**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Операционные системы»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9**

**Выполнили:**

Чу Ван Доан - N3247

Нгуен Тхе Вьет - N3247

Доан Тхи Хоай Тхыонг - N3245

**Проверил:**

Савков Сергей Витальевич

Санкт-Петербург

2023г.

**Запросы**

1. Простой вариант (одно из):

• Написать фильтр сетевых пакетов на основе nfqueue и iptables и протестировать скорость работы

• Протестировать работу сокетов tcp при различных настройках setsockopt

2. Сложный вариант (одно из):

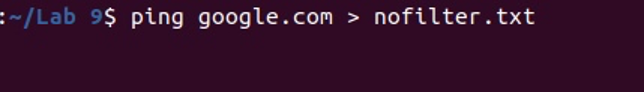
• Написать фильтр пакетов на основе интерфейса netfilter

• Реализовать rpc-программу для linux с поддержкой аутентификации (rpcinfo,rpcbind)

**I) Простой вариант: Написать фильтр сетевых пакетов на основе nfqueue и iptables и протестировать скорость работы**

* Сначала я проверил возможность подключения хоста google.com без фильтра:

**ping google.com > nofilter.txt**

****

Результаты написанны в файл nofilter.txt

* Программа фильтра сетевых пакетов на основе nfqueue и iptables

import netfilterqueue

import socket

import sys

from scapy.all import \*

def process(pkt):

data = pkt.get\_payload() #Returns pkt’s payload

p = IP(data)

p.ttl = 10 #time to live

del p.chksum #The statement removes variables, elements, keys, slices, and attributes

pkt.set\_verdict\_modified(nfqueue.NF\_ACCEPT, str(p), len(p))

#для отправки данных с помощью iptables и nfqueue

nfqueue = netfilterqueue.NetfilterQueue()

nfqueue.bind(1, process)

try:

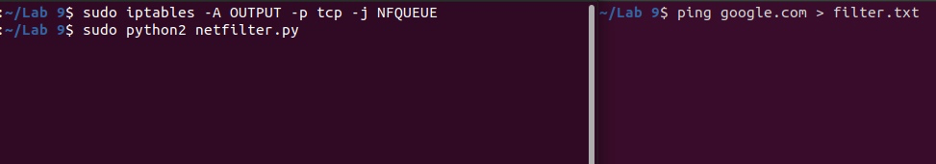
nfqueue.run()

except:

nfqueue.unbind()

sys.exit(1)

Я использовал iptables и nfqueue для получения пакета

****

- Iptables используется для установки, настройки и просмотра таблиц правил фильтрации IP-пакетов в ядре Linux. Соответственно в фильтре iptables все пакеты делятся на три аналогичные цепочки:

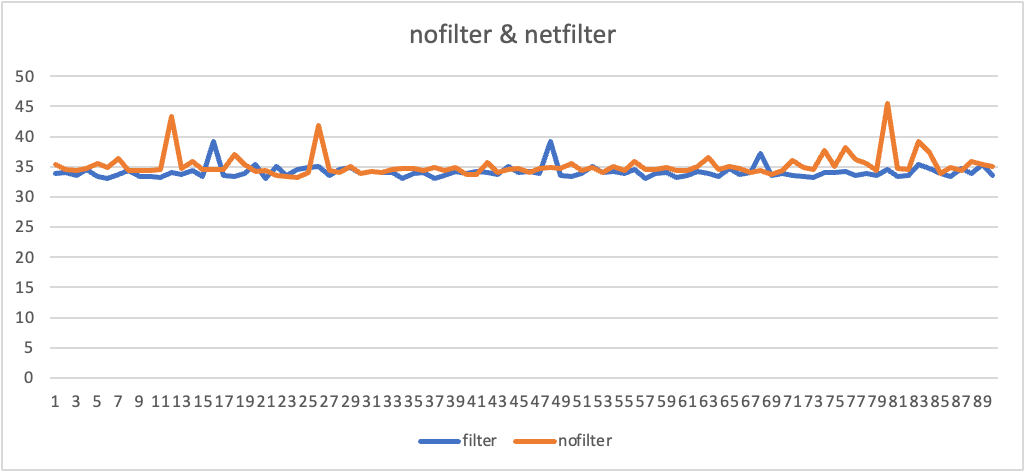
+ input - обрабатывает входящие пакеты и подключения. Например, если какой-либо внешний пользователь пытается подключиться к вашему компьютеру по ssh или любой веб-сайт отправит вам свой контент по запросу браузера;

+ forward - эта цепочка применяется для проходящих соединений. Сюда попадают пакеты, которые отправлены на ваш компьютер, но не предназначены ему, они просто пересылаются по сети к своей цели. Как я уже говорил, такое наблюдается на маршрутизаторах или, например, если ваш компьютер раздает wifi;

+ output - эта цепочка используется для исходящих пакетов и соединений. Сюда попадают пакеты, которые были созданы при попытке выполнить ping losst.ru или когда вы запускаете браузер и пытаетесь открыть любой сайт.

- Цель NFQUEUE используется почти так же, как цель QUEUE, и в основном является ее расширением. Цель NFQUEUE позволяет отправлять пакеты для отдельных и определенных очередей. Очередь идентифицируется 16-битным идентификатором.

* Результаты написанны в файл filter.txt



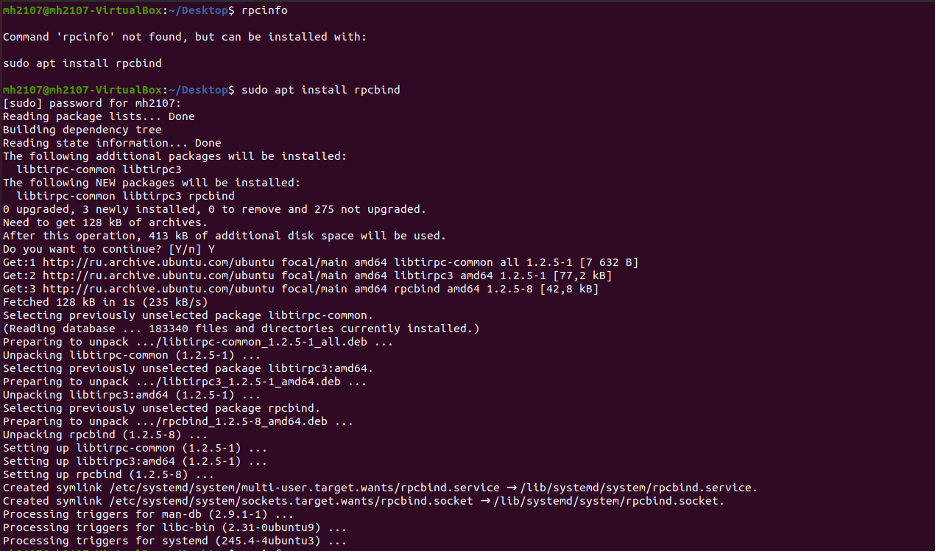
* Таблица среднее значение time (Время, которое потребовалось посылке, чтобы добраться до пункта назначения и вернуться в исходное положение)

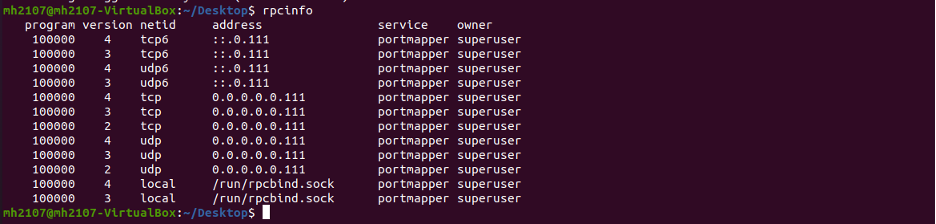
|  | netfilter | nofilter |
| --- | --- | --- |
| Среднее значение (ms) | 34.08666667 | 35.13777778 |

* Из графика и таблицы, мы видим, что после фильтра время было более стабильным и быстрым, чем до фильтра.

**II) Сложный вариант: Реализовать rpc-программу для linux с поддержкой аутентификации (rpcinfo, rpcbind)**

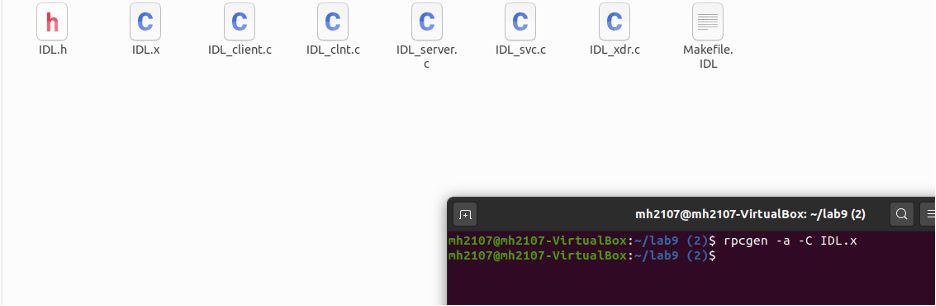
* Шаг 1: Установка rpcbind и проверка rpcinfo





* Шаг 2: Написание и компиляция файла IDL
* /\*Structure to hold the 2 values to be used in computation\*/
* struct values{
* float num1;
* float num2;
* char operation;
* };
* /\*Programm, version and procedure definition\*/
* program COMPUTE { //name of the programm.
* version COMPUTE\_VERS{ //the name of the programm version.
* float ADD(values) =1;
* float SUB(values)=2;float MUL(values)=3;
* float DIV(values)=4;
* } =6; // Arbitrary number used for identification
* } = 456123789; // Arbitrary number used for identification
* Шаг 3: Скомпилированные с помощью компилятора rpcgen

После этого, я получил несколько файлов:



* Шаг 4: Редактирование клиентских и серверных программ

**IDL\_client.c**

#include "IDL.h"

#include <stdio.h>

float compute\_6(char \*host,float a,float b,char op){

CLIENT \*clnt;

float \*result\_1;

values add\_6\_arg;

float \*result\_2;

values sub\_6\_arg;

float \*result\_3;

values mul\_6\_arg;

float \*result\_4;

values div\_6\_arg;

if(op=='+'){

add\_6\_arg.num1=a;

add\_6\_arg.num2=b;

add\_6\_arg.operation=op;

clnt = clnt\_create (host, COMPUTE, COMPUTE\_VERS, "udp");

if (clnt == NULL) {

clnt\_pcreateerror (host);

exit (1);

}

result\_1 = add\_6(&add\_6\_arg, clnt);

if (result\_1 == (float \*) NULL) {

clnt\_perror (clnt, "call failed");

}

clnt\_destroy (clnt);

return (\*result\_1);

}

else if(op=='-'){

sub\_6\_arg.num1=a;

sub\_6\_arg.num2=b;

sub\_6\_arg.operation=op;

clnt = clnt\_create (host, COMPUTE, COMPUTE\_VERS, "udp");

if (clnt == NULL) {

clnt\_pcreateerror (host);

exit (1);

}

result\_2 = sub\_6(&sub\_6\_arg, clnt);

if (result\_2 == (float \*) NULL) {

clnt\_perror (clnt, "call failed");

}

clnt\_destroy (clnt);

return (\*result\_2);

}

else if(op=='\*'){

mul\_6\_arg.num1=a;

mul\_6\_arg.num2=b;

mul\_6\_arg.operation=op;

clnt = clnt\_create (host, COMPUTE, COMPUTE\_VERS, "udp");

if (clnt == NULL) {

clnt\_pcreateerror (host);

exit (1);

}

result\_3 = mul\_6(&mul\_6\_arg, clnt);

if (result\_3 == (float \*) NULL) {

clnt\_perror (clnt, "call failed");

}

clnt\_destroy (clnt);

return (\*result\_3);

}

else if(op=='/'){

if(b==0){

printf("You are trying to divide by zero. Please insert a valid number.\n");

exit(0);

}

else{

div\_6\_arg.num1=a;

div\_6\_arg.num2=b;

div\_6\_arg.operation=op;

clnt = clnt\_create (host, COMPUTE, COMPUTE\_VERS, "udp");

if (clnt == NULL) {

clnt\_pcreateerror (host);

exit (1);

}

result\_4 = div\_6(&div\_6\_arg, clnt);

if (result\_4 == (float \*) NULL) {

clnt\_perror (clnt, "call failed");

}

clnt\_destroy (clnt);

return (\*result\_4);

}

}

}

int main (int argc, char \*argv[]){

char \*host;

float number1,number2;

char oper;

printf("Enter the 2 numbers followed by the operation to perform:\n");

scanf("%f",&number1);

scanf("%f",&number2);

scanf("%s",&oper);

host = argv[1];

printf("Answer= %f\n",compute\_6 (host,number1,number2,oper));

exit(0);

}

**IDL\_server.c**

#include "IDL.h"

#include <stdio.h>

float \*add\_6\_svc(values \*argp, struct svc\_req \*rqstp){

static float result;

result = argp->num1 + argp->num2;

return &result;

}

float \*sub\_6\_svc(values \*argp, struct svc\_req \*rqstp){

static float result;

result = argp->num1 - argp->num2;

return &result;

}

float \*mul\_6\_svc(values \*argp, struct svc\_req \*rqstp){

static float result;

result = argp->num1 \* argp->num2;

return &result;

}

float \*div\_6\_svc(values \*argp, struct svc\_req \*rqstp){

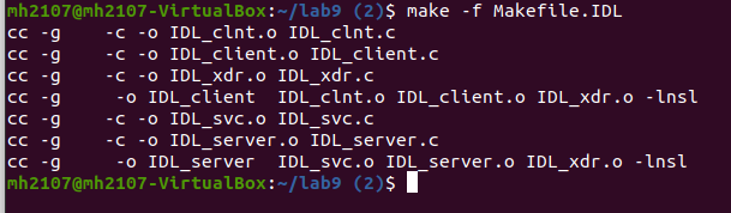
static float result;

result = argp->num1 / argp->num2;

return &result;

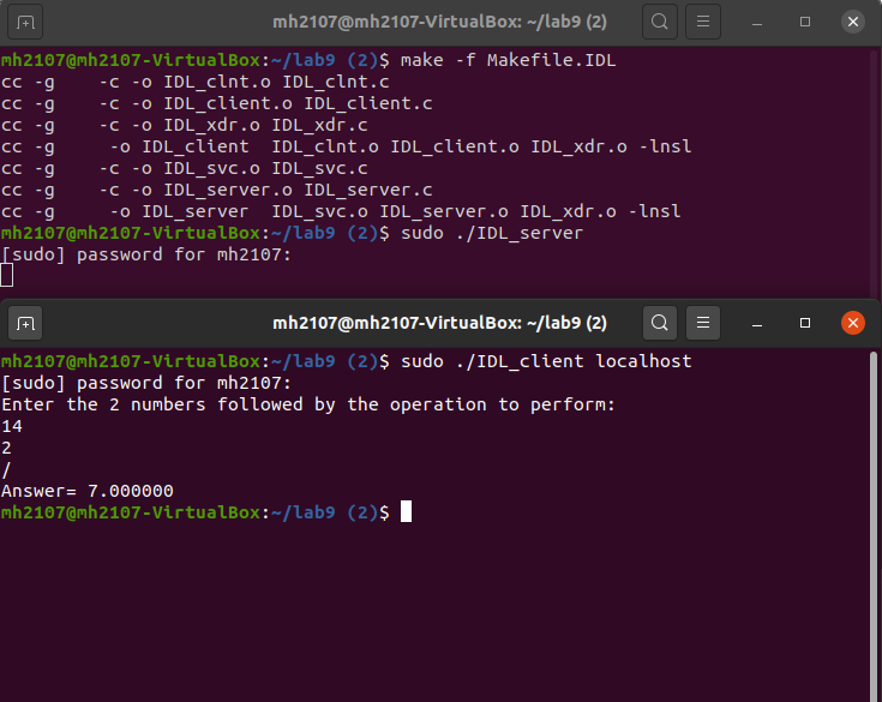
}

* Шаг 5: Компиляция всех файлов



* Шаг 6: Запуск сервера и клиента

Я открыл другой терминал и запустил клиент там.



**II) Вывод:**

- Написал код питон для фильтра сетевых пакетов на основе **nfqueue** и **iptables**

и протестировал скорость хоста **google.com**

- **firewalling** решает, каким пакетам разрешено входить и выходить из системы. **firewalling** обеспечивается с помощью **netfilter**. **iptables** - это просто интерфейс для **netfilter**.

- Если **INPUT** цепочка не означает более безопасную, потому что при работе сети используются как **INPUT**, так и **OUTPUT**.

- После выполнения работы мы написали rpc-программу на linux с поддержкой аутентификации.