CyberSecBot: A LangChain-Powered OWASP Training Chatbot

Nishant Patil, Tejas Wagh
Department of Computer Science and Engineering
Indian Institute of Technology Ropar
Emails: 2022csb1097@iitrpr.ac.in, 2022csb1144@iitrpr.ac.in

***Abstract*—CyberSecBot is an intelligent conversational assistant designed to support cybersecurity learning and OWASP-based vulnerability training. It leverages modern LLM (Large Language Model) architectures through Ollama's local Llama3:8B model, enhanced with LangChain for memory management and document retrieval. ChromaDB serves as a vector store for semantic search, while a Node.js + Express backend provides APIs to a React-based frontend. The system offers context-aware responses to cybersecurity queries and enables future extensions such as file uploads and interactive challenge guidance.**

***Index Terms*—Chatbot, LangChain, Llama3, Ollama, ChromaDB, OWASP, Cybersecurity Training, RAG (Retrieval-Augmented Generation)**

I. Introduction

Cybersecurity training is a critical aspect of modern digital literacy. However, learners often face difficulties navigating large sets of security guidelines and practical challenges. The **CyberSecBot** project aims to simplify cybersecurity learning using a domain-specific conversational agent.

This system provides a chatbot capable of understanding cybersecurity-related queries and answering them contextually using curated OWASP documentation. The chatbot runs locally, ensuring data privacy and customizability.

II. System Architecture

The proposed **CyberSecBot** architecture follows a modular design to ensure scalability, adaptability, and maintainability. It integrates a React-based frontend, an Express.js backend, LangChain for intelligent query processing, ChromaDB for vector storage, and Ollama for local language model inference. The high-level workflow and component interactions are explained below.

A. RAG Workflow

The overall workflow of the Retrieval-Augmented Generation (RAG) pipeline used in our chatbot is illustrated in Figure 1. It highlights the process of document ingestion, text splitting, embedding, vector storage, and retrieval for contextual response generation.

B. System Flow Diagram

The complete end-to-end data flow of the chatbot, from user interaction to response generation, is shown in Figure 2. This diagram illustrates the communication between the frontend, backend, vector database, and large language model components.

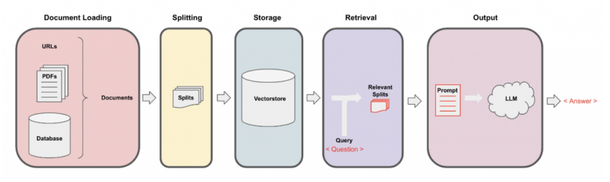

Fig. 1. Retrieval-Augmented Generation (RAG) pipeline used in CyberSecBot.

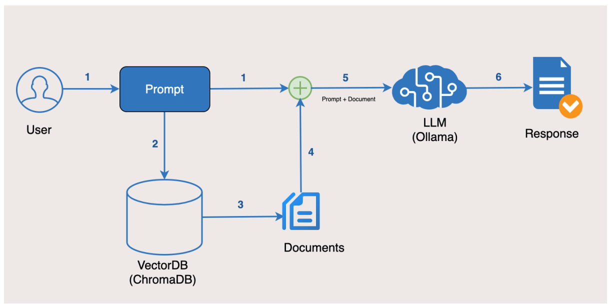

Fig. 2. Overall CyberSecBot system architecture showing the flow between components.

The system consists of five major components:

C. Frontend (React.js)

A lightweight user interface built with React for interactive chatting. It supports conversation history and can be extended for file or document uploads.

D. Backend (Node.js + Express)

Provides RESTful endpoints for chat messages, document ingestion, and memory management. It acts as a bridge between the frontend, LangChain, and ChromaDB.

E. LLM Integration (Ollama + Llama3:8B)

The Ollama runtime hosts the Llama3:8B model locally, which processes queries and generates natural language responses without cloud dependency.

F. LangChain Framework

LangChain manages model orchestration, embedding generation, prompt templates, and conversation memory. It also handles retrieval from the vector database using the RAG pipeline.

G. ChromaDB (Vector Database)

ChromaDB stores the embedded representations of cybersecurity documents (.md or .txt files). It enables semantic search and context retrieval for user queries.

III. Methodology

A. Data Preparation

The chatbot uses cybersecurity documents (mainly OWASP guides and challenge explanations) in Markdown or plain-text format. Each document corresponds to a specific vulnerability or topic (e.g., SQL Injection, XSS, CSRF).

```
Title: SQL Injection
Description: SQL injection
occurs when untrusted input is
concatenated into SQL queries...
Mitigation: Use parameterized
queries and input sanitization.
```

B. Embedding and Storage

- **Embedding Model:** LangChain's default embedding model (or local embedding if configured).
- **Vector Store:** ChromaDB running inside Docker with persistent volume mapping for long-term storage.

C. Query Flow

1) User enters a query on the frontend.
2) Backend sends it to LangChain's pipeline.
3) LangChain retrieves semantically relevant documents from ChromaDB.
4) Context + query is fed into the Ollama-hosted Llama3:8B model.
5) Model generates a contextual and conversational response.
6) Response and query are stored in memory for continuity.

D. Conversation Memory

LangChain's `ConversationBufferMemory` is integrated to maintain session-based context, allowing the chatbot to remember prior exchanges.

IV. Implementation Details

A. Backend Setup

Dependencies installed:

```
npm install express @langchain/core
@langchain/community
@langchain/ollama chromadb
```

Endpoint	Method	Description
/api/chat	POST	Send a user query and get chatbot response

TABLE I
BACKEND API ENDPOINTS

B. Backend API Endpoints

C. ChromaDB Docker Configuration

```
docker run -d \
  -p 8000:8000 \
  -v ./chromadb_data:/chromadb/data \
  chromadb/chroma:latest
```

This ensures persistence of vectors between sessions.

D. LangChain + Ollama Integration

```
import { Ollama } from "@langchain/ollama";

const model = new Ollama({
    model: "llama3:8b",
    baseUrl: "http://localhost:11434"
});
```

V. Results and Discussion

CyberSecBot successfully responds to cybersecurity-related queries, including OWASP vulnerability explanations, threat classifications, and prevention techniques. The retrieval augmentation from ChromaDB enhances factual accuracy, while Llama3 ensures natural, conversational tone.

Sample Query:

```
What is Cross-Site Scripting (XSS)
and how to prevent it?
```

Response:

Cross-Site Scripting (XSS) allows attackers to inject malicious JavaScript into webpages viewed by other users. It can be mitigated by escaping output, validating input, and using Content Security Policy (CSP).

VI. Future Work

1) **Challenge Integration:** Connect chatbot responses with practical CTF challenges.
2) **Document Upload:** Allow admins to upload new materials from the frontend.
3) **User Authentication:** Add user accounts and training progress tracking.
4) **Multi-agent Setup:** Introduce specialized agents for different cybersecurity domains.
5) **Deployment:** Containerize the backend and frontend using Docker Compose.

VII. Conclusion

CyberSecBot demonstrates the potential of locally hosted, privacy-preserving AI systems for domain-specific education. Using LangChain and Ollama's open models, it enables dynamic interaction with cybersecurity materials, bridging the gap between theory and practical training.

References

[1] OWASP Foundation. *OWASP Top 10 - 2023*. [Online] Available: https://owasp.org

[2] LangChain Documentation. https://js.langchain.com/

[3] Ollama. https://ollama.ai

[4] ChromaDB. https://docs.trychroma.com/

[5] Meta AI, *LLaMA 3 Model Card*, 2024.