SWEN438 - DevOps

Lecture 14 - Fuzzing

Kirita-Rose Escott

School of Engineering and Computer Science
Victoria University of Wellington

Recap: The Third Way

"Focuses on creating a culture of **continual learning** *and* **experimentation***."*

– The DevOps Handbook

- Enabling learning and a safety culture.
- Improvement of daily work.
- Local discoveries to global improvements.
- Inject resilience patterns.
- Leaders reinforce a learning culture.

Recap: Techniques

- Stop-the-line mentality.
- Plan and rehearse for failures.
- Fast fix forward and roll back.

Recap: Industry

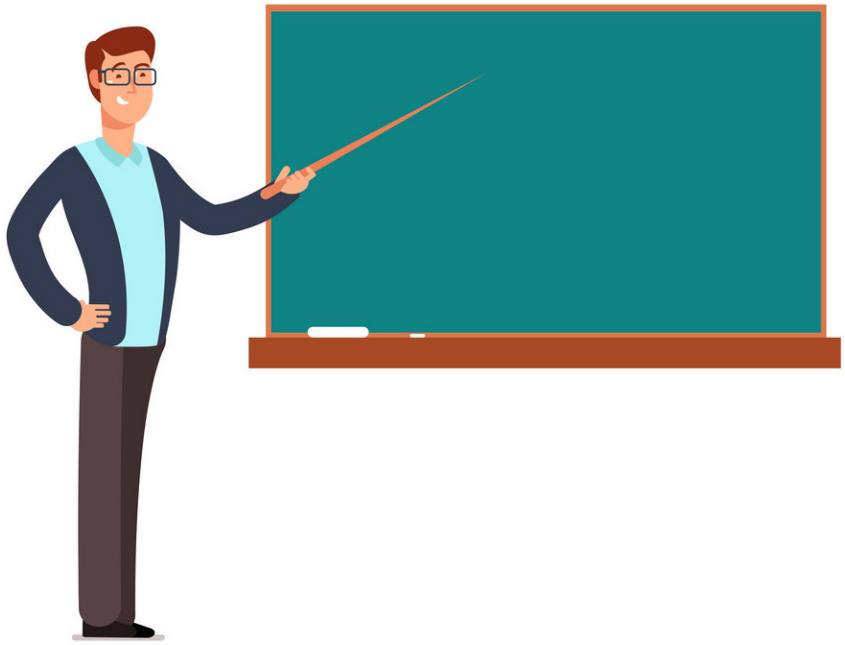
- **Different companies, different experiences**. Products teams vs. Service teams. *Need to be adaptable.*
- **Service Level Agreements (SLA)**. Contractual agreements between companies and their users. *Ensure minimum level of service is provided to users.*
- **Appropriate tooling**. There are lots of different tools, not all will be appropriate. *Need alert when SLAs in danger of being broken.*

Whakataukī

Hōhonu kakī, pāpaku uaua

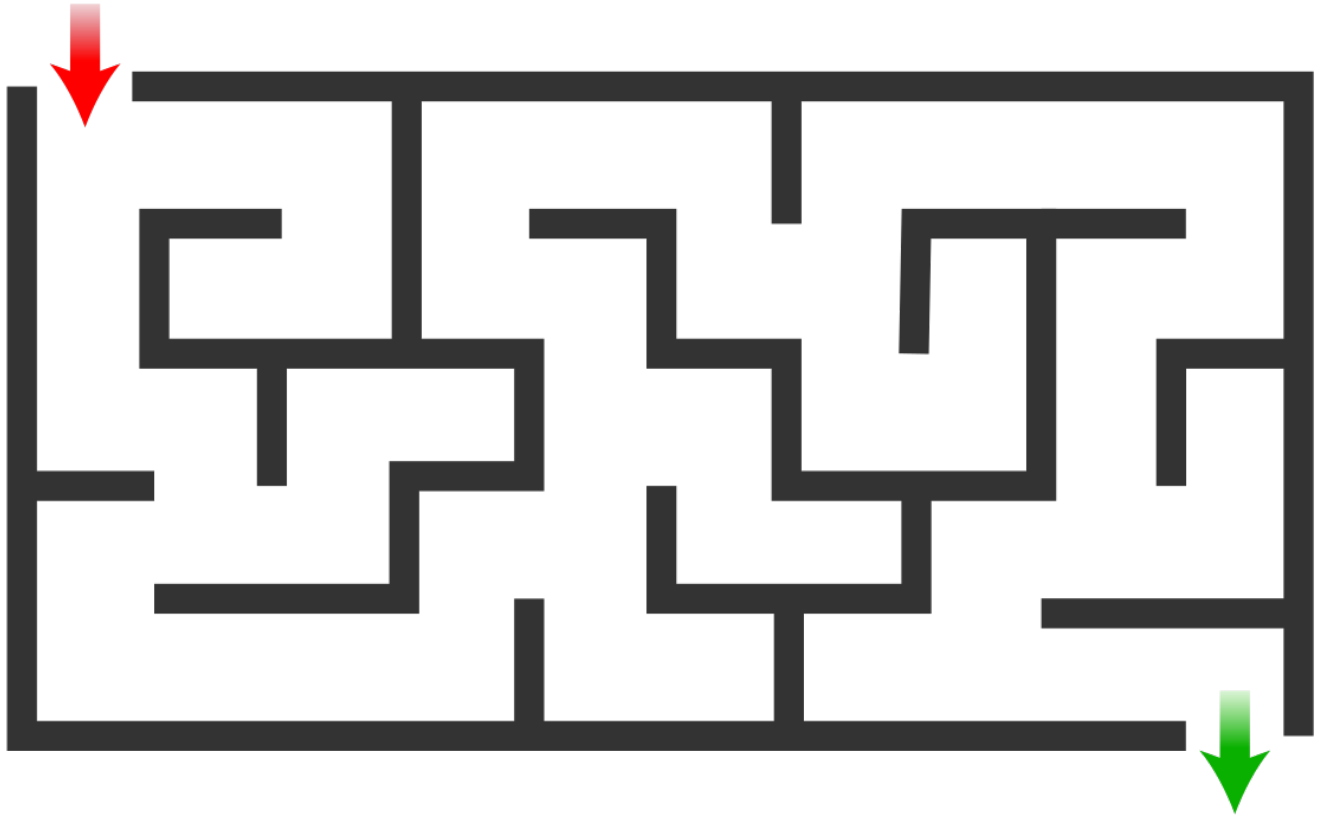
Long on words, short on actions

What is Fuzzing?

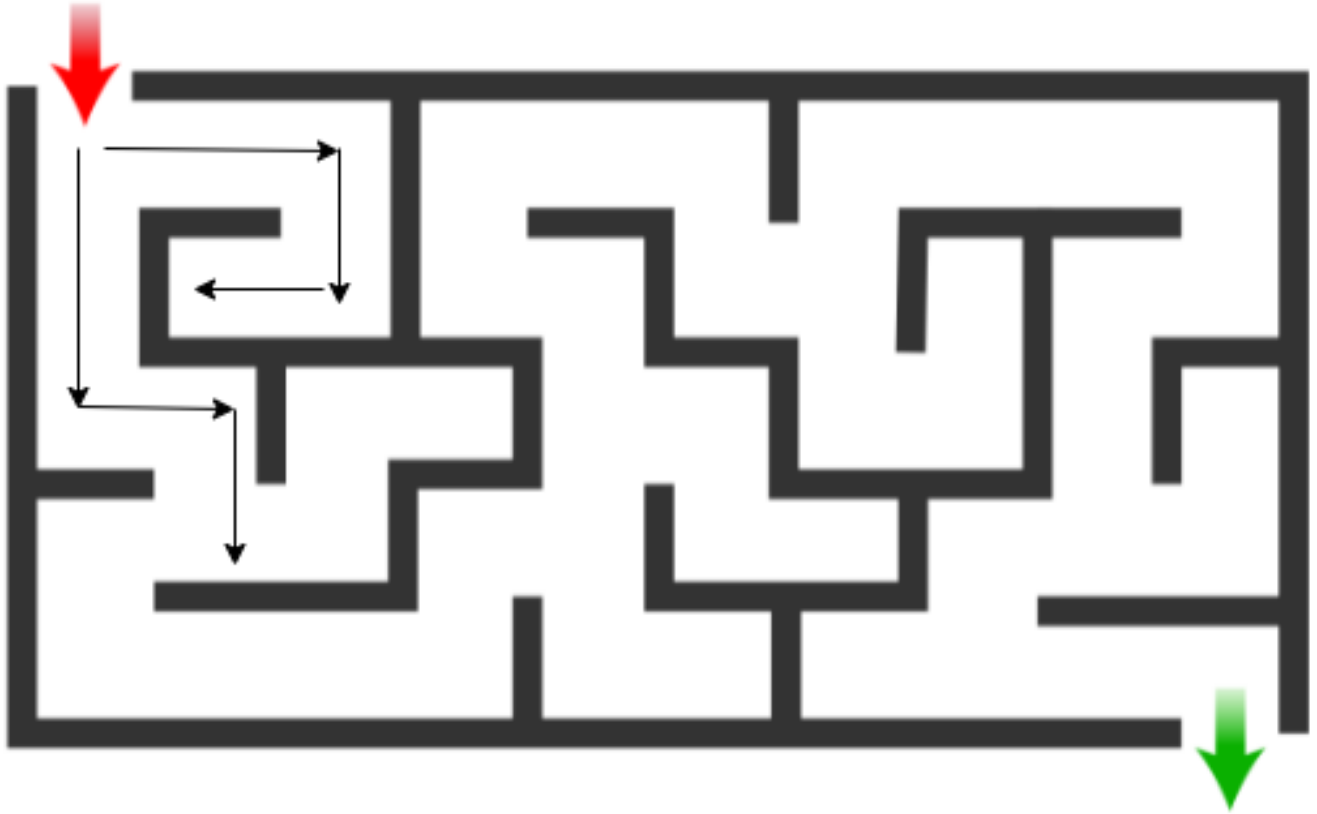


- Professor Bart Miller, University of Wisconsin and his students
- When giving random input to Unix, Windows and Apple applications
- Applications would crash one third of the time

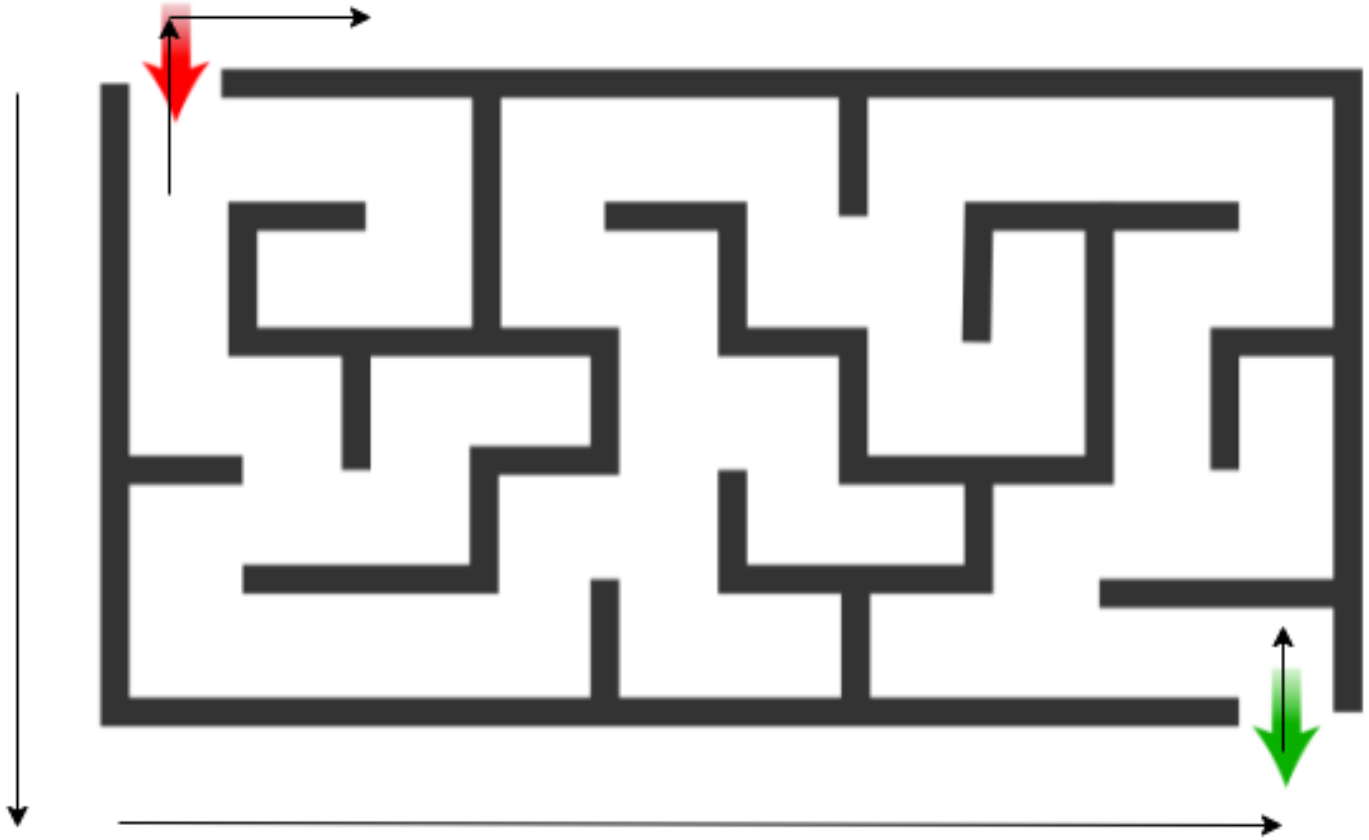
Maze Example



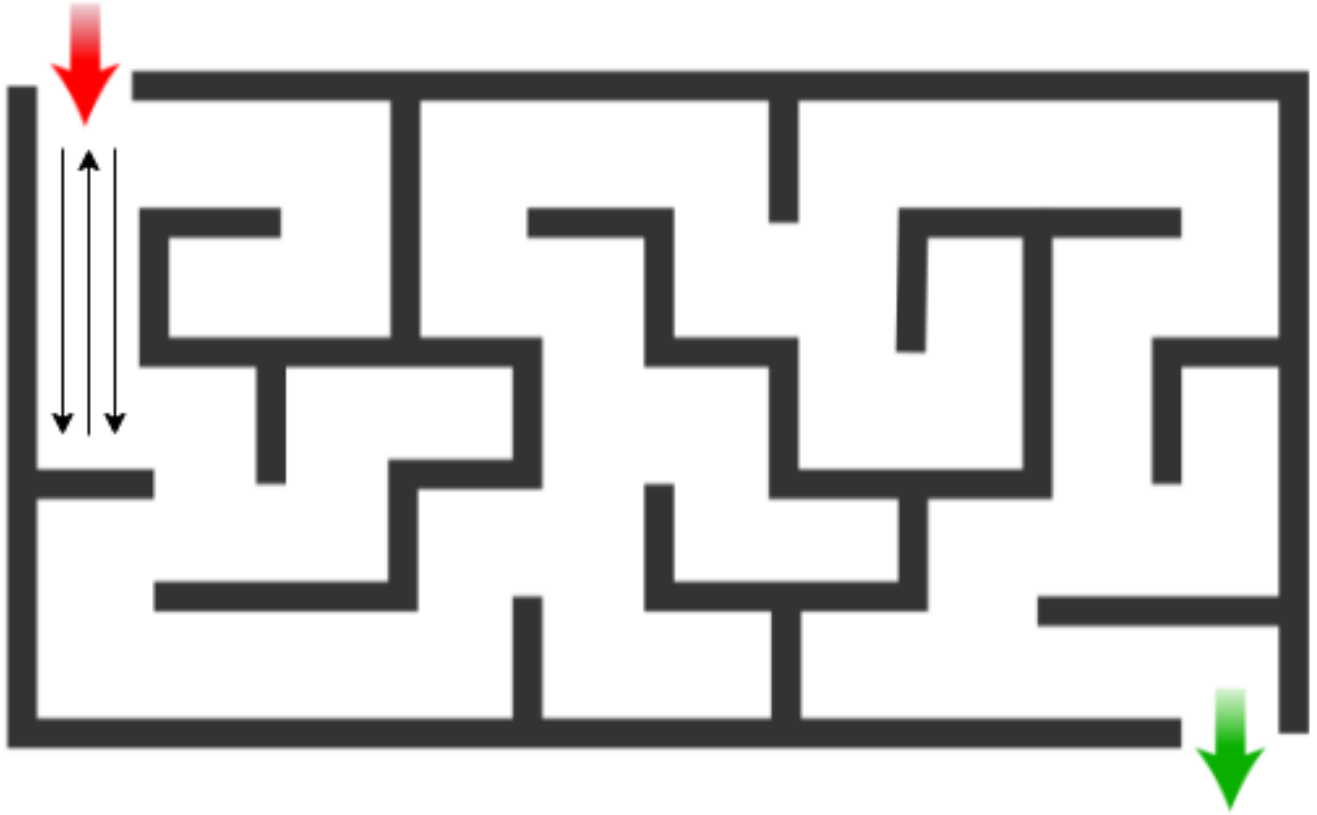
Maze Example



Maze Example



Maze Example



Why Fuzzing?

The purpose of fuzzing relies on the assumption that there are bugs within every program, which are waiting to be discovered. Fuzzing can add another point of view to classical software testing techniques (hand code review, debugging) because of it's non-human approach. It doesn't replace them, but is a reasonable complement.

– OWASP

What is Fuzzing?

Fuzzing is an automated testing technique that involves **invalid**, **unexpected** *or* **random data** *as inputs to a computer program. The goals is to then record the way the program responds in terms of any* **crashes**, **failed exceptions**, **hangs** *or* **memory leaks**.

Examples of Fuzzing Inputs

- **Numbers**. signed/unsigned integers/floats etc
- **Chars**. urls, command-line inputs
- **Metadata**. user-input text
- **Binary sequences**.

American Fuzzy Lop (AFL)

American Fuzzy Lop is a brute-force fuzzer coupled with an exceedingly simple but rock-solid instrumentation-guided genetic algorithm. It uses a modified form of edge coverage to effortlessly pick up subtle, local-scale changes to program control flow.

– American Fuzzy Lop

What is AFL?

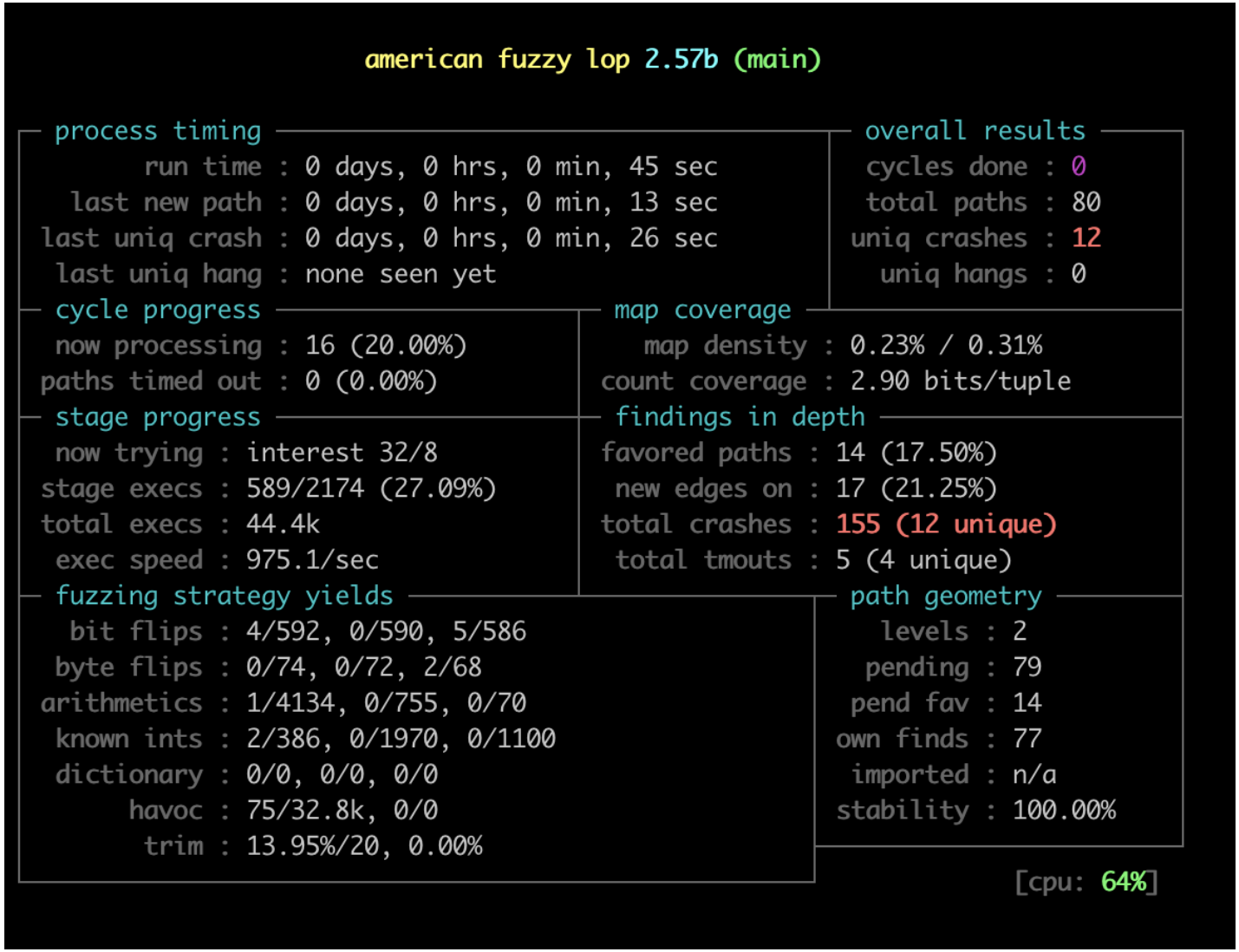
American Fuzzy Lop (AFL) is a widely-used fuzzing tool developed by Google.
AFL is used in Project Zero which is a team of security researchers at Google who study zero-day vulnerabilities in the hardware and software systems used by users around the world.

What does it do?

The overall algorithm can be summed up as:

1. Load user-supplied initial test cases into the queue,
2. Take next input file from the queue,
3. Attempt to trim the test case to the smallest size that doesn't alter the measured behavior of the program,
4. Repeatedly mutate the file using a balanced and well-researched variety of traditional fuzzing strategies,
5. If any of the generated mutations resulted in a new state transition recorded by the instrumentation, add mutated output as a new entry in the queue.
6. Go back to 2.

What does it do? (Cont'd)



What does it do? (Cont'd)

AFL will continue to fuzz until stop it. At minimum, you want to allow the fuzzer to complete one queue cycle, which may take anywhere from a couple of hours to a week or so.

The output can be summed up as:

- **Queue**. Test cases for every distinctive execution path, plus all the starting files given by the user.

- **Crashes**. Unique test cases that cause the tested program to receive a fatal signal

- **Hangs**. Unique test cases that cause the tested program to time out.

Does fuzzing = reliability?

- A *technical* approach to further automated testing
- Underpinned by *The Third Way*
- An *aspect* of reliability, that may or may not be appropriate

What now?

Try fuzzing out for yourself in Lab 5!