《漏洞利用及渗透测试基础》实验报告

姓名： 学号： 班级：

**实验名称：**

程序插桩及Hook实验

**实验要求：**

复现实验一，基于Windows MyPinTool 或在Kali中复现malloctrace这个PinTool,理解Pin插桩工具的核心步骤和相关API，关于malloc和free函数的输入输出信息。

**实验过程：**

1. 复现实验一，基于Windows MyPinTool 或在Kali中复现malloctrace这个PinTool

消息Hook注入代码实现类似插桩作用

KeyHook.cpp

KeyBoardProc()

return 1;//实现拦截记事本

Hook函数

//导出函数

HookStart()

SetWindowsHookEx(WH\_KEYBOARD,KeyboardProc,g\_hInstance,0);

HookStop()

UnhookWindowsHookEx(g\_hHook);

//自定义得到动态链接库特定函数的地址

HookStart= （PFN\_HOOKSTART)GetProcAddress(hDll,DEF\_HOOKSTART);

HookStop = (PFN\_HOOKSTOP)GetProcAddress(hDll,DEF\_HOOKSTOP);

HookStart();

...

FreeLibrary(hDll);

//打开记事本输入被拦截，输入q推出后，可正常输入

APIHook

strcpyW((LPWSTR)dest,(LPWSTR)src);

hook\_iat();

Pin利用PinTool完成插桩功能的包装和实现

iscount0.cpp

static Uinit64 icount = 0;

void docount(){}

voi instruction(){}

int main(){

//初始化Pin

//打开输出文件

//调用指令级插桩，注册回调函数，回调函数调用自定义都count()

}

//编译运行，动态链接库

make inscount0.test target -intel64

FirstCpp.c

#include<stdio.h>

void main(){

printf(“hello world!”);

}

//编译

gcc -o First FirstCPP.c

//插桩(Linux)

./pin-t./source/tools/ManualExamples/obj-intel64/inscount0.so- ../testCPP/First

//输出

hello world!

//inscount.out(统计指令数量)

Count 192994

//自定义

void Instruction(INS ins,VOID \* v){

if(INS\_OPcode(ins)==XED\_ICLASS\_MOV&&INS\_ISMemoryRead(ins)&&INS\_OperandIsReg(ins,0)&&INS\_OperandIsMemory(ins,1))

{

icount++;

}

}

//重新编译再插桩

//打开inscount.out

Count 1806

2.理解Pin插桩工具的核心步骤和相关API

3.关于malloc和free函数的输入输出信息

（此处根据实际操作过程，留下具体操作步骤、附加一些自己的理解，即可）

**心得体会：**

消息Hook的是Windows的消息，APIhook为一些函数

此外，通过本实验，