CptS 418/518: Software Security
Spring 2025

Course credits: 3
Meeting time: Thursday, 5:00 pm – 7:50 pm, January 22 – May 06
Location: O'brian Hall 25
Course webpage: https://chapering.github.io/teaching/cse418518

Instructor: Dr. Haipeng Cai
Office locations:
- Physical - 345, Davis Hall
- Online - Zoom (id: 7862337400 or https://buffalo.zoom.us/my/hcaiub)

Email: haipengc@buffalo.edu
Office hours: Thursday, 3:00 pm – 4:00 pm, or by appointments

---

Course Description
Software/systems security concerns the systematic process of protecting systems and software from cyber threats and attacks which may compromise their security properties. The core of this subject lies in the concepts and principles as well as methodologies and techniques for achieving those security properties for computer systems and software. The state of the art and practice of software/systems security defenses, including practical defense tools, will also be covered in this course. Of particular importance, learning software/systems security principles from this course will be complemented and enhanced by applying the principles to cyber security practice through diverse course projects and managed team collaborations.

Course Content Overview
This advanced course on cyber security addresses key aspects of cyber security with an emphasis on software and systems security. Accordingly, the core content includes concepts, principles, methodologies, and techniques on measuring and defending the various security properties of both the operating systems and application software. The technical approaches will cover source code analysis, binary analysis, dynamic random testing (fuzzing), and reverse engineering techniques, as applied in the context of vulnerability discovery, malware analysis, risk mitigation, and digital forensics. The central focus is on how to systematically assess, prevent, and contain security risks, threats, and attacks at all levels of the computer system (from architecture all the way up to application code). Students will study, in-depth, binary reverse engineering, vulnerability classes, vulnerability analysis, exploit development, and defensive solutions, etc. to understand how to crack and protect software and systems. A key part

of studying security is putting skills to the test in practice. In this class the progress of students is evaluated by not only understanding concepts and ideas but actually breaking/building software/systems from security perspectives.

Specifically, the key topics to be covered include the following:

- Background on software security (instruction set, memory map, file permissions, system calls, system utilities, build tools)
- Operating system security (architectures, mechanisms, hardening)
- Machine learning software security (adversarial robustness, poisoning)
- Reverse engineering (decompilation, disassembly, deobfuscation, data/algorithm recovery, malware analysis)
- Source code/binary analysis (static/dynamic/hybrid analysis, data/control flow analysis, information flow analysis, symbolic execution)
- Software vulnerabilities (stack/heap-based buffer overflow, integer overflow, format string vulnerabilities, return-oriented programming, data-oriented vulnerabilities, race conditions)
- Vulnerability discovery (fuzzing, crash dumps, side channels, equities, mitigations)
- Vulnerability exploitation (PoCs, exploits, shellcode)

In particular, the first three focus on concepts and principles of key facets of the holistic software security landscape, while the last five focus on practical techniques and tools for assuring software security.

Learning Outcomes and ABET Mapping (Computing Programs)
Students will gain knowledge and expertise on the core content of this course, which are to be embodied through their ability to define appropriate, specific security goals and develop actionable plans to realize the goals. Students will learn techniques that enable and empower software and systems security measurement, diagnoses, and improvement. In addition, students will also sharpen valuable skills necessary for software security defense practices, including the use of modern security tools, collaborating in a security defense team, and technical communication and peer evaluation. Students will participate in a semester-long group project to gain hands-on experiences applying the principles and techniques learned.

Specifically, the direct learning outcomes of this course include:

1. Identify various types of software vulnerabilities at source code and binary code level
2. Ethically exploit the discovered software vulnerabilities
3. Understand existing defensive mechanisms to defeat exploitation

Connections of ABET Computing Programs learning outcomes include:

- By analyzing the source code and/or binary of a program and looking for its weaknesses, students acquire an ability to analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
- By understanding the weaknesses of software, learning ethical hacking, and implementing the exploits, students acquire an ability to design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
- By articulating their exploitation methodologies and finishing the experiment reports, students acquire an ability to communicate effectively in a variety of professional contexts.
- By learning the history of real-world cyberspace incidents and their social and legal implications, students acquire an ability to recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.

Prerequisites

A prerequisite for undergraduate students is *CSE 220 Systems Programming*, whereas there is no prerequisite for graduate students. Students are expected to have a background in the C programming language. Solid background from the following classes helps significantly: (1) *CSE 341 Computer Organization*; (2) *CSE 365 Introduction to Computer Security*; and (3) *CSE 421/521 Operating Systems*.

Textbook

- No textbook is required; the lectures serve as the textbook

Suggested Readings

- Mitnick, Kevin. *The art of invisibility: The world's most famous hacker teaches you how to be safe in the age of big brother and big data*. Little, Brown, ISBN: 9780316380492. 2017.
- Hubbard, Douglas W., and Richard Seiersen. *How to measure anything in cybersecurity risk*. Wiley, ISBN: 9781119085294. (2016).
- Randal Bryant, David O'Hallaron. *Computer Systems: A Programmer's Perspective*, 3rd Edition.
- Dennis Andriesse. *Practical Binary Analysis: Build Your Own Linux Tools for Binary Instrumentation, Analysis, and Disassembly*.
- Laszlo Szekeres, Mathias Payer, Tao Wei, Dawn Song. *SoK: Eternal War in Memory*. IEEE Symposium on Security and Privacy, 2013.
- Yan Shoshitaishvili, Ruoyu Wang, Christopher Salls, Nick Stephens, Mario Polino, Andrew Dutcher, John Grosen, Siji Feng, Christophe Hauser, Christopher Kruegel, Giovanni Vigna. *SoK: (State of) The Art of War: Offensive Techniques in Binary Analysis*. IEEE Symposium on Security and Privacy, 2016.

- Hovav Shacham. *The Geometry of Innocent Flesh on the Bone*: *Return-into-libc without Function Calls (on the x86)*. ACM CCS, 2007.
- Tyler Bletsch, Xuxian Jiang, Vince W. Freeh, Zhenkai Liang. *Jump-Oriented Programming: A New Class of Code-Reuse Attack*. ACM AsiaCCS, 2011.
- Andrea Bittau, Adam Belay, Ali Mashtizadeh, David Mazieres, Dan Boneh. *Hacking Blind*. IEEE Security and Privacy 2014.

Grading
The final course grade will be calculated using the following breakdown and be converted from numeric numbers to letter grades using the scale mapping as follows.

Breakdown

Coursework	%
Participation	5%
Project	60%
Midterm exam	15%
Final exam	20%

Scale mapping

Score	*Grade*	Score	*Grade*	Score	*Grade*
>=93	*A*	[80,83)	*B-*	[66,70)	*D+*
[90,93)	*A-*	[77,80)	*C+*	[60,66)	*D*
[87,90)	*B+*	[73,77)	*C*	<60	*F*
[83,87)	*B*	[70,73)	*C-*		

Course Project
A key component of the coursework is a semester-long group project. The progress of this group project will be measured by deliverables. The objectives, requirements, and due date of each deliverable will be communicated well prior to the due date such that each group will have reasonably enough time to complete the tasks required for the deliverable. For each deliverable, each group will be required to submit a written report and/or code (along with test cases) to demonstrate the progress of the group. Each member of the group will initially receive the same credit based on the quality and timeliness of group submissions, and will be later adjusted according to in-group peer evaluation by the end of the project period. A list of sample project topics along with the concrete details on the deliverables will be given in a separate document (e.g., *project description*). Students are encouraged to choose different topics of their own, but should be approved by the instruction before starting the project.

Unless specified otherwise, each project deliverable shall be created and submitted electronically *as a single PDF* on Canvas prior to the deadline which is 11:59 pm of the posted due date. Any late submission will receive a zero if no notice is sent to the instructor about the expected delay ***prior to*** the deadline; if a notice is sent to the instructor prior to the deadline, then the penalty of the late submission is a flat *10% point deduction* for every day after the original due date until the submission is received or the point left becomes zero. Note that by the end of the semester, missing any project deliverable will lead to *an "I" (incomplete) grade* except for extenuating cases communicated to the instructor in advance -- the *I* grade for a student means the student does not complete the course. Students missing an exam by the end of the semester will also receive an 'I' grade for this course.

Expectations for 418 versus 518 students

This is a conjoined course for both undergraduate (418) and graduate (518) students. Coursework and learning objectives are common between these two student groups except for the following (mainly, the additional work 518 students are expected of):

- Topics on ML security and binary analysis mainly target 518 students
- 418 students are not expected to take on complex systems security topics for course projects like MLAdversary, which 518 students are expected to take
- 518 students are expected to give a technical presentation in addition to live demo during project reporting, while 418 students are not
- 418 students may work in groups of 4-6 members while 518 student groups should typically not be larger than 4 members in size

Communication

We will communicate announcements, assignments, lecture materials and other learning resources all on UB Learns. In particular, we will host off-class Q&A through Piazza. UB Learns is also the portal to be used for project deliverables submission and grading. For questions on course materials, lectures, and course project milestones, contact the instructor on Piazza by sending posts instead of by emails, so as to facilitate communication. You have options for sending *private (anonymous)* posts. Make sure you **subscribe to each** of the forums there so that you won't be missing important information about the course logistics and extended lecture discussions initiated by questions raised by other students.

Participation

Class attendance is required at all lectures. Although lecture slides and other supplementary learning materials will be posted online, these materials as well as the suggested reading materials are only used as references by the instructor in developing the lectures. Thus, studying these materials serves the purpose of getting better prepared for attending in-class lectures, but would by no means substitute for class attendance. Also, the course project requires each team member to be responsible and collaborative as well as to contribute equally; thus, missing lectures without justifiable reasons and then relying on other team members to catch up missed topics is not acceptable. You are also expected to participate in class discussions, which aids learning and provides valuable feedback on the lecture. If you know you will miss a lecture for a justifiable reason such as a university activity or a medical appointment, notify the instructor by email at least 12 hours before the lecture. While attendance will not be taken in every class, it will be sampled randomly at the discretion of the instructor. The basic participation credit that accounts for 5% of the final grade will be calculated using the sampled attendance records (in addition to active participation in classroom discussion).

In addition, students are expected to maintain a professional and respectful classroom environment, for which students are suggested to:

- silence personal electronics (non-disruptive ones may be used during class)
- arrive on time and attend the entire class session

Late Submission Policy

Late penalty is a flat 10% deduction per day. Late assignments may be turned up to one week after the original due date. An advanced notice must be given to the instructor via email at least 24 hours before the deadline for a late submission. The instructor may allow for late submissions without penalty if extenuating cases are explained in the notice email sent to the instructor.

Expected Effort

Beyond the time for lecture attendance, students in this class are expected to invest a minimum of 1-2 hours outside class for each lecture equivalent (or 2-4 hours per week), including the time for working on the course project.

Policy on the Use of LLMs/Generative AI

LLMs or any other forms of generative AI should *not be used for any graded assignments* (exams and projects milestones). For these assignments, what you submit must be absolutely your own work without involving anything from generative AI---you can use it for learning purposes outside your work on these assignments.

If there is evidence to suggest a substantial part of the assignment has been generated by generative AI, then it will be treated as a violation of Academic Integrity and necessary steps will be taken as per the academic integrity policy stated in the syllabus.

If you are in doubt of what is permitted and what is not, ask the instructor!

Academic Integrity

Academic integrity is critical to the learning process. It is your responsibility as a student to complete your work in an honest fashion, upholding the expectations your individual instructors have for you in this regard. The ultimate goal is to ensure that you learn the content in your courses in accordance with UB's academic integrity principles, regardless of whether instruction is in-person or remote. As an institution of higher learning, UB expects students to behave honestly and ethically at all times, especially when submitting work for evaluation in conjunction with any course or degree requirement. Thank you for upholding your own personal integrity and ensuring UB's tradition of academic excellence.

The academic integrity policy is available at https://buffalo.edu/academic-integrity and https://engineering.buffalo.edu/computer-science-engineering/information-for-students/graduate-program/cse-graduate-academic-policies/cse-academic-integrity-policy.html

Specifically for this course, students are allowed to discuss homework assignments. However, students are NOT allowed to share code, exploits, write-ups, and homework with each other. Plagiarism or any form of cheating in homework or exams is subject to serious academic penalty. All violations will be reported to the UB Office of Academic Integrity. There is a zero tolerance policy in this class.

- A minor academic integrity violation of a specific homework assignment (i.e., plagiarism on one question) may result in a 0 on that assignment.
- More serious violations (e.g., falsification) may result in deduction of the final grade and even an F or >F< on the final grade.
- The CSE department has a policy to upgrade the penalty to F on the final grade for a second academic integrity violation of any degree.

Syllabus Update
Information in the syllabus may be subject to change with reasonable advance notice.