EXERCISE #3

COMPUTABILITY REVIEW

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

- Briefly describe how you might create a *sound* analysis that detects null pointer errors. Your analysis should be non-trivial (i.e. it should detect at least SOME true positives)

ANALYSIS CATEGORIES

EECS 677: Software Security Evaluation

Drew Davidson

- Exercise grades posted this weekend

ADMINISTRIVIA AND ANNOUNCEMENTS

CLASS PROGRESS

TO EXPLORE *SECURITY* ANALYSIS, WE'RE GETTING A GROUNDING IN PROGRAM ANALYSIS

MANY (ALL?) PROGRAM MISBEHAVIORS HAVE SECURITY IMPLICATIONS

LAST TIME: COMPUTABILITY

REVIEW: COMPUTABILITY

Theoretical Limit of Analysis: Rice's Theorem

- Analysis cannot be perfect
- We can bound the type of imperfection:
    - Soundness (no Type I errors)
    - Completeness (no Type II errors)

LECTURE OUTLINE

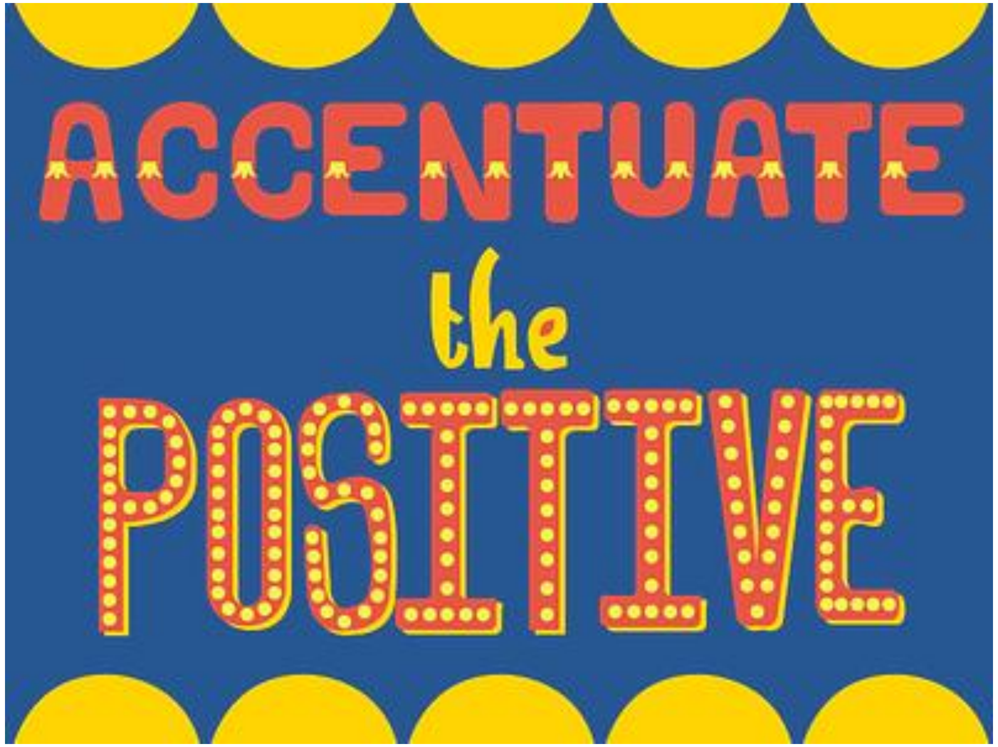
- Consequences of Rice's Theorem
- Categorizing Analyses
  - Dynamic
  - Static

BUILDING AN ANALYSIS

CONSEQUENCES OF RICE'S THEOREM

Analysis doesn't demand perfection

- Fertile grounds for exploring different techniques

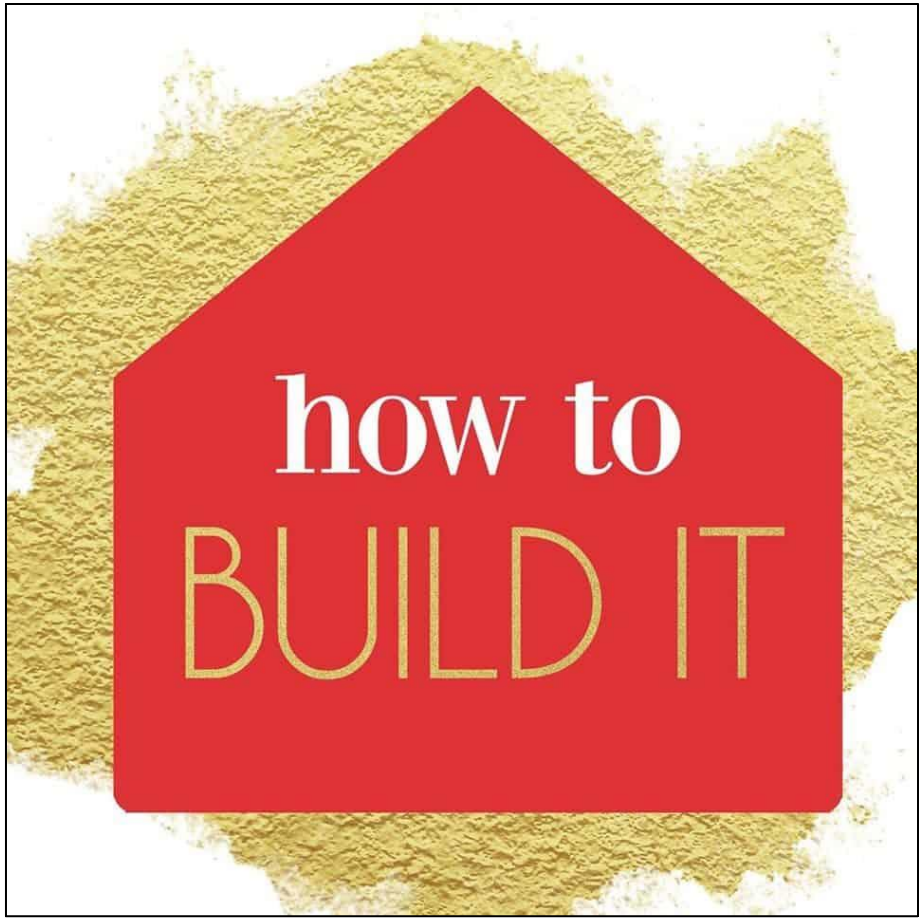


PRACTICAL ANALYSIS CONSIDERATIONS

CONSEQUENCES OF RICE'S THEOREM

We'll explore some of the ways an analysis may be structured

- Also spare a thought for assessing the quality and appropriateness of an analysis

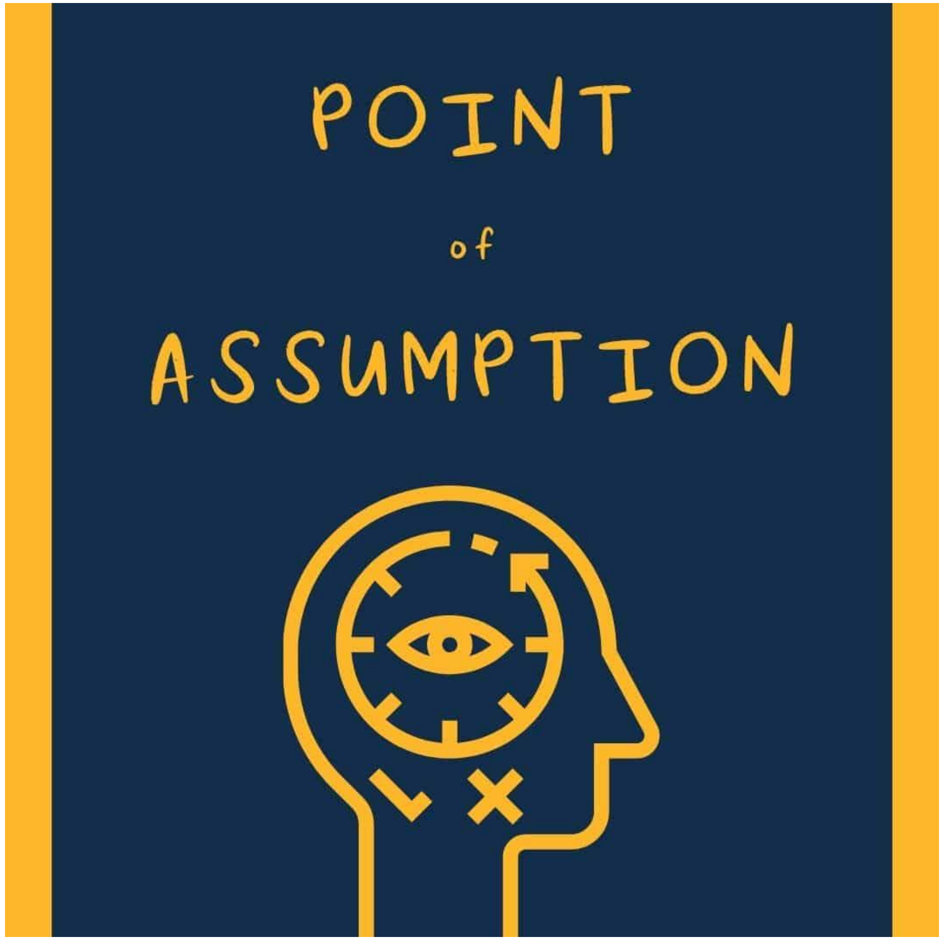


GUARANTEES WITH CAVEATS

CONSEQUENCES OF RICE'S THEOREM

Soundness/completeness aren't the whole story on analysis quality

- They are still super nice to have!
- Often useful to have a guarantee under some assumption

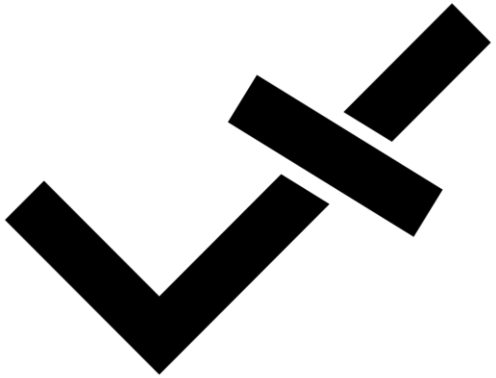


PARTIAL CORRECTNESS

CONSEQUENCES OF RICE'S THEOREM

Definition: An algorithm is partially correct if it only returns correct answers

- Definition allows for sometimes not returning an answer!



LECTURE OUTLINE

- Consequences of Rice’s Theorem
- Categorizing Analyses
  - Dynamic
  - Static

STATIC VS DYNAMIC ANALYSIS

CATEGORIZING ANALYSES

One distinction in analysis is how the analysis treats the target

- Static analysis – Operates without running the program
- Dynamic analysis - Operates with running the program

ANALYSIS METHOD VS ERRORS

CATEGORIZING ANALYSES

It’s natural to consider the types of compromises of each analysis method

- Static analysis
  - Often builds a model of the program, makes inferences on that model
  - Tends to make completeness easier
  - Scalability concerns for large programs
- Dynamic analysis
  - Often performs the analysis by straight up running the program, observing behavior
  - Tends to make soundness easier
  - Coverage problems

SOME FORMS OF DYNAMIC ANALYSIS

CATEGORIZING ANALYSES

Testing

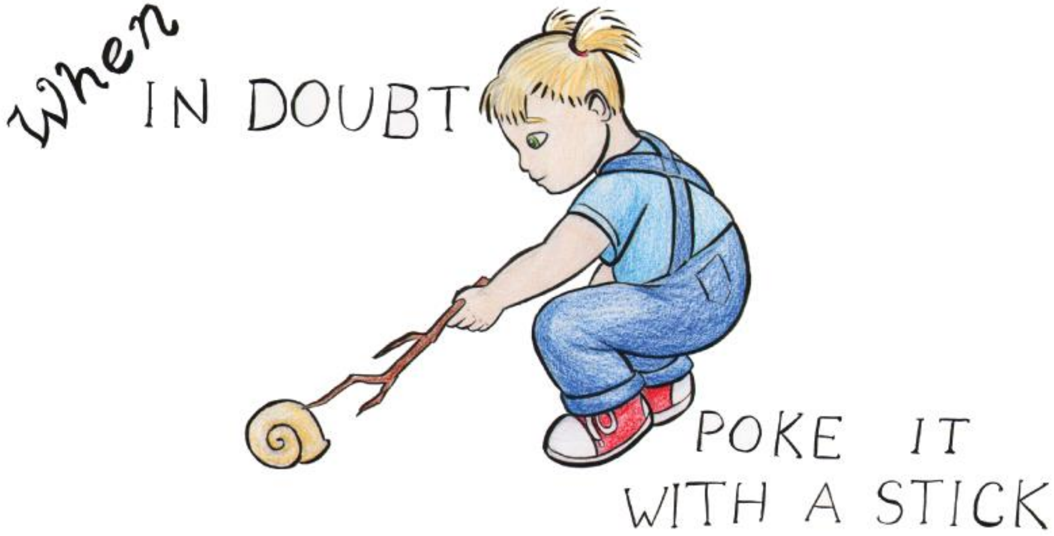
Fuzzing

Symbolic Execution

TESTING

CATEGORIZING ANALYSIS

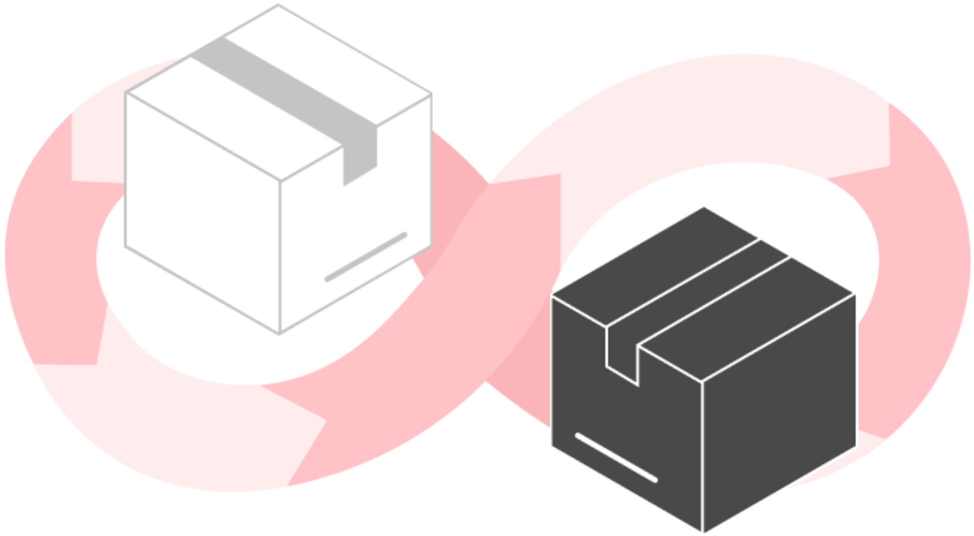
What happens when we do <this>?



TESTING

CATEGORIZING ANALYSIS

What happens when we do <this>?



CLASSIC LIMITATIONS OF TESTING

CATEGORIZING ANALYSIS

It’s hard to predict what might go wrong (presumably you’d have fixed it in this first place)

"FIXING" TESTING

CATEGORIZING ANALYSIS

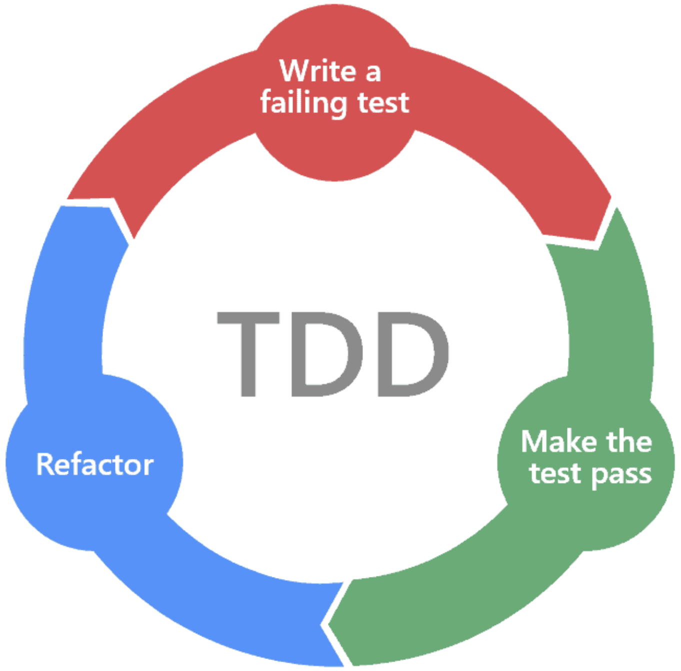
It's hard to predict what might go wrong (presumably you'd have fixed it in this first place)

- Could try to make a more intentional correspondence (TDD)
- Could try to leverage tools (Fuzzing)

TEST-DRIVEN DEVELOPMENT
CATEGORIZING ANALYSIS

1. Write a test case (expecting it to fail)
2. Implement enough functionality to pass the test case
3. Fix up the program

(repeat)



FUZZING

CATEGORIZING ANALYSIS

Automatically creating test cases

$x = \alpha$

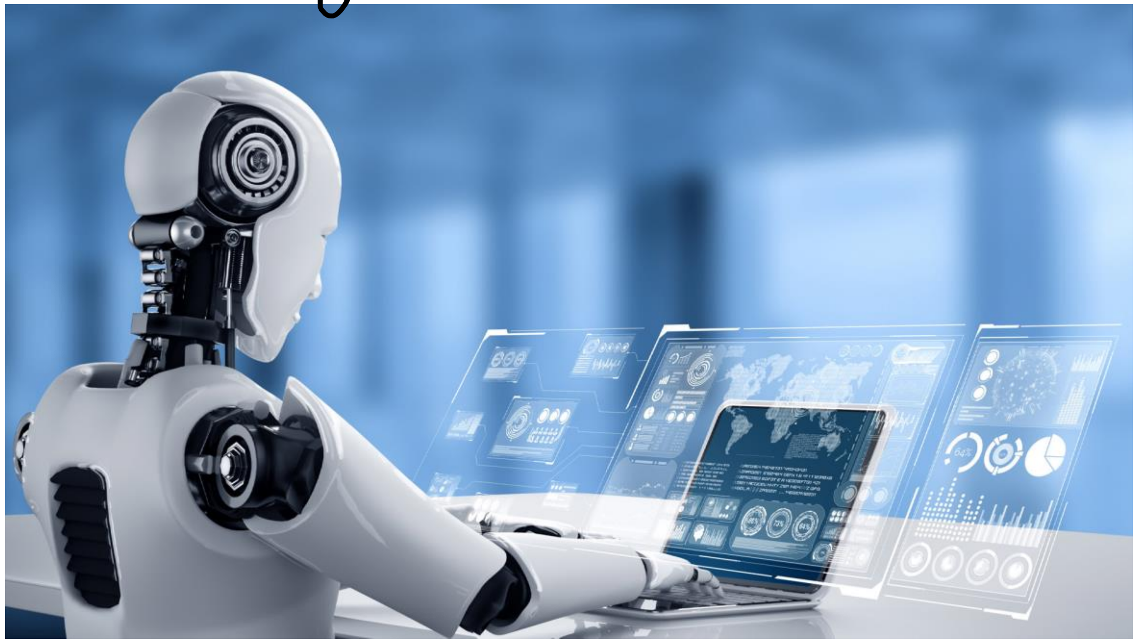
```
if (x == 200) {
    int *p = 0;
    *p = 7;
}
```

$\alpha = 200 \longrightarrow$

$\alpha \,!= 200 \longrightarrow$

$x = 1$

$x = 2$



HOW GOOD IS A DYNAMIC ANALYSIS?

CATEGORIZING ANALYSIS

At least in theory, an analysis can be measured in terms of how much of the state space is explored

- Since the dynamic analysis is executing one configuration at a time, we know how many states we're exploring
- What is much harder to determine is the total number of distinct configurations

State space: the collection of all possible configurations of a program

easy

$$\frac{\#\text{ confs executed}}{\text{total confs}}$$

hard

COVERAGE METRICS

CATEGORIZING ANALYSIS

```
int f(bool b) {
    Obj * o = null;
    int v = 2;
    if (b) {
        o = new Obj ();
        v = rand_int();
    }
    if (v == 2){
        o->setInvalid()
    }
    return o->property();
}
```

Line coverage

Branch coverage

Path coverage

LECTURE OUTLINE

- Consequences of Rice’s Theorem
- Categorizing Analyses
  - Dynamic
  - Static

SOME FORMS OF STATIC ANALYSIS

CATEGORIZING ANALYSES

Syntax Analysis

Dataflow Analysis

Abstract Interpretation

SYNTAX ANALYSIS

CATEGORIZING ANALYSES

Some troubling behavior of a program may be discoverable via simply observing syntactic structure

ANALYSIS SPECIFICITY

CATEGORIZING ANALYSIS

```
int f(bool b) {
    Obj * o = null;
    int v = 2;
    if (b) {
        o = new Obj ();
        v = rand_int();
    }
    if (v == 2){
        o->setInvalid()
    }
    return o->property();
}
```

Flow Sensitive

ANALYSIS SPECIFICITY

CATEGORIZING ANALYSIS

```
int f(bool b) {
    Obj * o = null;
    int v = 2;
    if (b) {
        o = new Obj ();
        v = rand_int();
    }
    if (v == 2){
        o->setInvalid()
    }
    return o->property();
}
```

Path Sensitive

ABSTRACT INTERPRETATION

CATEGORIZING ANALYSES

(Over)approximate the state of the program
(Over)approximate the domain of values

ABSTRACT INTERPRETATION

CATEGORIZING ANALYSES

(Over)approximate the state of the program
(Over)approximate the domain of values

Anything that isn't crystal clear to a static analysis tool probably isn't clear to your fellow programmers, either. The classic hacker disdain for "bondage and discipline languages" is short-sighted – the needs of large, long-lived, multi-programmer projects are just different than the quick work you do for yourself

- John Carmack

OVERVIEW DONE!

CATEGORIZING ANALYSES

We'll cover many of these techniques (and more!)

Next up:

- Looking at the kinds of program flaws that can cause problems
- Start looking at toolsets to build our analyses

LECTURE END!

- Consequences of Rice's Theorem
- Categorizing Analyses
  - Dynamic
  - ~~Static~~

Anything that isn’t crystal clear to a static analysis tool probably isn’t clear to your fellow programmers, either. The classic hacker disdain for “bondage and discipline languages” is short-sighted – the needs of large, long-lived, multi-programmer projects are just different than the quick work you do for yourself.John Carmack's Static Code Analysis post