МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №52

ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ

# ПРЕПОДАВАТЕЛЬ

Старший преподаватель Матвеев Н.В.

должность, уч. степень, звание подпись, дата инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 1

Анализ и эмулирование сетевого трафика в сетях стандарта 802.3

по курсу: ОСНОВЫ КОНСТРУИРОВАНИЯ ТЕХНОЛОГИИ И ЭКСПЛУАТАЦИИ ТЕЛЕКОММУНИКАЦИОННОГО ОБОРУДОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 5721 Г. А. Михайлов

подпись, дата инициалы, фамилия

Санкт-Петербург 2020

* Цель и постановка задачи

Целью данной лабораторной работы является освоение базовых навыков настройки и анализа трафика в локальных сетях передачи данных стандарта 802.3. А так же изучение основных методов настройки маршрутизируемых компьютерных сетей на примере сети, состоящей из компьютеров под управлением ОС Linux и маршрутизаторов на примере cisco.

* Описание допуска к лабораторной работе

Мои исходные данные: Схема 4, Вариант 4.

Для выполнения допуска к лабораторной работе необходимо собрать схему, как на рис.1

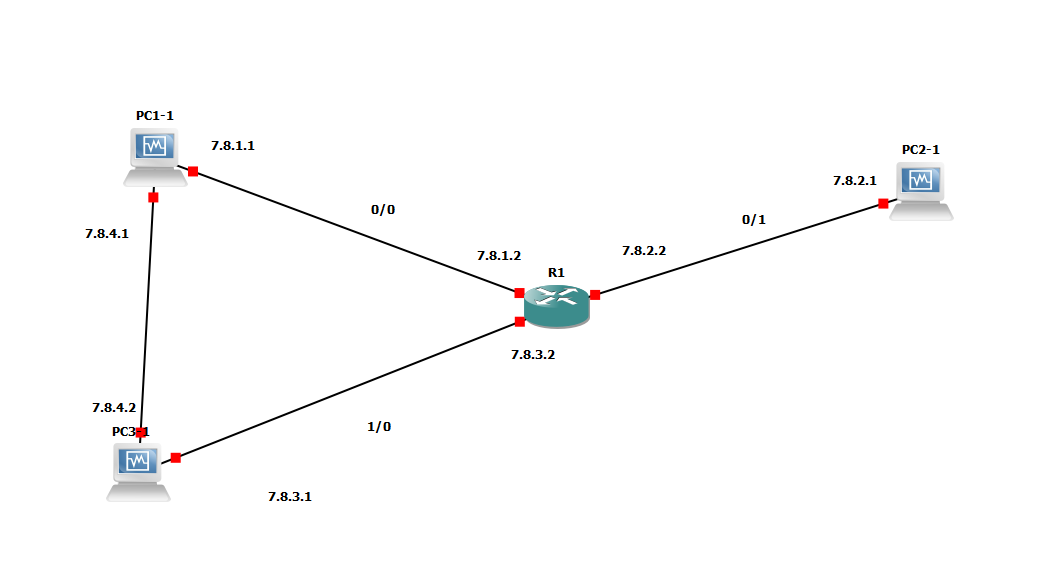
Затем на системах из данной схемы необходимо выполнить два задание с утилитами iperf и tcpdump

Задание:

Утилита iperf: Передать файл с произвольными данными размером 32 Мбайт.

Утилита tcpdump: Перехватить только пакеты содержащие файл.

* Описание выполнения допуска к лабораторной работе



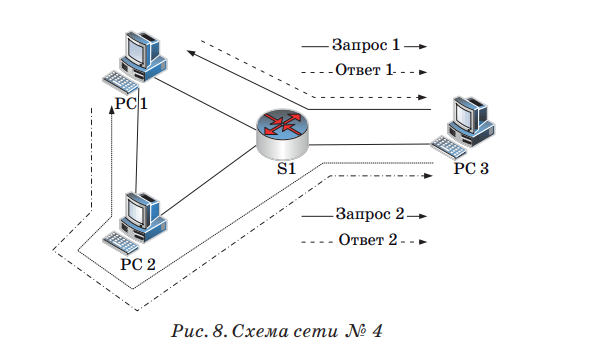


Рис.1 – Схема макета с выбранными IP адресами

При выполнении задания в программном эмуляторе GNS3 получилась схема представленная на Рис. 1

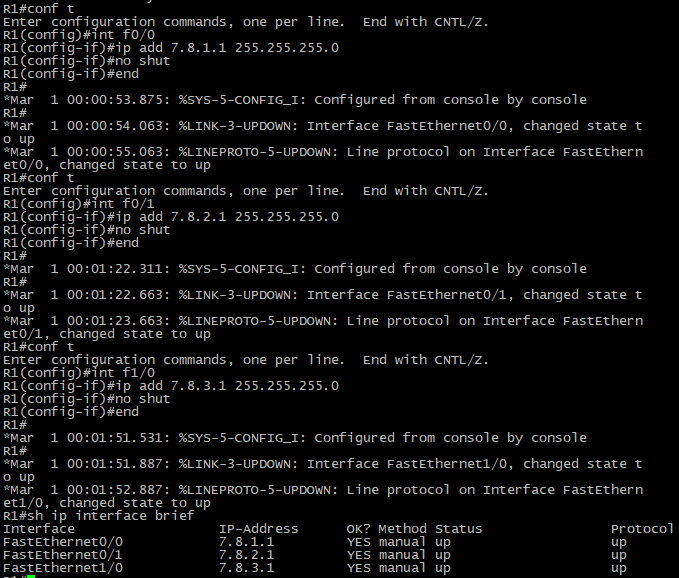


Рис.2 – Добавление новых ip для маршрутизатора

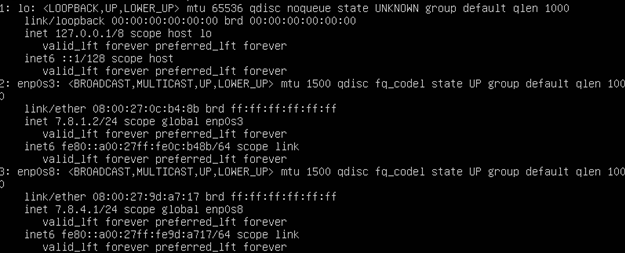


Рис.3 – Настройка интерфейсов на машине 1

Интерфейсы на остальных виртуальных машинах настраиваются аналогично.

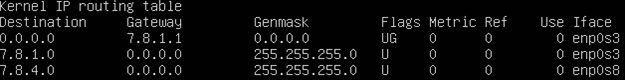


Рис.6 – Таблица маршрутизации на машине 1 для 1 запроса

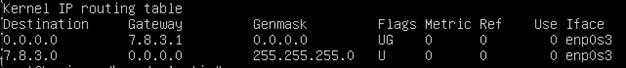


Рис.7 – Таблица маршрутизации на машине 3

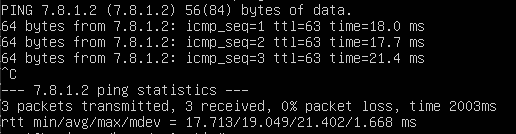


Рис.8 – Результат работы первого запроса

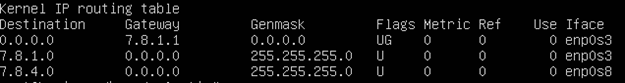


Рис.9 – Таблица маршрутизации на 1 машине для запроса 2

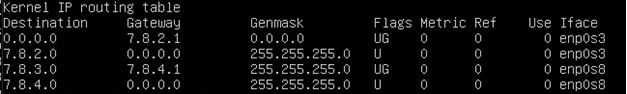


Рис.10 – Таблица маршрутизации на машине 2 для запроса 2

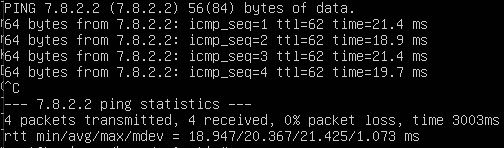


Рис.11 – Результат работы 2 запроса

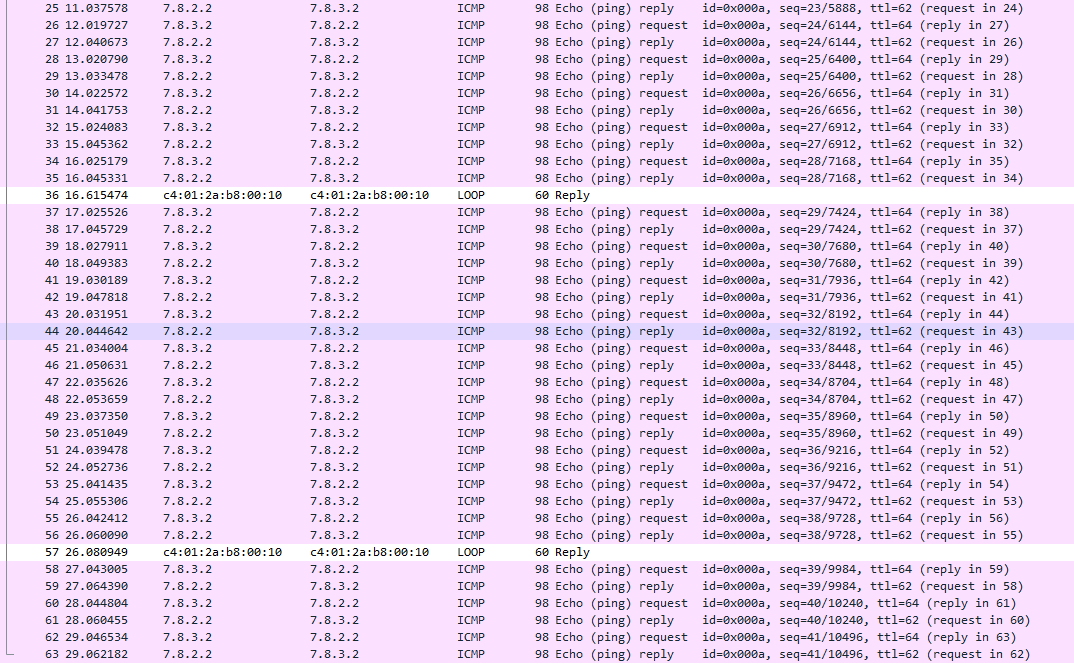


Рис.12 – Результат работы в wireshark

Команды необходимые для выполнения допуска

Прописываем следующую команду для создания файла:

head –c 32M</dev/urandom >myfile

Iperf: iperf3 -s на серверной части

Ключ -s означает запуск серверной части

Iperf3 -c 7.8.1.2 -z -F file.txt –b 500m –w 3m

Ключ -c означает запуск клиентской части, -z(ZEROCOPY), -F файл который нужно передать, -b 500m – формат отчетов для bandwidth (средней скорости), -w 3m – размер сетевого окна.

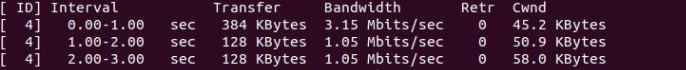


Рис.13 – Передача файла

Tcpdump: Перехватить пакеты только содержащие файл:

tcpdump –vv tcp

добавить итоги работы

* Описание задания к лабораторной работе

Вариант 4: Перехватить пакеты по протоколу http, собрать из пакетов в отдельную папку картинки с сайта.

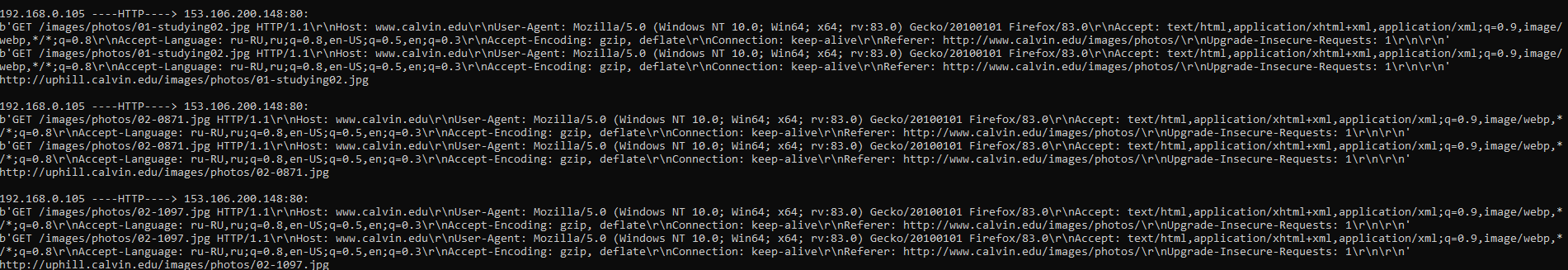


Рис.13 – Результат работы программы

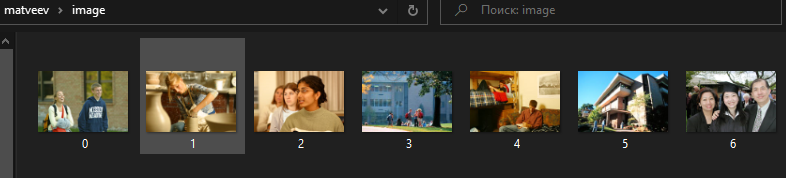


Рис.14 – Скачанные картинки

* Выводы

В ходе данной лабораторной работы были изучены:

1. Основы программы GNS3 для моделирования сетей
2. Процесс настройки машин под управлением ОС Linux
3. Настройка маршрутизатора cisco
4. Основы сетевого трафика

Так же была разработана программа для получения данных из сетевого трафика, благодаря чему мы разобрались с таким вещами, как пакеты передачи информации.

Листинг программы

import base64  
import re  
from urllib.request import urlopen  
  
from scapy.all import \*  
from scapy.layers.dns import DNS, DNSQR  
from scapy.layers.inet import TCP, IP  
import logging  
import urllib  
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)  
import re  
  
jpg\_url\_list = []  
png\_url\_list = []  
gif\_url\_list = []  
  
  
def packet\_callback(packet):  
 if packet[TCP].payload:  
 pkt = str(packet[TCP].payload)  
 if packet[IP].dport == 80:  
 print("\n{} ----HTTP----> {}:{}:\n{}".format(packet[IP].src, packet[IP].dst, packet[IP].dport, str(bytes(packet[TCP].payload))))  
 print(str(bytes(packet[Raw])))  
 TCPHttpExtract(packet)  
  
def get\_data(network\_packets):  
 decoded\_commands = []  
 decoded\_data = ""  
 for packet in network\_packets:  
 if DNSQR in packet:  
 if packet[DNS].id == 0x1337:  
 decoded\_data = base64.b64decode(str(packet[DNS].an.rdata))  
 print(decoded\_data.decode('utf-8'))  
 return decoded\_data  
  
  
def TCPHttpExtract(packet):  
 if packet.haslayer(TCP) and packet.getlayer(TCP).dport == 80 and packet.getlayer(Raw):  
 StringJPG = ""  
 StringPNG = ""  
 StringGIF = ""  
 liste = []  
  
 for line in packet.getlayer(Raw):  
 liste.append(line)  
 StringPacket = re.findall('(\s\/.\*?\s)', str(liste))  
 StringJPG = re.findall('.\*\.jpg', str(StringPacket))  
 StringPNG = re.findall('.\*\.png', str(StringPacket))  
 StringGIF = re.findall('.\*\.gif', str(StringPacket))  
 if (StringJPG):  
 data = socket.gethostbyaddr(packet[IP].dst)  
 host = repr(data[0])  
 image\_url = "http://" + host.replace("'", '') + ''.join(StringJPG).replace("[' ", '')  
 #print(get\_data(packet))  
 print(image\_url)  
 jpg\_url\_list.append(image\_url)  
 if (StringPNG):  
 data = socket.gethostbyaddr(packet[IP].dst)  
 host = repr(data[0])  
 image\_url = "http://" + host.replace("'", '') + ''.join(StringPNG).replace("[' ", '')  
 print(image\_url)  
 png\_url\_list.append(image\_url)  
 if (StringGIF):  
 data = socket.gethostbyaddr(packet[IP].dst)  
 host = repr(data[0])  
 image\_url = "http://" + host.replace("'", '') + ''.join(StringGIF).replace("[' ", '')  
 print(image\_url)  
 gif\_url\_list.append(image\_url)  
  
def download\_image():  
 i = 0  
 for image in jpg\_url\_list:  
 urllib.request.urlretrieve(image, "image/" + i.\_\_str\_\_() + ".jpg")  
 i+=1  
 for image in png\_url\_list:  
 urllib.request.urlretrieve(image, "image/" + i.\_\_str\_\_() + ".png")  
 i+=1  
 for image in gif\_url\_list:  
 urllib.request.urlretrieve(image, "image/" + i.\_\_str\_\_() + ".gif")  
 i+=1  
  
def capture\_int(name\_int, file, time):  
 packets = sniff(timeout=time, iface=name\_int)  
 wrpcap(file, packets)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 sniff(timeout=30, filter="tcp", prn=packet\_callback ,store=0)  
 download\_image()