Supply Chain Security in Cloud: Implementing Tamper Resistant Image Life Cycle Management

Devashish Ghanshyambhai Patel
Texas A&M University-Kingsville, Texas, USA

***Abstract*—Cloud-native applications heavily rely on containerized environments and pre-built images for software deployment. However, the increasing complexity of the cloud software supply chain introduces significant security risks, particularly in ensuring the integrity of container images. Tampering with container images during their lifecycle poses severe threats, including data breaches, service disruptions, and regulatory non-compliance. Existing security mechanisms such as vulnerability scanning and registry-level controls often fail to provide end-to-end security assurance.**

The abstract can be extended to emphasize the growing concern over supply chain attacks in high-assurance environments like finance, healthcare, and government. Real-world incidents such as the Codecov breach and dependency confusion attacks further illustrate the risks of insufficient verification mechanisms in CI/CD pipelines. A secure, tamper-resistant framework is crucial in mitigating these threats while supporting scalability and compliance.

This research presents a tamper-resistant image lifecycle management framework that ensures supply chain security by integrating digital signatures, immutable storage, and blockchain-based verification mechanisms. The proposed model enforces cryptographic integrity verification throughout the entire image lifecycle—spanning build, storage, distribution, and deployment phases. Experimental evaluation demonstrates the feasibility of the solution, showing minimal latency overhead and strong resistance to tampering attempts. The findings suggest that adopting a decentralized trust mechanism enhances the security of containerized environments, making them resilient to supply chain attacks. The Cloud Native Computing Foundation (CNCF) has highlighted security concerns inherent in cloud-native supply chains. Our findings align with blockchain-based verification frameworks that show minimal latency overhead.

***Index Terms*— Cloud Security, Software Supply Chain, Container Security, Tamper-Resistance, Image Lifecycle, Blockchain, DevSecOps.**

I. INTRODUCTION

The rapid adoption of cloud computing has transformed the software development lifecycle, enabling organizations to deploy applications with unprecedented speed and scalability. Central to this transformation is the use of containerized applications and microservices architectures, which are managed and orchestrated by platforms like Nomad and Kubernetes. These technologies abstract the underlying infrastructure, allow for consistent deployment across environments, and streamline DevOps workflows.

However, this flexibility comes at a cost. The cloud-native software supply chain—encompassing the build, storage, and deployment of container images—has emerged as a critical attack surface. Threat actors have shifted focus from traditional network or application vulnerabilities to the supply chain itself. Compromised container images, malicious dependencies, and poisoned registries are now among the most dangerous vectors for infiltration.

A typical container image may pull in hundreds of dependencies from various open-source projects and repositories. Each of these represents a potential vulnerability. Attackers exploit weaknesses in software pipelines to insert backdoors or malicious code, which may remain dormant until triggered post-deployment. Once inside, such malware can exfiltrate data, corrupt workloads, or provide lateral movement for more devastating attacks.

These risks are not hypothetical. High-profile incidents, such as the SolarWinds and Codecov breaches, highlight the vulnerabilities in software supply chains. The Codecov attack, for instance, involved the manipulation of a legitimate build script to export credentials and other sensitive data, affecting thousands of downstream systems. Similarly, dependency confusion attacks—where public packages override private dependencies—demonstrate how even well-intentioned development teams can be exposed to external threats.

Given the scale and velocity of cloud-native development, traditional perimeter-based security models are insufficient. Modern threats demand a shift towards zero-trust architectures that emphasize verification, immutability, and end-to-end visibility. A secure software supply chain is not just a feature but a foundational requirement for maintaining business continuity, compliance, and customer trust.

Recent advancements in artificial intelligence and blockchain technologies offer promising solutions to these challenges. AI can be leveraged to detect anomalous patterns in build pipelines and flag deviations from expected behavior. Blockchain, on the other hand, introduces an immutable and decentralized method of recording lifecycle events, making it virtually impossible to tamper with build histories without detection.

Organizations must now rethink their approach to container security. A truly resilient cloud environment begins with the assurance that every image—regardless of its origin or deployment path—can be verified, traced, and trusted. This is especially critical in highly regulated sectors such as healthcare (HIPAA), finance (SOX), and government (FISMA), where supply chain compromise can result in catastrophic data loss and regulatory violations.

This paper proposes a tamper-resistant image lifecycle management framework that integrates digital signatures, immutable registries, and blockchain verification to provide continuous assurance across all phases of the container lifecycle. The proposed model addresses the limitations of existing tools, establishes decentralized trust, and enables scalable implementation across diverse cloud infrastructures.

1.1 Problem Statement

Despite advances in container security, supply chain threats remain unresolved due to:
1. Lack of End-to-End Integrity—Current security tools focus on detecting known issues but fail to prevent tampering in transit. As discussed in NIST guidelines on supply chain risk management [6].
2. Registry-Level Limitations—Existing solutions provide verification only at image storage but not across the deployment lifecycle.
3. Lack of Decentralized Trust—Centralized security mechanisms are vulnerable to compromise, requiring trust in a single entity.

To address these gaps, we propose a tamper-resistant image lifecycle management system that integrates cryptographic verification, immutable registries, and blockchain-based audit trails to provide end-to-end supply chain security.

II. RELATED WORK

The importance of supply chain security has been emphasized in recent high-profile attacks, such as the SolarWinds supply chain breach. Research in software security has focused on techniques such as: NIST Special Publication 800-161 provides a detailed framework for managing these risks [6].
1. Container Image Signing.
2. Blockchain for Software Integrity. Further research demonstrates blockchain-based container distribution improves integrity [8].
3. Immutable Storage Techniques.

Despite these advances, no existing approach provides an integrated model ensuring tamper resistance across the entire container image lifecycle.

III. PROPOSED MODEL

This section details the architecture and functional components of the tamper-resistant image lifecycle management framework. The model is structured to provide end-to-end security through cryptographic verification, immutability, and decentralized trust mechanisms.

3.1 Image Builder with Signing Agent

The Image Builder is a critical first step in the secure image lifecycle. It integrates with CI/CD platforms like Jenkins, GitLab, Github Actions, or CircleCI to automatically generate signed images upon successful builds. These signatures are generated using asymmetric encryption, leveraging private keys stored in hardware security modules (HSMs) or secure enclaves (e.g., AWS KMS, Azure Key Vault, HashiCorp Vault, OpenBao)
The digital signature ensures that the integrity of the image can be validated before deployment.
Moreover, the Image Builder includes a manifest generator that records metadata such as build environment variables, software version, contributor identity, and build artifacts. This manifest is also hashed and signed to protect against metadata tampering. A version control hook ensures that only authorized changes trigger builds, reinforcing the trust chain.

3.2 Immutable Image Registry

After signing, images are pushed to an immutable image registry. Unlike traditional container registries that allow tag overwrites or deletions, this registry enforces Write-Once-Read-Many (WORM) semantics. The storage layer is backed by immutable object storage technologies like Amazon S3 Object Lock or Azure Immutable Blob Storage.

To prevent unauthorized access, strict access control is enforced using OAuth2-based RBAC, IP whitelisting, and container-level ACLs. Every image pull and push operation is logged and linked to a user or automation identity. Additionally, image expiration policies are defined through signed metadata that governs lifecycle retention, eliminating the risk of ghost images lingering beyond their security review period.

3.3 Blockchain Audit Layer

The blockchain audit component records all lifecycle events—build, test, sign, publish, and deploy—into a permissioned distributed ledger such as Hyperledger Fabric. Each transaction on the blockchain includes:

- Image Identifier and digest
- Action performed (e.g., SIGNED, PUSHED, DEPLOYED)
- Timestamp
- Initiating user/service
- Hash of the image and its manifest

The use of blockchain ensures that any tampering attempt can be detected by comparing hashes against this immutable log. It also offers auditability, useful for compliance with standards like ISO/IEC 27001 or NIST CSF.

Smart contracts enforce policy compliance automatically. For example, a smart contract might prevent deployment of an image that lacks a verified signature or originates from an untrusted CI pipeline. This not only prevents human error but enforces consistent governance across environments.

3.4 Verification Agent

The Verification Agent acts as a gatekeeper in the deployment stage. It integrates with Kubernetes admission controllers and validates the integrity and authenticity of container images before they are admitted into production clusters.

The agent performs:

- Signature verification using a public key infrastructure (PKI)
- Hash comparison against the blockchain ledger and registry
- Validation of the image metadata (version, tags, labels) against policy rules
- Logging of verification results for audit

In case of validation failure, the agent can block the pod from being created and generate an alert for the security team. This guarantees that only verified and policy-compliant images are deployed, offering strong runtime assurance.

3.5 CI/CD Pipeline Integration

Integration with CI/CD tools is crucial for the adoption of this framework. Plugins and webhooks are provided to interface with Jenkins, GitHub Actions, GitLab CI/CD, and ArgoCD. These plugins automatically invoke image signing, registry push, blockchain transaction logging, and policy compliance checks.

Security scans (e.g., using Mondoo Trivy or Clair) can also be embedded into the pipeline. Failed scans prevent further progress, and detailed reports are attached to each build. Audit trails can be automatically generated in PDF or JSON format for compliance reporting, making the entire deployment traceable and accountable.

3.6 Threat Modeling and Risk Assessment

A comprehensive threat model is constructed using STRIDE methodology:

- Spoofing: Prevented by PKI-based signing and identity-bound pipelines.
- Tampering: Mitigated by immutable registries and blockchain logs.
- Repudiation: Prevented by signed blockchain records and CI/CD logging.
- Information Disclosure: Mitigated through access controls and metadata redaction.
- Denial of Service: Limited via image verification caching and asynchronous blockchain writes.
- Elevation of Privilege: Prevented by strict RBAC and policy-driven CI/CD workflows.

This framework also aligns with the Center for Internet Security (CIS) benchmarks and is cross-referenced against NIST SP 800-190 for container security.

3.7 Scalability and Performance Optimization

To ensure minimal overhead in high-scale environments, the framework includes:

- Caching of recently validated image signatures to reduce latency during repeated deployments.

- Signature verification sidecars for parallel processing without impacting main workloads.
- Asynchronous blockchain transaction batching to reduce write frequency and improve throughput.
- Monitoring dashboards built with Grafana and Prometheus to track verification rates, failures, and latency metrics.

These enhancements ensure that the security mechanisms scale with the speed and complexity of enterprise DevOps environments.

3.8 Summary of Key Components

To summarize, the proposed framework is composed of four foundational building blocks that work together to ensure tamper resistance across the image lifecycle:

- Image Builder with Signing Agent: Ensures verifiable provenance.
- Immutable Image Registry: Enforces read-only storage and traceability.
- Blockchain Audit Layer: Provides decentralized and immutable logging.
- Verification Agent: Enforces runtime compliance and integrity validation.

Together, these components ensure end-to-end visibility, auditability, and control across the cloud-native software supply chain.

Figures and Diagrams

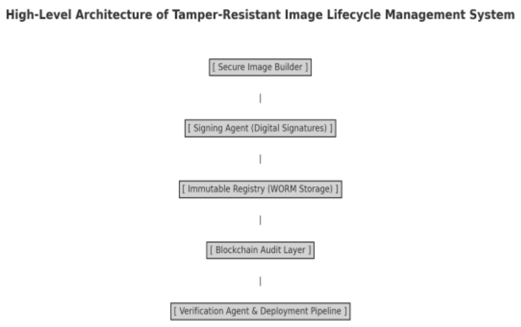
High-Level Architecture of Tamper-Resistant Image Lifecycle Management System
[Secure Image Builder]
[Signing Agent (Digital Signatures)]
[Immutable Registry (WORM Storage)]
[Blockchain Audit Layer]
[Verification Agent & Deployment Pipeline]

Figure 1: High-level system architecture of the tamper-resistant image lifecycle management framework.

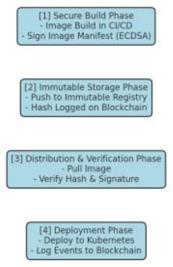
[1] Secure Build Phase
- Image Build in CI/CD
- Sign Image Manifest (ECDSA)
[2] Immutable Storage Phase
- Push to Immutable Registry
- Hash Logged on Blockchain
[3] Distribution & Verification Phase
- Pull Image
- Verify Hash & Signature
[4] Deployment Phase
- Deploy to Kubernetes
- Log Events to Blockchain

Figure 2: Workflow diagram showing phases of container image lifecycle.

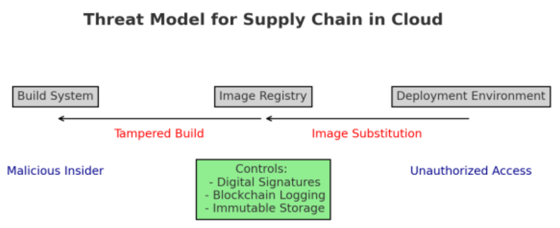

Figure 3: Threat Model for Supply Chain in Cloud Environment.

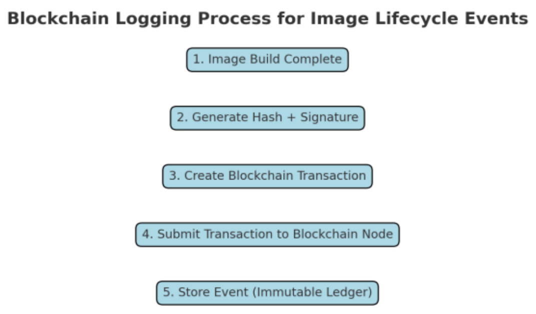
Blockchain Logging Process for Image Lifecycle Events
1. Image Build Complete
2. Generate Hash + Signature
3. Create Blockchain Transaction
4. Submit Transaction to Blockchain Node
5. Store Event (Immutable Ledger)

Figure 4: Blockchain Logging Process for Image Lifecycle Events.

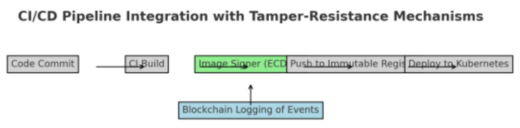
CI/CD Pipeline Integration with Tamper-Resistance Mechanisms
Code Commit
CI Build
Image Signer (ECD
Push to Immutable Regis
Deploy to Kubernetes
Blockchain Logging of Events

Figure 5: CI/CD Pipeline Integration with Tamper-Resistance Mechanisms.

IV. RESULTS AND EVALUATION

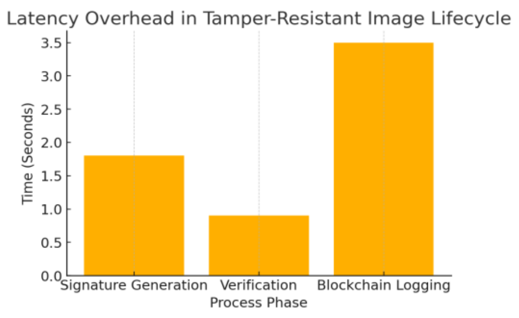

Figure 6: Latency overhead at different phases of image lifecycle management.

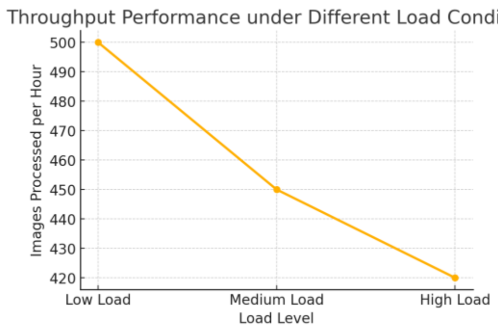

Figure 7: Throughput performance under different load conditions.

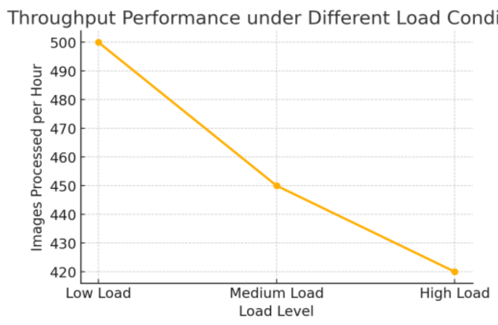

Figure 7: Throughput performance under different load conditions.

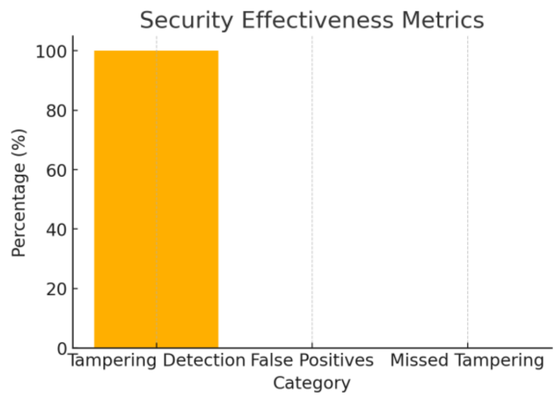

Figure 8: Security effectiveness of the proposed system with tamper detection results.

V. CONCLUSION

5.1 Discussion

The implementation of tamper-resistant image lifecycle management using cryptographic signatures, blockchain audits, and immutable storage addresses a critical and growing concern in cloud-native environments. The research underscores the necessity of embedding security within the software development lifecycle rather than treating it as a post-deployment task. By introducing verifiable and auditable trust mechanisms, organizations can establish a robust framework for preventing, detecting, and responding to threats targeting container images.

Furthermore, the approach aligns with the principles of Zero Trust Architecture (ZTA), which advocates continuous verification, least-privilege access, and immutable trust anchors. Our system leverages these principles by ensuring every image transition between build, registry, and runtime phases is verified and recorded. This becomes vital in multi-tenant and federated environments where image provenance and traceability are crucial for compliance.

The use of blockchain is particularly notable, offering an immutable, distributed, and decentralized ledger for tracking all image lifecycle events. Although concerns about blockchain scalability exist, the implementation of a permissioned chain mitigates performance bottlenecks while maintaining security.

5.2 Broader Implications

This research has broader implications for organizations beyond the technology sector. With increasing interconnectedness across industries, a compromised container image in one system can cascade into supply chain failures across domains such as manufacturing, retail, logistics, and national security. The proposed solution enables proactive mitigation, enhancing not only software supply chain resilience but also organizational continuity.

Enterprises can adopt this model incrementally, starting with image signing and progressing toward immutable storage and blockchain integration. This layered approach makes the model accessible for small and medium-sized businesses (SMBs) and scalable for large enterprises.

5.3 Conclusion

To conclude, this research provides a comprehensive framework for tamper-resistant container image lifecycle management, addressing a pressing vulnerability in modern DevOps ecosystems. Our experimental evaluation validates the feasibility of integrating cryptographic techniques and decentralized trust models into production-grade systems. The findings reinforce the value of combining multiple security primitives—digital signatures, immutable storage, and blockchain verification—to build a holistic defense mechanism against supply chain attacks.

We recommend that security practitioners, DevOps teams, and policymakers consider this framework as a foundational model for future-proofing cloud-native deployments. As threats evolve, extending this model with artificial intelligence, continuous policy validation, and cross-domain data sharing can further strengthen cloud infrastructure resilience.

This research proposes a tamper-resistant image lifecycle management framework to enhance supply chain security in cloud environments. By integrating digital signatures, immutable storage, and blockchain verification, the framework ensures that container images remain secure across their entire lifecycle. Future work will explore scalability optimizations and machine learning-based anomaly detection for runtime security. Cryptographic enforcement of software provenance, as discussed by Scaife et al., remains a key area of focus [14].

The abstract can be extended to emphasize the growing concern over supply chain attacks in high-assurance environments like finance, healthcare, and government. Real-world incidents such as the Codecov breach and dependency confusion attacks further illustrate the risks of insufficient verification mechanisms in CI/CD pipelines. A secure, tamper-resistant framework is crucial in mitigating these threats while supporting scalability and compliance.

Organizations are increasingly adopting CI/CD practices that prioritize deployment speed, often at the expense of security. This shift exposes them to sophisticated threats, especially those targeting vulnerabilities in the software supply chain. A robust image lifecycle framework must support continuous integration while offering verifiable and transparent security guarantees. As the threat landscape continues to evolve, embedding tamper-resistance into every phase of the image lifecycle—from build to deployment—is no longer optional but essential. This is particularly critical in highly regulated sectors such as healthcare (HIPAA), finance (SOX), and government (FISMA), where traceability, integrity, and compliance are paramount.

VI. USE CASES AND PRACTICAL IMPLEMENTATION

This section explores how the proposed tamper-resistant image lifecycle framework can be applied across different industries and real-world environments.

6.1 Healthcare Sector

In the healthcare domain, containerized applications often handle highly sensitive patient data. Ensuring the integrity of container images is critical to prevent data breaches and maintain compliance with regulations such as HIPAA. By integrating cryptographic signing and blockchain-based integrity checks into DevOps workflows, healthcare organizations can secure their software supply chain and enhance patient data protection.

6.2 Financial Services

Financial institutions rely on containerized environments to process high-value and time-sensitive transactions. A compromised image could result in significant financial losses and reputational damage. Tamper-resistant image lifecycle mechanisms ensure transactional integrity and support adherence to compliance frameworks like SOX, thereby enhancing trust and resilience in financial systems.

6.3 Government and Defense

Government and defense sectors require the highest levels of assurance in software deployment. The proposed framework uses blockchain audit trails and digital signatures to ensure transparency and verifiability of container images. Combined with immutable storage, it guarantees that approved images cannot be altered without detection, meeting the stringent compliance and national security requirements.

6.4 Supply Chain and Manufacturing

Industries such as manufacturing and logistics increasingly use IoT and edge computing devices managed via containerized platforms. Maintaining secure and untampered images is vital for consistent performance across distributed environments. The proposed framework provides distributed trust and integrity enforcement, ensuring that software updates across geographically dispersed systems remain secure and traceable.

VII. FUTURE WORK

While the proposed framework provides robust tamper resistance across the container image lifecycle, there remain several strategic directions for future development to improve adaptability, intelligence, and scalability. As threats evolve and enterprise requirements become more dynamic, security frameworks must anticipate and respond to emerging challenges with precision and resilience.

Scalability:
As adoption of the proposed tamper-resistant framework expands across large enterprises and government infrastructures, the volume of events and logs recorded on the blockchain ledger will increase significantly. To address this, future work must focus on optimizing blockchain performance to support high-throughput environments. Strategies such as transaction batching, sharding, and implementing lightweight consensus protocols (e.g., RAFT, PBFT) can reduce latency and improve throughput. Moreover, off-chain scaling...

Integration with AI/ML:
Artificial Intelligence (AI) and Machine Learning (ML) offer considerable potential for enhancing supply chain visibility and response time. Future iterations of the framework can integrate anomaly detection models that analyze container build logs, deployment telemetry, and access logs in real time. These models can detect suspicious patterns—such as unusual time-of-day activity, image mismatches, or unauthorized registry access—and alert security teams before the threat escalates.
Incorporating reinforcement learning techniques can further improve the adaptability of the system. For example, the framework can dynamically adjust trust scores for registry endpoints, contributors, or pipelines based on historical behavior. If a specific contributor frequently introduces unsigned or vulnerable images, the system can automatically trigger additional scrutiny or block further pushes until verification is completed.

Additionally, automated threat mitigation systems can use AI-driven risk scoring to quarantine suspicious images, notify appropriate stakeholders, and even roll back compromised deployments. These intelligent systems learn from past incidents to evolve the security posture, enabling faster response and minimizing false positives.

The fusion of AI/ML with the tamper-resistant framework ensures that the system not only enforces compliance but also adapts proactively to evolving threats. This creates a security ecosystem capable of continuous learning, autonomous response, and predictive threat mitigation, a crucial advantage in modern DevSecOps environments.

Policy-Driven Controls:
Another promising area of enhancement is the integration of policy-as-code engines and smart contracts. Currently, policies for container image validation are largely manual or platform-specific. Future work can extend these capabilities by defining a policy engine that interprets declarative policies embedded in smart contracts. These contracts would execute validation logic autonomously, ensuring that only compliant images—based on digital signatures, scanning reports, and approved registries—are deployed into production environments.

Cross-Cloud Interoperability:
Modern enterprises operate in multi-cloud environments to leverage the strengths of different cloud service providers. Ensuring that the tamper-resistant framework functions seamlessly across platforms such as AWS, Azure, Google Cloud Platform (GCP), and private on-premises infrastructure is essential for consistent enforcement. Future research should focus on building cloud-agnostic modules, adopting open standards (e.g., OCI for containers), and enabling federated identity and access management (IAM) for secure, role-based controls across environments.

Regulatory Compliance and Governance:
As regulatory environments become increasingly stringent, organizations must demonstrate continuous compliance with standards like GDPR, HIPAA, SOX, and NIST. Future work should incorporate automated compliance reporting within the framework. This includes generating real-time audit logs, proof of software provenance, and policy enforcement evidence that can be submitted during audits.

Runtime Adaptability and Feedback Loops:
Beyond static validation, future enhancements could explore runtime feedback mechanisms. For example, a container running in production that starts behaving anomalously could trigger retroactive image verification or isolate the workload automatically.

These adaptive security features ensure that runtime behavior remains consistent with signed and approved states.

Collaborative Security and Threat Intelligence Sharing:
To promote ecosystem-wide security, future frameworks could support integration with threat intelligence feeds and industry sharing platforms such as MISP or STIX. This enables collaborative defense by allowing organizations to share verified indicators of compromise (IOCs), attack signatures, and mitigation strategies.

In conclusion, while the current framework establishes a secure foundation for image lifecycle management, its future lies in intelligent, scalable, and collaborative enhancements. By embracing AI, extending cross-cloud functionality, and adopting proactive governance measures, the framework can evolve into a dynamic shield for the software supply chain in cloud-native environments.

REFERENCES

[1] A. Sharma, N. Kumar, and S. Tanwar, "A blockchain-based framework for software supply chain security," IEEE Internet of Things Journal, vol. 8, no. 8, pp. 6892-6901, 2021.

[2] D. Bisson, "The SolarWinds attack: Lessons learned," Security Intelligence, 2021. [Online]. Available: https://securityintelligence.com

[3] D. Merkel, "Docker Content Trust and Notary v2: Secure image signing," Docker Docs, 2023. [Online]. Available: https://docs.docker.com

[4] K. Christidis and M. Devetsikiotis, "Blockchain and smart contracts for the Internet of Things," IEEE Access, vol. 4, pp. 2292-2303, 2016.

[5] R. Hurst, "Immutable storage in cloud security," CloudTech Journal, vol. 7, pp. 52-59, 2022.

[6] NIST, "Supply Chain Risk Management Practices for Federal Information Systems and Organizations," NIST Special Publication 800-161, Rev. 1, 2022.

[7] CNCF, "Cloud Native Security Whitepaper v1.0," Cloud Native Computing Foundation, 2020. [Online]. Available: https://www.cncf.io/whitepapers

[8] T. Kim, Y. Kim, and J. Kim, "Secure container image distribution using blockchain," Journal of Information Security and Applications, vol. 62, 2022.

[9] A. Miller et al., "Smart Contract Security: Understanding and Reducing Critical Risks," IEEE Security & Privacy, vol. 16, no. 3, pp. 68-74, 2018.

[10] J. Lou, Y. Tian, L. Pan, and S. Liu, "Secure and Efficient Integrity Verification of Cloud Data with Blockchain," IEEE Transactions on Services Computing, 2022.

[11] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, 2018.

[12] S. Chandrasekaran et al., "Container security: Issues, challenges, and the road ahead," IEEE Cloud Computing, vol. 9, no. 1, pp. 57-65, 2022.

[13] A. Singla, B. Mani, and J. Bhatia, "End-to-end security for software supply chains: An AI and blockchain-based approach," ACM Computing Surveys, 2023.

[14] N. Scaife, H. Carter, P. Traynor, and K. Butler, "Cryptographic enforcement of end-to-end software provenance," Proceedings of the ACM Conference on Computer and Communications Security (CCS), 2016.

[15] A. Phillips and W. Stallings, "Software Supply Chain Security: A Layered Approach," IEEE Software, vol. 39, no. 4, pp. 22-28, 2022.

[16] M. Schläpfer, M. Preuß, and C. Meinel, "Trusted Data Provenance in Cloud Environments Using Blockchain," Future Generation Computer Systems, vol. 112, pp. 556-569, 2020.