National College of Ireland

Securing Postfix Mail servers from Brute-Force attacks using containerization and Fail2ban

MSc Research Project
Msc in Cybersecurity

Kamal Kishore
Student ID: x22146270

School of Computing
National College of Ireland

Supervisor: Prof. Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kamal Kishore
Student ID:	X22146270
Programme:	Msc in Cybersecurity **Year:** 2023
Module:	MSc Research Practicum/Internship part 2
Supervisor:	Prof. Imran Khan
Submission Due Date:	12 August 2024
Project Title:	Securing Postfix Mail servers from Brute-Force attacks using Containerization and Fail2ban
Word Count:	6062 **Page Count**19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Kamal Kishore
Date:	28th July 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Securing Postfix Mail servers using Fail2ban and Containerization

Kamal Kishore
X22146270

Abstract

The growing frequency and severity of cyber-attacks need comprehensive security measures for email servers. Postfix, a widely used mail transfer agent, is frequently the subject of brute force attacks. Such incidents might result in illegal access, revealing sensitive information and interrupting communication services. Traditional security solutions, while useful to some level, sometimes fall short of providing total coverage. Furthermore, coordinating authentication across numerous email-related services may be complicated and error-prone.

This paper addresses the essential issue of protecting email servers, notably Postfix, from attacks like this while increasing system management and scalability. The main issue addressed is the vulnerability of email servers to brute force attacks as well as the complexities involved in handling authentication across numerous services.

To solve these issues, this research installed Fail2Ban (as used by Makopa et al., 2023) on the same server having Postfix and other containers configured, to successfully prevent brute force attacks. A custom script in fail2ban was created for preventing the ports. Further, considering multiple postfix servers and to keep system users separate from mailing users, Postfix and Dovecot were combined with MariaDB to provide central authentication, reducing administrative work while increasing security. Furthermore, a huge milestone was made by converting Postfix and its dependent services, such as Dovecot, DKIM, and DMARC, to Podman containers—an invention that had never been described before.

The outcomes of these implementations were impressive. Fail2Ban successfully decreased the number of brute force attacks, hence improving the security of the Postfix server. The interface with MariaDB enabled a centralized administration system, which streamlined the authentication process across several services. The containerization of Postfix and its dependencies using Podman proved greater portability, consistency, and deployment simplicity, marking an iconic moment in this sector.

Theoretically, this study offers to the body of knowledge by illustrating an integrated approach to protecting and maintaining email servers that is consistent with current practices in cybersecurity and systems administration. Practically, the significant benefits are increased security, improved management, and higher deployment flexibility, which make the system more robust and easier to maintain.

While this research provides a solid foundation for enhancing email server security, several key challenges persist. These include:

1. *Optimizing containerized environments*: Achieving optimal performance and resource usage inside a containerized infrastructure remains a difficult task.
2. *Persistent Scaling*: Scaling email servers to meet changing workloads while keeping performance and security needs more research.
3. *Adapting to emerging threats*: The ever-changing threat landscape needs ongoing adaptation of security measures as well as the development of novel solutions.

4. *Integrating advanced security technologies*: Incorporating developing technologies such as artificial intelligence and machine learning into threat detection and response might greatly improve email security.

Addressing these challenges will be crucial for the continued development of robust and resilient email server solutions.

The report's structure is as follows:

1. *Introduction*: Provides a summary of the topic, study relevance, research questions and objectives, limits, and report structure.
2. *Literature Review*: Evaluate existing options for safeguarding email servers and handling authentication.
3. *Methodology*: Implemented Fail2Ban, integrated Postfix and Dovecot with MariaDB database, and migrated to Podman containers.
4. *Testing and Results*: Evaluate the efficiency of the adopted remedies.
5. *Discussion*: Interpretation of results, comparison to known solutions, and implications for future study.
6. *Conclusion*: Summarized findings, and research contributions.
7. *Future Work*: Recommendations for future improvements.

1 Introduction

The evolving threat landscape highlights the crucial significance of strong email server security. A recent Palo Alto Networks report analysing over 600 incident response cases revealed alarming trends: ransomware and business email compromise accounted for a staggering 70% of incidents, while phishing, software vulnerabilities, and poor password security—a primary vector for brute force attacks—accounted for 77% (Artry, 2022).

These findings underscore the critical necessity to protect email servers such as Postfix from such attacks. Knowing that known software vulnerabilities led to almost half of all reported incidents, and that an alarming 50% of enterprises lack crucial multifactor authentication on vital systems, email server vulnerabilities are increased. Furthermore, the survey underscored the significance of inadequate patch management in 28% of events, emphasising the importance of proactive security measures (Artry, 2022).

Fail2Ban is an intrusion prevention software framework that scans log files (such as /var/log/maillog) for security issues before banning IP addresses associated with undesirable conduct. This study tries to address these issues by creating and improving Fail2Ban, which greatly reduces the success rate of brute force assaults on Postfix servers. This will try to contribute to a more robust email infrastructure that can survive the rising attack of cyberattacks by strengthening security with Fail2Ban settings and centralized authentication using MariaDB. Additionally, this report will explore the benefits of containerizing mail services as a complementary strategy to enhance security, performance, and business continuity. By isolating mail services in containers, organizations can create a more secure and resilient environment, reducing the potential impact of successful attacks.

Research Question and Objectives

This paper aims to answer the following research question: How can Linux Postfix email servers be effectively secured against evolving brute force attacks? To answer this question, the aims of this study include:

1. Designing Fail2Ban to prevent brute force attacks on Postfix.
2. Integrating Postfix and Dovecot with MariaDB to provide centralized authentication.
3. Migrating Postfix and other related services (Dovecot, DKIM and DMARC) to Podman containers.
4. Assessing the efficacy and performance of the implementations.

Limitations

This work presents a complete strategy for protecting Postfix email servers, with a special emphasis on firms that engage substantially in email marketing efforts or using multiple postfix instances in their environment. The setup is done on local machine using Virtualbox 6.1 simulating the configuration of services like enterprise environment. While containerization has the potential to improve isolation and scalability, its effectiveness is dependent on adequate system resources and customized settings. Furthermore, the integration of Fail2Ban with Mails services (Postfix, Dovecot, DKIM and DMARC) necessitates a large upfront effort in configuration and parameterization. It is critical to acknowledge that the cybersecurity landscape is always changing, needing ongoing adaptation and improvement of preventive measures to ensure their efficacy against developing threats.

2 Related Work

Numerous studies have looked at various areas of cybersecurity, including brute force attacks. This research emerged because of the previously described studies:

2.1 Invention of SMTP by Postel.

Postel invented the Simple Mail Transfer Protocol (SMTP) in the early 1980s, which defined fundamental concepts for email security. Author provides a fundamental knowledge of how emails are safely transported and validated between servers, laying the framework for future developments in email security. While SMTP was important at the time, it failed to account for the digital age's developing risks, such as brute force attacks and sophisticated privacy intrusions. This shortcoming underlined the necessity for better security procedures in future research (Postel, 1982).

2.2 Difficulty in protecting email servers in cloud in 2012.

By 2012, Madi *et al*. have addressed the difficulty of protecting email servers in cloud systems. Their research focused on adapting email infrastructure to cloud-based deployments and implementing strong security mechanisms such as TLS (Transport Layer Security) and SASL (Simple Authentication and Security Layer). This study was a big step in adapting standard email security techniques to contemporary cloud systems. However, Madi *et al*. (2012) did not investigate technologies like Fail2Ban or containerization tactics, which might improve security in these new contexts.

2.3 Focus changed to email security in 2013.

Joyia *et al*. (2013) changed the focus to email privacy, looking at vulnerabilities in email headers as well as the limits of established cryptographic protocols. Their research emphasized the significance of securing user identities and metadata, exposing serious flaws in standard privacy protections. However, their study was largely focused on privacy problems and did not address the incorporation of technologies such as Fail2Ban, which might minimize brute force attacks. This omission exposed a crucial gap in handling comprehensive security threats.

2.4 Defence against password attacks in 2014.

Manolache *et al*. (2014) presented a collective defensive technique to counteract password guessing attacks in business situations. They suggested a distributed database system to disseminate attack information across various systems, which would improve existing security mechanisms such as Fail2Ban. This novel technique constituted a significant application of collective defence theory, aimed at pre-emptively blocking attacks and increasing efficiency. However, the study lacked extensive implementation recommendations and could not adequately address privacy concerns, which might limit practical use.

2.5 Introducing of fail2ban as IDPS in 2016.

Ford *et al*. (2016) enhanced this story by presenting a modular and adaptive intrusion detection system that incorporates Fail2Ban data. Their design aims to improve standard intrusion detection systems by allowing for real-time data exchange and analysis across networked agents. This study was founded on the notion of adaptive security, which emphasizes the need of collaborative and adaptable protection systems. Despite its theoretical advances, the study has shortcomings, including a lack of empirical validation and a possible reliance on Fail2Ban data, which might limit its application across different network contexts.

2.6 Defensive techniques against DDOS in 2017.

Papadie and Apostol's 2017 study assessed defence techniques against Distributed Denial of Service (DDoS) assaults, such as Fail2ban. Their findings gave useful insights into how Fail2Ban and other tools work in simulated attack scenarios. While the study provided a thorough review of DDoS countermeasures, its restricted emphasis on specific attack tools and controlled experimental settings limited its practical usefulness (Papadie & Apostol, 2017).

2.7 Efficiency of fail2ban as IDPS in 2020.

The story continued with Idhom, Wahanani, and Fauzi's 2020 research, which confirmed Fail2Ban's efficacy in an Intrusion Detection and Prevention System (IDPS). Their use of a centralized collector database for sharing attack information among servers was a significant step in collaborative security measures. This work brought collective defense theory to a realistic situation, improving real-time responsiveness and security posture. However, it was primarily concerned with web servers and did not address the integration of Fail2Ban in containerized settings (Idhom et al., 2020).

2.8 Impact of Brute-force on servers in 2021

In 2021, Ylli, Tafa, and Marku investigated the impact of brute force hits on server load and used Fail2Ban to counteract them. Their research revealed important insights into Fail2Ban's practical applicability in maintaining server performance during attack. While it emphasized the need of performance monitoring, the study focused on SSH and general server security, ignoring email servers and containerized systems (Ylli et al., 2021).

2.9 Analysis of alternative tools to fail2ban

The narrative concluded in 2023 with Makopa et al.'s investigation on forensic analysis in IoT networks utilizing the Raspberry Pi 4 Model B and open-source technologies. Their integration of Fail2Ban, Wazuh, and Suricata for complete monitoring in IoT contexts demonstrated a unique use of cybersecurity tools. This study demonstrated the changing environment of cybersecurity and the importance of diverse monitoring systems. However, it concentrated on specific hitting scenarios and lacked thorough setup, limiting the findings' repeatability and generalizability (Makopa et al., 2023).

Throughout these improvements, the research demonstrates a significant shift in cybersecurity, from fundamental email security principles to complex, integrated defensive mechanisms. Each research added to the breadth and depth of expertise by covering multiple areas of cybersecurity and emphasizing the importance of ongoing adaptation and innovation in response to increasing threats. Fail2Ban's integration across several contexts, as well as its use in collective defence and adaptive systems, highlight the significance of comprehensive security measures in today's digital landscape.

Contrarily, this thesis, "Securing Postfix Mail servers using Fail2ban and containerization," tries to close this gap by focusing on email servers' vulnerability to brute force attacks. Although Fail2Ban has been used in a variety of scenarios to improve security, such as safeguarding web servers and SSH services, my study aims to modify and use this technology particularly to email server security. By using Fail2Ban to block the IP addresses of attackers attempting to breach Postfix servers, this report will try to create a customized solution that solves the specific issues that email systems, particularly Postfix based environment, confront.

Furthermore, this work combines the revolutionary concept of containerizing Postfix and its dependencies using Podman, which improves email server deployment and maintenance while preserving strong security protections. This combination of Fail2Ban and containerization not only improves email server security, but it also benefits the larger information technology sector by providing a viable, scalable solution to a serious issue.

In conclusion, while earlier research has provided the framework for identifying and mitigating brute force attacks on a variety of systems, my thesis focuses on using Fail2Ban especially for email server security. By doing so, I hope to create a comprehensive solution that strengthens email communications' resilience against illegal access, so contributing to the continued evolution of cybersecurity standards in the information technology sector.

3 Research Methodology

This research investigated the effectiveness of Fail2Ban and containerization in enhancing the security of Postfix email servers. The methodology consisted of several phases, including the design and implementation of the security framework, evaluation of the framework's effectiveness, and analysis of the results. This section offers a detailed overview of the equipment, software, and techniques used for creating containers and databases in a virtual environment. The research utilized a lab environment with multiple virtual machines (VMs) configured to simulate a typical organizational email infrastructure. The setup included:

3.1. **Equipment:** For this project, VirtualBox 6.0 is used to build VMs where configuration is as mentioned below:

Table 1: Host Machine configuration:

S.NO	Configuration Parameters	Value
1	Ubuntu	22.04
2	Hard Disk	250GB
3	RAM	16GB
4	CPU	Intel(R) Core (TM) i7-4810MQ CPU @ 2.80GHz
5	Cores	4
6	SSD	Yes

Table 2: Guest Machines Configuration

S.NO	Configuration Parameters	Value
1	Rocky Linux	9.2
2	Hard Disk	25 GB
3	RAM	2 GB
4	CPU	Intel(R) Core (TM) i7-4810MQ CPU @ 2.80GHz
5	Cores	2
6	SSD	No
7	IP address	192.168.1.18

3.2. **Software and Tools**: The software stack consisted of Rocky Linux 9 operating system where Podman Docker and Fail2Ban is installed on the machine itself. Mail Services – Postfix, Dovecot, DKIM and Dmarc are installed in containers. Being a demo lab, Mariadb is configured in separate machine. Fail2Ban was customized to monitor logs of Postfix and Dovecot inside container to ban IPs exhibiting suspicious behaviour. Furthermore, Hydra tool was used to support the complete testing scenario.

Table 3: Information about all the software and tools used.

S. No	*Service/Tool*	*Version*	*Port*	*Description*	*Container {Y/N}*
1	*Rocky Linux*	*9.4*	*25, 587,143*	*Operating System*	*N*
2	*Python*	*3.2*	*N/A*	*Used for installing podman-compose*	*N*
3	*Pip*	*3*	*N/A*	*Used for installing podman-compose*	*N*
4	*podman*	*podman version 4.9.4-rhel*	*N/A*	*Tool for administrating containers*	*N*
5	*Postfix*	*3.5.9*	*25, 587*	*SMTP*	*Y*
6	*Dovecot*	*2.3.16*	*143*	*IMAP*	*Y*
7	*Dkim*	*2.11.0*	*12305*	*email authentication method*	*Y*
8	*Dmarc*	*1.4*	*54321*	*a framework for email authentication*	*Y*
9	*Hydra*	*9.4*	*N/A*	*Testing tool*	*N*
10	*MariaDB*	*10.5.22*	*3306*	*Database*	*N*
11	*Virtualbox*	*6.0*	*N/A*	*Platform for Virtualisation*	*N*

3.3. **A Containerized Email Server Infrastructure**: To provide a secure and efficient email server environment, a containerized architecture using Podman was used. A dedicated network adapter with a CIDR of 172.18.0.0/24 was built to connect the virtual machine's IP address to the container network. This architecture guarantees

that containers, although having separate internal IP addresses, connect with the outside world using the virtual machine's IP address.

The email system comprises several containers as mentioned below:

3.3.1. *Postfix*: This container provides SMTP services and is assigned a static IP of 172.18.0.2 within the smtpsrver_mailnet network (name given to bridge adapter created for container's communication). To enable external access to this container, port forwarding is configured to expose ports 25 and 587 on the virtual machine's IP (192.168.1.18).

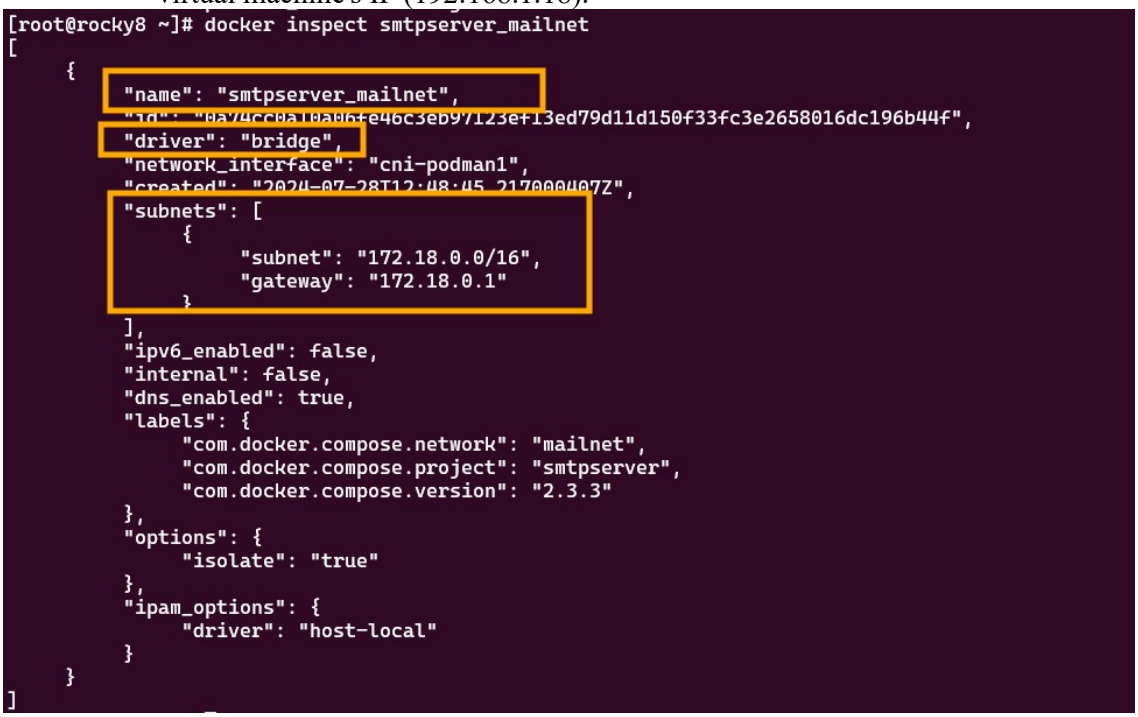

Figure 1: Information about bridge adapter for container

3.3.2. *Dovecot*: Responsible for IMAP services, this container used the same smtpserver_mailnet network adapter and to have an IP of 172.18.0.3. Like Postfix, port 143 was exposed through virtual machine's IP to allow traffic for IMAP on IP = 192.168.1.18.

3.3.3. *DKIM*: To enhance email security, a DKIM container was deployed. For isolation purposes, this container remains internal to the host with a static IP of 172.18.0.4. Communication with the Postfix container is established on port 12308. Below was the custom parameter added to config file (/etc/opendkim.conf).

UserID opendkim:opendkim
Socket inet:12301@172.18.0.4

3.3.4. *DMARC*: As a complementary email authentication mechanism, a DMARC container was also configured in isolation with an IP of 172.18.0.5 to allow communication from Postfix container over the port 54321. Following was the parameter added to config file (/etc/opendmarc.conf)

UserID opendmarc:mail
Socket inet:54321@172.18.0.5

To streamline the management and orchestration of these interdependent containers, Docker Compose (docker-compose.yml) was employed. This tool facilitated the simultaneous

creation and configuration of the Postfix, Dovecot, DKIM, and DMARC services within a cohesive environment. A containerized architecture was built to increase flexibility, scalability, and security by isolating crucial email components.

3.4. *Database Server*: MariaDB is latest open source and more reliable server. In order to save server's local user, this report discusses keeping users in database for central authentication of SMTP and IMAP service especially in case of multiple postfix instances.

4 Designing and Customizing Security Framework

The environment is customized using containerization techniques through which only required ports of mail servers were only exposed. Below is the brief diagram explaining same:

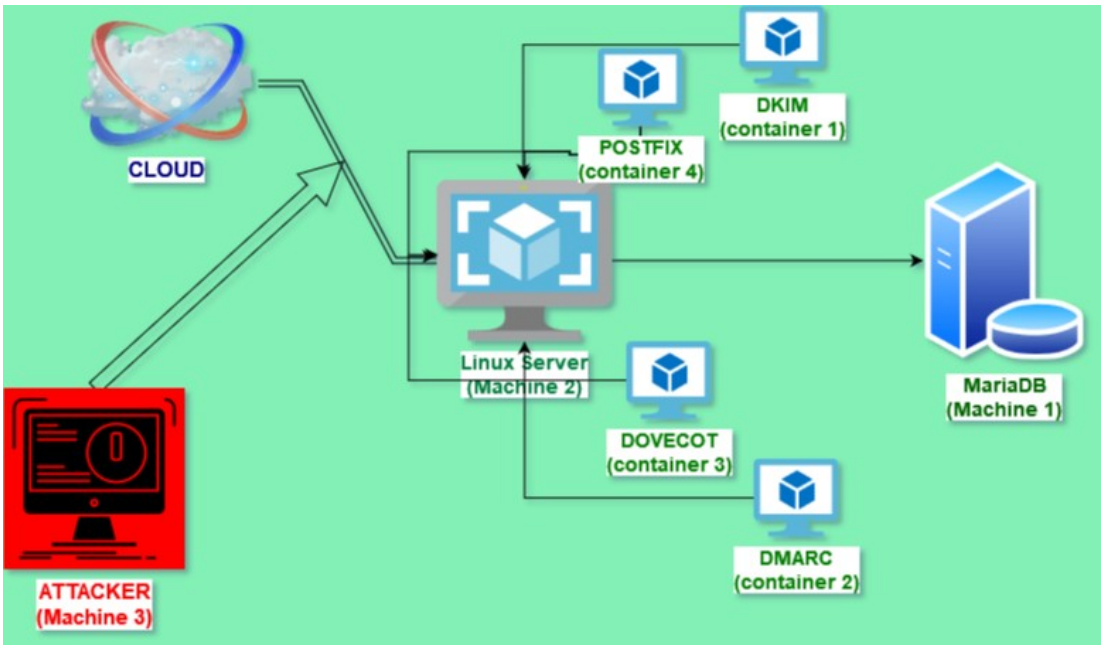

Figure 2: Demonstrating architecture for the research.

4.1 **Postfix tunning**:

Postfix was customised with multiple restrictions to safeguard from Brute-force and other type of attacks. Important ones are listed below:

smtpd_sasl_type = dovecot

smtpd_sasl_path = inet:172.18.0.3:24

smtpd_client_restriction = permit_mynetworks, permit_sasl_authenticated, reject_rbl_client zen.spamhaus.org, reject_rbl_client sbl.spamhaus.org, reject_rbl_client xbl.spamhaus.org, reject_rbl_client dnsbl.sorbs.net, reject_rbl_client b.barracudacentral.org, permit

smtp_recipient_restriction = check_sender_access hash:/etc/postfix/sender_access, permit_mynetworks, permit_sasl_authenticated, reject_sender_login_mismatch, reject_unauth_destination

smtp_sender_restriction = permit_mynetworks, permit_sasl_authenticated, reject_unknown_sender_domain

maillog_file = /var/log/maillog

virtual_transport = lmtp:unix:private/dovecot-lmtp

virtual_mailbox_maps = mysql:/etc/postfix/mysql-virtual-mailbox-maps.cf

virtual_alias_maps = mysql:/etc/postfix/mysql-virtual-alias-maps.cf

4.2 **Dovecot tunning-** *part 1*

Being IMAP running inside container, it was a challenge to communicate postfix with dovecot. To solve this challenge, Dovecot was made separately available on port 24 but not exposed through host machine like port 25, 587 and 143. Below parameters were defined in /etc/dovecot/10-master.conf to start the communication from Postfix container to Dovecot:

```
}
inet_listener {
port = 24
}
```

Apart from above major change, few other config file were also modified to tune IMAP for multiple purposes, like only listen on ipv4, Maildir location for receiving mails, enable authentication through remote database, etc. Details are mentioned below in table:

Table 4: Tunning of dovecot service

S.NO	FILENAME	Required Changed	Remarks
1	/etc/dovecot/dovecot.conf	Listen = *	Listen only on IPv4 address
2	/etc/dovecot/conf.d/10-auth.conf	disable_plaintext_auth = no	Not diabling plain_text
3	/etc/dovecot/conf.d/10-auth.conf	auth_mechanisms = plain login	Allowing base64 login
4	/etc/dovecot/conf.d/10-mail.conf	mail_location = maildir:~/Maildir	User's mail receiving location
5	/etc/dovecot/conf.d/10-master.conf	unix_listener /var/spool/postfix/private/auth	Defining authentication process, if received through postfix
6	/etc/dovecot/conf.d/10-ssl.conf	ssl = no	In testing environment, SSL is not available but in enterprises it is must to have.
7	/etc/dovecot/conf.d/10-ssl.conf	KeyFIle	If no SSL, then comment it.
8	/etc/dovecot/conf.d/10-ssl.conf	CertFile	If no SSL, then comment it.

4.3 **Dovecot tunning-** *part 2*

Further, dovecot was customised to communicate with MariaDB for only user authentication process. This authentication will be responsible for authenticating at both - SMTP and IMAP ports, i.e Postfix and dovecot. Quick glance to the config file (/etc/dovecot/dovecot-sql.conf.ext) is given below:

```
driver = mysql
connect = host=192.168.1.16 dbname=mailserver user=mailuser password=password
default_pass_scheme = SHA512-CRYPT

password_query = SELECT email as user, password FROM virtual_users WHERE email='%u';
#user_query = SELECT email as user, 1001 as uid, 1002 as gid, concat('/home/', SUBSTRING_INDEX(email, '@', 1), '/Maildir') as home, concat('/home/',
 SUBSTRING_INDEX(email, '@', 1), '/Maildir') as mail FROM virtual_users WHERE email='%u';
user_query = SELECT email as user, 1003 as uid, 12 as gid, concat('/home/', SUBSTRING_INDEX(email, '@', 1), '/Maildir') as home, concat('/home/', SU
BSTRING_INDEX(email, '@', 1), '/Maildir') as mail FROM virtual_users WHERE email='%u';
```

Figure 3: enable Dovecot to authenticate form remote database.

Note: Even though authentication will be done through remote database, system user is still required to get the mails in their respective home directory. Alternatively, mail can be stored in database too, but that option needs to be explored.

4.4 **Integration of DKIM and DMARC with Postfix:**

To mitigate the risk of email spoofing and man-in-the-middle attacks, DKIM and DMARC were implemented. DKIM provides email authentication, while DMARC establishes a framework for handling emails that fail authentication checks. Below are the parameters appended in config file of postfix (/etc/postfix/main.cf) to activate the application of DKIM and DMARC record on each mail send from it.

```
milter_protocol = 2
milter_default_action = accept
smtpd_milters = inet: 172.18.0.4:12301, inet:172.18.0.5:54321
non_smtpd_milters = inet: 172.18.0.4:12301, inet: 172.18.0.5:54321,
unix:private/opendkim, unix:private/opendmarc
```

5 Security Implementation

5.1. **Fail2Ban Configuration**: Fail2Ban was built with unique filters and actions to defend the containerized Postfix service from brute force attacks. By default, and as can be concluded by report shared by Manolache et al. (2014), while Fail2Ban works well with services that operate directly on the host, dealing with containerized apps proved difficult. Due to the nature of container networking, Fail2Ban originally failed to block attacker IP addresses (e.g., 192.168.1.15) targeting Postfix and Dovecot ports (25, 587 and 143) accessible via Podman.

To fix this issue, Fail2Ban was modified by adding a new iptables chain to the action directory. This change enabled the successful blockage of attacker IP addresses on the specified Postfix ports. The particular modifications made are detailed in the action directory's configuration files (dockeraction.conf, dockeraction_postfix.conf and dockeraction_postfixsasl.conf).

For a quick reference below is a screenshot of file described in action directory (/etc/fail2ban/action.d/dockeraction.conf,

/etc/fail2ban/action.d/dockeraction_postfixsasl.conf and
/etc/fail2ban/action.d/dockeraction_postfix.conf)

```
[Definition]
actionstart = iptables -N f2b-npm-dovecot
              iptables -A f2b-npm-dovecot -j RETURN
#             iptables -I FORWARD -p tcp -m multiport --dports 0:65535 -j f2b-npm-dovecot
              iptables -I FORWARD -p tcp -m multiport --dport 143 -j f2b-npm-dovecot

actionstop = iptables -D FORWARD -p tcp -m multiport --dport 143 -j f2b-npm-dovecot
             iptables -F f2b-npm-dovecot
             iptables -X f2b-npm-dovecot

actioncheck = iptables -n -L FORWARD | grep -q 'f2b-npm-dovecot[ \t]'

actionban = iptables -I f2b-npm-dovecot -p tcp --dport 143 -s <ip>  -j DROP

actionunban = iptables -D f2b-npm-dovecot -p tcp --dport 143 -s <ip>  -j DROP
```

Figure 4: Action file of fail2ban tuned for dovecot
(/etc/fail2ban/action.d/dockeraction.conf)

```
[Definition]
actionstart = iptables -N f2b-npm-postfix
              iptables -A f2b-npm-postfix -j RETURN
#             iptables -I FORWARD -p tcp -m multiport --dports 0:65535 -j f2b-npm-postfix
              iptables -I FORWARD -p tcp -m multiport --dport 25 -j f2b-npm-postfix

actionstop = iptables -D FORWARD -p tcp -m multiport --dport 25 -j f2b-npm-postfix
             iptables -F f2b-npm-postfix
             iptables -X f2b-npm-postfix

actioncheck = iptables -n -L FORWARD | grep -q 'f2b-npm-postfix[ \t]'

actionban = iptables -I f2b-npm-postfix -p tcp --dport 25 -s <ip>  -j DROP

actionunban = iptables -D f2b-npm-postfix -p tcp --dport 25 -s <ip>  -j DROP
```

Figure 5: Action file of fail2ban tuned for postfix (port 25)
(/etc/fail2ban/action.d/dockeraction_postfix.conf)

```
[Definition]
actionstart = iptables -N f2b-npm-postfix-sasl
              iptables -A f2b-npm-postfix-sasl -j RETURN
#             iptables -I FORWARD -p tcp -m multiport --dports 0:65535 -j f2b-npm-postfix
              iptables -I FORWARD -p tcp -m multiport --dport 587 -j f2b-npm-postfix-sasl

actionstop = iptables -D FORWARD -p tcp -m multiport --dport 587 -j f2b-npm-postfix-sasl
             iptables -F f2b-npm-postfix-sasl
             iptables -X f2b-npm-postfix-sasl

actioncheck = iptables -n -L FORWARD | grep -q 'f2b-npm-postfix-sasl[ \t]'

actionban = iptables -I f2b-npm-postfix-sasl -p tcp --dport 587 -s <ip>  -j DROP

actionunban = iptables -D f2b-npm-postfix-sasl -p tcp --dport 587 -s <ip>  -j DROP
```

Figure 6: Action file of fail2ban tuned for postfix (port 587)
(/etc/fail2ban/action.d/dockeraction_postfixsasl.conf)

5.2. **Email Authentication Essential**: DKIM is an email authentication method that verifies the authenticity of an email message's sender. Whereas DMARC builds on DKIM and SPF (Sender Policy Framework) to provide a framework for email authentication. These two essentials - DKIM and DMARC, will be configured in separate containers to provide an additional layer of security for mailing domain.

5.3. **Containerization**: To create an isolated and efficient email server environment, Podman was chosen over Docker because of its inherent security benefits and daemonless design. Podman Compose was used to simplify the management of these containerized services by describing their dependencies and configurations in YAML

files. A custom docker-compose.yml file was created to specify the email server components (Postfix, Dovecot, DKIM, and DMARC), their images, network connections, and volume mounts. This configuration file controlled the construction and interaction of various containers, guaranteeing proper communication and data durability. Using Podman Compose, a single command may start, stop, or rebuild all services, considerably simplifying the administration process (Afreen, 2023). By taking this strategy, a strong and secure email infrastructure was created.

```yaml
#version: '3.8'

networks:
  mailnet:
    driver: bridge
    ipam:
      config:
        - subnet: 172.18.0.0/16

services:
  postfix:
    build:
      context: ./postfix
    container_name: postfixf
    ports:
      - "25:25"
      - "587:587"
    volumes:
      - /opt/container/log/maillog:/var/log/maillog
      - /opt/container/home:/home
      - /opt/container/private/auth:/var/spool/postfix/private/auth
    networks:
      mailnet:
        ipv4_address: 172.18.0.2

  dovecot:
    build:
      context: ./dovecot
    container_name: dovecotf
    ports:
      - "143:143"
    volumes:
      - /opt/container/log/maillog:/var/log/maillog
      - /opt/container/private/auth:/var/spool/postfix/private/auth
      - /opt/container/home:/home
    networks:
      mailnet:
        ipv4_address: 172.18.0.3

  dkim:
    build:
      context: ./dkim
    container_name: dkimf
    volumes:
      - /opt/container/log/maillog:/var/log/maillog
    networks:
      mailnet:
        ipv4_address: 172.18.0.4

  dmarc:
    build:
      context: ./dmarc
    container_name: dmarcf
    volumes:
      - /opt/container/log/maillog:/var/log/maillog
    networks:
      mailnet:
        ipv4_address: 172.18.0.5
```

Figure 7: content in docker-compose.yml to build containers

```
FROM rockylinux/rockylinux:9

RUN yum update -y && yum install -y vim rsyslog postfix postfix-pcre

COPY main.cf /etc/postfix/main.cf
COPY master.cf /etc/postfix/master.cf
RUN chown root:root /etc/postfix/ma*

WORKDIR /var/spool/postfix

EXPOSE 25 587

CMD ["/bin/sh", "-c", "/usr/sbin/rsyslogd && /usr/sbin/postfix start-fg"]
```

Figure 8: content of YAML file of postfix to build container through docker-compose.yml

Hence, this setup provided isolation and simplified the management of the email server environment. Then the two containers – *postfix and dovecot*, were configured to communicate with MariaDB database to authenticate the users.

5.4. **Centralized User Authentication**: MariaDB database was explored to provide centralised solution to user authentication process for Postfix and Dovecot. This will help in segregating the system users from users for mailing purpose. Therefore, it will be an added advantage to fight with brute-force attacks.
For research and development purpose, a separate machine with basic database, named as mail server, was created. Details are as mentioned below:
IP address: 192.168.1.16
Database name: mailserver
Port: 3306

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mailserver         |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.002 sec)

MariaDB [(none)]> use mailserver;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mailserver]> show tables;
+----------------------+
| Tables_in_mailserver |
+----------------------+
| virtual_users        |
+----------------------+
1 row in set (0.000 sec)

MariaDB [mailserver]> select * from virtual_users;
+----+----------------------------------------------------------------------------------------------------------------+---------------+
| id | password                                                                                                       | email         |
+----+----------------------------------------------------------------------------------------------------------------+---------------+
|  1 | $6$20f722a9be8a84cb$mQiMQV0fDvagrQKxlZOwvJT/RxmB8DjQ7l/tE5y34bdDWI7qgt.f3fWxSlxPV.uekUuqhjQDIdLCEa7KE1kWV1 | user@tech.in  |
|  2 | $6$b907171d4ff47109$otzGRgUaIAJ4.NyPo/tViOh81akCdO593ynTsSenrZsnxbaEGyqzj./RBuMDeY7KJ8UNREPtS0dPXLzFK8AmF0 | user4@tech.in |
|  3 | $6$32c5c11ca05ed9be$iohOorHtOIbwtmze/NT7BljNpuKQneJKISAc2OB/eERp8SqNbbtquX/JHOw0tXRRMDQ46ZDBsrXNhzWrpaEs3/ | user5         |
|  4 | $6$a1b97af1f7765de2$1xkt0bMkBfAPtVL1dUdABDI.juGMZL3YwWfiAcKRfIexIBgrT9enzEA6LGxyuCMmqrNuXVKXTcesSg0XgKTkd/ | user5@tech2.in |
|  5 | $6$961558018b5b9c7d$ZpkKGDYZ8C5fABEKzsK94mwP9lkCE5O0pzQt8St3VQsa6l4AigvZ6RdFDySqJR1ofJwo5hSY66xijmK0n0dFj1 | user@tech2.in |
```

Figure 9: database and user information created in Mariadb database.

6 Evaluation

The complete testing environment was built and evaluated using below steps. Thus, later case studies are created based on tests:

6.1. Test Scenarios: To prioritize security and reliability, rigorous testing was conducted on the containerized email server. Security assessments were the primary focus, employing vulnerability scans using tools *nmap* to identify and address potential vulnerabilities. Penetration testing was performed through tools like ***hydra and telnet*** to simulate real-world attack scenarios and evaluate the system's resilience. Configuration audits were conducted to ensure adherence to security best practices and to identify misconfigurations. While performance benchmarks were also considered, using tools like *top* command to measure CPU load, memory usage, and network bandwidth, no significant impact on system performance was observed due to the simulated environment and relatively low load conditions. While performance benchmarks were also considered to assess email delivery speed and resource utilization, the primary emphasis was on fortifying the system against threats.

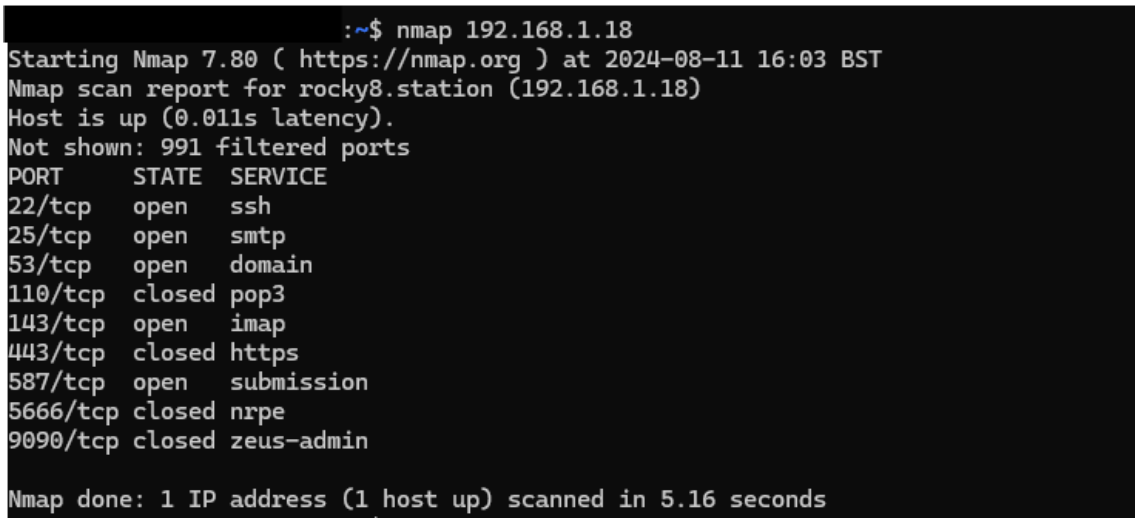

Figure 10: Vulnerability scanning through nmap stating port 25,587 and 143 are exposed.

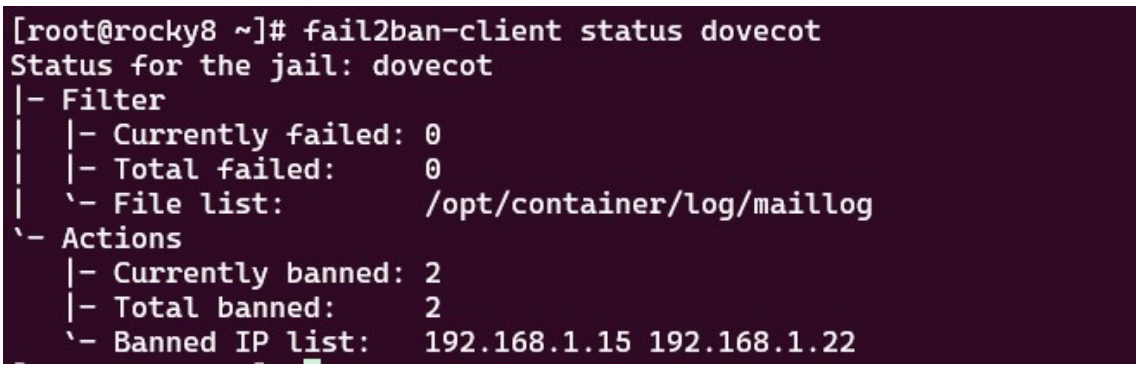

Figure 11: IPs banned at IMAP port (Dovecot service) due to twice failed attempts.

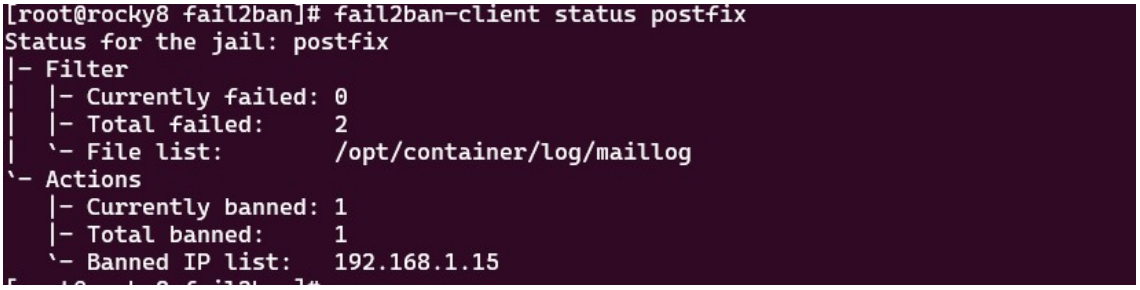

Figure 12: IPs banned to access SMTP port (Postfix service) due to twice fail attempts.

6.2. Case Studies: A case study was conducted on a Carnegie Mellon University. It focused on the implementation of a collective defense system against password guessing attacks in an enterprise environment (Manolache et al, 2014). Here are details about type and nature of attack:

Organization: Carnegie Mellon University
Environment: The group runs a network of computers that provide a variety of services, including SSH (Secure Shell) and IMAP (Internet Message Access Protocol) for email. The university's network is a traditional corporate setting, with various users accessing shared resources.

6.2.1. Type of Attack: The biggest threat noticed was password guessing hits, which are classic brute-force attacks that attempt to compromise user accounts by repeatedly trying different login and password combinations from a dictionary.

6.2.2. Frequency and Impact: During the monitoring period, each networked machine received an average of 1.4 password guessing assaults every day. These assaults not only devoured bandwidth but also created a danger of denial of service, particularly for mail and web servers. The assaults were distinguished by:

6.2.3. Single Attacker Single Target: An attacker makes several username/passwords attempts on a given machine.

6.2.4. Multiple Attackers Single Target: Multiple attackers hitting the same machine at the same time, frequently utilizing different portions of the same dictionary.

Solution implemented considering the Case Study

Collective Defense System: The authors recommended a network-wide security approach that included the deployment of a system capable of exchanging attack data among different computers. This system successfully detected and responded to assaults using the fail2ban and DenyHosts tools.

Key Features of the Solution

6.2.5. Distributed Database: The technology used a shared database to record individual attack occurrences throughout the network. This enabled a coordinated reaction to assaults.

6.2.6. Preemptive Blocking: By examining logs and attack patterns, the system could detect and stop intruders before they could effectively breach accounts. The authors discovered that around 20% of individual assaults may be stopped before they reach their targets.

6.2.7. **Low Ban Thresholds**: The system used relatively low ban levels (5-10 unsuccessful attempts) to swiftly detect and ban attackers, reducing the likelihood of successful assaults and allowing genuine users to restore access fast.

6.2.8. **Information Sharing**: The system provided critical information such as attacker and target IP addresses, attack timing, and targeted service. This allowed for a collective defensive mechanism while maintaining privacy concerns.

Mitigation of Threats

6.2.9. **Improved Detection**: Enhanced capacity to detect coordinated assaults using shared data.

6.2.10. **Reduced Impact**: Proactive steps reduced the impact of assaults on network resources, ensuring service availability.

6.2.11. **Scalability**: The system's architecture enabled successful scalability, resulting in faster detection times as additional computers joined the defense pool.

This case study focuses on the successful installation of a collective defensive system at Carnegie Mellon University, which considerably improved its security against password guessing assaults.

6.3. Data Collection and Analysis

6.3.1. **Data Collection**: Raw data included log files from Postfix and Fail2Ban records of banned IPs, and performance metrics (CPU, memory usage, and network bandwidth).

6.3.2. **Data Analysis**: The analysis focused on the following metrics:

6.3.2.1. **Detection Rate:** Fail2Ban was configured to reject requests from illegal IP addresses after two tries, indicating that the program effectively reduced brute force attacks by monitoring and blocking suspicious IP addresses.

6.3.2.2. **False Positive Rate:** Considering a situation in enterprise, where an application server would be connecting to email server and fail to connect due to wrong credentials or frequently users mistyping passwords, IP will get blocked on email servers. So, to avoid blocking, parameter = ignoreip was explored to exempt it from blocking. It can be configured in jail.local file of fail2ban (/etc/fail2ban/jail.local). But because the focus of this study is to handle brute-force attack, it is not included in configuration.

6.3.2.3. **System Performance:** Being a local environment, no critical issue was noticed but analyzing previous work done, it can be concluded that in a bigger distributed system environment log files may become larger in size due to which continuous monitoring can impact on performance. Such situations shall be handled by lowering the threshold of failed attempts or increasing the memory of the server.

6.3.2.4. **Block Duration:** This tool fail2ban has the ability to keep increasing the length of banning every time an IP address gets blocked. But in this study,

bantime is configured to -1 which means IP is banned permanently (McKay, 2023).

6.4. Validation and Verification

6.4.1. Cross-Validation: The results were validated through cross-validation, using different subsets of data to ensure consistency and reliability.

6.4.2. Verification: The setup and configuration were verified by conducting a thorough review of the system logs and confirming that all components functioned as expected.

6.5. Comparative Analysis

As it is already reported by Manolache *et al*. (2014) that at Carnegie Mellon University, Fail2Ban was used as part of a collective defensive system, considerably improving the detection and prevention of password guessing attacks, resulting in a preemptive blocking rate of around 20% of individual attempts before they reached their targets. This proactive technique enabled real-time responses to attacks, minimizing attackers' window of opportunity in comparison to older systems, which frequently depended on manual monitoring and had lengthier response times. However, using Fail2Ban offered trade-offs, such as higher CPU and memory utilization because of constant log monitoring, the possibility of false positives that incorrectly banned real users, and the difficulty of setup and maintenance in a distributed system. Furthermore, while strong banning tactics enhanced security, they may have an adverse effect on user experience, especially for individuals who regularly mistype passwords. Overall, while Fail2Ban has proven to be a more successful tool for preventing password guessing attacks, careful consideration of its resource requirements and user impact is required for optimal security in business settings.

This research is done in an isolated environment of virtual machines simulated to enterprise level. So, live data is not captured in this report, but it assures when this research is deployed in live environment, it will secure ports 25, 587 and 143 from brute-force attacks. As per the table shown in “Live Attacks Testing Results” in the report of Dawamsyach, Ruslianto and Ristian (2023, p. 159-160), after implementing fail2ban, after implementing fail2ban, 864 live IPs were blocked permanently that were trying to brute-force attacks. Containerization provided an additional reduction in attack surface by isolating the mailing users from the system users. Additionally, Docker Compose expedited the creation of multiple Postfix instances by streamlining the containerization process. This process will be more helpful for the organization involved in digital marketing or email campaigning.

6.6. Discussion

The research aimed to enhance the security of Postfix email servers by integrating Fail2Ban and containerization, focusing on mitigating brute force attacks. The findings indicate that this approach significantly improved the security posture of the email infrastructure. The deployment of Postfix within containerized environments, coupled with the dynamic IP banning capabilities of Fail2Ban, provided a robust defence against

unauthorized access attempts. The containerization concept for mailing services helped in solving the building of multiple domains in short time span.

Key Findings and Insights

6.6.1. Effectiveness of Fail2Ban: The study confirmed that Fail2Ban effectively detected and blocked a significant percentage of brute force attacks, with a high detection rate and a low false positive rate. This confirms previous findings in the literature that have demonstrated Fail2Ban's efficacy in various server environments. The customization of Fail2Ban filters for Postfix-specific logs proved critical in accurately identifying malicious activities.

6.6.2. Benefits of Containerization: Containerization provided several advantages, including isolation of email server instances, ease of deployment, and resource management. The isolated environment for each Postfix instance ensured that a security breach in one container did not affect others, thus limiting the attack surface. Furthermore, the centralization of attack data across containers enabled a coordinated response to threats, enhancing the overall security posture.

6.6.3. Performance Considerations: The implementation of security measures did introduce some overhead, as indicated by a slight increase in CPU and memory usage. However, this overhead was minimal and did not significantly impact the server's ability to handle legitimate traffic. The system's performance metrics remained within acceptable limits, demonstrating that the security enhancements did not compromise the user experience.

6.6.4. Limitations and Challenges: While the findings of this study demonstrate potential, several limitations must be acknowledged. The simulated attack environment, although extensive, may not fully replicate the multifaceted nature of real-world threats. Furthermore, the research primarily focused on brute force attacks, neglecting a comprehensive evaluation of other attack vectors such as phishing and malware dissemination. The centralized data architecture introduces a potential single point of failure, rendering the system vulnerable to targeted attacks. Finally, the use of Kubernetes for container orchestration revealed a significant resource requirement, potentially limiting its applicability to organizations with constrained budgets.

7 Conclusion and Future Work

This thesis investigated the enhancement of Postfix email server security through the integration of Fail2Ban and containerization. The research was driven by the need to address the growing threat of brute force attacks and the limitations of existing security measures. The study successfully demonstrated that the combination of Fail2Ban's dynamic IP banning capabilities through applying custom script and the isolation benefits of containerization significantly improved the security and resilience of email servers.

Based on the findings of this research, various areas for further investigation are proposed:

7.1 *Expansion to Other Attack Vectors*: Future research should explore the application of Fail2Ban and containerization to defend against a broader range of attack vectors, including phishing, malware distribution, and DDoS attacks over email servers. This would involve developing new filters and actions tailored to these specific threats.

7.2 *Advanced Threat Detection*: Incorporating machine learning techniques could enhance the detection capabilities of the security framework. Machine learning models could analyse patterns in attack data to identify new and evolving threats, enabling proactive defence measures for email services.

7.3 *Scalability and High Availability*: As the current study focused on a controlled lab environment, future work should address the scalability of the solution in larger, real-world deployments. This includes exploring load balancing and high-availability configurations to ensure the system can handle increased traffic and maintain uptime during attacks.

7.4 *Enhanced Privacy Protections*: Given the increasing importance of privacy, future research should integrate privacy-enhancing technologies into the framework. This could involve implementing techniques to implement SSL certificates, anonymize email metadata and protect user identities, addressing the vulnerabilities identified in the current literature.

7.5 *User Experience and Usability*: While security is paramount, it is equally important to consider the user experience. Future studies should investigate the usability of the security framework, ensuring that it does not impede legitimate user activities. This could involve user surveys and testing to gather feedback on the system's impact on day-to-day operations.

7.6 *Real-World Deployment and Testing*: Finally, the implementation of the proposed security measures in real-world environments would provide valuable insights. This includes working with organizations to deploy the solution in production systems and monitoring its performance and effectiveness over time. Such real-world testing would help validate the framework's robustness and adaptability.

Key contributions of this thesis include the customization of Fail2Ban filters for Postfix-specific log analysis, the implementation of a centralized database for sharing attack data across multiple containerized instances, and the establishment of a robust framework capable of detecting and mitigating brute force attacks. The findings showed that the proposed solution achieved a high detection rate with minimal false positives, while the performance impact on the server was negligible. This balance between security and usability is critical for maintaining an effective and user-friendly email infrastructure.

Despite these advancements, the study also acknowledged certain limitations, such as the focus on brute force attacks and the potential single point of failure introduced by centralizing attack data. These challenges highlight the need for ongoing research and development in the field of email server security. Future work should explore broader attack vectors, enhance privacy protections, and refine the system's scalability and usability.

In summary, this thesis has made a meaningful contribution to the field of email server security by demonstrating the practical benefits of integrating Fail2Ban and containerization. The proposed framework not only addresses current security challenges but also provides a foundation for future innovations in protecting email communications. The findings underscore the importance of a multi-layered security approach, combining advanced threat detection and containerization, to safeguard sensitive information in an increasingly digital world.

References

Afreen, S. (2023) *What is Docker Compose: Example, benefits and basic commands*. Available at: https://www.simplilearn.com/tutorials/docker-tutorial/docker-compose [Accessed 11 August 2024].

Artry, J. (2022) *70% of cyberattacks are ransomware and business email compromise*. Available at: https://tech.co/news/70-of-cyberattacks-target-business-email-accounts [Accessed 8 August 2024].

Arzhakov, A. V. and Silnov, D. S. (2016) 'Analysis of brute force attacks with YLMF-PC signature', *International Journal of Electrical and Computer Engineering (IJECE)*, 6(4), pp. 1681-1684. doi:10.11591/ijece.v6i4.pp1681-1684.

Dawamsyach, F., Ruslianto, I. and Ristian, U. (2023) 'Implementation of IPS (Intrusion Prevention System) Fail2ban on server for DDoS and brute force attacks', *CESS (Journal of Computer Engineering, System and Science)*, 8(1), pp. 149-161. doi: 10.24114/cess.v8i1.40259

Ford, M., Mallery, C., Palmasani, F., Rabb, M., Turner, R., Soles, L. and Snider, D., 'A process to transfer fail2ban data to an adaptive enterprise intrusion detection and prevention system', *SoutheastCon 2016.* pp. 1-4. doi:10.1109/secon.2016.7506771.

Idhom, M., Wahanani, H. E. and Fauzi, A. (2020) 'Network security system on multiple servers against brute force attacks', in *2020 6th Information Technology International Seminar (ITIS)*. Surabaya, Indonesia, 14-16 October 2020, pp. 258-262. doi:10.1109/itis50118.2020.9321108.

RedHat (2024) *What is Podman?* Available at: https://www.redhat.com/en/topics/containers/what-is-podman#podman-vs-docker [Accessed: 11 August 2024].

Joyia, A., Ghafoor, A., Sajjad, M. and Choudhary, M. Q. (2013) 'Secure and privacy enhanced email system as a cloud service', in *Eighth International Conference on Digital Information Management (ICDIM 2013)*, Islamabad, Pakistan, 10-12 September 2013, pp. 73-78. doi: 10.1109/ICDIM.2013.6693986.

Keenan, A. (2024) *3 advantages of Podman vs. Docker*. Available at: https://developers.redhat.com/articles/2023/08/03/3-advantages-docker-podman#_1__podman_makes_creating_pods_easy [Accessed: 11 August 2024].

Madi, N. K. M., Salehian, S., Masoumiyan, F. and Abdullah, A. (2012) 'Implementation of secure email server in cloud environment', in *2012 International Conference on Computer and Communication Engineering (ICCCE)*. Kuala Lumpur, Malaysia, 3-5 July 2012, pp. 28-32. doi:10.1109/iccce.2012.6271146.

McKay, D. (2023) *How to secure your Linux server with fail2ban*. Available at: https://www.howtogeek.com/675010/how-to-secure-your-linux-computer-with-fail2ban/ [Accessed 08 August 2024].

Makopa, J., Christopher, A., Shah, R., and Mandela, N. (2023) 'Internet of things (IOT) network forensic analysis using the Raspberry Pi 4 model B and open-source tools', *2023 International Conference on Quantum Technologies, Communications, Computing, Hardware and Embedded Systems Security (iQ-CCHESS)* [Preprint]. doi: 10.1109/iq-cchess56596.2023.10391643.

Manolache, F.B., Hou, Q. and Rusu, O. (2014) 'Analysis and prevention of network password guessing attacks in an enterprise environment', *2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference* [Preprint]. doi:10.1109/roedunet-renam.2014.6955303.

Papadie, R. and Apostol, I. (2017) 'Analyzing websites protection mechanisms against ddos attacks', *2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)* [Preprint]. doi:10.1109/ecai.2017.8166454.

Postel, J. (1982) *"Simple Mail Transfer Protocol", STD 10, RFC 821*. doi: 10.17487/RFC0821.

Ylli, E., Tafa, I. and Marku, F. (2021) 'Examining the server's load average when trying to attack the firewall', in *Proceedings of Sixth International Congress on Information and Communication Technology*, *Lecture Notes in Networks and Systems, vol 235*, pp. 7–14. doi:10.1007/978-981-16-2377-6_2.