OSS-Fuzz: Fuzzing Everything

Abhishek Arya // Google

Fabian Meumertzheim // Code Intelligence

FuzzCon Europe

March 24, 2021

About Us

Abhishek Arya

- Principal Engineer, Google
- Founder, OSS-Fuzz
- Manager, Google Open Source Security Team

Fabian Meumertzheim

- Software Engineer, Code Intelligence
- Maintainer of several OSS projects
- Contributor, Chromium

OSS-Fuzz: Goals

Find **security** and **stability bugs** with **modern fuzzing** and **scalable distributed execution**

Give developers an **easy self-service fuzzing platform** and encourage quality integrations with **rewards program**

Ensure that all security vulnerabilities are **fixed** within a **reasonable timeframe**

OSS-Fuzz: History

- Dec 2016: Launched with C/C++ support and sample projects
- May 2017: OSS-Fuzz Integration Rewards for developers
- Feb 2019: Open-sourced ClusterFuzz infra that powers OSS-Fuzz
- Sep 2019: Go fuzzing support using go-fuzz
- Mar 2020: FuzzBench: Fuzzer Benchmarking as a Service
- Jun 2020: Rust fuzzing support using cargo-fuzz
- Summer 2020: Fuzzing Internships for Open Source Software
- Dec 2020: Python fuzzing support using Google's Atheris
- Mar 2021: Java fuzzing support using Code Intelligence's Jazzer **(NEW!)**

Google Open Source Blog

The latest news from Google on open source releases, major projects, events, and student outreach programs.

Announcing OSS-Fuzz: Continuous fuzzing for open source software

Thursday, December 1, 2016

We are happy to announce OSS-Fuzz, a new Beta program developed over the past years with the Core Infrastructure Initiative community. This program will provide continuous fuzzing for select core open source software.

Open source software is the backbone of the many apps, sites, services, and networked things that make up "the internet." It is important that the open source foundation be stable, secure, and reliable, as cracks and weaknesses impact all who build on it.

Recent security stories confirm that errors like buffer overflow and use-after-free can have serious, widespread consequences when they occur in critical open source software. These errors are not only serious, but notoriously difficult to find via routine code audits, even for experienced developers. That's where fuzz testing comes in. By generating random inputs to a given program, fuzzing triggers and helps uncover errors quickly and thoroughly.

Usual Fuzzing Workflow

- Learn build system
- Write fuzz targets
- Create builds with memory instrumentation
- Run fuzz targets at scale
- Analyze, de-duplicate and file crashes
- Verify vulnerability fixes
- Notify consumers of fixed vulnerabilities

Simplified Fuzzing Workflow

Dev

- ~~Learn build system~~
- **Write fuzzer unittest (<100 LoC)**

OSS-Fuzz

- ✓ Create builds with memory instrumentation
- ✓ Run fuzz targets at scale
- ✓ Analyze, de-duplicate and file crashes
- ✓ Verify vulnerability fixes

OSV

- ✓ Notify consumers of fixed vulnerabilities

OSS-Fuzz: Results

- **420** OSS projects
- **6,000** security vulnerabilities
- **22,000** functional bugs
- **100,000** cpu cores

libxml2

Available for: Windows 7 and later

Impact: Multiple issues in libxml2

Description: Multiple memory corruption issues were addressed with improved input validation.

CVE-2019-8749: found by OSS-Fuzz

CVE-2019-8756: found by OSS-Fuzz

Sec Bug #77831 **Heap-buffer-overflow in exif_iif_add_value in EXIF**

Submitted: 2019-04-02 06:44 UTC **Modified:** 2019-04-15 06:53 UTC
From: stas@php.net **Assigned:** stas (profile)
Status: Closed **Package:** EXIF related
PHP Version: 7.1.27 **OS:** *
Private report: No **CVE-ID:** 2019-11035

View | Add Comment | Developer | Edit

[2019-04-02 06:44 UTC] stas@php.net

```
Description:
------------
Another OSS-Fuzz bug: https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=13938
```

GNUTLS-SA-2017-3	CVE-2017-7869	Memory corruption	It was found using the OSS-FUZZ fuzzer infrastructure that decoding a specially crafted OpenPGP certificates could lead to (**A**) an integer overflow, resulting to an invalid memory write, (**B**) a null pointer dereference resulting to a server crash, and (**C**) a large allocation, resulting to a server out-of-memory condition. These affect only applications which utilize the OpenPGP certificate functionality of GnuTLS. The issues were fixed in 3.5.10. **Recommendation:** The support of OpenPGP certificates in GnuTLS is considered obsolete. As such, it is not recommended to use OpenPGP certificates with GnuTLS. To address the issues found upgrade to GnuTLS 3.5.10 or later versions.

Collaboration with

code intelligence

Jazzer – A Modern Fuzzer for Java

- Coverage-guided: based on libFuzzer & JaCoCo
- In-process: very fast (~1M exec/s on empty target)
- Bytecode instrumentation: no sources required

github.com/CodeIntelligenceTesting/jazzer

Jazzer's Advanced Features

Tried and tested libFuzzer features:

- Minimization with deduplication
- Parallel fuzzing in fork mode
- Value profile

Additional Java smarts:

- Pure Java reproducers
- "Keep going" mode
- Method hooking framework

Cracking All the Checks

```java
public class ExampleFuzzer {
  private static String base64(byte[] input) {
    return Base64.getEncoder().encodeToString(input);
  }
  private static long insecureEncrypt(long input) {
    long key = 0xefe4eb93215cb6b0L;
    return input ^ key;
  }
  public static void fuzzerTestOneInput(FuzzedDataProvider data) {
    if (base64(data.consumeBytes(6)).equals("SmF6emVy")) return;
    long[] plaintextBlocks = data.consumeLongs(2);
    if (plaintextBlocks.length != 2) return;
    if (insecureEncrypt(plaintextBlocks[0]) != 0x9fc48ee64d3dc090L) return;
    if (insecureEncrypt(plaintextBlocks[1]) != 0x888a82ff483ad9c2L) return;
    throw IllegalStateException("not reached");
  }
}
```

Cracking All the Checks

```
$ ./jazzer --cp=. --target_class=ExampleFuzzer -use_value_profile=1
...
#2426817     REDUCE cov: 12 ft: 324 corp: 291/5492b lim: 4096 exec/s: 606704 rss: 312Mb L: 22/27 MS: 1 EraseBytes-
#2477313     REDUCE cov: 12 ft: 324 corp: 291/5490b lim: 4096 exec/s: 619328 rss: 316Mb L: 22/27 MS: 1 EraseBytes-
#2506214     REDUCE cov: 12 ft: 324 corp: 291/5489b lim: 4096 exec/s: 501242 rss: 319Mb L: 23/27 MS: 1 EraseBytes-
#2561055     REDUCE cov: 12 ft: 324 corp: 291/5488b lim: 4096 exec/s: 512211 rss: 324Mb L: 22/27 MS: 1 EraseBytes-

== Java Exception: java.lang.IllegalStateException: not reached
      at ExampleFuzzer.fuzzerTestOneInput(ExampleFuzzer.java:17)
DEDUP_TOKEN: 985e7866de639615
== libFuzzer crashing input ==
MS: 1 ChangeBinInt-; base unit: 176d76cac0d2d25c007fdffd6758a992a4fe919f
0x4a,0x61,0x7a,0x7a,0x65,0x72,0x20,0x76,0x61,0x6c,0x75,0x65,0x20,0x70,0x72,0x6f,0x66,0x69,0x6c,0x69,0x6e,0x67,
Jazzer value profiling
artifact_prefix='/tmp/'; Test unit written to /tmp/crash-293dc3aec0ffd72fbdf63dce2a853976a787920c
Base64: SmF6emVyIHZhbHVlIHByb2ZpbGluZw==
reproducer_path='/tmp'; Java reproducer written to /tmp/Crash_293dc3aec0ffd72fbdf63dce2a853976a787920c.java
```

Why Fuzz Java?

Functional bugs:

- Uncaught exceptions
- Assertions
- Inconsistent implementations (*differential fuzzing*)

Security issues:

- Infinite loops
- OutOfMemoryError
- XSS (Frameworks, Sanitizers, ...)
- RCE (Jakarta, Serialization, ...)
- ...

json-sanitizer build passing

Given JSON-like content, The JSON Sanitizer converts it to valid JSON.

Output

The output is well-formed JSON as defined by RFC 4627. The output satisfies these additional properties:

- The output will not contain the substrings (case-insensitively) `"<script"`, `"</script"` or `"<!--"` and can thus be embedded inside an HTML script element without further encoding.

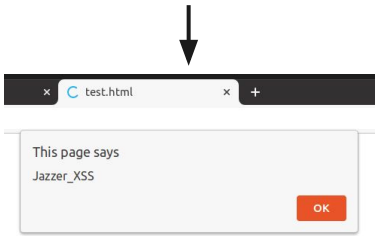
```
<script>
  const config = {"foo": true, "bar": [1, 2]};
  doThings(config);
</script>
```

```
public static void fuzzerTestOneInput(FuzzedDataProvider data) {
  String input = data.consumeRemainingAsString();
  String safeJson;
  try {
    safeJson = JsonSanitizer.sanitize(input, 10);
  } catch (Exception e) {
    return;
  }
  assert !safeJson.contains("</script")
      : new FuzzerSecurityIssueHigh("Output contains </script");
}
```

XSS in json-sanitizer (CVE-2021-23899)

```
b</\script><\script>alert(`Jazzer_XSS`);//
```

```
<script>
  const config = "b</script><script>alert(`Jazzer_XSS`);//";
  doThings(config);
</script>
```



```
public byte[] readFully(int length) throws IOException {
  byte[] b = new byte[length];
  int bytesRead = read(b);
  while (bytesRead < length)
  {
    bytesRead += read(b, bytesRead, length - bytesRead);
  }
  return b;
}
```

Where could the bug hide?

Infinite Loop in Apache PDFBox (CVE-2021-27807)

```
public byte[] readFully(int length) throws IOException {
  byte[] b = new byte[length];
  int bytesRead = read(b);
  while (bytesRead < length)
  {
    bytesRead += 0;
  }
  return b;
}
```

Infinite Loop in Apache PDFBox (CVE-2021-27807)

```
public static void fuzzerTestOneInput(byte[] input) {
  try {
    Loader.loadPDF(input, null, null, null,
      MemoryUsageSetting.setupMainMemoryOnly(100_000_000));
  } catch (IOException ignored) { }
}
```

Infinite Loop in Apache Commons Compress

```
public static void fuzzerTestOneInput(byte[] input) {
  try {
    new TarFile(input).close();
  } catch (IOException ignored) {}
}
```

`TarFile` allowed negative sizes for file metadata.

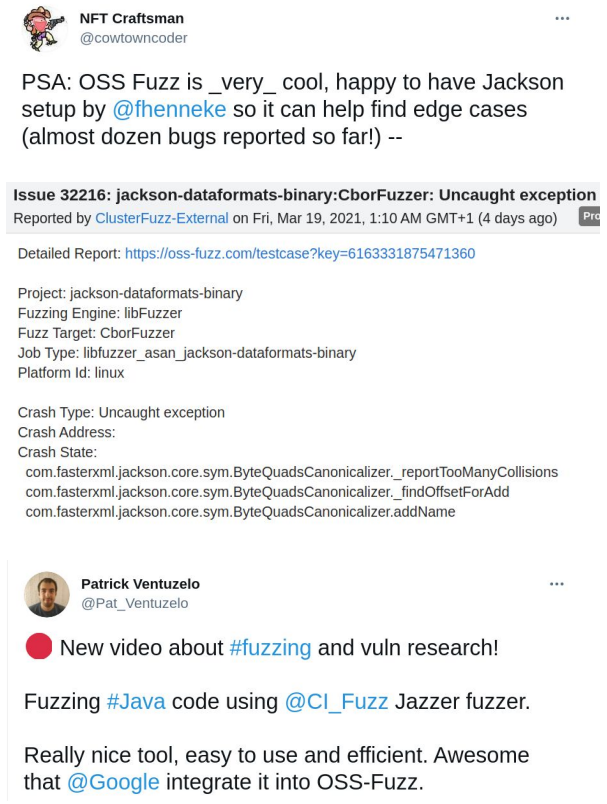
Jazzer produced a .tar with claimed metadata size `-METADATA_LENGTH`.

Infinite Loop in Apache Commons Compress

```
java.lang.OutOfMemoryError: Java heap space
    at java.base/sun.nio.cs.UTF_8.newDecoder(UTF_8.java:70)
    at org.apache.commons.compress.archivers.zip.NioZipEncoding.newDecoder(NioZipEncoding.java:182)
    at org.apache.commons.compress.archivers.zip.NioZipEncoding.decode(NioZipEncoding.java:135)
    at org.apache.commons.compress.archivers.tar.TarUtils.parseName(TarUtils.java:311)
    at org.apache.commons.compress.archivers.tar.TarUtils.parseName(TarUtils.java:275)
    at org.apache.commons.compress.archivers.tar.TarArchiveEntry.parseTarHeader(TarArchiveEntry.java:1550)
    at org.apache.commons.compress.archivers.tar.TarArchiveEntry.<init>(TarArchiveEntry.java:554)
    at org.apache.commons.compress.archivers.tar.TarArchiveEntry.<init>(TarArchiveEntry.java:570)
    at org.apache.commons.compress.archivers.tar.TarFile.getNextTarEntry(TarFile.java:250)
    at org.apache.commons.compress.archivers.tar.TarFile.<init>(TarFile.java:211)
    at org.apache.commons.compress.archivers.tar.TarFile.<init>(TarFile.java:94)
```

Jazzer: Results

- **>50** bugs found
- **>15** OSS projects fuzzed
- **8** security issues
- **5** CVEs



NFT Craftsman
@cowtowncoder

PSA: OSS Fuzz is _very_ cool, happy to have Jackson setup by @fhenneke so it can help find edge cases (almost dozen bugs reported so far!) --

Issue 32216: jackson-dataformats-binary:CborFuzzer: Uncaught exception
Reported by ClusterFuzz-External on Fri, Mar 19, 2021, 1:10 AM GMT+1 (4 days ago)

Detailed Report: https://oss-fuzz.com/testcase?key=6163331875471360

Project: jackson-dataformats-binary
Fuzzing Engine: libFuzzer
Fuzz Target: CborFuzzer
Job Type: libfuzzer_asan_jackson-dataformats-binary
Platform Id: linux

Crash Type: Uncaught exception
Crash Address:
Crash State:
com.fasterxml.jackson.core.sym.ByteQuadsCanonicalizer._reportTooManyCollisions
com.fasterxml.jackson.core.sym.ByteQuadsCanonicalizer._findOffsetForAdd
com.fasterxml.jackson.core.sym.ByteQuadsCanonicalizer.addName

Patrick Ventuzelo
@Pat_Ventuzelo

New video about #fuzzing and vuln research!

Fuzzing #Java code using @CI_Fuzz Jazzer fuzzer.

Really nice tool, easy to use and efficient. Awesome that @Google integrate it into OSS-Fuzz.

Thank You!

code intelligence