

Fuzzing

For Automated Software Testing

Cornelius Aschermann

Ruhr University Bochum

Verifaction & Automated Bug finding

Security Consultant

 @is_eqv

 github.com/eqv

 cornelius.aschermann@rub.de



Testing



Testing

- + Testing is **AWSOME!**
(But...)



Testing

- + Testing is **AWSOME!**
(But...)
- Expensive
30% or more



Testing

- + Testing is **AWSOME!**
(But...)
- Expensive
30% or more
- Cognitive biases



Testing

- + Testing is **AWESOME!**
(But...)
- Expensive
30% or more
- Cognitive biases
- Pathological cases



Testing

- + Testing is **AWSOME!**
(But...)
- Expensive
30% or more
- Cognitive biases
- Pathological cases
- Security testing



Testing

- + Testing is **AWSOME!**
(But...)
- Expensive
30% or more
- Cognitive biases
- Pathological cases
- Security testing
- Integration tests are hard



Manually
Created
Testcases



Verification

Verifi~~cation~~tion

Fuzzers

How do Fuzzer Work?

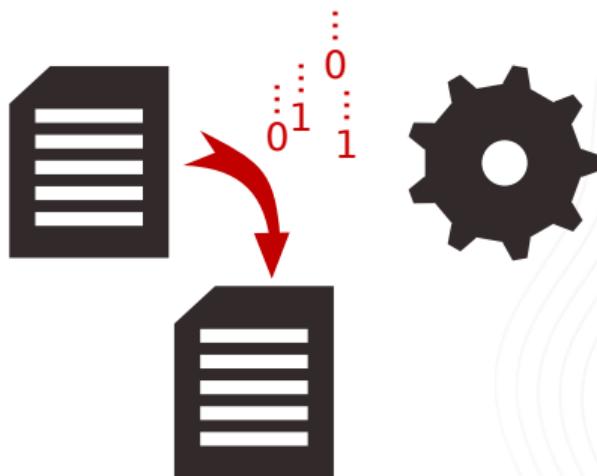
How do Fuzzer Work?



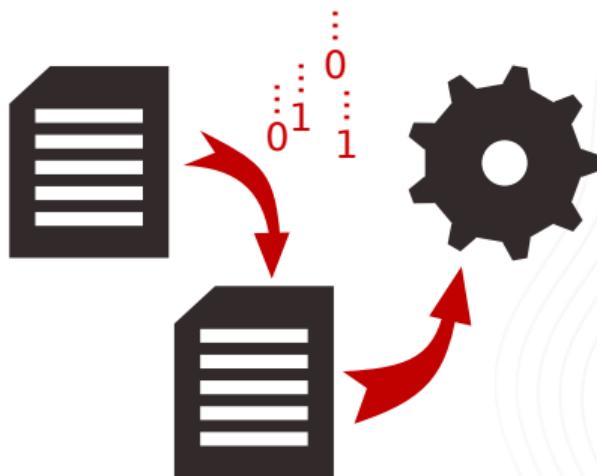
How do Fuzzer Work?



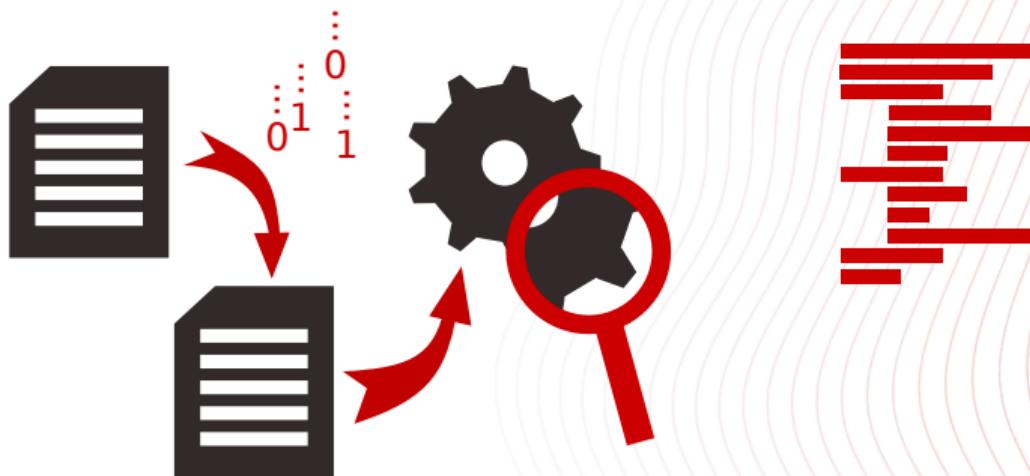
How do Fuzzer Work?



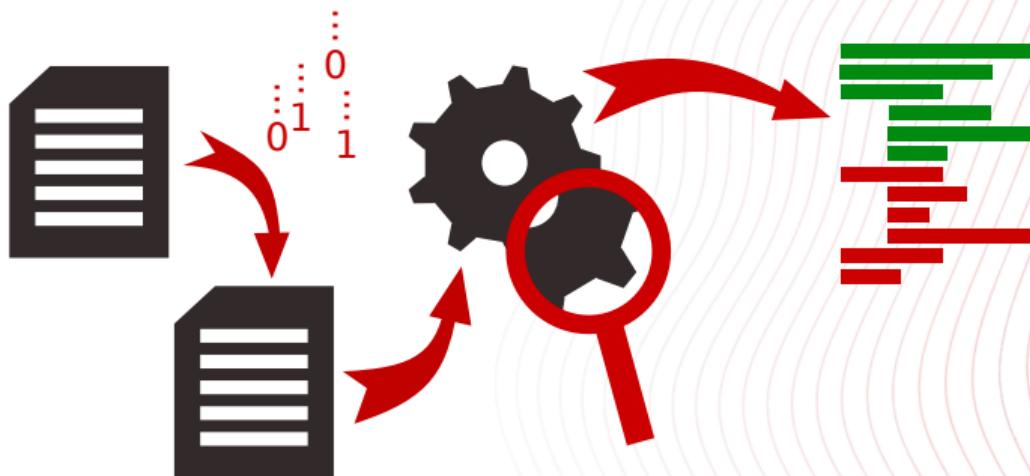
How do Fuzzer Work?



How do Fuzzer Work?



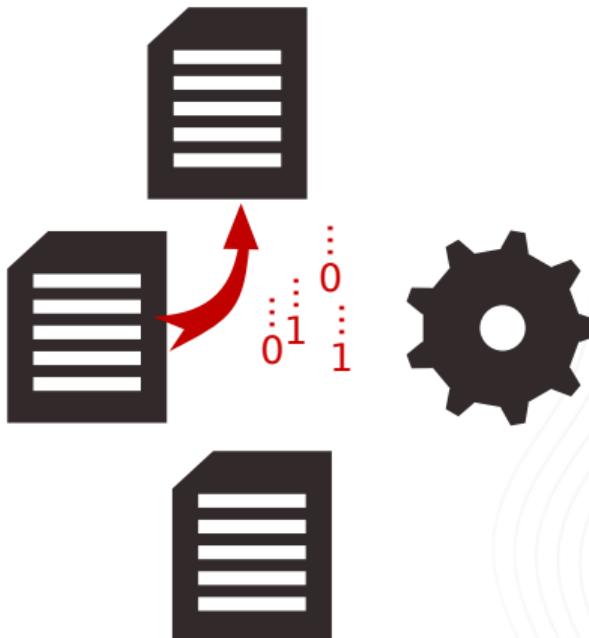
How do Fuzzer Work?



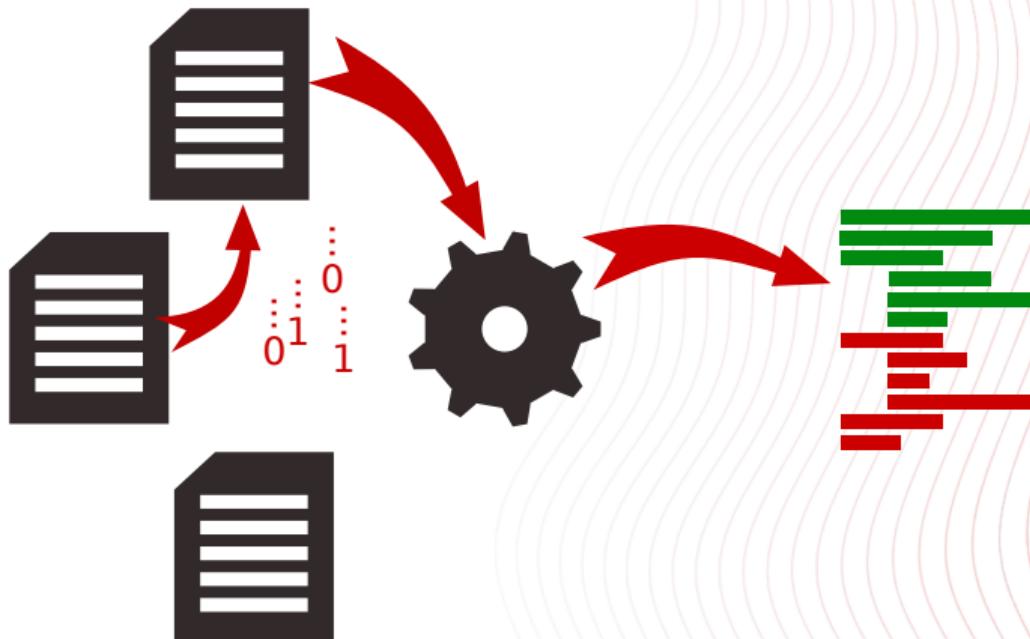
How do Fuzzer Work?



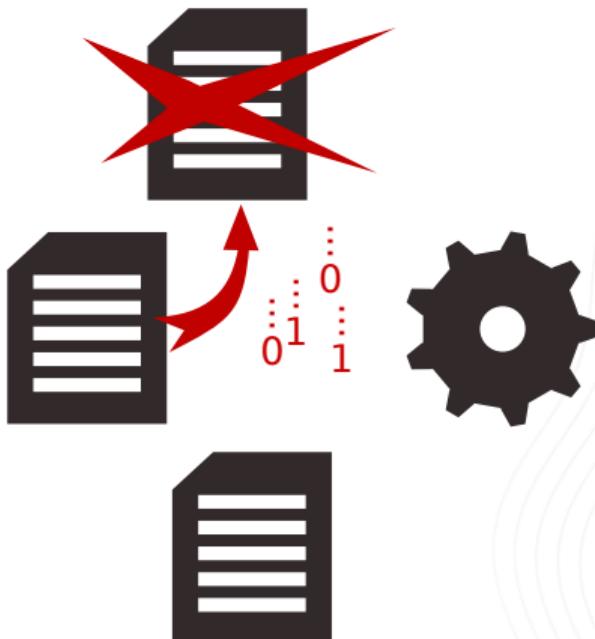
How do Fuzzer Work?



How do Fuzzer Work?



How do Fuzzer Work?



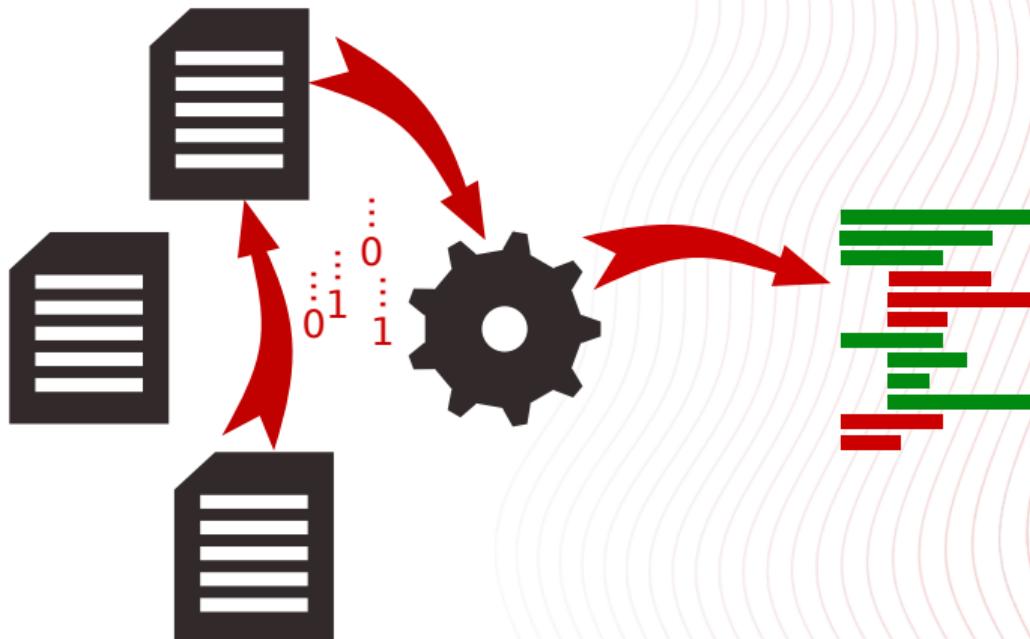
How do Fuzzer Work?



How do Fuzzer Work?



How do Fuzzer Work?

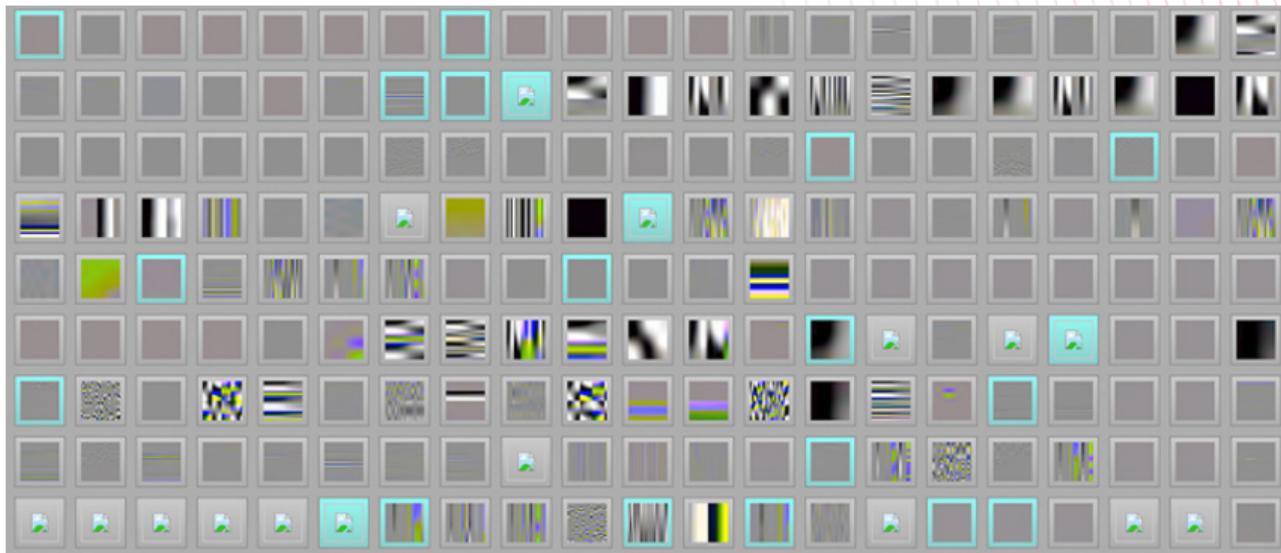


How do Fuzzer Work?





TESTING
STAGE



[1] <https://lcamtuf.blogspot.com/2014/11/pulling-jpegs-out-of-thin-air.html>



@Microsoft

>70% of patched vulnerabilities found by fuzzing [1]

>33% of "Million Dollar Bugs" [2]

SDL mandates fuzzing! [3]

[1] https://www.owasp.org/images/5/5b/OWASP_IL_7_FuzzGuru.pdf

[2] <https://www.microsoft.com/en-us/security-risk-detection/>

[3] [https://docs.microsoft.com/en-us/previous-versions/software-testing/cc162782\(v=msdn.10\)](https://docs.microsoft.com/en-us/previous-versions/software-testing/cc162782(v=msdn.10))

@Google

> **9000** Bugs in external software [1]

31 Vulnerabilities in Browsers for 1000\$ compute [2]

Google develops multiple fuzzers!

[1] <https://security.googleblog.com/2018/11/a-new-chapter-for-oss-fuzz.html>

[2] <https://googleprojectzero.blogspot.com/2017/09/the-great-dom-fuzz-off-of-2017.html>

@Our Research

> **60** Vulnerabilities

Operation Systems

Programming languages

Proprietary third party components

Test Coverage

MRuby: 55% (+6 Vulns)

Lua: 60%

Fuzzers

Bascially Testcase Generation Als



Fuzzing

- + Cheap



Fuzzing

- + Cheap
- + Avoids biases



Fuzzing

- + Cheap
- + Avoids biases
- + Pathological cases



Fuzzing

- + Cheap
- + Avoids biases
- + Pathological cases
- + Security testing



Fuzzing

- + Cheap
- + Avoids biases
- + Pathological cases
- + Security testing
- + Integration Test Coverage



Fuzzing

- + Cheap
- + Avoids biases
- + Pathological cases
- + Security testing
- + Integration Test Coverage
- Error Detection

Fuzzing

For Automated Software Testing

Case Studies: Integration Level

Case Studies: Integration Level

Kernel Drivers

Full OS context

17 Vulns

Case Studies: Integration Level

Kernel Drivers

Full OS context

17 Vulns

Programming Languages

7 Vulns

Case Studies: Integration Level

Kernel Drivers

Full OS context

17 Vulns

Programming Languages

7 Vulns

Django Apps

Increased Coverage

Case Studies: Integration Level

Kernel Drivers

Full OS context

17 Vulns

Programming Languages

7 Vulns

Django Apps

Increased Coverage

Hypervisors (WIP)

X Vulns

Case Studies: Integration Level

Kernel Drivers

Full OS context

17 Vulns

Programming Languages

7 Vulns

Django Apps

Increased Coverage

Hypervisors (WIP)

X Vulns

+ Inject faults

+ Performance tests

Case Studies: Blackbox

Case Studies: Blackbox

Malicious Counter-Strike 1.6 servers used zero-days to infect users with malware

DrWeb: 39 percent of all Counter-Strike 1.6 servers were malicious and tried to infect users with malware.



By Catalin Cimpanu for Zero Day | March 13, 2019 -- 21:25 GMT (21:25 GMT) | Topic: Security

Case Studies: Contracts

Case Studies: Contracts

```
def some_fn(self, xs, ys):  
  
    for (a,b) in zip(xs,ys):  
        some(a,b)
```

Case Studies: Contracts

```
def some_fn(self, xs, ys):  
  
    for (a,b) in zip(xs,ys):  
        some(a,b)
```

Case Studies: Contracts

```
def some_fn(self, xs, ys):  
  
    for (a,b) in zip(xs,ys):  
        some(a,b)
```

Silent if length is different



Case Studies: Contracts

```
def some_fn(self, xs, ys):
    assert( len(xs) == len(ys) )
    for (a,b) in zip(xs,ys):
        some(a,b)
```

Case Studies: Contracts

Must NOT invalidate cache



```
def some_fn(self, callback):
    self.calc_cache()
    callback(self)
```

Case Studies: Contracts

```
def some_fn(self, callback):
    self.calc_cache()
    callback(self)
    assert(self.cache_valid())
```

Case Studies: Contracts

```
def some_fn(self, xs, ys):
    assert( len(xs) == len(ys) )
    for (a,b) in zip(xs,ys):
        some(a,b)
```

```
def some_fn(self, callback):
    self.calc_cache()
    callback(self)
    assert(self.cache_valid())
```

Encode assumptions in Assertions,
If violation leads to hard-to-track or silent errors!

(assertions can be disabled using python -O)

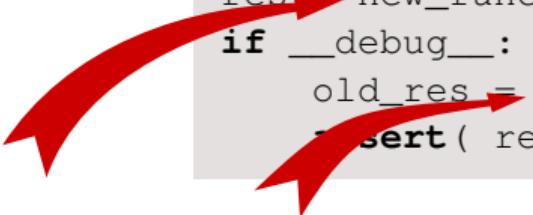
Case Studies: Refactoring

Case Studies: Refactoring

```
res = new_func(x)
if __debug__:
    old_res = old_func(x)
    assert( res == old_res )
```

Case Studies: Refactoring

```
res = new_func(x)
if __debug__:
    old_res = old_func(x)
    assert( res == old_res )
```



Edge cases of BOTH functions

Case Studies: Refactoring

```
res = new_func(x)
if __debug__:
    old_res = old_func(x)
    assert( res == old_res )
```

For all x : $\text{new_func}(x) == \text{old_func}(x)$

Case Studies: Refactoring

```
res = new_func(x)
if __debug__:
    old_res = old_func(x)
    assert( res == old_res )
```

For most x : $\text{new_func}(x) == \text{old_func}(x)$

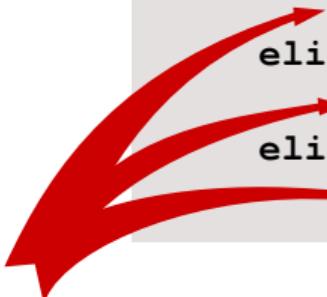
Case Studies: APIs

Case Studies: APIs

```
while len(input) > 0 :  
    case = input.pop()  
    if case==1:  
        api_fn_1(input.pop())  
    elif case==2:  
        api_fn_2(input.pop(), input.pop())  
    elif case==3:  
        api_fn_3(input.pop())
```

Case Studies: APIs

```
while len(input) > 0 :  
    case = input.pop()  
    if case==1:  
        api_fn_1(input.pop())  
    elif case==2:  
        api_fn_2(input.pop(), input.pop())  
    elif case==3:  
        api_fn_3(input.pop())
```



Fuzzer tries different combinations of all API calls

Case Studies: APIs

api_fn_1(0)
api_fn_3(100000000)
api_fn_3(0)
api_fn_3(2145)
api_fn_1(0)
api_fn_2(90362,0)
api_fn_3(0)
api_fn_2(1,1)

api_fn_1(-1)	api_fn_1(0)	api_fn_3(0)	api_fn_3(24282)
api_fn_3(83308)	api_fn_1(0)	api_fn_3(1)	api_fn_2(-1,74092)
api_fn_3(1)	api_fn_3(1)	api_fn_2(0)	api_fn_2(7289,1)
api_fn_3(-1)	api_fn_2(0,1)	api_fn_2(18273,-1)	api_fn_3(1)
api_fn_2(1,1)	api_fn_2(1,1)	api_fn_1(-1)	api_fn_1(-1)
api_fn_1(0)	api_fn_2(1,172393)	api_fn_2(80655,1)	api_fn_2(57319,-1)
api_fn_2(354,0)			

■ ■ ■

Property Based Testing

Property Based Testing

```
optimized(x) == slow(x) //refactoring example
```

Property Based Testing

```
optimized(x) == slow(x) //refactoring example
```

```
clean_data(x) == clean_data(clean_data(x))
```

Property Based Testing

```
optimized(x) == slow(x) //refactoring example
```

```
clean_data(x) == clean_data(clean_data(x))
```

```
len(str) == len(to_lower(str)) //woopsie
```

Property Based Testing

optimized(x) == slow(x) //refactoring example

clean_data(x) == clean_data(clean_data(x))

len(str) == len(to_lower(str)) //woopsie

speach_recognize(x) == speach_recognize(speedup(x))

Property Based Testing

optimized(x) == slow(x) //refactoring example

clean_data(x) == clean_data(clean_data(x))

len(str) == len(to_lower(str)) //woopsie

speach_recognize(x) == speach_recognize(speedup(x))

encode(decode(x)) == x

Property Based Testing

optimized(x) == slow(x) //refactoring example

clean_data(x) == clean_data(clean_data(x))

len(str) == len(to_lower(str)) //woopsie

speach_recognize(x) == speach_recognize(speedup(x))

encode(decode(x)) == x

//Nothing crashes



THE CLASSIC

Targets C/C++ ...

Blackbox components (AFL-Qemu)

Coverage Guided

Found a terrifying amount of bugs

american fuzzy lop 0.47b (readpng)		overall results
process timing	run time : 0 days, 0 hrs, 4 min, 43 sec	cycles done : 0
	last new path : 0 days, 0 hrs, 0 min, 26 sec	total paths : 195
	last uniq crash : none seen yet	uniq crashes : 0
	last uniq hang : 0 days, 0 hrs, 1 min, 51 sec	uniq hangs : 1
cycle progress	now processing : 38 (19.49%)	map coverage
	paths timed out : 0 (0.00%)	map density : 1217 (7.43%)
stage progress	now trying : interest 32/8	count coverage : 2.55 bits/tuple
	stage execs : 0/9990 (0.00%)	findings in depth
	total execs : 654k	favored paths : 128 (65.64%)
	exec speed : 2306/sec	new edges on : 85 (43.59%)
fuzzing strategy yields	bit flips : 88/14.4k, 6/14.4k, 6/14.4k	total crashes : 0 (0 unique)
	byte flips : 0/1804, 0/1786, 1/1750	total hangs : 1 (1 unique)
	arithmetics : 31/126k, 3/45.6k, 1/17.8k	path geometry
	known ints : 1/15.8k, 4/65.8k, 6/78.2k	levels : 3
	havoc : 34/254k, 0/0	pending : 178
	trim : 2876 B/931 (61.45% gain)	pend fav : 114
		imported : 0
		variable : 0
		latent : 0

```
$ make
```



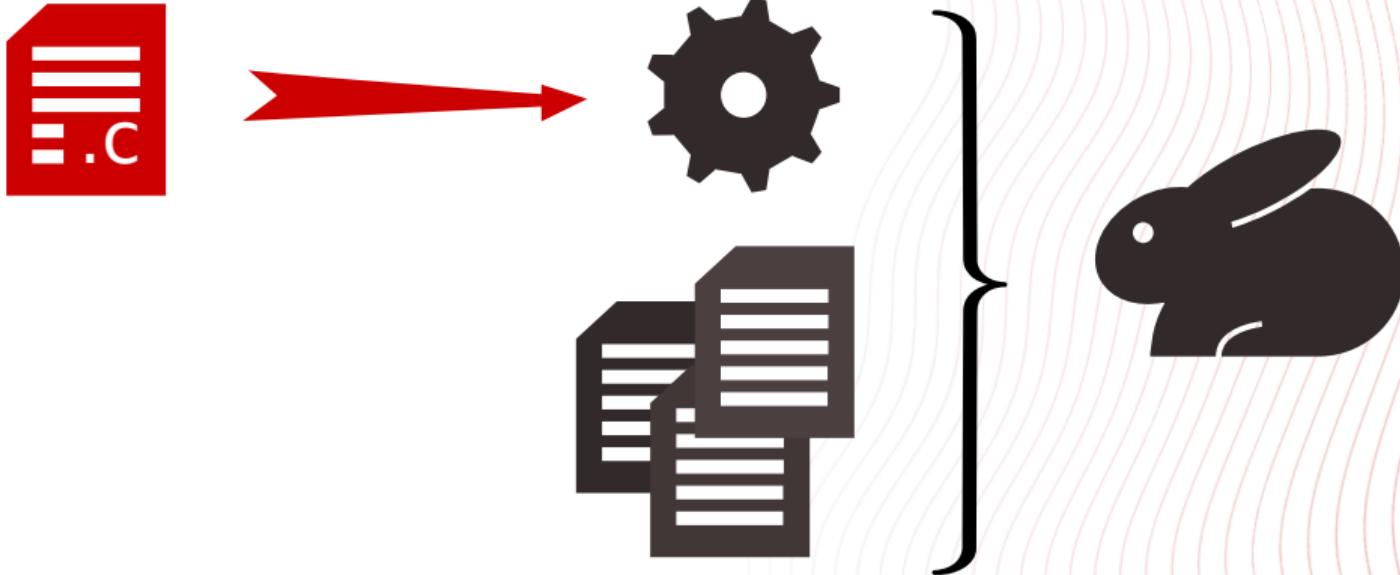
```
$ CC=afl_clang_fast make
```



```
$ CC=afl_clang_fast make
```



```
$ CC=afl_clang_fast make
```



```
$ afl-fuzz -i seeds/ -o output/ -- target @@
```

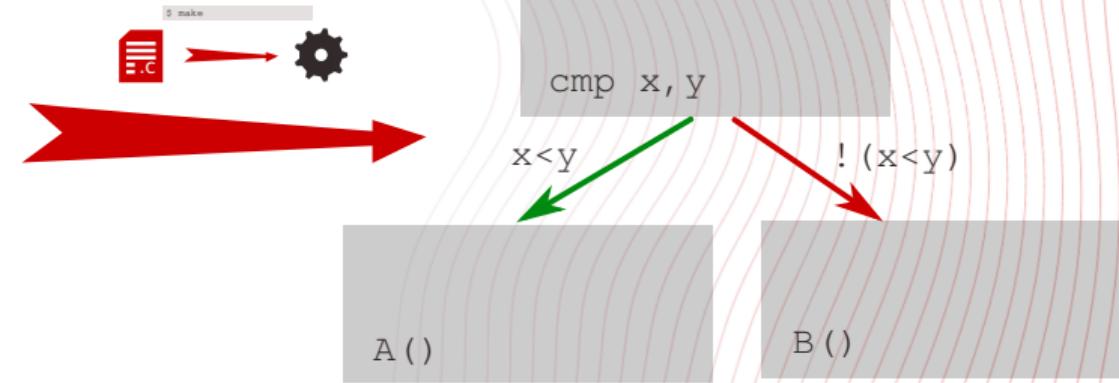
Coverage

Coverage

```
if(x<y) {  
    A()  
} else {  
    B()  
}
```

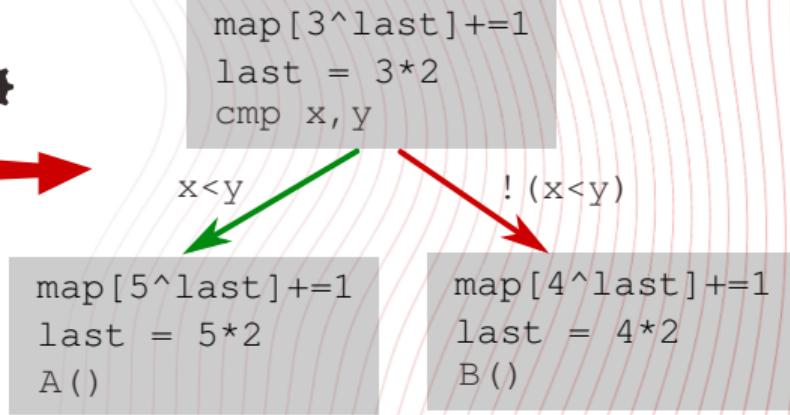
Coverage

```
if(x<y) {  
    A()  
} else {  
    B()  
}
```

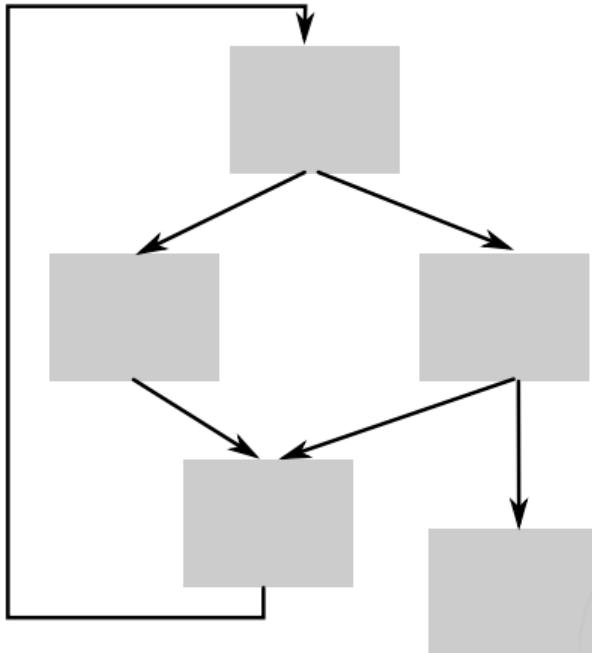


Coverage

```
if(x<y) {  
    A()  
} else {  
    B()  
}
```



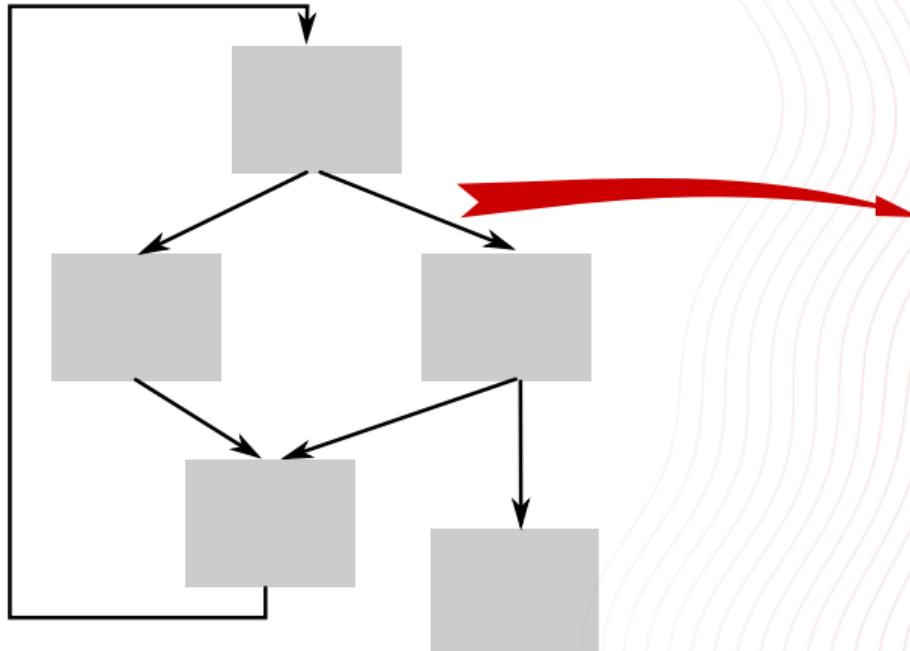
Coverage



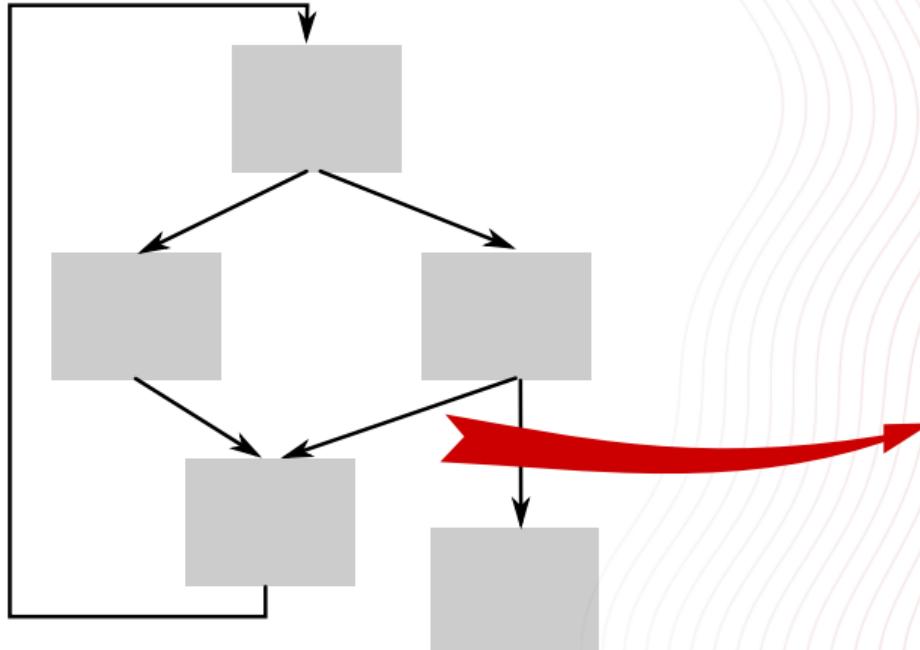
Bitmap



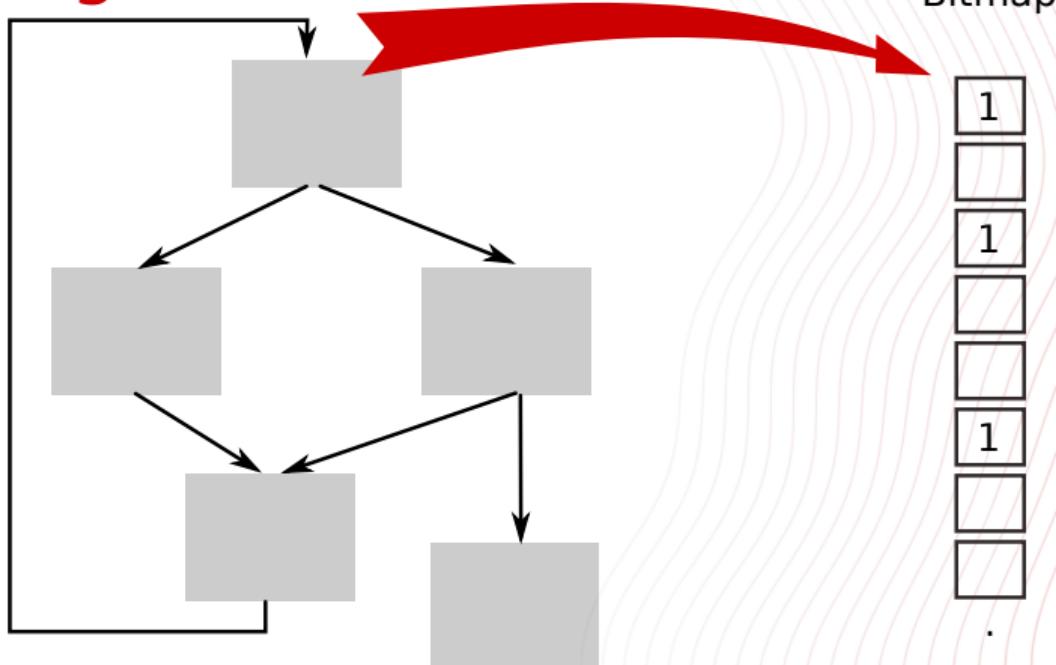
Coverage



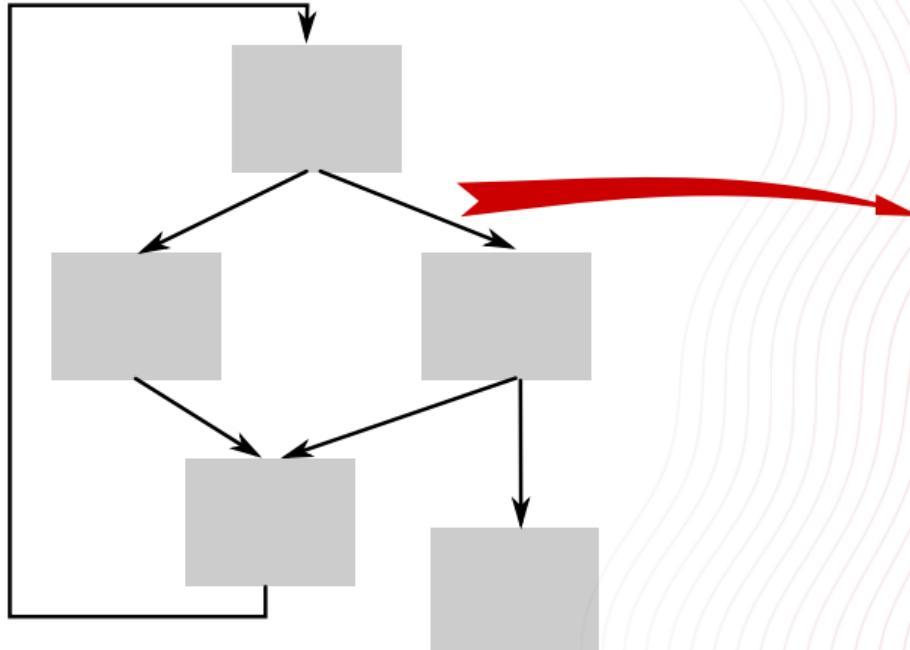
Coverage



Coverage



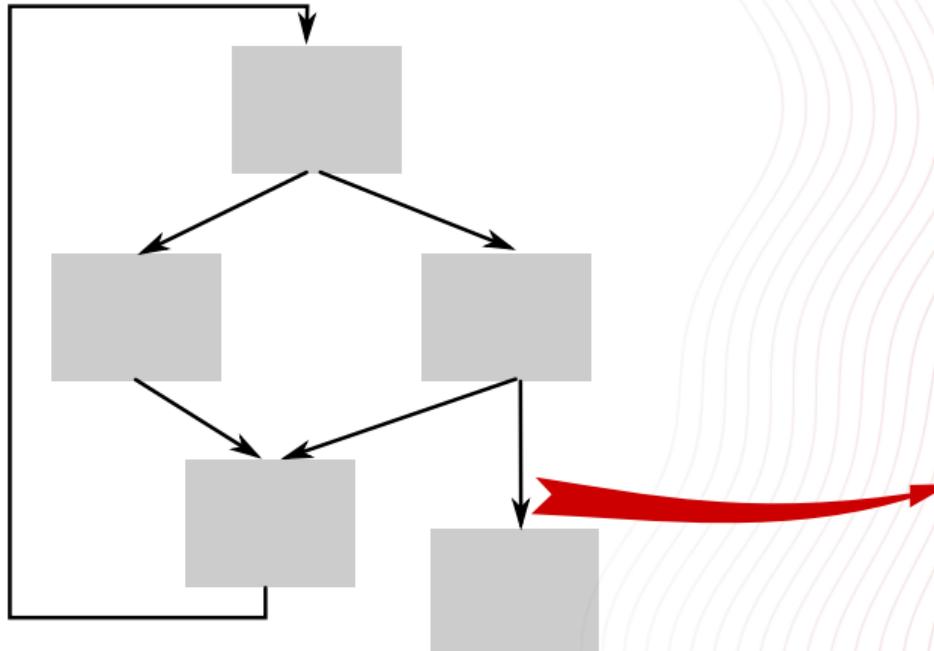
Coverage



Bitmap

1
2
1
.
.
.

Coverage



honggfuzz



Targets C/C++ ...

Coverage Guided

Like AFL but more Experimental

```
/bin/bash
-----[ 0 days 00 hrs 14 mins 00 secs ]-----/ honggfuzz 1.3 /-
Iterations : 398,052 [398.05k]
Mode : Feedback Driven Mode (2/2)
Target : './httpd/httpd -X -f /home/jagger/fuzzer/apache/dist/conf/h ...
Threads : 8, CPUs: 8, CPU%: 261% (32%/CPU)
Speed : 323/sec (avg: 473)
Crashes : 90 (unique: 1, blacklist: 0, verified: 0)
Timeouts : [5 sec] 32
Corpus Size : entries: 1,147, max size: 1,048,792, input dir: 8522 files
Cov Update : 0 days 00 hrs 00 mins 05 secs ago
Coverage : edge: 17,019 pc: 410 cmp: 187,266
----- [ LOGS ] -----
Crash (dup): './SIGABRT.PC.7ffff5ef10bb.STACK.18819c8652.CODE.-6.ADDR.(nil).INST
R.mov_____0x108(%rsp),%rcx.fuzz' already exists, skipping
[2018-01-18T22:21:22+0100][W][3343] arch_checkWait():308 Persistent mode: PID 21
623 exited with status: SIGNALLED, signal: 6 (Aborted)
Persistent mode: Launched new persistent PID: 24520
Crash (dup): './SIGABRT.PC.7ffff5ef10bb.STACK.18819c8652.CODE.-6.ADDR.(nil).INST
R.mov_____0x108(%rsp),%rcx.fuzz' already exists, skipping
[2018-01-18T22:21:23+0100][W][3346] arch_checkWait():308 Persistent mode: PID 18
231 exited with status: SIGNALLED, signal: 6 (Aborted)
Persistent mode: Launched new persistent PID: 25094
Size:296441 (i,b,hw,edge,ip,cmp): 0/0/0/0/0/1, Tot:0/0/0/17019/410/187266
```

go-fuzz



Targets Golang

Coverage Guided

Found many bugs in StdLib

go-fuzz



```
package png
import "bytes"
import "image/png"

func Fuzz(data []byte) int {
    png.Decode(bytes.NewReader(data))
    return 0
}
```

Hypothesis

Targets Python (& Java)

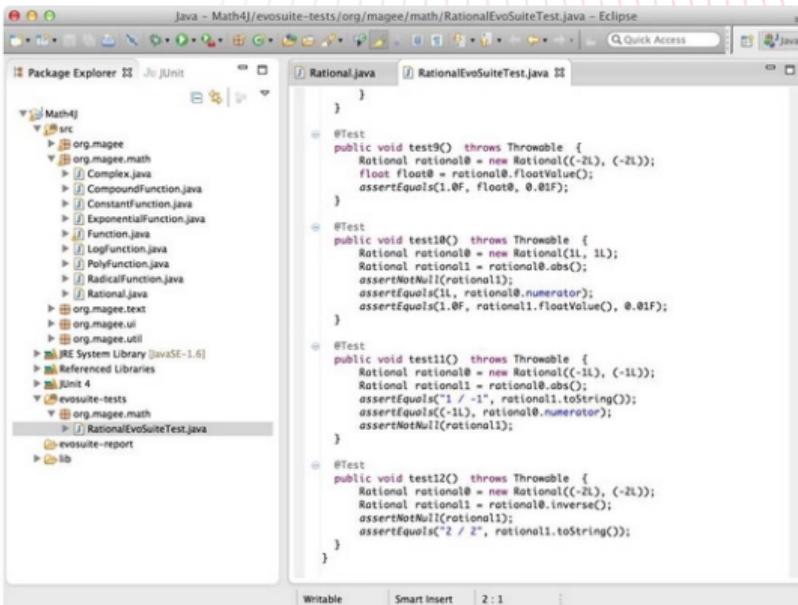
Not Coverage Guided

Property Based Testing

Hypothesis

```
1 | from hypothesis import given
2 | from hypothesis.strategies import text
3 |
4 | @given(text())
5 | def test_decode_inverts_encode(s):
6 |     assert decode(encode(s)) == s
```

Targets Java
Coverage Guided
Generates JUnit Tests



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer View:** Shows the project structure under "Math4J". The "src" folder contains packages like org.magee, org.magee.math, and org.magee.util. The "evosuite-tests" folder contains the file "RationalEvoSuiteTest.java".
- Rational.java View:** Shows the implementation of the Rational class.
- RationalEvoSuiteTest.java View:** Shows the generated JUnit test cases for the Rational class. The code includes annotations for @Test and @Test methods, and assertions like assertEquals and assertNotNull.

```
Java - Math4J/evosuite-tests/org/magee/math/RationalEvoSuiteTest.java - Eclipse
-----
package org.magee.math;

import org.junit.Test;
import static org.junit.Assert.*;

public class RationalEvoSuiteTest {
    @Test
    public void test9() throws Throwable {
        Rational rational0 = new Rational(-2L, -2L);
        float float0 = rational0.floatValue();
        assertEquals(1.0F, float0, 0.01F);
    }

    @Test
    public void test10() throws Throwable {
        Rational rational0 = new Rational(1L, 1L);
        Rational rational1 = rational0.abs();
        assertNotNull(rational1);
        assertEquals(1L, rational1.numerator());
        assertEquals(1.0F, rational1.floatValue(), 0.01F);
    }

    @Test
    public void test11() throws Throwable {
        Rational rational0 = new Rational(-1L, -1L);
        Rational rational1 = rational0.abs();
        assertEquals("1 / -1", rational1.toString());
        assertEquals(-1L, rational1.numerator());
        assertNotNull(rational1);
    }

    @Test
    public void test12() throws Throwable {
        Rational rational0 = new Rational(-2L, -2L);
        Rational rational1 = rational0.inverse();
        assertNotNull(rational1);
        assertEquals("2 / 2", rational1.toString());
    }
}
```

wfuzz



Targets Web Servers

Not Coverage Guided

Manual Specifications :(

Targets Web Servers

Not Coverage Guided

Manual Specifications :(

```
$ wfuzz -w wordlist.txt http://testphp.com/FUZZ.php
```

Nautilus



Targets C/C++

Coverage Guided

Needs a Grammar Spec

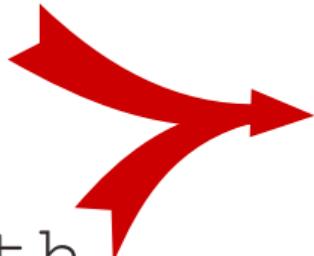
Text-based interfaces

```
h = {}
a = []
200.times do |n|
  a[0] = n
  h[a] = 0
  puts h.to_a.clone
end
```

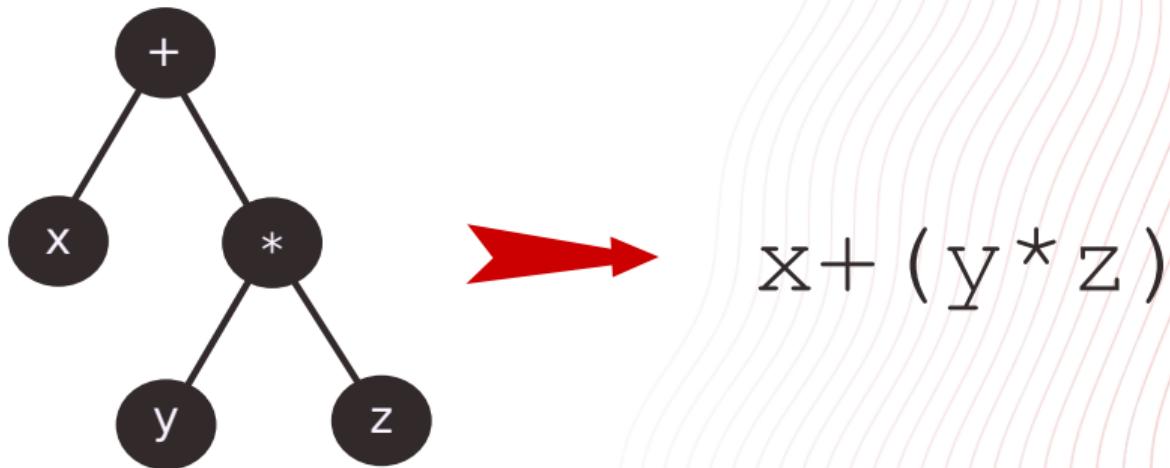
Grammar Agnostic Fuzzing

$$z = x + y$$

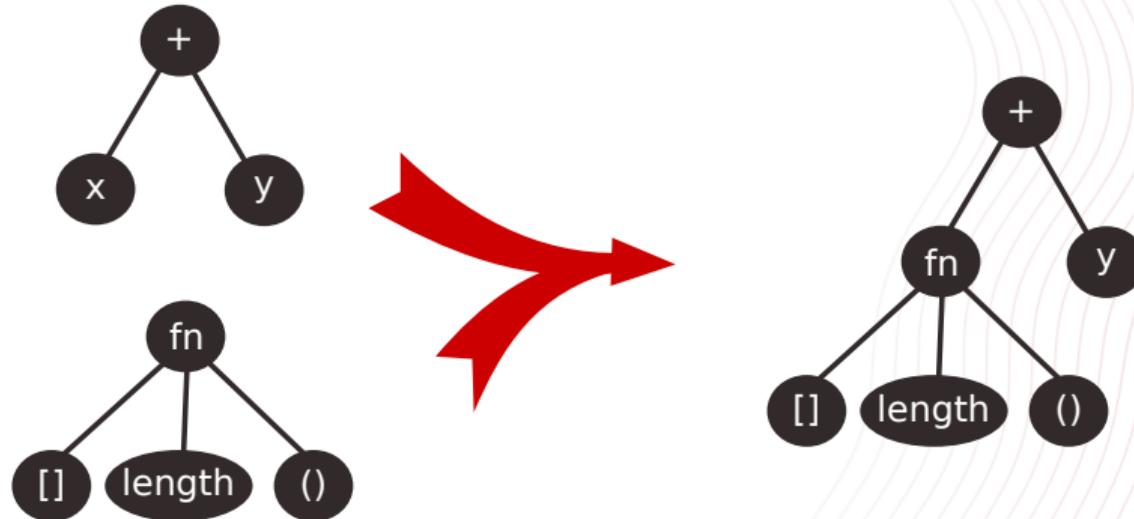
```
[ ].length
```


$$z = x + \text{ngth}$$

Generative Fuzzing

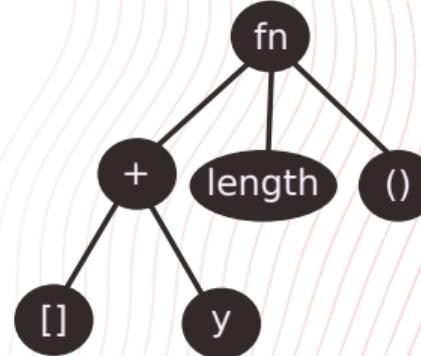
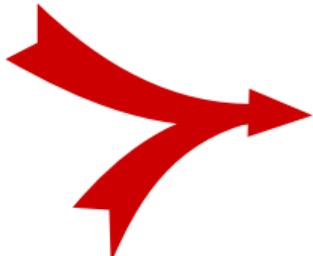
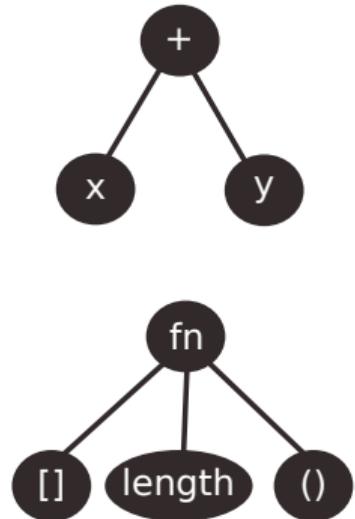


Grammar+Feedback Fuzzing



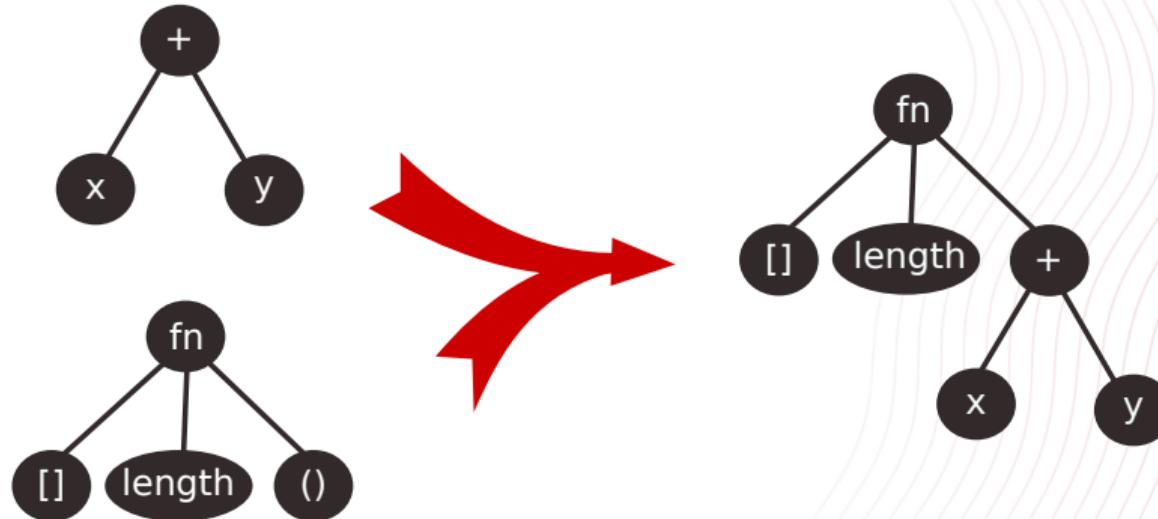
[] .length () +y

Grammar+Feedback Fuzzing



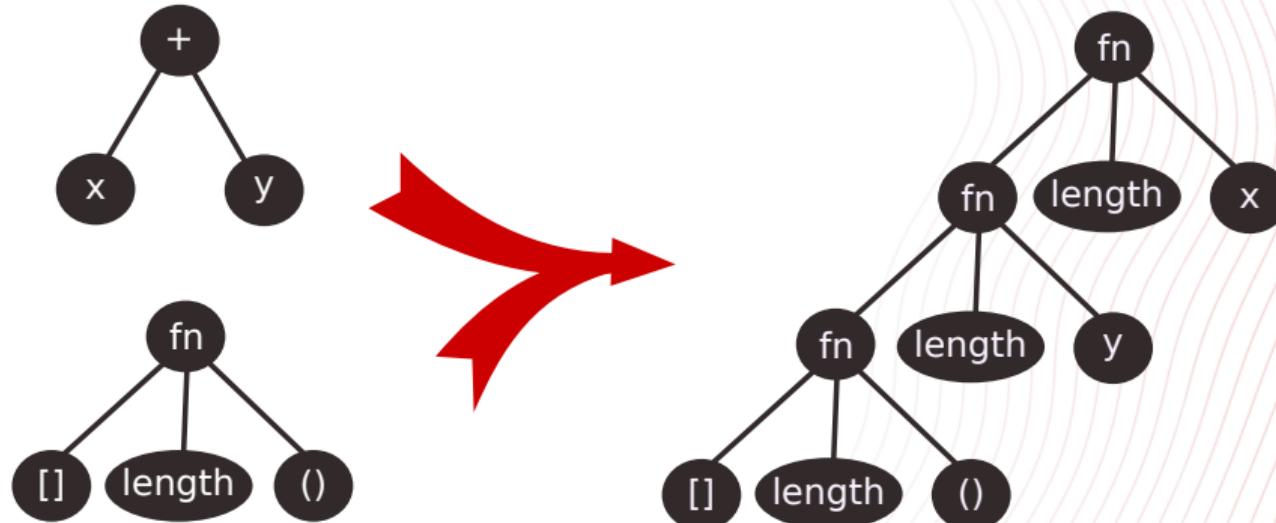
([] +y) .length()

Grammar+Feedback Fuzzing

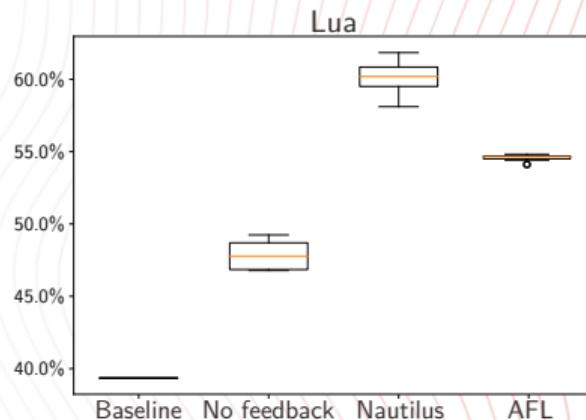
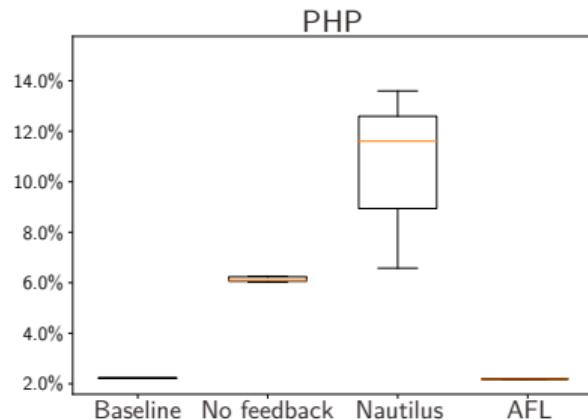
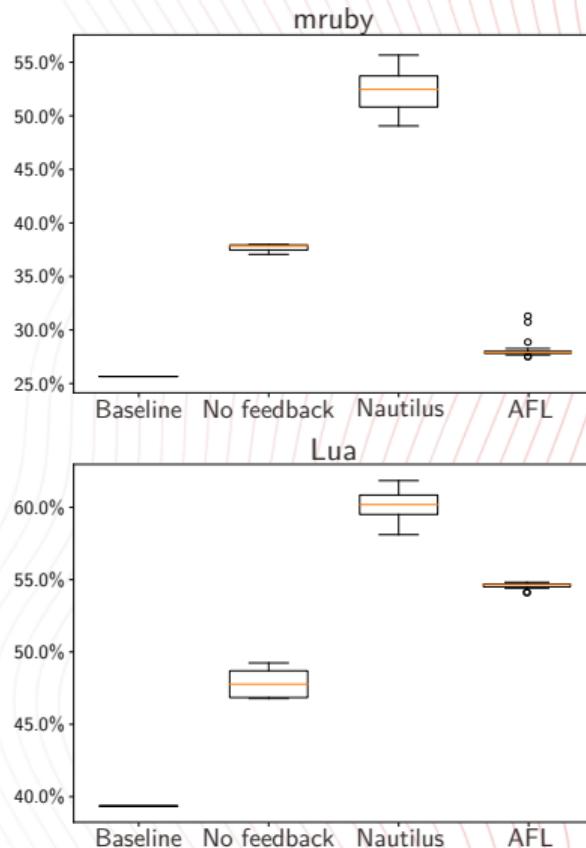
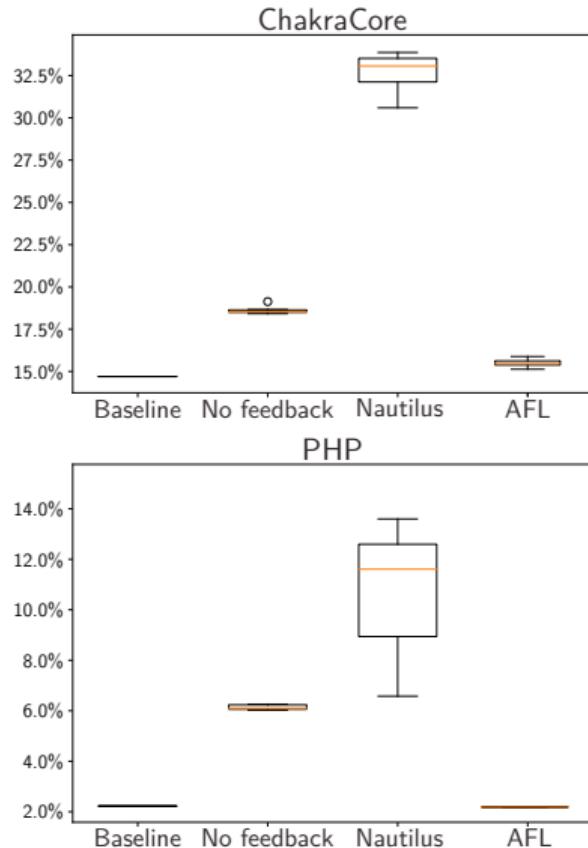


[] .length (x+y)

Grammar+Feedback Fuzzing



```
["PROGRAM", "{STATEMENT}{NEWLINE}{PROGRAM}"],  
["PROGRAM", ""],  
  
["STATEMENT", "def {VAR}.{IDENTIFIER}({ARGS}){NEWLINE}{PROGRAM} end"],  
["STATEMENT", "{VAR} = {VAR}.{IDENTIFIER}({ARGS})\\{|{ARGS}| {PROGRAM} \\|}"],  
["STATEMENT", "{VAR} = {VAL}"],  
["STATEMENT", "return {VAR}"],  
["STATEMENT", "raise {VAR}"],  
["STATEMENT", "yield {VAR}"],  
["STATEMENT", "continue {VAR}"],  
["STATEMENT", "break {VAR}"],  
["STATEMENT", "next {VAR}"],  
  
.
```



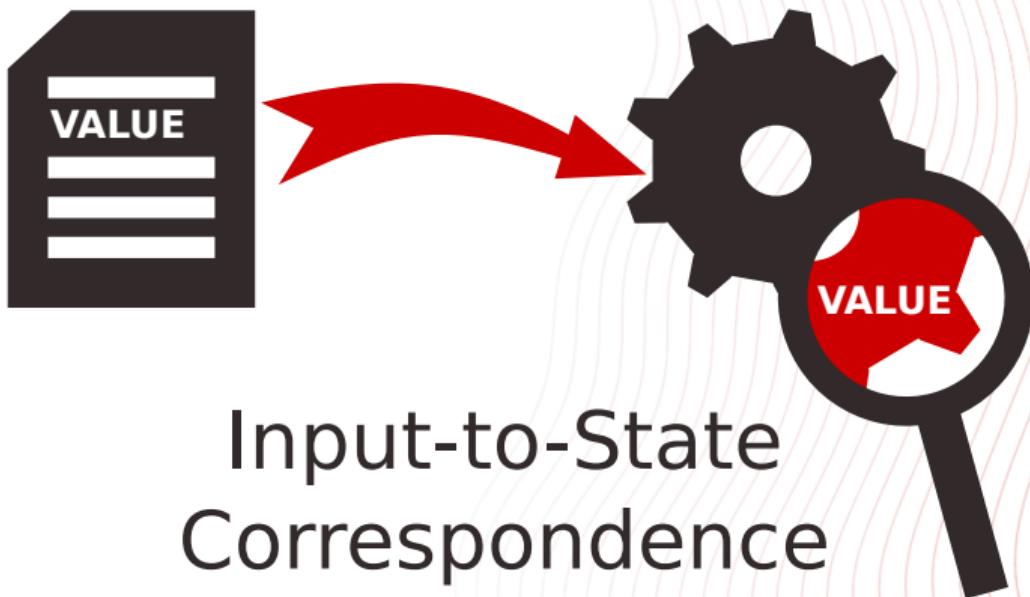
Redqueen



Binaries & Kernels & ...
Blackbox Components
Coverage Guided
Research Code

```
/* magic number */  
if( u64(input) == u64 ("MAGICHDR" ) )  
    bug(1);
```

```
/* checksum */  
if( u64(input) == hash(input[8..len]) )  
    if( input[16] == 'R' && input[17] == 'Q' )  
        bug(2);
```



Input-to-State Correspondence



TESTING
STAGE

KAFL

Target

VM



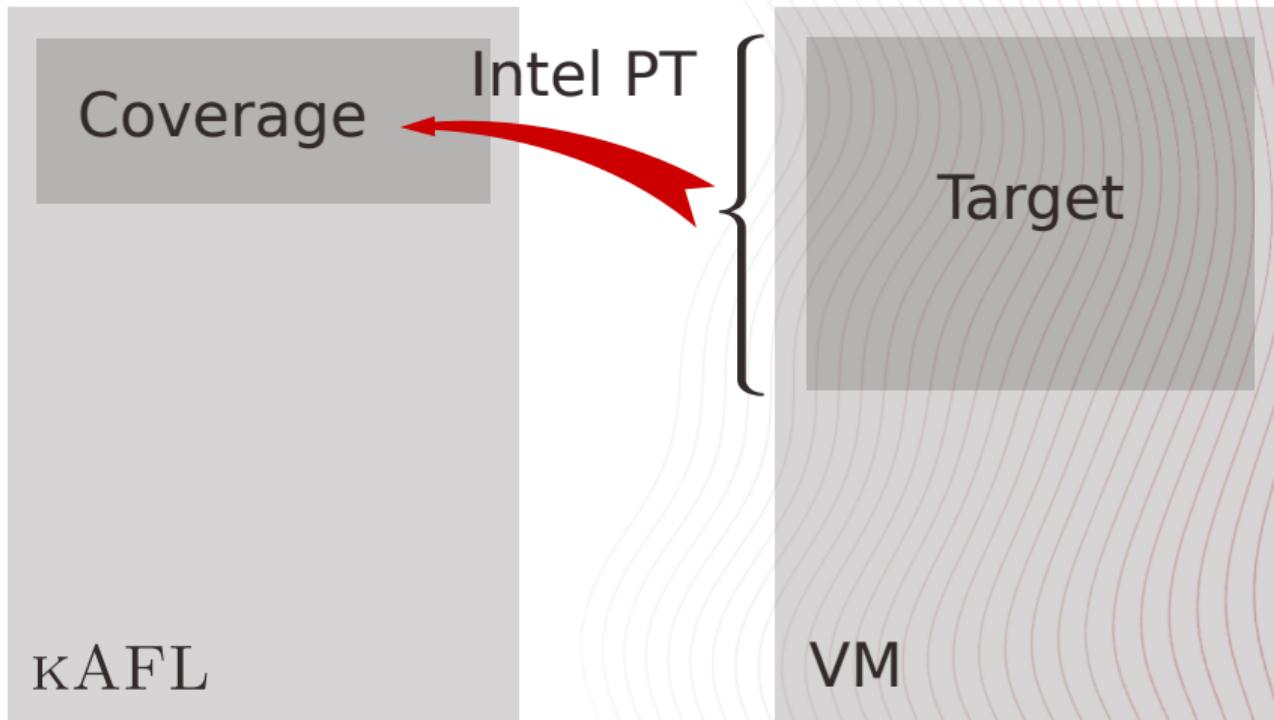
TESTING
STAGE

Coverage

KAFL

Target

VM





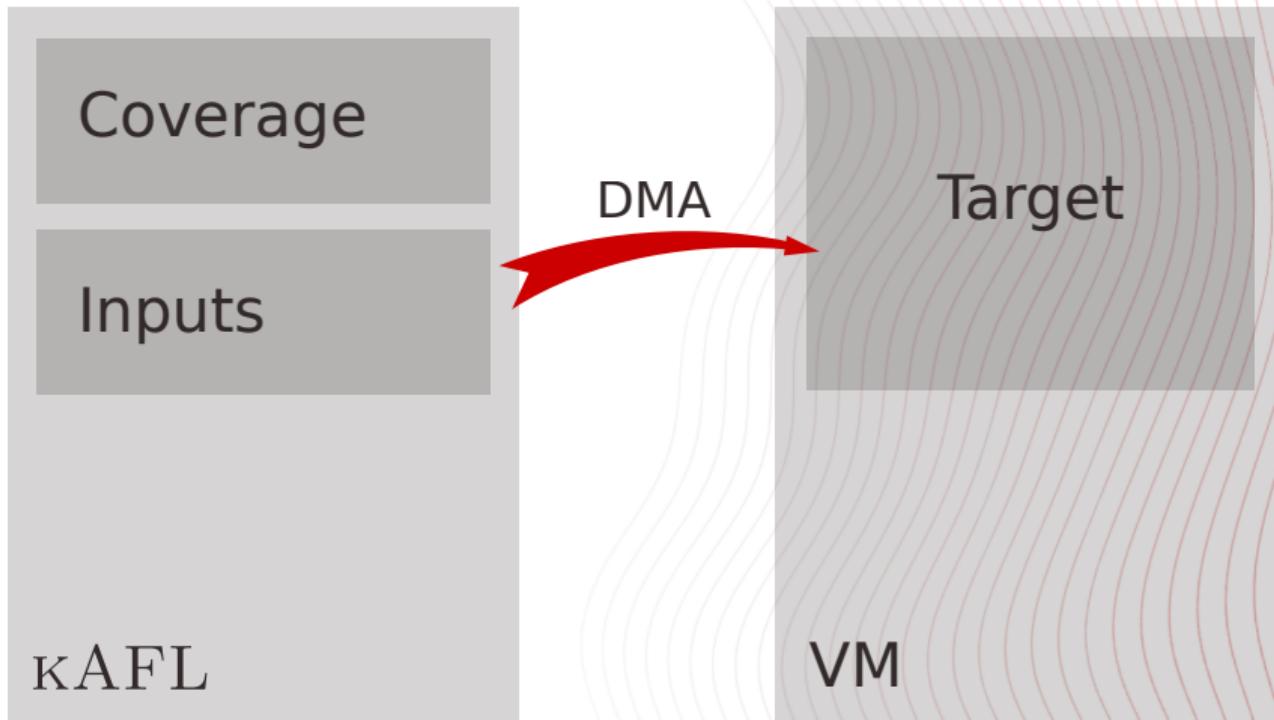
Coverage

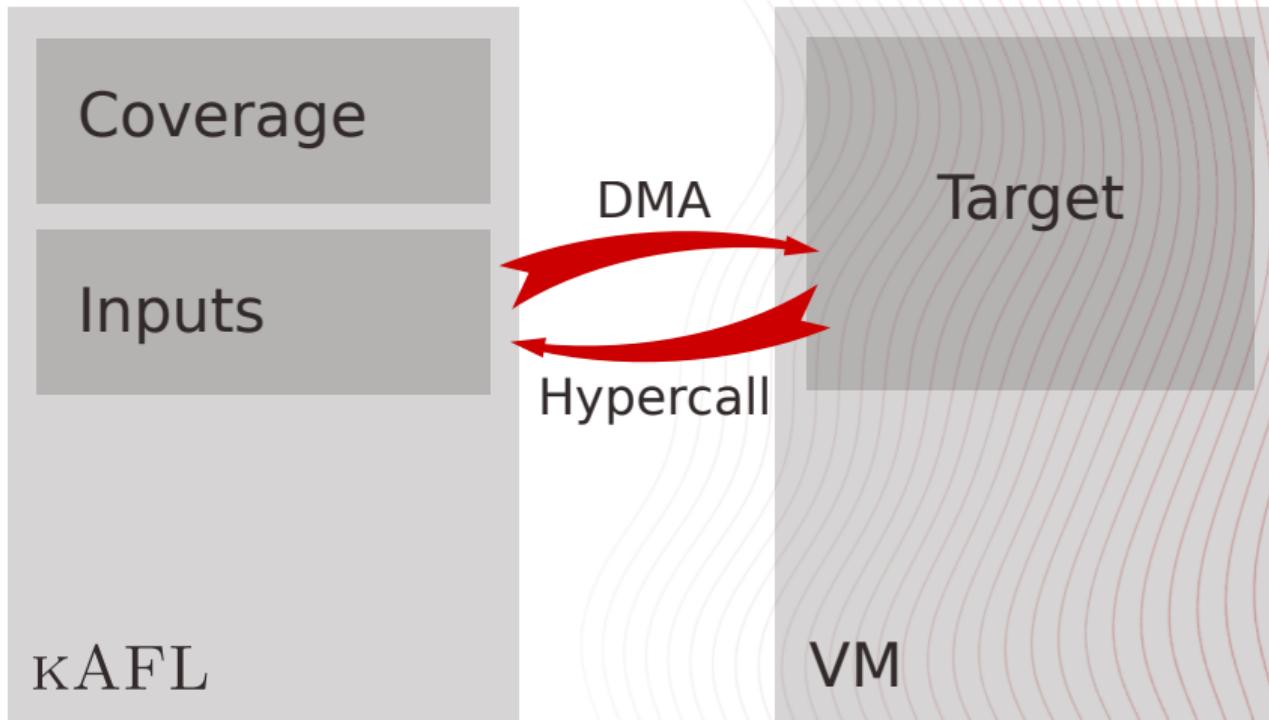
Inputs

KAFL

Target

VM







Coverage

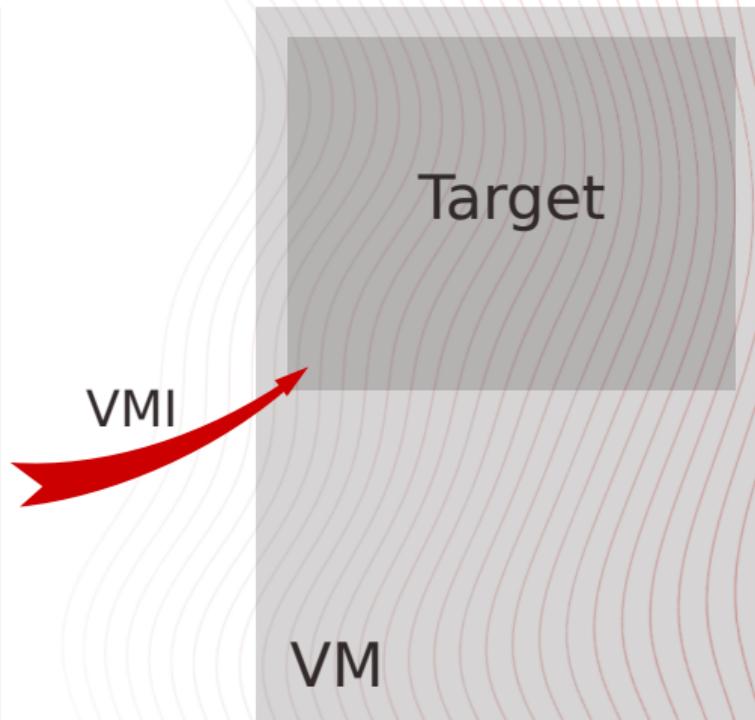
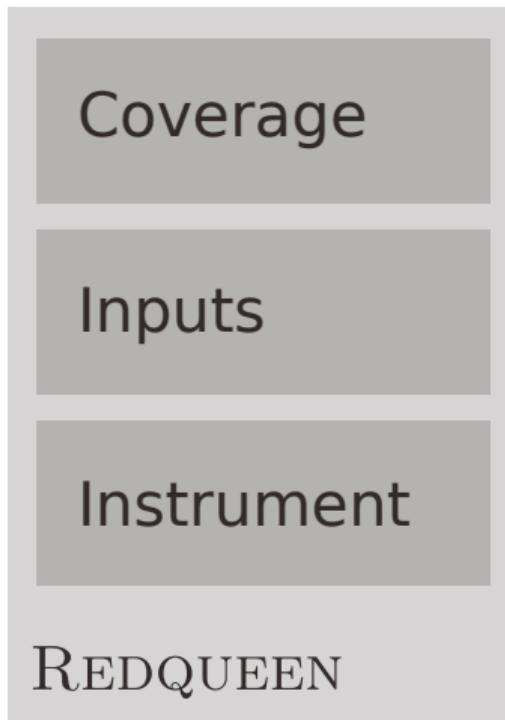
Inputs

Instrument

REDQUEEN

Target

VM



Current Research

Current Research

Root Cause Analysis

Current Research

Root Cause Analysis

Automated Exploit/Patch Generation

Current Research

Root Cause Analysis

Automated Exploit/Patch Generation

Anti Fuzzing

Current Research

Root Cause Analysis

Automated Exploit/Patch Generation

Anti Fuzzing

Extended Coverage

Current Research

Root Cause Analysis

Automated Exploit/Patch Generation

Anti Fuzzing

Extended Coverage

Taint Tracking

Current Research

Root Cause Analysis

Automated Exploit/Patch Generation

Anti Fuzzing

Extended Coverage

Taint Tracking

Symbolic Execution / SMT solvers

Questions?

Property based testing

Feedback Fuzzing

Tools

Grammar Fuzzing

Research