



IJEAST

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY



VOLUME : 11 ISSUE : 05 Print / Issue Publication Date: September 2026



ISSN : 2455-2143



Indexed In



WWW.IJEAST.COM

editor@ijeast.com



STATIC ANALYSIS AND LLM-BASED REVIEW FOR TERRAFORM SECURITY AND COST ISSUES: AN EXPLORATORY COMPARATIVE STUDY

Keerthana K S
Student, MCA

School of Science and Computer Studies
CMR University, Bangalore, India

Abstract— Infrastructure-as-Code (IaC) tools such as Terraform enable declarative provisioning of cloud infrastructure but can also encode security misconfigurations and cost-inefficient settings. This exploratory study compares a rule-based static analyzer, Checkov, with a custom large language model (LLM)-based Terraform reviewer. The primary evaluation uses 12 Terraform files containing secure controls, security misconfigurations, cost-related issues, and a mixed security-and-cost case. The two tools are applied to the same source files and evaluated against a predefined ground-truth issue list. Checkov provides deterministic, fast detection of known security and compliance patterns, whereas the LLM reviewer produces consolidated explanations, suggested fixes, and contextual cost observations. A supplementary three-run evaluation on a stratified six-file subset is used to examine limited run-to-run consistency of the LLM reviewer. The results are interpreted as exploratory and dataset-specific rather than as general performance estimates. The findings indicate complementary strengths: static analysis provides consistent coverage of predefined security rules, while the LLM reviewer identifies contextual issues that are outside the evaluated scanner's rule scope and presents findings in more accessible language. These observations motivate a hybrid pipeline that combines static analysis, LLM-based contextual review, and scanner-based verification of proposed fixes.

Keywords—Infrastructure as Code, Terraform, static analysis, large language models, cloud security, DevSecOps, cost analysis, hybrid review.

I. INTRODUCTION

Cloud infrastructure is increasingly provisioned through Infrastructure as Code (IaC), in which declarative configuration files describe computing resources, storage, networking, access permissions, and deployment policies. Terraform, developed by HashiCorp, is a widely used multi-

cloud IaC language that supports a common provisioning workflow across platforms such as Amazon Web Services, Microsoft Azure, and Google Cloud.

IaC improves repeatability and deployment consistency, but it can also propagate insecure configurations at scale. A publicly accessible storage bucket, an unrestricted network rule, an over-permissioned identity policy, or an exposed secret may create a security weakness when the configuration is deployed. IaC review before deployment is therefore an important control for identifying security and compliance problems before they reach production. Automated IaC scanners parse configuration files and compare declared resources with predefined security and compliance policies [1], [2].

Two broad approaches are increasingly used for automated IaC review. The first is rule-based static analysis, represented in this study by Checkov. Static-analysis tools apply deterministic rules to identify known configuration patterns. The second approach uses large language models (LLMs) to review IaC configurations contextually, explain risks in natural language, and suggest possible remediation steps [12].

This paper presents an exploratory comparison of Checkov and a custom LLM-based Terraform reviewer using the same 12 Terraform source files. A supplementary analysis applies the LLM reviewer three times to a stratified six-file subset to examine limited run-to-run consistency. The study does not claim universal superiority for either approach; instead, it investigates issue coverage, false-positive behavior, explanation quality, actionability, and contextual scope.

1.1 Research Questions

RQ1: How do Checkov and the LLM-based reviewer compare in their observed detection of predefined security and cost-related issues in the same Terraform files?

RQ2: How do the two approaches differ in the explainability and actionability of their reported findings?

RQ3: What potential benefits and verification requirements arise from combining rule-based static analysis with LLM-based contextual review in a hybrid pipeline?

II. RELATED WORK

Research on IaC security has expanded alongside cloud automation and the adoption of LLMs in software engineering [8]. The literature relevant to this study can be organized into static analysis for IaC, LLM-based IaC generation, LLM-based review and repair, and verification of AI-assisted infrastructure changes.

Rule-based scanners such as Checkov evaluate Terraform configurations against predefined security and compliance policies. An empirical study examined security-practice adoption in public IaC repositories using Checkov and reported variation across security-practice categories [2]. Static analysis is effective for known patterns such as public storage, unrestricted network access, missing encryption, excessive permissions, and exposed secrets, but its coverage depends on the enabled rule set and may not capture workload context or cost implications [10].

A survey of LLM-based IaC generation examined cloud platforms, IaC challenges, and the use of LLMs for code generation [3]. Syntactic validity does not guarantee security or compliance. A security-focused Text-to-Terraform evaluation further supports separating syntax, functional validity, and security compliance [4].

LLMs have also been investigated for Terraform code-smell detection [5] and tool-grounded IaC repair [6]. These approaches can consolidate findings, explain issues, and suggest repairs, but proposed changes may remain incomplete or introduce new problems [9].

TerraProbe proposed layered verification for LLM-assisted Terraform repairs, including targeted scanning, full scanning, validation, planning, and plan comparison [1]. This work shows that clearing one scanner warning does not necessarily prove that the underlying security requirement has been satisfied.

The present study addresses a narrower practitioner-oriented question: how does a rule-based scanner compare with an LLM-based reviewer when both are applied to the same Terraform configurations containing security, cost, and mixed issues? It evaluates issue coverage, false-positive behavior, explanation quality, actionability, and limited run-to-run consistency.

III. METHODOLOGY

3.1 Dataset and Ground Truth

The final dataset contains twelve Terraform files representing secure controls, security misconfigurations, cost inefficiency, and a mixed security-and-cost configuration. The primary experiment uses all twelve files. The original four-file study was treated as a pilot and is not reported as a separate primary experiment. Each file has a predefined ground-truth issue list created before tool comparison. A finding is a true positive when it maps to an injected issue; multiple scanner checks referring to one underlying problem are consolidated into one core issue.

TABLE I. DATASET COMPOSITION

Category	Files	Purpose
Security controls	2	False-positive assessment
Security-misconfiguration files	6	Access, network, secrets, storage, serverless, and database risks
Cost-focused files	2	Cost-efficiency analysis
Mixed security and cost	1	Co-occurrence test
Cost-dimension control	1	Cost-only control
Total	12	Primary dataset

3.2 Tools and Procedure

Checkov was executed against each Terraform file using its configured Terraform security and compliance rules. Passed checks, failed checks, check identifiers, severities where available, and runtime were recorded. The LLM reviewer was implemented using a Flask dashboard and Google Gemini API. A fixed prompt requested security, cost, and best-practice findings in a structured severity, issue, rationale, and fix format. The primary LLM results contain one recorded review per file. A supplementary repeatability analysis reviewed a stratified six-file subset three times using the same model, prompt, and generation settings.

TABLE II. EVALUATION DESIGN

Analysis	Files	LLM runs per file	Purpose
Primary comparison	12	1	Compare Checkov and LLM findings
Repeatability subset	6	3	Measure exploratory run-to-run consistency

3.3 Evaluation Criteria

TABLE III. EVALUATION CRITERIA

Criterion	Definition
Recall	$TP/(TP+FN)$ against predefined issues
Precision	$TP/(TP+FP)$ under the annotation protocol
False positives	Findings on verified-clean security controls
Explainability	Quality of issue rationale
Actionability	Usefulness of proposed remediation
Runtime	Wall-clock analysis time
Repeatability	Agreement across repeated LLM runs

IV. RESULTS

The primary analysis evaluated both tools on the same twelve Terraform source files. Findings were compared with the predefined issue list. The primary LLM results represent one



recorded review per file; the supplementary three-run analysis is reported separately.

4.1 Checkov Results

Checkov detected 16 of 20 predefined security issues, giving an observed security recall of 80.0%. It detected 1 of 9 predefined cost issues, giving an observed cost recall of 11.1%. Across the 31 issues included in the category-level comparison, Checkov detected 18, corresponding to an observed recall of 58.1%. Four findings occurred on the two verified-clean security controls.

TABLE IV. CHECKOV DETECTION PERFORMANCE

Metric	Security issues	Cost issues	Overall
Number of issues	20	9	31
True positives	16	1	18
False negatives	4	8	13
Observed recall	80.0%	11.1%	58.1%
False positives on clean controls	4	N/A	4

4.2 LLM Reviewer Results

The LLM reviewer detected 18 of 20 predefined security issues and 8 of 9 predefined cost issues. The observed recalls were 90.0% and 88.9%, respectively. Including the two mixed-file issues, the reviewer detected 28 of 31 issues, corresponding to an overall observed recall of 90.3%. Five findings occurred on the two verified-clean security controls.

TABLE V. LLM REVIEWER PERFORMANCE

Metric	Security issues	Cost issues	Overall
Number of issues	20	9	31
True positives	18	8	28
False negatives	2	1	3
Observed recall	90.0%	88.9%	90.3%
False positives on clean controls	5	N/A	5

4.3 Qualitative Comparison

TABLE VI. QUALITATIVE COMPARISON

Criterion	Checkov	LLM reviewer
Detection behavior	Deterministic rule matching	Contextual interpretation
Explainability	Rule ID and fixed description	Natural-language rationale
Actionability	Limited repair guidance	Issue-specific suggested fixes
Security coverage	Strong for enabled rules	Variable by prompt and model
Cost analysis	Limited in selected configuration	Contextual cost observations
Runtime	Local and generally fast	Higher due to API processing

The LLM reviewer generally consolidated related findings into fewer observations and provided more accessible explanations. Checkov produced deterministic policy-linked findings. These outputs are complementary, but LLM suggestions require independent verification.

4.4 Three-Run Consistency Analysis

A stratified subset of six files was reviewed three times by the LLM reviewer. The subset included security, cost, mixed, and verified-clean control cases. Across the five issue-bearing files, the reviewer detected 17 of 20 issues in Run 1, 18 of 20 in Run 2, and 17 of 20 in Run 3. The corresponding observed recalls were 85.0%, 90.0%, and 85.0%.

TABLE VII. THREE-RUN LLM CONSISTENCY ANALYSIS

File	Ground truth	Run 1	Run 2	Run 3	Mean	Range
bad_s3_bucket.tf	4	4	4	4	4.00	4–4
bad_ec2_security_group.tf	5	3	3	3	3.00	3–3
bad_cost_inefficient.tf	5	4	5	5	4.67	4–5
bad_cost_inefficient_2.tf	4	4	4	3	3.67	3–4
mixed_security_and_cost.tf	2	2	2	2	2.00	2–2
good_s3_bucket.tf control	0	3	3	3	3.00	3–3

The security and mixed cases were stable in detection count, while cost-focused files varied because some findings required workload-context judgments. Findings on the clean control were treated as false positives and excluded from recall calculations. The six-file analysis is supplementary and does not establish general LLM reliability.

V. DISCUSSION

5.1 Static Analysis Strengths

Checkov provides deterministic output for a fixed version, configuration, and input. Its findings are linked to explicit policies and check identifiers, making them suitable for auditing and CI/CD policy gates. Its limitations arise from rule scope: it can detect conditions represented by enabled rules but may not capture workload context or cost implications.



5.2 LLM Reviewer Strengths

The LLM reviewer provides natural-language explanations, consolidates related findings, and suggests remediation steps. It also identifies contextual cost concerns outside the evaluated scanner rules. However, LLM outputs may contain false positives, false negatives, inconsistent issue descriptions, or incomplete repairs [11].

5.3 Proposed Hybrid Pipeline

The results motivate a three-stage pipeline: 1) static analysis for known security and compliance violations; 2) LLM review for contextual explanation and remediation suggestions; and 3) independent rescanning and validation of proposed changes. Static analysis is treated as an independent verification layer rather than absolute ground truth.

5.3.1 Prototype Verification

A prototype was demonstrated on `bad_s3_bucket.tf`. The initial scan identified 12 failed checks. An LLM-generated modification restored public-access blocking, restricted the bucket policy, enabled encryption, and enabled versioning. A second scan showed that the core injected issues were resolved while secondary best-practice findings remained. Because the workflow was demonstrated on one file, this is a proof of concept rather than a full hybrid-system validation.

5.4 Threats to Validity

Construct validity: the files, injected issues, and prompt were authored by one researcher, and cost judgments may depend on workload assumptions. Internal validity: one annotator mapped findings to the ground truth, and the tools have different operating characteristics. External validity: the dataset is small, manually constructed, AWS-specific, and Terraform-specific. Reliability: the primary LLM results are single-run observations, while the repeated analysis covers six files; the observed five-point range is descriptive for that subset, not an error margin or confidence interval for the full dataset. Conclusion validity: the sample is too small for strong inferential claims, and no universal superiority is claimed.

VI. CONCLUSION AND FUTURE WORK

This paper compared Checkov with a custom LLM-based reviewer on a 12-file Terraform dataset containing security misconfigurations, cost-related issues, and a mixed configuration. Checkov achieved observed recalls of 80.0% for security issues and 11.1% for cost issues under the selected rules. The LLM reviewer achieved observed recalls of 90.0% for security issues and 88.9% for cost issues in the primary single-run evaluation. These are dataset-specific descriptive results, not general performance estimates. Checkov provided deterministic policy-linked findings, while the LLM reviewer provided consolidated explanations, suggested fixes, and contextual cost observations. The supplementary repeated-run analysis showed greater variation for judgment-dependent cost findings than for binary security settings.

A three-stage hybrid pipeline was demonstrated on one representative Terraform file using static analysis, LLM-based contextual review, and scanner-based verification. The demonstration showed how resolved and unresolved findings can be distinguished, but the complete pipeline was not evaluated across all twelve files. Future work should use larger externally sourced Terraform repositories, multiple cloud providers and IaC languages, multiple LLMs and prompt strategies, repeated evaluation of the complete dataset, independent annotators, inter-rater agreement, and verified-repair metrics such as residual vulnerability rate, newly introduced issue rate, review time, token usage, and developer effort.

VII. REFERENCES

- [1]. Alsaid, Manar; Nebolisa, Chimdumebi; Abbas, Faris, 2026, TerraProbe: A Layered-Oracle Framework for Detecting Deceptive Fixes in LLM-Assisted Terraform, DOI: 10.48550/arXiv.2606.26590.
- [2]. Verdet, Alexandre; Hamdaqa, Mohammad; Da Silva, Léuson M. P.; Khomh, Foutse, 2023, Exploring Security Practices in Infrastructure as Code: An Empirical Study, DOI: 10.48550/arXiv.2308.03952.
- [3]. Srivatsa, Kalahasti Ganesh; Mukhopadhyay, Sabyasachi; Katrapati, Ganesh; Shrivastava, Manish, 2024, A Survey of Using Large Language Models for Generating Infrastructure as Code, DOI: 10.48550/arXiv.2404.00227.
- [4]. Santos Vargas, Francis Luis; Brandão Mansilha, Rodrigo; Kreutz, Diego, 2026, Security-First Evaluation of Text-to-Terraform: Benchmarking LLMs and SLMs for Secure IaC Generation, DOI: 10.48550/arXiv.2608.02672.
- [5]. Vo, Q.-H.; Dao, H.; Fukuda, K., 2025, Harnessing the Power of LLMs for Code Smell Detection in Terraform Infrastructure as Code, pp. 533–542, DOI: 10.1109/COMPSAC65507.2025.00075.
- [6]. Mengistu, Minase Mekete; Di Rocco, Juri; Nguyen, Phuong T.; Di Ruscio, Davide, 2026, TerraRepair: A Tool-Grounded LLM Agent for Infrastructure-as-CodeRepair, DOI: 10.48550/arXiv.2607.11390.
- [7]. Ramamoorthy, U.; Loganathan, A., 2022, Analysis of Video Steganography in Military Applications on Cloud, pp. 897–903, The International Arab Journal of Information Technology, Vol. 19, No. 6.
- [8]. Hou, Xinyi; Zhao, Yanjie; Liu, Yue; Yang, Zhou; Wang, Kailong; Li, Li; Luo, Xiapu; Lo, David; Grundy, John; Wang, Haoyu, 2024, Large Language Models for Software Engineering: A Systematic Literature Review, DOI: 10.1145/3695988.
- [9]. Zhou, Xin; Cao, Sicong; Sun, Xiaobing; Lo, David, 2025, Large Language Model for Vulnerability Detection and Repair: Literature Review and the Road Ahead, DOI: 10.1145/3708522.



- [10]. Ntentos, Evangelos; Lueger, Nicole Elisabeth; Simhandl, Georg; Zdun, Uwe; Schneider, Simon; Scandariato, Riccardo; Díaz Ferreyra, Nicolás E., 2024, On the Understandability of Design-Level Security Practices in Infrastructure-as-Code Scripts and Deployment Architectures, DOI: 10.1145/3691630.
- [11]. Belozarov, Vladislav; Barclay, Peter J.; Sami, Ashkan, 2026, Secure Coding with AI – From Detection to Repair, DOI: 10.1007/s10664-026-10812-8.
- [12]. Sheng, Ze; Chen, Zhicheng; Gu, Shuning; Huang, Heqing; Gu, Guofei; Huang, Jeff, 2025, LLMs in Software Security: A Survey of Vulnerability Detection Techniques and Insights, DOI: 10.1145/3769082.

IJEAST

INTERNATIONAL JOURNAL OF ENGINEERING APPLIED SCIENCE AND TECHNOLOGY



AUTHORS

Keerthana K S



ABOUT IJEAST

International journal of Engineering Applied Science and Technology (IJEAST) is a peer-reviewed, open access journal that publishes high - quality research paper in the field of Engineering, Applied Science and Technology.

IJEAST aims to provide a platform for researchers, academicians, and professionals to share their innovative ideas, research findings, and practical experiences with the global scientific community.

JOURNAL METRICS



H-INDEX

33



i10-INDEX

154



IJEAST
CITATIONS

9,647

(IJEAST CITATIONS)



PEER-REVIEWED
& OPEN ACCESS

PUBLISHED
MONTHLY



PEER REVIEWED

All submission are rigorously peer reviewed to ensure quality.



AUTHORS ARE OUR PRIORITY

We value our authors and recognize their contribution to the advancement of research and knowledge.



OPEN ACCESS

Free and unrestricted access to research for all.



GLOBAL REACH

Connecting researchers and professionals worldwide.



TIMELY PUBLICATION

We ensure a swift and efficient publication process.



For more information, visit our website
www.ijeast.com



editor@ijeast.com



www.ijeast.com



India



2455-2143