**DOCKER PROJECT**

# Documentation

**TITLE:** **Quiz App Vulnerability Scanning**

**MCA CC & DEVOPS**

**Submitted by:**

Rohit Jaiswal - 23MCC20024

Shivam Ahuja - 23MCC20006

**Submitted to:**

Ms. Sofia (Project Supervisor)

# Acknowledgement

The successful completion of this project, “**Quiz App Vulnerability Scanning**” required immense support and guidance from several individuals. We are fortunate to have received continuous assistance throughout the project.

We would like to express our sincere gratitude and appreciation to our project mentor, **Ms. Sofia**, whose guidance, invaluable feedback, and constant encouragement were essential to the development of this project. Her time and effort spent reviewing our work and providing suggestions were instrumental to our success.

We would also like to extend our thanks to the UIC Department for granting us the opportunity to work on this project, which has significantly enhanced our skills and broadened our understanding of security in software development.

Lastly, we express our heartfelt appreciation to all those who contributed directly or indirectly towards the successful completion of this project.

# Abstract

The "Web App Vulnerabilities Scanning" project enhances the security of a React-based quiz application, **quizzapp**, by implementing a vulnerability scanning process using Docker and Trivy. Leveraging Docker for containerization ensures a consistent environment for the application and scanning tools. Trivy, an open-source vulnerability scanner, analyzes the application’s container image to identify known vulnerabilities and outdated libraries. A detailed vulnerability report is generated, highlighting security flaws and offering remediation strategies. This project underscores the importance of integrating vulnerability scanning into the development lifecycle and promoting security best practices to strengthen the overall security posture of the quizzapp application.

**Project Overview**

This project aimed to implement a robust vulnerability scanning process for a React-based quiz web application, **quizzapp**, utilizing Docker and Trivy. By containerizing the application and leveraging Trivy's capabilities, we sought to streamline deployment and proactively identify potential security risks. The ultimate goal was to ensure a secure and reliable application that meets the high standards of modern web security.

**DevOps Approach and Tools**

**1. Docker**

Docker was employed as the primary tool for containerization, allowing us to create a consistent and reproducible environment for both the application and the scanning tools. This approach simplifies the development workflow, enabling seamless transitions between different stages of deployment and testing.

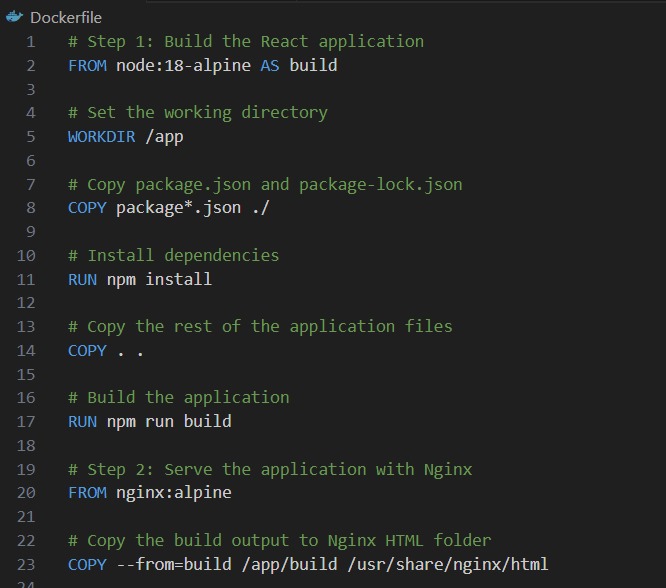
**2. Trivy**

Trivy, an open-source vulnerability scanner developed by Aqua Security, was utilized to analyze the application's container image for known vulnerabilities. With its ability to scan both OS packages and application dependencies, Trivy provides comprehensive coverage for identifying security flaws, outdated libraries, and other potential risks.

**Project Setup and Configuration**

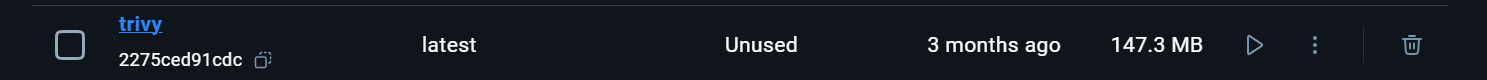
**1. Dockerfile**

A multi-stage Dockerfile was created to efficiently build the React application and serve it using Nginx. The two-stage process not only reduces the final image size but also isolates the build environment from the runtime environment, enhancing security and performance.



**2. Trivy Integration**

Trivy was configured to scan the application image and generate a detailed vulnerability report. This integration is vital for maintaining an up-to-date awareness of the security status of the application, allowing for timely remediation of identified issues.



**Vulnerability Scanning Process**

**1. Container Image Creation**

The Dockerfile facilitated the creation of the application’s container image, ensuring that all dependencies and configurations were encapsulated within a single, portable unit. This process involved:

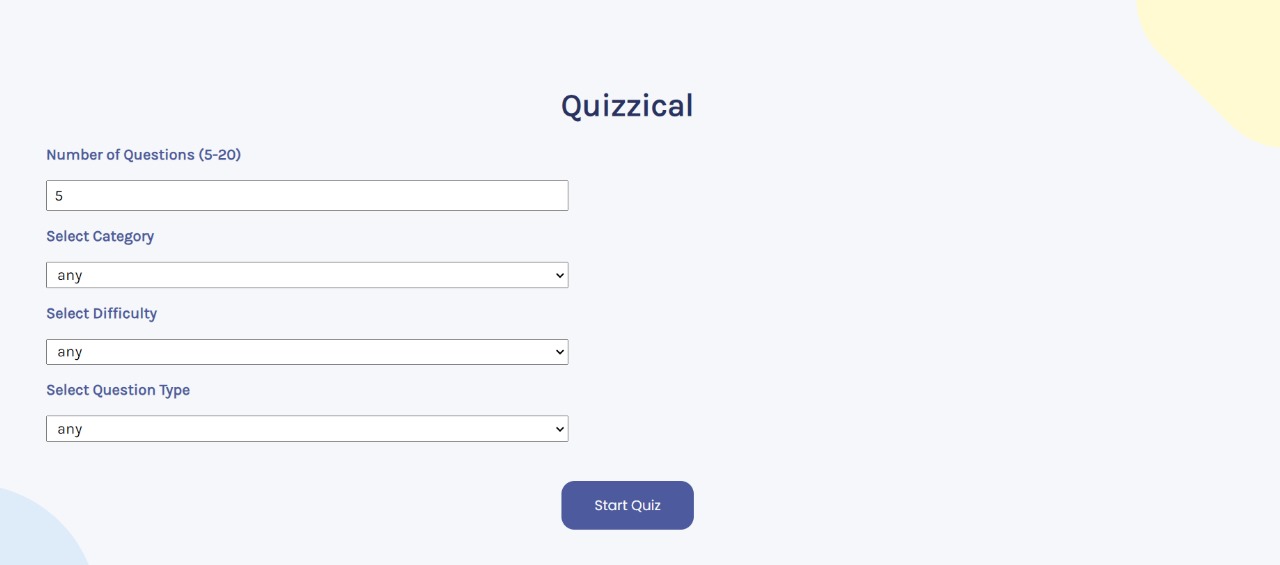
* **Building the React Application**: Using Node.js, the application was built for production.
* **Serving the Application with Nginx**: The built application was then served via an Nginx server, providing a lightweight and efficient means of hosting.

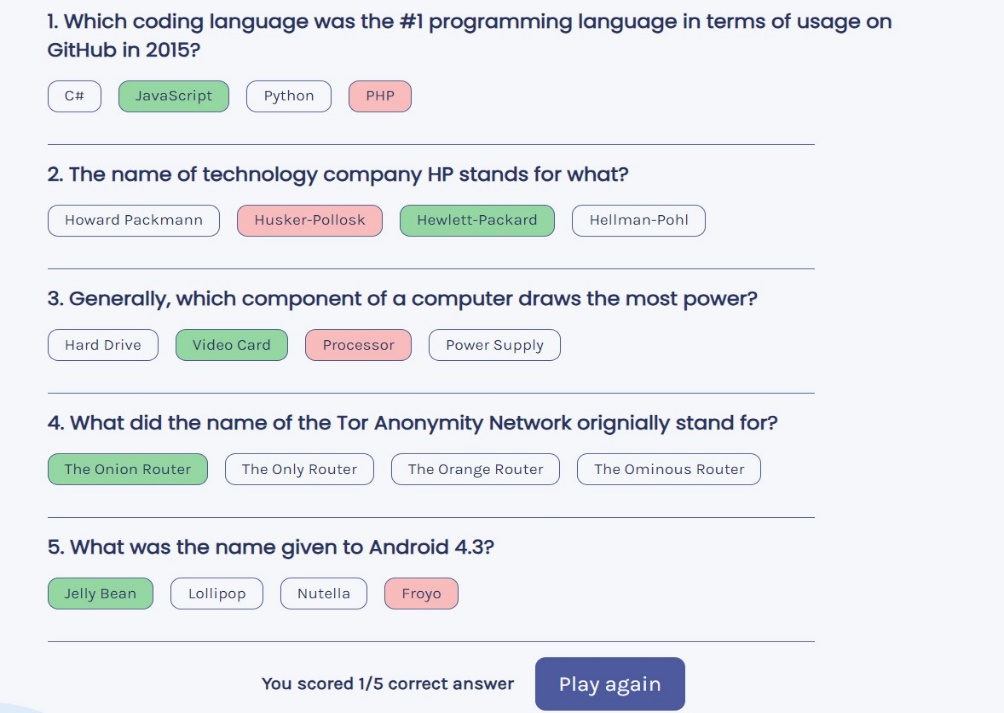
**2. Trivy Scan**

Trivy was executed against the container image to identify vulnerabilities. This process involved running the following command in the Docker environment:

**Command:** “docker run --rm -v /var/run/docker.sock:/var/run/docker.sock aquasec/trivy image quizzapp > vulnerability-report.txt”

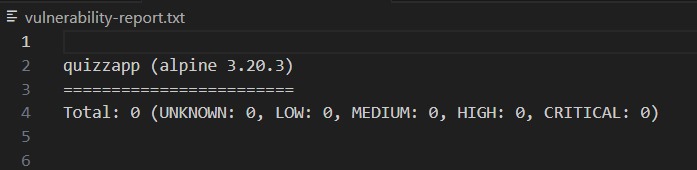
**Project Interface**





**Report Generation**

A comprehensive vulnerability report was generated, detailing discovered vulnerabilities, their severity, and potential remediation steps. The report is crucial for developers and security teams to prioritize and address vulnerabilities effectively.

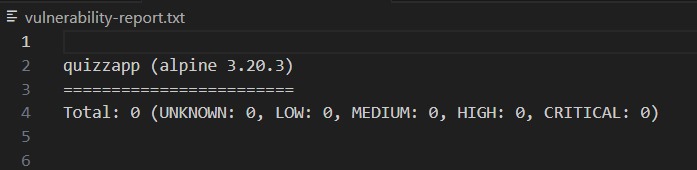


**Key Findings and Recommendations**

**1. Vulnerability Identification**

Trivy successfully identified several vulnerabilities within the application's dependencies, including outdated libraries and known security flaws. Some of the notable findings included:

* Vulnerabilities in libraries such as React and its associated packages, which could lead to exploitation if left unaddressed.
* Outdated versions of dependencies that were known to have critical security patches available.



**2. Prioritization and Remediation**

Based on the report, vulnerabilities were prioritized according to their severity and potential impact. Recommended remediation strategies included:

* Updating to the latest versions of affected libraries to mitigate known vulnerabilities.
* Applying security patches provided by library maintainers promptly.

**3. Security Best Practices**

In addition to remediation, the implementation of additional security measures is crucial to mitigate risks. Recommendations include:

* **Input Validation**: Ensuring that user inputs are validated to prevent injection attacks.
* **Output Encoding**: Encoding outputs to avoid cross-site scripting (XSS) vulnerabilities.
* **Secure Coding Practices**: Following best practices in code development to minimize security flaws.

**Future Enhancements**

**1. Continuous Scanning**

Integrating vulnerability scanning into a continuous integration/continuous delivery (CI/CD) pipeline is essential for maintaining a secure application throughout its lifecycle. Automated scans can catch vulnerabilities early in the development process, reducing the risk of production deployment.

**2. Security Awareness**

Promoting security awareness among development teams is vital in preventing vulnerabilities from being introduced in the first place. Regular training and workshops on secure coding practices can enhance the overall security posture of the organization.

**3. Advanced Scanning Techniques**

Exploring more advanced scanning techniques, such as dynamic application security testing (DAST) and interactive application security testing (IAST), can provide deeper insights into potential vulnerabilities. These methods can complement static analysis and offer a more holistic view of application security.

**Conclusion**

This project successfully demonstrated the effectiveness of using Docker and Trivy for vulnerability scanning in a web application. By identifying and addressing potential security risks proactively, we have enhanced the overall security posture of the **quizzapp** application. Continuous monitoring and improvement of security practices are essential to ensure ongoing protection against emerging threats. As the landscape of security threats evolves, adopting a proactive and integrated approach to security will be vital in safeguarding applications and maintaining user trust.