#), which involved HttpSpy. In that case, a Go-based .scr executable served as the dropper, loading a second-stage payload via rundll32.exe and calling the hello export function. This same pattern was also documented in [Kaspersky's HelloDoor report](#).

3. Security Software Spoofing

 Security software spoofing attack overview
Security software spoofing attack overview

3.1. Fake Security Software Installation Page

Around March 2026, we discovered a campaign where Kimsuky distributed malware through a fake page impersonating the security software installation page of a South Korean B2B messaging service. Given that the spoofed service targets enterprises rather than the general public, this campaign likely targeted messaging administrators within corporate environments.

URL: `hxxps://www.ibizplus.n-e[.]kr/install.html`

 Fake security software installation page
Fake security software installation page

Clicking the "전체설치"("Full Install") or "다운로드"("Download") buttons triggers a malware download. The files delivered by each button are listed in the table below.

Button	URL	Malware
다운로드 - Personal PC Firewall (ASTX)	<code>hxxps://www.ibizplus.n-e[.]kr/download.php?id=30382119</code>	<code>astx-setup.exe</code>
다운로드 - Keyboard Security (NOS)	<code>hxxps://www.ibizplus.n-e[.]kr/download.php?id=30382120</code>	<code>nos-setup.exe</code>
전체설치	<code>hxxps://www.ibizplus.n-e[.]kr/download.php?id=30382121</code>	<code>nos-setup.exe</code>

Download URLs and payloads per button

Upon reviewing the HTML source, we found that the attacker had copied the HTML from 우리은행's (Woori Bank's) security software installation page rather than the messaging service provider's, modified it, and added the provider's logo.

 Traces of copied Woori Bank HTML code
Traces of copied Woori Bank HTML code

 Woori Bank security software installation page
Woori Bank security software installation page

3.2. Droppers and Downloaders

3.2.1. nos-setup.exe & astx-setup.exe

`nos-setup.exe` and `astx-setup.exe` are dropper malware that differ only in the legitimate installer used as a decoy file. Their malicious behavior, subsequent stages, and metadata are all identical. Details are shown in the table below.

File Name	<code>nos-setup.exe</code>	<code>astx-setup.exe</code>
Decoy	nProtect Online Security V1.0	AhnLab Safe Transaction

Dropped malware MD5	bea602695d58cbf25fff058834e36c1d	bea602695d58cbf25fff058834e36c1d
Dropped malware path	C:\Programdata\[a-z]{8}.dat	C:\Programdata\[a-z]{8}.dat
Self-deletion file path	%Temp%\msbuil.bat	%Temp%\msbuil.bat
RC4 Key	#RsfsetraW#@EsfesgsgAJOPj4eml;	#RsfsetraW#@EsfesgsgAJOPj4eml;
Linker	Microsoft Linker(14.36.33523)	Microsoft Linker(14.36.33523)
Compiler	Microsoft Visual C/C++(19.36.33523)[LTCG/C++]	Microsoft Visual C/C++(19.36.33523)[LTCG/C++]
Toolchain	Visual Studio(2022, v17.6)	Visual Studio(2022, v17.6)
Language	C++	C

Security software spoofing malware comparison

Upon execution, the malware decrypts embedded RC4-encrypted data to produce a second-stage payload and a decoy file. The decoy file is written to the current directory as `nos_setup.exe` or `astx-setup.exe`, while the second-stage payload is written to `C:\Programdata\[a-z]{8}.dat`. The RC4 key used for decryption is as follows.

RC4 key: `#RsfsetraW#@EsfesgsgAJOPj4eml;`

DLL name generation routine

DLL name generation routine

The decrypted second-stage payload is a DLL with the original filename "MemLoader.dll" in its export table. The malware is executed via `regsvr32.exe` using the `CreateProcess` function, while the decoy file is launched separately via `ShellExecuteW`.

Legitimate `nos_setup.exe` execution result

Legitimate `nos_setup.exe` execution result

Legitimate `astx-setup.exe` execution result

caption - Legitimate `astx-setup.exe` execution result

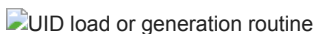
Once all operations are complete, `nos_setup.exe` and `astx-setup.exe` generates and runs a batch file to delete itself. The file path and contents are as follows:

Batch file path: `%Temp%\msbuild.bat`

```
:L1
del /F "<path of the running file>"
if exist "<path of the running file>" goto L1
del /F "%Temp%\msbuild.bat"
```

3.2.2. MemLoader.dll

Upon execution, the malware first checks whether a `[running filename].cfg` file exists in the same directory as the currently running file. If the file exists, the malware reads its contents and uses it as a UID. If not, it generates a random 8-character string from the `[0-9a-fA-F]` range as the UID. Once the UID is determined, the malware prepends `S-` if the current process has administrator privileges, or `U-` if not. The finalized UID is then saved to the `[running filename].cfg` file.

UID load or generation routine

UID load or generation routine

After setting the UID, the malware writes an embedded payload to `C:\programdata\calc.exe` and executes it via `ShellExecuteW`. `calc.exe` spawns a temporary server on localhost port 62001. When it receives a request to the `/ping` path with a `callback` parameter, it returns a JSONP-formatted response wrapped in the specified callback function. This server allows the distribution page to verify whether the victim has been infected, as detailed in Section 3.4.

Port binding routine

Port binding routine

To establish persistence, the malware registers a scheduled task that runs at one-minute intervals. The task varies depending on the execution format and privilege level of the running malware. Details for each case are shown in the table below.

Format	Privilege	Task Name	Command
DLL	Admin	ChromeUpdate	Uses regsvr32 + highest privileges
DLL	User	EdgeUpdate	Uses regsvr32
EXE	Admin	ChromeUpdate	Direct execution + highest privileges
EXE	User	EdgeUpdate	Direct execution

Scheduled task registration by execution format and privilege level

The malware sends a GET request to the C&C server, using the UID as the bearer token. If the response contains data, it is decrypted with RC4, loaded into memory, and the `hello` function is called. The RC4 key is identical to the one used by the dropper. The URL used for the download request is as follows.

URL: `hxxps://load.serverpit[.]com/fwrite.php`

Payload download request routine
Payload download request routine

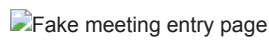
We were unable to retrieve the subsequent payload through by directly requesting the URL. The attacker likely monitored the recurring GET requests from the malware and selectively delivered payloads to specific victims.

4. Webex Spoofing

4.1. Fake Webex Meeting Page

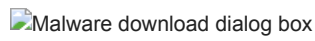
In April 2026, we identified a case where Kimsuky distributed malware through a malicious page impersonating Webex, Cisco's online meeting service. Notably, the fake meeting page was crafted based on an actual Webex meeting schedule, suggesting that the attacker had previously compromised a attendee's account or device to obtain the schedule. Rendering the HTML code retrieved from the URL below displays a loading UI over a blurred meeting entry page background.

URL: `hxxps://conference.birdriver[.]org/`

Fake meeting entry page
Fake meeting entry page

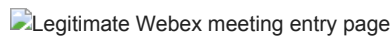
Five seconds after the page loads, the fake meeting page displays a dialog box instructing the user to install and run a camera patch script, and prompts them to click the confirm button. Clicking the button downloads an ALZip archive containing a `jse` file. The URL endpoint and parameter structure are identical to those used in the fake installation page from the security software spoofing case.

URL: `hxxps://download.birdriver[.]org/download.php?id=425623`

Malware download dialog box
Malware download dialog box

Executing the downloaded `jse` file ultimately installs an HttpSpy variant on the system. The malware also drops and opens `meeting.html`, which immediately redirects the victim to a Webex meeting room.

Accessing the redirect URL opens a legitimate Webex meeting room. We confirmed that the meeting was a legitimate scheduled event around the time of the malware distribution. This indicates that the attacker likely compromised an service member's device or account to obtain the meeting schedule, then crafted a fake meeting page to distribute malware to the other attendees.

Legitimate Webex meeting entry page
Legitimate Webex meeting entry page

4.2. jse Dropper and Downloader

4.2.1. fix-camera.jse

fix-camera.jse drops and executes a base64-encoded malware payload and a decoy HTML file to C:\ProgramData\mTSTCv8.mdxm and C:\ProgramData\meeting.html, respectively. The jse script employs several obfuscation techniques, including assigning base64-encoded data to junk variables and slicing strings to reassemble them with "+" concatenation.

 Obfuscated jse script

Obfuscated jse script

The malware is double base64-encoded. The script first decodes it and drops the result to C:\ProgramData\mTXDZew.sz8f, then uses certutil to perform a second decode and saves the final payload to C:\ProgramData\mTSTCv8.mdxm. mTSTCv8.mdxm is executed via the following command.

```
powershell.exe -windowstyle hidden regsvr32.exe /s C:\ProgramData\mTSTCv8.mdxm
```

4.2.2. mTSTCv8.mdxm (loadDll.dll)

mTSTCv8.mdxm is a downloader that retrieves a second-stage payload from the C&C server. Its export name is loadDll.dll, and it contains the following PDB path.

```
PDB Path: C:\Users\jira\Documents\My_Received_Files\loadDll\x64\Release\loadDll.pdb
```

Upon execution, the malware checks for VM environments and analysis tools, terminating the process if either is found. The VM detection routine reads the following two registry values and checks for the strings "VMware" and "VirtualBox".

```
HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\BIOS\SystemManufacturer
```

```
HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\BIOS\SystemProductName
```

For analysis tool detection, the malware enumerates all running processes and open window titles, checking them against a hardcoded list of strings. The full list of detection targets can be found in Appendix D, "Analysis tool detection list" Finally, it downloads an additional payload from the URL below, loads it into the current process memory, resolves the address of the Play export function, and executes it.

```
URL : hxxps://download[.]birdriver[.]org/download.php?id=393156
```

 Malware download and execution routine

Malware download and execution routine

The downloaded payload goes through a three-stage installation process that ultimately installs an HttpSpy variant on the victim's system.

4.3. HttpSpy Variant Execution Flow

4.3.1. engine.dat (spyInster.dll)

engine.dat is the payload downloaded by loadDll.dll, with the export name spyInster.dll. As the export name suggests, its role is to install the final payload, HttpSpy, on the victim's system.

engine.dat employs two types of string obfuscation. Wide character (wchar) strings are decoded by subtracting a fixed offset from each character, while regular strings are XORed with a key that incorporates the position index. engine.dat dynamically restores obfuscated API names, hashes them with FNV-1a, and compares the results against the export table entries of loaded modules to resolve function pointers.

 wchar string obfuscation routine

wchar string obfuscation routine

 Regular string obfuscation routine

Regular string obfuscation routine

engine.dat decrypts an embedded RC4-encrypted payload and drops it to C:\Users\Public\cacheMon.dat. It then RC4-decrypts the configuration data and appends it to cacheMon.dat as a DATA_CONF alternate data stream (ADS). The RC4 keys used to decrypt cacheMon.dat and the ADS configuration file are as follows.

- cacheMon.dat decryption RC4 key: %^fseRw#r3qwrwfsddREfGEgse)(14);
- ADS decryption RC4 key: Rgdcsefd@#%dg9ser3\$#\$^@34sdfx1

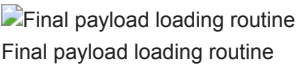
Just before appending the configuration file, the malware patches a 32-bit value computed by `rand() * rand() / 2` into offset 0x1228 of the config data. This value is later used by the HttpSpy main module as a victim identifier. Finally, the malware registers an autorun command in the registry and directly executes `C:\Users\Public\cacheMon.dat` using `regsvr32.exe`. The registered registry value and data are as follows.

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\MSEdgeUpdateInstaller = C:\Windows\System32\regsvr32.exe /s
C:\Users\Public\cacheMon.dat
```

4.3.2. cacheMon.dat (spyLoader.dll)

`cacheMon.dat`, dropped by `engine.dat`, is an HttpSpy loader with the export name `spyLoader.dll`. It decrypts the RC4-encrypted HttpSpy main module, loads it into the current process memory, resolves the address of the `hello` export function, and executes it. The RC4 decryption key is as follows.

```
RC4 Key: #RsfsetraW#@EsfesgsgAJ0Pj4eml;
```



4.3.3. HttpSpy Main Module

The final payload is a RAT with the export name `httpSpy.dll`. It shares the same export name as the final payload used in the May 2025 CJ Olivenetworks certificate abuse attack, and its C&C communication protocol and remote command structure are highly similar. The compilation timestamp is recorded as October 15, 2025, suggesting the malware may have been in use since the latter half of 2025.

In its main loop, the malware loads the C&C server URL from the configuration data previously appended to `cacheMon.dat` as an ADS, and receives remote commands via HTTP POST. If a proxy address is set at offset +0x410 (1040) in the configuration file, communication is routed through that proxy, with `2[. ]2[. ]2[. ]2` used as the proxy bypass address. The configuration data extracted from the analyzed sample is shown in the table below.

Offset	Size	Purpose	Initial Value
+0	520	Primary C&C URL (wide string)	hxxp://hdrgrdfes[.]chickenkiller[.]com/index.php
+520	520	Secondary C&C URL (wide string)	(Empty)
+1040	520	Proxy Address (wide string)	(Empty)
+1560	520	HTTP Auth Username (wide string)	(Empty)
+2080	520	HTTP Auth Password (wide string)	(Empty)
+2600	2048	Unused Space	
+4648	4	Session ID (DWORD)	rand()*rand()/2

HttpSpy configuration data structure

The parameters and values used in C&C server communication are shown in the table below.

Parameter	Value
_sessionchk	ck_param_auto
_logininfo	login_ok for receiving commands / login_fail for sending data
_pkgparam	Session ID (8-digit HEX)
_lockdata	Encrypted and base64-encoded data

HttpSpy C&C communication parameters

All data sent and received during communication is encrypted with the following RC4 key and then base64-encoded.

```
RC4 Key: RGDcsedfd@#%dg9ser3$#$^@34sdfx1
```

The supported command codes and their actions are shown in the table below.

Command Code	Action
d	Execute a shell command, convert ANSI output to Unicode, and send to C&C server
e	Download a file from the C&C server
f	Upload a specified file to the C&C server in 384KB chunks
g	Execute a specified command in a hidden window via CreateProcessW
h	Execute a process in a specified WTS session
i	Overwrite a specified file with the 0x5F pattern, rename it to random lowercase characters, and delete it
j	Capture a full-screen BMP via GDI and upload to C&C server
k	Send the internal configuration buffer to the C&C server
l	Receive new configuration data from the C&C server and update the ADS
m	Perform a TCP connect test to a specified IP:Port
n	Sleep for a specified duration (in hours)
o	Copy the timestamp from a source file to a target file
p	Delete itself, remove persistence entries, and terminate (uninstall)
q	No operation
r	Execute a shell command and send raw binary output to C&C server
s	Inject a DLL path into a specified PID process (Remote DLL Injection)

HttpSpy remote command codes

For command codes d and r, if the argument starts with cd, the malware changes the working directory using SetCurrentDirectoryW and sends the updated path to the C&C server. For all other commands, it executes them via cmd.exe /c, redirects standard output to a temporary file matching %TEMP%\NK[0-9a-fA-F]+\ .tmp, collects the output, and sends it to the C&C server.

Command processing routine

Command processing routine

BMP data captured by command code j is saved to a temporary file matching %TEMP%JG[0-9a-fA-F]{4}\ .tmp, uploaded to the C&C server, and deleted after the upload completes.

Screenshot routine

Screenshot routine

4.4. Additional Security Software Spoofing Pages

Two new variants of security software installation spoofing HTML files were found on the same C&C server as the fake meeting page. Both pages periodically poll the C&C server and, depending on the response value, perform one of three actions: generate a personal information input form, insert a tracking pixel, or redirect the victim. Details of each HTML file are shown in the table below.

File Name	보안 검사.html	보안프로그램 확인.html
MD5	be978477fe7c179cb9607a6e08a05dff	8833a270ddef0f464d5916958b6778e6
Masqueraded Org	National Health Insurance Service	DB Insurance
Identifier	gateless	gateless

User-Agent Mobile Detection	O	X
Polling Interval	3 seconds	5 seconds
Polling URL	hxxps://pipeline[.]embeddedonline[.]org/check.php?x-csrf-token=gateless	hxxps://pipeline[.]embeddedonline[.]org/check.php?x-csrf-token=gateless
Tracking Pixel URL	hxxps://appview[.]imagetemplate[.]com/gateless_icon<Counter>.png	hxxps://appview[.]imagetemplate[.]com/gateless_icon<Counter
Functionality	Tracking pixel insertion	

Additional security software spoofing page comparison

Both HTML files display a loading screen with the message "보안프로그램 검사 중"("Checking security software") upon access.

 New security software check spoofing page 1

New security software check spoofing page 1

 New security software check spoofing page 2

New security software check spoofing page 2

보안프로그램 확인.html communicates with a local server via JSONP to verify malware execution status. The page registers a unique global callback function in the format "vp20_" + Date.now() on the window object, then injects a <script> tag into the DOM that sends a request to localhost:16106 with this function as the callback parameter.

 Callback registration routine

Callback registration routine

Since browsers do not enforce the Same-Origin Policy (SOP) on <script> tags, the GET request reaches the local server set up by the malware. When the response executes as a script, the pre-registered callback is invoked and inspects the value of the returned object. A value of 0 indicates that the malware is running. We have dubbed this technique "JSONPing."

 JSONPing request routine

JSONPing request routine

If the JSONPing check determines that the malware is not running, the page displays an installation prompt. When the victim clicks the confirm button, the downloadProgram function is called to trigger a malware download. However, the malware download URL was inactive at the time of analysis, so the subsequent attack chain could not be recovered.

 Security software installation dialog box

Security software installation dialog box

5. Additional Malware - Spear Phishing

During our analysis, we obtained four Kimsuky dropper samples (scr, exe, jse) believed to have been used in spear phishing operations. All of these droppers display an embedded decoy document upon execution before launching a downloader. Several samples used internal documents from South Korean companies as decoys, strongly suggesting that corporate personnel were targeted via spear phishing. Details of the additional dropper samples are shown in the table below.

File Name	MD5	Downloader Compile Time	Download URL	Final Payload
Screenshot 2026-02-24 085012.scr	d09c0744273355b6da719fdb62923bed	February 20, 2026, 00:18:02 UTC	hxxps://load[.]serverpit[.]com/fwrite.php	Unknown

Unknown (decoy document: 엑셀데이터베이스활용설문지_성명.hwp)	f57a9e973e1cecd6b361467041e464f4	March 4, 2026, 02:17:26 UTC	hxxps://load[.]erasescloud[.]n-e[.]kr/login.php	Unknown
2026-0146-에스큐아이소프트-망연계시스템 유지보수 발주.pdf.jse	1efaf988fded55cd3b974c66f4ca8f7e	April 6, 2026, 06:26:12 UTC	hxxps://bigfile[.]jaycloudlab[.]com/download.php?id=745896	HttpSpy
420109_최종본_수정요청2차_반영.hwp.jse	bd8e948a6e61436532cd2ed2b62db3f3	April 20, 2026, 13:45:20 UTC	hxxps://download[.]birdriver[.]org/download.php?id=393156	HttpSpy

Spear phishing dropper samples

Among these, 420109_최종본_수정요청2차_반영.hwp.jse and 2026-0146-에스큐아이소프트-망연계시스템 유지보수 발주.pdf.jse were confirmed to deliver the exact same HttpSpy variant observed in the Webex spoofing case. The document metadata of the decoy file 420109_최종본_수정요청2차_반영.hwp records the last saved-by user as jira, matching the username found in the PDB path of mTSTCv8.mdxm analyzed in the Webex spoofing case.

Last saved by	Created	Modified
jira	Feb 21, 2026 03:27:45 UTC	Apr 19, 2026 20:57:40 UTC

Decoy document metadata

The engine.dat downloaded by 420109_최종본_수정요청2차_반영.hwp.jse is an identical file to the installer from the Webex spoofing case. The cache.db downloaded by 2026-0146-에스큐아이소프트-망연계시스템 유지보수 발주.pdf.jse is a different installer that ultimately executes the same HttpSpy main module.

File Name	Downloaded payload	Download URL	RC4 key
2026-0146-에스큐아이소프트-망연계시스템 유지보수 발주.pdf.jse	cache.db	hxxps://bigfile[.]jaycloudlab[.]com/download.php?id=745896	#RsfsetraW#@EsfesgsgAJOPj4eml;RGdcsefd@#%dg9ser3\$#\$^@34sdfxl
420109_최종본_수정요청2차_반영.hwp.jse	engine.dat	hxxps://download[.]birdriver[.]org/download.php?id=393156	%^fseRW#r3qwrwfsddRefGEgse)(14);RGdcsefd@#%dg9ser3\$#\$^@34sdfxl

Additional HttpSpy delivery dropper comparison

6. Attribution

6.1. Code Similarities

6.1.1. Obfuscated jse Loader

The attacker used obfuscated jse droppers to drop and execute decoy documents and malware. This type of jse loader is a well-documented tool that Kimsuky has consistently employed over time. A jse script used in an attack [reported by ESTSecurity in 2021](#), attributed to Thallium (a threat actor considered synonymous with or closely linked to Kimsuky), exhibits the same base64 decoding routine, drops to C:\ProgramData, and execute DLLs via regsvr32.exe.

 jse dropper used in the Webex spoofing case
 jse dropper used in the Webex spoofing case

 jse dropper used in the 2021 Thallium case (Source: <https://blog.alyac.co.kr/3754>)
 jse dropper used in the 2021 Thallium case (Source: <https://blog.alyac.co.kr/3754>)

6.1.2. Malware Execution Flow

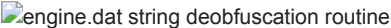
The dropper from the security software spoofing case and spyLoader.dll from the Webex spoofing case both reuse the following RC4 key for decrypting subsequent payloads. This same key was previously used to decrypt HttpTroy and additional payloads in a case [reported by Gen Digital in 2025](#).

Payload decryption RC4 key: #RsfsetraW#@EsfesgsgAJ0Pj4eml;

The following RC4 key, used by both analyzed spyInster.dll variants to decrypt configuration data, also matches the key from the May 2025 HttpSpy case.

Configuration data decryption RC4 key: RGdcsefd@#%dg9ser3\$#\$^@34sdfx1

Additionally, the XOR-based string deobfuscation algorithm and API hashing technique observed in engine.dat are identical to those found in the HttpTroy case.

 engine.dat string deobfuscation routine
engine.dat string deobfuscation routine

 HttpTroy string deobfuscation routine (Source: <https://www.gendigital.com/blog/insights/research/dprk-kimsuky-lazarus-analysis>)
HttpTroy string deobfuscation routine (Source: <https://www.gendigital.com/blog/insights/research/dprk-kimsuky-lazarus-analysis>)

Finally, the hello export function name was found in both the additional payload from the security software spoofing case and the HttpSpy main module from the Webex spoofing case. This is a common trait shared by previously distributed HttpTroy malware and Kimsuky malware documented in [Kaspersky's HelloDoor report](#).

6.2. Infrastructure Overlaps

6.2.1. HTTPS Certificate and ASN

The ENKI Whitehat Threat Research Team has been tracking a pattern in which Kimsuky repeatedly uses the default XAMPP HTTPS certificate across its infrastructure. The same certificate has been observed on Appleseed and HttpTroy C&C servers, phishing pages, and malware distribution servers. All infrastructure analyzed in this report used the same certificate. Details are shown in the table below.

Fingerprint	Subject DN	Names
9de541b039cfdb96c7810df49efd958b28cc2df73e314f67c1a91469a2b19796	C=DE, ST=Berlin, L=Berlin, O=Apache Friends, CN=localhost	localhost

Default XAMPP HTTPS certificate used across Kimsuky infrastructure

On the IP side, we also observed overlap with ASNs favored by Kimsuky. All C&C server IPs in this report, with the exception of 27.102.113[.]106, belong to the following two ASNs that Kimsuky has repeatedly used.

ASN: 19318, 26666

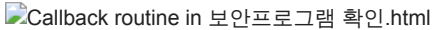
Neither the default XAMPP certificate nor the specific ASNs are unique to Kimsuky on their own, but their co-occurrence serves as a reliable indicator for identifying Kimsuky infrastructure.

6.2.2. Callback Verification Routine in Phishing Pages

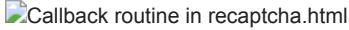
Since 2025, Kimsuky has actively incorporated localhost server requests and callback registration routines into its social engineering distribution techniques. recaptcha.html, uploaded to VirusTotal in November 2025, was hosted on 157.250.202[.]123, an IP we had been monitoring as known Kimsuky infrastructure. A comparison with the analyzed 보안프로그램 확인.html is shown in the table below.

File Name	MD5	Parameters	Port
recaptcha.html	02897faac6e41781152f480565e5d572	callback	62001
—	—	—	—
보안프로그램 확인.html	a2547836564b0732c6d02a78702da7e6	callback data	16106
—	—	—	—

JSONPing routine comparison



Callback routine in 보안프로그램 확인.html



Callback routine in recaptcha.html

Notably, `recaptcha.html` contains far more detailed Korean-language comments and debug logging code than typical malicious HTML. This strongly suggests that a Korean-speaking attacker made extensive use of an LLM to generate the HTML code.

7. Course of Action

7.1. Verifying URL Legitimacy

The attacker uses domains that closely resemble legitimate service domains or contain keywords from the same category to avoid raising suspicion. The domains used for initial access are shown in the table below.

Legitimate service domain	Phishing domain
ibizplus[.]co[.]kr	ibizplus[.]n-e[.]kr
meet1754245389211-9925[.]webex[.]com	conference[.]birdriver[.]org

Legitimate and phishing domain comparison

Kimsuky frequently uses domains issued through [내도메인.한국](#), a free Korean domain registration service. Extra caution is warranted when encountering the following domain suffixes.

- p-e[.]kr
- o-r[.]kr
- n-e[.]kr
- r-e[.]kr
- kro[.]kr

Also, legitimate services typically direct users to their official download pages when additional software installation is required. If an unofficial page demands immediate installation of security software or updates, users should always verify that the page's domain matches the service's official domain.



Webex official download center

7.2. Avoid Running Unverified Installers

Most executable file extensions (excluding `.lnk`) can be directly identified in File Explorer by enabling the "File name extensions" option. Users should always enable this option and carefully verify both the icon and extension of any file downloaded from the web or network.

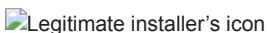


jse file as shown in Windows File Explorer

The security software spoofing malware analyzed in this report used the same icons and file extensions as their legitimate counterparts, making it nearly impossible to tell them apart by appearance. Before running any file downloaded from the web, users should scan it with antivirus software, check it on VirusTotal, or verify its integrity through the vendor's official channel.



Malicious program's icon



Legitimate installer's icon

8. Conclusion

This report examined a series of Kimsuky operations directed at South Korean military and enterprises identified through April 2026. Kimsuky went beyond simple malware distribution, introducing sophisticated mechanisms to maximize delivery success, including real-time infection verification via JSONPing and crafting a fake page using stolen meeting schedule.

On the malware side, we observed structural changes, including the separation of the HttpSpy execution flow into three stages (Installer, Loader, Main Module) and the adoption of obfuscated jse droppers in place of Go-based .scr droppers. Despite these changes, key indicators remained consistent with previous campaigns: RC4 keys, the hello export function name, XOR-based string obfuscation, and HTTPS certificates. These persistent traits provide a reliable basis for tracking Kimsuky, even as the threat actor's TTPs continue to evolve.

The ENKI Whitehat Threat Research Team will continue to track major threat actors including Kimsuky and deliver actionable threat intelligence.

9. Appendix

Appendix A. MITRE ATT&CK

Tactics	Techniques
Initial Access	T1566: Phishing
Execution	T1059.003: Command and Scripting Interpreter: Windows Command Shell T1059.007: Command and Scripting Interpreter: JavaScript T1204.002: User Execution: Malicious File
Persistence	T1547.001: Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder T1053.005: Scheduled Task/Job: Scheduled Task
Privilege Escalation	T1134.002: Access Token Manipulation: Create Process with Token
Defense Evasion	T1027.009: Obfuscated Files or Information: Embedded Payloads T1027.010: Obfuscated Files or Information: Command Obfuscation T1027.013: Obfuscated Files or Information: Encrypted/Encoded File T1036.004: Masquerading: Masquerade Task or Service T1140: Deobfuscate/Decode Files or Information T1620: Reflective Code Loading T1497.001: Virtualization/Sandbox Evasion: System Checks T1055.001: Process Injection: Dynamic-link Library Injection T1070.004: Indicator Removal: File Deletion T1070.006: Indicator Removal: Timestomp T1070.009: Indicator Removal: Clear Persistence
Discovery	T1057: Process Discovery T1010: Application Window Discovery T1012: Query Registry T1082: System Information Discovery T1083: File and Directory Discovery
Collection	T1113: Screen Capture T1005: Data from Local System
Command and Control	T1071.001: Application Layer Protocol: Web Protocols T1132.001: Data Encoding: Standard Encoding T1573.001: Encrypted Channel: Symmetric Cryptography T1090: Proxy
Exfiltration	T1041: Exfiltration Over C2 Channel

MITRE ATT&CK

Appendix B. IOCs

MD5

MD5	File Name
4a476abcf741323b367eda0ec49f8c38	nos-setup.exe
00f957b7dafd8d210e717041add02eab	nos_setup.exe
cc837d2b2af4bd9c1c3faf61cefeb848	astx-setup.exe
97b4c2e67e5e18b70949690a69820c2a	astx_setup.exe
bea602695d58cbf25fff058834e36c1d	MemLoader.dll
ea5f32e1273ec93d43ee09a337fb60e1	calc.exe
0d07fb6d1a3736ea543ab8364115e435	install.html
6d2dfd7ca77530afec000a197d6b8677	fix-camera.zip
b4dd4c76d7deef4cf532e240b7f84c9d	fix-camera.jse
be31a38bab026f229afd5e3174c363f7	meeting.html
c61a6efe1a169c6c1d8595af3ff0dd74	mTSTCv8.mdxm (loadDll.dll)
c6de1be41dcfbad9cae76c58eae7f5a3	engine.dat (spyInster.dll)
50f619aaba1d28882022ced135b13a07	cacheMon.dat (spyLoader.dll)
a581fdea0970f8a5b6cfec4853c802d7	httpSpy.dll
9df5ca76ac085b89c1ddcb3963e9fe97	Webex Phishing Page
be978477fe7c179cb9607a6e08a05dff	Security Check.html
8833a270ddef0f464d5916958b6778e6	Verify Security Program.html
d09c0744273355b6da719fdb62923bed	Screenshot 2026-02-24 085012.scr
fcaf03060e34a73fe499b906492d9f13	Screenshot 2026-02-24 085012.png
dd47c97b44408e0a5ecd8f482fcd0dbc	MemLoader.dll (png)
f57a9e973e1cecd6b361467041e464f4	Malware using hwp decoy file
3369b911cf3706a2660d2af9b3c35f9a	ExcelDatabaseUtilizationSurvey_Name.hwp
39e091e981d9daab56e680927508bd1f	MemLoader.dll (hwp)
00fd7272f9a3044b5f62680d9e576e55	420109_Final_Modification_Request_2nd_Reflected.egg
bd8e948a6e61436532cd2ed2b62db3f3	420109_Final_Modification_Request_2nd_Reflected.jse
3315229011b2fa2b05bd4c7b4fbb58e3	d5HaQHe.bt6L
a87cd5fd8fe223816005e81e0da70b21	420109_Final_Modification_Request_2nd_Reflected.hwp
c05f074c70a6cacb0e6f05578aab3c9d	2026-0146-SQISoft-Network_Integration_System_Maintenance_Order.egg
1efaf988fdd55cd3b974c66f4ca8f7e	2026-0146-SQISoft-Network_Integration_System_Maintenance_Order.pdf.jse
50e89a4e50392e4235822e9e92df4c32	h3ZYBzb.dbei
7b9484d719f39faa71abd90f57525cc8	2026-0146-SQISoft-Network_Integration_System_Maintenance_Order.pdf

91d1a7153606dedef92502553962cf67	cache.db (spyInster.dll)
02897faac6e41781152f480565e5d572	recaptcha.html

MD5

IP

- 163.245.221[.]218
- 163.245.215[.]46
- 27.102.113[.]106
- 157.250.202[.]123

URL

- hxxps://www.ibizplus.n-e[.]kr/install.html
- hxxps://load.serverpit[.]com/fwrite.php
- hxxps://load[.]erasescloud[.]n-e[.]kr/login.php
- hxxps://conference.birdriver[.]org/
- hxxps://download.birdriver[.]org/download.php?id=425623
- hxxps://download[.]birdriver[.]org/download.php?id=393156
- hxxp://hdrgrfes[.]chickenkiller[.]com/index.php
- hxxps://pipeline[.]embeddedonline[.]org/check.php?x-csrf-token=gateless
- hxxps://appview[.]imagetemplate[.]com/gateless_icon<Counter>.png
- hxxps://pipeline[.]embeddedonline[.]org/download3.php?sessid=54126&user-token=babymetalsave
- hxxps://pipeline[.]embeddedonline[.]org/check.php?x-csrf-token=babymetalsave
- hxxps://appview[.]imagetemplate[.]com/babymetalsave_icon<Counter>.png
- hxxps://meet1754245389211-9925[.]webex[.]com/meet1754245389211-9925/j.php?MTID=mb755b0b9133ae8f9e3608b0b519d6a35
- hxxps://bigfile[.]crabdance[.]com/recaptcha.html

PDB Path

C:\Users\jira\Documents\My_Received_Files\loadDII\x64\Release\loadDII.pdb

User-Agent

Mozilla / 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit / 537.36 (KHTML, like Gecko) Chrome / 120.0.0.0 Safari / 537.36

Appendix C. Metadata

RC4 Key

- #RsfsetraW#@EsfesgsgAJOPj4eml;
- %^fseRW#r3qwrwfsddREfGEgse)(14);
- RGdcsedfd@#%dg9ser3\$#\$^@34sdfxl

Username

jira

Appendix D. Analysis tool detection list

Process Names

List	List
ollydbg	ProcessHacker
tcpview	autoruns
filemon	procmon
regmon	procexp
idaq	ImmunityDebugger
Wireshark	dumpcap
HookExplorer	ImportREC
PETools	LordPE
SysInspector	proc_analyzer
sysAnalyzer	sniff_hit
windbg	joeboxcontrol
joeboxserver	x32dbg
x64dbg	Fiddler
httpdebugger	cheatengine
frida-helper	systemexplorer
dbgview	vmtoolsd
vm3dservice	

Process Name

Window Name

List	List
debugview	sysinternals
ida	Process Hacker
Immunity Debugger	

Window Name



ENKI Whitehat

ENKI Whitehat

Offensive security experts delivering deeper security through an attacker's perspective.