MisConfAI: A Framework for Detecting Configuration Vulnerabilities in Complex Systems

Savi Juneja, and Geeta Yadav

Indian Institute of Technology Ropar, Punjab, India

The Security Gap: Why Misconfigurations Persist

- **Problem:** Configuration vulnerabilities are frequently in the OWASP Top 5, stemming from insecure settings, not source code flaws.
- Software systems rely heavily on configuration files to define security, network, database, and application behavior.
- **Impact:** Lead to severe breaches (e.g., DoD Azure leak, Uber).
- **Goal:** Bridge the semantic gap by leveraging contextual language modeling.

Proposed Solution:

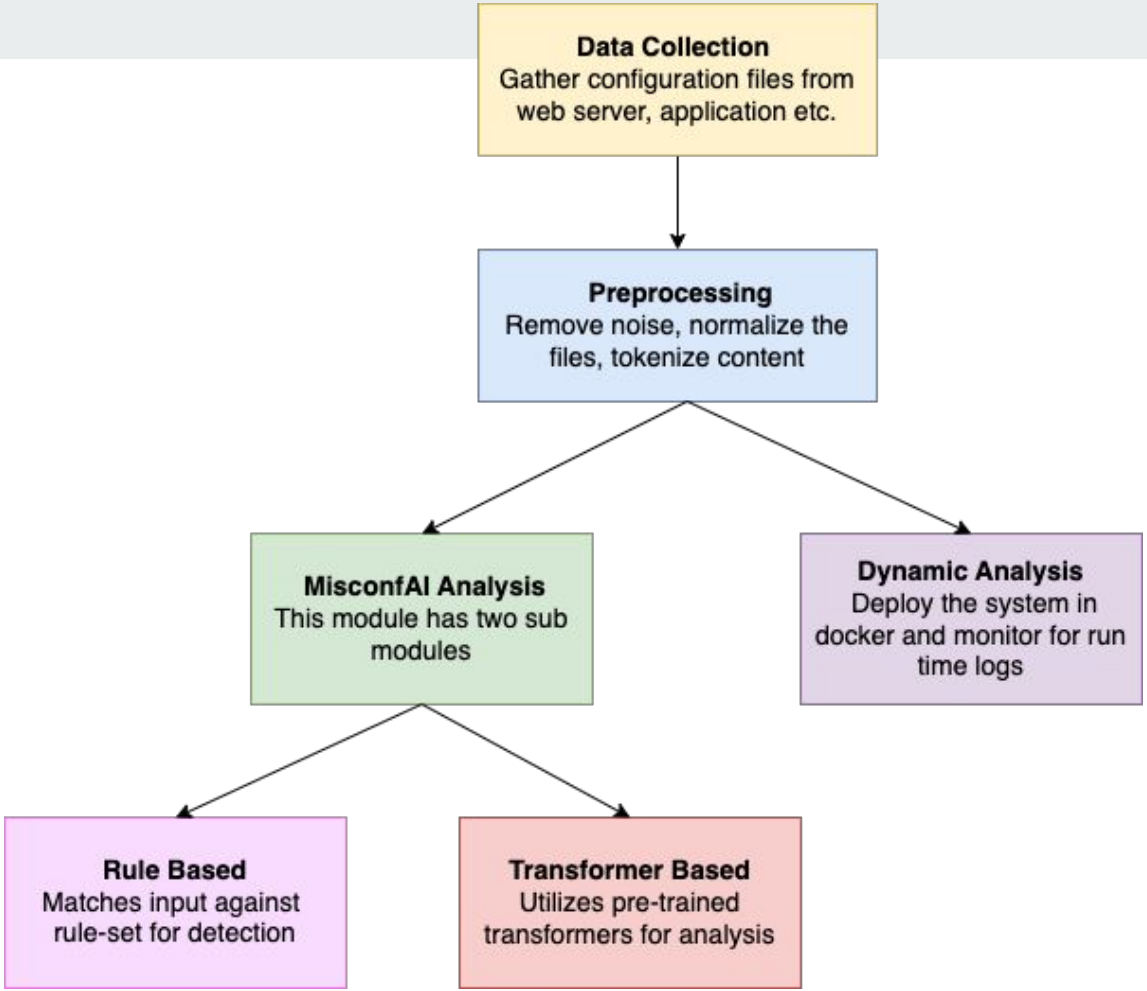
MisConfAI integrates Rule Based Analysis and Transformer Based Analysis. And We also explored Dynamic Analysis.

- **Rule-Based Analysis:** Fast, high-precision detection of strict, known patterns.
- **Transformer-Based Analysis :** Uses fine-tuned LLMs for semantic understanding and contextual flaw detection.
- **Dynamic Analysis:** Runtime validation in Dockerized environments to confirm the *operational impact* of the static findings.

Workflow

The diagram shows the workflow

of our approach.



Data Collection

Custom Dataset: Critical for training a domain-specific model.

- **Source:** GitHub, industry benchmarks, and synthetic scripts for augmentation.
- **Scale:** 4550 files (2315 labeled as Vulnerable).
- **Domain Diversity (5 Categories):** Web Server, Application, Cloud, Database, DevOps/CI-CD.

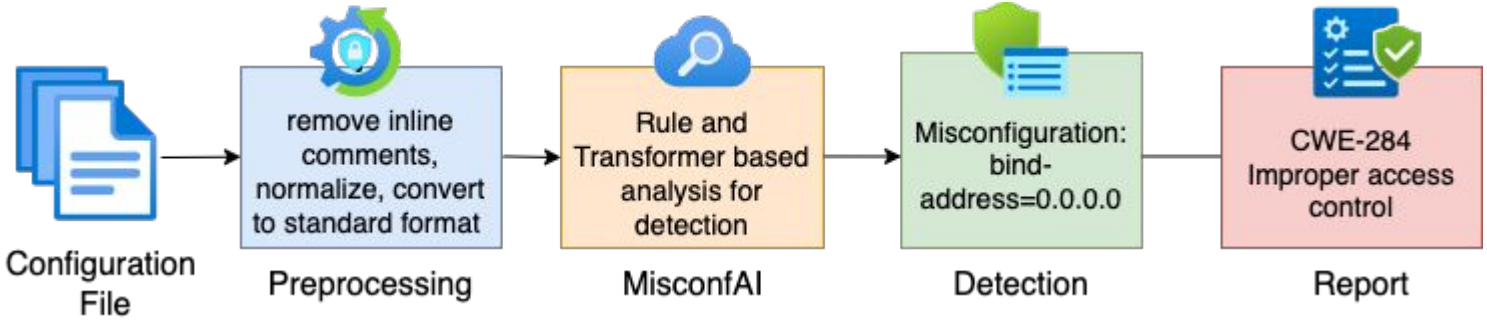
Vulnerable Files

The table shows the total number of files and vulnerable files among them.

Category	Total Files	Vulnerable Files
Web Server	943	548
Application Framework	783	353
Cloud Infrastructure	813	469
Database Configuration	1295	577
DevOps/CI-CD	716	368
Total Files	4550	2315

MisConfAI

- Rule based analysis
  YAML rules (regex + CWE mapping) to detect known misconfiguration patterns.
- Transformer based analysis
  Fine-tuned models trained on labelled configs to detect both known and unknown misconfiguration with semantic understanding.

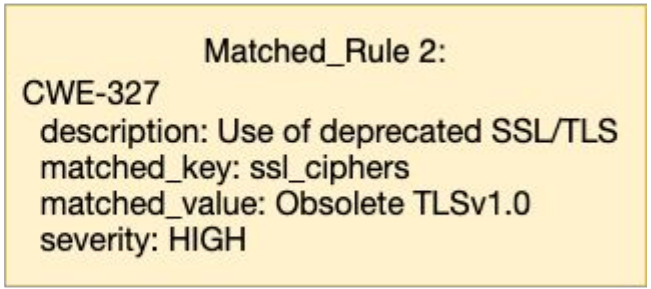


Rule Based Analysis

- Uses YAML rules to detect misconfigurations
- Each rule includes:
  - **Key** to check
  - **Regex** pattern to match values
  - **CWE ID**, **severity**, and **description**
- Files labelled as **vulnerable** if any rule matches

Rule-based misconfiguration detector analysis

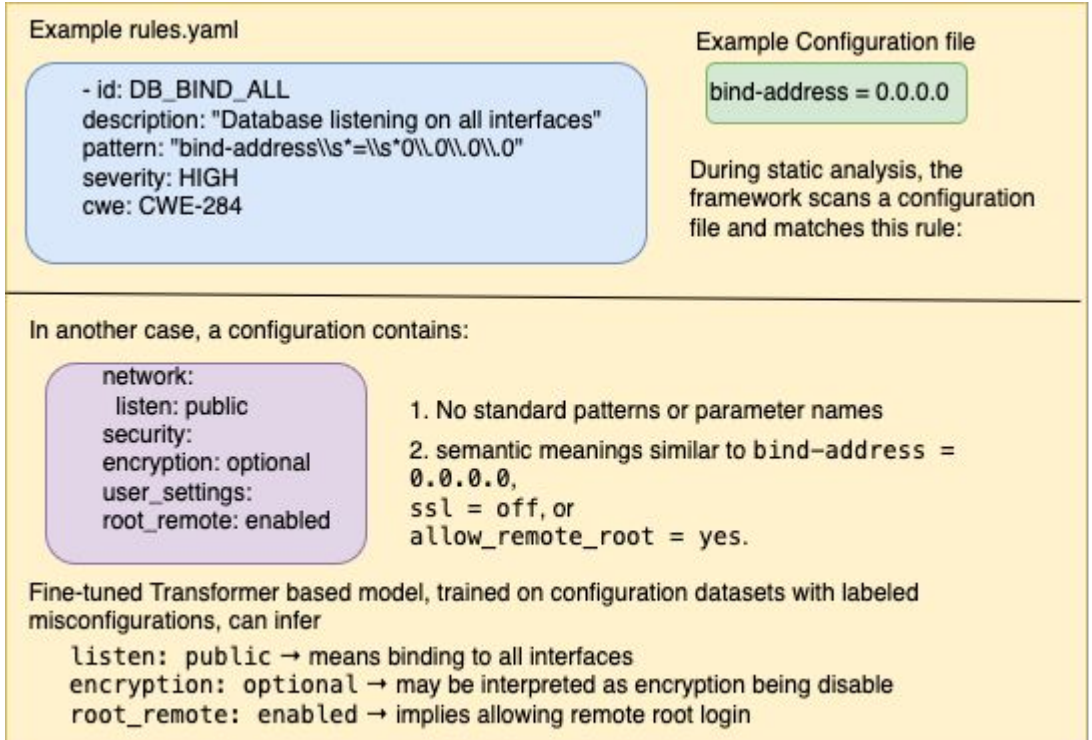
Rule-Based Baseline achieved 87.84% Precision, but only **54.90% Recall**. This means it missed nearly half the vulnerabilities.



Limitation

As shown, the recall is 54%.

The rule will miss the keyword with similar meanings.

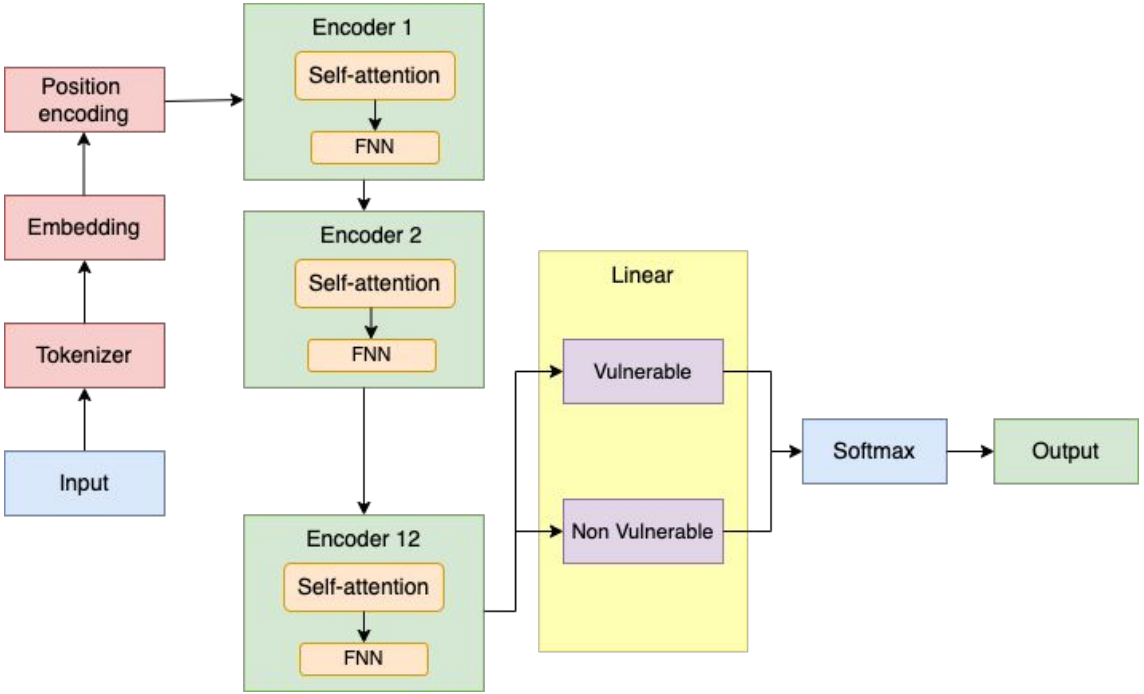


Transformer Based Analysis

- Uses fine-tuned **transformer models** on configuration files
- Models evaluated:
  - **BERT**
  - **CodeBERT**
  - **RoBERTa**
  - **ELECTRA**
  - **XLNet**
- Configuration files as structured text for classification

Model Architecture

- Input
- Input embedding
- 12 Encoder layer
- Linear layer
- Softmax
- Output



Tokenizer and Input Embedding:

- BERT, ELECTRA
  - Uses WordPiece tokenizer
  - Input Embedding: WordPiece(token)+ Position(i) + Segment(A/B)
- RoBERTa, CodeBERT
  - Uses Byte-Pair Encoding (BPE) tokenizer
  - Input Embedding: BPE(token)+Position(i)
- XLNet
  - Uses SentencePiece tokenizer
  - InputEmbedding: SentencePiece(token)+RelativePos(i)+Segment(A/B)

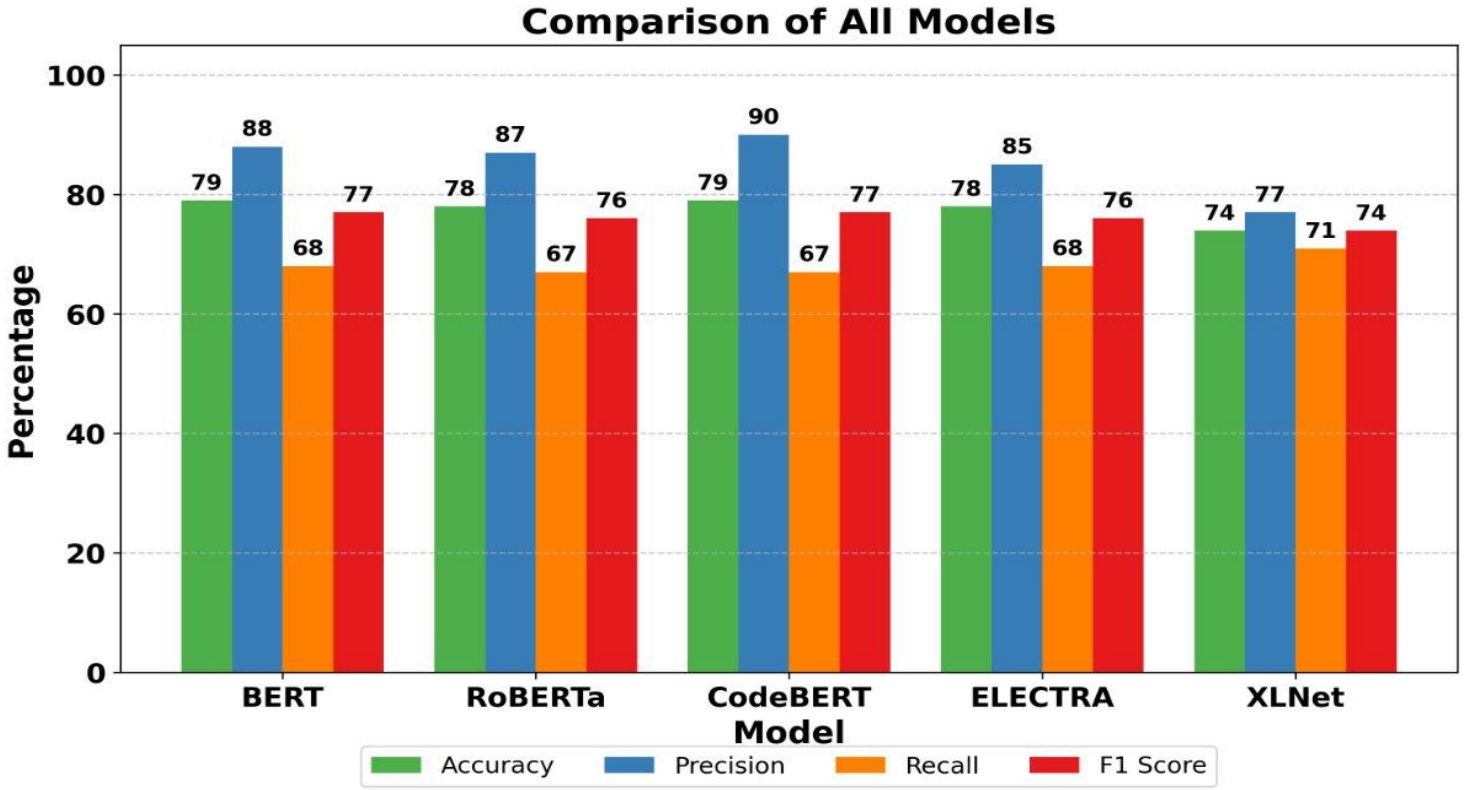
Training Configurations:

- Batch size : 8
- Epochs : 5
- Learning Rate : 2e-5
- Optimizer : AdamW
- Loss Function : Binary Cross Entropy
- Train-validation split : 80-20

Result per Category showing the Accuracy of each model

Category	Vulnerable Files	BERT (%)	RoBERTa (%)	CodeBERT (%)	ELECTRA (%)	XLNet (%)
Application	65	78	78	77	77	80
Cloud	84	60	55	61	60	68
Database	120	53	55	53	53	55
DevOps	90	60	57	57	61	66
Web Server	104	95	95	95	95	91
Avg. True positive (%)		69	68	68	69	72

Comparison of all models



Dynamic Analysis: Methodology

- Detect runtime misconfigurations by analysing system logs generated from deployed configurations.
- Vulnerable and non-vulnerable configurations deployed in **Docker Containers**.
- Simulated normal and attack activity
- Logs are generated in mysql_logs for each container.
- Each container writes logs to its mapped directory.

Data Collection and Result

- Logs matched against CWE-mapped patterns (e.g., DROP DATABASE →CWE-287).
- Generated binary labels:
  - 1: Vulnerable log event
  - 0: Non Vulnerable log event
- The resulting dataset exhibits class imbalance
- With 90,050 non-vulnerable samples and 2,298 vulnerable samples

Conclusion and Key Takeaways

- **MisConfAI successfully addresses the semantic gap** in configuration vulnerability detection.
- **CodeBERT is the optimal model** for this domain, achieving a robust **77% F1 Score**.
- **Impact:** A tool to identify configurations that would otherwise be missed.
- **Future Work:**
  - **Dynamic Analysis Automation:** Log analysis for all five interfaces (not just Database) to scale runtime validation.
  - **Real-Time Predictive Models:** Develop mechanisms to alert administrators to potential security issues *in real time* using the insights gained from our hybrid analysis.

Q&A

Thankyou