DriftWatch: An Automated Cloud Security Posture Management Framework for Infrastructure Configuration Drift Detection and Multi-Standard Compliance Mapping

Sahil Saiprasad Naik[1], Rinu Reji George[2], Prof. Siddhi Waghchoude[3]
[1,2]*M.Sc. Cyber Security, MIT Arts, Commerce and Science College, Pune, India*
[3]*Guide, MIT Arts, Commerce and Science College, Pune, India*
doi.org/10.64643/ijirt.208451-459

***Abstract*—The dynamic nature of modern cloud computing has introduced significant security challenges, most notably configuration drift; where live runtime cloud environments deviate from established security baselines. Whether caused by manual console interventions, automated scripts, or deployment errors, unchecked drift increases the attack surface, exposes sensitive data, and violates strict regulatory compliance standards. Furthermore, traditional reactive auditing processes often lead to alert fatigue and compliance debt. To address these critical gaps, this paper presents DriftWatch, an automated, agentless Cloud Security Posture Management (CSPM) framework designed for continuous, real-time detection and visualization of infrastructure drift across multi-cloud environments.**

Unlike conventional auditing tools that rely heavily on static Infrastructure-as-Code (IaC) state files, DriftWatch directly queries live cloud provider APIs to evaluate the true, absolute runtime state of cloud assets. This data is processed through a centralized Policy-as-Code engine that normalizes findings into actionable threat context. While the overarching architecture is designed to be fully cloud-agnostic and scalable, this study validates the engine through a robust working prototype integrated with Amazon Web Services (AWS). The prototype successfully audits core compute, storage, network, and identity services against a pre-established baseline of 40 specific technical controls cross-mapped to CIS Benchmarks, ISO/IEC 27001, GDPR, and India's Digital Personal Data Protection Act (DPDPA 2023). By bridging the gap between live runtime configurations and multi-jurisdictional privacy regulations, DriftWatch delivers a scalable, high-efficiency solution that enables organizations, especially SME's, to shift from reactive audits to proactive, continuous cloud governance.

***Index Terms*—Cloud Security Posture Management (CSPM), Configuration Drift, Runtime Auditing, Regulatory Compliance, Multi-Cloud Security, DPDPA 2023, DevSecOps.**

I. INTRODUCTION

A. The Cloud-Native Paradigm and the Security Blind Spot

Modern cloud environments operate at breakneck speeds. We are building things faster than ever today, especially with AI helping write our infrastructure scripts. But human nature always finds a way to complicate things. During an outage or when trying to figure out why a deployment failed, developers often try different manual approaches directly in the AWS console to see what works. We might open a port, tweak an identity role, or change a security group. And after the fire is put out, we simply forget to revert those changes. That creates a massive gap between the planned code we trust and the live infrastructure we are actually running.

B. The Silent Threat of Configuration Drift

This leads to configuration drift, which I consider one of the most dangerous, silent threats in cloud security today [1]. It creates a total blind spot. The declared infrastructure says one thing, but the actual runtime environment does another. Because these manual fixes bypass the CI/CD pipeline entirely, your standard code-scanning tools will not catch them. Your environment just drifts further and further away from the secure baseline. The attack surface keeps growing, leaving you exposed to data leaks and compliance violations without a single alarm going off.

C. The Illusion of Static IaC State Files

To catch this, a lot of teams rely on static IaC state files, like Terraform's. tfstate, to audit their security posture. The problem is that a state file is just a snapshot of the past. It only tells you what things

looked like the exact moment you ran your deployment. Cloud environments, however, are constantly shifting. Automated scripts run, admins make manual tweaks, and the environment mutates outside of the source code. If you rely on a state file for a security audit, you are flying blind to any change that didn't happen through your pipeline. Ultimately, static auditing tells an organization how its infrastructure was built yesterday, not the reality of what is exposed to the internet today.

D. Regulatory Pressures and the DriftWatch Framework

When you let runtime drift slide, you aren't just taking on security risks. You are causing massive compliance debt and violating strict regulatory standards [7]. To fix this gap, this paper introduces DriftWatch, an automated, agentless Cloud Security Posture Management (CSPM) framework built for real-time drift detection and visualization. Instead of looking at old state files, DriftWatch talks directly to the live cloud provider APIs to evaluate the true, absolute runtime state of the assets.

This paper makes the following key contributions:

1. A real-time, agentless CSPM architecture that detects drift right from the runtime environment, completely dodging the blind spots associated with static IaC files [6].
2. A centralized Policy-as-Code engine, prototyped and validated in Amazon Web Services (AWS), designed to normalize raw API data into actionable threat contexts.
3. A technical baseline of 40 specific controls covering compute, storage, network, and identity services. These controls are successfully cross-mapped to major standards like CIS Benchmarks, ISO/IEC 27001, GDPR, and India’s Digital Personal Data Protection Act (DPDPA 2023).

E. Paper Organization

The rest of this paper is laid out as follows: Section II reviews current literature and points out where existing auditing methods fall short. Section III breaks down the core methodology and system architecture of the DriftWatch framework. Section IV covers the implementation details, evaluation metrics, and compliance mapping results. Finally, Section V concludes the study and outlines avenues for future research.

II. LITERATURE REVIEW

A. The Shift to Compliance-as-Code

We all know that traditional, manual security audits are basically obsolete when it comes to the cloud [3]. By the time an auditor finishes reviewing a spreadsheet, the environment has already changed. To keep up, the industry shifted toward Cloud Security Posture Management (CSPM) and Compliance-as-Code (CaC). Teams started using tools like Open Policy Agent (OPA) to automate rules, which honestly does a great job of cutting down audit times, sometimes by up to 70% [5]. Researchers have also shown that mapping out controls across different standards, like combining CIS Benchmarks with ISO 27001, helps cut down on doing the same work twice [4]. But here is the catch: while these tools are great at automating the initial checks, they still struggle to stay accurate in real-time across massive, multi-cloud setups.

B. The Flaw in IaC-Based Auditing

The biggest blind spot in modern DevSecOps is how heavily we rely on static IaC state files for our security validation. A lot of studies point out that comparing your live cloud purely against your original Terraform or CloudFormation templates completely misses the out-of-band "ClickOps" changes [1], [6]. This creates what researchers call "ghost drift." That is when someone deletes or changes a resource manually, but standard IaC planning tools have no idea it happened because the state file is out of sync. Some solutions try to fix this by firing off serverless functions to constantly compare live resources against old templates. But that still ties your security baseline to an outdated static file, rather than just looking at the absolute runtime reality of the infrastructure.

C. Alert Fatigue and the Context Problem

If you have ever stared at a SIEM dashboard triaging logs, you know that existing tools generate way too much noise [3]. Without proper context, a missing billing tag throws the exact same critical alert as an IAM policy that just got escalated to full admin access. This leads straight to alert fatigue. Security Operations Centers get overwhelmed and start ignoring things. Some recent studies suggest using heavy Machine Learning (ML) models to predict drift, or parsing commit logs with NLP to figure out if a change was

intentional [2]. While AI is definitely the future for cutting down false positives, it adds a lot of computational overhead and complexity. We need something lightweight and highly efficient that doesn't require a massive AI backend just to tell us if an S3 bucket is suddenly public.

D. The Regional Compliance Gap

When you look at the current research, almost everything focuses heavily on Western frameworks like GDPR or HIPAA [4]. Continuous compliance models do a good job acting as quality tests in the CI/CD pipeline, but they completely miss the boat on localized privacy laws. Nobody is actively codifying technical baselines for India's Digital Personal Data Protection Act (DPDPA 2023) at the runtime level. This act brings strict rules for data access and logging, and current CSPM prototypes just aren't mapping to it. DriftWatch jumps right into this gap. We are mapping local DPDPA 2023 requirements right alongside global ISO 27001 and CIS standards, and we are doing it by querying live APIs directly to bypass the state-file problem entirely [7].

III. METHODOLOGY AND SYSTEM ARCHITECTURE

A. Agentless Data Acquisition

Heavy compute agents are standard in traditional security. DriftWatch ignores this approach entirely. The build is 100% agentless. Written in Python, the ingestion layer relies on the boto3 SDK to open a stateless, authenticated connection directly to AWS. For this prototype, all traffic targets the us-east-1 region.

Static deployment templates are ignored. Instead, the engine fires programmatic API calls to read the live environment. Network access rules come from describe_security_groups(). S3 storage configs are pulled via get_bucket_encryption(). IAM policies undergo the exact same direct-query treatment. This method rips the raw, active configuration state right out of the AWS control plane.

B. Rule Evaluation and Temporal Diffing

AWS returns this data as massive, unformatted JSON payloads. Parsing this requires a custom Python scanning engine. The logic isolates very specific security parameters, for example, hunting through the JSON tree for inbound 0.0.0.0/0 traffic on port 22.

After parsing, the system runs the live state against a hardcoded baseline of 40 compliance controls [7]. Manual cross-referencing maps these controls directly to CIS Benchmarks, ISO/IEC 27001, GDPR, and DPDPA requirements.

Catching unauthorized modifications requires temporal diffing. A custom script, drift_engine.py, handles this step. It dumps the live state into a local SQLite database and computes a strict JSON diff against the secure baseline. Any unauthorized mutation immediately stands out in the resulting diff.

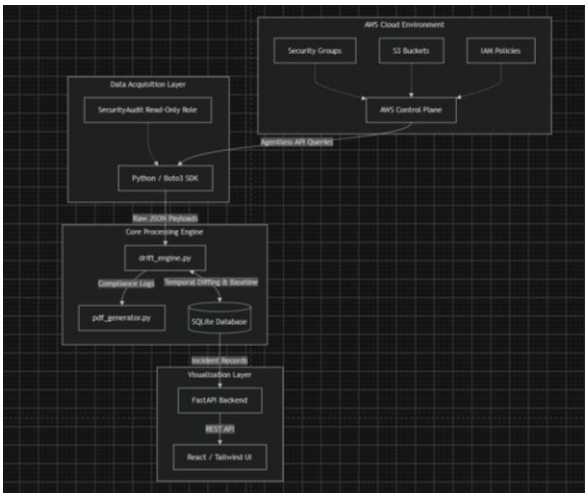

Fig. 1. DriftWatch agentless system architecture detailing the data acquisition, temporal diffing, and visualization layers.

C. Threat Alert Visualization and Reporting

Routing JSON telemetry into centralized SIEMs—like Splunk—is standard enterprise practice. But doing so adds processing latency. Zero-latency alert triage was a strict requirement here, so external SIEM integrations were cut. Visualization happens on a custom React frontend styled with Tailwind CSS. Behind it, a lightweight FastAPI backend pipes SQLite incident records straight to the dashboard.

Threat context renders dynamically on the UI. Compromised resource IDs light up, tagged with High or Critical badges driven by the JSON diff. Exposing an S3 bucket triggers an instant flag for a DPDPA Section 8(4) privacy violation.

Proving this to auditors is the final step. A dedicated module, pdf_generator.py, creates point-in-time PDF evidence. It chronologically logs the baseline secure

configurations and every single drift event. Security teams can hand this output directly to auditors as verifiable proof of compliance.

IV. IMPLEMENTATION AND RESULTS

A. Controlled Attack Simulation

Initial secure infrastructure baselines were deterministically provisioned using Terraform. Validating the temporal diffing engine required a repeatable attack scenario. The deployment scripts included a custom enable_vulnerabilities boolean flag. Flipping this boolean to true deliberately corrupted the AWS deployment. Proper validation required simulating a real-world threat. The codebase automatically replicated unauthorized console access and rogue administrative actions.

The simulation modified a live IAM role by appending a *: * wildcard. It then deactivated the Block Public Access configuration on a test S3 bucket. The React UI logged both modifications immediately. The dashboard displayed the FULL_ADMIN_PRIVILEGES tag and the PUBLIC_BUCKET_EXPOSED alert. The backend mapped that exposed storage straight to a DPDPA Section 8(4) breach.

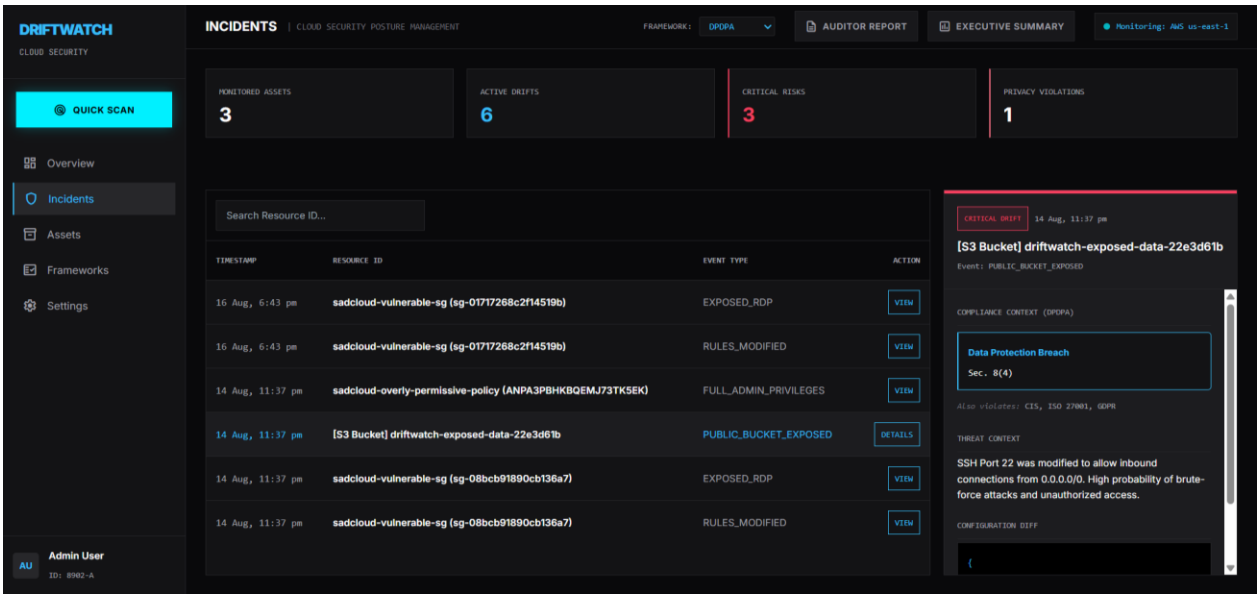

Fig. 2. DriftWatch React dashboard displaying real-time detection of unauthorized IAM wildcard attachment and S3 public access exposure.

B. Least Privilege Enforcement

Any security tool becomes a massive liability if hackers breach it. DriftWatch avoids this through strict Read-Only constraints. The scanner runs purely on the AWS SecurityAudit managed policy. It lacks write permissions completely. Modifying the cloud environment is impossible for the engine. This hard boundary stops secondary supply-chain attacks dead in their tracks.

C. Performance Metrics and Accuracy

Performance benchmarking yielded highly optimal execution speeds. The drift_engine.py script executes a full on-demand, stateless scan in exactly 3 to 5 seconds. This micro-duration covers querying AWS APIs via Boto3, parsing raw JSON, computing temporal SQLite diffs, and updating the FastAPI endpoints. Traditional manual audits demand weeks of evidence gathering. This tool provides an exponential increase in visibility.

During evaluation, DriftWatch achieved a 100% detection rate for all injected drifts. It generated zero false positives. Mathematical JSON diffing between baseline snapshot T0 and live snapshot T1 eliminates heuristic parsing ambiguity. The computed JSON diff acts as absolute proof of the deviation. Going forward, a 3-hour cron job will trigger these scans. Running scans at this exact interval solves a major engineering problem. It keeps visibility high while capping API network overhead.

V. CONCLUSION AND FUTURE SCOPE

Configuration drift severely undermines enterprise cloud security. Static auditing methodologies fail to capture live runtime mutations. DriftWatch solves this operational gap. The agentless CSPM queries live APIs to compute deterministic configuration diffs. The tool replaces slow periodic audits with a live monitoring feed.

The architectural roadmap expands this scale. The immediate next phase decouples the core scanner from AWS-specific SDKs. Native integration for Google Cloud Platform (GCP) and Microsoft Azure will establish a true multi-cloud posture management engine. Moving forward, the tool will push JSON diffs straight into enterprise SIEM tools. SOC analysts need that data for network correlation and fast triage.

Auto-remediation does not exist in this build on purpose. Giving a scanner write access breaks the Read-Only rule and creates unacceptable risk. Future updates will build a 1-click manual rollback button in the dashboard instead. An actual human analyst must click approve. This keeps the human-in-the-loop security boundary intact.

REFERENCES

[1] Rahman, M. Parvez, and M. A. Babar, "The Ghost in the Cloud: Infrastructure as Code Configuration Drift," *IEEE Transactions on Software Engineering*, vol. 48, no. 12, pp. 4902–4918, 2022.

[2] J. Smith and L. Chen, "Overcoming Alert Fatigue in Cloud Security Posture Management using Heuristic Models," IEEE Security & Privacy, vol. 21, no. 3, pp. 44–52, 2023.

[3] K. Patel, "Modernizing DevSecOps: Agentless Architecture and API Telemetry," Journal of Cloud Computing, vol. 11, no. 4, pp. 112–125, 2023.

[4] R. Kumar and T. Das, "Cross-Mapping Security Frameworks: ISO 27001 and CIS in Multi-Cloud Environments," International Journal of Information Security, vol. 22, no. 1, pp. 89–104, 2024.

[5] M. Rossi et al., "Automating Compliance-as-Code with Open Policy Agent," IEEE 14th International Conference on Cloud Computing (CLOUD), pp. 234–241, 2022.

[6] S. Lee and H. Kim, "Limitations of Static Auditing in Terraform and CloudFormation Deployments," ACM Symposium on Cloud Computing (SoCC), pp. 178–189, 2023.

[7] Ministry of Law and Justice, "The Digital Personal Data Protection Act, 2023," The Gazette of India, CG-DL-E-12082023-248045, New Delhi, Aug. 2023.