

网络安全实验报告

题 目 基于 libnet 的程序设计

专 业 信息安全 .

指 导 教 师 王彦 .

# 一、实验目的

掌握 libnet 数据包的构造原理。

**二、实验内容**

1. 掌握 libnet 数据包的构造原理
2. 编程实现基于 libnet 的数据包构造，结合前面实验给出验证过程。能够对源码进行解释。

# 三、实验过程

**基于 libnet 的数据包构造实验基本信息：**

实验环境：Ubuntu16.04 x64 编程语言：C 语言

## 1. 需求分析

需要使用 libnet 构造并发送一个数据包，并验证这个数据包被成功发送了。验证这一过程需要用到实验二中的捕包程序 pcap，将生成的数据包从虚拟机 B 发送到虚拟机 A，虚拟机 A 中的捕包程序会自动将其捕获，通过检查各项信息，证明捕获的数据包就是从虚拟机 B 此程序 createPac 中发送的数据包。

追加：在虚拟机 A 中编写接收来自相应端口 udp 数据包的程序 recvUDP，验证该数据包确实可以被正确接收。

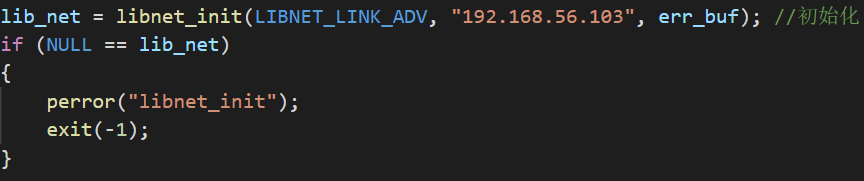
## 2. 程序结构

利用libnet函数库开发应用程序的基本步骤：

1）数据包内存初始化

调用 libnet\_init()函数，选择构造数据包的类型和发送该数据包的网络接口， 获得libnet句柄。

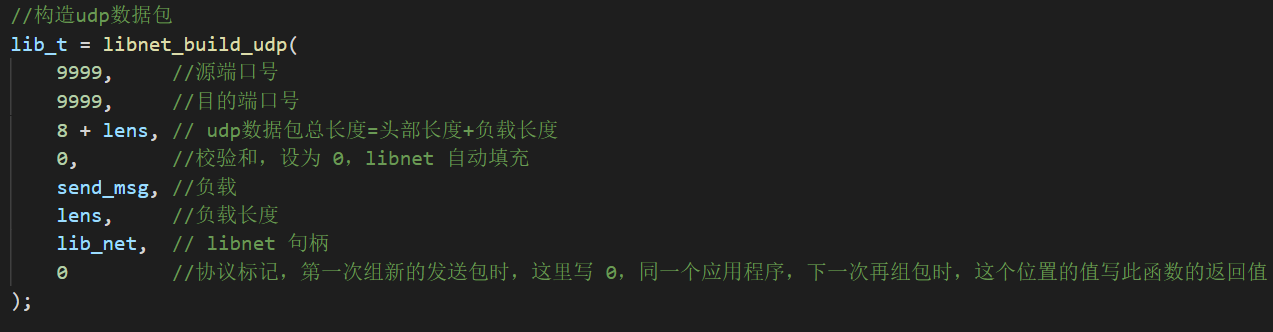
这里选择IP地址为虚拟机192.168.56.103的网络接口发送，如下图。



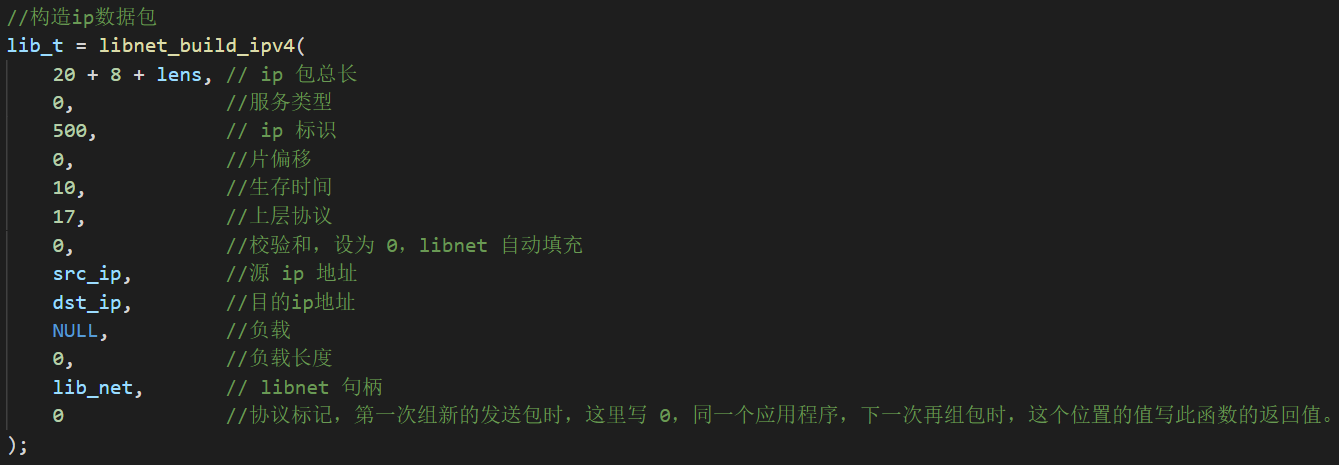
2）构造数据包

分别调用函数构造各层数据包，从UDP数据报，到IP分组，再到以太网帧进行 构建：分别先构建上一层的PDU，作为SDU传递给下一层，逐层封装；顺序不能改 变。

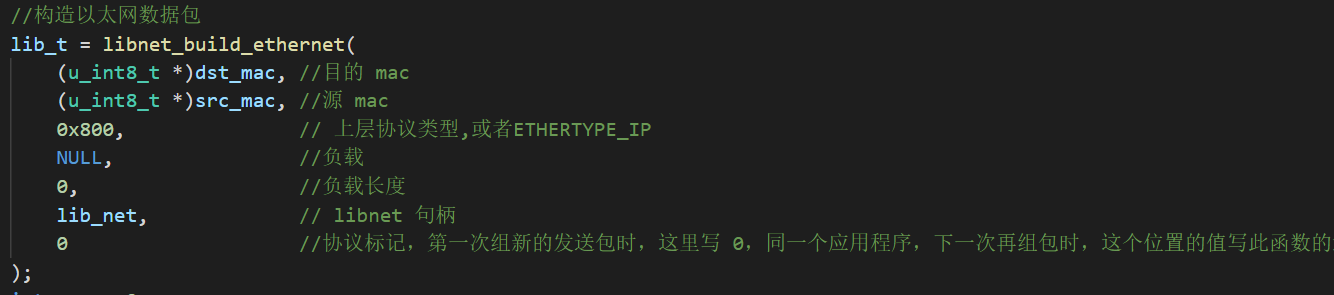
UDP数据报



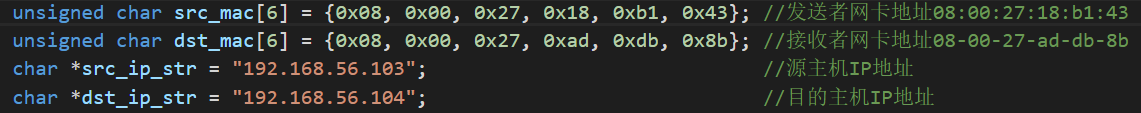
IP分组

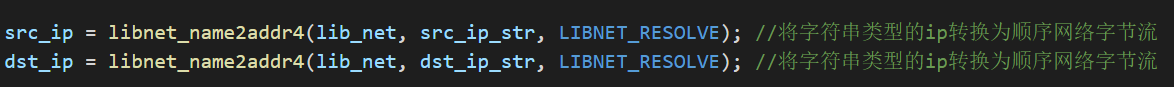


以太网帧



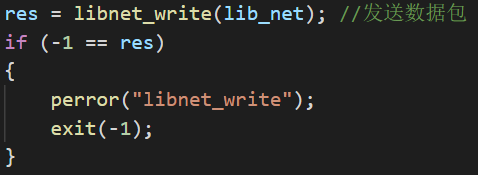
其中源和目的端口选择9999，MAC地址和IP地址为实验一的两个虚拟机IP地 址。





3）发送数据

调用函数libnet\_write()发送数据包。



4）释放资源

调用函数libnet\_destroy()释放资源。

IMG_256

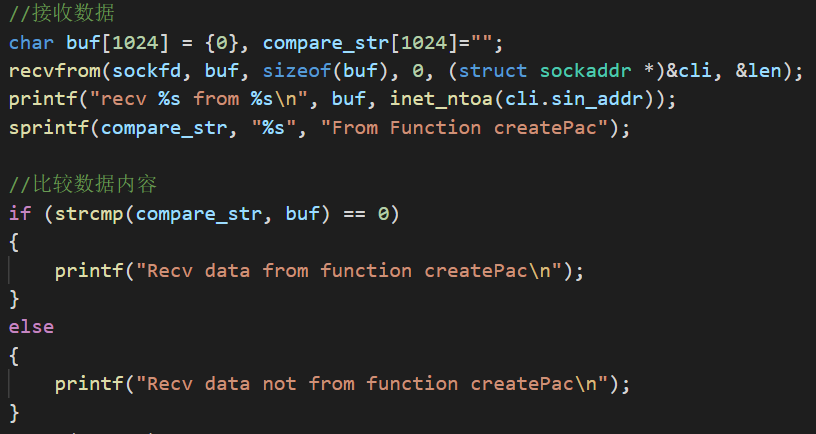
## 3. 进一步验证

在这里我们将udp数据包的数据内容设置为“From Function createPac”，方便验证数据包是否发送成功，是否来自函数createPac。

IMG_256

1)使用实验二中的捕包程序pcap.c，使用过滤规则“udp”（或其他过滤规则如”host 192.168.56.103”等，由于本人网络端口非混合模式接收udp数据包数量较少，这里过滤udp即可）捕捉由另一个虚拟机构造并发送的数据包，查看其MAC地址、IP地址是否对应正确。

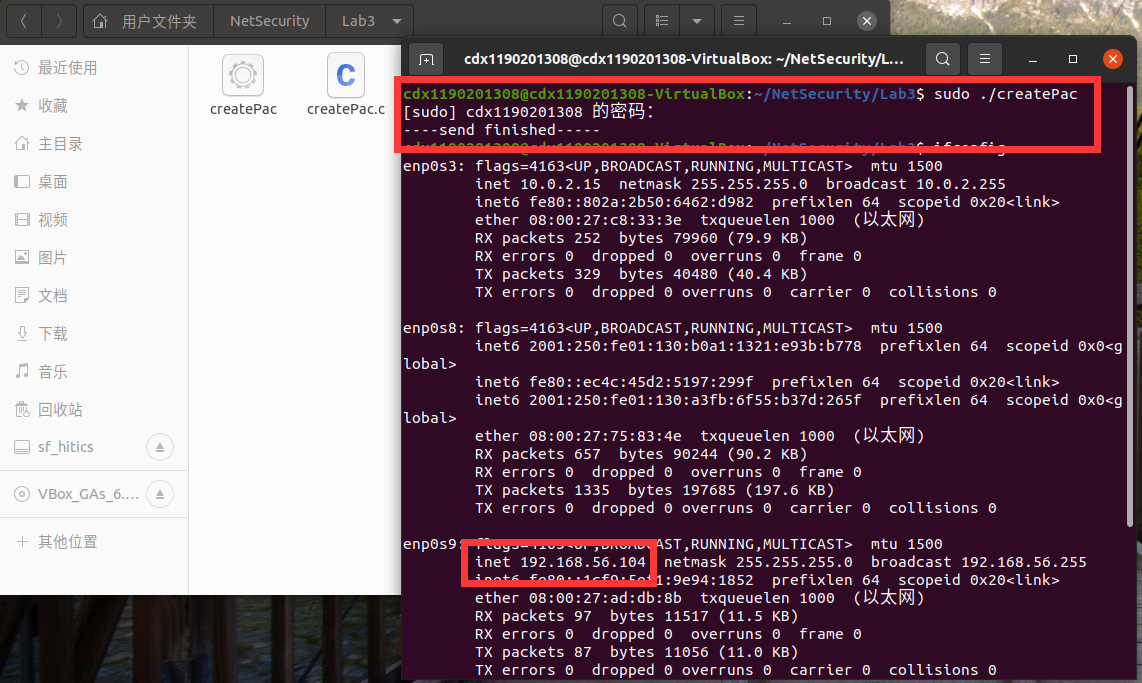
2)编写UDP接收程序recvUDP.c，运行在接收端虚拟机，查看能否接受到数据包，并且对比其数据是否是“From Function createPac”。



# 四、实验结果

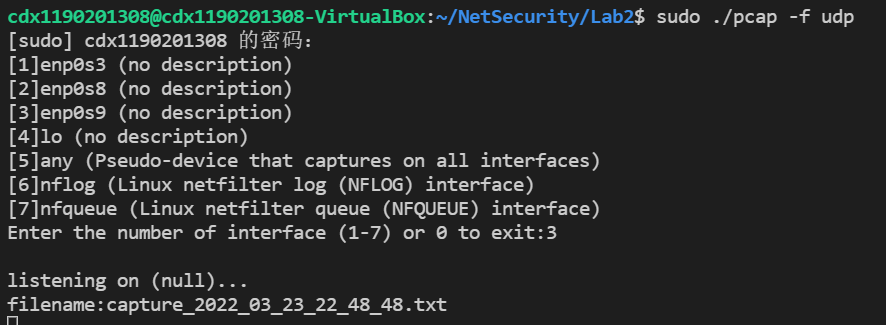
# 虚拟机192.168.56.103作为接收方，调用程序recvUDP.c和pcap.c接收构造的数据包，虚拟机192.168.56.104作为发送方，调用程序createPac发送使用libnet构造的数据包（上面的程序截图展示的是虚拟机192.168.56.103的程序代码，在虚拟机192.168.56.104调用程序时，需要调换源和目的的MAC地址和IP地址等，这里不再展示虚拟机192.168.56.104的程序截图）。

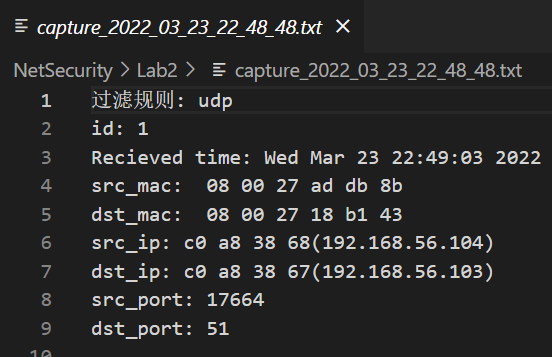
1.在虚拟机192.168.56.104使用createPac.c程序发送构造的数据包（逻辑上先在在虚拟机192.168.56.103调用recvUDP.c和pcap.c程序阻塞等待虚拟机192.168.56.104发送数据包）。



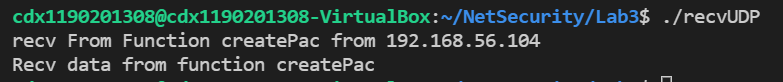
2.在虚拟机192.168.56.103使用recvUDP.c和pcap.c程序接收构造的数据包。

1）pcap.c





2）recvUDP.c



# 五、心得体会 （出现问题分析）

libnet构造数据包函数调用较简单，但编写程序后验证时程序出错。一开始出现了recvUDP.c能正确接收，但pcap.c捕捉不到的情况；后来发现自己误认为fprintf函数是行缓存，而程序持续捕捉，没有考虑缓存导致不能实时输出，于是由于UDP包数量太少，pcap.c没有输出，误以为没有捕捉到。将fprintf的缓存大小手动置为0，程序能够实时输出捕捉到的包。