UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Improving Vulnerability Prediction of JavaScript Functions Using Process Metrics

Tamás Viszkok, Péter Hegedűs, Rudolf Ferenc

16th International Conference on Software Technologies
Online, 6th July 2021

Motivation

- Code security is a crucial non-functional property
  - E.g., IoT, network enabled devices, everything is „connected”
  - Vulnerabilities in source code are one of the enablers of cyber-crime
- Detecting vulnerabilities in source code
  - SAST tools
    - High false-positive rates
  - ML models
    - What should be the features?

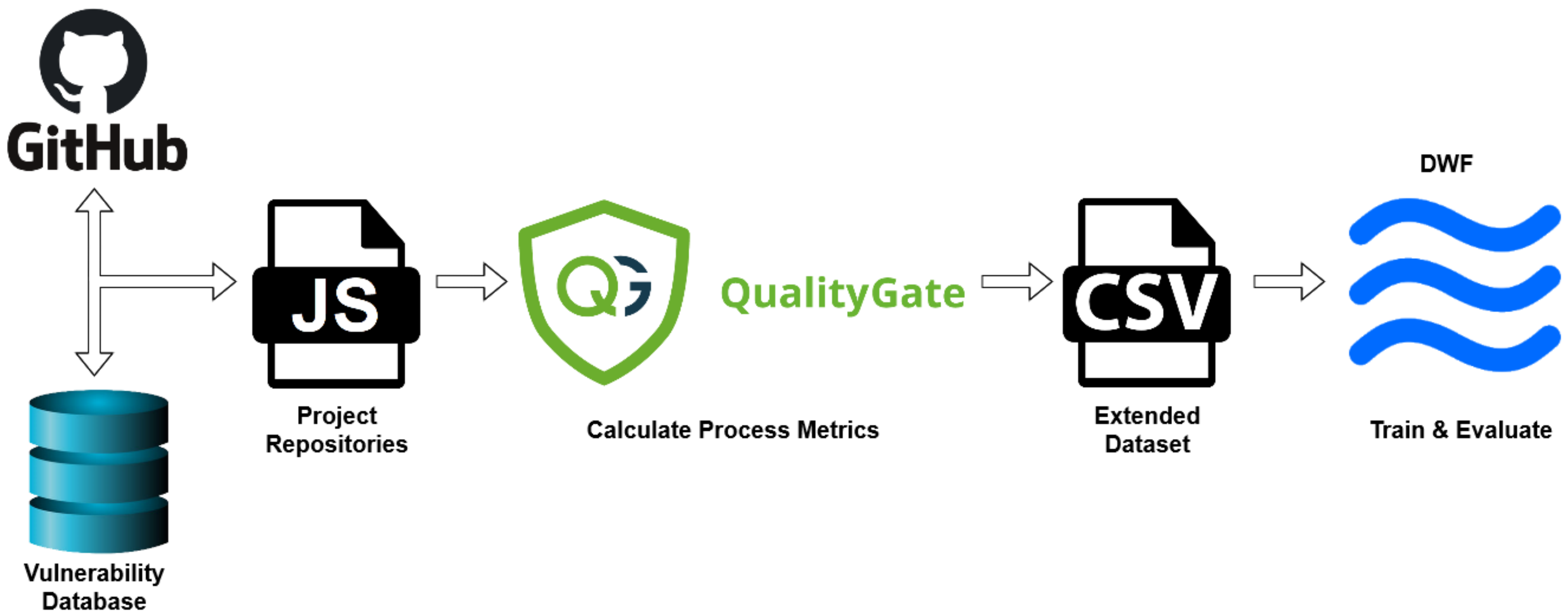
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Research Goals

- Context
  - We showed in a previous work [1] that **static source code metrics** are good predictors of JS vulnerabilities
- Goal of current research
  - Investigate the impact of **process metrics** on ML prediction
- RQs
  - **RQ1:** Can process metrics as features improve existing JavaScript vulnerability prediction models based only on static source code metrics?
  - **RQ2:** If process metrics do improve the performance of vulnerability prediction models, how significant it is in terms of precision, recall, and F-measure?

[1] Ferenc, R., Hegedűs, P., Gyimesi, P., Antal, G., Bán, D., and Gyimóthy, T. (2019). Challenging Machine Learning Algorithms in Predicting Vulnerable JavaScript Functions. In Proceedings of the 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, RAISE '19, page 8–14. IEEE Press.

Research Process



https://doi.org/10.5281/zenodo.4590021

Features and ML Models

- 42 static source code metrics (baseline)
  - Results from previous work [1]
  - McCabe's Complexity, Lines of Code, number of Parameters, …
- 19 process metrics
  - Extracted from version control system
  - Average Time Between Changes, Average Number of Modified Lines, Number of Contributors, …
- Applied ML models
  - SVM, KNN, Linear/Logistic Regression, Naive Bayes, Decision Trees, Random Forest, Neural Networks

Vulnerability Dataset

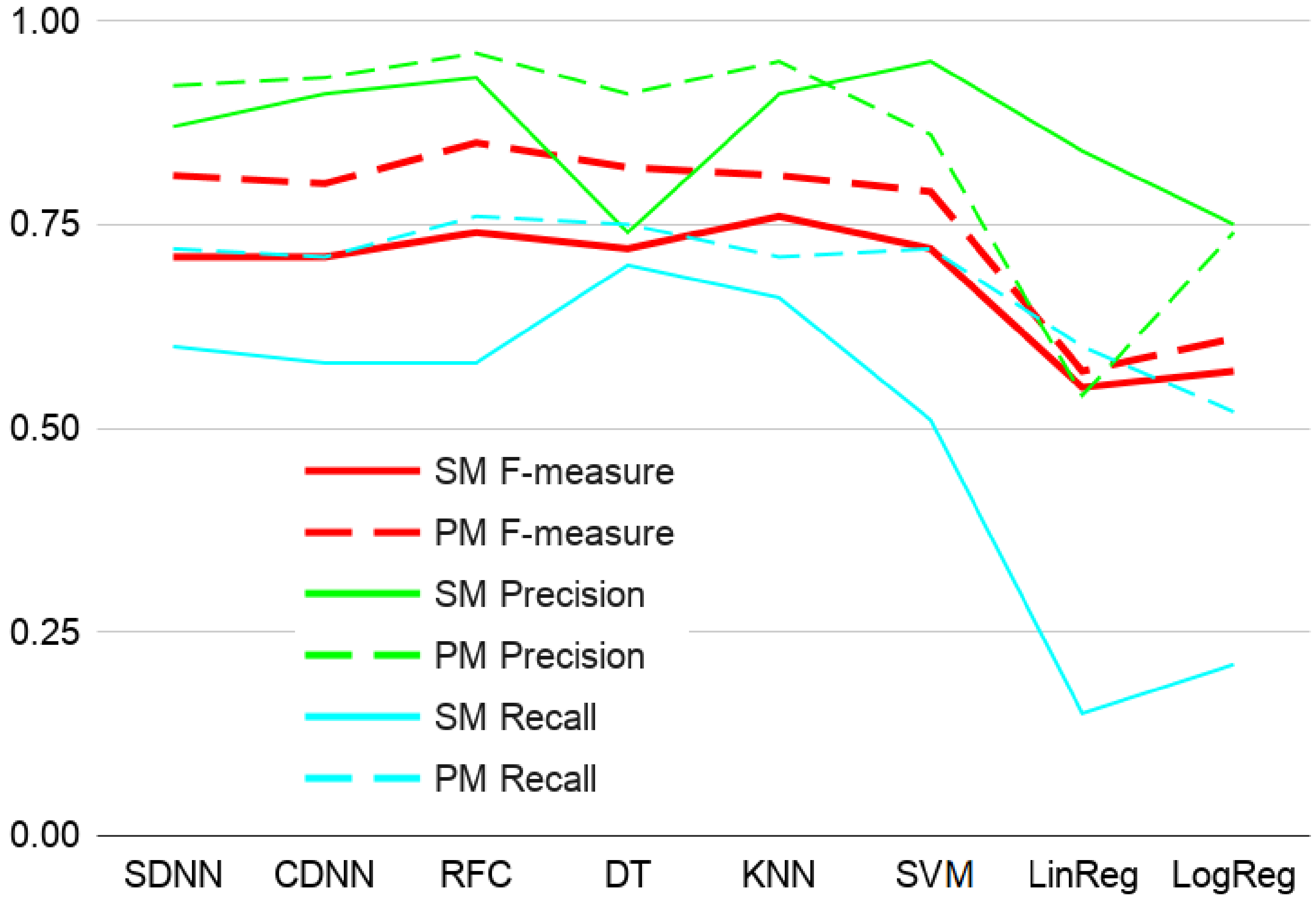
- 12,125 JavaScript functions
  - Vulnerable/not vulnerable flags
  - Set of features
    - Static source code metrics
    - Process metrics
- Imbalanced
  - ~15% vulnerable entries
  - Data re-sampling is needed before ML

Results – Improvement

Classifier	TP	TN	FP	FN	Accuracy	Precision	Recall	F-measure
RFC	699	7054	24	261	**96.5%**	96.7%	72.8%	**83.1% (+11.8%)**
DT	**723**	7006	72	**237**	96.2%	90.9%	**75.3%**	82.4% (+10.8%)
CDNN	685	7027	51	275	95.9%	93.1%	71.4%	80.8% (+10%)
SDNN	665	7037	41	295	95.8%	94.2%	69.3%	79.8% (+8.7%)
KNN	613	7059	19	347	95.5%	**97.0%**	63.9%	77.0% (+0.6%)
SVM	548	**7060**	**18**	412	94.7%	96.8%	57.1%	71.8% (+5%)
LogReg	332	7007	71	628	91.3%	82.4%	34.6%	48.7% (+15.6%)
LinReg	274	7051	27	686	91.1%	91.0%	28.5%	43.5% (+17.4%)
NB	115	6779	299	845	85.8%	27.8%	12.0%	16.7% (+1.4%)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Results – Overview



UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Conclusions

- Process metrics significantly improve the predictive power of JavaScript vulnerability prediction models
  - Average improvement
    - 8.4% in F-measure
    - 3.5% in precision
    - 6.3% in recall
  - All significant based on a McNemar’s test
- Best performing model was Random Forest
  - F-measure of 0.85 (0.96 precision and 0.76 recall)

Acknowledgement

This work was supported by the **SETIT** project (no. 2018-1.2.1-NKP-2018-00004) and the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program (**MILAB**). The research was partly supported by the EU funded project **AssureMOSS** (Grant no. 952647).