



南開大學
Nankai University

网络空间安全学院
《恶意代码分析与防治技术》课程实验报告

实验三：基本动态分析技术

姓名：王峥

学号：2211267

专业：信息安全

指导教师：王志、邓琮弋

2024 年 10 月 10 日

目录

1 实验目的	2
2 实验原理	2
2.1 Lab3-1	2
2.2 Lab3-2	2
2.3 Lab3-3	2
2.4 Lab3-4	2
2.5 YARA 规则	3
3 实验过程	3
3.1 Lab3-1	3
3.2 Lab3-2	8
3.3 Lab3-3	13
3.4 Lab3-4	16
3.5 VirusTotal 查看实验样本	17
3.6 YARA 检测	20
4 实验结论及心得体会	24
4.1 实验结论	24
4.2 心得体会	24

1 实验目的

- 实验首先将复习理论课上动态分析的基础技术的基础知识。
- 使用动态分析基础技术来分析实验样例，发现在不同文件中的恶意代码，以便理解其行为、感染迹象、运行方式和网络特征码。

2 实验原理

2.1 Lab3-1

- 导入函数与字符串列表：通过动态分析，可以监视 Lab03-01.exe 文件的运行并识别其调用的导入函数和使用的字符串列表。
- 感染迹象特征：观察主机上的异常行为、文件变化或注册表修改等迹象，以确定感染。
- 网络特征码：检查网络通信，如域名、IP 地址或传输协议，以了解恶意代码是否与特定网络活动相关。

2.2 Lab3-2

- 自行安装：监视恶意代码的安装过程，以了解其自动安装的方式。
- 运行恶意代码：观察其运行方式，并识别启动机制。
- 进程识别：使用 Process Explorer 查找与恶意代码相关的运行进程。
- 过滤器设置：在 ProcMon 中设置过滤器，专注于收集与恶意代码相关的信息。
- 感染迹象特征：寻找主机上的异常行为和文件修改等迹象。
- 网络特征码：检查网络通信，识别与恶意代码相关的特定流量。

2.3 Lab3-3

- Process Explorer 监视：使用 Process Explorer 工具监控恶意代码的行为，关注异常进程或资源使用情况。
- 内存修改：检查内存中是否存在恶意代码对进程或系统内存的修改。
- 感染迹象特征：观察主机的异常行为和资源占用等特征。
- 恶意代码目的：尝试理解恶意代码的意图，如数据窃取或系统破坏。

2.4 Lab3-4

- 文件运行：运行 Lab03-04.exe 文件并监视其行为。
- 动态分析障碍：理解动态分析可能受到的限制，例如反分析技术、加密等。
- 其他运行方式：探索是否有其他方式可以运行恶意代码，如反向工程分析或虚拟化环境。

2.5 YARA 规则

Yara 规则是一种特定的语法，用于描述恶意软件的特征。这些规则可以与已知样本进行匹配，以识别潜在的恶意活动。Yara 规则由一系列规则语句组成，每个规则都包含了一个或多个条件和一个或多个操作。规则的条件描述了恶意软件的特征，而操作则定义了匹配成功时要执行的动作。

3 实验过程

3.1 Lab3-1

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

(1): 在进行动态分析之前我们首先对实验样本进行简单的静态分析。根据我们实验一已有的经验：首先使用 PEView 查看 Lab03-01.exe 文件

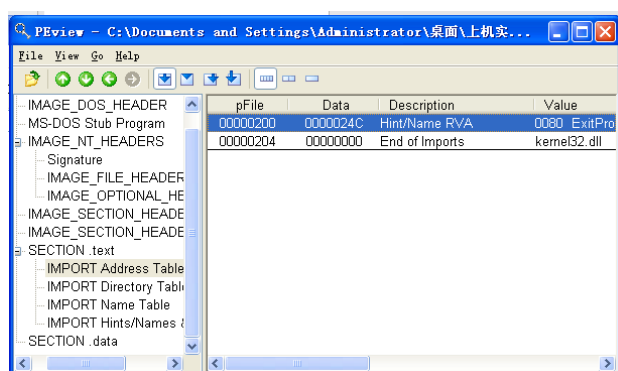


图 3.1: 使用 PEView 打开 Lab03-01.exe

上图我们查看了 SECTION.text 下的 IMPORT Address Table，可以发现这个恶意代码的导入函数内容较正常的 exe 文件少，只有一个 ExitProcess，理论上看无法使一个正常的可执行程序运行，因此我们怀疑是加壳或是混淆。

(2): 综上，我们使用 PEiD 来检查加壳信息

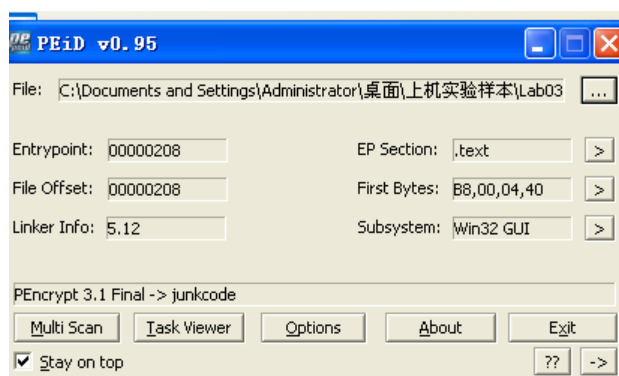


图 3.2: 使用 PEiD 查看 Lab03-01.exe

我们仔细观察壳是 **PEncript 3.1 Final -> junkcode**

(3): 由上述分析，我们知道该文件是一个加壳文件，我们可以用 Strings 尝试查找更多有价值的字符串，继而我们根据这些字符串来进行动态分析。

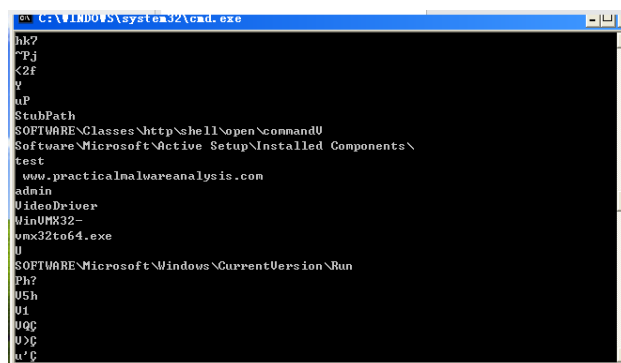


图 3.3: Lab03-01.exe 的部分字符串

可以看到我们查找到一些异常的字符串,譬如注册表路径,域名,WinVMX32、VideoDriver、vmx32to64.exe 等内容

(4): 接下来我们进行动态分析,首先我们使用虚拟机进行快照,以便试验结束后恢复。接下来我们打开 Process Explorer,选中 Lab03-01.exe

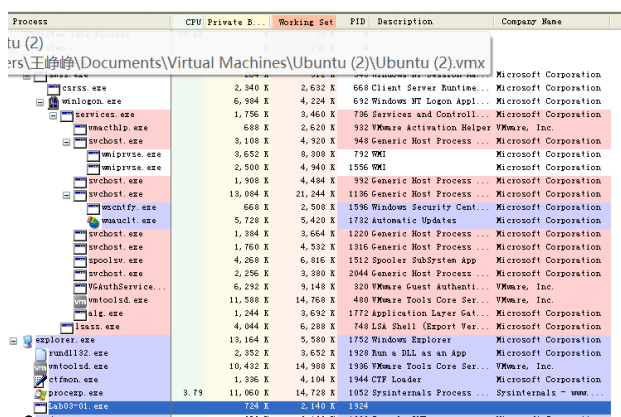


图 3.4: 使用 Process Explorer 查看

接下来我们通过 **View -> Low Pane View -> Handles** 来查看,发现创建了互斥量 WinVMX32

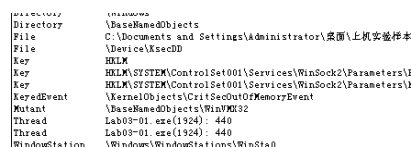


图 3.5: 查看 Handles

然后我们再点击 **View -> Low Pane View -> DLL** 查看,发现了 ws2_32.dll, wshtcpip.dll 等联网的库

advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\WINDOWS\system32\advapi32.dll
advpack.dll	ADVPACK	Microsoft Corporation	C:\WINDOWS\system32\advpack.dll
ctype.nls			C:\WINDOWS\system32\ctype.nls
dnsapi.dll	DNS Client API DLL	Microsoft Corporation	C:\WINDOWS\system32\dnsapi.dll
gdi32.dll	GDI Client DLL	Microsoft Corporation	C:\WINDOWS\system32\gdi32.dll
hnetcfg.dll	Home Networking Configuratio...	Microsoft Corporation	C:\WINDOWS\system32\hnetcfg.dll
imm32.dll	Windows XP IMM32 API Client...	Microsoft Corporation	C:\WINDOWS\system32\imm32.dll
kernel32.dll	Windows NT BASE API Client...	Microsoft Corporation	C:\WINDOWS\system32\kernel32.dll
Lab03-01.exe			C:\Documents and Settings\Administrator\桌面\...
locale.nls			C:\WINDOWS\system32\locale.nls
lpk.dll	Language Pack	Microsoft Corporation	C:\WINDOWS\system32\lpk.dll
mswrt.dll	Windows NT CRT DLL	Microsoft Corporation	C:\WINDOWS\system32\mswrt.dll
mswsock.dll	Microsoft Windows Sockets ...	Microsoft Corporation	C:\WINDOWS\system32\mswsock.dll
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\WINDOWS\system32\ntdll.dll
ole32.dll	Microsoft OLE for Window...	Microsoft Corporation	C:\WINDOWS\system32\ole32.dll
rasadhlp.dll	Remote Access AutoDial Helper	Microsoft Corporation	C:\WINDOWS\system32\rasadhlp.dll
rpcrt4.dll	Remote Procedure Call Runtime	Microsoft Corporation	C:\WINDOWS\system32\rpcrt4.dll
secur32.dll	Security Support Provider ...	Microsoft Corporation	C:\WINDOWS\system32\secur32.dll
setupapi.dll	Windows Setup API	Microsoft Corporation	C:\WINDOWS\system32\setupapi.dll
sortkey.nls			C:\WINDOWS\system32\sortkey.nls
sorttbls.nls			C:\WINDOWS\system32\sorttbls.nls
unicode.nls			C:\WINDOWS\system32\unicode.nls
user32.dll	Windows XP USER API Client...	Microsoft Corporation	C:\WINDOWS\system32\user32.dll
usp10.dll	Uniscribe Unicode script p...	Microsoft Corporation	C:\WINDOWS\system32\usp10.dll
version.dll	Version Checking and File ...	Microsoft Corporation	C:\WINDOWS\system32\version.dll
winrar.dll	LDAP Rsh Provider DLL	Microsoft Corporation	C:\WINDOWS\system32\winrar.dll
wldap32.dll	Win32 LDAP API DLL	Microsoft Corporation	C:\WINDOWS\system32\wldap32.dll
ws2_32.dll	Windows Socket 2.0 32-Bit DLL	Microsoft Corporation	C:\WINDOWS\system32\ws2_32.dll
wshelp.dll	Windows Socket 2.0 Helper ...	Microsoft Corporation	C:\WINDOWS\system32\wshelp.dll
wshhlp.dll	Windows Sockets Helper DLL	Microsoft Corporation	C:\WINDOWS\system32\wshhlp.dll
wshvcip.dll	Windows Sockets Helper DLL	Microsoft Corporation	C:\WINDOWS\system32\wshvcip.dll

图 3.6: 查看 DLL

(5): 接下来进入 Process Monitor 查看, 我们首先添加三个过滤:

- Process Name is Lab03-01.exe
- Operation is WriteFile
- Operation is RegSetValue

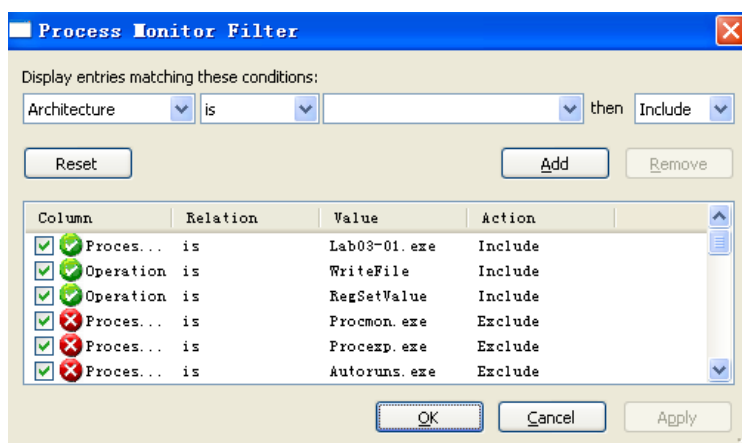


图 3.7: 部分过滤结果

然后我们查看过滤的结果, 并进行详细分析

Process Monitor - Sysinternals: www.sysinternals.com						
Time	Process Name	PID	Operation	Path	Result	Detail
11:5...	Lab03-01.exe	1260	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: REG_BIN...
11:5...	Lab03-01.exe	1260	WriteFile	C:\WINDOWS\system32\vmx32to64.exe	SUCCESS	Offset: 0, Le...
11:5...	Lab03-01.exe	1260	RegSetValue	HKLM\SOFTWARE\Microsoft\Window...	SUCCESS	Type: REG_SZ...
11:5...	Lab03-01.exe	1260	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: REG_BIN...
11:5...	Lab03-01.exe	1260	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: REG_BIN...
11:5...	Lab03-01.exe	1260	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: REG_BIN...
11:5...	Lab03-01.exe	1260	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: REG_BIN...
11:5...	Lab03-01.exe	1260	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: REG_BIN...
11:5...	Lab03-01.exe	1260	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: REG_BIN...
11:5...	Lab03-01.exe	1776	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: REG_BIN...
11:5...	Lab03-01.exe	406	RegSetValue	HKLM\SOFTWARE\Microsoft\Crypto...	SUCCESS	Type: REG_BIN...

图 3.8: 部分过滤结果

- 我们双击 WriteFile 进行查看, 我们看到其向操作系统写入了一个文件

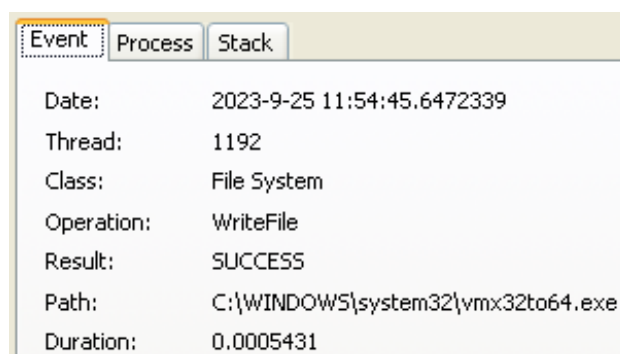


图 3.9: 查看 WriteFile 这个项

我们看到它的具体写入路径是 **C:/WINDOWS/system32/vmx32to64.exe**
我们在系统中尝试找出

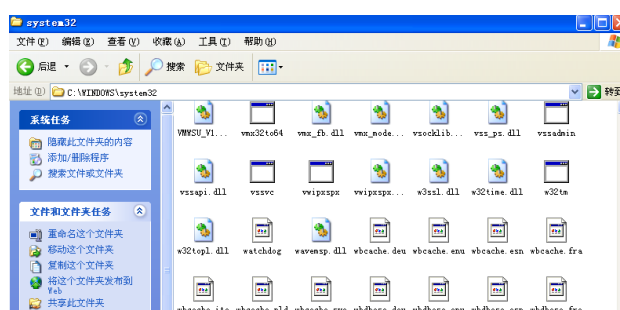


图 3.10: 在 C:/WINDOWS/system32 目录下查找

通过比较 Lab03-01.exe 和 vmx32to64.exe 的哈希值能看出二者内容相同。

- 再双击 RegSetValue 一项,我们发现该操作向注册表写入了 **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run** 这一项, 该文件将 vmx32to64.exe 写入到了开机启动的项目中。

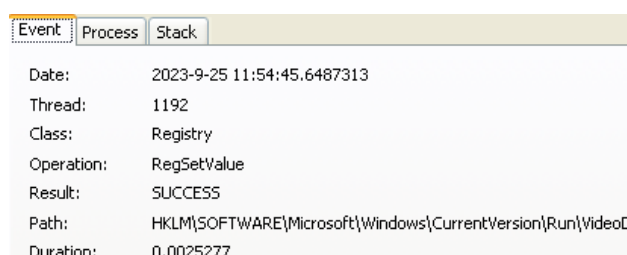


图 3.11: 查看 RegSetValue 项

我们使用 regedit 打开注册表, 根据刚才查看的结果, 我们定位查看:

名称	类型	数据
(默认)	REG_SZ	(数值未设置)
BluetoothAuth...	REG_SZ	rundll32.exe bthprops.cpl,,BluetoothAuthent...
IMJPMIG8.1	REG_SZ	"C:\WINDOWS\IME\imjp8_1\IMJPMIG.EXE" /Spoil...
MSPY2002	REG_SZ	C:\WINDOWS\system32\IME\PIINTLGH\ImScInst.e...
PHIME2002A	REG_SZ	C:\WINDOWS\system32\IME\TINTLGH\TINTSETP.E...
PHIME2002ASync	REG_SZ	C:\WINDOWS\system32\IME\TINTLGH\TINTSETP.E...
VideoDriver	REG_SZ	C:\WINDOWS\system32\vmx32to64.exe
VMware User P...	REG_SZ	"C:\Program Files\VMware\VMware Tools\vmtoo...

图 3.12: 查看注册表发现 VideoDriver 注册表项

(6): 接下来进入 ApatеDNS, 然后设置 DNS Reply IP 为 127.0.0.1, 我们检测到意代码向域名 www.practicalmalwareanalysis.com 发送了请求, 该请求每隔 61 秒便重新发送一次。

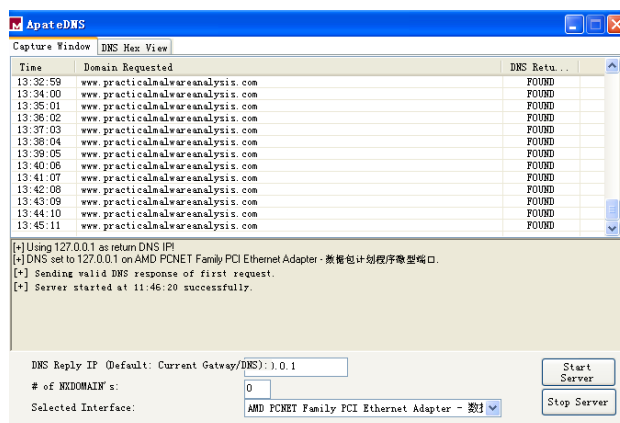


图 3.13: ApatеDNS 监测

(7): 打开 Wireshark, 看到恶意代码进行 www.practicalmalwareanalysis.com 的域名解析后, 持续地广播大小为 256 字节的数据包。

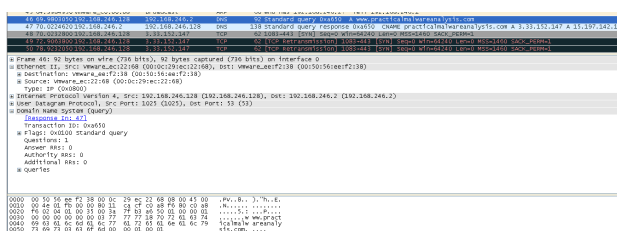


图 3.14: Wireshark 查看数据包

问题一：找出这个恶意代码的导入导出表

导入函数：

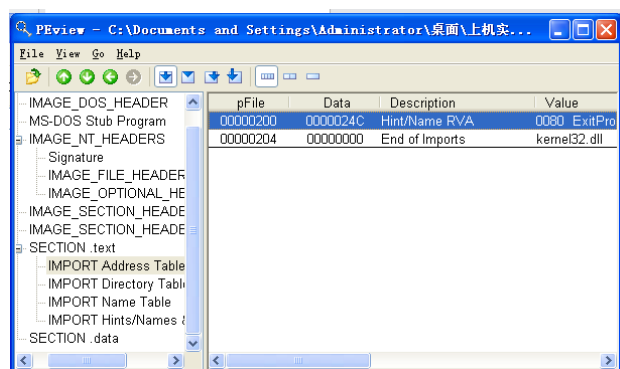


图 3.15: 使用 PEView 查看导入函数

字符串列表：

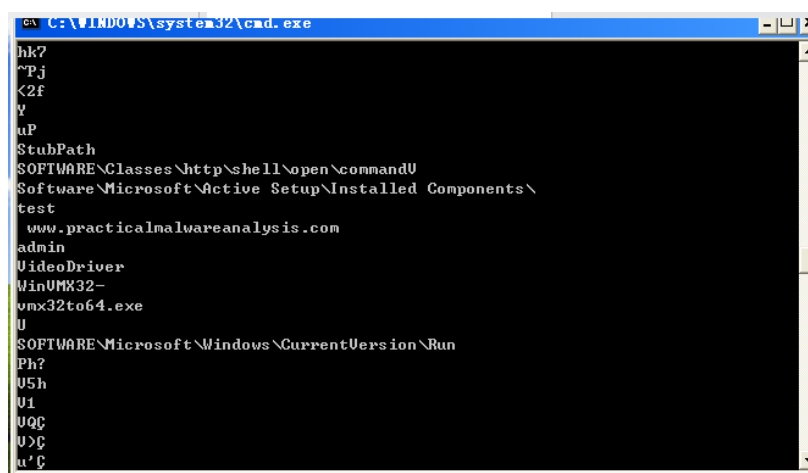


图 3.16: 使用 Strings 查看的部分字符串列表

问题二：这个恶意代码在主机上的感染迹象特征是什么？

根据上述分析,我们判断该恶意代码先创建了一个名为 WinVMX32 的互斥量,复制自身到 C:/Windows/System32 并安装自己到系统自启动项中,通过创建注册表键值 HKLM/SOFTWARE/Microsoft/Windows/CurrentVersion/Run 并将其设置为复制副本的位置。

问题三：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？

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。

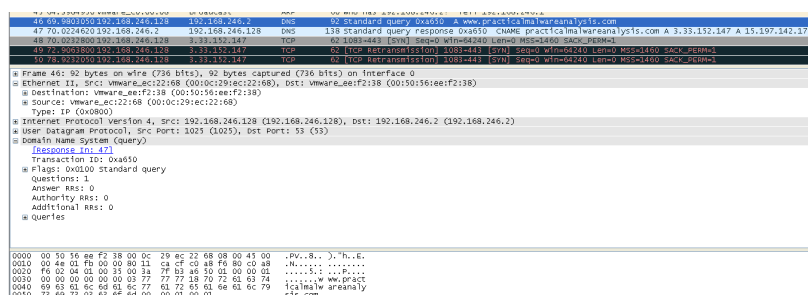
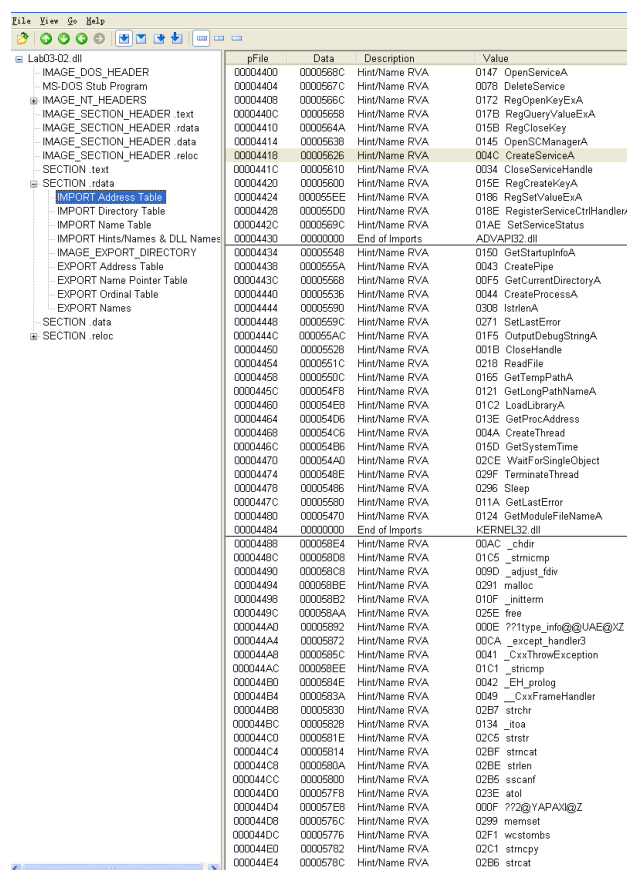


图 3.17: 利用 Wireshark 查看网络特征

3.2 Lab3-2

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。

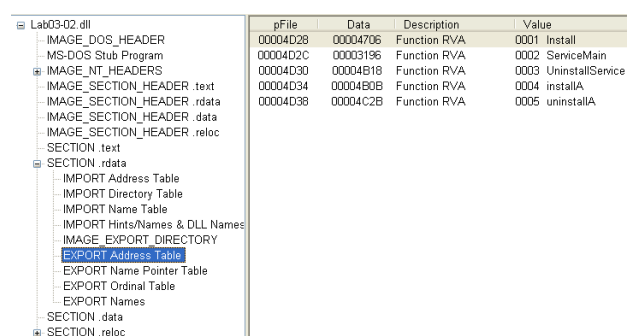
(1): 与上一个实验一样，在进行动态分析之前首先进行静态分析。先使用 PEView 查看导入函数：



pFile	Data	Description	Value
00004400	0000568C	Hint/Name RVA	0147 OpenServiceA
00004404	0000567C	Hint/Name RVA	0078 DeleteService
00004408	0000566C	Hint/Name RVA	0172 RegOpenKeyExA
0000440C	00005658	Hint/Name RVA	017B RegQueryValueExA
00004410	0000564A	Hint/Name RVA	015B RegCloseKey
00004414	00005638	Hint/Name RVA	0145 OpenSCManagerA
00004418	00005626	Hint/Name RVA	004C CreateServiceA
0000441C	00005610	Hint/Name RVA	0034 CloseServiceHandle
00004420	00005600	Hint/Name RVA	015E RegCreateKeyA
00004424	000055EE	Hint/Name RVA	0186 RegSetValueExA
00004428	000055D0	Hint/Name RVA	018E RegisterServiceCtrlHandlerA
0000442C	0000559C	Hint/Name RVA	01AE SetServiceStatus
00004430	00000000	End of Imports	ADVAPI32.dll
00004434	00005548	Hint/Name RVA	0150 GetStartupInfoA
00004438	0000555A	Hint/Name RVA	0043 CreatePipe
0000443C	00005568	Hint/Name RVA	00F5 GetCurrentDirectoryA
00004440	00005536	Hint/Name RVA	0044 CreateProcessA
00004444	00005590	Hint/Name RVA	0308 lstrlenA
00004448	0000559C	Hint/Name RVA	0271 SetLastError
0000444C	000055AC	Hint/Name RVA	01F5 OutputDebugStringA
00004450	00005528	Hint/Name RVA	001B CloseHandle
00004454	0000551C	Hint/Name RVA	0218 ReadFile
00004458	0000550C	Hint/Name RVA	0165 GetTempPathA
0000445C	000054F8	Hint/Name RVA	0121 GetLongPathNameA
00004460	000054E8	Hint/Name RVA	01C2 LoadLibraryA
00004464	000054D6	Hint/Name RVA	013E GetProcAddress
00004468	000054C6	Hint/Name RVA	004A CreateThread
0000446C	000054B6	Hint/Name RVA	015D GetSystemTime
00004470	000054A0	Hint/Name RVA	02CE WaitForSingleObject
00004474	0000548E	Hint/Name RVA	029F TerminateThread
00004478	00005486	Hint/Name RVA	0296 Sleep
0000447C	00005580	Hint/Name RVA	011A GetLastError
00004480	00005470	Hint/Name RVA	0124 GetModuleFileNameA
00004484	00000000	End of Imports	USER32.dll
00004488	000059E4	Hint/Name RVA	00AC chdir
0000448C	000059D8	Hint/Name RVA	01C5 _stricmp
00004490	000059C8	Hint/Name RVA	009D _adjust_fdiv
00004494	000059BE	Hint/Name RVA	0291 malloc
00004498	000059B2	Hint/Name RVA	010F _initterm
0000449C	000059AA	Hint/Name RVA	025E free
000044A0	00005982	Hint/Name RVA	000E ??Type_info@@@UAE@vZ
000044A4	00005972	Hint/Name RVA	00CA _except_handler3
000044A8	0000595C	Hint/Name RVA	0041 _CxxThrowException
000044AC	000059EE	Hint/Name RVA	01C1 _stricmp
000044B0	0000594E	Hint/Name RVA	0042 _EH_prolog
000044B4	0000593A	Hint/Name RVA	0049 __CxxFrameHandler
000044B8	00005930	Hint/Name RVA	0267 strchr
000044BC	00005928	Hint/Name RVA	0134 _ftoa
000044C0	0000591E	Hint/Name RVA	02C5 strstr
000044C4	00005914	Hint/Name RVA	02BF strncat
000044C8	0000590A	Hint/Name RVA	02BE strlen
000044CC	00005800	Hint/Name RVA	02B5 sscanf
000044D0	000057F8	Hint/Name RVA	023E atol
000044D4	000057E8	Hint/Name RVA	000F ??2@YAPAVI@Z
000044D8	000057EC	Hint/Name RVA	0299 memset
000044DC	00005776	Hint/Name RVA	02F1 wcsombs
000044E0	00005762	Hint/Name RVA	02C1 strncpy
000044E4	0000578C	Hint/Name RVA	02B6 strcat

图 3.18: 查看导入函数

我们在导入函数中可以发现一些如 `xxService` 等与服务有关的函数, `RegSetValue` 等与注册表操作有关的函数。接下来我们再查看导出函数:



pFile	Data	Description	Value
00004D28	00004706	Function RVA	0001 Install
00004D2C	00003196	Function RVA	0002 ServiceMain
00004D30	00004B18	Function RVA	0003 UninstallService
00004D34	00004B0B	Function RVA	0004 installA
00004D38	00004C2B	Function RVA	0005 uninstallA

图 3.19: 查看导出函数

我们看到其中有 `Install`、`ServiceMain`、`uninstall` 等函数, 再结合刚才的导入函数, 我们推测恶意代码会安装成一个服务, 从而使其能够正常运行。

(2): 接下来我们使用 `Strings` 查看字符串列表:

```
GET
HTTP/1.1
%0a%0a
1234567890123456
quit
exit
getFile
cmd.exe /c
ABCDEFGHijklmnopqrstuvwxy@123456789+/
--!>
<!--
.PaX
.PaD
DependOnService
RpcSs
ServiceDll
GetModuleFileName() get dll path
Parameters
Type
Start
ObjectName
LocalSystem
ErrorControl
DisplayName
Description
Depends INet+, Collects and stores network configuration and location information
, and notifies applications when this information changes.
ImagePath
%SystemRoot%\System32\svchost.exe -k
SYSTEM\CurrentControlSet\Services\
CreateService(<sz> error %d
Intranet Network Awareness (INA+)
%SystemRoot%\System32\svchost.exe -k netsvcs
OpenSCManager()
You specify service name not in Svchost\netsvcs, must be one of following:
RegQueryValueEx(Svchost\netsvcs)
netsvcs
RegOpenKeyEx(<sz> KEY_QUERY_VALUE success.
RegOpenKeyEx(<sz> KEY_QUERY_VALUE error .
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
```

图 3.20: 字符串列表

我们看到有许多与服务相关的字符串，也更加确定刚才的预测，即使用导出函数将自身注册成一个服务运行。

(3): 接下来就开始动态分析，刚才一样相对虚拟机进行快照。结合刚才静态分析的结论，我们知道他将会注册表有一定操作，但由于实验样本并不是.exe 文件，无法像 Lab3-1 一样使用 Process Monitor 进行监视，所以我们使用 Regshot 来查看安装前后的注册表变化。在运行前，我们先用 Regshot 进行快照。然后我们再使用 rundll32.exe Lab03-02.dll,InstallA 来安装这个 dll 文件，紧接着再进行快照。对比两次快照，可以看到恶意代码为自己安装了一个 IPRIP 服务，且将其加入到开机启动项目中。

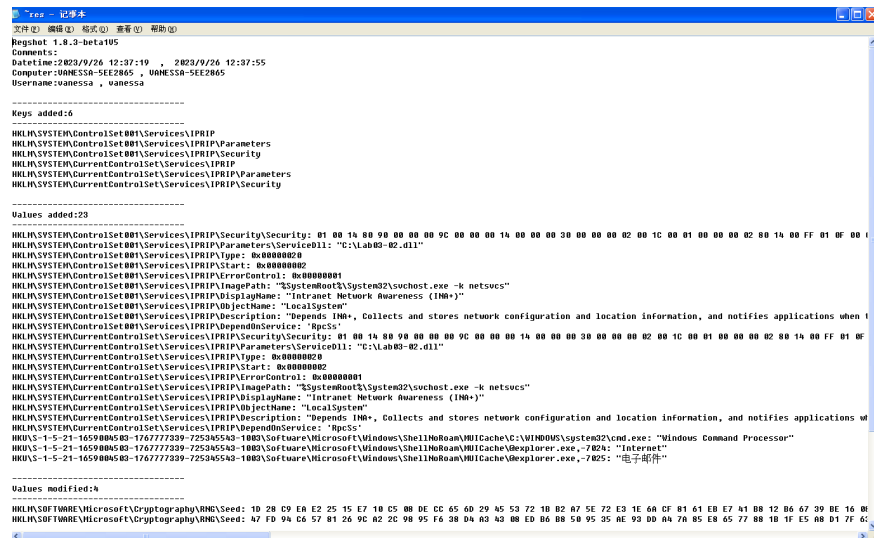


图 3.21: 两次快照的对比

(4): 接下来我们使用 **net start IPRIP** 来运行该服务。再“运行”中输入“services.msc”查看服务

名称 /	描述	状态	启动类型	登录为
HTTP SSL	此...		手动	本地系统
Human Interfac...	启...	已禁用		本地系统
IMAPI CD-Burni...	用...		手动	本地系统
Indexing Service	本...		手动	本地系统
Intranet Netwo...	Dep...	已启动	自动	本地系统
IPSEC Services	管...	已启动	自动	本地系统
Logical Disk M...	监...	已启动	自动	本地系统
Logical Disk M...	配...		手动	本地系统

图 3.22: services.msc 中查看服务

(5): 接下来打开 Process Explorer, 点击 Find-> Find Handle or DLL...-> “Lab03-02.dll”, 可以看到该恶意代码依附于 “svchost.exe” 运行, PID 为 1136。

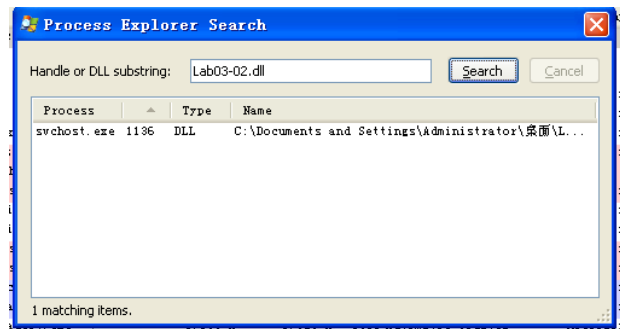


图 3.23: Process Explorer 查看

(6): 在 Process Monitor 中打开过滤器, 选择 “Parent PID”, 填入 “1136” 进行过滤查看, 我们可以看到一些父进程的 svchost.exe 操作

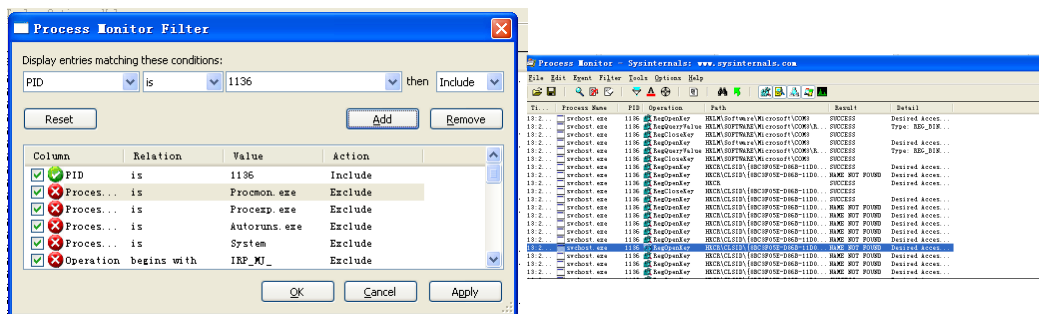


图 3.24: Process Monitor 查看

(7): 接下来我们借助 Wireshark 来查看, 我们看到恶意代码与该域名进行数据交互请求, 我们再由之前分析的结果进行搜索过滤, 找到 http.request.method==”GET”

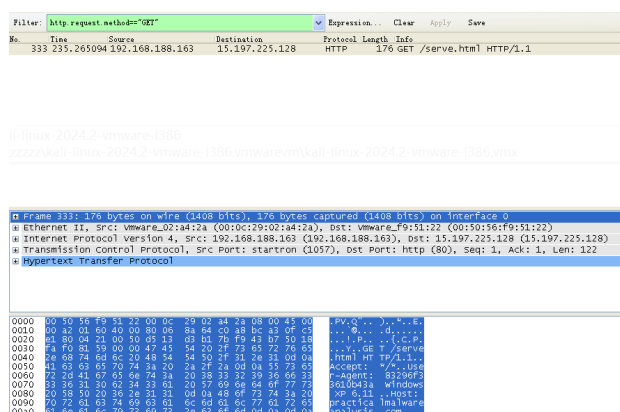


图 3.25: 使用 Wireshark 过滤查看

观察发现 user-agent 的前半部分是主机名, 后半部分的 windows xp 6.11 是固定的, 可作为网络特征。

问题一：你怎样才能让这个恶意代码自行安装？

我们使用 rundll32.exe Lab03-02.dll,installA 指令将该代码自动安装。

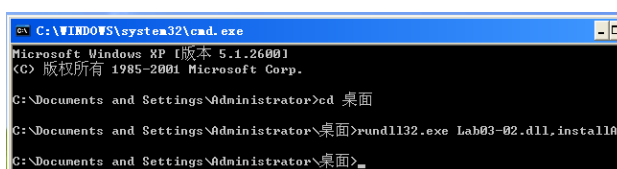


图 3.26: 自行安装

问题二：在安装之后，你如何让这个恶意代码运行起来？

使用 net start IPRIP 指令。



图 3.27: 代码运行

问题三：你怎么能找到这个恶意代码是在哪个进程下运行的？

主要是使用 Process Explorer 来确定哪个进程正在运行服务。由于恶意代码将会运行在一个系统上的 svchost.exe 进程中, 因此查看每个进程, 直到看到该服务名。或者是直接在 Process Explorer 的 Find Dll 功能来搜索 Lab03-02.dll, 进行确定。

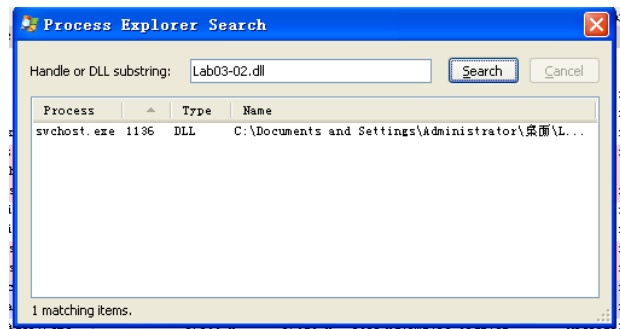


图 3.28: Process Explorer 的 Find Dll 功能来搜索 Lab03-02.dll

问题四：你可以在 procmon 工具中设置什么样的过滤器，才能收集这个恶意代码的信息？

在 procmon 工具中，可以使用在 Process Explorer 中发现的 PID 进行过滤。



图 3.29: procmon 中使用 PID 进行过滤

问题五：这个恶意代码在主机上的感染迹象特征是什么？

默认情况下，恶意代码将安装为 IPRIP 服务，显示的服务名称为 Intranet Network Awareness (INA+), 描述为 “Depends INA+, Collects and stores network configuration and location information, and notifies applications when this information changes”。它将自身安装在注册表中 HKLM/SYSTEM/CurrentControlSet\Services/IPRIP/ImagePath 项中，值为 C:\Program Files\Intranet Network Awareness\Lab03-02.dll。如果将 Lab03-02.dll 重命名为其他文件名，如 malware.dll，那么该恶意代码就会把 malware.dll 写入到注册表项中，而不是使用名称 Lab03-02.dll。

问题六：这个恶意代码是否存在一些有用的网络特征码？

我们在查看恶意代码字符串发现的域名 practicalmalwareanalysis.com，经分析，恶意代码首先申请解析域名 practicalmalwareanalysis.com，然后通过 80 端口连接到这台主机，使用的协议看起来似乎是 HTTP 协议，它在做一个 GET 请求 serve.html，其用户代理为 “主机名 Windows XP 6.11”。

3.3 Lab3-3

在一个安全的环境中执行 Lab03-03.exe 文件中发现的恶意代码，同时使用基础的动态行为分析工具监视它的行为。

(1): 动态分析前，先进行基本的静态分析。使用 PEiD 打开该文件，可以看到该文件没有加壳迹象。

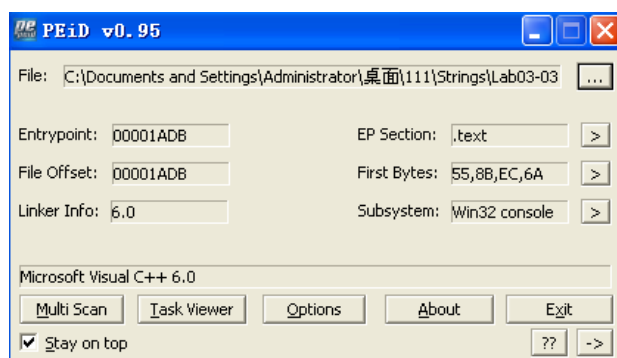


图 3.30: 使用 PEiD 查看是否加壳

(2): 再使用 Strings 来查看样本的字符串，我们发现该文件中有大量的字符 A:

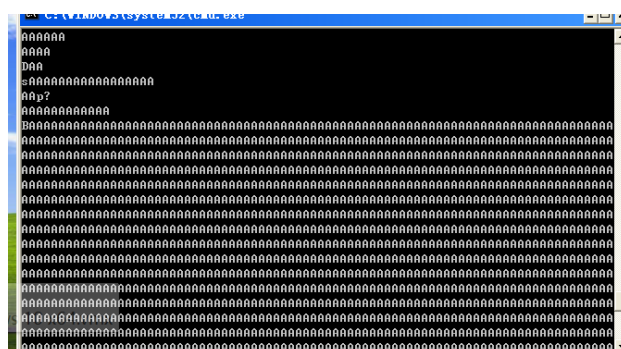


图 3.31: 查看 Lab03-03.exe 中的字符串部分结果

(3): 接下来进行动态分析。打开 Process Explorer 并运行实验样本

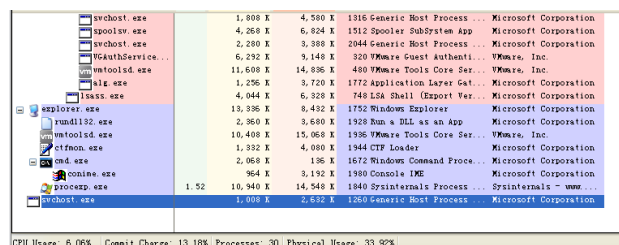


图 3.32: Process Explorer 查看进程

我们看到打开文件后，该文件不仅创建了自己的进程，还创建了一个 svchost.exe 进程，反而 Lab03-03.exe 的进程自行退出，仅剩下子程序独自运行，一般，svchost.exe 是 services.exe 的子进程。所以我们对该进程继续分析：

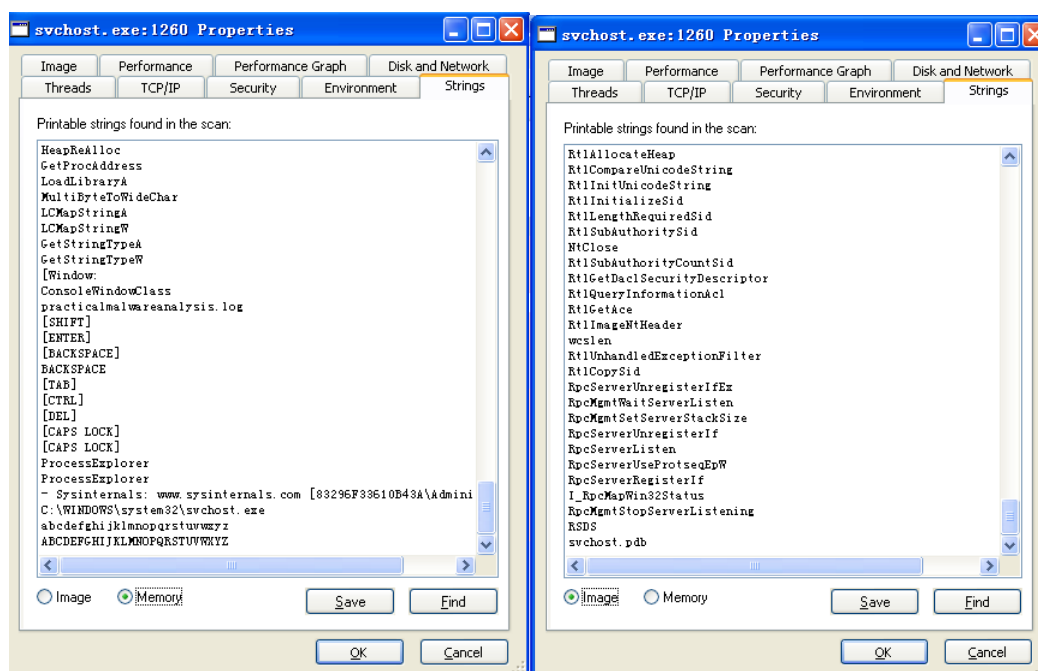


图 3.33: Process Monitor 查看

通过分别查看磁盘与内存中的字符串,能发现内存中多出了 practicalmalwareanalysis.log 和 [SHIFT]、[ENTER]、[BACKSPACE] 等不该出现的字符串,推测是一个敲击键盘的记录器。

(4): 打开 Process Monitor, 添加过滤 PID is 1260 进行查看:

Time...	Process Name	PID	Operation	Path	Result	Detail
16:3...	svchost.exe	1260	CreateFile	C:\Documents and Settings\Admini...	SUCCESS	Desired Acces...
16:3...	svchost.exe	1260	QueryStanda...	C:\Documents and Settings\Admini...	SUCCESS	AllocationSi...
16:3...	svchost.exe	1260	WriteFile	C:\Documents and Settings\Admini...	SUCCESS	Offset: 296,
16:3...	svchost.exe	1260	WriteFile	C:\Documents and Settings\Admini...	SUCCESS	Offset: 308,
16:3...	svchost.exe	1260	WriteFile	C:\Documents and Settings\Admini...	SUCCESS	Offset: 330,
16:3...	svchost.exe	1260	WriteFile	C:\Documents and Settings\Admini...	SUCCESS	Offset: 334,
16:3...	svchost.exe	1260	CloseFile	C:\Documents and Settings\Admini...	SUCCESS	
16:3...	svchost.exe	1260	CreateFile	C:\Documents and Settings\Admini...	SUCCESS	Desired Acces...
16:3...	svchost.exe	1260	QueryStanda...	C:\Documents and Settings\Admini...	SUCCESS	AllocationSi...
16:3...	svchost.exe	1260	WriteFile	C:\Documents and Settings\Admini...	SUCCESS	Offset: 336,
16:3...	svchost.exe	1260	CloseFile	C:\Documents and Settings\Admini...	SUCCESS	
16:3...	svchost.exe	1260	CreateFile	C:\Documents and Settings\Admini...	SUCCESS	Desired Acces...
16:3...	svchost.exe	1260	QueryStanda...	C:\Documents and Settings\Admini...	SUCCESS	AllocationSi...
16:3...	svchost.exe	1260	WriteFile	C:\Documents and Settings\Admini...	SUCCESS	Offset: 338,
16:3...	svchost.exe	1260	CloseFile	C:\Documents and Settings\Admini...	SUCCESS	
16:3...	svchost.exe	1260	CreateFile	C:\Documents and Settings\Admini...	SUCCESS	Desired Acces...
16:3...	svchost.exe	1260	QueryStanda...	C:\Documents and Settings\Admini...	SUCCESS	AllocationSi...
16:3...	svchost.exe	1260	WriteFile	C:\Documents and Settings\Admini...	SUCCESS	Offset: 340,
16:3...	svchost.exe	1260	CloseFile	C:\Documents and Settings\Admini...	SUCCESS	

图 3.34: 在 Process Monitor 过滤查看

根据之前的猜测，首先在桌面新建一个记事本文件，在文件中输入字符串，再回到 Process Monitor 观察相关过滤内容，发现有许多 CreateFile 和 WriteFile 操作。我们根据其中提示的路径找到 practicalmalwareanalysis.log 文件，打开该日志文件，能看到我刚刚进行的键盘输入都被记录：

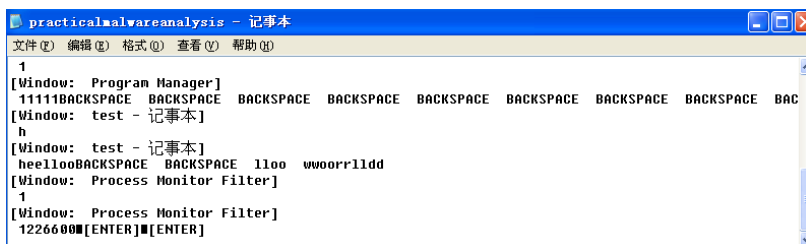


图 3.35: practicalmalwareanalysis.log 文件中的内容

问题一：当你使用 Process Explorer 工具进行监视时，你注意到了什么？

打开 Process Explorer 工具监听，然后我启动 Lab03-03.exe 后发现，该文件创建了一个子进程 svchost.exe 并很快结束掉自己的进程，留下子进程单独运行。

问题二：你可以找出任何的内存修改行为吗？

我们对比内存映像与内存映像中的 svchost.exe 发现他们并不是一样的，在内存映像中有像 practicalmalwareanalysis.log 和 [ENTER] 这样的字符串，但在磁盘映像中却没有。

问题三：这个恶意代码在主机上的感染迹象特征是什么？

该恶意代码的主要功能是实现了一个键盘记录器，将键盘输入的所有信息都保存在 practicalmalwareanalysis.log 文件中。

问题四：这个恶意代码的目的是什么？

该恶意代码将内存中的 svchost.exe 进程替换成一个键盘键入记录器，从而实现记录用户的键盘输入信息。

3.4 Lab3-4

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

(1): 首先先进行基础的静态分析，使用 PEiD 打开文件，观察文件是否有被加壳的迹象：

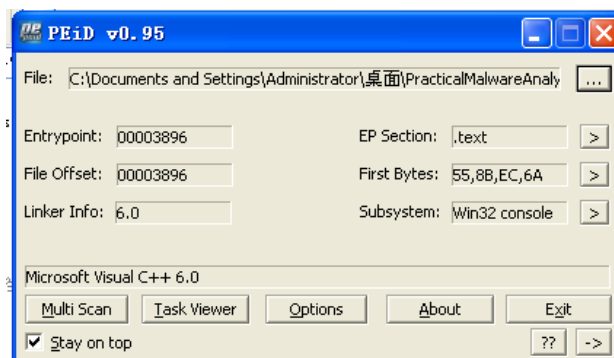


图 3.36: PEiD 查看是否有加壳迹象

(2): 通过 Strings 查看 Lab03-04.exe 字符串：

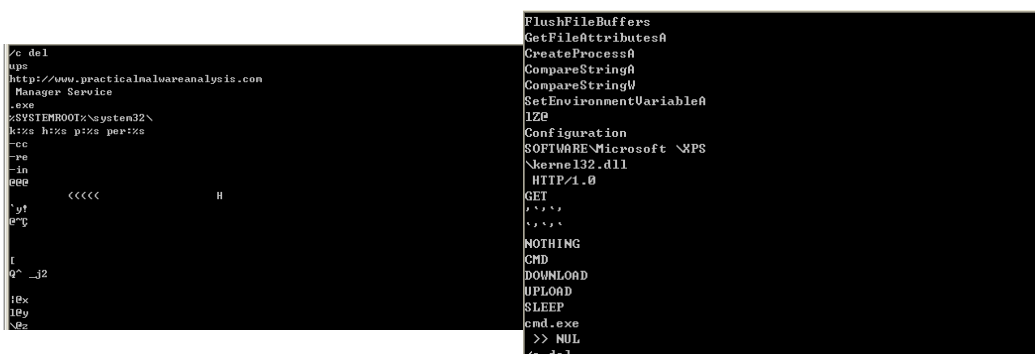


图 3.37: Strings 查看 Lab03-04.exe 字符串 (部分)

根据查询结果我们进行简单的判断：

- 猜测文件和环境的函数名：GetFileAttributes、GetEnvironmentStrings

- 系统命令：cmd.exe、/c del、CMD、SLEEP、DOWNLOAD、UPLOAD
- 疑似命令行参数：-cc、-re、-in、k:%s h:%s p:%s per:%s
- HTTP 命令：HTTP/1.0、GET
- 域名：http://www.practicalmalwareanalysis.com
- 系统文件：%SYSTEMROOT%/system32

(3): 接下来使用 Dependency Walker 打开文件，初步找到一些文件中的可疑函数。

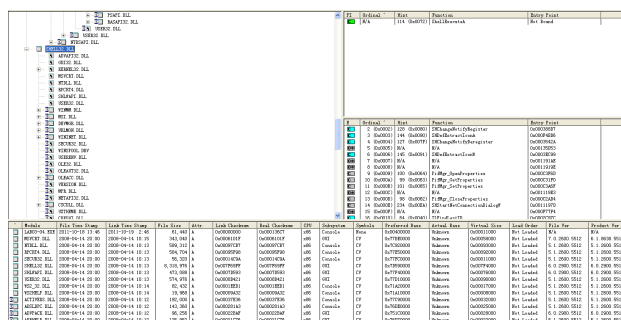


图 3.38: Dependency Walker 查找可疑函数

(4): 接下来我们使用动态分析技术。先打开 Process Monitor 监听程序，然后我们运行 Lab03-04.exe，发现运行几秒后文件就被自动删除了。我们再用 Process Monitor 中添加过滤规则 Process Name is Lab03-04.exe 和 Operation is Process Create 进行查看：

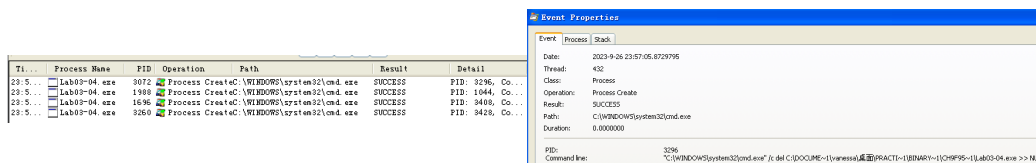


图 3.39: 在 Process Monitor 中查看

由此我们猜测该恶意代码启动了新的进程,该进程是通过 cmd 执行了“del C:/DOCUMENTI/vanessa/桌面/PRACTI/I/BINARY/I/CH9F95/I/Lab03-04.exe » NUL”从而实现对自身文件的删除。

问题一：当你运行这个文件时，会发生什么呢？

我们发现该文件运行后就消失了，即将自己删除了。

问题二：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。

问题三：是否有其他方式来运行这个程序？

通过查询，我们发现可以尝试使用在字符串列表中显示的一些命令行参数，比如-in，但并没有得出有效结果，所以需要更深入的分析。

3.5 VirusTotal 查看实验样本

Lab03-01.exe

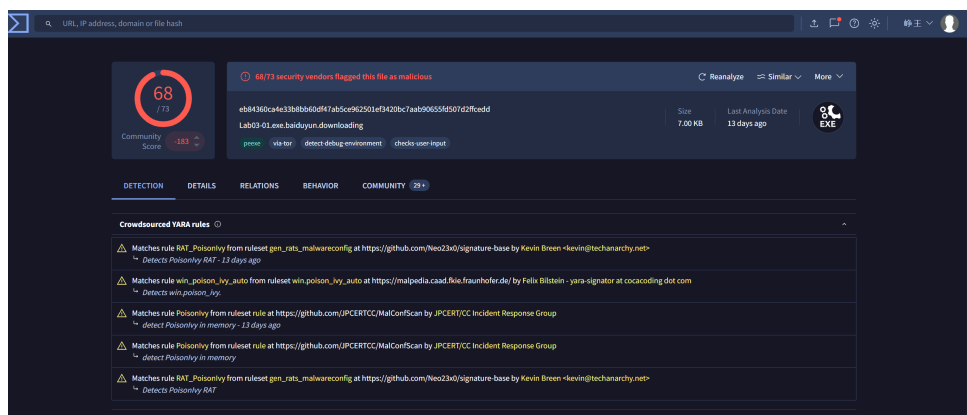


图 3.40: VirusTotal 中 Lab03-01.exe 分析

我们该文件可以匹配很多的病毒特征，有 68 个报告为高危，community score 达到-183，证明其是恶意代码。查看详细信息：

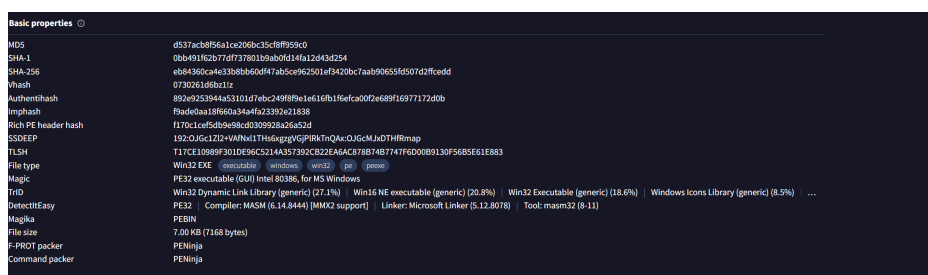


图 3.41: Lab03-01.exe 的详细信息

Lab03-02.dll

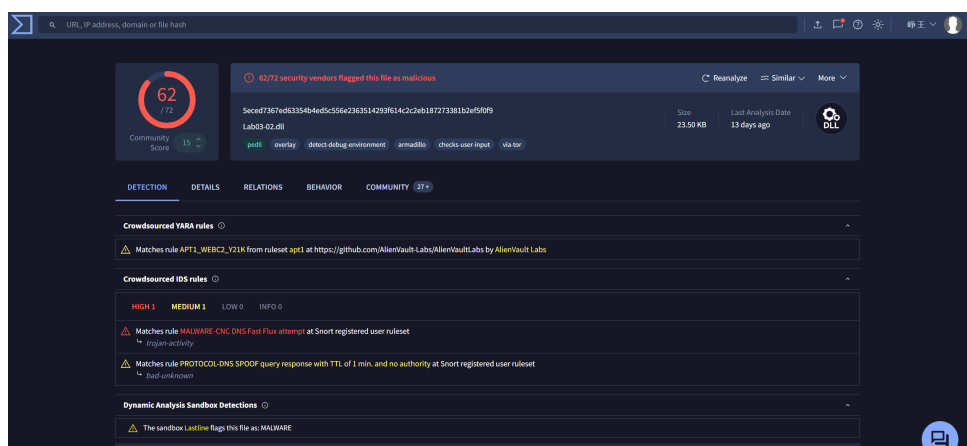


图 3.42: VirusTotal 中 Lab03-02.dll 分析

我们看到有 62 个报告为高危，我们再查看详细信息发现与我们之前的分析有关：

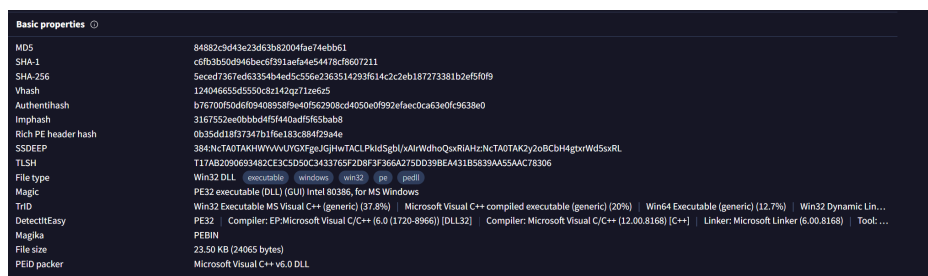


图 3.43: Lab03-02.dll 的详细信息

Lab03-03.exe

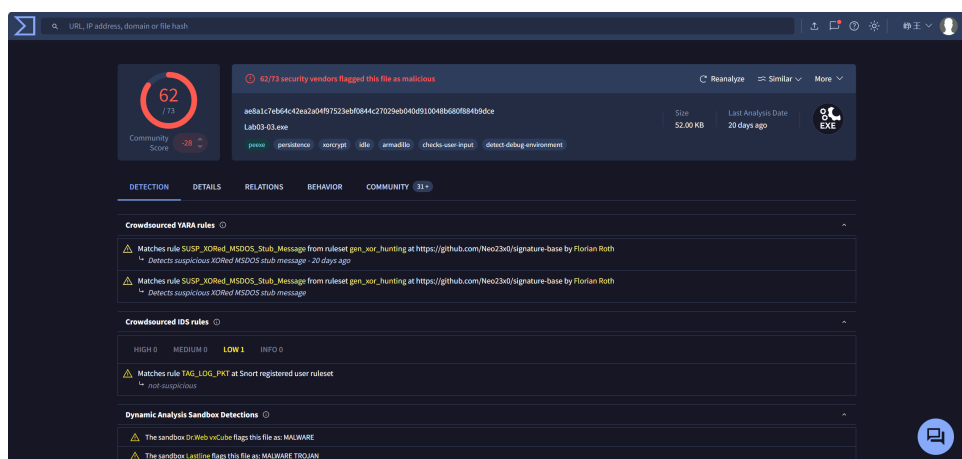


图 3.44: VirusTotal 中 Lab03-03.exe 分析

我们看到该文件匹配了 62 个恶意特征，我们再查看详细信息发现与我们之前的分析有关：

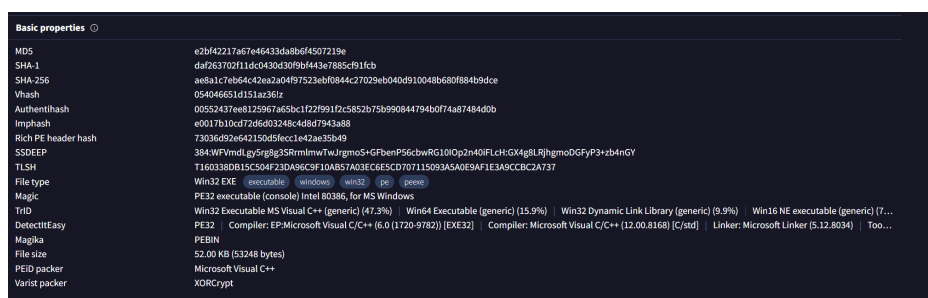


图 3.45: Lab03-03.exe 的详细信息

Lab03-04.exe

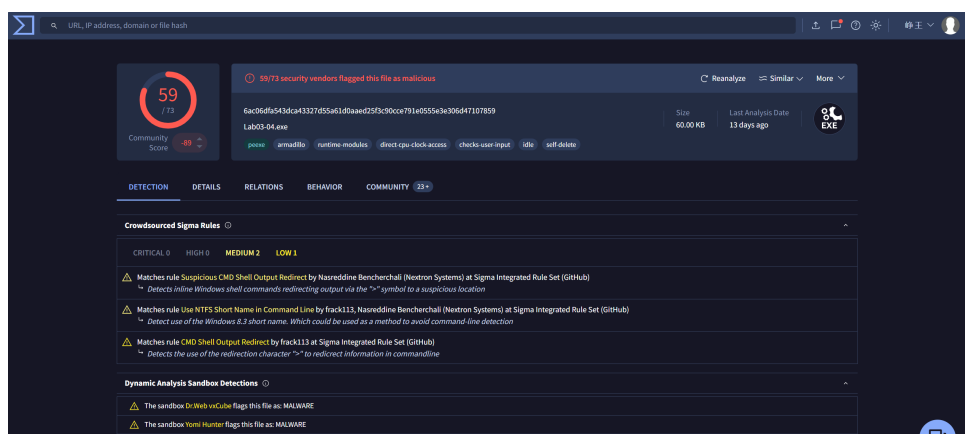


图 3.46: VirusTotal 中 Lab03-04.exe 分析

该文件匹配上了 59 个恶意特征，社区评分为-89 分，可以认定为恶意文件。查看文件的详细信息：

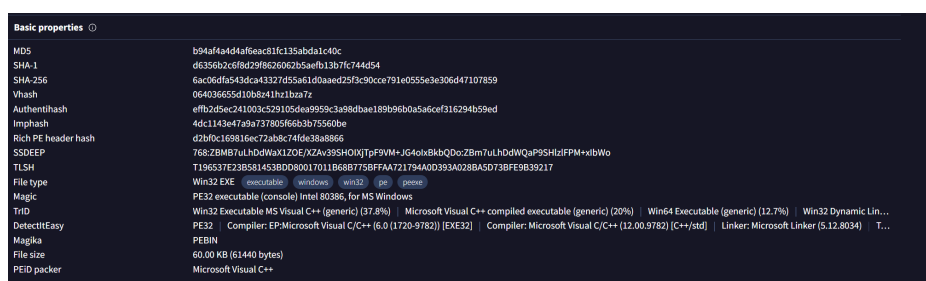


图 3.47: Lab03-04.exe 的详细信息

3.6 YARA 检测

1. 编写 yara 规则

我们根据上述的分析，编写 YARA 规则：

Yara 规则

```
1 rule Lab03_01 {
2   meta:
3     description = "Lab03-01.exe"
4   strings:
5     $s1 = "vmx32to64.exe" fullword ascii
6     $s2 = "SOFTWARE\\Classes\\http\\shell\\open\\commandV" fullword ascii
7     $s3 = " www.practicalmalwareanalysis.com" fullword ascii
8     $s4 = "CONNECT %s:%i HTTP/1.0" fullword ascii
9     $s5 = "advpack" fullword ascii
10    $s6 = "VideoDriver" fullword ascii
11    $s7 = "AppData" fullword ascii /* Goodware String - occurred 74 times */
12    $s8 = "6I*h<8" fullword ascii /* Goodware String - occurred 1 times */
13    $s9 = "StubPath" fullword ascii /* Goodware String - occurred 1 times */
14    $s10 = "WinVMX32-" fullword ascii
```

```
15     $s11 = "Software\\Microsoft\\Active Setup\\Installed Components\\" fullword
        ascii /* Goodware String - occured 4 times */
16     $s12 = "^-m-m<|<|<|M" fullword ascii
17 condition:
18     uint16(0) == 0x5a4d and filesize < 20KB and 8 of them
19 }
20
21 rule Lab03_02 {
22     meta:
23         description = "Lab03-02.dll"
24     strings:
25         $x1 = "%SystemRoot%\\System32\\svchost.exe -k " fullword ascii
26         $x2 = "cmd.exe /c " fullword ascii
27         $s3 = "RegOpenKeyEx(%s) KEY_QUERY_VALUE error ." fullword ascii
28         $s4 = "Lab03-02.dll" fullword ascii
29         $s5 = "practicalmalwareanalysis.com" fullword ascii
30         $s6 = "RegOpenKeyEx(%s) KEY_QUERY_VALUE success." fullword ascii
31         $s7 = "GetModuleFileName() get dll path" fullword ascii
32         $s8 = "dW5zdXBwb3J0" fullword ascii /* base64 encoded string 'unsupport' */
33         $s9 = "Y29ubmVjdA==" fullword ascii /* base64 encoded string 'connect' */
34         $s10 = "OpenService(%s) error 2" fullword ascii
35         $s11 = "OpenService(%s) error 1" fullword ascii
36         $s12 = "CreateService(%s) error %d" fullword ascii
37         $s13 = "You specify service name not in Svchost//netsvcs, must be one of
            following:" fullword ascii
38         $s14 = "RegQueryValueEx(Svchost\\netsvcs)" fullword ascii
39         $s15 = "netsvcs" fullword ascii
40         $s16 = "serve.html" fullword ascii
41         $s17 = "DependOnService" fullword ascii
42         $s18 = "::$2:K:U:\\:1:" fullword ascii
43         $s19 = "uninstall is starting" fullword ascii
44         $s20 = "uninstall success" fullword ascii
45     condition:
46         uint16(0) == 0x5a4d and filesize < 70KB and 1 of ($x*) and 4 of them
47 }
48
49 rule Lab03_03 {
50     meta:
51         description = "Lab03-03.exe"
52     strings:
53         $s1 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ascii
54         $s2 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
```

```
55  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ascii
56  $s3 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
57  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ascii
58  $s4 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
59
60  "AAAAAAAAAAAAAAAAAAAA" ascii
61  $s5 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
62  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ascii
63  $s6 = "aAAAAAAAAAAAAAAAAAAAAA" ascii
64  $s7 = "\\svchost.exe" fullword ascii
65  $s8 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ascii
66  $s9 = "BAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
67  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ascii
68  $s10 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAA" ascii
69  $s11 = "AAAAAABAAAAA" ascii
70  $s12 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
71  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ascii
72  $s13 = "AAqAApAAAsAAArAAAUAAAtAAAwAAAvAAAYAAAXAAA" fullword ascii
73  $s14 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
74  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ascii
75  $s15 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
76  "AAAAAAq" fullword ascii
77
78  $s16 = "wqpWLKla/.5a$/.4&)a21 \"$a'.3a5)3$ %a% 5 LKALK #/.3, -a13.&3 ,a5$3,(/
    5(./LKAAAA" fullword ascii
79  $s17 = "- 22AA13 \"$5(\" -, -6 3$ / -82(2o-.&AAAAaAAA" fullword ascii
80  $s18 = "+A+A+A+A" fullword ascii /* reversed goodwill string 'A+A+A+A+' */
81  $s19 = "(\"3.2.'5a" fullword ascii /* hex encoded string '2Z' */
82  $s20 = "wqswLKla/.5a$/.4&)a21 \"$a'.3a25%(.a/(5( -(; 5(./LKAAAA" fullword
    ascii
83  condition:
84      uint16(0) == 0x5a4d and filesize < 200KB and 8 of them
85  }
86
87  rule Lab03_04 {
88      meta:
89          description = "Lab03-04.exe"
90      strings:
91          $s1 = "http://www.practicalmalwareanalysis.com" fullword ascii
92          $s2 = "%SYSTEMROOT%\\system32\\" fullword ascii
93          $s3 = " HTTP/1.0" fullword ascii
94          $s4 = " Manager Service" fullword ascii
```

```
95 $s5 = "UPLOAD" fullword ascii /* Goodware String - occurred 1 times */
96 $s6 = "DOWNLOAD" fullword ascii /* Goodware String - occurred 26 times */
97 $s7 = "command.com" fullword ascii /* Goodware String - occurred 55 times */
98 $s8 = "COMSPEC" fullword ascii /* Goodware String - occurred 140 times */
99 $s9 = "\"WSH(" fullword ascii /* Goodware String - occurred 1 times */
100 $s10 = "SOFTWARE\\Microsoft \\XPS" fullword ascii
101 $s11 = "k:%s h:%s p:%s per:%s" fullword ascii
102 $s12 = ">> NUL" fullword ascii
103 $s13 = "/c del " fullword ascii
104 $s14 = "6KRich" fullword ascii
105 condition:
106     uint16(0) == 0x5a4d and filesize < 200KB and 8 of them
107 }
```

2.Yara 扫描检测

首先我们扫描对应放置实验样本的文件夹进行查看：

```
PS C:\Users\王峥\Desktop\恶意代码\上机实验样本> .\yara64 -r aaa.yar lab3
Lab03_01 lab3\Lab03-01.exe
Lab03_02 lab3\Lab03-02.dll
Lab03_04 lab3\Lab03-04.exe
Lab03_03 lab3\Lab03-03.exe
```

图 3.48: 对相关文件夹进行检测

对 C 盘进行扫描，代码如下：

```
import json
import os
import time

begin_time=time.time()

os.chdir(r"C:\Users\王峥\Desktop\恶意代码\上机实验样本")

os.system(r"yara64.exe -r lab3.yar C:\\")

end_time=time.time()

print(end_time-begin_time)
```

图 3.49: 对 C 盘进行扫描的代码

用时结果如下：

```
145.63248467445374
Process finished with exit code 0
```

图 3.50: 用时结果输出

我们看到可以将对应的实验样本成功扫描出来接下来我们尝试扫描‘桌面’文件查看结果：


```
PS C:\Users\王峥\Desktop\恶意代码\上机实验样本> .\yara64 -r aaa.yar C:\Users\王峥\Desktop
Lab03_04 C:\Users\王峥\Desktop\恶意代码\BinaryCollection\Chapter_16\Lab16-01.exe
Lab03_01 C:\Users\王峥\Desktop\恶意代码\BinaryCollection\Chapter_3L\Lab03-01.exe
Lab03_02 C:\Users\王峥\Desktop\恶意代码\BinaryCollection\Chapter_3L\Lab03-02.dll
Lab03_04 C:\Users\王峥\Desktop\恶意代码\BinaryCollection\Chapter_3L\Lab03-04.exe
Lab03_04 C:\Users\王峥\Desktop\恶意代码\BinaryCollection\Chapter_9L\Lab09-01.exe
Lab03_03 C:\Users\王峥\Desktop\恶意代码\BinaryCollection\Chapter_12L\Lab12-02.exe
Lab03_03 C:\Users\王峥\Desktop\恶意代码\BinaryCollection\Chapter_17L\Lab17-03.exe
Lab03_03 C:\Users\王峥\Desktop\恶意代码\BinaryCollection\Chapter_3L\Lab03-03.exe
Lab03_01 C:\Users\王峥\Desktop\恶意代码\上机实验样本\Lab3\Lab03-01.exe
Lab03_02 C:\Users\王峥\Desktop\恶意代码\上机实验样本\Lab3\Lab03-02.dll
Lab03_04 C:\Users\王峥\Desktop\恶意代码\上机实验样本\Lab3\Lab03-04.exe
Lab03_03 C:\Users\王峥\Desktop\恶意代码\上机实验样本\Lab3\Lab03-03.exe
```

图 3.51: 对桌面进行检测

观察扫描的结果,我们发现该规则不仅能将 Lab3 中的实验样本检测出来,还能将 Chapter_9L 和 Chapter_16L 中的一些文件扫描初,可以简单的判断这些实验样本也与 Chapter_3L 中的文件相类似的恶意代码类型。

4 实验结论及心得体会

4.1 实验结论

通过本次实验,在之前掌握虚拟机的使用和静态分析的具体操作的基础上,着重学习了动态分析的操作方式,包括在静态分析的基础上尝试进行动态分析,对动态分析工具的使用。

- 在分析 Lab03-01.exe 时,我们识别了恶意代码的导入函数、字符串列表、感染迹象和网络特征码,从而了解了其行为,包括自复制、添加到启动项以及与特定域名的通信。
- 对于 Lab03-02.dll 的分析,我们发现该文件会自行安装为服务,并可通过命令行参数控制其行为。同时,我们学会了利用过程监视工具监测进程活动。
- 在 Lab03-03.exe 的分析中,我们注意到它包含一个键盘记录器,能够记录用户输入,这显示出其潜在的恶性性质。
- 在分析 Lab03-04.exe 时,发现它会删除自身文件,并且需要特定的命令行参数才能激活这一行为,这使得动态分析的效果受到一定限制。

由上述分析我们可以发现动态分析的优点有可以检测复杂的内存处理错误并且精度更高,但也具有像速度慢、效率低、复杂度更高,不能保证完整的代码覆盖率,可伸缩性差,难以进行大规模测试等等缺点。

同时,根据上述的动态分析结果,我们编写相关的 YARA 规则,使得对相关的恶意代码扫描更加迅速。

4.2 心得体会

这次实验在静态分析的基础上,首次进行动态分析。由于动态分析的特殊性,我们需要在虚拟机中完成,并进行虚拟快照从而在后续实验结束后能快速恢复虚拟机。

实验主要使用了各种动态分析工具,如 Process Monitor、Process Explorer、Wireshark 等,这些工具对于监视和理解恶意代码的行为至关重要。它们能够捕获关键的系统操作和网络通信,帮助分析人员追踪恶意行为,在实验中我也感觉到工具的便利。

手动分析结束后,我们根据上面实验样本中的特性进行 YARA 规则的编写,我也发现 YARA 规则在大规模的样本集中检测中发挥着积极的作用。通过实验我们也发现其检测的成功率也较高,是非常有效的检测方式。