FINAL REPORT CSN PROJECT

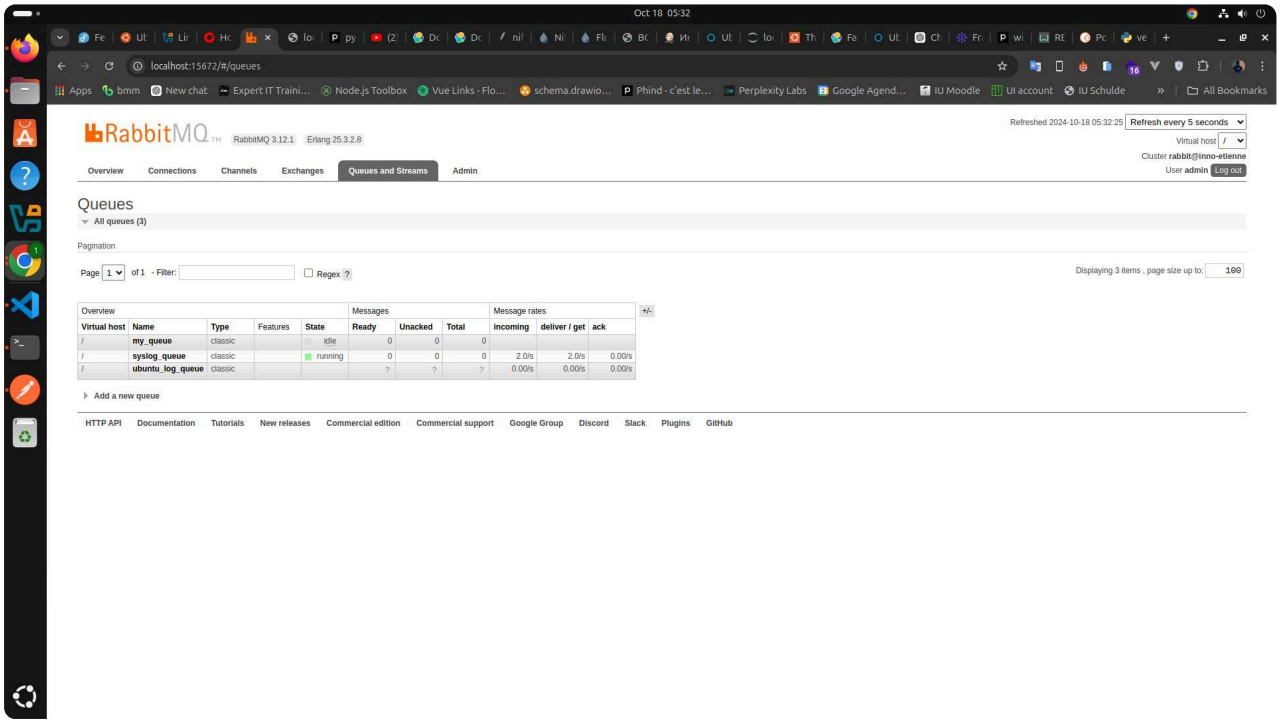
By Sossou PADONOU Etienne FUH Isaac WOMOAKOR

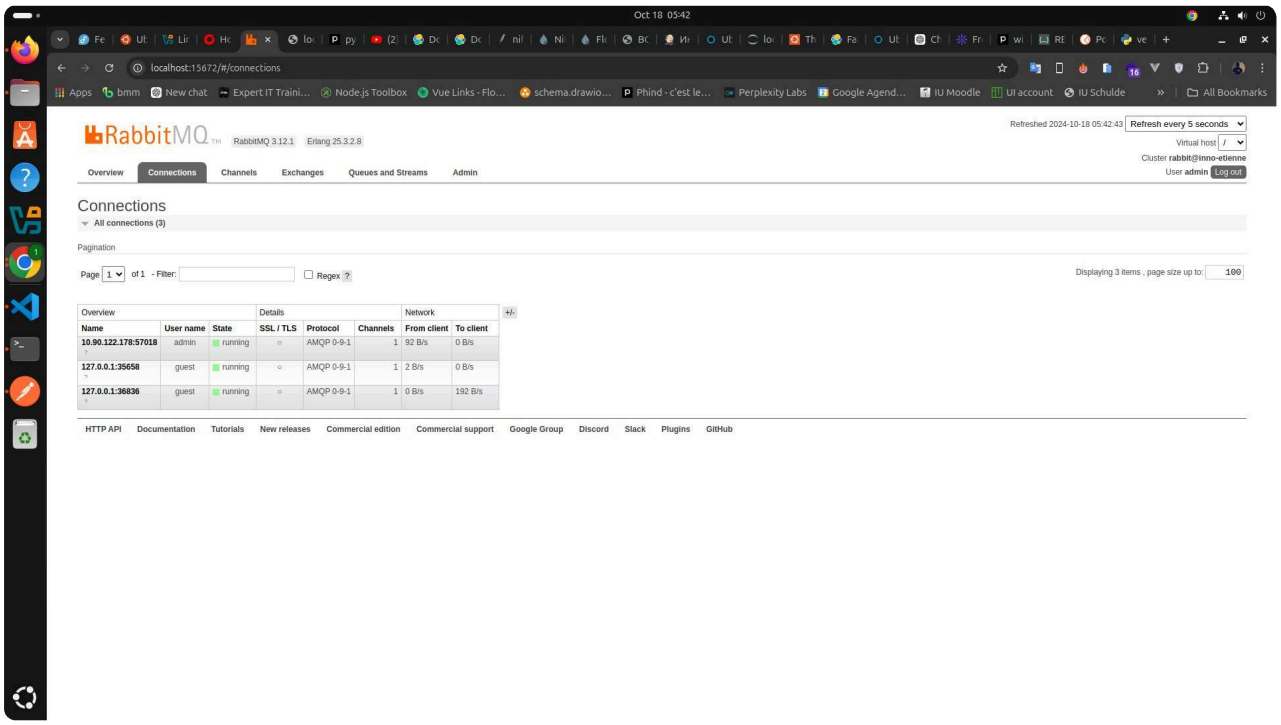
I- Introduction

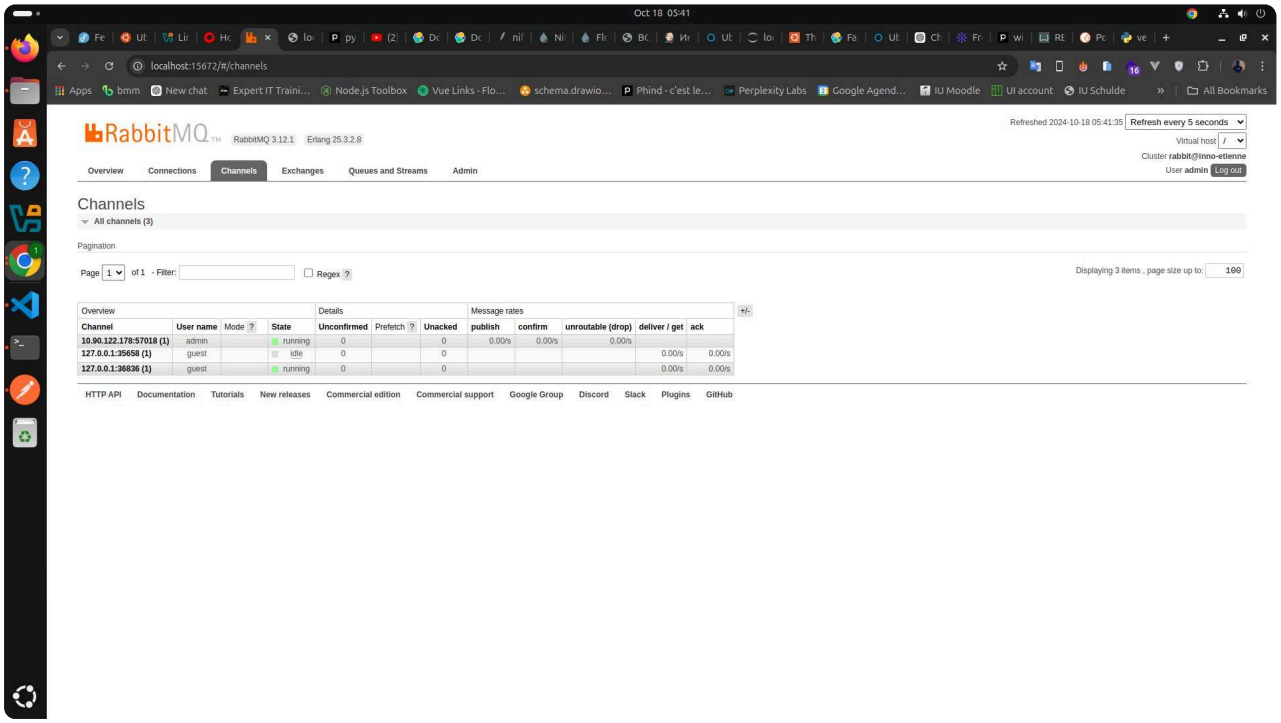
In a world where cybersecurity threats are continuously evolving, ensuring real-time monitoring and alerting systems for suspicious activity is essential for the protection of sensitive data. The RabbitMQ SecLog Alert project addresses this challenge by leveraging a combination of RabbitMQ for centralized log management, FreeIPA for authentication and logging, and Telegram for real-time notifications.
This system is designed to detect potential cybersecurity incidents, such as failed login attempts and unauthorized access, and send immediate alerts to system administrators through Telegram. This report outlines the system architecture, methods used for log filtering and incident detection, and the results of implementing this real-time alerting system.

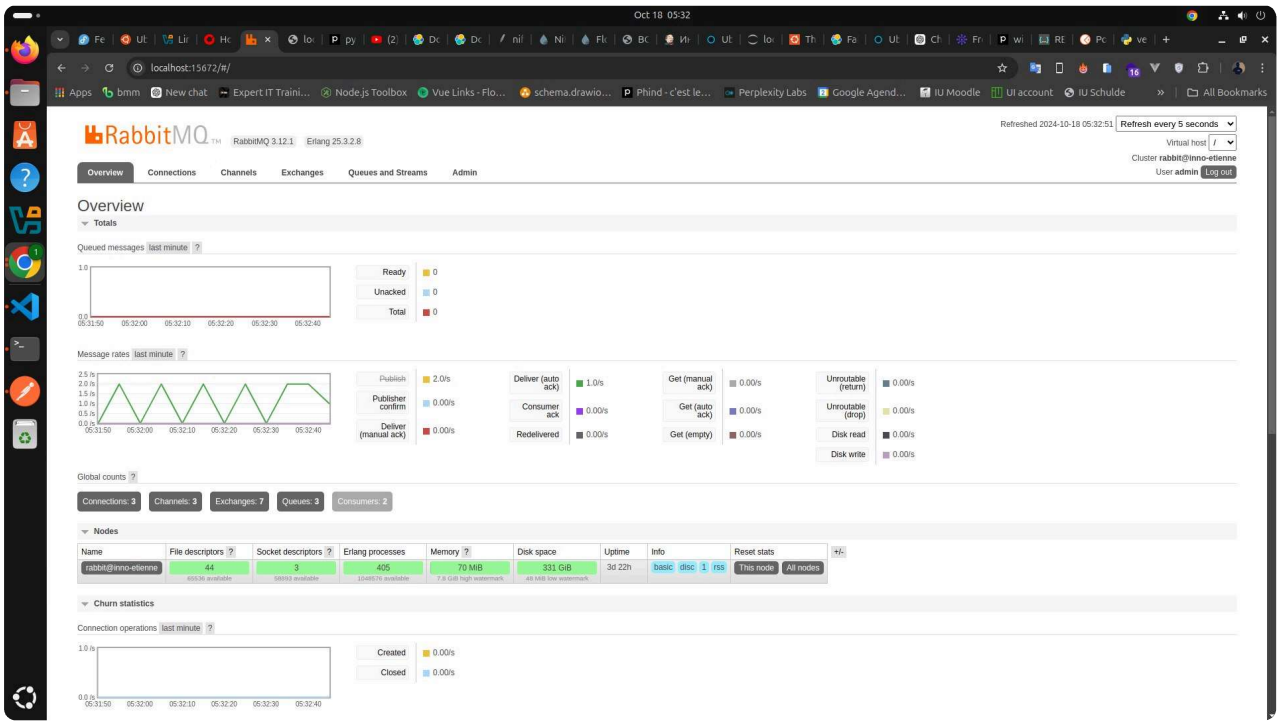
II- Methods

1. ***System Architecture***
   The system comprises four main components: FreeIPA, RabbitMQ, Log Filtering Machine, and Telegram API. Each component plays a critical role in ensuring that logs are collected, processed, and used for generating real-time alerts.
   * ***FreeIPA (Log Generation):*** FreeIPA is responsible for managing centralized authentication, policy enforcement, and secure access to the network. It generates logs based on user authentication attempts, capturing key events such as login successes, failures, and suspicious activities (e.g., repeated failed logins).
   * ***RabbitMQ (Message Queuing):***
   RabbitMQ acts as the message broker in the system. Logs generated by FreeIPA are sent to RabbitMQ, where they are queued and held for further processing. RabbitMQ decouples log generation from log processing, ensuring asynchronous transmission of logs to the Log Filtering Machine


Oct 18 05:32
localhost:15672/#/queues
RabbitMQ 3.12.1
Erlang 25.3.2.8
Overview
Connections
Channels
Exchanges
Queues and Streams
Admin
Queues
All queues (3)
Pagination
Page 1 of 1 - Filter:
Regex
Displaying 3 items , page size up to: 100
Overview
Messages
Message rates
Virtual host
Name
Type
Features
State
Ready
Unacked
Total
incoming
deliver / get
ack
/ my_queue classic idle 0 0 0
/ syslog_queue classic running 0 0 0 2.0/s 2.0/s 0.00/s
/ ubuntu_log_queue classic ? ? ? 0.00/s 0.00/s 0.00/s
Add a new queue
HTTP API
Documentation
Tutorials
New releases
Commercial edition
Commercial support
Google Group
Discord
Slack
Plugins
GitHub


Oct 18 05:42
localhost:15672/#/connections
RabbitMQ 3.12.1
Erlang 25.3.2.8
Overview
Connections
Channels
Exchanges
Queues and Streams
Admin
Connections
All connections (3)
Pagination
Page 1 of 1 - Filter:
Regex
Displaying 3 items , page size up to: 100
Overview
Details
Network
Name
User name
State
SSL / TLS
Protocol
Channels
From client
To client
10.90.122.178:57018 admin running AMQP 0-9-1 1 92 B/s 0 B/s
127.0.0.1:35658 guest running AMQP 0-9-1 1 2 B/s 0 B/s
127.0.0.1:36836 guest running AMQP 0-9-1 1 0 B/s 192 B/s
HTTP API
Documentation
Tutorials
New releases
Commercial edition
Commercial support
Google Group
Discord
Slack
Plugins
GitHub





* ***Log Filtering Machine (Incident Detection):***
The log filtering machine subscribes to the RabbitMQ queue to receive and analyze logs. The filtering system is designed to identify potential cybersecurity incidents by applying predefined algorithms. These algorithms search for specific patterns, such as repeated failed login attempts or unauthorized access, which indicate potential threats.

* ***Telegram API (Alert Notification):***
Once an incident is detected, the system sends a real-time alert to the system administrator via the Telegram API. A bot is created to handle the transmission of messages, ensuring that the administrator is immediately informed of any

suspicious activities.

4. ***Workflow of the System***

* ***Log Generation:***

FreeIPA generates logs for authentication events, including login attempts and user access.

* ***Log Queuing:***

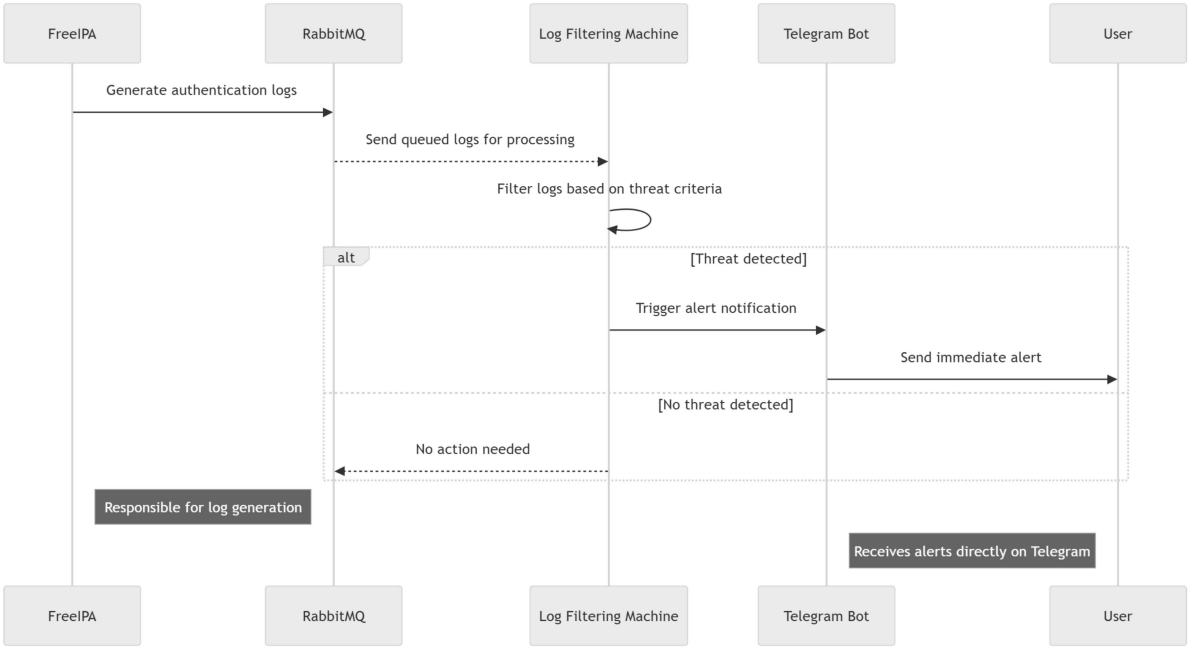
The generated logs are sent to RabbitMQ, where they are queued for processing.

* ***Log Filtering:***

The log filtering machine subscribes to RabbitMQ, pulls the logs, and filters them based on threat detection criteria.

* ***Alert Notification:***

If a threat is detected, the system triggers the Telegram bot to send an immediate alert to the administrator's Telegram account.



5. ***Incident Detection Algorithm***

The log filtering machine is equipped with an algorithm that detects three consecutive failed login attempts.

6. ***Implementation***

* ***Step 1:*** Integration of FreeIPA for Log Generation.

FreeIPA was installed on a server to manage authentication and policy enforcement.The server was configured to generate logs for all authentication-related events, including successful and failed login attempts, unauthorized access, and changes to user permissions. These logs were formatted to be

compatible with RabbitMQ and sent to RabbitMQ queues via a custom script.

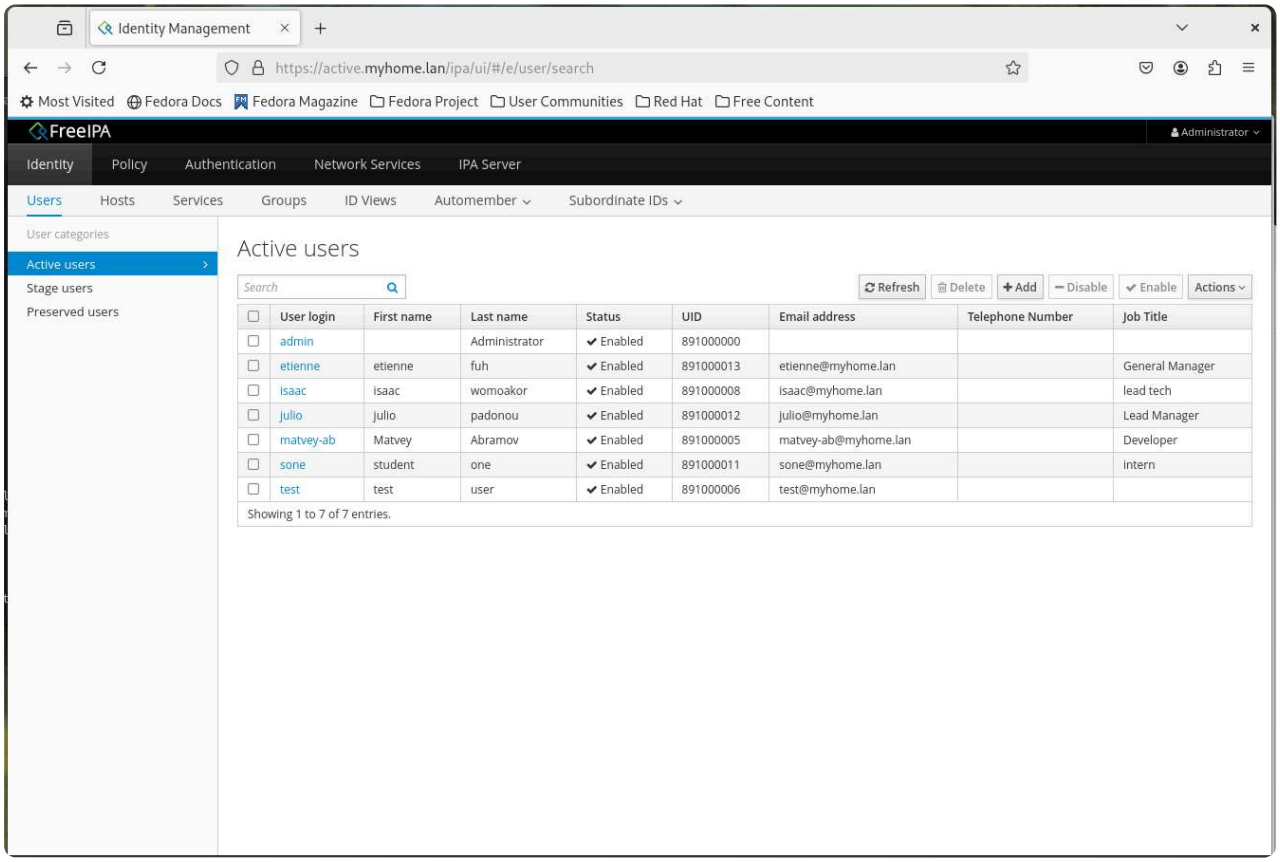


Figure 2 - FreeIPA GUI

While FreeIPA offers powerful identity management capabilities, its current logging limitations significantly hinder the ability to monitor and respond to account-related events in real-time. By implementing a comprehensive centralized logging solution, organizations can gain valuable insights into account activities, enhance security posture, and streamline administration tasks.

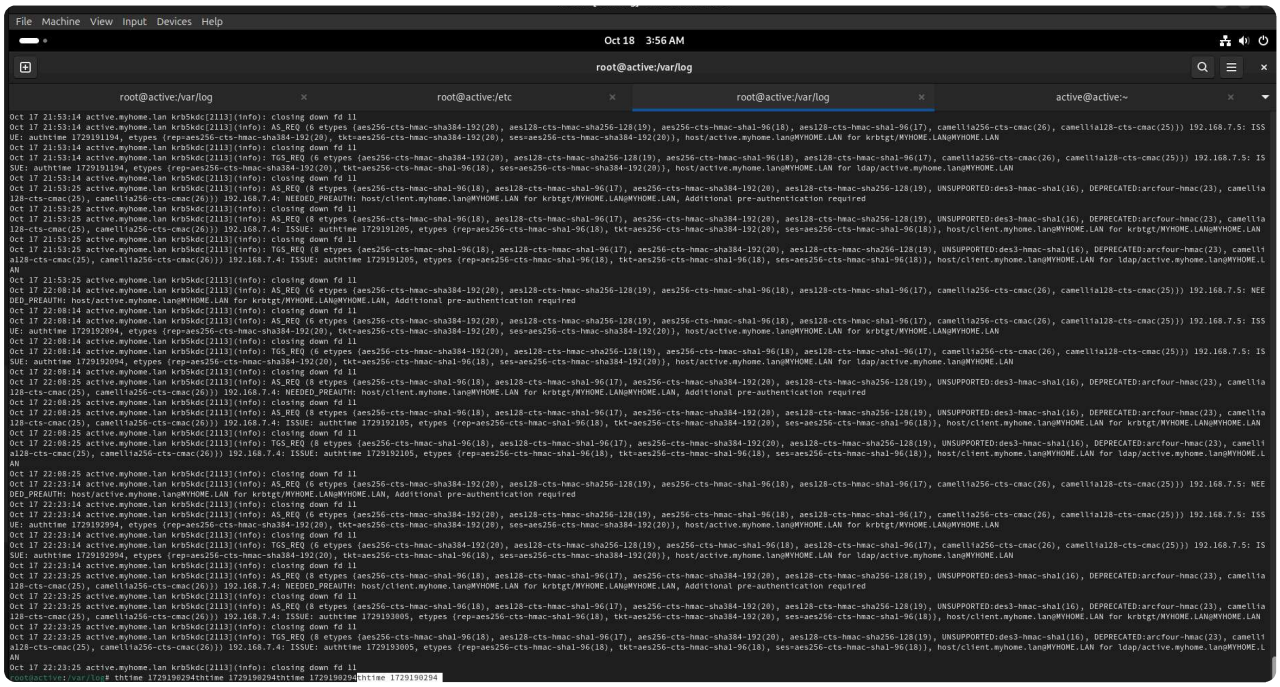


Figure 3- Logs from user accounts

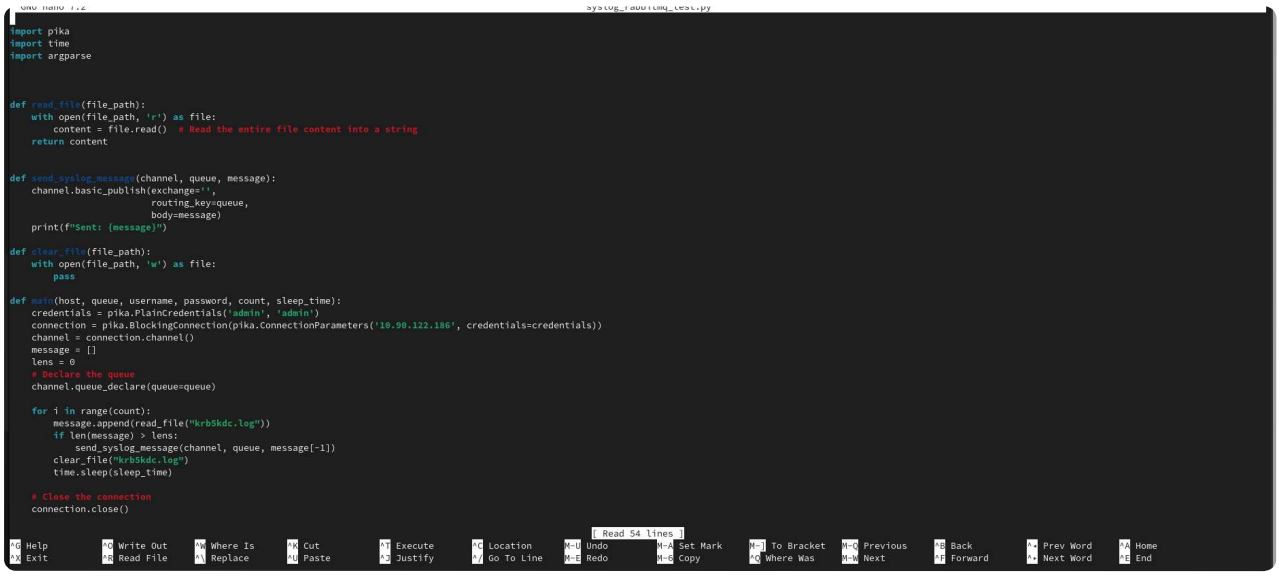
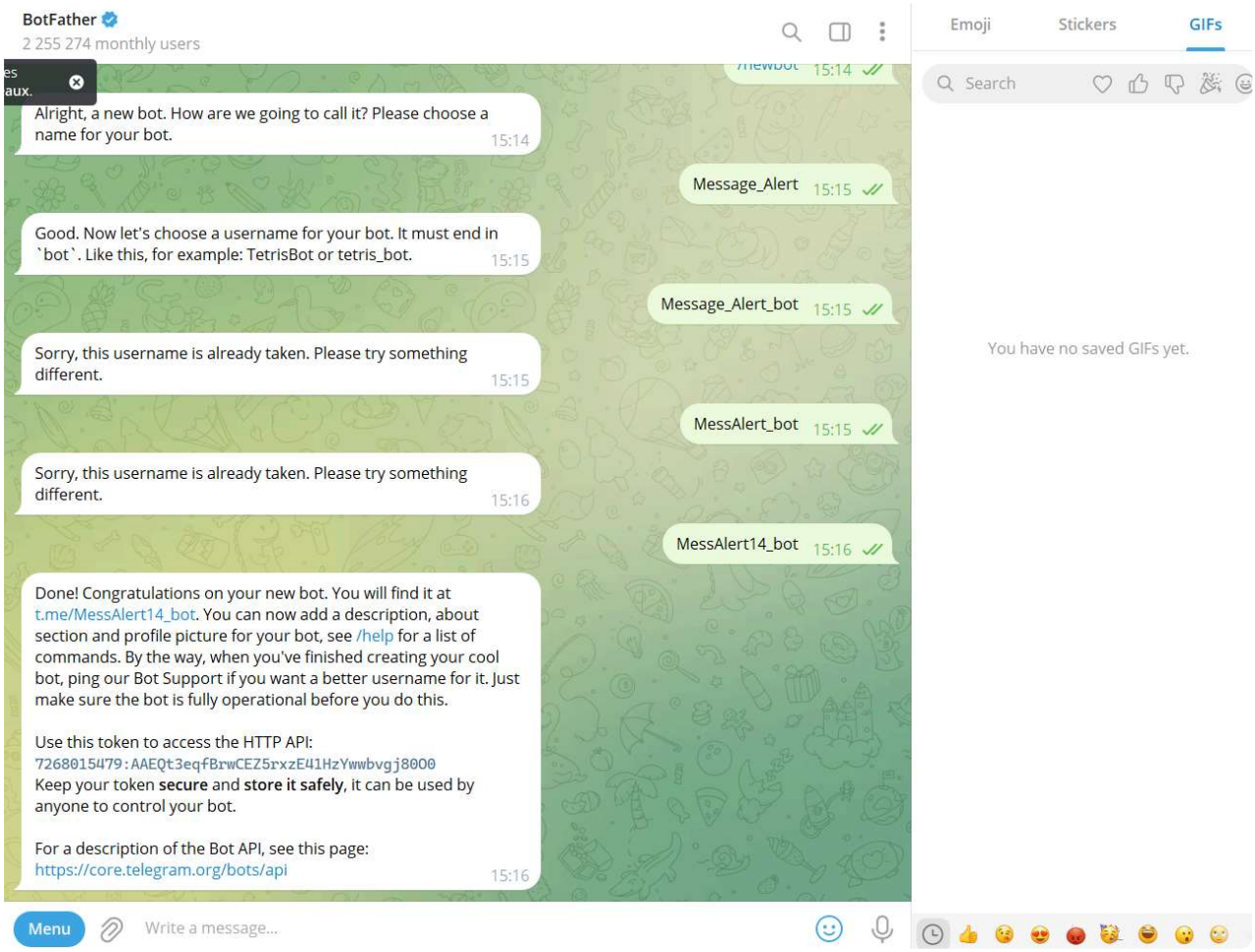


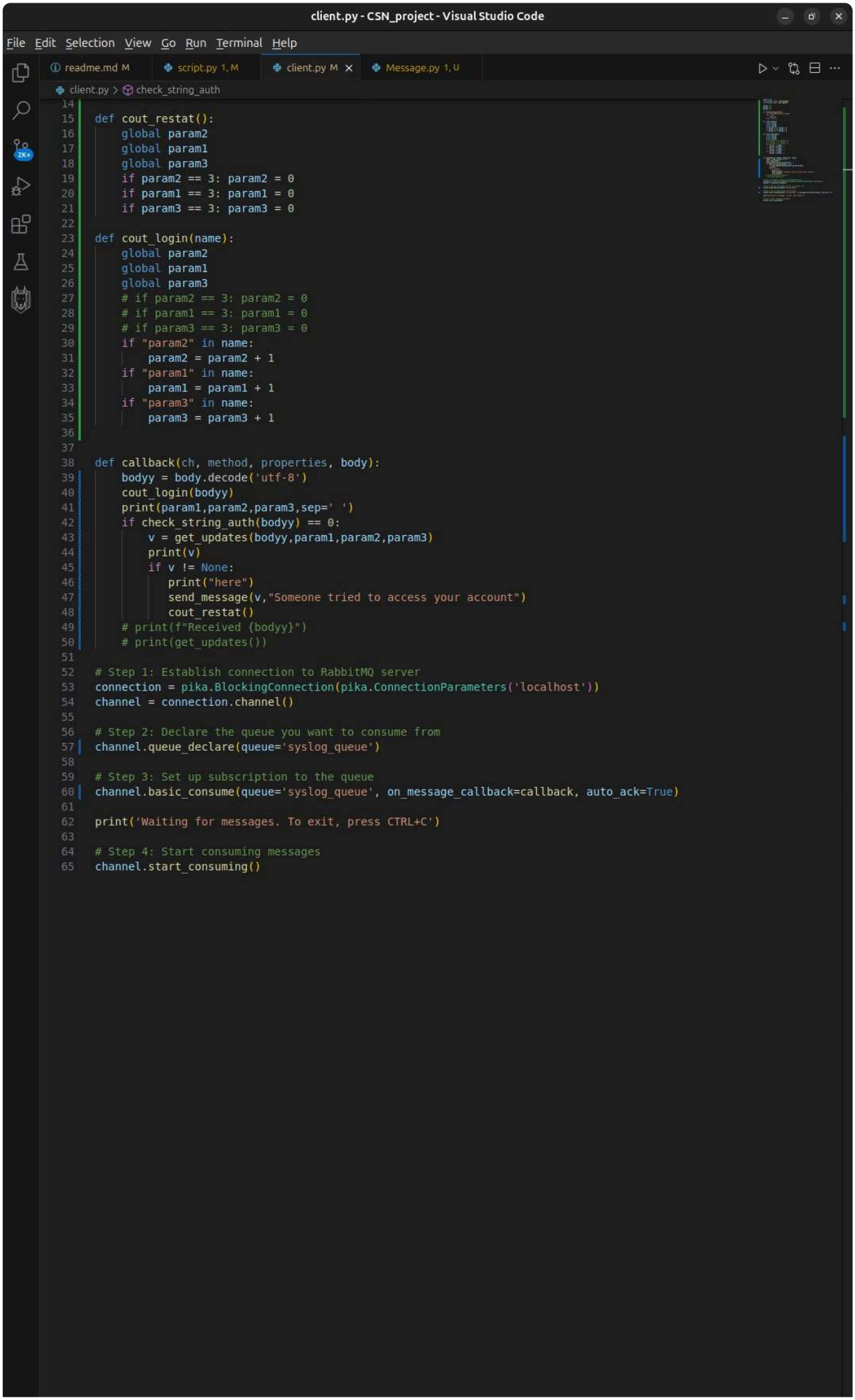
Figure 4- Script for log transfer

This log transfer script represents a significant improvement over FreeIPA's native logging capabilities. By centralizing log data in RabbitMQ, our implementation enables administrators and security teams to monitor account activities in real-time, enhancing the organization's ability to respond quickly to potential threats and maintain optimal system performance

* ***Step 2:*** Configuring RabbitMQ as the Message Broker
RabbitMQ was installed and configured to act as the central message broker. A specific queue was created to store FreeIPA logs. This queue enables asynchronous processing and decouples log generation from log filtering. FreeIPA logs were published to the RabbitMQ queue in real-time using the pika Python library. This allowed the system to reliably queue logs for processing.
* ***Step 3:*** Log Filtering and Cybersecurity Incident Detection
A log filtering machine was set up to subscribe to the RabbitMQ queue using the pika library. This machine pulls logs in real-time for analysis. A custom algorithm was implemented to filter logs and detect suspicious patterns. The algorithm focused on detecting common threat indicators, such as: Consecutive failed login attempts.
* ***Step 4:*** Implementing Telegram Bot for Real-Time Alerts
A Telegram bot was created using BotFather to serve as the alert mechanism.



The bot was integrated into the log filtering machine using the Telegram API. A Python script (requests library) was used to send real-time notifications to the system administrator's Telegram account. The alerts were customized to include key information such as the username involved.




```python
def cout_restat():
    global param2
    global param1
    global param3
    if param2 == 3: param2 = 0
    if param1 == 3: param1 = 0
    if param3 == 3: param3 = 0

def cout_login(name):
    global param2
    global param1
    global param3
    # if param2 == 3: param2 = 0
    # if param1 == 3: param1 = 0
    # if param3 == 3: param3 = 0
    if "param2" in name:
        param2 = param2 + 1
    if "param1" in name:
        param1 = param1 + 1
    if "param3" in name:
        param3 = param3 + 1


def callback(ch, method, properties, body):
    bodyy = body.decode('utf-8')
    cout_login(bodyy)
    print(param1,param2,param3,sep=' ')
    if check_string_auth(bodyy) == 0:
        v = get_updates(bodyy,param1,param2,param3)
        print(v)
        if v != None:
            print("here")
            send_message(v,"Someone tried to access your account")
            cout_restat()
    # print(f"Received {bodyy}")
    # print(get_updates())

# Step 1: Establish connection to RabbitMQ server
connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

# Step 2: Declare the queue you want to consume from
channel.queue_declare(queue='syslog_queue')

# Step 3: Set up subscription to the queue
channel.basic_consume(queue='syslog_queue', on_message_callback=callback, auto_ack=True)

print('Waiting for messages. To exit, press CTRL+C')

# Step 4: Start consuming messages
channel.start_consuming()
```

III- Result

The RabbitMQ SecLog Alert system was successfully implemented and tested. The following key results were observed:

* ***Log Transmission and Processing***
Logs generated by FreeIPA were reliably transmitted to RabbitMQ in real-time, with minimal delays. RabbitMQ efficiently queued the logs, ensuring they were available for processing by the log filtering machine without any message loss.
* ***Incident Detection***
The log filtering machine was able to detect predefined threat patterns, such as consecutive failed login attempts.
* ***Real-Time Alerts***
Alerts were sent to the system administrator's Telegram account within seconds of incident detection. The notification included details such as the username involved. The Telegram API integration ensured that alerts were received in real-time, allowing administrators to respond quickly to potential threats.

IV- Discussion

The RabbitMQ SecLog Alert system demonstrates the effectiveness of using a centralized log management system with real-time alerting capabilities for detecting cybersecurity incidents. By combining FreeIPA for authentication logs, RabbitMQ for message queuing, and the Telegram API for notifications, the system offers a scalable and efficient solution for monitoring potential security threats.

1. ***Strengths of the System***
   * **Real-time Alerts:** The system's ability to send real-time alerts ensures that administrators can respond quickly to potential security incidents, minimizing damage.
   * **Scalability:** The use of RabbitMQ for message queuing allows the system to handle large volumes of logs without performance degradation.
   * **Modular Design:** Each component (FreeIPA, RabbitMQ, Log Filtering Machine, and Telegram API) is decoupled, making the system modular and easy to expand.

2. ***Limitations***
   Limited Threat Detection Scope: This implementation only takes into account the case where an intruder tries to connect more than once (3 times), which is not the only security challenge.
   Single Notification Channel: The system currently relies on Telegram for alert notifications. Expanding the system to support multiple channels (e.g., email, SMS) would increase reliability in case of messaging delays or failures.

V- References

- RabbitMQ documentation: https://www.rabbitmq.com/ (https://www.rabbitmq.com/)
- FreeIPA documentation: https://www.freeipa.org/ (https://www.freeipa.org/)
- Telegram Bot API documentation: https://core.telegram.org/bots/api (https://core.telegram.org/bots/api)