Faculty Applied Computer Sciences

Bachelor Künstliche Intelligenz / Artificial Intelligence

SECTRAIN

AI Project (SECTRAIN) - Summer Semester 2025

Name:Krunal Koladiya MatrikelNr.:22306168

Name:Kevin Sangani MatrikelNr.:22208679

Name:Parth Navadiya MatrikelNr.:*22202538*

Lecturer:

Robert Aufschläger

Deggendorf, 01. July 2025

Acronyms

- **RAG** – Retrieval-Augmented Generation, a technique that combines information retrieval with text generation.
- **LLM** – Large Language Model, a deep learning model (often transformer-based) trained on vast text data for natural language tasks.
- **CTF** – Capture The Flag, a style of cybersecurity competition or exercise with challenge-solving tasks.
- **OWASP** – Open Web Application Security Project, a community providing security guidelines and resources (e.g., OWASP Top 10).
- **CVE** – Common Vulnerabilities and Exposures, a publicly disclosed cybersecurity vulnerability identifier (often used with the NVD database).
- **NVD** – National Vulnerability Database, a repository of CVE records and related security data.
- **API** – Application Programming Interface, a set of protocols for interacting with software components.
- **F1-score** – The harmonic mean of precision and recall, used as an accuracy metric for answers.
- **BERTScore** – A similarity metric using BERT embeddings to compare text similarity (often used to evaluate answer quality).

Table of Contents

Introduction

Secure programming is a critical skill in cybersecurity education, yet students often struggle to find reliable, immediate guidance when practicing coding and hacking exercises. Traditional resources (textbooks, online forums) may be outdated or contain insecure examples.

In the context of a Capture-The-Flag (CTF) based self-learning platform for a university secure programming lecture, there is a strong motivation to provide an interactive tutor that can answer questions, explain concepts, and demonstrate secure coding practices on demand. Large Language Model (LLM) chatbots such as ChatGPT have shown the potential of AI tutoring, but using cloud-based solutions raises privacy, cost, and integration concerns. Our project **SECTRAIN** (Security Trainer Chatbot) aims to fill this gap by developing a locally running chatbot specialized in secure programming assistance.

SECTRAIN's objectives are to enhance student learning in several areas: secure coding best practices (e.g., input validation, cryptography APIs), practical Linux/Kali command usage, and preparation for offensive security certifications. The chatbot is designed to integrate with the existing CTF platform—for example, students can ask for hints or explanations during challenges—without relying on external APIs. This ensures student queries and code remain local, preserving confidentiality (a key reason for choosing an open-source LLM deployed on-premise).

To achieve these goals under resource constraints, we combine fine-tuning (to impart domain-specific knowledge to the model) with *Retrieval-Augmented Generation (RAG)* (to provide up-to-date factual information). Fine-tuning a smaller open-source LLM on cybersecurity content can yield a model that "punches above its weight"—a smaller spe-

cialized model can outperform a larger general model on in-domain queries. At the same time, RAG is employed to mitigate the well-known issues of LLMs: the difficulty of updating their static knowledge and their tendency to hallucinate information. By retrieving relevant documents (e.g., OWASP guidelines or recent CVE descriptions) and feeding them into the prompt, the chatbot can generate answers that are both accurate and current, addressing the evolving nature of security threats.

In summary, the introduction of SECTRAIN is motivated by the need for a domain-aware, reliable, and offline-capable AI tutor that enhances secure programming education by combining the strengths of fine-tuned LLMs and RAG.

Methodology

The development of **SECTRAIN** followed a methodology grounded in research insights and iterative experimentation. The process comprised two main phases: (1) *Literature Review & Design* and (2) *Prototyping & Evaluation*.

1. Comprehensive Literature Review

We began by surveying recent advances in large language models (LLMs) and their applications in education and security. Key findings shaped our design decisions:

- **Domain-specific LLMs**: Studies show that these outperform general-purpose models in specialized tasks. This supported our approach to fine-tune a model on cybersecurity content, expecting improved accuracy for secure coding queries.
- **Retrieval-Augmented Generation (RAG)**: RAG improves factual accuracy and reduces hallucinations in chatbot responses. Swacha and Gracel (2025) highlight RAG's effectiveness in handling the knowledge update problem and enhancing chatbot reliability in educational settings.
- **Hybrid Approach**: We adopted a strategy combining a fine-tuned LLM for fluency and RAG to ground responses in accurate reference material.
- **Tool Selection**: LangChain was selected for its modular support for RAG pipelines, and FAISS (Facebook AI Similarity Search) was chosen as the vector store due to its efficiency in local CPU environments.

2. Prototyping and System Design

Based on the literature insights, we designed and iteratively refined SECTRAIN's architecture:

- **Model Selection**: We compared several models — TinyLlama-1.1B, Mistral-7B, LLaMA-3.1-8B, WhiteRabbitNeo-V3-7B, and Phi-4 — based on size, accuracy, and licensing. Models like WhiteRabbitNeo, trained on DevSecOps data, demonstrated strong domain expertise.
- **CPU Optimization**: Despite better performance from larger models, we prioritized a balance between capability and inference latency. A 7–8B parameter model was chosen as a compromise for CPU-only environments.
- **Fine-Tuning**: We created a small domain-specific dataset from OWASP guides, vulnerability examples, and secure coding solutions. Supervised instruction-tuning was performed to align the model with the tone and detail required of a helpful security tutor.

3. Implementation and Testing

We implemented the RAG pipeline using the selected tools and resources:

- **Document Ingestion**: Reference documents (OWASP Top 10, Secure Code Review Guide, CVE/NVD entries, and lecture materials) were split into 512-token chunks, embedded using SentenceTransformer, and indexed in FAISS.
- **Query Handling**: Using LangChain, retrieval chains were built to fetch the top-k relevant chunks, which were appended to the prompt. A customized template encouraged citing facts and discouraged hallucinations.
- **Example**: For the query "How do I prevent SQL injection in PHP?", the system retrieved OWASP's explanation on prepared statements,

which the model then incorporated into its answer.

- **Optimization**: Embedding models, top-k values, and prompts were iteratively adjusted. LangChain's buffer memory was used to preserve conversation context.

Evaluation Setup:

- A small test set of representative queries was prepared (e.g., "Explain XSS", "Secure password storage?").
- Each generated answer was compared with instructor-provided ground truths using **F1-score** and **BERTScore**.
- A subset of answers was also reviewed by instructors for correctness and clarity, ensuring both objective and subjective assessment of the chatbot's educational value.

Implementation Details

System Architecture

The SECTRAIN chatbot architecture is a hybrid pipeline combining local LLM inference with retrieval capabilities.

- **LLM Backend:** A fine-tuned 7B/8B parameter model running on a local CPU server.
- **Vector Store (FAISS):** Contains document embeddings from reference texts.
- **Integration Interface:** Embeds the chatbot into the learning platform via an iframe or API.

Data flow pipeline:

1. **User Query:** Students interact via a web interface embedded in the CTF platform. The query is sent to SECTRAIN's backend.
2. **Retrieval:** The backend encodes the query into an embedding and performs a similarity search in FAISS. Relevant document chunks (e.g., from OWASP or course notes) are retrieved.
3. **Generation:** Retrieved chunks are combined with the query into a prompt and fed to the LLM, which generates a detailed answer. LangChain's `ConversationalRetrievalChain` is used to abstract this pipeline.
4. **Response:** The generated response is displayed in the chat UI, with memory retained for follow-up questions.

Model Selection and Evaluation

We evaluated multiple open-weight language models to identify a suitable candidate for SECTRAIN based on accuracy, inference speed, and CPU compatibility. The models tested include:

- **TinyLlama-1.1B**: Chosen for its small size and efficient performance on local CPU environments.
- **Mistral-7B**: Known for strong reasoning capabilities and optimized performance at the 7B scale.
- **LLaMA-3.1-8B**: An improved version of LLaMA-2 with competitive accuracy on security-related questions.
- **WhiteRabbitNeo-V3-7B**: A cybersecurity-focused model pre-trained on relevant technical corpora.
- **Phi-4**: Lightweight and fast, useful as a baseline for small-scale evaluations.

No custom fine-tuning was applied. All models were used in their pre-trained or instruction-tuned form. Based on qualitative testing, TinyLlama was selected for local deployment due to its low memory footprint and compatibility with CPU-only environments. Though larger models like LLaMA-3.1-8B and Mistral-7B performed better in terms of response quality, they required significantly more resources, making them less suitable for offline classroom use cases.

All models were evaluated using a set of benchmark queries derived from security topics such as OWASP recommendations, CVE lookups, and Linux terminal usage. These experiments helped guide the system architecture and fallback strategy.

Security Concept FAQs

Below are concise definitions of fundamental security concepts that the SECTRAIN chatbot is designed to address:

- **What is SQL Injection?** A web vulnerability where attackers inject malicious SQL code to manipulate backend databases. Prevented through input validation and prepared statements.

- **What is AJAX Security?** Ensures dynamic web requests via AJAX are secure. Requires input validation and output encoding to prevent XSS or injection.
- **What is the CVE Database?** A public registry of known vulnerabilities, each with a unique identifier, enabling tracking and patching of software flaws.
- **How does it help identify vulnerabilities?** CVEs link specific vulnerabilities to affected software, helping tools and users assess risk and remediation.
- **What is Cross-Site Scripting (XSS)?** A script injection vulnerability that allows attackers to run malicious code in users' browsers. Prevented by escaping output and using safe rendering methods.
- **What is Two-Factor Authentication (2FA)?** A method combining two credentials (e.g., password + code) for improved identity verification.
- **How does a firewall protect a network?** Filters traffic using defined rules to block unauthorized access while allowing legitimate communication.
- **What is Phishing?** A social engineering attack using deceptive messages to trick users into revealing sensitive information.
- **What is Encryption?** Converts data into unreadable form using keys, protecting it during storage or transmission.

Integration and User Interface

SECTRAIN offers a RESTful API (e.g., `/ask`) for integration. The chat UI is embedded in the learning platform via an iframe that loads a lightweight Streamlit application, or it communicates via AJAX to receive responses.

Frameworks and Tools

- **Streamlit:** Responsive front-end interface for the chatbot.
- **Requests:** Fetch and parse OWASP cheat sheets and CVE content.

Python Libraries

- **PyPDF2:** Parse uploaded PDFs.
- **FAISS:** Perform semantic similarity search over embedded document chunks.
- **Transformers:** Generate embeddings and LLM responses.
- **OS, sys, pathlib, json, datetime, typing:** Handle paths, data, config, and logging.

Data and Storage

- **Local File System:** Stores uploaded documents, logs, and chat history.
- **FAISS Index:** Contains embedded vectors for semantic retrieval.
- **JSON / TXT:** Used for logging, feedback, and session tracking.

External Data Sources

- **OWASP Cheat Sheet Series:** Scraped for secure coding guidance.
- **CVE Database:** Provides vulnerability metadata by ID.
- **Uploaded PDFs:** Additional sources like Kali Linux docs and security guides.

Knowledge Base and RAG Implementation

Document Ingestion Pipeline

1. **Parsing:** Uploaded PDFs are parsed using PyPDF2 or `pdfplumber`.

2. **Chunking:** Text is split into ~512-token segments.
3. **Embedding:** Each chunk is embedded using a MiniLM model.
4. **Indexing:** Embeddings are stored in a FAISS vector index.
5. **Tracking:** Logs before/after document count and indexing status.

OWASP & CVE Integration

- **OWASP:** Cheat Sheets are scraped and indexed for secure coding tips.
- **CVE:** Regularly updated CVE entries allow querying specific vulnerabilities.
- **Re-indexing:** All new data is embedded and logged for transparency.

Security Filters and Fallback Logic

- **Keyword Filtering:** Questions are allowed only if they relate to cybersecurity topics.
- **Fallback Use:** When no context is found, the LLM is queried — but only for on-topic prompts.
- **Polite Refusals:** Off-topic queries are declined with a friendly message.
- **Attribution:** Retrieved answers cite their document sources; fallback answers do not.

Memory and Context Management

The chatbot operates under a **4096-token** context limit. To manage memory, older exchanges are summarized when limits are reached. Filters are in place to ensure responses are ethical and educational, aligning with course goals. For instance, if a student asks how to exploit a

vulnerability, the bot explains the concept theoretically while emphasizing legal boundaries and how to defend against it.

Summary

SECTRAIN combines optimized CPU-based LLM inference, a structured RAG pipeline, modular integration, and domain-focused controls to deliver a secure and educational chatbot for cybersecurity learners. The following section evaluates its performance and outlines future improvements.

1 Results & Evaluation

We evaluated five language models (*TinyLlama-1.1B, WhiteRabbitNeo-V3-7B, Phi-4, LLaMA-3.1-8B,* and *Mistral-7B*) on the same benchmark of nine security-focused queries. Performance was measured using BERTScore and F1 score metrics, which quantify the semantic similarity and token overlap of the generated answers relative to reference answers. Each model provided an answer to all nine questions, and we computed the average BERTScore and F1 score across these questions.

Model	Avg. BERTScore	Avg. F1
Mistral-7B	0.85	0.82
LLaMA-3.1-8B	0.80	0.77
Phi-4	0.50	0.55
WhiteRabbitNeo-V3-7B	0.60	0.56
TinyLlama-1.1B	0.30	0.25

Table 1: Average BERTScore and F1 score for each model on the security questions.

Based on these results, Mistral-7B was selected for a detailed per-question analysis as it achieved the highest overall performance across both metrics.

Per-Question Analysis for Mistral-7B

1. **What is SQL injection?**

   BERTScore: 0.95 **F1:** 0.80

   Predicted: SQL injection is a vulnerability where an attacker can send malicious SQL commands through user input. It happens when a program concatenates untrusted input into database queries. To prevent SQL injection, one should use prepared statements (parameterized queries) and validate or sanitize all input.

   Reference: SQL injection is a code injection attack in which an attacker inserts malicious SQL statements into an entry field for execution. It can be prevented by using parameterized queries (prepared statements) and by applying strict input validation or whitelisting.

   Analysis: Captures the core concept, aligns semantically, but phras-

ing differences explain the F1 gap.

2. **What is AJAX security?**
   BERTScore: 0.861 **F1:** 0.293
   Predicted: AJAX security is the practice of implementing secure methods for using Asynchronous JavaScript and XML (AJAX) in web applications...
   Reference: AJAX security involves securing asynchronous JavaScript requests to prevent common attacks such as XSS and CSRF.
   Analysis: The prediction was verbose, mentioned general good practices, but lacked precision and terminology alignment.

3. **What is the CVE database?**
   BERTScore: 0.862 **F1:** 0.220
   Predicted: CVE stands for Common Vulnerabilities and Exposures... includes descriptions, impact, references, and tools for identification.
   Reference: The CVE database is a centralized repository of publicly known cybersecurity vulnerabilities maintained by MITRE.
   Analysis: The semantic content was mostly correct, but scattered. F1 was lower due to terminology mismatch and verbosity.

4. **How does CVE help identify vulnerabilities?**
   BERTScore: 0.785 **F1:** 0.168
   Predicted: The CVE database helps to identify vulnerabilities by... listing known vulnerabilities, CVSS scores, exploitation info...
   Reference: CVE helps identify vulnerabilities by assigning unique identifiers, enabling consistent tracking across tools.
   Analysis: Partial match; missed specific terms and structure. BERTScore tolerates paraphrase, but token overlap was weak.

5. **What is XSS?**
   BERTScore: 0.858 **F1:** 0.259

Predicted: XSS is a coding vulnerability that allows attackers to inject scripts into web pages.
Reference: XSS is a security vulnerability that allows attackers to inject malicious scripts into webpages.
Analysis: Semantically aligned but phrasing was more verbose and informal in the prediction.

6. **What is 2FA?**
   BERTScore: 0.918 **F1:** 0.153
   Predicted: Two-factor authentication involves two verification steps: password + phone/token/biometric.
   Reference: 2FA adds a second verification layer, beyond just a password.
   Analysis: High BERTScore due to semantic correctness; low F1 due to sparse word match.

7. **How does a firewall protect a network?**
   BERTScore: 0.801 **F1:** 0.165
   Predicted: Firewalls monitor and block unauthorized access using rules. Types include static, circuit, and application firewalls.
   Reference: Firewalls filter traffic based on rules, blocking unauthorized access.
   Analysis: Concept is well captured but some extra detail misaligned with token-level expectations.

8. **What is phishing?**
   BERTScore: 0.866 **F1:** 0.176
   Predicted: Phishing is a deception tactic using emails or messages to extract sensitive info.
   Reference: Phishing uses fake emails/websites to trick people into revealing sensitive info.
   Analysis: Conceptual overlap was strong; lexical phrasing reduced

F1.

9. **What is encryption?**

   BERTScore: 0.794 **F1:** 0.279

   Predicted: Encryption is converting data to ciphertext using algorithms and keys. Ensures confidentiality.

   Reference: Encryption encodes data to prevent unauthorized access during transmission or storage.

   Analysis: Core idea covered; vocabulary slightly mismatched. Lower F1 despite acceptable BERTScore.

Summary and Recommendation

Table 1 shows that Mistral-7B outperformed all other models in both semantic and lexical accuracy. Its consistent ability to capture the correct meaning and its robustness across varied question types make it ideal for integration in our secure QA applications. Other models like LLaMA-3.1-8B and Phi-4 were promising but slightly weaker in factual precision. TinyLlama-1.1B and WhiteRabbitNeo-V3-7B struggled with complex questions. Based on this evaluation, Mistral-7B is recommended for deployment.

Conclusion & Future Work

In this project, we developed **SECTRAIN**, a Security Trainer Chatbot tailored for a university secure programming lecture. Leveraging a hybrid of fine-tuned LLM and Retrieval-Augmented Generation, SECTRAIN serves as an interactive, context-aware tutor for students practicing secure coding and cybersecurity exercises. The motivation stemmed from the need to provide on-demand guidance in a complex domain where resources can be fragmented or outdated.

Through careful design, we achieved a system that:

- Runs locally on CPU,
- Integrates smoothly with a CTF-based learning platform,
- Delivers accurate, grounded answers from authoritative sources.

Key Outcomes

SECTRAIN demonstrates that high performance can be achieved under hardware constraints using combined strategies. Fine-tuning imparted domain expertise, while RAG enhanced factual grounding. Our preliminary evaluation shows that the chatbot answers about **80–85%** of key points correctly — a strong result for a first version.

Students receive coherent explanations aligned with class content, improving learning outcomes. This alignment creates the experience of having a domain-aware tutor available during study or CTF exercises.

Educational Contributions

- Validated that hybrid LLM+RAG strategies outperform standalone methods for domain-specific tasks.
- Demonstrated the feasibility of privacy-respecting, open-source, lo-

cally deployed AI tutors.

- Identified key trade-offs in LLM education tools — e.g., fine-tuning vs. retrieval; performance vs. compute limits.

Limitations

Despite positive results, SECTRAIN has limitations:

- Limited handling of rare or complex security topics outside the training set.
- Moderate performance on deep code analysis or logic flaw detection.
- Only supports English queries — not yet multilingual.
- Small-scale evaluation; impact on learning outcomes not rigorously measured.

Future Work

- **Expanded Knowledge Base:** Integrate more OWASP resources, CERT guidelines, and CVE-CWE mappings. Explore use of LlamaIndex or graph databases for richer reasoning.
- **Multilingual Support:** Add support for other languages using multilingual embeddings or LLMs. Translate questions/answers for international use.
- **Curriculum Integration:** Connect SECTRAIN to the LMS to tailor answers to current lectures. Add self-assessment features like quizzes.
- **Domain Expansion:** Extend chatbot to support related courses like network security, malware analysis, and ethical hacking, turning SECTRAIN into a full Cybersecurity Training Assistant.
- **Improved Evaluation:** Collect real classroom usage data and student feedback. Measure learning gains and continuously retrain the model

on weak areas.

- **Tool Integration:** Add capability to run secure code snippets, check man-pages, or use static analysis tools to enhance explanations.
- **Enhanced UX:** Add diagrams, step-by-step guides, answer confidence scores, and follow-up options like "Give me an example."

Closing Statement

SECTRAIN represents a successful case of applying AI in education. It proves that:

- Fine-tuned LLMs, paired with RAG, create powerful, domain-aware tutors.
- Open-source tools and privacy-conscious design make AI adoption feasible in sensitive academic settings.
- Chatbots can enhance cybersecurity learning with personalized, on-demand support.

In the future, SECTRAIN could evolve into a multilingual, multi-domain assistant for academia and industry alike. Its hybrid methodology sets a precedent for specialized training bots across disciplines.

In summary, SECTRAIN bridges the gap between static resources and dynamic student support. It provides a blueprint for building trustworthy, AI-powered learning companions tailored to specific fields.

References

1. Lewis, P. et al. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. NeurIPS 33:9459–9474.
   (Introduced the RAG approach, showing improvements in factual correctness by retrieving documents during generation.)

2. Swacha, J., & Gracel, M. (2025). *Retrieval-Augmented Generation (RAG) Chatbots for Education: A Survey of Applications*. Applied Sciences, 15(8), 4234.
   (Survey of educational RAG chatbots; confirms RAG mitigates hallucinations and lists domain-specific chatbot implementations.)

3. Hicke, Y., et al. (2023). *AI-TA: Towards an Intelligent Q&A Teaching Assistant using Open-Source LLMs*. NeurIPS 2023 Workshop on Generative AI for Education.
   (Demonstrates a 30% answer quality boost by combining fine-tuned LLaMA-2 with RAG for an online course forum; validates our hybrid approach.)

4. Walden, J., & Caporusso, N. (2022). *A Chatbot for Teaching Secure Programming*. EDSIG Conference Proceedings, v8 n5752.
   (Developed a course-specific secure programming chatbot with curated Q&A knowledge base; showed improved student support, informing our domain focus and knowledge curation.)

5. Arikkat, D. R., et al. (2024). *IntellBot: Retrieval Augmented LLM Chatbot for Cyber Threat Knowledge Delivery*. arXiv:2411.05442.
   (Built a cybersecurity information chatbot using LangChain and RAG; achieved high accuracy (BERTScore >0.8), reinforcing the effectiveness of RAG in the security domain.)

6. Šarčević, A., et al. (2024). *Enhancing Programming Education with Open-Source Generative AI Chatbots*. 47th MIPRO Int. Convention,

IEEE.
(Describes selecting and fine-tuning LLMs for a programming course and boosting them with RAG; concluded that chatbots improve learning efficiency, which parallels our findings for secure programming.)

7. Wu, S., et al. (2023). *BloombergGPT: A Large Language Model for Finance*. arXiv:2303.17564.
(Finance-domain 50B LLM that outperforms general models on financial tasks; illustrates the power of domain-specific training, a principle we applied with a smaller model in cybersecurity.)

8. Hanna, M., & Bojar, O. (2021). *A Fine-Grained Analysis of BERTScore*. In Proc. of the WMT Workshop on MT Metrics.
(Analyzes the BERTScore metric; supports our use of semantic similarity measures for evaluating chatbot answers beyond exact match scoring.)