EXERCISE 27

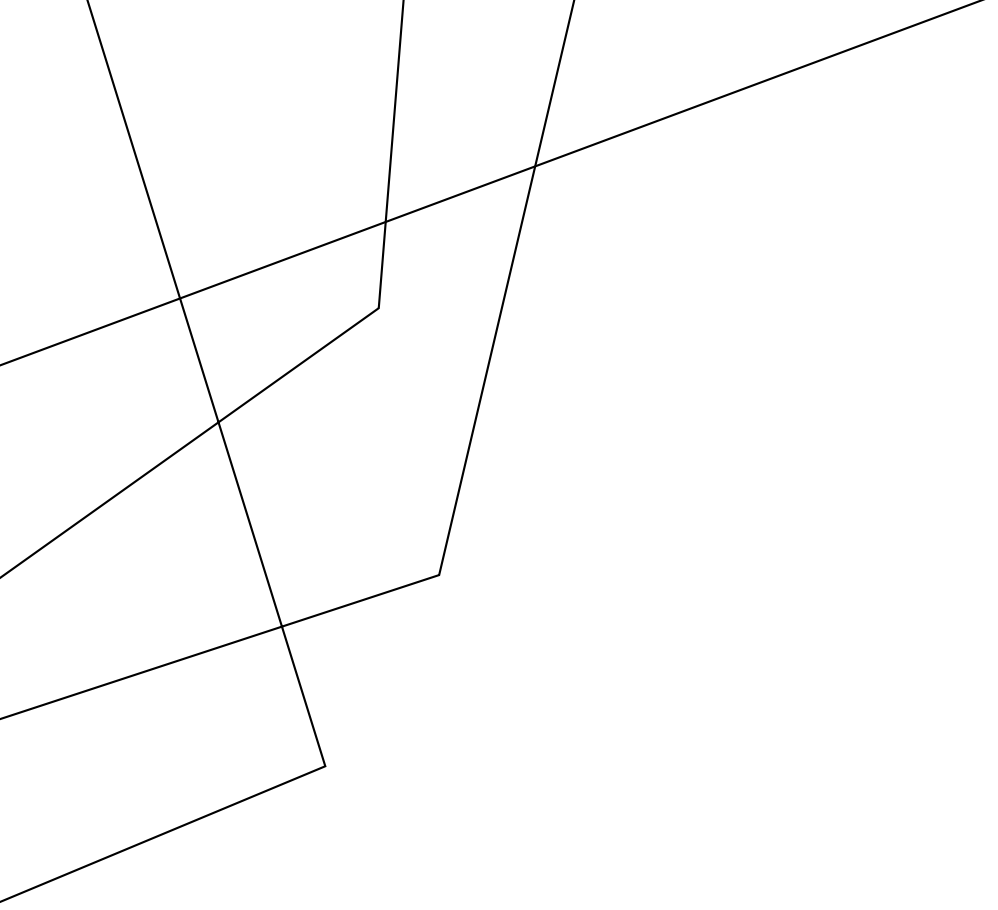
DYNAMIC ANALYSIS REVIEW

Write your name and answer the following on a piece of paper

What is the difference between unit testing and application testing?

EXERCISE 27: SOLUTION

DYNAMIC ANALYSIS REVIEW



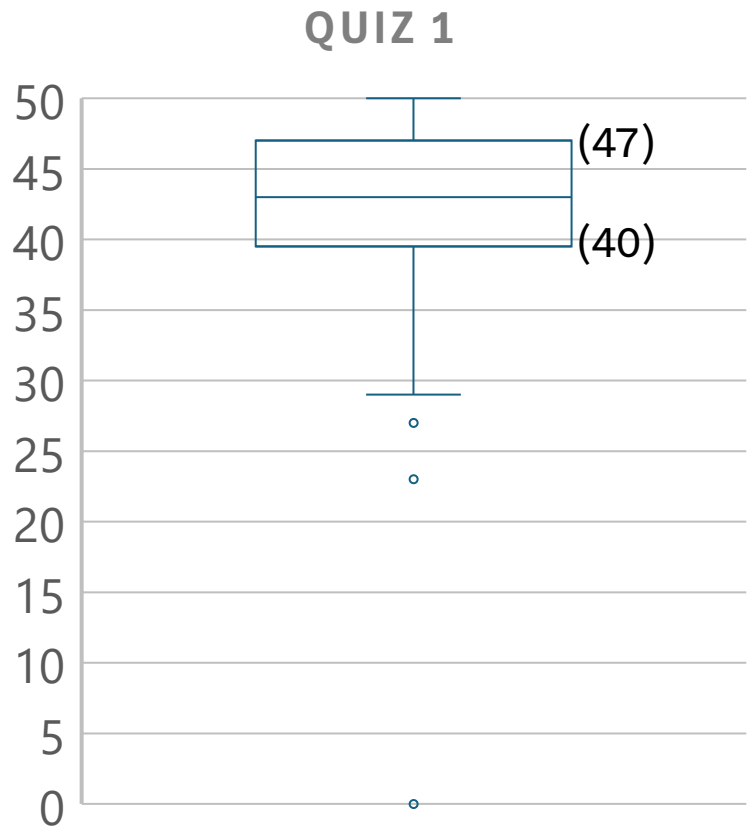
ADMINISTRIVIA AND ANNOUNCEMENTS

Quiz 1 Graded!

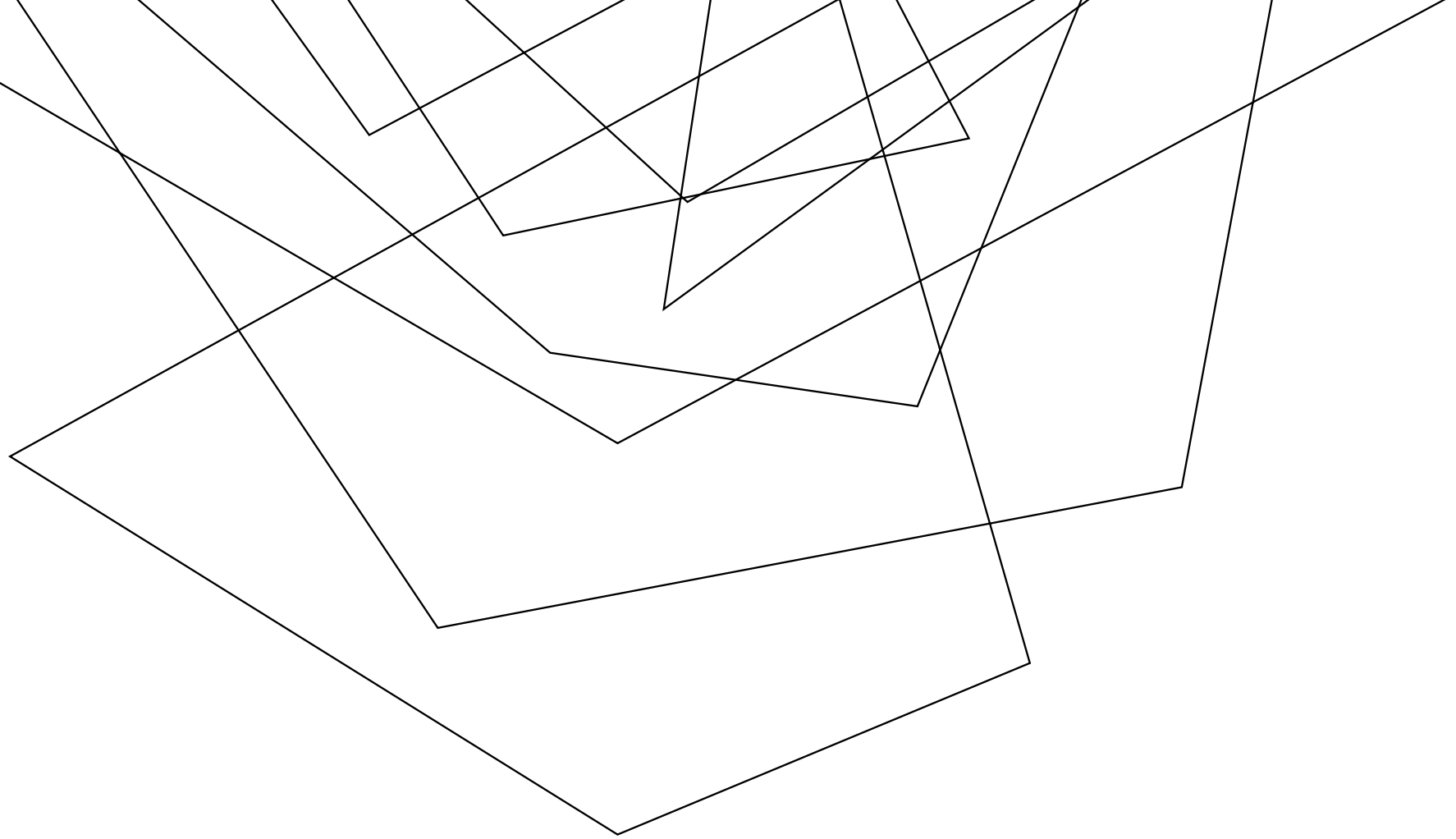
ADMINISTRIVIA AND ANNOUNCEMENTS

QUIZ 1: DATA AND THOUGHTS

ADMINISTRIVIA



Perfect	~15%
A (not perfect)	~33%
B	~27%
C	~14%
D	~4%
F	~7%



FUZZING

EECS 677: Software Security Evaluation

Drew Davidson

PREVIOUSLY: TESTING

REVIEW: DYNAMIC ANALYSIS

Usage of LLVM built-in instrumentation Analysis

Described commands to use PGO for line coverage analysis

Setup for a Custom LLVM Analysis

Described the basic infrastructure necessary to craft a custom instrumentation

THIS LESSON: FUZZING

OUTLINE / OVERVIEW

GENERATING GOOD TEST CASES

Cases that increase coverage of program behaviors

Cases that exercise unexpected behavior

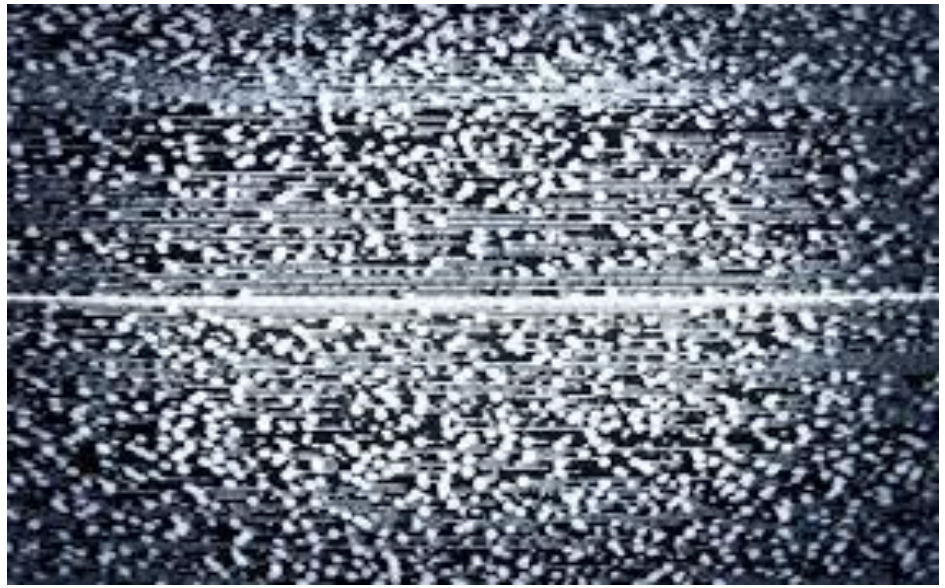
PREVIOUS STABS AT THIS TOPIC

Consider testing as an intrinsic part of the SSDLC methodology

Test-driven development

Post-hoc evaluation via coverage metrics

TODAY: JUST GUESS



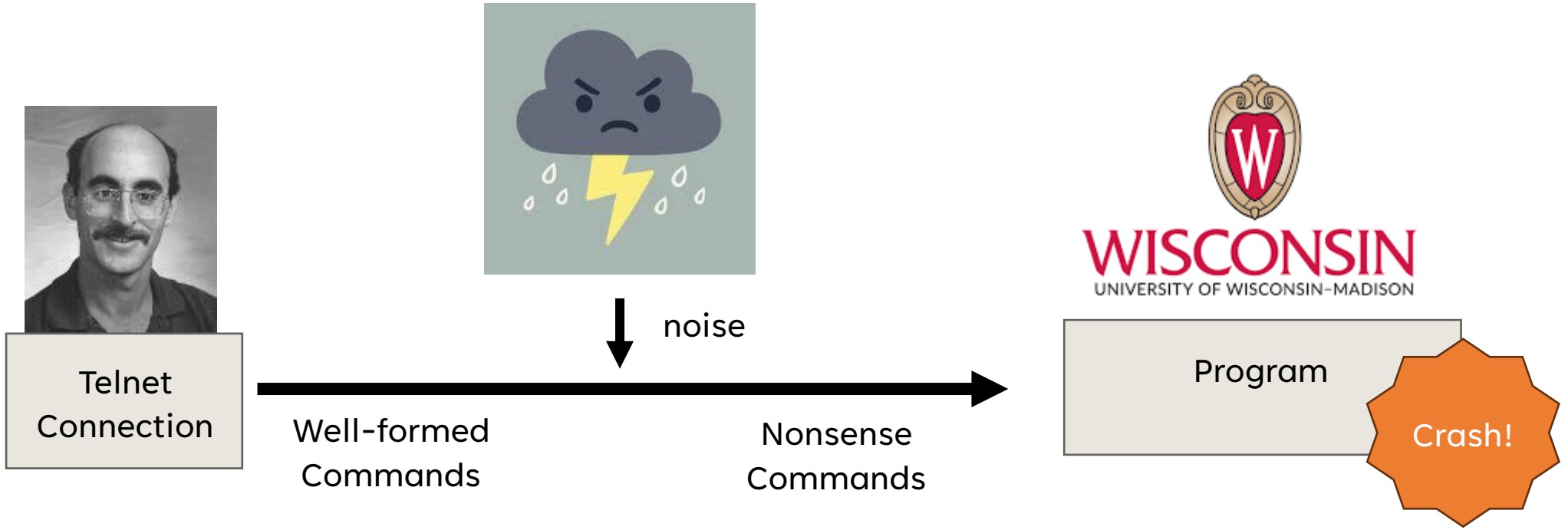
The random “fuzz” of white noise

HISTORY OF FUZZING

OUTLINE / OVERVIEW

1988: It was a Dark and Stormy Night

Professor Bart Miller attempts to work from home...



BREAKING CIRCULAR LOGIC

OUTLINE / OVERVIEW

AUTOMATED TEST CASE GENERATION RESOLVES A FUNDAMENTAL CONFLICT IN TESTING...

Tautologically impossible to predict unpredictable behavior

Apply a technique that obviated the need for expectations

GRACEFUL FAILURE

OUTLINE / OVERVIEW

Any error should be anticipated and handled by the system, with an informative error message should recovery become impossible

A KEY PRINCIPLE IN THE VALIDITY OF FUZZING

“The user should never see a seg fault”

THE SIMPLEST FUZZER

FUZZ TESTING

The Most Basic Form of Fuzzing

```
cat /dev/random | program
```

A study in the 90s basically did this, finding bugs in...
adb, as, bc, cb, col, diction, emacs, eqn, ftp, indent, lex, look, m4, make, nroff, plot, prolog, ptx, refer!, spell, style, tsort, uniq, vgrind, vi

AN ACTUAL TOOL!

FUZZ TESTING

The Most Basic Form of Fuzzing

`zzuf`

A study in the 90s basically did this, finding bugs in...
adb, as, bc, cb, col, diction, emacs, eqn, ftp, indent, lex, look, m4, make, nroff, plot, prolog, ptx, refer!, spell, style, tsort, uniq, vgrind, vi

EXPLORING UNEXPECTED BEHAVIOR

FUZZING

Random input is surprisingly effective

Numerous bugs found in practice via fuzzing...

Busybox utilities

Windows bugs

Linux Kernel bugs

Benefits of fuzzing

Very easy to run

Instant results

Highly scalable

PRIORITIZING INPUT

FUZZING

The challenge of fuzzers is (usually) getting past the first validation check

```
if (!sane_input()){
  exit 1;
}
//The rest of the program
```

SIMPLE TESTING STRATEGY

FUZZING

Consider "Interesting" Input

Values close to the maximum, minimum, middle, etc

Case Study: Card reader input: [Frisby et al., 2012]

MUTATION-BASED FUZZERS

Explore deviations from known input

Example mutations:

Binary input

– Bit flips

- Byte flips
- Change random bytes
- Insert random byte chunks
- Delete random byte chunks
- Set randomly chosen byte chunks to interesting values e.g. INT_MAX, INT_MIN, 0, 1, -1, ... §

Text input

- Insert random symbols or keywords from a dictionary

BLACK-BOX FUZZING

THROW RANDOM INPUT AT THE APPLICATION INTERFACE

REPRESENTATIVE TOOL: ZZUF

BLACK-BOX FUZZING

THE "MULTIPURPOSE FUZZER"

zzuf cat /dev/zero | hd -vn 32

zzuf -r 0.05 hd -vn 32 /dev/zero

WHITE-BOX FUZZING

FUZZING

THROW RANDOM INPUT AT THE UNIT INTERFACE

REPRESENTATIVE TOOL: AFL

FUZZING

AFL (American Fuzzy Lop)

Maintained by Google

State of the art

Generally considered the best, state-of-the-art fuzzer

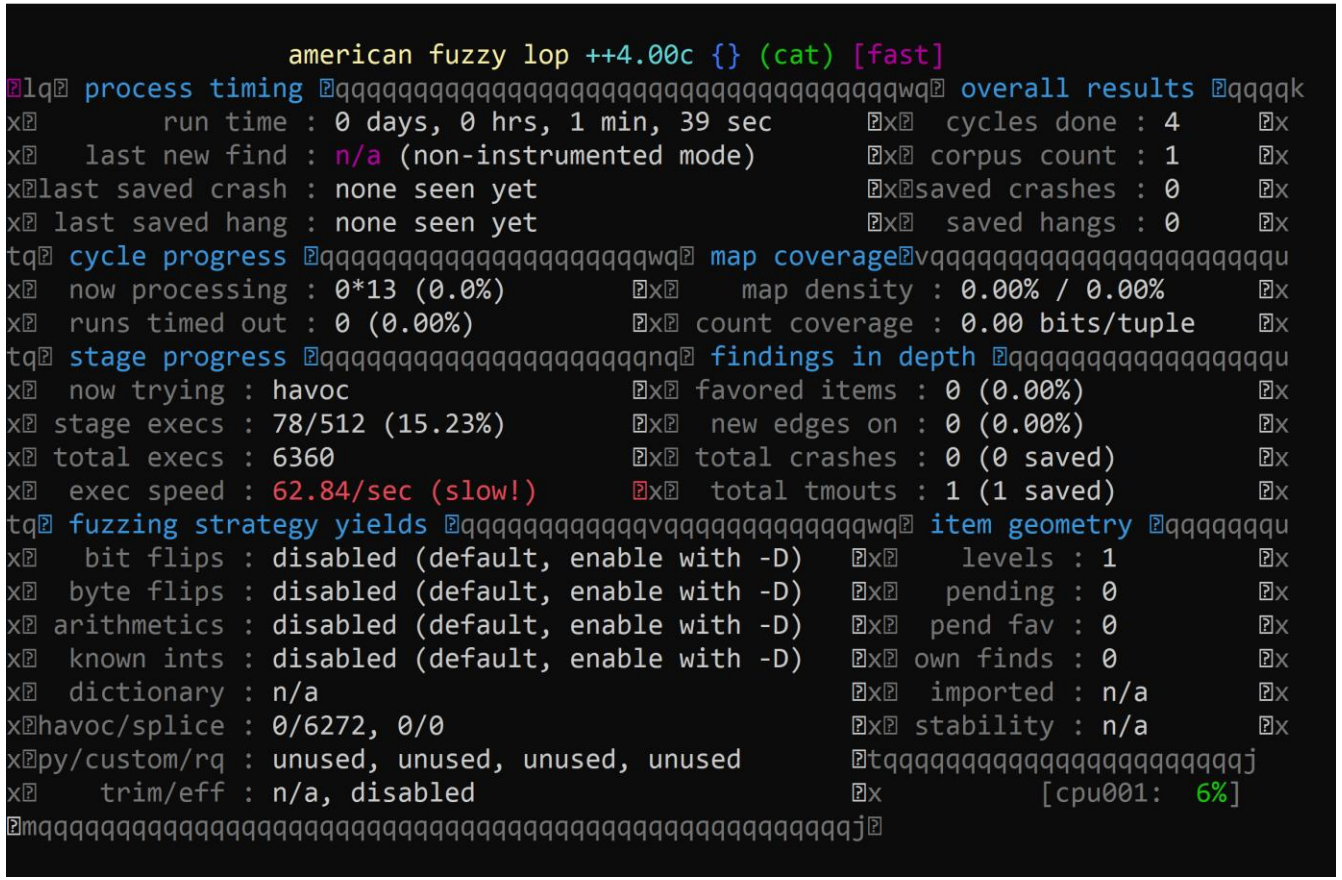
REPRESENTATIVE TOOL: AFL

FUZZING

EXAMPLE COMMAND

"TRADITIONAL FUZZING"

```
mkdir in_dir
echo "hello" > in_dir/hello
afl-fuzz -n -i in_dir -o out_dir cat
```



REPRESENTATIVE TOOL: AFL

FUZZING

INSTRUMENTATION MODE

1) Compile the program with coverage probes
2) Attempt to prioritize / mutate test cases that extend coverage

```
afl-clang++ <build command>
```

GENERATION-BASED FUZZING

Attempt to discern patterns / Alternatives in Input

Potentially with the help of some guide of guide / input grammar

FUZZING ORACLES

Beyond graceful Failure

In C/C++ there are a lot of violations of proper behavior that are invisible

“Seems fine until it’s a huge problem”

Sanitizers

UBSan – Undefined behavior sanitizer
ASan – Address sanitizer
TSan – Thread sanitizer

RESEARCH DIRECTION: “GUNKING”

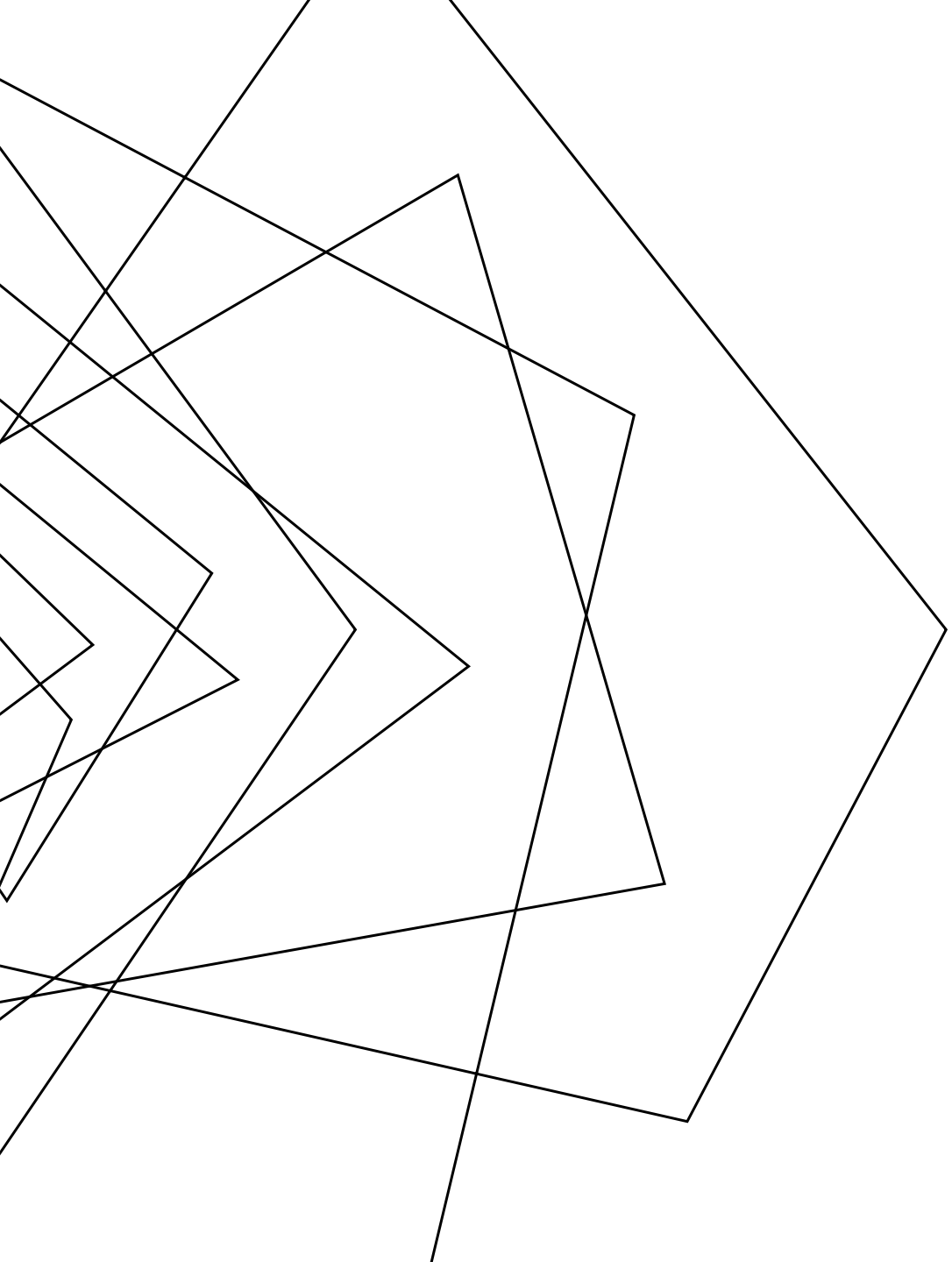
FUZZING

Fuzzing as adversarial recon

Fuzzing is so good at finding bugs that even the bad guys do it

Perhaps a program should deploy Anti-Fuzzing tech

What would that look like?



WRAP-UP

INTRODUCED THE CONCEPT AND THE "INDUSTRY STANDARD" TOOL OF FUZZING

A simple, elegant idea