Addressing Security Debt in Cloud-Native Development Environments

CNIT 581 ACC - Research Paper

Tarun Aiyappa
Department of Computer and Information Technology
Purdue University
West Lafayette, IN
taiyappa@purdue.edu

***Abstract*—Cloud-native development tools (Docker, Vagrant, Kind) and managed cloud services (AWS ECS, Azure ACI, GCP Cloud Run) streamline application deployment but introduce systemic security risks such as privilege escalation, supply chain vulnerabilities, and configuration drift. This paper synthesizes existing research on vulnerabilities across hybrid development-to-production workflows, critically evaluating case studies such as CVE-2024-1753 and Azurescape. It examines the strengths and limitations of contemporary mitigation tools including Open Policy Agent (OPA), Falco, Trivy, and Sigstore, identifying persistent gaps in secure adoption across continuous integration and deployment pipelines. By analyzing recurring vulnerability propagation patterns and highlighting strategic improvement pathways, this study advocates for a holistic, resilience-focused approach to security in cloud-native ecosystems without sacrificing developer productivity.**

***Index Terms*—Cloud Security, Container Security, Privilege Escalation, Supply Chain Risks, DevSecOps**

I. Introduction

Modern software development has undergone a paradigm shift with the widespread adoption of cloud-native tools and managed services. Technologies such as Docker, Vagrant, and Kind have enabled developers to streamline workflows, enhance scalability, and accelerate application delivery. Tools like Docker allow applications to be containerized for testing in isolated environments, while Vagrant facilitates the provisioning of virtualized infrastructure for rapid prototyping. Kubernetes in Docker (Kind) offers developers a lightweight way to emulate Kubernetes clusters locally, simplifying the testing and validation of distributed systems [7]. On the production side, managed services such as AWS Elastic Container Service (ECS), Azure Container Instances (ACI), and Google Cloud Run provide scalable platforms for deploying containerized workloads with minimal operational overhead.

Despite their undeniable benefits, these tools and services often prioritize usability and developer convenience over secure defaults. The 2024 Sysdig report [12] indicates that approximately 13% of containers analyzed still ran with root privileges, a configuration that can lead to host escape vulnerabilities if exploited. Similarly, research has found that 62% of AWS ECS tasks are configured with overly permissive IAM roles [15], broadening the attack surface for potential adversaries. Real-world incidents such as the Azurescape vulnerability [14] underscore how shared infrastructure weaknesses can be exploited in cloud environments, resulting in cross-tenant attacks.

Security debt, a concept gaining increasing academic attention, describes the accumulation of unresolved security risks over time due to trade-offs made for faster development or operational efficiency [28]. In cloud-native ecosystems, where local development practices seamlessly transition into production deployments, security debt can propagate rapidly if not addressed systematically. Early-stage vulnerabilities such as insecure container configurations, embedded credentials, or misconfigured access policies can become deeply embedded within production pipelines, amplifying systemic risks [3].

Numerous tools and frameworks are already prevalent that help with the mitigation of security risks at multiple stages; these include vulnerability scanning ((e.g., Trivy [20]) and policy framework enforcement (e.g., OPA [19]). However, these mitigation strategies are implemented in a skewed manner, often fragmented in their approach. As such, they fail to have comprehensive integration across hybrid workflows. This disjointed approach, while good for compartmentalized mitigation in an incident response plan, has some flaws. It does not show various attack vectors that can encompass the system as a whole, thus causing a strain in the integration and continuous deployment (CI/CD) pipelines which ensures seamless flow across all stages of the stack - from deployment to production.

Keeping in mind all the above points, this study aims to understand these inconsistencies that exist in current mitigation strategies across multiple hybrid cloud-native workflows. This paper aims to structurally and theoretically evaluate how current mitigation tools perform in the real-world, and with this it would help current research explore multiple practical ways to improve resilience in the aforementioned environments. The overall aim of this paper is to understand what can be done to improve mitigation strategies while keeping a very large focus on maintaining and improving security standards that exist, whilst ensuring operational efficiency across deployments.

II. Research Gap

Despite significant advancements in cloud-native security, critical research gaps persist in addressing vulnerabilities that propagate across hybrid development and production environments. Existing literature often treats local development tools and managed cloud services as isolated systems, rather than considering the seamless workflows that modern DevOps pipelines enable.

A. Tool-Specific Focus

During the literature review, we reviewed many existing studies that analyze containerization tools such as Docker, Kubernetes, or AWS ECS in isolation, often neglecting how vulnerabilities can propagate across multiple platforms in real-world workflows. For example, a misconfigured Docker image or an insecure Vagrant environment may appear low-risk during development but can introduce significant security flaws when deployed on cloud services like Azure ACI [10]. While research on container orchestration frameworks has addressed risks to operational efficiency, it does not tell the whole story, often underestimating the cumulative impact of insecure practices introduced earlier in the aforementioned pipeline [3]. This narrow perspective leaves hybrid workflows vulnerable to a multitude of attack vectors that attack from weaknesses or backdoors that are persisting across the stack.

B. Fragmented Mitigation Strategies

While we do claim that there is a significant research gap that needs to be explored, there are existing tools and frameworks, such as Open Policy Agent (OPA) [19], Trivy [20], and Falco [21], that offer robust capabilities for vulnerability prevention and detection. They are often deployed independently and from third-party vendors with an outside perspective for the purposes of security validation and auditing. There is limited comprehensive guidance on how to integrate these tools effectively within continuous integration and deployment (CI/CD) workflows. Furthermore, many of these solutions assume a 'hard' or rigid boundary between development and production environments, overlooking the fluid movement of artifacts in modern pipelines [8]. This misjudged view may very well lead to multiple attack vectors as mentioned before, to be much more prevalent as the deployment moves along the stack, and have a very good chance of being undetected if they are able to move past the initial security validation testing.

C. Productivity-Security Tradeoffs

In the current development lifecycle, the approach of "shit-left" is regarded widely as a step in the right direction regarding security validation of these deployments. However, practical challenges to its adoption has proven to be very difficult to implement. For instance, Docker's rootless mode offers clear security advantages, yet its complexity and networking limitations have hindered widespread use [12]. Along with that, tools like Sigstore can strengthen supply chain integrity but are sometimes viewed as barriers in rapid development environments, such as those in startups or immediate product based deployments [18]. Studies such as those by Kumar and Nandakumar [22] show that design shortcuts taken for the sake of convenience can accumulate very significant security debt over time. As a result, balancing strong security controls with developer productivity continues to pose a substantial, yet predictable challenge.

To advance the field, a more unified approach is needed—one that connects tool-specific vulnerabilities, addresses fragmented mitigation practices, and acknowledges the tradeoffs developers across all facets of the development life-cycle face. This paper proposes an integrated security framework designed to support both secure and efficient cloud-native development, from local environments to production deployments.

III. Significance

This paper has thus far, established a clear research gap that is very much an issue to tackle in the current development landscape. Thus, this research responds to a clear research gap in the existing literature: the rapid rise of cloud-native development introduces both new possibilities and new challenges for secure software engineering. Its significance lies in identifying vulnerabilities that are recursive in nature, within hybrid development-to-deployment workflows, and in rethinking security debt not only as an isolated flaw for one stage of the development cycle, but as an evolving byproduct of interconnected systems. The study contributes meaningfully to both theory and practice by offering new ways to conceptualize, monitor, and mitigate these risks across modern software pipelines.

A. Theoretical Significance

This paper expands it's significance in a two pronged approach - from a theoretical and practical perspective. Hence, from a theoretical perspective, this study contributes to the understanding and improvement of security debt. It does this by looking at security debt as a hybrid phenomenon that spans local development tools and managed cloud services. Current literature states that existing models often treat technical debt and security debt independently within isolated phases of the software development lifecycle [23]. More often than not, it neglects how vulnerabilities introduced in early stages propagate through continuous integration and deployment pipelines. By providing a deeper insight into these topics and combining various perspectives and views from technical debt research [22], cloud architecture risks [3], and supply chain security literature [8], this paper proposes a clean and efficient unified taxonomy categorizing vulnerabilities into privilege escalation, supply chain compromise, and ephemeral misconfiguration.

In addition to the above prong, this study challenges some of the core assumptions behind the "shift-left" security model that is used during the development cycle. Although the shift-left approach promotes early mitigation of vulnerabilities [6], real-world incidents like the Azurescape breach [14] reveal the limitations of localized fixes when broader systemic risks are overlooked. Instead, this paper advocates for a high-level

perspective that sees security not as a one-time intervention, but as a continuous thread throughout the entire development and deployment cycle. This should be paramount during the development lifecycle and this paper shows theoretically why that is so.

Another important theoretical advancement is the positioning of security debt within a dynamic, operational context within any organization. This paper posits that rather than viewing this as a fixed technical issue, we should build more research and frameworks based on emerging studies treating any form of security debt as more of a dynamic entity, essentially to assume that this specified debt evolves over time and across the development stack and this is due to many factors that influence this debt including continuous and rapid deployments, scaling activities to the infrastructure and modifications to the existing architecture [26]. This framing encourages future researchers to explore adaptive models of security debt monitoring and mitigation in cloud-native ecosystems.

B. Practical Significance

Now from the second prong, this paper explores the same research question from a practical standpoint. It offers guidance and actionable strategies for members of the development team across all kinds of organizations. For developers, it provides detailed guidelines on minimizing privilege escalation risks during containerization by adopting rootless container modes [12] and reducing dependency on elevated permissions. These developers should also integrate supply chain security practices such as artifact signing via Sigstore [18] into their daily workflows without disrupting development processes.

For DevOps teams, this work synthesizes best practices across policy-as-code enforcement (e.g., Open Policy Agent) [19], continuous vulnerability scanning (e.g., Trivy) [20], and runtime anomaly detection (e.g., Falco) [21], illustrating how these tools can be orchestrated cohesively across CI/CD pipelines. It emphasizes the need to coordinate these tools effectively within CI/CD processes, instead of treating security validation and testing as a separate task. This cohesion is critical, as it reduces the likelihood of the system drifting from its intended configuration. It also ensures that potential risks have a higher chance of being caught early.

Similarly, platform providers and tool maintainers have an equally significant role. As stated earlier in this paper, if these platform providers were to provide secure-by-default configurations, it would be beneficial to development teams and organizations in need of robustness, as it lowers the need for additional security implementations if the default ones are effective at lower levels of the stack. These configurations could be as simple as enforcing minimal Identity and Access Management (IAM) mechanisms for access control or validating deployment policies—both of which reduce the need for large-scale security audits at the end of the development lifecycle. Based on relevant literature on cloud elasticity and resource optimization [2], this study also calls for platforms to adopt intelligent defaults, including smarter business continuity plans for the products they offer. This reduces the burden on end-users and customers and promotes stronger guarantees for security.

To sum up, these contributions bridge the gap between theoretical understanding and practical application. By addressing both how vulnerabilities spread and why mitigation efforts often fall short, this study lays out a comprehensive framework for building more resilient cloud-native systems—ones that do not sacrifice scalability and agility for the sake of security, but instead align all three in a sustainable and forward-looking way.

IV. Methodology

The methodology section highlights the analysis approach this paper takes. This research takes a qualitative approach, combining a structured review of relevant literature with thematic analysis and critical evaluation. The overall goal is to bring together different perspectives on vulnerabilities that are prevalent in hybrid cloud-native deployments, with a specific focus on those that emerge across the development and deployment spectrum. This will help this study and future literature to assess how current mitigation tools hold up under real-world conditions.

A. Literature Review Scope

The first step in this detailed methodology involved reviewing published research on cloud-native security, technical debt, DevSecOps, and hybrid cloud risk management. To achieve this goal, relevant literature was sourced from peer-reviewed venues like IEEE, ACM, and USENIX, along with well-regarded industry reports (e.g., Sysdig [12], Cloud Security Alliance [5]).

The selection criteria involved prioritizing works published recently, between 2015 to 2025, especially studies that addressed container security, supply chain threats, and the complexities of securing hybrid environments. This study also used multiple real-world attacks and malware scenarios, such as Azurescape [14], as well as publicly available Common Vulnerabilities and Exposures (CVEs), such as CVE-2024-1753 [13], that provided a grounded approach to the theoretical and practical analysis this paper aimed to complete.

B. Analysis Approach

After gathering the relevant literature, a thematic analysis was carried out to highlight common patterns across different works. This helped analyze and examine recurring issues, like privilege escalation, insecure supply chains, and misconfiguration, that tend to crop up across the software development life-cycle. These patterns were then grouped into broader categories to clarify how they contribute to accumulating security debt.

Alongside this, the paper also takes a closer look at several widely used mitigation tools. Open Policy Agent (OPA) [19], Trivy [20], Falco [21], and Sigstore [18] were evaluated for their strengths and weaknesses—particularly how well they integrate into CI/CD pipelines and whether they create

usability trade-offs for developers. This part of the analysis was based on a mix of official documentation, real-world use cases, and feedback from the broader developer community.

Rather than conducting primary experiments, the analysis focused on identifying strengths, limitations, and gaps across existing security models, thereby informing strategic recommendations for improving hybrid cloud-native security postures.

C. Expected Contributions

This research delivers three major contributions aligned with its abstract and research statement.

First, it develops a structured analysis of how vulnerabilities exist and propagate across hybrid cloud-native workflows. By systematically analyzing how privilege escalation risks, supply chain compromises, and ephemeral misconfigurations manifest and evolve from development to production stages, the study provides a more in-depth and connected understanding of security debt.

Second, it critically examines current mitigation tools and strategies, such as the aforementioned OPA, Trivy, Falco and Sigstore, highlighting not just their technical strengths, but also the challenges developers face when trying to use them in complex, fast-moving environments.

Thirdly, the research proposes realistic ways to improve and hopefully balance security validation and testing in cloud-native settings without slowing down development. The goal is to find a balance that can address both the technical and human factors influencing the management of security debt.

Essentially, these contributions help provide a cohesive contribution including theory and practice, offering both a high-level view and grounded insights into how cloud-native systems can be made more secure and resilient.

V. Discussion

This section synthesizes critical insights into vulnerability propagation across hybrid cloud-native workflows, evaluates the limitations of current mitigation strategies, highlights integration challenges, and proposes strategic directions for enhancing security resilience without diminishing developer agility.

In the discussion section, the paper discusses key and critical insights on the propagation of vulnerabilities through hybrid cloud-native workflows. It also adds important reflections on the limitations of current mitigation tools and highlights integration challenges. This section goes further and suggests practical directions for improving security resilience without the negative aspect of slowing down developer productivity.

A. Analysis of Vulnerability Propagation

In the current cloud landscape that is prevalent in DevOps teams, hybrid cloud-native workflows show that vulnerabilities propagate in patterns that arise due to the rapid movement of artifacts, configurations, and credentials from local development to deployment. For example, misconfigured containers, such as those running with elevated privileges during development, often bypass deep security validation and qualitative analysis before being deployed in cloud infrastructures [3].

Cases like the Azurescape incident [14] show how weaknesses introduced early—at the container or virtual machine level—can lead to larger issues in shared production environments, especially where multitenancy increases exposure. The complexity of modern cloud services, where tenants share virtualized infrastructure layers, amplifies the impact of a single insecure artifact reaching production.

Another added layer of security risks involves supply chain risks. Unverified dependencies or misconfigured Infrastructure-as-Code (IaC) templates can introduce vulnerabilities early in the pipeline [8]. Once a compromised dependency is incorporated into a base image, it often remains undetected across successive deployment iterations. Over time, these overlooked flaws accumulate as security debt [28], making systems increasingly fragile. The danger here is that this debt often stays invisible until it results in a major failure, or a critical failure point in production environments.

B. Critical Evaluation of Mitigation Tools

Current mitigation tools offer valuable point solutions but remain insufficient for ensuring comprehensive hybrid security resilience. Open Policy Agent (OPA) enables declarative policy enforcement during build-time and deployment stages [19], but making and implementing detailed policies, rules, frameworks that account for multiple can be very resource heavy.

Trivy provides effective vulnerability scanning for containers and Infrastructure-as-Code templates [20], but it remains limited to known vulnerabilities, failing to detect novel attack vectors or runtime misbehaviors. Furthermore, frequent updates to vulnerability databases are required to maintain accuracy, introducing additional operational burdens on security teams.

Falco's eBPF-based runtime detection [21] allows for monitoring anomalous container behaviors, but its effectiveness is not entirely reactive, often identifying attacks after the initial propagation of said attacks. It also requires finely tuned rule sets to avoid generating excessive false positives, a problem that can overwhelm operational security teams if improperly configured.

Sigstore addresses supply chain risks through artifact signing and transparency logs [18], but operational adoption in real-time has been slow in key management workflows. Developers often view this process of setup and signing artifacts as complicated and disrupting workflow and already existing practices that do not need updating without an excessive need for it in the near future.

Overall, while each tool addresses specific risks, they're often adopted in isolation. This fragmented approach leaves parts of the CI/CD process unmonitored, allowing vulnerabilities to slip through unnoticed.

C. Integration Challenges in Hybrid Workflows

The integration of all these security tools and recommendations into a single, cohesive plan of action in the regular pipeline remains a major challenge from a technical and cultural standpoint. The existence of CI/CD pipelines is still developing as we go and the security of those pipelines is very much the same. These workflows still treat security as a gatekeeping step, such as running static scans during builds, instead of enforcing validation continuously at every stage of the pipeline. This often results in artifacts being validated once and then ignored, even as they move into production. In many cases, security checks are skipped entirely under deadline pressure.

Studies have shown that artifacts signed during early builds often bypass validation at deployment or runtime stages due to pipeline drift and incomplete enforcement policies [12]. Moreover, drift frequently occurs when developers create ad hoc deployment templates or override default configurations under deadline pressures, introducing new attack surfaces not present during initial build-time validation.

However, beyond these tools and their technicalities, another prevalent issue is the organizational and cultural barriers in their adoption. These tools are seen by teams to be disruptive in nature, taking more time for the whole development cycle. An example of this is how rootless containers or artifact signing slow down their tasks and computational load is increased [12]. As such, there arises a higher need for user-centric and usability-centered design of these security tools, that do not overburden resources - time and money. This results in a constant struggle for developer teams to maintain a balance between speed and safety in cloud-native engineering currently.

D. Strategic Directions for Future Improvements

As concluded from the previous views and considerations, to effectively reduce security debt in hybrid cloud-native systems, we need to shift from fragmented, reactive fixes to ongoing, integrated security efforts.

First, preventive, detective, and integrity-preserving measures must be embedded continuously throughout the software delivery lifecycle. Policy enforcement via tools like OPA should extend beyond static validations to runtime state inspections, while vulnerability scanning through solutions like Trivy should incorporate dynamic threat modeling updates to capture emerging risks in near real-time. Furthermore, CI/CD pipelines should integrate automatic signature verification at each artifact promotion stage, enforcing supply chain trustworthiness without developer intervention.

Second, mitigation tooling must evolve to prioritize usability alongside security robustness. Research suggests that adopting developer-centered design methodologies can substantially improve the operational adoption rates of security practices without diminishing developer experience [22]. Simplified policy generation, seamless artifact signing workflows, and frictionless vulnerability scanning interfaces are essential for achieving sustainable security adoption. Security should be embedded as an invisible, low-friction part of the default development workflow rather than an explicit afterthought.

Lastly, future work should focus on real-time monitoring and dynamic models of security debt. Instead of treating security debt as a fixed problem, organizations could track it over time using telemetry and predictive analytics [26]. Machine learning-driven monitoring tools could help teams detect early signs of risk accumulation and take action before issues escalate.

Through these strategic improvements, it becomes possible to build cloud-native ecosystems that are not only scalable and agile but are constantly evolving to be resilient against evolving threat landscapes.

VI. Conclusion and Future Work

A. Conclusion

The adoption of cloud-native development tools and managed services has revolutionized software engineering workflows but has simultaneously introduced complex and systemic security risks. This study critically synthesized how vulnerabilities propagate from local development artifacts into cloud production deployments, amplifying systemic security debt through privilege escalation vectors, supply chain compromises, and ephemeral misconfigurations.

Through a critical evaluation of contemporary mitigation strategies such as Open Policy Agent, Falco, Trivy, and Sigstore, the research highlighted significant gaps in preventive, detective, and integrity-preserving controls when applied across hybrid workflows. While these tools provide essential defensive capabilities, their fragmented adoption, usability challenges, and reactive operational models allow security debt to accumulate invisibly until exploited.

The study also underscored how fragmented validation and the lack of seamless integration across CI/CD pipelines contribute to the accumulation of security debt. Organizational habits and development culture—such as prioritizing speed over scrutiny—make it harder to enforce consistent security practices. Without embedding continuous validation and making tools more accessible to developers, cloud-native environments will continue to face elevated security risks.

Finally this paper proposes strategic recommendations for addressing these challenges. By focusing more on continuous validation, improving tool usability for multiple user-centric approaches and treating security debt as a dynamic, measurable risk factor, teams can adopt more security resilient development practices without sacrificing speed, time and other resources.

B. Future Work

Future research must prioritize the development of integrated hybrid security pipelines that embed preventive, detective, and trust validation controls as continuous, low-friction processes rather than isolated checkpoints. Practical studies exploring the operationalization of policy-as-code at runtime, dynamic anomaly detection pipelines, and continuous artifact

trust validation in cloud-native CI/CD environments are essential to advance this vision.

Moreover, investigating developer-centric usability enhancements in security tooling presents a promising direction. Empirical studies quantifying how simplified workflows, invisible validation, and low-friction artifact signing influence developer adoption rates would bridge the gap between security engineering and operational practices.

Finally, advancing dynamic security debt modeling remains an open research challenge. Predictive telemetry systems capable of quantifying debt accumulation, detecting drift, and forecasting risk trajectories in hybrid workflows would enable organizations to proactively manage security debt as a first-class operational metric.

By aligning future research toward continuous resilience engineering, usability-driven adoption models, and proactive debt monitoring, the next generation of cloud-native security strategies can achieve sustainable scalability without sacrificing security integrity.

References

[1] J. Anderson, "Cloud-native security: Challenges and strategies," in *Proceedings of the 14th USENIX Conference on Cloud Computing (USENIX Cloud)*, 2017.

[2] C. Mera-Gómez, F. Ramírez, R. Bahsoon, and R. Buyya, "A Debt-Aware Learning Approach for Resource Adaptations in Cloud Elasticity Management," in *Service-Oriented Computing. ICSOC 2017*, Lecture Notes in Computer Science, vol. 10601, Springer, Cham, 2017.

[3] D. Bernstein and J. Vij, "Security risks in container orchestration frameworks," *IEEE Cloud Computing*, vol. 6, no. 5, pp. 34–41, 2019.

[4] T. Garfinkel and M. Rosenblum, "When virtual is harder than real: Security challenges in virtual machine-based computing environments," in *Proceedings of the 10th USENIX Security Symposium*, 2015.

[5] Cloud Security Alliance, "Top Threats to Cloud Computing: Egregious Eleven," CSA, 2020. [Online]. Available: https://cloudsecurityalliance.org

[6] S. Sharma and H. Singh, "DevSecOps: Integrating security into DevOps," *International Journal of Computer Applications*, vol. 975, no. 8887, pp. 12–18, 2020.

[7] B. Burns, B. Grant, D. Oppenheimer, E. Brewer and J. Wilkes, "Borg, Omega, and Kubernetes," *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.

[8] S. Kim and H. Lee, "Supply chain security in cloud environments: Challenges and opportunities," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–34, 2022.

[9] H. Raj, S. Rajamani, and M. Schunter, "Towards secure container orchestration," in *Proceedings of the 30th USENIX Security Symposium*, 2021.

[10] I. Mavridis and S. Karatza, "Security risks and countermeasures in container-based cloud infrastructures," *Future Generation Computer Systems*, vol. 100, pp. 837–846, 2020.

[11] A. Kontorovich, "Security threats in Docker and Kubernetes environments," *Black Hat USA*, 2016.

[12] Sysdig, "2024 Cloud-Native Security and Usage Report," Sysdig, 2024. [Online]. Available: https://sysdig.com/press-releases/sysdig-2024-usage-report/

[13] GitHub Security Advisory, "Podman - CVE-2024-1753: Container Escape at Build Time," GitHub, Mar. 18, 2024. [Online]. Available: https://github.com/containers/podman/security/advisories/GHSA-874v-pj72-92f3

[14] TechTarget, "Azurescape: New Azure Vulnerability Fixed by Microsoft," TechTarget, Sep. 9, 2021. [Online]. Available: https://www.techtarget.com/searchsecurity/news/252506509/Azurescape-New-Azure-vulnerability-fixed-by-Microsoft

[15] Cyscale, "AWS Cloud Security and Compliance," Cyscale, 2024. [Online]. Available: https://cyscale.com/use-cases/aws-cloud-security/

[16] Quorum Cyber, "Docker Authorization Bypass Vulnerability," Quorum Cyber, Jul. 25, 2024. [Online]. Available: https://www.quorumcyber.com/threat-intelligence/docker-authorisation-bypassvulnerability/

[17] CRN, "Google Cloud Run Is 'Being Abused' by Cyberattacks via Microsoft Installers," CRN, Feb. 22, 2024. [Online]Available:

[18] Sigstore, "Sigstore: Overview," [Online]. Available: https://docs.sigstore.dev/about/overview/

[19] Spacelift, "What Is OPA? Open Policy Agent Examples & Tutorial," Spacelift, Feb. 5, 2025. [Online]. Available: https://spacelift.io/blog/what-is-open-policy-agent-and-how-it-works

[20] Aqua Security, "Trivy: Vulnerability Scanner," Aqua Security, 2024. [Online]. Available: https://aquasecurity.github.io/trivy/v0.17.2/

[21] Falco, "Falco: Runtime Security Tool," Falco, Feb. 11, 2025. [Online]. Available: https://falco.org/

[22] M. Kumar and A. N. Nandakumar, "Exploring Multilateral Cloud Computing Security Architectural Design Debt in Terms of Technical Debt," in *Smart Computing and Informatics*, Smart Innovation, Systems and Technologies, vol. 78, Springer, Singapore, 2018.

[23] E. Alzaghoul and R. Bahsoon, "Evaluating Technical Debt in Cloud-Based Architectures Using Real Options," in *23rd Australian Software Engineering Conference (ASWEC)*, 2014, pp. 1–10.

[24] S. Kumar, R. Bahsoon, T. Chen and R. Buyya, "Identifying and Estimating Technical Debt for Service Composition in SaaS Cloud," in *2019 IEEE International Conference on Web Services (ICWS)*, 2019, pp. 121–125.

[25] E. F. A. Alzaghoul, "Value-and Debt-Aware Selection and Composition in Cloud-Based Service-Oriented Architectures Using Real Options," Ph.D. dissertation, University of Birmingham, 2015.

[26] G. Skourletopoulos et al., "Elasticity Debt Analytics Exploitation for Green Mobile Cloud Computing: An Equilibrium Model," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 1, pp. 122–131, Mar. 2019.

[27] E. Alzaghoul and R. Bahsoon, "CloudMTD: Using Real Options to Manage Technical Debt in Cloud-Based Service Selection," in *4th International Workshop on Managing Technical Debt (MTD)*, 2013, pp. 55–62.

[28] M. M. Kruke, A. Martini, D. S. Cruzes, and M. Iovan, "Defining Security Debt: A Case Study Based on Practice," in *Product-Focused Software Process Improvement. PROFES 2024*, Lecture Notes in Computer Science, vol. 15452, Springer, Cham, 2025.