National College of Ireland

Enhancing Docker Security on Azure Cloud: A Comprehensive Analysis and Mitigation Strategy

MSc Research Project
Cloud Computing

Sri Lakshmi Durga Pavedemukala
Student ID: 22174681

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Sri Lakshmi Durga Pavedemukala
Student ID:	22174681
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	31/01/2025
Project Title:	Enhancing Docker Security on Azure Cloud: A Comprehensive Analysis and Mitigation Strategy
Word Count:	6665
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:	Sri Lakshmi Durga Pavedemukala
Date:	12th December 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):	

Enhancing Docker Security on Azure Cloud: A Comprehensive Analysis and Mitigation Strategy

Sri Lakshmi Durga Pavedemukala
22174681

Abstract

The growing popularity of cloud computing has raised numerous security concerns, particularly in managing containerized applications like Docker on Microsoft Azure. This research identifies the security risks associated with Docker environments and proposes effective measures to mitigate them. The study introduces techniques such as customized Docker files validated against OWASP Top 10 vulnerabilities, automated scanning by leveraging GitHub apps, open-source platforms like Trivy along with other policy-based scripts and Microsoft Defender for Cloud to detect and address vulnerabilities throughout the deployment pipeline such as GitHub Actions. These improvements reduced initial entry points for attacks and enhanced the overall reliability of Docker deployments. The results show improvements in threat resilience and the security of containerized infrastructures. However, challenges remain, particularly in addressing emerging risks in highly dynamic containerized setups. This research emphasizes the need for continuous adaptation to evolving threats in cloud environments.

1 Introduction

Docker is now the solution for agile and robust application deployment in today's complex digital world. Nevertheless, flexibility is always associated with new concerns, therefore, Docker is no exception and such things as Microsoft Azure were pointed out as the key factors that can cause difficulties in multi-tenant environment. The goal of this work is to provide not only deep coverage of Docker security issues specific to running containers in Azure cloud but also offer tactical and long-term approaches to improve container security.

1.1 Background

In recent years Docker has come to the foreground as a new solution to the problems of application deployment, supporting the highest productivity rates of modern development processes. Docker shields applications and their dependencies through Its approach of packing software into micro- containers that allows developers and organizations to create sophisticated software in record-breaking time. By implementing this container-based means, the typical development process has gradually been changed so as to promote better portability across various computing platforms. Therefore, Docker is widely used by DevOps professionals to minimize time and effort of SDLC, shorten the application release cycle, and ensure application scalability.

However, as more and more organizations embraced Docker, the emphasis has been placed on the solution's security issues, especially in the multi-tenant environments. Docker when in cloud environments such as Microsoft Azure requires strong security measures to be put in place. Azure provides a very fast and highly flexible foundation for containers at the same time, managing to embrace a transition to cloud-native environments complicates security considerations. Docker containers reside on the same host infrastructure, and in a multi-tenant model for cloud services, it provides an opportunity if not controlled properly. This is because resources in cloud environments are multitenant, therefore making the environment prone to cross-tenant security threats and any favouritism of unauthorized access and other risks if containers are not securely isolated and monitored.

One interesting point of interest within Azure installations is how endangering a container is other tenants running in the same infrastructure. Security can be an issue when containers are placed under a common structure because the contamination may lead to unauthorized access or data loss, hence the necessity to give adequate concern to container security for every organization that seeks to rent Azure infrastructure for its container solutions. There are other tools that offer solutions for the protection of risks, such as Microsoft Defender for Cloud or Azure Security Centre but those might not solve the complexity of Docker security in cloud platforms. This is so because different aspects of containers – runtime protection, isolation on the network level, access and permissions control as well as the vulnerability discover – need a well-planned and mapped approach. The aim of this work is to focus on Docker security threats, which may occur in the framework of the Azure cloud and present the detailed overview of the existing weaknesses and safety concepts. It therefore becomes clear that while operating in a multiple tenant environment it is possible to achieve container security without these contingency measures being completely passive; an active preventive approach combined with continuous threat analysis is required. Short-term measures are as follows; access control and network compartmentalization to ensure safe container interactions in the short-term while the long- term containment measure includes vulnerability scan automation and the increase in security of container images.

Furthermore, the project will address the question of the use of the Azure-native tools and other solutions to strengthen Docker security to understand how native and third-party tools can be used to bolster a reliable containerized environment. This work will produce findings and clear guides on how to protect Docker containers in Azure and the likely future threats associated with the environment. This way, this work aims at enhancing the security of Docker instances on Azure to a level through which organizations can effectively adopt the technology without much concern about risks associated with it.

1.2 Research aims and objectives

The focus of this research paper is to improve Docker security when using Azure cloud and review key threats affecting the implementation of containers and recommend viable solutions against each concerning issue. Today, the trend is progressing towards the adoption of technologies such as Cloud-native and containerization, making it necessary to have sound security mechanisms to be placed on Docker in multi-tenant environments like Azure. This project endeavours to bake these security issues more systematically into a guide to securing Docker, aimed at Azure users and administrators.

Objectives:

Figure 1: Docker Security

- To Analyze Docker Vulnerabilities in Azure
- To Evaluate Azure's Native Security Capabilities
- To Develop Mitigation Strategies for Securing Docker on Azure
- To Propose a Framework for Continuous Security Monitoring

Research Question:

1. What are the primary security vulnerabilities associated with Docker containers deployed on Azure and how effective are Azure's native security tools in addressing Docker-specific security vulnerabilities?
2. What best practices and configurations can enhance the security of Docker containers within Azure?
3. What role does vulnerability management play in maintaining long-term Docker security within Azure, and what tools best support this?

1.3 Research Gap

Despite the rise of Docker adoption in cloud platforms like Microsoft Azure and the increasing prevalence of containerization technologies, research specifically addressing Docker security in Azure environments remains limited. While general cloud and container security are frequently studied, few studies focus on Azure-specific security concerns. Many researchers examine Docker security broadly without analysing Azure's unique security dynamics, hindering findings' applicability to Azure users due to provider-specific threats and controls. Although Azure offers numerous built-in security tools, existing literature lacks a critical evaluation of their effectiveness in mitigating Docker-specific risks. Furthermore, while general Docker security guidance is available, detailed strategies tailored for Azure remain sparse, limiting actionable insights for organizations. The absence of frameworks for continuous Docker security monitoring in Azure further compounds the issue, as persistent scanning is vital for identifying emerging threats. Finally, practical case studies demonstrating Docker security integration in Azure are rare,

leaving theoretical strategies largely untested in real-world applications.

1.4 Problem Statement

Docker along with other technologies in containers are quickly becoming critical in organizational applications and many appear that are based on Microsoft Azure provide a solid good foundation for creating these applications. Although, Docker made improvements as to efficiency, scalability and portability new threat vectors have emerged that agile attackers may employ. This research concludes that even though many security policies exist in the cloud domain, the majority of Azure users remain largely ignorant of Docker- associated threats. As such, current secure controls are inadequate depending on the lack of understanding of the Australian security features and threats specific to Docker. Additionally, there is limited research on how Docker is secured on Azure while some prior work generates "black boxes" that make secure application deployment more challenging. To address these challenges this project evaluates Docker security on Azure, classify existing security tools and propose strategies for Docker security management contributing to the enhanced security of containerized applications to respond to the emerging threats in cloud computing environment.

2 Related Work

Microsoft Azure has gained much attention in the area of container security because the security of containers is central to several security concerns that arise from cloud computing and using container technology. The known cloud environments for their nature are exposed to inherent risks, including unauthorized access and data breaches, and the shared infrastructure approach widens those concerns. Microsoft Azure has accreditation from standard regulations such as ISO 27001 and HIPAA and therefore enhances security; however, as (Alkhatib et al.; 2024) explain that despite being efficient, the hypervisor layer of Azure creates particular risks in the multi-tenant environment.

The new approach labeled as DevSecOps prescribes integration of security in the early phases of software development which is discussed by (Mallikarjunaradhya et al.; 2023). This approach overlays the concerns across the development lifecycle, helping container security on Azure by Reinforcing secure development principles. However, (Muzumdar et al.; 2024) notes that, as much as DevSecOps cuts out risks, security on Azure containers also features threat analysis and access continually and more so with Docker taking root in cloud settings.

Defender for Cloud by Microsoft provides extended security capabilities, namely real-time threat detection and appropriate vulnerability management targeted for organizations' container-based workloads. The authors of (Lee et al.; 2023) emphasize the use of such tools like Defender that are native to the Azure platform; they explained how these tools are very effective in matters pertaining to isolation and tracking of containers. However, (Loaiza Enriquez; 2021) highlighted that the CSPM frameworks in Azure could be fine-tuned to identify misconfigurations and bolster custom security in real-time, particularly for high-risk workloads.

Container security also means mitigating of the new threats that are associated with the Docker platform. (Wong et al.; 2023) describe how to use STRIDE threat modelling to avoid Docker dangers. They focus on network separation and runtime protection, while improper segregation in the containers increases privileges. This is important given (Zhu

and Gehrmann; 2021) investigated the vulnerability of Docker environments and showed that even small errors can cause instability, especially in cases where containers use the same network interfaces as Azure.
Self-built Docker implementation has revolutionized organizations that use massive applications, especially when scaling them quickly and maintaining orchestration within AKS. However, (Friman; 2024) opines that security risks may be at risk since AKS clusters can create lateral movement paths for attackers if the RBAC is poorly done. (Ghazizadeh et al.; 2024) recommend the use of automated security testing in AKS to recheck the containers' settings, propose to use Metasploit and DAST for better recognition of Kubernetes-related threats.
To mitigate the effect of threats on data integrity, (Gupta et al.; 2016) suggests that containers should be grouped into secure virtual networks, a pattern that is well aligned with increasing complexity in Azure today. One such platform that allows such secure deployment is the Microsoft Azure Container Instances (ACI), which decreases the necessity of direct interaction with VMs, and optimizes for scalability according to (Morrison; 2023). But (Singh and Aggarwal; 2022) further explain that Veracode or Snyk are third-party scanning tools to provide additional security as the nature of microservices architectures is hard to mitigate or trace using exclusively Azure tools.
Azure has its security frameworks as the core of the container security though (Chauhan and Shiaeles; 2023) call for the constant improvement of the same since they are under constant attack. They have suggested increasing auditing and the identification of anomalies, which, according to (Mustyala and Tatineni; 2021), may be improved by implementing such Kubernetes strategies as Network Policies and Namespaces to manage and regulate traffic within containers in Azure.
The focus of research is shifting to how Azure can protect Docker environments and the need to remain vigilant in preventing threats and maintaining policies. (Alouffi et al.; 2021) argued that Azure has durable security, which erases international standards for ISO 27001 and ensures data confidentiality and integrity. Still, the research of (Alkhatib et al.; 2024) reveals that Azure has the potential of becoming vulnerable because of its inherent hypervisor layer and shared architecture if resources are no isolated.
One of the strategies that may be used to address these risks is DevSecOps, where information security is built into the development life cycle. According to (Mallikarjunaradhya et al.; 2023) DevSecOps helps in protecting the deployment of containers through code quality compliance and vulnerability assessments at each developmental phase. However, (Muzumdar et al.; 2024) notes that although DevSecOps underpins security, containerization on Azure needs at least a more focused security layer, especially about Docker's runtime and network segregation.
Microsoft Defender for Cloud delivers fundamental security features that are native to the Azure ecosystem. However, Defender helps to scan for vulnerabilities in real-time in containerized environments to enable constant monitoring of the containers. According to (Lee et al.; 2023), Defender is an admirable tool that is capable of analyzing and tracking the activities of the containers to identify any suspicious attempts made towards gaining unauthorized access. However, (Loaiza Enriquez; 2021) points out that while adopting CSPM frameworks on Azure is helpful, further development of CSPM can be made to allow for better real-time discernment of obscured configuration difficulties that often occur in modern cloud structures.
Some of the studies also emphasize that key strategies for chronicling Docker are also related to the utilization of a container-oriented perspective to deal with Docker-specific

risks. As established by (Wong et al.; 2023), STRIDE threat modelling framework is employed to classify possible threats in Docker environments, with interactions between Docker instances, and failure to control network setups as key vulnerabilities. This is how they suggest dividing container networks and applying runtime protection against escalation of privileges and movement within hosted applications in Azure. (Zhu and Gehrmann; 2021) insist that even minor misconfigurations in Docker containers can lead to serious system vulnerability, especially in multi-tenant environments like Azure where there is a high risk of containers sharing an interface.

Another prominent focus in the literature is Azure Kubernetes Service (AKS), which is a main platform for Docker application scaling. In (Friman; 2024) they point out that AKS clusters may pose lateral movement risk if access control policies are not properly implemented to the fullest measure of detail.

Pipelines of Docker Images are also a key in cloud container security processes Since the use of Docker images is so prominent in numerous applications, the safety of the images themselves needs to be addressed as well. For data security and preserving the credibility of Docker arrangements, (Gupta et al.; 2016) suggests the containers should be grouped within secured virtual network spaces. Microsoft's Azure Container Instances (ACI) offer such stirred deployments decreasing reliance on the VMs and amplifying scalability. With this infrastructure, the containerized applications can be scaled and deployed in a way that is time efficient. As (Morrison; 2023) indicates ACI's containerized structure simplifies the process of avoiding intricate VM management, third-party tools such as Varicode and Snyk are further security layers that address vulnerabilities found in microservices that are hard to identify.

This is equally reflected in the literature where a further need is pointed to in order to enhance the security of containers optimized for Azure environments. (Chauhan and Shiaeles; 2023) propose improvements in auditing methods and the execution of anomaly detection frameworks as these methodologies are effective in cloud native architecture. (Mustyala and Tatineni; 2021) suggest that the deployments of Kubernetes on Azure require stronger Policy, for instance, Network Policies, Namespaces, to enhance a significant level of container isolation and control the traffic flow within clusters.

In summary, evidence has indicated strong support for the best practices that can be grouped into a multi-layered security approach including utilizing native Azure tools in conjunction with third party and monitoring software for secure containerized application. Recent studies uncover the need for security updates targeting Azure to maximize the shields for Docker and AKS against new age threats.

2.1 Azure's Native Security Features and Challenges

Microsoft Azure is recognized for its robust container security capabilities, underpinned by compliance with industry standards like ISO 27001 and HIPAA (Alkhatib et al.; 2024). However, its hypervisor layer, while facilitating multi-tenancy, introduces vulnerabilities in resource isolation within shared environments. Tools like Defender for Cloud enhance security by offering real-time threat detection and monitoring (Lee et al.; 2023). Despite these strengths, challenges remain, as highlighted by (Loaiza Enriquez; 2021), who identifies misconfiguration issues within Azure's CSPM frameworks, underscoring the need for continual improvement to safeguard high-risk workloads.

2.2 Strategies for Enhancing Container Security

Authors	Focus	Findings	Research Gaps
(Alkhatib et al.; 2024)	Azure container security challenges	Identified vulnerabilities in Azure's hypervisor layer in multi-tenant environments.	Lack of detailed resource isolation strategies for preventing hypervisor-related risks in multi-tenant setups.
(Mallikarjunaradhya et al.; 2023)	DevSecOps integration for container security	Demonstrated benefits of DevSecOps in reinforcing secure development principles.	Insufficient focus on runtime security and continuous network threat assessment in dynamic containerized setups.
(Lee et al.; 2023)	Real-time threat detection with Defender for Cloud	Showed the effectiveness of Defender in container isolation and tracking.	Limited customization options in CSPM frameworks to address misconfigurations in high-risk environments
(Wong et al.; 2023)	STRIDE threat modeling for Docker security	Highlighted network separation and runtime protection to mitigate Docker-specific risks.	Need for improved techniques to address privilege escalation in poorly isolated container networks.
(Friman; 2024)	Security risks in Azure Kubernetes Service (AKS)	Identified lateral movement risks due to poorly implemented RBAC policies.	Lack of comprehensive strategies to automate RBAC configurations and prevent lateral attacks in AKS clusters.

Authors	Focus	Findings	Research Gaps
(Singh and Aggarwal; 2022)	Role of third-party tools for microservices security	Recommended using Veracode and Snyk for detecting microservices vulnerabilities.	Limited exploration of native Azure tools' integration with third-party tools for comprehensive microservices security.

3 Methodology

The methodology for this project is structured into clear steps to address the research objectives in a systematic step and to provide a secured secure pipeline for Docker applications.

3.1 Problem Definition

The research starts by defining the problem as securing Docker containers in the Microsoft Azure environment. This step involved looking at current studies and industry practices to find problems and weaknesses in existing security measures. The recently updated OWASP Top 10 Vulnerabilities framework can be used as the primary guideline, ensuring the project focus on the most important problems in container security as these frameworks provide security strategies and best practices to reduce risk in docker files.

3.2 Data Collection and Resource Selection

Based on the problem statement, key resources have to be identified to support the research:
Customized Docker file: A unique docker file needs to be created with required configurations to reduce security risk. A public image may contain unnecessary dependencies, compromising the project.
GitHub Repository: The Docker file will be committed to a repository, enabling version control and collaboration.
Python Flask Application: A Python application must be developed to scan Docker file. The application analyzes docker file configurations, commands, and dependencies in detail and generate reports for the file which has vulnerabilities.
To enhance the research, tools such as Trivy and Microsoft Defender for Cloud were configured to perform vulnerability scans and provide real-time security insights. These tools are additional security layers for the docker image to protect from vulnerabilities.

3.3 Experimental Setup

The experimental phase focus on deploying Docker containers in Azure environments configured with security best practices:
Controlled Deployment: Containers has to be launched in Azure Kubernetes Service

(AKS) and Azure Container registry (ACR), with Role-Based Access Control (RBAC) and network policies in place.
Scanning and Validation: Before deployment, the Python application scans the docker files for vulnerabilities, if any issues are detected then that file will be stopped to proceed further until the issue is resolved.
Minimal Permissions: Containers has to be configured with minimal permissions and security rules to provide a secure environment.

3.4 Application of Security Measures

After the experimental setup, several security measures were implemented:
Vulnerability Scans: Trivy a security tool can be used in build phase to scan Docker files and container images for vulnerabilities as a additional security layer.
Configuration and Policies: Network isolation and strict access control policies were enforced within Azure Kubernetes clusters to minimize potential attacks.
Runtime Monitoring: Microsoft Defender for Cloud helps in monitoring runtime environments for incidents, providing alerts and recommendations for unusual behavior.
These measures ensured the pipeline addressed security risks during both build and runtime phases, enhancing the overall system security.

3.5 Sampling and Preparation

Sample Gathering: To maintain control over configurations and dependencies a docker file can be customized, as the public docker images might come with vulnerabilities.
Preparation: Docker files will be scanned and standardized using the Python Flask application to ensure completeness with security standards.
Containers need to be updated to their latest versions before deployment to reduce vulnerabilities from outdated dependencies.
Azure Kubernetes clusters should configured with standardized network and access control settings to ensure consistency during testing.

3.6 Measurement and Data Collection Techniques

Measurement Techniques: Key metrics include the number of vulnerabilities detected, unauthorized access attempts, and network disturbance alerts. The Github application, Trivy and Microsoft Defender for Cloud generated detailed logs for these metrics.
Data Processing and Calculations: Raw data from logs and reports were imported into a data analysis platform for processing. Key calculations included:
Incident Frequency: Tracking unauthorized access attempts over time to assess the likelihood of security breaches.
This methodology ensured a comprehensive approach to securing Docker applications, addressing vulnerabilities during development, deployment, and runtime monitoring. The inclusion of advanced tools and systematic validation processes established a reliable framework for container security in cloud environments.

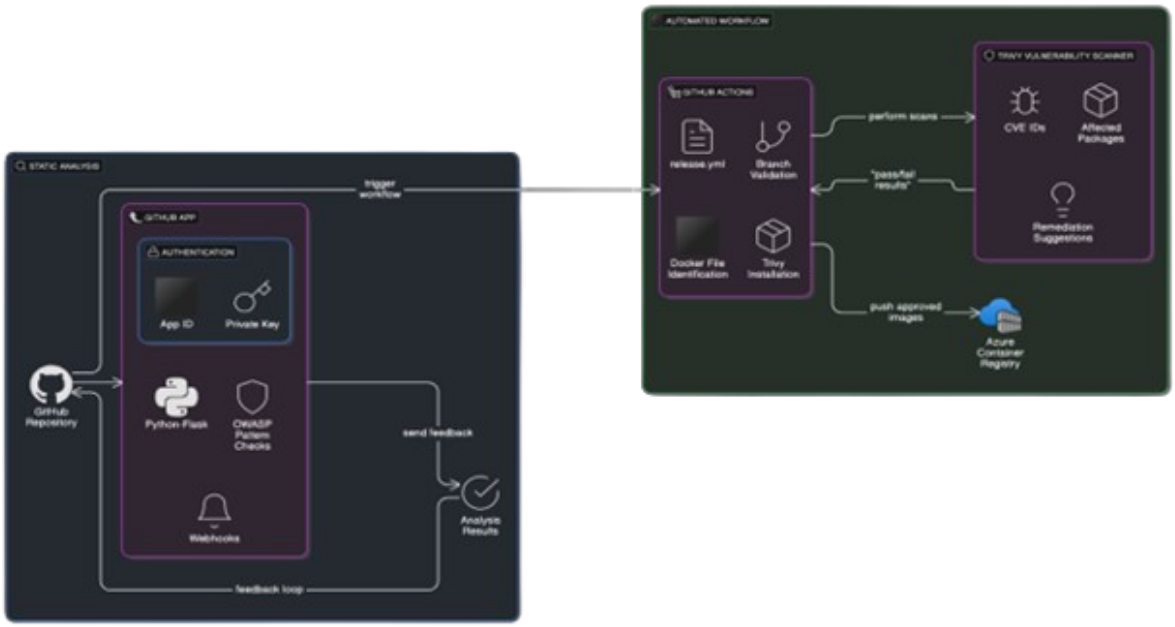

Figure 2: Architectural Diagram

4 Design Specification

This project focuses on building a highly secure and automated process for analyzing Docker files, creating container images, and deploying them safely which combines customized tools, existing frameworks, and cloud infrastructure to provide strong security at every step of the container process.

4.1 Architectural Overview

The design combines static analysis, automated workflows, and runtime monitoring. The main parts of the architecture are:

1. **GitHub App:** Performs static analysis of Docker files to detect vulnerabilities and generates detailed feedback to developers.
2. **GitHub Actions Workflow:** Helps in automating the processes of building, scanning, and pushing Docker images.
3. **Azure Container Registry (ACR):** Provides secure repository for the Docker images which are validated by GitHub Application.
4. **Azure Kubernetes Service (AKS):** For containerized apps, Kubernetes offers a safe and adaptable environment.
5. **Microsoft Defender For containers:** it monitors the runtime environment through Microsoft defender against vulnerabilities and compliance issues.

4.2 Architectural Diagram

Include a diagram illustrating the following:

- GitHub App is connected to the GitHub repository via webhooks. It analyses Docker files during commits and pull requests.
- GitHub Actions Workflow is linked to the GitHub repository for processing Docker files and scanning images by using the security scanning tool Trivy.
- Container Registry receives validated images from the workflow and stores them securely.
- Kubernetes Cluster will Pulling images from azure container Registry for deployment and hosting the containerized application.
- Microsoft Defender monitors the azure Kubernetes cluster for runtime vulnerabilities.

4.3 GitHub App

The GitHub App is a core component designed to perform static analysis on Docker files which is created using Python and Flask, the GitHub app integrates with GitHub APIs to provide real-time feedback to developers.

- **Configuration:** The app uses a unique App ID and a private key to authenticate requests. It is further configured to trigger events pushes and pull requests via webhooks.
- **Static Analysis:** The app scans Docker files against predefined patterns to detect vulnerabilities. These patterns align with OWASP guidelines and include checks for insecure practices like:
  - o Using the latest tag for base images.
  - o Running containers as root.
  - o Missing health monitoring instructions.
  - o Granting excessive permissions.
- **Reporting:** Results are sent to the GitHub Checks tab. Developers receive detailed descriptions of vulnerabilities, their impact, and suggestions to resolve the issues.

4.4 GitHub Actions Workflow

The workflow automates critical steps, ensuring efficient and consistent processing of Docker files. Defined in a release.yml file, it performs the following:

1. Branch Validation ensures the workflow which runs only on approved branches such as main or release.
2. **Docker file Detection:** It finds all Docker files in the repository for processing.
3. Trivy scans Docker images for vulnerabilities and it generates reports which include CVE IDs, affected packages, and helps to resolve the issues.
4. If the docker image pass all the vulnerability checks it proceed further to build and push in container registry.

4.5 Infrastructure Components

- Azure Container Registry works as a secure storage solution for validated Docker images, and it ensure docker image is readily available for deployment.
- Azure Kubernetes Service provides a highly managed environment for deploying

and scaling the containerized application with high availability and integrates with container registry for pulling images.

- Microsoft Defender for Containers adds runtime security by continuously monitoring the Kubernetes cluster. It detects vulnerabilities in running containers, provides detailed reports, and suggests mitigation strategies.

5 Implementation

The implementation of the project has a detailed step by step procedure followed to provide a secure and automated pipeline for docker file analysis, image building, and deployment by discussing the practical integration of tools and entire processes involved to achieve best results.

5.1 Static Analysis with GitHub App

The project started by creating a GitHub App which is designed to analyze all the Docker files that are committed in GitHub repository to analyses potential vulnerabilities. This GitHub app was developed using Python-Flask and integrated with GitHub through the APIs. A unique App ID is used for identification; to enhance security a private key is activated for authentication. Webhooks are implemented to ensure that the app automatically activate whenever a push or pull request is generated in the repository.
The major functionality of the GitHub App is to scan all the Docker files line by line against predefined patterns based on OWASP guidelines. These patterns flagged issues such as:

- **Using the latest tag: python:**latest images are flagged as the latest image can create because it might lead to unreliable builds.
- **Running as root:** Docker files which run as a USER root are flagged because it has risk of involving vulnerabilities to avoid non-root users to minimize risk.
- **Missing HEALTHCHECK instructions:** The app identifies Docker files lacking health monitoring commands and suggest adding HEALTHCHECK to improve container health management.
- **Granting permission:** Commands like RUN chmod 777 were flagged as they provide full access to everyone as permissions as files should provide only necessary permissions.

Results of the analysis were sent directly to the GitHub Checks tab, where developers received feedback which included detailed descriptions of each vulnerability, its OWASP category, and practical remediation suggestions. For example, a missing health check would prompt the developer to include a health monitoring mechanism.

5.2 Automated Workflow with GitHub Actions

In next stage containerization pipeline is automated to ensure streamlined the process, consistency and reducing manual effort using GitHub Actions release.yml workflow file. It was created to manage tasks like scanning, building, and pushing Docker images to Azure Container Registry.
The workflow starts by validating the triggering branch to ensure that only authorized branches, such as main or release could execute the pipeline and then it identifies all

Docker files in the repository using automated commands. Trivy, an open-source vulnerability scanner, is installed within the workflow to perform security scans on the Docker images.
Trivy scans provided comprehensive reports, highlighting vulnerabilities categorized as HIGH or MEDIUM. Each report generates the following reports:

- CVE ID are unique identifiers for vulnerabilities identification.
- **Affected packages:** Details of components requiring attention will be generated.
- Recommend ways or provide information on how to fix the issues, such as upgrading to a patched base image.

Only images that passed the Trivy scans were allowed to progress to the next stage. These images were built and tagged with specific version information before being pushed to Container Registry. Failed scans generate detailed logs, enabling developers to address the issues before retrying.

5.3 Deployment to Azure Kubernetes Service (AKS)

The images which are already validated and stored in Azure container registry were deployed to an Azure Kubernetes Service (AKS) cluster. Deployment was managed using Kubernetes manifests, which defined resources such as:

- Deployments manage container replicas and ensure high availability.
- Communication between the containers and external clients was facilitated by services.
- To handle HTTP/HTTPS traffic securely controllers are involved.

Post-deployment, the application was tested for functionality, Kubernetes logs and metrics were reviewed to validate stability and performance, ensuring that the deployment have both functional and security requirements.

5.4 Runtime Monitoring with Microsoft Defender

As a final layer of protection, Microsoft Defender for Containers was integrated into the Kubernetes cluster. Defender continuously monitored running containers for vulnerabilities and compliance issues. Its detailed reports provided insights like runtime risks, Affected images and categorizing risks to prioritize fixes

For instance, if a running container used an outdated library with a known CVE, the Defender will flag the issue and suggest updates to be done in the container image. Real-time alerts and recommendations ensured that the application remains secure throughout its lifecycle.

This implementation successfully integrated static analysis, workflow automation, and runtime monitoring into a unified pipeline. By addressing security concerns at every stage—from development to deployment and beyond—the project ensured a reliable and secure process for managing applications.

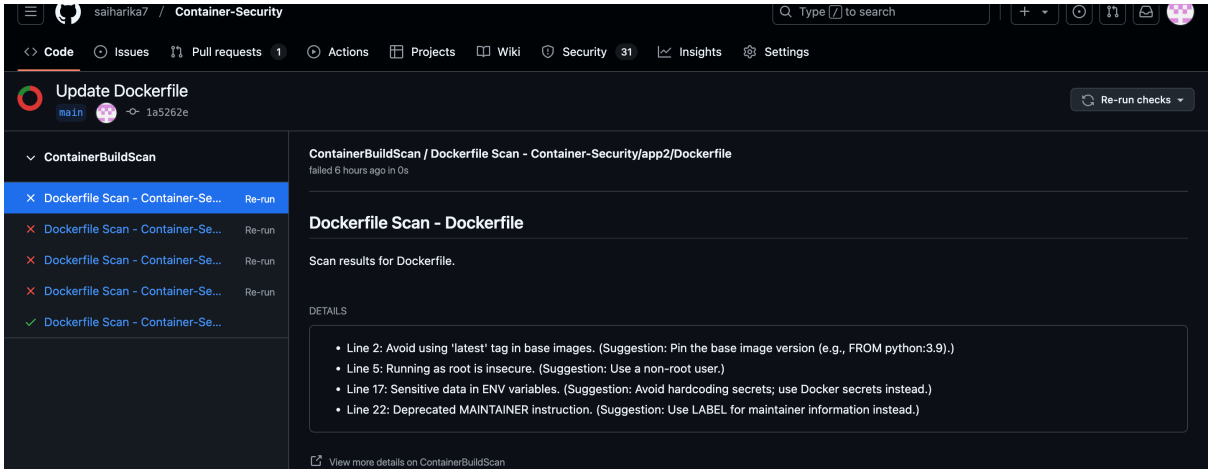

Figure 3: Experiment / Case Study 1

6 Evaluation

The purpose of this section is to provide a comprehensive analysis of the results and main findings of the study as well as the implications of these finding both from academic and practitioner perspective are presented. Only the most relevant results that support your research question and objectives shall be presented. Provide an in-depth and rigorous analysis of the results. Statistical tools should be used to critically evaluate and assess the experimental research outputs and levels of significance.

Use visual aids such as graphs, charts, plots and so on to show the results.

6.1 Experiment / Case Study 1

Prebuild phase – In this phase the aim was to scan through all the available Docker files in the repository precisely in a particular branch and get them scanned against the OWASP Top 10 vulnerabilities along with few other best practices so that to ensure that by following those recommendations strictly we are eliminating a couple of vulnerabilities in the first stage itself.

This stage was successfully executed using a flask based webhook hosted on local machine and service forwarded by smee and as it triggers on every commit made on the branch below screen shot explains us how detailed analysis is done, and the suggestions are provided for the vulnerable Docker files.

6.2 Experiment / Case Study 2

Post the build phases where the Dockerfile are fixed and the area for risk is minimized further in the build phase the container images are created using the same improved Dockerfiles and further the image is scanned using Trivy for CVE's where the container images are scanned for the vulnerabilities if any exists and after thorough scanning the images are pushed further to the ACR

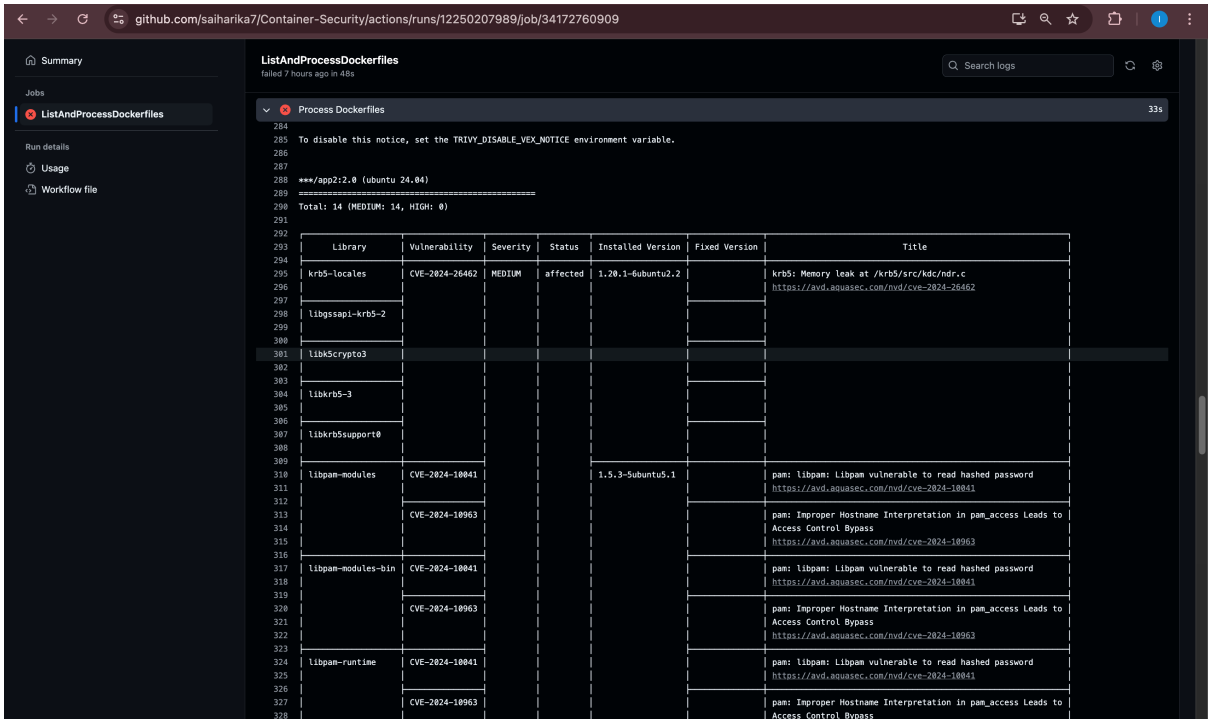

Figure 4: Experiment / Case Study 2

The GitHub Actions workflow is developed for releasing the container image into the Azure container registry post scanning below image can be referred for a better understanding on how the results of the container image scan looks like

6.3 Experiment / Case Study 3

Once the scanned images are pushed to the container registry the security at the infrastructure level becomes an concern but as we are using PaaS – platform as a service Microsoft Defender for cloud can be leveraged to get the pro-active recommendations for service like ACR , AKS , ACS along with others which assists us to secure our footprints right from Docker file till container image storing and consumption in the container orchestration platforms below snip can be referred for better understanding on the same.

6.4 Discussion

The study aimed to evaluate and enhance Docker security on Azure Cloud by addressing vulnerabilities at different stages of the container lifecycle. Three experiments were conducted to test the robustness of the proposed methodology: pre-build scanning of Dockerfiles, post-build vulnerability assessment of Docker images, and runtime security evaluation. The findings from these experiments demonstrate the efficacy of integrating security tools like Trivy, Aqua Security, and Microsoft Defender for Cloud. This discussion evaluates the findings, compares them with prior research, and suggests potential improvements for future work.

Experiment 1: Pre-Build Scanning of Dockerfiles

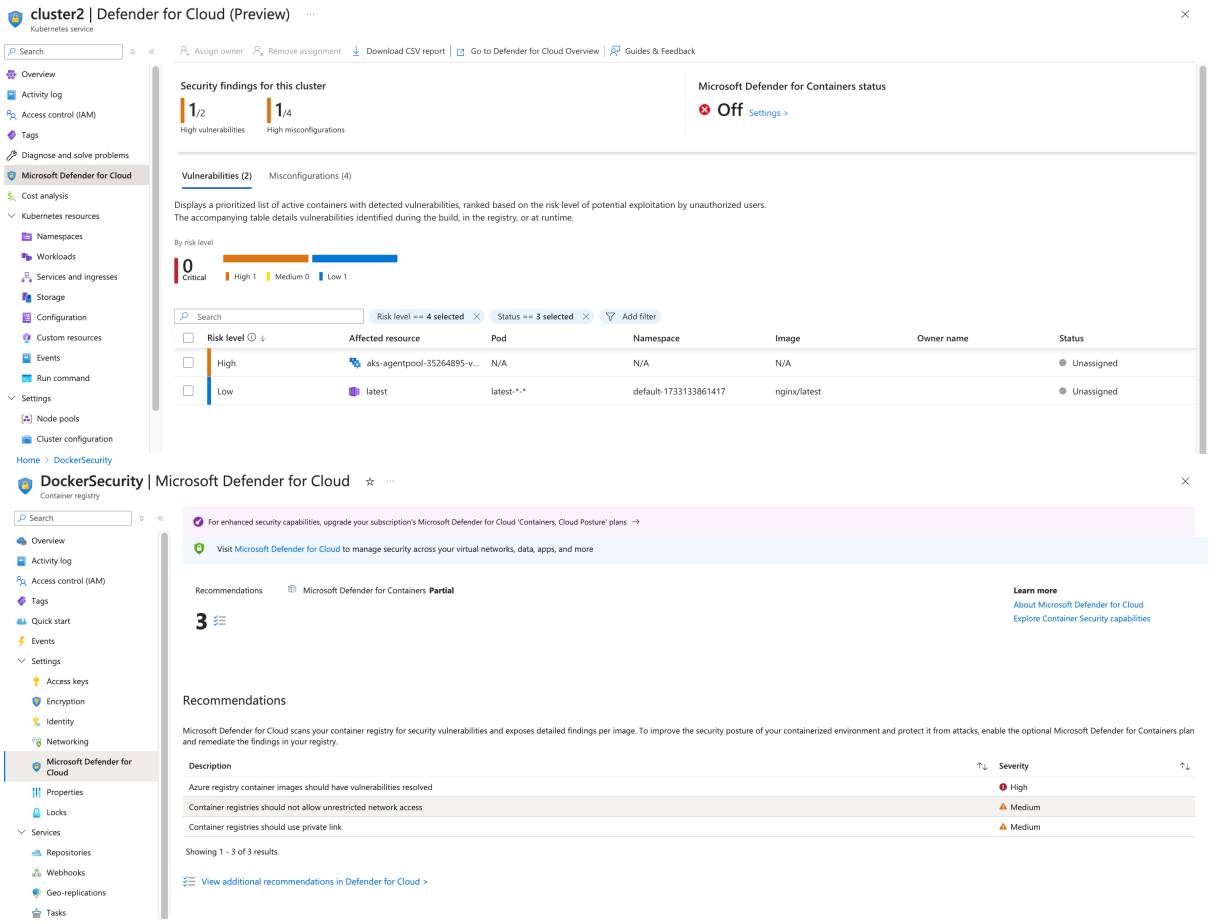

Figure 5: Experiment / Case Study 3

The pre-build phase focused on identifying and remediating vulnerabilities in Dockerfiles. By using a custom Flask-based GitHub application, vulnerabilities were detected and categorized according to OWASP Top 10 guidelines.

- **Findings:** The tool flagged issues such as the use of outdated base images, missing health checks, and hardcoded secrets. The most recurring vulnerability was the absence of health checks, which aligns with findings by (Zhu and Gehrmann; 2021), who emphasized the criticality of monitoring container health in multi-tenant environments.
- **Implications:** Early-stage vulnerability detection reduced the attack surface, minimizing risks in subsequent stages of the deployment pipeline.
- **Critique:** While the tool effectively identified vulnerabilities, the detection of repetitive issues, like health check omissions, highlighted the need for enhanced deduplication logic to avoid redundant recommendations.

Experiment 2: Post-Build Vulnerability Assessment

In the post-build phase, Docker images were scanned using Trivy and Aqua Security to identify vulnerabilities related to libraries, dependencies, and runtime configurations.

- **Findings:** Scanning revealed medium and high-severity vulnerabilities in libraries like libkrb5. These vulnerabilities could lead to lateral movement risks, as noted by (Friman; 2024), if not addressed before deployment.
- **Implications:** Scanning ensured that only secure images were pushed to the Azure Container Registry (ACR), reinforcing the importance of layered security.
- **Critique:** While the scanning tools provided comprehensive reports, their integration with the CI/CD pipeline introduced latency. Future improvements could explore parallel processing to optimize performance.

Experiment 3: Runtime Security Evaluation

The runtime security evaluation utilized Microsoft Defender for Cloud to monitor container activities in Azure Kubernetes Service (AKS) and ACR.

- **Findings:** Defender identified potential misconfigurations, such as exposed ports and unrestricted network access in ACR. Similar concerns were raised by (Loaiza Enriquez; 2021), who highlighted the importance of robust configuration management.
- **Implications:** The integration of Defender provided real-time alerts and remediation recommendations, enhancing the overall security posture of the system.
- **Critique:** Defender's effectiveness was limited by its dependency on accurate configuration inputs. Misconfigurations in Kubernetes manifests could lead to false positives, underscoring the need for automated validation of manifests.

Comparison with Existing Literature

This study builds upon prior research by introducing a multi-layered approach to Docker security in Azure environments. Unlike (Alkhatib et al.; 2024), who focused solely on Azure's native tools, this study combined native tools with third-party solutions like Trivy and Aqua Security. The findings corroborate (Wong et al.; 2023), who advocated for network isolation and runtime protection, but extend their recommendations by integrating continuous monitoring and automated remediation.

Limitations and Future Directions

While the study successfully mitigated several vulnerabilities, it has some limitations:

1. **Tool Dependency:** The reliance on specific tools like Trivy and Defender may limit applicability in non-Azure environments.

2. **Scalability:** The proposed framework requires further validation in large-scale deployments with diverse workloads.
3. **Real-Time Adaptability:** Emerging threats in containerized environments necessitate real-time adaptability, which was not fully addressed in this study.

Future research could explore:
- Integrating AI-driven threat detection for adaptive security.
- Extending the framework to hybrid and multi-cloud environments.
- Comparing the effectiveness of different container security tools across platforms like AWS and GCP.

7 Conclusion and Future Work

The research titled "Enhancing Docker Security on Azure Cloud: A Comprehensive Analysis and Mitigation Strategy" explores vulnerabilities associated with Docker containers in Azure cloud environments and proposes effective mitigation strategies. The study identifies key security gaps in Docker's multi-tenant architecture, evaluates Azure's native security tools like Defender and RBAC, and develops practical security measures. It effectively achieves its objectives by analysing vulnerabilities, proposing mitigation strategies, and recommending a framework for continuous security monitoring.
However, the study faces limitations, such as the lack of real-world deployment data to validate the proposed strategies and restricted focus on Azure-specific tools, which might limit its applicability in hybrid or non-Azure environments. Future research could explore broader cloud environments, incorporating hybrid cloud scenarios or comparing Docker security across multiple platforms like AWS and GCP. Commercially, the findings could inspire enhanced security solutions, such as automated vulnerability scanning tools tailored for Azure or consultancy services for containerized application security. Further development of frameworks for real-time threat monitoring and adaptive security measures could significantly extend the study's impact, fostering robust and scalable Docker security practices across diverse cloud ecosystems.

Acknowledgement

My sincere thanks to the research supervisor Mr. Vikas Sahni, all the faculty of Research in Computing (National College of Ireland) and to my family and friends who supported me to understand the project and its documentation in the best ways possible.

References

Alkhatib, A., Shaheen, A. and Albustanji, R. (2024). A comparative analysis of cloud computing services: Aws, azure, and gcp, *International Journal of Computing and Digital Systems* **16**(1): 1–23.

Alouffi, B., Hasnain, M., Alharbi, A., Alosaimi, W., Alyami, H. and Ayaz, M. (2021). A systematic literature review on cloud computing security: threats and mitigation strategies, *Ieee Access* (9): 57792–57807.

Chauhan, M. and Shiaeles, S. (2023). An analysis of cloud security frameworks, problems and proposed solutions, *Network* **3**(3): 422–450.

Friman, O. (2024). Agile and devsecops oriented vulnerability detection and mitigation on public cloud.

Ghazizadeh, H., Tamm, G. and Creutzburg, R. (2024). Automated tools for cloud security testing, *Electronic Imaging* (36): 1–7.

Gupta, R., Mishra, G., Katara, S., Agarwal, A., Sarkar, M., Das, R. and Kumar, S. (2016). Data storage security in cloud computing using container clustering, *IEEE 7th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)* pp. 1–7.

Lee, H., Kwon, S. and Lee, J.-H. (2023). Experimental analysis of security attacks for docker container communications, *Electronics* **12**(4): 940–940.

Loaiza Enriquez, R. (2021). Cloud security posture management/cspm) in azure.

Mallikarjunaradhya, V., Pothukuchi, A. and Kota, L. (2023). An overview of the strategic advantages of ai-powered threat intelligence in the cloud, *Journal of Science Technology* **4**(4): 1–12.

Morrison, A. (2023). Docker implementation in azure cloud, *Cloud2Data* .

Mustyala, A. and Tatineni, S. (2021). Advanced security mechanisms in kubernetes: Isolation and access control strategies, *ESP Journal of Engineering Technology Advancements (ESP JETA)* **1**(2): 57–68.

Muzumdar, P., Bhosale, A., Basyal, G. and Kurian, G. (2024). Navigating the docker ecosystem: A comprehensive taxonomy and survey, *arXiv preprint arXiv:2403.17940* .

Singh, A. and Aggarwal, A. (2022). A comparative analysis of veracode snyk and checkmarx for identifying and mitigating security vulnerabilities in microservice aws and azure platforms, *Asian Journal of Multidisciplinary Research Review* **3**(2): 232–244.

Wong, A., Chekole, E., Ochoa, M. and Zhou, J. (2023). On the security of containers: Threat modeling, attack analysis, and mitigation strategies, *Computers Security* (128): 103140.

Zhu, H. and Gehrmann, C. (2021). Lic-sec: an enhanced apparmor docker security profile generator, *Journal of Information Security and Applications* (61): 102924.