



OSS-Fuzz: 容器和云计算在模糊测试中的应用

韩寒

虚拟化测试工程师
2017-10-22

AGENDA

Fuzzing_[1]

How to contribute to oss-fuzz

Container in testing

OSS-Fuzz_[2]

Demo : add project to oss-fuzz

Open source software severe vulnerabilities

2017

- WPA **KRACK**_[3] - *wpa_supplicant/hostapd*
- Bluetooth **BlueBorne**_[4] - *kernel*
- The **stack slash**_[5] - *sudo,at,ld...etc*

2016

- **Dirty cow**_[6] - *kernel*

2015

- **GHOST**_[7] - *glibc*

2014

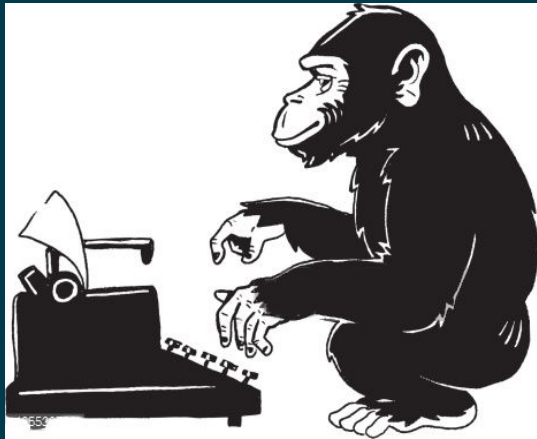
- **Shellshock**_[8] - *bash*
- **Heartbleed**_[9] - *openssl*

**What could we do
to contribute to OSS security?**

Fuzzing (模糊测试)

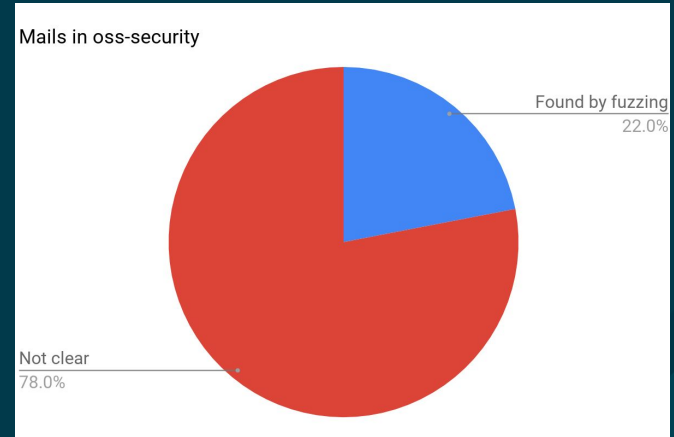
What is fuzzing

Fuzzing is an **automated** software **testing** technique that involves providing **invalid, unexpected, or random** inputs to a program.



Simple but powerful

Near **22%**(58/264 Jul. to Sep.) vulnerabilities are claimed found by fuzzing in **oss-security** [10] mail list.



Fuzzing

How to fuzzing

*Invalid, unexpected, **random***



Crash, overflows, mem leak

Advantages

- Automatically executed.
- Powerful to find security issue.
- Simple to start.

Challenges

- Fuzzing ENVs(fuzzers)
- Execution ENV isolated.
- Resource control



Fuzzing in container

Advantages of container

- Fast & easy deployment
- Lightweight Virtualization
- Cgroup resource control
- Quick boot and shut-off
- Fuzzing ENVs(fuzzers)
- Execution ENV isolated.
- Resource control

Limitations: cannot provide complex environment for **system** or **integration** test.

**What if
fuzzing + container + cloud ?**

OSS-FUZZ

What is oss-fuzzing

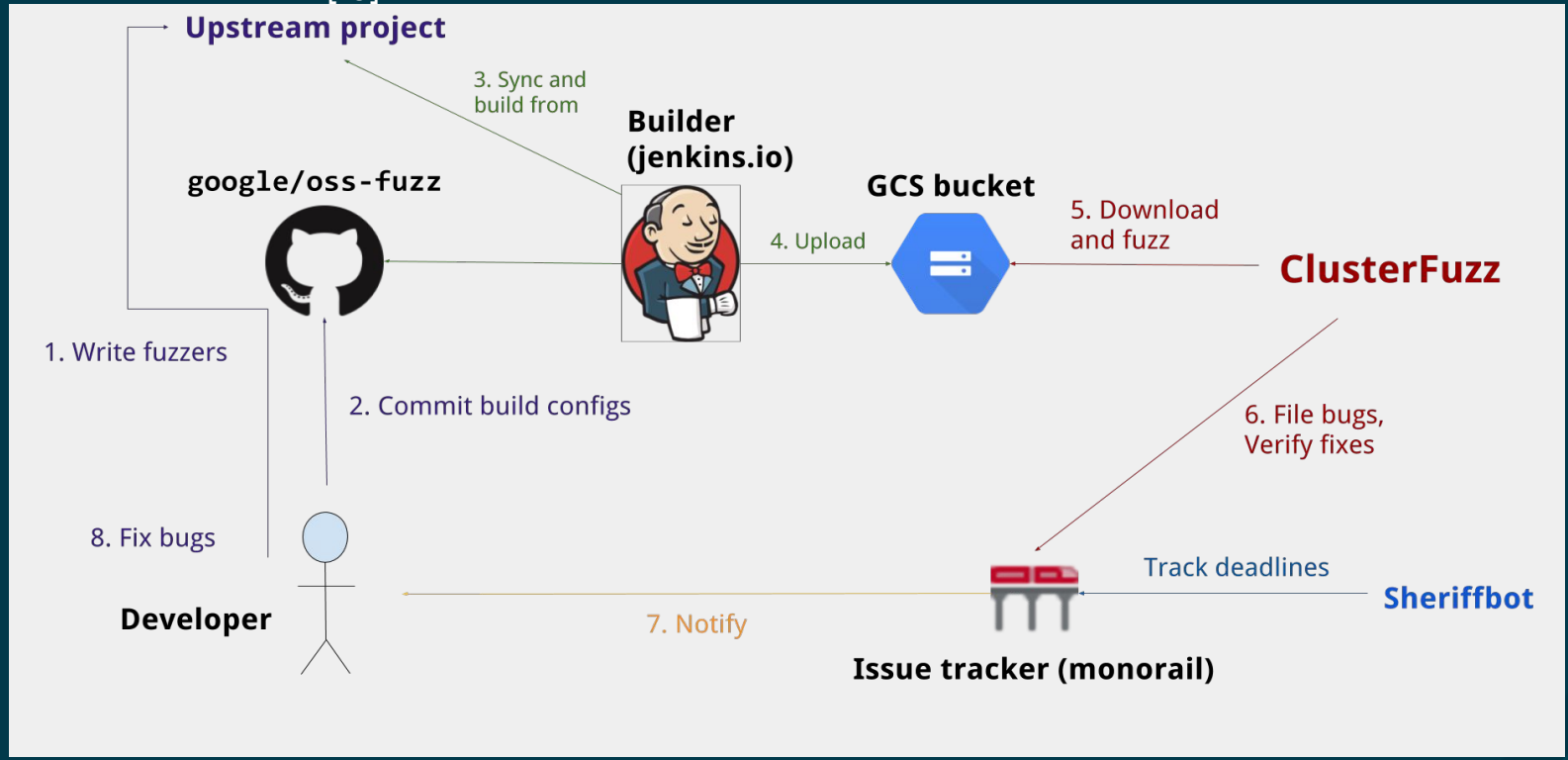
Continuous **fuzzing framework** for open source software in C/C++, aims to make common open source software more **secure and stable** by combining **modern fuzzing** techniques and **scalable distributed execution**.

Features:

- Multiple fuzzer engines: **libFuzzer**_[11], **afl**_[12], **honggfuzz**_[13].
- Multiple sanitizers: **address**_[14]/**memory**_[15]/**undefined**_[16].
- Fuzzing execution and reporting by **ClusterFuzz**_[17].
- Fuzzing under **docker**.

OSS-FUZZ

Process Overview [18]



OSS-FUZZ

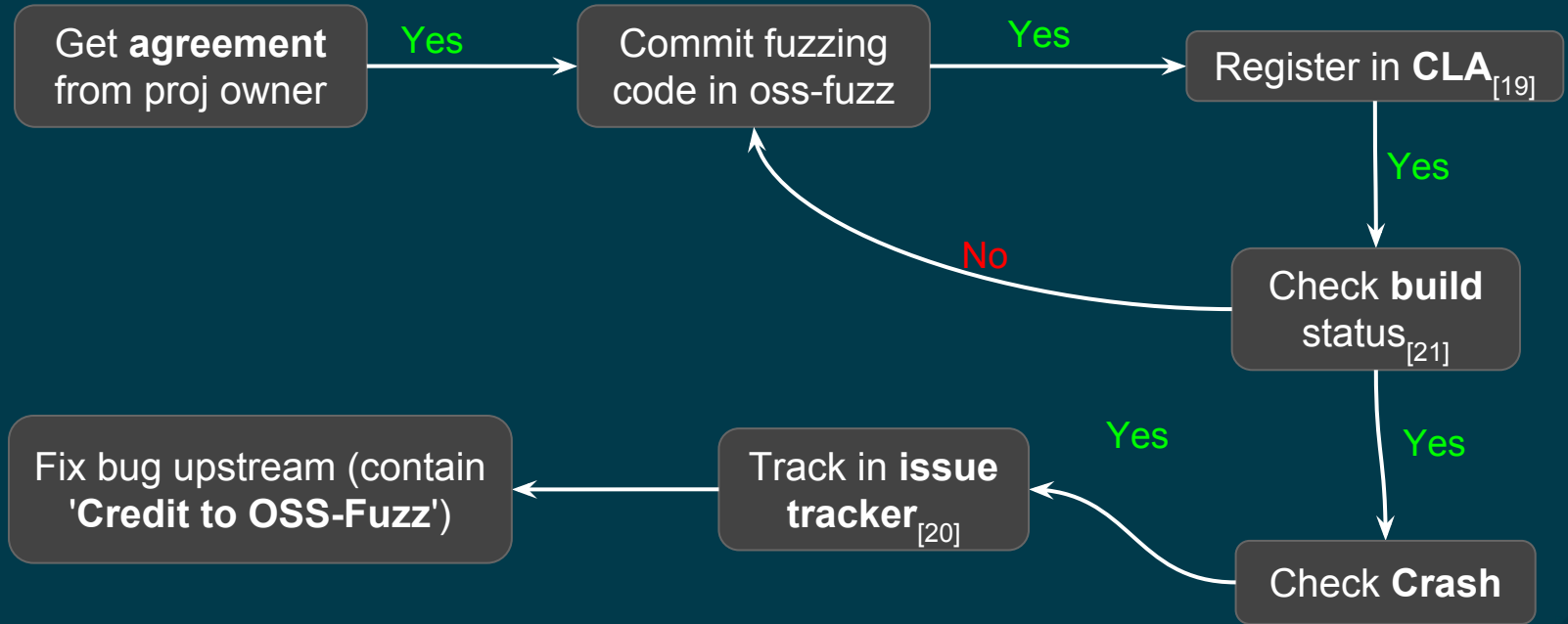
Achievement

From Dec,2016 to Sep,2017:

- **92** open source software projects integrated.
- **1843** non-security bugs found and fixed.
- **585** security bugs found and fixed.

Contribute to oss-fuzz

Developer workflow



Demo: add project to oss-fuzz

Code structure

Project code is located in *projects/<project_name>*

Mandatory files:

- **project.yaml**: project metadata file.
- **Dockerfile**: Provides docker env preparation.
- **build.sh**: Build project and its fuzzer.

Optional files(Or provide in upstream repository):

- **Fuzzer.cc**: Provides fuzzer code.

Demo: add project to oss-fuzz

Project metadata file

Attributes:

- **homepage**: Project's homepage.
- **primary_contact, auto_ccs**: Primary contact and CCs list.
- **sanitizers (optional)** - List of sanitizers to use.
 - **address**: detect memory issues like **mem-leak, buffer overflow, use-after-free, double-free**
 - **memory**: detect **uninitialized reads**.
 - **undefined**: detect **misaligned or null pointer, int or float overflow**

Example:

```
homepage: "https://curl.haxx.se/"
primary_contact: "daniel@haxx.se"
auto_ccs:
  - "daniel.haxx@gmail.com"
sanitizers:
  - address
```

Demo: add project to oss-fuzz

Docker env

Steps:

1. Pull image with clang toolchain and fuzzer engines.
2. Set **maintainer** for this file.
3. Install **required packages** for building project.
4. Pull upstream codes.
5. (Optional) Copy build script and fuzzer codes to container.

Example:

```
FROM gcr.io/oss-fuzz-base/base-builder
MAINTAINER YOUR_EMAIL
RUN apt-get update && apt-get install -y ...
RUN git clone <git_url> <checkout_dir>
WORKDIR <checkout_dir>
COPY build.sh fuzzer.cc $SRC/
```

Demo: add project to oss-fuzz Fuzzer

Build fuzzing code with fuzzing engine(e.g libFuzzer)

Code example(libFuzzer)

```
bool FuzzMe(const uint8_t *Data, size_t DataSize) {  
    return DataSize >= 3 &&  
        Data[0] == 'F' &&  
        Data[1] == 'U' &&  
        Data[2] == 'Z' &&  
        Data[3] == 'Z'; // :-<  
}  
  
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {  
    FuzzMe(Data, Size);  
    return 0;  
}
```

Demo : add project to oss-fuzz

Build script

To build project and fuzzer.

build.sh example:

```
#!/bin/bash -eu

# configure scripts usually use correct environment variables.

./configure --enable-static # build static libraries

make clean

make -j$(nproc) all

$CXX $CXXFLAGS -std=c++11 -I src/ $SRC/parse_fuzzer.cc -o
OUT/parse_fuzzer \ -IFuzzingEngine .libs/libXX.a # build fuzzer with static libraries
```

Demo : add project to oss-fuzz

Testing locally(in oss-fuzz repo dir)

- Build docker image(you may need to set **proxy** first)

```
python infra/helper.py build_image $PROJECT_NAME
```

- Build fuzzer

```
python infra/helper.py build_fuzzers --sanitizer <address/memory/undefined> $PROJECT_NAME
```

- Run fuzzer

```
python infra/helper.py run_fuzzer $PROJECT_NAME <fuzz_target>
```

- Run coverage

```
python infra/helper.py coverage $PROJECT_NAME <fuzz_target>
```


Demo : add project to oss-fuzz

ClusterFuzz


ClusterFuzz is the **distributed** fuzzing execution and reporting infrastructure behind OSS-Fuzz.

Web interface of ClusterFuzz:

- Testcase reports
- Fuzzer stats
- Coverage reports
- Performance analyzer
- Crash stats

Demo : add project to oss-fuzz

ClusterFuzz: build status

<p>! ffmpeg Last built 10/18/2017, 12:32:19 PM</p> <p>! fuchsia_fidl Last built 10/18/2017, 12:19:34 PM</p> <p>! lcms Last built 10/18/2017, 1:23:54 PM</p> <p>! libxml2 Last built 10/18/2017, 12:37:56 PM</p> <p>! llvm Last built 10/18/2017, 1:09:48 PM</p> <p>! open62541 Last built 10/18/2017, 1:39:26 PM</p> <p>! tpm2 Last built 10/18/2017, 1:42:02 PM</p> <p>! woff2 Last built 10/18/2017, 12:21:17 PM</p> <p>arduinojson Last built 10/18/2017, 12:26:41 PM</p> <p>augeas Last built 10/18/2017, 2:32:02 PM</p>	<p> starting build "19bcd6e9-b288-4e7f-b0f9-df5c23492669"</p> <p>FETCHSOURCE BUILD Step #0: Already have image (with digest): gcr.io/cloud-builders/git Starting Step #0 Step #0: Cloning into 'oss-fuzz'... Finished Step #0 Step #1: Already have image (with digest): gcr.io/cloud-builders/docker Starting Step #1 Step #1: Sending build context to Docker daemon 7.68kB Step #1: Step 1/7 : FROM gcr.io/oss-fuzz-base/base-builder Step #1: latest: Pulling from oss-fuzz-base/base-builder Step #1: ae79f2514705: Already exists Step #1: 5ad56d5fc149: Already exists Step #1: 170e558760e8: Already exists Step #1: 395460e233f5: Already exists Step #1: 6f01dc62e444: Already exists Step #1: e27cbc5be051: Pulling fs layer Step #1: 0ca10c4d4511: Pulling fs layer Step #1: 29cb5d7ebba4: Pulling fs layer Step #1: 46578f3fa231: Pulling fs layer Step #1: 4ca48487f2f8: Pulling fs layer Step #1: df8f19b0ad3b: Pulling fs layer Step #1: 3abdd5ee4339: Pulling fs layer Step #1: ce76eaa45e61: Pulling fs layer Step #1: c7c1cb16d0fd: Pulling fs layer Step #1: 466bf86194c3: Pulling fs layer Step #1: 019d5cc70427: Pulling fs layer Step #1: efe3ff1d832f: Pulling fs layer Step #1: df8f19b0ad3b: Waiting Step #1: 3abdd5ee4339: Waiting Step #1: 019d5cc70427: Waiting Step #1: 46578f3fa231: Waiting Step #1: ce76eaa45e61: Waiting Step #1: 466bf86194c3: Waiting Step #1: 4ca48487f2f8: Waiting Step #1: c7c1cb16d0fd: Waiting Step #1: 0ca10c4d4511: Download complete Step #1: 46578f3fa231: Download complete Step #1: 29cb5d7ebba4: Verifying Checksum Step #1: 29cb5d7ebba4: Download complete Step #1: e27cbc5be051: Verifying Checksum Step #1: e27cbc5be051: Download complete Step #1: df8f19b0ad3b: Download complete Step #1: ce76eaa45e61: Verifying Checksum</p>
--	---

Demo: add project to oss-fuzz

Cluster fuzz: Crash report

[CREATE ISSUE](#)[REDO TASK](#)

OVERVIEW

Crash State: **NULL**

Crash Type: **Use-of-uninitialized-value**

Fuzzer: **libFuzzer_augeas_escape_name_fuzzer**

Security: **YES (Medium)**

Crash Address: **---**

Job Type: **libfuzzer_msan_augeas**

Reproducible: **NO**

Issue: **None**

Platform: **linux**


Fixed: **NA** 

Created: **Wed, Oct 18, 2017, 8:36 PM**

Sanitizer: **memory (MSAN)**

Project: **augeas**

Deletion: **Will be auto-deleted on 10/25/2017 if flaky crash no longer seen**

Minimized Testcase: **NA** Unminimized Testcase:   (0 B) Re-upload testcase: 

You can reproduce this crash painlessly with [our reproduce tool](#). For Googlers, install [the required libraries](#) and run `prodaccess && /google/data/ro/teams/clusterfuzz-tools/releases/clusterfuzz reproduce 6442776221712384`. For non-Googlers, see [the installation section](#). Report any issues at clusterfuzz-dev@chromium.org.

FIXED REVISION RANGE

NA

REGRESSION REVISION RANGE

NA

Demo: add project to oss-fuzz

ClusterFuzz: Code coverage

File	Coverage
<i>Files with zero coverage are not shown.</i>	
<u>/src/augeas/src/augeas.c</u>	027%
<u>/src/augeas/src/builtin.c</u>	007%
<u>/src/augeas/src/errcode.c</u>	015%
<u>/src/augeas/src/info.c</u>	022%
<u>/src/augeas/src/internal.c</u>	006%
<u>/src/augeas/src/internal.h</u>	029%
<u>/src/augeas/src/memory.c</u>	073%
<u>/src/augeas/src/pathx.c</u>	044%
<u>/src/augeas/src/ref.c</u>	075%
<u>/src/augeas/src/syntax.c</u>	021%
<u>/src/augeas/escape_name_fuzzer.cc</u>	090%
<u>/usr/local/include/c++/v1/memory</u>	100%
<u>/usr/local/include/c++/v1/string</u>	100%

Demo : add project to oss-fuzz

ClusterFuzz:Performance

date	perf_report	logs	tests_executed	total_crashes	new_crashes	known_crashes	edge_cov
Oct 18, 2017	Performance	Logs	676,631,973	0	0	0	26.99% (1561/5783)

func_cov	cov_report	corpus_sizeSize of the minimized corpus generated based on code coverage (number of testcases and total size on disk)	corpus_bakup	avg_exec_per_sec	new_tests_added	slowest_test_time_sec	peak_memory_mb	oom_count
42.19% (154/365)	Coverage	501 (69 KB)	Download	392.178	15,616	0	135	0

Demo: add project to oss-fuzz

Issue tracker

	ID ▾	Type ▾	Component ▾	Status ▾	Proj ▾	Reported ▾	Owner ▾	Summary + Labels ▾
☆	112	Bug	----	New	libchewing	----	----	libchewing: record ClusterFuzz Reproducible
☆	209	Bug	----	New	libchewing	----	----	libchewing: pgdata->chiSymbolBufLen >= pgdata->chiSymbolCursor ClusterFuzz Reproducible
☆	226	Bug	----	New	expat	----	mmoroz@google.com	Direct-leak in addBinding ClusterFuzz Reproducible
☆	359	Bug	----	New	libarchive	----	----	libarchive: Out-of-memory in libarchive_fuzzer ClusterFuzz Reproducible
☆	373	Bug	----	New	libchewing	----	----	libchewing: pgdata->choiceInfo.oldChiSymbolCursor <= pgdata->chiSymbolBufLen ClusterFuzz Reproducible
☆	570	Bug	----	New	libchewing	----	----	libchewing: chiSymbolCursor < ARRAY_SIZE(pgdata->preeditBuf) ClusterFuzz Reproducible
☆	572	Bug	----	New	libprotobuf-mutator	----	----	libprotobuf-mutator: Direct-leak in xz_error ClusterFuzz Reproducible
☆	589	Bug	----	New	grpc	----	----	grpc: Undefined-shift in gpr_stack_lockfree_push ClusterFuzz Reproducible
☆	602	Bug	----	New	libass	----	----	libass: Integer-overflow in parse_tag ClusterFuzz Reproducible
☆	603	Bug	----	New	libass	----	----	libass: Integer-overflow in ass_lazy_track_init ClusterFuzz Reproducible
☆	613	Bug	----	New	tpm2	----	----	tpm2: Undefined-shift in ObjectAllocateSlot ClusterFuzz Reproducible
☆	621	Bug	----	New	tpm2	----	----	tpm2: Undefined-shift in TPMTI_DH_CONTEXT_Unmarshal ClusterFuzz Reproducible
☆	623	Bug	----	New	tpm2	----	----	tpm2: Undefined-shift in TPMTI_DH_CONTEXT_Unmarshal ClusterFuzz Reproducible
☆	624	Bug	----	New	libass	----	----	libass: Integer-overflow in parse_tag ClusterFuzz Reproducible
☆	632	Bug	----	New	libtsm	----	----	libtsm: Crash in tsm_screen_tab_left ClusterFuzz Reproducible

Reference

1. **Fuzzing**: <https://en.wikipedia.org/wiki/Fuzzing>
2. **oss-fuzz**: <https://github.com/google/oss-fuzz>
3. **Krack**: <https://www.krackattacks.com/>
4. **BlueBorne**: <https://www.armis.com/blueborne/>
5. **Stack slash**: <https://www.qualys.com/2017/06/19/stack-clash/stack-clash.txt>
6. **Dirty cow**: <https://dirtycow.ninja/>
7. **GHOST**: <https://access.redhat.com/articles/1332213>
8. **Shellshock**: [https://en.wikipedia.org/wiki/Shellshock_\(software_bug\)](https://en.wikipedia.org/wiki/Shellshock_(software_bug))
9. **Heartbleed**: <http://heartbleed.com/>
10. **oss-security**: <http://www.openwall.com/lists/oss-security/>
11. **libFuzzer**: <https://llvm.org/docs/LibFuzzer.html>
12. **Afl**: <http://lcamtuf.coredump.cx/afl/>
13. **honggfuzz**: <https://github.com/google/honggfuzz>
14. **AddressSanitizer**: <https://clang.llvm.org/docs/AddressSanitizer.html>
15. **MemorySanitizer**: <https://clang.llvm.org/docs/MemorySanitizer.html>
16. **UndefinedBehaviorSanitizer**: <https://clang.llvm.org/docs/UndefinedBehaviorSanitizer.html>
17. **ClusterFuzz**: <https://github.com/google/oss-fuzz/blob/master/docs/clusterfuzz.md>
18. **Process Overview**: <https://github.com/google/oss-fuzz#process-overview>
19. **CLA(Contributor License Agreement)**: <https://cla.developers.google.com/>
20. **Issue tracker**: <https://bugs.chromium.org/p/oss-fuzz/issues/list>
21. **ClusterFuzz build status**: <https://oss-fuzz-build-logs.storage.googleapis.com/index.html>

Summary

- What is fuzzing
- Advantages of fuzzing with container and cloud
- What is oss-fuzz
- How to contribute to oss-fuzz

Q&A



THANK YOU