**Project Report**

**Title: SYN Flood Mitigation System Using Python**

**Group Members:**

**Ibrar Ul Hassan Shami (UW-23-CY-BS-018)**

**Muhammad Azfar Waqas (UW-23-CY-BS-013)**

**Saad Ali (UW-23-CY-BS-050)**

Table of Contents

[**1.** **Introduction:** 3](#_Toc188222733)

[**1.1** **Purpose:** 3](#_Toc188222734)

[**1.2** **Code Structure:** 3](#_Toc188222735)

[**1.3** **Targeted Operating Systems:** 4](#_Toc188222736)

[**1.4** **Compatibility with other Systems:** 5](#_Toc188222737)

[**1.5** **Targeted Hardware Platforms:** 5](#_Toc188222738)

[**2.** **Overall Description:** 6](#_Toc188222739)

[**2.1** **Manageability:** 6](#_Toc188222740)

[**2.2** **User Classes and Characteristics:** 7](#_Toc188222741)

[**2.3** **Design and Implementation Constraints:** 9](#_Toc188222742)

[**3.** **Non-Functional Requirement:** 10](#_Toc188222743)

[**3.1** **Performance Element:** 10](#_Toc188222744)

[**3.2** **Security Requirements:** 12](#_Toc188222745)

[**3.3** **Software Quality Attributes:** 13](#_Toc188222746)

[**4.** **Other Requirements:** 14](#_Toc188222747)

[**4.1** **Hardware Interface:** 14](#_Toc188222748)

[**4.2** **Communication Interface:** 15](#_Toc188222749)

[**5.** **Execution:** 15](#_Toc188222750)

# **Introduction:**

Syn Flood attacks are a form a **Denial-of-Service** (DoS) attack that exploit the **TCP Handshake** mechanism, overwhelming the server’s resources with a huge amount of traffic (SYN packets) leading to critical time delays and unavailability of the server for legitimate clients.

## **Purpose:**

The purpose of our **SYN Flood Mitigation System** is to detect and mitigate **SYN flood attacks**. The system aims to:

* Detect Malicious Activity
* Block Malicious IP Address
* Prevent System Disturbances
* Enhance Network Security
* Real-Time Monitoring and Visualization
* Alert System
* Automated Response
* Rate Limiting

## **Code Structure:**

The System’s code is designed as follows:

1. **Libraries:**

* **scapy:** Packet sniffing and analysis.
* **datetime:** Record time differences.
* **collections:** Use a special dictionary.
* **pandas:** Log attacks & blocked IP addresses.
* **matplotlib:** Visualize SYN packets analysis graph.
* **threading:** Enable multitasking.
* **smtplib:** Emailing services.
* **email.mime.multipart:** Formatting email.
* **email.mime.text:** Formatting body of email.
* **subprocesses:** Allows the execution of system commands.
* **os:** Access OS level functionality

1. **Functions:**

* **send\_email\_alert:** Sends email using the **TLS** protocol via smtplib, email.mime.multipart and email.mime.text using the port **587**.
* **log\_attack:** Logs the attack using the datetime and pandas Dataframe into a csv file by making 4 columns which include **timestamps**, **Ip**, **count** & **additional info.**
* **is\_rule\_exists:** Checks if an IP address is blocked within the firewall using **subprocesses** & **netsh** command.
* **block\_ip:** Uses the netsh firewall command to block both inbound and outbound traffic due to a specific IP address.
* **kill\_connections:** Kill’s connection to the server on a specific port of a specific IP Address using **PID.**
* **detect\_syn:** Checks, detects and manages all of the SYN packets coming towards the server on the specified interface.
* **get\_network\_interface:** Detects all available interfaces on the device.
* **start\_sniffing:** Starting the sniffing process on the **selected interface** via scapy.
* **update\_plot:** Visualizes the **Real-Time SYN Packet Analysis** Graph via matplotlib.

## **Targeted Operating Systems:**

The targeted operating system for your **SYN Flood Mitigation System** is primarily **Windows**, based on the following:

* **Firewall Commands:** The use of netsh commands is specific to Windows' advanced firewall management.
* **Process Management:** The program uses task kill to terminate processes associated with malicious connections using netsh.
* **Python Dependencies:** Functions like os.system and subprocess.run are used to execute Windows-specific commands.
* **Graphical Backend:** The matplotlib library uses tkinter for plotting, which works cross-platform but requires proper configuration, especially for Windows systems.

## **Compatibility with other Systems:**

The system could potentially be adapted for other operating systems like **Linux** or **macOS**, but changes would be required, such as:

* Replacing netsh with iptables or ufw for Linux firewall management.
* Substituting taskkill with kill commands.
* Adjusting dependencies to match the target OS's networking and process management capabilities.

## **Targeted Hardware Platforms:**

The **target hardware platforms** for the **SYN Flood Mitigation System** primarily include systems capable of running Python, interacting with network interfaces, and managing the computational and memory requirements for real-time network traffic analysis.

1. **Personal Computers:**

Typical desktops/laptops with at least **4-8 GB of RAM** and a **modern processor (e.g., Intel Core i5/i7 or AMD equivalent)** can handle the program efficiently.

1. **Servers:**

Servers with **16-32 GB of RAM** or more can handle large-scale traffic analysis.

1. **Embedded Systems:**

For lightweight or experimental deployments, Single Board Computers (SBCs) like Raspberry Pi 4 (with at least 4 GB of RAM) can run the program for smaller network environments.

1. **Virtual Machines:**

The program can run on virtualized environments for testing or deployment purposes. Assigning at least **2 CPU cores**, **4 GB of RAM**, and **network bridge configurations** is recommended for efficient operation.

1. **Specializes Hardware:**

Hardware optimized for packet capture, like **Dell PowerEdge R-series servers** or similar, can handle heavy loads of traffic with minimal latency.

# **Overall Description:**

## **Manageability:**

Maintainability, readability, flexibility, and robustness are essential aspects of code quality. The design and implementation of the SYN Flood Mitigation system emphasize these principles to ensure it remains easy to understand, modify, and extend. Below are the key aspects that contribute to these qualities in the code:

* **Modular Design:**
* **Contribution:** The code is divided into modular components, such as packet detection (detect\_syn), IP rate limiting (rate\_limit\_ip), attack logging (log\_attack), and real-time plotting (update\_plot). Each module focuses on a specific functionality, allowing developers to maintain and update individual components without affecting the entire system.
* **Descriptive Variable and Function Name:**
* **Contribution:** Functions like block\_ip, kill\_connections, and send\_email\_alert and variable names like syn\_timestamps, malicious\_syn\_packet\_counts are named to reflect their purpose. This enhances the code's clarity, making it easier for developers to understand the purpose and behavior of each part of the system.
* **Comments and Documentation:**
* **Contribution:** Inline comments and detailed docstrings for functions provide essential context, explaining their functionality and logic. This documentation helps current and future developers understand the purpose of the code and how to extend or modify it.
* **Consistent Coding Style:**
* **Contribution:** The code follows a consistent style, including proper indentation, clear separation of logical blocks, and adherence to Python conventions (e.g., PEP 8). This consistency ensures that the codebase is cohesive and easier to read and review.
* **Code Reusability:**
* **Contribution:** Functions like rate\_limit\_ip and log\_attack are designed to be reusable across different parts of the system. This minimizes redundancy and allows components to be leveraged in other scenarios
* **Unit Testing:**
* **Contribution:** The design facilitates the inclusion of unit tests for individual components, such as testing the rate-limiting logic, logging functionality, or plotting mechanism. Testing ensures that each module behaves as expected, contributing to the robustness of the system and preventing regressions during future updates.

## **User Classes and Characteristics:**

In the **SYN Flood Mitigation System**, different user groups interact with the system based on their roles and responsibilities. The expected level of technical expertise varies among these user types. Understanding these user classes ensures the system is designed to meet the needs of all stakeholders effectively, including user interfaces, training, and documentation.

* **Network Administrator:**
* **Technical Expertise**: High
* **Interaction with Code**: Extensive
* **Responsibilities:**
* Monitoring real-time SYN flood activity using visualizations.
* Configuring thresholds for rate limiting.
* Managing and enforcing firewall rules and IP blocking mechanisms.
* Debugging and updating the system to address emerging threats.
* **Developers/Programmers:**
* **Technical Expertise:** High
* **Interaction with Code**: Extensive
* **Responsibilities**:
* Developing and extending functionalities such as detection algorithms or reporting mechanisms.
* Conducting maintenance and unit testing.
* Integrating the system with existing network infrastructure or attack response tools.
* **IT Security Analyst:**
* **Technical Expertise**: High
* **Interaction with Code**: Moderate
* **Responsibilities**:
* Analyzing logs and reports to identify attack patterns.
* Investigating flagged malicious activity.
* Providing recommendations to system administrators for tuning parameters to improve detection.
* **Management/Decision Makers:**
* **Technical Expertise**: Low to Moderate
* **Interaction with Code**: Limited
* **Responsibilities**:
* Reviewing high-level reports generated by the system.
* Making decisions about infrastructure upgrades or security investments.
* Delegating tasks to relevant teams for response and improvement.
* **General Users: (Non-Technical Staff)**
* **Technical Expertise:** Low
* **Interaction with Code:** Minimal to None
* **Responsibilities:**
* May receive alerts from the system but have no direct interaction with the code.
* Escalate critical alerts to system administrators or security analysts.

## **Design and Implementation Constraints:**

The design and implementation of the SYN Flood Mitigation System were shaped by several constraints and requirements to ensure the system functions effectively, meets the needs of users, and adheres to best practices in network security. Below are the key constraints that influenced the design:

* **Regulatory Compliance**  
  **Requirement:** Compliance with network security standards and data protection regulations (e.g., GDPR, NIST guidelines).  
  **Impact:** The system design ensures secure handling of logs, proper encryption of data where applicable, and adherence to legal requirements for monitoring and mitigation.
* **Performance Requirements**  
  **Requirement:** The ability to handle high volumes of network traffic without degrading system performance.  
  **Impact:** The system employs multithreading to efficiently monitor traffic in real time and mitigate attacks without causing latency for legitimate users.
* **Data Security and Privacy**  
  **Requirement:** Prevent unauthorized access to logs, configurations, and mitigation processes.  
  **Impact:** The system implements role-based access and ensures sensitive information like IP logs is stored securely.
* **Availability and Reliability**  
  **Requirement:** The system must remain operational even during heavy traffic loads or attacks.  
  **Impact:** Robust design using modular code ensures high reliability, allowing the system to continue functioning even during partial failures.
* **Network Environment Constraints**  
  **Requirement:** The system must work within diverse network setups, such as public clouds, private networks, or hybrid architectures.  
  **Impact:** Flexible configuration options and customizable thresholds allow it to adapt to varying environments.
* **Resource Limitations**  
  **Requirement:** Efficient resource utilization to ensure that the system runs on commodity hardware without excessive resource consumption.  
  **Impact:** The use of lightweight libraries like Scapy and Matplotlib ensures minimal CPU and memory usage.
* **Usability and Accessibility**  
  **Requirement:** The system should be easy to use for network administrators and security analysts.  
  **Impact:** A user-friendly interface with real-time visualizations and clear error messages simplifies interaction for non-developers.
* **Scalability and Flexibility**  
  **Requirement:** The system should be scalable to handle increased traffic and adaptable to new attack patterns.  
  **Impact:** Modular design and configurable parameters allow easy scaling and integration of additional mitigation strategies.

# **Non-Functional Requirement:**

## **Performance Element:**

The performance requirements for the SYN Flood Mitigation System focus on ensuring high efficiency and robustness during operation, even under heavy network traffic. Below are the key performance benchmarks and targets:

* **Detection Response Time:**
* **Benchmark:** The time taken to detect and flag potential SYN flood attacks.
* **Target:** Detection should occur in near real-time (within milliseconds) to allow for immediate mitigation.
* **Mitigation Response Time:**
* **Benchmark:** The time taken for the system to apply mitigation measures (e.g., rate-limiting, IP blocking) once an attack is detected.
* **Target:** Mitigation should be deployed within one second of detecting an attack to minimize damage and disruption.
* **Concurrent Traffic Handling:**
* **Benchmark:** The maximum amount of network traffic the system can process simultaneously without performance degradation.
* **Target:** The system should be capable of handling traffic loads of up to 10,000 packets per second (or a configurable threshold) while maintaining performance.
* **Memory Usage:**
* **Benchmark:** The amount of system memory consumed during normal and peak operation.
* **Target:** Memory usage should be optimized to ensure the system remains stable, with usage not exceeding 70% of available memory during peak traffic.
* **CPU Utilization:**
* **Benchmark:** The percentage of CPU resources consumed during operation.
* **Target:** CPU usage should remain below 80% under normal conditions and below 90% during heavy traffic.
* **Packet Analysis Throughput:**
* Benchmark: The rate at which the system can analyze and classify incoming packets.
* Target: The system should process at least 95% of incoming packets within the defined detection window, ensuring minimal packet loss.

## **Security Requirements:**

Security is paramount in ensuring the SYN Flood Mitigation System operates effectively without introducing vulnerabilities. The following measures define how the system safeguards its functionality:

* **Firewall and Network Protection**:
* The system should operate in a secure network environment with firewalls to prevent unauthorized external access.
* Access to the mitigation scripts and tools should be restricted to trusted IPs or internal networks.
* **Regular Updates:**
* The system should support regular updates to address security vulnerabilities and include automatic patching for known issues.
* **Environmental Variables:**
* The system should always have environmental variables sets for better security.

## **Software Quality Attributes:**

The SYN Flood Mitigation System adheres to several software quality attributes to ensure reliability, maintainability, and performance:

* **Reliability**:
  + The system must operate continuously without failures, detecting and mitigating SYN flood attacks effectively.
  + Redundant mechanisms should ensure uninterrupted operation even in case of hardware or software issues.
* **Maintainability**:
  + The system code is modular, with clear documentation, allowing for easy updates and modifications.
  + Consistent naming conventions and in-line comments improve code readability for developers.
* **Performance**:
  + The system must handle high volumes of network traffic efficiently, ensuring no significant delays in detection or mitigation processes.
* **Scalability**:
  + The architecture should support scaling to accommodate larger networks or additional monitoring interfaces without performance degradation.
* **Portability**:
  + The system must operate across different operating systems (e.g., Linux, Windows) with minimal configuration changes.
* **User-Friendly Interface**:
  + Command-line operations and real-time visualization are intuitive and provide comprehensive insights into system performance and detected threats.
* **Error Handling**:
  + Comprehensive error handling prevents the system from crashing during unexpected scenarios, with meaningful error messages guiding corrective actions.
* **Extensibility**:
  + The system is designed to allow easy integration of additional features or enhancements, such as new mitigation techniques or monitoring protocols.

# **Other Requirements:**

## **Hardware Interface:**

The SYN Flood Mitigation System requires specific hardware interfaces to ensure proper functioning and compatibility:

* **Direct Hardware Interaction:**
* Certain implementations may involve interaction with hardware like packet-capturing devices, network switches, or specialized network adapters. Specific libraries or drivers may be required for these devices.
* **Network Interface:**
* The system uses network interfaces to capture packets and communicate with databases or other external systems. This interaction occurs over standard network protocols such as TCP/IP.

## **Communication Interface:**

Effective communication between system components and external entities is achieved through the following:

* **Network Protocols:**
* The system utilizes protocols like TCP/IP and TLS for secure and efficient data exchange with servers, clients, and other systems.
* **Message System:**
* The system utilizes the secures emailing system of google.com to convey alerts.

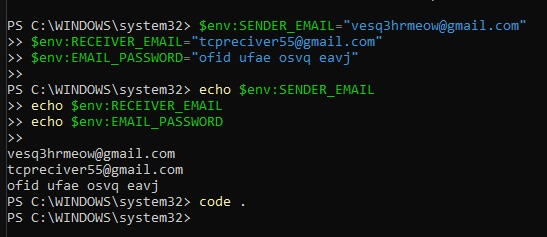
# **Execution:**

For the sake of simulation of the **SYN Flood Mitigation System** some other scrips were also made which include:

* TCP Server Script
* TCP Flood Script
* TCP Server Check Script

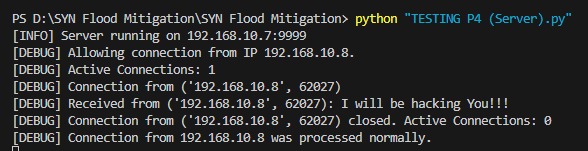
I’ll guide you through the steps to run the simulation (2 PCs required):

* Run the following commands in PowerShell terminal as administrator

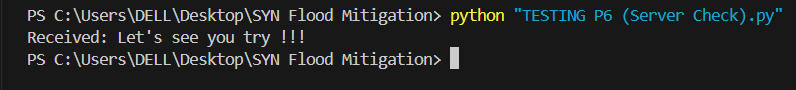


* Run the TCP Server Script and then run the Server Check Script, you will see the following outputs on the respective screens

**From Server:**

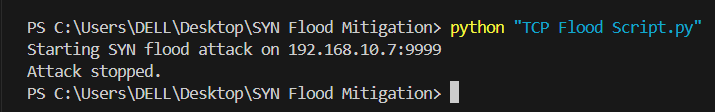


**From Server Check:**

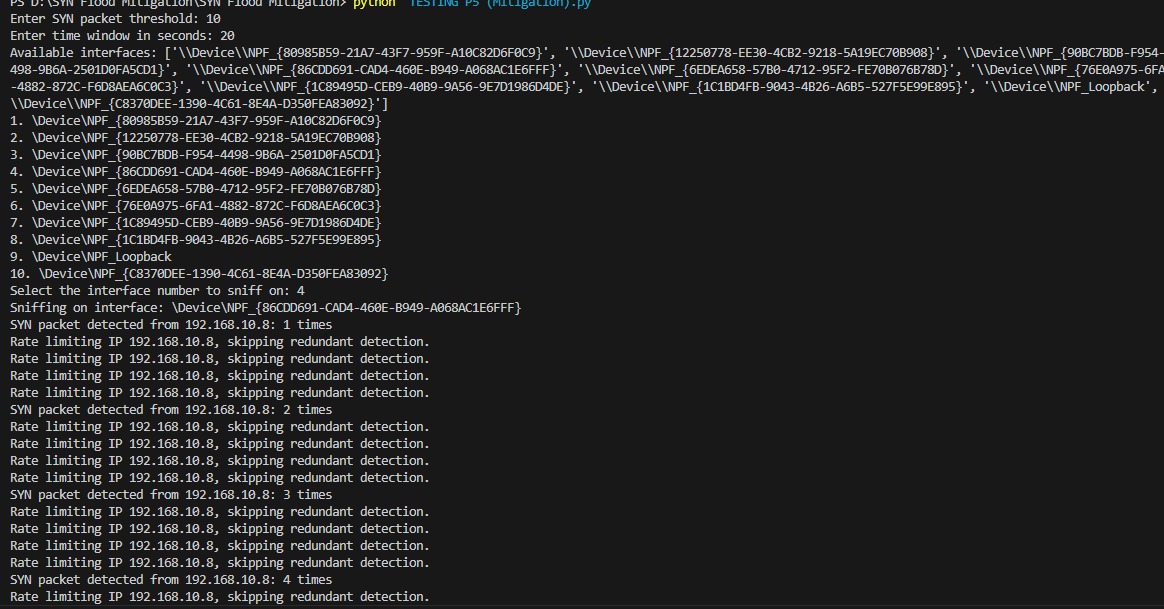
****

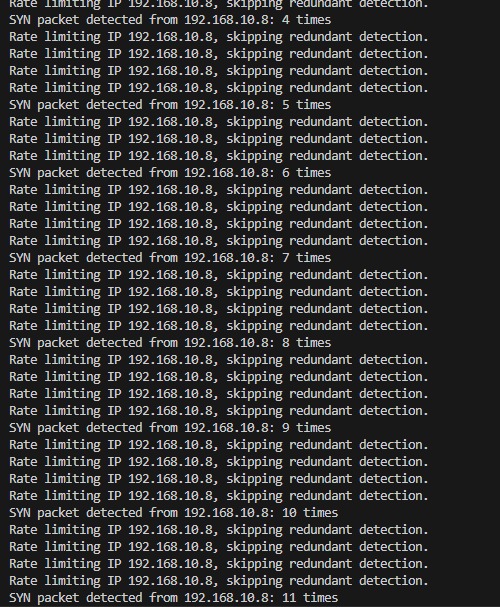
* Now run the SYN Flood Mitigation Script and input the necessary fields after that run the TCP Flood Script to launch an attack on the server

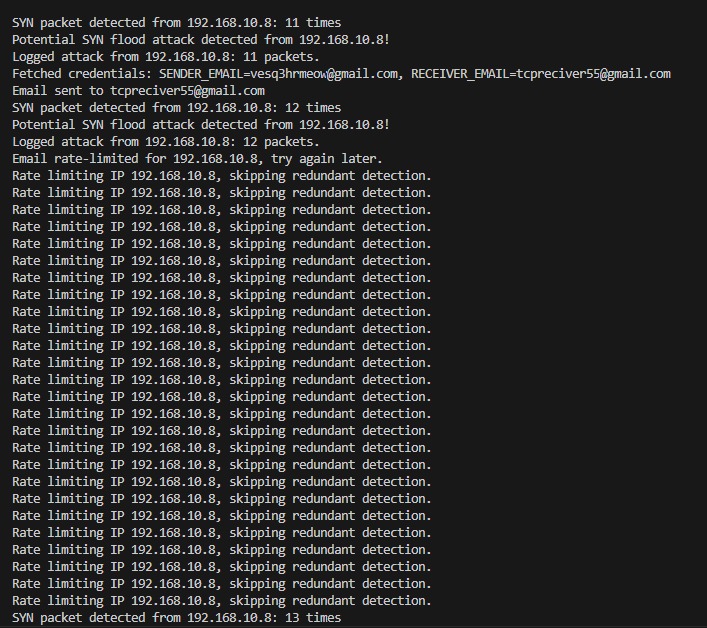
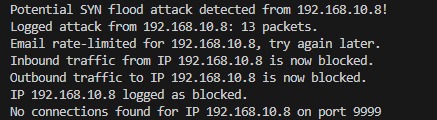
**From Flood:**

****

**From Mitigation Script:**





* Now if you were to check server again its output will be:

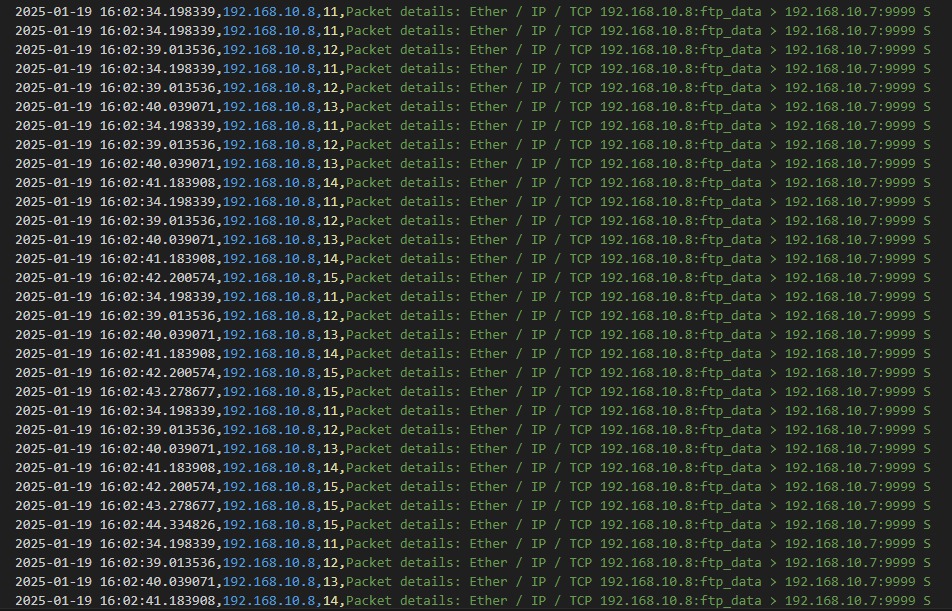


* Now observe your email, log attack file, blocked ips file and Graph to analyze the attack

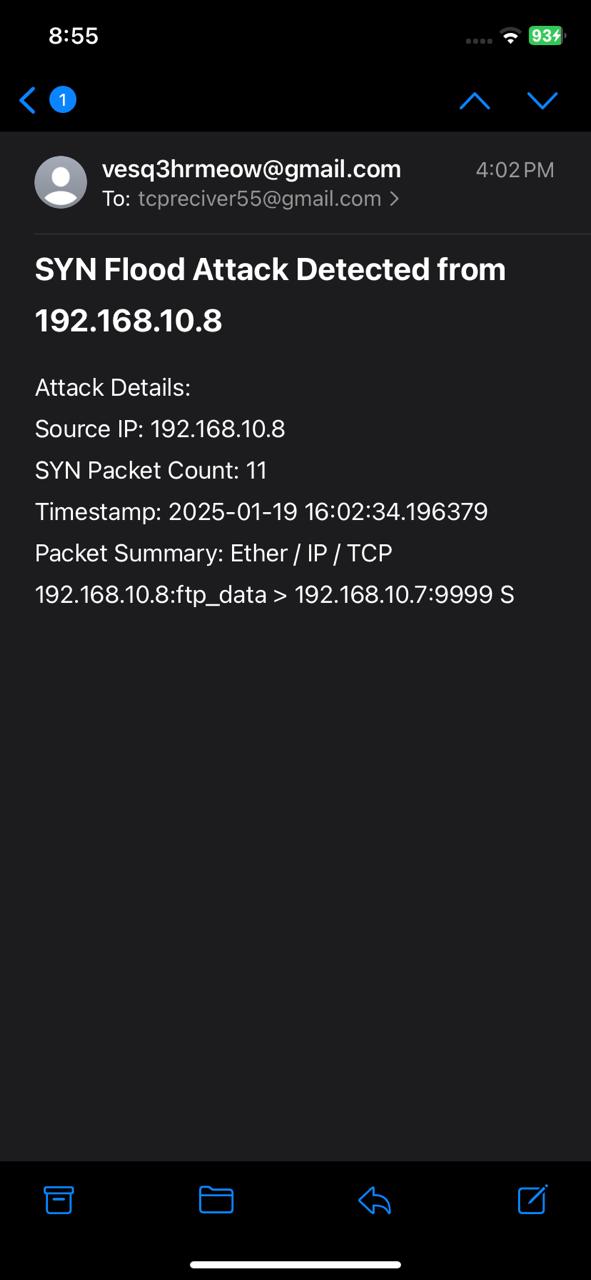
**From blocked\_ips\_log:**



**From attack\_log:**



**From Mail:**



**From Graph:**

