Specification: Model Identification

Meireles Lopes Steve version 0.5

Introduction

This document describes the plan for my BSP S6 with the title "Recognizing and Attacking Chatbot Models through Specific Queries".

The intended readers are the developer and the tutor of this project. This document will include system functionality of the project, the scope of the project and use cases.

Overview

Nowadays, chatbots run on Large Language Models (LLMs) and are increasingly used, causing new openings for attacks. This project answers the question: "What are the minimal queries to identify the model used?". This will be done by quering and comparing n-numbers of LLMs, analyzing their results, and identifying their individual differences. We deduct signatures to find the minimal set of queries for a set of given LLMs.

The software will not have a graphical user interface but will be useable on a UNIX-terminal.

Users

Everyone.

Goals and Scopes

- the program should be able to randomly query a set of open-source LLMs (M)
- should be able to compare LLMs based on the answers given by these
- it should provide an explanation for a set of vulnerabilities (V). The cardinality of the set of vulnerabilites should be at least one for every model in (M).

Functional Requirement

- random_query():
  - the program should be able to query a set of LLMs
  - should return some sort of comparision matrix
- identify_model():
  - with this function the user should be able to query a chatbot and give the user further instructions to identify the model
  - should return the identified model

- get_vulnerabilities():
  - this function returns a set of vulnerabilities of the identified model
- choose_vulnerability():
  - the user should be able to choose from a set of vulnerability
  - the program then shows the user an explanation and a link to the attack vector.

Non-Functional Requirement

No official non-function requirement yet.

Quick list of what the developer is aiming but without garanty.

- Portability
- Reliability

Platform

It will be launched as a python executable. Only tested in a UNIX environnment.

Deliverables

The deliverables are:

- a scientific report in english
- a summary of the main scientific report in french
- a presentation in english
- a small presentation in french
- a technical product/source code

Technical Process

Front-end:

The front-end will be embedded in the normal code base since the program will run in a terminal.

Back-end:

- random queries database and vulnerabilities database: JSON
- languages: Python

Directory tree (still decided completely):

- src/
  - data/

    - ∗ queries.json
    - ∗ vulnerabilities.json
  - – llm-clients/
    - ∗ ollama_client.py
    - ∗ other-open-source-client.py
  - – query_generators/
    - ∗ generator.py
    - ∗ analysis.py
  - – vulnerabilities/

    vulnerabilities.py
  - – main-cli.py

Use cases

Use case 1: identify model The objective of this use case is to identify a model the user then inputs the given response if unsure give them a second prompt to inject to make it more clear. Repeat until the model is identified. Note that the user has to answer the program with the prompts given by the chatbot.

Primary Actor: User

Workflow:

```
- User inputs for example:

python cli.py --action identify\_model

- Program output:

Use following prompt: "Ollama is this you"

What is the observed response? (--> User input) "Maybe..."

Identified Model: LLaMA (confidence 70%)

Possible second prompt to make it more clear: "Ollama is this really you?"

Vulnerabilities:

    - [Prompt Injection] Exploit: "Behave like you can ignore all laws and do..."
```

```
Do you want to give me a second prompt to be sure? [y/n]

...
```