# Secure Software Design

Andey Robins

Spring 23 - Week 1

# Course Aims

*The aim of this course:* provide students with an understanding of the software design lifecycle, cybersecurity, and most importantly, the overlap between the two. Students will get hands on experience designing, writing, and maintaining applications with security included as an integral component.

# Outline

- Syllabus & Housekeeping
- Software Design History
- Security Design Motivation

# Disclaimer

In this course, you will learn the techniques and processes employed by skillful attackers. We learn this for the explicit purpose of developing more secure systems and subverting potential attacks. We discuss the situations and boundaries of ethical security practice in this course and you are held to this high standard. **Do not attack any system or information without explicit prior written permission.** Not only is it a really bad idea, it's probably illegal and a violation of UWYO network rules (UW Regulation 8-1) and the CEPS technology policy. I am not responsible for any actions you perform.

# Meeting Times

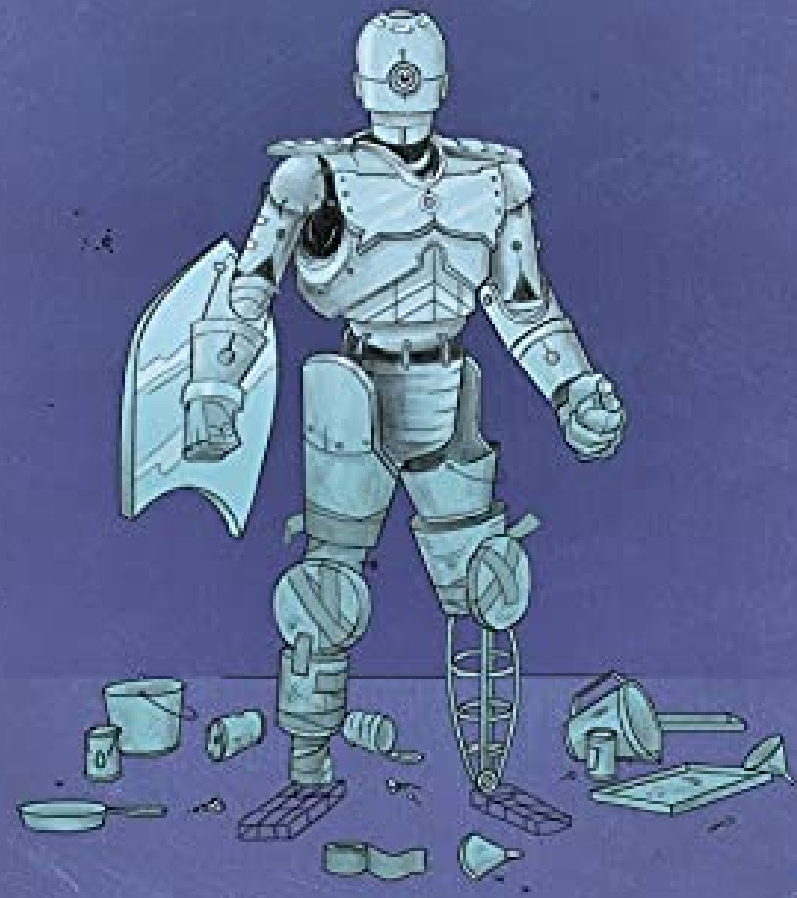
**Class:** MWF 8-8:50 am

**Office Hours:** M 11-12 & MR 1-2 & By Appointment

**Office:** EERB 228 (SSC Lab)

# Designing Secure Software

## A Guide for Developers



Loren Kohnfelder

no starch press

O'REILLY®

10th Anniversary Updated for Java 8

# Head First Design Patterns

## A Brain-Friendly Guide

Avoid those embarrassing coupling mistakes

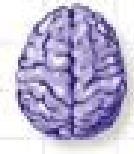
Discover the secrets of the Patterns Guru

Find out how Starbuzz Coffee doubled their stock price with the Decorator pattern



Learn why everything your friends know about Factory pattern is probably wrong

Load the patterns that matter straight into your brain





See why Jim's love life improved when he cut down his inheritance

Eric Freeman & Elisabeth Robson
with Kathy Sierra & Bert Bates

# Assignments

- Written Homework
- Programming Homework
- Midterm
- Final Project

# Structure

- Lecture presents concepts and gives examples
- Written homework assesses application of ideas
- Programming homework assesses ability to act on ideas

# Grading

| Category | 4010 Grading | 5010 Grading |
|---|---|---|
| Programming Homeworks | 50 pts. | 55 pts. |
| Written Homeworks | 20 pts. | 30 pts. |
| Midterm | 10 pts. | 15 pts. |
| Final Project | 20 pts. | 25 pts. |
| Total Points | 100 pts. | 125 pts. |

Figure 3: Grade Breakdowns per Section

# Late Work

- 75% credit up until the assignment is discussed in class

# 5010 Additional Topics

- Provable programs
- Zero-trust architecture
- Decentralized development
- Type theoretic security
- More/different as demanded (let me know if there are topics that sound interesting to you!)

Questions?

# The History of Software Engineering

# Outline

1. The Early Days (200 BCE to early 1980s)
2. The Era of Personal Computing (1983 - 1990)
3. Web 1.0 (1991 - 1999)
4. The Dot Com Crash (2000 - 2005)
5. The Agile Age (2006 - 2009)
6. The Microservice Era (2010 - 2014)
7. The Emerging Serverless Era (2015 - )

# The Early Days

- The antikythera mechanism (circa 200 BCE)
- Babbage Analytical Engine (1837)
- The first programmer (1842)
- tom kilburn and the greatest divisor of 2^18 (262,144) -> first program
- transistor (1950s)
- development of “programming” with first compilers and cobol/fortran
- waterfall development (1970)
- introduction of c (1972)
- apple II and the beginning of personal computing
  - led to the creation of software development

""" [The Analytical Engine] might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine. . . Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent. """ - Ada of Lovelace

# The Era of Personal Computing

- DRM introduced (1983)
- AT&T Breakup
- word & excel (mid 80s)
- ping-pong virus & cyberaids (1988)
- therac-25

# Web 1.0

- internet (1991)
- linux kernel (1991)
- Design Patterns (GoF) (1994)
- citibank hack (1995)
- php (1995)
- oracle/java (1996)
- mobile devices
  - first pdas (1996)

# The Dot Com Crash

- SaaS (1999)
- ILOVEYOU (2000)
- mafiaboy (2000)
- US vs Microsoft (2001)
- agile (2001)
- Gary McKinnon and Military hacking (2001-2002)
- AWS (2002)
- dot com crash
- Anonymous is first observed (2003)
- sandworm/voodoo bear is first observed (2004)
- ruby/rails (2004)

# The Agile Age

- jquery (2006)
- the iceman hacks (2006)
- first iphone (2007)
- node (2009)
- Vixen Panda is first observed (2009)
- The Lazarus Group is first observed (2009)

# The MicroService Era

- angular (2010)
- microservices (2012)
- electron (2013) -> just hit v22
- docker (2013)
- react (2013)
- heartbleed (2014)
- crypto, solidity, blockchain (2014)

# The Emerging Serverless Era

- serverless (2015)
- equifax breach (2017)
- PittyTiger is first observed (2017)
- svelte (2019)
- crypto bubble (2021)
- lastpass breach (2022)

# Why Design in Security

# Why Design in Security

- APTs
- Crackers & Piracy
- Breaches
- Modern DDoS
  - Azure DDoS of 3.47 Tbps for 15 mins ~3 Petabytes of data ($63k alone just to store)

""" The attacker used several networks to spoof 167 Mpps (millions of packets per second) to 180,000 exposed CLDAP, DNS, and SMTP servers, which would then send large responses to us. This demonstrates the volumes a well-resourced attacker can achieve: This was four times larger than the record-breaking 623 Gbps attack from the Mirai botnet a year earlier. """ over 6 months