A Study on Zero Trust Security Based Email Spam Detection System

1st Laiba Khan
Department of Computer Science
University of Engineering and Technology
Lahore, Pakistan
Email: 2022cs70@student.uet.edu.pk

2nd Wali Allah
Department of Computer Science
University of Engineering and Technology
Lahore, Pakistan
Email: 2022cs86@student.uet.edu.pk

***Abstract*—In the modern era of cyber threats, ensuring strong security for user authentication and data is very necessary. This research paper presents a detailed Information Security system designed with a zero-trust security principle, giving the view point of "never trust, always verify." The system implements secure login and signup mechanisms with an intelligent email spam detection module, which gives a system that protects from adversial attacks that can fool the model. User authentication is applied using multifactor authentication (MFA), including one-time passwords (OTP) that ensure the authorized access. In addition, the system implements session control, store log, and timeout implementation to reduce the risk of unauthorized access during or after user activity. For the email spam detection part, a machine learning-based model is employed with the main concern of safety from adversial attacks, e.g. inputs specifically designed to fool the model. This checks that the system not only accurately detects spam, but is also strong against bad and manipulation attempts. The combination of Zero Trust principles with adversarial aware spam detection, MFA, and advanced session management ensures strong defense methods that can serve as a secure backbone for modern digital applications.**

***Index Terms*—Zero Trust Security, Multi-Factor Authentication (MFA), One-Time Password (OTP), Email Spam Detection, Adversarial Attacks, Session Control, Information Security, Logging and Monitoring, Secure Authentication, Data Protection, Cybersecurity.**

I. Introduction

The most widely used communication tool on both professional and personal levels is email. It plays a very important role in various digital system such as the online banking system. Due to its high usage in educational and business fields, there is a high chance of any unauthorized activities like spam, phishing, and malware attacks. Therefore, the security of emails is very important.

A. Background of Spam Detection

As emails have become more engaging communication source but it is often used to send unwanted and harmful messages which is known as spam. Old spam filters use fixed rules, but smart attackers can fool them by changing their messages

B. Need for Secure Spam Filtering Systems

As online threats and attacks such as data poisoning or adversarial attacks have become stronger, it is also very important that spam filters are not only accurate, but also secure. These systems need to protect the data they handle, create a barrier of any unauthorized access, and stay strong even if someone tries to guess them with fake or altered messages.

C. Brief Overview of Approach

This project is all about an AI-based system that checks if emails are spam or not. It has a secure web interface made with React (Vite) and Flask. To make the system safe, we used Zero Trust Security Principles, which means that no one can use it without proving valid identity. We also added extra protection to the AI model so it can not be easily fooled by tricky or fake emails.

D. Contributions of Our Work

We built a complete web application that can check emails for spam in real time using React and Flask. For more security we used Zero Trust security methods which make sure only verified users can access the system.

II. Related Work

In this section, I will discuss some relevant researches that secure the system with multiple approaches.
One of the researchers made a tool called "SpamDam" to check the spam messages with keeping people's messages private. It learns from social media spam and uses "federated learning" which means it gets smarter without needing to see personal messages. It also tests if the system can catch typical trickful spam messages that try to fool it [1]. There are powerful spam filters using machine learning but users are still facing more fraud and attacks. The main problems are that spam keeps changing and spammers are finding new ways to get approach the system. This paper talks about these problems and shows that if we do not manage the changes then spam filters would not work as well [2]. Email is a cheap and easily accessible method of communicating, but the Internet also increases security risks. This paper explores how deep learning can detect threats like phishing, malware, and spam in email security. Recheck the methods used to protect emails and compares the advantages and disadvantages of using deep learning for better security [3]. Phishing and spam emails are

increasing fast so we need smart filters to catch them. AI and machine learning can help by checking parts of the email like sender info, subject and content. It also mentions useful methods and areas where more research is needed [4]. Spam is still a big issue online. The study explores how machine learning and deep learning can help detect spam better and faster using best methods with bigger datasets [5].

Emails are used everywhere now so keeping them safe and checking for misuse is important. The study focus on real email crime cases and the tools used to investigate them. It also compares different software used for email security and investigation [6]. Spam emails are increasing day by day and machine learning is helping to filter them better. This review focus on how email services like Gmail and Yahoo use machine learning and what challenges they face. We suggest using deep learning in the future to improve spam filtering [7]. Cybersecurity is all about keeping our personal information safe online, focusing on both people and technology. This paper looks at the weaknesses of current security models and suggests ways to improve risk protection in the future [8]. Our work is slightly different from others because in all the researches mostly email detection takes all the importance rather than its security or if any security will be implemented, then it can easily breach the data and confuse the model. We added Zero Trust Security to check user identity and stop misuse. We also used adversarial techniques to protect the system from spam that tries to fool the AI.

III. Methodology

The primary focus of the methodology followed in this project was to secure an AI-based email spam detection system using Zero Trust Security principles. First of all, we selected a simple and effective machine learning model for spam detection. We obtained a Logistic Regression classifier trained on a labeled email dataset from publicly available resources on GitHub. This model was already trained using TF-IDF feature extraction to transform text data into numerical features suitable for classification. Instead of retraining the model, we focused on enhancing its security by building a complete secure web application around it.

A. Zero Trust Security Approach

Zero Trust Security is the idea that no user or device should be trusted automatically, even if they are inside the network. Every request must be verified before giving access. In our project, we strictly followed this principle to secure both user authentication and the spam detection system.

We applied Zero Trust in the following ways:

- **Multi-Factor Authentication (MFA):** Every user needs to get their identity verified through two steps: entering their correct email and password first, and then entering a One-Time Password (OTP) which is sent to their email. If the user is unable to complete both steps successfully, the user cannot access the system.
- **Role-Based Access Control (RBAC):** Different parts of the system are available to users based on their roles. For example, only users with an ”admin” role can view security logs and attack events. Normal users can only access the spam checker feature. This helps reduce damage if someone’s account gets hacked.
- **Token-Based Authentication:** A secure JSON Web Token (JWT) is generated after a successful login. This token includes important information such as the user’s role, email, and IP address. Every time a request is made by the user, the user’s identity gets verified by the backend via this token.
- **IP Address Verification:** To make sure the session is safe, we linked the JWT token to the user’s IP address. If the IP address changes during a session, the system automatically forces the user to logout. This helps stop someone else from using your account without permission.
- **Rate Limiting:** We applied strict limits on how many requests a user can send (like spam checks or login attempts) per minute. The limit is set to 10 requests per minute in our system; any additional request (starting from the 11th) is blocked by the system. This protects the system from brute-force attacks, flooding, and abuse.
- **Session Expiry and Timeout:** Each JWT token is valid for only a limited time period (8 hours). After that, the token expires and the user is logged out automatically. This reduces the risk of long lasting sessions being misused.
- **Adversarial Input Checking:** Before any email is checked by the spam detection model, it is first scanned for suspicious patterns like SQL Injection, XSS attacks, or fake content. If an adversarial input is detected, the system blocks it and logs the event.
- **Security Monitoring and Logging:** Every security event, such as failed logins, wrong OTP attempts, adversarial input detection, and rate limit violations, is recorded in a special collection inside MongoDB. By the help of these logs administrators can monitor the system’s health and detect attacks early.

By adding these multiple layers of verification and protection, we made sure that no part of the system trusts any user, input, or connection by default. Every action and access request must first prove that it is trustworthy. This approach makes our spam detection system highly resistant to both external and internal threats.

B. System Architecture

The high-level system architecture is shown in Fig. 1. It describes how different components interact securely in our Zero Trust Security based spam detection system.

The architecture includes:

- **Frontend (React Vite App):** Handles user authentication, input validation, encryption, and communication with the backend.
- **Backend (Flask API):** Processes user requests, validates authentication tokens, checks for adversarial inputs, and communicates with the spam detection model.

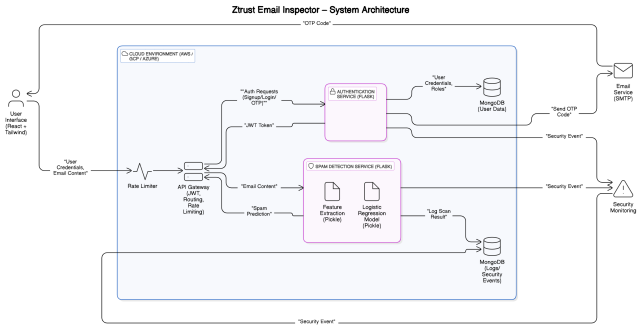

Fig. 1. System Architecture of Zero Trust Based Email Spam Detection

- **Spam Detection Module:** A logistic regression model that predicts whether an email is spam after adversarial checks.
- **MongoDB Atlas:** Stores user data, OTPs, logs, and security events.
- **Security Monitoring:** Real-time logging of attacks, authentication failures, and suspicious activities.

C. Spam Detection Model

Our system used a Logistic Regression model that was trained to classify emails as Spam or Not Spam. Before sending any email content to the model:

- We cleaned the input to remove anything harmful.
- We checked the email to make sure it did not contain anything harmful or unusual.
- We used a method to measure how random or strange a message is that can cause to confuse the model.

After making sure it was safe and clean, we gave the input to the model.

D. Backend Security Measures

We made the backend safe by doing the following:

- **Limiting requests:** We limited how many times a user can send requests to keep the system safe.
- **Password protection:** We used hashing to turn passwords into secret codes before saving them.
- **Encryption:** Encryption is used temporarily during the registration process to securely pass the password between the /register/initiate and /register/verify endpoints.
- **Suspicious activity tracking:** We saved records of unusual actions like failed logins or too many requests in MongoDB.
- **OTP safety:** If someone tried the wrong OTP more than 5 times, the system blocked further attempts.

E. Frontend Security Measures

On the React frontend, we did the following to keep things safe:

- **Input validation:** We checked and cleaned all user inputs (like email, password, OTP) before sending them to the backend.
- **JWT token storage:** We stored JWT tokens in `localStorage` so they stay saved even after closing the browser.
- **Error handling:** We used Axios interceptors to catch errors safely and log important security events.

F. Deployment and Hosting

For hosting the project securely:

- The frontend was deployed on Vercel and the backend was deployed on Railway.
- MongoDB Atlas was used as the cloud database with secure connection strings.
- HTTPS was used in all communications between frontend, backend, and database.
- Environment variables were used to keep all secrets and API keys hidden from the public.

G. Security Testing

We tested our system against different types of attacks such as:

- Submitting fake or malicious emails to see if the model was fooled.
- Trying to guess OTPs manually and checking if the system blocked the user after too many wrong attempts.
- Checking if expired JWT tokens are used and if the system logged out automatically.
- Changing the IP address after login to see if the session ends automatically.

IV. Results

We made our spam checker secure and then tried it to see if it works properly and keeps everything safe and correct. A brief discussion of the results follows:

A. Security Performance

The security steps we used worked just like we wanted:

- **MFA Verification:** OTP-based login and signup worked smoothly, with OTPs expiring after 10 minutes and blocking after 5 wrong attempts.
- **Session Control:** JWT tokens expired automatically after 8 hours or when the IP address changed and asking the user to again login securely.
- **Adversarial Input Detection:** Inputs containing SQL injection patterns, XSS scripts or unusually random content were successfully detected and blocked before reaching the model.
- **Rate Limiting:** If a user tried to login or check spam too many times then the system blocked them for doing it more than allowed.

B. Monitoring and Logging

All important security events were properly logged in the database, including:

- Failed login attempts
- OTP guess attempts
- Adversarial input detections

- Rate limit violations

This showed that real time monitoring was active and could help detect any attack or strange behavior early.

C. Deployment and User Experience

The frontend and backend integration was successful. Users were able to:

- Register and login with strong verification.
- Safely submit emails for spam detection.
- Get instant predictions along with a confidence score.

The website worked smoothly, loaded quickly and kept all data safe using HTTPS.

D. Screenshots of System

To demonstrate the working of the secured system we provide two important screenshots.

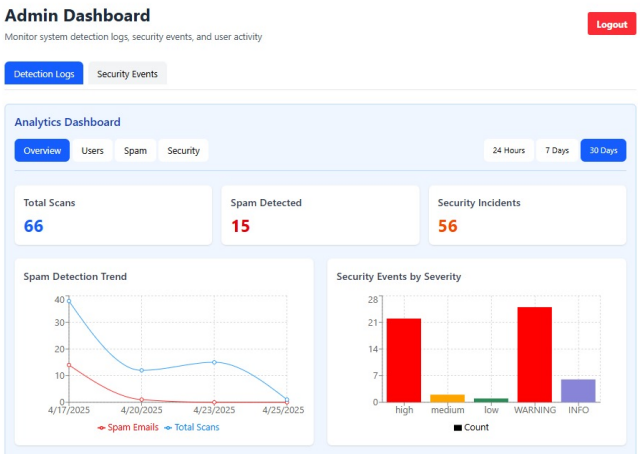

Fig. 2. Admin dashboard showing graphs for logs and security events

As shown in Fig. 2 the admin dashboard shows live data to track security events and detect any unusual activity.

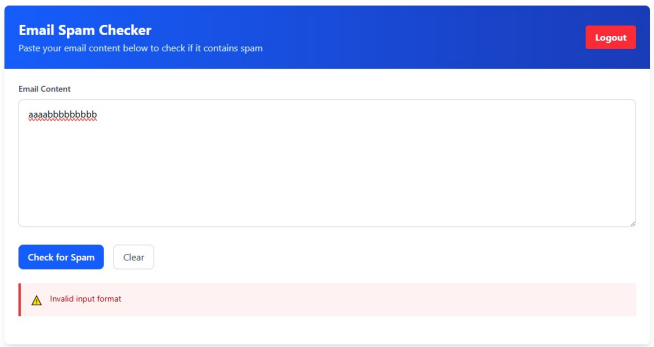

Fig. 3. User-side view showing adversarial input detection and rejection

As shown in Fig. 3 if a user tries to submit an adversarial email input then the system detects and blocks the input before it reaches the model while keeping the spam checker working correctly and safely.

During implementation, we explored how well our method works. We tested how Zero Trust Security can be used with an AI system. We added features like OTP login, session control and checks to block harmful inputs. These methods helped keep the spam checker safe from fake emails and hackers. Some security steps like OTP, login take a little delays but the system still worked fast and smoothly. Overall our results showed that Zero Trust made the system safer by stopping attacks and blocking unwanted access.

V. Conclusion

This project showed that using machine learning with Zero Trust Security can help make a safe and smart system to check for spam emails. We added features like one-time passwords (OTP), secure login tokens (JWT), input checks and activity logs to make the system harder to attack. Our results showed that the model gave good results and the security features helped protect it from common problems.

Acknowledgment

We would like to thank our instructor Sir Waqas Ali from UET Lahore for his guidance and support throughout this project. His clear instructions and helpful feedback kept us on track. We also appreciate the open source community for providing the spam detection model that made this project possible.

References

[1] Yekai Li, Rufan Zhang, Wenxin Rong, and Xianghang Mi. Spamdam: towards privacy-preserving and adversary-resistant sms spam detection. *arXiv preprint arXiv:2404.09481*, 2024.

[2] Francisco Jáñez-Martino, Rocío Alaiz-Rodríguez, Víctor González-Castro, Eduardo Fidalgo, and Enrique Alegre. A review of spam email detection: analysis of spammer strategies and the dataset shift problem. *Artificial Intelligence Review*, 56(2):1145–1173, 2023.

[3] Mozamel M Saeed and Zaher Al Aghbari. Survey on deep learning approaches for detection of email security threat. *Computers, Materials & Continua*, 77(1), 2023.

[4] Asif Karim, Sami Azam, Bharanidharan Shanmugam, Krishnan Kannoorpatti, and Mamoun Alazab. A comprehensive survey for intelligent spam email detection. *Ieee Access*, 7:168261–168295, 2019.

[5] Andronicus A Akinyelu. Advances in spam detection for email spam, web spam, social network spam, and review spam: Ml-based and nature-inspired-based techniques. *Journal of Computer Security*, 29(5):473–529, 2021.

[6] Esra Altulaihan, Abrar Alismail, MM Hafizur Rahman, and Adamu A Ibrahim. Email security issues, tools, and techniques used in investigation. *Sustainability*, 15(13):10612, 2023.

[7] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, and Opeyemi Emmanuel Ajibuwa. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6), 2019.

[8] Kutub Thakur, Meikang Qiu, Keke Gai, and Md Liakat Ali. An investigation on cyber security threats and security models. In *2015 IEEE 2nd international conference on cyber security and cloud computing*, pages 307–311. IEEE, 2015.