Domain-Specific Fuzz Testing

Rohan Padhye

Homepage: rohan.padhye.org
Email: rohanpadhye@cmu.edu
Twitter: @moarbugs

Berkeley UNIVERSITY OF CALIFORNIA

Carnegie Mellon University

isr institute for SOFTWARE RESEARCH

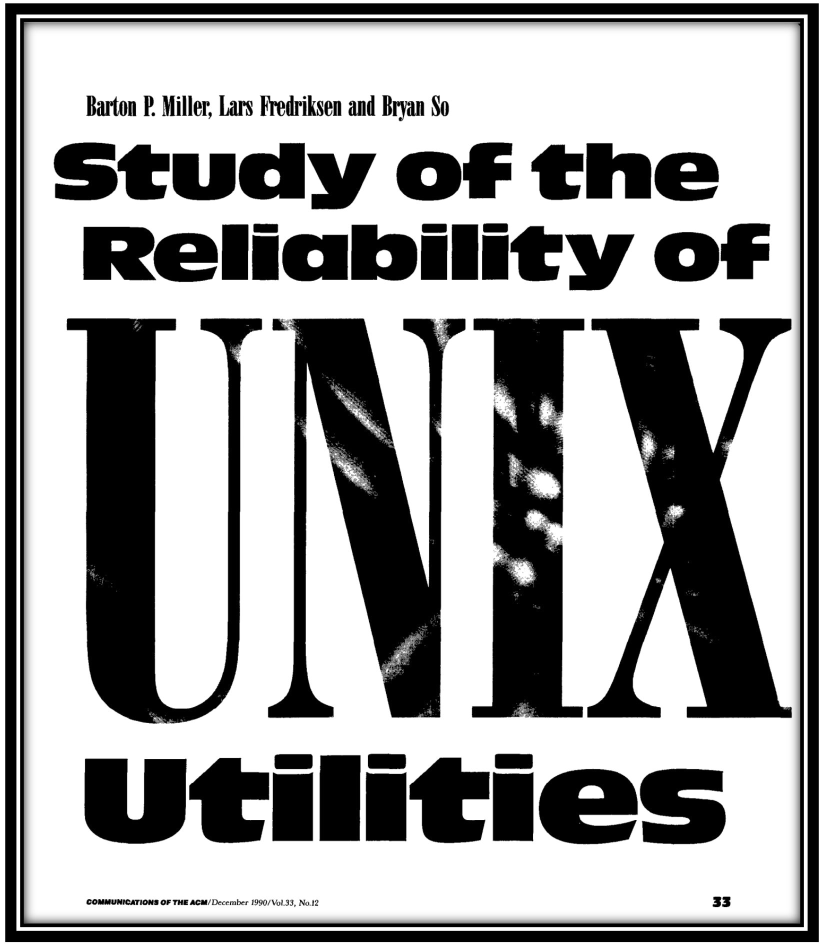
Carnegie Mellon University Security and Privacy Institute CyLab

How can we find dormant software bugs
(+ security vulnerabilities) in real software?

Fuzz Testing

Generate inputs randomly until program crashes

Fuzz Testing



Barton P. Miller, Lars Fredriksen and Bryan So

Study of the Reliability of UNIX Utilities

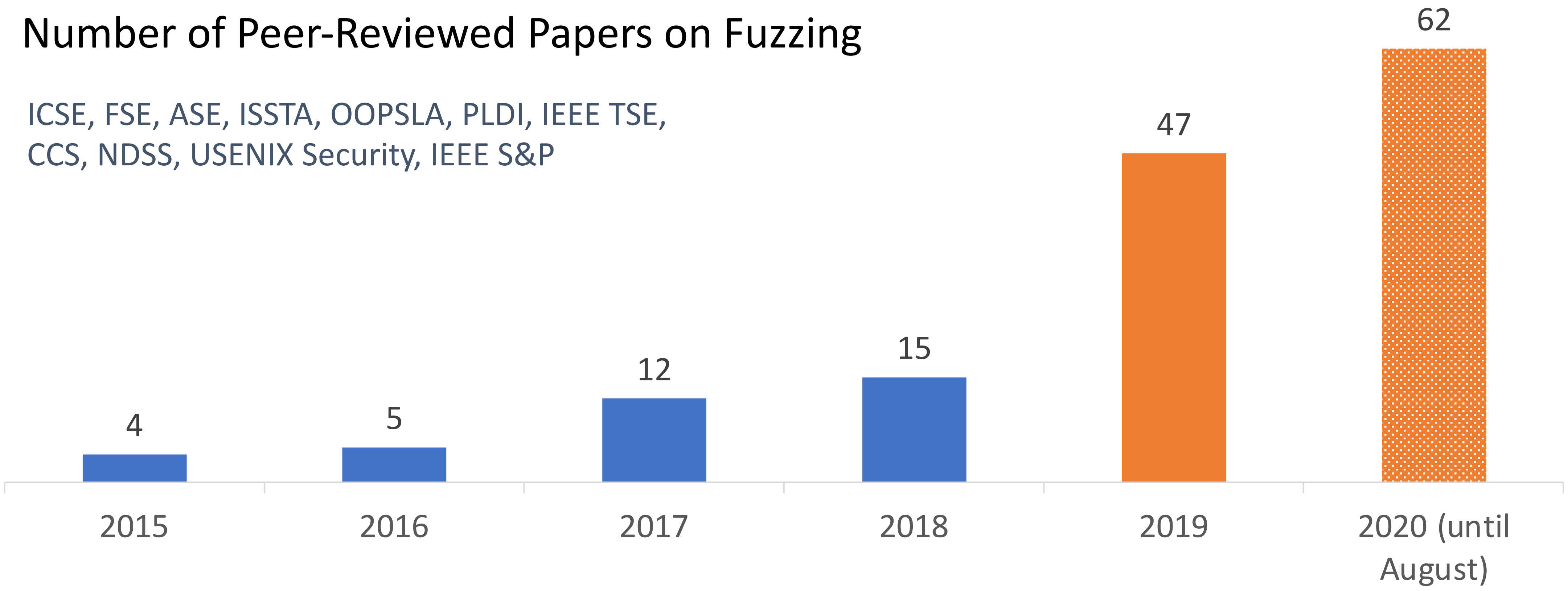
COMMUNICATIONS OF THE ACM/December 1990/Vol.33, No.12

33

Communications of the ACM (1990)

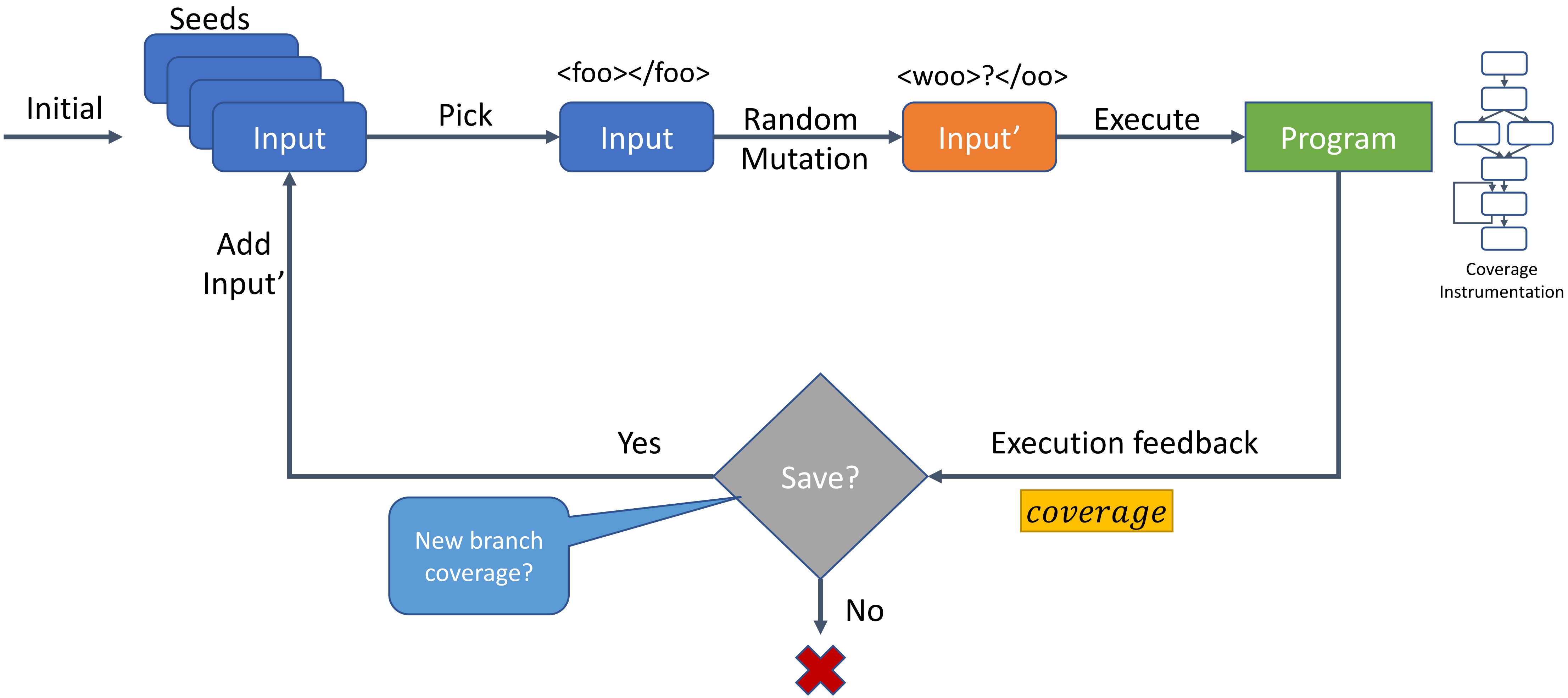
Crashed:
adb, as, bc, cb, col, diction, emacs, eqn, ftp, indent, lex, look, m4, make, nroff, plot, prolog, ptx, refer, spell, style, tsort, uniq, vgrind, vi

Fuzz Testing 30 years on…

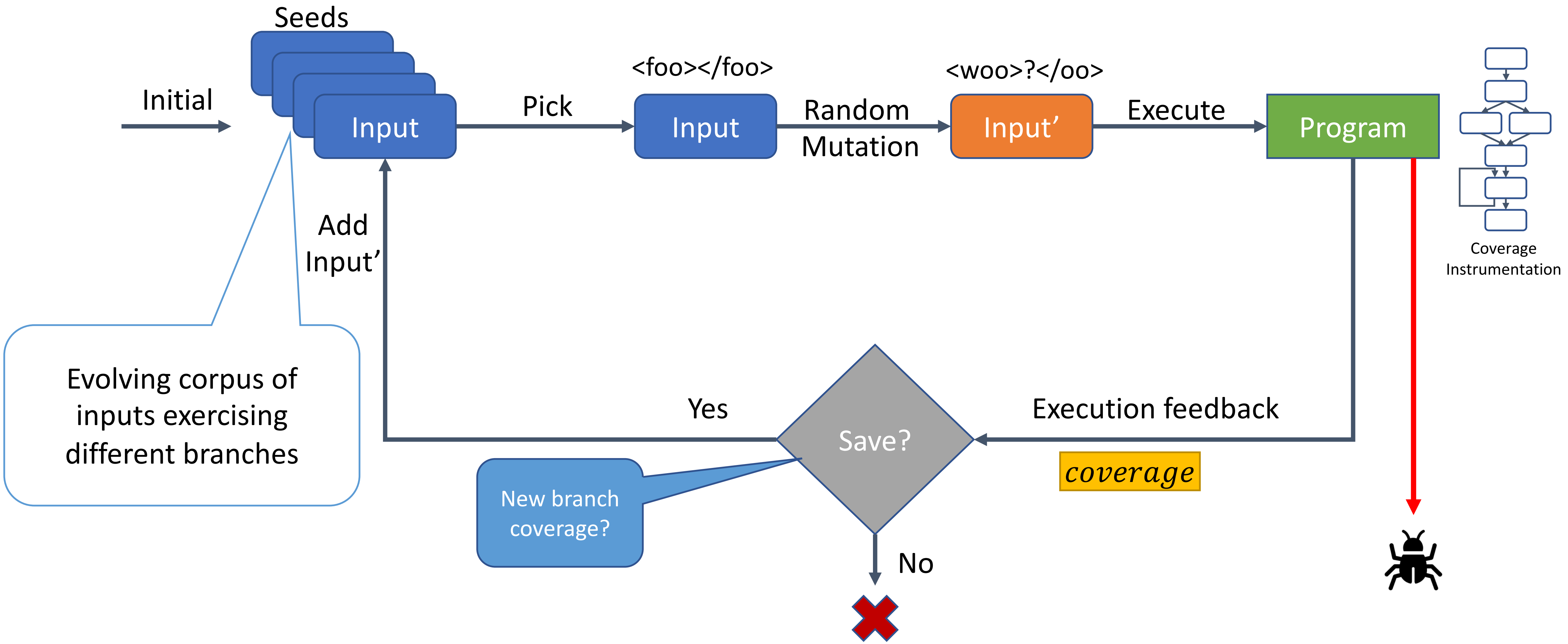


Data source: https://wcventure.github.io/FuzzingPaper

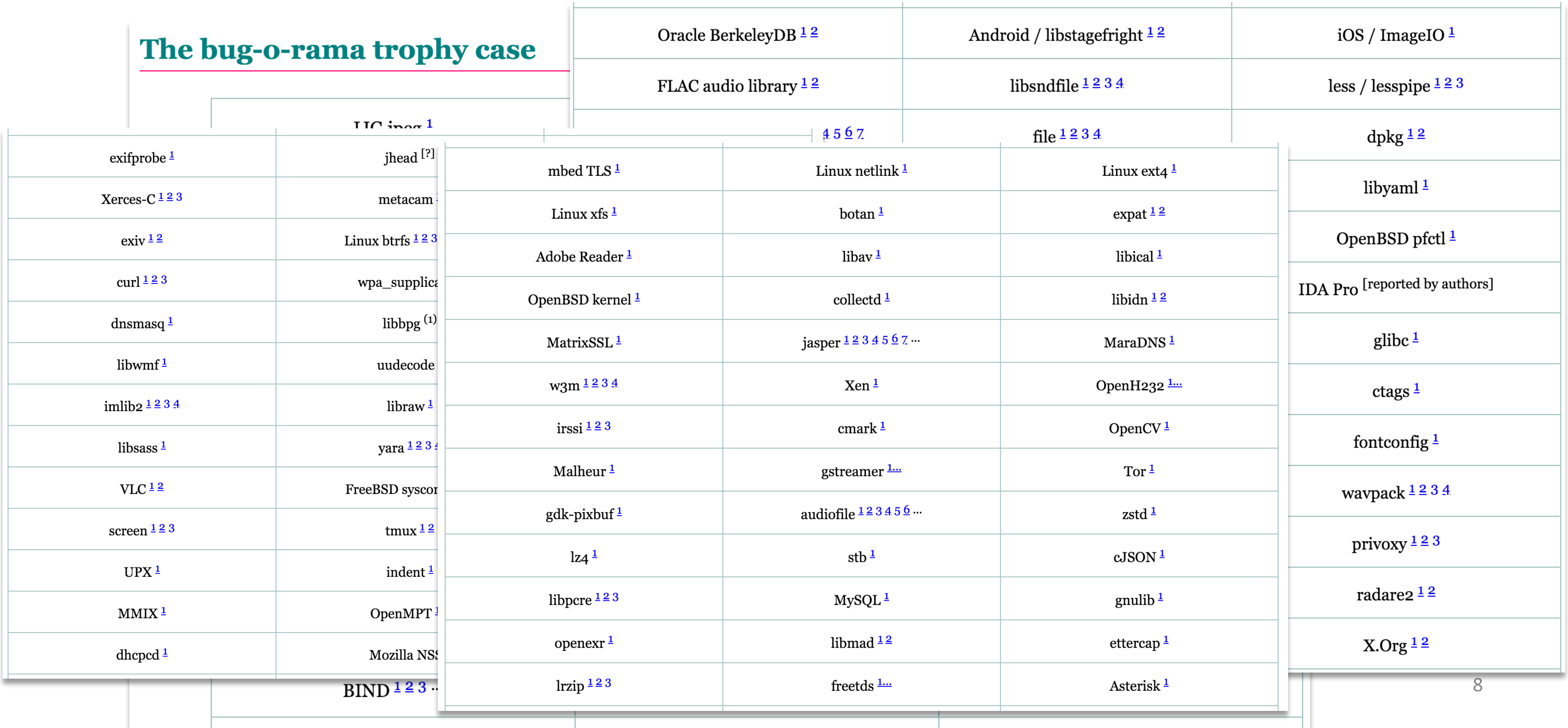
Coverage-Guided Fuzzing (e.g. AFL, libFuzzer)



Coverage-Guided Fuzzing (e.g. AFL, libFuzzer)



Coverage-Guided Fuzzing with AFL



The bug-o-rama trophy case

IJG jpeg 1

Oracle BerkeleyDB 1 2	Android / libstagefright 1 2	iOS / ImageIO 1
FLAC audio library 1 2	libsndfile 1 2 3 4	less / lesspipe 1 2 3
4 5 6 7	file 1 2 3 4	dpkg 1 2
		libyaml 1
		OpenBSD pfctl 1
		IDA Pro [reported by authors]
		glibc 1
		ctags 1
		fontconfig 1
		wavpack 1 2 3 4
		privoxy 1 2 3
		radare2 1 2
		X.Org 1 2

exifprobe 1	jhead [?]
Xerces-C 1 2 3	metacam
exiv 1 2	Linux btrfs 1 2 3
curl 1 2 3	wpa_supplica
dnsmasq 1	libbpg (1)
libwmf 1	uudecode
imlib2 1 2 3 4	libraw 1
libsass 1	yara 1 2 3
VLC 1 2	FreeBSD syscor
screen 1 2 3	tmux 1 2
UPX 1	indent 1
MMIX 1	OpenMPT
dhcpcd 1	Mozilla NSS

BIND 1 2 3 ...

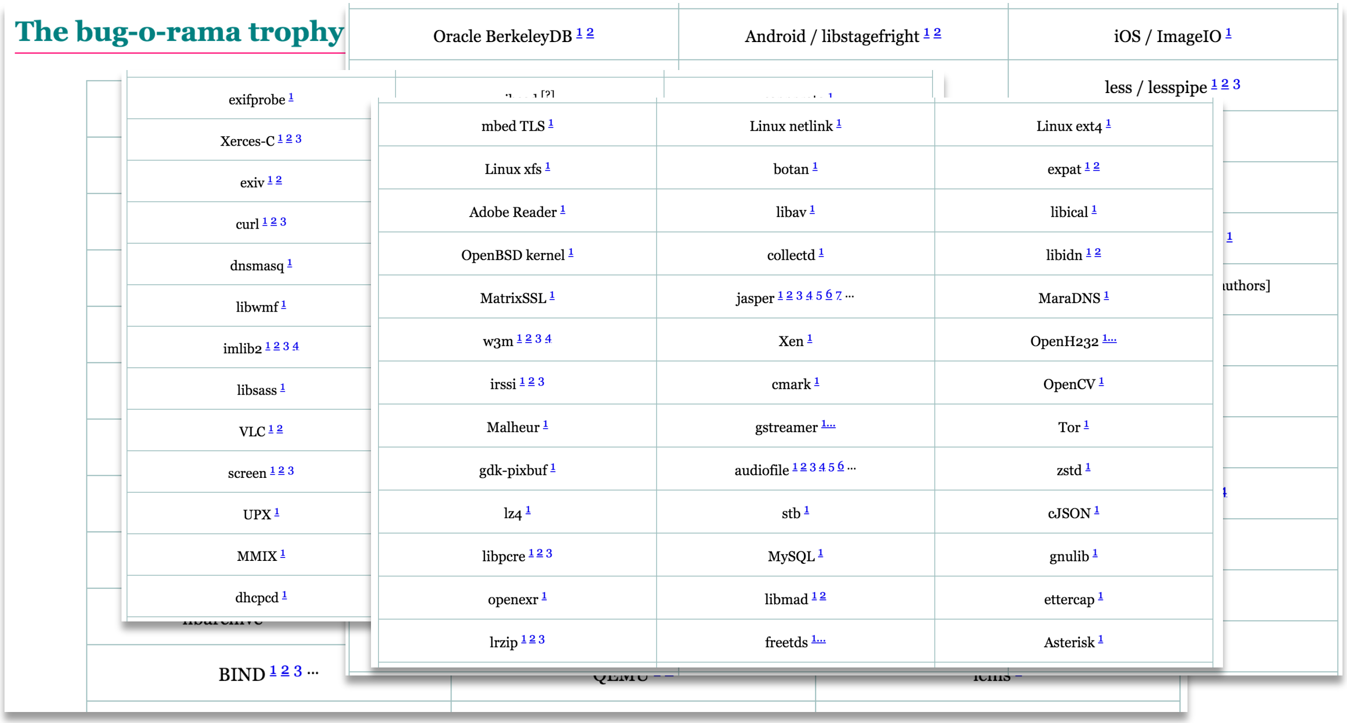
mbed TLS 1	Linux netlink 1	Linux ext4 1
Linux xfs 1	botan 1	expat 1 2
Adobe Reader 1	libav 1	libical 1
OpenBSD kernel 1	collectd 1	libidn 1 2
MatrixSSL 1	jasper 1 2 3 4 5 6 7 ...	MaraDNS 1
w3m 1 2 3 4	Xen 1	OpenH232 1...
irssi 1 2 3	cmark 1	OpenCV 1
Malheur 1	gstreamer 1...	Tor 1
gdk-pixbuf 1	audiofile 1 2 3 4 5 6 ...	zstd 1
lz4 1	stb 1	cJSON 1
libpcre 1 2 3	MySQL 1	gnulib 1
openexr 1	libmad 1 2	ettercap 1
lrzip 1 2 3	freetds 1...	Asterisk 1

Limited class of programs:

- ✓ Binary data decoders
- ✓ Parsers of text formats

Limited class of bugs:

- ✓ Invalid input validation (buffer overflows, segfaults, div-by-zero, null ptr)



The bug-o-rama trophy

Oracle BerkeleyDB 1 2	Android / libstagefright 1 2	iOS / ImageIO 1
		less / lesspipe 1 2 3

exifprobe 1
Xerces-C 1 2 3
exiv 1 2
curl 1 2 3
dnsmasq 1
libwmf 1
imlib2 1 2 3 4
libsass 1
VLC 1 2
screen 1 2 3
UPX 1
MMIX 1
dhcpcd 1

mbed TLS 1	Linux netlink 1	Linux ext4 1
Linux xfs 1	botan 1	expat 1 2
Adobe Reader 1	libav 1	libical 1
OpenBSD kernel 1	collectd 1	libidn 1 2
MatrixSSL 1	jasper 1 2 3 4 5 6 7 ...	MaraDNS 1
w3m 1 2 3 4	Xen 1	OpenH232 1...
irssi 1 2 3	cmark 1	OpenCV 1
Malheur 1	gstreamer 1...	Tor 1
gdk-pixbuf 1	audiofile 1 2 3 4 5 6 ...	zstd 1
lz4 1	stb 1	cJSON 1
libpcre 1 2 3	MySQL 1	gnulib 1
openexr 1	libmad 1 2	ettercap 1
lrzip 1 2 3	freetds 1...	Asterisk 1

BIND 1 2 3 ...

Limited class of programs:

- ✓ Binary data decoders
- ✓ Parsers of text formats

Limited class of bugs:

- ✓ Invalid input validation (buffer overflows, segfaults, div-by-zero, null ptr)

New classes of issues?

Semantics-related

compiler optimization failures, config issues

Performance-related

worst-case complexity, memory usage

Domain-specific

side-channel leaks, privacy violations, object detection failures

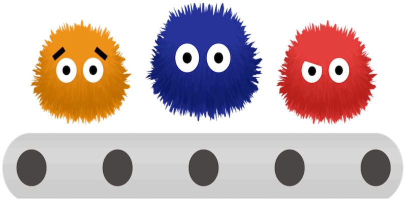
New classes of issues?

Semantics-related

compiler optimization failures, config issues

Performance-related

worst-case complexity, memory usage



Domain-specific

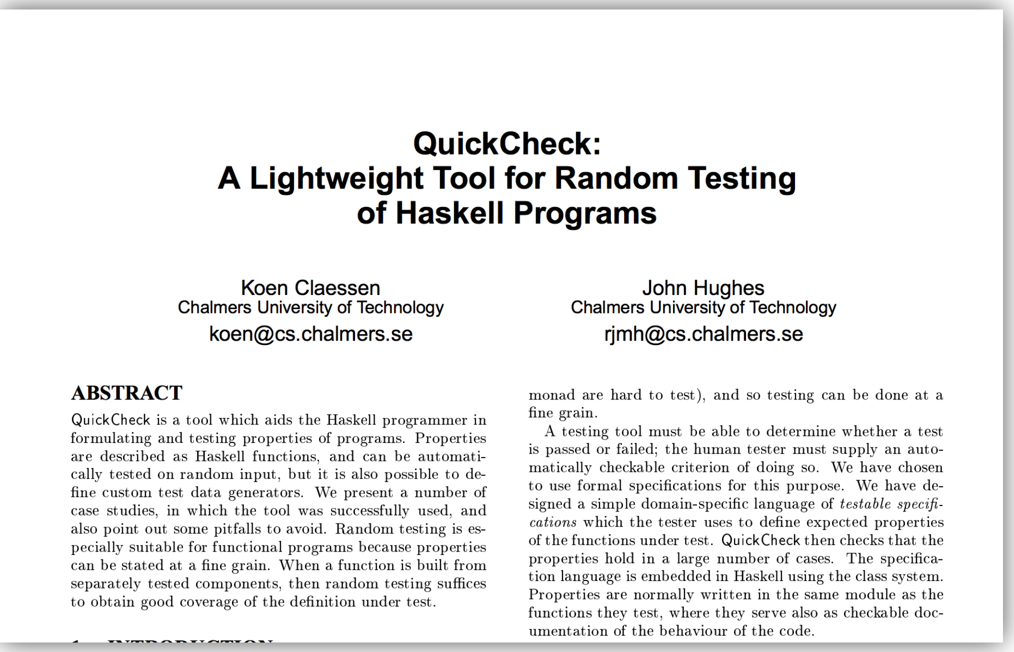
side-channel leaks, privacy violations, object detection failures

Key insight: Transform search space using domain knowledge

JQF + Zest	PerfFuzz	FuzzFactory

How can we test programs that expect inputs with complex structure and semantics?

JQF + Zest
[ISSTA’19]



QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs

Koen Claessen
Chalmers University of Technology
koen@cs.chalmers.se

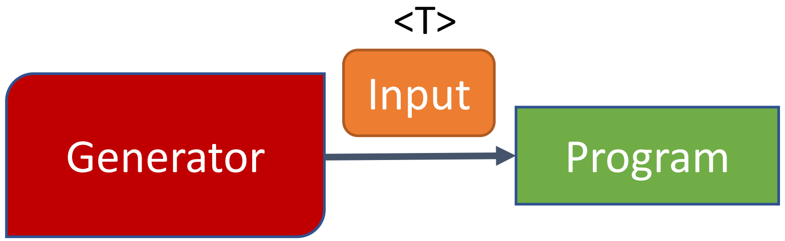
John Hughes
Chalmers University of Technology
rjmh@cs.chalmers.se

ABSTRACT

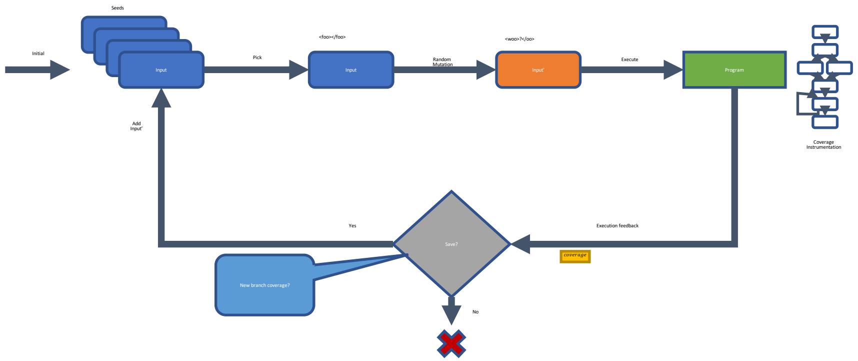
QuickCheck is a tool which aids the Haskell programmer in formulating and testing properties of programs. Properties are described as Haskell functions, and can be automatically tested on random input, but it is also possible to define custom test data generators. We present a number of case studies, in which the tool was successfully used, and also point out some pitfalls to avoid. Random testing is especially suitable for functional programs because properties can be stated at a fine grain. When a function is built from separately tested components, then random testing suffices to obtain good coverage of the definition under test.

monad are hard to test), and so testing can be done at a fine grain.

A testing tool must be able to determine whether a test is passed or failed; the human tester must supply an automatically checkable criterion of doing so. We have chosen to use formal specifications for this purpose. We have designed a simple domain-specific language of *testable specifications* which the tester uses to define expected properties of the functions under test. QuickCheck then checks that the properties hold in a large number of cases. The specification language is embedded in Haskell using the class system. Properties are normally written in the same module as the functions they test, where they serve also as checkable documentation of the behaviour of the code.

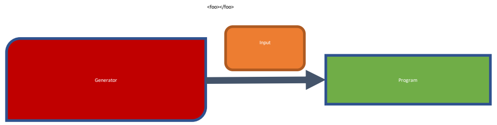


Sampling procedure for inputs of type <T>



Coverage-guided Fuzzing

+



Generator-based Fuzzing

+

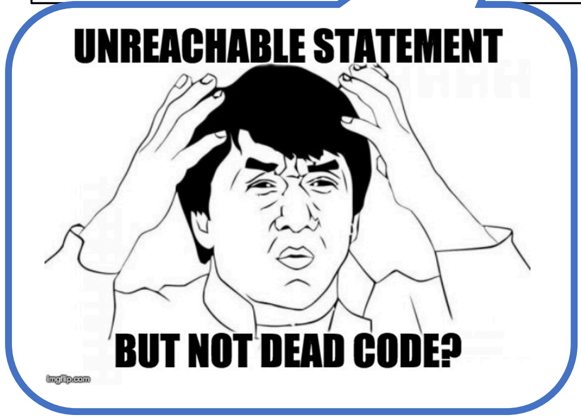
=

JQF + Zest

JQF + Zest: Compiler Testing

```
while ((l_0)){
  while ((l_0)){
    if ((l_0))
    { break;var l_0;continue }
    { break;var l_0 }
  }
}
```

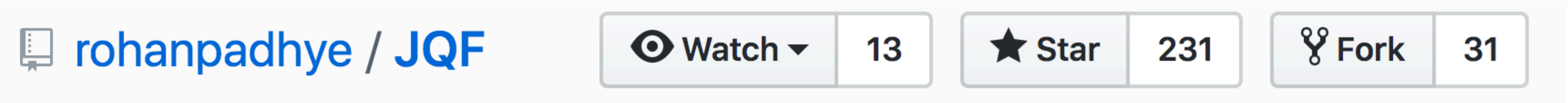
Program generated with **Zest**



Google Closure Compiler

(250k LoC, JavaScript optimizer)

Incorrect Dead-Code Elimination



rohanpadhye / JQF

JQF + Zest: Bug Trophy Case

- **Google Closure Compiler**: #2842, #2843, #3220, #3173
- **OpenJDK**: JDK-8190332, JDK-8190511, JDK-8190512, JDK-8190997, JDK-8191023, JDK-8191076, JDK-8191109, JDK-8191174, JDK-8191073, JDK-8193444, JDK-8193877
- **Apache Ant**: #62655
- **Apache Maven**: MNG-6374, MNG-6375, MNG-6577
- **Apache Commons**: LANG-1385, COMPRESS-424, COLLECTIONS-714, **CVE-2018-11771**
- **Apache PDFBox**: PDFBOX-4333, PDFBOX-4338, PDFBOX-4339, **CVE-2018-8036**
- **Apache TIKA**: **CVE-2018-8017**, **CVE-2018-12418**
- **Apache BCEL**: BCEL-303, BCEL-307, BCEL-308, BCEL-309, BCEL-310, BCEL-311, BCEL-312, BCEL-313
- **Mozilla Rhino**: #405, #406, #407, #409, #410

Found by
OSS community
+ industry

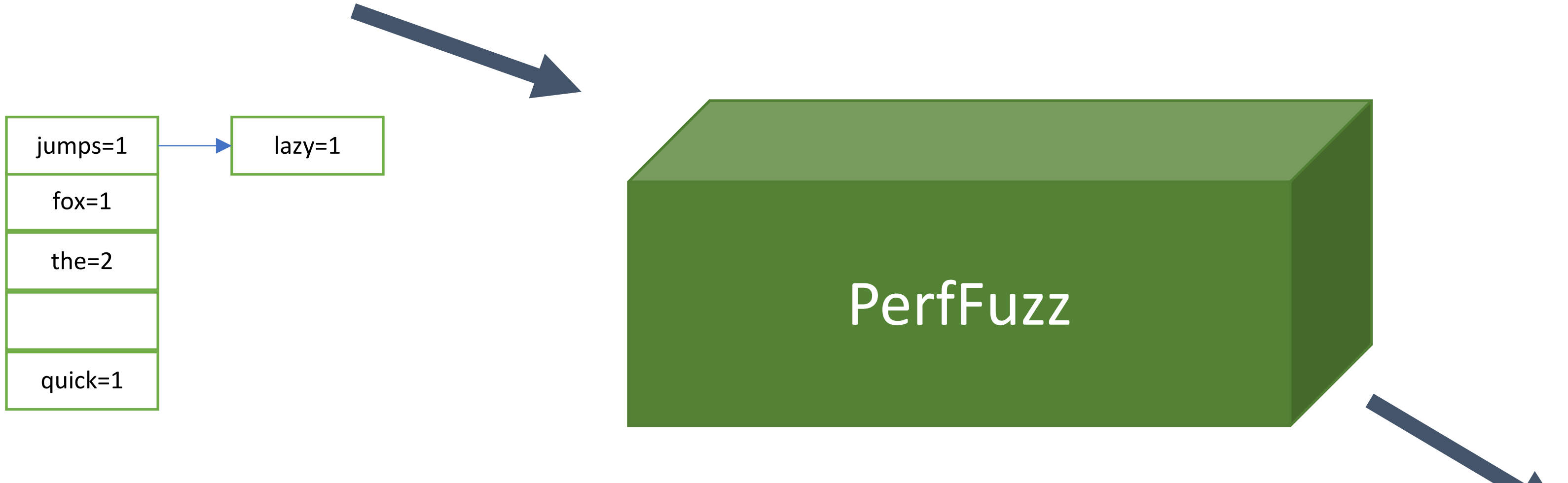
Officially supported by **GitLab** – CI Workflow

Can we use existing functional test cases to find algorithmic performance issues?

PerfFuzz
[ISSTA’18]

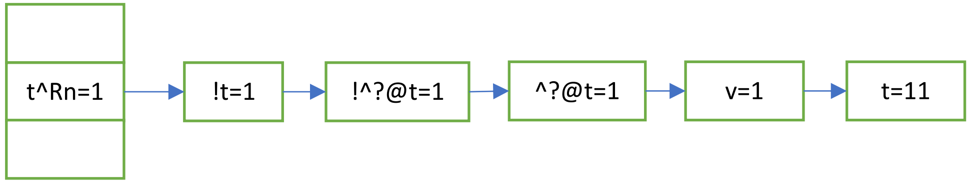
PerfFuzz automatically synthesizes pathological inputs

Input:
the quick brown fox jumps over the lazy dog

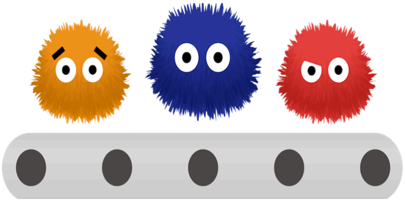


New Input:

“t v ^?@t !^?@t !t t^Rn t t t t t t t t t"

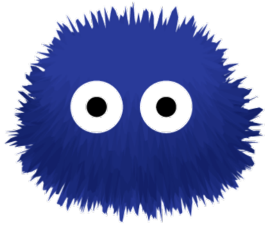


How can we rapidly **create** and **combine** custom fuzzing applications?



FuzzFactory
[OOPSLA'19]

FuzzFactory enables **rapid protoyping** and **composition**



Loc	Hamming
…	…
Line 42	0
…	…

CMP

Discovers magic constants / checksums

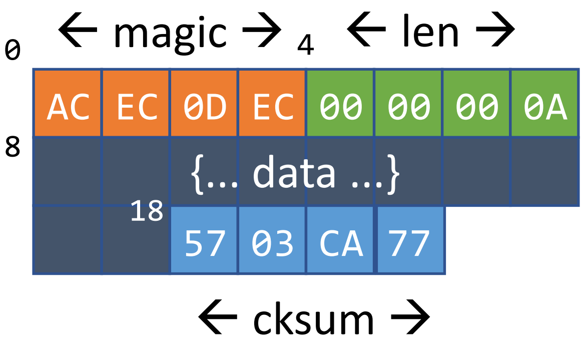
∘

(compose)

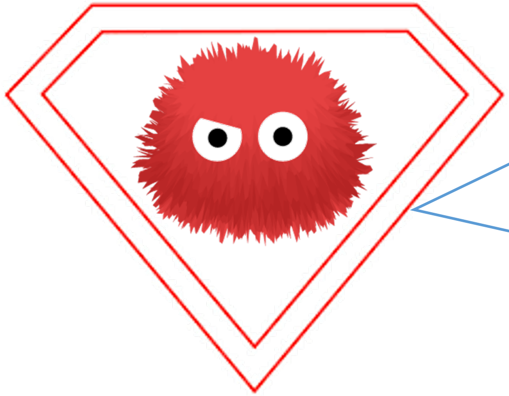
Loc	Bytes Allocated
…	…
Line 44	4294967296
…	…

MEM

Exacerbates malloc()s



AC	EC	0D	EC	00	00	00	0A
57	03	CA	77				



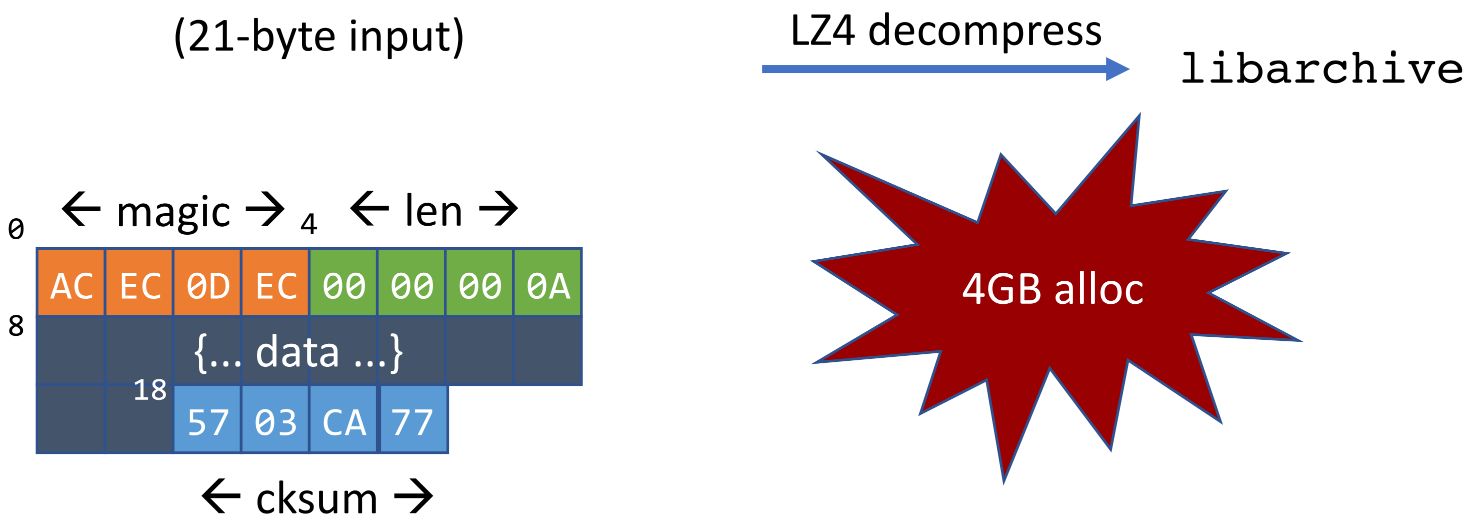
CMP-MEM

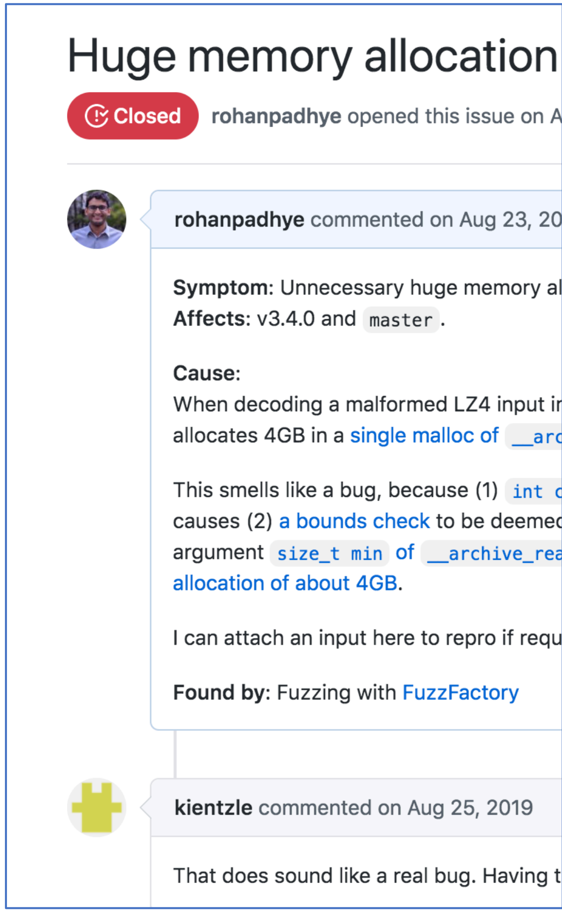
Exacerbates malloc()s while satisfying checksums

```
42: if (hash(data) == cksum) {
      while (i++ < count) {
44:     p = malloc(size)
        ...
      }
    }
```

Super-Fuzzer: CMP-MEM

Compression bombs!





Huge memory allocation

Closed rohanpadhye opened this issue on A

rohanpadhye commented on Aug 23, 20

Symptom: Unnecessary huge memory al
Affects: v3.4.0 and `master`.

Cause:
When decoding a malformed LZ4 input in
allocates 4GB in a single malloc of `__arc`

This smells like a bug, because (1) `int c`
causes (2) a bounds check to be deemed
argument `size_t min` of `__archive_rea`
allocation of about 4GB.

I can attach an input here to repro if requ

Found by: Fuzzing with FuzzFactory

kientzle commented on Aug 25, 2019

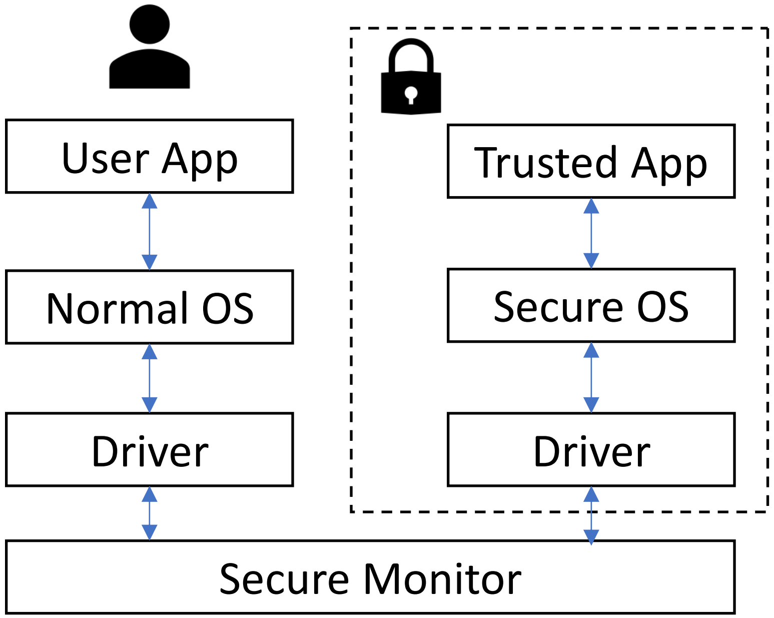
That does sound like a real bug. Having t

Fuzzing Trusted Execution Environments

SAMSUNG
RESEARCH AMERICA

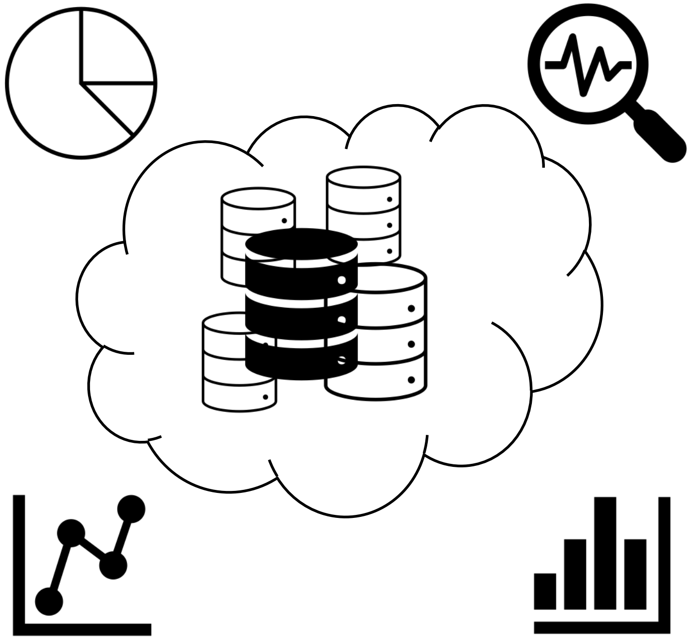
PartEmu
Full-System Emulation
+ FuzzFactory
=
48 Security Vulnerabilities!

”PartEmu: Enabling Dynamic Analysis of Real-World TrustZone Software Using Emulation",
L. Harrison, H. Vijayakumar, R. Padhye, K.Sen, M. Grace.
USENIX Security 2020.



Fuzzing Big Data Analytics

BigFuzz
= Scaling JQF to big data
with Framework Abstraction



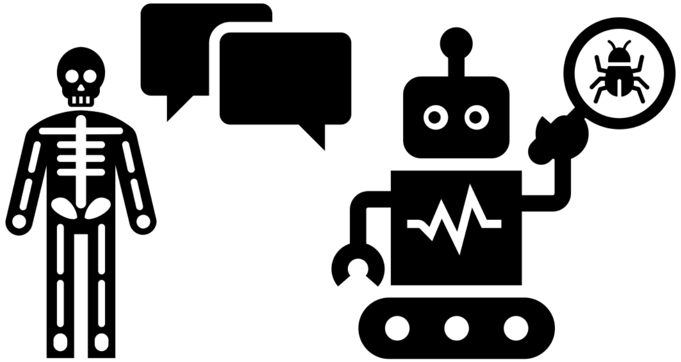
”BigFuzz: Efficient Fuzz Testing for Data Analytics using
Framework Abstraction",
Q. Zhang, J. Wang, M. Gulzar, R. Padhye, M. Kim
ASE 2020.

Open Questions / Opportunities

What are the best interaction models for program analysis tools?

How to easily target new domains?

What can we learn from surrounding context?



Rohan Padhye

Homepage: rohan.padhye.org
Email: rohanpadhye@cmu.edu
Twitter: @moarbugs