Predicting Security Weaknesses in Microservice Architectures using Structural Metrics – Short Paper

S. Benni[1], M. Hathat[1], J. Vergara-Vargas[2,3], S. Zellagui[1], C. Tibermacine[2] and S. Sadou[2]

[1] École Supérieure d'Informatique (ESI), Algiers, Algeria

[2] IRISA, Université Bretagne Sud, France

[3] Universidad Nacional de Colombia, Bogota, Colombia

ICSOC 2025, Dec. 2nd 2025

Shenzhen, China

Outline

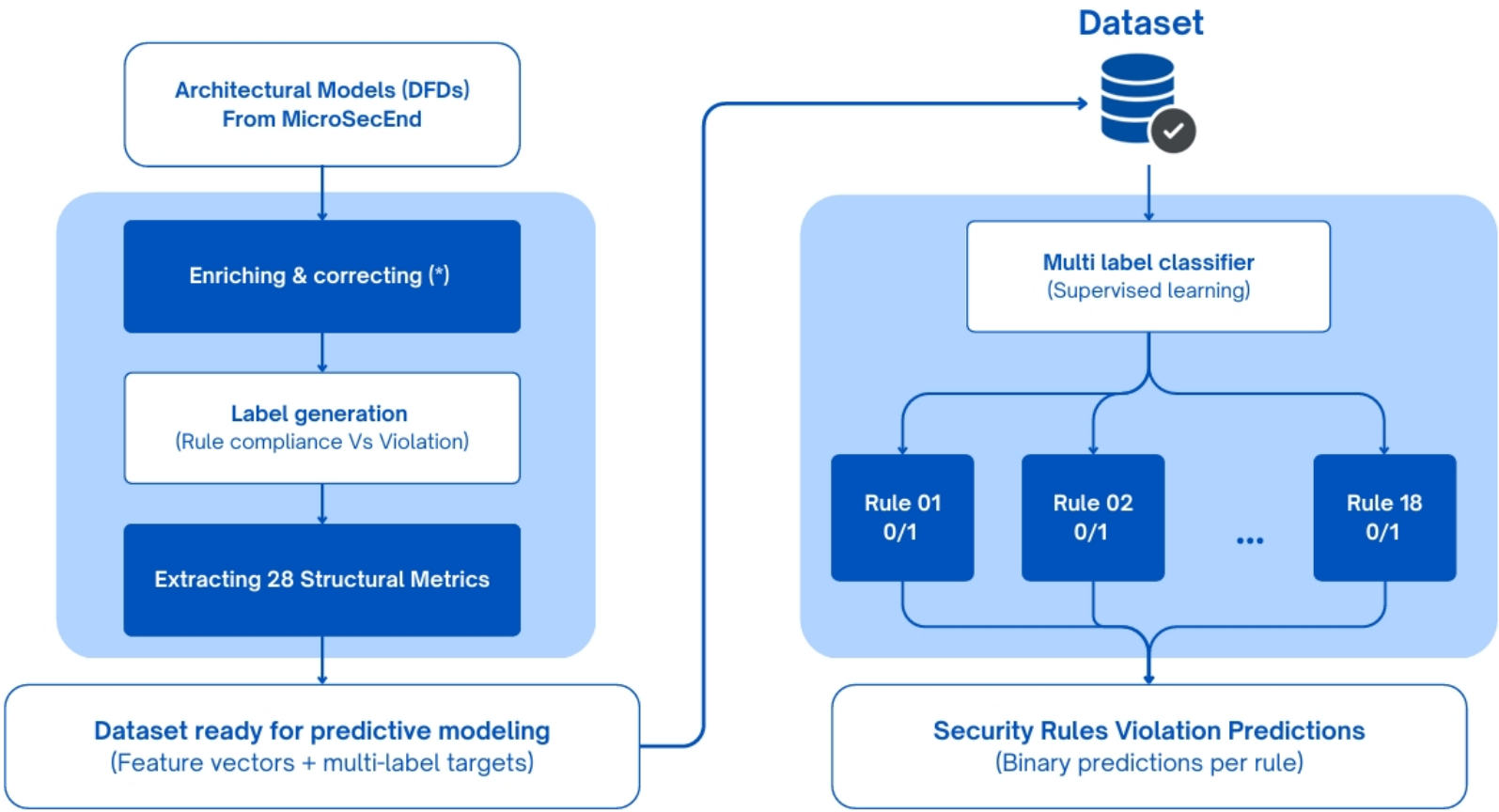
1. Introduction: Problem Statement and Approach
2. Dataset Preparation
3. Experimentation and Results
4. Conclusion: Related Work, Wrap-Up & Future Work

Outline

Introduction

- Microservice architectures bring many *ilities*: fine-grained scalability, modular deployability and agility.
- They also increase security risks: multiplication of running processes ⇒ larger attack surface exposure.
- Post-deployment correction of security weaknesses is costly and disruptive.
- Goal: Predict security weaknesses at the design stage using architectural metrics.
- Hypothesis: Architecture-level structural knowledge (patterns) can serve as reliable predictors of security compliance.

General Approach



- Use Data Flow Diagrams (DFDs) from MicroSecEnD dataset.
- Extending DFDs, measuring 28 structural metrics and labelling.
- Training a multi-label classifier.

Outline

Dataset Preparation

Labels = Security Rules – Rule Violation = Security Weakness

- 17 security rules from OWASP, NIST, and Cloud Security Alliance.
- Rules cover authentication, encryption, logging, availability, and secret management.

Features = Architectural Metrics

- We defined 28 structural metrics that we measured on DFDs.
- Metrics span system size, coupling, communication complexity, centrality, interface and exposure, observability and more.

Dataset Details

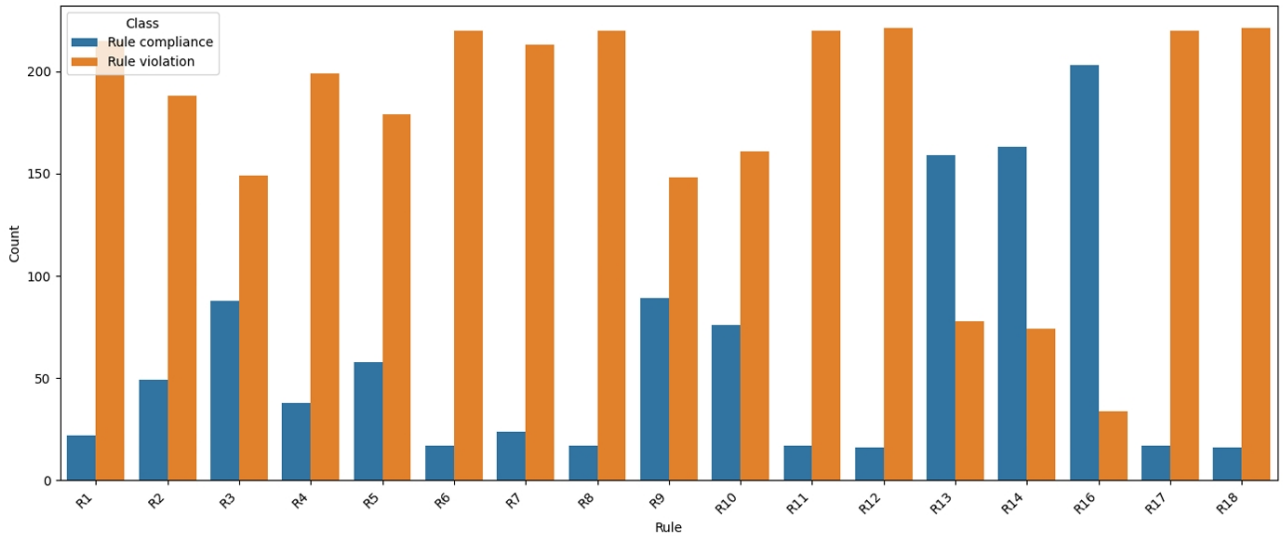
- Contains 226 Data Flow Diagrams of Java Spring-based software systems.
- Each system is represented by a base model with derived variants in which some security rules are violated.
- Labelled with compliance to the 17 security rules.

Preprocessing and Labeling

- Binary Relevance Strategy : converted multiple labels into binary labels indicating compliance or violation per rule.
- Why one classifier per rule?
  1. interpretable outputs.
  2. tailored handling of class imbalance.
  3. performance and computational scalability.
- We addressed inconsistencies and completed missing variants.
- This resulted in binary labels for 237 DFD models.
- Each DFD encoded as a 28-dimensional feature vector.

Prediction Modeling

- Multi-label classification ⇒ Binary classif. per rule (17 in total).
- Different complementary algorithms:
  1. Decision Trees (DT): interpretability.
  2. Support Vector Machines (SVM): non-linearity.
  3. Random Forests (RF): ensemble robustness.
- Address class imbalance (see bar chart below) with cost-sensitive learning (class weights).



Outline

Experimentation and Results

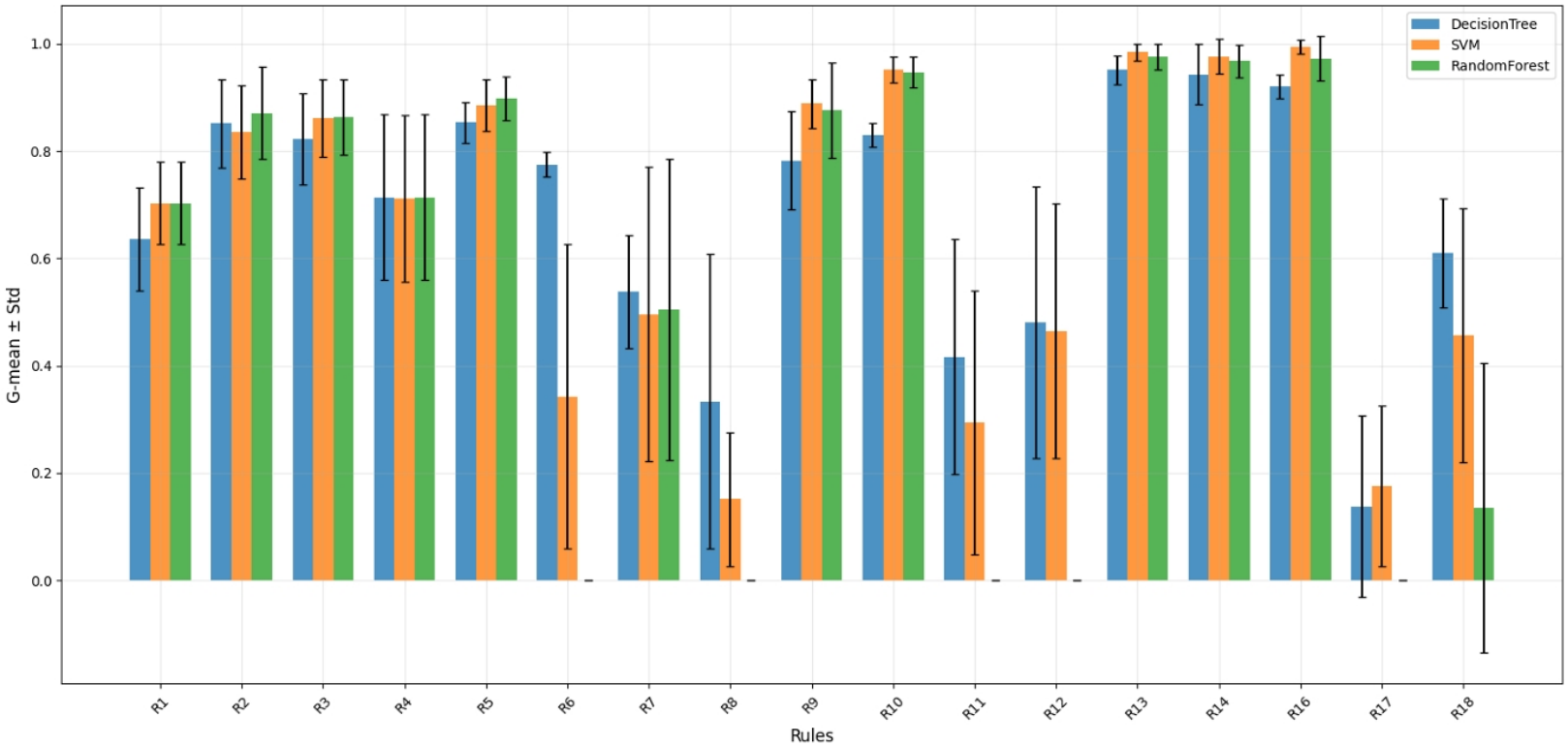
Experimentation Running

- 80% of the dataset for training and 20% for testing with 5-fold stratified cross-validation.
- Measure TP Rate (Sensitivity), TN Rate (Specificity) and G-Mean, for managing class imbalance as recommended in the literature.

Classifier Performance

- SVMs often achieved the best performance and stability.
- RF and DT yielded comparable results on several rules.
- All models struggled on rules R8, R11 and R17 (related to encryption and authN policies) with G-Mean dropped < 0.3.

Classifier Refinement



- Rules grouped into 3 categories based on performance/stability.
- Optimizations: feature selection and tuning improved performance and stability for some rules.
- Persistent poor results for some other rules due to confirmed (quantified) data quality issues.

Final Models and Interpretability

Final Classifiers

- Selected based on cross-validated G-Mean and stability in the first two groups of rules (10 out of 17).
- SVMs for R9, R10, R13, R14, and R16; RFs for R1-R5.
- TPR = 0.977, TNR = 0.832, G-Mean = 0.897

Key Features for Interpretability

- Coupling, flow complexity and exposure, *e.g.* `cycle_ratio_and_count`, `longest_shortest_path` and `num_external_entities`.
- Provide interpretable insights for proactive architectural security assessment.

Outline

1. Introduction: Problem Statement and Approach
2. Dataset Preparation
3. Experimentation and Results
4. Conclusion: Related Work, Wrap-Up & Future Work

Related Work

1. Metric-based methods : tailored for microservices, but most of the time, not related to security.
2. Architecture-centric methods: based on rule checking or knowledge graphs, but do not leverage AI (ML).

No solution: i) based on AI (ML), ii) targeting security and iii) considering microservice-based architectures.

Conclusion

- Two main contributions:
  1. Tackled the challenge of building a security-labeled microservice architecture dataset.
  2. Showed that architecture alone can help predict security issues early.
- Key result: Security is a measurable architectural quality.
- Presented model predicted 10 out of 17 rules correctly.
- Approach enables early, cost-effective detection of some security issues ⇒ possible integration in DevSecOps workflows.
- Limitation: current structural metrics cannot encode protocol low-level details to detect some weaknesses.

Future Work

- Incorporate semantic and configuration-level aspects (richer features) to capture broader security weaknesses.
- Assess scalability and generalizability on larger datasets.
- Integrate predictive capability into architecture modeling by building a full, explainable design-to-prediction pipeline.

Thank you for your attention!

Questions?

chouki.tibermacine@univ-ubs.fr