

Fuzzing

Challenges and Reflections

Marcel Böhme
ARC DECRA Fellow
Senior Lecturer (A/Prof)
Monash University
 [@mboehme_](https://twitter.com/mboehme_)





Organizers



Abhik
Roychoudhury



Cristian
Cadar



Marcel
Böhme

2019 Shonan Meeting on
Fuzzing and Symbolic Execution:
Reflections, Challenges, and Opportunities

Keynote Speakers



Patrice
Godegroid
@Microsoft



Kostya
Serebryany
@Google

FUZZING and SYMBOLIC EXECUTION

SEPTEMBER 24-27, 2019

Cristian Cadar

Naohiro Yoshida

Michael Pradel

Patrik Godefroid

Sergey Mechtaev

Antonio Bertolins

M. Christakis

Filip Nikšić

Marcel Böhme

Willen Visser

Mauro Pister

Sarfraz Khurshid

Tevfik Bultan

Martin Veith

Xinmeng Zhang

David Trabisk

Milos Gligoric

Ashishyandey

Raul Marinescu

Peter Goodman

Darko Marinov

Andreas Zeller

Alessandra Gorb

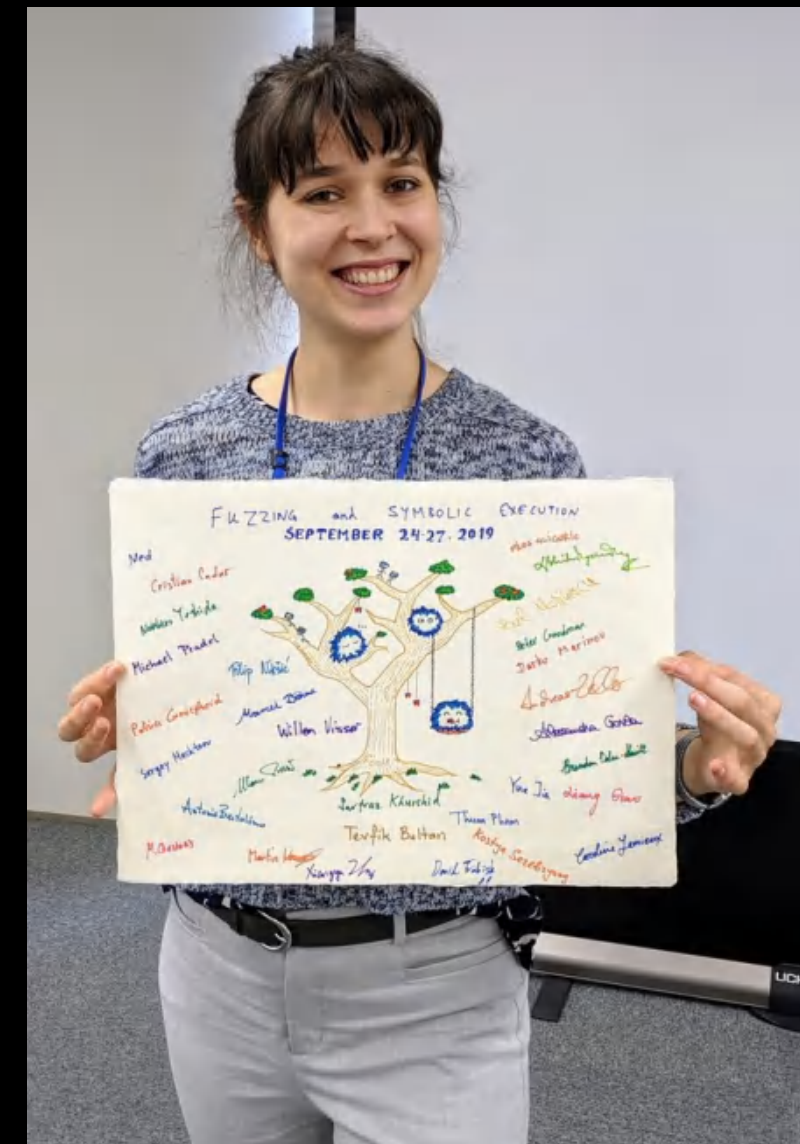
Brendan Dalziel-Haritt

Yue Jia Liang Gao

Thuan Pham

Kostya Serebryany

Caroline Lemieux



Caroline Lemieux
@cestlemieux

Live Tweets

bringing discussions to the larger community

🔄 You Retweeted



Brandon Falk @gamozolabs · Sep 24, 2019

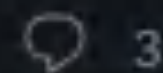
I think benchmarking fuzzers is dangerous. One of the biggest issues I see in lots of fuzzing research is that people stop after finding some known bugs (or even worse, after a few new bugs). At the end of the day `/dev/urandom` piped into programs still routinely finds bugs.



Marcel Böhme @mboehme_ · Sep 24, 2019

"We need proper approaches to benchmark fuzzers. Retire LAVA & CGC. Use real bugs, e.g. github.com/google/fuzzer-..."

[Show this thread](#)



3



14



50



[Show this thread](#)

🔄 You Retweeted



David Brumley @thedavidbrumley · Sep 24, 2019

Great post. And adding fuzzing during development isn't any harder than adding a unit test. I think in 5 years we'll look back and many will be fuzzing just for QA; not security.



Marcel Böhme @mboehme_ · Sep 24, 2019

Kostya's keynote: LibFuzzer hasn't found new bugs in <big software companie>'s library. We didn't know why. Later we got a note that they are now using LibFuzzer during regression testing in CI and that it prevented 3 vulns from reaching to production.

[Show this thread](#)

Survey

validating our findings with the larger community

Section 1 of 5

Fuzzing: Challenges and Reflections

Fuzzing is an emerging research field. We are currently writing up a short paper, with some directions on important topics in fuzzing. We are now organizing a survey to collect the diverse opinions of various researchers in the field. Please find the draft at <https://www.dropbox.com/s/oljac6pt1uuueai/FuzzingChallenges.pdf?dl=0> (6 pages). The paper is a result of a recent Shonan meeting on "Fuzzing and Symbolic Execution: Reflection Challenges and Opportunities" (<https://shonan.nii.ac.jp/seminars/160/>).

It takes about 10 minutes to fill this questionnaire.
There are three very short and two longer sections.

Among these high-level challenges, choose 3 that you find most important to be addressed going forward?

- ☐ Fuzzing Theory (Fundamental limitations of different approaches to fuzzing)
- ☐ Security Auditor in the Loop (Usability, interactive fuzzing w/ human-in-the-loop)
- ☐ Automation (Improving scalability, efficiency, effectiveness, deep bugs)
- ☐ Fair evaluation of specialized fuzzers (Level playing field)
- ☐ Valid measures of effectiveness/efficiency (coverage, synthetic bugs, known bugs, time budget)?
- ☐ How do we evaluate techniques instead of tools?

Reflections

we are all stakeholders of secure open-source.

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

Reflections

we are all stakeholders of secure open-source.

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

```
$ git clone https://github.com/google/oss-fuzz
$ ls -1 oss-fuzz/projects | wc -l
356
```

- Encryption/Decryption (openssl, gnutls, cryptlib, mbed, wolfssl)
- Compression (bzip2, brotli, gzip, lzma, xz, lz4, libarchive)
- Streaming (ffmpeg, gstreamer, libvlc)
- Parser libraries (xml, json, jpg, png, gif, avi, mpg, pcre)
- Databases (mysql, redis, postgres, derby, sqlite)
- Compilers/Interpreter (gcc, llvm [clang,...], php, javascript)
- Protocol implementations (http/http2, ftp, smtp, ssh, tls/ssl, rtsp)
- Server implementations (httpd, nginx, node.js, tomcat, lighttpd)
- Operating systems (ubuntu, debian, android, glibc)

Reflections

fuzzing is having substantial impact!



Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for automatic vulnerability discovery.

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for **automatic vulnerability discovery**.

1. The worldwide information security market is forecast to reach \$170.4 billion in 2022.

([Gartner](#))

2. 62% of businesses experienced phishing and social engineering attacks in 2018. ([Cybint](#)

[Solutions](#))

3. 68% of business leaders feel their cybersecurity risks are increasing. ([Accenture](#))

4. Only 5% of companies' folders are properly protected, on average. ([Varonis](#))

7. 52% of breaches featured hacking, 28% involved malware and 32–33% included phishing or social engineering, respectively. ([Verizon](#))

19. The average cost of a data breach is \$3.92 million as of 2019. ([Security Intelligence](#))

Who led the digital transformation of your company?

A) CEO

B) CTO

C) COVID-19

Reflections

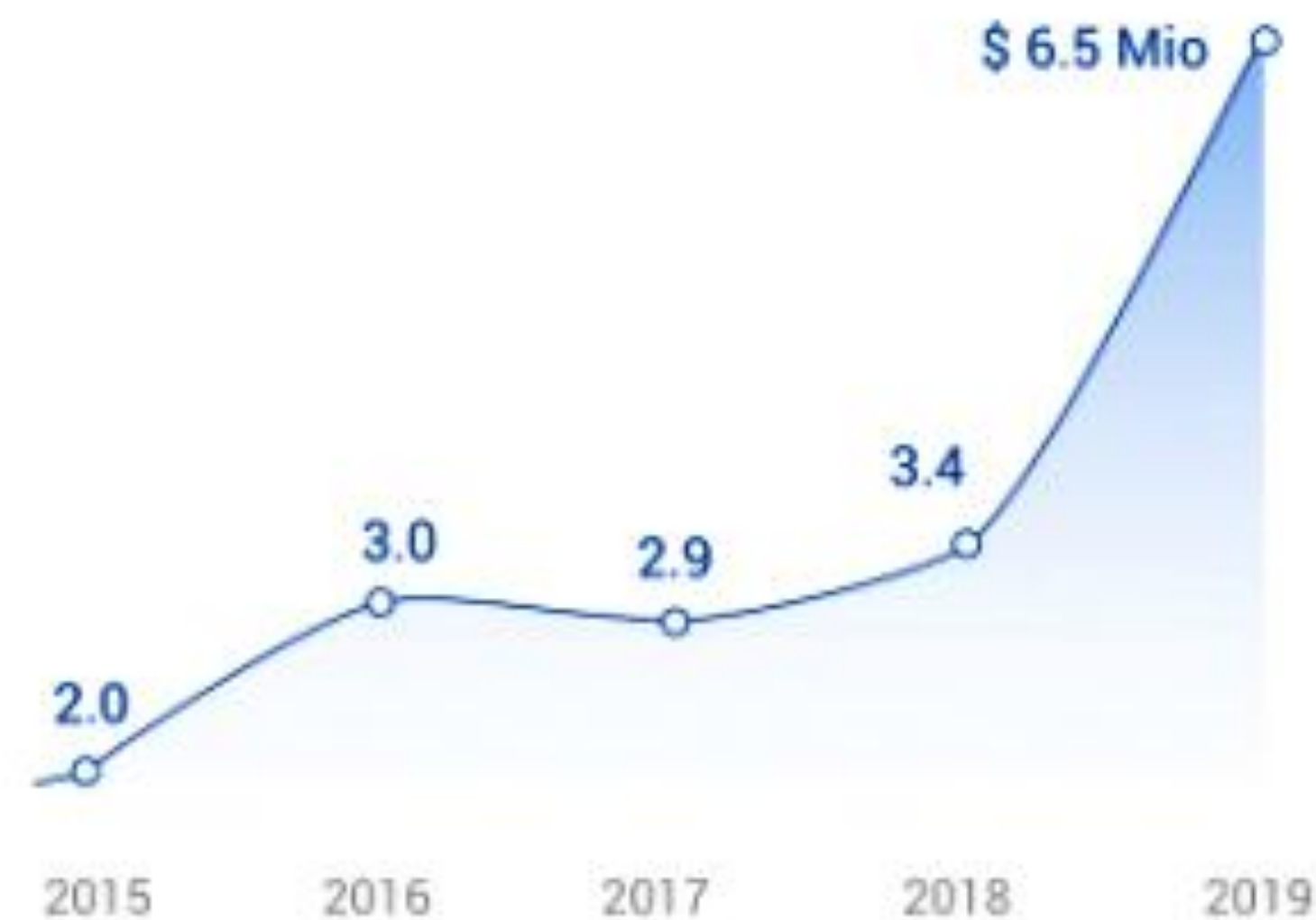
what enabled this recent surge of interest?

- There is a tremendous **need** for automatic vulnerability discovery.

Google has paid security researchers over \$21 million for bug bounties, \$6.5 million in 2019 alone

Emil Protalinski
@EPro
January 28, 2020 12:50 PM

Security
VentureBeat.com



Facebook Paid \$2.2 Million in Bug Bounty Rewards in 2019

By [Ionut Arghire](#) on February 10, 2020

[SecurityWeek.com](#)

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for automatic vulnerability discovery.

Mozilla Security Blog

Firefox has one of the oldest security bug bounties on the internet, dating back to 2004. From 2017-2019, we paid out \$965,750 to researchers across 348 bugs, making the average payout \$2,775 – but as you can see in the graph below, our most common payout was actually \$4,000!

Tom Ritter | April 23, 2020

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for automatic vulnerability discovery.
- We now have the **incentives** and the required mindset.

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for **automatic vulnerability discovery**.
- We now have the **incentives** and the required mindset.

HACKERS EARN RECORD-BREAKING \$100 MILLION ON HACKERONE

May 27, 2020

<https://www.hackerone.com/press-release>

- **214%:** Year-over-year hacker-powered security growth in the federal government
- **85.6%:** The year over year growth in total bounty payments, with 17.5% increase since February when COVID-19 was declared a pandemic.
- **343%:** The increase in signups over the past year on Hacker101 — HackerOne's free online classes for aspiring hackers.
- **Over 170,000:** The number of vulnerabilities hackers have uncovered in nearly 2,000 customer programs

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for automatic vulnerability discovery.
- We now have the **incentives** and the required mindset.
- We now have the **tools** for automatic vulnerability discovery.

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for automatic vulnerability discovery.
- We now have the **incentives** and the required mindset.
- We now have the **tools** for automatic vulnerability discovery.

```
~/libxml2/fuzz on master! © 10:36:36
$ afl-fuzz -i in -o out -- ../.libs/xmllint -o /dev/null @@
```

```
$llvm-gcc -c --emit-llvm -I /opt/klee/include/ maze_klee.c
```

```
====0x40000000: AddressSanitizer: heap-buffer-overflow on address 0x40000000 at
WRITE of size 4 at 0x40000000 thread T0
#0 0x7ff7733cbf85 in fgetsin /tmp/libbsd-0.8.1/src/fgets.c:79
#1 0x401381 in test_fgetsin_single /tmp/libbsd-0.8.1/test/fgets.c:141
#2 0x401301 in test_fgetsin /tmp/libbsd-0.8.1/test/fgets.c:189
#3 0x401301 in main /tmp/libbsd-0.8.1/test/fgets.c:200
#4 0x7ff77338420f in __libc_start_main (/lib64/libc.so.6+0x2002f)
#5 0x401301 in _start (/tmp/libbsd-0.8.1/test/.libs/fgets+0x401301)

0x40000000 is located 0 bytes to the right of 512-byte region [0x40000000,
allocated by thread T0 here:
#0 0x7ff7733843a76 in __interceptor_malloc (/usr/lib/gcc/x86_64-linux-gnu/4
#1 0x7ff7733cbdf7 in fgetsin /tmp/libbsd-0.8.1/src/fgets.c:71

SUMMARY: AddressSanitizer: heap-buffer-overflow /tmp/libbsd-0.8.1/src/fgets.c:79
```

- open-source and freely available.
- easy to use (modulo Matt's concerns 🤔)
- very successful in finding bugs!

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for automatic vulnerability discovery.
- We now have the **incentives** and the required mindset.
- We now have the **tools** for automatic vulnerability discovery.
- **Meaningful engagement** between industry and academia (via open-science) leading to rapid advances in fuzzing!

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for automatic vulnerability discovery.
- We now have the **incentives** and the required mindset.
- We now have the **tools** for automatic vulnerability discovery.
- **Meaningful engagement** between industry and academia (via open-science) leading to rapid advances in fuzzing!



Community
building



Entropic @
ClusterFuzz

Industry
adoption

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for automatic vulnerability discovery.
- We now have the **incentives** and the required mindset.
- We now have the **tools** for automatic vulnerability discovery.
- **Meaningful engagement** between industry and academia (via open-science) leading to rapid advances in fuzzing!



<https://github.com/AFLplusplus>

About

The fuzzer afl++ is afl with community patches, AFLfast power schedules, qemu 3.1 upgrade + laf-intel support, MOpt mutators, InsTrim instrumentation, unicorn_mode, Redqueen and a lot more!

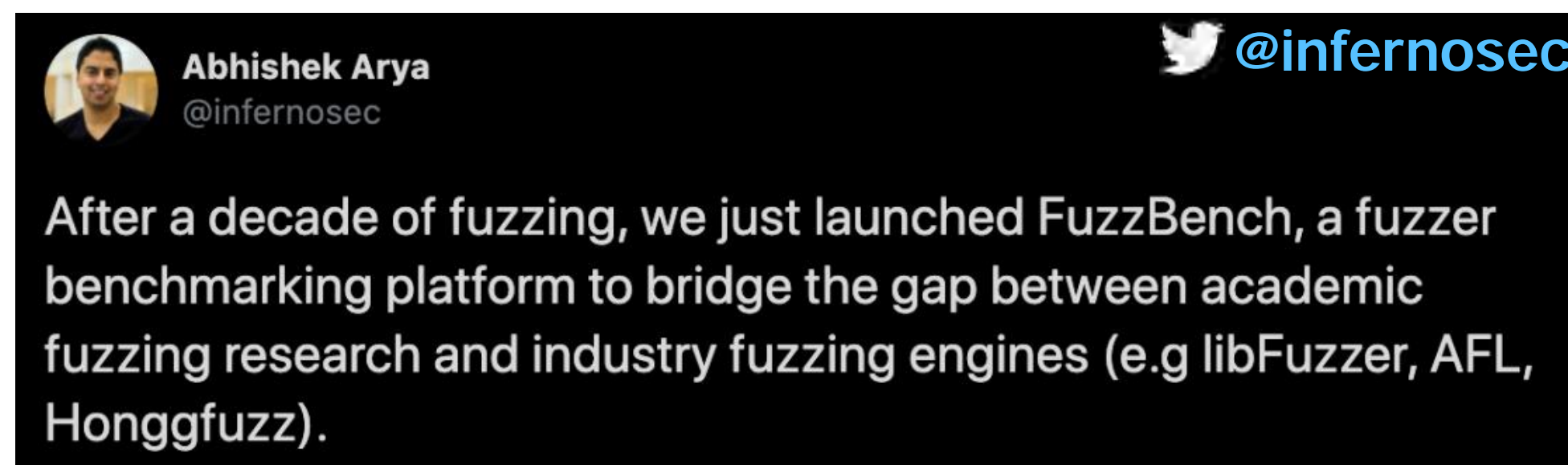
aflplusplus.com

Industry
adoption

Reflections

what enabled this recent surge of interest?

- There is a tremendous **need** for **automatic vulnerability discovery**.
- We now have the **incentives** and the required mindset.
- We now have the **tools** for **automatic vulnerability discovery**.
- **Meaningful engagement** between **industry** and **academia** (via open-science) leading to **rapid advances** in fuzzing!



FuzzBench (compute resources and infrastructure for fuzzer benchmarking)



Paper Reviews et al. (twitch.tv/gamozo)

Challenges



Disclaimer:

We put forward **only questions**.
We have **no answers** (only ideas).

Challenges

- **Automating** vulnerability discovery.
Considered **most important** challenge.

Challenges

- **Automating** vulnerability discovery.
 - [C.1] How can we fuzz **more types of software systems**?

14:45 - 15:15



What's different about fuzzing Automotive Software?

Rakshit Amarnath (Project Lead R&D, Bosch)

17:00 - 17:30



Fuzzing the Solidity compiler

Bhargava Shastry (Security Engineer, Ethereum Foundation)

Challenges

- **Automating** vulnerability discovery.
 - [C.1] How can we fuzz **more types of software systems**?

We know how to fuzz **command line tools** (e.g., AFL).

We know how to fuzz **individual units / functions** (e.g., libfuzzer).

What about **cyber physical systems**, **machine learning systems**,
stateful software, **polyglot software**, **GUI-based software**, .. ?

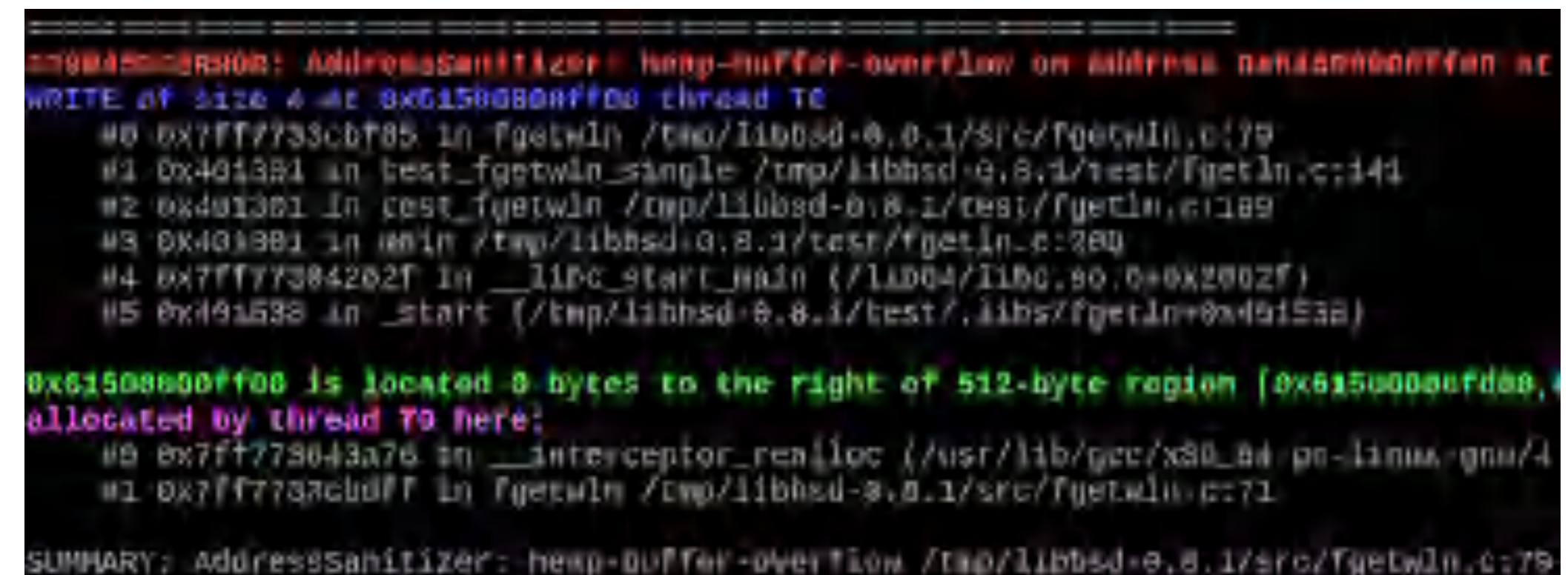
14:45 - 15:15		What's different about fuzzing Automotive Software? Rakshit Amarnath (Project Lead R&D, Bosch)
17:00 - 17:30		Fuzzing the Solidity compiler Bhargava Shastry (Security Engineer, Ethereum Foundation)

Challenges

- **Automating** vulnerability discovery.
 - [C.1] How can we fuzz **more types of software systems**?
 - [C.2] How can the fuzzer identify **more types of vulnerabilities**?

Challenges

- **Automating** vulnerability discovery.
 - [C.1] How can we fuzz **more types of software systems**?
 - [C.2] How can the fuzzer identify **more types of vulnerabilities**?
 - How to **detect** various **side-channels** (incl. information leaks)?
 - How to **detect** **domain-specific vulns.** (incl. sandbox escapes, kernel exploits)?
 - How to **detect** **language-specific vulns**?
 - How to **detect** **other causes** of arbitrary / remote code execution?



```
====  
SUMMARY: AddressSanitizer: heap-buffer-overflow on address 0x4010000000000000 at  
WRITE of size 4 at 0x0150000000000000 thread T0  
#0 0x7ff7733cbf85 in fgetsln /tmp/libbsd-0.8.1/src/fgetsln.c:79  
#1 0x401381 in test_fgetsln_single /tmp/libbsd-0.8.1/test/fgetsln.c:141  
#2 0x401381 in test_fgetsln /tmp/libbsd-0.8.1/test/fgetsln.c:189  
#3 0x401381 in main /tmp/libbsd-0.8.1/test/fgetsln.c:280  
#4 0x7ff77384202f in __libc_start_main (/lib64/libc.so.0+0x2002f)  
#5 0x401538 in _start (/tmp/libbsd-0.8.1/test/.libs/fgetsln+0x401538)  
  
0x0150000000000000 is located 0 bytes to the right of 512-byte region [0x0150000000000000,  
allocated by thread T0 here:  
#0 0x7ff7738043a76 in __interceptor_realloc (/usr/lib/gcc/x86_64-linux-gnu/4  
#1 0x7ff7738c0d0f in fgetsln /tmp/libbsd-0.8.1/src/fgetsln.c:71  
  
SUMMARY: AddressSanitizer: heap-buffer-overflow /tmp/libbsd-0.8.1/src/fgetsln.c:79
```

We need to go beyond memory corruption bugs (ASAN, TSAN).

Challenges

- **Automating** vulnerability discovery.
 - [C.1] How can we fuzz **more types of software systems**?
 - [C.2] How can the fuzzer identify **more types of vulnerabilities**?
 - [C.3] How can we find “**deep bugs**” that have evaded detection?

15:15 - 15:45	○	Structure-aware Grey-box Fuzzing Cornelius Aschermann (Security Researcher, Facebook), Sergej Schumilo (Security Researcher, Ruhr-University Bochum)
15:45 - 16:15	○	Symbolic Execution - what's that, and how to make it efficient? Sebastian Pöplau
17:30 - 18:00	○	Top N challenges of "deep" fuzzing Kostya Serebryany (Principal Software Engineer, Google)
18:00 - 18:30	○	Expanding the Reach of Fuzz Testing Caroline Lemieux (Security Researcher, UC Berkeley)

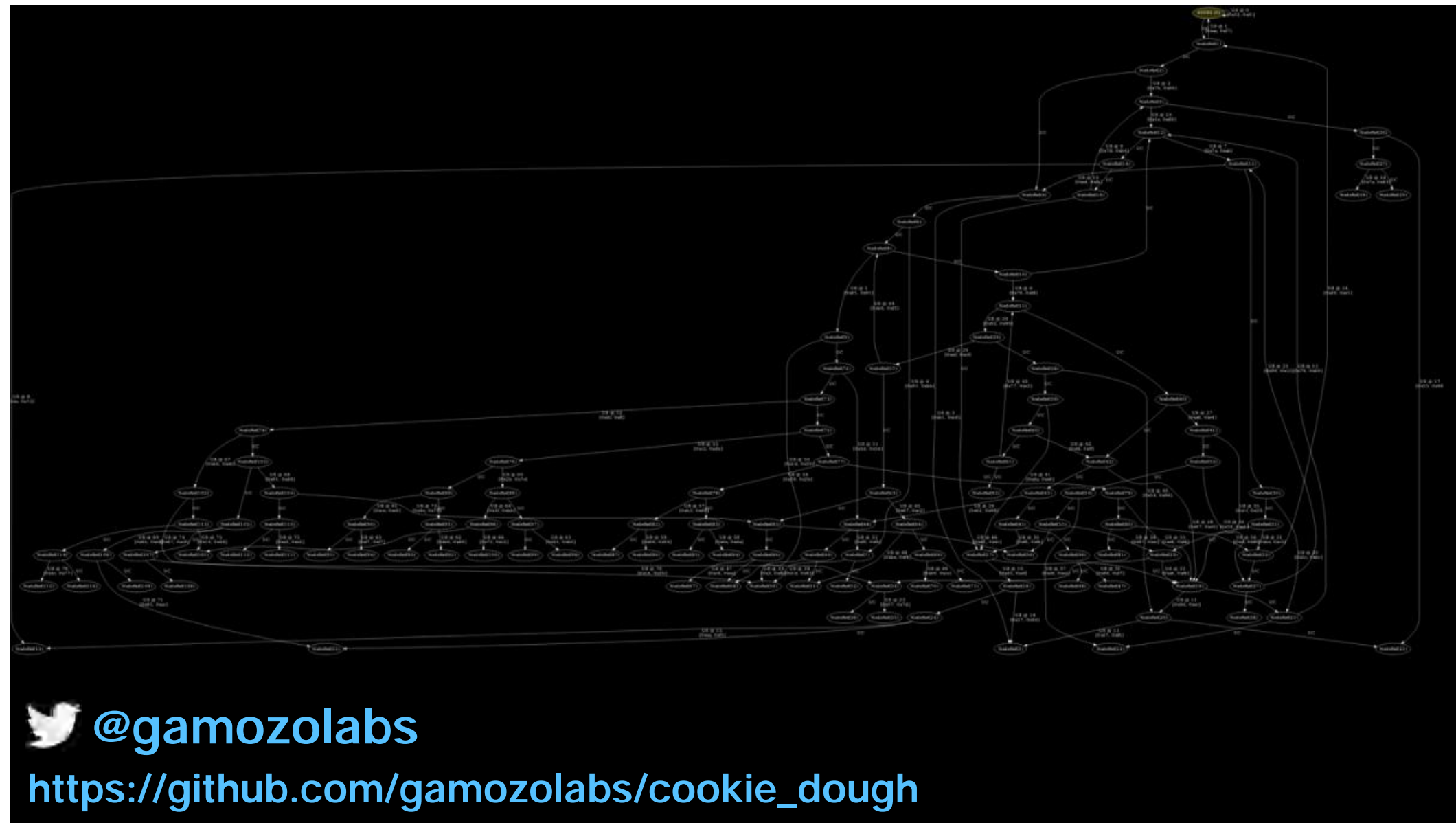
Challenges

- **Automating** vulnerability discovery.
 - [C.1] How can we fuzz **more types of software systems**?
 - [C.2] How can the fuzzer identify **more types of vulnerabilities**?
 - [C.3] How can we find “**deep bugs**” that have evaded detection?
 - How to mine **dictionaries**, **grammars**, and **protocols**?
 - How to identify input dependencies (e.g. checksums)?
 - How identify and rectify **fuzzer roadblocks**?

15:15 - 15:45	○	Structure-aware Grey-box Fuzzing Cornelius Aschermann (Security Researcher, Facebook), Sergej Schumilo (Security Researcher, Ruhr-University Bochum)
15:45 - 16:15	○	Symbolic Execution - what's that, and how to make it efficient? Sebastian Pöplau
17:30 - 18:00	○	Top N challenges of "deep" fuzzing Kostya Serebryany (Principal Software Engineer, Google)
18:00 - 18:30	○	Expanding the Reach of Fuzz Testing Caroline Lemieux (Security Researcher, UC Berkeley)

Challenges

- **Automating** vulnerability discovery.
 - [C.1] How can we fuzz **more types of software systems**?
 - [C.2] How can the fuzzer identify **more types of vulnerabilities**?
 - [C.3] How can we find “**deep bugs**” that have evaded detection?
 - [C.4] What is the **empirical nature** of undiscovered vulnerabilities?



- Which **types of vulnerabilities** are difficult to discover by fuzzing and why?
- What are **fuzzer roadblocks**?

Challenges

- **Automating** vulnerability discovery.
- The **human component** in fuzzing.
 - [C.5] **HITL**: How can fuzzers leverage the **ingenuity** of the auditor?

We need the auditor-in-the-loop.

13:30 - 14:15		Taming Fuzzers Andreas Zeller (Professor, CISPA Helmholtz Center for Information Security)
14:15 - 14:45		Fuzzing Suricata: Finding Vulnerabilities in Large Projects Sirko Höer (Vulnerability Expert, German Federal Office for Information Security)

Challenges

- **Automating** vulnerability discovery.
- The **human component** in fuzzing.
 - [C.5] **HITL**: How can fuzzers leverage the **ingenuity** of the auditor?



@NedWilliamson
Project Zero

1. Write a good **fuzzer harness**
2. **Identify fuzzer roadblocks** (via code coverage).
3. **Patch out** roadblocks.
4. Goto 2 - until vulnerability is found.
5. **Patch back** roadblocks, **"repair" reproducer**.

13:30 - 14:15



Taming Fuzzers

Andreas Zeller (Professor, CISPA Helmholtz Center for Information Security)

14:15 - 14:45



Fuzzing Suricata: Finding Vulnerabilities in Large Projects

Sirko Höer (Vulnerability Expert, German Federal Office for Information Security)

Challenges

- **Automating** vulnerability discovery.
- The **human component** in fuzzing.
 - [C.5] **HITL**: How can fuzzers leverage the **ingenuity** of the auditor?
 - [C.6] **Usability**: How can we improve the **usability** of fuzzing tools?

10:30 - 11:00		Usability Issues of Modern Fuzzers Matthew Smith (Professor, University of Bonn / Fraunhofer FKIE)
11:30 - 12:00		The Human Component in Automated Bug Finding Christian Holler (Staff Security Engineer, Mozilla)
16:30 - 17:00		CI Fuzz - Continuous Fuzzing of Network Services Khaled Yakdan (Chief Scientist, Code Intelligence)

Challenges

- **Automating** vulnerability discovery.
- The **human component** in fuzzing.
 - [C.5] **HITL**: How can fuzzers leverage the **ingenuity** of the auditor?
 - [C.6] **Usability**: How can we improve the **usability** of fuzzing tools?
 - Fuzzing in **Continuous Integration** / Deployment
 - We need** Fuzzing in IDEs (**JUnit-like Fuzzing**)
 - Fuzzing in processes (**Fuzz-driven Development**)

10:30 - 11:00		Usability Issues of Modern Fuzzers Matthew Smith (Professor, University of Bonn / Fraunhofer FKIE)
11:30 - 12:00		The Human Component in Automated Bug Finding Christian Holler (Staff Security Engineer, Mozilla)
16:30 - 17:00		CI Fuzz - Continuous Fuzzing of Network Services Khaled Yakdan (Chief Scientist, Code Intelligence)

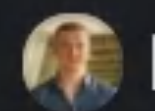
Challenges

- **Automating** vulnerability discovery.
- The **human component** in fuzzing.
 - [C.5] **HITL**: How can fuzzers leverage the **ingenuity** of the auditor?
 - [C.6] **Usability**: How can we improve the **usability** of fuzzing tools?



David Brumley @thedavidbrumley · Sep 24, 2019

Great post. And adding fuzzing during development isn't any harder than adding a unit test. I think in 5 years we'll look back and many will be fuzzing just for QA; not security.



Marcel Böhme @mboehme_ · Sep 24, 2019

Kostya's keynote: LibFuzzer hasn't found new bugs in <big software companie>'s library. We didn't know why. Later we got a note that they are now using LibFuzzer during regression testing in CI and that it prevented 3 vulns from reaching to production.



Challenges

- **Automating** vulnerability discovery.
- The **human component** in fuzzing.
- **Fuzzing theory** and scientific foundations.
Considered **second most important** challenge.

Challenges

- **Automating** vulnerability discovery.
- The **human component** in fuzzing.
- **Fuzzing theory** and scientific foundations.
 - [C.7] How can we assess **residual security risk** if the fuzzing campaign was **unsuccessful**?
 - [C.8] What are **fundamental limitations** of each approach?

How much **more efficient** is an attacker that has
an **order of magnitude** more **computational resources**?

When to stop fuzzing? How to deal with adaptive bias?

We need foundations.

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
 - [C.9] How can we evaluate specialised fuzzers?
 - Works only in a specific program domain
Command line, parser libraries, network protocols, GUIs, browsers, compilers, kernels, Android apps)
 - Focusses on a specific use case
CI/CD [directed fuzzers], specific classes of bugs [UAF, concurrency, deserialization attacks]
 - Suggestion was:
 - Make available special benchmark categories for specialised fuzzers (as in Test-Comp).

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
 - [C.9] How can we evaluate specialised fuzzers?
 - [C.10] How can we prevent overfitting to a specific benchmark?

Goodhart's Law

"When a measure becomes a target, it ceases to be a good measure." —

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
 - [C.9] How can we evaluate **specialised fuzzers**?
 - [C.10] How can we prevent **overfitting** to a specific benchmark?
- Suggestions were:
 1. **Submit and peer-review benchmarks** in addition to fuzzers ([Test-Comp](#)).
 2. Regularly evaluate on **new and unseen benchmarks** ([Rode0Day](#)).
 3. **Continuous evaluation** on a large and **growing** set of diverse, real-world benchmarks ([FuzzBench](#)).

Goodhart's Law

“When a measure becomes a target, it ceases to be a good measure.” —

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
- What is a **good measure** of **fuzzer performance**?
Considered **third most important** challenge.

Evaluation and Benchmarking


Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
- What is a **good measure** of **fuzzer performance**?
 - [C.11] Are synthetic bugs representative?
 - Fuzzer developers can **synthesize** a **large number** of benchmark subjects for their **special use case**, or domain.

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
- What is a **good measure** of **fuzzer performance**?
 - [C.11] Are synthetic bugs representative?
 - Fuzzer developers can **synthesize** a **large number** of benchmark subjects for their **special use case**, or domain.




Brendan Dolan-Gavitt @moyix · Sep 24, 2019
"Time to retire LAVA & CGC, they are actively harmful" – I think at this point I agree about LAVA-M (too small and unrealistic), but I still have hope that improved techniques for bug injection (and newer synthetic datasets) can be useful for evaluations

"Time to retire Lava & CGC, they are actively harmful"
KCC @ Shonan



Brandon Falk @gamosolabs · Sep 6
Of course I'm always biased. But I really like the direction I started down of generating programs. It'll take a lot of work to make them "realistic" but being able to generate a program with many parameters allows us to see how things perform on slightly different programs.



Brandon Falk @gamosolabs · Sep 6
I'm very comfortable with nuance and recognizing there is signal, although the signal is not realistic in programs. But I'm still interested in it as I'm sure some real bugs would be found. These random programs found an RNG bug in honggfuzz where it wouldn't generate past 64k

"I really like the direction [...] of generating programs. [...] These random programs found an RNG bug in honggfuzz."
Brandon Falk @ Twitter

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
- What is a **good measure** of **fuzzer performance**?
 - [C.11] Are synthetic bugs representative?
 - [C.12] Are real bugs representative?
 - Is your set of real bugs **large enough** to be representative?



Magma has 114 CVEs + 4 bugs
in 7 open-source C programs.

A ground-truth binary fuzzing benchmark
suite based on real programs with real bugs.

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
- What is a **good measure** of **fuzzer performance**?
 - [C.11] Are synthetic bugs representative?
 - [C.12] Are real bugs representative?
 - Is your set of real bugs **large enough** to be representative?
 - Are **discovered** bugs representative of **undiscovered** bugs?

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
- What is a **good measure** of **fuzzer performance**?
 - [C.11] Are synthetic bugs representative?
 - [C.12] Are real bugs representative?
 - [C.13] Is code coverage a good measure of fuzzer effectiveness?
 - Measuring coverage achieved is **cheaper** than measuring the number of bugs found.
 - Coverage feedback is the classic measure of progress in greybox fuzzing.
 - If small correlation, how are bugs/vulnerabilities distributed over the code?

We need more empirical studies.

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a **fair fuzzer benchmark**?
- What is a **good measure** of **fuzzer performance**?
 - [C.11] Are synthetic bugs representative?
 - [C.12] Are real bugs representative?
 - [C.13] Is code coverage a good measure of fuzzer effectiveness?
 - [C.14] What is a fair choice of time budget?

We need more empirical studies.

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a fair fuzzer benchmark?
- What is a good measure of fuzzer performance?
- How do we evaluate techniques, not implementations?

Evaluation and Benchmarking

Which fuzzer finds a **larger number** of **important bugs** within a **reasonable time** in software that **we care about**?



Marcel Böhme

@mboehme_

LibFuzzer, AFL++, and Honggfuzz went through major performance improvements -- enabled by FuzzBench.

I'm interested in which strategies work, not which tools.

We are interested in both! FuzzBench is used by fuzzer developers to find the best strategies all the time, e.g., libFuzzer devs noticed AFL did better on one benchmark and thought its handling of seeds might be responsible. So they added a **patched version** of libFuzzer implementing this strategy to see if libFuzzer benefits from this strategy. AFL++ devs continuously experiment and A/B test different strategies (configs). Honggfuzz went through a **series of major improvements** due to such FuzzBench experiments. These developments often happen by focusing on an individual benchmark first, evaluating a single change, similarly to your workflow described in the beginning of your post. One thing that FuzzBench enables is evaluating whether that change or strategy *generalizes* (and didn't just happen to work for a single target). We can tell this by running the experiment on a wide, diverse set of benchmarks, with many trials, so proper statistical analysis and conclusions can be made.

10:34 PM · Sep 5, 2020 · Twitter Web App



The Hacker's Choice @hackerschoice · 3h

Replying to @mboehme_

Without fuzzbench, afl++ would not be where it is now. Or will be. @metzmanj
@infernosec

FuzzBench

- Continuous benchmarking.
- Open-source (Submit PRs).
- Submit your **fuzzer**.
- Submit your **benchmarks**.
- Submit your **feature requests**.
- **Free Compute !!!**

Rode0day

A continuous bug finding competition



hexhi ve Magma

Test-Comp

Tool Competition

And many others...!

Opportunities

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

Opportunities

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

- How do we address this **at scale**?
 - **Open-source, open-science, open discourse**
 - has fostered a **meaningful engagement** between industry and academia,
 - has fostered **tremendous recent advances**
 - in symbolic execution-based whitebox fuzzing, and
 - in coverage-guided greybox fuzzing.

Opportunities

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

- How do we address this **at scale**?
 - **Open-source**, **open-science**, **open discourse**
 - has fostered a **meaningful engagement** between industry and academia,
 - has fostered **tremendous recent advances**
 - in symbolic execution-based whitebox fuzzing, and
 - in coverage-guided greybox fuzzing.

Friday, September 18 • 12:00 - 13:00 @Cppcon

Introducing Microsoft's New Open Source Fuzzing Platform



stryde_hax @stryde_hax · Sep 2

CppCon 9/18: Introducing **Microsoft's** New Open Source **Fuzzing** Platform

sched.co/e7C0 @cppcon #cppcon @sched

Opportunities

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

- How do we address this **at scale**?
 - **Open-source**, **open-science**, **open discourse**.
 - **Educate** developers and students on fuzzing.

Opportunities

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

- How do we address this **at scale**?
 - Open-source, open-science, open discourse.
 - Educate developers and students on fuzzing.
 - Develop educational content, such as tutorials and textbooks.
 - Integrate software security courses into university curriculum.

An ethical hacker about <https://fuzzingbook.com>

Welcome to pwn.college!

pwn.college is a first-stage education platform for students (and other interested parties) to learn about, and practice, core cybersecurity concepts in a hands-on fashion. It is designed to take a “white belt” in cybersecurity to becoming a “yellow belt”, able to approach (simple) CTFs and wargames. The philosophy of pwn.college is “practice makes perfect”.

pwn.college: MOOC-style ASU Computer Systems Security / CTF course



Opportunities

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

- How do we address this **at scale**?
 - **Open-source**, **open-science**, **open discourse**.
 - **Educate** developers and students on fuzzing.
 - **Get organised** and support others.
- As **organization**, take matters into your hands.
 - **Adopt fuzzing** (e.g., in continuous integration).
 - Make your **tools available** as open-source.
 - Establish competitive **bug bounty programs**.
 - Join **cross-organisational security efforts**.
(Open Source Security Foundation; <https://openssf.org/>)



Opportunities

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

- How do we address this **at scale**?
 - **Open-source**, **open-science**, **open discourse**.
 - **Educate** developers and students on fuzzing.
 - **Get organised** and support others.
- As **organization**, take matters into your hands.
- As **individual**, take matters into your hands.
 - **Join the fuzzing community**
 - Submit PRs to Klee, AFL++, LLVM LibFuzzer, OSS-Fuzz,...
 - Make your tools available as open-source.
 - **Organize and support** hackathons, capture-the-flags, hacking clubs, ethical hackers.
 - **Support an open-source project**
(e.g., add it to OSSFuzz or fund it on hackerone.com).



2019 Cyber Security
Challenge Australia (CySCA)

Reflections

- What enabled this recent surge of interest?
 - There is a tremendous need for automatic vulnerability discovery.
 - We now have the incentives and the required mindset.
 - We now have the tools for automatic vulnerability discovery.
 - Meaningful engagement between industry and academia (via open-science) leading to rapid advances in fuzzing!

Evaluation and Benchmarking

Which fuzzer finds a larger number of important bugs within a reasonable time in software that we care about?

- What makes a fair fuzzer benchmark?
- What is a good measure of fuzzer performance?
- How do we evaluate techniques, not implementations?

Challenges

- Automating vulnerability discovery.
- The human component in fuzzing.
- Fuzzing theory and scientific foundations.

Opportunities

The Internet and the world's Digital Economy runs on a shared, critical OSS infrastructure that no one is accountable for.

- How do we address this at scale?
 - Open-source, open-science, open discourse.
 - Educate developers and students on fuzzing.
 - Get organised and support others.