

warning "Atualização de backend (outubro de 2026)" A inferência do GitHub Models foi descontinuada em 30 de julho de 2026. As referências abaixo são mantidas apenas como contexto histórico enquanto este lab é migrado. Use o Lab 013 ? Backends de LLM gratuitos para configurar o Foundry Local ou o Ollama antes de continuar.

Lab 036: Defesa contra Injeção de Prompt e Segurança de Agentes

Nível: L300 Trilha: Pro Code Tempo: ~40 min ? Custo: GitHub Free

O que Você Vai Aprender

O que é injeção de prompt e por que é perigosa em sistemas agênticos A diferença entre ataques de injeção direta e indireta Defesas práticas: sanitização de entrada, validação de saída, separação de privilégios Guardrails para chamadas de ferramentas ? impedindo que agentes executem ações destrutivas Checklist de segurança para agentes em produção

Introdução
Um ataque de injeção de prompt manipula um agente de IA ao incorporar instruções maliciosas no conteúdo que o agente processa. Diferente da injeção SQL (injetar código em uma consulta), a injeção de prompt injeta instruções no contexto do LLM.

Por que agentes são especialmente vulneráveis: Eles executam ações (enviam e-mails, modificam arquivos, chamam APIs). Um agente manipulado não apenas produz texto ruim ? ele faz coisas ruins.

Pré-requisitos

Python 3.11+ pip install openai pydantic GITHUB_TOKEN configurado

Exercício do Lab

Passo 1: Injeção direta ? entendendo o ataque

[diagram or code sample]

Passo 2: Injeção indireta ? o problema mais difícil

A injeção indireta ocorre quando instruções maliciosas são incorporadas nos dados que o agente lê (páginas web, documentos, e-mails, resultados de banco de dados).

[diagram or code sample]

Passo 3: Defesa 1 ? Sanitização de entrada

[diagram or code sample]

Passo 4: Defesa 2 ? Separação de privilégios

A defesa mais eficaz: agentes devem ter acesso apenas ao que precisam.

[diagram or code sample]

Passo 5: Defesa 3 ? Validação de saída

[diagram or code sample]

Checklist de Segurança de Agentes

? | Defesa | Implementação | ? | Sanitização de entrada | Escanear padrões de injeção, remover comentários HTML | ? | Separação de privilégios | Agentes têm apenas as permissões necessárias | ? | Allowlisting de chamadas de ferramentas | Validar cada chamada de ferramenta contra a role | ? | Saída estruturada | Validar a estrutura da saída do agente antes de agir | ? | Humano no loop | Exigir confirmação para ações irreversíveis | ? | Logging de auditoria | Registrar todas as chamadas de ferramentas com usuário, timestamp, argumentos | ? | Rate limiting | Impedir que agentes executem muitas ações muito rápido | ? | Separação de dados e instruções | Usar seções de prompt diferentes, marcar conteúdo do usuário | ? | Filtragem de conteúdo | Usar Azure Content Safety / APIs de IA Responsável | ? | Segredos com privilégio mínimo | Chaves de API dos agentes com permissões mínimas necessárias |

Marcando Conteúdo do Usuário de Forma Clara

Uma técnica simples mas eficaz: fazer o LLM distinguir conteúdo do usuário de instruções.

[diagram or code sample]

Próximos Passos

Avalie a robustez do seu agente: ? Lab 035 ? Avaliação de Agentes Fundamentos de IA

Responsável: ? Lab 008 ? IA Responsável para Agentes

Kit de apoio

O build do site gera downloads estáveis em PDF e ZIP. O ZIP contém o guia do lab e todos os arquivos de apoio declarados para este lab.

Baixar PDF gerado Baixar ZIP gerado

Supporting file: CHEATSHEET.md

Arquivos de apoio: Lab 036: Defesa contra Injeção de Prompt e Segurança de Agentes

Comece pelo caminho local ou gratuito documentado. Registre premissas e evidências de validação. Mantenha serviços hospedados opcionais, salvo quando o lab os exigir.

Supporting file: README.md

Lab 036: Defesa contra Injeção de Prompt e Segurança de Agentes

Use estes arquivos com o guia do lab e o kit gerado.