Project 2

Kain Mason

CS405

10-19-2025

# Recording & Submission Notes

Record as an MP4 with voice-over narration. Upload as Unlisted to YouTube. Paste the link below before submitting in Brightspace.

YouTube Link: https://youtu.be/mVkiMhku7ts

# Narration Script

## Slide 1 – Title

Hi, I’m Kain Mason. This presentation walks through Green Pace’s security policy—our guiding principles, coding standards, encryption, Triple‑A framework, unit tests, automation, risks, and recommendations.

## Slide 2 – Overview: Why This Policy

We’re standardizing secure development across teams and products. The goal is clear, repeatable practices that align with defense‑in‑depth and are easy to verify in automation.

## Slide 3 – Overview: Threats Matrix & Automation

I’ll use a risk matrix—Likelihood times Impact—to prioritize. Automation catches regressions: SAST, SCA, secret scanning, and DAST. Unit tests keep security behaviors locked in.

## Slide 4 – Threats Matrix (Sample)

Here’s a sample matrix. SQL injection and XSS rank critical. Dependency CVEs and auth issues also score high. Each entry includes the automation signal that would alert us.

## Slide 5 – 10 Guiding Principles

These ten principles drive our standards: validate input, least privilege, fail securely, defense in depth, complete mediation, keep it simple, secure by design, least common mechanism, psychological acceptability, and assume breach.

## Slide 6 – Coding Standards (Priority Order)

Standards map from those principles. We prioritize input validation and encoding, strong auth and authz, modern crypto and secret management, error handling and logging, dependency hygiene, secure configs, data protection, and safe serialization.

## Slide 7 – Encryption Strategy

In flight we enforce TLS 1.3, at rest we use AES‑256‑GCM with managed keys and rotation, and in use we minimize plaintext handling. Passwords use Argon2id or bcrypt with per‑user salts and a pepper stored in KMS.

## Slide 8 – Triple‑A

Authentication is MFA‑ready with lockouts and secure hashing. Authorization is deny‑by‑default with RBAC or ABAC. Accounting means strong audit logs, traceability, and tamper‑evident storage.

## Slide 9 – Unit Test: Input Validation

These tests reject dangerous inputs and enforce size and format constraints. I also seed fuzz cases in CI.

## Slide 10 – Unit Test: Authentication Flow

Tests verify hashing, timing‑safe compares, lockouts, and token/session expiry. This prevents weak auth flows.

## Slide 11 – Unit Test: Authorization Matrix

We verify least privilege and deny-by-default, with negative tests for IDOR and other access vulnerabilities.

## Slide 12 – Unit Test: Crypto & Secrets

Tests ensure only strong ciphers and proper key handling via KMS. Secrets must never come from plain env files.

## Slide 13 – Unit Test: Logging & Errors

We assert no sensitive data in logs, structured formats, and correlation IDs are present. Logs go to an append‑only sink.

## Slide 14 – Automation Summary (DevSecOps)

Here’s where each tool runs: pre‑commit, CI build, security gates, staging DAST, and container/IaC scans, then signed deployments.

## Slide 15 – Risks & Benefits

Quick wins reduce a lot of risk fast. Some items need phased work—like auth migration and logging centralization. Benefits include fewer incidents and stronger audit posture.

## Slide 16 – Recommendations & Roadmap

Adopt the standards, turn on CI gates with documented exceptions, migrate password hashing, enable MFA, formalize RBAC, and centralize logging with alerts.

## Slide 17 – Conclusion

Policy plus automation and testing give us a durable security baseline. We’ll measure progress and iterate.

## Slide 18 – References

References include OWASP, NIST, CISA KEV, Microsoft SDL, and TLS and Argon2 guidance.