

CLOUD-NATIVE DIGITAL FORENSICS FOR SECURING MODERN CLOUD ENVIRONMENTS

Veena Vinod Shinde¹, Prof. Deepika Falak², Madhavi Netke³

^{1,2,3}*Department of Computer Engineering, Trinity College of Engineering and Research, Pune*

Abstract : Modern cloud environments are increasingly composed of short-lived virtual machines, containers, Kubernetes workloads, serverless functions, managed databases, software supply chains, and provider-controlled control planes. These properties improve elasticity but weaken traditional host-centric forensic assumptions because evidence is distributed, volatile, multi-tenant, and frequently dependent on provider APIs. This paper proposes a Cloud-Native Digital Forensic Readiness (CNDFR) architecture that treats evidence generation, preservation, normalization, correlation, and chain-of-custody as continuous security controls rather than post-incident activities. The framework integrates cloud control-plane audit logs, identity telemetry, Kubernetes audit events, runtime/eBPF observations, workload logs, network and DNS telemetry, storage/database audit trails, snapshots, artifact provenance, and serverless traces into a canonical evidence layer protected by trusted time, cryptographic hashing, digital signing, and immutable retention. A reproducible analytical evaluation maps eight representative cloud-native attack scenarios to nine evidence classes and compares reactive acquisition, native-provider logging, and CNDFR under explicit capture, integrity, temporal-alignment, and provenance assumptions. The proposed design yields a modeled mean scenario evidence coverage of 0.967 and a forensic-readiness index of 0.958, compared with 0.785/0.759 for native logging and 0.527/0.497 for reactive acquisition. A response-delay model further shows why continuously externalized telemetry is important for ephemeral evidence. The contribution is an implementable, provider-agnostic forensic-readiness pattern that aligns incident response with NIST cloud-forensic guidance while remaining compatible with AWS, Azure, Google Cloud, Kubernetes, and container-security practices.

IndexTerms - Cloud forensics, cloud-native security, digital forensics, Kubernetes, containers, serverless computing, forensic readiness, audit logging, chain of custody, incident response.

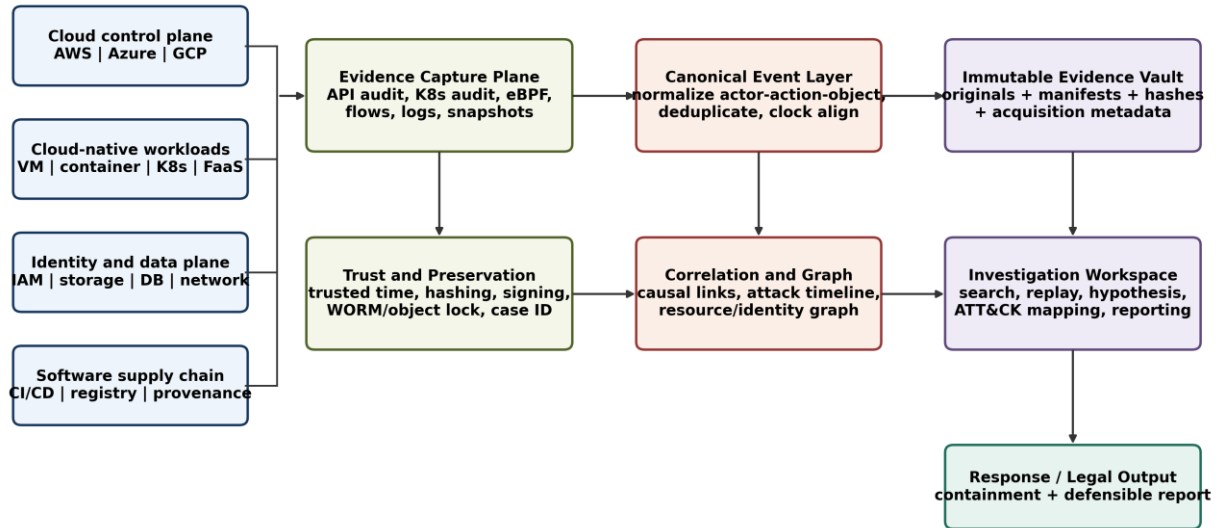
I. INTRODUCTION

Cloud computing has shifted enterprise infrastructure from relatively stable hosts and network perimeters to programmable, API-driven environments. Infrastructure-as-a-Service, managed platforms, Kubernetes, containers, serverless functions, and continuous delivery pipelines can create and destroy computational state within seconds. This elasticity complicates digital investigations because relevant artifacts may exist only in provider control planes, short-lived containers, identity systems, managed services, or remote storage rather than on a single seizable device. NIST identifies jurisdiction, data location, multi-tenancy, time synchronization, logging, chain of custody, and dependence on cloud service providers among the recurring challenges that affect cloud forensic science [1].

The publication of NIST SP 800-201 formalized cloud forensic readiness as an architectural concern rather than a purely reactive investigative activity [2]. This perspective is particularly important in cloud-native systems. Kubernetes auditing records security-relevant API actions, but the evidentiary value of those actions depends on audit-policy coverage, off-cluster retention, timestamp quality, and resistance to tampering [8]. Major cloud providers expose control-plane and data-access logs, yet default retention and coverage differ, and some high-volume data-access logs require explicit configuration [9–12]. Therefore, a modern forensic design must deliberately externalize, correlate, and preserve evidence before an adversary or autoscaler removes the underlying workload.

The research problem addressed in this work is the absence of a compact, provider-agnostic architecture that connects cloud-native telemetry to forensic preservation and evidentiary reasoning. Existing cloud-forensics literature established the challenges of remote evidence acquisition, trust, and provider dependence [3–7], while contemporary cloud platforms now expose richer audit, runtime, and provenance signals. The contribution is fourfold: (i) a layered CNDFR architecture separating evidence capture from analysis; (ii) a canonical event and evidence-manifest model; (iii) an evidence-volatility and response-delay formulation; and (iv) a transparent analytical evaluation using attack-to-evidence requirements rather than unverified experimental claims. Figure 1 presents the architecture.

Cloud-Native Digital Forensic Readiness (CNDFR) Architecture



Continuous capture is separated from investigation to preserve provenance and reduce evidence loss.

Figure 1. Cloud-native forensic readiness architecture.

II. RELATED WORK AND RESEARCH GAP

Early cloud-forensics research showed that conventional imaging techniques do not directly transfer to distributed infrastructure. Dykstra and Sherman demonstrated both the feasibility and trust limitations of acquiring evidence from Infrastructure-as-a-Service systems [3]. Martini and Choo proposed an iterative framework emphasizing preservation and collection changes required by cloud computing [4]. FROST later showed that management-plane acquisition could obtain OpenStack virtual disks, API logs, and firewall logs with cryptographic integrity support [5]. Meta-studies and later operational systems continued to identify provider dependence, evidence segregation, provenance, scale, and legal jurisdiction as persistent problems [6,7].

The cloud-native stack introduces evidence surfaces not covered by older VM-centric models. Kubernetes audit records can identify who performed an API action, on which resource, from where, and at which processing stage [8]. AWS CloudTrail supports SHA-256 hashing and RSA-signed digest files for log-file validation [9]. Google Cloud Audit Logs separates Admin Activity, Data Access, System Event, and Policy Denied records, with Admin Activity and System Event generated by default [10]. Azure Activity Log captures subscription-level control-plane events, retains them for 90 days by default, and supports export for longer retention [11]. These capabilities remain fragmented and do not themselves guarantee cross-layer provenance or a continuous chain of custody.

Table 1 compares representative contributions and exposes the gap addressed by CNDFR: integrating provider audit, orchestration audit, runtime observation, immutable preservation, provenance, and forensic correlation in one readiness model.

Table 1. Representative cloud-forensics approaches

Source	Primary contribution	Cloud-native coverage	Remaining gap
Dykstra & Sherman [3]	IaaS evidence acquisition and trust analysis	VM/IaaS focused	Limited orchestration/runtime telemetry
Martini & Choo [4]	Iterative cloud forensic framework	General cloud process	No native telemetry integration
FROST [5]	Management-plane acquisition with hashes	OpenStack IaaS	Provider/platform specific
Zawoad & Hasan [6]	Challenges and research agenda	Cross-cloud conceptual	No continuous evidence pipeline
CETS [7]	Large-scale public-cloud evidence tracing	Operational cloud investigations	Architecture not provider-agnostic
NIST SP 800-201 [2]	Forensic-reference architecture/readiness	Broad cloud capabilities	Requires deployment-specific implementation
Proposed CNDFR	Continuous capture, preservation, normalization and correlation	VM, container, K8s, FaaS, managed services	Needs production benchmark validation

III. THREAT MODEL AND DESIGN OBJECTIVES

The threat model assumes an enterprise operating public-cloud accounts or subscriptions, Kubernetes clusters, managed data services, and CI/CD infrastructure. The adversary may obtain valid cloud credentials, escalate privileges, deploy malicious workloads, escape a container, steal secrets, disable telemetry, compromise a serverless function, or exfiltrate data. The model does

not assume compromise of the cloud provider cryptographic root of trust. The defender controls tenant-side logging configuration, cross-account or cross-project export, retention policy, evidence-vault permissions, and incident-response automation.

The design objectives are: O1, evidence completeness across control plane, data plane, identity, workload, network, storage, and software supply chain; O2, cryptographic integrity and immutable retention; O3, temporal consistency through trusted time; O4, provenance binding each artifact to acquisition method, source, case identifier, and processing history; O5, multi-tenant minimization; O6, provider portability through a canonical event model; and O7, correlation with an attack-technique vocabulary. MITRE ATT&CK for Containers includes techniques such as Deploy Container, Escape to Host, Steal Application Access Token, Disable or Modify Tools, and Resource Hijacking [13].

Figure 2 models relative evidence volatility. Its half-lives are explicit analytical assumptions used later to study response delay, not provider retention guarantees. The design implication is that runtime, orchestration, and workload evidence must be streamed out of the affected trust zone instead of collected only after an alert.

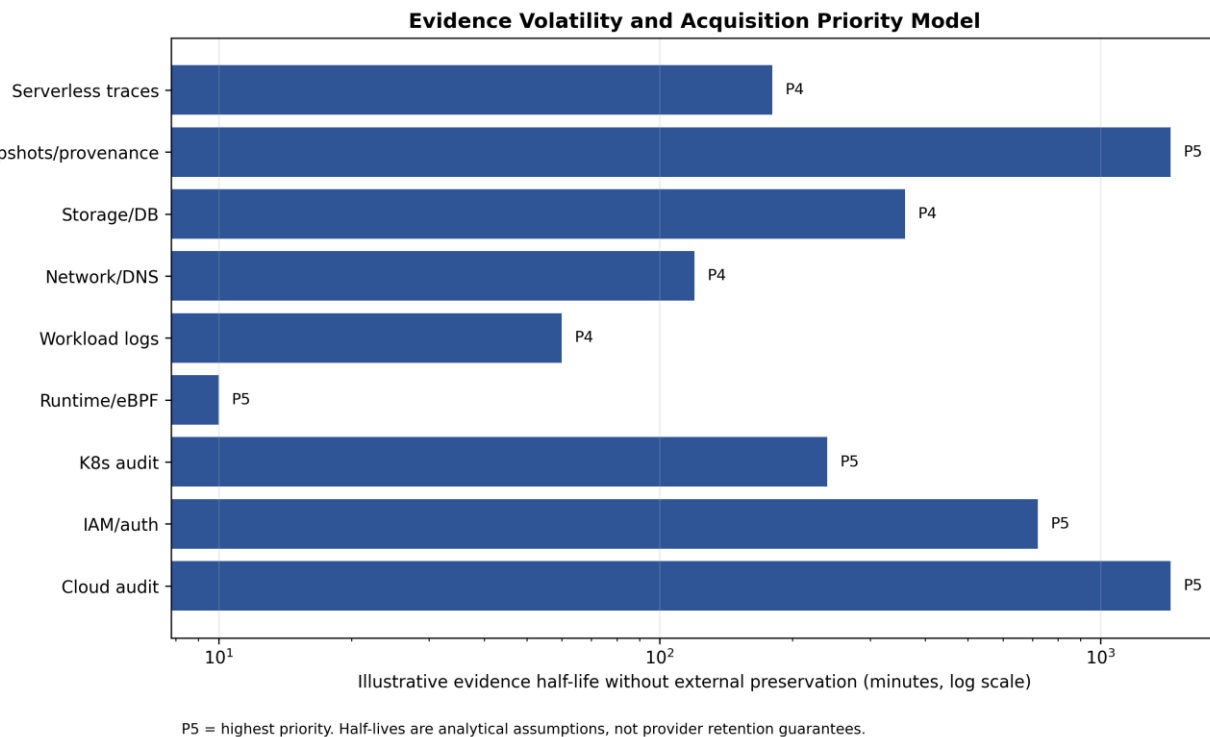


Figure 2. Evidence volatility and acquisition priority.

Table 2 summarizes the evidence surfaces captured by CNDFR and the investigative questions supported by each source.

Table 2. Cloud-native forensic evidence surfaces

Evidence source	Typical artifacts	Primary forensic questions	Preservation control
Cloud control plane	API audit, resource and policy changes	Who changed what resource, when, and from where?	Cross-account export; immutable archive
IAM/authentication	Sign-ins, token issuance, role assumption, MFA	Which principal authenticated and which credentials were used?	Central identity log export
Kubernetes audit	API requests, verbs, users, resources, stages	Who changed workloads, secrets, RBAC, or webhooks?	Off-cluster audit backend
Runtime/eBPF	Process, file, socket, syscall, container metadata	What executed inside or across container boundaries?	Protected streaming collector
Workload logs	Service logs, traces, application audit	What application action occurred?	Structured logging with trace IDs
Network/DNS	Flow, load-balancer, DNS logs	Which endpoints communicated and by what path?	Central flow/DNS retention
Storage/database	Object access, DB queries, key events	What data was read, changed, or exported?	Data-access audit and archive
Snapshots/provenance	Disk snapshots, images, SBOM/attestations	What code/data state existed and where did it originate?	Read-only snapshot + digest
Serverless traces	Invocation, identity, trigger, downstream API traces	What invoked the function and what followed?	Provider audit + distributed trace

IV. PROPOSED CNDFR FRAMEWORK

A. Continuous evidence capture

CNDFR uses an out-of-band capture plane. Provider audit records, IAM events, Kubernetes audit events, runtime telemetry, application logs, flow/DNS records, data-service audit logs, snapshots, and software-provenance metadata are exported

continuously to a security account, project, or subscription that production administrators cannot modify. This separation reduces the probability that an attacker who compromises a workload or production control plane can retroactively alter the evidence repository. Kubernetes audit policies are configured to record security-relevant metadata without routinely collecting sensitive request bodies unless justified [8]. Provider data-access logging is enabled where default coverage is insufficient; for example, Google Cloud Data Access logs generally require explicit enablement outside BigQuery [10].

B. Preservation, integrity and chain of custody

Every acquired object is bound to an evidence manifest containing case identifier, source system, acquisition principal, acquisition API or sensor, source timestamp, collector receipt timestamp, cryptographic digest, storage object identifier, and processing history. Let artifact i be $E_i=(d_i, m_i, t_i, s_i)$, where d_i is raw evidence, m_i metadata, t_i trusted acquisition time, and s_i source. Integrity is $h_i=SHA-256(d_i \parallel Canonical(m_i))$. A manifest batch B_k is signed as $\sigma_k=Sign_K(H(h_1 \parallel h_2 \parallel \dots \parallel h_n \parallel B_{k-1}))$, forming a hash-linked sequence. This generalizes the same principle used by provider mechanisms such as CloudTrail digest signing [9].

C. Canonical event model and correlation

Provider-specific events are normalized into an actor-action-object-result model containing actor identity, assumed role or service account, action, resource, network context, cloud/cluster/namespace identifiers, event time, ingestion time, trace/request identifier, and evidence pointer. This enables one query to follow an identity across a provider API call, Kubernetes admission, container execution, outbound DNS request, and object-store access. Table 3 defines the minimal schema.

Table 3. Canonical event and evidence schema

Field	Definition	Purpose
event_id	Globally unique event identifier	Deduplication and graph-node identity
event_time / ingest_time	Source and collector timestamps	Clock-skew analysis and ordering
actor	User, service account, role, workload identity	Identity attribution
action	Normalized verb/API/syscall activity	Cross-provider behavior comparison
object	Cloud resource, K8s object, file, secret, DB/object	Target attribution
source_context	IP, session, cluster, node, namespace	Origin and scope
result	Success, deny, error, policy decision	Outcome interpretation
trace_id / request_id	Native correlation identifier	Causal linkage
artifact_hash	SHA-256 of preserved artifact	Integrity verification
case_id / custody_state	Investigation ownership and state	Chain-of-custody tracking

D. Forensic-ready investigation lifecycle

Figure 3 presents the operational lifecycle. Readiness establishes logging, retention, clock synchronization, evidence-vault isolation, and policy constraints before an incident. A trigger creates a case identifier and preservation scope; high-volatility evidence is captured first, after which logs and snapshots are normalized and correlated. Analytical outputs are generated from immutable originals or verified working copies. The feedback stage converts each case into improved retention, detection, and acquisition rules.

Forensic-Ready Investigation Lifecycle

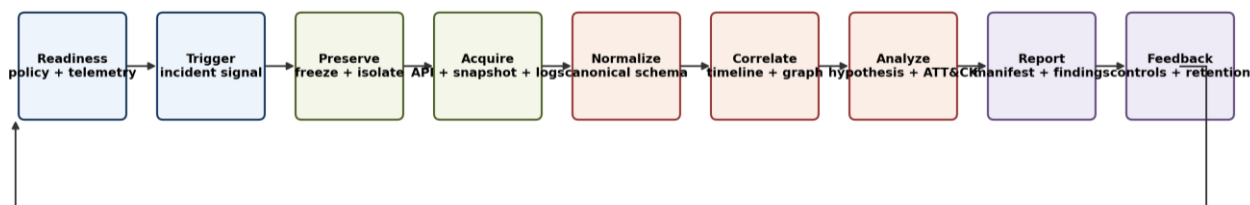


Figure 3. Forensic-ready investigation lifecycle.

V. ANALYTICAL EVALUATION METHODOLOGY

The framework is evaluated analytically rather than by claiming unperformed production experiments. Eight representative attack scenarios are mapped to evidence classes required to reconstruct actor, action, target, sequence, and consequence. Table 4 defines the scenarios. The mapping can be extended using NIST SP 800-201 challenge data and MITRE ATT&CK techniques [2,13]. Figure 4 visualizes the scenario-to-evidence requirement matrix.

Table 4. Representative cloud-native attack scenarios

ID	Scenario	Required evidence classes	Investigative objective
S1	Stolen cloud account	Cloud audit, IAM, network/DNS, storage/DB	Attribute credential use and affected resources
S2	Privilege escalation	Cloud audit, IAM, K8s audit, runtime	Reconstruct permission changes and execution
S3	Malicious pod deployment	Cloud audit, K8s audit, runtime, provenance	Link deployment actor, image and runtime behavior
S4	Container escape	K8s audit, runtime, workload, network	Establish container-to-host transition
S5	Secret theft	IAM, K8s audit, runtime, storage/DB	Identify secret access and subsequent use
S6	Log disabling	Cloud audit, IAM, K8s audit, provenance	Detect evidence impairment and actor
S7	Serverless compromise	Cloud audit, IAM, network/DNS, storage/DB, FaaS traces	Trace invocation and downstream services
S8	Data exfiltration	Cloud audit, IAM, workload, network/DNS, storage/DB	Link data access to outbound transfer

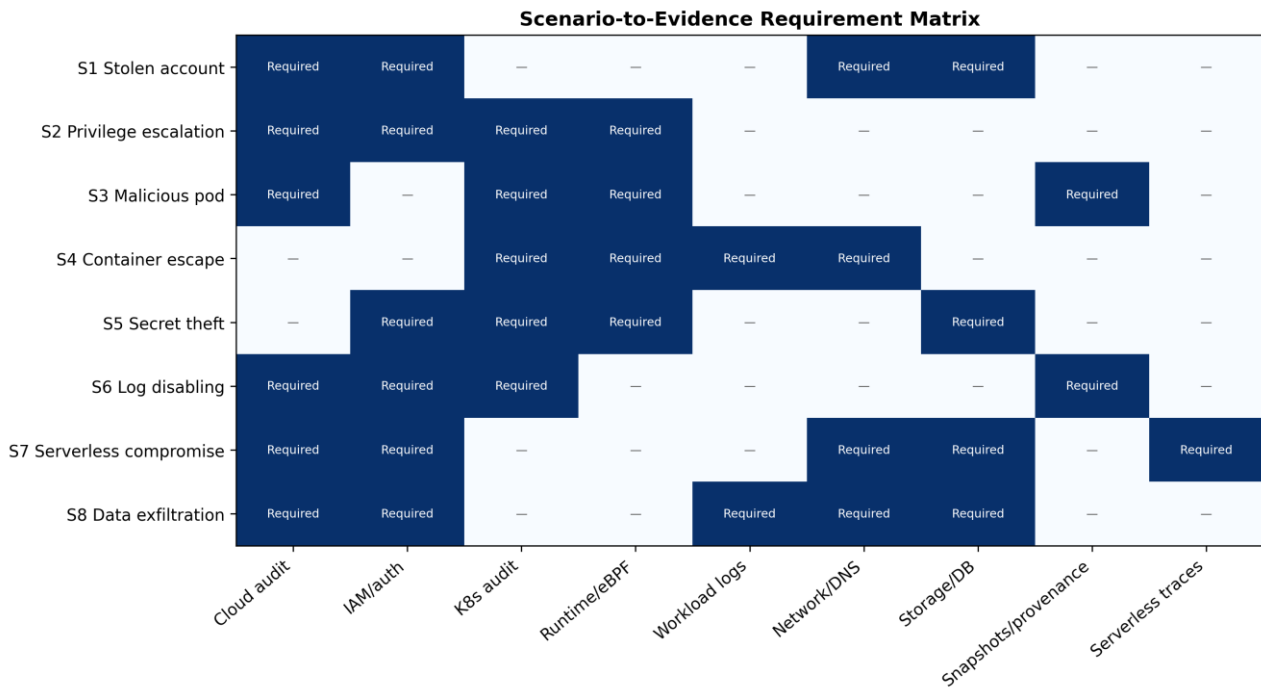


Figure 4. Scenario-to-evidence requirement matrix.

A. Coverage and readiness metrics

For scenario s with required evidence set R_s , strategy j has evidence-availability probability $p_{\{j,e\}}$. Scenario coverage is $C_{\{j,s\}} = (1/|R_s|) \sum_{e \in R_s} p_{\{j,e\}}$; mean coverage is $\bar{C}_j = (1/S) \sum_s C_{\{j,s\}}$. A composite Forensic Readiness Index combines coverage, integrity, temporal alignment, and provenance: $FRI_j = 0.35\bar{C}_j + 0.25I_j + 0.20T_j + 0.20P_j$. The weights emphasize evidence existence while requiring trustworthiness and reconstructable timing.

B. Response-delay model

Each evidence class e receives an analytical half-life τ_e representing how quickly unexternalized evidence can disappear because of workload termination, log rotation, node replacement, or service retention. Residual recoverability after delay t is $q_e(t) = 2^{-t/\tau_e}$. For strategy j , a durable fraction $d_{\{j,e\}}$ has already been exported to protected storage, yielding $Q_j(t) = \sum_e w_e[d_{\{j,e\}} + (1 - d_{\{j,e\}})q_e(t)]$, with $\sum_e w_e = 1$. Table 5 lists the explicit comparative parameters. These values are analytical assumptions, not empirical provider measurements.

Table 5. Analytical evaluation parameters

Parameter	Reactive acquisition	Native logging	Proposed CNDFR
Mean capture probabilities by evidence class	0.25–0.85	0.35–0.97	0.92–0.99
Integrity score I	0.45	0.72	0.96
Temporal-alignment score T	0.60	0.82	0.95
Provenance score P	0.40	0.70	0.95
Continuous externalization	Low	Partial	High
Primary trust boundary	Compromised workload/provider APIs	Provider-native logging	Isolated evidence account + providers
Half-life range in delay model	10–1440 min	10–1440 min	10–1440 min

VI. RESULTS AND DISCUSSION

The analytical comparison shows a substantial difference between collecting evidence after an incident and engineering evidence capture before the incident. Table 6 reports scenario-averaged values. Reactive acquisition reaches mean evidence coverage of 0.527 because short-lived orchestration, runtime, and network evidence may already be gone when investigation begins. Native-provider logging improves coverage to 0.785 but remains weaker for runtime execution, provenance, and some data-plane events unless explicitly enabled. CNDFR reaches modeled coverage of 0.967 because all nine evidence classes are continuously externalized into a protected evidence path.

Table 6. Analytical forensic-readiness results

Strategy	Mean coverage	Integrity	Temporal alignment	Provenance	FRI
Reactive acquisition	0.527	0.450	0.600	0.400	0.497
Native logging	0.785	0.720	0.820	0.700	0.759
Proposed CNDFR	0.967	0.960	0.950	0.950	0.958

Figure 5 shows the response-delay model. With a 60-minute investigation delay, modeled evidence completeness is 0.723 for reactive acquisition, 0.816 for native logging, and 0.982 for CNDFR. At 120 minutes, the corresponding values are 0.621, 0.778, and 0.976. The difference is driven by the durable-export fraction: once evidence has been copied and integrity-protected outside the compromised environment, delay affects investigator convenience more than evidence existence.

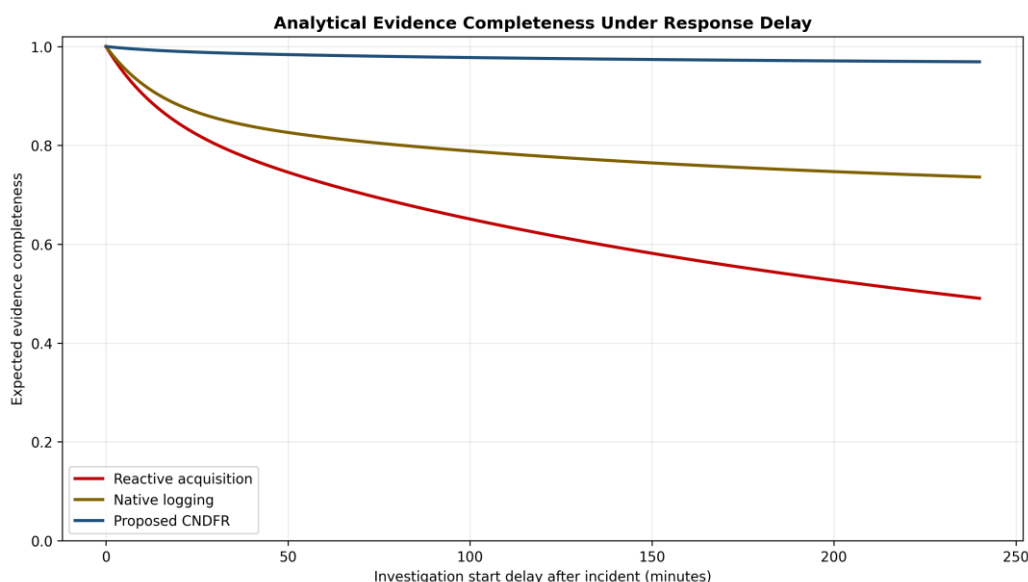


Figure 5. Evidence completeness under response delay.

Figure 6 summarizes composite readiness scores. The result is not a field benchmark or vendor ranking; it demonstrates how evidence coverage, integrity, time alignment, and provenance interact. An organization can replace the assumed probabilities and half-lives with measured telemetry-loss rates, logging-configuration audits, and incident-exercise data to obtain an environment-specific readiness score.

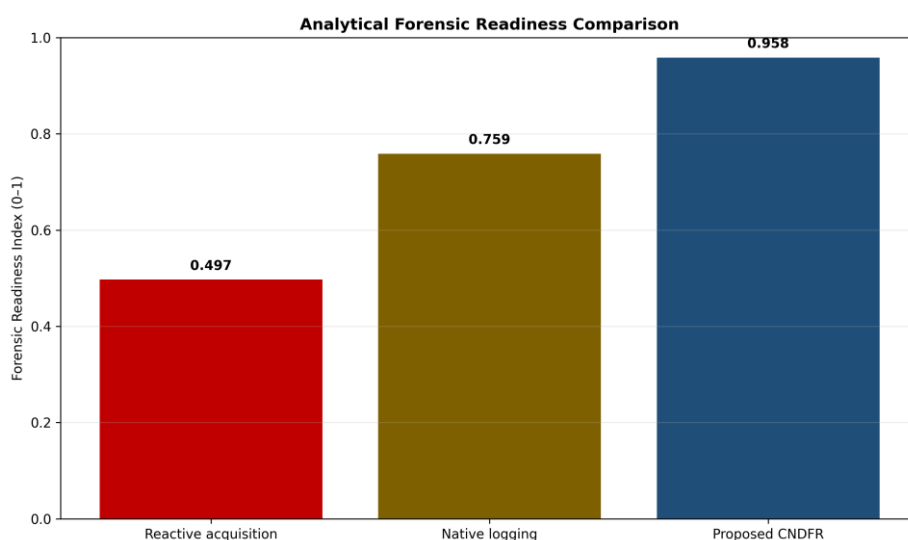


Figure 6. Analytical forensic-readiness comparison.

Provider-native controls remain necessary but insufficient. AWS CloudTrail integrity validation can establish whether delivered log files were modified or deleted [9]; Google Cloud labels Cloud Audit Log entries immutable and separates Admin Activity from Data Access logging [10]; Azure retains Activity Log events for 90 days by default and recommends export for longer retention [11]. CNDFR composes these controls with Kubernetes audit, runtime evidence, application telemetry, provenance, and an independent evidence manifest so investigators can reason across layers.

VII. IMPLEMENTATION AND SECURITY CONSIDERATIONS

A. Evidence-plane isolation

The evidence vault should reside in a separate administrative boundary with write-only ingestion roles, read access restricted to the forensic function, independent encryption keys, object-lock or WORM retention where available, and alerting on retention-policy or key-management changes. Production break-glass administrators should not automatically inherit delete privileges in the vault. Provider-signed audit-integrity metadata should be preserved alongside CNDFR manifests rather than replaced.

B. Kubernetes and container telemetry

Kubernetes audit policies should prioritize authentication, authorization, RBAC, workload creation and mutation, secrets metadata, admission-controller changes, exec/attach/port-forward activity, and audit-policy changes. Runtime sensors should bind process, file, network, and container metadata to immutable workload identifiers. NIST SP 800-190 organizes container security across image, registry, orchestrator, container, and host layers [12]; the same layers form useful forensic evidence boundaries. Secret values and excessive request bodies should not be logged merely for investigative convenience.

C. Privacy, cost and data minimization

Forensic readiness increases logging volume and may expose sensitive information. Retention must therefore be risk-based and legally justified. High-volume sources can use tiered storage: hot normalized metadata for rapid correlation, warm compressed logs for incident windows, and cold immutable originals for evidentiary retention. Field-level redaction or tokenization should be applied before centralization when it does not destroy investigative value. Data-access logs disabled by default should be selectively enabled for sensitive services rather than indiscriminately across all workloads.

D. Reproducibility

The analytical model is reproducible from three inputs: the scenario-to-evidence matrix in Table 4, strategy-specific capture probabilities, and evidence half-lives/durable fractions in Table 5 and Figure 2. Production validation can replace assumptions with measurements from incident-response exercises, chaos experiments that terminate workloads, audit-log disablement tests, and cross-account evidence-retrieval drills.

VIII. LIMITATIONS AND FUTURE VALIDATION

The quantitative results are model-based and are not empirical measurements of AWS, Azure, Google Cloud, Kubernetes, or any commercial forensic product. The evaluation instead verifies internal behavior under explicit assumptions and exposes which parameters determine the outcome. Future work should deploy CNDFR in a multi-cloud cyber range and measure event-loss rate, ingestion latency, cryptographic-validation overhead, storage cost, cross-source timestamp error, investigator query time, and recovery completeness after workload termination.

Legal admissibility and jurisdiction cannot be reduced to technical integrity alone. Evidence acquisition must follow organizational policy, contracts, applicable law, and case-specific authorization. Provider-specific semantics also mean that canonical normalization can lose native detail; original artifacts and provider metadata should therefore be preserved, with the normalized layer serving as an investigative index rather than the sole evidentiary record.

IX. CONCLUSION

Cloud-native digital forensics requires a shift from endpoint seizure to continuous, architecture-level forensic readiness. The proposed CNDFR framework unifies control-plane, identity, Kubernetes, runtime, workload, network, storage, provenance, and serverless evidence; separates production from evidence custody; and normalizes events for cross-layer correlation while retaining immutable originals. A transparent analytical model demonstrates why continuous externalization can preserve substantially more context than post-incident collection when workloads and logs are ephemeral. The framework is compatible with NIST SP 800-201, provider audit mechanisms, Kubernetes auditing, and container-security guidance, and provides a reproducible foundation for future multi-cloud benchmark validation. Its central principle is that forensic evidence should be engineered as a security product of the cloud platform, not treated as a by-product discovered after compromise.

REFERENCES

- [1] Herman, M., Iorga, M., Salim, A. M., Jackson, R. H., Hurst, M. R., Leo, R., Lee, R., Landreville, N., Mishra, A. K., Wang, Y., Sardinas, R., 2020. NIST Cloud Computing Forensic Science Challenges. NISTIR 8006. <https://doi.org/10.6028/NIST.IR.8006>.
- [2] Herman, M., Iorga, M., Salim, A. M., Jackson, R. H., Hurst, M. R., Leo, R. A., Mishra, A. K., Landreville, N. M., Wang, Y., 2024. NIST Cloud Computing Forensic Reference Architecture. NIST SP 800-201. <https://doi.org/10.6028/NIST.SP.800-201>.
- [3] Dykstra, J., Sherman, A. T., 2012. Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques. Digital Investigation, 9(S), S90-S98. <https://doi.org/10.1016/j.diin.2012.05.001>.

- [4] Martini, B., Choo, K.-K. R., 2012. An integrated conceptual digital forensic framework for cloud computing. *Digital Investigation*, 9(2), 71-80. <https://doi.org/10.1016/j.diin.2012.07.001>.
- [5] Dykstra, J., Sherman, A. T., 2013. Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform. *Digital Investigation*, 10(S), S87-S95. <https://doi.org/10.1016/j.diin.2013.06.010>.
- [6] Zawoad, S., Hasan, R., 2013. Cloud Forensics: A Meta-Study of Challenges, Approaches, and Open Problems. *arXiv:1302.6312*.
- [7] Li, X. et al., 2022. Cloud Evidence Tracing System: An integrated forensics investigation system for large-scale public cloud platform. *Forensic Science International: Digital Investigation*, 41, 301391. <https://doi.org/10.1016/j.fsidi.2022.301391>.
- [8] Kubernetes Documentation, 2026. Auditing. [Kubernetes.io](https://kubernetes.io), accessed September 2026.
- [9] Amazon Web Services, 2026. Validating CloudTrail log file integrity. *AWS CloudTrail User Guide*, accessed September 2026.
- [10] Google Cloud, 2026. Cloud Audit Logs overview. *Cloud Logging Documentation*, accessed September 2026.
- [11] Microsoft, 2026. Activity Log in Azure Monitor. *Microsoft Learn*, accessed September 2026.
- [12] Souppaya, M., Morello, J., Scarfone, K., 2017. Application Container Security Guide. NIST SP 800-190. <https://doi.org/10.6028/NIST.SP.800-190>.
- [13] MITRE, 2026. MITRE ATT&CK Enterprise Containers Matrix. *ATT&CK Knowledge Base*, accessed September 2026.
- [14] Kent, K., Chevalier, S., Grance, T., Dang, H., 2006. Guide to Integrating Forensic Techniques into Incident Response. NIST SP 800-86.
- [15] Cichonski, P., Millar, T., Grance, T., Scarfone, K., 2012. Computer Security Incident Handling Guide. NIST SP 800-61 Rev. 2.
- [16] National Institute of Standards and Technology, 2011. The NIST Definition of Cloud Computing. NIST SP 800-145. <https://doi.org/10.6028/NIST.SP.800-145>.
- [17] Docker, 2026. Docker Engine security and logging documentation. *Docker Docs*, accessed September 2026.
- [18] Cloud Native Computing Foundation, 2026. Falco Documentation: Runtime security for cloud-native environments. *Falco Project*, accessed September 2026.
- [19] OpenTelemetry Authors, 2026. OpenTelemetry Specification: Traces, logs and context propagation. *OpenTelemetry.io*, accessed September 2026.
- [20] The in-toto Authors, 2026. in-toto Attestation Framework and Supply Chain Layout. *in-toto Project Documentation*, accessed September 2026.

Copyright & License:

© Authors retain the copyright of this article. This work is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), permitting unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.