Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЁТ

к лабораторной работе №3

на тему

**Атаки при установке TCP-соединения и протоколов прикладного уровня**

Выполнил Даниленко И.А.

Проверил Лещенко Е. А.

Минск 2024

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc160203598)

[2 Результат выполнения 4](#_Toc160203599)

[ЗАКЛЮЧЕНИЕ 7](#_Toc160203600)

[Приложение А Листинг кода 8](#_Toc160203601)

# 1 ЦЕЛЬ РАБОТЫ

Целью выполнения данной лабораторной работы является изучение теоретических сведений по работе адресации в сети Internet, физического, канального и транспортного уровней, а также разработка программы, реализующей работу TCP-протокола и атак на TCP-соединение.

# 2 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

В ходе выполнения лабораторной работы было создано консольное приложение, реализующее работу TCP-протокола и атак на TCP-соединение. На рисунке 2.1 показана работа выполнения программы без различных атак на протокол TCP.

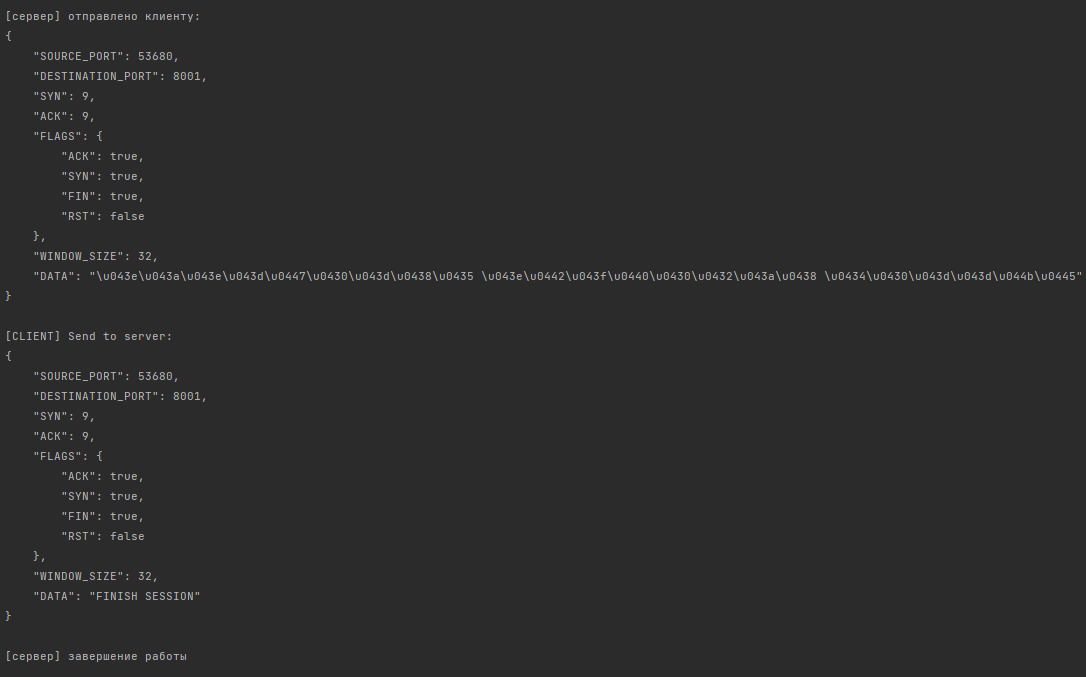


Рисунок 2.1 – Результат выполнения программы без атак

В качестве атак на протокол TCP были выбраны две атаки: SYN Flooding и TCP Reset Attack. На рисунке 2.2 показано выполнение работы атаки SYN Flooding. В этой атаке злоумышленник посылает большое количество SYN запросов на целевой сервер, но не завершает установку соединения, не отправляя ACK в ответ на SYN+ACK пакеты от сервера.

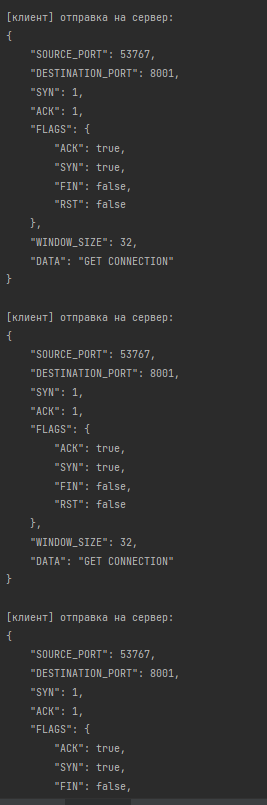


Рисунок 2.2 – Результат работы атаки SYN Flooding

Следующей атакой является TCP Reset Attack. Злоумышленник может подделать пакет RST (Reset) и отправить его на сервер или клиент, чтобы принудительно завершить TCP-соединение. Это может быть использовано для прерывания активного соединения и вызывать проблемы в работе приложения. На рисунке 2.3 показана работа выполнения атаки TCP Reset.

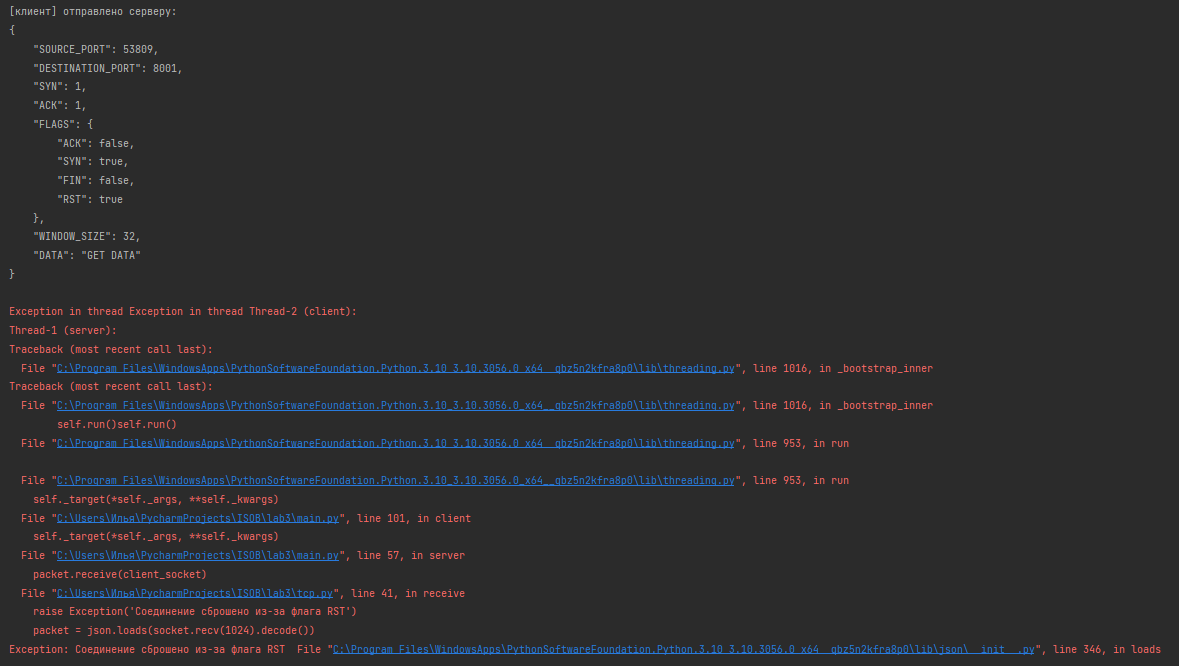


Рисунок 2.3 – Результат работы атаки TCP Reset

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы были изучены теоретические сведения по работе адресации в сети Internet, физического, канального и транспортного уровней, а также была разработана программа, реализующая работу TCP-протокола и атак на TCP-соединение

# ПРИЛОЖЕНИЕ А Листинг кода

main.py

import socket

import threading

import time

from constants import IP\_ADDRESS, SERVER\_PORT, DATA\_SIZE, MESSAGE

from tcp import TCPPacket, divide\_into\_packages

server\_address = (IP\_ADDRESS, SERVER\_PORT)

BUFF\_CLIENT = -1

BUFF\_SERVER = 1

def server():

global BUFF\_CLIENT

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

server\_socket.bind(server\_address)

# start server

server\_socket.listen(1)

print('[сервер] Сервер открыт для подключения', '\n')

time.sleep(0.5)

# get client connection

client\_socket, client\_address = server\_socket.accept()

print(f'[сервер] соединение установлено с {client\_address}', '\n')

time.sleep(0.5)

# get connection request from client and procces it to send back

packet = TCPPacket()

packet.receive(client\_socket)

packet.SYN += packet.ACK

packet.flags['ACK'] = True

print('[сервер] отправка клиенту:')

print(packet, '\n')

packet.send(client\_socket)

packet.receive(client\_socket)

# SYN Flooding (серверная часть)

# while packet.data != 'GET DATA':

# packet.receive(client\_socket)

if packet.data == 'GET DATA':

BUFF\_CLIENT = packet.SYN + len(packet.data)

packet.data = MESSAGE

packets = divide\_into\_packages(packet, DATA\_SIZE)

last\_packet = None

i = 0

for packet\_ in packets:

# fictional statement

if i > 0:

packet.receive(client\_socket)

# TCP Reset Attack (Server side)

if packet.flags['RST']:

raise Exception('Соединение сброшено из-за флага RST')

# send one of the packets sing buffer variables

packet\_.ACK = BUFF\_CLIENT

packet\_.flags['ACK'] = True

packet\_.SYN = BUFF\_SERVER

print('[сервер] отправлено клиенту:')

print(packet\_, '\n')

packet\_.send(client\_socket)

last\_packet = packet\_

i += 1

packet.receive(client\_socket)

# form and send finish request to client

last\_packet.flags['FIN'] = True

last\_packet.data = 'окончание отправки данных'

print('[сервер] отправлено клиенту:')

print(last\_packet, '\n')

last\_packet.send(client\_socket)

# get from client confirmation to disconnect

last\_packet.receive(client\_socket)

if last\_packet.data == 'FINISH SESSION':

print('[сервер] завершение работы', '\n')

client\_socket.close()

server\_socket.close()

finally:

client\_socket.close()

server\_socket.close()

def client():

global BUFF\_SERVER

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

try:

# start client

client\_socket.connect(server\_address)

print('[клиент] установлено соединение с сервером.', '\n')

client\_port = client\_socket.getsockname()[1]

# get connection request

packet = TCPPacket(client\_port, SERVER\_PORT, 0, 1, DATA\_SIZE, 'GET CONNECTION', syn=True)

print('[клиент] отправлено серверу:')

print(packet, '\n')

packet.send(client\_socket)

while True:

packet.receive(client\_socket)

# SYN Flooding (клиентская часть:)

# while True:

# print('[клиент] отправка на сервер:')

# print(packet, '\n')

# packet.send(client\_socket)

# TCP Reset Attack (клиентская часть)

packet.flags['RST'] = True

# send confirmation to server about finishing connection

if packet.flags['FIN']:

packet.data = 'FINISH SESSION'

print('[CLIENT] Send to server:')

print(packet, '\n')

packet.send(client\_socket)

break

# после первого получения пакета

if BUFF\_CLIENT != -1:

BUFF\_SERVER = packet.SYN + len(packet.data)

packet.ACK = BUFF\_SERVER

packet.SYN = BUFF\_CLIENT

packet.flags['ACK'] = False

packet.data = 'GET DATA'

print('[клиент] отправлено серверу:')

print(packet, '\n')

packet.send(client\_socket)

finally:

client\_socket.close()

if \_\_name\_\_ == '\_\_main\_\_':

server\_thread = threading.Thread(target=server)

client\_thread = threading.Thread(target=client)

server\_thread.start()

client\_thread.start()

server\_thread.join()

client\_thread.join()

constants.py

IP\_ADDRESS = '127.0.0.1'

SERVER\_PORT = 8001

DATA\_SIZE = 32

MESSAGE = 'test message'

**tcp.py**

import json

import socket

import time

class TCPPacket:

def \_\_init\_\_(self, source\_port=None, destionation\_port=None, syn\_n=0,

ack\_n=0, data\_size=None,

data='', ack=False, syn=False, fin=False, rst=False) -> None:

self.source\_port = source\_port

self.destionation\_port = destionation\_port

self.SYN = syn\_n # номер последовательности для установки соед

self.ACK = ack\_n # номер для подтверждения получения данных

self.flags = {

'ACK': ack,

'SYN': syn,

'FIN': fin,

'RST': rst

}

self.data\_size = data\_size

self.data = data

def \_\_str\_\_(self) -> str:

return json.dumps({

'SOURCE\_PORT': self.source\_port,

'DESTINATION\_PORT': self.destionation\_port,

'SYN': self.SYN,

'ACK': self.ACK,

'FLAGS': self.flags,

'WINDOW\_SIZE': self.data\_size,

'DATA': self.data

}, indent=4)

def send(self, socket: socket.socket) -> None:

socket.sendall(self.\_\_str\_\_().encode())

time.sleep(0.5)

def receive(self, socket: socket.socket) -> None:

packet = json.loads(socket.recv(1024).decode())

self.\_\_init\_\_(packet['SOURCE\_PORT'], packet['DESTINATION\_PORT'],

packet['SYN'], packet['ACK'],

packet['WINDOW\_SIZE'], packet['DATA'],

ack=packet['FLAGS']['ACK'],

syn=packet['FLAGS']['SYN'], fin=packet['FLAGS']['FIN'],

rst=packet['FLAGS']['RST'])

time.sleep(0.5)

def divide\_into\_packages(tcp\_packet: TCPPacket, data\_size: int) -> list[TCPPacket]:

size\_of\_data = data\_size // 8

data = [tcp\_packet.data[i:i + size\_of\_data] for i in range(0, len(tcp\_packet.data), size\_of\_data)]

packets = []

for d in data:

packet = TCPPacket(tcp\_packet.source\_port,

tcp\_packet.destionation\_port,

tcp\_packet.SYN, tcp\_packet.ACK,

tcp\_packet.data\_size, d,

tcp\_packet.flags['ACK'], tcp\_packet.flags['SYN'],

tcp\_packet.flags['FIN'], tcp\_packet.flags['RST'])

packets.append(packet)

return packets