

# Fuzzing

---

20190225

# Outline

Introduction

Principles of fuzzing

The new generation

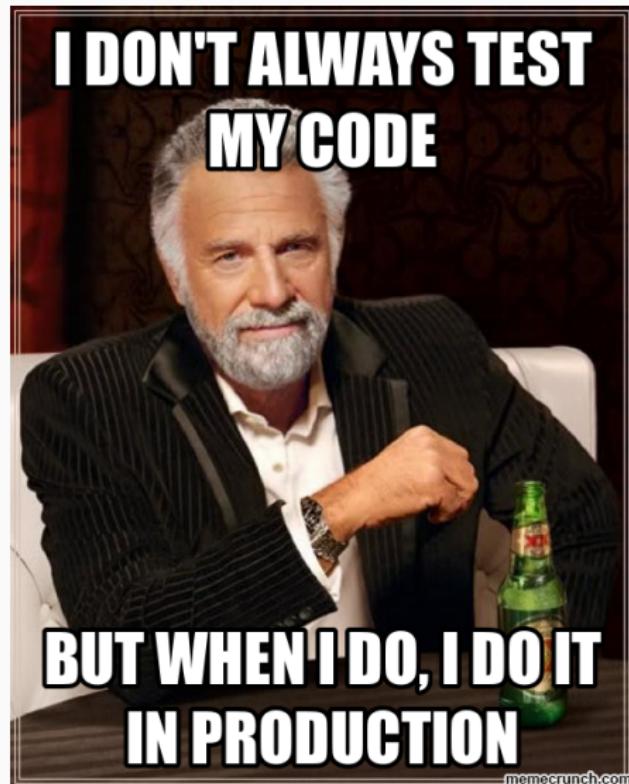
# Introduction

---

# Fuzzing : your code is buggier than mine



It's about testing



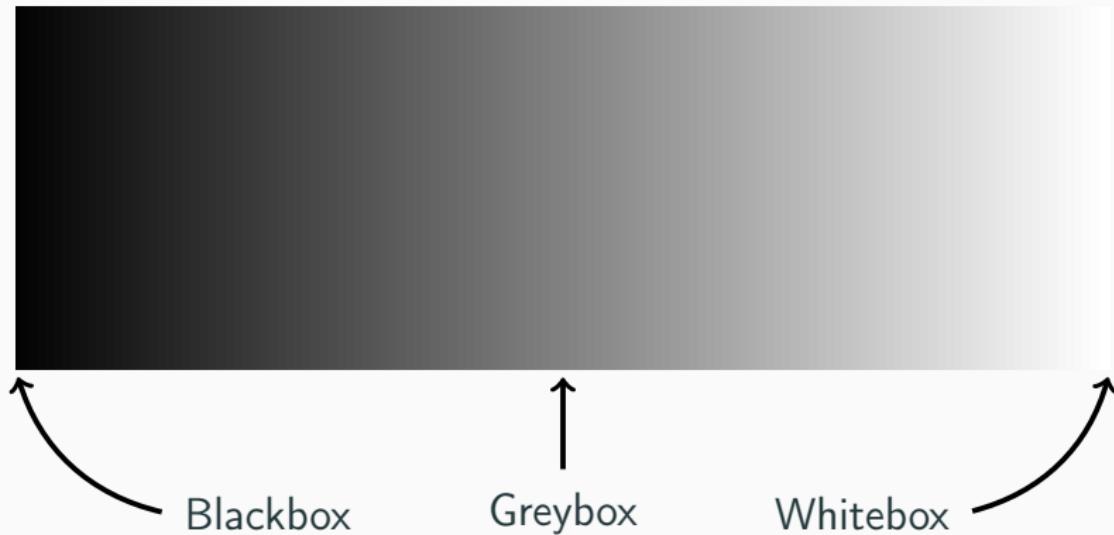
# What is fuzzing ?

At its core, fuzzing **is** random testing.

# A short (selective) pre-history

- 1981** Random testing is a cost-effective alternative to systematic testing techniques (Duran & Natos)
- 1983** "The Monkey" (Capps)
- 1988** Birth of the term "fuzzing" (Miller)

# Shades of fuzzing



# Blackbox

## Key property

A blackbox fuzzer is unaware of the program structure

Blackbox fuzzers can be considered **dumb**.

# Whitebox

## Key distinction

A whitebox fuzzer leverages program analysis to reach its targets:

Usual targets are:

1. code coverage;
2. program location.

This is usually synonym with "Dynamic Symbolic Execution".

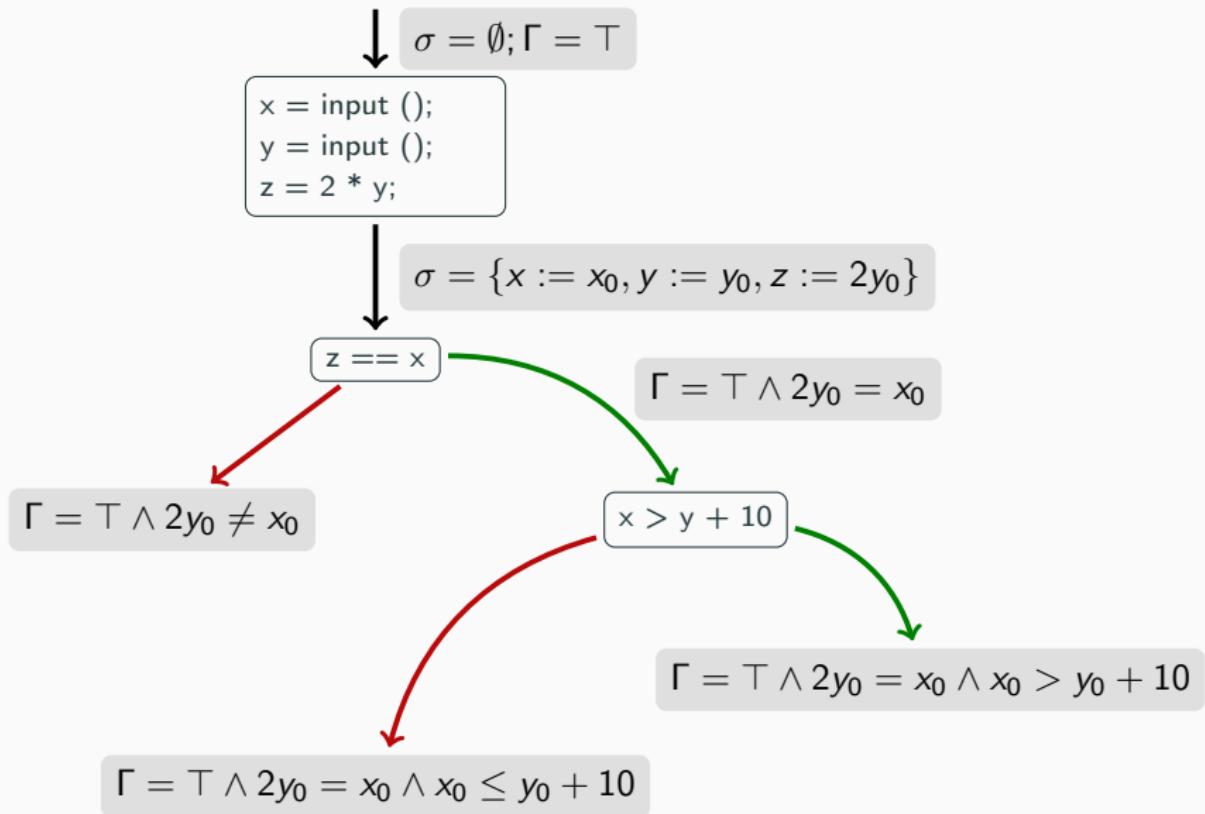
Whitebox fuzzers are **smart**.

# Coverage criteria

```
1 void
2 f(int a, int b, int * x)
3 {
4     if (a > 1 && b == 0)
5         *x = *x / a;
6     if (a == 2 || *x > 1)
7         *x = *x + 1;
8 }
```

- Instructions (I)
- All decisions (D)
- All simple conditions (C)
- All conditions / decisions (DC)
- All combinations of conditions (MC)
- All paths (P)

# Wait ? Whitebox fuzzers ?



# Greybox

Greybox fuzzers favor leveraging **instrumentations** instead of program analysis.

# Principles of fuzzing

---

# Stages

1. Preprocess
2. Scheduling
3. Input Generation
4. Input Evaluation
5. Configuration Updating
6. Continue

# Algorithm (Manes et al. 2019 )

---

Input:  $\mathbb{C}$ ,  $t_{\text{limit}}$

Output:  $\mathbb{B}$  // a finite set of bugs

- 1  $\mathbb{B} \leftarrow \emptyset$
  - 2  $\mathbb{C} \leftarrow \text{Preprocess}(\mathbb{C})$
  - 3 while  $t_{\text{elapsed}} < t_{\text{limit}} \wedge \text{Continue}(\mathbb{C})$  do
  - 4      $\text{conf} \leftarrow \text{Schedule}(\mathbb{C}, t_{\text{elapsed}}, t_{\text{limit}})$
  - 5      $\text{tcs} \leftarrow \text{InputGen}(\text{conf})$   
      //  $O_{\text{bug}}$  is embedded in a fuzzer
  - 6      $\mathbb{B}', \text{execinfos} \leftarrow \text{InputEval}(\text{conf}, \text{tcs}, O_{\text{bug}})$
  - 7      $\mathbb{C} \leftarrow \text{ConfUpdate}(\mathbb{C}, \text{conf}, \text{execinfos})$
  - 8      $\mathbb{B} \leftarrow \mathbb{B} \cup \mathbb{B}'$
  - 9 return  $\mathbb{B}$
-

# Stages in Fuzzing

## Preprocess

- User supplies set of fuzz configurations as input and get a potentially-modified set of fuzz configurations.
- May perform a variety of actions (inserting instrumentation code to PUTs, measuring the execution speed of seed files).

## Schedule

Takes in the current set of fuzz configurations, current time, and a timeout as input, and selects a fuzz configuration to be used for the current fuzz iteration.

# Stages ii

## Input Generation

- Takes a fuzz configuration as input and returns a set of concrete test cases.
- Some fuzzers use a seed in for generating test cases, while others use a model or grammar as a parameter.

# Stages iii

## Input Evaluation

- Takes a fuzz configuration, a set of test cases, and a bug oracle as input.
- Executes PUT on test cases and checks if executions violate the security policy using the bug oracle.
- Outputs set of bugs found  $\mathbb{B}'$  and information about each of the fuzz runs.

## Stages iv

### Configuration Update

- Takes a set of fuzz configurations, the current configuration, and the information of each of the fuzz runs. as input.
- May update the set of fuzz configurations. For example, many grey-box fuzzers reduce the number of fuzz configurations based on fuzz runs.

# Stages v

## Continue

- Takes a set of fuzz configurations as input and outputs a boolean indicating whether a next fuzz iteration should happen or not.
- This models white-box fuzzers that can terminate when there are no more paths to discover.

# Preprocess: Instrumentation

## Goal

Gather execution feedback during runs

## Types of instrumentation

**Static** usually at compile time, sometimes rewriting the binary (e.g., afl-gcc)

**Dynamic** more costly but can instrument dlls (e.g., afl-qemu)

# Preprocess: Seed Selection

## Goal

Find minimal set that maximizes a coverage metric

## Examples of metrics

**AFL** branch coverage with logarithmic counters on each branch

**Honggfuzz** # executed instructions, branches, unique basic blocks

# Scheduling Problem

## Goal

Pick the configuration that is the most likely to lead to the most favorable outcome.

- finding most unique bugs
- maximizing coverage

## Exploration vs Exploitation

Scheduling balances:

- time gathering more information on configurations (**exploration**) and
- time fuzzing configurations believed to lead to favorable outcomes (**exploitation**).

# Scheduling: Blackbox

Blackbox fuzzers can only use fuzz outcomes (time spent, bugs found).

## Examples

- Favoring the configuration with a high success rate ( $\# \text{bugs} / \# \text{runs}$ )
- Prefer faster configurations (collection of information on them is quicker) & fix the time per configurations selection instead the number of runs (avoid spending a lot of time in slow configuration).

## Scheduling: Greybox i

### AFL

- Maintain a population of configurations with a fitness metric, apply some degree of genetic transformation (mutation, recombination)
- On a control-flow edge, AFL considers fastest and smallest inputs as "fitter"
- Fix number of run per selection.

## Scheduling: Greybox ii

### AFLFast

- On a control-flow edge, favor the one that has been chosen least
- On tie, favor the one that exercises path that has been selected the least
- The fuzzing time follows a power schedule

# Input generation: Generation-based

## Predefined model

- Network protocols
- EBNF
- System calls (types and number of arguments)

## Inferred

- Synthesize the grammar of the parser
- Capture packets and infer network protocols
- Observe I/O behavior and infer state machine
- Machine learning

## Input generation: Mutation-based

- Use initial seeds providing a structure of a valid input (file, network packets, . . . )
- Mutate portions of previous inputs to generate new *mostly valid* test cases.

# Bit-flip

## Basics

Flip:

- a fixed-number of bits or
- a random number of bits

## Mutation ratio

Determines the number of bit flips for a single generated input.

A good mutation ratio can be inferred through program analysis.

# Arithmetic mutation

## Basics

Perform an arithmetic operation on a sequence of bytes seen as an integer.

## Example

AFL selects 4-bytes values  $i$ , generates a random integer  $r$  and applies  $i + r$  or  $i - r$ .

The range of  $r$  is tunable.

# Block-based mutation

## Block

Arbitrary sequence of bytes

## Mutations

1. Insert a new block at a random position
2. Delete a randomly selected block
3. Replace a randomly selected block
4. Permutes the order of block sequences
5. Resize a seed by appending a block
6. Take a random block from another seed to insert/replace

# Dictionary-based mutation

Some fuzzers use "magic" values like 0 or  $-1$ , or format strings for mutation.

## Examples

**AFL** Uses 0, 1,  $-1$  for integer mutations

**Radamsa** unicode strings

**GPF** `%x`, `%s` for string mutations

# What about whitebox fuzzing?

DSE Figure HERE

# Input Evaluation: Bugs

## Default policy

A program execution terminated by a fatal signal is a violation

This is enough for memory vulnerabilities since they usually trigger segmentation faults.

## Problem

The default policy will not detect corruptions leading to valid addresses.

To this effect, code **sanitizers** are needed, i.e., a form of runtime monitor that will trigger a fatal signal on property violation.

# Input Evaluation: Triage

## Deduplication

Prune test cases triggering the same bugs as another one.

Implemented by stack backtrace hashing or coverage-based deduplication (AFL).

## Prioritization

Rank or group violating test cases according to severity and uniqueness, aka determines a form of **exploitability**.

`!exploitable`: 4-rank scales (yes, probably, unknown, not likely).

# Input Evaluation: Triage ii

## Test case minimization

Identify the part of the test case that is necessary to trigger the violation.

Produce a smaller test case (e.g., AFL sets bytes to 0 and shorten the test case).

This is not specific to fuzzers and thus dedicated techniques have been developed and can be reused here.

# Configuration Updating

## Evolving the seed pool

Select the fittest.

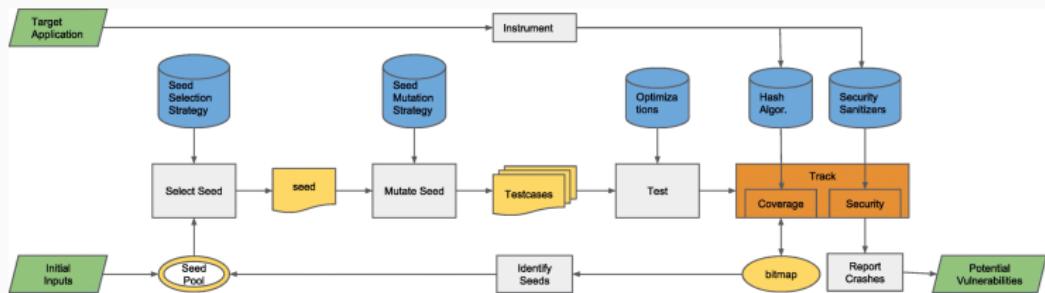
## Example

- Use on node/branch coverage: a newly discovered branch/node is sign of fitness !
- AFL also takes into account the number of times a branch has been taken
- Angora also adds calling context
- Steelix checks input offsets affect the progress in comparison instructions

## A concrete example : AFL



# The AFL loop



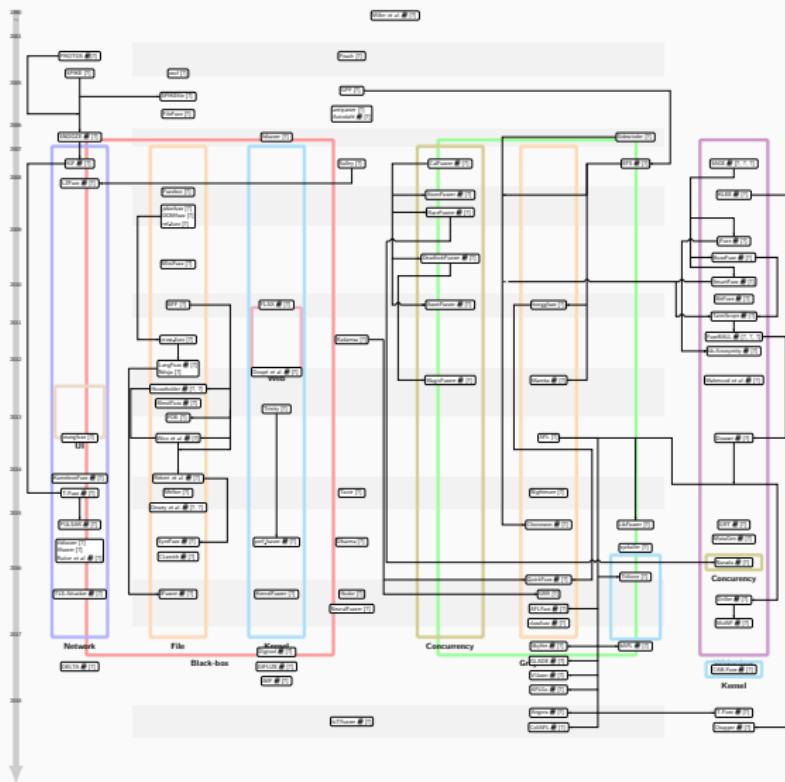
# Efficiency

clang / lld [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#) [15](#) [16](#) [17](#) [18](#) [19](#) [20](#) [21](#) [22](#) [23](#) [24](#) [25](#) [26](#) [27](#) [28](#) [29](#) [30](#) [31](#) [32](#) [33](#) [34](#) [35](#) [36](#) [37](#) [38](#) [39](#) [40](#) [41](#) [42](#) [43](#) [44](#) [45](#) [46](#) [47](#) [48](#) [49](#) [50](#) [51](#) [52](#) [53](#) [54](#) [55](#) [56](#) [57](#) [58](#) [59](#) [60](#) [61](#) [62](#) [63](#) [64](#) [65](#) [66](#) [67](#) [68](#) [69](#) [70](#) [71](#) [72](#) [73](#) [74](#) [75](#) [76](#) [77](#) [78](#) [79](#) [80](#) [81](#) [82](#) [83](#) [84](#) [85](#) [86](#) [87](#) [88](#) [89](#) [90](#) [91](#) [92](#) [93](#) [94](#) [95](#) [96](#) [97](#) [98](#) [99](#) [100](#) [101](#) [102](#) [103](#) [104](#) [105](#) [106](#) [107](#) [108](#) [109](#) [110](#) [111](#) [112](#) [113](#) [114](#) [115](#) [116](#) [117](#) [118](#) [119](#) [120](#) [121](#) [122](#) [123](#) [124](#) [125](#) [126](#) [127](#) [128](#) [129](#) [130](#) [131](#) [132](#) [133](#) [134](#) [135](#) [136](#) [137](#) [138](#) [139](#) [140](#) [141](#) [142](#) [143](#) [144](#) [145](#) [146](#) [147](#) [148](#) [149](#) [150](#) [151](#) [152](#) [153](#) [154](#) [155](#) [156](#) [157](#) [158](#) [159](#) [160](#) [161](#) [162](#) [163](#) [164](#) [165](#) [166](#) [167](#) [168](#) [169](#) [170](#) [171](#) [172](#) [173](#) [174](#) [175](#) [176](#) [177](#) [178](#) [179](#) [180](#) [181](#) [182](#) [183](#) [184](#) [185](#) [186](#) [187](#) [188](#) [189](#) [190](#) [191](#) [192](#) [193](#) [194](#) [195](#) [196](#) [197](#) [198](#) [199](#) [200](#) [201](#) [202](#) [203](#) [204](#) [205](#) [206](#) [207](#) [208](#) [209](#) [210](#) [211](#) [212](#) [213](#) [214](#) [215](#) [216](#) [217](#) [218](#) [219](#) [220](#) [221](#) [222](#) [223](#) [224](#) [225](#) [226](#) [227](#) [228](#) [229](#) [230](#) [231](#) [232](#) [233](#) [234](#) [235](#) [236](#) [237](#) [238](#) [239](#) [240](#) [241](#) [242](#) [243](#) [244](#) [245](#) [246](#) [247](#) [248](#) [249](#) [250](#) [251](#) [252](#) [253](#) [254](#) [255](#) [256](#) [257](#) [258](#) [259](#) [260](#) [261](#) [262](#) [263](#) [264](#) [265](#) [266](#) [267](#) [268](#) [269](#) [270](#) [271](#) [272](#) [273](#) [274](#) [275](#) [276](#) [277](#) [278](#) [279](#) [280](#) [281](#) [282](#) [283](#) [284](#) [285](#) [286](#) [287](#) [288](#) [289](#) [290](#) [291](#) [292](#) [293](#) [294](#) [295](#) [296](#) [297](#) [298](#) [299](#) [300](#) [301](#) [302](#) [303](#) [304](#) [305](#) [306](#) [307](#) [308](#) [309](#) [310](#) [311](#) [312](#) [313](#) [314](#) [315](#) [316](#) [317](#) [318](#) [319](#) [320](#) [321](#) [322](#) [323](#) [324](#) [325](#) [326](#) [327](#) [328](#) [329](#) [330](#) [331](#) [332](#) [333](#) [334](#) [335](#) [336](#) [337](#) [338](#) [339](#) [340](#) [341](#) [342](#) [343](#) [344](#) [345](#) [346](#) [347](#) [348](#) [349](#) [350](#) [351](#) [352](#) [353](#) [354](#) [355](#) [356](#) [357](#) [358](#) [359](#) [360](#) [361](#) [362](#) [363](#) [364](#) [365](#) [366](#) [367](#) [368](#) [369](#) [370](#) [371](#) [372](#) [373](#) [374](#) [375](#) [376](#) [377](#) [378](#) [379](#) [380](#) [381](#) [382](#) [383](#) [384](#) [385](#) [386](#) [387](#) [388](#) [389](#) [390](#) [391](#) [392](#) [393](#) [394](#) [395](#) [396](#) [397](#) [398](#) [399](#) [400](#) [401](#) [402](#) [403](#) [404](#) [405](#) [406](#) [407](#) [408](#) [409](#) [410](#) [411](#) [412](#) [413](#) [414](#) [415](#) [416](#) [417](#) [418](#) [419](#) [420](#) [421](#) [422](#) [423](#) [424](#) [425](#) [426](#) [427](#) [428](#) [429](#) [430](#) [431](#) [432](#) [433](#) [434](#) [435](#) [436](#) [437](#) [438](#) [439](#) [440](#) [441](#) [442](#) [443](#) [444](#) [445](#) [446](#) [447](#) [448](#) [449](#) [450](#) [451](#) [452](#) [453](#) [454](#) [455](#) [456](#) [457](#) [458](#) [459](#) [460](#) [461](#) [462](#) [463](#) [464](#) [465](#) [466](#) [467](#) [468](#) [469](#) [470](#) [471](#) [472](#) [473](#) [474](#) [475](#) [476](#) [477](#) [478](#) [479](#) [480](#) [481](#) [482](#) [483](#) [484](#) [485](#) [486](#) [487](#) [488](#) [489](#) [490](#) [491](#) [492](#) [493](#) [494](#) [495](#) [496](#) [497](#) [498](#) [499](#) [500](#) [501](#) [502](#) [503](#) [504](#) [505](#) [506](#) [507](#) [508](#) [509](#) [510](#) [511](#) [512](#) [513](#) [514](#) [515](#) [516](#) [517](#) [518](#) [519](#) [520](#) [521](#) [522](#) [523](#) [524](#) [525](#) [526](#) [527](#) [528](#) [529](#) [530](#) [531](#) [532](#) [533](#) [534](#) [535](#) [536](#) [537](#) [538](#) [539](#) [540](#) [541](#) [542](#) [543](#) [544](#) [545](#) [546](#) [547](#) [548](#) [549](#) [550](#) [551](#) [552](#) [553](#) [554](#) [555](#) [556](#) [557](#) [558](#) [559](#) [560](#) [561](#) [562](#) [563](#) [564](#) [565](#) [566](#) [567](#) [568](#) [569](#) [570](#) [571](#) [572](#) [573](#) [574](#) [575](#) [576](#) [577](#) [578](#) [579](#) [580](#) [581](#) [582](#) [583](#) [584](#) [585](#) [586](#) [587](#) [588](#) [589](#) [590](#) [591](#) [592](#) [593](#) [594](#) [595](#) [596](#) [597](#) [598](#) [599](#) [600](#) [601](#) [602](#) [603](#) [604](#) [605](#) [606](#) [607](#) [608](#) [609](#) [610](#) [611](#) [612](#) [613](#) [614](#) [615](#) [616](#) [617](#) [618](#) [619](#) [620](#) [621](#) [622](#) [623](#) [624](#) [625](#) [626](#) [627](#) [628](#) [629](#) [630](#) [631](#) [632](#) [633](#) [634](#) [635](#) [636](#) [637](#) [638](#) [639](#) [640](#) [641](#) [642](#) [643](#) [644](#) [645](#) [646](#) [647](#) [648](#) [649](#) [650](#) [651](#) [652](#) [653](#) [654](#) [655](#) [656](#) [657](#) [658](#) [659](#) [660](#) [661](#) [662](#) [663](#) [664](#) [665](#) [666](#) [667](#) [668](#) [669](#) [670](#) [671](#) [672](#) [673](#) [674](#) [675](#) [676](#) [677](#) [678](#) [679](#) [680](#) [681](#) [682](#) [683](#) [684](#) [685](#) [686](#) [687](#) [688](#) [689](#) [690](#) [691](#) [692](#) [693](#) [694](#) [695](#) [696](#) [697](#) [698](#) [699](#) [700](#) [701](#) [702](#) [703](#) [704](#) [705](#) [706](#) [707](#) [708](#) [709](#) [710](#) [711](#) [712](#) [713](#) [714](#) [715](#) [716](#) [717](#) [718](#) [719](#) [720](#) [721](#) [722](#) [723](#) [724](#) [725](#) [726](#) [727](#) [728](#) [729](#) [730](#) [731](#) [732](#) [733](#) [734](#) [735](#) [736](#) [737](#) [738](#) [739](#) [740](#) [741](#) [742](#) [743](#) [744](#) [745](#) [746](#) [747](#) [748](#) [749](#) [750](#) [751](#) [752](#) [753](#) [754](#) [755](#) [756](#) [757](#) [758](#) [759](#) [760](#) [761](#) [762](#) [763](#) [764](#) [765](#) [766](#) [767](#) [768](#) [769](#) [770](#) [771](#) [772](#) [773](#) [774](#) [775](#) [776](#) [777](#) [778](#) [779](#) [780](#) [781](#) [782](#) [783](#) [784](#) [785](#) [786](#) [787](#) [788](#) [789](#) [790](#) [791](#) [792](#) [793](#) [794](#) [795](#) [796](#) [797](#) [798](#) [799](#) [800](#) [801](#) [802](#) [803](#) [804](#) [805](#) [806](#) [807](#) [808](#) [809](#) [8010](#) [8011](#) [8012](#) [8013](#) [8014](#) [8015](#) [8016](#) [8017](#) [8018](#) [8019](#) [8020](#) [8021](#) [8022](#) [8023](#) [8024](#) [8025](#) [8026](#) [8027](#) [8028](#) [8029](#) [8030](#) [8031](#) [8032](#) [8033](#) [8034](#) [8035](#) [8036](#) [8037](#) [8038](#) [8039](#) [8040](#) [8041](#) [8042](#) [8043](#) [8044](#) [8045](#) [8046](#) [8047](#) [8048](#) [8049](#) [8050](#) [8051](#) [8052](#) [8053](#) [8054](#) [8055](#) [8056](#) [8057](#) [8058](#) [8059](#) [8060](#) [8061](#) [8062](#) [8063](#) [8064](#) [8065](#) [8066](#) [8067](#) [8068](#) [8069](#) [8070](#) [8071](#) [8072](#) [8073](#) [8074](#) [8075](#) [8076](#) [8077](#) [8078](#) [8079](#) [8080](#) [8081](#) [8082](#) [8083](#) [8084](#) [8085](#) [8086](#) [8087](#) [8088](#) [8089](#) [8090](#) [8091](#) [8092](#) [8093](#) [8094](#) [8095](#) [8096](#) [8097](#) [8098](#) [8099](#) [80100](#) [80101](#) [80102](#) [80103](#) [80104](#) [80105](#) [80106](#) [80107](#) [80108](#) [80109](#) [80110](#) [80111](#) [80112](#) [80113](#) [80114](#) [80115](#) [80116](#) [80117](#) [80118](#) [80119](#) [80120](#) [80121](#) [80122](#) [80123](#) [80124](#) [80125](#) [80126](#) [80127](#) [80128](#) [80129](#) [80130](#) [80131](#) [80132](#) [80133](#) [80134](#) [80135](#) [80136](#) [80137](#) [80138](#) [80139](#) [80140](#) [80141](#) [80142](#) [80143](#) [80144](#) [80145](#) [80146](#) [80147](#) [80148](#) [80149](#) [80150](#) [80151](#) [80152](#) [80153](#) [80154](#) [80155](#) [80156](#) [80157](#) [80158](#) [80159](#) [80160](#) [80161](#) [80162](#) [80163](#) [80164](#) [80165](#) [80166](#) [80167](#) [80168](#) [80169](#) [80170](#) [80171](#) [80172](#) [80173](#) [80174](#) [80175](#) [80176](#) [80177](#) [80178](#) [80179](#) [80180](#) [80181](#) [80182](#) [80183](#) [80184](#) [80185](#) [80186](#) [80187](#) [80188](#) [80189](#) [80190](#) [80191](#) [80192](#) [80193](#) [80194](#) [80195](#) [80196](#) [80197](#) [80198](#) [80199](#) [80200](#) [80201](#) [80202](#) [80203](#) [80204](#) [80205](#) [80206](#) [80207](#) [80208](#) [80209](#) [80210](#) [80211](#) [80212](#) [80213](#) [80214](#) [80215](#) [80216](#) [80217](#) [80218](#) [80219](#) [80220](#) [80221](#) [80222](#) [80223](#) [80224](#) [80225](#) [80226](#) [80227](#) [80228](#) [80229](#) [80230](#) [80231](#) [80232](#) [80233](#) [80234](#) [80235](#) [80236](#) [80237](#) [80238](#) [80239](#) [80240](#) [80241](#) [80242](#) [80243](#) [80244](#) [80245](#) [80246](#) [80247](#) [80248](#) [80249](#) [80250](#) [80251](#) [80252](#) [80253](#) [80254](#) [80255](#) [80256](#) [80257](#) [80258](#) [80259](#) [80260](#) [80261](#) [80262](#) [80263](#) [80264](#) [80265](#) [80266](#) [80267](#) [80268](#) [80269](#) [80270](#) [80271](#) [80272](#) [80273](#) [80274](#) [80275](#) [80276](#) [80277](#) [80278](#) [80279](#) [80280](#) [80281](#) [80282](#) [80283](#) [80284](#) [80285](#) [80286](#) [80287](#) [80288](#) [80289](#) [80290](#) [80291](#) [80292](#) [80293](#) [80294](#) [80295](#) [80296](#) [80297](#) [80298](#) [80299](#) [80300](#) [80301](#) [80302](#) [80303](#) [80304](#) [80305](#) [80306](#) [80307](#) [80308](#) [80309](#) [80310](#) [80311](#) [80312](#) [80313](#) [80314](#) [80315](#) [80316](#) [80317](#) [80318](#) [80319](#) [80320](#) [80321](#) [80322](#) [80323](#) [80324](#) [80325](#) [80326](#) [80327](#) [80328](#) [80329](#) [80330](#) [80331](#) [80332](#) [80333](#) [80334](#) [80335](#) [80336](#) [80337](#) [80338](#) [80339](#) [80340](#) [80341](#) [80342](#) [80343](#) [80344](#) [80345](#) [80346](#) [80347](#) [80348](#) [80349](#) [80350](#) [80351](#) [80352](#) [80353](#) [80354](#) [80355](#) [80356](#) [80357](#) [80358](#) [80359](#) [80360](#) [80361](#) [80362](#) [80363](#) [80364](#) [80365](#) [80366](#) [80367](#) [80368](#) [80369](#) [80370](#) [80371](#) [80372](#) [80373](#) [80374](#) [80375](#) [80376](#) [80377</a](#)

# The new generation

---

## Genealogy (Manes et al. 2019)



# Driller (2016)

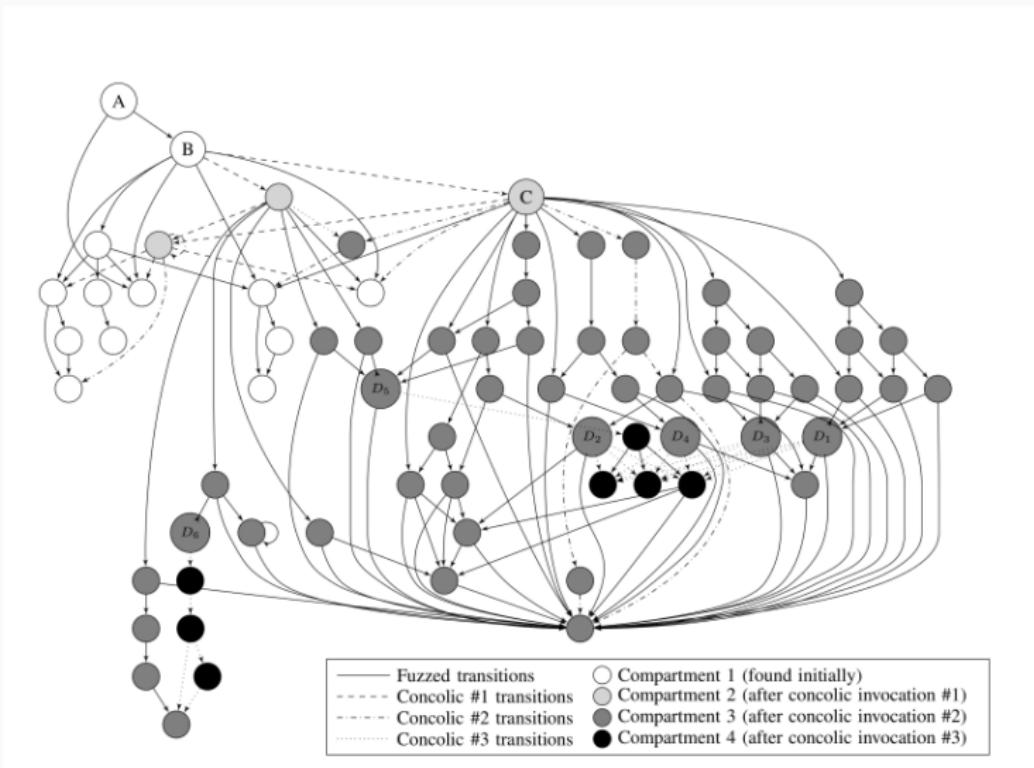
## Goal

Combination of SE & greybox fuzzing

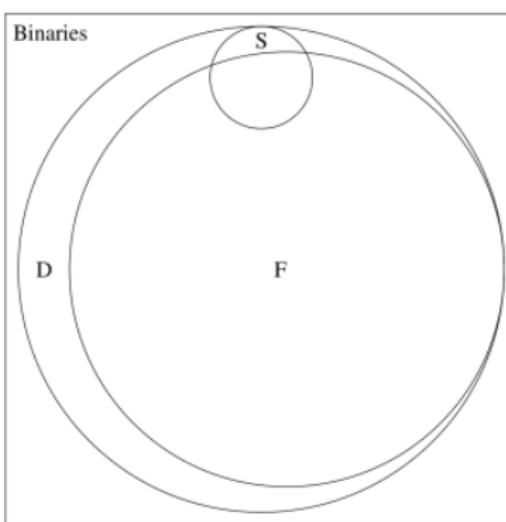
## Key insight

When the fuzzer finds "hard conditions", launch SE to solve them, then let the fuzzer continue with this new input.

# Driller : Inner workings



# Driller : Results



Method	Crashes Found
Fuzzing	68
Fuzzing $\cap$ Driller	68
Fuzzing $\cap$ Symbolic	13
Symbolic	16
Symbolic $\cap$ Driller	16
Driller	77

# Vuzzer (2017) i

## Proposal

Configurations are added only upon discovering a new non-error handling block (statically determined).

The configuration fitness is the weighted sum of the log of the frequency over exercised blocks.

## Key insight

Error-handling blocks lower the chance of vulnerabilities.

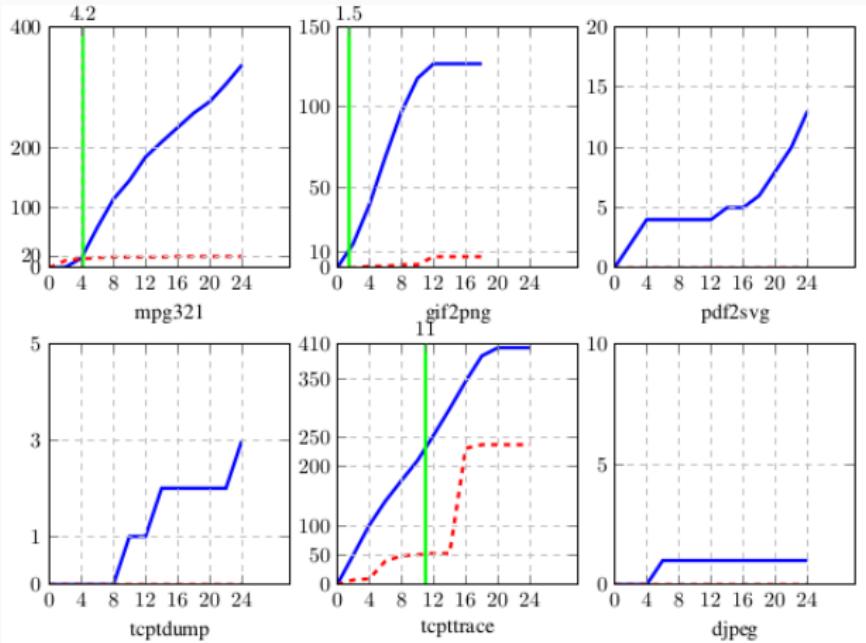
## Consequence

VUzzer prefers normal blocks that are rare according to CFG random walks.

# Vuzzer (2017) ii

Application	VUzzer		AFL	
	#Unique crashes	#Inputs	#Unique crashes	#Inputs
mpg321	337	23.6K	19	88.3K
gif2png+libpng	127	43.2K	7	1.84M
pdf2svg+libpoppler	13	5K	0	923K
tcpdump+libpcap	3	77.8K	0	2.89M
tcptrace+libpcap	403	30K	238	3.29M
djpeg+libjpeg	1'	90K	0	35.9M

# Vuzzer (2017) iii



# Vuzzer (2017) iv

Program	Bug Type	Already fixed?	Reported?
tcpdump	Out-of-bounds Read	Yes	No
mpg321	Out-of-bounds Read	No	Yes [2]
mpg321	Double free	No	Yes [3]
pdf2svg	Null pointer deref (write)	Seems to be fixed in poppler 0.49	No
pdf2svg	Abort	Seems to be fixed in poppler 0.49	No
pdf2svg	Assert fail (abort)	Yes [4]	No
tcptrace	Out-of-bounds Read	No	Yes [5]
gif2png	Out-of-bounds Read	No	Yes [6]

# AFLGo (2017) i

## Goal

Generating inputs with the objective of reaching a set of program locations.

Key idea : Directed Greybox Fuzzing

## Applications

- Patch testing
- Crash reproduction
- Static analysis report verification
- Information flow detection

# AFLGo (2017) ii

## Measure

- Distance between a function & a set of target functions  
= harmonic mean
- Harmonic mean can distinguish between a node that is close to one target and further from another and one that is equidistant from both (average mean may be equal).

## Scheduling

Key insight :: simulated annealing

Use more energy to fuzz seeds closer to the targets.

Enter exploitation after given exploration time has elapsed.

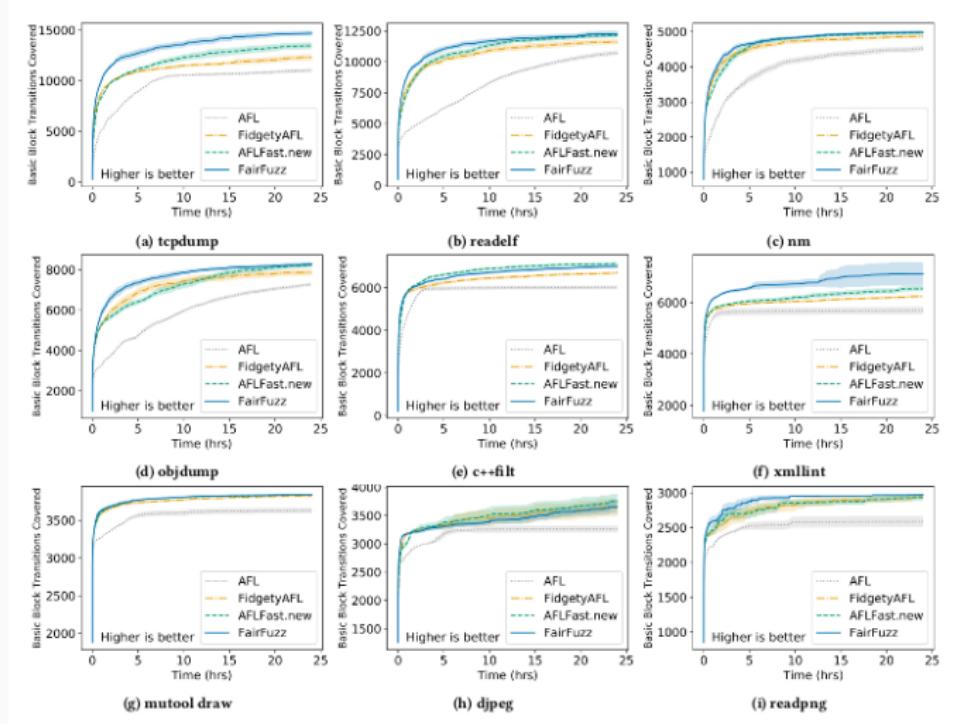
## Goal

- Achieve better branch coverage for AFL
- No extra instrumentation

## Key steps

1. Identify *rare* branches
2. New mutation technique to increase probability of hitting rare branches.

# FairFuzz (2018) ii



# Angora (2018) i

## Goal

Increase branch coverage by solving path constraints *without symbolic execution*

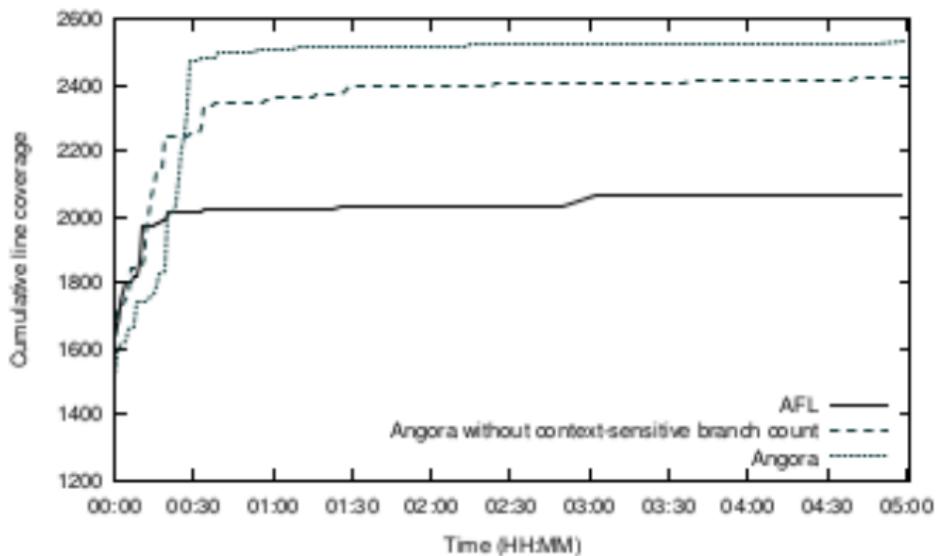
## Key ingredients

- Context-sensitive branch coverage
- Byte-level taint tracking
- Gradient descent-based search
- Type & shape inference

## Angora (2018) ii

Program	Listed bugs	Bugs found by each fuzzer					
		Angora	AFL	FUZZER	SES	VUzzer	Steelix
<i>uniq</i>	28	29	9	7	0	27	7
<i>base64</i>	44	48	0	7	9	17	43
<i>md5sum</i>	57	57	0	2	0	Fail	28
<i>who</i>	2136	1541	1	0	18	50	194

# Angora (2018) iii



# Angora (2018) iv

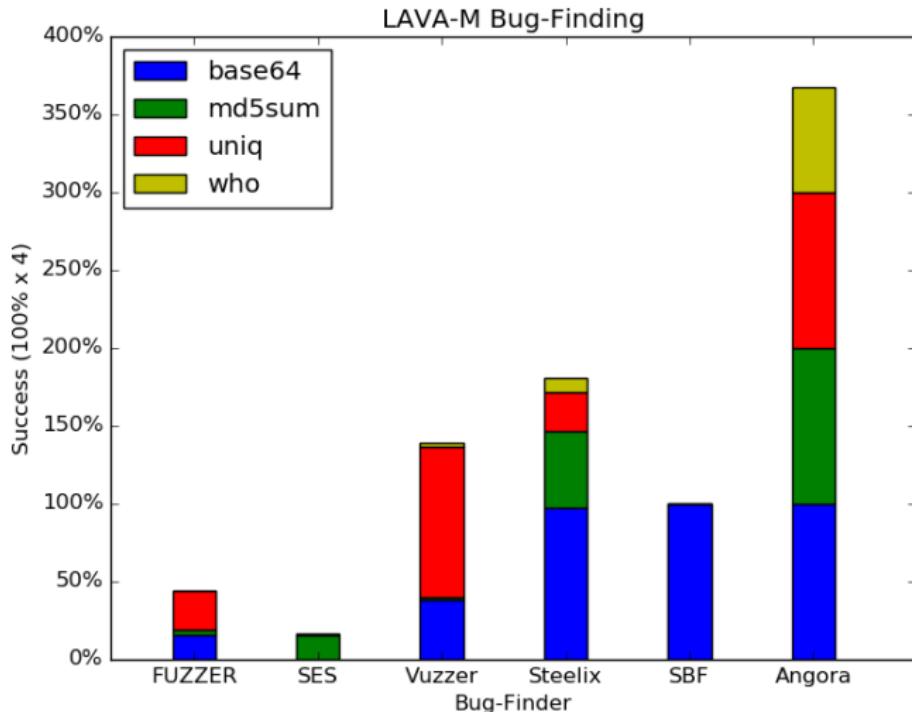
Program	Argument	Size (kB)	Line coverage			Branch coverage			Unique crashes	
			AFL	Angora	Increase	AFL	Angora	Increase	AFL	Angora
<i>file-5.32</i>		617	2070	2534	21.2 %	1462	1899	29.9 %	0	6
<i>jhead-3.00</i>		120	347	789	127.4 %	218	532	144.0 %	19	52
<i>xmllwf(expat)-2.2.5</i>		791	1980	2025	2.3 %	2905	3158	8.7 %	0	0
<i>djpeg(jpeg)-v9b</i>		790	5401	5509	2.0 %	1677	1782	6.3 %	0	0
<i>readpng(libpng)-1.6.34</i>		972	1592	1799	13.0 %	872	1007	15.5 %	0	0
<i>nm-2.29</i>	-C	6252	6372	7721	21.2 %	4105	4693	14.3 %	12	29
<i>objdump-2.29</i>	-x	9063	3448	6216	80.3 %	2071	3393	63.8 %	4	40
<i>size-2.29</i>		6207	2839	4832	70.2 %	1792	2727	52.2 %	6	48

# Achievements summary

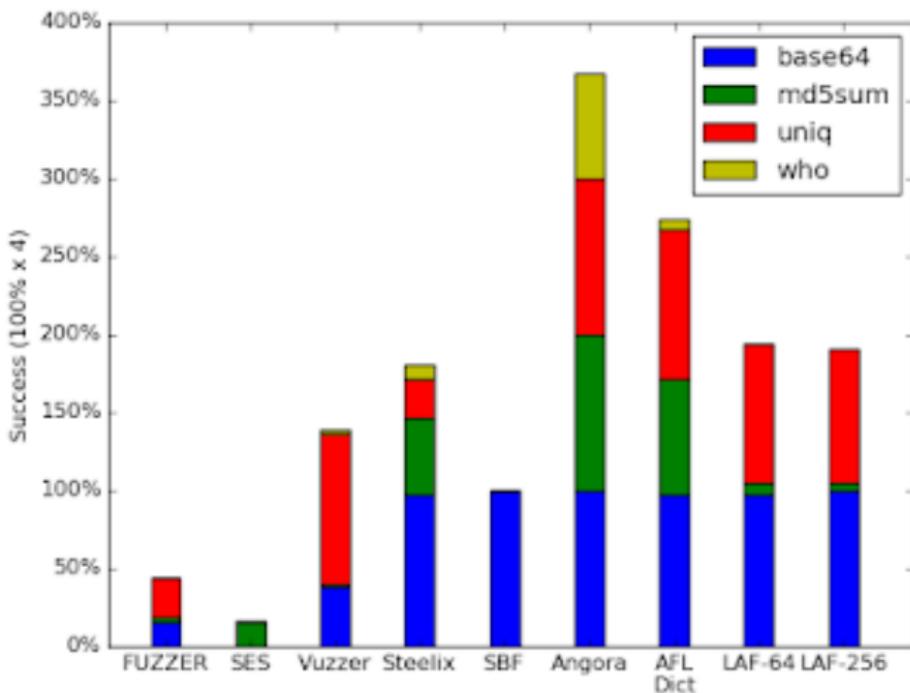
Ideas have been explored

- Better branch coverage (deeper, broader)
- Directed targeting
- Lightweight dynamic constraint solving
- Combination with other analyses:
  - Symbolic execution
  - Static analyses

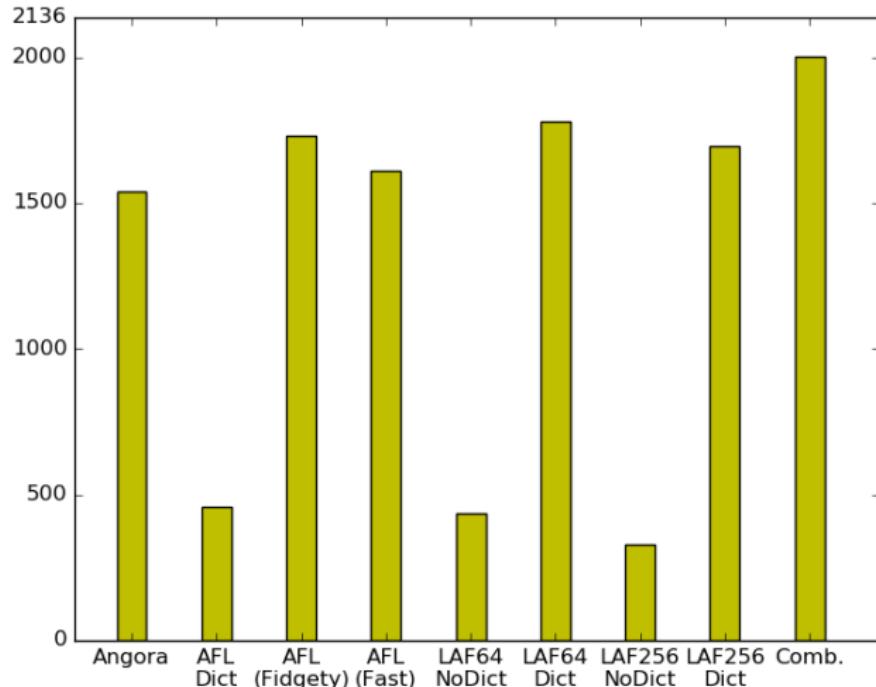
# Standard benchmarks: an emerging concern



# Baseline



# Who's who



# More goodness to come . . .

Fuzzing is a **very active** research area

## New developments in 2019

- Hawkeye (CCS '18)
- DigFuzz (NDSS '19)
- MemFuzz (ICSE '19)
- ...

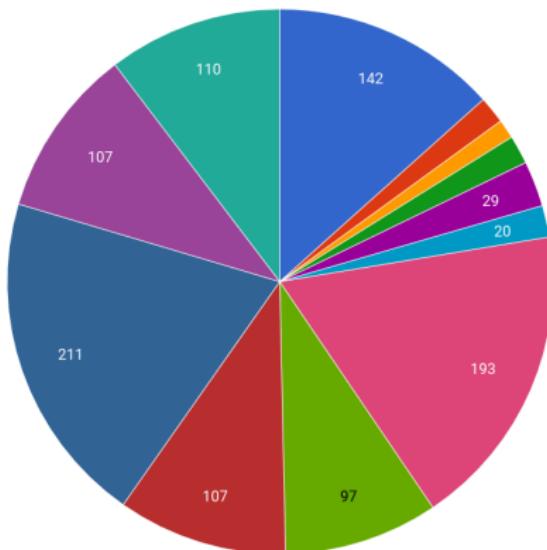
# FAAS

Recent years have seen initiatives from "MAANG" members

**Google OSS-Fuzz**

**Microsoft Project Springfield**

# Types of bugs (OSS-Fuzz 2016)



- heap buffer overflows
- global buffer overflows
- stack buffer overflows
- use after frees
- uninitialized memory
- stack overflows
- timeouts
- ooms
- leaks
- ubsan
- unknown crashes
- other (e.g. assertions)

# Questions ?



<https://rbonichon.github.io/teaching/2019/asi36/>