



恶意代码分析与防治技术实验报告

实验一

网络空间安全学院

信息安全专业

2211985 李佳璐

一、实验目的

1. 使用VirusTotal、MicroSoft Defender AntiVirus等恶意代码分析工具分析样本，对恶意代码有初步认识，初步掌握对恶意代码特征的分析过程。
2. 学习编写Yara检测规则，并使用样本测试规则。

二、实验原理

实验环境：Windows11（本机）、WindowsXP（VMware虚拟机）、Windows10（VMware虚拟机）

实验工具：VirusTotal, PEvent, PEiD, DependencyWalker, strings, Yara, MicroSoft Defender AntiVirus

三、实验过程

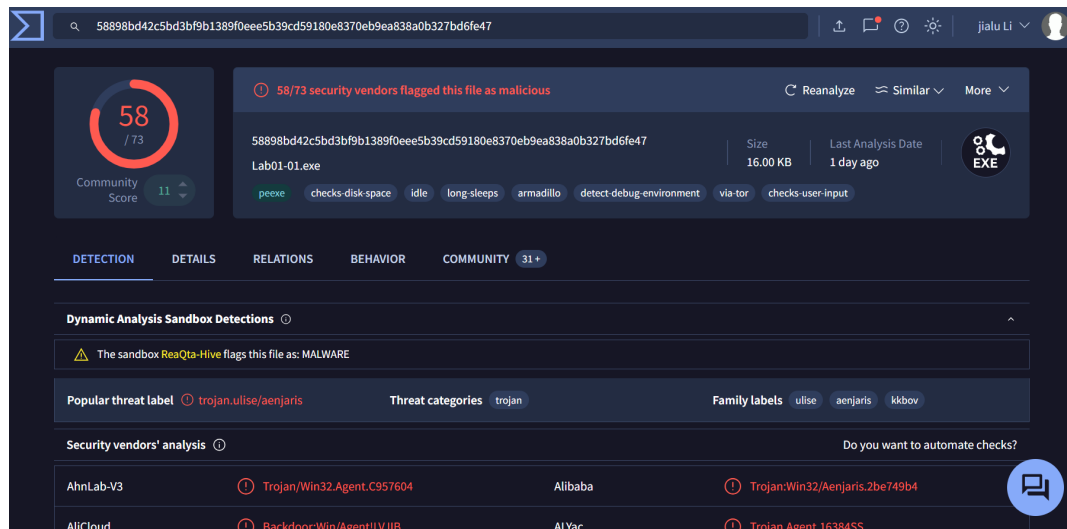
• Lab1-1

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。

1. 将文件上传至 <http://www.VirusTotal.com/> 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？

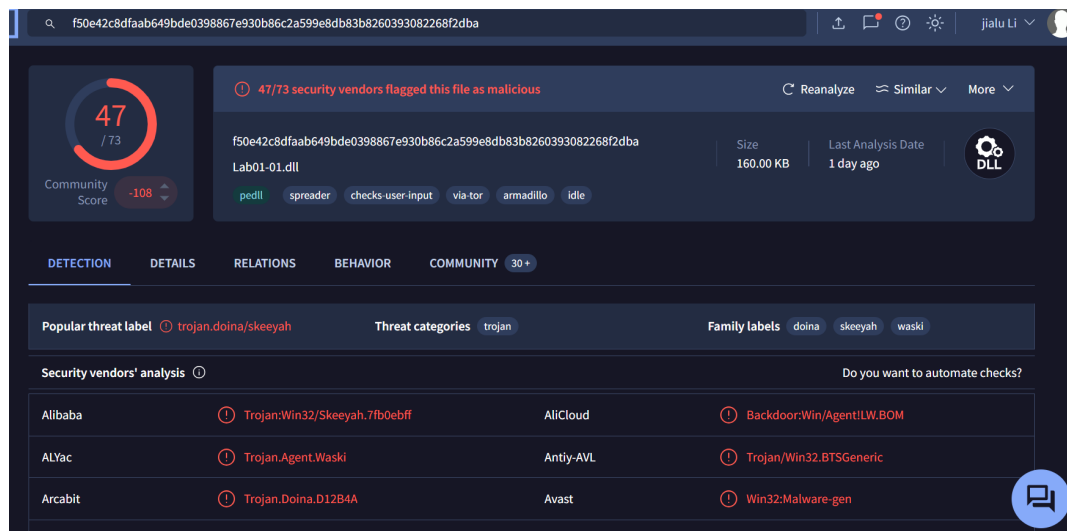
将Lab01-01.exe文件上传后，可见如下检测结果（58/72）

共58个反病毒供应商成功识别并匹配到已知的病毒特征，进而检测出文件中的潜在威胁。



将Lab01-01.dll文件上传后，可见如下检测结果（47/72）

共47个反病毒供应商成功识别并匹配到已知的病毒特征，进而检测出文件中的潜在威胁。



2. 这些文件是什么时候编译的？

使用PEview工具来打开文件。对于每个文件通过浏览IMAGE_NT_HEADERS查看时间戳对应的十六进制数，将其转换为十进制后利用UTC在线时间戳转换网站得到其对应时间。

PEView - D:\deskbook\恶意代码\上机实验样本\Chapter_1\Lab01-01.exe

	pFile	Raw Data	Value
Lab01-01.exe			
IMAGE_DOS_HEADER	000000E8	50 45 00 00 4C 01 03 00 D3 2F 0E 4D 00 00 00 00	PE...L.../.M...
MS-DOS Stub Program	000000F8	00 00 00 00 E0 00 0F 01 0B 01 06 00 00 10 00 00I.....
IMAGE_NT_HEADERS	00000108	00 20 00 00 00 00 00 00 20 18 00 00 00 10 00 00>.....
IMAGE_SECTION_HEADER .text	00000118	00 20 00 00 00 00 40 00 00 10 00 00 00 10 00 00@.....
IMAGE_SECTION_HEADER .rdata	00000128	04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00@.....
IMAGE_SECTION_HEADER .data	00000138	00 40 00 00 00 10 00 00 00 00 00 00 03 00 00 00@.....
SECTION .text	00000148	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00>.....
SECTION .rdata	00000158	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00>.....
SECTION .data	00000168	7C 20 00 00 3C 00 00 00 00 00 00 00 00 00 00 00<.....
	00000178	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....
	00000188	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....
	00000198	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....
	000001A8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....
	000001B8	00 00 00 00 00 00 00 00 00 20 00 00 6C 00 00 00!.....
	000001C8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....
	000001D8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....

时间戳的十六进制数4D0E2FD3转换为十进制1,292,775,379

再利用UTC在线时间戳转换器转换为时间

时间戳	1292775379	秒(s) ▼	转换 >	2010-12-20 00:16:19
-----	------------	--------	------	---------------------

另一个dll文件同理可得

PEView - D:\deskbook\恶意代码\BinaryCollection\Chapter_1\Lab01-01.dll

	pFile	Raw Data	Value
Lab01-01.dll			
IMAGE_DOS_HEADER	000000E0	50 45 00 00 4C 01 04 00 E6 2F 0E 4D 00 00 00 00	PE...L.../.M...
MS-DOS Stub Program	000000F0	00 00 00 00 E0 00 0E 21 0B 01 06 00 00 10 00 00I.....
IMAGE_NT_HEADERS	00000100	00 60 02 00 00 00 00 00 FA 12 00 00 00 10 00 00>.....
Signature	00000110	00 20 00 00 00 00 00 10 00 10 00 00 00 10 00 00>.....
IMAGE_FILE_HEADER	00000120	04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00>.....
IMAGE_OPTIONAL_HEADER	00000130	00 80 02 00 00 10 00 00 00 00 00 00 02 00 00 00>.....
IMAGE_SECTION_HEADER .text	00000140	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00>.....
IMAGE_SECTION_HEADER .rdata	00000150	00 00 00 00 10 00 00 00 B0 21 00 00 16 3E 02 00!...>.....
IMAGE_SECTION_HEADER .data	00000160	5C 20 00 00 50 00 00 00 00 00 00 00 00 00 00 00\...P.....
IMAGE_SECTION_HEADER .reloc	00000170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....
SECTION .text	00000180	00 70 02 00 64 00 00 00 00 00 00 00 00 00 00 00p...d.....
SECTION .rdata	00000190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....
SECTION .data	000001A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....
SECTION .reloc	000001B0	00 00 00 00 00 00 00 00 00 20 00 00 5C 00 00 00\.....
	000001C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....
	000001D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>.....

时间戳	1292775398	秒(s) ▼	转换 >	2010-12-20 00:16:38
-----	------------	--------	------	---------------------

这两个文件都是在**2010年12月20日**被编译的，两者编译时间差仅为19秒，可证实这两个文件同属一个恶意代码包。

3. 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，也有适当大小的良好组织的文件节。PEiD工具也都标记为未加壳的代码，且是由Microsoft Visual C++编译的，可见这两个文件没有被加壳。

PEiD v0.95

File: ...

Entrypoint: EP Section: >

File Offset: First Bytes: >

Linker Info: Subsystem: >

Microsoft Visual C++ 6.0 DLL

☒ Stay on top

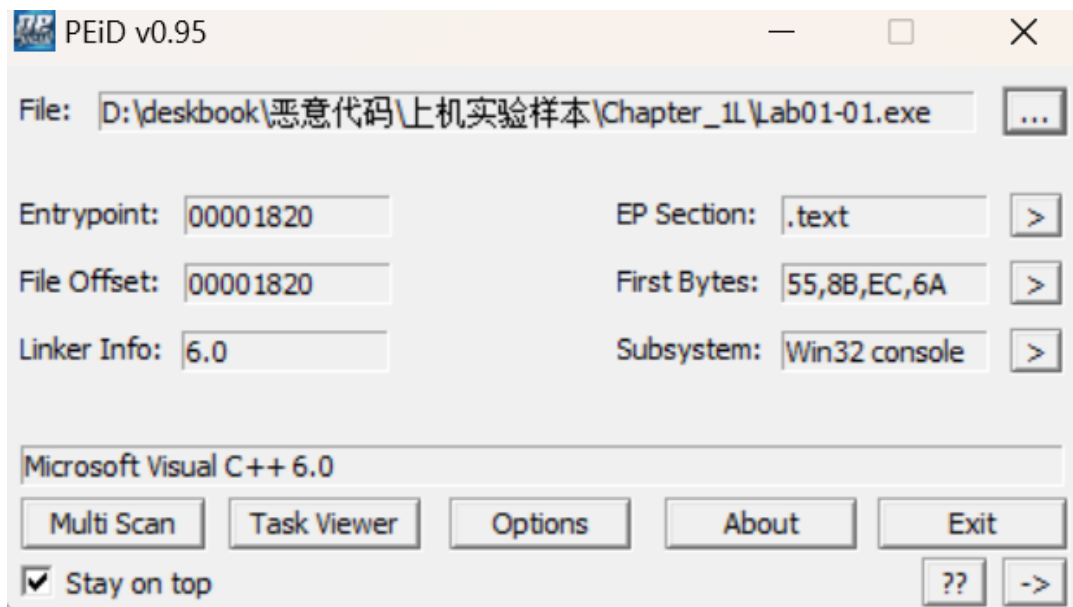
PE Details

Basic Information

EntryPoint:	<input type="text" value="000012FA"/>	SubSystem:	<input type="text" value="0002"/>
ImageBase:	<input type="text" value="10000000"/>	NumberOfSections:	<input type="text" value="0004"/>
SizeOfImage:	<input type="text" value="00028000"/>	TimeDateStamp:	<input type="text" value="4D0E2FE6"/>
BaseOfCode:	<input type="text" value="00001000"/>	SizeOfHeaders:	<input type="text" value="00001000"/>
BaseOfData:	<input type="text" value="00002000"/>	Characteristics:	<input type="text" value="210E"/>
SectionAlignment:	<input type="text" value="00001000"/>	Checksum:	<input type="text" value="00000000"/>
FileAlignment:	<input type="text" value="00001000"/>	SizeOfOptionalHeader:	<input type="text" value="00E0"/>
Magic:	<input type="text" value="010B"/>	NumOfRvaAndSizes:	<input type="text" value="00000010"/>

Directory Information

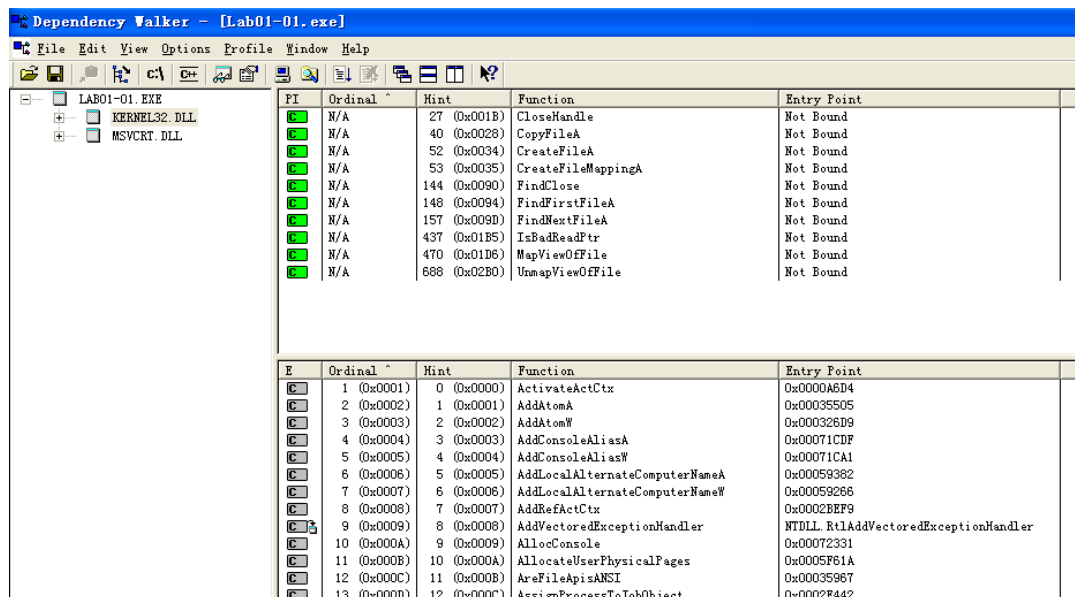
	RVA	SIZE	
ExportTable:	<input type="text" value="000021B0"/>	<input type="text" value="00023E16"/>	<input type="button" value="..."/> <input type="button" value=">"/>
ImportTable:	<input type="text" value="0000205C"/>	<input type="text" value="00000050"/>	<input type="button" value="..."/> <input type="button" value=">"/>
Resource:	<input type="text" value="00000000"/>	<input type="text" value="00000000"/>	
TLSTable:	<input type="text" value="00000000"/>	<input type="text" value="00000000"/>	
Debug:	<input type="text" value="00000000"/>	<input type="text" value="00000000"/>	



4. 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?

在VMware虚拟机Windows XP环境下使用Dependency Walker工具查看文件导入表。

- Lab01-01.exe



打开Lab01-01.exe, 可见KERNEL32.DLL和MSVCRT.DLL两个动态链接库。

从MSVCRT.DLL导入的函数通常是被每一个可执行文件都包含的, 因为他们是作为包装代码被编译器加入可执行文件的。

查看KERNEL32.DLL导入的函数, 可见一些打开与操作文件的函数, 以及FindFirstFile和FindNextFile函数。可见该恶意代码会对文件系统进行搜索, 以及打开和修改文件。

- Lab01-01.dll

Dependency Walker - [Lab01-01.dll]

File Edit View Options Profile Window Help

Module	PI	Ordinal	Hint	Function	Entry Point
LAB01-01.DLL					
KERNEL32.DLL		N/A	11 (0x000B)	AreFileApisANSI	0x7C835967
WS2_32.DLL		N/A	39 (0x0027)	CancelIo	0x7C8300CA
ADVAPI32.DLL		N/A	49 (0x0031)	CloseHandle	0x7C809B07
KERNEL32.DLL		N/A	54 (0x0036)	CompareFileTime	0x7C810B69
NTDLL.DLL		N/A	66 (0x0042)	CopyFileW	0x7C82F863
RPCRT4.DLL		N/A	75 (0x004B)	CreateEventA	0x7C83089D
WINTRUST.DLL		N/A	76 (0x004C)	CreateEventW	0x7C80A739
SECUR32.DLL		N/A	79 (0x004F)	CreateFileA	0x7C801A28
ADVAPI32.DLL		N/A	80 (0x0050)	CreateFileMappingA	0x7C8094EE
KERNEL32.DLL		N/A	81 (0x0051)	CreateFileMappingW	0x7C809420
NTDLL.DLL		N/A	82 (0x0052)	CreateFileW	0x7C8107F0
NETAPI32.DLL					
ADVAPI32.DLL		832 (0x0340)	831 (0x033F)	ShowConsoleCursor	0x00072F84
KERNEL32.DLL		833 (0x0341)	832 (0x0340)	SignalObjectAndWait	0x000366AE
MSVCRT.DLL		834 (0x0342)	833 (0x0341)	SizeofResource	0x0000BCF9
NTDLL.DLL		835 (0x0343)	834 (0x0342)	Sleep	0x00002446
RPCRT4.DLL		836 (0x0344)	835 (0x0343)	SleepEx	0x000023A0
SECUR32.DLL		837 (0x0345)	836 (0x0344)	SuspendThread	0x00003974A
NTDSAPI.DLL		838 (0x0346)	837 (0x0345)	SwitchToFiber	0x00010702
NETRAP.DLL		839 (0x0347)	838 (0x0346)	SwitchToThread	0x0000329AA
		840 (0x0348)	839 (0x0347)	SystemTimeToFileTime	0x00010BAC
		841 (0x0349)	840 (0x0348)	SystemTimeToTzSpecificLocalTime	0x00002E991

其中导入了WS2_32.dll中的函数，这些函数提供了联网功能

以及在kernel32.dll中导入了CreateProcess和Sleep这两个函数，而这两个函数普遍在后门程序中使用。

5. 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？

使用Strings工具查看文件中的字符串，结合上题中发现的一些导入函数进行分析。

- Lab01-01.exe

```
D:\>deskbok\strings.exe -a \deskbok\Malware\example1\Chapter_1L\Lab01-01.exe

Strings v2.51
Copyright (C) 1999-2013 Mark Russinovich
Sysinternals - www.sysinternals.com
```

```
_adjust_fdiv
__p__commode
__p__fmode
__set_app_type
_except_handler3
_controlfp
stricmp
kerne132.dll
kerne132.dll
.exe
C:\*
C:\windows\system32\kerne132.dll
Kernel32.
Lab01-01.dll
C:\Windows\System32\Kernel32.dll
WARNING_THIS_WILL_DESTROY_YOUR_MACHINE
```

根据上题中的一些导入函数可知恶意代码会对文件系统进行搜索，但.exe字符串说明，恶意代码正在搜索目标系统上的可执行文件。

kerne132.dll文件用数字1代替了字母l，显然是想将自己冒充混淆为Windows的系统文件kernel32.dll。因此，kerne132.dll可以作为一个基于主机的迹象来发现恶意代码感染。

- Lab01-01.dll

```
D:\>deskbook\strings.exe -a \deskbook\Malware\BinaryCollection\Chapter_1L\Lab01-01.dll

Strings v2.51
Copyright (C) 1999-2013 Mark Russinovich
Sysinternals - www.sysinternals.com
```

```
MSVCRT.dll
free
_initterm
malloc
_adjust_fdiv
exec
sleep
hello
127.26.152.13
SADFHUHF
/0I0[0h0p0
141G1[1l1
1Y2a2g2r2
3!3}3
```

CreateProcess和Sleep这两个导入函数普遍在后门程序中使用，这两个函数在与exec和sleep字符串结合使用时，需要特别的关注。

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。

6. 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？

```
MSVCRT.dll
free
_initterm
malloc
_adjust_fdiv
exec
sleep
hello
127.26.152.13
SADFHUHF
/0I0[0h0p0
141G1[1l1
1Y2a2g2r2
3!3}3
```

使用Strings工具可以看到Lab01-01.dll文件中包含一个**私有子网IP地址127.26.152.13的字符串**。结合其调用的WS2_32.dll，猜测该程序可能联网通信。该样本仅为实验示例，若为真正的恶意代码则可指向一个可路由的公网IP地址。该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。

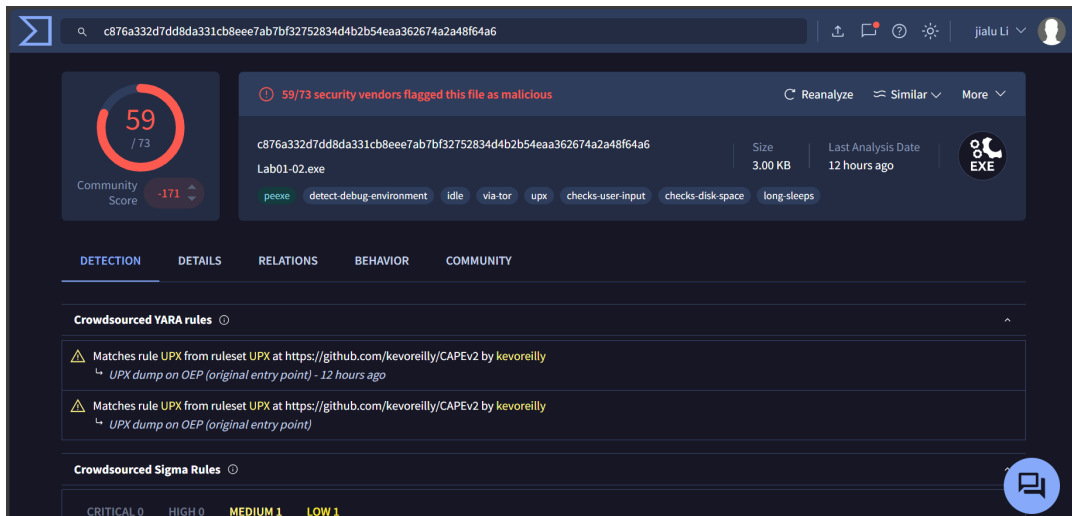
7. 你猜这些文件的目的是什么？

.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。

1. 将 Lab01-02.exe 文件上传至 <http://www.VirusTotal.com/> 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？



将Lab01-02.exe上传之后，可见文件匹配到了59个反病毒引擎。

2. 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。

迹象：

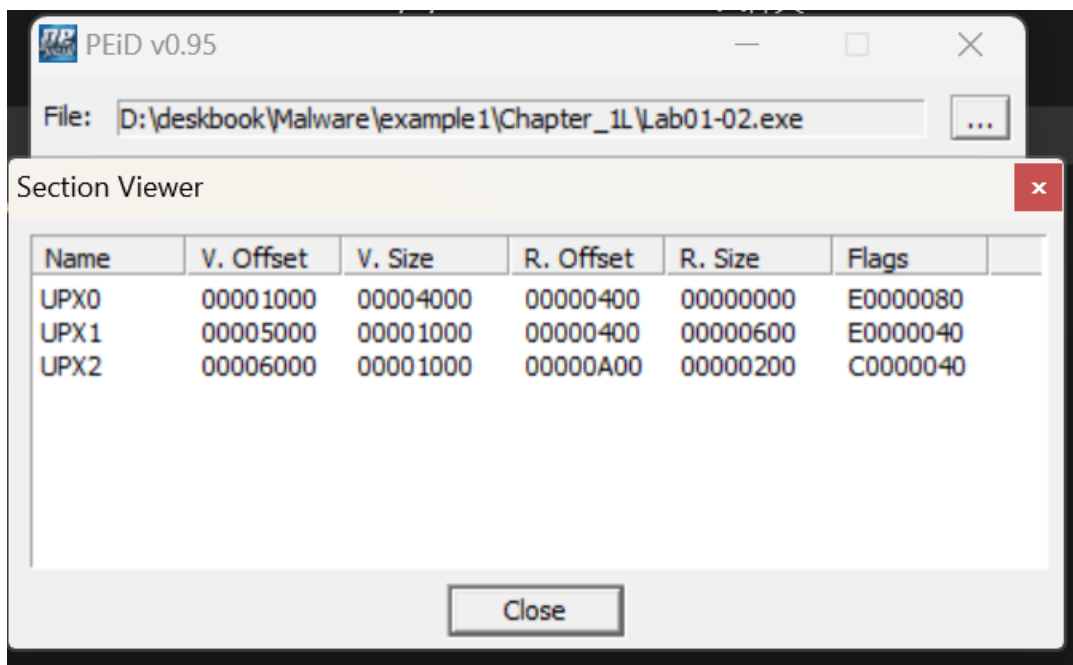
- 首先根据VirusTotal中的报告可知该文件采取了UPX加壳。



- 同时可以利用PEview工具打开该文件，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。

	pFile	Raw Data	Value
Lab01-02.exe			
IMAGE_DOS_HEADER	00000400	EF DD 77 FF 83 EC 10 8D 44 24 00 C7 03 10 30 40	..w....D\$....0@
MS-DOS Stub Program	00000410	00 50 08 08 40 10 40 10 B7 FD E9 DC 0C 00 00 07	.P..@.@.....
IMAGE_NT_HEADERS	00000420	10 FF 15 04 20 15 6A 01 BD FD FB 5D E8 0D 3C 83	...j...].<.
IMAGE_SECTION_HEADER UPX0	00000430	C4 18 C3 90 00 81 EC 00 04 0F 68 28 30 E9 BE E9	...h(0...
IMAGE_SECTION_HEADER UPX1	00000440	FE 1C 68 01 00 1F 29 20 85 C0 74 08 6A 0B 1C 67	...h...)..t.j..g
IMAGE_SECTION_HEADER UPX2	00000450	DF 17 AC 56 1E 0F 2C 45 03 0B 08 F6 6D EF 36 8B	...V...E...m.6.
SECTION UPX0	00000460	F0 7E 1C 68 E8 03 44 50 13 14 65 76 B7 FD 0B 01	...h...DP...ev...
SECTION UPX1	00000470	8D 4C 24 2C 05 51 0A 02 6A 10 03 D9 6C 63 EE 68	..L\$.Q..j...l.c.h
SECTION UPX2	00000480	1C 45 04 56 3B 00 33 D2 66 B7 EB BE 14 89 54 24	..E.V;3..f...T\$
	00000490	04 29 04 07 08 50 04 10 51 6C 49 B6 DF 18 66 C0	...P..QI...f..

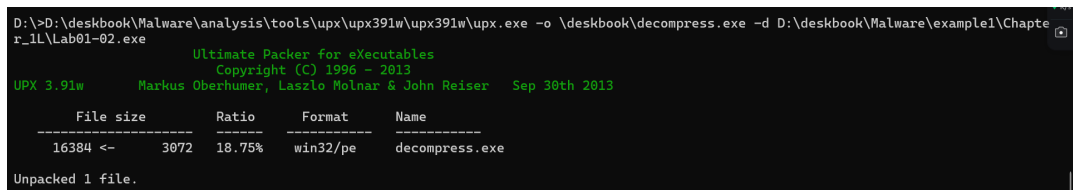
- 另外还可以使用PEiD工具观察到，该文件恶意代码拥有一个过小的导入表，即UPX0，虚拟大小为0x4000，而原始数据大小却为0.UPX0是长度最大的节，标记为可执行的，因此其中很可能包含了原始的未加壳代码。



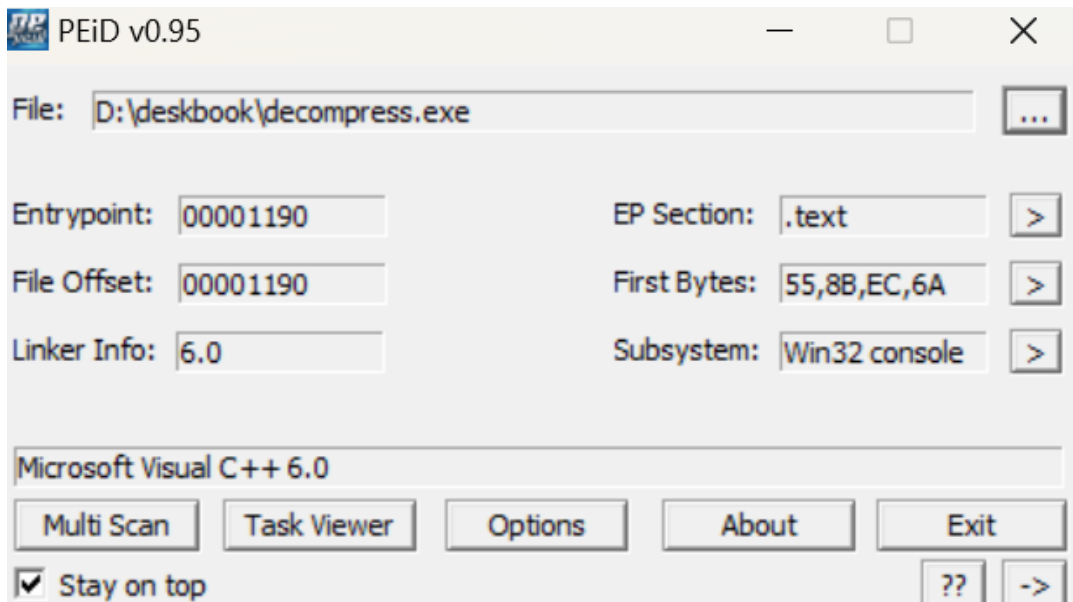
脱壳:

使用UPX工具对文件进行脱壳

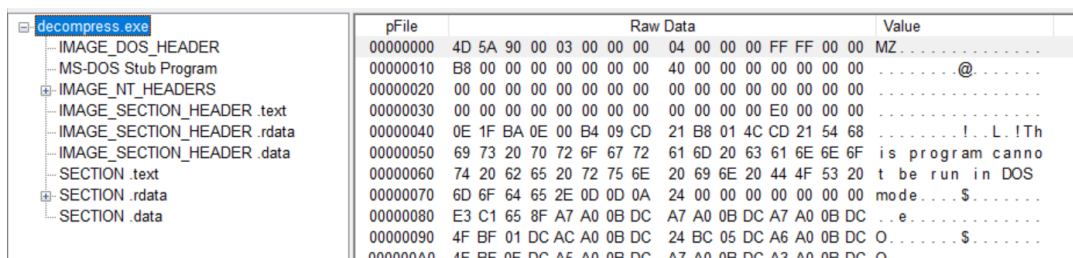
```
upx -o newFilename -d originalFilename
```



脱壳后文件再次使用PEiD和PEview



可见Microsoft Visual C++6.0编译

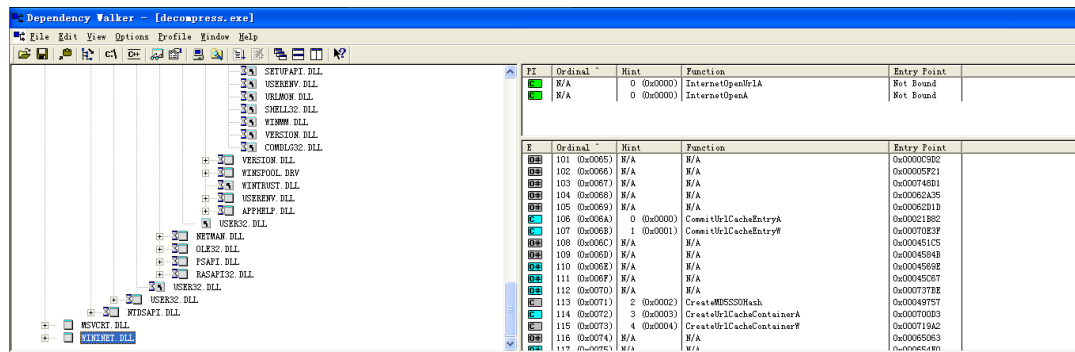


以及正常文件节结构

说明脱壳成功。

3. 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?

将脱壳后的恶意代码放入Dependency Walker, 查看恶意代码的导入表。



这个恶意代码的导入函数都是从kernel32.dll和msvcrt.dll, 而这些函数几乎被每个程序都导入, 所以它们能够告诉关于这个恶意代码的信息很少。

从wininet.dll导入的函数InternetOpen和InternetOpenURL, 可见这个恶意代码会进行联网操作; 从advapi32.dll导入的函数CreateService可见该代码会创建一个服务。

4. 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?

使用strings工具查看脱壳后文件字符串列表

```
D:\>deskbook\strings.exe -a \deskbook\decompress.exe
```

```
Strings v2.51  
Copyright (C) 1999-2013 Mark Russinovich  
Sysinternals - www.sysinternals.com
```

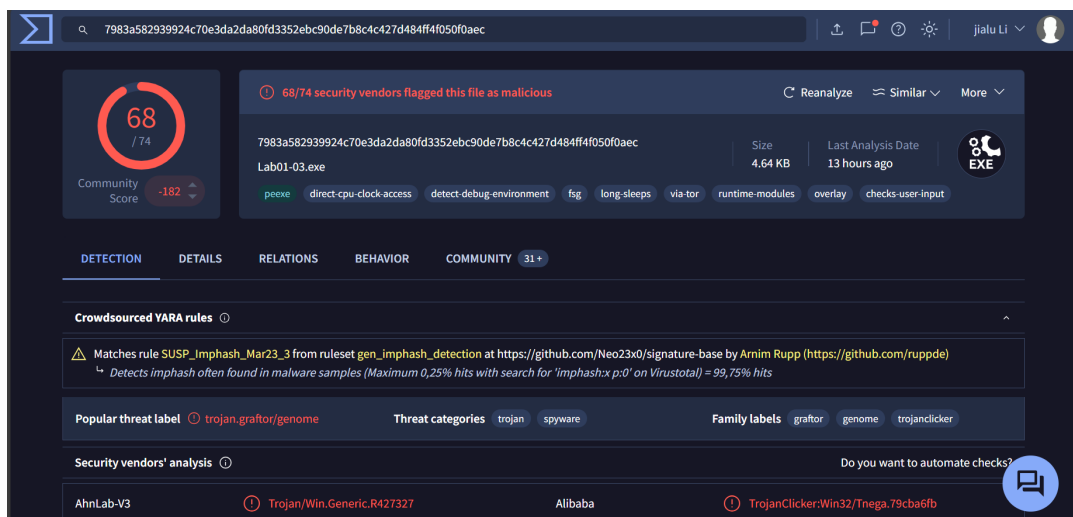
```
_except_handler3  
_controlfp  
InternetOpenUrlA  
InternetOpenA  
MalService  
Malservice  
HGL345  
http://www.malwareanalysisbook.com  
Internet Explorer 8.0
```

<http://www.malwareanalysisbook.com>, 这可能是InternetOpenURL函数中所打开的URL; MalService字符串, 这可能是所创建的服务名称。

• Lab1-3

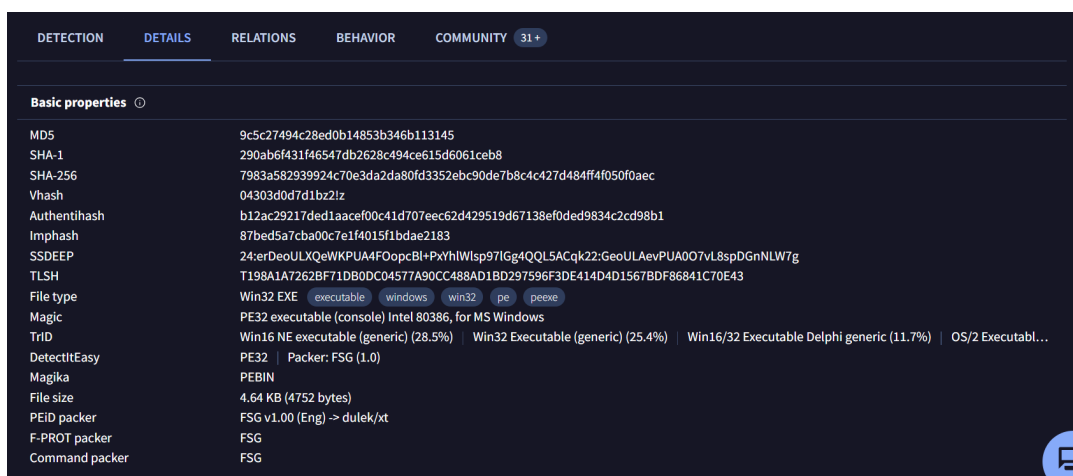
分析Lab01-03.exe文件。

1. 将Lab01-03.exe 文件上传至 <http://www.VirusTotal.com>/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?

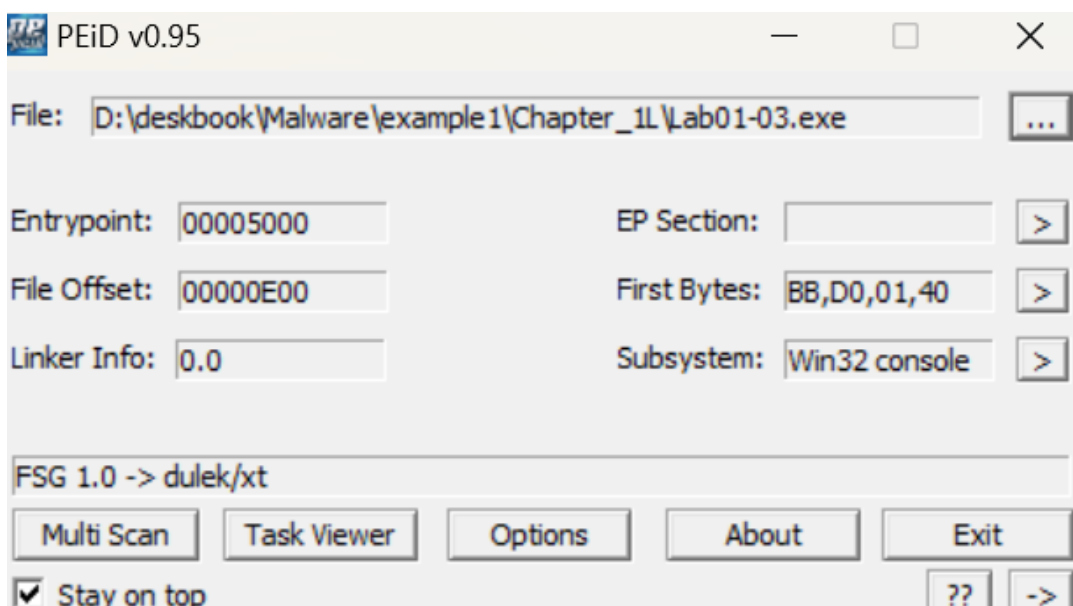


文件上传后，可见该文件匹配到了68个反病毒引擎。

2. 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。



在上题中的报告中可见该文件是一个由FSG加壳器进行加壳的文件。



使用PEiD工具打开Lab01-03.exe，可见该文件被标注为FSG 1.0 -> dulek/xt

由上述特征可以判断这个文件是加壳的，未脱壳前，只能在 kernel32.dll 中看到 LoadLibrary 和 GetProcAddress 导入函数。经过寻找万能脱壳软件、自学部分手动脱壳知识等还未能成功进行脱壳。因此后两问目前无法回答。

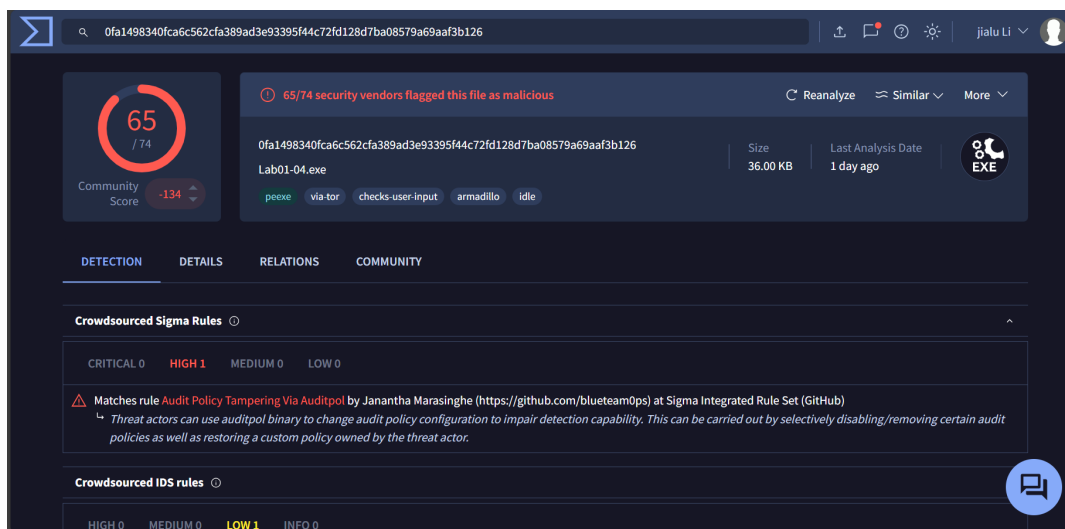
3. 有没有任何导入函数能够暗示出这个程序的功能?如果是，是哪些导入函数，它们会告诉你什么?

4. 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？

• Lab1-4

分析Lab01-04.exe文件。

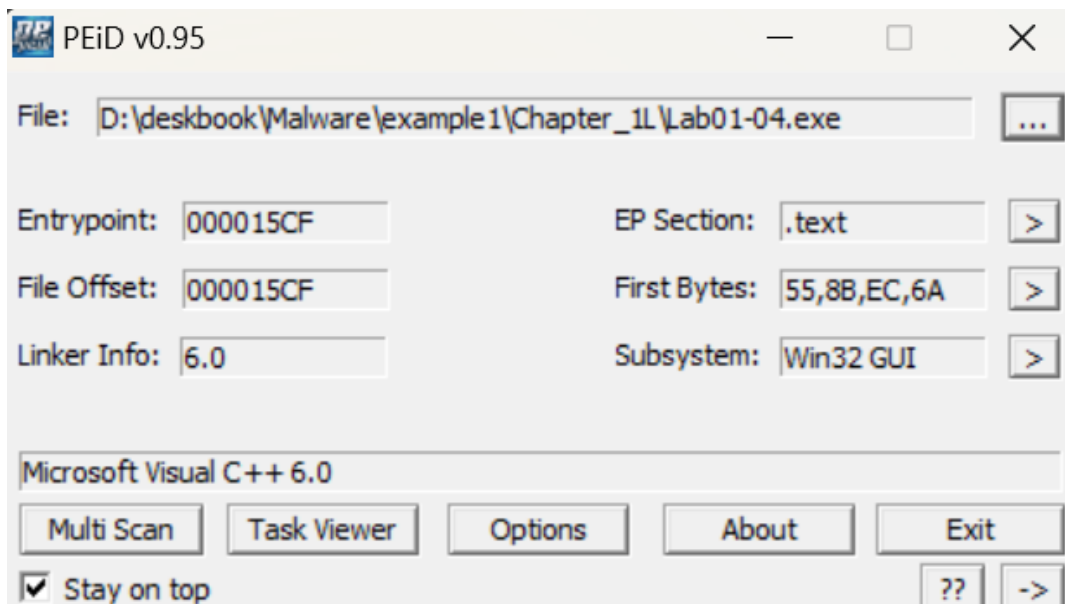
1. 将 Lab01-04.exe 文件上传至 <http://www.VirusTotal.com/> 进行分析并查看报告。文件匹配到了已有的反病毒软件特征吗？



该文件匹配到了65个反病毒引擎

2. 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。

将Lab01-04.exe导入到PEiD工具中，发现有适当大小的良好组织的文件节。PEiD工具也都标记为未加壳的代码，且是由Microsoft Visual C++编译的，可见文件没有被加壳。



Name	V. Offset	V. Size	R. Offset	R. Size	Flags
.text	00001000	00000720	00001000	00001000	60000020
.rdata	00002000	000003D2	00002000	00001000	40000040
.data	00003000	0000014C	00003000	00001000	C0000040
.rsrc	00004000	00004060	00004000	00005000	40000040

3. 这个文件是什么时候被编译的?

	pFile	Raw Data	Value
000000E8	50 45 00 00 4C 01 04 00	B3 A2 69 5D 00 00 00 00	PE . . L i]
000000F8	00 00 00 00 E0 00 0F 01	0B 01 06 00 00 10 00 00	
00000108	00 70 00 00 00 00 00 00	CF 15 00 00 00 10 00 00	. p
00000118	00 20 00 00 00 00 40 00	00 10 00 00 00 10 00 00 @
00000128	04 00 00 00 00 00 00 00	04 00 00 00 00 00 00 00	
00000138	00 90 00 00 00 10 00 00	00 00 00 00 02 00 00 00	
00000148	00 00 10 00 00 10 00 00	00 00 10 00 00 10 00 00	
00000158	00 00 00 00 10 00 00 00	00 00 00 00 00 00 00 00	
00000168	A4 20 00 00 50 00 00 00	00 40 00 00 60 40 00 00	. . . P . . . @ . . . @ .
00000178	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000188	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000198	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000001A8	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000001B8	00 00 00 00 00 00 00 00	00 20 00 00 94 00 00 00	
000001C8	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000001D8	00 00 00 00 00 00 00 00		

5D69 A2B3 => 1,567,204,019

时间戳	1567204019	秒(s)	▼	转换	2019-08-31 06:26:59
-----	------------	------	---	----	---------------------

借助PEview工具进行查看时间戳,可见文件编译时间为2019年8月,但同其创建时间相比,显然这个编译时间是伪造的,现下还不能确定这个文件是什么时候被编译的。

4. 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?

使用Dependency Walker工具,查看该文件导入表

#	Ordinal	Hint	Function	Entry Point
1	(0x0001)	294 (0x0128)	I_SGetCurentGroupStateW	0x000868FC
2	(0x0002)	0 (0x0000)	A_SwapInitial	0x0000B1FD
3	(0x0003)	1 (0x0001)	A_SwapInit	0x0000B140
4	(0x0004)	2 (0x0002)	A_SwapUpdate	0x0000B1A1
5	(0x0005)	3 (0x0003)	AbortSystemShutdownW	0x00004290
6	(0x0006)	4 (0x0004)	AbortSystemShutdownW	0x0000242B
7	(0x0007)	5 (0x0005)	AccessCheck	0x00007390
8	(0x0008)	6 (0x0006)	AccessCheckAndAuditAlarmW	0x00003210
9	(0x0009)	7 (0x0007)	AccessCheckAndAuditAlarmW	0x00003C3C
10	(0x000A)	8 (0x0008)	AccessCheckByType	0x0000F199

从ADVAPI32.DLL导入的函数,表示程序做了一些与权限有关的事情,可能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LoadResource, FindResource和SizeofResource可知这个程序从资源节中装载数据,以及CreateFile, WriteFile可知该程序将数据写到磁盘上,另外WinExec可知程序接着执行一个磁盘上的文件。

该程序还导入了GetWindowsDirectory, 程序可能将文件写入到了系统目录。

5. 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?

使用strings工具查看该文件所包含的字符串

```
D:\>deskbok\strings.exe -a \deskbok\Malware\example1\Chapter_1L\Lab01-04.exe
```

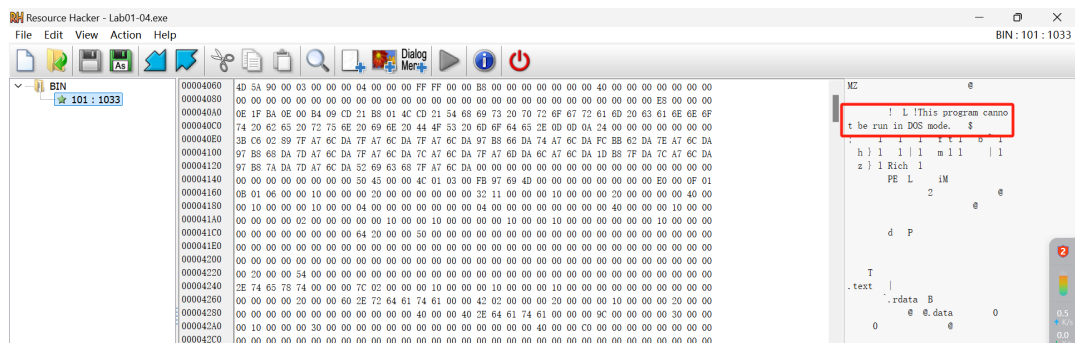
```
Strings v2.51
Copyright (C) 1999-2013 Mark Russinovich
Sysinternals - www.sysinternals.com
```

```
__set_app_type
_except_handler3
_controlfp
\winup.exe
%s%s
\system32\wupdmgrd.exe
%s%s
http://www.practicalmalwareanalysis.com/updater.exe
```

可见字符串<http://www.practicalmalwareanalysis.com/updater.exe>，可能是保存下载恶意代码的网络位置；另外还有字符串\system32\wupdmgrd.exe，结合GetWindowsDirectory函数调用，这表明恶意代码在C:\Windows\System32\wupdmgrd.exe位置创建或者修改了一个文件。

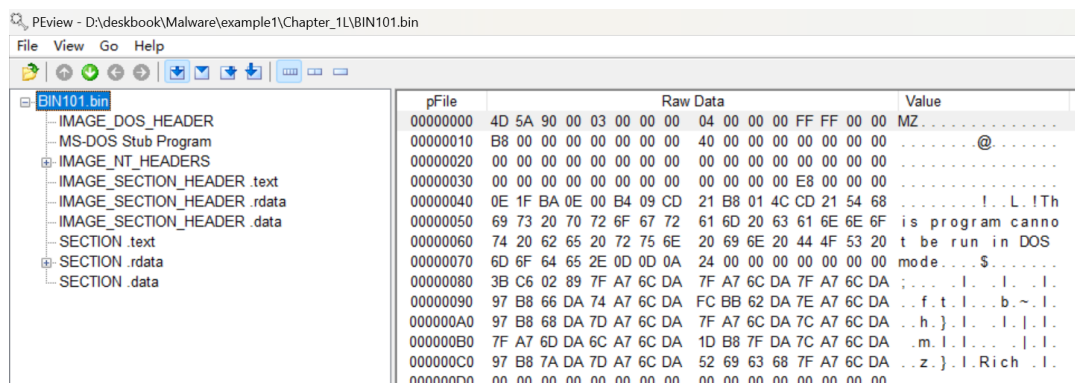
6. 这个文件在资源段中包含一个资源。使用Resource Hacker 工具来检查资源，然后抽取资源。从资源中你能发现什么吗？

使用ResourceHacker工具来检查资源



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。所以可见这一资源其实是在Lab01-04.exe资源节中存储的另一个可执行文件。

为了在Resource Hacker工具中继续分析这一文件，存储该资源为二进制文件，使用PEview打开这个文件，分析内嵌的文件。



	pFile	Raw Data	Value
IMAGE_DOS_HEADER	000020F0	00 22 00 00 0E 22 00 00 20 22 00 00 00 00 00 00
MS-DOS Stub Program	00002100	48 21 00 00 00 00 00 00 7D 01 47 65 74 57 69 6E	H!.....}GetWin
IMAGE_NT_HEADERS	00002110	64 6F 77 73 44 69 72 65 63 74 6F 72 79 41 00 00	dowsDirectoryA..
IMAGE_SECTION_HEADER .text	00002120	D3 02 57 69 6E 45 78 65 63 00 65 01 47 65 74 54	..WinExec.e.GetT
IMAGE_SECTION_HEADER .rdata	00002130	65 6D 70 50 61 74 68 41 00 00 4B 45 52 4E 45 4C	empPathA..KERNEL
IMAGE_SECTION_HEADER .data	00002140	33 32 2E 64 6C 6C 00 00 3E 00 55 52 4C 44 6F 77	32.dll..>..URLDow
SECTION .text	00002150	6E 6C 6F 61 64 54 6F 46 69 6C 65 41 00 00 75 72	nloadToFileA..ur
SECTION .rdata	00002160	6C 6D 6F 6E 2E 64 6C 6C 00 00 AE 01 5F 73 6E 70	lmon.dll....._snp
SECTION .data	00002170	72 69 6E 74 66 00 4D 53 56 43 52 54 2E 64 6C 6C	rintf.MSVCRT.dll
	00002180	00 00 D3 00 5F 65 78 69 74 00 48 00 5F 58 63 70_exit.H._Xcp
	00002190	74 46 69 6C 74 65 72 00 49 02 65 78 69 74 00 00	tFilter.l.exit..
	000021A0	64 00 5F 5F 70 5F 5F 5F 69 6E 69 74 65 6E 76 00	d...p..._initenv.
	000021B0	58 00 5F 5F 67 65 74 6D 61 69 6E 61 72 67 73 00	X..._getmainargs.
	000021C0	0F 01 5F 69 6E 69 74 74 65 72 6D 00 83 00 5F 5F	..._initterm....
	000021D0	73 65 74 75 73 65 72 6D 61 74 68 65 72 72 00 00	setusermatherr..
	000021E0	9D 00 5F 61 64 6A 75 73 74 5F 66 64 69 76 00 00	.._adjust_fdiv..
	000021F0	6A 00 5F 5F 70 5F 5F 63 6F 6D 6D 6F 64 65 00 00	j..._p..._commode..
	00002200	6F 00 5F 5F 70 5F 5F 66 6D 6F 64 65 00 00 81 00	o..._p..._fmode....
	00002210	5F 5F 73 65 74 5F 61 70 70 5F 74 79 70 65 00 00	.._set_app_type..
	00002220	CA 00 5F 65 78 63 65 70 74 5F 68 61 6E 64 6C 65	.._except_handle
	00002230	72 33 00 00 B7 00 5F 63 6F 6E 74 72 6F 6C 66 70	r3...._controlfp
	00002240	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
	00002250	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

通过查看导入表，可见嵌入文件在访问一些网络函数，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。

• Yara

对Lab1样本编写Yara检测规则，并进行测试。

检测规则均依据先前对恶意代码分析的结果

判断是否为PE文件，规则内容如下

```
filesize<10MB and uint16(0)==0x5A4D and uint16(uint16(0x3C))==0x00004550
```

o Lab01-01.exe

KERNEL32.DLL导入的函数中包括一些打开与操作文件的函数，以及FindFirstFile和FindNextFile函数。可见该恶意代码会对文件系统进行搜索，以及打开和修改文件，kerne132.dll冒充混淆为Windows的系统文件kernel32.dll。因此，kerne132.dll可以作为一个基于主机的迹象来发现恶意代码感染。

```
rule lab11exe
{
  strings:
    $string1 = "kerne132.dll"
    $string2 = "FindFirstFile"
    $string3 = "FindNextFile"
  condition:
    filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) ==
    0x00004550 and $string1 and $string2 and $string3 and $string4
}
```

```
C:\Users\DELL>D:\deskbook\Malware\analysis\tools\yara\yara64.exe D:\deskbook\Malware\analysis\tools\yara\Lab1-1.yar D:\deskbook\Malware\example1\Chapter_11\lab11exe D:\deskbook\Malware\example1\Chapter_11\Lab01-01.exe
```

(测试时检测路径设置到对应文件夹)

o Lab01-01.dll

kernel32.dll中导入了CreateProcess和Sleep这两个函数，而这两个函数普遍在后门程序中使用。这两个函数在与exec和sleep字符串结合使用时，需要特别的关注。


```
rule lab11dll
{
  strings:
    $string1 = "sleep"
    $string2 = "exec"
    $string3 = "CreateProcess"
  condition:
    filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) ==
    0x00004550 and $string1 and $string2 and $string3
}
```

```
C:\Users\DELL>D:\deskbook\Malware\analysis\tools\yara\yara64.exe D:\deskbook\Malware\analysis\tools\yara\Lab1-12.yar D:\deskbook\Malware\BinaryCollection\Chapter_11\lab11dll D:\deskbook\Malware\BinaryCollection\Chapter_11\Lab01-01.dll
```

○ Lab01-02.exe

从wininet.dll导入的函数InternetOpen和InternetOpenURL，可见这个恶意代码会进行联网操作；从advapi32.dll导入的函数CreateService可见该代码会创建一个服务。MalService字符串，这可能是所创建的服务名称。<http://www.malwareanalysisbook.com>，这可能是InternetOpenURL函数中所打开的URL。

- 直接利用Yara对<http://www.malwareanalysisbook.com>字符串进行分析可以发现，无法发现该恶意代码，于是利用IDA进行静态分析，可以发现一些残缺的混有其他字符的字符串。

```
rule lab12exe
{
  strings:
    $string1 = "HGL345"
    $string2 = "MalService"
    $location = {68 74 74 70 3A 2F 2F 77 FF B7 BF DD 00 2E 6D 1E 77 61 72 65
    61 6E 07 79 73 69 73 62 6F 6F 6B 2E 63 6F FF DB DB 6F 6D}

  condition:
    filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) ==
    0x00004550 and $string1 and $string2 and $location
}
```

```
C:\Users\DELL>D:\deskbook\Malware\analysis\tools\yara\yara64.exe D:\deskbook\Malware\analysis\tools\yara\Lab1-2.yar D:\deskbook\Malware\example1\Chapter_11\lab12exe D:\deskbook\Malware\example1\Chapter_11\Lab01-02.exe
```

○ Lab01-03.exe

由于未能成功脱壳，故只能对脱壳前程序中可疑字符串进行分析检测。

```
rule lab13exe
{
  strings:
    $string1 = "ole32.vd"
    $string2 = "OLEAUTLA"
    $string3 = "_getmas"
  condition:
    filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) ==
    0x00004550 and $string1 and $string2 and $string3
}
```

```
C:\Users\DELL>D:\deskbook\Malware\analysis\tools\yara\yara64.exe D:\deskbook\Malware\analysis\tools\yara\Lab1-3.yar D:\deskbook\Malware\example1\Chapter_11\lab13exe D:\deskbook\Malware\example1\Chapter_11\Lab01-03.exe
```

o Lab01-04.exe

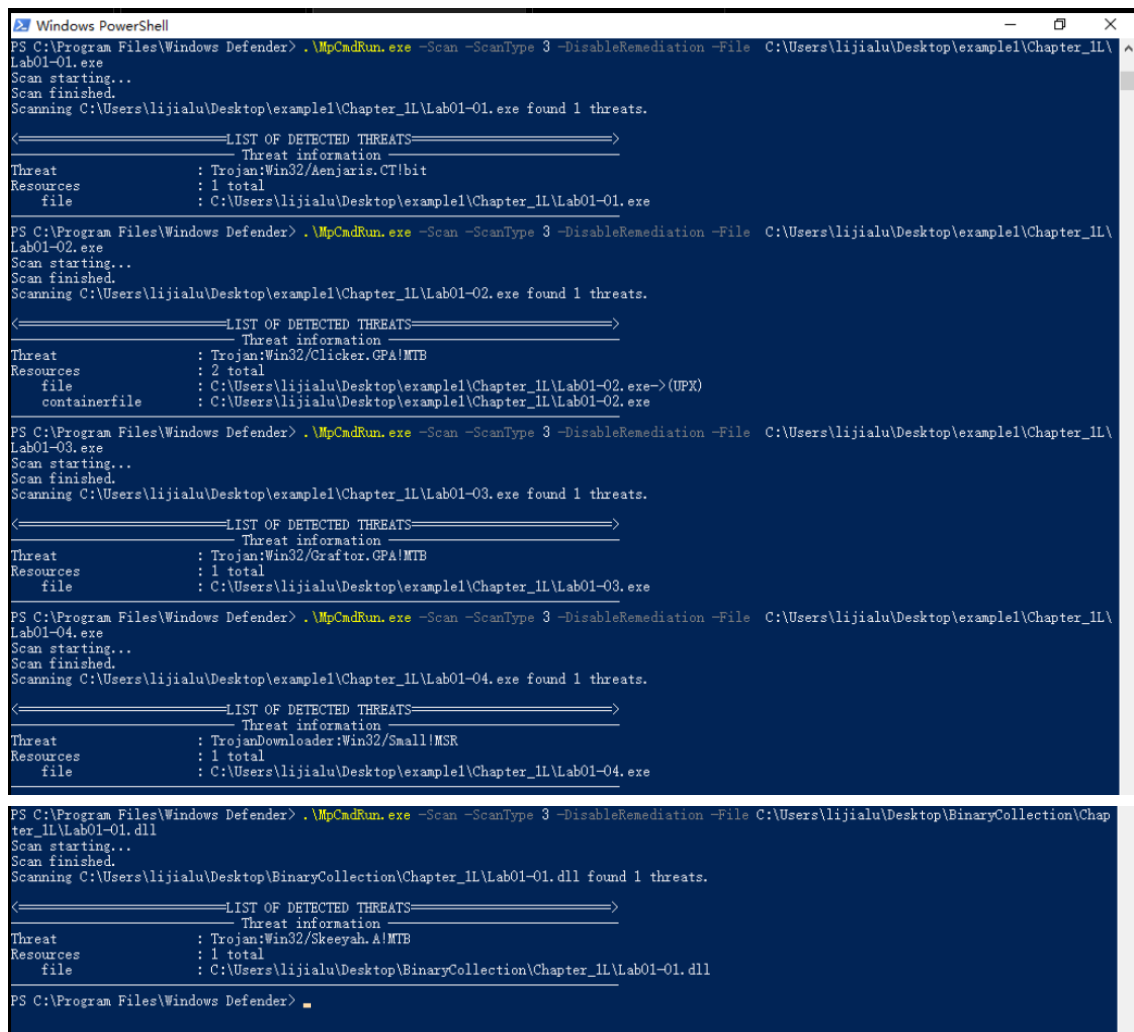
通过上述分析, 可发现其中含有诸如LoadResource、FindResource和SizeofResource等的函数, 以及"\\system32\\wupdmgr.exe"等一些可疑字符串, 将其作为检测规则。

```
rule lab14exe
{
  strings:
    $string1 = "LoadResource"
    $string2 = "FindResource"
    $string3 = "SizeofResource"
    $string4 = "\\system32\\wupdmgr.exe"
    $string5 = "http://www.practicalmalwareanalysis.com/updater.exe"
  condition:
    filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) ==
    0x00004550 and $string1 and $string2 and $string3 and $string4 and
    $string5
}
```

```
C:\Users\DELL>D:\deskbook\Malware\analysis\tools\yara\yara64.exe D:\deskbook\Malware\analysis\tools\yara\Lab1-4.yar D:\deskbook\Malwa
re\example1\Chapter_11\
lab14exe D:\deskbook\Malware\example1\Chapter_11\Lab01-04.exe
```

• Microsoft Defender AntiVirus

在本机上有电脑管家防护, 转到win10上进行Microsoft Defender AntiVirus检测, 均成功检查出威胁。



```
Windows PowerShell
PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -DisableRemediation -File C:\Users\lijialu\Desktop\example1\Chapter_11\
Lab01-01.exe
Scan starting...
Scan finished.
Scanning C:\Users\lijialu\Desktop\example1\Chapter_11\Lab01-01.exe found 1 threats.

<=====LIST OF DETECTED THREATS=====>
Threat information
Threat      : Trojan:Win32/Aenjaris.CT!bit
Resources   : 1 total
file        : C:\Users\lijialu\Desktop\example1\Chapter_11\Lab01-01.exe

PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -DisableRemediation -File C:\Users\lijialu\Desktop\example1\Chapter_11\
Lab01-02.exe
Scan starting...
Scan finished.
Scanning C:\Users\lijialu\Desktop\example1\Chapter_11\Lab01-02.exe found 1 threats.

<=====LIST OF DETECTED THREATS=====>
Threat information
Threat      : Trojan:Win32/Clicker.GPA!MTB
Resources   : 2 total
file        : C:\Users\lijialu\Desktop\example1\Chapter_11\Lab01-02.exe->(UPX)
containerfile : C:\Users\lijialu\Desktop\example1\Chapter_11\Lab01-02.exe

PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -DisableRemediation -File C:\Users\lijialu\Desktop\example1\Chapter_11\
Lab01-03.exe
Scan starting...
Scan finished.
Scanning C:\Users\lijialu\Desktop\example1\Chapter_11\Lab01-03.exe found 1 threats.

<=====LIST OF DETECTED THREATS=====>
Threat information
Threat      : Trojan:Win32/Graftor.GPA!MTB
Resources   : 1 total
file        : C:\Users\lijialu\Desktop\example1\Chapter_11\Lab01-03.exe

PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -DisableRemediation -File C:\Users\lijialu\Desktop\example1\Chapter_11\
Lab01-04.exe
Scan starting...
Scan finished.
Scanning C:\Users\lijialu\Desktop\example1\Chapter_11\Lab01-04.exe found 1 threats.

<=====LIST OF DETECTED THREATS=====>
Threat information
Threat      : TrojanDownloader:Win32/Small!MSR
Resources   : 1 total
file        : C:\Users\lijialu\Desktop\example1\Chapter_11\Lab01-04.exe

PS C:\Program Files\Windows Defender> .\MpCmdRun.exe -Scan -ScanType 3 -DisableRemediation -File C:\Users\lijialu\Desktop\BinaryCollection\Chap
ter_11\Lab01-01.dll
Scan starting...
Scan finished.
Scanning C:\Users\lijialu\Desktop\BinaryCollection\Chapter_11\Lab01-01.dll found 1 threats.

<=====LIST OF DETECTED THREATS=====>
Threat information
Threat      : Trojan:Win32/Skeeyah.A!MTB
Resources   : 1 total
file        : C:\Users\lijialu\Desktop\BinaryCollection\Chapter_11\Lab01-01.dll

PS C:\Program Files\Windows Defender>
```

分析结果说明:

1. Lab01-01.exe:

Trojan:Win32/Aenjaris.CT!bit: 该木马通常用于感染系统并执行恶意活动，例如下载其他恶意软件或窃取敏感信息。

2. Lab01-02.exe:

Trojan:Win32/Clicker.GPA!MTB: 这种木马通常会伪装成合法软件，主要用于生成虚假流量，可能会导致广告弹出和其他不必要的干扰。

3. Lab01-03.exe:

Trojan:Win32/Graftor.GPA!MTB: 此木马可以远程控制受感染的设备，通常用于下载其他恶意软件或进行数据窃取。

4. Lab01-04.exe:

TrojanDownloader:Win32/Small!MSR: 这种下载器木马专门用于下载并安装其他恶意软件，可能导致系统进一步受到感染。

5. Lab01-01.dll:

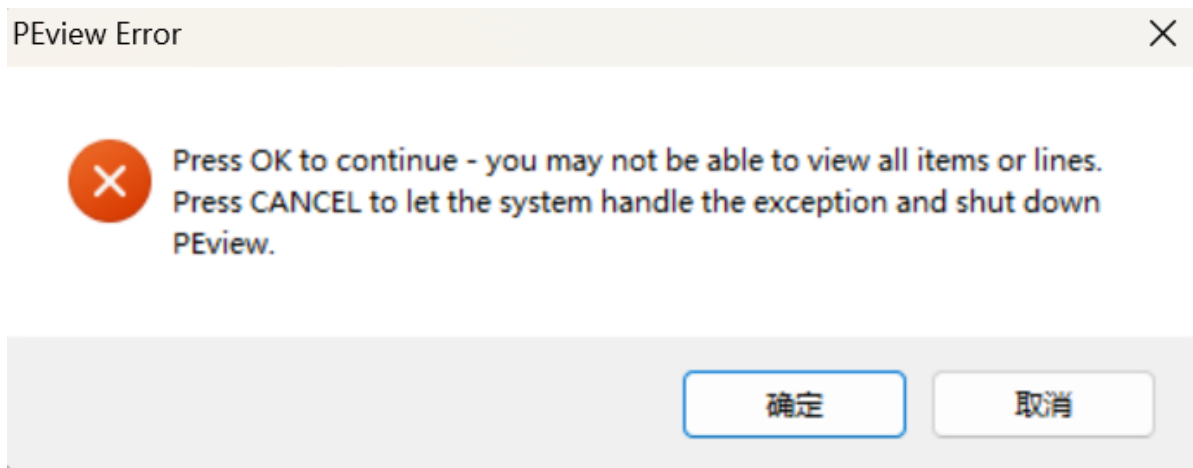
Trojan:Win32/Skeeyah.A!MTB: 该木马通常与网络攻击和数据盗窃相关，能够影响系统安全并执行未经授权的操作。

四、实验结论及心得体会

1. 实验过程中遇到的部分问题

(1) 查看时间戳

- 正常情况下，对于每个文件可通过浏览IMAGE_NT_HEADERS -> IMAGE_FILE_HEADER -> TIME Data Stamp字段查看时间戳。
- 但在本次实验过程中点击IMAGE_FILE_HEADER时会报错：Press OK to continue - you may not be able to view all items or lines.Press CANCEL to let the system handle the exception and shut downPEview.



IMAGE_FILE_HEADER下不显示时间戳

Lab01-01.dll	pFile	Data	Description	Value
IMAGE_DOS_HEADER	000000E4	014C	Machine	IMAGE_FILE_MACHINE_I386
MS-DOS Stub Program	000000E6	0004	Number of Sections	
IMAGE_NT_HEADERS				
Signature	000000EC	00000000	Pointer to Symbol Table	
IMAGE_FILE_HEADER	000000F0	00000000	Number of Symbols	
IMAGE_OPTIONAL_HEADER	000000F4	00E0	Size of Optional Header	
IMAGE_SECTION_HEADER .text	000000F6	210E	Characteristics	
IMAGE_SECTION_HEADER .rdata			0002	IMAGE_FILE_EXECUTABLE_IMAGE
IMAGE_SECTION_HEADER .data			0004	IMAGE_FILE_LINE_NUMS_STRIPPED
IMAGE_SECTION_HEADER .reloc			0008	IMAGE_FILE_LOCAL_SYMS_STRIPPED
SECTION .text			0100	IMAGE_FILE_32BIT_MACHINE
SECTION .rdata			2000	IMAGE_FILE_DLL
SECTION .data				
SECTION .reloc				

- 可能原因为：获取日期格式化显示时我们地区是中文 unicode 字符，导致缓冲区溢出，无法正常显示

[记一次 PView 的报错修正 peviewx-CSDN博客](#)

- 直接通过查看NT头查找时间戳

[chapter1 静态分析技术-08PE文件分析 PView-CSDN博客](#)

PView - D:\deskbook\恶意代码\上机实验样本\Chapter_1\Lab01-01.exe				
File View Go Help				
Lab01-01.exe	pFile	Raw Data		Value
IMAGE_DOS_HEADER	000000E8	50 45 00 00 4C 01 03 00	D3 2F 0E 4D 00 00 00 00	PE . . L . . . / . M . . .
MS-DOS Stub Program	000000F8	00 00 00 00 E0 00 0F 01	0B 01 06 00 00 10 00 00
IMAGE_NT_HEADERS	00000108	00 20 00 00 00 00 00 00	20 18 00 00 00 10 00 00
IMAGE_SECTION_HEADER .text	00000118	00 20 00 00 00 00 40 00	00 10 00 00 00 10 00 00 @
IMAGE_SECTION_HEADER .rdata	00000128	04 00 00 00 00 00 00 00	04 00 00 00 00 00 00 00
IMAGE_SECTION_HEADER .data	00000138	00 40 00 00 00 10 00 00	00 00 00 00 03 00 00 00	. @
SECTION .text	00000148	00 00 10 00 00 10 00 00	00 00 10 00 00 10 00 00
SECTION .rdata	00000158	00 00 00 00 10 00 00 00	00 00 00 00 00 00 00 00
SECTION .data	00000168	7C 20 00 00 3C 00 00 00	00 00 00 00 00 00 00 00	. . <
	00000178	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
	00000188	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
	00000198	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
	000001A8	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
	000001B8	00 00 00 00 00 00 00 00	00 20 00 00 6C 00 00 00 I
	000001C8	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
	000001D8	00 00 00 00 00 00 00 00	

另外也可以通过查找程序补丁解决该问题。

(2) Lab01-02.exe yara规则检测

直接利用Yara对<http://www.malwareanalysisbook.com>字符串进行分析可以发现，无法发现该恶意代码，于是利用IDA进行静态分析，可以发现一些残缺的混有其他字符的字符串。

规则转换为ASCII码

```
$location = {68 74 74 70 3A 2F 2F 77 FF B7 BF DD 00 2E 6D 1E 77 61 72 65 61 6E 07
79 73 69 73 62 6F 6F 6B 2E 63 6F FF DB DB 6F 6D}
```

2. 心得体会

(1) 初步了解了恶意代码的简单分析过程

在整个实验过程中，对恶意代码的分析过程有了初步了解。先将文件上传到 <http://www.VirusTotal.com/>，对代码先进行初步检查，针对各病毒引擎的检查结果对恶意代码类型和加工方式有初步认识，同时需要检查是否有加壳的情况，如有，则最好先进行脱壳再继续后续分析，然后分析其导入函数和字符串，结合考察该恶意代码的功能。

(2) 初步认识了恶意代码的分析工具的功能和使用

在上述恶意代码的分析过程中，每个阶段使用不同的恶意代码的分析工具，在实验过程中也学习了各实验工具的使用方法，能够较熟练地使用各分析工具，同时对于各分析工具的功能在实验过程中也有了更深的理解。

(3) 对于一些恶意代码处理仍有不足

现阶段只掌握一些简单的静态分析方法，也只会使用简单工具进行针对性脱壳，对于像FSG加壳情况还无法处理，经过寻找万能脱壳软件、自学部分手动脱壳知识等还未能成功进行脱壳。需要后期学习手动脱壳技巧后才能完整分析Lab1-3这个恶意代码。