

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.

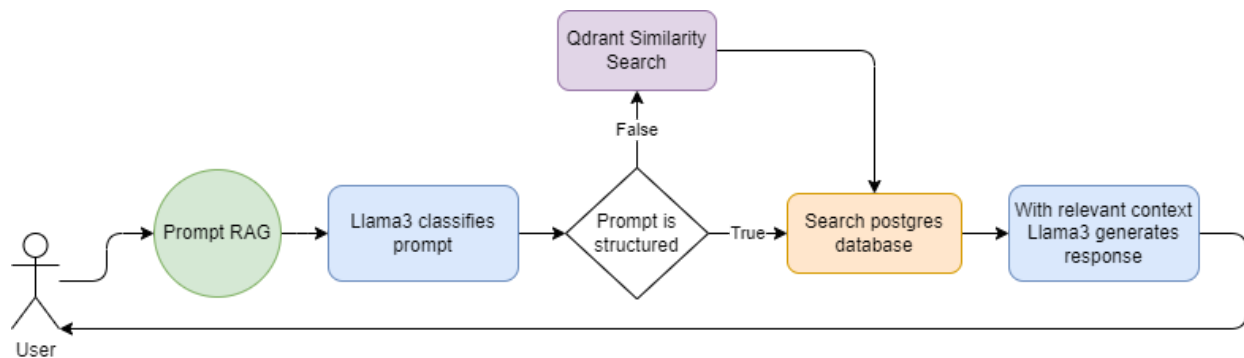


Diagram of Penetration Testing RAG

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.

```
Llama -> Here is the response in JSON format:
[
  {
    "Exploit": "AbsoluteTelnet 11.12 - 'license name' Denial of Service (PoC)",
    "Type": "Denial of Service (DoS) Local",
    "Information": "Author: chuyreds, Date Published: 2020",
    "Codes": "",
    "Description": "This exploit is a Denial of Service (DoS) attack that crashes AbsoluteTelnet 11.12 by sending a large amount of data to the server.",
    "link": "https://gitlab.com/exploit-database/exploitdb/-/blob/main/exploits/windows/dos/48005.py"
  },
  {
    "Exploit": "AbsoluteTelnet 11.12 - 'ssh1/username' Denial of Service (PoC)",
    "Type": "Denial of Service (DoS) Local",
    "Information": "Author: chuyreds, Date Published: 2020",
    "Codes": "",
    "Description": "This exploit is a Denial of Service (DoS) attack that crashes AbsoluteTelnet 11.12 by sending a large amount of data to the server.",
    "link": "https://gitlab.com/exploit-database/exploitdb/-/blob/main/exploits/windows/dos/48005.py"
  }
]
```

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](#)

How do I break out of a Docker container?

Gemini -- [View Relevant Context](#)

Exploit: Docker - Container escape

Type: Local exploit that targets Linux

Information: Published in 2019 by Dominikczarnotatob

Description: This exploit utilizes the "cgroup" feature to escape a docker container and execute commands on the host. By creating a "release agent" with the ability to access the host filesystem, the container can execute arbitrary commands by utilizing command substitution and piping the results to the host. The steps involved in this exploit include: 1. Mounting the "cgroup" filesystem into the container. 2. Creating a "release agent" file that references the host filesystem. 3. Creating a script that executes commands on the host. 4. Using command substitution ("\$(") to execute the script on the host. This exploit is effective against containers that have access to the "SYS_ADMIN" capability and are running with an unconfined apparmor profile. The code snippets provided in the exploit data include instructions on how to create the necessary files and execute the exploit within the docker container.

<https://gitlab.com/exploit-database/-/blob/main/exploits/linux/local/47147.txt>

What are common ways to evade IDS/IPS during an attack?

Llama3 -- [View Relevant Context](#)

Exploit: Snort unified 1 IDS Logging Alert Evasion, Logfile Corruption/Alert Falsify

Type: IDS evasion, Logfile corruption

Information: Pablo Rincón Crespo, 2009

Codes: osvdb-58264

Description: This exploit is a DOS attack that targets Snort unified 1 IDS logging, causing alert evasion and logfile corruption.

<http://pablo-secdev.blogspot.com/2009/09/snort-28-285stable-unified1-output-bug.html>

Exploit: Suricata < 4.0.4 - IDS detection bypass

Type: IDS detection bypass

Information: Kirill Shipulin, Positive Technologies, 2018

Give me an example of a HTML Injection.

Gemini -- [View Relevant Context](#)

Exploit: Salicru -SLC-20-cube3(5) - HTML Injection

Type: HTML Injection targeting web applications on hardware

Information: Published in 2019 by Ramikan

Codes: [cve-2019-10887]

Description: This exploit targets the Salicru -SLC-20-cube3(5) device and exploits a reflected HTML injection vulnerability. The vulnerability is triggered through the use of specific parameters in affected URLs, such as 'log', 'name', and 'data'. The attacker can inject malicious HTML code into these parameters, which will be reflected back to the victim's browser, potentially allowing the attacker to execute JavaScript code or steal sensitive information.

<https://gitlab.com/exploit-database/-/blob/main/exploits/hardware/webapps/46667.txt>

Exploit: e107 website system 0.554 - HTML Injection

Type: HTML Injection targeting web applications on PHP



CANADA CYBER FOUNDRY



CYBERSCIENCELAB

Secure the future with us

cybersciencelab.com