Fuzzing Experiences

Erik Poll

Digital Security

Radboud University Nijmegen

Radboud Universiteit Nijmegen

TRU/e Master in Cyber Security

Overall impressions

Overall experiences are conform expectations:

- **afl outperforms dumber fuzzers Radamsa & zzuf**
- **ASan doubles cpu time, but is worth it**

Eg for id3v2 library for reading tags in mp3 files, group 7 found

- **known issue with afl, Radamsa, and zzuf**
- **new issue(s) with afl, and more with afl+ASan or valgrind**

But there are plenty of exceptions to this expected behaviour!

Outliers from overall impressions & expectations

Sometimes Radamsa & zzuf are just good as afl

- as group 16 experienced for LunaSVG
- as group 17 experienced for media player mpv

- This is typically the case for low-quality code?

- Still, afl keeps an advantage in that it tries to weed out duplicates to get ‘unique’ crashes
  - but beware: different crashes that afl considers ‘unique’ may still point to the same bug

Outliers from overall impressions & expectations

- Sometimes ASan does not improve things in any way
  - One group even found fewer bugs with ASan (or was timing different?)

- Group 10 got lucky and hit error with manual testing?

- Some groups reported afl++ was much faster than afl, another group that it made no difference

Initial test corpus

Important parameter in the fuzzing: the initial test corpus

- Size matters! You want test files to be small

- How many do you need?
  - if application takes several formats, at least one of each.
  - for “container” types like .mkv which can hold other (multimedia) objects you may need many?
- And which ones?
  - Group 6 only found results (quickly) with afl after finding the right set of 9 PDFs
  - Group 14 found many more flaws with (slightly) largerDummy.pdf than with smaller dummy.pdf in pdf-text
- afl-cmin can reduce test set by removing similar initial files

Initial test corpus

- **Two groups used Radamsa to generate mutations from one initial file, to then use these mutations as initial files for afl.**

Do you think this make sense?

I think not. Beter to let afl use its clever mutation strategy for the one initial file, instead of relying of Radamsa's dumber strategy

Additional parameters to tweak?

- For zzuf, the fuzzing ratio -r

Adding bugs? Looking for known bugs?

Adding bugs or looking for known issues are the only way to see if tools have false negatives.

- **It is generally easier to find false positives than false negatives. Not just for fuzzing.**

- **group 15 added a bug in Clementine music player,**
  which afl (even without ASan?) found,
  and run with ASan provides good info pointing to that bug

Crashes vs flaws vs errors vs warnings…

- For afl, crash = any termination by program error signal
  - eg seg-fault, divide-by-zero, …
- For Radamsa, it's up to you to define it…

Not always clear which errors are really security concerns

- If an application aborts with its own errors or warnings, should these be counted as (potential) security flaws?

1. `Corrupt JPEG data: premature end of data segment`
2. `Warning CRC error`
3. `Truncated or incomplete image`
4. `ERROR: assertion 'contributor->n1 >= contributor->n0' failed`

Consistency checks

- If a file format has built-in consistency check,
  - like PNG using CRC checks

  then ideally you want to remove these checks from the application!
  - Because fuzzing will generate lots of invalid examples that will be caught by such checks
  - Also, if detection results in errors, you get lots of false positives!

  Several groups mention running into CRC checks

Mature open source & therefore robust?

Quote from one of the reports

"It should be noted that MuPDF has been around for 15 years now, which means it is expected to be a robust open source project which has been thoroughly tested not only by the developers but also by the community."

Group 6 found crashes with afl(++) in Mutool v1.12 (2017)
but none in latest version v1.13

Some interesting results

- Group 3: new CVE found in GdkPixbuf thumbnail generator
- Group 6: plenty of bugs in 2017 version of MuPDF
- Group 7: known & new bugs in id3v2 library to read mp3 tags
- Group 15: known CVE from 2018 found with Radamsa (though earlier Radamsa run did not reveal it?) in Clementine music player

Tool practicalities

- Experiences in getting tools to run vary wildly between groups
  - depending on OS, but also on complexity of building the application under test
- winafl not a workable option for Windows users

  Getting some central Linux servers for running experiments?
  Also, to avoid ASan's problems in VMs
- afl vs afl++
- Ditch CERT BFF?

- Spending more time & paying more attention to selection of interesting use case - too less mature test cases and/or smaller ones