...Operation（Giỗ Tổ Hùng Vương）hurricane：浅谈新海莲花组织在内存中的技战术

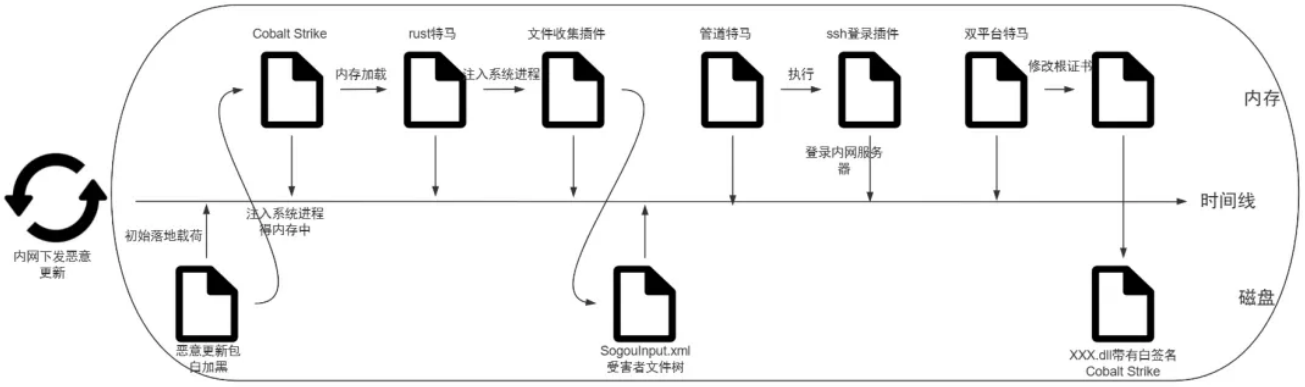
概述

新海莲花组织最早出现于2022年中，直到2023年底转入不活跃状态，2024年11月重新活跃并被我们快速制止并披露[1]，在2023年全年新海莲花组织展示出于以往完全不同的技战术，进攻水平也比之前提升很多，该组织使用多个 0day 漏洞针对我国军工、能源、航空等领域开展间谍活动，意图窃取我国能源和军工领域在中东、中亚、非洲、东亚的部署情况。

本文仅作为安全研究，我们不关注初始样本载荷，主要披露新海莲花组织内存插件和间谍目的，天擎EDR可以在内存中精准告警新海莲花组织所有内存插件，我们建议政企客户启用云查功能来发现未知威胁。

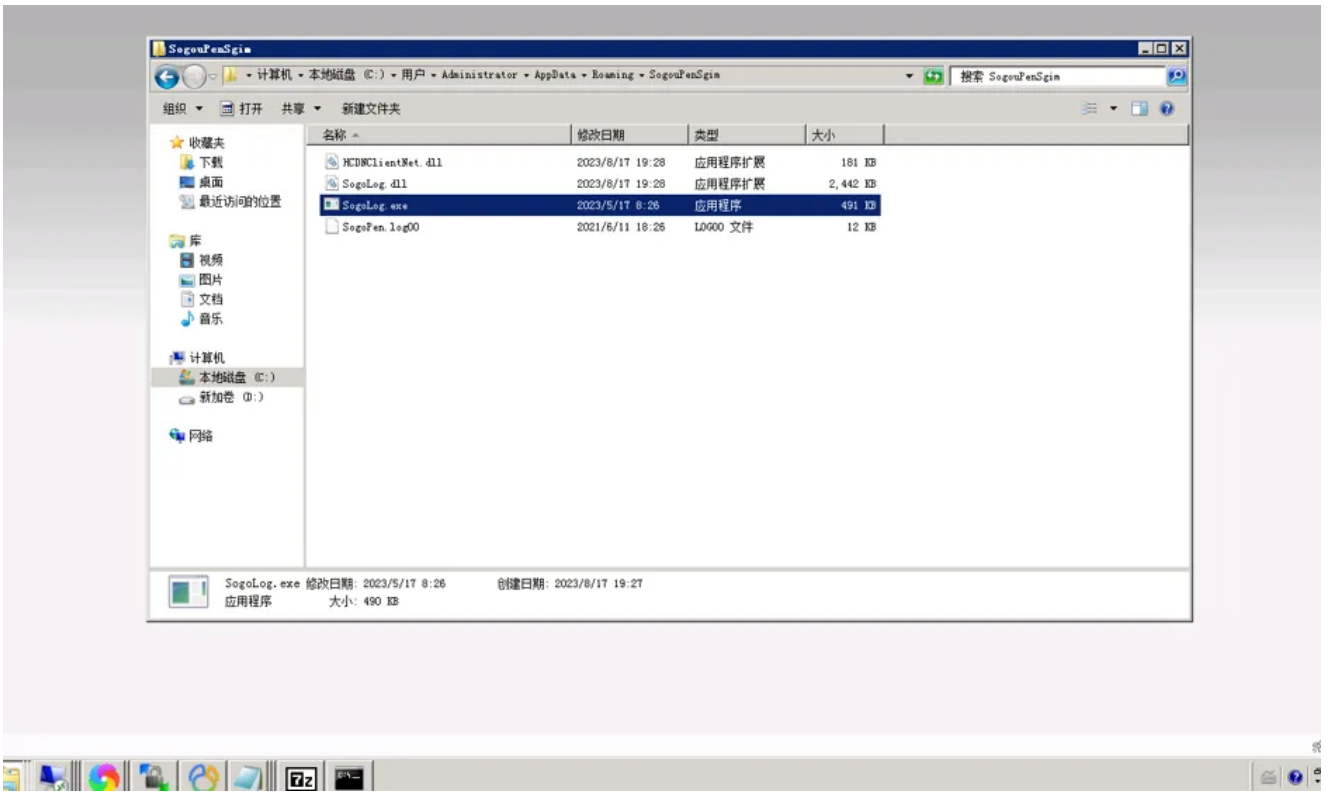
内存技战术

新海莲花组织通过终端软件 0day 漏洞向内网特定终端下发恶意更新，实现供应链攻击。在目前国内错综复杂的安全产品生态下，这种攻击模式是所有 APT 组织的最优解，并不是海莲花组织独有的手法[2]，我们甚至观察到针对国内的勒索软件运营商也有类似的操作，区别在于勒索运营商会向全内网终端下发勒索软件，而 APT 组织则是选择特定的目标终端后下发，新海莲花组织内存技战术　下：



新海莲花组织在 2022-2023 年所使用的 Cobalt Strike 有一个非常明显的特征：

木马运行后会自动将当前的屏幕截图保存为 PNG 格式并发送到 C2 服务器上， 果恰好此时攻击者正处于 RDP 的状态，那么就能在受害者机器上得到一张攻击者双击木马程序的照片：



Cobalt Strike 注入到系统进程后会在当前进程中内存加载 Rust 特马并回连新的 C2，Rust 特马的分析友商已经有过分析，故不再赘述。接着通过 Process Hollowing 的方式将文件目录收集插件注入到系统进程中。

文件名收集插件（内存态）

该插件为 20 KB大小的 Shellcode，首先会获取 Temp 路径，并生成一个 UUID 与路径拼接作为中间文件。

```
v28(260i64, v54);                       // GetTempPathW
sub_20F4DC5((__int64)v53, 0, 100i64);
sub_20F33FD(v18, v53);
v19(v54, v53);                          // PathAppendW
v7 = v24(v54, 0x40000000i64, 1i64, 0i64, 2, 128, 0i64);// CreateFileW
                                        // L"C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\5fff4bed-85ea-4ba1-8795-
v8 = v7;
```

接着会遍历文件，收集受害设备上的指定后缀文件，并将结果与 0xF1 异或后写入文件。

```
  do
  {
    *v4++ ^= 0xF1u;
    --v5;
  }
  while ( v5 );
}
v7 = 0;
return (*(__int64 (__fastcall **)(__int64, _BYTE *, __int64, int *, _QWORD))(a1 + 8))(a4, a2, a3, &v7, 0i64);// writefile
```

解密后的文件格式 下，会收集文件名和该文件的创建、修改、访问时间。

```
  Documents
desktop.ini*402*38*09/01/2022T20:04:54*01/13/2025T16:17:36*09/01/2022T22:10:37
  My Music

  My Pictures

  My Videos

  SweetScape

  Tencent Files
```

收集的指定后缀 下：pdf、png、jpg、ppt、pptx、one、ini、pfx、config、xmind、conf、ofd、7z、wpt。

写入完毕后会重新读取该文件到内存中，并删除该文件

```
v9 = v24(v54, 0x80000000i64, 1i64, 0i64, 3, 128, 0i64);// CreateFileW
                                                       // L"C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\5fff4bed-85ea-4ba1-8795-
v10 = v27(v9, 0i64);                                   // GetFileSize
v55 = v10;
if ( v10 )
{
  v11 = v10;
  v12 = v20(0i64, v10, 12288i64, 4i64);                // VirtualAlloc
  v13 = (_BYTE *)v12;
  if ( v12 )
  {
    if ( v26(v9, v12, v55, &v55, 0i64) )               // ReadFile
    {
      v25(v9);                                         // CloseHandle
      v30(v54);                                        // DeleteFileW
```

将读取到的内容重新与 0xF1 解密后，重新进行加密。

```
      do
      {
        *v15++ ^= 0xF1u;
        --v16;
      }
      while ( v16 );
      v14 = v55;
    }
    sub_20F1143(v4, (_DWORD)v5, (unsigned int)v34, (_DWORD)v13, v14, (__int64)v35);// decode
  }
  v21(v13, v11, 0x8000i64);
```

加密算法为 128 位 AES 算法。

```
{
  (*(void (__fastcall **)(char *, __int64, __int64))(a1 + 144))(v29, a2, 16i64);// memcpy
  if ( (*(unsigned int (__fastcall **)(__int64, int *, __int64, _QWORD, _DWORD, __int64 *))(a1 + 16))(// CryptImportKey
        v27,                                         // ALG_ID = 660E
        v28,
        28i64,
        0i64,
        0,
        &v26) )
```

最后将加密的内容再与 0xF2 异或后写入文件 C:\Programdata\SogouInput.xml中。

```
    do
    {
      *v13++ ^= 0xF2u;
      --v14;
    }
    while ( v14 );
  }
  if ( (unsigned int)sub_20F1246(v20, a6, v12) )// writefile
                                                // C:\\Programdata\\SogouInput.xml
```

攻击者会在后端对 xml 文件进行分析，最终挑选出窃取的目标文件，窃取完文档后攻击者　果选择进一步横向移动，一般会通过 Process Hollowing 的方式将管道特马注入到系统进程中。

管道特马（内存态）

该管道特马所在的内存块固定大小 0x35000，创建名为 \\.\pipe\InitStarts 的管道循环监听。

```
0014113C   56              push esi
0014113D   52              push edx
0014113E   6A 03           push 0x3
00141140   68 90471500     push 0x154790                UNICODE "\\.\pipe\InitStarts"
00141145   FF15 08E01400   call dword ptr ds:[0x14E008] kernel32.CreateNamedPipeW
0014114B   8BF0            mov esi,eax
0014114D   85F6            test esi,esi
0014114F   74 10           je short 00141161
00141151   6A 00           push 0x0
00141153   56              push esi
00141154   FF15 0CE01400   call dword ptr ds:[0x14E00C] kernel32.ConnectNamedPipe
0014115A   85C0            test eax,eax
0014115C   75 15           jnz short 00141173
0014115E   56              push esi
0014115F   FFD3            call ebx                     kernel32.CloseHandle
00141161   57              push edi
00141162   FF15 14E01400   call dword ptr ds:[0x14E014] kernel32.Sleep
```

读取管道中的数据：

```
23  v13 = 0x2000;
24  v14 = 0x2000;
25  v3 = 0;
26  for ( i = 0; ; v3 = i )
27  {
28    NamedPipeW = kernel32_CreateNamedPipeW(1394576, 3, v1, v2, v13, v14, v3, 0);
29    v5 = NamedPipeW;
30    if ( !NamedPipeW )
31      goto LABEL_5;
32    if ( kernel32_ConnectNamedPipe(NamedPipeW, 0) )
33      break;
34    kernel32_CloseHandle(v5);
35LABEL_5:
36    kernel32_Sleep(25);
37    v1 = v12;
38    v2 = v11;
```

通信过程中定义了一个结构体，创建线程并将结构体当作参数传入。

```
*(_DWORD *)(v6 + 308) = 0;
*(_DWORD *)(v6 + 312) = 0;
*(_DWORD *)(v6 + 316) = 0;
*(_DWORD *)(v6 + 320) = 0;
*(_DWORD *)(v6 + 324) = 0;
*(_DWORD *)(v6 + 328) = 0;
*(_DWORD *)(v6 + 8) = v5;                    // 管道句柄传递到了结构体
*(_DWORD *)v6 = 1;
kernel32_InterlockedIncrement(v9);
*(_DWORD *)(v6 + 4) = v15;
ntdll_RtlInitializeCriticalSection(v6 + 12);
*(_DWORD *)(v6 + 300) = 0x40000;
*(_DWORD *)(v6 + 304) = 11 * kernel32_GetCurrentProcessId();
*(_DWORD *)(v6 + 304) = 11 * kernel32_GetCurrentThreadId();
*(_DWORD *)(v6 + 304) += kernel32_GetTickCount();
*(_BYTE *)(v6 + 296) = 1;
kernel32_InterlockedIncrement(v6);
i = 0;
v7 = kernel32_CreateThread(0, 0x20000, 0x141860, v6, 0, &i);
if ( v7 )
{
  kernel32_CloseHandle(v7);
  return v6;
}
else
```

该线程会持续读取管道中的数据，并将数据解密后传递给工作线程，同时获取工作线程执行后的数据再解密传输到管道中。

```
2 {
3   while ( *(_BYTE *)(a1 + 296) )
4   {
5     if ( !fun_ReadPipew(a1) )
6       break;
7     if ( !fun_WritePipe(a1) )
8       break;
9     kernel32_Sleep(25);
10  }
11  sub_141730((_DWORD *)a1);
12  return 0;
```

加密算法　下：

```
38  v4 = *(_DWORD *)(this + 300);
39  v23 = 0;
40  v24 = 0;
41  v5 = (v3 * v3 * v3 + 18) % 0xB8A5u;
42  v6 = (unsigned __int8)v5;
43  *(_DWORD *)(this + 304) = v5;
44  v26 = 0;
45  v27 = 0;
46  v25 = (unsigned __int8)v5;
47  sub_141D40(&v22, 0, (int)&v25, (int)v28, (int)v21);
48  ntdll_RtlEnterCriticalSection(this + 12);
49  v7 = *(char **)(this + 320);
50  v8 = *(_DWORD *)(this + 324) - (_DWORD)v7;
51  v29 = v7;
52  if ( v4 <= v8 )
53    v8 = v4;
54  v25 = v6 ^ v8 ^ 0x2B48;
55  v26 = (v6 ^ v8 ^ 0x6C502B48) >> 16;
56  v27 = (v6 ^ v8 ^ 0x6C502B48) >> 24;
57  sub_141D40(&v22, v23, (int)&v25, (int)v28, (int)v7);
58  if ( v8 )
59  {
60    sub_141D40(&v22, v23, (int)v29, (int)&v29[v8], (int)v29);
61    v9 = v30;
62    v10 = *(_DWORD *)(v30 + 320);
63    v11 = v8 + v10;
64    if ( v10 != v8 + v10 )
65    {
66      v12 = *(_DWORD *)(v30 + 324) - v11;
67      sub_145290(v10, v11, v12);
68      *(_DWORD *)(v9 + 324) = v12 + v10;
69    }
```

工作线程中有大量的功能性函数，例　：文件管理、shellcode加载、命令执行等。

```
166  v5 = *((_DWORD *)a1 + 12);
167  switch ( v5 )
168  {
169    case 0:
170      return result;
171    case 1:
172      v159 = kernel32_InterlockedIncrement(a1, a2);
173      v6 = (_DWORD *)sub_166675(12);
174      *v6 = 0;
175      v6[1] = 0;
176      v6[2] = 0;
177      ntdll_RtlEnterCriticalSection(a1 + 4, a3);
178      v7 = (_DWORD *)*((_DWORD *)a1 + 7);
179      v8 = (int *)v7[1];
180      while ( !*((_BYTE *)v8 + 13) )
181      {
182        if ( v8[4] >= (int)v159 )
183        {
184          v7 = v8;
185          v8 = (int *)*v8;
186        }
187        else
188        {
189          v8 = (int *)v8[2];
190        }
191      }
192      if ( v7 == *((_DWORD **)a1 + 7) || (signed int)v159 < v7[4] )
```

我们观察到新海莲花组织通过管道内存加载了 ssh 登录插件。

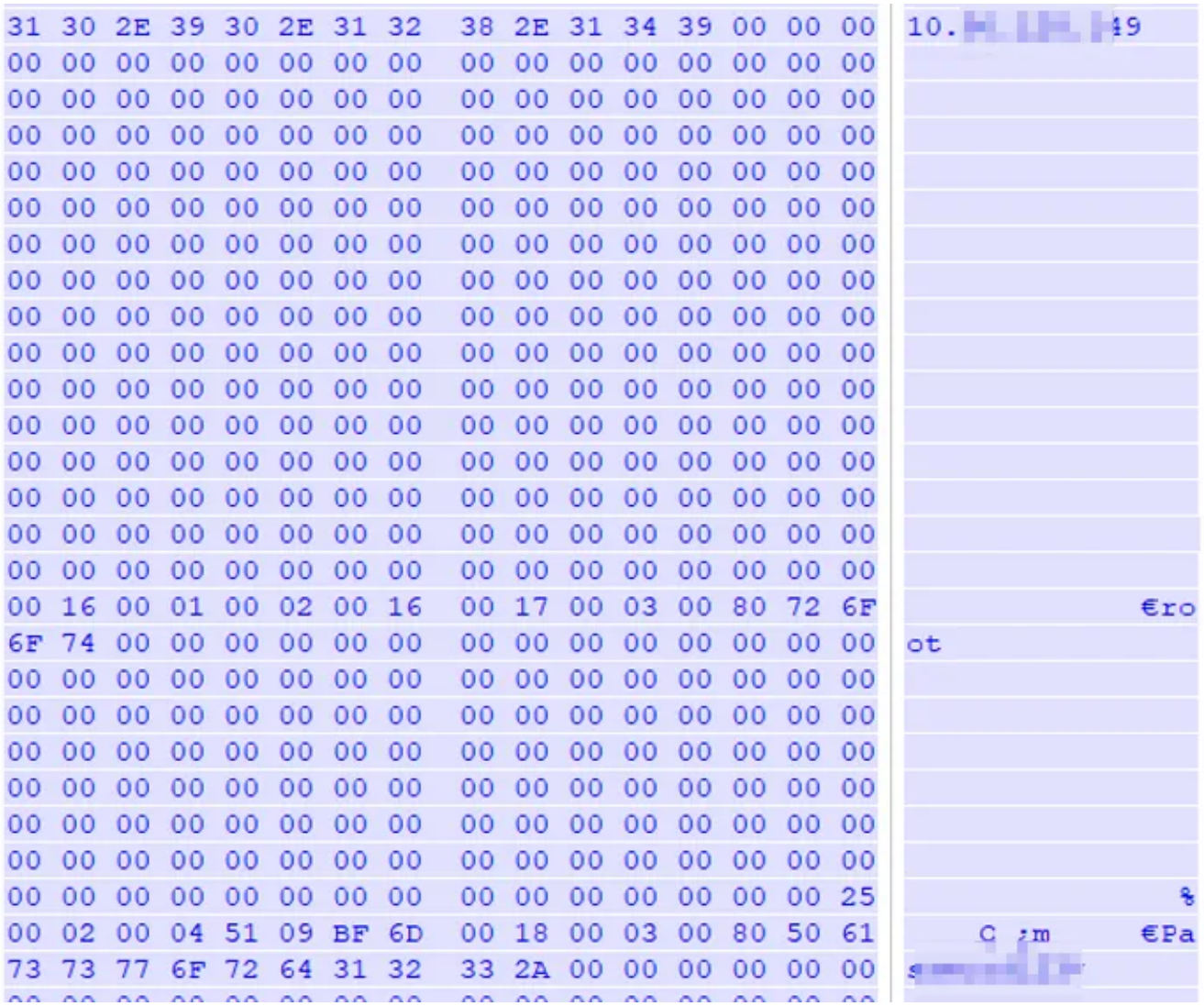
ssh登录插件（内存态）

该插件功能就是通过内置的账密登录内网 linux 服务器：

```
seg000:1000404E                 mov     ds:dword_100734D0, eax
seg000:10004053                 call    _memset
seg000:10004058                 add     esp, 10h
seg000:1000405B                 xor     eax, eax
seg000:1000405D                 mov     cl, 2Eh ; '.'
seg000:1000405F                 nop
seg000:10004060
seg000:10004060 loc_10004060:                           ; CODE XREF: sub_10004030+3C↓j
seg000:10004060                 xor     ds:byte_100673E0[eax], cl
seg000:10004066                 inc     eax
seg000:10004067                 cmp     eax, 2000h
seg000:1000406C                 jl      short loc_1000
seg000:1000406E                 push    2000h
seg000:10004073                 mov     ecx, offset by
seg000:10004078                 lea     eax, [esp+24h+
seg000:1000407C                 call    sub_10002B50
seg000:10004081                 add     esp, 4
seg000:10004084                 mov     esi, eax
seg000:10004086                 call    sub_10002B90

; char byte_100673E0[8192]
byte_100673E0   db 0E0h                 ; DATA XREF: sub_100
                                        ; sub_10004030+43↑o
                db 0E4h
                db 0E0h
                db 0E2h
                db 0E0h
                db 0E4h
                db 0E0h
```

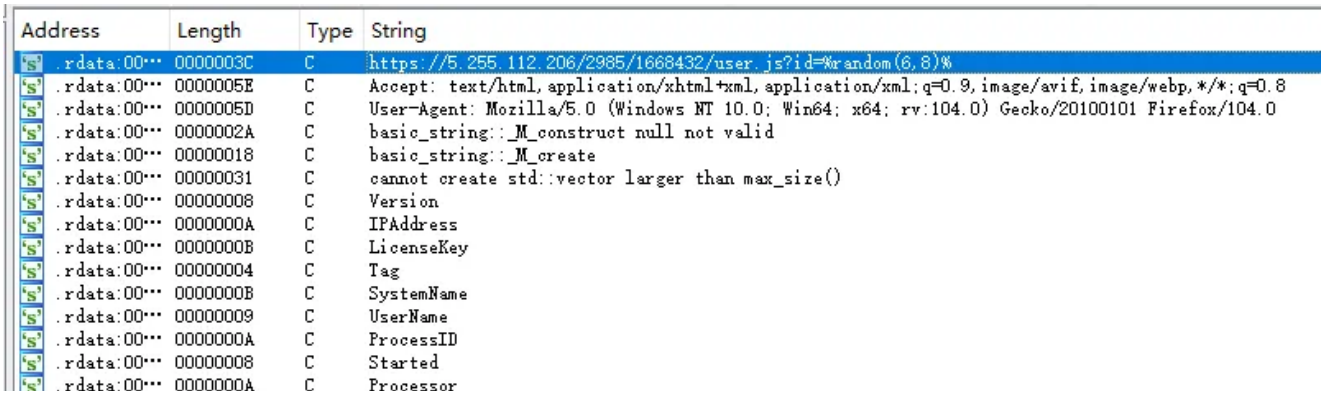
密码为弱口令，可以推测攻击者是通过爆破的方式获取到服务器密码



通过 history 日志可以确认攻击者在浏览服务器上的目录并打包数据。

双平台特马（内存态）

新海莲花组织在入侵边界服务器时 ：web服务器、防火墙等设备会使用一款 Win | linux 双平台的特马，该特马最早在防火墙上被发现，很长一段时间内我们认为该特马只在边界服务器上部署，但是在一次对抗的过程中发现该特马被注入到 Windows的系统进程中，并且注入时间离 Cobalt Strike 的植入时间晚一周。



Address	Length	Type	String
.rdata:00…	0000003C	C	https://5.255.112.206/2985/1668432/user.js?id=%random(6,8)%
.rdata:00…	0000005E	C	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
.rdata:00…	0000005D	C	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
.rdata:00…	0000002A	C	basic_string::_M_construct null not valid
.rdata:00…	00000018	C	basic_string::_M_create
.rdata:00…	00000031	C	cannot create std::vector larger than max_size()
.rdata:00…	00000008	C	Version
.rdata:00…	0000000A	C	IPAddress
.rdata:00…	0000000B	C	LicenseKey
.rdata:00…	00000004	C	Tag
.rdata:00…	0000000B	C	SystemName
.rdata:00…	00000009	C	UserName
.rdata:00…	0000000A	C	ProcessID
.rdata:00…	00000008	C	Started
.rdata:00…	0000000A	C	Processor

新海莲花组织通过该特马执行 CMD 命令，添加根证书 “certutil -addstore "ROOT" client.cer”，添加完成后选择在磁盘落地 DLL，此时的 DLL 带有数字签名，用于免杀 EDR。

```
[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\SystemCertificates\ROOT\Certificates\9E5613FF73B1F8CA0C6D1305F9CE141889A86819]
"Blob"=hex:19,00,00,00,01,00,00,00,10,00,00,00,84,3d,47,4e,24,95,eb,01,58,ff,\
  b0,f5,47,61,6b,ba,0f,00,00,00,01,00,00,00,14,00,00,00,ab,00,36,43,4d,c2,db,\
  15,cc,9a,92,9e,6e,b7,1f,33,98,96,88,75,03,00,00,00,01,00,00,00,14,00,00,00,\
  9e,56,13,ff,73,b1,f8,ca,0c,6d,13,05,f9,ce,14,18,89,a8,68,19,14,00,00,00,01,\
  00,00,00,14,00,00,00,dd,a5,0e,b5,16,88,e9,6a,11,e7,9c,56,1e,bb,16,cf,de,08,\
  fa,06,20,00,00,00,01,00,00,00,3e,03,00,00,30,82,03,3a,30,82,02,26,a0,03,02,\
  01,02,02,10,24,b4,31,9e,c7,c7,6b,ad,41,47,46,ac,cb,20,8d,40,30,09,06,05,2b,\
  0e,03,02,1d,05,00,30,27,31,25,30,23,06,03,55,04,03,13,1c,53,75,6d,61,74,72,\
  61,20,56,65,72,69,66,69,65,64,20,43,65,72,74,69,66,69,63,61,74,65,30,1e,17,\
  0d,32,33,30,34,31,39,30,30,30,30,30,30,5a,17,0d,32,34,30,34,32,30,30,30,30,\
  30,30,30,5a,30,27,31,25,30,23,06,03,55,04,03,13,1c,53,75,6d,61,74,72,61,20,\
  56,65,72,69,66,69,65,64,20,43,65,72,74,69,66,69,63,61,74,65,30,82,01,22,30,\
  0d,06,09,2a,86,48,86,f7,0d,01,01,01,05,00,03,82,01,0f,00,30,82,01,0a,02,82,\
  01,01,00,e4,6a,e6,4e,f0,60,64,bb,06,80,8e,35,39,79,19,d7,36,77,de,94,9f,41,\
  07,95,c6,5b,60,28,8c,ee,c7,d9,d9,2a,9e,47,ef,f2,c2,92,9b,52,b9,d7,d2,22,ae,\
  89,ff,11,f9,c8,dd,39,32,71,e9,bd,2e,ad,e1,c0,98,05,1a,fb,0a,99,6f,f0,26,bd,\
  7a,12,15,42,cf,a7,d7,3d,9a,39,37,6e,18,d8,7f,91,31,c7,a8,50,ec,12,7b,d0,19,\
  1d,b1,53,54,4c,9d,45,85,42,6c,a7,80,21,36,3b,1f,97,d3,c4,6a,16,a0,20,46,6f,\
  2f,1d,cd,6e,c3,7f,e5,f5,5b,10,6d,fd,60,66,0c,2c,1c,06,23,3a,b4,76,02,e6,c8,\
  82,8e,6d,c9,58,7f,0f,e3,26,f3,db,f6,e6,76,a5,7b,40,fb,f1,39,7b,51,f3,90,84,\
  1a,fb,56,b3,e9,4b,a0,c1,f0,f2,2b,8b,db,45,e0,24,69,46,c9,b9,f3,b2,a2,9b,f5,\
  d7,6a,d1,b9,94,3d,f1,be,58,d6,71,29,e0,5b,ee,5c,ac,6a,1c,fe,b1,1c,45,41,d6,\
  ed,4e,ed,8b,65,32,57,49,b5,a9,73,fe,4c,cd,d5,db,93,e6,bb,fa,86,b8,2e,40,e8,\
  97,c1,63,59,4c,48,a8,ed,c5,02,03,01,00,01,a3,6a,30,68,30,0c,06,03,55,1d,13,\
  01,01,ff,04,02,30,00,30,58,06,03,55,1d,01,04,51,30,4f,80,10,f1,72,05,96,e1,\
  70,25,2c,cc,7e,6a,8e,ac,0e,18,13,a1,29,30,27,31,25,30,23,06,03,55,04,03,13,\
  1c,53,75,6d,61,74,72,61,20,56,65,72,69,66,69,65,64,20,43,65,72,74,69,66,69,\
  63,61,74,65,82,10,24,b4,31,9e,c7,c7,6b,ad,41,47,46,ac,cb,20,8d,40,30,09,06,\
  05,2b,0e,03,02,1d,05,00,03,82,01,01,00,bd,79,3c,46,68,40,28,1c,ca,96,72,38,\
  3c,37,94,89,77,d1,0e,5a,0a,34,c4,c0,00,c2,2f,fb,38,3e,16,3a,04,80,a7,d6,d4,\
  4a,ec,04,49,51,42,20,e1,4d,0c,e2,10,a9,ca,dd,4c,e0,54,19,6a,fe,cc,fd,e7,17,\
  86,fe,12,86,b0,30,24,2b,b2,ca,9a,e3,c9,21,fe,e5,b3,bf,0f,ff,e9,29,47,c0,ef,\
  dd,bd,f6,15,80,bc,fa,38,ed,1c,95,43,1b,45,48,db,ed,73,a6,6a,38,f4,da,1b,ce,\
  6e,34,2f,cc,32,fd,22,e5,60,50,97,9b,29,8b,28,10,a7,d8,d2,f4,5d,a2,ab,66,20,\
  49,03,37,c6,5f,47,71,4a,cd,ba,47,4c,37,a2,fc,90,23,1d,87,15,a9,78,18,bd,5e,\
  e5,73,ee,64,82,80,24,b7,62,aa,85,07,ea,bd,26,71,db,ad,c0,c1,05,c7,c3,1d,3b,\
  47,ea,14,2d,9a,56,a8,4e,35,f4,4d,63,8c,17,f6,a5,a1,3b,6c,f9,65,18,28,63,a1,\
  37,d1,65,84,dd,56,34,cc,f3,1a,32,fc,b3,60,c6,6c,7d,29,3b,39,ea,cf,f1,be,a2,\
  34,d3,ce,53,ab,85,13,be,08,c5,2e,36,57,68,34,20,41,fb,1e
```

杀软对抗

新海莲花组织似乎是为数不多能够分清楚360安全卫士和天擎EDR两款杀软的APT组织，在此之前很多APT组织都认为只要能够免杀360安全卫士就能够绕过天擎EDR。新海莲花刚开始活动时使用了两种新方法分别针对360安全卫士和天擎EDR，但是很快被我们和友商发现并及时进行对抗。释放名为 propsys.dll，判断加载该 DLL 的进程是否为 360baobiao.exe

```
{
  HANDLE v0; // eax
  CHAR String1[272]; // [esp+1Ch] [ebp-110h] BYREF

  memset(String1, 0, 0x104u);
  v0 = GetCurrentProcess();
  GetModuleBaseNameA(v0, 0, String1, 260);
  return lstrcmpA(String1, "360Baobiao.exe") == 0;
}
```

DLL 的主要功能通过 DeviceIOCtontol 关闭自保。

```
hDevice = hObject;
if ( !hObject )
{
  v1 = sub_6BAC1530();
  if ( !v1 )
    return v1;
  hDevice = hObject;
}
BytesReturned = 0;
InBuffer[0] = 0;
v1 = DeviceIoControl(hDevi
if ( !v1 )
  CloseHandle(hObject);
hObject = 0;
```

之后休眠 zhudongfangyu.exe 和 360rps.exe 进程，实现致盲的效果。

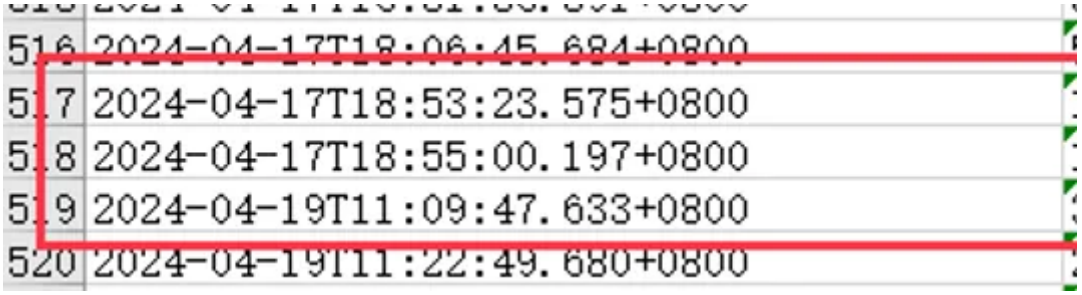
```
41        }
42        while ( v3 != v4 );
43      }
44      if ( lstrcmpW(L"ZHUDONGFANGYU.EXE", pe.szExeFile) && lstrcmpW(L"360RPS.EXE", pe.szExeFile) )
45      {
46        if ( !lstrcmpW("3", pe.szExeFile)
47          || !lstrcmpW(&word_6BAC4102, pe.szExeFile)
48          || !lstrcmpW(&word_6BAC4116, pe.szExeFile) )
49        {
50          v12[v10++] = pe.th32ProcessID;
51        }
52      }
53      else
54      {
55        v11[v1++] = pe.th32ProcessID;
56      }
57    }
58    while ( Process32NextW(v9, &pe) );
59    CloseHandle(v9);
60    if ( v1 )
61    {
62      v6 = 0;
63      while ( sub_6BAC16C0(v11[v6]) )
64      {
65        if ( ++v6 == v1 )
```

目前该手法已经无效。

UTC+7

上述复杂的内存 TTP 似乎只是“昙花一现”，从2023年12月份新海莲花转入不活跃状态之后，我们就再也没有观察到类似的技战术，在此之后2024年3月老海莲花继承了其攻击资源又发起了两波 0day 供应链事件，并让我们最终确认攻击者位于 UTC +7 时区，海莲花通过一些渠道购买国内 VPS 服务器将其当作代理一直在请求目标单位的终端管理服务器并挑选要入侵的目标人员。（不止海莲花，在2024年我们观察到几乎所有方向的APT组织都在通过代理公司或者黑产四件套来购买国内 VPS 厂商的资源，将其当作代理和C2，甚至还有找国内人员给木马后门代签数字签名的行为，APT已经深度融入国内黑灰产上下游中，我们建议对国内下游人员进行打击，将这些通道彻底遏制）。

在 UTC+8 的时区下，基于天眼设备可以观察到攻击者每天从早上10点一直工作到晚上19-20点，标准的八小时工作制，到点下班有双休，与国内红队的工作强度相比不太饱和，攻击者唯独在4月18号这一天缺勤，并且第二天“上班”时间较晚。



经过搜索发现4月18号为东南亚某国的法定节假日，称之为“雄王节”（Hung Kings/ Giỗ Tổ Hùng Vương）。

目的

目前已经确认攻击者位于东南亚国家，假设其为所在国提供情报服务。在研究其目的时，我们仅挑选终端服务器定向下发的案例来研究，因为攻击者在终端服务器上可以看到目标单位所有的组织架构和人员信息，其挑选的目标终端必然具有定向性，而其他的间谍活动比　批量入侵防火墙、web 服务等都是非定向性的，无法作为研究的数据源。

我们整理了2021-2024年间数起终端下发事件，大部分情况下其目标都聚焦在西南省份的环境和交通数据，以及我国在东亚的能源部署，这些都符合东南亚国家的利益，转折点在新海莲花出现之后，2023-2024年大规模刺探我国能源、军工领域在中亚、中东、北亚、非洲的项目和部署情况，受害终端上甚至包含向境外派遣的人员名单，这些数据并不是东南亚小国能够消化完的，更像是域外大国关注的领域，而新海莲花组织出现的时间又恰好和东南亚某国与域外大国达成网络安全合作的时间点相吻合。

以上只是我们作为网络安全厂商所观察到的事实的陈述，不针对任何国家和个人。

总结

目前，基于奇安信威胁情报中心的威胁情报数据的全线产品，包括奇安信威胁情报平台（TIP）、天擎、天眼高级威胁检测系统、奇安信NGSOC、奇安信态势感知等，都已经支持对此类攻击的精确检测。

参考链接

[1].https://ti.qianxin.com/blog/articles/new%20-trend-in-msi-file-abuse-new-ocean lotus-group-first-to-use-mst-files-to-deliver-special-trojan-cn/
[2].https://mp.weixin.qq.com/s/3bmehaRuvaL5TnvdZXwYWA