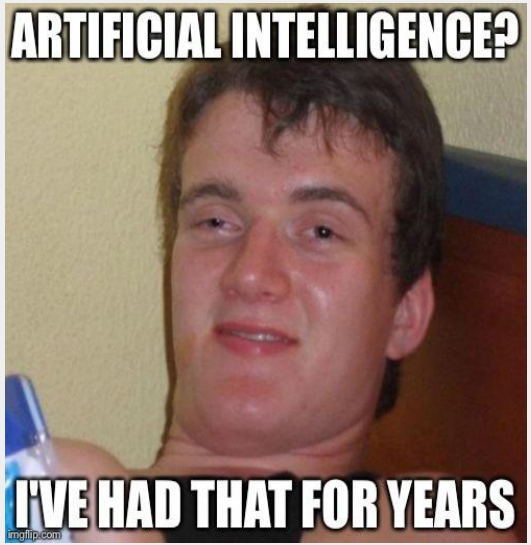
Intro To Fuzzing with AFL

Or ‘How To Hunt Bugs While You Sleep’

cat /etc/*elease; uname -a

- Undergrad for 11 years over 5 different majors
- Not a ‘Cyber Security Professional’
- I work in Data Science
- Recently converted to CS Pro path



ls -lah presentation/

- Fuzzing intro
  - What it is
  - Why it is
  - Types of fuzzers
- Intro to AFL
- Demo
- Other Features

caveats

- I like interactive presentations
- I love answering questions
- This will be very high level
  - Happy to answer more technical questions after (or during), if I can
  - This talk was designed for students and people literally getting started with fuzzing
- I've only done this in Linux (specifically, Debian based)
  - More specifically... AFL seems to have issues with Ubuntu 18.04, so I'm using 16.04
  - Not even sure if all of this works on Windows
  - It definitely doesn't work on Mac

what is fuzzing?

- Technically, brute forcing
  - *Fuzzing: Brute Force Vulnerability Discovery*, by Sutton, Greene, and Amini [1]

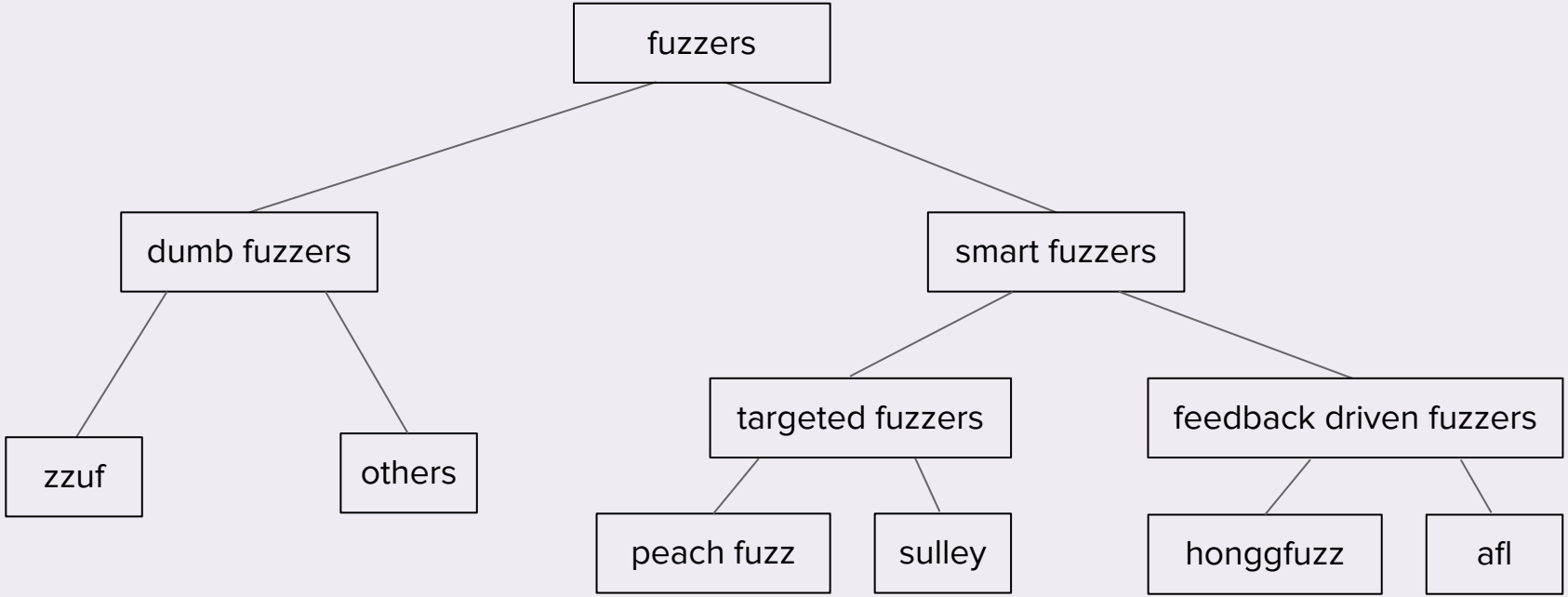
- Sending “random” input to an application
  - We all know what “random” means when it comes to computers
  - Usually trying to get a program to crash
    - Because crashes mean possible memory corruption vulnerabilities
      - Which means possible exploits...
      - Vulnerability? Exploit?
  - Algorithms determine input

- More time means more test cases
  - Patience is wise; though, not always

so, why aren't we always fuzzing?

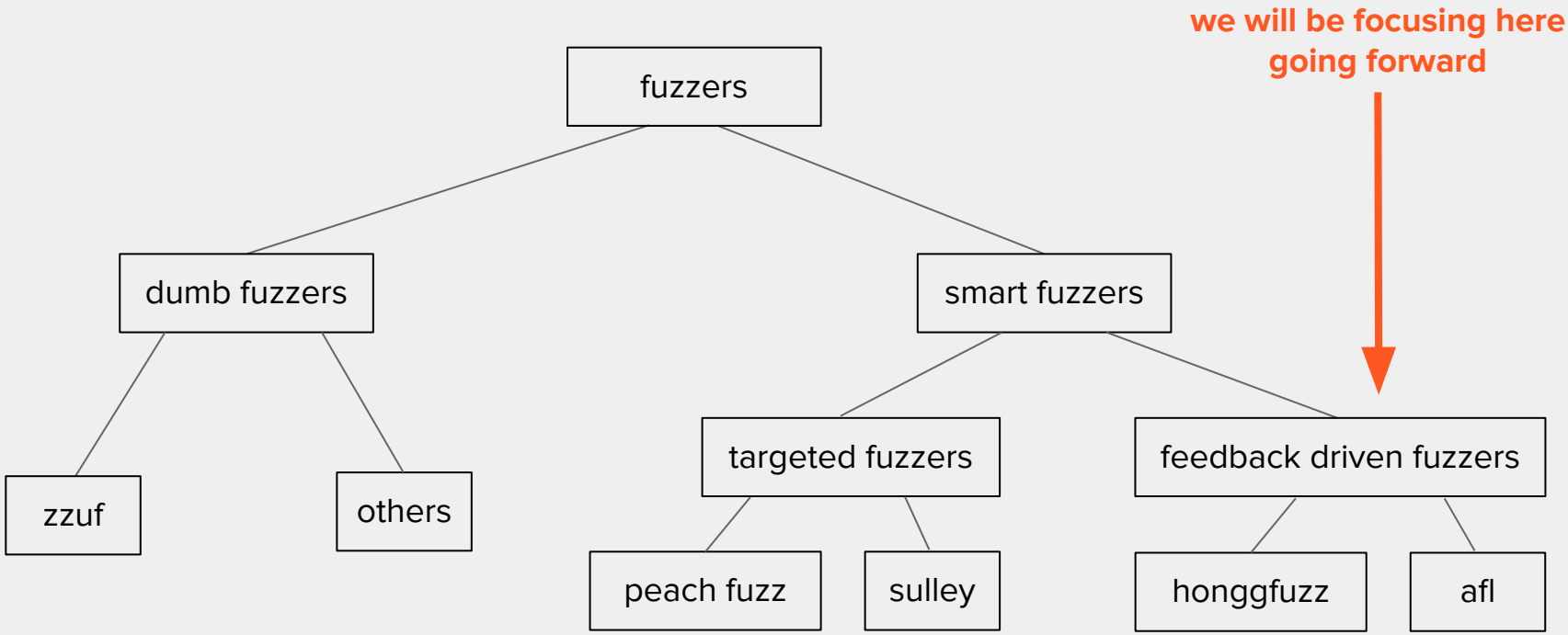
- "We" are!
  - AND YOU CAN TOO!
  - Check this out (CVEs for vulns found by fuzzing -- not comprehensive) [2]
- Longest reported fuzz job was 15 months (Windows Vista OS tools) [3]
- Fuzzing is a common step before tool release*
- Also, common first step for researchers*

types of fuzzers



(dfs - focs)

types of fuzzers



(dfs - focs)

But...

How to Get Started?

step 1

- Find Hardware
  - Rent a Server?
  - Enjoy paying power bills?
  - Don’t use a VPS service

- It depends on who you ask
  - I use a machine from home
  - Power Bill metrics coming soon!

steps 2 & 3

- Find software to fuzz
  - Depending on who you ask...
    - Everything has been fuzzed
      - Except Google Chrome...
      - Except others...
    - Not enough has been fuzzed
  - So...
    - Google, then fuzz

- Find a fuzzer(s)
  - As mentioned previously, there are different fuzzers
  - Genetic fuzzing is cool, but there are reasons to use certain fuzzers
  - I prefer one over others....

hello, AFL

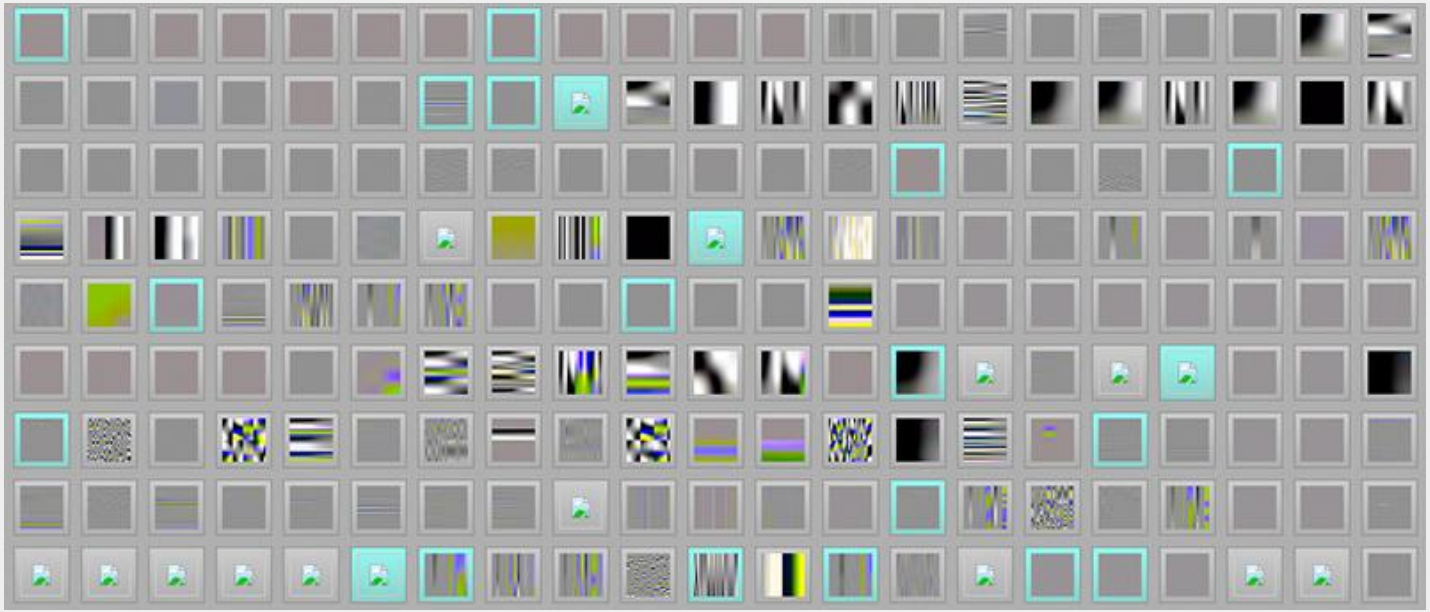
- American Fuzzy Lop, primarily written by Michal Zalewski [4]
  - Ideas/Maybe some code submitted by Tavis Ormandy and others
- Download, compile, and start fuzzing x86 CLI apps in less than an hour
  - Less than 10 minutes if nothing goes wrong/nothing needs upgrading
    - a.k.a. You use the recommended platform and default settings
- Impressive trophy case
  - AFL Bug-O-Rama Trophy Case [4]
- Impressive tool suite (which we will talk more about in a bit)

what is happening?

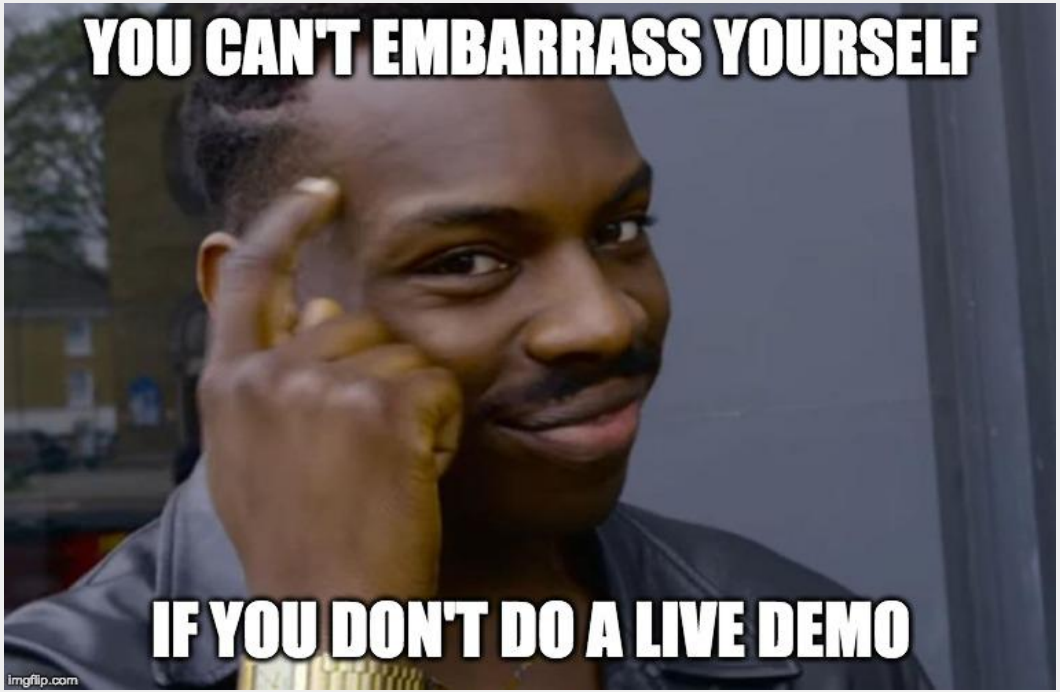
- Code is instrumented and compiled using AFL's custom compilers
  - "Flags" are inserted around basic blocks to keep track of paths followed in the program
- Seed files are passed as input to the program
  - Trace map is created of paths followed through program
- Seed files that find new paths are mutated and re-added to queue
  - New paths recorded as code block tuples
  - AFL moves from deterministic to random
- AFL trims files until they affect the checksum of the trace map
  - Small input size is important for fast fuzzing
  - AFL has a tool that does this more efficiently (instead of while fuzzing)

small example; djpeg [5]

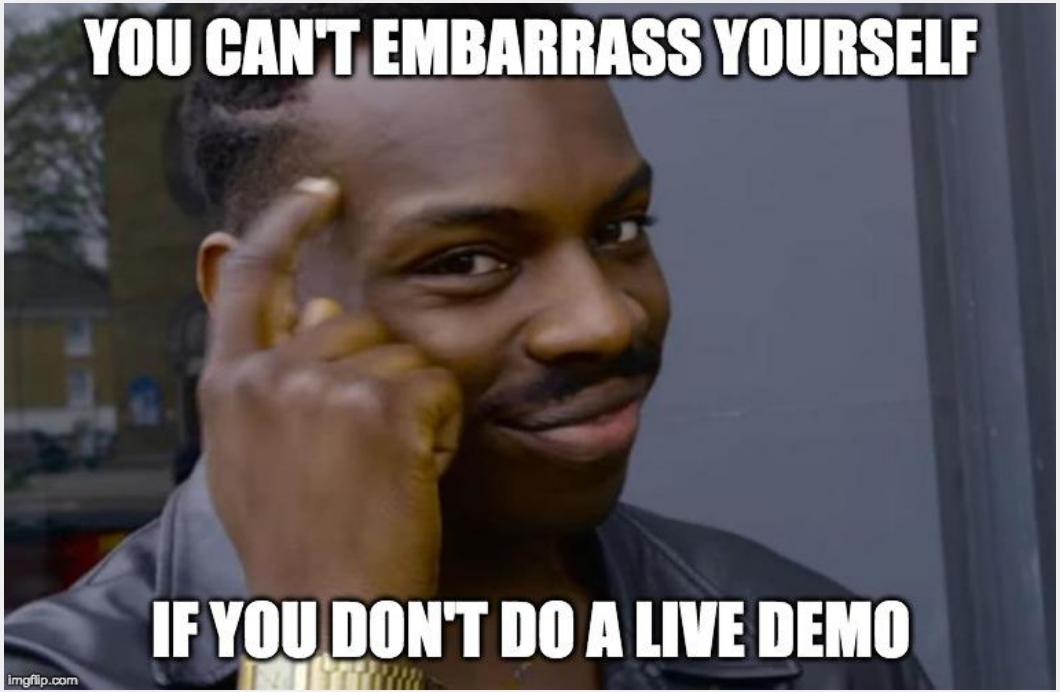
- Input file was a simple text file that read ‘hello’
- AFL generate this as a test case after ~7 hours of fuzzing:



fundamental rule of presenting...



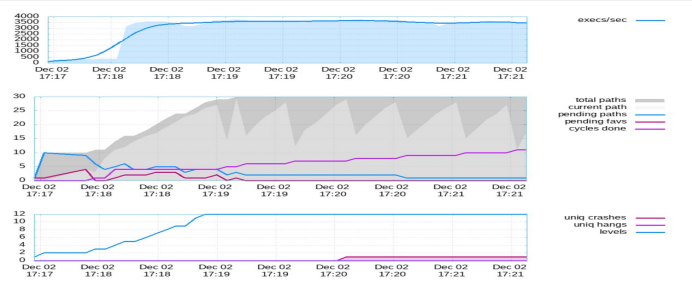
fundamental rule of presenting...



but, we will anyway...

DEMO

what else?

- Network fuzzing (not built-in, but possible)
- Non-x86 && Blackbox fuzzing (built-in, but hard/tedious)
  - As opposed to Whitebox fuzzing
- Parallel Fuzzing
  - Master Process is Deterministic, Slave Processes are “Random”
- Other Free and Open Source addons and plugins
- Built-In Graphing Utility
  - 

network fuzzing

- Requires some... uh... work
  - Some articles on getting it going [6], [7]
  - CVE-2015-5477 (BIND remote DoS) found with AFL
- This is due to how AFL passes input to binaries
- Not much more on this now...

non-x86 && blackbox

- Both supported the same way
  - Uses QEMU
  - So, it can work with any architecture supported by QEMU...
- 2-5x Slower than fuzzing with source code
- This brings me to some of the work I've done...
  - Fuzzing MIPS/ARM/etc. compiled binaries
  - Why fuzz an outdated version of a project (as we just did)?

why are you doing this?

- There is an end goal...
- Setup fuzzing environment for IoT
- Fuzzing is easy and useful
  - If you aren't, you should
  - If you haven't, you should
  - If you are, good
  - Okay... it's not *all* good

oh, also...

- This is only part 1...
  - Finding vulnerabilities is great, but...
    - Finding exploits is better!
  - If you find this interesting, get ready to learn gdb

- I have scripts to help with a lot of this
  - Probably around here: vr0n.tech/afl (lol, I lost this domain)
  - But, I have the scripts!

bonus slide

- Projects I’m currently following
  - FIRM-AFL
  - Drop Chat
  - Movuscator

- Fuzzing Resources
  - The Fuzzing Project
  - Google OSS-Fuzz
  - Curated list of Fuzzing Resources

references

1. Fuzzing - Brute Force Vulnerability Discovery
2. AFL CVE List
3. High Performance Fuzzing presentation by Cisco Talos VULNDEV Team
4. AFL
5. djpeg fuzzing article
6. Article 1 on getting started with Network Fuzzing
7. Article 2 on getting started with Network Fuzzing