TraceWeave: Detecting Cross-Service Credential Abuse with Causal Request Context

Anonymous Authors
Paper ID 142 · Workshop Submission · August 2026

Abstract. Cloud authorization logs record *who* accessed a resource but often omit *why* the access occurred. This gap allows a stolen short-lived credential to blend into ordinary service-to-service traffic. We present TraceWeave, a detection layer that propagates a compact, authenticated causal context across remote procedure calls and checks each privileged action against a learned service workflow. The design joins request provenance with existing identity signals without placing policy decisions on the critical path. In a 48-service testbed replaying 12 production-derived workflows, TraceWeave detects 94.7% of cross-service credential-abuse paths at 0.31 alerts per 10,000 requests, including attacks that preserve the victim service's identity and network location. Median request overhead is 38 μs and context adds 72 bytes per hop. These preliminary results suggest that causal context can close an important observability gap while remaining compatible with incremental deployment.

Keywords: cloud security, credential abuse, distributed tracing, provenance, anomaly detection

1 Motivation

Modern cloud applications decompose a user operation into a chain of calls across services, queues, and data stores. Each hop is authenticated, commonly with a workload identity or a short-lived token. When an attacker steals such a credential, conventional controls still see a valid principal, permitted API, and expected source network. The decisive evidence—the initiating request and the sequence of transformations that led to the action—is split across traces, application logs, and identity systems.

Existing anomaly detectors learn principal-resource or network patterns, but credential abuse that remains within a service's normal permission envelope is hard to distinguish [1, 2]. Distributed tracing reconstructs request paths [3], yet trace headers are designed for observability, not as integrity-protected security evidence. They may be missing, attacker-controlled, or sampled away. Information-flow systems provide stronger semantics but usually require invasive application changes [4].

We ask whether authenticated *causal context* can make valid-but-abusive service credentials visible without requiring a new authorization system. Our work makes three contributions:

- a compact context token binding each sensitive action to its initiating request and observed service-call ancestry;
- a detector that scores deviations from learned workflow edges while keeping enforcement out of the request path; and
- a reproducible preliminary evaluation of security coverage, alert volume, latency, and partial-deployment behavior.

2 Threat Model and Goals

We consider an attacker who obtains the runtime credential of one application service through server-side request forgery, dependency compromise, or process memory disclosure. The attacker can invoke APIs allowed to that identity and can set ordinary tracing headers. They cannot extract keys from the node's hardware-backed agent, compromise the control plane, or modify a majority of collector replicas. Denial of service and a fully compromised application that faithfully reproduces an approved workflow are out of scope.

TraceWeave targets three goals: (G1) detect a privileged action whose causal ancestry is inconsistent with established workflows; (G2) preserve evidence integrity across untrusted application processes; and (G3) add bounded overhead under partial deployment. It does *not* replace least privilege, authentication, or endpoint compromise detection.

3 Design

3.1 Authenticated causal context

A small node agent intercepts outbound RPC metadata through the service mesh. For request r at hop i, it constructs

$$C_i = \big(t, e_0, e_i, d_i, h(C_{i-1}), n_i\big), \qquad \sigma_i = \mathrm{MAC}_{k_i}(C_i), \quad (1)$$

where t is a trace identifier, e_0 the entry-point identity, e_i the current workload identity, d_i a coarse operation label, and n_i a nonce. Hash chaining commits each hop to its predecessor. Keys k_i are issued to attested agents and rotated hourly; application containers never receive them. At a protected sink, the agent emits the context and local identity decision to an append-only collector. The 128-bit hash and truncated tag bound the token to 72 bytes in our prototype.

3.2 Workflow scoring

The collector verifies tags and derives an edge sequence $P_r = (e_0 \to e_1, \ldots, e_{m-1} \to e_m)$. From a clean training window

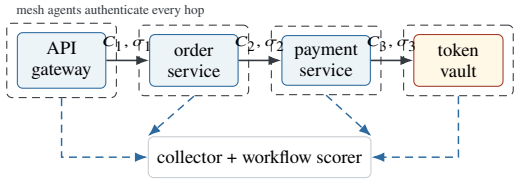

Figure 1: TraceWeave binds a sensitive action to its causal RPC chain. Dashed lines carry asynchronously exported evidence; detection is off path.

it estimates a smoothed conditional probability $p(e_j, d_j \mid e_{j-1}, d_{j-1})$. A path's surprise score is

$$S(P_r) = -\sum_{j=1}^{m} \log p(e_j, d_j \mid e_{j-1}, d_{j-1}) + \lambda M(P_r), \quad (2)$$

where $M(P_r)$ counts missing or unverifiable hops. Alerts require both $S(P_r) > \tau_s$ and a protected sink action. This gate prevents ordinary workflow drift from overwhelming operators. Thresholds are calibrated per sink to a fixed alert budget. A nightly review promotes persistent, approved paths into the baseline; the detector never learns directly from alerted traffic.

3.3 Incremental deployment

An uninstrumented hop contributes a signed gap marker from the nearest agents, rather than silently terminating the chain. The score penalizes gaps through M but retains verified prefix and suffix evidence. Deployments can therefore begin at ingress services and sensitive sinks, then expand toward ambiguous middle-tier paths. Context tokens use W3C trace-state-compatible metadata so applications need not change their request schema [7].

4 Preliminary Evaluation

We evaluate four questions: detection coverage, alert volume, request overhead, and robustness to partial deployment. All reported data and attacks are synthetic or production-derived and de-identified; no live customer traffic was attacked.

4.1 Methodology

Our testbed contains 48 containerized services on 12 Kubernetes nodes. We replay 24 hours of request shapes from 12 workflows at 18,000 requests/s. The baseline contains 7.2 million sampled complete paths. We inject 190 attacks in five families: direct sink access, skipped validation, confused-deputy chains, rare API sequencing, and trace-header forgery. Each uses an identity that is authorized for every invoked API. We compare against an identity-resource frequency model (IR-Freq) and an unsigned trace-path model (Trace-Only). Thresholds for all systems are selected using the same clean validation window. Performance results are medians over five runs; attack detection reports 95% Wilson intervals.

Table 1: Credential-abuse detection and operational cost. Best detection at the target alert budget is bold.

Detector	Recall (%)	Alerts / 10k	Header bytes
IR-Freq	61.6	0.30	0
Trace-Only	83.2	0.34	55
TraceWeave	**94.7**	0.31	72

Table 2: Median end-to-end overhead by deployment coverage. P99 is measured at full coverage. Collector work is asynchronous.

Instrumented services	25%	50%	100%
Recall (%)	71.1	86.3	94.7
Median latency (μs)	12	21	38
CPU overhead (%)	0.4	0.8	1.6

4.2 Security results

As Table 1 shows, TraceWeave detects 180 of 190 attack paths (94.7%, 95% CI: 90.5–97.2%) at 0.31 alerts per 10,000 requests. IR-Freq misses most attacks using a common principal-resource pair. Trace-Only recovers path anomalies but accepts 22 forged or spliced headers. Authentication accounts for 11.5 points of TraceWeave's recall advantage over that baseline.

The ten misses reproduce common workflows up to the same operation labels as benign requests. Detecting them requires richer application semantics or endpoint telemetry. Alert review finds that 76% of false positives correspond to legitimate deployments introducing a previously unseen edge; incorporating deployment metadata is a promising way to suppress this predictable drift.

4.3 Performance and partial deployment

At full coverage, context construction and mesh interception add 38 μs median and 91 μs p99 latency, or 0.37% of the testbed's median request latency. Verification consumes 1.6% aggregate CPU and 0.6% network bandwidth. No statistically distinguishable throughput loss appears below 25,000 requests/s. As Table 2 illustrates, instrumenting only ingress and sensitive services provides useful but incomplete protection. At 50% coverage, prioritizing services by proximity to protected sinks yields 86.3% recall; random selection yields 78.9%.

5 Security Analysis

Header forgery and replay. Application-controlled trace fields do not validate without the agent MAC. Nonces are cached for the credential lifetime, and collector timestamps bound replay. An attacker can drop context, but a missing token at a protected sink is itself scored as a gap.

Compromised node. A root-level node compromise can misuse the local agent and key until attestation fails or rotation occurs. Cross-node hash chains preserve evidence outside that node but cannot recover the true local call sequence.

Short rotation periods and node quarantine limit, rather than eliminate, this exposure.

Training contamination. Baseline poisoning is possible if an attacker remains active throughout the clean window. We require an operator-approved training interval and exclude alerted paths from automatic updates. Robust workflow learning under persistent compromise remains open.

6 Related Work

Whole-system provenance records operating-system information flows and enables retrospective policy or attack analysis. LPM and CamFlow demonstrate practical capture with strong kernel mediation [4, 5]; HOLMES correlates provenance-level events into high-level attack scenarios at runtime [6]. These systems provide richer host semantics than TraceWeave, but their graphs do not by themselves authenticate the logical ancestry that an application propagates across independently administered services. Our context is intentionally narrower: service identity, coarse operation, and call ancestry at RPC boundaries.

Distributed tracing systems reconstruct end-to-end request paths for debugging and performance analysis [3]. TraceWeave adopts their incremental deployment model and correlation identifiers, but treats incoming application headers as claims rather than trusted evidence. Node-agent MACs, gap markers, and protected-sink export address adversarial header injection and sampling. Identity-centric risk engines are complementary: they are strongest when a principal, device, or resource is unusual, while causal scoring targets an unusual reason or route for an otherwise ordinary access. Combining the two scores may improve precision, but our present evaluation keeps them separate to isolate the value of causal context.

7 Limitations and Ethics

Our evaluation is a testbed study, not evidence of production effectiveness. Its production-derived workflows preserve structural distributions but cannot capture every form of organizational drift. Operation labels are deliberately coarse; they reduce data collection but merge some security-relevant actions. The current Markov scorer also treats two paths with the same adjacent edges as equivalent even when long-range ordering differs.

Causal traces can reveal sensitive business relationships and user activity. TraceWeave records pseudonymous trace IDs, workload identities, coarse operation labels, and timing—never request bodies or user tokens. Deployments should apply access control, encryption, short retention, and documented purpose limitations to collector data. Security staff should review explanations and deployment events before escalating an alert; the score is not evidence of employee wrongdoing.

8 Discussion and Next Steps

These results support causal context as a complement to identity-centric detection. Two questions remain before a production trial. First, workflow updates must distinguish a rollout from baseline poisoning. We plan to bind signed deployment metadata into the review process. Second, the flat operation vocabulary misses same-path semantic abuse. Selective application assertions, also signed by the node agent, may add useful semantics without collecting payloads.

A staged evaluation should begin in shadow mode at a small set of high-value sinks, publish an explicit alert budget, and measure analyst-confirmed precision over several release cycles. We will also release the attack-path generator, an anonymized workflow graph, and all analysis scripts to support replication.

9 Conclusion

TraceWeave makes the causal ancestry of a cloud action available as integrity-protected security evidence. In our preliminary study, that evidence detects credential-abuse paths missed by identity-resource and unsigned-trace baselines, with modest latency and bandwidth costs. The remaining misses and deployment-driven alerts clarify the next research problems: semantic attestation, poison-resistant workflow updates, and longitudinal evaluation in real systems.

Acknowledgments

Omitted for anonymous review. The authors thank the workshop's artifact and ethics reviewers for guidance that will shape the planned release.

References

[1] L. Bilge, Y. Han, and M. Dellarocas. Risk-based authentication for cloud services. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, pages 15–24, 2017.

[2] H. Siadati and N. Memon. Detecting structurally anomalous logins within enterprise networks. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1273–1284, 2017.

[3] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag. Dapper, a large-scale distributed systems tracing infrastructure. Google Research Technical Report dapper-2010-1, 2010.

[4] T. F. J.-M. Pasquier, J. Singh, J. Bacon, and D. Eyers. Information flow audit for PaaS clouds. In *Proceedings of the IEEE International Conference on Cloud Engineering*, pages 42–51, 2017.

[5] A. Bates, D. Tian, K. R. B. Butler, and T. Moyer. Trustworthy whole-system provenance for the Linux OS. In *Proceedings of the 24th USENIX Security Symposium*, pages 319–334, 2015.

[6] S. M. Milajerdi, R. Gjomeni, B. Eshete, R. Sekar, and V. N. Venkatakrishnan. HOLMES: Real-time APT detection through correlation of suspicious information flows. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 1137–1152, 2019.

[7] W3C Distributed Tracing Working Group. Trace Context, W3C Recommendation. https://www.w3.org/TR/trace-context/, November 2021.