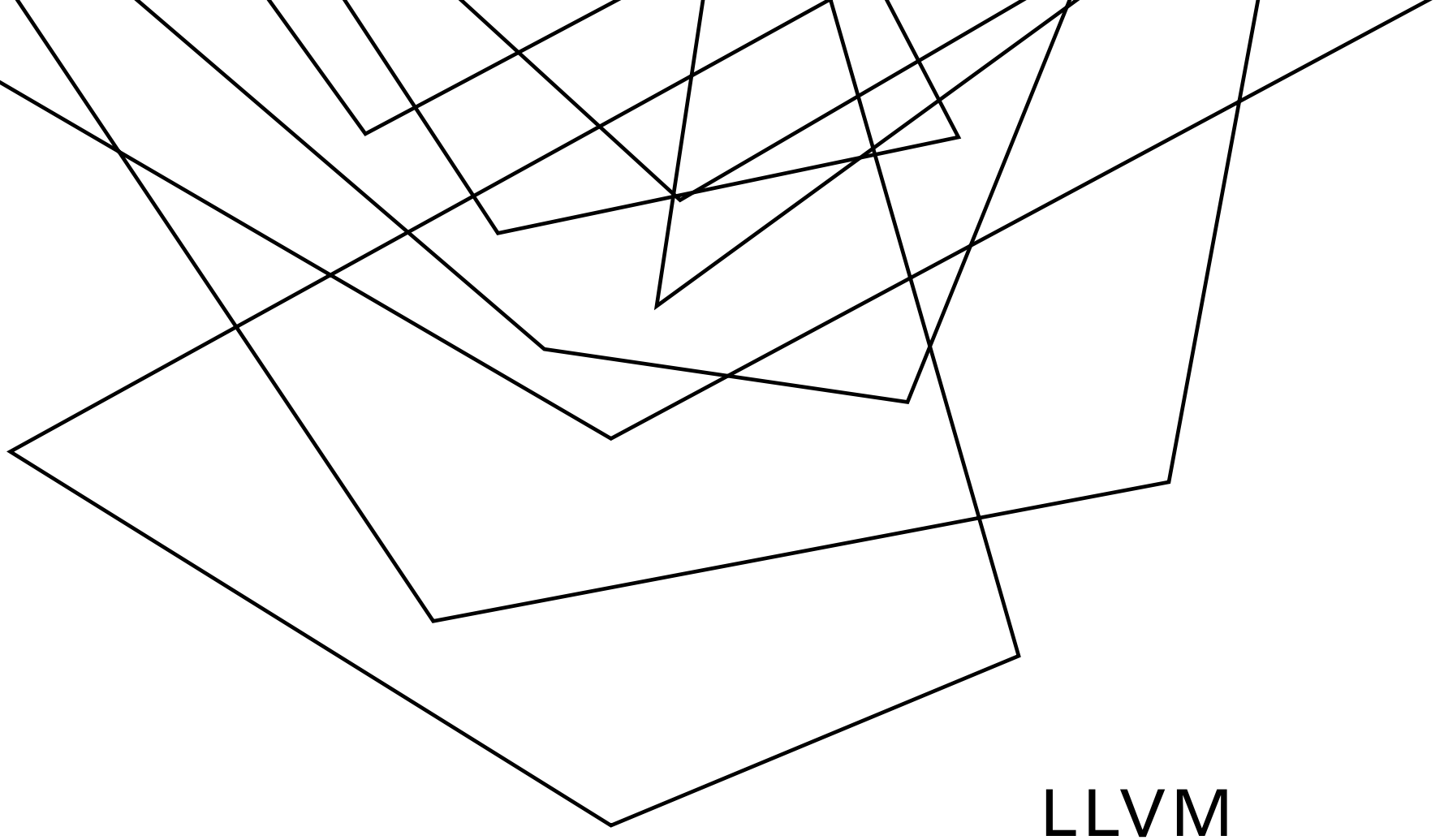
EXERCISE #27

LLVM INSTRUMENTATION REVIEW

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

How is line coverage profiling implemented in LLVM?

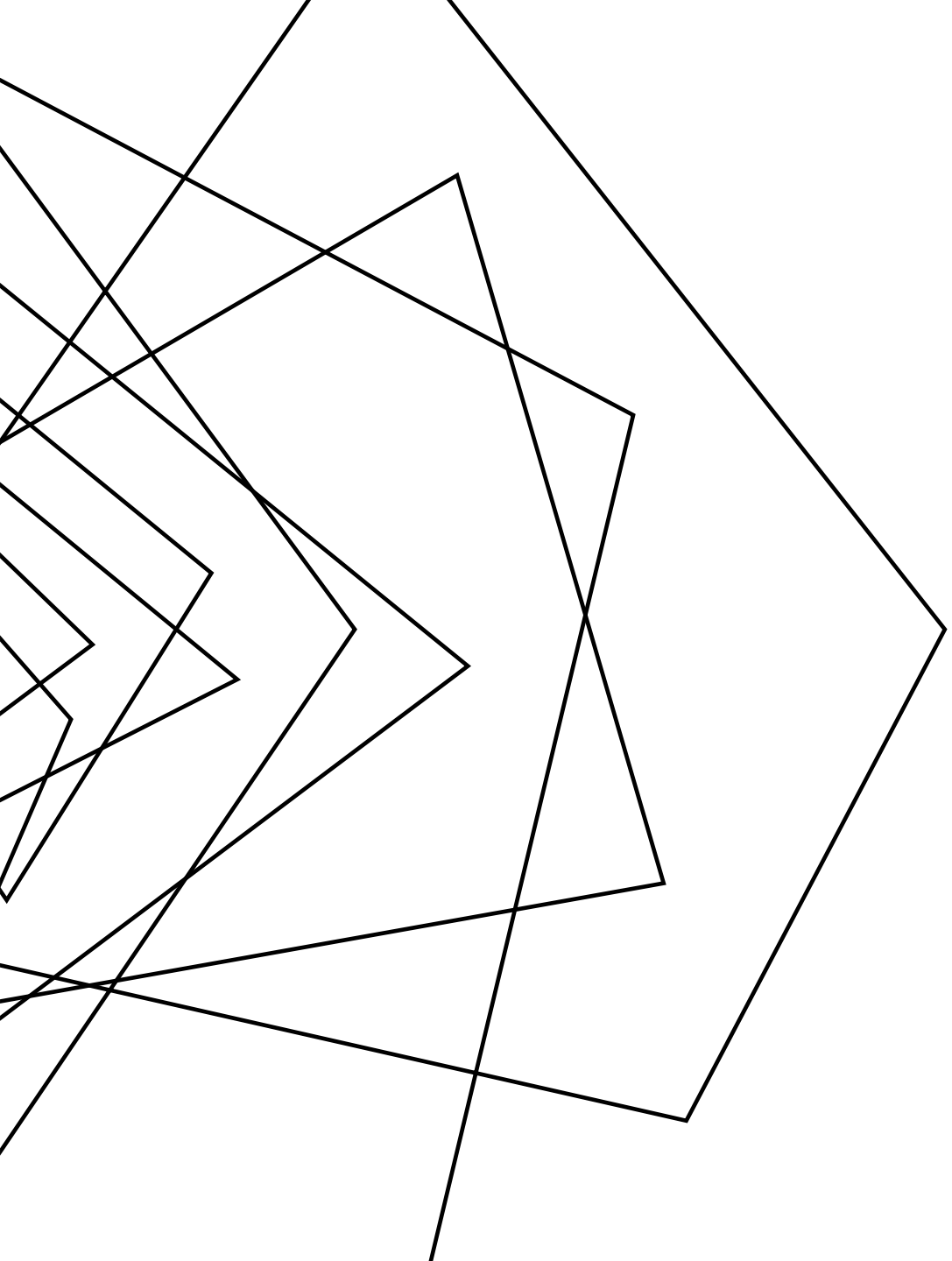
ADMINISTRIVIA AND ANNOUNCEMENTS



LLVM INSTRUMENTATION

EECS 677: Software Security Evaluation

Drew Davidson



WHERE WE’RE AT

EXPLORING PROGRAM INSTRUMENTATION

An approach to dynamic analysis

PREVIOUSLY: STATIC INSTRUMENTATION

REVIEW: LAST LECTURE

PRACTICAL TOOLS FOR PROGRAM INSTRUMENTING

GCC – Internal transformation passes
LLVM opt – IR rewriting

gcc - gnu c compiler
GCC - gnu compiler collection

USAGE OF LLVM BUILT-IN INSTRUMENTATION ANALYSIS

SETUP FOR A CUSTOM LLVM ANALYSIS

THIS LESSON: WRITING INSTRUMENTATION

OUTLINE / OVERVIEW

Practical Tools for Program Instrumenting

GCC – Internal transformation passes
LLVM opt – IR rewriting

Usage of LLVM built-in instrumentation Analysis

Providing more detail

Setup for a Custom LLVM Analysis

Presenting an example

SETUP / ASSUMPTIONS

LLVM BUILT-IN INSTRUMENTATION

THIS PORTION OF THE LECTURE USES A CLANG++ INSTALLATION.

Should work for many versions of LLVM (tested on clang++-9)
Works on clang++14 (which is installed on the cycle servers)

clang++-14

Currently at LLVM-17 stable

INSTALLATION (ON A LOCAL MACHINE)

sudo apt install clang-~~cm~~ llvm-dev

LLVM COVERAGE INSTRUMENTATION

LLVM BUILT-IN INSTRUMENTATION

GOAL: ASSESS THE COVERAGE OF A TEST SUITE

APPROACH: USE LLVM'S BUILT-IN INSTRUCTION INSTRUMENTATION

Piggyback on LLVM's PGO facilities

↑ Profile-guided optimization

1) Insert PGO probes

2) Interpret probes as coverage measurements

3) Generate a coverage report

LLVM: INSERTING PGO PROBES

LLVM BUILT-IN INSTRUMENTATION

`-fprofile-instr-generate`

Generate profile information at the source instruction level

`-fprofile-generate`

Generate profile information at the IR level

We can actually see the instrumented code live!

LLVM: INSERTING PGO PROBES

LLVM BUILT-IN INSTRUMENTATION

Let’s write a simple LLVM program, then observe the probes...

```
clang++ prog.ll –o prog-instr.ll -fprofile-generate –emit-llvm
```

LLVM COVERAGE INSTRUMENTATION

LLVM BUILT-IN INSTRUMENTATION

GOAL: ASSESS THE COVERAGE OF A TEST SUITE

APPROACH: USE LLVM'S BUILT-IN INSTRUCTION INSTRUMENTATION

Piggyback on LLVM's PGO facilities

1) Insert PGO probes

2) Interpret probes as coverage measurements

3) Generate a coverage report

LLVM: COVERAGE MAPPING

LLVM BUILT-IN INSTRUMENTATION

For understanding line coverage, we need to map changes to source code

```
clang++ prog.cc -o prog -fprofile-instr-generate -emit-llvm
-fcoverage-mapping
```

This will cause the program to output an additional coverage file in the location of the environment variable LLVM_PROFILE_FILE

```
export LLVM_PROFILE_FILE=test1.prof
```

LLVM COVERAGE INSTRUMENTATION

LLVM BUILT-IN INSTRUMENTATION

GOAL: ASSESS THE COVERAGE OF A TEST SUITE

APPROACH: USE LLVM'S BUILT-IN INSTRUCTION INSTRUMENTATION

Piggyback on LLVM's PGO facilities

1) Insert PGO probes

2) Interpret probes as coverage measurements

3) Generate a coverage report

LLVM: COVERAGE REPORT

LLVM BUILT-IN INSTRUMENTATION

The profile file is useful for a variety of things (i.e. PGO). As such, it is not (immediately) human-readable

We'll use some extra tools to generate a readable report

```
llvm-profdata merge -sparse test1.prof -o final.profdata

llvm-cov-9 show badcalc -instr-profile=final.profdata >& profile.report
```

PUTTING IT ALL TOGETHER

REVIEW: LAST LECTURE

THESE COMMANDS WORK FINE FOR 1 TEST RUN, BUT WE CARE ABOUT TEST SUITES

```
clang++ prog.ll –o prog -fprofile-instr-generate -fcoverage-mapping
export LLVM_PROFILE_FILE=test1.prof
./prog
export LLVM_PROFILE_FILE=test2.prof
./prog
llvm-profdata merge -sparse test*.prof -o final.profdata
llvm-cov-9 show badcalc -instr-profile=final.profdata >& profile.report
```

THIS LESSON: WRITING INSTRUMENTATION

OUTLINE / OVERVIEW

Practical Tools for Program Instrumenting

GCC – Internal transformation passes
LLVM opt – IR rewriting

Usage of LLVM built-in instrumentation Analysis

Providing more detail

Setup for a Custom LLVM Analysis

Presenting an example

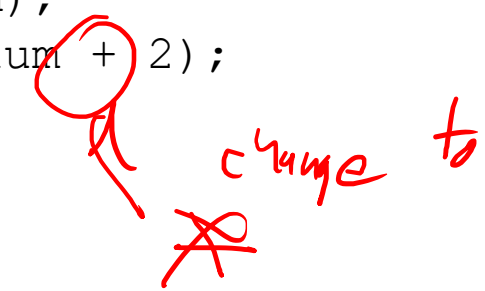
EXAMPLE: CUSTOM LLVM INSTRUMENTATION

PROGRAM INSTRUMENTATION: APPROACH

LET'S REMOVE AND ADD SOME INSTRUCTIONS!

Consider a simple "add2" program:

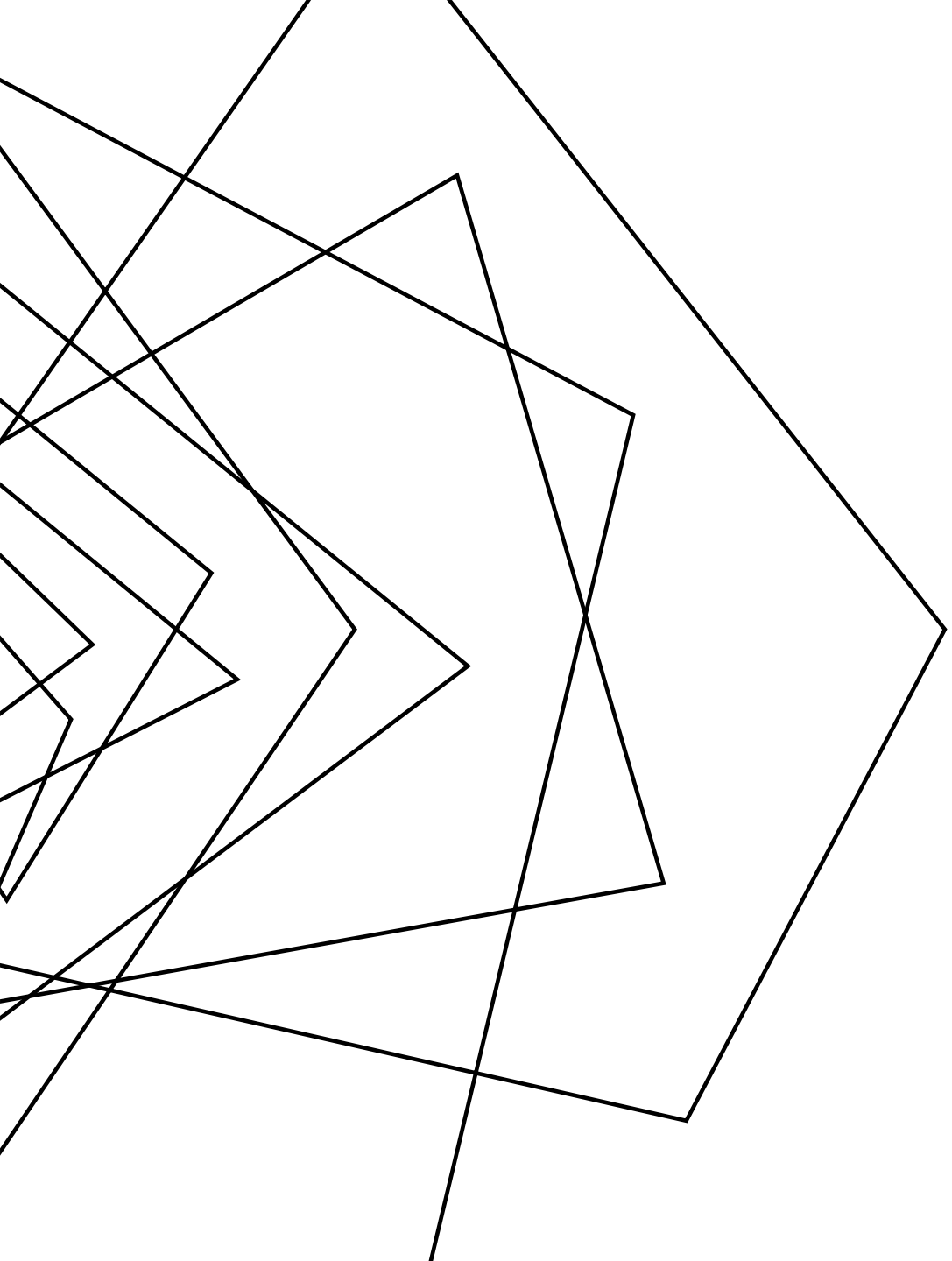
```
#include <stdio.h>
int main(int argc, const char**
argv) {
    int num;
    scanf("%i", &num);
    printf("%i\n", num + 2);
    return 0;
}
```



EXAMPLE: LLVM COVERAGE INSTRUMENTATION

PROGRAM INSTRUMENTATION: APPROACH

LET'S TAKE IT TO THE TERMINAL!



WRAP-UP

WE'VE DESCRIBED THE THEORY AND PRACTICE OF PROGRAM INSTRUMENTATION

Next time: Consider how we generate test cases