Usability Enhancement of IMAP Agent System with Considering Privacy Preservation Using Docker Container

1st Xiuyuan Chen
Tokyo University of Agriculture and Technology
Tokyo, Japan
Rockcxy@net.cs.tuat.ac.jp

2nd Nariyoshi Yamai
Tokyo University of Agriculture and Technology
Tokyo, Japan
nyamai@cc.tuat.ac.jp

3rd Rei Nakagawa
Tokyo University of Agriculture and Technology
Tokyo, Japan
rnakagawa@go.tuat.ac.jp

4th Tomoaki Tsutsumi
University of Tsukuba
Ibaraki, Japan
tsutsumi.tomoaki.gn@u.tsukuba.ac.jp

5th Yong Jin
Tokyo Institute of Technology
Tokyo, Japan
yongj@gsic.titech.ac.jp

***Abstract*—Today, email is a ubiquitous means of communication, with billions of users worldwide relying on it for personal and professional correspondence. Internet Message Access Protocol (IMAP) [1] is a widely used email protocol that allows users to view, organize, and synchronize their email across multiple devices while storing messages on the server. An IMAP Agent [2] is a system that connects to a mail server constantly to retrieve mail as soon as it arrives at the destination server. In this paper, we propose a solution for enhancing the usability of the IMAP Agent System while considering users' privacy preservation. The proposed system seamlessly integrates the creation of Docker containers [3] with the deployment of IMAP Agent, all within a user-friendly software application. The Docker images [4] are built locally on the user's device and then securely uploaded to a private registry, ensuring that sensitive information is neither exposed to external entities nor stored insecurely. This approach not only simplifies user interaction with IMAP systems but also enhances privacy and security by leveraging Docker containers' isolated environments, thereby reducing vulnerabilities to attacks and improving usability for commercial applications. Our approach is validated through a series of simulation experiments that demonstrate its effectiveness in safeguarding user information while significantly improving usability.**

***Index Terms*—Email System, Security, Internet Message Access Protocol**

I. INTRODUCTION

Today, the Internet is widely used worldwide and has become an indispensable part of daily life. According to Oberlo "Email Usage Statistics 2024" [5], the number of people using email is expected to reach 4.48 billion in 2024, which is nearly half of the world's population. That number is projected to grow to over 4.8 billion by 2027. Email is a popular communication tool that enables users to send and receive text, images, files, and other data through network transmission. The email communication process involves the sender writing the email, specifying the recipient's email address, and sending the email to the sender's server through the email client. Mail server protocols play a crucial role in the framework of email. These protocols describe the communication rules that client software and mail servers must adhere to, ensuring a standardized process across various email systems. The primary protocols in this ecosystem include Simple Mail Transfer Protocol (SMTP) [6], Post Office Protocol 3 (POP3) [7], and Internet Message Access Protocol (IMAP) [1], each serving distinct yet complementary roles in the email delivery and retrieval process.

The mail server protocols, particularly IMAP [1], play a crucial role in modern email systems, allowing users to manage their email over a TCP/IP connection [8]. Unlike POP3, which is less favored due to its security drawbacks and method of removing emails from the server after download, IMAP allows emails to remain on the server, supporting multi-device synchronization and enhancing data recovery capabilities. This capability of IMAP to maintain a continuous server connection significantly underpins its usability. Building on this foundation, IMAP Agent [9] is a system that communicates with an email server based on the IMAP protocol. The IMAP Agent allows users to connect to email servers and view, manage, and process emails. Users can perform various operations such as reading, marking, deleting emails, and creating folders. IMAP Agent is important in maintaining security, managing permissions, optimizing performance, and providing users with convenient, flexible, and secure email access methods. IMAP Agent enables connectivity and operation with the mail server upon user provision of their username and password. Thus, users can configure and add desired operations even without administrative or operational authority over the server space.

The advancement of the IMAP Agent system aims to enhance its usability, thus enhancing its potential for future commercialization. After thoroughly examining existing methodologies, Docker emerged as a pivotal technology capable of improving system usability while preserving privacy.

Docker is renowned for its lightweight containerization and rapid deployment capabilities. Encapsulating the IMAP Agent within Docker containers allows users to create isolated environments that simplify interactions and ensure consistency across different computing platforms.

This paper presents a novel approach to enhance the usability of IMAP Agent systems through Docker's containerization and deployment strengths while also safeguarding user credentials and enhancing privacy and security. A critical feature of this approach is enabling users to input their login credentials locally, generate a Docker image, and then upload this image to a private registry. This method simplifies the process for users and ensures that their sensitive information remains secure, addressing privacy and data security concerns.

II. RELATED WORK

A. IMAP Agent

IMAP Agent [2] is a mechanism that detects the arrival of a new mail by consistently maintaining an IMAP session with the mail server and performing processing on the mail. New mail can be detected by receiving a status update notification with the IDLE command [10] or by detecting the status of the IMAP server with the NOOP command. The IMAP Agent system [11] facilitates the addition of new email processing features that are more straightforward and cost-effective than traditional proxy server models. The design supports cross-device sharing of email statuses and transparent processing, eliminating the need for repetitive setups across multiple devices. As an example of IMAP Agent application, reference [12] proposes a solution that utilizes IMAP Agent to constantly monitor an email server for new messages that meet certain user-defined criteria of urgency and importance. When such messages are detected, the system immediately notifies the user through alternative means other than email. This practical application of the IMAP Agent system is shown in Fig.1.

However, the complexity of the current IMAP Agent systems presents significant challenges for general users, impacting their ability to utilize these systems effectively. The intricate setup and management processes require technical knowledge that many users may not possess, thereby limiting the system's usability and broader adoption. To address this, our primary objective is to enhance the usability of the IMAP Agent system. By streamlining the process and making it more accessible, we aim to allow users to manage their email with minimal technical effort.

Additionally, for IMAP Agents to function, they must authenticate with email servers using user credentials (email addresses and passwords). If these credentials are stored in plaintext within the agent's backend code or configuration files, they become visible to administrators and, potentially, anyone accessing the system's files. This storage method inherently lacks security, making credentials susceptible to unauthorized viewing. To protect against the unauthorized disclosure of user credentials, it is imperative to strengthen the security of the IMAP Agent's operational environment.

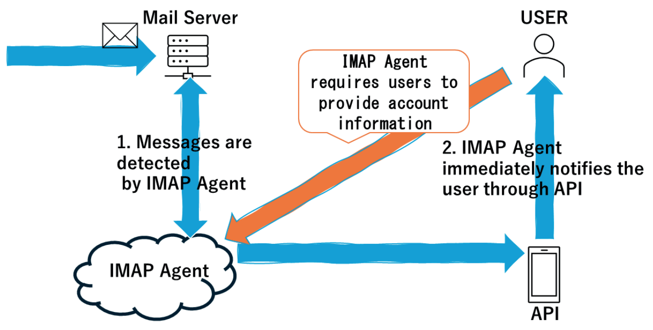

Mail Server
USER
IMAP Agent requires users to provide account information
1. Messages are detected by IMAP Agent
2. IMAP Agent immediately notifies the user through API
IMAP Agent
API

Fig. 1. Practical application of IMAP Agent system and security vulnerability

B. Docker

Docker [3] is an open-source platform that automates the deployment, scaling, and management of applications within lightweight and portable containers. This containerization technology significantly simplifies the process of developing, testing, and deploying applications across different environments, ensuring consistency and efficiency. Docker's architecture revolves around several key components: Docker Engine, Docker Images, and Docker Containers. The Docker Engine is the core component, consisting of the Docker daemon and the Docker client. The daemon runs on the host machine and manages the containers, while the client is the user interface for interacting with Docker. Docker Images [4] serve as the blueprint for Docker containers, containing the application code, libraries, dependencies, and configuration files needed to run the application. These images are created from Dockerfiles, which are simple text files containing the commands to assemble an image. Once built, Docker images are stored in a registry, such as Docker Hub, from where they can be pulled and used to instantiate containers. Docker containers are ephemeral and immutable, meaning they can be easily started, stopped, moved, and deleted without affecting the underlying host or other containers. This leads to a highly flexible and scalable system architecture. The use of containers ensures that applications run consistently regardless of the environment, making Docker an ideal solution for development, testing, and production deployment.

Docker [13] provides efficient resource utilization. Containers are lightweight and share the host system's kernel, which allows for more efficient use of system resources compared to traditional virtual machines. This efficiency leads to faster startup times and lower overhead. Docker's portability is another significant advantage, as containers can run on any system that supports Docker, including various cloud platforms. This portability simplifies the process of moving applications across different environments without modification.

Docker's containerization technology provides a powerful solution for developing, testing, and deploying applications in a consistent, efficient, and secure manner. By incorporating Docker into the IMAP Agent system, we can leverage these benefits to enhance usability, security, and scalability, making the system more robust and user-friendly.

C. Local operation in improving usability and security

In recent years, several systems and tools have been developed to enhance usability and security by allowing users to perform operations locally. This approach not only simplifies the user interaction but also ensures that sensitive information is handled securely within the user's trusted environment. Password Managers such as 1Password [14] exemplify this approach by enabling users to generate and manage passwords locally on their devices. 1Password allows users to create strong, unique passwords for each of their accounts, storing them in an encrypted vault that can only be unlocked with a master password. This vault is stored locally on the user's device, and only encrypted data is synced to the cloud. This means that even if the cloud service is breached, the data remains unreadable without the master password. This local-first approach ensures that users retain full control over their sensitive information, enhancing both security and usability.

Drawing from the above-mentioned prior research, this study proposes a new system designed further to enhance the usability and security of IMAP Agents. The proposed system allows users to perform operations locally to generate a secure Docker image, which is then uploaded to private registries. This approach simplifies the complex process associated with IMAP Agent deployment, enhancing usability while ensuring a high level of security.

III. MOTIVATION

A. Societal Benefit of IMAP Agent System

IMAP Agents have the potential to revolutionize digital communication by enhancing information access, bolstering security, boosting productivity, and supporting educational endeavors. By organizing and prioritizing emails, they ensure vital information is readily accessible, protect against cyber threats, streamline inbox management, and facilitate critical communications in education. These agents embody a leap forward in making digital interactions more efficient, secure, and meaningful. Here are a few examples [12] of how IMAP agents can benefit the whole society.

- **Increased Productivity**: By automating the organization of incoming emails based on user-defined rules, IMAP Agents can significantly reduce the time individuals and organizations spend managing their inboxes. This leads to increased productivity and allows more time for time-constrained tasks.
- **Support for Education and Learning**: IMAP Agents can streamline communication between educators and students, ensuring that important educational materials, announcements, and feedback are highlighted. This can enhance the learning experience and ensure that students do not miss out on essential information.

B. Enhancing Usability of IMAP Agent System

To make the IMAP Agent System more user-friendly and commercially viable, it is crucial to simplify the process for users to manage their email credentials securely. The existing IMAP Agent systems are overly complicated, making it difficult for users to interact with them efficiently. Our goal is to develop a solution where users can easily input their credentials, generate Docker images, and upload them to a private registry, all within a seamless and intuitive software application.

- **Simplified User Interaction**: By enabling users to input their credentials locally and automate the creation of Docker images, we reduce the technical burden on the user. This approach makes the system more accessible to non-technical users and ensures that the process is straightforward and efficient.
- **Enhanced Privacy and Security**: While focusing on usability, this approach also enhances security by ensuring that sensitive information is processed locally and securely. By leveraging Docker's isolated environments, we reduce the risk of unauthorized access and data breaches.

By focusing on these improvements, we aim to advance the usability of the IMAP Agent System, making it more practical and appealing for broader adoption and commercial use. This approach highlights the complementary benefits of enhancing usability and improving data security.

IV. SYSTEM DESIGN

A. Innovative Usability Framework

This study aims to enhance the usability of the IMAP Agent System by creating a more user-friendly and commercially viable solution. Current complexities in managing email credentials pose significant challenges for users, impacting their experience and efficiency. Our research proposes a novel system that simplifies this process by integrating Docker containers into the deployment of IMAP Agents within an intuitive software application.

Docker images and containers provide a robust security framework. Docker images are immutable, ensuring consistency and preventing unauthorized changes, and they encapsulate all dependencies, isolating them from the host system. Containers leverage Linux namespaces for process and file system isolation, and control groups to manage resource allocation, preventing denial-of-service attacks. By default, containers run with reduced privileges, adhering to the principle of least privilege, and can be further secured with customizable security profiles using AppArmor and SELinux. Seccomp profiles limit the system calls a container can make, reducing the attack surface. Network isolation and firewall rules control traffic to and from containers, enhancing security. Additionally, Docker supports automated vulnerability scanning and regular updates to maintain security. This multi-layered approach ensures that applications running in Docker environments are protected from a wide range of security threats, making Docker a highly secure platform for modern application deployment. By leveraging Docker's capabilities, our system enables users to securely input their email credentials on their local device, generate Docker images, and upload them to a private registry. This ensures that sensitive information is processed locally, reducing the risk of data breaches and enhancing privacy.

B. Implementation Process

The overview of the enhanced IMAP Agent system using Docker is shown in Fig. 2. The implementation of this system involves several key steps:

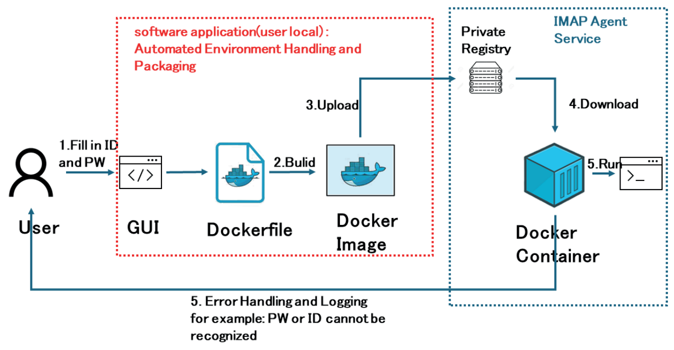

software application(user local):
Automated Environment Handling and Packaging
Private Registry
IMAP Agent Service
3.Upload
4.Download
1.Fill in ID and PW
2.Bulid
5.Run
User
GUI
Dockerfile
Docker Image
Docker Container
5. Error Handling and Logging
for example: PW or ID cannot be recognized

Fig. 2. Overview of the Enhanced IMAP Agent System Using Docker

- **User Interface for Credential Input**: Develop a secure application or web interface where users can input their email credentials. This interface ensures that the credentials are entered securely and are not exposed to unauthorized parties.
- **Docker Image Creation**: Once the credentials are entered, the application generates a Dockerfile that includes the necessary configuration to run the IMAP Agent. The Dockerfile is then used to build a Docker image locally on the user's device.
- **Secure Image Upload**: After the Docker image is created, it is uploaded to a private Docker registry. This registry is securely managed to ensure that only authorized users can access and deploy the images.
- **Deployment of IMAP Agent**: The Docker image is then pulled from the private registry and deployed on the target server, where it runs the IMAP Agent in an isolated and secure environment.

By focusing on these improvements, our motivation is to advance the usability of the IMAP Agent System, making it more practical and appealing for broader adoption and commercial use. This approach highlights the dual benefits of enhancing user experience and improving data security in a complementary manner.

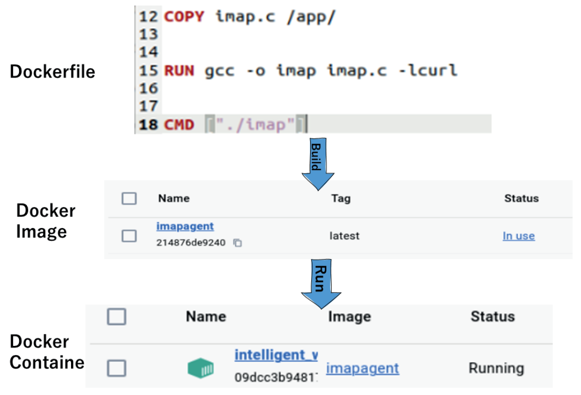
V. IMAP Agent System Using Docker Container

To enhance the usability and security of the IMAP Agent System, this paper introduces a method that leverages Docker Container technology. Encapsulating the IMAP Agent within a Docker Container simplifies the process for users to manage their email credentials securely while also improving the system's overall security.

To encapsulate the IMAP Agent within a Docker container and ensure its isolated operation, the following steps are undertaken [15]:

- **Dockerfile Creation**: Begin by crafting a Dockerfile that specifies the build process of the Docker image, defining the environment and dependencies required by the IMAP Agent.
- **Copying Executable Files**: Transfer the IMAP Agent's executable files into the Docker environment to ensure all necessary components are included within the container.
- **Building the Docker Image**: Utilize the docker build command to create an Ubuntu-based Docker image labeled "IMAP Agent" within the directory that contains the Dockerfile.
- **Running the IMAP Agent**: Activate the IMAP Agent by executing the docker run command, which launches the agent within the newly constructed Docker container.

This process, illustrated in Fig. 3, demonstrates the steps involved in creating and deploying a Docker container. This approach allows the IMAP Agent to operate efficiently within its isolated environment, showcasing the feasibility and effectiveness of containerization.

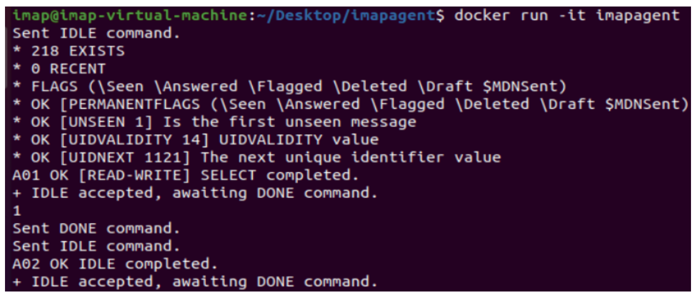


```
COPY imap.c /app/


RUN gcc -o imap imap.c -lcurl


CMD ["./imap"]
```

	Name	Tag	Status
☐	imapagent 214876de9240	latest	In use

		Name	Image	Status
☐		intelligent_v 09dcc3b9481	imapagent	Running

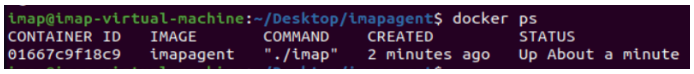
Fig. 3. The process of creating a Docker container



```
imap@imap-virtual-machine:~/Desktop/imapagent$ docker run -it imapagent
Sent IDLE command.
* 218 EXISTS
* 0 RECENT
* FLAGS (\Seen \Answered \Flagged \Deleted \Draft $MDNSent)
* OK [PERMANENTFLAGS (\Seen \Answered \Flagged \Deleted \Draft $MDNSent)
* OK [UNSEEN 1] Is the first unseen message
* OK [UIDVALIDITY 14] UIDVALIDITY value
* OK [UIDNEXT 1121] The next unique identifier value
A01 OK [READ-WRITE] SELECT completed.
+ IDLE accepted, awaiting DONE command.
1
Sent DONE command.
Sent IDLE command.
A02 OK IDLE completed.
+ IDLE accepted, awaiting DONE command.
```

Fig. 4. IMAP Agent running with Docker

To monitor the container's activity and the operational status of the IMAP Agent within Docker, the docker ps command



```
imap@imap-virtual-machine:~/Desktop/imapagent$ docker ps
CONTAINER ID   IMAGE        COMMAND    CREATED         STATUS
01667c9f18c9   imapagent    "./imap"   2 minutes ago   Up About a minute
```

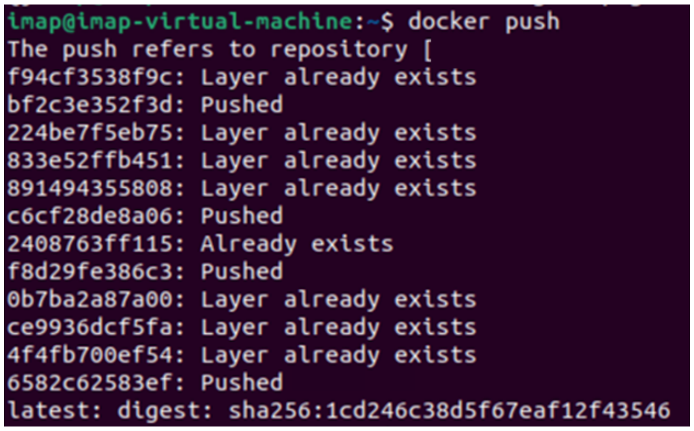
Fig. 5. Docker PS

is utilized. Fig. 4 and Fig. 5 depict the container's running state and the IMAP Agent's active status within Docker, respectively. This encapsulation ensures that only authorized individuals can access the IMAP Agent's functions, safeguarding sensitive user information.

By allowing users to input their credentials locally and automate the creation of Docker images, this method significantly simplifies the process for users. The streamlined process ensures that users, even those with limited technical knowledge, can securely manage their IMAP Agent. Furthermore, Docker's isolated environment enhances system security by reducing the risk of unauthorized access and data breaches. This dual focus on usability and security makes the IMAP Agent System more practical and appealing for broader adoption and commercial use.

VI. Usability Enhancement of IMAP Agent System

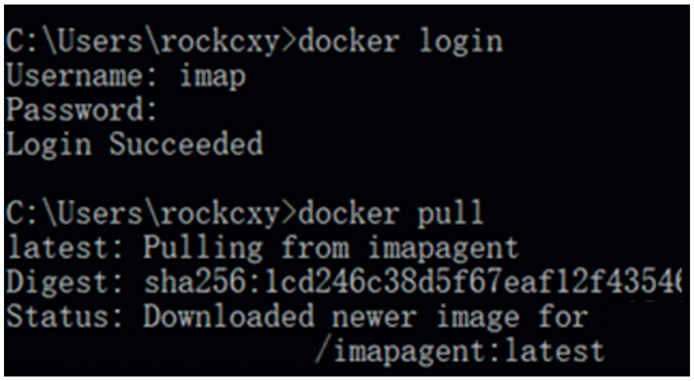
To further enhance the usability and security of the IMAP Agent System, this paper proposes a user-centric method where users can input their email credentials, generate Docker images, and upload them to a private registry. Given that the user's operating environment and the IMAP Agent manager's environment may differ, this study tests the system on both Windows and Ubuntu OS. Once the Docker image is created, it is uploaded to a private Docker registry, ensuring that only authorized users can access and deploy the image. This process involves securely uploading the Docker image to the private registry using secure authentication mechanisms and managing access control to ensure that only authorized users have access to the uploaded Docker images. This user-driven process is illustrated in Fig.6, showing the steps from credential input to Docker image upload.



```
imap@imap-virtual-machine:~$ docker push
The push refers to repository [
f94cf3538f9c: Layer already exists
bf2c3e352f3d: Pushed
224be7f5eb75: Layer already exists
833e52ffb451: Layer already exists
891494355808: Layer already exists
c6cf28de8a06: Pushed
2408763ff115: Already exists
f8d29fe386c3: Pushed
0b7ba2a87a00: Layer already exists
ce9936dcf5fa: Layer already exists
4f4fb700ef54: Layer already exists
6582c62583ef: Pushed
latest: digest: sha256:1cd246c38d5f67eaf12f43546
```

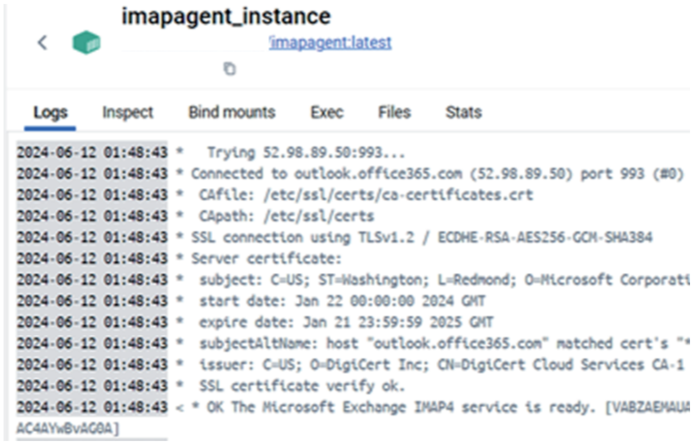
Fig. 6. Upload Docker Image to Private Registry(Ubuntu OS)

Once the Docker image is uploaded to the private registry, the next step involves downloading the image to the target system and deploying it. This process ensures that the IMAP Agent runs in a secure and isolated environment. The authorized user accesses the private registry from the target system and uses secure authentication mechanisms to download the Docker image, ensuring that only authorized personnel can retrieve the image for deployment. After downloading, the user deploys the Docker image using Docker Desktop or a similar Docker management tool, running the Docker container on the target system and setting up the IMAP Agent with the provided configuration. This process is illustrated in Fig. 7 and Fig. 8, showing the steps involved in downloading the Docker image from the private registry and running the Docker container on the target system.



```
C:\Users\rockcxy>docker login
Username: imap
Password:
Login Succeeded

C:\Users\rockcxy>docker pull
latest: Pulling from imapagent
Digest: sha256:1cd246c38d5f67eaf12f4354(
Status: Downloaded newer image for
                      /imapagent:latest
```

Fig. 7. Download Docker Image from Private Registry(Windows OS)



```
2024-06-12 01:48:43 *   Trying 52.98.89.50:993...
2024-06-12 01:48:43 * Connected to outlook.office365.com (52.98.89.50) port 993 (#0)
2024-06-12 01:48:43 *  CAfile: /etc/ssl/certs/ca-certificates.crt
2024-06-12 01:48:43 *  CApath: /etc/ssl/certs
2024-06-12 01:48:43 * SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
2024-06-12 01:48:43 * Server certificate:
2024-06-12 01:48:43 *  subject: C=US; ST=Washington; L=Redmond; O=Microsoft Corporati
2024-06-12 01:48:43 *  start date: Jan 22 00:00:00 2024 GMT
2024-06-12 01:48:43 *  expire date: Jan 21 23:59:59 2025 GMT
2024-06-12 01:48:43 *  subjectAltName: host "outlook.office365.com" matched cert's "*
2024-06-12 01:48:43 *  issuer: C=US; O=DigiCert Inc; CN=DigiCert Cloud Services CA-1
2024-06-12 01:48:43 *  SSL certificate verify ok.
2024-06-12 01:48:43 < * OK The Microsoft Exchange IMAP4 service is ready. [VABZAEMAUA
AC4AYwBvAG0A]
```

Fig. 8. Running Docker Container in Docker Desktop(Windows OS)

By focusing on these steps, our approach not only enhances the usability of the IMAP Agent System but also ensures that the entire process, from credential input to deployment, is secure and user-friendly. This dual focus on usability and security makes the IMAP Agent System more practical and appealing for broader adoption and commercial use.

VII. Conclusion and Future Work

This paper introduced a novel approach to enhance the usability and security of the IMAP Agent System by integrating Docker Containers. Our methodology leverages Docker's efficient containerization to create a user-friendly software application that simplifies the process for users to manage their email credentials securely. By enabling users to input their credentials locally, generate Docker images, and upload these images to a private registry, we ensure that sensitive information is processed securely and efficiently. The primary advantage of this approach lies in its ability to streamline user interaction with the IMAP Agent System, making it more accessible to non-technical users while simultaneously enhancing the system's privacy and security. By containing the IMAP Agent within Docker containers, we create isolated environments that mitigate the risk of unauthorized access and potential data breaches. This not only improves the usability of the system but also provides a robust layer of security, safeguarding user credentials throughout the process.

Our approach is validated through a series of simulation experiments that demonstrate its effectiveness in safeguarding user information while significantly improving usability. This dual focus on usability and security makes the IMAP Agent System more practical and appealing for broader adoption and commercial use.

As we look to the future, our focus will extend toward further enhancing the usability and security of the IMAP Agent System. Future work will involve exploring additional features to make the system even more user-friendly and efficient, such as automated updates and enhanced user interfaces. Additionally, we will investigate methods to further optimize the performance of the system, ensuring it meets the demands of a rapidly evolving technological landscape. This will include the quantitative assessment of performance metrics such as speed, efficiency, and scalability to ensure the system remains robust and adaptable to emerging threats and user needs. By continually refining and improving the IMAP Agent System, we aim to create a solution that not only meets the highest standards of security but also provides an exceptional user experience, paving the way for its widespread adoption and commercial success.

References

[1] A. Melnikov and B. Leiba, "Internet message access protocol (imap) - version 4rev2," RFC 9051, August 2021.

[2] K. Tashiro, N. Yamai, and N. Kitakawa, "Flexible email forwarding system using imap agent without changing server configuration," in *DICOMO2019*, July 2019, pp. 1795–1800.

[3] Docker, "Docker website," 2024. [Online]. Available: https://www.docker.io/

[4] S. Winkel, "Security assurance of docker containers: Part 1," *ISSA Journal*, April 2017.

[5] Oberlo, "Oberlo website," 2024. [Online]. Available: https://www.oberlo.com/

[6] J. Klensin, "Simple mail transfer protocol," RFC 5321, October 2008.

[7] R. Gellens, C. Newman, and L. Lundblade, "Pop3 extension mechanism," RFC 2449, November 1998.

[8] M. West and S. McCann, "Tcp/ip field behavior," RFC 4413, March 2006.

[9] N. Yamai, "Additional services of email enabled by imap agent," in *DICOMO2018*, July 2018, pp. 1403–1406.

[10] B. Leiba, "Imap4 idle command," RFC 2177, June 1997.

[11] I. Hyodo, N. Yamai, and N. Kitakawa, "A warning system for suspicious emails by message threading and imap agent," in *IOTS2020*, November 2020, pp. 115–116.

[12] X. Chen, T. Murakami, N. Yamai, and R. Nakagawa, "Emergency notification system for important emails using imap agent," IOT056, February 2022.

[13] T. Bui, "Analysis of docker security," arXiv:1501.02967, January 2015.

[14] 1Password, "1password - the world's most-loved password manager," 2024. [Online]. Available: https://1password.com/jp

[15] Docker, "Docker container create," 2024. [Online]. Available: https://docs.docker.com