南開大學

恶意代码分析与防治技术实验报告

Lab3

学院：网络空间安全学院

专业：信息安全、法学

学号：2113203

姓名：付政烨

班级：信安法班

摘要

本次的四个实验涵盖了恶意软件分析的多个关键方面，从基本的静态和动态分析技术到更深入的行为分析。在 Lab 3-1 中，我们学习了如何通过查看 PE 文件结构和字符串以及动态分析工具来了解恶意软件的功能和行为，包括文件操作、注册表修改和网络通信。Lab 3-2 进一步展示了如何将恶意软件作为 Windows 服务安装，并通过进程监视和注册表分析来跟踪其持久性。此外，我们观察到恶意软件与远程服务器的通信，这强调了网络分析的重要性。在 Lab 3-3 中，我们深入研究了恶意软件的隐藏技巧，包括对系统进程的替代操作和内存与磁盘镜像的比较。这些技术对于检测恶意活动至关重要。最后，在 Lab 3-4 中，我们面对了一种具有自毁特性的恶意软件，强调了恶意软件分析的挑战性和复杂性，以及需要深入的配置和分析才能揭示其真正的威胁。

关键字：动态分析技术；　静态分析技术；　恶意软件分析

目录

一、 实验目的

本实验旨在教授我们有关恶意软件分析的基本原理和技术。通过一系列不同的实验任务，我们将学习如何静态和动态分析恶意软件，以便理解其功能、行为和潜在威胁。在实验 3-1 中，我们首先通过查看 PE 文件结构和字符串，识别了恶意软件的特征，并使用动态分析工具监视了进程、文件系统和注册表的变化，以深入了解其行为。实验 3-2 聚焦于分析恶意服务的安装与持久性，我们学会了如何将恶意软件作为 Windows 服务安装，通过监视系统进程和注册表来检测其存在，并通过网络通信分析了其与远程服务器的交互。实验 3-3 涉及恶意软件的内存与磁盘分析，我们了解了如何检测恶意软件替换系统进程的行为，以及通过比较内存和磁盘镜像来识别不一致性。最后，在实验 3-4 中，我们面对具有自毁特性的恶意软件，强调了分析的挑战性和复杂性，促使我们深入思考并解决问题。通过这些实验，我们将培养恶意软件分析的关键技能，包括静态和动态分析、持久性检测、网络通信追踪以及发现隐藏功能，为未来的安全工作和研究提供坚实基础。这一系列实验任务旨在加深我们对恶意软件的理解，提高在网络安全领域的技术能力。

二、 实验原理

(一) Lab 3-1

上述实验涵盖了对恶意软件的基本静态和动态分析技术的应用。首先，通过查看 PE 文件结构和字符串，我们发现恶意软件仅导入了 kernel32.dll，并且存在一些有趣的字符串，如注册表路径、域名以及关键字。动态分析方面，我们配置了一系列工具，包括 Process Explorer、Procmon、ApateDNS、Netcat 和 Wireshark。通过 Process Explorer，我们观察到恶意软件创建了名为 WinVMX32 的互斥体，同时动态加载了一些 DLL，表明它具备网络功能。Procmon 工具被用来监控文件系统和注册表的变化，从中我们发现恶意软件将自身复制到 C:\WINDOWS\system32\vmx32to64.exe ，并创建了启动项。最后，通过网络分析工具，如 ApateDNS 和 Netcat，我们检查了 DNS 请求和网络通信，发现恶意软件似乎在端口 53 上发送随机数据包。这些分析技术帮助我们了解了恶意软件的功能和行为，包括文件操作、注册表修改以及网络通信，有助于进一步的分析和对策制定。

(二) Lab 3-2

本实验涉及对恶意代码的分析。首先，通过运行“rundll32.exe”实用程序并执行恶意软件 DLL 文件（如 Lab03-02.dll）中的特定导出函数，将恶意软件作为 Windows 服务安装，使其能够在系统启动时自动运行。接着，使用“net”命令启动恶意软件服务，其服务名称为“IPRIP”。为了确定运行恶意软件服务的进程，可以使用“Process Explorer”工具，识别托管恶意软件服务的 svchost.exe 进程，通过服务名称或 Lab03-02.dll DLL 来作为指示器。然后，通过“Procmon”监视系统活动，通过根据从“Process Explorer”获得的进程 ID（PID）来筛选与恶意软件进程相关的事件。恶意软件确保持久性的方式是将自身添加到 Windows 注册表中，使用服务名称“IPRIP”，显示名称为“Intranet Network Awareness (INA+)”，并提供特定的描述。注册表中的位置为 HKLM\SYSTEM\CurrentControlSet\Services\IPRIP\Parameters\ServiceDll ，其中指定了 DLL 文件的路径。最后，恶意软件通过解析域名 practicalmalwareanalysis.com 并通过 HTTP 在端口 80 上与该主机通信。它发送 GET 请求以获取“serve.html”，并将 User-Agent 标头设置为“%ComputerName% Windows XP 6.11”，这表明恶意软件试图从远程服务器获取其他指令或数据。

(三) Lab 3-3

本实验涉及了恶意软件分析和检测的关键原理。首先,恶意软件执行了对系统进程 svchost.exe 的替换操作,以在系统中运行自己的代码。其次,通过比较内存镜像和磁盘镜像,发现它们不一致,内存镜像包含了特定字符串如"practicalmalwareanalysis.log" 和"[ENTER]", 而磁盘镜像则没有这些内容,这是一种检测恶意活动的技术。第三,恶意软件还创建了名为"practicalmalwareanalysis.log" 的日志文件, 通常用于记录受害系统上的活动。最后, 恶意软件通过进程替换操作, 启动了一个键盘记录器, 以窃取用户的键盘输入信息, 可能包括敏感信息如密码。

(四) Lab 3-4

本实验涉及对一种恶意软件的行为分析。当双击运行该恶意软件时, 它会立即删除自身, 表现出自毁特性。研究人员怀疑运行该软件可能需要附加命令行参数或缺少必要组件, 并尝试使用字符串列表中的参数进行测试, 但未获成功。这暗示恶意软件可能需要特定的配置或更深入的分析才能揭示其真正的功能和威胁。这个实验强调了恶意软件分析的挑战性和复杂性。

三、 实验过程

(一) Lab 3-1

1. What are this malware's imports and strings?

通过 IDA Pro 对 Lab03-01.exe 进行分析后, 我们获取了导入表和字符串的详细信息, 结果如下图1,2所示。

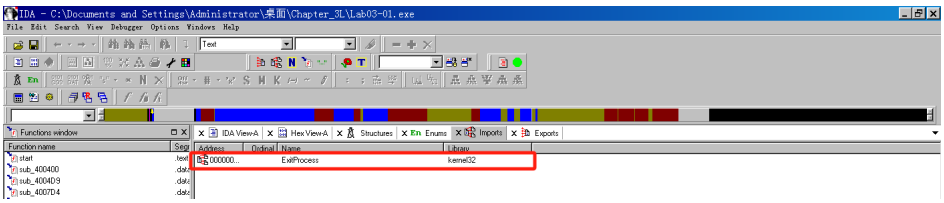


图 1: 导入函数

在图1中, 我们可以观察到仅存在一个名为 "kernel32.dll" 的动态链接库 (DLL) 的引用, 而且仅导入了一个函数, 即 "exitprocess"。这个结果引发了我们的猜测, 即该程序可能经过了一层壳的保护, 其中导入函数将在程序运行时进行解析。

这种情况下, 程序通常会在运行时动态加载其他依赖的 DLL, 以保护其内部逻辑免受逆向工程分析的威胁。因此, 我们需要进一步的分析和解密以了解程序的真实行为和内部结构, 以克服壳的保护机制, 揭示程序的潜在功能和漏洞, 还原程序的原始状态和功能。

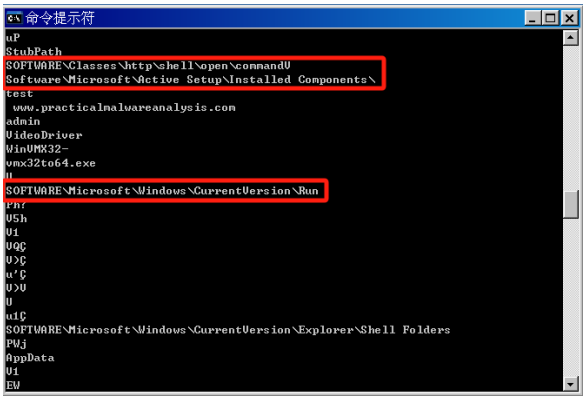


图 2: 字符串

接下来,我们将对以下关键信息进行字符串分析。在图2中,我们观察到了一些具有潜在重要性的注册表键值路径, 其中包括 `SOFTWARE \Classes\http\shell\vopen\commandV` 和 `Software\Microsoft\Active Setup\Installed Components` 等。此外，我们还注意到存在与系统启动相关的注册表键值，位于 `SOFTWARE\Microsoft\Windows\CurrentVersion\Run`。这些观察结果指示着对注册表的修改，旨在实现系统的自启动功能。对于这些注册表位置的修改，我们需要特别关注，因为它们可能与系统的启动行为和程序自动运行有关。

2. What are the malware’s host-based indicators?

虽然虚拟机可用作沙箱以将计算机病毒与外部环境隔离开来，但许多计算机病毒需要与网络连接以触发其特定行为。因此，为了模拟一个尽可能真实的网络环境，我们采取了以下步骤。首先，我们使用了 “Process Monitor” 工具在虚拟机中清除了所有系统事件记录。接下来，我们通过 “Process Explorer” 配置了虚拟网络环境，以模拟网络连接。为了进一步模拟网络行为，我们使用了 “ApateDNS” 工具，将 DNS 解析的 IP 地址重定向到虚拟机内部的回环地址 127.0.0.1。最后，我们开启了”netcat” 工具以监听特定端口的信息传输。

在这一过程中，我们运行了可能包含计算机病毒的 Lab03-01.exe 程序，并使用 “Process Explorer” 来监视系统中的各个进程，得到了如下图3的结果:

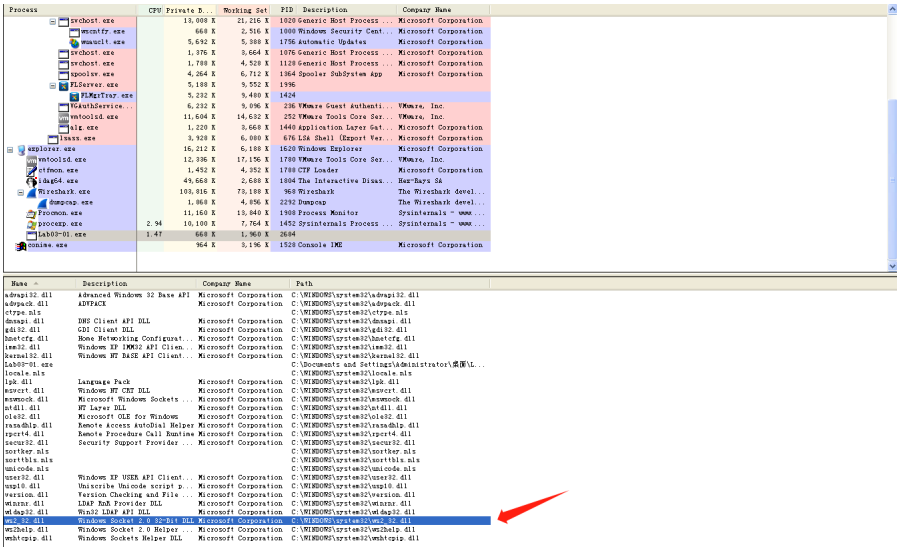


图 3: Process Explorer 监测结果

在研究结果中，我们观察到存在一个名为 “ws2_32.dll” 的应用程序接口。这一接口通常用于支持互联网网络功能的应用程序。因此，从这一观察可以推测，与该 exe 程序相关的软件可能具备与互联网连接相关的功能或者能力。

随后，我们启动了 Process Monitor 以深入进行分析。为了排除无关信息的干扰，我们进行了必要的过滤器配置，具体设置如图4所示。这一步操作产生了如图5所示的分析结果。

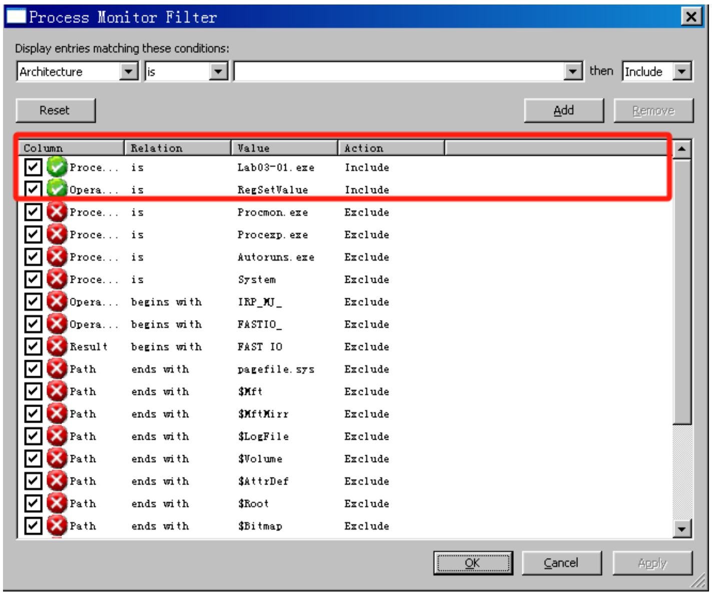


图 4: Process Monitor 过滤器设置

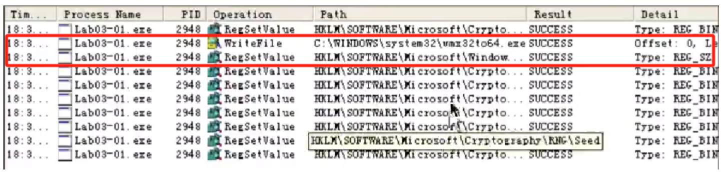


图 5: Process Monitor 监测结果 1

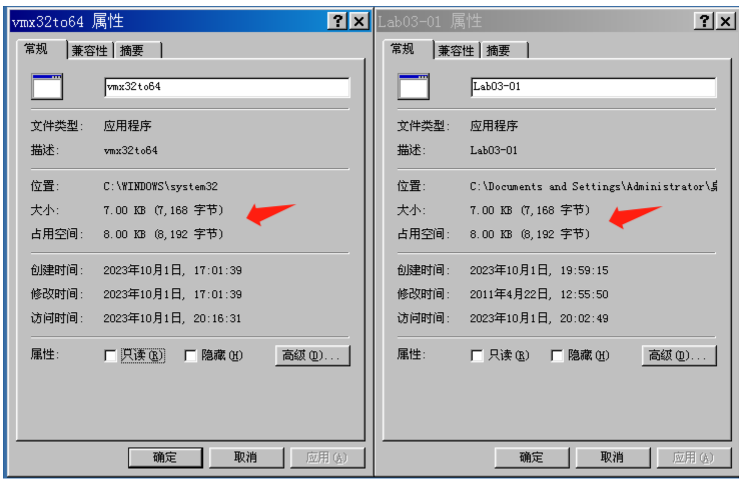


图 6: 文件比对

在进行数据过滤后，我们能够观察到一些与 Cryptography 相关的随机数生成器。然而，这些生成器并不具备实际价值。相反，第二和第三方面的信息则显得十分重要，它们提供了关于“WriteFile”操作的详细信息。该操作涉及将一个名为“vmx32to64.exe”的文件写入到 `C:\WINDOWS\system32\` 路径下，文件大小为 7168 字节，与“Lab03-01”中的文件大小相匹配。这引发了我们的猜测，即这两个文件可能是同构文件（Isomorphic Files）。为了验证这一猜测，我们进行了在该路径下找到此文件，并进行了 MD5 哈希值的比对。如下图7,8所示：

发现在对恶意代码进行分析时，我们注意到两个重要的观察结果。首先，我们观察到恶意代码的 MD5 哈希值存在相等的情况，这意味着两个或多个样本的内容在某种程度上是相似的或相同的。这种相似性证明了我们的猜测。其次，我们观察到这些恶意代码的主要功能之一是将其自身复制到 `C:\WINDOWS\system32` 目录中。这一行为表明恶意代码试图深入操作系统核心，通过将自身复制到系统目录中，利用系统级权限来执行恶意操作。

图 7: vmx32to64.exe

图 8: Lab03-01.exe

观察到另一个重要细节，恶意代码采取了一种潜在危险的行为，即在系统的注册表中创建了一个名为 `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\VideoDriver` 的注册表键值。这个行为表明，恶意代码试图利用 Windows 操作系统的自动启动机制，将自身添加到系统启动项中，从而在每次系统启动时自动运行。这种行为是具有潜在恶意意图的，因为它可以使恶意代码在系统启动时立即执行，可能导致对系统的进一步侵害或隐私数据的窃取。

图 9: Process Monitor 监测结果 2

值得注意的是，注册表是 Windows 操作系统中用于存储配置信息和系统设置的关键组成部分，而 HKLM（HKEY_LOCAL_MACHINE）是其中的一个重要分支，通常包含系统级别的配置信息。因此，对注册表的未经授权的修改可能会对系统的稳定性和安全性造成严重威胁。

总结而言，该恶意软件采取了一系列危险行动以实现其恶意目的。首先，它创建了一个名为 WinVMX32 的互斥量，用于确保只有一个实例在系统中运行。接下来，它将自身复制到 `C:\WINDOWS\system32` 目录下并重命名为 vmx32to64.exe，以混淆其存在并掩盖其恶意特性。此外，该恶意软件还通过将自身添加到系统的自启动项中，来确保在每次系统启动时都会运行。为此，它在系统注册表中创建了一个注册表键值 `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\VideoDriver`，并将其设置为指向其自身复制的位置，以便在系统启动时自动加载。这一系列行动使得该恶意软件能够在背景中持续运行，并具备潜在的危险威胁。因此，及早发现和清除此类恶意软件至关重要，以确保系统的安全性和稳定性。

3. Are there any useful network-based signatures for this malware? If so, what are they?

在进行网络流量分析时，我们采用了 WireShark 工具，通过该工具的分析功能，我们观察到了一系列发送至特定域名“www.practicalmalwareanalysis.com”的 DNS 查询请求。这些 DNS 查询请求的存在与我们先前观察到的字符串内容相符。这些请求会被发送至目标主机的 53 端口，端口 53 通常用于 DNS 通信。值得注意的是，这些请求所携带的数据看似随机生成，其数据包的大小恒定为 0x100（或 256 字节），如图10所示。值得一提的是，通过多次运行观察，我们发现无论运行次数如何，数据包的长度始终保持不变，然而数据内容却呈现出随机性。这种现象可能引发对于数据包生成机制的进一步深入研究和分析的兴趣，以便更全面地理解网络通信中的这些不寻常行为。

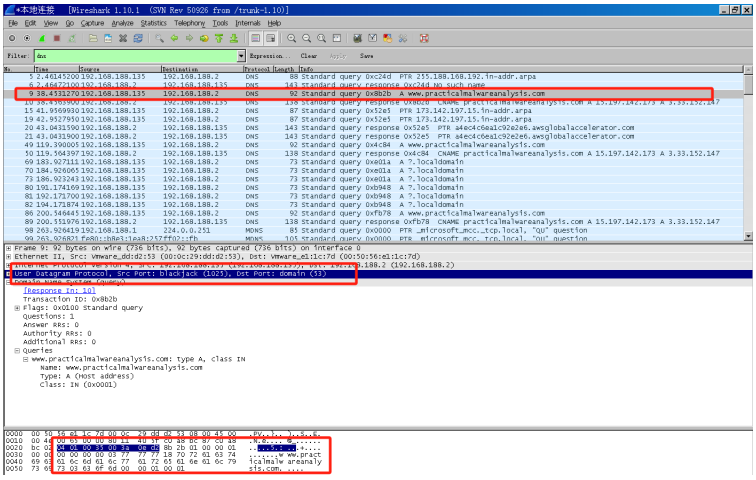
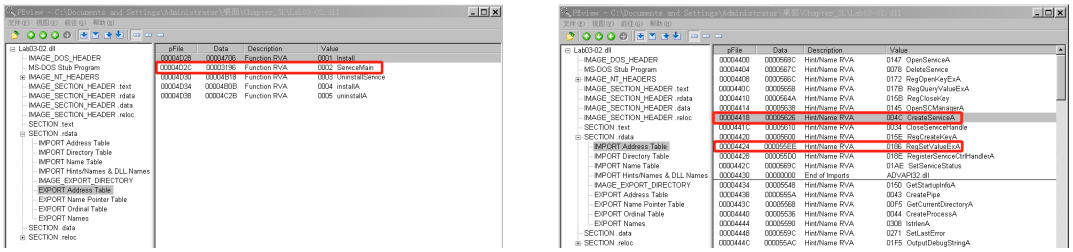


图 10: WireShark 监测结果

(二) Lab 3-2

1. How can you get this malware to install itself?

使用 PEview 工具进行文件分析，如图11a，11b。值得注意的是，被研究的文件是一个动态链接库（DLL）文件。因此，我们将重点分析其导出函数。在仔细检查导出函数列表时，我们发现了一个名为”ServiceMain” 的函数。从这一发现可以推断，该 DLL 文件需要被安装为一个系统服务才能正常运行。此外，在分析图表时，我们还观察到一些与服务操作相关的函数，例如”CreateService”，以及一些与注册表操作相关的函数，例如”RegSetValue” 等。这些发现表明，该 DLL 文件可能被设计成用于与 Windows 服务管理和注册表操作相关的任务。这种文件通常用于扩展或支持其他应用程序或系统功能。



(a) 结果 1 (b) 结果 2

图 11: PEview 分析

接下来，我们将进行一项动态分析的过程。之前的静态分析结果指出，为了将自身成功注册为一个服务，需要使用导出函数 InstallA。为了实现这一目标，我们将利用操作系统内置的 rundll32.exe 工具（需要注意的是，本次实验所需的 DLL 文件已被放置在 C 盘的根目录下）。然后，我们使用 regshot 工具的快照功能来执行运行前后的系统状态比对，以进一步了解和验证该注册过程的影响和结果。这一系列的步骤将有助于我们深入研究和评估所注册的服务对系统的影响以及与其他系统组件的互动。

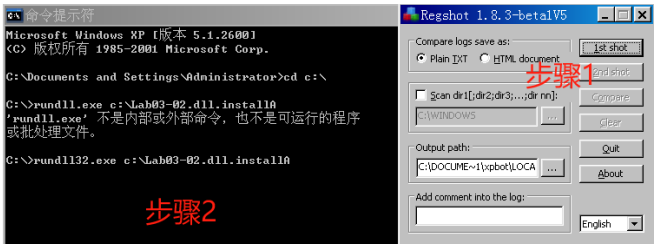


图 12: 使用 rundll32.exe 和 regshot

由此，便回答了第一问的问题。此恶意代码借助于操作系统中的 rundll32.exe 工具来实施其恶意行为。具体而言，它使用了以下命令：rundll32.exe c:\Lab03-02.dll,installA。这一命令的目的在于启动并执行一个被恶意代码导出的 installA 函数。此举的结果是，该恶意代码成功地将自身安装为一个系统服务，从而为其后续的恶意操作提供了可行的执行环境。需要强调的是，这一行为明确表现出了该恶意代码旨在绕过系统的安全机制，以便在受害计算机上长期存在并进行潜在的恶意活动。

2. How would you get this malware to run after installation?

接下来, 我们观察经 regshot 比对后的数据, 可观察到一项关于恶意代码的发现, 即该代码已成功自我安装为 IPRIP (Internet Router Information Protocol) 服务的一部分。进一步深入分析数据, 我们可以注意到在其"imagepath" (镜像路径) 属性中, 该恶意代码被配置为以"svchostexe"的形式运行。值得注意的是, "svchost.exe" 是微软 Windows 操作系统中的系统文件, 官方解释表明它充当了运行动态链接库 (DLL) 中的服务的通用主机进程。这一进程对于维持系统的正常运行至关重要, 因此不得被终止。众多服务通过注入到"svchost.exe" 中以启动, 这也解释了在系统中可能存在多个该文件进程的现象。

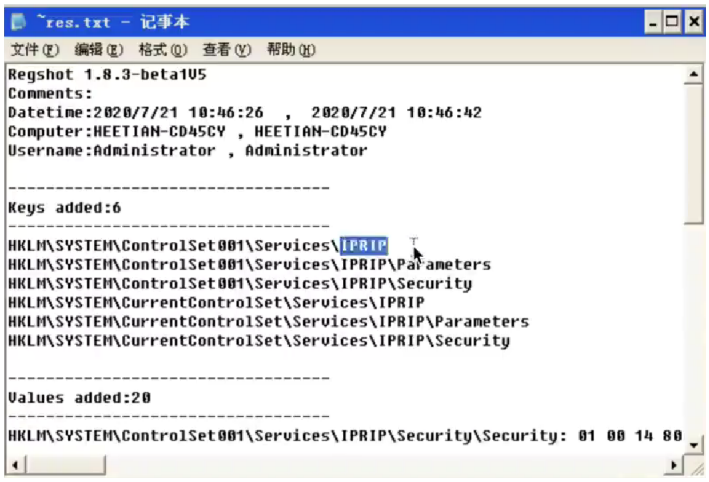


图 13: regshot 的快照内容比对

此次分析还显示, 该恶意代码的 DLL 文件已被成功挂载在"svchost.exe" 进程上运行。此外, 我们还注意到了一些与"INA+" 等信息相关的特征, 这些特征可能是该 DLL 执行后所产生的独有标识。一旦成功安装为服务, 该恶意代码便可被启动并运行。(这里回答了问题 5 所提出的问题)

接下来, 回答本小节提出的问题: 为执行恶意代码, 攻击者可借助"net" 命令执行"net start IPRIP" 指令, 以激活已经部署的恶意代码所依赖的服务, 如图14

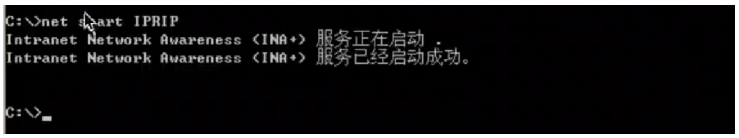


图 14: net start IPRIP

3. How can you find the process under which this malware is running?

在先前的内容中, 我们观察到 Lab03-02.dll 已成功安装为 IPRIP 服务, 并且我们可以利用 Windows 操作系统内置的 'net' 命令进行其启动。在启动该应用程序之前, 我们推荐打开一些监控工具, 以便更好地跟踪和管理其行为。这些监控工具包括 Process Monitor、Process Explorer 以及 Wireshark。

首先, 我们启动 Process Monitor, 这将有助于我们监测应用程序的活动。同时, 使用 Process Explorer 来检查正在运行的进程。需要注意的是, 由于该特定软件是以动态链接库 (DLL) 的形式存在的, 因此在 Process Explorer 中, 它可能不会直接显示在进程列表中。为了找到它, 我们

需要进行一些额外的步骤，如图15所示：

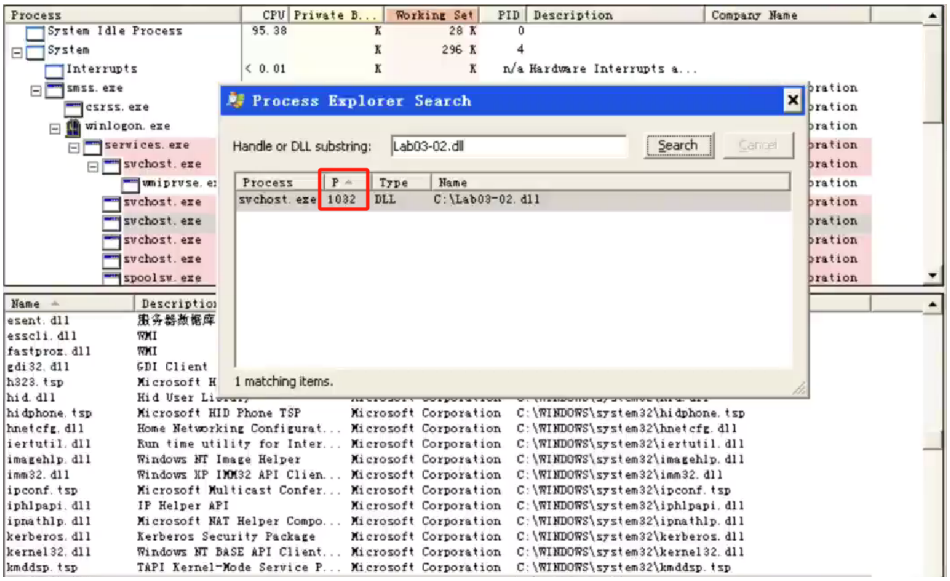


图 15: 在 Process Explorer 中找到该 DLL 文件

在图中可以看到，这个 DLL 由 PID 为 1032 的 svchost.exe 加载。由此，问题三得到了解答：利用 ProcessExplorer 工具，识别系统中正在运行的服务所对应的进程。而恶意代码隐藏就在系统的 svchost.exe 进程中。因此，为了确定特定服务所对应的进程，需要逐一检查每个运行中的进程，直到找到与目标服务相关联的进程名称。另外，也可以利用 ProcessExplorer 的 Find DII 功能来搜索特定的动态链接库文件（DLL），以定位与 Lab03-02.dll 相关的信息。

4. Which filters could you set in order to use procmon to glean information?

在 procmon 工具中，可以使用在 Process Explorer 中发现的 PID 进行过滤，如图15,16。

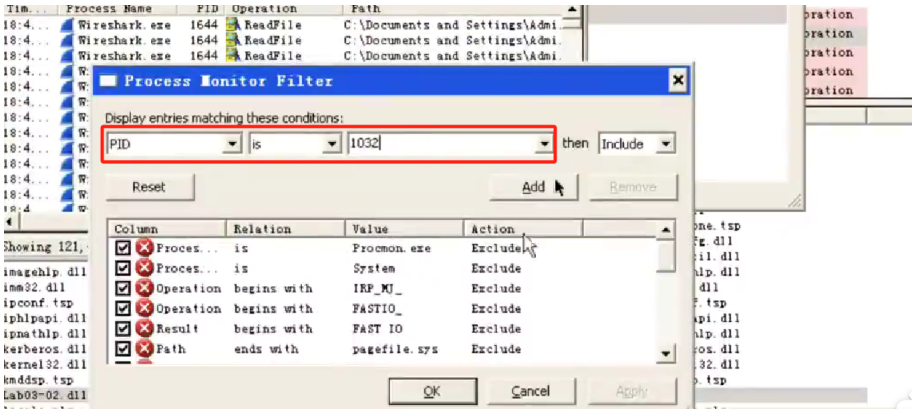


图 16: 过滤 PID

5. What are the malware's host-based indicators?

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 “DependsINA+, Collects and stores network configuration and location information, and notifies applications when this information changes”。它将自身持久地安装在注册表中

HKLM\SYSTEM\CurrentControlSet\Services\IPRIP\Parameters\ServiceDll:%CurTentDirectory%\Lab03-02.dll。

6. Are there any useful network-based signatures for this malware?

通过对 Wireshark 进行观察，鉴于网络流量的庞大数量，首先进行了一项筛选操作（图??）。在此筛选过程中，注意到了一系列 DNS 请求，其中包含了对”practicalmalwareanalysis.com” 域名的请求。随后，我们进一步细化筛选条件，专注于 HTTP GET 请求，以便深入了解与之相关的网络活动。在这一步筛选后，我们观察到系统将会访问”malware.com/serve.html” 资源（图18，这一行为与之前进行的静态恶意软件样本分析结果相一致。值得注意的是，在 HTTP 请求的头部信息中，我们发现了一种有趣的特征，即 User-Agent 字段的构成。该字段的前半部分包含主机名信息，而后半部分则始终为”windowsxp 6.11”。这种 User-Agent 构成的一致性可以被视为潜在的网络特征，用于识别与此次网络活动相关的主机或恶意软件实例。

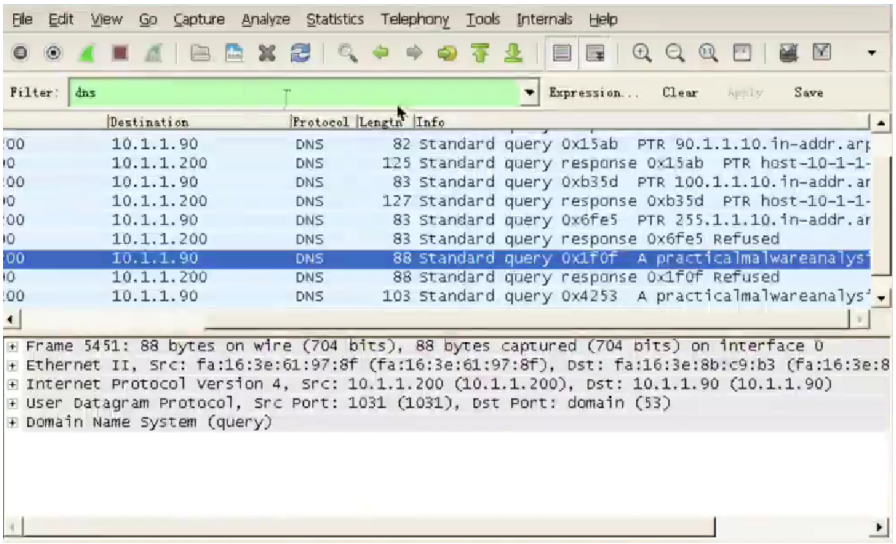


图 17: 过滤 DNS

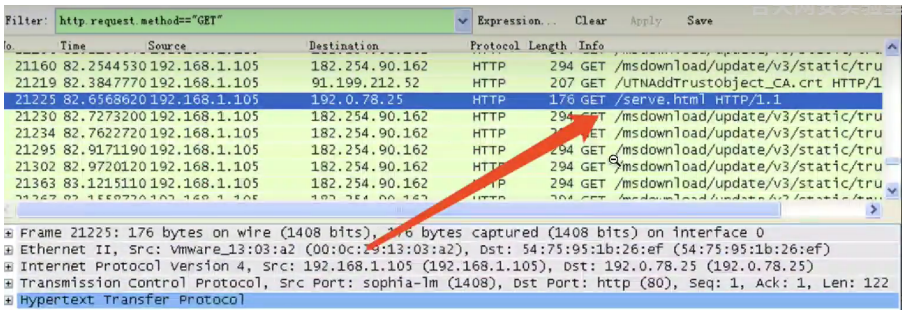


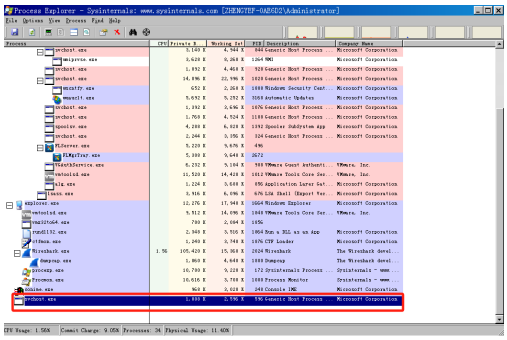
图 18: 过滤 GET

(三) Lab 3-3

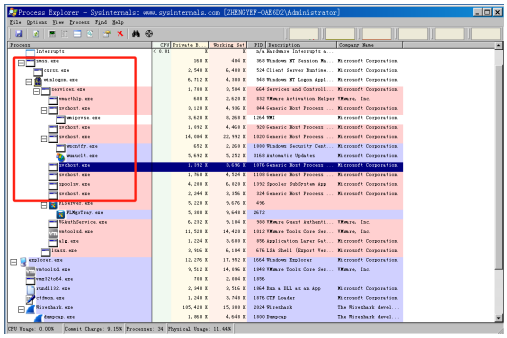
1. What do you notice when monitoring this malware with Process Explorer?

在进行进程分析时，首先打开 Process Explorer。接下来，执行程序 lab03-03.exe。在此操作过程中，观察到 Process Explorer 会在启动后迅速闪现并消失。这个现象的背后机制涉及到

lab03-03.exe 程序的行为。具体而言，lab03-03.exe 程序在启动后会创建一个名为”svchosL.exe” 的新进程。随后，lab03-03.exe 进程自身便自动终止，但是值得注意的是，新创建的”svchosL.exe” 进程将被保留并运行。这种情况下，”svchosL.exe” 进程可以被描述为一个孤立的进程，即没有与之相关联的父进程，它独立运行在系统中（图19a），这是可疑的。相比在正常情况下，schost.exe 应当如图19b所示。



(a) 异常 svchosL.exe 进程



(b) 正常的 svchosL.exe 进程

图 19: 两种 svchosL.exe 进程对比

这一现象可能涉及到一些高级的系统或程序设计技巧，例如自删除功能，以确保程序的执行在不留下痕迹的情况下进行。这种进程行为可能涉及到对系统资源的隐蔽性利用或其他安全性相关的操作。深入研究这一现象需要进一步的分析，以确定潜在的恶意行为。

由此我们回答了第一问，即恶意代码成功执行了对”svchost.exe” 文件的替换操作，取代了原本有效的”svchost.exe” 文件。

2. Can you identify any live memory modifications?

在所检视的内存镜像中，我们观察到存在 Log 日志文件以及一系列与标准键盘符号相似的字符串，如”Tab”、”Enter” 等（图20）。这些字符串在经典的 svchost.exe 进程中通常是不会出现的。这一观察结果引发了对系统内部安全性和进程完整性的担忧，因为这些异常字符的存在可能指示着潜在的安全威胁或异常操作。为了深入理解这些现象，进一步的分析和研究将是必要的。

由此,我们给出第二问的回答: 在比较内存映像与磁盘映像中的 svchost.exe 时,我们可以观察到它们之间存在显著差异。具体而言,内存映像包含了一些特定字符串,例如”practicalmalwareanalysis.log” 和”[ENTER]”，然而，在磁盘映像中并未发现这些字符串的存在。

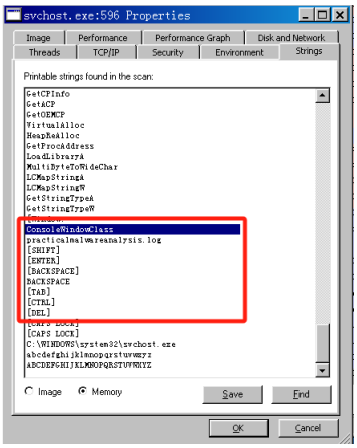


图 20: 异常的内存镜像

3. What are the malware’s host-based indicators?

根据以上分析，我们推测所涉可能涉及的恶意活动与键盘记录器相关。为了进一步验证这一猜测，我们采取了以下步骤：首先，我们随意打开一个文本编辑器并输入一些文字（图21）。然后，我们获取该进程的进程标识符（PID），以便在进程监视器（Process Monitor）中进行监视（图22）。同时，我们通过过滤处理以排除无关信息的方式，精炼我们的监视结果（图23）。

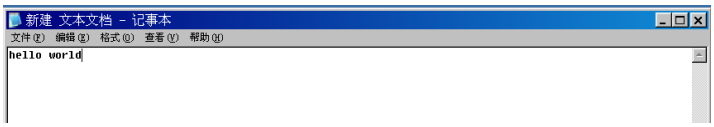


图 21: 记事本

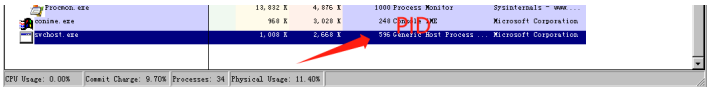


图 22: 获取 PID

在经过信息筛选后的结果中，我们观察到恶意代码似乎持续在操作一个日志文件（图23）。为了更深入地了解其内容，我们使用文件路径信息成功打开了该日志文件。在该文件中，我们发现了详细的记录，包括我们所处的窗口以及我们输入的键盘操作，这一发现进一步验证了我们的推测（图24）。

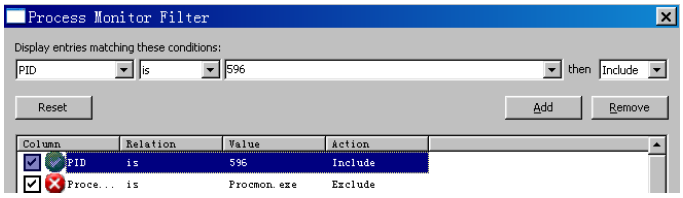


图 23: 日志文件

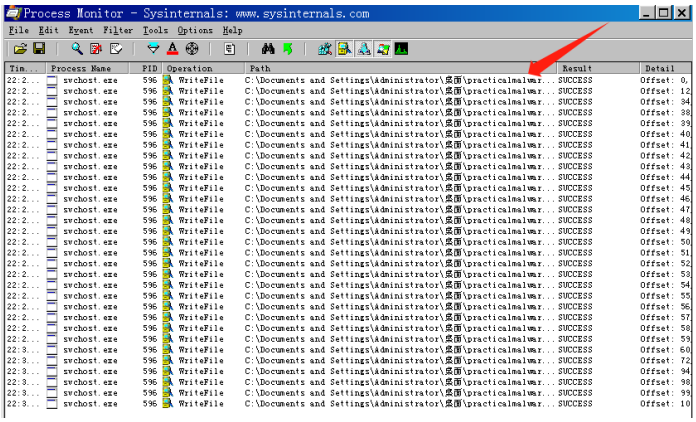


图 24: 恶意代码操纵的文件

由此回答了第三问：这恶意代码生成了一个名为”practicalmalwareanalysis.log” 的日志文件。

4. What is the purpose of this program?

在上一节的讨论中也回答了本问的问题：该程序进行了进程替换操作，以在 svchost.exe 进程上启动一个键盘记录器。

(四) Lab 3-4

1. What happens when you run this file?

我们对 lab03-04.exe 进行了深入分析,首先进行了静态分析(图25,26)。在载入 PEView 进行导入函数查看时, 我们注意到其中涉及了与注册表相关的操作, 同时还存在一个名为"deleteservice"的函数,其功能将在后续分析中进一步阐明。此外,我们还寻找到了一些特殊的字符串,包括域名、注册表位置（例如 SOFTWARE\Microsoft\XPS)、"http/1.0" 以及"DOWNLOAD" 和"UPLOAD" 等信息。基于这些发现，我们初步猜测 lab03-04.exe 可能是一个后门程序，其可能未经授权访问互联网资源。此外，我们还观察到存在一些命令行参数，如"-cc"、"-re" 和"-in"，其中"-in" 可能是"install" 的缩写。

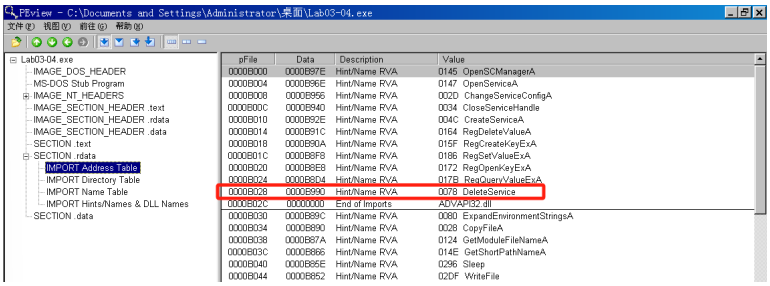


图 25: 静态分析 1

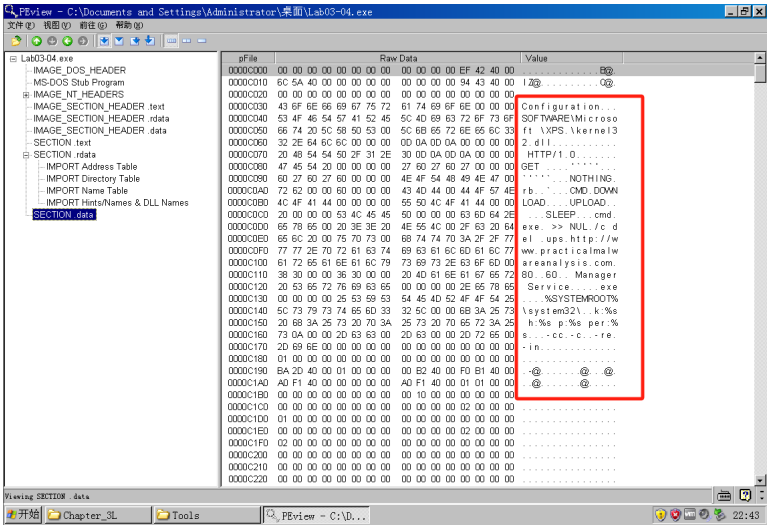


图 26: 静态分析 2

接下来，我们进行了动态分析，以监视程序的行为。为了实现此目的，我们打开了 Process Monitor 和 Process Explorer 工具。在将 lab03-04.exe 复制到桌面后，我们双击运行该程序，首先出现一个弹窗，然后程序本身被自动删除（由此我们回答了第一问)。而与此同时，整个过程中，process explore 的进程列表没有任何变化。进而我们转向 process monitor，添加相应的过滤条件（图27)。

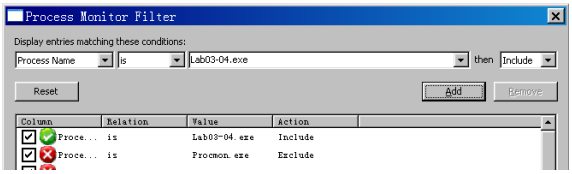


图 27: 设置过滤器

可以观察到存在”process create” 操作，该操作涉及打开一个 cmd 程序，用户可以通过双击查看事件属性，如图所示（见图28,reffig:M)。从中我们可以得知，通过 cmd 程序执行了一条命令，而该命令的字符串内容与我们之前进行静态分析时所观察到的内容相符。这条命令的主要功能是删除当前进程自身，由此我们也回答了第二问的问题。

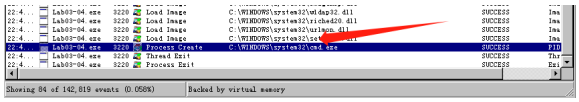


图 28: 查看异常事件属性 1

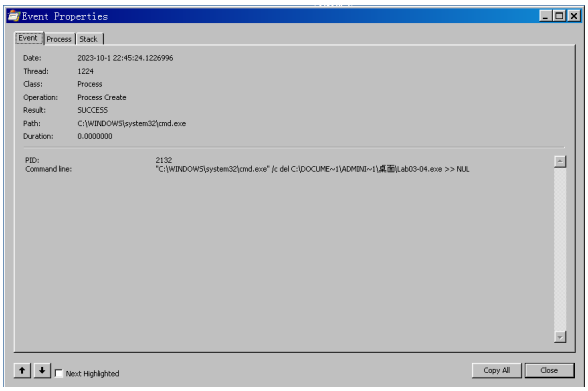


图 29: 查看异常事件属性 2

2. What is causing the roadblock in dynamic analysis?

我们怀疑该程序的动态运行需要提供一个命令行参数，或者此程序的某些组件可能存在缺失。这可能是由于程序执行过程中的异常现象或者功能不完整引发的。在解决这一问题时，有必要对程序的输入参数和内部组件进行仔细检查，以确定问题的确切原因，并采取适当的措施进行修复或恢复程序的正常运行状态。

3. Are there other ways to run this program?

我们试图运用一些命令行参数，例如前文提到的”-in,” ”-re,” ”-cc” 等参数，以解析字符串列表。然而，这一尝试并未产生有效的结果，表明需要进行更深入的分析以解决问题。

(五) Yara 规则编写

1. Lab 3-1

这个 YARA 规则是为了检测”Lab03-01.exe” 这个恶意文件。规则中定义了多个字符串模式，如 “vmx32to64.exe”、特定的注册表路径和一个恶意网址等。这些模式被用来在文件中进行匹配。

触发规则的条件包括: 文件的前两个字节为 “MZ” (PE 文件的标准标识); 检查 PE 头的标识来确定文件是否为 PE 格式; 文件大小小于 20KB; 并确保上述定义的字符串中至少有 6 个存在于目标文件中。简而言之, 这个规则通过检测特定的字符串模式和文件属性, 来识别与”Lab03-01.exe” 相关的恶意行为或特征。

```
rule Lab03_01_exe {
   meta:
      description = "Lab03-01.exe"
   strings:
      $exe_name = "vmx32to64.exe" fullword ascii
      $reg_path = "SOFTWARE\\Classes\\http\\shell\\open\\commandV" fullword ascii
      $malware_url = " www.practicalmalwareanalysis.com" fullword ascii
      $http_connect = "CONNECT ???  HTTP/1.0" fullword ascii
      $lib_name = "advpack" fullword ascii
      $driver_name = "VideoDriver" fullword ascii
      $app_data = "AppData" fullword ascii
      $vmx_prefix = "WinVMX32-" fullword ascii
      $setup_path = "Software\\Microsoft\\Active Setup\\Installed Components\\"
          fullword ascii
   condition:
      // Check for MZ header
      uint16(0) == 0x5a4d and
      // Check for PE header
      uint32(uint32(0x3c))==0x00004550 and
      filesize < 20KB and
      6 of them
}
```

2. Lab 3-2

规则首先定义了一系列要在文件中搜索的字符串, 如系统路径中的 “svchost.exe”、命令行执行的字符串等。这些字符串都有特定的编码和匹配方式, 如 ASCII 编码和完整单词匹配。规则的条件部分确保文件具有 Windows 可执行文件的标准 MZ 和 PE 标头, 同时文件大小必须小于 70KB。最后, 为了触发规则, 文件中必须至少包含一个系统相关的字符串和其他四个定义的字符串。总的来说, 这个规则结合了文件属性和特定的字符串模式, 以准确地识别和标记恶意软件样本。

```
rule Lab03_02_dll {
   meta:
      description = "Revised version of Lab03-02.dll"

   strings:
      $systemRootString = "%SystemRoot%\\System32\\svchost.exe -k " fullword ascii
      $cmdExecutionString = "cmd.exe /c " fullword ascii
      $regOpenErrorString = "RegOpenKeyEx(%s) KEY_QUERY_VALUE error ." fullword
          ascii
      $malwareDomainString = "practicalmalwareanalysis.com" fullword ascii
      $dllNameString = "Lab03-02.dll" fullword ascii
      $regOpenSuccessString = "RegOpenKeyEx(%s) KEY_QUERY_VALUE success." fullword
          ascii
      $htmlFileString = "serve.html" fullword ascii
      $dllPathString = "GetModuleFileName() get dll path" fullword ascii
      $netsvcsString = "netsvcs" fullword ascii
      $openServiceError2String = "OpenService(%s) error 2" fullword ascii
```

```
        $openServiceError1String = "OpenService(%s) error 1" fullword ascii
        $createServiceErrorString = "CreateService(%s) error %d" fullword ascii
        $svcNameErrorString = "You specify service name not in Svchost//netsvcs, must
             be one of following:" fullword ascii
        $regQueryValueString = "RegQueryValueEx" fullword ascii
        $dependsINAString = "Depends INA+" fullword nocase
        $uninstallSuccessString = "uninstall success" fullword ascii
        $uninstallStartString = "uninstall is starting" fullword ascii

    condition:
        // Check for MZ header
        uint16(0) == 0x5a4d and
        // Check for PE header
        uint32(uint32(0x3c)) == 0x00004550 and
        // File size limit
        filesize < 70KB and
        // At least one of the system-related strings and 4 other strings
        1 of ($systemRootString, $cmdExecutionString) and 4 of them
}
```

3. Lab 3-3

该 YARA 规则主要包含两个关键字符串：完整词 “svchost.exe” 和 “+A+A+A+A”。规则的条件部分定义了三个主要检测点：首先，它会检查文件的前两个字节是否为 PE 文件的标准魔数 “0x5a4d”；接着，它会进一步确认文件是否为 PE 格式，通过检查特定偏移量处的值是否为 “0x00004550”；最后，它还会确保文件大小小于 200KB。只有当这些条件都满足，且文件中至少包含上述两个关键字符串中的一个时，该规则才会触发。这个规则为安全研究者提供了一个有效的工具，用于检测和识别特定的恶意文件或行为。

```
rule Lab03_03_exe {
    meta:
        description = "Revised Lab03-03.exe detection"

    strings:
        $string_svchost = "\\svchost.exe" fullword ascii
        $string_pattern = "+A+A+A+A" fullword ascii

    condition:
        uint16(0) == 0x5a4d and
        uint32(uint32(0x3c)) == 0x00004550 and
        filesize < 200KB and
        ( $string_svchost or $string_pattern )
}
```

4. Lab 3-4

在该规则中我们定义了一系列字符串模式，如特定的 URL、系统路径和命令，这些模式可能与某恶意软件样本相关。规则的 “meta” 部分提供了描述信息，指出这是为 “Lab03-04.exe malware sample” 设计的检测。在 “strings” 部分，我们列出了要在目标文件中搜索的具体字符串或字节模式。而 “condition” 部分则定义了规则匹配的条件。为了确保目标是一个 Windows PE 可执行文件，规则检查文件的开头是否有标准的 “MZ” 和 “PE\0\0” 标头。此外，它还要求文件大小必须小于 200KB，并且至少包含上述定义的 6 个字符串。简而言之，这个 YARA 规则旨在检测

一个特定的、小于 200KB 的 PE 文件，该文件包含与 “Lab03-04.exe” 恶意软件样本相关的特定模式。

```
rule Lab03_04_exe {
   meta:
      description = "Detection for Lab03-04.exe malware sample"

   strings:
      $url_string = "http://www.practicalmalwareanalysis.com" fullword ascii
      $system_path = "%SYSTEMROOT%\\system32\\" fullword ascii
      $http_protocol = " HTTP/1.0" fullword ascii
      $manager_service = " Manager Service" fullword ascii
      $upload_cmd = "UPLOAD" fullword ascii
      $download_cmd = "DOWNLOAD" fullword ascii
      $command_com = "command.com" fullword ascii
      $comspec_env = "COMSPEC" fullword ascii
      $registry_key = "SOFTWARE\\Microsoft \\XPS" fullword ascii
      $delete_cmd = "/c del " fullword ascii
      $null_redirect = " >> NUL" fullword ascii

   condition:
      uint16(0) == 0x5a4d and  // MZ header
      uint32(uint32(0x3c)) == 0x00004550 and  // PE header
      filesize < 200KB and
      6 of them
}
```

5. 扫描对应文件

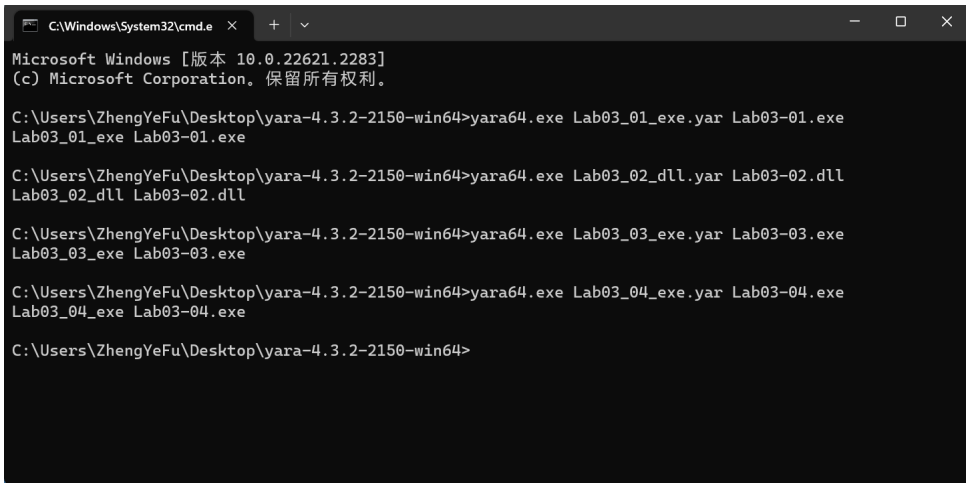


图 30: 扫描结果

四、 实验结论及心得体会

通过进行恶意软件分析实验，我们获得了宝贵的经验和深刻的认识。首先，我们深刻理解了静态分析和动态分析在恶意软件分析中的互补性。静态分析揭示了关于文件结构和字符串的重要信息，但为了全面了解恶意软件的行为，动态分析是不可或缺的。通过结合使用工具如 Process Explorer、Procmon、ApateDNS、Netcat 和 Wireshark，我们能够监控进程、文件系统、注册

表和网络通信，揭示了恶意软件的活动模式和意图。这个结论对于我们更好地理解和对抗恶意软件具有重要意义。其次，我们认识到工具的选择对于恶意软件分析至关重要。专门设计的工具如 Process Explorer 和 Procmon 为我们提供了深入分析的能力，让我们能够跟踪进程、检测文件系统和注册表的变化，发现了恶意软件的隐藏行为。另一方面，网络分析工具如 ApateDNS 和 Netcat 帮助我们审查 DNS 请求和网络通信，揭示了潜在的网络活动。因此，精心选择工具是成功进行恶意软件分析的关键因素之一。第三，我们了解到不同的恶意软件可能具有不同的行为和特征。在实验中，我们面对各种类型的恶意软件，它们的功能和行为各不相同。有些恶意软件追求持久性，尝试在系统启动时自动运行，而其他一些则具备自毁特性，以避免被检测和分析。这种多样性强调了我们作为恶意软件分析人员需要具备广泛知识和技能的重要性，以应对不同威胁并采取适当的对策。最后，我们认识到恶意软件分析是一项复杂而具有挑战性的任务。我们在实验中遇到了各种困难，如需要特定配置或深入分析才能揭示恶意软件的真正功能。这凸显了持续学习和不断更新技能的必要性，以跟上恶意软件不断演进的威胁和攻击技巧。总而言之，通过这些实验，我们不仅积累了有关恶意软件分析的知识，还培养了分析和解决问题的能力，这将有助于我们更有效地应对未来的恶意软件挑战。

参考文献

[1] SM-D.Practical Malware Analysis[J].Network Security, 2012, 2012(12):4-4.DOI:10.1016/S1353-4858(12)70109-5.