Learning Feedback Fuzzers: AFL and libFuzzer

Dr. Jared DeMott
CTO AND FOUNDER

@jareddemott www.vdalabs.com

Overview

Feedback fuzzing

- Research
  - Genetic algorithm
  - Constraint solver
- Integrated unit testing
- libFuzzer
- AFL

Feedback Fuzzing Research

PathCrawler (2004)

DART (2004)

KLEE (2005-2006)

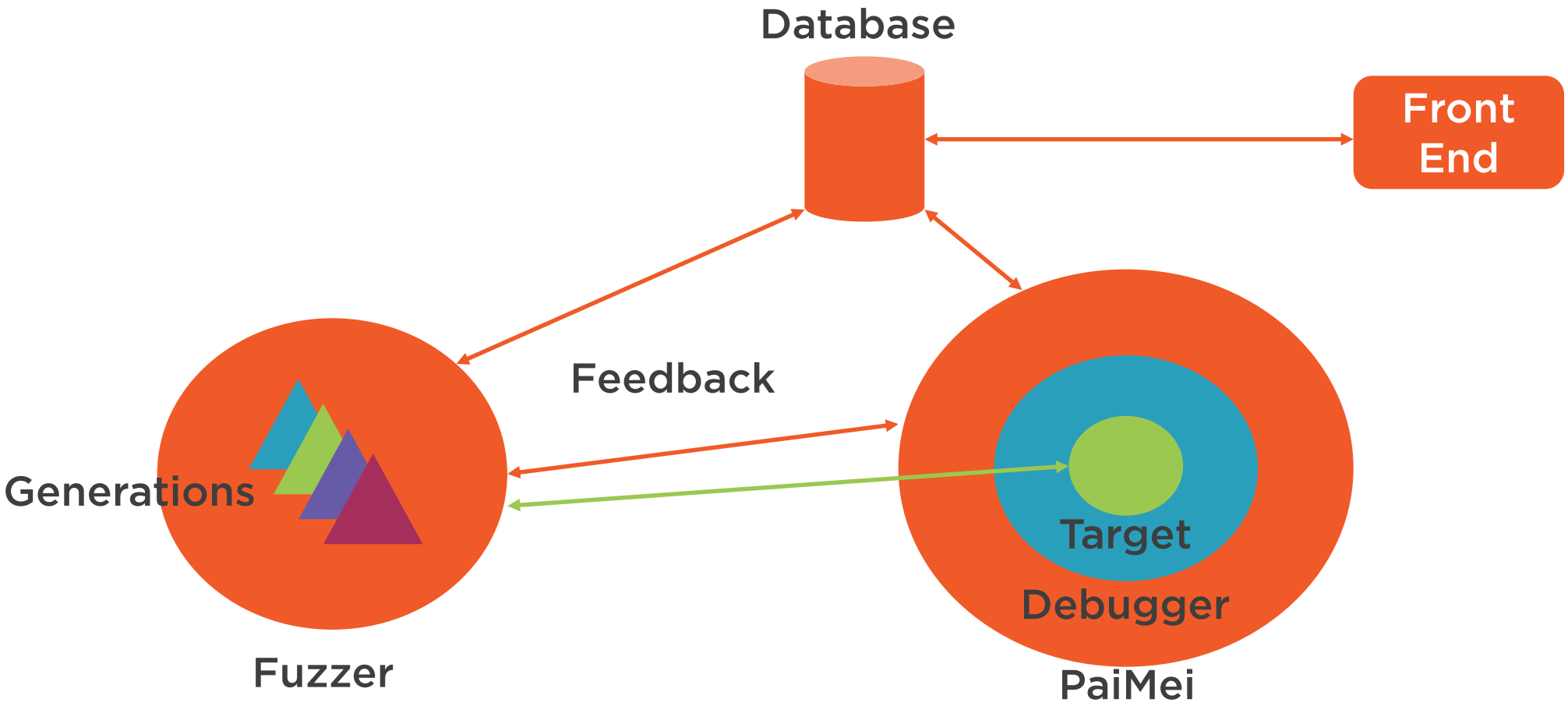
CUTE and jCUTE (2005-2006)

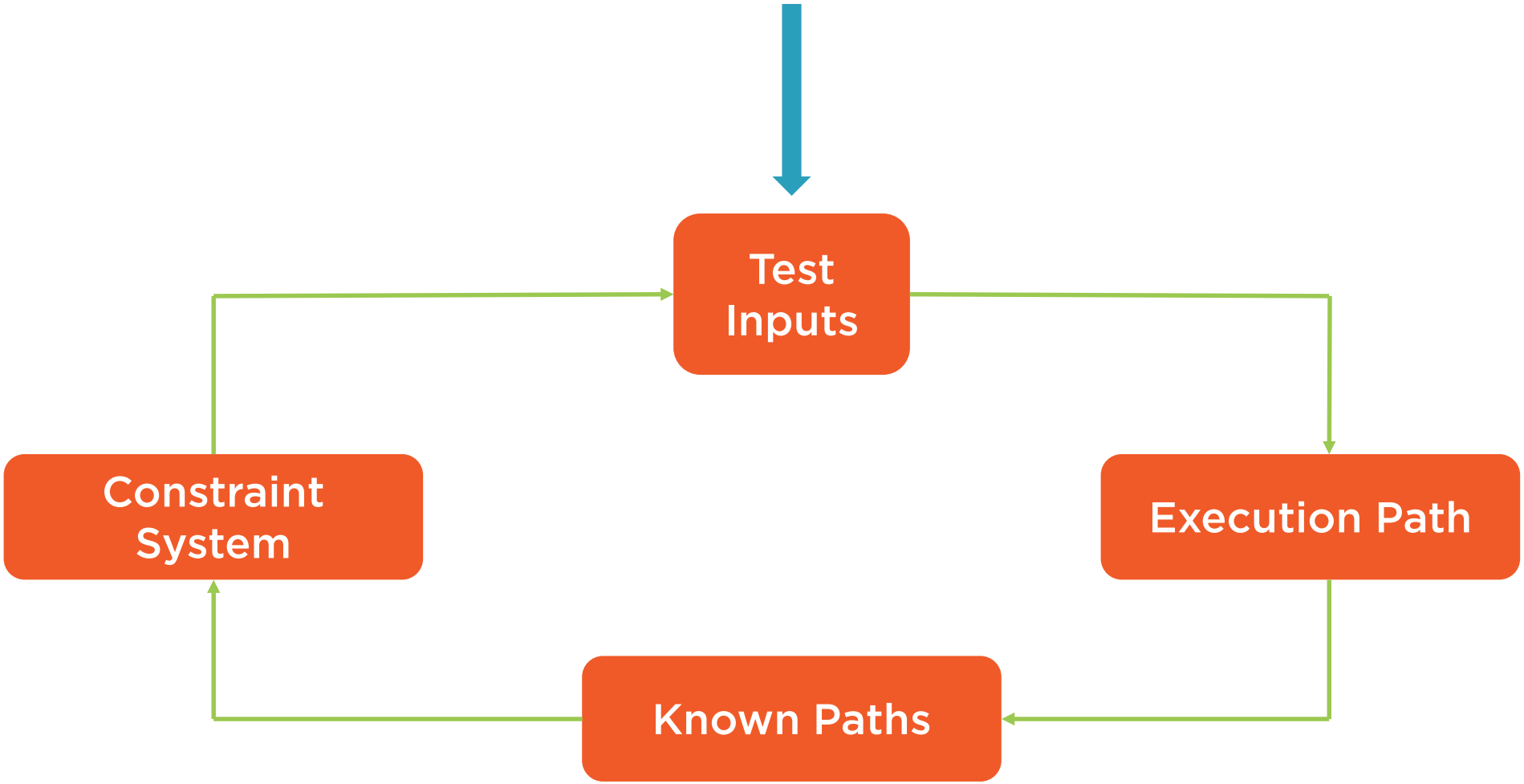
- Early tools were all Unit test tools for C
- jCUTE was Java

EFS (2007)

- First to use GA for fuzzing?

None of these were really fit for general use


Database
Front End
Feedback
Generations
Target
Debugger
Fuzzer
PaiMei



Basic White-box Fuzzing Methodology

Concolic Testing

Symbolic execution + constraint solving + dynamic testing == Concolic

- *A combination of whitebox and blackbox*
  - Reports on real issues like BB
- Finds more issues like WB
  - Increased accuracy
    - Can find the corner cases
  - Decreased runtime
    - Less need for continued test repeats or irrelevant tests

```
void f(int x, int y) {
    int z = 2 * y;
    if (x == 100000) {
        if (x < z) {
            assert(0); // bug!
        }
    }
}
```

Randomly reaching this code would be unlikely
A fuzzing sample that was close would be required
x = 100000, y = 50001

Limitations

Path Explosion

Program exhibits nondeterministic behavior

- May follow a different path than the intended one
- This can lead to non-termination of the search and poor coverage

Limitations

Even in a deterministic - issues

- Imprecise symbolic representations
  - Pointers, floating point numbers, etc.
- Incomplete theorem proving
- Failure to search most fruitful portion of a large or infinite path tree

Programs which thoroughly mix the state of their variables

- "if (md5_hash(input) == 0xdeadbeef)"

SAGE

Operates on native binaries

Found past “hard bugs”

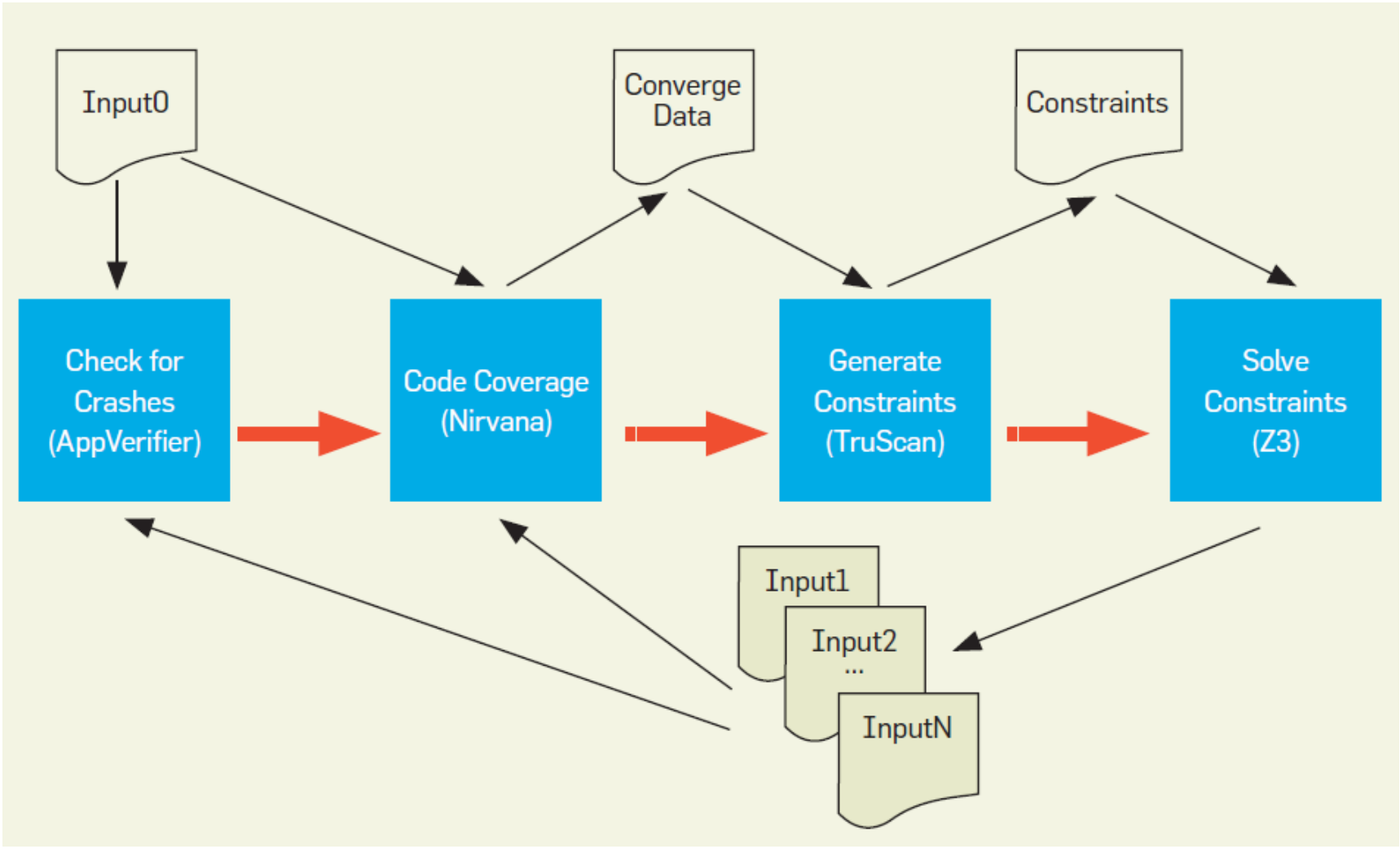
- ANI-format bug that blackbox and whitebox missed
- MS07-017

Found 1/3 of all file fuzzing bugs during the security testing of Windows 7

- SAGE is typically run last, meaning those bugs were missed by other approaches

Internal only tool

- Except for Springfield


Input0
Converge Data
Constraints
Check for Crashes (AppVerifier)
Code Coverage (Nirvana)
Generate Constraints (TruScan)
Solve Constraints (Z3)
Input1
Input2
...
InputN

Automatic Unit Tests

Pex

- Generates test suites with high code coverage using automated white box analysis
  - Works for .NET code
- Moles
  - Simulates interfaces, such as the file system or a database, so that unit testing can be conducted

Automatic Unit Tests

IntelliTest

- Explores.NET code to generate a suite of unit tests
  - For every statement, a test input is generated that will execute it
- https://msdn.microsoft.com/en-us/library/dn823749.aspx
- C# / 32 bit

libFuzzer

In-process, coverage-guided, evolutionary fuzzing engine

Linked with the library under test

- Feeds fuzzed inputs to the library via a specific fuzzing entrypoint
- Tracks which areas of the code are reached, and generates mutations on the corpus of input data in order to maximize the code coverage
- Code coverage information is provided by LLVM's SanitizerCoverage instrumentation

libFuzzer

Developer tool

- Build tools, write stub, compile, and run

Corpus

- GA search tools are much more efficient with a good set of seeds
- Samples that achieve new paths are added to this folder as they're discovered

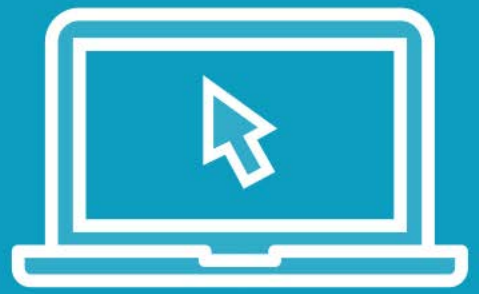
libFuzzer

You need source

Run

- http://llvm.org/docs/LibFuzzer.html
- https://github.com/google/fuzzer-test-suite/blob/master/tutorial/libFuzzerTutorial.md

Demo



AFL fuzzer

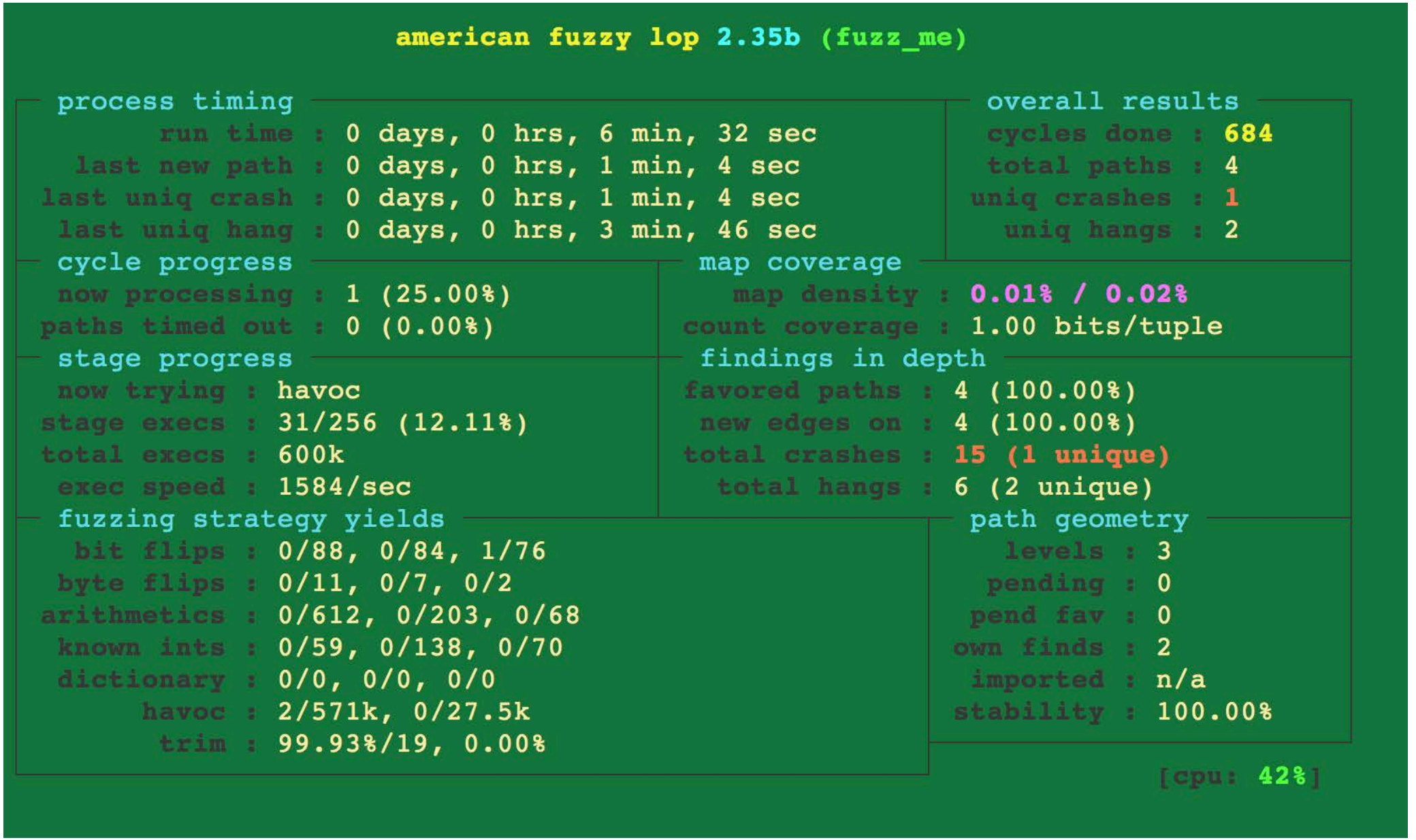
American Fuzzy Lop

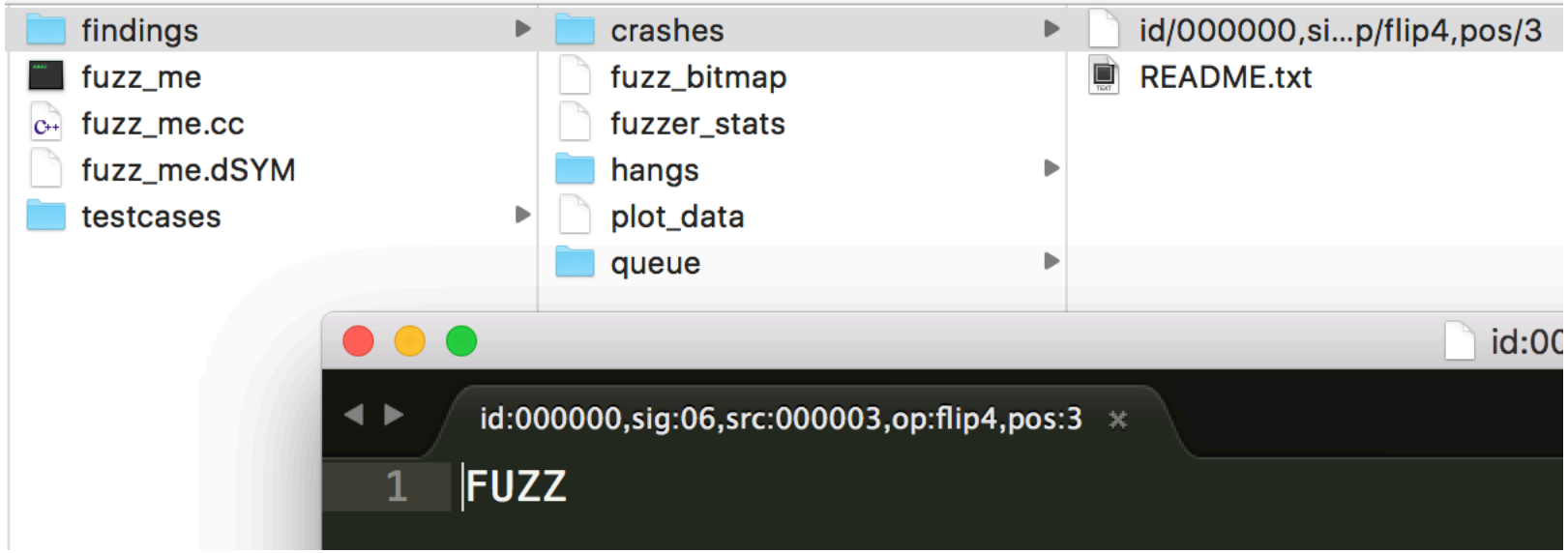
2014 released

- Compile-time instrumentation and genetic algorithms to discover clean, interesting test cases that trigger new internal states in the targeted binary
- http://lcamtuf.coredump.cx/afl/

```
bool FuzzMe(const uint8_t *Data, size_t DataSize) {
    if( Data[0] == 'F')
    {
        if(Data[1] == 'U')
        {
            if(Data[2] == 'Z')
            {
                if(Data[3] == 'Z')
                {
                    abort(); // bug
                }
            }
        }
    }

    return 0;
}

int Test(const uint8_t *Data, size_t Size) {
  FuzzMe(Data, Size);
  return 0;
}

int main(int argc, char *argv[], char *envp[]) {
    FILE *f;
    char buf[4];
    if( argc == 2) {

        f = fopen(argv[1], "r");
        if(f)
        {
            fread(buf, 1, 4, f);
            Test((uint8_t *)buf, sizeof(buf));
        }
    }
}

```

```
Jareds-MacBook-Pro-2:simple jareddemott$ ../../afl-clang fuzz_me.cc -o fuzz_me
afl-cc 2.35b by <lcamtuf@google.com>
afl-as 2.35b by <lcamtuf@google.com>
[+] Instrumented 23 locations (64-bit, non-hardened mode, ratio 100%).
Jareds-MacBook-Pro-2:simple jareddemott$ ../../afl-fuzz -i testcases/ -o findings/ -- ./fuzz_me @@
```


american fuzzy lop 2.35b (fuzz_me)
process timing
run time : 0 days, 0 hrs, 6 min, 32 sec
last new path : 0 days, 0 hrs, 1 min, 4 sec
last uniq crash : 0 days, 0 hrs, 1 min, 4 sec
last uniq hang : 0 days, 0 hrs, 3 min, 46 sec
overall results
cycles done : 684
total paths : 4
uniq crashes : 1
uniq hangs : 2
cycle progress
now processing : 1 (25.00%)
paths timed out : 0 (0.00%)
map coverage
map density : 0.01% / 0.02%
count coverage : 1.00 bits/tuple
stage progress
now trying : havoc
stage execs : 31/256 (12.11%)
total execs : 600k
exec speed : 1584/sec
findings in depth
favored paths : 4 (100.00%)
new edges on : 4 (100.00%)
total crashes : 15 (1 unique)
total hangs : 6 (2 unique)
fuzzing strategy yields
bit flips : 0/88, 0/84, 1/76
byte flips : 0/11, 0/7, 0/2
arithmetics : 0/612, 0/203, 0/68
known ints : 0/59, 0/138, 0/70
dictionary : 0/0, 0/0, 0/0
havoc : 2/571k, 0/27.5k
trim : 99.93%/19, 0.00%
path geometry
levels : 3
pending : 0
pend fav : 0
own finds : 2
imported : n/a
stability : 100.00%
[cpu: 42%]


findings
fuzz_me
fuzz_me.cc
fuzz_me.dSYM
testcases
crashes
fuzz_bitmap
fuzzer_stats
hangs
plot_data
queue
id/000000,si...p/flip4,pos/3
README.txt
id:00
id:000000,sig:06,src:000003,op:flip4,pos:3
1
FUZZ

Why It Works?

Not just the feedback

- Practical engineering
  - Uses fuzzing best practices to find new paths quickly
    - Not many knobs
  - Includes helpers like minset
  - Monitoring, coverage, distribution – all in one tool

AFL

Limitations:

- You mostly need source
  - There is a Linux/QEMU combo that allows black-box fuzzing
- Not every program is suitable

Summary

Closed

.NET

Open

- libFuzzer
- AFL
- GA works well in practice, but research still continues around constraint solving

Finish with metrics