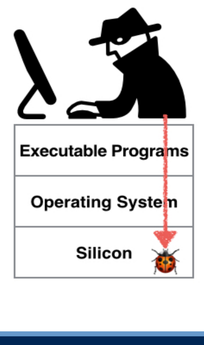
Identifying Security-Critical Properties of a Processor using Machine Learning

Rui Zhang, Natalie Stanley, Christopher Griggs, Andrew Chi, Cynthia Sturton
University of North Carolina, Chapel Hill, NC

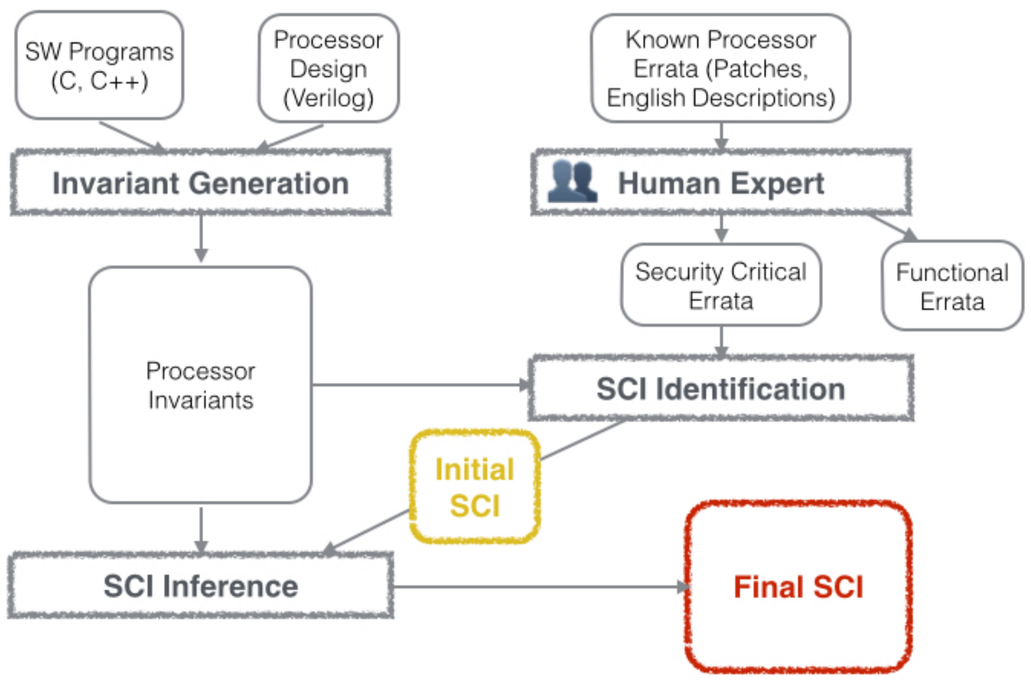
BACKGROUND

- Bugs in processors present vulnerabilities that are exploitable by well-crafted attacks.
- Verification of security properties can prevent the exploitation of vulnerabilities in a processor.



Overview

- A semi-automated methodology to find security critical invariants (SCI) for use in processor verification.
- A tool chain implementing our methodology.
- An evaluation of SCIFinder on the OR1200 RISC processor.



SCIFinder Tool Chain Workflow.

1. Collecting a set of invariants that govern how processor state is updated.
2. Using published errata, identify those invariants violated by prior, exploitable bugs.
3. Using machine learning, find additional invariants that are critical to security.

RESEARCH QUESTION

How to identify the security-critical properties of a processor?

SCI Inference

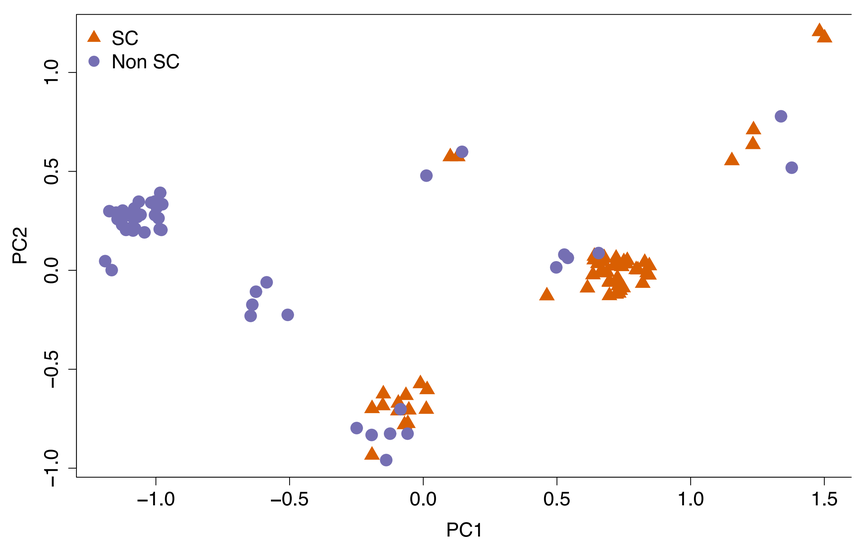
- We use a penalized logistic regression model with elastic net penalty.
- We manually classify whether an invariant is security-critical or not.
- We model the probability of an invariant i to be security-critical or not as follows (y is the class label):

$$p_i = probability(y_i = non\ security\ critical)$$

$$1 - p_i = probability(y_i = security\ critical)$$

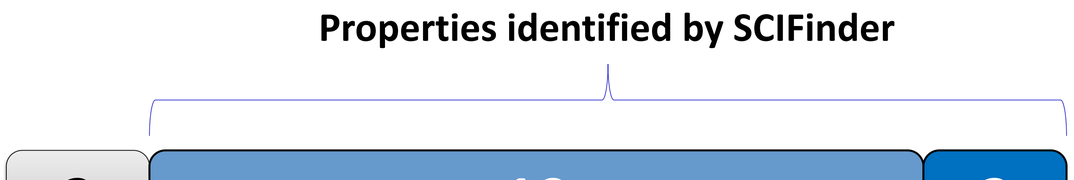
- Let x_i be the set of measured features (general purpose registers, flags, memory addresses, operators).
- We relate p_i to x_i as (β and β_0 are the vector of regression model coefficients and the intercept term):

$$\log\left(\frac{p_i}{1-p_i}\right) = x_i^T\beta + \beta_0$$



PCA on the invariants with selected features.
Invariants cluster adequately according to class label.

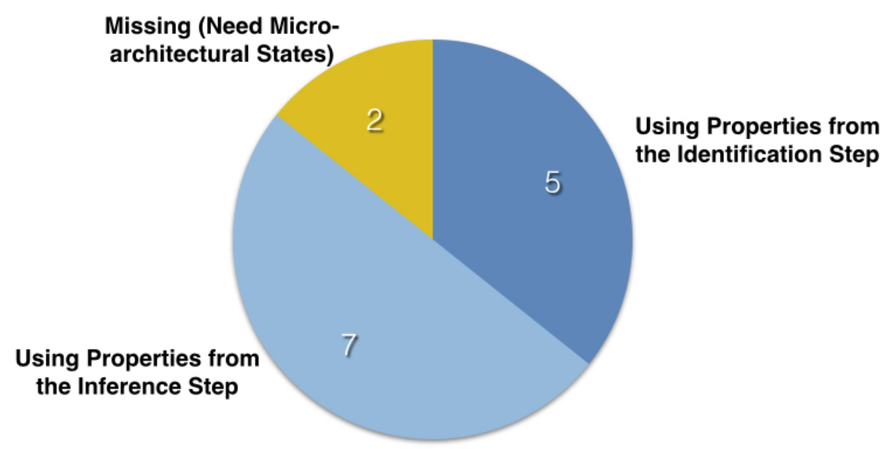
RESULTS



Properties manually crafted in prior work [1, 2]

Example: Link address should not be modified during function call execution

Result: Identifying Security Properties from Prior Work.



Result of detecting 14 AMD errata from SPECS project (bugs not used in the development of the assertions).

Result: Stopping New Bugs.