CMPT 479/886
Automated Software Analysis & Security
Nick Sumner

Much adapted from Xiangyu Zhang, Antony Hosking, Sorin Lerner, Jonathan Aldrich, Sam Blackshear
Course Website

- www.cs.sfu.ca/~wsumner/teaching/886/18/
  - Schedule
  - Policies
  - Assignments
  - Paper Suggestions
Why are you here?
Why are you here?

- Programs are big, complex, and difficult to reason about.
Why are you here?

• Programs are big, complex, and difficult to reason about.

1001101
0101011
1101011
1101011
0001110
Why are you here?

- Programs are big, complex, and difficult to reason about.
Why are you here?

- Programs are big, complex, and difficult to reason about.

```
1001101
0101011
1101011
0001110
```

```
Foo.c
Bar.c
Baz.c
```
Why are you here?

- Programs are big, complex, and difficult to reason about.
Why are you here?

• Programs are big, complex, and difficult to reason about.
Why are you here?

- Programs are big, complex, and difficult to reason about.
Why are you here?

- Programs are big, complex, and difficult to reason about.
Why are you here?

- Programs are big, complex, and difficult to reason about.

1001101
0101011
1101011
0001110

Foo.c
Bar.c
Baz.c
Why are you here?

• Programs are big, complex, and difficult to reason about.

1001101
0101011
1101011
0001110

Foo.c
Bar.c
Baz.c

A+
Why are you here?

- Programs are big, complex, and difficult to reason about.
Why are you here?

• Programs are big, complex, and difficult to reason about.

Are there more efficient designs?
Why are you here?

- Programs are big, complex, and difficult to reason about.

Are there more efficient designs?

What is the cause of a bug?
Why are you here?

- Programs are big, complex, and difficult to reason about.

  Are there more efficient designs?

  What is the cause of a bug?

  How do I find new bugs?
Why are you here?

- Programs are big, complex, and difficult to reason about.

- Are there more efficient designs?
- What is the cause of a bug?
- How do I find new bugs?
- How do I find security vulnerabilities?
Why are you here?

- Programs are big, complex, and difficult to reason about.

- Are there more efficient designs?
- What is the cause of a bug?
- How do I find new bugs?
- How do I find security vulnerabilities?
- Can I protect against them?
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data.
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze Foo.c
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze

It looks like you have a bug!
A causes B at line 5.
B causes C at line 20.
C causes a crash at line 25!
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze
  - Transform
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze
  - Transform

EasyToHack.c
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze
  - Transform

EasyToHack.c → HardToHack.c
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze
  - Transform
  - Synthesize
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze
  - Transform
  - Synthesize
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze
  - Transform
  - Synthesize

\[
x + y + z = A1 + \text{if}(A1 > B1, A1 + B1, A1 \times C1)
\]
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze
  - Transform
  - Synthesize programs just like we can for other forms of data!
Why are you here?

- Programs are big, complex, and difficult to reason about.
- Programs are data. We can use computers to
  - Analyze
  - Transform
  - Synthesize

  programs just like we can for other forms of data!
- The family of techniques, representations, and tasks for analyzing programs comprise *program analysis*. 
Goal

- Learn how the difficult tasks in development can be pushed onto computers.
Goal

- Learn how the difficult tasks in development can be pushed onto computers.
  - Survey of *program analysis* techniques & papers
Goal

- Learn how the difficult tasks in development can be pushed onto computers.
  - Survey of *program analysis* techniques & papers
    - Profiling
      - (Speed, Potential Concurrency, Memory, ...)

Goal

- Learn how the difficult tasks in development can be pushed onto computers.
  - Survey of *program analysis* techniques & papers
    - Profiling
    - Testing

More effective tests. Bridge testing & verification
Goal

- Learn how the difficult tasks in development can be pushed onto computers.
  - Survey of program analysis techniques & papers
    - Profiling
    - Testing
    - Debugging
      - Explaining or locating the causes of bugs
Goal

- Learn how the difficult tasks in development can be pushed onto computers.
  - Survey of program analysis techniques & papers
    - Profiling
    - Testing
    - Debugging
    - Concurrency

How to explain race conditions?
Atomicity violations?
How to find 'Heisenbugs'?
Goal

- Learn how the difficult tasks in development can be pushed onto computers.
  - Survey of *program analysis* techniques & papers
    - Profiling
    - Testing
    - Debugging
    - Concurrency
    - Security

How to find vulnerabilities before attackers.
(...or as attackers)
Goal

- Learn how the difficult tasks in development can be pushed onto computers.
  - Survey of program analysis techniques & papers
    - Profiling
    - Testing
    - Debugging
    - Concurrency
    - Security
    - Verification
      How to prove the absence of behaviors.
Lens

Guiding questions:
Lens

Guiding questions:

- These problems are *impossible* to precisely solve in general.
Guiding questions:

- These problems are *impossible* to precisely solve in general. *What are the compromises?*
Guiding questions:

- These problems are *impossible* to precisely solve in general. What are the compromises?
  - What corner cases make them fail?
Guiding questions:

- These problems are *impossible* to precisely solve in general. What are the compromises?
  - What cornercases make them fail?
  - *Why* do these cornercases exist?
Lens

Guiding questions:

- These problems are *impossible* to precisely solve in general. What are the compromises?
  - What cornercases make them fail?
  - Why do these cornercases exist?

- How do authors present their work? Why?
Guiding questions:

• These problems are *impossible* to precisely solve in general. What are the compromises?
  – What cornercases make them fail?
  – Why do these cornercases exist?

• How do authors present their work? Why?
  – What is highlighted? What is hidden?
Lens

Guiding questions:

• These problems are *impossible* to precisely solve in general. What are the compromises?
  – What cornercases make them fail?
  – Why do these cornercases exist?

• How do authors present their work? Why?
  – What is highlighted? What is hidden?
  – How is it evaluated?
Structure

- First few weeks are review & background
  - I present.
  - There may be quizzes
  - You think about papers you'd like to present
Structure

- First few weeks are review & background
  - I present.
  - There may be quizzes
  - You think about papers you'd like to present

- Reading foundational & new papers
  - 2 student presentations & paper discussions per week
  - Brief critique (1-2 pages) on weeks you don't present
Structure

• First few weeks are review & background
  – I present.
  – There may be quizzes
  – You think about papers you'd like to present

• Reading foundational & new papers
  – 2 student presentations & paper discussions per week
  – Brief critique on weeks you don't present

• 3 small projects to introduce core skills
Structure

- First few weeks are review & background
  - I present.
  - There may be quizzes
  - You think about papers you'd like to present
- Reading foundational & new papers
  - 2 student presentations & paper discussions per week
  - Brief critique on weeks you don't present
- 3 small projects to introduce core skills

Available now!
Structure

- First few weeks are review & background
  - I present.
  - There may be quizzes
  - You think about papers you'd like to present
- Reading foundational & new papers
  - 2 student presentations & paper discussions per week
  - Brief critique on weeks you don't present
- 3 small projects to introduce core skills
- 1 large course project
Presentations

• Guidelines on website

• 2 Goals
  – Help reinforce the material for the class
  – Lead an interesting discussion to examine the trade offs of each technique. (I'll be helping.)
Presentations

- Guidelines on website
- 2 Goals
  - Help reinforce the material for the class
  - Lead an interesting discussion to examine the trade-offs of each technique. (I'll be helping.)
- **Show** how the technique behaves in the **best case**
- **Show** or lead discussion on where it might behave poorly
Presentations

- Guidelines on website

2 Goals
  - Help reinforce the material for the class
  - Lead an interesting discussion to examine the trade offs of each technique. (I'll be helping.)

- Show how the technique behaves in the best case
- Show or lead discussion on where it might behave poorly

- Groups of TBD (5?) will present each paper.
  - Volunteer or be volunteered
Critiques

- Guidelines on website
- 1-2 page response to 1 paper each week that you do not present.
Critiques

- Guidelines on website
- 1-2 page response to 1 paper each week that you do not present.
- Primarily meant to prepare you for the discussion on the paper that week.
Term Projects

- Groups of 4. (Grad groups can be smaller)
- 1 page proposals due October 9.
- Brief meetings with me on October 10.
Term Projects

- Groups of 4. (Grad groups can be smaller)
- 1 page proposals due October 9.
- Brief meetings with me on October 10.
- Find something that interests (or irritates) you and go after it!
Term Projects

- Groups of 4. (Grad groups can be smaller)
- 1 page proposals due October 9.
- Brief meetings with me on October 10.
- Find something that interests (or irritates) you and go after it!
  - Maybe look at how these techniques can help your existing research
Term Projects

- Groups of 4. (Grad groups can be smaller)
- 1 page proposals due October 9.
- Brief meetings with me on October 10.
- Find something that interests (or irritates) you and go after it!
  - Maybe look at how these techniques can help your existing research
  - You can use office hours to help find a direction.
Participation

• A class of this nature is driven by *discussion*.
Participation

● A class of this nature is driven by *discussion*.
  – You should not just show up but also contribute.
Participation

- A class of this nature is driven by *discussion*.
  - You should not just show up but also contribute.
  - Even the projects may require discussion for you to succeed.
Participation

- A class of this nature is driven by *discussion*.
  - You should not just show up but also contribute.
  - Even the projects may require discussion for you to succeed.

- Think about things in advance.
Participation

• A class of this nature is driven by *discussion*.
  – You should not just show up but also contribute.
  – Even the projects may require discussion for you to succeed.

• Think about things in advance.

• Come to class with questions (or answers).
Participation

• A class of this nature is driven by discussion.
  – You should not just show up but also contribute.
  – Even the projects may require discussion for you to succeed.
• Think about things in advance.
• Come to class with questions (or answers).

What you get out of a class like this is driven by what you are willing to put into it.
Let’s get started...