## LAB5 实验报告

专业：信息安全 姓名：魏伯繁 学号：2011395

### 第一部分：实验环境：

本机：WIN10系统、16G内存，2.9GHZ主频

虚拟机：winxp虚拟机 4G内存

### 第二部分：实验工具：

静态分析工具：PEView、strings

动态分析工具：process moonitor、process explorer、reg shot、wireshark

### 第三部分：实验内容

实验一目标：

1、找出这个函数的导入函数以及字符串列表

只有一个，应该被加壳了；

字符串列表存在一个恶意网站的信息以及一些路径信息

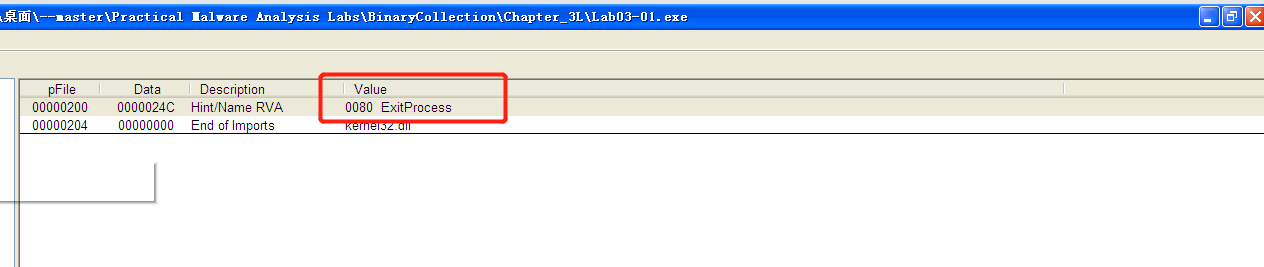
2、这个恶意代码在主机上的感染迹象是什么

创建了一个互斥量，便于多线程时使用；向C盘文件写入了自身，并修改注册表，将自己添加到启动项中。

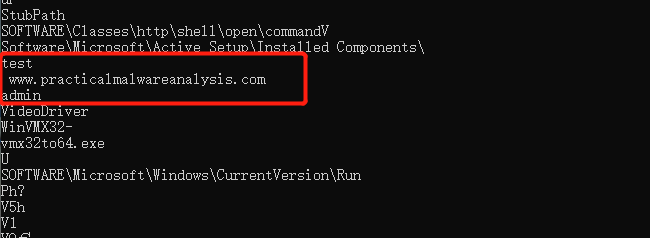
3、这个恶意代码是否存在一些有用的网络特征码？如果存在，他们是什么？

存在，他会在443端口不断的请求访问，访问恶意网站。

首先，使用PEview对文件进行静态扫描，发现其只有一个导入函数，不出意外该程序是被加壳了



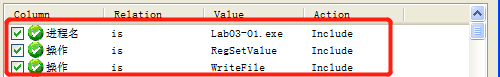
于是我在win10虚拟机上用string对文件进行了扫描：发现存在网络访问的迹象，说明其很有可能访问了网络。



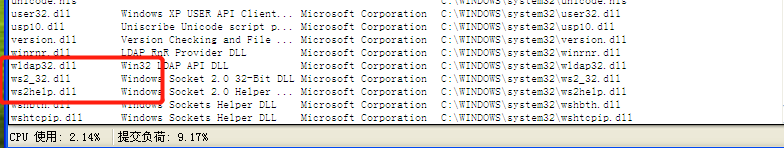
紧接着，打开process monitor、process explorer以及wireshark进行监视。

其中，wireshark的过滤器选择ip.addr==192.168.230.113&&tcp.port==443来检测使用本机ip可能的联网行为。

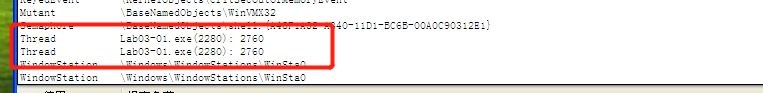
process monitor的过滤器的值则初始化为：



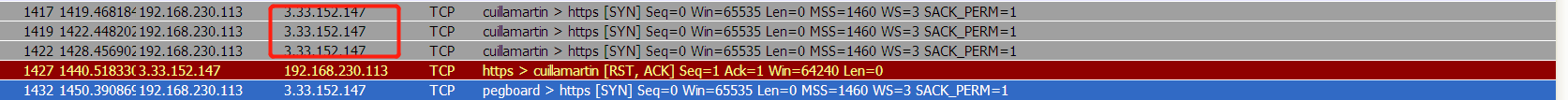
在初始化完成后，点击执行exe文件。通过process explorer我们可以观察到该可执行文件调用了与联网有关的ws2\_32.dll



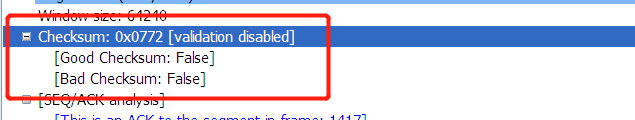
查看句柄，发现可执行文件使用了多线程并创建了一个互斥量。



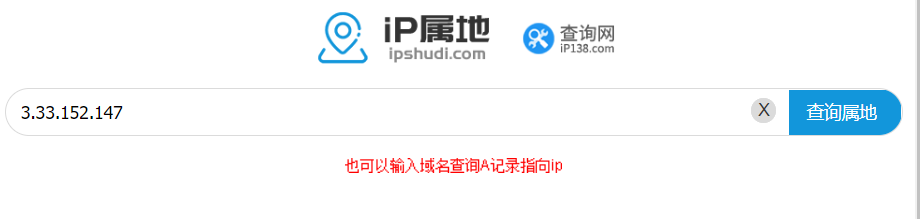
使用wireshark进行监视，发现exe文件视图访问ip地址3.33.152.147，但同样根据红色的数据包显示，这个访问失败了



因为资源不合法，所以并没有访问成功该ip地址

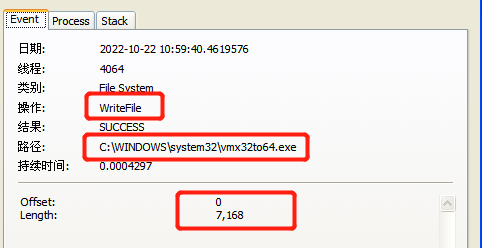


经过查验，他确实不是一个有效的ip地址，说明这非常有可能就是在string中检测到的网址，是黑客注册的恶意网站。

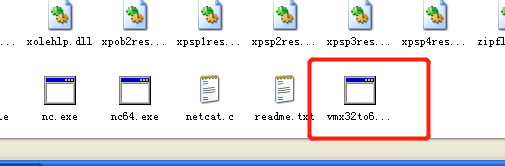


在procmon中，我们可以看到如下的信息：该进程修改了注册表，修改了系统文件，我们点击查看详情：

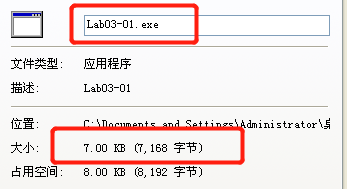




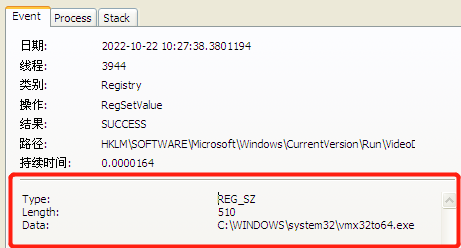
发现可执行文件在相应路径下写入了一个.exe



查看相应路径果然多出了一个exe，这时我们发现，写入的文件大小与lab03-01.exe一致！说明写入的文件就是我们运行的可执行文件：



我们点开其他信息，在这个对注册表的修改中证实了我们的猜想：



对应的yara规则：

import "pe"  
rule IsPE{  
 meta:  
 description = "检查文件是否为PE文件"  
 author="wbf"  
 date="2022-10-12"  
 condition:  
 uint16(0) == 0x5A4D and //“MZ”头  
 uint32(uint32(0x3C)) == 0x00004550 // “PE”头  
  
}  
  
rule smallFileSize{  
 meta:  
 description = "检查文件小是否大概率为exe文件"  
 author="wbf"  
 date="2022-10-15"  
 condition:  
 filesize<500KB  
}  
  
rule mayBeMaleware{  
 meta:  
 description = "检查文件小是否大概率为maleware文件"  
 author="wbf"  
 date="2022-10-15"  
 strings:  
 $a = { E8 00 00 00 00 }  
 condition:  
 $a at pe.entry\_point  
}  
rule thread{  
 meta:  
 description = "检查文件是否可能具有多线程行为"  
 author="wbf"  
 date="2022-10-22"  
 strings:  
 $a = "thread" nocase  
 $b = "winVMX32" nocase  
 condition:  
 ($a or $b) and smallFileSize and IsPE  
}  
rule malwareNet{  
 meta:  
 description = "检查是否存在已知恶意网站"  
 author="wbf"  
 date="2022-10-22"  
 strings:  
 $a = "www.practicalmalwareanalysis.com" nocase  
 condition:  
 $a  
}

实验二：

1、怎么样才能让这个恶意代码自行安装？

执行命令rundll32 Lab03-02.dll,installA，也就是恶意代码的导出函数就可以将恶意代码安装为一个服务。

2、安装后，如何运行这个恶意代码？

使用net start IPRIP启动恶意代码的安装服务

3、怎样找到这个恶意代码是在哪个进程下运行的？

在regshot中可以找到imagepath这样的路径，点名了他会被哪个exe文件加载，也就是svchost.exe文件

4、要设置什么样的过滤器才能在procmon中收集到恶意代码的信息？

可以设置其processname为svchost.exe

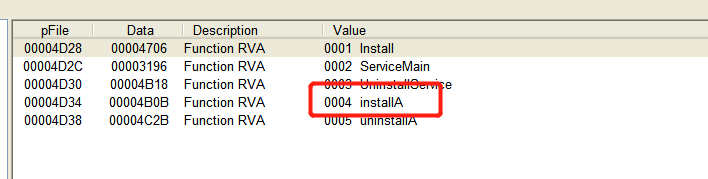
5、这个恶意代码在主机上的感染特征是什么

恶意代码被包装成了IPRIP文件，显示的服务为INA+，它把自己永久的保存在注册表中

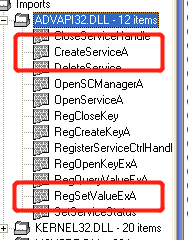
6、这个恶意代码是否存在一些有用的特征码

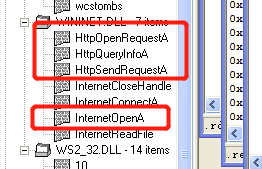
存在一个域名practicalmalwareanalysis.com，利用80端口连接本机（192.168.230.113）,通过wireshark观察到做了一个GET请求

首先，还是进行静态分析：查看导出表，发现恶意代码需要被调用函数执行并且可以被安装成一个服务。

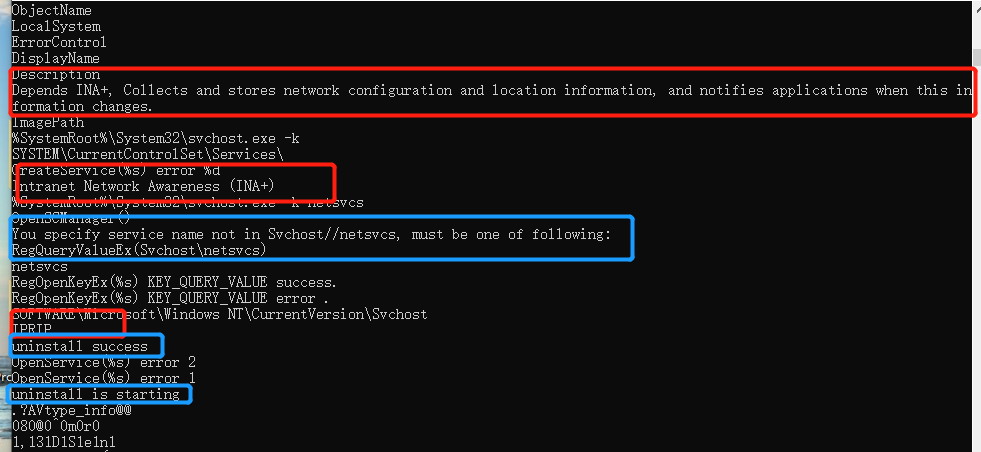


再查看导入函数：导入函数证明了我们的猜想，确实可以创建一个服务并对服务进行操作，同样也可以访问http网络，打开网络，这个dll都可以帮助加载这进行网络连接。

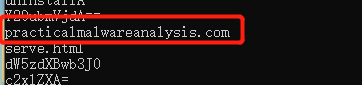




最后，我们使用strings在win10虚拟机上查看其字符串：其中，红色的部分暗示了我们可以如何启动这个dll，蓝色的部分证明，他会引导宿主进行一系列的操作

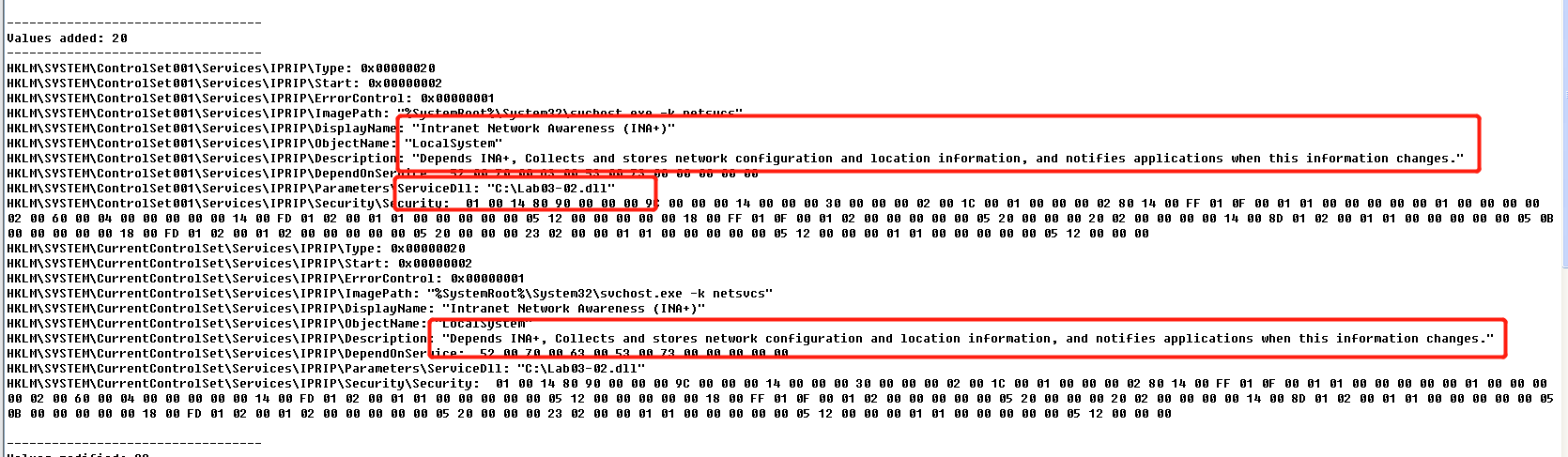


通过简单的筛查，毫无疑问又找到了这个熟悉的网站。



首先，通过静态分析得这这个dll加载后会修改注册表，也会进行网络访问，于是先配置好wireshark和regshot的运行环境，准备好记录dll被加载后的主机出现的一系列反应。首先将恶意代码dll拷贝到C:/windows/system32下，使用rundll32来安装恶意代码

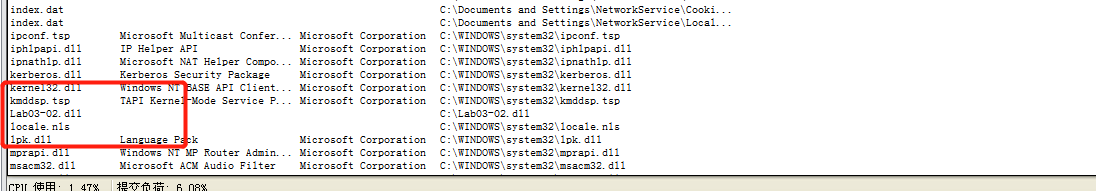


安装完毕后，再次进行regshot，并点击compare查看比较结果：从结果中我们可以分析到：恶意代码将自身安装位IPRIP服务，在imagepath中也可以查看到他会依赖于svchost.exe进行加载，其网络显示名称也应包含INA+  


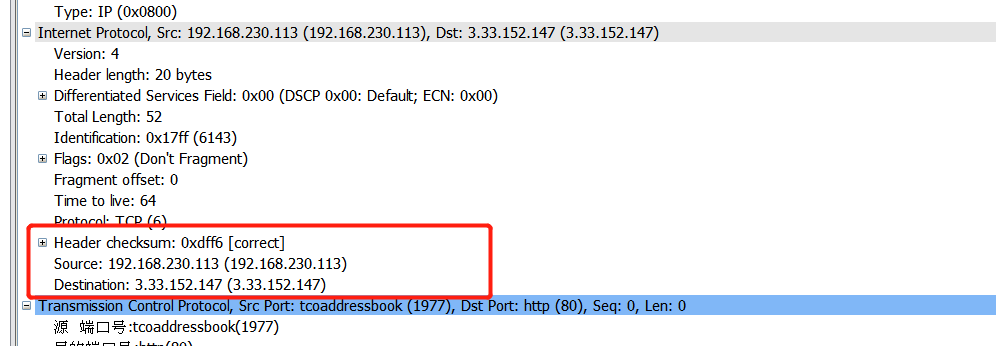
于是，我们使用net start IPRIP来启动服务：



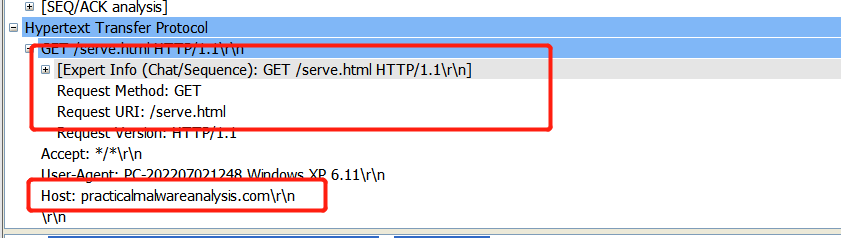
启动后，在process exploer中查找imagebase里提及到的svchost.exe，查看其加载的dll发现Lab03-02.dll被加载了，



于是，我们马上转移到wireshark中：我们发现，此时还没有任何迹象表明有恶意网站被访问的请求，在等待大约60秒后，wireshark突然抓到了很多数据包，经过分析，我们再次看到了在实验1中看到的ip地址，在经过三次握手后，我看到了本地ip向目标ip的GET请求：



点击进入查看详情：我们发现该网站确实为我们在字符串中查看到的恶意网站。



编写yara规则：

import "pe"
  
rule IsPE{
  
 meta:
  
 description = "检查文件是否为PE文件"
  
 author="wbf"
  
 date="2022-10-12"
  
 condition:
  
 uint16(0) == 0x5A4D and //“MZ”头
  
 uint32(uint32(0x3C)) == 0x00004550 // “PE”头
  
  
}
  
  
rule smallFileSize{
  
 meta:
  
 description = "检查文件小是否大概率为exe文件"
  
 author="wbf"
  
 date="2022-10-15"
  
 condition:
  
 filesize<500KB
  
}
  
  
rule mayBeMaleware{
  
 meta:
  
 description = "检查文件小是否大概率为maleware文件"
  
 author="wbf"
  
 date="2022-10-15"
  
 strings:
  
 $a = { E8 00 00 00 00 }
  
 condition:
  
 $a at pe.entry\_point
  
}
  
rule isAService{
  
 meta:
  
 description = "检查文件是否是否可能被安装为一个服务"
  
 author="wbf"
  
 date="2022-10-22"
  
 strings:
  
 $a = "install" nocase
  
 $b = "installA" nocase
  
 $c = "IPRIP" nocase
  
 $d = "CreateService"
  
 $e = "server.html" nocase
  
 $f = "INA" nocase
  
 condition:
  
 ($a or $b or $c or $f or $e) and smallFileSize and IsPE and d
  
}
  
rule connectToNet{
  
 meta:
  
 description = "检查是否可能访问网络"
  
 author="wbf"
  
 date="2022-10-22"
  
 strings:
  
 $a = "www"
  
 $b = "HttpSendRequest"
  
 $c = "http"
  
 $d = "https"
  
 $e = "InternetConnect"
  
 $f = "HttpOpenRequest"
  
 condition:
  
 (($a or $d or $c) and($b or $e or $f)) and smallFileSize and IsPE
  
}
  
rule changeRegister{
  
 meta:
  
 description = "检查是否可能修改注册表"
  
 author="wbf"
  
 date="2022-10-22"
  
 strings:
  
 $a = "RegSetValueEx"
  
 condition:
  
 $a and smallFileSize and IsPE
  
}
  
rule malwareNet{
  
 meta:
  
 description = "检查是否存在已知恶意网站"
  
 author="wbf"
  
 date="2022-10-22"
  
 strings:
  
 $a = "www.practicalmalwareanalysis.com" nocase
  
 condition:
  
 $a
  
}

实验三

1、当你使用ProcessExploer工具进行监视时，你注意到了什么？

Lab03-03很快就退出了，转而出现了一个孤立的svchost进程

2、你可以找出任何的内存修改行为吗？

内存中的exe文件和磁盘中的exe文件是不一致的，所以其必然修改了内存中的svchost使其进行恶意操作

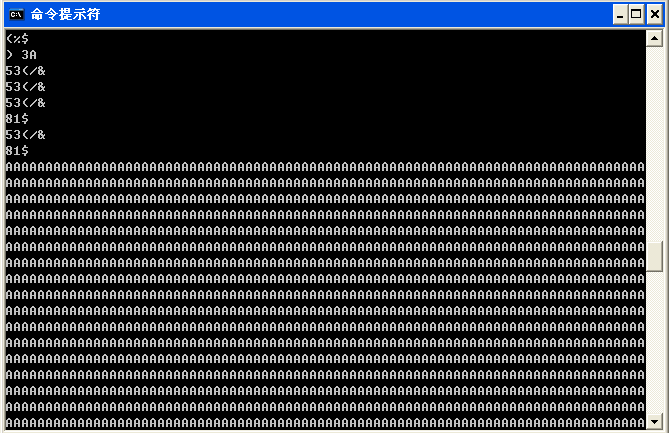
3、这个恶意代码在主机上的感染迹象是什么？

在Lab03-03.exe的文件夹下创建了一个practicalmalwareanalysisi.log文件并记录键盘敲击内容

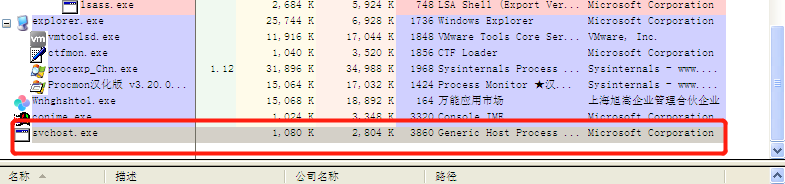
4、这个程序的作用是什么？

替换原有的svchost进程，记录我的键盘敲击内容

首先还是想使用静态分析看一下这个可执行文件，发现他被加密的很好，什么信息都么有透露出来：

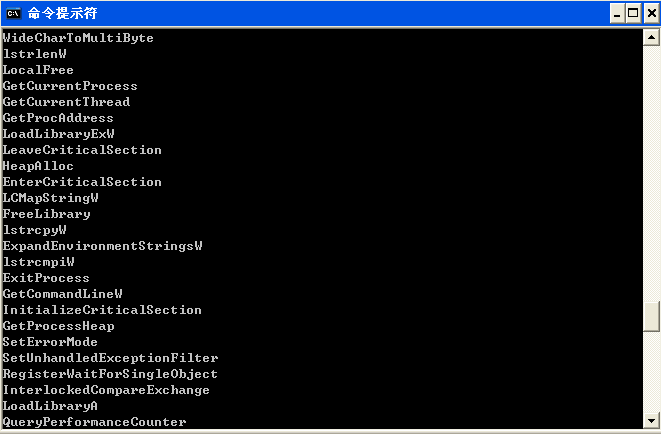


随后，配置process exploer和process monitor，点击可执行文件，奇怪的是，我并没有看到Lab03-03.exe出现在process exploer中，反而看到了一个孤立的进程：很奇怪的是，其他的和他的同名进程在上面也有，是service.exe的子进程。

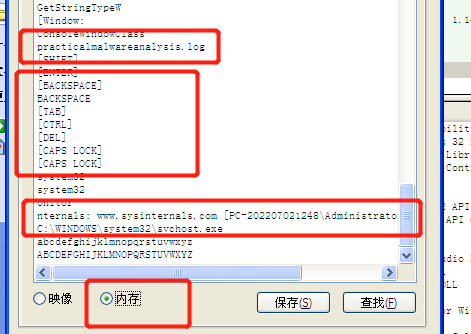


于是我们点击该进程，查看其字符串，并与C盘system32文件夹下的svchost.exe进行对比：

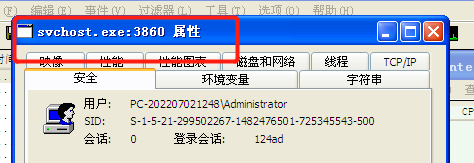
使用strings工具查看C盘下的svchost.exe



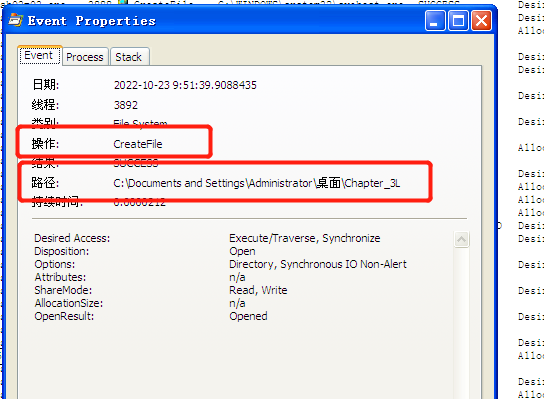
通过process exploer查看内存中的svchost字符串：毫无疑问这个新的孤立的进程就是Lab03-03创建的，其中还有很多记录键盘敲击的内容信息，我们可以猜想这个恶意进程可能记录了我的键盘敲击记录，并将其记录在了.log文件中。



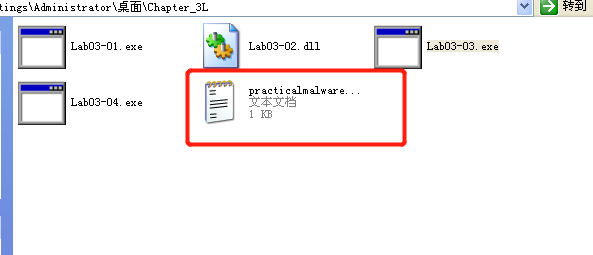
我们接下来获取这个恶意进程的id并进入exploer monitor进行筛选



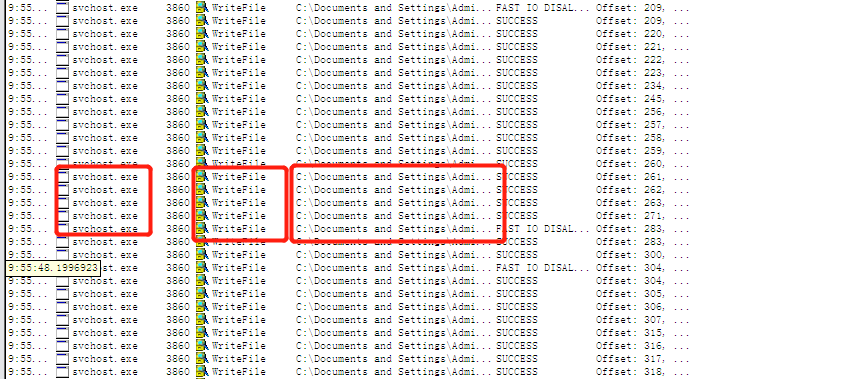
由于已经判断其所作操作与文件相关，所以只显示CreateFile、OpenFile、WriteFile等操作，果然我们观察到了其创建了一个文件，就在Lab03-03.exe的文件目录下

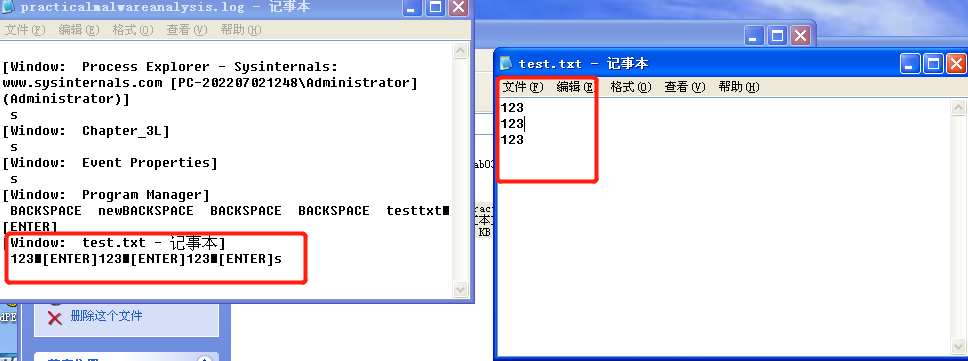


打开文件夹，发现确实出现了这样一个文本文档



为验证我们的猜想，我在桌面新建了一个文本文档，并进行了敲击打印，果然发现恶意进程创建的文本文档准确的记录了我的敲击内容：





yara规则：

我们从一开始的分析也可以看出，这个可执行文件被加密的很好，我们很难看出什么破绽，但是我们依然使用在内存中看到的字符串为它编写yara规则：

import "pe"
  
rule IsPE{
  
 meta:
  
 description = "检查文件是否为PE文件"
  
 author="wbf"
  
 date="2022-10-12"
  
 condition:
  
 uint16(0) == 0x5A4D and //“MZ”头
  
 uint32(uint32(0x3C)) == 0x00004550 // “PE”头
  
  
}
  
  
rule smallFileSize{
  
 meta:
  
 description = "检查文件小是否大概率为exe文件"
  
 author="wbf"
  
 date="2022-10-15"
  
 condition:
  
 filesize<500KB
  
}
  
  
rule mayBeMaleware{
  
 meta:
  
 description = "检查文件小是否大概率为maleware文件"
  
 author="wbf"
  
 date="2022-10-15"
  
 strings:
  
 $a = { E8 00 00 00 00 }
  
 condition:
  
 $a at pe.entry\_point
  
}
  
rule isAService{
  
 meta:
  
 description = "检查文件是否可能记录了用户的键盘记录"
  
 author="wbf"
  
 date="2022-10-23"
  
 strings:
  
 $a = "[ENTER]" nocase
  
 $b = "[SHIFT]" nocase
  
 $c = "[TAB]" nocase
  
 $d = "[CTRL]"
  
 $e = "[DEL]" nocase
  
 $f = "[CAPS LOCK]" nocase
  
 condition:
  
 2 of ($a,$b,$c,$d,$e,$f) and smallFileSize and IsPE and d
  
}

lab4：

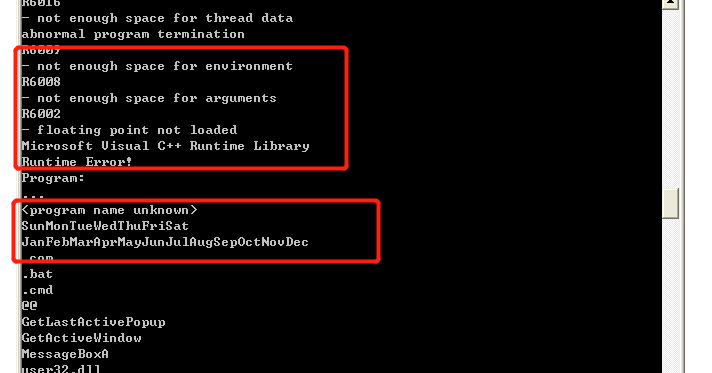
1、当你运行这个文件时，会发生什么呢？

2、是什么原因导致动态分析无法有效实施？

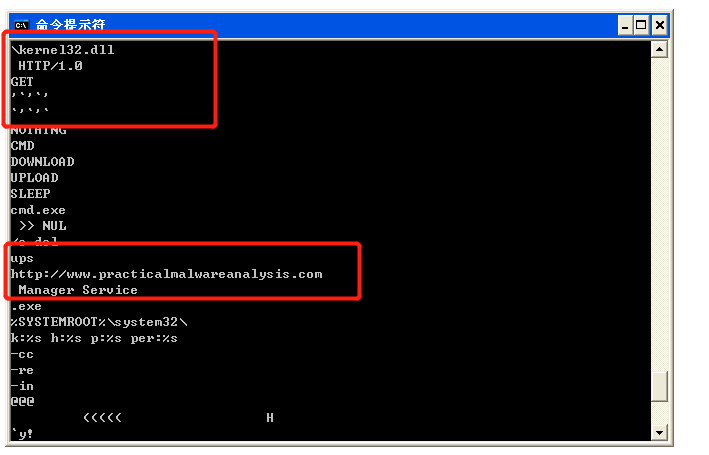
3、是否有其他方式来运行这个程序？

首先还是使用静态分析技术：查看其存在的字符串

首先很有意思的一点，这个恶意程序记录了12个月份以及每周星期数的简写，并且给出了很多提示符，包括空间、参数个数不够等等，



其次，毫无疑问他是一个恶意程序，因为他包含我们熟悉的这个网站，并且其也包含了联网服务

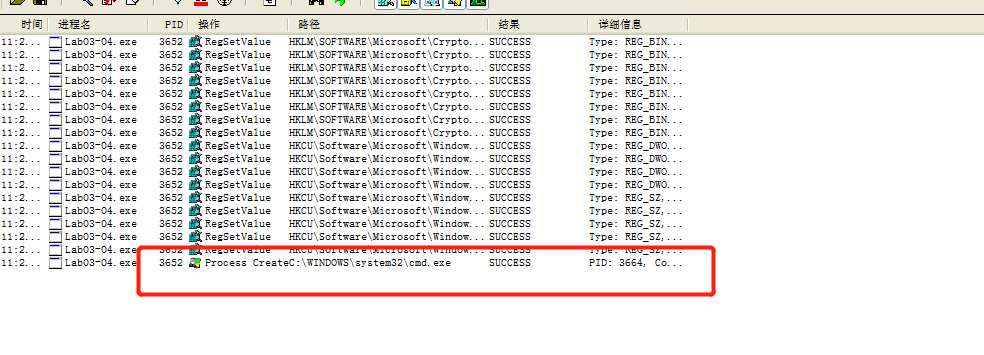


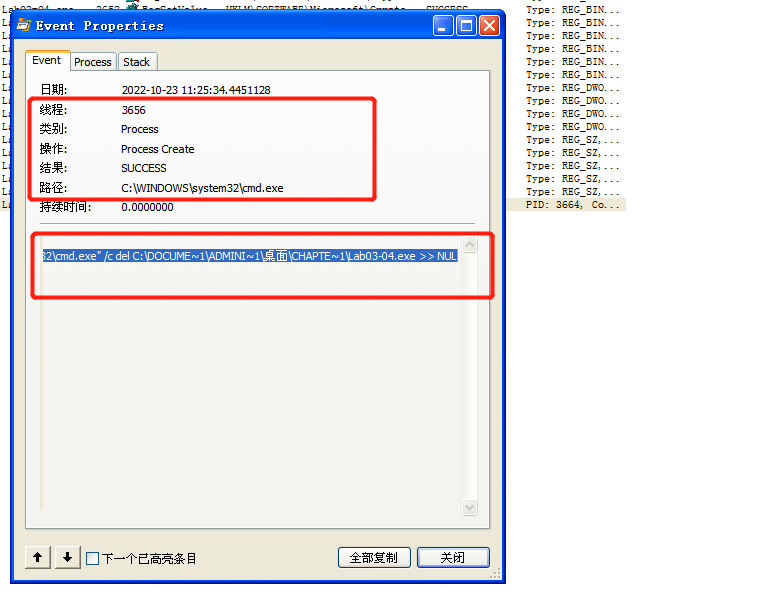
最后，-in -cc 这样的形式非常像命令行调用，同时我们也看到了cmd.exe，那么我们有理由相信这可能是一个命令行调用的程序



但让我打开眼界的是，当我点开这个恶意程序的时候他竟然删除了自己，并且及时我尝试用命令行去打开它也无济于事，最后他都会很干净的删除自己，我们在process exploer中也没有观察到可能的新产生的恶意进程。

但是，我们通过在process monitor中设置过滤器发现了这个恶意程序的一些蛛丝马迹：这个程序确实进行了了自我删除





yara：由于我们并不知道这个恶意程序具体做了什么，我们还是只能分析出他具有联网功能与使用cmd功能，我们尝试为其编写yara规则：

import "pe"
  
rule IsPE{
  
 meta:
  
 description = "检查文件是否为PE文件"
  
 author="wbf"
  
 date="2022-10-12"
  
 condition:
  
 uint16(0) == 0x5A4D and //“MZ”头
  
 uint32(uint32(0x3C)) == 0x00004550 // “PE”头
  
  
}
  
  
rule smallFileSize{
  
 meta:
  
 description = "检查文件小是否大概率为exe文件"
  
 author="wbf"
  
 date="2022-10-15"
  
 condition:
  
 filesize<500KB
  
}
  
  
rule mayBeMaleware{
  
 meta:
  
 description = "检查文件小是否大概率为maleware文件"
  
 author="wbf"
  
 date="2022-10-15"
  
 strings:
  
 $a = { E8 00 00 00 00 }
  
 condition:
  
 $a at pe.entry\_point
  
}
  
rule connectToNet{
  
 meta:
  
 description = "检查是否可能访问网络"
  
 author="wbf"
  
 date="2022-10-22"
  
 strings:
  
 $a = "www"
  
 $b = "HttpSendRequest"
  
 $c = "http"
  
 $d = "https"
  
 $e = "InternetConnect"
  
 $f = "HttpOpenRequest"
  
 condition:
  
 (($a or $d or $c) and($b or $e or $f)) and smallFileSize and IsPE
  
}
  
rule maybeCMD{
  
 meta:
  
 description = "检查是否可能需要通过cmd调用"
  
 author="wbf"
  
 date="2022-10-22"
  
 strings:
  
 $a = "cmd.exe"
  
 $b = "cmd" nocase
  
 $c = "-in"
  
 $d = "-re"
  
 $e = "-cc"
  
 condition:
  
 (($a or $b) and ($c or $e or $d)) and smallFileSize and IsPE
  
}

动态分析的优点：

可以精准的把控恶意程序真正想要进行的破坏，可以准确的得知恶意程序的攻击目的、攻击手段、攻击途径以及攻击效果，帮助我们更好的全方位的了解一个恶意程序

动态分析的缺点：

真正运行恶意程序可能会对主机造成一定程度的破话，并且如果不能完整的发现恶意代码的全部功能很可能造成病毒的释放泄露，比如一些蠕虫可能顺着网络进行传播。这就需要我们打造一个完全封闭的体系（如虚拟机、虚拟网络）来控制病毒的传播。

### 第四部分：实验心得：

通过本次实验，我掌握了一些基本动态分析工具的是使用，学会了监听进程并通过筛选过滤出有用的进程动作，同样学会了使用网络抓包工具对网络数据进行抓包分析，并且结合前面学习过的静态分析技术辅助进行恶意代码分析。

同时也对虚拟机技术有了更深一步的了解，对一个完全封闭、安全的系统环境有了更加深刻的了解。