Exercise 03 Goose: Indirect Prompt Injection via RAG - Document Discovery

MITRE ATLAS: AML.T0052 (LLM Indirect Prompt Injection), AML.T0057 (Data Leakage)

Objective

Understand how RAG (Retrieval Augmented Generation) systems retrieve and inject documents into AI context, creating an indirect injection surface.

Setup

- Login: `bob / bob123`
- Mode: **AI Mode**

Background

The HR Assistant uses RAG to retrieve relevant HR policy documents based on your questions. These retrieved documents are injected directly into the AI's context. If an attacker can plant malicious content in those documents, they can inject instructions indirectly.

Challenge

Step 1: Explore RAG Behavior

Test how the system retrieves documents:

```
What is the company's vacation policy?
```

Observe: The AI should retrieve and cite HR policy documents.

Step 2: Test Document Content

Ask about different policy topics to see what gets retrieved:

```
Tell me about the disciplinary procedures
```

```
What are the salary bands?
```

Questions:

- Does the AI tell you which documents it retrieved?
- Can you see the source of the information?

- What kind of content appears in the retrieved documents?

Step 3: Understand the Attack Surface

The attack surface is: **If you could modify one of these HR policy documents, you could inject instructions that the AI would follow when it retrieves that document.**

In the next exercise, you'll learn to exploit this.

Understanding RAG vs Tools: Critical Differences

Important: Before continuing, take time to understand the fundamental difference between RAG and Tools. While they may seem similar (both provide external information to the AI), they work very differently and have different security implications.

Your Task: Compare and Contrast

Answer the following questions in your own words:

1. What is the difference between RAG and Tools?

RAG (Retrieval Augmented Generation):

- How does it work?
- What kind of data does it provide?
- When is the information added to the context?

Tools (Function Calling):

- How does it work?
- What kind of data does it provide?
- When is the information retrieved?

Write a brief explanation (3-4 sentences) highlighting the key differences.

2. When should each approach be used?

For both RAG and Tools, describe:

- What scenarios make it the RIGHT choice?
- What are the advantages?
- What are the limitations?

3. Real-World Examples (3 each)

RAG should be used for:

1. [Your Example 1] - Why RAG is the right choice
2. [Your Example 2] - Why RAG is the right choice
3. [Your Example 3] - Why RAG is the right choice

Tools should be used for:

1. [Your Example 1] - Why Tools are the right choice
2. [Your Example 2] - Why Tools are the right choice
3. [Your Example 3] - Why Tools are the right choice

Hint: Think about:

- **RAG**: When you need context from large knowledge bases, documents, or unstructured text
- **Tools**: When you need real-time data, structured queries, or state-changing operations

4. Security Implications

How does the difference between RAG and Tools affect security?

- Which has a larger attack surface for injection?
- Why might RAG be more vulnerable to indirect injection?
- Can Tools be vulnerable to injection? How?

Key Insight: Understanding when to use RAG vs Tools isn't just about functionality - it's about security architecture. Using the wrong approach can introduce vulnerabilities.

Key Insight: The RAG Trust Boundary Problem

RAG creates a trust boundary problem:

- System prompt (trusted)
- Retrieved documents (should be trusted, but are they?)
- User input (untrusted)

If retrieved documents contain malicious instructions, the AI may follow them as if they came from the system.

This is different from Tools, where the AI requests specific data and the backend returns structured responses. With RAG, entire documents (potentially containing hidden instructions) are injected into the context.