恶意代码分析与防治技术实验报告——lab1

> **学号：2013921**
> **姓名：周延霖**
> **专业：信息安全**

实验环境

由于本人的本机是macOS Montery 12.4，所以在本机上对Windows的.exe文件的漏洞分析较为困难，于是我选择用VMware Fusion这个虚拟机管理软件来运行Windows XP操作系统来完成本次对四种恶意代码分析的实验（但对于在VirusTotal上网站进行的分析是直接从本机上上传的）

实验工具

由于本次是第一次实验，所以用到的分析工具都较为基础，并且用到的工具较少，现将其列举如下：

- PEview
- UPX
- Resource Hacker
- PEiD

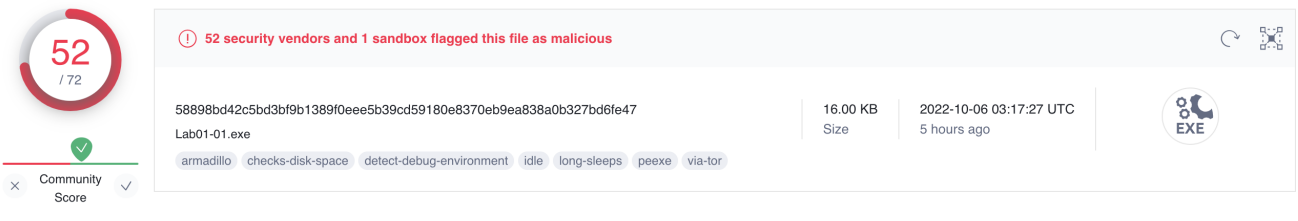
实验内容

lab1-1

这个实验使用Lab01-01.exe和Lab01-01.dll文件，使用本章描述的工具和技术来获取关于这些文件的信息.

> Q1.将文件上传至VirusTotal进行分析并查看报告。文件匹配到了已有的反病毒软件特征吗?

上传可执行文件后的结果如下图所示：



可以看到有52个安全供应商和1个沙盒将其标记为恶意程序

> Q2.这些文件是什么时候编译的?

使用PEview工具打开.exe和.dll文件。对于每个文件，观察IMAGE_NT_HEADERS->IMAGE_ FILE_ HEADER->Time Date Stamp字段，这个字段显示了文件的编译时间，如以下两张图片所示：

pFile	Data	Description	Value
000000EC	014C	Machine	IMAGE_FILE_MACHINE_I386
000000EE	0003	Number of Sections	
000000F0	4D0E2FD3	Time Date Stamp	2010/12/19 16:16:19 UTC
000000F4	00000000	Pointer to Symbol Table	
000000F8	00000000	Number of Symbols	
000000FC	00E0	Size of Optional Header	
000000FE	010F	Characteristics	
		0001	IMAGE_FILE_RELOCS_STRIPPED
		0002	IMAGE_FILE_EXECUTABLE_IMAGE
		0004	IMAGE_FILE_LINE_NUMS_STRIPPED
		0008	IMAGE_FILE_LOCAL_SYMS_STRIPPED
		0100	IMAGE_FILE_32BIT_MACHINE

pFile	Data	Description	Value
000000E4	014C	Machine	IMAGE_FILE_MACHINE_I386
000000E6	0004	Number of Sections	
000000E8	4D0E2FE6	Time Date Stamp	2010/12/19 16:16:38 UTC
000000EC	00000000	Pointer to Symbol Table	
000000F0	00000000	Number of Symbols	
000000F4	00E0	Size of Optional Header	
000000F6	210E	Characteristics	
		0002	IMAGE_FILE_EXECUTABLE_IMAGE
		0004	IMAGE_FILE_LINE_NUMS_STRIPPED
		0008	IMAGE_FILE_LOCAL_SYMS_STRIPPED
		0100	IMAGE_FILE_32BIT_MACHINE
		2000	IMAGE_FILE_DLL

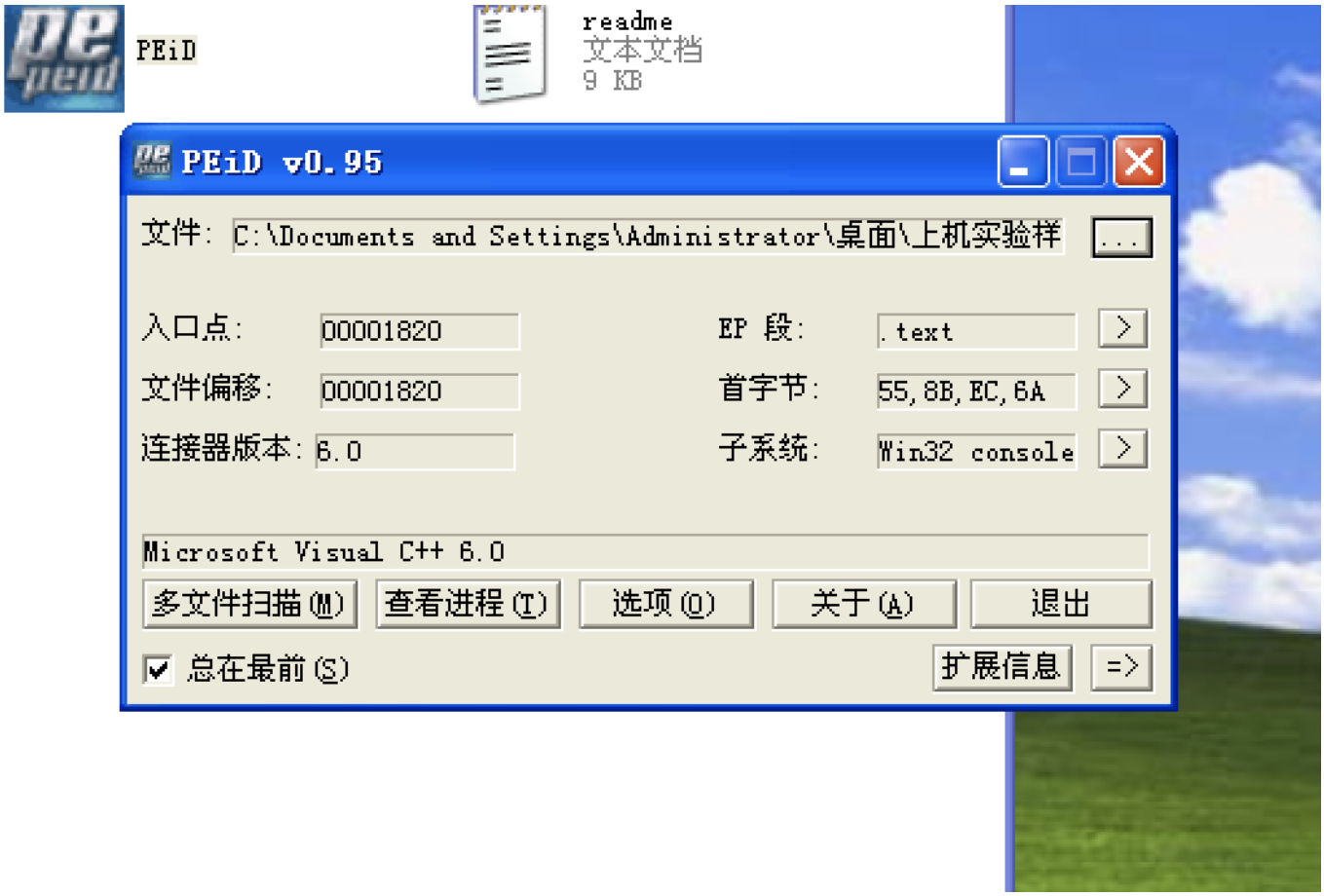
这两个文件都是在2010年12月19日被编译的，两者编译时间在1分钟以内。这证实了认为这两个文件同属一个恶意代码包的怀疑。事实上，编译时间非常接近说明是由同一作者在同时间创建了这些文件。所以这些文件是关联的，因为它们的编译时间与被发现的位置。这个.exe文件可能是用来使用或安装.dll文件的，因为DLL动态链接库文件无法运行自己。

> Q3. 这两个文件中是否存在迹象说明它们是否被加壳或混淆了？如果是，这些迹象在哪里？

这两个文件的导入表数量都很少，也都有着适当大小的良好组织的文件节，如下图所示：

pFile	Data	Description	Value
00002000	00002116	Hint/Name RVA	0296 Sleep
00002004	0000211E	Hint/Name RVA	0044 CreateProcessA
00002008	00002130	Hint/Name RVA	003F CreateMutexA
0000200C	00002140	Hint/Name RVA	01ED OpenMutexA
00002010	00002108	Hint/Name RVA	001B CloseHandle
00002014	00000000	End of Imports	KERNEL32.dll
00002018	0000219C	Hint/Name RVA	009D _adjust_fdiv
0000201C	00002192	Hint/Name RVA	0291 malloc
00002020	00002186	Hint/Name RVA	010F _initterm
00002024	0000217E	Hint/Name RVA	025E free
00002028	00002168	Hint/Name RVA	02C0 strncmp
0000202C	00000000	End of Imports	MSVCRT.dll
00002030	80000017	Ordinal	0017
00002034	80000073	Ordinal	0073
00002038	8000000B	Ordinal	000B
0000203C	80000004	Ordinal	0004
00002040	80000013	Ordinal	0013
00002044	80000016	Ordinal	0016
00002048	80000010	Ordinal	0010
0000204C	80000003	Ordinal	0003
00002050	80000074	Ordinal	0074
00002054	80000009	Ordinal	0009
00002058	00000000	End of Imports	WS2_32.dll

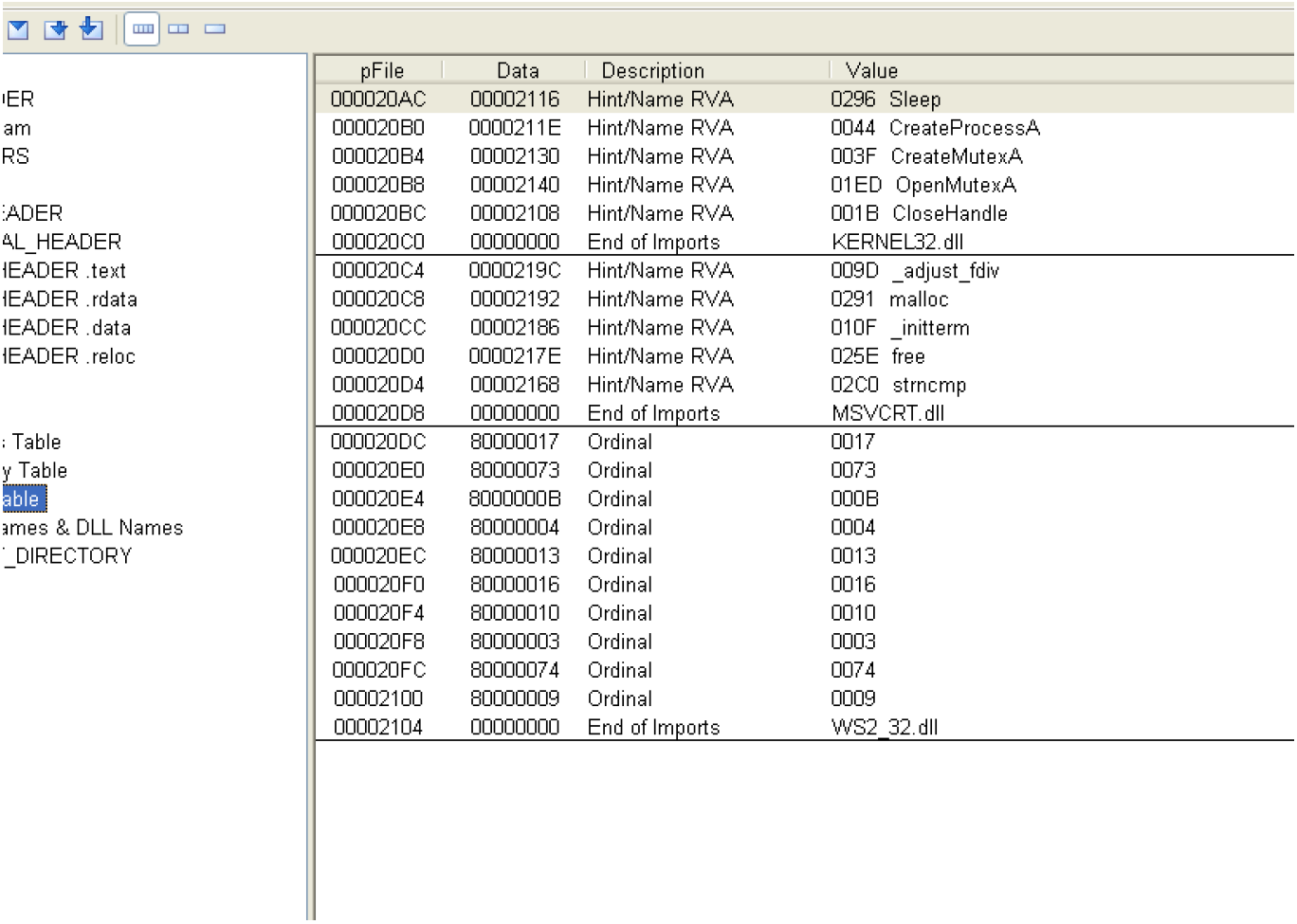
PEiD工具标记为未加壳的代码，并是由Microsoft Visual C++编译的，如下图所示：



这说明，这些文件并没有被加壳。事实上，文件只有少量的导入表说明它们很可能只是一些小程序但是该DLL文件并没有导出表，这是不正常的，但并不是被加壳的迹象。

> Q4. 是否有导入函数显示出了这个恶意代码是做什么的？如果是，是哪些导入函数？

对`.exe`文件的导入表与字符串列表进行查看。从`msvcrt.dll`导入的通常是被每一个可执行文件都包含的，因为它们是作为包装代码被编译器加入可执行文件的，当查看从`kernel32.dll`导入的函数时，可以看到一些打开与操作文件的函数，以及`FindFirstFile`和`FindNextFile`函数。如下图所示：



pFile	Data	Description	Value
000020AC	00002116	Hint/Name RVA	0296 Sleep
000020B0	0000211E	Hint/Name RVA	0044 CreateProcessA
000020B4	00002130	Hint/Name RVA	003F CreateMutexA
000020B8	00002140	Hint/Name RVA	01ED OpenMutexA
000020BC	00002108	Hint/Name RVA	001B CloseHandle
000020C0	00000000	End of Imports	KERNEL32.dll
000020C4	0000219C	Hint/Name RVA	009D _adjust_fdiv
000020C8	00002192	Hint/Name RVA	0291 malloc
000020CC	00002186	Hint/Name RVA	010F _initterm
000020D0	0000217E	Hint/Name RVA	025E free
000020D4	00002168	Hint/Name RVA	02C0 strncmp
000020D8	00000000	End of Imports	MSVCRT.dll
000020DC	80000017	Ordinal	0017
000020E0	80000073	Ordinal	0073
000020E4	8000000B	Ordinal	000B
000020E8	80000004	Ordinal	0004
000020EC	80000013	Ordinal	0013
000020F0	80000016	Ordinal	0016
000020F4	80000010	Ordinal	0010
000020F8	80000003	Ordinal	0003
000020FC	80000074	Ordinal	0074
00002100	80000009	Ordinal	0009
00002104	00000000	End of Imports	WS2_32.dll

这些导入函数表明恶意代码会对文件系统进行搜索以及打开和修改文件。还不能确定恶意代码在搜索什么文件，但.exe字符串说明，恶意代码正在寻找搜索目标系统上的可执行文件。

> Q5. 是否有任何其他文件或基于主机的迹象，让你可以在受感染系统上查找？

可以发现C:\windows\System32\Kernel32.dll和C:Windows\System32\kerne132.dll（字母l和数字1的区别），如下图所示：

```
a                                       Value
00 00 00 00 00 00 00  ...............
64 6C 6C 00 00 00 00  kerne132.dll....
64 6C 6C 00 00 00 00  kernel32.dll....
2A 00 00 2E 2E 00 00  .exe....\*......
00 00 00 43 3A 5C 77  ....C:\*....C:\w
73 74 65 6D 33 32 5C  indows\system32\
64 6C 6C 00 00 00 00  kerne132.dll....
00 00 00 4C 61 62 30  Kernel32....Lab0
00 00 00 43 3A 5C 57  1-01.dll....C:\W
73 74 65 6D 33 32 5C  indows\System32\
64 6C 6C 00 00 00 00  Kernel32.dll....
48 49 53 5F 57 49 4C  WARNING_THIS_WIL
5F 59 4F 55 52 5F 4D  L_DESTROY_YOUR_M
00 00 00 00 00 00 00  ACHINE.........
00 00 00 00 00 00 00  ...............
00 00 00 00 00 00 00  ...............
00 00 00 00 00 00 00  ...............
-- -- -- -- -- -- --
```

`kerne132.dll`文件显然是想将自己冒充混淆为`Windows`的系统文件`kernel32.dll`。因此，`kernel/32.dll`可以作为一个基于主机的迹象来发现恶意代码感染，并且是分析恶意代码所需要关注的一个线索。

> Q6. 是否有基于网络的迹象，可以用来发现受感染机器上的这个恶意代码?

查看`Lab01-01.dll`的导入表和字符串列表。其中导入了`WS2_32.dll`的一些函数。因为这些导入函数是按照序号进行导入的，现在还不知道导入的到底是哪些函数。从`kernel32.dll`导入了两个有趣的函数：`Createprocess`和`Sleep`，如下图所示：

escription	Value
/Name RVA	0296 Sleep
/Name RVA	0044 CreateProcessA
/Name RVA	003F CreateMutexA
/Name RVA	01ED OpenMutexA
/Name RVA	001B CloseHandle
l of Imports	KERNEL32.dll
/Name RVA	009D _adjust_fdiv
/Name RVA	0291 malloc
/Name RVA	010F _initterm
/Name RVA	025F free

这两个函数普遍在后门程序中使用，这些函数在与`exec`与`sleep`字符串结合使用时，需要特别的关注。`.dll`文件中包含一个私有子网IP地址`127.26.152.13`的字符串。这个地址预示着这个程序是为了教学目的而不是恶意目的而编制的，如果这是真正的恶意代码，这个地址应该是可路由的公网地址，它会是一个很好的基于网络的恶意代码感染迹象，可以用来识别这个恶意代码。

> Q7. 你猜这些文件的目的是什么?

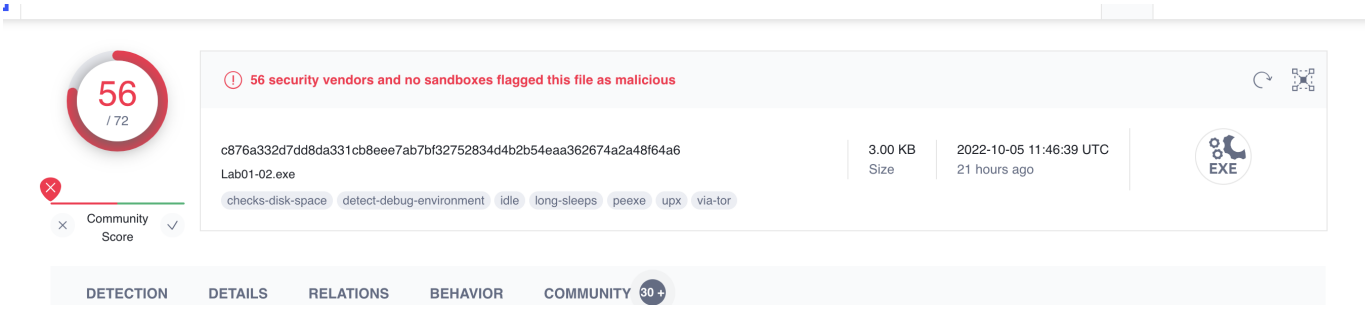
`exec`字符串可能是通过网络来给后门程序传送命令，让它通过`createprocess`函数运行一个程序的。`sleep`字符串可能用于命令后门程序进入休眠模式。`.dll`文件可能是一个后门。而`.exe`文件是用来安装与运行`DLL`文件的。

lab1-2

分析`Lab01-02.exe`文件.

> Q1.将`Lab01-02.exe`文件上传至`VirusTotal`进行分析并查看报告。文件匹配到了已有的反病毒软件特征吗?

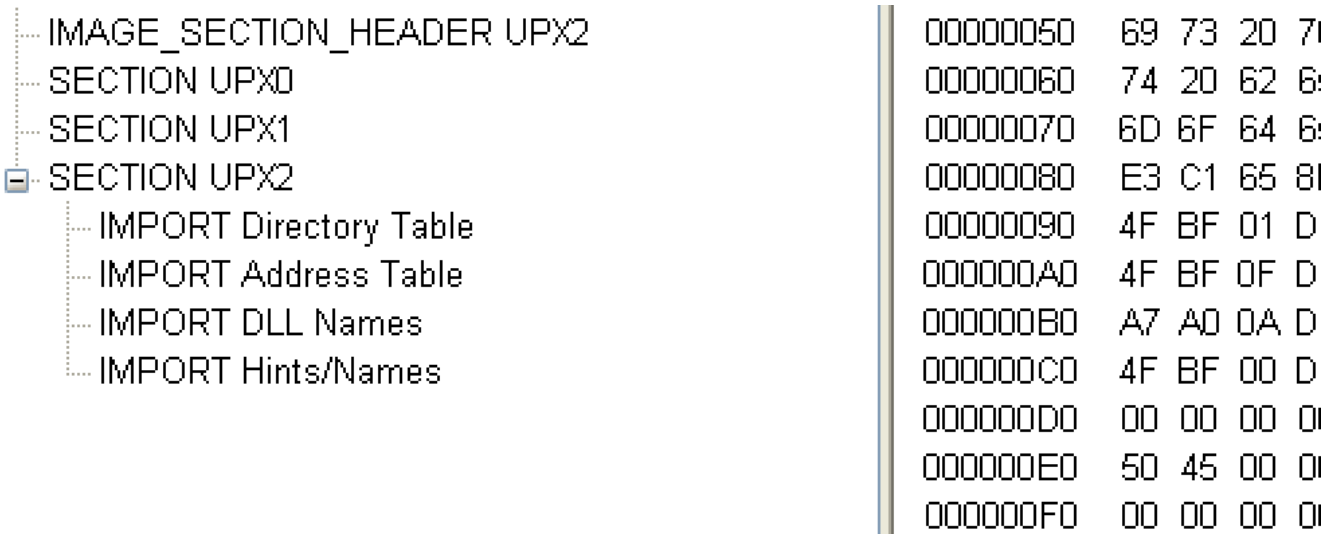
上传可执行文件后的结果如下图所示:



可以看到有56个安全供应商将其标记为恶意程序

> Q2. 是否有这个文件被加壳或混淆的任何迹象？如果是这样，这些迹象是什么？如果该文件被加壳，请进行脱壳，如果可能的话。

使用PEview工具打开这个文件，几个迹象可以表明这个文件是被加壳的。其中最明显的迹象是名为UPX0、UPX1和UPX2的节，明显是由UPX进行加壳后恶意代码程序的节名称，如下图所示：



所以使用PEiD工具，来确认文件的加壳类型，在确定出加壳类型之后，从upx下载地址下载UPX工具，并运行以下命令对其进行脱壳：

```
upx -o newFilename -d originalFilename
```

-d选项表示对文件进行脱壳，-o选项指定了输出文件名。

> Q3. 有没有任何导入函数能够暗示出这个程序的功能？如果是，是哪些导入函数，它们会告诉你什么？

脱壳之后，审查脱壳恶意代码的导入表和字符串列表，如下图所示：

pFile	Data	Description	Value
00000A00	00000000	Import Name Table RVA	
00000A04	00000000	Time Date Stamp	
00000A08	00000000	Forwarder Chain	
00000A0C	00006098	Name RVA	KERNEL32.DLL
00000A10	00006064	Import Address Table RVA	
00000A14	00000000	Import Name Table RVA	
00000A18	00000000	Time Date Stamp	
00000A1C	00000000	Forwarder Chain	
00000A20	000060A5	Name RVA	ADVAPI32.dll
00000A24	00006080	Import Address Table RVA	
00000A28	00000000	Import Name Table RVA	
00000A2C	00000000	Time Date Stamp	
00000A30	00000000	Forwarder Chain	
00000A34	000060B2	Name RVA	MSVCRT.dll
00000A38	00006088	Import Address Table RVA	
00000A3C	00000000	Import Name Table RVA	
00000A40	00000000	Time Date Stamp	
00000A44	00000000	Forwarder Chain	
00000A48	000060BD	Name RVA	WININET.dll
00000A4C	00006090	Import Address Table RVA	
00000A50	00000000		
00000A54	00000000		
00000A58	00000000		
00000A5C	00000000		

这个恶意代码的导入函数都是从`kernel32.dll`和`msvert.dll`，而这些函数几乎被每个程序都导入，所以它们能够显示关于这个恶意代码的信息很少。从`wininet.dll`导入的函数显示这个恶意代码会进行联网操作(`Internetopen`和`InternetOpenURL`），从`advapi32.dll`的导入函数(`Createservice`）显示这个代码会创建一个服务。

> Q4. 哪些基于主机或基于网络的迹象可以被用来确定被这个恶意代码所感染的机器?

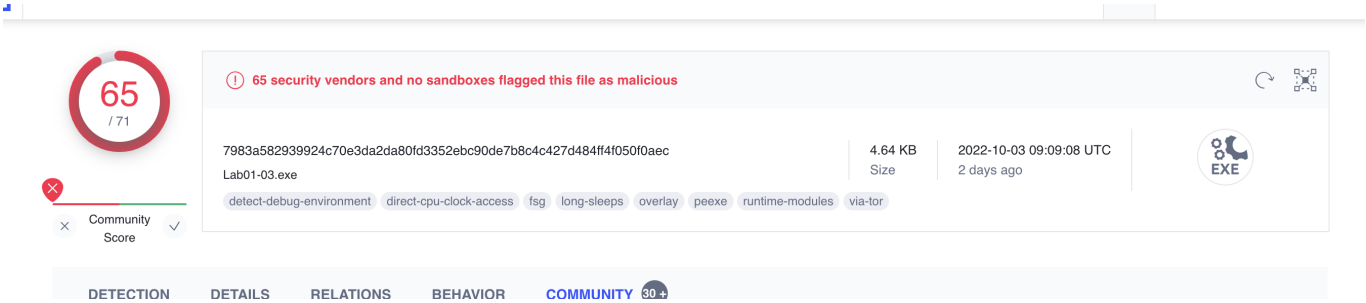
在检查字符串列表时，我们看到`www.malwareanalysisbook.com`，这可能是`InterneOpenURL`函数中所打开的`URL`；以及`Malservice字符串，这可能是所创建的服务名称。虽然不能肯定这个程序会做什么，但这些线索，能够在网络上和系统里来搜索这个恶意代码。

lab1-3

分析`Lab01-03.exe`文件.

> Q1. 将`Lab01-03.exe`文件上传至`VirusTotal`进行分析并查看报告。文件匹配到了已有的反病毒软件特征吗?

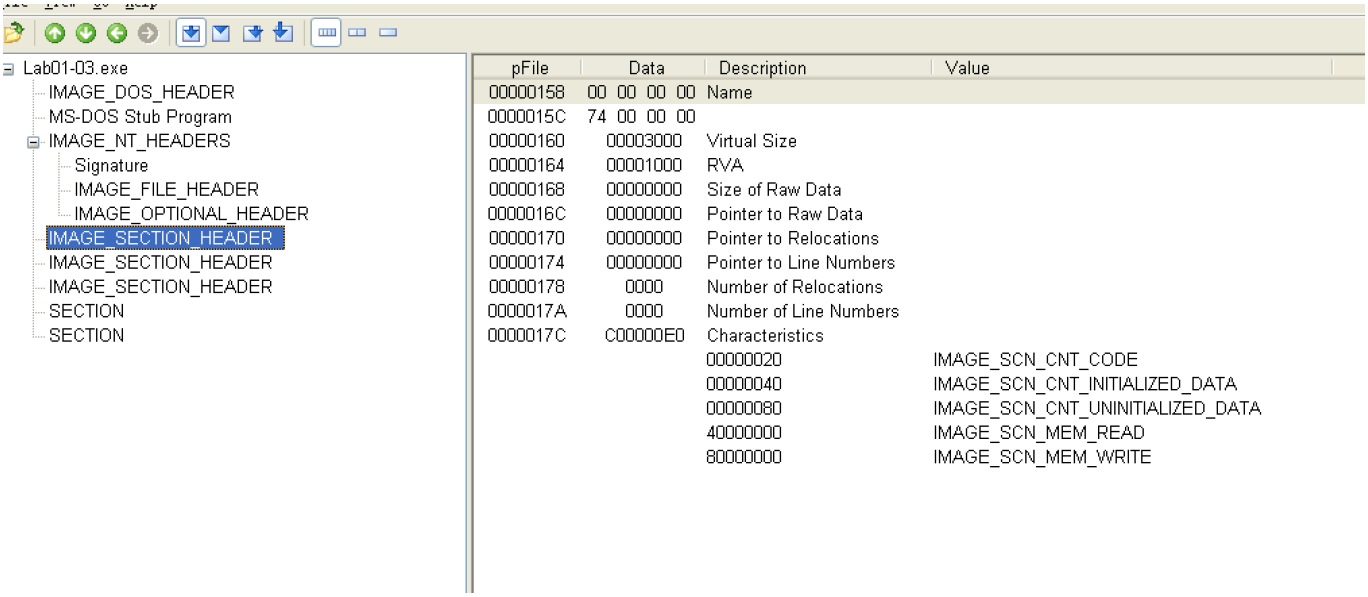
上传可执行文件后的结果如下图所示:



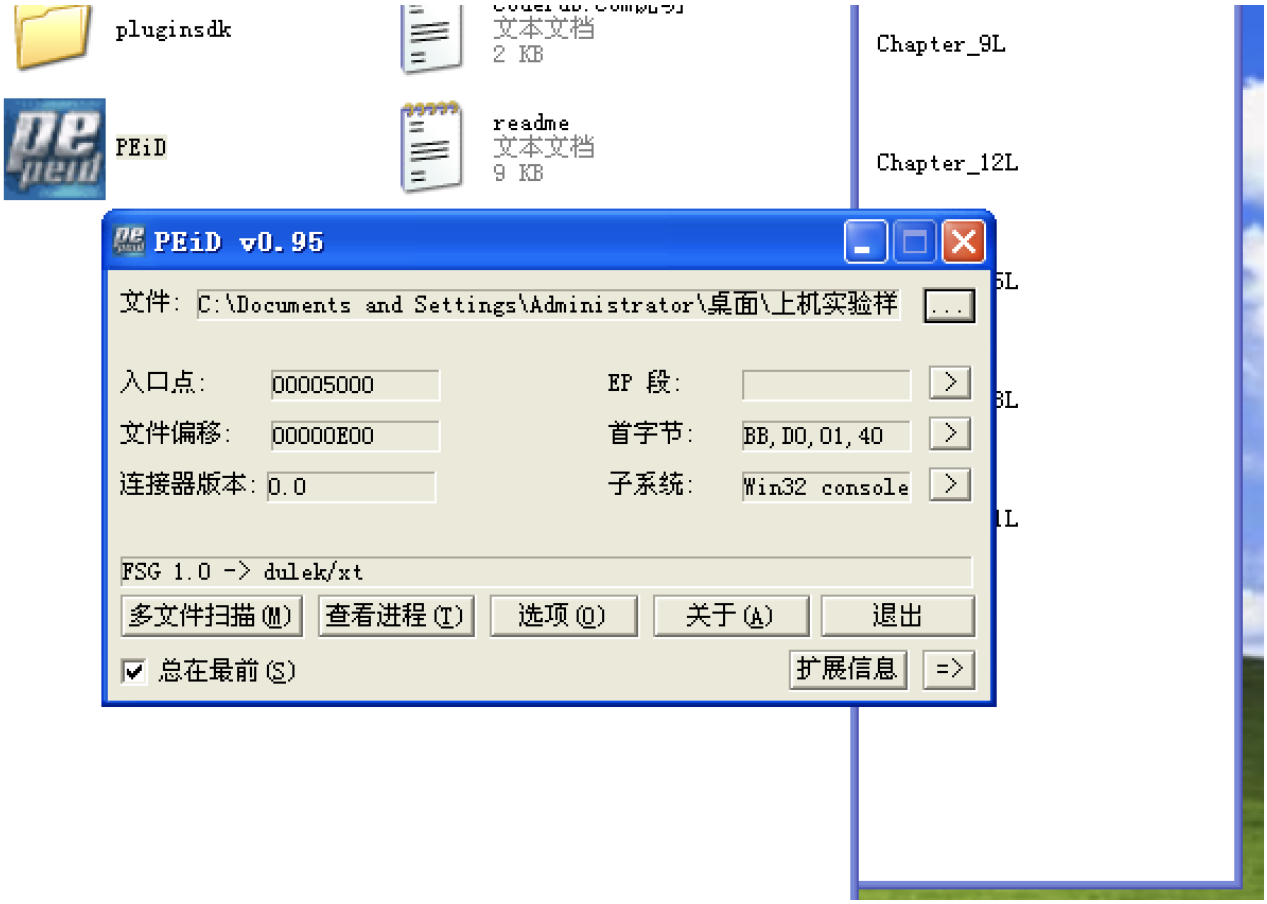
可以看到有65个安全供应商将其标记为恶意程序

> Q2. 是否有这个文件被加壳或混淆的任何迹象？如果是这样，这些迹象是什么？如果该文件被加壳，请进行脱壳，如果可能的话。

这个文件是加壳的，但以我现在的技术还没有办法对其进行脱壳操作，当使用`PEview`工具打开文件时，可以观察到一些迹象表明这个文件是加壳的，如下图所示：



首先是文件的节并没有名字，首节的虚拟大小为`0x3000`，但原始数据大小却为0。接下来运行`PEiD`工具进行确认，它将加壳器标识为`FSG 1.0– > dulek /xt`，如下图所示：



为了确认这个文件是被加壳的，搜索导入表却没有发现。一个没有导入表的可执行文件是极为罕见的，没有导入表这个奇怪现象说明应该尝试另一种工具，因为PEview无法正常处理这个文件。

然后使用Dependency Walker打开这个文件，看到它拥有一个导入表，但其中只有两个函数：LoadLibrary和GetProcAddress。加壳文件往往只有这两个导入函数，这进一步表明了这个文件是被加过壳的。

> Q3. 有没有任何导入函数能够暗示出这个程序的功能？如果是，是哪些导入函数，它们会告诉你什么？

由于Q2中的分析，我现在还不能对其进行脱壳处理，所以现在还不能看到它的具体函数。

> Q4. 有哪些基于主机或基于网络的迹象，可以被用来确定被这个恶意代码所感染的机器？

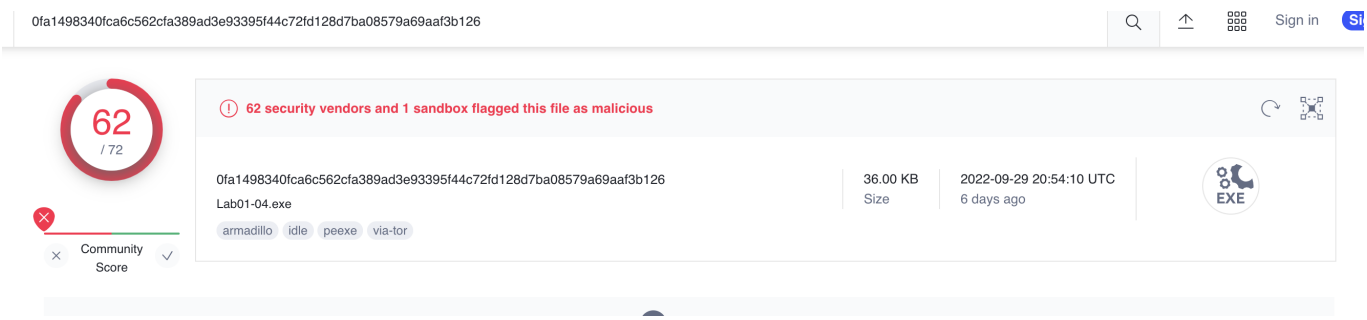
由于Q2中的分析，我现在还不能对其进行脱壳处理，所以现在还不能确定基于主机或基于网络的迹象。

lab1-4

分析Lab01-04.exe文件.

> Q1. 将Lab01-04.exe文件上传至VirusTotal进行分析并查看报告。文件匹配到了已有的反病毒软件特征吗？

上传可执行文件后的结果如下图所示：



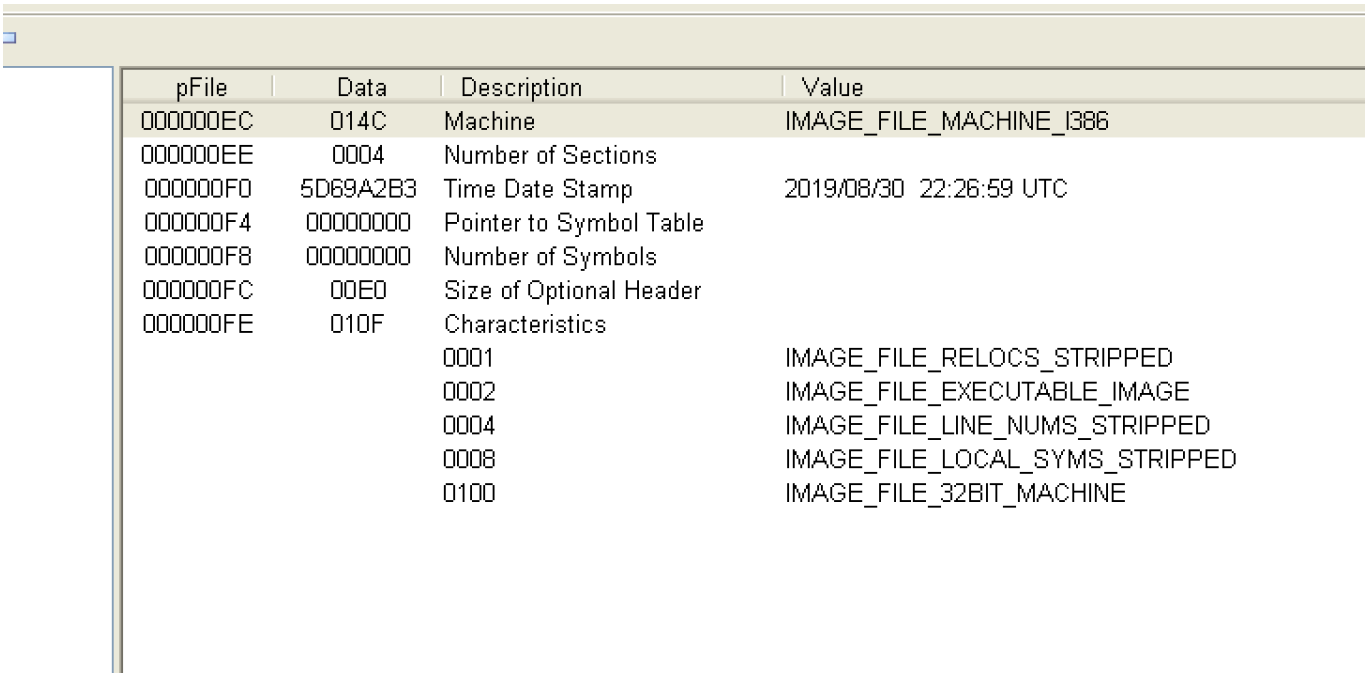
可以看到有62个安全供应商和1个沙盒将其标记为恶意程序，对于`Lab01-04.exe`文件，从`VirusTotal.com`得到的结果表明这个程序是与下载器相关的。

> Q2. 是否有这个文件被加壳或混淆的任何迹象？如果是这样，这些迹象是什么？如果该文件被加壳，请进行脱壳，如果可能的话。

`PEview`没有显示迹象表明这个文件是加壳或是混淆的。

> Q3. 这个文件是什么时候被编译的?

根据文件头，这个文件是在2019年8月编译的，如下图所示：

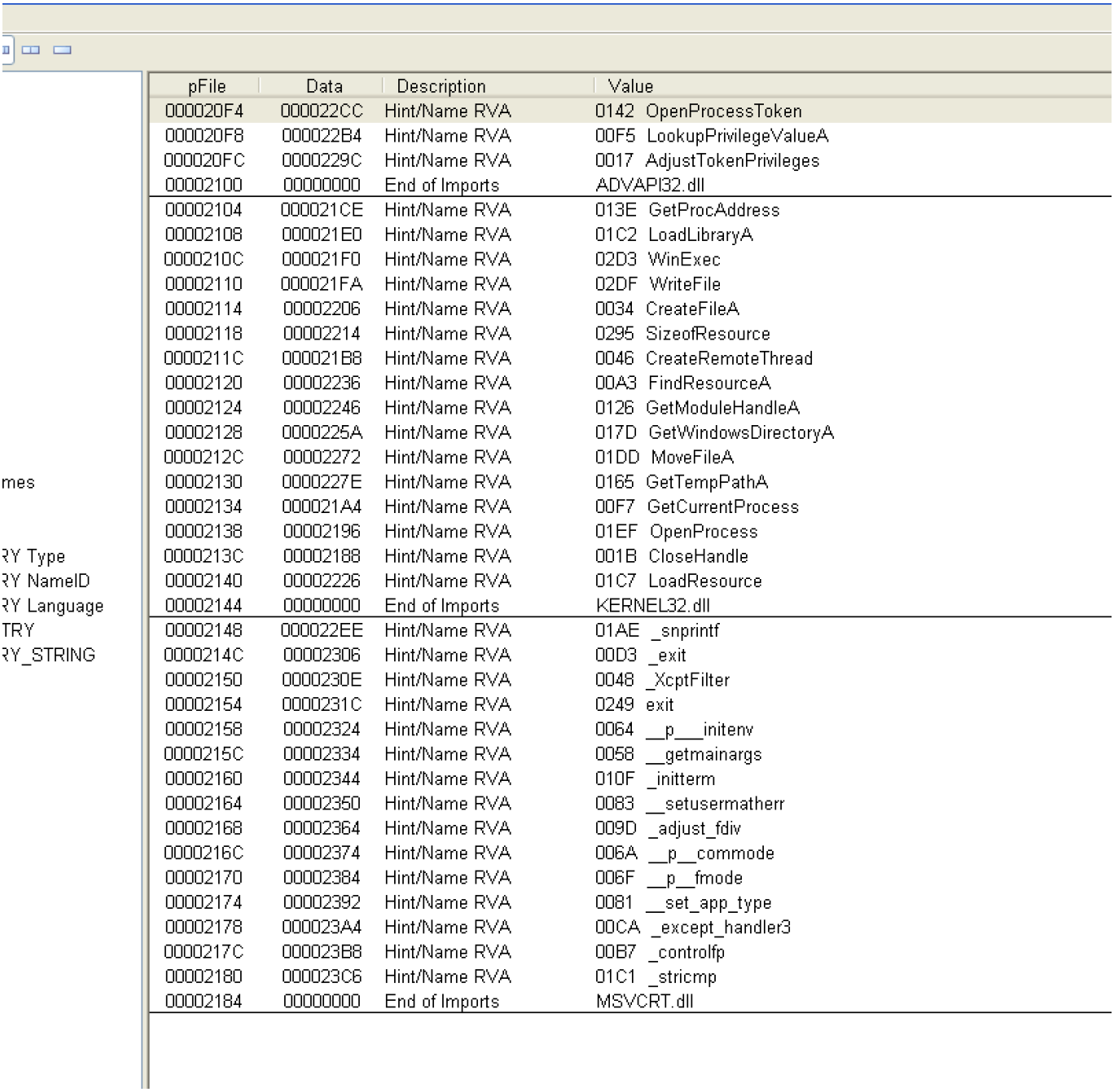


pFile	Data	Description	Value
000000EC	014C	Machine	IMAGE_FILE_MACHINE_I386
000000EE	0004	Number of Sections	
000000F0	5D69A2B3	Time Date Stamp	2019/08/30 22:26:59 UTC
000000F4	00000000	Pointer to Symbol Table	
000000F8	00000000	Number of Symbols	
000000FC	00E0	Size of Optional Header	
000000FE	010F	Characteristics	
		0001	IMAGE_FILE_RELOCS_STRIPPED
		0002	IMAGE_FILE_EXECUTABLE_IMAGE
		0004	IMAGE_FILE_LINE_NUMS_STRIPPED
		0008	IMAGE_FILE_LOCAL_SYMS_STRIPPED
		0100	IMAGE_FILE_32BIT_MACHINE

显然，通过了解到这个书的发布时间，这个编译时间是伪造的，所以还不能确定这个文件到底是什么时候被编译的。

> Q4. 有没有任何导入函数能够暗示出这个程序的功能？如果是，是哪些导入函数，它们会告诉你什么？

导入函数表如下图所示：

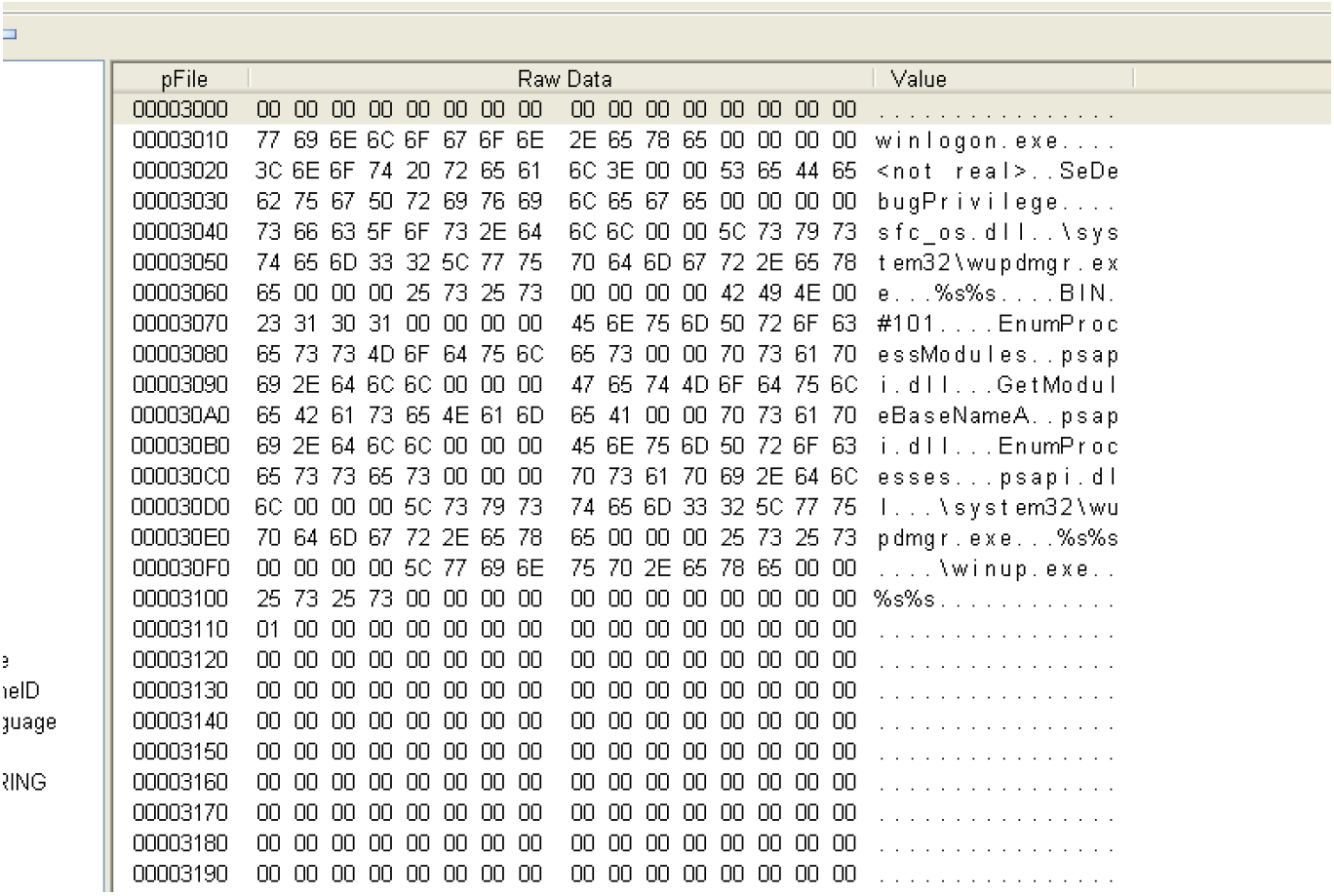


pFile	Data	Description	Value
000020F4	000022CC	Hint/Name RVA	0142 OpenProcessToken
000020F8	000022B4	Hint/Name RVA	00F5 LookupPrivilegeValueA
000020FC	0000229C	Hint/Name RVA	0017 AdjustTokenPrivileges
00002100	00000000	End of Imports	ADVAPI32.dll
00002104	000021CE	Hint/Name RVA	013E GetProcAddress
00002108	000021E0	Hint/Name RVA	01C2 LoadLibraryA
0000210C	000021F0	Hint/Name RVA	02D3 WinExec
00002110	000021FA	Hint/Name RVA	02DF WriteFile
00002114	00002206	Hint/Name RVA	0034 CreateFileA
00002118	00002214	Hint/Name RVA	0295 SizeofResource
0000211C	000021B8	Hint/Name RVA	0046 CreateRemoteThread
00002120	00002236	Hint/Name RVA	00A3 FindResourceA
00002124	00002246	Hint/Name RVA	0126 GetModuleHandleA
00002128	0000225A	Hint/Name RVA	017D GetWindowsDirectoryA
0000212C	00002272	Hint/Name RVA	01DD MoveFileA
00002130	0000227E	Hint/Name RVA	0165 GetTempPathA
00002134	000021A4	Hint/Name RVA	00F7 GetCurrentProcess
00002138	00002196	Hint/Name RVA	01EF OpenProcess
0000213C	00002188	Hint/Name RVA	001B CloseHandle
00002140	00002226	Hint/Name RVA	01C7 LoadResource
00002144	00000000	End of Imports	KERNEL32.dll
00002148	000022EE	Hint/Name RVA	01AE _snprintf
0000214C	00002306	Hint/Name RVA	00D3 _exit
00002150	0000230E	Hint/Name RVA	0048 _XcptFilter
00002154	0000231C	Hint/Name RVA	0249 exit
00002158	00002324	Hint/Name RVA	0064 __p___initenv
0000215C	00002334	Hint/Name RVA	0058 __getmainargs
00002160	00002344	Hint/Name RVA	010F _initterm
00002164	00002350	Hint/Name RVA	0083 __setusermatherr
00002168	00002364	Hint/Name RVA	009D _adjust_fdiv
0000216C	00002374	Hint/Name RVA	006A __p__commode
00002170	00002384	Hint/Name RVA	006F __p__fmode
00002174	00002392	Hint/Name RVA	0081 __set_app_type
00002178	000023A4	Hint/Name RVA	00CA _except_handler3
0000217C	000023B8	Hint/Name RVA	00B7 _controlfp
00002180	000023C6	Hint/Name RVA	01C1 _stricmp
00002184	00000000	End of Imports	MSVCRT.dll

从advapi32.dll导入的函数可以看到，程序做了一些与权限有关的事情。可以假设它试图访问使用了特殊权限进行保护的文件。从kernel32.dll的导入函数可以看出这个程序从资源节中装载数据(LoadResource、FindResource和SizeofResource），并写一个文件到磁盘上(CreateFile和writeFile），接着执行一个磁盘上的文件(WinExec）。可以猜测这个程序将文件写入到了系统目录，因为它调用了GetwindowsDirectory函数。

> Q5.有哪些基于主机或基于网络的迹象，可以被用来确定被这个恶意代码所感染的机器?

通过检查字符串，如下图所示：



pFile	Raw Data	Value
00003000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00003010	77 69 6E 6C 6F 67 6F 6E 2E 65 78 65 00 00 00 00	winlogon.exe....
00003020	3C 6E 6F 74 20 72 65 61 6C 3E 00 00 53 65 44 65	<not real>..SeDe
00003030	62 75 67 50 72 69 76 69 6C 65 67 65 00 00 00 00	bugPrivilege....
00003040	73 66 63 5F 6F 73 2E 64 6C 6C 00 00 5C 73 79 73	sfc_os.dll..\sys
00003050	74 65 6D 33 32 5C 77 75 70 64 6D 67 72 2E 65 78	tem32\wupdmgr.ex
00003060	65 00 00 00 25 73 25 73 00 00 00 00 42 49 4E 00	e...%s%s....BIN.
00003070	23 31 30 31 00 00 00 00 45 6E 75 6D 50 72 6F 63	#101....EnumProc
00003080	65 73 73 4D 6F 64 75 6C 65 73 00 00 70 73 61 70	essModules..psap
00003090	69 2E 64 6C 6C 00 00 00 47 65 74 4D 6F 64 75 6C	i.dll...GetModul
000030A0	65 42 61 73 65 4E 61 6D 65 41 00 00 70 73 61 70	eBaseNameA..psap
000030B0	69 2E 64 6C 6C 00 00 00 45 6E 75 6D 50 72 6F 63	i.dll...EnumProc
000030C0	65 73 73 65 73 00 00 00 70 73 61 70 69 2E 64 6C	esses...psapi.dl
000030D0	6C 00 00 00 5C 73 79 73 74 65 6D 33 32 5C 77 75	l...\system32\wu
000030E0	70 64 6D 67 72 2E 65 78 65 00 00 00 25 73 25 73	pdmgr.exe...%s%s
000030F0	00 00 00 00 5C 77 69 6E 75 70 2E 65 78 65 00 00	....\winup.exe..
00003100	25 73 25 73 00 00 00 00 00 00 00 00 00 00 00 00	%s%s............
00003110	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00003120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00003130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00003140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00003150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00003160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00003170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00003180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00003190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....

可以发现www.malwareanalysisbok.com/updater.exe字符串，这可能是保存下载恶意代码的网络位置。还可以发现字符串\system32\wupdmgr.exe，结合GetwindowsDirectory函数调用，这表明恶意代码在C:\Windows\System32\wupdmgr.exe位置创建或者修改了一个文件。现在已经有比较充分的证据，知道这个恶意文件下载了一个新的恶意代码。已经知道它是从哪个位置下载恶意代码的，并且可以猜到它会将下载到的恶意代码存储在什么位置。

> Q6.这个文件在资源段中包含一个资源。使用Resource Hacker工具来检查资源，然后抽取资源。从资源中你能发现什么吗?

这个恶意代码样本最有趣的就是它的资源节，当使用Resource Hacker工具打开这个恶意代码时，只看到了一个资源。Resource Hacker工具识别这个资源的类型是二进制，说明是任意的二进制数据，但当查看数据时，它的绝大部分都是无意义的，唯一例外的是字符串!This program cannot be run in Dos mode，这个字符串是在所有PE文件开始处的DOS头部中都会包含的错误消息。于是，可以得出结论，认为这一资源其实是在Lab01-04.exe资源节中存储的另一个可执行文件。这种技术在恶意代码中使用得非常普遍。

为了在Resource Hacker工具中继续分析这一文件，单击Action--Save resource as binary file。在存储完资源后，使用PEview来打开文件，分析内嵌的文件。通过查看导入表，可以发现嵌入文件在访问一些网络函数，它调用了URLDownloadToFile,一个由恶意下载器普遍使用的函数，它调用了WinExec函数，可能执行了下载到的文件。

实验心得

这一次的实验是恶意代码与防治分析的第一次实验，由于本身参与CTF竞赛，所以对这个领域有一定的了解，通过课堂上的理论与实验课的动手相结合，对于恶意代码的检查以及比如PEview、Resource

Hacker、PEiD等工具也更加的熟悉，也了解到VirusTotal这个网站的强大之处。 通过本次实验，认识到自己作为一名信息安全专业学生的责任，更加期待本学期后续的实验，最后也希望自己能有更好的发展。