Three Colours of Fuzzing:
Reflections and Open Challenges

Cristian Cadar

Image credits: aitoff, jackmac34, Gam-Ol, JessBaileyDesign, No-longer-here, Walkerssk @ Pixabay, Powerpoint

Keynote @ FUZZING 2023
Seattle, USA, 17 July 2023

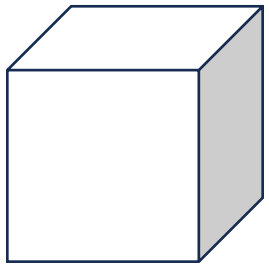
Execution Generated Test Cases: How to Make Systems Code Crash Itself

Cristian Cadar and Dawson Engler*

Computer Systems Laboratory,
Stanford University,
Stanford, CA 94305, U.S.A

Abstract. This paper presents a technique that uses code to automatically generate its own test cases at run-time by using a combination of symbolic and concrete (i.e., regular) execution. The input values to a

2005



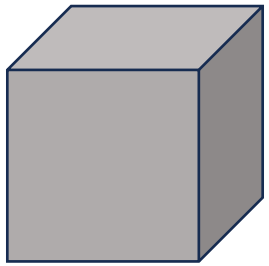
Running Symbolic Execution Forever

2020

Frank Busse
Imperial College London
United Kingdom
f.busse@imperial.ac.uk

Martin Nowack
Imperial College London
United Kingdom
m.nowack@imperial.ac.uk

Cristian Cadar
Imperial College London
United Kingdom
c.cadar@imperial.ac.uk

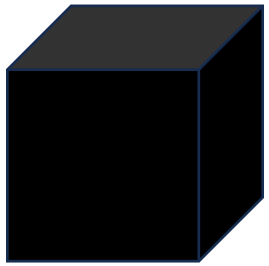


SnapFuzz: High-Throughput Fuzzing of Network Applications

Anastasios Andronidis
Imperial College London
London, United Kingdom
a.andronidis@imperial.ac.uk

Cristian Cadar
Imperial College London
London, United Kingdom
c.cadar@imperial.ac.uk

2022



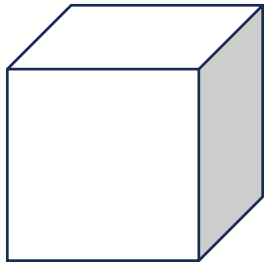
Grammar Mutation for Testing Input Parsers (Registered Report)

2023

Bachir Bendrissou
Imperial College London
London, United Kingdom
b.bendrissou@imperial.ac.uk

Cristian Cadar
Imperial College London
London, United Kingdom
c.cadar@imperial.ac.uk

Alastair F. Donaldson
Imperial College London
London, United Kingdom
alastair.donaldson@imperial.ac.uk

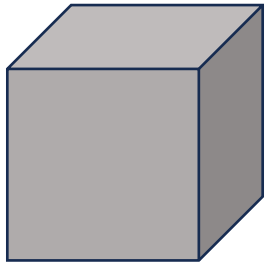


KLEE

Open-source tool widely used in both research and industry

Microsoft SAGE

Found one-third of file fuzzing bugs during development of Windows 7

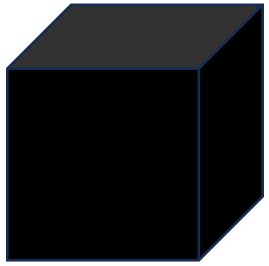


AFL

Revolutionary greybox fuzzer with a long list of trophies

Google's OSS-Fuzz

Fuzzing platform for OSS, found 8K+ vulnerabilities and 28K+ bugs in 850+ projects



Csmith and EMI

Compiler fuzzers, discovered hundreds of bugs in mature compilers like GCC & LLVM

SQLancer

DBMS fuzzer, found 400+ bugs in popular DBMS like SQLite & PostgreSQL

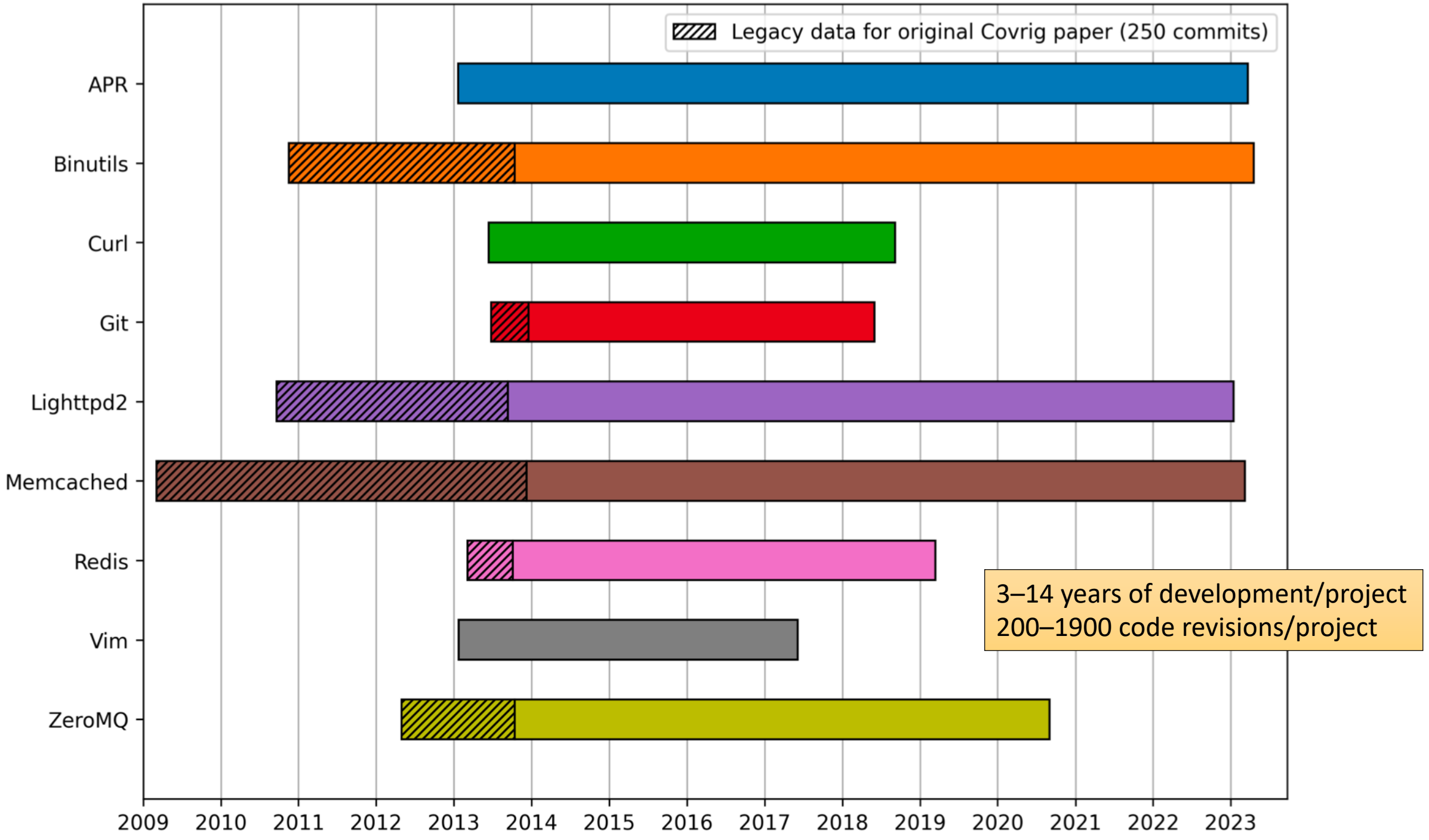
COVRIG: A Framework for the Analysis of Code, Test, and Coverage Evolution in Real Software

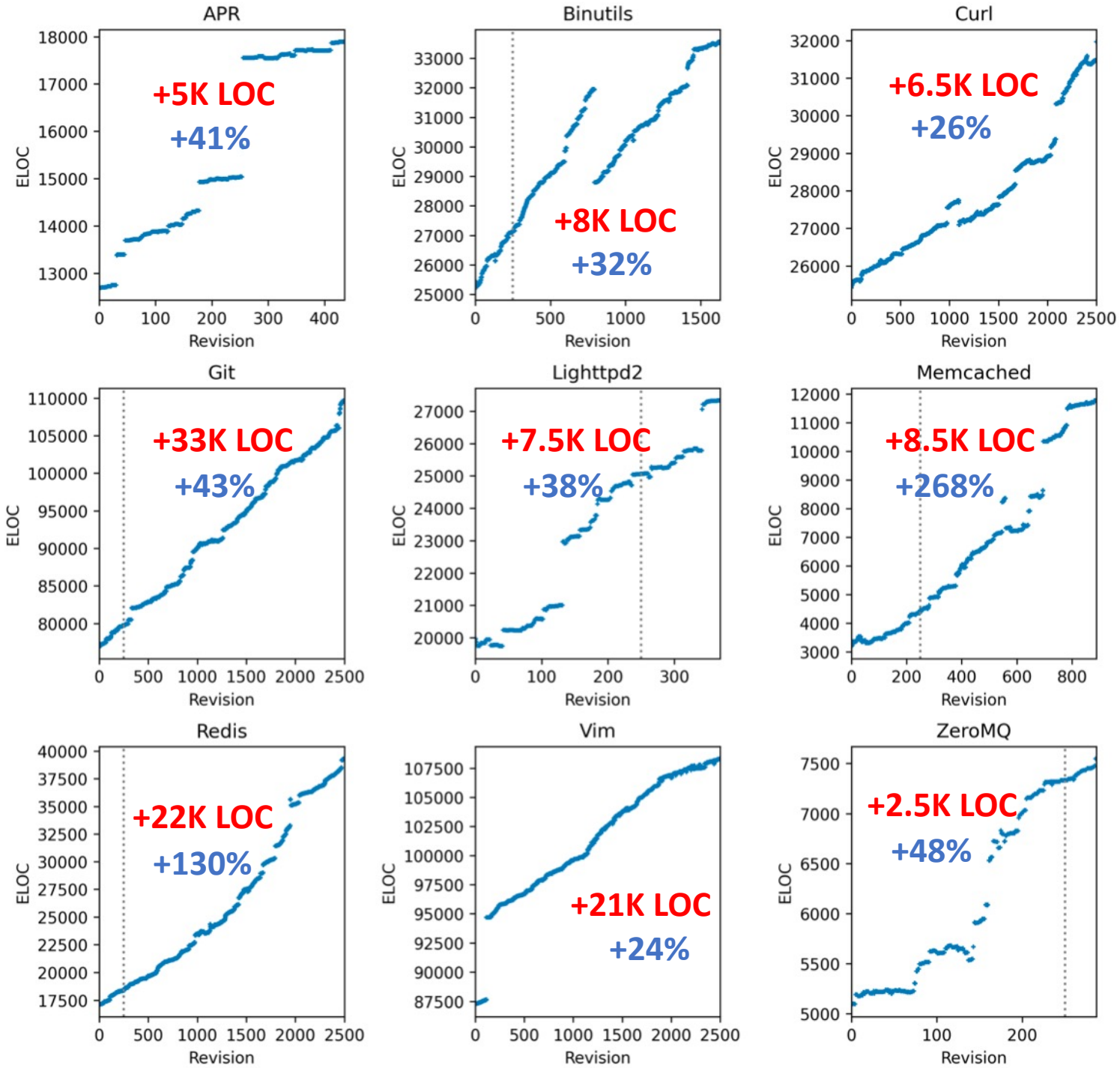
Paul Marinescu, Petr Hosek, Cristian Cadar
Department of Computing
Imperial College London, UK
{p.marinescu,p.hosek,c.cadar}@imperial.ac.uk

ISSTA 2014

- 6 popular open-source systems
- Analysed 250 revisions per app
- Conclusion: LOTS of code added or modified without being tested

A decade later: Have things changed?
Tom Bailey, C.C., WiP


Legacy data for original Covrig paper (250 commits)
APR
Binutils
Curl
Git
Lighttpd2
Memcached
Redis
Vim
ZeroMQ
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
3–14 years of development/project
200–1900 code revisions/project

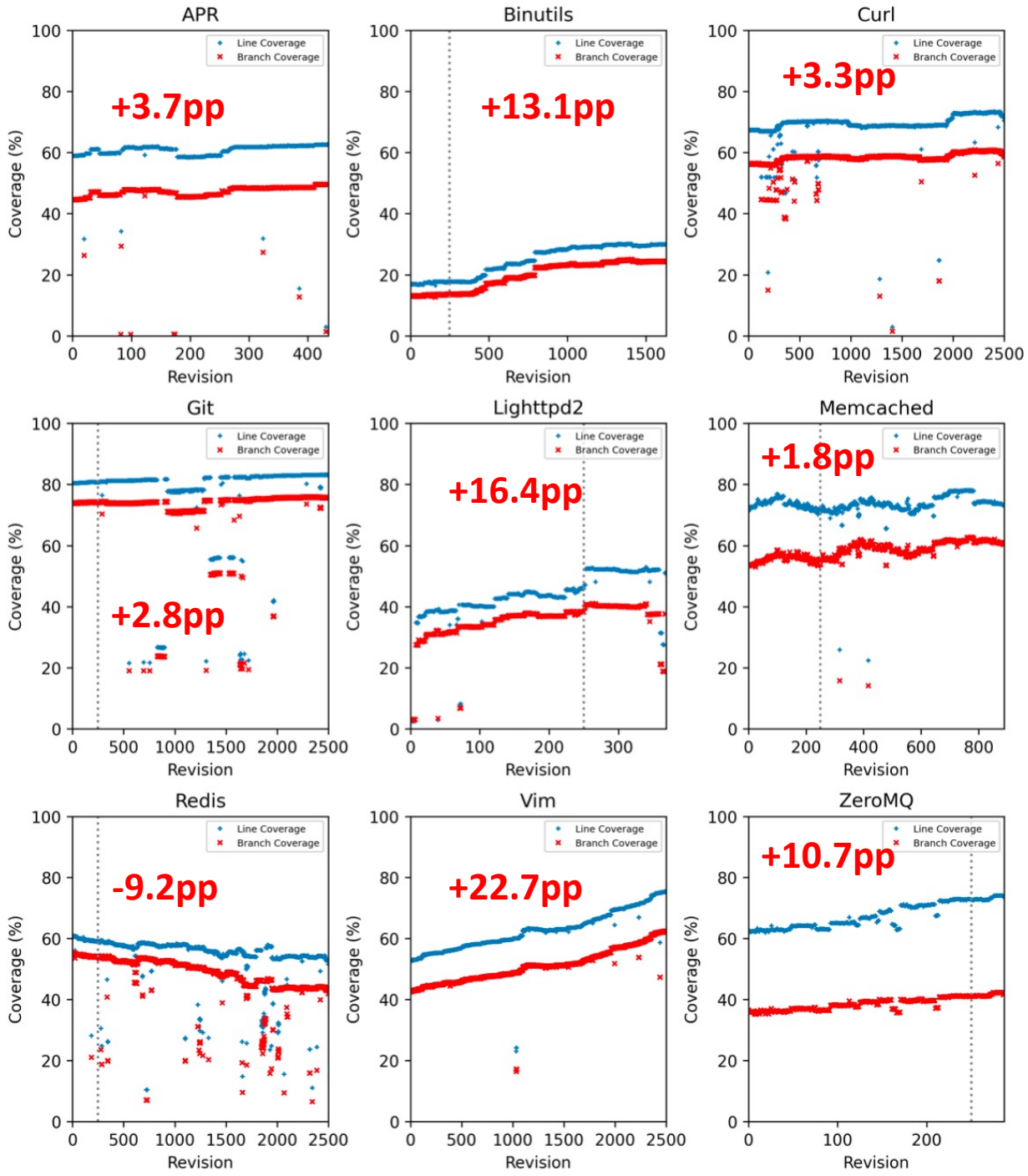


ELOC/time

Code increases of
2.5K – 33K LOC,
24% – 268%

Coverage Evolution

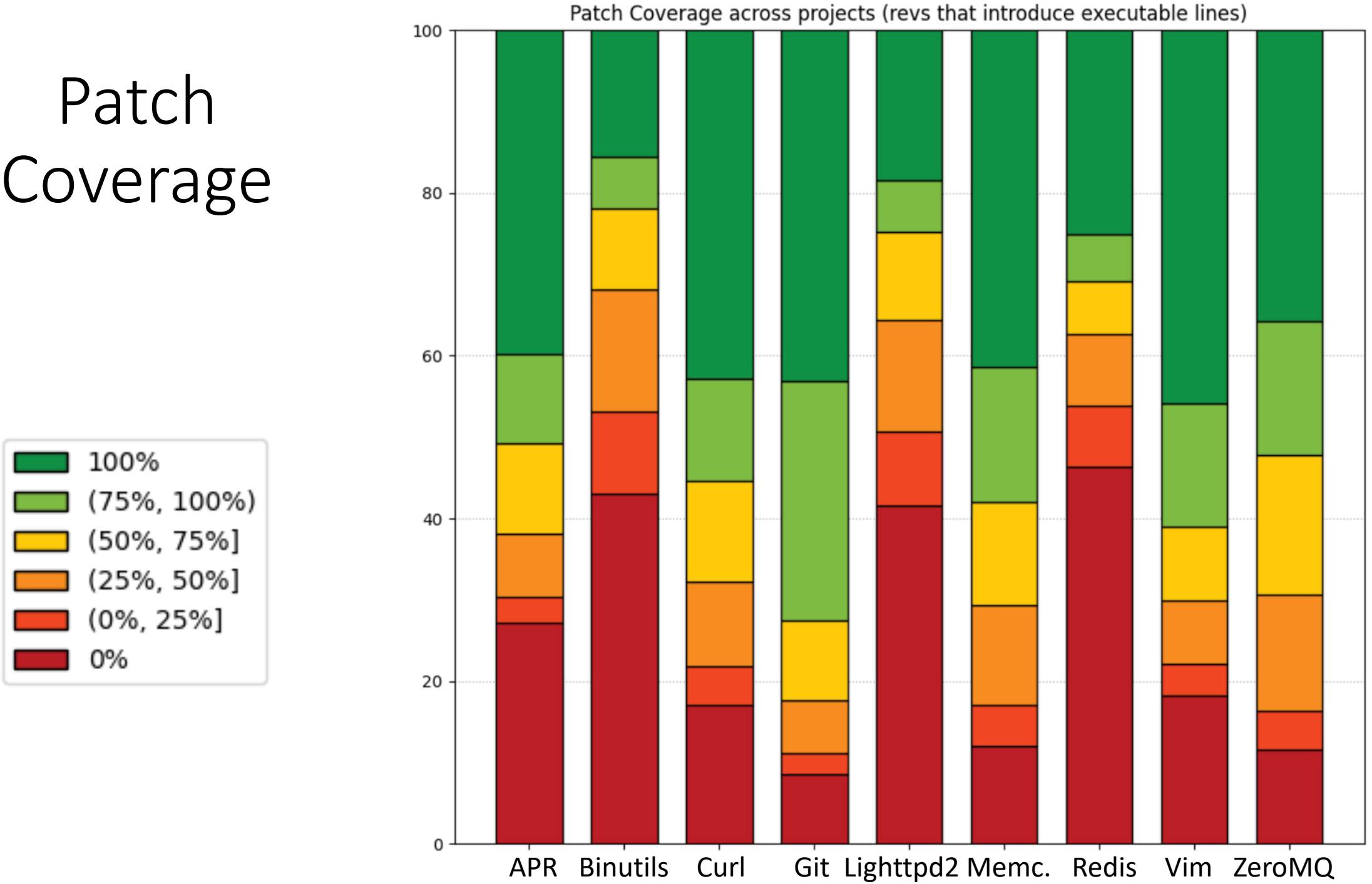
Coverage increases by 2.8 – 22.7pp
It decreases in Redis by 9.2pp



Line coverage

Branch coverage

Patch Coverage

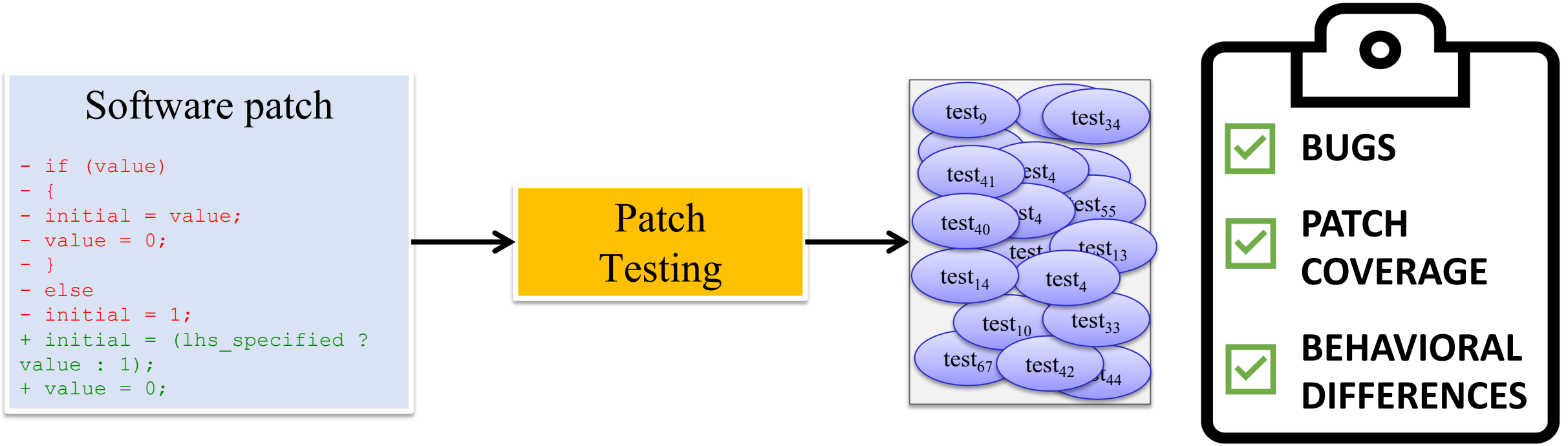


Can Fuzzers Help?

YES, BUT…

Not Agile Enough

- Most techniques focus on whole-program testing (it's easier!)
- Most of our benchmark suites measure global metrics (bugs, coverage)
- Good progress on patch testing techniques, but results still poor overall



CIFuzz

- CI version of OSS-Fuzz: 10 minutes/patch
- Runs only the fuzzers that reached the changed files

OSS-Fuzz Projects	CIFuzz Runs	Crash Uploads
293	108836	1627
grok	41406	1052

According to http://cifuzz.appspot.com/, 7 July 2023 (dashboard active only for a limited time)

Great initiative, but would love to see similar reports, trophies & community attention as for OSS-Fuzz

*Thanks to Jonathan Metzman for answering my CIFuzz questions

Not Fast Enough
24h too long for patch testing

Targeted exploration

KATCH: High-Coverage Testing of Software Patches

Paul Dan Marinescu
Department of Computing
Imperial College London, UK
p.marinescu@imperial.ac.uk

Cristian Cadar
Department of Computing
Imperial College London, UK
c.cadar@imperial.ac.uk

Targeted exploration

Directed Greybox Fuzzing

Marcel Böhme
National University of Singapore, Singapore
marcel.boehme@acm.org

Van-Thuan Pham*
National University of Singapore, Singapore
thuanpv@comp.nus.edu.sg

Manh-Dung Nguyen
National University of Singapore, Singapore
dungnguy@comp.nus.edu.sg

Abhik Roychoudhury
National University of Singapore, Singapore
abhik@comp.nus.edu.sg

Reusing previous analysis results

Running Symbolic Execution Forever

Frank Busse
Imperial College London
United Kingdom
f.busse@imperial.ac.uk

Martin Nowack
Imperial College London
United Kingdom
m.nowack@imperial.ac.uk

Cristian Cadar
Imperial College London
United Kingdom
c.cadar@imperial.ac.uk

Reusing previous analysis results

Running only the fuzzers that reached the changed files (CIFuzz)

Not Automated Enough

APPLICATION	OSS-FUZZ
APR	
BINUTILS	✓
CURL	✓
GIT	
LIGHTTPD	✓
MEMCACHED	
REDIS	
VIM	
ZEROMQ	

According to Fuzz Introspector, https://introspector.oss-fuzz.com/, 28 June 2023

Not Automated Enough

APPLICATION	OSS-FUZZ	COVERAGE
APR		
BINUTILS	✓	35.31%
CURL	✓	5.05%
GIT		
LIGHTTPD	✓	34.58%
MEMCACHED		
REDIS		
VIM		
ZEROMQ		

According to Fuzz Introspector, https://introspector.oss-fuzz.com/, 28 June 2023

Not Automated Enough

APPLICATION	OSS-FUZZ	COVERAGE	FUZZ TARGETS
APR			
BINUTILS	✓	35.31%	26
CURL	✓	5.05%	20
GIT			
LIGHTTPD	✓	34.58%	1
MEMCACHED			
REDIS			
VIM			
ZEROMQ			

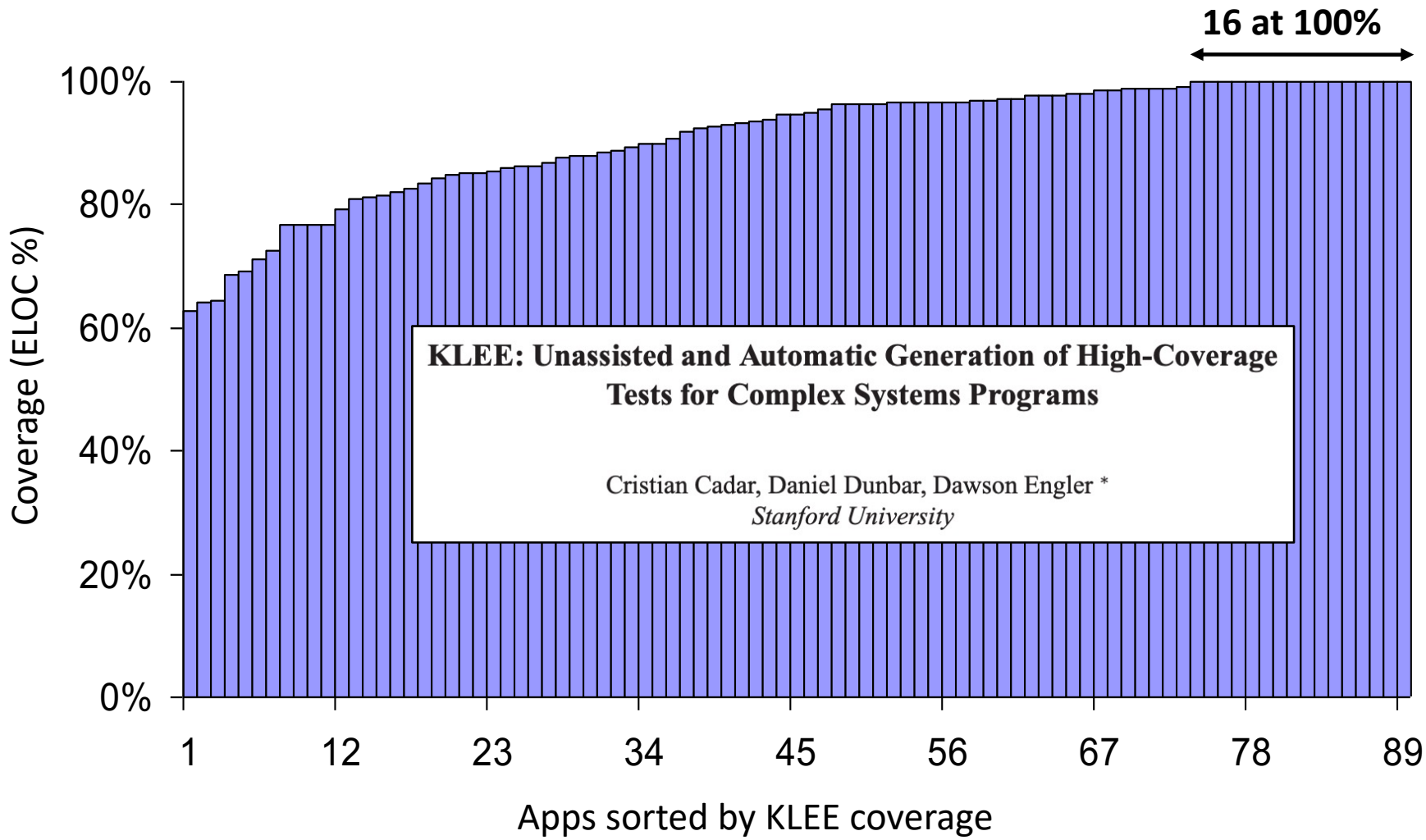
According to Fuzz Introspector, https://introspector.oss-fuzz.com/, 28 June 2023

Papers on improving fuzzing heuristics

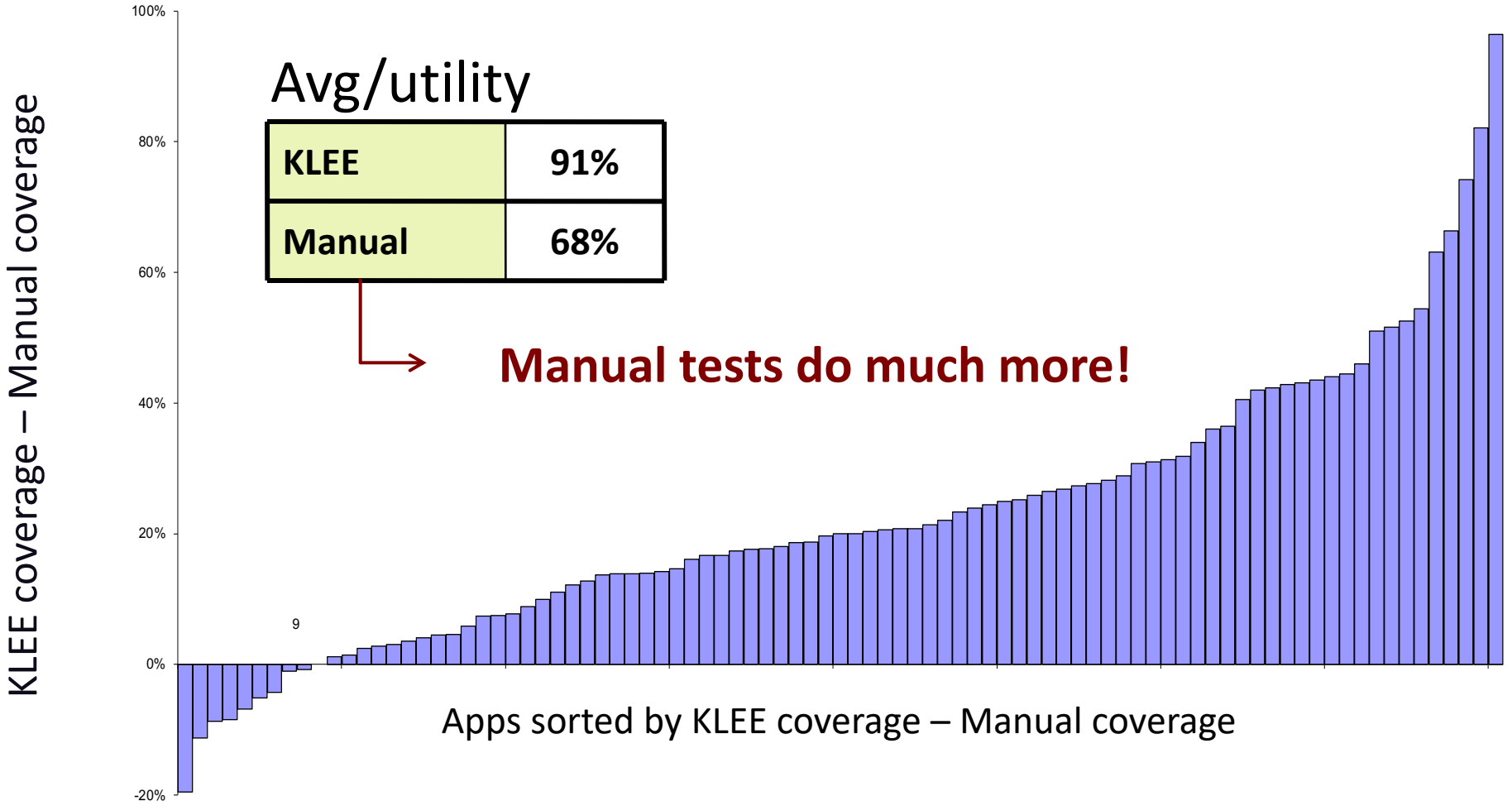
Papers on test driver generation

Improving Test Suites with Fuzzing?

GNU Coreutils: ls, mkdir, echo, sort, …
Overall: 84%, Average 91%, Median 95%



Improving Test Suites with Fuzzing?



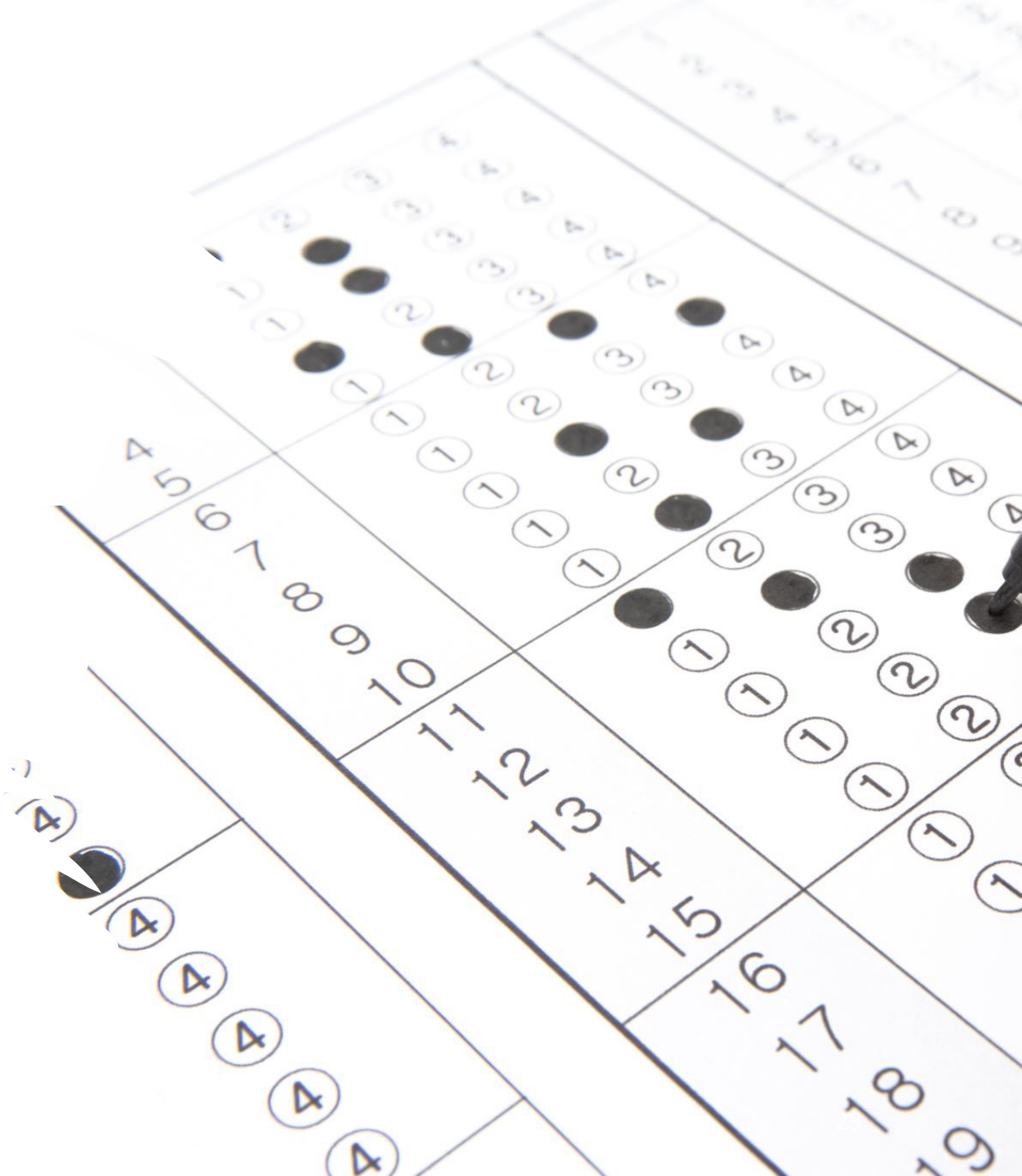
Value of Test Cases

Quality assurance

Debugging Aid

Documentation

. . .



Test Suites: Desired Properties

High Code Coverage

High Feature Coverage

A good test suite performs a combination of code-based & specification-based testing

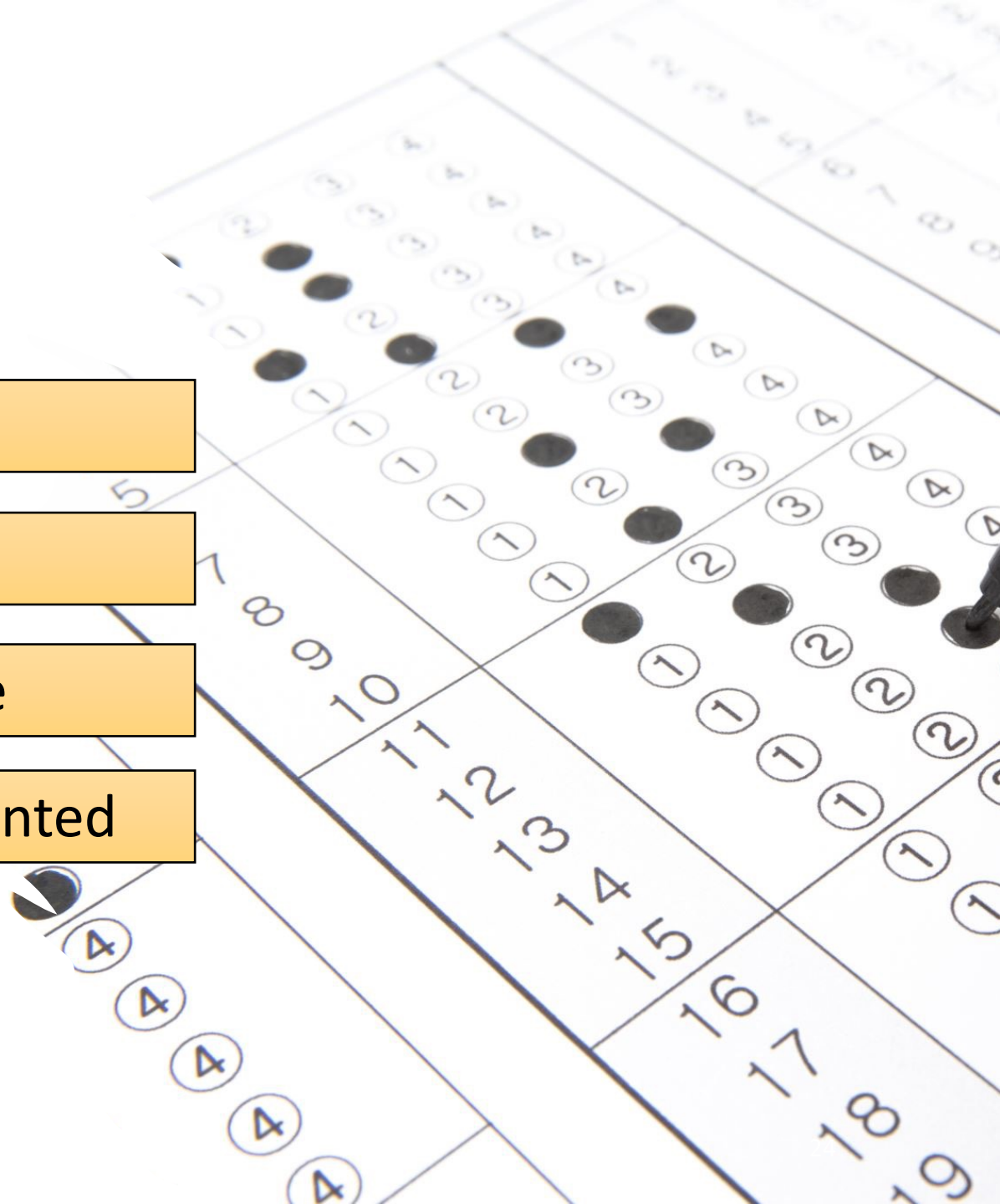
Small

Fast

Readable

Well-documented

. . .



Improving Test Suites with Fuzzing?

- In recent work, we contributed fuzzer-generated tests to the LLVM test suite (16/24 tests accepted)
- Main challenge: oracles and input minimisation

GrayC: Greybox Fuzzing of Compilers and Analysers for C

Karine Even-Mendoza*†
karine.even_mendoza@kcl.ac.uk
Department of Informatics, King's College London
London, United Kingdom

Arindam Sharma*
arindam.sharma@imperial.ac.uk
Department of Computing, Imperial College London
London, United Kingdom

Alastair F. Donaldson
alastair.donaldson@imperial.ac.uk
Department of Computing, Imperial College London
London, United Kingdom

Cristian Cadar
c.cadar@imperial.ac.uk
Department of Computing, Imperial College London
London, United Kingdom

ISSTA 2023, **Wed @ 14:30, Fuzzing 2 session**

Oracles

Input Minimisation

Oracles

Manual tests are (typically) written with good functional oracles

Fuzzer-generated tests:

- Generic/crash bugs in general software (main focus in grey- and whitebox fuzzing)
- Logical bugs in software for specific domains (main focus in blackbox fuzzing)

> *"One fuzzing researcher of particular note is Manuel Rigger […] Most fuzzers only look for assertion faults, crashes, undefined behavior (UB), or other easily detected anomalies. Dr. Rigger's fuzzers, on the other hand, are able to find cases where SQLite computes an incorrect answer."*
>
> *– SQLite webpage*

Logical bugs in general code?

What is the sweet spot?

Patch Specifications

Specifications encoding cross-patch properties

```
assert(out == out_prev + 1)
```

Patch Specifications via Product Programs

Cristian Cadar
Department of Computing
Imperial College London
London, UK
c.cadar@imperial.ac.uk

Daniel Schemmel
Department of Computing
Imperial College London
London, UK
d.schemmel@imperial.ac.uk

Arindam Sharma
Department of Computing
Imperial College London
London, UK
arindam.sharma@imperial.ac.uk

We need a way to make the state of both versions available to the analyser

Product Programs

A mechanism for merging multiple program versions into a single program

Used to reason about hyperproperties in a security context

- Particularly non-interference
- Product program of program P with itself

1) **Can product programs work for multiple versions of a program?**
2) **Can they be constructed automatically for large programs?**
3) **Can they facilitate the writing of patch specifications?**

G. Barthe, J. M. Crespo, C. Kunz, "Relational verification using product programs"
Proc. of the 17th International Symposium on Formal Methods (FM'11)

Example

Previous version

```
x = y - 1;
z = x / 4;
```

Current version

```
x = y - 1;
z = x >> 2;
```

Product program

```
x_prev = y_prev - 1;
x = y - 1;

z_prev = x_prev / 4;
z = x >> 2;
```

```
assert(z == z_prev);
```

Preliminary Experience

- We wrote patch specs for several patches from CoreBench
- CoreBench: a collection of complex real-world patches from popular OSS
- We constructed test drivers around the functions involved in patches
- We used AFL++ and KLEE to look for violations of the patch specs

M. Böhme and A. Roychoudhury, "CoREBench: Studying complexity of regression errors",
In *Proc. of the International Symposium on Software Testing and Analysis (ISSTA'14)*

Patch in ls

```
static char * make_link_name (char const *name,
                              char const *linkname);

make_link_name(”A/B/f.txt", "g.txt") = ”A/B/g.txt"
```

“Do not hard-code ’/’. Use IS_ABSOLUTE_FILE_NAME and dir_len instead. Use stpcpy/stpncpy in place of strncpy/strcpy.”

Patch in ls

AFL++ and KLEE both find a spec violation:

name = /a

linkname = x

```
if (*linkname == '/')
   return xstrdup (linkname);
char const *linkbuf = strrchr (name, '/');
if (linkbuf == NULL)
   return xstrdup (linkname);
size_t bufsiz = linkbuf - name + 1;
char *p = xmalloc (bufsiz + strlen (linkname) + 1);
strncpy (p, name, bufsiz);
strcpy (p + bufsiz, linkname);
return p;
```

Bug made it into a release, was reported by a user and fixed

```
if (IS_ABSOLUTE_FILE_NAME (linkname))
   return xstrdup (linkname);
size_t prefix_len = dir_len (name);
if (prefix_len == 0)
   return xstrdup (linkname);
char *p = xmalloc (prefix_len + 1 + strlen (linkname) + 1);
stpcpy (stpncpy (p, name, prefix_len + 1), linkname);


return p;
assert( strcmp(p, p_prev) == 0 );
```

Patch in ls

AFL++ and KLEE find new spec violation:

name = /x//y

linkname = a

```
if (*linkname == '/')
   return xstrdup (linkname);
char const *linkbuf = strrchr (name, '/');
if (linkbuf == NULL)
   return xstrdup (linkname);
size_t bufsiz = linkbuf - name + 1;
char *p = xmalloc (bufsiz + strlen (linkname) + 1);
strncpy (p, name, bufsiz);
strcpy (p + bufsiz, linkname);
return p;
```

```
if (IS_ABSOLUTE_FILE_NAME (linkname))
   return xstrdup (linkname);
size_t prefix_len = dir_len (name);
if (prefix_len == 0)
   return xstrdup (linkname);
char *p = xmalloc (prefix_len + 1 + strlen (linkname) + 1);
stpcpy (stpncpy (p, name, prefix_len + 1), linkname);
if ( ! ISSLASH (name[prefix_len - 1]))  ++prefix_len;
stpcpy (stpncpy (p, name, prefix_len), linkname);
return p;
assert( strcmp(p, p_prev) == 0 );
```

Code patch to fix reported bug

Patch in ls

No more spec violations found is path-based equality is used

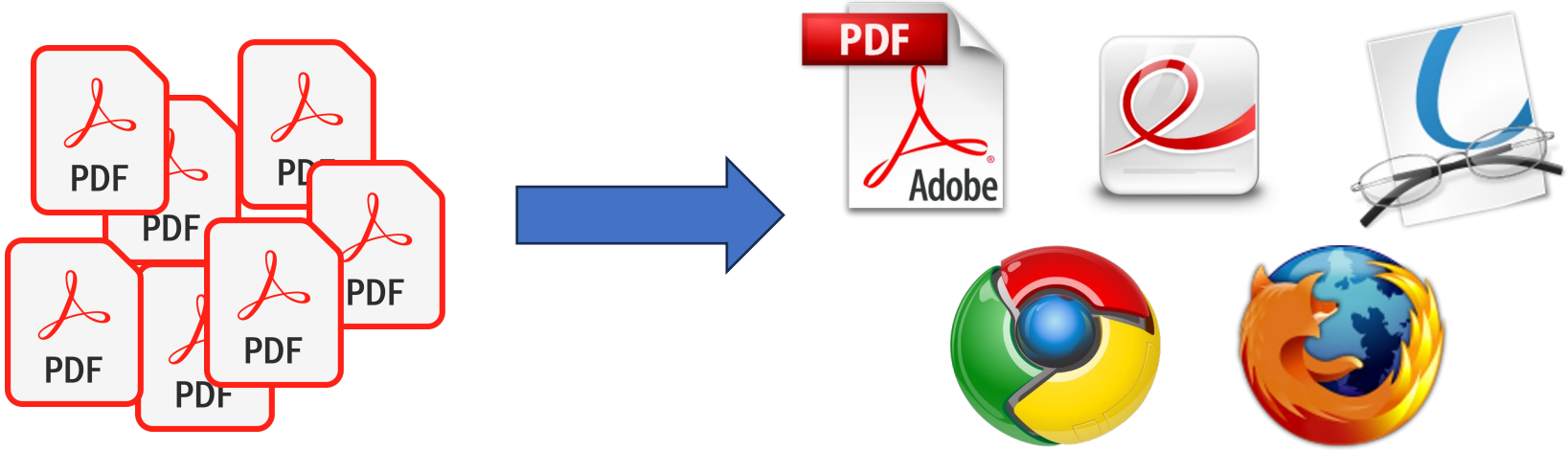
```
if (*linkname == '/')
   return xstrdup (linkname);
char const *linkbuf = strrchr (name, '/');
if (linkbuf == NULL)
   return xstrdup (linkname);
size_t bufsiz = linkbuf - name + 1;
char *p = xmalloc (bufsiz + strlen (linkname) + 1);
strncpy (p, name, bufsiz);
strcpy (p + bufsiz, linkname);
return p;
```

```
if (IS_ABSOLUTE_FILE_NAME (linkname))
   return xstrdup (linkname);
size_t prefix_len = dir_len (name);
if (prefix_len == 0)
   return xstrdup (linkname);
char *p = xmalloc (prefix_len + 1 + strlen (linkname) + 1);
stpcpy (stpncpy (p, name, prefix_len + 1), linkname);
if ( ! ISSLASH (name[prefix_len - 1]))  ++prefix_len;
stpcpy (stpncpy (p, name, prefix_len), linkname);
return p;
assert( patheq(p, p_prev) == 0 );
```

Oracles

Input Minimisation

Automated Bug Finding *WITHOUT* Any Input Generation



> 230K files

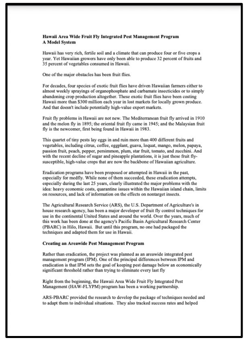
11 Linux readers
(+8 for Windows)

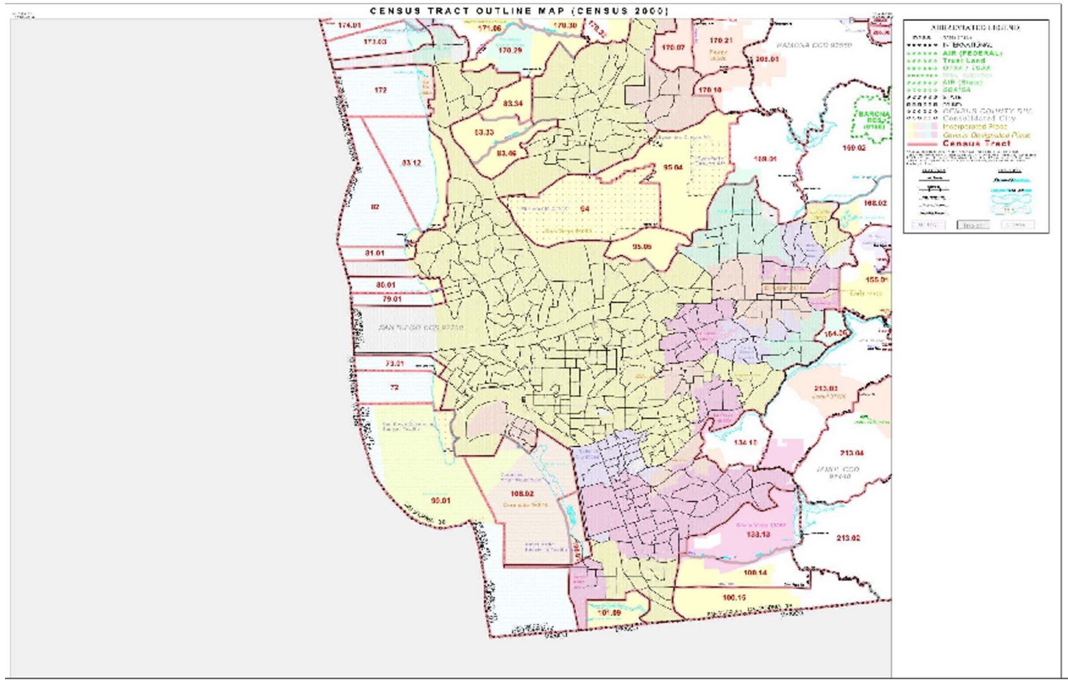
Bug Reports

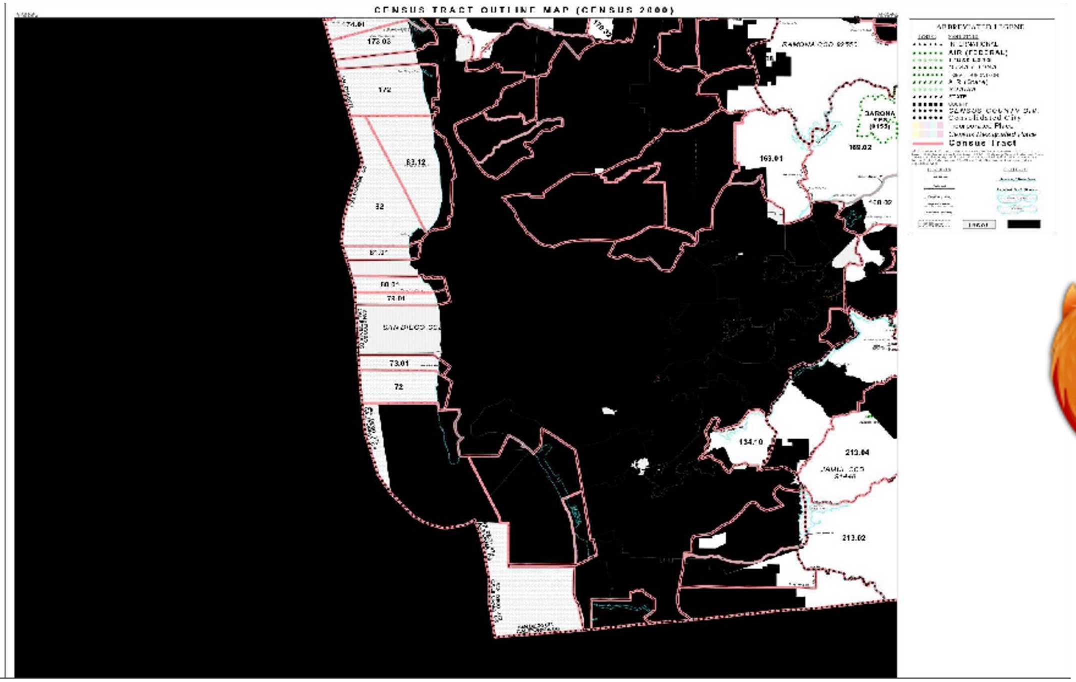
On the correctness of electronic documents: studying, finding, and localizing inconsistency bugs in PDF readers and files

Tomasz Kuchta[1] · Thibaud Lutellier[2] · Edmund Wong[2] · Lin Tan[2] · Cristian Cadar[1]

Examples: Chrome and Firefox Bugs







Automated Bug Finding *WITHOUT* Any Input Generation

Plenty of inputs that matter: real-world, human-created

Oracle challenge

- Detecting meaningful cross-reader inconsistencies

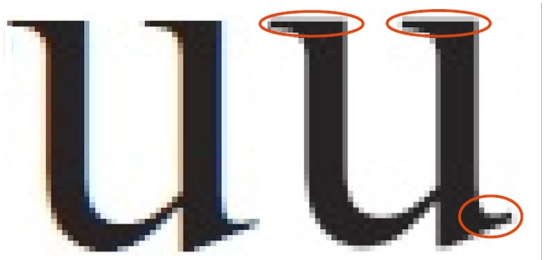
Too many bugs

- 2% of docs crash at least one reader
- 13% of docs trigger inconsistencies

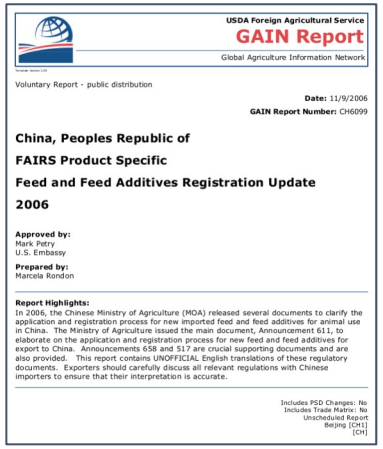
Too large inputs (documents)

PDF Domain: Solutions

- Ignore inconsistencies imperceptible to the human eye: we use CW-SSIM algorithm
- Cluster documents based on warnings and errors emitted by the readers
- Minimise documents based on delta debugging at the level of PDF objects



Evince Chromium


USDA Foreign Agricultural Service
GAIN Report
Global Agriculture Information Network

Voluntary Report - public distribution

Date: 11/9/2006
GAIN Report Number: CH6099

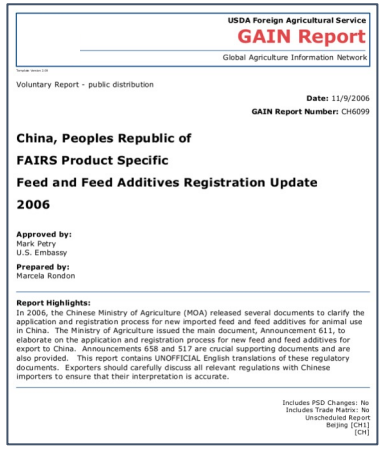
China, Peoples Republic of
FAIRS Product Specific
Feed and Feed Additives Registration Update
2006

Approved by:
Mark Petry
U.S. Embassy

Prepared by:
Marcela Rondon

Report Highlights:
In 2006, the Chinese Ministry of Agriculture (MOA) released several documents to clarify the application and registration process for new imported feed and feed additives for animal use in China. The Ministry of Agriculture issued the main document, Announcement 611, to elaborate on the application and registration process for new feed and feed additives for export to China. Announcements 658 and 517 are crucial supporting documents and are also provided. This report contains UNOFFICIAL English translations of these regulatory documents. Exporters should carefully discuss all relevant regulations with Chinese importers to ensure that their interpretation is accurate.

Includes PSD Changes: No
Includes Trade Matrix: No
Unscheduled Report
Beijing [CH1]
[CH]


USDA Foreign Agricultural Service
GAIN Report
Global Agriculture Information Network

Voluntary Report - public distribution

Date: 11/9/2006
GAIN Report Number: CH6099

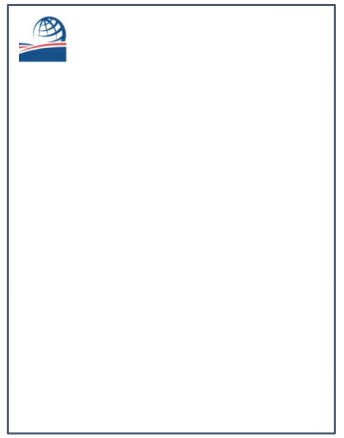
China, Peoples Republic of
FAIRS Product Specific
Feed and Feed Additives Registration Update
2006

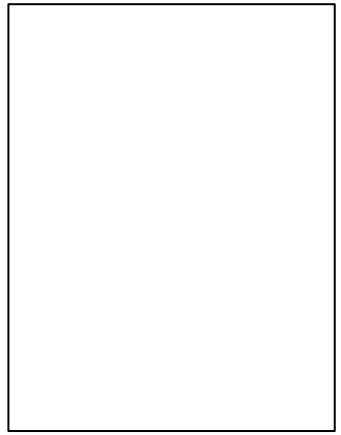
Approved by:
Mark Petry
U.S. Embassy

Prepared by:
Marcela Rondon

Report Highlights:
In 2006, the Chinese Ministry of Agriculture (MOA) released several documents to clarify the application and registration process for new imported feed and feed additives for animal use in China. The Ministry of Agriculture issued the main document, Announcement 611, to elaborate on the application and registration process for new feed and feed additives for export to China. Announcements 658 and 517 are crucial supporting documents and are also provided. This report contains UNOFFICIAL English translations of these regulatory documents. Exporters should carefully discuss all relevant regulations with Chinese importers to ensure that their interpretation is accurate.

Includes PSD Changes: No
Includes Trade Matrix: No
Unscheduled Report
Beijing [CH1]
[CH]





Fuzzing and the Bystander Effect
Success or Failure?

Should developers rely on fuzzers to replace test suites?

Ongoing project with Ahmed Zaki and Arindam Sharma

"xmlsec is integrated with OSS-Fuzz and is continuously fuzzed with the latest libxml2 code from the master branch. So your tests offer very little on top of that."

– libxml2 developer, listing one reason for not accepting some test contributions

Show entries Search: xmlsec

Project name	Language	Fuzz build status	Coverage build status	Introspector build status
xmlsec	c++	fail [log]	fail [log]	fail [log]

Three Colours of Fuzzing: Reflections and Open Challenges

Fuzzing is about finding bugs

…but our objective should be to **improve software**

Key challenge: Better integration of fuzzing into development process

- Automatically generate test drivers / fuzz targets
- Using fuzzing in an incremental fashion
- Generating inputs that trigger different behaviours across versions
- Using fuzzing to enhance test suites
- Moving beyond crash bugs
- . . .