



# **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



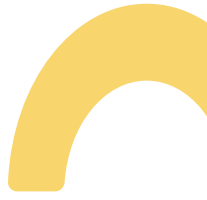
CS7A

20CSA34



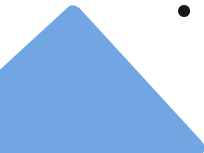

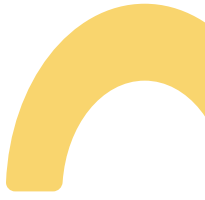


# CONTENTS

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



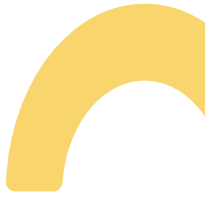


# INTRODUCTION

- 
- Containers are lightweight virtualization that shares host kernel
  - This leads to security issues like host access and limited isolation
  - Docker is the most popular container technology
  - Docker-sec secures containers by implementing access policies to limit capabilities
- 
- 



# BACKGROUND


- Containers vs virtual machines - architecture, performance, isolation
  - Docker components - engine, client, daemon, containerd
  - Container image creation and lifecycle management
- 
- 
- 

# CONTAINER VS VIRTUAL MACHINES

- Overhead:
  - Containers have less overhead than virtual machines.
  - Containers run through a shared kernel with the host computer, resulting in lower resource usage compared to VMs .
- Isolation:
  - Containers are considered to be more vulnerable to attacks.
  - Containers and hosts share the same kernel, malicious containers can potentially exploit the host kernel.



# CONTAINER VS VIRTUAL MACHINES

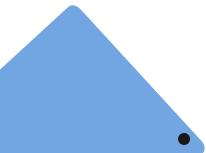
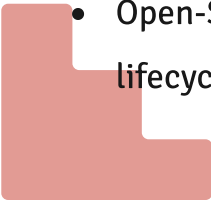
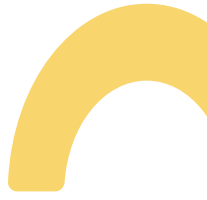
- 
- Deployment Efficiency:
    - Containers can run on the same operating system, making the server more efficient and allowing for faster application deployment.
    - Multiple containers can be bundled into a single operating system, reducing complexity and dependencies .
  - Security Concerns
    - Containers have been found to have security problems, including malicious container images and vulnerabilities in container applications.
    - Automated and standardized approaches are needed to implement security updates to Docker images

# CONTAINER VS VIRTUAL MACHINES

- Performance:
  - Docker containers, in particular, are described as having better performance and lower computational cost compared to virtual machines.
  - They offer advantages such as speed, portability, scalability, and rapid delivery.



# 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.
- 
- 
- 



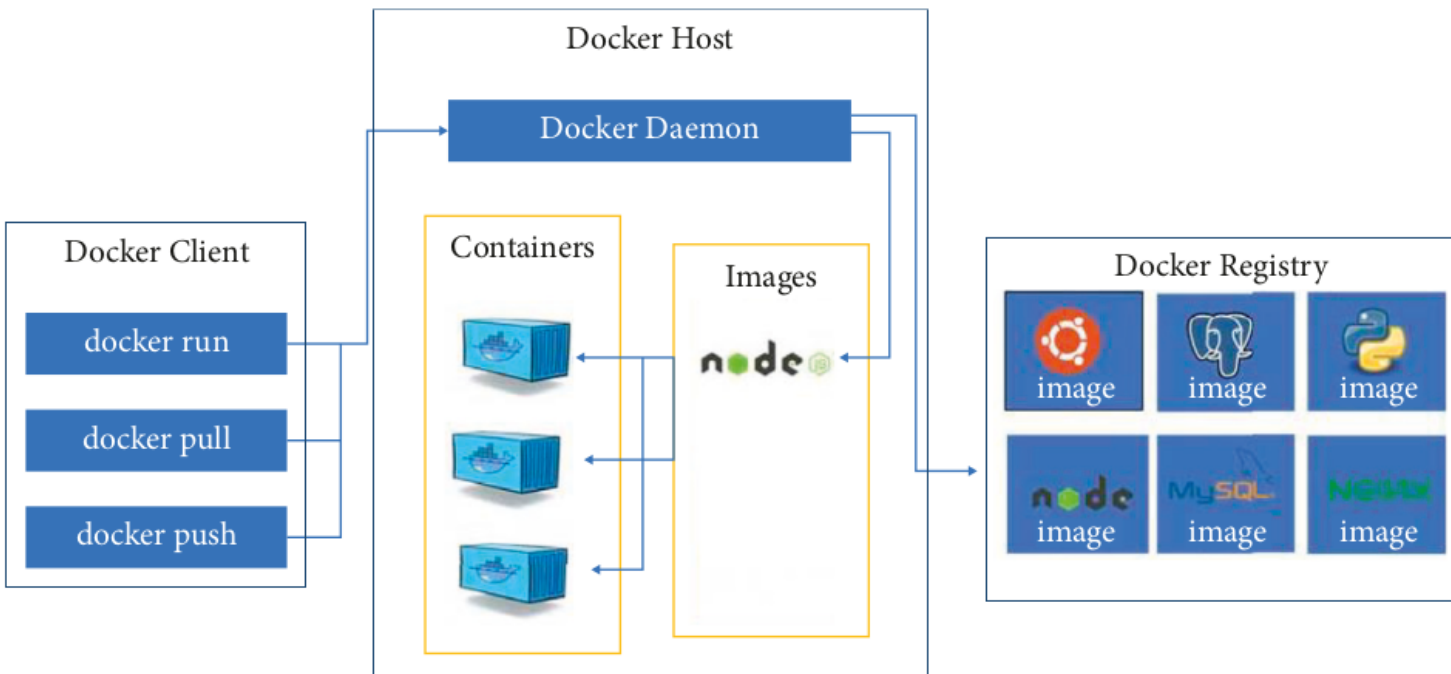

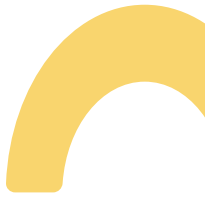


FIGURE 1: Docker architecture.





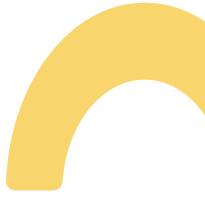
# CONTAINER IMAGE CREATION AND LIFECYCLE MANAGEMENT

- 
- 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.






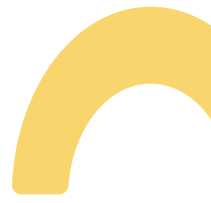
# CONTAINER IMAGE CREATION AND LIFECYCLE MANAGEMENT

- 
- Image Versioning:
    - Container images can have multiple versions to track changes and updates. Versioning allows for easy rollback to a previous version if issues arise.
    - It is important to maintain a versioning strategy and use appropriate tagging conventions to manage and track image versions effectively.
  - Image Registry:
    - Container images are stored in a registry, which acts as a centralized 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.
- 
- 




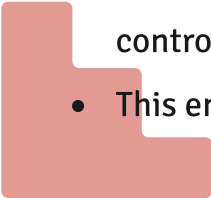
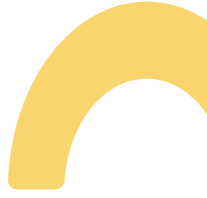
# CONTAINER IMAGE CREATION AND LIFECYCLE MANAGEMENT

- 
- 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.






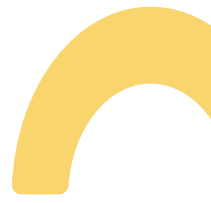
# CONTAINER IMAGE CREATION AND LIFECYCLE MANAGEMENT

- 
- Image 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.
  - Image Lifecycle Management:
    - Container images have a lifecycle that includes creation, distribution, deployment, and retirement.
    - It is important to have a well-defined process for managing image lifecycles, including version control, image promotion, and image deprecation.
    - This ensures that only trusted and up-to-date images are used in production environments.
- 
- 



# CONTAINER SECURITY CONCERNS

- 
- Vulnerable images - dependencies, supply chain attacks
  - Host kernel access - quickly escape isolation
  - Resource sharing - processes, filesystem, network
  - Limited isolation - namespaces, capabilities



# PROPOSED DOCKER-SEC SYSTEM

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

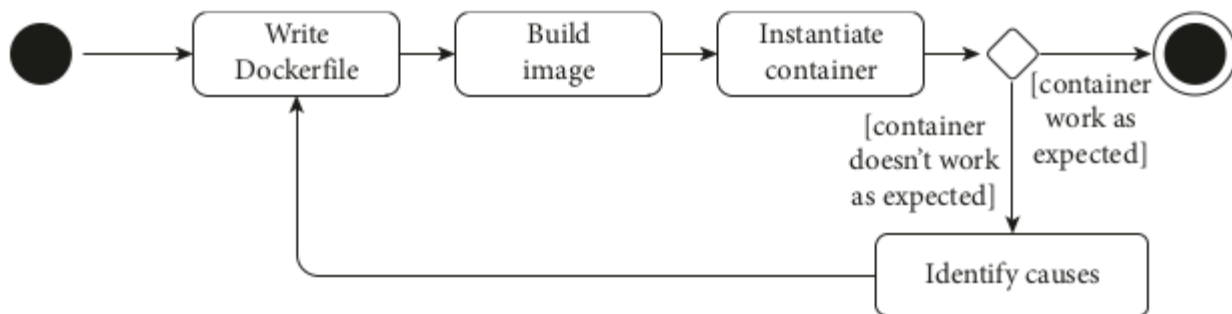


FIGURE 3: workflow when developing a Dockerfile.

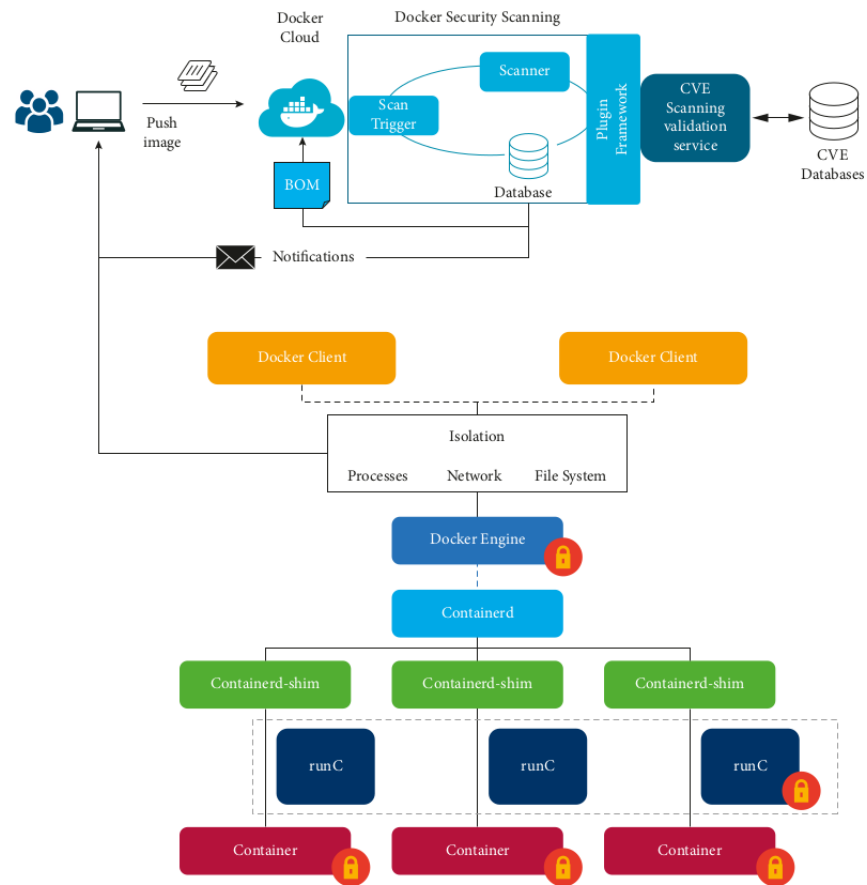


FIGURE 2: Proposed system architecture.



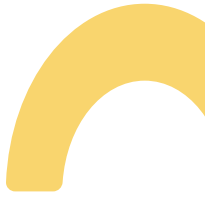


# DOCKER-SEC ARCHITECTURE

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



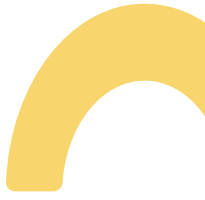


# CREATING CONTAINER PROFILES

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




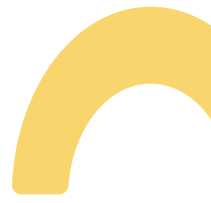
# CREATING APPARMOR PROFILES

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



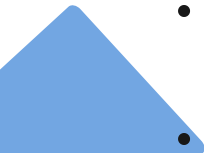

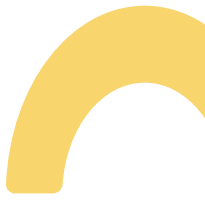
# STATIC ANALYSIS

- 
- 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 .





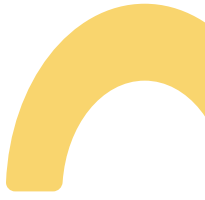


# DYNAMIC TESTING

- 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 .
- 
- 
- 

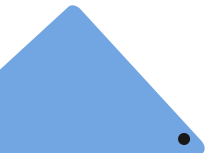

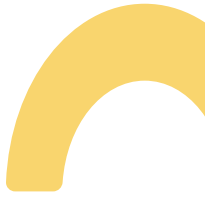


# CREATING APPARMOR PROFILES

- 
- Combination of static analysis and dynamic testing enables Docker-sec to create highly customized AppArmor profiles for each container.
  - The initial profiles are based on the container's settings, such as the mounted directories and files.
  - These profiles are then dynamically enhanced with rules extracted during the container's execution.
  - This approach ensures that the profiles accurately reflect the container's behavior and provide effective security measures .
  - Docker-sec enhances container security by restricting the container's access to resources and minimizing the attack surface.
  - The profiles define the allowed accesses and enforce them, preventing unauthorized actions and reducing the risk of container-based attacks .
- 
- 



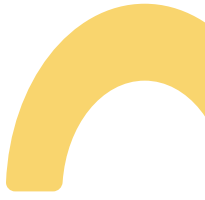


# BUILDING SECURED RUNC PROFILE

- 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 b571bf7ceb68b556dd37e8ae861ec7d05f0bf1c9e74180a236365074f68e14b
2022/06/01 20:19:25 [info] Analyzing 7ceb68b556dd37e8ae861ec7d05f0bf1c9e74180a236365074f68e14bb571bf
2022/06/01 20:19:25 [info] Analyzing a236365074f68e14bb571bf7ceb68b556dd37e8ae861ec7d05f0bf1c9e74180
2022/06/01 20:19:25 [info] Analyzing dd37e8ae861ec7d05fb571bf7ceb68b5560bf1c9e74180a236365074f68e14b
2022/06/01 20:19:25 [info] Analyzing dv37e8ae861f7ceb68b5560bf1c9e74180a236365074f68e14bec7d05fb571b
2022/06/01 20:19:25 [info] Analyzing 1c9e74180add37e8ae861ec7d05fb571bf7ceb68b5560bf236365074f68e14b
2022/06/01 20:19:25 [info] Analyzing f68b5560bf1dd37e8ae861ec7d05fb571bf7ceba236365074f68e19e741804bc
2022/06/01 20:19:25 [info] Analyzing 7d05fb571bdd37e8ae861ec7ceb68b5560bf1c9e74180a236365074f68e14b
2022/06/01 20:19:25 [info] Analyzing a236365074f68e14bdd37e8ae861ec7d05fb571bf7ceb68b5560bf1c9e74180
2022/06/01 20:19:25 [WARN] Image [java:latest] contains 37 total vulnerabilities
2022/06/01 20:19:25 [Error] 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. <a href="https://security-tracker.debian.org/tracker/CVE-2021-2021">https://security-tracker.debian.org/tracker/CVE-2021-2021</a>
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. <a href="https://security-tracker.debian.org/tracker/CVE-2021-2021">https://security-tracker.debian.org/tracker/CVE-2021-2021</a>
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. <a href="https://security-tracker.debian.org/tracker/TEMP-0780712-D00002">https://security-tracker.debian.org/tracker/TEMP-0780712-D00002</a>
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 <u>actually is</u> an access control bypass. <a href="https://security-tracker.debian.org/tracker/CVE-2022-1949">https://security-tracker.debian.org/tracker/CVE-2022-1949</a>

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 ttb571bf7cebfb68b556dd37e8agge861ec7d05f0bf1c97e74180a2936365074b
2022/06/11 21:29:35 [info] Analyzing 9cebsf38b556d376e8ae8gg61ecj7d05uf0bf1oc9e6745180a23i63650j571bf
2022/06/11 21:29:35 [info] Analyzing bcva23u6i36507h4f68e14bb571bf7cebfb68b556dd37e8ae861ec7d05f0bf1c9
2022/06/11 21:29:36 [info] Analyzing efgdtd377e8ae8h61ec7d05fb571bf7cebfb68b58560b9f1c9e704180a236365
2022/06/11 21:29:36 [info] Analyzing obv37e8ae861f7cebfb68b5560bf1c9e74180a236365074f68e14bec7dd05fb57
2022/06/11 21:29:36 [info] Analyzing 4fc9e74180add37e8ae861ec7d05fb571bf7cebfb68b5560bf236365074f68e16
2022/06/11 21:29:36 [info] Analyzing hkf68b5560bf1dd37e8ae861ec7d05fb571bf7ceba236365074f68e19e74108g
2022/06/11 21:29:37 [info] Analyzing ybn7d05fb571bdd37e8ae861ecf7cebfb68b5560bf1c9e74180a236365074f68k
2022/06/11 21:29:37 [info] Analyzing t36365074f68e14bdd37e8ae861ec7d05fb571bf7cebfb68b5560bf1c9e741g1o
2022/06/11 21:29:37 [WARN] Image [myapp:latest] contains 13 total vulnerabilities
2022/06/11 21:29:38 [Error] Image [myapp:latest] contains 13 unapproved vulnerabilities



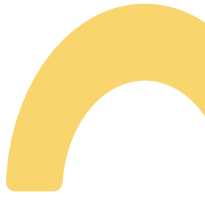
```

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. <a href="https://lists.denx.de/pipermail/u-boot/2022-June/486113.html">https://lists.denx.de/pipermail/u-boot/2022-June/486113.html</a>
Unapproved	Low CVE-2022-34911	wget	1.12.1+dfsg	XSS can occur in configurations that allow a JavaScript payload in a username. <a href="https://security-tracker.debian.org/tracker/CVE-2022-34911">https://security-tracker.debian.org/tracker/ CVE-2022-34911</a>

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

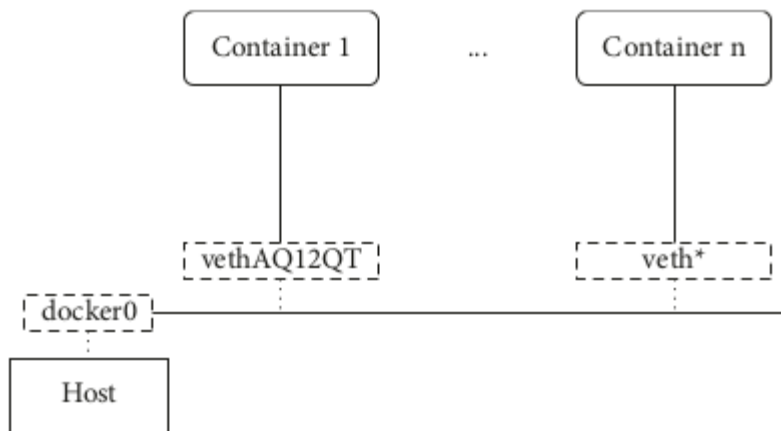


FIGURE 4: Docker's networking model.

# PERFORMANCE OVERHEAD

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



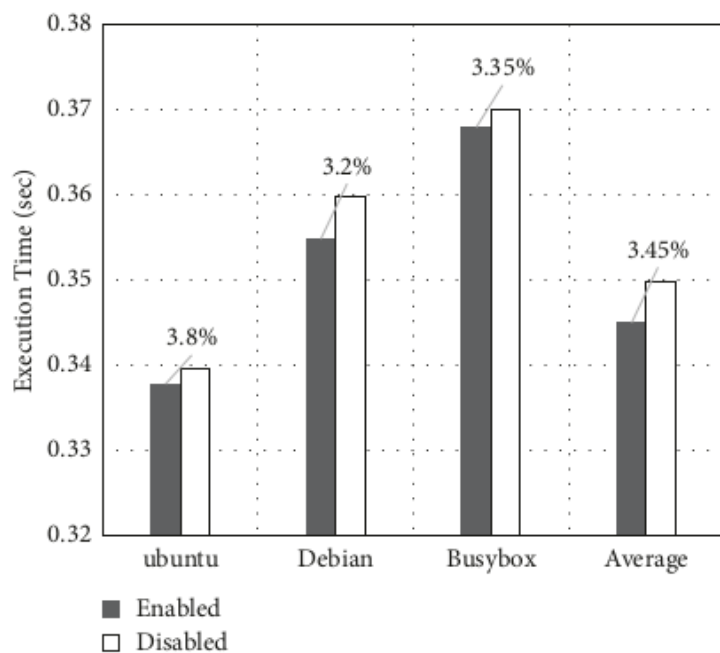


FIGURE 7: Docker-Sec's performance overhead.

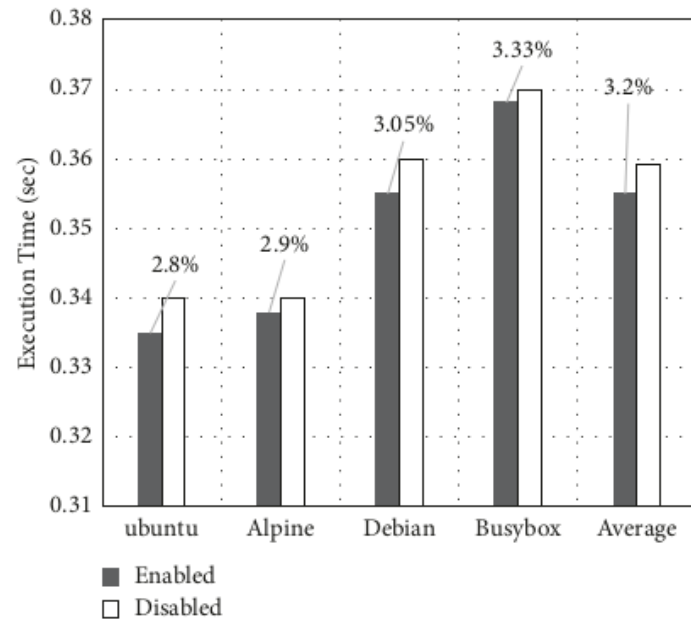
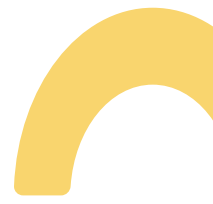


FIGURE 8: Docker-Sec's performance overhead with different images.

# CONTAINER ISOLATION

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

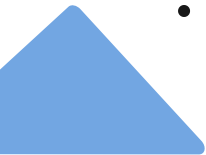

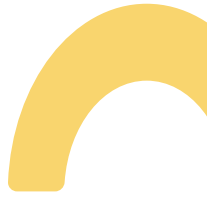


# USE CASES

- 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**

