CSC 472: Software Security

Fall 2026 • Section 01 • CRN 34672

Syllabus version: August 24, 2026

Instructor	Dr. Si Chen
Email	schen@wcupa.edu
Office	25 University Ave., Room 131
Department Phone	610-436-6998
Office Hours	Tue/Thu 12:30-2:00 PM (by appointment); Wed 1:00-3:00 PM
Credits	3
Campus	West Chester
Instructional Method	In Person (0-25% Distance Education)
Meeting Time	Tuesday / Thursday, 2:00 PM - 3:15 PM
Location	25 University Ave., Room 158
Term Dates	August 24 - December 12, 2026
Course Site	Course website: wcu.ninja/csc472 (slides, schedule, materials). D2L: submissions, grades, announcements.

1. Catalog Information and Course Description

Official Course: CSC 472 - Software Security (3 credits)

This course is primarily aimed at people interested in software security, reverse engineering, and low-level software. Students will explore the foundations of software security, including important software vulnerabilities and attacks that exploit them—such as buffer overflows, SQL injection, and session hijacking—and defenses that prevent or mitigate these attacks, including advanced testing and program analysis techniques.

Prerequisites: Successful completion of CSC 231 or CSC 242; and CSC 302.

Delivery: This section is scheduled in person. Distance education components, if any, will not exceed the section's published instructional-method designation.

Catalog reference: WCU CSC 472 catalog entry

Expected Technical Preparation

CSC 472 is an advanced course. Students are expected to enter with the following background:

- Basic programming concepts and systematic debugging skills.
- Working proficiency in C, including pointers, arrays, loops, functions, and memory-related behavior.
- Comfort with Unix/Linux, the command-line shell, and gdb or comparable debugging tools.
- Familiarity with Intel x86/IA-32 assembly language and basic computer architecture.
- Basic web and networking concepts, including HTML, HTTP, TCP, and client/server communication.

2. Textbook, Computing Environment, and Tools

Required textbook: None.

Reference books:

- Randal E. Bryant and David R. O'Hallaron, Computer Systems: A Programmer's Perspective, 3rd Edition, ISBN 978-0134092669.
- Kris Kaspersky, Hacker Disassembling Uncovered, 2nd Edition, ISBN 978-1931769648.
- Eldad Eilam, Reversing: Secrets of Reverse Engineering, 1st Edition, ISBN 978-0764574818.

Programming language: Python is used to construct attack, test, and automation scripts. Python libraries such as pwntools may be used where appropriate.

Course environment: Hands-on work will use the instructor's Badger CTF environment hosted in the Computer Science laboratories. Credentials and connection instructions will be provided. Students should be able to use SSH and a Linux shell.

Security-analysis tools: Course demonstrations and labs may use debuggers, disassemblers, binary-inspection utilities, and other software-security tools appropriate to the topic.

Hardware: You need a computer with a modern operating system and reliable Internet access that can connect to the course environment.

D2L: Unless otherwise announced, labs, reports, project materials, announcements, and grades will be distributed or submitted through D2L. The public course website, wcu.ninja/csc472, posts the schedule, lecture slides, and supporting materials; graded submissions and grades remain in D2L.

3. Course Student Learning Outcomes and Core Topics

Upon successful completion of CSC 472, students will be able to:

- Explain x86/IA-32 registers, byte ordering, calling conventions, stack frames, and system-call behavior relevant to vulnerability analysis.
- Read C and assembly code to identify software-security defects and reason about exploitability.
- Construct and test Python-based exploit scripts for vulnerable programs, including stack-overflow, return-oriented programming, and multi-stage exploit scenarios.
- Explain how common mitigations such as ASLR, NX/DEP, PIE, and stack canaries change exploit strategy and program behavior.
- Analyze additional vulnerability classes including format-string bugs, heap corruption, kernel-level memory-safety problems, and selected web-security flaws.
- Work effectively as a team to analyze vulnerable software and solve an integrated Capture-The-Flag (CTF) security challenge.

Program alignment: Course work emphasizes ABET-1 (analyzing complex computing problems) and ABET-2 (designing, implementing, and evaluating computing-based solutions), while the final project also requires effective team collaboration.

Primary Content Sequence

Unit	Topic
1	Introduction to software security and course environment
2	IA-32/x86 registers and byte ordering
3	x86 assembly
4	Stack and stack frames
5	System calls and shellcode
6	Stack overflow
7	Return-oriented programming (ROP)
8	Return-to-libc and ASLR
9	PLT, GOT, return-to-PLT, and GOT overwrite
10	Multi-stage exploits, StackGuard/canaries, and format strings
11	Heap overflow
12	Kernel overflow / kernel exploitation
13	Web security, including SQL injection and session hijacking

Unit	Topic
14	Integrated exploit development and CTF project

4. Assignments and Learning Activities

Five Hacking Labs: Hands-on individual labs require students to analyze a vulnerable target, develop and test an exploit or security analysis, and document reproducible technical results. Planned lab themes are Stack/Stack Frame, Stack Overflow, ROP, Multi-Stage Exploits, and Kernel Overflow.

Five Short Quizzes: Quizzes assess recent concepts, code/assembly reading, vulnerability reasoning, and the technical mechanisms covered in lecture and lab.

Final Team CTF Project: Students work in teams on a comprehensive Capture-The-Flag activity that integrates vulnerability analysis, exploit development, troubleshooting, and clear explanation of results.

In-Class Practice: Course meetings include code/assembly walkthroughs, debugger demonstrations, exploit tracing, and guided security-analysis exercises. Active participation contributes to attendance/participation credit.

5. Course Evaluation Policy and Grading

Evaluation	Weight	Planned Number
Attendance / Participation	5%	Ongoing
Short Quizzes	25%	5 (5% each)
Hacking Labs	50%	5 (10% each)
Final Team CTF Project	20%	1

Letter Grade Scale: A = 90-100 | B = 80-89 | C = 70-79 | D = 60-69 | F = 0-59

Feedback: Lab and quiz grades will be posted in D2L. Students should review posted grades promptly and contact the instructor about apparent errors. Lab feedback will emphasize exploit correctness, reproducibility, technical reasoning, and documentation.

6. Generative AI and Security-Tool Use

AI-assisted work is permitted and encouraged in this course unless a specific activity explicitly states otherwise. Software security increasingly involves AI-assisted code analysis, debugging, scripting, documentation, and hypothesis generation. The goal is to learn how to use these systems effectively while still developing the ability to validate security claims with concrete program behavior.

Permitted Uses of AI

- Explain C, Python, x86 assembly, calling conventions, stack frames, debugger output, and vulnerability mechanisms.
- Generate, modify, or troubleshoot Python/pwntools scripts, test harnesses, payload-building code, and small analysis utilities for labs and the final project.
- Suggest exploit hypotheses, debugging steps, breakpoint strategies, or interpretations of crashes and register/memory state.
- Help locate relevant documentation, summarize technical concepts, and improve the clarity of lab reports or project presentations.
- Compare alternative exploit approaches and help reason about how defenses such as ASLR, NX/DEP, PIE, and stack canaries affect them.

Technical Responsibility

- You remain responsible for every command, script, payload, explanation, and conclusion you submit. You should be able to explain and modify AI-assisted work.
- AI output is not evidence by itself. Security claims must be validated against the actual course target using debugger output, program behavior, successful exploit execution, or other reproducible evidence.
- AI systems can hallucinate addresses, offsets, instruction semantics, API behavior, and mitigation details. Treat generated output as a hypothesis to test, not as ground truth.
- Do not give external AI systems course credentials, private CTF flags, or another student's unpublished work.

Quizzes: AI tools are not permitted during a quiz unless that quiz is explicitly designated as open-AI. Labs and the final CTF project may use AI under the policy above.

Collaboration: AI use does not change the collaboration rules. Individual labs remain individual work; the final CTF project permits collaboration within the assigned team. Sharing another student's completed lab solution or private flags is not permitted.

7. Course Policies

Attendance and Participation

Regular, on-time, prepared attendance is expected. Attendance/participation is 5% of the course grade. University-excused absences will be handled consistently with WCU policy; students remain responsible for completing required academic work and communicating about missed work as early as practical.

Late Work

No credit will be given for unexcused late assignments. If a valid excuse affects a deadline, contact the instructor before the deadline whenever possible. D2L timestamps are authoritative for electronic submissions unless otherwise announced.

Lab Scope and Course Systems

For graded exploitation work, use the targets and infrastructure assigned for the course. Do not attack unrelated systems as part of a class activity. Report course-environment problems promptly rather than attempting to bypass infrastructure controls unrelated to the lab objective.

Email and Course Communication

Official course communication will use WCU email and D2L. Students are responsible for maintaining access to their WCU email account and checking D2L announcements. For technical help, include the lab/topic, relevant code or command, exact error/output, what you expected, and what you have already tried.

Course Mode and Pre-Course Work

CSC 472-01 is scheduled as an in-person course. No graded coursework is assigned before the first official day of the semester.

8. Tentative Course Schedule

The schedule may be adjusted to match class progress. D2L announcements are authoritative for lab due dates, quiz dates, and minor sequencing changes. The table below preserves the technical sequence of the prior CSC 472 syllabus while aligning it to the Fall 2026 calendar.

Week / Dates	Topics	Planned Evaluation / Activity	Calendar Notes
1 Aug 25, 27	Course orientation; software-security model; IA-32/x86 registers; byte ordering	Environment setup; diagnostic practice	First class Tue Aug 25
2 Sep 1, 3	x86 assembly; compiler-generated code; gdb workflow	Quiz 1; guided binary analysis	
3 Sep 8, 10	Calling conventions; stack; stack frames	Lab 1: Stack and Stack Frame	
4 Sep 15, 17	System calls; shellcode; payload structure	Shellcode practice	
5 Sep 22, 24	Stack overflow fundamentals; control-flow overwrite	Quiz 2; Lab 2: Stack Overflow	
6 Sep 29, Oct 1	Stack-overflow exploitation; NX/DEP; exploit debugging	Lab 2 work / review	
7 Oct 6, 8	Return-oriented programming (ROP); gadgets; chaining	Lab 3: ROP	
8 Oct 13, 15	Return-to-libc; ASLR; address-space reasoning	Quiz 3; ROP/ret2libc practice	No class Tue Oct 13 (Fall Break)
9 Oct 20, 22	PLT/GOT; return-to-PLT; GOT overwrite	Exploit-analysis exercises	Withdrawal deadline Oct 22
10 Oct 27, 29	Multi-stage exploits; StackGuard/canaries; format strings	Quiz 4; Lab 4: Multi-Stage Exploits	
11 Nov 3, 5	Heap overflow and heap-oriented vulnerability concepts	Heap analysis practice	
12 Nov 10, 12	Kernel overflow / kernel exploitation fundamentals	Lab 5: Kernel Overflow	
13 Nov 17, 19	Web security: SQL injection, session hijacking, related attack surfaces	Quiz 5; web-security practice	
14 Nov 24, 26	Final CTF architecture; integrated exploit chains; team planning	Final project / CTF preparation	No class Thu Nov 26

Week / Dates	Topics	Planned Evaluation / Activity	Calendar Notes
			(Thanksgiving)
15 Dec 1, 3	Integrated exploitation, defense review, and CTF work	Final Team CTF Project — completed & presented this week (due Thu Dec 3)	Last regular meetings; final-project week (no separate final-exam block)

9. Academic Integrity and University Policies

Academic Integrity

Students are expected to follow WCU academic-integrity requirements and the Computer Science Department's standards for academic honesty. In this course, permitted AI assistance is not itself an academic-integrity violation; however, misrepresenting another person's work as your own, unauthorized collaboration, copying another student's lab solution, or using prohibited resources on a quiz remains academic misconduct.

WCU Academic Integrity policy: Undergraduate Catalog - Academic Integrity

Office of Educational Accessibility (OEA)

West Chester University is committed to equitable access for students with disabilities. Students seeking accommodations should contact the Office of Educational Accessibility (OEA). Approved accommodations apply after the instructor receives the accommodation letter. Please share your approved accommodation letter as early as practical so arrangements can be made.

OEA: Wayne Hall 107 | 610-436-2564 | oea@wcupa.edu | OEA website

University-Sanctioned and Other Excused Absences

Notify the instructor in advance when possible about university-sanctioned events, required military service, or other absences covered by WCU policy. Appropriate documentation may be requested. The instructor will provide a fair opportunity to complete required work consistent with the applicable university policy.

Reporting Sexual Misconduct / Title IX

WCU faculty have reporting responsibilities for disclosures of sexual misconduct as defined by university policy. Students who need confidential support or current reporting/resource information should consult WCU's Title IX and sexual-misconduct resources.

Emergency Preparedness

Students are encouraged to enroll in WCU ALERT for official emergency notifications. In an emergency, follow university instructions and classroom/building procedures.

WCU ALERT

Final Project (in lieu of a final examination)

This course has no separate final examination. The Final Team CTF Project is the course's final assessment and is completed and presented during the last week of regularly scheduled classes (Week 15, December 1 and 3, 2026). There is no separate WCU final-examination block for this section.

APSCUF

I am a member of APSCUF, the Association of Pennsylvania State College and University Faculties. APSCUF supports excellence in teaching, scholarship, and service.

10. Classroom Learning Environment

Recording Policy

All classroom work, including lectures, discussions, demonstrations, and group activities, is not subject to public disclosure or distribution without express permission. Unauthorized recording of class is prohibited except when recording is an approved OEA accommodation. No recordings may be published, rebroadcast, or released without express permission.

Academic Freedom

Consistent with the collective bargaining agreement and principles of academic freedom, faculty are entitled to freedom to teach, discuss, research, and publish in their academic field. Classroom discussion may include relevant material that is challenging or controversial when it advances the subject matter, debate, and learning.

Respectful Ethical Communication and Anti-Doxxing

To maintain a safe and ethical learning environment, students may not share or post personal information about the instructor or classmates, such as home addresses or personal contact information, without express permission. Classroom conduct and use of course materials must comply with relevant university policies.

Critical Thinking and Academic Inquiry

This course is designed to support critical thinking: examining evidence, testing technical claims, comparing alternative explanations, asking questions, and forming conclusions from reproducible results. Grades are based on the stated assignment criteria, quality of reasoning, evidence, and technical correctness—not on whether a conclusion matches the instructor's preferred interpretation.

11. How to Succeed in CSC 472

- Work from evidence. When an exploit fails, inspect registers, memory, stack state, addresses, protections, and exact program output before changing the payload.
- Build exploits incrementally. First prove control, then prove the address or primitive, then add the next stage. Avoid debugging a long payload all at once.
- Keep a reproducible lab notebook: commands, offsets, addresses, environment assumptions, payload versions, and debugger observations.
- Use AI aggressively as a technical collaborator, but verify its claims in the actual target environment. A plausible explanation is not a working exploit.
- Ask for help early. Bring the command/script, exact output, target behavior, and the specific hypothesis you are testing.

This syllabus is a course plan. Reasonable adjustments may be announced in class and through D2L when needed to support course learning outcomes, lab progress, and the university calendar.