Relatório de Impacto à Proteção de Dados Pessoais (RIPD)
Sistema de Agentes Inteligentes em Ambiente Kubernetes

HÉLIO HENRIQUE MEDEIROS SILVA	202020553
LEONARDO ELIAS RODRIGUES	202020776
FILIPE DE OLIVEIRA VILAS BOAS	202120935
AARON MARTINS LEÃO FERREIRA	202120496
MAURÍCIO MARTINS DAMASCENO	202020884

Introdução

O presente trabalho aborda a implementação de um sistema experimental envolvendo agentes inteligentes em um ambiente containerizado, utilizando tecnologias modernas como Docker e Kubernetes (através do Minikube). Este projeto representa uma exploração prática da integração entre sistemas de inteligência artificial e arquiteturas de microsserviços, com foco especial na orquestração de containers e na análise dos riscos operacionais associados.

Contextualização

No cenário atual da computação, a utilização de agentes inteligentes tem se tornado cada vez mais comum em diferentes contextos aplicacionais. Paralelamente, a conteinerização de aplicações emergiu como uma prática fundamental no desenvolvimento de software moderno, oferecendo benefícios significativos em termos de isolamento, portabilidade e escalabilidade. A intersecção dessas duas tendências tecnológicas apresenta desafios únicos e oportunidades interessantes de investigação.

Etapa 6: Identificação e Avaliação dos Riscos

6.1 Análise STRIDE para Componentes de Hardware
(Focando apenas nos riscos aplicáveis ao projeto)

1. Spoofing (Falsificação)
- Risco: Alto
- **Ameaças Identificadas:**
  - Roubo de credenciais da API Server, feito por phishing ou vazamento de tokens
  - Se passar por administrador e executar comandos maliciosos no sistema
  - Spoofing de Pods, criando pods falsos para capturar tráfego, executar ataques MitM

2. Tampering (Alteração de Dados)
- Risco: Alto
- Ameaças Identificadas:
  - Modificação do ConfigMap, alterando configurações críticas do sistema
  - Modificação de Imagens de Container podem manipular/roubar dados e se propagar pelo ambiente

3. Repudiation (Repúdio)
- Risco: Médio
- Ameaças Identificadas:
  - Falha no rastreamento das requisições entre WebUI e Ollama
  - Perda de logs de execução do modelo deepseek-coder
  - Impossibilidade de identificar qual réplica processou determinada requisição

- Falhas no registro de uso compartilhado da GPU (time-slicing)

4. Information Disclosure (Vazamento de Informações)

- Risco: Alto

- Ameaças Identificadas:

- Exposição de logs com credenciais sensíveis como tokens de API, senhas e chaves privadas
- Exposição de Secrets do Kubernetes através de variáveis de ambiente
- Exposição da API Server do Kubernetes sem autenticação permitindo manipulação serviços

5.Denial of Service (Negação de Serviço)

- Risco: Alto

- Ameaças Identificadas:

- Sobrecarga da GPU devido ao time-slicing entre 4 réplicas
- Esgotamento do PVC (10Gi) devido a downloads de modelos
- Falha no initContainer durante carregamento do modelo
- Gargalo de memória (limitada a 6096MB)
- Conflitos no acesso ao volume ReadWriteOnce entre réplicas
- Timeout nas inferências do modelo devido a contenção de recursos

6. Elevation of Privilege (Elevação de Privilégio)

- Risco: Médio

- Ameaças Identificadas:

- Acesso privilegiado através do NodePort exposto (31434)
- Acesso não autorizado à interface web (porta 31000)
- Privilégios excessivos no acesso à GPU

6.2 Avaliação de Riscos com TORR

Classificado em 3 fatores principais:

- **Potencial de Exploração:** Qual a probabilidade da ameaça ser explorada?
- **Impacto:** Qual o dano causado se a ameaça for explorada?
- **Dificuldade de Mitigação:** O quão difícil é reduzir esse risco?

Tabela de Priorização dos Riscos

Ameaça	Potencial de Exploração	Impacto	Dificuldade de Mitigação	Prioridade
Exposição da API Server	Alta	Alta	Média	Crítica
Elevação de Privilégio nos Pods	Médio	Alta	Alta	Crítica
Ataques de negação de serviço	Média	Alta	Baixa	Alta
Vazamento de credenciais dos pods	Média	Alta	Média	Alta

Etapa 7: Medidas para Tratamento dos Riscos

7.1 Medidas de Mitigação por Categoria STRIDE

1. Spoofing

- Implementação de RBAC (Role-Based Access Control) para restringir permissões
- Usar autenticação multifator (MFA) para usuários administrativos
- Evitar armazenar credenciais diretamente no código-fonte
- Habilitar políticas de segurança para restringir comunicação entre pods'
- Utilizar certificados TLS para autenticar serviços internos
- Restringir a criação de pods não autorizados com Admission Controllers

2. Tampering

- Criptografar comunicações internas com TLS
- Habilitar políticas de segurança contra tráfego de pods não confiáveis
- Assinar imagens de container com Notary ou Sigstore
- Aplicar políticas de integridade para bloquear imagens de container não confiáveis
- Restringir permissões sobre ConfigMaps via RBAC (Role-Based Access Control)

3. Repudiation

- Implementação de logs detalhados em todos os containers
- Configuração do Kubernetes logging para manter histórico de execuções
- Monitoramento das ações realizadas pelos agentes
- Implementação de IDs únicos para cada requisição processada

4. Information Disclosure (Vazamento de Informações)

- Redigir logs sensíveis para remover dados confidenciais
- Utilizar volumes de Secrets em vez de variáveis de ambiente
- Restringir o acesso a Secrets via RBAC (Role-Based Access Control) e monitorar acessos
- Bloquear acesso externo à API Server

5. Denial of Service

- Definição adequada de recursos (requests e limits) nos pods
- Implementação de health checks nos containers
- Configuração de auto-scaling baseado em métricas de uso
- Monitoramento de performance do cluster
- Implementação de circuit breakers para evitar sobrecarga
- Definição de timeouts apropriados

6. Elevation of Privilege

- Configuração de RBAC (Role-Based Access Control) no Kubernetes
- Aplicação do princípio do menor privilégio nos containers
- Remoção de capabilities desnecessárias dos containers
- Uso de security contexts adequados nos pods

7.2 Medidas para Tratamento dos Riscos

1. Métricas Críticas

- Uso de GPU por réplica
- Tempo de inferência do modelo
- Uso do volume persistente
- Latência entre WebUI e Ollama
- Taxa de falhas no time-slicing

2. Alertas Prioritários

- GPU utilização > 90%

- Memória do pod > 80%
- Falhas no carregamento do modelo
- Volume persistente > 80%
- Latência de inferência > 5s

3. Procedimentos de Recuperação

- Script de backup do PVC
- Procedimento de reinicialização do GPU Operator
- Rollback automatizado em caso de falha
- Limpeza periódica de modelos não utilizados

Conclusão

O presente trabalho demonstrou a implementação bem-sucedida de um sistema de agentes inteligentes em ambiente containerizado, utilizando tecnologias modernas como Kubernetes, Docker e GPU sharing através do NVIDIA GPU Operator. A arquitetura implementada, baseada no Ollama para execução de modelos de IA (deepseek-coder:1.5b) e OpenWebUI para interface com usuário, representa uma solução experimental que equilibra complexidade técnica e praticidade operacional.

Resultados Alcançados

A implementação atingiu seus objetivos principais através de:

1. Containerização Eficiente

- Successful deployment de 4 réplicas do Ollama
- Configuração adequada do time-slicing de GPU
- Integração efetiva entre componentes através de services

2. Otimização de Recursos

- Compartilhamento eficiente de GPU entre múltiplas instâncias
- Gestão adequada de memória e CPU
- Persistência de dados através de PVC

3. Análise de Riscos

- Identificação de vulnerabilidades potenciais
- Proposição de medidas de mitigação
- Foco em aspectos operacionais críticos

Desafios Enfrentados

Durante o desenvolvimento, diversos desafios foram encontrados e superados:

1. Técnicos

- Configuração do GPU sharing
- Balanceamento de recursos entre réplicas
- Gestão de persistência de dados

- Orquestração de containers
- Configuração de networking

- Gestão de deployments

Limitações do Projeto

É importante reconhecer as limitações do trabalho atual:

1. Segurança
- Ausência de autenticação
- Exposição de portas via NodePort
- Controles de acesso básicos

2. Escalabilidade
- Limitação no número de réplicas por GPU
- Restrições de volume persistente
- Dependência de recursos de hardware específicos

Oportunidades de Melhoria

Para trabalhos futuros, identificamos várias oportunidades de evolução:

1. Técnicas
- Implementação de monitoramento avançado
- Otimização do uso de recursos
- Melhoria na resiliência do sistema

2. Segurança
- Implementação de network policies
- Configuração de RBAC
- Hardening dos containers

3. Operacionais
- Automação de backups
- Implementação de CI/CD
- Melhoria na observabilidade

Contribuições

Este trabalho contribui para o campo de estudo através de:

1. Prática
- Demonstração de implementação real de GPU sharing
- Exemplo de integração entre diferentes tecnologias
- Modelo de análise de riscos aplicada

2. Metodológica
- Abordagem estruturada para análise de riscos
- Framework para implementação de sistemas similares
- Documentação detalhada do processo

Considerações Finais

A implementação demonstrou ser viável e eficiente, cumprindo seus objetivos iniciais de criar um ambiente containerizado para execução de modelos de IA com compartilhamento de recursos de GPU. A análise de riscos realizada forneceu insights valiosos sobre os aspectos críticos da operação e segurança do sistema.

Embora existam limitações e oportunidades de melhoria, o trabalho estabelece uma base sólida para desenvolvimentos futuros e serve como referência para implementações similares. A experiência adquirida e as lições aprendidas durante o desenvolvimento contribuem significativamente para o conhecimento prático na área de containerização de aplicações de IA e gerenciamento de recursos de GPU em ambientes Kubernetes.

O projeto demonstra que é possível criar um ambiente eficiente para execução de modelos de IA em containers, mesmo com recursos limitados, através do uso adequado de tecnologias de orquestração e compartilhamento de recursos.