Better Safe Than Sorry! Automated Identification of Functionality-Breaking Security-Configuration Rules

Patrick Stöckle
patrick.stoeckle@siemens.com
Siemens Technology &
Technical University of Munich (TUM)

Michael Sammereier
michael.sammereier@tum.de
TUM

Bernd Grobauer
bernd.grobauer@siemens.com
Siemens Technology

Alexander Pretschner
alexander.pretschner@tum.de
TUM

4th ACM/IEEE International Conference on Automation of Software Test (AST 2023)
15.05.2023
Melbourne, AUS 🇦🇺

SIEMENS

Motivation

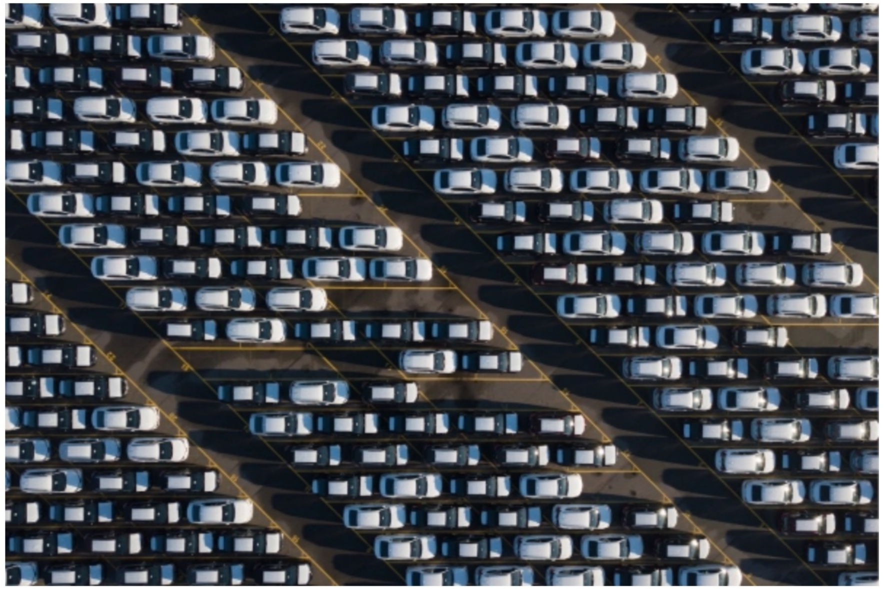
SIEMENS

Recent Motivating Example

… about 2.15 million customers whose personal and vehicle information were left exposed to the internet after a "cloud misconfiguration" …

Toyota Japan exposed millions of vehicles' location data for a decade

Zack Whittaker @zackwhittaker / 3 days



https://techcrunch.com/2023/05/12/toyota-japan-exposed-millions-locations-videos/

Background: Configuration Hardening

Problem 1: Consumer software is not configured securely by default

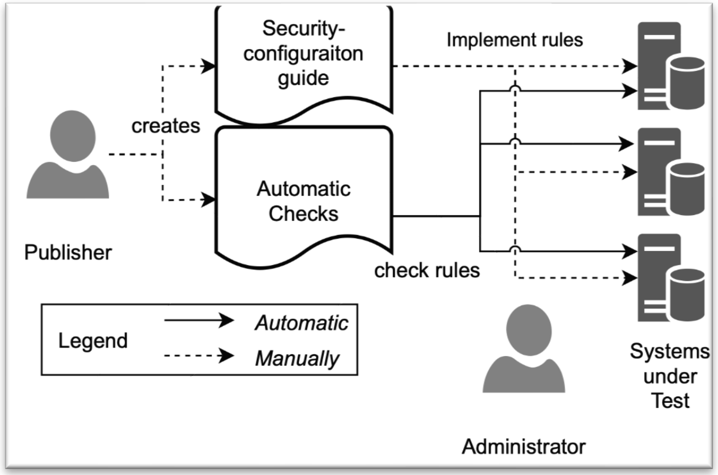
- Function and usability more critical than security
- Conflicting interests, e.g., telemetry

Solution: We must configure them securely!

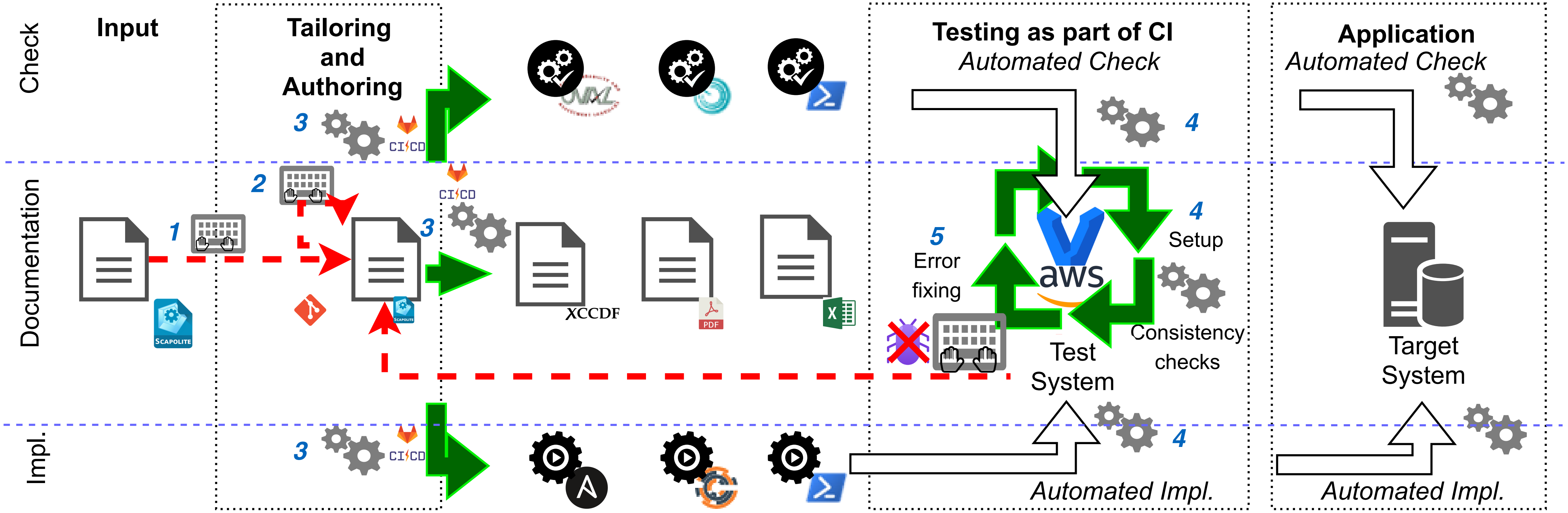
Problem 2: We want to configure them securely, but we do not know all the settings

Solution: Implementation of public security-configuration guides, e.g., from

- Center for Internet Security
- Defense Information Systems Agency

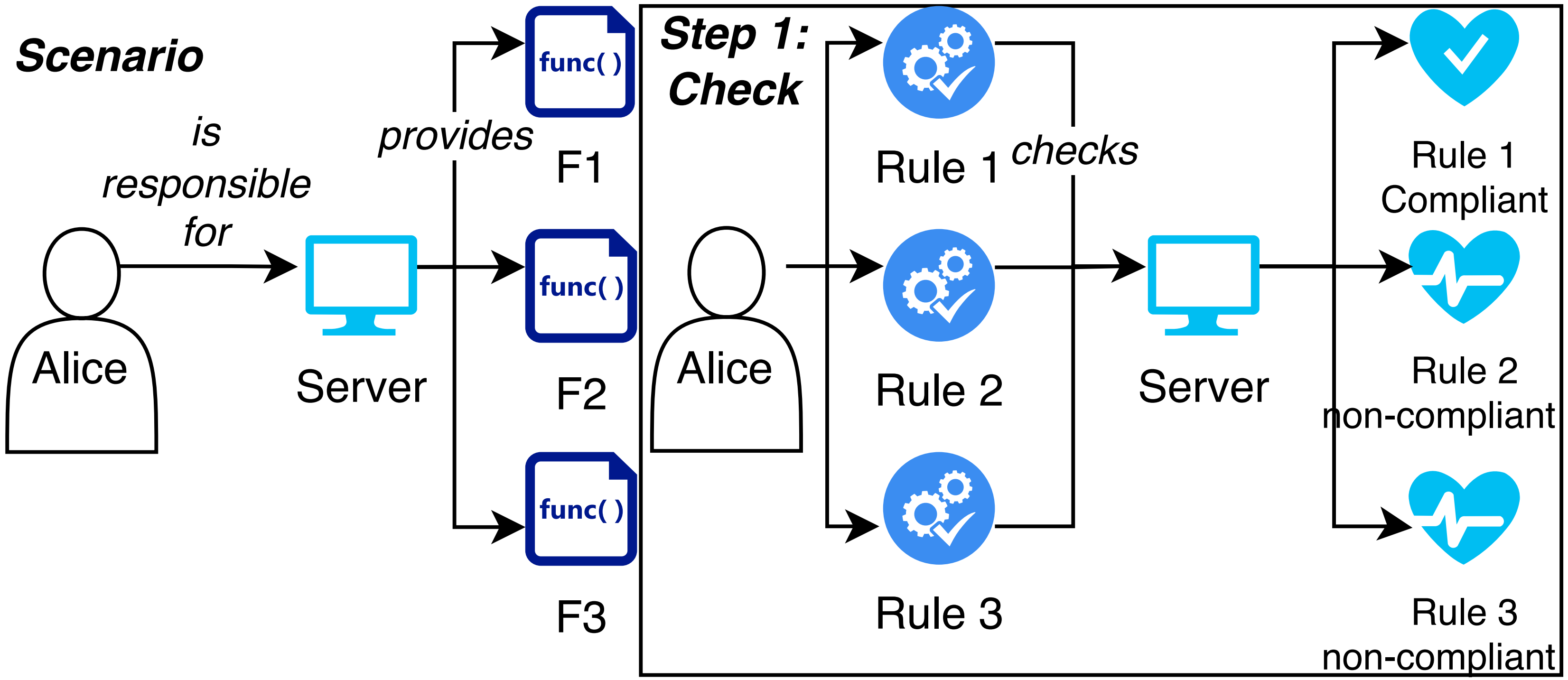


Automated Hardening Process at Siemens

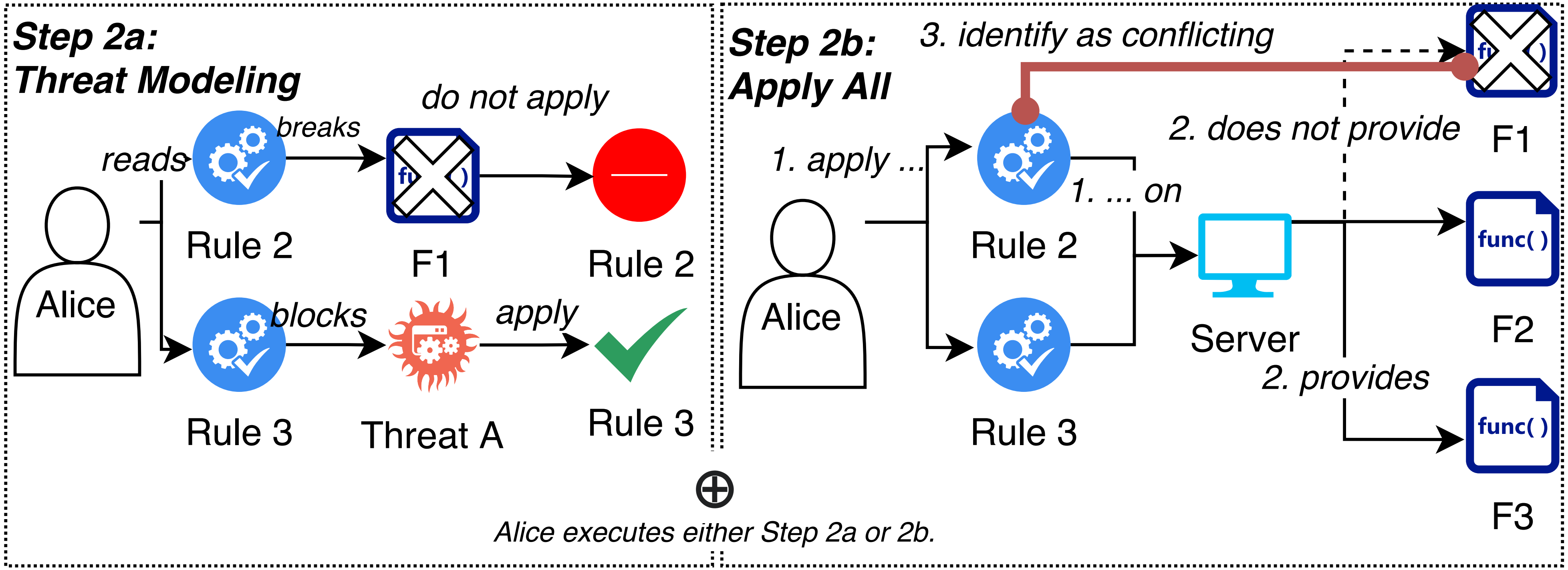


SIEMENS

Unsolved Problem



Unsolved Problem



SIEMENS

Unsolved Problem

Both approaches are **time-consuming** and, therefore, **expensive**

Threat analysis needs **security experts**

Alice normally doesn't know how **many rules** she must exclude and how the rules work together

↓

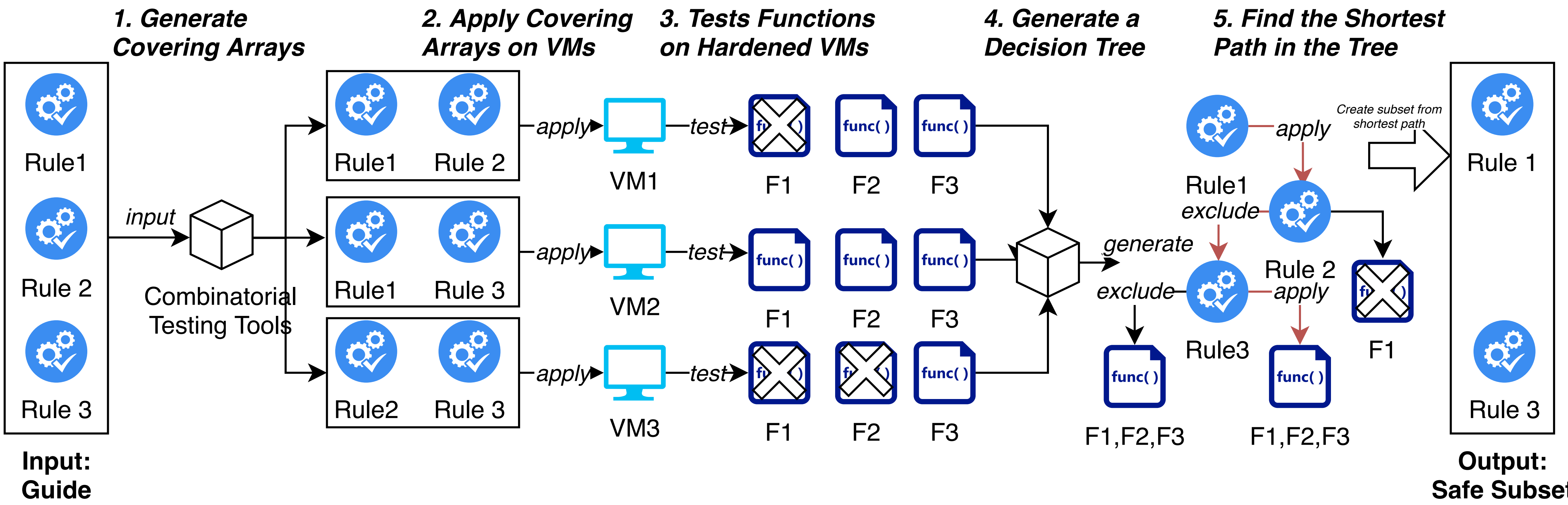
Alice will **only** implement **rules** where she is 100% sure that the rules will **not break any function**

... or generally **refrain** from hardening running systems if it is not forced to do so

Solution

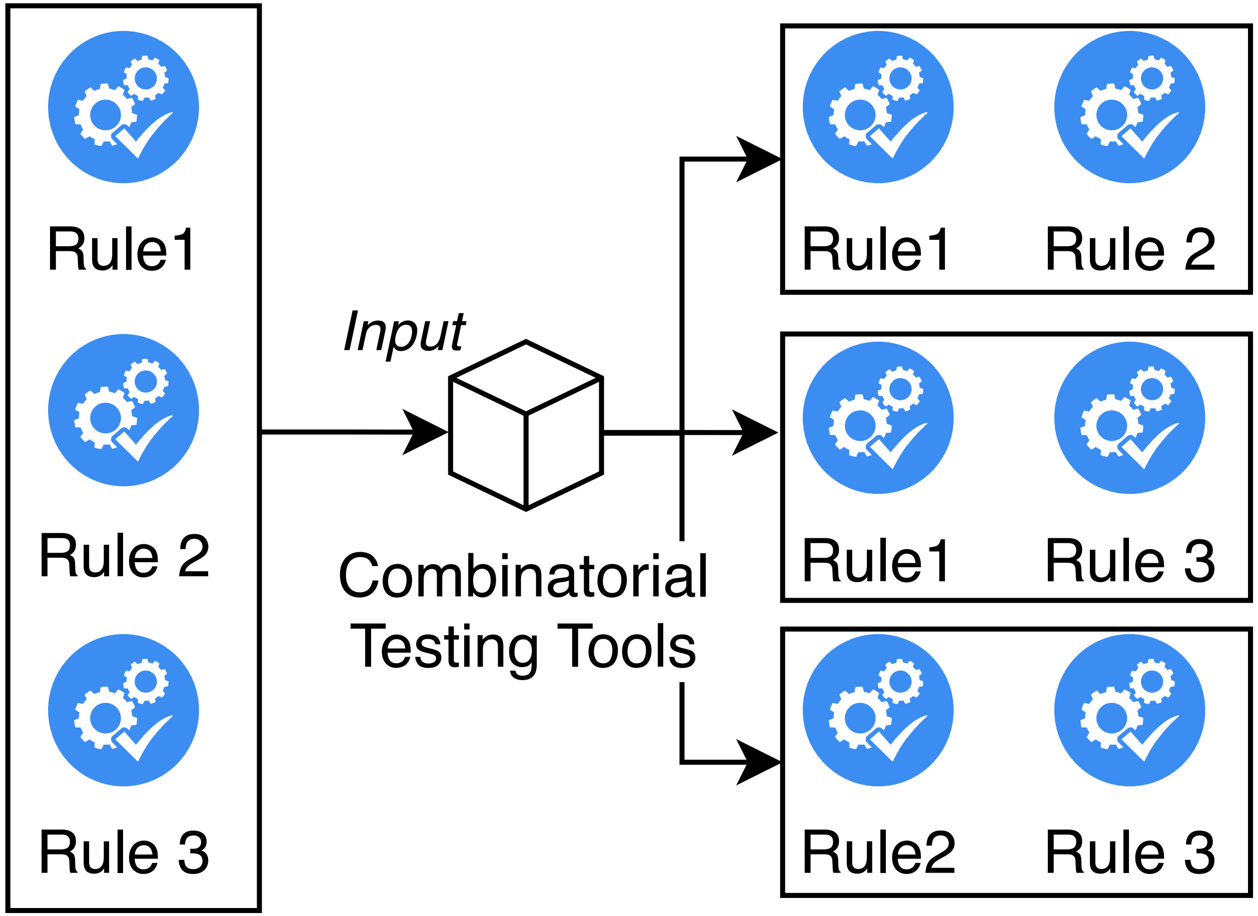
SIEMENS

Proposed Solution



SIEMENS

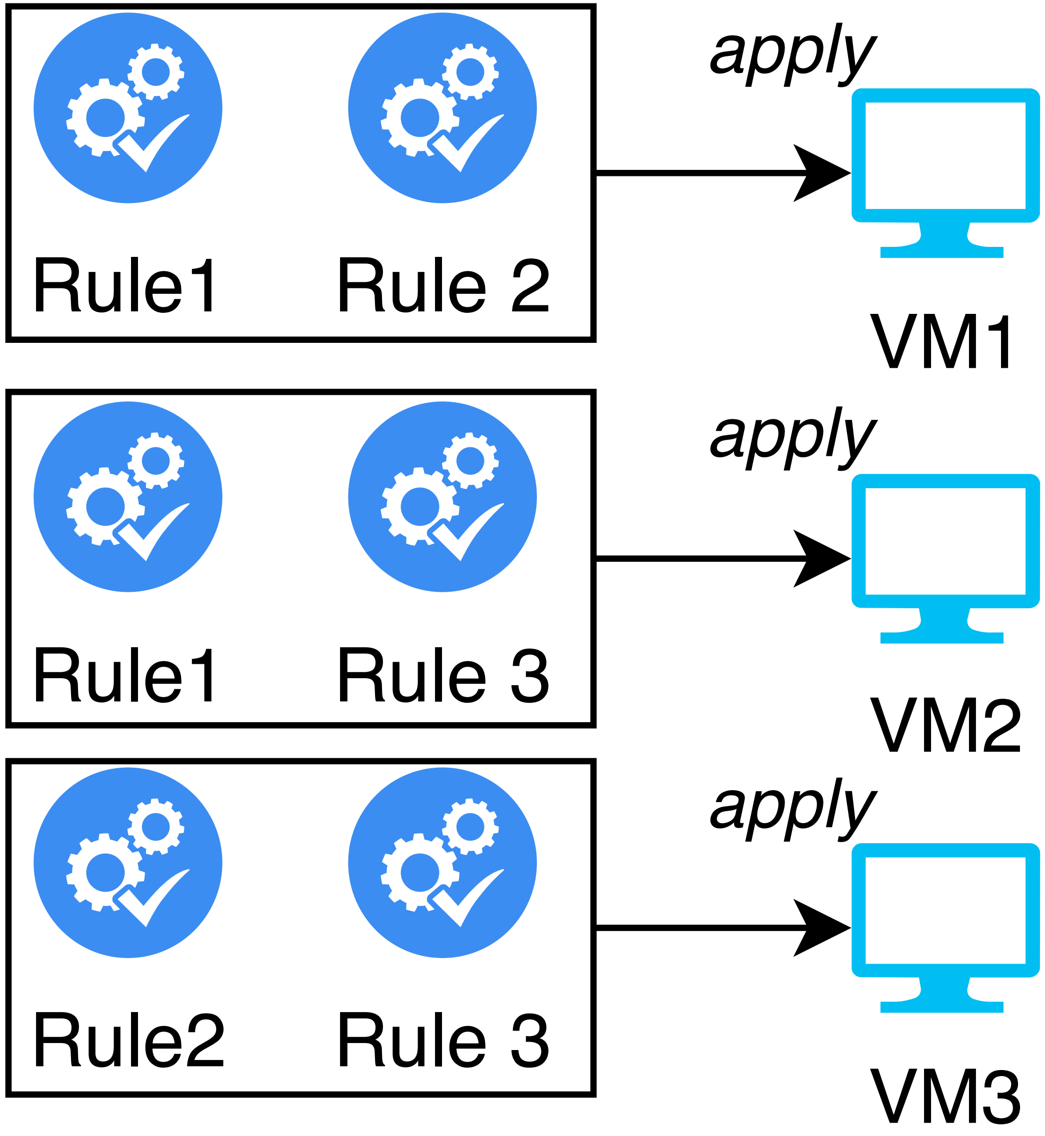
Generate Covering Arrays



Input:
Guide

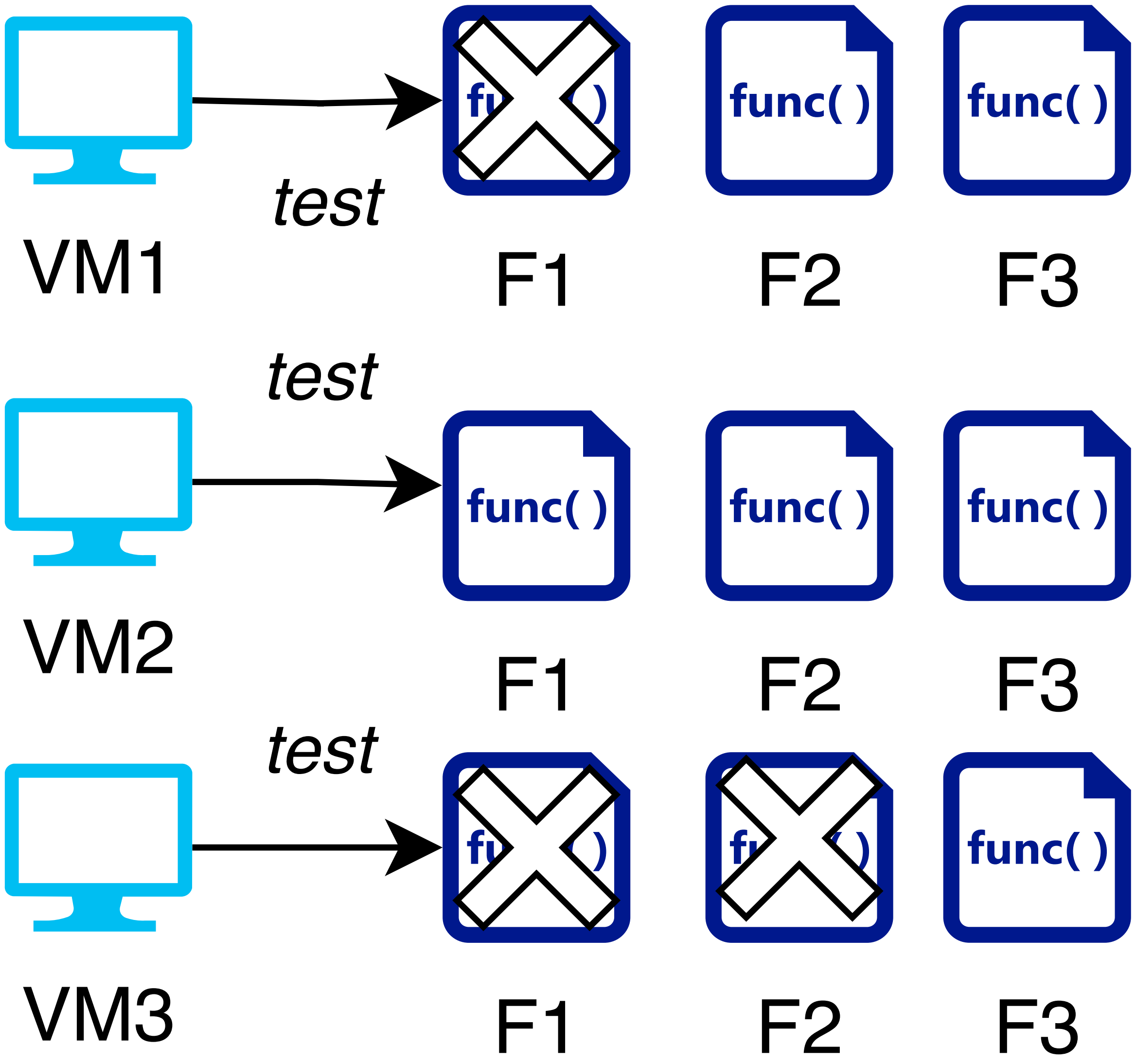
SIEMENS

Apply Covering Arrays on VMs

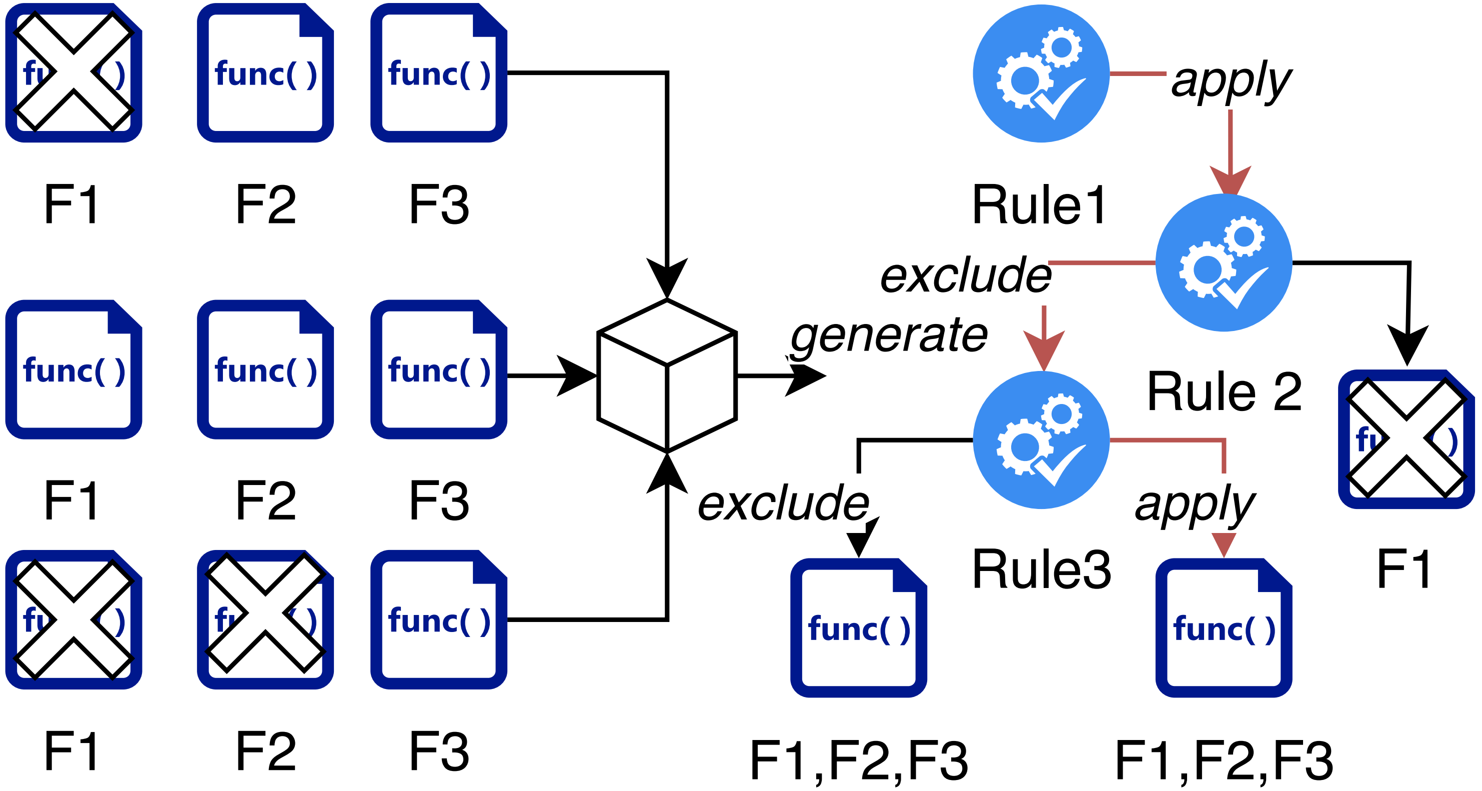


SIEMENS

Test Functions on Hardened VMs

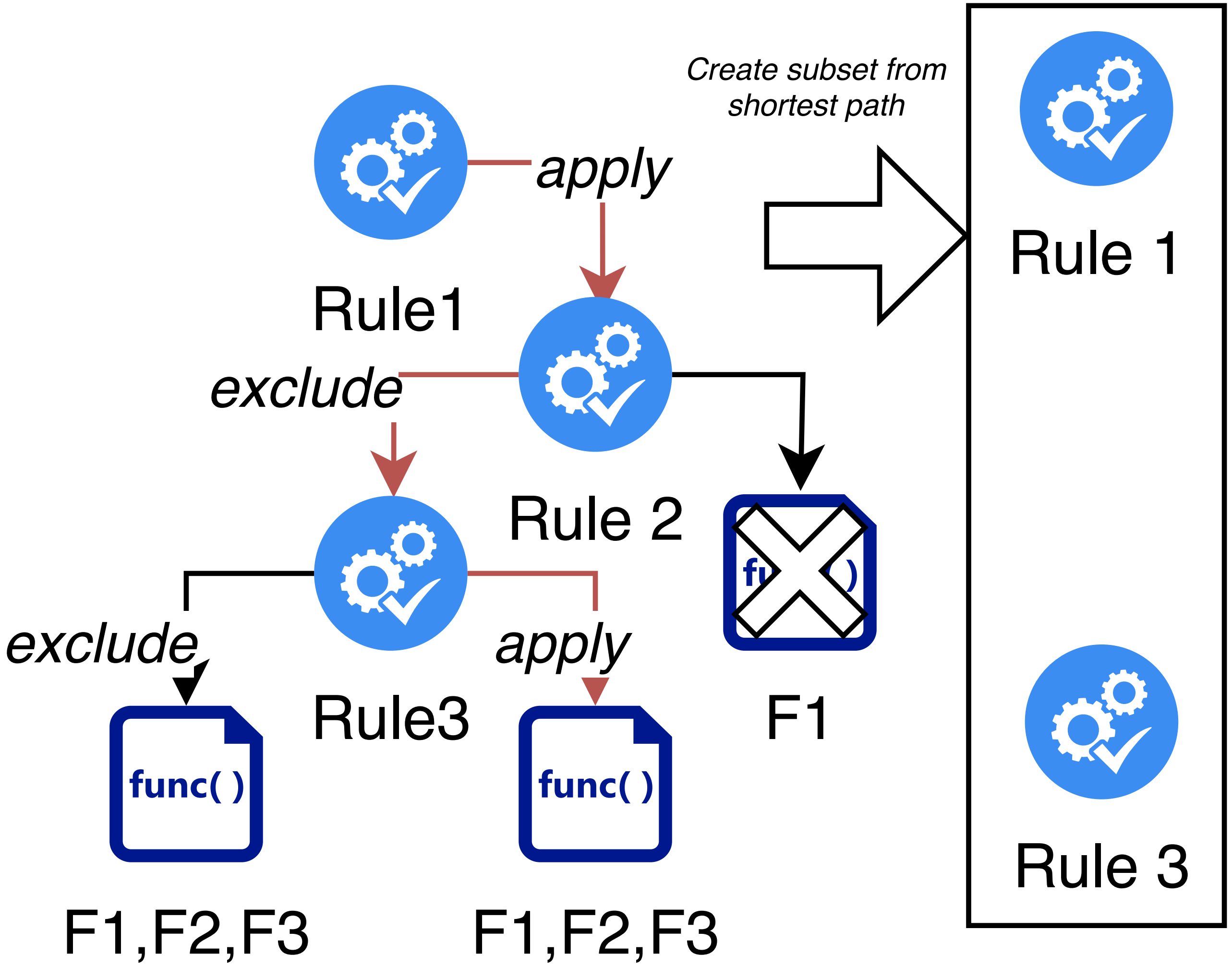


Generate a Decision Tree



SIEMENS

Find the Shortest Path in the Tree



Output: Safe Subset

Code

github/tum-i4/Better-Safe-Than-Sorry

SIEMENS

Evaluation

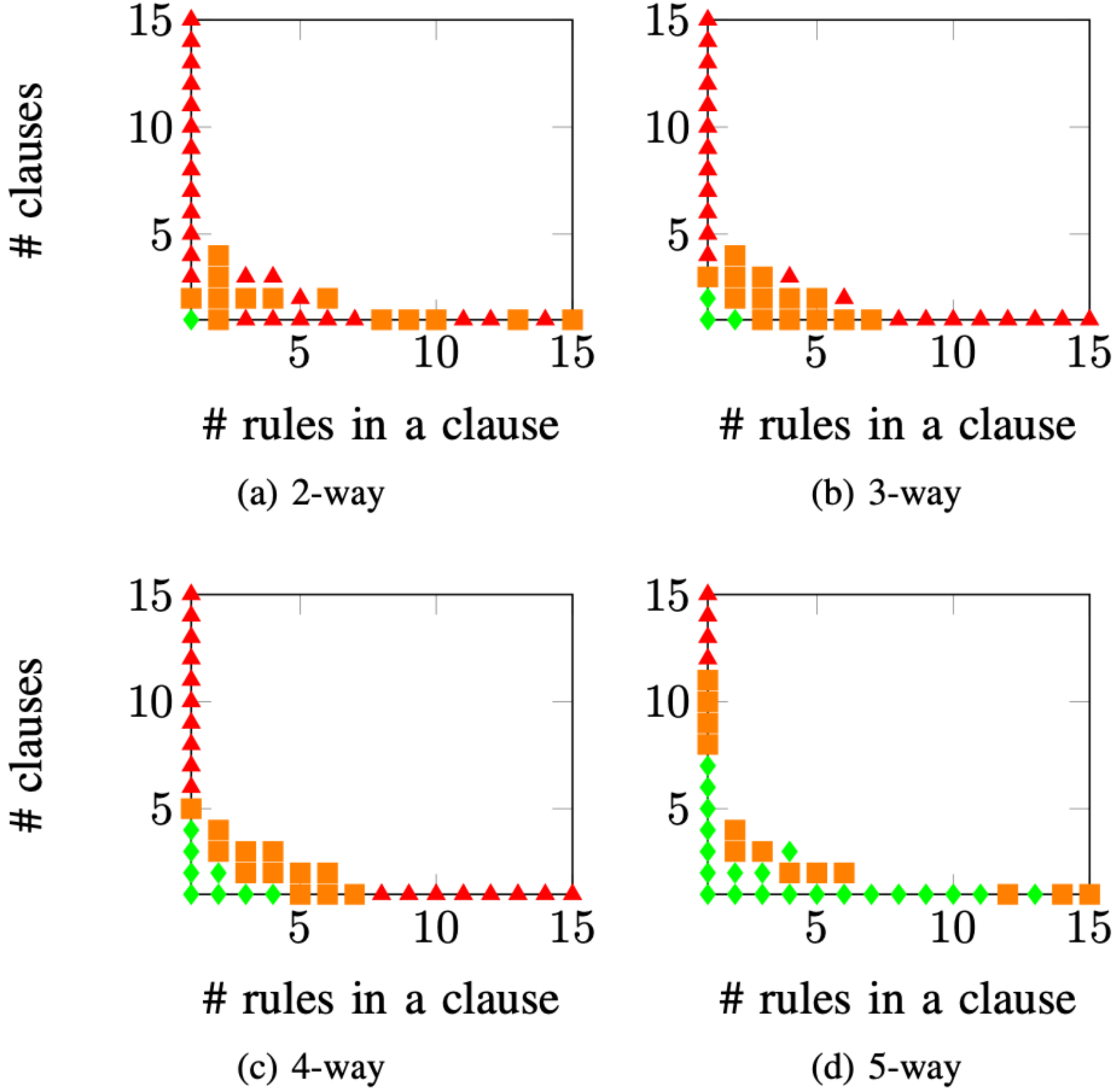
SIEMENS

Generation Time

Algorithm \ Strength	2	3	4	5
IPOG	0.7	374	179149	-
IPOG-D	0.2	16	8451	1184478

SIEMENS

Distribution of the Breaking Rules Sets



Results

Correctness

- Overall, the approach finds an optimal solution for 77% of our examples
- In the *realistic* examples, the approach finds an optimal solution in almost 100%

Time exposure

- Time for generation of covering arrays depends on strength ⇒ Under 1s for strength 2 and 14 days for strength 5
- Time required for generation is only justifiable for covering arrays up to a strength of 4
- Time required to execute the tests at strength 4 over 12h ⇒ Possible, but more suitable for integration tests

Threats to Validity

Tests

- We need **automated tests** ⇒ especially with complex systems or graphical user interfaces, tests are often still carried out manually
- We need **good** tests. If the tests succeed, but productive systems fail, we must revert all rules
- We need **fast** tests because we must run the tests once for each covering array ⇒ complex systems in particular need many and complex tests that take time

Setup

- We need systems that we can set up, install and test **automatically** ⇒ difficult or even impossible, especially with complex and heterogeneous systems

Missing data about configuration issues

- Since few organizations harden the configuration of their systems, we have **no empirical information** on problematic combinations and instead must use data from other domains

Conclusion

Conclusion

More tests!

- We need more tests and automated tests to be able to implement hardening measures on running systems without side effects!

A/B testing?

- If we don't have these automated tests, the covering arrays could also be tested as part of A/B testing: one covering array at a time is applied to a subset.
- As soon as a problem occurs, the rules are directly reset, and the covering array is marked as breaking. When all arrays have been tested, we can determine the unproblematic subset

Hardening right from the beginning!

- The hardening of running systems is and remains time-consuming. Therefore, the systems should be hardened as soon as possible so that problems can be analyzed and solved directly

SIEMENS

Contact

Patrick Stöckle
Research Scientist
T CST SEA-DE
Siemens Technology

patrick.stoeckle@siemens.com

Twitter	@p_stoeckle
LinkedIn	patrick-stoeckle
GitHub	pstoeckle

SIEMENS