INTERNATIONAL HELLENIC UNIVERSITY
DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

THESIS

ELECTRONIC MAIL MANAGEMENT SYSTEMS

Student:
Dimitrios Mouroudelis
Student ID: 154499

Supervisor:
Antonis Sidiropoulos

26 May 2024

Title of Dissertation ELECTRONIC MAIL MANAGEMENT SYSTEMS
Code of Dissertation 22271
Student's full name Dimitrios Mouroudelis
Supervisor's full name Antonis Sidiropoulos
Date of undertaking 20-10-2022
Date of completion 26-05-2024

We hereby affirm the authorship of this paper as well as the acknowledgment and credit of whichever assistance We received in its composition. We have, furthermore, noted the various sources from which We extracted data, ideas, visual or written material, in paraphrase or exact quotation. Moreover, we affirm the exclusive composition of this paper by myself only, for the purpose of it being a dissertation, in the Department of Information and Electronic Engineering of the I.H.U.

This paper constitutes the intellectual property of Dimitrios Mouroudelis the student that composed it. According to the open-access policy, the author/composer offers the International Hellenic University authorization to use the right to reproduce, borrow, publicly present and digitally distribute the paper globally, in electronic form and media of all kinds, for teaching or research purposes, voluntarily. Open access to the full text, by no means grants the right to trespass the intellectual property of the author/composer, nor does it authorize the reproduction, republication, duplication, selling, commercial use, distribution, publication, downloading, uploading, translation, modification of any kind, in part or summary of the paper, without the explicit written consent of the authors.

The approval of this dissertation by the Department of Information and Electronic Engineering of the International Hellenic University does not necessarily entail the adoption of the author's views, on behalf of the Department.

Dedication

This particular thesis is dedicated to my family and friends who supported and encouraged me to conclude it. I would also like to thank my Professor Mr. Antonios Sidiropoulos for his great collaboration and guidance throughout this thesis.

Prolog

Electronic mail, since its inception, has transformed how we communicate, making it an essential tool both personally and in professional environments. This generational invention shaped the modern world while retaining the same functionality. Over the years with the expansion of the internet, more and more issues started to arise threatening users and the data that was exchanged. The plethora of new challenges paved the way for developing more sophisticated Email Management Systems to address them and bring new functionalities. This thesis explores the intricate landscape of electronic mail management systems, focusing on the integration and evaluation of essential protocols and technologies that deal with those issues.

The journey began with an interest in understanding the challenges and solutions in managing email security, authentication, and spam prevention issues that profoundly affect personal and professional communication on the internet. The motivation to explore this topic derives from the curiosity to acknowledge those issues and come up with their solutions by constructing and managing a mailserver.

This thesis explores the complexities of electronic mail management systems, focusing on the implementation and integration of key protocols and technologies to ensure secure, efficient, and reliable email communications. By addressing the challenges of spam, security, and user authentication, this work aims to enhance the functionality and robustness of mail servers. This study underscores the importance of continuously evolving email systems to meet modern digital communication needs.

Abstract

The subject of this thesis is Electronic mail Management Systems. The complexity of these systems as well as the critical nature of the data exchanged poses several challenges in maintaining their efficiency, security, and management. To overcome these challenges, this thesis studies the essential protocols and technologies involved in such systems, culminating in the creation and configuration of a functional mail server. The thesis begins with the analysis of protocols such as SMTP, IMAP, and POP that are essential for the transmission and retrieval of emails. It also examines security protocols such as SSL/TLS and STARTTLS, and email authentication protocols like SPF, DKIM, and DMARC while discussing the importance of related DNS records. The analysis then shifts to the mail server technologies, with an evaluation of MTAs such as Postfix, Exim, and Sendmail, as well as MDAs like Dovecot. Mechanisms for spam prevention such as SpamAssassin and ClamAV were also discussed. Additionally, user authentication technologies like SASL and LDAP are addressed. The practical part of this thesis included the construction of an email server by utilizing the necessary technologies and methods to ensure its functionality, security, and reliability. Finally, several tests were conducted with their results highlighting the effectiveness of these technologies in addressing security, authentication, and spam prevention challenges contributing valuable insights regarding electronic mail management systems.

Περίληψη

Το αντικείμενο μελέτης της συγκεκριμένης πτυχιακής εργασίας αφορά τα Συστήματα διαχείρισης ηλεκτρονικού ταχυδρομείου. Η πολυπλοκότητα των συγκεκριμένων συστημάτων και η κρισιμότητα των δεδομένων που διακινούνται μέσω αυτών ελοχεύει αρκετές προκλήσεις όσον αφορά επίτευξη της ορθής και ασφαλούς λειτουργίας άλλα και διαχείρισης τους. Για να ξεπερασθούν αυτές οι προκλήσεις η πτυχιακή πραγματοποιεί μελέτη σε βασικά πρωτόκολλα και τεχνολογίες που αφορά αυτά τα συστήματα καταλήγοντας στη δημιουργία και ρύθμιση ενός λειτουργικού διακομιστή ηλεκτρονικής αλληλογραφίας. Η πτυχιακή εργασία ξεκινά με την ανάλυση των πρωτοκόλλων αλληλογραφίας SMTP IMAP και POP που είναι απαραίτητα για την επικοινωνία από και προς τον διακομιστή ηλεκτρονικής αλληλογραφίας. Επιπλέον αναλύονται πρωτόκολλα ασφαλείας όπως το SSL/TLS και το STARTLS αλλά και πρωτόκολλα αυθεντικοποίησης ηλεκτρονικής αλληλογραφίας όπως το SPF το DKIM και το DMARC. Γίνεται επίσης λόγος και για την σημασία των εμπλεκομένων εγγραφών DNS. Η ανάλυση συνεχίζεται με τις τεχνολογίες του συστήματος αξιολογώντας αρχικά τα MTA όπως το Postfix το Exim και το Sendmail άλλα και MDA όπως το Dovecot καθώς επίσης γίνεται η διερεύνηση των μηχανισμών αποφυγής ανεπιθύμητης αλληλογραφίας όπως το SpamAssassin και το ClamAV. Αναφορά γίνεται επίσης και τεχνολογίες αυθεντικοποίησης των χρηστών του διακομιστή αλληλογραφίας όπως το SASL και το LDAP. Η πτυχιακή επίσης περιλαμβάνει και πρακτικό κομμάτι περιγράφοντας την κατασκευή ενός διακομιστή αλληλογραφίας καθώς και την προσθηκη απαραίτητων τεχνολογιών και μεθόδων που διασφαλίζουν την ορθή, ασφαλή και αξιόπιστη λειτουργία του. Τέλος, πραγματοποιήθηκαν δοκιμές, τα αποτελέσματα των οποίων αναδεικνύουν την αποτελεσματικότητα αυτών των τεχνολογιών όσον αφορά την λειτουργία του αλλά και την αντιμετώπιση ζητημάτων ασφαλείας, αυθεντικοποίησης και πρόληψης ανεπιθύμητης αλληλογραφίας, συμβάλλοντας με πολύτιμες γνώσεις στη διαχείριση συστημάτων ηλεκτρονικού ταχυδρομείου.

Contents

List of Figures

Listings

Chapter 1

Introduction

Since the early days of the internet, the exchange of information and direct communication between users has been a critical issue. Email management systems emerged to address this need, allowing users to exchange emails via the Internet. As the Internet became more widespread, the development of such email systems proved vital for the operations of organizations and companies. Because of the internet expansion, email systems need to be more sophisticated to overcome numerous challenges, including efficient spam filtering, robust security measures to protect sensitive information, and reliable authentication methods to ensure the legitimacy of email sources.
To address these challenges, a comprehensive approach involving multiple technologies and protocols was implemented. First of all, the mailserver was developed using the Postfix MTA and Dovecot MDA. To address the security issues, protocols like SSL/TLS and STARTTLS were integrated to encrypt user communications. Utilities like SpamAssassin and ClamAV were added in to deal with email Spam and malware issues. Additionally, authentication technologies like SASL and LDAP were added to verify user credentials and maintain secure access. There is also an implementation of email authentication protocols such as SPF, DKIM, and DMARC. It provides the background of what the DNS records of a mailserver should be to ensure uninterrupted communication with the internet.
This thesis also discusses the functionality of mail protocols such as SMTP, IMAP and POP noting their importance and role in email communications. There is also a comparison between different MTA's such as Postfix, Exim and Sendmail discussing their advantages and disadvantages over different mailserver environments.

1.1 Problem Statement

The core issue of this thesis is managing and securing an email server. Providing solutions to these issues has become increasingly complex. Email systems must address several critical issues to ensure efficient and reliable service.
One of the primary challenges is to determine the core architecture of the mailserver. Choosing the appropriate MTA and MDA solution for every mailserver environment can be challenging. By conducting a thorough analysis on the Different MTAs and writing down their advantages and disadvantages, Posfix was chosen as the core mailserver.
The second challenge was to ensure secure email transmission and the protection of sensitive information from interception and tampering. To address this issue the SSL/TLS and STARTTLS

mechanisms of encryption were implemented.
Thirdly, mail authentication protocols were necessary integration in order to verify the legitimacy of email sources. For this purpose, SPF, DKIM, and DMARC were deployed as solutions to this issue creating the appropriate DNS records.
Among the challenges is combating the proliferation of spam and malicious emails, which can compromise user security and data integrity. To end this, several anti-spam and anti-malware mechanisms were added to the mailserver like SpamAssassin and ClamAV.
Lastly, there was a need to find a feasible solution on user authentication and authorization issues. Those issues were addressed by developing the appropriate directives in order to integrate user authentication mechanisms in the likes of SASL authentication and LDAP for connecting to the mail server.
Integrating these multifaceted requirements into a cohesive and efficient mail server infrastructure presents a significant challenge, necessitating the use of advanced protocols and technologies to meet the demands of modern email communication.

1.1.1 Contribution

This thesis makes several contributions to the subject of electronic mail management systems by addressing important issues and offering workable solutions by combining different technologies and protocols:

1. Comparative Analysis of MTAs: Evaluated and compared Postfix, Exim, and Sendmail, highlighting their strengths, weaknesses, and best-use scenarios.

2. Implementation of Secure Communication Protocols: Integrated SSL/TLS and STARTTLS to ensure encrypted email transmissions, protecting data from interception and tampering.

3. Advanced Spam and Malware Protection: Implemented SpamAssassin and ClamAV for robust spam filtering and malware detection, enhancing email security.

4. Email Authentication Mechanisms: Deployed SPF, DKIM, and DMARC protocols to verify email legitimacy and prevent spoofing, improving trustworthiness.

5. Efficient Email Storage and Retrieval: Utilized Dovecot as the Mail Delivery Agent (MDA) for managing email storage and retrieval, ensuring efficient performance.

6. User Authentication and Directory Services: Integrated SASL and LDAP for secure user authentication and directory services, maintaining secure access control.

7. Configuration and Management of DNS Records: Configured critical DNS entries, including MX records, to effectively manage and route email traffic.

By addressing these critical areas, this thesis contributes to the development of a comprehensive, secure, and efficient electronic mail management system, capable of meeting the demands of modern digital communication

1.2 Thesis structure

The structure of this thesis is designed to systematically address the key components and challenges of electronic mail management systems. The following chapters provide a comprehensive overview of the protocols and technologies involved, as well as a detailed analysis of their implementation in the mailserver.

1. **Chapter 1: Introduction** The introductory chapter outlines the project's issues this thesis is trying to solve, innovative contributions, and thesis format.

2. **Chapter 2: A Thorough Analysis of the Mailserver Protocols** This chapter provides a thorough analysis of the protocols involved in the mail server, including SMTP, IMAP, POP, SSL/TLS, STARTTLS, SPF, DKIM, and DMARC. Each protocol's utility, functionality, and integration into the email system are detailed, along with the significance of DNS entries like MX records.

3. **Chapter 3: A Comprehensive Study of Mailserver Technologies** This chapter explores the technologies used in the mail server. It covers Mail Transfer Agents (MTAs) such as Postfix, Exim, and Sendmail, including a comparative analysis of their strengths and weaknesses. It also details the role of Mail Delivery Agents (MDAs) like Dovecot, the implementation of SSL certificates from Let's Encrypt, spam protection mechanisms like SpamAssassin and ClamAV, and authentication technologies such as SASL and LDAP

4. **Chapter 4: Creating the Mailserver** This chapter provides a detailed guide on setting up the mail server, including the integration of protocols and technologies discussed in previous chapters. It features step-by-step instructions, configuration files, and code snippets to illustrate the implementation process.

5. **Chapter 5: Experiments and Conclusion** This chapter details various tests conducted to ensure the functionality and security of the mail server. It includes tests for email authentication protocols (SPF, DKIM, DMARC), encryption (SSL/TLS, STARTTLS), and protocol functionality (SMTP, IMAP). The effectiveness of spam prevention mechanisms (SpamAssassin) and HELO access restrictions are also evaluated. The conclusion summarizes the study of the previous chapters and suggests future work on developing an online user interface.

Chapter 2

A thorough analysis of the Mailserver protocols

This chapter delves into the essential protocols that form the backbone of email communication systems. Email protocols facilitate the sending, receiving, and securing of email messages. It discusses the primary protocols for outgoing and incoming email traffic, the security protocols that safeguard email transmissions, and the email authentication protocols that verify the legitimacy of email sources.

2.1 Outgoing connection protocols

Outgoing connection protocols play a crucial role in email communication as they help facilitate the transfer of emails from the sender's server to the recipient's server. The Simple Mail Transfer Protocol (SMTP), which guarantees the dependable delivery of emails over the Internet, is the main protocol used for this purpose.

2.1.1 SMTP

SMTP[1] was introduced in 1982 by Jon Postel[2]. It was created to improve on previous protocols like the Mail Transfer Protocol "MTP" and enable the sending of email messages across networks. Because of its ease of use and efficiency, SMTP became one of the top standards for email delivery. In 2008 SMTP was upgraded and included security measures and refinements, including Extended SMTP, or ESMTP[3].
The SMTP functions using a store-and-forward method to transport emails through a network of intermediary servers before arriving at their final destination. Through a series of command/response exchanges, SMTP manages the initial connection, sender and recipient specification, message transfer, and session termination. The SMTP utilizes port 25 for regular transmission without security. The other two ports that SMTP is using are the 587 for the TLS connection and the 465 for the SSL connection.
In order to begin an SMTP communication, the client must connect to the server by issuing an EHLO command to initiate the mail session. With the MAIL FROM command, the sender's email address is specified, while the RCPT TO command the recipient's email addresses are provided. Issuing the DATA command transmits the email's content, including headers, the subject of the email, and its

```
Escape character is '^]'.
220 mail.mtest1.iee.ihu.gr
EHLO mtest1.iee.ihu.gr
250-mail.mtest1.iee.ihu.gr
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250-SMTPUTF8
250 CHUNKING
MAIL FROM: jimouroudelis@mtest1.iee.ihu.gr
250 2.1.0 Ok
RCPT TO: jimouroudelis@gmail.com
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
SUBJECT: This is a Test Mail
Hello, this is a test email for my thesis.
.
250 2.0.0 Ok: queued as 8631A1A032
quit
221 2.0.0 Bye
Connection closed by foreign host.
root@mailtest:~#
```

Figure 2.1: SMTP communication

body. The and of the body of the email is marked with a single dot ".". Once the message is transferred the SMTP connection indicates its queue code. Finally, the session is terminated by issuing the QUIT command, completing the email transmission process. In the figure 2.1 is an example of an SMTP communication.

One issue that the SMTP faces is the fact that in its simplest version, it lacks built-in security protections, making it vulnerable to interception and eavesdropping. To improve its security, SMTP can be used with Transport Layer Security (TLS), which encrypts communication between the email client and the server. With the help of the STARTTLS extension, the security of email communication is improved by converting an existing plaintext connection into a secure TLS connection.

To sum up, SMTP is a stable and adaptable protocol for sending emails, and it is supported by a lot of add-ons and improvements that make email delivery secure, reliable, and efficient. Because of its compatibility with TLS, SSL, and STARTTLS security protocols, it is a crucial part of today's email communication systems.

2.2 Incoming connection protocols

Incoming mail protocols are essential for retrieving and managing emails from a mail server. They allow users to access their email messages and their inboxes from different devices and locations. Two primary protocols are used to serve the incoming mail traffic. The first one is IMAP (Internet Message Access Protocol) and the second one is POP (Post Office Protocol). Both protocols enable users to view and download emails from the server to their local devices, but they operate differently in terms of how emails are stored and managed.

2.2.1 IMAP

The Internet Message Access Protocol (IMAP)[4] is an application layer Internet protocol and its primary goal is to allow users to read and manage their emails directly on the server. IMAP was developed in 1986 at Stanford University by Mark Crispin[5]. It's the most common protocol used by email clients like Thunderbird[6] or Bluemail[7] to retrieve messages from a mail server. One key feature of IMAP is that it allows access to emails from multiple devices by synchronizing their state. This means that actions such as reading, deleting, or organizing emails are reflected on the server and all connected devices that are connected to it. This protocol is particularly useful for users who need to access their email from various locations and devices, as it maintains the email structure and organization on the server, ensuring consistency and accessibility. IMAP utilizes port 143 for unencrypted communication and 993 for encrypted communication using SSL/TLS protocol.

2.2.2 POP

The Post Office Protocol(POP)[8] is an application-layer Internet standard protocol that was developed in the early 1980s. POP has had several updated versions, with POP3[9] being the most widely used. Unlike IMAP, which keeps emails on the server, POP downloads emails to the local device deleting them from the server. This makes POP suitable for users who primarily access their email from a single device and also helps manage the storage on the server. The new version of POP3, unlike its predecessor which the only option was to download emails to the local devices, introduced features that allowed users to leave copies on the server. This marked a significant improvement in email accessibility and management. Over time, POP3 became widely adopted due to its simplicity and ease of implementation. POP utilizes port 110 for unencrypted communication and port 995 for secure, encrypted communication using SSL/TLS protocol.

2.3 Securty protols

Security protocols are essential in email services to protect communications from unauthorized access and ensure data integrity. This section covers the primary security protocols used in mail services.

2.3.1 SSL

The first version of SSL[10] was designed by Netscape in the mid-1990s. It was developed to offer encryption, which secures conversations over a network. Before any data is transmitted, the protocol

establishes a secure connection through a handshake process that involves exchanging encryption keys, encryption methods, and certificates. To protect the confidentiality and integrity of the communication, SSL uses symmetric encryption for data transfer and asymmetric encryption for key exchange. Version 1.0 of the SSL protocol was never released to the general public due to serious security flaws. Despite its problematic predecessor, version 2.0 of SSL was released publicly in 1995. Upon its release, it was found that it had a number of usability and security issues. It used the same cryptography keys to authenticate and encrypt messages. Due to its weak MAC design, which used the MD5 hash function with a secret prefix, it was vulnerable to length extension attacks. Moreover, it provided no protection against the first handshake or explicit message closures, making man-in-the-middle attacks undetected. These flaws led to the development of the Version 3.0 SSL. Despite its groundbreaking role in internet security, the vulnerabilities lead to its deprecation in favor of TLS. SSL 3.0 was the last version, now considered obsolete, but its legacy paved the way for more secure protocols like TLS, which continue to protect online communications today.

2.3.2 TLS

The successor for SSL, Transport Layer Security (TLS)[11], improves network data security by utilizing stronger encryption and authentication mechanisms. Created by the IETF, TLS is an improvement over SSL thanks to its secure hash functions and more powerful encryption algorithms. Additionally, TLS utilizes digital certificates to authenticate the communicating parties, preventing man-in-the-middle attacks and ensuring that emails are exchanged between verified entities. TLS encrypts email transfers in mail servers to provide private, secure communication. TLS is the industry standard for secure network communications because of its improved security and performance in versions 1.0 through the most recent 1.3.

2.3.3 STARTTLS

Building upon the secure foundation provided by SSL/TLS, STARTTLS[12] offers a method to upgrade an existing plain text connection to a secure one. The STARTTLS command, issued within the context of an established connection, allows email clients and servers to negotiate the transition to encrypted communication dynamically. This is vital for email protocols such as SMTP, IMAP, and POP3, enhancing the security of communications by protecting data against eavesdropping and tampering during transmission.[13] STARTTLS thus serves as a versatile and adaptable security tool, enabling legacy systems to implement robust encryption without extensive infrastructure modifications, seamlessly integrating into the existing communication protocols.

2.4 DNS records and Email authentication protocols

DNS (Domain Name System)[14] records serve as a directory that converts domain names into IP addresses, among other purposes, making them vital parts of the internet's infrastructure. The DNS servers keep these records, which direct internet traffic routing. Domain names are mapped to IPv4 addresses via A records, while IPv6 addresses are mapped to AAAA records via DNS records. MX (Mail Exchange) records, which identify the servers in charge of receiving email on behalf of a domain, are essential for forwarding messages to mail servers. By linking to a different domain name rather

than an IP address, CNAME (Canonical Name) records enable a domain to be recognized by more than one name. Text data is stored in TXT records as well. Those records are mostly used on email security protocols. Those protocols are DKIM (DomainKeys Identified Mail), DMARC (Domain-based Message Authentication, Reporting, and Conformance), and SPF (Sender Policy Framework) to confirm domain ownership. Every kind of DNS record has a specific function that contributes to a safe and effective network connection.

2.4.1 MX

The Mail Exchange (MX)[15] record has a crucial role in the DNS by identifying the mail servers that are assigned to receive emails for a specific domain. This record is critical in directing the path of electronic messages across the vast expanse of the Internet. As an email is dispatched to an address affiliated with a particular domain, the sending mail server consults the DNS to retrieve the MX records associated with that domain. These MX records contain essential details, including the priorities of the mail servers, dictating the sequence in which they should be contacted. This prioritization ensures the seamless routing and delivery of emails to the designated mail servers. In essence, MX records serve as the architects of the email delivery path, ensuring that messages are delivered precisely and efficiently to their intended destinations. They are critical components of the Mail server infrastructure that ensures the reliability and accuracy of electronic communication.
After a Dig command here is the MX record that is used in our mail server:
mtest1.iee.ihu.gr. 10800 IN MX 10 mail.mtest1.iee.ihu.gr.

2.4.2 SPF

The Sender Policy Framework (SPF)[16] record serves as a critical component in the realm of email security, functioning as a beacon of authenticity for sender domains. Published in the Domain Name System (DNS) settings, the SPF record meticulously outlines the authorized mail servers permitted to send emails on behalf of a specific domain. Employing a syntax of mechanisms and qualifiers, SPF empowers email recipients to validate the legitimacy of incoming messages by cross-referencing the sending server against the designated list of approved sources. This authentication process acts as a robust defense against email spoofing and phishing attempts, providing a clear distinction between genuine communications and potentially malicious forgeries.

- The first mechanism, v=spf1 identifies this as an SPF record.
- The second mechanism +mx says messages originating from the given MX record should be considered valid.
- The third mechanism provides options for handling failed checks. In our record a ”SoftFail” (all) indicates a warning, allowing the email to be delivered but marking it as potentially suspicious.

2.4.3 DKIM

The DomainKeys Identified Mail (DKIM)[17] protocol emerges as a pivotal facet within the landscape of email authentication, offering a cryptographic seal of legitimacy to digital communications. Imple-

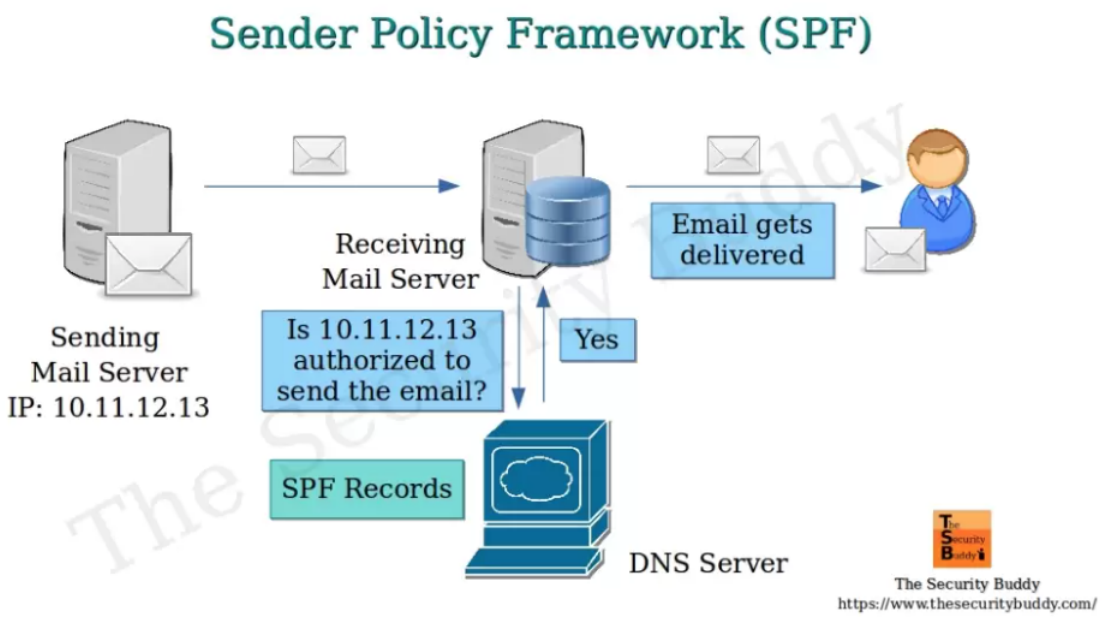


Figure 2.2: SPF Authentication

mented through cryptographic signatures appended to outgoing emails, DKIM provides a means for the recipient's mail server to verify the message's origin and integrity. In essence, the sender domain signs its outgoing emails with a private key, and the recipient's server, equipped with the corresponding public key fetched from the sender's DNS records, can authenticate the signature. This process ensures that the email has not been tampered with during transit and originates from an authorized source. DKIM significantly enhances email security by mitigating risks associated with phishing, spoofing, and unauthorized email modification. Its adoption within email infrastructures exemplifies a commitment to fostering trust and reliability in digital communication channels.

2.4.4 DMARC

The Domain-based Message Authentication[18], Reporting, and Conformance (DMARC) protocol is a robust framework that plays a crucial role in enhancing the security of email communications. Think of DMARC as a set of rules that domain owners establish to ensure that emails sent on behalf of their domain are legitimate. It collaborates with two other security protocols, Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM), to verify the authenticity of the sender's identity. By setting up DMARC policies in the Domain Name System (DNS), domain owners can specify how receiving mail servers should handle emails that fail SPF or DKIM checks. This alignment mechanism ensures that the sender's address in the email matches what SPF and DKIM records say it should be, significantly reducing the risk of email spoofing and phishing attacks.
In addition to its role in email authentication, DMARC provides valuable reporting features. It allows domain owners to receive feedback reports from receiving mail servers about the outcome of their email authentication checks. These reports provide insights into how emails from their domain are being handled, enabling domain owners to monitor for suspicious activities and potential impersonation attempts. DMARC, with its comprehensive approach to email security, empowers organizations

to take proactive measures against unauthorized use of their domain and strengthens the overall trustworthiness of email communication.

Chapter 3

A Comprehensive Study of Mailserver Technologies

3.1 Introduction

In the continuously advancing realm of digital communication, ensuring the security and efficiency of email systems is paramount. This chapter discusses the key technologies that strengthen electronic mail management systems by providing solid security and reliable performance. We examine how Postfix and Dovecot, fundamental components of mail servers, work together. Additionally, we review how SSL/TLS and Let's Encrypt secure data transmissions. Tools like SpamAssassin and ClamAV are explored for their roles in blocking spam and preventing virus infections, respectively. Lastly, we discuss LDAP and SASL authentication mechanisms that support user verification processes. Together, these technologies create a comprehensive system that ensures the secure and efficient functioning of email systems, meeting both current demands and future challenges.

3.2 MTA

Mail Transfer Agents[19] (MTAs) are a crucial component of the email infrastructure, responsible for transferring email messages from one computer to another using the Simple Mail Transfer Protocol (SMTP). As the backbone of email delivery, MTAs accept outgoing emails from users and then deliver these messages to the appropriate recipients. This routing process involves a complex series of individual steps, including DNS lookups to find the recipient's mail server and determining the best path for the message across multiple intermediate servers. In addition to routing, MTAs play a key role in enforcing policies related to email delivery, such as retrying delivery when a recipient's server is unavailable, handling bounced messages if the email cannot be delivered for some reason, and implementing security measures to block spam or verify sender identities. In order to be effective, MTAs must be highly reliable and secure, ensuring the delivery of emails while simultaneously protecting against potential threats like spamming and phishing attacks. Some popular examples of MTA software are Postfix, Exim, and Sendmail, each offering different features and configurations to meet the different infrastructure needs.

3.2.1 Postfix

Postfix[20], is a widely-used Mail Transfer Agent (MTA) known for its high performance, simple configuration, and strong security features. It offers a simpler, more secure, and more efficient mechanism for routing and delivering emails. Postfix operates by receiving emails from mail clients, processing them, and then deciding the next hop in the email delivery chain based on DNS lookup results. It can handle local mail delivery within a system or relay mail to other MTAs over the network. Notably, Postfix's architecture is modular, allowing it to work seamlessly with other software such as spam filters and antivirus programs. This modularity also makes it highly adaptable, enabling it to fit into various setups from small-scale systems to large enterprise environments. Postfix supports numerous protocols including SMTP, SMTP AUTH an extension of SMTP that allows clients to log into the mail server to send email, and TLS for secure communication, ensuring that all data transfers maintain confidentiality and integrity.
Key features that made Postfix one of the most widely used MTA are explained below [21]:

1. Security-Centric Architecture: Postfix sets itself apart with a security-focused design, incorporating measures that strengthen email infrastructures' resistance to possible attacks. Postfix operates in a chroot environment, which limits the system's vulnerability surface by limiting its operations to a certain directory. This helps to minimize the impact of security breaches. The main.cf file's flexible access controls and limits give administrators the ability to precisely adjust the permissions of different components, strengthening the system's defenses. Because of its unwavering dedication to security, Postfix is positioned to protect users' email communications' confidentiality and integrity by acting as a steadfast bulwark against malicious activity.

2. Efficient Queue Management: Postfix is excellent at handling the email queue, guaranteeing that messages are delivered consistently and on time. By effectively prioritizing and organizing messages, the queue manager reduces the possibility of message loss and maximizes email traffic flow. This feature is essential for managing fluctuating loads and network circumstances, which enhances the email system's overall responsiveness and stability. For organizations where email responsiveness is critical, Postfix's queue management features make it a dependable option, whether handling a large volume of outgoing messages or incoming mail from various sources.

3. Modular and Extensible Architecture: Postfix relies on its extensible and modular architecture, which allows for customization and adaptability to a wide range of use cases. Because of its modularity, system administrators can add or remove parts to customize the mail server to fit particular organizational needs. Given its built-in adaptability, Postfix can easily interface with other programs, making it possible to use programs like Dovecot for email archiving and retrieval. Postfix is a flexible solution that can grow and adapt to meet the changing needs of various environments, ranging from small businesses to large corporations.

These combined features make Postfix a secure, adaptable, and efficient choice for organizations seeking a reliable Mail Transfer Agent to underpin their email infrastructure.

3.2.2 Exim

Another widely used MTA is Exim[22][23]. Exim is mostly known for its high flexibility in configuration. It is commonly used on Unix-like operating systems. It was originally designed at the

University of Cambridge in 1995 and it was destined to replace Smail-3[24]. Exim was mainly designed with a focus on flexibility and configurability, allowing it to adapt to many different network environments without requiring significant modifications to its source code. This flexibility is largely achieved through its configuration file, which allows for complex routing and delivery rules tailored to specific needs.
Strong aspects and features of Exim that made it one of the most used MTA in the world are explained below:

1. Configurability: Exim's extended configuration system allows administrators to define complex routing rules and handling procedures through its single, yet every complex, configuration file. The level of control that Exim's configuration provides is very beneficial for constructing the system to specific operational requirements without the need to modify its source code.

2. Security: Security is a great aspect of Exim's design. It offers the capability to run each connection in a separate unprivileged process. This leads to minimizing the risk of larger security breaches. Furthermore, Exim uses TLS to secure email communications, ensuring that all data sent is encrypted while it travels across the internet.

3. Performance: Although Exim is highly customizable, it remains fast and efficient. It also can handle large amounts of email data, maintaining a good balance between adaptability and performance.

Although Exim is maintaining its position by being one of the most flexible and good-performing MTAs, the high learning curve and the complexity of its configuration can pose challenges, particularly for those less familiar with mail server management. The extensive range of configurable options, while powerful, requires a significant understanding of mail systems to avoid misconfigurations, potentially leading to security vulnerabilities and difficulties in the deployment of the systems.
To sum up, Exim is a powerful tool in the hands of an experienced system administrator. It could be exceptional in environments where detailed, specific configurations are necessary to meet unique operational demands. While it demands a higher set of technical skills, the level of customization it offers makes it a strongly demanded solution for complex email delivery scenarios.

3.2.3 Sendmail

Sendmail[25] is one of the oldest mail transfer agents (MTAs) available for Unix-based systems. It was developed in the early 1980s by Eric Allman at UC Berkeley and has since become a fundamental component in many email systems worldwide[26]. It was mostly widely used in the past while other MTAs like Postfix and Exim haven't yet developed although, despite their emergence, Sendmail's extensive functionality and powerful yet complex configuration capabilities still make it relevant in many enterprise environments.
The core features of Sendmail that are praised are explained below:

1. Configurability: Sendmail is known for its highly configurable nature, allowing detailed control over mail routing and processing. It uses a configuration file known as sendmail.mc, which is famous for its complexity but offers major control over every aspect of mail handling.

2. Alias creation and Forwarding Capabilities: One of the historical strengths of Sendmail is its robust support for aliases and mail forwarding, a critical feature in managing email redirection and handling user email routes within an organization. This allows for the easy creation of email addresses that can redirect to one or more other addresses, an essential function for businesses and educational institutions.

3. Performance: While Sendmail is capable of handling large volumes of email, its performance can be affected by the complexity of its configuration. The fine-tuning of the configuration is essential to optimize Sendmail's performance, especially in high-load environments.

4. Flexibility in Routing: Sendmail stands out for its exceptional routing capabilities, which allow administrators to define extremely precise mail handling rules. This level of flexibility is enabled by its Milter (mail filter) interface, which facilitates the customization of mail processing through various filters. These can modify or act upon messages as they are being processed, providing unparalleled control that is particularly valuable in complex networking environments.

On the other hand, Sendmail[27] has fallen behind compared to other MTAs. The reasons behind its downfall could be explained due to its significant configuration complexity, as its intricate macro language can lead to daunting setup processes and potential misconfigurations, heightening security risks. Historically plagued by notable security vulnerabilities, Sendmail has a history of significant security vulnerabilities, demanding careful management and frequent updates to protect against threats, which can be resource-intensive. Furthermore, its adaptability to modern IT environments is constrained by these legacy issues and the need for extensive customization, making it less appealing for organizations seeking straightforward, agile email solutions. These factors combine to make Sendmail a less favorable option for new installations, particularly when ease of use and quick deployment are prioritized.
To summarize, Sendmail serves as a testament to the evolution of email infrastructure technology. While it offers powerful capabilities for those who need intricate control over email handling, the demands of modern email security and ease of use have led many to adopt newer, more user-friendly MTAs. Despite this, Sendmail remains a critical piece of the internet's email infrastructure, particularly for legacy systems.

3.2.4 Comparison of the MTAs

When it comes to the choice among Postfix, Sendmail, and Exim[28], it should be guided by specific organizational needs and the technical expertise of the administrative team. Postfix is recommended for those valuing simplicity and security, making it ideal for most modern applications. Sendmail, though powerful, suits niche cases where its complex capabilities are necessary and can be securely managed. Exim serves well in scenarios where flexibility is needed, but not at the cost of excessive complexity. According to this analysis, while each MTA has a specialty, evolving security concerns and the need for easier management are leading many to prefer solutions like Postfix and Exim over the more traditional Sendmail which tends to be favored where legacy systems exist.

3.3 Dovecot

Dovecot [29, 30], a pivotal component in modern email infrastructures, is an open-source email server software renowned for its prowess in providing robust and efficient access to electronic messages. Dovecot specializes in offering both Internet Message Access Protocol (IMAP) and Post Office Protocol (POP3) services. IMAP enables users to access and manage their emails directly on the mail server, maintaining synchronization across multiple devices. POP3, on the other hand, allows users to download emails to their local devices. Dovecot seamlessly integrates with Mail Transfer Agents (MTAs) like Postfix to handle the storage and retrieval aspects of email communication. Its modular architecture allows for straightforward integration with various authentication mechanisms, including Lightweight Directory Access Protocol (LDAP), enhancing security and simplifying user management. With a commitment to high performance, scalability, and support for industry standards, Dovecot stands as an indispensable companion to MTAs, contributing to the efficient and secure delivery of electronic messages in contemporary email systems. It is also designed to handle a high volume of concurrent connections and efficiently manage large mailboxes. These features make Dovecot suitable for both small-scale deployments and enterprise-level email systems with diverse user requirements.

3.4 Encryption and Certificates

Encryption and certificates play a fundamental role in securing communications across networks, especially in environments where sensitive information is transmitted, such as in email systems. By the term ”Encryption” we are referring to the process of encoding messages or information in such a way that only authorized parties can access it. Certificates, issued by trusted entities known as Certificate Authorities (CAs), help establish the identity of parties and enable secure connections, effectively proving that the public keys held by entities are authentic and have not been tampered with. These technologies are crucial for maintaining the confidentiality and integrity of data as it traverses the internet.[31]

3.4.1 Let’s Encrypt

Let’s Encrypt[32] is the authority of the certificate that I used in my Thesis. It provides free SSL/TLS certificates, significantly simplifying the process of securing web communications. It is particularly instrumental in promoting the widespread adoption of encrypted connections on the internet. Operated by the Internet Security Research Group (ISRG), Let’s Encrypt automates the process of certificate issuance, renewal, and revocation, thereby removing manual complexities and reducing the cost barriers associated with securing communication channels. By offering these services at no charge, Let’s Encrypt not only encourages but also enables website administrators and email server operators to implement encryption easily. This democratization of security services aligns closely with the broader shift towards universal secure communication, further enhancing the trust and integrity of data transmitted across the internet. Through its support for automatic renewals and its easy integration with existing servers and software, Let’s Encrypt plays a critical role in ensuring that SSL/TLS protection is more accessible and maintainable, thus contributing to a safer internet ecosystem.

3.5 Email Filtering and Scanning

3.5.1 ClamAV

ClamAV, or Clam AntiVirus, [33, 34, 35] is an open-source antivirus tool specifically designed to secure network gateways and email servers from a plethora of malware threats, including viruses, worms, and Trojans. In the context of electronic mail management systems, the purpose of integrating ClamAV is multifaceted, aiming not only to protect the integrity of the data within the emails but also to safeguard the overall health of the network infrastructure.
Emails are a common vector for cyber threats, often used by attackers to distribute malicious software that can compromise individual user systems and corporate networks. The consequences of such breaches can range from simple nuisances to significant financial loss and severe damage to an organization's reputation. Hence, employing a robust antivirus solution like ClamAV is crucial for preventing malware from entering and spreading through the internal systems via email attachments or embedded links.
ClamAV's implementation in mail servers acts as a proactive measure to intercept and neutralize these threats before they reach the end user. By scanning incoming and outgoing messages for known malware signatures and anomalous patterns, ClamAV helps maintain a secure communication environment. This not only enhances the security posture of an organization but also builds trust among users who rely on the safety and confidentiality of their electronic communications.
Additionally, ClamAV's role extends to compliance with regulatory requirements that mandate the protection of sensitive information against unauthorized access and data breaches. Organizations, especially those in regulated industries such as finance, healthcare, and public services, are often required to implement stringent security measures, including effective malware detection and prevention strategies. ClamAV, with its comprehensive scanning capabilities and regular updates, fulfills these requirements, ensuring that organizations adhere to legal and regulatory standards concerning data security.
In summary, the integration of ClamAV into email management systems is essential for preventing the dissemination of malware, ensuring compliance with security regulations, and maintaining the trustworthiness and reliability of electronic communications. As a result, ClamAV not only serves the technical purpose of scanning and detecting malware but also plays a strategic role in an organization's overall cybersecurity defense strategy.
ClamAV operates by scanning attachments and links within emails for signatures and heuristics that are known to be associated with malware. Here's a detailed look at its functionality:

1. Signature-Based Detection: ClamAV utilizes a traditional signature-based approach to detect known malware. This method relies on a database of unique identifiers for various malware strains. The database is regularly updated to include new malware signatures discovered by cybersecurity researchers worldwide. When an email comes in, ClamAV scans the contents against this database to identify any known threats.

2. Heuristic Analysis: Beyond simple signature matching, ClamAV also employs heuristic analysis to detect new or unknown viruses. This technique involves examining the behavior of an email attachment or a linked file to predict its potential for malicious activity based on irregular patterns or suspicious characteristics. Heuristic analysis helps in catching malware that has not yet been officially cataloged.

3. File Format Support: One of the strengths of ClamAV is its broad support for various file formats, which is critical in an email context where attachments can come in many forms—from PDFs and Microsoft Office documents to executables and archived files. This versatility ensures that the antivirus can scan a wide array of attachment types for potential threats. Integration with Mail Servers: ClamAV is designed to integrate seamlessly with mail servers through its command-line interface, which can be used with filtering tools like Amavis (A Mail Virus Scanner). This integration allows ClamAV to scan incoming and outgoing emails automatically, providing real-time protection against the transmission of viruses.

4. Open Source Advantage: As an open-source project, ClamAV benefits from the support of a global community of developers who contribute to its codebase, ensuring that the antivirus is not only free but also continually evolving. Open-source software typically offers greater transparency in its operations, allowing users to verify the security and integrity of the software they are relying on.

In conclusion, ClamAV's role in an electronic mail management system is to serve as a first line of defense against email-borne malware, leveraging both established and innovative virus detection techniques to provide comprehensive protection. Its ability to integrate with existing mail server configurations and support a wide range of file types makes it an indispensable component in the architecture of modern email security solutions.[36]

3.5.2 SpamAssasin

SpamAssassin[37] is a widely used open-source email filtering tool that helps identify spam. It is designed to assist email administrators in preventing unwanted and potentially harmful emails from reaching user inboxes. As spam continues to evolve in complexity and volume, tools like SpamAssassin are crucial for maintaining the cleanliness and integrity of email communications.
The primary purpose of SpamAssassin is to reduce the volume of unsolicited email (spam) by rigorously analyzing incoming email messages for spam characteristics. Spam not only includes unsolicited and unwanted advertisements but also phishing attempts and other malicious content. By filtering out such emails, SpamAssassin helps protect users from potential scams, malware, and other security threats posed by spam. Moreover, reducing spam improves email server efficiency and user productivity by ensuring that users spend less time managing and deleting irrelevant emails.
SpamAssassin employs a variety of mechanisms to detect spam, which include:

1. Header Analysis: SpamAssassin analyzes the headers of incoming emails, looking for discrepancies or typical spam indicators such as misleading subject lines or addresses. Text Analysis: It analyzes the text of the email to look for known spam indicators, which include certain phrases commonly found in spam emails.

2. Bayesian Filtering[38]: This is one of the core features of SpamAssassin. It uses Bayesian statistics to dynamically learn from both the contents of spam and non-spam emails to improve its filtering accuracy over time. Users can train their SpamAssassin installation to recognize what is considered spam and what is not, making the filtering process highly adaptive and personalized.

3. Blacklists and Whitelists: SpamAssassin checks emails against various DNS-based blacklists like Spamhouse, SORBS, and CBL and whitelist databases to assess the reputation of the sender's IP address. Emails from blacklisted IPs are more likely to be flagged as spam, whereas those from whitelisted IPs are generally considered safe.

4. Third-Party Tests: SpamAssassin incorporates tests that use external services to evaluate if content or links within messages are related to known spam.

5. Spam Score Threshold: After evaluating an email through various tests, SpamAssassin assigns a spam score to each email based on the number of failed tests. Administrators can set a threshold score that determines whether an email is flagged as spam or allowed through to the recipient.

SpamAssassin plays a critical role in the management of electronic mail by reducing the burden of spam on both users and email infrastructure. Its continued development and widespread adoption underscore its effectiveness and necessity in combating the evolving threats associated with spam in the digital age.[39]

3.6 Authentication Technologies

Authentication mechanisms in mail servers are fundamental to ensuring that only authorized users can access and send email securely. These systems utilize a variety of protocols and techniques to verify the identity of users and maintain the integrity of communications. Commonly implemented methods include password-based authentication, digital certificates, and more sophisticated protocols such as LDAP (Lightweight Directory Access Protocol) and SASL (Simple Authentication and Security Layer). Each method provides a layer of security that helps protect against unauthorized access and potential threats, making the authentication process a critical component in the overall security architecture of email systems. These mechanisms not only safeguard user data but also enhance trust in email communication as a secure and reliable medium for personal and business exchanges.

3.6.1 SASL

SASL[40] is a protocol used to add authentication support to connection-based protocols through a pluggable, decoupled method. It abstracts the process of authentication from application protocols, allowing flexibility and versatility in its implementation. SASL is used in various Internet protocols including LDAP, SMTP, and IMAP, serving as a critical component in secure data exchange across network services, particularly in email systems.
SASL works by negotiating a choice of authentication mechanisms between clients and servers, depending on what both support. This negotiation allows the protocol to accommodate a wide range of authentication methods without requiring each application to implement these mechanisms independently. SASL supports mechanisms ranging from plain username and password to more sophisticated methods like Kerberos or client certificates, enhancing the flexibility and security of authentication processes. The primary purpose of SASL is to ensure authentication integrity and confidentiality. It can be used in conjunction with encryption protocols such as SSL/TLS to provide a layer of security that authenticates users and encrypts the data in transit. This dual-layer security is crucial in environments that handle sensitive or personal data, ensuring that only authenticated users can access services and that their data cannot be intercepted or tampered with during transmission.

Within email systems, SASL enhances the capability of SMTP, IMAP, and POP3 protocols by providing a secure method for client-server authentication. This is particularly important to prevent unauthorized access and to safeguard the confidentiality and integrity of email communications. By integrating SASL, email servers can offer a more robust security framework that supports various authentication methods, catering to diverse client capabilities and security requirements.
One of the challenges with SASL is the complexity of managing multiple authentication mechanisms, especially in heterogeneous environments with diverse client capabilities and security needs. Additionally, the security of SASL-dependent systems heavily relies on the correct configuration and implementation of both SASL itself and its associated authentication mechanisms. Misconfigurations can lead to vulnerabilities, potentially exposing systems to unauthorized access or data breaches.
In conclusion, SASL is an essential protocol for modern email servers, facilitating secure and flexible authentication. Its ability to support a wide array of authentication methods makes it adaptable to various security policies and user environments. By decoupling authentication from application protocols, SASL not only simplifies the integration of security features into existing systems but also enhances the overall security posture by enabling comprehensive, multi-layered protection strategies.

3.6.2 LDAP

LDAP (Lightweight Directory Access Protocol)[41] is an application protocol used for accessing and maintaining distributed directory information services over an IP network. It enables organizations to centralize the management of user information, which LDAP directories can store in a structured and hierarchical manner, akin to a phone directory. This protocol facilitates querying and modifying the users' directory effectively and securely, making it an indispensable tool for managing authentication in complex systems, including mail servers.
LDAP operates on a client-server model, where the client makes a request, and the server responds with the information or confirmation of an update. The protocol supports a variety of operations that allow users to search for directory entries, create and delete entries, and modify entry attributes. It is designed to work over TCP/IP, which ensures reliability in communications. LDAP uses a simple string format for data interchange, known as Distinguished Names (DN), which uniquely identifies each entry in the directory.
In terms of security, LDAP supports multiple authentication methods. The most basic is anonymous authentication, which allows minimal access to the directory for public information. More secure is simple authentication, which transmits a username and password in clear text (thus requiring a secure connection such as via SSL/TLS). LDAP also supports SASL mechanisms that provide additional security features, including encryption and multi-factor authentication, for protecting communication between clients and the server.[42]
In mail server environments, LDAP is primarily used to store and retrieve user information, including email addresses, mail preferences, and authentication credentials. This usage allows email systems to scale efficiently by centralizing user management and authentication processes. LDAP can be integrated with other security protocols, like SSL/TLS, to ensure secure access to directory data, crucial for maintaining the confidentiality and integrity of user data in email transactions. While LDAP offers numerous advantages in terms of flexibility and scalability, it also presents challenges, particularly in the areas of security and management. LDAP directories can be targets for attacks if not properly secured with strong encryption and authentication measures. Additionally, the management of LDAP servers, especially in large organizations or those with high volumes of directory access transactions,

can become complex and resource-intensive.[43]
As it stands up, LDAP is a strong protocol for handling remote directory data, and it works especially well in settings like email servers that demand efficient user management and authentication. It is a crucial part of contemporary IT architecture due to its scalability, versatility in handling complicated data, and ability to connect with other security mechanisms.

Chapter 4

Creating the Mail server

4.1 Introduction

This chapter will delve into the practical application of the technologies discussed in the previous sections, focusing on how they have been implemented to enhance the security and efficiency of the mail server setup. I will explain the configuration of the core technologies such as Postfix and Dovecot and the setup of the DNS zone with records such as MX, SPF, DKIM, and DMARC which are essential for the functionality of the mail server. Also, I will present the way of implementing mail filtering technologies such as SpamAssassin and I will demonstrate how I managed to issue the Let's Encrypt SSL certificate. Lastly, I will explain the implementation of LDAP and SASL authentication. Each of the components plays a pivotal role in ensuring the robustness and reliability of the service, addressing specific needs from secure email transit to effective spam filtering.
The implementation process is detailed through comprehensive code snippets and configurations that illustrate the integration and customization of these technologies. The presentation will come gradually, starting with the presentation of the operating system on which the email server is based and then adding its components. For each component, we discuss the rationale behind configuration choices, challenges encountered during setup, and the solutions devised to overcome these hurdles. This practical exposition aims to provide insights into the technical nuances that are often the linchpins of successful system architecture. Finally, I will detail my attempts to configure the designated mail server to support multiple domains, rather than a single one. Despite the ultimate failure of this endeavor, the process proved to be a valuable learning experience. It offered me significant insights into the complexities involved in expanding the functionality of this specific mail server. This experience has not only enhanced my understanding of mail server development but also prepared me for addressing similar challenges in future projects.
Through this detailed presentation, the chapter seeks to bridge the gap between theoretical knowledge and practical application, providing a clear view of how abstract concepts are transformed into concrete, working solutions in real-world scenarios.

4.2 Tools and Technologies

In this section, I will provide an overview of the various tools, and technologies, employed to set up and implement in my project. These components have been carefully chosen to ensure seamless

integration, efficient development, and optimal Security of the Mail server.

4.2.1 Infrastructure

The infrastructure for this thesis project was established using a virtual machine environment to ensure both flexibility and control over the experimental setup. The chosen operating system for the virtual machine was Debian 11 (Bullseye) GNU/Linux, selected for its stability, security, and robust community support. Deployed within the secure confines of the university's network, behind its robust firewall, this setup provided an added layer of security and network control, crucial for safeguarding the mail server from external threats. Debian 11, known for its conservative approach to updates and proven reliability, provided a solid foundation for deploying the various email-related technologies explored in this thesis. The use of a virtual machine facilitated a contained and controlled environment, allowing for precise configuration adjustments and straightforward replication of the setup, which is crucial for testing and validation purposes. This setup not only ensured that the environment was isolated from external inconsistencies but also enabled easier scaling and management, pivotal for handling the complexities of a mail server configuration and ensuring accurate performance assessments under controlled conditions.

4.2.2 Languages

The configuration of both the Mail Transfer Agent (MTA) and Mail Delivery Agent (MDA) within our mail server infrastructure does not involve traditional programming languages. Instead, these systems are configured through directive-based instructions specific to each software application. These directives, and parameters set in configuration files, dictate how the software should operate under various conditions.

4.2.3 Debug and log files

In the development and maintenance of a mail server on Debian 11, effective debugging is crucial for ensuring reliable operation and troubleshooting issues. Debian and its mail server applications, such as Postfix and Dovecot, utilize a comprehensive logging system that records various events and errors into separate log files. These files are invaluable for diagnosing problems, monitoring performance, and ensuring security compliance. Postfix stores its logs to ***/var/log/mail.log*** for general mail activities, ***/var/log/mail.err*** for recording error messages, and /var/log/mail.info for informational messages. Dovecot, on the other hand, primarily logs to ***/var/log/dovecot.log***, where details about the handling of IMAP connections are stored. Additionally, system-wide messages, including those related to mail services, can be found in ***/var/log/syslog***.

4.3 Adding the frame of the mail server

4.3.1 Installing Postfix

First of all, the initial step in setting up our mail server involves installing the Postfix package, a crucial component that acts as the Mail Transfer Agent (MTA).

```
root@mailtest:~# sudo apt-get update
root@mailtest:~# sudo apt-get install postfix
```

Listing 4.1: Postfix installation

After that step, a Setup wizard appeared from which I chose the default option ”Internet Site”. This will provide us with an initial configuration file for Postfix which I will later edit.

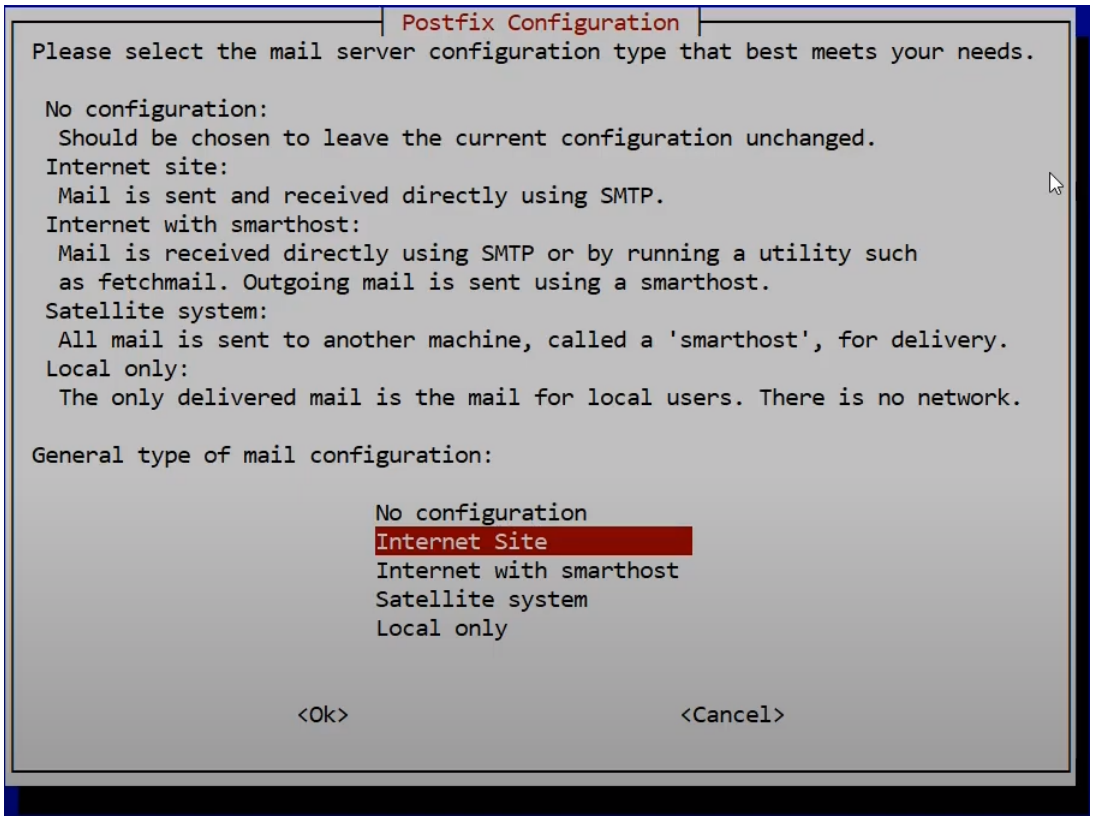


Figure 4.1: Postfix Wizard

After choosing this option another window appears to set the domain name the mail server will host.

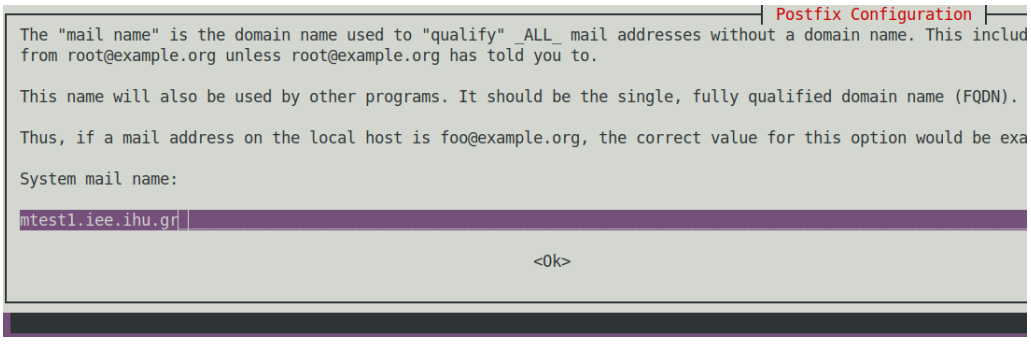


Figure 4.2: Setting the domain

4.3.2 Installing Dovecot

```
root@mailtest:~# sudo apt-get install dovecot-common dovecot-imapd
```

Listing 4.2: Dovecot installation

The Dovecot-common package includes common files and components needed by both the core Dovecot service and its various sub-packages. The ***Dovecot-imapd*** package specifically installs the IMAP daemon part of Dovecot. This package allows Dovecot to act as an IMAP server allowing clients to read messages without downloading them directly to their local device.

4.3.3 Maildir format

In the var/log/main.cf file I am adding the following lines of configuration.

```
home_mailbox = Maildir/
mailbox_command =
```

Listing 4.3: Enabling Maildir

The first line instructs Postfix to use the Maildir format to store emails. Unlike the traditional mbox format, which stores all emails in a single file, Maildir creates a unique file for each email.
In the second line keeping no value to the command ensures that Postfix will not attempt to use any external command or program to deliver mail to the mailbox. Instead, it will directly place the emails into the specified Maildir directory. This configuration simplifies the delivery process and reduces the potential for misconfigurations or security issues that could arise from using external mailbox delivery commands.

```
root@mailtest:~# sudo maildirmake.dovecot /etc/skel/Maildir/
root@mailtest:~# sudo maildirmake.dovecot /etc/skel/Maildir/.Drafts/
root@mailtest:~# sudo maildirmake.dovecot /etc/skel/Maildir/.Sent/
root@mailtest:~# sudo maildirmake.dovecot /etc/skel/Maildir/.Spam/
root@mailtest:~# sudo maildirmake.dovecot /etc/skel/Maildir/.Template/
root@mailtest:~# sudo maildirmake.dovecot /etc/skel/Maildir/.Trash/
```

Listing 4.4: Creating the Maildir structure in skel

Executing those commands sets up the initial directory structure necessary for storing emails in the Maildir format under /etc/skel/Maildir/. This structure will then be replicated in the home directory of each new user created on the system, enabling Dovecot to handle their emails efficiently. This preparation is crucial in multi-user environments where each user will receive and manage their own email independently.
The last setting in the Maildir setup is to tell Dovecot about the location of the mailbox. I managed this by adding the following rule and commenting out the previous ***mail_location*** in the ***/etc/dovecot/conf.d/10-mail.conf*** file.

```
mail_location = maildir:~/Maildir
```

Listing 4.5: Setting Mail_location

4.4 Securing the mail server

4.4.1 Hardening Rules

Another crucial aspect of the mail server configuration is adding hardening rules to prevent malicious users from gaining access and sending Emails. In Listing 4.6 I will explain the hardening rules I used in order to achieve this which were added in the ***/etc/postfix/main.cf*** file.

```
smtpd_recipient_restrictions =
        permit_sasl_authenticated,
        permit_mynetworks,
        reject_unauth_destination,
        check_policy_service unix:private/policyd-spf
```

Listing 4.6: The recipient_restrictions setting

The ***smtpd_recipient_restrictions*** parameter is used to define rules that the server follows to decide whether to accept, reject, or defer an email based on the characteristics of the email's recipient. In the second line, rule ***permit_sasl_authenticated*** allows emails from clients that have successfully authenticated themselves to the server using the SASL authentication mechanism. The rule ***permit_mynetworks*** ensures that the mail server will permit emails from IP addresses that are trusted and which are defined in the ***mynetworks*** parameter in the main.cf configuration. The ***reject_unauth_destination*** rule is a very important directive that tells Postfix to reject emails if the destination address does not match any domain hosted on the server. This protects the mail server from acting as a relay Server thus sending unsolicited emails to external domains. Finally, the setting ***check_policy_service unix:private/policyd-spf*** enables the SPF check on our mail server.

4.4.2 Enabling and configuring HELO

Another security aspect of the SMTP connections is enabling and configuring the HELO command. When an email client connects to a mail server, the first step in the SMTP interaction is for the client to introduce itself using either a HELO or an EHLO command followed by the client's Hostname. Enabling this enhances the server's ability to manage the incoming connections effectively and also prevents anonymous or malicious mail delivery attempts. In Listing 4.7 I will explain the configuration I added in the /etc/postfix/main.cf.

```
smtpd_helo_required = yes
smtpd_helo_restrictions =
        permit_mynetworks,
        permit_sasl_authenticated,
        reject_invalid_helo_hostname,
        reject_non_fqdn_helo_hostname,
        reject_unknown_helo_hostname,
        check_helo_access hash:/etc/postfix/helo_access
```

Listing 4.7: Enabling and configuring HELO

Setting the ***smtpd_helo_required*** directive to yes Enables the HELO process. Therefore we are able to add some restrictions into the HELO configuration by adding ***smtpd_helo_restrictions***. Starting by ***permit_mynetworks*** in order to allow any client within the predefined "mynetworks" to proceed without any further unnecessary HELO checks. The ***permit_sasl_authenticated*** allows clients that were

authenticated via SASL to proceed without the need for additional verification at the HELO stage. The directive ***reject_invalid_helo_hostname*** rejects the connections where the HELO hostname argument is invalid, such as incorrectly formatted hostnames. This enhances the Security of the mail server from poorly configured or potentially malicious SMTP clients. The ***reject_non_fqdn_helo_hostname***helps reject HELO commands that do not use a fully qualified domain name (FQDN). By adding the ***reject_unknown_helo_hostname*** directive the mail server rejects the HELO commands where the hostname cannot be resolved via DNS. This is important for verifying that the sender's hostname is valid and operational thus reflecting on a well-configured mail server. Finally, the ***check_helo_access hash:/etc/postfix/helo_access*** tells Postfix to consult a specific access file called (helo_access) which I created and I will explain later in order to allow or deny mail based on the HELO hostname provided by the client.
Following, the previous set of rules I created a file called ***/etc/postfix/helo_access*** that matches the path on the last line of Listing 4.8.

```
mtest1.iee.ihu.gr       REJECT  Sorry, den epitrepetai i prosvasi.
mail.mtest1.iee.ihu.gr  REJECT  Sorry, den epitrepetai i prosvasi.
```

Listing 4.8: Configuring helo_access

By adding the domain in the helo_access file I am making sure that no one can imitate the hostname of the mail server and the domain in order to send Spam. This was part of a loophole that avoids the server's checks because our domain is being treated as a valid domain that the server could use. Lastly, in order to make this file usable by postfix I used the command shown in Listing 4.9.

```
root@mailtest:~# postmap /etc/postfix/helo_access
```

Listing 4.9: Postmap command

The Postmap command maps the helo_access file compiling it into a hash database so Postfix knows its existence.

4.4.3 Enabling SASL authentication

In order to enable SASL authentication using Dovecot as the authentication service involves specifying a few directives in the ***/etc/postfix/main.cf*** file. In Listing 4.10 I will explain how these directives tell Postfix to interact with Dovecot in order to authenticate users.

```
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth = yes
```

Listing 4.10: Configuring main.cf for SASL

By setting ***smtpd_sasl_type*** to Dovecot, Postfix is instructed to use Dovecot's SASL implementation for authentication processes. The ***smtpd_sasl_path*** directive specifies ***private/auth*** as the location of the Unix domain socket through which Postfix communicates with Dovecot's authentication services, ensuring secure, local transmission of authentication data. Lastly, the directive ***smtpd_sasl_auth_enable = yes*** activates SASL authentication, requiring SMTP clients to authenticate successfully with Dovecot before emails are accepted, thus enhancing the mail server's security by preventing unauthorized email sending and relay. These three directives work together in order to establish a secure, authenticated communication channel between Postfix and Dovecot.

Following the configuration of Postfix, in order to enable SASL authentication on dovecot I added a new service auth block in ***/etc/dovecot/conf.d/10-master.conf*** file which I am showing in the Listing 4.11.

```
service auth {
        unix_listener /var/spool/postfix/private/auth {
                mode = 0660
                user = postfix
                group = postfix
        }
}
```

Listing 4.11: Configuring 10-master.conf for SASL

The service auth block is used to define how authentication services should interact securely with Postfix. Firstly the unix_listener is specified at /var/spool/postfix/private/auth, which is the path to a Unix domain socket used by Postfix for secure communications with Dovecot. The permissions of the socket are set with mode = 0660, allowing read and write access only to the user and group defined as postfix. This setting ensures that only the Postfix and Dovecot processes have access to the authentication socket, preventing unauthorized access and improving the security of the mail server. The final step in enabling and setting up SASL authentication is to add the configuration in the ***/etc/dovecot/conf.d/10-auth.conf*** which I will show you in the Listing 4.12.

```
auth_mechanisms = plain login
disable_plaintext_auth = yes
```

Listing 4.12: Configuring 10-auth.conf for SASL

The setting ***auth_mechanisms = plain login*** specifies the authentication mechanisms available for users who try to connect to the mail server. The ”plain” mechanism transmits the username and password as plain text. The ”login” mechanism is similar but uses a slightly different protocol exchange. By using both those options, Dovecot allows clients to choose the mechanism they support, enhancing compatibility across different email clients. The The directive ***disable_plaintext_auth = yes*** ensures that authentication with plain text (without encryption) is disabled. This directive makes sure that Dovecot requires the clients' connection to be encrypted before it accepts any authentication attempts in plain text form. This is a critical security feature that protects user credentials from man-in-the-middle attacks.

4.4.4 Issuing SSL certificate using Certbot

One key security feature is the issuing and installation of an SSL certificate in the mail server domain. To achieve that I used the Certbot[44] tool. In Listing 4.13 I point to the command to install this tool.

```
root@mailtest:~# sudo apt install certbot
```

Listing 4.13: Certbot installation

In order to issue the SSL I used the command in Listing 4.14.

```
root@mailtest:~# sudo certbot certonly --standalone -d mail.mtest1.iee.ihu.gr
```

Listing 4.14: SSL issuing

In this command, the ***certonly*** option helps to obtain the certificate but not install so it gives the opportunity to manually handle its configuration on the server. Using the ***–standalone*** plugin tells Certbot to operate its own temporary web server for the purpose of completing the ACME challenge[45], which verifies domain ownership. The challenge is completed by a correct response to a request made in port 80. Finaly, ***-d mail.mtest1.iee.ihu.gr*** specifies the domain in which the SSL is going to be issued.
After issuing the certificate, its files are stored in a specific path in the mail server. It is crucial to configure Postfix and Dovecot on where those files are stored, so the SSL certificate can be used by those services.
In order to configure Postfix accordingly I added the directives shown in Listing 4.15 in the ***/etc/postfix/main.cf***.

```
smtpd_tls_cert_file=/etc/letsencrypt/live/mail.mtest1.iee.ihu.gr/fullchain.pem
smtpd_tls_key_file=/etc/letsencrypt/live/mail.mtest1.iee.ihu.gr/privkey.pem
```

Listing 4.15: Postfix SSL path

The directive ***smtpd_tls_cert_file*** the path where the SSL/TLS certificate is stored while the directive ***smtpd_tls_key_file*** specifies where the private key is stored on the mail server.
In order to configure Dovecot I edited the file ***/etc/dovecot/conf.d/10-ssl.conf***.

```
ssl_cert = </etc/letsencrypt/live/mail.mtest1.iee.ihu.gr/fullchain.pem
ssl_key = </etc/letsencrypt/live/mail.mtest1.iee.ihu.gr/privkey.pem
```

Listing 4.16: Dovecot SSL path

Those two directives show Dovecot the path for the SSL certificate as well as the private key.

4.4.5 Enabling STARTTLS

Another key aspect of the security of the mail server is to enable TLS encryption both on incoming and outgoing mail traffic. The directives I used to enable TLS encryption are presented in Listing 4.17 and were added in the ***/etc/postfix/main.cf*** file.

```
smtp_use_tls = yes
smtpd_use_tls = yes
smtpd_tls_received_header = yes
```

Listing 4.17: Enabling STARTTLS

The directive ***smtp_use_tls = yes*** tells Postfix to start a TLS session for outgoing connections to other mail servers, ensuring that the email contents are encrypted during transit. Similarly but for incoming connections, this setting ***smtpd_use_tls = yes*** instructs Postfix to accept TLS connections from SMTP clients. This means that when other servers or email clients connect to the Postfix server to send email, they can use TLS to secure the connection. Finally, the directive ***smtpd_tls_received_header = yes*** adds a "***Received***" header in emails that details the TLS status of the connection over which the email was received. This header provides information about the security level of the incoming connection, including the version of TLS used and the cipher suite.

4.5 Spam prevention

4.5.1 Installing SpamAssassin

SpamAssassin is an essential tool that helps to reduce unwanted email. It effectively decreases the volume of spam emails and helps maintain the cleanliness and usability of email inboxes. The following command can be used for the installation of the SpamAssassin package.

```
root@mailtest:~# sudo apt-get install spamassassin
```

Listing 4.18: Installing SpamAssassin

4.5.2 Configuring SpamAssassin

In order to configure SpamAssassin the first thing to do is to add the directives I mention in Listing 4.19 into ***/etc/spamassassin/local.cf*** file.

```
rewrite_header Subject *****SPAM*****
report_safe 0
use_bayes 1
bayes_auto_learn 1
```

Listing 4.19: Configuring SpamAssassin

The directive ***rewrite_header Subject *****SPAM******* alters the subject part of any email that has been identified as Spam adding the sign "********SPAM*******" at the start making it easier to be recognized. The directive ***report_safe 0*** is responsible for the way SpamAssassin delivers spam emails. With the value 0, SpamAssassin delivers spam emails as they are, marking them as spam but not altering the way the content is presented. This mechanic ensures that the email, including any malicious content, is not hidden in attachments or altered in ways that might obscure its original intention. The next two settings are referring to the Bayesian filtering. The first Setting ***use_bayes 1*** enables the Bayesian filtering technique while the second setting ***bayes_auto_learn 1*** enables the automatic learning process for the Bayesian filter in order to train it from the feedback of the receiving Mail traffic that filtered as Spam.

4.5.3 Enabling SpamAssassin Service

In order to ensure that SpamAssassin is always running, we have to enable it so it starts automatically during the system's boot process. This can happen by using the commands shown in the Listing 4.20.

```
root@mailtest:~# update-rc.d spamassassin enable
root@mailtest:~# service spamassassin start
```

Listing 4.20: Enabling SpamAssassin Service

4.5.4 Telling Postfix to use SpamAssassin

The last step to successfully set up the SpamAssassin tool in the mail server is to configure the Postfix service in order to integrate it as a content filter. To achieve this I added the settings shown in Listings 4.21 and 4.22 in the ***/etc/postfix/master.cf*** file.

```
smtp      inet  n       -       -       -       -       smtpd
    -o content_filter=spamassassin
```

Listing 4.21: Implementing SpamAssassin

This configuration line tells Postfix to run an SMTP server that listens for incoming mail and uses SpamAssassin to filter all incoming emails for spam. The ***content_filter*** option hooks the SMTP server directly to SpamAssassin, ensuring that every email is checked for spam characteristics based on SpamAssassin's rules and settings.

```
spamassassin    unix  -       n       n       -       -       pipe user=debian
   -spamd argv=/usr/bin/spamc -f -e /usr/sbin/sendmail -oi -f ${sender} ${
   recipient}
```

Listing 4.22: Postfix SpamAssassin

This configuration directs Postfix to pass all emails through SpamAssassin for spam filtering using spamc. After SpamAssassin processes the email, it is passed back to Postfix's sendmail for delivery, using the sendmail interface but with modified content as per SpamAssassin's analysis. In order to guarantee that all incoming and outgoing emails are checked for spam prior to final delivery, this configuration is necessary to easily integrate spam filtering into the mail delivery process.

4.5.5 Installing Dovecot's LMTP and sieve deamons

One of Dovecot's daemons called LMTP stands for Local Mail Transfer Protocol is used for the delivery of emails to the assigned mailbox. This daemon has a unique usage called sieve. In sieve, we can write a set of rules in order to Sort and filter emails delivering them to the appropriate folder. To install it I used the command on Listing 4.23.

```
root@mailtest:~# apt-get install dovecot-lmtpd, dovecot-sieve
```

Listing 4.23: Installing dovecot-lmtpd

4.5.6 Configuring Dovecot LMTP

In order to tell Dovecot to use the LMTP protocol, there are some configuration adjustments that need to be done in the file ***/etc/dovecot/dovecot.conf*** which I show in the Listing 4.24.

```
protocols = imap lmtp
```

Listing 4.24: Configuring LMTP on dovecot.conf

This line specifies which protocols Dovecot should enable and support on the mail server. With this configuration, Dovecot can handle both the retrieval of emails by clients via IMAP and the receipt of incoming emails from Postfix via LMTP.
Another configuration needs to be done in the file ***/etc/dovecot/conf.d/20-lmtp.conf*** is explained in Listing 4.25.

```
lmtp_save_to_detail_mailbox = yes
protocol lmtp {
  mail_plugins = $mail_plugins sieve
  postmaster_address = postmaster@mtest1.iee.ihu.gr
```

```
}
```

Listing 4.25: Configuring 20-lmtp.conf

The directive ***lmtp_save_to_detail_mailbox = yes*** tells Dovecot's LMTP service to save emails to the recipient's detailed mailbox instead of just the primary mailbox. This is particularly useful when the server handles multiple domains and the service has to respect the full recipient address, not just the username of the email address.
The ***protocol lmtp*** block, starts the configuration that is specific to the LMTP protocol, setting up various rules and plugins that will only apply to the LMTP service. Inside the block, the directive ***mail_plugins = $mail_plugins sieve*** specifies that the sieve plugin should be loaded along with any other plugins already specified in the variable $mail_plugins.The Sieve plugin is used for filtering and sorting emails according to the set of rules it is already configured. Finally, the postmaster This setting specifies the email address for the postmaster of the server. The directive ***postmaster_address = postmaster@mtest1.iee.ihu.gr*** is used to set the account postmaster@mtest1.iee.ihu.gr in order to receive the automated email responses generated by the server, such as non-delivery receipts or other system messages. It is crucial to add an address for administrative purposes in order to handle errors and administrative alerts in a mail system.
Continuing with the configuration of the LMTP service in the ***/etc/dovecot/conf.d/10-master.conf*** file.

```
service lmtp {
  unix_listener /var/spool/postfix/private/dovecot-lmtp {
    mode = 0666
  }
}
```

Listing 4.26: Configuring LMTP on 10-master.conf

The ***service lmtp*** block is used to define the handling of the lmtp service in Dovecot and sets up a Unix socket for communication between Dovecot and Postfix. The ***unix_listener /var/spool/postfix/private/dovecot-lmtp*** directive sets up a Unix domain socket listener at the path /var/spool/postfix/private/dovecot-lmtp while the line ***mode = 0666*** sets the permissions of the Unix socket so Postfix can communicate with Dovecot through it.
Another file that needs to be configured accordingly in order to support the LMTP protocol is the /etc/postfix/main.cf.

```
mailbox_transport = lmtp:unix:private/dovecot-lmtp
```

Listing 4.27: Configuring LMTP on main.cf

By setting this directive, Postfix is instructed to hand off emails to Dovecot via LMTP using a Unix domain socket for the final delivery phase to users' mailboxes.

4.5.7 Creating a Script for Sieve

In the configuration of mail servers, it is crucial to implement effective spam management to ensure that unwanted emails are correctly filtered and organized. As part of this effort, adding the Sieve script of Listing 4.28 to the ***/home/jimouroudelis/.dovecot.sieve*** file provides an automated mechanism to handle spam emails.

```
require ["fileinto"];
if header :contains "X-Spam_Flag" "YES" {
  fileinto "Spam";
  stop;
}
```

Listing 4.28: Making a .dovecot.sieve script

The primary goal of this script is to automatically move emails marked as spam into a designated ”Spam” folder, ensuring that the user's inbox remains uncluttered and free of unsolicited messages. In order to achieve this check the headers of incoming emails for the presence of ***X-Spam-Flag*** with the value ”YES”. If the condition is met, the ***fileinto*** action moves the identified spam email into the ”Spam” folder. Finally, the ***stop*** command ensures that no further processing is done on the email once it has been moved to the ”Spam” folder. Without this script, users would have to manually identify and move spam emails, which is inefficient and could lead to missed legitimate emails or retained spam.

4.6 Implementation of Email Authentication and Security Protocols

4.6.1 Setting FQDN

FQDN stands for the fully qualified domain name. Setting it for the EHLO greeting is a crucial step in configuring the mail server especially when it comes to outgoing email traffic. It ensures that the mail server is uniquely identified, making it harder for malicious actors to spoof its identity. Moreover, spam filters analyze the EHLO greeting as part of their spam detection algorithms. An FQDN that resolves correctly via DNS and matches the server's IP address enhances the server's reputation, making it less likely that emails will be flagged as spam. In order to successfully set up FQDN I added the directives shown in Listing 4.29 in the ***/etc/postfix/main.cf*** file.

```
myhostname = mail.mtest1.iee.ihu.gr
smtpd_banner = $myhostname
```

Listing 4.29: Setting up FQDN

The directive ***myhostname = mail.mtest1.iee.ihu.gr*** sets up the hostname ***mail.mtest1.iee.ihu.gr*** on the mail server. The directive ***smtpd_banner = $myhostname*** uses ***$myhostname*** variable to dynamically include the server's FQDN in the greeting banner.

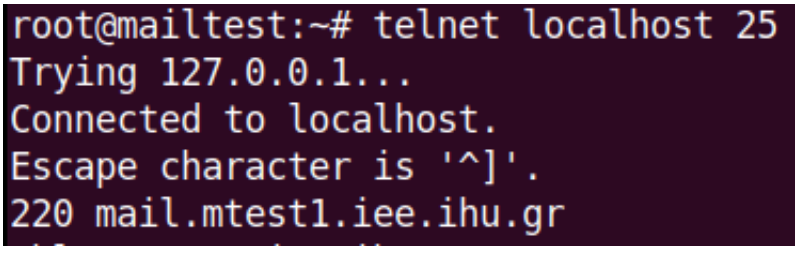


Figure 4.3: Smtpd banner

In the 4.3 the ***smtpd_banned*** is adding ***mail.mtest1.iee.ihu.gr*** in the EHLO greeting.

4.6.2 Implementing SPF protocol

The Sender Policy Framework (SPF) is a widely adopted email authentication protocol designed to prevent email spoofing by allowing domain owners to specify which mail servers are authorized to send emails on their behalf. In order to implement the SPF protocol involves configuring DNS records for your domain. At first, the appropriate pack that needs to be installed is shown in Listing 4.30.

```
root@mailtest:~# apt install postfix-policyd-spf-python
```

Listing 4.30: Installing SPF packet

In order to enable the service it is required to add the following configuration in the ***/etc/postfix/master.cf*** file.

```
policyd-spf  unix  -       n       n       -       0       spawn
    user=policyd-spf argv=/usr/bin/policyd-spf
```

Listing 4.31: Enabling SPF service

This configuration sets up a service in Postfix to handle SPF checks using policyd-spf. When an email is received, Postfix uses policyd-spf to verify if the sending server is authorized to send emails on behalf of the domain.

4.6.3 Implementing DKIM protocol

One key aspect of the Email authentication process of the mail server is the implementation of the DKIM protocol. This email authentication protocol is designed to allow email receivers to verify that an email message originated from the domain it claims to have come from and that the message was not altered in transit. Implementing DKIM involves creating a public-private key pair, adding a DKIM DNS record, and configuring the mail server to sign outgoing emails with the private key. In this case, I used the OpenDKIM[46] tool which is an open-source implementation of the DKIM protocol. The appropriate packets that need to be installed are shown in Listing 4.32 as well as adding the Postfix user to the Opendkim group.

```
root@mailtest:~# apt install opendkim opendkim-tools -y
root@mailtest:~# gpasswd -a postfix opendkim
```

Listing 4.32: Installing DKIM packets

After installing the OpenDKIM tool I emptied the file ***/etc/opendkim.conf*** in order to add the following configuration shown in Listing 4.33.

```
Syslog                  yes
UMask                   002
Canonicalization        simple
Mode                    sv
SubDomains              no
AutoRestart       yes
AutoRestartRate         10/1M
Background          yes
DNSTimeout          5
SignatureAlgorithm      rsa-sha256
Socket                  local:/var/spool/postfix/opendkim/opendkim.sock
PidFile               /var/run/opendkim/opendkim.pid
```

```
OversignHeaders         From
TrustAnchorFile       /usr/share/dns/root.key
UserID                opendkim
KeyTable             refile:/etc/opendkim/key.table
SigningTable         refile:/etc/opendkim/signing.table
ExternalIgnoreList   /etc/opendkim/trusted.hosts
InternalHosts /etc/opendkim/trusted.hosts
```

Listing 4.33: Configuring opendkim.conf

In configuring OpenDKIM for the mail server, I set several key parameters in the ***opendkim.conf*** file. The configuration enables logging to syslog for monitoring purposes ***Syslog yes*** and sets file creation permissions ***UMask 002***. The canonicalization method is specified as simple to balance simplicity and compatibility, while the operational mode is set to sign and verify ***Mode sv***. Subdomain signing is disabled ***SubDomains no***, and automatic restarts are configured to support the reliability of the mailerver ***AutoRestart yes, AutoRestartRate 10/1M***. The service runs in the background ***Background yes***, with a DNS timeout of 5 seconds ***DNSTimeout 5***. For enhanced security, the rsa-sha256 signature algorithm is integrated with the directive ***SignatureAlgorithm rsa-sha256***. In order to establish the communication with Postfix I set a local Unix socket using the directive ***Socket local:/var/spool/postfix/opendkim/opendkim.sock***, and the process ID is stored in ***/var/run/opendkim/opendkim.pid***. The service runs under the opendkim user ***UserID opendkim***. Finally, the key and signing tables are defined as ***KeyTable refile:/etc/opendkim/key.table***, ***SigningTable refile:/etc/opendkim/ signing.table***, and both internal and external trusted hosts are listed in the file ***/etc/opendkim/trusted.hosts***.
To integrate DKIM into the mail server setup, several directories, and permission configurations are necessary. In Listing 4.34 I demonstrate the commands in order to create and secure directories needed for OpenDKIM to function correctly:

```
root@mailtest:~# mkdir /etc/opendkim
root@mailtest:~# mkdir /etc/opendkim/keys
root@mailtest:~# chown -R opendkim:opendkim /etc/opendkim
root@mailtest:~# chmod go-rw /etc/opendkim/keys
```

Listing 4.34: Creating DKIM directories

In the DKIM setup process, configuring the ***signing.table***, ***key.table***, and ***trusted.hosts*** files is a critical step known as configuring DKIM policy and key management. This involves defining the specific policies for how DKIM keys are used, mapping domains and selectors to their respective keys, and specifying trusted hosts. In the following Listings, I demonstrate the setup process of each file respectively.

```
*@mtest1.iee.ihu.gr default._domainkey.mtest1.iee.ihu.gr
```

Listing 4.35: Configuring the signing.table

The ***signing.table*** file maps the domain to the corresponding DKIM selector. This table is responsible for telling the DKIM service which keys to use for signing outgoing emails from the specific domain.

```
default._domainkey.mtest1.iee.ihu.gr mtest1.iee.ihu.gr:default:/etc/opendkim/
    keys/mtest1.iee.ihu.gr/default.private
```

Listing 4.36: Configuring the key.table

The key.table file defines the association between the DKIM selector and the actual private key files used for signing. This table ensures that the correct private key is used for each selector. In this case, the selector is set to default.

```
127.0.0.1
localhost
*.mtest1.iee.ihu.gr
```

Listing 4.37: Configuring the trusted.hosts

The trusted.hosts file lists the IP address and the domain that are allowed to bypass DKIM checks, such as internal mail servers or known relay hosts. This helps in managing which hosts are trusted to send emails without requiring DKIM verification.
In the process of configuring DKIM for a mail server, generating and managing DKIM keys is essential. In Listing 4.38 I explain the way the keys are generated.

```
root@mailtest:~# mkdir /etc/opendkim/keys/mtest1.iee.ihu.gr
root@mailtest:~# opendkim-genkey -b 2048 -d mtest1.iee.ihu.gr -D /etc/opendkim
    /keys/mtest1.iee.ihu.gr -s default -v
root@mailtest:~# chown opendkim:opendkim  /etc/opendkim/keys/mtest1.iee.ihu.gr
    /default.private
```

Listing 4.38: Generating DKIM keys

The first command creates a directory specifically for storing DKIM keys for the domain ***mtest1.iee.ihu.gr***. The second command generates a DKIM key pair for the domain. The output of this command is two files containing the public and the private keys. The public key is stored in the file ***/etc/opendkim/keys/mtest1.iee.ihu.gr/default.txt*** and it contains the TXT DNS record which will be added in the DNS zone of the domain. The private key is stored in the file ***/etc/opendkim/keys/mtest1.iee.ihu.gr/default.private*** and it is used by the mail server to sign outgoing emails. The third command changes the ownership of the ***default.private*** file to the opendkim user and group. This ensures that only the opendkim service can access the private key preventing unauthorized access to the key.
Setting up the directory and permissions for the OpenDKIM socket is an essential step to ensure proper communication between Postfix and OpenDKIM. In Listing 4.39 I showcase this process.

```
root@mailtest:~# sudo mkdir /var/spool/postfix/opendkim
root@mailtest:~# sudo chown opendkim:postfix /var/spool/postfix/opendkim/
```

Listing 4.39: Configuring OpenDKIM socket

The first command creates a directory in ***/var/spool/postfix*** specifically for the OpenDKIM socket. The second command changes the ownership of the socket directory was setting it to the opendkim for the user and the postfix for the group. This configuration ensures that both OpenDKIM and Postfix can access the socket for inter-process communication.
Continuing with the configuration of the OpenDKIM socket I added the directive shown Listing 4.40 in the***/etc/default/opendkim*** file.

```
SOCKET="local:/var/spool/postfix/opendkim/opendkim.sock"
```

Listing 4.40: Configuring OpenDKIM file

The last step to integrate OpenDKIM with Postfix involves configuring the ***/etc/postfix/main.cf*** file to use the OpenDKIM milter (mail filter). The directives I added are shown in Listing 4.41.

```
milter_default_action = accept
milter_protocol = 6
smtpd_milters = local:/var/spool/postfix/opendkim/opendkim.sock
non_smtpd_milters = $smtpd_milters
```

Listing 4.41: Configuring OpenDKIM milter

Configuring ***milter_default_action = accept*** instructs Postfix to accept and deliver emails even if the OpenDKIM milter is facing an issue. The directive ***milter_protocol = 6*** sets the protocol version used for communication between Postfix and the milter. The use of version 6 of the milter protocol, which supports advanced features and better compatibility with modern mail filters like OpenDKIM. The directive ***smtpd_milters = local:/var/spool/postfix/opendkim /opendkim.sock*** sets the Unix domain socket for the Postfix SMTP daemon to communicate with the OpenDKIM milter thus enabling DKIM verification and signing for incoming emails. Finally, by setting up ***non_smtpd_milters = $smtpd_milters*** it ensures that both SMTP and non-SMTP email submissions use the same milter socket configuration, which enables consistent DKIM handling for every email sent.

4.7 Implementation of LDAP authentication

Integrating Lightweight Directory Access Protocol (LDAP) authentication into a mail server is essential for secure and efficient user management. LDAP provides a centralized way to authenticate users, ensuring that only authorized individuals can access the mail server. This method enhances security and simplifies the process of managing user accounts. In this thesis, I will explain how LDAP authentication was integrated into the mail server, starting with the installation of necessary packages and followed by the configuration of the services.
The packets in the following Listing need to be installed in order to integrate LDAP on the mail server.

```
root@mailtest:~# apt install nslcd nslcd-utils
root@mailtest:~# apt install postfix-ldap
root@mailtest:~# apt install dovecot-ldap
```

Listing 4.42: Installing LDAP packets

The Name Service LDAP[47] (NSLCD) daemon provides essential LDAP integration for system-wide authentication. Its purpose is to connect to an LDAP server and retrieve user and group information for system authentication and authorization. In configuring LDAP authentication for the mail server, the ***/etc/nslcd.conf*** file was set up to enable the NSLCD (Name Service LDAP) daemon to connect to the LDAP server and retrieve user and group information.

```
uid nslcd
gid nslcd
uri ldap://192.168.6.81:389
base dc=it,dc=teithe,dc=gr
tls_reqcert try
tls_cacertfile /etc/ssl/certs/ca-certificates.crt
pam_password_prohibit_message Please visit https://apps.it.teithe.gr/user#
   settings to change your password.
pagesize 500
filter passwd (info=users)
map    passwd uid              uid
map    passwd homeDirectory    homeDirectory
map    passwd gecos            title
map    shadow uid              uid
map    shadow shadowLastChange pwdChangedTime
filter group  (&(objectClass=posixGroup)(gidNumber=*))
```

Listing 4.43: Configuring nslcd.conf

The daemon operates under the nslcd user and group by adding the directives ***uid nslcd*** and ***gid nslcd***. The LDAP server is specified with the ***URI ldap://192.168.6.81:389***. The directive ***base DN dc=it,dc=teithe,dc=gr*** specifies the base distinguished name ***DN*** for LDAP searches. This is the starting point in the LDAP directory tree where searches for user and group information will begin. Using the directive ***tls_reqcert try***, TLS encryption is attempted to secure communications with the LDAP server, while utilizing CA certificates located at ***/etc/ssl/certs/ca-certificates.crt***. Additionally, via the directive ***pam_password_prohibit _message*** a custom message is provided to users attempting to change their password, directing them to the URL that it specifies. The configuration also includes performance optimizations, such as setting the ***pagesize 500*** for limiting the entries returned to 500 on a single LDAP query. Filters and attribute mappings are established to align LDAP attributes with local system attributes, such as mapping the LDAP uid to the local uid and filtering group entries to those with an objectClass of posixGroup. This configuration ensures efficient and secure retrieval of user and group information from the LDAP server, supporting centralized authentication for the mail server.
Another file that needs to be configured is the ***/etc/nsswitch.conf***. This file configures the system's Name Service Switch (NSS), which determines the sources from which to retrieve various system databases, including user and group information. In Listing 4.44 I show the configuration I used.

```
passwd:         compat ldap systemd
group:          compat ldap systemd
shadow:         compat ldap
```

Listing 4.44: Configuring nsswitch.conf

To authenticate users, the ***passwd***, *group*, and ***shadow*** databases were set to use compat, ldap, and systemd, ensuring that the system first checks local files and then queries the LDAP server. The passwd and group entries in particular include systemd to integrate with systemd-based user and group management. This configuration ensures that LDAP authentication is integrated into the mail server working along with the traditional methods.
Integrating LDAP authentication on Postfix requires the creation of a lookup configuration file. This configuration defines how and where Postfix should retrieve user information from the LDAP Server by specifying the parameters for connecting to the LDAP server, the search base, the search filters, and the attributes to be returned. The user lookup configuration file I used is shown in Listing 4.45 and is stored in the path ***/etc/postfix/ldap-users.cf***.

```
server_host = ldap://192.168.6.81:389
search_base = dc=it,dc=teithe,dc=gr
bind = no
query_filter = (&(objectClass=posixAccount)(uid=%s))
result_attribute = mailp
```

Listing 4.45: Configuring ldap-users.cf

To conclude the integration of LDAP authentication on Postfix, I added the appropriate directives shown in Listing 4.46 in the ***/etc/postfix/main.cf***.

```
local_recipient_maps = ldap:/etc/postfix/ldap-users.cf
```

Listing 4.46: Configuring LDAP on main.cf

The directive ***local_recipient_maps*** is used to verify local recipient addresses by querying an LDAP directory. The LDAP configuration file ***/etc/postfix/ldap-users.cf*** contains the necessary details for connecting to the LDAP server and performing queries.

Integrating LDAP on Dovecot requires the creation of the file ***/etc/dovecot/dovecot-ldap.conf.ext*** that contains the settings for Dovecot to authenticate users against the LDAP server. The configuration is shown in the Listing 4.47.

```
hosts = ldap://192.168.6.81:389
dn =
dnpass =
base = dc=it,dc=teithe,dc=gr
scope = subtree
tls_require_cert = try
tls_ca_cert_file = /etc/ssl/certs/ca-certificates.crt
user_attrs = uid=homeDirectory,mail=maildir:~/Maildir
user_filter = (&(objectClass=posixAccount)(uid=%u))
pass_attrs = uid=user,userPassword=password
pass_filter = (&(objectClass=posixAccount)(uid=%u))
```

Listing 4.47: Configuring dovecot-ldap.conf.ext

Another setting that needs to be added is the directive to include the LDAP authentication mechanism in the mail server. To achieve that I added the following directive in the ***/etc/dovecot/conf.d/10-auth.conf*** file.

```
!include auth-ldap.conf.ext
```

Listing 4.48: Adding LDAP authentication on Dovecot

This directive includes the file ***auth-ldap.conf.ext***. This file contains details about how to connect to the LDAP server, how to query it for user information, and how to authenticate users against it.

Chapter 5

Experiments and Conclusion

5.1 Testing the mail server

This Section presents the experiments conducted to evaluate the functionality and security of the implemented mail server. To demonstrate the effectiveness of the system, an email was sent from the Postfix mail server to a Gmail[48] account. Given Gmail's stringent rules for verifying the originality and authenticity of the sender, it serves as an ideal platform to test the mail server's current configuration. The experiments cover validating email authentication protocols (SPF, DKIM, DMARC), encryption mechanisms (SSL/TLS, STARTTLS), protocol functionality (SMTP, IMAP), spam prevention, and access restrictions. Each experiment aims to ensure the reliability, security, and efficiency of the mail server. By showcasing the server's performance in these areas, this chapter provides a comprehensive evaluation of the implemented solutions, highlighting the system's strengths and identifying any areas for improvement.

5.1.1 Sending an email

An email was sent to a personal Gmail address by connecting via telnet to port 25 which is designated to the SMTP server. The connection and composition processes are demonstrated in the figure 5.1.

```
root@mailtest:~# telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mail.mtest1.iee.ihu.gr
ehlo mtest1.iee.ihu.gr
250-mail.mtest1.iee.ihu.gr
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250-SMTPUTF8
250 CHUNKING
mail from: jimouroudelis@mtest1.iee.ihu.gr
250 2.1.0 Ok
rcpt to: jimouroudelis@gmail.com
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
To: jimouroudelis@gmail.com
Subject: This is a Test Email
This Email was send in order to test the protocols I integrated to the Mailserver.
.
250 2.0.0 Ok: queued as E57011A032
quit
221 2.0.0 Bye
Connection closed by foreign host.
```

Figure 5.1: Composing the Email

The figure 5.2 showcases the inbox of the Gmail in which the email was delivered.

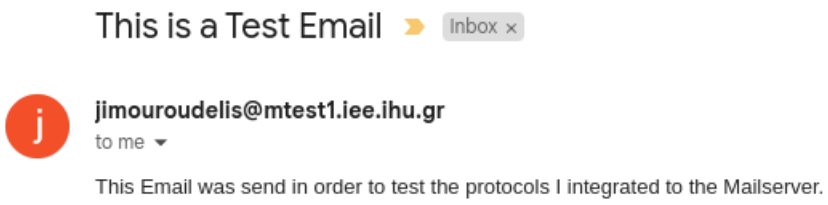
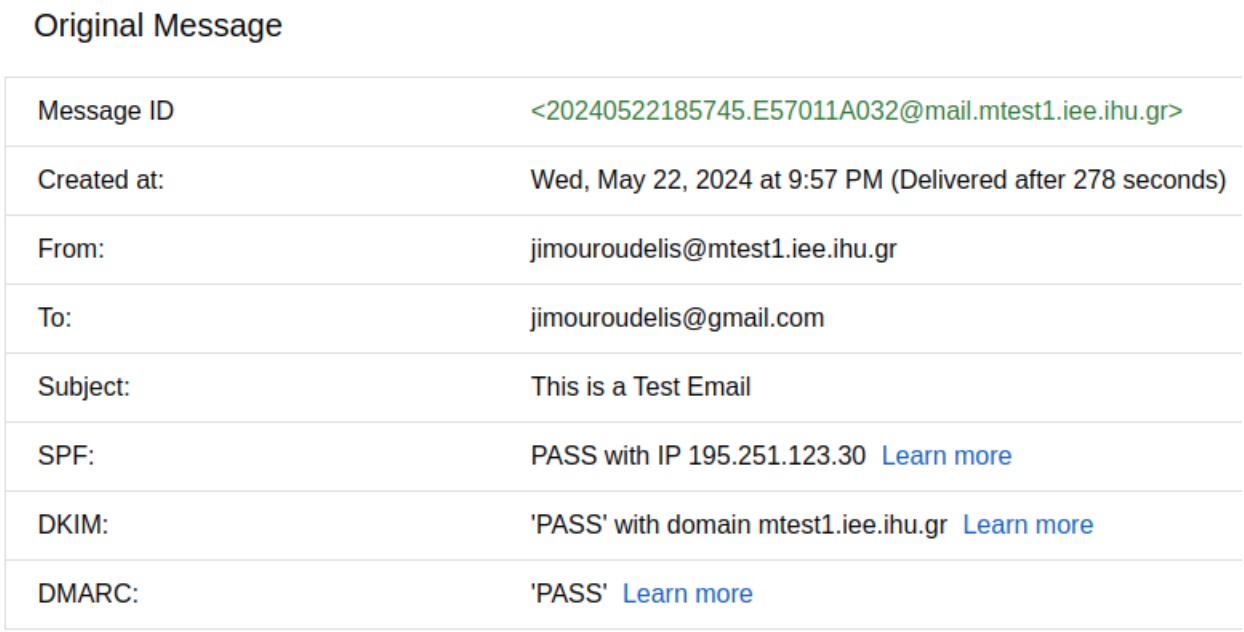


Figure 5.2: Email arrived in the Inbox

After pressing the ”Show Original” button in Gmail, the headers reveal that the SPF, DKIM, and DMARC protocols are functional, and Google successfully reads these authentication results from our mail server.



Original Message

Message ID	<20240522185745.E57011A032@mail.mtest1.iee.ihu.gr>
Created at:	Wed, May 22, 2024 at 9:57 PM (Delivered after 278 seconds)
From:	jimouroudelis@mtest1.iee.ihu.gr
To:	jimouroudelis@gmail.com
Subject:	This is a Test Email
SPF:	PASS with IP 195.251.123.30 Learn more
DKIM:	'PASS' with domain mtest1.iee.ihu.gr Learn more
DMARC:	'PASS' Learn more

Figure 5.3: Email Headers

For a more in-depth analysis of the headers, in figure 5.4, I showcase the raw information to illustrate how Google's email system has validated the SPF, DKIM, and DMARC protocols. The detailed header examination will provide insights into further interpreting the Email's headers.

```
Authentication-Results: mx.google.com;
       dkim=pass header.i=@mtest1.iee.ihu.gr header.s=default header.b=KhAphevU;
       spf=pass (google.com: domain of jimouroudelis@mtest1.iee.ihu.gr designates 195.251.123.30 as permitted sender)
       dmarc=pass (p=NONE sp=NONE dis=NONE) header.from=mtest1.iee.ihu.gr
```

Figure 5.4: Raw authentication protocol headers

The headers also import the public key of the DKIM protocol which is shown in the figure 5.5.

```
DKIM-Signature: v=1; a=rsa-sha256; c=simple/simple; d=mtest1.iee.ihu.gr; s=default; t=1716404515;
h=To:Subject:Date:From:From; b=KhAphevUlVtEUas6lPErQ2vy4drdsZNdDpAN8vql4SDyymeQrSfYumUl2fBCm5OSl
         KjImtjj9Sf8pbD2bVB26Mo9zN2JVXix9l7qw1hljxgbY12c7Y9uyXZHOh5kewkbGvi
         Hu18qugr4LmA/53bBM+hS6J9kdnWzBpqIg7x2eeBTjfe3x67erpnfh0r43X/6i5pNH
         wxkeImebV9kGXh3KlAupSHx3+GsK7a1h9/+UwQt4AH8SLy+sMjXTiE2MwCWhial6ZD
         q7Ebm+ip5eSL6BGdsU6yhmALalweEJN98S4yHDfAJWnSLmOECIj4infNPO8b9CFO7I
         lDDFgX4Bq8TGw==
```

Figure 5.5: DKIM public key

To further enhance the header analysis, I will demonstrate the addition of STARTTLS, showcasing how email transmission is encrypted to ensure data security and integrity. The figure 5.6 illustrates the successful negotiation and application of the TLS 1.3 protocol during the email transmission process.

```
Received: from mail.mtest1.iee.ihu.gr (mailtest.iee.ihu.gr. [195.251.123.30])
        by mx.google.com with ESMTPS id 4fb4d7f45d1cf-578433af185si84507a12.5.2024.05.22.12.01.56
        for <jimouroudelis@gmail.com>
        (version=TLS1_3 cipher=TLS_AES_256_GCM_SHA384 bits=256/256);
```

Figure 5.6: TLS v1.3

To demonstrate SpamAssassin's functionality, the figure 5.7 shows the "X-Spam-Flag" and "X-Spam-Score" fields. These fields reveal how SpamAssassin evaluates the email content, assigns a spam score, and determines whether the email is flagged as spam. The detailed header information will provide insights into the spam filtering process and its effectiveness in identifying and handling unwanted emails.

```
X-Spam-Checker-Version: SpamAssassin 3.4.6 (2021-04-09) on mailtest.iee.ihu.gr
X-Spam-Level:
X-Spam-Status: No, score=-1.0 required=5.0 tests=ALL_TRUSTED autolearn=ham autolearn_force=no version=3.4.6
Received: from mtest1.iee.ihu.gr (localhost [127.0.0.1]) by mail.mtest1.iee.ihu.gr (Postfix) with ESMTP id E57011A032
```

Figure 5.7: SpamAssassin Headers

5.1.2 Receiving an Email

In order to illustrate the email reception process on the mail server, the figure 5.8 showcases the ***/var/log/mail.log*** file, mentioning the detailed steps an email takes as it travels through various daemons. Initially, the connection is established when Gmail's server connects to our mail server. The SPF check is then performed by ***policyd-spf***, which verifies the sender's SPF record. Postfix receives the email, logs the message ID and client information, and then opendkim processes and verifies the DKIM signature of the sender. The email is subsequently queued for further processing.
SpamAssassin evaluates the email, assigning a spam score based on the body of the email. Postfix then delivers the email via SpamAssassin, logging the delivery status. Finally, Postfix uses LMTP to deliver the email to the user's mailbox via Dovecot, ensuring that the email is securely and correctly stored. This comprehensive sequence demonstrates how incoming emails are processed, verified, and delivered within the mail server.

```
May 23 00:51:33 mailtest postfix/smtpd[157542]: connect from mail-yb1-f170.google.
com[209.85.219.170]
May 23 00:51:35 mailtest policyd-spf[157548]: prepend Received-SPF: Pass (mailfrom
) identity=mailfrom; client-ip=209.85.219.170; helo=mail-yb1-f170.google.com; enve
lope-from=jimouroudelis@gmail.com; receiver=<UNKNOWN>
May 23 00:51:35 mailtest postfix/smtpd[157542]: 3747B1A081: client=mail-yb1-f170.g
oogle.com[209.85.219.170]
May 23 00:51:35 mailtest postfix/cleanup[157549]: 3747B1A081: message-id=<CAD0BFaW
i1r2-C5rxxVwKcmcr845bLOLoyaE9AEYj_dd=bmamGw@mail.gmail.com>
May 23 00:51:35 mailtest opendkim[156749]: 3747B1A081: s=20230601 d=gmail.com a=rs
a-sha256 SSL
May 23 00:51:35 mailtest postfix/qmgr[157538]: 3747B1A081: from=<jimouroudelis@gma
il.com>, size=5517, nrcpt=1 (queue active)
May 23 00:51:35 mailtest spamd[1193]: spamd: connection from 127.0.0.1 [127.0.0.1]
:33562 to port 783, fd 5
May 23 00:51:35 mailtest spamd[1193]: spamd: setuid to debian-spamd succeeded
May 23 00:51:35 mailtest spamd[1193]: spamd: processing message <CAD0BFaWi1r2-C5rx
xVwKcmcr845bLOLoyaE9AEYj_dd=bmamGw@mail.gmail.com> for debian-spamd:127
May 23 00:51:35 mailtest postfix/smtpd[157542]: disconnect from mail-yb1-f170.goog
le.com[209.85.219.170] ehlo=2 starttls=1 mail=1 rcpt=1 bdat=1 quit=1 commands=7
May 23 00:51:35 mailtest spamd[1193]: spamd: clean message (-1.1/5.0) for debian-s
pamd:127 in 0.5 seconds, 5583 bytes.
May 23 00:51:35 mailtest spamd[1193]: spamd: result: . -1 - DKIM_SIGNED,DKIM_VALID
,DKIM_VALID_AU,FREEMAIL_FROM,HTML_MESSAGE,RCVD_IN_DNSWL_NONE,RCVD_IN_MSPIKE_H2,SPF
_HELO_NONE,SPF_PASS scantime=0.5,size=5583,user=debian-spamd,uid=127,required_scor
e=5.0,rhost=127.0.0.1,raddr=127.0.0.1,rport=33562,mid=<CAD0BFaWi1r2-C5rxxVwKcmcr84
5bLOLoyaE9AEYj_dd=bmamGw@mail.gmail.com>,autolearn=ham autolearn_force=no
May 23 00:51:35 mailtest spamd[1058]: prefork: child states: II
May 23 00:51:35 mailtest postfix/pickup[157537]: C32D91A084: uid=127 from=<jimouro
udelis@gmail.com>
May 23 00:51:35 mailtest postfix/pipe[157550]: 3747B1A081: to=<jimouroudelis@mtest
1.iee.ihu.gr>, relay=spamassassin, delay=1, delays=0.45/0.01/0/0.54, dsn=2.0.0, st
atus=sent (delivered via spamassassin service)
May 23 00:51:35 mailtest postfix/qmgr[157538]: 3747B1A081: removed
May 23 00:51:35 mailtest postfix/cleanup[157549]: C32D91A084: message-id=<CAD0BFaW
i1r2-C5rxxVwKcmcr845bLOLoyaE9AEYj_dd=bmamGw@mail.gmail.com>
May 23 00:51:35 mailtest postfix/qmgr[157538]: C32D91A084: from=<jimouroudelis@gma
il.com>, size=6171, nrcpt=1 (queue active)
May 23 00:51:36 mailtest postfix/lmtp[157555]: C32D91A084: to=<jimouroudelis@mtest
1.iee.ihu.gr>, relay=mail.mtest1.iee.ihu.gr[private/dovecot-lmtp], delay=0.23, del
ays=0.05/0.01/0.04/0.12, dsn=2.0.0, status=sent (250 2.0.0 <jimouroudelis@mtest1.i
ee.ihu.gr> WNx6NOdoTmZ0ZwIAzDcxFg Saved)
May 23 00:51:36 mailtest postfix/qmgr[157538]: C32D91A084: removed
```

Figure 5.8: The mail.log report after receiving an email

5.1.3 EHLO restrictions

To ensure the security of the mail server, I conducted an EHLO spoofing attack to verify the effectiveness of the HELO access restrictions. This involved attempting to connect to the mail server from another mail server via Telnet and sending an email using the EHLO command. The test specifically targeted the rules set in the /etc/postfix/helo_access file, which is configured to reject connections that do not meet the criteria shown in the Listing 4.8.
During the test, the connection was initiated, and the EHLO command was issued with a hostname that falls under the restricted entries in the /etc/postfix/helo_access file. As expected, the mail server

correctly identified the unauthorized EHLO hostname and rejected the connection according to the security rules I had configured. The figure 5.9 demonstrates the outcome of this test.

```
root@s495968:~# telnet 195.251.123.30 25
Trying 195.251.123.30...
Connected to 195.251.123.30.
Escape character is '^]'.
220 mail.mtest1.iee.ihu.gr
ehlo mail.mtest1.iee.ihu.gr
250-mail.mtest1.iee.ihu.gr
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250-SMTPUTF8
250 CHUNKING
mail from: jimouroudelis@mtest1.iee.ihu.gr
250 2.1.0 Ok
rcpt to: jimouroudelis@gmail.com
554 5.7.1 <mail.mtest1.iee.ihu.gr>: Helo command rejected: Sorry, den epitrepertrte prosvasi.
```

Figure 5.9: EHLO spoofing attack

5.1.4 Spam Filtering

To test the effectiveness of SpamAssassin, I sent a test email containing the GTUBE[49] (Generic Test for Unsolicited Bulk Email) body, which is designed to trigger spam filters. Upon receiving the email, the mail headers showed that SpamAssassin successfully identified and marked the email as spam.

The email headers included specific indicators such as ”X-Spam-Flag: YES” and a high ”X-Spam-Score” value, signifying that the message was classified as spam. Additional header entries, such as ”X-Spam-Report,” detailed the various rules triggered by the GTUBE body, confirming that the email matched patterns commonly associated with unsolicited bulk email. This test validated SpamAssassin’s ability to effectively detect and flag spam emails, thereby enhancing the security and reliability of the mail server. The figure 5.10shows the corresponding headers inserted from SpamAssassin.

```
X-Spam-Checker-Version: SpamAssassin 3.4.6 (2021-04-09) on mailtest.iee.ihu.gr
X-Spam-Flag: YES
X-Spam-Level: **************************************************
X-Spam-Status: Yes, score=998.9 required=5.0 tests=DKIM_SIGNED,DKIM_VALID,
        DKIM_VALID_AU,FREEMAIL_FROM,GTUBE,HTML_MESSAGE,RCVD_IN_DNSWL_NONE,
        RCVD_IN_MSPIKE_H2,SPF_HELO_NONE,SPF_PASS autolearn=no
        autolearn_force=no version=3.4.6
X-Spam-Report:
        * 1000 GTUBE BODY: Generic Test for Unsolicited Bulk Email
        * -0.0 SPF_PASS SPF: sender matches SPF record
        *  0.0 SPF_HELO_NONE SPF: HELO does not publish an SPF Record
        *  0.0 FREEMAIL_FROM Sender email is commonly abused enduser mail
        *      provider
        *      [jimouroudelis[at]gmail.com]
        *  0.0 HTML_MESSAGE BODY: HTML included in message
        * -0.1 DKIM_VALID Message has at least one valid DKIM or DK signature
        * -0.1 DKIM_VALID_AU Message has a valid DKIM or DK signature from
        *      author's domain
        *  0.1 DKIM_SIGNED Message has a DKIM or DK signature, not necessarily
        *       valid
        * -1.0 RCVD_IN_MSPIKE_H2 RBL: Average reputation (+2)
        *      [209.85.219.171 listed in wl.mailspike.net]
        * -0.0 RCVD_IN_DNSWL_NONE RBL: Sender listed at
        *      https://www.dnswl.org/, no trust
        *      [209.85.219.171 listed in list.dnswl.org]
```

Figure 5.10: SpamAssassin Headers

5.2 Conclusions

This thesis has set out on a thorough and comprehensive implementation of an electronic mail management system, addressing the critical components and challenges involved in ensuring secure, efficient, and reliable email communication. The study began with a detailed analysis of the protocols such as SMTP for outgoing traffic, and IMAP and POP for incoming traffic. Security protocols such as SSL/TLS and STARTTLS, along with email authentication protocols like SPF, DKIM, and DMARC were also examined. The significance of DNS entries was also mentioned. From there, the investigation transitioned into mail server technologies evaluating and comparing Mail Transfer Agents such as Postfix, Exim, and Sendmail. Moreover, the role of Dovecot as a Mail Delivery Agent (MDA) and the implementation of SSL certificates from Let's Encrypt were also discussed. Another field explored was spam protection mechanisms using SpamAssassin and ClamAV, and authentication technologies like SASL and LDAP. The practical contribution of this thesis was the setup of the mail server and the integration of various protocols and technologies. This thesis also provided a step-by-step guide to the mail server setup by including configuration files and code snippets to illustrate the process. Finally, various tests were conducted to validate the functionality of the mail server. These include demonstrating email authentication using SPF, DKIM, and DMARC, and ensuring STARTTLS encryption with TLS 1.3 via sending and receiving emails. Additionally, spam prevention was also tested with SpamAssassin, and the effectiveness of HELO access restrictions in enhancing security was displayed.

5.3 Future Work

Future work on the mail server may implicate developing a user-friendly interface for email management that could significantly improve the usability and accessibility of the mail server. This browser-based interface would allow users to easily manage their emails, offering functionalities such as composing, reading, and organizing emails, without needing a separate email client.

Bibliography

[1] MDPI, "Email security issues, tools, and techniques," *Proceedings*, 2018. Accessed: 18-05-2024.

[2] Wikipedia, "Simple mail transfer protocol." https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol, 2024. Accessed: 19-05-2024.

[3] IONOS, "Smtp & esmtp protocol: Explanation, port, example & more." https://www.ionos.com/digitalguide/e-mail/technical-matters/smtp/, 2024.

[4] Cloudflare, "What is imap? imap vs. pop3." https://www.cloudflare.com/learning/email-security/imap/, 2024. Accessed: 19-05-2024.

[5] Wikipedia, "Internet message access protocol." https://en.wikipedia.org/wiki/Internet_Message_Access_Protocol, 2024. Accessed: 19-05-2024.

[6] Thunderbird, "Thunderbird — free your inbox." https://www.thunderbird.net, 2024. Accessed: 20-05-2024.

[7] BlueMail, "Bluemail — the best email app for windows, mac, linux, ios & android." https://bluemail.me/, 2024. Accessed: 20-05-2024.

[8] M. Sampson, "Post office protocol," *Encyclopedia of Information Systems*, 2003. Accessed: 20-05-2024.

[9] J. Myers and M. Rose, "Post office protocol - version 3 (pop3)." https://www.rfc-editor.org/rfc/rfc1939.txt, 1996. Accessed: 20-05-2024.

[10] IEEE, "Understanding ssl protocol and its cryptographic weaknesses," *IEEE Xplore*, 2013. Accessed: 20-05-2024.

[11] R. Holz, J. Amann, O. Mehani, M. Wachs, and M. A. Kaafar, "Tls in the wild: An internet-wide analysis of tls-based protocols for electronic communication," *arXiv preprint arXiv:1511.00341*, 2015.

[12] E. F. Foundation, "Technical deep dive: Starttls everywhere." https://www.eff.org/deeplinks/2018/06/technical-deep-dive-starttls-everywhere, 2018. Accessed: 20-05-2024.

[13] B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux, *Password interception in a SSL/TLS channel*. Springer, 2003.

[14] Cloudflare, "Dns records explained." https://www.cloudflare.com/learning/dns/dns-records/, 2024. Accessed: 07-04-2024.

[15] Cloudflare, "What is an mx record?." https://www.cloudflare.com/learning/dns/dns-records/dns-mx-record/, 2024. Accessed: 07-04-2024.

[16] "What are spf, dkim and dmarc?." https://www.thesecuritybuddy.com/email-security/what-are-spf-dkim-and-dmarc/, 2024. Accessed: 10-04-2024.

[17] Cloudflare, "Dns dkim record." https://www.cloudflare.com/learning/dns/dns-records/dns-dkim-record/, 2024. Accessed: 07-04-2024.

[18] M. Kucherawy and E. Zwicky, "Domain-based Message Authentication, Reporting, and Conformance (DMARC)," RFC 7489, RFC Editor, 2015. [Online; accessed 5-4-2024].

[19] R. Hildebrandt and P. Koetter, *The book of Postfix: state-of-the-art message transport*. No Starch Press, 2005.

[20] K. D. Dent, *Postfix: The Definitive Guide: A Secure and Easy-to-Use MTA for UNIX*. " O'Reilly Media, Inc.", 2003.

[21] "Porstfix." https://www.plesk.com/wiki/postfix/#:~:text=Postfix%20is%20a%20hugely%2Dpopular,of%20UNIX%2Dlike%20operating%20systems. Date accesed: 24-3-2024.

[22] P. Hazel, *Exim: The Mail Transfer Agent*. " O'Reilly Media, Inc.", 2001.

[23] Wikipedia contributors, "Exim." https://en.wikipedia.org/wiki/Exim, 2024. Accessed: 05-05-2024.

[24] Exim, "Exim internet mailer." https://www.exim.org/index.html, 2024. Accessed: 05-05-2024.

[25] B. Costales, C. Assmann, G. Jansen, and G. N. Shapiro, *sendmail: Build and Administer sendmail*. " O'Reilly Media, Inc.", 2007.

[26] R. Farrow, "Interview with eric allman.," *login Usenix Mag.*, vol. 42, no. 2, 2017.

[27] Wikipedia contributors, "Sendmail." https://en.wikipedia.org/wiki/Sendmail, 2024. Accessed: 06-05-2024.

[28] P. developers, "Choosing the best mta: Postfix vs sendmail vs exim: a comprehensive comparison." https://www.plesk.com/blog/various/postfix-vs-sendmail-vs-exim/, 2024. Accessed: 06-05-2024.

[29] Dovecot, "Dovecot official website." https://www.dovecot.org/, 2024. Accessed: 03-04-2024.

[30] A. Vazquez and A. Vazquez, "Mail server," *Learn CentOS Linux Network Services*, pp. 229–288, 2016.

[31] C. Gentry, *Certificate-based encryption and the certificate revocation problem*. Springer, 2003.

[32] M. Aertsen, M. Korczyński, G. C. Moura, S. Tajalizadehkhoob, and J. Van Den Berg, *No domain left behind: is Let's Encrypt democratizing encryption?* Association for Computing Machinery, 2017.

[33] R. Herbosa, G. Díaz, and M. Castro, "Securing the email services,"

[34] T. Kojm, "Clam antivirus: User manual," *ClamAV*, 2012.

[35] S. Khanji, R. Jabir, L. Ahmad, O. Alfandi, and H. Said, "Evaluation of linux smtp server security aspects—a case study," in *2016 7th International Conference on Information and Communication Systems (ICICS)*, pp. 252–257, IEEE, 2016.

[36] Cisco Systems, Inc., "Clamav." `https://www.clamav.net/`, 2024. Accessed: 13-4-2024.

[37] A. McDonald, *SpamAssassin: A practical guide to integration and configuration*. Packt Publishing Ltd, 2004.

[38] A. K. Seewald, "Combining bayesian and rule score learning: Automated tuning for spamassassin," *Intelligent Data Analysis. Technical report, TR-2004-11 Austrian Research Institute for Artificial Intelligence, Vienna, Austria*, 2004.

[39] Apache Software Foundation, "Apache spamassassin." `http://spamassassin.apache.org/`, 2024. Accessed: 22-4-2024.

[40] R. Siemborski and A. Melnikov, "Smtp service extension for authentication," tech. rep., RFC Editor, 2007.

[41] T. Howes, M. Smith, and G. S. Good, *Understanding and deploying LDAP directory services*. Addison-Wesley Professional, 2003.

[42] K. Zeilenga, "Lightweight Directory Access Protocol (LDAP): The Protocol." `https://www.rfc-editor.org/rfc/rfc4511`, 2006. [Accessed: 28-4-2024].

[43] C. Donley, *LDAP programming management and integration*. Manning Publications, 2003.

[44] E. F. Foundation, "Certbot." `https://certbot.eff.org`, 2024. Accessed: 14-05-2024.

[45] L. Encrypt, "Challenge types." `https://letsencrypt.org/docs/challenge-types/`, 2024. Accessed: 06-05-2024.

[46] T. O. Project, "Opendkim documentation." `https://www.opendkim.org/documentation.html`, 2024. Accessed: 10-05-2024.

[47] A. de Jong, "nss-pam-ldapd documentation." `https://arthurdejong.org/nss-pam-ldapd/`, 2024. Accessed: 03-05-2024.

[48] Google, “Gmail.” https://mail.google.com, 2024. Accessed: 22-05-2024.

[49] A. S. Foundation, “The gtube - generic test for unsolicited bulk email.” https://spamassassin.apache.org/gtube/, 2024. Accessed: 22-05-2024.