CYSE 690 – Cyber Security Engineering Graduate Capstone Project Plan (Spring 2026)

Risk-Aware Compliance-as-Code: An ML-Gated Secure CI/CD Pipeline for Software Supply Chain Integrity

Ayra Islam, Alexander Castro, Derrick Addai Buabeng, Joseph Jnr A Yeboah

---

1. End Product Vision

The goal of this project is to design and implement a risk-aware, compliance-as-code CI/CD pipeline that integrates machine learning to make intelligent deployment decisions. Unlike traditional rule-based pipelines that rely on rigid pass/fail thresholds, the final system will evaluate software builds holistically and classify them as ALLOW, WARN, or BLOCK based on contextual risk.

The final product will include:

- A documented system architecture describing the CI/CD pipeline, security tooling, and ML-based risk gate
- A functional prototype demonstrating SBOM generation, security analysis, and ML-based classification
- A risk and compliance analysis aligned with Executive Order 14028
- A final presentation summarizing the problem, design decisions, system behavior, and lessons learned

Deliverables for this project include:

- Project status updates every three weeks
- Ongoing technical and status reviews
- Final system documentation and architecture
- Final project presentation
- Peer evaluation as required by the course at the end of the semester

---

2. Team Structure and Collaboration Breakdown

The team consists of four members. While responsibilities are assigned to uphold accountability, collaboration and cross-review are expected for all team members.

Proposed Role Structure

- Alex - System Architecture & Integration Lead
- Joseph - Security & Compliance Lead
- Ayra - Machine Learning & Risk Classification Lead
- Derrick - Documentation, Research & Presentation Lead

All members are expected to contribute to design discussions, reviews, and project updates.

---

3. Communication Plan

Communication will be conducted primarily through Microsoft Teams, in alignment with course expectations.

Frequency

- Weekly informal team sync meetings (30–45 minutes)
- Formal project updates every three weeks
- Ongoing and frequent asynchronous communication over group messaging channel (Teams "CYSE 690")

Means of Communication

- Microsoft Teams (primary platform for meetings, chat, and collaboration)
- GMU email for coordination and instructor communication

Document and File Sharing

- Shared Microsoft Teams folder will serve as the single source of truth
- All working documents will be stored and versioned in Microsoft 365 formats

---

4. Rules of Engagement

Professional expectations

- All communication will remain respectful, professional, and transparent

- Team members will communicate blockers early (preferably at least 2-3 days prior to the due date)
- Workloads will be distributed equitably

Decision-making

- Ideas and design alternatives are open for discussion
- Major technical or scope decisions require team consensus
- Unresolved disagreements may be escalated to the instructor if necessary

Meetings

- Meetings will be documented with notes and action items (rotating note-takers amongst team members)
- Meetings will be time-boxed and agenda-driven
- Missed meetings must be communicated in advance and justified

Boundaries

- No major architectural changes will be made close to deadlines without team agreement
- No individual may submit group work without team approval

---

5. Formal Project Updates and Reporting

Formal project updates will be delivered every three weeks, consistent with course expectations. Each update will include:

- Documented summary of the project status and completed tasks
- Address any team identified risks or blockers to ensure proper direction
- Discuss the next steps within the project

In addition to written updates, the team will schedule at least one formal 30–45 minute review meeting with the course instructor during the semester. This meeting will be conducted virtually via Microsoft Teams and will focus on reviewing system architecture decisions, progress against defined KPIs, and any required scope or design adjustments.

If risks or blockers are identified that may impact the project timeline or objectives, the team will take corrective action such as reprioritizing tasks, parallelizing work where feasible, increasing meeting frequency, or adjusting scope to preserve core system functionality. Any significant risks or adjustments will be documented and communicated during formal project updates.

Key Performance Indicators (KPIs)

Detailed discussion on the following key performance indicators (KPIs) to indicate technical progress and design evolution:

- **SBOM Depth and Accuracy:** Ratio of identified components in the SBOM versus the actual dependencies found within the build environment. This aims to measure the "visibility" the pipeline provides

$$SBOM\ Accuracy = \frac{C_{SBOM} \cap C_{Actual}}{C_{Actual}}$$

- **Decision Latency:** The decision latency as represented by the sum of the scan, inference, and enforcement should be kept to fraction of the total build procedure.

$$Decision\ Latency = T_{scan} + T_{inference} + T_{enforcement}$$

- **False Override Rate:** How often does an operator have to override a "BLOCK". This is defined by the sum of manual overrides divided by the number of "BLOCKS" provided by the model

$$False\ Overide\ Rate = \frac{\sum O_{manual}}{\sum D_{block}}$$

- **Decision Congruence:** How often does the model align with manual review. This is defined by the sum of where the ML classification aligns with manual review divided by the number of samples. Note: The use of this KPI is dependent on research for the manual data values.

$$Decison\ Congruence = \frac{\sum(D_{ML} \equiv D_{manual})}{Samples}$$

---

6. Work Breakdown and Expected Effort Flow

Phase 1: Scoping and Requirements

- **Define Assumptions:** Outline any assumptions made by the team in developing the system
- **Define Constraints:** Outline any constraints or limiting factors that define the project's boundaries
- **Stakeholder Alignment:** Map any regulatory requirements (Executive Order 14028) to the system's functional requirements
- **Success Criteria:** Establish quantitative success criteria for all KPIs

Phase 2: Architecture and Design

- **Design CI/CD Pipeline Workflow:** Map a series of 3rd party open-source tools within a GitHub Actions workflow to conduct application build process. This involves finalizing the high-level design and selecting the 3rd party open-source software tools
- **Schema Definition:** Define a unified schema for normalizing deterministic outputs from SAST and SBOM stages for ingestion into ML classification. This bridges the gap between the traditional deterministic CI/CD stages and the ML classification
- **Design ML-Classification Tool:** Per the schema definition, map SAST and SBOM output into a ML inference tool to compute deployment gate decisions

Phase 3: Parallel Execution

- **Develop CI/CD Pipeline Workflow:** Develop and deploy the hardened CI/CD environment to a functional GitHub repository.
- **Develop ML-Classification Tool:** Develop and deploy ML-Classification model against sample outputs per the unified schema definition
- **Incremental Documentation:** Documentation will be updated incrementally as progress is made, including iterative metrics collection for each of the KPIs
- **Incremental Research:** Ongoing research should be conducted to ensure that project developments and direction align with functional requirements

Phase 4: Integration and Validation

- **Integrate Decision Components:** Integrate the ML-Classification tool into the CI/CD pipeline's enforcement gate stage. This should complete the end-to-end pipeline
- **Quantify KPIs:** Conduct iterative analysis of the pipeline with a variety of software inputs to validate the efficacy of the pipeline
- **Qualify And Quantify Risk:** Based upon metrics captured from the pipeline, gauge any qualitative and quantitative risks posed by using the pipeline
- **Access Auditability and Compliance:** Ensure that build process, SAST, SBOM, and ML classification decisions are adequately logged. Ensure logs are immutable and include proper retention for compliance

Phase 5: Finalization

- **Scope Freeze:** Transition from feature development to bug fixing and review 2–4 weeks before final submission
- **Final Integrity Verification:** Conduct and document final end-to-end walkthrough to ensure the entire system satisfies all functional requirements

- **Technical Documentation:** Finalize all end-product deliverables

All tasks within each phase will be assigned to a specific team member, tracked against defined milestones, and completed according to the project timeline aligned with the course syllabus.

---

7. Research Strategy

Research will be conducted continuously throughout the semester in parallel with implementation. Research findings will directly inform:

- Architecture decisions
- Risk modeling and classification logic
- Compliance and auditability justification

Sources will include software supply chain security literature, Executive Order 14028 guidance, and relevant machine learning research. Generative AI tools of various types will be used to guide research and formalize ideas. All sources will be cited where necessary.

Research findings will be periodically reviewed by the team and incorporated into design and implementation decisions during each project phase to ensure alignment with evolving requirements and best practices.

---

8. Timeline and Milestones

All milestones and due dates will align with the official course syllabus.

Key milestones include:

- Project Plan submission (Week 3)
- Phased development boundaries as defined in section 6
- Regular checkpoint reviews throughout the semester
- Final document preparation beginning 2–4 weeks before the final presentation
- Final project presentation at the end of the semester

Milestone progress will be reviewed during weekly team sync meetings and summarized as part of each formal project update to ensure alignment with planned timelines.

---

9. Task Tracking and Progress Monitoring

Project tasks will be tracked using a shared task tracking document created and maintained by Ayra.

The tracker will include:

- Task name
- Linked deliverables
- Task details
- Assigned owner
- Status
- Due date
- Dependencies
- Notes

Each discrete task will be:

- Clearly scoped and independently testable
- Assigned to a single owner
- Mapped to a milestone or deliverable
- Reviewed by at least one other team member

The task tracker will be reviewed and updated during weekly team sync meetings and prior to each formal project update.

---

10. Format and Tooling

Primary tools:

- Microsoft Teams for communication and collaboration
- Microsoft Word, Excel, and PowerPoint for documentation and deliverables

Optional tools:

- Excel for visual task tracking (if needed)
- Diagramming tools such as draw.io for architecture diagrams

A shared GitHub repository will be used to host pipeline code, configuration files, and related artifacts, with version control supporting collaboration, traceability, and review.

Initial strategy for the pipeline will be to utilize the following technologies, although these ideas are subject to change before finalizing the design:

- **GitHub Actions:** Orchestration platform to automate build processing and enable retention of build logs and artifacts
- **Python:** Programming language for application source code. The “venv” package will be used as a dependency vehicle for application dependencies
- **Pytest:** Industry standard unit-testing platform for automated testing. Will be utilized for asserting that application functionality is retained upon iterative development
- **Docker:** Industry standard containerization platform for packaging application source code alongside the environment in which it runs
- **Bandit (PyCQA/bandit):** Specifically designed SAST package for the python ecosystem. Produces a highly structured and standardized output detailing security flaws within source code logic as well as dependencies
- **Trivy (aquasecurity/trivy):** Container image SAST platform that can calculate container image SBOM for OS-level and application packages. Additionally, the tool can utilize the SBOM to find vulnerabilities within the OS and/or application dependencies

AI MODEL CITATION

- Gemini 3 (browser-based SaaS)
- Chat GPT 5.2

Generative AI tools were used to assist with research, idea development, and drafting support, and all content was reviewed, validated, and finalized by the team