# Computer Security: Attacks and Defenses

**Instructor: Kristopher Micinski**
**Number: 323**
**Prerequisites: Experience in C programming, CMSC245 at Haverford, CMSC246 at Bryn Mawr, experience w/ or willingness to learn new languages (Python, SQL)**
**Workload: 3 hours/week in class, 1 hour/week lab, approximately 6-10 hours/week outside of class. This will be a lab / project intensive class, and significant work outside of class is expected.**
**Cap: 25 (may be lifted to 35)**

# Course Overview

This course will serve as a broad introduction to the field of computer security, from two concurrent perspectives: attacks on systems, and defenses against those attacks. The goal of this course will be to help build intuition so that--when designing your own systems--you can intelligently assess and mitigate security risks.

To understand how attackers think, we will learn about the attacks they employ. We will dissect a number of real-world attacks (such as Heartbleed or WannaCry) and reflect upon what could have been done to prevent them. But understanding a collection of attacks is not alone sufficient for helping us understand how to build secure systems. So alongside attacks, we will also learn the theoretical underpinnings of security, and use it to build defenses into our systems.

Labs will transition theory into practice. We will conclude with a group project exploring advanced topics relevant to the state of the art in computer security. The course will begin with a discussion on ethical application of techniques we learn.

# Topics covered

We will cover parts of the following topics, adjusted for time and pace of the course, along with student interest in each area.

- Low-level memory attacks and defenses
    - Buffer overflows
    - Stack canaries
    - Access space randomization / derandomization
    - Return to libc / return-oriented-programming
- Cryptography
    - Symmetric and asymmetric-key cryptography
    - Certificates, CAs, and PKI
    - SSL / TLS
- Web security

    - SQL injections
    - Cross-site scripting
    - Cross-site request forgery
- Social engineering and security ethics
- UI design for security
    - App permissions design
    - Best practices for security UI
    - Permission lifetime and revocation
    - Case study in privacy controls:
        - Facebook privacy controls
        - Android permissions
- Information flow control in web apps
- Reverse engineering
- Theoretical underpinnings of security
    - Full abstraction
    - Information flow

# Projects and Labs

Projects will be started in labs, and then continued individually. Some labs are structured so that they begin with a concrete assignment to work on as an individual in the first week, and then move on to a group assignment to complete a larger task.

### Project 1: Memory attacks (Weeks 1-4 inclusive) (Uses C programming)

This project will cover low-level memory attacks using the C programming language. The students will begin by executing an attack from starter code provided. They will then implement their own buffer overflow attack, and demonstrate a way to prevent the attack by intelligent programming, and also facilities provided by the compiler. After completing this task, students will form groups to complete a more advanced attack studying ASLR or ROP. 1.5 weeks will be allocated for independent programming, and 1.5 weeks will be allocated for group work.

### Project 2: Cryptography (weeks 4-7 incl.) (Uses Python programming)

This project will involve creating a public / private key pair and manually exchanging keys to collaborate secretly communicate with group members. The next week, students will either implement a secure chat using cryptographic primitives provided, or explore an attack on an insecure cryptographic hash.

### Project 3: Web security (weeks 7-9 incl.) (Uses Python programming)

Students will be given an insecure web app written in Python which is vulnerable to an SQL injection attack. They will then craft an input which causes the app to leak secret information (in this case, student grades from a synthetic gradebook consisting of fictitious students). They will then fix this attack in the app. Finally, students will attempt to break other students' fixes.

**Final project (weeks 10-14):**
This will be a final project, either in a group or alone. Students requesting to work alone need prior approval for a topic and expectations will be calibrated accordingly. Students will select one of the following projects, or propose their own project:

- **Information flow specification (uses Python/Jeeves)**
  Implement privacy policies for a secure student grades database using Jeeves, an extension to the Python programming language.
- **Designing a privacy UI (uses Python / Javascript / etc..)**
  Use best practices to propose and implement a new UI for some privacy-related mechanism, and perform a mock implementation
- **Malware reverse engineering**
  Use reverse engineering tools to understand and discuss how a particular piece of malware works.
- **Implement signature-based antivirus**
  Students will read about and implement a variant of signature-based antivirus detection for a small sample of malware

Students will check in with the professor regularly, and collaboration will occur via Github.

# Grading

- **Labs and Projects: 50%**
  - **Individual components: 30%**
  - **Group components: 20%**
- **Two midterm exams: 30% (take home and open-note)**
  - **Given 1/3rd and 2/3rd of the way through the course**
- **Final (group) project: 20%**

Evaluation for group projects will be based on mutual student feedback and oral exam with individual group members.

# Books

**Required**

- **Security Engineering, Second Edition**, by Ross Anderson
  - This book is freely available online from the author
- Online resources will be distributed throughout the course. These include blog articles (e.g., by the Facebook privacy group), academic papers, and websites (e.g., the Android security internals). These will all be freely available.

**Optional**

- **The Web Application Hacker's Handbook**, by Dafydd Stuttard & Marcus Pinto
- **Applied Cryptography, Second Edition,** by Bruce Schneier