

# Poster: A Tight Integration of Symbolic Execution and Fuzzing

*Sébastien Bardin, Yaëlle Vinçont, Michaël Marcozzi*



September 15-16, 2022  
London, UK

# About Fuzzing

- Automatic test input generation
- Black-box fuzzing = random generation (usually specification-based)



2000+ bugs in GCC/LLVM  
1000+ bugs in Z3/CVC4  
400+ bugs in SQL DBMS

- Grey-box fuzzing = semi-random generation (code coverage feedback)



```
american fuzzy top 0.47b (reading)
process timing : 0 days, 0 hrs, 4 min, 43 sec
last run path : 0 days, 0 hrs, 0 min, 26 sec
last uniq crash : none seen yet
last uniq hang : 0 days, 0 hrs, 1 min, 51 sec
cycle progress :
now processing : 38 (19.49%)
paths timed out : 0 (0.00%)
stage progress :
now trying : interest 32/8
stage execs : 0/990 (0.00%)
total execs : 654k
exec speed : 2306/sec
fuzzing strategy yields :
bit flips : 80/14.4k, 6/14.4k, 6/14.4k
byte flips : 0/189, 0/189, 1/150
arithmetics : 31/126k, 3/45.6k, 1/17.8k
known ints : 1/15.8k, 4/65.8k, 6/78.2k
havoc : 34/254k, 0/0
trim : 2876 8/931 (61.45% gain)
overall results :
cycles done : 0
total paths : 195
total crashes : 0
uniq crashes : 0
uniq hangs : 1
map coverage : 1217 (7.43%)
code coverage : 2.55 bits/tuple
Findings in depth :
favored paths : 128 (65.66%)
new edges on : 85 (43.99%)
total crashes : 0 (0 unique)
total hangs : 1 (1 unique)
path geometry :
levels : 3
pending : 178
pend fav : 114
imported : 0
variable : 0
latent : 0
```

Many CVEs and bugs in many apps:  
iOS, Firefox, dpkg, OpenSSH, etc.

# Fuzzing vs Symbolic Execution

- Symbolic execution = white-box fuzzing (path constraints solving)
- Grey-box and white-box fuzzing seem complementary
  - Grey-box fuzzing: no *path explosion*, nor *complex path constraints*
  - Symbolic execution: easily penetrates *paths guarded by an infrequent condition*
- Several tools aim at hybrid fuzzing (best of both worlds)

DRILLER

Qsym

Pangolin

Angora

MATRYOSHKA

Eclipser

# Confuzz: Goals and Principles

	Analysis of path constraints				Fuzzing		Well-integrated components
	Symbolic	Cheap	Targeted	Correct	Efficient	Constraints	
Fuzzing SE	-	-	-	-	✓	✗	-
Driller	✓	✗	✓	✓	✓	✗	✗
Qsym	✓	✓	✓	✗	✓	✗	✗
Pangolin	✓	✓	✓	✗	✓	✓	✓
Angora	✗	✓	✓	✗	~	✓	ok
Matryoshka	✗	✓	✓	✗	~	✓	ok
Eclipser	✗	✓	✗	✗	✓	✗	✗
<b>ConFuzz</b>	✓	✓	✓	✓	✓	✓	✓

- Confuzz = yet another hybrid fuzzing tool
- Goal: be better than the others 😊
- Based on two main principles:
  - Rely on grey-box fuzzing to explore path space
  - For paths beyond infrequent conditions:
    - Create *easy-enumerable* version of prefix constraint
    - Use the grey-box fuzzer to *find correct solutions*

# Preliminary Evaluation and Future Work

- Good **preliminary results** vs AFL++ and KLEE, on 3 LAVA-M programs

		AFL	AFL++	KLEE	ConFuzz
<b>base64</b>	<b>Avg</b>	0	0.2	10.0	<b>38.8</b>
<b>3kloc</b>	<b>Dev (<math>\sigma</math>)</b>	0	0.4	1.3	0.4
<b>44 bugs</b>	$\hat{A}_{12}$	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	-
<b>md5sum</b>	<b>Avg</b>	0	0	0	<b>9</b>
<b>3kloc</b>	<b>Dev (<math>\sigma</math>)</b>	0	0	0	1.7
<b>57 bugs</b>	$\hat{A}_{12}$	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	-
<b>uniq</b>	<b>Avg</b>	0	0.4	5	<b>26.9</b>
<b>3kloc</b>	<b>Dev (<math>\sigma</math>)</b>	0	0.5	0	3.6
<b>28 bugs</b>	$\hat{A}_{12}$	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	-

- TODO: **extend** constraint language, **evaluate** at scale vs s.o.t.a. tools