Compiler Fuzzing: How Much Does It Matter?

Michaël Marcozzi* Qiyi Tang* Alastair F. Donaldson Cristian Cadar

*The presented experimental study has been carried out equally by M. Marcozzi and Q. Tang.
Outline

1. **Context:** compiler fuzzing

2. **Problem:** importance of fuzzer-found miscompilations is unclear

3. **Goal:** a study of the practical impact of miscompilation bugs

4. **Methodology** for bug impact measurement

5. **Experiments and results**

6. **Conclusions**
Compiler Bugs

- Software developers intensively rely on compilers, often with blind confidence

- Compilers are software: they have bugs too (~150 fixed bugs/month in LLVM compiler)

- In worst case, unnoticed miscompilation (silent generation of wrong code)

History of LLVM Bug Tracking System (2003-2015) [Sun et al., ISSTA’16]
Compiler Validation (1/2)

- Classical software validation approaches have been applied to compilers
  - Formal verification: CompCert verified compiler, Alive optimisation prover, etc.
  - Testing: LLVM test suite, etc.
Compiler Validation (2/2)

• Recent surge of interest in compiler fuzzing:

• Automatic and massive random generation of test programs to compile

• Automatic miscompilation detection via differential or metamorphic testing

• e.g. 200+ miscompilations found in LLVM by Csmith\(^1\), EMI\(^2\), Orange\(^3\) and Yarpgen\(^4\)

\(^1\) [Yang et al., PLDI’11] [Regehr et al., PLDI’12] [Chen et al., PLDI’13]
\(^2\) Equivalence Modulo Inputs [Le et al., PLDI’14, OOPSLA’15] [Sun et al., OOPSLA’16]
\(^3\) [Nagai et al., T-SLDM] [Nakamura et al., APCCAS’16]
\(^4\) [https://github.com/intel/yarpgen]

---

Csmith

---

M. Marcozzi
5
Compiler Fuzzing: How Much Does It Matter?
Outline

1. **Context:** compiler fuzzing

2. **Problem:** importance of fuzzer-found miscompilations is unclear

3. **Goal:** a study of the practical impact of miscompilation bugs

4. **Methodology** for bug impact measurement

5. **Experiments and results**

6. **Conclusions**
Audience of our talks on compiler fuzzers often question the importance of found bugs.

In our experience, this is a contentious debate and people can be poles apart:

In my opinion, compiler bugs are extremely dangerous, period. Thus, regardless of the real-world impact of compiler bugs, I think that techniques that can uncover (and help fix) compiler bugs are extremely valuable.

One anonymous reviewer of this paper at a top P/L conference

I would suggest that compiler developers stop responding to researchers working toward publishing papers on [fuzzers]. Responses from compiler maintainers is being becoming a metric for measuring the performance of [fuzzers], so responding just encourages the trolls.

'The Shape of Code' weblog author
(former UK representative at ISO International C Standard)
Importance of Fuzzer-Found Miscompilations (2/2)

• In this work, we consider a **mature compiler** in a **non-critical environment**:
  
  • The compiler has been **intensively tested by its developers and users**
  
  • **Trade-offs between software reliability and cost** are acceptable and common

• In this context, **doubting the impact of fuzzer-found bugs is reasonable**:
  
  🟢 It is unclear if mature compilers **leave much space to find severe bugs**
  
  🟢 Fuzzers find bugs affecting **generated code**, whose patterns may not occur in real code
Outline

1. Context: compiler fuzzing

2. Problem: importance of fuzzer-found miscompilations is unclear

3. Goal: a study of the practical impact of miscompilation bugs

4. Methodology for bug impact measurement

5. Experiments and results

6. Conclusions
Goal and Challenges

• In this work, our objectives are to:

  - [x] Show specifically that compiler fuzzing matters or does not matter
  - [✓] Study the impact of miscompilation bugs in a mature compiler over real apps
  - [✓] Compare impact of bugs from fuzzers with others (e.g. found by compiling real code)

• Operationally, we aim at overcoming the following challenges:
  
  • Take steps towards a methodology to measure the impact of a miscompilation bug
  • Apply it over a significant but tractable set of bugs and real applications
Outline

1. Context: compiler fuzzing

2. Problem: importance of fuzzer-found miscompilations is unclear

3. Goal: a study of the practical impact of miscompilation bugs

4. Methodology for bug impact measurement

5. Experiments and results

6. Conclusions
Bug Impact Measurement Methodology

• **Assumption**: Restrict to **publicly fixed bugs in open-source compilers**, to extract
Bug Impact Measurement Methodology

- **Assumption**: Restrict to **publicly fixed bugs in open-source compilers**, to extract

  ![Buggy Compiler Source](image1)

  ![Fixing Patch](image2)

  *written by developers*
Assumption: Restrict to publicly fixed bugs in open-source compilers, to extract Bug Impact Measurement Methodology.
Bug Impact Measurement Methodology

- **Assumption**: Restrict to **publicly fixed bugs in open-source compilers**, to extract

- **Assumption**: impact of miscompilation bug = **ability to change semantics of real apps**
Bug Impact Measurement Methodology

- **Assumption**: Restrict to **publicly fixed bugs in open-source compilers**, to extract

![Buggy Compiler Source](image1) ![Fixing Patch written by developers](image2) ![Fixed Compiler Source](image3)

- **Assumption**: impact of miscompilation bug = **ability to change semantics of real apps**

- We **estimate** the **impact** of the compiler **bug over a real app** in **three stages**:
Bug Impact Measurement Methodology

- **Assumption**: Restrict to **publicly fixed bugs in open-source compilers**, to extract

  ![Diagram](image)

  **Buggy Compiler Source** → **Fixing Patch written by developers** → **Fixed Compiler Source**

- **Assumption**: impact of miscompilation bug = **ability to change semantics of real apps**

- **We estimate** the **impact** of the compiler **bug over a real app** in **three stages**:
  1. Is the buggy compiler code reached and triggered **during compilation**?
Bug Impact Measurement Methodology

- **Assumption**: Restrict to publicly fixed bugs in open-source compilers, to extract

![Diagram showing transition from buggy to fixed compiler source](image)

- **Assumption**: impact of miscompilation bug = ability to change semantics of real apps

- **We estimate** the **impact** of the compiler **bug over a real app** in **three stages**:
  1. Is the buggy compiler code reached and triggered during compilation?
  2. How much does a triggered bug change the **binary code**?
Bug Impact Measurement Methodology

- **Assumption**: Restrict to **publicly fixed bugs in open-source compilers**, to extract

  ![Diagram showing the process from Buggy Compiler Source to Fixed Compiler Source using a fixing patch written by developers.]

- **Assumption**: impact of miscompilation bug = **ability to change semantics of real apps**

- **We estimate** the **impact** of the compiler **bug over a real app** in **three stages**:
  1. Is the buggy compiler code reached and triggered **during compilation**?
  2. How much does a triggered bug change the **binary code**?
  3. Can the binary changes lead to differences in **binary runtime behaviour**?
Stage 1: Compile-Time Analysis

```java
if (Not.isPowerOf2())
/* Code transformation */
```

```java
fix for LLVM bug #26323
```

Buggy Compiler Source

```java
if (Not.isPowerOf2())
&& C->getValue().isPowerOf2() && Not != C->getValue())
/* Code transformation */
```

Fixed Compiler Source
Stage 1: Compile-Time Analysis

```c
if (Not.isPowerOf2())
    /* Code transformation */
else
    if (!(C->getValue().isPowerOf2() && Not != C->getValue()))
        warn("Bug triggered!");
    else /* Code transformation */
```

```c
if (Not.isPowerOf2())
    /* Code transformation */
else if (C->getValue().isPowerOf2() && Not != C->getValue())
    warn("Fixing patch reached!");
```

Fix for LLVM bug #26323

**Warning-Laden Compiler**

M. Marcozzi

Compiler Fuzzing: How Much Does It Matter?
Stage 1: Compile-Time Analysis

if (Not.isPowerOf2())
   /* Code transformation */

if (Not.isPowerOf2())
   && C->getValue().isPowerOf2() && Not != C->getValue() )
   /* Code transformation */

fix for LLVM bug #26323

Buggy Compiler Source

Fixed Compiler Source

warn("Fixing patch reached!");
if (Not.isPowerOf2()) {
    if (! (C->getValue().isPowerOf2() && Not != C->getValue() ))
        warn("Bug triggered!");
    else /* Code transformation */
}

Warning-Laden Compiler
Stage 1: Compile-Time Analysis

Buggy Compiler Source

```cpp
if (Not.isPowerOf2())
/* Code transformation */
```

fix for LLVM bug #26323

Fixed Compiler Source

```cpp
if (Not.isPowerOf2())
&& C->getValue().isPowerOf2() && Not != C->getValue())
/* Code transformation */
```

M. Marcozzi

Compiler Fuzzing: How Much Does It Matter?

if (Not.isPowerOf2()) {
  if (!(C->getValue().isPowerOf2() && Not != C->getValue()))
    warn("Bug triggered!");
  else
    /* Code transformation */
}

warn("Fixing patch reached!");

C

grep logs

"Fixing patch reached!"
| "Bug triggered!"

Warning-Laden Compiler

101011

LLVM bug #26323
Stage 2: Syntactic Binary Analysis

Buggy Compiler

```
if (Not.isPowerOf2())
```

Fixed Compiler

```
if (Not.isPowerOf2() 
   && C->getValue().isPowerOf2() 
   && Not != C->getValue())
```
Stage 2: Syntactic Binary Analysis

Buggy Compiler

\[
\text{if (Not.isPowerOf2())}
\]

Fixed Compiler

\[
\text{if (Not.isPowerOf2())} \\
\text{\&\& C->getValue().isPowerOf2() \&\& Not != C->getValue())}
\]
**Stage 2: Syntactic Binary Analysis**

### Buggy Compiler

```c
if (Not.isPowerOf2())
    // Code execution
```

### Fixed Compiler

```c
if (Not.isPowerOf2())
    & & C->getValue().isPowerOf2() & & Not != C->getValue())
```

M. Marcozzi  
Compiler Fuzzing: How Much Does It Matter?
Stage 2: Syntactic Binary Analysis

Buggy Compiler

if (Not.isPowerOf2())

Fixed Compiler

if (Not.isPowerOf2())
&& C->getValue().isPowerOf2() && Not != C->getValue())

Check for syntactic differences in assembly

M. Marcozzi
Stage 3: Dynamic Binary Analysis
Stage 3: Dynamic Binary Analysis
Stage 3: Dynamic Binary Analysis

Count divergent test results
Stage 3: Dynamic Binary Analysis

No test divergence does not mean that binaries are semantically equivalent.
Stage 3: Dynamic Binary Analysis

XX: mov $5, %eax
≠
XX: addl $4, %esp
Stage 3: Dynamic Binary Analysis

Manual crafting of inputs to trigger runtime divergence

XX: mov $5, %eax
≠
XX: addl $4, %esp
Outline

1. Context: compiler fuzzing
2. Problem: importance of fuzzer-found miscompilations is unclear
3. Goal: a study of the practical impact of miscompilation bugs
4. Methodology for bug impact measurement
5. Experiments and results
6. Conclusions
Experiments (1/2)

We apply our bug impact measurement methodology over a sample of:

- 45 miscompilations bugs in the open-source LLVM compiler (C/C++ → x86_64)

- 27 fuzzer-found bugs (12% of miscompilations from Csmith, EMI, Orange and Yarpgen)

- 10 bugs detected by compiling real code and 8 bugs from Alive formal verification tool
We apply our bug impact measurement methodology over a sample of:

- 309 Debian packages totalling 10M+ lines of C/C++ code
- Not part of the LLVM test suite
- Diverse set of applications w.r.t. type, size, popularity and maturity
A lot of manual effort and 5 months of computation happen here
Results

- 27 fuzzer-found bugs
- 10 bugs affecting real code
- 8 formal verification bugs

Stage 1a:
- Patch reached: 70%
- Patch triggered: 65%
- Different binary: 43%

Stage 1b:
- Bug triggered: 28%
- 19%
- 13%

Stage 2:
- Different binary: 6%
- 2%
- 7%
- Test divergence: 0.01%
- 0.01%
- 0%

Stage 3:
- Test divergence: 0.01%
- 0.01%
- 0%

M. Marcozzi
Compiler Fuzzing: How Much Does It Matter?
Results

Only a tiny fraction of the code is affected.
Results

- 27 fuzzer-found bugs
- 10 bugs affecting real code
- 8 formal verification bugs

Stage 1a:
- Patch reached: 70%
- Bug triggered: 65%
- Different binary: 43%

Stage 1b:
- Patch reached: 70%
- Bug triggered: 28%
- Different binary: 19%

Stage 2:
- Patch reached: 70%
- Bug triggered: 6%
- Different binary: 2%

Stage 3:
- Patch reached: 70%
- Bug triggered: 28%
- Different binary: 13%

One test failure in zsh (+ one extra test failure in SQLite)
One test failure in leveldb

5 bugs affecting real code
8 formal verification bugs

M. Marcozzi
Compiler Fuzzing: How Much Does It Matter?
Results

Sample of Package Test Suites
47% average statement coverage
Half suites > 50% statement coverage

One test failure in zsh
(+ one extra test failure in SQLite)

One test failure in leveldb

27 fuzzer-found bugs
10 bugs affecting real code
8 formal verification bugs

Results

M. Marcozzi

Compiler Fuzzing: How Much Does It Matter?
Results

27 fuzzer-found bugs
10 bugs affecting real code
8 formal verification bugs

Manual Inspection
the ~50 inspected binary differences...
either have no semantic impact
or require very specific runtime circumstances
to impact behaviour

M. Marcozzi
Compiler Fuzzing: How Much Does It Matter?
Outline

1. Context: compiler fuzzing
2. Problem: importance of fuzzer-found miscompilations is unclear
3. Goal: a study of the practical impact of miscompilation bugs
4. Methodology for bug impact measurement
5. Experiments and results
6. Conclusions
Conclusions

- Our **two major take-aways** are that miscompilations bugs in a mature compiler…
- *seldom impact* app reliability (as probed by test suites and manual inspection)
- have *similar impact* no matter they were found in real or fuzzer-generated code
- A **possible explainer** for these results is that, in a mature compiler…
  - all the bugs **affecting patterns frequent in real code** have already been **fixed**
  - only **corner-case bugs remain**, affecting real and generated code similarly
Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!

> Preprint and artifact available
https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available
https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Version: 2019
Author: MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA
Department of Computing, Imperial College London, UK

---

Compiler Fuzzing: How Much Does It Matter?

MICHAEL MARCOZZI, QIYI TANG, ALASTAIR E. DONALDSON, and CRISTIAN GADOA

Imperial College London, United Kingdom

Preprint and artifact available at https://srg.doc.ic.ac.uk/projects/compiler-bugs

> Postdoc position available at https://srg.doc.ic.ac.uk/vacancies/postdoc-comp-pass-19

---

Thank you for listening!