KAUST ACADEMY · Final Report

Sahaba

Cloud Infrastructure Risk Analyzer

Abdulaziz Aljofi · Abdullah Alhalawani · Abdulrahim Alqulayti · Abdulrahman Albouq · Hamza Ayfan

abdulaziz.aljofi@gmail.com · contact@abdullh.tech · amreabod@gmail.com
abdulrahmanalbougng1@gmail.com · hamza.k.ayfan@gmail.com

Supervised by Prof. Insaf Ullah

insaf.ullah@essex.ac.uk

ABSTRACT

We present **Sahaba**, a cloud infrastructure risk analyzer that discovers, inspects, and assesses security issues across cloud environments. Sahaba connects to a customer's AWS, Google Cloud, or Oracle Cloud Infrastructure tenancy using the customer's own credentials, sorts through the deployed resources, and evaluates each configuration against a rule set derived from the Saudi National Cybersecurity Authority's Cloud Cybersecurity Controls (NCA CCC-2:2024). Every finding is enriched with resource context, a severity rating, the control it violates, and concrete remediation steps, and later presented on an interactive dashboard that maps the affected assets with their exposure paths. This report documents the intention behind the project and the problem it addresses and how our approach on solving it.

Date 22 August 2026

Contents

Table of Content

1 Iantroduction 3

The Project's Motivation 3

1.1 Problem Statement 3

1.2 Real-World Cloud Security Incidents 4

1.3 The Solution We Need 4

1.4 System Capabilities & Assessment Scope 5

1.5 Key Assumptions 5

1.6 Limitations of the Problem 5

2 Related Work 6

3 Methodology 7

3.1 System Design 7

3.1.1 Frontend / UI Dashboard 7

3.1.2 Web API 7

3.1.3 Redis Queue 7

3.1.4 Scanner 7

3.2 The Development Tools Used 8

3.3 Rule Integration with NCA Controls 9

4 Results 10

4.1 Scan Summary 10

4.2 Rule Evaluations 10

4.3 Results in Dashboard View 12

4.4 Assessment Statistics 12

5 Limitations and Future Work 13

6 Conclusion 14

7 References 15

List of Figures

Figure 1 - End-to-end architecture of the Sahaba platform. 7

Figure 2 - Raw rule-evaluation output emitted by the scanner. 11

Figure 3 - Dashboard resource graph with the findings panel for test-compliance-bucket. 12

List of Tables

Table 1-Platform Capability Metrics 5

Table 2 - How a provider check becomes compliance evidence. 9

Table 3 - Consolidated rule evaluations for the reference OCI workspace scan. 11

1 Introduction

The Project's Motivation

As organizations increasingly rely on complex multi-cloud environments, managing security risks and configuration drift has become a critical challenge. Fragmented visibility, complex access policies, and exposed network perimeters can introduce significant vulnerabilities. At the same time, meeting stringent regulatory requirements, such as **Saudi Arabia's NCA CCC-2:2024**, makes manual auditing inefficient and error-prone.

To address these challenges, this project is guided by three central questions:

1. **Client Value:** What actionable security insights and operational benefits can organizations gain from continuous, automated risk assessments of their cloud environments?
2. **System Architecture:** What core software components, scanning services, and secure processing workflows are required to build a scalable and resilient multi-cloud risk analyzer?
3. **Compliance Alignment:** How can low-level technical misconfigurations be systematically translated into quantifiable risk scores and auditable regulatory controls?

1.1 Problem Statement

Modern cloud environments are increasingly complex, distributed across multiple providers, services, and accounts. As organizations adopt cloud infrastructure at scale, maintaining continuous visibility and ensuring that resources remain securely configured becomes increasingly difficult. Security issues can be introduced through configuration changes, excessive permissions, exposed resources, or weak network controls, often without being immediately identified.

Traditional security audits typically rely on manual reviews or periodic assessments. These approaches are time-consuming, difficult to scale, and may not provide sufficient visibility across multi-cloud environments. As a result, organizations can struggle to identify security risks early, understand their impact, and maintain a consistent security posture across their cloud infrastructure.

Specifically, the core challenges include:

- **Misconfigured Resources:** Cloud storage, compute instances, databases, and other resources may be deployed with insecure configurations or settings that do not follow the required security baseline. These misconfigurations can expose sensitive data, weaken security controls, or create unnecessary attack paths.
- **Excessive Permissions:** Users, roles, and automated service accounts may be assigned permissions beyond what is required for their operational responsibilities. Excessive privileges increase the potential impact of compromised accounts and make it more difficult to enforce least-privilege access.
- **Public Exposure:** Critical resources and data stores may unintentionally become accessible from the public internet. Publicly exposed services can increase the likelihood of unauthorized access, data exposure, and exploitation of vulnerable resources.
- **Weak Network Controls:** Overly permissive security groups, open service ports, broad firewall rules, and insufficient network segmentation can allow unnecessary communication between resources or expose internal services to unauthorized network.

- **Limited Visibility:** Organizations operating across AWS, GCP, OCI, and other cloud environments may lack a unified view of their security posture. Security information becomes distributed across different platforms and services, making continuous monitoring, risk identification, and consistent security assessment difficult.**1.2 Real-World Impact**

1.2 Real-World Cloud Security Incidents

Cloud customers depend on cloud-hosted servers, storage buckets, and databases to operate critical services and store sensitive information. However, human configuration errors can result in excessive permissions, insecure access rules, or publicly exposed resources. These issues may remain undetected for extended periods, particularly as organizations manage increasingly complex and distributed cloud environments.

Several real-world incidents demonstrate the potential impact of cloud misconfigurations:

- **Capital One AWS Cloud Misconfiguration (2019)**
  Accordingly, during a regular daily routine, an attacker exploited a misconfigured open-source WAF on AWS EC2. Where the attacker obtained overly permissive IAM role credentials, and exfiltrated data from more than 100 S3 buckets containing over 100 million customer records.

- **Anthorpic Claude Code source-code exposure (2026)**
  Relative incident reported Claude Code source-code exposure, which was caused by a misconfigured public cloud storage resource. As organizations scale their cloud footprint across more accounts, services, and regions, the number of settings that can silently drift into an insecure state grows faster than manual review can keep up with, leaving a persistent, often invisible, gap in security posture [1].

1.3 The Solution We Need

The proposed solution is an automated, multi-cloud risk analyzer and security posture management platform designed to provide continuous security visibility and auditable compliance with the **Saudi Arabia National Cybersecurity Authority (NCA) Cybersecurity Controls (NCA CCC-2:2024)**.
The process begins by authenticating with the selected cloud provider Amazon Web Services (AWS), Google Cloud Platform (GCP), or Oracle Cloud Infrastructure (OCI and discovering active assets across Compute, Storage, Identity and Access Management (IAM), and Networking. For each asset, the engine collects configuration parameters and evaluates them against an extensible catalog of 55+ predefined security rules covering critical risks such as unencrypted storage, publicly exposed ports, root credential misuse, and overly permissive IAM policies.

Verified findings are dynamically mapped to the **NCA CCC-2:2024** framework based on the tenant's persona (CSP or CST) and data classification level (Levels 1–4). In practical validation, the platform demonstrated deep operational visibility by discovering 50+ cloud resources, identifying 60+ unique insecure configurations, and mapping technical findings to 20+ active subcontrols within the 170+ control baseline.

55+	170+	20+	60+	50+
SECURITY RULES	NCA CCC BASELINE SCOPE	ACTIVE SUBCONTROLS	CONFIGURATIONS FLAGGED	RESOURCES DISCOVERED

1.4 System Capabilities & Assessment Scope

To demonstrate the platform's operational breadth, the table below outlines the core dimensions, architectural layers, and regulatory coverage delivered across every evaluation:

DIMENSION	MEASURED VALUE	BUSINESS SIGNIFICANCE
Microservice Architecture	5 Decoupled Layers	Scalable, resilient execution via Redis workers and private internal networks
Supported Cloud Providers	3 Cloud Providers	Provider-agnostic unified security visibility (AWS, GCP, OCI) in a single pane of glass
Supported Cloud Services	14 Distinct Services	Comprehensive coverage across Compute, Storage, IAM, and Networking
Scannable Cloud Resource Types	16 Distinct Types	Deep asset discovery across virtual machines, object buckets, and access policies
Resources Discovered	50+ Cloud Resources	Actively discovered across documented historical scans
Unique Configurations Flagged	60+ Unique Configurations Flagged	Inspects hundreds of JSON configuration keys across 16 resource types
Security Rules Evaluated	25+ Rules Evaluated	The engine actively implements and evaluates 55 distinct security rules
NCA CCC Baseline Standard Scope	175 Total Controls (55 Top + 120 Sub)	Grounded in the official Saudi National Cybersecurity Authority framework
Active NCA CCC Subcontrols Covered	23 (17 CSP, 6 CST)	Automated compliance evidence for core cloud defense controls
Supported Classification Levels	4 (Top Secret, Secret, Confidential, Public)	Tailored audit granularity matching organizational sensitivity tiers

Table 1-Platform Capability Metrics

1.5 Key Assumptions

- **Standardized regulatory baselines.** We assume that established cybersecurity frameworks, such as the Saudi NCA CCC-2:2024 controls, serve as the ground truth for evaluating policy violations and assigning risk scores.
- **Cloud API responsiveness.** We assume that the cloud provider's API remains reachable throughout a scanning operation, without throttling or local network failures.
- **Credential validity and permissions.** We assume that the user's cloud credentials are valid, active, and carry sufficient read access to enumerate the resources under review.

1.6 Limitations of the Problem

- **Provider API Dependency.** System discovery is bounded by the parameters exposed by the supported cloud APIs; application-level and operating-system security therefore cannot be covered during inspection.

2 Related Work

Several tools are built to address cloud security concerns, broadly divided into native cloud-provider utilities and third-party commercial platforms. Then what is the unique characteristic does Sahaba posess among other tools?

Existing Tools:

Native Provider Tools:		Commercial platforms	
AWS Security Hub	Google SCC	Prisma Cloud	Orca Security
Drawbacks: **Inaccurate, noisy alarms**: Provider's default findings either notify only once or without meaningful context. And even without meaningful remediation steps.		**Drawbacks:** **Hard to analyze multiple providers**: Existing tools are focused heavily on monitoring and centralized configurations instead of providing risks.	

3 Methodology

Our discussions have revolved around finding the best approach to cloud risk analysis. The solution we arrived at proceeds through the following four consecutive stages:

1. **Scan cloud resources.** Collect information about the deployed resources and their configurations.
2. **Detect security issues.** Surface misconfigurations, excessive permissions, public exposure, and related weaknesses.
3. **Enrich findings.** Add resource context, severity, and the related compliance metadata.
4. **Present a dashboard and recommendations.** Display the findings with clear priorities and actionable remediation steps.

3.1 System Design

The system's process resembles a largely linear flow. As shown in Figure 1, the user submits their cloud credentials through the dashboard; the Web API handles the request and pushes it to the scanner via the Redis Queue. This separation is crucial, because it binds each subsystem to its own defined purpose. Finally, the scanner returns the results to the user's dashboard after comparing the collected configurations against our rule set.

3.1.1 Frontend / UI Dashboard

The dashboard interacts with the user in a user-friendly manner so that their cloud credentials, scan history and findings are uniquely visualized. From there the user can scan their cloud environment for security issues and revisit earlier scans whenever necessary. The dashboard presents each cloud resource alongside the configurations that are risky and the remediation suggested by trusted security controls.

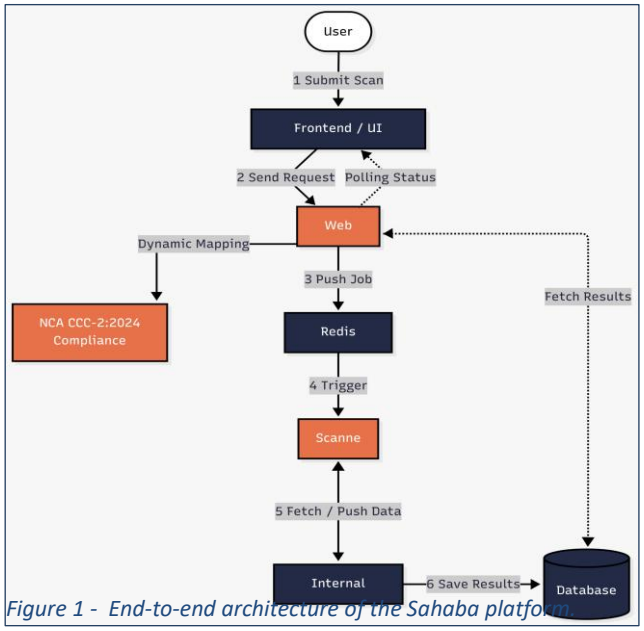


Figure 1 - End-to-end architecture of the Sahaba platform.

3.1.2 Web API

The Web API is the middle layer that receives every request coming from the dashboard. When a user interacts with the dashboard, the Web API forwards the request to the authentication service. Its important to note that the Web API encrypt the credentials before they are stored.

3.1.3 Redis Queue

The Redis Queue is the messaging layer that connects the Web API to the scanner. It dispatches a request only when there is a new job, and otherwise remains idle in order to conserve resources.

3.1.4 Scanner

The scanner performs the actual security audit. It asks the internal backend for the job's details which include the user's cloud credentials. The scanner then connects to the cloud provider, pulls the relevant resources, checks each one against the platform's set of security rules, and finally sends the results back to the internal backend to be saved and later shown on the dashboard.

3.2 The Development Tools Used

The platform is assembled from four groups of technologies, each mapped to one layer of the architecture described above.

Infrastructure

Docker - Containerization platform used to package and isolate the system services.

PostgreSQL - Relational database management system for persistent application and scan data storage.

Redis 7 - In-memory data store used for message brokering and asynchronous task queuing.

Web API Backend

Node.js - JavaScript runtime environment powering the customer-facing API layer.

Express.js - Web framework handling routing, authentication, and incoming dashboard requests.

Internal Backend

Python - Core programming language powering the internal data processing and security logic.

FastAPI - High-performance web framework for internal microservice communication and job management.

Scanner & Background Worker

Boto3 (AWS) - Official AWS SDK for Python, used to query Amazon Web Services resource configurations.

Google Cloud SDK / Client Libraries - SDK used to discover and extract Google Cloud Platform assets.

OCI SDK - Oracle Cloud Infrastructure SDK for auditing OCI resources and network security policies.

Redis Client - Asynchronous worker integration used to consume and execute queued scanning tasks.

3.3 Rule Integration with NCA Controls

Every rule in the engine is authored against a control rather than against a vendor feature. The NCA Cloud Cybersecurity Controls document is loaded first, as a structured baseline: 55 top-level controls and 120 subcontrols, 175 in total, each retaining its official identifier, its control text, and its applicability tier.

A rule is then declared with the control identifier it satisfies as a required field. When the scanner evaluates a resource it emits the rule identifier, the severity, and the supporting evidence; the dynamic mapping layer shown on the left of Figure 1 resolves that identifier against the loaded baseline and attaches the full control text to the finding before it is persisted.

LAYER	RESPONSIBILITY	EXAMPLE
Control baseline	The NCA CCC-2:2024 document loaded as structured data 55 top-level controls and 120 subcontrols, each keeping its official identifier and applicability tier.	2-2-T-1-1 — identity and access management for cloud credentials across their full lifecycle
Rule definition	A provider-specific check that names, as a required field, the control identifier it produces evidence for.	OCI-NET-002 — public SSH access allowed on a security list
Mapping layer	Resolves each rule result to its control or controls at scan time and filters the applicable set by the workspace classification level.	17 CSP-side and 6 CST-side subcontrols currently active
Enriched finding	The record persisted and rendered: evidence, severity, control text, and remediation steps in one object.	The findings panel shown in Figure 3

Table 2 - How a provider check becomes compliance evidence.

Three consequences follow. Findings reach the dashboard already carrying their compliance justification, which is why the side panel in Figure 3 can show control 2-2-T-1-1 beside a public bucket without a second lookup. Coverage becomes measurable rather than asserted: the platform currently exercises 23 subcontrols, 17 on the CSP side and 6 on the CST side, a figure derived from the mapping table itself. And adding a provider does not mean re-authoring the compliance logic — a new rule need only name the control it evidences.

Where a classification level is set on the workspace, the mapping layer additionally narrows the applicable control set, so a tenancy marked Public is not audited against controls the framework reserves for Confidential, Secret, or Top Secret workloads.

4 Results

We defined our results by using the NCA Cloud Cybersecurity Controls (CCC) framework as the reference against which findings are judged. Rather than reporting a surface inventory of assets, the engine descends into the configuration payload of every resource it discovers: for each of the sixteen scannable resource types it parses the provider's response key by key, resolving nested policy documents, security-list rule sets, encryption settings, and IAM statements before any judgement is made. That depth is what allows a single pass to flag more than sixty distinct configuration properties across a modest tenancy, and to separate a bucket that is merely public from one that is public and unprotected by a customer-managed encryption key.

Because the inspection is configuration-level rather than perimeter-level, the same pass yields both the finding and the evidence for it which is what makes each result directly actionable rather than advisory. Results are presented below in four parts: a summary of the scan and the assessment surface behind it, the raw rule evaluations produced by the scanner, the dashboard view assembled from them, and the aggregate assessment statistics.

4.1 Scan Summary

To validate the platform’s detection accuracy, a dedicated reference cloud environment was configured with intentional security misconfigurations. A validation scan was performed against the target infrastructure on 18 August 2026 and completed successfully without errors, detecting all expected risk vectors. These included the default security list of **vcn-20260710-1853**, which contained open ingress rules; the **test-compliance-bucket** object storage resource, which had public access and lacked encryption; and the **tenant administrator policy**, which granted overly permissive IAM privileges.

4.2 Rule Evaluations

Each discovered resource is evaluated against the rule set, and every rule returns a severity together with a description and, where a violation is detected, a recommendation. The table below consolidates the evaluations produced for the scanned workspace.

RULE	SEVERITY	AFFECTED RESOURCE	OBSERVATION
Public SSH Access Allowed (OCI-NET-002)	HIGH	Default Security List for vcn-20260710-1853	The security list permits inbound SSH traffic (TCP 22) from all source IP addresses (0.0.0.0/0). Recommendation: restrict the source CIDR blocks to trusted addresses.
Public RDP Access Allowed (OCI-NET-003)	SAFE	Default Security List for vcn-20260710-1853	The rule was evaluated against inbound RDP traffic (TCP 3389) and returned a SAFE result; no remediation is required.
Public Object Storage Bucket	HIGH	test-compliance-bucket (Object Storage, us-east-1)	The bucket allows public access, exposing its objects to read or write operations from the internet. Mapped to NCA CCC-2:2024 control 2-2-T-1-1.

RULE	SEVERITY	AFFECTED RESOURCE	OBSERVATION
Bucket Customer-Managed Key Encryption Disabled	MEDIUM	test-compliance-bucket (Object Storage, us-east-1)	The bucket is not encrypted with a customer-managed key, leaving key lifecycle control with the provider.

Table 3 - Consolidated rule evaluations for the reference OCI workspace scan.

The scanner emits the same information as structured console output, which is the form consumed by the internal backend before it is persisted and rendered:

```
Rule Evaluations (Checks: 2):
  - [HIGH] Rule: Public SSH Access Allowed (OCI-NET-002)
    Description:    Security List allows incoming SSH traffic (port 22) from all source IPs (0.0.0.0/0).
    Recommendation: Restrict source CIDR blocks to trusted IP addresses.
  - [SAFE] Rule: Public RDP Access Allowed (OCI-NET-003)
    Description:    Security List allows incoming RDP traffic (port 3389) from all source IPs (0.0.0.0/0).
```

Figure 2 - Raw rule-evaluation output emitted by the scanner.

4.3 Results in Dashboard View

The dashboard reassembles the same findings into a resource graph. Affected assets are outlined in red and connected to the exposure path that reaches them, while the side panel expands the selected resource into its individual findings, each carrying a severity badge, the mapped NCA CCC-2:2024 control, an explanation of the risk, and the numbered action steps required to remediate it.

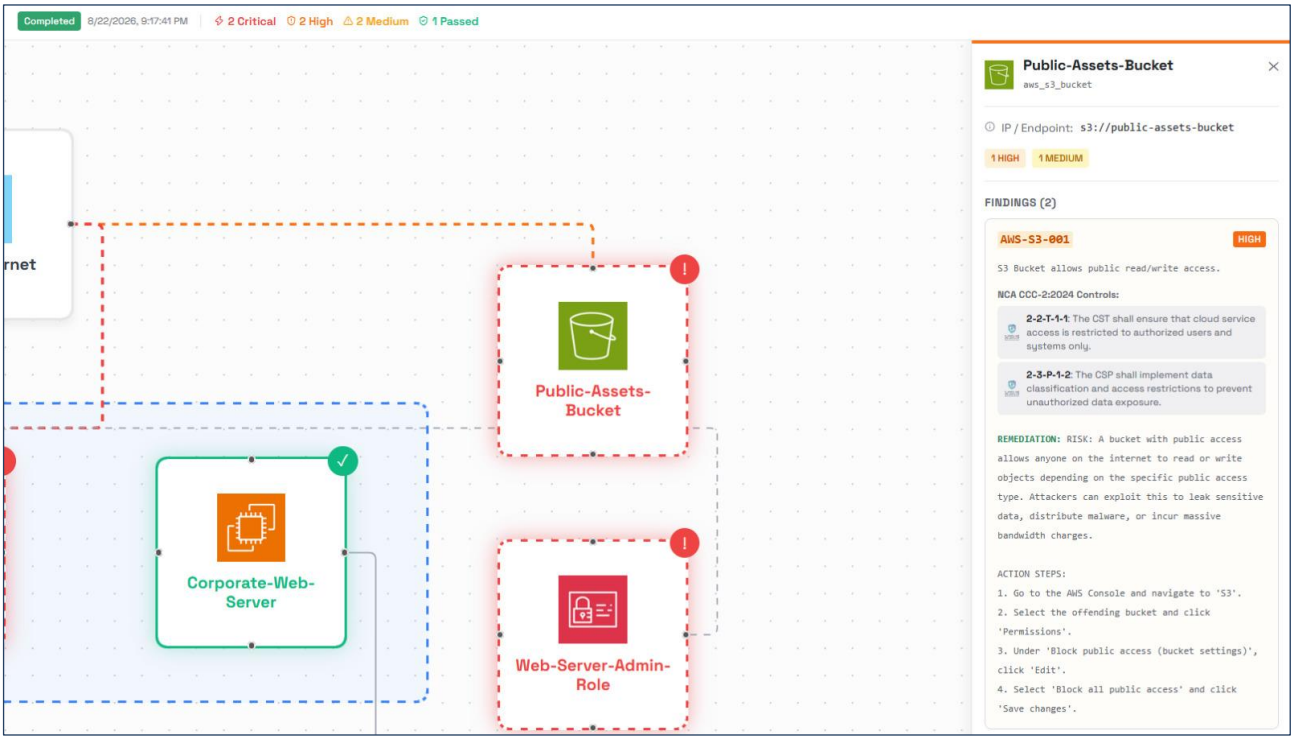


Figure 3 - Dashboard resource graph with the findings panel for test-compliance-bucket.

4.4 Assessment Statistics

Aggregated across the scans documented above, the figures below describe the assessment surface the platform exercises in practice rather than the surface it could theoretically reach.

The baseline scope bounds what can be audited, the active subcontrol count bounds what is audited currently, and the gap between them is the roadmap set out in Section 6.

5 Limitations and Future Work

In our current progress it appears that we have overcome the limitations identified so far, few key areas needs to be addressed for better result:

1. **Advanced Simulation:** We need a more advanced testing environment to stress-test the platform under high-traffic user loads.
2. **Environment Scope & Hybrid Architecture:** This concept should evaluate the platform's effectiveness across on-premises and hybrid-cloud environments.

Future Plans:

1. **Infrastructure as Code (IaC) Suite:** Comprehensive scanner, builder, and architecture visualizer.
2. **Scanning Frequency:** Set up periodic scans (daily, weekly, monthly) that can also be triggered by infrastructure changes or integrated into CI/CD pipelines upon commits (e.g., GitHub Actions).
3. **Advanced Alerting System:** Automated notification system alerting the cloud engineer — or any other responsible party — via sms , email or call after detecting a high or critical risk by the platform.
4. **Agentic Operation:** An agentic AI engineer capable of scanning, notifying, migrating response, work as a expert cloud engineer, setup your environment securely with best practice using trusted security standards.
5. **Compliance & Multi-Cloud Support:** Extended rule sets for other cloud providers and more options of compliance standards for wider coverage.

6 Conclusion

Over this reporting period we moved Sahaba from an open-ended question about what a cloud risk analyzer ought to be into a working, end-to-end system. The architecture we settled on separates concerns cleanly: the dashboard collects credentials and displays results, the Web API authenticates users and encrypts secrets before they are stored, the Redis Queue decouples request handling from long-running work, and the scanner performs the audit in isolation before returning its findings to the internal backend for persistence.

Running that pipeline against a live Oracle Cloud Infrastructure tenancy produced real and actionable output rather than a synthetic demonstration. Publicly reachable SSH on a default security list, a publicly accessible object storage bucket, and a bucket left without customer-managed key encryption were each detected, scored, mapped to an NCA CCC-2:2024 control, and paired with concrete remediation steps — which is precisely the behaviour the problem statement called for.

The work also clarified what remains. Coverage is still bounded by the granularity of the provider APIs, so application-level and operating-system weaknesses fall outside the scanner's reach, and the platform has not yet been exercised at high user concurrency. Our next steps are to broaden the rule set, add further compliance frameworks and cloud providers, and evaluate the use of AI as a secondary reviewer of the overall security posture. Taken together, the results so far support the premise we began with: that a systematic, standards-anchored analyzer can surface misconfigurations that would otherwise go unnoticed for prolonged periods.

7 References

[1] B. Krebs, "What We Can Learn from the Capital One Hack," Krebs on Security, Aug. 2, 2019. https://krebsonsecurity.com/2019/08/what-we-can-learn-from-the-capital-one-hack/

[2] M. Parmar, "Claude Code source code accidentally leaked in NPM package," BleepingComputer, Mar. 31, 2026. https://www.bleepingcomputer.com/news/artificial-intelligence/claude-code-source-code-accidentally-leaked-in-npm-package/

[3] National Cybersecurity Authority (NCA), *Cloud Cybersecurity Controls (CCC)*, Kingdom of Saudi Arabia. https://nca.gov.sa

[4] Amazon Web Services, *Boto3 Documentation — AWS SDK for Python*. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[5] Google Cloud, *Python Client Libraries Reference*. https://cloud.google.com/python/docs/reference

[6] Oracle, *OCI SDK for Python Documentation*. https://docs.oracle.com/en-us/iaas/tools/python/latest/

[7] Docker Inc., *Docker Documentation*. https://docs.docker.com/

[8] Redis Ltd., *Redis Documentation*. https://redis.io/docs/latest/

[9] PostgreSQL Global Development Group, *PostgreSQL Documentation*. https://www.postgresql.org/docs/

[10] OpenJS Foundation, *Express.js Documentation*. https://expressjs.com/

[11] S. Ramírez, *FastAPI Documentation*. https://fastapi.tiangolo.com/