# Project Two: Security Policy Presentation — Script

Name: Chris Phillips

Date: October 17, 2025

Course: CS 405 Secure Coding

Assignment: Project Two — Security Policy Presentation

YouTube Link: [https://youtu.be/isRODpYvujA]

|  |  |
| --- | --- |
| Slide Number | Narrative |
| 1 | Title — Green Pace Security Policy Presentation. I am Christopher Phillips. This presentation states our security policy, coding standards, testing approach, and recommendations. |
| 2 | Overview. We needed a single policy to align a growing team. This policy reduces the attack surface, sets clear expectations, and supports defense in depth. The outcome is consistent reviews, automated checks, and measurable risk reduction. |
| 3 | Threats Matrix. The matrix lists ten standards with severity, likelihood, and priority. Priority 1 items—SQL injection, buffer overflow, input validation, secrets in code, and missing authorization—block release in Continuous Integration. Priority 2 and 3 items have SLAs for remediation. |
| 4 | Principles. Ten principles keep the policy coherent: minimize attack surface, least privilege, secure defaults, defense in depth, input validation, output encoding, strong crypto, secure secrets, complete mediation, and auditability. Each standard maps to at least one principle. |
| 5 | Coding Standards and Priority. We group the ten standards by business impact, exploitability, and detectability. Priority 1 issues are immediate blockers. This prevents high‑impact vulnerabilities from shipping. |
| 6 | Encryption Policies. TLS 1.3 in flight; encryption at rest with keys in KMS; and minimal plaintext in use. No keys in code. Rotate keys and audit usage. Zeroize sensitive buffers after use. |
| 7 | Triple‑A. Authentication uses strong MFA; service-to-service access uses scoped tokens. Authorization happens in the service layer with deny-by-default policies. Accounting uses structured, tamper‑evident logs. |
| 8 | Unit Test 1: Reject oversized input. I show a test where a 300‑character string is rejected and no memory is overwritten. This prevents buffer issues and abuse of resources. |
| 9 | Unit Test 2: Parameterized SQL. I pass an injection payload. The prepared statement returns zero rows and the log shows bound parameters. This proves string concatenation is not used. |
| 10 | Unit Test 3: Authorization in service layer. A Viewer role tries to delete a resource and receives 403. There are no side effects, and the denial is logged. |
| 11 | Unit Test 4: Output encoding. A script tag in the name field is escaped. The browser renders harmless text and security headers are present. |
|  |  |
| 12  13 | Automation Summary. The Dev Security Ops flow places security tools at pre‑commit, build, test, and deploy. Security gates use policy to block releases with unaddressed risks.  Tools:Pre-commit blocks secrets and basic issues. PR checks and S.A.S.T gate merges. Build runs strict compile plus C.p.p check. We scan dependencies and produce an S.B.O.M; Critical CVEs block. Unit and integration tests enforce validation, parameterized SQL, authz, and encoding. IaC and container scans block weak configs. ZAP and fuzzing probe the running app; sanitizer builds catch memory errors. We sign artifacts, then enforce policy and monitor in runtime. Gates stop risky changes; reports provide evidence for audits. |
| Slides 14-16 | Risks and Benefits. Acting now reduces breach likelihood and rework. Delays raise cost and compliance risk. We phase remediation: Priority 1 first, then Priority 2 and 3. |
| 17 | Recommendations. Gaps include inconsistent encoding, limited service‑layer tests, and manual key rotation. We expand security tests, centralize encoding helpers, and adopt automated key rotation in KMS. |
| 18 | Conclusions. Enforce the ten standards, keep automation as gates, and measure outcomes through pass rates, MTTR, and security test coverage. |
| 20 | References. I will include OWASP, Microsoft documentation for C++ testing, and NIST SP 800‑53. |