IS (2025/2026 - C.Oprișa & A. Coleșa): Project activity & Proposed themes

Project Guide for the Information Security Course

Master in Data Science, Fall 2025

Overview

For this project, students will work in teams of 2 members.

Each team will analyze or prototype a **software application** from an **information security perspective**.

The project develops students' ability to assess risks, design secure architectures, and apply security controls in data-driven environments.

Each team selects **one application theme** (see below) and follows the same **five project phases** across the 14 weeks of the semester and 7 project classes. Each theme must have a different theme. Themes can be selected on a first-come first-serve basis and the chosen theme must be registered in the Project Team Assignments.xlsl file, in the Files of the General channel of the Teams course.

Project Timeline (14 Weeks)

Week	Activity	Deliverable
02	Project kickoff, team formation, topic selection	1-page proposal (given)
04	Phase 1: State of the Art & Security Requirements	5–7 slide presentation & 2–3 page report
06	Phase 2: Data Modeling & Roles (DFD, data classification, access matrix)	5–7 slide presentation & 2–3 page report
08	Phase 3: Security Architecture, Threat Model & Controls	5–7 slide presentation & 2–3 page report
10	Phase 4: Security Implementation (Configuration notes, screenshots, or code)	5–7 slide presentation & 2–3 page report
12	Phase 5: Integration & Security Testing. Final Presentation	5–7 slide presentation & 2–3 page report
14	Recoveries (if the case)	What is to be recovered

Project Phases (for any chosen theme)

1. **State of the Art & Security Requirements**
    - Research existing applications
    - Research applicable security standards
    - Research reported incidents: give statistics and describe the most important ones. Use reported vulnerability databases, like CVE, media coverage, security researchers' technical reports.
    - Define specific confidentiality, integrity, and availability needs for the given type of application.
2. **Data & Roles Modeling**
    - Identify data types (public, confidential, PII).
    - If the case, identify data confidentiality level in the organization.
    - Draw Data Flow Diagram (DFD).
    - Define actors, roles and access rights.
3. **Security Architecture & Threat Modeling**
    - Describe system components and security architecture.
    - Perform threat modeling (STRIDE or PASTA).
    - Conduct qualitative risk assessment
    - Propose security controls.
4. **Security Implementation Details**
    - Implement or configure core security mechanisms: authentication, encryption, access control, logging, etc.
    - If using an open-source app, show secure configuration and testing.
5. **Final Presentation & Conclusion**
    - Key findings, security posture, and lessons learned.
    - Deliver final written report and demo.

Project Themes

1. 🏥 Healthcare Data Dashboard

Analyze or develop a healthcare dashboard handling patient data.

Data: Synthetic (e.g., Synthea)
Focus: Privacy, data encryption, role-based access, data anonymization.
Implementation: Flask dashboard or secure OpenMRS instance
Threats: Unauthorized access, PHI leakage, ransomware
Deliverables: Data flow diagram, threat model, secure setup demo

---

2. 🛒 E-Commerce Web Platform

Simplified online shop with product browsing, orders, and authentication.

Data: Synthetic products & customers
Focus: Web vulnerabilities (XSS, SQLi), authentication, HTTPS
Implementation: Flask/Django mockup or Magento/OpenCart configuration
Threats: Injection, CSRF, weak sessions
Deliverables: Threat model, secure login flow, vulnerability test report

---

3. 🏠 Smart Home IoT Platform

Simulate or configure an IoT platform (sensors → dashboard).

Data: Synthetic sensor readings (temperature, humidity)
Focus: Device authentication, secure communication (TLS/MQTT)
Implementation: Mosquitto + Python clients or Home Assistant
Threats: Replay attacks, device impersonation, insecure firmware
Deliverables: Network diagram, MQTT security configuration

---

4. 🎓 University Student Information System

System for managing student grades, profiles, and courses.

Data: Synthetic student/course data
Focus: Access control, audit logging, confidentiality of grades
Implementation: Flask/Django mockup or Moodle/OpenSIS configuration
Threats: Privilege escalation, data tampering
Deliverables: Role matrix, secure CRUD demo

---

5. 📊 Online Survey or Polling App

Web app collecting anonymous survey responses.

Data: Synthetic surveys/responses
Focus: Privacy, pseudonymization, input sanitization
Implementation: Streamlit or LimeSurvey configuration
Threats: Deanonymization, data injection
Deliverables: DFD, anonymization explanation, secure demo

---

6. 💰 Financial Data Analytics Dashboard

Dashboard for analyzing and visualizing financial datasets.

Data: Stock data (Yahoo Finance API), synthetic transactions
Focus: Integrity, secure APIs, access control
Implementation: Streamlit/Dash mockup or ERPNext setup
Threats: Insider abuse, data tampering, insecure APIs
Deliverables: Secure dashboard + audit logging

---

7. 🏭 Industrial IoT Monitoring System

System collecting data from industrial devices or sensors.

Data: Synthetic telemetry data
Focus: Network segmentation, secure protocols (Modbus, MQTT)
Implementation: Node-RED or Python MQTT simulation
Threats: MITM, ransomware, insecure endpoints

Deliverables: Architecture diagram, risk assessment

---

8. 🤖 Machine Learning-as-a-Service (MLaaS)

Web service where users upload data and train ML models.

Data: CSV datasets (iris, MNIST subset)
Focus: Data validation, sandboxing, poisoning protection, secret leakage protection, i.e. protect the MLaaS against being trick to leak a secret it knows (e.g. restrict answers to a predefined theme / domain). **Implementation:** Flask + scikit-learn
Threats: Model theft, data poisoning, prompt-based exfiltration
Deliverables: Secure pipeline diagram, testing report

---

9. ☁️ Cloud-Based File Sharing / Backup Service

Analyze or deploy a private file sharing solution.

Data: Synthetic office files
Focus: Encryption, access policies, secure sharing
Implementation: Nextcloud or Flask mockup
Threats: Metadata leakage, ransomware, privilege escalation
Deliverables: Secure configuration report, sharing demo

---

10. 🗂️ Public Sector Open Data Portal

Evaluate the security of an open data publishing system.

Data: Public datasets from data.gov or data.europa.eu
Focus: Anonymization, data integrity, controlled release
Implementation: CKAN or Streamlit portal
Threats: Re-identification, data poisoning
Deliverables: Security policy, anonymization flow

Small social platform with user posts, comments, and profiles.

Data: Synthetic user data and posts
Focus: Privacy, input validation, content moderation, RBAC
Implementation: Flask/Django app or Mastodon/HumHub configuration
Threats: XSS, CSRF, data scraping, impersonation
Deliverables: Threat model, secure authentication demo

---

12. 🧱 Security Solution Application (End-user, Enterprise, Cloud)

Analyze a security product (antivirus, EDR, SIEM, and cloud security dashboard).

Data: Synthetic logs, telemetry, alerts
Focus: Trust chain, integrity, secure communication
Implementation: Wazuh, Security Onion, or Python telemetry demo
Threats: Insider misuse, agent tampering, data exfiltration
Deliverables: Architecture diagram, log security workflow

---

13. 🧾 Accounting and HR Application (Retail Company)

System managing payroll, HR data, and accounting records.

Data: Synthetic employee and financial data
Focus: Confidentiality, segregation of duties, audit logging
Implementation: ERPNext or Odoo
Threats: Insider fraud, privilege escalation, data leaks
Deliverables: Risk matrix, secure workflow demo

---

14. 💻 Software Development and Management Tools

DevSecOps ecosystem: Git, CI/CD, and issue tracking systems.

Data: Synthetic code repositories, issue logs
Focus: Secret management, pipeline hardening, access control
Implementation: GitLab, Jenkins, or GitHub Actions
Threats: Credential theft, malicious commits, supply chain attacks

Deliverables: Secure CI/CD documentation, threat model

---

15. 🧠 LLM-as-a-Service Application (Protected Against Prompt Injection)

Expose an LLM API while protecting it from malicious inputs and data leaks.

Data: Synthetic prompts/responses
Focus: Prompt injection mitigation, sandboxing, input/output filtering
Implementation: Flask + OpenAI/Hugging Face model
Threats: Prompt injection, data exfiltration, abuse of APIs
Deliverables: Threat model, secure prompt handling pipeline, injection test report

---

Deliverables Summary

Phase	Deliverable	Format
1	State-of-the-art and requirements	5–7 slide presentation & 2–3 page report
2	Data flow, classification, roles	5–7 slide presentation & 2–3 page report (Diagrams, tables)
3	Threat model and security design	5–7 slide presentation & 2–3 page report
4	Implementation/configuration	5–7 slide presentation & 2–3 page report (Screenshots/code + notes)
5	Final report & presentation	5–7 slide presentation & 2–3 page report

---

Grading Rubric

Criterion	Weight
Understanding of security requirements & research depth	15%
Data & roles modeling clarity	15%
Threat modeling & architecture quality	25%
Technical implementation or configuration	25%
Presentation quality & report structure	20%

Optional Extensions

- **Cross-Team Review:** Evaluate another team's project from a security testing perspective.
- **Compliance Mapping:** Compare your system's controls to ISO 27001, GDPR, or SOC 2.
- **Policy Writing:** Draft a 2-page "Security Policy" for your application.
- **Ethics Reflection:** Discuss ethical and privacy implications of your app.

Submission Guidelines

- Teams of **2 students**.
- Workload: **~2–3 hours / week / student**.
- All code/configs and deliverables should be uploaded to a private **GitHub repository**, shared between the team members. The repository should be created by accepting the GitHub Classroom assignment.
- Submit all deliverables in **PPTX, PDF or Markdown format**.
- All presentations: **8 minutes + 2 minutes Q&A**.