Coverity, Crash Testing, Fuzzing The Numbers

Caolán McNamara,
Red Hat
2016-09-09

- Coverity Status
- Crash Testing Status
- Fuzzing Status

Process integration

- Run about twice a week
  - Those are the nums of slots coverity makes available to a project of this size
- Typically back to back
  - One to collect warnings
  - One after warnings fixed
- Results now mailed to the list
- Takes about 4-6 hours to build
- Takes about 12+ hours to analyze server-side

Defect Density 2015

Open Source Defect Density

LibreOffice: 7,102,667 line of code and 0.00 defect density

Open Source Defect Density By Project Size

Line of Code (LOC)	Defect Density
Less than 100,000	0.35
100,000 to 499,999	0.5
500,000 to 1 million	0.7
More than 1 million	0.65

Note: Defect density is measured by the number of defects per 1,000 lines of code, identified by the Coverity platform. The numbers shown above are from our 2013 Coverity Scan Report, which analyzed 250 million lines of open source code.

2014 density at conference time was 0.08

Defect Density 2016

Open Source Defect Density

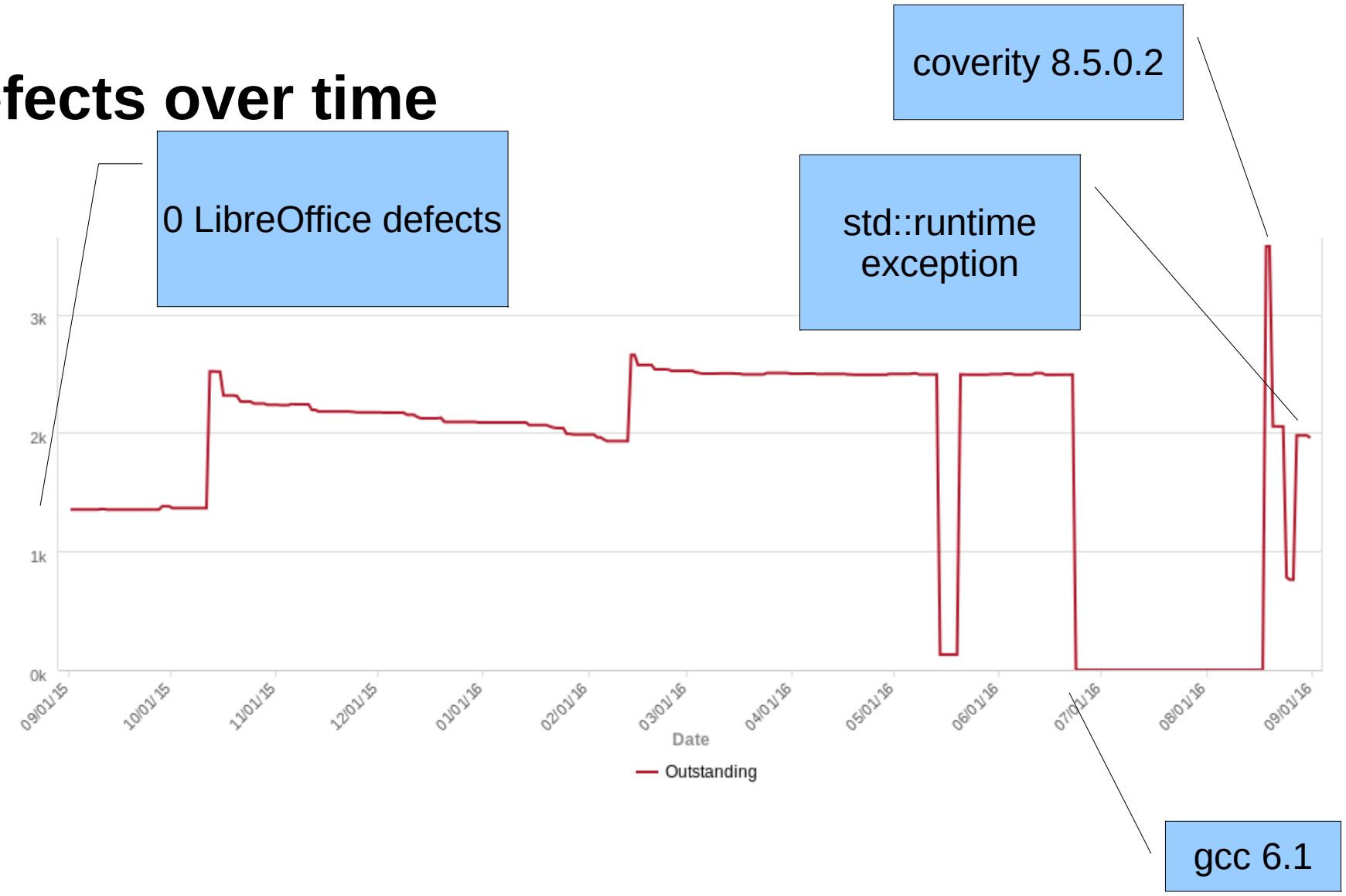
LibreOffice: 7,085,811 line of code and 0.02 defect density

Open Source Defect Density By Project Size

Line of Code (LOC)	Defect Density
Less than 100,000	0.35
100,000 to 499,999	0.5
500,000 to 1 million	0.7
More than 1 million	0.65

Note: Defect density is measured by the number of defects per 1,000 lines of code, identified by the Coverity platform. The numbers shown above are from our 2013 Coverity Scan Report, which analyzed 250 million lines of open source code.

Defects over time



Here, “ignored” third party module warnings are counted.

What's Changed

- We've 16,856 less lines of code apparently
- Now using latest version of coverity 8.5.0.2
- Works with gcc 6.1, previous release doesn't
- Has extra warnings for C++11

Extra C++11 related Warnings (1/2)

- Wrapper Object use after free knows about std::unique_ptr
- std::begin/std::end support on arrays seems broken (Illegal Address computation)
- Confusing “Misused comma operator” report for accessing static member variables through pointer/ref to an instance

Extra C++11 related Warnings (2/2)

- MISSING_MOVE_ASSIGNMENT
  - MMA is where the 0.02 comes from
  - Mixture of implementing move assignment, removing unnecessary temp objects, removing non-default methods which block the generation of the default move assignment
- Some new java warnings for changes in java apis from 1.6 to 1.7
  - Resource leak on an exceptional path where stuff grew a “close” api we don’t call


Crash Testing

What it does

- Loads a bunch of documents
  - 118 different columns for formats in output
    - Includes staroffice binary formats, which are ~supported again?
  - See if anything crashes or triggers an assert
- Saves a bunch of documents
  - Exports to 12 different formats from all the compatible import formats
  - Export to doc, docx, odb, odg, odp, ods, odt, ppt, pptx, rtf, xls, xlsx

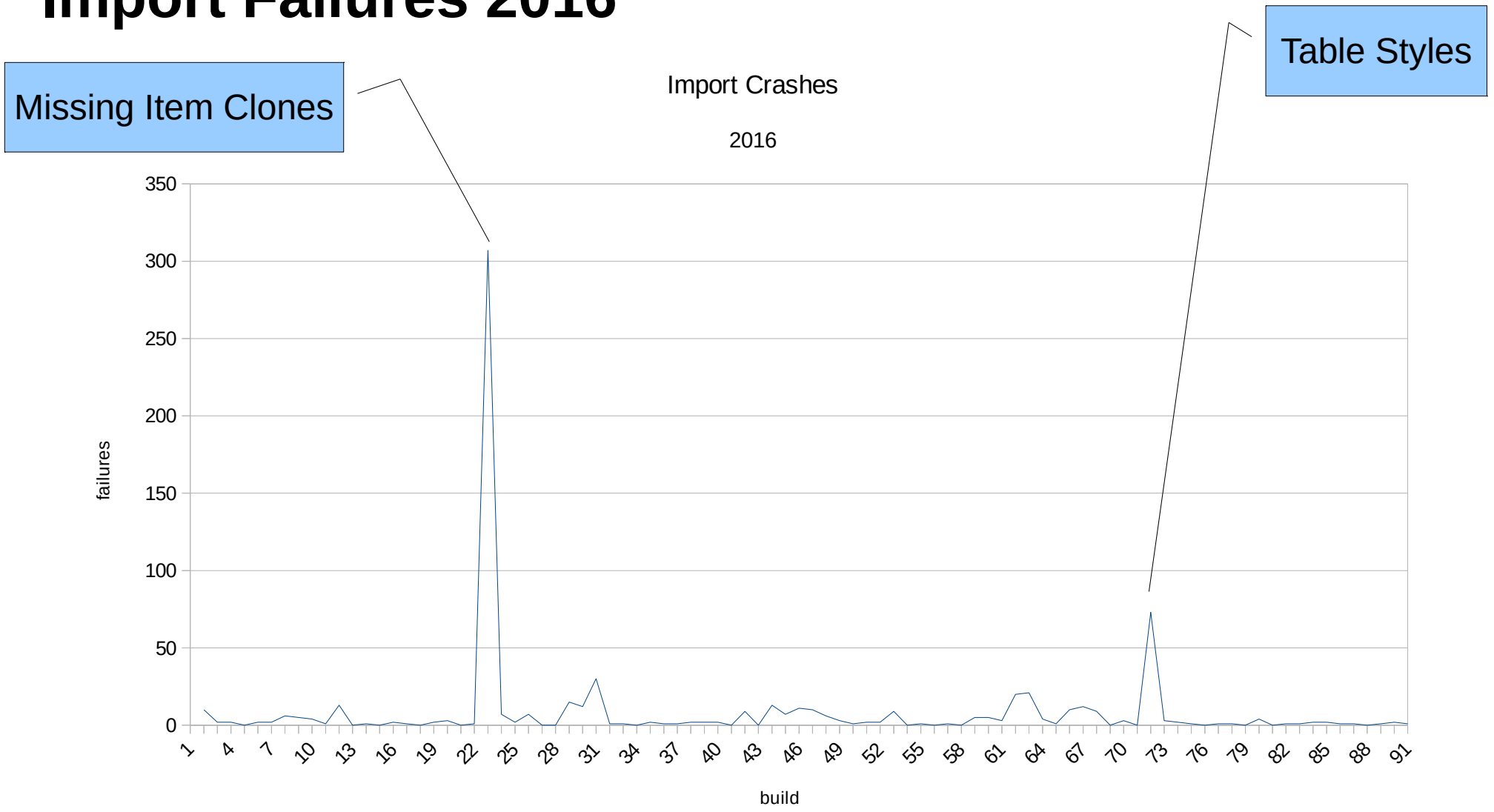
Process integration

- Typically run once or two a week
  - Takes about two days to complete
- Approx 93,000 documents in the document horde
  - Up 10,000 from last year
  - Mostly populated from get-bugzilla-by-mimetype
  - + w3c svg test documents
  - + various interesting documents that have caused trouble for some app or other in the past

Horde Updating

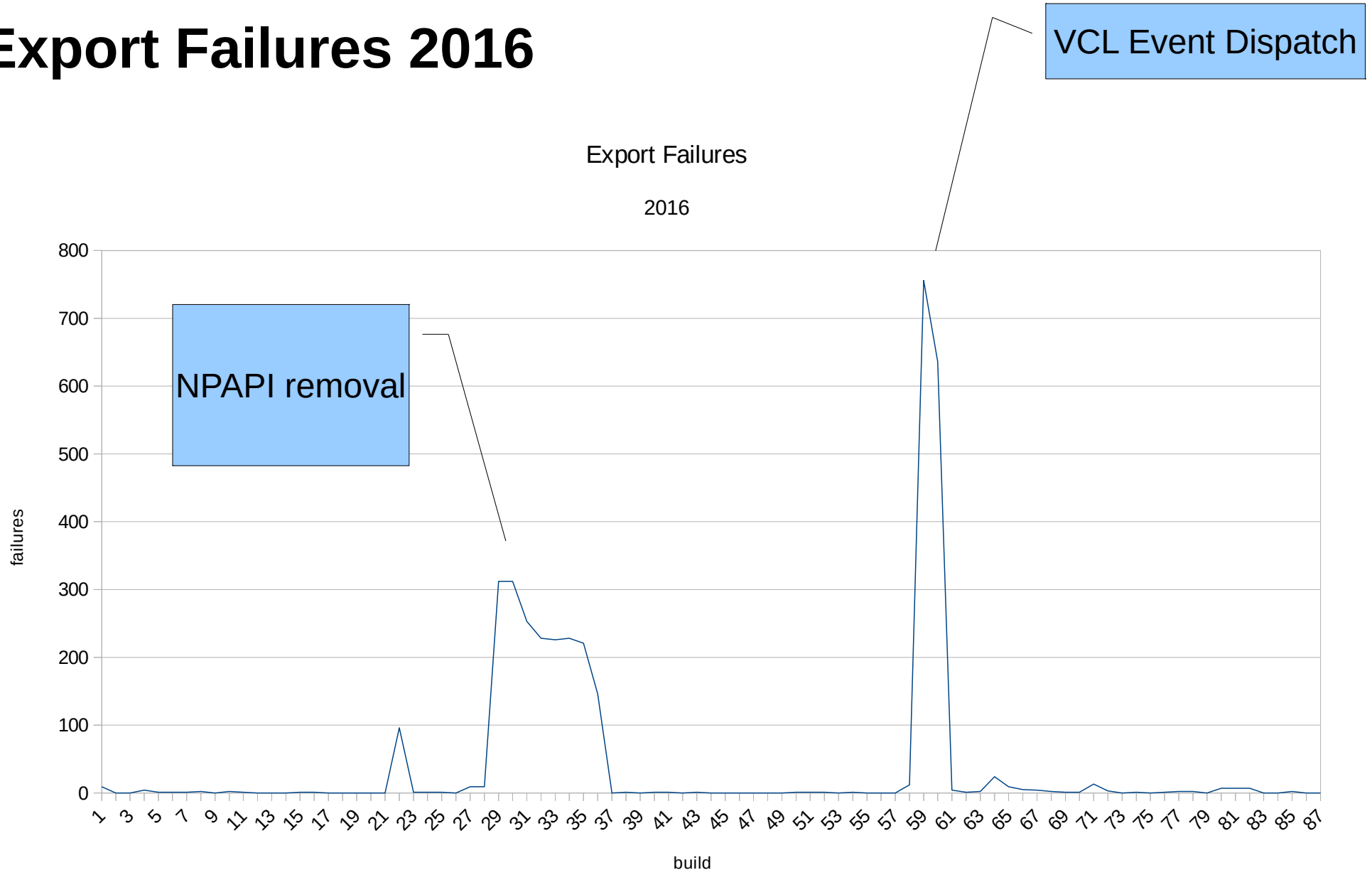
- Typically fairly rarely
- Full update takes about 12/13 hours
- Downloads are cached, so only new documents are updated
- Bugzilla is trusted wrt the mime-type
  - Lots of miscategorized stuff
  - Doesn't really matter, rtfs pretending to be docs, etc
  - Just made doc import filter look a little worse than it was

Import Failures 2016



Build 1 is 31 Aug 2015, final build was 1 Sep 2016

Export Failures 2016



Build 1 is 31 Aug 2015, final build was 1 Sep 2016

This week

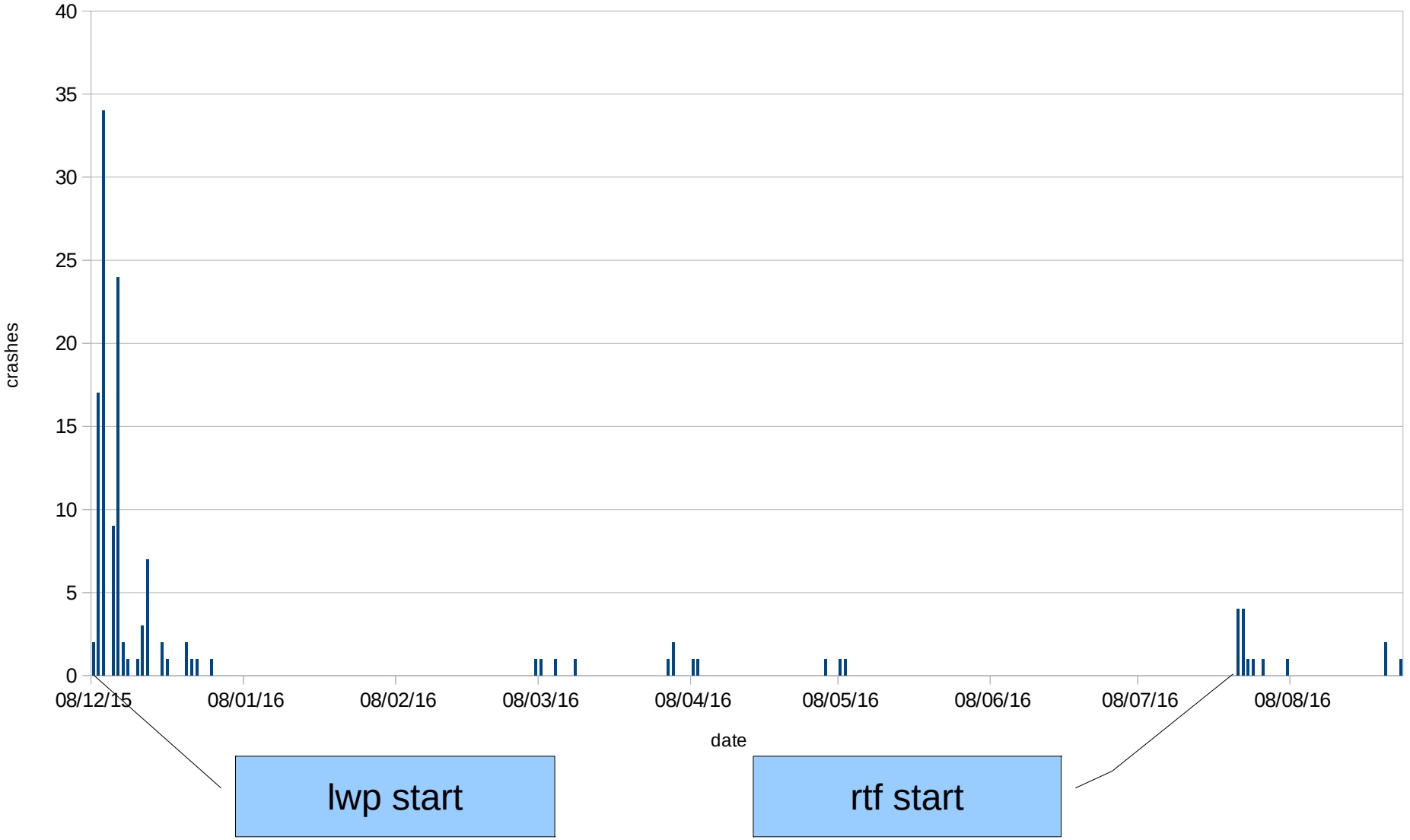
- ~40 coverity warnings
- 0 import failure
- 1 export failure

Caolán McNamara

American-fuzzy-lop integration

- fftester is the streamlined file format loader for format testing
- Cuts out some slow config-related paths
- Supports afl-server mode
- afl-cmin over the crashtesting horde to find best spread of unique inputs for given format
  - Then throw out the big ones
- Tend to get most of the good stuff early on
  - Then long pauses and flurries of activity

This years fuzzing Yields



Thanks for your time