

个人信息

姓名：付文轩

学号：1911410

专业：信息安全

Lab 10-1

部分实验过程

首先使用Strings工具进行一个简单的静态分析，查看一下有没有哪些比较值得注意的字符串

```
ControlService
StartServiceA
OpenServiceA
CreateServiceA
OpenSCManagerA
ADVAPI32.dll
GetModuleHandleA
GetStartupInfoA
GetCommandLineA
GetVersion
ExitProcess
TerminateProcess
GetCurrentProcess
UnhandledExceptionFilter
GetModuleFileNameA
FreeEnvironmentStringsA
FreeEnvironmentStringsW
WideCharToMultiByte
GetEnvironmentStrings
GetEnvironmentStringsW
SetHandleCount
GetStdHandle
GetFileType
HeapDestroy
HeapCreate
VirtualFree
HeapFree
RtlUnwind
WriteFile
GetCPIInfo
GetACP
GetOEMCP
HeapAlloc
VirtualAlloc
HeapReAlloc
GetProcAddress
LoadLibraryA
MultiByteToWideChar
LCMapStringA
LCMapStringW
GetStringTypeA
GetStringTypeW
KERNEL32.dll
."@
Lab10-01
```

```
C:\Windows\System32\Lab10-01.sys
```

```
CE
```

```
xBE
```

```
HB
```

```
$B
```

```
`A
```

```
<A
```

```
`y!
```

```
e~C
```

```
[
```

```
Q^ _j2
```

```
<<<<<
```

```
H
```

```
&File
```

```
iE&xit
```

```
&Help
```

```
h&About ...
```

```
About
```

```
System
```

```
RegWriterApp Version 1.0
```

```
Copyright (C) 2011
```

```
RegWriterApp
```

```
Hello World!
```

```
REGWRITERAPP
```

从字符串里可以看到有一个指定文件 `C:\Windows\System32\Lab10-01.sys`，从上面的函数名称猜测会有服务相关的行为，结合之前的文件，猜测可能服务会执行这个文件中的内容。

对这个sys文件进行简单的静态分析

```

!This program cannot be run in DOS mode.
t<
Rich
.text
h.rdata
H.data
INIT
.rsrc
B.reloc
QSU
WSJ
_^I
EnableFirewall
\Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall\StandardProfile
\Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall\DomainProfile
\Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall
\Registry\Machine\SOFTWARE\Policies\Microsoft
RSDS
p-
\><"
c:\winddk\7600.16385.1\src\general\regwriter\wdm\sys\objfre_wxp_x86\i386\sioctl.
pdb

NE

RtlWriteRegistryValue
RtlCreateRegistryKey
KeTickCount
ntoskrnl.exe
US_VERSION_INFO
StringFileInfo
040904B0
CompanyName
Windows (R) Win 7 DDK provider
FileDescription
Important System Driver
FileVersion
6.1.7600.16385 built by: WinDDK
InternalName
Lab10-01.sys
LegalCopyright
ABC Corp.
OriginalFilename
Lab10-01.sys
ProductName
Windows (R) Win 7 DDK driver
ProductVersion
6.1.7600.16385
VarFileInfo
Translation
9!939:9?9H909

```

可以看到有类似注册表的操作，以及ntoskrnl.exe这个可执行文件。

接下来使用IDA简单查看一下exe程序的所有功能

```

push    0F003Fh          ; dwDesiredAccess
push    0                ; lpDatabaseName
push    0                ; lpMachineName
call    ds:OpenSCManagerA
mov     edi, eax
test    edi, edi
jnz     short loc_401020

loc_401020:
push    esi
push    0                ; lpPassword
push    0                ; lpServiceStartName
push    0                ; lpDependencies
push    0                ; lpdwTagId
push    0                ; lpLoadOrderGroup
push    offset BinaryPathName ; "C:\\Windows\\System32\\Lab10-01.sys"
push    1                ; dwErrorControl
push    3                ; dwStartType
push    1                ; dwServiceType
push    0F01FFh          ; dwDesiredAccess
push    offset ServiceName ; "Lab10-01"
push    offset ServiceName ; "Lab10-01"
push    edi              ; hSCManager
call    ds:CreateServiceA
mov     esi, eax
test    esi, esi

```

```

push    0F01FFh          ; dwDesiredAccess
push    offset ServiceName ; "Lab10-01"
push    edi              ; hSCManager
call    ds:OpenServiceA
mov     esi, eax
test    esi, esi
jz      short loc_401086

```

```

loc_401069:                ; lpServiceArgVectors
push    0
push    0                ; dwNumServiceArgs
push    esi              ; hService
call    ds:StartServiceA
test    esi, esi
jz      short loc_401086

```

```

lea     eax, [esp+24h+ServiceStatus]
push    eax              ; lpServiceStatus
push    1                ; dwControl
push    esi              ; hService
call    ds:ControlService

```

我们发现这个程序的功能比较简单，就是使用了OpenSCManagerA获取当前服务管理器的句柄，然后CreateServiceA创建一个名为Lab10-01的服务，之后会调用StartServiceA启动服务，最后调用ControlService，在经过查阅资料以后可以发现ControlService传入的第二个参数是1，这个参数的意思就是会卸载驱动。

同时我们可以注意到，在创建服务的时候使用了之前分析的驱动程序的代码，并且dwStartType设置为3，也就是说这个服务会以内核级运行。

接下来使用IDA查看一下这个驱动文件

```

INIT:00010959      public DriverEntry
INIT:00010959      DriverEntry      proc near
INIT:00010959
INIT:00010959      DriverObject      = dword ptr 8
INIT:00010959      RegistryPath      = dword ptr 0Ch
INIT:00010959
INIT:00010959      mov     edi, edi
INIT:0001095B      push    ebp
INIT:0001095C      mov     ebp, esp
INIT:0001095E      call   sub_10920
INIT:00010963      pop     ebp
INIT:00010964      jmp     sub_10906
INIT:00010964      DriverEntry      endp
INIT:00010964
INIT:00010964      ;
INIT:00010969      align 4
INIT:0001096C      __IMPORT_DESCRIPTOR_ntoskrnl_exe dd rva off_10994
INIT:0001096C      ; DATA XREF: HEADER:000102D8 to
INIT:0001096C      ; Import Name Table
INIT:00010970      dd 0
INIT:00010974      dd 0
INIT:00010978      dd rva aNtoskrnl_exe
INIT:0001097C      dd rva RtlCreateRegistryKey ; Import Address Table
INIT:00010980      dd 5 dup(0)
INIT:00010984      ;
INIT:00010994      ; Import names for ntoskrnl.exe
INIT:00010994      ;
INIT:00010994      off_10994      dd rva word_109BC
INIT:00010998      dd rva word_109D4
INIT:0001099C      dd rva word_109A4

```

在00010964位置可以看到一个无条件跳转指令，跳转到了sub_10906，也就是说这个驱动程序真正的入口点应该是在这个函数的位置。跟踪过去查看一下内容

```

; Attributes: bp-based frame

sub_10906 proc near

arg_0= dword ptr 8

mov     edi, edi
push    ebp
mov     ebp, esp
mov     eax, [ebp+arg_0]
mov     dword ptr [eax+34h], offset sub_10486
xor     eax, eax
pop     ebp
retn    8
sub_10906 endp

```

但是仅从这里并不能看出来有什么，只看见把一个偏移地址放到了内存中去。

问题1

Does this program make any direct changes to the registry? (Use procmon to check.)

使用Procmon工具，经过过滤之后得到的内容为

Ti...	Process Name	PID	Operation	Path	Result	Detail
18:2...	Lab10-01.exe	1412	Process Start		SUCCESS	Parent PID: 1...
18:2...	Lab10-01.exe	1412	Thread Create		SUCCESS	Thread ID: 1720...
18:2...	Lab10-01.exe	1412	QueryNameIn...	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	Name: \Docume...
18:2...	Lab10-01.exe	1412	Load Image	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	Image Base: 0...
18:2...	Lab10-01.exe	1412	Load Image	C:\WINDOWS\system32\ntdll.dll	SUCCESS	Image Base: 0...
18:2...	Lab10-01.exe	1412	QueryNameIn...	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	Name: \Docume...
18:2...	Lab10-01.exe	1412	CreateFile	C:\WINDOWS\system32\kernel32.dll	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Lab10-01.exe	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	CreateFile	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	Desired Acces...
18:2...	Lab10-01.exe	1412	FileSystemC...	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	Control: FSCT...
18:2...	Lab10-01.exe	1412	QueryOpen	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe Local	NAME NOT FOUND	Image Base: 0...
18:2...	Lab10-01.exe	1412	ReadFile	C:\WINDOWS\system32\kernel32.dll	SUCCESS	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Acces...
18:2...	Lab10-01.exe	1412	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DW...
18:2...	Lab10-01.exe	1412	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Image Base: 0...
18:2...	Lab10-01.exe	1412	Load Image	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	Offset: 16, 38...
18:2...	Lab10-01.exe	1412	Load Image	C:\WINDOWS\system32\advapi32.dll	SUCCESS	Image Base: 0...
18:2...	Lab10-01.exe	1412	Load Image	C:\WINDOWS\system32\secur32.dll	SUCCESS	Image Base: 0...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Acces...
18:2...	Lab10-01.exe	1412	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DW...
18:2...	Lab10-01.exe	1412	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Image Base: 0...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Secur32.dll	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\SECRT4.dll	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ADVAPI32.dll	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Desired Acces...
18:2...	Lab10-01.exe	1412	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat	SUCCESS	Type: REG_DW...
18:2...	Lab10-01.exe	1412	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server	SUCCESS	Type: REG_DW...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Acces...
18:2...	Lab10-01.exe	1412	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\LeakTrack	NAME NOT FOUND	Length: 144
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon	SUCCESS	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Diagnostics	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ntdll.dll	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\kernel32.dll	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	ReadFile	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	Offset: 4, 096...
18:2...	Lab10-01.exe	1412	ReadFile	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	Offset: 20, 48...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Rpc\RpcSs\Buffers	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Rpc	SUCCESS	Desired Acces...
18:2...	Lab10-01.exe	1412	RegQueryValue	HKLM\SOFTWARE\Microsoft\Rpc\MaxRpcSize	NAME NOT FOUND	Length: 144
18:2...	Lab10-01.exe	1412	RegCloseKey	HKLM\SOFTWARE\Microsoft\Rpc	SUCCESS	
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Lab10-01.exe\RpcThreadPoolThrott...	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\Software\Policies\Microsoft\Windows NT\Rpc	NAME NOT FOUND	Desired Acces...
18:2...	Lab10-01.exe	1412	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Acces...
18:2...	Lab10-01.exe	1412	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\SafeDllSearchMode	NAME NOT FOUND	Length: 16
18:2...	Lab10-01.exe	1412	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	
18:2...	Lab10-01.exe	1412	QueryNameIn...	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	BUFFER OVERFLOW	Name: \D
18:2...	Lab10-01.exe	1412	QueryNameIn...	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	Name: \Docume...
18:2...	Lab10-01.exe	1412	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BIN...
18:2...	Lab10-01.exe	1412	SetEndOfFile...	C:\WINDOWS\system32\config\software.LOG	SUCCESS	EndOfFile: 8,192
18:2...	Lab10-01.exe	1412	SetEndOfFile...	C:\WINDOWS\system32\config\software.LOG	SUCCESS	EndOfFile: 8,192
18:2...	Lab10-01.exe	1412	SetEndOfFile...	C:\WINDOWS\system32\config\software.LOG	SUCCESS	EndOfFile: 16
18:2...	Lab10-01.exe	1412	Thread Exit		SUCCESS	Thread ID: 17...
18:2...	Lab10-01.exe	1412	Process Exit		SUCCESS	Exit Status: ...
18:2...	Lab10-01.exe	1412	CloseFile	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-01.exe	SUCCESS	

可以看到这个程序的Operation中有很多关于注册表的操作，再次过滤后可以看见

Ti...	Process Name	PID	Operation	Path	Result	Detail
18:2...	Lab10-01.exe	1412	RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BIN...

对注册表的写操作，其实只有这一条，并且修改的是

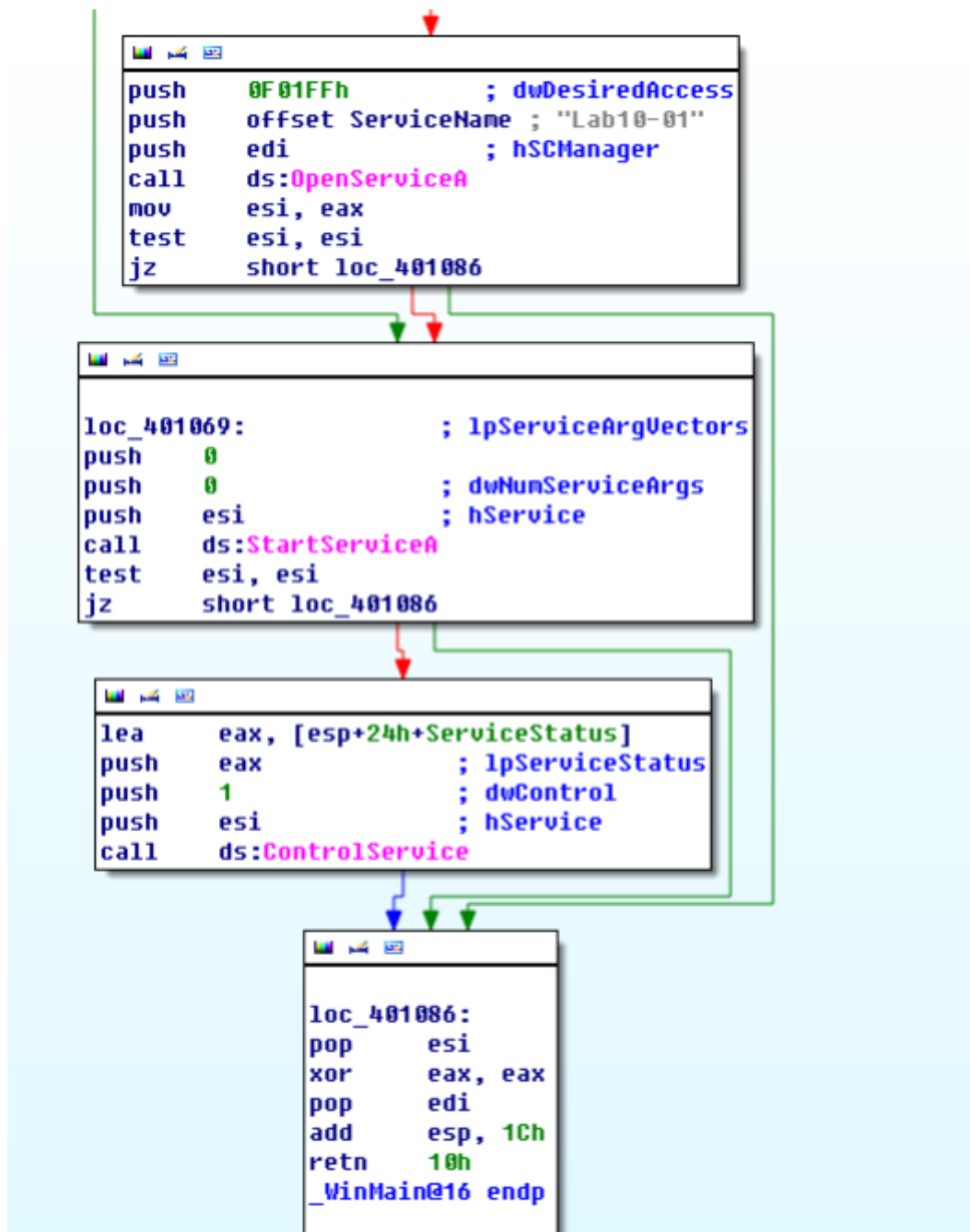
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed。

也就是说，能够使用Procmon监视到的对注册表的操作仅有这一条。

问题2

The user-space program calls the ControlService function. Can you set a breakpoint with WinDbg to see what is executed in the kernel as a result of the call to ControlService?

通过前述分析，我们知道了**恶意代码打开 Lab10-01 服务，启动服务并最终通过 ControlService 关闭它**，如下图所示：



中断内核调试器并使用 `!object \Driver` 命令来显示已加载的驱动程序

```

-----
nt!RtlpBreakWithStatusInstruction:
80527bdc cc          int      3
kd> !object \Driver
Object: e101d910  Type: (8a360418) Directory
      ObjectHeader: e101d8f8 (old version)
      HandleCount: 0  PointerCount: 87
      Directory Object: e1001150  Name: Driver

```

Hash	Address	Type	Name
00	8a0fd3a8	Driver	Beep
	8a20c3b0	Driver	NDIS
	8a053438	Driver	KSecDD
01	8a0ad7d0	Driver	Mouclass
	8a04ae38	Driver	Raspti
	89e28de8	Driver	es1371
02	89e29bf0	Driver	vmx_svga
03	89e57b90	Driver	Fips
	8a1f87b0	Driver	Kbdclass
04	89fe6f38	Driver	VgaSave
	89ee6030	Driver	NDProxy
	8a2a60b8	Driver	Compbatt
05	8a292ec8	Driver	Ptilink
	8a31e850	Driver	MountMgr
	8a2717e0	Driver	wdmaud
07	89e0d7a0	Driver	dmload
	8a2b0218	Driver	isapnp
	89e3c2c0	Driver	swmidi
08	8a28b948	Driver	redbook
	8a1f75f8	Driver	vmmouse
	8a0ed510	Driver	atapi
09	89e0ea08	Driver	vm SCSI
10	89fe4da0	Driver	IpNat
	8a0e3728	Driver	RasAcid
	8a072d68	Driver	PSched

通过观察我们发现SERVICE_CONTROL_STOP 将调用 DriverUnload 函数，要弄清楚 DriverUnload 函数的地址是什么，首先要在 Lab10-01 驱动程序条目上设置一个断点，使用的命令为 `bu`

`Lab10_01!DriverEntry`

接下来我们需要单步执行，直到 Lab10_01.sys 被加载。使用 `step out` 直至看到 `nt!`

`IopLoadUnloadDriver + 0x45`。使用 `!object \Driver` 来列出已经加载的驱动程序，然后我们使用

`dt` 命令 (display out) 来查看lab10-01 驱动程序：

```

kd> !object 89d3ada0
Object: 89d3ada0  Type: (8a3275b8) Driver
      ObjectHeader: 89d3ad88 (old version)
      HandleCount: 0  PointerCount: 2
      Directory Object: e101d910  Name: Lab10-01
kd> dt _DRIVER_OBJECT 89d3ada0
nt!_DRIVER_OBJECT
+0x000 Type           : 0n4
+0x002 Size           : 0n168
+0x004 DeviceObject   : (null)
+0x008 Flags          : 0x12
+0x00c DriverStart    : 0xbaf7e000 Void
+0x010 DriverSize     : 0xe80
+0x014 DriverSection  : 0x89e3b630 Void
+0x018 DriverExtension : 0x89d3ae48 _DRIVER_EXTENSION
+0x01c DriverName     : _UNICODE_STRING "\Driver\Lab10-01"
+0x024 HardwareDatabase : 0x80670ae0 _UNICODE_STRING "\REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM"
+0x028 FastIoDispatch : (null)
+0x02c DriverInit     : 0xbaf7e959      long  +0
+0x030 DriverStartIo  : (null)
+0x034 DriverUnload   : 0xbaf7e486      void  +0
+0x038 MajorFunction  : [28] 0x804f354a      long  nt!IopInvalidDeviceRequest+0

```

在上图中我们可以看见，DriverUnload 函数的地址是0xbaf7e486（图片中的倒数第二行），使用指令 `bp 0xbaf7e486` 在这个位置上打个断点。重新启动lab10-01.exe，在这一处的断点被触发，之后已使用 `g` 进行单步执行


```

kd> p
Lab10_01+0x48d:
baf7e48d 56          push     esi
kd> p
Lab10_01+0x48e:
baf7e48e 8b3580e7f7ba  mov     esi,dword ptr [Lab10_01+0x780 (baf7e780)]
kd> p
Lab10_01+0x494:
baf7e494 57          push     edi
kd> p
Lab10_01+0x495:
baf7e495 33ff        xor      edi,edi
kd> p
Lab10_01+0x497:
baf7e497 68bce6f7ba  push    offset Lab10_01+0x6bc (baf7e6bc)
kd> p
Lab10_01+0x49c:
baf7e49c 57          push     edi
kd> du baf7e6bc
baf7e6bc "\Registry\Machine\SOFTWARE\Polic"
baf7e6fc "ies\Microsoft"
kd> t
Lab10_01+0x49d:
baf7e49d 897dfc      mov     dword ptr [ebp-4],edi
kd> t
Lab10_01+0x4a0:
baf7e4a0 ffd6        call    esi
kd> t
nt!RtlCreateRegistryKey:
805ddafe 8bff        mov     edi,edi

```

在红框标记的位置可以看见有对注册表的操作，这里是调用了RtlCreateRegistryKey函数。

RtlCreateRegistryKey function (wdm.h)

10/22/2021 • 2 minutes to read

[Is this page helpful?](#)

The **RtlCreateRegistryKey** routine adds a key object in the registry along a given relative path.

根据微软官方文档可以知道这个函数的作用就是在注册表中添加一个key。

同时在后面可以看见还有对RtlWriteRegistryValue 的调用，这样一个完整的键值对就被创建成功了。

通过观察调试时的操作，我们可以得到如下的参数列表内容：

```

1  NTSTATUS RtlWriteRegistryValue(
2  _In_ ULONG RelativeTo = 0,
3  _In_ PCWSTR Path = offset Lab10_01+0x5a8(f7a545a8),
4  _In_ PCWSTR ValueName = offset Lab10_01+0x4ee(f7a544ee),
5  _In_ ULONG ValueType = 4,
6  _In_opt_ PVOID ValueData = eax,
7  _In_ ULONG ValueLength = 4
8  );

```

在这个参数列表中比较有价值、需要关注的就是Path参数，经过分析以后可以知道这个参数的值应该是\Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall\DomainProfile，然后ValueName的值是 EnableFirewall

我们注意到这个ValueData设置的是eax中的值，根据上面汇编代码可以知道eax中存放的是[ebp-4]。很幸运的是，windbg是有能够直接查看寄存器中值的指令的，通过 `r ebp` 可以得到当前ebp中的值是 f78ded58，那么ebp-4也就是f78ded54。使用指令 `dc f78ded54` 可以看到当前这个位置上的值是0，也就是说这里参数ValueData的值是被设置为0的。这里的意思就是将 EnableFirewall 这个的值设置为了0，意义就是从内核禁止了Windows的防火墙功能

继续向下分析，可以发现下面调用的函数是RtlWriteRegistryValue，这个和上一个是一套使用的，同样是观察一下他的参数结构，经过分析以后可以得到

```
1 NTSTATUS RtlWriteRegistryValue(  
2     _In_ ULONG RelativeTo = 0,  
3     _In_ PCWSTR Path = ebx,  
4     _In_ PCWSTR ValueName = edi,  
5     _In_ ULONG ValueType = 4,  
6     _In_opt_ PVOID ValueData = eax,  
7     _In_ ULONG ValueLength = 4  
8 );
```

这里需要关注的几个参数的具体值为：

Path 的值应该是

\Registry\Machine\SOFTWARE\Policies\Microsoft\WindowsFirewall\StandardProfile

ValueName 的值是 EnableFirewall

ValueData 的值是 0

从效果上来说和之前的函数是一样的，也是从内核关闭防火墙

之后剩下的内容就是对栈的处理，程序执行结束。

问题3

What does this program do?

这个程序创建了一个名为Lab10-01的服务，然后驱动代码会创建两个注册表键，这两个注册表键的效果是禁用防火墙。

Lab 10-2

部分实验过程

首先使用strings工具简单查看一下

```
WriteFile
SizeofResource
CreateFileA
LoadResource
FindResourceA
KERNEL32.dll
CloseServiceHandle
StartServiceA
CreateServiceA
OpenSCManagerA
ADVAPI32.dll
GetCommandLineA
GetVersion
ExitProcess
TerminateProcess
GetCurrentProcess
UnhandledExceptionFilter
GetModuleFileNameA
FreeEnvironmentStringsA
FreeEnvironmentStringsW
WideCharToMultiByte
GetEnvironmentStrings
GetEnvironmentStringsW
SetHandleCount
GetStdHandle
GetFileType
GetStartupInfoA
HeapDestroy
HeapCreate
VirtualFree
HeapFree
RtlUnwind
HeapAlloc
GetCPIInfo
GetACP
GetOEMCP
VirtualAlloc
HeapReAlloc
GetProcAddress
LoadLibraryA
GetLastError
FlushFileBuffers
SetFilePointer
MultiByteToWideChar
LCMapStringA
LCMapStringW
GetStringTypeA
GetStringTypeW
SetStdHandle
D2D
Failed to start service.
Failed to create service.
486 WS Driver
Failed to open service manager.
```

```

c:\winddk\7600.16385.1\src\general\rootkit\wdm\sys\objfre_wxp_x86\siocctl.pd
b
E
98t

NE

KeServiceDescriptorTable
NtQueryDirectoryFile
RtlCompareMemory
NtQueryDirectoryFile
MmGetSystemRoutineAddress
RtlInitUnicodeString
KeTickCount
ntoskrnl.exe
US_VERSION_INFO
StringFileInfo
040904B0
CompanyName
Windows (R) Win 7 DDK provider
FileDescription
Sample IOCTL Driver
FileVersion
6.1.7600.16385 built by: WinDDK
InternalName
SIOCTL.sys
LegalCopyright
Microsoft Corporation. All rights reserved.
OriginalFilename
SIOCTL.sys
ProductName
Windows (R) Win 7 DDK driver
ProductVersion
6.1.7600.16385
VarFileInfo
Translation
7"7.707[7b7s7

```

可以看见这个程序里有对服务的创建、打开等一套操作，还有对文件的写操作等。

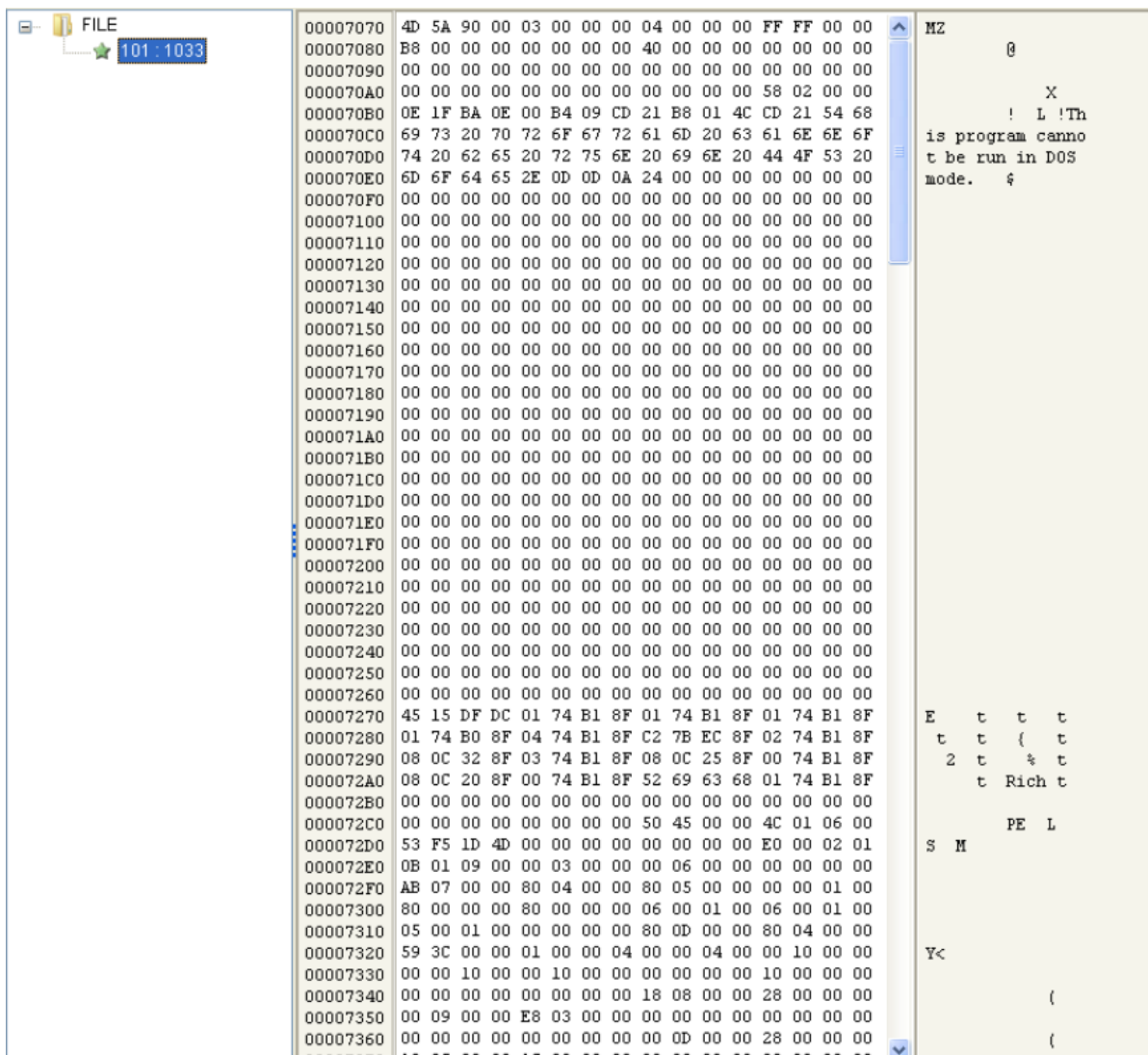
使用IDA进行简答的静态分析

```

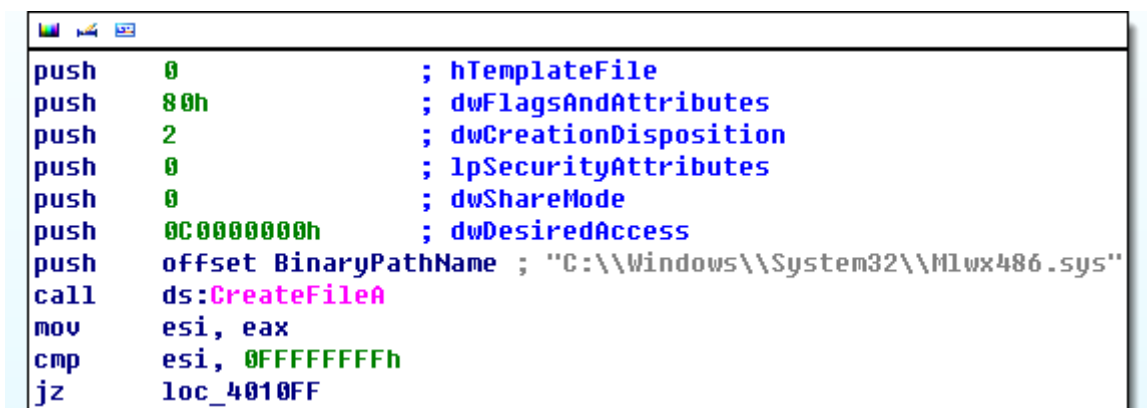
push    ebx                ; lpname
push    0                  ; hModule
call     ds:FindResourceA
mov     edi, eax
push    edi                ; hResInfo
push    0                  ; hModule
call     ds:LoadResource
test    edi, edi
mov     ebx, eax
jz      loc_4010FF

```

首先可以看见这个程序调用了两个对Resource的操作，也就是说这个程序会对他的src节进行操作，使用resource hacker查看一下他的资源节都有什么内容



发现一个比较神奇的事，这个资源节的开头两个字节是 4D 5A，也就是说这个资源节里是包含一个PE文件的。



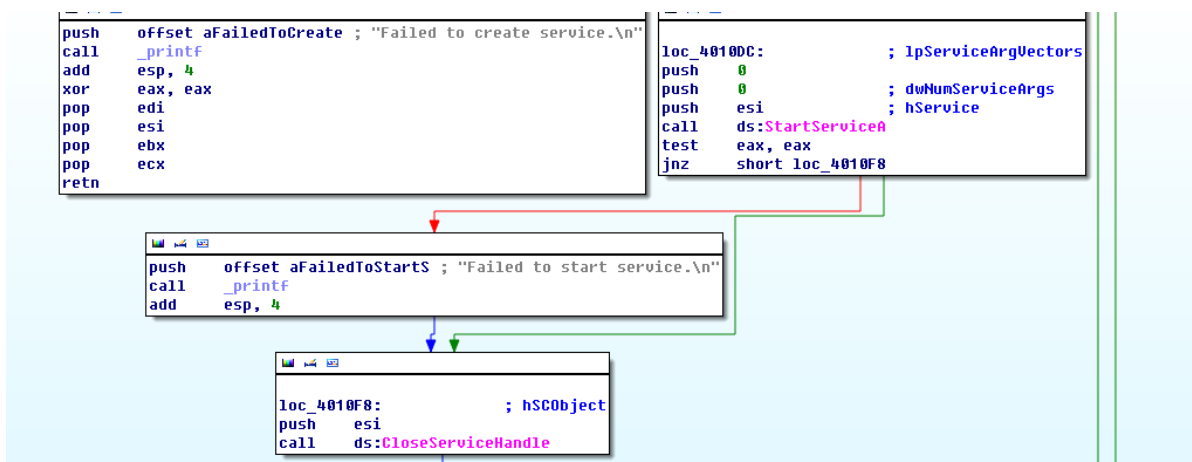
继续向下分析可以看见这里创建了一个名为Mlwx486.sys的驱动文件

```

loc_401097:                ; lpPassword
push    0
push    0                  ; lpServiceStartName
push    0                  ; lpDependencies
push    0                  ; lpdwTagId
push    0                  ; lpLoadOrderGroup
push    offset BinaryPathName ; "C:\\Windows\\System32\\Mlwx486.sys"
push    1                  ; dwErrorControl
push    3                  ; dwStartType
push    1                  ; dwServiceType
push    0F01FFh           ; dwDesiredAccess
push    offset DisplayName ; "486 WS Driver"
push    offset DisplayName ; "486 WS Driver"
push    eax                ; hSCManager
call     ds:CreateServiceA
mov     esi, eax
test    esi, esi
jnz     short loc_4010DC

```

然后利用这个文件创建了一个名为486 WS Driver的服务



创建结束以后自然就是打开已经创建好的服务

使用Procmon进一步观察可以看见

21:5...	Lab10-U2.exe	3876	CreateFile	C:\WINDOWS\system32\ntdll.dll
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\kernel32.dll
21:5...	Lab10-02.exe	3876	CreateFile	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\Lab10-02.exe
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\advapi32.dll
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\rpcrt4.dll
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\secur32.dll
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\apphelp.dll
21:5...	Lab10-02.exe	3876	CreateFile	C:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\Mlwx486.sys
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\conime.exe
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\apphelp.dll
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\apphelp.dll
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\AppPatch\sysmain.sdb
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\AppPatch\sysmain.sdb
21:5...	Lab10-02.exe	3888	CreateFile	C:\WINDOWS\Prefetch\LAB10-02.EXE-1732BEBB.pf

这里创建了另一个exe文件，当然除了这些还创建了很多其他的文件。

问题1

Does this program create any files? If so, what are they?

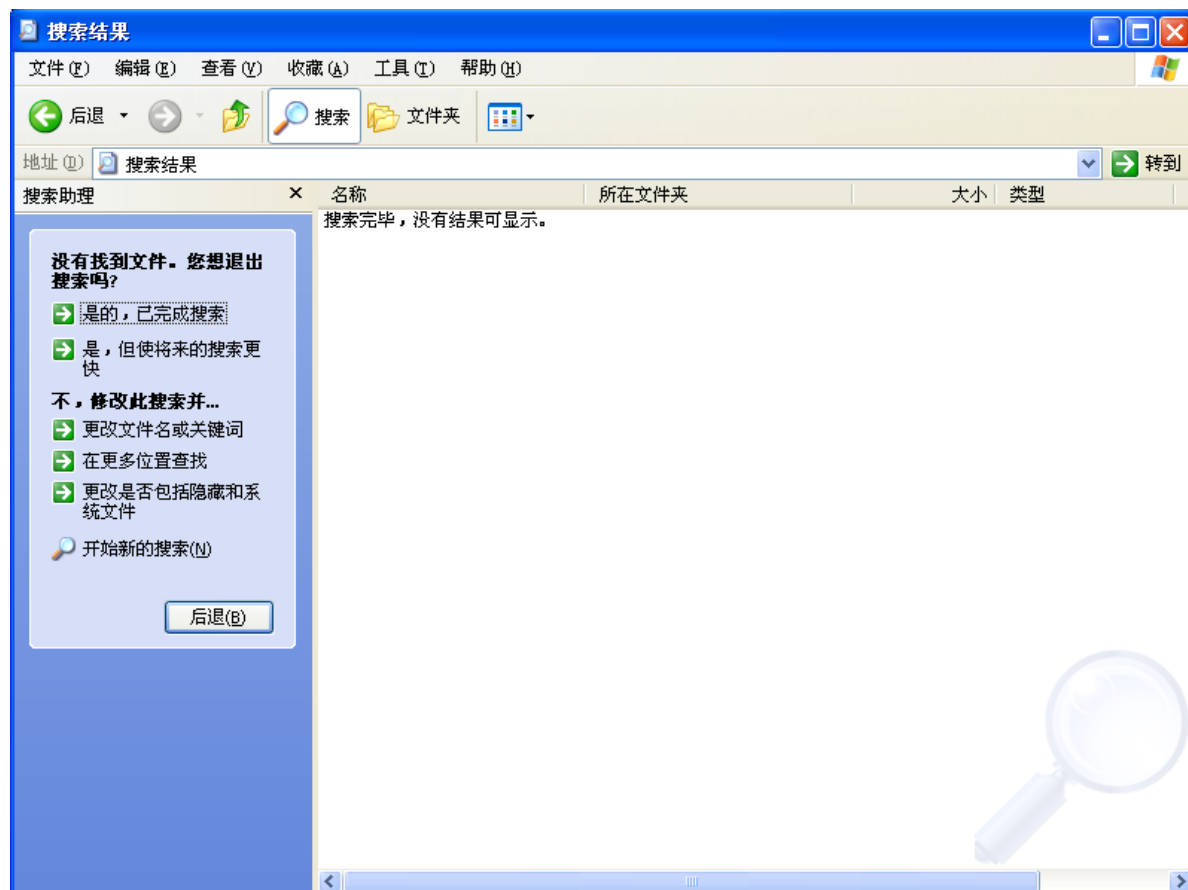
为了更直观的看见这个程序具体执行了哪些操作，我们使用Procmon进行监视

在设置过滤条件进行过滤之后可以看见

21:5...	Lab10-U2.exe	3876	CreateFile	U:\Documents and Settings\Administrator\桌面\上机实验样本\Chapter_10\
21:5...	Lab10-02.exe	3876	CreateFile	C:\WINDOWS\system32\Mlwx486.sys

这里创建了 C:\WINDOWS\system32\Mlwx486.sys 这个文件

但是当我们进入到这个文件夹下，想要找这个文件的时候，却发现这个目录下并没有这个文件



但是在使用cmd进行搜索的时候，我们看见这个目录下是有这样的一个文件的

```
C:\WINDOWS\system32>dir Ml*.sys
驱动器 C 中的卷没有标签。
卷的序列号是 1CEE-E82C

C:\WINDOWS\system32 的目录

2021-11-22  19:11                3,456 Mlwx486.sys
               1 个文件                3,456 字节
               0 个目录 36,361,113,600 可用字节
```

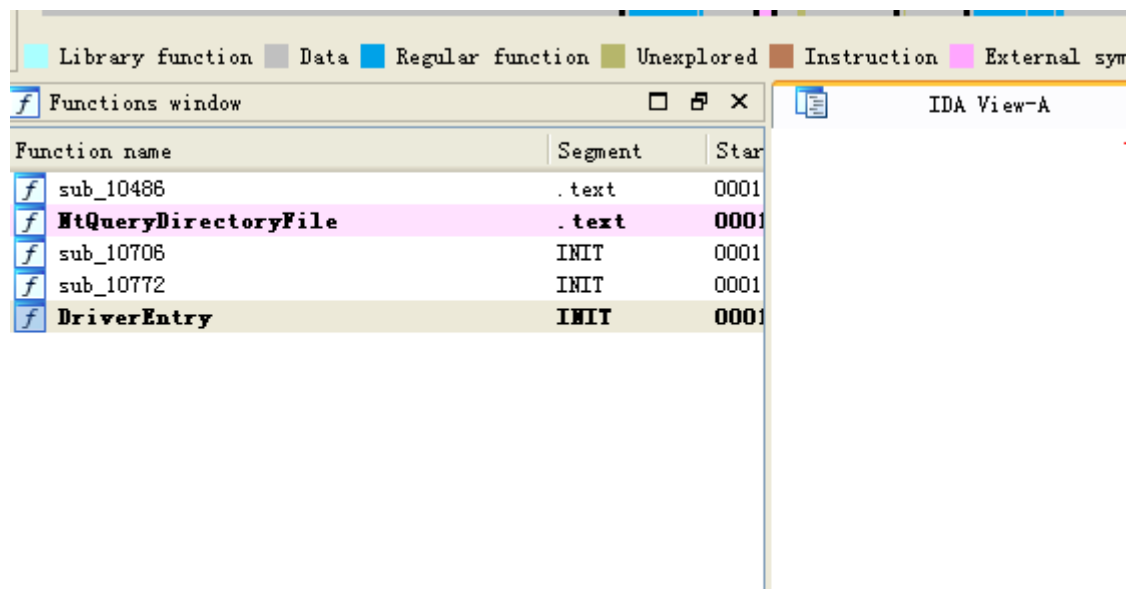
也就是说这个文件被隐藏起来了

问题2

Does this program have a kernel component?

使用resource hacker可以把资源节里的内容提取出来并保存

由于前2个字节是4D 5A，我们将其保存为.exe文件，并用IDA进行打开



打开以后我们可以看见这个的入口是DriverEntry，也就是说这个就是驱动程序了。

那么综上所述，Lab10-02.exe有一个内核模块，这个模块被存放在程序的资源节中，然后这个驱动程序会被写入到硬盘作为一个服务加载到内核。

问题3

What does this program do?

Lab10-02.exe这个程序的目的其实就是为了将驱动程序装载到内核、启动服务，也就是说这个程序的主要功能模块是在驱动程序中，所以这里具体分析一下驱动程序都干了什么

```
; Attributes: bp-based frame

; NTSTATUS __stdcall DriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath)
public DriverEntry
DriverEntry proc near

DriverObject= dword ptr 8
RegistryPath= dword ptr 0Ch

mov     edi, edi
push    ebp
mov     ebp, esp
call    sub_10772
pop     ebp
jmp     sub_10706
DriverEntry endp
```

可以看到这个驱动程序主要功能位置是sub_10706函数，进入到这个函数体内部


```

push    esi
mov     esi, ds:RtlInitUnicodeString
push    edi
push    offset SourceString ; "N"
lea     eax, [ebp+DestinationString]
push    eax                ; DestinationString
call    esi ; RtlInitUnicodeString
push    offset aKeservicedescr ; "KeServiceDescriptorTable"
lea     eax, [ebp+SystemRoutineName]
push    eax                ; DestinationString
call    esi ; RtlInitUnicodeString
mov     esi, ds:MmGetSystemRoutineAddress
lea     eax, [ebp+DestinationString]
push    eax                ; SystemRoutineName
call    esi ; MmGetSystemRoutineAddress
mov     edi, eax
lea     eax, [ebp+SystemRoutineName]
push    eax                ; SystemRoutineName
call    esi ; MmGetSystemRoutineAddress
mov     eax, [eax]
xor     ecx, ecx

```

恶意软件调用MmGetSystemRoutineAddress 以获取指向 NtQueryDirectoryFile 和 KeServiceDescriptorTable 子例程的指针。然后它遍历服务描述符表，寻找NtQueryDirectoryFile 的地址。一旦找到，它将用恶意钩子（自定义子程序）来覆盖地址。

在驱动程序中，**NTQueryDirectoryFile** 函数被调用

根据 MSDN，此函数返回有关给定文件句柄指定的目录中的文件的各种信息。

再往下，我们可以看到对 RtlCompareMemory 的调用。文件名和&word_1051A 处的字符串之间进行了比较。如果匹配，则隐藏文件。&word_1051A 指向的地址内容如下：

```

.text:0001050B          ; -----
.text:0001050E  CC CC CC CC CC CC          db 6 dup(0CCh)
.text:00010514          ; [00000006 BYTES: COLLAPSED FUNCTION NtQueryDirectoryFile. PRESS KEYPAD CTRL-"" TO EXPAND]
.text:0001051A  4D 00 6C 00 77 00 78 00 word_1051A dw 'M', 'l', 'w', 'x' ; DATA XREF: sub_10486+46To
.text:00010522  00          db 0
.text:00010523  00          db 0
.text:00010524  00 00 00 00 00 00 00 00+ align 80h
.text:00010524  00 00 00 00 00 00 00 00+_text ends
.text:00010524  00 00 00 00 00 00 00 00+

```

所以说&word_1051A 字符串就是“MlwX”。要查看所有这些 win 操作，需要启动 Windbg 并将其附加到内核。

使用命令 dps nt!KiServiceTable l 100 列出服务描述符表。此表尚未被篡改：

80501d6c	8060d896	nt!NtOpenMutant
80501d70	805ea704	nt!NtOpenObjectAuditAlarm
80501d74	805c1296	nt!NtOpenProcess
80501d78	805e39fc	nt!NtOpenProcessToken
80501d7c	805e3660	nt!NtOpenProcessTokenEx
80501d80	8059f722	nt!NtOpenSection
80501d84	8060b254	nt!NtOpenSemaphore
80501d88	805b977a	nt!NtOpenSymbolicLinkObject
80501d8c	805c1522	nt!NtOpenThread
80501d90	805e3a1a	nt!NtOpenThreadToken
80501d94	805e37d0	nt!NtOpenThreadTokenEx
80501d98	8060d1b0	nt!NtOpenTimer
80501d9c	8063bc78	nt!NtPlugPlayControl
80501da0	805bf346	nt!NtPowerInformation
80501da4	805eddce	nt!NtPrivilegeCheck
80501da8	805e9a16	nt!NtPrivilegeObjectAuditAlarm
80501dac	805e9c02	nt!NtPrivilegedServiceAuditAlarm
80501db0	805ada08	nt!NtProtectVirtualMemory
80501db4	806052dc	nt!NtPulseEvent
80501db8	8056c0ce	nt!NtQueryAttributesFile
80501dbc	8060cb50	nt!NtSetBootEntryOrder
80501dc0	8060cb50	nt!NtSetBootEntryOrder
80501dc4	8053c02e	nt!NtQueryDebugFilterState
80501dc8	80606e68	nt!NtQueryDefaultLocale
80501dcc	80607ac8	nt!NtQueryDefaultUILanguage
80501dd0	8056f074	nt!NtQueryDirectoryFile
80501dd4	805b3de0	nt!NtQueryDirectoryObject
80501dd8	8056f3ca	nt!NtQueryEaFile
80501ddc	806053a4	nt!NtQueryEvent
80501de0	8056c222	nt!NtQueryFullAttributesFile
80501de4	8060c2dc	nt!NtQueryInformationAtom
80501de8	8056fc46	nt!NtQueryInformationFile
80501dec	805cbee0	nt!NtQueryInformationJobObject
80501df0	8059a6fc	nt!NtQueryInformationPort
80501df4	805c2bfc	nt!NtQueryInformationProcess
80501df8	805c17c8	nt!NtQueryInformationThread
80501dfc	805e3afa	nt!NtQueryInformationToken
80501e00	80607266	nt!NtQueryInstallUILanguage
80501e04	8060e060	nt!NtQueryIntervalProfile
80501e08	8056ddda	nt!NtQueryIoCompletion
80501e0c	8061b97e	nt!NtQueryKey
80501e10	806193d4	nt!NtQueryMultipleValueKey

kd>

使用 `bu M!wx486!DriverEntry` 命令设置断点。运行 Lab10-02.exe, windbg 会中断。在 `nt!IopLoadDriver+0x66a` 处设置断点并让程序再次运行。一旦内核中断, 就可以运行 `!object \Driver` 列出加载的驱动程序。恶意代码的 `DriverInit` 在此阶段尚未被执行, 因此可以在此设置断点

```

nt!DbgLoadImageSymbols+0x42:
80527e02 c9          leave
kd> gu
nt!MmLoadSystemImage+0xa80:
805a41f4 804b3610      or          byte ptr [ebx+36h],10h
kd> gu
nt!IopLoadDriver+0x371:
80576483 3bc3         cmp         eax,ebx
kd> gu
nt!IopLoadUnloadDriver+0x45:
8057688f 8bf8         mov         edi,eax
kd> !object \Driver\
Object: e101d910  Type: (8a360418) Directory
ObjectHeader: e101d8f8 (old version)
HandleCount: 0  PointerCount: 85
Directory Object: e1001150  Name: Driver

Hash Address  Type      Name
----
00  8a0f46e8 Driver    Beep
    8a0eble0 Driver    NDIS
    8a0ebd28 Driver    KSecDD
01  8a191520 Driver    Mouclass
    89de1030 Driver    Raspti
    89e43eb0 Driver    es1371
02  8a252c98 Driver    vmx_svga
03  8a0a3da0 Driver    Fire
kd>

```

而后再次运行 `kd> dps nt!KiServiceTable 1 100`，能够看出服务描述符表已被修改

```

00501dd0 00000000 nt!NtSetBootEntryOrder
80501dc0 8060cb50 nt!NtSetBootEntryOrder
80501dc4 8053c02e nt!NtQueryDebugFilterState
80501dc8 80606e68 nt!NtQueryDefaultLocale
80501dcc 80607ac8 nt!NtQueryDefaultUILanguage
80501dd0 baecb486 Mlwx486+0x486
80501dd4 805b3de0 nt!NtQueryDirectoryObject
80501dd8 8056f3ca nt!NtQueryEaFile
80501ddc 806053a4 nt!NtQueryEvent
80501de0 8056c222 nt!NtQueryFullAttributesFile
80501de4 8060c2dc nt!NtQueryInformationAtom
80501de8 8056fc46 nt!NtQueryInformationFile
80501dec 805cbee0 nt!NtQueryInformationJobObject
80501df0 8059a6fc nt!NtQueryInformationPort
80501df4 805c2bfc nt!NtQueryInformationProcess
80501df8 805c17c8 nt!NtQueryInformationThread
80501dfc 805e3afa nt!NtQueryInformationToken
kd>

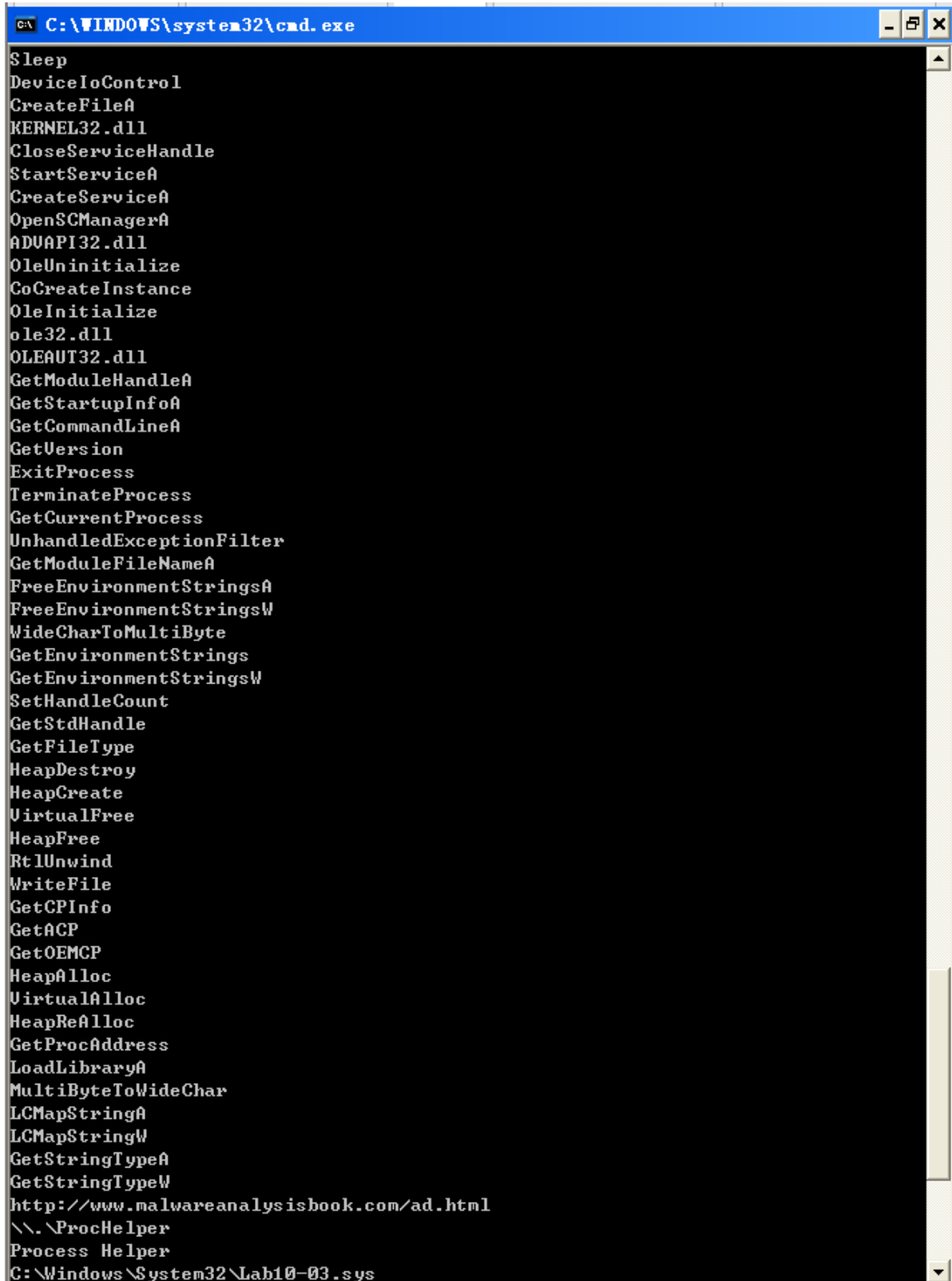
```

从上述的分析来看，这个程序是被用来隐藏文件的Rootkit，并且他是使用服务描述符表的Hook来隐藏以“Mlwx”开头的文件。

Lab 10-3

部分实验过程

首先使用strings工具简单查看一下exe文件



```
C:\WINDOWS\system32\cmd.exe

Sleep
DeviceIoControl
CreateFileA
KERNEL32.dll
CloseServiceHandle
StartServiceA
CreateServiceA
OpenSCManagerA
ADVAPI32.dll
OleUninitialize
CoCreateInstance
OleInitialize
ole32.dll
OLEAUT32.dll
GetModuleHandleA
GetStartupInfoA
GetCommandLineA
GetVersion
ExitProcess
TerminateProcess
GetCurrentProcess
UnhandledExceptionFilter
GetModuleFileNameA
FreeEnvironmentStringsA
FreeEnvironmentStringsW
WideCharToMultiByte
GetEnvironmentStrings
GetEnvironmentStringsW
SetHandleCount
GetStdHandle
GetFileType
HeapDestroy
HeapCreate
VirtualFree
HeapFree
RtlUnwind
WriteFile
GetCPInfo
GetACP
GetOEMCP
HeapAlloc
VirtualAlloc
HeapReAlloc
GetProcAddress
LoadLibraryA
MultiByteToWideChar
LCMapStringA
LCMapStringW
GetStringTypeA
GetStringTypeW
http://www.malwareanalysisbook.com/ad.html
\\.\ProcHelper
Process Helper
C:\Windows\System32\Lab10-03.sys
```

可以看见这个程序里有对服务的创建、打开等一套操作，同时还有一些以前没有见过的函数，比如获取环境中的字符串；然后可以看到有一个http的网址，结合之前的创建服务，猜测服务中可能会包含有对这个URL的访问。

再查看一下sys文件

```

c:\winddk\7600.16385.1\src\general\rootkitprochide\wdm\sys\objfre_wxp_x86\i386\si
ioctl.pdb

\DosDevices\ProcHelper
FCh
Fpf
F4*
EPE

NE

\DosDevices\ProcHelper
\Device\ProcHelper
IoCompleteRequest
IoDeleteDevice
IoDeleteSymbolicLink
RtlInitUnicodeString
IoGetCurrentProcess
IoCreateSymbolicLink
IoCreateDevice
KeTickCount
ntoskrnl.exe
US_VERSION_INFO
StringFileInfo
040904B0
CompanyName
Windows (R) Win 7 DDK provider
FileDescription
Important Process Helper
FileVersion
6.1.7600.16385 built by: WinDDK
InternalName
Lab10-03.sys
LegalCopyright
ABC Corp.
OriginalFilename
Lab10-03.sys
ProductName
Windows (R) Win 7 DDK driver
ProductVersion
6.1.7600.16385
VarFileInfo
Translation
696C6M6X6m6
7>7G7R717d7t7

```

可以看见在sys文件中出现了InternalName字符串，结合之前exe中的猜测，想来可能是会进行访问。其他的从这里也看不出来什么了。

接下来使用IDA简单查看一下功能

```

push    0                ; lpLoaderOrderGroup
push    offset BinaryPathName ; "C:\\Windows\\System32\\Lab10-03.sys"
push    1                ; dwErrorControl
push    3                ; dwStartType
push    1                ; dwServiceType
push    0F01FFh          ; dwDesiredAccess
push    offset DisplayName ; "Process Helper"
push    offset DisplayName ; "Process Helper"
push    eax              ; hSCManager
call    ds:CreateServiceA
mov     esi, eax
test    esi, esi
jz      short loc_401057

```

```

push    0                ; lpServiceArgVectors
push    0                ; dwNumServiceArgs
push    esi              ; hService
call    ds:StartServiceA

```

```

loc_401057:                ; hSCObject
push    esi
call    ds:CloseServiceHandle
push    0                ; hTemplateFile

```

可以看到前面和之前一样是创建一个服务，服务的名字是Process Helper，并且设置的驱动文件路径和之前lab 10-01是同一个目录下。之后是熟悉的打开服务。

接下来的内容和之前就有些不一样了

```

push    0                ; dwIoControlCode
push    eax              ; hDevice
call    ds:DeviceIoControl
push    0                ; pvReserved
call    ds:OleInitialize
test    eax, eax
jl      short loc_401131

```

```

lea     edx, [esp+2Ch+ppv]
push    edi
push    edx              ; ppv
push    offset riid      ; riid
push    4                ; dwClsContext
push    0                ; pUnkOuter
push    offset rclsid     ; rclsid
call    ds:CoCreateInstance
mov     eax, [esp+30h+ppv]
test    eax, eax
jz      short loc_40112A

```

```

lea     eax, [esp+30h+pvarg]
push    eax              ; pvarg
call    ds:VariantInit
push    offset psz        ; "http://www.malwareanalysisbook.com/ad.h"...
mov     [esp+34h+var_10], 3
mov     [esp+34h+var_8], 1
call    ds:SysAllocString
mov     edi, ds:Sleep
mov     esi, eax

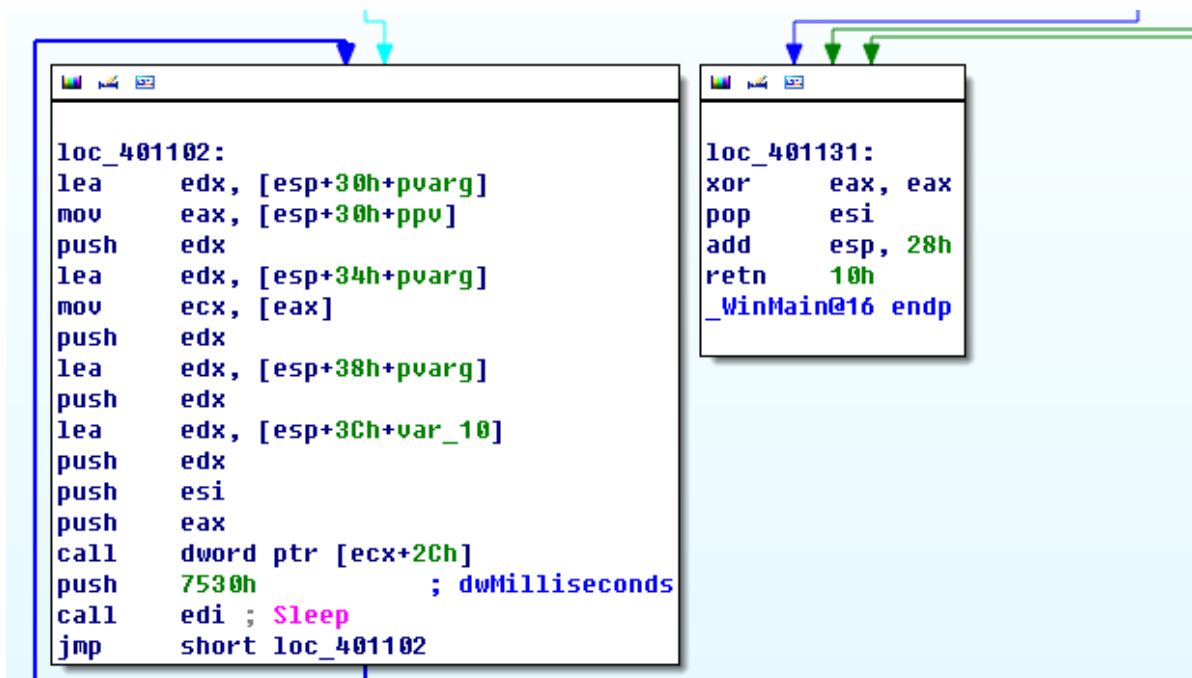
```

```

loc_40112A:
call    ds:OleUninitialize
pop     edi

```

可以看见这里压入了之前分析的url，然后会打开一个网页，之后调用sleep



可以看见这里是一个循环，每次sleep时间是0x7530h ms

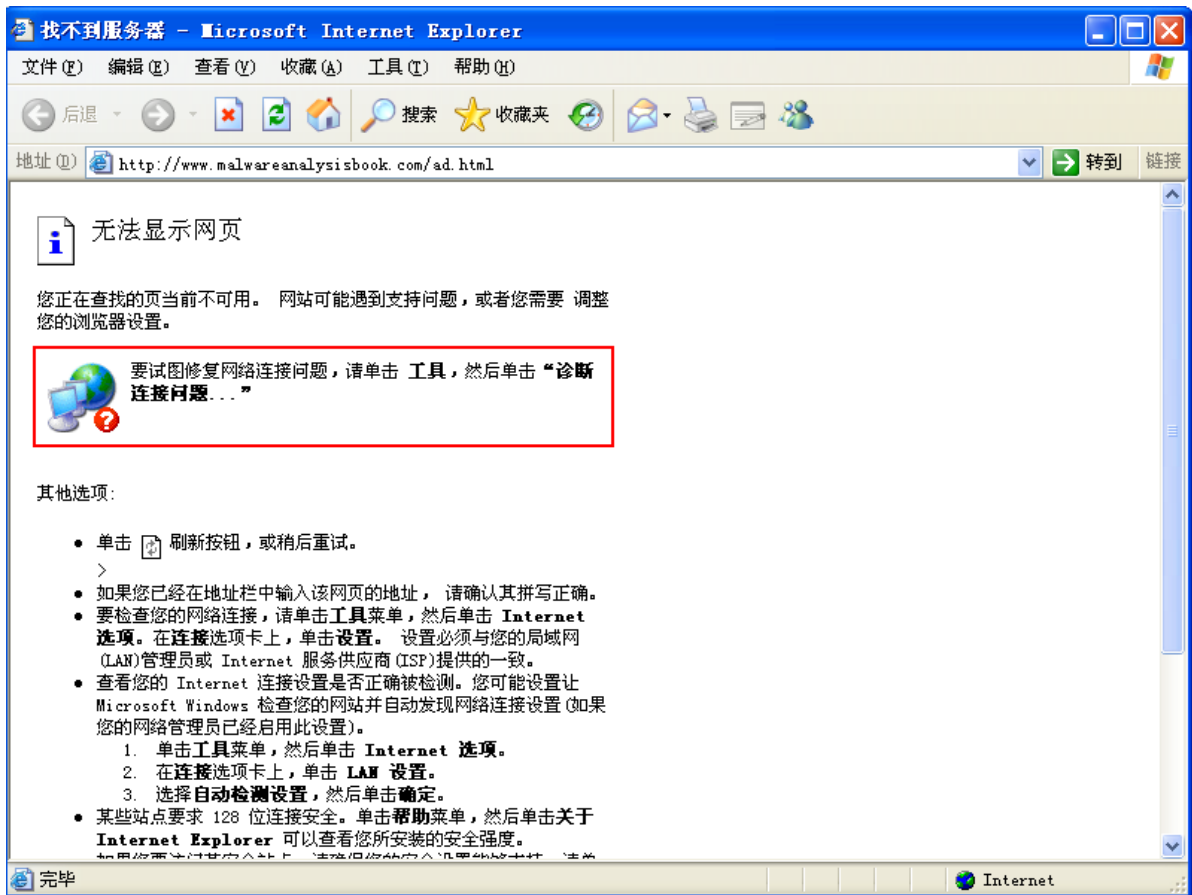
经过资料查询，可以看到有个DeviceIoControl函数的调用，这个函数有一个比较特殊的地方就是作为参数传递给她的输入和输出参数，将会被发送到内核中去，所以需要特别关注一下这个函数的参数。

```
loc_40108C:
lea     ecx, [esp+2Ch+BytesReturned]
push    0               ; lpOverlapped
push    ecx             ; lpBytesReturned
push    0               ; nOutBufferSize
push    0               ; lpOutBuffer
push    0               ; nInBufferSize
push    0               ; lpInBuffer
push    0ABCDEF01h      ; dwIoControlCode
push    eax             ; hDevice
call    ds:DeviceIoControl
push    0               ; pvReserved
call    ds:OleInitialize
test    eax, eax
jl      short loc_401131
```

可以看见参数中LpOutBuffer和LpInputBuffer都被设置为空，然后在dwIoControlCode这里的参数也是有点特殊，传递进去的是0ABCDEF01h，具体作用在内核位置再进行分析。

尝试使用Procmon查看一下这个程序具体的影响

首先在双击运行的时候，可以看见弹出了一个浏览器窗口

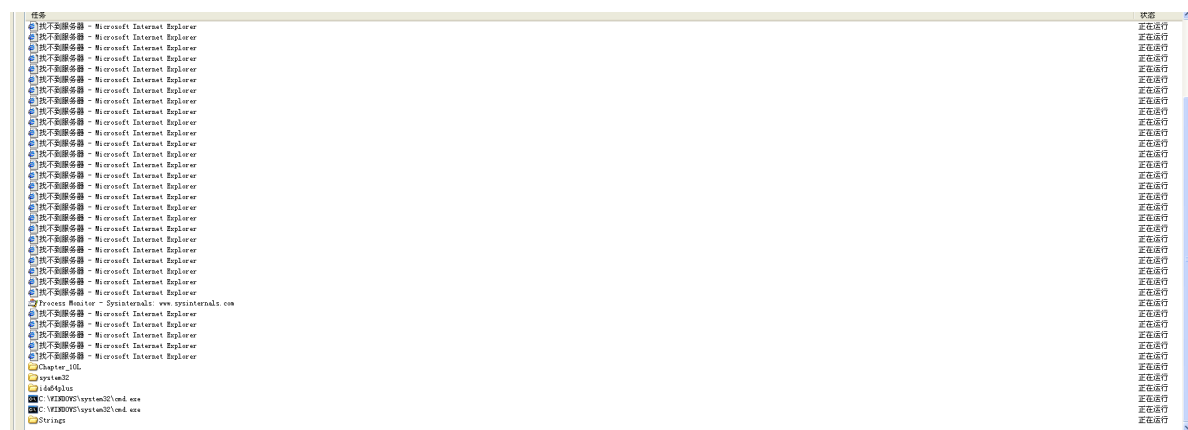


观察可以发现打开的就是之前我们在strings中看见的url, 同时可以注意到每过一段时间都会打开一个这个页面, 也就是上面分析的30s。

查看Procmon中的结果, 可以看到有很多对注册表、文件的操作

Process Monitor - Sysinternals: www.sysinternals.com						
File Edit Event Filter Tools Options Help						
Time	Process Name	PID	Operation	Path	Result	Detail
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\shdocvw.dll	SUCCESS	Offset: 899,5...
19:5...	Lab10-03.exe	4020	QueryStandardInfo...	C:\WINDOWS\system32\shdocvw.dll	SUCCESS	AllocationSiz...
19:5...	Lab10-03.exe	4020	CreateFileMapping	C:\WINDOWS\system32\shdocvw.dll	SUCCESS	SyncType: Syn...
19:5...	Lab10-03.exe	4020	QueryStandardInfo...	C:\WINDOWS\system32\shdocvw.dll	SUCCESS	AllocationSiz...
19:5...	Lab10-03.exe	4020	CreateFileMapping	C:\WINDOWS\system32\shdocvw.dll	SUCCESS	SyncType: Syn...
19:5...	Lab10-03.exe	4020	RegCloseKey	HKCR\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1\0\win32	SUCCESS	
19:5...	Lab10-03.exe	4020	RegCloseKey	HKCR\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1\0	SUCCESS	
19:5...	Lab10-03.exe	4020	RegCloseKey	HKCR\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1	SUCCESS	
19:5...	Lab10-03.exe	4020	RegQueryValue	HKCR\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}	SUCCESS	
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCU\Software\Classes\TypeLib	NAME NOT FOUND	Query: Name
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCR\TypeLib	NAME NOT FOUND	Desired Acces...
19:5...	Lab10-03.exe	4020	RegQueryValue	HKCR\TypeLib	NAME NOT FOUND	Query: Name
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCU\Software\Classes\TypeLib\{00020430-0000-0000-C000-000000000004}	NAME NOT FOUND	Desired Acces...
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}	SUCCESS	Desired Acces...
19:5...	Lab10-03.exe	4020	RegQueryValue	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}	SUCCESS	Query: Name
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCU\Software\Classes\TypeLib\{00020430-0000-0000-C000-000000000004}	NAME NOT FOUND	Desired Acces...
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0	SUCCESS	Desired Acces...
19:5...	Lab10-03.exe	4020	RegQueryValue	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0	SUCCESS	Query: Name
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCU\Software\Classes\TypeLib\{00020430-0000-0000-C000-000000000004}	NAME NOT FOUND	Desired Acces...
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0	SUCCESS	Desired Acces...
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCU\Software\Classes\TypeLib\{00020430-0000-0000-C000-000000000004}	NAME NOT FOUND	Desired Acces...
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0\win32	SUCCESS	Desired Acces...
19:5...	Lab10-03.exe	4020	RegCloseKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0\win32	SUCCESS	
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0	SUCCESS	Query: Name
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCU\Software\Classes\TypeLib\{00020430-0000-0000-C000-000000000004}	NAME NOT FOUND	Desired Acces...
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0	SUCCESS	Desired Acces...
19:5...	Lab10-03.exe	4020	RegQueryValue	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0	SUCCESS	Query: Name
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCU\Software\Classes\TypeLib\{00020430-0000-0000-C000-000000000004}	NAME NOT FOUND	Desired Acces...
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0\win32	SUCCESS	Desired Acces...
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCU\Software\Classes\TypeLib\{00020430-0000-0000-C000-000000000004}	NAME NOT FOUND	Desired Acces...
19:5...	Lab10-03.exe	4020	RegOpenKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0\win32	SUCCESS	Type: REG_SZ...
19:5...	Lab10-03.exe	4020	CreateFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Desired Acces...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 0, Le...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 192, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 196, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 440, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 512, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 528, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 672, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 674, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 544, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 560, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 592, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 608, ...
19:5...	Lab10-03.exe	4020	ReadFile	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	Offset: 640, ...
19:5...	Lab10-03.exe	4020	QueryStandardInfo...	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	AllocationSiz...
19:5...	Lab10-03.exe	4020	CreateFileMapping	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	SyncType: Syn...
19:5...	Lab10-03.exe	4020	QueryStandardInfo...	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	AllocationSiz...
19:5...	Lab10-03.exe	4020	CreateFileMapping	C:\WINDOWS\system32\stdole2.tlb	SUCCESS	SyncType: Syn...
19:5...	Lab10-03.exe	4020	RegCloseKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0\0	SUCCESS	
19:5...	Lab10-03.exe	4020	RegCloseKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}\2.0	SUCCESS	
19:5...	Lab10-03.exe	4020	RegCloseKey	HKCR\TypeLib\{00020430-0000-0000-C000-000000000004}	SUCCESS	

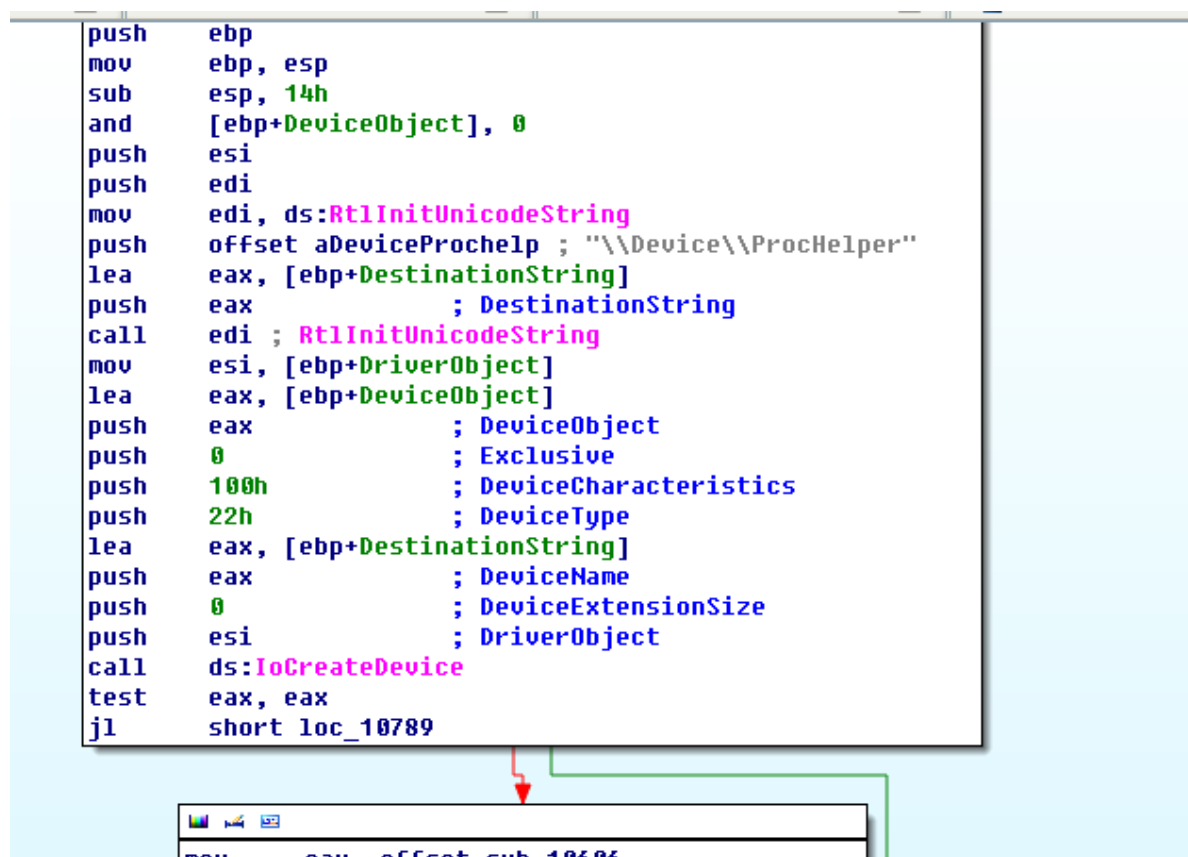
由于这个程序是会sleep，然后每隔一段时间就打开一次浏览器，所以在进程管理器里查看一下他的常驻进程名是什么



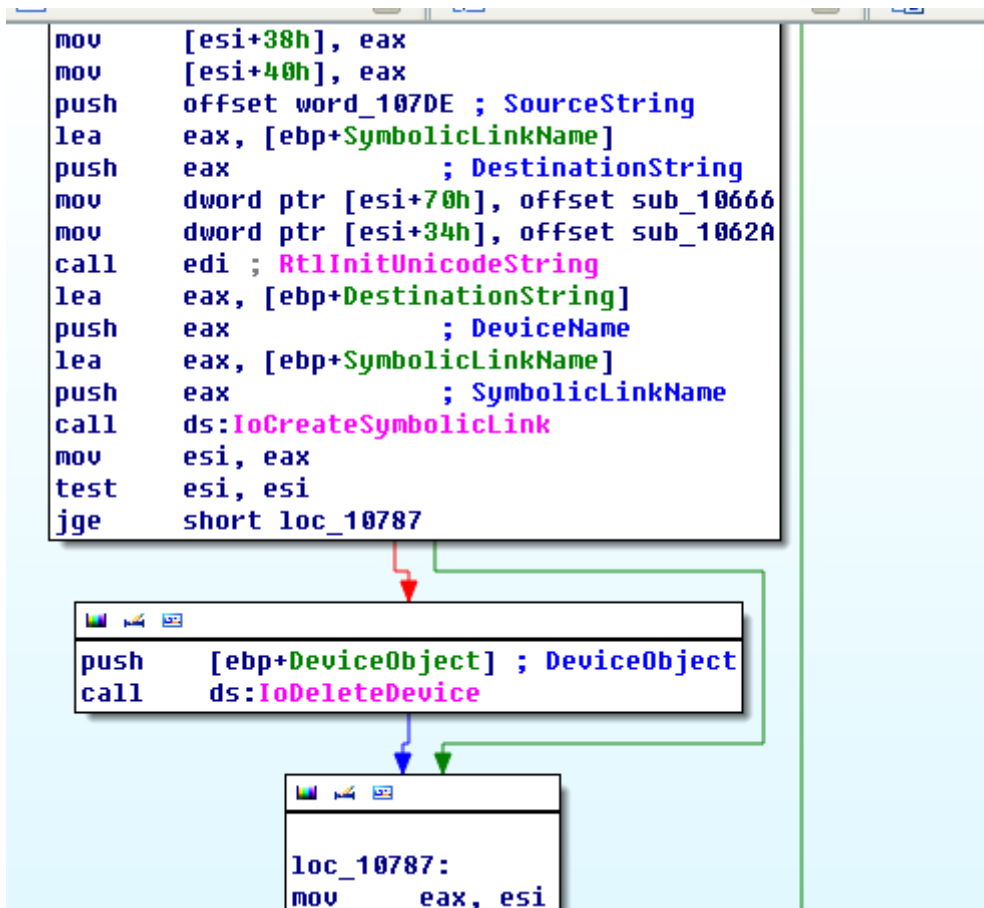
可是意外的是，在进程管理器里并没有看见关于这个进程的信息，也就是说想要从进程管理器里关闭这个进程从而达到关闭浏览器弹出是不可行的。具体是如何隐藏进程信息的在之后会进行分析。

IDA分析驱动

和之前一样，可以看见这个驱动真正的入口点不是在一开始的Entry处，而是在jmp后面的地址，直接跳转到那个地址进行分析



可以看见驱动调用了函数IoCreateDevice，创建了一个名为\Device\ProcHelper的设备



之后在下面调用了IoCreateSymbolicLink来创建了一个名为\DosDevices\ProHelper的符号链接，来提供用户态的应用程序访问。

同时我们还可以看见调用了RtlInitUnicodeString函数，这个函数有两个参数，其中第一个参数是这个驱动的名字：`DosDevices\ProHelper`，第二个函数是eax中的值，从边上的注释可以看出这个参数是DestinationString

最后一个函数是IoDeleteDevice，这个函数就是用来删除驱动程序。

注意到以下代码：

```

mov     eax, offset sub_10606
mov     [esi+38h], eax
mov     [esi+40h], eax
push    offset word_107DE ; SourceString
lea     eax, [ebp+SymbolicLinkName]
push    eax                ; DestinationString
mov     dword ptr [esi+70h], offset sub_10666
mov     dword ptr [esi+34h], offset sub_1062A
call    edi ; RtlInitUnicodeString

```

这里将3个函数的偏移地址放到了内存中，其中第一个位置将 MajorFunction [IRP_MJ_DEVICE_CONTROL]设置为 0x10666

根据 MSDN，IRP_MJ_DEVICE_CONTROL 请求，是由 I/O 管理器、其他操作系统组件或其他内核模式驱动程序发送的。通常，此 IRP 代表已调用 Microsoft Win32 DeviceIoControl 函数的用户模式应用程序或代表已调用 ZwDeviceIoControlFile 的内核模式组件

注意到10666位置处的函数获取了当前进程，猜测这里和隐藏进程有关，主要分析一下这一段的代码

```

mov     edi, edi
push    ebp
mov     ebp, esp
call    ds:IoGetCurrentProcess
mov     ecx, [eax+8Ch]
add     eax, 88h
mov     edx, [eax]
mov     [ecx], edx
mov     ecx, [eax]
mov     eax, [eax+4]
mov     [ecx+4], eax
mov     ecx, [ebp+Irp] ; Irp
and     dword ptr [ecx+18h], 0
and     dword ptr [ecx+1Ch], 0
xor     dl, dl ; PriorityBoost
call    ds:IoCompleteRequest
xor     eax, eax
pop     ebp
retn    8
sub_10666 endp

```

mov ecx, [eax+8Ch]指令获取列表中指向下一项的指针。

mov edx, [eax]指令获取列表中指向前一项的指针。

mov [ecx], edx 指令覆盖下一项的 BLINK 指针，使其指向前一项，也就是

mov [ecx], edx 指令。在此之前，下一项的 BLINK 指针指向当前项 mov ecx, [eax]。mov [ecx], edx 会覆盖 BLINK 指针，从而使它跳过当前指针。

mov ecx, [eax]; mov eax, [eax+4]; mov [ecx+4], eax;指令执行相同的步骤，除了覆盖列表中前一项的 FIINK 指针来跳过当前项。

除了修改当前进程的 EPROCESS 结构之外，上述代码还会修改进程链中的前一个或者后一个进程的 EPROCESS 结构。这个六条指令通过从加载进程的列表中解除链接，来隐藏当前进程。其实也就是在获取进程的时候，将链表的结构断开，然后将左右两边的节点连接起来，从而达到隐藏的效果。

问题1

What does this program do?

程序装载了一个驱动，并且设置了每间隔30s就弹出一个浏览器页面。同时意外的发现这个进程在进程管理器中是被隐藏的，无法从进程管理器中直接关闭这个进程。

问题2

Once this program is running, how do you stop it?

从之前的行为可以发现，想从进程管理器里关闭这个进程是做不到的，因为他被隐藏起来了，所以需要重启电脑或恢复快照（虚拟机）。由于这个程序并没有设置开机自启动，所以直接重启电脑是能够关闭的。

问题3

What does the kernel component do?

这个驱动程序创建了名为\Device\ProcHelper的设备，并以 \\Device \\ProcHelper 作为其符号链接。之后修改进程管理结构中的值，从链表的角度来看就是将当前进程节点从链表上断开，将左右的节点连接在一起，从而达到隐藏当前进程的效果。

Yara

根据内容编写yara规则如下:

```
1  import "pe"
2
3  rule UrlRequest {
4      strings:
5          $http = "http"
6          $com = /[a-zA-Z0-9_]*.com/
7      condition:
8          $http or $com
9  }
10
11 rule EXE {
12     strings:
13         $exe = /[a-zA-Z0-9_]*.exe/
14     condition:
15         $exe
16 }
17
18 rule Regedit {
19     strings:
20         $system = "Registry"
21         $software = "SOFTWARE"
22     condition:
23         $system or $software
24 }
25
26 rule DriverFile {
27     strings:
28         $name = ".sys"
29     condition:
30         $name
31 }
32
33 rule Device {
34     strings:
35         $name = "Device"
36     condition:
37         $name
38 }
39
40 rule Service {
41     strings:
42         $create = "CreateService"
43         $start = "StartService"
44     condition:
45         $create or $start
46 }
47
48 rule ResourceFile {
49     strings:
50         $name = ".rsrc"
51     condition:
```

```
52 |         $name
53 |     }
```

实验结果如下

```
D:\Study\terms\3. Junior\FirstSemester\计算机病毒与防治技术（王志）\homework>yara64.exe -r yara_rules/lab10.yar Chapter_10L
DriverFile Chapter_10L\Lab10-03.exe
Device Chapter_10L\Lab10-03.exe
Service Chapter_10L\Lab10-03.exe
DriverFile Chapter_10L\Lab10-01.exe
Service Chapter_10L\Lab10-01.exe
ResourceFile Chapter_10L\Lab10-01.exe
EXE Chapter_10L\Lab10-01.sys
Regedit Chapter_10L\Lab10-01.sys
ResourceFile Chapter_10L\Lab10-01.sys
EXE Chapter_10L\Lab10-03.sys
Device Chapter_10L\Lab10-03.sys
ResourceFile Chapter_10L\Lab10-03.sys
EXE Chapter_10L\Lab10-02.exe
DriverFile Chapter_10L\Lab10-02.exe
Service Chapter_10L\Lab10-02.exe
ResourceFile Chapter_10L\Lab10-02.exe
```