Compiler Fuzzing: How Much Does It Matter?

Michaël Marcozzi, Qiyi Tang, Cristian Cadar, Alastair Donaldson

10th South of England Regional Programming Language Seminar (S-REPLS 10)
Birkbeck, University of London, 18 September 2018

Outline

1. About compiler fuzzing
2. Measuring the impact of a compiler bug
3. Impact of compiler bugs found by fuzzing: ongoing study
4. Preliminary conclusion
Outline

1. About compiler fuzzing
2. Measuring the impact of a compiler bug
3. Impact of compiler bugs found by fuzzing: ongoing study
4. Preliminary conclusion

Compilers

• Core component of software development toolchain
• Often relied on with some kind of blind confidence
• But vulnerable to all issues affecting software, including bugs:

[Sun et al., ISSTA’16]
Compiler bugs

• Consequence of a compiler bug:
  • Compiler crash:
    • Assertion violation, internal error, segfault, timeout, RAM exhaustion...
    • Moderate severity: does not affect the compiled app at production time
  • Wrong-code generation:
    • The compiler silently emits target code not semantically equivalent to source
    • Critical severity: can go unnoticed until the compiled app misbehaves in production
    • Main rationale for extensive compiler verification!

• Approaches to extensive compiler verification: formal proof and fuzzing

Compiler fuzzing (1/2)

• Automated random testing of compilers

• Recently attracted much research, following CSmith tool [Yang et al., PLDI’11]
• Researchers found solutions to common test automation challenges:
  • Input generation: create bug-triggering input programs for compilers
  • Oracle production: detect when wrong-code generation occurs
  • Test reduction: find the minimal miscompiled part of a program
Compiler fuzzing (2/2)

- **Fuzzers reported many bugs** in mainstream open-source C/C++ compilers:
  - **Csmith** [Yang et al., PLDI’11]: 400+ bugs in GCC/LLVM
  - **EMI** [Le et al., PLDI’14]: 1500+ bugs in GCC/LLVM
  - **Orange** [Nakamura et al., APCCAS’16]: 50+ bugs in GCC/LLVM
  - **Yarpgen** (Intel): 140+ bugs in GCC/LLVM

- How much do these bugs make real apps fail in production? **2 threats to impact:**
  - Fuzzers find bugs that occur when compiling artificial, randomly created apps
  - Miscompilations can be spotted when apps are tested and never reach production

- **Our goal**: measure the actual impact of these bugs over real apps

Outline

1. About compiler fuzzing
2. Measuring the impact of a compiler bug
3. Impact of compiler bugs found by fuzzing: ongoing study
4. Preliminary conclusion
Outline

1. About compiler fuzzing
2. Measuring the impact of a compiler bug
3. Impact of compiler bugs found by fuzzing: ongoing study
4. Preliminary conclusion

Bug impact estimation (1/2)

- Bugs in open-source compilers are reported on compiler web site
- A bug report typically contains:
  - Sample source code triggering bug
  - Discussion of priority and fix by compiler developers
  - SVN/Git revision number $N_{fix}$ where fix was applied and passed regression tests
Bug impact estimation (2/2)

- Given an app to compile, we consider 3 impact levels for a compiler bug:
  - Level 1: buggy compiler code is triggered (compiler dynamic time)
  - Level 2: faulty binary app code is generated (application static time)
  - Level 3: faulty binary code is spotted during app testing (application dynamic time)

- Trusting the fix proposed by compiler developers, we have:
  - At \( N_{\text{fix}}-1 \), the bad buggy compiler
  - At \( N_{\text{fix}} \), the good fixed compiler

- We use good and bad compilers to estimate the bug level for an app

---

Estimating level 1 impact

LLVM bug #26323

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

```cpp
if (Not.isPowerOf2() && C->getValue(), isPowerOf2())
```

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

```cpp
   { /* PRINT WARNING HERE */ }
```

```cpp
... 
```

Warning?
Estimating level 2 impact

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

Mismatch?

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

Estimating level 3 impact

Mismatch?

```java
if (C->getValue () == D->getValue ()
```
Outline

1. About compiler fuzzing
2. Measuring the impact of a compiler bug
3. Impact of compiler bugs found by fuzzing: ongoing study
4. Preliminary conclusion
Compiler bugs sampling

For each (fuzzer, compiler) pair, we picked 15 high-priority bugs:

- Triggering **wrong-code generation**
- Can be easily reproduced on a **at most 10 years** old **x86/Linux** config
- **Confirmed** by compiler developers and **ranked** at least P3/normal
- **Fix** provided in **isolation of other code changes**

<table>
<thead>
<tr>
<th>Fuzzer</th>
<th>GCC</th>
<th>LLVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Csmith (fuzzer)</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>EMI (fuzzer)</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Orange (fuzzer)</td>
<td>15</td>
<td>all (6)</td>
</tr>
<tr>
<td>Intel Yarpgen (fuzzer)</td>
<td>15</td>
<td>all (4)</td>
</tr>
<tr>
<td>Alive (model-checking)</td>
<td>n.a.</td>
<td>all (8)</td>
</tr>
<tr>
<td>User-reported</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>TOTAL</td>
<td>75</td>
<td>63</td>
</tr>
</tbody>
</table>

Application sampling

- **79 applications for a total of 3.6M lines of code** (and more to come)
- Part of the **Ubuntu Minimal Linux** distribution:
  - C or C++ only
  - Can be compiled with most recent versions of GCC/LLVM
- System utilities, network protocols, DBMS, compression, text processing...
- **Examples**: SQLite, Coreutils, Bzip2, Bash...
Ongoing study

• **Measure bug impact level for each** of the 10,902 *(bug, application)* pairs
  ➢ Evaluate fuzzers ability to find bugs impacting real code (level 1 & 2)
  ➢ Compare this ability:
    • Between each of the four fuzzers
    • Between the fuzzer and the model-checking tool
    • Between using the fuzzers or considering user-reported bugs

➢ Evaluate fuzzers ability to find bugs unseen by app test suites (level 2 ¬3)

• **Preliminary result:** some bugs have level-2 impact for 47% of applications

Outline

1. About compiler fuzzing
2. Measuring the impact of a compiler bug
3. Impact of compiler bugs found by fuzzing: ongoing study
4. Preliminary conclusion
Outline

1. About compiler fuzzing
2. Measuring the impact of a compiler bug
3. Impact of compiler bugs found by fuzzing: ongoing study
4. Preliminary conclusion

Preliminary conclusion

• Hard to have a proper conclusion without full results
• Nice to remember that:
  • Compilers are full of bugs (hundreds are fixed every month)
  • These bugs can make your app fail even if code is correct and no compiler warning
• Future news about this project on our group website:
  https://srg.doc.ic.ac.uk

• My personal website:
  www.marcozzi.net