Securonix Threat Research Security Advisory: Analysis of New DEEP#GOSU Attack Campaign Likely Associated with North Korean Kimsuky Targeting Victims with Stealthy Malware

By Securonix Threat Research: D. Iuzvyk, T. Peck, O. Kolesnikov

tldr:

The Securonix Threat Research team has uncovered an elaborate multi-stage attack campaign likely associated with the North Korean Kimsuky group.

The Securonix Threat Research (STR) team has been monitoring a new campaign tracked as DEEP#GOSU likely associated with the Kimsuky group, which features some new code/stagers as well as some recycled code and TTPs that were reported in the past. While the targeting of South Korean victims by the Kimsuky group happened before, from the tradecraft observed it's apparent that the group has shifted to using a new script-based attack chain that leverages multiple PowerShell and VBScript stagers to quietly infect systems. The later-stage scripts allow the attackers to monitor clipboard, keystroke, and other session activity.

The threat actors also employed a remote access trojan (RAT) software to allow for full control over the infected hosts, while the background scripts continued to provide persistence and monitoring capabilities.

All of the C2 communication is handled through legitimate services such as Dropbox or Google Docs allowing the malware to blend undetected into regular network traffic. Since these payloads were pulled from remote sources like Dropbox, it allowed the malware maintainers to dynamically update its functionalities or deploy additional modules without direct interaction with the system .

The malware used in the DEEP#GOSU campaign likely enters the system through typical means where the user downloads a malicious email attachment containing a zip file with a single disguised file using the extension: pdf.lnk, (IMG_20240214_0001.pdf.lnk) in this case.

Stage 1: Initial execution: LNK files [T1204.002]

The use of shortcut files, or .lnk files by threat actors is nothing new. However, in the case of DEEP#GOSU, the methodology behind the code execution is quite different from what we have typically seen in the past.

First, as seen in the figure below, the length of the command is quite impressive and it's clear that the executed PowerShell is designed to perform several complex functions. Additionally, standing at about 2.2MB, it's clear that there is more to this shortcut file than what meets the eye.

Figure 1: IMG_20240214_0001.pdf.lnk – command line execution

The embedded PowerShell script contained within the shortcut file is designed to take byte data from itself, which extracts embedded files, AESDecrypt and executes further malicious code downloaded from the internet (/step2/ps.bin) and clean up traces of its execution. The use of encryption and cloud services for payload retrieval indicates some level of sophistication intended to evade detection and analysis. This type of infrastructure typically takes much more time to set up and maintain versus simply hosting files on rented servers.

Let's first analyze the reason that the shortcut file is over 2MB in size. Upon close analysis, the shortcut file appears to have an entire embedded PDF concatenated to it after tens of thousands of "A" characters. Those characters may be an attempt to pad the size of the file to evade AV detections.

The figure below demonstrates how this looks when using a hexadecimal editor to view the file's raw data. On the left, we can see the end of the shortcut code which calls cmd.exe (to eventually call powershell.exe) and the start of the sequence of "A" characters. Over on the left the A's terminate and the start of a PDF header appears!

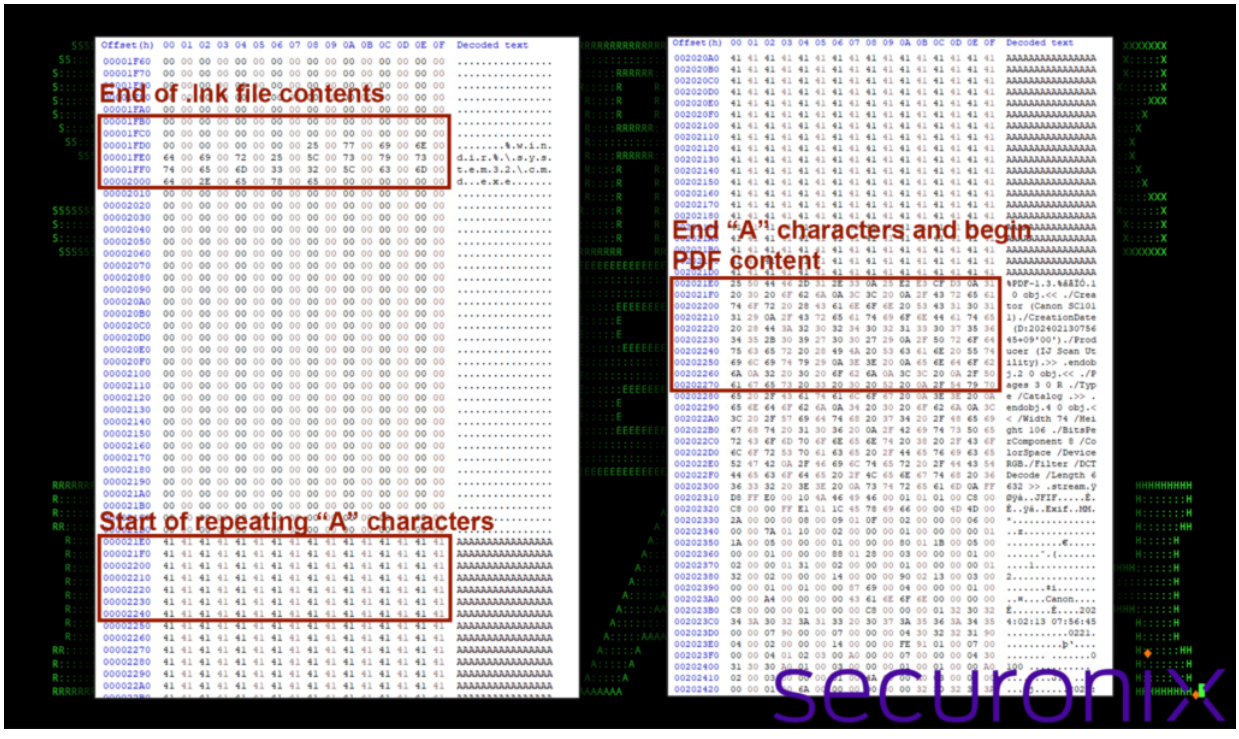


Figure 2: Hex bytes of IMG_20240214_0001.pdf.lnk highlighting the embedded PDF file

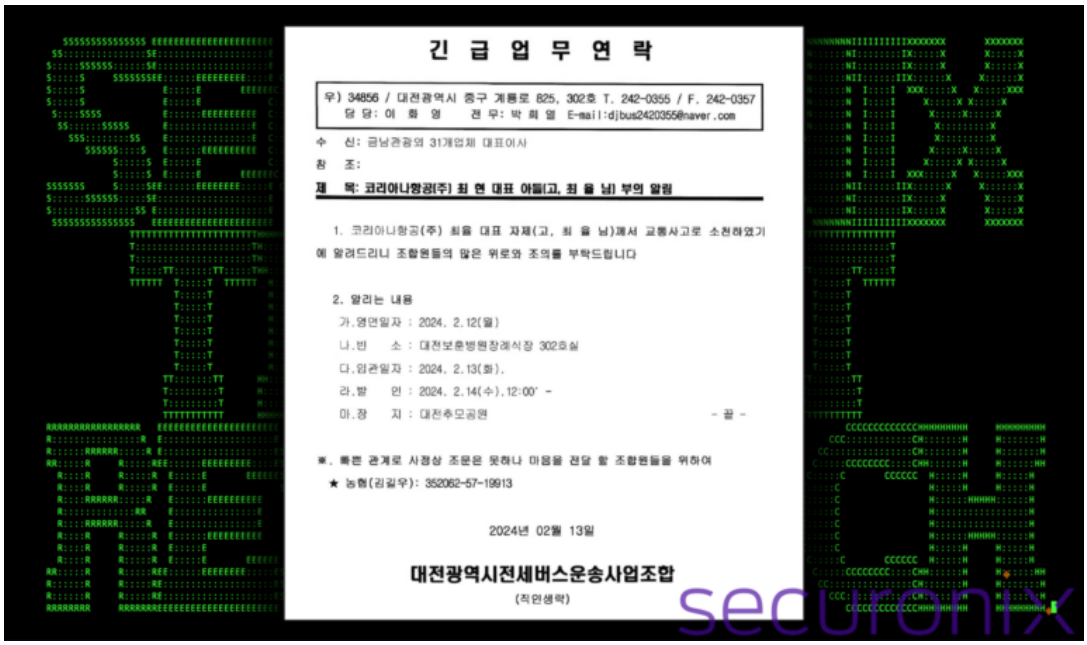
So, the shortcut file has a concatenated PDF file attached to it. The PowerShell code contains a clever function that performs a few tasks. The PowerShell code below is taken from the code from within the shortcut file (figure 1) and then cleaned up a bit so it's easier to read:

Figure 3: IMG_20240214_0001.pdf.lnk – extract PDF portion from itself

- This portion of the script extracts the PDF portion of the .lnk file's content based on specific byte positions which exists between byte values 2105824 and 2282653 ($len1 to $len2).
- The script writes out the progress at each operational task such as "readfileend", "exestart" and "exeend".
- The alias "sc" is used to instantiate a new object to hold the PDF file.
- This extracted content is then eventually saved to a new variable $path, and then executed using the PowerShell Start-Process commandlet.
- The PDF content is then executed which will then open in the system's default PDF viewer which opens as "IMG_20240214_0001.pdf".
- All files are then deleted.

What makes this tactic clever is that there is technically no PDF file contained within the initial zip file sent to the victim. When the user clicks the PDF lure (shortcut file) they're immediately presented with a PDF file thus removing any concern that anything unexpected happened.

The PDF lure document is in Korean and appears to be an announcement regarding the son of Koreana Air CEO Choi Hyun (the late Choi Yul) and states that the son has passed away due to a car accident. The rest contains details and dates of the funeral hall.



긴 급 업 무 연 락

우) 34856 / 대전광역시 중구 계룡로 825, 302호 T. 242-0355 / F. 242-0357
담 당: 이 화 영 전 무: 박 희 열 E-mail:djbus2420355@naver.com

수 신: 금남관광의 31개업체 대표이사

참 조:

제 목: 코리아니항공(주) 최 현 대표 아들(고. 최 을 남) 부의 알림

1. 코리아니항공(주) 최을 대표 자제(고. 최 을 남)께서 교통사고로 소천하였기에 알려드리니 조합원들의 많은 위로와 조의를 부탁드립니다

2. 알리는 내용

가.영면일자 : 2024. 2.12(월)

나.빈 소 : 대전보훈병원장례식장 302호실

다.입관일자 : 2024. 2.13(화).

라.발 인 : 2024. 2.14(수).12:00' -

마.장 지 : 대전추모공원 - 끝 -

※. 빠른 근계로 사정상 조문은 못하나 마음을 전달 할 조합원들을 위하여

★ 농협(김길우): 352062-57-19913

2024년 02월 13일

대전광역시전세버스운송사업조합

(직인생략)

Figure 4: IMG_20240214_0001.pdf lure document

In addition to extracting and executing the PDF document, the shortcut file also executes the malware's next stage payload from a Dropbox URL (hxxps://content.dropboxapi[.]com/2/files/download/step2/ps.bin). Despite its name, the ps.bin file is actually another PowerShell script which we'll dive into later.

Since Dropbox requires authentication, all of the required parameters are embedded into the shortcut's original PowerShell script (figure 1). With the PowerShell code cleaned up, the portion of the script responsible for downloading and executing the next-stage payload ($newString) is highlighted below.

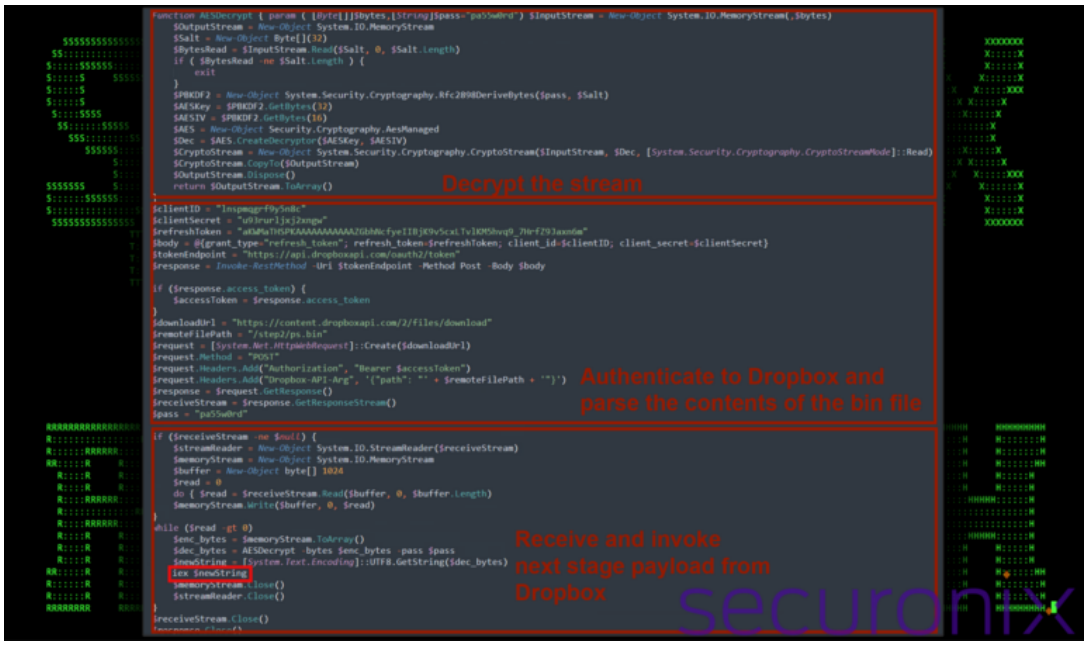


Figure 5: IMG_20240214_0001.lnk – download and invoke next-stage payload from Dropbox

To sum up, the PowerShell script contained with the shortcut file is designed to silently find and execute the specifically crafted malicious .lnk file (itself), extract and execute the embedded PDF lure document, authenticate, decrypt and execute further malicious code downloaded from Dropbox, and then clean up traces of its execution. The use of encryption and cloud services for payload retrieval indicates a level of sophistication intended to evade detection and analysis.

Stage 2: Invoked code from Dropbox [T1102]

At this stage, the initial shortcut file has downloaded and invoked a remote payload from Dropbox called ps3.bin. The PowerShell code contained within the .bin file defines a function (Load) that performs several operations which includes downloading, decompressing, and dynamically loading and executing .NET assembly code from a different Dropbox URL.

- Define a decompression helper function (GzExtract):
  - This inner function takes a byte array as input in the form of GZIP compressed data.
  - Decompresses this data and return the resulting byte array
- Dynamically loading .NET assemblies:
  - The script dynamically loads assemblies related to System.Drawing, System.Windows.Forms, and PresentationCore
  - This enables the script to use advanced graphical UI capabilities which have been used in the past for features such as screenshots or screen recording by Dark Pink malware among others.
- Authenticating with Dropbox and downloading next-stage remote payload:
  - Similar to the shortcut file's PowerShell script, it authenticates with Dropbox once again using a refresh token, client ID, and client secret to obtain an access token.
  - A file named r_enc.bin is downloaded from Dropbox (stage 3).
  - After downloading the file, it attempts to decompress the payload using the GzExtract function defined earlier.
  - The script implies this payload is a .NET assembly in binary form, though compressed to evade detection.
- Dynamically loading and executing the .NET assembly:
  - It loads the decompressed .NET assembly into memory without writing it to disk which can help cut down AV detections.
  - It iterates through types and methods within the loaded assembly to find and invoke a specific method (makeProbe1). The invocation is commented out, but it suggests that the method would execute with a hardcoded parameter, which is partially shown and then truncated.
  - This dynamic loading and execution allow the malware to perform virtually any action the .NET framework supports, based on the code within the downloaded assembly.

Figure 6: Example of PowerShell ps.bin

In addition to the above the script also invokes a method on an object instance using reflection in PowerShell, with a parameter that appears to be a Base64-encoded string. The string can be seen in the figure below.

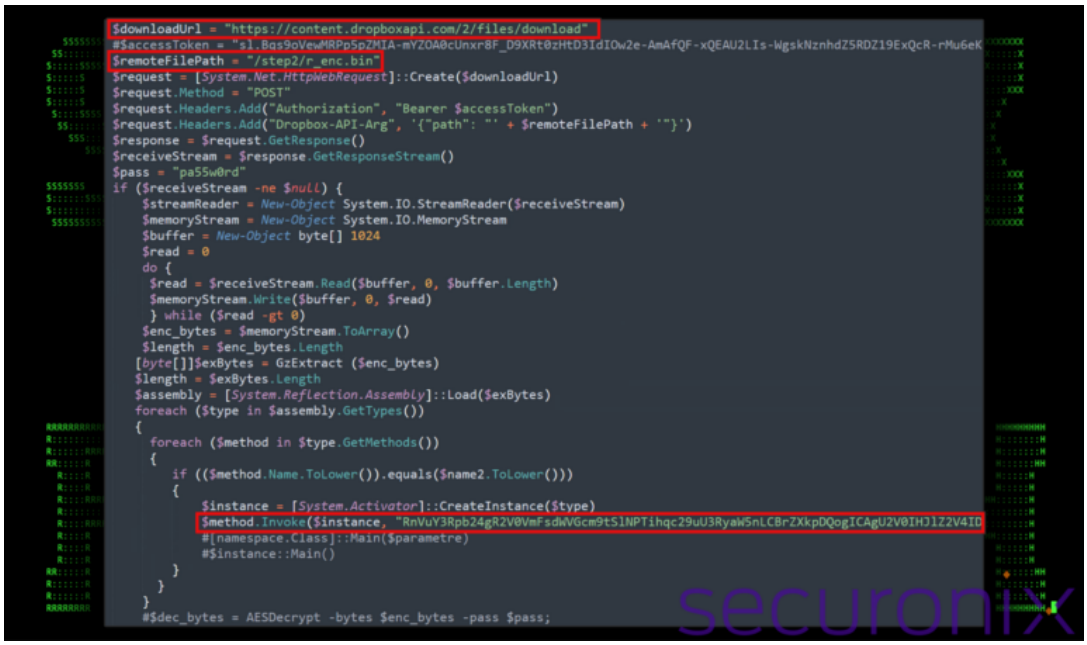


Figure 7: ps.bin PowerShell invokes next stage payloads

The $method variable is set up and holds a reference to a “MethodInfo” object, which represents a specific method of a class. The “$instance” variable contains the instance of the class which in turn contains the method you want to invoke. The string is encoded in Base64 and then passed as an argument to the method.

Since at this point the code is doing two pretty interesting things simultaneously, let’s follow the loading and execution of “r_enc.bin” from Dropbox further down (Stage 4) which is loaded from the following Dropbox URL:

hxxps://content.dropboxapi[.]com/2/files/download/step2/r_enc.bin

We’ll continue with the invocation of the new Base64 encoded method (Stage 4) further down.

Stage 3: TutClient [C# RAT] (r_enc.bin)

When analyzing the PowerShell script in Stage 2 we determined that the script once again reached back out to Dropbox and downloaded a compressed Base64 string. The file itself is indeed a binary file which can be easily confirmed using a tool such as CyberChef. If we place the large Base64 string (r_enc.bin) inside the input field, select “From Base64” and “Gunzip”, we see the MZ header and other common strings for Windows executables inside the output.

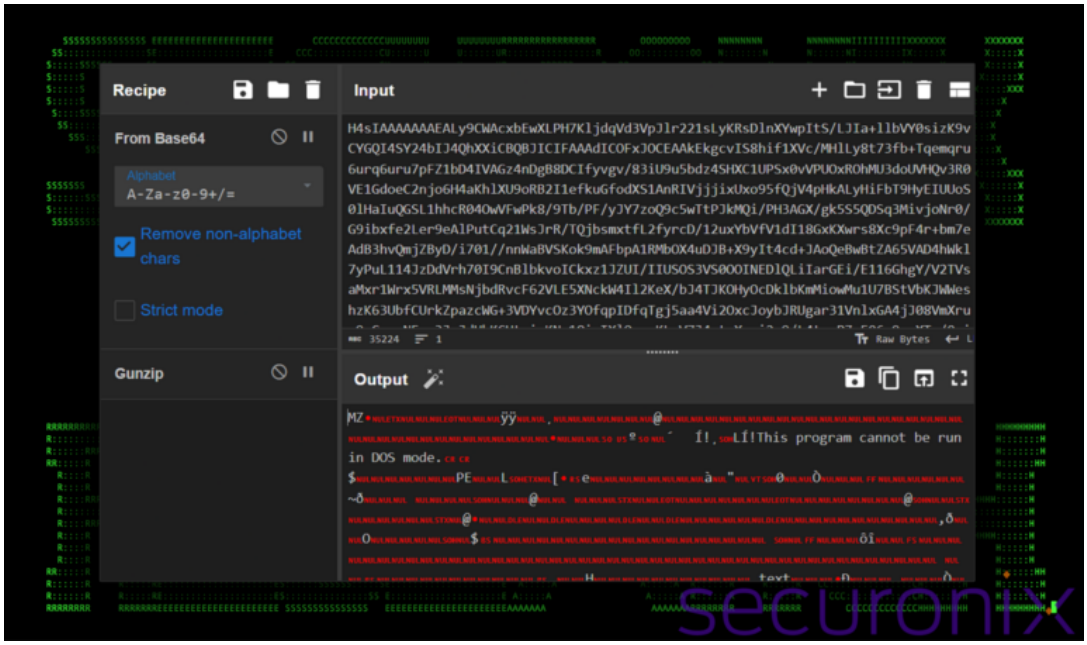


Figure 8: Decoding r_enc.bin in CyberChef

The decompressed binary file ends up being an open source RAT (remote access trojan), known as TruRat, TutRat or C# R.A.T. which generates a commonly named client called TutClient.exe.

As the name suggests, the RAT is coded in C# and is open source. Since the source of the application can be found online, we won't go too deep into the binary code analysis portion as it's available online, but rather discuss its capabilities.

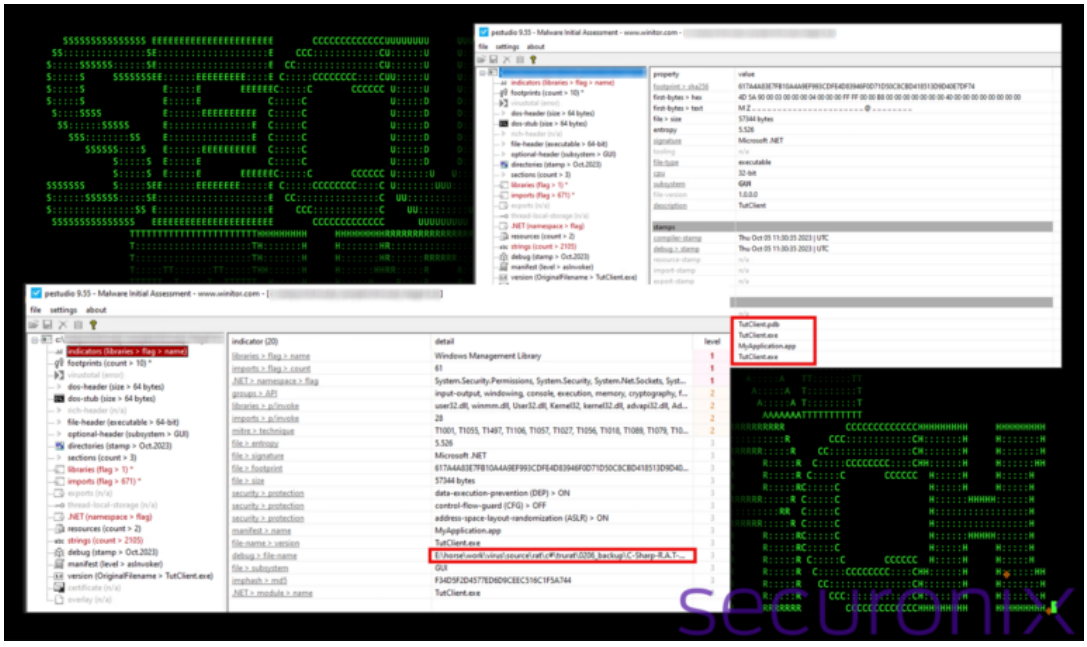


Figure 9: C# RAT executable client overview

Currently this particular RAT software is quite old and likely to be picked up by most antivirus vendors. However, given the unique method in which this binary is loaded and executed directly into memory (stage2), it's likely to skirt some detections.

Execution of the payload in memory, also known as "fileless" execution, is a technique used by attackers to evade detection by traditional file-based antivirus solutions. Since the payload does not touch the disk, it leaves fewer traces, making it harder for security tools to detect and mitigate the threat.

According to the C# Rat's GitHub page, the malware supports a wide range of features including:

- Keylogger
- Remote desktop
- Mic and cam spy
- Remote Cmd prompt
- Process and file manager
- Fun menu (hiding desktop icons, clock, taskbar, showing messagebox, triggering Windows sound effects)
- DDoS with target validation
- Password manager (supporting: Internet Explorer, Google Chrome, Firefox)

Interestingly enough, this is not the first time that we've seen this RAT used against Korean targets. A year ago the Kimsuky group was identified delivering TutRAT and xRAT payloads through other methods.

Stage 4: VBScript execution (invoked code from stage 2) [T1059.005]

Circling back to Stage 2, if you recall, we observed a large Base64 encoded string getting invoked. After decoding the string we reveal a VBScript code segment which once again is designed to connect back to Dropbox by interacting with specific web APIs.

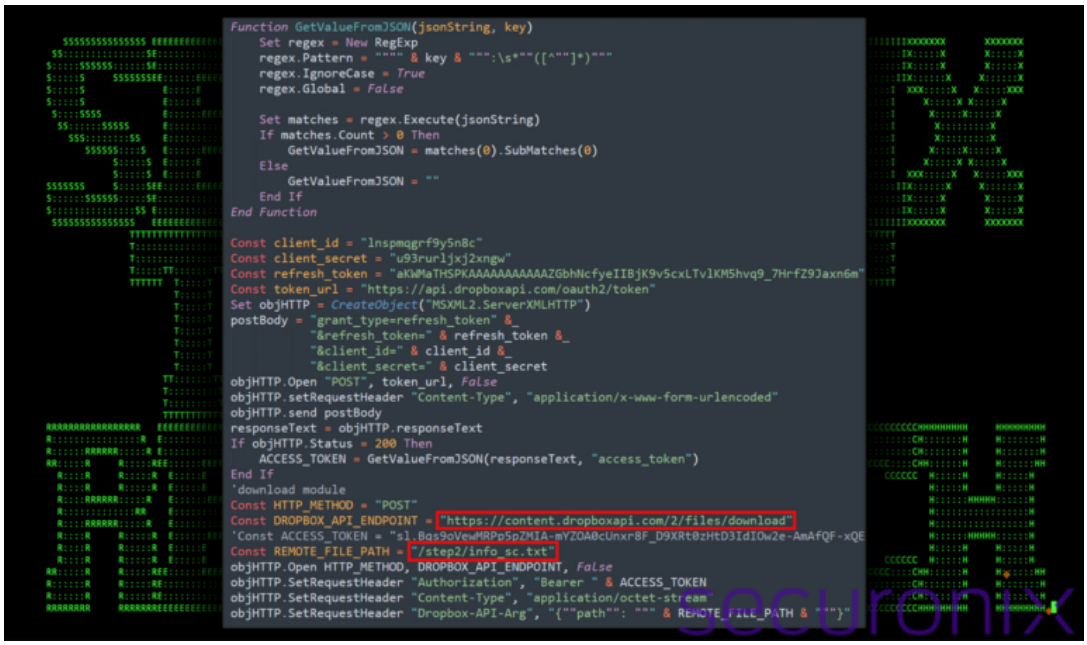


Figure 10: Stage 4 VBScript execution – download info_sc.txt from Dropbox (from stage 2)

The next stage is downloaded from Dropbox in the same manner we observed during the last several stages. Using a unique client ID, refresh token and secret, the file “info_sc.txt” is downloaded from the URL:

hxxps://content.dropboxapi[.]com/2/files/download/step2/info_sc.txt

Once the file is downloaded, it is written to a VB Stream object then switches the stream’s type to text and reads it as a UTF-8 encoded string. This is a method to convert binary data (the downloaded file content) into a readable string.

The crucial part of this script is the “Execute” statement, which executes the string read from the stream as VBScript code. This means the downloaded content is not just data but executable code, which makes the purpose for Stage 4 run arbitrary VBScript code fetched from Dropbox.

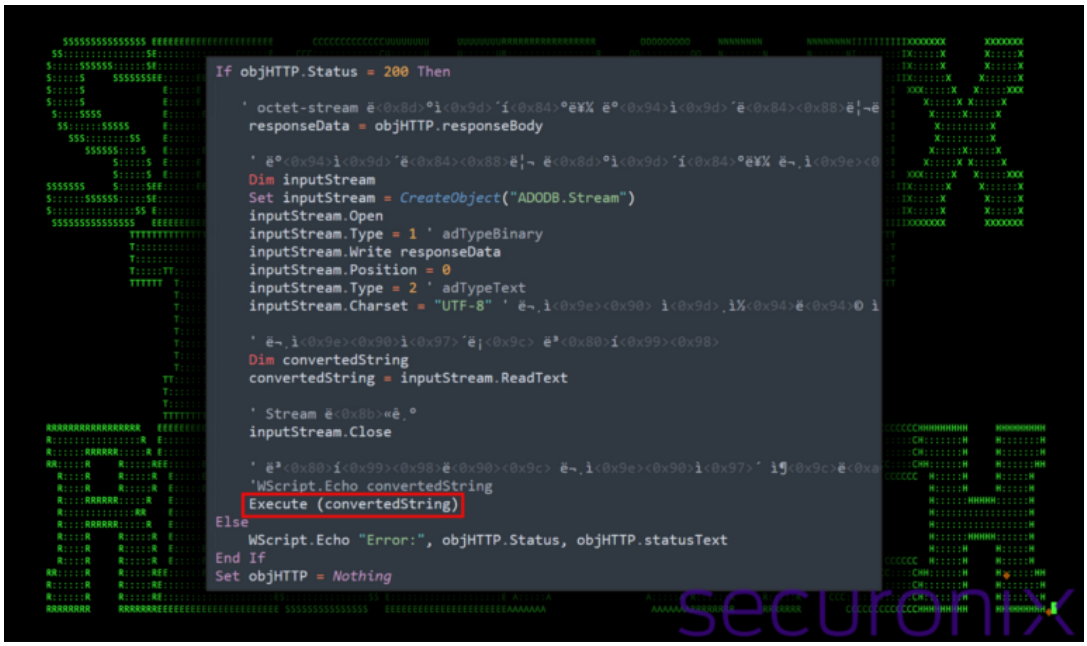


Figure 11: Stage 4 VBScript execution execute downloaded code

With the code downloaded from Dropbox, parsed and then converted, it’s placed inside “convertedString” and then executed.

Lastly, the script dynamically writes a PowerShell file on the disk and then executes it (Stage 7). This file was written to:

c:\users\[redacted]\appdata\roaming\microsoft\windows\w568232.ps1

Originally the script dropped the file named w568232.ps12x , however it was immediately renamed to w568232.ps1 using the following command:

cmd /c rename c:\users\[redacted]\appdata\roaming\microsoft\windows\w568232.ps12x w568232.ps1

Stage 5: VBScript execution (info_sc.txt) [T1059.005]

If you thought at this point we were done with Dropbox stages, you might be right, depending on the OS version the victim system is running. But for now, a closer look at this script reveals several indications of more traditional malware such as persistence indicators and WMI (Windows Management Instrumentation) activity.

The script is quite complex, though it did not feature any form of obfuscation which needed to be decoded. Let's go over some of the more interesting routines and functions to better understand its capabilities.

WMI Execution [T1047]

At the beginning of the script there is a WMProc Subroutine which uses WMI to execute commands on the system. It takes a single parameter p_cmd which specifies the executable or script that is launched by the WMI service.

Additionally, there is a commented out line with instructions to download, save and execute a remote .hwp document file. Kimsuky has been known to use disguised hwp files in the past, so this could be an artifact of an older attack chain. The commented out line references a remote server at regard.co[.]kr, however we did not observe any network communication to that domain throughout the course of the DEEP#GOSU campaign.

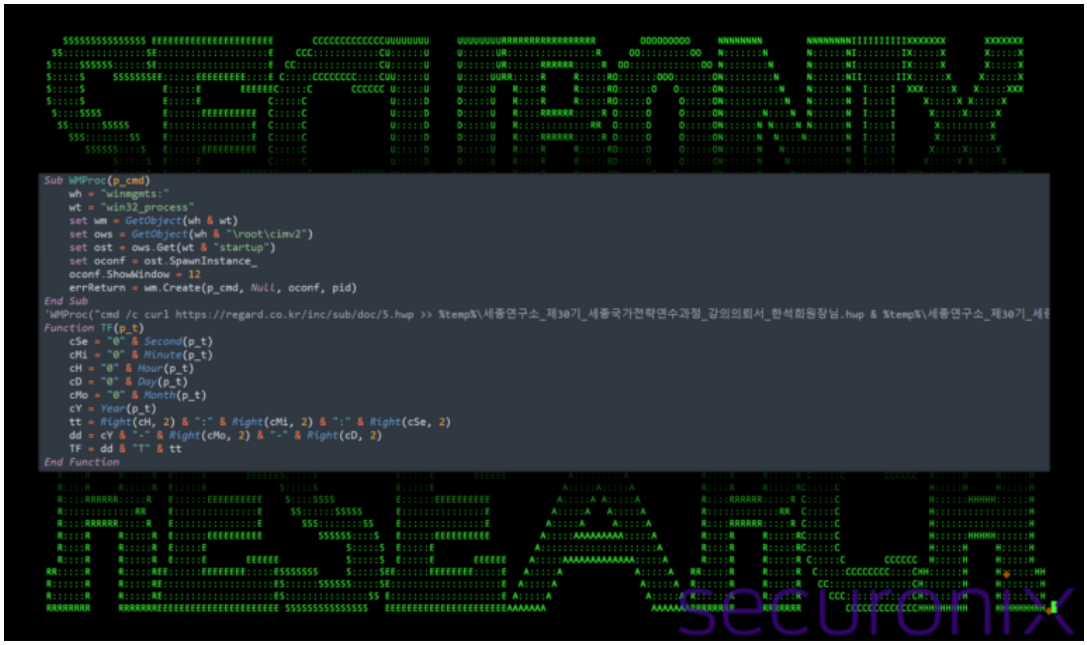


Figure 12: Stage 5 VBScript execution – WMProc and TF functions

Scheduled tasks [T1053]

The TF function works with the Reg and Reg1 subroutines which are used to schedule tasks on the system. Additionally, the TF function formats a timestamp for scheduling, and the Reg subroutine actually schedules a new task. This task is configured to execute a script or command at a later time, ensuring that the malware maintains persistence on the system.

Figure 13: Stage 5 VBScript execution – Reg and Reg1 functions

Remote payload download

At this point the script checks the version of the operating system and branches its behavior accordingly. For OS versions prior to Windows 10, it uses Internet Explorer functionality to download and execute a script fetched from a remote server at

hxxp://gbionet[.]com/inc/basl/up1/list.php?query=6

After contacting the URL above, the script captures the “innerText” of the page’s body, which is the text content of the response from the server, excluding any HTML tags.

For systems running Windows 10 or later, it uses a PowerShell script which is saved into a single VBScript variable to download and execute a payload from Dropbox using similar methods we witnessed prior.

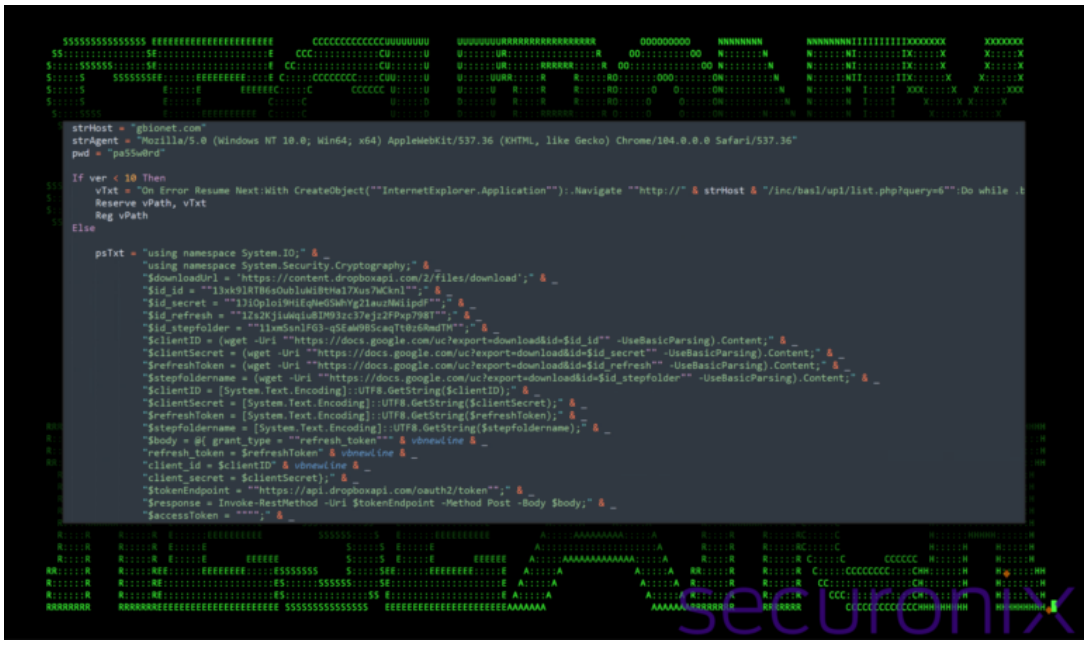


Figure 14: Stage 5 VBScript execution – Next stage download

The inclusion of Google Docs URLs in the PowerShell script encapsulated within the psTxt variable is a method used to dynamically retrieve configuration data for the Dropbox connection. This could be useful for when payloads, or

Dropbox account data needs to be changed, without having to change the script itself.

As we witnessed previously, the PowerShell script uses a hard-coded password (pa55w0rd), and then executes the decrypted content. This also helps reduce the malware's detection footprint. Using these types of services to fetch configuration data or payloads can blend in with legitimate network traffic, reducing the likelihood of network-based detection.

Figure 15: Stage 5 VBScript/PowerShell execution – invoke next stage

The decrypted content uses a predefined password and AES decryption. Since the downloaded content is encrypted another layer of protection against detection is added.

Interestingly enough, the $uh variable is not defined anywhere in the script. This is used by the Invoke-Command alias (icm) to execute a PowerShell scriptblock. This could be a mistake by the malware authors, or used in context with other more broad malware operations where it could be used with portions of code not included in the samples identified by the team.

Lastly, the decrypted content is then executed directly in memory using a PowerShell invoke-expression, which leads us into Stage 6!

Stage 6: PowerShell execution – system enumeration [T1082]

Circling back to PowerShell, the next script that gets executed is an interesting script which attempts to enumerate the victim system as much as it can. Once again, Dropbox is used, however rather than downloading the next-stage payload, it issues a carefully-crafted POST request to submit its enumeration findings.

As you can see in the data below, it formats the data into sections with headers containing plus signs on either side of the header text.

Figure 16: Stage 6 PowerShell system enumeration example

The script enumerates the following items:

- Running processes (tasklist)
- Firewall status for all profiles (Netsh Advfirewall show allprofiles)
- Registered antivirus products via Security Center (AntiVirusProduct class from ROOT\SecurityCenter and ROOT\SecurityCenter2 namespaces)
- User profile directories:
  - Desktop ($user_dir\Desktop)
  - Documents ($user_dir\Documents)
  - Downloads ($user_dir\Downloads)
- Application data and start menu programs:
  - Recent documents ($appdata\Microsoft\Windows\Recent)
  - Start Menu Programs ($appdata\Microsoft\Windows\Start Menu\Programs)
- Program files directories:
  - Default Program Files ($env:ProgramFiles)
  - Program Files (x86) for 64-bit systems ($env:ProgramFiles(x86))
- All drives and their content, including:
  - Drive label, type, format
  - Directories and files within each accessible drive

Once the information is gathered it encrypts the data using AES functions similar to that of the AES decrypt functions we discussed earlier. The script then constructs an HTTP POST request to upload encrypted data. The script attempts to refresh an OAuth token for Dropbox using a client ID, secret, and refresh token, then uses this token to authorize an upload to Dropbox.

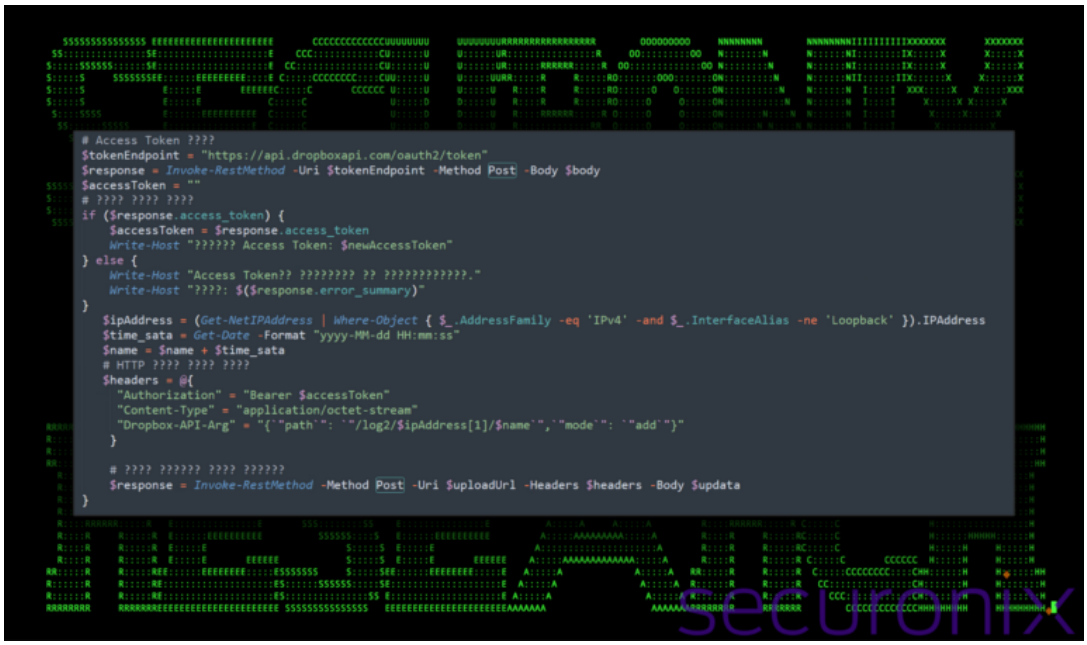


Figure 17: Stage 6 PowerShell upload enumeration data

Stage 7: stealth and persistence in PowerShell [T1041]

If you recall, this script is created and saved to the disk from Stage 5 (\appdata\roaming\microsoft\windows\w568232.ps1). The purpose of this script appears to be designed to serve as a tool for periodic communication with a command and control (C2) server via Dropbox. Its main purposes include encrypting and exfiltrating or downloading data.

Most of the script once again contains PowerShell code for handling Dropbox connections and AES encryption/decryptors however there are a few interesting functions worth mentioning.

Figure 18: stage 7 various functions inside w568232.ps1

To ensure persistent, stealthy operation, it contains unique functions for both mutex-based singleton execution ($bMute) and variable intervals for network connectivity (GetTimeInterval). The time is set to a random interval between 10000 seconds (2.78 hours). Essentially, the script acts as a versatile backdoor that allows attackers to continuously monitor and control their infected systems.

Stage 8: Keylogging [T1056.001]

The purpose of this (and final) script is to act as a keylogging and clipboard monitoring component to monitor and log user activity on the compromised system.

It achieves this by first obtaining access to Windows native APIs using .NET assemblies, and then using the Add-Type PowerShell module to call the Core class within the session. The script uses some targeted variable substitution obfuscation throughout the defined strings.

Figure 19: stage 8 obfuscated .NET assemblies

The script uses functions such as GetAsyncKeyState to monitor the state of individual keys on the keyboard, capturing key presses and releases.

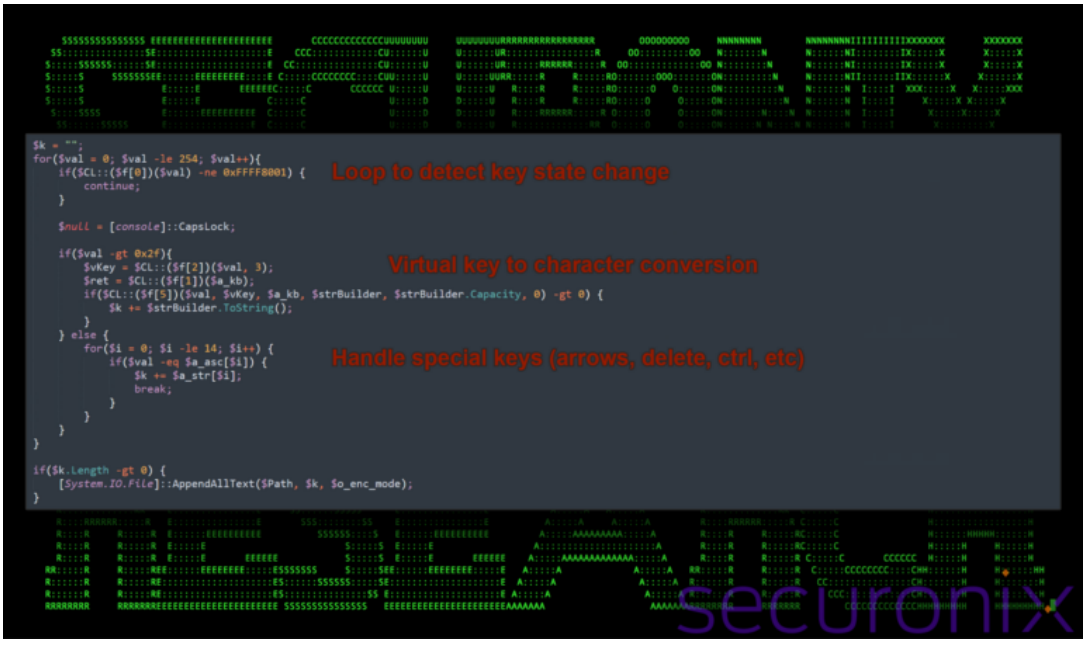


Figure 20: stage 8 PowerShell keylogging functions

The PowerShell script includes functionality to monitor and log changes in the clipboard content. It does this by using the GetClipboardSequenceNumber function to retrieve the current clipboard sequence number, which changes anytime the content of the clipboard changes.

It then compares the current clipboard sequence number in $curClip with the previously stored sequence number in $oldClip. If they differ, it indicates the clipboard content has changed. If the format is verified as “text” it then uses

[Windows.Clipboard]::GetText() to retrieve the new clipboard text.

Lastly, it appends the content into the $Path (Version.xml) variable using [System.IO.File]::AppendAllText.

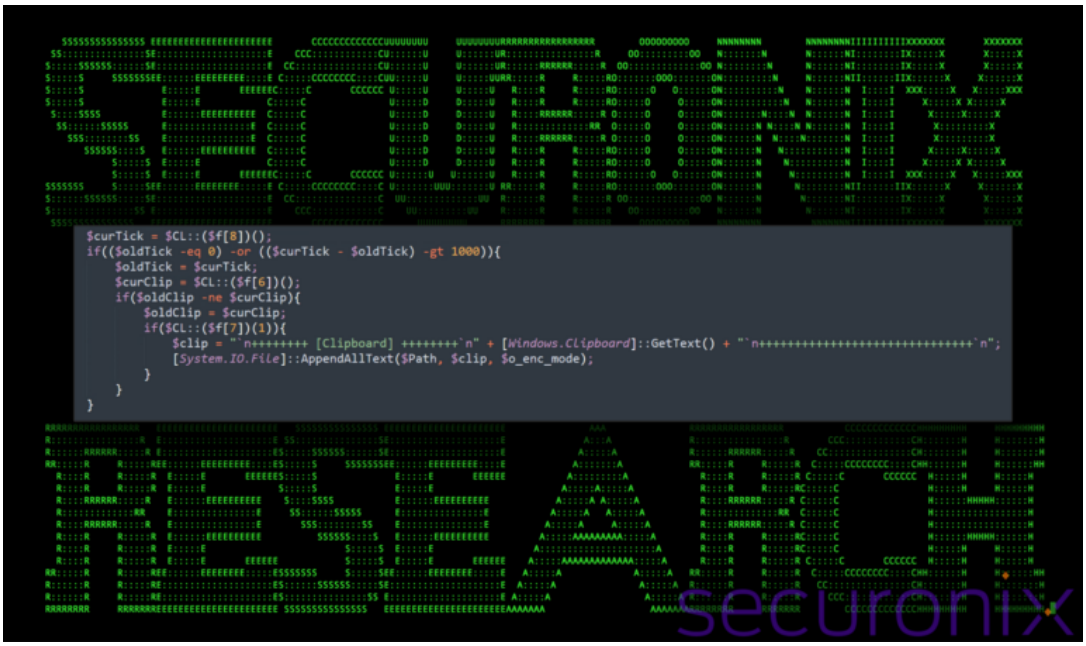


Figure 21: stage 8 PowerShell clipboard monitoring

Additional functionality:

- **Window monitoring:** It uses both GetForegroundWindow and GetWindowText to track the active window and its title, enabling the script to log which application the user is interacting with alongside the captured keystrokes or clipboard.
- **System tick count:** GetTickCount is also used to manage the timing of log entries (clipboard, keystrokes, etc), ensuring that entries are spaced out and potentially reducing the volume of logged data to focus on periods of activity.
- **Encoding and file writing:** All of the captured data is saved into the variable path $Path ("$env:appdata\Microsoft\Windows\Themes\version.xml"), using UTF-8 encoding (created and exfiltrated in stage 7.)

Wrapping up…

The malware payloads used in the DEEP#GOSU represent a sophisticated, multi-stage threat designed to operate stealthily on Windows systems especially from a network-monitoring standpoint. It relied on both PowerShell and VBScript for its execution which interestingly enough used very minimal obfuscation. Each stage was encrypted using AES and a common password and IV which should minimize network, or flat file scanning detections.

Its capabilities included keylogging, clipboard monitoring, dynamic payload execution, and data exfiltration, and persistence using both RAT software for full remote access, scheduled tasks as well as self-executing PowerShell scripts using jobs.

Securonix recommendations

Since many malware infections begin outside the organization, exercise caution especially around unsolicited emails, especially when the email is unexpected or employs a sense of urgency. When it comes to prevention and detection, the Securonix Threat Research Team recommends:

- Avoid downloading files or attachments from external sources, especially if the source was unsolicited.

- Monitor common malware staging directories, especially script-related activity in world-writable directories. In the case of this campaign the threat actors staged in subdirectories in %APPDATA%
- Since all of the network communication in the DEEP#GOSU campaign is encrypted and employs legitimate services such as Dropbox or Google Docs, we strongly recommend deploying robust endpoint logging capabilities. This includes leveraging additional process-level logging such as Sysmon and PowerShell logging for additional log detection coverage.
- Securonix customers can scan endpoints using the Securonix hunting queries below.

C2 and infrastructure

C2 Address
https://content.dropboxapi.com/2/files/download/step2/ps.bin
https://content.dropboxapi.com/2/files/download/step2/r_enc.bin
https://content.dropboxapi.com/2/files/download/step2/info_sc.txt
https://content.dropboxapi.com/2/files/download/step2/info_ps.bin
https://content.dropboxapi.com/2/files/download/step2/ad_ps.bin
https://content.dropboxapi.com/2/files/download/step2/info_sc.txt
gbionet.com

MITRE ATT&CK Matrix

Tactics	Techniques
Defense Evasion	T1027: Obfuscated Files or Information T1027.010: Obfuscated Files or Information: Command Obfuscation T1070.004: Indicator Removal: File Deletion T1140: Deobfuscate/Decode Files or Information
Discovery	T1057: Process Discovery T1082: System Information Discovery T1083: File and Directory Discovery
Execution	T1059: Command and Scripting Interpreter T1059.001: Command and Scripting Interpreter: PowerShell T1059.005: Command and Scripting Interpreter: Visual Basic T1204.001: User Execution: Malicious Link
Exfiltration	T1567.002 – Exfiltration Over Web Service: Exfiltration to Cloud Storage
Persistence	T1053 – Scheduled Task/Job
Command and Control	T1102: Web Service T1132.001: Data Encoding: Standard Encoding T1219 – Remote Access Software T1573: Encrypted Channel
Collection	T1115 – Clipboard Data T1056.001 – Input Capture: Keylogging

Relevant provisional Securonix detections

- EDR-ALL-623ER
- EDR-ALL-335-RU
- EDR-ALL-336-RU
- EDR-ALL-928-RU

Relevant hunting queries

(remove square brackets “[ ]” for IP addresses or URLs)

- index = activity AND rg_functionality=”Next Generation Firewall” AND requesturl CONTAINS “content.dropboxapi[.]com/2/files/download/step2/” AND (requesturl CONTAINS “ps.bin” OR requesturl

CONTAINS “r_enc.bin” OR requesturl CONTAINS “info_sc.txt” OR requesturl CONTAINS “info_ps.bin” OR requesturl CONTAINS “ad_ps.bin”)

- index = activity AND rg_functionality = “Endpoint Management Systems” AND (deviceaction = “File created” OR deviceaction = “File created (rule: FileCreate)”) AND customstring49 ENDS WITH “Appdata\Microsoft\Windows\Themes\version.xml”
- index = activity AND rg_functionality = “Microsoft Windows Powershell” AND (message CONTAINS “content.dropboxapi[.]com/2/files/download” OR message CONTAINS “content.dropboxapi[.]com/2/files/upload”)
- index = activity AND rg_functionality = “Endpoint Management Systems” AND (deviceaction = “File created” OR deviceaction = “File created (rule: FileCreate)”) AND customstring49 CONTAINS “\AppData\Local\Temp\” AND customstring49 CONTAINS “.zip” AND customstring49 ENDS WITH “.lnk”

Analyzed files/hashes

Name	FILE HASH
IMG_20240214_0001.pdf.lnk	F262588C48D2902992FFD275D2BE6362FE7F02E2F00A44AB8C75AC1A2827C6E 1617587CCDF5B0344089559ECF8FE7D39F6E07A6A64F74F2B44BFA2C8CB6798
트레이딩 스파르타코스 강의안-100불남(2차).zip	46A5D54C264152CE915792AF31C75824A558AF7D7340D78B34E146D8C6249E79
트레이딩_스파르타코스_강의안_100불남_2차.pdf.lnk	1B75F70C226C9ADA8E79C3FDD987277B0199928800C51E5A1E55FF01246701DE
IMG_20240214_0001.pdf	69C917EA96DB28DBD5B67073CA0AAC234D25651A849171B45F20979EAFA05A1
PowerShell file hashes	60666CACDD6806ED05771F32EAA719E3EFD2F4DB55F28A447D383C3EAC1DC7 B72CAAB78D164637FEA0937D7A94FC470579EC6BB4FA87DADB6F0FA7826E217 89CAD9A57985CC0AB3B7403A943AD0AA7B167DC7A3C38557417FEDEA67A77B

References:

1. North Korean Advanced Persistent Threat Focus: Kimsuky

https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-301a

1. Threat Intelligence Report: Kimsuky
   https://www.genians.co.kr/hubfs/blogfile/20231030_threat_inteligence_report_Kimsuky.pdf
2. Dark Pink – New APT hitting Asia-Pacific, Europe that goes deeper and darker
   https://www.group-ib.com/blog/dark-pink-apt/
3. February 2023 – Threat Trend Report on Kimsuky Group
   https://asec.ahnlab.com/wp-content/uploads/2023/04/ATIP_2023_Feb_Threat-Trend-Report-on-Kimsuky-Group.pdf
4. Malware Disguised as HWP Document File (Kimsuky)
   https://asec.ahnlab.com/en/54736/