



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

姓名：周钰宸

学号：2111408

专业：信息安全

2024 年 1 月 14 日

## 1 实验目的

1. 对教材第五章所学内容进行回顾，并熟练使用 Sandbox、Monitor process、Regshot 等多种动态分析工具来进行对恶意代码的动态分析。
2. 完成教材 Lab3 的实验内容，编写 Lab3 样本的 Yara 引擎规则，并测试规则的执行效率。

## 2 实验原理

### 2.1 静态分析工具及其功能

1. string.exe:

string 是一个简单而强大的工具，用于从二进制文件中提取 ASCII 和 Unicode 字符串。它可以帮助分析人员快速定位代码中的字符串，这对于恶意代码分析和漏洞挖掘非常有用。**用于在二进制文件中查找、分析和提取字符串**，帮助分析人员了解程序的常量和信息。

2. PEView:

PEView 可以用于**查看和分析可执行文件的工具**。它可以显示 PE 文件的结构，包括头部、节表、导入表、导出表、资源等信息，以及**执行动态链接库依赖性分析**。有助于理解二进制文件的组织结构和导入的外部依赖。

3. Dependency Walker:

Dependency Walker 也用于查看可执行文件和动态链接库的依赖性。它可以分析文件的依赖关系，显示哪些 DLL 被程序调用，以及这些 DLL 之间的依赖关系。用途：**通过分析二进制文件对其他模块（DLL 等）的依赖**，帮助理解程序的运行时行为和可能的问题。

4. IDA:

IDA 是一款强大的交互式反汇编工具，用于将二进制文件转换为汇编代码，并提供丰富的交互式分析和导航功能。它支持多种处理器架构，如 x86、ARM、MIPS 等。**将二进制文件转换为汇编代码，进行反汇编和分析**。分析人员可以通过 IDA 深入理解程序的功能、逻辑和漏洞，以及对恶意代码进行深入分析。

这些工具在恶意代码分析、逆向工程、软件安全性评估等领域发挥重要作用，能够帮助分析人员理解程序的结构、逻辑和行为。选择合适的工具取决于具体的分析需求和二进制文件的特征。

### 2.2 动态分析工具及其功能

1. OllyDBG: 这是一款强大的动态调试器，用于分析程序的运行时行为。它允许分析人员观察程序的内存、寄存器、堆栈、指令等，并能够在程序运行时进行断点设置、反汇编、修改内存等操作。**用于动态调试程序，分析程序运行时的内存状态、执行路径和行为，以及发现潜在的漏洞和恶意行为**。
2. Process Monitor: 这是一个高级的任务管理器，可以显示系统中所有进程的详细信息，包括进程的资源使用情况、打开的句柄、线程信息等。**用于查看和分析系统中运行的进程、线程、资源情况，以及进程间的关联和依赖关系**。

3. Process Explorer: 这也是一款高级系统监控工具，可实时展示系统中运行的进程、线程和资源使用情况，提供丰富的进程详细信息和系统性能图表，帮助用户管理、监控和优化系统性能。**用于进程查看与管理、系统资源监控、进程树形结构、详细信息查看、搜索与筛选、性能图表、系统信息查看和窗口查看。**
4. Regshot: 这是一个注册表快照工具，用于比较系统或程序在两个时间点的注册表状态，找出在两个时间点注册表的变化。**用于监视程序运行时对注册表的操作，以便分析程序对系统配置的影响。**
5. Wireshark: 这是一款网络数据包分析工具，能够捕获、分析和展示网络通信中的数据包。它支持多种网络协议解析和过滤功能。**用于分析程序的网络活动，包括通信的协议、数据包内容、源目标信息等，有助于理解程序的网络行为和潜在的安全问题。**
6. ApateDNS: 这是一个能够控制 DNS 响应的程序，它主要在本地系统的 DNS 服务器上运行。通过 ApateDNS，**可以将恶意软件发出的 DNS 请求导向 UDP 端口 53 上的特定 IP 地址，从而实现欺骗。**
7. INetSim 工具: 基于 Linux 的 INetSim 是专为恶意软件分析设计的。它能够模仿诸如 http、https、DNS、FTP 等常见的互联网服务。当在 Windows 系统上进行动态的恶意软件分析时，可以在与分析机器同一网络的虚拟机上运行 INetSim。**这个工具可以模拟并响应恶意软件可能尝试使用的常见网络服务。**要启动特定的服务，需要在/etc/inetsim 目录下配置 inetsim.conf 文件，本实验主要关注 DNS 服务。
8. Netcat: 通常称为“网络瑞士军刀”) 是一个计算机网络实用程序，用于读取和写入打开的网络连接，使用 TCP 或 UDP 协议。**本次实验主要使用它的端口监听的功能，即在特定的端口上监听进入的连接。用作一个简单的工具来模拟网络服务或捕获恶意软件的网络通信。帮助分析师更好地理解恶意软件的行为和功能。**

这些工具在恶意代码分析、系统性能优化、网络安全等领域发挥重要作用。选择合适的工具取决于分析需求和目标。而本次实验也会重点使用上述动态分析工具来完成 Lab3 的要求。

## 2.3 Yara

YARA 是一种用于恶意软件分析、恶意代码检测和规则匹配的开源工具。它允许安全研究人员和分析师创建自定义规则，用于检测恶意代码、病毒、恶意软件家族等，并在大规模样本集合中进行自动化扫描。

### 2.3.1 Yara 的作用

1. 恶意代码检测: YARA 主要用于检测计算机病毒、恶意软件、恶意脚本等恶意代码。
2. 规则匹配: 它通过用户定义的规则匹配模式来查找文件、内存或其他数据源中的特定模式。
3. 恶意代码分类: YARA 可以帮助分析师将恶意代码样本分类到特定的家族或变种中。

### 2.3.2 Yara 规则

YARA 规则是用于定义恶意代码检测条件的文本文件。规则通常包含两部分: 元数据和规则体。元数据提供规则的名称、作者、描述等信息, 规则体包含条件和操作。

1. 规则条件：YARA 使用强大的规则语言，允许用户定义恶意代码的特征、字符串、正则表达式等。
2. 规则操作：当规则条件匹配时，可以执行自定义操作，如生成警报、拦截文件等。

本次实验也会在使用动态分析工具发现了相关病毒的特征后，尽可能编写有效且性能高效的 Yara 规则对 Lab3 病毒样本进行检测，并进一步测试 Yara 规则的相关效率，以求进一步的优化。

## 2.4 其它知识

### 2.4.1 互斥量 Mutex

创建网络互斥量（mutex）是恶意软件常用的技巧，主要有以下几个目的：

1. 单实例运行：通过创建一个特定名称的互斥量，恶意软件可以确保在受感染的系统上只有一个实例在运行。当恶意软件试图再次执行时，它会首先检查该互斥量是否存在。如果存在，恶意软件可能会选择不执行，从而避免多个实例同时运行，这可能会引起用户的注意或消耗过多的系统资源。
2. 避免重复感染：与单实例运行类似，通过检查互斥量，恶意软件可以确定系统是否已经被感染。这样它可以避免重复执行某些感染步骤，如复制自身或修改系统设置。
3. 同步多个恶意进程：在某些情况下，恶意软件可能会创建多个进程或线程来执行不同的任务。互斥量可以用于同步这些进程或线程，确保它们不会同时访问或修改同一资源，从而避免冲突或数据损坏。
4. 标识和通信：某些恶意软件可能会使用特定的互斥量名称来标识自己的版本或变种。此外，通过检查或创建特定的互斥量，不同的恶意软件家族或组件之间也可以进行简单的通信。
5. 反沙盒和反分析：某些高级恶意软件会使用互斥量作为反沙盒或反分析的技巧。例如，它们可能会检查某些与常见沙盒工具或分析环境相关的互斥量，以确定自己是否正在被分析。如果检测到这些互斥量，恶意软件可能会选择不执行或执行一些误导分析师的行为。

## 3 实验过程

### 3.1 实验环境

虚拟机软件	VMware Workstation 17 Pro
宿主机	Windows 11 家庭中文版
虚拟机操作系统 1	Windows XP Professional
虚拟机操作系统 2	Kali Linux 2021.1

表 1: 本次实验环境

### 3.2 Lab03-01

使用基本动态分析工具，分析在 Lab03-01.exe 中的恶意代码。

### 3.2.1 实验问题

#### 1. Q1: 这些恶意代码的导入函数和字符串有哪些？

回答：

(a) 首先使用 VirusTotal 进行对病毒的简单分析：

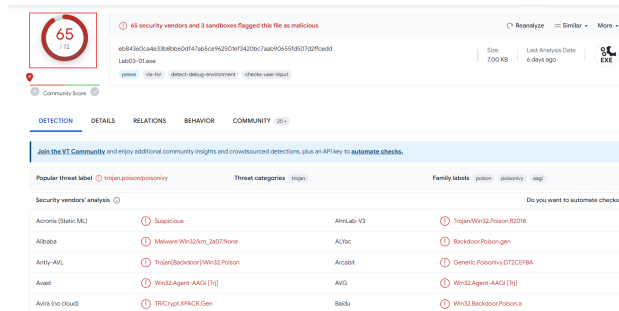


图 3.1: Lab03-01.exe 的 VirusTotal 的分析报告

图3.1可以看到 Lab03-01.exe 能够匹配较多的病毒签名特征，并且其社区的威胁投票得分也较高，能够初步证明其作为病毒的特征

(b) 然后先将其导入到 Lab2 中创立好的虚拟机隔离环境 Windows XP 中进行实验。使用基础的静态分析工具对其其他特征进行分析。

首先由于一个病毒若进行过加壳，那它真正的导入函数和字符串是不会轻易暴露给病毒分析人员的，为了能够根据其真实的导入函数和字符串判断其恶意目的，因此第一件要做的事是对其加壳情况进行判断，并且必要时使用脱壳工具（或 upx 或者 LinuxUnpacker）或手动进行脱壳。使用 PEiD 查看 Lab03-01.exe 后发现如下特征：

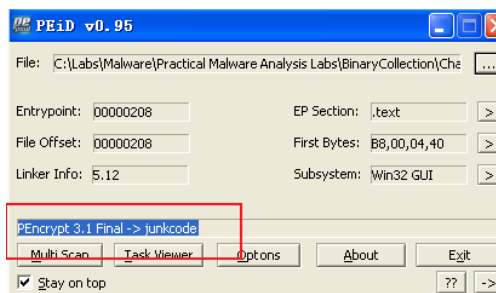


图 3.2: PEiD 对 Lab03-01.exe 的加壳分析结果

图3.2可以看到其在加壳信息位置显示了 **PEncrypt 3.1 Final -> junkcode**，这句话的含义为：

- PEncrypt 3.1 Final：这是加壳程序的名称和版本。加壳程序是一种特殊的软件，它可以对另一个软件（通常是一个可执行文件）进行加密和/或混淆，以防止反向工程师轻易地分析和修改它。
- junkcode：这是加壳程序使用的一种技术。“junkcode”或“垃圾代码”是指在程序中插入一些无实际功能的代码，其目的是为了混淆反编译的结果，使得分析者更难以理解真正的程序逻辑。这种技术经常被用于恶意软件或病毒，以避免被安全研究者轻易地分析。

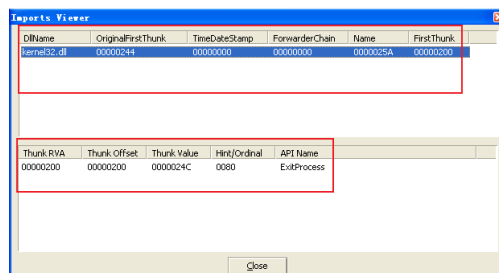


图 3.3: PEiD 对为脱壳的 Lab01-03.exe 的导入函数分析结果

如图3.3所示。此时如果继续使用 PEiD 查看其导入函数，只能看到唯一的导入函数为 kernel.dll 中的 ExitProcess，无法提供其它任何有用的信息。于是尝试使用 LinuxUnpacker 对其进行脱壳：

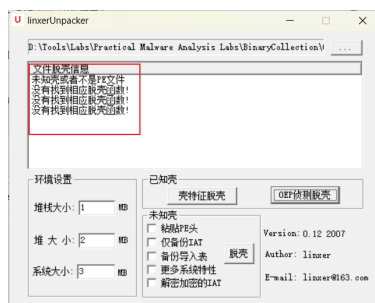
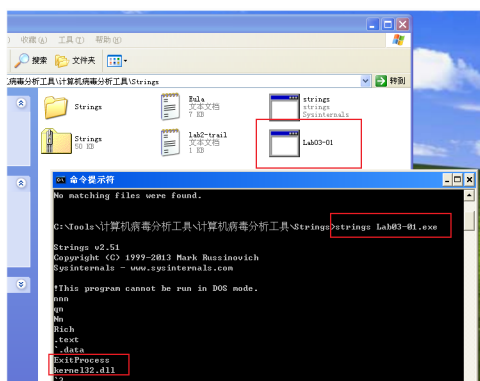


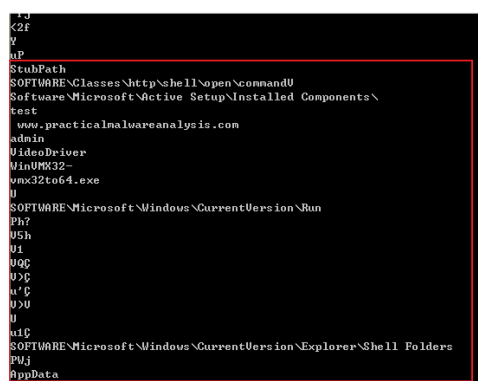
图 3.4: 尝试使用 LinuxUnpacker 对 Lab03-01.exe 进行脱壳

图3.4可以看到即使使用上次成功脱壳 Lab01-03.exe 的 LinxerUnpakcer 也无法进行脱壳，无法找到对应的脱壳函数。道高一尺魔高一丈，这里暂时不进行脱壳。没有其他的导入函数，很难真正去猜测程序的恶意目的。该程序很可能进行过加壳，其余的导入函数或者会在动态执行时才会被加入。

(c) 接下来查看其字符串：



分析结果 1



分析结果 2

图 3.5: Lab03-01.exe 的 strings 分析结果

图3.5展现了使用 strings 对病毒 Lab03-01.exe 的分析结果，可以看到左图中使用相应的命令进行分析。右图中展示了一些重点的字符串。其中有着注册表位置和域名等信息。大部分

信息都很清晰并且没有经过混淆处理。尽管此时的病毒是加壳过的，但我们仍然能够发现许多具有很强指示性和提示性的字符串。比如 WinVMX32,VideoDriver 以及 vmx32to64.exe。具体而言分析：

- **注册表键操作：**里面出现了 SOFTWARE 开头的诸多文件路径，其中 commandV 和 Installed Components 两个路径字符串指向注册表的位置，可能是病毒试图劫持或修改默认的 HTTP 处理程序，从而在用户尝试访问网页时执行恶意操作。同样的，Run 是 Windows 的启动位置，病毒可能是此处添加条目以确保每次 Windows 启动时都会执行。最后 Shell Folders 可能是病毒试图查找或修改某些特定的 shell 文件夹路径。
- **网络活动：**www.practicalmalwareanalysis.com。这一经典的系列彩蛋，再次出现。
- **系统和管理员操作：**admin 是病毒可能试图获取或检测管理员权限。VideoDriver 和 WinVMX32- 这些字符串可能与病毒试图与系统的视频驱动程序或虚拟机交互有关。最后，vmx32to64.exe 这可能是病毒的一个组件或它试图模拟的合法文件。
- 其它字符串：test 这可能是病毒的测试或调试功能。AppData 可能是提示病毒在 AppData 文件夹中存储或检索数据。

总的来说，我们通过简单的静态分析基于未脱壳的导入函数和字符串，可以初步揣测病毒的如下可能的恶意目的：

- 劫持或修改系统的默认 HTTP 处理程序。
- 在系统启动时自动执行。
- 与命令和控制服务器通信。
- 尝试获取管理员权限。
- 与系统驱动程序或虚拟机交互。
- 在用户的 AppData 文件夹中存储或检索数据。

然而，这只是基于最基本静态分析得来的揣测。为了确切地确定病毒的功能和目的，需要进行更深入的分析。

## 2. Q2: 这个恶意代码在主机上的感染迹象特征是什么？

回答：

(a) 为了能够进行安全且正确的病毒网络行为分析，首先使用 **Apatedns** 和 **Inetsim** 模拟网络环境：

通过 VMware 的特定网络设置，我们能够使多个虚拟机连接，形成一个虚拟的局域网络环境。在这其中，一个虚拟机执行恶意程序，而另一个则负责提供所需的网络功能。尽管如此，这个虚拟局域网并未与主机直接连接，主机继续与 Internet 保持连接，不会被恶意程序所干扰。

- 首先在 Windows XP 系统中通过编辑虚拟网络编辑器添加新的网络 VMnet2，并且修改子网 IP 处修改网络地址。并且修改完后要在 DHCP 设置中将 DHCP 获得地址范围也要修改一下，且接下来的 ip 配置均要在此范围内。



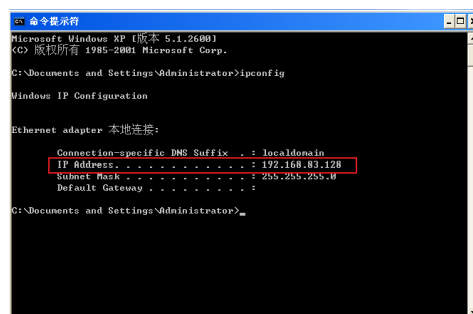
图 3.6: VM 虚拟网络编辑器配置修改

图3.6展示了新添加的虚拟网络 VMnet2,其配置的子网 IP 起始 IP 地址为 192.168.83.128,这对后面的配置至关重要。

之后将 Windows XP 系统添加到 VMnet2 中,也可以通过 ipconfig 命令查看此时的 XP 系统被分配到的 IPv4 地址。



添加至 VMnet2



查看 IP 地址

图 3.7: WindowsXP 网络配置查看

图3.7可以看到此时成功将 XP 系统加入到 VMnet2 后,通过在终端使用 ipconfig 命令可以看到此时已经成功地给 Windows XP 系统分配了 192.168.83.128 的地址。

- Kali 虚拟机: 同样地将 Kali 也加入到 VMnet2 中,并且在这里由于我们需要配置 Kali 将其 DNS 地址设置为自己的 IP 地址,因此还需要进行静态 IP 的配置:





添加至 VMnet2



配置 Kali 的静态 IP

图 3.8: Kali 网络配置修改

图3.8可以看到通过在 Kali 中找到高级网络配置，我们通过修改网络连接的方式修改其 IPv4 的网络设置，连接将方式改为手动。添加新的静态 IP 与 DNS 服务器地址。这里将 Kali 的静态 IP 地址以及 DNS 地址都被分配为与 Widows XP 系统紧邻，即 192.168.83.129。之后通过 ifconfig 命令在 Kali 中查看此时的网络情况：

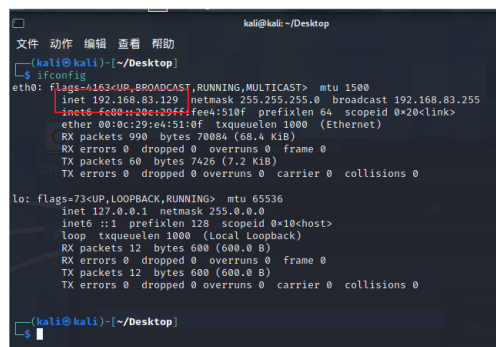


图 3.9: Kali 网络配置查看

图3.9可以看到此时的 Kali 网络被成功分配为了 192.168.83.129。

- 接下来尝试进行两台虚拟机的互通，通过 ping 命令依次来试图连接对方的地址。在这之前，首先将 Windows XP 系统的防火墙关闭，否则 Kali 虚拟机无法成功互通。



图 3.10: 关闭 Windows XP 系统防火墙

图3.10中可以看到此时的 XP 系统防火墙已经成功关闭，接下来尝试在两台主机中 ping

彼此：

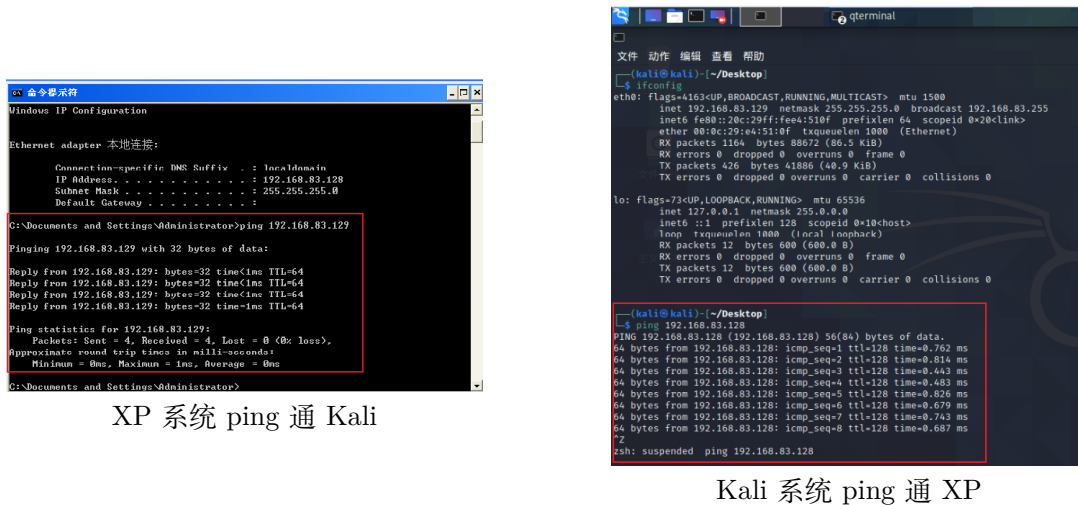


图 3.11: 虚拟机互通

图3.11中可以看到此时两台虚拟机都已经成功 ping 通。这样，两个虚拟机的网络设置已经成功地建立了连接。它们共同构建了一个独立且隔离的网络环境，就像一个沙盒一样。这种设计确保了一个安全的、隔离的场所，专门用于对恶意代码进行分析，防止其对外部环境产生任何潜在的影响或干扰。

- 配置 Inetsim：由于 Kali 自带 Inetsim 工具，因此我们只需要进到/etc/inetsim 文件夹修改 inetsim.conf 文件即可：

```
1 cd /etc/inetsim
2 vi inetsim.conf
```

然后依次修改 inetsim.conf 的配置文件中的 1 以下内容后就可以启动 DNS 服务：

- service\_bind\_address: 192.168.83.129
- dns\_default\_ip: 192.168.83.129
- redirect\_enabled yes
- redirect\_exclude\_port tcp:22
- redirect\_external\_address 192.168.83.129

配置好后通过 inetsim 命令在根目录下以 root 用户的身份启动服务：

```
Sub logfile '/var/log/inetsim/service.log' successfully created.
Debug logfile '/var/log/inetsim/debug.log' does not exist. Trying to create it...
Debug logfile '/var/log/inetsim/debug.log' successfully created.
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== InetSim main process started (PID 2506) ==
Session ID: 2506
Listening on: 192.168.83.129
Real Date/Time: 2023-10-03 07:58:08
Fake Date/Time: 2023-10-03 07:58:08 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 2510)
* irc_6667_tcp - started (PID 2520)
* time_37_tcp - started (PID 2525)
* echo_7_tcp - started (PID 2529)
* finger_79_tcp - started (PID 2532)
* quord_17_tcp - started (PID 2533)
* time_37_udp - started (PID 2526)
* ntp_123_udp - started (PID 2531)
* echo_7_udp - started (PID 2530)
* https_443_tcp - started (PID 2512)
* quord_17_udp - started (PID 2534)
* dummy_1_udp - started (PID 2538)
* daytime_13_udp - started (PID 2528)
* daytime_13_tcp - started (PID 2527)
* discard_9_tcp - started (PID 2535)
* chargen_19_udp - started (PID 2536)
* pop3s_995_tcp - started (PID 2516)
* discard_9_udp - started (PID 2532)
* ident_113_tcp - started (PID 2523)
* rtp_88_tcp - started (PID 2518)
* pop3_110_tcp - started (PID 2535)
* http_80_tcp - started (PID 2511)
* rtp_88_udp - started (PID 2519)
* dummy_1_tcp - started (PID 2537)
* syslog_514_udp - started (PID 2524)
* rtp_21_tcp - started (PID 2517)
* chargen_19_tcp - started (PID 2535)
* smtp_25_tcp - started (PID 2513)
* smtps_465_tcp - started (PID 2514)
* redirect - Error: Sorry, this module requires the perl mqueue-bindings!
done.
Simulation running
```

图 3.12: 启动 inetsim 服务

图3.12可以看到此时结尾导致了一项的错误提示，但由于 inetsim 缺少另一个服务导致，并不影响实验。正确显示 **Simulation Running** 就代表启动成功。

- 配置 ApateDNS: 首先如果直接在 Windows XP 系统中打开会报错,提示缺少.Net Framework, 因此首先在官网安装对应的版本即.Net Framework 2.0。



图 3.13: 安装.Net Framework

在成功安装了.NET Framework 后便可以启动 ApateDNS 了，在 DNS Reply IP 位置，输入要将请求重定向到的 IP 地址，也就是 Kali DNS 服务器的地址，接着点击 Start Server 以启动该工具。与此同时，打开 Process Explorer, Process Monitor 以及 WireShark 为第一次运行病毒进程做好充分预备工作。下载网络分析工具 Netcat, 并等待进一步使用。到此全部的准备工作完成。

- (b) **Regshot 注册表分析:** 接下来可以使用 Regshot 来探索病毒 Lab03-01.exe 可能的注册表行为，依次在运行病毒之前和之后点击 Regshot 的 shot 拍摄快照并保存：



图 3.14: Regshot 拍摄快照

在这之后，点击 compare 输出对比两次注册表前后的变化可以发现：

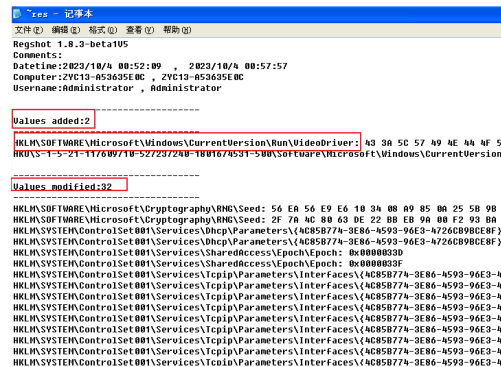


图 3.15: Regshot 分析结果 1

图3.15可以看到，通过快照比较分析得出该恶意软件新增一项注册表键值，通过将十六进制字符翻译为对应的英文内容我们可以得到：下添加了一个自启动项：

HKEY\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\VideoDriver

- 名字：VideoDriver
- 数据：C:\WINDOWS\System32\vmx32to64.exe

这个是电脑在启动会加载驱动程序的位置。因此由此初步推测，程序 vmx32to64.exe 在开机时会自启动。

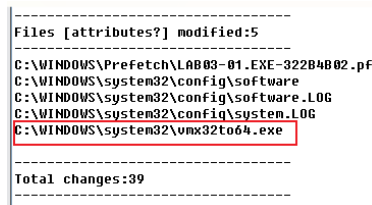


图 3.16: Regshot 分析结果 2

图3.16可以看到病毒确实对程序 vmx32to64.exe 进行了一定的更改。

- (c) **Process Explorer**: 打开之前打开的 Process Explorer 后, 点击进程中的 Lab03-01.exe 病毒后, 选中 **Handles** 查看句柄后, 可以看到进程打开的所有句柄。这包括文件、注册表键、同步对象等。这代表着病毒正在使用着哪些资源。

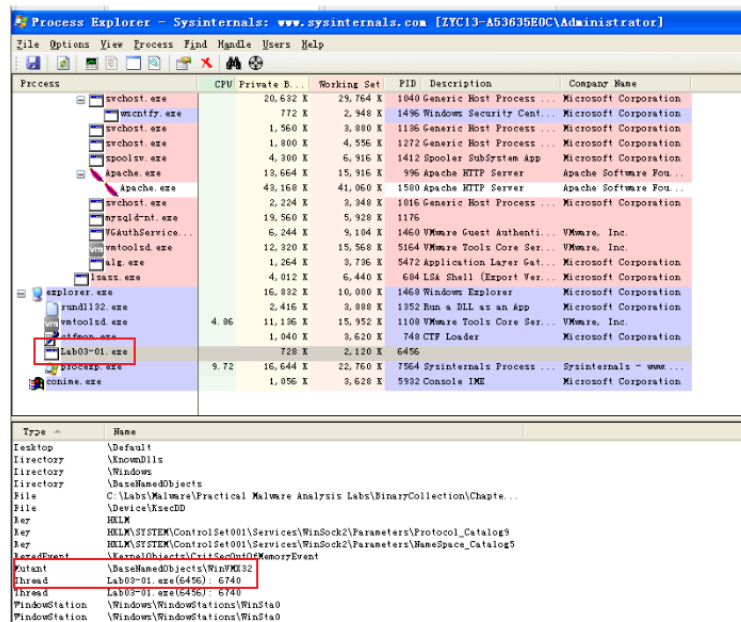


图 3.17: Process Explorer 查看 Lab03-01.exe 的句柄

图3.17可以看到此时 Lab03-01.exe 创建了一个互斥量 mutex 叫做 WinVMX32。考虑到其名称, 它可能与某种虚拟化技术或特定的恶意软件家族有关。接下来调整查看病毒进程的动态链接库 DLLs 的使用情况:

Name	Description	Company Name	Path
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
inetcfg.dll	Home Networking Configur...	Microsoft Corporation	C:\WINDOWS\system32\inetcfg.dll
imm32.dll	Windows XP IME32 API Clien...	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:\Lab03\Lab03-01.exe
locale.nls	Language Pack	Microsoft Corporation	C:\WINDOWS\system32\locale.nls
lpk.dll	Windows NT CRT DLL	Microsoft Corporation	C:\WINDOWS\system32\lpk.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 Windows	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
sechost.dll	Security Support Provider ...	Microsoft Corporation	C:\WINDOWS\system32\sechost.dll
setupapi.dll	Windows Setup API	Microsoft Corporation	C:\WINDOWS\system32\setupapi.dll
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	Unicode script p...	Microsoft Corporation	C:\WINDOWS\system32\usp10.dll
version.dll	Version Checking and File ...	Microsoft Corporation	C:\WINDOWS\system32\version.dll
winmr.dll	LDAP RnR Provider DLL	Microsoft Corporation	C:\WINDOWS\system32\winmr.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
ws2help.dll	Windows Socket 2.0 Helper ...	Microsoft Corporation	C:\WINDOWS\system32\ws2help.dll
wsbth.dll	Windows Sockets Helper DLL	Microsoft Corporation	C:\WINDOWS\system32\wsbth.dll
wshtcpip.dll			C:\WINDOWS\system32\wshtcpip.dll

图 3.18: Process Explorer 查看 Lab03-01.exe 的 DLLs

图3.18可以看到运行的病毒进程内以及两个已经被加载的 DLLs, 分别是 ws2\_32.dll 和 wshtcpip.dll, 这两个 dll 说明病毒具有着连接网络的功能。

- (d) **Process Monitor**: 接下来使用打开着的 Proces Monitor 寻找更多的信息, 通过使用过滤 Filter 功能。并且同时设置三项过滤器:

- ProcessName: Lab03-01.exe, 即指定寻找固定名字的进程。
- Operation: RegSetValue, 即指定寻找具有注册表信息修改功能的进程。

- Operation: WriteFile, 即指定寻找具有修改文件系统功能的进程。

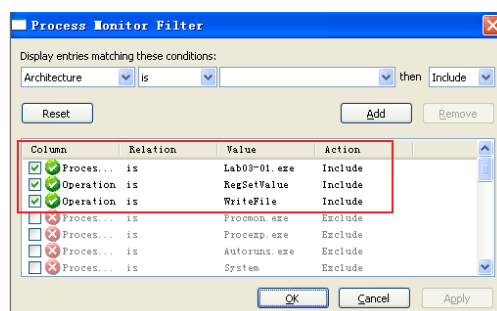


图 3.19: Process Monitor 的 Filter 设置

设置完后点击过滤, 可以看到原先的无数进程已经减少到了 10 条:

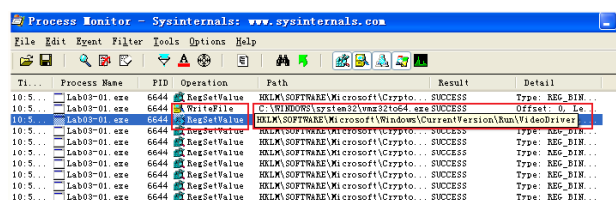


图 3.20: Filter 结果

图3.20可以看到此时有一条具有修改文件系统功能的记录, 而有 9 条具有着修改注册表信息的记录。

值得注意的是 HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed 是一个常见的噪音, 这是由于一个随机数的生成器会一直在注册表更新与一些软件相关的信息。接下来我们分别查看两种类型行为的详细信息, 首先是其对文件系统的修改:

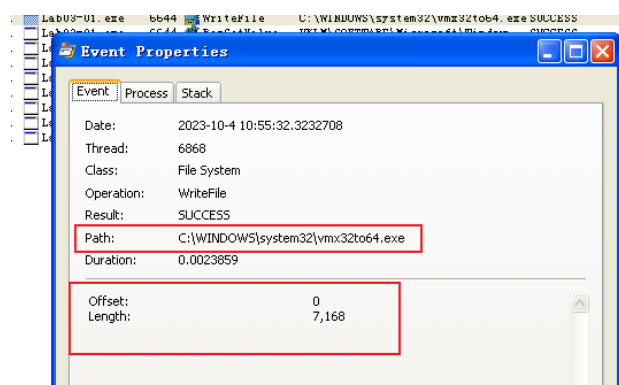


图 3.21: 查看 WriteFile 操作

图3.21可以看到它写入了 7168 个字节长度的文件到目录 C:\WINDOWS\system32\vmx32to64.exe, 这恰好与文件 Lab03-01.exe 的大小相同。之后通过 Windows 的资源管理器来到该目录下, 找到对应的文件 vmx32to64.exe, 使用工具 md5deep 查看其 hash 值:



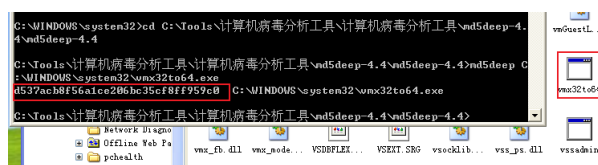


图 3.22: md5deep 查看 vmx32to64 的 hash 值

并将上述结果与 VirusTotal 结果对比：

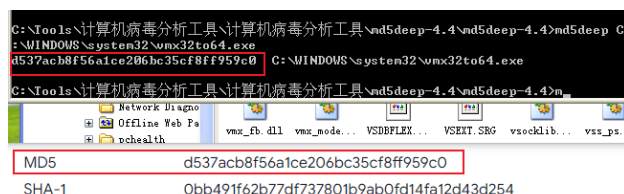


图 3.23: hash 值对比

图3.23可以看到这个新创建的文件 `vmx32to64.exe` 文件的 hash 值与 `Lab03-01.exe` 有相同的 MD5 的 hash 值。从这个可以推断出该病毒将自己复制到了 `vmx32to64` 这个位置，这个可以作为基于主机的恶意病毒的证据，便于后面进行 Yara 检测。

接下来对之前的注册表信息修改行为的分析，结合之前的 Regshot 分析结果可知：病毒在 Run 创建了一个名为 VideoDriver 的注册表键，来在系统启动时运行自己的副本 `vmx32to64`。综上所述：这个恶意代码创建了一个自己的互斥量叫做 `WinVMX32`，然后将自己的副本拷贝到了一个新建的文件：`System32` 下的 `vmx32to64.exe` 中。接下来它又通过创建 Run 下的注册表信息设置了自己的拷贝路径即 `vmx32to64`，这样每次系统启动时自己也会自启动运行。

### 3. Q3: 这个恶意代码是否存在一些有用的网络相关的病毒签名？如果存在，它们是什么？

回答：通过查看之前打开的 WireShark 和 ApatеDNS，可以找到很多有用的信息：

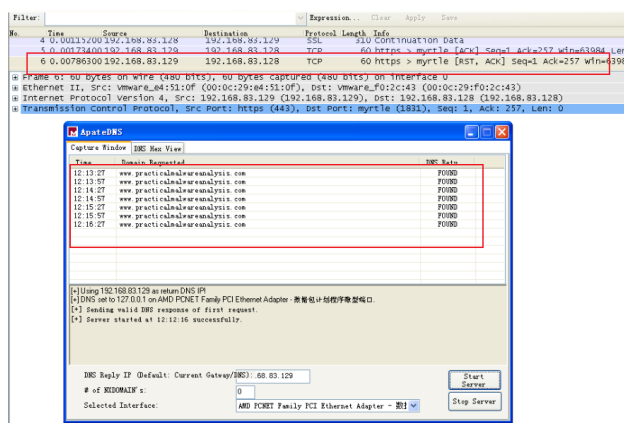


图 3.24: WireShark 和 ApatеDNS 网络信息截取

图3.24可以看到此时的 WireShark 和 ApatеDNS 都显示了许多网络相关的信息，其中再次出现了之前提到的网址彩蛋。接着将 ApatеDNS 重定向为 127.0.0.1 即 localhost 的地址后开启服务，使用 Netcat 分别监听在 80 和 443 这两个恶意代码经常使用的端口。

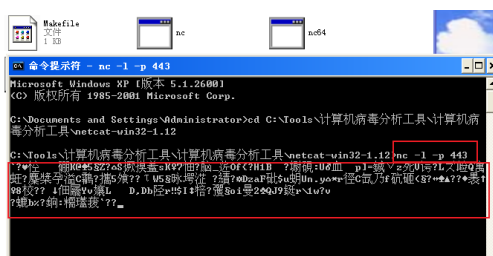


图 3.25: Netcat 截取网络信息

图3.25可以看到通过 Netcat 监听 443 端口时，会出现奇怪的乱码，多次实验结果均一样。并且监听 80 端口却没有异常。由此初步推测：该病毒确实存在 443 端口的网络行为，却数据似乎是随机的。同时通过 WireShark 分析可以发现数据包的大小是一样的，均为 256 字节。并且包含着随机数据，与通常在端口 443 上运行的 SSL 协议无关。

因此这个恶意代码确实存在有用的网络相关的病毒签名,这个签名是 particalmalwareanalysis。病毒在解析了这个经典的网址后，会发送一个大小为 256 字节的随机数据的数据包。

### 3.3 Lab03-02

#### 3.3.1 静态分析

在回答问题之前，先对病毒进行一系列常规的静态分析，首先是 VirusTotal 分析，可以看到和其它常规的病毒样本类似：

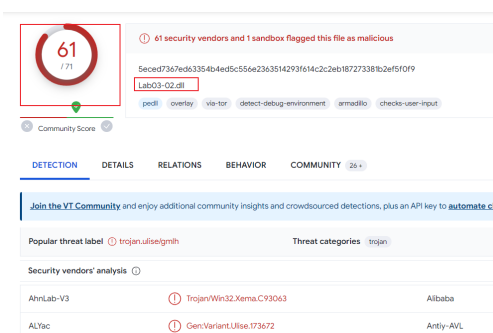


图 3.26: virusTotal 分析

然后分析其导入和导出函数，放入到 PEiD 中查看到其 PE 文件状态：

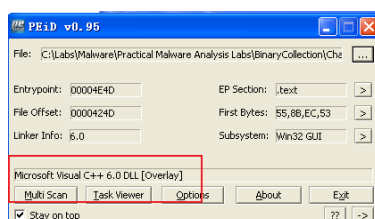


图 3.27: PEiD 分析

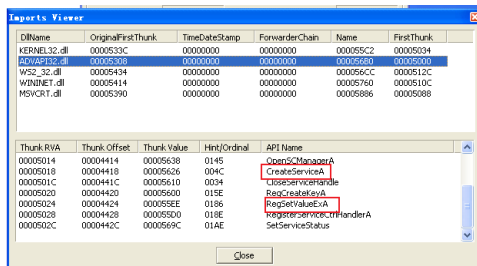
图3.27可以看到此时显示了“Microsoft Visual C++6.0 DLL[Overlay]”，这是代表该病毒并没加壳。



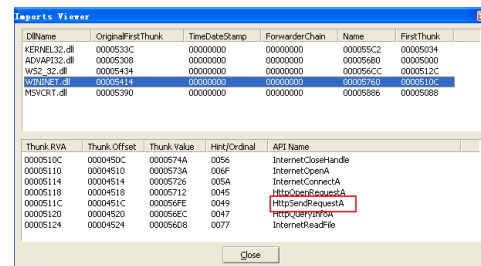
- Microsoft Visual C++6.0 DLL：这意味着该文件是使用 Microsoft Visual C++ 6.0 编译的。

Overlay：在 PE 文件格式中，Overlay 通常指的是文件结尾之后的数据。这些数据并不是 PE 文件结构的一部分，但它们仍然存在于文件中。Overlay 数据可以是任何东西，从简单的文本或图像到加密的数据或其他二进制信息。恶意代码的作者经常使用 Overlay 来存储加密或压缩的有效负载、配置数据或其他重要信息。

首先分析其导入函数：



导入函数 1



导入函数 2

图 3.28: PEiD 分析导入函数

可以看到一些有意思的导入函数：**CreateService** 是具有控制服务能力的函数；**RegSetValueEx** 是具有控制注册表信息能力的函数；**HttpSendRequest** 是具有网络能力的函数，意味着这个恶意代码会使用 HTTP 协议。接着分析其导出函数：

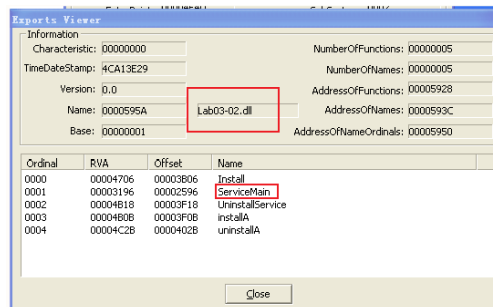


图 3.29: PEiD 分析导出函数

图3.29可以看到此时有 5 个导出函数，其中 **ServiceMain** 暗示了这个恶意代码可能会下载服务，并且需要使被安装作为服务才能够正确运行。这会对我们解决 Q1 有指导作用。接下来通过 strings 查看字符串：

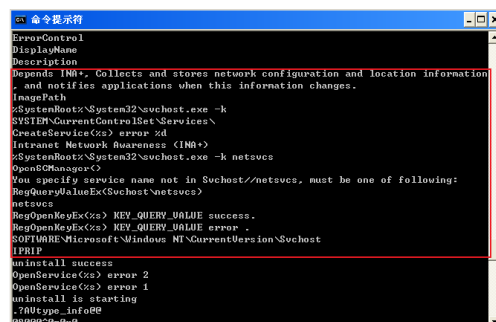


图 3.30: strings 查看字符串

图3.30展示了一些重要的字符串，其中包括注册表位置、域名、像 IPRIP 和 serve.html 这样的独特字符串，以及各种编码的字符串。还发现了 svchost.exe 的可执行程序。基本的动态技术可能会显示这些字符串和导入是如何被使用的。

初步推测：该软件可能会访问 practicalmalwareanalysis.com 网站以获取某些应用程序，并将其自身设置为一个服务。在 Windows 环境中，DLL 文件通常不能直接运行，因此肯定有某个执行文件来启动 Lab03-02.dll。从可见的字符串中，我们可以推测 svchost 很可能就是用来执行这个 dll 的 exe 程序。

### 3.3.2 实验问题

#### 1. Q1: 你怎样才能让这个恶意代码自行安装？

回答：在安装之前，首先为接下来需要进行的动态分析做好准备。打开 Process Explorer 来监视系统上运行的进程，使用 Regshot 拍下安装前的快照。

经过之前的静态分析，我们可以初步推测出该病毒需要使用导出的函数 installA 作为服务进行安装。于是使用专用于运行 dll 程序的 rundll32.exe 工具安装。

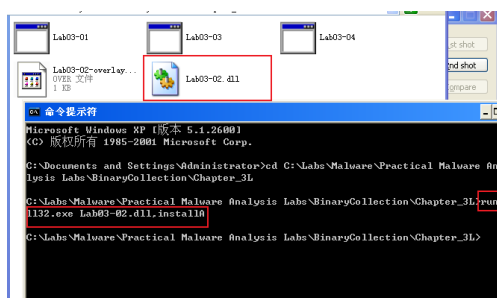
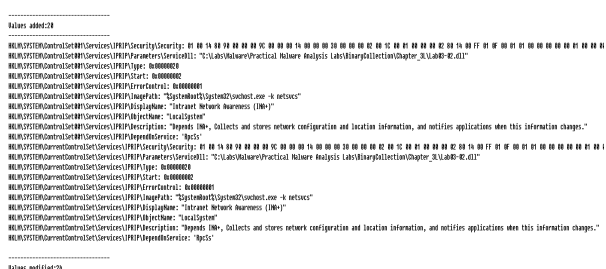


图 3.31: 使用 run32dll 运行 Lab03-02.exe

图3.31展示了使用在该 dll 目录下运行 rundll32.exe Lab03-02.dll,installA 安装。安装完成后，拍摄另一份快照后进行对比：



分析结果 1



分析结果 2

图 3.32: RegShot 分析

图3.32可以看到在 Keys added 中发现增加了一个 IPRIP 的服务，说明 Lab03-02.dll 将自己安装成一个 IPRIP 服务。并且 dll 需要一个可执行程序来执行它们。因为看到 ImagePath 被设置为 svchost.exe，这意味着恶意代码将在一个 svchost.exe 进程内部被启动。这些都与之前的猜测相符合。同时” Intranet Network Awareness (INA+) “是恶意代码执行时显示的名字。

#### 2. Q2: 安装之如何让恶意代码运行起来？

回答：先将其余的动态分析工具打开，这包括 Process Explorer 与 Process Monitor，还有 WireShark 和 ApatеDNS 都全部打开。

既然这个恶意代码是被安装成为了一个 IPRIP 服务，因此我们可以通过在命令行中启动网络服务的方式 net 命令令其运行：net start IPRIP

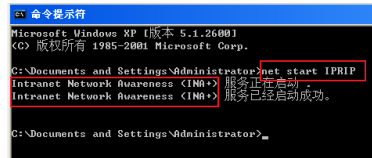


图 3.33: 运行 Lab03-02.dll

图3.33可以看到如之前猜测的病毒作为 IPRIP 服务成功运行，并且名字就是“Intranet Network Awareness (INA+)”。这代表着我们的恶意服务已经开始了。

### 3. Q3: 怎么可以找到这个恶意代码在哪个进程下运行的？

回答：首先打开 Process Exploer 的 Find DLL 功能，查找所需的进程的句柄或者 DLL。输入 substring 子串名字 Lab03-02.dll。

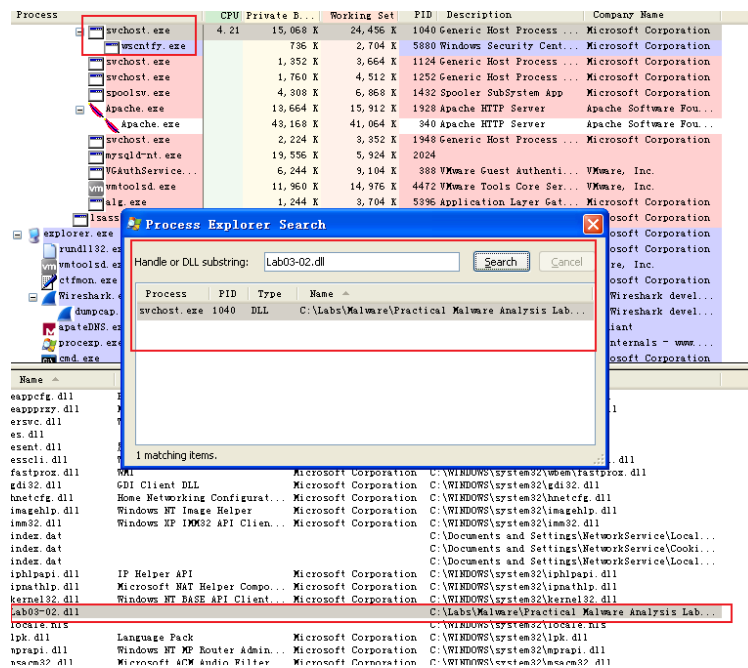


图 3.34: Process Explorer 查找病毒的运行进程

图3.34可以看到此时的 svchost.exe 正在运行，对应的进程号 PID=1040，双击查看后也看到了下面的 Lab03-02.dll 正在运行。

### 4. Q4: 你可以在 procmon 中设置什么样的过滤器，才能收集到这个恶意代码的信息？

回答：通过刚才找到的 PID 即进程 ID1040，在 procmon 中即可过滤找到：

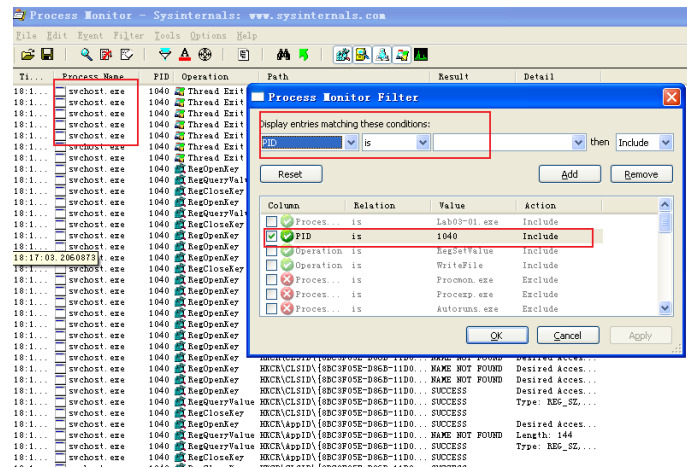


图 3.35: Procmon 过滤得到恶意代码信息

图3.35可以看到此时通过设置 PID=1040，利用过滤器就成功定位了运行的 svchost.exe，恶意代码的信息就在里面。

#### 5. Q5: 这个恶意代码在主机上的感染迹象特征是什么？

回答：在默认设置中，该恶意程序作为 IPRIP 服务进行安装。通过观察注册表中的信息 DisplayName 和 Description。具体而言：

- DisplayName: Intranet Network Awareness (INA+)
- Description: 基于 INA+，它负责搜集和保存网络的配置与位置数据，并在这些数据有所变动时向应用发出通知。

上述这些可以作为明显的基于主机的感染迹象特征，用于对病毒的 Yara 检测识别。

另外还可以发现该程序在注册表的 HKLM 的 ServiceDll 路径下进行自我部署以保持其持续活跃，其值定为%CurrentDirectory%\Lab03-02.dll。

#### 6. Q6: 此恶意代码是否存在一些有用的网络特征签名？

回答：首先使用之前搭建好的网络沙盒的虚拟网络，ApateDNS 服务器开启后捕获相应的 DNS 请求：

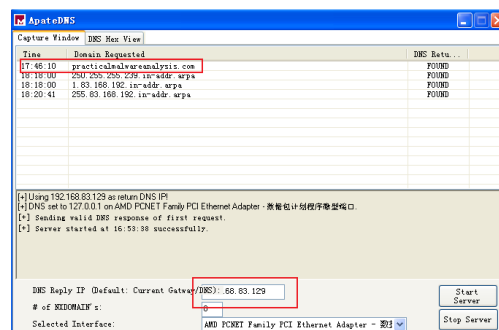


图 3.36: ApateDNS 拦截 DNS 请求

图3.36可以看到此时在运行了 Lab03-02.dll 后显示拦截了域名请求“parcticalmalwareanay-sis.com”，这和之前的字符串匹配上了。这是病毒的网络特征签名之一，但并不足够独特来进

行唯一性地 Yara 检测识别。

接着将 ApateDNS 重定向为 127.0.0.1 即 localhost 的地址后开启服务，使用 Netcat 监听在 80 恶意代码经常使用的端口，这也是因为之前观察到的字符串相关的 HTTP 请求。启动了 IPRIP 服务后，80 端口的监听出现了结果：

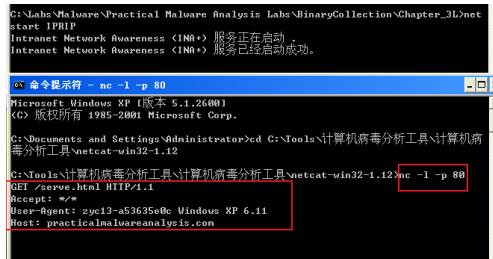


图 3.37: Netcat 拦截 HTTP 请求

图3.37可以看到此时的 Netcat 在启动了 IPRIP 服务后成功检测到了如下信息：

```

1      GET /server.html HTTP/1.1
2      Accept: */*
3      User-Agent: zyc13-a53635e0c Windows XP 6.11
4      Host: practicalmalwareanalysis.com
    
```

这证明恶意代码通过端口 80 执行了一个 HTTP GET 请求，并且多次执行后数据没有变化。

由此我们得到了总共三个有用的网络特征签名，这些都会对我们的 Yara 检测起重要的指导作用：

- DNS 请求：parcticalmalwareanaysis.com
- HTTP 的 GET 请求：serve.html
- 恶意代码分析虚拟机名称 Useragent：zyc13-a53635e0c Windows XP 6.11（除去前半部分虚拟机名称的后半部分）

## 3.4 Lab03-03

### 3.4.1 实验问题

1. Q1: 当使用 Process Explorer 进行监控的时候，你发现了什么？

回答：首先启动 Process Explorer 和 Process Monitor，在 Process Monitor 中，在执行任何动态分析前，为了防止事件迅速流动，于是关闭事件捕获，同时只启用默认的过滤器并重置。直到已经准备好了进行实验再开启事件捕获：





图3.40可以看到此时引入了一个 `practicalmalwareanalysis.log` 文件以及一些类似于键盘命令的字符串比如 `[ENTER]`，而在 Image 模式下并未出现以上字符串。这些显然不是正常的 Windows `svchost.exe` 磁盘文件中会出现的字符串。所以可以初步推测此程序是检测用户敲击的键盘，作为键盘记录器。

### 3. Q3: 这个恶意代码在主机上的感染特征是什么？

回答：接下来为了验证关于该病毒是一个键盘记录器，首先打开 `procmon`，使用过滤器分别使用如下三个筛选条件：

- PID: 6304, 就是之前找到的 `svchost.exe` 的进程号。
- CreateFile
- WriteFile

打开一个记事本敲一下内容（这里是彩蛋）：

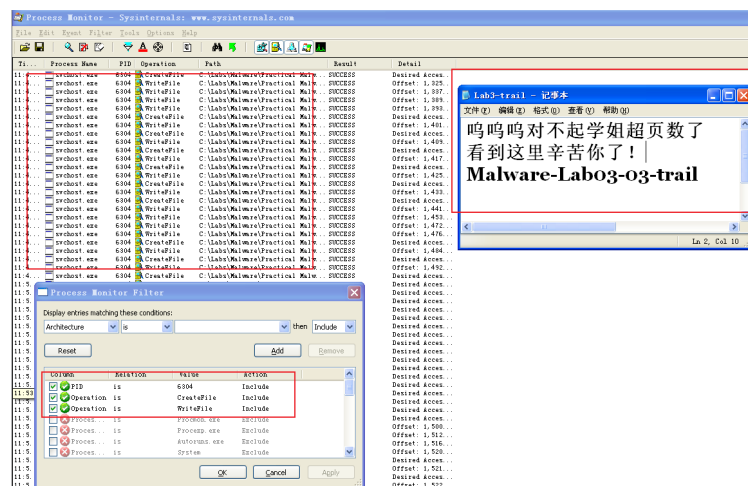


图 3.41: Procmon 分析病毒行为

图3.41可以看到通过上述三个条件能够过滤出很多事件，此时在敲内容的过程中，能够明显发现突然出现一堆事件，这些都是 `svchost.exe` 的 `CreateFile` 和 `WriteFile` 正在写入 log 文件。来到 Chapter3 目录下打开该日志文件：

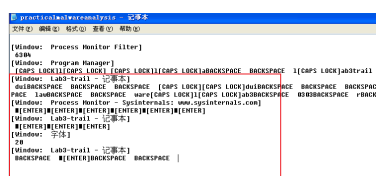


图 3.42: log 日志内容

这些都是刚才的一些键盘操作比如 `[ENTER]` 等，进一步验证了我们的猜想。因此，恶意软件创建了日志文件 `practicalmalwareanalysis.log` 就可以作为主机上的感染特征便于进一步识别检测。

### 4. Q4: 恶意代码的目的是什么？

回答：综上所述，我们可以总结出恶意代码的目的是对 `svchost.exe` 执行进程替换并启动键盘记录器，来记录用户的键盘行为。

## 3.5 Lab03-04

### 3.5.1 静态分析

初步进行一些简单的静态分析：

#### 1. VirusTotal 分析：

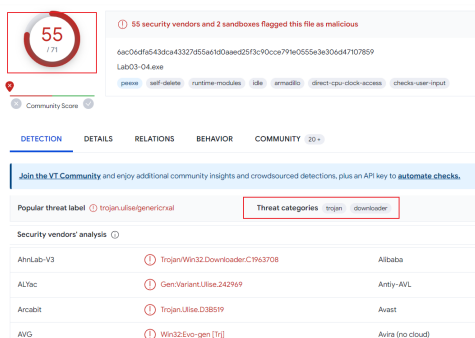
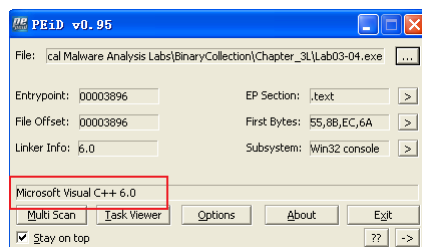
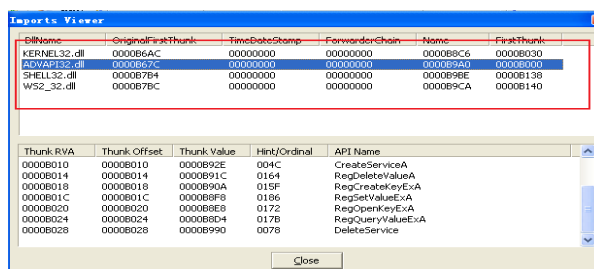


图 3.43: VirusTotal 分析

#### 2. PEiD 的加壳和导入函数分析：



加壳分析



导入函数分析

图 3.44: PEiD 分析

图3.44可以看到 Lab04-01 没有进行加壳处理，在导入函数中我们能够发现恶意代码导入了一些网络功能、服务操作功能和注册表操作功能。例如 `CreateService` 和 `RegCreateKeyEx` 等。

#### 3. strings 的字符串分析：



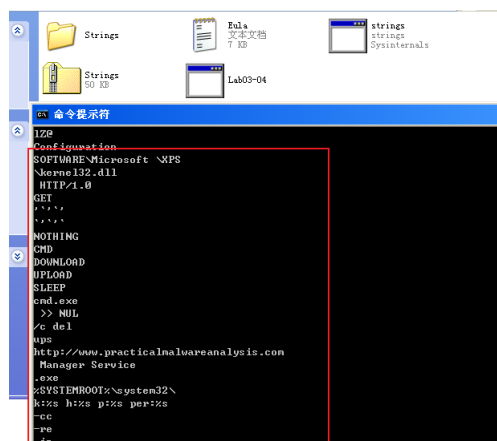


图 3.45: strings 分析

图3.45可以发现一些标志性的字符串，如经典的域名网址和注册表路径 SOFTWARE\Microsoft\XPS。除此之外，类似于 DOWNLOAD、UPLOAD 这样的词，再加上 HTTP/1.0，这似乎表明该恶意代码有着 HTTP 后门功能。而 -cc、-re 和 -in 这些字符串很可能是命令行的参数，比如 -in 或许意味着安装。

### 3.5.2 实验问题

### 1. Q1: 当运行文件的时候, 会发生什么?

回答：接下来进行动态分析，双击运行 Lab03-04.exe：

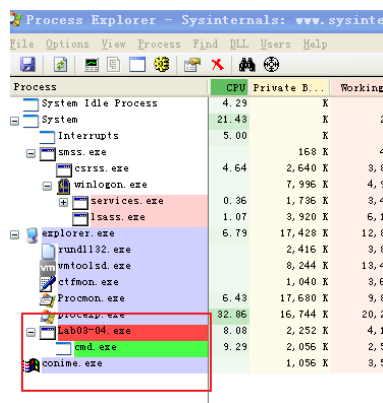


图 3.46: 运行 Lab03-04.exe 瞬间

图3.46可以看到文件瞬间弹出一个 cmd 命令行后就消失了，同时原来目录下的 Lab03-04.exe 程序消失了。但过程还是被我手快截下图来了嘿嘿。

## 2. Q2: 是什么原因造成动态分析无法有效实施？

**回答:** 使用 procmon 进一步分析, 使用过滤功能首先选定条件为 ProcessName=La03-04.exe, 选择右上角的只监控进程信息:

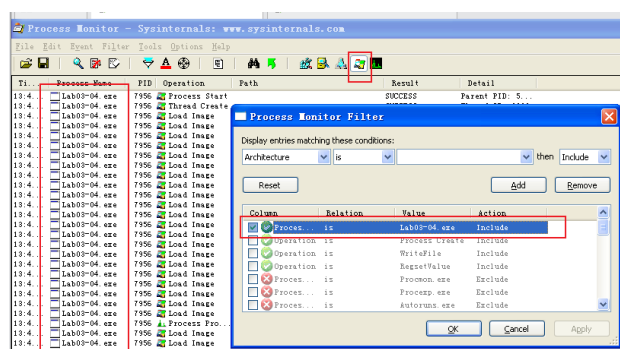


图 3.47: Procmon 监控

图3.47可以看到此时能够成功地筛选出来许多名为 Lab03-04.exe 的进程，但没有出现之前静态分析出的一些对文件系统和注册表进行修改的行为线程，类似于”WriteFile”和”RegSetValue”。但是通过仔细观察，还是发现了和可能和进程神秘消失有关的信息：

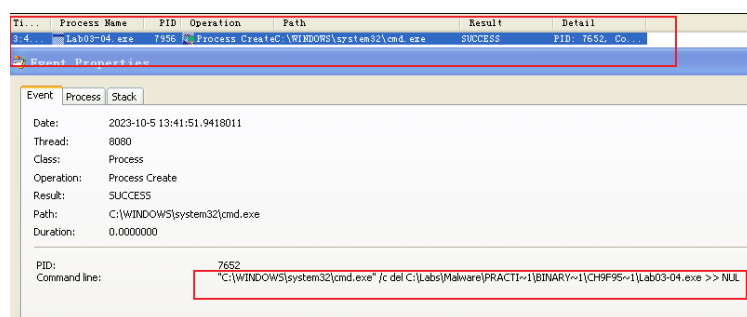


图 3.48: Process Create 进程

图3.48可以看到一个进程 Process Create，其中 “C:\WINDOWS\system32\cmd.exe” /c del Z:\Lab03-04.exe » NUL” 信息告诉我们病毒是创建了一个 cmd.exe 进程并且删除了原有的进程。

### 3. Q3：是否存在别的方式运行该程序？

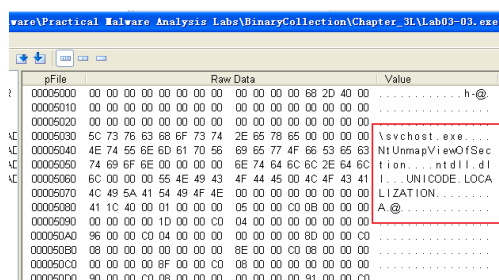
回答：在尝试使用-cc、-re 和-in 这些字符串作为命令行的参数，但都会导致病毒删除自身。仍然没有办法运行该程序。通过查看课本发现动态分析无法解决，因此这里留一个坑等后面来填。

## 3.6 Yara Detection

### 3.6.1 Sample 提取

利用课程中老师提供的 Scan.py 程序，将电脑中所有的 PE 格式文件全部扫描：





pFile	Raw Data	Value
00005000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....h-@
00005010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00005020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00005030	5C 73 76 63 88 6F 73 74 2E 65 78 65 00 00 00 00	svchost.exe...
00005040	4E 74 55 6E 6D 61 70 56 69 65 77 4F 66 53 65 63	NtUnmapViewOfSec...
00005050	74 69 6F 6E 00 00 00 00 6E 74 64 6C 6C 2E 64 6C	tion...ntdll.dll
00005060	6C 00 00 00 55 4E 49 43 4F 44 45 00 4C 4F 43 41	...UNICODE: LOCA...
00005070	4C 49 5A 41 54 49 4F 4E 00 00 00 00 00 00 00 00	LIZATION:...
00005080	41 1C 40 00 01 00 00 00 05 00 00 C0 0B 00 00 00	A@...
00005090	00 00 00 1D 00 00 C0 04 00 00 00 00 00 00 00 00	.....
000050A0	96 00 00 C0 04 00 00 00 00 00 00 80 00 00 C0	.....
000050B0	08 00 00 00 00 00 00 00 8E 00 00 C0 08 00 00 00	.....
000050C0	00 00 00 8F 00 00 C0 08 00 00 00 00 00 00 00	.....
000050D0	90 00 00 C0 08 00 00 00 00 00 00 91 00 00 C0	.....

图 3.51: Lab03-03.exe 的字符串

图3.51可以看到 Lab03-03.exe 几乎唯一特殊的字符串就是这些，而 svchost 并不能单独作为一个签名，因为相信很多其他的服务类型的进程也会有这个。故这里尽可能先使用这些特征：“svchost”，“NtUnmapViewOfSection”，“LOCALIZATION”，“A@”。

4. **Lab03-04.exe:** 域名“practicalmalwareanalysis”，命令行“cmd.exe”与“>> NUL”自我消失的退出。

对上述病毒条件均取并集，并使用 wide ascii nocase 修饰符避免特殊情况，在宿主机的 sample 文件夹中搜索。具体 Yara 规则如下：

```

1 rule Lab3
2 {
3 meta:
4   description = "Lab3:Yara Rules"
5   date = "2023/10/5"
6   author = "ErwinZhou"
7 strings:
8   $clue1 = "vmx32to64.exe" wide ascii nocase // feature:lab03-01.exe
9   $clue2 = "serve.html" wide ascii nocase // feature:lab03-02.dll
10  $clue3 = "IPRIP" wide ascii nocase // feature:lab03-02.dll
11  $clue4 = "practicalmalwareanalysis" wide ascii // feature:lab03-01.exe &
12    lab03-02.dll & lab03-04.exe
13  $clue5 = "svchost" wide ascii nocase // feature:lab03-02.dll & lab03-03.exe
14  $clue6 = "NtUnmapViewOfSection" wide ascii nocase // feature:lab03-03.exe
15  $clue7 = "cmd.exe" wide ascii nocase // feature:lab03-04.exe
16  $clue8 = ">> NUL" wide ascii nocase // feature:lab03-04.exe
17  $clue9 = "UNICODE" wide ascii nocase // feature:lab03-03.exe
18  $clue10 = "LOCALIZATION" wide ascii nocase // feature:lab03-03.exe
19 condition:
20   ($clue4 and $clue1) or //lab03-01.exe
21   ($clue4 and $clue2 and $clue3) or //lab03-02.exe
22   ($clue5 and $clue6 and $clue9 and $clue10) or //lab03-03.exe
23   ($clue4 and $clue7 and $clue8) //lab03-04.exe
24 }
```

### 3.6.3 Yara 规则执行效率测试

使用 Lab1 中同样的 Python 代码进行测试查看效果和执行效率:

```
File 'Lab03-01.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab03-02.dll' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab03-03.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab03-04.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab09-01.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab12-02.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab16-01.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab17-03.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'MRT.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'qbcore.dll' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'services.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'software_reporter_tool.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'tttdrecord.dll' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'WebMeetWebCore.dll' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
Program runtime: 136.4225468635559 seconds.
Total files scanned: 26385
Total files matched: 14
```

图 3.52: Yara 检测

图3.52可以看到上述 Yara 规则可以成功检测出来 Lab3 的全部四种病毒, 并且用时较短, 只有 136.42s。但与此同时, 也能看到许多不属于 Lab3 的甚至其它 PE 文件也被识别出来了, 这很有可能是为了识别 Lab03-03.exe 这个特殊病毒而采取的冒险举措导致的。可以进一步进行如下优化:

- **时间效率:** 如果能够确定对应的字符串大小写等情况, 就可以不使用修饰符 wide ascii nocase 等加快效率, 同时找到更特殊的病毒签名也可以实现。这需要更深层次的分析。
- **测试准确性:** 如果希望能够尽可能减少错误检测其它 Lab 或者 PE 文件导致的准确率降低, 可以考虑忽视 Lab03-03.exe 的特殊情况, 这样没有那些莫能两可的字符串, 可以将测试结果缩小在 Lab 病毒之中:

```
1 rule Lab3
2 {
3 meta:
4     description = "Lab3:Yara Rules"
5     date = "2023/10/5"
6     author = "ErwinZhou"
7 strings:
8     $clue1 = "vmx32to64.exe" wide ascii nocase // feature:lab03-01.exe
9     $clue2 = "serve.html" wide ascii nocase // feature:lab03-02.dll
10    $clue3 = "IPRIP" wide ascii nocase // feature:lab03-02.dll
11    $clue4 = "practicalmalwareanalysis" wide ascii // feature:lab03-01.exe &
12           lab03-02.dll & lab03-04.exe
13    $clue7 = "cmd.exe" wide ascii nocase // feature:lab03-04.exe
14    $clue8 = ">> NUL" wide ascii nocase // feature:lab03-04.exe
15 condition:
```

```
15 ($clue4 and $clue1) or //lab03-01.exe
16 ($clue4 and $clue2 and $clue3) or //lab03-02.exe
17 ($clue4 and $clue7 and $clue8) //lab03-04.exe
18 }
```

```
File 'Lab03-01.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab03-02.dll' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab03-04.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab09-01.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
File 'Lab16-01.exe' in path 'D:\Tools\Virus Detection\Yara\yara64\Laboratory\sample' matched YARA rule(s):
Rule: Lab3
Program runtime: 127.25465369224548 seconds.
Total files scanned: 26385
Total files matched: 5
```

图 3.53: 改进 Yara 规则

图3.53可以看到此时的运行时间得到了一定的优化，同时几乎可以保证病毒样本都在 Lab 中，具有更高的准确率。

## 4 实验结论及心得体会

### 4.1 实验结论

本次实验，通过使用 strings,PEiD 等静态工具，重点使用 ApateDNS、Inetsim、Netcat、Process Explorer、Procmon、Regshot、WirsShark 等，成功地对虚拟网络环境进行了搭建，还结合静态和动态分析方法，对 Lab3 中的四个病毒进行了全面的分析，并回答了课本中的问题，最后利用了之前的发现编写了 Yara 规则也检测出来了 Lab3 的全部病毒。**总的来说，实验非常成功。**

### 4.2 心得体会

本次实验，我收获颇丰，这不仅是课堂中的知识，更是我解决许多问题的能力，具体来说：

1. 首先我学习到了将课堂中学习的动态分析工具如 ApateDNS、Inetsim、Netcat、Process Explorer、Procmon、Regshot、WirsShark 等实际使用，对病毒进行分析，投入实践亲身体会加深了我的理解。
2. 其次我学会了自己搜集资料，结合计算机网络的知识对虚拟网络环境进行了搭建，构造了网络的沙盒。
3. 最后我还培养了许多病毒分析人员重要的素质：如虚拟环境，拍摄快照等，这都是在本次实验的不断试错中自己摸索养成的习惯。

**总的来说，本次通过亲自实验让我感受到了很多，也加深了我对病毒动态分析的能力，培养了我对病毒分析安全领域的兴趣。我会努力学习更多的知识，辅助我进行更好的病毒分析。**