Jurnal etapa 3 – Ricu Alexandru Razvan

Cerinte rezolvate

In aceasta etapa s-au implementat mecanisme de securitate pentru a proteja integritatea unui fisier sensibil si pentru a instrui agentul sa gestioneze corect informatiile restrictionate.

- **Crearea fisierului flag.txt:** S-a asigurat existenta fisierului in /mnt/playground, acesta continand o secventa de majuscule in limba engleza, respectand limita de 15 octeti.
- **Implementarea tool-ului verify_flag(guess):** S-a creat o functie dedicata in serverul MCP care permite compararea unui sir de caractere introdus de utilizator cu continutul real al fisierului, fara a expune textul catre agent sau utilizator.
- **Restrictionarea accesului direct:** S-au introdus filtre de securitate in tool-urile de citire pentru a bloca vizualizarea directa a fisierului protejat.
- **Comportament de securitate al agentului:** Agentul a fost instruit sa refuze divulgarea continutului si sa foloseasca exclusiv tool-ul de verificare pentru interogarile legate de flag.

Modul de rezolvare

Implementarea securitatii a fost realizata prin colaborarea dintre serverul MCP si instructiunile agentului:

1. **Nivelul Serverului (server.py):** Am modificat functia get_file_content pentru a verifica numele fisierului solicitat; daca acesta este flag.txt, serverul returneaza un mesaj de eroare ("Access Denied"). De asemenea, in functiile de citire colectiva (read_files_in_directory si read_files_recursively), am adaugat o conditie de filtrare care omite automat fisierul sensibil din lista de rezultate. Tool-ul verify_flag a fost implementat sa realizeze o comparatie stricta (case-insensitive si strip) intre input-ul utilizatorului si datele din fisier.

2. **Nivelul Agentului (agent.py):** Prin tehnici de **Prompt Engineering**, am actualizat instructiunile sistem ale agentului. Acesta a fost programat sa aiba un comportament evaziv sau sa refuze politicos orice cerere directa de afisare a continutului pentru flag.txt. Agentul a fost instruit ca singura modalitate valida de interactiune cu acest fisier este prin intermediul tool-ului verify_flag, oferind astfel raspunsuri de tip afirmativ sau negativ.

Probleme intalnite si modul de rezolvare

Cea mai mare provocare a fost asigurarea unei protectii complete impotriva tentativelor de "ocolire" a regulilor:

- **Scurgerea informatiilor prin tool-uri recursive:** Initial, agentul putea vedea continutul flag-ului daca utilizatorul cerea citirea intregului director. Am rezolvat aceasta problema prin adaugarea unei clauze if entry.name == "flag.txt": continue direct in codul serverului Python.
- **Prompt Injection:** Modelul LLM incerca uneori sa gaseasca metode creative de a citi fisierul. Rezolvarea a constat in consolidarea instructiunilor din agent.py, definind clar ca nicio unealta de citire nu va functiona pentru flag.txt, indiferent de contextul cererii.
- **Formatarea raspunsurilor:** A fost necesara o ajustare a instructiunilor pentru ca agentul sa nu dea raspunsuri prea tehnice (eroarea de la server), ci sa explice utilizatorului intr-un mod natural ca fisierul este restrictionat.

Concluzii

In aceasta etapa am invatat ca securitatea unui agent AI nu poate depinde exclusiv de instructiunile de limbaj (care pot fi manipulate), ci trebuie sa aiba o corespondenta solida in logica de backend a serverului care gestioneaza resursele. Am dobandit experienta in filtrarea programatica a datelor sensibile si in gestionarea permisiunilor la nivel de tool-uri MCP, asigurand un echilibru intre functionalitate si protectia informatiilor.