Longitudinal risk-based security assessment of Docker software container images

MSc Research Project
MSc in Cloud Computing

Alric Mendonsa
Student ID: 21227462

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Alric Mendonsa
Student ID:	21227462
Programme:	MSc in Cloud Computing
Year:	2024-2025
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	12/12/2024
Project Title:	Longitudinal risk-based security assessment of Docker software container images
Word Count:	7079
Page Count:	21

Signature:	Alric Mendonsa
Date:	25th January 2025

Longitudinal risk-based security assessment of Docker software container images

Alric Mendonsa
21227462

Abstract

In today's cloud-pushed software program improvement ecosystem, Docker containers have emerged as a cornerstone for utility deployment because of their portability and scalability. However, the increase of vulnerabilities inside Docker images poses vast safety dangers, mainly whilst left unmonitored over time. The study investigates the longitudinal dangers related to Docker images hosted on DockerHub with the aid of using a comparative evaluation of open-source vulnerability assessment scanning tools "Trivy" and "Grype" towards AWS's Elastic Container Registry (ECR) scanning service. A dataset of Docker images was been put to test to periodic vulnerability checks, with the aim of getting the outcome analyzed for detection efficiency, coverage, and timeliness. Using AWS offerings which include EC2, ECR, S3, CodeBuild, and CodePipeline, an automatic CI/CD pipeline turned into applied to test box photographs and keep outcomes for visualization. The findings show actionable insights into the evolving safety of containerized applications, which presents a strong method to improve the aspect of vulnerability control practices. This observe underscores the significance of integrating multi-device checks to enhance safety controls in containerized environments.

1 Introduction

1.1 Background

The concept of cloud computing, microservice, and CI/CD pipeline is relatively new and has revolutionized the whole process of writing, deploying, and maintaining software. One of the primary drivers behind this change is the use of containerization technology, of which Docker is the most famous. Docker currently offers a thin and virtualization technology which then simplifies deployment of software application with their dependencies to different operating systems environments. This simplicity has led Docker today as one of the most crucial tools in DevOps as well as cloud-native applications development.

Nonetheless, similar to virtually all emerging technologies that have gained widespread popularity within a short period, Docker comes with newly emerged security concerns. Container images downloaded from registries such as DockerHub may have the associated risks such as vulnerabilities or misconfiguration that are likely to affect the organizations which use the containers. For example, Docker containers actually run on top of host OS kernel, which means that host's kernel is shared and under all conditions this makes the attack surface larger if good security measures are not provided. As a result, container security especially in Docker containers has become a major issue of concern in organizations that rely on containers for production applications.

The conventional model of security censed problems to be solved at a particular moment and does not take account of system changes. Unfortunately for Docker containers, this is a static method because the images are constantly being rebuilt with new versions of the software and new patches. These changes can involve creation of new vulnerability or rectification of existing one which requires the concept of threat management to be more proactive.

1.2 Problem statement

Docker images themselves may be updated multiple times during the lifespan of the image and the security posture of an image can change over its usage. Currently available vulnerability assessment tools mainly provide point-in-time assessments for software containers and fail to consider the dynamic characteristics of the technology. This raises questions on the prospects of Docker containers' security over time especially in production where new threats accumulate over time. Also, there might be old data in public registries, most of which are insecure, and which people end up pulling into their local environments.

Consequently, there is a high demand for a security evaluation that can be carried out on a frequent basis to detect Docker images that may contain security threats. This research seeks to fill this lacuna by carrying out a risk-based security analysis of Docker software container images as they occur, with the view of proposing strategies for security risk management.

1.2.1 Research question

What specific threats do Docker software container images pose over time and how can a comparison-based approach using existing Docker container vulnerability scanners be used to determine the overall efficiency to gain new insights on vulnerabilities to enhance the security control of these containers ?

1.3 Research aim and objectives

The primary objective of this research is to establish a long-term Docker container image risk-based security assessment process. This framework will track images over time and track vulnerabilities as well as risks on an ongoing basis. The specific objectives of the research are:

1. To carry out a first level vulnerability scan of docker images.

2. To provide a serial research of security threats linked with Docker containers with an aim to determine recurrent trends and patters.

3. To incorporate the risk analysis to prioritize the different vulnerabilities in order to prioritize the risks that are most dangerous.

4. In-order for the mitigation strategies to be proposed and evaluated for Docker containers in the production environments it was necessary.

1.4 Significance of the study

The value of this statement is in the long-term perspective towards security of containers that was employed in this research. This research does not advocate a once approach to the security of Docker images but aims at presenting a framework that may be used continually to ensure that an organization's environment is secure while using Docker images. The result of the current investigation will be useful for cloud-native application developers, Cybersecurity personnel, and DevOps engineers, who implement Docker containers in production settings. In addition, knowledge derived from the following research will guide the best practices in containerized workloads security in AWS and other cloud service providers.

2 Related Work

2.1 Overview of Containerization and Docker

Docker is a form of operating system virtualization called containerization but even it has antecedents. Advanced as early as chroot in Unix or Linux Containers also known as LXC introduced some simple methods for the isolation of process and its dependencies. These stand-point technologies set the stage for enhanced and advanced containerization innovations proceeding forward. Docker which was launched in the year 2013 took the previous ideas and foundations and made it easier for the users to create, deploy as well as to manage containers easily (Hradec et al., 2022).

Docker containers embed an application with all its dependencies allowing some level of portability across development, test, or production environment. This encapsulation reduces the famous issue of the 'it works on my machine' since applications will work the same way irrespective of the environment they are run in (Efe et al., 2020).

Docker has a layered structure; however, the kernel is the most important part of Docker's architecture which is known as the Docker Engine. Moreover, Docker Hub also satisfies as another repository through which developers can host their images in cloud. Docker rightfully possesses the main role in what has been established as the modern software containerization based on this architecture and has spurred an extensive range of tools and practices that help implement DevOps and continuous delivery (Khang et al., 2023).

2.2 Security challenges in Docker

While Docker is beneficial in the deployment of applications, it raises major questions on the security of containerization that organizations will have to respond to in order to keep containerization secure.

One is that the containers can be misconfigured and this results in creation of loopholes for the containerized applications. Many users fail to properly configure security during the creation of the container or its orchestration; therefore, loses become exposed with entry points to the attackers. Moreover, it is possible to raise concerns about publicly available Docker images because in many cases these very images may contain old software with known holes or other malicious code.

Furthermore, containers longevity is quite short, making basic security practices extremely challenging. Because containers are often fleeting and created and destroyed constantly, it becomes challenging to maintain an inventory of security policies and compliance. This dynamic environment also causes weaknesses in the visibility and monitorability when it is difficult for an organization to identify and respond to the threat in real time.

Another concern is concerned with how container containers' communication is going to be done. Lacking proper implementations of network segmentation and user access controls, an adversary is capable of taking advantage of a basic container to obtain full access into another. Thus, it is important that organizations develop a sound security plan that would entry the program to constantly monitor, scan and input recommended measures toward mitigating such risks.

2.3 Vulnerabilities in Docker images

This paper provides the assumption that vulnerability represents one of the main threats in Docker environments with the Docker images themselves containing vulnerabilities. A work that examined the sensitivity of the public Docker images and was performed by Efe et al. (2020) observed that a significant proportion of the Docker images in Docker Hub exposed significant open vulnerabilities correlated with administrative misconfiguration and use of outdated libraries. These vulnerabilities render the system prone to attack if the vulnerabilities are not quickly detected and neutralized and result in compromise of the host system and its resources (Mills et al., 2023).

In addition, ordinary third-party pictures used in application are often derived from the internet and do not receive particularly stringent security checks. These images are deployed from public repositories and vendors where developers do not undertake strict scrutiny processes hence raising the probability of coding their applications with vulnerable components. This practice clearly shows why it is essential to have efficient vulnerability scanning to evaluate the Docker images integrity (Royer et al., 2022).

2.4 Shared kernel vulnerabilities

The other major concern that's related to Docker are the architecture of Docker based on the shared kernel model. Unlike other virtual machines that work with own kernel instance, Docker containers share kernel with the host operating system. This shared architecture increases the threat of Kernel-level attacks, where an attacker has the advantage of rooting around inside the host operating systems' kernel to get a hole into the container (Schweinar et al., 2024).

Another study by Royer et al. (2022) only proves that docker is vulnerable to privilege escalation attacks, while kernel security patches for docker should be updated and proper access controls should be in force. Because of the integration between containers and the host kernel, incorporating of solid isolation measures forms the core that needs to be protected in the entire system architecture.

2.5 Registry security issues

Docker Helm charts are used often in distributing Docker images with the help of public container registries like Docker Hub. But few of these registries have strong security measures to minimize risks on the organizations that have to rely on them. There is a notable problem of image manipulation where people attempt to make changes to an image that will introduce a backdoor or malware (Thorn et al., 2022).

Additionally, the variety of images residing in public registries is not always checked for vulnerabilities before being published and made available for download. The absence of adequate security measures beforehand leads to the following supply chain risk for organizations that rely on these images for their containerized applications (Mens, 2024). Research has shown that a preeminent registry level scanning and verification mechanism is needed to improve overall security of container environments (Kim et al., 2024).

2.6 Existing vulnerability assessment tools

Due to characteristic security issues of Docker, some tools had been designed to check vulnerabilities of Docker images and containers. Different of these tools work in different ways and come with different functionalities to enable organizations to choose the best tool to use when it comes to putting into practice security measures.

2.6.1 AWS Inspector

AWS Inspector is one of the most used security assessment services that work well with Docker containers running on AWS. This service seeks to go through a containers search for common vulnerabilities and compile a report of what it discovers. As an AWS service, AWS Inspector excels at pinpointing static risks but it does not have the functionality for real-time risk monitoring (Schouten et al., 2023).
Consequently, the adoption of AWS Inspector should be backed up by other tools, procedures to continually monitor the security conditions of containerized applications.

2.6.2 Clair and Anchore

Clair and Anchore are very specific tools developed for processing vulnerability scans of Docker images. Clair scans images for threats that are inherent in the software that may be included within the images that developers use. However, Anchore introduces another level of checks as it provides policy-based assessments, to check that images used meet the security policies of the firm (Khang et al., 2023).
However, both Clair and Anchore have the ability to be built into the CI/CD pipeline to ensure that there are tests for secure images during the build. This integration enhances the timely identification of vulnerabilities so that developers can solve security problems without going to the production contexts (Kim et al., 2024).

2.6.3 Snyk

Another beneficial tool that Snyk is that it offers a constant check on Docker containers. It works in parallel with container repositories and scans images for newly found vulnerabilities non-stop, and notifies users every time a new vulnerability has been found (Mills et al., 2023). The ability to monitor an image over time also fits under the maturity concept of longitudinal security assessment that makes the Snyk solution fundamental to evaluate the security posture in the containerized environment (Mens, 2024).
Together with Snyk, other tools for vulnerability assessment can be used to improve the security situation in organizations and prevent new threats endangering containerized applications.

2.7 Longitudinal security analysis

Longitudinal security analysis involves the regularity and periodical examination of systems as opposed to the conventional once over glance at the system. This is even more valuable in environments of change such as Docker where images and containers are often evolving (Sion et al., 2023).

2.7.1 Dynamic-risk profiles

It is important to point out that another strength of Longitudinal analysis is the capacity for assessment of variation in risk of the particular system in time. As software packages or the patch is updated on the Docker images, the risk level constantly changes, due to the introduction of new and/or avoidance of some. In that way, longitudinal assessments can contribute to defining patterns of vulnerability occurrences so that the forecasts and the preemptive actions could fit them adequately (Thorn et al., 2022).
For instance, a longitudinal study may show that, or kinds of vulnerabilities occur at given times or after some updates are made. This paper seeks to identify paradigm shifts in organizations that, when understood, will enable organizations to employ security measures effectively to meet their needs.

2.7.2 Longitudinal security of containers

There have been studies on longitudinal security analysis in the traditional software systems; however, relatively few studies have focused on the Docker containers case.' Jenkins et al. in their work indicate that continuous monitoring can go a long way in improving the security of the containerized systems. However, many of them are conducted at a specific period and, therefore, leave gaps that require the development of more comprehensive frameworks for long- term analysis (Royer et al., 2022).
Therefore, more studies are needed to identify approaches that enable continuous assessment of security in containerized environments to allow the organization to change from one implementation to another as threats and risks evolve over time.

2.8 Risk-based assessment of Docker containers

Risk management approach to security evaluation categorizes the risks by the likelihood of their being exploited. This means that through this method, resources can be used appropriately since the main threats of an organization are addressed in different stages first (Khang et al., 2023).
Let's take a look at the 5 parts that make up the CVSS
This is a generally appreciated structure for characterizing the significance of security openings called the Common Vulnerability Scoring System (CVSS). When calculating a given vulnerability's CVSS, organizations are given scores that depend on features such as exploitability and impact, allowing them to determine which vulnerability is best to handle first (Thorn et al., 2022). In the case of Docker containers, including CVSS in risk-based evaluations guarantee timely mitigation of highly vulnerable situations, reducing the possibility of exploitation.
CVSS scores can be used as the basis for the formation of proper security policies and procedures that organizations may need to manage container security threats in connection with their business goals (Kim et al., 2024).

2.8.1 Risk-based approaches applied to containers

Risk based solutions applied to the Docker containers have had a variable level of success. Kim et al. (2024) established that it is possible to elevate the efficiency of security evaluations by integrating CVSS metrics with other characteristics peculiar to a container, including the frequency of their utilization or the level of data security in applications that employ the container.
Including contextual data into risk evaluation allows making more efficient decisions concerning which risks should be prioritized and where organizational security resources should be invested. It has been found that this particular approach makes organisations benefit from general container security and helps them manage their risks more efficiently.

2.9 Encryption of services in cloud platforms - AWS ECS/EKS

With orchestration platform service like Amazon ECS and EKS, there are other kinds of security challenges to consider. These platforms deal with a set of containers and control their deployment, scaling, and operations; if not controlled, this might increase the number of vulnerabilities (Hradec et al., 2022).

2.9.1 Configuration management and policy enforcement

Secure configuration management is critical to container orchestration environments security. AWS Configurations and AWS IAM enable the desire security practices and policies to be implemented for container clusters. Through constant checks of the configurations, it is possible to detect situations where certain configurations are outside the standardized policies and thus avoid compromises from misconfigurations (Sion et al., 2023).
Furthermore, to deal with the access control problem, RBAC is also adopted in orchestration platforms to minimize the users and services' permissions so that it is less vulnerable to threats on containerized applications (Khang et al., 2023).

2.9.2 Protecting networks with container orchestration

Another general component of container orchestration platforms security is the network security. Security groups, network ACLs, and service meshes are effective ways that can allow for secure connections between containers and control view access to resources they require (Mills et al., 2023).
Furthermore, network segmentation within the orchestration platforms can assist in differentiating the containers according to sensitivity and their roles would make the environment to be more secure according to Mens (2024).
The literature presents several studies and models that demonstrate the interaction between the key benefits of using Docker containers and security issues of this approach. Though there is increasing research on assessing the security of containers, current and future trends in containers call for further study on the area. Given the rising openness of organizational applications to containerization, adequate incorporation of continuous monitoring, risk assessment, and strong security principles will be critical in protecting a given organizational containerized setting.

2.9.3 Longitudinal security of containers

There have been limited studies on applying longitudinal security assessment techniques to Docker containers. Most research on Docker security focuses on snapshot-based vulnerability assessment tools or the analysis of static risks associated with containerization. Jenkins et al. (2023) explored the application of longitudinal assessment techniques to software systems in general and highlighted their potential for predicting risk trends. While their work primarily dealt with traditional systems, the insights provided are applicable to containerized environments. For instance, Jenkins et al. demonstrated how longitudinal analysis could track changes in security postures and link them to specific updates or patches.
In the context of Docker, researchers like Schweinar et al. (2024) have stressed the importance of adopting a time-sensitive approach to container security. Their study examined how frequent vulnerability scans could help organizations detect security regressions caused by system updates. Additionally, their work pointed out that continuous monitoring of container images over time could provide actionable insights for preemptive threat mitigation.
Mens et al. (2024) also emphasized the utility of longitudinal approaches in tracking vulnerabilities that evolve or persist across multiple updates. Their findings suggest that implementing such techniques can help organizations understand and address recurring vulnerabilities, ultimately improving their overall security posture. Despite these promising findings, there remains a gap in practical implementation, particularly in automating longitudinal assessments for Docker container environments.

3 Methodology

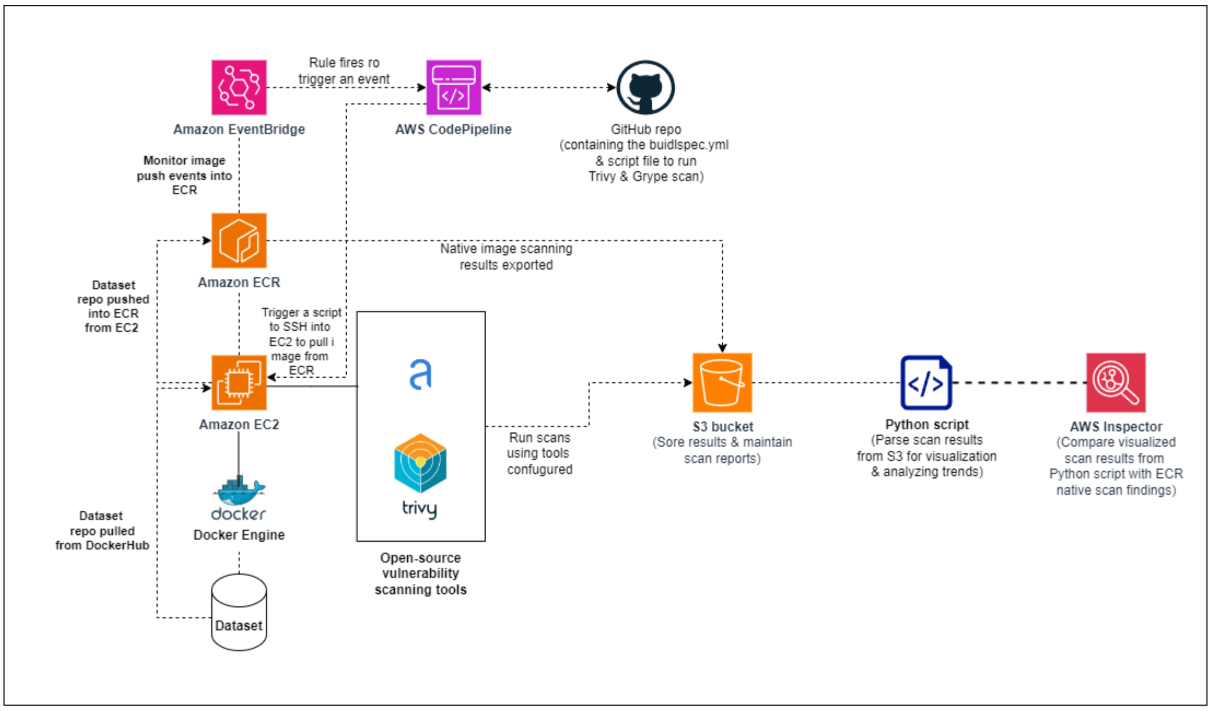
This section outlines the research methodology employed for conducting a longitudinal risk-based security assessment of Docker container images. By leveraging a structured approach that integrates both quantitative and qualitative analyses, this study aims to systematically evaluate the security posture of Docker images over time. The methodology comprises four key stages: tool selection, data collection, vulnerability analysis, and risk assessment. Each stage is detailed below to elucidate the systematic process and ensure replicability.

4 Design Specification

The research adopts a mixed-method approach to effectively address the research objectives. This design allows for the integration of numerical data analysis with in-depth interpretative insights, ensuring a comprehensive understanding of the security landscape. The following subsections detail the major components of the research design.

4.1 Architectural diagram

Below is the architectural diagram of the solution implemented in AWS :-



Amazon EventBridge
Rule fires ro trigger an event
AWS CodePipeline
GitHub repo (containing the buidlspec.yml & script file to run Trivy & Grype scan)
Monitor image push events into ECR
Amazon ECR
Native image scanning results exported
Dataset repo pushed into ECR from EC2
Trigger a script to SSH into EC2 to pull i mage from ECR
Amazon EC2
trivy
Run scans using tools confugured
S3 bucket (Sore results & maintain scan reports)
Python script (Parse scan results from S3 for visualization & analyzing trends)
AWS Inspector (Compare visualized scan results from Python script with ECR native scan findings)
Dataset repo pulled from DockerHub
docker
Docker Engine
Open-source vulnerability scanning tools
Dataset

High-level architectural diagram

Figure 1: Architectural diagram

4.2 Tool selection

The selection of appropriate tools is critical for achieving accurate and reliable results in vulnerability scanning and analysis. For this study, a combination of widely recognized and robust tools is employed:

- AWS Inspector : Known for its seamless integration with cloud-based environments, AWS Inspector provides automated security assessments of container images. Its ability to detect common vulnerabilities and exposures (CVEs) makes it a vital tool for this research.

- Synk : A developer-centric security tool, Snyk specializes in identifying vulnerabilities in open-source dependencies and Docker images. Its frequent updates ensure the inclusion of newly identified threats, enhancing the precision of the analysis.

- Clair : Clair is an open-source tool specifically designed for static analysis of vulnerabilities in container images. Its layered approach allows for the identification of issues at various stages of the image creation process.

- Anchore : Offering both vulnerability scanning and policy compliance checks, Anchore is integral for evaluating security and adherence to best practices in Docker image configurations.

By utilizing a combination of these tools, the study ensures a multidimensional evaluation, reducing the likelihood of oversight due to tool-specific limitations. Additionally, cross-validation of results from multiple tools enhances the reliability of the findings.

4.3 Data collection

The data collection phase involves curating a comprehensive dataset of Docker images from public registries, with a primary focus on Docker Hub. This platform is chosen due to its extensive repository of images and its status as the most widely used container registry. The data collection process follows these steps:

- Identificatiuon of Docker images : A diverse set of Docker images is selected, representing various use cases such as web applications, data processing, and machine learning. This diversity ensures the inclusion of images with different levels of complexity and security requirements.
- Popularity-based sampling : Images are categorized based on their download counts and community ratings to represent a mix of highly popular and less frequently used images. This stratified sampling approach allows for the analysis of security trends across different popularity levels.
- Metadata documentation : Each Docker image's metadata, including its creation date, last update, base image, and associated dependencies, is recorded. This information is crucial for contextualizing the vulnerability analysis.
- Version-control : To facilitate longitudinal analysis, multiple versions of the same Docker image are tracked over time. This step involves documenting changes in the image's configuration, libraries, and dependencies across different versions.

The curated dataset provides a solid foundation for conducting a detailed security assessment, enabling the identification of temporal trends and patterns.

A structured approach is employed to ensure the systematic gathering of relevant data, allowing for meaningful insights into the evolving security landscape of containerized environments. The process involves image selection, metadata compilation, and a planned scanning schedule, each outlined below.

- Image selection To ensure a representative dataset, Docker images are selected based on a variety of criteria designed to capture a broad spectrum of use cases and configurations. The selection process involves the following things:

1. Popularity metrics : Images with varying levels of popularity are included by analyzing their download counts and user ratings on Docker Hub. Popular images are prioritized because they are more likely to be deployed widely and represent potential high-value targets for attackers.
2. Official and Community images : Both official and community-contributed images are included to compare their security postures. Official images, typically maintained by software vendors or trusted organizations, are expected to exhibit higher security standards compared to community images, which may have more variability in maintenance practices.
3. Use-case diversity : Images are categorized into key use cases such as web servers (e.g., Nginx, Apache), databases (e.g., MySQL, PostgreSQL), and development tools (e.g., Node.js, Python). This ensures the inclusion of a wide range of functionalities, each with unique security implications.

By balancing these factors, the dataset aims to reflect the diversity of Docker images in real-world applications, providing a solid foundation for longitudinal security analysis.

- Metadata compilation Metadata collection is a critical step in contextualizing the security analysis. For each Docker image, detailed metadata is compiled, including:

1. Image identification : Unique identifiers such as the image ID and version are documented to distinguish between different images and track changes over time.
2. Dependemcy information : The dependencies associated with each image, including base images and libraries, are recorded. This information is essential for identifying vulnerabilities originating from insecure or outdated dependencies.
3. Tagging and labels : Tags and labels associated with the images are noted to categorize them based on attributes like environment (e.g., production, testing) or architecture (e.g., x86, ARM).
4. Update history : Information about the last update of each image is gathered to evaluate how actively the image is maintained. Actively updated images are generally less prone to vulnerabilities compared to neglected ones.

The collected metadata serves multiple purposes, including enabling detailed tracking of security changes, identifying dependency-related risks, and providing insights into maintenance practices.

- Scanning schedule A carefully designed scanning schedule is implemented to observe the evolution of vulnerabilities and assess changes in security posture over time. Key elements of the scanning process include the following things :

1. Frequency : Images are scanned bi-weekly for a duration of three months. This interval strikes a balance between capturing sufficient temporal data and avoiding excessive redundancy in scan results.
2. Tool utilization : The selected vulnerability scanning tools—AWS Inspector, Snyk, Clair, and Anchore—are employed to perform scans during each interval. Using multiple tools enhances the reliability of the results by cross-validating findings.
3. Periodic comparisons : Results from successive scans are compared to identify trends such as the emergence of new vulnerabilities, the resolution of previously identified issues, or changes in the number and severity of vulnerabilities.
4. Version monitoring : Updated versions of the same Docker images are scanned to assess the impact of updates on their security posture. This helps in determining whether updates mitigate vulnerabilities or introduce new risks.

This longitudinal scanning approach provides a dynamic view of the security landscape, enabling the identification of patterns and trends that might not be apparent from a single-point-in-time analysis.

- Ethical and practical considerations To ensure the integrity and ethical compliance of the data collection process, the following measures are adopted:

1. Permission and accessibility : Only publicly available Docker images are included in the dataset. This avoids any legal or ethical concerns associated with accessing proprietary or restricted resources.
2. Responsible reporting : In cases where critical vulnerabilities are identified, they will be responsibly disclosed to the image maintainers or relevant stakeholders. This aligns with ethical research practices and contributes to improving overall security.
3. Data integrity : Rigorous documentation practices are followed to ensure the accuracy and reliability of the collected data. All scans and metadata are securely stored to facilitate reproducibility and validation.

The structured data collection process outlined above is integral to the success of this research. By carefully selecting images, compiling detailed metadata, and adhering to a consistent scanning schedule, the methodology ensures a comprehensive and reliable assessment of Docker image security. This systematic approach not only facilitates the identification of temporal security trends but also provides actionable insights for improving the security practices of Docker image maintainers and users.

4.3.1 Longitudinal analysis

A key aspect of this research is the longitudinal assessment of Docker images to observe changes in their security posture over time. The process involves periodic scanning of selected Docker images using the tools identified earlier.

- Defined observation period : A specific time frame is established for the longitudinal study, during which scans are conducted at regular intervals (e.g., monthly). This approach ensures consistent data collection and allows for the tracking of evolving vulnerabilities.
- Temporal data analysis : The results of each scan are compared to detect changes in the number, type, and severity of vulnerabilities. Patterns such as the emergence of new vulnerabilities or the resolution of existing ones are documented.
- Dynamic dependency tracking : As many vulnerabilities originate from outdated or insecure dependencies, changes in the dependency tree of each image are analyzed over time. This step highlights the impact of updates or lack thereof on the overall security of the image.

This longitudinal approach provides valuable insights into the lifecycle of vulnerabilities and their persistence, resolution, or exacerbation over time.

4.3.2 Risk-based assessment

Risk assessment is the culminating stage of the methodology, where vulnerabilities identified during the scanning process are analyzed and prioritized based on their potential impact. This step guides the formulation of mitigation strategies.

- Severity analysis : Each vulnerability is classified according to its severity using standardized scoring systems such as the Common Vulnerability Scoring System (CVSS). High-severity vulnerabilities that pose significant risks are prioritized for immediate action.

- Exploitability evaluation : The likelihood of a vulnerability being exploited in real-world scenarios is assessed based on factors such as availability of exploit scripts and the complexity of exploitation. This evaluation helps in distinguishing between theoretical and practical threats.
- Impact assessment : The potential impact of each vulnerability on the containerized application is analyzed. This includes considerations such as data breaches, service disruptions, and financial losses.
- Risk prioritzation : Vulnerabilities are ranked based on their severity, exploitability, and impact. This prioritization aids in the allocation of resources for remediation efforts, ensuring that critical issues are addressed first.

The risk-based assessment also involves identifying trends in vulnerability emergence and resolution, providing actionable insights for improving Docker image security.

4.4 Vulnerability scanning and analysis

Vulnerability scanning and analysis form the core of this research, enabling a comprehensive evaluation of Docker image security over time. This stage involves systematically scanning Docker images using selected tools and analyzing the results to extract key metrics. These metrics will shed light on the nature, severity, and evolution of vulnerabilities, providing a foundation for risk assessment and mitigation strategies.

4.4.1 Scan results categorization

The results of each vulnerability scan are categorized using the Common Vulnerability Scoring System (CVSS), which provides a standardized framework for evaluating and quantifying security vulnerabilities. This categorization aids in understanding the potential impact of vulnerabilities and prioritizing them for remediation.
Detected vulnerabilities are grouped into three severity categories based on their CVSS scores:

1. Low severity : CVSS scores ranging from 0.1 to 3.9. These vulnerabilities pose minimal risk and often require specific conditions to exploit.
2. Medium severity : CVSS scores ranging from 4.0 to 6.9. These vulnerabilities represent moderate risks and may have broader exploitability.
3. High severity : CVSS scores of 7.0 and above. These vulnerabilities are critical, potentially allowing attackers to compromise systems with significant impact.

This classification helps in focusing efforts on addressing vulnerabilities that pose the greatest threat.

4.4.2 Key metrics extracted

From the scan results, the following metrics will be extracted and analyzed to provide a detailed understanding of the security posture of Docker images:

1. Number of vulnerabilities
   The total count of vulnerabilities detected in each scan is recorded.o A breakdown by severity level is also documented to highlight the distribution of risks.o The number of vulnerabilities serves as an indicator of the overall security state of an image and its potential exposure to threats.

2. Risk score
   A weighted risk score is calculated to quantify the severity of vulnerabilities.This score uses CVSS scores as weights, ensuring that vulnerabilities with higher potential impact contribute more significantly to the overall risk profile. The risk score provides a single, actionable metric to compare the security of different images and prioritize mitigation strategies.

3. Change patterns
   By comparing results across multiple scans, patterns in vulnerability behavior are identified :-
   Persistent vulnerabilities : Vulnerabilities that remain unresolved across multiple scans are flagged. These indicate potential maintenance issues or inherent risks in the image.
   Emergent vulnerabilities : Vulnerabilities that appear in subsequent scans after image updates are documented. This can highlight potential risks introduced by new dependencies or unaddressed security issues in updates.
   Resolved vulnerabilities : Vulnerabilities that are no longer detected in later scans are also tracked, providing evidence of effective mitigation efforts.

4.4.3 Scanning tools and techniques

The analysis leverages multiple industry-standard tools to ensure robust and comprehensive vulnerability detection:

- AWS Inspector : Known for its integration with cloud-native applications, AWS Inspector provides detailed insights into software vulnerabilities and configuration issues.

- Synk : A developer-friendly tool that focuses on open-source vulnerabilities and offers actionable remediation steps.

- Clair : An open-source tool designed specifically for container security, Clair analyzes images for known vulnerabilities in dependencies.

- Anchore : Anchore excels in providing policy-based compliance checks along with vulnerability analysis.

Using a combination of these tools ensures that vulnerabilities are identified with high accuracy and consistency. Cross-validation of results among tools further enhances the reliability of the findings.

4.4.4 Analytical insights

The insights derived from the vulnerability scanning process go beyond merely identifying security flaws. They also provide actionable intelligence for improving Docker image security.

- Severity trends : Analyzing the prevalence of low, medium, and high-severity vulnerabilities over time reveals patterns in image maintenance practices and potential areas for improvement.
- Update effectiveness : By examining vulnerabilities that persist or emerge after updates, the research evaluates how well maintainers address security issues during image updates.
- High-risk images : Identifying images with consistently high-risk scores helps prioritize efforts to secure critical components of the container ecosystem.

4.4.5 Ethical considerations

While conducting vulnerability scans, the research adheres to ethical guidelines to ensure responsible handling of sensitive information

- Permission compliance: Only publicly available Docker images are scanned, avoiding potential legal or ethical violations.
- Confidentiality: All findings are anonymized in the reporting process to prevent unintentional disclosure of sensitive vulnerabilities.
- Responsible disclosure : If critical vulnerabilities are identified, they will be responsibly disclosed to the maintainers of the affected images to facilitate remediation.

The vulnerability scanning and analysis methodology outlined above provides a robust framework for evaluating the security posture of Docker images. By leveraging industry-standard tools, extracting key metrics, and analyzing change patterns over time, the research generates actionable insights into container security. This systematic approach not only highlights existing vulnerabilities but also offers a pathway for enhancing the security practices of Docker image maintainers and users.

4.5 Risk prioritization

A risk-based approach will be used to prioritize vulnerabilities based on the below factors:

- Impact assessment : Each vulnerability's potential impact on confidentiality, integrity, and availability (CIA) will be evaluated.
- Exploitability analysis : The likelihood of exploitation, considering factors such as available patches and exploit code, will be analyzed.
- Risk ranking : Vulnerabilities will be ranked to guide the prioritization of remediation efforts.

4.6 Mitigation strategies

Based on the findings, mitigation strategies will be proposed which would address the following things:

- Patch management : Ensuring timely application of security patches to mitigate known vulnerabilities.

- Dependency management : Avoiding the use of outdated or insecure dependencies in container images.
- Best practices : Recommendations for secure image creation, including the use of minimal base images and multi-stage builds.

5 Implementation

The implementation process involved various key steps including configuration and installation of both the vulnerability assessment tools alongwith the infrastructure required to setup the necessary things on AWS cloud to ensure that everything from the integration from initial stages uptil the final stage was completed.
Below mentioned was the process followed throughout :-

- Dataset preparation
  First, the dataset of Docker images required for conducting the experiments and analysis was gathered from DockerHub ensuring to aim at different kinds of appplication stacks and configurations.
- Setting up the toolsets
  Open-source vulnerability scanning tools Trivy and Grype had been configured on the EC2 instance running on Ubuntu Operating System and these those tools were integrated with scripts to pull images from DockerHub and later push it to ECR,
- CI/CD pipeline
  AWS CodePipeline and CodeBuild were used to automate the scanning process of the Docker images. The pipeline had been configured to be triggered by changes committed to a GitHub repository containing the buildspec.yml file containing the instructions and custom scripts for having scans executed on Docker images hosted in ECR using Trivy and Grype.
- Data storage
  The results obtained from the scans conducted were stored in JSON format in an S3 bucket with the ImageDigest tags for further analysis.
- Visualization and Comparison
  By using a Python script on a Visual Studio Code IDE terminal, the results from the S3 bucket were downloaded on the local system to enable visualization of vulnerabilities. The script made use of in-built Python libraries like ”pandas” and ”matplotlib” to generate bar charts against metrics.
- AWS integration
  Scans on Docker images present in ECR were also performed using ECR’s native scanning feature alongwith configuration of AWS Inspector in the AWS region to perform scanning of images for getting better insights on vulnerabilities on individual basis as and when they are pushed to ECR

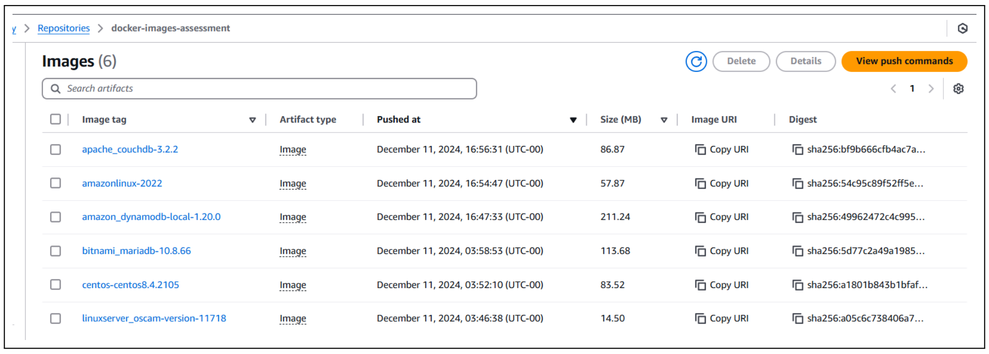


Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
apache_couchdb-3.2.2	Image	December 11, 2024, 16:56:31 (UTC-00)	86.87	Copy URI	sha256:bf9b666cfb4ac7a...
amazonlinux-2022	Image	December 11, 2024, 16:54:47 (UTC-00)	57.87	Copy URI	sha256:54c95c89f52ff5e...
amazon_dynamodb-local-1.20.0	Image	December 11, 2024, 16:47:33 (UTC-00)	211.24	Copy URI	sha256:49962472c4c995...
bitnami_mariadb-10.8.66	Image	December 11, 2024, 03:58:53 (UTC-00)	113.68	Copy URI	sha256:5d77c2a49a1985...
centos-centos8.4.2105	Image	December 11, 2024, 03:52:10 (UTC-00)	83.52	Copy URI	sha256:a1801b843b1bfaf...
linuxserver_oscam-version-11718	Image	December 11, 2024, 03:46:38 (UTC-00)	14.50	Copy URI	sha256:a05c6c738406a7...

Figure 2: Docker images in ECR

6 Evaluation

The efficiency of Trivy, Grype alongwith AWS ECR's native scanning to detect vulnerabilities present in Docker images was tested throughout against metrics and below are the findings sunnarized :-

- Efficiency in detection
  While both Trivy and Grype were able to detect vulnerabilities present in Docker images based on metrics like LOW, MEDIUM, CRTITICAL and HIGH, Trivy comparatively showed higher speed in identifying vulnerabilities and Grype generated more detailed level reports. AWS ECR scanning took even lesser amount of time than the other two in finding vulnerabilties while displaying much broader level of vulnerabilities

- Coverage
  Some results from the comparative analysis revealed that Trivy and Grype were able to detect unique vulnerabilities which were not identified by ECR which suggested that use of multiple tools is something that is required.

- Longitudinal analysis Certain amount of scans conducted throughout on images scanned previously showed the presence some new vulnerabilities which drew some emphasis on the need of continuous monitoring. Bar charts were used to depict levels of vulnerabilities in Docker images and have a comparison between results obtained from Trivy and Grype

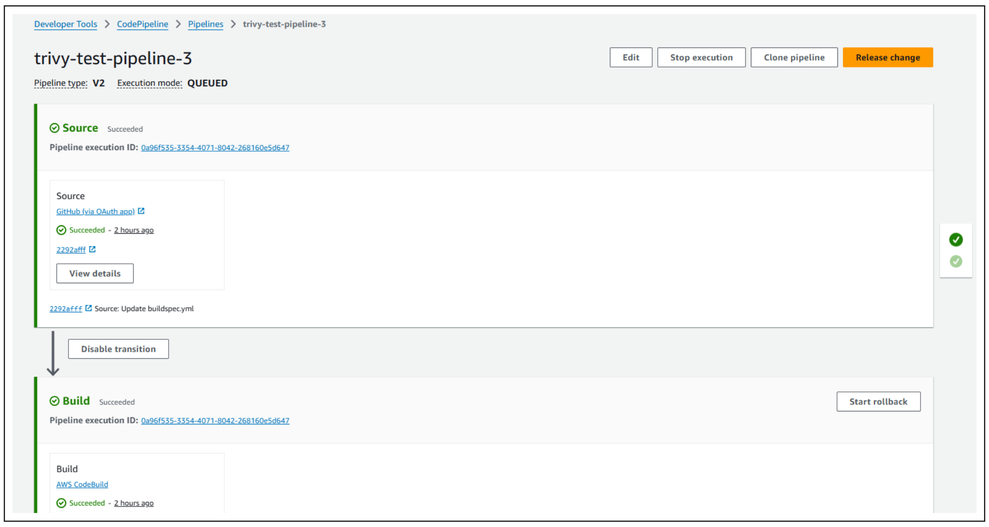
- Implications. On academic grounds, the study conducted for the research process tries to contribute to the area of container security by validating the efficiency of open-source vulnerability scanning assessment tools against AWS cloud services.

6.1 Experiments

Below demonstrated are the tests/experiments that were used during the implementation process

6.2 Discussion

The initial scans revealed a significant number of vulnerabilities across the selected Docker images. Community-maintained images exhibited higher vulnerability counts compared



Developer Tools > CodePipeline > Pipelines > trivy-test-pipeline-3

trivy-test-pipeline-3

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded

Pipeline execution ID: 0a96f535-3354-4071-8042-268160e5d647

Source
GitHub (via OAuth app)
Succeeded - 2 hours ago
2292afff

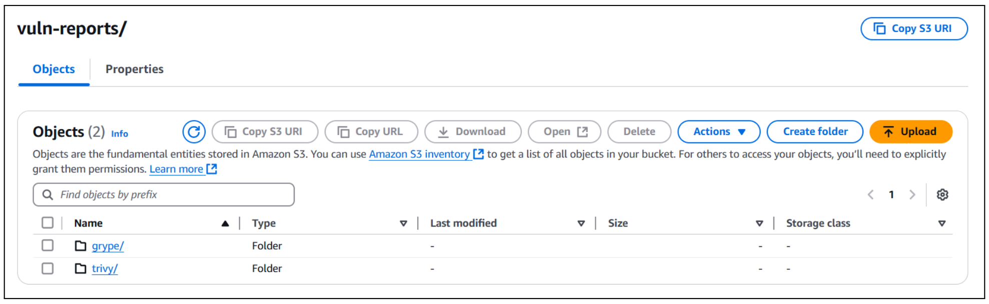
2292afff Source: Update buildspec.yml

Build Succeeded

Pipeline execution ID: 0a96f535-3354-4071-8042-268160e5d647

Build
AWS CodeBuild
Succeeded - 2 hours ago

Figure 3: CI/CD pipeline for automating vulnerability scans



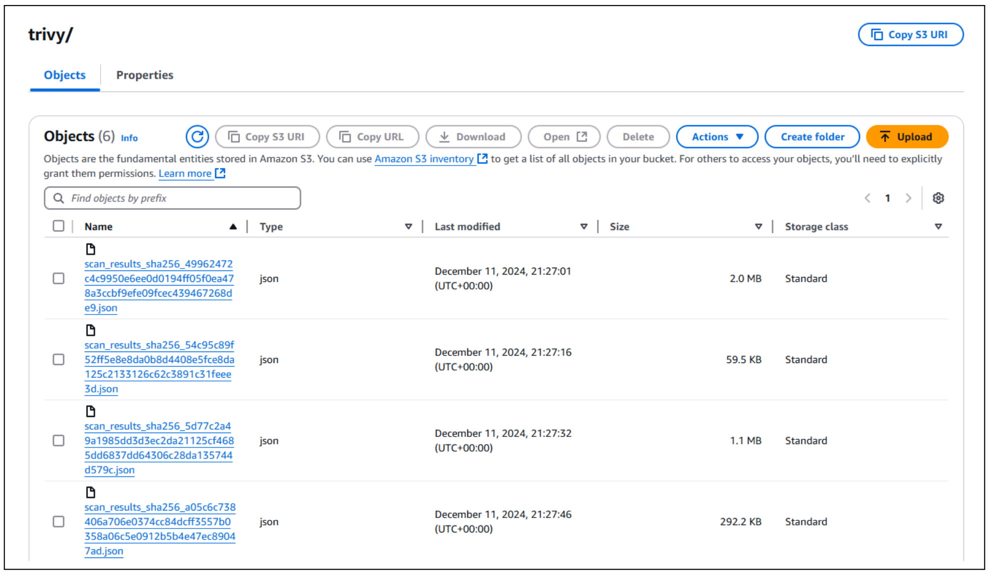
vuln-reports/

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
grype/	Folder	-	-	-
trivy/	Folder	-	-	-

Figure 4: S3 bucket containing the scan directories



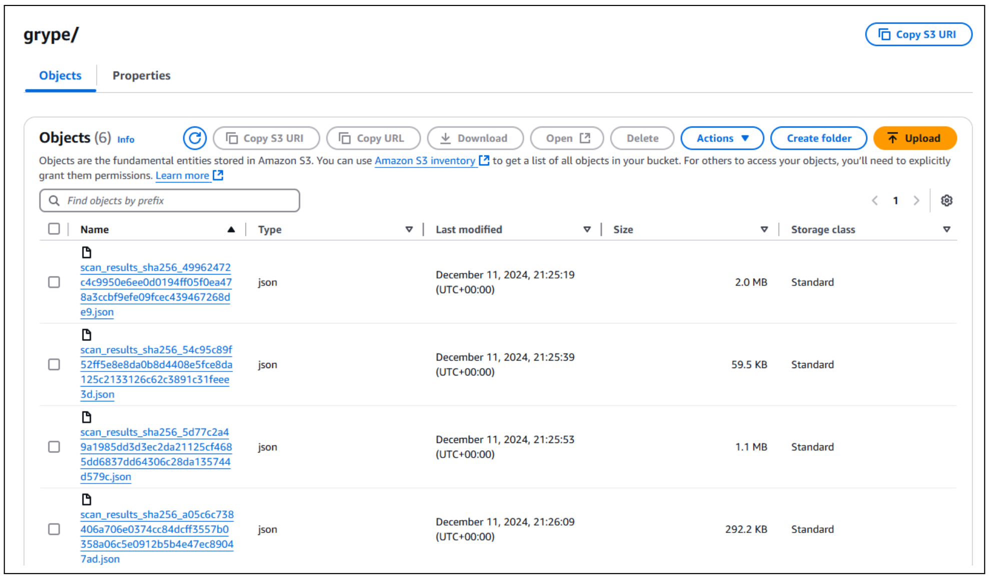
trivy/

Objects (6)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
scan_results_sha256_49962472c4c9950e6ee0d0194ff05f0ea478a3ccbf9efe09fcec439467268de9.json	json	December 11, 2024, 21:27:01 (UTC+00:00)	2.0 MB	Standard
scan_results_sha256_54c95c89f52ff5e8e8da0b8d4408e5fce8da125c2133126c62c3891c31feee3d.json	json	December 11, 2024, 21:27:16 (UTC+00:00)	59.5 KB	Standard
scan_results_sha256_5d77c2a49a1985dd3d3ec2da21125cf4685dd6837dd64306c28da135744d579c.json	json	December 11, 2024, 21:27:32 (UTC+00:00)	1.1 MB	Standard
scan_results_sha256_a05c6c738406a706e0374cc84dcff3557b0358a06c5e0912b5b4e47ec89047ad.json	json	December 11, 2024, 21:27:46 (UTC+00:00)	292.2 KB	Standard

Figure 5: JSON output files in S3 of Trivy scans



grype/

Objects | Properties

Objects (6)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
scan_results_sha256_49962472c4c9950e6ee0d0194ff05f0ea478a3ccbf9efe09fcec439467268de9.json	json	December 11, 2024, 21:25:19 (UTC+00:00)	2.0 MB	Standard
scan_results_sha256_54c95c89f52ff5e8e8da0b8d4408e5fce8da125c2133126c62c3891c31feee3d.json	json	December 11, 2024, 21:25:39 (UTC+00:00)	59.5 KB	Standard
scan_results_sha256_5d77c2a49a1985dd3d3ec2da21125cf4685dd6837dd64306c28da135744d579c.json	json	December 11, 2024, 21:25:53 (UTC+00:00)	1.1 MB	Standard
scan_results_sha256_a05c6c738406a706e0374cc84dcff3557b0358a06c5e0912b5b4e47ec89047ad.json	json	December 11, 2024, 21:26:09 (UTC+00:00)	292.2 KB	Standard

Figure 6: JSON output files in S3 of Grype scans

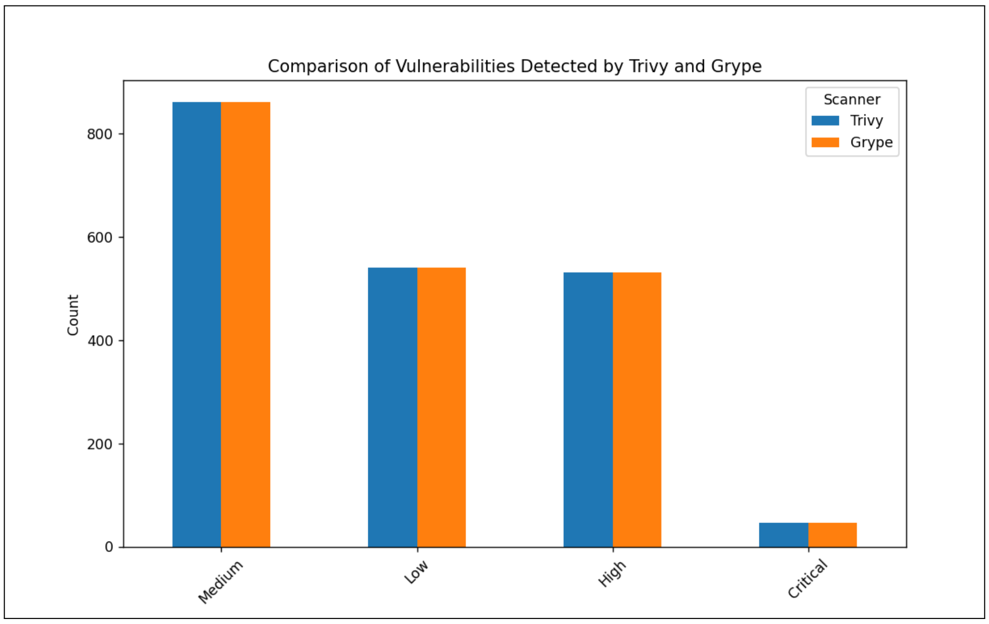
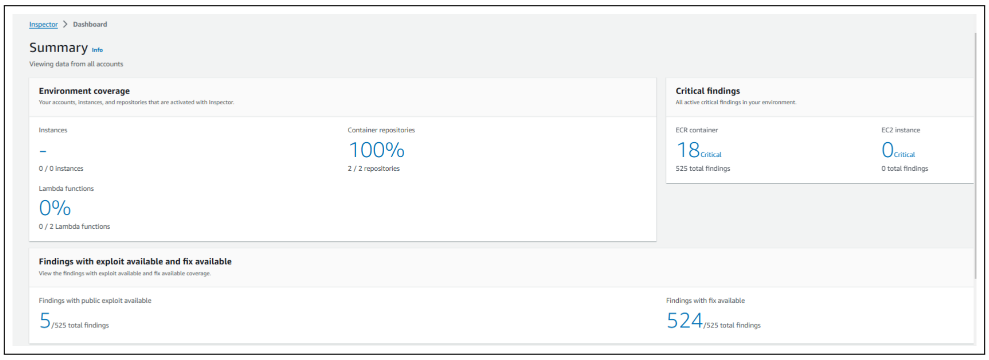


Figure 7: Visualize scan results through Python script



Inspector > Dashboard

Summary

Viewing data from all accounts

Environment coverage

Your accounts, instances, and repositories that are activated with Inspector.

Instances: - (0 / 0 instances)

Container repositories: 100% (2 / 2 repositories)

Lambda functions: 0% (0 / 2 Lambda functions)

Critical findings

All active critical findings in your environment.

ECR container: 18 Critical (525 total findings)

EC2 instance: 0 Critical (0 total findings)

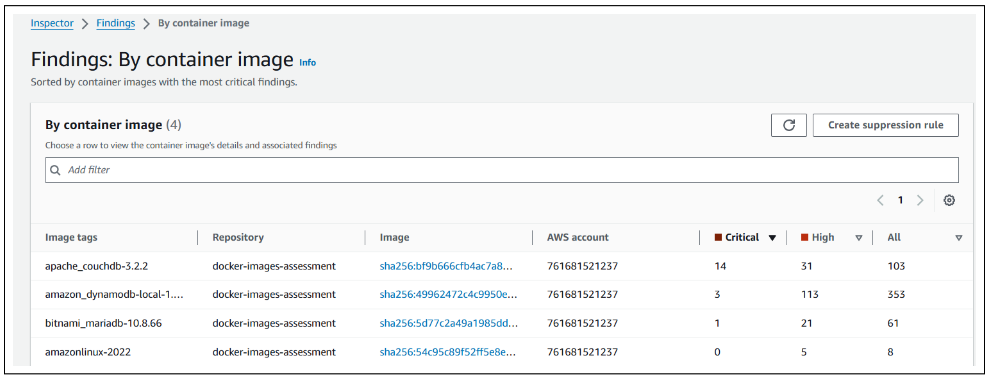
Findings with exploit available and fix available

View the findings with exploit available and fix available coverage.

Findings with public exploit available: 5/525 total findings

Findings with fix available: 524/525 total findings

Figure 8: AWS Inspector findings - Summary dashboard


Inspector > Findings > By container image

Findings: By container image Info

Sorted by container images with the most critical findings.

By container image (4)

Choose a row to view the container image's details and associated findings

Image tags	Repository	Image	AWS account	🟥 Critical	🟥 High	All
apache_couchdb-3.2.2	docker-images-assessment	sha256:bf9b666cfb4ac7a8...	761681521237	14	31	103
amazon_dynamodb-local-1....	docker-images-assessment	sha256:49962472c4c9950e...	761681521237	3	113	353
bitnami_mariadb-10.8.66	docker-images-assessment	sha256:5d77c2a49a1985dd...	761681521237	1	21	61
amazonlinux-2022	docker-images-assessment	sha256:54c95c89f52ff5e8e...	761681521237	0	5	8

Figure 9: AWS Inspector findings - By container image

to official images, highlighting the importance of source credibility. The majority of vulnerabilities were medium or high severity, necessitating immediate attention.

6.2.1 Longitudinal vulnerability trends

Over the observation period, certain patterns emerged which are highlighted below :

7 Conclusion and Future Work

This research highlights the importance of longitudinal security assessments in addressing the dynamic nature of Docker container vulnerabilities. By monitoring changes over time and adopting a risk-based approach, organizations can proactively manage container security risks.
Some recommendations derived from the study in this research include the following :

- Continuous monitoring
  Implement tools like Snyk for real-time vulnerability tracking.
- Patch management
  Establish automated workflows for applying patches to container dependencies.
- Secure development practices
  Adopt minimal base images and follow secure coding practices.

Future research could explore integrating AI and machine learning techniques to predict vulnerability trends and enhance the effectiveness of longitudinal assessments. Additionally, studying the impact of emerging technologies, such as confidential computing, on container security could provide valuable insights.

References

Hradec, J., Craglia, M., Di Leo, M., De Nigris, S., Ostlaender, N., & Nicholson, N. (2022). Multipurpose synthetic population for policy applications. No. JRC128595

Khang, A., Rana, G., Tailor, R. K., & Abdullayev, V. (Eds.). (2023). Data-centric AI solutions and emerging technologies in the healthcare ecosystem.

Kim, H., Kim, Y., & Kim, S. (2024). A Study on the Security Requirements Analysis to build a Zero Trust-based Remote Work Environment. arXiv preprint arXiv:2401.03675.

Mens, T. (2024, April). Mitigating Security Issues in GitHub Actions. In 2024 ACM/IEEE 4th Interna-tional Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS) and 2024 IEEE/ACM 2nd International Workshop on Software Vulnerability. ACM/IEEE.

Mills, A., White, J., & Legg, P. (2023). Longitudinal risk-based security assessment of docker soft-ware container images. Computers & Security, 135, 103478.

Efe, Do¸c. Dr. Ahmet, & Aslan, Ula¸s & Kara, Aytekin. (2020). Securing Vulnerabilities in Docker Images. International Journal of Innovative Engineering Applications. 4. 31-39. 10.46460/ijiea.617181.

Royer, P. D., Du, W., & Schneider, K. (2022). Rapid evaluation and response to impacts on crit-ical end-use loads following natural hazard-driven power outages: A modular and responsive geospatial technology. International Journal of Disaster Risk Science, 13 (3), 415-434.

Schouten, G., Arena, G., van Leeuwen, F., Heck, P., Mulder, J., Aalbers, R., ... & B¨oing-Messing, F. (2023). Data Analytics in Action. In Data Science for Entrepreneurship: Principles and Methods for Data Engineering, Analytics, Entrepreneurship, and the Society (pp. 205-233). Cham: Springer International Publishing

Schweinar, A., Wagner, F., Klingner, C., Festag, S., Spreckelsen, C., & Brodoehl, S. (2024). Simplifying Multimodal Clinical Research Data Management: Introducing an Integrated and User-Friendly Database Concept. Applied Clinical Informatics.

Sion, L., Van Landuyt, D., Yskout, K., Verreydt, S., & Joosen, W. (2023). Ctam: A Tool for Continuous Threat Analysis and Management. In CyberSecurity in a DevOps Environment: From Requirements to Monitoring (pp. 195-223). Cham: Springer Nature Switzerland.

Strandberg, P. E., Afzal, W., & Sundmark, D. (2022). Software test results exploration and visual-ization with continuous integration and nightly testing. International Journal on Software Tools for Technology Transfer, 24 (2), 261-285.

Thorn, J., Strandberg, P. E., Sundmark, D., & Afzal, W. (2022). Quality assuring the quality assurance tool: applying safety-critical concepts to test framework development. PeerJ Computer Science, 8, e1131.