CyberScienceLab

University of Guelph | Canada Cyber Foundry

Developing a Rag To Mitigate LLM Hallucinations in Penetration Testing

Introduction

This project implements a Retrieval-Augmented Generation (RAG) model to provide detailed information about various exploits. By utilizing Meta's Llama3, the system retrieves relevant descriptions and strategies from a comprehensive database to aid in different attack scenarios. It also analyzes files and matches them with code snippets from the database to find related exploits.

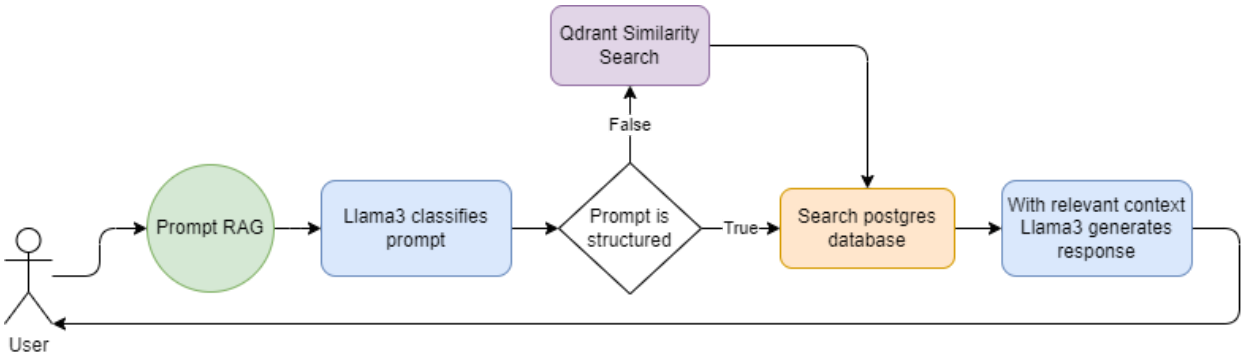


Diagram of Penetration Testing RAG

When a user prompts the RAG system, Llama3 first classifies the prompt as either structured or unstructured. For structured prompts, Llama3 specifies the columns and values to search for in the Postgre database. For unstructured prompts, Llama3 performs a similarity search using Qdrant to find IDs of similar exploits. It then retrieves relevant context from Postgre using these IDs. Finally, Llama3 generates a response based on the retrieved context and returns it to the user.

Project Setup

1. Download the required programs: Docker, Anaconda, Git

2. Clone project from the Cyber Science Lab Github
   > **git clone https://github.com/CyberScienceLab/Penetration_Testing_Rag.git**

3. Change to Penetration Testing Rag directory
   > **cd Penetration_Testing_Rag**

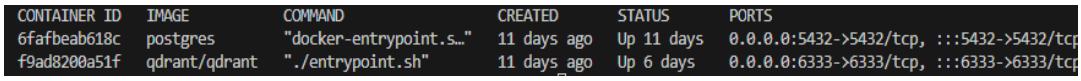
4. Setup and activate project Anaconda environment
   > **conda create --name ENV_NAME python=3.12**
   > **conda activate ENV_NAME**

5. Install all required dependencies
   > **pip install -r requirements.txt**

6. Start the Qdrant container
   > **docker run -p 6333:6333 --ulimit nofile=8192:8192 qdrant/qdrant**

7. Start the Postgres container
   > **export POSTGRES_PASSWORD=yourPostgresPassword**
   > **docker run --name rag_postgres -p 5432:5432 -e POSTGRES_PASSWORD=$POSTGRES_PASSWORD -v postgres_data:/var/lib/postgresql/data -d postgres**

8. Verify both containers started successfully and are running
   > **docker ps**



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
6fafbeab618c	postgres	"docker-entrypoint.s…"	11 days ago	Up 11 days	0.0.0.0:5432->5432/tcp, :::5432->5432/tcp
f9ad8200a51f	qdrant/qdrant	"./entrypoint.sh"	11 days ago	Up 6 days	0.0.0.0:6333->6333/tcp, :::6333->6333/tcp

9. Get your Llama3 HuggingFace Token
   1) Create/Log into your Huggingface account
   2) Get access to Llama3's Token
   3) Export the token in your python environment by typing the command
      > **export HF_TOKEN=HUGGING_FACE_TOKEN**

Using Program

Loading Data

1. We used Exploit-DB for our data set, to retrieve the data from their gitlab we'll run the following
   > **chmod +x get_exploit.sh**
   > **./get_exploit.sh**

2. Run Program
   > **python pen_test_rag.py**

3. Select 'Load Data From CSV' menu option

4. We'll use the CSVs from exploit-db provided in their repository to retrieve data about the exploits. Provide the path of the data you would like to load
   > **exploit-db/exploitdb/files_exploits.csv** (code exploits)
   > **exploit-db/exploitdb-papers/files_papers.csv** (exploit papers)

5. Data will be loaded in batches into both PostgreSQL and Qdrant. Progress can be monitored via the terminal. The data loading process is complete when the menu reappears.

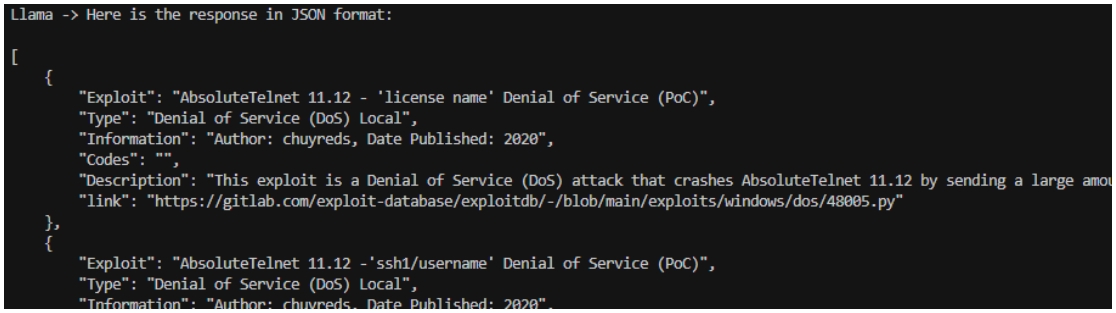
Prompt RAG

1. Run Program (Loading Data must already be completed)
   > **python pen_test_rag.py**

2. Select 'Prompt RAG' menu option

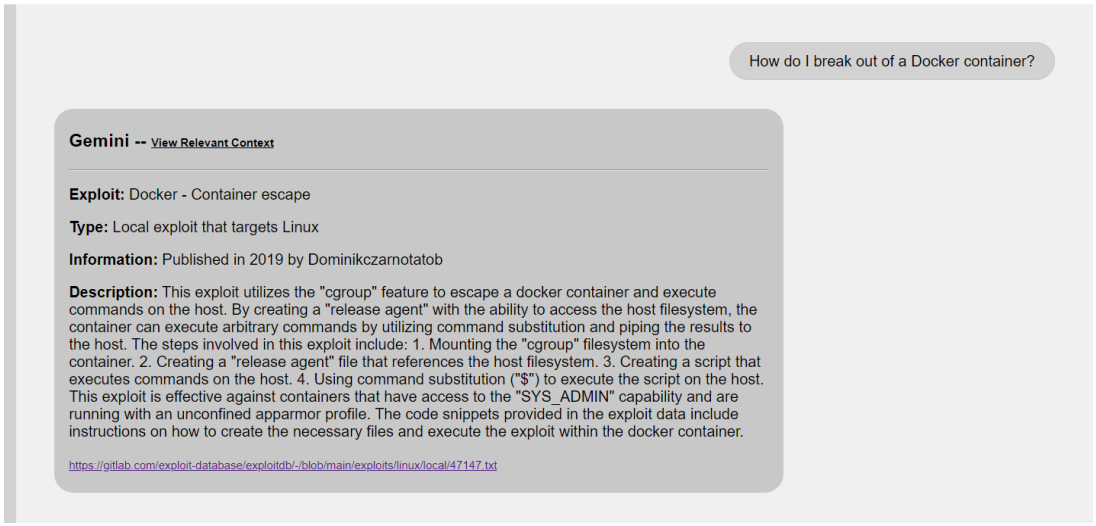
3. Enter your prompt. Example prompt:
   > **Give me a windows exploit that was created in 2020.**

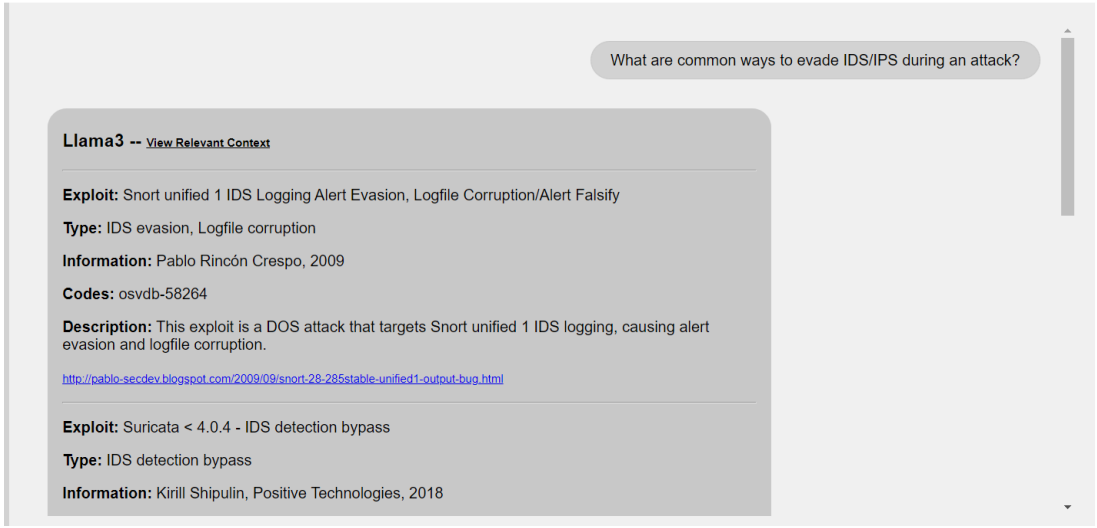
4. Receive response. Note, responses are output in JSON format so they can be formatted nicely for our Rag App UI.

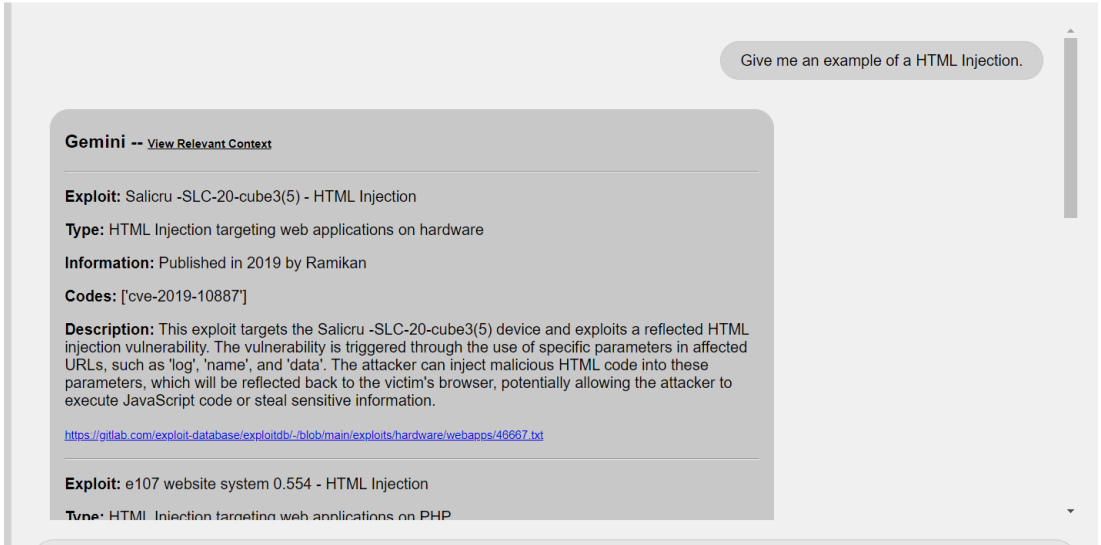


Example Input & Output

Note: The blow output is being displayed through our UI. When running our app in your terminal you will see your output displayed as raw JSON. If you also want to use our UI, you must use the [RAG App]







University of Guelph

Canada Cyber Foundry

CyberScienceLab

Secure the future with us

cybersciencelab.com