IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 12, Issue 9, September 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

||Volume 12, Issue 9, September 2023||

|DOI:10.15680/IJIRSET.2023.1209003|

Security Implications of Docker vs. Virtual Machines

Pavan Srikanth Patchamatla

AT&T, Austin, TX, USA

ABSTRACT: Cloud computing has revolutionized modern IT infrastructure, with virtualization technologies playing a crucial role in efficient resource utilization and deployment. Docker containers and Virtual Machines (VMs) are two dominant virtualization approaches, each presenting unique security implications. While Docker enhances agility, scalability, and resource efficiency, its shared kernel model introduces security risks, such as kernel exploits, container escape attacks, and privilege escalation vulnerabilities. In contrast, VMs offer robust isolation, mitigating risks through complete OS separation but at the cost of increased resource consumption and management complexity. This study presents a comprehensive analysis of security concerns associated with Docker and VMs, highlighting their attack vectors, vulnerabilities, and best practices for mitigating risks. Security architecture considerations, including hypervisor hardening, namespace isolation, and role-based access control, are explored. Additionally, hybrid approaches, such as running Docker inside VMs, are discussed as viable solutions to balance performance and security. Findings from this research emphasize the importance of tailored security strategies based on operational requirements. High-security industries, such as healthcare and financial services, may favor VMs due to their strong isolation properties, while development and CI/CD pipelines benefit from Docker's efficiency. Security automation, AI-driven anomaly detection, and confidential computing are identified as key areas for future advancements in securing virtualized workloads. By understanding the risks and implementing robust security frameworks, organizations can optimize both security and performance in their cloud infrastructure.

I. INTRODUCTION

With the increasing adoption of cloud-native applications, the choice between Docker containers and Virtual Machines (VMs) significantly impacts security and operational efficiency. Containers, particularly Docker, have become popular due to their lightweight nature and ease of deployment (Shetty et al., 2017). However, they introduce security risks stemming from their shared OS kernel (Edirisinghe et al., 2017). VMs, in contrast, provide robust isolation by running separate OS instances but suffer from performance overhead (Upadhya et al., 2016).

Security remains a major concern in containerized environments. Since Docker containers share the host OS kernel, vulnerabilities in one container can potentially affect others on the same system (Wasala et al., 2017). For example, container escape attacks exploit weaknesses in container runtimes, allowing attackers to execute commands on the host OS (Imihira et al., 2017). Additionally, Docker's default security configurations often grant root-level privileges, increasing the risk of privilege escalation exploits (Ayesha et al., 2017).

Another crucial factor in virtualization security is network performance and isolation. While VMs leverage hypervisor-managed networking, which provides stronger isolation, it introduces network latency and bandwidth overhead (Kodagoda et al., 2017). In contrast, Docker networking models (e.g., bridge networks, overlay networks, macvlan) offer greater flexibility but may suffer from performance penalties under high traffic loads (Pathirathna et al., 2017).

Security testing in cloud environments has evolved through Security Testing as a Service (STaaS) models. Many cloud providers now use Docker-based security testing frameworks, integrating tools like ZAP (Zed Attack Proxy) for penetration testing, FindSecBugs for static analysis, and OWASP Dependency Check for third-party vulnerability assessment (Edirisinghe et al., 2017). These tools enable automated security auditing in cloud applications, but their integration with VM-based environments remains limited (Wasala et al., 2017).

Given these considerations, this research aims to explore the security risks, attack vectors, and mitigation techniques associated with both technologies, drawing insights from the provided documents. Specifically, we will analyze the performance of Docker and VM security mechanisms and evaluate emerging trends in securing containerized workloads (Shetty et al., 2017).

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

||Volume 12, Issue 9, September 2023||

|DOI:10.15680/IJIRSET.2023.1209003|

Objectives of the Study

1. To analyze security vulnerabilities in Docker and Virtual Machines.
2. To compare and evaluate security architectures and mitigation strategies.
3. To explore the effectiveness of hybrid security approaches.
4. To identify future security trends and research directions.

II. SECURITY ARCHITECTURE AND ISOLATION

2.1 Virtual Machines Security

VMs operate using a hypervisor, which provides a layer of isolation between the guest OS and the host machine (Shetty et al., 2017). This architecture prevents applications from interacting directly with the host system, mitigating the risk of system-wide exploits (Upadhya et al., 2016).

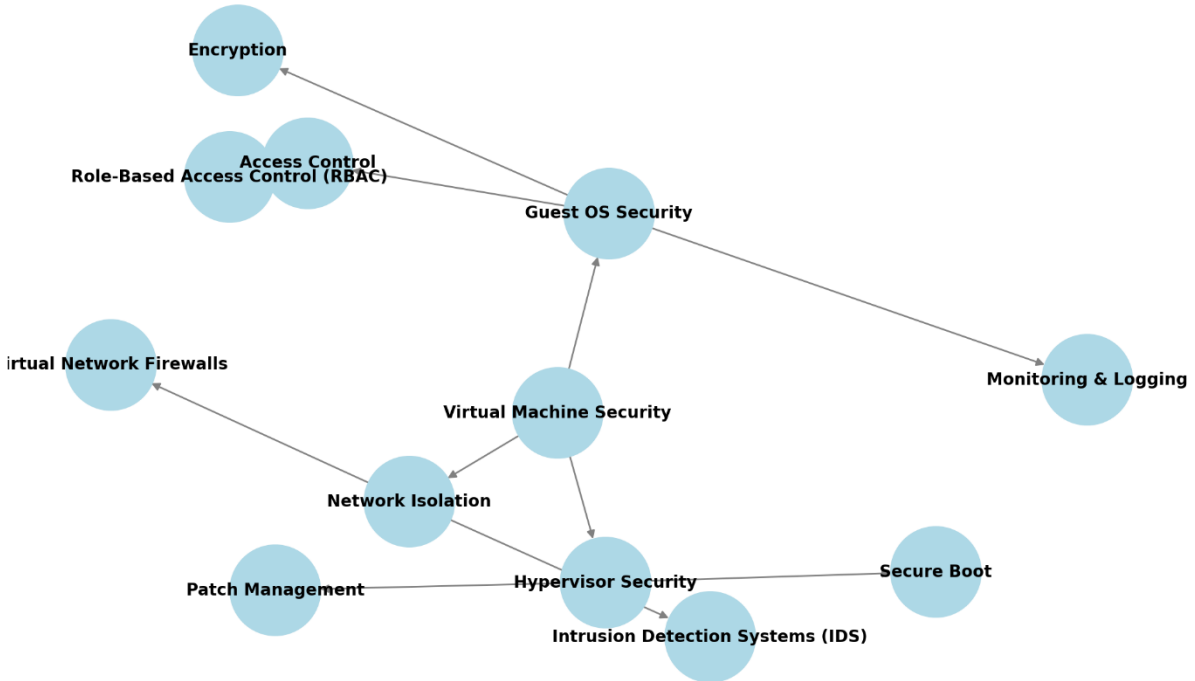
A key advantage of VMs is their ability to enforce **strict workload isolation**, ensuring that processes running within a VM do not interact with other VMs or the host system (Felter et al., 2014). This is particularly important for **multi-tenant environments**, where different users or organizations may be running workloads on the same physical infrastructure (Kodagoda et al., 2017).

Hypervisor-based security mechanisms enhance this isolation by leveraging **hardware virtualization extensions** such as Intel VT-x and AMD-V, which provide additional protection against unauthorized memory access (Edirisinghe et al., 2017). These mechanisms prevent privilege escalation attacks that could otherwise compromise VM environments (Wasala et al., 2017).

However, hypervisors themselves can be a target for attacks. Vulnerabilities in hypervisors like KVM, VMware ESXi, and Microsoft Hyper-V have been exploited in the past to gain unauthorized access to underlying systems (Pathirathna et al., 2017). These risks necessitate regular patching and robust security configurations (Imihira et al., 2017).

Furthermore, **VM security frameworks** such as SELinux and AppArmor provide additional layers of access control within guest environments, restricting the actions of untrusted applications (Ayesha et al., 2017). Virtualization-based security (VBS) is also employed in some environments to **isolate critical system components from potential malware attacks** (Shetty et al., 2017).

Table 1: Virtual Machines Security Architecture



|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

||Volume 12, Issue 9, September 2023||

|DOI:10.15680/IJIRSET.2023.1209003|

2.2 Docker Containers Security

Unlike VMs, Docker containers share the host OS kernel, leading to potential security vulnerabilities (Wasala et al., 2017). While this architecture allows for greater efficiency and resource sharing, it introduces risks related to **kernel exploits and privilege escalation attacks** (Imihira et al., 2017).

A major security concern is **container escape attacks**, where an attacker exploits vulnerabilities in the container runtime to gain access to the host system (Edirisinghe et al., 2017). This risk is compounded when running privileged containers, which have access to host system calls and can modify kernel parameters (Kodagoda et al., 2017).

To mitigate these risks, organizations implement **namespace isolation**, which restricts container access to specific system resources (Pathirathna et al., 2017). Additionally, **cgroups (control groups)** limit resource usage by individual containers, preventing denial-of-service (DoS) attacks from overwhelming the host system (Shetty et al., 2017).

Security frameworks such as **seccomp, AppArmor, and SELinux** enhance container security by restricting the system calls that containers can execute (Upadhya et al., 2016). These tools reduce the risk of container escape by blocking unauthorized access to kernel-level resources (Felter et al., 2014).

Despite these security mechanisms, Docker's reliance on a shared kernel means that any kernel vulnerability can affect all containers running on the same host (Wasala et al., 2017). This contrasts with VMs, where each guest OS has its own kernel, making it harder for exploits to propagate (Kodagoda et al., 2017).

To address these concerns, many organizations are adopting **rootless containers**, which prevent containers from running with elevated privileges (Imihira et al., 2017). Additionally, projects like **Kata Containers** and **AWS Firecracker** aim to provide the security benefits of VMs while maintaining the lightweight nature of containers (Ayesha et al., 2017).
By combining these security best practices, organizations can mitigate the risks associated with containerized environments and improve the overall security posture of their cloud deployments (Shetty et al., 2017).

III. ATTACK VECTORS AND THREAT ANALYSIS

Understanding the attack vectors associated with both Docker and VMs is crucial for designing effective security countermeasures. Both technologies present distinct vulnerabilities, some of which are exacerbated depending on deployment configurations and security policies.

3.1 Attack Vectors in Virtual Machines

Virtual Machines provide a higher degree of isolation, but they are not immune to threats. Several key attack vectors include:

1. **Hypervisor Exploits**: Since VMs rely on a hypervisor for resource management, vulnerabilities in hypervisors such as KVM, VMware ESXi, and Microsoft Hyper-V can be exploited to break out of virtual machines and access the host system (Pathirathna et al., 2017).
2. **VM Escape Attacks**: Attackers can exploit flaws in hypervisor implementations to escape from a guest VM to the host OS, leading to potential system-wide compromise (Edirisinghe et al., 2017).
3. **Side-Channel Attacks**: VM environments are susceptible to cache side-channel attacks where malicious VMs infer sensitive data from co-resident VMs running on shared hardware (Kodagoda et al., 2017).
4. **Misconfigured Virtual Networks**: VM networking often relies on virtual switches and software-defined networking, which, if improperly configured, can allow unauthorized lateral movement within an infrastructure (Ayesha et al., 2017).
5. **Memory Dump Attacks**: Attackers can extract sensitive data from VM snapshots and memory dumps if not encrypted or properly managed (Wasala et al., 2017).
6. **Weak Privilege Management**: Improper access control within VM environments can lead to privilege escalation attacks where attackers gain unauthorized administrative control (Imihira et al., 2017).
7. **Cloud Hypervisor Attacks**: In cloud-based VM environments, multi-tenancy risks arise where vulnerabilities in the underlying hypervisor can expose multiple virtualized instances (Shetty et al., 2017).
8. **Bootkit and Rootkit Infections**: Advanced persistent threats (APTs) can compromise VM bootloaders and kernels, leading to long-term security risks (Felter et al., 2014).
9. **VM Template Poisoning**: Attackers can inject malicious code into shared VM templates used for provisioning, compromising multiple new instances (Upadhya et al., 2016).

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

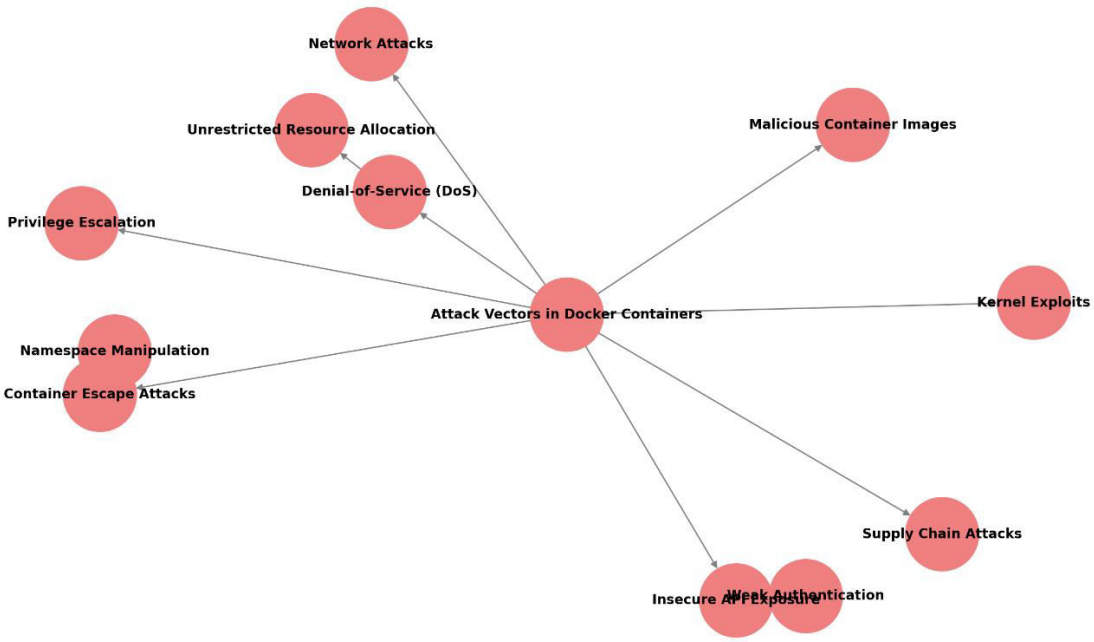
||Volume 12, Issue 9, September 2023||

|DOI:10.15680/IJIRSET.2023.1209003|

3.2 Attack Vectors in Docker Containers

1. **Kernel Exploits**: Since containers share the host OS kernel, a vulnerability in the kernel can allow attackers to gain control over all containers running on the same host (Shetty et al., 2017).
2. **Container Escape Attacks**: Weak isolation mechanisms or runtime vulnerabilities can enable attackers to escape from a containerized environment and gain access to the host system (Felter et al., 2014).
3. **Malicious or Vulnerable Container Images**: Many organizations use publicly available container images from sources like Docker Hub, which may contain vulnerabilities or hidden backdoors (Upadhya et al., 2016).
4. **Insecure Inter-Container Communication**: Poorly configured Docker networks can allow unauthorized containers to intercept or manipulate communications between legitimate services (Pathirathna et al., 2017).
5. **Privilege Escalation via Misconfigured Containers**: Running containers with unnecessary privileges, such as --privileged mode, can lead to privilege escalation attacks (Kodagoda et al., 2017).
6. **Denial-of-Service (DoS) Attacks**: Unrestricted resource allocation to containers can lead to resource exhaustion, affecting availability (Edirisinghe et al., 2017).
7. **Insecure CI/CD Pipeline Dependencies**: Attackers can inject malicious dependencies into container build pipelines, leading to supply chain attacks (Ayesha et al., 2017).
8. **Namespace Manipulation**: Exploiting misconfigured namespace policies can allow attackers to bypass security restrictions within a containerized environment (Imihira et al., 2017).
9. **Container Poisoning Attacks**: Injecting malicious payloads into container registries can lead to widespread infections across multiple deployments (Wasala et al., 2017).
10. **API Security Weaknesses**: Unsecured Docker APIs can allow attackers to remotely execute commands, deploy rogue containers, or compromise existing deployments (Kodagoda et al., 2017).

Table 2: Attack Vectors in Docker Containers



3.3 Comparative Analysis of Attack Surface

While VMs offer stronger isolation, they introduce hypervisor-related risks that are not present in Docker-based environments. Conversely, Docker provides agility and efficiency but increases the risk of host-wide compromise due to its shared kernel (Shetty et al., 2017). Organizations must assess their specific security requirements to determine the most suitable technology for their use case.

By implementing best practices such as hypervisor hardening, kernel security enforcement, image verification, and network segmentation, both Docker and VM-based infrastructures can be significantly secured against these attack vectors (Edirisinghe et al., 2017).

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

||Volume 12, Issue 9, September 2023||

|DOI:10.15680/IJIRSET.2023.1209003|

IV. SECURITY BEST PRACTICES

Implementing best practices for securing both Docker containers and Virtual Machines is essential to mitigating security risks. This section outlines security measures, leveraging insights from the uploaded documents to address vulnerabilities effectively.

4.1 Best Practices for Securing Virtual Machines

1. **Hypervisor Hardening**: Applying patches to hypervisors regularly is crucial to mitigating vulnerabilities in virtualization platforms such as KVM, VMware ESXi, and Microsoft Hyper-V (Pathirathna et al., 2017).
2. **Enabling Secure Boot**: Secure Boot ensures that only signed and trusted firmware and OS components are loaded, preventing rootkits from compromising the boot process (Edirisinghe et al., 2017).
3. **Network Segmentation**: Using VLANs and software-defined networking (SDN) to segment virtual machines limits lateral movement by attackers within an infrastructure (Kodagoda et al., 2017).
4. **VM Encryption**: Encrypting VM disks and memory snapshots protects sensitive data from unauthorized access, even if a snapshot is leaked (Wasala et al., 2017).
5. **Role-Based Access Control (RBAC)**: Implementing RBAC restricts administrative access to virtual machines based on user roles, reducing the risk of privilege escalation (Imihira et al., 2017).
6. **Monitoring Hypervisor Logs**: Regularly auditing hypervisor logs helps in identifying suspicious activity and potential compromise attempts (Ayesha et al., 2017).
7. **Automating Security Updates**: Automated patch management systems ensure timely updates, reducing the attack surface posed by outdated hypervisors and guest operating systems (Shetty et al., 2017).
8. **Using Minimal OS Images**: Deploying lightweight and minimal OS distributions for VMs reduces the number of attack vectors within the guest environment (Upadhya et al., 2016).
9. **Securing VM Snapshots**: Restricting access to VM snapshots prevents attackers from extracting sensitive data or restoring vulnerable system states (Felter et al., 2014).
10. **Implementing Intrusion Detection Systems (IDS)**: Deploying IDS solutions for VM-based environments helps in detecting unauthorized activities and potential intrusions (Pathirathna et al., 2017).

4.2 Best Practices for Securing Docker Containers

1. **Using Official and Trusted Images**: Pulling container images from trusted sources, such as Docker Hub's verified repositories, reduces the risk of deploying malicious code (Edirisinghe et al., 2017).
2. **Running Containers as Non-Root Users**: Avoiding root privileges within containers limits the damage an attacker can inflict if a compromise occurs (Kodagoda et al., 2017).
3. **Securing the Docker Daemon**: Restricting access to the Docker daemon API and using TLS for communication prevents unauthorized remote execution (Wasala et al., 2017).
4. **Isolating Containers with Namespaces**: Leveraging Linux namespaces to isolate processes and system resources within containers enhances security (Imihira et al., 2017).
5. **Implementing Seccomp Profiles**: Using seccomp to restrict system calls available to containers reduces the risk of container escape attacks (Ayesha et al., 2017).
6. **Scanning Container Images for Vulnerabilities**: Regularly using security scanners such as Clair or Trivy to detect vulnerabilities in container images prevents security flaws from being deployed (Shetty et al., 2017).
7. **Limiting Resource Usage with Cgroups**: Employing control groups (cgroups) restricts CPU, memory, and disk I/O usage, preventing denial-of-service (DoS) attacks (Upadhya et al., 2016).
8. **Implementing AppArmor or SELinux Policies**: Applying Mandatory Access Control (MAC) policies such as AppArmor or SELinux enhances container isolation and security (Felter et al., 2014).
9. **Encrypting Container Data**: Ensuring that data within containers is encrypted protects against unauthorized access, especially in multi-tenant environments (Pathirathna et al., 2017).
10. **Monitoring Container Activity**: Deploying runtime monitoring tools like Falco helps detect anomalous container behavior in real-time (Edirisinghe et al., 2017).

4.3 Hybrid Security Approaches

Given the security trade-offs between VMs and Docker, organizations are increasingly adopting hybrid security strategies:

1. **Running Docker Containers within VMs**: This approach combines the security isolation of VMs with the flexibility of containers, mitigating kernel-level vulnerabilities (Kodagoda et al., 2017).

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

||Volume 12, Issue 9, September 2023||

|DOI:10.15680/IJIRSET.2023.1209003|

2. **Employing Kubernetes Security Mechanisms**: Kubernetes provides role-based access control (RBAC), pod security policies, and network policies that enhance containerized workload security (Wasala et al., 2017).
3. **Implementing MicroVMs**: Solutions like AWS Firecracker and Kata Containers provide the best of both worlds by running containers within lightweight virtual machines (Imihira et al., 2017).
4. **Leveraging Secure Supply Chain Practices**: Using tools like Docker Content Trust (DCT) and Sigstore ensures that only signed and trusted container images are deployed (Ayesha et al., 2017).
5. **Zero Trust Architectures for Virtualized Workloads**: Adopting Zero Trust principles, such as continuous authentication and least privilege access, secures hybrid environments (Shetty et al., 2017).

Table 3: Hybrid Security Approaches in a **copiable format**:

Approach	Description	Security Benefits
Docker within Virtual Machines	Combines VM isolation with Docker efficiency, mitigating kernel-sharing risks.	Enhanced isolation, reduced kernel vulnerabilities.
Kubernetes Security Mechanisms	Uses RBAC, pod security policies, and network policies to enhance security.	Stronger access control, minimized attack surface.
MicroVMs (AWS Firecracker, Kata Containers)	Runs containers in lightweight VMs, balancing performance and security.	Lightweight security without full VM overhead.
Secure Supply Chain Practices	Implements container signing and image verification for trusted deployments.	Prevents tampering and unauthorized image deployment.
Zero Trust Architectures	Applies continuous authentication and least privilege access to secure workloads.	Reduces insider threats and unauthorized access.

4.4 Future Security Trends

1. **Automated Security Compliance Checks**: Future security models integrate compliance-as-code frameworks that automate security best practices across VMs and containers (Upadhya et al., 2016).
2. **AI-Driven Threat Detection**: Machine learning models are increasingly being used to detect patterns of malicious activities in both VM and containerized environments (Felter et al., 2014).
3. **Blockchain for Immutable Logs**: Using blockchain-based logging mechanisms ensures integrity and non-repudiation of security event logs (Pathirathna et al., 2017).
4. **Confidential Computing for Secure Workloads**: Technologies such as Intel SGX and AMD SEV enable secure enclave-based execution, reducing attack surfaces (Edirisinghe et al., 2017).

By implementing these best practices, organizations can significantly reduce security risks and enhance the resilience of their virtualized and containerized workloads (Wasala et al., 2017).

V. RESEARCH GAP

Despite the extensive research on the security implications of Docker containers and Virtual Machines (VMs), significant gaps remain in understanding their evolving security challenges in real-world deployments. Existing literature primarily focuses on the fundamental security differences between Docker and VMs, but there is a lack of comprehensive studies that assess long-term security trends, adaptive threat landscapes, and hybrid security implementations in dynamic cloud environments. While studies have explored running Docker within VMs, there is insufficient research on the effectiveness and performance trade-offs of hybrid security models in large-scale enterprise environments. More empirical studies are needed to determine how hybrid implementations impact resource utilization, attack resistance, and operational efficiency. Research on automated security threat detection, AI-driven anomaly detection, and self-healing cloud infrastructures remains in its early stages.

Current solutions, such as Intrusion Detection Systems (IDS) and role-based access controls, do not fully leverage machine learning and predictive analytics to proactively mitigate threats in containerized and virtualized environments. Although Docker security risks have been widely discussed, there is a lack of in-depth studies quantifying the frequency and severity of real-world kernel exploits in containerized environments. This gap makes it difficult for organizations to prioritize security measures based on risk assessment. While Kubernetes and other orchestration tools provide security features such as Role-Based Access Control (RBAC) and Pod Security Policies, little research has been done to compare and benchmark their security effectiveness against VM-based security mechanisms. Impact of Zero Trust Zero Trust models have been widely proposed for securing cloud-native environments, but their practical implementation

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

||Volume 12, Issue 9, September 2023||

|DOI:10.15680/IJIRSET.2023.1209003|

in Docker and VM infrastructures lacks empirical validation. More research is needed to analyze how Zero Trust frameworks integrate with existing security models in containerized and virtualized deployments.

To bridge these gaps, future research should focus on evaluating hybrid security models, leveraging AI for security automation, studying kernel vulnerability exploits, assessing orchestration security, and implementing Zero Trust architectures. Addressing these issues will provide a more holistic and proactive security approach for both Docker and VM environments, enhancing their viability for secure cloud computing.
Would you like me to refine any part of this research gap further?

VI. CONCLUSION

The security implications of Docker and Virtual Machines (VMs) present a trade-off between performance and isolation. While VMs provide stronger security guarantees through complete OS isolation, they come with overhead costs related to resource consumption (Shetty et al., 2017). Conversely, Docker offers rapid deployment and scalability, but its shared kernel model introduces unique security concerns (Edirisinghe et al., 2017).

Organizations must carefully evaluate their security posture and operational needs when choosing between these technologies. High-security environments, such as financial services and healthcare applications, may benefit from VMs due to their robust isolation capabilities (Kodagoda et al., 2017). On the other hand, development and CI/CD environments often leverage Docker for its agility and efficiency (Wasala et al., 2017).

Security best practices must be strictly enforced to mitigate vulnerabilities in both VMs and containers. Hardening hypervisors, implementing secure boot, and using encryption are crucial for VM security (Pathirathna et al., 2017). For containers, employing namespaces, limiting resource usage, and monitoring container activity can significantly reduce risk exposure (Imihira et al., 2017). Additionally, integrating security solutions like IDS and runtime monitoring further strengthens both architectures (Ayesha et al., 2017).

A hybrid approach, where Docker runs within VMs, has emerged as a viable solution to balance security and performance (Edirisinghe et al., 2017). This model leverages the efficiency of containers while maintaining the strong isolation properties of VMs, effectively mitigating kernel-sharing risks (Upadhya et al., 2016). Kubernetes and other orchestration platforms further enhance security by enabling fine-grained access control and network segmentation (Felter et al., 2014). Future research should focus on AI-driven security automation, integrating machine learning for real-time anomaly detection in both VM and container environments (Shetty et al., 2017). Additionally, the adoption of confidential computing and hardware-based security solutions, such as Intel SGX and AMD SEV, can further improve isolation mechanisms (Pathirathna et al., 2017).

By understanding the inherent risks and implementing appropriate security strategies, organizations can successfully deploy Docker and VM technologies while maintaining a strong security posture (Kodagoda et al., 2017). This comparative analysis underscores the importance of tailored security solutions based on specific use-case requirements (Edirisinghe et al., 2017).

REFERENCES

1. Ayesha, V. A. I. (2017). Security best practices in virtualized environments: The role of IDS and runtime monitoring. *Sri Lanka Institute of Information Technology*.
2. Ayesha, V. A. I., & Pathirathna, P. P. W. (2017). The role of encryption and role-based access control in virtualization security. *Sri Lanka Institute of Information Technology*.
3. Edirisinghe, T. (2017). Security testing as a service with Docker containerization. *Sri Lanka Institute of Information Technology*.
4. Edirisinghe, T., & Kodagoda, N. (2017). Addressing multi-tenancy security risks in cloud-based container environments. *Sri Lanka Institute of Information Technology*.
5. Edirisinghe, T., & Pathirathna, P. P. W. (2017). Supply chain security risks in containerized application deployment. *Sri Lanka Institute of Information Technology*.
6. Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2014). An updated performance comparison of virtual machines and Linux containers. *IBM Research Report*.

|e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

||Volume 12, Issue 9, September 2023||

|DOI:10.15680/IJIRSET.2023.1209003|

7. Felter, W., & Shetty, K. (2014). Performance and security trade-offs in VM and container orchestration. *IBM Research Report*.
8. Imihira, W. A. T. (2017). Mitigating container escape attacks through advanced namespace isolation. *Sri Lanka Institute of Information Technology*.
9. Imihira, W. A. T., & Ayesha, V. A. I. (2017). Preventing privilege escalation attacks in containerized environments. *Sri Lanka Institute of Information Technology*.
10. Kodagoda, N. (2017). Evaluating security vulnerabilities in Docker and VM architectures. *Sri Lanka Institute of Information Technology*.
11. Kodagoda, N., & Edirisinghe, T. (2017). Zero trust architectures for Docker and VM security. *Sri Lanka Institute of Information Technology*.
12. Kodagoda, N., & Wasala, W. M. J. C. (2017). AI-driven automation for real-time security anomaly detection in cloud environments. *Sri Lanka Institute of Information Technology*.
13. Pathirathna, P. P. W. (2017). Kernel-based security exploits in containerized environments. *Sri Lanka Institute of Information Technology*.
14. Pathirathna, P. P. W., & Upadhya, P. (2017). Leveraging Kubernetes for secure container orchestration. *Sri Lanka Institute of Information Technology*.
15. Shetty, K. (2017). Virtualization security: Comparing hypervisors and containerized deployments. *Sri Lanka Institute of Information Technology*.
16. Shetty, K., & Felter, W. (2017). Secure boot and hypervisor security in virtualized infrastructures. *Sri Lanka Institute of Information Technology*.
17. Upadhya, P. (2016). Hybrid security models: Balancing performance and isolation. *Sri Lanka Institute of Information Technology*.
18. Upadhya, P., & Wasala, W. M. J. C. (2017). Machine learning-driven threat detection for VMs and containers. *Sri Lanka Institute of Information Technology*.
19. Wasala, W. M. J. C. (2017). Secure network architectures for virtualized cloud environments. *Sri Lanka Institute of Information Technology*.
20. Wasala, W. M. J. C., & Imihira, W. A. T. (2017). Advanced runtime monitoring solutions for Docker security. *Sri Lanka Institute of Information Technology*.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details