

CPSC 539L: Topics in Programming Languages

Automated Testing, Bug Detection, and Program Analysis

September 5th, 2024

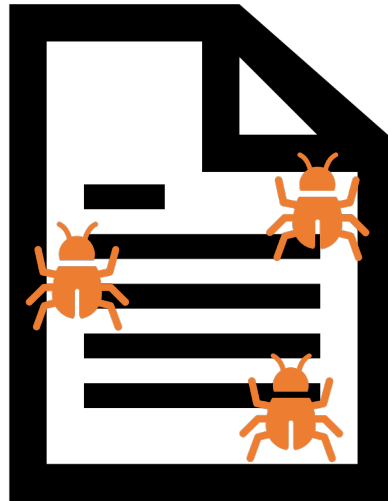
Instructor: Caroline Lemieux

Term: 2024W1

Class website: carolemieux.com/teaching/CPSC539L_2024w1.html

Sign up for class piazza: piazza.com/ubc.ca/winterterm12024/cpsc5391

Software Has Bugs



Bugs Have Increasing Consequences



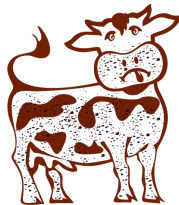
Bugs Have Increasing Consequences



Badlock



Cloudbleed



Dirty COW



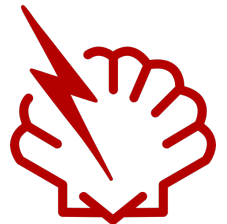
GHOST



Heartbleed



StageFright



ShellShock



Bugs in OpenSSL



Heartbleed

Severity: **7.5 HIGH**

Introduced: 14 Mar 2012

Discovered: 1 Apr 2014

Fixed: 7 Apr 2014

“... can be used to reveal up to 64k of memory to a connected client or server ...”

Costs:

- >\$500 million
- 30,000 X.509 certificates compromised
- 4.5 million patient records compromised
- CRA website shutdown, 900 SINs leaked
- ...

Bugs in OpenSSL



Heartbleed

Severity: 7.5 HIGH

Introduced: 14 Mar 2012

Discovered: 1 Apr 2014

Fixed: 7 Apr 2014

“... can be used to reveal up to 64k of memory to a connected client or server ...”

Costs:

- >\$500 million
- 30,000 X.509 certificates compromised
- 4.5 million patient records compromised
- CRA website shutdown, 900 SINs leaked
- ...

Bugs in OpenSSL



Heartbleed

Severity: **7.5 HIGH**

Introduced: 14 Mar 2012

Discovered: **1 Apr 2014**

Fixed: 7 Apr 2014

“... can be used to **reveal up to 64k of memory** to a connected client or server ...”

Costs:

- >\$500 million
- 30,000 X.509 certificates compromised
- 4.5 million patient records compromised
- CRA website shutdown, 900 SINs leaked
- ...

Bugs in OpenSSL



Heartbleed

Severity: **7.5 HIGH**

Introduced: 14 Mar 2012

Discovered: 1 Apr 2014

Fixed: 7 Apr 2014

“... can be used to reveal up to 64k of memory to a connected client or server ...”

Costs:

- >\$500 million
- 30,000 X.509 certificates compromised
- 4.5 million patient records compromised
- CRA website shutdown, 900 SIDs leaked
- ...



CVE-2016-6309

Severity: **9.8 CRITICAL**

Introduced: 22 Sep 2016

Discovered: 23 Sep 2016

Fixed: 26 Sep 2016

“... likely to result in a crash, however it could potentially lead to execution of arbitrary code ...”

Costs:

- minimal

Bugs in OpenSSL



Heartbleed

Severity: 7.5 HIGH

Introduced: 7 Apr 2014
Discovered: 7 Apr 2014
Fixed: 7 Apr 2014
by honggfuzz
(modern coverage-guided fuzzer)

“... can be used to reveal up to 64k of memory to a connected client or server ...”

Costs:

- >\$500 million
- 30,000 X.509 certificates compromised
- 4.5 million patient records compromised
- CRA website shutdown, 900 SINs leaked
- ...



CVE-2016-6309

Severity: 9.8 CRITICAL

Introduced: 22 Sep 2016

Discovered: 23 Sep 2016

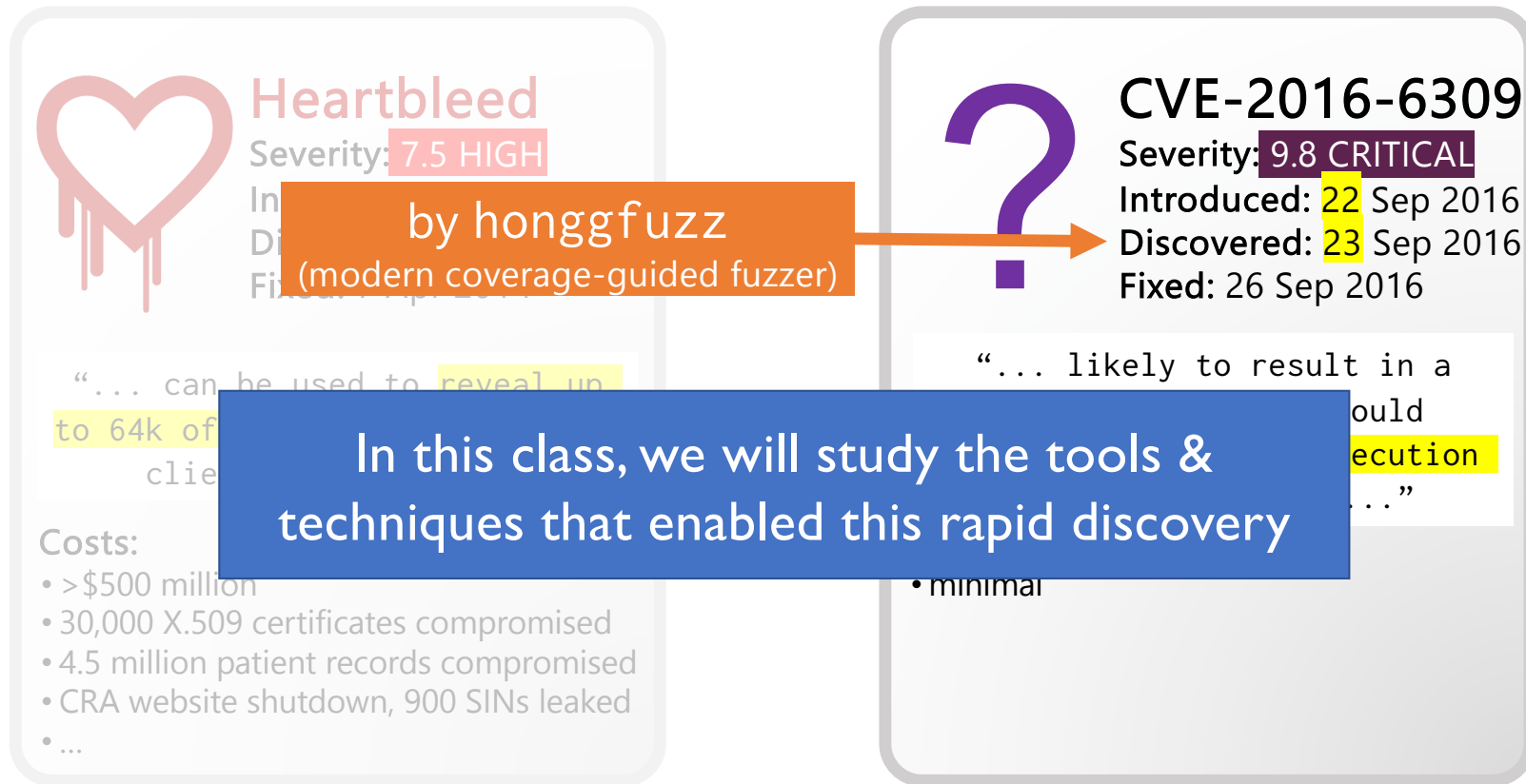
Fixed: 26 Sep 2016

“... likely to result in a crash, however it could potentially lead to execution of arbitrary code ...”

Costs:

- minimal

Bugs in OpenSSL



What this Course is

- A seminar-style course
 - We'll read and discuss papers
- A project-based course
 - You'll conduct a small research project (in groups unless you have good reason not to)
- A course whose goal is to give you solid foundations to **conduct research** in software testing & analysis

What this Course is Not

- **NOT:** An “*undergrad-style*” class, with well-structured assignments, quizzes, exams, etc.
 - Uncertainty! Exploration.
- **NOT:** A “*practical*” course for developing your industry-relevant software engineering + testing skills
 - Fuzz testing *is used* in industry, but this course focuses on research advances
- If you’re not looking for a **research-focused** course, this may not be the course for you
 - We have a few waitlisted students and have to set the course discussion lead schedule: drop ASAP if you think you’re headed there

Learning Objectives

- **Describe** the problems solved by the following techniques, **recognize** the scenarios in which each technique can be used, and **analyze** the pros and cons of each technique, for the following techniques:
 - Test-input generation: *blackbox fuzzing, greybox fuzzing, symbolic execution*
 - Test oracles: *crashing oracles, differential oracles, automated test-oracle generation*
 - *Property-based testing*
 - *Whole test-suite generation*
 - *Pattern-based Static Analysis*
 - *Fault Localization*
 - *Dynamic Data Race Detection*
- **Identify** contributions and room for improvement in existing research papers;
- **Debate** the positives and negatives of existing research papers;
- **Categorize** and **summarize** concerns and questions about existing research papers
- **Design** and **conduct** the experimental evaluation of a program testing or analysis tool.

Schedule for Today

- Introductions
- Class format/logistics
- What is automated testing, bug detection, program analysis?
- Black-box/Random Fuzzing
- TODOs for next time

Schedule for Today

- Introductions
- Class format/logistics
- What is automated testing, bug detection, program analysis?
- Black-box/Random Fuzzing
- TODOs for next time

Introductions

- Name, year, advisor/lab/research interests
- Why are you interested in this class?

Schedule for Today

- Introductions
- **Class format/logistics**
- What is automated testing, bug detection, program analysis?
- Black-box/Random Fuzzing
- TODOs for next time

Class Format

This class has 2 main components:

- Paper responses
- Course Project

Paper Responses

Before each class, you will read a research paper + post a response on Piazza

Paper responses should summarize the paper + your opinions to help spark discussion in class

In-class discussions of the paper will be led by a discussion lead (student). Goal of the discussion is to deepen the understanding + critical analysis of the subject matter.

Class Format

This class has 2 main components:

- **Paper responses** (45%)
 - (22.5%) Responses, due 22 hours before class
 - (15%) In class participation in discussions + Piazza participation
 - (7.5%) Discussion lead: read other students' responses and prepare to lead discussion in-class
- **Course Project**

Project

Open-ended, choose a topic related to automated testing, program analysis, bug detection.

Potential project types (see website for concrete suggestions):

- develop a new tool for testing in a particular domain
- tweaking an existing algorithm: evaluate effect of change
- an extensive re-evaluation of an existing tool
- a reimplementations of an algorithm in a new domain
- or creating a benchmark suite.

You may work in groups, or alone (only recommended if you have a topic closely connected to your research field)

Some Projects From Last Year

- Implementing JaCoCo as a coverage backend to [JQF](#).
(tweaking an existing algorithm: evaluate effect of change)
- Creating a tool to find NaN poisoning exploits in C/C++. Built on LLVM tooling.
(develop a new tool for testing in a particular domain)
- Creating an automated testing tool (based on property-based testing) for Haskell Rewrite Rules.
(develop a new tool for testing in a particular domain/an extensive re-evaluation of an existing tool)

Project

Proposal: describe background of the project, goal + intended deliverables, evaluation plan, timeline, division of work.

Check-in: 1 month in, update on any changes to plan since proposal

Writeup: background of the project, intended goal, what was accomplished, evaluation results, division of work

Presentation: summarize background and key achievements to class

Class Format

This class has 2 main components:

- **Paper responses** (45%)
 - (22.5%) Responses, due 22 hours before class
 - (15%) In class participation in discussions + Piazza participation
 - (7.5%) Discussion lead: read other students' responses and prepare to lead discussion in-class
- **Course Project** (55%)
 - (12.5%) Proposal: due Fri Oct 6 @ 5pm
 - (2.5%) Check-in: due Fri Nov 8 @ 5pm
 - (30%) Writeup: due Dec 6 @ 5pm
 - (10%) Presentation: week of Dec 3rd, length TBD

General Expectations

- You are (mostly) all graduate students
- You are here to learn, rather than achieve a good grade
- My goal is to assist in your learning, not
 - I understand if you are legitimately sick, have a conference to go to, etc.
 - I expect you to do your best to learn
 - Plagiarism and (improper) use of GenAI is against your own interests

Attendance + Late Policy

- This class relies on regular in-person attendance of discussions
 - Reading papers alone: ☹️
 - Discussing papers with others: 😊
- Paper responses must be submitted on-time, 2:30pm
- All other class deadlines are Fridays at 5pm
- If you anticipate being unable to attend class for some legitimate reason, please inform me ahead of time
 - *Especially* if you are signed up as discussion lead
- If you are sick, let me know, come back when you are better

Academic Honesty

- **Please review:** carolemieux.com/teaching/academic_honesty.html

Why Is Plagiarism Bad?

(a “too much text” slide)

From: carolemieux.com/teaching/academic_honesty.html

- **Learning outcomes.** Original writing is a key part of the research process. Part of the purpose of the written assignments (including responses) in this class is to give you practice in original writing. If you are not writing your own text, you are not getting practice writing. Further, *it is easy to copy text* from someone else's paper. It is much harder to understand the concepts in that text, and even harder to write them in your own words. Writing the key concepts of a paper in your own words conveys what you have understood—even if that understanding is only partial.
- **Values.** Academic honesty is a cornerstone of not only the values of this class, but of UBC, and the academic community as a whole. Part of the purpose of this class is to train you to conduct original research, and academic honesty is a *vital* part of that.
- **Consequences.** Because academic honesty is such a core value in the research community, there are serious consequences to misrepresenting another's work as your own. [Penalties for severe plagiarism](#) in ACM papers include ***retraction of the published work and 5-year publication bans in ACM publications***. This would have a devastating impact on both you and your collaborators' research career.

Academic Honesty

- **Responses:** You may discuss papers with other students, but write your paper responses alone. Do not read other students' responses on Piazza before submitting yours. Do not plagiarize text *from the paper* being reviewed either.
- **Project:** You may use other people's code as a base for your project; attribute the source of any piece of code outside of the main project. Do not plagiarize any text for your proposal + writeup: it should be written by you and your teammates alone.

Academic Honesty: GenAI

Paper Responses

- The goal of reading the papers is for you to learn to:
 - Effectively get the essentials out of a research paper
 - Communicate which parts of a paper you *don't* understand
 - Identify directions for research investigation
- Asking GenAI to summarize a whole paper will not meet these goals.
- Using GenAI for copyediting? Ok

Academic Honesty: GenAI

Project

- The goal of the project is for you to learn to:
 - Propose novel research questions in the realm of software testing/analysis
 - Plan and conduct an experimental analysis of software testing/analysis tooling
 - Give a research talk
 - Write a research report
- Coding: fine to use GenAI if it helps you
- Using GenAI for copyediting report? Ok
- Using GenAI to generate whole report sections? Not OK

Schedule for Today

- Introductions
- Class format/logistics
- **What is automated testing, bug detection, program analysis?**
- Black-box/Random Fuzzing
- TODOs for next time

Automated Testing \neq Test Automation

Test Automation

<https://www.functionize.com/automated-testing>

What is automated testing?

Automated testing refers to any approach that makes it possible to **run your tests without human intervention**. Traditional testing has been done manually. A human follows a set of steps to check whether things are behaving as expected. By contrast, an **automated test is created once** and then can run any time you need it.

For a long time, developers have automated their unit testing. That is, the tests that check whether a given function is working properly. Then automated testing frameworks like Selenium were developed. These allow modules or entire applications to be tested automatically.

These frameworks allow a test script to interact with your UI, replicating the actions of a user. For instance, they allow you to find a specific button and click it. Or locate a text entry box and fill it out correctly. They also allow you to verify that the test has completed correctly.



<https://www.atlassian.com/devops/devops-tools/test-automation>

What is test automation?

Test automation is the practice of automatically reviewing and validating a software product, such as a web application, to make sure it meets predefined quality standards for code style, functionality (business logic), and user experience.

Testing practices typically involve the following stages:

- **Unit testing:** validates individual units of code, such as a function, so it works as
- **Integration testing:** ensures several pieces of code can work together without u consequences
- **End-to-end testing:** validates that the application meets the user's expectation:
- **Exploratory testing:** takes an unstructured approach to reviewing numerous are application from the user perspective, to uncover functional or visual issues

The different types of testing are often visualized as a pyramid. As you climb up the the number of tests in each type decreases, and the cost of creating and running te increases.

https://en.wikipedia.org/wiki/Test_automation

Test automation

From Wikipedia, the free encyclopedia

See also: *Manual testing*



This article includes a list of general **references**, but it **lacks sufficient corresponding inline citations**. Please help to **improve** this article by **introducing** more precise citations. *(February 2009)* (*Learn how and when to remove this template message*)

In **software testing**, **test automation** is the use of **software** separate from the software being tested to control the execution of tests and the comparison of actual outcomes with predicted outcomes.^[1] Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place, or perform additional testing that would be difficult to do manually. Test automation is critical for **continuous delivery** and **continuous testing**.^[2]

This is not what we will cover in class

<https://www.functionize.com/automated-testing>

What is automated testing?

Automated testing refers to any approach that makes it possible to run your tests without human intervention. Traditional testing has been done manually. A human follows a set of steps to check whether things are behaving as expected. By contrast, an **automated test is created once** and then can run any time you need it.

For a long time, developers have automated their unit testing. That is, the tests that check whether a given function is working properly. Then automated testing frameworks like Selenium were developed. These allow modules or entire applications to be tested automatically.

These frameworks allow a test script to interact with your UI, replicating the actions of a user. For instance, they allow you to find a specific button and click it. Or locate a text entry box and fill it out correctly. They also allow you to verify that the test has completed correctly.

<https://www.atlassian.com/devops/devops-tools/test-automation>

What is test automation?

Test automation is the practice of automatically reviewing and validating a software product, such as a web application, to make sure it meets predefined quality standards for code style, functionality (business logic), and user experience.

Testing practices typically involve the following stages:

- **Unit testing:** validates individual units of code, such as a function, so it works as intended.
- **Integration testing:** ensures several pieces of code can work together without unexpected consequences.
- **End-to-end testing:** validates that the application meets the user's expectations.
- **Exploratory testing:** takes an unstructured approach to reviewing numerous areas of an application from the user perspective, to uncover functional or visual issues.

The different types of testing are often visualized as a pyramid. As you climb up the pyramid, the number of tests in each type decreases, and the cost of creating and running them increases.

https://en.wikipedia.org/wiki/Test_automation

Test automation

From Wikipedia, the free encyclopedia

See also: *Manual testing*

This article includes a list of general references, but it lacks sufficient corresponding inline citations.

Please help to improve this article by introducing more precise citations. (February 2009) (Learn how and when to remove this template message)

In software testing, **test automation** is the use of software separate from the software being tested to control the execution of tests and the comparison of actual outcomes with predicted outcomes.^[1] Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place, or perform additional testing that would be difficult to do manually. Test automation is critical for continuous delivery and continuous testing.^[2]

Automated Testing

- Test-input generation
 - *Generate test inputs that expose bugs in a program*
- Test case / Test Suite Generation
 - *Generate test suites that expose bugs in a program*

Automated Testing

- Test-input generation
 - *Generate test inputs that expose bugs in a program*
- Test case / Test Suite Generation
 - *Generate test suites that expose bugs in a program*

Randoop, EvoSuite: will cover later in class

Automated Testing

- Test-input generation

Fuzzing, Concolic + Symbolic Execution

- *Generate test inputs that expose bugs in a program*

- Test case / Test Suite Generation

- *Generate test suites that expose bugs in a program*

Test-Input Generation

- Assume a program P which takes in input i

Example Program P

```
def cgi_decode(s: str) -> str:
    """Decode the CGI-encoded string `s`: * replace '+' by ' ' * replace "%xx" by the character
    with hex number xx. Return the decoded string. Raise `ValueError` for invalid inputs."""
    t = ""
    i = 0
    while i < len(s):
        c = s[i]
        if c == '+':
            t += ' '
        elif c == '%':
            digit_high, digit_low = s[i + 1], s[i + 2]
            i += 2
            if digit_high in hex_values and digit_low in hex_values:
                v = hex_values[digit_high] * 16 + hex_values[digit_low]
                t += chr(v)
            else:
                raise ValueError("Invalid encoding")
        else:
            t += c
        i += 1
    return t
```


Example Program P

```
def cgi_decode(s: str) -> str:
    """Decode the CGI-encoded string `s`: * replace '+' by ' ' * replace "%xx" by the character
    with hex number xx. Return the decoded string. Raise `ValueError` for invalid inputs."""
    t = ""
    i = 0
    while i < len(s):
        c = s[i]
        if c == '+':
            t += ' '
        elif c == '%':
            digit_high, digit_low = s[i + 1], s[i + 2]
            i += 2
            if digit_high in hex_values and digit_low in hex_values:
                v = hex_values[digit_high] * 16 + hex_values[digit_low]
                t += chr(v)
            else:
                raise ValueError("Invalid encoding")
        else:
            t += c
        i += 1
    return t
```

Example Program *P*, Input *i*

Hello%21+World%22

```
def cgi_decode(s: str) -> str:
    """Decode the CGI-encoded string `s`: * replace '+' by ' ' * replace "%xx" by the character
    with hex number xx. Return the decoded string. Raise `ValueError` for invalid inputs."""
    t = ""
    i = 0
    while i < len(s):
        c = s[i]
        if c == '+':
            t += ' '
        elif c == '%':
            digit_high, digit_low = s[i + 1], s[i + 2]
            i += 2
            if digit_high in hex_values and digit_low in hex_values:
                v = hex_values[digit_high] * 16 + hex_values[digit_low]
                t += chr(v)
            else:
                raise ValueError("Invalid encoding")
        else:
            t += c
        i += 1
    return t
```

Example Program *P*, Input *i*

Hello%21+World%22

```
def cgi_decode(s: str) -> str:
    """Decode the CGI-encoded string `s`: * replace '+' by ' ' * replace "%xx" by the character
    with hex number xx. Return the decoded string. Raise `ValueError` for invalid inputs."""
    t = ""
    i = 0
    while i < len(s):
        c = s[i]
        if c == '+':
            t += ' '
        elif c == '%':
            digit_high, digit_low = s[i + 1], s[i + 2]
            i += 2
            if digit_high in hex_values and digit_low in hex_values:
                v = hex_values[digit_high] * 16 + hex_values[digit_low] t += chr(v)
            else:
                raise ValueError("Invalid encoding")
        else:
            t += c
        i += 1
    return t
```

Example Program P , Input i , Result $P(i)$

Hello%21+World%22

That input exercised the
code highlighted green

Returned normally

```
def cgi_decode(s: str) -> str:
    """Decode the CGI-encoded string `s`: * replace '+' by ' ' * replace "%xx" by the character
    with hex number xx. Return the decoded string. Raise `ValueError` for invalid inputs."""
    t = ""
    i = 0
    while i < len(s):
        c = s[i]
        if c == '+':
            t += ' '
        elif c == '%':
            digit_high, digit_low = s[i + 1], s[i + 2]
            i += 2
            if digit_high in hex_values and digit_low in hex_values:
                v = hex_values[digit_high] * 16 + hex_values[digit_low]
                t += chr(v)
            else:
                raise ValueError("Invalid encoding")
        else:
            t += c
        i += 1
    return t
```

Example Program P , Input i , Result $P(i)$

Hello%2V+World%22

That input exercised the
code highlighted green

Returned a **ValueError**

```
def cgi_decode(s: str) -> str:
    """Decode the CGI-encoded string `s`: * replace '+' by ' ' * replace "%xx" by the character
    with hex number xx. Return the decoded string. Raise `ValueError` for invalid inputs."""
    t = ""
    i = 0
    while i < len(s):
        c = s[i]
        if c == '+':
            t += ' '
        elif c == '%':
            digit_high, digit_low = s[i + 1], s[i + 2]
            i += 2
            if digit_high in hex_values and digit_low in hex_values:
                v = hex_values[digit_high] * 16 + hex_values[digit_low]
                t += chr(v)
            else:
                raise ValueError("Invalid encoding")
        else:
            t += c
            i += 1
    return t
```

Test-Input Generation

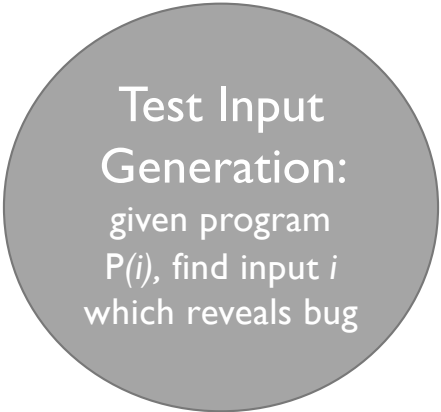
- Assume a program P which takes in input i
- Goal of automated Test-Input Generation:
 - Given P , generate inputs i which expose bugs

Test-Input Generation

- Assume a program P which takes in input i
- Goal of automated Test-Input Generation:
 - Given P , generate inputs i which expose bugs... or other interesting behaviors

Bug Detection

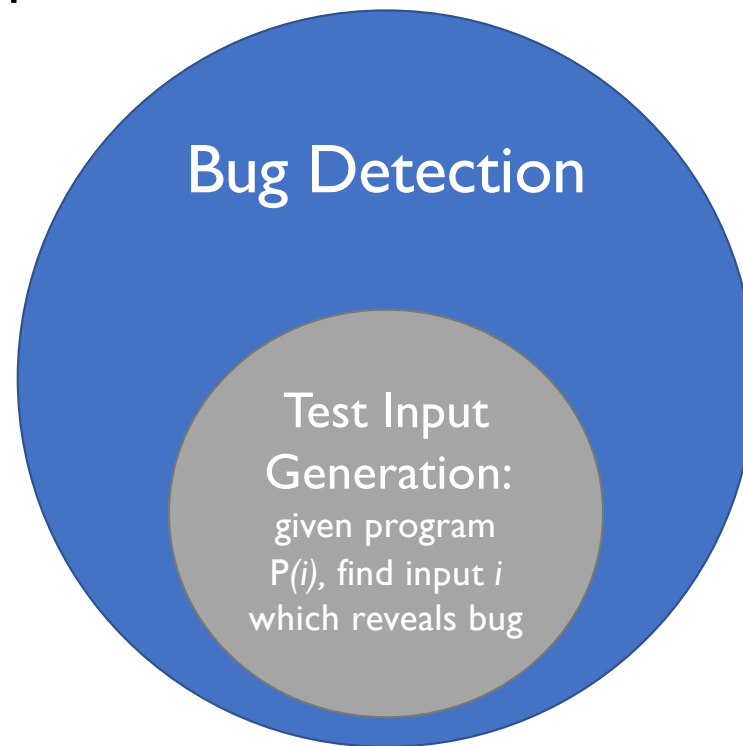
Broader than Test-Input Generation



Test Input
Generation:
given program
 $P(i)$, find input i
which reveals bug

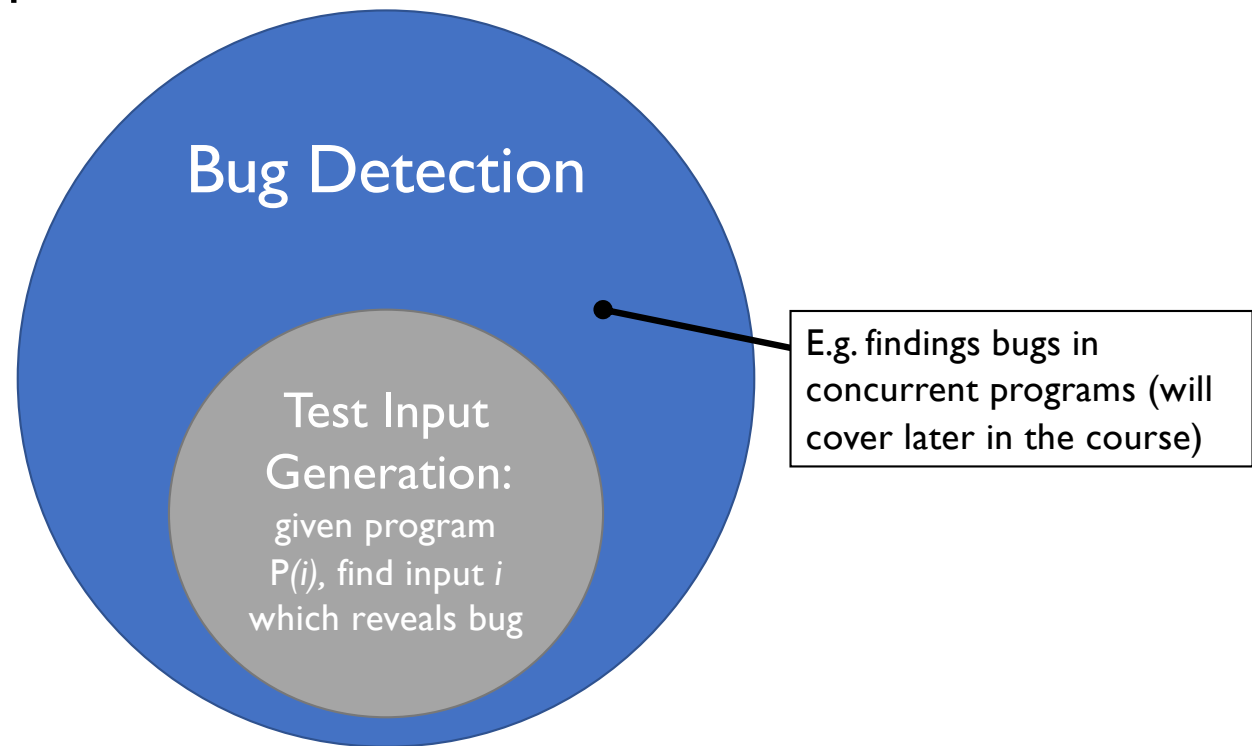
Bug Detection

Broader than Test-Input Generation



Bug Detection

Broader than Test-Input Generation



Program Analysis

Dynamic Analysis

Given a program P , and an input i , analyze P while it executes on input i : $analyze(P, i)$

e.g. taint analysis: which parts of the input i are used in different parts of the program?

Static Analysis

Analyze a program P independent of input i : $analyze(P)$

e.g. data flow analysis, pattern checking in your compiler

Schedule for Today

- Introductions
- Class format/logistics
- What is automated testing, bug detection, program analysis?
- **Blackbox/Random Fuzzing**
- TODOs for next time

What is Fuzzing/Fuzz Testing?

Aims to solve the test-input generation problem:

“Given program P generate inputs i which expose bugs or other interesting behaviors ”

Fuzzing algorithms are test-input generation algorithms where:

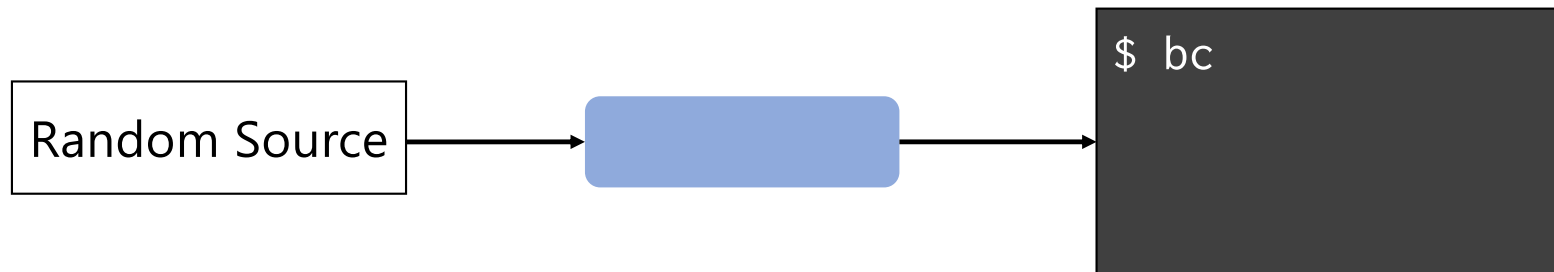
- Fuzzing algorithm has some elements of randomness
- Fuzzing algorithm may use feedback from program execution: $P(i)$ or $analyze(P,i)$ to guide the generation of the next input

Simplest: Random Fuzzing

Given a a program P , generate input i randomly.

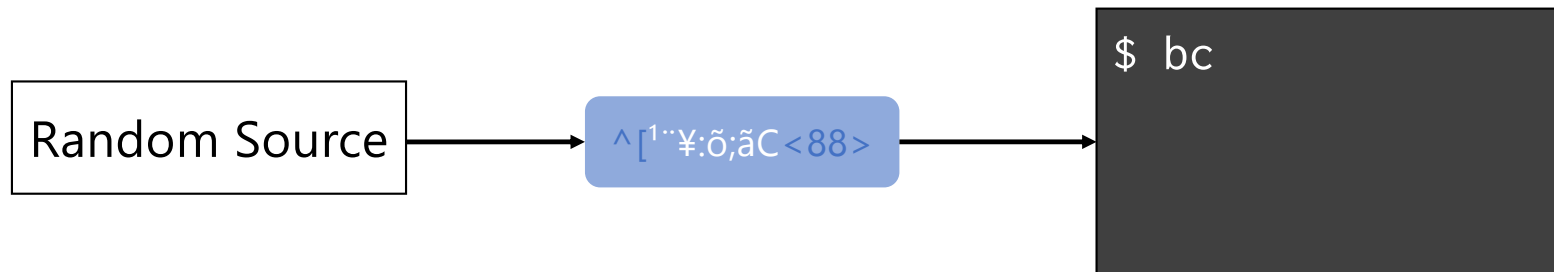
Called “blackbox fuzzing” because it is not using any feedback from the program under test $[P(i)$ or $analyze(P,i)]$ to guide input generation

Random Fuzzing



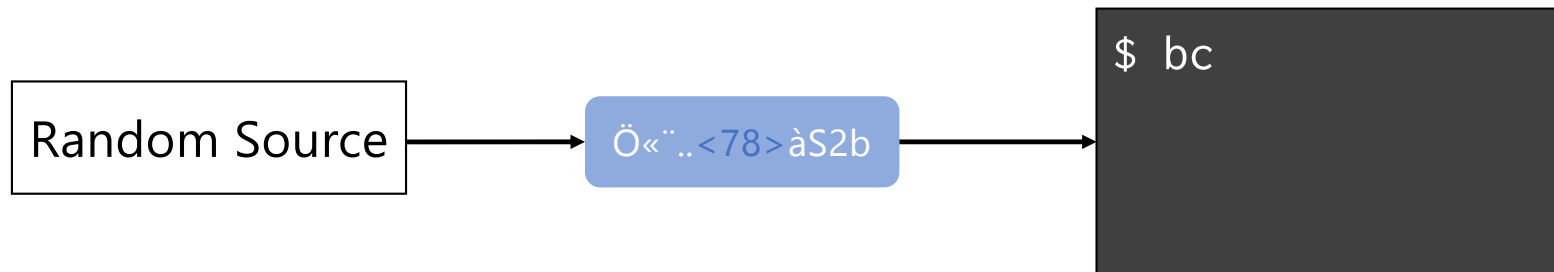
B. Miller, L. Fredriksen, B. So. *An Empirical Study of the Reliability of Unix Utilities*. Communications of the ACM, 1990.

Random Fuzzing



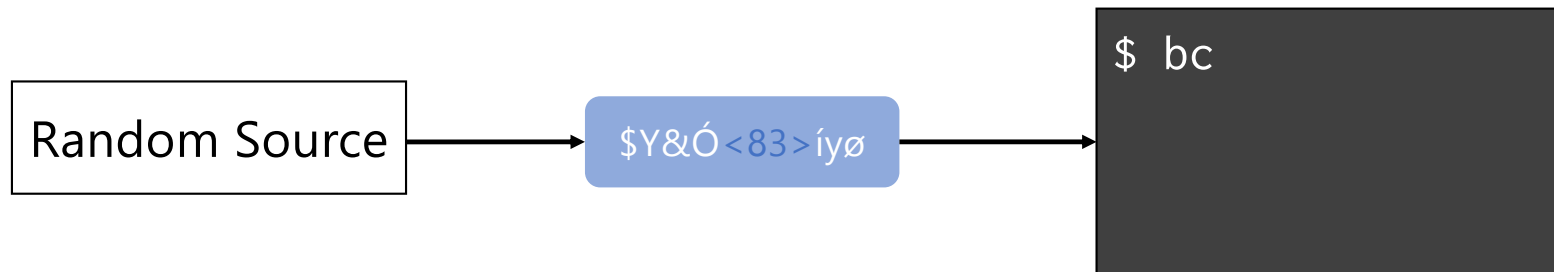
B. Miller, L. Fredriksen, B. So. *An Empirical Study of the Reliability of Unix Utilities*. Communications of the ACM, 1990.

Random Fuzzing



B. Miller, L. Fredriksen, B. So. *An Empirical Study of the Reliability of Unix Utilities*. Communications of the ACM, 1990.

Random Fuzzing



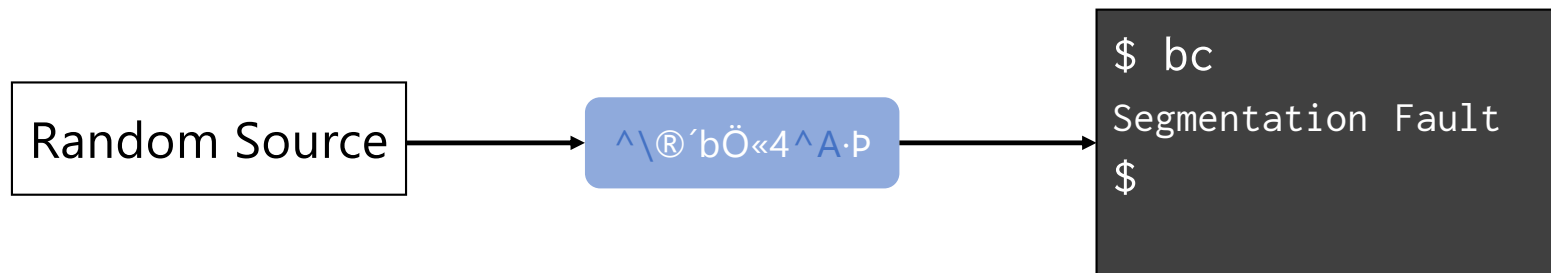
B. Miller, L. Fredriksen, B. So. *An Empirical Study of the Reliability of Unix Utilities*. Communications of the ACM, 1990.

Random Fuzzing



B. Miller, L. Fredriksen, B. So. *An Empirical Study of the Reliability of Unix Utilities*. Communications of the ACM, 1990.

Random Fuzzing



B. Miller, L. Fredriksen, B. So. *An Empirical Study of the Reliability of Unix Utilities*. Communications of the ACM, 1990.

Schedule for Today

- Introductions
- Class format/logistics
- What is automated testing, bug detection, program analysis?
- Black-box/Random Fuzzing
- **TODOs for next time**

TODOs

- Sign up for Piazza
 - *In this course, you will be using Piazza, which is a tool to help facilitate discussions. When creating an account in the tool, you will be asked to provide personally identifying information. Please know you are not required to consent to sharing this personal information with the tool, if you are uncomfortable doing so. If you choose not to provide consent, you may create an account using a nickname and a non-identifying email address, then let your instructor know what alias you are using in the tool.*
- On Piazza: Respond to sign-up for discussion lead
- Paper response for first paper due Monday 2:30pm