RCABench: Open Benchmarking Platform for Root Cause Analysis

Keisuke Nishimura, Yuichi Sugiyama, Yuki Koike, Masaya Motoda, Tomoya Kitagawa, Toshiki Takatera, Yuma Kurogome

{keisuken, yuichis, yukik, masayam, tomoyak, toshikit, yumak}
@ricsec.co.jp

Ricerca Security, Inc., Tokyo, Japan

RICERCA SECURITY

Fuzzers find a lot of bugs automatically

- OSS-Fuzz: 8,900+ vulnerabilities and 28,000+ bugs
- ClusterFuzz: ~27,000 bugs in Google

“We got inputs that cause crashes automatically.”
“How do we process them? Manual analysis?”

RCA (Root Cause Analysis) a.k.a Fault Localization

Automatic crash analysis

Program
(Souce Code)

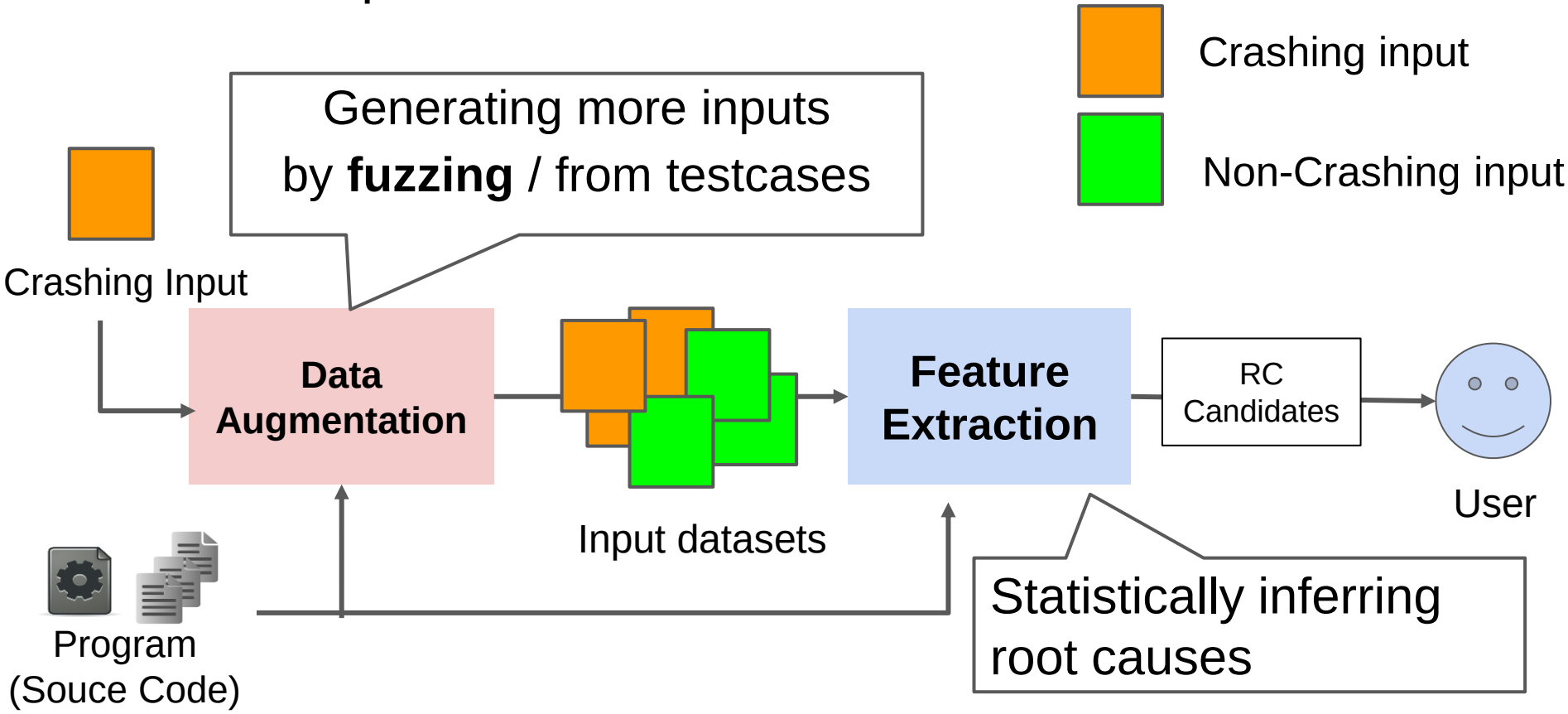
Crashing Input

RCA
tool

Candidates of Root Cause

- Lines
- Basic Blocks
- …

Internal components of RCA tools



Evaluation of RCA techniques is challenging…

#1: Non-uniqueness of root cause definition

#2: Tightly coupled RCA steps

#3: Variance of Data Augmenatation

Challenge #1: Non-uniqueness of root cause location

Multiple possible patches for CVE-2017-15232

```
for (row = 0; row < num_rows; row++) {

  jzero_far((void *) output_buf[row],
   (size_t) (width * sizeof(JSAMPLE)));
```

NULL-able

Original souce code

Challenge #1: Non-uniqueness of root cause location

Multiple possible patches for CVE-2017-15232

```
for (row = 0; row < num_rows; row++) {

  jzero_far((void *) output_buf[row],
   (size_t) (width * sizeof(JSAMPLE)));
```

NULL-able

Original souce code

Where is/are the RC location(s)?

Challenge #1: Non-uniqueness of root cause location

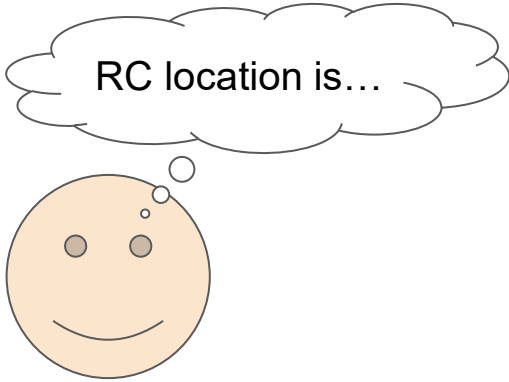
Multiple possible patches for CVE-2017-15232

HERE

```
for (row = 0; row < num_rows; row++) {

  jzero_far((void *) output_buf[row],
   (size_t) (width * sizeof(JSAMPLE)));
```

Original souce code



RCA Tool #1

Challenge #1: Non-uniqueness of root cause location

Multiple possible patches for CVE-2017-15232

```
+ if (output_buf == NULL) {
+   ERREXIT();
+ }

  for (row = 0; row < num_rows; row++) {

    jzero_far((void *) output_buf[row],
     (size_t) (width * sizeof(JSAMPLE)));
```

Possible Patch #1

Challenge #1: Non-uniqueness of root cause location

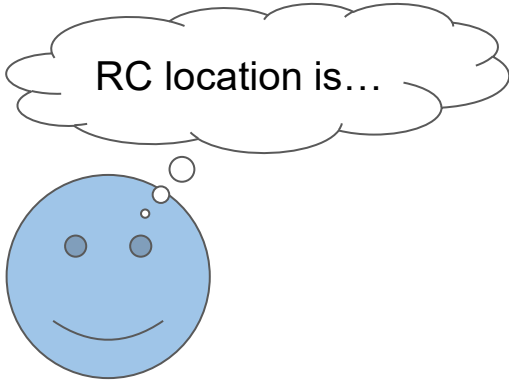
Multiple possible patches for CVE-2017-15232

```
for (row = 0; row < num_rows; row++) {

                HERE

    jzero_far((void *) output_buf[row],
     (size_t) (width * sizeof(JSAMPLE)));
```

Original souce code



RCA Tool #2

Challenge #1: Non-uniqueness of root cause location

Multiple possible patches for CVE-2017-15232

```
for (row = 0; row < num_rows; row++) {

+  if (output_buf == NULL) {
+    ERREXIT();
+  }

    jzero_far((void *) output_buf[row],
     (size_t) (width * sizeof(JSAMPLE)));
```

Possible Patch #2

Challenge #1: Non-uniqueness of root cause location

Many possible candidates for root cause locations

The ground truth is …

Line 10 in function A

Evaluator 1

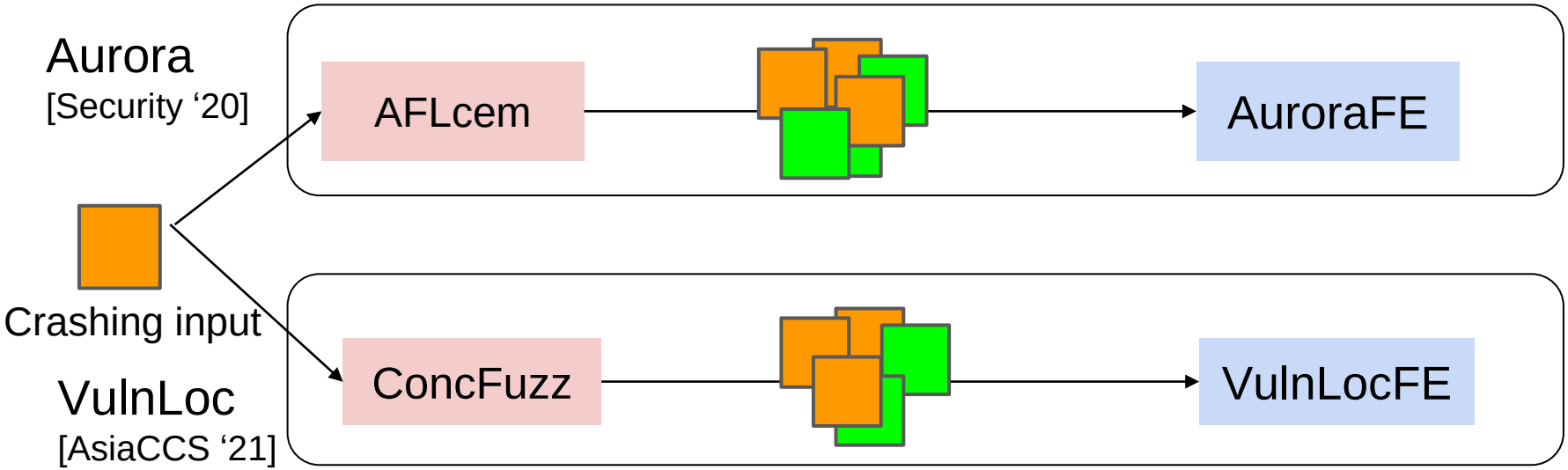
Line 50 in function A
Line 10 in function B

Evaluator 2

Line 80 in function A
Line 50 in function A
Line 10 in function B

Evaluator 3

Challenge #2: Tightly coupled RCA steps



Data Augmentation

Collecting inputs

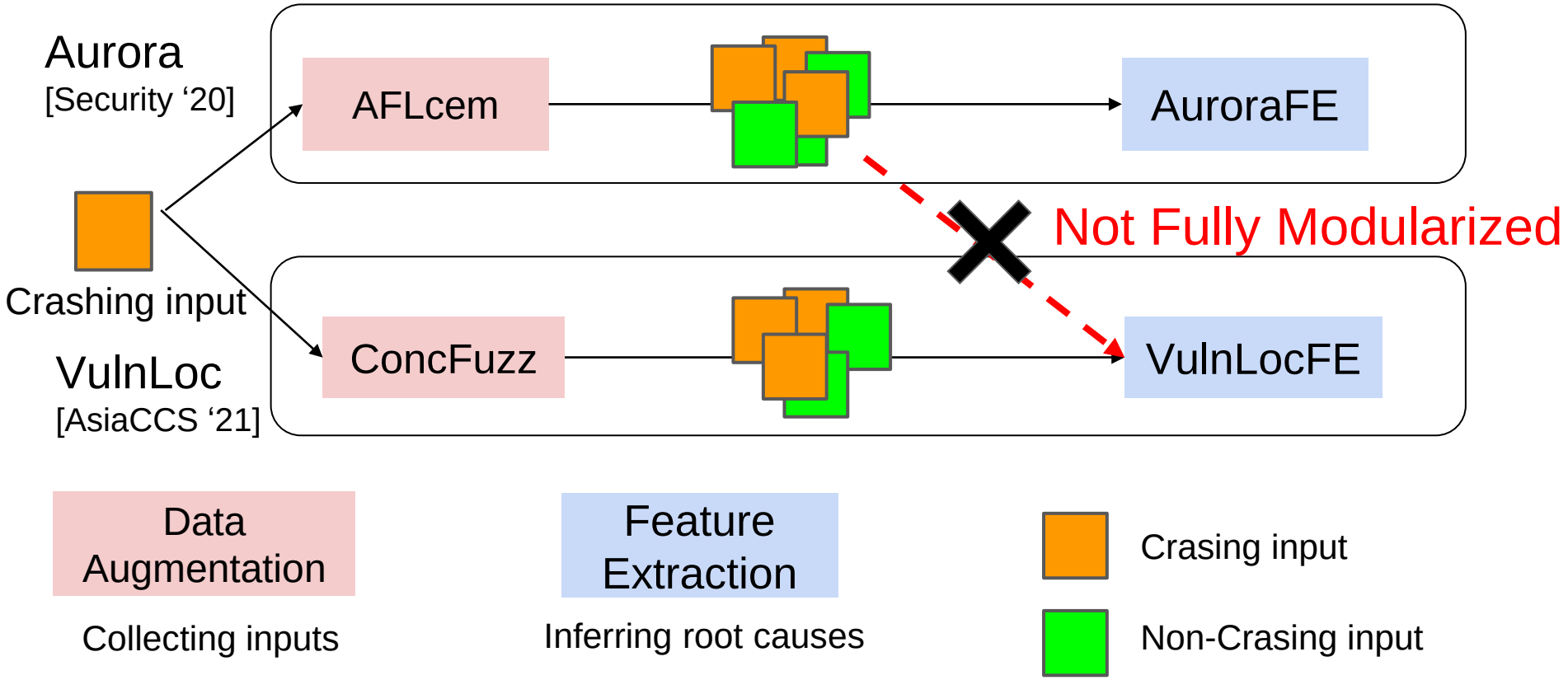
Feature Extraction

Inferring root causes

Crasing input

Non-Crasing input

Challenge #2: Tightly coupled RCA steps



Challenge #3: Variance of Data Augmentation

- **Data Augmentation (internal fuzzing) Time**

  Longer time = More inputs = More accurate ??

- **Initial seeds (crashing inputs) of Data Augmentation**

  Do seeds affect accuracy, like fuzzing *?

- **Randomness of Data Augmentation**

  Quality of generated dataset may change?

* A. Herrera, et al. “Seed selection for successful fuzzing,” ISSTA ’21

RCABench: Open Benchmarking Platform for RCA

Existing challenges

1. Non-uniqueness RC
2. Tightly coupled RCA steps
3. Variance of D.A.

RCABench supports:

Predefined public RC locations

Decoupled RCA steps

Variance-aware evaluation

RCABench: Open Benchmarking Platform for RCA

7 real-world bugs/vulnerabilities with predefined public RC

from VulnLoc/Aurora's evaluations
But no public RC…

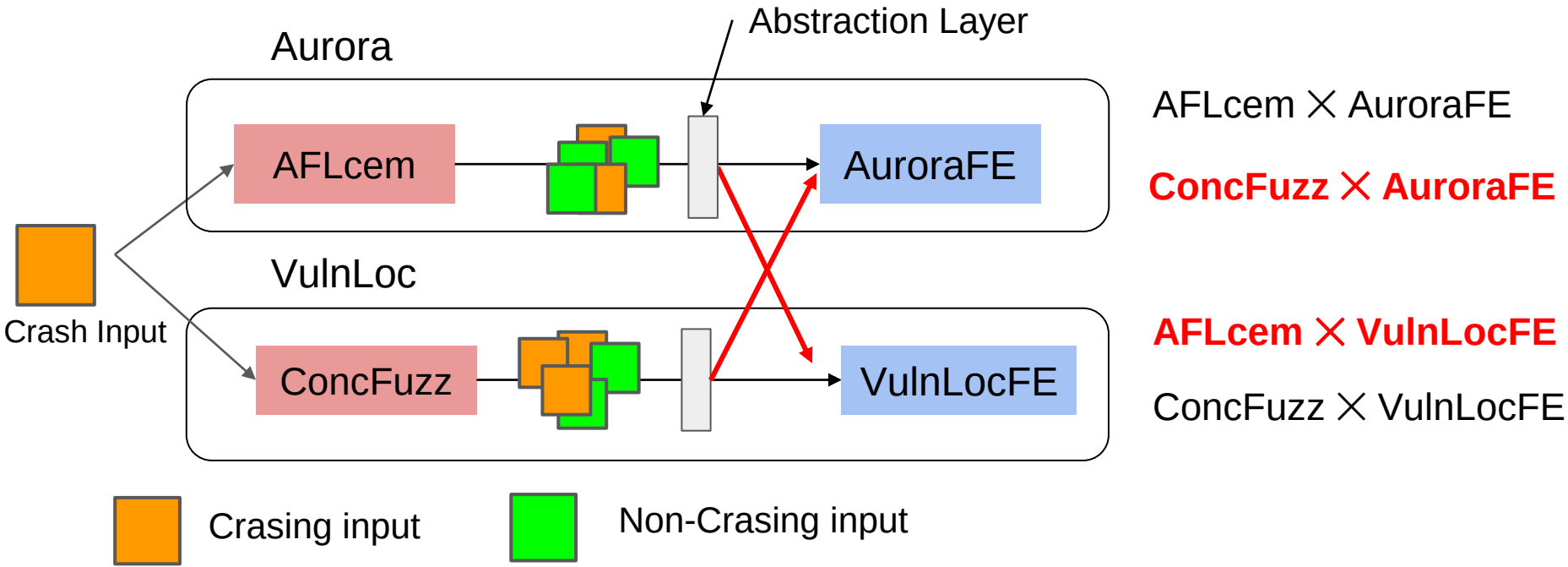
	Program	CVE ID	Root Cause	Crash Cause
#1	LibTIFF	CVE-2016-10094	off-by-one error	heap buffer overflow
#2	Libjpeg	CVE-2018-19664	incomplete check	heap buffer overflow
#3	Libjpeg	CVE-2017-15232	missing check	null pointer dereference
#4	Libxml2	CVE-2017-5969	incomplete check	null pointer dereference
#5	mruby	None	missing check	type confusion
#6	readelf	CVE-2019-9077	missing check	heap buffer overflow
#7	Lua	CVE-2019-6706	missing check	use-after-free

Target #5 was not assigned a CVE ID but was assigned ID 185041 in the HackerOne platform.

We plan to add more targets….

RCABench: Open Benchmarking Platform for RCA

Decoupling and modularization of D.A and F.E.



RCABench: Open Benchmarking Platform for RCA

Supporting variance-aware evaluation

- Multiple initial crashing inputs for some targets
- Multiple Data Augmentation times
- Configuration based easy multiple benchmarking

Results of RCABench

RQ1: Which RCA technique is most accurate?

RQ2: Does D.A. time length affect accuracy?

RQ3: Do initial seeds affect accuracy?

RQ4: Does the randomness of D.A. affect accuracy?

Please see the paper for the detail 😀

RQ1: Which RCA technique is most accurate?

RCA techniques:

- AFLcem x AuroraFE = Aurora[Security '20]
- ConcFuzz x VulnLocFE = VulnLoc[AsiaCCS '21]
- AFLcem x VulnLocFE
- ConcFuzz x AuroraFE

} Newly tested

	Program	D.A. Time	A × A	C × A	A × V	C × V
#1	LibTIFF	15 m	15	9	2	13
		2 h	9	33	2	12
		4 h	9	47	2	12
#2	Libjpeg	15 m	–	–	–	32
		2 h	–	15	–	23
		4 h	–	14	–	17
#3	Libjpeg	15 m	22	–	6	1
		2 h	10	–	6	1
		4 h	9	–	6	1
#4	Libxml2	15 m	28	–	57	82
		2 h	29	–	19	83
		4 h	28	–	19	89
#5	mruby	15 m	29	94	–	46
		4 h	27	71	–	45
		12 h	25	74	–	45
#6	readelf	15 m	1	4	4	4
		2 h	1	1	4	4
		4 h	1	1	4	4
#7	Lua	15 m	–	–	–	1
		4 h	–	N/A	–	N/A
		12 h	32	N/A	–	N/A

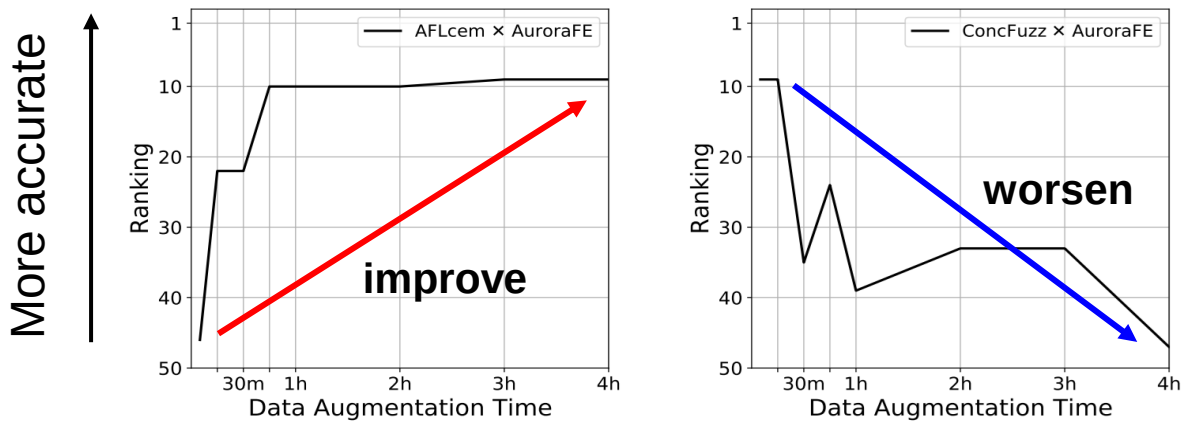
Answer:

There was no obviously universal technique that was most accurate for all targets.

RQ2: Does D.A. time length affect accuracy?

Answer:

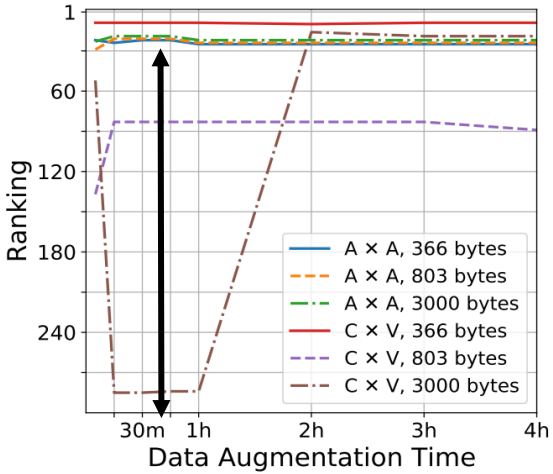
- **Accuracy improved or did not change over time in many cases.**
- **There were a few cases in which the accuracy was degraded.**



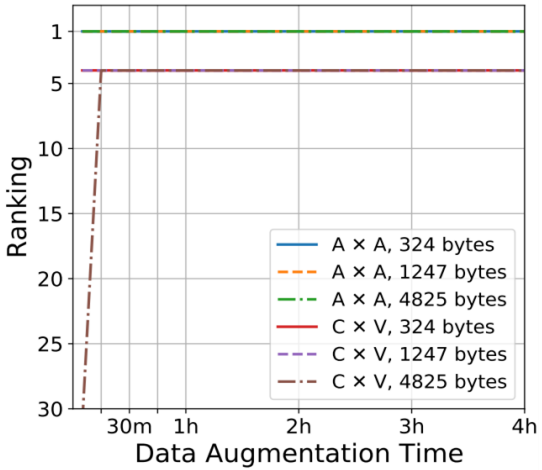
(a) Target #3 (b) Target #1

RQ3: Do initial seeds affect accuracy?

Answer: Initial seeds sometimes affect accuracy.



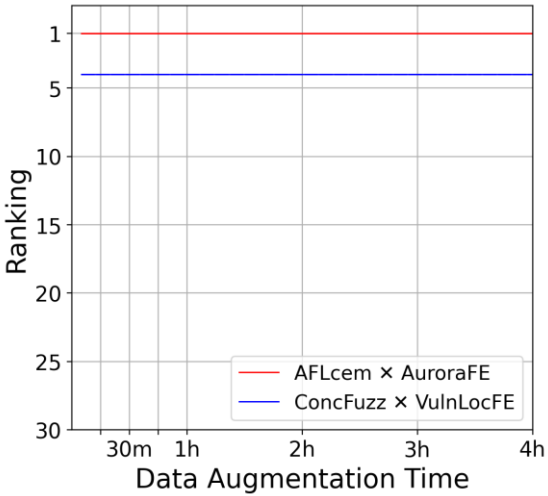
(a) Target #4



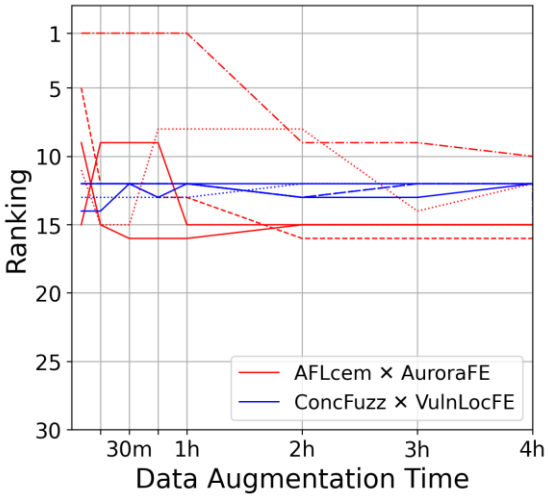
(b) Target #6

RQ4: Does the randomness of D.A. affect accuracy?

Answer: Randomness in DA can lead to non-negligible variances in accuracy.



Target #6



Target #1

Limitations and future work

- **Mores statistical evaluation considering randomness**
  - D.A. randomness affected the RCA results (RQ4).
  - This threatens the validity of previous RCA evaluations.

- **More abundant targets with diverse root causes**
  - We plan to add more diverse targets
    - Fuzzing benchmark (Magma, FuzzBench…)
    - Real-world vulnerabilities

Conclusion

- Motivation: Evaluation of RCA techniques are challenging
- **RCABench (end-to-end benchmarking platform)**
  - Predefined and public root cause locations for seven targets
  - Decoupling RCA steps (D.A. and F.E)
  - Variance-aware evaluation for Data Augmentation
    (DA time/initial seed/fuzzing randomness)

https://github.com/RICSecLab/RCABench

Limitations and future work

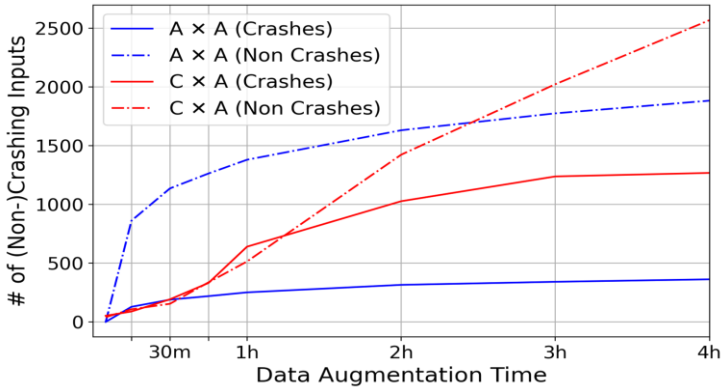
- **Modular framework for fair and objective RCA evaluation**
  - Implementation differences can spoil fair comparisons.
    - Tracing: Intel PIN, DynamoRIO…
    - Language: Python, C++
    - Misc: parallelization, file I/O, log…
  - Basic Blocks for implementation is needed.
    c.f. modular framework for fuzzing [LibAFL, fuzzuf]

LibAFL: https://github.com/AFLplusplus/LibAFL fuzzuf: https://github.com/fuzzuf/fuzzuf

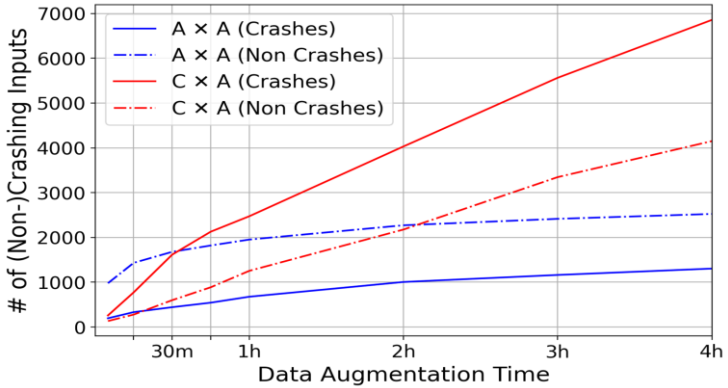
What affects the quality of the DA's results?

- Number of inputs
- Ratio of crashing/non-crashing inputs

} depends on combinations of targets and methods.



Target #1



Target #6

Target selection

- Diverse Root Cause (Missing check, Incomplete check)
- Diverse crash causes (heap overflow, UAF …)
- Real-world software

Any contributions are welcome.

Question: What about targets with poor accuracy?

if statement at the patch point is executed regardless of the value of `count`.

```
  if(TIFFGetField(input, TIFFTAG_JPEGTABLES, &count, &jpt) != 0) {
-   if (count >= 4) {
+   if (count > 4) {
      int retTIFFReadRawTile;
      _TIFFmemcpy(buffer, jpt, count - 2);
```

Target #1: CVE-2016-10094

More precise evaluation for randomness

- Average of rankings
- User’s perspective
  - Is 1000 candidates of RC practical
  - Internal thresholds to reduce the output cadidates

- More fundamental solution such as formalization is needed.

Question: Number of figures

	Program	D.A. Time	A × A	C × A	A × V	C × V
#1	LibTIFF	15 m	15	9	2	13
		2 h	9	33	2	12
		4 h	9	47	2	12
#2	Libjpeg	15 m	–	–	–	32
		2 h	–	15	–	23
		4 h	–	14	–	17
#3	Libjpeg	15 m	22	–	6	1
		2 h	10	–	6	1
		4 h	9	–	6	1
#4	Libxml2	15 m	28	–	57	82
		2 h	29	–	19	83
		4 h	28	–	19	89
#5	mruby	15 m	29	94	–	46
		4 h	27	71	–	45
		12 h	25	74	–	45
#6	readelf	15 m	1	4	4	4
		2 h	1	1	4	4
		4 h	1	1	4	4
#7	Lua	15 m	–	–	–	1
		4 h	–	N/A	–	N/A
		12 h	32	N/A	–	N/A

- RCA techniques shows the candidates of root causes ordered **by the level of confidence.**
- The number means the ranks of the actual root cause we defined.

Candidate

1. a.c:100
2. b.c:200 ← Rank 2
3. a.c:105

Root cause

b.c:200
a.c:500

Non-uniqueness of root cause location

Multiple possibilities of root causes locations

```
function buggy(size) {
  data = malloc(size);
  return data;
}

function crashable(idx) {
  data = buggy(size);
  data[idx] = 1;
}
```

Patch 1 in function buggy()

```
if (data == NULL) {
  exception();
}
```

Missing check of malloc()'s return value

Patch 2 in function crashable()

```
if (data == NULL) {
  exception();
}
```

Missing check of buggy()'s return value