CSA 3350: Secure Programming Practices

Department of Natural and Behavioral Sciences, Computer Science Program
Computer Science-INTL
Fall 2026 (16-week term)

Meeting Days / Time: Online, anytime (asynchronous) Location: None (fully online) Section: W01 CRN: 12132
Modality: Online Anytime Term: Aug 24 - Dec 9, 2026

Faculty Information

Dr. Mainuddin Shaik

Email: mainuddin.shaik@sulross.edu

Office Hours: Email to schedule an appointment (in person or via Teams)

Response time: I reply to email within one business day, and I return feedback on each weekly submission within five days. Please use your Sul Ross email and put “CSA 3350” in the subject line.

Course Description

This course is about writing code that can be trusted, and being able to show why it can be trusted. Students take a single small web application that is riddled with realistic security flaws and, across the semester, find those flaws, fix them, prove the fixes hold with tests, audit what the application depends on, and assemble the evidence into a security assessment they defend on camera. The work is done in Python, in one version-controlled repository per student, so the semester’s history is the record of what was learned. By the end of the course, students will:

- Locate where untrusted data enters a program and reason about what each entry point may and may not assume.
- Identify, exploit, and repair common vulnerability classes: injection, broken access control, authentication and secrets failures, and unsafe dependencies.
- Write regression tests that prove a specific fix works and keeps working.
- Audit a project’s dependencies and configuration and justify what is fixed and what is accepted.
- Produce and defend an evidence-based security assessment of a codebase.

Prerequisites: CSA 1309 Computer Science I and a data structures course.

CSA 3350 is a programming course. Students write, read, and modify real Python throughout, and are expected to be comfortable with functions, files, control flow, and basic data structures before the first week.

Where this course sits. CSA 3350 is the applied, code-level counterpart to CSA 2372 Security Design and Information Assurance. CSA 2372 reasons about security at the level of an organization and its decisions. CSA 3350 takes the same ideas down to the level of source code: a trust boundary becomes a line of input validation, least privilege becomes a file permission and an authorization check, and evidence becomes a passing test and a coverage report. Taking CSA 2372 first is helpful but not required; this course re-introduces each concept as it lands on code.

Scope of the Course

This is a deliberately focused course, not a survey of everything called "security." What it covers is the set of mistakes that ordinary application programmers actually make and can actually fix.

What this course does not do: teach C or memory-safety vulnerabilities such as buffer overflows; teach the mathematics of cryptography; teach penetration-testing tools or offensive tradecraft; teach network, cloud, or container security; or prepare you for a certification exam. Those are real subjects and they are other courses.

What this course does do: build the habit and the skill of writing Python that resists the most common attacks, and of producing the evidence that it does.

Required Materials

Textbook: None required. The course runs on freely available material: the OWASP Top 10:2025, selected entries from the CWE Top 25, official Python security documentation, and instructor-authored labs.

The course application. A small, deliberately vulnerable Python web application is provided in Week 1. Every assignment in the course operates on this one application. You will not deploy it anywhere; it runs on your own machine or in a browser-based notebook.

Software and tools:

- Git. You keep one repository for the whole semester, in Google Colab or on your own machine, and commit your work to it as you go. Each contribution is submitted by zipping the repository folder including its `.git` directory and uploading that zip to Blackboard. Do not use `git archive`: it exports a snapshot without `.git`, so it strips the commit history this course grades. The commit history inside the zip is part of the assessment, so commit in real, honest steps rather than one commit at the end. Week 1 walks you through the minimal Git commands you need; no prior Git experience is assumed.
- Google Colab in a browser, the recommended route, with nothing to install. Python 3.11 or later on your own machine is a supported alternative if you prefer to work locally. Both are free.
- The command-line tools introduced in the course (`pytest`, `pip-audit`, `bandit`, `coverage`) are free and install with `pip`. Each is introduced with instructions in the week it is first used.
- The course application is a small Flask web application. You do not need prior Flask or web-development experience: Week 1 includes a short tour of how it is laid out, and you will be reading and making small, localized changes to it rather than building a web app from scratch.
- Blackboard Ultra for the weekly modules, announcements, grades, and video submissions.
- A way to record a short screen-and-voice video (Blackboard, Teams, or any screen recorder).

One rule, and it is an integrity rule, not a preference. The only system you attack, probe, or exploit in this course is the provided course application. Do not run any technique from this course against real websites, real institutional systems, your employer's systems, or any system you do not own. Doing so is both an academic-integrity violation under the policy below and, in many cases, a crime. The course application exists precisely so you never need a real target.

Important Dates (Fall 2026)

- Mon Aug 24: Course opens; Week 1 module available
- Mon Sep 7: Labor Day (university holiday; asynchronous work continues)

- Wed Sep 9: Twelfth class day (census); last day to drop without an academic record
- Fri Nov 6: Last day to withdraw with a grade of W (4:00 PM, Registrar's Office)
- Wed Nov 25 to Fri Nov 27: Thanksgiving holiday
- Wed Dec 2: Last day of classes
- Fri Dec 4, Mon Dec 7 to Wed Dec 9: Final examination period; final report and defense video due; grades finalized

This is an asynchronous course. Each weekly module opens on a Monday and its contribution is due the following Sunday at 11:59 PM unless the module states otherwise. All due dates are posted in Blackboard in Week 1.

Program Student Learning Outcomes

Upon completion, students will be able to:

- Define and explain fundamental concepts of computer science, including algorithms, data, abstraction, and systems.
- Identify and evaluate strategies for solving problems computationally.
- Analyze the security, privacy, and reliability implications of computing systems and the data they hold.
- Reason about the ethical, legal, and professional responsibilities that attach to computing work.
- Apply critical thinking to computational problems, systems, and case studies under real-world constraints.
- Communicate technical reasoning, and its limitations, to both technical and non-technical audiences.

Course Student Learning Outcomes

Upon successful completion of this course, students will be able to:

1. Map the trust boundaries of a program by identifying every point where untrusted input enters it and stating what that input may not be assumed to be.
2. Identify, exploit in a controlled setting, and repair injection vulnerabilities, including SQL injection and command injection.
3. Identify and repair broken access control, including insecure direct object references and missing authorization checks.
4. Identify and repair authentication and secrets-management failures, including insecure password storage and hardcoded credentials.
5. Write regression tests that fail against a known vulnerability and pass once it is fixed, and use them as evidence.
6. Audit a project's third-party dependencies for known vulnerabilities and justify remediation decisions under constraint.
7. Apply static analysis and coverage tools to a codebase and interpret their output, including their false positives and blind spots.
8. Produce and orally defend an evidence-based security assessment of a codebase.

Marketable Skills

Students will develop:

- **Secure Coding:** writing Python that validates input, handles secrets correctly, and fails safely.
- **Vulnerability Analysis:** finding and demonstrating security flaws in existing code.
- **Test-Driven Assurance:** proving that a fix works and does not regress.
- **Dependency and Configuration Hygiene:** auditing what a project relies on and how it is set up.
- **Technical Communication:** documenting and defending security decisions to a reader who was not there.

Controlled Vocabulary

These terms have fixed meanings in this course. Use them consistently in your writing and your commit messages.

Term	Meaning in this course
Trust boundary	A point in the code where data crosses from a less-trusted source into a more-trusted context, such as a web request entering a request handler
Entry point	A specific place in the program where external input arrives: a route parameter, a form field, a file upload, an environment variable
Input validation	Checking that incoming data matches what the code requires before the code acts on it
Injection	A vulnerability in which untrusted input is interpreted as code or commands, such as SQL or a shell command
Parameterized query	A database call in which data is passed separately from the query text, so data can never be read as SQL
Access control	The enforcement of who may perform which action on which resource
Insecure direct object reference	A flaw in which a user can reach another user's data by changing an identifier, because the code never checks ownership
Least privilege	Granting only the access a task requires, and no more
Secret	A credential or key that grants access and must not be exposed: a password, an API key, a token
Hashing	A one-way transformation used to store passwords so

Term	Meaning in this course
	the original cannot be recovered
Regression test	An automated test that fails against a specific defect and passes once it is fixed, guarding against its return
Dependency	A third-party package the program imports and relies on
Static analysis	Examining code for flaws without running it
Coverage	The share of code actually exercised by a test suite
Security assessment	A structured account of what was wrong in a codebase, what was changed, the evidence that it is fixed, and what risk remains
Evidence	Something that supports a security claim: a passing test, a tool report, a diff, a coverage number

Course Assignments and Grading

Assignment Type	% of Final Grade	Description
Weekly Contributions (C1 through C8)	55%	Eight cumulative deliverables built on the one course application: environment proof, entry-point map, injection finding, injection patch, regression tests, access-control patch and tests, secrets sweep, and dependency audit. Each builds on the last and lands in the same repository.
Midterm Walkthrough Video	10%	A five-minute recorded walkthrough of your own repository so far, explaining each change and why it is correct.
Final Security Assessment and Defense (C9)	25%	A written security assessment of the application as you left it, plus a five-to-seven-minute defense video answering three instructor questions.
Weekly Notes and Self-Checks	10%	Short reflections and self-check questions in each module. Completion-graded.

Assignment Type	% of Final Grade	Description
Total	**100%**	

Grading Scale:

A: 90 to 100 B: 80 to 89 C: 70 to 79 D: 60 to 69 F: below 60

Late Assignment Statement:

Contributions are due by 11:59 PM on the posted due date. Late submissions are accepted up to 48 hours past the deadline with a 10% deduction per day. Work submitted more than 48 hours late will not be accepted unless prior arrangements have been made.

Late work costs more in this course than the deduction suggests, because each contribution is the input to the next. A patch cannot be reviewed before the finding it fixes exists, and a test cannot prove a fix that has not been written. If you fall behind, contact me before the deadline, not after.

Because feedback is returned within five days and the next module opens the following Monday, you will sometimes start a contribution before you have my feedback on the previous one. That is expected. Build on your own submitted work as it stands; if my feedback later shows that an earlier contribution was off, correcting it and letting the correction flow forward is itself part of the work, and the change memo is where you record it. You are never blocked waiting on me.

Attendance and Participation:

This is an asynchronous course with no scheduled meetings. “Attendance” means steady weekly progress: opening the module, doing the work, and submitting on time. Because there are no class sessions to fall back on, the weekly modules and my written feedback on your submissions are the teaching. Read the feedback; the next contribution usually depends on it.

Grading Rubrics

Weekly Contributions rubric (applied to each of C1 through C8):

Criterion	Share	Standard
Correctness	40%	The finding is real, the patch actually closes it, or the test actually proves it. Verified by running your code.
Traceability to prior work	25%	The contribution builds on your own earlier work: a patch changes the lines your finding named, a test targets the flaw your patch fixed.
Reasoning and rationale	25%	You explain why the flaw is a flaw and why the fix is proportionate, in the course’s vocabulary.

Criterion	Share	Standard
Clarity and repository hygiene	10%	Readable diffs, honest commit messages, a repository someone else could follow.

Final Security Assessment and Defense rubric (25% of grade):

Criterion	Share
Completeness and accuracy of the assessment	40%
Quality of evidence (tests, tool reports, diffs)	25%
Honest treatment of residual risk and what remains unfixed	20%
Clarity of the written report and the spoken defense	15%

Defense protocol. You receive the three question categories and the rubric in advance. I ask about one fix and why it is correct, one flaw you chose not to fix and why, and one thing your tests do not cover. The defense is about your own repository, so there is nothing to memorize and nothing to ambush.

Course Schedule

The application and every tool are introduced in the week they are first needed. The schedule may be adjusted; changes are announced in Blackboard.

Week	Dates	Topic	OWASP / CWE anchor	Contribution and its dependency
1	Aug 24 - 30	Orientation. Get the course application running, run its test suite, tour what it does. Set up the repository.	—	**C1:** environment proof (test output, repository initialized) and a two-minute recorded tour of the application in your own words. **Due Sunday, September 6.**
2	Aug 31 - Sep 6	Trust boundaries in code. Where outside data enters the application, and what each entry point may not assume.	CWE-20 Input Validation	**C2: entry-point map.** Every route, parameter, form field, upload, and environment variable, with what each currently trusts. Every later contribution cites this map. **Due Sunday, September 13.**
3	Sep 7 - 13	Injection I: SQL and command injection, read and exploited in your own copy.	A05:2025 Injection; CWE-89, CWE-78	**C3: injection finding.** One injection flaw documented with file, line, a proof-of-concept input, and the harm to the organization, citing your C2 map. **Due Sunday, September 20.** Note: Labor Day Sep 7; census Sep 9.
4	Sep 14 - 20	Injection II: fixing it. Parameterized queries, safe subprocess use, allowlists.	A05:2025 Injection	**C4: the patch.** A diff that changes the exact lines C3 documented, plus a one-page fix rationale. **Due Sunday, September 27.**
5	Sep 21 - 27	Authentication and secrets failures I: plaintext passwords, weak sessions, the application's login.	A07:2025 Authentication Failures; CWE-522	Guided lab, with no new contribution opening. **C4 is due this week, Sunday, September 27.** C5 is due at the end of Week 6.
6	Sep 28 - Oct 4	Proving fixes: pytest as evidence. A test that fails on the old commit and passes on the new.	—	**C5: regression tests** for the injection patch and the authentication fix, with the failing run and the passing run both captured.
7	Oct 5 - 11	Access control I: insecure direct object references, the admin page, least privilege as code.	A01:2025 Broken Access Control; CWE-639	**Midterm walkthrough video:** five minutes over your repository so far, explaining each change and why it is right.
8	Oct 12 - 18	Access control II: fixing it. Authorization as a consistent pattern, not scattered checks.	A01:2025 Broken Access Control	**C6: access-control patch and tests**, with the same fail-then-pass evidence as C5.
9	Oct 19 - 25	Secrets and cryptographic failures:	A04:2025 Cryptographic Failures;	**C7: secrets sweep.** Find every secret

Week	Dates	Topic	OWASP / CWE anchor	Contribution and its dependency
		the hardcoded key, fake encryption, environment variables, hashing passwords correctly.	CWE-798	in the code, fix how each is stored, and hash the passwords properly.
10	Oct 26 - Nov 1	Errors and logging: the bare `except` that hides an attack, what to log and what must never be logged.	A10:2025 Mishandling Exceptional Conditions; A09:2025 Logging Failures	Guided lab; feeds C8 and the final report.
11	Nov 2 - 8	Software supply chain: auditing your own dependencies, the vulnerable pinned package, lockfiles.	A03:2025 Software Supply Chain Failures	**C8: dependency audit** using `pip-audit`, with the upgrade applied and a written justification for anything left unfixed. Note: last day to withdraw with a W, Nov 6.
12	Nov 9 - 15	Static analysis and coverage: `bandit` and `coverage`, and why the application's original green test suite proved nothing.	A08:2025 Data Integrity Failures	Tool outputs feed C9, plus a short reflection on what `bandit` caught, missed, and falsely flagged.
13	Nov 16 - 22	Misconfiguration and insecure design: debug mode, unsafe defaults, and the flaw that no patch can fix. Begin the assessment dossier.	A02:2025 Security Misconfiguration; A06:2025 Insecure Design	Drafting week for C9.
14	Nov 23 - 24	Final hardening. Full suite green, dossier assembled. Light week for the holiday.	—	Thanksgiving holiday Nov 25 to 27.
15	Nov 30 - Dec 2	The assessment and the defense.	—	**C9: security assessment report** and **final defense video** (five to seven minutes, answering three instructor questions).
Finals	Dec 4, 7 - 9	Final submission window.	—	Final report and defense video due; grades finalized.

The AI Connection

Generative AI tools are permitted in this course, with disclosure, and they are useful for exactly the kind of code you will be working with. They are also, routinely, wrong about security in confident and specific ways: they suggest fixes that do not close the flaw, invent reassurances a test would disprove, and recommend dependencies with known vulnerabilities.

That is why the assessment is built the way it is. Your repository history and your two walkthrough videos are what make the work yours. A patch you did not understand will not survive the question “why is this correct?” on camera, and a test you did not write will not fail on the right commit. Use the tools, disclose them, and verify everything they hand you, because in this course verification is the actual skill.

Generative AI Use

University Statement (required):

The University does not recommend or endorse any specific AI tools or resources. Students should be aware that many generative AI tools (e.g., ChatGPT, Google Gemini, Microsoft Copilot) store user input and may use this data to train future models. For this reason, students should never upload or share personal, confidential, or identifiable information, such as names, ID numbers, health data, or assignment submissions containing such details, into any generative AI platform. When using AI tools, students should verify whether the tool complies with student privacy standards as indicated by the University. Faculty may recommend specific tools that better align with institutional data privacy policies, but ultimate responsibility for data protection rests with users. Students are encouraged to use faculty-recommended platforms when engaging in coursework involving generative AI. The University is not liable for any adverse experience or impact when students interact with these tools.

Course Policy: Generative AI is permitted with disclosure.

You may use generative AI on any assignment in this course when you disclose the use and remain responsible for the correctness of what you submit. Each contribution includes a brief AI-use statement identifying (a) whether AI was used, (b) which tool, (c) for what purpose, (d) what you accepted, modified, or rejected, and (e) how you verified it. “No AI was used” is a complete and acceptable statement.

Never enter into an AI tool any real credential, real system detail, or personal or protected information. The course application is fictional and safe to discuss; nothing else about a real system is.

The disclosure statement is not graded and is not a detection mechanism. What holds the course together is that every contribution builds on your own prior commits and that you explain your own work on camera twice. Code you cannot defend is code you cannot pass with.

You remain responsible for every line submitted under your name, and you must be able to explain and defend it. Fabricated evidence, a test that does not test what it claims, undisclosed substantive use, or misrepresented authorship falls under the Academic Integrity policy below.

ADA Statement

SRSU Accessibility Services. Sul Ross State University (SRSU) is committed to equal access in compliance with the Americans with Disabilities Act of 1973. It is SRSU policy to provide reasonable accommodations to students with documented disabilities. It is the student’s responsibility to initiate a request each semester for each class. Students seeking accessibility/accommodations services must contact Mrs. Mary Schwartze Grisham, LPC, SRSU’s Accessibility Services Director, or Ronnie Harris, LPC, Counselor, at 432-837-8203 or email

mschwartze@sulross.edu or ronnie.harris@sulross.edu. RGC students can also contact Alejandra Valdez at 830-758-5006 or email alejandra.valdez@sulross.edu. Our office is located on the first floor of Ferguson Hall, room 112, and our mailing address is P.O. Box C122, Sul Ross State University, Alpine, Texas, 79832.

Accessible Paths in This Course

Every assessment in this course is already asynchronous and self-paced. The two video contributions may be submitted as screen recordings with voice-over, audio-only narration over a screen capture, or, where an approved accommodation requires it, a written walkthrough of equivalent depth. Choosing an alternative format for the videos requires no documentation.

Student Responsibilities Statement

All full-time and part-time students are responsible for familiarizing themselves with the Student Handbook and the Undergraduate & Graduate Catalog and for abiding by the University rules and regulations. Additionally, students are responsible for checking their Sul Ross email as an official form of communication from the university. Every student is expected to obey all federal, state, and local laws and is expected to become familiar with the requirements of such laws.

SRSU Distance Education Statement

Students enrolled in distance education courses have equal access to the university's academic support services, such as library resources, online databases, and instructional technology support. For more information about accessing these resources, visit the SRSU website.

Students should correspond using Sul Ross email accounts and submit online assignments through Blackboard, which requires a secure login. Students enrolled in distance education courses at Sul Ross are expected to adhere to all policies pertaining to academic honesty and appropriate student conduct, as described in the student handbook. Students in web-based courses must maintain appropriate equipment and software, according to the needs and requirements of the course, as outlined on the SRSU website. Directions for filing a student complaint are located in the student handbook.

Counseling

Sul Ross has partnered with TimelyCare, where all SR students have access to nine free counseling sessions. You can learn more about this 24/7/365 support by visiting TimelyCare/SRSU. The SR Counseling and Accessibility Services office will continue to offer in-person counseling in Ferguson Hall room 112 (Alpine campus), and telehealth Zoom sessions for remote students and RGC students.

Libraries

The Bryan Wildenthal Memorial Library and Archives of the Big Bend in Alpine offer FREE resources and services to the entire SRSU community. Access and borrow books, articles, and more by visiting the library's website, library.sulross.edu. Off-campus access requires logging in with your LoboID and password. Librarians are a tremendous resource for your coursework and can be reached in person, by email (srsulibrary@sulross.edu), or by phone (432-837-8123).

No matter where you are based, public libraries and many academic and special libraries welcome the general public into their spaces for study. SRSU TexShare Cardholders can access additional services and resources at various libraries across Texas. Learn more about the TexShare program by visiting library.sulross.edu/find-and-borrow/texshare/ or ask a librarian by emailing srsulibrary@sulross.edu.

Mike Fernandez, SRSU Librarian, is based in Eagle Pass (Building D-129) to offer specialized library services to students, faculty, and staff. Utilize free services such as InterLibrary Loan (ILL), ScanIt, and Direct Mail to get materials delivered to you at home or via email.

Academic Integrity

Students in this class are expected to demonstrate scholarly behavior and academic honesty in the use of intellectual property. Students should submit work that is their own and avoid the temptation to engage in behaviors that violate academic integrity, such as turning in work as original that was used in whole or part for another course and/or professor; turning in another person's work as one's own; copying from professional works or internet sites without citation; or collaborating on a course assignment, examination, or quiz when collaboration is forbidden. Use of generative AI tools is governed by the course AI policy above. Violations of academic integrity can result in failing assignments, failing a class, and/or more serious university consequences. These behaviors also erode the value of college degrees and higher education overall.

One rule specific to this course. The only system you may attack, probe, exploit, or test with the techniques taught here is the provided course application. Applying these techniques to any real system you do not own is an academic-integrity violation and, in most cases, a violation of state and federal law. The course application exists so that you never need, and are never permitted, a real target.

Classroom Climate of Respect

Importantly, this class will foster free expression, critical investigation, and the open discussion of ideas. This means that all of us must help create and sustain an atmosphere of tolerance, civility, and respect for the viewpoints of others. Similarly, we must all learn how to probe, oppose, and disagree without resorting to tactics of intimidation, harassment, or personal attack. No one is entitled to harass, belittle, or discriminate against another on the basis of race, religion, ethnicity, age, gender, national origin, or sexual preference. Still, we will not be silenced by the difficulty of fruitfully discussing politically sensitive issues.

Supportive Statement

I aim to create a learning environment for my students that supports various perspectives and experiences. I understand that economic disparity, health concerns, or unexpected life events may impact the conditions necessary for you to succeed. My commitment is to be there for you and help you meet the learning objectives of this course. I do this to demonstrate my commitment to you and to the mission of Sul Ross State University to create a supportive environment and care for the whole student as part of the Sul Ross Familia. If you feel like your performance in the class is being impacted by your experiences outside of class, please don't hesitate to come and talk with me. I want to be a resource for you.

Where This Course Sits in the Sequence

Four ideas run across CSA 1309, CSA 2372, and CSA 3350, and this course is where they land on code.

Trust boundary. CSA 1309 introduces the function and module boundary. CSA 2372 owns the system and organizational trust boundary. CSA 3350 makes the trust boundary concrete: it is the line in a request handler where input validation happens or fails to happen.

Least privilege. CSA 1309 does not name it. CSA 2372 owns it as a design principle and an access-control model. CSA 3350 implements it in file permissions, authorization checks, and API scopes.

Failure and correctness. CSA 1309 introduces debugging and test cases. CSA 2372 owns fail-safe defaults and residual risk. CSA 3350 implements it in error handling, regression tests, and defensive coding.

Evidence. CSA 1309 does not name it. CSA 2372 owns it as the assurance argument. CSA 3350 makes it literal: a passing test, a coverage report, a static-analysis result, and a diff.

If you have taken CSA 2372, this course is where its arguments become code you can run. If you have not, each idea is re-introduced here at the level of the line you are editing.

Reference Links

- OWASP Top 10:2025: https://owasp.org/Top10/2025/
- 2025 CWE Top 25 Most Dangerous Software Weaknesses: https://cwe.mitre.org/top25/
- OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/
- Python security documentation: https://docs.python.org/3/library/security_warnings.html
- pip-audit: https://pypi.org/project/pip-audit/
- Bandit static analyzer: https://bandit.readthedocs.io/