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?
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?

Research Process

GitHub → Project Repositories → QualityGate → CSV → Train & Evaluate

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

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

![Graph showing various metrics for different algorithms]

- SM F-measure
- PM F-measure
- SM Precision
- PM Precision
- SM Recall
- PM Recall

Algorithms compared:
- SDNN
- CDNN
- RFC
- DT
- KNN
- SVM
- LinReg
- LogReg

ICSOFT 2021, Online, 6th July 2021
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).