Threat Mitigation in Containerized Environments

Budi Pranoto

Department of Computer Science, Universitas Gadjah Mada

Abstract

Containerized environments have revolutionized application development and deployment, offering unmatched flexibility, scalability, and consistency across diverse infrastructures. However, these environments also introduce unique security challenges, stemming from the shared nature of resources, the immutability of container images, and the complexities of container orchestration platforms like Docker and Kubernetes. This paper provides an in-depth exploration of the threat landscape within containerized environments, focusing on key areas such as container escape, image vulnerabilities, network attacks, and supply chain risks. We also discuss robust mitigation strategies, including the use of hardened images, network segmentation, and container-specific security tools. The analysis culminates in a set of best practices aimed at securing containerized environments against evolving threats.

Keywords: Container Security, Threat Mitigation, Containerized Environments, Docker Security, Kubernetes Security, Security Best Practices, Image Vulnerabilities, Network Segmentation, Container Escape, Secure Containers, Supply Chain Security

Introduction

The rapid adoption of containerized environments has transformed the landscape of software development and deployment. Technologies like Docker, Kubernetes, and other container orchestration tools have enabled developers to build, test, and deploy applications in a consistent manner, regardless of the underlying infrastructure. Containers encapsulate an application and its dependencies, ensuring that it runs identically in development, testing, and production environments. This capability has led to widespread adoption, with organizations of all sizes leveraging container technology to improve efficiency, reduce costs, and accelerate delivery timelines.

However, the benecomes of containerization come with significant security challenges. Containers, by design, share the underlying host operating system, which can lead to security vulnerabilities if not properly managed. Unlike traditional virtual machines, where each instance is isolated with its own OS, containers are more lightweight but also more intertwined with the host system. This shared architecture introduces risks such as container escape, where an attacker can potentially gain access to the host system and other containers running on it.

Moreover, the ease of pulling images from public repositories, while convenient, can lead to the deployment of unverified or vulnerable software components. Attackers often exploit these vulnerabilities to gain unauthorized access, inject malicious code, or launch attacks on other containers and network segments. The dynamic and often ephemeral nature of containers further complicates security monitoring and incident response.

This paper aims to provide a comprehensive examination of the security threats inherent in containerized environments and to propose effective strategies for mitigating these risks. We begin by identifying and analyzing common threat vectors, including container escape, image vulnerabilities, network attacks, and supply chain risks. Next, we explore various mitigation strategies, emphasizing the importance of robust security policies, the use of hardened images, and the implementation of network segmentation. Finally, we present a set of best practices for secure container management, offering practical guidance for organizations looking to safeguard their containerized environments.

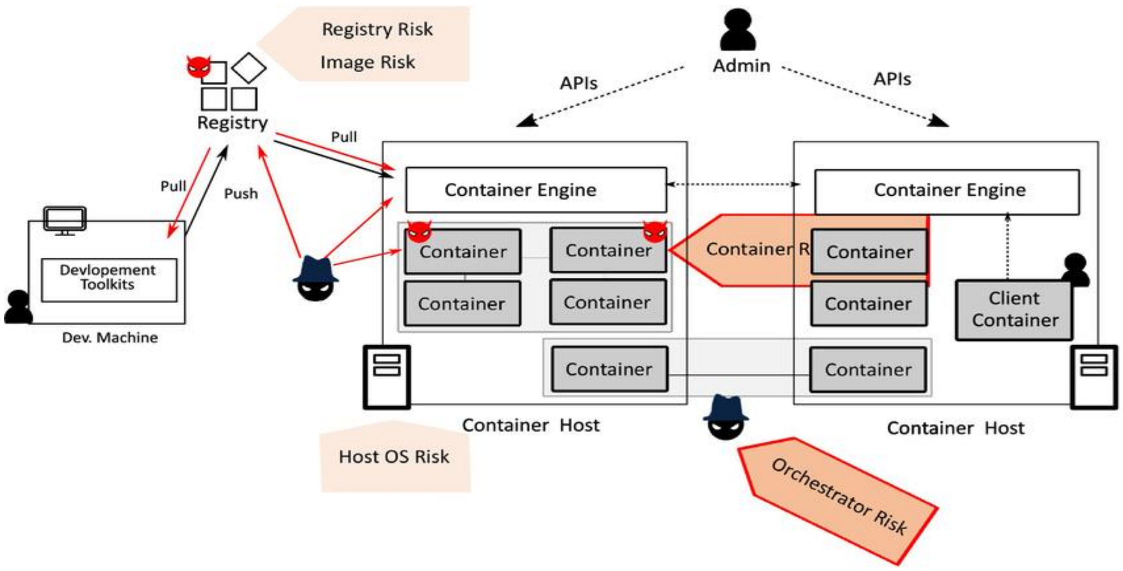
Common Threat Vectors in Containerized Environments

Containerized environments, while offering significant advantages in terms of deployment efficiency and scalability, also present a unique set of security challenges. Understanding these challenges is critical for effectively mitigating the risks associated with containerization. Below, we explore some of the most common threat vectors in containerized environments, including container escape, image vulnerabilities, network attacks, insecure configurations, and supply chain attacks. [1]

1. Container Escape

Container escape represents one of the most severe security risks in containerized environments. In a typical setup, containers are designed to be isolated from each other and from the host system. This isolation is achieved through the use of namespaces, cgroups, and other Linux kernel features. However, vulnerabilities in the container runtime, misconfigurations, or flaws in the underlying operating system can lead to a scenario where a process within a container breaks out of its isolated environment and gains access to the host system.

Once an attacker has successfully executed a container escape, they can potentially access sensitive data on the host, manipulate other containers, or escalate their privileges to gain complete control over the host system. This type of attack can be particularly devastating in multi-tenant environments, where multiple containers belonging to different users or organizations share the same host. [2]



To illustrate the severity of this threat, consider the case of the "Dirty Cow" vulnerability (CVE-2016-5195), a privilege escalation bug in the Linux kernel. This vulnerability, if exploited within a container, could allow an attacker to gain root access on the host machine. Similarly, vulnerabilities in container runtimes like runc (CVE-2019-5736) have demonstrated the potential for container escape attacks.

2. *Image Vulnerabilities*

Container images are the building blocks of containerized environments, encapsulating the application code, dependencies, and system libraries required to run an application. While this approach simplifies deployment, it also introduces significant security risks. Container images are often sourced from public repositories like Docker Hub, where they are created and shared by a global community of developers. However, these images may contain vulnerabilities, either in the base operating system or in the software packages they include.

Vulnerabilities in container images can arise from several sources. First, outdated packages within an image may contain known security flaws that have been patched in newer versions. If these vulnerabilities are not addressed before deployment, they can be exploited by attackers to gain unauthorized access or execute malicious code. Second, some images may be intentionally malicious, with embedded backdoors or malware designed to compromise systems where they are deployed.

The risk of image vulnerabilities is compounded by the practice of "layering," where a new image is built on top of an existing base image. If the base image contains vulnerabilities, those vulnerabilities are inherited by all derived images, creating a cascading effect that can impact an entire containerized environment. This risk is further exacerbated by the fact that many organizations lack visibility into the components and dependencies included in the images they use, making it difficult to assess and mitigate potential security risks.

3. Network Attacks

Network security is a critical aspect of containerized environments, particularly as containers often communicate over networks to perform their functions. Containers within the same host or across different hosts in a cluster may need to interact with each other, access external services, or expose APIs to the outside world. This networked nature introduces a variety of security challenges, including man-in-the-middle (MITM) attacks, denial-of-service (DoS) attacks, and unauthorized access to network resources.

In a containerized environment, network traffic is typically managed by the container orchestration platform, such as Kubernetes. However, misconfigurations in network policies, inadequate encryption of network traffic, and the exposure of unnecessary network ports can create opportunities for attackers to intercept or manipulate communications. For example, an attacker who gains access to a container with inadequate network segmentation could launch a MITM attack, capturing sensitive data or injecting malicious traffic into the network.

Another common network-related threat in containerized environments is the exploitation of insecure service discovery mechanisms. In Kubernetes, for example, services are often exposed via ClusterIP, NodePort, or LoadBalancer, which if not properly secured, can be exploited by attackers to gain unauthorized access to services or to launch attacks on other parts of the network. [3]

4. Insecure Configurations

Misconfigurations are a prevalent source of vulnerabilities in containerized environments. Despite the robust security features offered by container technologies, improper configuration can negate these protections and expose containers to a range of security risks. Common misconfigurations include running containers with root privileges, exposing unnecessary network ports, and failing to apply proper access controls to sensitive resources.

Running containers as root is a particularly dangerous practice, as it grants the container full access to the underlying host system. In the event of a container compromise, an attacker could exploit this elevated access to gain control over the host, escalate privileges, or move laterally to other containers. To mitigate this risk, it is essential to run containers with the least privilege necessary to perform their functions, and to avoid using the root user whenever possible.

Exposing unnecessary network ports is another common misconfiguration that can lead to security vulnerabilities. Containers may expose services on network ports to enable communication with other containers or external clients. However, if these ports are not properly secured, they can become entry points for attackers seeking to exploit vulnerabilities in the exposed services. Implementing strict network policies and limiting the exposure of ports to only those required for the application to function can help mitigate this risk.

5. Supply Chain Attacks

The container supply chain encompasses the entire lifecycle of a container, from development to deployment and beyond. This lifecycle involves multiple stages, including the creation of container images, the distribution of those images through registries, and the deployment of containers in production environments. Each stage presents opportunities for attackers to introduce malicious code, exploit vulnerabilities, or compromise the integrity of the containerized environment.

Supply chain attacks can take many forms. For example, an attacker may compromise a public image repository and inject malicious code into popular images. When these images are pulled and deployed by unsuspecting users, the malicious code is executed, potentially leading to data breaches, unauthorized access, or other forms of compromise. Similarly, attackers may target the build process itself, injecting malicious code into the software before it is packaged into a container image.

Another significant risk in the container supply chain is the use of unverified or untrusted third-party dependencies. Containers often include a wide range of open-source libraries and components, many of which may have their own vulnerabilities or be subject to compromise. If these dependencies are not properly vetted, they can introduce significant security risks into the containerized environment.

Mitigation Strategies

Effective threat mitigation in containerized environments requires a multifaceted approach that addresses the various risks and vulnerabilities inherent in these environments. The following sections outline key mitigation strategies that organizations can implement to enhance the security of their containerized deployments.

1. Implementing Robust Security Policies

Security policies are the foundation of any effective container security strategy. These policies should govern the behavior of containers, define access controls, and establish guidelines for the secure management of containerized environments. Implementing robust security policies requires a comprehensive understanding of the threats facing containerized environments and the ability to enforce these policies consistently across the organization.

One critical aspect of security policy implementation is the principle of least privilege. This principle dictates that containers should only have access to the resources they need to perform their functions and nothing more. By limiting access in this way, organizations can reduce the potential impact of a security breach. For example, containers should be configured to run as non-root users wherever possible, and access to sensitive system resources should be tightly controlled. [4]

Another important security policy is the enforcement of network segmentation. Network segmentation involves dividing the containerized environment into isolated segments, each with its own security policies and access controls. This approach limits the potential impact of a security breach by preventing an attacker from moving laterally across the network. In practice, network segmentation can be implemented using container orchestration tools like Kubernetes, which allows administrators to define network policies that control traffic between containers and between containers and external services.

Security policies should also address the use of container images.

Organizations should establish guidelines for the creation, use, and management of container images, including requirements for image scanning, signing, and verification. By ensuring that only trusted and verified images are used in production environments, organizations can reduce the risk of deploying vulnerable or malicious software.

2. Using Hardened Images

The use of hardened images is a critical strategy for mitigating the risks associated with container image vulnerabilities. Hardened images are specifically designed to minimize the attack surface by removing unnecessary components, applying security patches, and configuring software with secure defaults. By using hardened images, organizations can significantly reduce the likelihood of vulnerabilities being exploited within their containerized environments.

Hardened images are typically based on minimalistic base images that include only the essential components required to run the application. This approach reduces the number of potential vulnerabilities by eliminating unnecessary software and libraries that could be exploited by attackers. Additionally, hardened images are regularly updated to include the latest security patches, ensuring that known vulnerabilities are addressed promptly.

To further enhance security, organizations should consider using container image signing and verification mechanisms. Image signing involves creating a cryptographic signature for a container image, which can be used to verify the integrity and authenticity of the image before it is deployed. By verifying image signatures, organizations can ensure that only trusted images are used in their environments, reducing the risk of deploying compromised or tampered images.

Table 1: Comparison of Base Images and Hardened Images

Base Image	Hardened Image
General-purpose	Security-focused
May contain unused packages	Minimalistic, essential packages only
Potentially outdated	Regularly updated
More attack surface	Reduced attack surface

In addition to using hardened images, organizations should implement regular image scanning as part of their security processes. Image scanning tools, such as Clair, Trivy, or Aqua Security, can automatically detect vulnerabilities in container images before they are deployed. These tools scan images for known vulnerabilities in operating system packages, application dependencies, and other components, providing administrators with detailed reports that highlight potential security risks. [5]

3. Employing Network Segmentation

Network segmentation is a powerful technique for mitigating the risks associated with network attacks in containerized environments. By dividing the network into isolated segments, organizations can limit the potential impact of a security breach and prevent attackers from easily moving laterally across the network. Network segmentation can be achieved through a combination of container orchestration features, network policies, and security tools.

In a Kubernetes environment, network segmentation is typically implemented using network policies. Kubernetes network policies allow administrators to define rules that control the flow of traffic between pods, services, and external endpoints. For example, a network policy might restrict access to a database pod, allowing only specific application pods to communicate with it. This type of segmentation helps to ensure that even if an attacker compromises one container, they cannot easily access other containers or services.

Table 2: Example Kubernetes Network Policy

Policy Name	Description
Allow-DB-Access	Allows traffic from application pods to database pods
Deny-All-External	Denies all external traffic to sensitive internal services
Restrict-API-Access	Restricts access to API endpoints to specific trusted IP addresses
Allow-Internal-Comm	Allows internal communication between specific application pods

In addition to Kubernetes network policies, organizations can enhance network segmentation by implementing virtual networks, firewalls, and security groups. Virtual networks, such as those provided by cloud platforms like AWS VPC or Azure Virtual Network, allow organizations to create isolated network segments with their own routing and access controls. Firewalls and security groups can be used to enforce additional restrictions on traffic between network segments, ensuring that only authorized communications are allowed.

Network segmentation is not limited to internal traffic within the containerized environment. Organizations should also consider the security of external network connections, such as those between the containerized environment and external services or clients. Encrypting network traffic using protocols like TLS can help protect sensitive data in transit and prevent attackers from intercepting or tampering with communications.

4. Regular Security Audits

Regular security audits are essential for maintaining the security of containerized environments. Security audits provide an opportunity to identify and address vulnerabilities, misconfigurations, and compliance issues before they can be exploited by attackers. A comprehensive security audit should include vulnerability scanning, configuration reviews, access control assessments, and network security assessments.

Table 3: Security Audit Checklist

Audit Component	Description
Vulnerability Scanning	Automated tools to identify known vulnerabilities
Configuration Review	Checking for insecure configurations
Access Control Assessment	Ensuring proper access controls are in place
Network Security Assessment	Analyzing network configurations and traffic
Compliance Review	Ensuring adherence to security policies and regulatory requirements
Incident Response Plan Review	Evaluating the effectiveness of the incident response plan

Vulnerability scanning is a critical component of the security audit process. Automated scanning tools can quickly identify known vulnerabilities in container images, host systems, and network configurations. These tools provide detailed reports that highlight potential security risks and recommend remediation actions. Regular vulnerability scanning helps ensure that security patches are applied promptly and that vulnerabilities are addressed before they can be exploited.

Configuration reviews are another important aspect of security audits. Misconfigurations are a common source of security vulnerabilities, and regular reviews can help identify and correct these issues. Configuration reviews should focus on critical areas such as access controls, network settings, and container runtime configurations. By identifying and addressing misconfigurations, organizations can reduce the risk of security breaches and ensure that their containerized environments are properly secured.

Access control assessments are also crucial for maintaining security in containerized environments. These assessments should evaluate the effectiveness of access controls at both the container and host levels, ensuring that only authorized users and processes have access to sensitive resources. Access control assessments should also consider the principle of least privilege, ensuring that containers and users have only the minimum level of access required to perform their functions.

Network security assessments are essential for identifying potential weaknesses in the network infrastructure of a containerized environment. These assessments should evaluate network segmentation, traffic encryption, and firewall configurations, ensuring that the network is properly secured against attacks. Network security assessments should also include a review of external network connections, such as those between the containerized environment and external services, to ensure that sensitive data is protected in transit.

Finally, security audits should include a review of the organization's incident response plan. An effective incident response plan is critical for responding to security breaches and minimizing their impact. The incident response plan review should evaluate the organization's ability to detect, respond to, and recover from security incidents, and should include recommendations for improving the plan based on the findings of the security audit.

5. Using Container-Specific Security Tools

The use of container-specific security tools is essential for effectively mitigating threats in containerized environments. These tools are designed to address the unique security challenges of containers, providing features such as vulnerability scanning, runtime protection, and security monitoring. [6]

Docker Bench for Security is one of the most widely used tools for assessing the security of Docker containers. This tool provides a comprehensive security audit based on Docker's security best practices, checking for issues such as insecure container configurations, outdated software, and unnecessary privileges. Docker Bench for Security generates a detailed report that highlights potential security risks and provides recommendations for remediation. [7]

Another important tool is Clair, an open-source vulnerability scanner for container images. Clair integrates with container registries to automatically scan images for known vulnerabilities, providing detailed reports that highlight potential security risks. Clair supports a wide range of container image formats, including Docker images, and can be integrated with CI/CD pipelines to ensure that images are scanned for vulnerabilities before they are deployed.

Falco is a runtime security tool that provides real-time monitoring and detection of suspicious activity in containerized environments. Falco uses a set of customizable rules to detect potential security threats, such as unauthorized access, privilege escalation, and abnormal network traffic. When a rule is triggered, Falco generates an alert that can be used to initiate a response, such as blocking the suspicious activity or triggering an incident response process. [8]

Table 4: Comparison of Container-Specific Security Tools

Tool	Description	Key Features
Docker Bench for Security	Security audit tool for Docker containers	Configuration checks, security best practices, reporting
Clair	Vulnerability scanner for container images	Image scanning, vulnerability database integration, reporting
Falco	Runtime security monitoring tool for containerized environments	Real-time monitoring, customizable rules, alerting
Aqua Security	Comprehensive container security platform	Image scanning, runtime protection, network segmentation

Using a combination of these tools can significantly enhance the security of containerized environments. Docker Bench for Security provides a solid foundation for ensuring that Docker containers are configured securely, while Clair and Falco offer powerful capabilities for identifying and responding to security threats. Additionally, tools like Aqua Security provide a comprehensive security platform that integrates with container orchestration tools like Kubernetes, offering features such as network segmentation, runtime protection, and compliance reporting.

Best Practices for Secure Container Management

Implementing best practices for secure container management is essential for protecting containerized environments from security threats. The following sections outline key best practices that organizations should adopt to enhance the security of their containerized environments.

1. Adopt a Defense-in-Depth Approach

A defense-in-depth approach is a layered security strategy that involves implementing multiple security measures at different levels of the containerized environment. This approach ensures that even if one security measure fails, other measures remain in place to protect the environment. Defense-in-depth includes a combination of technical controls, such as network segmentation and runtime protection, as well as organizational controls, such as security policies and incident response plans.

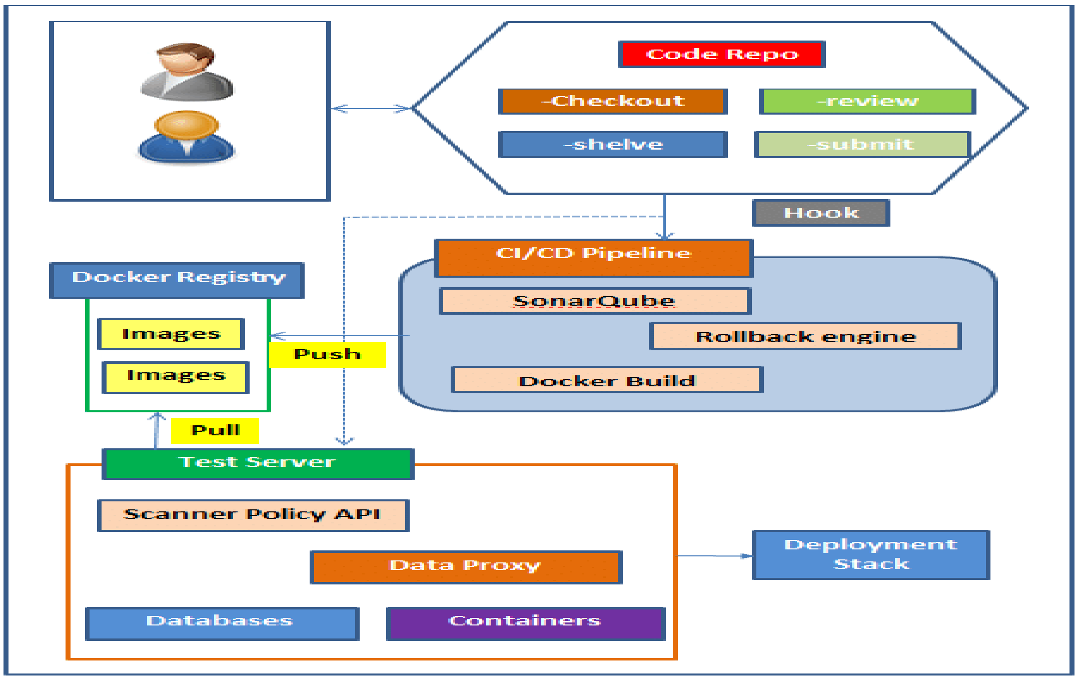
One key aspect of defense-in-depth is the use of multiple security tools to address different types of threats. For example, organizations can use Docker Bench for Security to ensure that containers are configured securely, Clair to scan container images for vulnerabilities, and Falco to monitor runtime activity for suspicious behavior. By using a combination of tools, organizations can detect and respond to a wide range of security threats. [9]

Another important aspect of defense-in-depth is the implementation of network segmentation. As discussed earlier, network segmentation involves dividing the network into isolated segments, each with its own security policies and access controls. This approach limits the potential impact of a security breach by preventing attackers from moving laterally across the network.

In addition to technical controls, organizations should implement strong security policies and procedures. These policies should govern the behavior of containers, define access controls, and establish guidelines for the secure management of containerized environments. Security policies should also include guidelines for incident response, ensuring that the organization is prepared to detect, respond to, and recover from security incidents.

2. Regularly Update and Patch Systems

Keeping the container runtime, host system, and container images up to date is critical for preventing attacks that exploit known vulnerabilities. Regular updates and patches should be applied promptly to minimize the risk of exploitation. Organizations should establish a process for regularly checking for updates and applying patches, ensuring that all components of the containerized environment are protected against the latest security threats. [10]



In addition to updating and patching container images, organizations should also ensure that the host system and container runtime are kept up to date. The host system is the foundation of the containerized environment, and vulnerabilities in the host can have serious security implications for the entire environment. Regularly updating the host operating system and applying security patches is essential for maintaining the security of the containerized environment.

Similarly, the container runtime, such as Docker or containerd, should be regularly updated to ensure that it is protected against known vulnerabilities. The container runtime is responsible for managing the execution of containers, and vulnerabilities in the runtime can lead to serious security issues, such as container escape. Regular updates and patches should be applied to the container runtime to ensure that it remains secure. [11]

Organizations should also consider implementing automated update and patch management processes.

Automated tools can help ensure that updates and patches are applied promptly, reducing the risk of vulnerabilities being exploited. These tools can also provide alerts and notifications when updates are available, helping organizations stay informed about the latest security threats. [12]

3. Enforce Least Privilege

The principle of least privilege is a fundamental security concept that dictates that users, processes, and containers should have only the minimum level of access required to perform their functions. Enforcing least privilege is essential for minimizing the risk of security breaches and reducing the potential impact of a security incident.

In the context of containerized environments, enforcing least privilege involves several key practices. First, containers should be configured to run as non-root users wherever possible. Running containers as root is a dangerous practice that can lead to serious security vulnerabilities, as it grants the container full access to the underlying host system. By running containers as non-root users, organizations can reduce the risk of a security breach and limit the potential damage if a container is compromised.

Second, access controls should be implemented to restrict access to sensitive resources. Containers should have access only to the resources they need to perform their functions, and access to sensitive files, directories, and system resources should be tightly controlled. This can be achieved through the use of Linux capabilities, which allow administrators to grant specific privileges to containers, and through the use of security profiles, such as AppArmor or SELinux, which provide additional layers of access control.

Third, network access should be restricted based on the principle of least privilege. Containers should only be allowed to communicate with other containers, services, or external endpoints that are necessary for their operation. Network policies, firewalls, and security groups can be used to enforce these restrictions, ensuring that containers do not have unnecessary access to network resources.

Finally, organizations should regularly review and update access controls to ensure that they remain aligned with the principle of least privilege. As the containerized environment evolves, new containers, services, and users may be introduced, and existing access controls may need to be adjusted to reflect these changes. Regular reviews help ensure that access controls remain effective and that security risks are minimized.

4. Monitor and Log Container Activity

Continuous monitoring and logging of container activity are essential for detecting and responding to security threats in real time. Monitoring tools can provide visibility into the behavior of containers, allowing organizations to identify suspicious activity, such as unauthorized access, privilege escalation, or abnormal network traffic. Logging tools can capture detailed records of container activity, providing valuable information for forensic analysis and incident response.

One important practice for monitoring container activity is the use of real-time security monitoring tools, such as Falco. These tools use customizable rules to detect potential security threats and generate alerts when suspicious activity is detected. For example, Falco can be configured to monitor for unauthorized access attempts, changes to critical files, or abnormal network connections. When a rule is triggered, Falco generates an alert that can be used to initiate a response, such as blocking the suspicious activity or triggering an incident response process. [13]

Table 5: Monitoring and Logging Best Practices

Practice	Description
Centralized Logging	Aggregate logs from all containers and hosts for centralized analysis
Real-Time Monitoring	Use tools to monitor container activity in real time
Anomaly Detection	Identify unusual patterns in container behavior using monitoring tools
Incident Response	Implement procedures for responding to detected threats

Centralized logging is another key practice for managing container security. In a containerized environment, logs are generated by multiple components, including the container runtime, the host system, and the application itself. Centralized logging involves aggregating these logs into a single location, where they can be analyzed for security threats. Centralized logging platforms, such as the ELK stack (Elasticsearch, Logstash, and Kibana), provide powerful tools for searching, analyzing, and visualizing log data, making it easier to detect and investigate security incidents. [5]

Anomaly detection is an important aspect of security monitoring in containerized environments. Anomaly detection tools can analyze container activity and identify patterns that deviate from normal behavior, such as unexpected spikes in CPU usage, unusual network connections, or changes to critical files. By detecting these anomalies, organizations can identify potential security threats before they lead to a full-blown incident. [10]

Incident response procedures are critical for ensuring that security threats are addressed promptly and effectively. Organizations should establish clear procedures for responding to security incidents, including steps for investigating the incident, containing the threat, and restoring normal operations. Incident response procedures should also include communication protocols for notifying relevant stakeholders and documenting the incident for future analysis.

Conclusion

Containerized environments offer significant advantages in terms of scalability, flexibility, and consistency, but they also introduce unique security challenges. To protect these environments from evolving threats, organizations must implement a comprehensive security strategy that addresses the various risks associated with containerization.

This paper has explored the common threat vectors in containerized environments, including container escape, image vulnerabilities, network attacks, insecure configurations, and supply chain attacks. We have also outlined effective mitigation strategies, such as implementing robust security policies, using hardened images, employing network segmentation, conducting regular security audits, and using container-specific security tools.

In addition, we have provided a set of best practices for secure container management, including adopting a defense-in-depth approach, regularly updating and patching systems, enforcing least privilege, and monitoring and logging container activity. By following these best practices, organizations can enhance the security of their containerized environments and protect against a wide range of security threats.

As the use of containers continues to grow, staying ahead of potential threats will require ongoing vigilance, regular updates, and the adoption of security best practices tailored to containerized environments. Organizations must remain proactive in their approach to container security, continuously evaluating and improving their security posture to address new and emerging threats.

References

[1] Zhang H.. "Rainbowd: a heterogeneous cloud-oriented efficient docker image distribution system."

Jisuanji Xuebao/Chinese Journal of Computers 43.11 (2020): 2067-2083.

[2] Yang H.. "Design and implementation of fast fault detection in cloud infrastructure for containerized iot services." Sensors (Switzerland) 20.16 (2020): 1-13.

[3] Jani, Y. "Security best practices for containerized applications." Journal of Scientific and Engineering Research 8.8 (2021): 217-221.

[4] Zahoor S.. "Resource management in pervasive internet of things: a survey." Journal of King Saud University - Computer and Information Sciences 33.8 (2021): 921-935.

[5] Iacobucci D.. "A chronology of health care marketing research." Foundations and Trends in Marketing 13.2-4 (2019): 77-529.

[6] Joseph C.T.. "Straddling the crevasse: a review of microservice software architecture foundations and recent advancements." Software - Practice and Experience 49.10 (2019): 1448-1484.

[7] Meshcheryakov R.. "Analysis of modern methods to ensure data integrity in cyber-physical system management protocols." Informatics and Automation 19.5 (2020): 1089-1122.

[8] Doan T.P.. "Davs: dockerfile analysis for container image vulnerability scanning." Computers, Materials and Continua 72.1 (2022): 1699-1711.

[9] Nguyen V.L.. "Security and privacy for 6g: a survey on prospective technologies and challenges." IEEE Communications Surveys and Tutorials 23.4 (2021): 2384-2428.

[10] Farris I.. "A survey on emerging sdn and nfv security mechanisms for iot systems." IEEE Communications Surveys and Tutorials 21.1 (2019): 812-837.

[11] Walkowski M.. "Efficient algorithm for providing live vulnerability assessment in corporate network environment." Applied Sciences (Switzerland) 10.21 (2020): 1-16.

[12] Sutikno T.. "Insights on the internet of things: past, present, and future directions." Telkomnika (Telecommunication Computing Electronics and Control) 20.6 (2022): 1399-1420.

[13] Dissanayaka A.M.. "Security assurance of mongodb in singularity lxcs: an elastic and convenient testbed using linux containers to explore vulnerabilities." Cluster Computing 23.3 (2020): 1955-1971.