Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЕТ

к лабораторной работе №3

на тему

**АТАКИ ПРИ УСТАНОВКЕ TCP-СОЕДИНЕНИЯ И ПРОТОКОЛОВ ПРИКЛАДНОГО УРОВНЯ**

Студент Д. С. Кончик

Преподаватель Е. А. Лещенко

Минск 2024

СОДЕРЖАНИЕ

[Введение 3](#_Toc157471308)

[1 Результат выполнения 4](#_Toc157471309)

[Заключение 5](#_Toc157471310)

[Приложение А (обязательное) Листинг кода 6](#_Toc157471311)

**ВВЕДЕНИЕ**

Целью выполнения данной лабораторной работы является изучение теоретического сведения, касающегося протокола TCP, а также создание приложения, реализующего атаки на протокол при установке TCP-соединения и в рамках заданного протокола прикладного уровня.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате работы было создано приложение, которое имитирует связь клиента с сервером через TCP соединение и организует на него атаку SYN Flood. Происходит эмуляция клиента, сервера и пакета через классы Python, каждый раз клиент вписывает в поле IP адреса источника случайно сгенерированное значение.

На рисунке 1 показана работа программы при проведении атаки.

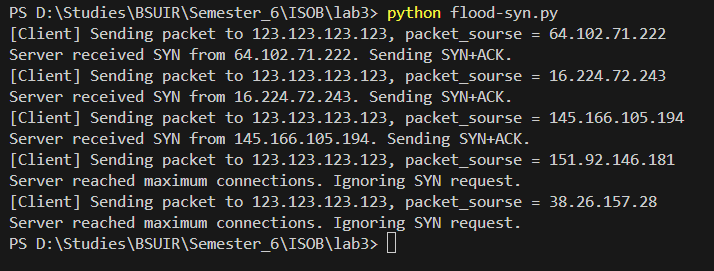


Рисунок 1 – Результат SYN Flood атаки

# ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы было создано приложение, реализующее атаки на протокол при установке TCP-соединения и в рамках заданного протокола прикладного уровня.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл *lab3.py*

import random

import time

class Packet:

    def \_\_init\_\_(self, src\_ip, dst\_ip, syn=False, ack=False):

        self.src\_ip = src\_ip

        self.dst\_ip = dst\_ip

        self.syn = syn

        self.ack = ack

class Server:

    def \_\_init\_\_(self, max\_pending\_connections, server\_ip):

        self.server\_ip = server\_ip

        self.max\_pending\_connections = max\_pending\_connections

        self.pending\_connections = {}

    def receive\_packet(self, packet):

        if packet.syn and packet.src\_ip not in self.pending\_connections:

            if len(self.pending\_connections) < self.max\_pending\_connections:

                self.pending\_connections[packet.src\_ip] = True

                print(f"Server received SYN from {packet.src\_ip}. Sending SYN+ACK.")

                return Packet(src\_ip=packet.dst\_ip, dst\_ip=packet.src\_ip, syn=True, ack=True)

            else:

                print("Server reached maximum connections. Ignoring SYN request.")

                return None

        elif packet.ack and packet.src\_ip in self.pending\_connections:

            del self.pending\_connections[packet.src\_ip]

            print(f"Server received ACK from {packet.src\_ip}. Connection established.")

        else:

            print("Server received unexpected packet. Ignoring.")

def generate\_random\_ip():

    return ".".join(str(random.randint(0, 255)) for \_ in range(4))

class Client:

    def send\_packet(self, server):

        src\_ip = generate\_random\_ip()

        packet = Packet(src\_ip=src\_ip, dst\_ip=server.server\_ip, syn=True)

        print(f"[Client] Sending packet to {packet.dst\_ip}, packet\_sourse = {packet.src\_ip}")

        server.receive\_packet(packet)

        return packet

def main():

    server = Server(max\_pending\_connections=3, server\_ip='123.123.123.123')

    client = Client()

    for i in range(5):  # Sending 5 SYN packets

        response = client.send\_packet(server)

if \_\_name\_\_ == "\_\_main\_\_":

    main()