Student: Sebastian Eydam (eydamseb@b-tu.de) **Date**: 07.01.2019
Tool: American Fuzzy Lop - AFL
Source: AFL is written and maintained by Michal Zalewski (lcamtuf@google.com).
Target languages: AFL supports programs written in C, C++ or Objective C compiled with either gcc or clang. There are variants and derivatives of AFL that allow you to fuzz Python, Rust or other languages (http://lcamtuf.coredump.cx/afl/ at *Download & other useful links*).
Platform: The tool is confirmed to work on x86 Linux, OpenBSD, FreeBSD and NetBSD, both 32- and 64-bit. It should also work on MacOS X and Solaris.

The functionality in a nutshell: Fuzzing is an automated testing techique, where the fuzzer creates structured, relatively random inputs for your program to create unexpected behavior and expose corner cases. AFL has customized compilers (e.g. afl-clang++) that instruments your code, so that AFL can observe the execution path the program takes for a given input. AFL takes some input seeds the user has to provide, feeds it to the program and mutates the input to trigger new execution paths in the programm. If the tool finds an input that crashes the program, it will report the crash and the input.
Experiences:

- Installation: AFL is easy to install
  - Make sure you have installed gcc, clang and make
  - after that, its just:
  - `mkdir AFL && cd AFL`
  - `wget http://lcamtuf.coredump.cx/afl/releases/afl-latest.tgz`
  - `tar xzf afl-latest.tgz`
  - `make`
  - `sudo make install`
- User Interface: as there is not much to configure, there is no ui. After starting afl with `afl-fuzz`, afl will show you a command-line-interface to show you the current results.
- AFL is kinda hard to learn, because you need to have a program that can be reasonably fuzzed. Michal Zalewski says a few words about this in the last three paragraphs here: https://lcamtuf.blogspot.com/2014/11/pulling-jpegs-out-of-thin-air.html.

AFL is a powerful tool to find bugs in programs, but it can take a long time to fuzz a program and you have to be able to evaluate if your program can be reasonably fuzzed.

References:

American Fuzzy Lop main page
http://lcamtuf.coredump.cx/afl/

Pulling JPEGs out of thin air
https://lcamtuf.blogspot.com/2014/11/pulling-jpegs-out-of-thin-air.html

Binary fuzzing strategies: what works, what doesn't
https://lcamtuf.blogspot.com/2014/08/binary-fuzzing-strategies-what-works.html

afl-fuzz: crash exploration mode
https://lcamtuf.blogspot.com/2014/11/afl-fuzz-crash-exploration-mode.html

Fuzzing Android
https://www.blackhat.com/docs/us-15/materials/us-15-Drake-Stagefright-Scary-Code-In-The-Heart-Of-And
pdf#page=44

Wikipedias article about fuzzing
https://en.wikipedia.org/wiki/American_fuzzy_lop_(fuzzer)

Some slides about fuzzing and the AFL
https://bart.disi.unige.it/zxgio/phd-course-2017/fuzzing_slides.pdf

A blogpost about the AFL
https://research.aurainfosec.io/hunting-for-bugs-101/

Another blogpost about AFL
http://moyix.blogspot.com/2016/07/fuzzing-with-afl-is-an-art.html

A mail in a mailing list about fuzzing the clang test suite
http://article.gmane.org/gmane.comp.compilers.llvm.devel/79491