Web Application Security

Dr. Digvijaysinh Rathod
Professor
School of Cyber Security and Digital Forensics
National Forensic Sciences University

digvijay.rathod@nfsu.ac.in

Docker containers security

Introduction

- ✓ Docker is a popular containerization platform that allows applications to run in isolated environments. While containers are lightweight and fast, they also introduce specific security risks. Understanding Docker security is essential for cloud computing, DevOps, and cyber security domains.

2. Why Docker Security Matters

- ✓ Containers share the host OS kernel, so a vulnerability in the kernel can affect all containers.
- ✓ Containers can run malicious or vulnerable images if not verified.
- ✓ If not configured properly, containers can escape isolation and affect the host.
  - ✓ Security is critical during:
  - ✓ Image creation
  - ✓ Deployment
  - ✓ Runtime execution
  - ✓ Orchestration (Kubernetes, Swarm)

a. Namespace Isolation

Docker uses Linux namespaces to isolate:

- Process IDs
- Network
- Mount points
- Inter-process communication

This prevents containers from seeing each other's processes.

b. cgroups (Control Groups)

- Limit CPU, RAM, disk I/O.
- Prevents resource exhaustion attacks (DoS).

c. Union File Systems

- Support layered images.
- Helps verify and control what changes happen at runtime.

3. Image Security

- Use trusted, official, or internal registries only.
- Avoid using random Docker Hub images.
- Scan images for vulnerabilities using:
  - Trivy, Clair, Anchore, Aqua.
- Use signed images (Docker Content Trust).
- Keep images minimal (e.g., Alpine Linux, Distroless).

4. Securing Container Runtime

- Do not run containers as root inside the container.
- Disable privileged mode (--privileged).
- Use a read-only filesystem when possible.
- Limit syscalls using seccomp, and apply AppArmor / SELinux profiles.
- Restrict access to host devices (--device).

5. Host System Security

- Keep host OS and kernel updated.
- Restrict access to Docker daemon (/var/run/docker.sock).
- Avoid exposing Docker API without TLS encryption.
- Run Docker in rootless mode when feasible.

6. Network Security

- Use user-defined networks to isolate containers.
- Disable inter-container communication (ICC) unless needed.
- Use network firewalls and iptables to restrict traffic.
- Use encrypted communication between nodes in Kubernetes or Docker Swarm.

7. Secrets and Configuration Security

- Never store passwords, API keys, or tokens inside images.
- Use:
  - Docker Secrets
  - HashiCorp Vault
  - AWS/GCP/Azure KMS
- Rotate secrets regularly.

8. Supply Chain Security

- Secure the CI/CD pipeline.
- Use automated tools to check Dockerfile best practices.
- Enable policy-as-code (OPA Gatekeeper, Kyverno).

9. Monitoring and Logging

- Centralize logs using:
  - ELK stack (Elasticsearch, Logstash, Kibana)
  - Fluentd
  - Grafana + Loki/Promtail
- Monitor runtime behavior via:
  - Falco (syscall-level monitoring)
  - Sysdig
  - Aqua Runtime Protection
- Detect anomalies like unusual syscalls, unexpected network traffic, or privilege escalation.

10. Orchestration-Level Security (Kubernetes)

- Enforce RBAC (Role-Based Access Control).
- Apply Network Policies for container communication.
- Avoid privileged containers.
- Use Pod Security Standards (PSS).
- Secure etcd database with TLS.

Summary

1. Secure the host (kernel, daemon, access control).
2. Secure the image (trusted sources, scanning, minimal base).
3. Secure the container at runtime (least privilege, isolation, rootless).
4. Secure networking (policies, segmentation, encryption).
5. Secure secrets (Vault, Docker Secrets).
6. Monitor continuously (Falco, ELK).
7. Secure orchestration (Kubernetes RBAC, policies).

Mobile Phone Security

Dr. Digvijaysinh Rathod
Professor
School of Cyber Security and Digital Forensics
National Forensic Sciences University

digvijay.rathod@nfsu.ac.in