****

**《软件安全课程设计》  
报告**

**基于API HOOK的软件行为  
分析系统**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓 名** | **：** |  |  |  |  |
| **学 号** | **：** |  |  |  |  |
| **专业班级** | **：** |  |  |  |  |
| **指导教师** | **：** |  |  |  |  |
| **日期** | **：** |  | **2024年月日** |  |  |

|  |  |  |
| --- | --- | --- |
| 评分标准 | | |
| 1 | 个人与团队  （10分） | 1具有团队意识和团队能力，能够在多学科背景下的团队中承担个体、团队成员以及负责人的角色。  2 参与小组安排的交流、讨论；能够理解团队的意义，了解团队的角色，在团队中主动与其他成员沟通、合作、开展工作；或承担成员角色，完成个体工作；优9-10分；良8分；中7分；及格6分；不及格0-5分。教师根据答辩中分工、报告中团队相关介绍，给出该项成绩。 |
| 2 | 设计解决方案  （40分） | 能够针对信息安全领域特定需求，利用数学模型和工具，规划和设计完整的系统。包括：对行为分析需求任务的分析；初步总体设计、各模块的详细设计；能在设计环节体现创新意识，考虑社会、安全、法律、文化、环境等因素。优（34-40分）；良（30-33）；中（26-29分）；及格（24-25分）；不及格(0-23分)。教师根据学生答辩中思路讲解环节、提交的报告中设计内容，综合给出该项成绩。 |
| 3 | 实现解决方案  （40分） | 根据设计内容，能开发、实现系统的功能。按任务要求完成内容1、框架设计、行为截获 2、行为分析 3、交互界面4、测试样本 5、创新。  优（34-40分）；良（30-33）；中（26-29分）;及格（24-25分）；未完成的项目设计的,不及格(0-23分)。教师根据小组答辩中演示环节、报告中成果展示部分、最终实现源代码等，综合给出该项成绩。 |
| 4 | 报告规范  （10分） | 具有良好的人文社会科学素养、社会责任感，能够在工程实践中理解并遵守工程职业道德和规范，履行责任。报告结构合理，语言通顺、流畅；按华中科技大学本科毕业论文格式规范，字体、段落、图片、表格的版面美观、对他人资料列入参考文献并标注引用。非常规范，优（9-10分）；少量不规范，（1-3项）良（8分）；不规范项目（4-6项），中（7分）；不规范项目（7-9项），及格（6分），不规范项目（10项以上）（0-5分）。教师根据报告，给出该项成绩。 |

评分表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 个人与团队（10%） | 设计解决方案（40%） | 实现解决方案（40%） | 报告规范（10%） | 成绩 | 教师签名 |
|  |  |  |  |  |  |

# 

# 摘 要

本实验基于Detours框架实现API HOOK，以实现对目标程序的文件操作、堆操作、注册表操作以及网络操作进行实时监控和分析，并通过图形界面展示目标程序的API调用情况，从而实现程序危险操作的分析和过滤。项目采用Detours框架对目标程序进行API调用抓取，监控和记录API调用的参数，对API调用信息进行解析和分类。结果显示，本项目可以有效地捕获目标程序的各类操作，包括文件的读写、内存的分配释放、注册表的读写以及网络通信，为进一步的安全分析提供了数据支持。通过对目标程序的API调用进行监控和分析，可以及时发现程序的潜在安全风险，并采取相应的措施进行防范和修复。未来的工作可以进一步完善分析算法，提高对危险操作的准确性和实时性。

**关键词：** API HOOK；进程监控；进程操作分析

**目 录**

[摘 要 I](#_Toc1011785335)

[目 录 II](#_Toc1159857174)

[课程设计任务书 1](#_Toc798259792)

[1． 课程设计目的 1](#_Toc1951257717)

[2． 课程设计要求 1](#_Toc469888211)

[3． 系统环境 1](#_Toc1370296374)

[4． 实验过程记录 1](#_Toc433286898)

[5． 团队分工与个人负责内容说明 2](#_Toc1893151511)

[一 绪言 3](#_Toc1259326502)

[项目背景 3](#_Toc1577029100)

[二 系统方案设计 4](#_Toc1559567331)

[项目目标 4](#_Toc1639099068)

[项目系统架构 4](#_Toc1870370376)

[三 系统实现 6](#_Toc3545198)

[1． DLL注入器实现 6](#_Toc750231480)

[2． API HOOK实现 7](#_Toc684559659)

[3． socket通信实现 8](#_Toc1525847554)

[4． API分析器实现 8](#_Toc1530465848)

[5． 图形界面实现 8](#_Toc670273671)

[四 系统测试 11](#_Toc971259845)

[1． API截获测试 11](#_Toc1772774711)

[2． 文件操作截获分析测试 11](#_Toc721378962)

[3． 堆操作截获分析测试 12](#_Toc229816387)

[4． 注册表操作截获分析测试 13](#_Toc1678667408)

[5． 网络操作截获分析测试 14](#_Toc1940563114)

[6． API拦截测试 15](#_Toc1024499317)

[五 总结与展望 16](#_Toc24811708)

[自评及反馈 17](#_Toc1950667618)

[参考文献 18](#_Toc426060294)

**课程设计任务书**

1. **课程设计目的**

在Windows平台上利用Detours开源项目包提供的接口，通过截获程序的API调用，在无源码的情况下分析样本程序的行为。

1. **课程设计要求**

实现第三方进程WindowsAPI截获框架。

实现程序异常操作行为分析。

实现程序运行交互界面。

编写行为检测样本程序。

项目开源及项目文档编写。

1. **系统环境**

操作系统：Windows10 LTSC 2019。

集成开发环境：Visual Studio 2022（C++桌面开发、.NET桌面开发）。

GUI开发环境：Qt Designer 5.11.1，Python 3.12.2，PyQt6 6.6.1。

1. **实验过程记录**
2. 开源仓库搭建

在gitee.com平台Fork HustDetors项目，根据项目需要重新初始化仓库目录，设置仓库开源许可协议，编写项目提交规范，调整仓库权限。

1. 选择项目框架

根据课程设计要求，确定合适的项目框架结构，划分项目模块，确定开发语言及开发测试环境。

1. 团队分工

按照项目模块的难度、任务量分为多个任务，团队成员根据个人能力选择任务完成，并负责所承担模块的文档编写等。

1. 项目框架搭建

根据项目的主体结构和模块划分，配置项目解决方案，简易实现各模块样例，配置项目构建方案。

1. 项目功能实现

在各模块样例的基础上，编写模块代码，实现各模块完整功能。

1. 项目测试

对项目的各模块进行单独测试，并对整体项目进行功能测试，检测项目是否满足目标，根据测试结构对项目进行调整。

1. 项目发布

构建项目Release版本，完善项目发布文档。

1. **团队分工与个人负责内容说明**

团队成员及负责内容如表1所示.

表 1 团队分工

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **ID** | **职责** | **负责内容** |
| noxke |  | 组长 | 项目框架、GUI界面、项目文档等 |
|  |  | 组员 | 程序分析模块 |
|  |  | 组员 | hook模块 |
|  |  | 组员 | 测试样例 |
|  |  | 组员 | 测试样例 |

在本项目中，本人在项目中承担组长职责，负责项目的整体架构设计、项目开发环境配置，负责项目注入器模块、socket通信模块和图形界面模块，还负责项目文档编写、项目仓库维护等。

**一 绪言**

**项目背景**

随着计算机软件的发展，安全性已经成为软件开发过程中的一个重要关注点。然而，对于已有的程序，尤其是第三方程序，由于其源代码不可用或不受控制，难以确保其安全性。因此，本课程设计旨在利用Detours框架提供的功能，通过API HOOK技术实现对目标程序的实时监控和分析，以便及时发现并防范程序的潜在安全风险。

Detours框架是一个开源项目，提供了强大的API Hooking功能，可以截获程序的API调用，对其进行监控和修改。本项目利用了Detours框架的这一特性，通过对目标程序进行API HOOK，实现了对其文件操作、堆操作、注册表操作以及网络操作的实时监控和分析。同时，通过图形界面展示API调用情况，使用户能够清晰地了解目标程序的行为特征，从而实现了对程序危险操作的分析和过滤。

在项目实施过程中，团队成员根据课程设计任务书确定了项目的具体要求和环境。团队首先搭建了开源仓库，并选择了合适的项目框架结构和开发环境。根据项目的模块划分，团队分工合作，各自完成了相应模块的开发和文档编写工作。在功能实现阶段，团队在各模块样例的基础上编写代码，实现了项目的各项功能。最后，团队对项目进行了全面的测试，并发布了项目的Release版本，完善了项目发布文档，为进一步的安全分析工作奠定了基础。

通过本项目的实施，我们得以深入理解了API HOOK技术的原理和应用，提高了对程序安全性的认识，为今后的安全分析工作提供了重要的经验和支持。

**二 系统方案设计**

**项目目标**

除了满足课程的基本要求外，本项目还额外设计了一些目标，旨在丰富项目功能，增强项目的使用性和实用性，以满足更广泛的应用场景和用户需求。

动态注入方式的支持：考虑到目标分析程序可能是正在运行的进程，也可能是二进制可执行文件，项目需要实现进程的动态注入和二进制文件的启动时注入功能。同时，针对32位和64位程序，项目需要提供x86与x64两种架构的支持。

多进程监控功能：为了增强项目的使用性，项目需要具备监控、分析和控制多个目标进程的能力，实现多进程监控功能，以满足不同场景下的需求。

动态配置能力：由于不同目标程序的分析侧重点可能存在差异，项目需要实现对不同功能API的hook的动态配置能力，在监控分析过程中能够灵活调整hook的API，以满足用户的个性化需求。

API调用拦截功能： 除了监控与分析，项目还需要实现对目标API调用的拦截功能，能够屏蔽API的执行，以避免目标程序执行危险功能，提高系统的安全性。

分析记录的保存与加载功能： 为了方便用户对分析记录的管理和查看，项目需要实现分析记录的保存与加载功能，使用户能够随时查看历史分析结果，并进行进一步的分析和比较。

**项目系统架构**

根据项目需求，项目选择采用主从架构，其中分析平台作为主节点，待分析程序作为从节点。分析平台(EzAPIGuard)包含了dll注入器、socket服务器、API分析器以及GUI界面。待注入到待分析程序中的动态链接库(GuardDll)则包含API钩子和socket客户端功能。

在系统架构中，分析平台使用dll注入器将动态链接库注入待分析程序中。成功注入后，待分析程序与分析平台建立起socket通信，以传输API调用参数信息。API调用参数信息经由socket客户端传递到socket服务器端，在服务器端，API分析器对API调用进行解析。解析结果随后通过GUI界面进行展示，使用户能够清晰地了解目标程序的API调用情况。

项目的分析平台(EzAPIGuard)中，dll注入器和API分析器基于Win32 API实现，使用C++语言编写，并构建为动态链接库(EzGuardLib)。而socket服务器则基于Python的socket模块实现，GUI界面基于PyQt6框架实现，使用Python语言编写。项目最终使用pyinstaller工具进行打包，以便于部署和使用。

注入到待分析程序中的动态链接库(GuardDll)基于Detours和Win32 API实现，同样使用C++语言编写。

项目的系统框架如图2-1所示。

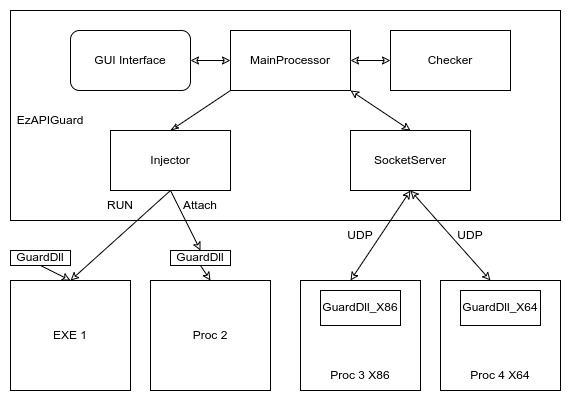


图 2-1 项目系统框架图

**三 系统实现**

1. **DLL注入器实现**

对于运行中的程序，使用创建远程线程的方式调用进程内LoadLibraryA API实现dll注入，对于未运行的PE文件，为确保程序执行第一条指令前完成dll注入，注入器使用调试启动进程，对程序第一条指令下软件断点，捕获断点后创建远程线程实现dll注入，完成注入后恢复软件断点，继续运行主线程。

在标准的远程线程注入中，首先使用WriteProcessMemory将待注入的dll路径写入目标进程内存，再使用CreateRemoteThread创建远程线程执行LoadLibraryA，参数为写入目标内存的dll路径，线程运行后将加载dll到目标进程中。在Windows系统中，不同进程内kernel32.dll加载地址相同，因此LoadLibraryA的地址也相同，创建远程线程时直接使用当前进程中的LoadLibraryA地址即可，不需要额外获取目标进程内的LoadLibraryA地址。但本项目中的注入器需要同时兼容32位程序与64位程序，项目本身为64位程序，因此注入32位进程时，无法使用上述方式获取LoadLibraryA地址，故项目选择使用ToolHelp32对目标进程创建快照，遍历进程内模块找到进程中的kernel32.dll的加载地址，再加上偏移地址获取目标进程内的LoadLibraryA地址。

注入器模块提供4个API接口，LoadLibraryArva86与LoadLibraryArva64为两种架构下LoadLibraryA在kernel32.dll中的偏移，InjectByPID实现注入运行中的进程，RunInject实现运行PE文件并注入。

1. // 两种架构下LoadLibraryA在kernel32.dll中的偏移
2. DLL\_EXPORT **extern** uint64\_t LoadLibraryArva86;
3. DLL\_EXPORT **extern** uint64\_t LoadLibraryArva64;
5. // 注入dll到目标进程
6. DLL\_EXPORT **BOOL** InjectByPID(uint32\_t dwPID, **const** **char** \*dllPath);
8. // 启动目标程序并注入dll
9. DLL\_EXPORT uint32\_t RunInject(**const** **char** \*exePath, **char** \*cmdLine, **const** **char** \*dllPath);
10. **API HOOK实现**

目标进程的API拦截基于Detours的DetoursAttach功能实现，对于每一个需要拦截的API，使用OldApiName保存原始API地址，使用NewApiName对API进行替换，在NewApiName中实现API拦截后的参数处理以及与分析平台通信，并根据HOOK配置与分析平台反馈决定是否执行原API。

为简化拦截API的编写，项目使用宏定义的方式对API进行参数处理，将大量重复出现的代码替换为宏，避免在修改功能时需要对所有API进行修改。

API拦截后，进行API参数处理与分析平台通信，最后根据allow的情况决定是否绕过原API的执行，当允许原API执行时，使用保存的OldApiName和调用参数完成原API执行，当禁止原API执行时，根据API返回对应的错误值。

以MessageBoxA API为例，用于拦截的NewMessageBoxA实现如下：

1. **int** (WINAPI\* OldMessageBoxA)(\_In\_opt\_ **HWND** hWnd, \_In\_opt\_ **LPCSTR** lpText, \_In\_opt\_ **LPCSTR** lpCaption, \_In\_ **UINT** uType) = MessageBoxA;
3. DLL\_EXPORT **int** WINAPI NewMessageBoxA(\_In\_opt\_ **HWND** hWnd, \_In\_opt\_ **LPCSTR** lpText, \_In\_opt\_ **LPCSTR** lpCaption, \_In\_ **UINT** uType)
4. {
5. // uType参数省略
6. API\_HOOK\_BEGIN\_MACRO(API\_MessageBoxA, 3);
8. //将hWnd转换为窗口名
9. **char** buffer[MAX\_ARG\_LEN];
10. GetWindowTextA(hWnd, (**LPSTR**)buffer, MAX\_ARG\_LEN);
11. API\_ARG\_MACRO(**HWND**, hWnd, buffer, strlen(buffer));
12. API\_ARG\_STR\_MACRO(**LPCSTR**, lpText);
13. API\_ARG\_STR\_MACRO(**LPCSTR**, lpCaption);
15. API\_HOOK\_END\_MACRO(API\_MessageBoxA);
17. **if** (allow)
18. {
19. **return** OldMessageBoxA(hWnd, lpText, lpCaption, uType);
20. }
21. **else**
22. {
23. **return** IDCANCEL; // 返回窗口取消值
24. }
25. }
26. **socket通信实现**

待分析程序与分析平台的通信使用socket实现，以方便实现多进程监控的功能。项目使用udp协议进行数据传输，被分析程序中创建线程执行ClientSocketThread与分析平台进行持续通信，接收和调整API拦截配置，发送进程存活状态与API拦截情况。API拦截后由负责拦截的API函数创建额外的udp端口向分析平台发送API参数信息，并根据拦截配置等待分析平台的回复。

分析平台使用UdpSocketServer线程与所有的待分析程序进行通信，接收API调用信息，使用回调函数处理API调用信息。

1. **API分析器实现**

API分析模块接收API调用信息，根据API的类型进行分类，对于文件操作、堆操作注册表操作以及网络操作进行分别处理，根据API调用的参数信息判断API调用的危险程度，分析API的危险操作，将分析信息返回到分析平台进行显示。

API分析模块提供checker接口供分析平台调用。

1. DLL\_EXPORT **void** checker(**char**\* inBuffer, uint16\_t inbufferlen, **char**\* outbuffer, uint16\_t outbufferlen)；
2. **图形界面实现**

在图形界面的开发中使用Qt Designer进行图形界面的绘制和排版，并使用Python实现图形界面对应的逻辑功能。

项目包含MainWindow、ProcessView、APIRequestDialog和AboutDialog四个图形窗口。MainWindows是项目的主窗口，包括信息的展示，项目配置等功能。ProcessView是进程视图，提供系统中运行的进程名、进程ID的显示，用于选择运行中的进程进行注入分析操作。APIRequestDialog是API请求窗口，当API的拦截模式设置为请求时，显示API请求窗口展示API参数信息及分析器分析结构，并由用户选择允许或拒绝。AboutDialog是程序信息窗口，用于显示项目的版本和描述信息等。

主窗口MainWindow被划分为5个区域（如图3-1所示），区域1为程序菜单栏，包含File（记录保存与加载）、Process（分析程序附加或启动）、View（重置视图）、About（程序信息）4项菜单设置。区域2为进程列表，显示监控的进程名称、进程PID和监控状态。区域3为API调用记录，显示被监控程序的API调用参数信息和记录搜索等。区域4为配置区域，包含显示视图的切换（选择的程序或全局），API拦截规则设置，API计数器，HOOK配置和警告计数等。区域5为分析器记录，用于显示API调用的分析结果，分析告警等信息。

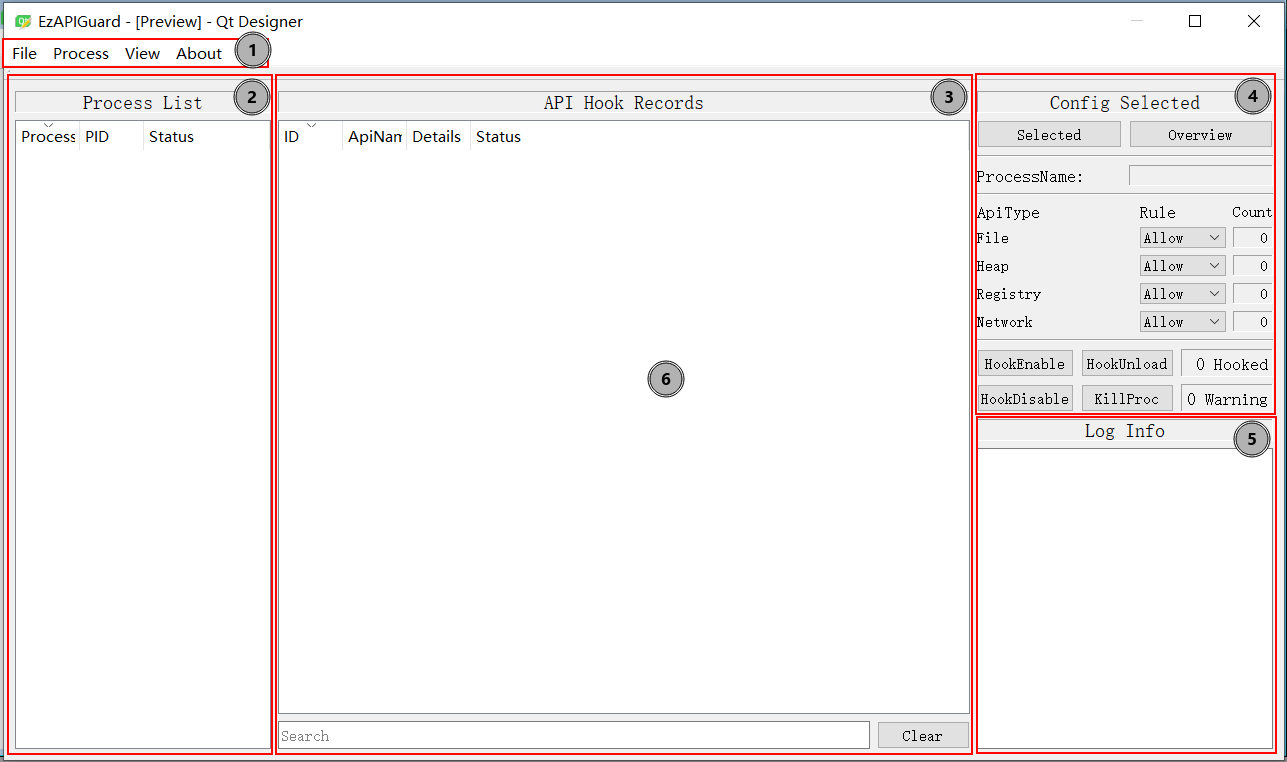


图 3-1 主窗口视图

进程视图ProcssView显示系统中的进程信息，包含进程搜索和选择功能，选择进程后使用Attach按钮完成进程的附加监控，如图3-2所示。

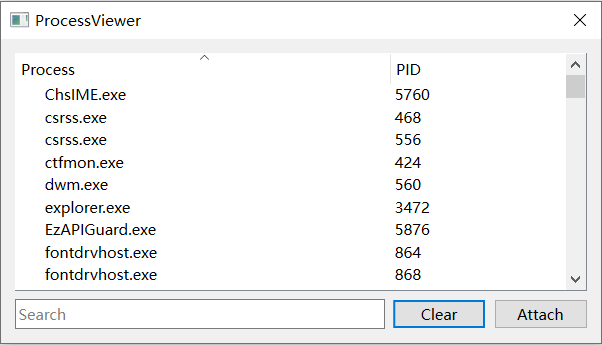


图 3-2 进程视图

图像界面的逻辑部分使用异步处理机制，用户操作界面后，图形界面发出信号，触发控件绑定的槽函数实现用户操作处理，图形界面的刷新显示由刷新线程实现，后端部分需要更新视图时，将视图进行缓存，等新的刷新周期开始时进行视图更新，以确保数据显示的同步性，降低视图更新的资源消耗。

项目的记录保存加载由前端Record模块实现，触发记录保存操作时，使用json模块对分析数据进行序列化，将分析继续以json格式保存至文件，触发记录加载操作时，读取保存的json文件，使用json模块进行反序列化，解析分析结果，将分析结果添加到视图中。

程序的配置功能包含两种控制模式，全局视图模式和选中进程视图模式，配置模式为全局模式时，API记录和分析记录显示所有监控进程的记录，控制菜单中的配置修改对所有监控进程有效。配置模式为选中进程模式时，API记录和分析记录仅显示当前选中的进程记录，控制菜单中二配置修改仅对当前选中进程有效。对文件操作API、堆操作API、注册表操作API和网络操作API的单独控制，当API配置调整后，通过UdpServer将配置发送至被监控程序，被监控程序调整HOOK配置。配置菜单还包括HOOK的启动与关闭，注入dll卸载与关闭进程等功能。

**四 系统测试**

1. **API截获测试**

使用测试程序对分析平台进行测试，检测分析平台的API截获和显示功能，检查API截获的参数。

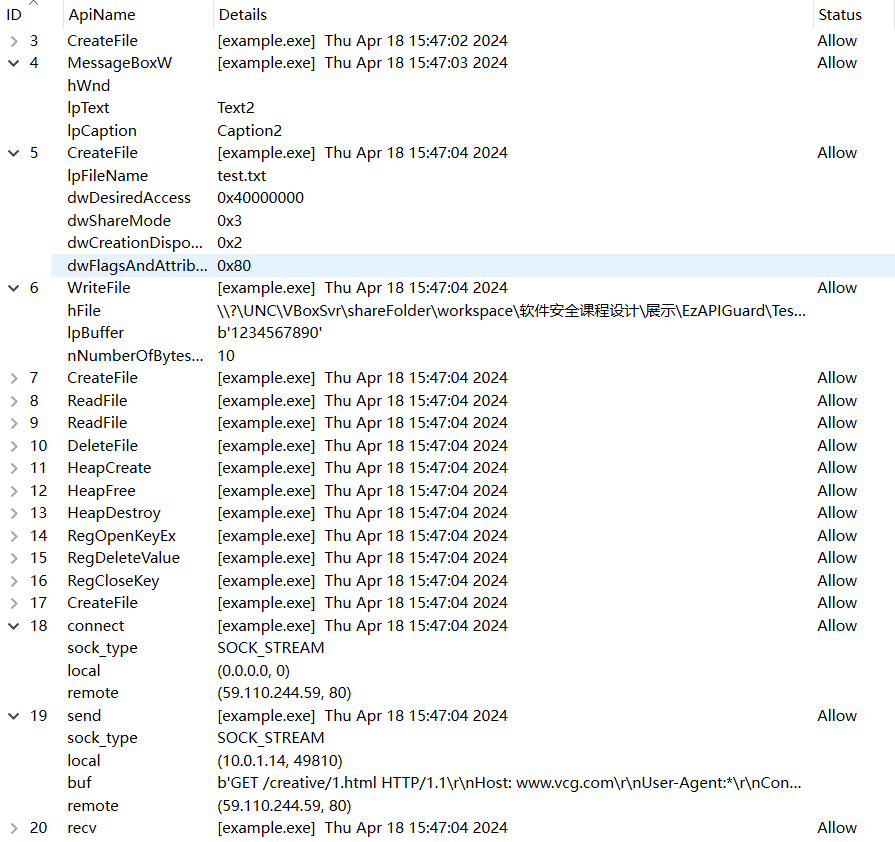


图 4-1 API截获测试

如图4-1所示，对于窗口API、文件操作API、堆操作API、注册表操作API和网络操作API，平台均能正常截获并显示API的名称、调用信息和参数信息，对于API的参数，格式处理正确，能够正确显示参数中的路径、字符串、缓冲区等。

1. **文件操作截获分析测试**

在文件操作相关API的截获上，项目实现了对常用文件API，如CreateFile、WriteFile、ReadFile和DeleteFile的截获，截获时能够自动提取操作文件的文件名，操作缓冲区和操作大小等（如图4-2所示）。

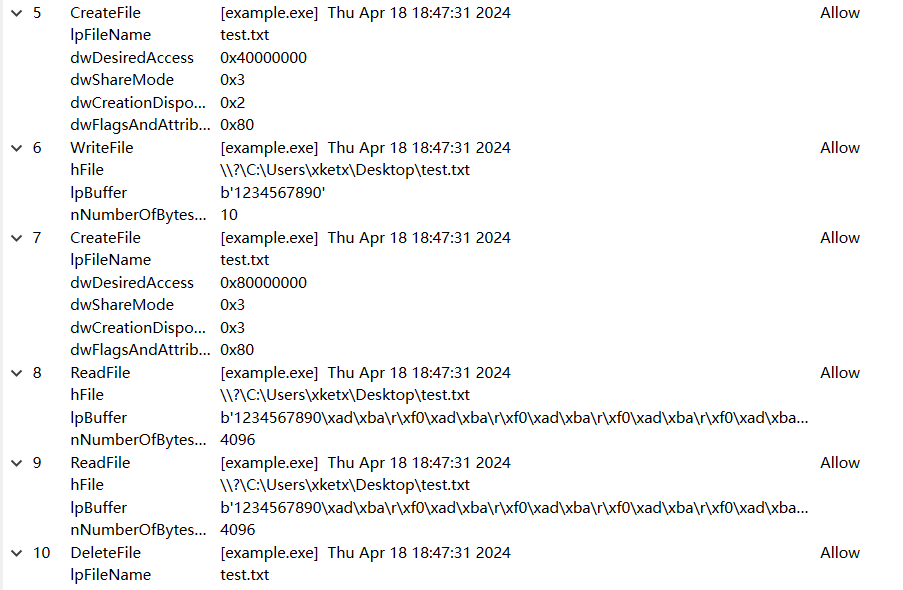


图 4-2 文件操作API截获测试

在文件操作的分析上，项目实现了对文件夹操作检测、程序自我复制检测和修改可执行文件检测。

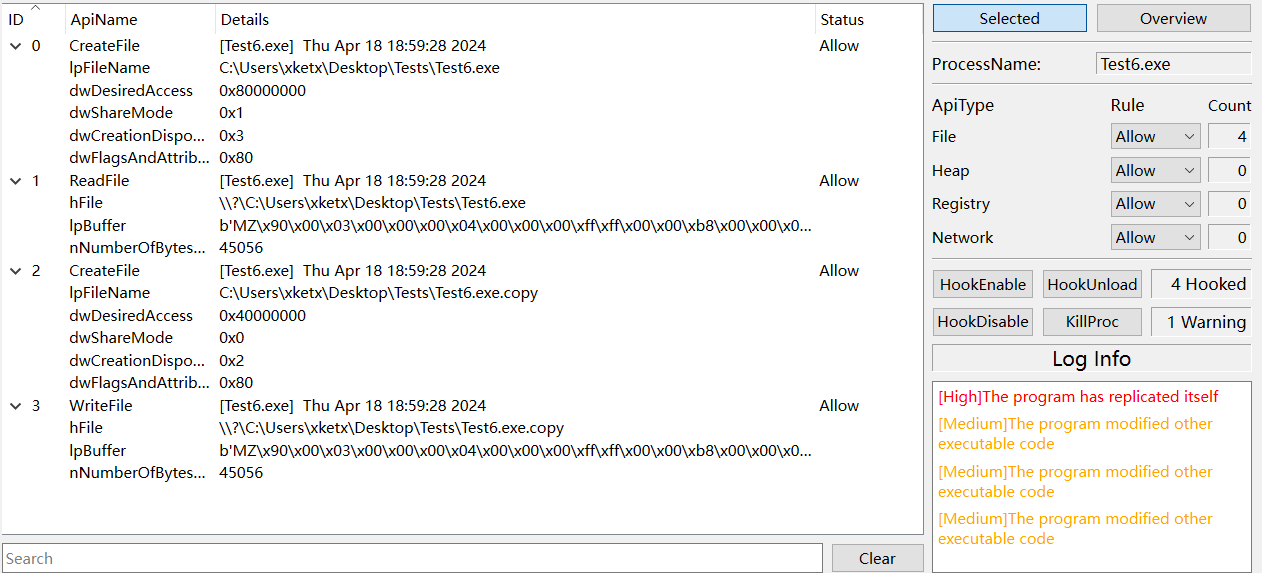


图 4-3 文件操作分析

如图4-3所示，目标进程尝试复制自身并修改可执行文件，分析平台检测到API的调用并显示告警信息提醒程序复制和程序修改，文件操作的截获和分析功能正常。

1. **堆操作截获分析测试**

在堆操作相关API的截获上，项目实现了对HeapCreate、HeapFree和HeapDestory的截获，截获时能够获取堆的句柄、操作的堆的大小以及操作的内存地址等参数。

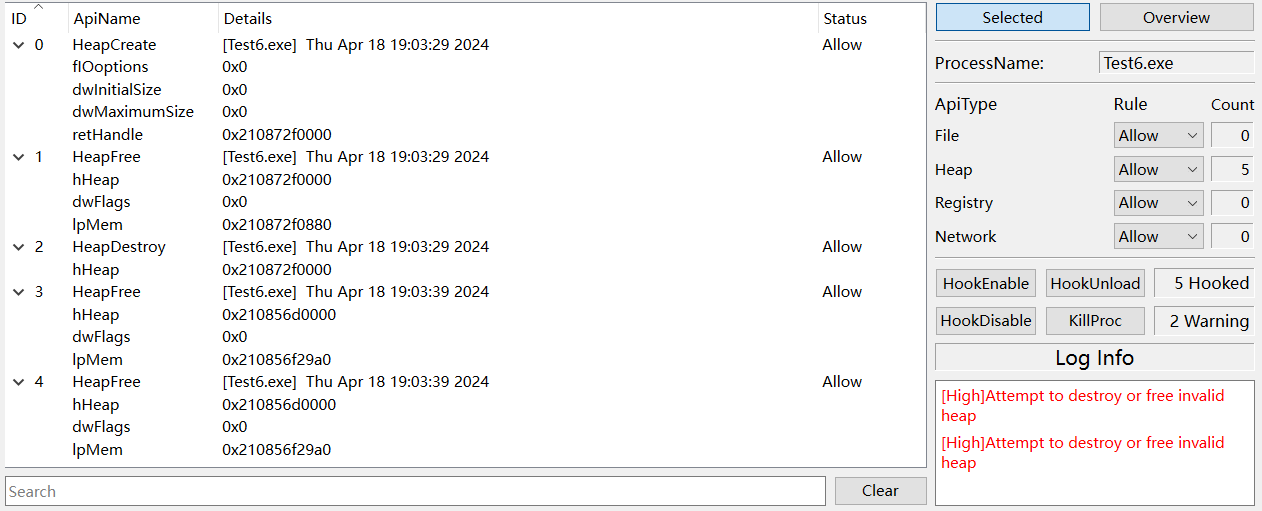


图 4-4 堆操作API截获分析

如图4-4所示，目标程序尝试创建堆、分配堆内存、释放堆内存以及销毁堆，后需又对同一地址进行两次释放操作，尝试触发Double Free漏洞，分析平台成功截获堆操作的参数情况，并成功检测出程序的堆释放漏洞，显示堆操作告警信息。

1. **注册表操作截获分析测试**

在注册表操作的截获上，项目对常用的注册表操作API，如RegCreateKeyEx、RegOpenKeyEx、RegCreateKeyEx、RegSetKeyEx等均实现了截获，能够检测注册表的打开、读取注册表值、修改注册表值、创建注册表、删除注册表等多种操作，并且能够正确检测操作的注册表路径、注册表值、值类型等参数信息。

如图4-5所示，目标程序尝试创建注册表值，修改系统启动项，并尝试将自身设置为系统启动项，分析平台成功截获目标程序的注册表操作信息，正确显示注册表路径等参数、并将检测到目标程序修改系统启动项的操作，对程序存在的危险操作做出了正确的告警信息。

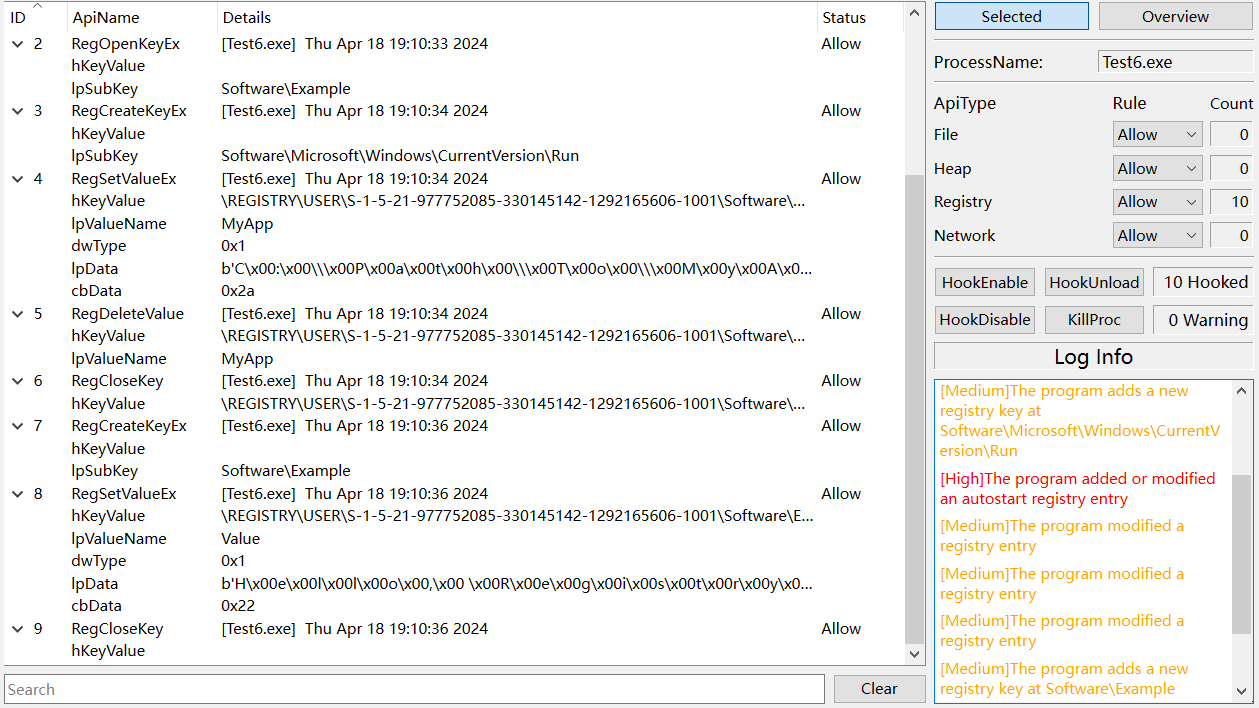


图 4-5 注册表API截获分析

1. **网络操作截获分析测试**

在网络操作的分析上，项目实现了对常用网络API的截获，包括connnect、recv、send、recvfrom、sendto等，实现了网络操作的ip地址和端口的获取，能够自动获取网络发送和接受的内容，在网络操作分析上，项目实现了http明文协议检测、实现了UDP协议与TCP协议的检测。

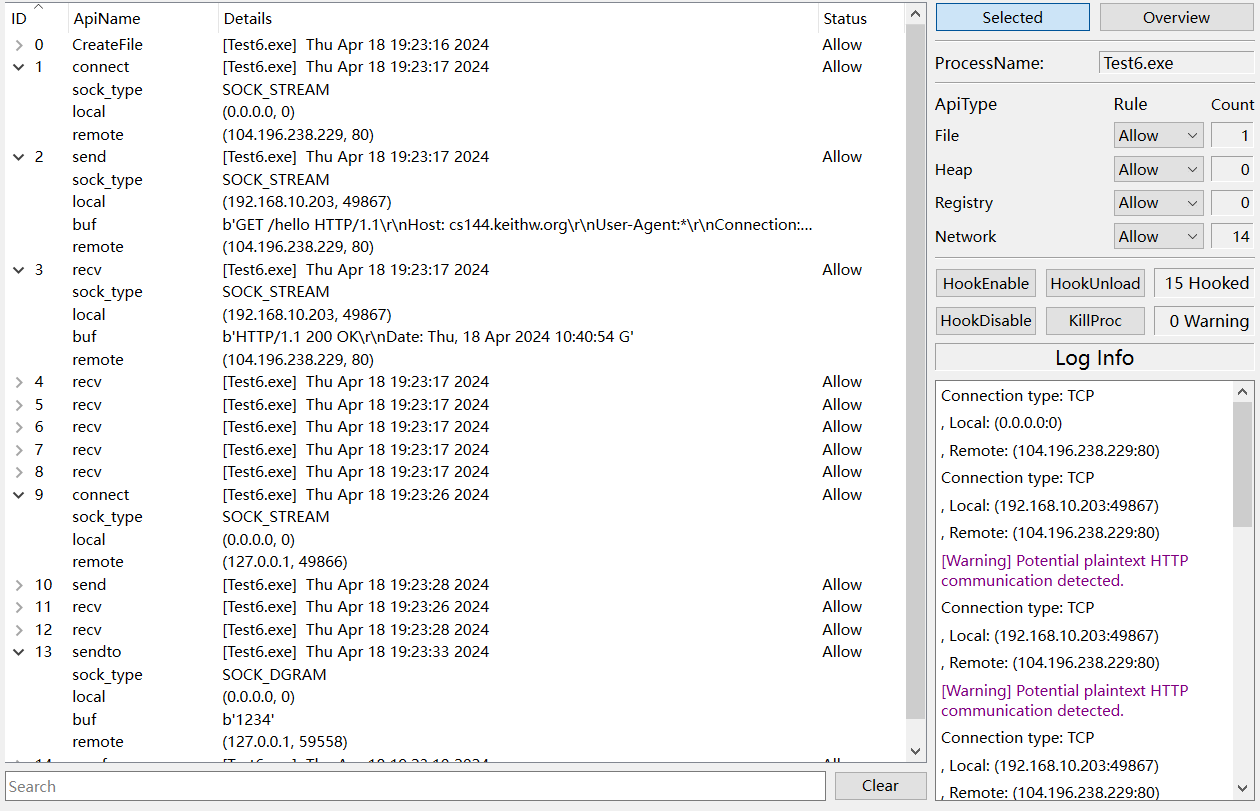


图 4-6 网络操作API截获分析

如图4-6所示，目标程序进行TCP连接，并使用http协议明文传输内容，后续使用udp协议进行内容收发，分析平台正确检测到网络连接的双方ip地址与端口，能够正确检测socket类型和正常显示网络协议中传输的数据，对于网络操作，能够正确识别UDP协议与TCP协议，能够对不安全的http明文传输做出告警信息。

1. **API拦截测试**

项目除了常规API的截获操作外，还实现了API的拦截操作，能够拒绝目标程序的API的执行，并根据API类型返回执行错误结果，避免目标程序对计算机造成损害，如图4-7所示，目标程序尝试复制自身，分析平台配置文件操作策略为Reject，默认拒绝程序的文件操作，因此程序的文件操作失败。

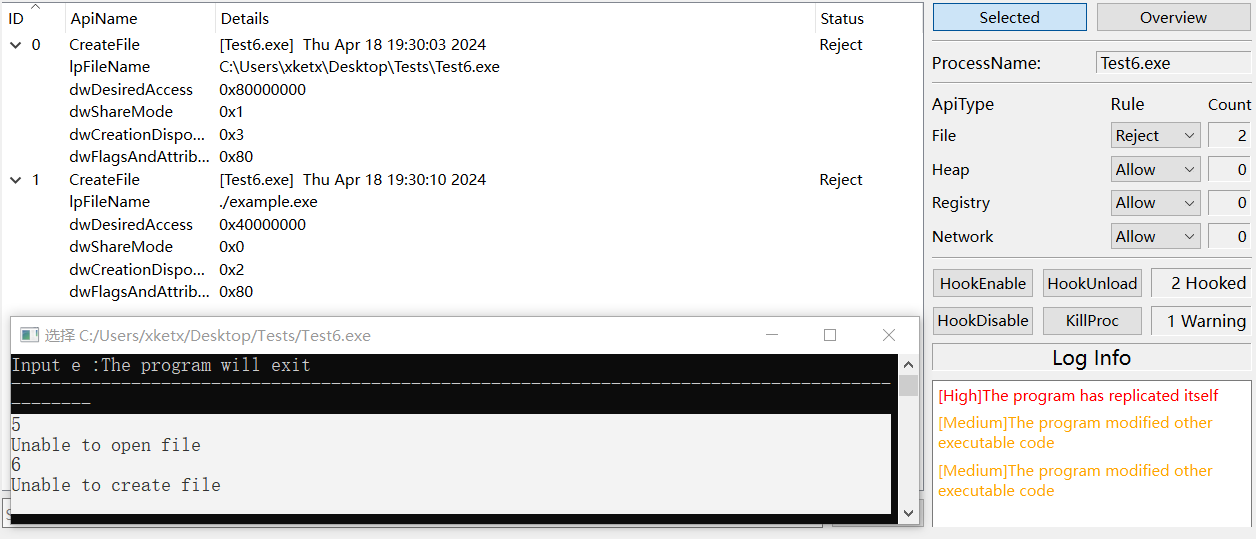


图 4-7 API拦截测试

**五 总结与展望**

在本次课程实践中，以团队合作形式实现了一个简易的程序行为分析工具，能够对程序的文件操作、堆操作、注册表操作以及网络操作进行拦截和分析，过滤程序的危险操作。在项目的开发中，学习使用了dll注入、API HOOK等技术，并利用动态链接库实现了程序的跨语言调用，利用PyQt实现了简单的图形化界面对项目进行展示与交互。

在团队合作中，本人承担了组长职责，负责项目整体规划和项目仓库维护，学习了团队分工与任务分配，学习了开源项目的开发和维护，了解了项目的开发、测试流程。

项目虽然实现了API的调用分析功能，但在实际环境中，分析结果存在较多的误报情况，分析的准确度还有待提高，后续开发可能需要大量的测试样例进行分析，提取真正存在风险的操作，修改分析过滤模型以提高项目准确性。在API的拦截功能中，实现了拒绝执行和手动选择的功能，后续可以考虑根据分析结果实现危险操作自动拦截绕过，以增强项目的实用性。此外，在项目的通信选择udp协议，但是在API消息处理时未使用异步处理，导致在API调用次数过多或同时分析的目标过多时消息处理线程的处理速度跟不上API消息的到来速度，存在消息丢失和进程阻塞的可能，后续开发需要考虑修改消息处理的方法，使用异步调用替换回调函数。

在项目的扩展功能上，可考虑增加插件功能和分析器样本动态更新功能，以增强项目的实用性，此外，由于项目基于udp协议通信，后续可考虑增加远程控制，多机器管理的功能，实现一台分析机控制多台目标主机的功能。

**自评及反馈**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **指标点** | **评价标准** | | | | **自评** |
| **优** | **良** | **中** | **差** |
| **3.设计开发解决方案** | 设计的方案能够完全覆盖所需要解决的问题，使用掌握的技术，并能够在实验报告中完整准确的说明设计思路。 | 设计的解决方案能大部分覆盖所需要解决的问题， 并能够在实验报告中较完整准确的说明设计思。 | 实现了任务书中提出的主要功能要求，具备在调试和实现过程中分析问题和解决问题的基本能力，熟悉开发过程中的细节，但对关键部分解释含糊 | 设计的解决方案不能覆盖解决所提出的提问题，使用当前掌握的技术不具备实现可行性， 在实验报告中未能清楚说明设计思路。 | 优 |
| **9.个人和团队** | 在分小组进行课程设计的过程中，能非常积极主动地承担具有难度的任务，按时完成个人负责的模块，并能主动协助组内成员；善于跨组纵向学习和组内合作。 | 在分小组进行课程设计的过程中，能保证质量的完成个人负责的模块，能配合组内其他成员的工作。能较好完成跨组纵向学习和组内合作。 | 在分小组进行课程设计的过程中，基本能完成个人负责的模块，通过组内学习，完成程序的装配。能基本完成跨组纵向学习和组内合作。 | 在分小组进行课程设计的过程中，不能按时完成个人负责的模块，不能与其他组员合作开展工作。 | 优 |
| **10.沟通** | 能针对本课程设计的问题，灵活运用专业知识，通过撰写报告或者陈述发言，清晰准确的说明系统的设计方案、实现方法和测试结果；文字或语言表达通顺流畅。 | 能针对本课程设计的问题，运用专业知识，通过撰写报告或者陈述发言，较为准确的说明系统的设计方案、实现方法和测试结果；文字或语言表达较为通顺流畅。 | 能针对本课程设计的问题，运用专业知识，通过撰写报告或者陈述发言，基本能说明系统的设计方案、实现方法和测试结果；文字或语言表达比较通顺。 | 能针对本课程设计的问题，运用专业知识，通过撰写报告或者陈述发言，系统的设计方案、实现方法和测试结果说明不充分；文字或语言表达不通顺。 | 良 |

意见或建议反馈：

本次课程实验采用团队开发实现，但在实际的任务中，可能存在部分成员积极性较低，在项目中参与较少，建议课程增加奖励机制，对项目完成度高、功能完整的团队进行除课程加分外的奖励，以增强成员的积极性。

**参考文献**

[1] 软件安全 课程设计实验指导书

[2] Microsoft Research Detours Wiki

[3] Microsoft Win32 API的编程参考

[4] Qt for Python Documentation

[5] git Book