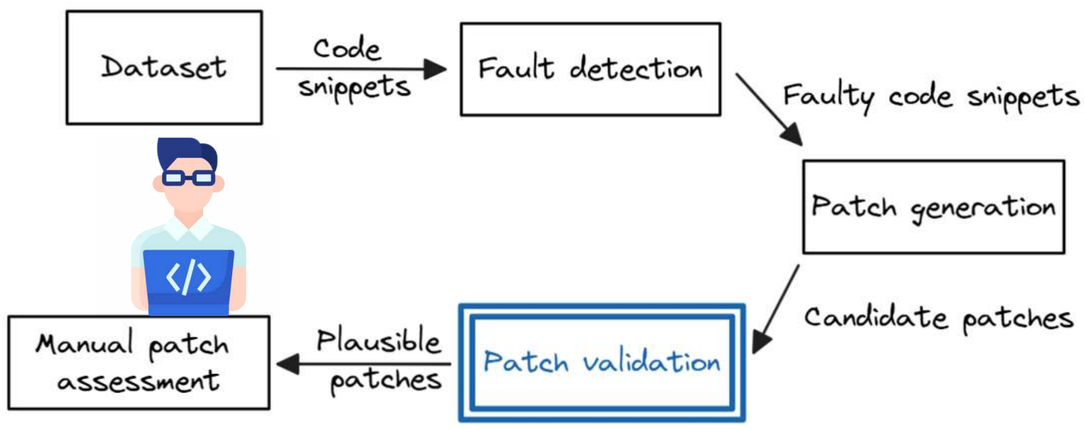
Using ML filters to help automated vulnerability repairs: when it helps and when it doesn't

ICSE 2025 - 47th IEEE/ACM International Conference on Software Engineering

Università di Trento, Dipartimento di Ingegneria e Scienza dell'Informazione

ICT DAYS

Automated Program Repair



Not all the patches produced by APR actually fix code faults.
We need tests to verify whether the functionality of the code is preserved, but they are not always available and they can be computationally expensive to run.

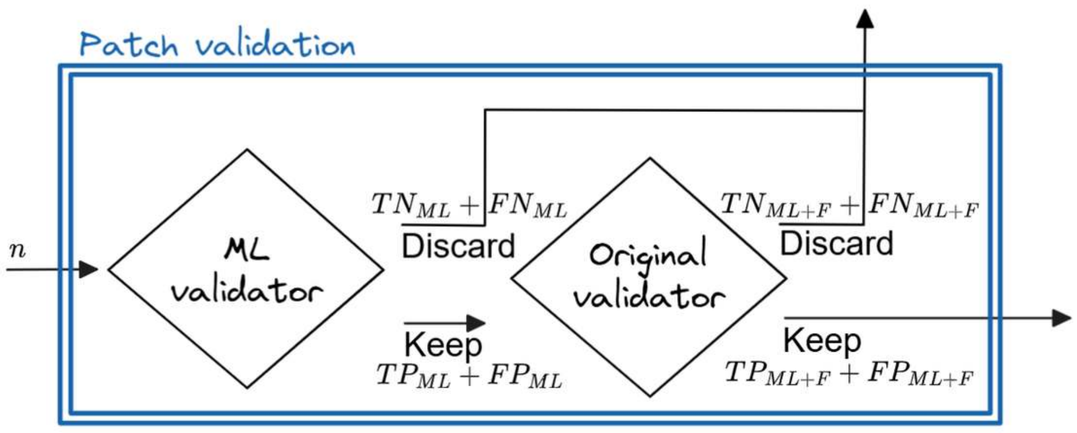
Università di Trento, Vrije Universiteit Amsterdam

H2022 Program Sec4AI4Sec (Grant n.101120393)

SERICS SECURITY AND RIGHTS IN THE CYBERSPACE

Project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU

Improving patch validation with Machine Learning



We investigate under which conditions an ML model can act as an effective pre-screener before a more expensive validation step.

The ML model should either

- improve the time efficiency of the validation process by quickly discarding most of the unpromising patches
- improve the number of correct patches found in a certain time frame by allowing to process more candidate patches in the same amount of time

REQUIREMENTS

For the ML to be an effective pre-screener:

- the number of correct patches at the end of the pipeline should increase
- the time to process the total number of candidates should decrease

BOUNDARIES

$$\frac{\Delta n}{n} \geq \frac{1}{R_M} - 1$$

The lower the model recall, the more candidate patches are required.

$$\tau_M \leq \tau_V \cdot R_M \cdot (1 - \frac{\pi}{P_M})$$

The lower the model precision and recall, the quicker the model should be.

Our process

1. We gathered the performance data of different vulnerability detection models in the literature.
2. We used the boundaries to compute how faster the ML models should be compared to the original testing time
3. We computed the time limit for the ML models to be effective pre-screeners for at least 75% of the projects in Vul4J, a dataset of Java vulnerabilities

Tools	Time limit (s)
VulDeePecker	5.23
VulDeePecker on ReVeal	4.70
IVDetect on ReVeal	4.95
LineVul	5.67
LineVD	4.85
CodeJIT FastRGCN	3.67
CodeJIT RGCN	3.87

Takeaway

Even the most effective model should take less than 5.67s (including pre-processing) to classify a patch to be an effective pre-validator

Future Works

1. Experimental evaluation
2. Considering ML models with different targets

MARIA CAMPORESE [1]
maria.camporese@unitn.it

FABIO MASSACCI [1,2]
fabio.massacci@ieee.org

(1) Università di Trento, Italy; (2) Vrije Universiteit Amsterdam, The Netherlands