

ECE 49595 – Software Development Plan

AI RedTeam — AI-Driven Red Team Simulation Platform

ECE 49595SD – Software Track – Team S 06, Fall 2025

Team Members:

Si Ci Chou	(Simon)	chou170@purdue.edu
Yu-Kuang Chen	(Eric)	chen5292@purdue.edu
Apostolos Cavounis	(Paul)	acavouni@purdue.edu

I. Project Overview

AI RedTeam is an open-source, AI-driven red-team simulation platform that enables users to conduct realistic system hardening without the cost of traditional penetration testing. The system leverages a locally hosted Large Language Model (LLM) through the Ollama stack, enhanced with a Retrieval-Augmented Generation (RAG) pipeline, to simulate intelligent attack behaviors and generate context-aware vulnerability reports. By combining automation with human oversight, AI RedTeam allows security analysts and developers to safely perform reconnaissance, identify common web vulnerabilities, and visualize findings through a real-time dashboard. The goal is to make advanced cybersecurity testing affordable, repeatable, and accessible to small businesses, educational institutions, and security enthusiasts.

II. Scope

1. In Scope

- **LLM-Driven Attack Simulation:** Integrate a locally hosted LLM to generate realistic, text-based red-team scenarios and attack plans.
- **RAG Integration:** Implement a retrieval pipeline that supplies the AI with relevant security documentation, playbooks, and previous scan outputs for contextually accurate responses.
- **Reconnaissance and vulnerability detection:** Perform authorized reconnaissance using open-source tools such as Nmap, Nikto
- **Dashboard Visualization:** Create an interactive dashboard displaying engagement logs, vulnerability reports, and severity analytics.
- **Human in the Loop Safeguard:** Require user confirmation before any high-impact or potentially harmful command is executed
- **Authorization & Ownership Verification:** A mandatory workflow that verifies the user has explicit permission (via token or file) to test a target before any active scans are enabled.
- **Sandboxed Execution Environment:** All active scanning tools (Nmap, SQLMap, etc.) will run within isolated Docker containers to prevent unintended network interaction with the host system.
- **RAG Source Vetting:** The RAG pipeline will include metadata tracking (timestamps, source origin) and a whitelist to ensure only trusted security documentation is used for generation.

- **Deployment & Onboarding:** Development of a Docker Compose configuration and quick-start script to allow non-technical users to deploy the local stack easily.
- **Rate Limiting & Safe Defaults:** The system will default to "Passive Mode" (non-intrusive) and enforce rate limits on active scans to prevent accidental Denial of Service (DoS).

2. Under Evaluation

- **Adaptive RAG Updates:** Automatically scrape trusted sources (OWASP, etc) to update the document store periodically.
- **Advanced Exploit Chaining:** The LLM can reason across multiple discovered vulnerabilities at once to generate a combined attack path.
- **Attack Replay Mode:** Save and replay simulated attack sequences for retraining analysts
- **GitHub Integrations:** Integration with GitHub Actions to scan code automatically during pull requests.

3. Out of Scope

- **Full model training from scratch:** Developing or fine-tuning LLMs requires advanced machine learning expertise, large datasets, long training cycles, and GPU infrastructure that exceed the time and cost constraints of a senior design project.
- **Running large-scale or parallel scans:** Building a system that can automatically run many scans at once across different servers or targets would require complex coordination, a lot of computing power, and extensive testing. For this project, we will focus on running one scan at a time in a controlled environment, to start.
- **Building a mobile/desktop application for the dashboard:** Creating platform-specific applications for IOS, Android, or Windows would add a lot of development time and design work. The project will instead focus on a single, web-based dashboard that works on most devices.

4. Assumptions & Constraints

- **Assumptions:**
 - **Authorized testing only:** All scans and attack simulations will be performed only on the systems that the user has explicit permission to test.
 - **Local Environment:** Users will have access to a computer capable of running a small, locally hosted LLM through the Ollama framework without requiring cloud infrastructure.
 - **Technical Familiarity:** The primary users will have a basic understanding of cybersecurity concepts and responsible testing practices.
- **Constraints:**
 - **Limited Compute:** The team will be restricted to standard desktop hardware.
 - **Time Boundaries:** The project must be completed within two semesters, limiting experimentation with advanced features
 - **Data Privacy:** No User or target data will leave the local environment; all model queries and results must stay self-contained.
 - **Model Performance Variability:** Because we rely on locally hosted LLMs, output quality and response times may vary based on hardware capability and model

size.

III. Requirements

- **CAP-1 — LLM-driven scenario generation & execution.**
The platform's ability to generate attack plans with the self-hosted LLM, run authorized reconnaissance/exploitation workflows, and produce execution logs.
- **CAP-2 — RAG ingestion & contextual retrieval.**
The pipeline that ingests docs (playbooks, READMEs, prior scans) into a vector index and returns context for the LLM during planning.
- **CAP-3 — Dashboard reporting**
The UI and export functions that present findings, remediation guidance, and downloadable reports.
- **CAP-4 — Human approval & safety controls.**
The human-in-the-loop gating system blocks high-impact actions until explicit user confirmation and records approvals in the audit log.

1. Minimum Functional Requirements (suggested placeholders)

- **ID: FR-1 Authorized Red-Team Scenario Execution**
 - **Statement:** The system shall support creating and executing an authorized red-team scenario against a user-supplied target, **defaulting to "Passive Mode" until the user explicitly enables active testing.**
 - **Rationale:** Reliable, affordable, and safe scenario execution is the core mission. Defaulting to passive mode prevents accidental damage during initial setup.
 - **Test method:** Run 10 scenarios. Verify that the system denies active scanning commands until the "Active Mode" toggle is enabled. For medium scenarios 95% of runs must finish ≤ 30 minutes.
 - **Supporting context:** Medium-complexity target = a small web app with DB backend and several endpoints; high-impact command = any action modifying remote state or affecting availability.
 - **Trace:** Traces to capability CAP-1: LLM-driven scenario generation & execution and CAP-3: Dashboard reporting
 - **Priority:** Must Have.
- **ID: FR-2 LLM-Powered Attack Simulation Against User-Supplied Targets**
 - **Statement:** The system shall support generating red-team attack plans using a locally hosted LLM based on a user-supplied target and context.
 - **Rationale:** Simulated attack planning is a core feature that helps users understand potential attack vectors and strengthen system defenses.
 - **Test method:** Run 10 simulations. Verify that the false positive rate for suggested vulnerabilities is below 20% by cross-referencing suggestions against a known vulnerable testbed (e.g., DWVA).
 - **Supporting context:** The LLM is hosted locally via the Ollama framework. Inputs include target URL, known vulnerabilities, and environment configuration.
 - **Trace:** Traces to capability CAP-1: LLM-driven scenario generation & execution
 - **Priority:** Must Have

- **ID:** FR-3 Contextual Document Retrieval via RAG Pipeline
 - **Statement:** The system shall ingest user-supplied documents into a vector index and capture metadata (source, timestamp, trust level) to ensure provenance of retrieval.
 - **Rationale:** Retrieval-Augmented Generation (RAG) enhances accuracy. Tracking provenance ensures the LLM does not hallucinate based on outdated or untrusted data.
 - **Test method:** Upload 20 documents. Query the system and verify that citations provided in the response match the correct metadata tags (source/date) stored in the vector database.
 - **Supporting context:** Uses FAISS or similar vector database. Supported formats include PDF, TXT, and Markdown.
 - **Trace:** Traces to capability CAP-2: RAG ingestion & contextual retrieval
 - **Priority:** Should Have

- **ID:** FR-4 Interactive Dashboard with Vulnerability Reports and Reproducible PoC Steps
 - **Statement:** The system shall present scan results in a web-based dashboard, grouping findings by severity (Critical, High, Medium, Low) and allowing users to filter "noisy" low-impact logs.
 - **Rationale:** A visual dashboard helps users understand weaknesses. Filtering is necessary to prevent information overload for small-team users.
 - **Test method:** Run a scan generating 50+ logs. Verify that the user can filter to show only "High" and "Critical" alerts and that the default view prioritizes severity.
 - **Supporting context:** The dashboard is built using web technologies and supports interactive filtering and export.
 - **Trace:** Traces to capability CAP-3: Dashboard reporting
 - **Priority:** Must Have

- **ID:** FR-5 Human-in-the-Loop Approval Workflow for High-Impact Commands
 - **Statement:** The system shall require explicit user approval before executing any high-impact command that could modify remote state or affect system availability.
 - **Rationale:** This safeguard ensures ethical and legal compliance by preventing unsafe or destructive actions during automated testing.
 - **Test method:** Run 10 simulations that include high-risk commands (e.g., SQL modification, DoS tests). Verify that the system blocks execution and prompts user approval in 100% cases.
 - **Supporting context:** High-impact commands are defined as any action that modifies data, disrupts services, or triggers security alerts.
 - **Trace:** Traces to capability CAP-4: Human approval & safety controls
 - **Priority:** Must Have

- **ID:** FR-6 Mandatory Authorization Verification
 - **Statement:** The system shall require verifiable proof of authorization (e.g., a signed token or ownership file) from the user before enabling any active or high-

- impact scans.
- **Rationale:** To ensure legal and ethical operation, the platform must confirm the operator has permission to test the specified target.
- **Test method:** Attempt to initiate an active scan (e.g., Nmap) without uploading authorization proof; the system shall block the scan. Upload a valid token and verify the system accepts it.
- **Supporting context:** Authorization may be demonstrated by a target-IP whitelist or owner verification token.
- **Trace:** Traces to CAP-4 (Human approval & safety controls).
- **Priority:** Must Have.
- **ID:** FR-7 Sandboxed Execution for Active Scans
 - **Statement:** The system shall execute all active scanning tools (Nmap, SQLMap) within an isolated, containerized sandbox environment.
 - **Rationale:** Sandboxing ensures that active probing does not accidentally affect the host machine or bleed into unauthorized networks.
 - **Test method:** Initiate a scan and inspect the host process list to confirm the scanner is running inside a Docker container. Verify network traffic is constrained to the sandbox interface.
 - **Supporting context:** Implemented via Docker with ephemeral networks.
 - **Trace:** Traces to CAP-1 and CAP-4.
 - **Priority:** Must Have.

2. Minimum Non-Functional Requirements

- **ID:** NFR-1 Dashboard Performance
 - **Statement:** The system shall return dashboard query results in under 2 seconds under normal load conditions.
 - **Rationale:** Fast response times improve user experience and ensure the dashboard remains usable during active analysis sessions.
 - **Test method:** Simulate 50 dashboard queries under normal load (defined as 10 concurrent users). Measure response time for each query; at least 95% must complete within 2 seconds.
 - **Supporting context:** Normal load assumes a single-machine deployment with standard desktop hardware.
 - **Trace:** Traces to capability CAP-3: Dashboard reporting
 - **Priority:** Should Have
- **ID:** NFR-2 Audit Log Security
 - **Statement:** The system shall retain audit logs for at least 90 days and prevent unauthorized tampering or deletion.
 - **Rationale:** Secure audit logs are essential for compliance, accountability, and forensic analysis in case of misuse or unexpected behavior.
 - **Test method:** Perform 20 simulated user actions and verify that each is recorded in the audit log. Attempt unauthorized modification and confirm that the system blocks changes.
 - **Supporting context:** Logs are stored in an append-only format with cryptographic integrity checks.

- **Trace:** Traces to capability CAP-4: Human approval & safety controls
- **Priority:** Must Have
- **ID:** NFR-3 Code Maintainability
 - **Statement:** The system shall follow a documented code style guide and run automated unit tests on every pull request via CI.
 - **Rationale:** Consistent code style and automated testing improve maintainability, reduce bugs, and support long-term scalability.
 - **Test method:** Review codebase for style guide adherence and verify that CI pipeline runs unit tests on all PRs. At least 90% of PRs must pass tests before merging.
 - **Supporting context:** CI pipeline uses GitHub Actions; style guide based on PEP8 (for Python) and ESLint (for JavaScript).
 - **Trace:** Traces to capability CAP-1: LLM-driven scenario generation & execution
 - **Priority:** Could Have
- **ID:** NFR-4 Ease of Deployment (Onboarding)
 - **Statement:** The system shall provide a unified Docker Compose configuration and startup script that deploys the full stack (LLM, Dashboard, Database) in under 15 minutes on standard hardware.
 - **Rationale:** Small business users lack expertise to set up complex stacks manually; accessibility is a core project goal.
 - **Test method:** execute the "quickstart" script on a fresh machine (meeting minimum specs) and verify all services are "healthy" within 15 minutes.
 - **Priority:** Must Have.
- **ID:** NFR-5 Data Privacy & Local Processing
 - **Statement:** The system shall ensure 100% of LLM inference and scan data processing occurs locally, with no data transmission to external cloud services.
 - **Rationale:** To protect sensitive vulnerability data of the user's organization from exfiltration or third-party exposure.
 - **Test method:** Monitor network traffic during a full scan simulation using Wireshark; verify zero outbound packets to external API endpoints (excluding local network targets).
 - **Priority:** Must Have.

IV. Deliverables

1. **Deliverable Description:**
 A working web-based dashboard that displays vulnerability reports and severity ratings. Includes export functionality for PDF and JSON formats.
Relevant Requirements: FR-4, NFR-1
2. **Deliverable Description:**
 A functional RAG ingestion pipeline that indexes user-supplied documents and enables contextual retrieval for LLM queries. Includes a sample corpus of security documents.
Relevant Requirements: FR-3

3. **Deliverable Description:**
A locally hosted LLM integration using the Ollama framework, capable of generating multi-step attack plans based on user-supplied targets and context.
Relevant Requirements: FR-2, FR-1
4. **Deliverable Description:**
A human-in-the-loop approval system that blocks high-impact commands and logs all approvals in a secure audit trail.
Relevant Requirements: FR-5, NFR-2
5. **Deliverable Description:**
A documentation package including a user guide, API specification, and quick start instructions. Developer documentation includes code style guide, CI setup, and test coverage summary.
Relevant Requirements: NFR-3
6. **Deliverable Description:**
A final test report summarizing scenario execution results, dashboard performance benchmarks, and audit log integrity tests.
Relevant Requirements: FR-1, NFR-1, NFR-2
7. **Deliverable Description:**
Final project report and presentation for course submission, summarizing system architecture, capabilities, and evaluation results.
Relevant Requirements: All requirements

V. Development Methodology

1. Selected Methodology: Agile Scrum

For the development of AI RedTeam, the team has selected the **Agile Scrum** methodology, structured around two-week sprints.

Rationale:

- **Adaptability to LLM Uncertainty:** Working with locally hosted LLMs (Ollama) introduces variability in model output and performance. Scrum allows us to inspect and adapt our prompt engineering and RAG pipeline strategies at the end of every sprint rather than waiting for a final release.
- **Iterative Testing:** The project requires distinct phases of testing, such as verifying sandboxed execution and human-in-the-loop safeguards. Scrum facilitates continuous integration, ensuring that safety controls are tested iteratively as new features are added.
- **Team Size & Roles:** With a team of four members, a lightweight Scrum framework allows for clear role distribution (e.g., separating Dashboard frontend work from RAG pipeline backend work) while maintaining frequent communication through stand-ups to resolve blockers quickly.

VI. Verification and Validation Plan

PDF in GitHub Link: <https://github.com/FalsedeerX/AI-RedTeam/blob/672be399bc0a10f3e6fff5751c163067738168b9/Verification%20and%20Validation%20-%20Team%206.pdf>

VII. Gantt Chart

Google Sheet Gantt Chart Link:

https://docs.google.com/spreadsheets/d/1cqIy9I1Q1dPheey5YL1nW1TAs_tfTveG2ZRlWCyNg6k/edit?usp=sharing