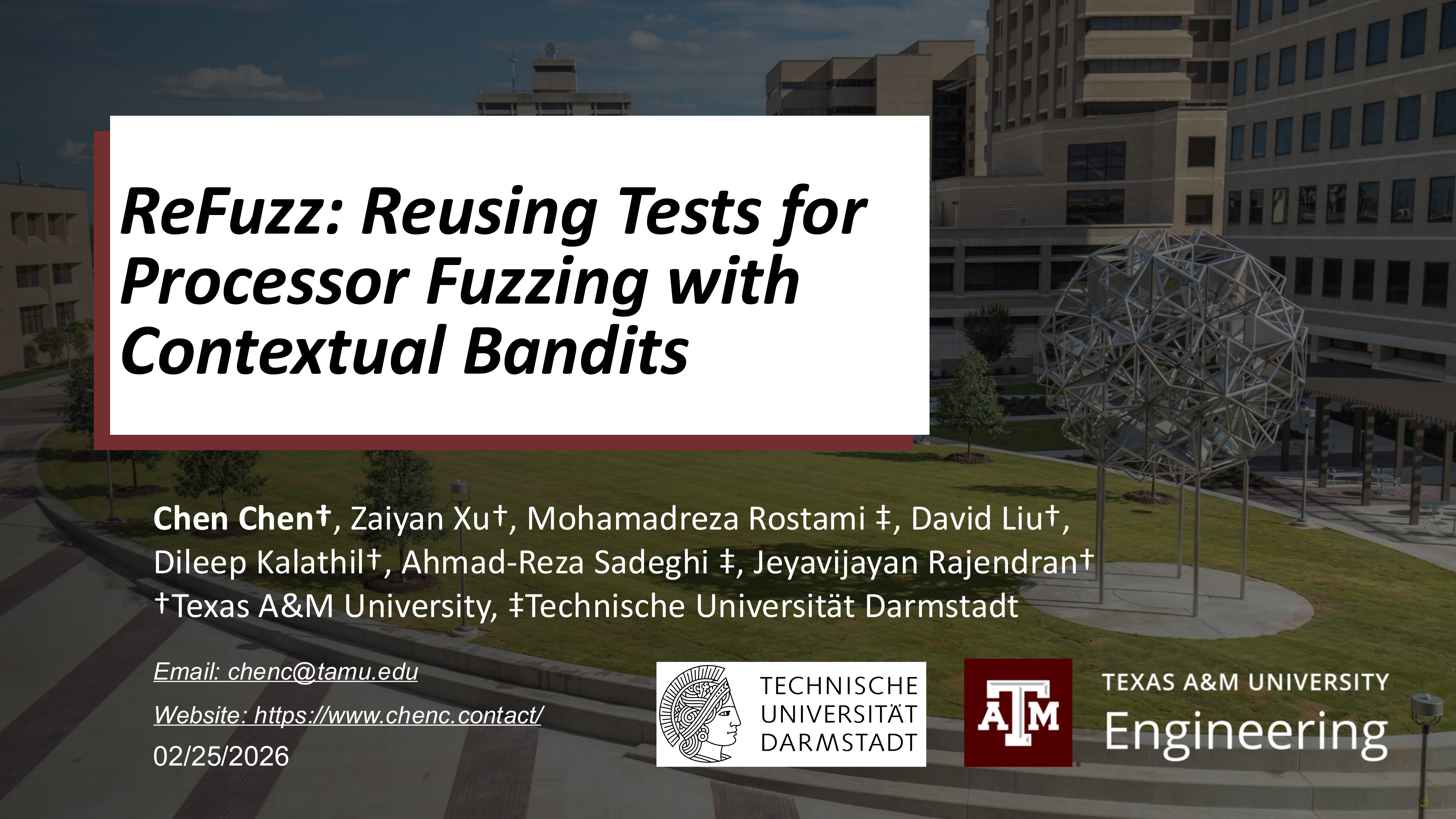


ReFuzz: Reusing Tests for Processor Fuzzing with Contextual Bandits

Chen Chen†, Zaiyan Xu†, Mohamadreza Rostami ‡, David Liu†,
Dileep Kalathil†, Ahmad-Reza Sadeghi ‡, Jeyavijayan Rajendran†
†Texas A&M University, ‡Technische Universität Darmstadt

Email: chenc@tamu.edu

Website: https://www.chenc.contact/

02/25/2026

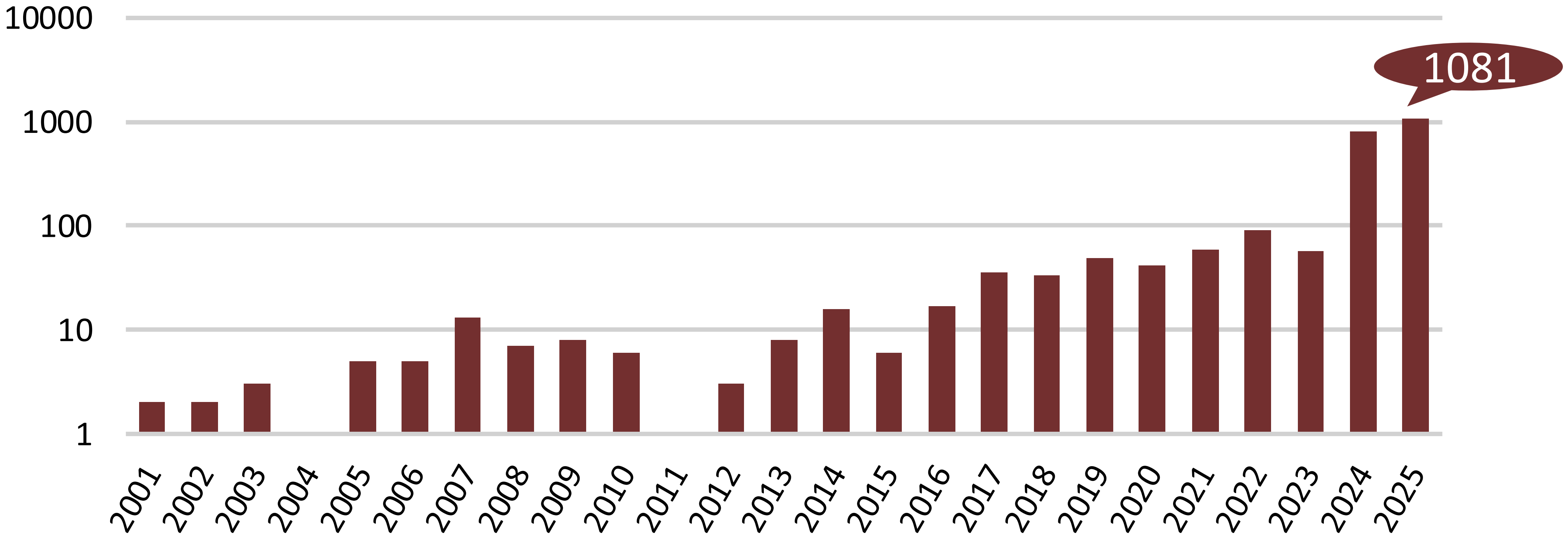
TECHNISCHE UNIVERSITÄT DARMSTADT

TEXAS A&M UNIVERSITY Engineering

Hardware Security: a New Bottleneck

Massive growth of hardware vulnerabilities

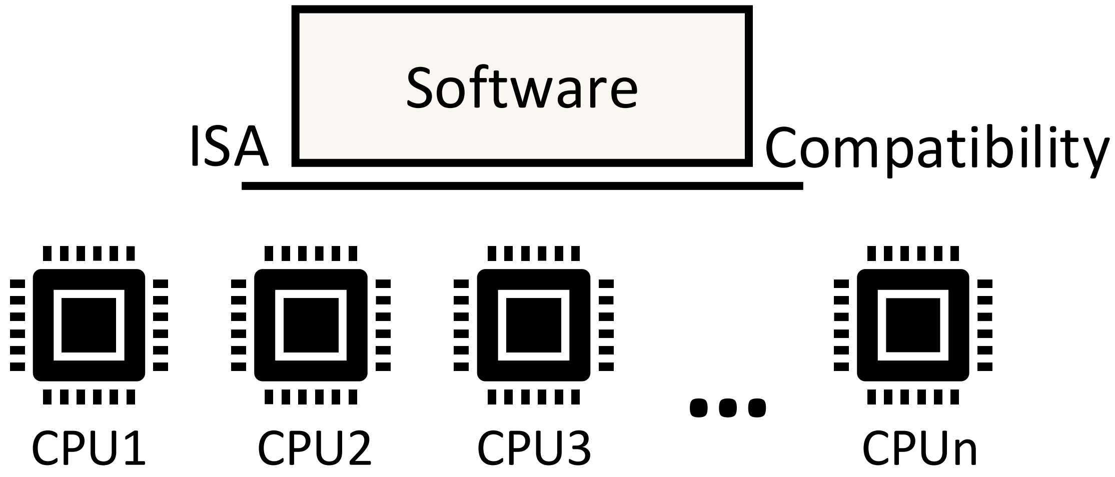
Total documented hardware vulnerabilities (CVEs) by year



Source: National Vulnerability Database NVD (09/2025)

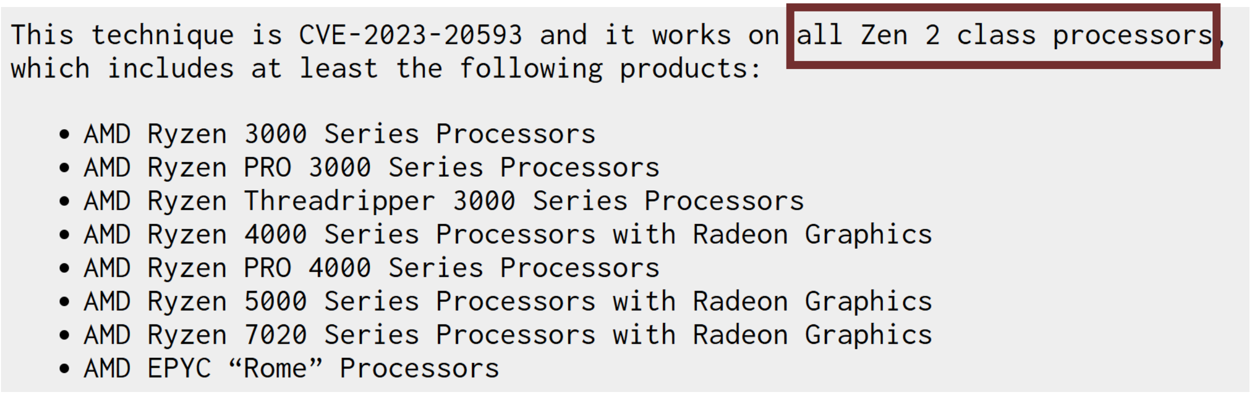
https://nvd.nist.gov/

Design Reuse is Typical in Hardware



Processor designs:

- Compatible with software
- Less modifications on functionalities and I/O spaces
- New microarchitectural features



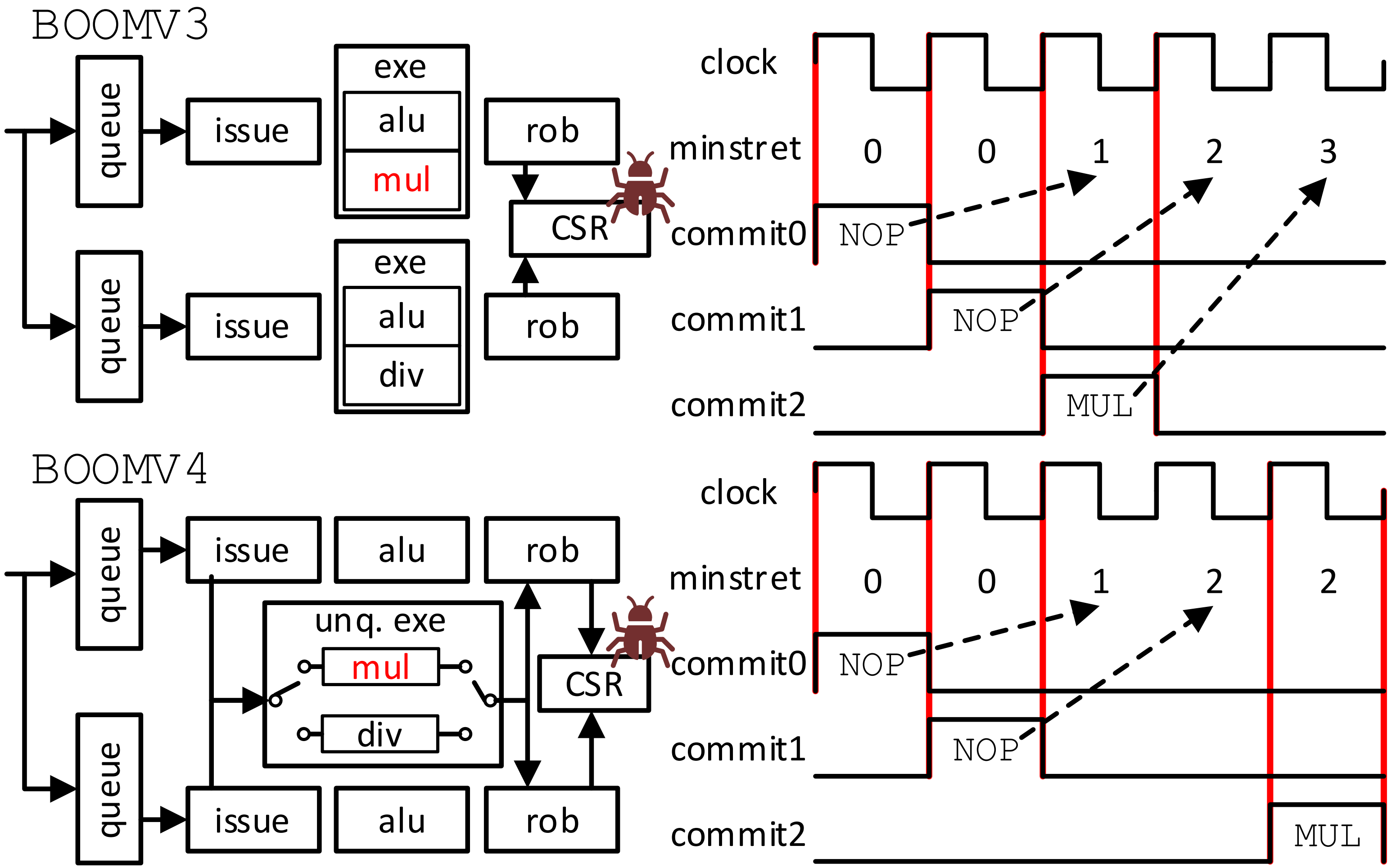
```
This technique is CVE-2023-20593 and it works on all Zen 2 class processors,
which includes at least the following products:

   • AMD Ryzen 3000 Series Processors
   • AMD Ryzen PRO 3000 Series Processors
   • AMD Ryzen Threadripper 3000 Series Processors
   • AMD Ryzen 4000 Series Processors with Radeon Graphics
   • AMD Ryzen PRO 4000 Series Processors
   • AMD Ryzen 5000 Series Processors with Radeon Graphics
   • AMD Ryzen 7020 Series Processors with Radeon Graphics
   • AMD EPYC “Rome” Processors
```

- IP reuse accounts for **67%**
- Develop new processors based on prior ones
- Vulnerabilities propagate to designs
- **Reuse tests across generations**

[1] https://www.icmanage.com/design-data-management-worldwide-trends-survey-report
[2] https://lock.cmpxchg8b.com/zenbleed.html

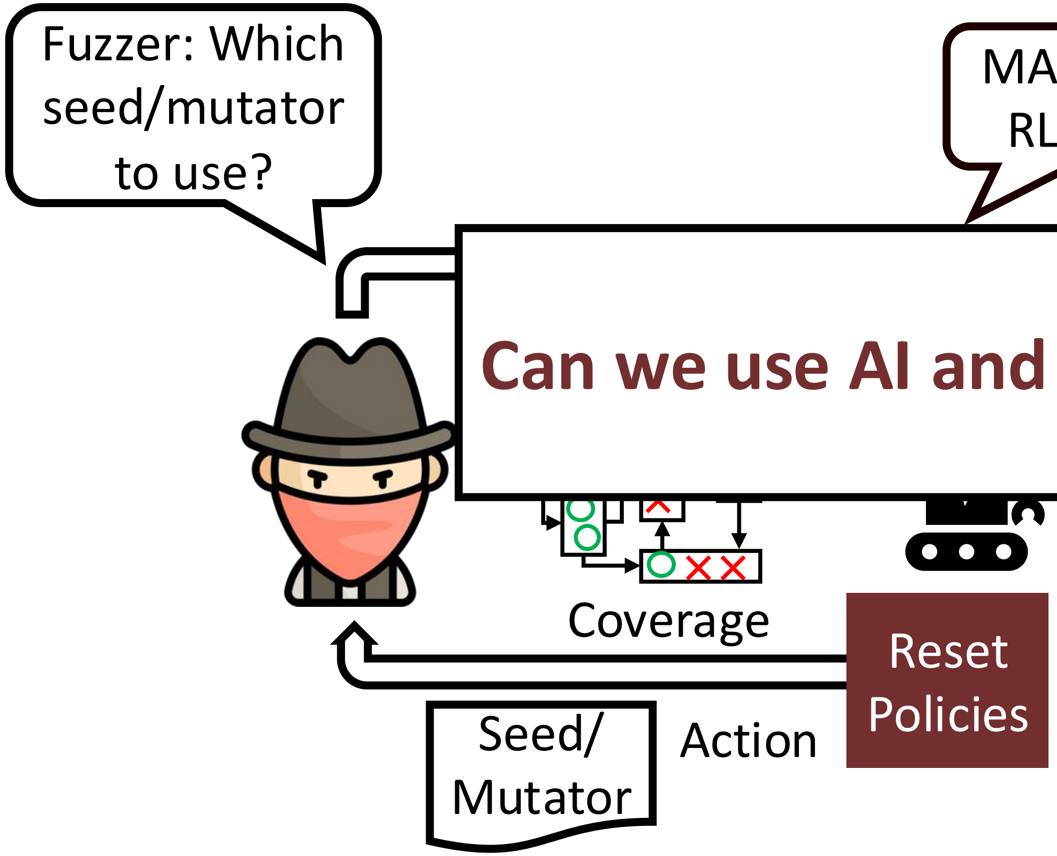
Vulnerability Across Processors

- Shared root cause: **two-cycle** delay to update the `minstret` register after commit
- `BOOMV4` has variants due to the `Unique Execution Unit`

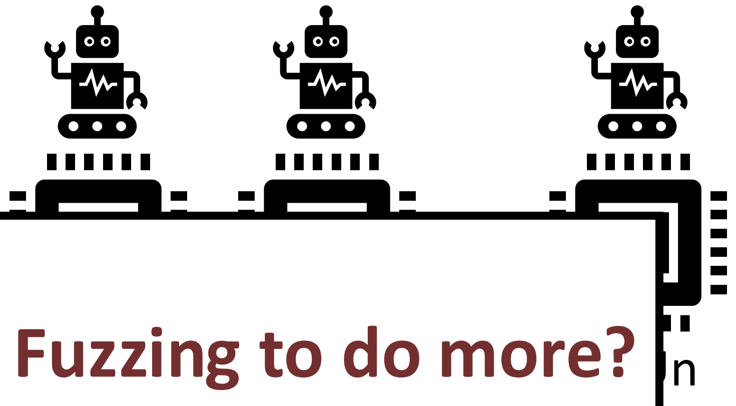


AI for Hardware Fuzzing

AI for searching

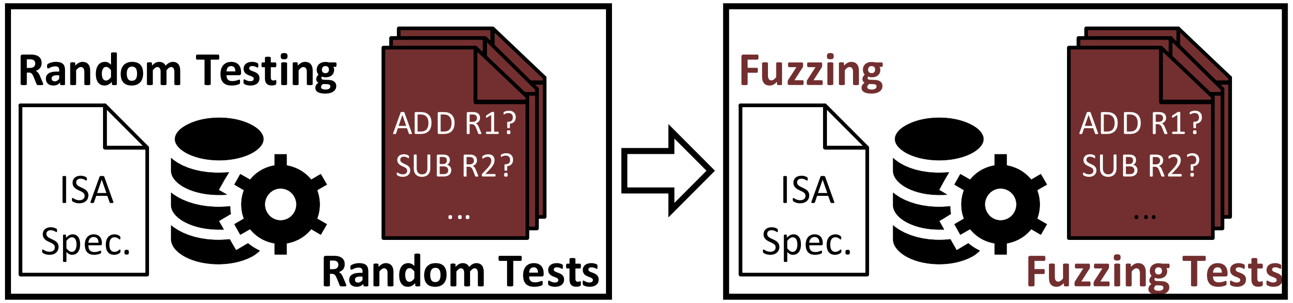


Deploy an AI agent on each PUT



Can we use AI and Hardware Fuzzing to do more?

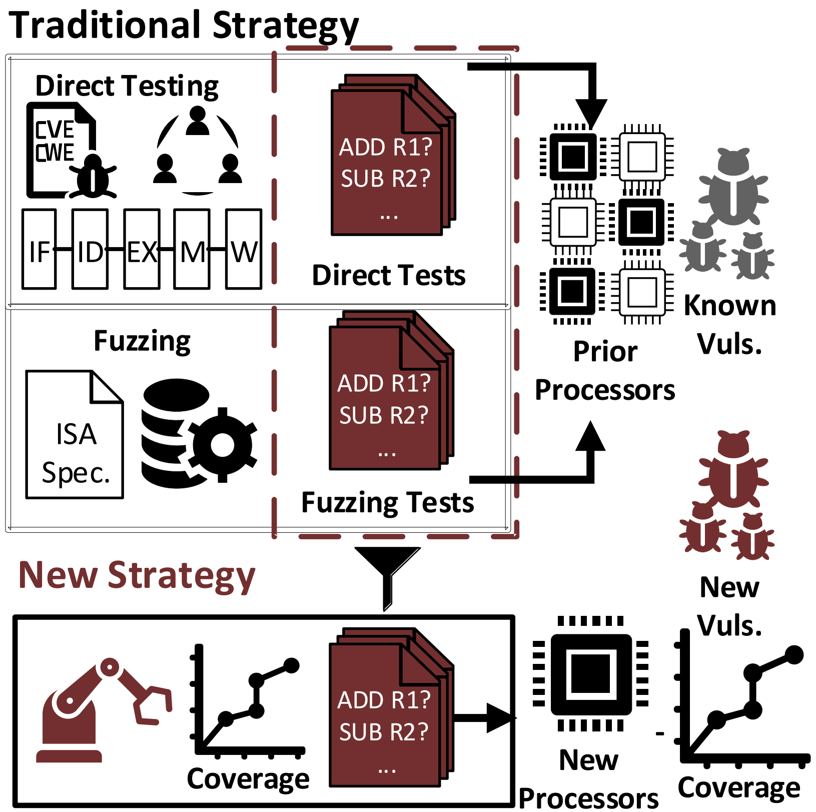
ndom testing

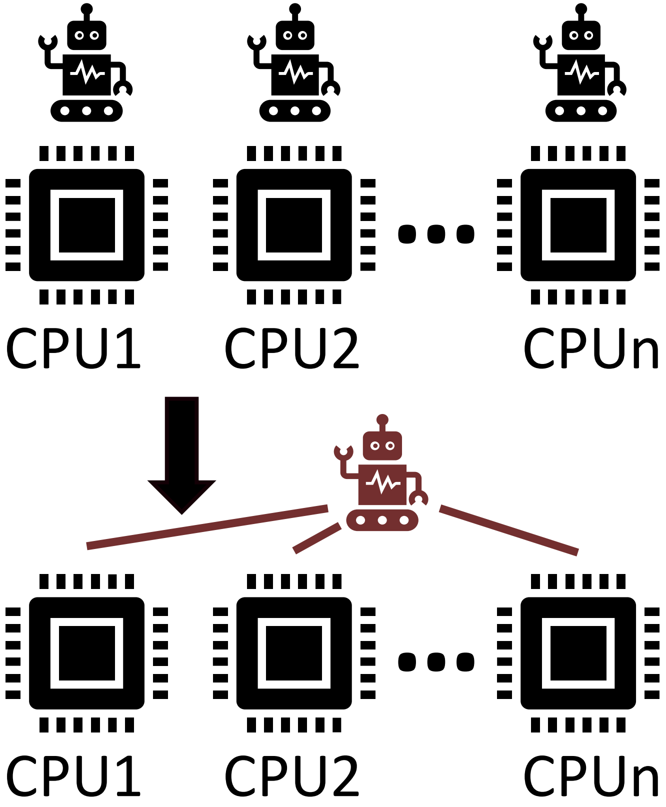


Enhance Fuzzing by Reusing Tests

Q1: Reuse effective tests from prior processors

Q2: Use AI as learning approaches instead of searching

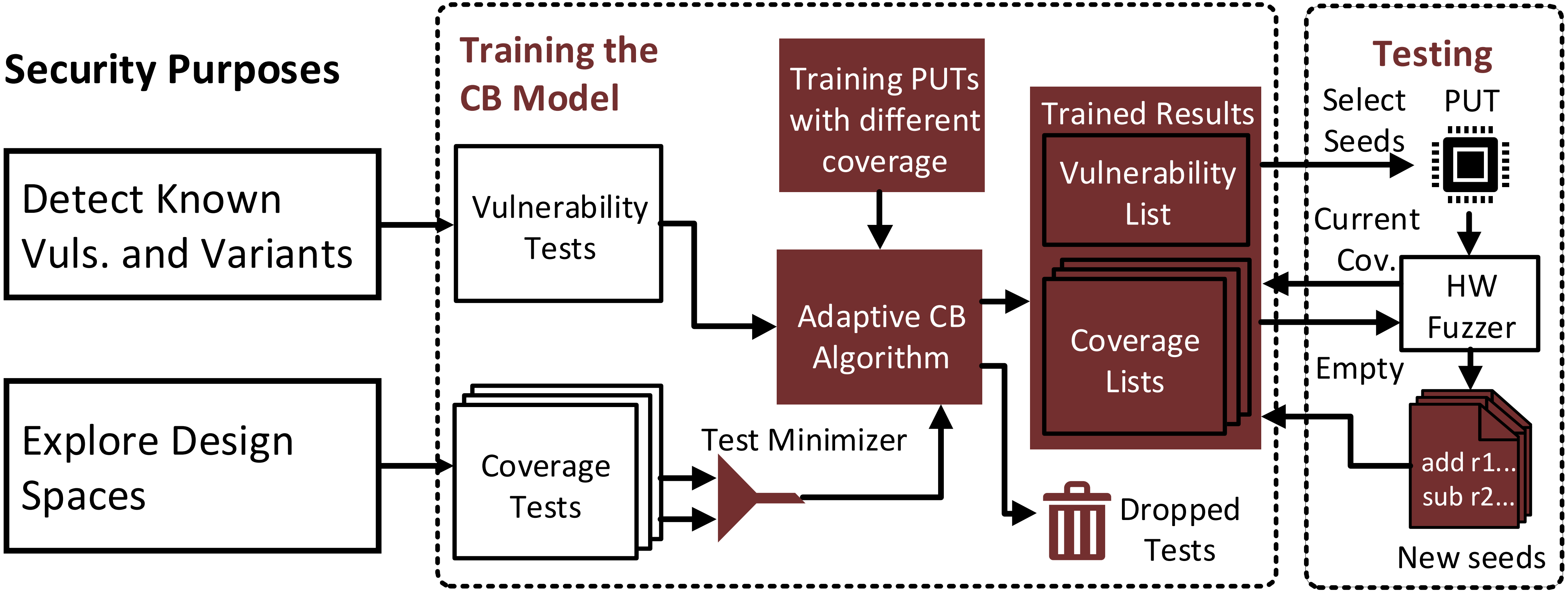




ReFuzz: 1st Test Reuse Fuzzing

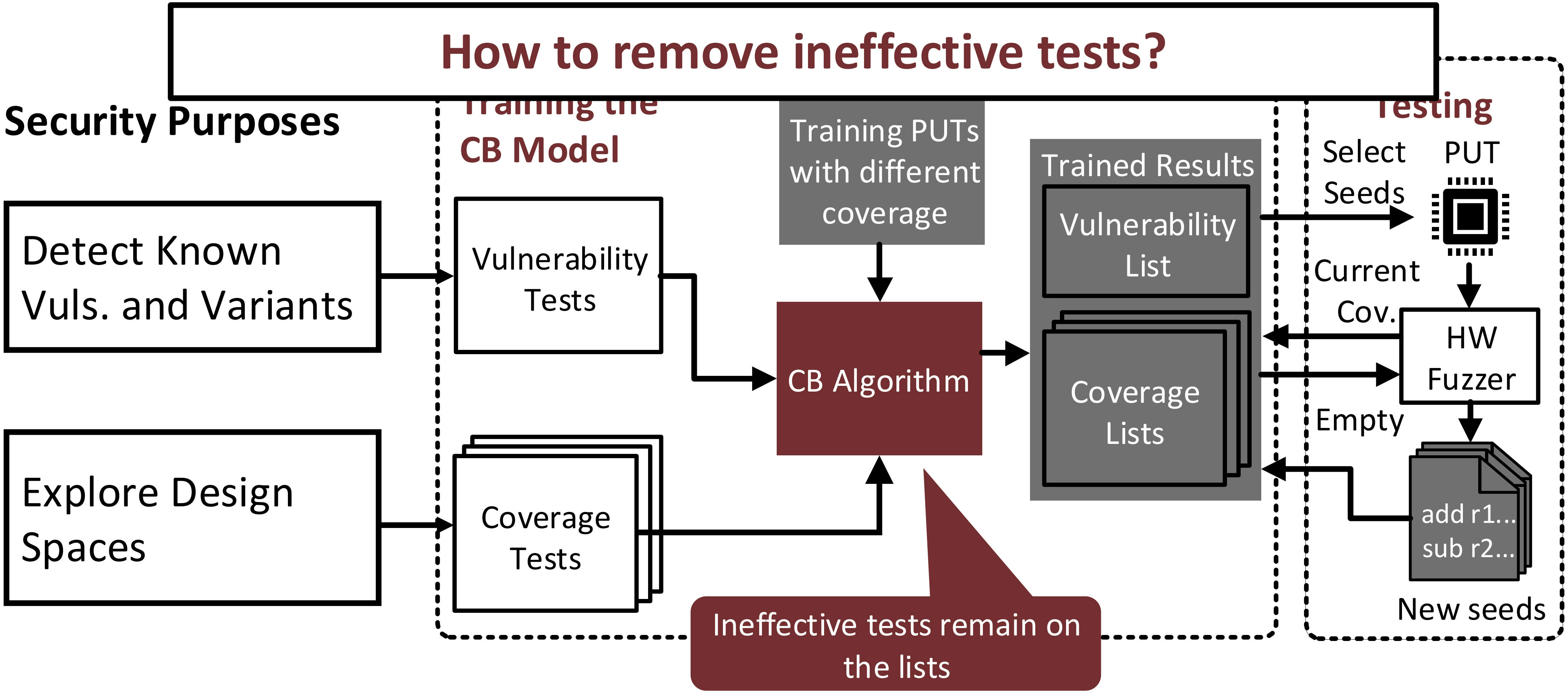
Contextual Bandit (CB) + Fuzzing

- CB: Identify and select effective tests
- Fuzzing: Generate test variants



Challenge 1: Ineffective Tests Remain on Trained Results

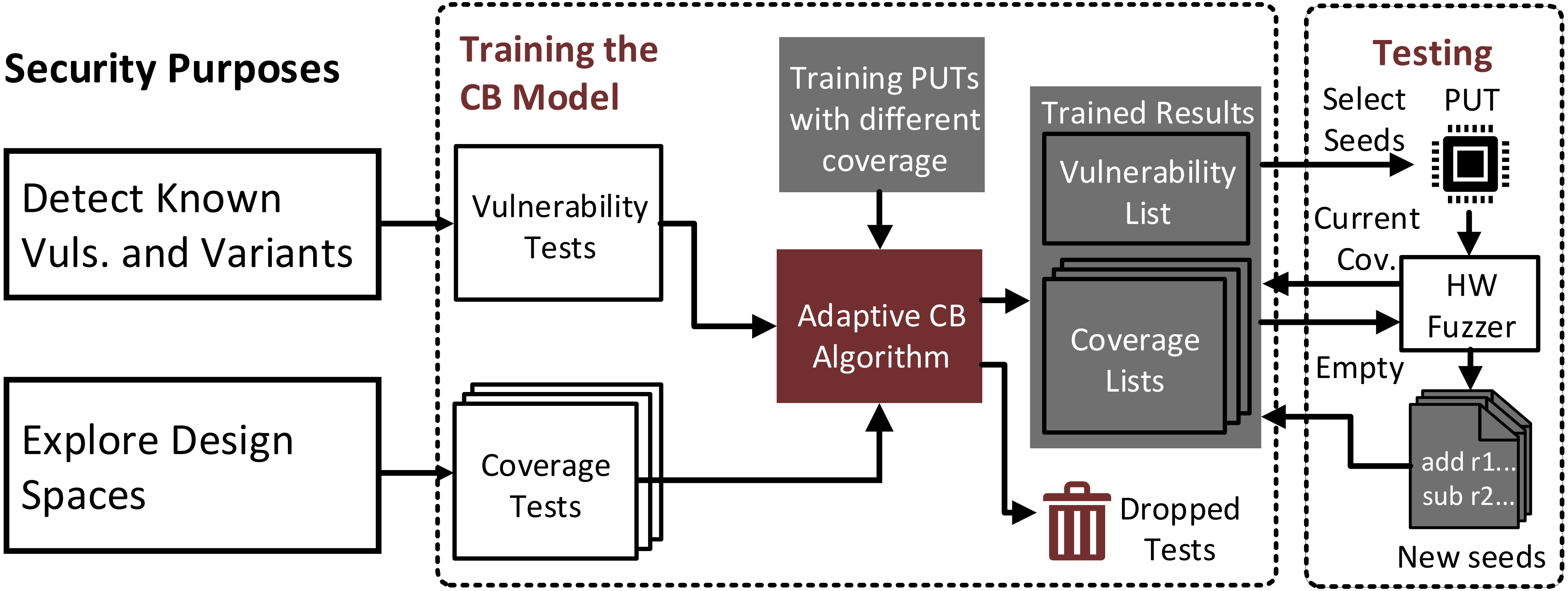
How to remove ineffective tests?



Ineffective tests remain on the lists

Solution 1: Drop Ineffective Tests

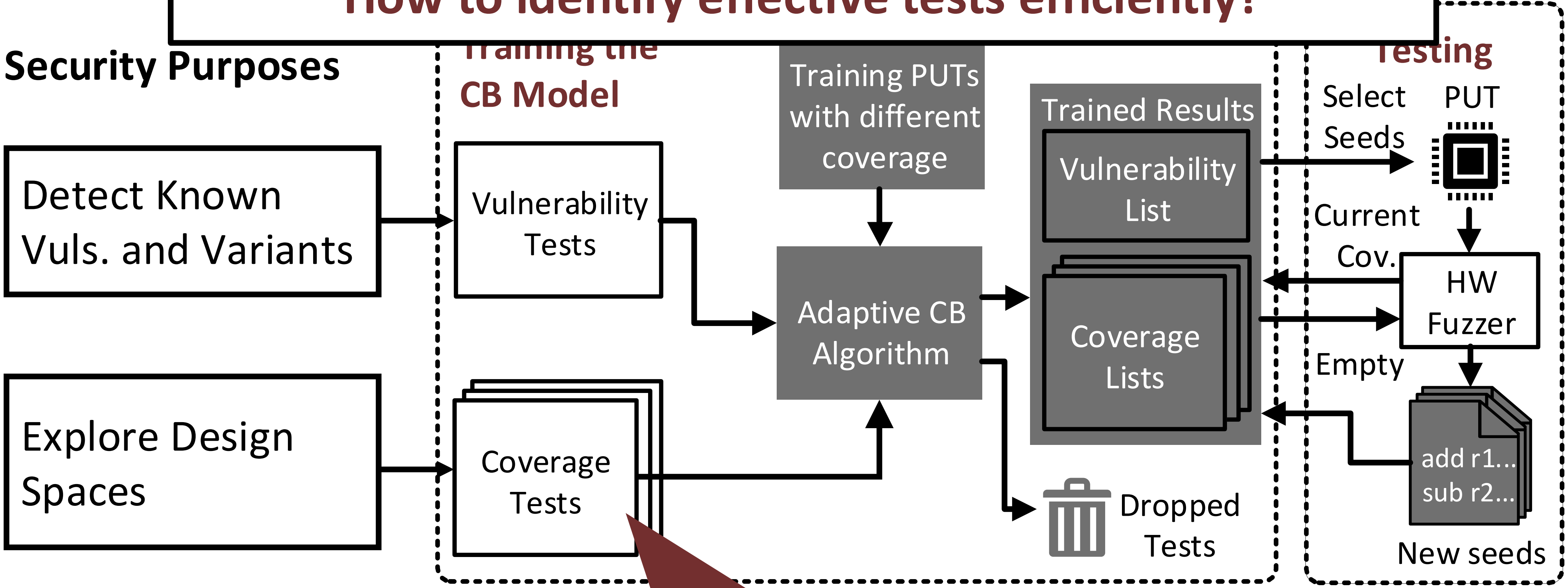
Adaptive CB: CB + *Elimination Function*



Challenge 2: Large Training Test Suite

Fuzzers g tests

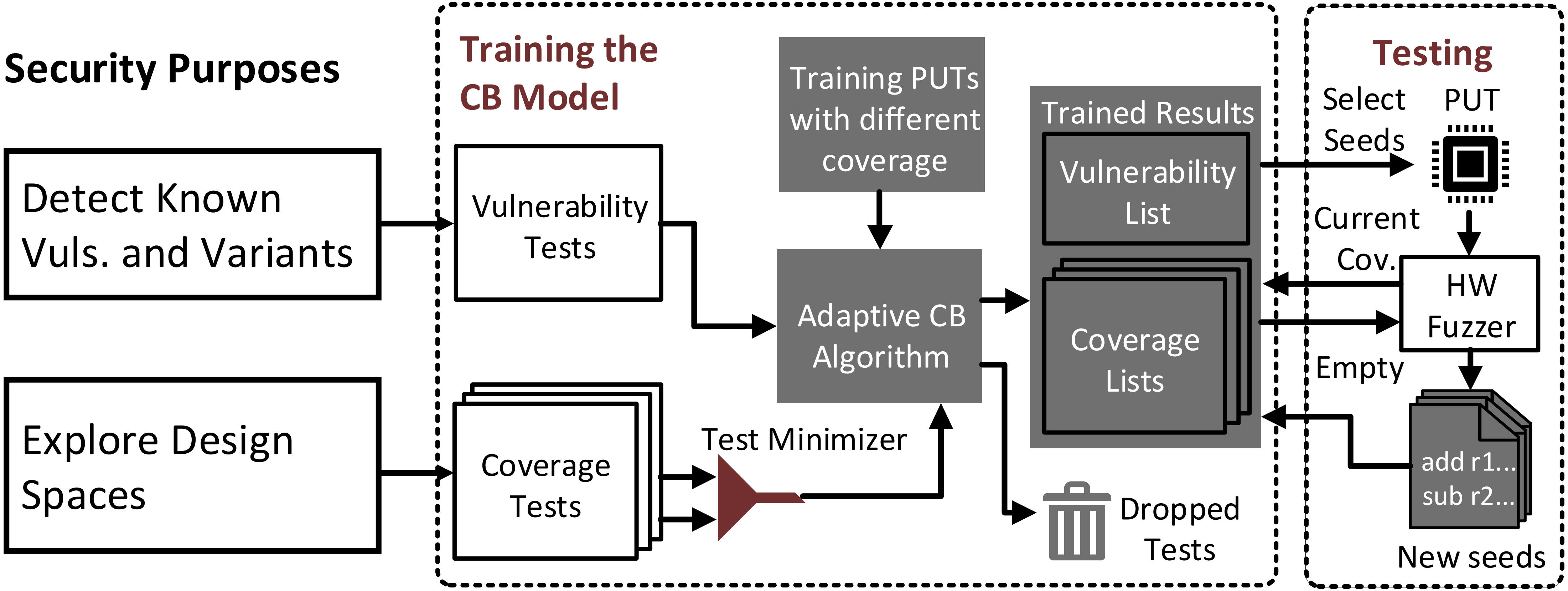
How to identify effective tests efficiently?



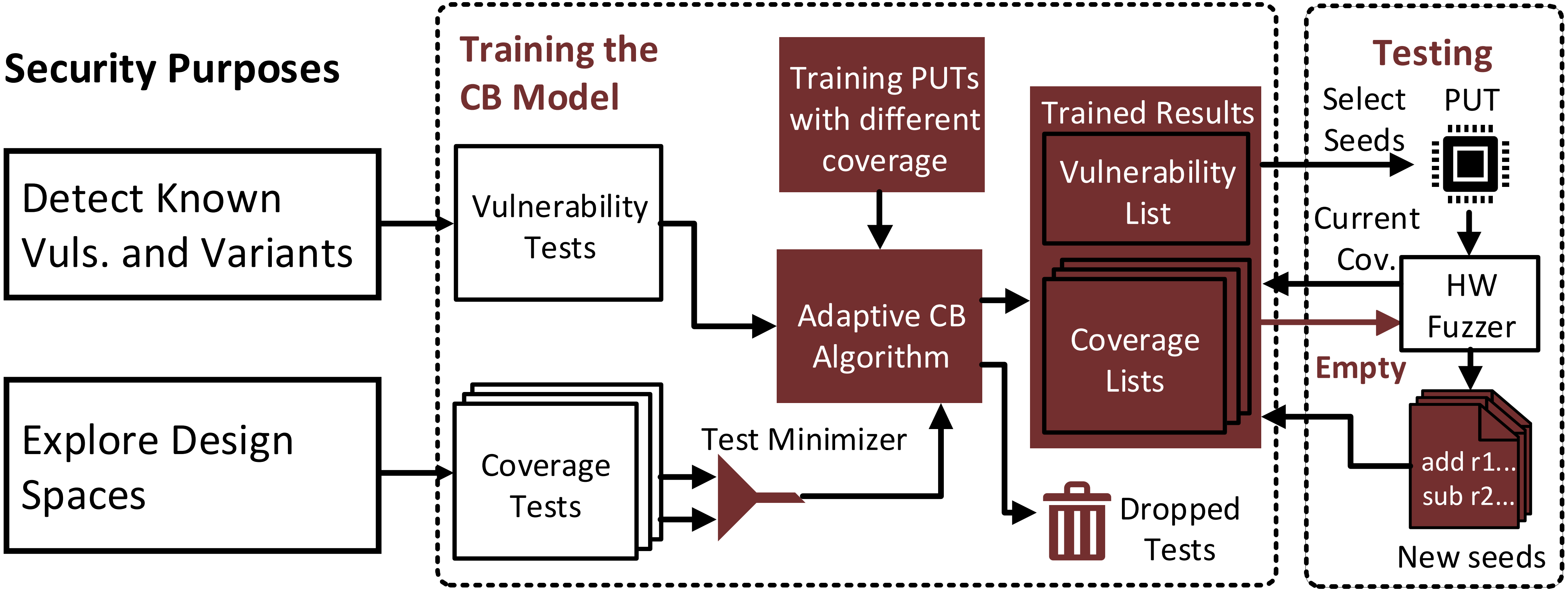
A large number of tests

Solution 2: Test Minimizer

Test minimizer: identify the minimal suite that achieves the same coverage



Putting it All Together



ReFuzz Impact

- Agnostic to baseline Fuzzers: *TheHuzz*, *Cascade*
- Benchmarks: CVA6, Rocket Core, BoomV3, BoomV4, RSD

- Vulnerability Detection
  - **Three** new vulnerabilities. One on RSD is detected by reusing tests from CVA6
  - **Two** new bugs
    - One is cross Rocket Core, BoomV3, and BoomV4
    - One is cross BoomV3 and BoomV4, and BoomV4 has more variants

- Coverage
  - $\mathbf{511.23}\times$ speedup and **9.33%** more total coverage

TEXAS A&M UNIVERSITY

Engineering

THANK YOU