RAID 2021

SyML: Guiding Symbolic Execution Toward Vulnerable States Through Pattern Learning

Nicola Ruaro, Lukas Dresel, Kyle Zeng, Tiffany Bao, Mario Polino, Andrea Continella, Stefano Zanero, Christopher Kruegel, Giovanni Vigna

POLITECNICO MILANO 1863

UC SANTA BARBARA

Dynamic Symbolic Execution?

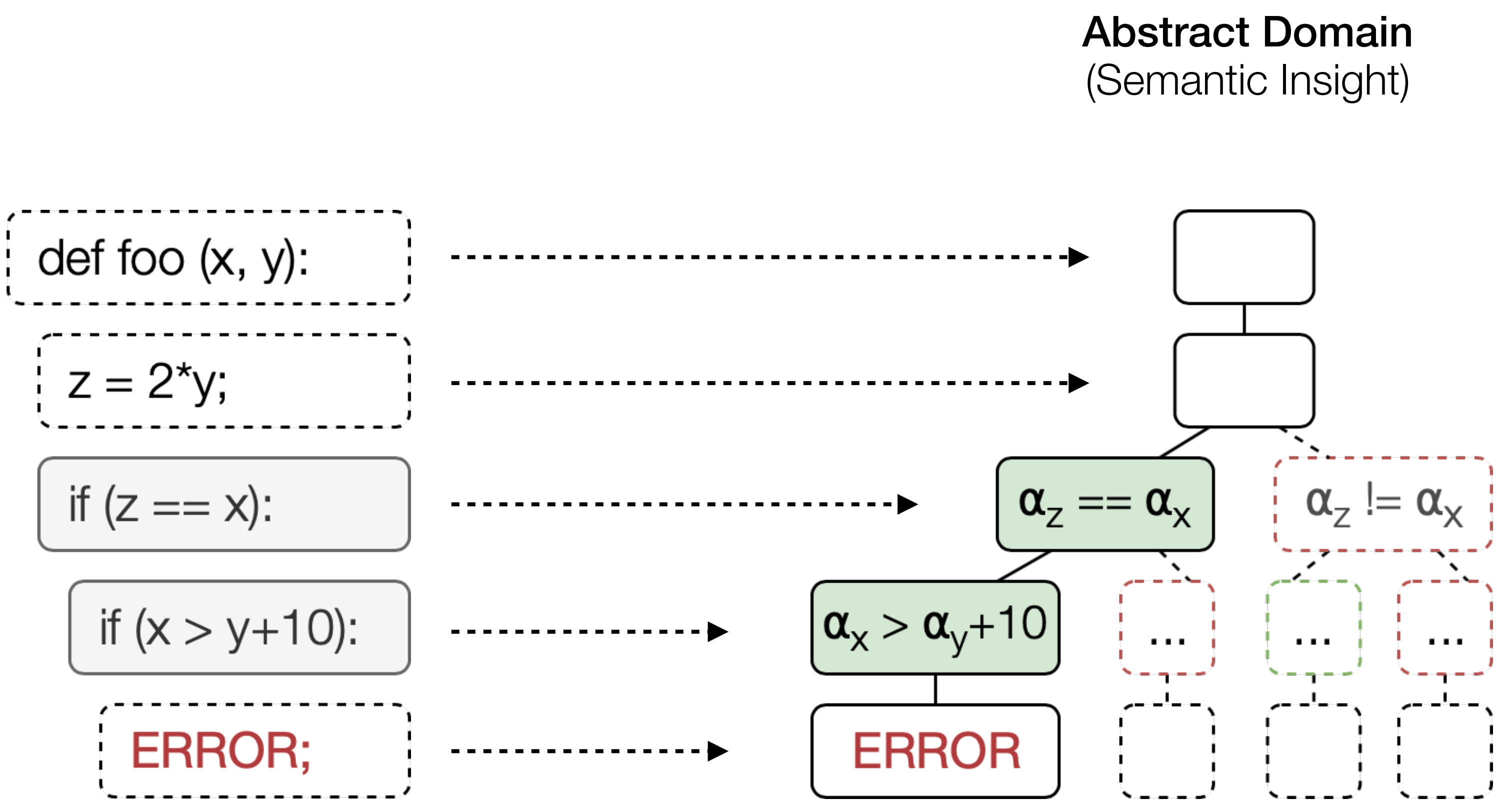
Dynamic Symbolic Execution

Dynamic?

Emulated Environment
(Replayability)

```
def foo (x, y):
  z = 2*y;
  if (z == x):
    if (x > y+10):
      ERROR;
```

Symbolic?



Abstract Domain
(Semantic Insight)
def foo (x, y):
z = 2*y;
if (z == x):
if (x > y+10):
ERROR;
$\alpha_z == \alpha_x$
$\alpha_z != \alpha_x$
$\alpha_x > \alpha_y+10$
...
...
...
ERROR

Dynamic Symbolic Execution

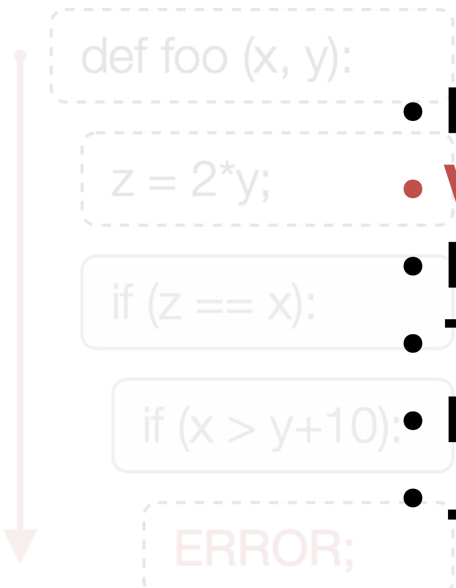
Emulated Environment
(Replayability)

Abstract Domain
(Semantic Insight)

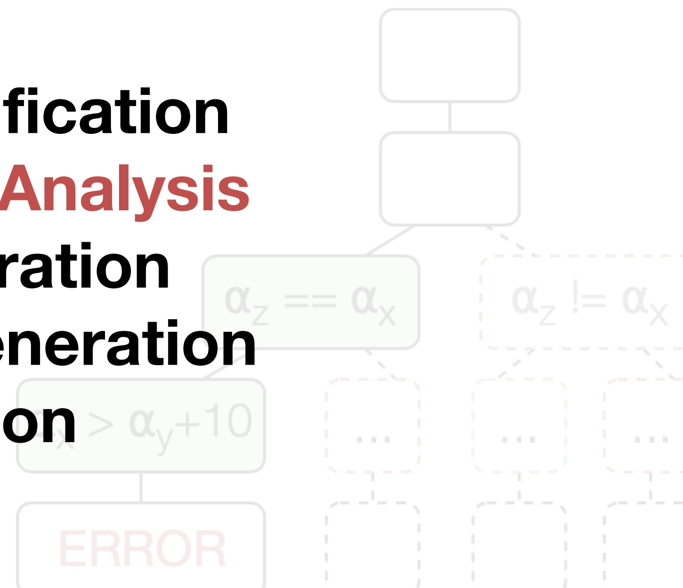
- **Program Verification**
- **Vulnerability Analysis**
- **Exploit Generation**
- **Test-case Generation**
- **De-obfuscation**
- **...**

Dynamic Symbolic Execution

Emulated Environment
(Replayability)



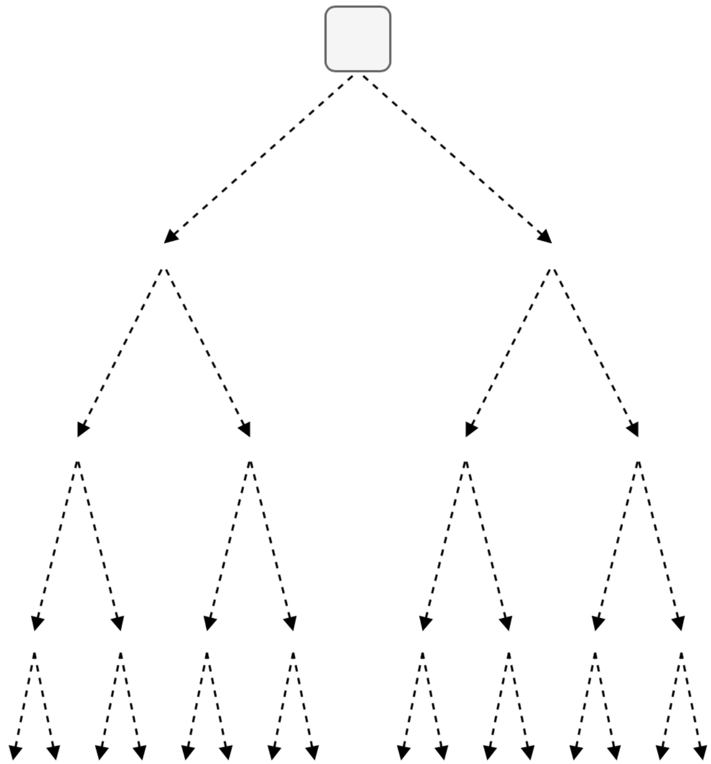
Abstract Domain
(Semantic Insight)



- **Program Verification**
- **Vulnerability Analysis**
- **Exploit Generation**
- **Test-case Generation**
- **De-obfuscation**
- **...**

Path Explosion Problem

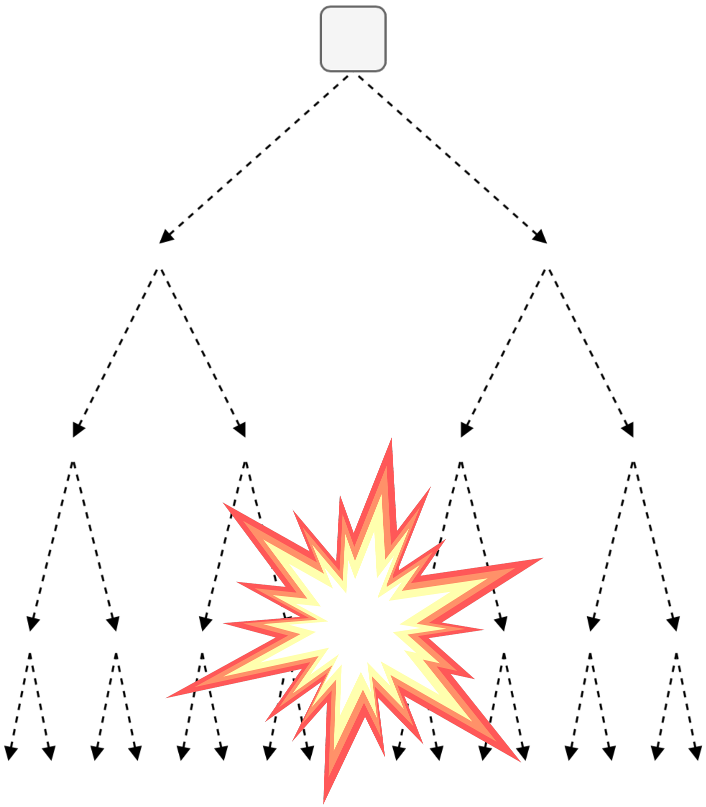
N Conditional Nodes



Path Explosion Problem

N Conditional Nodes
2^N Execution Paths

Limit exploration to a selected subset of execution paths

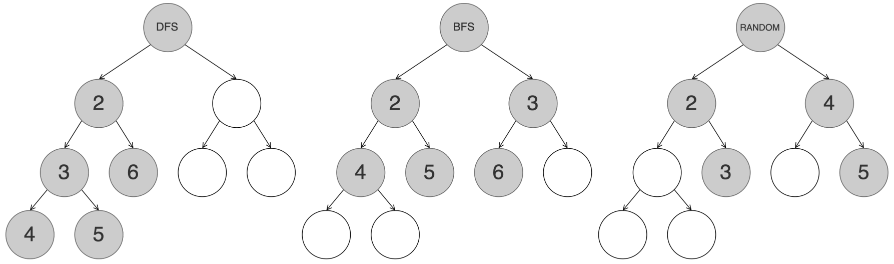


State-of-the-art

1. Symbolic-Assisted Fuzzing (Driller)
2. Under-Constrained Symbolic Execution
3. Merging Execution Paths (Veritesting)
4. Interleaved Symbolic Execution (Symbion)
5. Path Prioritization

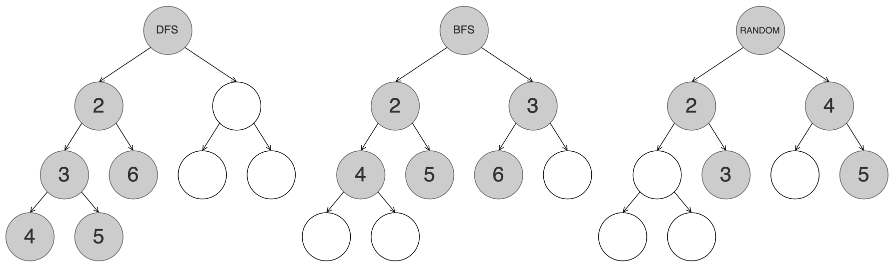
State-of-the-art

1. Symbolic-Assisted Fuzzing (Driller)
2. Under-Constrained Symbolic Execution
3. Merging Execution Paths (Veritesting)
4. Interleaved Symbolic Execution (Symbion)
5. **Path Prioritization**
    A. Classic Tree Traversal
        - Depth First
        - Breadth First
        - Random
    B. Heuristic-Based
        - Loop Exhaustion
        - Coverage Optimization
        - ...



State-of-the-art

1. Symbolic-Assisted Fuzzing (Driller)
2. Under-Constrained Symbolic Execution
3. Merging Execution Paths (Veritesting)
4. Interleaved Symbolic Execution (Symbion)
5. **Path Prioritization**
   A. Classic Tree Traversal
      - Depth First
      - Breadth First
      - Random
   B. Heuristic-Based
      - Loop Exhaustion
      - Coverage Optimization
      - ...

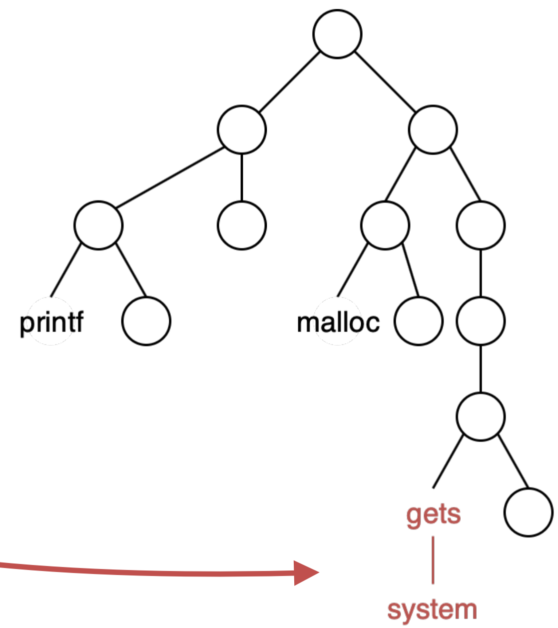


⇨ Shallow and Vulnerability-specific

Approach

Intuition

- More **coverage != ** more **bugs**
- Replicate the expertise of a **human analyst**
- Similar bugs == **similar patterns**
  (API calls, complex functions, etc.)
- Find **interesting execution contexts**



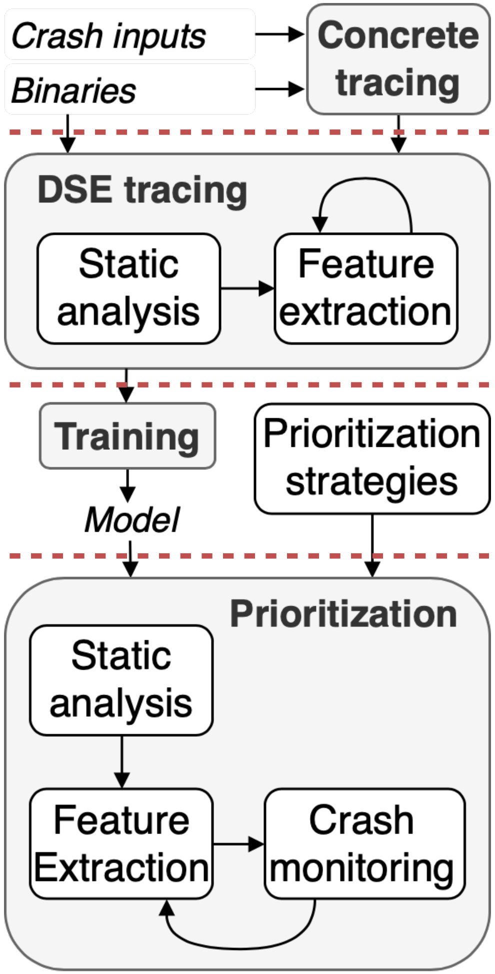
Approach Overview

Stage 1: Concrete Tracing

Stage 2: (Dynamic) Symbolic Tracing

Stage 3: Training

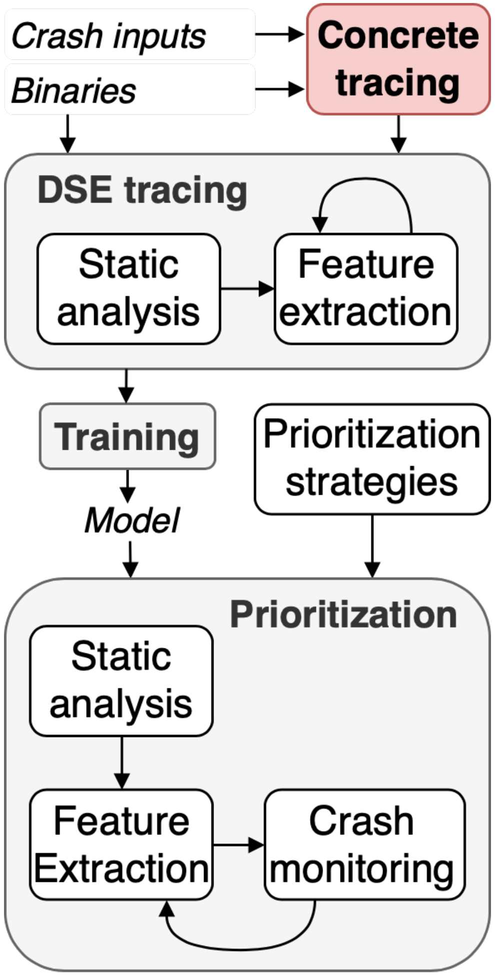
Stage 4: Prioritization



= angr

Stage 1: Concrete Tracing

- Dataset (**binaries** and **known vulnerabilities**)
- Run binary inside the QEMU emulator
- Send crashing input
- Monitor the execution
- Collect **execution traces**

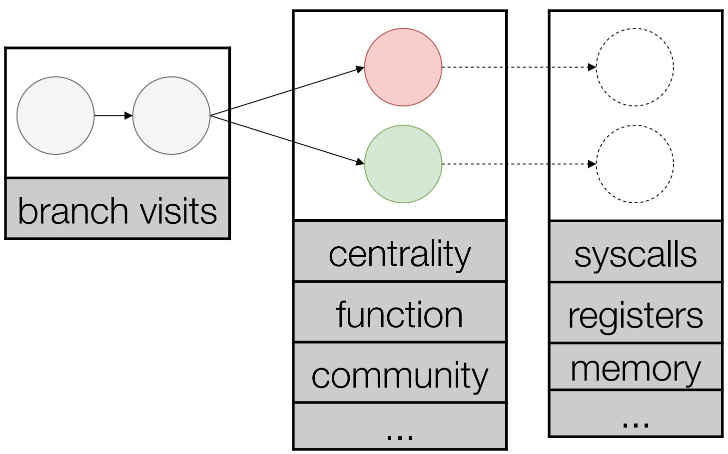


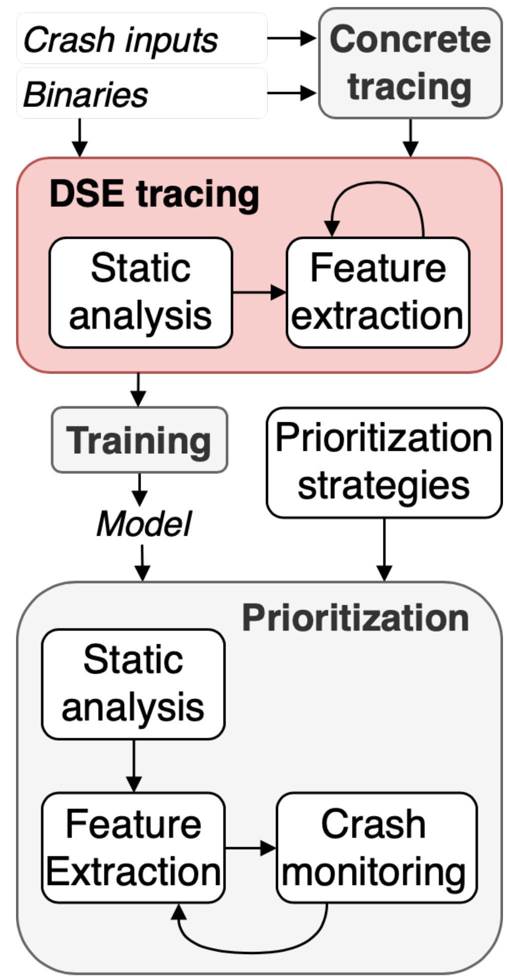
Stage 2: Symbolic Tracing

- **Static analysis** *(CFG, symbols, etc.)*

- Execute in angr
- **Synchronize** execution with recorded trace

- At every conditional node:
  - Create 2 new training points
  - **Extract features**

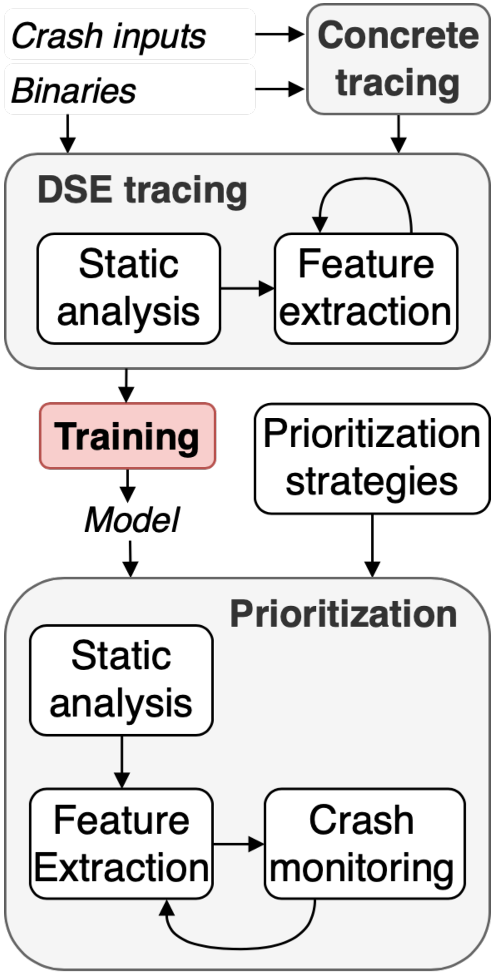




Stage 3: Training

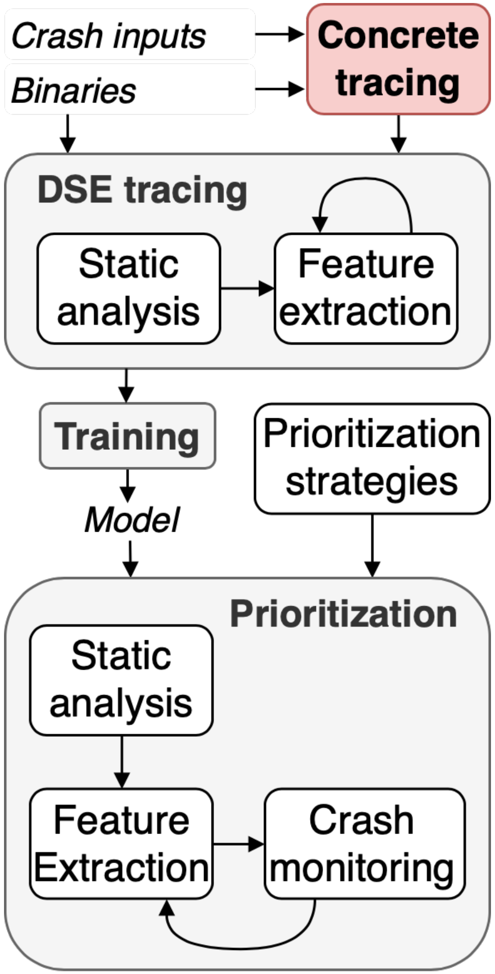
Clean Dataset:
- Numerical features
- Categorical features

Models: Log. Regression, SVM, Dec. Tree, etc.
Metrics: Accuracy, Coverage, F-1, etc.
Cross Validation: Leave-One-Out

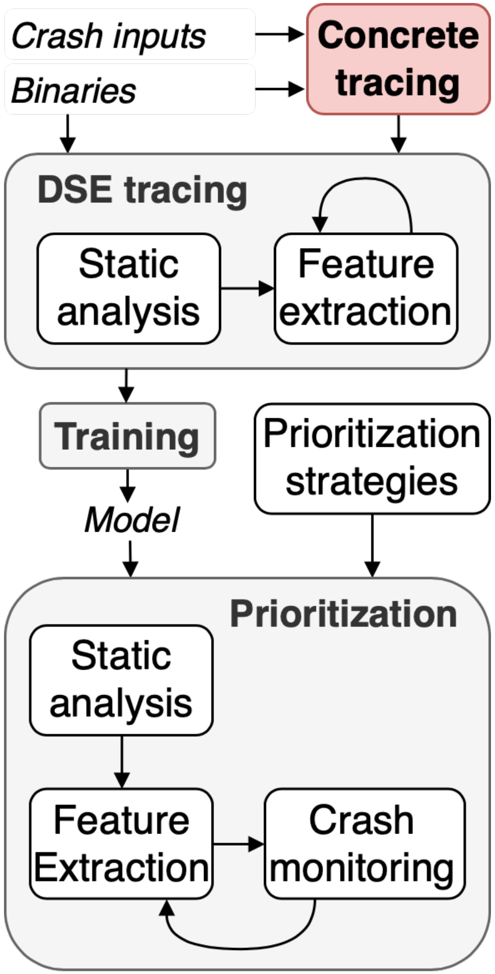


Example

Stage 1: Concrete Tracing

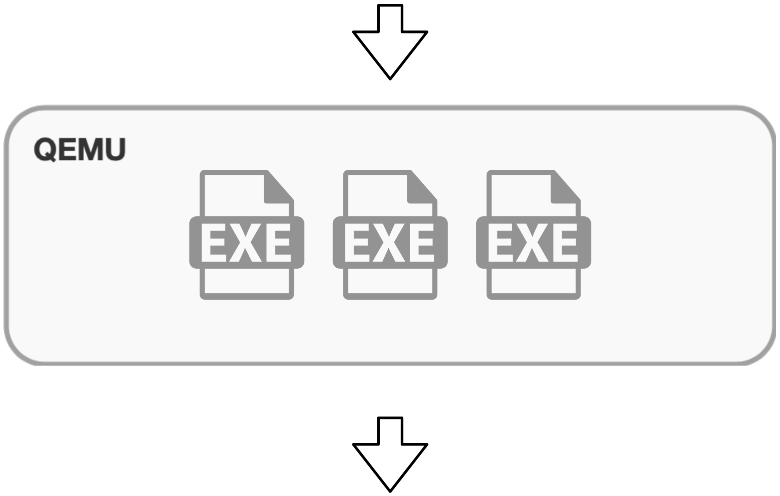


Stage 1: Concrete Tracing

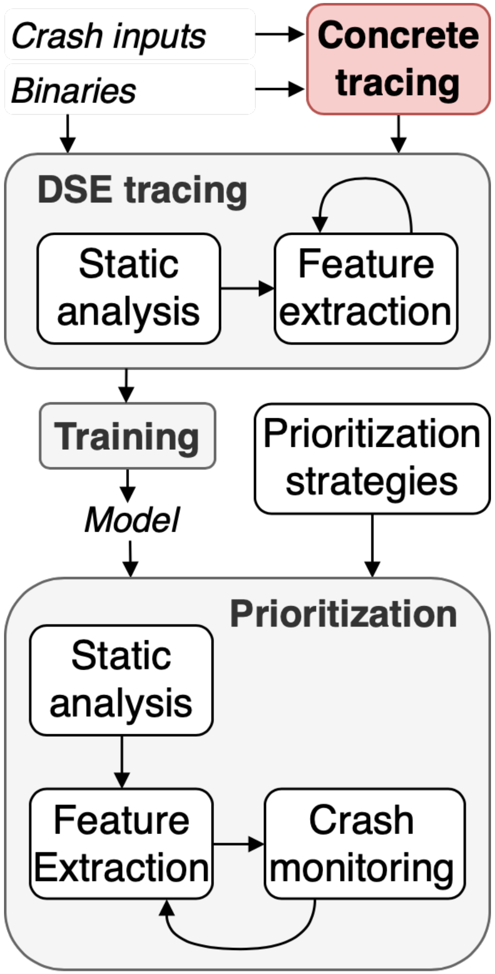


Stage 1: Concrete Tracing

INPUT: A!@^F^J%$#@!~(

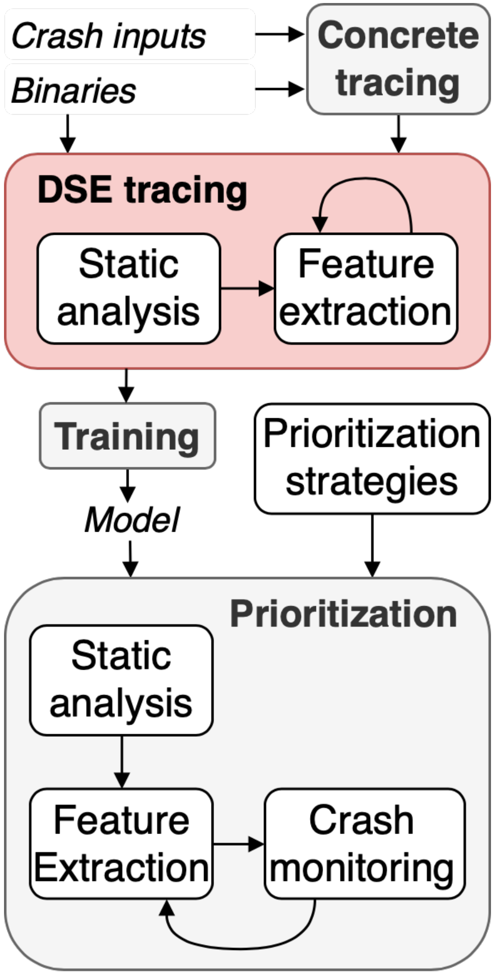


TRACE: 1, 2, 5, 6, 8, 5, 6, 8, 13 ..



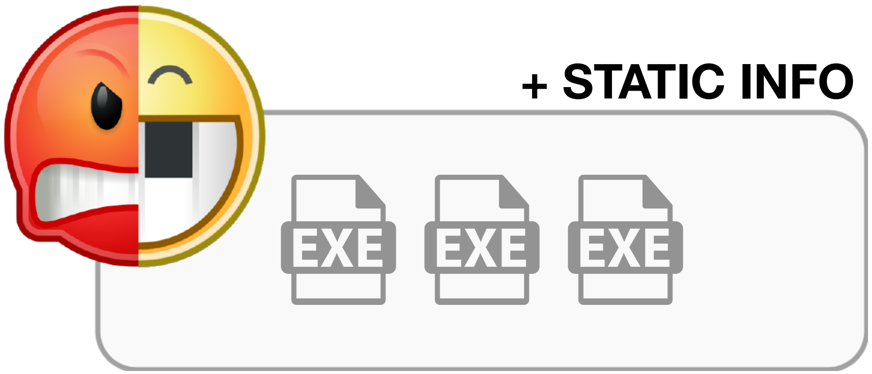
Stage 2: Symbolic Tracing

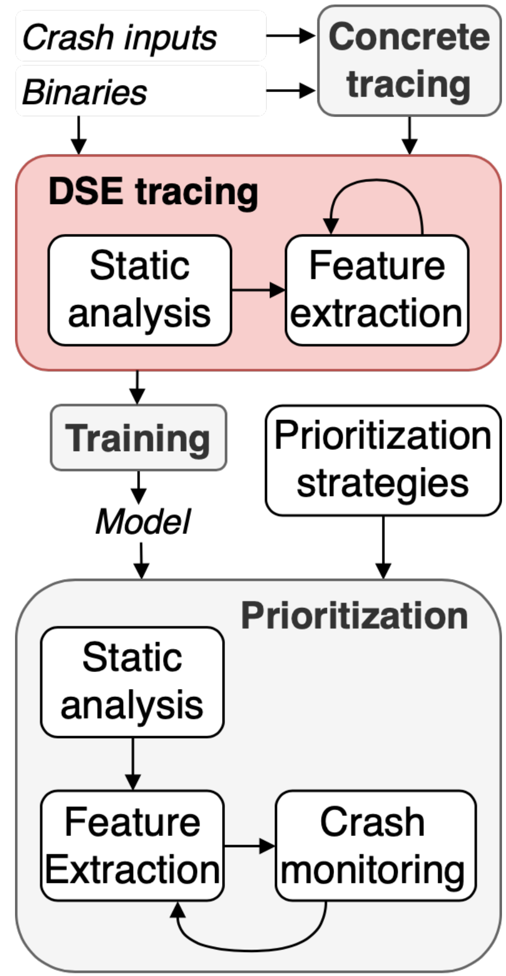
1, 2, 5, 6, 8, 5, 6, 8, 13 ..



Stage 2: Symbolic Tracing

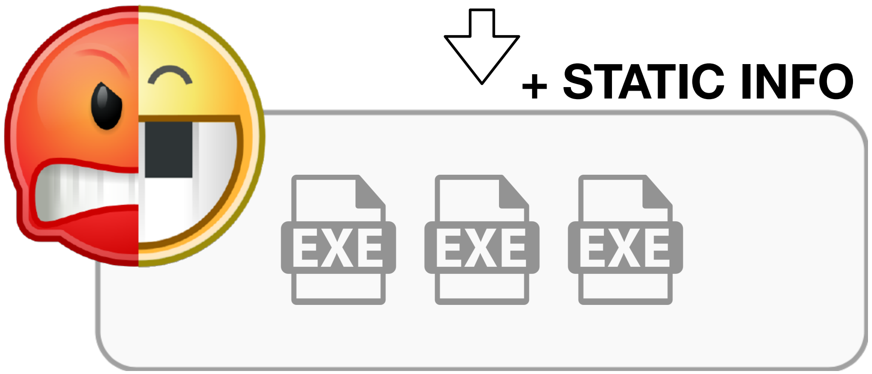
1, 2, 5, 6, 8, 5, 6, 8, 13 ..



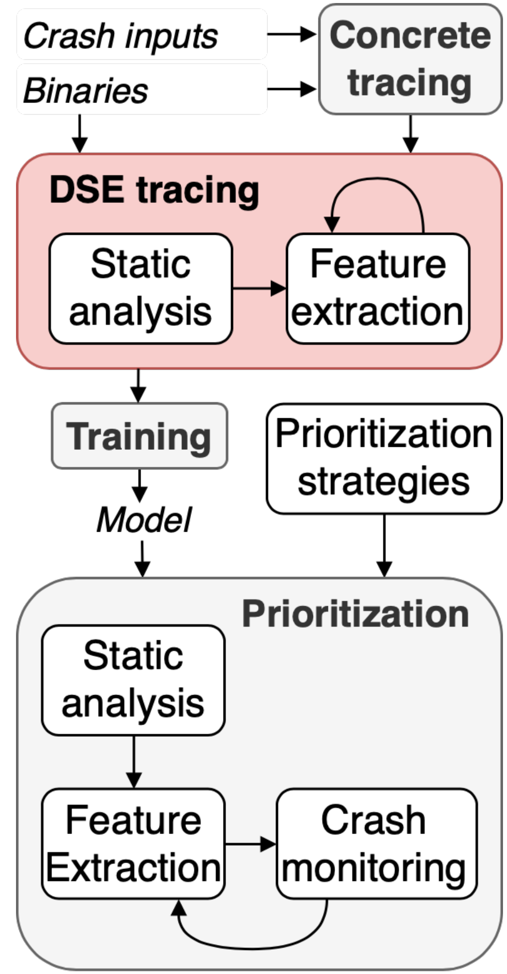


Stage 2: Symbolic Tracing

1, 2, 5, 6, 8, 5, 6, 8, 13 ..

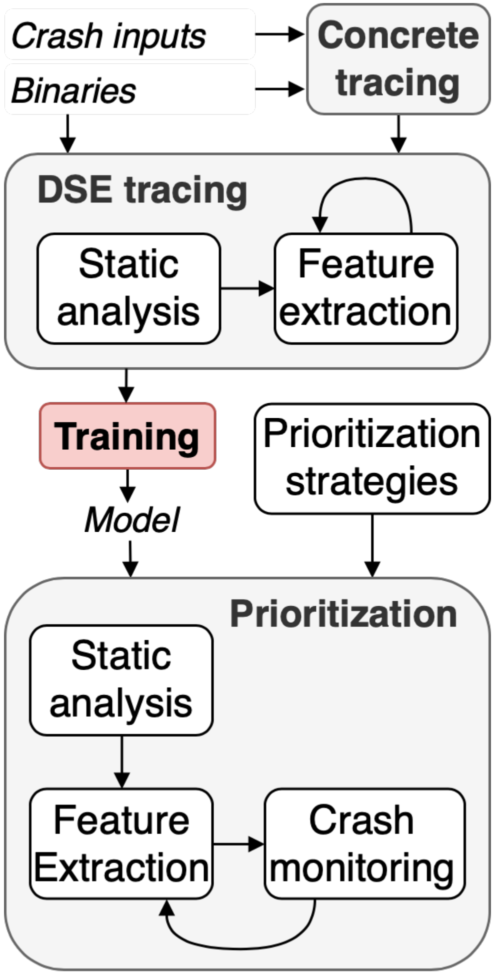


FALSE	111	12	0.0929	{'transmit', '_terminate'}	{'__ne__(SYM,CONCR)'}
TRUE	117	13	0.0112	{'_terminate', 'receive'}	{'CONCR'}



Stage 3: Training

FALSE	111	12	0.0929	{'transmit', '_terminate'}	{'__ne__(SYM,CONCR)'}
TRUE	117	13	0.0112	{'_terminate', 'receive'}	{'CONCR'}



Stage 3: Training

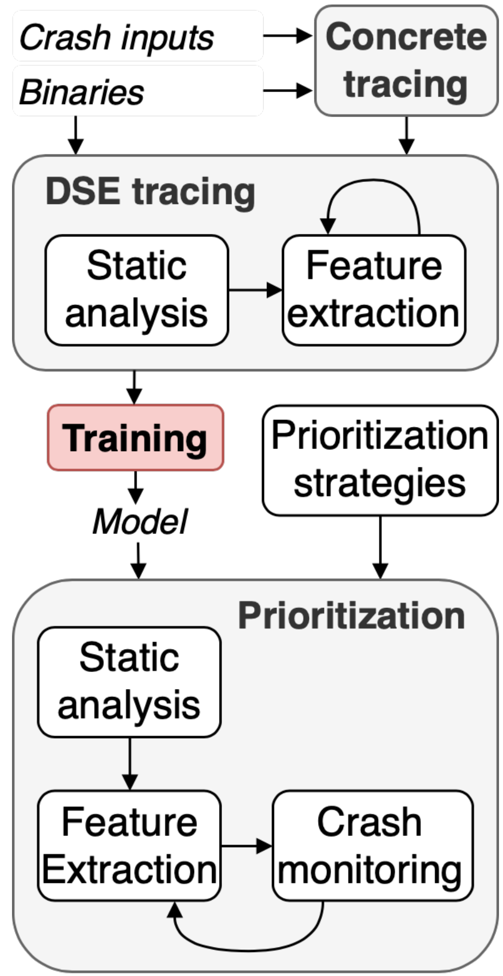
FALSE	111	12	0.0929	{'transmit', '_terminate'}	{'__ne__(SYM,CONCR)'}
TRUE	117	13	0.0112	{'_terminate', 'receive'}	{'CONCR'}

0	111	12	0.0929
1	117	13	0.0112

Numerical

1	1	0	1	0
0	1	1	0	1

Categorical

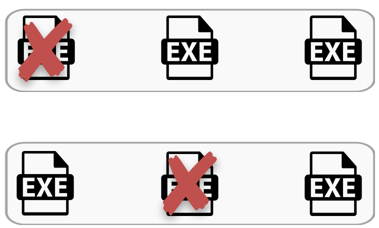


Stage 3: Training

FALSE	111	12	0.0929	{'transmit', '_terminate'}	{'__ne__(SYM,CONCR)'}
TRUE	117	13	0.0112	{'_terminate', 'receive'}	{'CONCR'}

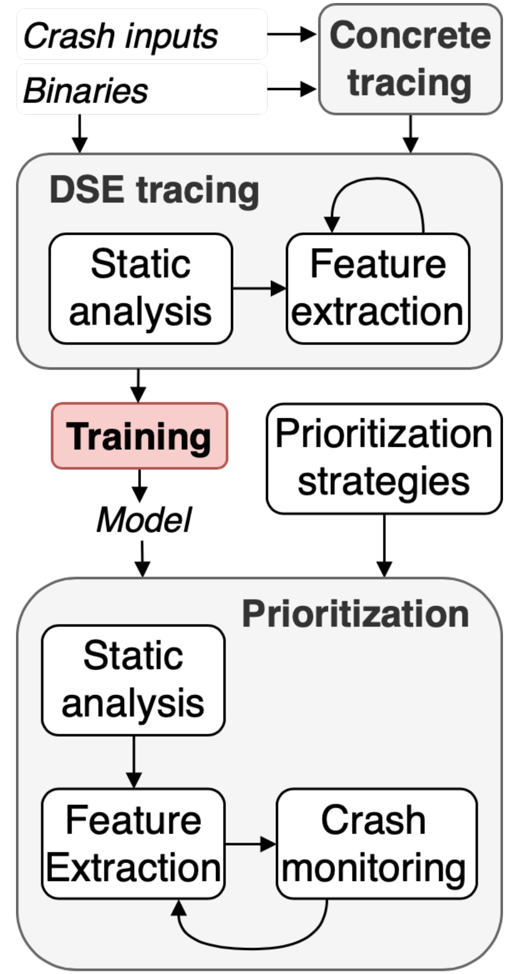
0	111	12	0.0929
1	117	13	0.0112

1	1	0	1	0
0	1	1	0	1



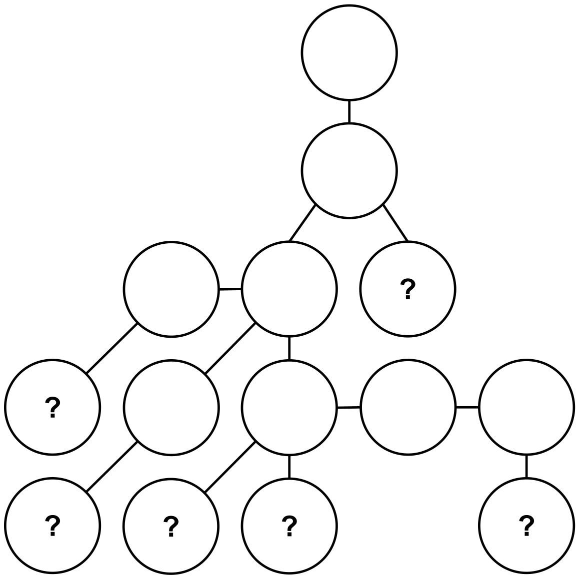
Accuracy?
Coverage?
Time-to-Score?

XGBoost model



Stage 4: Prioritization

XGBoost model

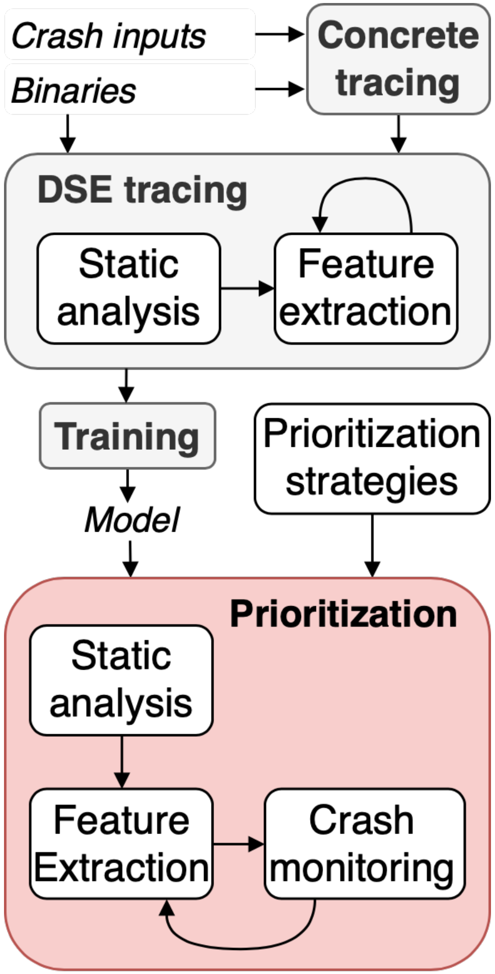


Fast strategy

$$next = \arg\max_{p \in active} \{\overline{score}(p)\}$$

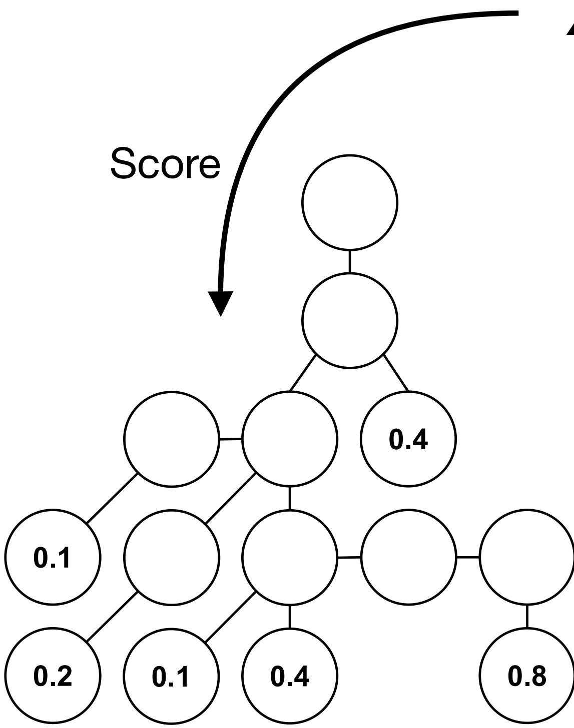
Balanced strategy

$$\Pr(next = p) = \overline{score}(p)$$



Stage 4: Prioritization

XGBoost model

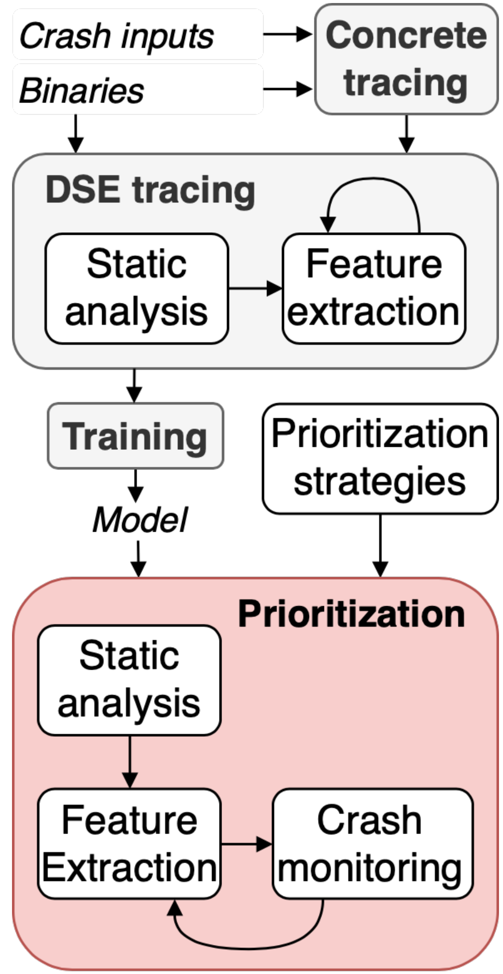


Fast strategy

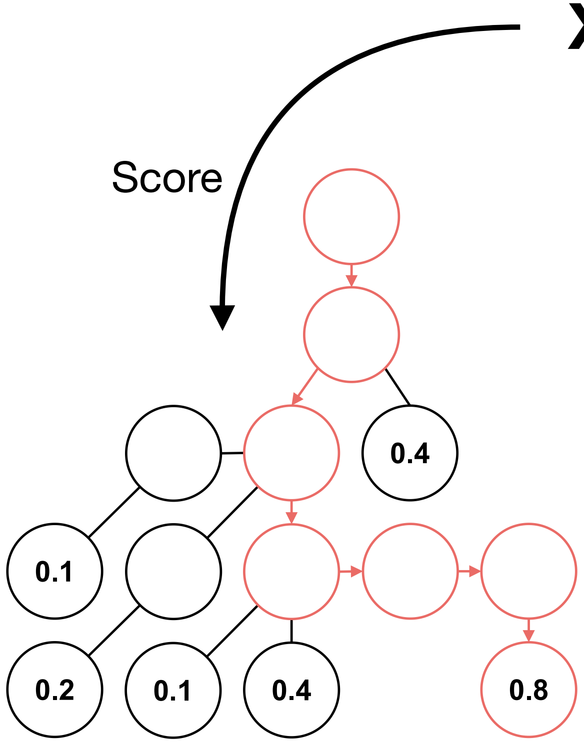
$$next = \arg\max_{p \in active} \{\overline{score}(p)\}$$

Balanced strategy

$$\Pr(next{=}p) = \overline{score}(p)$$



Stage 4: Prioritization



XGBoost model

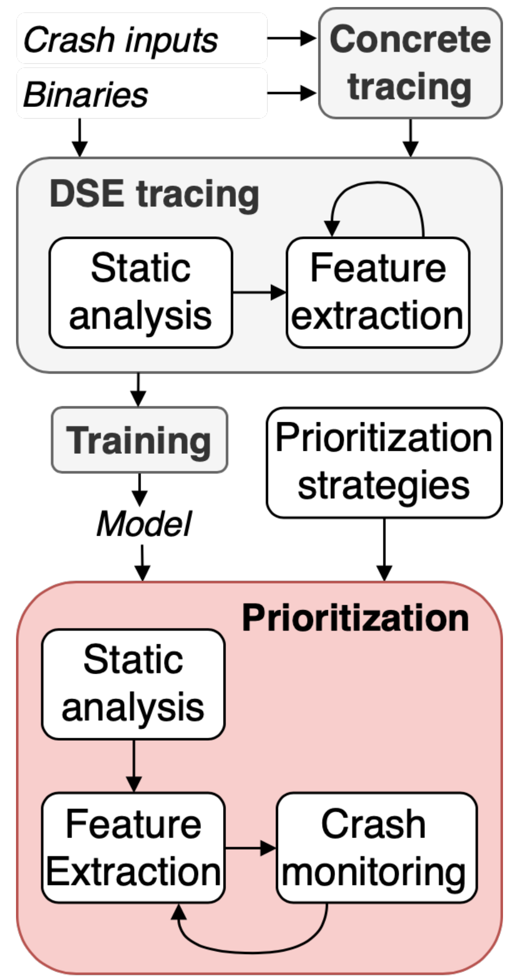
Fast strategy

$$next = \arg\max_{p \in active} \{\overline{score}(p)\}$$

Balanced strategy

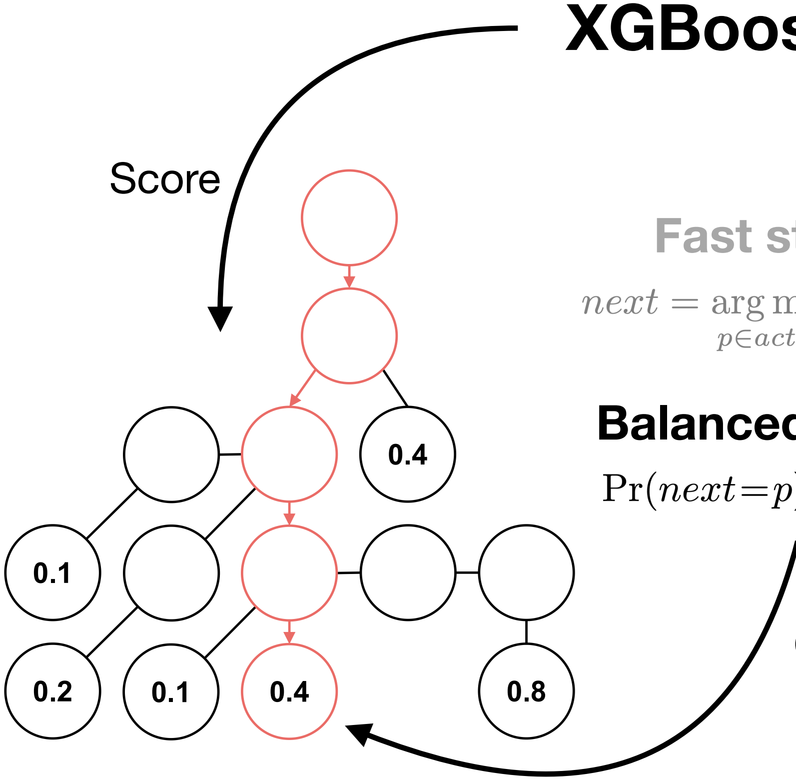
$$\Pr(next{=}p) = \overline{score}(p)$$

Choose



Stage 4: Prioritization

XGBoost model

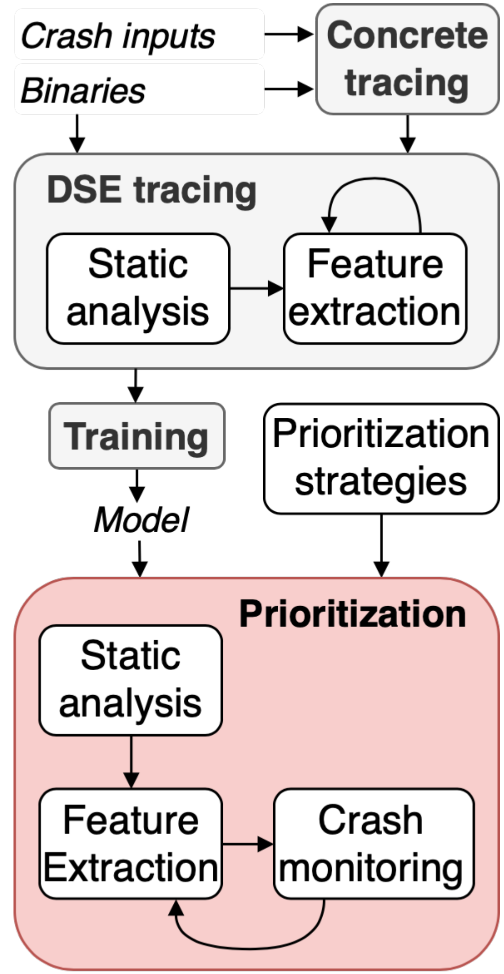


Fast strategy

$$next = \arg\max_{p \in active} \{\overline{score}(p)\}$$

Balanced strategy

$$\Pr(next = p) = \overline{score}(p)$$



Evaluation

Experimental Setup

- Reimplement the state-of-the-art in a **unified framework** (angr)
  - AEG Loop Exhaustion
  - KLEE Coverage Optimization
  - KLEE Random

- Binaries and crashing inputs
  - **CGC Dataset**
  - 3 real-world **Linux CVEs** (transfer learning)

- 1 Binary per CPU Core (3,6GHz)
- Run and monitor for 24 hours
- **Check and classify crashes**

Dataset

- **CGC** dataset (**binaries** and **known vulnerabilities**)
  - Statically compiled x86 binaries
  - Semantics equivalent to Linux binaries
  - Running on DECREE—a different OS with a smaller set of system calls

- **Linux CVEs**
  - CVE-2004-1261 (asp2php)
  - CVE-2004-1288 (o3read)
  - CVE-2004-1292 (ringtonetools)

Model Choice

Model	F1	Accuracy	Trace Coverage	Time-to-Score
LogRegr	77%	66%	73%	0.01s
LinDiscr	76%	68%	75%	0.01s
KNN	79%	63%	70%	0.1s
SVM	82%	76%	72%	0.04s
MLP	81%	80%	68%	0.04s
DecisionTree	85%	80%	78%	0.02s
RandomForest	92%	90%	90%	0.32s
AdaBoost	93%	91%	83%	0.02s
XGBoost	94%	93%	91%	0.2s

Model Choice

Model	F1	Accuracy	Trace Coverage	Time-to-Score
LogRegr	77%	66%	73%	0.01s
LinDiscr	76%	68%	75%	0.01s
KNN	79%	63%	70%	0.1s
SVM	82%	76%	72%	0.04s
MLP	81%	80%	68%	0.04s
DecisionTree	85%	80%	78%	0.02s
RandomForest	92%	90%	90%	0.32s
AdaBoost	93%	91%	83%	0.02s
XGBoost	94%	93%	91%	0.2s

Model Choice

Model	F1	Accuracy	Trace Coverage	Time-to-Score
LogRegr	77%	66%	73%	0.01s
LinDiscr	76%	68%	75%	0.01s
KNN	79%	63%	70%	0.1s
SVM	82%	76%	72%	0.04s
MLP	81%	80%	68%	0.04s
DecisionTree	85%	80%	78%	0.02s
RandomForest	92%	90%	90%	0.32s
AdaBoost	93%	91%	83%	0.02s
XGBoost	94%	93%	91%	0.2s

Performance constraints:

- **Simpler/Faster model**

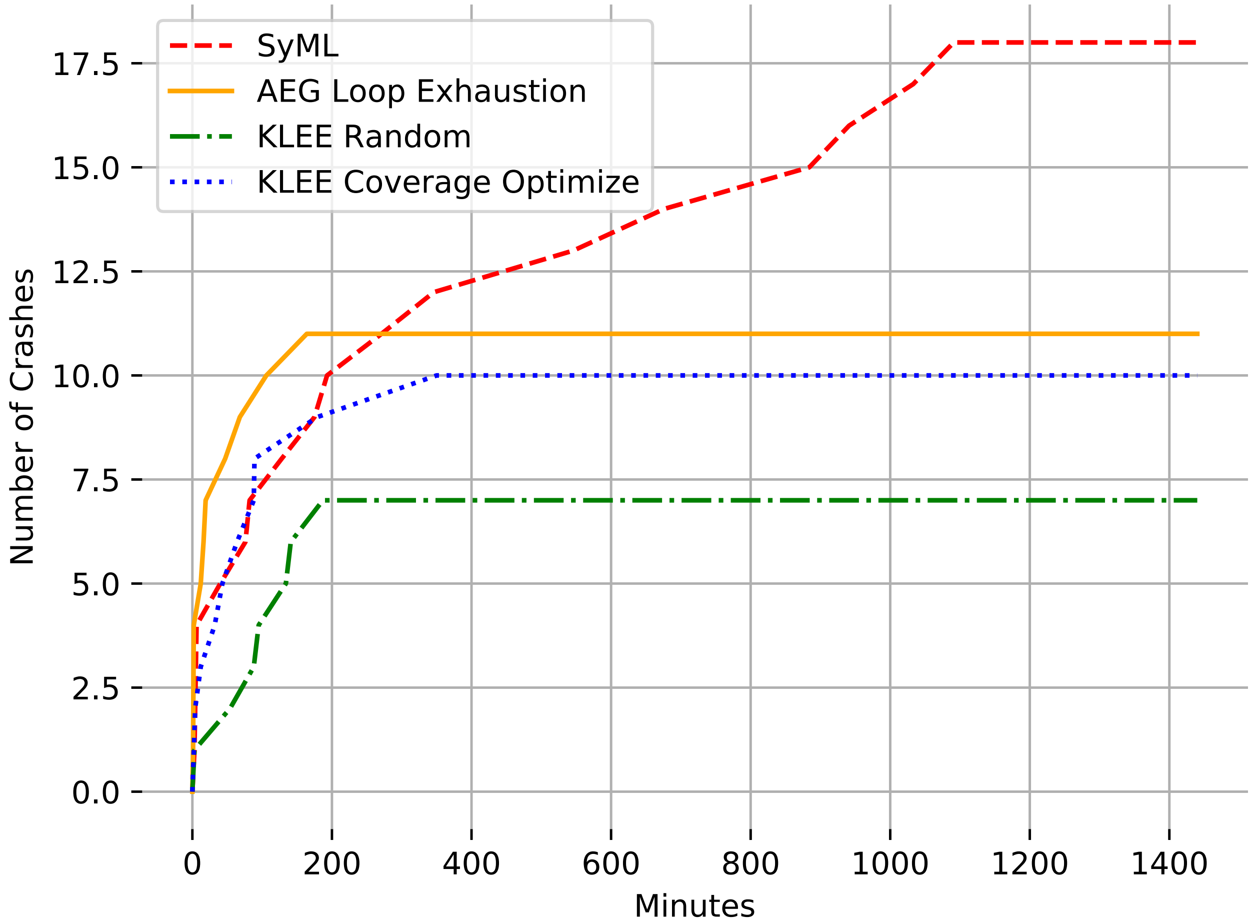
Model Choice

Model	F1	Accuracy	Trace Coverage	Time-to-Score
LogRegr	77%	66%	73%	0.01s
LinDiscr	76%	68%	75%	0.01s
KNN	79%	63%	70%	0.1s
SVM	82%	76%	72%	0.04s
MLP	81%	80%	68%	0.04s
DecisionTree	85%	80%	78%	0.02s
~~RandomForest~~	~~92%~~	~~90%~~	~~90%~~	~~0.32s~~
~~AdaBoost~~	~~93%~~	~~91%~~	~~83%~~	~~0.02s~~
XGBoost	94%	93%	91%	0.2s

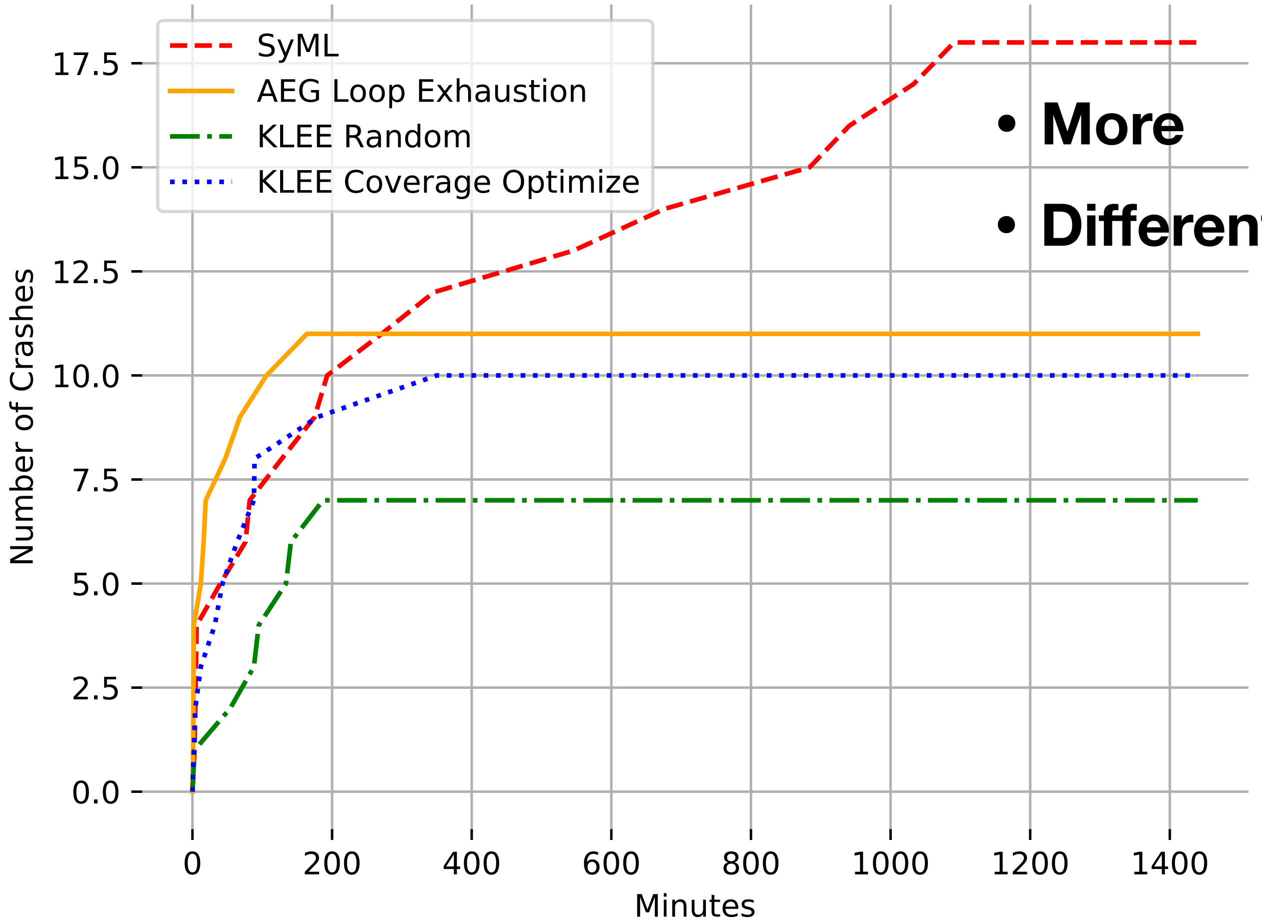
Performance constraints:

- **Simpler/Faster model**

Comparison Results



Comparison Results



- **More**
- **Different**

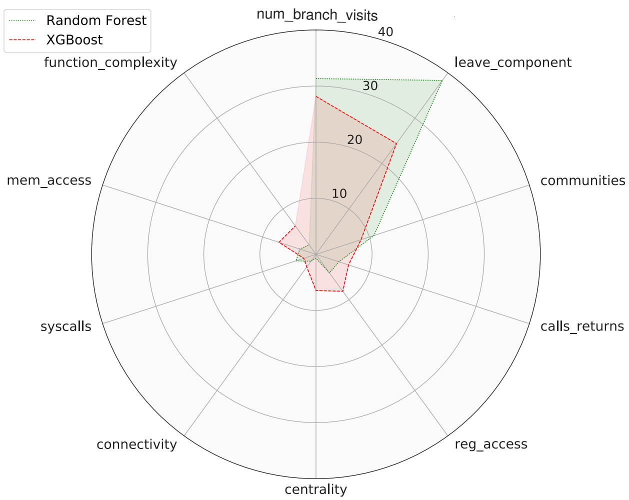
Model Analysis

Features Importance

Prediction Scores Distribution

Model Analysis

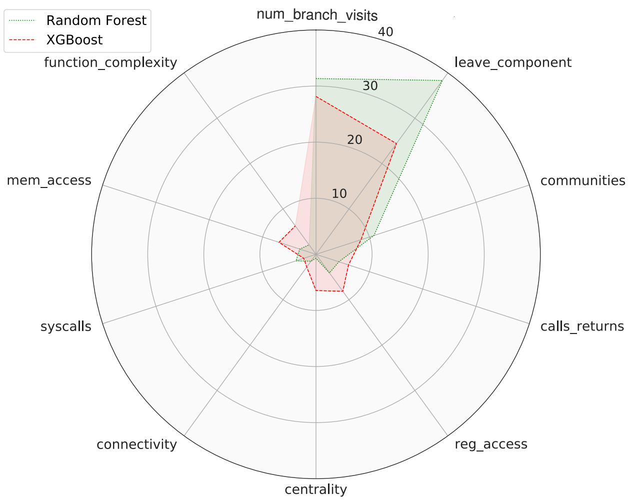
Features Importance

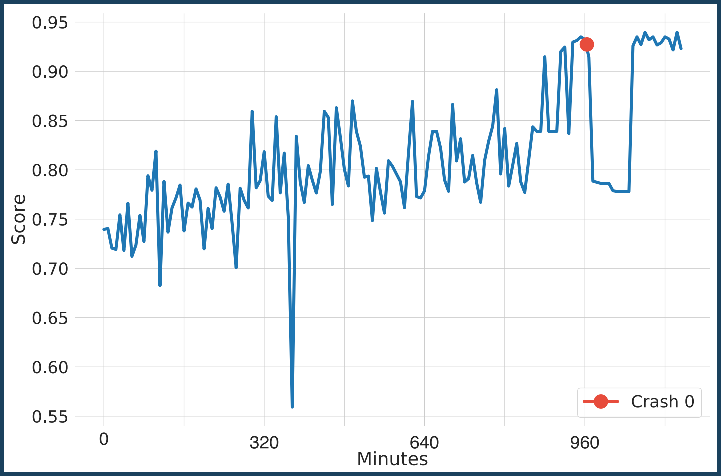


Prediction Scores Distribution

Model Analysis

Features Importance





Prediction Scores Distribution

Transfer Learning

- DSE inaccuracies make it hard to re-trace Linux binaries
- CGC semantics are analogous to the Linux x86 semantics
  - This allows us to transfer some of the knowledge learned from the larger CGC dataset to the Linux dataset

Model	F1	Accuracy
RandomForest (Linux)	63%	70%
AdaBoost (Linux)	63%	63%
XGBoost (Linux)	51%	56%
XGBoost (CGC)	69%	54%
XGBoost (CGC+Linux)	77%	66%

Conclusion

- We propose a **novel path prioritization** approach, leveraging supervised learning algorithms to steer DSE and reach interesting paths
- We evaluate our approach on the CGC dataset, **outperforming prior work** with more (and different) vulnerabilities
- We effectively **transfer the models learned** on the CGC dataset to achieve a better prediction accuracy on 3 real-world CVEs affecting Linux

Future Work

- Train on a **large dataset of Linux binaries** using a different re-tracing framework
- Adapt and apply to guide **hybrid fuzzing**

RAID 2021

Thank You!

SyML: Guiding Symbolic Execution Toward Vulnerable States Through Pattern Learning

Nicola Ruaro, Lukas Dresel, Kyle Zeng, Tiffany Bao, Mario Polino, Andrea Continella, Stefano Zanero, Christopher Kruegel, Giovanni Vigna

Nicola Ruaro - ruaronicola@ucsb.edu

POLITECNICO MILANO 1863

UC SANTA BARBARA