

# 《软件安全》实验报告

姓名：刘星宇 学号：2212824 班级：信息安全法学双学位班

## 实验名称：

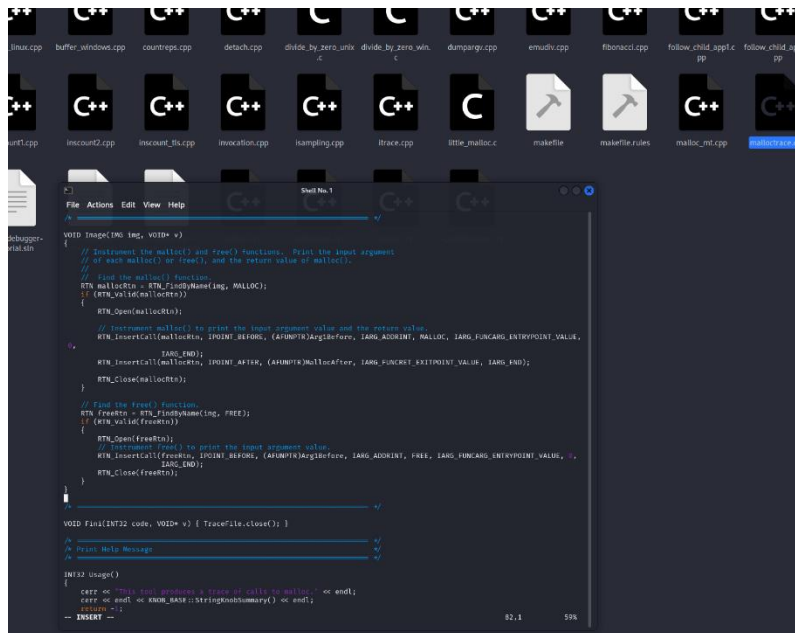
程序插桩及 Hook 实验

## 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，关注 malloc 和 free 函数的输入输出信息。

## 实验过程：

1. 观察 malloctrace.cpp 代码，理解其核心步骤，malloctrace 的核心代码如下：  
在这段代码中，对 malloc() 和 free() 函数进行插桩，以便在这两个函数被调用时打印输入参数和返回值。



```
VOID Trace(INT32 argc, VOID* argv)
{
    // Instrument the malloc() and free() functions. Print the input argument
    // of each malloc() or free(), and the return value of malloc().
    //
    // Note: the malloc() function
    RTN_Open(malloc);
    RTN_InsertCall(malloc, IPPOINT_BEFORE, (AFUNC_PTR)ArgBefore, IARG_ARG0, IARG_FUNCARG_ENTRYPOINT_VALUE,
    IARG_END);
    RTN_Close(malloc);

    // Instrument the free() function
    RTN_Open(free);
    RTN_InsertCall(free, IPPOINT_BEFORE, (AFUNC_PTR)ArgBefore, IARG_ARG0, IARG_FUNCARG_ENTRYPOINT_VALUE,
    IARG_END);
    RTN_Close(free);
}

VOID Fini(INT32 code, VOID* v) { TraceFile.Close(); }

// Print Help Message
//
// Usage:
//
// gcc -o malloctrace test.c -DTRACE -DTRACE_FILE=malloctrace.log -DTRACE_FILE_SIZE=1024 -DTRACE_FILE_MODE="a"
//
// -- INSERT --
```

2. 对 malloctrace 进行编译

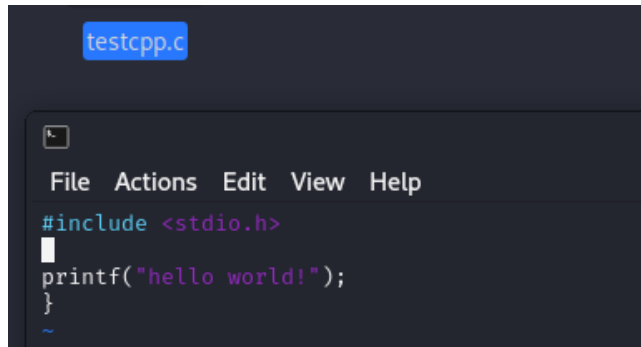
用指令：make malloctrace.test TARGET=intel64，对 malloctrace 进行编译，在 obj-intel64 文件夹中出现了编译后的 malloctrace.o 和 malloctrace.so 文件。



配了大小 0x5 的内存块，并返回了地址 0x562d8169b2a0  
free(0) 表示尝试释放空指针。

#### 4. 编写 hello.c 程序并编译

编写的代码如下，作用是打印一个 hello world!



```
testcpp.c

File Actions Edit View Help

#include <stdio.h>
int main()
{
    printf("hello world!");
}
~
```

使用指令：gcc -o hello hello.c，编译出可执行文件。

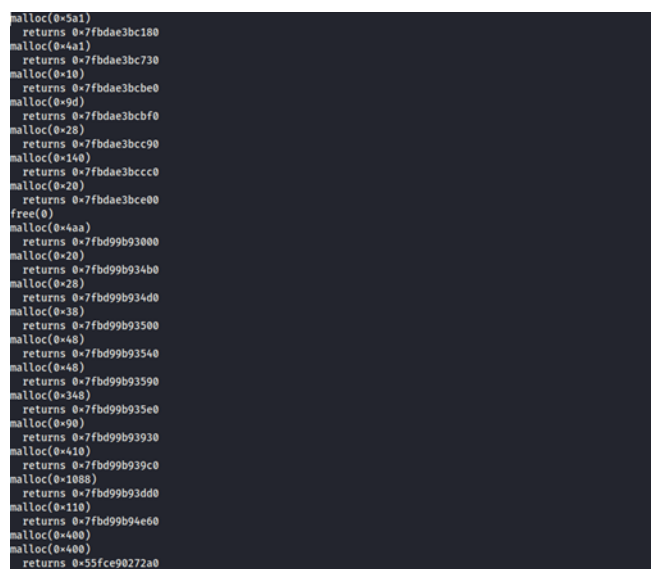
#### 5. 对 hello.exe 进行插桩

使用指令：

```
./pin -t ./source/tools/ManualExamples/obj-intel64/malloctrace.so -
```

- ./hello

进行插桩，在 malloctrace.out 中查看插桩后程序的输出结果：



```
malloc(0x5a1)
returns 0x7fbd9b93000
malloc(0x4a1)
returns 0x7fbd9b93000
malloc(0x10)
returns 0x7fbd9b93000
malloc(0x9d)
returns 0x7fbd9b93000
malloc(0x28)
returns 0x7fbd9b93000
malloc(0x140)
returns 0x7fbd9b93000
malloc(0x20)
returns 0x7fbd9b93000
free(0)
malloc(0x4aa)
returns 0x7fbd9b93000
malloc(0x20)
returns 0x7fbd9b93000
malloc(0x28)
returns 0x7fbd9b93000
malloc(0x38)
returns 0x7fbd9b93000
malloc(0x48)
returns 0x7fbd9b93000
malloc(0x48)
returns 0x7fbd9b93000
malloc(0x348)
returns 0x7fbd9b93000
malloc(0x90)
returns 0x7fbd9b93000
malloc(0x410)
returns 0x7fbd9b93000
malloc(0x1088)
returns 0x7fbd9b93000
malloc(0x110)
returns 0x7fbd9b93000
malloc(0x400)
returns 0x7fbd9b93000
malloc(0x400)
returns 0x7fbd9b93000
```

可以看到，hello 程序中进行了多次 malloc 和 free 函数的调用，但因为程序较短，调用这两个函数的次数少于 bash.exe 程序。

#### 心得体会：

在本次实验中，我选择了在 Windows 环境下使用 MyPinTool 或在 Kali Linux 中复现 malloctrace 这个 PinTool，以深入理解 Pin 插桩工具的核心步骤和相关 API，并特别关注了 malloc 和 free 函数的输入输出信息。