Security Vulnerabilities in Infrastructure as Code

A Critical Analysis of Detection and Prevention Mechanisms in Terraform and AWS CDK

Juber Nunes | C00315740 | MSc Cybersecurity, Privacy and Trust | South East Technological University (SETU) | 2026

Abstract

This study empirically compares security vulnerabilities in 6 Terraform and 6 AWS CDK production repositories using 4 static analysis tools and 8 practitioner interviews across 205 validated findings.

Terraform and CDK share the same vulnerability categories. The key difference is visibility: Terraform findings are explicit in source; CDK findings often emerge only through synthesis, defaults, or generated artefacts.

Literature Review & Research Gap

Key themes from the literature:

- IAM misconfiguration, network exposure & data protection gaps are the dominant IaC vulnerability categories across academic and industry sources
- Static analysis tools achieve only 60–70% detection effectiveness — significant blind spots remain across all major scanners
- CDK abstraction opacity is industry-observed but not yet systematically measured
- No prior study empirically compares vulnerability patterns across declarative vs. imperative IaC paradigms under controlled conditions

Research Gap

To the author's knowledge, no existing study systematically measures whether CDK abstraction layers produce observable, quantifiable differences

Research Questions

RQ1	What vulnerability patterns emerge across Terraform and CDK repositories?
RQ2	How effective are static IaC tools at detecting high-severity vulnerabilities?
RQ3	How do CDK abstraction layers affect security visibility vs. Terraform?
RQ4	What evidence-based recommendations emerge from the findings?

Methodology

Mixed-methods: quantitative repository analysis + qualitative practitioner triangulation

12 Repositories

- 6 Terraform (T1–T6)
- 6 AWS CDK (C1–C6)
- Production-only — no demos

4 Static Tools

- Checkov v2
- TFSec / KICS v2.1.20
- Trivy

Manual Validation

- 100% CRITICAL & HIGH
- 15–20% MEDIUM sampled
- TP / FP / RC classification

8 Interviews

- 7–20+ yrs experience
- Terraform + CDK practitioners
- Data-driven triangulation

CDK Abstraction Analysis

- cdk synth on all 6 CDK repos
- Source vs. synthesised comparison
- 5-type auditability typology

Repo Selection Criteria

- Remote state management
- Multi-env + security configs
- Modular architecture + CI/CD

Repository selection used production indicators — not star counts or popularity metrics: remote state, multi-environment patterns, security configurations (VPC, IAM, encryption), modular architecture, CI/CD integration, and <6 months commit activity.

Key Findings at a Glance

205	34	96	4×
Total Validated Findings	CRITICAL Findings	HIGH Findings	CDK synth uplift C2: 5→20 findings

RQ1 — Vulnerability Patterns

Same vulnerability categories in both paradigms. Difference is visibility — not type.

4 Universal Terraform Patterns — present regardless of architecture:

- Unrestricted SG egress 0.0.0.0/0 — all 5 repos with SGs
- HTTP-only ALB listeners — all 4 repos with ALB
- S3 access logging disabled — all 5 repos with S3
- VPC Flow Logs disabled — 4 of 5 repos with VPC

Architecture-specific patterns:

- T4 (EKS): Kubernetes + external module risks
- T5 (Serverless): API GW auth, ECR mutability, Lambda root user
- T6 vs T2: Monolithic vs modular — same finding profile (structure ≠ security)

CDK findings similar in category — more often surfaced through synthesised output than TypeScript source.

RQ2 — Tool Detection Effectiveness

No single tool = complete coverage. Multi-tool scanning is the minimum.

T3 — Checkov: zero findings
TFSec + KICS found CRITICAL/HIGH — strongest false-negative case

T2 — KICS-only secret detection
Hardcoded DB password missed by 3 other tools

T5 — Severity disagreement
Same finding: KICS=CRITICAL, others=HIGH

TFSec: zero CDK findings
Zero coverage across all 6 CDK repos — structural blind spot

C2 — 4× synthesis uplift
5 source findings → 20 synthesised CF findings (Checkov)

C4 — Trivy-only CRITICAL
Lambda permission finding missed by all other tools

Tool	Terraform	CDK Source	CDK Synth CF
Checkov	Yes	Partial	Yes
TFSec	Yes	None	None
KICS	Yes	Partial	Yes
Trivy	Yes	Partial	Yes

RQ3 — CDK Abstraction & Visibility ★ Primary Contribution

CDK abstraction relocates security risk into defaults and generated artefacts — less visible to developers, less consistently detected by tools.

Terraform baseline: All 6 repos statically auditable from HCL source — zero external dependencies required.

CDK Synthesis Typology — Auditability Spectrum

Type 1 C1	Clean synthesis Full source vs. generated comparison possible
Type 1 variant C4, C5	Generated templates committed to repo Insecure defaults remain — reduced friction
Type 2 C2	Environment variable dependency Independent auditing constrained
Type 2.5 C6	Partial synthesis — multi-stack Primary application stack unassessable
Type 3 C3	Synthesis failure — CDK internal bug Primary IaC layer entirely non-auditable

C2 Synthesis Evidence — Primary Quantitative Finding

Checkov on TypeScript source: 5 findings

Checkov on synthesised CloudFormation: 20 findings (4×)

15 findings invisible to developers in authored code

Practitioner Interviews — Triangulation

8 participants · 7–20+ yrs experience · Unanimous themes:

- CDK abstraction is a trust gap — defaults trusted without inspection
- Synthesised CloudFormation never reviewed routinely
- False positive fatigue causes teams to bypass scanning
- Multi-tool scanning is the practical minimum
- Private repos = more complacency — access control ≠ security
- Culture and workflow matter more than tool selection

RQ4 — Conclusions & Recommendations

Thesis Position

Terraform and CDK share the same vulnerability categories. The key difference is visibility and auditability — not intrinsic paradigm security. CDK abstraction layers relocate risk into generated artefacts and hidden defaults that reduce security visibility and weaken tool coverage.

7 Evidence-Based Recommendations:

1. Use multiple tools — no single scanner is complete
2. CDK synthesis is a mandatory security gate, not optional
3. Scan CDK source AND synthesised CloudFormation
4. Design CDK stacks to synthesise without live AWS credentials
5. Treat committed cdk.out/ as a security-relevant artefact
6. Manual validate CRITICAL/HIGH — tool output is not ground truth
7. Treat modules & constructs as supply-chain artefacts

Research Timeline

Repos + Tools	CDK Synthesis	Interviews	Writing	Submission
Feb–Mar	Mar–Apr	Mar–Apr	Apr–May	11 May