Fuzzing Java Code

Tobias "floyd" Ospelt

5#

PENTAGRID

5#

PENTAGRID

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

CYBERSECURITY
LIECHTENSTEIN

Topics

About the tools

Fuzzing

AFL

JQF-AFL

Bug classes

JQF-Zest

AFL

Michal Zalewski

AFL++

Forks

AFL security issues

IJG jpeg, libjpeg-turbo, libpng, libtiff, mozjpeg, PHP, Mozilla Firefox, Internet Explorer, Apple Safari, Adobe Flash / PCRE, sqlite, OpenSSL, LibreOffice, poppler , freetype, GnuTLS, GnuPG, OpenSSH, PuTTY, ntpd, nginx, bash (post-Shellshock), tcpdump, JavaScriptCore, pdfium, ffmpeg, libmatroska, libarchive , wireshark, ImageMagick, BIND , QEMU, lcms, Oracle BerkeleyDB, Android / libstagefright, iOS / ImageIO, FLAC audio library, libsndfile, less / lesspipe, strings (+ related tools), file, dpkg, rcs, systemd-resolved, libyaml, Info-Zip unzip, libtasn, OpenBSD pfctl, NetBSD bpf, man & mandoc, IDA Pro, clamav, libxml, glibc, clang / llvm, nasm, ctags, mutt, procmail, fontconfig, pdksh, Qt , wavpack, redis / lua-cmsgpack, taglib, privoxy, perl , libxmp, radare, SleuthKit, fwknop, X.Org, exifprobe, jhead, capnproto, Xerces-C, metacam, djvulibre, exiv, Linux btrfs, Knot DNS, curl, wpa_supplicant, libde [reported by author], dnsmasq, libbpg, lame, libwmf, uudecode, MuPDF, imlib, libraw, libbson, libsass, yara, WC tidy-html, VLC, FreeBSD syscons, John the Ripper, screen, tmux, mosh, UPX, indent, openjpeg, MMIX, OpenMPT, rxvt, dhcpcd, Mozilla NSS, Nettle, mbed TLS, Linux netlink, Linux ext, Linux xfs, botan, expat, Adobe Reader, libav, libical, OpenBSD kernel, collectd, libidn, MatrixSSL, jasper , MaraDNS, wm, Xen, OpenH, irssi, cmark, OpenCV, Malheur, gstreamer , Tor, gdk-pixbuf, audiofile , zstd, lz, stb, cJSON, libpcre, MySQL, gnulib, openexr, libmad, ettercap, lrzip, freetds , Asterisk, ytnef, raptor, mpg, Apache httpd, exempi, libgmime, pev, Linux mem mgmt, sleuthkit, Mongoose OS, iOS kernel,

JQF

Rohan Padhye

Usability

JQF-AFL and JQF-Zest

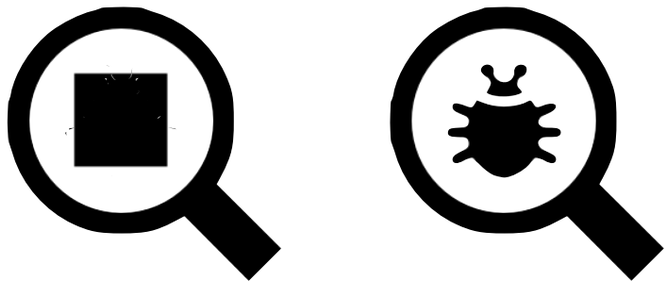
5#

PENTAGRID

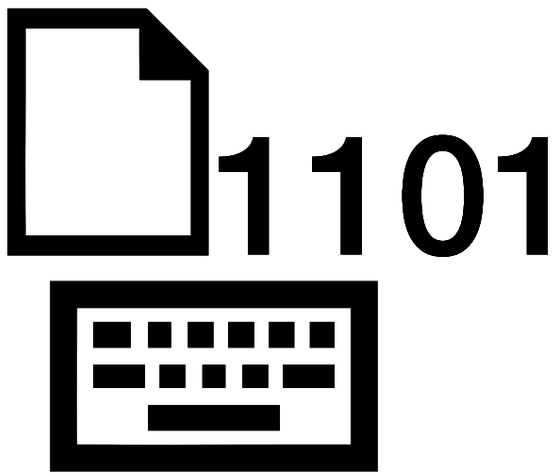
JQF bugs

Google Closure Compiler
OpenJDK
Mozilla Rhino
Apache Commons
Apache Maven
Apache BCEL
Apache PDFBox, Apache Tika

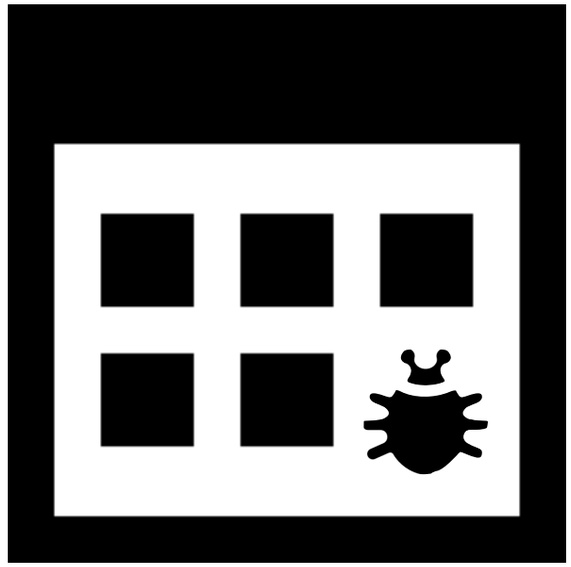
Fuzzing



Instrumentation



Input

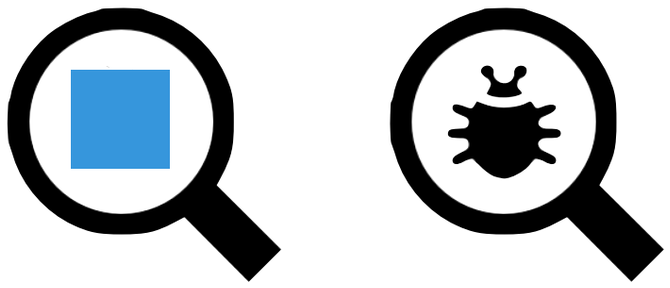


Program

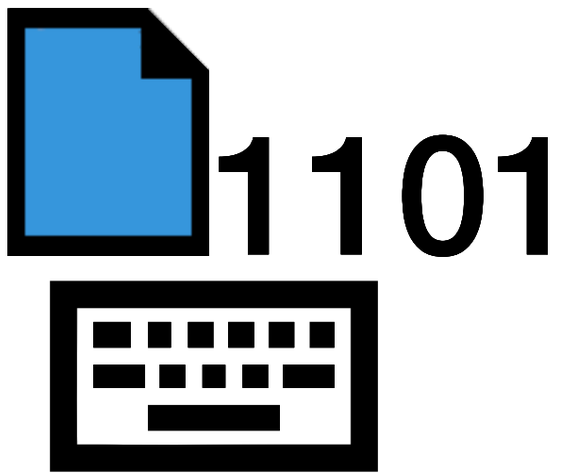
5#

PENTAGRID

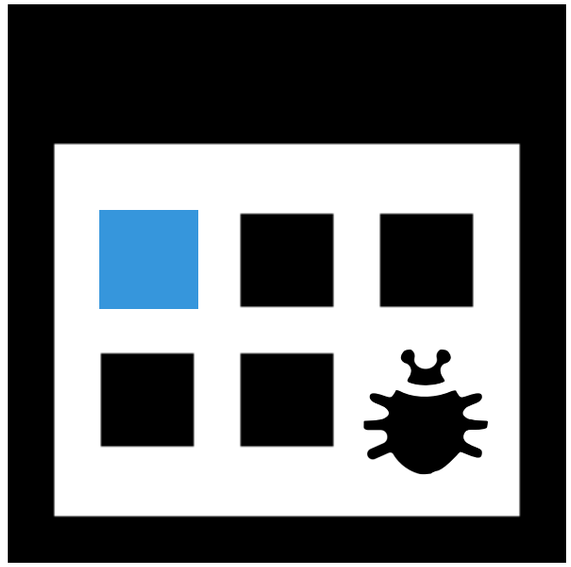
AFL



Instrumentation



Input



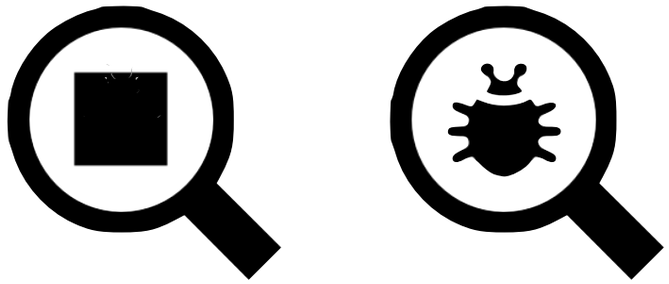
Program

5#

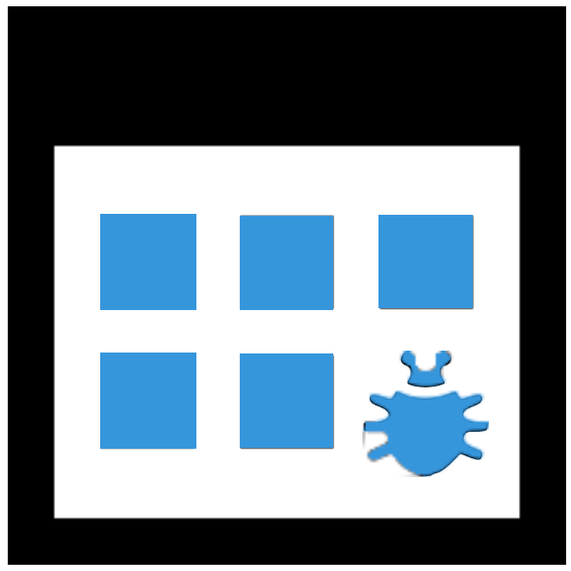
PENTAGRID

AFL

Input



Instrumentation

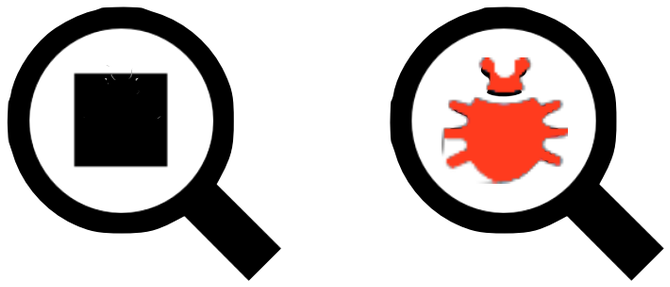


libpng

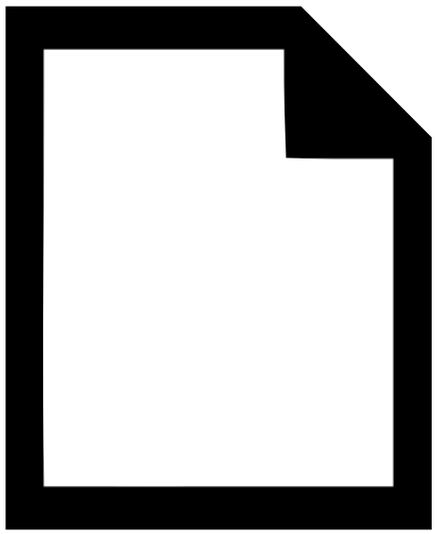
```
                        american fuzzy lop 2.38b

┌─ process timing ─────────────────────┬─ overall results ─────┐
│        run time : 2 days, 21 hrs, 14 min, 3 sec │  cycles done : 5      │
│   last new path : 0 days, 5 hrs, 17 min, 16 sec │  total paths : 871    │
│ last uniq crash : 0 days, 16 hrs, 12 min, 2 sec │ uniq crashes : 124    │
│  last uniq hang : 0 days, 13 hrs, 59 min, 12 sec │   uniq hangs : 221    │
├─ cycle progress ─────────────────────┼─ map coverage ────────┤
│  now processing : 162* (18.60%)      │    map density : 0.50% / 7.03%    │
│ paths timed out : 33 (3.79%)         │ count coverage : 2.07 bits/tuple  │
├─ stage progress ─────────────────────┼─ findings in depth ───┤
│  now trying : interest 32/8          │ favored paths : 2647 (303.90%)    │
│ stage execs : 42.8k/102k (41.90%)    │  new edges on : 93 (10.68%)       │
│ total execs : 5.71M                  │ total crashes : 78.7k (124 unique)│
│  exec speed : 35.60/sec (slow!)      │   total hangs : 550k (221 unique) │
├─ fuzzing strategy yields ────────────┴─ path geometry ──────┤
│   bit flips : 206/483k, 60/483k, 8/482k  │   levels : 7               │
│  byte flips : 2/60.4k, 0/53.1k, 0/53.1k  │  pending : 730             │
│ arithmetics : 35/1.69M, 1/646k, 0/238k   │  pend fav : 0              │
│  known ints : 2/60.8k, 1/302k, 2/251k    │ own finds : 211            │
│  dictionary : 0/0, 0/0, 5/104k           │  imported : 175            │
│       havoc : 3/8338, 0/0                │ stability : 40.15%         │
│        trim : 16.62%/31.7k, 12.14%       └────────────────────────────┘
└──────────────────────────────────────────────────────────────────────┘
                                                          [cpu000:151%]
```

AFL

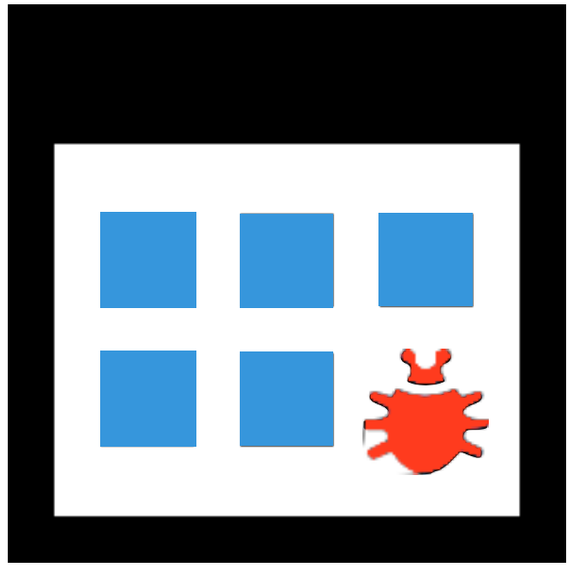
```
overall results
cycles done : 5
 total paths : 871
uniq crashes : 124
  uniq hangs : 221
```



Instrumentation



Input



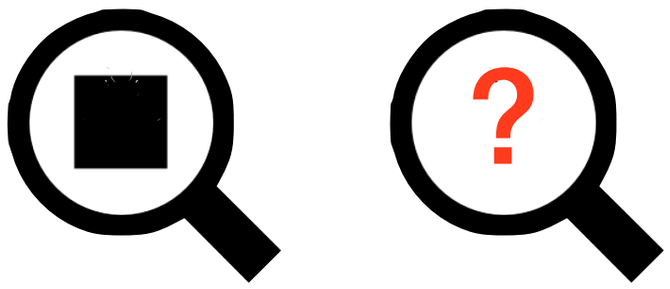
libpng

5#

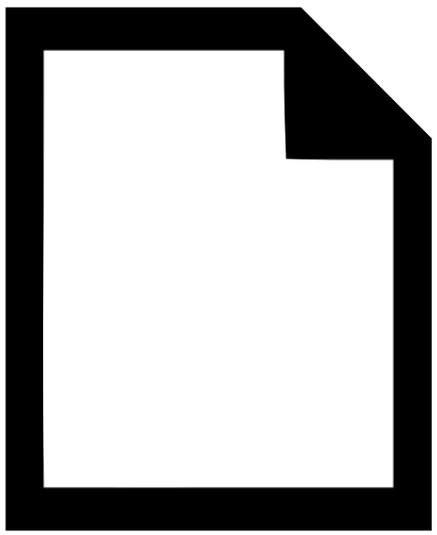
PENTAGRID

JQF-AFL

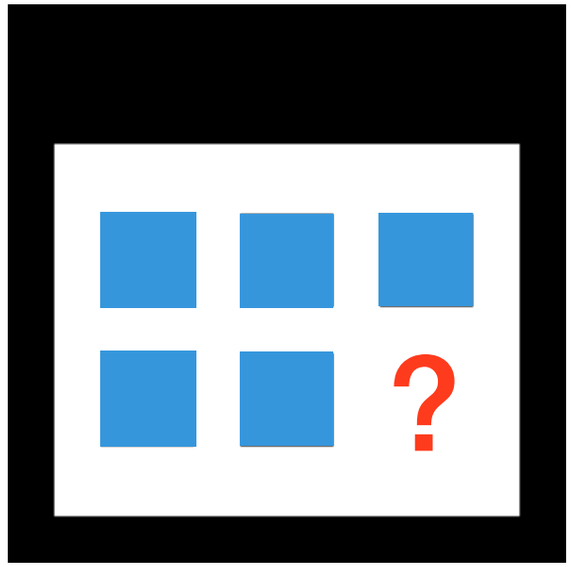
```
overall results
cycles done : 5
 total paths : 871
uniq crashes : 124
  uniq hangs : 221
```



Instrumentation



Input



Java ImageIO

5#

PENTAGRID

https://github.com/rohanpadhye/jqf/wiki/Fuzzing-with-AFL

Here is an example of a test driver that tests the in-built PNG image decoding logic in the JDK ImageIO library:

```
import javax.imageio.ImageIO;
import javax.imageio.ImageReader;
import javax.imageio.stream.ImageInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

import edu.berkeley.cs.jqf.fuzz.Fuzz;
import edu.berkeley.cs.jqf.fuzz.JQF;
import org.junit.Assume;
import org.junit.runner.RunWith;

@RunWith(JQF.class)
public class PngTest {

    @Fuzz /* JQF will generate inputs to this method */
    public void testRead(InputStream input)  {
        // Create parser
        ImageReader reader = ImageIO.getImageReadersByFormatName("png").next();

        // Decode image from input stream
        try {
            reader.setInput(ImageIO.createImageInputStream(input));

            // Bound dimensions to avoid OOM
            Assume.assumeTrue(reader.getHeight(0) <= 256);
            Assume.assumeTrue(reader.getWidth(0) <= 256);

            // Decode first image in the input stream
```

5#

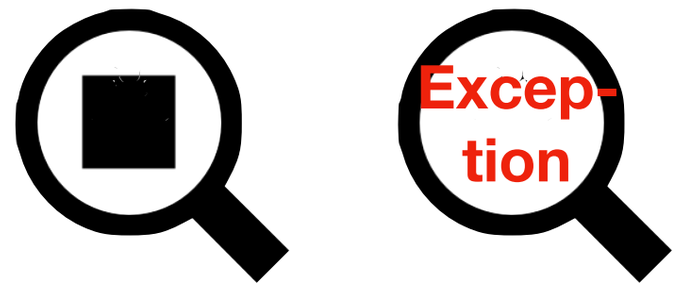
5#

IndexOutOfBoundsException: Attempt to read past end of image sequence!

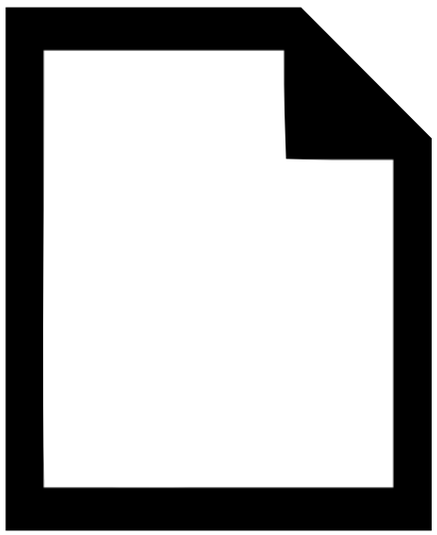
```
  at
com.sun.imageio.plugins.gif.GIFImageReader.readM
etadata(GIFImageReader.java:786)
```

JQF-AFL

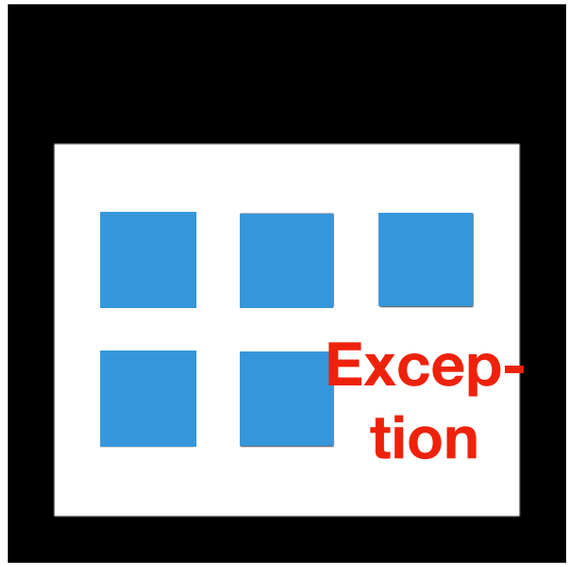
```
overall results
cycles done : 5
total paths : 871
uniq crashes : 124
  uniq hangs : 221
```



Instrumentation



Input



Java ImageIO

5#

PENTAGRID

Infinite loops DoS

CVE-2018-8017, CVE-2018-8036, CVE-2018-11771, CVE-2018-12418, CVE-2018-3214

Unit Tests!

5#

PENTAGRID

JQF-Zest

AFL-approach

+

JUnit Quickcheck Generators

JQF-Zest

Not only InputStream

Understands assume

Maven plugin

```
import org.apache.commons.collections4.trie.PatriciaTrie;
//[...]

@Fuzz
public void testCopy(Map<String, Integer> map, String key) {
    assumeTrue(map.containsKey(key));

    Trie trie = new PatriciaTrie(map);

    assertTrue(trie.containsKey(key));
}
```

https://issues.apache.org/jira/browse/COLLECTIONS-714

Commons Collections / COLLECTIONS-714

PatriciaTrie ignores trailing null characters in keys

Details

Type:	Bug	Status:	OPEN
Priority:	Critical	Resolution:	Unresolved
Affects Version/s:	4.3	Fix Version/s:	None

Reporter: Rohan Padhye

```
PatriciaTrie<Integer> trie = new PatriciaTrie<>();
trie.put("x", 0);
Assert.assertTrue(trie.containsKey("x")); // ok
trie.put("x\u0000", 1);
Assert.assertTrue(trie.containsKey("x")); // fail
```

```
Semantic Fuzzing with Zest
--------------------------

Test name:              examples.TestAsn#testWithByteArray
[redacted]

Elapsed time:           47s (no time limit)
[redacted]

Unique failures:        6
[redacted]

Execution speed:        2,220/sec now | 1,618/sec overall
Total coverage:         579 branches (0.88% of map)
[redacted]
```

Bouncycastle ASN.1

```
root#
```

5#

CVE-2019-17359 DoS

```
java.lang.OutOfMemoryError: Java heap space
  at org.bouncycastle.asn1.DefiniteLengthInputStream.toByteArray(Unknown Source)
at org.bouncycastle.asn1.ASN1StreamParser.readObject(Unknown Source)
at org.bouncycastle.asn1.ASN1StreamParser.readVector(Unknown Source)
at org.bouncycastle.asn1.DERExternalParser.getLoadedObject(Unknown Source)
at org.bouncycastle.asn1.ASN1StreamParser.readVector(Unknown Source)
at org.bouncycastle.asn1.ASN1StreamParser.readTaggedObject(Unknown Source)
at org.bouncycastle.asn1.ASN1InputStream.buildObject(Unknown Source)
at org.bouncycastle.asn1.ASN1InputStream.readObject(Unknown Source)
at Test.testWithByteArray(Test.java:13)
at Test.main(Test.java:27)
```

5#

PENTAGRID

more instrumentation

=

more bug classes

Instrument for bug classes?

Cryptographic bugs?

~~XSS, CSRF~~

Injections?

SSRF?

Logic bugs?

Future:

Java Security Policy Manager

5#

PENTAGRID

@pentagridsec
https://pentagrid.ch