Week 1: Lecture A

Course Introduction

Monday, January 8, 2024

Reminders

- Be sure to join the course **Canvas** and **Piazza**
  - See links at top of course page
  - cs.utah.edu/~snagy/courses/cs5963/
- Trouble accessing? See me after class!
  - Or email me at: snagy@cs.utah.edu

Today's Class

- **Welcome to CS 5963/6963** 😀
- Course Overview
- What is software testing?
  - How does it work?
  - Why do we use it?
- Ethics and Academic Integrity

About Me

Stefan Nagy
Assistant Professor, KSoC

cs.utah.edu/~snagy
twitter.com/snagycs
@snagy@infosec.exchange

Co-founder and Co-director:

SSG UTAH SOFTWARE SECURITY GROUP
SCHOOL OF COMPUTING | THE UNIVERSITY OF UTAH

Places I've been:

University of Utah, 2022–now

Virginia Tech, Ph.D. 2016–2022

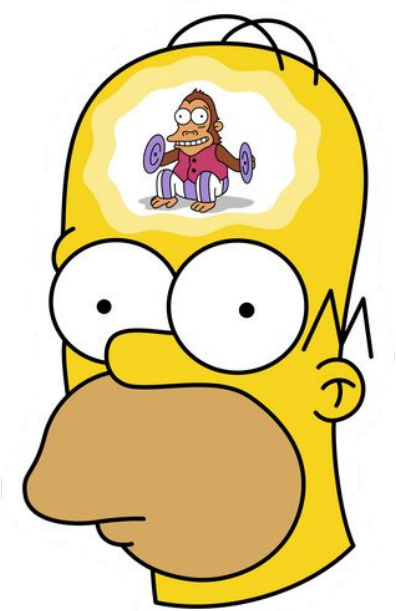
Univ. of Illinois, B.S. 2012–2016

My Research Group

Our work: systems and software security, binary analysis, fuzzing

Course Overview

What brought you here?

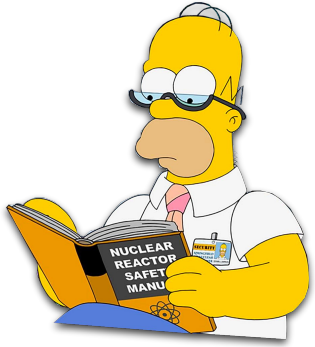


Course Goals

- Help you become **better researchers**
- Expose you to **different perspectives**
- Experience with **state-of-the-art tools**
- Get course credit so you can graduate?
- All while learning about **software testing**

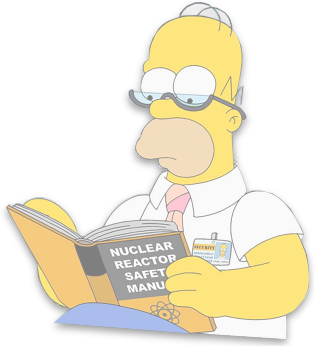
Course Components

- Reading & evaluating research
  - Contextualize
  - Pros vs. cons
  - Contribution
  - Summarizing
  - Identify assumptions



Course Components

- Reading & evaluating research
  - Contextualize
  - Pros vs. cons
  - Contribution
  - Summarizing
  - Identify assumptions



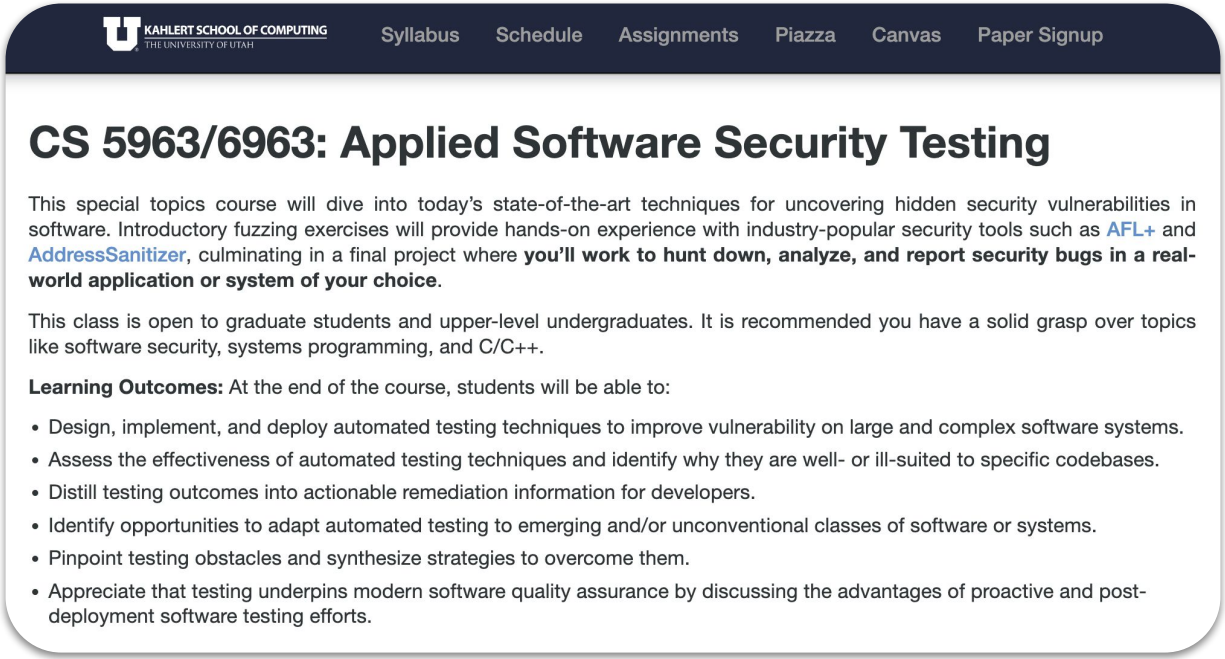
- Conducting & presenting research
  - Identify an open problem and solve it
  - Develop new tooling and release it
  - Evaluate and disseminate your work
  - **Help society by finding security bugs**

Course Format

- **Meetings:** Mondays & Wednesdays at 1:25 – 2:45 PM
- **Locations:** **WEB L114** (class), **MEB 3446** (office hours)
  - Office hours held from 2:45 – 3:30 PM following lecture
- **20 – 30 min:** instructor-led lecture on topic of the day
  - Slides will be posted on the course website Schedule
- **40 – 50 min:** student-led paper presentation & discussion
  - One or two papers per day related to the lecture topic

Course Website

cs.utah.edu/~snagy/courses/cs5963



Schedule

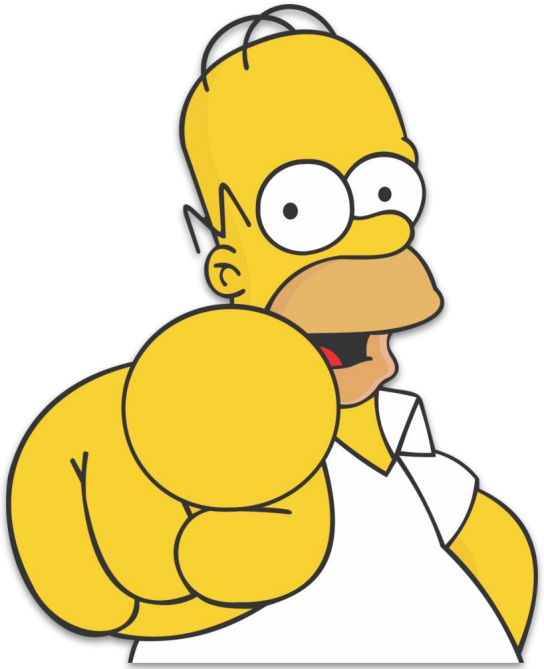
- **Weeks 1 – 3:** Course Intro & Systems Research 101
- **Weeks 4 – 9:** Fundamentals of Software Fuzzing
  - Three (relatively easy) labs
  - Semester Project begins on Week 6
- **Weeks 10 – 12:** Emerging Enhancements in Fuzzing
- **Weeks 13 – 16:** New Frontiers & Project Presentations

Grading

- **10%** – Attendance & Paper Discussions
- **10%** – Paper Presentations (one per student)
- **15%** – Lab 1: Beginner Fuzzing
- **15%** – Lab 2: Crash Triage
- **15%** – Lab 3: Harnessing
- **35%** – Final Project

Attendance & Participation

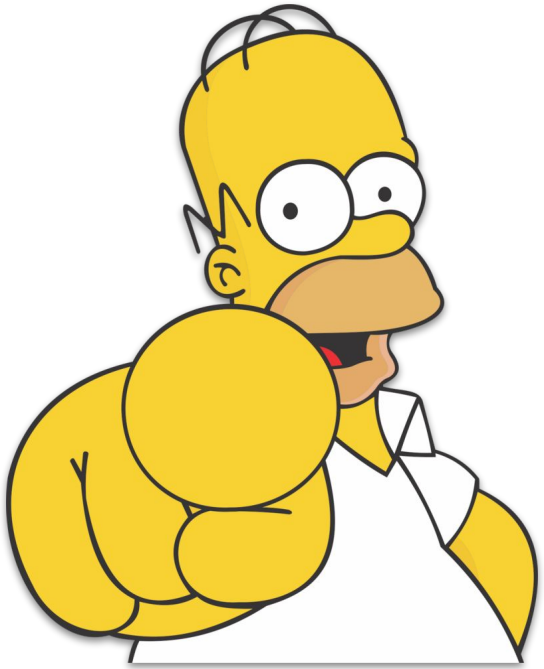
- **Requirement 1:** Show up to class
  - Contact me about absences in advance



Attendance & Participation

- **Requirement 1:** Show up to class
  - Contact me about absences in advance

- **Requirement 2:** Participate during other students' presentations
  - Ask thoughtful questions
  - Understand the science
  - **Help your classmates learn**



Paper Presentations

- **Two paper presentations** per lecture, followed by **5–10 minute discussions**

Paper Presentations

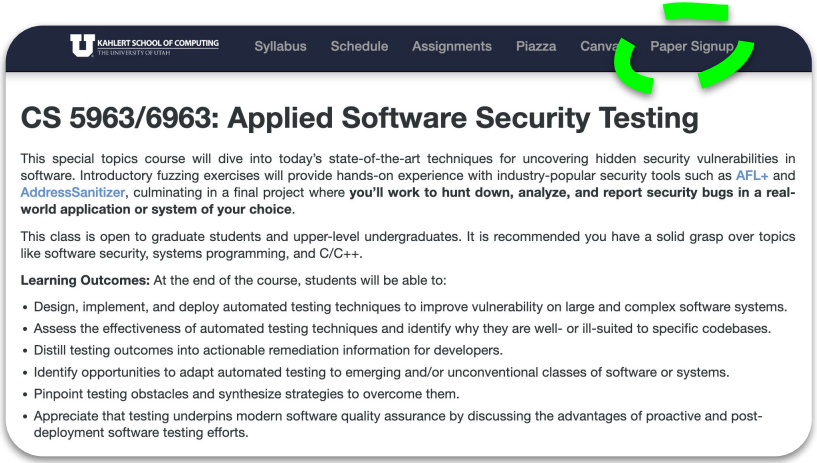
- **Two paper presentations** per lecture, followed by **5–10 minute discussions**
- **Audience:** you are **not** required to read the paper
  - … but you are required to **participate** in the discussion!

Paper Presentations

- **Two paper presentations** per lecture, followed by **5–10 minute discussions**
- **Audience:** you are **not** required to read the paper
  - … but you are required to **participate** in the discussion!
- **Presenters:** your job is to **teach us the paper**
  - Summarizing
  - Contextualize
  - Pros vs. cons
  - Contributions
  - Key assumptions
  - **Prepare a short slide deck** (you can get “inspired” from existing presentations)
  - **15 – 20 minute presentation** (with a 5–10 minute audience discussion to follow)

Paper Presentations

- **Signup sheet** available on course website (must use **UofU gcloud** account)
  - **38 fuzzing papers** from top venues in security, software engineering, and some workshops
  - Choose one paper by **Monday, January 22**



KAHLERT SCHOOL OF COMPUTING
THE UNIVERSITY OF UTAH

Syllabus Schedule Assignments Piazza Canvas Paper Signup

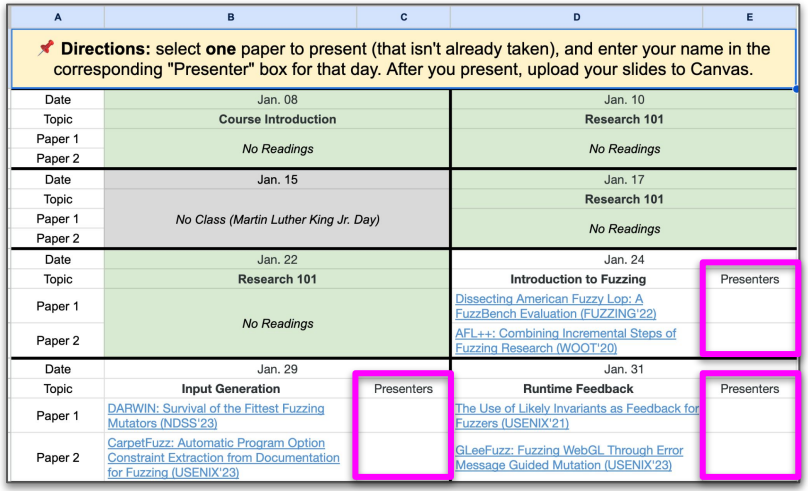
CS 5963/6963: Applied Software Security Testing

This special topics course will dive into today's state-of-the-art techniques for uncovering hidden security vulnerabilities in software. Introductory fuzzing exercises will provide hands-on experience with industry-popular security tools such as AFL+ and AddressSanitizer, culminating in a final project where **you'll work to hunt down, analyze, and report security bugs in a real-world application or system of your choice**.

This class is open to graduate students and upper-level undergraduates. It is recommended you have a solid grasp over topics like software security, systems programming, and C/C++.

Learning Outcomes: At the end of the course, students will be able to:

- Design, implement, and deploy automated testing techniques to improve vulnerability on large and complex software systems.
- Assess the effectiveness of automated testing techniques and identify why they are well- or ill-suited to specific codebases.
- Distill testing outcomes into actionable remediation information for developers.
- Identify opportunities to adapt automated testing to emerging and/or unconventional classes of software or systems.
- Pinpoint testing obstacles and synthesize strategies to overcome them.
- Appreciate that testing underpins modern software quality assurance by discussing the advantages of proactive and post-deployment software testing efforts.



A	B	C	D	E
📌 **Directions:** select **one** paper to present (that isn't already taken), and enter your name in the corresponding "Presenter" box for that day. After you present, upload your slides to Canvas.				
Date	Jan. 08		Jan. 10	
Topic	**Course Introduction**		**Research 101**	
Paper 1	*No Readings*		*No Readings*	
Paper 2				
Date	Jan. 15		Jan. 17	
Topic	*No Class (Martin Luther King Jr. Day)*		**Research 101**	
Paper 1			*No Readings*	
Paper 2				
Date	Jan. 22		Jan. 24	
Topic	**Research 101**		**Introduction to Fuzzing**	Presenters
Paper 1	*No Readings*		Dissecting American Fuzzy Lop: A FuzzBench Evaluation (FUZZING'22)	
Paper 2			AFL++: Combining Incremental Steps of Fuzzing Research (WOOT'20)	
Date	Jan. 29		Jan. 31	
Topic	**Input Generation**	Presenters	**Runtime Feedback**	Presenters
Paper 1	DARWIN: Survival of the Fittest Fuzzing Mutators (NDSS'23)		The Use of Likely Invariants as Feedback for Fuzzers (USENIX'21)	
Paper 2	CarpetFuzz: Automatic Program Option Constraint Extraction from Documentation for Fuzzing (USENIX'23)		GLeeFuzz: Fuzzing WebGL Through Error Message Guided Mutation (USENIX'23)	

Hands-on Labs

- Three (relatively easy) labs to be completed **solo**
  - **Lab 1:** Beginner fuzzing
  - **Lab 2:** Crash triage
  - **Lab 3:** Target harnessing

Hands-on Labs

- Three (relatively easy) labs to be completed **solo**
  - **Lab 1:** Beginner fuzzing
  - **Lab 2:** Crash triage
  - **Lab 3:** Target harnessing
- Paced with the introductory content from Weeks 4–9
  - Apply the techniques you've learned in class
  - Get familiar with state-of-the-art tools like **AFL** and **ASAN**
  - **Deliverables:** a short report (1–3 pages) of what you've learned

Hands-on Labs

- Three (relatively easy) labs to be completed **solo**
  - **Lab 1:** Beginner fuzzing
  - **Lab 2:** Crash triage
  - **Lab 3:** Target harnessing
- Paced with the introductory content from Weeks 4–9
  - Apply the techniques you've learned in class
  - Get familiar with state-of-the-art tools like **AFL** and **ASAN**
  - **Deliverables:** a short report (1–3 pages) of what you've learned
- **Designed to prepare you for the Semester Final Project**

Semester Final Project

- Objective: **uncover new bugs in a real-world program**
- Team up in groups of **1 – 4**
- Select an “interesting” target program of your choice; e.g.:
  - Popular applications
  - Nintendo emulators
  - Old computer games
  - MacOS Rosetta
  - **GET CREATIVE!**
- **Figure out how to fuzz** your target, **find bugs**, and **responsibly disclose them**
- **Deliverables:** a report, disclosure of bugs, and open-source your team’s fuzzer

Semester Final Project

- Objective: u
- Team up in
- Select an "in
  - Popular
  - Nintend
  - Old computer games
  - MacOS Rosetta
  - GET CRE
- Figure out h e them
- Deliverables n's fuzzer

5-minute project **proposal** on Feb. 28

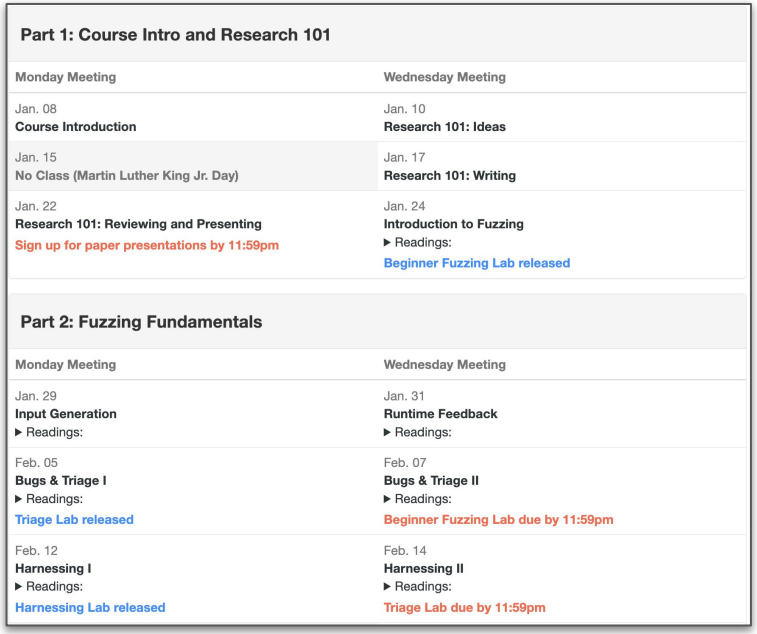
Final presentations at semester's end

You have full creative liberty—get creative and **fuzz something fun**!

Key Dates

- **Jan. 15** No class (MLK Jr. Day)
- **Jan. 22** **Select one paper to present**
- **Feb. 07** Lab 1 due
- **Feb. 14** Lab 2 due
- **Feb. 19** No class (President's Day)
- **Feb. 28** Lab 3 due
- **Feb. 28** **5-minute project proposals**
- **Mar. 04 & 06** No class (Spring Break)
- **Apr. 17 & 22** **Final project presentations**

cs.utah.edu/~snagy/courses/cs5963/schedule



Part 1: Course Intro and Research 101

Monday Meeting	Wednesday Meeting
Jan. 08 **Course Introduction**	Jan. 10 **Research 101: Ideas**
Jan. 15 **No Class (Martin Luther King Jr. Day)**	Jan. 17 **Research 101: Writing**
Jan. 22 **Research 101: Reviewing and Presenting** Sign up for paper presentations by 11:59pm	Jan. 24 **Introduction to Fuzzing** ▸ Readings: Beginner Fuzzing Lab released

Part 2: Fuzzing Fundamentals

Monday Meeting	Wednesday Meeting
Jan. 29 **Input Generation** ▸ Readings:	Jan. 31 **Runtime Feedback** ▸ Readings:
Feb. 05 **Bugs & Triage I** ▸ Readings: Triage Lab released	Feb. 07 **Bugs & Triage II** ▸ Readings: Beginner Fuzzing Lab due by 11:59pm
Feb. 12 **Harnessing I** ▸ Readings: Harnessing Lab released	Feb. 14 **Harnessing II** ▸ Readings: Triage Lab due by 11:59pm

Lateness Policy

- Assignments will be posted on course website
  - See cs.utah.edu/~snagy/courses/cs5963/assignments
- Due by **11:59 PM** on the specified deadline date
  - Late assignments will **not** be accepted
- If you are sick / traveling / abducted by aliens…
  - Try to keep me posted and we will figure something out

Course Materials

- No textbook is required for this course
- Some excellent resources on fuzzing are:
  - **The Fuzzing Book** by Zeller, Gopinath, Böhme, Fraser, and Holler
  - **Fuzzing Against the Machine** by Antonio Nappa and Blazquez
- Other general computer security textbooks:
  - **Introduction to Computer Security** by Goodrich and Tamassia
  - **Security Engineering** by Ross Anderson
- These are are linked on the course syllabus
  - cs.utah.edu/~snagy/courses/cs5963/

No Exams

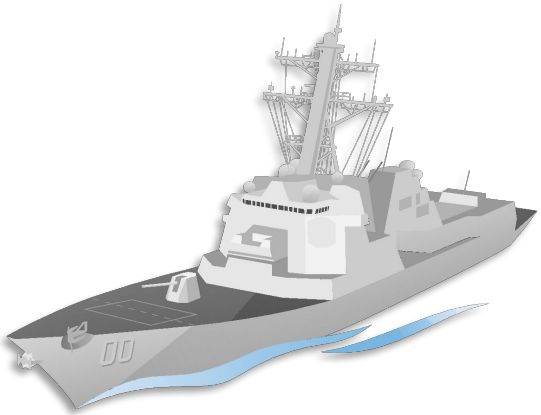
Questions?

A Brief Overview of Software Testing

Our world depends on software...

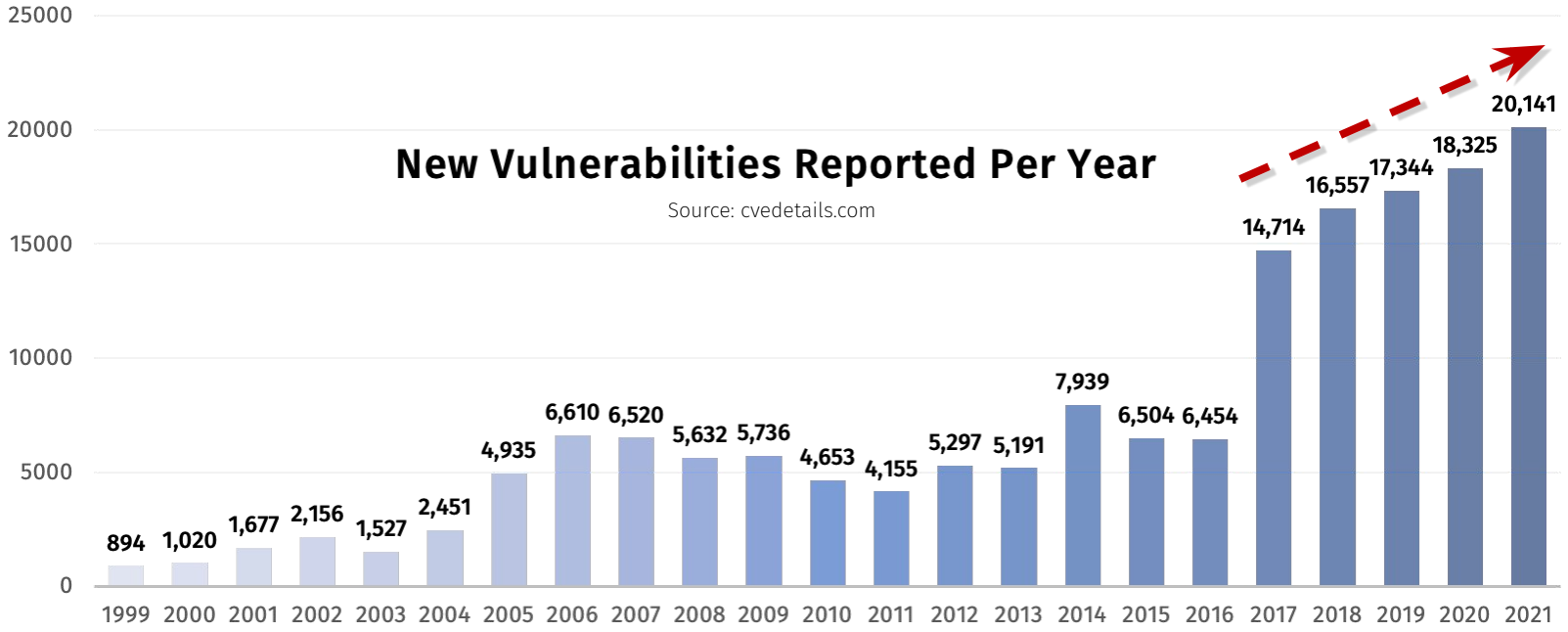
Personal Technology

Infrastructure & Industry



Military and Government

… and software security is a *nightmare*



… and software security is a *nightmare*

Amnesty says NSO's Pegasus used to hack phones of Palestinian rights workers

'A cyber-attack disrupted my cancer treatment'

Cyber-attack hits UK internet phone providers

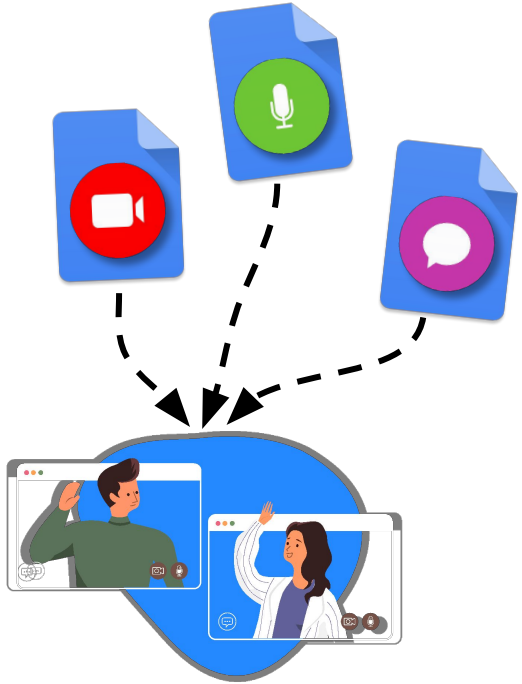
Janesville school district hit by ransomware attack

Solarwinds hackers are targeting the global IT supply chain, Microsoft says

New York subway hacked in computer breach linked to China

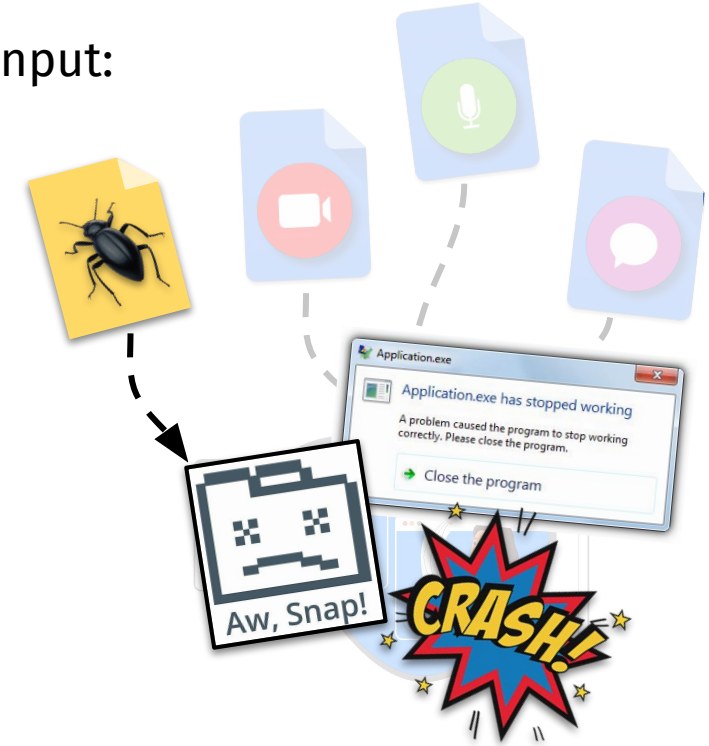
Why is software insecure?

- Modern applications accept many sources of input:
  - **Files**
  - **Arguments**
  - **Environment variables**
  - **Network packets**



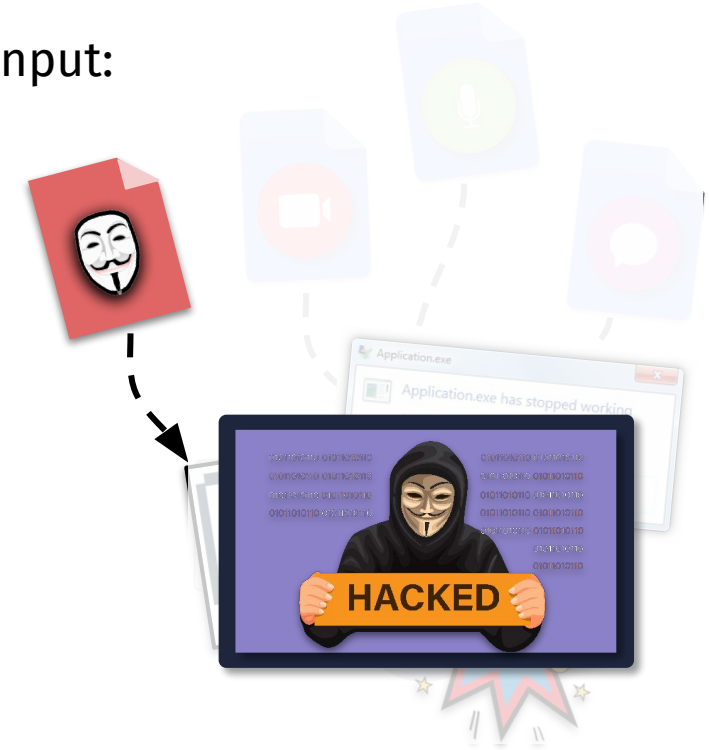
Why is software insecure?

- Modern applications accept many sources of input:
  - **Files**
  - **Arguments**
  - **Environment variables**
  - **Network packets**
- Developer mistakes create **software bugs**
  - Pointer mismanagement, bounds checking, etc.

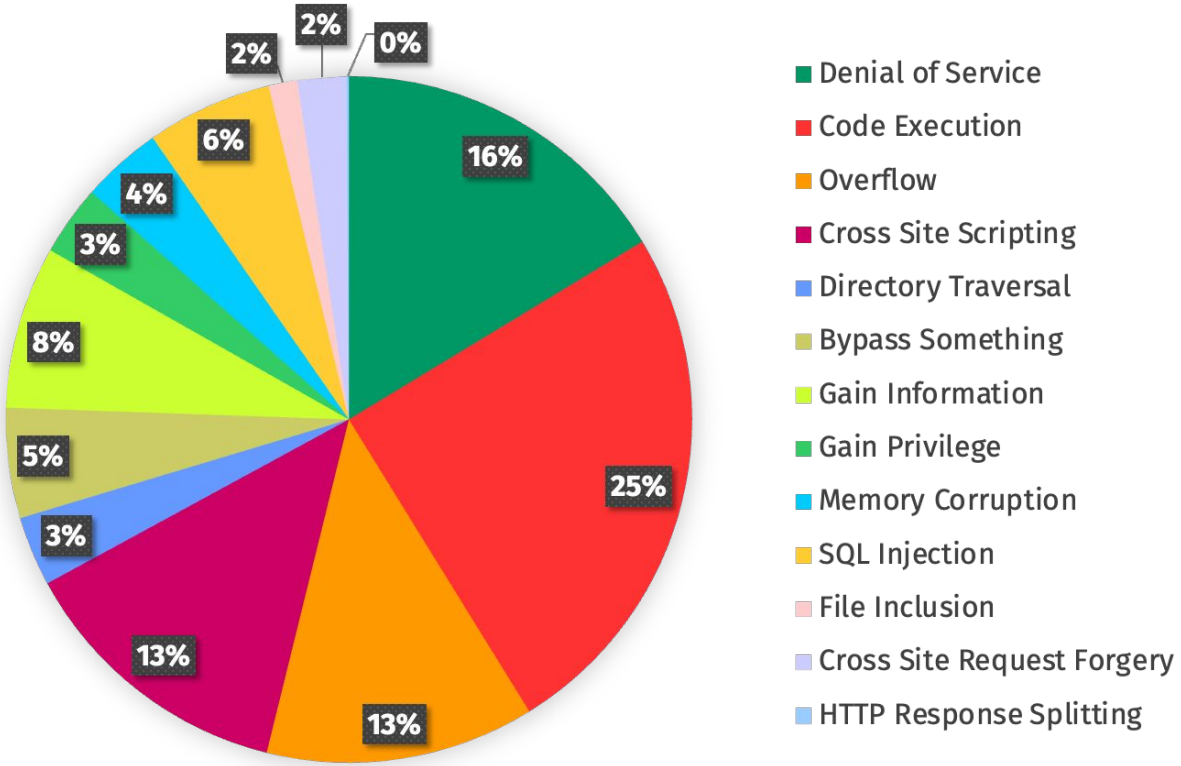


Why is software insecure?

- Modern applications accept many sources of input:
  - **Files**
  - **Arguments**
  - **Environment variables**
  - **Network packets**
- Developer mistakes create **software bugs**
  - Pointer mismanagement, bounds checking, etc.
- Many bugs are **exploitable** **by attackers**
  - Denial of service, info leakage, code execution



Software Security Vulnerabilities



Source: cvedetails.com

Software Security Vulnerabilities

- **WH:** **$100+ billion** in annual cybersecurity damages
- **NIST:** **25 vulnerabilities** per every 1,000 lines of code
- **NASA:** **1–100 million lines of code** in modern software
- **DHS:** **80% of attacks exploit unknown** vulnerabilities

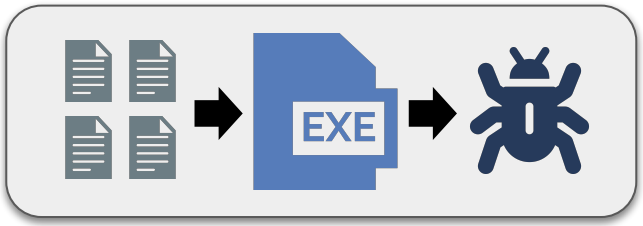
We need effective, scalable approaches for
vetting *all* software and systems

Proactive Vulnerability Discovery

Static Analysis:

- Analyze program **without running it**
- Accuracy a major concern
  - **False negatives** (vulnerabilities missed)
  - **False positives** (results are unusable)
- As code size grows, **speed drops**

Dynamic Testing:



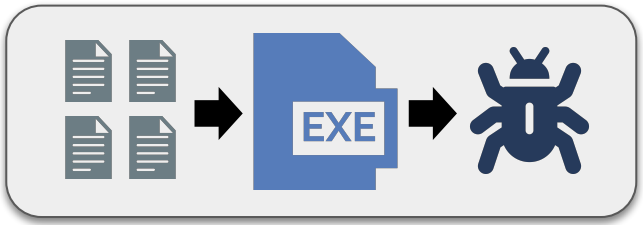
- Analyze program **by executing it**
- Better accuracy: **no false positives**
  - Execution reveals only what exists
  - Program crashed? You found a bug!
- Capable of very **high throughput**

Proactive Vulnerability Discovery

- **Widely deployed** in industry today:

- **Over 36,000 errors** in 550 codebases
- **Over 18,000 errors** in Google Chrome
- **Over 11,000 errors** in Linux's kernel

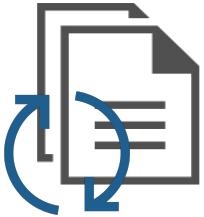
Dynamic Testing:



- Analyze program **by executing it**
- Better accuracy: **no false positives**
  - Execution reveals only what exists
  - Program crashed? You found a bug!
- Capable of very **high throughput**

Key Approach: Fuzz Testing

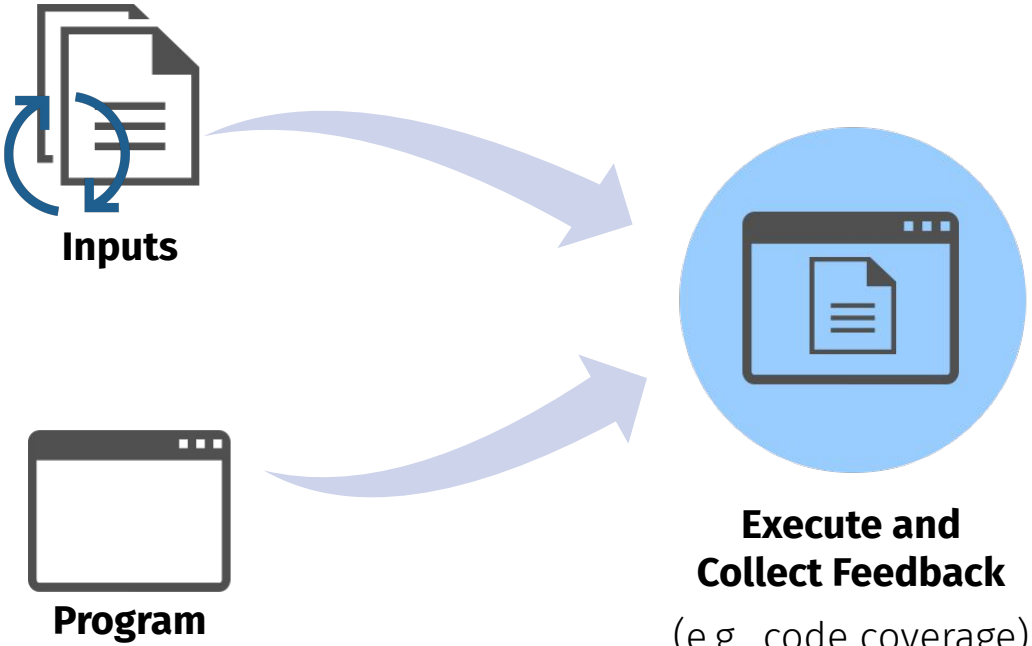
Key Approach: Fuzz Testing



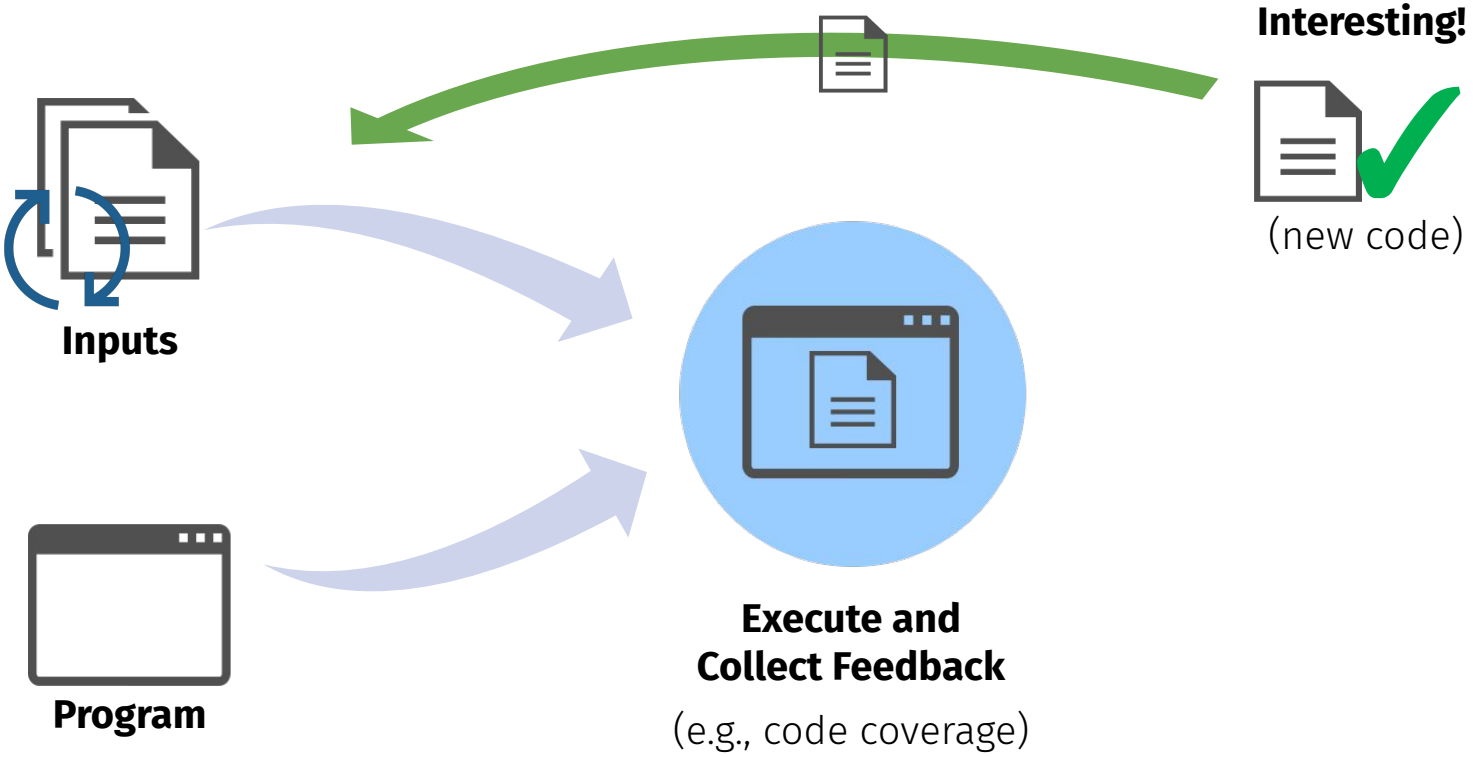
Inputs

Program

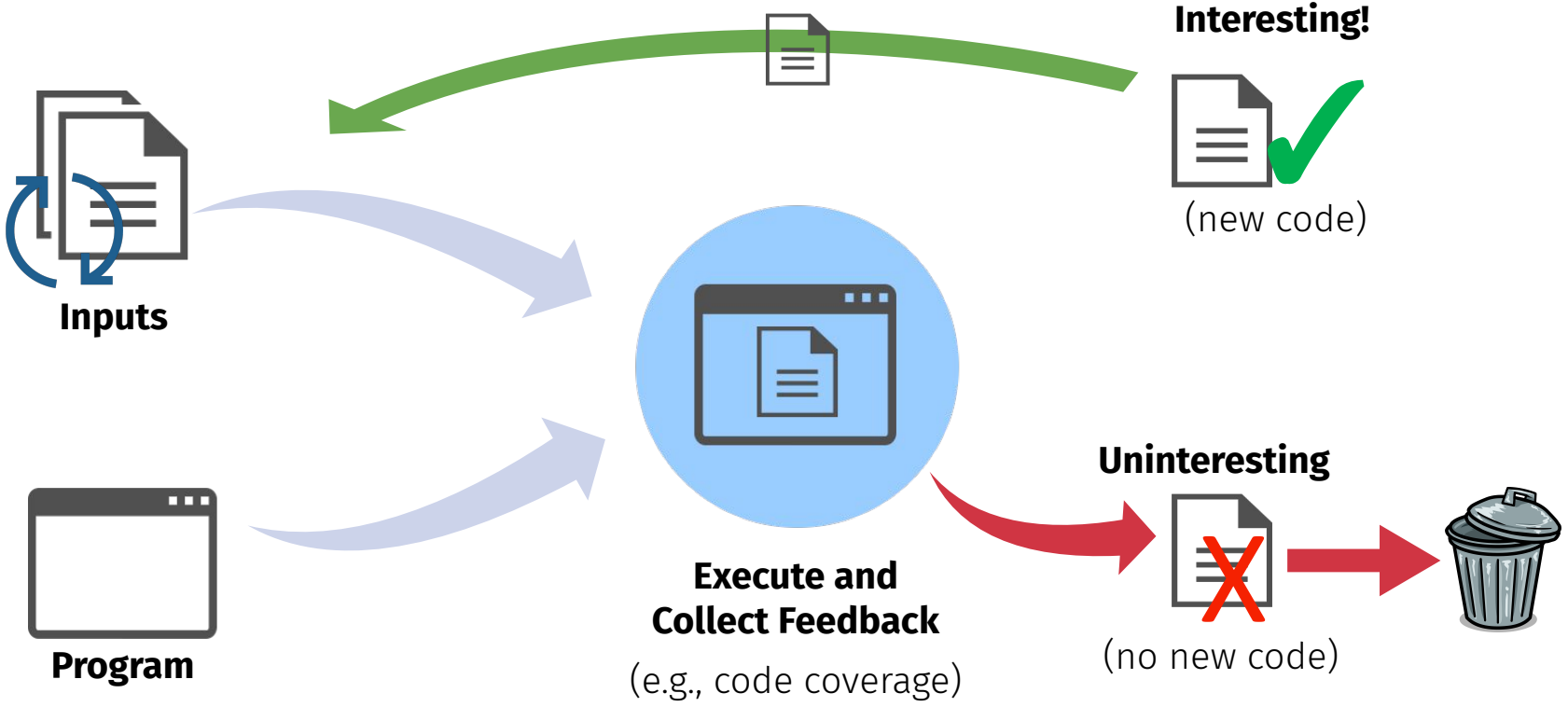
Key Approach: Fuzz Testing



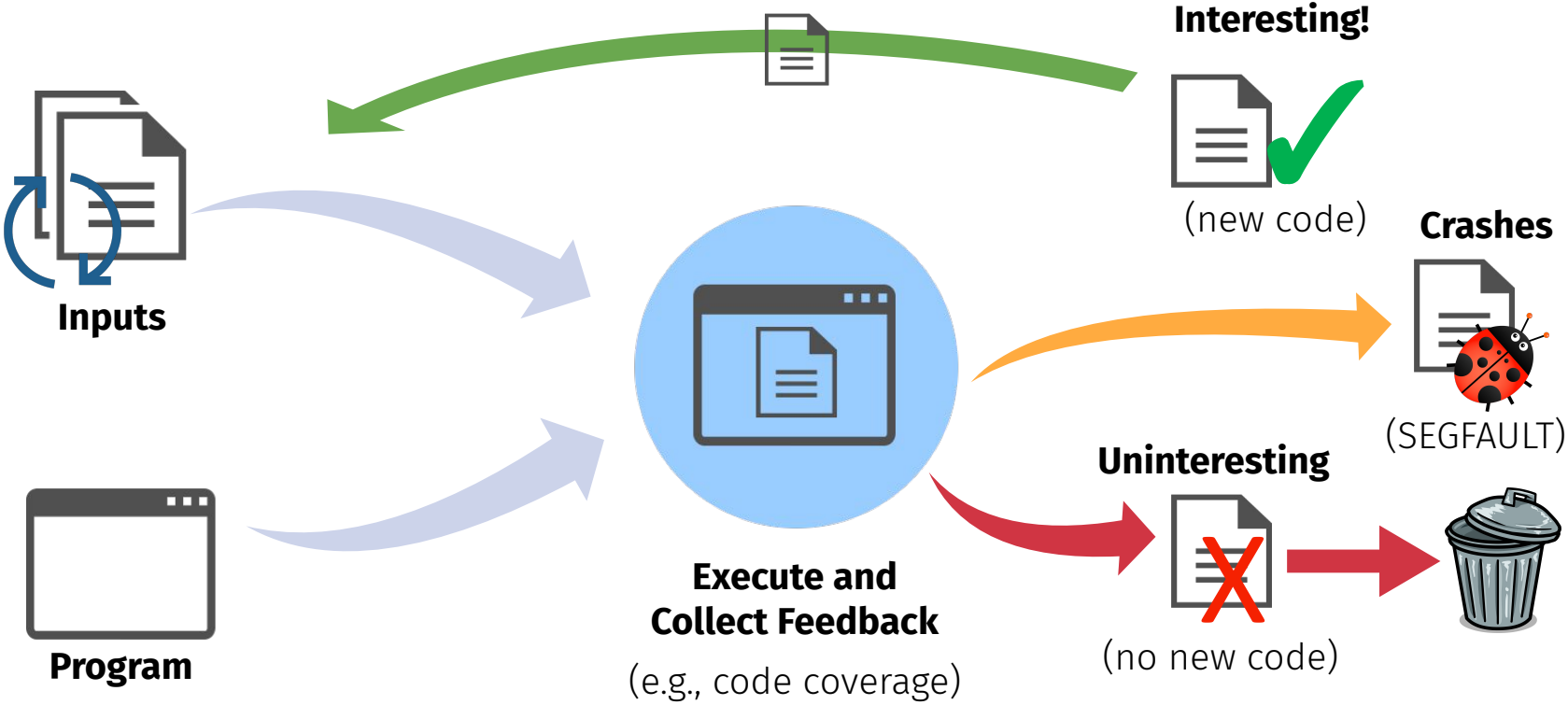
Key Approach: Fuzz Testing



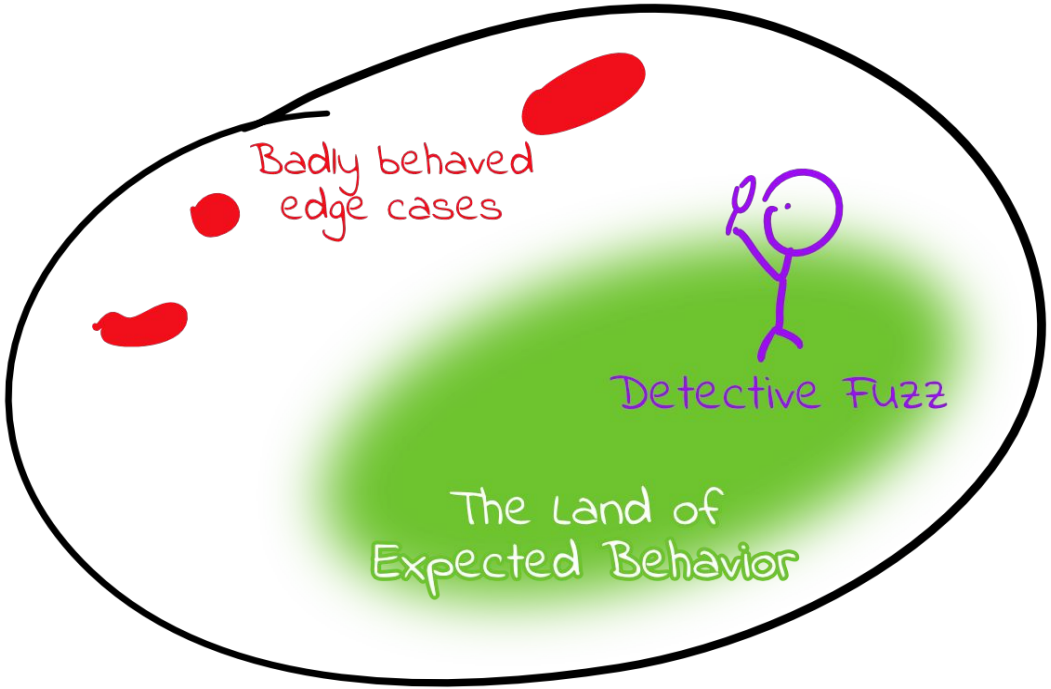
Key Approach: Fuzz Testing



Key Approach: Fuzz Testing



Key Approach: Fuzz Testing



Source: https://blog.trailofbits.com/2020/10/22/lets-build-a-high-performance-fuzzer-with-gpus/

Key Approach: Fuzz Testing

Google: We've open-sourced ClusterFuzz tool that found 16,000 bugs in Chrome

New fuzzing tool finds 26 USB bugs in Linux, Windows, macOS, and FreeBSD

Fuzzing continues to remain today's most **popular** and **successful** software security testing approach

Source: https://blog.trailofbits.com/2020/10/22/lets-build-a-high-performance-fuzzer-with-gpus/

My Research: Extending Fuzzing's Reach

Closed-source Binaries
Linux Binaries, Firmware
Windows, MacOS Binaries
Obfuscated Executables

Can closed-source code be fuzzed as well as open-source?

Prior Work:
Fast Coverage Tracing
Fast Process Execution

Code Dev/Analysis Tools
Compilers, Debuggers
Language Transpilers
Binary Analysis Tools

Where do these tools fail?
How can we find their bugs?

Ongoing Work:
Fuzzing Decompilers
Fuzzing Transpilers

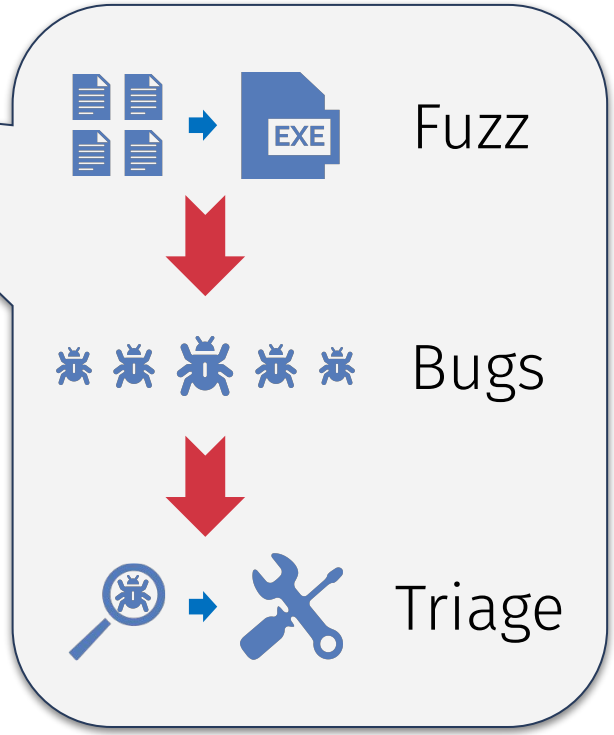
Complex Codebases
Applications, Kernels
Software Product Lines
Heterogeneous Software

What code aren't we fuzzing?
Are there bugs we are missing?

Ongoing Work:
Configuration Fuzzing
Automated Harnessing

Topics in this Course

Topics in this Course



- Input generation
- Runtime feedback
- Optimization
- Harnessing

- Sanitizers
- Bug oracles
- Property testing
- Differential testing

- Bug reporting
- Deduplication
- Root cause analysis
- Severity analysis

Questions?

Ethical Considerations

A Note on Ethics

NOTE: **Under no circumstances** may you **exploit or misuse** any bugs that you find (e.g., zero-day vulnerabilities) for unauthorized access or other illegal activity.

Violations of this policy will be referred to Student Conduct.

A Note on Ethics

NOTE: **Under no circumstances may you exploit or misuse** any bugs that you find (e.g., zero-day vulnerabilities) for unauthorized access or other illegal activity.

Violations of this policy will be referred to Student Conduct.

Our goals in this course are to **help devs & users**, **have fun**, and **learn**!

Questions / Professor AMA

Next time on CS 5963/6963...

Research 101: Ideas