**CS 405 Journal: Don’t Leave Security to the End**

Kain Mason

**Explain what the following statement means as a best practice in secure coding: “Don’t leave security to the end.”**

It means security is a design constraint, not an afterthought. If I try to bolt security on at the end, I’m patching symptoms instead of preventing root causes. Building it in early—requirements, design, and code—changes the defaults: inputs are validated, least‑privilege is planned, and dangerous patterns never ship. From a DevSecOps view, this shifts work left: we run static analysis and dependency checks on each commit, we threat‑model before features are built, and we write tests that prove secure behavior. The payoff is fewer rewrites, fewer emergency fixes, and clearer audit evidence.

**Describe the steps you can take to prevent threats.**

• Plan: Define security user stories and acceptance criteria (validation, authZ, logging). Perform lightweight threat modeling (STRIDE) per feature.  
• Create: Use IDE linters and secure code snippets; default to safe types (`std::size\_t`, `std::string`).  
• Verify (per‑commit): Run SAST (e.g., Coverity/Semgrep), SCA for libraries, and unit tests that include negative cases. Fail the build on High/Critical.  
• Pre‑Production: Run sanitizers (ASan/UBSan), fuzz key parsers, and integration tests with least‑privilege service accounts.  
• Release: Sign artifacts and publish an SBOM; enable infrastructure as code checks (policy‑as‑code) before deploy.  
• Prevent/Detect: Enforce TLS 1.3, rotate keys in KMS, centralize immutable logs, and monitor with alerts tied to controls.  
• Respond/Adapt: Track vulnerabilities, expire waivers, and update standards when incidents or tests reveal gaps.

**Provide one example for your Project Two presentation that ensures security is addressed intrinsically.**

Example: “Security Unit Tests as Gates.” For each API endpoint, I’ll add unit/integration tests that assert:  
1) Input validation rejects out‑of‑range and malformed data (HTTP 400 with safe error body).  
2) Authorization blocks access without the correct role/scope (HTTP 403) and allows only least‑privilege paths.  
3) Logging captures who/what/when (no secrets), using a structured logger.  
These tests run in CI on every pull request. If any fail, the merge is blocked. This makes security intrinsic because the feature isn’t “done” unless its security behaviors are proven by tests. I’ll also include a small Semgrep policy that forbids string‑built SQL, so parameterized queries are enforced automatically.