APR4Vul: An Empirical Study of Automatic Program Repair Techniques on Real-world Java Vulnerabilities

Quang-Cuong (Cuong) Bui [1]

Ranindya Paramitha [2]

Duc-Ly Vu [3]

Fabio Massacci [2,4]

Riccardo Scandariato [1]

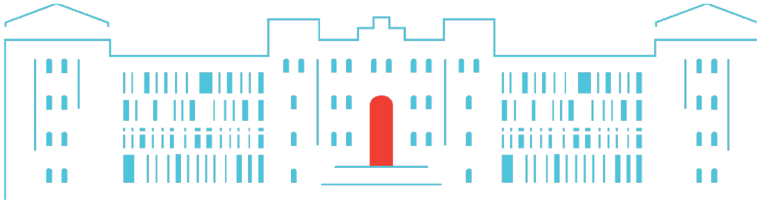
[1] Institute of Software Security, Hamburg University of Technology, Germany

[2] University of Trento, Italy

[3] University of Information Technology, Ho Chi Minh City, Vietnam

[4] Vrije Universiteit Amsterdam, Netherlands

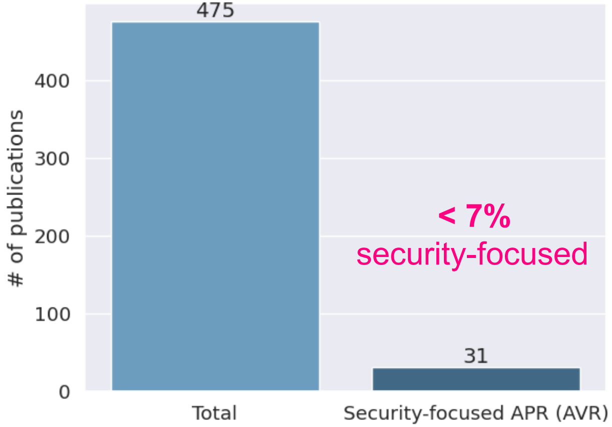
ICSE'24 - 17th April, 2024
Lisbon, Portugal



Automatic Vulnerability Repair (AVR) is still underexplored

- **Idea:** Explore the wealth of Automatic Program Repair (APR) to fix security bugs
- This work comes as a foundation study
  - Evaluate performance of traditional APRs on repairing vulnerabilities in the Vul4J dataset[1]
  - Analyze the differences between vulnerability patches (ExtraVul) and bug patches (Defects4J)

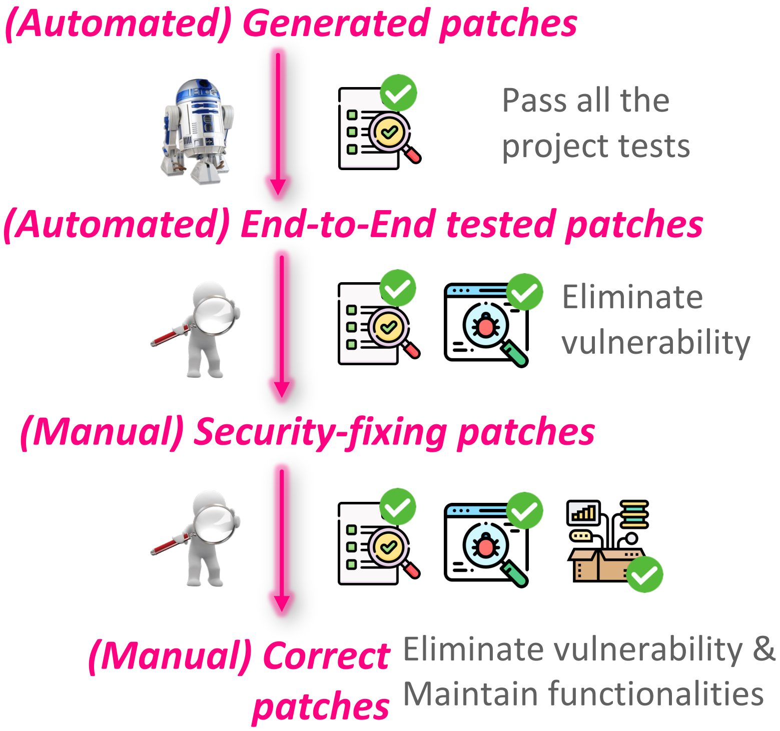
Source: program-repair.org



[1] Bui et al. *Vul4J: A Dataset of Reproducible Java Vulnerabilities Geared Towards the Study of Program Repair Techniques*. MSR'22.

Methodological challenges

- Issues with built-in test executors of APR tools on real-world projects
  - Customized Maven/Gradle cmds
  - Feed exact vulnerable locations
- Assessment of both the security and functional correctness of the patches
  - Carried out by three researchers with cross-validation



(Automated) Generated patches

Pass all the project tests

(Automated) End-to-End tested patches

Eliminate vulnerability

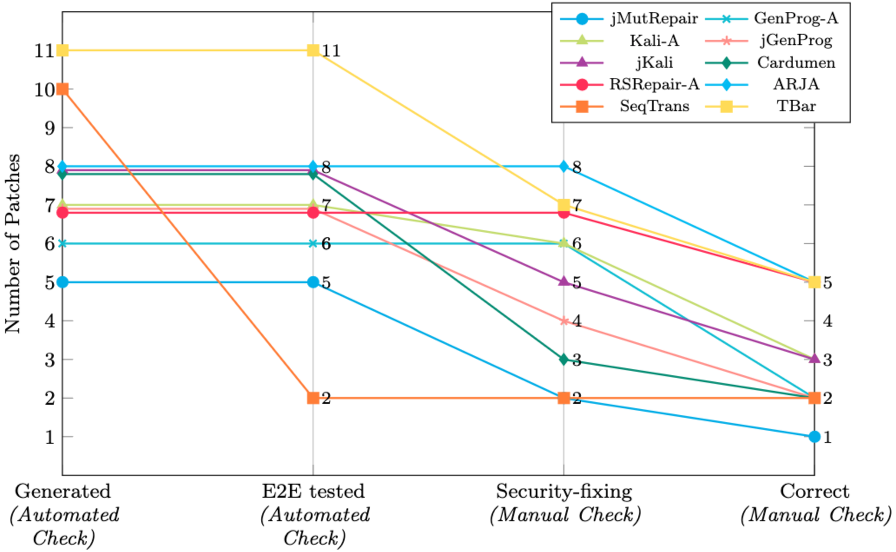
(Manual) Security-fixing patches

(Manual) Correct patches

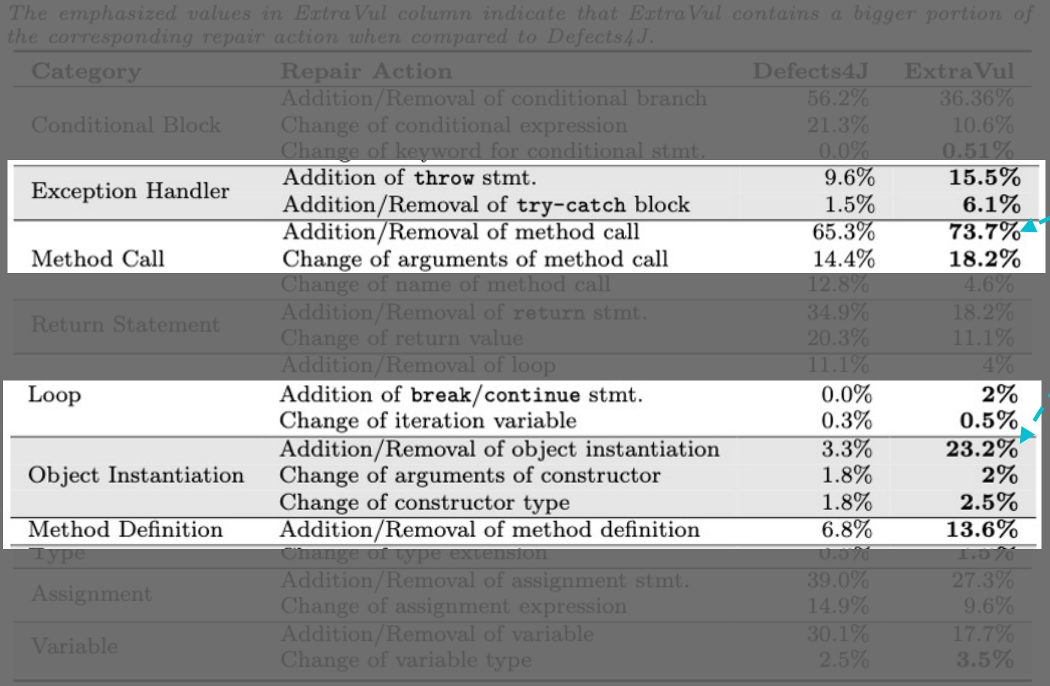
Eliminate vulnerability &
Maintain functionalities

Tools' performance

- Generate E2E tested patches for **only 20%** of vulnerabilities in Vul4J
- Best performers: ARJA, RSRepair-A, TBar
- On average, if an APR tool manages to generate *ten* E2E tested patches:
  - *three* are useless
  - *three* eliminate vulnerability yet break functionalities
  - *only four* can be used as-is



Repair actions



The emphasized values in ExtraVul column indicate that ExtraVul contains a bigger portion of the corresponding repair action when compared to Defects4J.

Category	Repair Action	Defects4J	ExtraVul
Conditional Block	Addition/Removal of conditional branch	56.2%	36.36%
	Change of conditional expression	21.3%	10.6%
	Change of keyword for conditional stmt.	0.0%	**0.51%**
Exception Handler	Addition of `throw` stmt.	9.6%	**15.5%**
	Addition/Removal of `try-catch` block	1.5%	**6.1%**
Method Call	Addition/Removal of method call	65.3%	**73.7%**
	Change of arguments of method call	14.4%	**18.2%**
	Change of name of method call	12.8%	4.6%
Return Statement	Addition/Removal of `return` stmt.	34.9%	18.2%
	Change of return value	20.3%	11.1%
Loop	Addition/Removal of loop	11.1%	4%
	Addition of `break/continue` stmt.	0.0%	**2%**
	Change of iteration variable	0.3%	**0.5%**
Object Instantiation	Addition/Removal of object instantiation	3.3%	**23.2%**
	Change of arguments of constructor	1.8%	**2%**
	Change of constructor type	1.8%	**2.5%**
Method Definition	Addition/Removal of method definition	6.8%	**13.6%**
Type	Change of type extension	0.5%	**1.5%**
Assignment	Addition/Removal of assignment stmt.	39.0%	27.3%
	Change of assignment expression	14.9%	9.6%
Variable	Addition/Removal of variable	30.1%	17.7%
	Change of variable type	2.5%	**3.5%**

Explain: APRs adding code hit more correct patches e.g., ARJA, RSRepair

**More often
in vulnerability patches**

Repair patterns

MC = Method Call

Group	Repair Patterns
Infinite Loop Handling	- Add `break/continue/throw` to exit loop - Update loop header
Secure Object Instantiation	- Use secure constructor, e.g., SecureRandom - Avoid deserialization of vulnerable class
User's Permission Management	- Add MC to check permission of executing user - Add MC to restrict user's permission
Secure Configuration	- Add MC to enable/disable secure/insecure features of XML parsers
External Input Validating and Handling	- Add MC to sanitize input - Add If-condition + `throw/return` to reject invalid input/state - Update Regular Expression for validating input - Add '/' to system path or URI path to prevent Path Traversal
Others	- Remove code to avoid sensitive data/API exposure - Move code

Most of the repair patterns do not exist in general bug patches

Most frequently used repair patterns

Fix ~34% of the vulnerabilities in the dataset

Key takeaways

/tuhh-softsec/APR4Vul

- Traditional APR tools have poor performance in fixing vulnerabilities → do not use them as-is
- New repair patterns enable the ability to fix vulnerabilities → improve the APR tools

Published in the **Journal of Empirical Software Engineering**