2023-12-24

北京理工大学计算机学院

计算机病毒防治

实验四 汇编中的C结构实验报告

目录

[实验准备 3](#_Toc156227445)

[一、 实验环境和所用到的软件 3](#_Toc156227446)

[二、 基本知识 3](#_Toc156227447)

[Lab 6-1 3](#_Toc156227448)

[一、 由main函数调用的唯一子过程中发现的主要代码结构是什么 3](#_Toc156227449)

[二、 位于0x40105F的子过程是什么 4](#_Toc156227450)

[三、 这个程序的目的是什么 6](#_Toc156227451)

[Lab 6-2 7](#_Toc156227452)

[一、 Main函数调用的第一个子过程执行了什么操作 7](#_Toc156227453)

[二、 位于0x40117F的子过程是什么 8](#_Toc156227454)

[三、 被main函数调用的第二个子过程做了什么 8](#_Toc156227455)

[四、 在这个子过程中使用了什么类型的代码结构 12](#_Toc156227456)

[五、 在这个程序中有任何基于网络的暗示吗 12](#_Toc156227457)

[六、 这个恶意代码的目的是什么 13](#_Toc156227458)

[Lab 6-3 14](#_Toc156227459)

[一、 比较与Lab6-2的main函数调用，main中调用的新函数是什么 14](#_Toc156227460)

[二、 这个新的函数的参数是什么 14](#_Toc156227461)

[三、 这个函数包含的主要代码结构是什么 15](#_Toc156227462)

[四、 这个函数能够做什么 17](#_Toc156227463)

[五、 这个恶意代码中有什么基于主机的特征吗 19](#_Toc156227464)

[六、 这个恶意代码的目的是什么 19](#_Toc156227465)

[Lab 6-4 20](#_Toc156227466)

[一、 Lab 6-3和Lab 6-4中的main函数中的调用函数的区别 20](#_Toc156227467)

[二、 什么新的代码结构被添加到main中 20](#_Toc156227468)

[三、 这个实验解析HTML的函数和前面实验中的有什么区别 21](#_Toc156227469)

[四、 在连接到互联网的情况下这个程序会运行多久 22](#_Toc156227470)

[五、 在这个恶意代码中有什么新的基于网络的迹象吗 23](#_Toc156227471)

[六、 这个恶意代码的目的是什么 23](#_Toc156227472)

[实验总结 23](#_Toc156227473)

# 实验准备

1. 实验环境和所用到的软件

* Windows 10 x86
* IDA Pro等

1. 基本知识
2. 汇编中的C结构：

* If条件结构
* For循环结构
* While循环结构
* 数组
* 结构体

# Lab 6-1

1. 由main函数调用的唯一子过程中发现的主要代码结构是什么

使用IDA Pro打开Lab06-01.exe，结果如下：

图形用户界面, 文本, 应用程序

描述已自动生成

从上图可以看出，main函数唯一调用的子过程是sub\_401000。双击跳转到sub\_401000处，并按下空格键转换为图形模式。

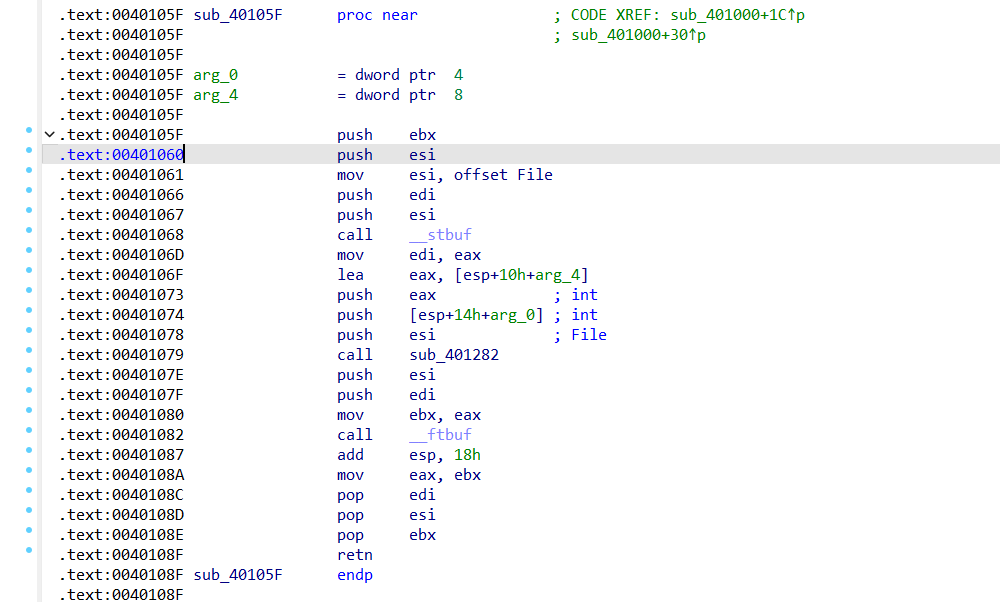
图形用户界面, 应用程序

描述已自动生成

从该图可以看出子函数的结构是if-else结构。其主要功能为根据InternetGetConnectedState调用结果来执行不同的条件。

1. 位于0x40105F的子过程是什么

如果只是分析位于0x40105F的子过程，我们会发现这个子过程调用了一个非常复杂的子过程，很难分析。



图示, 示意图

描述已自动生成

这时我们可以通过查看该函数的输入参数来推测该函数的功能。按下“X”键找出sub\_40105F的交叉引用。

图形用户界面, 文本, 应用程序

描述已自动生成

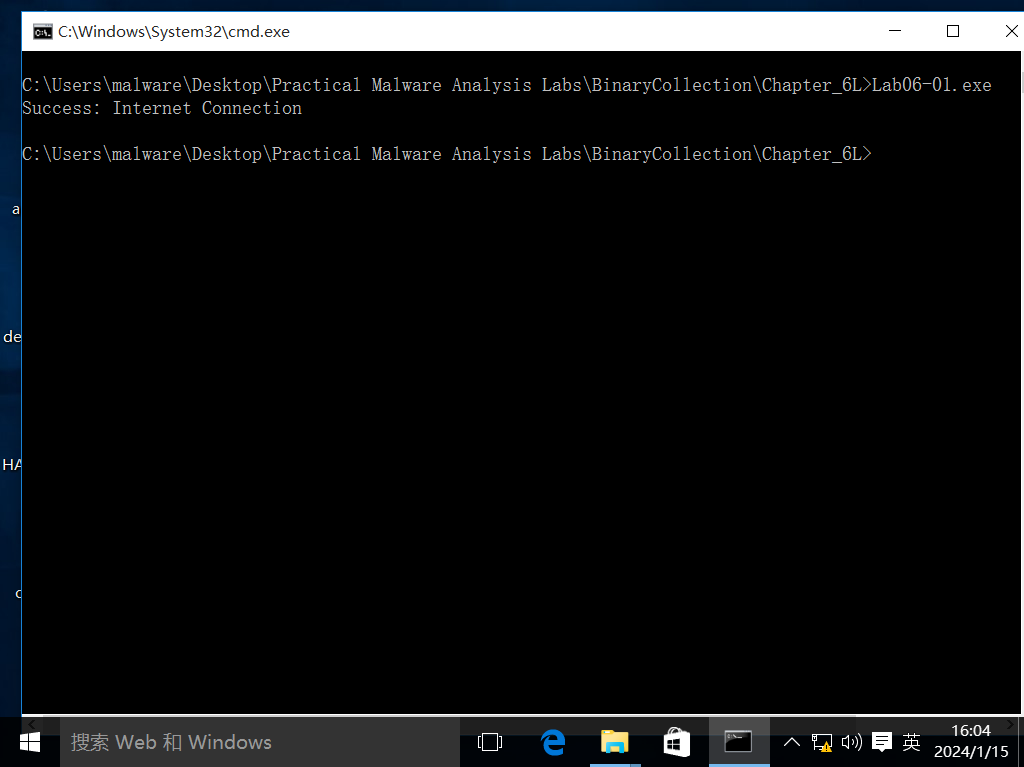
发现在子过程sub\_401000中的不同条件都调用了位于0x40105F的子过程。根据其的参数为一个字符串可以猜测这个子过程是一个输出函数。实际上IDA Pro不能识别printf函数，所以该函数实际上是printf函数，我们可以将其重命名为printf。

图形用户界面, 应用程序

描述已自动生成

1. 这个程序的目的是什么

综上所述，这个程序的作用是检测本地系统的网络连接状态，如果成功，则打印“Success:Internet Connection”并返回1，如果没有网络连接则打印“Error 1.1:No Internet”并返回0。



# Lab 6-2

1. Main函数调用的第一个子过程执行了什么操作

使用IDA Pro打开Lab06-02.exe，结果如下：

图形用户界面, 文本, 应用程序

描述已自动生成

可以发现main函数调用的第一个子函数为sub\_401000。双击跳转到sub\_401000，并按下空格键转换为图形模式。

图形用户界面, 应用程序

描述已自动生成

从该图可以看出，该子过程是一个if-else结构，和Lab6-1相似。

1. 位于0x40117F的子过程是什么

和Lab6-1中相似，通过函数参数来推测该函数应该是printf函数。

文本

中度可信度描述已自动生成

1. 被main函数调用的第二个子过程做了什么

被main函数调用的第二个子过程是sub\_401040，双击跳转到sub\_401040，结果如下：

图形用户界面, 文本, 应用程序

描述已自动生成

该子过程首先调用了InternetOpenA函数来初始化WinINet库的使用。然后调用了InternetOpenUrlA函数来打开URL<http://www.practicalmalwareanalysis.com/cc.htm>，然后将返回值赋给hFile并根据返回值来判断是否连接成功，如果没有连接成功，则输出“Error 2.1: Fail to OpenUrl”。

之后调用函数InternetReadFile来读取网页，并将读取到的数据按照从高地址向低地址方向放到Buffer中。如果返回值为0，说明读取失败，打印“Error 2.2：Fail to ReadFile”。

文本

中度可信度描述已自动生成

如果读取成功，则判断读取到网页是否以“<!--”开头，只是HTML中注释的格式。如果不是则输出“Error 2.3：Fail to get command”。

图形用户界面, 文本, 应用程序

描述已自动生成

如果是的话返回下一个字符并结束函数。

这里可以按下Ctrl+K键去修复此函数的堆栈，从而可以正确显示Buffer数组。单击Buffer的第一个字节，定义一个元素大小为1字节的512个元素数组。

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

1. 在这个子过程中使用了什么类型的代码结构

从以上分析可以看出，这个子过程使用了if类型的代码结构。

1. 在这个程序中有任何基于网络的暗示吗

从上述分析过程中，可以知道该程序使用Internet Explorer 7.5/pma作为作为HTTP的User-Agent字段、对http://www.practicalmalwareanalysis.com/cc.htm进行了数据访问（下载了该页面）。

文本

低可信度描述已自动生成

1. 这个恶意代码的目的是什么

继续分析该程序，如果成功解析到了命令，则程序会输出并休眠60s，否则直接退出。

图片包含 日程表

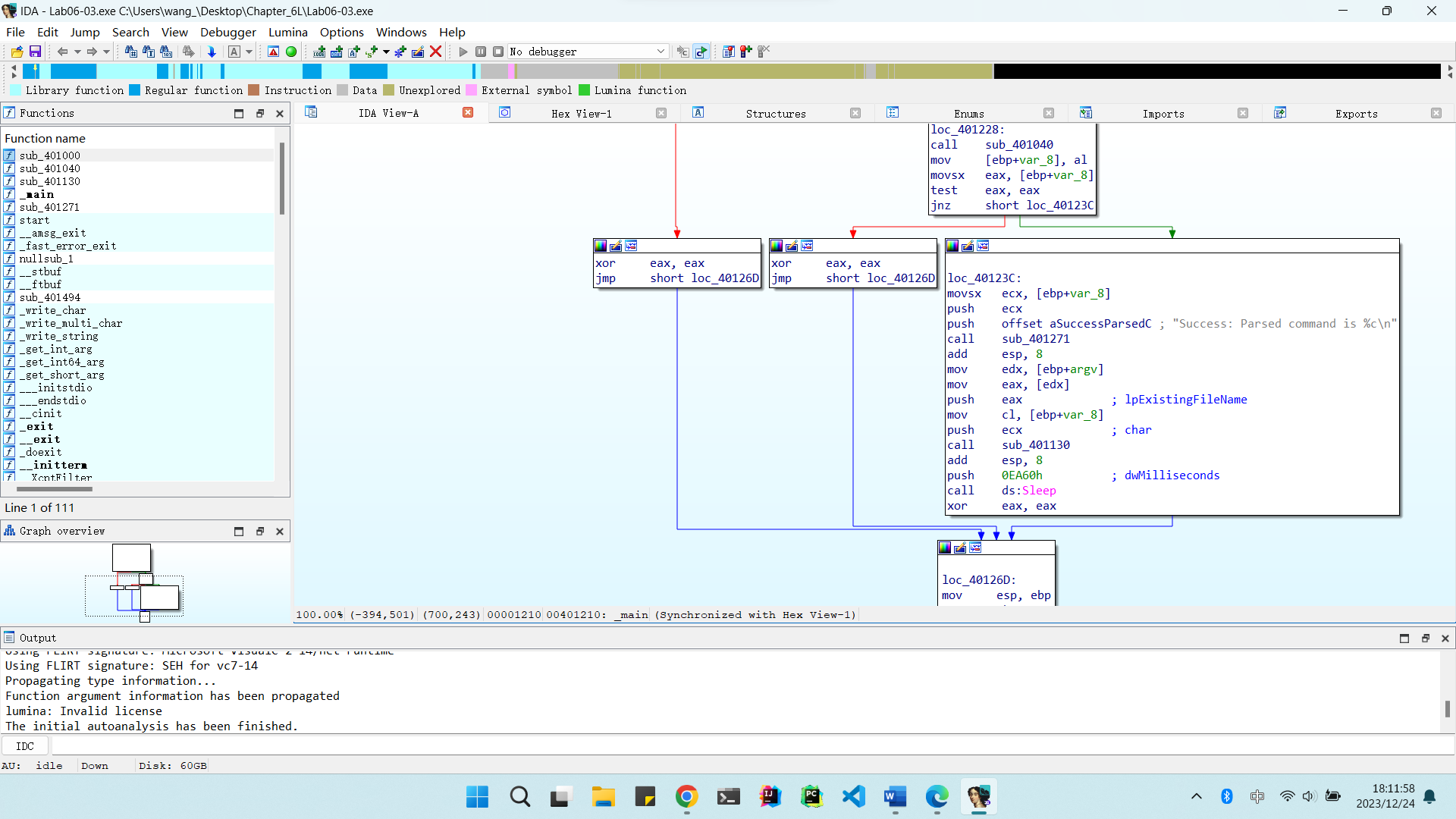
描述已自动生成

综上所述，该恶意软件检查活动的互联网连接，然后下载包含字符串的“<!--”开始的HTML页面，然后解析出来命令输出并休眠。

# Lab 6-3

1. 比较与Lab6-2的main函数调用，main中调用的新函数是什么

通过比较发现新调用的函数为sub\_401130。



1. 这个新的函数的参数是什么

从上图可以看出，该函数的参数有两个，第一个被标识为char，第二个被标识为lpExistingFileName。

文本

描述已自动生成

经过分析我们可以得出，第一个参数为调用sub\_401040的返回值，即HTML页面源码中注释标签后跟着的指令字符；第二个参数为main函数的argv参数中指向第一个字符串的指针，即程序名。

1. 这个函数包含的主要代码结构是什么

双击函数进行跳转，并按下空格键切换为图形模式。

电脑屏幕截图

描述已自动生成

先分析这个函数的开头部分，我们可以知道arg\_0存放的是从HTML中解析出来的指令，被存放到var\_8中，然后减去a的ASCII码，如果大于4，则跳转到def\_401153处。

图形用户界面, 文本, 应用程序

描述已自动生成

如果var\_8小于等于4则进入跳转表，根据不同指令执行不同代码。

图形用户界面, 文本, 应用程序, Word

描述已自动生成

综上所述，这个函数是一个switch结构。注意到当指令为d时有一个if结构。

当看到单个框转到多个不同的框时，我们可以猜测其为switch语句，可以通过查看代码逻辑和跳转表来确认。

1. 这个函数能够做什么

如果指令不是a、b、c、d、e，则进入默认分支。def\_401153是默认分支，输出“Error 3.2: Not a valid command provided”。

图形用户界面, 文本

描述已自动生成

如果指令为a，则进入loc\_40115A，其作用是在C:\\Temp下创建目录。

图形用户界面, 文本

描述已自动生成

如果指令为b，则将该程序复制到C:\\Temp\cc.exe。

图形用户界面, 文本, 应用程序

描述已自动生成

如果指令为c，则删除C:\\Temp\cc.exe。

文本

描述已自动生成

如果指令为d，则在注册表中创建“Software\Microsoft\Windows\CurrentVersion\Run\Malware”键，并将值设置为“C:\Temp\cc.exe”，以实现cc.exe在系统启动时的自启动。如果添加失败则输出“Error 3.1: Could not set Registry value”。

图形用户界面, 文本, 应用程序

描述已自动生成

如果指令为e，则睡眠100s。

图形用户界面, 文本

描述已自动生成

1. 这个恶意代码中有什么基于主机的特征吗

由以上分析可以，注册表键“Software\Microsoft\Windows\CurrentVersion\Run\Malware”和位于C:\\Temp\cc.exe都是该恶意软件基于主机的特征。

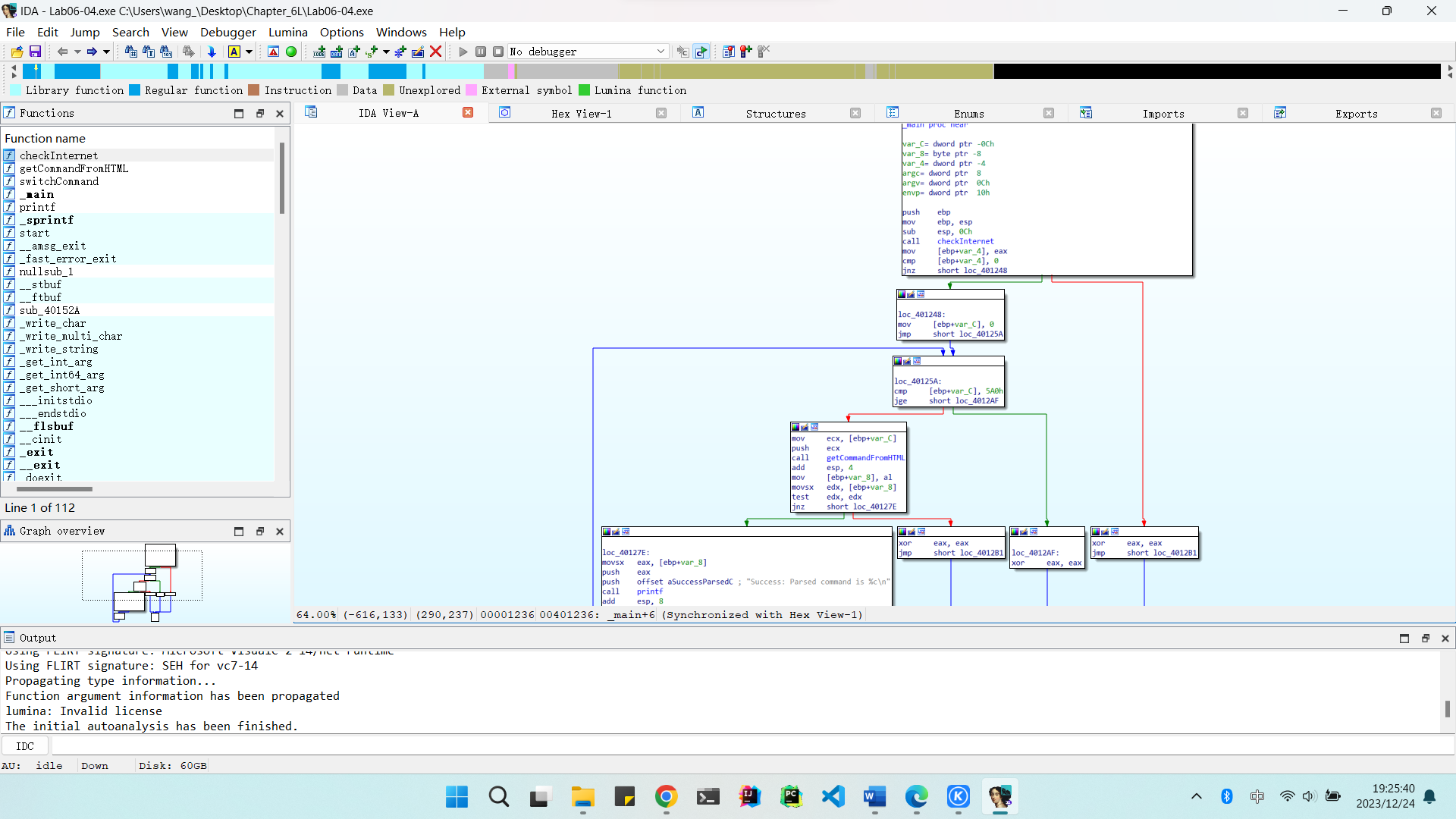
1. 这个恶意代码的目的是什么

综合以上分析，该恶意代码检查Internet连接，如果没有有效的Internet连接，程序终止。否则，程序会尝试下载一个网页，包含以“<!--”开头的HTML注释。程序从该注释中解析下一个字符，并在switch语句中使用该字符来确定在本地系统上执行的操作：删除文件、创建目录、设置注册表、复制文件或睡眠100秒。

# Lab 6-4

1. Lab 6-3和Lab 6-4中的main函数中的调用函数的区别

使用IDA Pro打开Lab06-04.exe，结果如下：



Main函数中调用的函数和之前实验的功能相似，这里我们将其重命名为更直观的函数名。

与Lab6-3的区别为调用解析命令函数之后出现了一个jmp指令向前跳转。

1. 什么新的代码结构被添加到main中

分析main函数中的指令，我们可以得到var\_C为计数变量，初始化为0，之后和1440进行比较，如果小于则执行循环体中的内容。

图形用户界面, 文本, 应用程序

描述已自动生成

在循环体结束后，会把var\_C加1，并跳转到比较处。

文本, 应用程序

描述已自动生成

从以上结构可以看出这是一个for循环。

1. 这个实验解析HTML的函数和前面实验中的有什么区别

首先从函数调用参数中看出，这个版本的函数需要传递一个参数，该参数为循环变量var\_C。

图形用户界面, 文本, 应用程序

描述已自动生成

之后双击跳转到函数，可以发现sprintf函数被调用，它用循环变量格式化User-Agent字符串。这意味着也就是说每一次网页访问请求时，HTTP的User-Agent值都不一样，且在User-Agent值的末尾会指明这是第几次发出访问请求。

文本

中度可信度描述已自动生成

1. 在连接到互联网的情况下这个程序会运行多久

忽略执行其他代码的时间，由于会执行1440次循环，每次循环中会休眠1分钟，所以这个程序会执行1440分钟，相当于24小时。但如果请求页面数据中指令字符有e，则会使程序休眠100秒，会增加程序运行时间。

图形用户界面, 文本

描述已自动生成

1. 在这个恶意代码中有什么新的基于网络的迹象吗

从上述分析可以得该恶意代码会请求同一页面多达1440次，而每次HTTP请求的User-Agent值都不一样，值为Internet Explorer 7.50/pma%d，其中%d值指明是第几次请求。

1. 这个恶意代码的目的是什么

综上所述，该恶意代码先会检查网络连接，有则进行1440次循环，每次循环做出以下行为：请求网页数据，从HTML源码中解析注释标签，然后取出指令字符，再根据指令字符来决定后续行为，可能会创建目录、复制文件、设置注册表键值、删除文件、休眠100秒，最后每次循环结尾会休眠60秒。

# 实验总结

由于这学期学校开设了汇编语言和计算机系统导论，所以这章的实验难度还行，很多内容都比较基础。本章也加深了我对IDA Pro的操作，对于常见的程序结构有了更为形象的认识，加上IDA Pro提供的图形的模式，很多C语言结构不用读代码就可以进行猜测。