Fuzzing: The Next Unit Testing

Kostya Serebryany <kcc@google.com>

May 2017

Agenda

- Why fuzz
    - Detour: the Sanitizers
- How to fuzz
    - libFuzzer
- Live demo (30+ minutes)
    - “Hello World” & Heartbleed
    - Seed corpus, parallel fuzzing, dictionaries
    - Detecting logical bugs
    - Structured fuzzing
- How to fuzz continuously
    - OSS-Fuzz

Testing vs Fuzzing

```
void TestMyApi() {
    MyApi(Input1);
    MyApi(Input2);
    MyApi(Input3);
}
```

```
void FuzzMyApi() {
    int N = 10000000;
    for (int i = 0; i < N; i++) {
        MyApi(GenerateInput());
    }
}
```

Why Fuzz

- Bugs specific to C/C++ that require e.g. the *sanitizers* to catch:
    - Use-after-free, buffer overflows, Uses of uninitialized memory, Memory leaks
- Arithmetic bugs:
    - Div-by-zero, Int/float overflows, bitwise shifts by invalid amount
- Plain crashes:
    - NULL dereferences, Uncaught exceptions
- Concurrency bugs:
    - Data races, Deadlocks
- Resource usage bugs:
    - Memory exhaustion, hangs or infinite loops, infinite recursion (stack overflows)
- Logical bugs:
    - Discrepancies between two implementations of the same protocol (example)
    - Assertion failures

Detour: the Sanitizers

- AddressSanitizer (ASan)
    - Use-after-free, buffer overflow, memory leaks, etc
- ThreadSanitizer (TSan)
    - Data races and deadlocks
- MemorySanitizer (MSan)
    - Use of uninitialized memory
- UndefinedBehaviorSanitizer (UBSan)
    - Integer overflow, bitwise shifts by invalid amount, many more kinds of UB

ASan report example: use-after-free

```
int main(int argc, char **argv) {
   int *array = new int[100];
   delete [] array;
   return array[argc]; } // BOOM
% clang++ -O1 -fsanitize=address a.cc && ./a.out
==30226== ERROR: AddressSanitizer heap-use-after-free
READ of size 4 at 0x7faa07fce084 thread T0
   #0 0x40433c in main a.cc:4
0x7faa07fce084 is located 4 bytes inside of 400-byte region
freed by thread T0 here:
   #0 0x4058fd in operator delete[](void*) _asan_rtl_
   #1 0x404303 in main a.cc:3
previously allocated by thread T0 here:
   #0 0x405579 in operator new[](unsigned long) _asan_rtl_
   #1 0x4042f3 in main a.cc:2
```

Fuzzing strategies

- Grammar-based Generation
  - Generate random inputs according to grammar rules
- Blind mutations
  - Collect a corpus of representative inputs, apply random mutations to them
- Coverage-Guided fuzzing
  - Build the target code with coverage instrumentation
  - Run the target on the initial test corpus, collect coverage
  - Run the target on random mutations of the elements of the corpus
  - If new coverage is discovered add the mutation back to the corpus
    - Repeat

Fuzz Target - API of fuzzing engines

```
bool TargetAPI(const uint8_t* Data, size_t Size) {
   if (Size >= 3)
      return Data[0] == 'F' &&
             Data[1] == 'U' &&
             Data[2] == 'Z' &&
             Data[3] == 'Z';
    return true;
}
extern "C" int LLVMFuzzerTestOneInput(const uint8_t* Data, size_t Size) {
    TargetAPI(Data, Size);
    return 0;
}
```

libFuzzer - an engine for guided in-process fuzzing

- libFuzzer: a library; provides main()
- Build your target code with extra compiler flags
- Link your target with libFuzzer
- Pass a directory with the initial test corpus and run

```
% clang++ -g my-code.cc libFuzzer.a -o my-fuzzer \
  -fsanitize=address -fsanitize-coverage=trace-pc-guard


% ./my-fuzzer MY_TEST_CORPUS_DIR
```

tutorial.libFuzzer.info

Structured fuzzing

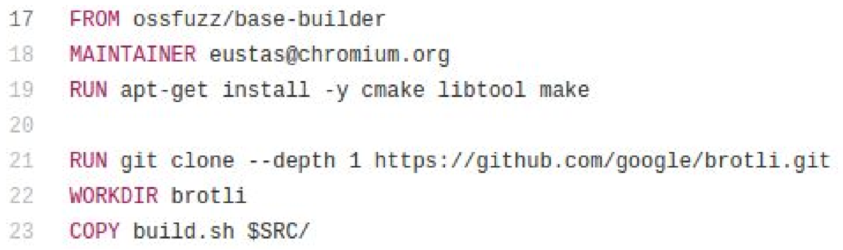
- Define your input as a protobuf message
- Use libFuzzer with https://github.com/google/libprotobuf-mutator

OSS-Fuzz - continuous fuzzing service for OSS

- 3 tiny config files:
  - Docker image
  - Settings (e-mails, etc)
  - Builds script

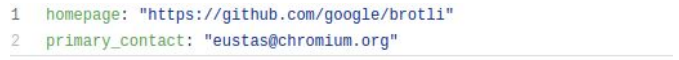
- OSS-Fuzz will fuzz 24/7
  - libFuzzer, AFL, Radamsa
  - ASan, UBSan, MSan

- Beta stage: accepting only “widely used” projects.

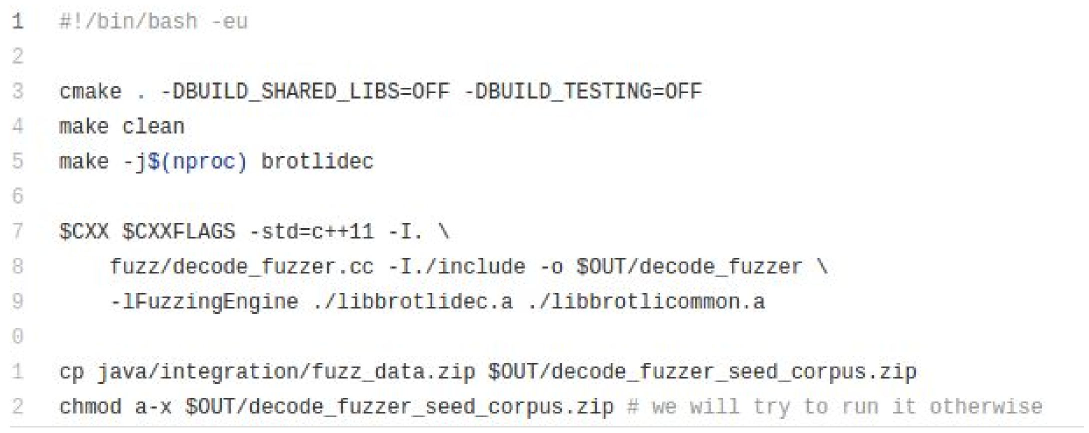


```
FROM ossfuzz/base-builder
MAINTAINER eustas@chromium.org
RUN apt-get install -y cmake libtool make

RUN git clone --depth 1 https://github.com/google/brotli.git
WORKDIR brotli
COPY build.sh $SRC/
```



```
homepage: "https://github.com/google/brotli"
primary_contact: "eustas@chromium.org"
```



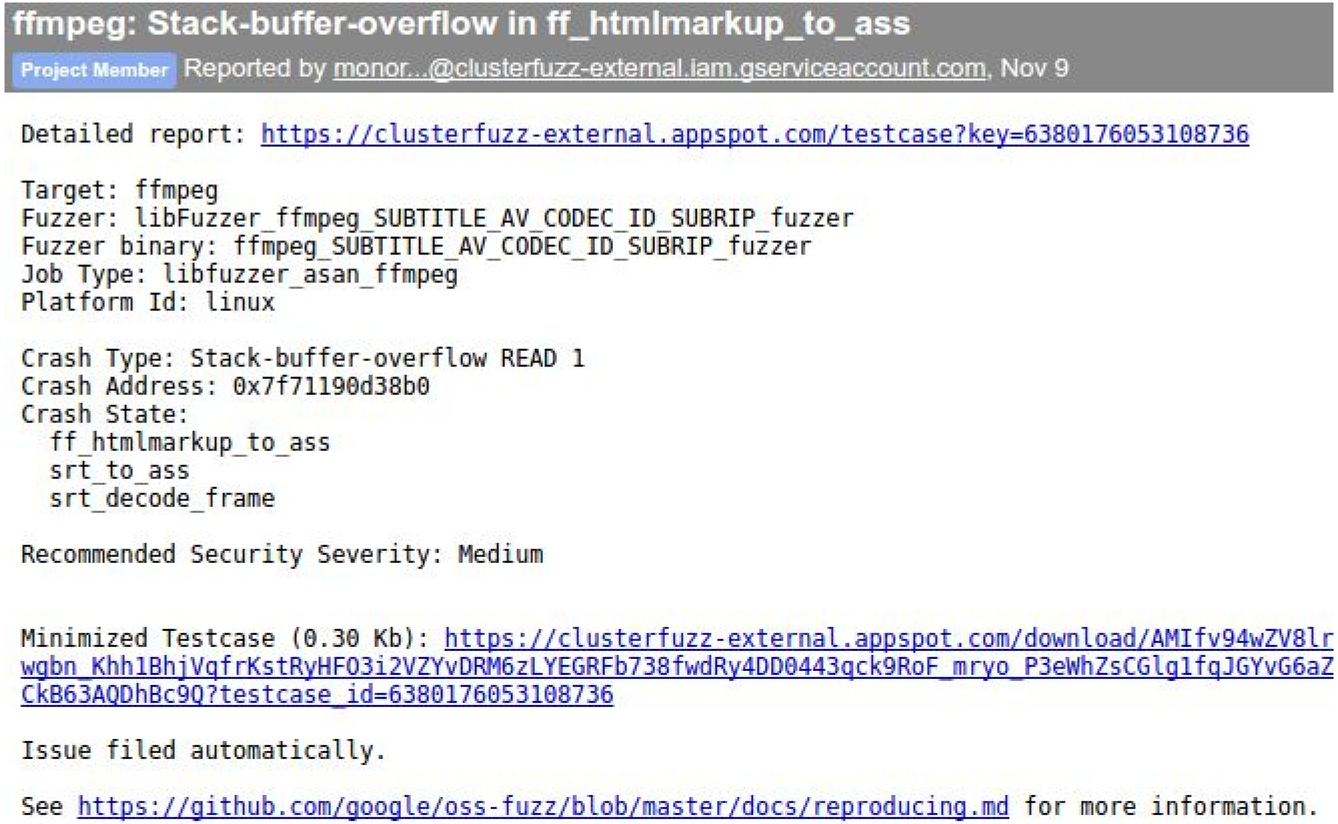
```
#!/bin/bash -eu

cmake . -DBUILD_SHARED_LIBS=OFF -DBUILD_TESTING=OFF
make clean
make -j$(nproc) brotlidec

$CXX $CXXFLAGS -std=c++11 -I. \
    fuzz/decode_fuzzer.cc -I./include -o $OUT/decode_fuzzer \
    -lFuzzingEngine ./libbrotlidec.a ./libbrotlicommon.a

cp java/integration/fuzz_data.zip $OUT/decode_fuzzer_seed_corpus.zip
chmod a-x $OUT/decode_fuzzer_seed_corpus.zip # we will try to run it otherwise
```

OSS-Fuzz automatically files bugs



ffmpeg: Stack-buffer-overflow in ff_htmlmarkup_to_ass

Project Member Reported by monor...@clusterfuzz-external.iam.gserviceaccount.com, Nov 9

```
Detailed report: https://clusterfuzz-external.appspot.com/testcase?key=6380176053108736

Target: ffmpeg
Fuzzer: libFuzzer_ffmpeg_SUBTITLE_AV_CODEC_ID_SUBRIP_fuzzer
Fuzzer binary: ffmpeg_SUBTITLE_AV_CODEC_ID_SUBRIP_fuzzer
Job Type: libfuzzer_asan_ffmpeg
Platform Id: linux

Crash Type: Stack-buffer-overflow READ 1
Crash Address: 0x7f71190d38b0
Crash State:
  ff_htmlmarkup_to_ass
  srt_to_ass
  srt_decode_frame

Recommended Security Severity: Medium


Minimized Testcase (0.30 Kb): https://clusterfuzz-external.appspot.com/download/AMIfv94wZV8lr
wgbn_Khh1BhjVqfrKstRyHFO3i2VZYvDRM6zLYEGRFb738fwdRy4DD0443qck9RoF_mryo_P3eWhZsCGlg1fqJGYvG6aZ
CkB63AQDhBc9Q?testcase_id=6380176053108736

Issue filed automatically.

See https://github.com/google/oss-fuzz/blob/master/docs/reproducing.md for more information.
```

Fuzz targets as regression tests

- Testing is still cheaper and more reliable for continuous integration

- Fuzz targets + corpus make a great regression test

- Example: openssl/fuzz
  - Same fuzz targets as used for fuzzing
  - Minimized corpus stored in git & periodically updated
  - Tested in openssl’s CI on every commit

Summary

- Fuzzing is often more powerful than unit testing
- libFuzzer makes fuzzing easy, even for structured data
- OSS-Fuzz makes continuous fuzzing easy

Q & A

libFuzzer.info

tutorial.libFuzzer.info

github.com/google/oss-fuzz