

**实验报告**

**软件安全-恶意代码实验**

**专业班级： 信安XXXX班**

**学 号： U2021XXXXX**

**姓 名： MTX**

**指导教师： 刘 铭**

**报告日期： 2023年12月20日**

**网络空间安全学院**

**要 求**

**（略）详见学习通作业要求**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 基本功能实现 | 代码分析 | 文档格式规范 | 创新扩展功能 | 总分 |
| 25 | 25 | 35 | 15 | 100 |
|  |  |  |  |  |

目 录

[一、实验过程记录 4](#_Toc155614293)

[任务一：在windows下的两次弹窗 4](#_Toc155614294)

[任务二：修改PE入口点，只显示弹窗2 6](#_Toc155614295)

[任务三：重定位 9](#_Toc155614296)

[任务四：获取kernel32.dll的首地址 12](#_Toc155614297)

[任务五：获取关键函数地址 13](#_Toc155614298)

[扩展功能：利用C++实现病毒 13](#_Toc155614299)

[二、实验总结 17](#_Toc155614300)

[三、目标达成度自我评价 17](#_Toc155614301)

[参考文献 18](#_Toc155614302)

## 实验过程记录

实验环境：

操作系统：Microsoft Windows 10 家庭版，X86-based PC

编译工具：masm32

16进制编辑器：HxD

反汇编工具：OllyDBG

内存查看工具：VMMap

### 任务一：在windows下的两次弹窗

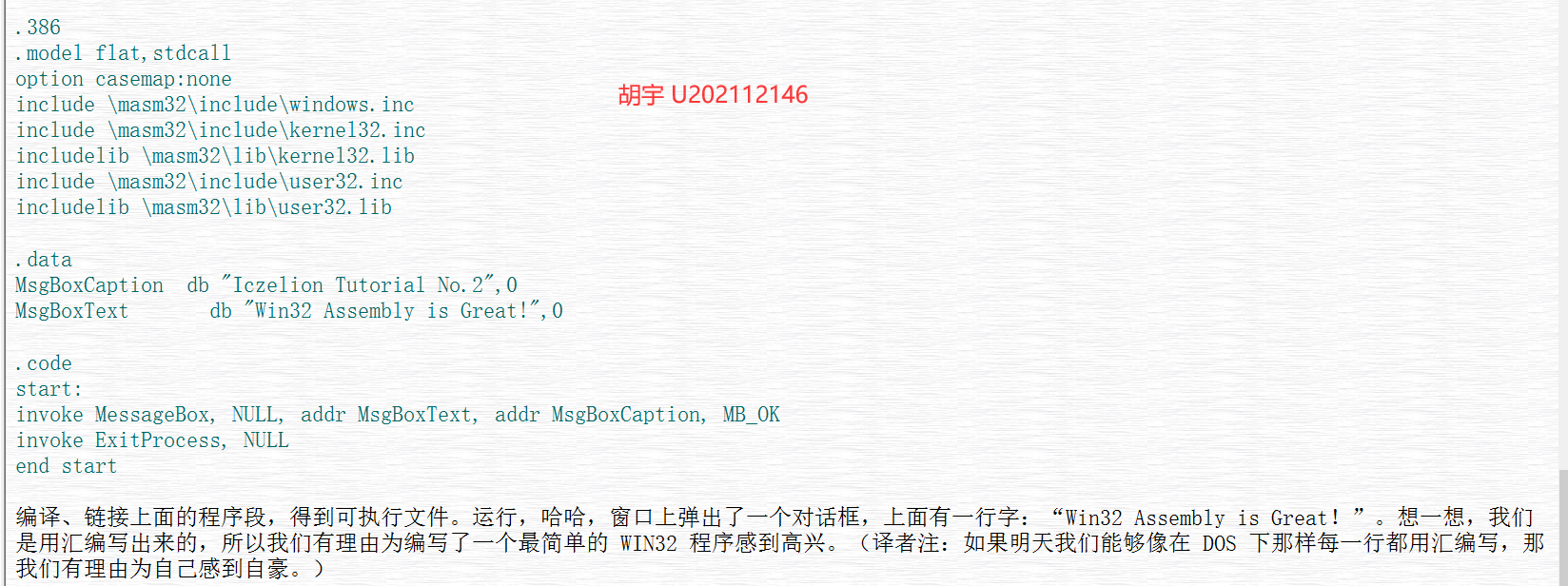


图1 Win32汇编弹窗模板代码

1.学习win32汇编的教程，根据图一模板编写弹出对话框的汇编代码，目标汇编代码功能为实现两次弹窗，如图2所示。

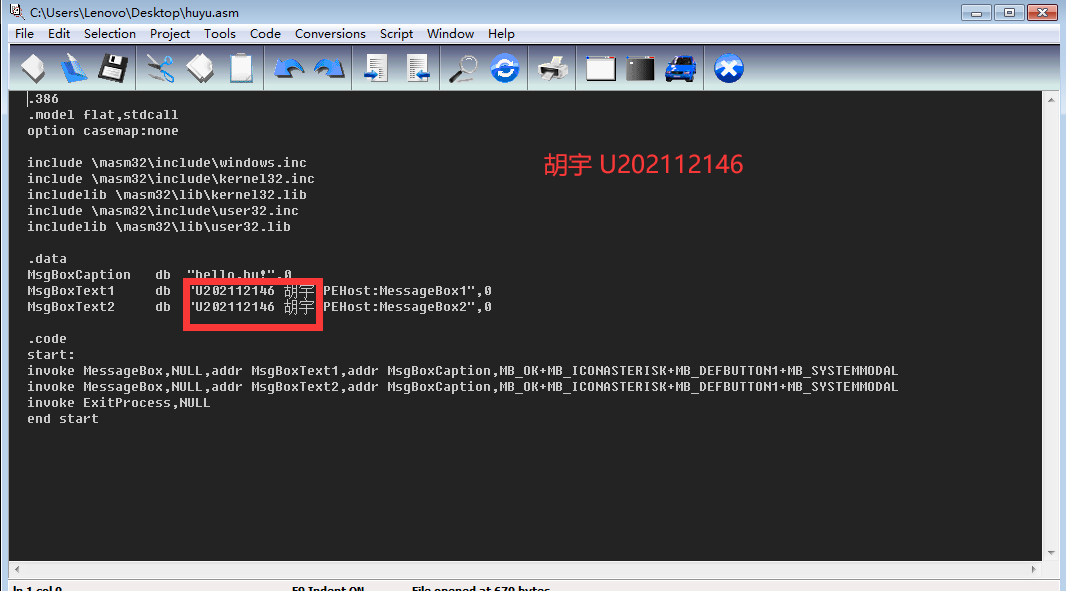


图2 两次弹窗汇编程序

2.写一个设置路径的脚本文件setpath.bat，具体内容如图3。

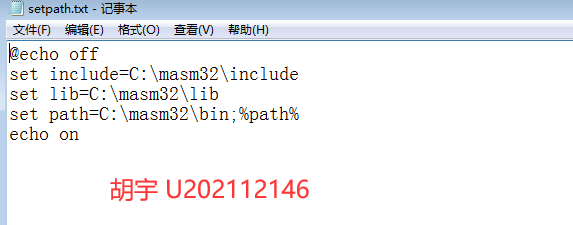


图3 setpath.bat文件

3.在终端先运行setpath.bat，配置路径，然后使用ml /coff /Cp huyu.asm /link /subsystem:windows /section:.text,rwe命令，命令使用 MASM 编译 "huyu.asm" 汇编文件，生成一个 COFF 目标文件，然后进行链接以创建一个带有特定节配置的 Windows 可执行文件，用于存放可执行代码，如图4所示。

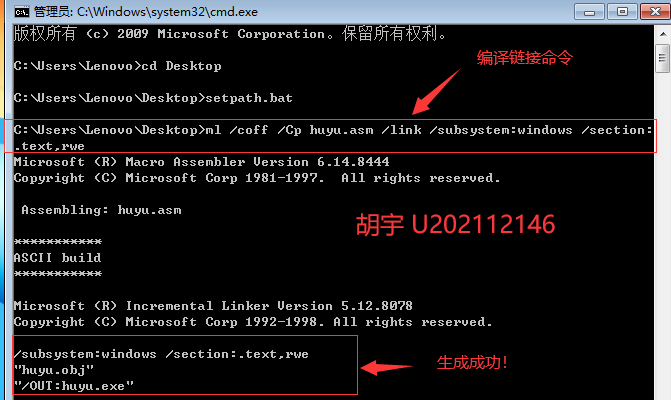


图4 成功生成可执行文件

4.点击桌面生成的可执行文件huyu.exe，如图5所示，成功运行，实现两次弹窗，如图6，7所示。

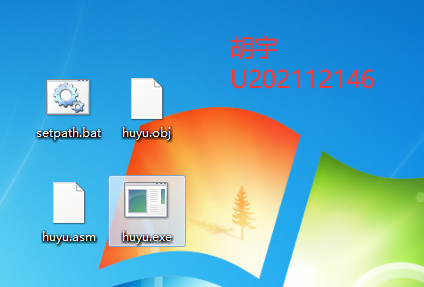


图5 生成的huyu.exe文件截图

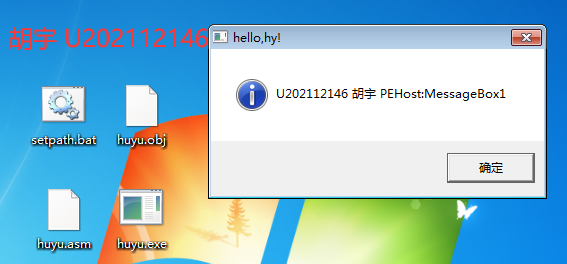


图6 第一次弹窗截图

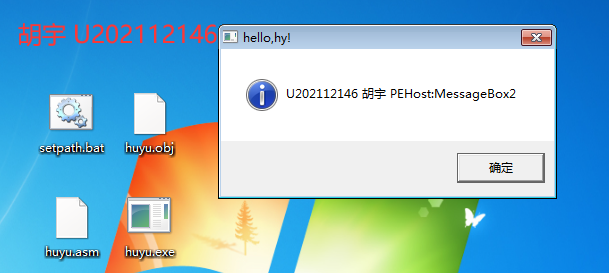


图7 第二次弹窗截图

### 任务二：修改PE入口点，只显示弹窗2

1．首先，我们很容易通过OD的可执行模块，便可以找到huyu.exe，如图8



图8 huyu.exe

2．一般来说exe文件的ImageBase值为00400000（如图9所示），能发现两个弹窗的虚拟地址分别为00401000，00401016，如图10所示，可以查看故第二个弹窗的偏移量为00001016



图9 PE文件头地址（即PE基址）

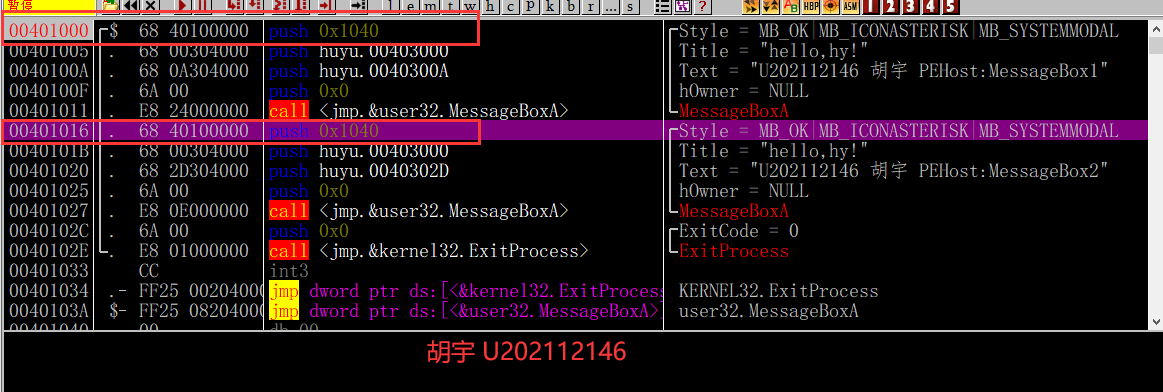


图10 两个弹窗的虚拟地址

3.现在我们只需要找到程序入口地址所在的地址，便将其改为第二个弹窗相对基址的偏移量即可。根据VA(虚拟地址) = ImageBase(PE基址) + RVA(相对虚拟地址)，如图11，我们知道PE文件头的位置在004000B0

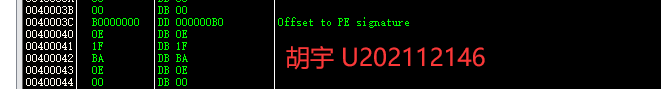


图11 PE文件头地址

找到004000B0那个区间，便可以找到最重要的程序入口字段AddressOfEntryPoint，如图12所示，为004000D8，减去基址00400000，即为000000D8



图 12 AddressOfEntryPoint地址

4.利用HxD打开huyu.exe，将000000D8处记录的地址数据由0001000改为0001016即可，如图13，14所示，程序就会直接跳过第一个弹窗，直接执行第二个弹窗。

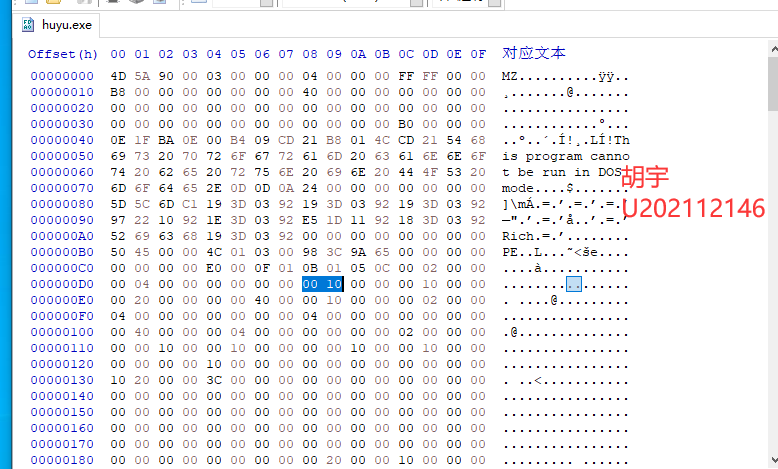


图13 修改之前的huyu.exe

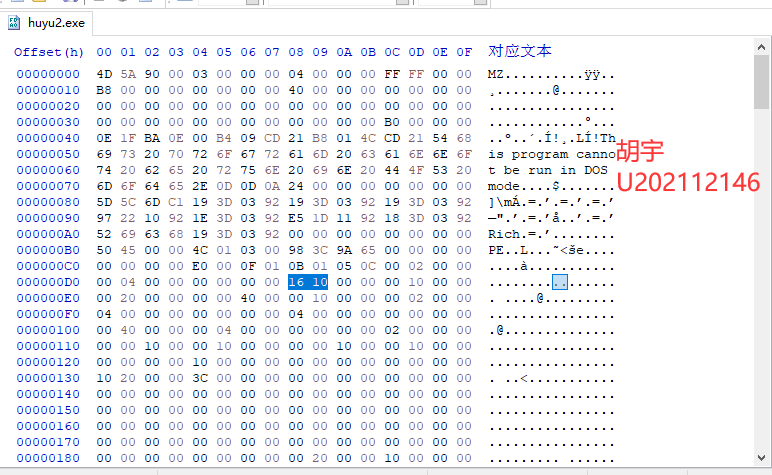


图14 修改之后的huyu.exe

5.运行修改之后的huyu.exe,发现其只触发第二个弹窗，如图15所示，实验成功！

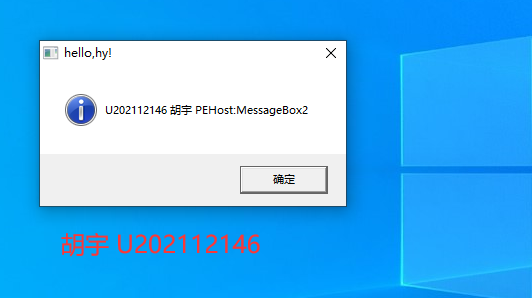


图15 仅出现第二个弹窗示意图

### 任务三：重定位

1.使用MASM32 Editor打开win7virus.asm,发现因为不同编码方式的问题会出现乱码，如图16所示，将其用txt记事本打开，发现没有乱码，将记事本种内容复制过来即可。



图16 文件乱码

2.找到需要修改的代码区，补全代码即可，代码作用是计算 "hy\_des" 过程相对于当前指令的偏移量，并将结果存储在寄存器 ebx 中，然后返回到调用该过程的地方，如图17所示。



图17 代码补全结果

call hy\_des：这条指令将当前的程序控制流转移到标签为 "hy\_des" 的地方，即调用了一个名为 "hy\_des" 的过程。hy\_des:：这是一个标签，表示过程的起始位置。pop ebx：这条指令将栈顶的值弹出并存储到寄存器 ebx 中。通常，这用于恢复调用过程之前的寄存器状态。sub ebx, offset hy\_des：这条指令计算偏移量，并将其从寄存器 ebx 中减去。在这种情况下，它计算了 "hy\_des" 标签相对于当前指令的偏移量。ret：这条指令表示过程的结束，并将程序控制流返回到调用该过程的地方。

3.接着同任务一，利用ml /coff /Cp win7virus\_hy.asm /link /subsystem:windows /section:.text,rwe命令编译链接即可，如图18.



图18 编译成功生成可执行程序

4.运行可执行程序，大致的感染过程如下图19。

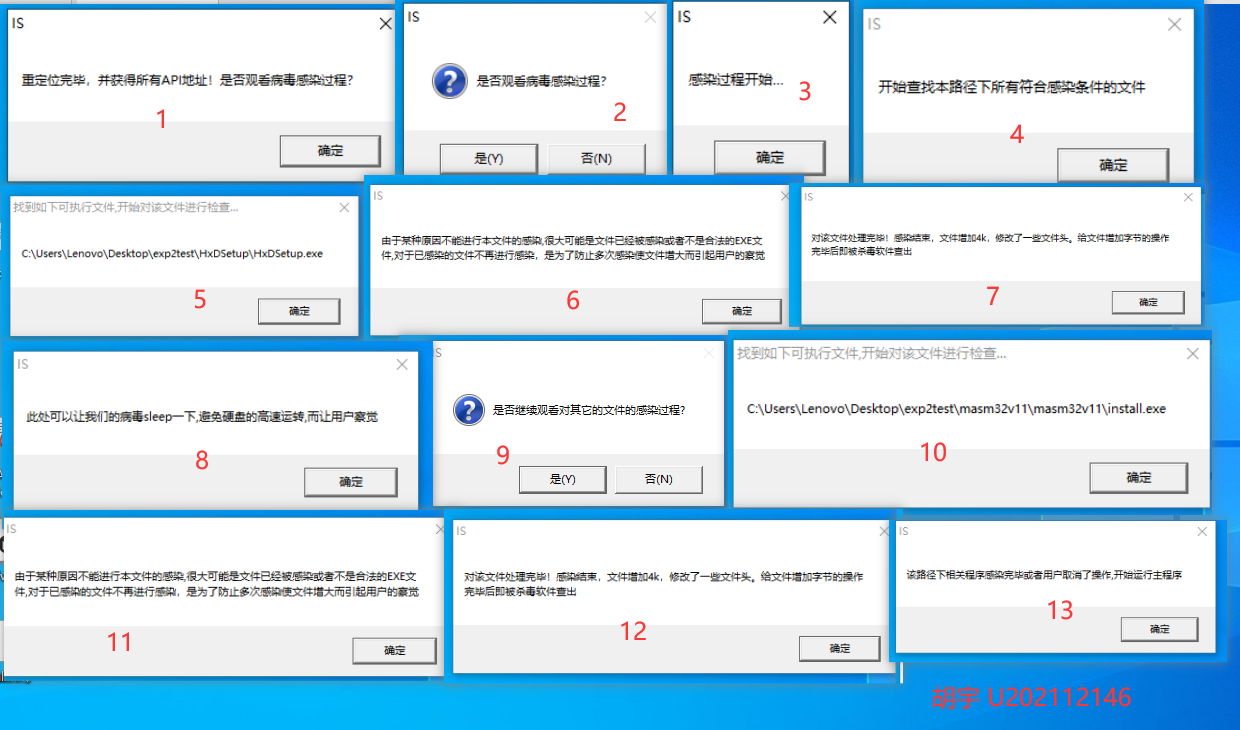


图19 感染过程

具体感染了指定文件夹下的huyu.exe和HxDSetup.exe两个文件。

5.运行被感染文件huyu.exe进行验证和示范，如图20所示，可知被感染文件运行时会先运行病毒文件再运行自己的文件，从而继续去传播感染其他文件。

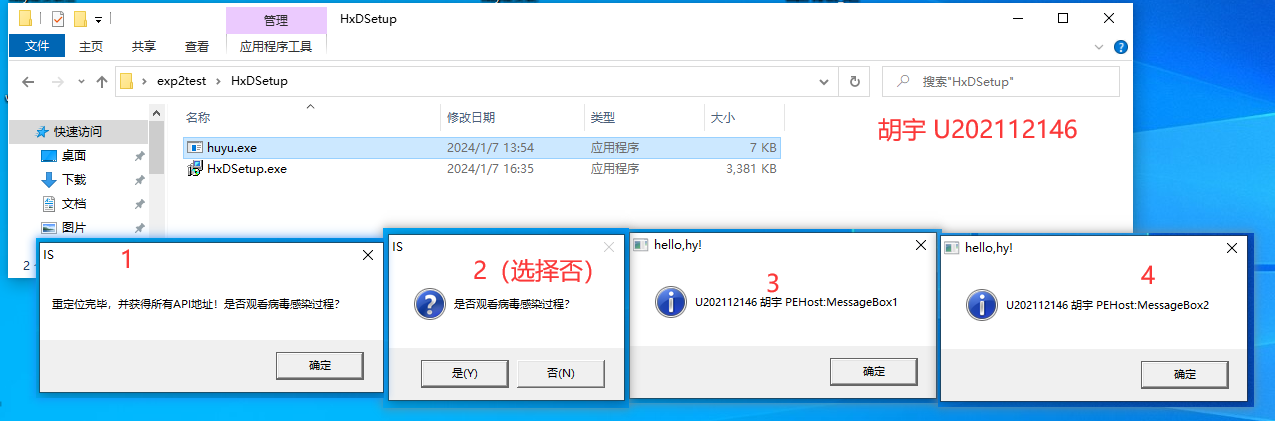


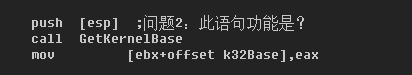
图20 被感染文件运行示意图

6.回答win7virus.asm注释中的问题

问题1：重定位功能

答：该功能具体已在任务3中实现了。

问题2



答：push [esp] 的作用是保存 ESP 寄存器指向的内存单元的值到栈中，以实现保存函数地址，这里是保存Kernel32.dll

库中某函数的地址。

问题3和问题4



答：问题3中语句的功能是：对寄存器 edi 的值进行按位与运算。具体地说，它将 edi 的值与 0FFFF0000h 进行按位与运算，目的是将 edi 的低 16 位清零，只保留高 16 位的值。这样做的目的是将 dwKernelRet 中指向内存地址的最后两个字节清零，以便后续用于进行比较。

问题4语句的功能是：这段代码通过一系列操作来验证指定地址处的内存是否包含有效的PE文件（可执行文件）。

* @@: 标签是一个代码标记，通常用于循环或跳过块中的代码。
* cmp word ptr [edi], IMAGE\_DOS\_SIGNATURE：这条指令将 edi 指向的内存地址处的值与 IMAGE\_DOS\_SIGNATURE 进行比较。如果比较结果不相等，说明内存位置 edi 不是 kernel32.dll 的基址，则跳转到 PageError 标签。
* jne PageError：如果比较结果不相等，跳转到 PageError 标签，表示发现了无效的 DOS 头。
* mov esi, edi：将 edi 的值赋给 esi 寄存器，作为备份。
* add esi, [esi+003Ch]：将 esi 的值与 esi 指向的内存地址偏移量为 3Ch 的值相加。这样可以获取 PE 文件头的地址。
* cmp word ptr [esi], IMAGE\_NT\_SIGNATURE：这条指令将 esi 指向的内存地址处的值与 IMAGE\_NT\_SIGNATURE 进行比较。如果比较结果不相等，说明内存位置 edi 不是 kernel32.dll 的基址则跳转到 PageError 标签。
* jne PageError：如果比较结果不相等，跳转到 PageError 标签，表示发现了无效的 NT 头。
* mov dwReturn, edi：将 edi（即找到的kernel32.dll 的基址）的值赋给 dwReturn，作为返回值。
* jmp @f：无条件跳转到当前位置后的标签（通常用于循环或跳过块中的代码）。
* PageError: 标签是用于处理验证失败的错误情况。
* sub edi, 010000h：将 edi 减去 010000h，相当于将 edi 的值减去 65536。
* cmp edi, 070000000h：将 edi 与 070000000h 进行比较，检验是否遍历了整个地址空间。如果 edi 小于 070000000h，则跳转到 @f 标签。
* jb @f：如果比较结果为小于，则无条件跳转到当前位置后的标签。
* jmp @b：无条件跳转到前面的 @@: 标签，形成一个循环。
* @@: 标签是一个代码标记，表示循环的起始。
* pop fs:[0]：将栈顶的值弹出并存储到 fs 寄存器的偏移地址为 0 的位置。
* add esp, 0ch：将 esp 寄存器的值增加 0Ch，相当于释放栈上的一些数据。
* popad：将栈中的数据依次弹出并存储到相应的通用寄存器中。
* mov eax, dwReturn：将 dwReturn 的值赋给 eax 寄存器。
* ret：返回到调用 GetKernelBase 的位置，并将 eax 中存储的值作为返回值。

### 任务四：获取kernel32.dll的首地址

1.在OD中根据找到kernel32.dll对应的代码部分,如图21，22所示

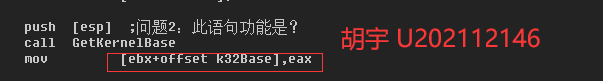


图21 源码中kernel32.dll对应部分

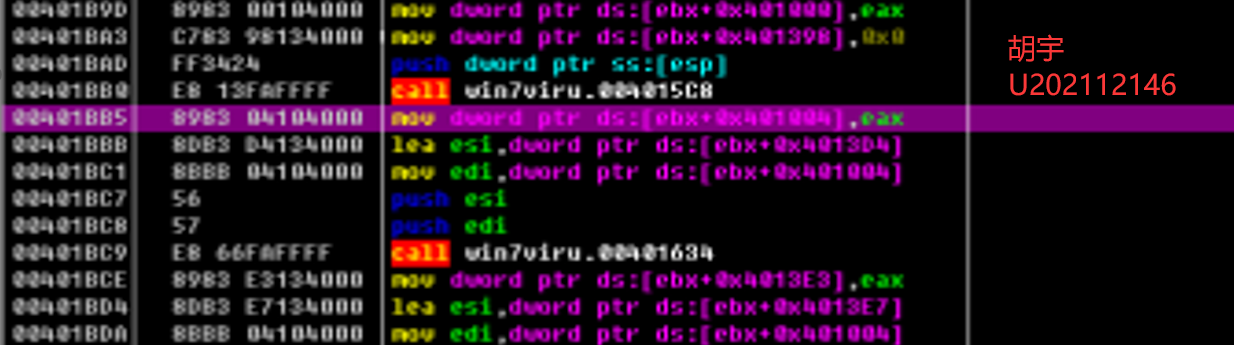


图22 OD中kernel32.dll对应部分

2通过之前分析的问题4可以得知GetKernelBase 函数能获取 kernel32.dll 的基址并存储到 dwReturn 变量中返回，并将结果存在了eax中。

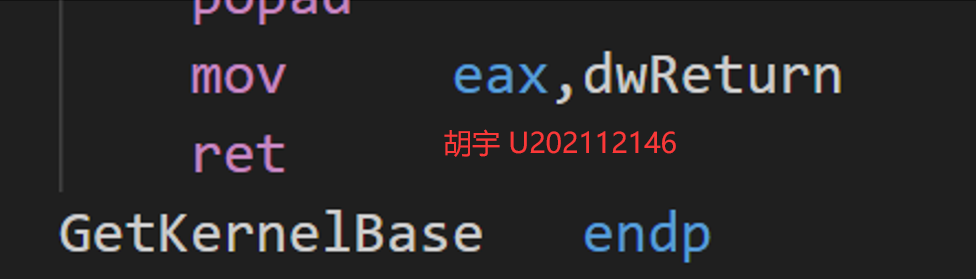


图23 GetKernelBase部分代码

3.故在此设置硬件断点，运行即可，如图24，eax中就是目标函数首地址。



图24 kernel32.dll首地址示意图

### 任务五：获取关键函数地址

1.查看病毒的asm源文件，如图25，不难发现关键函数GetProcAddress，LoadLibraryA

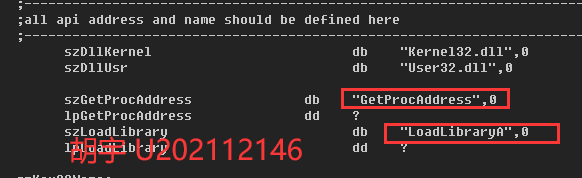


图25 关键函数

2．在OD中打开win7virus.asm，逐步执行，其实大体思路和任务4一样，关键函数地址都会存在eax中。结果如图26，27所示。

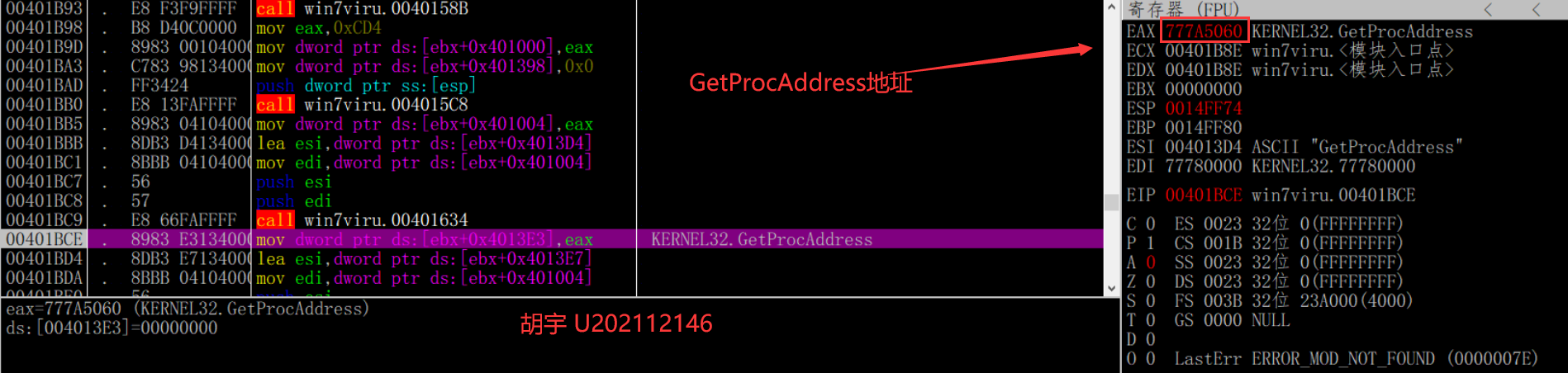
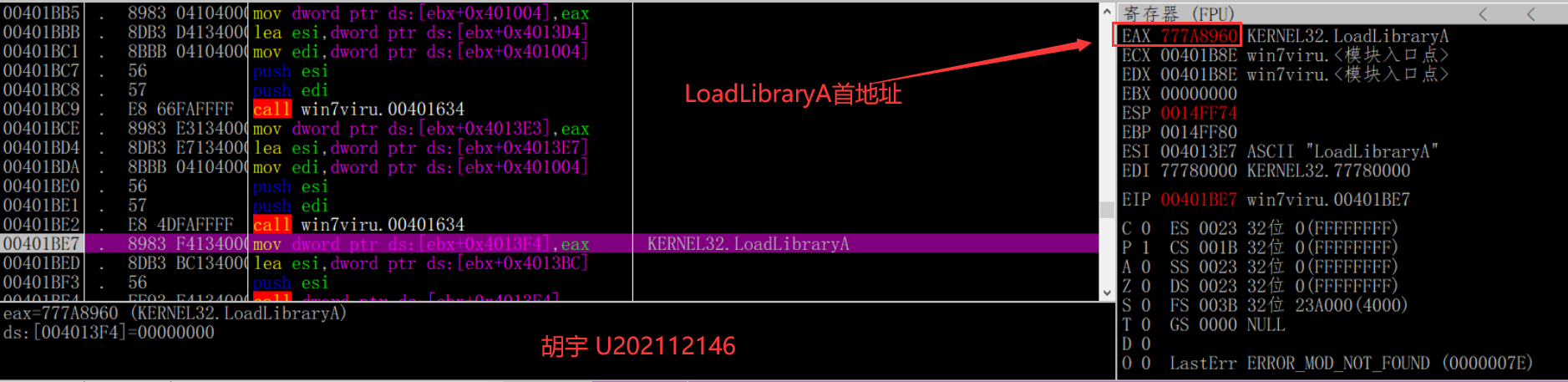


图26 GetProcAddress地址

图27 LoadLibraryA地址

### 扩展功能：利用C++实现病毒

为了实现一些其他的扩展功能，我利用C++实现了一个简易病毒，病毒具体功能如下：能感染某指定目录下所有PE文件，能够实现自我复制，能够添加一些垃圾文件。实验如下：

我们在CODE文件夹下进行实验，先开始没有感染时如图27，28所示，文件夹中仅有三个文件，且test.exe运行正常，输出为abcde。

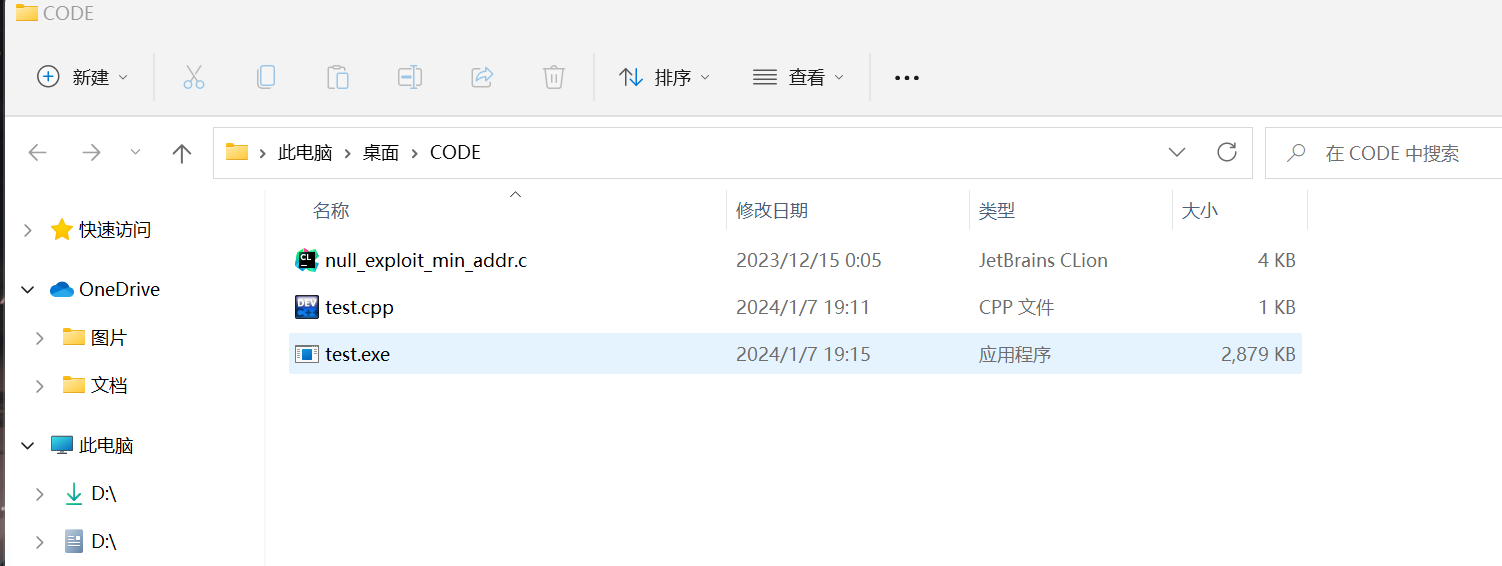


图27 未感染文件夹示意图

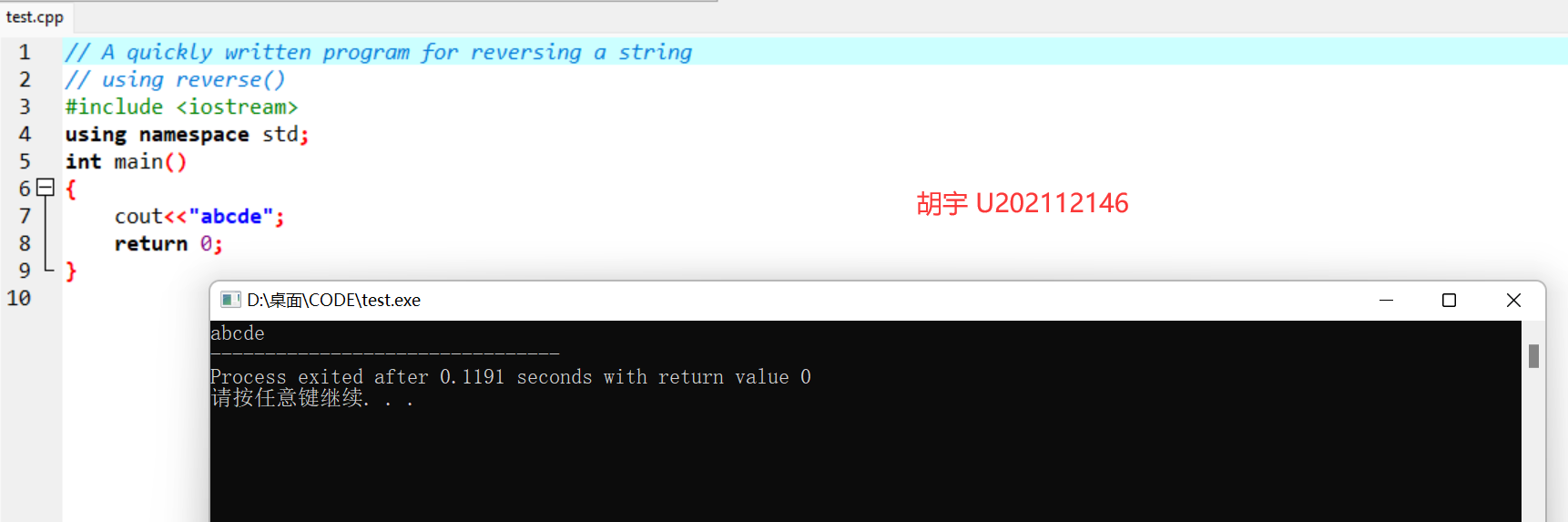


图28 未感染test.exe输出结果

然后利用编写病毒文件进行感染，过程如图29所示，可以发现感染后文件夹中多了四个文件，LitterFile的三个是随机生成的垃圾文件，1234.exe是人为制造的恶意垃圾文件。

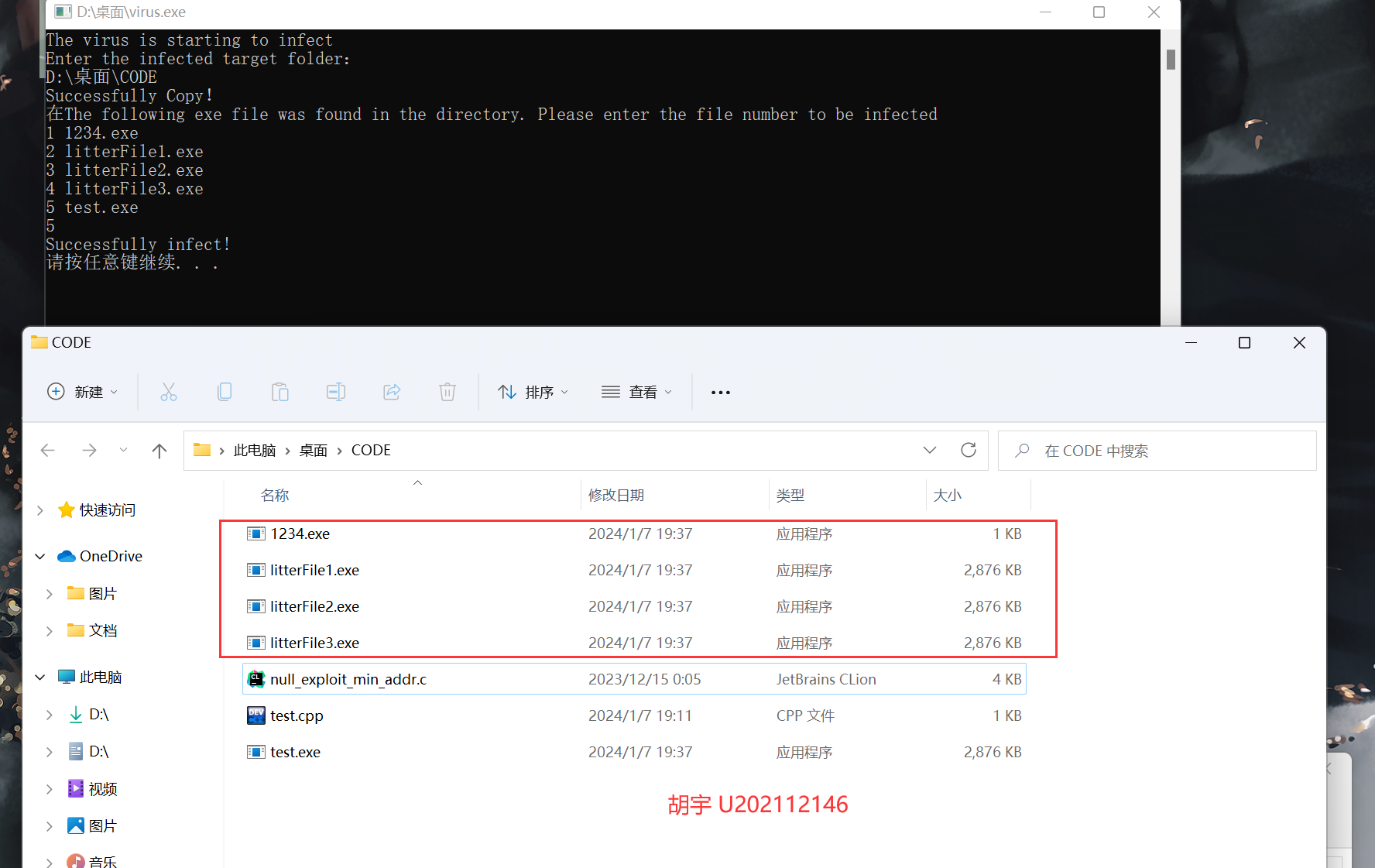


图29 感染后的文件夹示意图

同时，感染之后再运行test.exe进行验证，结果如图30所示，发现已经成功感染！

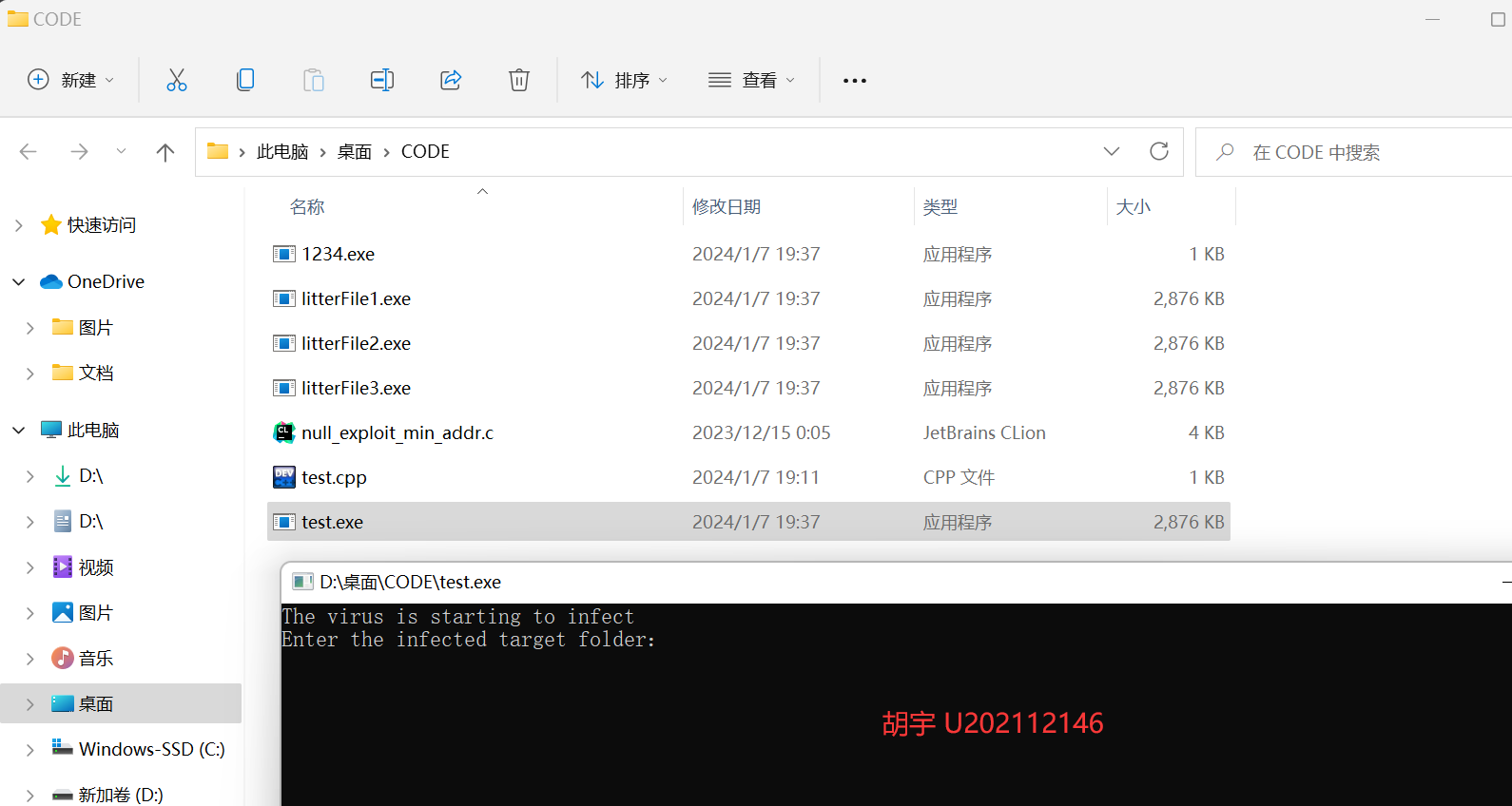


图30 感染成功示意图

接下来对病毒源代码进行一定的分析，文件采用模块化编程的思路分为4部分：创建可执行文件、寻找目录下的PE文件、文件复制、创造垃圾文件部分，分别如图31、32、33、34所示，实现代码比较简单，就不过多赘述了。

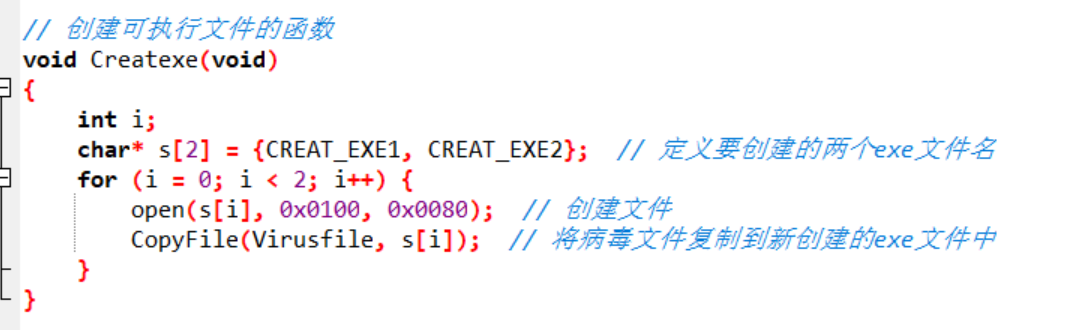


图31 创建可执行文件

Createxe函数创建了两个可执行文件，并将病毒代码复制到其中。



图 32 寻找目录下的PE文件

exe函数在目标文件夹中查找可执行文件并打印它们的名称。

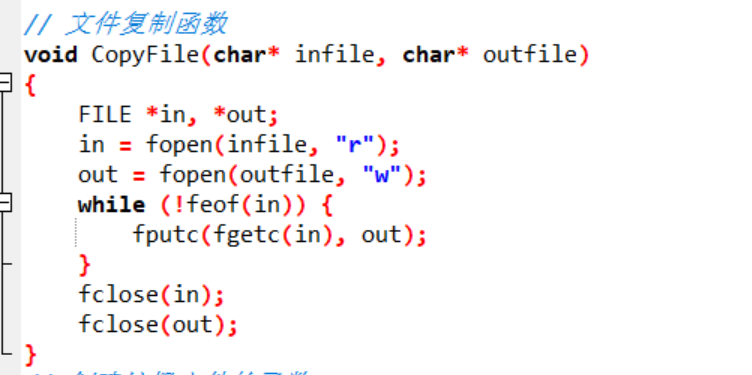


图 33 文件复制

CopyFile函数将一个文件的内容复制到另一个文件。

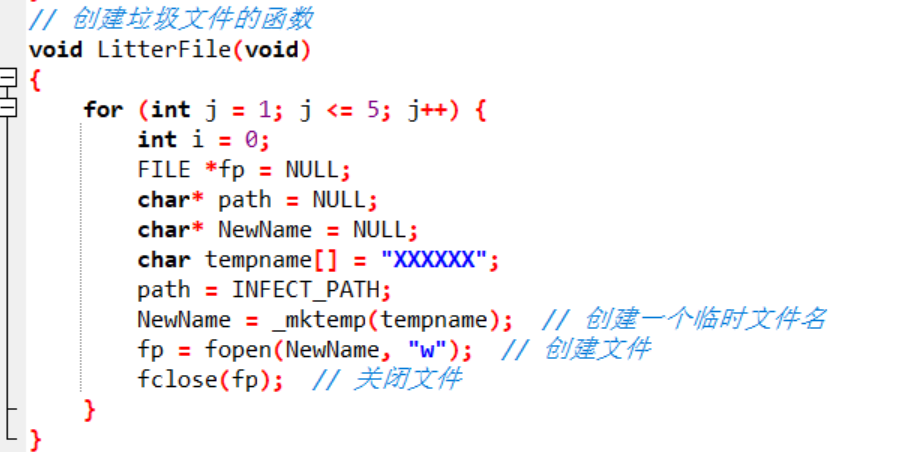


图34 创造垃圾文件部分

LitterFile函数在目标文件夹中创建了几个"垃圾"文件。 main函数是程序的入口点,它获取病毒文件本身的路径,提示用户输入要感染的目标文件夹,将病毒文件复制到目标文件夹中的多个文件中，创建"垃圾"文件,创建了两个可执行文件，并将病毒代码复制到其中,调用exe函数显示目标文件夹中可执行文件的列表,提示用户选择要感染的文件,将病毒代码复制到所选文件中,打印成功消息，并等待用户输入以退出。

## 二、实验总结

1、完成的基本功能

实现了两个病毒代码。

1）Asm病毒

这个是在老师给的asm代码基础上进行了补全，操作系统为win10家庭版，使用的开发工具有masm32，

16进制编辑器HxD，反汇编工具OllyDBG，恶意代码的基本功能为感染指定目录下的所有PE文件。

2）C++病毒

这个是自己学习搜集资料编写的病毒代码，操作系统为win11，使用的开发工具为vscode，devc++，恶意代码的基本功能为感染指定目录下的所有PE文件。

2、完成的扩展功能

C++病毒代码中，不仅可以感染指定目录下的PE文件，还能在该目录下创建没有意义的垃圾文件，创建人为制造的垃圾文件，人为制造的垃圾文件中就可以嵌入更多更丰富的攻击手段。

3、作品的优点

Asm病毒能完美实现老师提出的5个task，C++病毒代码采用模块化编写，每一模块代码逻辑清晰，接口设计得当，同时能实现感染某指定目录下所有PE文件，能够实现自我复制，能够添加一些垃圾文件等多种功能！主要模块功能如下：CopyFile函数将一个文件的内容复制到另一个文件，LitterFile函数在目标文件夹中创建了几个"垃圾"文件，exe函数在目标文件夹中查找可执行文件并打印它们的名称，Createxe函数创建了两个可执行文件，并将病毒代码复制到其中。

4、作品的不足之处

C++病毒中有些边界设计不够明晰，对于一些错误输入没有做判断和补救，同时还有一些功能例如：删除指定文件、Copy指定文件发送给他人等功能没有实现（要复习考试了，没时间写了，）。

## 三、目标达成度自我评价

通过实验，结合前面实验心得中的内容，在下面的表格中，完成自我评价。

|  |  |  |
| --- | --- | --- |
| 毕业目标 | 自我评价的具体内容 | **目标达成的满意度 自评**  ☑标记 |
| 毕业要求3设计/开发解决方案（解决方案） | 3.1设计和开发的全周期、全流程的方法和技术。在实验过程中，根据拟定的功能，设计出恶意代码的感染、触发、恶意表现等功能流程，并能开发相应的解决方案，编程实现。 | ☑非常满意  □满意  □一般  □不满意 |
| 毕业要求3设计/开发解决方案（解决方案） | 3.2领域特定需求完成基础部件、单元、算法的设计与开发。能理解恶意代码基本功能，对其中的基础部件、单元的关键代码（重定位、模块定位、函数查找、文件检索等），设计替代的算法，并编码实现。 | ☑非常满意  □满意  □一般  □不满意 |
| 毕业要求5使用现代工具（技术、工具） | 5.2选择、使用现代工具设计、预测、模拟与实现，分析局限。在完成实验过程种，通过资料查阅，选择、使用现代工具进行设计（masm32 vs vscode）；探查不同环境下，相同恶意代码的不同表现（win7 vs win10 vs Linux）；分析工具及操作系统选择带来的局限性。 | ☑非常满意  □满意  □一般  □不满意 |

## 参考文献

[1] <https://www.cnblogs.com/zheh/p/4008268.html>

[2] https://blog.csdn.net/DUXS11/article/details/132121414?ops\_request\_misc=&request\_id=&biz\_id=102&utm\_term=C++%E7%97%85%E6%AF%92&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduweb~default-1-132121414.142^v99^pc\_search\_result\_base4&spm=1018.2226.3001.4187

[3] Ed S. Lein, Michael J. Hawrylycz, Nancy Ao, Mikael Ayres, Amy Bensinger, Amy Bernard, et al. Genome-wide atlas of gene expression in the adult mouse brain. Nature, 2007, 445(7124): 168-176