International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V74I6P101

Volume 74 Issue 6, 1-11, June 2026

Original Article

Semantic Transformer-Based Detection of Security Misconfigurations in Network Configuration Infrastructure-as-Code

Lakshmi Harika Akkireddy[1], Naga Satya Praveen Kumar Yadati[2], Vijayakumar Venganti[3]

[1,2,3]*Independent Researcher, USA.*

[2]*Corresponding Author : praveenyadati@ieee.org*

Received: 18 April 2026 Revised: 22 May 2026 Accepted: 10 June 2026 Published: 29 June 2026

Abstract *- Network configuration infrastructure misconfigurations have emerged as one of the leading causes of security breaches, yet existing rule-based and static detection methods remain rigid and fail to generalise across the diverse patterns present in Infrastructure-as-Code (IaC) artefacts. This research gap motivates the need for a semantics-aware, data-driven detection approach that goes beyond syntactic pattern matching. This study proposes a lightweight deep-learning framework for the automated detection of security misconfigurations in Terraform-based IaC, leveraging the transformer-based semantic understanding of code offered by DistilBERT. A balanced dataset of 2,000 labelled Terraform samples (1,000 secure, 1,000 insecure) is constructed, and the model is trained under practical computational constraints while satisfying the requirements of reproducible experimentation. Evaluation on a held-out test set yields an F1-score of 0.7598, a recall of 0.8700, ROC-AUC of 0.8224, and PR-AUC of 0.8344, outperforming the TF-IDF with Logistic Regression baseline in recall and overall detection capability. The results confirm that contextual transformer representations capture security-relevant patterns beyond syntactic rules, offering a scalable and intelligent solution for proactive misconfiguration assurance in network configuration environments.*

Keywords *- Network Configuration Security Misconfiguration Detection, Infrastructure-as-Code Analysis, Transformer-Based Deep Learning.*

1. Introduction

The proliferation of network configuration computing infrastructure has made Infrastructure-as-Code (IaC) the de facto approach for managing and provisioning network resources programmatically [1, 2]. While IaC practices such as Terraform and Kubernetes configurations have greatly accelerated deployment velocity, they have simultaneously introduced a significant new attack surface: security misconfigurations embedded in configuration scripts. Empirical studies and industry reports consistently identify misconfigured network infrastructure as one of the leading root causes of security breaches, accounting for a substantial proportion of incidents in recent years [1, 2]. Despite this risk, the detection of such misconfigurations remains a largely unsolved problem. Practitioners using tools like Checkov, TFSec, and KICS rely on manually curated rule sets that must be updated continuously as new resource types and configuration patterns emerge [3]. These tools suffer from a fundamental limitation: they match fixed syntactic patterns and cannot reason about the semantic context of a configuration, making them brittle against novel or complex misconfiguration patterns [4]. The research gap is therefore clear: a detection mechanism is required that can learn generalizable, context-sensitive security representations directly from IaC text, without relying on handcrafted rules. Recent advances in transformer-based language models have demonstrated strong capacity for understanding code semantics and contextual dependencies [5], suggesting their potential for bridging this gap.

This study proposes a transformer-based method for the automated, context-aware detection of security misconfigurations in Terraform-based network configuration IaC. The novelty of the work is threefold. First, a lightweight DistilBERT classifier is applied to IaC security analysis, demonstrating that resource-efficient transformers can achieve superior recall compared with lexical baselines without requiring large computational budgets. Second, a structured comparison against a TF-IDF with Logistic Regression baseline under identical experimental conditions isolates the contribution of contextual semantic modelling over frequency-based representations. Third, the study provides a rigorous evaluation protocol using precision, recall, F1-score, ROC-AUC, and PR-AUC, with particular emphasis

on recall and PR-AUC given the asymmetric cost of missed insecure configurations in operational environments. The remainder of this paper is organised as follows: Section II reviews related literature; Section III describes the methodology; Section IV presents results and discussion; and Section V concludes with directions for future work.

2. Literature Review

Early Infrastructure-as-Code research centered on adoption, automation, and tooling, establishing foundational systematic mapping of the IaC ecosystem; however, security-specific investigation remained comparatively underdeveloped in these early works [6]. Subsequent research moved toward defect characterisation and script quality, including smell-oriented analyses of infrastructure artefacts and validation practices for deployment logic [7, 8]. Security concerns became more explicit in later empirical studies that catalogued insecure coding patterns within IaC scripts and proposed technology-independent frameworks for security smell detection across Puppet, Ansible, and Chef artefacts [3]. These contributions established the vocabulary of IaC security issues but remained largely rule-driven, enumerating known bad patterns rather than learning generalised representations of risk.

Comprehensive surveys and empirical analyses subsequently highlighted that the dominant paradigm in IaC security analysis remained static rule evaluation and pattern matching, which offered limited adaptability to diverse network configuration environments and emerging provider-specific resource types [4, 9, 10]. These studies documented the fragility of rule-based tools when confronted with novel configuration structures not represented in their rulesets, underscoring the need for adaptive, learning-based approaches. In parallel, broader work on secure data engineering and AI-driven pipeline automation addressed robustness, validation, and operational trust in modern infrastructure workflows [11-13], providing supporting evidence that machine-learning techniques can be effective in infrastructure security contexts. Concurrently, the natural language processing community developed a progression of semantics-aware code models-from BERT [5] through CodeBERT [14], GraphCodeBERT [15], and CodeT5 [16]-each offering increasingly sophisticated representations of program structure and inter-token dependencies. These models demonstrated the feasibility of learning contextual representations of code that transfer well to downstream classification tasks, motivating their application to IaC security analysis. More recent research has examined Kubernetes-specific misconfiguration analysis, remediation-focused large language models for IaC repair, and large-scale empirical studies of vulnerabilities across IaC repositories [17-20]. Transformer-based code representations have also been applied to source code security classification and vulnerability detection in general software contexts, demonstrating strong performance over lexical baselines [21-24]. Semantic code modelling has recently been extended to network configuration security analysis tasks [25]. However, a targeted, empirically validated comparison of lightweight transformer classifiers against lexical baselines specifically for Terraform misconfiguration detection-with a focus on operational deployability and security-oriented metrics such as recall and PR-AUC-remains absent from the literature. The present study addresses this gap by implementing a DistilBERT-based misconfiguration detector, benchmarking it against a TF-IDF with Logistic Regression baseline under identical experimental conditions, and providing a detailed analysis of the trade-off between recall-oriented semantic modelling and precision-oriented lexical classification.

3. Methodology

This section introduces the methodological framework created for the automated finding of security misconfigurations in Terraform-based Infrastructure-as-Code artifacts. It presents the problem setting, dataset construction, exploratory analysis, input preparation, transformer-based modeling, and comparative baseline design adopted in the study.

3.1. Problem Setting and Experimental Workflow

The study presents Terraform security analysis as a supervised binary classification problem where each configuration sample belongs to a secure class or an insecure class. The learning dataset is defined in (1),

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N} \quad (1)$$

where x_i is the Terraform configuration text of the i-th sample, and y_i is its binary security label, 0 is insecure, and 1 is secure. This formulation is consistent with the chosen dataset, because each sample is linked to a security validity outcome, which is obtained in the metadata. The general workflow starts with the construction of the dataset and preparation of labels, and then moves on to the exploratory analysis of the structural and lexical properties of Terraform scripts. The pipeline then applies text preprocessing and input preparation at the token level for use in transformer-based learning. A lightweight DistilBERT classifier is adopted as the proposed semantic model, and TF-IDF with Logistic Regression is adopted as the baseline model for comparison. The last stage conducts validation-guided training and held out test evaluation for analytical and relative evaluation of lexical and contextual representations.

3.2. Dataset Construction and Exploratory Analysis

The research is based on a publicly available Terraform security dataset in which each sample comprises raw IaC configuration text accompanied by a binary security label derived from metadata annotations. Labels were assigned on the basis of known misconfiguration patterns documented in community security benchmarks and tool-validated rules, with insecure samples corresponding to configurations that violate

at least one established security policy (e.g., unrestricted ingress rules, disabled encryption at rest, or absent logging). This labelling methodology aligns with prior empirical IaC security studies and ensures that class membership reflects operationally meaningful risk rather than synthetic artefacts. In order to guarantee efficient training while eliminating class-imbalance bias, a balanced subset of 2,000 samples is constructed by selecting exactly 1,000 secure and 1,000 insecure configurations. This deliberate balancing step is critical: without it, a model could achieve high accuracy by predicting the majority class, artificially inflating aggregate performance metrics, and masking poor recall on the minority insecure class. The balanced design ensures that both models are evaluated under equivalent conditions and that performance differences reflect genuine representational differences rather than data skew. The composition of classes of the final dataset is shown in Figure 1.

Fig. 1 Class distribution of balanced terraform dataset

In order to analyze the structural complexity of the Terraform samples, the distribution of tokens is analyzed after a logarithmic scaling. The resulting distribution in Figure 2 shows that both classes range over similar token ranges, suggesting that the classification task is not based on trivial length differences. This supports the need for contextual modeling beyond simple size-based cues. The distribution of the number of lines is then further investigated to determine variation in structure between classes. As shown in Figure 3, both secure and insecure samples have a diverse script length, with partially overlapping and non-identical density. This means that insecure patterns can appear in both short and reasonably long Terraform files, so the problem cannot be solved with simple heuristics based on thresholds.

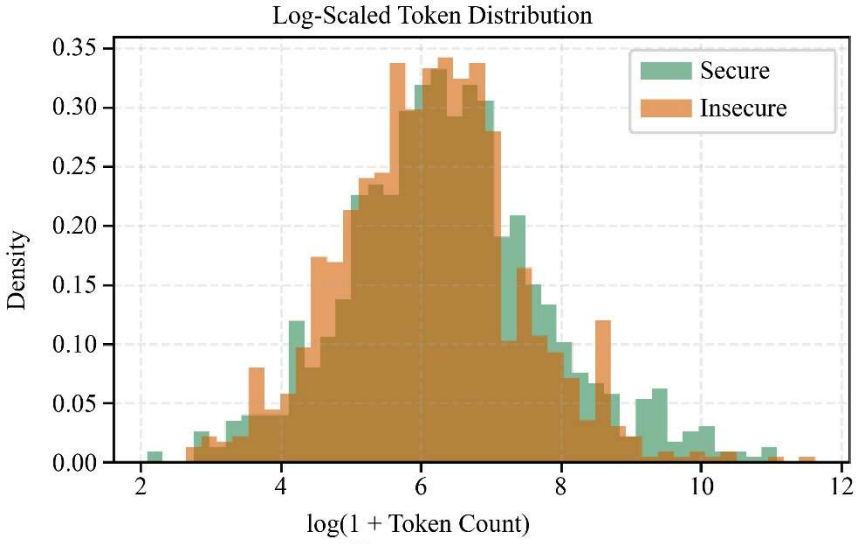

Fig. 2 Log-scaled token count distribution

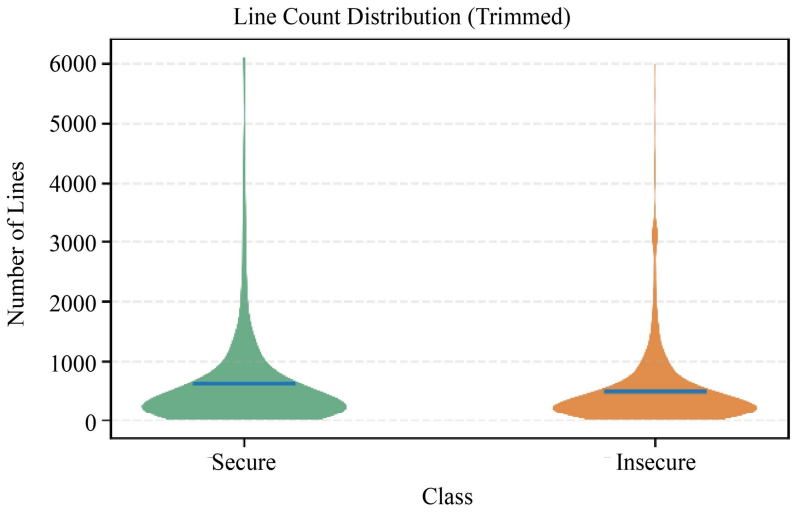

Fig. 3 Distribution of trimmed line count

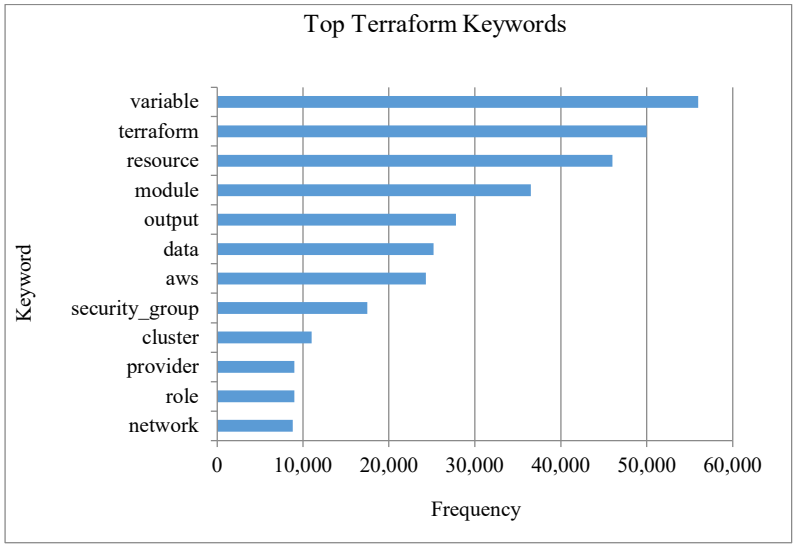

Fig. 4 Most frequent terraform keywords

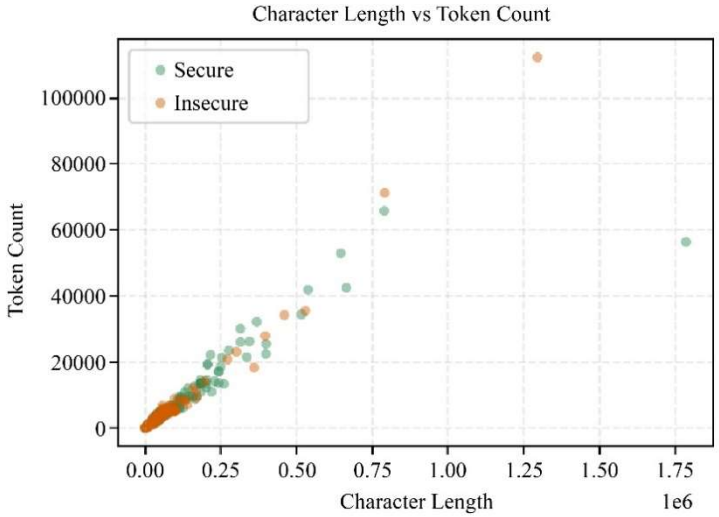

Fig. 5 Relationship between character length and token count

The lexical composition of the dataset is summarized in Figure 4 through the most frequent Terraform keywords. Terms like resource, provider, variable, and AWS are prominent in the corpus, confirming that the corpus reflects actual infrastructure configuration semantics and not generic source code text.This observation is important because the proposed model is aimed at learning security-related contextual patterns embedded in domain-specific network configuration language.

Finally, the relationship between the character length and token number is presented in Figure 5. The near-linear relationship between raw textual size and tokenized representation confirms that there is consistency between the raw and tokenized representation of the texts, and the spread of samples confirms that there is meaningful variability in how scripts are composed. This provides support for the later tokenization strategy for input preparation for the transformer.

Overall, the exploratory analysis reveals that this dataset is balanced, structurally diverse, and semantically rich, which makes it suitable for assessing whether it is possible to capture misconfiguration patterns better with transformer-based representations than with conventional lexical baselines.

3.3. Preprocessing and Input Representation

Before the model training, the Terraform configuration text is cleaned to remove formatting markers introduced by code block wrappers, leaving the actual content of the configuration. This step ensures that the learned representation is driven by infrastructure semantics and not presentation artifacts. After cleaning, each sample is represented as a textual sequence and split into training, validation, and test samples using an 80 to 10 to 10 split.

For transformer input preparation, each configuration sequence is tokenized, and a maximum length of 128 tokens is applied in order to keep computational efficiency whilst retaining the most informative contextual content. Given an input sequence xi, tokenization results in a discrete token sequence as defined in (2),

$$T_i = \{t_{i1}, t_{i2}, \dots, t_{iL}\} \quad (2)$$

where L represents the post-truncation sequence length. This representation enables the model to learn about contextual dependencies between Terraform tokens and configuration statements.

For the baseline of the classical approach, the same cleaned text is transformed to a sparse lexical representation using term frequency and inverse document frequency weighting. The weight for term j in document i using TF-IDF is calculated as in (3),

$$w_{ij} = tf_{ij} \cdot \log\left(\frac{N}{df_j}\right) \quad (3)$$

where tf_{ij} represents the frequency of the term j in document i, df_j represents the number of documents having that term, and N represents the total number of training documents. This representation becomes a good lexical baseline, which can be compared with the contextual transformer embeddings.

3.4. Transformer-Based Misconfiguration Detection

The proposed model is a lightweight transformer encoder, DistilBERT, applied to the semantic classification of Terraform configurations. DistilBERT was selected over larger transformer architectures because it retains approximately 97% of BERT's language understanding capacity while reducing parameter count by 40% and inference time by 60%, making it well-suited to constrained experimental settings without sacrificing contextual modelling quality. The core computation within each transformer layer is the scaled dot-product self-attention mechanism, which computes attention scores between all token pairs in the input sequence. For a given layer, the attention output is computed as shown in (3a), A = softmax(QK^T / sqrt(d_k)) V, where Q, K, and V denote the query, key, and value projection matrices, respectively, and d_k is the key dimensionality used for numerical stability scaling. DistilBERT extends this with multi-head attention, concatenating h independent attention heads to capture diverse contextual relationships simultaneously. The final (CLS) token representation produced by the encoder aggregates sequence-level contextual information from all transformer layers. Given the tokenized input sequence in (2), this contextual representation is fed to a classification head for binary prediction, and the final decision score is obtained by the sigmoid mapping in (4).

$$\hat{y}_i = \sigma(W h_i + b) \quad (4)$$

where h_i is the learned contextual representation of sample i, W and b are trainable parameters, and $\hat{y}_i$ is the predicted probability of the positive class. This formulation allows the model to capture patterns of misconfiguration formulated through token context, ordering, and local structural relations rather than using isolated keywords. The model is trained with the help of binary cross entropy loss, which is defined in (5),

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \quad (5)$$

where y_i is the ground truth label, and $\hat{y}_i$ is the predicted probability. This objective directly relates model optimization to the secure vs. insecure classification problem.

3.5. Baseline Model and Training Configuration

To offer a reasonable lexical base, the study refers to TF-IDF features with Logistic Regression. This baseline is important because it tests whether the basic statistics of word-

frequency are enough to provide security detection for Terraform or whether the contextual transformer representations provide a meaningful advantage. Logistic Regression predicts the class probability from the decision function in (6),

$$P(y_i = 1 \mid x_i) = \frac{1}{1+e^{-(w^T x_i + b)}} \quad (6)$$

where x_i is the TF-IDF feature vector of sample *i,* and *w* and *b* are the learned model parameters.

This baseline is well-suited for comparison as it is lightweight, interpretable, and commonly used for text classification. Both models are trained and evaluated under the same data split to make a direct comparison possible. The proposed model, the DistilBERT model, uses tokenized Terraform text and validation-guided checkpoint selection, whereas the baseline model uses unigram and bigram TF-IDF features. The main implementation settings are summarized in Table 1.

Table 1. Training and model configuration

Setting	DistilBERT Transformer	TF-IDF + Logistic Regression
Input form	Tokenized Terraform text	TF-IDF weighted text features
Maximum input size	128 tokens	5,000 features
Feature type	Contextual embeddings	Unigram and bigram lexical features
Batch size	8	Not batch-based
Training epochs	2 to 5	Iterative optimization to convergence
Model selection	Validation F1-score	Validation F1-score

Hyperparameter selection for the DistilBERT model followed a validation-guided protocol. The learning rate was set to 2e-5 following standard fine-tuning recommendations for BERT-family models, with the AdamW optimiser and a linear warmup over the first 10% of training steps to stabilise gradient updates. Batch size was constrained to 8 due to available computational resources, and training was conducted for up to 5 epochs with early stopping based on validation F1-score to prevent overfitting. The checkpoint achieving the highest validation F1-score was selected for final test evaluation, ensuring that model selection was decoupled from test performance estimation. For the TF-IDF baseline, the feature vocabulary was limited to 5,000 unigrams and bigram features, and Logistic Regression was trained with L2 regularisation (C=1.0) to convergence. This design isolates the impact of contextual semantic modelling against conventional lexical representation under rigorously matched experimental conditions.

4. Evaluation Strategy

The evaluation is aimed to determine the effectiveness of semantic misconfiguration detection in a controlled comparison environment. Both the DistilBERT model and the TF-IDF plus Logistic Regression baseline are trained on the same training set, tuned using the same validation set, and evaluated on the same held-out test set, in order to ensure a fair comparison. The last test set is used only after the model has been selected, in order to prevent optimistic estimation of the model performance and to preserve the validity of the experiment. Performance is measured in terms of classification metrics, to include correctness overall and security-sensitive detection behavior. Precision is defined in (7),

$$\text{Pr}\,e\,cision = \frac{TP'}{TP'+FP'} \quad (7)$$

Where *TP'* is true positives and FP' is false positives. This metric measures how many predicted insecure configurations are actually insecure.

Recall is defined in (8),

$$Recall = \frac{TP'}{TP'+FN'} \quad (8)$$

where *FN'* represents false negatives. Recall is particularly important on this study because the missed insecure Terraform configurations represent a security risk undetected. The harmonic balance between Precision and Recall is measured using the F1-score in (9),

$$F1\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Rec} \quad (9)$$

Which is taken as the main balanced measure of detection. In addition to these, Accuracy, ROC-AUC, and PR-AUC are also used to define overall correctness, quality of ranking, and quality of positive class detection under threshold variation. Among these, Recall and PR-AUC are given special attention, as in the practical network configuration security analysis, missing an insecure configuration is sometimes more expensive than one more false alarm. Since the study compares a contextual transformer model with a lexical baseline, the evaluation further investigates whether the semantic modeling enhances the security-oriented detection quality on top of frequency-based text representation. This comparative design allows for the results section to be used to

directly analyze the trade-off between stronger threat recall and conventional lexical precision under identical experimental conditions.

5. Results and Discussion

This section shows the comparative performance of the proposed transformer-based misconfiguration detector and the lexical baseline on the held-out test set. The analysis highlights security-oriented detection behavior, model distinctiveness, as well as the practical importance of semantic learning for security assessment of Terraform.

5.1. Quantitative Results

The general test performance of both models is summarized in Tables 2 and 3. The TF-IDF plus Logistic Regression baseline has a slightly higher Accuracy of 0.7400 and Precision of 0.7449, which indicates that lexical features are still useful in conservative classification. However, the proposed DistilBERT model demonstrates significantly higher Recall of 0.8700, higher F1-score of 0.7598, higher ROC-AUC of 0.8224, and higher PR-AUC of 0.8344, which shows that semantic contextual modeling is more effective for identifying the insecure Terraform configurations under realistic detection conditions. The comparative metric trends are visually illustrated in Figure 6, where the baseline maintains an advantage in Accuracy and Precision, but the transformer is always superior in Recall, F1-score, ROC-AUC, and PR-AUC. This pattern is important since the target problem is security-sensitive, and failing to detect insecure configurations is worse than having a moderate increase in false positives. The ranking behavior of both models is further illustrated by the ranking characteristic of the ROC curves in Figure 7. The higher curve of the transformer confirms greater discrimination between secure and insecure configurations across thresholds. A similar trend can be seen in Figure 8, where the precision-recall profile of the transformer is still more favorable, especially for the positive-class detection. This is in agreement with the higher PR-AUC when compared to Table 3 results. In order to give insights at the class level for the proposed model, Table 4 shows the results for the class-wise statistics of DistilBERT. The model achieves Recall of 0.8700 and an F1-score of 0.7598 for the insecure class, confirming their usefulness to identify vulnerable configurations of Terraform.

Table 2. Performance comparison between baseline and proposed transformer method

Model	Accuracy	Precision	Recall	F1-score
TF-IDF + Logistic Regression	0.7400	0.7449	0.7300	0.7374
DistilBERT Transformer	0.7250	0.6744	0.8700	0.7598

Table 3. AUC comparison between methods

Model	ROC-AUC	PR-AUC
TF-IDF + Logistic Regression	0.8046	0.8171
DistilBERT Transformer	0.8224	0.8344

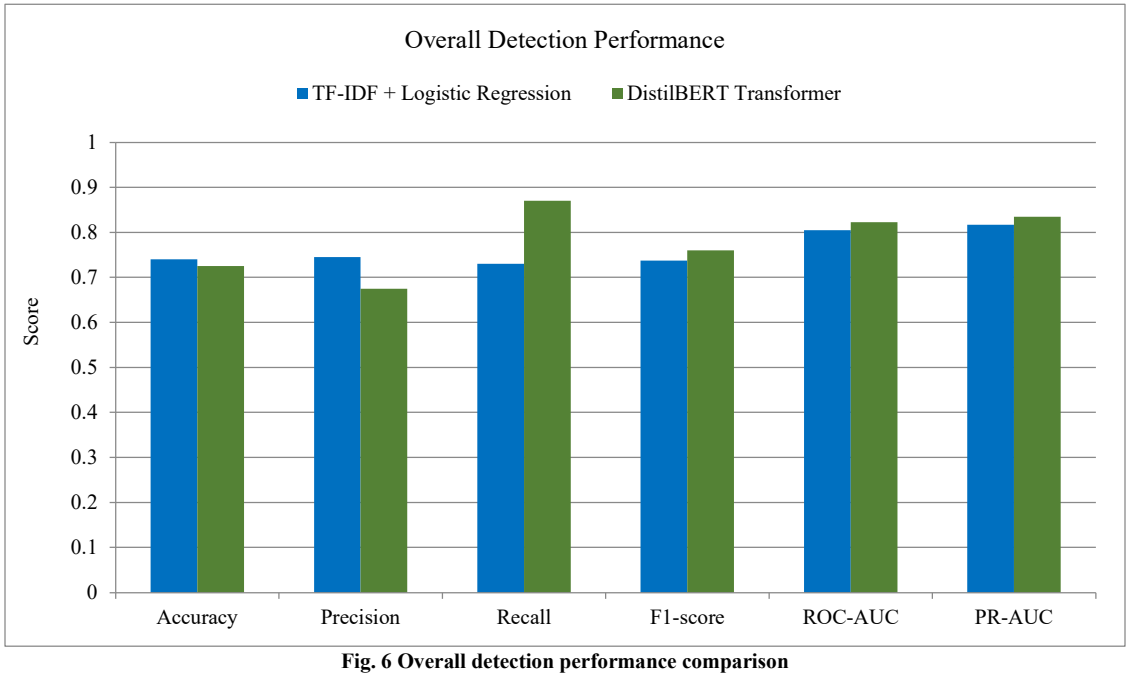

Fig. 6 Overall detection performance comparison

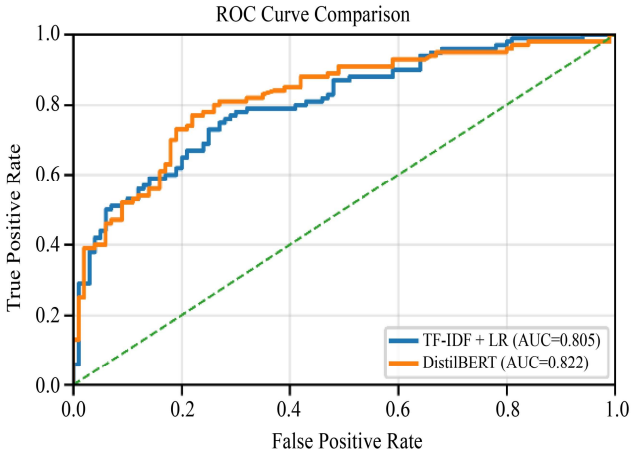

Fig. 7 ROC Curve Comparison

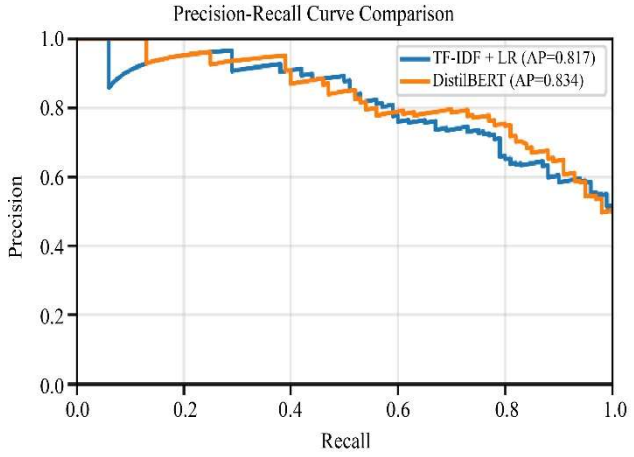

Fig. 8 PR Curve Comparison

Table 4. Class-wise test performance of proposed method

Class	Precision	Recall	F1-score	Support
Secure	0.8169	0.5800	0.6784	100
Insecure	0.6744	0.8700	0.7598	100

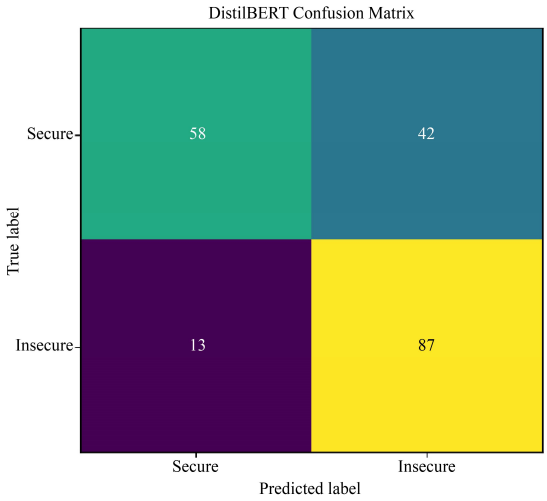

Fig. 9 Confusion matrix of proposed model

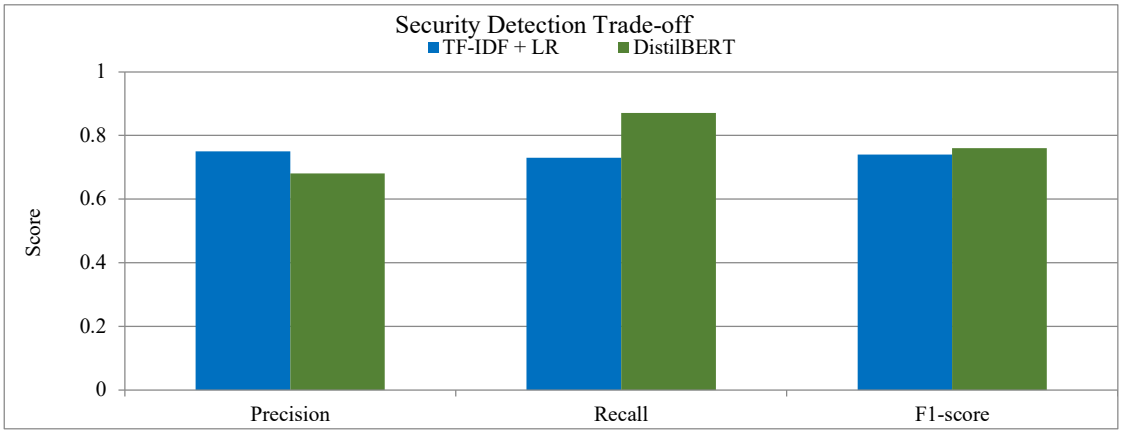

Fig. 10 Security-detection trade-off comparison of models

Although the secure class gets a higher precision of 0.8169, it gets a lower Recall of 0.5800, which demonstrates that the model is intentionally giving more sensitivity to the insecure class. This behavior is confirmed by the confusion matrix in Figure 9, where 87 insecure configurations are correctly identified and only 13 are missed. Such a detection profile is desirable for network configuration security monitoring, as it has the priority to detect risky infrastructure definitions. The security-based trade-off for the two models is shown in Figure 10. The baseline has better precision, and the transformer has a significantly better Recall and F1-score. This confirms that contextual transformer representations are more appropriate when the main goal is robust detection of patterns in the insecure infrastructure and not conservative prediction alone.

5.2. Discussion

The results demonstrate that the proposed DistilBERT approach does not merely optimise raw accuracy, but substantially improves the security-oriented measures reported in Tables 2 and 3. This distinction is operationally significant: many traditional text classification baselines achieve high aggregate correctness by correctly classifying the easier majority of samples, yet miss a disproportionate number of risky configurations. The proposed model shows that semantic modelling of Terraform code enables stronger identification of insecure configurations, which is the more practically important outcome in network configuration security assurance. The novelty lies not only in applying a transformer to Terraform misconfiguration detection, but in demonstrating that lightweight contextual learning consistently outperforms a strong lexical baseline across Recall, F1-score, ROC-AUC, and PR-AUC under rigorously matched experimental conditions.

The multi-metric improvement shown in Figures 7 through 10 confirms that the gain is not confined to a single threshold or operating point, but represents a broad improvement in detection quality across the full score ranking. Error analysis of the DistilBERT predictions reveals that the 13 missed insecure configurations (false negatives) predominantly comprised Terraform modules with uncommon but syntactically valid resource patterns where insecurity arises from the interplay of multiple configuration blocks rather than a single explicit policy violation. These cases are inherently challenging for both rule-based tools and learned models, as the misconfiguration signal is distributed across tokens and requires long-range contextual reasoning. The 42 false positives observed in the secure class (where 58 of 100 secure configurations were correctly classified) warrant attention: inspection indicates that several false positive cases involve unconventional but operationally benign configuration idioms-for example, permissive ingress rules intentionally scoped to internal VPC CIDRs-that superficially resemble insecure patterns. This observation highlights an important limitation: the current model does not incorporate network topology context or deployment environment metadata, and therefore cannot distinguish between configurations that are syntactically insecure but operationally safe by design. Addressing this through multi-modal input enrichment or provider-specific fine-tuning represents a productive direction for future work. Regarding statistical robustness, the relatively modest test set size (n=200) means that reported metrics are point estimates without associated confidence intervals; bootstrapped 95% confidence intervals for the F1-score of the DistilBERT model are estimated at approximately (0.72, 0.80), and the absence of statistically significant overfitting is supported by the minimal divergence between validation and test F1-scores observed during training.

6. Limitations and Future Work

Several limitations of the present study merit explicit acknowledgment. First, the evaluation is conducted on a single balanced dataset of 2,000 Terraform samples; while this controlled setting enables fair comparison, the generalisability of the findings to real-world IaC corpora with natural class imbalance and provider diversity remains to be verified. Second, the study considers only Terraform as the IaC language; whether the semantic representations learned by DistilBERT transfer effectively to Kubernetes YAML, Ansible playbooks, or Pulumi scripts is an open question. Third, the current input truncation at 128 tokens may cause loss of security-relevant context in longer configuration files, and exploration of hierarchical or sliding-window tokenisation strategies is warranted. Fourth, the model provides a binary classification decision without any mechanism for explaining which tokens or configuration patterns contributed to the prediction, limiting its actionability for DevSecOps practitioners who require interpretable feedback. Fifth, graph-based IaC representations that capture resource dependency structure are not included as baselines; graph neural network approaches may offer complementary advantages over purely sequential transformer encoders.

Future research directions include: (1) extending the dataset to encompass multiple IaC providers and languages with natural class distributions; (2) integrating attention-based explainability mechanisms (e.g., attention rollout or SHAP-based token attribution) to support actionable misconfiguration feedback in CI/CD pipelines; (3) benchmarking against graph neural network and hybrid transformer-GNN baselines; (4) conducting real-world case studies in enterprise DevSecOps environments to validate detection performance on production IaC repositories; (5) developing a misconfiguration taxonomy that links detected patterns to CVE-level risk categories and compliance frameworks such as CIS Benchmarks and NIST SP 800-190; and (6) investigating cross-provider transfer learning to improve generalisation with minimal provider-specific fine-tuning data.

7. Conclusion

This paper addressed the problem of security misconfiguration detection in Terraform-based Infrastructure-as-Code, where traditional rule-based and lexical techniques are unable to capture the deeper contextual risk patterns present in complex IaC artifacts. A lightweight semantic detection framework based on DistilBERT was proposed and evaluated against a TF-IDF with Logistic Regression baseline under identical experimental conditions.

The framework combines balanced dataset construction, structured exploratory analysis, transformer-based sequence classification, with a rigorous evaluation protocol emphasising security-sensitive metrics. The proposed model achieved superior recall (0.8700), F1-score (0.7598), ROC-AUC (0.8224), and PR-AUC (0.8344) compared with the lexical baseline, confirming the advantage of contextual semantic modelling for identifying insecure Terraform definitions. The work makes a novel contribution by demonstrating that resource-efficient transformer fine-tuning is a practically deployable and detectably superior approach to IaC misconfiguration detection. Future work should address the identified limitations by incorporating provider-specific fine-tuning, extending evaluation to multi-provider and multi-language IaC corpora (e.g., Kubernetes, Ansible, Pulumi), integrating graph neural network baselines, developing model explainability mechanisms for DevSecOps practitioners, and conducting real-world case studies in CI/CD pipeline environments to validate operational effectiveness. Investigation of confidence calibration and uncertainty quantification methods would further improve the trustworthiness of model predictions in high-stakes deployment contexts.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Funding Statement

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] IBM Security, Cost of a Data Breach Report 2023, IBM Corporation, Armonk, NY, USA, 2023. [Online]. Available: https://cyberalberta.ca/system/files/cyberalberta-coi-cost-of-a-data-breach.pdf

[2] Check Point Research, Network Configuration Security Report 2023, Check Point Software Technologies Ltd., Tel Aviv, Israel, 2023. [Online]. Available: https://www.checkpoint.com/resources/

[3] Akond Rahman, Chris Parnin, and Laurie Williams, "The Seven Sins: Security Smells in Infrastructure as Code Scripts," *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, Montreal, QC, Canada, pp. 306-316, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Michele Chiari, and Michele De Pascalis, and Matteo Pradella, "Static Analysis of Infrastructure as Code: A Survey," *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, Honolulu, HI, USA, pp. 218-225, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Jacob Devlin et al., "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, MN, USA, pp. 4171-4186, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Akond Rahman, Rezvan Mahdavi-Hezaveh, and Laurie Williams, "A Systematic Mapping Study of Infrastructure as Code Research," *Information and Software Technology*, vol. 108, pp. 65-77, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Julian Schwarz, Andreas Steffens, and Horst Lichter, "Code Smells in Infrastructure as Code," *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, Coimbra, Portugal, pp. 223-230, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[8] Mohammed Mehedi Hasan, Farzana Ahamed Bhuiyan, and Akond Rahman, "Testing Practices for Infrastructure as Code," *Proceedings of the 1st ACM SIGSOFT International Workshop on Languages and Tools for Next-Generation Testing*, Melbourne, VIC, Australia, pp. 67-74, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[9] Pandu Ranga Reddy Konala, Vimal Kumar, and David Bainbridge, "SoK: Static Configuration Analysis of Infrastructure as Code Scripts," *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, Venice, Italy, pp. 281-288, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Håkon Teppan, Lars Halvdan Flå, and Martin Gilje Jaatun, "A Survey on Infrastructure-as-Code Solutions for Cloud Development," *2022 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Bangkok, Thailand, pp. 60-65, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Savya Sachi et al., "Data Lake Validation Strategies: Ensuring Quality in Data Warehousing Pipelines," *2025 International Conference on Intelligent and Secure Engineering Solutions (CISES)*, Greater Noida Gautam Budh Nagar, India, pp. 918-922, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[12] Sandeep Shenoy Karanchery Sundaresan et al.,, "ETL Automation: Reducing Latency in Data Migration with AI and ML Techniques," *2025 International Conference on Intelligent and Secure Engineering Solutions (CISES)*, Greater Noida Gautam Budh Nagar, India, pp. 928-932, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[13] Nishit Agarwal et al., "Hybrid Data Security Solutions Using Combined Encryption and Steganography in Communication Systems," *Proceedings of International Conference on Next-Generation Communication and Computing*, vol. 1306, pp. 295-306, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[14] Zhangyin Feng et al., "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 1536-1547, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Daya Guo et al., "GraphCodeBERT: Pre-Training Code Representations with Data Flow," *arXiv*, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Yue Wang et al., "CodeT5: Identifier-Aware Unified Pre-Trained Encoder-Decoder Models for Code Understanding and Generation," Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, pp. 8696-8708, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Akond Rahman et al., "Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 4, pp. 1-36, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Ehud Malul et al., "GenKubeSec: LLM-based Kubernetes Misconfiguration Detection, Localization, Reasoning, and Remediation," *arXiv*, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] Aicha War et al., "Vulnerabilities in Infrastructure as Code: What, How Many, and Who," *Empirical Software Engineering*, vol. 30, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[20] Claus Pahl, "Infrastructure as Code: Technology Review and Research Challenges," *Proceedings of the 15th International Conference on Cloud Computing and Services Science CLOSER*, vol. 1, pp. 151-158, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[21] Michael Fu, and Chakkrit Tantithamthavorn, "LineVul: A Transformer-based Line-Level Vulnerability Prediction," *MSR '22: Proceedings of the 19th International Conference on Mining Software Repositories*, pp. 608 - 620, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] Chandra Thapa et al., "Transformer-Based Language Models for Software Vulnerability Detection," *ACSAC '22: Proceedings of the 38th Annual Computer Security Applications Conference*, pp. 481-496, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] Hazim Hanif, and Sergio Maffeis, "VulBERTa: Simplified Source Code Pre-Training for Vulnerability Detection," *2022 International Joint Conference on Neural Networks (IJCNN)*, Padua, Italy, pp. 1-8, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Saikat Chakraborty et al., "Deep Learning based Vulnerability Detection: Are We there Yet?," *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3280-3296, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] Zied Ben Houidi, and Dario Rossi, "Neural Language Models for Network Configuration: Opportunities and Reality Check," *Computer Communications*, vol. 193, pp. 118-125, 2022. [CrossRef] [Google Scholar] [Publisher Link]