

南开大学

恶意代码分析与防治技术实验报告

实验三：动态分析基础技术



学 院 网络空间安全学院
专 业 信息安全、法学
学 号 2112514
姓 名 辛浩然
班 级 信息安全、法学

一、实验目的

1. 熟悉动态分析基础技术；
2. 综合使用静态分析技术和动态分析基础技术分析恶意代码；
3. 编写 Yara 规则。

二、实验原理

动态分析是在运行恶意代码后进行检查的过程。通常，动态分析技术在静态分析技术无法继续进行时使用，比如当恶意代码经过混淆或者已经尝试了所有可用的静态分析技术。动态分析技术包括在恶意代码运行时监控其行为，以及在代码执行后检查系统状态。

与静态分析不同，动态分析允许观察恶意代码的实际功能。在二进制程序中存在一个行为并不意味着它会被执行。动态分析也是一种有效识别恶意代码功能的方法。例如，如果要分析的恶意代码是键盘记录程序，动态分析可以帮助找到键盘记录程序的日志文件，发现其记录内容，追踪信息发送至何处等。这种深入的洞察力通常难以通过静态分析技术获得。

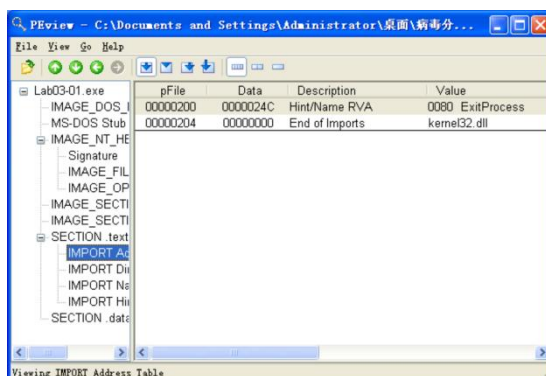
尽管动态分析技术非常有用，但它们应该在静态分析技术之后使用，并且必须确保建立安全环境。因为动态分析可能会对网络和系统造成风险。此外，动态分析技术也有其局限性，因为在代码执行过程中，并不是所有代码都会被执行。例如，在需要参数的命令行恶意代码中，每个参数可能会触发不同的程序功能。如果不了解所有命令行选项，就无法通过动态分析获得所有的程序功能。解决如何执行恶意代码的所有功能的问题通常需要借助高级的动态或静态分析技术。

三、实验过程

Lab 3-1 使用动态分析基础技术来分析在 Lab03-01.exe 文件中发现的恶意代码。

3.1.1 找出这个恶意代码的导入函数与字符串列表。

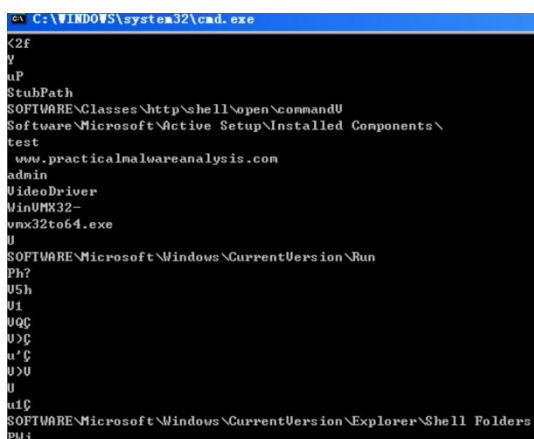
使用 PEView 工具查看恶意代码的导入表：



可以发现，只有一个导入函数——ExitProcess。猜测可能是加过壳的程序，于是在 PEiD 中打开，确认确实是加过壳的。



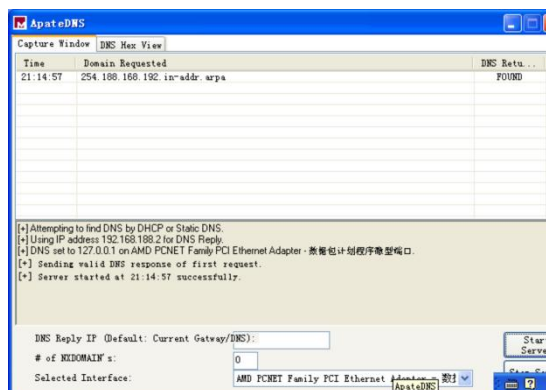
再来查看字符串，可以看到一些域名、注册表位置、VideoDriver、vmx32to64.exe 等字符串。初步猜测恶意代码可能会修改注册表信息，具有联网功能。



3.1.2 这个恶意代码在主机上的感染迹象特征是什么？

对恶意代码进行动态调试。在运行恶意代码之前，首先配置相关环境。

先运行 Process Monitor 工具，并清除所有事件；启动 Process Explorer，同时配置出一个虚拟网络，包括 ApateDNS、netcat 监听(端口 80 和 443)以及用于网络数据包捕获的 Wireshark。

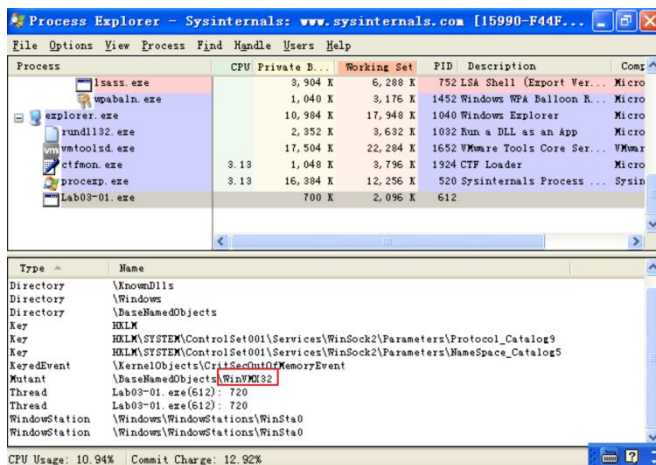


在 Regshot 中拍摄快照。

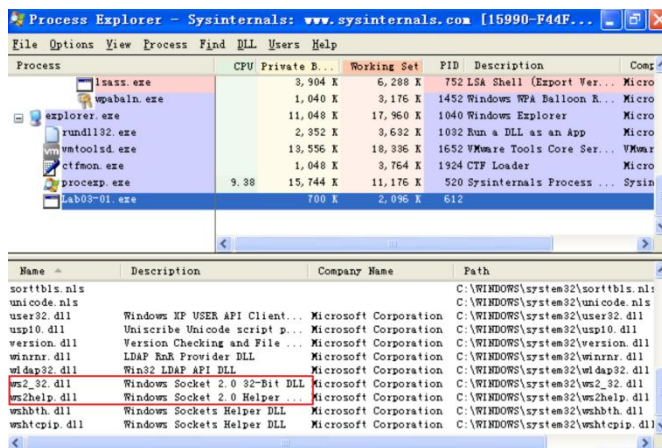


运行恶意代码。

在 Process Explorer 中可以检查到该进程。进一步查看 handles，我们可以看到，恶意代码创建了一个名为 WinVMX32 的互斥量；



看到恶意代码动态装载的 DLL 文件，如 ws2_32.dll 和 wshtcpip.dll，这意味着它具有联网功能。



在运行恶意代码之后再次使用 Regshot 拍摄快照，对比两次快照，可以发现系统值的增加和修改：

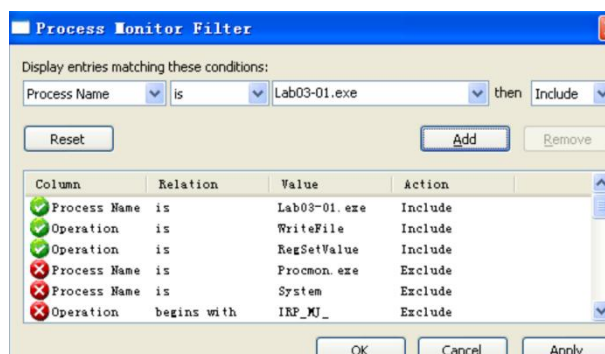
Values added: 8

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\VideoDriver: 43 00 3A 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
HKU\S-1-5-21-1993962763-2025429265-1606980848-500\Software\Microsoft\Windows\
HKU\S-1-5-21-1993962763-2025429265-1606980848-500\Software\Microsoft\Windows\
HKU\S-1-5-21-1993962763-2025429265-1606980848-500\Software\Microsoft\Windows\
HKU\S-1-5-21-1993962763-2025429265-1606980848-500\Software\Microsoft\Windows\
HKU\S-1-5-21-1993962763-2025429265-1606980848-500\Software\Microsoft\Windows\
HKU\S-1-5-21-1993962763-2025429265-1606980848-500\Software\Microsoft\Windows\
HKU\S-1-5-21-1993962763-2025429265-1606980848-500\Software\Microsoft\Windows\
```

```
49 00 4E 00 44 00 4F 00 57 00 53 00 5C 00 73 00 79 00 73 00 74 00 65 00 6C
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.on\Explorer\UserAssist\{75048700-EF1F-11D0-9888-000697DEACF9}\Count\HRZR_E
.MUICache\@shell132.dll,-31254: "重命名这个文件夹"
.MUICache\@shell132.dll,-31256: "移动这个文件夹"
.MUICache\@shell132.dll,-31258: "复制这个文件夹"
.MUICache\@shell132.dll,-31380: "以电子邮件形式发送该文件夹内的文件"
.MUICache\@shell132.dll,-31262: "删除这个文件夹"
.MUICache\C:\Documents and Settings\Administrator\桌面\病毒分析样本\BinaryC
```

可以发现新建了注册表项 VideoDriver，以及进行了一些文件操作。

接下来，使用 Process Monitor 工具详细分析。首先设置三个过滤器：一个是对进程名称的过滤和两个操作上的过滤，包含了 RegSetValue 和 WriteFile，来查看恶意代码对文件系统和注册表的修改操作。

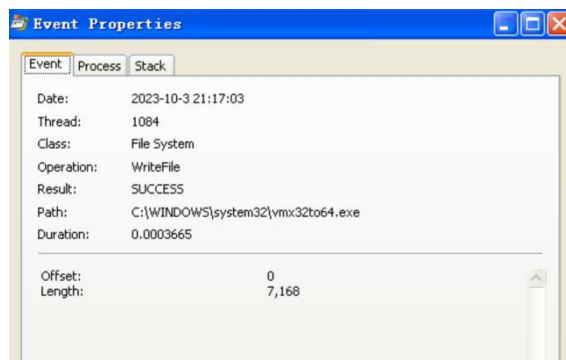


过滤到如下条目信息：

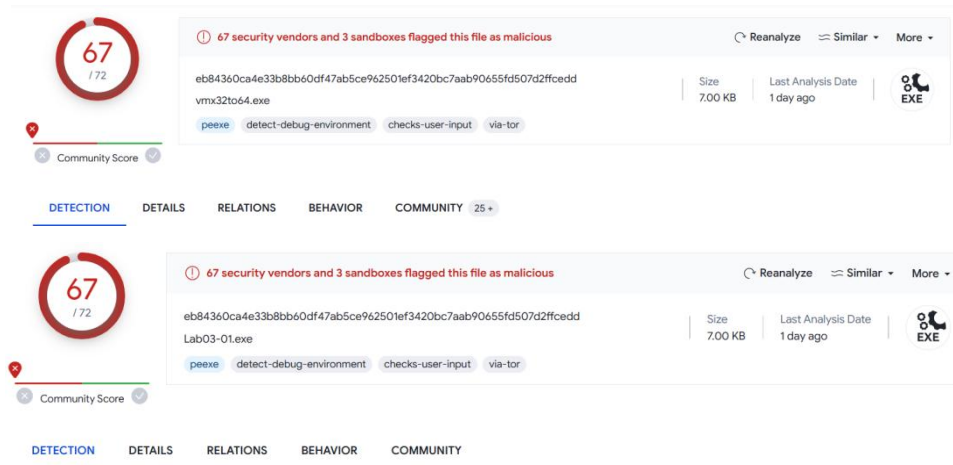
Process Monitor - Sysinternals: www.sysinternals.com						
File Edit Event Filter Tools Options Help						
Time	Process Name	PID	Operation	Path	Result	Detail
21:1...	Lab03-01.exe	2364	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: RE
21:1...	Lab03-01.exe	2364	WriteFile	C:\WINDOWS\system32\vmx32to64.exe	SUCCESS	Offset:
21:1...	Lab03-01.exe	2364	RegSetValue	HKLM\SOFTWARE\Microsoft\Window...	SUCCESS	Type: RE
21:1...	Lab03-01.exe	2364	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: RE
21:1...	Lab03-01.exe	2364	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: RE
21:1...	Lab03-01.exe	2364	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: RE
21:1...	Lab03-01.exe	2364	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: RE
21:1...	Lab03-01.exe	2364	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: RE
21:1...	Lab03-01.exe	2364	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: RE
21:1...	Lab03-01.exe	2364	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: RE

这些记录中有一定数量的噪声。比如 LM\SOFTWARE\Microsoft\Cryptography\RNG\Seed 键值上的 RegSetValue 操作是典型的噪声，因为随机数发生器的种子会有软件在注册表中不停地更新。

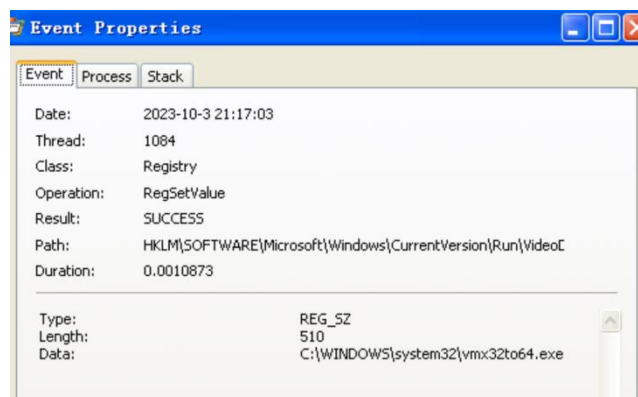
查看 WriteFile 操作记录：



记录显示，恶意代码往 C: WINDOWS\System32\vmx32to64.exe 中写了 7168 字节。这恰好是 Lab03-01.exe 文件的大小。比较新创建的 vmx32to64.exe 和 Lab03-01.exe，可以看到二者具有相同的 MD5 哈希值，这说明恶意代码已经复制本身到这个文件系统位置上。这是一个非常有用的感染主机迹象特征，因为它使用了一个硬编码的文件名。



查看另一条目，创建了一个新注册表项。

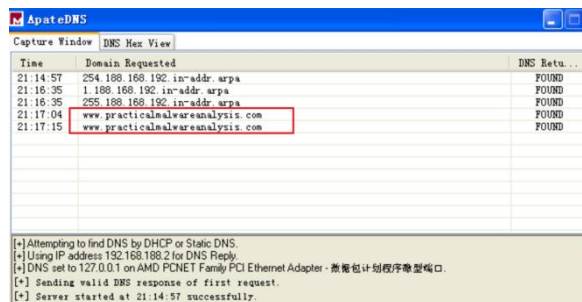


这个新创建的注册表项在 HKLM\HKLMI\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 位置，名为 VideoDriver，用于在系统启动时自动运行 vmx32to64.exe。

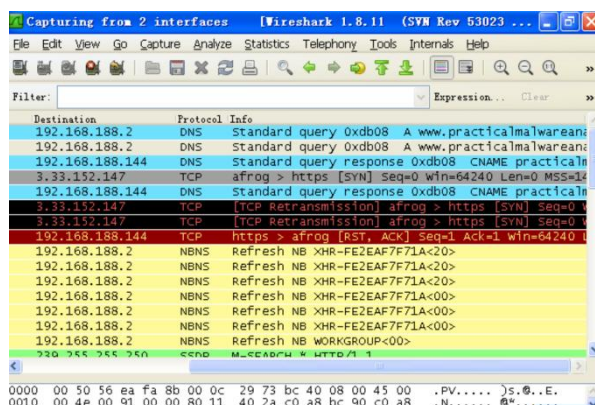
结合以上分析，我们知道，该恶意代码创建了一个名为 WinVMX32 的互斥量，并复制自身到 C:\Windows\System32\vmx32to64.exe；通过创建注册表键值 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\VideoDriver，在系统启动时自动运行 vmx32to64.exe。

3.1.3 这个恶意代码是否存在一些有用的网络特征码？如果存在，它们是什么？

首先，检查 ApateDNS 查看恶意代码是否执行了 DNS 请求，可以看到有一个 www.practicalmalwareanalysis.com 域名的请求。



使用 wireshark 进行抓包分析：



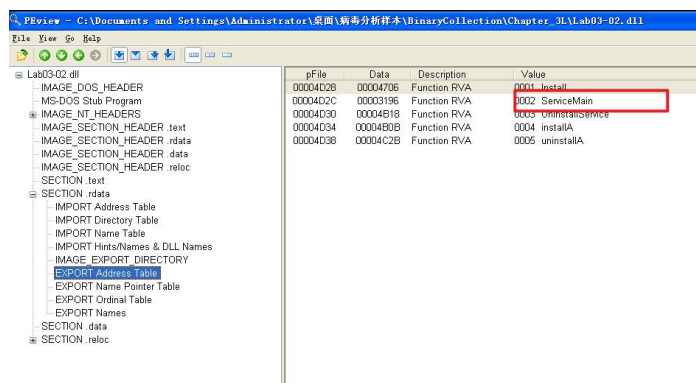
说明恶意代码在进行 www.practicalmalwareanalysis.com 的域名解析后，持续地广播大小为 256 字节的数据包，其中包含看似随机的二进制数据。

Lab 3-2 使用动态分析基础技术来分析在 Lab03-02.dll 文件中发现的恶意代码。

实验过程：

首先进行静态分析，先查看 PE 文件结构。

查看文件的导出函数，导出函数 ServiceMain 表明，恶意代码需要安装一个服务，使其能够正常运行。



接下来查看导入函数，包括一些服务操作函数，比如 CreateService；注册表操作函数，比如 RegSetValueEx；网络操作函数，比如 HttpSendRequest，表明恶意代码使用了 HTTP。

pFile	Data	Description	Value	pFile	Data	Description	Value
00004400	0000569C	HintName RVA	0141 OpenServiceA	00004400	0000569A	HintName RVA	025E Init
00004404	0000567C	HintName RVA	0078 DeleteService	00004404	0000569D	HintName RVA	0000 ?????_info@_UAE@XZ
00004408	0000567B	HintName RVA	0172 RegOpenKeyExA	00004404	00005697	HintName RVA	00CA _except_handler3
0000440C	00005658	HintName RVA	017B RegQueryValueExA	00004408	00005695	HintName RVA	0241 _CxxThrowException
00004410	00005644	HintName RVA	015B RegOpenKeyA	0000440C	0000568E	HintName RVA	01C1 _ehrfp
00004414	00005638	HintName RVA	0145 RegOpenKeyExA	00004408	00005684	HintName RVA	0142 _Init_protog
00004418	00005626	HintName RVA	004C CreateServiceA	00004404	0000568A	HintName RVA	0049 _CxxFrameHandler
0000441C	00005610	HintName RVA	0034 CloseServiceHandle	00004408	00005683	HintName RVA	02B7 _strchr
00004420	0000560D	HintName RVA	015E RegCreateKeyExA	0000440C	00005682	HintName RVA	013A _foo
00004424	000055EE	HintName RVA	0186 RegSetValueExA	0000440C	00005681	HintName RVA	025C _strstr
00004428	000055D0	HintName RVA	018E RegOpenServiceA	0000440C	00005680	HintName RVA	02B6 _strcat
0000443C	0000559C	HintName RVA	014E SetServiceStatus	0000440C	00005680	HintName RVA	025E _scanf
00004438	00000000	End of Imports	ADVAPI32.dll	00004400	0000567F	HintName RVA	025C _atoi
00004434	00005548	HintName RVA	0150 GetStartupInfoA	00004404	0000567C	HintName RVA	0259 _memset
00004438	0000555A	HintName RVA	0043 CreatePipe	0000440C	00005678	HintName RVA	005A _strchr_s
0000443C	00005569	HintName RVA	00F5 GetCurrentDirectoryA	00004408	0000567C	HintName RVA	02C1 _strncpy
00004440	00005536	HintName RVA	0044 CreateProcessA	00004404	0000567C	HintName RVA	02B6 _strcat
00004444	0000559D	HintName RVA	0308 lstrlenA	00004408	00005676	HintName RVA	026A _strcpy
00004448	0000559C	HintName RVA	0271 SetLastError	0000440C	00005674	HintName RVA	025E _atoi
00004450	00005540	HintName RVA	01F5 OutputDebugStringA	00004404	0000567D	HintName RVA	024F _flush
00004450	0000552B	HintName RVA	0108 CloseHandle	00004408	00005675	HintName RVA	025E _atoi
00004454	0000551C	HintName RVA	0218 ReadFile	0000440C	0000567C	HintName RVA	0266 _fopen
00004458	0000550C	HintName RVA	0165 GetTempPathA	00004500	00005676	HintName RVA	025E _write
00004460	00005478	HintName RVA	0121 GetLongPathNameA	00004504	0000567C	HintName RVA	02C3 _strchr
0000446C	0000546B	HintName RVA	01C2 LoadLibraryA	00004508	00000000	End of Imports	MSVCRT.dll
00004470	000054D5	HintName RVA	013E GetProcAddress	0000450C	00005674	HintName RVA	0056 InternetCloseHandle
00004474	0000545D	HintName RVA	004A CreateThread	00004510	0000567A	HintName RVA	006F InternetOpenA
00004478	00005486	HintName RVA	0150 GetSystemTime	00004514	00005678	HintName RVA	0054 InternetConnectA
00004480	00005440	HintName RVA	022E WaitForSingleObject	00004518	00005672	HintName RVA	0045 HttpOpenRequestA
00004474	0000548E	HintName RVA	029F TerminateThread	0000451C	0000567E	HintName RVA	0049 HttpOpenRequestA
00004478	00005486	HintName RVA	0236 Sleep	00004520	0000566C	HintName RVA	0054 HttpAddRequestHeadersA
0000447C	00005480	HintName RVA	011A GetLastErrorMessage	00004524	00005608	HintName RVA	0077 InternetSetFileTime
00004480	00005570	HintName RVA	0124 GetModuleFileNameA	00004528	00000000	End of Imports	WINNINET.dll
				0000452C	00000000	End of Imports	USER32.dll
				00004530	0000568E	HintName RVA	006D WSASocketA

```

C_CxxFrameHandler2
_EH_prolog
_CxxThrowException
except_handler3
__XCRT_dll
??type_info@QUAER8Z
Free
__initterm
__wllnc
__adjust_fdiv
__strnicmp
__chdir
__stricmp
lab98-02.dll
__install
ServiceMain
__uninstallService
__installA
__uninstallA
V29ubnd.dll==
serviceinstallwareanalysis.com
crve.html
005000000000
2xiLZKA=
V21h
X0pda==
m/*
Windows XP 6.11
CreateProcessA
kernel32.dll
.exe
GEI
HTTP/1.1
%* %*
1234567890123456
quit
exec
getfile
end.exe /c
ABCD EFGH I JKL MNOPQRST UVWXYZabc defghijklmnopqrstuvwxy z0123456789 +/
{ }
( )
.P8K
.P0D
DependOnService
hpcss
ServiceDll
GetModuleFileName() get dll path
Parameters

```

安装前先使用 regshot 拍摄快照，然后使用 rundll32.exe 安装恶意代码：

安装后再拍摄快照，进行比较：

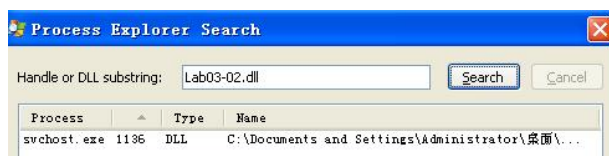
在 Keys added 中，显示了恶意代码将自身安装为 IPRIP 服务，由于这个恶意代码是个 DLL 文件，它依赖于一个可执行文件来执行它。

在下面可以看到 ImagePath 被设置为 svchost.exe，这意味着这个恶意代码将会在一个 svchost.exe 进程中启动。其余的信息，比如 DisplayName 和 Description。可以作为识别这个恶意服务的独特指纹特征。

由于恶意代码安装为 IPRIP 服务，启动 IPRIP 服务，运行恶意代码。

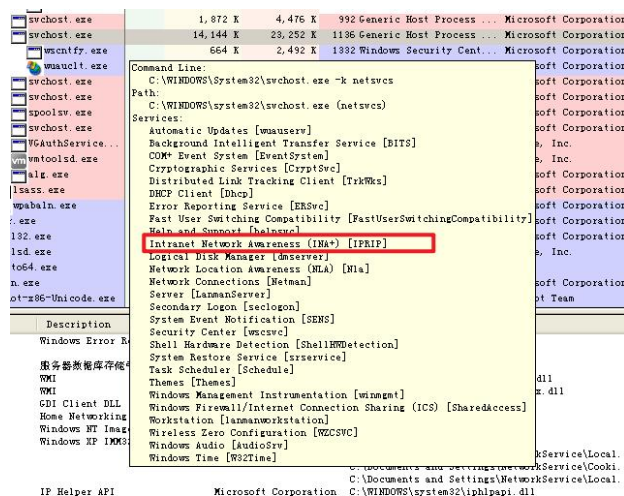
```
C:\Documents and Settings\Administrator>net start IPRIP
Intranet Network Awareness (INA+) 服务正在启动。
Intranet Network Awareness (INA+) 服务已经启动成功。
```

打开 Process Explorer，使用 FIND DLL 功能寻找恶意代码运行的进程。

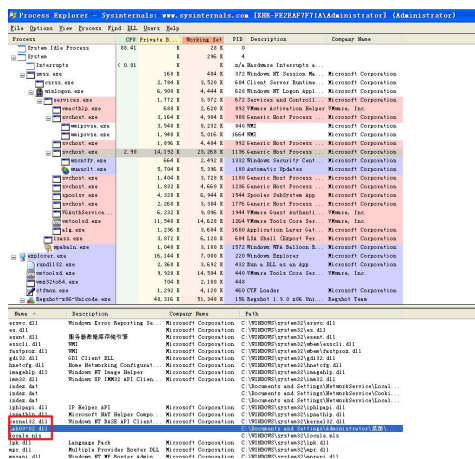


这说明，Lab03-02.dll 是由 PID 为 1136 的 svchost.exe 进程加载的。

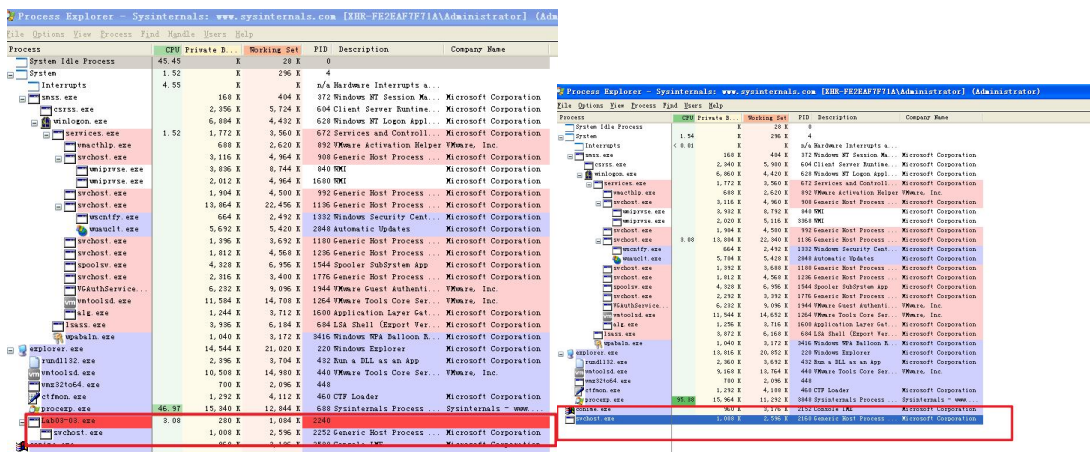
具体找到该进程，在该进程的服务中可以看到 IPRIP，证实了恶意代码在 svchost.exe 进程中运行。



同时查看 dlls，可以看到 Lab03--02.dll 被装载。

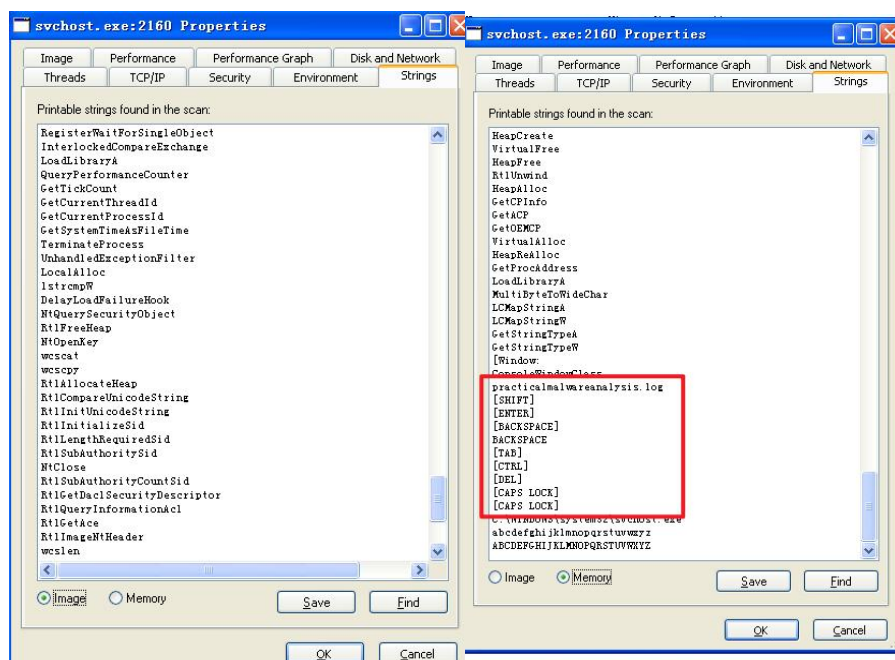


运行恶意代码文件，在 Process Explorer 中可以看到 Lab03-03.exe，它还创建了子进程 svchost.exe，创建之后它就退出了。Svcchost.exe 进程继续作为一个孤儿进程执行。



这个进程看起来像是一个合法 svchost.exe 进程，但这个 svchost.exe 是很可疑的，因为 svchast.exe 通常是 services.exe 的子进程。

选择该进程，右击选择 Properties，选择 Strings 显示在磁盘镜像中和内存镜像中可执行文件的字符串列表。内存镜像中的字符串列表里包含了 practicalmalwareanalysis.log 和 [ENTER]，而它们都不会在磁盘镜像中一个典型的 svchost.exe 文件中出现。

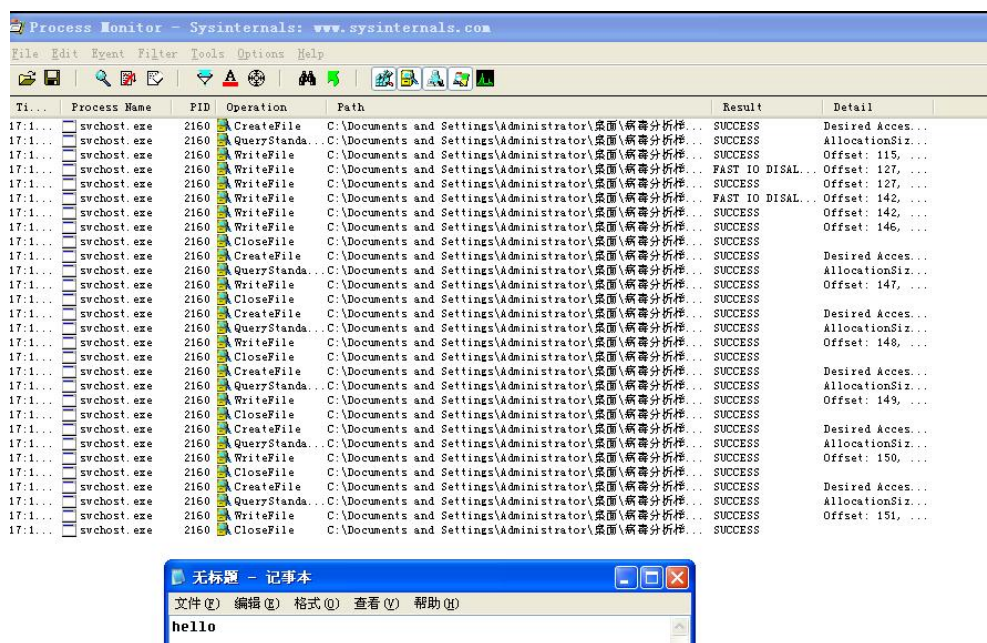


practicalmalwareanalysis.log 字符串的存在，再加上出现了 [ENTER] 和 [CAPS LOCK] 这样的字符串，表明这个程序很可能是一个击键记录器。

接下来验证这一点。首先，使用 svchost.exe 的 PID 在 procmon 工具中创建一个过滤器。然后，打开记事本程序，键入信息。

可以发现，svchost.exe 的 CreateFile 和 WriteFile 事件正在写一个名为

practicalmalwareanalysis.log 的文件。



打开日志文件，可以发现刚刚的击键记录被记录：



3.3.1 当你使用 Process Explorer 工具进行监视时，你注意到了什么？

Lab03-01.exe 运行和，创建了 svchost.exe 文件，然后退出，scvhost.exe 进程继续作为一个孤儿进程执行。它执行了对 svchost.exe 文件的替换。

3.3.2 你可以找出任何的内存修改行为吗？

在前面的实验过程中，对比了内存映像与磁盘映像中的 svwchost.exe，显示它们并不是一样的。内存映像拥有如 practicalmalwareanalysis.log 和[ENTER]这样的字符串，而磁盘镜像中却没有，这应该是恶意代码所修改的。

3.3.3 这个恶意代码在主机上的感染迹象特征是什么？

恶意代码创建了 practicalmalwareanalysis.log 文件，并将击键记录在该日志文件中。这可以作为感染的迹象特征。

3.3.4 这个恶意代码的目的是什么？

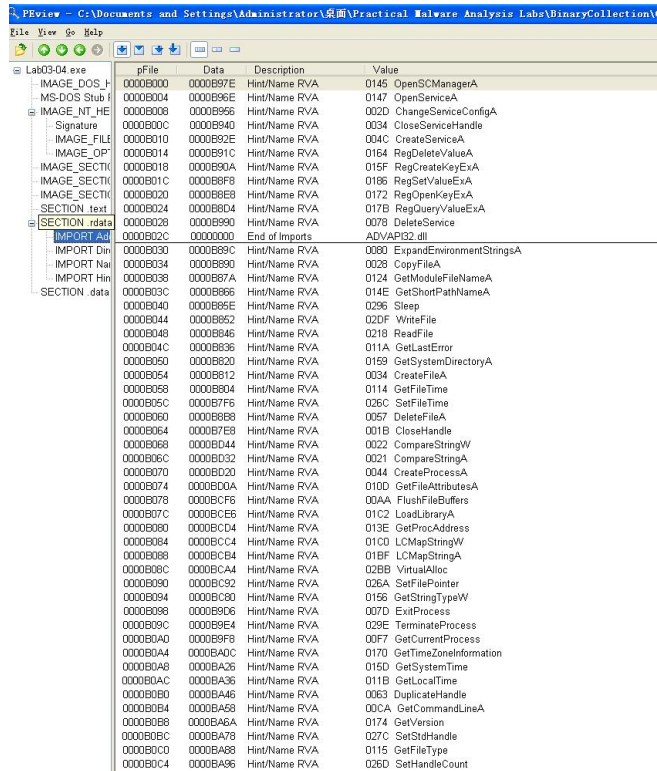
这个程序在 svchost.exe 进程上执行了进程替换，来启动一个击键记录器，将击键记录在

创建的日志文件中。

Lab 3-4 使用基础的动态行为分析工具来分析在 Lab03-04.exe 文件中发现的恶意代码。

实验过程：

首先查看 PE 文件结构，查看导入表，可以发现导入了一些服务操作函数、注册表操作函数、联网功能函数等。



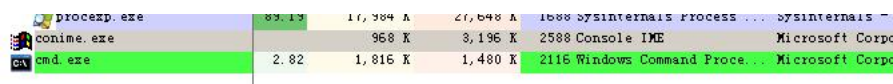
File	Data	Description	Value
Lab03-04.exe			
IMAGE_DOS_HEADER	0000B000	Hint/Name RVA	0145 OpenSCManagerA
MS-DOS Stub	0000B004	Hint/Name RVA	0147 OpenServiceA
IMAGE_NT_HEADERS_V	0000B008	Hint/Name RVA	0020 ChangeServiceConfigA
Signature	0000B00C	Hint/Name RVA	0034 CloseServiceHandle
IMAGE_FILE_HEADER	0000B010	Hint/Name RVA	004C CreateServiceA
IMAGE_OPTIONAL_HEADER	0000B014	Hint/Name RVA	0164 RegDeleteValueA
IMAGE_SECTION_HEADER	0000B018	Hint/Name RVA	015F RegCreateKeyExA
IMAGE_SECTION_HEADER	0000B01C	Hint/Name RVA	0186 RegSetValueExA
IMAGE_SECTION_HEADER	0000B020	Hint/Name RVA	0173 RegOpenKeyExA
SECTION .text	0000B024	Hint/Name RVA	017B RegQueryValueExA
SECTION .data	0000B028	Hint/Name RVA	0078 DeleteService
IMPORT .data	0000B02C	End of Imports	ADVAPI32.dll
IMPORT .data	0000B030	Hint/Name RVA	0080 ExpandEnvironmentStringsA
IMPORT .data	0000B034	Hint/Name RVA	0020 CopyFileA
IMPORT .data	0000B038	Hint/Name RVA	0124 GetModuleFileNameA
IMPORT .data	0000B03C	Hint/Name RVA	014E GetShortPathNameA
IMPORT .data	0000B040	Hint/Name RVA	0296 Sleep
IMPORT .data	0000B044	Hint/Name RVA	02DF WriteFile
IMPORT .data	0000B048	Hint/Name RVA	0218 ReadFile
IMPORT .data	0000B04C	Hint/Name RVA	011A GetLastError
IMPORT .data	0000B050	Hint/Name RVA	0159 GetSystemDirectoryA
IMPORT .data	0000B054	Hint/Name RVA	0034 CreateFileA
IMPORT .data	0000B058	Hint/Name RVA	0114 GetFileTime
IMPORT .data	0000B05C	Hint/Name RVA	026C SetFileTime
IMPORT .data	0000B060	Hint/Name RVA	0057 DeleteFileA
IMPORT .data	0000B064	Hint/Name RVA	001B CloseHandle
IMPORT .data	0000B068	Hint/Name RVA	0022 CompareStringW
IMPORT .data	0000B06C	Hint/Name RVA	0021 CompareStringA
IMPORT .data	0000B070	Hint/Name RVA	0044 CreateProcessA
IMPORT .data	0000B074	Hint/Name RVA	010D GetFileAttributesA
IMPORT .data	0000B078	Hint/Name RVA	004A FlushFileBuffers
IMPORT .data	0000B07C	Hint/Name RVA	01C2 LoadLibraryA
IMPORT .data	0000B080	Hint/Name RVA	013E GetProcAddress
IMPORT .data	0000B084	Hint/Name RVA	01C0 LCMAPStringW
IMPORT .data	0000B088	Hint/Name RVA	01BF LCMAPStringA
IMPORT .data	0000B08C	Hint/Name RVA	02BB VirtualAlloc
IMPORT .data	0000B090	Hint/Name RVA	026A SetFilePointer
IMPORT .data	0000B094	Hint/Name RVA	0156 GetStringTypeW
IMPORT .data	0000B098	Hint/Name RVA	007D ExitProcess
IMPORT .data	0000B09C	Hint/Name RVA	029E TerminateProcess
IMPORT .data	0000B0A0	Hint/Name RVA	00F7 GetCurrentProcess
IMPORT .data	0000B0A4	Hint/Name RVA	0170 GetTimeZoneInformation
IMPORT .data	0000B0A8	Hint/Name RVA	015D GetSystemTime
IMPORT .data	0000B0AC	Hint/Name RVA	011B GetLocalTime
IMPORT .data	0000B0B0	Hint/Name RVA	0063 DuplicateHandle
IMPORT .data	0000B0B4	Hint/Name RVA	00CA GetCommandLineA
IMPORT .data	0000B0B8	Hint/Name RVA	0174 GetVersion
IMPORT .data	0000B0BC	Hint/Name RVA	027C SetStdHandle
IMPORT .data	0000B0C0	Hint/Name RVA	0115 GetFileType
IMPORT .data	0000B0C4	Hint/Name RVA	026D SetHandleCount

查看字符串，看到域名、注册表位置、像 DOWNLOAD、UPLOAD 这样的命令字符串，以及 HTTP/1.0 字符串等。这些表明恶意代码可能是一个 HTTP 后门程序。字符串-cc、-re、-in 应该是一些命令行参数(例如-in 可能是 install 的缩写)。



```
CompareStringA
CompareStringW
SetEnvironmentVariableA
IIZE
Configuration
SOFTWARE\Microsoft\XPS
kernel32.dll
HTTP/1.0
GET
NOTHING
CMD
DOWNLOAD
UPLOAD
SLEEP
cmd.exe
>> NUL
/c del
ups
http://www.practicalmalwareanalysis.com
Manager Service
.exe
%SYSTEMROOT%\system32\
k:%s h:%s p:%s per:%s
```

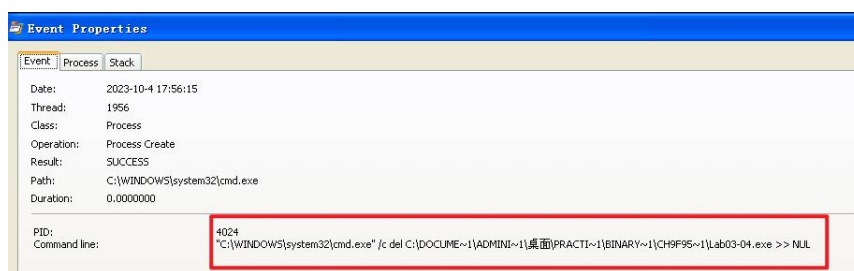
启动 Process Explorer，运行恶意代码。可以发现，快速运行了 cmd.exe，然后自行退出。



Process Name	Private Bytes	Working Set	Session ID	Company Name
Lab03-04.exe	83,132 K	1,784 K	2,148	Microsoft Corp...
cmd.exe	968 K	3,196 K	2588	Microsoft Corp...
cmd.exe	2,820 K	1,816 K	1,480	Microsoft Corp...

同时发现，运行恶意代码后，恶意代码删除了自身。

接下来使用 procmon 工具，设置进程名称为 Lab03-04.exe 的过滤器，在过滤的信息中，有一个 ProcessCreate 的条目：



可以发现，恶意代码通过运行 cmd.exe，写入命令将自己删除。

3.4.1 当你运行这个文件时，会发生什么呢？

启动 Process Explorer，运行恶意代码。可以发现，快速运行了 cmd.exe，然后自行退出。



同时结合 procmon 中过滤的信息，可以发现，运行恶意代码后，恶意代码运行了 cmd.exe，写入命令删除了自身。

3.4.2 是什么原因造成动态分析无法有效实施？

有可能需要提供一个命令行参数，或者是这个程序缺失某个部件。

3.4.3 是否有其他方式来运行这个程序？

尝试使用命令行运行恶意代码，并使用在字符串列表中发现的一些命令行参数(-in、-re、-cc)，但都以失败告终，结果程序还是会删除自身。



Lab3-5 编写 Yara 规则

结合前面的分析，编写 yara 规则如下：

```
rule Lab03_01 {
  meta:
    description = "Lab03-01.exe"
  strings:
    $s1 = "vmx32to64.exe" fullword ascii
    $s2 = "SOFTWARE\\Classes\\http\\shell\\open\\commandV" fullword ascii
    $s3 = " www.practicalmalwareanalysis.com" fullword ascii
    $s4 = "advpack" fullword ascii
    $s5 = "VideoDriver" fullword ascii
    $s6 = "WinVMX32-" fullword ascii
    $s7 = "Software\\Microsoft\\Active Setup\\Installed Components\\" fullword ascii
```

```

condition:
    uint16(0) == 0x5a4d and
    uint32(uint32(0x3c))==0x00004550 and filesize < 20KB and
    5 of them
}
rule Lab03_02 {
    meta:
        description = "Lab03-02.dll"
    strings:
        $x1 = "%SystemRoot%\System32\svchost.exe -k " fullword ascii
        $s3 = "RegOpenKeyEx(%s) KEY_QUERY_VALUE error ." fullword ascii
        $s4 = "practicalmalwareanalysis.com" fullword ascii
        $s5 = "Lab03-02.dll" fullword ascii
        $s6 = "RegOpenKeyEx(%s) KEY_QUERY_VALUE success." fullword ascii
        $s7 = "serve.html" fullword ascii
        $s8 = "GetModuleFileName() get dll path" fullword ascii
        $s9 = "netsvcs" fullword ascii
        $s10 = "OpenService(%s) error 2" fullword ascii
        $s11 = "OpenService(%s) error 1" fullword ascii
        $s12 = "CreateService(%s) error %d" fullword ascii
        $s13 = "You specify service name not in Svchost/netsvcs, must be one of following:" fullword
ascii
        $s14 = "RegQueryValueEx" fullword ascii
        $s15 = "Depends INA+" fullword nocase
    condition:
        uint16(0) == 0x5a4d and
        uint32(uint32(0x3c))==0x00004550 and filesize < 70KB and
        all of ($x*) and 6 of them
}
rule Lab03_03 {
    meta:
        description = "Lab03-03.exe"
    strings:
        $s1 = "\\svchost.exe" fullword ascii
        $s2 = "+A+A+A+A" fullword ascii
    condition:
        uint16(0) == 0x5a4d and
        uint32(uint32(0x3c))==0x00004550 and filesize < 200KB and
        all of them
}

```

```

}
rule Lab03_04 {
  meta:
    description = "Lab03-04.exe"
  strings:
    $s1 = "http://www.practicalmalwareanalysis.com" fullword ascii
    $s2 = "%SYSTEMROOT%\system32\\" fullword ascii
    $s3 = " HTTP/1.0" fullword ascii
    $s4 = " Manager Service" fullword ascii
    $s5 = "UPLOAD" fullword ascii
    $s6 = "DOWNLOAD" fullword ascii
    $s7 = "command.com" fullword ascii
    $s8 = "COMSPEC" fullword ascii
    $s9 = "SOFTWARE\Microsoft\XPS" fullword ascii
    $s10 = "/c del " fullword ascii
    $s11 = ">> NUL" fullword ascii
  condition:
    uint16(0) == 0x5a4d and
    uint32(uint32(0x3c)) == 0x00004550 and filesize < 200KB and
    8 of them
}

```

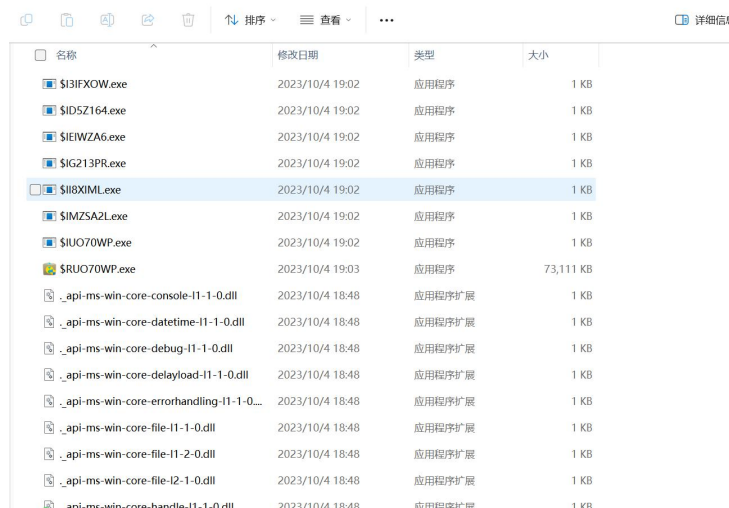
运行 yara 规则，能够扫描到对应的恶意代码文件：

```

C:\Documents and Settings\Administrator\桌面\yara-v2.0.0-win32>yara32 lab03.yar
Chapter_3L
Lab03_01 Chapter_3L\Lab03-01.exe
Lab03_02 Chapter_3L\Lab03-02.dll
Lab03_03 Chapter_3L\Lab03-03.exe
Lab03_04 Chapter_3L\Lab03-04.exe

```

利用 scan.py 程序，自动收集电脑上的所有 PE 结构文件，文件保存到 sample 文件夹中。



名称	修改日期	类型	大小
\$I3FXOW.exe	2023/10/4 19:02	应用程序	1 KB
\$ID5Z164.exe	2023/10/4 19:02	应用程序	1 KB
\$IEIWZA6.exe	2023/10/4 19:02	应用程序	1 KB
\$IG213PR.exe	2023/10/4 19:02	应用程序	1 KB
\$I8XIML.exe	2023/10/4 19:02	应用程序	1 KB
\$IMZSA2L.exe	2023/10/4 19:02	应用程序	1 KB
\$IUO70WP.exe	2023/10/4 19:02	应用程序	1 KB
\$RUO70WP.exe	2023/10/4 19:03	应用程序	73,111 KB
_api-ms-win-core-console-l1-1-0.dll	2023/10/4 18:48	应用程序扩展	1 KB
_api-ms-win-core-datetime-l1-1-0.dll	2023/10/4 18:48	应用程序扩展	1 KB
_api-ms-win-core-debug-l1-1-0.dll	2023/10/4 18:48	应用程序扩展	1 KB
_api-ms-win-core-delayload-l1-1-0.dll	2023/10/4 18:48	应用程序扩展	1 KB
_api-ms-win-core-errorhandling-l1-1-0.dll	2023/10/4 18:48	应用程序扩展	1 KB
_api-ms-win-core-file-l1-1-0.dll	2023/10/4 18:48	应用程序扩展	1 KB
_api-ms-win-core-file-l1-2-0.dll	2023/10/4 18:48	应用程序扩展	1 KB
_api-ms-win-core-file-l2-1-0.dll	2023/10/4 18:48	应用程序扩展	1 KB
_api-ms-win-core-handle-l1-1-0.dll	2023/10/4 18:48	应用程序扩展	1 KB

文件信息:

	sample
类型:	文件夹
位置:	D:\NKU\code\Python
大小:	17.1 GB (18,457,236,441 字节)
占用空间:	17.2 GB (18,484,576,256 字节)
包含:	14,589 个文件, 0 个文件夹
创建时间:	2023年10月4日, 18:39:02

编写 c++程序, 对 sample 文件夹进行扫描, 并得到扫描时间。

```
#include <iostream>
#include <windows.h>
#include <string>
using namespace std;

string cmdPopen(const string& cmdLine) {
    char buffer[1024] = { '\0' };
    FILE* pf = NULL;
    pf = _popen(cmdLine.c_str(), "r");
    if (NULL == pf) {
        printf("Open pipe failed\n");
        return string("");
    }
    string ret;
    while (fgets(buffer, sizeof(buffer), pf)) {
        ret += buffer;
    }
    _pclose(pf);
    return ret;
}

int main() {
    // 设置工作目录
    wstring workingDir = L"D:\\NKU\\23Fall\\yara-master-1798-win64";
    if (!SetCurrentDirectory(workingDir.c_str())) {
        cout << "Failed to set the working directory" << endl;
        return 1;
    }

    long long start, end, freq;
    string cmdLine = ".\\yara64 -r lab03.yar D:\\NKU\\code\\Python\\sample"; // 执行的指令
```

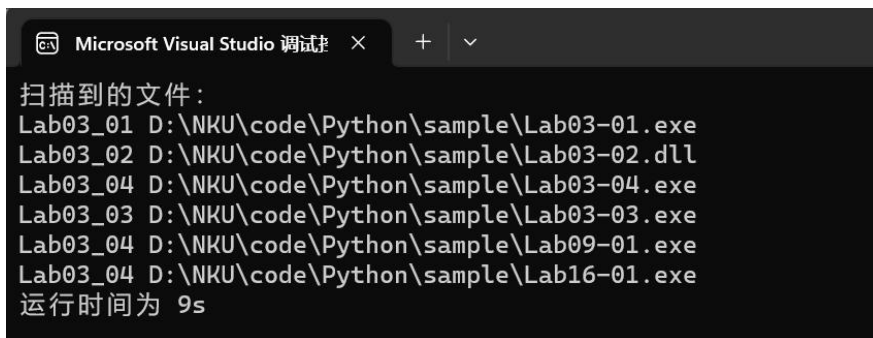


```

QueryPerformanceFrequency((LARGE_INTEGER*)&freq);
QueryPerformanceCounter((LARGE_INTEGER*)&start);
string res = cmdPopen(cmdLine);
QueryPerformanceCounter((LARGE_INTEGER*)&end);
cout << "扫描到的文件: " << endl;
cout << res; // 输出 cmd 指令的返回值
cout << "运行时间为 " << (end - start) / freq << "s" << endl;
return 0;
}

```

运行结果如下：



```

Microsoft Visual Studio 调试  x  +  v
扫描到的文件:
Lab03_01 D:\NKU\code\Python\sample\Lab03-01.exe
Lab03_02 D:\NKU\code\Python\sample\Lab03-02.dll
Lab03_04 D:\NKU\code\Python\sample\Lab03-04.exe
Lab03_03 D:\NKU\code\Python\sample\Lab03-03.exe
Lab03_04 D:\NKU\code\Python\sample\Lab09-01.exe
Lab03_04 D:\NKU\code\Python\sample\Lab16-01.exe
运行时间为 9s

```

四、实验结论及心得体会

本次实验综合使用静态分析技术和动态分析基础技术分析了恶意代码，掌握了基本动态分析技术。