Team-Based Final Projects in Cybersecurity

Project Overview

This final project is designed to provide hands-on experience in cybersecurity through collaborative, team-based work. Each project emphasizes practical implementation, integration of multiple skills, and real-world application of cybersecurity concepts. Students will work in teams of 3–5 members to design, implement, and demonstrate functional cybersecurity solutions.

All teams must deliver a **complete, functional project** by **December 12, 2025** — including **source code, a working demo, and comprehensive documentation**. No separate presentation session is required; the demo video and documentation will serve as your official submission for evaluation.

Available Team-Based Projects

1. Brute Force Attack Simulator & Defense Toolkit

Objective: Simulate brute force attacks to understand vulnerabilities and develop basic countermeasures.
Team Roles & Tasks:

- *Attacker Module Developer:* Implements password-guessing logic using common password lists.
- *Defender Module Developer:* Builds login system with rate-limiting, lockout, or CAPTCHA.
- *Tester & Analyst:* Evaluates effectiveness, logs attempts, and documents findings.

Deliverable: A working demo showing both attack simulation and mitigation strategies, plus a short report on lessons learned.

2. Secure File Vault: Encryption & Decryption Tool

Objective: Build a user-friendly tool to securely encrypt and decrypt files using modern cryptographic standards.
Team Roles & Tasks:

- *Encryption Engineer:* Implements file encryption/decryption using Python's `cryptography` library.

- *UI/UX Designer:* Creates a simple command-line or GUI interface for file selection and key management.
- *Security Auditor:* Validates key handling, ensures no plaintext leakage, and writes usage guidelines.

Deliverable: Functional tool + security documentation explaining algorithm choices and safe usage.

3. SSL/TLS Certificate Health Monitor

Objective: Develop a utility that inspects and reports on the validity and security posture of website SSL/TLS certificates.
Team Roles & Tasks:

- *Core Developer:* Uses Python's `ssl` and `socket` modules to fetch and validate certificates.
- *Reporting Specialist:* Formats clear, actionable alerts for expired, self-signed, or weak-cipher certificates.
- *Integration Lead:* Adds batch-check capability for multiple URLs and logs results.

Deliverable: Certificate checker script with sample report output and recommendations for sysadmins.

4. Lightweight Intrusion Detection System (IDS)

Objective: Build a simplified network-based IDS that flags suspicious traffic patterns.
Team Roles & Tasks:

- *Traffic Monitor:* Configures packet capture (e.g., using Scapy or integrates with Snort/Suricata rules).
- *Rule Developer:* Defines signatures for common threats (e.g., port scans, known malware patterns).
- *Alerting & Logging:* Designs real-time alerts and secure log storage.

Deliverable: Functional IDS prototype with rule set, detection demo, and explanation of detection logic.

5. Password Manager with Multi-Factor Authentication (MFA)

Objective: Create a secure local password manager that enforces MFA for access.
Team Roles & Tasks:

- *Core Storage Developer:* Implements encrypted password storage (e.g., using AES + master password).
- *MFA Integrator:* Adds TOTP-based authentication using `pyotp` and QR code setup.

- *Security & Usability Tester:* Evaluates recovery options, brute-force resistance, and user workflow.

Deliverable: Working password manager + demo of login with MFA + threat model analysis.

6. End-to-End Secure File Sharing Platform

Objective: Design a system for sharing files securely—encrypted in transit and at rest—with access control.
Team Roles & Tasks:

- *Encryption Layer:* Handles file encryption before upload and decryption after download.
- *Network & Auth Layer:* Implements secure transfer (e.g., HTTPS/FTPS) and user authentication.
- *Access Control & Audit Lead:* Manages permissions and logs file access events.

Deliverable: Minimal secure file-sharing system (local or localhost) with encryption, auth, and audit trail.

7. Ransomware Behavior Detector (ML-Based)

Objective: Use machine learning to detect early signs of ransomware activity on a host system.
Team Roles & Tasks:

- *Data Engineer:* Collects/generates simulated file system behavior data (normal vs. ransomware-like).
- *ML Model Developer:* Trains and validates a classifier (e.g., using Scikit-learn).
- *Response System Designer:* Implements alerts or protective actions (e.g., halt process, backup trigger).

Deliverable: Trained model + detection script + evaluation metrics (precision, recall) on test data.

8. Smart Secure Room: IoT Access & Logging System

Objective: Build a physical security prototype that grants access only when environmental and user conditions are safe, with encrypted audit logs.
Team Roles & Tasks:

- *Hardware Integrator:* Connects and reads data from PIR (motion), ultrasonic (distance), DHT11 (temp/humidity), and WH-011 (door/window).
- *Policy Enforcer:* Programs access logic (e.g., "allow entry only if room is unoccupied and temp < 35°C").
- *Security & Logging Lead:* Implements encrypted, timestamped log chaining (e.g., using hash chains) and LED status indicators.

Deliverable: Working prototype with sensor integration, access control logic, secure logs, and clear visual feedback.

9. AI-Powered Phishing URL Detector

Objective: Build a machine learning model that classifies URLs as legitimate or phishing based on lexical and structural features.
Open Dataset Suggestions:

- PhishStorm
- Malicious URLs Dataset (Kaggle)

Team Roles & Tasks:

- *Data Preprocessor:* Extracts features from URLs (e.g., length, number of dots, presence of IP address, suspicious keywords).
- *ML Model Developer:* Trains a classifier (e.g., Random Forest, Logistic Regression, or lightweight neural net via Scikit-learn/TensorFlow).
- *Evaluator & Deployer:* Tests model accuracy, creates a simple web or CLI interface for real-time URL checking.

Deliverable:
A trained phishing detection model + a demo tool where users input a URL and receive a risk prediction, along with a brief report on feature importance and model performance.

10. Network Anomaly Detection Using AI

Objective: Use machine learning to identify anomalous network traffic that may indicate cyberattacks (e.g., DDoS, port scans).
Open Dataset Suggestions:

- CICIDS2017 (Comprehensive, labeled traffic including normal and attack types)
- NSL-KDD (Classic intrusion detection dataset)

Team Roles & Tasks:

- *Data Analyst:* Cleans dataset, performs exploratory analysis, and selects key features (e.g., packet rate, protocol type, duration).
- *AI Developer:* Implements an unsupervised (e.g., Isolation Forest) or supervised (e.g., XGBoost) anomaly detector.
- *Visualization & Alerting Lead:* Builds dashboards (e.g., using Matplotlib or Streamlit) to show normal vs. anomalous traffic and generates alerts.

Deliverable:
A working anomaly detection system that processes sample network logs, flags suspicious

activity, and visualizes results—accompanied by a performance summary (e.g., confusion matrix, F1-score).

Final Submission Guidelines

Deadline:

All submissions due by: December 12, 2025, 11:59 PM

`No extensions will be granted. Late submissions will not be accepted.`

What to Submit (Single ZIP File):

Name your ZIP file:
`TeamName_ProjectNumber.zip`
(e.g., "TeamAlpha_Project9.zip")

Inside the ZIP, include **three clearly labeled folders**:

1. `code/`

- All source code files (.py, .ipynb, etc.)
- `README.md` with:
  - Project title and team members
  - Step-by-step setup instructions (e.g., install dependencies, run script)
  - How to test the system
- `requirements.txt` listing all Python packages
- Folder structure must be clean and organized

2. `demo/`

- One **MP4 video file** (max 10 minutes)
- Content must include:
  - Live demonstration of the full working system
  - Explanation of core features and security mechanisms
  - Clear demonstration of attack detection, encryption, access control, or AI classification
  - Brief reflection on challenges and how they were solved
- Video must be clearly labeled: `demo.mp4`

3. `documentation/`

- One **PDF document** (max 10 pages) titled: `Report.pdf`
- Must include:

    - Executive summary
    - Technical design and architecture
    - Security analysis (threat model, assumptions, limitations)
    - Testing methodology and results (e.g., accuracy, false positives)
    - Team member roles and contributions
    - References and datasets used

Submission Method

- Upload your ZIP file via the course's designated learning management system (LMS).
- Confirm your submission by checking the system receipt.
- **No email submissions** — only LMS uploads will be graded.

Grading Criteria

Criteria	Weight
Functionality & Completeness (Does it work as intended?)	35%
Code Quality & Documentation (Clean, readable, well-documented?)	25%
Security Design & Implementation (Are security principles applied correctly?)	25%
Demo Quality & Clarity (Is the video clear, concise, and convincing?)	10%
Report Depth & Team Contribution (Is analysis thoughtful? Are roles clear?)	5%