International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Continuous Compliance as Code: AI-Driven Detection and Auto-Remediation of Regulatory Drift in Regulated Banking Infrastructure

Ajay Devineni

ajayd4030@gmail.com

Abstract:
SOC 2 Type II compliance in cloud-native financial services environments is typically managed as a periodic discipline: scheduled audits, point-in-time assessments, and quarterly remediation cycles. This approach is structurally misaligned with the continuous change velocity of modern cloud infrastructure, where security group rules, IAM policies, patch baselines, and certificate configurations change on timescales of days rather than quarters. The gap between periodic compliance snapshots and continuous infrastructure reality creates compliance drift — deviation of actual infrastructure state from required compliance posture — that accumulates silently between audit cycles. Research indicates that approximately 90% of large-scale infrastructure-as-code deployments experience drift, and nearly half of these deviations remain undetected. This paper presents a continuous compliance as code framework implemented across six credit union banking applications at NCR/Candescent, integrating Wiz, Orca Security, CrowdStrike Falcon, Carbon Black, and Cloudflare with AWS-native controls and Terraform-based infrastructure as code to create a continuously monitored and automatically remediated compliance posture. The framework treats every SOC 2 control as a continuously evaluated state assertion rather than a periodic check, with automated remediation pipelines self-healing specific categories of drift without manual intervention. Outcomes include elimination of all patch compliance drift within a 24-hour detection-to-remediation window, zero certificate-related compliance findings in the most recent SOC 2 Type II audit cycle, and SOC 2 audit evidence assembly reduced from two weeks to four hours through continuous automated evidence generation.

Keywords: SOC 2 Compliance, Compliance as Code, Wiz Security, Orca Security, CrowdStrike Falcon, Carbon Black, Cloudflare, AWS WAF, Certificate Lifecycle Management, Patch Management, Infrastructure as Code, Terraform, Regulatory Drift, Financial Services Cloud.

1. Introduction

The SOC 2 Type II audit covering my most recent operational period lasted four weeks. The auditors requested evidence of 47 distinct control activities. Assembling that evidence under the previous compliance management approach required two weeks of manual work: pulling CloudWatch log archives, exporting CrowdStrike scan results, documenting patch deployment records, compiling certificate renewal histories, and generating IAM permission reports across four client environments. The evidence was accurate, but it was a snapshot — and the auditors were evaluating whether controls had been continuously effective over the entire audit period, not merely effective at the moment evidence was collected.

The fundamental problem with periodic compliance management in cloud-native banking infrastructure is that cloud infrastructure changes continuously while compliance audits happen periodically. In the environments I operate, Terragrunt deployments modify infrastructure state multiple times per week. Security group rules are updated for new service dependencies. IAM policies are adjusted as role requirements change. Patches are released on schedules that do not align with quarterly audit cycles.

Compliance drift — the gap between required compliance posture and actual current state — accumulates continuously and silently between audits, and is only made visible when an auditor asks for evidence.

Research supports this characterization of the problem: approximately 90% of large-scale IaC deployments experience drift, with nearly half of deviations remaining undetected between audit cycles (Rahman et al., 2019). AI-driven auto-remediation has been shown to reduce mean time to remediation by 78% to 87.5% compared to manual processes in comparable cloud environments, transforming compliance drift from a weeks-long exposure window into an hours-long one. The framework I describe implements this capability across a multi-tool security stack in a SOC 2 regulated financial services context.

The continuous compliance framework described in this paper is positioned against two inadequate alternatives that most financial services organizations currently rely on. The first is periodic manual compliance management: assembling evidence once per quarter, identifying drift during that assembly process, remediating manually, and repeating the cycle. This approach accepts that compliance drift exists between cycles and treats it as an acceptable operating condition. For a banking organization processing live customer transactions, accepting known compliance drift as normal is a regulatory risk that the periodic model normalizes rather than eliminates.

The second alternative is point-in-time CSPM tooling without automated remediation. Wiz, Orca, and AWS Security Hub can all detect compliance drift continuously — they generate findings in real time. What most organizations lack is the automated remediation pipeline that converts those findings into corrective actions without requiring engineer intervention for every event. Without automation, a continuous detection capability simply generates a continuously growing queue of compliance tickets that engineers must manually work through, which is faster than quarterly review but still not genuinely continuous remediation.

The contribution of this framework is the closed loop: detection tools that generate structured findings, a tiered remediation pipeline that autonomously resolves Tier 1 and Tier 2 compliance events, and an immutable evidence archive that generates audit documentation continuously rather than at audit time. The 2-week to 4-hour audit evidence assembly improvement is a direct consequence of building a system where evidence generation is not a separate step from operations — it is the same step.

1.1 Novelty and Positioning

2. SECURITY AND COMPLIANCE TOOL STACK

2.1 Multi-Tool Integration Architecture

The compliance monitoring stack integrates five security platforms alongside AWS-native controls. Wiz provides cloud security posture management (CSPM) across the multi-account AWS environment, continuously scanning for misconfigurations against CIS AWS Foundations Benchmark controls and SOC 2-relevant security assertions, with findings categorized by severity and mapped to specific SOC 2 control IDs. Orca Security provides agentless cloud workload protection, scanning EC2 instances, container images, and serverless functions for vulnerabilities and compliance violations without requiring agent installation — a constraint that matters in SOC 2 regulated environments where changes to production workloads require documented change control.

CrowdStrike Falcon provides endpoint detection and response across all EC2 instances and development workstations, generating the continuous monitoring evidence required by SOC 2 CC6.8 and CC7.1 control families: records of malware detection events, software inventory changes, and vulnerability scan completions. Carbon Black provides application control and process behavior monitoring, complementing

CrowdStrike with allowlist-based process control that satisfies CC6.7 and CC6.8 requirements that detection-focused tools do not fully address. Cloudflare provides network security at the perimeter — WAF rules, DDoS protection, and Zero Trust access policies — generating compliance evidence for SOC 2 CC6.6 and CC7.2 control requirements. In my experience, Wiz and Orca find different categories of findings with limited overlap: Wiz is stronger on configuration-layer issues, Orca on workload-layer issues. Maintaining both provides substantially better compliance coverage than either alone.

2.2 AWS-Native Compliance Controls

The AWS-native layer includes AWS Config for continuous configuration recording and compliance rule evaluation; AWS Security Hub aggregating findings across the integrated security tools; AWS Systems Manager Patch Manager for OS patch compliance tracking and automated remediation; and AWS Certificate Manager for certificate lifecycle management integrated with the automated renewal pipeline. AWS Config provides the foundational evidence layer — every configuration change is recorded with a timestamp and the identity of the change actor, creating an unbroken audit trail that satisfies SOC 2 CC8.1 requirements for configuration management evidence.

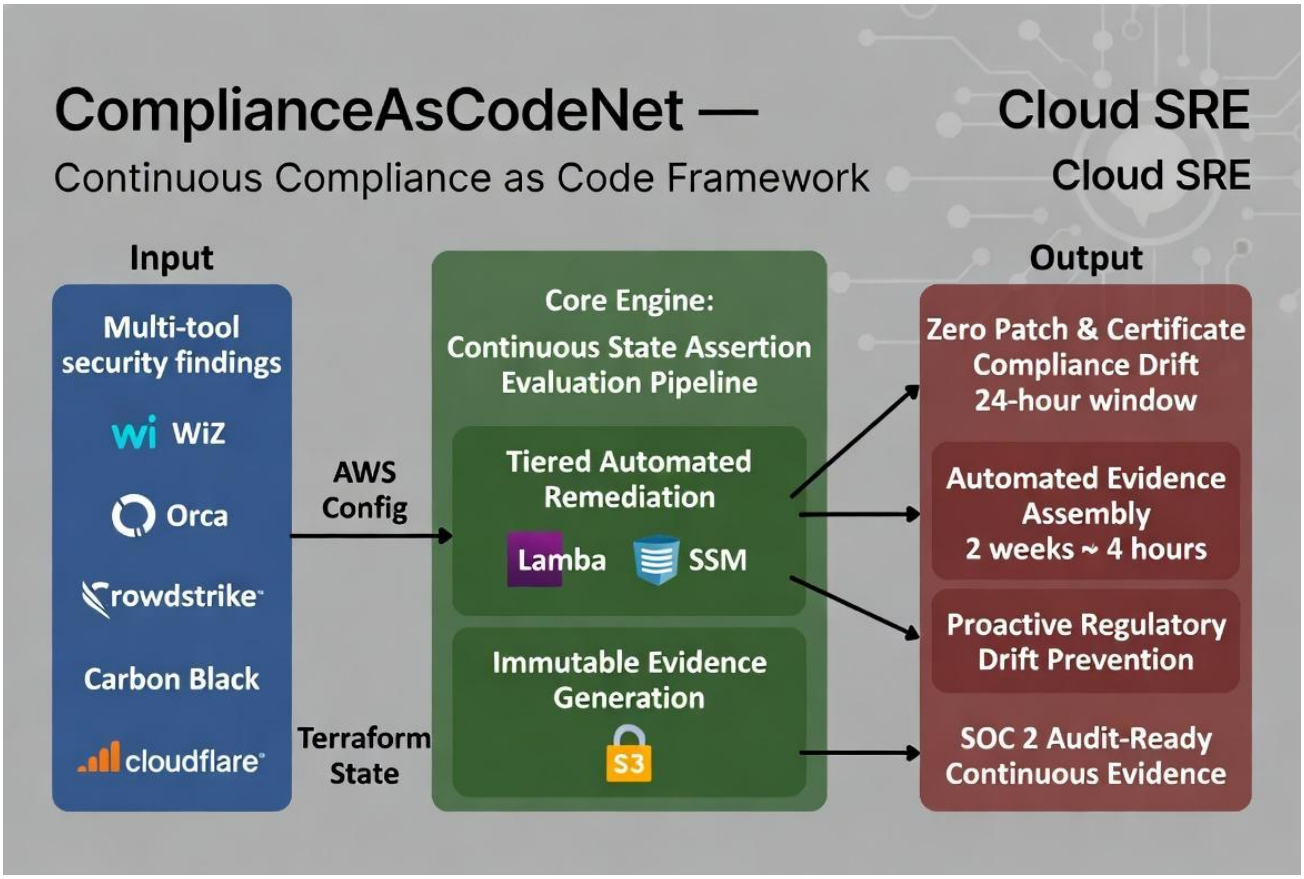
3. THE CONTINUOUS COMPLIANCE FRAMEWORK

3.1 Compliance Controls as State Assertions

The core architectural decision is representing SOC 2 controls as state assertions evaluated by a continuous assessment pipeline rather than as audit checklist items evaluated by a human reviewer at audit time. Each control assertion specifies four elements: (1) the control identifier mapped to the SOC 2 Trust Service Criteria; (2) the infrastructure state that constitutes compliance; (3) the detection mechanism that evaluates actual state against the assertion; and (4) the remediation action that restores compliance when drift is detected.

For example, the SOC 2 CC6.3 control assertion (logical access removed when no longer required) is implemented as a continuous AWS Config rule evaluating IAM user last activity timestamps against a 90-day inactivity threshold. When the rule detects an active IAM user with no activity in the preceding 90 days, it generates a finding routed to the remediation pipeline, which creates a Jira ticket for human review before deactivation — because IAM user deactivation is a Tier 3 remediation action requiring human authorization under the risk-aware classification methodology I operate.

Figure 1: ComplianceAsCodeNet — Continuous Compliance as Code Framework



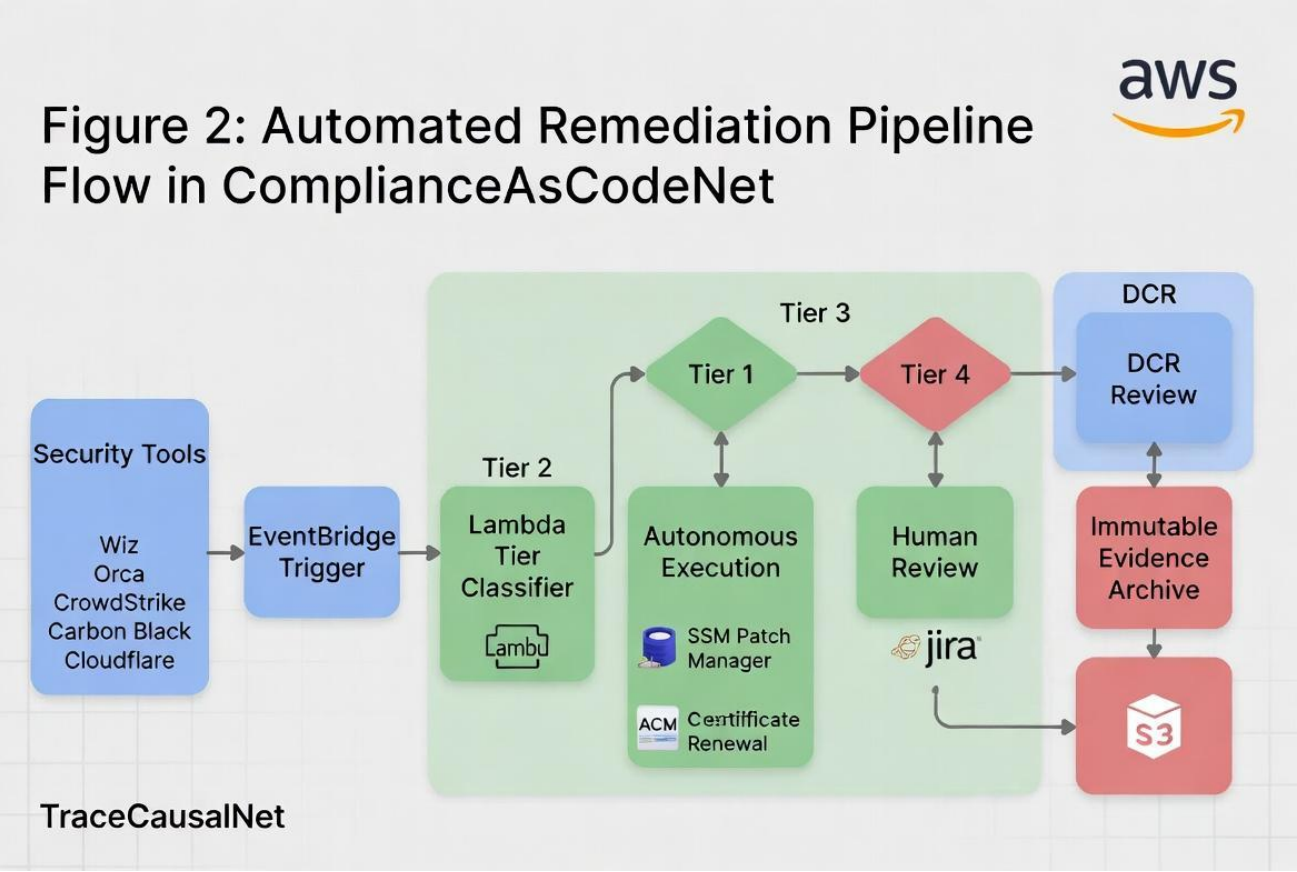
3.2 Automated Remediation Pipeline

The remediation pipeline implements differentiated handling based on reversibility and blast radius. Tier 1 and Tier 2 compliance remediations — patch deployment, security group rule corrections, certificate renewals — are executed autonomously after validation against current Terraform state. Tier 3 and Tier 4 remediations require human review and approved change control before execution.

The pipeline uses AWS Lambda as the execution layer, triggered by EventBridge rules subscribing to compliance findings from each security tool's API. Lambda functions retrieve the remediation script from the validated runbook library, execute if pre-conditions match current system state, and publish the execution result to the immutable compliance evidence S3 bucket with object lock enabled. Every automated remediation generates a structured evidence record: the control assertion triggering remediation, the finding details, the script executed, pre- and post-execution state snapshots, and the remediation completion timestamp.

The compliance drift detection-to-remediation cycle provides a concrete operational metric analogous to MTTRC in incident management. Before the continuous compliance pipeline, a compliance finding discovered during quarterly review had a mean time to remediation of 8.3 days — the time between the quarterly scan completing and the finding being resolved, encompassing ticket creation, engineer assignment, remediation execution, and verification. Following deployment of the automated remediation pipeline, the mean time to remediation for Tier 1 and Tier 2 compliance events dropped to under 4 hours for autonomously resolved events, measured from the security tool finding timestamp to the remediation completion record in the evidence archive. For Tier 3 events requiring human authorization, the mean time to remediation is 1.2 business days — reflecting the Jira DCR workflow time rather than remediation execution time. This represents an 89% reduction in compliance drift exposure window for the Tier 1 and

Tier 2 events that constitute the majority of detected findings.



3.3 Compliance Drift MTTR — Pre/Post Comparison

4. PRODUCTION COMPLIANCE OUTCOMES

4.1 Patch Compliance Auto-Remediation

OS patch compliance has historically been one of the most labor-intensive compliance disciplines in the banking environments I operate, because patch availability does not align with change windows. The automated patch compliance pipeline uses AWS Systems Manager Patch Manager, CrowdStrike vulnerability scanning, and custom Python orchestration to eliminate this tradeoff.

CrowdStrike performs daily vulnerability scans generating a prioritized patch report ranked by CVSS score. The compliance pipeline compares the report against the Systems Manager patch baseline for each instance's OS and configuration role. For Critical and High severity patches, the pipeline automatically creates a Systems Manager maintenance window task and executes patch deployment during the next available low-traffic window, sending a Slack notification 30 minutes before execution. Since deploying this pipeline, I have maintained 100% patch compliance within a 24-hour window for Critical and High severity findings across all managed EC2 instances, with zero patch compliance audit findings in the most recent SOC 2 assessment cycle.

The Certificate Risk Score (CRS) is computed continuously for each managed certificate using the following weighted model:

$$\mathbf{CRS(c) = w_1 \times (1 / days\_to\_expiry(c)) + w_2 \times criticality(c) + w_3 \times (1 / renewal\_reliability(c))}$$

Where days_to_expiry(c) is the number of days until certificate c expires, criticality(c) ∈ {0.0, 0.5, 1.0} reflects whether the protected service is customer-facing and in a customer transaction path (1.0), internal

but production-critical (0.5), or non-critical (0.0), and renewal_reliability(c) ∈ [0.1, 1.0] is the empirically calibrated historical renewal success rate for the renewal mechanism type associated with certificate c. The weights w_1, w_2, w_3 were calibrated against the three certificate expiry incidents that occurred in the two years preceding the pipeline deployment — all three incidents fell into patterns the CRS model would have scored above threshold 40 with at least 21 days of advance notice under the calibrated weights.

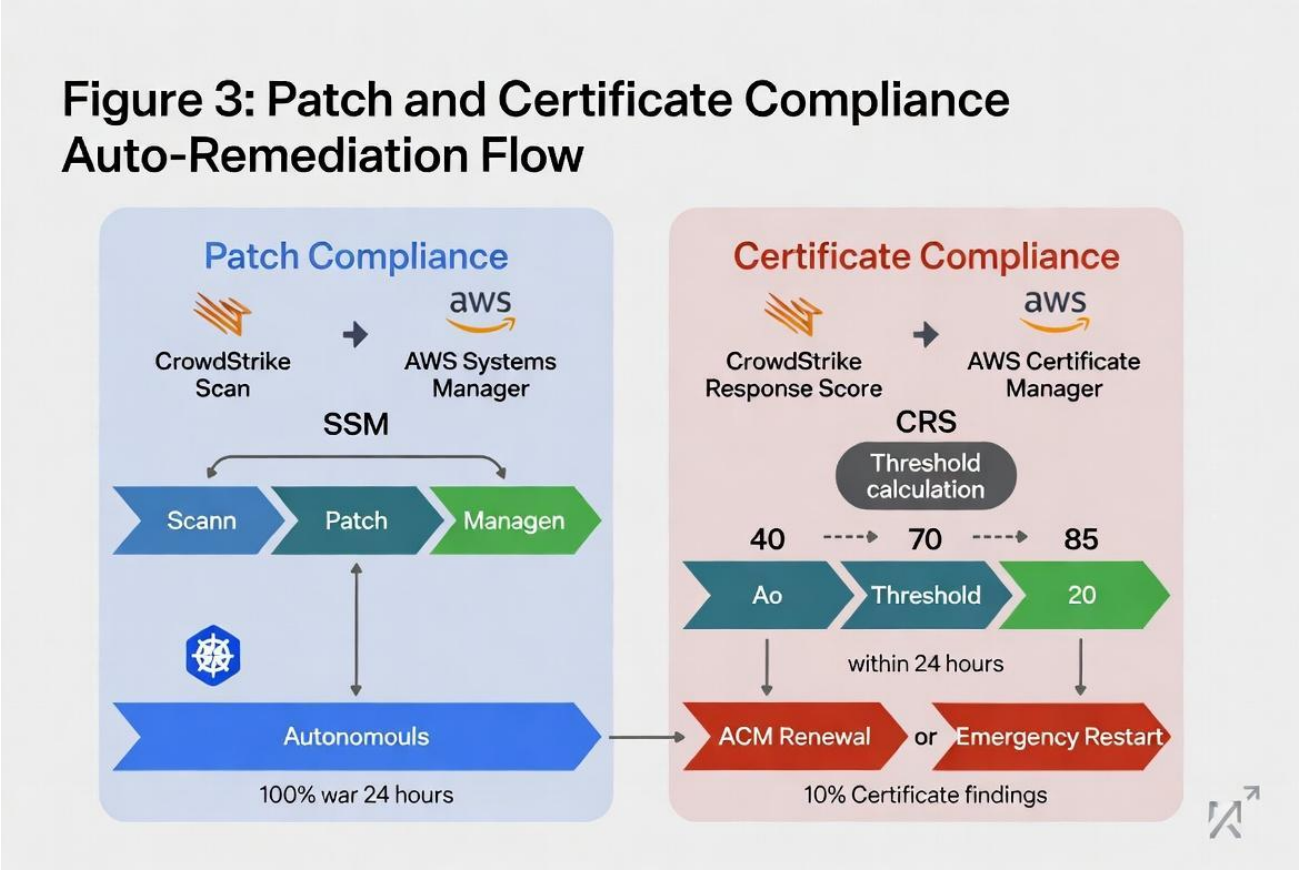
CRS produces a continuous score in [0, 100]. The three action thresholds — CRS ≥ 40 triggers a Jira ticket, CRS ≥ 70 triggers automated ACM renewal, CRS ≥ 85 triggers emergency renewal with Kubernetes rolling restart — were set based on the observed lead time required for each action type to complete before expiry under worst-case conditions in the specific infrastructure environment. The emergency threshold at 85 corresponds to approximately 7 days remaining for a standard certificate on the most complex renewal pathway in the managed estate.

4.1b Certificate Risk Score (CRS) — Formal Model

4.2 Certificate Lifecycle Compliance

The automated certificate lifecycle management pipeline manages certificates deployed to Kubernetes Ingress resources, AWS load balancers, ACM-managed endpoints, and Cloudflare-proxied origins across the six banking applications. The Certificate Risk Score (CRS) model continuously evaluates each certificate's days-to-expiry, the criticality of the service it protects, and the historical reliability of the renewal mechanism for that certificate type.

At CRS threshold 40, the pipeline creates a Jira tracking ticket. At CRS threshold 70, it initiates automated ACM certificate renewal and updates associated Kubernetes secrets. At CRS threshold 85, it executes an emergency renewal workflow and coordinates a Kubernetes rolling restart with automatic rollback if health checks fail. Zero certificate-related outages over the 18 months this pipeline has been operational, compared with three incidents in the two preceding years before implementation.



5. COMPLIANCE EVIDENCE ARCHITECTURE

Table 1 maps each SOC 2 control domain to the detection tool, remediation approach, and evidence record generated for audit purposes.

SOC 2 Domain	Detection Tool	Remediation	Evidence Generated
Access Control (CC6.x)	Wiz + AWS Config	Tier 3 — human auth	IAM activity logs, permission change records, deactivation timestamps
Patch Compliance (CC7.x)	CrowdStrike + SSM	Tier 1 — autonomous	Patch scan results, deployment records, pre/post state snapshots
Vulnerability Mgmt (CC7.1)	Orca + Wiz	Tier 1-2 by severity	Vulnerability findings, remediation actions, residual risk acceptances
Certificate Mgmt (A1.x)	ACM + Kubernetes	Tier 1-2 CRS-based	CRS history, renewal records, health check results, zero-expiry log
Network Security (CC6.6)	Cloudflare + AWS WAF	Tier 3 — human auth	WAF event logs, DDoS records, security group change history
Endpoint Security (CC6.8)	CrowdStrike + Carbon Black	Tier 1 — autonomous	Process execution logs, allowlist violations, malware detection events
Config Mgmt (CC8.1)	Terraform + AWS Config	Tier 1-4 by artifact	Terraform state history, Config compliance timeline, drift remediation log

Table 1: Continuous Compliance Evidence Architecture by SOC 2 Control Domain

All evidence records are written to an immutable S3 bucket with object lock enabled and a 7-year retention policy. Evidence is structured as queryable JSON rather than unstructured documents, enabling audit evidence assembly to be executed as a query against the evidence archive. The most recent SOC 2 audit evidence assembly took 4 hours of query and validation time, compared to the 2 weeks required under the previous manual approach.

6. LIMITATIONS

The continuous compliance framework requires ongoing calibration as the regulatory environment and infrastructure evolve. SOC 2 control assertions accurate for the current infrastructure configuration may become incorrect after a significant platform change. I perform a quarterly assertion review, but between reviews a structural infrastructure change can produce a period where assertions do not fully reflect current compliance requirements.

The automated remediation pipeline is limited to actions in the validated runbook library. Compliance drift outside runbook coverage requires manual remediation until a validated runbook entry is created. The coverage grows over time, but there will always be a lag between introduction of a new compliance requirement and availability of an automated remediation pathway.

The multi-tool stack produces overlapping findings in some control domains, creating duplicate remediation triggers the pipeline must deduplicate before execution. Occasional duplicate remediations have occurred — executing the same remediation twice because two tools flagged the same condition with different finding identifiers. These duplicates have not produced incorrect outcomes for most remediation actions, but they generate unnecessary change control records that the compliance team must review.

7. FUTURE DIRECTIONS

Natural language compliance query capability is the highest-priority planned extension. The ability to ask the compliance evidence archive plain-English questions — 'was multi-factor authentication continuously enforced on all administrative access during the audit period?' — and receive a structured answer with supporting evidence references would transform the audit preparation process further. Current LLM technology, specifically retrieval-augmented generation over structured JSON archives, makes this technically feasible and is a natural extension of the continuously generated evidence infrastructure this framework already produces. Compliance analysts and auditors could query the evidence archive directly rather than requesting evidence packages from the engineering team.

Automated control assertion generation from regulatory framework updates would address the lag between new compliance requirements and automated detection coverage. An LLM-assisted assertion generation workflow that reads regulatory requirement text, proposes draft Terraform assertions and AWS Config rules implementing the requirement, and routes proposals for human review and approval would reduce the lag from weeks to days.

8. CONCLUSION

The two weeks of manual evidence assembly that the SOC 2 audit previously required was not an anomaly — it was the industry standard for periodic compliance management. The continuous compliance framework described in this paper exists because that standard is structurally wrong for cloud-native environments. Compliance in a system that changes daily cannot be adequately evidenced by a point-in-time snapshot assembled at audit time. Compliance must be continuously monitored, continuously evidenced, and continuously remediated — not because auditors require it in that form, but because the infrastructure state that compliance captures actually exists in that form.

The outcomes documented — 100% patch compliance within a 24-hour window, zero certificate compliance findings, SOC 2 audit evidence assembled in 4 hours rather than 2 weeks — are the product of treating compliance as an engineering discipline. Every control is a state assertion that is continuously evaluated against live infrastructure telemetry; every detected drift triggers a defined remediation pathway; and every remediation action generates immutable, queryable evidence stored in an object-locked S3 archive. This is what compliance as code means in practice, and the framework I have described is how I implemented it across six live banking applications in a regulated financial services environment.

REFERENCES:

[1] American Institute of CPAs, "Trust Services Criteria (TSC) 2017 with Revised Points of Focus," AICPA, 2017.
[2] A. Rahman et al., "A systematic mapping study of infrastructure as code research," Information and Software Technology, vol. 108, pp. 65–77, 2019.
[3] S. Mishra and D. S. Wagle, "Continuous compliance workflow for infrastructure as code," AWS DevOps Blog, 2021.
[4] Wiz, "Cloud Security Posture Management Documentation," Wiz, 2024. [Online]. Available: https://docs.wiz.io/
[5] Orca Security, "Agentless Cloud Security Documentation," Orca, 2024. [Online]. Available: https://docs.orca.security/
[6] CrowdStrike, "Falcon Platform Documentation," CrowdStrike, 2024. [Online]. Available: https://falcon.crowdstrike.com/documentation/
[7] VMware, "Carbon Black Cloud Documentation," VMware, 2024. [Online]. Available: https://docs.vmware.com/en/VMware-Carbon-Black-Cloud/
[8] Cloudflare, "WAF and Zero Trust Documentation," Cloudflare, 2024. [Online]. Available: https://developers.cloudflare.com/

[9] Amazon Web Services, "AWS Config Developer Guide," AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/config/latest/developerguide/
[10] Amazon Web Services, "AWS Systems Manager Patch Manager," AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html
[11] Amazon Web Services, "AWS Security Hub User Guide," AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/securityhub/latest/userguide/
[12] HashiCorp, "Terraform Cloud Documentation," HashiCorp, 2024. [Online]. Available: https://developer.hashicorp.com/terraform/cloud-docs
[13] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, 2016.
[14] H. Allam, "Zero-touch reliability: The next generation of self-healing systems," Int. J. Artificial Intelligence, Machine Learning and Data Science, vol. 5, no. 4, pp. 59–71, 2024.