Container Performance and Vulnerability Management for Container Security Using Docker Engine

TAHIR ALYAS , SIKANDAR ALI , HABIB ULLAH KHAN , ALI SAMAD , KHALID ALISSA , MUHAMMAD ASIF SALEEM

Jaison Dennis
CS7A
20CSA34

Contents

1. Container Performance and Vulnerability Management for Container Security Using Docker Engine
2. Introduction
3. Literature Survey
4. Proposed Docker-Sec System
5. Docker-Sec Architecture
6. Evaluation Methodology
7. Container Isolation
8. Use Cases
9. Conclusions

Introduction

- Containers have much less overhead than virtual machines when they run through a kernel that they share with the host computer as user-space operations.
- The device modules can also be used as lightweight units, and their delivery and execution are simplified.
- It allows for the automatic control of large-scale systems .
- In short, everything involves building the container in whatever running environment, application, and all the libraries, triggers, and configuration files needed to run it.
- Since containers and hosts use the same kernel, malicious containers can quickly leave their environment and make host kernel attacks possible.
- Mandatory kernel access control is the best way to improve the security of a Linux container by using tools such as AppArmor or SELinux to prevent unintended operations on both the host and container sides

Docker Components

- Engine:
  - Client-server program responsible for creating and running Docker containers.
- Client :
  - Command-line interface (CLI) tool that allows users to interact with the Docker daemon and perform various operations related to Docker containers and images.
- Daemon :
  - Background process that runs on a host machine and is responsible for managing Docker containers.
  - Core component of the Docker Engine and handles the creation, execution, and monitoring of containers.
- Containerd :
  - Open-Source container runtime that provides a platform-agnostic interface for managing container lifecycle operations, such as creating, running, and deleting containers.

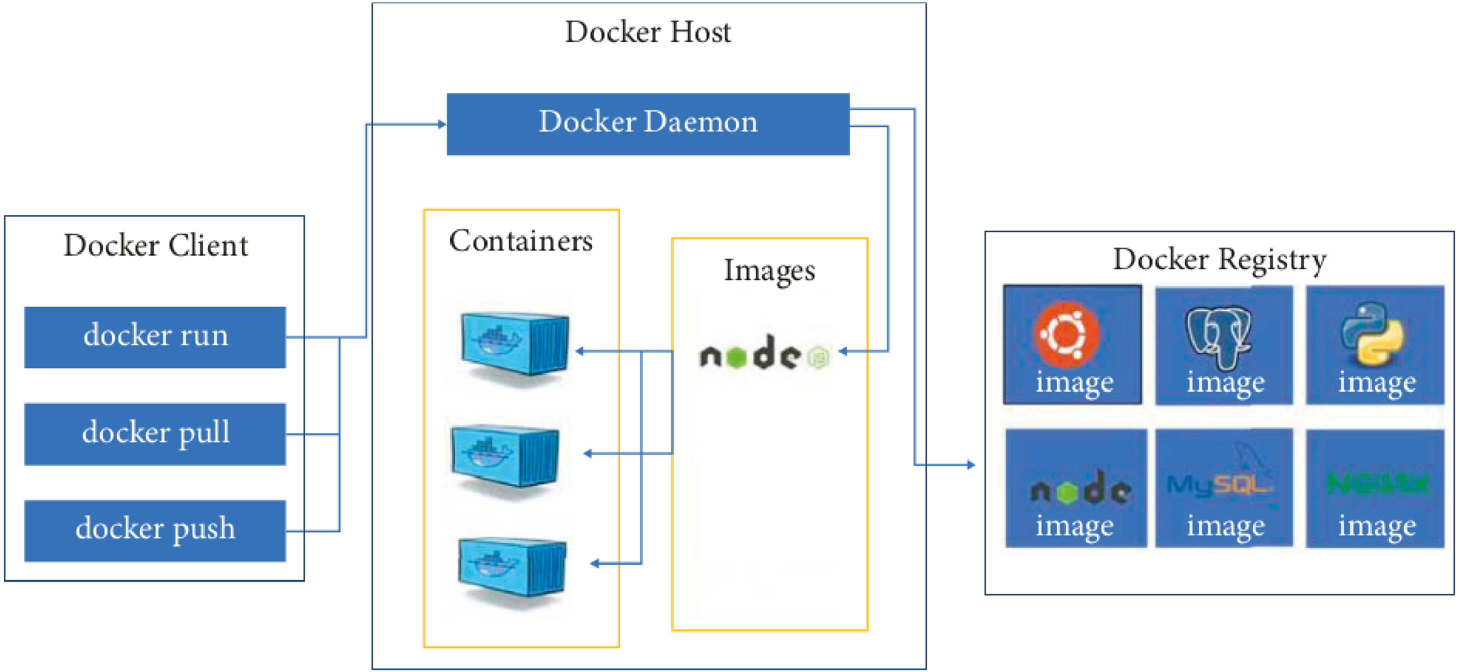


Figure 1: Docker architecture.

Container Image Creation and Lifecycle Management

- Image Creation:
  - Images are the building blocks of containers.
  - Contains the necessary files, libraries, and configurations required to run an application.
  - Images can be created manually or automatically using tools like Dockerfile, which specifies the steps to build an image.
  - The process involves selecting a base image, adding dependencies, configuring the environment, and packaging the application code.

Container Image Creation and Lifecycle Management

- Image Distribution:
  - Container images need to be distributed across different environments, such as development, testing, and production.
  - This can be done manually or through automated processes like continuous integration and deployment pipelines.
  - Proper image distribution ensures consistency and reproducibility across different environments.

Container Image Creation and Lifecycle Management

- Image Registry:
  - Container images are stored in a registry, which acts as a cenralized repository.
  - Registry allows users to push and pull images, making them accessible to different environments and users.
  - Examples are Docker Hub, Google Container Registry, and Amazon Elastic Container Registry.
- Image Security:
  - Container images should be scanned for vulnerabilities and security issues before deployment.
  - Vulnerability scanning tools can analyze the image and identify any known vulnerabilities in the software packages and libraries included in the image.
  - Regular scanning and updating of images help mitigate security risks.

Literature Survey

- **N. Tabassum, T. Alyas, M. Hamid, M. Saleem, S. Malik, and S. Binish Zahra, “Qos based cloud security evaluation using neuro fuzzy model,”**
  - DIVA System
  - Analyzed 356,218 images :
    - Contain more than 180 errors on average, taking into account all models,
    - Many images have not been updated for hundreds of days
- **M. Mohamed, R. Engel, A. Warke, S. Berman, and H. Ludwig, “Extensible persistence as a service for containers,”**
  - Explored the vulnerability-oriented of the Docker Environment
  - Explored safety consequences of using containers on traditional applications.
  - Docker can help in all solutions for container security.

Literature Survey

- **F. D’Urso, C. Santoro, and F. F. Santoro, “Wale: a solution to share libraries in Docker containers,”**
  - Described the security issues of containers and the problems associated with containers being lightweight and using the same kernel as the Host operating system.
  - Presents four cases and solutions obtained.
    - Securing the container from the applications inside it,
    - Inter-container protection,
    - Protecting the host from the containers
    - Securing the containers from a malicious or semi-honest host.

Literature Survey

- **S. Sultan, I. Ahmad, and T. Dimitriou, “Container security: issues, challenges, and the road ahead,”**
  - Described Docker and its performance analysis and took the stance that Docker has a protected layer on the container
  - Docker used a tool known as “Docker Engine” to execute the applications.
  - Provided a Docker Hub for sharing applications it worked the same as virtual machines.
  - Described the advantages of Docker containers over virtual machines.

Proposed Docker-Sec System

- Implements mandatory access control policies
- Constraints containers based on expected usage
- Static analysis for initial rules, dynamic monitoring adds rules

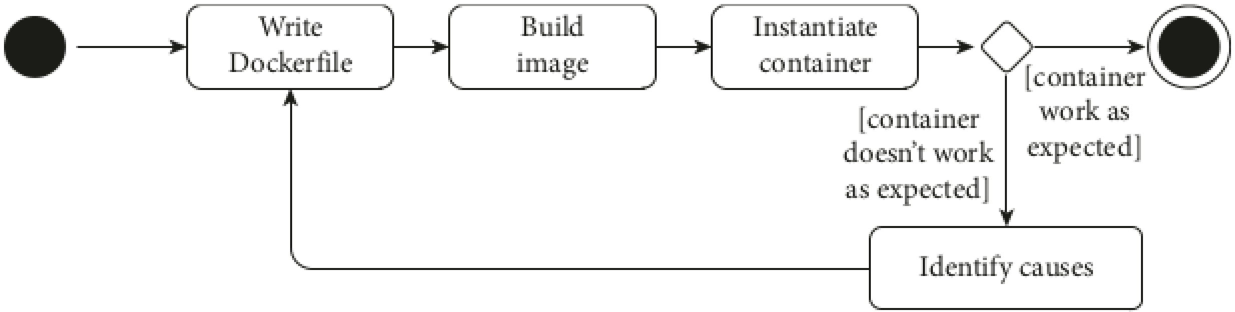

Write Dockerfile
Build image
Instantiate container
[container work as expected]
[container doesn't work as expected]
Identify causes

FIGURE 3: workflow when developing a Dockerfile.

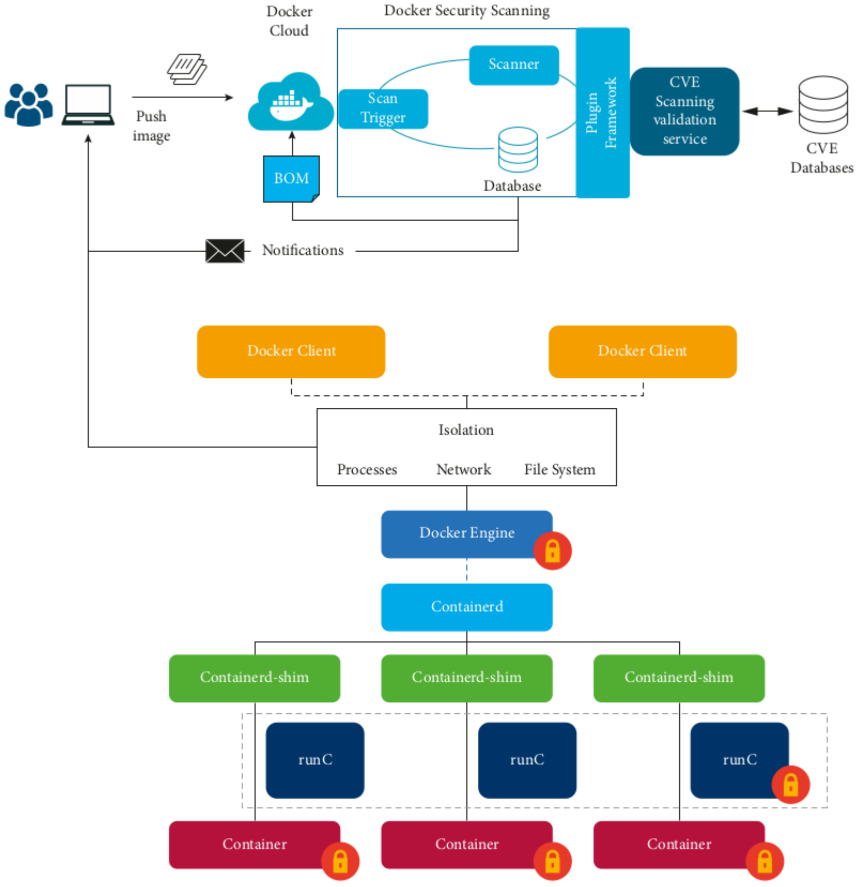


Figure 2: Proposed system architecture.

Docker-Sec Architecture

- Docker client, daemon, containerd runtime
- Static analyzer, dynamic monitor
- AppArmor profiles containers
- Vulnerability scanner checks images

Creating Container Profiles

- Extracts rules from config and expected usage
- Monitors container during training phase
- Limits capabilities to minimum required

Creating AppArmor Profiles

- Docker-sec aims to create highly customized AppArmor profiles for each container to enhance container security .
- Two main strategies:
  - Static Analysis
  - Dynamic Testing.

Static Analysis

- Generate the initial Docker profiles.
- Gathers valuable static information about the container and accesses its configuration.
- Collects information such as the container name, version, package manager, description of the fundamental components, and known vulnerabilities associated with those components.
- This information is used to construct the initial set of access rules for the container .

Dynamic Testing

- Improve the Docker profiles during container runtime.
- Monitors the container’s behavior and extracting additional rules that further constrain the container’s capabilities.
- Allows Docker-sec to represent the actual application behavior, file system, processes, and network isolation in the profiles.
- By tracking the container’s execution in real-time, Docker-sec can extract rules that provide more rigorous protection if needed .

Building Secured RunC Profile

- RunC directly interacts with containers
- RunC
  - Lightweight, portable command-line tool that provides the runtime environment for containers.
  - Open-source implementation of the Open Container Initiative (OCI) runtime specification .
  - Responsible for creating and managing containers based on OCI-compliant container images.
- Locked down profile prevents host access
- Allows only essential capabilities

Securing Docker Daemon

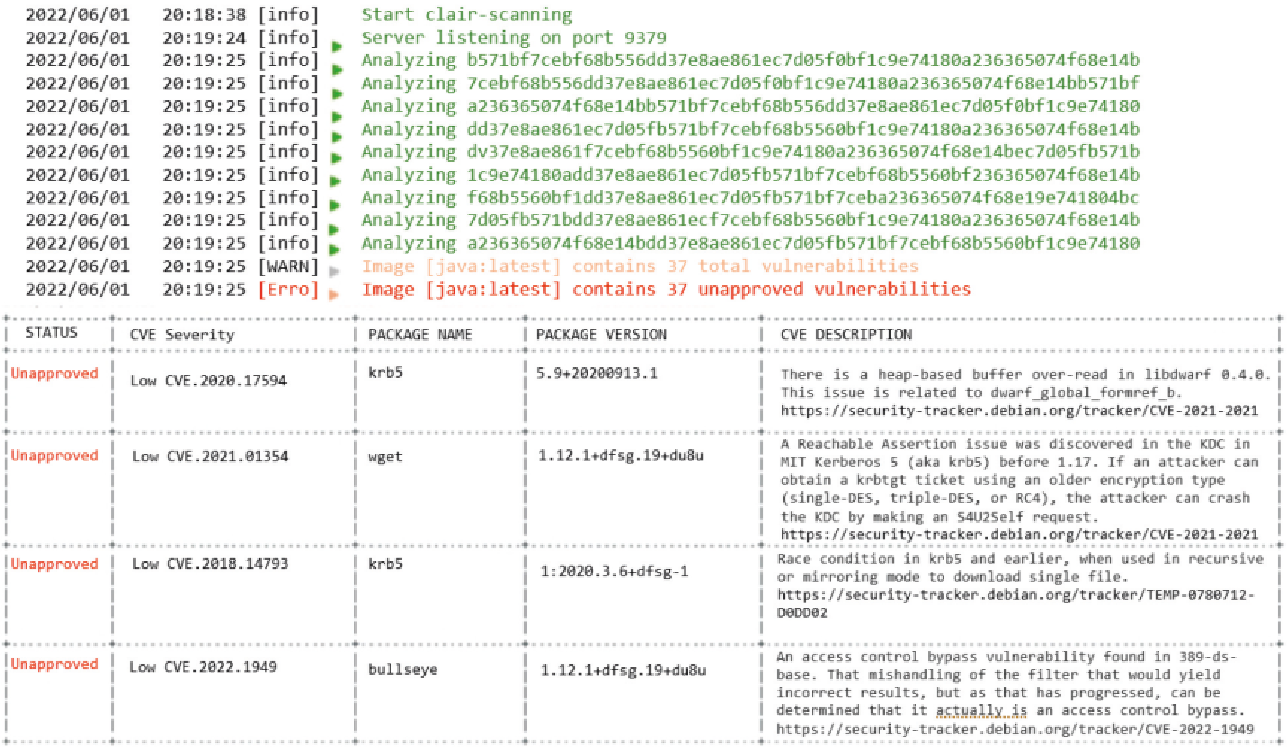
- Daemon runs and manages containers
- Profile limits access to required services
- Prevents unauthorized changes

Process Isolation

- PID namespaces separate container processes
- Capabilities limit process interactions
- Prevents inter-container attacks

Filesystem Isolation

- Mount namespaces separate filesystem
- Remove capabilities to limit access
- Protects host filesystem from containers



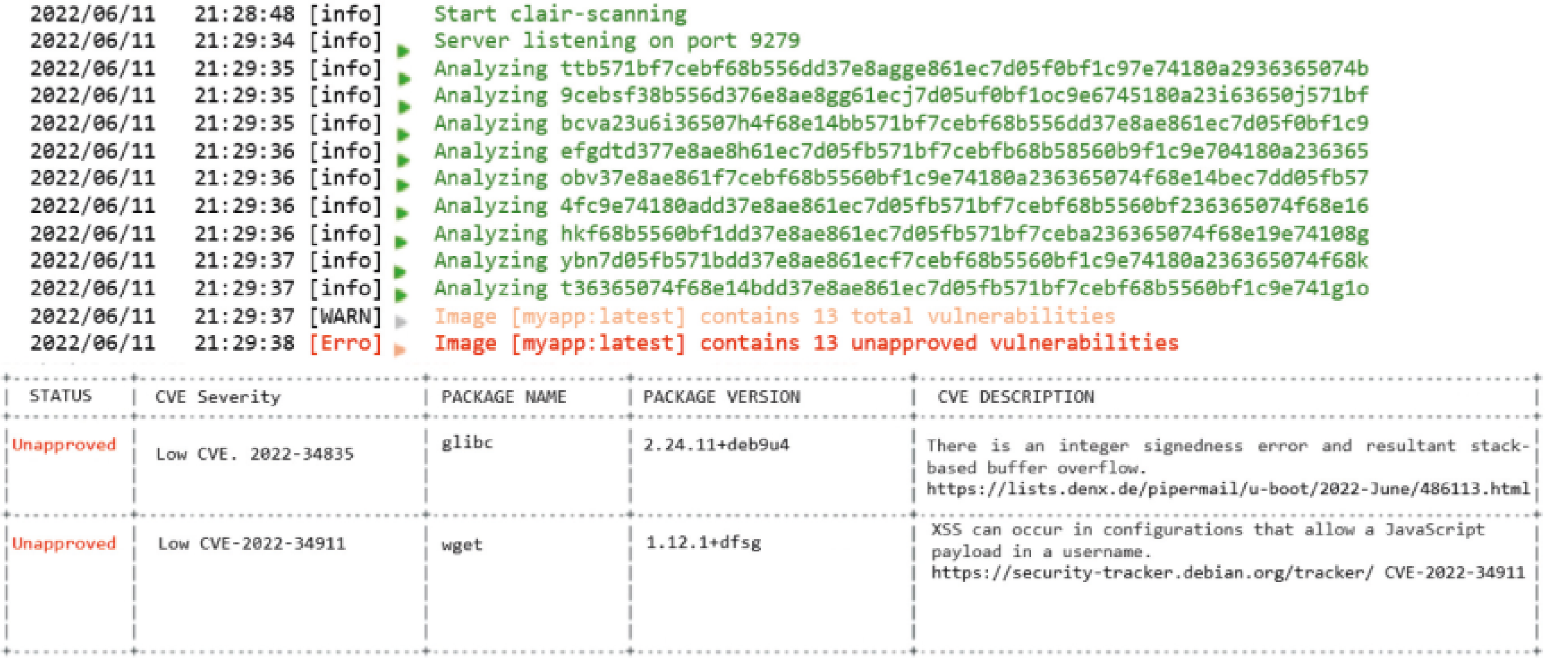
```
2022/06/01   20:18:38 [info]    Start clair-scanning
2022/06/01   20:19:24 [info]    Server listening on port 9379
2022/06/01   20:19:25 [info] ▶  Analyzing b571bf7cebf68b556dd37e8ae861ec7d05f0bf1c9e74180a236365074f68e14b
2022/06/01   20:19:25 [info] ▶  Analyzing 7cebf68b556dd37e8ae861ec7d05f0bf1c9e74180a236365074f68e14bb571bf
2022/06/01   20:19:25 [info] ▶  Analyzing a236365074f68e14bb571bf7cebf68b556dd37e8ae861ec7d05f0bf1c9e74180
2022/06/01   20:19:25 [info] ▶  Analyzing dd37e8ae861ec7d05fb571bf7cebf68b5560bf1c9e74180a236365074f68e14b
2022/06/01   20:19:25 [info] ▶  Analyzing dv37e8ae861f7cebf68b5560bf1c9e74180a236365074f68e14bec7d05fb571b
2022/06/01   20:19:25 [info] ▶  Analyzing 1c9e74180add37e8ae861ec7d05fb571bf7cebf68b5560bf236365074f68e14b
2022/06/01   20:19:25 [info] ▶  Analyzing f68b5560bf1dd37e8ae861ec7d05fb571bf7ceba236365074f68e19e741804bc
2022/06/01   20:19:25 [info] ▶  Analyzing 7d05fb571bdd37e8ae861ecf7cebf68b5560bf1c9e74180a236365074f68e14b
2022/06/01   20:19:25 [info] ▶  Analyzing a236365074f68e14bdd37e8ae861ec7d05fb571bf7cebf68b5560bf1c9e74180
2022/06/01   20:19:25 [WARN] ▶  Image [java:latest] contains 37 total vulnerabilities
2022/06/01   20:19:25 [Erro] ▶  Image [java:latest] contains 37 unapproved vulnerabilities
```

```
+------------+-----------------------+----------------+---------------------+------------------------------------------------------------+
| STATUS     | CVE Severity          | PACKAGE NAME   | PACKAGE VERSION     | CVE DESCRIPTION                                            |
+------------+-----------------------+----------------+---------------------+------------------------------------------------------------+
| Unapproved | Low CVE.2020.17594    | krb5           | 5.9+20200913.1      | There is a heap-based buffer over-read in libdwarf 0.4.0.  |
|            |                       |                |                     | This issue is related to dwarf_global_formref_b.           |
|            |                       |                |                     | https://security-tracker.debian.org/tracker/CVE-2021-2021  |
+------------+-----------------------+----------------+---------------------+------------------------------------------------------------+
| Unapproved | Low CVE.2021.01354    | wget           | 1.12.1+dfsg.19+du8u | A Reachable Assertion issue was discovered in the KDC in   |
|            |                       |                |                     | MIT Kerberos 5 (aka krb5) before 1.17. If an attacker can  |
|            |                       |                |                     | obtain a krbtgt ticket using an older encryption type      |
|            |                       |                |                     | (single-DES, triple-DES, or RC4), the attacker can crash   |
|            |                       |                |                     | the KDC by making an S4U2Self request.                     |
|            |                       |                |                     | https://security-tracker.debian.org/tracker/CVE-2021-2021  |
+------------+-----------------------+----------------+---------------------+------------------------------------------------------------+
| Unapproved | Low CVE.2018.14793    | krb5           | 1:2020.3.6+dfsg-1   | Race condition in krb5 and earlier, when used in recursive |
|            |                       |                |                     | or mirroring mode to download single file.                 |
|            |                       |                |                     | https://security-tracker.debian.org/tracker/TEMP-0780712-  |
|            |                       |                |                     | D0DD02                                                     |
+------------+-----------------------+----------------+---------------------+------------------------------------------------------------+
| Unapproved | Low CVE.2022.1949     | bullseye       | 1.12.1+dfsg.19+du8u | An access control bypass vulnerability found in 389-ds-    |
|            |                       |                |                     | base. That mishandling of the filter that would yield      |
|            |                       |                |                     | incorrect results, but as that has progressed, can be      |
|            |                       |                |                     | determined that it actually is an access control bypass.   |
|            |                       |                |                     | https://security-tracker.debian.org/tracker/CVE-2022-1949  |
+------------+-----------------------+----------------+---------------------+------------------------------------------------------------+
```

Figure 5: Docker image scanning with file system isolation.

Network Isolation

- Network namespaces for separate stacks
- Limit connectivity between containers
- Prevents snooping and MITM attacks



```
2022/06/11   21:28:48 [info]    Start clair-scanning
2022/06/11   21:29:34 [info] ▶  Server listening on port 9279
2022/06/11   21:29:35 [info] ▶  Analyzing ttb571bf7cebf68b556dd37e8agge861ec7d05f0bf1c97e74180a2936365074b
2022/06/11   21:29:35 [info] ▶  Analyzing 9cebsf38b556d376e8ae8gg61ecj7d05uf0bf1oc9e6745180a23i63650j571bf
2022/06/11   21:29:35 [info] ▶  Analyzing bcva23u6i36507h4f68e14bb571bf7cebf68b556dd37e8ae861ec7d05f0bf1c9
2022/06/11   21:29:36 [info] ▶  Analyzing efgdtd377e8ae8h61ec7d05fb571bf7cebfb68b58560b9f1c9e704180a236365
2022/06/11   21:29:36 [info] ▶  Analyzing obv37e8ae861f7cebf68b5560bf1c9e74180a236365074f68e14bec7dd05fb57
2022/06/11   21:29:36 [info] ▶  Analyzing 4fc9e74180add37e8ae861ec7d05fb571bf7cebf68b5560bf236365074f68e16
2022/06/11   21:29:36 [info] ▶  Analyzing hkf68b5560bf1dd37e8ae861ec7d05fb571bf7ceba236365074f68e19e74108g
2022/06/11   21:29:37 [info] ▶  Analyzing ybn7d05fb571bdd37e8ae861ecf7cebf68b5560bf1c9e74180a236365074f68k
2022/06/11   21:29:37 [info] ▶  Analyzing t36365074f68e14bdd37e8ae861ec7d05fb571bf7cebf68b5560bf1c9e741g1o
2022/06/11   21:29:37 [WARN] ▶  Image [myapp:latest] contains 13 total vulnerabilities
2022/06/11   21:29:38 [Erro] ▶  Image [myapp:latest] contains 13 unapproved vulnerabilities
+------------+---------------------+---------------+-------------------+-------------------------------------------------------------------+
| STATUS     | CVE Severity        | PACKAGE NAME  | PACKAGE VERSION   | CVE DESCRIPTION                                                   |
+------------+---------------------+---------------+-------------------+-------------------------------------------------------------------+
| Unapproved | Low CVE. 2022-34835 | glibc         | 2.24.11+deb9u4    | There is an integer signedness error and resultant stack-         |
|            |                     |               |                   | based buffer overflow.                                            |
|            |                     |               |                   | https://lists.denx.de/pipermail/u-boot/2022-June/486113.html      |
+------------+---------------------+---------------+-------------------+-------------------------------------------------------------------+
| Unapproved | Low CVE-2022-34911  | wget          | 1.12.1+dfsg       | XSS can occur in configurations that allow a JavaScript           |
|            |                     |               |                   | payload in a username.                                            |
|            |                     |               |                   | https://security-tracker.debian.org/tracker/ CVE-2022-34911       |
|            |                     |               |                   |                                                                   |
+------------+---------------------+---------------+-------------------+-------------------------------------------------------------------+
```

Figure 6: Docker image with network isolation scanning.

Vulnerability Scanning

- Checks images against CVE databases
- Identifies flaws like SQLi, XSS, injections
- Addresses vulnerabilities proactively

Evaluation Methodology

- Compare secured vs unsecured containers
- Different workloads - CPU, memory, disk, network
- Measure performance overhead

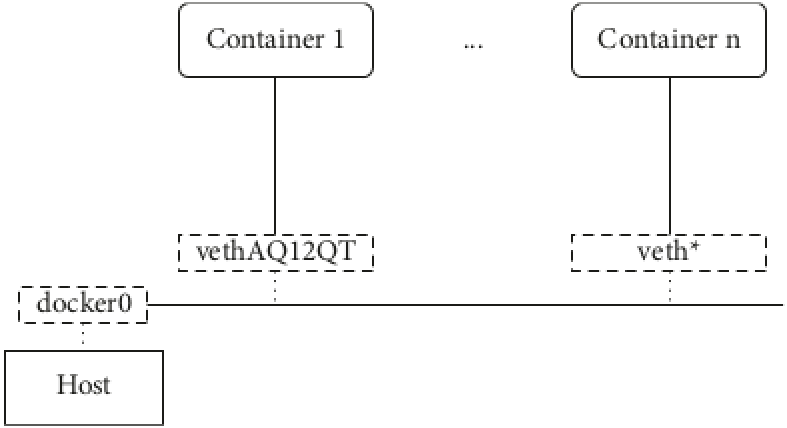


Figure 4: Docker's networking model.

Performance Overhead

- Low overhead around 2-4% for container startup
- Minimal impact on application performance
- Acceptable cost for significantly improved security

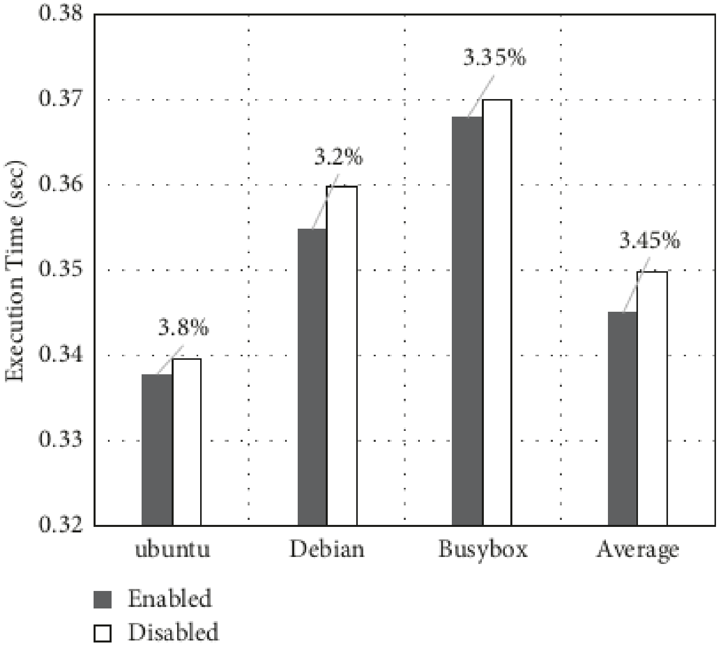


Figure 7: Docker-Sec's performance overhead.

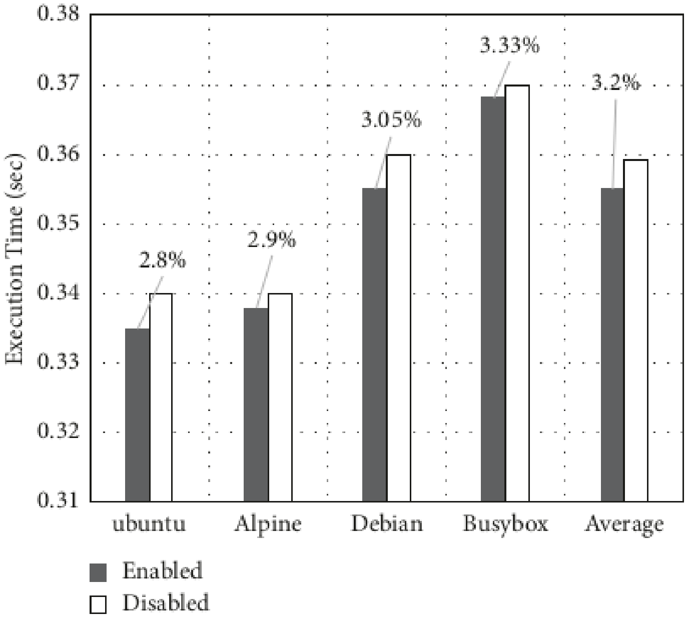


Figure 8: Docker-Sec's performance overhead with different images.

Container Isolation

- Prevents inter-container attacks and limits host access
- Reduces attack surface through restricted capabilities

Use Cases

- WordPress container deployment
- Arbitrary application containers
- Simulated attacks for validation

Conclusions

- Docker-sec constrains containers to only legitimate access -Low performance overhead around 2-4%
- Significantly enhances container security
- Additional Docker client commands for security functions
- User interface provides visibility into profiles

Questions

Thank You