Скрытые Каналы. Лабораторная работа #1.

Соколов А.Д. Б20-505

Вариант 7

Модель 4. Пакеты случайной длины передаются в случайные моменты времени.

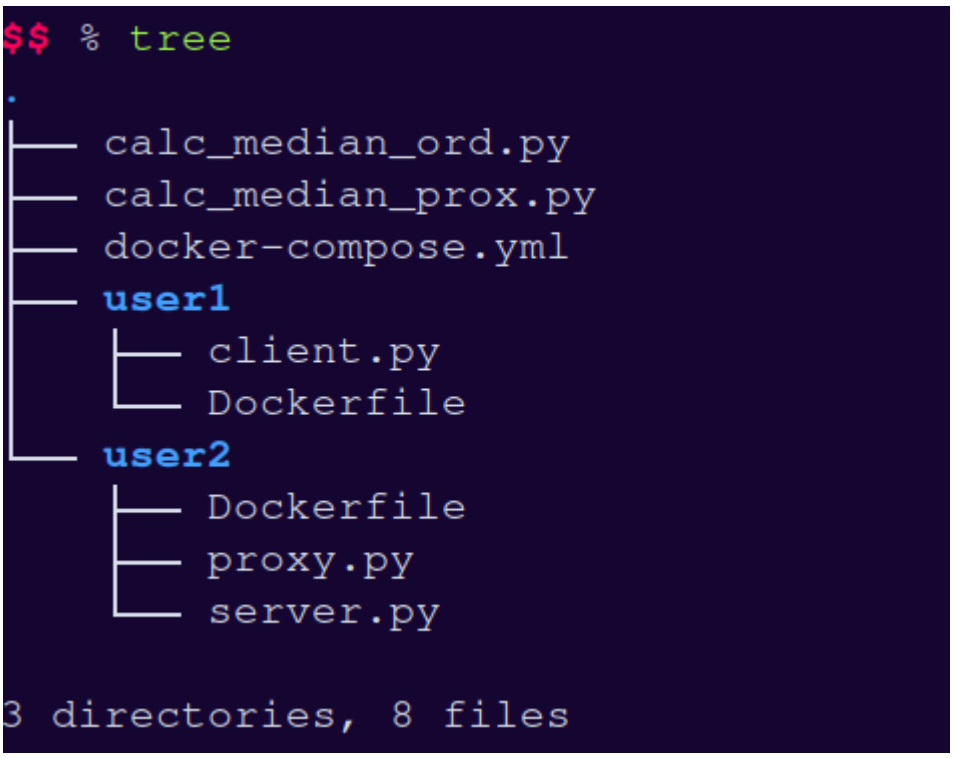
Пример 9. Time Relay Covert Channel

Исходя из измерения возможных значений задержек между пакетами в открытом
трафике, определяется медианное значение этого множества, и при передаче скрытого
сообщения для кодирования 0 выбирается значение межпакетного интервала из
подмножества значений, которые больше медианного, а для 1 — которые меньше

Возможности закладки:

- Буферизация трафика
- Генерация фиктивного трафика

Архитектура



Есть сокет сервера, который непрерывно слушает.
Есть сокет закладки, которая непрерывно слушает и делает свои дела с полученными пакетами.
Есть клиент - сокет который отправляет n пакетов на порт закладки.

Реализация

client.py

```python
import argparse
import socket
from os import urandom
from random import randint, random
from time import sleep


def model_packets(host, port, n_packets=200):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    sent = 0
    while sent < n_packets:
        data = urandom(randint(1, 100))
        data = len(data).to_bytes(4, "big") + data
        client.send(data)

        sent += 1
        sleep(random())
    client.close()


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-p", "--port", help="server running port", type=int, required=True
    )
    parser.add_argument("--host", help="server running host", type=str, required=True)
    args = parser.parse_args()

    model_packets(args.host, args.port)
```

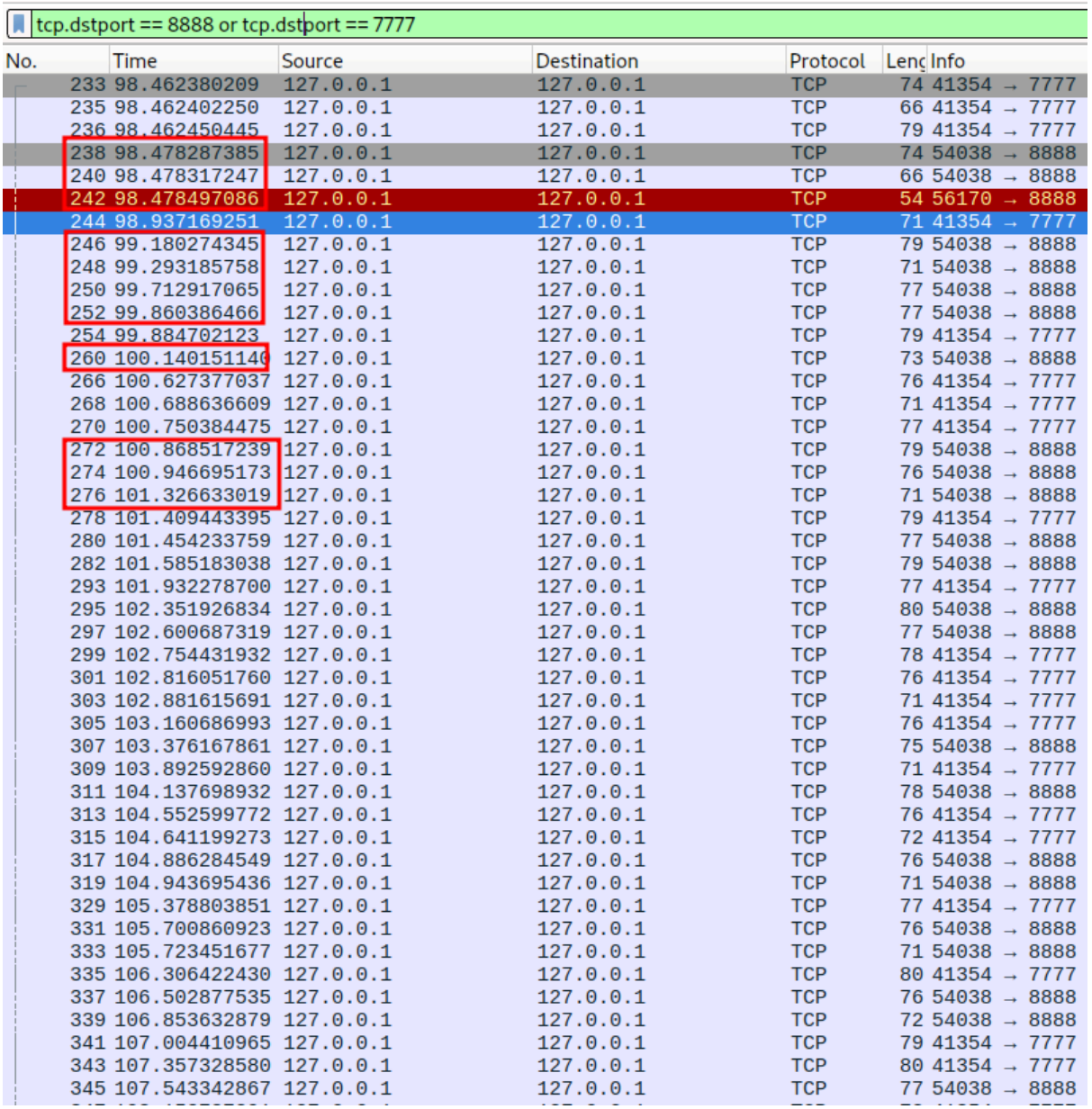
proxy.py

```python
import socket

from logging import error
import argparse

from time import sleep
from random import random, randint
from os import urandom

from queue import Queue
from threading import Thread

from termcolor import colored

# from concurrent.futures import ThreadPoolExecutor

M = 0.48871803283691406

def start_proxy(port: int, server_host: str, server_port: int, message: str):
    q = Queue()
    msg = int.from_bytes(message.encode(), "big")
    finished = False

    def listen_and_buffer(port):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        sock.bind(("localhost", port))
        sock.listen(1)

        client, addr = sock.accept()
        error(colored(f"PROXY::Connection from {addr}", "red"))

        while True:
            data = client.recv(1024)
            error(colored(f"PROXY::Received data {data}", "red"))
            if not data:
                break
            q.put(data)
        client.close()

        global finished
        finished = True

```

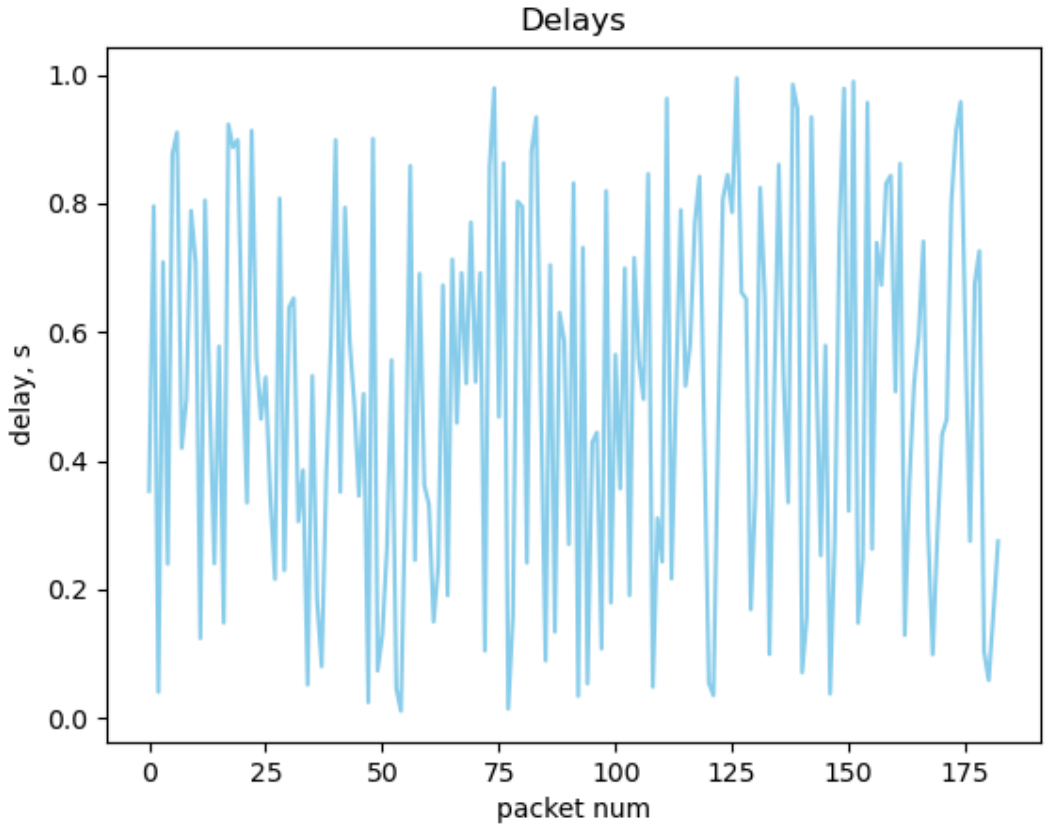
```
    def be_proxy_and_send_message(host: str, port: port, msg):
        data = b""
        delay = 0.0
        started = False
        server = None
        while msg or not finished:
            print(finished)
            if msg and started:
                if msg & 1:
                    delay = 0.1 * random() + 0.2 + M
                else:
                    delay = random() * 0.9 * M
                msg >>= 1

                if q.empty():
                    data = urandom(randint(1, 100))
                    data = len(data).to_bytes(4, "big") + data
                else:
                    data = q.get()

                error(colored(f"PROXY::sending data {data}", "blue"))
                sleep(delay)
                server.send(data)
            elif not q.empty():
                if started:
                    data = q.get()
                    server.send(data)
                else:
                    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                    error(colored(f"PROXY::trying {host}:{port}", "blue"))
                    server.connect((host, port))

                    started = True

        server.close()

    finished = False
    receive = Thread(target=listen_and_buffer, args=(port,))
    receive.start()
    send = Thread(
        target=be_proxy_and_send_message, args=(server_host, server_port, msg)
    )
    send.start()
```

```python
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-p", "--port", help="proxy running port", type=int, required=True
    )
    parser.add_argument(
        "--server-port", help="server running port", type=int, required=True
    )
    parser.add_argument(
        "--server-host", help="server running host", type=str, required=True
    )
    parser.add_argument(
        "-m", "--message", help="secret message", type=str, required=True
    )
    args = parser.parse_args()

    start_proxy(args.port, args.server_host, args.server_port, args.message)
```

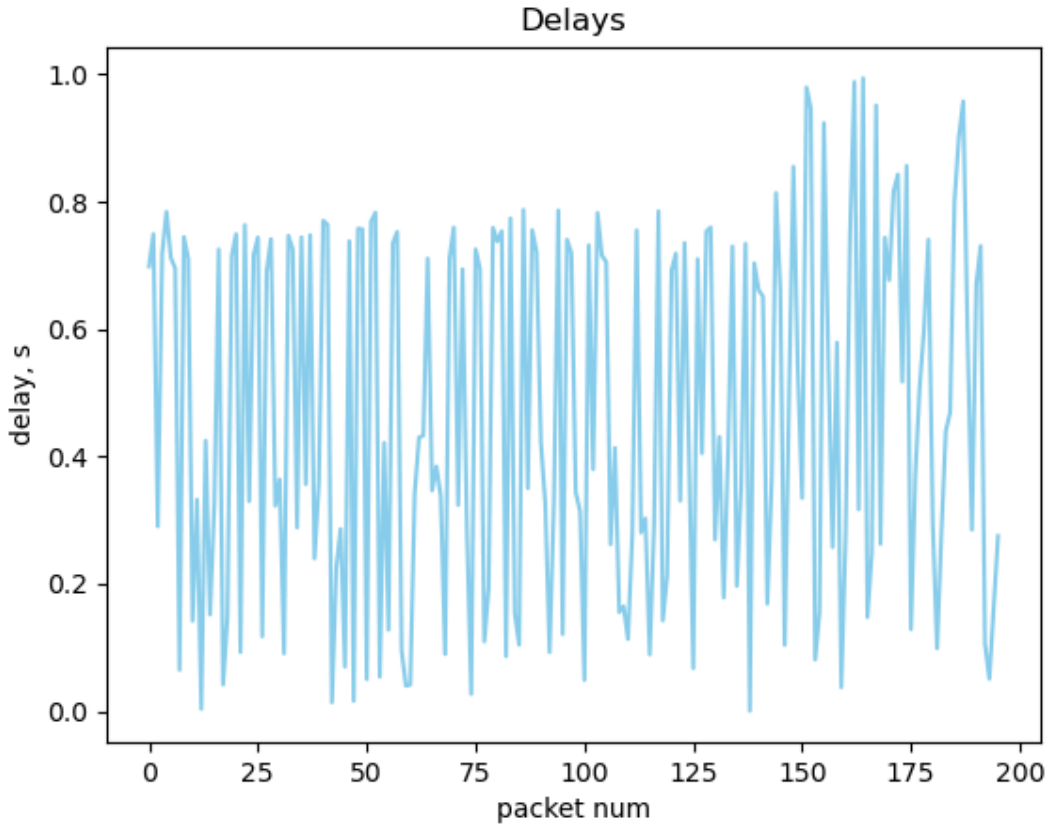
Демонстрация работы закладки

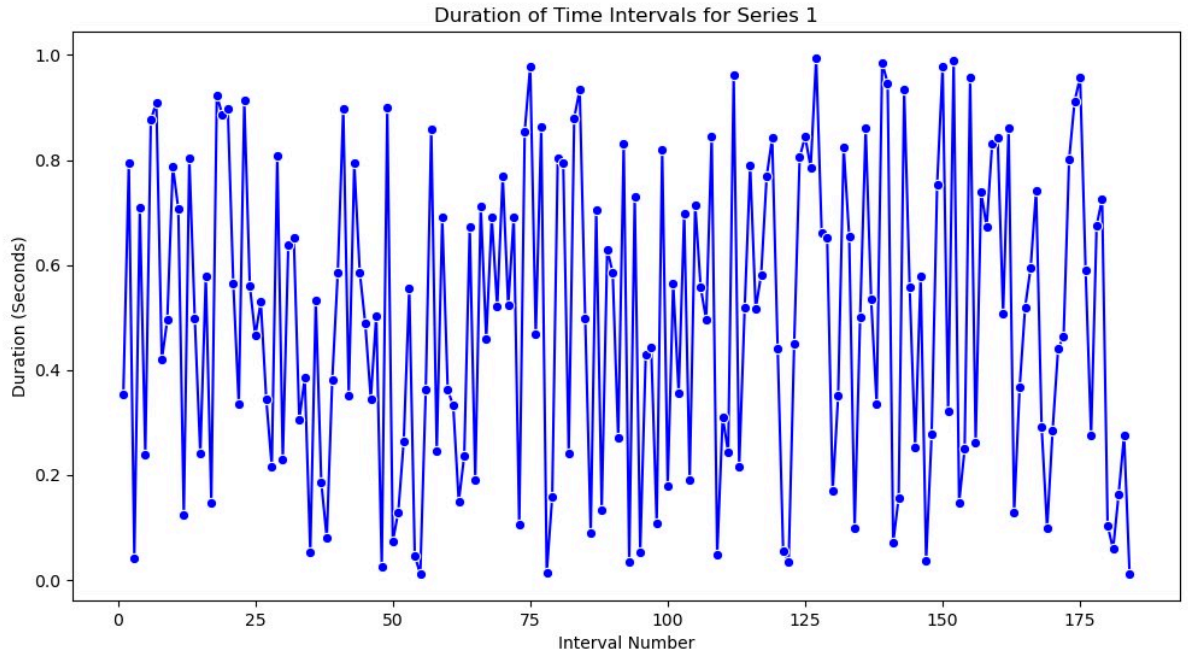

tcp.dstport == 8888 or tcp.dstport == 7777
No. Time Source Destination Protocol Leng Info
233 98.462380209 127.0.0.1 127.0.0.1 TCP 74 41354 → 7777
235 98.462402250 127.0.0.1 127.0.0.1 TCP 66 41354 → 7777
236 98.462450445 127.0.0.1 127.0.0.1 TCP 79 41354 → 7777
238 98.478287385 127.0.0.1 127.0.0.1 TCP 74 54038 → 8888
240 98.478317247 127.0.0.1 127.0.0.1 TCP 66 54038 → 8888
242 98.478497086 127.0.0.1 127.0.0.1 TCP 54 56170 → 8888
244 98.937169251 127.0.0.1 127.0.0.1 TCP 71 41354 → 7777
246 99.180274345 127.0.0.1 127.0.0.1 TCP 79 54038 → 8888
248 99.293185758 127.0.0.1 127.0.0.1 TCP 71 54038 → 8888
250 99.712917065 127.0.0.1 127.0.0.1 TCP 77 54038 → 8888
252 99.860386466 127.0.0.1 127.0.0.1 TCP 77 54038 → 8888
254 99.884702123 127.0.0.1 127.0.0.1 TCP 79 41354 → 7777
260 100.140151140 127.0.0.1 127.0.0.1 TCP 73 54038 → 8888
266 100.627377037 127.0.0.1 127.0.0.1 TCP 76 41354 → 7777
268 100.688636609 127.0.0.1 127.0.0.1 TCP 71 41354 → 7777
270 100.750384475 127.0.0.1 127.0.0.1 TCP 77 41354 → 7777
272 100.868517239 127.0.0.1 127.0.0.1 TCP 79 54038 → 8888
274 100.946695173 127.0.0.1 127.0.0.1 TCP 76 54038 → 8888
276 101.326633019 127.0.0.1 127.0.0.1 TCP 71 54038 → 8888
278 101.409443395 127.0.0.1 127.0.0.1 TCP 79 41354 → 7777
280 101.454233759 127.0.0.1 127.0.0.1 TCP 77 54038 → 8888
282 101.585183038 127.0.0.1 127.0.0.1 TCP 79 54038 → 8888
293 101.932278700 127.0.0.1 127.0.0.1 TCP 77 41354 → 7777
295 102.351926834 127.0.0.1 127.0.0.1 TCP 80 54038 → 8888
297 102.600687319 127.0.0.1 127.0.0.1 TCP 77 54038 → 8888
299 102.754431932 127.0.0.1 127.0.0.1 TCP 78 41354 → 7777
301 102.816051760 127.0.0.1 127.0.0.1 TCP 76 41354 → 7777
303 102.881615691 127.0.0.1 127.0.0.1 TCP 71 41354 → 7777
305 103.160686993 127.0.0.1 127.0.0.1 TCP 76 41354 → 7777
307 103.376167861 127.0.0.1 127.0.0.1 TCP 75 54038 → 8888
309 103.892592860 127.0.0.1 127.0.0.1 TCP 71 41354 → 7777
311 104.137698932 127.0.0.1 127.0.0.1 TCP 78 54038 → 8888
313 104.552599772 127.0.0.1 127.0.0.1 TCP 76 41354 → 7777
315 104.641199273 127.0.0.1 127.0.0.1 TCP 72 41354 → 7777
317 104.886284549 127.0.0.1 127.0.0.1 TCP 76 54038 → 8888
319 104.943695436 127.0.0.1 127.0.0.1 TCP 71 54038 → 8888
329 105.378803851 127.0.0.1 127.0.0.1 TCP 77 41354 → 7777
331 105.700860923 127.0.0.1 127.0.0.1 TCP 76 54038 → 8888
333 105.723451677 127.0.0.1 127.0.0.1 TCP 71 54038 → 8888
335 106.306422430 127.0.0.1 127.0.0.1 TCP 80 41354 → 7777
337 106.502877535 127.0.0.1 127.0.0.1 TCP 76 54038 → 8888
339 106.853632879 127.0.0.1 127.0.0.1 TCP 72 54038 → 8888
341 107.004410965 127.0.0.1 127.0.0.1 TCP 79 41354 → 7777
343 107.357328580 127.0.0.1 127.0.0.1 TCP 80 41354 → 7777
345 107.543342867 127.0.0.1 127.0.0.1 TCP 77 54038 → 8888

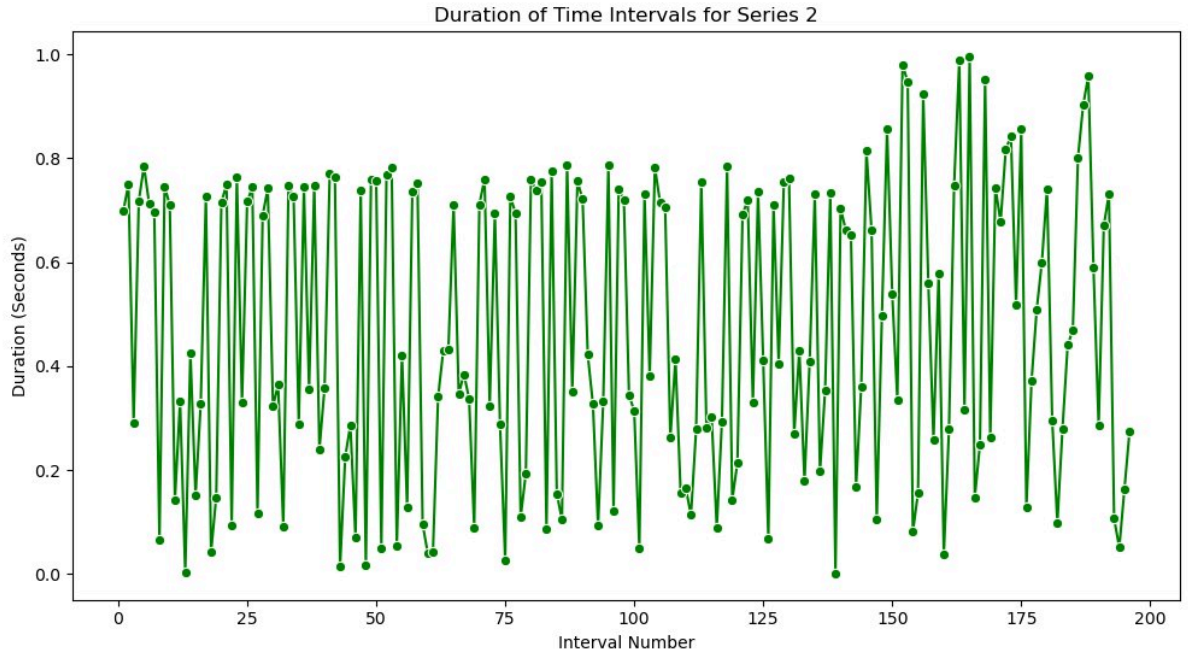
Задержка пакетов на закладке



Задержка пакетов на принимающей стороне




Duration of Time Intervals for Series 1
Duration (Seconds)
Interval Number
0.0
0.2
0.4
0.6
0.8
1.0
0
25
50
75
100
125
150
175


Duration of Time Intervals for Series 2
Duration (Seconds)
Interval Number
0.0
0.2
0.4
0.6
0.8
1.0
0
25
50
75
100
125
150
175
200

Демонстрация работы злоумышленника

```
413574, 0.7155959606170654, 0.4266533851623535, 0.7445380687713623, 0.7
677059173583984, 0.16148948669433594, 0.03881216049194336, 0.3609688282
0129395, 0.26783061027526855, 0.7463560104370117, 0.39174699783325195,
0.33321475982666016, 0.7039711475372314, 0.10402917861938477, 0.0837481
0218811035, 3.719329833984375e-05, 0.9298322200775146, 0.74792289733886
72, 0.792382001876831, 0.04205799102783203, 0.05711960792541504, 0.0549
2830276489258, 0.40166234970092773, 0.9581282138824463, 0.8423011302947
998, 0.9180877208709717, 0.720597505569458, 0.9336626529693604, 0.18301
057815551758, 0.49352431297302246, 0.04000735282897949, 0.7474784851074
219, 0.9871957302093506, 0.3458425998687744, 0.7870693206787109, 0.4049
344062805176, 0.14796233177185059, 0.12375855445861816, 0.6471223831176
758, 0.8337888717651367, 0.3461759090423584, 0.45174145698547363, 0.741
0516738891602, 0.733468770980835, 0.055536508560180664, 0.6226224899291
992, 0.1255474090576172, 0.1497974395751953, 0.605034351348877, 0.61773
34785461426, 0.5267014503479004, 0.5150542259216309, 0.9310648441314697
, 0.4831123352050781, 0.4852721691131592, 0.06131243705749512, 0.063564
53895568848, 0.12887811660766602, 0.7657132148742676, 0.515369653701782
2, 0.5218112468719482, 0.9163057804107666, 0.4704091548919678, 0.824307
2032928467, 0.6482932567596436, 0.4302225112915039, 0.6230626106262207,
 0.5633816719055176, 0.9282772541046143, 0.3579387664794922, 0.85307216
64428711, 0.49486804008483887, 0.8364005088806152, 0.48743462562561035,
 0.027339935302734375, 0.7213549613952637, 0.005144596099853516, 0.7663
300037384033, 0.8299777507781982, 0.5278987884521484, 0.253913164138793
95, 0.19715547561645508, 0.2476813793182373, 0.659451961517334, 0.36337
49485015869, 0.29915571212768555, 0.7518026828765869, 4.620675086975098
]
ERROR:root:Recovered message: b'\x03#\xa7v\xf0|\xb3\x16\xbe\x1cHid this
, check out!'
```