Assignment - 1
Edge Computing - SEZG586

BITS Pilani
Pilani Campus

Submitted by

Satish Kumar Sharma (2022MT93327)

1

Q1. Please choose any one of the papers from 2 of the papers attached, and create document of the extension of the work that can be recommended.

Answer: I have gone through the paper1 and expanding upon it as follows.

Edge Computing Perspectives: Architectures, Technologies, and Open Security Issues

Expanding “VIRTUALIZATION-ORIGINATED SECURITY ISSUES”.

Security issues mentioned in this paper related to docker container are addressed below to mitigate the risks mentioned in the Table III

Docker container security

Docker container security: Docker container security is a critical aspect of managing containerized applications. Containers provide a lightweight and portable way to package and run applications, but it's important to implement security best practices to protect the underlying infrastructure and data. Here are some key considerations for Docker container security.

***Docker** containers are popular and have revolutionized application deployment. They are lightweight, portable, and easy to manage. But their security is of utmost importance. Your applications and data could be vulnerable to various threats without proper safeguards.*

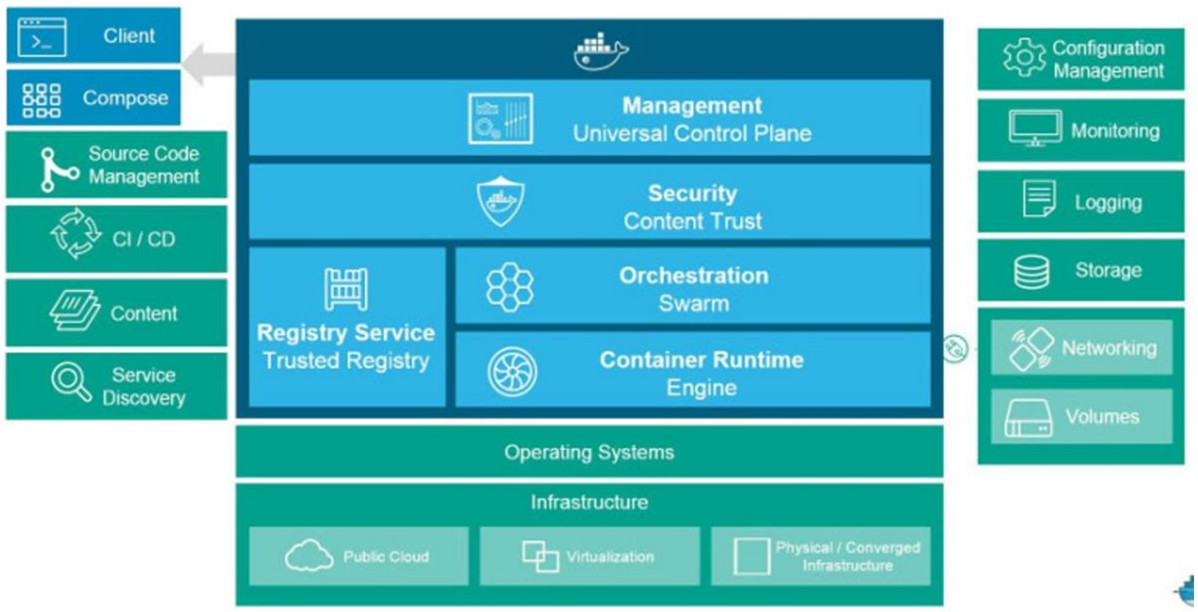


Fig: Docker Datacenter platform makes setting up a ready to run agile management system for your containers easier than ever. Docker

Building a secure image:

- **Base Official Images**: Using official Docker base images from trusted/official sources like Docker Hub provides a solid foundation. Official images undergo regular maintenance, reducing the risk of security vulnerabilities.
- **Embrace Minimal Base Image:** Opt for minimal base images to minimize the attack surface. Avoid full operating system images unless necessary, as they can increase the risk of potential vulnerabilities.
- **Implement Secure Image Building:** Build your image using the base image. The base image should be secure and should not contain any unnecessary software. Using multi-stage builds allows you to build your images in stages. This can help you to

reduce the size of your images and to remove unnecessary software. Follow security best practices when constructing Docker images.

- **Avoid root permissions:** By default, Docker containers run as root. This can be a security risk, as it allows any process running in the container full access to the host system. You should avoid running containers as root unless necessary. Instead, try creating a dedicated user for each container and run the container as that user.

Managing the secure image:

- **Limit resource usage:** Docker containers can consume a lot of resources, such as CPU, memory, and storage. They also protect against resource abuse and *denial-of-service* attacks by defining your containers. This ensures fair resource allocation and prevents resource exhaustion. You can do this by setting resource limits on the container when you create it.
- **Container Image Scanning:** Employ container image scanning tools to identify vulnerabilities and known security issues within your Docker images. This proactive approach enables you to detect and address potential risks before deploying containers. All the cloud providers give the option to scan your images. In AWS, the *AWS inspector* service can scan the image repositories. Also, there are several tools that you can use to scan Docker images for vulnerabilities.
  - Employ container image scanning tools to identify vulnerabilities and malware in your container images.
  - Integrate these tools into your CI/CD pipeline to automatically scan images before deployment.
- **Monitor your containers:** Container monitoring tools are crucial in maintaining containerized environments' health, performance, and security. This includes monitoring for unauthorized access, malicious activity, and resource usage anomalies. You can use various tools to monitor your containers for security threats. (*Prometheus, Grafana, Datadog, advisor, New Relic, Docker Stats, etc.*)
- **CrowdStrike Falcon**® Container Security is a comprehensive solution that can help you to secure your containerized environment. It is easy to use and deploy and offers a wide range of features to protect your containers from various threats. You can have the CS agent installed in your container.
- **Container Configuration:** Limit container capabilities by running them with the least privileges necessary. Use read-only file systems where possible. Avoid running containers as the root user.

Audits & Assessments (Where most of us miss this process):

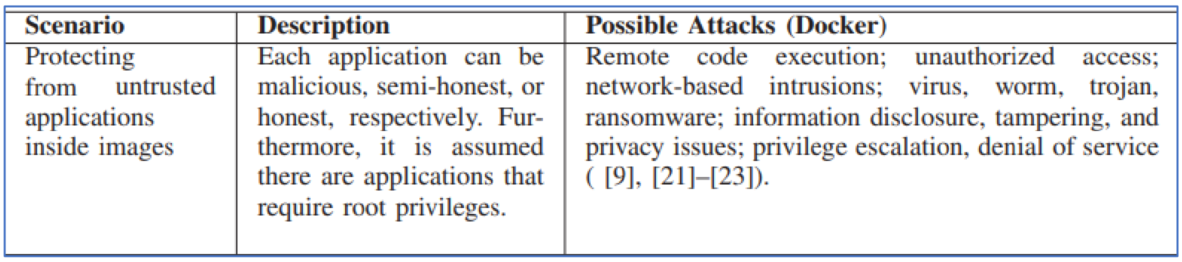
- **Keep your images updated:** Ensure you regularly update your Docker images. By doing so, your containers have the latest security patches and fixes, shielding them from known vulnerabilities.
- **Conduct quarterly Audits and Vulnerability Assessments**: Perform periodic security audits of your containers to identify vulnerabilities and misconfigurations. You maintain a strong security posture by following security best practices and conducting vulnerability assessments. Engage in a quarterly review with your cyber team to complete the audit.

Few Other Security best practices:

- **Network Security:** Use Docker's network features to securely isolate containers. Limit unnecessary network exposure by only exposing necessary ports. Implement network segmentation to control communication between containers.
- **Container Orchestration Security:** If using orchestration tools like Kubernetes, ensure proper security configurations. Regularly audit and monitor the orchestration environment for security vulnerabilities.
- **Runtime Security:** Employ runtime protection tools to monitor and detect suspicious activities within running containers. Utilize tools that can detect anomalies and potential security threats.
- **Secrets Management:** Avoid hardcoding sensitive information into your Dockerfiles. Use Docker's secrets management or external tools for securely handling sensitive data.
- **Logging and Auditing:** Implement logging to capture and analyze container activity. Regularly review logs for any suspicious or unauthorized activities.
- **Continuous Monitoring:** Set up continuous monitoring to detect and respond to security incidents promptly. Implement automated alerts for security events.
- **Regular Updates and Patching:** Keep Docker itself, as well as the host operating system, up to date with the latest security patches. Regularly update and patch the applications within your containers.
- **Education and Training:** Train your development and operations teams on container security best practices. Foster a security-aware culture within your organization.

By implementing these best practices, we can significantly enhance the security of the Docker containers and reduce the risk of security breaches. Additionally, always stay informed about the latest security developments and updates within the Docker and container ecosystem.

From the scenarios mentioned in Table III, below scenario and mitigation of possible attacks are listed.



Scenario	Description	Possible Attacks (Docker)
Protecting from untrusted applications inside images	Each application can be malicious, semi-honest, or honest, respectively. Furthermore, it is assumed there are applications that require root privileges.	Remote code execution; unauthorized access; network-based intrusions; virus, worm, trojan, ransomware; information disclosure, tampering, and privacy issues; privilege escalation, denial of service ([9], [21]–[23]).

Remote Code Execution (RCE) in a Docker container refers to the ability for an attacker to execute arbitrary code on a target system remotely. This is a serious security vulnerability that can have severe consequences, allowing unauthorized individuals to compromise the integrity and confidentiality of your containerized applications and, by extension, the host system.

Here are some measures you can take to mitigate the risk of Docker container RCE:

- **Update Docker and Base Images:** Regularly update Docker to the latest version to benefit from security improvements. Use up-to-date and patched base images to reduce the risk of known vulnerabilities.
- **Image Scanning:** Implement container image scanning tools to identify vulnerabilities in your images before deployment. Integrate scanning into your CI/CD pipeline to catch issues early in the development process.
- **Least Privilege Principle:** Run containers with the principle of least privilege. Avoid running containers as the root user whenever possible. Restrict container capabilities to the minimum required for the application to function.
- **Network Security:** Securely configure network settings for containers. Limit exposure by only exposing necessary ports. Consider using network segmentation to control communication between containers.
- **Container Isolation:** Use container orchestration tools (e.g., Kubernetes, Docker Compose) to manage and isolate containers.
- **Seccomp Profiles:** Use Seccomp profiles to restrict the system calls that a container can make.
- **Monitor Container Activity:** Set up monitoring for container activities and be alert for any suspicious behavior. Monitor logs for signs of unauthorized access or unusual activities.
- **Authentication and Authorization:** Strengthen authentication mechanisms for accessing Docker daemons. Regularly review and update access controls and permissions.
- **Firewalls and Intrusion Detection Systems:** Use firewalls to control traffic to and from containers. Implement intrusion detection and prevention systems to identify and block malicious activities.
- **Regular Security Audits:** Conduct regular security audits of your Docker setup and containerized applications. Perform penetration testing to identify and address potential vulnerabilities.
- **Educate Teams:** Educate development and operations teams on secure coding practices. Foster a culture of security awareness to reduce the risk of unintentional security vulnerabilities.

By following these best practices, we can significantly reduce the risk of remote code execution vulnerabilities in Docker containers. Regularly staying informed about security updates, industry best practices, and evolving threats is crucial to maintaining a robust container security posture.

- **Secure Docker Daemon:** Restrict access to the Docker daemon. Only trusted users should have permission to interact with Docker. Use strong authentication mechanisms for accessing the Docker daemon, such as TLS certificates or token-based authentication.
- **Least Privilege Principle:** Avoid running containers as the root user. Run containers with the principle of least privilege, using non-privileged users whenever possible. Limit container capabilities to reduce the potential impact of a security breach.

- **Network Security:** Securely configure network settings for containers. Limit exposure by only exposing necessary ports. Use firewalls and network segmentation to control traffic to and from containers.

Unauthorized access to Docker containers is a serious security concern that can lead to various issues, including data breaches, service disruption, and compromise of the host system.

Here are some measures to help mitigate the risk of unauthorized access to Docker containers:

- **Container Isolation:** Utilize container orchestration tools (e.g., Kubernetes, Docker Compose) to manage and isolate containers.
- **Access Control:** Regularly review and update access controls. Ensure that only authorized users have access to Docker-related resources. Utilize Role-Based Access Control (RBAC) mechanisms if available.
- **Monitor Container Activity:** Set up monitoring for container activities and be alert for any unusual behavior. Monitor logs for signs of unauthorized access or suspicious activities.
- **Authentication and Authorization:** Strengthen authentication mechanisms for accessing Docker and containerized applications. Use strong, unique passwords and consider multi-factor authentication where applicable.
- **Regular Security Audits:** Conduct regular security audits of your Docker setup and containerized applications. Perform penetration testing to identify and address potential vulnerabilities. **Update Docker and Base Images:** Keep Docker up to date to benefit from the latest security patches. Use up-to-date and patched base images to reduce the risk of known vulnerabilities.
- **Image Scanning:** Implement container image scanning tools to identify vulnerabilities in your images before deployment. Integrate scanning into your CI/CD pipeline to catch issues early in the development process.
- **Educate Teams:** Educate development and operations teams on secure coding practices. Foster a culture of security awareness to reduce the risk of unintentional security vulnerabilities.

By following these best practices, we can significantly reduce the risk of unauthorized access to Docker containers. Security is a continuous process, and it's important to stay informed about the latest security developments, apply updates promptly, and adapt your security measures to address evolving threats.

Network-based intrusions targeting Docker containers can pose significant security risks. To mitigate these risks and enhance the security of your containerized environment,

Consider the following best practices:

- **Secure Docker Daemon:** Restrict access to the Docker daemon. Only trusted users should have permission to interact with Docker. Enable TLS for securing communication between Docker clients and the daemon.

- **Network Segmentation:** Utilize Docker's network features to segment and isolate containers. Employ different network bridges for different sets of containers based on their roles and sensitivity.
- **Firewalls and Network Policies:** Implement firewalls to control incoming and outgoing traffic to and from containers. Use network policies (if using Kubernetes) to define and enforce communication rules between containers.
- **Exposed Ports:** Minimize the number of exposed ports on containers. Only expose ports that are necessary for the application to function. Regularly audit and review the necessity of exposed ports.
- **Container Orchestration Security:** If using container orchestration tools like Kubernetes, ensure that the orchestration layer is secure. Implement network policies to control communication between pods in Kubernetes.
- **Container Isolation:** Utilize technologies like gVisor or Kata Containers to add an additional layer of isolation between containers and the host system.
- **Network Monitoring:** Implement network monitoring tools to detect and alert on unusual or malicious network activities.

Regularly review logs and network traffic for signs of suspicious behavior.

- **Regular Security Audits:** Conduct regular security audits of your Docker setup, including network configurations. Perform penetration testing to identify and address potential vulnerabilities.
- **Update Docker and Base Images:** Keep Docker up to date to benefit from the latest security patches. Use up-to-date and patched base images to reduce the risk of known vulnerabilities.
- **Image Scanning:** Implement container image scanning tools to identify vulnerabilities in your images before deployment. Integrate scanning into your CI/CD pipeline to catch issues early in the development
- process.
- **Authentication and Authorization:** Strengthen authentication mechanisms for accessing Docker and containerized applications. Regularly review and update access controls and permissions.
- **Educate Teams:** Educate development and operations teams on secure coding practices and the importance of network security. Foster a culture of security awareness to reduce the risk of unintentional security vulnerabilities.

By implementing these best practices, we can enhance the network security of your Docker containers and reduce the risk of network-based intrusions. Regularly assess and update your security measures to adapt to evolving threats and changes in your containerized environment.

- **Image Security:** Use official and trusted base images from reputable sources. Regularly scan container images for vulnerabilities, malware, and known security issues before deploying them.
- **Image Signing and Verification:** Implement Docker Content Trust to sign and verify the integrity of container images. Ensure that only signed images are used in your environment. **Update and Patching:** Keep both Docker and the underlying host

system up to date with the latest security patches. Regularly update and patch the applications within your containers.
- **Least Privilege Principle:** Run containers with the principle of least privilege. Avoid running containers as the root user. Limit container capabilities and permissions to reduce the potential impact of security breaches.
- **Container Isolation:** Utilize container orchestration tools to manage and isolate containers. Consider using additional isolation technologies like gVisor or Kata Containers.
- **Network Security:** Securely configure network settings for containers. Limit exposure by only exposing necessary ports.

Use firewalls and network segmentation to control traffic to and from containers. Mitigating the risks associated with various types of malware (viruses, worms, trojans, ransomware) and addressing information disclosure, tampering, and privacy issues in Docker containers requires a multi-faceted approach.

Here are some best practices to enhance the security of your Docker containers:

- **Access Control:** Implement strong access controls. Regularly review and update user access permissions. Utilize Role-Based Access Control (RBAC) mechanisms if available.
- **Runtime Protection:** Use runtime protection tools to monitor and detect suspicious activities within running containers. Implement anomaly detection and behavioral analysis.
- **Regular Monitoring and Logging:** Set up monitoring for container activities and network traffic. Regularly review logs for signs of unauthorized access or malicious activities.
- **Data Encryption:** Encrypt sensitive data within containers and during communication between containers. Use secure communication protocols (TLS/SSL) for data in transit.
- **Secrets Management:** Avoid hardcoding sensitive information into Dockerfiles. Use Docker's secrets management or external tools for securely handling sensitive data.
- **Backup and Recovery:** Regularly back up containerized applications and their data. Establish a recovery plan to quickly restore services in the event of a security incident.
- **Security Audits and Testing:** Conduct regular security audits and penetration testing on your Docker environment. Test your containerized applications for security vulnerabilities.
- **Employee Training:** Educate development and operations teams on secure coding practices and the importance of security. Foster a security-aware culture within your organization.
- **Regulatory Compliance:** Ensure compliance with relevant privacy and security regulations based on your industry and region.

Implementing a combination of these security practices can significantly reduce the risks associated with malware, information disclosure, tampering, and privacy issues in Docker containers. Security is an ongoing process, and it's essential to stay informed about the latest security developments and adapt your security measures accordingly.

Mitigating the risks of privilege escalation and denial of service (DoS) attacks in Docker containers involves implementing security best practices and staying vigilant against potential vulnerabilities.

Here are some recommendations to enhance the security of your Docker containers in these areas:

Privilege Escalation:

- **Run as Non-Root:** Avoid running containers as the root user whenever possible. Use a non-privileged user to reduce the impact of potential security breaches.
- **Least Privilege Principle:** Limit container capabilities by restricting the set of Linux capabilities available to the container. Only grant the specific capabilities necessary for the application to function.
- **Seccomp Profiles:** Use Seccomp profiles to restrict the system calls that a container can make.
- Implement custom Seccomp profiles based on the principle of least privilege.
- **Isolation Technologies:** Consider using additional isolation technologies like gVisor or Kata Containers to provide an extra layer of security and prevent privilege escalation.
- **Image Scanning:** Regularly scan container images for vulnerabilities, including those related to privilege escalation. Integrate image scanning into your CI/CD pipeline to catch issues early.
- **Regular Security Audits:** Conduct regular security audits of your Docker setup, including privilege settings. Perform penetration testing to identify and address potential vulnerabilities.

Denial of Service (DoS):

- **Resource Limits:** Set resource limits for containers to prevent resource exhaustion and potential DoS attacks. Use Docker Compose or Kubernetes resource specifications to define CPU and memory limits.
- **Network Rate Limiting:** Implement network rate limiting to control the amount of traffic a container can send and receive. Use firewalls and network policies to restrict excessive traffic.
- **Container Restart Policies:** Define appropriate restart policies for containers to automatically restart them in case of failures. Adjust restart policies based on your application's requirements and resilience needs.
- **Load Balancing:** Distribute traffic across multiple containers using load balancing to prevent a single container from becoming a bottleneck.
- **Container Orchestration Tools:** Utilize container orchestration tools like Kubernetes to manage and distribute containers across the cluster, improving resilience and mitigating DoS risks.
- **Rate Limiting and Throttling:** Implement rate limiting and throttling mechanisms within your applications to control the rate of incoming requests.

- **Monitoring and Alerting:** Set up monitoring for resource usage and network traffic. Implement alerting to notify you of abnormal behavior indicative of a potential DoS attack.
- **Distributed Architecture:** Design your application to be distributed across multiple containers and nodes to reduce the impact of a DoS attack on a single instance.

By implementing these best practices, we can enhance the security of your Docker containers and reduce the risk of privilege escalation and denial of service attacks. Regularly review and update your security measures to adapt to evolving threats and changes in your containerized environment.

Conclusions

Docker containers are, by default, quite secure; especially if we run our processes as non-privileged users inside the container. We can add an extra layer of safety by enabling AppArmor, SELinux, GRSEC, or another appropriate hardening system. If we think of ways to make docker more secure, we welcome feature requests, pull requests, or comments on the Docker community forums.

As with most security problems, there is no silver bullet with container security. The technical, operational, and organizational moving pieces that go into protecting the company's container images often reside at the boundaries between teams, functions, and responsibilities. This adds complexity to an already complex problem. Rather than further adding to the burdens created by this complexity, you should look for tools that enable your teams to work together and reach a deeper understanding of where goals, risks, and priorities overlap and coexist.

References:

https://pachai-devaraj.medium.com/docker-container-security-simple-best-practices-aa1107df2331

https://www.docker.com/blog/container-security-and-why-it-matters/

https://cheatsheetseries.owasp.org/cheatsheets/Docker_Security_Cheat_Sheet.html

https://info.wiz.io/container-security.html?utm_source=google&utm_medium=ppc&utm_campaign=non-brand-solutions-search-us-ca&utm_term=container%20security%20software&utm_content={adgroup}&utm_device=c&gclid=EAIaIQobChMI2ZXdqtvNggMVjf3ICh0NwxcoEAAYASAAEgITmPD_BwE

Thank You.